

# 1. Col·lecció i reviews de pel·lícules

---

Anem a crear una aplicació que ens permetrà gestionar una col·lecció de pel·lícules o sèries.

📖 Aquestes dades seran emmagatzemades en una BBDD relacional, que crearem des de zero.

👤 Haurem de fere login en la aplicaicó per a poder entrar

💻 Afegirem un CRUD a aquestes dades recuperades.

💡 Proposates per part de l'alumnat

🎬 Com a complement adicional de la pràctica, les dades seran reals i les aconseguirem de <https://www.filmaffinity.com/es/main.html>, mitjançant una tècnica de *web scrapping*.

## 2. Web Scrapping de filmaffinity (completa pel professor - Informatiu)

---

!!! warning "Atenció"

Aquesta part de la pràctica és opcional. No cal realitzar-la, es proporciona un fitxer amb les dades ja recuperades per a poder continuar amb la resta de la pràctica.

El codi font d'aquesta pràctica es facilita com a curiositat i per a que l'alumnat pugui veure com es realitza un scrapping amb Java i Selenium. No és necessari entendre-ho tot.

Pots descarregar l'arxiu inicial amb les dades de treball a aules o en [aquest enllaç](./films.json).

El web scraping (o "raspat web") és la tècnica d'extreure informació de llocs web de manera automatitzada. En lloc de copiar i enganxar manualment les dades, creem un programa que visita la pàgina i n'extreu el contingut estructurat (DOM).

A continuació processem el DOM i accedint a les etiquetes HTML (XML) recuperem les dades de les mateixes.

### 2.1. JSOUP

És una llibreria Java dissenyada específicament per a aquesta tasca. És, essencialment, un analitzador d'HTML (HTML parser). Les seves funcions principals són:

- **Connectar-se i descarregar:** Pot fer una petició HTTP (com un **GET**) a una URL i rebre el codi font HTML inicial de la pàgina.

- **Analitzar (Parse):** Converteix aquest text HTML en una estructura de dades que podem manipular: el DOM (Document Object Model).
- **Extreure dades:** Ofereix una API molt còmoda (similar a jQuery) per navegar per aquest DOM i seleccionar elements concrets utilitzant selectors CSS (per exemple, `doc.select("div.product-title")`).

La gran dificultat del scrapping és:

- Navegar pel DOM de la web i trobar les etiquetes i classes on resideix la informació
- Vigilar els canvis, ja que qualssevol canvi en la web suposarà que el nostre programa deixi de funcionar

El problema és que JSOUP només veu el que veuria un usuari si fes clic a "Veure codi font" al navegador. **És una eina estàtica.**

## 2.2. El Repte: El JavaScript Dinàmic

En el passat, els servidors web enviaven un fitxer HTML complet. JSOUP era perfecte per a això. Hui en dia, la majoria de llocs web moderns (especialment les Single Page Applications - SPA, fetes amb React, Angular o Vue) funcionen de manera diferent:

- El servidor envia un fitxer HTML quasi buit (un "shell" o esquelet).
- Aquest HTML conté enllaços a grans fitxers JavaScript (.js).
- És el navegador de l'usuari qui ha d'executar aquest JavaScript.
- Aquest script, un cop executat, fa peticions addicionals (AJAX/Fetch) a una API per obtenir les dades reals (sovint en format JSON).
- Finalment, el JavaScript construeix dinàmicament el contingut de la pàgina (les taules, els productes, els comentaris) i l'insereix al DOM.

JSOUP no pot fer res d'això. JSOUP no és un navegador. No té un motor de JavaScript. Quan JSOUP descarrega la pàgina, només obté l'HTML buit del pas 1. Totes les dades que vols "raspar" simplement no existeixen en el codi font inicial que rep.

## 2.3. La Solució: Per què Selenium és més adequat?

Selenium no és un analitzador d'HTML; és una eina d'automatització de navegadors. No només descarrega una pàgina, sinó que controla un navegador web real (com Chrome, Firefox o Edge, sovint en mode "headless" o sense interfície gràfica). Això ens permet manipular el navegador des de codi.

El procés amb Selenium és el següent:

- **Automatització:** Selenium obri una instància real del navegador.
- **Navegació:** Li diu al navegador: "Visita aquesta URL".
- **Execució Completa:** El navegador, en ser un navegador real, descarrega l'HTML, executa tot el JavaScript, fa les crides AJAX i construeix el DOM final, exactament com ho faria un usuari humà.
- **Espera:** Podem dir-li a Selenium que "espere" fins que un element específic (que sabem que es carrega amb JavaScript) aparegui a la pàgina.
- **Extracció:** Un cop la pàgina està completament renderitzada, Selenium ens dona accés a aquest DOM final i complet. Si volem ara ja el podem processar amb JSOUP.

## 2.4. Classe per guardar la informació.

Haurem de definir una classe inicial per guardar la informació al nostre programa. Aquesta serà la següent:

```
public class FilmAffinity implements Serializable {  
  
    private static double serialVersionUID = 17L;  
  
    private String id;  
    private String titulo;  
    private int anyo;  
    private String pais;  
    private ArrayList<String> actores;  
    private ArrayList<String> generos;  
    private String sinopsis;  
    private float nota;  
}
```

Farem servir el mateix objecte, ja siga per a sèries, pel·lícules i o documentals.

!!! question "Implementació de la classe" Implementa la classe anterior de manera bàsica (constructors, getters, setters, etc). Confome avance la pràctica necessitaràs afegir noves funcionalitats

## 3. Validació de l'usuari

Abans d'accedir a qualsevol funcionalitat de l'aplicació, l'usuari haurà d'identificar-se. Revisa l'exercici desenvolupat a classe, on fem servir una base de dades per a emmagatzemar els usuaris i les seues contrasenyes.

### 3.1. Funcionament de l'apartat de registre i login

- Quan arranque el programa quee et pregunte dos coses: login o registre.
  - Cas de respondre login, introduïrem el nom d'usuari:
    - Posteriorment la contrasenya. Intenta fer que no es mostre la contrasenya per pantalla
    - Es realitzarà la validació o no desde la base de dades
- Cas de respondre registre,
  - Introduïrem un nom d'usuari,
  - una contrasenya
  - Repetireem la contrasenya
  - Es realitzarà l'alta a la base de dades.

!!! warning "Atenció"

- Guarda el password encriptat
- La BBDD SQLite que estiga en la arrel del projecte, anomenada `usuaris.db`

## 3.2. Implementació

- No cal implemtar una interfície gràfica. Tot es farà per consola
- La base de dades, a l'igual que l'exercici serà implementada amb SQLite.
- Pots importar tota la lògica d'aquesta part del programa.

## 4. Connexió al servidor de la base de dades

---

### ## Patró Singleton

El patró Singleton és un dels patrons de disseny "creacionals" més coneguts. El seu objectiu és molt simple i específic: **garantir que una classe tinga una única instància en tota l'aplicació i proporcionar un punt d'accés global a aquesta instància.**

Pensa en ell com l'interruptor general de la llum d'una casa. Només n'hi ha un, i des de qualsevol part de la casa, quan parles de "l'interruptor general", et refereixes sempre al mateix.

Per a aconseguir-ho, el patró es basa en tres pilars:

- **Constructor Private:** Per a evitar que qualsevol altra classe pugui crear noves instàncies lliurement amb l'operador `new`.
- **Instància Estàtica Privada:** La mateixa classe és responsable de crear i emmagatzemar la seua única instància.
- **Mètode Estàtic Públic:** Un mètode (normalment anomenat `getInstance()`) que retorna la instància única. Si encara no existeix, la crea la primera vegada que es crida.

### 4.1. Configuració de la connexió

En iniciar el nostre programa i passar la fase de login, demanarà les dades del servidor, que l'usuari les podrà indicar. Haurà d'existir un arxiu anomenat `connexio.props` de manera que si alguna propietat no s'indica de manera explícita (deixant-la en blanc) la carregue de dit fitxer.

```
...
$ Server: 127.0.0.1
$ Port: 3308
$ Username: root
$ Password:
$ (root) on 127.0.0.1:3308>
```

!!! question "Classe Connexió" Es demana:

- Implementa la classe `Connexió` que es comporte com un Singleton
- Fes el mètode `Connection getConnection()` que retorne la única connexió, creant-la la primera vegada que es demane
- Fes un mètode `createConnection()` que ha de complir lo següent:
  - S'ha de cridar només començar el programa
  - Ha de demanar les dades per a connectar-se al servidor a l'usuari

- Si hi ha alguna en blanc la carregarrà del ``connexio.props``
- Crearà un objecte de la classe ``Connexió`` (sense connectar-se encara a la BBDD)
- Retorna ``true`` si ha anat tot be o ``false`` en cas d'algun error

## 5. Aplicació pròpiament dita

Com podeu veure a l'última línia, el prompt és `(usuari) a IP:port>`. El nostre client mostra sempre on estem connectats. El programa deu quedar-se esperant a que l'usuari faci laguna de les opcions indicades. El comando `help` mostrarà les següents opcions, que son a implementar:

### 5.1. Menú

!!! tip "Opcions de menú" 1. ``init`` → crea i inicia la BBDD 2. ``scrap [web]`` → realitza el scrapping de la pàgina indicada 3. ``list_all film|actor|genere [/p]`` → llista tots els films, actors o gèneres 4. ``list_by actor nom_actor|genere_nom_genere asc|desc`` → mostrarà els films d'un actor donat o d'un gènere donat, en l'ordre indicar 5. ``update idFilm`` → actualitza la informació de la pel·lícula indicada. 6. lliure 7. lliure 8. lliure 9. `help` → mostra aquest quadre

Les opcions de menú 6, 7 i 8 les heu de crear vosaltres amb possibles opcions. Són lliures. Podeu fer consultes addicionals, esborrar pel·lícules, etc.

### 5.2. `init`

Crearà la BBDD `filmaffinity`. Ademés crearà la taula per a poder emmagatzemar el film anterior. Els actors i els gèneres guardar-los en una taula apart (sols el id i el nom), amb les corresponents claus alienes. Si la BBDD ja existia deurà informar a l'usuari d'aquest fet i no crear-la de nou.

> Es recomana escriure el script en un fitxer anomenar ``initBBDD.sql`` i executar-lo sobre la base de dades.

Les classes que es corresponen a les taules seran alguna cosa com:

```
public class Actor implements Serializable {
    private static double serialVersionUID = 17L;
    private int id;
    private String nombre;
}

public class Genere implements Serializable {
    private static double serialVersionUID = 17L;
    private int id;
    private String nombre;
}
```

```
public class Film implements Serializable {
    private static double serialVersionUID = 17L;
    private String id;
    private String titulo;
    private int anyo;
    private String pais;
    private ArrayList<Actor> actores;
    private ArrayList<Genere> generos;
    private String sinopsis;
    private float nota;
}
```

!!! warning "Atenció"

- Recorda que les llistes d'actors i gèneres s'han de guardar en taules separades, amb les corresponents claus alienes
- Cada cop que es faça init deuria de crear-se de nou la BBDD, esborrant-se la anterior si existia i purgant totes les dades. Informa d'aquest fet a l'usuari.

### 5.3. scrap [web]

Realitza el scrapping de la pàgina indicada, dins de la plataforma **filmaffinity**. Haurà d'anar inserint les dades conforme es recuperen des de internet en les taules de **Film**, **Actor** i **Genere**.

Opcionalment pots:

1. Es facilita una implementació del scrapping que genera un **ArrayList<FilmAffinity>**. Sols has de recórrer-lo i anar inserint les dades a la BBDD a les taules corresponent. El projecte el tens en aquest [enllaç](#) per si vols fer-lo servir com a base. Es recomana la opció 2 a conitunuació.
2. **[RECOMANAT]** Carregar un arxiu json amb les dades de les pel·lícules i anar inserint-les a la BBDD

Com a programador has de continuar en aquest codi per a generar tots els **Film**, **Actor** i **Genere** que siguen necessari i inserir-los a la BBDD

!!! tip "Recomanacions"

- Es recomana mètodes separats de inserció de cada taula. Com que `Film` conté `Actor` i `Genere`, desde `inserirFilm` hauràs de cridar a `inserirActor` i `inserirGenere`, si escau.
- Si alguna pel·lícula ja existia (mateix id) no l'hauràs d'inserir de nou. La botem i ja està. El mateix amb els actors i gèneres

### 5.4. list\_all film|actor|genere [/p]

Aquesta part del menú seleccionarà i mostrarà per pantalla totes les dades existents a la base de dades.

Deura de rebre 3 arguments:

1. `list_all`: nom del argument. Deu de ser exactamnet aquest. Si no és, informar a l'usuari de l'error i presentar de nou el menu.
2. `film|actor|genere`: deu ser un dels nom del que volem mostrar. Si no és, informar a l'usuari de l'error i presentar de nou el menu.
3. `[/p]`: aquest tercer argument és opcional, però cas d'estar ha de ser exactament `/p`. Això el que fa es mostrar les dades amb *pausa*, de manera que presentarà les 10 primeres, indicant de mostrar un lletra per mostrar les 10 següents, etc.

Aquest tercer argument és una millora o ampliació de la pràctica. Fes-ho primer sense ell

## 5.5. `list_by actor nom_actor|genere_nom_genere asc|desc`

Aques comandament rep exacmanter 4 argument:

1. `list_by`: nom del argument. Deu de ser exactamnet aquest. Si no és, informar a l'usuari de l'error i presentar de nou el menu.
2. El segon i tercer poden ser:
  1. `actor nom_actor`: mostrarà les pel·lícules en les quals ha participat l'actor que el nom conté part del `nom_actor`. Si li passe `Will` mostrara les pel·lícules on ix `Robbie Williams`, `Will Smith`, `Dowill MAcKennie`, etc.
  2. `genere_nom_genere`: mostrara les pel·lícules que tingen exactament algun d'eixos gèneres. Recorda que hi ha pel·lícules amb més d'un genere
  3. `asc|desc`: mostrara les pel·lícules ordenades per títol de manera ascendent o descentent

!!! tip "Exemples de consultes" `list_by actor bardem asc list_by actor BAR desc list_by genere comedia asc list_by genere thriller desc`

## 5.6. `update idFilm`

Per acabar implementarem la modificació de alguna pel·lícula. a partir del seu id. Exactament tindrà dos arguments:

1. `update`: nom del argument. Deu de ser exactamnet aquest. Si no és, informar a l'usuari de l'error i presentar de nou el menu.
2. `idFilm`: el id de una pel·lícula a modificar.

A continuació el que farà el programa és:

1. Es verificarà la existència d'aquesta pel·lícula. Cas de no existir s'informarà a l'usuari
2. Aniran mostrant-se el valor actual dels camps de la pel·lícula, mostrant a continuació la pregunta `"Vols modificar aquest camp (s/n)? "`.
  1. Si es contesta que no es passa al següent camps.
  2. Si es contesta que si apareixerà una nova pregunta: `"Quin serà el nou valor del camp?: "`. S'arreglarà aquest valor
3. Un cop passats per tots els camps ja tindrem la pel·lícula a punt per modificar-la. Es farà efectiva dita modificació en la base de dades.

!!! warning "Atenció"

1. Al fer la lectura de les dades verificar que el valor introduït és del tipus requerit per cada camp (int, float o String)
2. No modificar les llistes (ni Actors ni Genere)

#### ## 4.7 Altres

Es demana fer 3 opcions de menú addicionals a les ja implementades, valorant la dificultat i creativitat. És un bon moment per a demostrar el que saps !!!

© Primera versió Joan Gerard Camarena, October-2025