



SEMINARSKA NALOGA - SPLOŠNA MATURA

BAZA PODATKOV ZA VODENJE VIDEOTEKE

****Škofljica, 29.5.2020 Avtor: Klemen Remec****

****Gimnazija Vič, Tržaška c. 72, 1000 Ljubljana Mentor: prof. Klemen Bajec****

# Povzetek

Za lažje iskanje filmov po izbranih kategorijah sem napisal program, ki vse filme razvrsti po kategorijah: letnica izida, žanr in glavni igralci. Vse podatke o filmu dobi preko API klica, nato pa jih shrani v dokumentno bazo za kasnejšo uporabo in možne ročne popravke. Seveda sem zato moral raziskati postopke dela z API klici, podatkovno bazo MongoDB ter spoznati pisanja ukazov Linux serverjem v Shell scriptu.

## Ključne besede

Filmi, API, podatkovna baza, skripta, JSON.

Vsebina

[Povzetek 2](#_Toc72423256)

[Ključne besede 2](#_Toc72423257)

[Uvod 2](#_Toc72423258)

[Namen in cilji 2](#_Toc72423259)

[Teoretična podlaga 2](#_Toc72423260)

[Programski vmesnik (API) 2](#_Toc72423261)

[Shell script 2](#_Toc72423262)

[JSON - JavaScript Object Notation 2](#_Toc72423263)

[MongoDB 2](#_Toc72423264)

[Praktični del 2](#_Toc72423265)

[Posodobitev baze z novo dodanimi filmi 2](#_Toc72423266)

# Slike

[Slika 1: OMDb API spletna stran (vir: https://www.omdbapi.com/). 2](#_Toc99988254)

[Slika 2: Datotečna zbirka v podatkovni bazi znotraj programa MongoDB Compass. 2](#_Toc99988255)

[Slika 3: Funkcije, s katerimi lahko program dostopa do in dela z MongoDB bazo. 2](#_Toc99988256)

[Slika 4: Konfiguracijska datoteka, v kateri so zapisane poti do potrebnih direktorijev. 2](#_Toc99988257)

[Slika 5: Objekt "Movie", v katerem se shranjujejo podatki o filmu, s katerimi nato delam naprej. 2](#_Toc99988258)

[Slika 6: API klic in shranjevanje podatkov o filmu iz odgovora. 2](#_Toc99988259)

[Slika 7: Pisanje ukazov v shell scriptu za sortiranje po letnicah. 2](#_Toc99988260)

[Slika 8: Pisanje ukazov v shell scriptu za sortiranje po žanrih, določenih v vnaprej napisanem seznamu. 2](#_Toc99988261)

[Slika 9: Pisanje ukazov v shell scriptu za sortiranje po igralcih, določenih v datoteki z imeni igralcev, ki nas zanimajo. 2](#_Toc99988262)

[Slika 10: Izbris in ponovna vzpostavitev datotečnega sistema sortiranih filmov. 2](#_Toc99988263)

# Uvod

Dandanes vsak izmed nas gleda filme, velika večina preko televizijskih programov, zmeraj več pa tudi preko t.i. pretočnih storitev ("streaming services"), kot so Netflix, Disney Plus in HBO Max. Slednji imajo to prednost, da lahko sam izbereš, kaj boš gledal, in to kadarkoli, kar je izjemno prijazno uporabnikom teh storitev. (nadlani, 2021)  
Z domačimi videotekami pa ni zmeraj tako - filmov je veliko in niso urejeni. Glede na to, da se je zamudno sprehajati po stotinah filmov, sem napisal program, ki najde novo naložene filme, najde želene podrobnosti o filmu, vse zapiše v bazo, iz katere lahko vodimo evidenco, in napiše skripto, ki nam filme uredi po želenih kategorijah.

# Namen in cilji

Cilj je narediti program za izdelavo domače videoteke, ki pregleda sistem datotek v računalniku, nato pa pogleda, ali je film že vnesen v bazo, kjer hranimo vse podatke o filmu (letnico nastanka, igralce, žanre, itd.). Če še ni, preko programskega vmesnika (API) IMDB baze poizve za vse potrebne podatke o filmu, in jih vnese v bazo. Program nato sestavi t.i. shell script, ki ustvari urejen datotečni sistem, razvrščen po kategorijah, kot so nastopajoči igralci, žanri in leto nastanka. Tako dobimo urejeno videoteko, po kateri se je veliko lažje znajti.

# Teoretična podlaga

## Spletna zahteva (Web request)

Sporočilo oz. zahteva, ki ga odjemalec (npr. spletni brskalnik) pošlje strežniku, ki odjemalcu ponudi oz. pošlje nazaj zahtevane podatke. To je bistvenega pomena, saj tako uporabniku zagotovimo želene podatke (npr. spletne strani, spletne datoteke, itd.). Strežnik najprej obdela zahtevo, nato pa zahtevano vsebino pošlje nazaj kot odgovor, ki ga bo uporabniški vmesnik prikazal kot spletno stran, sliko ali interaktivno funkcijo. (Source Defense, 2022)  
Najbolj pogosti zahtevi sta:

* "GET" je ena najpogostejših spletnih zahtev, ki jih je mogoče narediti. Ta vrsta zahteve se izvede v imenu odjemalca, ko išče podatke iz določenega vira. Ko odjemalec vloži zahtevo za vir, bo strežnik obdelal zahtevo, našel informacije ali podatke in jih poslal nazaj odjemalcu.
* "POST" je še ena pogosta vrsta spletne zahteve, vendar je namen POST ustvariti ali posodobiti zahtevane vire. Ne shranjuje dodatnih podatkov, temveč je treba vsako zahtevo posodobiti in osvežiti ali ponovno poslati.

(Source Defense, 2022)

## Programski vmesnik (API - "Application Programming Interface")

Niz protokolov in orodij za gradnjo programskih aplikacij. (iprom, 2021) V nasprotju z uporabniškim vmesnikom, ki povezuje računalnik z uporabnikom, programski vmesnik povezuje računalnike ali kose programske opreme med seboj. Kot tak tako ni namenjen neposredni uporabi osebe (končnega uporabnika), razen računalniškega programerja, ki ga vgradi v programsko opremo; določa, kako je treba sestavne dele različne programske opreme uporabiti in kako naj sodelujejo in komunicirajo med sabo. API-ji so nastali v 40. letih prejšnjega stoletja, čeprav se je izraz pojavil šele v 60. in 70. letih prejšnjega stoletja. Izraz API se pogosto uporablja za spletne API-je, ki omogočajo komunikacijo med računalniki preko interneta. V tem primeru je API običajno opredeljen kot niz podatkov iz HTTP poizvedbe ("Hypertext Transfer Protocol request messages"), skupaj z nizom odgovorov, ki ga API pošlje nazaj računalniku. Odgovor je običajno zapisan v formatu zapisovanja podatkov, kot sta npr. XML in JSON. (omdbapi, 2021)

## JSON - JavaScript Object Notation

Oblika standardiziranega zapisa in serializacije (proces pretvorbe podatkov v obliko, primernejšo za shranjevanje ali prenos v pomnilnik, bazo podatkov ali datoteko) podatkov, ki omogoča efektivno shranjevanje in izmenjavo podatkov. JSON je izpeljan iz JavaScripta, a je kot format zapisa podatkov neodvisen od programskega jezika - načeloma je narejen za programske jezike, podobne C-ju, kot so C, C++, C#, Java, Python itd. Zaradi tega je uporaben pri izmenjavanju podatkov med različnimi platformami. Imena datotek JSON uporabljajo pripono .json. (json, 2021)

## Shell script

Računalniški program, narejen za delovanje v Unix shellu. Sam program naj bi vseboval verigo ukazov, ki se ob zagonu programa izvedejo po vrsti. Tipične operacije, ki jih izvaja, vključujejo delo z datotekami, izvajanje programov in tiskanje besedila. Najstarejši tip Shell scripta je "Bourne shell" s pripono .sh. Shell script se začne z vrstico, ki jo imenujemo "shebang" ali "hash-bang" ("#!/bin/sh"), s čimer sistemu povemo, kateri tolmač se naj uporabi za izvajanje skripte. (HPCWIKI, 2021) (Wikipedia, 2021)

## Simbolični linki v Linuxu

Delujejo na principu "shortcuta" oz. bližnjice, saj lahko kažejo na katerokoli datoteko ali direktorij, do katerega računalnik lahko dostopa - tudi na drugih diskih. V Linuxu uporabimo ukaz "ln -s" ("-s" pomeni "soft/symbolic link"), kateremu sledi pot do datoteke, ki jo želimo povezati in pot do direktorija, v katerem bo novonastali link. (freeCodeCamp, 2021)

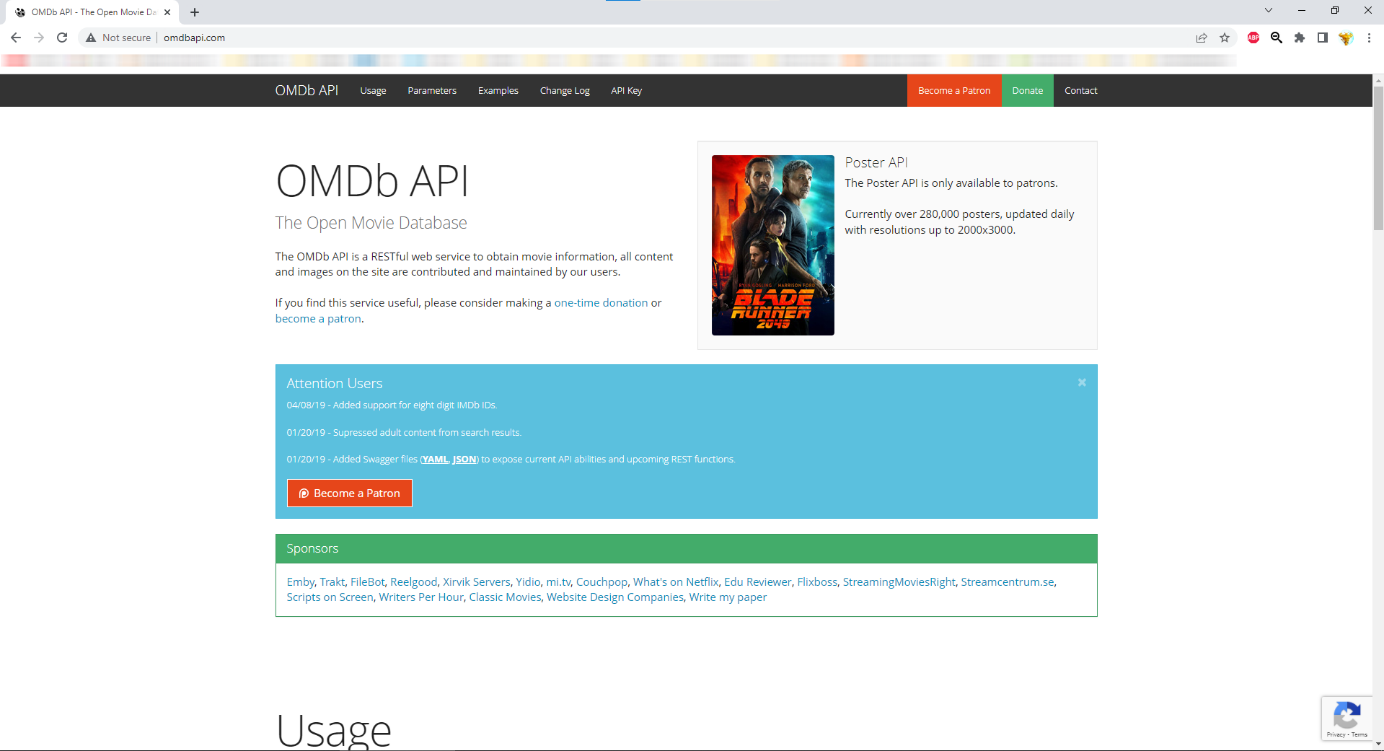
## MongoDB

Moderna dokumentna baza, ki kot klasificirana NoSQL baza podatke shranjuje v "dokumentih" v JSON obliki. MongoDB poleg enostavnega vodenja baze ponuja tudi filtriranje in iskanje po bazi. Znotraj baze je lahko shranjenih več zbirk ("collection"), v katerih so shranjeni dokumenti. Vsak dokument ima poleg glavnega ključa ("primary id", 12-bajtni šestnajstiški ključ) tudi polja, v katere se podatki shranijo. Do baze se lahko dostopa s programom MongoDB Compass - program za lažjo navigacijo in ročno urejanje baze, ali s programi in programskimi API-ji, ki omogočajo programu dostop in upravljanje z bazo. (MongoDB, 2022) (MongoDB Docs, 2022) (Tutorialspoint, 2022) (Wikipedia, 2021) (Wikipedia, 2022)

# Praktični del

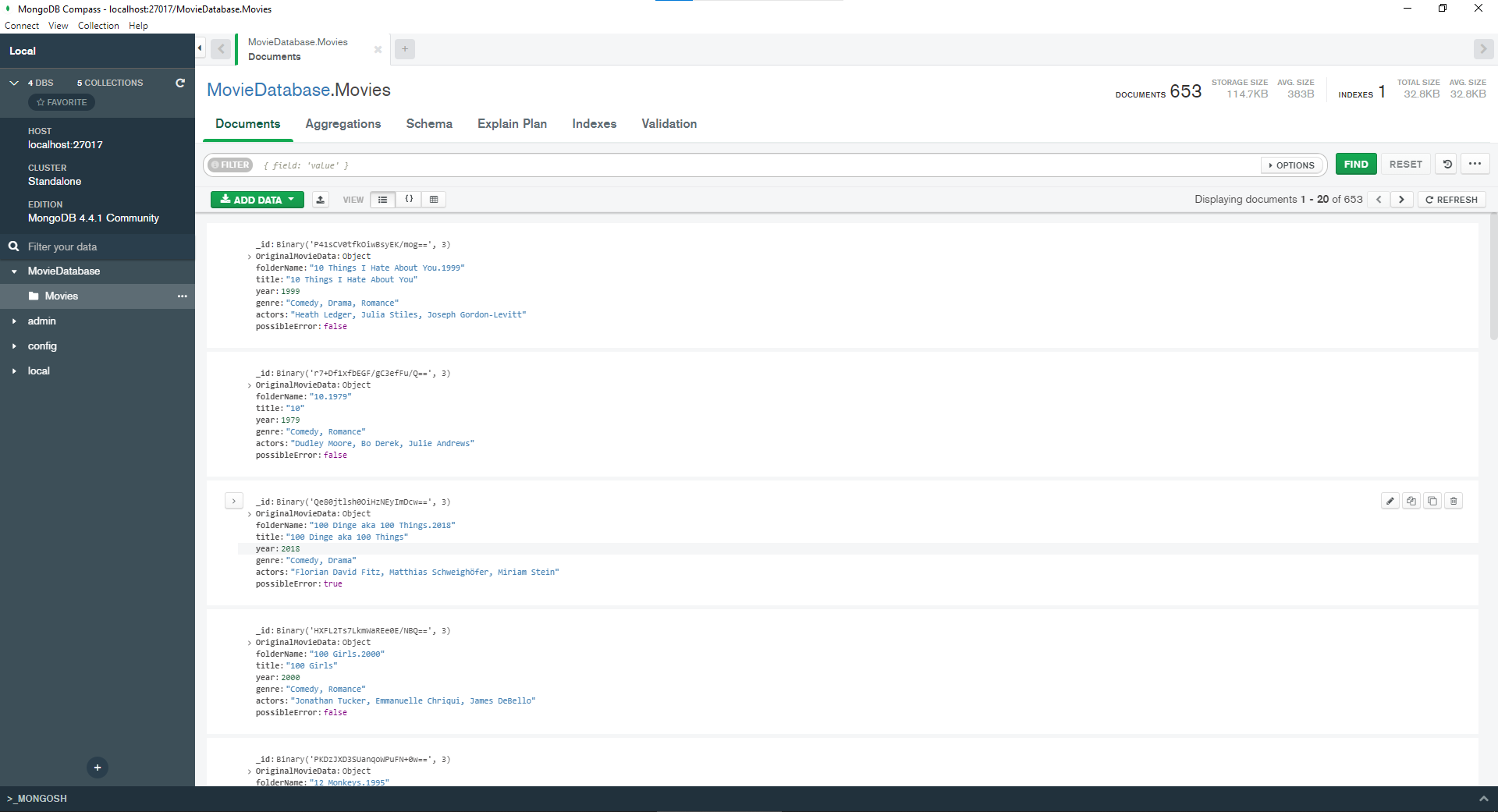
## Priprava okolja

Najprej sem moral najti ustrezen API, iz katerega bi program lahko pridobil podatke o posameznem filmu. Odločil sem se za OMDb API, spletno storitev za pridobivanje informacij o filmih. Da sem ga lahko začel uporabljati, sem moral dobiti API ključ, ki sem ga uporabljal v spletnih zahtevah.



Slika 1: OMDb API spletna stran (vir: https://www.omdbapi.com/).

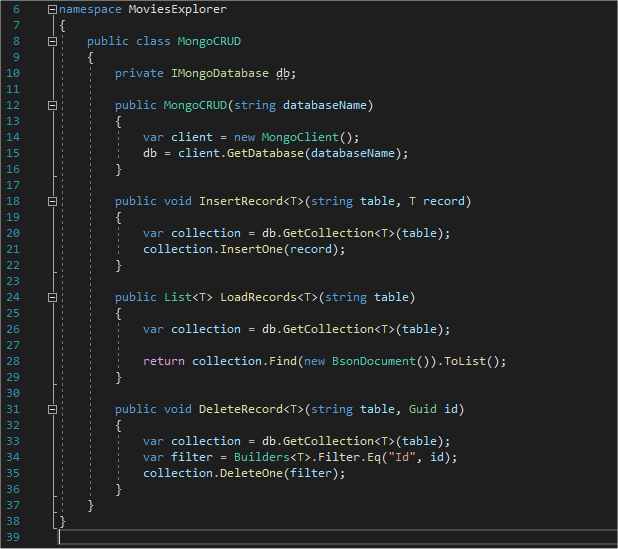
Moral sem tudi vzpostaviti bazo za shranjevanje informacij o filmih. Odločil sem se uporabiti MongoDB bazo. Iz uradne spletne strani sem naložil lokalni server ter MongoDB Compass. Ko sem se povezal na bazo, sem ustvaril novo podatkovno bazo in zbirko za dokumente.



Slika 2: Datotečna zbirka v podatkovni bazi znotraj programa MongoDB Compass.

Program sem napisal v jeziku C#, ki sem ga uporabljal do zdaj tudi pri drugih projektih, zato sem lahko začel hitro in efektivno delati.

Najprej sem moral spisati splošno programsko kodo - osnovne operacije za delo s podatkovno bazo ("CRUD operations"): C-Create (ustvarjanje datotek/zapis podatkov), R-Read (branje datotek iz baze), U-Update (spreminjanje obstoječih dokumentov/podatkov v bazi) in D-Delete (brisanje dokumentov/podatkov iz baze).



Slika 3: Funkcije, s katerimi lahko program dostopa do in dela z MongoDB bazo.

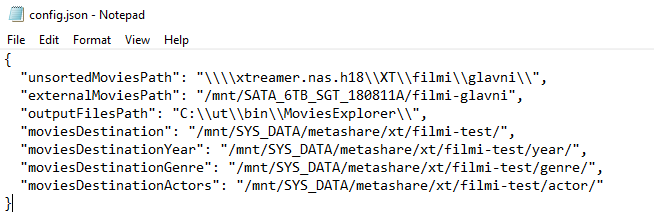
Sam program ima 2 funkcionalnosti:

1. Posodobitev baze z novo dodanimi filmi in
2. izpis shell scripte, ki ustvari urejen datotečni sistem, razvrščen po kategorijah, kot so nastopajoči igralci, žanri in leto nastanka filmov.

## Posodobitev baze z novo dodanimi filmi

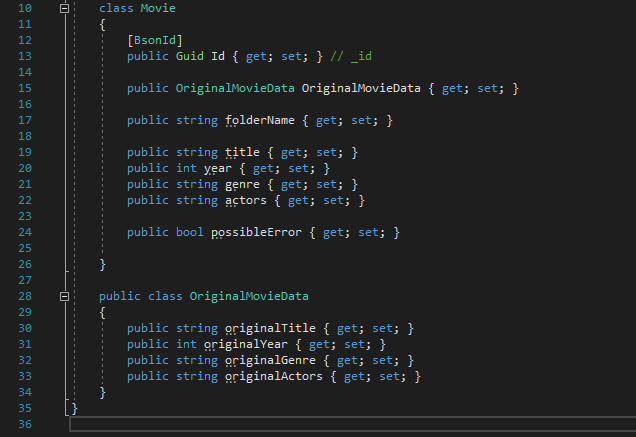
Filme imamo urejene po direktorijih, vsak film v svojem direktoriju, vsak direktorij pa je naslovljen po imenu filma in letnici izida, med katerima je pika (npr. "Top Gun.1986").

Program mora najprej rekurzivno najti vse direktorije s filmi iz določenega začetnega direktorija in izluščiti imena direktorijev, kjer so imena in letnice že usklajeno urejene. Pot do začetnega direktorija prebere iz konfiguracijske datoteke. Iz nje program bere določene nastavitve in preference, kot so poti do datotek in direktorijev, s katerimi dela. Tako se v primeru, da spremenimo strukturo domačega datotečnega sistema (npr. prestavimo filme v drug direktorij ali celo drug disk) izognemo popravljanju poti do teh lokacij znotraj kode, ampak to spremenimo v konfiguracijski datoteki.



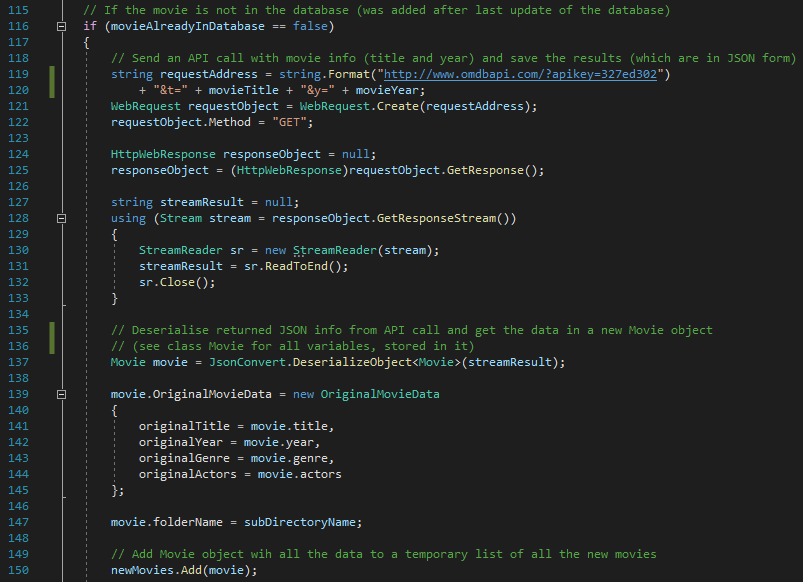
Slika 4: Konfiguracijska datoteka, v kateri so zapisane poti do potrebnih direktorijev.

Vsako prebrano ime direktorija s filmom program zapiše v objekt Movie, ki kasneje shrani dobljene podatke o filmu za prenos v bazo. Vsak dobljen string nato preoblikuje, pripravi in shrani v seznamu za kasnejše iskanje po APIju.



Slika 5: Objekt "Movie", v katerem se shranjujejo podatki o filmu, s katerimi nato delam naprej.

Nato program preveri za vsak film v seznamu, ali je že shranjen v bazi. Če še ni, pošlje web request z naslovom in letom filma na API (spletno stran), in odgovor shrani. Ker je odgovor, ki ga program dobi kot odgovor, string, napisan v formatu JSON, mora to najprej deserializirati v nov objekt. Objekt nato dopolni z ostalimi podatki, preden ga doda v bazo.  
Če se ime filma ne sklada z odgovorom APIja, program izpiše opozorilo, da je morda prišlo do napake pri pridobivanju informacij o filmu, ki jo kasneje lahko ročno popravimo.



Slika 6: API klic in shranjevanje podatkov o filmu iz odgovora.

Tako imamo na koncu seznam oz. "list" objektov s podatki o vseh filmih, ki smo jih našli v izbranih direktorijih.

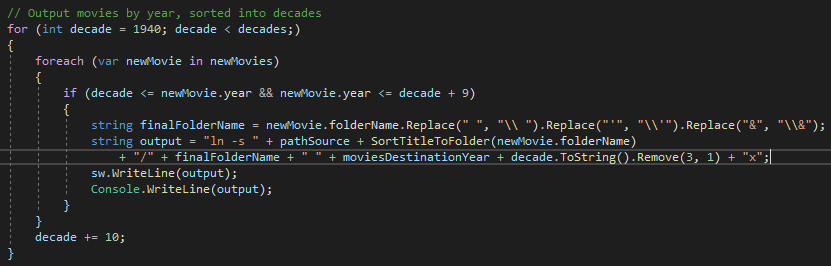
## Izpis shell skripte

Drugi del programa ustvari shell script dokument v direktoriju, določenem v konfiguracijski datoteki. Prav tako imam ločeno datoteko z imeni igralcev, ki me zanimajo za nadaljnje iskanje filmov.

Skripta najprej zamenja delovni direktorij na izbran direktorij, kjer bomo imeli razvrščeno videoteko. Delovni direktorij sprazni (če smo že kdaj prej zagnali skripto in ustvarili videoteko, se tako izognemo mešanju starih in novih seznamov). Nato ustvari direktorije za vse kategorije, po katerih smo filme iz baze razvrščali. V tem primeru so kategorije sledeče:

1. Letnica izida filma.

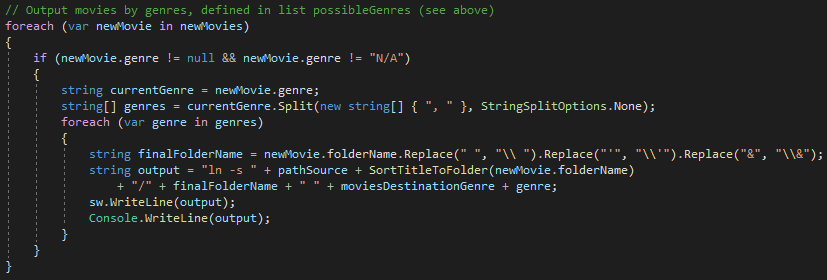
Skripta ustvari direktorije za vsako desetletje. Nato se program sprehodi po datotekah v bazi in za vsakega v skripto zapiše primeren ukaz, ki ustvari simboličen link do filma v direktoriju z ustreznim desetletjem.



Slika 7: Pisanje ukazov v shell scriptu za sortiranje po letnicah.

1. Žanr filma.

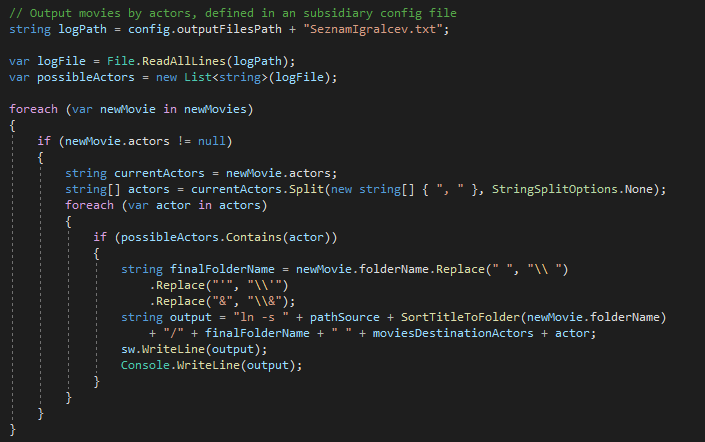
Skripta ustvari direktorije za vsako vnaprej določen žanr. Nato se program sprehodi po datotekah v bazi in za vsakega v skripto zapiše primeren ukaz, ki ustvari simboličen link do filma v direktoriju z ustreznim žanrom.



Slika 8: Pisanje ukazov v shell scriptu za sortiranje po žanrih, določenih v vnaprej napisanem seznamu.

1. Glavni igralci v filmu.

Skripta ustvari direktorije za vse vnaprej določene filmske igralce. Nato se program sprehodi po datotekah v bazi in za vsakega v skripto zapiše primeren ukaz, ki ustvari simboličen link do filma v direktoriju z ustreznim igralcem.



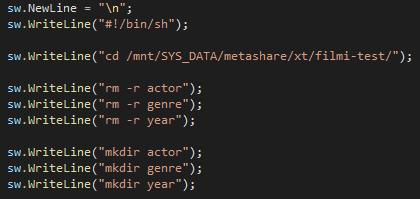
Slika 9: Pisanje ukazov v shell scriptu za sortiranje po igralcih, določenih v datoteki z imeni igralcev, ki nas zanimajo.

Skripta mora seveda paziti na vsa pravila, kako se morajo določeni simboli zapisati v shell scriptu za Linux (pred vsak presledek, apostrof, znak & moramo dati "\\", saj bi sicer tolmač skripte javil napako).

## Izpis že obstoječih filmov, ki so zapisani v bazi

Program najprej prebere in shrani vse podatke o filmih v seznam objektov, nato pa podobno kot prej (glej [prejšnje poglavje](#_Izpis_shell_skripte)) izpiše shell skripto.

Za razliko od prej program najprej zbriše strukturo sortiranih filmov, ki smo jo ustvarili pred tem, in jo na novo ustvari.



Slika 10: Izbris in ponovna vzpostavitev datotečnega sistema sortiranih filmov.

Funkcija tega drugega načina delovanja programa je, da lahko v primeru, če ročno popravljamo podatke o filmih v programu MongoDB Compass, samo izpišemo skripto za popravljeno sortiranje, in ne trošimo časa in API klicev, potrebnih za ponovno poizvedovanje - od prejšnjič namreč nismo dodali nobenega novega filma.

# Zaključek

Po vsem raziskovalnem in praktičnem delu sem uspešno prišel do zastavljenega cilja. Program dela po pričakovanjih, doma ga uporabljamo več kot pol leta in tako lažje brskamo po domači videoteki.

V prihodnosti mislim dodati še več funkcionalnosti, na primer več kategorij, informacije o filmu (na primer kratka obnova zgodbe, ocena filma na IMDB, RottenTomatoes in drugih spletnih straneh, kjer lahko ljudje filme ocenijo, itd.).

Z izdelavo programa in seminarske naloge sem se naučil marsikaj novega - delo z datotečnimi bazami, API-ji, poglobil sem znanje programiranja, teorije računalništva in informatike nasploh.

# Sklici

*E-računalništvo*. (22. 3 2022). Pridobljeno iz Objektno programiranje: https://gradiva.txt.si/index.php/objektno-programiranje-2/

*freeCodeCamp*. (21. 5 2021). Pridobljeno iz Symlink Tutorial in Linux – How to Create and Remove a Symbolic Link: https://www.freecodecamp.org/news/symlink-tutorial-in-linux-how-to-create-and-remove-a-symbolic-link/

*HPCWIKI*. (19. 5 2021). Pridobljeno iz Sh-file: https://hpc-wiki.info/hpc/Sh-file

*iprom*. (20. 5 2021). Pridobljeno iz Programski vmesnik: https://iprom.si/slovar/api-programski-vmesnik/

*json*. (20. 5 2021). Pridobljeno iz Introducing JSON: https://www.json.org/json-en.html

*mongoDB*. (20. 5 2021). Pridobljeno iz The database for modern applications: https://www.mongodb.com/

*MongoDB*. (22. 3 2022). Pridobljeno iz Why use MongoDB: https://www.mongodb.com/why-use-mongodb

*MongoDB Docs*. (22. 3 2022). Pridobljeno iz CRUD: https://docs.mongodb.com/manual/crud/

*nadlani*. (19. 5 2021). Pridobljeno iz Kaj je Netflix? Cena v Sloveniji, vsebina, kako ga gledati: https://www.nadlani.si/multimedija/netflix/

*omdbapi*. (21. 5 2021). Pridobljeno iz OMDb API: https://www.omdbapi.com/

*Source Defense*. (23. 03 2022). Pridobljeno iz Web request: https://sourcedefense.com/glossary/web-request/

*Tutorialspoint*. (22. 3 2022). Pridobljeno iz MongoDB: https://www.tutorialspoint.com/mongodb/index.htm

*Wikipedia*. (20. 5 2021). Pridobljeno iz Shell script: https://en.wikipedia.org/wiki/Shell\_script

*Wikipedia*. (20. 5 2021). Pridobljeno iz MongoDB Atlas: https://en.wikipedia.org/wiki/MongoDB#MongoDB\_Atlas

*Wikipedia*. (22. 3 2022). Pridobljeno iz MongoDB: https://en.wikipedia.org/wiki/MongoDB