ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη διαδικτυακού πληροφοριακού συστήματος προσωποποιημένης διαχείρισης ταινιών

Του φοιτητή

Καρακλά Κωνσταντίνου

Αρ. Μητρώου: 01/1889

Επιβλέπων καθηγητής

Αμανατιάδης Δημήτριος

Θεσσαλονίκη 2015

ΠΡΟΛΟΓΟΣ

Από τις αρχές της χιλιετίας οι ραγδαίες εξελίξεις στην ταχύτητα διασύνδεσης στο διαδίκτυο έδωσαν τη δυνατότητα στους χρήστες να έχουν εύκολη πρόσβαση σε μεγάλο όγκο πληροφοριών, μουσικής, ταινιών κλπ. Δυστυχώς, αντίστοιχα καθίσταται απαραίτητη μια οργανωτική δομή που θα εξασφαλίζει την εύκολή πρόσβαση και ταξινόμηση των πληροφοριών αυτών.

Στο κομμάτι των ταινιών, το πρόγραμμα της παρούσας πτυχιακής θα προσπαθήσει να βοηθήσει τον χρήστη να ταξινομήσει και να επιλέξει, βάσει των κριτηρίων του, ταινίες, αλλά και να εξατομικεύσει την εμπειρία που αποκομίζει, χρησιμοποιώντας το.

Αυτό γίνεται εφικτό με μια σειρά από τεχνολογίες, όπως οι βάσεις δεδομένων και η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί, αλλά κυρίως από την τεχνική screen scraping (parsing) που δίνει τη δυνατότητα απόκτησης πληροφοριών χωρίς την άδεια φέρ’ ειπείν των ιστοσελίδων αξιολόγησης ταινιών.

Τέλος, ανοίγει την πόρτα ώστε να επεκταθεί χρησιμοποιώντας ουσιαστικά data mining, που είναι, τρόπον τινά, η ανάλυση από μια σειρά πληροφοριών που αντλεί αρχικά από το parsing, και έπειτα από τις επιλογές του χρήστη, κατά την διάρκεια που χρησιμοποιεί το πρόγραμμα.

ΠΕΡΙΛΗΨΗ

Ο στόχος της παρούσας πτυχιακής ήταν η ανάπτυξη ενός προσωποποιημένου πληροφοριακού συστήματος διαχείρισης ταινιών. Στα παρακάτω κεφάλαια γίνεται ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν και κυρίως της τεχνικής parsing καθώς και προβλήματα που παρουσιάστηκαν κατά την υλοποίηση της. Τέλος αναφέρονται τα συμπεράσματα και μελλοντικές επεκτάσεις που θα μπορούσαν να προστεθούν.

ABSTRACT

The aim of this theses was to develop a personalized data management system to manage movies. In the following chapters, the technologies used are explained and specifically parsing, along with any problems encountered during its implementation. Finally conclusions and suggestions are presented, regarding possible future additions to the program.

ΕΥΧΑΡΙΣΤΙΕΣ (προαιρετικά)

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την υπομονή τους για ότι χρειάστηκε για την απόκτηση του πτυχίου μου, καθώς και φίλους και συμφοιτητές που με βοήθησαν ή συμπάσχαν μαζί μου όλα αυτά τα χρόνια.

Επίσης θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Δημήτρη Αμανατιάδη για την υπομονή και την καθοδήγηση του κατά της διάρκεια εκπόνησης της πτυχιακής.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Η γενική ιδέα της πτυχιακής ήταν ένα πληροφοριακό σύστημα προσωποποιημένης διαχείρισης ταινιών χωρίς να χρειάζονται ιδιαίτερες ενέργειες από τον χρήστη.

Αρχικά το μόνο που χρειάζεται, είναι ο χρήστης να διαλέξει ένα φάκελο με ταινίες, που το μόνο χαρακτηριστικό που πρέπει να έχουν είναι, τα ονόματα να είναι σωστά. Όπως βέβαια θα φανεί στη συνέχεια, κατά την διάρκεια του χρόνου, το πρόγραμμα ενδεχομένως θα χρειάζεται τροποποιήσεις.

Από την στιγμή που υπάρχουν οι πληροφορίες, χρειάζεται να αποθηκευτούν σε μια δομή, έτσι ώστε να είναι εύκολη η ανάκτηση τους βάση επιλογών του χρήστη. Σε αυτό το σημείο είναι η βασική αλληλεπίδραση του χρήστη με το πρόγραμμα.

Βάσει των κριτηρίων που επέλεξε ο χρήστης προκύπτει μια νέα λίστα ταινιών, από την οποία μπορεί να επιλέξει όποια επιθυμεί. Στη συνέχεια ο χρήστης έχει τη δυνατότητα να επιλέξει την ταινία της αρεσκείας του, οποία πλέον παρουσιάζει όλες τις απαραίτητες πληροφορίες γύρω από αυτήν. Βάσει των κριτηρίων που επέλεξε ο χρήστης, έχει σαν αποτέλεσμα μία ή περισσότερες ταινίες από τις οποίες θα επιλέξει μια και βάση αυτής, θα πάρει ένα εύρος πληροφοριών για να καταλήξει τελικά να την δει ή όχι.

Πολύ ενδιαφέρον είναι το γεγονός ότι σε μελλοντική έκδοση, θα μπορεί να είναι ακόμα πιο προσωποποιημένο -βάσει των επιλογών του χρήστη- έτσι ώστε να προτείνει παρόμοιες ταινίες, χωρίς περαιτέρω ενέργεια.

Στα επόμενα κεφάλαια σταδιακά θα αναλυθούν τα εργαλεία που χρειάστηκαν, τον τρόπο που χρησιμοποιήθηκαν και τις παρατηρήσεις, βάση των δυσκολιών που αντιμετωπίστηκαν.

ΚΕΦΑΛΑΙΟ 1

<ΕΡΓΑΛΕΊΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΎ>

ΕΙΣΑΓΩΓΗ

Στην πτυχιακή χρησιμοποιήθηκαν διάφορα εργαλεία & τεχνικές. Σε αυτό το κεφάλαιο θα αναλυθεί τι χρησιμοποιήθηκε και γιατί

1.1 Γλώσσα προγραμματισμού

Ίσως το πιο βασικό κομμάτι, όπως είναι και λογικό άλλωστε, είναι η γλώσσα προγραμματισμού. Στο συγκεκριμένο πρόγραμμα έγινε η επιλογή της Java για τους παρακάτω λόγους.

Αρχικά η Java είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού.

Ο όρος αντικειμενοστρεφής προγραμματισμός με την μοντέρνα έννοια του όρου, ακούστηκε για πρώτη φορά στο MIT στα τέλη της δεκαετίας του 1950, αρχές του 1960.

Η πρώτη αντικειμενοστρεφής προγραμματισμού γλώσσα θεωρείται η Simula (Sklenar, 1997), που όπως υποδηλώνει το όνομα χρησιμοποιούταν για προσομοιώσεις. Αναπτύχθηκε στο κέντρο πληροφορικής του Όσλο στη Νορβηγία από τους Ole-Johan Dahl & Kristen Nygaard.

Η Simula χάραξε το δρόμο για να δημιουργηθεί η C++, όταν στα τέλη της δεκαετίας του 1970, αρχή δεκαετίας του 1980, ο Bjorn Stroustrup ενσωμάτωσε στοιχεία αντικειμενοστραφούς προγραμματισμού στην C.

Στις αρχές της δεκαετίας του 1990 μια ομάδα της Sun άρχισε να αναπτύξει την Java σαν εναλλακτική της C++, στην αρχή για video on demand (VOD) εφαρμογές, αλλά τελικά για προγραμματισμό διαδικτυακών εφαρμογών.

Τι διαφορετικό έχει όμως μια αντικειμενοστρεφής γλώσσα προγραμματισμού ή object-oriented programming (OOP) (Codeproject, 2015). Σε αντίθεση με τις διαδικαστικές γλώσσες, που κάθε πρόταση είναι μια εντολή προς τον υπολογιστή, οι OOP γλώσσες είναι στημένες με βάση αντικείμενα & κλάσεις, η όλη φιλοσοφία τους είναι βασισμένη πάνω σε αυτά.

Στην πράξη αυτό σημαίνει ότι μπορούν να χρησιμοποιηθούν μέρη του κώδικα σε άλλα προγράμματα, αλλά και στο ίδιο το πρόγραμμα χρησιμοποιώντας για παράδειγμα κληρονομικότητα.

Τι είναι όμως το αντικείμενο. Το αντικείμενο μπορεί να είναι μια μεταβλητή, μια δομή δεδομένων ή μια λειτουργία.. Στην OOP, το αντικείμενο είναι ένα συγκεκριμένο στιγμιότυπο μια κλάσης, που μπορεί να περιλαμβάνει τα παραπάνω.

Στις OOP, παρατηρούνται τα παρακάτω χαρακτηριστικά που αναφέρθηκαν παραπάνω.

- Αντικείμενα (Objects)
- Κλάσεις (Classes)

Αλλά υπάρχουν ακόμα 4 βασικές έννοιες του OOP. (Oracle (Concepts), 2015)

- **Ενθυλάκωση (Encapsulation)**, είναι η ιδιότητα των κλάσεων να έχουν τα ιδιωτικά δεδομένα (private), «κρυμμένα» από το υπόλοιπο πρόγραμμα και να μπορούν να προσπεραστούν μόνο με τη βοήθεια δημόσιων μεθόδων (public)

- **Αφαιρετικότητα (Abstraction)**, είναι η ιδιότητα των κλάσεων που δίνει έμφαση στις ιδιότητες μια κλάσης για παράδειγμα, παρά σε συγκεκριμένα στοιχεία της. Χρησιμοποιείται για της διαχείριση πολυπλοκότητας σε μεγάλα προγράμματα.

- **Κληρονομικότητα (Inheritance)**, είναι η ιδιότητα των κλάσεων να επεκτείνονται σε καινούργιες κλάσεις «κληρονομώντας» τις μεθόδους τις αρχικής, αλλά δίνοντας τους τη δυνατότητα να προσθέσουν καινούργιες.

- **Πολυμορφικότητα (Polymorphism)**, είναι η ιδιότητα των κλάσεων να χρησιμοποιούνται με διαφορετικούς τρόπους ή «μορφές». Κάποιους από αυτούς τους τρόπους είναι οι: υπερφόρτωση μεθόδου (method overloading), υποσκέλιση μεθόδου (method overriding).

Υπάρχουν επίσης τα 5 principles (αρχές) του OOP που αποτελούν το ακρωνύμιο S.O.L.I.D (Martin, 2000) και είναι τα παρακάτω

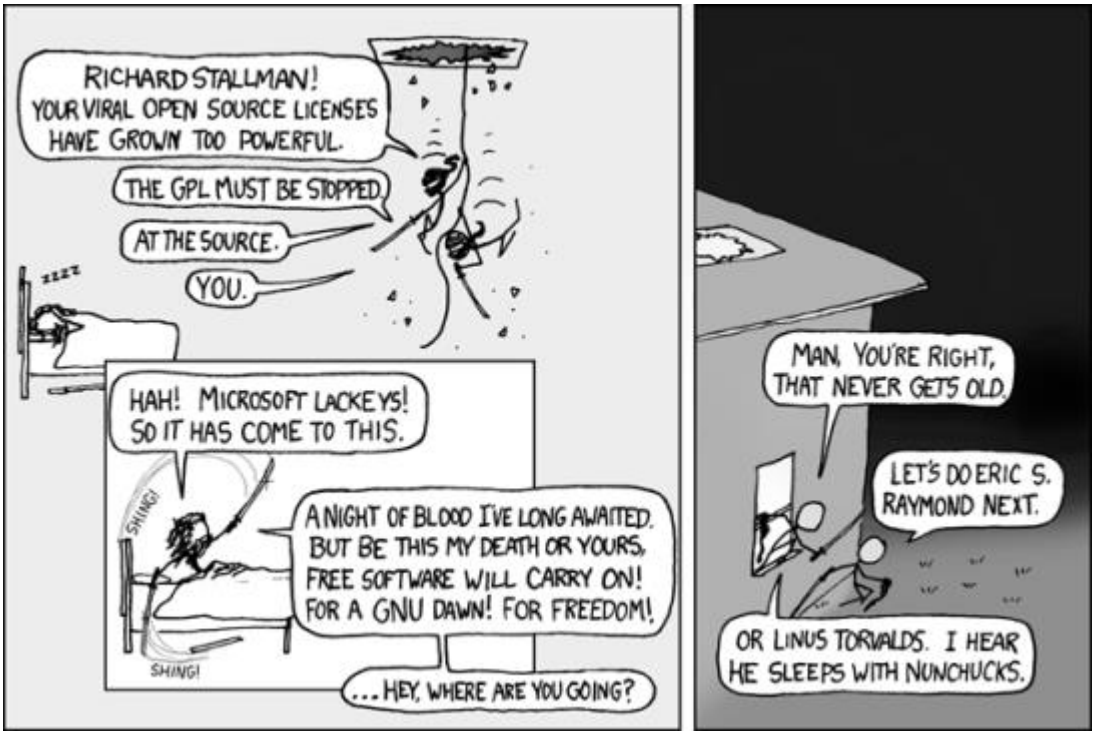
- **Single responsibility principle (Αρχή μοναδικής αρμοδιότητας).** Αυτή η αρχή περιγράφεται από τους Tom DeMarco & Meilir Page-Jones, και περιγράφεται απλά ότι η κλάση πρέπει να έχει μόνο μια αρμοδιότητα και οι κλάσεις-μέθοδοι της να εξυπηρετούν αυτόν το σκοπό στο πρόγραμμα.
- **Open closed principle (Αρχή ανοιχτότητας-κλειστότητας).** Ίσως μια από τις πιο σημαντικές αρχές, την ανέφερε πρώτος ο Bertrand Meyer και περιγράφεται ως εξής: Τα αντικείμενα – οντότητες πρέπει να είναι «ανοιχτά», ως προς την επέκταση των δυνατοτήτων τους, αλλά «κλειστά ως προς την τροποποίηση της υλοποίησης τους, πράγμα που καταφέρνουν οι OOP με τη βοήθεια των abstract classes και τη κληρονομικότητας.

- **Liskov Substitution Principle (Αρχή υποκατάστασης Λίσκοφ).** Αναφέρθηκε πρώτη φορά από την Barbara Liskov, και στην ουσία αναφέρεται στο γεγονός ότι functions που χρησιμοποιούν δείκτες ή αναφορές στην κύρια κλάση, πρέπει να μπορούν χρησιμοποιήσουν αντικείμενα που προέρχονται από την κύρια κλάση, χωρίς να ξέρουν όλες τις υλοποιήσεις αυτών των αντικειμένων.
- **Interface Segregation principle (Αρχή διαχωρισμού διασυνδέσεων).** Η αρχή του Robert C. Martin, στην ουσία προσπαθεί να εξηγήσει ότι, οι υποκλάσεις και τα interface που χρησιμοποιούνται σε ένα κομμάτι του προγράμματος να μη χρειάζεται να υλοποιήσουν κλάσεις που δεν χρησιμοποιούν.
- **Dependency Inversion principle (Αρχή αντιστροφής εξαρτήσεων).** Τέλος και πάλι από τον Robert C. Martin, και αφορά ιεραρχίες κληρονομικότητας κλάσεων & χρήση των αντικειμένων από εξωτερικά προγράμματα. Η συγκεκριμένη αρχή βασίζεται πάνω στην ιδιότητα της αφαιρετικότητας των κλάσεων και στην ουσία λέει ότι στοιχεία διαφορετικών επιπέδων πρέπει να βασίζονται, όχι μεταξύ τους αλλά με τη βοήθεια διαφορετικών διασυνδέσεων ή αφηρημένων κλάσεων.

Όσον αφορά την Java, υπάρχει μια πληθώρα λόγων που οδήγησαν στην απόφαση να χρησιμοποιηθεί. (Dumbill, 2011)

- **Compiler (Μεταγλωττιστής).** Αρχικά, ο πιο σημαντικός λόγος να χρησιμοποιήσεις Java, είναι ο compiler της. Ο JIT (Just-In-Time) compiler, σε αντίθεση με άλλους compilers, δεν κάνει compile σε γλώσσα μηχανής (code), αλλά σε bytecode, ανάλογα με το σύνολο εντολών του κάθε επεξεργαστή, με το αποτέλεσμα να είναι βέλτιστο και με τον τρόπο που κάνει τη μεταγλώττιση, να είναι ασφαλές (secure). (Oracle (JIT compiler), 2015)
- **Portability (Φορητότητα).** Λόγω του παραπάνω γεγονότος, η Java είναι η κατεξοχήν γλώσσα που τρέχει natively, cross – platform, πράγμα που την καθιστά ιδιαίτερα ελκυστική, ιδιαίτερα την εποχή που έχουμε πληθώρα από πλατφόρμες (βλέπε Android).
- **API (Application Program Interface).** Η διεπαφή με την αρχική εγκατάσταση, είναι πλούσια σε εργαλεία για οτιδήποτε χρειαστεί και πάνω από όλα είναι σταθερή & συμβατή (backwards compatible). Όπως είναι και το μότο, “Compile once and run forever”.

- **Open source libraries.** Λόγω της μεγάλης δημοτικότητας της Java, ότι δε καλύπτει το βασικό API, υπάρχουν βιβλιοθήκες όπως Apache, Google κλπ. που θα καλύψουν οποιαδήποτε ανάγκη μένει, δωρεάν…

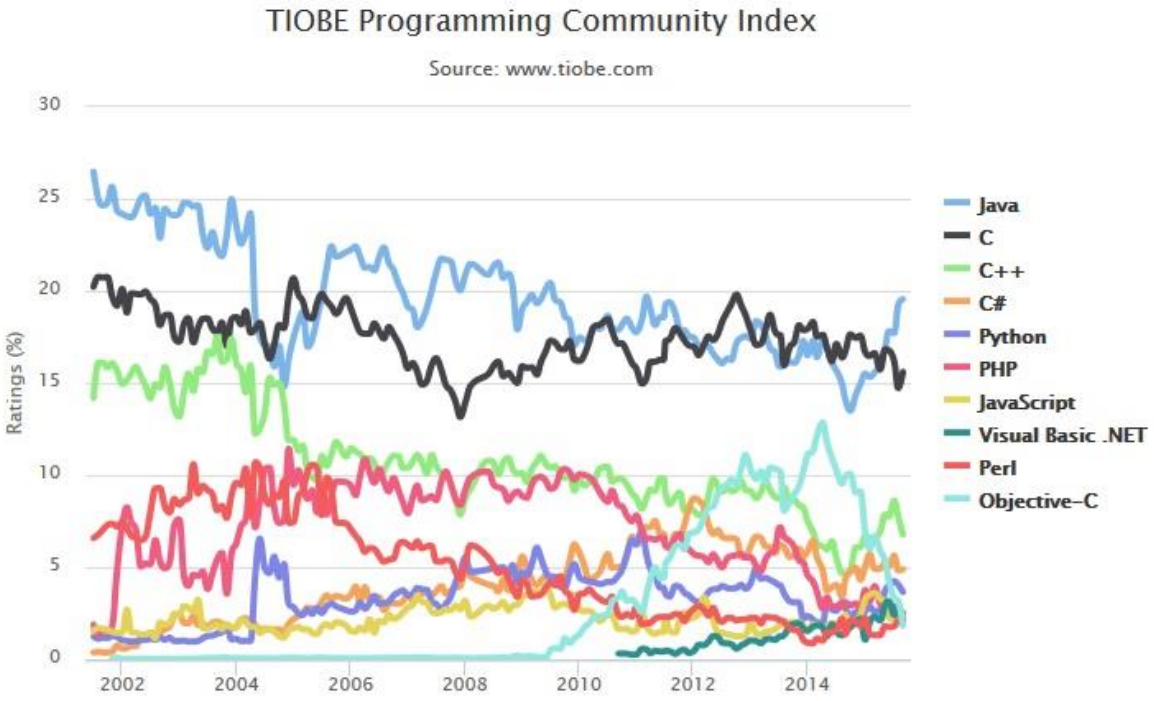


Εικόνα 1 – Σάτιρα της open source κουλτούρας

- **Δωρεάν εργαλεία.** Εκτός από το μεγάλο εύρος βιβλιοθηκών, που αναφέραμε, επίσης δωρεάν είναι η γλώσσα και το IDE (Integrated development environment). Έτσι αν θέλει κάποιος να γράψει σε Java, το αρχικό του κόστος είναι.. μηδέν, σε αντίθεση με άλλες πλατφόρμες.
- **Community.** Όπως καταλαβαίνουμε τα παραπάνω πάνε χέρι χέρι, με την κοινότητα που τα υποστηρίζει και προσθέτει σε αυτά συνεχώς.
- **Documentation.** Το Javadoc της Java το έκανε εύκολο, να μάθει και να κατανοήσει κάποιος τη γλώσσα, χωρίς να χρειάζεται να μελετά τον κώδικα.
- **Powerful IDE.** Όσον αφορά τα IDE που αναφέρθηκε, υπάρχουν πολύ δυνατά και καλά IDE για Java, όπως το Eclipse (Eclipse, 2015) και το NetBeans (Netbeans, 2015).

Το NetBeans ξεκίνησε το 1997 ως ένα ερευνητικό έργο στην Τσεχοσλοβακία (τώρα Τσεχία) και αρχικά ονομαζόταν Xelfi. Στόχος ήταν να γραφτεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης σε Java, βασισμένο σε Delphi like IDE's. Το 1999 αγοράστηκε από την Sun Microsystems αργότερα (το 2010) από την Oracle. Σήμερα είναι το κυρίαρχο Java IDE μαζί με το Eclipse. Με τις αυτοματοποιημένες λειτουργίες που διαθέτει προσφέρει μία αρκετά απλοποιημένη λύση για την ανάπτυξη λογισμικού, ειδικά σε Java Swing.

Από το παρακάτω index φαίνεται πάντως, ότι οι developers συμφωνούν με τους παραπάνω λόγους (Tiobe, 2015)



Εικόνα 2 – Programming community index

1.2 Βάση δεδομένων

Με τον όρο βάση δεδομένων, εννοείται μια συλλογή από αρχεία, οργανωμένα και συνδεδεμένα μεταξύ τους, ώστε να είναι δυνατή η ανάκτηση δεδομένων κατ' απαίτηση. Συγκεκριμένα στην πληροφορική όταν μιλάμε για βάσεις, εννοούμε το Σύστημα διαχείρισης βάσης δεδομένων (Database Management System) DBMS και συγκεκριμένα RDBMS (Relational DBMS), δηλαδή βάση βασισμένη στο σχεσιακό μοντέλο από τη δουλειά του E.F. Codd.

Σε τέτοιου τύπου βάσεων, υπάρχουν διαφορετικές οντότητες με ιδιότητες-χαρακτηριστικά που συνδέονται μεταξύ τους με συσχετίσεις. Όλα αυτά αρχίζουν από τη σχεδίαση της βάσης χρησιμοποιώντας το μοντέλο οντοτήτων-συσχετίσεων (ER model).

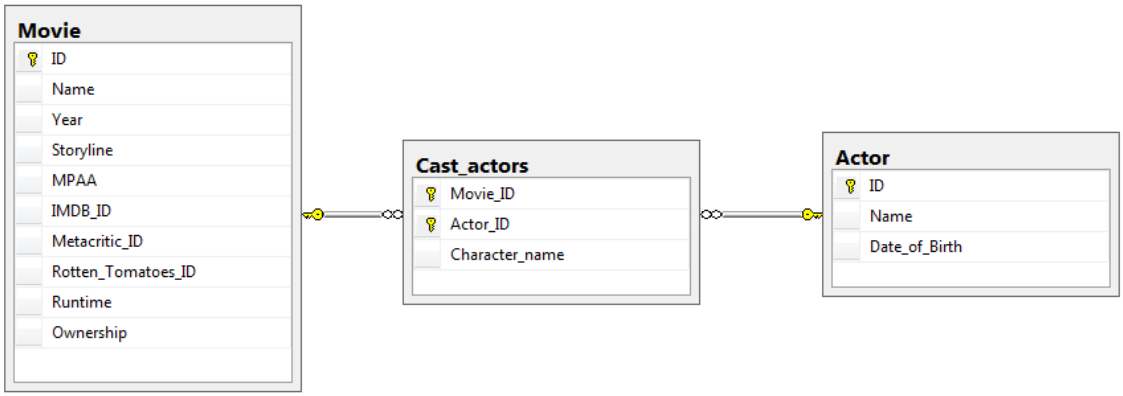
Όπως προαναφέρθηκε, υπάρχει μια οντότητα με κάποια χαρακτηριστικά, που ένα ή περισσότερα από αυτά, αποτελούν το κύριο κλειδί (primary key) και έχει μοναδική τιμή για όλα τα στιγμιότυπα της οντότητας.

Έπειτα υπάρχουν οι συσχετίσεις μεταξύ των οντοτήτων και υπάρχουν 3 βασικά είδη:

- **Ένα προς ένα.** Εδώ κάθε εμφάνιση μιας οντότητας συνδέεται με μια και μόνο εμφάνιση μιας άλλης οντότητας. Για παράδειγμα στη βάση της παρούσας πτυχιακής, κάθε ταινία έχει μια βαθμολογία στο IMDB.
- **Ένα προς πολλά.** Σε αυτή την περίπτωση, η κάθε οντότητα συνδέεται με πολλές εμφανίσεις μιας άλλης οντότητας. Για παράδειγμα στη βάση μας, μια ταινία μπορεί να παιχτεί σε πολλές γλώσσες.
- **Πολλά προς πολλά.** Τέλος εδώ, κάθε εμφάνιση μιας οντότητας συνδέετε με πολλές εμφανίσεις μιας άλλης οντότητας και αντίστροφα. Έτσι στη βάσης μας, ένα ηθοποιός μπορεί να παίζει σε πολλές ταινίες και μια ταινία έχει πολλούς ηθοποιούς.

Με τις συσχετίσεις λοιπόν, δημιουργείται η ανάγκη να υπάρχει ένα ξένο κλειδί στον πίνακα μας, το κλειδί αυτό είναι ένα χαρακτηριστικό του δεύτερου πίνακα πάνω στο οποίο γίνεται η συσχέτιση.

Ας δούμε λοιπόν ένα παράδειγμα από τη βάση μας. Παρακάτω βλέπουμε τους πίνακες Movie, Cast actors & Actor.



Εικόνα 3 – Συσχετίσεις μεταξύ των πινάκων

Εδώ στον πίνακα Movie που έχει στοιχεία από τις ταινίες, όπως όνομα, χρονιά, διάρκεια κλπ. Το κύριο κλειδί μας είναι το ID και παρατηρείται ότι υπάρχουν χαρακτηριστικά όπως το IMDB_ID, που είναι ξένο κλειδί από τον πίνακα IMDB (που έχει βαθμολογία για την κάθε ταινία).

Κατόπιν υπάρχει ο πίνακας Actor που έχει στοιχεία για τους ηθοποιούς και ο πίνακας Cast_actors που χρησιμοποιήθηκε για σύνδεση των δύο προηγούμενων πινάκων, με σχέση -όπως αναφέρθηκε προηγουμένως- πολλά προς πολλά.

Βλέποντας τις παρακάτω εγγραφές παρατηρείται αυτή την σχέση

	Name	Movie_ID	Character_Name	Actor_name
1	2 Guns (2013)	3714	Stig	Mark Wahlberg
2	Broken City (2013)	3738	Billy Taggart	Mark Wahlberg

Εικόνα 4 – Εγγραφές του πίνακα Cast actors

Εδώ φαίνεται ότι ο ηθοποιός Mark Wahlberg παίζει σε δύο (η περισσότερες) ταινίες και παρακάτω ότι η ταινία 2 guns έχει πολλούς ηθοποιούς (προφανώς)

	Name	Character_Name	Actor_name
1	2 Guns (2013)	Bobby	Denzel Washington
2	2 Guns (2013)	Stig	Mark Wahlberg
3	2 Guns (2013)	Deb	Paula Patton
4	2 Guns (2013)	Papi Greco	Edward James Olmos
5	2 Guns (2013)	Earl	Bill Paxton
6	2 Guns (2013)	Jessup	Robert John Burke
7	2 Guns (2013)	Quince	James Marsden
8	2 Guns (2013)	Chief Lucas	Greg Sproles
9	2 Guns (2013)	Admiral Tuwey	Fred Ward
10	2 Guns (2013)	Dr. Ken	Patrick Fischler

Εικόνα 5 – Εγγραφές από join πινάκων

Τα παραπάνω αποτελέσματα πάρθηκαν, χρησιμοποιώντας τη γλώσσα SQL (Structured Query Language).

Η SQL αναπτύχθηκε αρχικά στο IBM στις αρχές της δεκαετίας του 1970 αλλά προς το τέλος της δεκαετίας, η ORACLE, έβγαλε την πρώτη εμπορική μορφή της SQL, μόλις λίγες εβδομάδες πριν την IBM.

Όπως καταλαβαίνεται είναι μια γλώσσα που σχεδιάστηκε για τη διαχείριση βάσεων δεδομένων και δίνει τη δυνατότητα δημιουργίας, ανάκτησης, ενημέρωσης και τροποποίησης σχημάτων & πινάκων, πράγμα που καταφέρνουμε χρησιμοποιώντας statements, queries κλπ.

Για παράδειγμα για να πάρει κάποιος τις εγγραφές τις εικόνας μπορεί να χρησιμοποιήσει το παρακάτω query

```
select m.Name, c.Character_name as Character_Name, a.Name as Actor_name

from Cast_actors c

inner join Movie m on m.id=c.Movie_ID

inner join Actor a on a.ID=c.Actor_ID

where m.Name= '2 Guns (2013)'
```

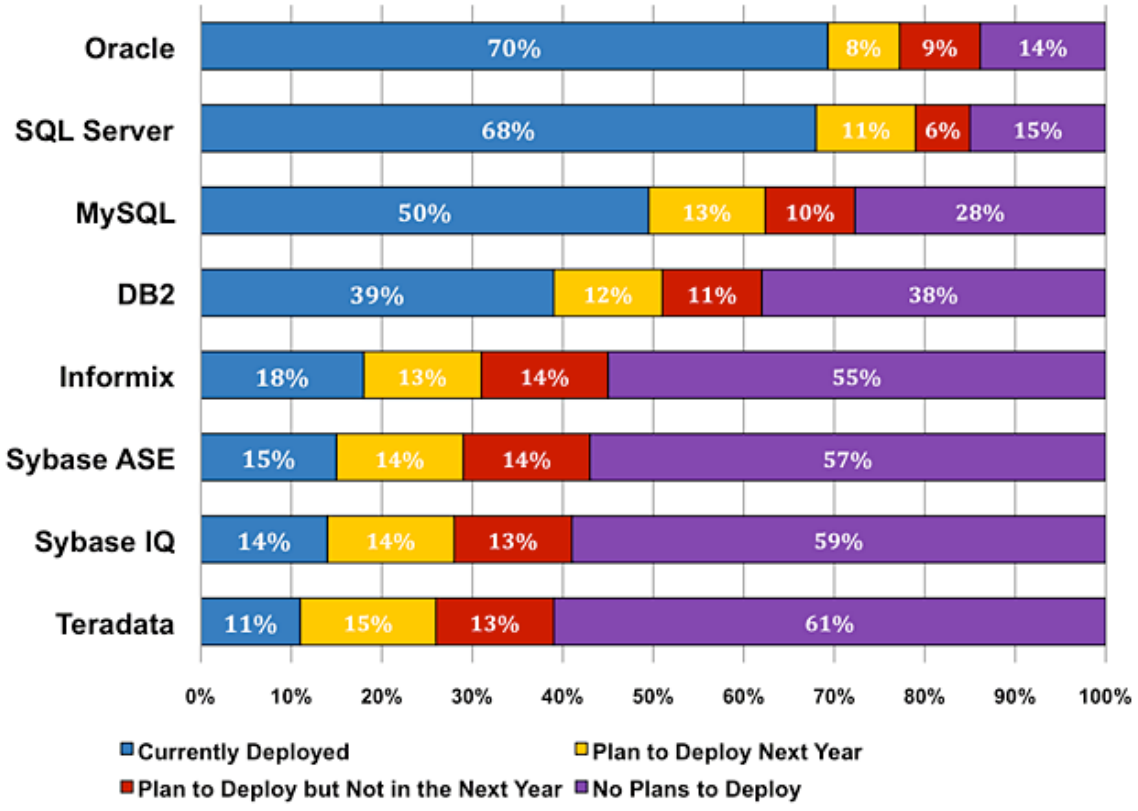
- Το select είναι οι στήλες που θέλει σαν αποτέλεσμα
- Το from είναι από ποιον πίνακα θέλει τις πληροφορίες (αν και στην περίπτωση αυτή είναι data από 3 πίνακες)Το inner join χρησιμοποιείται για να συνδεθούν οι πίνακες στα κλειδιά που θέλουμε
- Τέλος το where είναι η συνθήκη πάνω στην οποία θέλουμε τα αποτελέσματα

Χρησιμοποιώντας τέτοιου είδους queries και statements μπορεί να γίνει ανάκτηση και ενημέρωση της βάσης με τις πληροφορίες που θέλει ο χρήστης, αλλά ας μιλήσουμε για το ποια βάση χρησιμοποιήθηκε.

Στο συγκεκριμένο πρόγραμμα έγινε η επιλογή του Microsoft SQL server. Υπάρχει μια πληθώρα προγραμμάτων διαχείρισης βάσεων δεδομένων, όπως η Oracle & η IBM DB2 και open source συστήματα όπως η PostgreSQL & MySQL.

Η επιλογή του Microsoft SQL server έγινε διάφορους λόγους όπως:

- Φιλική διεπαφή για τον χρήστη
- Εύκολη ενσωμάτωση με λειτουργικά συστήματα Windows
- Μικτή ταυτοποίηση, είτε μέσω username-password, είτε μέσω windows authentication
- Εύκολη αποκατάσταση της βάσης από back-up
- Η δομή αδειοδότησης είναι καλύτερη από άλλα RDMS, για παράδειγμα το Studio express είναι δωρεάν.
- Το management studio είναι το καλύτερο εργαλείο για ένα προγραμματιστή
- Η διαχείριση & παρακολούθηση της βάσης χρησιμοποιώντας τα εργαλεία που προσφέρει την κάνει ιδιαίτερα ελκυστική σαν λύση



Εικόνα 6 – Μερίδιο αγοράς DBMS

1.3 Parsing

Το parsing γενικά είναι μια συντακτική ανάλυση ενός κειμένου έτσι ώστε να δημιουργηθεί μια δομή δεδομένων από τα στοιχεία που μας ενδιαφέρουν. Συγκεκριμένα στον προγραμματισμό, αναλύει τον πηγαίο κώδικα και επιστρέφει μια μορφή εσωτερικής αναπαράστασης.

Το web scraping (ή screen scraping) είναι μια τεχνική που χρησιμοποιείται για να εξαχθούν δεδομένα από το HTML μιας ιστοσελίδας.

Αρχικά να πούμε λίγα πράγματα για το HTML. Το HTML (HyperText Markup Language) είναι η τυπική markup language που χρησιμοποιείται για τη δημιουργία ιστοσελίδων. Στην ουσία η HTML περιγράφει σημασιολογικά τη δομή μιας σελίδας και για αυτό είναι markup language και όχι γλώσσα προγραμματισμού.

Η γλώσσα χρησιμοποιεί στοιχεία HTML που περικλείονται σε αγκύλες <html> τα οποία δεν είναι ορατά στο χρήστη, αλλά χρησιμοποιούνται από τους φυλλομετρητές ιστοσελίδων (web browsers) για ερμηνεύσουν το περιεχόμενο της ιστοσελίδας.

Για παράδειγμα

```
<!DOCTYPE html>
<html>
<head>
      <title>Page Title</title>
</head>
<body>
      <h1>This is a Heading</h1>
            <p>This is a paragraph.</p>
      </body>
</html>
```

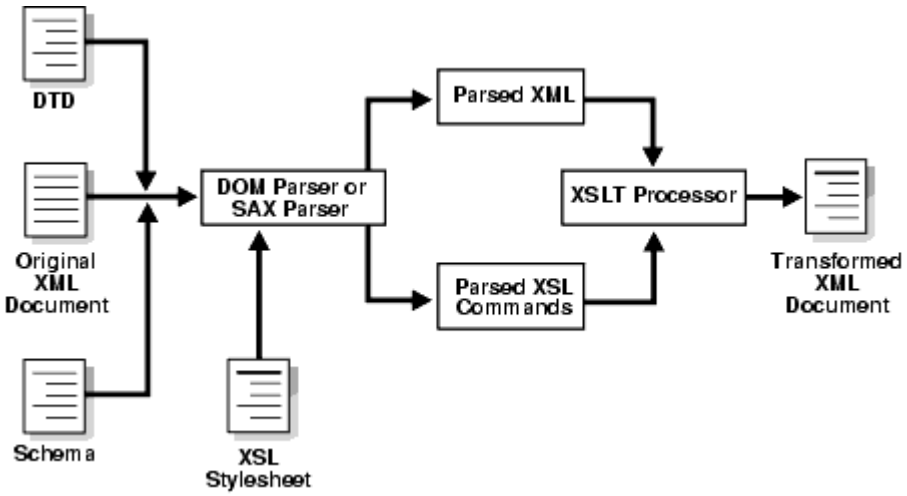
Επίσης η HTML μπορεί να ενσωματώσει scripts από γλώσσες όπως η JavaScript που επηρεάζουν τη συμπεριφορά της ιστοσελίδας, πολλές φορές σε συνδυασμό με τη CSS (Cascading Style Sheets) που επηρεάζει εμφάνιση και τη διάταξη του κειμένου.

Για να πάρουμε τα δεδομένα που θέλαμε υπήρχαν διάφοροι τρόποι όπως το API που προσφέρει η Rotten Tomatoes βασισμένο σε JSON (JavaScript Object Notation) ή API γραμμένα από χρήστες πάνω σε text files που δίνει η IMDB και αυτά χρησιμοποιώντας κυρίως JSON.

Το JSON (JavaScript Object Notation) είναι ένα πρότυπο ανταλλαγής δεδομένων, απλό με ιεραρχική μορφή, ανεξάρτητο από γλώσσες προγραμματισμού ή πλατφόρμες. (Ecma-international, 2013)

Χρησιμοποιώντας κάποιο API βασισμένο σε JSON, θα ήταν πολύ πιο εύκολο να πάρει κάνεις τα δεδομένα. Ωστόσο, κρίνεται ανούσιο επί της παρούσης, δεδομένου ότι το βασικό πρόβλημα που προσπαθεί να λύσει η παρούσα πτυχιακή είναι να πάρει δεδομένα χρησιμοποιώντας καθαρά κώδικα, χωρίς να χρησιμοποιεί «μασημένη τροφή».

Έτσι φτάνουμε στους HTML parsers. Ουσιαστικά υπάρχουν δύο τύποι, DOM και SAX parsers. (Oracle (Parsers), 2015)



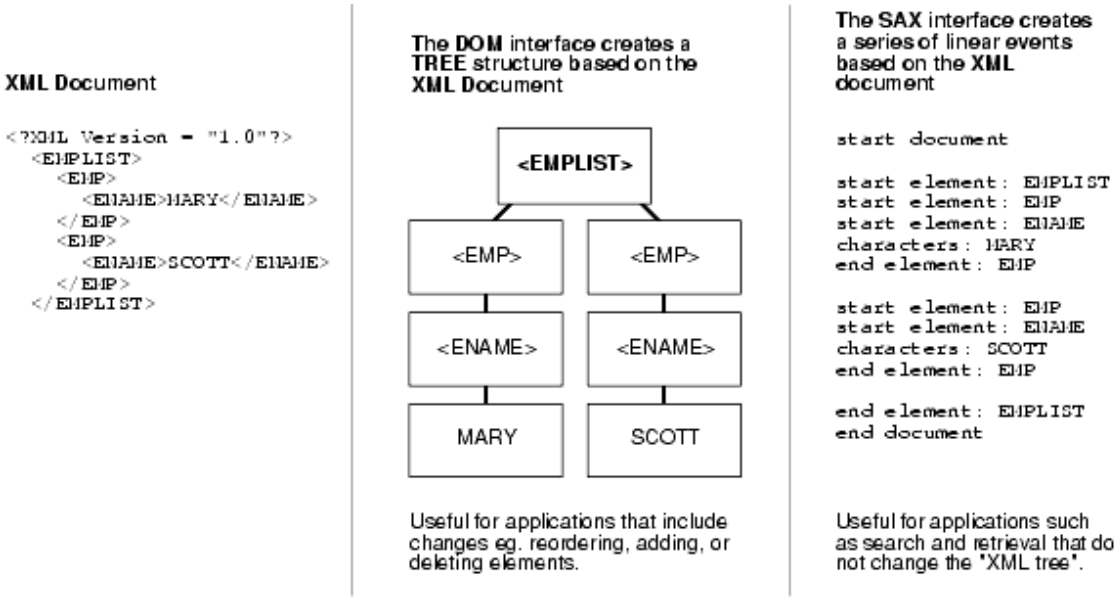
Εικόνα 7 – Διαδικασία parsing

Ο DOM (Document Object Model), είναι ένας tree based parser που κατασκευάζει στη μνήμη ένα κείμενο XML με τη μορφή αναπαράστασης δέντρου, παρέχοντας κλάσεις & μεθόδους για την περιήγηση και επεξεργασίας του δέντρου.

Ο SAX (Simple API for XML), είναι ένας event based parser που κάνει parse το κείμενο XML και αναγνωρίζει triggers που του έχουμε δώσει, κατά τη διάρκεια ανάγνωσης του XML, που αναγνωρίζονται από τους event handlers της JAVA.

Οι κύριες διαφορές τους είναι οι εξής:

- Ο DOM parser φορτώνει όλο το κείμενο XML στην μνήμη σε αντίθεση με τον SAX parser που φορτώνει μικρά κομμάτια.
- Λόγω της παραπάνω διαφοράς, ο DOM parser είναι πιο γρήγορος.
- Ακόμα λόγω αυτού, ο SAX είναι πιο κατάλληλος σε μεγάλα αρχεία XML
- Τέλος όπως αναφέρθηκε ο DOM parser είναι tree based ενώ ο SAX parser event based.



Εικόνα 8 – Διαφορές μεταξύ DOMS & SAX

Για την Java υπάρχουν διάφορες βιβλιοθήκες που θα μπορούσαν να χρησιμοποιηθούν όπως:

- Tagsoup (SAX based)
- HtmlUnit
- jARVEST
- Jsoup (DOM based)
- Jericho

Κρίθηκε σκόπιμο να χρησιμοποιηθεί ο Jsoup. Ο Jsoup είναι ένας DOMparser, γρήγορος και εύχρηστος, είναι open source που διανέμεται από το MITlicense και έχει γράψει ένας προγραμματιστής της Amazon, ο Jonathan Hedley

Προσφέρει μια πληθώρα μεθόδων χρησιμοποιώντας CSS / jquery like, selector syntax, DOM methods κλπ.

1.4 Movie Ratings

Αφού έχουν αναφερθεί τα εργαλεία που αφορούν το «πώς», μένει το «που». Η πιο γνωστή ιστοσελίδα για ταινίες είναι αδιαμφισβήτητα το IMDB. Είναι στις πρώτες 50 σελίδες στο Alexa Rank εδώ και χρόνια. Έχει ό,τι πληροφορία μπορεί να χρειάζεται κάποιος αναφορικά με μια ταινία & το cast της. Παράλληλα, προσφέρει τη δυνατότητα στους χρήστες να γράψουν αξιολογήσεις.

Φυσικά δεν είναι τέλεια. Για παράδειγμα αν κοιτάξουμε το TOP 250 βλέπουμε ότι εδώ και χρόνια η ταινία “The Shawshank Redemption (1994)” είναι πρώτη σε βαθμολογία με 9.2. Όποιος την έχει δει καταλαβαίνει ότι είναι καλή ταινία αλλά σίγουρα δεν είναι η καλύτερη ταινία που βγήκε ποτέ.

Εδώ έρχεται το RottenTomatoes, μια ακόμα γνωστή ιστοσελίδα που προσφέρει παρόμοιες πληροφορίες για ταινίες. Ωστόσο, η ιστοσελίδα αυτή δίνει περισσότερη σημασία κυρίως στις αξιολογήσεις, κάτι για το οποίο είναι στημένη εξάλλου. Επομένως, προσφέρει αξιολογήσεις τόσο από χρήστες όσο και από «επαγγελματίες» κριτικούς, με αποτέλεσμα να έχει πιο λογικές βαθμολογίες.

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, το Rotten Tomatoes προσφέρει API για queries σε JSON, και το IMDB έχει τα δεδομένα σε text αρχεία που μπορείς να κατεβάσεις και να χρησιμοποιήσεις. Στην παρούσα πτυχιακή όμως -όπως είπαμε- κάνουμε screen scraping τις ιστοσελίδες για να πάρουμε τα δεδομένα με ό,τι προβλήματα μπορεί να παρουσιαστούν.

Για παράδειγμα, κατά την διάρκεια εκπόνησης της πτυχιακής το IMDB άλλαξε μερικά μέρη κώδικα στην σελίδα με αποτέλεσμα να χρειαστεί να αλλάξει ο κώδικας για να προσαρμοστεί σε αυτές τις αλλαγές. Έτσι ένα μέρος του κώδικα που έπαιρνε τη εικόνα της ταινίας, έπαιρνε το πρώτο jpeg που έβρισκε, όμως το IMDB έβαλε στις καινούργιες ταινίες το trailer της ταινίας πάνω πάνω με αποτέλεσμα αντί για την εικόνα της ταινίας, να έπαιρνε την πρώτη εικόνα του trailer.

Καταλαβαίνουμε λοιπόν ότι τέτοιου είδους προγράμματα χρειάζονται συνεχώς συντήρηση και μικρό-αλλαγές, διότι βασίζεται σε κώδικα που δεν μπορεί να ελέγξει κάποιος.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο αναλύουμε τα εργαλεία που χρησιμοποιήσαμε στην υλοποίηση του προγράμματος. Στο επόμενο κεφάλαιο θα αναλύσουμε πως χρησιμοποιήσαμε το κάθε εργαλείο στην υλοποίηση.

ΚΕΦΑΛΑΙΟ 2

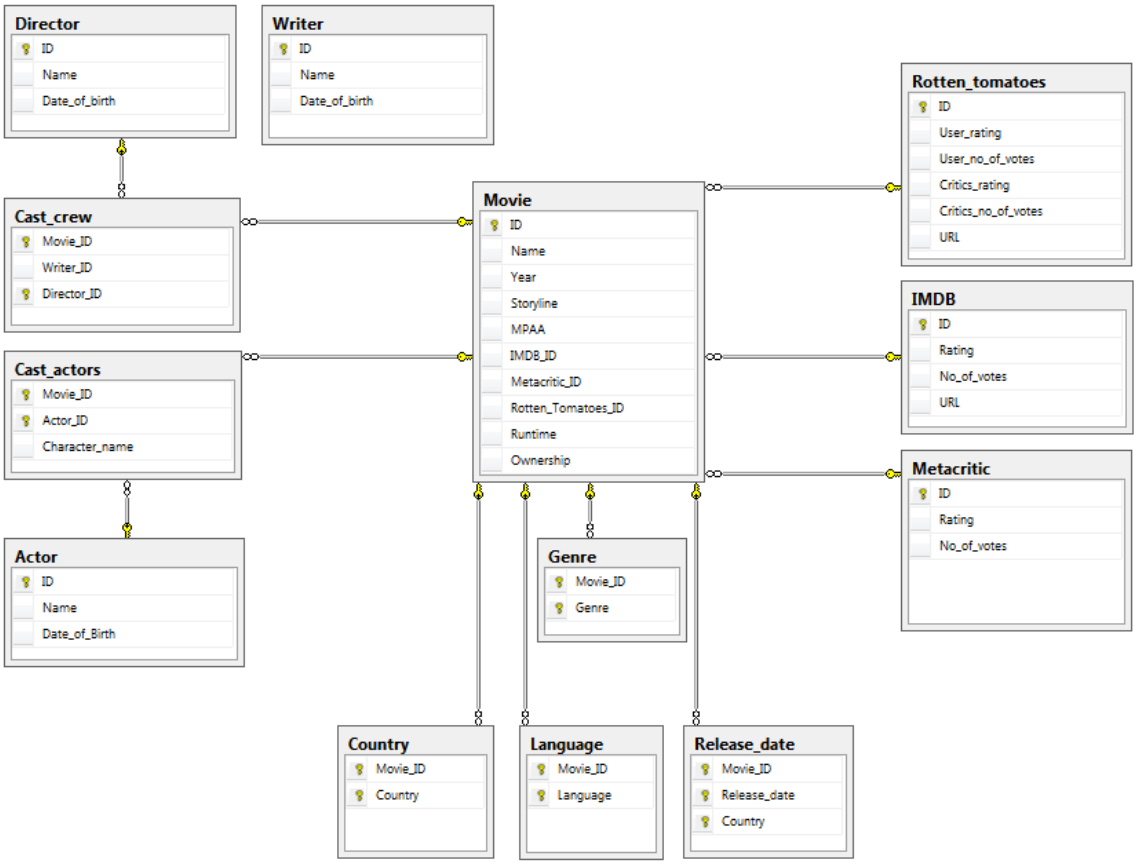
<ΥΛΟΠΟΙΗΣΗ>

ΕΙΣΑΓΩΓΗ

Σε αυτό κεφάλαιο παρουσιάζεται ο τρόπο υλοποίησης του προγράμματος σε κάθε κομμάτι του (NetBeans, SQL server, Jsoup κλπ.)

2.1 Βάση δεδομένων SQL server

Η βάση δεδομένων όπως είπαμε στήθηκε σε SQL server 2014. Παρακάτω φαίνεται το διάγραμμα της βάσης

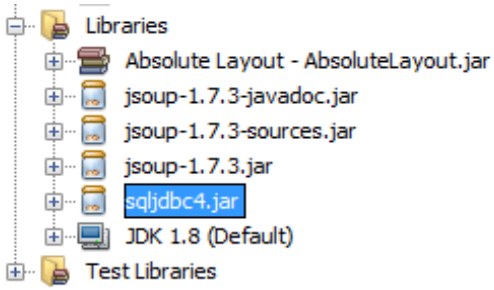


Εικόνα 9 – Διάγραμμα της βάσης δεδομένων

Κάποιοι πίνακες δε χρησιμοποιούνται σε αυτή την έκδοση του προγράμματος, όπως οι “Writer”, “Country”, “Release Date” για διάφορους λόγους. Για παράδειγμα ο Πίνακας “Release Date” που θα περιείχε τα στοιχεία πόλη πρεμιέρας &

ημερομηνία θα χρειαζόταν να γίνει parse μια ακόμα σελίδα για κάθε ταινία, πράγμα που θα καθυστερούσε την εκτέλεση του προγράμματος.

Για να λειτουργήσει η διασύνδεση της βάσης μας με το πρόγραμμα πρέπει να γίνει χρήση μιας βιβλιοθήκης, που στην περίπτωση μας είναι η Microsoft JDBC Driver 4.1. Κάνουμε add το αρχείο sqljdbc4.jar στα libraries



Εικόνα 10 – Microsoft JDBC Driver library

Έπειτα με τη μέθοδο “connect_to_SQL”, της κλάσης “Utilities” (όπως φαίνεται στο παράρτημα, γίνεται σύνδεση στη βάση όταν μια μέθοδος χρειάζεται πρόσβαση.

2.2 Jsoup

Το Jsoup, όπως είπαμε είναι ο HTML parser που χρησιμοποιήθηκε. Για να λειτουργήσει, χρειάζονται κάποιες βιβλιοθήκες που φαίνονται στη φωτογραφία παραπάνω. (Hedley, 2015)

Η βασική λειτουργία του Jsoup είναι να συνδεθεί σε μια σελίδα html για να μπορέσουμε να κάνουμε το parsing, ή σε περίπτωση αυτή screen scraping.

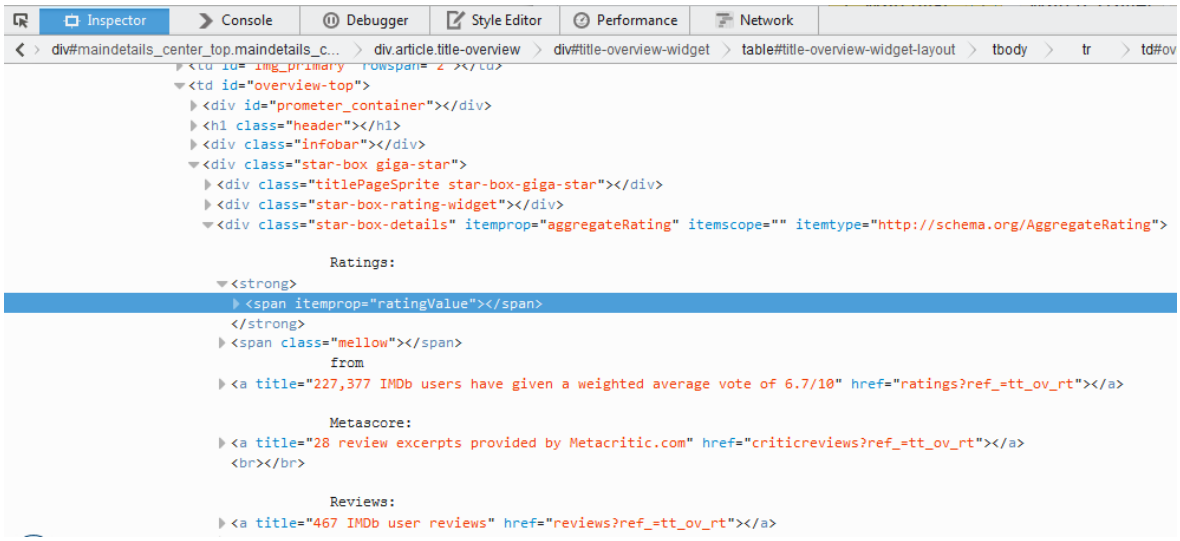
Αρχικά γίνεται τη σύνδεση με τη σελίδα με την παρακάτω εντολή

```
Document doc_IMDB = Jsoup.connect(IMDB_URL).get();
```

Αυτό που κάνει είναι μια σύνδεση με την σελίδα, παίρνει το html και το κάνει parse.

Από τη στιγμή που έχουμε την σελίδα που θέλουμε, μπορούμε να πάρουμε τα δεδομένα με την χρήση των “Elements”. Η κλάση “Element”, έχει όνομα, χαρακτηριστικά & κόμβους (nodes), τα οποία μπορούμε να χρησιμοποιηθούν για να για να γίνει ο χειρισμός του html.

Για παράδειγμα θέλει κάποιος να πάρει τη βαθμολογία IMDB από την ταινία “Expendables 2”. Υπάρχουν δύο τρόποι για να δει το source code, είτε κάνοντας δεξί κλικ και “Inspect Element” στο σημείο που θέλει, είτε δεξί κλικ και “View Page Source”. Παρακάτω φαίνονται και οι δύο τρόποι



Εικόνα 11 – Inspect page element

Εικόνα 12 – Page source

Από το html καταλαβαίνει ο χρήστης ότι η βαθμολογία που ψάχνει βρίσκεται στο tag

```
<span itemprop="ratingValue">6.7</span>
```

Χρησιμοποιώντας την κλάση “elements”, μπορεί να πάρει την πληροφορία με τον παρακάτω κώδικα

```
Elements link_IMDB_Rating =
doc_IMDB.select("span[itemprop=ratingValue]");
```

Κάποιες φορές η πληροφορία που χρειάζεται είναι σε tag που υπάρχει πολλές φορές στο html χωρίς κάποια διαφοροποίηση ή όταν δεν είναι μέσα σε tag. Για παράδειγμα στη μέθοδο “GetMovieURL”, ψάχνουμε τα link IMDB & Rotten Tomatoes για να πραγματοποιηθεί το parsing. Γίνεται η χρήση regular expression (*=) για να πάρουμε το link.

```
Elements links_rotten = doc_rotten.select("div[class*=kv]");
```

Κατ’ επέκταση για να πάρει το τελικό link χρειάζεται λίγη ακόμα δουλειά

```
String temp1= links_rotten.first().text();

String temp11= temp1.substring(0, temp1.indexOf(' '));

String Rotten_link= temp11.replaceAll("\\s+","");
```

Στην πραγματικότητα αυτό που γίνεται είναι ότι στην πρώτη εντολή παίρνει το πρώτο αποτέλεσμα του Google search.

Στην δεύτερη εντολή παίρνει μόνο το πρώτο κομμάτι του string που είναι το link.

Τέλος με την τρίτη εντολή χρησιμοποιώντας και πάλι regular expression (\s+), αφαιρούνται τυχόν κενά-tabs που έχει το string.

Καμιά φορά τη πληροφορία που χρειάζεται στην πραγματικότητα είναι ένα χαρακτηριστικό (attribute) του Element. Για παράδειγμα κοιτώντας το source code μιας ταινίας στο Rotten Tomatoes, ψάχνει το Rating count των user, δηλαδή πόσοι χρήστες ψήφισαν για την συγκεκριμένη βαθμολογία

```
<meta itemprop="ratingcount" content="141717"/>
```

Φαίνεται λοιπόν ότι ενώ το tag είναι το meta, στην πραγματικότητα ο αριθμός που θέλει είναι στο content που είναι attribute του meta, χρησιμοποιώντας την παρακάτω εντολή μπορεί να πάρει το στοιχείο που χρειάζεταο

```
String links_Audience_ReviewCount=
doc_ROTTEN.select("meta[itemprop=ratingcount]").attr("content");
```

Ακόμα υπάρχουν φορές που υπάρχει ένα αριθμός από αποτελέσματα και χρειάζεται κάποιο συγκεκριμένο, όπως στο παρακάτω παράδειγμα που χρειάζεται η εικόνα της ταινίας για να παρουσιαστεί στην τελική οθόνη με τα στοιχεία κάθε ταινίας.

```
for (org.jsoup.nodes.Element el : img) {

      String src = el.absUrl("src");

      URL url= new URL(src);

      BufferedImage image = ImageIO.read(url);

      int height = image.getHeight();

       if(height>300&&height<350) {

String Image_URL= src;

String Dest_Name= tempmoviename + ".jpg";

          SaveImage(Image_URL, Dest_Name);
```

Στην πραγματικότητα το parsing επιστρέφει ένα αριθμό από εικόνες, αλλά βρέθηκε ότι η σωστή εικόνα έχει ύψος ανάμεσα σε 300 και 350, οπότε μόλις γίνει parse, σταματάει το for.

Επίσης μπορεί να χρησιμοποιηθεί ένα στοιχείο Element και να για γίνει parse ένα καινούργιο στοιχείο Element όπως θα δούμε παρακάτω

```
Elements link_movie_temp = doc_ROTTEN.select("h1[class=movie_title]");

Elements link_Movie_Name= link_movie_temp.select("span[itemprop=name]");
```

Αυτό που συμβαίνει σε αυτή την περίπτωση είναι ότι το tag που χρειάζεται () υπάρχει πολλές φορές στον κώδικα του HTML, αλλά λιγότερες στο block h1 (από το οποίο φαίνεται η πρώτη εμφάνιση).

Όπως βλέπετε το όλο θέμα στο parsing είναι να αναλαλυφθεί που έχει αποθηκεύσει την πληροφορία που χρειάζεται, ο προγραμματιστής του site, που παρεμπιπτόντως πολλές φορές θα το κάνει δύσκολο για τον επίδοξο parser (για ακριβώς το λόγο, να μην είναι εύκολο να γίνει screen scraping το site) και τον τρόπο να πάρει αυτήν την πληροφορία χρησιμοποιώντας τα εργαλεία που του δίνει ο HTML parser, αλλά και άλλα εργαλεία, όπως regular expressions κλπ.

2.3 Eclipse & NetBeans

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, το βασικό κομμάτι του parsing είναι να βρεθέι στον κώδικα του HTML, οι πληροφορίες που χρειάζονται. Για αυτό το σκοπό χρησιμοποιήθηκε για τις δοκιμές το Eclipse IDE. Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο είναι ένα καταπληκτικό εργαλείο με πολλές δυνατότητες.

Στο Eclipse λοιπόν δοκιμάστηκαν & βρέθηκαν όλα τα data, πριν μεταφερθούν στο IDE του NetBeans για την τελική υλοποίηση.

Αρχικά χρειάζονται τα ονόματα από τις ταινίες. Αυτό μπορεί να γίνει από το Jtextfield “file_path_textfield”, είτε από το menu File->Import Folder, που είναι ένα FileChooser που έχει παραμετροποιηθεί να διαλέγει φακέλους αντί για αρχεία, όπως φαίνεται στο παράρτημα.

Από την στιγμή που υπάρχουν τα ονόματα των ταινιών, χρειάζονται τα URL τους στα site IMDB & Rotten Tomatoes. Αυτό θα επιτευχθεί με τη μέθοδο “GetMovieURL” της Utilities.

Έπειτα χρειάζονται ακόμα 3 parse για το IMDB, το cast και το Rotten Tomatoes για την απόκτηση των πληροφοριών της ταινίας που χρειάζονται, όπως φαίνεται αυτό στην μέθοδο “GetMovieData” της Utilities.

Εδώ πρέπει να αναφερθεί ότι τα δεδομένα αποθηκεύονται προσωρινά σε lists & Hash tables, πριν περαστούν στη βάση. Η απόφαση να μη χρησιμοποιηθούν κλάσεις Movie, Actor κλπ., ήταν για την μείωση της πολυπλοκότητας και την ταχύτητα εκτέλεσης του προγράμματος.

Από τη στιγμή που υπάρχουν τα δεδομένα προσωρινά στην ArrayList, Hashtable_of_Movies, που όπως υποδηλώνει το όνομα της, είναι μια λίστα από

Hash tables, που το καθένα αντιστοιχεί σε μια ταινία. Μετά μένει να εισαχθούν στην βάση, χρησιμοποιώντας τη μέθοδό “ImportDataSQL”, η οποία με statements SQL κάνει insert τα δεδομένα στη βάση, όπως υπάρχει στο παράρτημα.

Όπως παρατηρείται, σε διάφορα statements, πριν γίνει εισαγωγή των δεδομένων στη βάση, επεξεργάζονται περεταίρω με διάφορα εργαλεία όπως:

τη μέθοδο replace

```
str= Hashtable_of_Movies.get(i).get("Runtime").toString().
replaceAll("[^0-9.]", "");
```

που παίρνει μόνο τους αριθμούς

regular expression

```
str= Hashtable_of_Movies.get(i).get("Movie_Name").toString();
Matcher m = Pattern.compile("\\((.*?)\\)").matcher(str);
while (m.find()) {
str = m.group(1);
}
```

. Any character

* Matches the preceding pattern element zero or more times.

? Modifies the * regex that comes before to match as few times as possible.

τη μέθοδο substring

```
str= Hashtable_of_Movies.get(i).get("Genre").toString();
String genre_temp= str.substring(str.indexOf(":") + 1);
String[] genre_split= genre_temp.split("\\|");
for(int j=0; j<genre_split.length; j++) {
prSt.setInt(1, Movie_key);
     prSt.setString(2, genre_split[j].replaceAll("\\W",""));
     prSt.execute();
}
```

Εδώ χρειάζεται να χωριστούν τα Genre που γίνονται parse από το IMDB, τα οποία χωρίζονται με τον χαρακτήρα |, και επίσης πρέπει να αφαιρεθούν όλοι οι χαρακτήρες που δεν είναι αλφαριθμητικοί με το \W (κενά, tabs, special characters κλπ.)

Γενικά όπως βλέπετε, επειδή τα δεδομένα είναι σε τελείως raw μορφή, πρέπει να τα φέρουμε στην κατάσταση που τα θέλουμε πριν τα χρησιμοποιήσουμε.

Αφού λοιπόν έχει γίνει εισαγωγή των δεδομένων στη βάση, συνεχίζουμε στο tab Filters, όπου όπως είπαμε μπορεί να γίνει επιλογή κανενός ή περισσότερων φίλτρων για αναζήτηση ταινιών και με το κουμπί “Get Results” θα επιστρέψει τις ταινίες που αντιστοιχούν στα φίλτρα αυτά.

Αυτό επιτυγχάνεται με το ένα μεγάλο SQL select, στο οποίο έχουμε έχει γίνει inner join όλους τους πίνακες της βάσης

```
String dataset_1=

"select DISTINCT m.Name Movie_name\n" +

"from movie m\n" +

"inner join Rotten_tomatoes r on m.Rotten_Tomatoes_ID = r.ID\n" +

.

.

.

"inner join Actor a on a.ID = ca.Actor_ID\n" +

"where m.name like'"+ "%" + "'";
```

Έτσι όταν κάποιο compo box ή slider είναι επιλεγμένο προσθέτει το κριτήριο στο Where του statement για να πάρουμε τα αποτελέσματα, τα οποία τελικά ανοίγουν ένα καινούργιο instance της κλάσης Movie_info.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο έγινε ανάλυση του τρόπου υλοποίησης του προγράμματος και τα προβλήματα που προέκυψαν στην πορεία. Βλέπετε ότι λόγω των κλήσεων που κάνει ο parser και του γεγονότος ότι το DOM «κατεβάζει» όλο το html της κάθε ταινίας υπάρχει μια καθυστέρηση που αυξάνεται σημαντικά με τον αριθμό των ταινιών. Επίσης λόγω του ότι οι ιστοσελίδες δεν έχουν γραφτεί με τη λογική ότι κάποιος θα προσπαθήσει να κάνει (αυτό που κάνει το πρόγραμμα), δεν υπάρχουν όλες οι πληροφορίες που χρειάζονται στα tag του HTML.

ΚΕΦΑΛΑΙΟ 3

<ΠΕΡΑΙΤΕΡΩ ΑΝΑΠΤΥΞΗ>

ΕΙΣΑΓΩΓΗ

Για το εν λόγω σύστημα διαχείρισης υπάρχει μια πληθώρα από προσθήκες που μπορούν να γίνουν όσα αφορά την προσωποποίηση & τις λειτουργείες.

3.1 Επεκτάσεις προγράμματος

Αρχικά η πρώτη επέκταση που θα μπορούσε να υλοποιηθεί, είναι ένας media player. Έτσι όταν κάποιος βρει την ταινία που θέλει θα μπορεί να την δει μέσα στο πρόγραμμα (αν την έχει στην ταινιοθήκη του).

Κατά την διάρκεια που βλέπει την ταινία θα μπορεί να βάζει δείκτες σε συγκεκριμένα σημεία στο βίντεο κλιπ, ώστε π.χ. να ξαναδεί μια σκήνη που του άρεσε ή να πρέπει να φύγει και να θέλει να ξαναρχίσει από το ίδιο σημείο.

Αυτό έχει σαν πλεονέκτημα ότι το πρόγραμμα θα μπορεί να κρατάει στατιστικά βάση των προτιμήσεων του χρήστη και μετά από μερικές ταινίες που θα έχει, δει θα μπορεί να προτείνει βάση αυτών, παρόμοιες ταινίες, για παράδειγμα βάση είδους, ηθοποιού κλπ.

Αυτό το μοντέλο εφάρμοσε το Amazon με μεγάλη επιτυχία, χρησιμοποιώντας εξόρυξη πληροφοριών (data mining) βάση των προϊόντων που αγόραζε ή έψαχνε ο χρήστης. Έτσι για παράδειγμα με αλγόριθμους της μορφής παρακάτω, βρίσκουν παρόμοια προϊόντα για το οποία μπορεί να ενδιαφέρεται ο χρήστης. (Mangalindan, 2012)

For each item in product catalog, I1

For each customer C who purchased I1

For each item I2 purchased by customer C

Record that a customer purchased I1 and I2

For each item I2

Compute the similarity between I1 and I2 (Linden, et al., 2003)

Τέτοια αλγόριθμοι χρησιμοποιούνται εκτενώς σε προγράμματα data mining, σε ένα μεγάλο εύρος πεδίων τα τελευταία 15 χρόνια.

Έτσι για παράδειγμα αφού προτείνει τις ταινίες, θα μπορούσε να προτείνει και τρόπους απόκτησης τους, όπως να την αγοράσει ο χρήστης από το Amazon ή να την δει σε μια online υπηρεσία streaming

Έπειτα υπάρχουν προσθήκες που μπορούν να βελτιώσουν την εμπειρία του χρήστη, όπως για παράδειγμα ένα progress bar, που να δίνει μια εκτίμηση για το χρόνο που θα χρειαστεί το πρόγραμμα κατά την διαδικασία του parsing ή του update.

Τέλος θα μπορούσε να συνδέεται μεταξύ χρηστών, ώστε μπορεί ο χρήστης να βαθμολογεί & να σχολιάζει τις ταινίες, πληροφορία που θα ήταν διαθέσιμη σε άλλους χρήστες. Έτσι θα μπορούσαν να υπάρχουν πληροφορίες για τις προτιμήσεις του κάθε χρήστη σε ένα κεντρικό server και να βγαίνουν στατιστικά ανάλογα με ηλικία, φύλο κλπ.

ΕΠΙΛΟΓΟΣ

Φαίνεται ότι το πρόγραμμα μπορεί να καταλήξει να έχει μεγάλο αριθμό υπηρεσιών που να χρησιμοποιούν πολλαπλές τεχνολογίες της εποχής, όπως screen scraping, data mining, torrents, streaming services κλπ.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η αρχική ιδέα του προγράμματος, ήταν να συλλέγει όσο τον δυνατόν περισσότερα στοιχεία για τις ταινίες και τους συντελεστές που την αποτελούσαν. Φάνηκε όμως ότι κάθε στοιχείο που βρισκόταν σε διαφορετική ιστοσελίδα από τις αρχικές πρόσθετε σημαντικά στο χρόνο εκτέλεσης, σε συνδυασμό με τον αριθμό ταινιών. Έτσι πάρθηκε η απόφαση να μειωθούν οι πληροφορίες που επιστρέφει το πρόγραμμα για μια ταινία και τους συντελεστές, έτσι ώστε να ανταποκρίνεται πιο γρήγορα.

Κατά την διάρκεια προγραμματισμού, φάνηκε ότι οι ιστοσελίδες δεν είναι δομημένες κατάλληλα, ώστε να επιστραφούν τα κατάλληλα δεδομένα. Σίγουρα πολλές φορές, είναι επίτηδες έτσι η δομή ώστε να μην μπορεί να γίνει αυτό, που κάνει το πρόγραμμα. Έτσι χρειάζονται τεχνικές όπως τα regular expression που απομονώνουν την πληροφορία που χρειάζεται.

Συγκεκριμένα το IMDB, αναφέρει ότι απαγορεύεται το parsing, εκτός αν δώσουν άδεια

Robots and Screen Scraping: *You may not use data mining, robots, screen scraping, or similar data gathering and extraction tools on this site, except with our express written consent as noted below.*

Φαίνεται ότι αυτό το πρόβλημα είναι επιδημικό στον τεχνολογικό κόσμο και πολλές εταιρίες έχουνε πρόβλημα με ανταγωνίστριες εταιρίες που «κλέβουν» δεδομένα και τα παρουσιάζουν στη δικιά τους ιστοσελίδα ή τα πουλάνε σε τρίτους. (Ward, 2013)

Υπάρχουν κατά καιρούς εταιρίες που καταλήγουν στα δικαστήρια για να λύσουν τέτοιες διαμάχες με μικτά αποτελέσματα. Όπως την περίπτωση του hacker Andrew Auernheimer που χρησιμοποιώντας απλές τεχνικές web scraping, κατάφερε να πάρει 114000 διευθύνσεις ηλεκτρονικού ταχυδρομείου από την AT& T (Greenberg, 2012). Καταδικάστηκε αρχικά, αλλά μερικούς μήνες αργότερα η απόφαση ανατράπηκε.

Πάντως με την μεγάλη ανάπτυξη που έχει το data mining γενικά, φαίνεται ότι τα πνευματικά δικαιώματα & προσωπικά στοιχεία δεν είναι ασφαλή.

ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ

Codeproject, 2015. *www.codeproject.com.* [Ηλεκτρονικό]
Available at: http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep
[Πρόσβαση September 2015].

Dumbill, E., 2011. *www.oreilly.com.* [Ηλεκτρονικό]
Available at: http://radar.oreilly.com/2011/07/7-reasons-to-use-java.html
[Πρόσβαση September 2015].

Eclipse, 2015. *www.eclipse.org.* [Ηλεκτρονικό]
Available at:
http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm
[Πρόσβαση September 2015].

Ecma-international, 2013. *www.ecma-international.org.* [Ηλεκτρονικό]
Available at: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[Πρόσβαση September 2015].

Greenberg, A., 2012. *www.forbes.com.* [Ηλεκτρονικό]
Available at: http://www.forbes.com/sites/andygreenberg/2012/11/21/security-researchers-cry-foul-over-conviction-of-att-ipad-hacker/
[Πρόσβαση September 2015].

Hedley, J., 2015. *http://jsoup.org/.* [Ηλεκτρονικό]
Available at: http://jsoup.org/apidocs/
[Πρόσβαση September 2015].

Linden, G., Smith, B. & York, J., 2003. *http://www.cs.umd.edu/.* [Ηλεκτρονικό]
Available at: http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf
[Πρόσβαση September 2015].

Mangalindan, J., 2012. *www.fortune.com.* [Ηλεκτρονικό]
Available at: http://fortune.com/2012/07/30/amazons-recommendation-secret/
[Πρόσβαση September 2015].

Martin, R. C., 2000. *www.objectmentor.com.* [Ηλεκτρονικό]
Available at:
http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf
[Πρόσβαση September 2015].

Netbeans, 2015. *www.netbeans.org.* [Ηλεκτρονικό]
Available at: https://netbeans.org/about/history.html
[Πρόσβαση September 2015].

Oracle (Concepts), 2015. *www.oracle.com.* [Ηλεκτρονικό]
Available at: https://docs.oracle.com/javase/tutorial/java/concepts/
[Πρόσβαση September 2015].

Oracle (JIT compiler), 2015. *www.oracle.com.* [Ηλεκτρονικό]
Available at: http://docs.oracle.com/cd/E15289_01/doc.40/e15058/underst_jit.htm
[Πρόσβαση September 2015].

Oracle (Parsers), 2015. *www.oracle.com.* [Ηλεκτρονικό]
Available at:
http://docs.oracle.com/cd/B12037_01/appdev.101/b10794/adx04paj.htm
[Πρόσβαση September 2015].

Rottentomatoes, 2015. *www.rottentomatoes.com.* [Ηλεκτρονικό]
Available at: http://developer.rottentomatoes.com/docs
[Πρόσβαση September 2015].

Sklenar, J., 1997. *http://www.um.edu.mt/.* [Ηλεκτρονικό]
Available at: http://staff.um.edu.mt/jskl1/talk.html
[Πρόσβαση September 2015].

Tiobe, 2015. *www.tiobe.com.* [Ηλεκτρονικό]
Available at: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[Πρόσβαση September 2015].

Ward, M., 2013. *www.bbc.com.* [Ηλεκτρονικό]
Available at: http://www.bbc.com/news/technology-23988890
[Πρόσβαση September 2015].

ΠΑΡΑΡΤΗΜΑΤΑ

Connect to SQL

```
public static Connection connect_to_SQL (){

        try {

            Connection conn = null;

            String dbURL =
"jdbc:sqlserver://localhost\\SQL_P_M;database=Listmovies";

            String user = "admin";

            String pass = "1234";

            conn = DriverManager.getConnection(dbURL,user,pass);

            return conn;

        } catch (SQLException ex) {

            Logger.getLogger(Utilities.class.getName()).log(Level.SEVERE,
null, ex);

            return null;

        }

    }
```

Get movie Data (part of)

```
Document doc_IMDB = Jsoup.connect(IMDB_URL).get();

Document doc_IMDB_cast = Jsoup.connect(IMDB_cast).get();

Document doc_ROTTEN = Jsoup.connect(Rotten_URL).get();

//IMDB Movie Info

Elements link_IMDB_Rating =
doc_IMDB.select("span[itemprop=ratingValue]"); //IMDB Rating

Elements link_IMDB_Users_count =
doc_IMDB.select("span[itemprop=ratingCount]"); //IMDB Users count

Elements link_Storyline = doc_IMDB.select("div[class=inline canwrap"); //
Storyline

Elements link_MPPA_Rating =
doc_IMDB.select("span[itemprop=contentRating]"); // MPAA Rating

Elements link_Runtime = doc_IMDB.select("time[itemprop=duration]"); //
Runtime

Elements link_Metacritic_Data =
doc_IMDB.select("a[href=criticreviews?ref_=tt_ov_rt]"); // Metacritic
Rating - Users

Elements link_Genre = doc_IMDB.select("div[itemprop=genre]"); //Genres
```

```
Elements link_movie_temp = doc_ROTTEN.select("h1[class=movie_title]");

Elements link_Movie_Name= link_movie_temp.select("span[itemprop=name]");

String tempmoviename= link_Movie_Name.text().replaceAll(":","-
").replaceAll("'", "");

System.out.println(tempmoviename);

Elements link_Director= doc_ROTTEN.select("td[itemprop=director");

Elements img = doc_IMDB.getElementsByTag("img");

            for (org.jsoup.nodes.Element el : img) {

                  String src = el.absUrl("src");

                        URL url= new URL(src);

                        BufferedImage image = ImageIO.read(url);

                        int height = image.getHeight();

                  if(height>300&&height<350) {

                        String Image_URL= src;

                                    System.out.println (src);

                                    String Dest_Name= tempmoviename + ".jpg";

                                    SaveImage(Image_URL, Dest_Name);

                                    break;

                  }

            }
```

Import Data SQL (part of)

```
int IMDB_key = 0;

String sql = "INSERT INTO IMDB(Rating, No_of_votes, URL) VALUES(?,?,?)";

PreparedStatement prSt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

float fl =
Float.valueOf(Hashtable_of_Movies.get(i).get("IMDB_Rating").toString());

prSt.setFloat(1, fl);

String usercount =
Hashtable_of_Movies.get(i).get("IMDB_Users_count").toString().replace(","
, "");

int nm = Integer.valueOf(usercount);

prSt.setInt(2, nm);

String str = Hashtable_of_Movies.get(i).get("IMDB_URL").toString();

prSt.setString(3, str);
```

```
prSt.execute();

ResultSet keys = prSt.getGeneratedKeys();

keys.next();

IMDB_key = keys.getInt(1);

keys.close();
```

Get Movie URL

```
movie.updateTextArea("Getting URL's for movie: " + Imported_Movie);



String url_rotten=
"https://www.google.com/search?q="+Imported_Movie+"rotten&ie=utf-
8&oe=utf-8&gws_rd=cr&ei=7o7GVLjVE47lap6UgoAN";

String url IMDB=
"https://www.google.com/search?q="+Imported_Movie+"+imdb&ie=utf-8&oe=utf-
8&gws_rd=cr&ei=7o7GVLjVE47lap6UgoAN";



url_rotten= url_rotten.replace(" ", "+");

url_IMDB= url_IMDB.replace(" ", "+");



Document doc_rotten =
Jsoup.connect(url_rotten).userAgent("Mozilla").ignoreHttpErrors(true).tim
eout(0).get();

Document doc_IMDB =
Jsoup.connect(url_IMDB).userAgent("Mozilla").ignoreHttpErrors(true).timeo
ut(0).get();

Elements links_rotten = doc_rotten.select("div[class*=kv]");

Elements links_IMDB = doc_IMDB.select("div[class*=kv]");

String temp1= links_rotten.first().text();

String temp11= temp1.substring(0, temp1.indexOf(' '));

String Rotten_link= temp11.replaceAll("\\s+","");

Rotten_link= "http://" + Rotten_link;

System.out.println(Rotten_link);

String temp2= links_IMDB.first().text();

String temp22= temp2.substring(0, temp2.indexOf(' '));

IMDB_link= "http://" + IMDB_link;

System.out.println(IMDB_link);

Utilities.Movies_IMDB_URL.put(Imported_Movie, IMDB_link);

Utilities.Movies_Rotten_URL.put(Imported_Movie, Rotten_link);
```

Save Image

```
String Folder_Path = "C:\\Users\\mental-RoG\\Netbeans\\Movies\\images\\";
URL url = new URL(Image_URL);
InputStream is = url.openStream();
OutputStream os = new FileOutputStream(Folder_Path + Dest_File);
byte[] b = new byte[2048];
int length;
while ((length = is.read(b)) != -1) {
      os.write(b, 0, length);
      }
is.close();
os.close();
```

File chooser

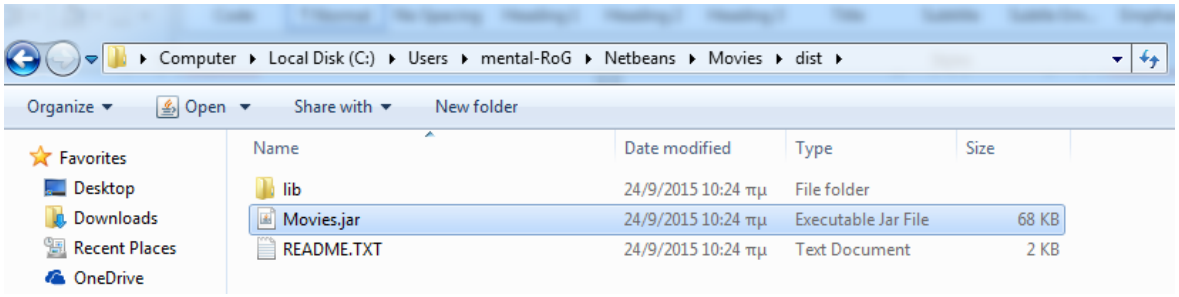
```
FileChooser chooser= new JFileChooser();
chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
chooser.showOpenDialog(this);
File f= chooser.getSelectedFile();
String filepath= f.getAbsolutePath();
file_path_textfield.setText(filepath);
```

Έπειτα χρησιμοποιώντας την κλάση File παίρνουμε τα ονόματα των ταινιών και τα παίρναμε σε μια λίστα

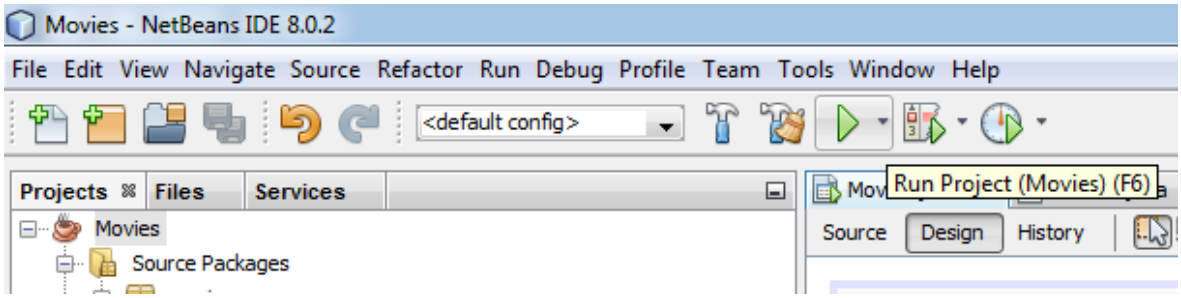
```
File directory = new File(temp);
File[] fList = directory.listFiles();
if(Utilities.ImportedMovies.isEmpty()) {
                    for (File file : fList) {
                        if (file.isDirectory()) {
                            Utilities.ImportedMovies.add(file.getName());
                        }
                    }
```

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Πως λειτουργεί η εφαρμογή. Μπορούμε να γίνει εκκίνηση του προγράμματος είτε εκτελώντας το αρχείο .jar που δημιουργεί το NetBeans κατά τη μεταγλώττιση του προγράμματος, είτε εκτελώντας (run) το πρόγραμμα.

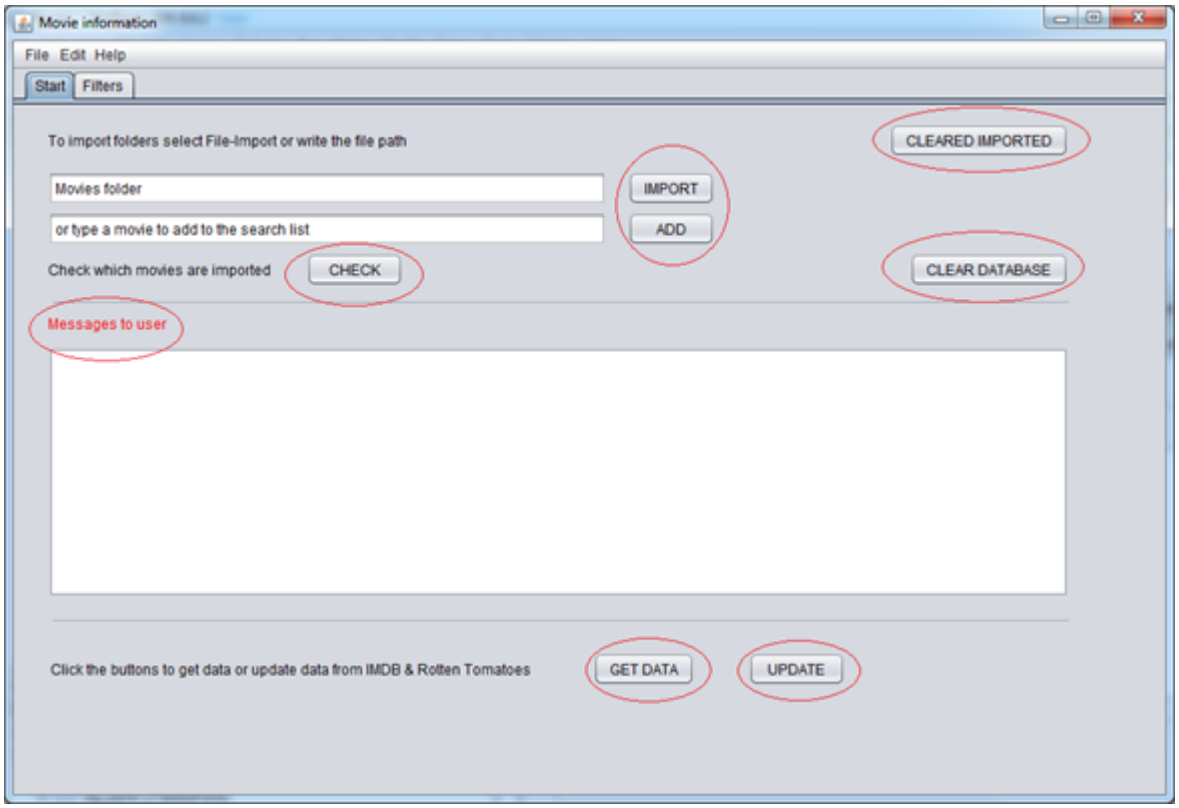


Εικόνα 13 – Εκτελέσιμο αρχείο .jar



Εικόνα 14 – Run project

Η αρχική οθόνη που βλέπει ο χρήστης είναι η παρακάτω

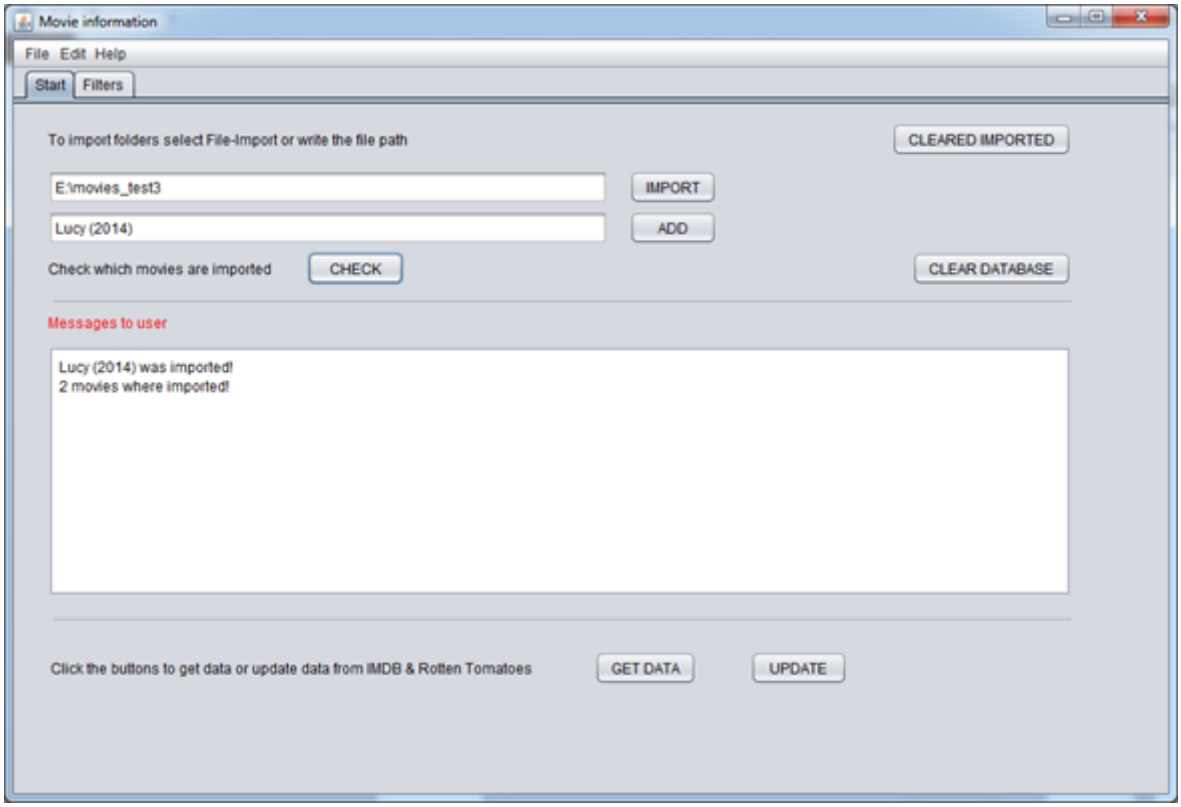


Εικόνα 15 – Αρχική οθόνη

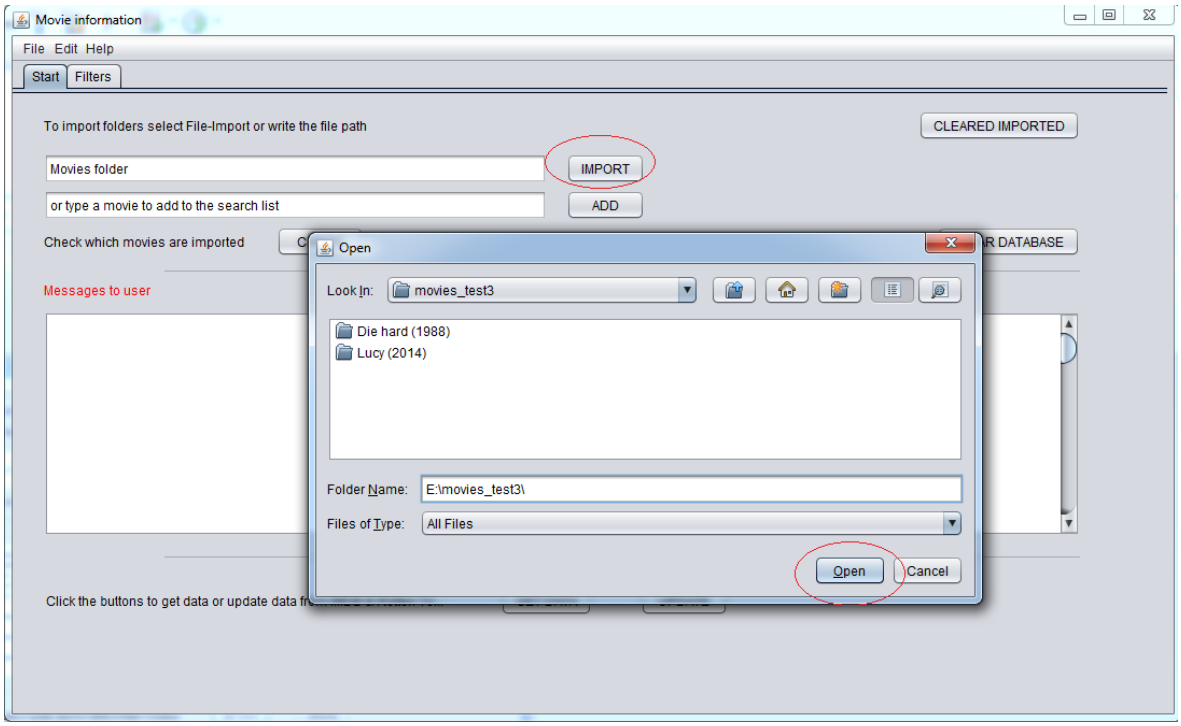
Οι βασικές λειτουργίες της αρχικής οθόνης είναι:

- Τα κουμπιά “Import” & “Add” χρησιμοποιούνται για να προστεθούν ταινίες είτε από φάκελο, είτε μία μια αντίστοιχα, στην λίστα ταινιών που θα ψάξει το πρόγραμμα να πάρει πληροφορίες.
- Το κουμπί “Check” για να ελέγξει τις ταινίες στις λίστα.
- Το κουμπί “Cleared Imported” που σβήνει τα στοιχεία της λίστας.
- Το κουμπί “Clear Database” που σβήνει όλη τη βάση.
- Το text area, “Messages to user” που δίνει feedback κατά την εκτέλεση του προγράμματος.
- Τα κουμπιά “Get Data” & “Update”, που ψάχνει στοιχεία για τις ταινίες στη λίστα, ανανεώνει κάποια στοιχεία των υπάρχοντών ταινιών στη βάση αντίστοιχα.
- Υπάρχουν κάποια menu items για διάφορές μικρές εργασίες ή για διαφορετικό τρόπο επιλογής φακέλου, File -> Import.
- Τέλος υπάρχει ένα progress bar, που δεν έχει υλοποιηθεί στην συγκεκριμένη έκδοση του προγράμματος.

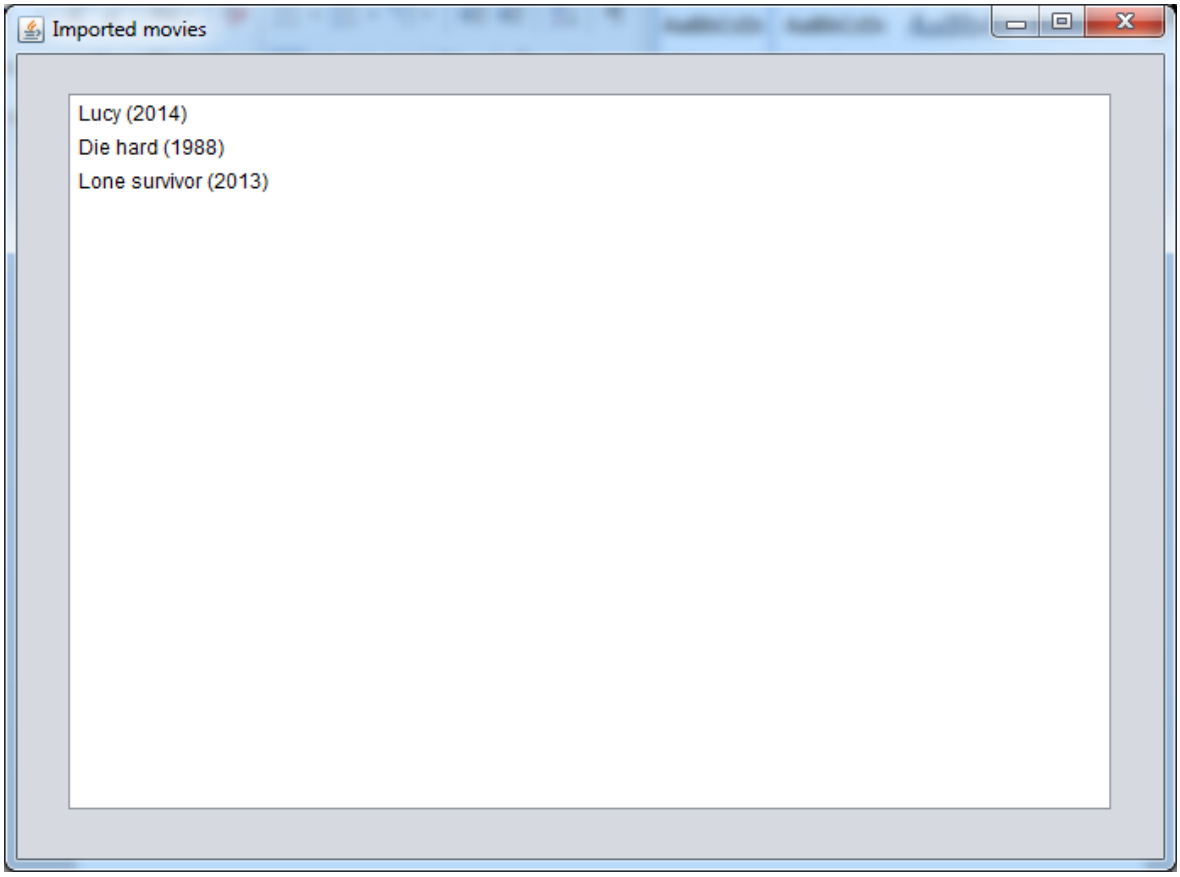
Παρακάτω φαίνεται την διαδικασία που χρησιμοποιείται για να προστεθούν καινούργιες ταινίες στη βάση



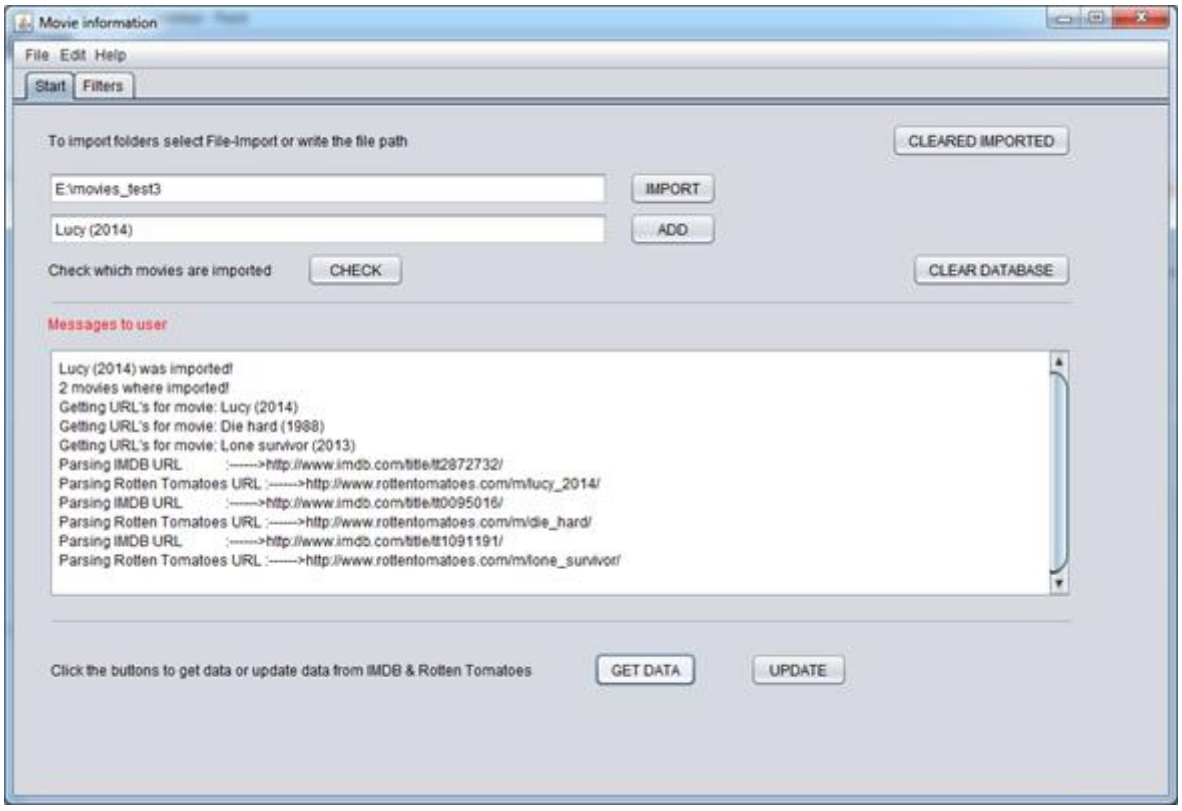
Εικόνα 16 – Εισαγωγή ταινιών



Εικόνα 17 – Προσθήκη ταινίας μέσω του μενού

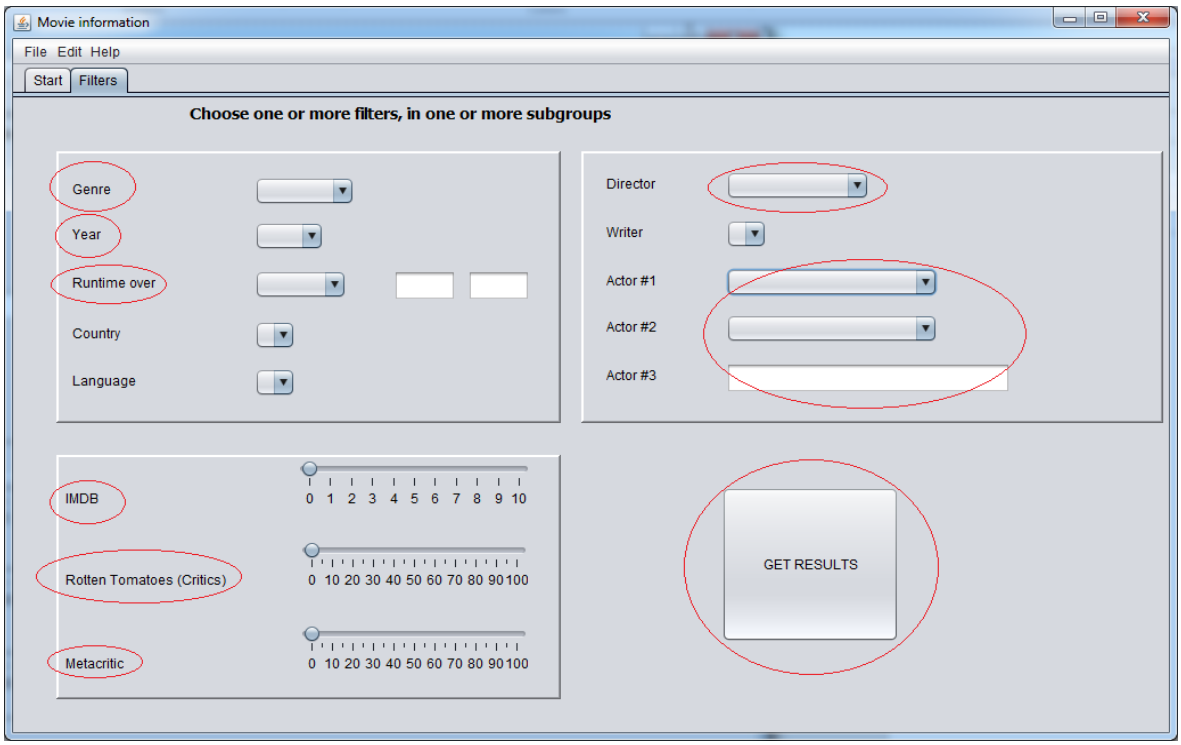


Εικόνα 18 – Έλεγχος των imported movies



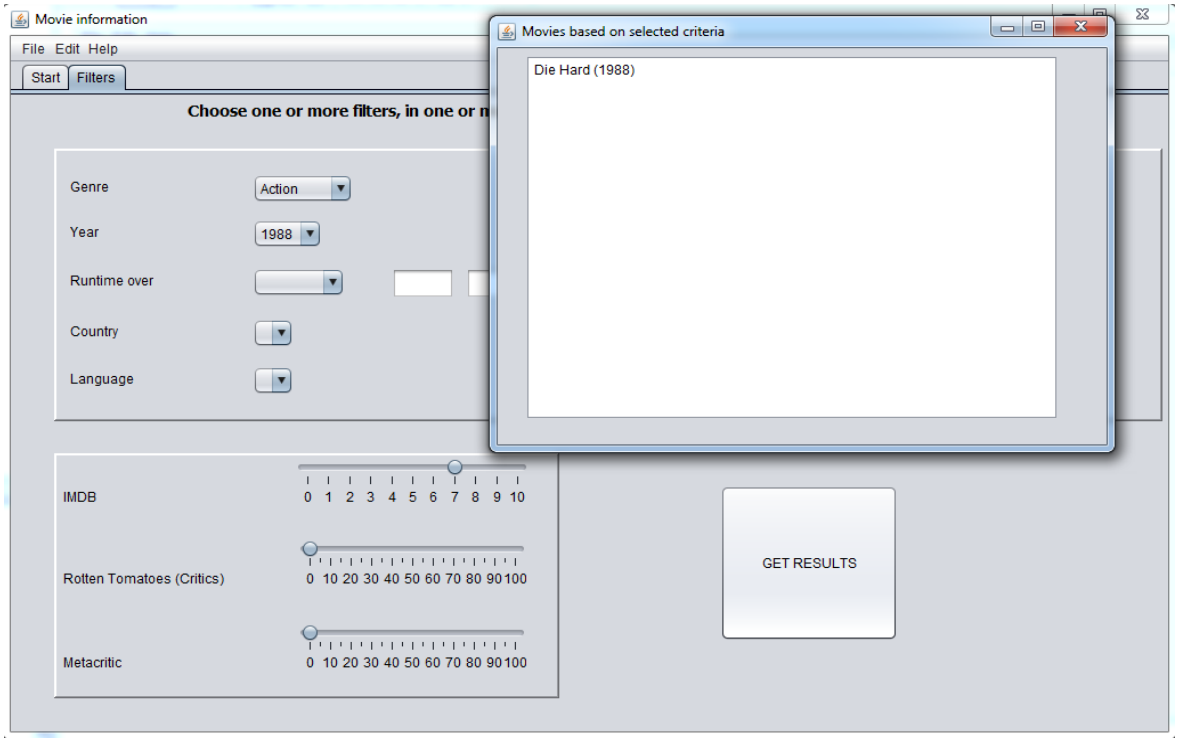
Εικόνα 19 – Parsing

Όπως φαίνεται έγινε η επιλογή ενός φάκελου με δύο ταινίες και πρόσθεση μιας τρίτης από το text field. Έπειτα συνεχίζει στο tab “Filters” όπου θα δει ότι έχει διάφορα κριτήρια για αναζήτηση ταινιών



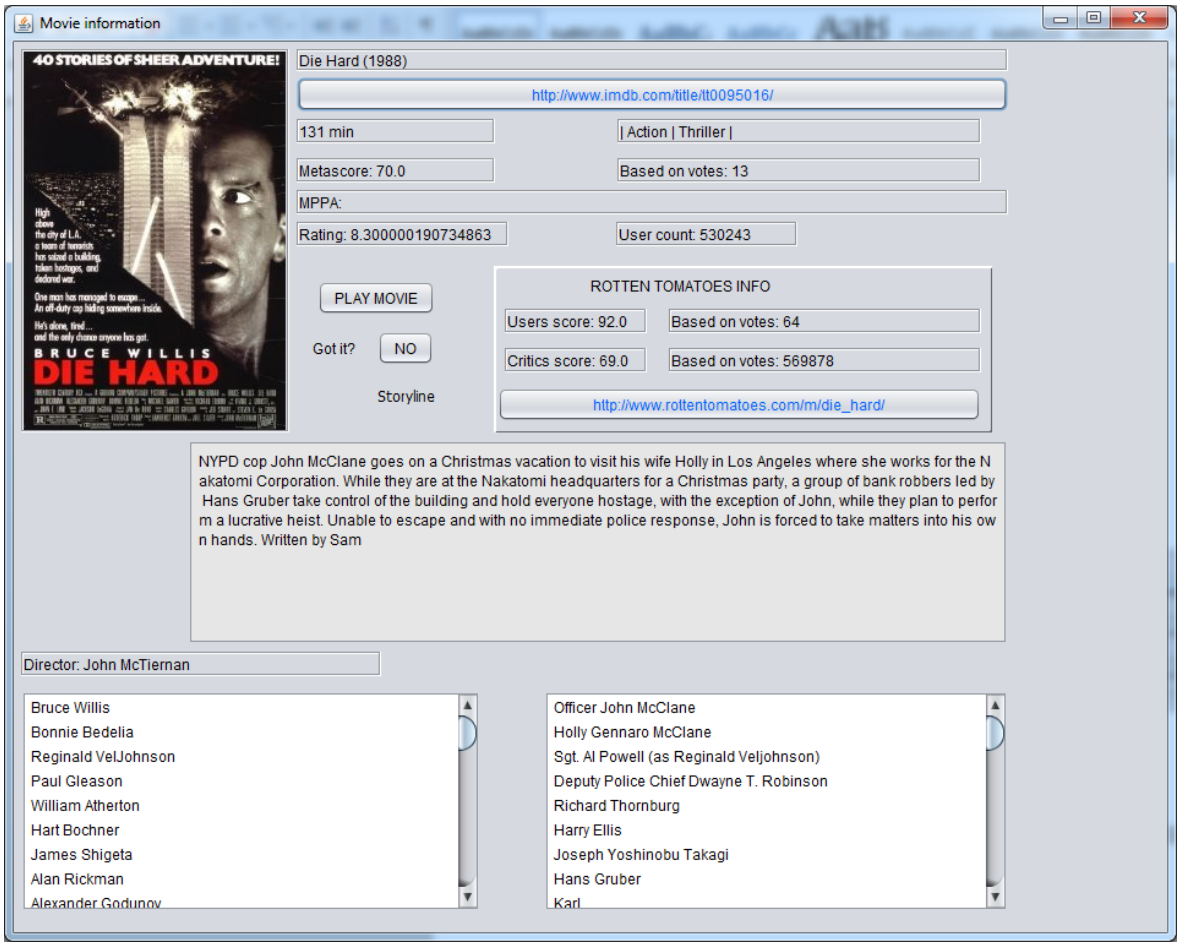
Εικόνα 20 – Επιλογή κριτηρίων

Διαλέγοντας κάποια κριτήρια εμφανίζεται ένα καινούργιο παράθυρο με αποτελέσματα ταινιών με βάση τα κριτήρια που επιλέχθηκαν (αν φυσικά υπάρχουν)



Εικόνα 21 – Αποτελέσματα

Τέλος διαλέγοντας την ταινία εμφανίζεται ένα παράθυρο με τα στοιχεία της ταινίας που επιλέχθηκαν, με βάση το IMDB & το Rotten Tomatoes



Εικόνα 22 – Πληροφορίες επιλεγμένης ταινίας

Βλέπουμε ότι υπάρχουν σύνδεσμοι στις δύο σελίδες αν χρειάζεται ο χρήστης περαιτέρω πληροφορίες. Ακόμα υπάρχει ένα check box για το αν έχει την ταινία στην κατοχή του και το button Play για το μελλοντικό media player.