

## 4. MicroServices 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 nagybetűvel.

### 1 A feladat leírása

A feladat egy mozifilm stream-elő szolgáltatás elkészítése, és ennek mikroszolgáltatásként való publikálása.

### 2 A back-end szolgáltatások

Összesen két back-end szolgáltatást kell készíteni, amelyek egyforma interfésszel rendelkeznek. A szolgáltatásokat Spring Boot Maven vagy ASP.NET Core alkalmazásban kell megvalósítani. A két technológia közül szabadon lehet választani.

Az egyik szolgáltatást egy **MoviePi\_NEPTUN** nevű projektben, a másikat pedig egy **MovieE\_NEPTUN** nevű projektben kell megvalósítani. Az első szolgáltatást egy **movie-pi-NEPTUN** nevű Docker konténeren belül, a másikat pedig egy **movie-e-NEPTUN** nevű Docker konténeren belül kell publikálni.

#### 2.1 Maven választása esetén

A szolgáltatást egy ugyanolyan Spring Boot Maven alkalmazásban kell megvalósítani, mint amilyen a tutorialban szerepel. A **pom.xml**-nek kötelezően a csatolt **SingleMovie/pom.xml**-nek kell lennie, de a projekt nevét le kell cserélni benne a megfelelő alkalmazásnévre.

A gRPC interfészt leíró proto fájlt az **src/main/proto** könyvtárban kell elhelyezni.

A forrásfájlok készülhetnek Java vagy Kotlin nyelven is.

A **movie-pi-NEPTUN** nevű Docker konténerből az alábbi URL-en lehessen elérni az első szolgáltatást:

**http://movie-pi-NEPTUN:80**

A **movie-e-NEPTUN** nevű Docker konténerből az alábbi URL-en lehessen elérni a második szolgáltatást:

**http://movie-e-NEPTUN:80**

#### 2.2 .NET Core választása esetén

A szolgáltatást egy ugyanolyan ASP.NET Core alkalmazásban kell megvalósítani, mint amilyen a tutorialban szerepel. A projektfájlnak kötelezően a csatolt **SingleMovie/MovieX\_NEPTUN.csproj**-nak kell lennie, de a fájl nevét le kell cserélni a megfelelő alkalmazásnévre.

A gRPC interfészt leíró proto fájlt a **Protos** könyvtárban kell elhelyezni.

A **movie-pi-NEPTUN** nevű Docker konténerből az alábbi URL-en lehessen elérni az első szolgáltatást:

**https://movie-pi-NEPTUN:443**

A **movie-e-NEPTUN** nevű Docker konténerből az alábbi URL-en lehessen elérni a második szolgáltatást:

**https://movie-e-NEPTUN:443**

## 2.3 Működés

Mindkét szolgáltatásnak gRPC-vel kell kommunikálnia a külvilággal. Mindkét szolgáltatás interfésze a következő (ld. a csatolt **protos/single-movie.proto** fájlt):

```
syntax = "proto3";

option csharp_namespace = "Streaming";
option java_package = "streaming";

package Streaming;

service SingleMovie {
  rpc GetTitle (GetTitleRequest) returns (GetTitleReply);
  rpc GetLength (GetLengthRequest) returns (GetLengthReply);
  rpc GetFrames (GetFramesRequest) returns (GetFramesReply);
}

message GetTitleRequest {
}

message GetTitleReply {
  string title = 1;
}

message GetLengthRequest {
}

message GetLengthReply {
  int32 length = 1;
}

message GetFramesRequest {
}

message GetFramesReply {
  repeated int32 frame = 1;
}
```

Egy **SingleMovie** interfészű szolgáltatás három függvénnyel rendelkezik. A **GetTitle** függvény visszaadja a film címét. A **GetLength** függvény visszaadja a film hosszát, vagyis a képkockák számát. A **GetFrames** függvény visszaadja a film összes filmkockáját.

A **MoviePi\_NEPTUN** nevű projektben a függvények visszatérési értékei a következők legyenek:

- **GetTitle:** „pi” (idézőjelek nélkül)
- **GetLength:** 1000
- **GetFrames:** a pi első 1000 számjegye, beleértve a tizedesvessző előtti számjegyeket is

A **MovieE\_NEPTUN** nevű projektben a függvények visszatérési értékei a következők legyenek:

- **GetTitle:** „e” (idézőjelek nélkül)
- **GetLength:** 500
- **GetFrames:** az Euler-féle szám első 500 számjegye, beleértve a tizedesvessző előtti számjegyeket is

### 3 A front-end szolgáltatás

A szolgáltatást egy Spring Boot Maven vagy egy ASP.NET Core webalkalmazásban kell megvalósítani. A két technológia közül szabadon lehet választani. A szolgáltatást egy **MovieStream\_NEPTUN** nevű projektben kell megvalósítani, és egy **movie-stream-NEPTUN** nevű Docker konténeren belül kell publikálni.

#### 3.1 Maven választása esetén

A szolgáltatást egy ugyanolyan Spring Boot Maven alkalmazásban kell megvalósítani, mint amilyen a tutorialban szerepel. A **pom.xml**-nek kötelezően a csatolt **MovieStream/pom.xml**-nek kell lennie, de a NEPTUN szót le kell cserélni benne a saját Neptun-kódra.

A gRPC interfészt leíró proto fájlokat az **src/main/proto** könyvtárban kell elhelyezni.

A forrásfájlok készülhetnek Java vagy Kotlin nyelven is.

A Docker konténerből az alábbi URL-en lehessen elérni a szolgáltatást:

**http://movie-stream-NEPTUN:80**

Docker-en kívülről, a host gépen az alábbi URL-en lehessen elérni a szolgáltatást:

**http://localhost:5000**

A lehetséges back-end szolgáltatásokat az **application.yml** fájlból kell felolvasni, ahol a következőképpen felsorolva szerepelnek a back-end szolgáltatások URL-jei:

```
grpc:
  client:
    movie-1:
      address: 'dns:///movie-pi-NEPTUN'
    movie-2:
      address: 'dns:///movie-e-NEPTUN'
```

A fenti konfiguráció csak egy példa, a tényleges konfigurációban más URL-ek is szerepelhetnek. Összesen maximum 10 db URL lehet felsorolva, a nevek **movie-1**, **movie-2**, ..., **movie-10** lehetnek.

### 3.2 .NET Core választása esetén

A szolgáltatást egy ugyanolyan ASP.NET Core alkalmazásban kell megvalósítani, mint amilyen a tutorialban szerepel. A projektfájlnak kötelezően a csatolt **MovieStream/MovieStream\_NEPTUN.csproj**-nek kell lennie, de a fájl nevében a NEPTUN szót le kell cserélni a saját Neptun-kódra.

A gRPC interfészt leíró proto fájlokat a **Protos** könyvtárban kell elhelyezni.

A Docker konténerből az alábbi URL-en lehessen elérni a szolgáltatást:

**https://movie-stream-NEPTUN:443**

Docker-en kívülről, a host gépen az alábbi URL-en lehessen elérni a szolgáltatást:

**https://localhost:5000**

A lehetséges back-end szolgáltatásokat az **appsettings.json** fájlból kell felolvasni, ahol egy tömbben felsorolva szerepelnek a back-end szolgáltatások URL-jei a következőképpen:

```
{
  "SingleMovies": {
    "Url": [ "https://movie-pi-NEPTUN", "https://movie-e-NEPTUN" ]
  }
}
```

A fenti konfiguráció csak egy példa, a tömbben más URL-ek is szerepelhetnek, de maximum 10 db URL van felsorolva.

### 3.3 Működés

A szolgáltatásnak gRPC-vel kell kommunikálnia a külvilággal. A szolgáltatás interfésze a következő (ld. a csatolt **protos/movie-stream.proto** fájlt):

```
syntax = "proto3";

option csharp_namespace = "Streaming";
option java_package = "streaming";

package Streaming;

service MovieStream {
  rpc GetMovies (GetMoviesRequest) returns (GetMoviesReply);
  rpc Watch (WatchRequest) returns (stream WatchReply);
}

message GetMoviesRequest {
}
```

```

message GetMoviesReply {
    repeated MovieInfo movies = 1;
}

message MovieInfo {
    string title = 1;
    int32 length = 2;
}

message WatchRequest {
    string title = 1;
    int32 startPosition = 2;
}

message WatchReply {
    int32 frame = 1;
}

```

A **MovieStream** szolgáltatás két függvénnyel rendelkezik. A **GetMovies** függvény visszaadja az elérhető filmek címeit és hosszát. A **Watch** függvény paraméterként megkapja annak a filmnek a címét, amelyiket meg szeretnénk nézni, valamint a kezdőpozíciót, ahonnan a stream-elést el kell kezdeni. A kezdőpozíciónak 0 és a film hossza közé kell esnie. A 0-ás pozíció jelenti a film elejét. A **Watch** függvény válaszában egy-egy képkockát küld vissza a kezdőpozíciótól kezdve egészen a film végéig. A képkockák most egész számok lesznek.

## 4 Beadandók

Beadandó egyetlen ZIP fájl. Más tömörítési eljárás használata tilos!

A ZIP fájl gyökerében három könyvtárnak és egy fájlnek kell lennie:

- **MovieStream\_NEPTUN:** a front-end szolgáltatás Maven vagy .NET Core projektje teljes egészében, benne a megfelelő Dockerfile-lal
- **MoviePi\_NEPTUN:** a „pi” film back-end szolgáltatás Maven vagy .NET Core projektje teljes egészében, benne a megfelelő Dockerfile-lal
- **MovieE\_NEPTUN:** az „e” film back-end szolgáltatás Maven vagy .NET Core projektje teljes egészében, benne a megfelelő Dockerfile-lal
- **docker-compose.yml:** a három projektből egy-egy Docker konténert indító docker-compose konfiguráció

A szolgáltatásnak a fent specifikált gRPC interfészeket kell támogatnia. Más formátum/elnevezés használata tilos!

A megoldásnak a telepítési leírásban meghatározott környezetben kell fordulnia és futnia, a **pom.xml**-ek illetve a **csproj** fájlok tartalmát illetve nevét a projektnév módosításán felül megváltoztatni 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 nagybetűvel!

#### 4.1 Maven választása esetén

Mielőtt a projektet becsomagolnánk a ZIP-be, a projekteken belül adjuk ki az **mvn clean** parancsot! Ez a parancs törli a **target** alkönyvtárat, amely elég nagy is lehet és egyébként sem lesz figyelembe véve a kiértékelés során.

A ZIP kibontása után mindhárom projektnek fordulnia és futnia kell Docker-ben a következő parancs kiadásával:

```
docker-compose up --build
```

#### 4.2 .NET Core választása esetén

Mielőtt a projektet becsomagolnánk a ZIP-be, a projekteken belül adjuk ki a **dotnet clean** parancsot! Ez a parancs törli a **bin** és **obj** alkönyvtárakat, amelyek elég nagyok is lehetnek és egyébként sem lesznek figyelembe véve a kiértékelés során.

A ZIP kibontása után mindhárom projektnek fordulnia és futnia kell Docker-ben a következő parancs kiadásával:

```
docker-compose up --build
```