JUMO™ - JukkaMovies service backend/frontend

Kehitysympäristön vaatimukset

- NodeJS runtime ja NPM / Yarn -paketinhallintasovellukset (suositaan Yarnia)
- Serverless komentotulkkityökalut (asennus: `npm install -g serverless`)
- Java runtime >= 8 (DynamoDB:n paikalliseen pyörittämiseen)

Deployment:

- AWS -tili ja kredentiaalit
- AWS komentotulkkityökalut
- (Ba)SH -komentotulkki POSIX -tyylin ympäristössä, kuten esim. GNU/Linux, macOS/BSD, jne. (ei täysin välttämätön, mutta helpottaa elämää ihanasti)

Backend

Toiminta

Backend on luotu ns. microservice -arkitehtuurin mukaisesti Serverlessiä hyödyntäen AWS Lambda -funktioksi, joka hallinnoi luomaansa DynamoDB -tietokantaa.

Asennus

```
cd api && yarn install && sls dynamodb install
```

Ajo

```
sls offline start
```

Backend käynnistyy (ja käynnistää samalla paikallisen DynamoDB -instanssin) osoitteeseen http://localhost:3000. Näinollen API -kutstuista muodostuu endpoint: http://localhost:3000/dev/movies.

Backend

Asennus

```
cd ui && yarn install
```

Ajo

```
yarn start
```

UI käynnistyy

JUMO™ API-kuvaus: `http://localhost/dev/movies`)

Polku	Metodi	JSON -rakenne	Selitys
/movies	GET	N/A	Hae kaikki tallennetut tietueet listana allaolevan JSON-rakenteen mukaisesti

/movies	POST	{
		  name: string,
		  year: integer,
		  ageLimit: integer,
		  rating: integer,
		  genres: [string, ...],
		  actors: [
		  {
		  firstName: string,
		  lastnName: string
		  } ...
		  ],
		  synopsis: string,
		  director: {
		  firstName: string,
		  lastName: string
		  }
		}
			Tallenna elokuvatieto. Backend vastaa samalla mitalla takaisin sillä erotuksella, että nyt tiedossa on mukana ID (uuid) -kenttä.

/movies/batch	POST	[
		  {
		  name: string,
		  year: integer,
		  ...
		  }, ...
		]
			Dev/admin -ympäristöön tarkoitettu useamman tiedon tallentamista samanaikaisesti. Vastauksena samanlainen listaus ID-tunnisteiden kera.

/movies/{id}	GET	N/A	Hae elokuvatieto ID:llä
/movies/{id}	DELETE	N/A	Poista elokuvatieto ID:llä