ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σύστημα Λογισμικού για Ανάλυση Δεδομένων Θεατρικών Έργων

Του φοιτητή
Εμμανουηλίδη Ευάγγελου
Αρ. Μητρώου: 123934

Επιβλέπων
Μιχαήλ Σαλαμπάσης
Καθηγητής

Ημερομηνία: 14/6/2021

Τίτλος Δ.Ε: Σύστημα Λογισμικού για Ανάλυση Δεδομένων Θεατρικών Κειμένων
Κωδικός Δ.Ε: 19051
Ονοματεπώνυμο φοιτητή: Εμμανουηλίδης Ευάγγελος
Ονοματεπώνυμο εισηγητή: Σαλαμπάσης Μιχαήλ
Ημερομηνία ανάληψης Δ.Ε: 6/12/2019
Ημερομηνία περάτωσης Δ.Ε: 14/6/2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η εξέλιξη της τεχνολογίας στη σημερινή εποχή έχει επιφέρει μεγάλες αλλαγές στον τρόπο που ζούμε και αυτό οφείλεται σε μεγάλο βαθμό στα δεδομένα. Καθώς τα δεδομένα υπάρχουν παντού και συνεχώς πολλαπλασιάζονται, έχουν πραγματοποιηθεί έρευνες και προσπάθειες τιθάσευσής τους για τη βέλτιστη λήψη αποφάσεων και την ανάπτυξη ορθών στρατηγικών. Αυτό δεν αποτελεί εξαίρεση και για το χώρο του θεάτρου που είναι και παραμένει μία από τις σπουδαιότερες μορφές ψυχαγωγίας και εξελίσσεται παράλληλα με τις ανάγκες της εποχής. Η κύρια ιδέα αυτής της εργασίας είναι να αναπτύξει ένα σύστημα ανάλυσης θεατρικών έργων χρησιμοποιώντας τα δεδομένα ως βασική παράμετρο και στη συνέχεια να εξάγει αποτελέσματα και πληροφορίες με βάση αυτά. Επιπρόσθετα θα επιχειρήσει να απαντήσει σε ερωτήματα όπως τι γνώση μπορεί να αποκτηθεί από την ανάλυση δεδομένων θεατρικών έργων, πώς μπορεί αυτή να χρησιμοποιηθεί και να εξάγει ανάλογα συμπεράσματα.

Περίληψη

Το αντικείμενο της παρούσας εργασίας επιχειρεί σε πρώτο στάδιο να αναδείξει τη σημαντικότητα των δεδομένων όπως και το ρόλο που παίζουν στη σύγχρονη εποχή και σε δεύτερο στάδιο αφορά τη δημιουργία μιας εφαρμογής που επεξεργάζεται θεατρικά έργα βάσει των δεδομένων τους. Στη συνέχεια αναλύει αυτά τα δεδομένα για την παραγωγή στατιστικών αποτελεσμάτων και την εξαγωγή πληροφορίων μέσω γραφικής αναπαράστασής τους.

Για την πραγματοποίηση και υλοποίηση της συγκεκριμένης εφαρμογής ακολουθήθηκε συγκεκριμένη μεθοδολογία με χρήση κατάλληλων μέσων και λογισμικού έτσι ώστε να επιτευχθούν οι στόχοι της εργασίας που σχετίζονται άμεσα με την απόκτηση νέας γνώσης και πολύτιμων συμπερασμάτων.

Software Development for Data Analysis of theatrical Plays

Emmanouilidis Euaggelos

Abstract

The subject of this thesis is firstly to attempt to highlight the importnace of data and their role in recent years and secondly it is about the development of an application that processes theatrical plays based on their data. It then analyzes this data in order to produce statistic results and information through graphical representation of said data.

When it comes to the development of this application, certain steps were followed along with the appropriate tools and software so that the main objective of this thesis can be achieved which is directly related to the acquisition of newfound knowledge and valuable conclusions.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά τον συμφοιτητή και φίλο μου Μοσχόπουλο Βασίλη για την υποστήριξή του, ο οποίος συνέβαλε θετικά με τις ιδέες και τις γνώσεις του.

Επιλέον θέλω να ευχαριστήσω τον κύριο Σαλαμπάση Μιχαήλ για την ευκαιρία που μου έδωσε να πραγματοποιήσω αυτή την εργασία, αλλά και για τις προτάσεις και την καθοδήγησή του.

Τέλος, θέλω να ευχαριστήσω ιδιαίτερα όσους με στηρίζουν και πιστεύουν σε μένα, ιδίως σε αυτές τις δύσκολες εποχές.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
HTML	HyperText Markup Language
XML	eXtensible Markup Language
XSD	XML Schema
DTD	Document Type Definition
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσεων Δεδομένων

Κεφάλαιο 1ο: Ανάλυση Δεδομένων

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει εισαγωγή στην έννοια των δεδομένων, τα οποία έχουν κυρίαρχο ρόλο στην καθημερινή μας ζωή και πώς μπορούμε να αποκτήσουμε ένα λογικό νόημα από αυτά. Θα γίνει αναφορά στις δύο από τις μεγαλύτερες επιστήμες που χειρίζονται τα δεδομένα, την Επιστήμη των Δεδομένων και την Ανάλυση Δεδομένων και θα εξηγήσουμε τη φιλοσοφία και τις δυνατότητες της καθεμίας. Κλείνοντας, θα αναφέρουμε την έννοια της ανάλυσης του θεάτρου και θα μιλήσουμε για το κυρίως θέμα και τους στόχους της Π.Ε. δηλαδή την ανάλυση δεδομένων θεατρικών έργων.

1.2 Τι είναι Δεδομένα

Τα δεδομένα είναι μια μη αξιολογημένη συλλογή από στοιχεία τα οποία λαμβάνουμε από το περιβάλλον και βρίσκονται παντού γύρω μας. Μπορούν να διακριθούν σε 2 κατηγορίες: Τα ποιοτικά δεδομένα (qualitative data) και τα ποσοτικά δεδομένα (quantitative data). Τα ποιοτικά δεδομένα αφορούν χαρακτηριστικά και περιγραφές που δεν μπορούν να μετρηθούν με αριθμούς, αλλά μπορούν να αξιολογηθούν με άλλους τρόπους όπως με τις αισθήσεις μας, για παράδειγμα μπορούμε να διακρίνουμε χρώματα με την όραση, να ξεχωρίσουμε οσμές με την όσφρηση, κλπ. Γενικά χρησιμοποιούμε τα ποιοτικά δεδομένα για να κατηγοριοποιήσουμε αντικείμενα και καταστάσεις σε ξεχωριστές ομάδες. Αντίστοιχα τα ποσοτικά δεδομένα είναι αυτά που μπορούν να μετρηθούν με αριθμούς όπως οι διαστάσεις, τα μεγέθη, η θερμοκρασία, κ.ο.κ. και μπορούν να χωριστούν σε 2 υποκατηγορίες: Τα συνεχή δεδομένα (continuous data) και τα διακριτά δεδομένα (discrete data). Τα συνεχή δεδομένα είναι αυτά που μπορούν να αναλυθούν περαιτέρω σε μικρότερα επίπεδα για πάραδειγμα το 1 μέτρο μπορεί να αναλυθεί σε δέκατα, εκατοστά, χιλιοστά, κλπ. Από την άλλη τα διακριτά δεδομένα δεν μπορούν να αναλυθούν και συνήθως είναι ακέραιοι αριθμοί.

Τα δεδομένα που λαμβάνουμε μπορούμε να τα πάρουμε από διαφορετικές πηγές. [1] Οι Πρωτογενείς πηγές (Primary Sources) θεωρούνται αυτές που αφορούν δημοσιεύματα από μελέτες και έρευνες και το περιεχόμενό τους δεν έχει υποστεί κάποιου είδους επεξεργασία αλλά παραθέτουν τις απόψεις και ευρήματα των ερευνητών, όπως επιστημονικά άρθρα και έγγραφα. Οι Δευτερογενείς πηγές (Secondary Sources) είναι αυτές που βασίζονται στις πρωτογενείς πηγές αλλά έχουν υποστεί επεξεργασία και έχουν τροποποιηθεί. Σε αυτή την κατηγορία ανήκουν το Διαδίκτυο, εκπαιδευτικά βιβλία, άρθρα εφημερίδων, κλπ.

Τα δεδομένα από μόνα τους δεν αρκούν για να αποκτήσουμε ένα νόημα από αυτά, απλώς υπάρχουν. Γι’ αυτό το λόγο είναι απαραίτητη η αξιολόγηση, η οργάνωση, η επεξεργασία και η ανάλυσή τους για να αποκτήσουμε αυτό το νόημα που ονομάζεται πληροφορία. Με άλλα λόγια η πληροφορία είναι μια οργανωμένη και επεξεργασμένη μορφή δεδομένων απ’ την οποία εξάγεται ένα λογικό νόημα και είναι απαραίτητη για τον τρόπο που ζούμε σε όλους τους τομείς και πτυχές.

1.3 Επιστήμη Δεδομένων

Με τον όρο της Επιστήμης Δεδομένων (Data Science) αναφερόμαστε στην επιστήμη που ασχολείται με αδόμητα ή οργανωμένα δεδομένα και εξάγει πληροφορίες και γνώση μέσα από αυτά.

Επικεντρώνεται στο να αποκτήσουμε νέες χρήσιμες πληροφορίες επεξεργάζοντας μεγάλες ποσότητες δεδομένων (Big Data) και συνεργάζεται με κλάδους επιστημών όπως η Πληροφορική (Computer Science), η Στατιστική (Statistics), η Μηχανική Μάθηση (Machine Learning) και η Εξόρυξη Δεδομένων (Data Mining). Η επιστήμη Δεδομένων προσπαθεί να ανακαλύψει μοτίβα και παρατηρήσεις που μπορεί να είναι πολύ σημαντικά για την ενίσχυση άλλων επιστημών όπως η Τεχνητή Νοημοσύνη, η Ιατρική και ολόκληρος ο κόσμος των επιχειρήσεων.

Όλες αυτές οι επιστήμες που αναφέραμε πιο πάνω πρέπει να συνεργαστούν με κάποιο τρόπο για να βγουν σωστά συμπεράσματα. Μια τυπική διαδικασία στην επιστήμη των Δεδομένων μπορεί να χωριστεί στα εξής βήματα:

- Ορισμός του προβλήματος: Πρώτα απ' όλα πρέπει να αναγνωρίσουμε ακριβώς το πρόβλημα που πρόκειται να αντιμετωπίσουμε. Ουσιαστικά πρέπει να θέσουμε τις σωστές ερωτήσεις έτσι ώστε θέσουμε ταυτόχρονα και τους σωστούς στόχους.
- Συλλογή των αδόμητων δεδομένων: Προφανώς για να λύσουμε ένα πρόβλημα πρέπει να δουλέψουμε με δεδομένα. Αυτό το κομμάτι της διαδικασίας αφορά τη λήψη των κατάλληλων δεδομένων που θα μας φέρουν πιο κοντά στη λύση του προβήματος.
- Επεξεργασία και καθαρισμός των δεδομένων: Σε αυτό το βήμα απαιτείται καθαρισμός των δεδομένων που αντί να συμβάλλουν στη λύση του προβλήματος, να το κάνουν πιο σύνθετο και πολύπλοκο, για παράδειγμα κενές ή λανθασμένες εγγραφές δεδομένων.
- Εξερεύνηση των δεδομένων: Πρόκειται για την κατανόηση των δεδομένων που θα χρησιμοποιηθούν για ανάλυση και την εύρεση σχέσεων και μοτίβων που μπορούν να προκύψουν από αυτά.
- Ανάλυση και Μοντελοποίηση δεδομένων: Εδώ τα δεδομένα αναλύονται με βάση την εξερεύνηση του προηγούμενου βήματος και διαφορετικά μοντέλα εκπαιδεύονται πάνω στα δεδομένα με χρήση τεχνικών Μηχανικής Μάθησης. Ουσιαστικά εξετάζουμε κατά πόσο καλά τα μοντέλα γενικεύουν τα δεδομένα. Σκοπός είναι να βρούμε το ιδανικό μοντέλο για το πρόβλημα που μας απασχολεί.
- Έλεγχος και Διάθεση: Το επιλεγμένο μοντέλο ελέγχεται περαιτέρω σε ειδικό περιβάλλον και αν φανεί αποτελεσματικό, διατίθεται για γενική χρήση.

Ανακεφαλαιώνοντας, η επιστήμη των Δεδομένων απασχολεί πολλές επιστήμες μαζί προκειμένου να διαχειριστεί κατά τον καλύτερο δυνατό τρόπο τα δεδομένα και επακόλουθα να παρθούν οι καλύτερες δυνατές αποφάσεις στον εκάστοτε τομέα. Όλες οι μεγάλες επιχειρήσεις όπως η Google, η Amazon, το Facebook και το Netflix χρησιμοποιούν τα δεδομένα για να βελτιστοποιήσουν τη δράση τους αλλά και να αυξήσουν το κέρδος τους.

1.4 Ανάλυση Δεδομένων

Η Ανάλυση Δεδομένων (Data analysis) είναι συγγενική επιστήμη με την Επιστήμη των Δεδομένων, αλλά η διαφορά ανάμεσα στις δύο είναι ότι η ανάλυση Δεδομένων επικεντρώνεται στην επεξεργασία και ανάλυση οργανωμένων δεδομένων και την παραγωγή στατιστικών ενώ η επιστήμη Δεδομένων αφορά τη διαχείριση αδόμητων δεδομένων και πώς μπορεί να τα χειριστεί με τον καλύτερο τρόπο, χωρίς να επικεντρώνεται ιδιαίτερα στην ανάλυσή τους. Με άλλα λόγια η επιστήμη Δεδομένων θέτει τις σωστές ερωτήσεις και η ανάλυση Δεδομένων θέτει τις σωστές απαντήσεις.

Η χρησιμότητα της ανάλυσης Δεδομένων προκύπτει από την απόκτηση νέας γνώσης και πληροφορίας η οποία συμβάλλει σημαντικά στη λήψη σωστών αποφάσεων (όπως και στην επιστήμη Δεδομένων) και ταυτόχρονα στη βελτίωση του αντικειμένου για το οποίο γίνεται η ανάλυση. Βασικό στοιχείο στην Ανάλυση Δεδομένων είναι το Data Frame. Το Data Frame είναι μια δυσδιάστατη δομή δεδομένων που μοιάζει με έναν πίνακα που διαθέτει γραμμές και στήλες. Η πρώτη γραμμή μπορεί να αποτελείται από τους τίτλους των στηλών του Data Frame και ονομάζεται header. Οι στήλες μπορούν να έχουν διαφορετικούς τύπους δεδομένων μεταξύ τους π.χ. ακέραιους ή συμβολοσειρές αλλά κάθε στήλη πρέπει να έχει τον ίδιο τύπο δεδομένων. Ακριβώς επειδή τα δεδομένα στο Data Frame είναι δομημένα και οργανωμένα, μπορούν πολύ εύκολα μετά να επεξεργαστούν και να αναλυθούν.

Η Ανάλυση Δεδομένων αφορά καθαρά την απόκτηση νέας γνώσης όπως αναφέραμε. Ο πιο συνηθισμένος τρόπος απόκτησης αυτής τη γνώσης είναι μέσω της περιγραφικής στατιστικής και των μαθηματικών, για παράδειγμα υπολογίζοντας τιμές όπως η μέση τιμή και η τυπική απόκλιση. Μερικές φορές όμως οι αριθμητικές τιμές μπορεί να μην είναι τόσο αποτελεσματικές ή να είναι αρκετά πολύπλοκες για να περιγράψουν τα ευρύματα των αποτελεσμάτων. Ακριβώς γι' αυτό η οπτικοποίηση των δεδομένων είναι ο καλύτερος τρόπος αναπαράστασης και περιγραφής τους καθώς τα σχήματα και οι εικόνες "μιλάνε" καλύτερα από τους αριθμούς. Μια τυπική διαδικασία ανάλυσης Δεδομένων περιλαμβάνει τις εξής ενέργειες οι οποίες μπορούν να επαναλαμβάνονται μέχρι να φτάσουμε στο ιδανικό αποτέλεσμα:

- Ανάλυση Δεδομένων: Εφόσον τα δεδομένα έχουν μια δομημένη και οργανωμένη μορφή, επεξεργάζονται κατάλληλα και εξάγονται ανάλογα αποτελέσματα.
- Οπτικοποίηση: Τα αποτελέσματα αυτά οπτικοποιούνται με τη μορφή γραφημάτων και διαγραμμάτων για την κατανόηση και εμπέδωση των ευρημάτων και παράλληλα αποκτάται νέα γνώση και εξάγονται συμπεράσματα.
- Επαναξιολόγηση και επανασχεδίαση: Το αντικείμενο για το οποίο γίνεται η ανάλυση αξιολογείται εκ νέου με βάση τη γνώση του προηγούμενου βήματος, επανασχεδιάζεται και γίνεται εκ νέου ανάλυση με γνώμονα τη βελτιστοποίησή του.

Με βάση τα παραπάνω η ανάλυση Δεδομένων αποσκοπεί στη βελτιστοποίηση ενός συστήματος ή αντικειμένου χρησιμοποιώντας δεδομένα στην κατάλληλη μορφή και εξάγονται συμπεράσματα για τη λήψη καλύτερων αποφάσεων. Η ανάλυση Δεδομένων χρησιμοποιείται ευρύτατα σε κλάδους όπως η Ιατρική για την πρόληψη ασθενειών, στη Μετεωρολογία για την πρόβλεψη του καιρού και στον κλάδο των επιχειρήσεων για την καλύτερη παροχή υπηρεσιών και αγαθών και τη μεγιστοποίηση του κέρδους.

1.4.1 Ανάλυση Δεδομένων Θεατρικών Έργων

Τα δεδομένα και η τεχνολογία έχουν εισχωρήσει για τα καλά στην καθημερινότητά μας. Χρησιμοποιούμε δεδομένα για να επικοινωνήσουμε, να ενημερωθούμε αλλά και να ψυχαγωγηθούμε. Ως προς την ψυχαγωγία, το θέατρο παραμένει σήμερα ένας από τους μεγαλύτερους και σημαντικότερους κλάδους ψυχαγωγίας και συνεχώς εξελίσσεται. Σύμφωνα με το άρθρο [4] η ανάλυση θεάτρου (theatre analytics) είναι η ανάλυση των δεδομένων θεάτρου είτε αυτό αφορά την ιστορία του θεάτρου, είτε αφορά το δράμα ή τη σκηνοθεσία. Μέσα από μια τέτοια προσέγγιση, τίθεται το ερώτημα του τι πληροφορίες και τι γνώση μπορούμε να εξάγουμε και πώς μπορεί να χρησιμοποιηθεί κατάλληλα αυτή η γνώση.

Στην παρούσα Π.Ε. θα ασχοληθούμε ιδιαίτερα με την ανάλυση δεδομένων θεατρικών έργων και την εξαγωγή στατιστικών και οπτικών γραφημάτων. Επιπλέον θα εξετασθεί κατά πόσο μπορούν να εξαχθούν ασφαλή συμπεράσματα αναλύοντας παραμέτρους όπως τις πράξεις, τις σκηνές, τους χαρακτήρες και τους διαλόγους ενός έργου και αντίστοιχα πώς μπορούν να αξιοποιηθούν αυτά τα συμπεράσματα. Η μεθοδολογία της ανάλυσης που θα ακολουθηθεί θα περιγραφτεί αναλυτικά σε επόμενο κεφάλαιο με χρήση των κατάλληλων εργαλείων λογισμικού.

1.5 Επίλογος

Σε αυτό το κεφάλαιο έγινε αναφορά στη χρησιμότητα των δεδομένων και στις κατηγορίες στις οποίες μπορούμε να τα διακρίνουμε και αναφέραμε τη σπουδαιότητα της πληροφορίας που προκύπτει από την επεξεργασία και ανάλυσή τους. Μιλήσαμε για τις δύο βασικές επιστήμες που χειρίζονται τα δεδομένα, το πώς λειτουργούν και πώς διαφέρουν μεταξύ τους. Τέλος, θέσαμε τους στόχους της Π.Ε. όσον αφορά την ανάλυση δεδομένων θεατρικών έργων και θα επιχειρήσουμε να βγάλουμε λογικά συμπεράσματα μέσα από αυτήν.

Κεφάλαιο 2ο: Τεχνολογίες και Λογισμικό Υλοποίησης

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράψουμε τις τεχνολογίες που θα χρησιμοποιηθούν για την υλοποίηση της εφαρμογής και θα αναλύσουμε βασικές έννοιες και ορισμούς. Θα μιλήσουμε για τη γλώσσα προγραμματισμού Java και τα χαρακτηριστικά της, όπως και για το γραφικό περιβάλλον Swing πάνω στο οποίο θα βασιστεί η υλοποίηση της εφαρμογής της Π.Ε. και θα γίνει αναλυτική περιγραφή των τεχνολογιών που θα συναντήσουμε σε αυτήν. Για την ακρίβεια, θα αναφέρουμε τις γλώσσες σήμανσης HTML και XML και το ρόλο της χρησιμότητάς τους, θα μιλήσουμε για την τεχνική συλλογής δεδομένων ιστού Web scraping και θα γίνει αναφορά στα Συστήματα Διαχείρισης Βάσεων Δεδομένων και πώς μας βοηθούν να αποθηκεύσουμε και να οργανώσουμε τα δεδομένα μας. Τέλος θα μιλήσουμε και για τις σχετικές βιβλιοθήκες που θα χρησιμοποιηθούν για την επίτευξη των στόχων της εφαρμογής.

2.2 Java

Η Java αποτελεί μια υψηλού επιπέδου, αντικειμενοστρεφής γλώσσα προγραμματισμού η οποία αναπτύχθηκε από την Sun Microsystems. Ο κύριος σκοπός της γλώσσας, η οποία βασίστηκε πάνω στη C++, ήταν η δημιουργία μιας πλατφόρμας για την ανάπτυξη λογισμικού σε μικροσυσκευές. Το 2010 η Sun Microsystems εξαγοράστηκε από την εταιρία λογισμικού Oracle η οποία πλέον κατέχει τα πνευματικά δικαιώματα. Αυτή τη στιγμή, η τελευταία έκδοση της γλώσσας είναι η Java SE 16.0.1 και παραμένει μια απ' τις πιο δημοφιλείς γλώσσες με περισσότερους από 12 εκατομμύρια προγραμματιστές που τη χρησιμοποιούν.

2.2.1 Χαρακτηριστικά της Java

Ένα από τα σημαντικότερα πλεονεκτήματα της Java σε σύγκριση με άλλες γλώσσες είναι ότι έχει σχεδιαστεί για να τρέχει ανεξάρτητα από το λειτουργικό σύστημα, προσφέροντας μεγαλύτερη ευελιξία και ανεξαρτησία στον προγραμματιστή, δηλαδή είναι φορητή. Με άλλα λόγια ο κώδικας σε Java μπορεί να τρέξει το ίδιο σε διαφορετικά λειτουργικά συστήματα όπως π.χ. Windows, Linux και Unix χωρίς την ανάγκη να γίνει εκ νέου μεταγλώττιση του κώδικα (recompilation) για κάθε ένα από αυτά.

Η Java είναι μια σχετικά απλή γλώσσα προγραμματισμού, καθώς σχεδιάστηκε για να είναι όσο το δυνατόν πιο κατανοητή και επομένως δεν φέρει τη χρήση δεικτών (pointers) όπως η συγγενική της C++, ενώ η διαχείριση της μνήμης γίνεται αυτόματα με τη διεργασία συλλογής "απορρημάτων" γνωστή ως Garbage Collector. Πρόκειται για την απελευθέρωση μνήμης όταν δεν χρησιμοποιούνται αντικείμενα και δεν υπάρχουν αντίστοιχες αναφορές στη σωρό μνήμης (heap) και με αυτό τον τρόπο ο προγραμματιστής δεν έχει λόγο να ανησυχεί για το αν θα χρειαστεί να ελευθερώσει ο ίδιος χώρο στη μνήμη ή για σφάλματα δεικτών. Εάν κληθεί μία μέθοδος για κάποιο ανύπαρκτο αντικείμενο, τότε ρίχνεται μία εξαίρεση που δηλώνει ότι το αντικείμενο δεν αντιστοιχεί πουθενά στη μνήμη (Nullpointer exception).

Επίσης η Java είναι αντικειμενοστρεφής γλώσσα και λειτουργεί σύμφωνα με το μοντέλο αντικειμενοστρεφούς προγραμματισμού, δηλαδή ο χειρισμός δεδομένων και διαδικασιών βασίζεται σε μια αυτόνομη οντότητα με δικά της χαρακτηριστικά που ονομάζεται αντικείμενο. Τα αντικείμενα αποτελούν στιγμιότυπα μιας ευρύτερης μονάδας που ονομάζεται κλάση. Οι κλάσεις χρησιμοποιούνται για την οργάνωση, την κωδικοποίηση και τον ορισμό νέων αντικειμένων. Οι αντικειμενοστρεφείς γλώσσες προγραμματισμού υποστηρίζουν συγκεκριμένες αρχές [2]:

- Η πρώτη ιδιότητα αφορά την ενθυλάκωση δεδομένων (data encapsulation) δηλαδή την ιδιότητα των κλάσεων να προστατεύουν τα δεδομένα τους από άλλα σημεία κώδικα και να ελέγχεται η πρόσβαση σε αυτά.
- Η αφαίρεση δεδομένων (data abstraction), είναι η ιδιότητα των κλάσεων να αναπαριστούν ένα πολυσύνθετο πρόβλημα μόνο με τις απαραίτητες μεταβλητές που μας ενδιαφέρουν και να αγνοούν σκόπιμα τις υπόλοιπες.
- Η κληρονομικότητα (inheritance), αποτελεί τη δυνατότητα επέκτασης μιας αφηρημένης κλάσης σε μια πιο εξειδικευμένη, θυγατρική της με πιο συγκεκριμένα χαρακτηριστικά και η επαναχρησιμοποίηση μεθόδων και ιδιοτήτων της γονικής κλάσης.
- Τέλος, η έννοια του πολυμορφισμού (polymorphism) που αφορά την ομοιόμορφη διαχείριση αντικειμένων της ίδιας ιεραρχίας.

Ένα άλλο χαρακτηριστικό που περιγράφει την Java είναι το γεγονός ότι είναι μεταγλωττιζόμενη και ερμηνευόμενη γλώσσα καθώς ο πηγαίος κώδικας μεταγλωττίζεται σε ενδιάμεσο κώδικα (bytecode) από το περιβάλλον εκτέλεσης ή αλλιώς Java Runtime Environment (JRE) και στη συνέχεια ερμηνεύεται από την εικονική μηχανή σε εκτελέσιμο κώδικα.

Η Java είναι strongly typed γλώσσα, δηλαδή ελέγχει αυστηρά τον ορισμό και τη δήλωση των τύπων δεδομένων οι οποίοι θα πρέπει να έχουν δηλωθεί εξαρχής. Επομένως δεν μπορεί να χρησιμοποιηθεί μια μεταβλητή χωρίς πρώτα να δηλωθεί ο τύπος της.

Τέλος, η Java είναι δυναμική έτσι ώστε να προσαρμόζεται σε διαφορετικά εξελισσόμενα περιβάλλοντα. Η γλώσσα και οι βιβλιοθήκες της ενημερώνονται συνεχώς με την προσθήκη νέου λογισμικού σύμφωνα με τα νέα τεχνολογικά δεδομένα.

2.2.2 Αρχιτεκτονική

Προκειμένου να τρέξουμε ένα πρόγραμμα γραμμένο σε Java σε μία μηχανή, πρέπει πρώτα να έχουμε εγκαταστήσει το κατάλληλο Java Runtime Environment (JRE) σε αυτήν. Όπως αναφέραμε πιο πριν ο κώδικας ενός προγράμματος σε Java πριν εκτελεστεί, μεταγλωττίζεται σε ενδιάμεσο κώδικα και μετά μετατρέπεται σε γλώσσα μηχανής, δηλαδή ο κώδικας είναι πλέον κατανοητός από αυτήν και μπορεί να τον εκτελέσει. Αυτή την εργασία αναλαμβάνει η Εικονική Μηχανή της Java γνωστή και ως Java Virtual Machine (JVM) και αποτελεί βασικό κομμάτι της αρχιτεκτονικής της γλώσσας. Η Εικονική Μηχανή λειτουργεί σαν δίαυλος επικοινωνίας ανάμεσα στον προγραμματιστή και τον υπολογιστή καθώς είναι υπεύθυνη για την εγκυρότητα του ενδιάμεσου κώδικα και δεν επιτρέπει την εκτέλεση ύποπτου ή κακόβουλου κώδικα που θα μπορούσε πιθανώς να βλάψει τη μηχανή. Η Εικονική Μηχανή είναι επιπλέον υπεύθυνη και για άλλες σημαντικές λειτουργίες όπως η φόρτωση των κλάσεων μέσω του Class Loader και τη βελτιστοποίηση διαχείρισης της μνήμης, χωρίζοντάς την σε κατηγορίες για την εκτέλεση των προγραμμάτων.

2.3 Java Swing

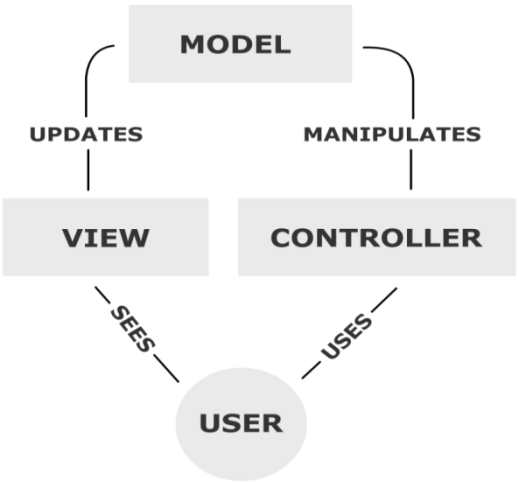
Η Swing αποτελεί μια εργαλειοθήκη της Java για τη δημιουργία εφαρμογών σε γραφικό περιβάλλον χρήστη ή αλλιώς γραφική διεπαφή χρήστη (Graphical User Interface, GUI). Το γραφικό περιβάλλον επιτρέπει σε έναν χρήστη να αλληλεπιδρά με μια ηλεκτρονική συσκευή χρησιμοποιώντας εικονίδια, κουμπιά και ενέργειες όπως την εισαγωγή κειμένου ως είσοδο (input) σε μια εφαρμογή. Οι γραφικές διεπαφές χρήστη αναδείχθηκαν και άρχισαν να κερδίζουν έδαφος μετά την πτώση σε δημοτικότητα των διεπαφών γραμμής εντολών που απαιτείται η εκτέλεση εντολών καθαρά στο πληκτρολόγιο.

Η Swing αναπτύχθηκε για να καλύψει ανάγκες για τις οποίες το προγενέστερο Abstract Windows Toolkit (AWT) ήταν ανεπαρκές. Η Swing επίσης παρέχει μια ευρεία γκάμα από πιο ευέλικτα και προηγμένα στοιχεία (components) σε σχέση με το AWT όπως λίστες, δέντρα, πίνακες, panels που περιέχουν καρτέλες (tabbed panels), scroll panes για κύλιση και άλλα. Επίσης σε αντίθεση με το AWT, τα στοιχεία της Swing είναι γραμμένα αποκλειστικά σε Java και επομένως δεν εξαρτώνται από την ίδια την πλατφόρμα.

2.3.1 Model View Controller

Το Model View Controller (MVC) είναι ένα μοντέλο αρχιτεκτονικής λογισμικού με σκοπό τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Σύμφωνα με το μοντέλο μια εφαρμογή χωρίζεται σε τρία επιμέρους κομμάτια τα οποία είναι το Controller, το Model και το View. Το καθένα από αυτά έχει το δικό του ρόλο και σε συνεργασία μαζί επιτρέπουν την καλύτερη προσπέλαση λογισμικού και ταυτόχρονα λιγότερη πολυπλοκότητα.

Πιο συγκεκριμένα, το Controller δέχεται πληροφορίες από το χρήστη σε μορφή αιτήματος (request) και στέλνει αντίστοιχες εντολές στο Model και επίσης μπορεί να ενημερώνει την κατάσταση του μοντέλου. Το Model δέχεται πληροφορίες από το Controller και είναι υπεύθυνο για τις ενέργειες που αφορούν τα δεδομένα, για πάραδειγμα μπορεί να αλληλεπιδρά με μια βάση δεδομένων και να ενημερώνει δεδομένα σε αυτή. Στη συνέχεια το Model ενημερώνει το Controller για την εκάστοτε αλλαγή στα δεδομένα και τέλος το View αναλαμβάνει την αναπαράσταση των δεδομένων του Model με γραφικό τρόπο, το οποίο είναι τελικά αυτό που βλέπει ο χρήστης. Στην Εικόνα 2.1 παρακάτω φαίνεται το μοντέλο MVC.



Εικόνα 2.1: Το μοντέλο MVC [5]

2.3.2 Βασικά Swing Στοιχεία

- JFrame: Το βασικότερο στοιχείο μέσα σε μια εφαρμογή Swing είναι το JFrame. Το αντικείμενο JFrame είναι πάντα το πρώτο container πάνω στο οποίο προσθέτουμε τα υπόλοιπα στοιχεία. Καθώς δημιουργούμε ένα JFrame μπορούμε να επιλέξουμε ένα layout για αυτό. Υπάρχουν διάφορες μέθοδοι για επεξεργασία του JFrame με τις οποίες μπορούμε να αλλάξουμε τις διαστάσεις του σε μήκος και πλάτος και να ορίσουμε τη θέση του σε σχέση με την οθόνη.
- JPanel: Μετά το JFrame, το αμέσως επόμενο βασικότερο component το οποίο προστίθεται μέσα στο JFrame είναι το JPanel. Το JPanel είναι αυτό στο οποίο προσθέτονται πάνω τα υπόλοιπα στοιχεία και διαθέτει ένα διαχειριστή τοποθέτησης στοιχείων (Layout Manager) για τη στοίχιση και οργάνωση των εσωτερικών στοιχείων. Μπορούμε να επεξεργαστούμε εμφανισιακά ένα JPanel προσθέτοντάς του ένα border ή αλλάζοντας τις διαστάσεις του απλά με τον κέρσορα του ποντικιού.
- JScrollPane: Εάν θέλουμε η εφαρμογή μας να διαθέτει μηχανισμό κύλισης (scrolling) εφόσον το περιεχόμενό μας για παράδειγμα είναι μεγαλύτερο από τις διαστάσεις του JPanel και δεν χωράει, το κατάλληλο στοιχείο για αυτή την περίπτωση είναι το JScrollPane. Συνήθως προστίθεται μέσα στο JPanel ή προσθέτονται μέσα σε αυτό στοιχεία που περιέχουν μορφή κειμένου.
- JSplitPane: Ένα άλλο χρήσιμο στοιχείο είναι το JSplitPane που λειτουργεί ως διαχωριστικό ανάμεσα σε άλλα δύο στοιχεία και έχει την ικανότητα μετακίνησης. Έτσι ο χρήστης μπορεί να αλλάζει διαστάσεις ανάμεσα σε αυτά τα δύο στοιχεία είτε οριζόντια είτε κάθετα.
- JTextArea, JTextPane & JEditorPane: Για τη συγγραφή κειμένου διατίθονται αυτά τα τρία στοιχεία. Από τα τρία αυτά, το JTextArea είναι το μόνο που δεν μπορεί να δεχθεί επεξεργασμένο κείμενο (π.χ bold, με υπογράμμιση και με χρώματα) παρά μόνο σε κανονική μορφή. Το JTextPane είναι πιο ευέλικτο και μπορεί να περιέχει εκτός από επεξεργασμένο κείμενο και άλλα στοιχεία όπως εικόνες και τέλος το JEditorPane είναι αυτό με τις μεγαλύτερες δυνατότητες και μπορεί να χρησιμοποιηθεί για αναπαράσταση πληροφορίας όπως φαίνεται σε έναν περιηγητή. Στις περισσότερες περιπτώσεις είναι θεμιτό και τα τρία στοιχεία να χρησιμοποιούνται μέσα σε ένα JScrollPane, για διευκόλυνση ανάγνωσης και αισθητικής.
- JLabel: Το στοιχείο JLabel αναπαριστά μία ετικέτα στην οποία μπορεί ο χρήστης να δώσει μία ονομασία και να την τοποθετήσει μέσα στο JPanel σε τέτοιο σημείο ώστε να αποκτά εμφανισιακά μια ιδιότητα, για παράδειγμα μπορεί να χρησιμοποιηθεί ως τίτλος για ένα άλλο στοιχείο πάνω από αυτό ή δίπλα από αυτό.
- JButton: Πέρα από το JLabel, εάν ο χρήστης επιθυμεί να αλληλεπιδράσει με την εφαρμογή το συνηθέστερο στοιχείο σε αυτή την περίπτωση είναι το JButton. Μπορεί να προστεθεί οπουδήποτε μέσα στο JPanel και να στοιχιστεί ανάλογα με τα υπόλοιπα στοιχεία που βρίσκονται ήδη μέσα (όπως και το JLabel) χάρη στο Layout Manager. Το κουμπί JButton από μόνο του δεν χρησιμεύει σε κάτι, επειδή κάθε φορά που ο χρήστης το πατάει θα πρέπει να ακολουθεί και μια συγκεκριμένη ενέργεια. Ακριβώς γι' αυτό τα κουμπιά JButton συνοδεύονται πάντα μαζί με κάποιο γεγονός (Swing event). Επίσης για παρόμοιες λειτουργίες, παραλλαγές του JButton είναι τα RadioButton που έχει μόνο δύο καταστάσεις (επιλογής ή όχι), το JToggleButton το οποίο παραμένει πατημένο εφόσον έχει πατηθεί και το ButtonGroup που είναι ένα σετ από κουμπιά.
- JTextField: Τέλος, αντίστοιχη φιλοσοφία με τα κουμπιά έχει το JTextField στο οποίο ο χρήστης μπορεί να εισάγει κάποιου είδους πληροφορία και πατώντας για παράδειγμα το Enter να ενεργοποιηθεί ένα γεγονός. Με αυτό τον τρόπο συλλέγεται η πληροφορία που εισήγαγε ο χρήστης και με βάση αυτή, εκτελείται κάποια λειτουργία.

2.3.3 Swing Events

Οι εφαρμογές σε Swing βασίζονται σε γεγονότα (events). Τα αντικείμενα μιας εφαρμογής μπορούν να επικοινωνούν μεταξύ τους μέσω των γεγονότων. Ένα γεγονός συμβαίνει όταν ο χρήστης της εφαρμογής εκτελέσει μια ενέργεια η οποία επακόλουθα πυροδοτεί το γεγονός, π.χ. με το πάτημα ενός κουμπιού. Ωστόσο πιο σπάνια, γεγονότα μπορούν να παραχθούν και από άλλα συμβάντα όπως ένα χρονόμετρο ή εναν Window manager. Τα γεγονότα αποτελούν αντικείμενα της κλάσης java.util.EventObject και των απογόνων της.

Στο μοντέλο γεγονότων υπάρχουν τρεις οντότητες: Το Event Source το οποίο αποτελεί το αντικείμενο του οποίου η κατάσταση αλλάζει και παράγει το event (π.χ. ένα κουμπί που έχει πατηθεί), το Event Object, δηλαδή το αντικείμενο που αντιπροσωπεύει το ίδιο το γεγονός και ενθυλακώνει τις αλλαγές του Event Source και κρατά πληροφορίες για το γεγονός (π.χ. ένα ButtonEvent). Τέλος, υπάρχει και ο ακροατής του γεγονότος (Event Listener) που ειδοποιείται όταν προκαλείται το γεγονός (π.χ. ένας ActionListener). Γενικότερα υπάρχουν διάφορα είδη events από τα οποία τα κυριότερα είναι τα ActionEvent, MouseEvent, KeyEvent, FocusEvent και ComponentEvent και τα καθένα από αυτά πυροδοτείται υπό συγκεκριμένες συνθήκες.

Για να διαχειριστούμε ένα γεγονός χρειάζεται να προσθέσουμε τον κατάλληλο τρόπο διαχείρισης του γεγονότος. Οι μέθοδοι που αναλαμβάνουν τη διαχείριση αυτών των γεγονότων περιέχονται στο αντίστοιχο interface που ονομάζονται ακροατές (listeners). Ένα Event Source στοιχείο μπορεί να έχει πολλούς ακροατές ταυτόχρονα και κάθε ακροατής μπορεί να "ακούει" επίσης πολλά στοιχεία. Στον Πίνακα 2.1 παρουσιάζονται κάποιοι τύποι ακροατών γεγονότων μαζί με τις ενέργειες που τα πυροδοτούν:

Πίνακας 2.1: Τύποι ακροατών σε συνδυασμό με τις ενέργειες που πυροδοτούν γεγονότα [2]

Είδος Ακροατή	**Ενέργεια που πυροδοτεί το Γεγονός**
ActionListener	Ο χρήστης πατάει ένα κουμπί, το Enter ή επιλέγει ένα αντικείμενο
WindowListener	Ο χρήστης κλείνει ένα παράθυρο
MouseListener	Ο χρήστης κάνει κλικ πάνω σε ένα αντικείμενο
MouseMotionListener	Ο χρήστης κινεί τον κέρσορα του ποντικιού πάνω σε ένα αντικείμενο
ComponentListener	Ένα αντικείμενο γίνεται εμφανές
FocusListener	Ένα αντικείμενο είναι έτοιμο να δεχθεί είσοδο από το πληκτρολόγιο
ListSelectionListener	Αλλάζει η επιλογή σε έναν πίνακα ή μια λίστα
PropertyChangeListener	Αλλάζει οποιαδήποτε τιμή σε ένα αντικείμενο

2.4 Maven

Η Maven είναι ένα εργαλείο (Build tool) που έχει αναπτυχθεί από την Apache Software Foundation με σκοπό τη βελτιστοποίηση διαχείρισης ενός project και χρησιμεύει στη δημιουργία και την τεκμηρίωση του εν λόγω project. Με αυτό τον τρόπο η Maven διευκολύνει το έργο των προγραμματιστών στη διαχείριση μεγάλων και πολύκλοκων projects. Συνήθως απευθύνεται σε projects γραμμένα σε Java, αλλά χρησιμοποιείται και για τη διαχείριση έργων σε άλλες γλώσσες όπως C#, Scala και Ruby.

Τα Build tools γενικότερα ασχολούνται με την αυτοματοποίηση της δημιουργίας ενός project και συμβάλουν στη δημιουργία του πηγαίου κώδικα, στο compiling του κώδικα, παρέχουν documentation, πακετάρουν τον κώδικα (packaging) σε αρχεία JAR και τέλος τον εγκαθιστούν σε ένα τοπικό (local repository), κεντρικό (central repository) ή απομακρυσμένο αποθετήριο (server repository). Η Maven συγκεκριμένα κατεβάζει τις εξαρτήσεις (dependencies) που αναπαριστούν τις βιβλιοθήκες ή τα αρχεία JAR για το project μας και τα εγκαθιστά ώστε να είναι έτοιμα προς χρήση.

Μερικά από τα οφέλη και τα πλεονεκτήματα της Maven περιλαμβάνουν τη λήψη των κατάλληλων JAR αρχείων ως προς την έκδοση καθώς μπορεί να υπάρχουν διαφορετικές εκδόσεις σε δύο διαφορετικά πακέτα του project και επομένως συμβάλει στην αποφυγή σεναρίων που μπορούν να προκύψουν προβλήματα ασυμβατότητας λόγω διαφορετικών εκδόσεων σε ένα μεγάλο project με πολλές βιβλιοθήκες. Επίσης για τη λήψη κάθε λογής βιβλιοθήκης δεν είναι ανάγκη να επισκεφτούμε την αντίστοιχη ιστοσελίδα της εταιρίας ή του συντάκτη του λογισμικού που μας ενδιαφέρει, αφού μπορούμε να βρούμε ότι βιβλιοθήκη θέλουμε συγκεντρωτικά στη σελίδα του Maven Repository [9]. Η Maven επιπλέον βοηθάει στη δημιουργία μιας κατάλληλης δομής του project και αυτοματοποιεί ενέργειες που αφορούν την κατασκευή, την τεκμηρίωση και πληροφορίες σχετικά με την έκδοση του project, όπως αναφέραμε πιο πάνω. Επιπρόσθετα μπορεί να παρέχει χρήσιμες πληροφορίες για το project, οι οποίες λαμβάνονται τόσο από το pom.xml αρχείο όσο και από τις πηγές του project. Επίσης τα περισσότερα περιβάλλοντα ανάπτυξης λογισμικού IDE υποστηρίζουν τη δημιουργία projects με Maven.

2.4.1 Το αρχείο pom.xml

Η Maven βασίζεται στο Project Object Model (POM). Το Project Object Model αποτελεί βασικό στοιχείο της Maven και είναι ένα XML αρχείο που περιέχει πληροφορίες σχετικά με το project και λεπτομέρειες διαμόρφωσης (configuration) για την κατασκευή του έργου [8]. Το αρχείο pom.xml βρίσκεται πάντα στην κορυφή (root) του καταλόγου σε οποιοδήποτε maven project και περιέχει τα πρόσθετα (plugins) και βιβλιοθήκες που χρησιμοποιούνται σ' αυτό. Κατά την εκτέλεση μιας εργασίας η Maven αναζητά το αρχείο pom.xml στο τρέχον directory και αφού το εντοπίσει, διαβάζει το αρχείο, λαμβάνει τις απαραίτητες πληροροφίες διαμόρφωσης και έπειτα εκτελεί τον κώδικα.

Το ελάχιστα στοιχεία (elements) που οφείλει να περιέχει ένα pom.xml αρχείο είναι:

- Το element <project> που είναι και το αρχικό element του XML αρχείου (root element).
- Το element <groupId> που λειτουργεί σαν ένα αναγνωριστικό Id του οργανισμού που αναπτύσσει το συγκεκριμένο project ή application.
- Το element <artifactId> που είναι το όνομα που θα αντιπροσωπεύει το ίδιο το project και όλον τον κώδικα που περιέχεται σε αυτό.

- Το element <version> στο οποίο αναγράφεται η έκδοση του project και μπορεί να ξεκινάει με τιμή 0.0.1-SNAPSHOT, όπου το SNAPSHOT δηλώνει ότι η εφαρμογή ή το project είναι ακόμα σε εξέλιξη.

Για την ακρίβεια, οι τιμές των τριών τελευταίων elements δέχονται τιμές διαμόρφωσης και αντιπροσωπεύουν το πλήρες αναγνωρισμένο όνομα του project. Κάτω από αυτά υπάρχει και το element <packaging> όπου αν διαθέτει την default τιμή jar δηλώνει ότι η συγκεκριμένη εφαρμογή θα πακεταριστεί σε ένα αρχείο JAR. Επίσης από κάτω μπορούν να υπάρχουν και τα elements <name> και <description> που παρέχουν πληροφορίες για λόγους τεκμηρίωσης.

Μια άλλη λειτουργία του αρχείου pom.xml είναι η κληρονομικότητα άλλων έργων και projects το οποίο λειτουργεί με τον ίδιο τρόπο που αναφέραμε στην ιδιότητα της κληρονομικότητας των κλάσεων στον αντικειμενοστρεφή προγραμματισμό. Αυτό που χρειάζεται να κάνουμε σε αυτή την περίπτωση είναι να προσθέσουμε ένα element <parent> μέσα στο αρχείο μας το οποίο θα έχει σαν εσωτερικά elements τα <groupId>, <artifactId> και <version> elements τα οποία όπως είπαμε πιο πριν αντιπροσωπεύουν το project στο οποίο αναφερόμαστε. Το <parent> element δηλώνει ότι το δικό μας project κληρονομεί πληροφορίες από αυτό το project. Σημαντικό σε αυτή την περίπτωση είναι το γονικό project που μας ενδιαφέρει να είναι ήδη εγκατεστημένο στο τοπικό μας αποθετήριο (local repository) και να βρίσκεται στο σωστό directory. Προκειμένου να αποφύγουμε προβλήματα σχετικά με την τοποθεσία του parent project στο directory υπάρχει το element <relativePath> μέσα στο οποίο δηλώνουμε την διαδρομή του parent project ως ../parent/pom.xml.

Μέχρι τώρα μιλήσαμε για τα κύρια elements που δομούν ένα project στο αρχείο pom.xml. Ωστόσο, ο κύριος λόγος που χρησιμοποιούμε το αρχείο pom.xml είναι για να έχουμε εύκολη πρόσβαση σε βιβλιοθήκες που είναι χρήσιμες ή και απαραίτητες για το δικό μας project. Αυτό το ρόλο παίζουν τα dependencies ή εξαρτήσεις τα οποία δηλώνουμε ως ξεχωριστό element με το ίδιο όνομα, δηλαδή <dependencies>. Μέσα σε αυτό προσθέτουμε ένα θυγατρικό element με όνομα <dependency> και εσωτερικά του προσθέτουμε το κυρίως σώμα της εξάρτησης που θέλουμε, δηλαδή τα βασικά elements <groupId>, <artifactId> και <version>. Αυτά τα elements αρκούν για να χρησιμοποιήσουμε τη βιβλιοθήκη στην οποία αναφέρονται. Κάθε φορά που προσθέτουμε νέα βιβλιοθήκη πρέπει να περιέχει τα βασικά τρία elements μέσα στο <dependency> και αυτό να βρίσκεται μέσα στο γονικό element <dependencies>. Αν έχουμε δύο ή παραπάνω βιβλιοθήκες τότε αυτές τοποθετούνται η καθεμία ως ξεχωριστό <dependency> το ένα κάτω απ' το άλλο και όλα μαζί μέσα στο ενιαίο <dependencies> element. Επιπρόσθετα μέσα στο <dependency> element μπορούμε να έχουμε το <type> element που δηλώνει τον τύπο της συγκεκριμένης εξάρτησης με default τιμή το jar, το <scope> element που χρησιμοποιείται για τον περιορισμό της μεταβατικότητας της εξάρτησης και default τιμή compile και τέλος το <optional> element που εκφράζει το γεγονός ότι αν αυτό το project χρησιμοποιηθεί ως εξάρτηση είναι προαιρετικό ως προς τη χρήση του. Στην εικόνα 2.2 δίνεται ένα παράδειγμα μιας εξάρτησης σε ένα pom.xml αρχείο.

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2.   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
3.   ...
4.   <dependencies>
5.     <dependency>
6.       <groupId>junit</groupId>
7.       <artifactId>junit</artifactId>
8.       <version>4.12</version>
9.       <type>jar</type>
10.      <scope>test</scope>
11.      <optional>true</optional>
12.    </dependency>
13.    ...
14.  </dependencies>
15.  ...
16. </project>
```

Εικόνα 2.2: Παράδειγμα μιας εξάρτησης ενός αρχείου pom.xml [7]

2.5 HyperText Markup Language (HTML)

Η γλώσσα υπερκειμένου (markup language) είναι μία από τις κατηγορίες γλωσσών τις οποίες καταλαβαίνει ο υπολογιστής. Σε αυτές τις κατηγορίες επίσης ανήκουν οι γλώσσες προγραμματισμού (programming languages), όπως η Java που αναφέραμε, η C++, η Python, κλπ. και οι γλώσσες σεναρίου (scripting languages) όπως η Php, Javascript, Asp, κλπ. Μια γλώσσα υπερκειμένου είναι ένα σύστημα που στοχεύει στο σχολιασμό ενός εγγράφου έτσι ώστε να διαχωρίζεται αυτός ο σχολιασμός από το ίδιο το κείμενο. Με την επεξεργασία του εγγράφου ο σχολιασμός της γλώσσας δεν είναι εμφανής αλλά χρησιμοποιείται για τον τρόπο με τον οποίο θα εμφανιστεί το κείμενο του εγγράφου.

Με την έννοια υπερκείμενο εννοούμε ότι το κείμενο του διαδικτύου απαρτίζεται από συνδέσμους στους οποίους μπορούμε να μεταβούμε ανά πάσα στιγμή και δεν υπάρχει περιορισμός στον τρόπο πλοήγησης, σε αντίθεση για παράδειγμα με ένα βιβλίο, το οποίο για να κατανοήσουμε πρέπει να το διαβάσουμε με τη σειρά, σελίδα-σελίδα. Στην περίπτωση του βιβλίου το κείμενο είναι γραμμικό, δηλαδή έχει μια συγκεκριμένη σειρά που πρέπει να διαβαστεί και η έννοια “υπέρ” στο υπερκείμενο δηλώνει την υπέρβαση από αυτό τον περιορισμό.

Η γλώσσα σήμανσης υπερκειμένου HTML (από το HyperText Markup Language) αποτελεί την κύρια γλώσσα σήμανσης για τις ιστοσελίδες του Διαδικτύου, όπου η σήμανση γίνεται μέσω των ετικετών της γλώσσας, τα λεγόμενα HTML tags. Οι περιηγητές ιστού (web browsers) χρησιμοποιούν τα HTML έγγραφα και διαβάζουν τις ετικέτες αυτές οι οποίες “λένε” στον περιηγητή πως να εμφανίσει το περιεχόμενο της σελίδας, το οποίο μπορεί να περιέχει κεφαλίδες, τίτλους, παραγράφους, εικόνες, πίνακες, κλπ. Οι ετικέτες του εγγράφου περικλείονται μέσα σε αγκύλες και είναι της μορφής < />. Μέσα σε αυτές τις αγκύλες ορίζονται οι διάφοροι τύποι των ετικετών με τον καθένα από αυτούς να προσδίδει διαφορετική σημασία στο κείμενο. Κάθε ετικέτα αποτελείται από τρία μέρη, την ετικέτα έναρξης (π.χ. <html>), το περιεχόμενο μέσα στην ετικέτα (π.χ. απλό κείμενο) και την ετικέτα λήξης, η οποία είναι ίδια με την ετικέτα έναρξης με τη μόνη διαφορά ότι διαθέτει και μια κάθετη διαχωριστική γραμμή (π.χ. στην περίπτωσή μας </html>). Επιπλέον υπάρχουν και τα εμφωλευμένα tags τα οποία λειτουργούν με τον ίδιο τρόπο απλά είναι μέσα σε ένα άλλο tag, δηλαδή αποτελούν περιεχόμενό του. Δεν διαβάζουν

όλοι οι περιηγητές την HTML με τον ίδιο τρόπο και μπορεί να υπάρχουν μικρές διαφορές στην εμφάνιση αλλά η φιλοσοφία καθολικά παραμένει η ίδια.

Ένα HTML έγγραφο έχει πάντα σαν πρώτο στοιχείο το <!DOCTYPE html> το οποίο δεν αποτελεί <html> στοιχείο, αλλά δηλώνει την έκδοση HTML του εγγράφου και δίνει οδηγίες στον περιηγητή για τον τύπο του εγγράφου και πώς να το φορτώσει κατάλληλα. Μετά το doctype, το έγγραφο ξεκινάει και τελειώνει με το <html> tag, (<html> για έναρξη και </html> για λήξη) και χωρίζεται σε 2 μικρότερα μέρη το <head> tag και το <body> tag. Το <head> tag μπαίνει πρώτο και περιέχει πληροφορίες για το έγγραφο και τα μεταδεδομένα (metadata), δηλαδή δεδομένα που περιγράφουν άλλα δεδομένα και δεν εμφανίζεται το περιεχόμενό του στη σελίδα. Όλο το ορατό μέρος του εγγράφου περιέχεται στο <body> tag που είναι και το κύριο σώμα του εγγράφου και διαθέτει όλες τις πληροφορίες για το πώς θα εμφανιστεί το περιεχόμενο.

Μερικά από τα κυριότερα tags που συναντάμε σε ένα έγγραφο είναι τα header tags, από τα οποία υπάρχουν 6 επίπεδα (από <h1> μέχρι <h6>) και χρησιμοποιούνται για να επισημάνουν τους τίτλους και τις επικεφαλίδες του κειμένου με το <h1> να είναι ο μεγάλος τίτλος και <h6> η μικρότερη επικεφαλίδα. Επίσης τα headings είναι σημαντικά για την οργάνωση του HTML εγγράφου, καθώς επίσης και για μια διαδικασία που ονομάζεται Βελτιστοποίηση Μηχανών Αναζήτησης ή Search Engine Optimization (SEO) και αυτό διότι οι μηχανές αναζήτησης ψάχνουν συγκεκριμένα στοιχεία στις σελίδες με τα headings να έχουν πρωταρχικό ρόλο. Επιπλέον τα tags χρησιμοποιούνται για το απλό κείμενο ή παραγράφους, τα tags χρησιμοποιούνται για να χωρίσουν ή να ομαδοποιήσουν κομμάτια του εγγράφου και συνήθως παίρνουν άλλα εσωτερικά tags στο περιεχόμενό τους. Επίσης συμβάλουν στο να ξεχωρίσουν αυτά τα στοιχεία που περιέχουν προκειμένου να προσθέσουμε κάποιου είδους στυλ μορφοποίησης (style) σε αυτά που θα δούμε παρακάτω. Τα <table> tags δηλώνουν πίνακες με γραμμές (<tr> tags) και κελιά των γραμμών (<td> tags). Οι εικόνες ως στοιχεία σε ένα έγγραφο δηλώνονται ως <img> tags με κάποιες επιπλέον πληροφορίες όπως την πηγή της φωτογραφίας (src), το εναλλακτικό κείμενο (alt), το μήκος (width) και το πλάτος (height). Τέλος, υπάρχουν και τα σχόλια της HTML που ξεκινούν με <!-- και τελειώνουν με --> και οτιδήποτε βρίσκεται μέσα σε αυτά θεωρείται ως σχόλιο και ο περιηγητής το αγνοεί και δεν το λαμβάνει υπόψη στην εμφάνιση του εγγράφου.

Σε μια HTML σελίδα μπορούμε επίσης να προσθέσουμε κώδικα για στυλ μορφοποίησης με χρήση Cascading Style Sheets (CSS) που χρησιμοποιούνται για να αποκτήσουμε πλήρη έλεγχο για την εμφάνιση ενός εγγράφου τόσο στο ίδιο το κείμενο, π.χ. στο μέγεθος των γραμμάτων (font-size), την κατηγορία γραμματοσειράς (font-family), το χρώμα (color) και άλλα, όσο και για τον τρόπο που είναι οργανωμένα τα στοιχεία της σελίδας, πόσο απέχουν μεταξύ τους, τη στοίχισή τους, κλπ. Αυτό το επιτυγχάνουμε προσθέτοντας την ιδιότητα style μέσα στην ετικέτα έναρξης του στοιχείου που μας ενδιαφέρει και μέσα στο style γράφουμε τις ιδιότητες που θέλουμε να δώσουμε στο συγκεκριμένο στοιχείο. Προκειμένου να αλλάξουμε το χρώμα μιας παραγράφου σε πράσινο θα είχαμε για παράδειγμα: <p style=”color: green;”>ΤΟ ΚΕΙΜΕΝΟ ΠΟΥ ΘΕΛΟΥΜΕ ΝΑ ΓΙΝΕΙ ΠΡΑΣΙΝΟ . Αντίστοιχα μπορούμε να επεξεργαστούμε οποιοδήποτε στοιχείο προσθέτοντάς του την κατάλληλη style ιδιότητα.

Εάν θέλουμε η HTML σελίδα μας να μην είναι απλά μια στατική σελίδα αλλά να είναι διαδραστική, μπορούμε να ενσωματώσουμε κώδικα της γλώσσας σεναρίου Javascript μέσα σε αυτήν και με τις κατάλληλες εντολές να εκτελείται κάποιο σενάριο. Αυτό το πετυχαίνουμε προσθέτοντας την ετικέτα <script> και εσωτερικά τις εντολές Javascript που μας ενδιαφέρουν και κλείνοντας μετά την ετικέτα </script>. Το <script> tag μπορεί να μπει είτε στο <head> μέρος, είτε στο <body> ανάλογα με το τι σενάριο θέλουμε να εκτελεστεί στον περιηγητή. Σημαντικό εδώ να αναφέρουμε ότι η Javascript

είναι μια client-side γλώσσα δηλαδή εκτελείται τοπικά στον περιηγητή του χρήστη κάθε φορά που φορτώνεται η αντίστοιχη σελίδα.

Στο επόμενο κεφάλαιο που θα αναλύσουμε τη δημιουργία της εφαρμογής, θα χρησιμοποιήσουμε HTML έγγραφα τα οποία είναι τα θεατρικά έργα που θα επεξεργαστούμε και στη συνέχεια θα αναλύσουμε.

2.6 Web Scraping

Στη σύγχρονη εποχή που ζούμε με τη ραγδαία εξέλιξη της τεχνολογίας, τα δεδομένα έχουν κυρίαρχο ρόλο στον τρόπο που επικοινωνούμε, εκφραζόμαστε, ενημερωνόμαστε και γενικότερα μας συνδέουν με τον υπόλοιπο κόσμο. Σε αυτό έρχεται να συμβάλλει και το Διαδίκτυο, το οποίο αποτελεί μέχρι σήμερα αναπόσπαστο κομμάτι της καθημερινότητάς μας και παράλληλα είναι αστείρευτη πηγή πληροφορίας και δεδομένων. Αρκετές φορές είναι επιθυμητό ή και απαραίτητο να συλλέξουμε δεδομένα από κάποια πηγή, προκειμένου να τα επεξεργαστούμε, να τα συγκρίνουμε και να τα αναλύσουμε. Η συλλογή δεδομένων είναι σημαντική για την υποστήριξη επιστημονικών, ιατρικών και στατιστικών ερευνών αλλά και για την βελτίωση της τεχνολογίας και την προώθηση της οικονομίας και του εμπορίου. Ιδιαίτερα για τον επιχειρηματικό κόσμο, η συλλογή δεδομένων έχει ανεκτίμητη αξία καθώς ένας μεγάλος αριθμός εταιρειών και επιχειρήσεων έχουν πρόσβαση στα προσωπικά δεδομένα των πελατών τους προκειμένου να αναλύσουν αυτά τα δεδομένα και να βγάλουν κάποια συμπεράσματα που θα τις βοηθήσουν να βελτιώσουν τις υπηρεσίες που προσφέρουν, να αναπτύξουν νέες στρατηγικές επενδύσεων και να μεγιστοποιήσουν το κέρδος τους.

Η απόσπαση ή "απόξεση" δεδομένων ιστού κοινώς γνωστή ως Web Scraping είναι μια τεχνική συλλογής και απόκτησης δεδομένων από το Διαδίκτυο η οποία έχει ως σκοπό την εξαγωγή αυτών των δεδομένων σε κατάλληλη μορφή για ανάλυση και επεξεργασία. Η ενέργεια της συλλογής δεδομένων μπορεί να γίνει χειρονακτικά από οποιονδήποτε, ωστόσο δεν είναι καθόλου αποτελεσματική όταν ο όγκος των δεδομένων που επιθυμεί να συγκεντρώσει είναι πολύ μεγάλος και επομένως απαιτείται χρόνος και ενέργεια για να επιτευχθεί αυτό. Ακριβώς αυτό το πρόβλημα αντιμετωπίζει το web scraping το οποίο εκτελεί αυτόματα και ταχύτατα τη συλλογή δεδομένων από κάποιο ιστότοπο ή σελίδα του Διαδικτύου. Η τεχνική αυτή χρησιμοποιείται από προγράμματα που διαβάζουν και σκανάρουν μια πηγή η οποία είναι κατανοητή από τον άνθρωπο (π.χ. μια σελίδα ηλεκτρονικού καταστήματος με προϊόντα) και με τις κατάλληλες εντολές εξάγουν τα επιθυμητά δεδομένα από την πηγή. Η πρακτική της συλλογής δεδομένων από ιστοσελίδες θεωρείται νόμιμη γενικά, ωστόσο η συλλογή δεδομένων από ιστοτόπους τρίτων πρέπει να γίνεται σε λογικά πλαίσια.

Τα προγράμματα που είναι υπεύθυνα για τη συλλογή δεδομένων από ιστότοπους λέγονται web scrapers. Επειδή στο Διαδίκτυο υπάρχει ένας τεράστιος αριθμός ιστοσελίδων που διαφέρουν σημαντικά μεταξύ τους ως προς το μέγεθος και το περιεχόμενό τους, οι web scrapers μπορούν να ποικίλουν ως προς τη λειτουργικότητά και τα χαρακτηριστικά τους. Όσον αφορά το πως λειτουργούν, οι scrapers δέχονται συνήθως ένα η περισσότερα URL της ιστοσελίδας της οποίας θα συλλεχθούν τα δεδομένα και στη συνέχεια φορτώνουν το HTML περιεχόμενό της. Μερικές φορές μπορούν να φορτώσουν και ολόκληρη την ιστοσελίδα συμπεριλαμβανομένου του CSS και της Javascript. Έπειτα οι scrapers εξάγουν είτε όλα τα δεδομένα της ιστοσελίδας ή τα συγκεκριμένα που επιλέγει ο χρήστης και τον ενδιαφέρουν, σε μια μορφή που είναι χρήσιμη και φιλική για το χρήστη, για πάραδειγμα σε ένα csv αρχείο ή ένα excel υπολλογιστικό φύλλο. Εφόσον τα δεδομένα είναι αποθηκευμένα σε αυτού του είδους τη μορφή, μπορούν εύκολα μετά να αναλυθούν για τον εκάστοτε σκοπό.

2.6.1 Jsoup

Το Jsoup [10] είναι μια βιβλιοθήκη ανοιχτού κώδικα (open source) σε Java και απευθύνεται σε προγραμματιστές που επιθυμούν να εξάγουν δεδομένα μέσα από HTML έγγραφα. Επίσης έχει τη δυνατότητα διαχείρισης πληροφορίας και εξαγωγής σε HTML, αλλά και την ικανότητα ανάλυσης και δημιουργίας XML. Διαθέτει Application Programming Interface (API) για την ανάκτηση URL και το χειρισμό δεδομένων, χρησιμοποιώντας HTML 5 DOM μεθόδους.

Πιο συγκεκριμένα, το Jsoup προσφέρει:

- Συλλογή (scraping) και ανάλυση (parsing) HTML δεδομένων που παρέχονται από ένα URL, ένα αρχείο ή μια συμβολοσειρά (String). Η ανάλυση γίνεται με χρήση της μεθόδου Jsoup.parse().
- Εύρεση και εξαγωγή των κατάλληλων δεδομένων, χρησιμοποιώντας διέλευση του Document Object Model (DOM) ή CSS επιλογείς. Για την εύρεση δεδομένων με χρήση του DOM οι κυριότερες μέθοδοι είναι οι getElementById(), getElementsByTag(), getElementsByClass(), getElementsByAttribute(), nextElementSibling(), siblingElements(), firstElementSibling() αλλά υπάρχουν και άλλες. Για την επιλογή δεδομένων επίσης υπάρχει η μέθοδος select() και μέσα μπορεί να περιέχεται ένα tagname, ένα id attribute , μια κλάση, κλπ.
- Επεξεργασία και χειρισμό HTML στοιχείων, χαρακτηριστικών (element attributes) και κειμένου. Για να προσθέσουμε ένα attribute σε ένα element υπάρχει η μέθοδος Element.attr(key, value) όου key το όνομα του attribute και value η τιμή του. Για να προσθέσουμε κλάση υπάρχει η μέθοδος Element.addClass(className) με className το όνομα της κλάσης που θα δώσουμε. Για να πάρουμε την HTML ενός element χρησιμοποιούμε τη μέθοδο html() και αντίστοιχα για να πάρουμε μόνο το περιεχόμενο του δηλαδή απλό κείμενο, χρησιμοποιούμε τη μέθοδο text().
- Εξαγωγή "καθαρής" HTML, καθώς έχει σχεδιαστεί για να επεξεργάζεται όλων των ειδών της μορφής HTML. Αυτό είναι ιδιαίτερα σημαντικό, στην περίπτωση που διαθέτουμε έναν ιστότοπο και επιτρέπουμε σε μη αξιόπιστους χρήστες να υποβάλλουν στοιχεία (π.χ. σχόλια σε μια φόρμα), τότε υπάρχει πιθανότητα ο ιστότοπός μας να δεχθεί μια Cross-site scripting (XSS) επίθεση. Σε αυτό το σενάριο, οι κακόβουλοι χρήστες εκμεταλλεύονται ευπάθειες στην ασφάλεια εφαρμογών και ιστοσελίδων και χρησιμοποιώντας HTML και κώδικα Javascript μπορούν να παρακάμψουν ελέγχους πρόσβασης, να αποκτήσουν κωδικούς ασφαλείας και να ζημιώσουν σοβαρά τους χρήστες της εφαρμογής. Γι' αυτό το λόγο, το Jsoup "καθαρίζει" αναξιόπιστη HTML χρησιμοποιώντας τη μέθοδο clean( ), η οποία δέχεται HTML σε μορφή String και μέσω του Whitelist, το προσθέτει σε μια λίστα που δηλώνει ότι ο κώδικας είναι ασφαλής.

Στο επόμενο κεφάλαιο θα χρησιμοποιηθεί η βιβλιοθήκη Jsoup για τη συλλογή των δεδομένων που μας ενδιαφέρουν από τα θεατρικά έργα και θα δείξουμε παραδείγματα scraping για διαφορετικές περιπτώσεις.

2.7 eXtensible Markup Language (XML)

Η XML είναι μια γλώσσα σήμανσης όπως η HTML η οποία αναπτύχθηκε για να διευκολύνει την επικοινωνία μεταξύ υπολογιστών που ανταλλάζουν δεδομένα μέσω του Διαδικτύου και άλλων

μέσων. Κύριο χαρακτηριστικό της XML είναι η κωδικοποίηση και μορφοποίηση κειμένων βάσει κάποιων κανόνων ώστε να μπορούν εύκολα να οργανωθούν αυθαίρετα δεδομένα σε μια απλή και δομημένη μορφή που μπορεί να κατανοηθεί από έναν υπολογιστή. Κύρια προδιαγραφή είναι η XML 1.0 που δημιουργήθηκε από το Διεθνή Οργανισμό Προτύπων W3C (World Wide Web Consortium) [12] αλλά υπάρχουν και άλλες προδιαγραφές ανοιχτών προτύπων.

Αυτή η οργάνωση των δεδομένων μπορεί να έχει αρκετά πλεονεκτήματα καθώς συμβάλλει στην αποτελεσματικότητα αποθήκευσης και μεταφοράς δεδομένων και είναι ανεξάρτητη από οποιαδήποτε τεχνολογία, λογισμικό ή πλατφόρμα όσον αφορά την ανταλλαγή δεδομένων. Για παράδειγμα δύο εφαρμογές που έχουν δημιουργηθεί σε διαφορετικές πλατφόρμες μπορούν να ανταλλάξουν δεδομένα μέσω XML εγγράφων χωρίς να αποτελεί εμπόδιο η διαφορά λογισμικού. Επίσης η οργάνωση των δεδομένων σε XML μπορεί να κατανοηθεί και από τον υπολογιστή αλλά και από τον άνθρωπο ο οποίος μπορεί να διαβάσει αλλά και να εμπλουτίσει το έγγραφο χωρίς περιορισμό στον ορισμό των σημάνσεων και έτσι προκύπτει η ονομασία eXtensible. Αυτό πρακτικά σημαίνει ότι ένα έγγραφο μπορεί να επεκταθεί με νέα στοιχεία, όπως επίσης μπορούν και να αφαιρεθούν στοιχεία, αρκεί να παραμένει σωστή η δομή και η σύνταξη του εγγράφου. Η XML διαθέτει ισχυρή υποστήριξη για το πρότυπο κωδικοποίησης Unicode, έτσι ώστε να μπορούν να αποθηκευτούν δεδομένα σε όλες τις γλώσσες του κόσμου συμπεριλαμβανομένων και ειδικών χαρακτήρων.

Αντίστοιχα με την HTML, τα δεδομένα της XML αποθηκεύονται με τη μορφή εγγράφων που λέγονται XML documents. Κάθε τέτοιο αρχείο έχει την κατάληξη .xml και για τη σήμανση χρησιμοποιούνται ετικέτες ανάλογες με αυτές που υπάρχουν στην HTML, δηλαδή της μορφής < /> και με μία ετικέτα λήξης για κάθε ετικέτα έναρξης. Μέσα σε κάθε ετικέτα ορίζεται το περιεχόμενό της και μπορούν να υπάρχουν εμφωλευμένες ετικέτες μέσα σε άλλες χωρίς κανένα περιορισμό. Επίσης μπορεί να υπάρχουν και ετικέτες χωρίς περιεχόμενο ή κενές ετικέτες, π.χ. <tag/>. Αυτό ισοδυναμεί με μια ετικέτα έναρξης <tag>, καθόλου περιεχόμενο και μια ετικέτα λήξης </tag>. Το πρώτο στοιχείο ή η πρώτη γραμμή σε κάθε αρχείο πρέπει να δηλώνει ότι αποτελεί XML έγγραφο συμπεριλαμβανομένης και της έκδοσης της XML που χρησιμοποιείται και είναι μια καλή πρακτική να περιλαμβάνεται και η κωδικοποίηση, αλλά αυτό είναι προαιρετικό. Παράδειγμα δήλωσης XML εγγράφου: <?xml version="1.0" encoding="UTF-8" standalone="no"?>. Εδώ βλέπουμε ότι η XML έκδοση είναι η 1.0, η κωδικοποίηση που χρησιμοποιείται είναι UTF-8 και το standalone δηλώνει εάν το δεδομένο έγγραφο εξαρτάται από μια εξωτερική δήλωση σήμανσης, που σε αυτή την περίπτωση δεν εξαρτάται. Τα υπόλοιπα στοιχεία μπορούν να δηλωθούν από το συντάκτη του εγγράφου ξεκινώντας πάντα από το πρώτο στοιχείο που ονομάζεται ρίζα ή root element και μέσα σ' αυτό προστίθενται όλα τα υπόλοιπα στοιχεία που λέγονται child elements, δηλαδή στοιχεία παιδιά του root element. Τα child elements μπορούν να έχουν και αυτά δικά τους στοιχεία παιδιά και κάπως έτσι δημιουργείται μια ιεραρχική δομή δέντρου. Επιπρόσθετα, όπως και στην HTML τα στοιχεία μπορούν να έχουν το καθένα δικά τους χαρακτηριστικά (attributes) τα οποία πρέπει να δηλώνονται μέσα στην ετικέτα έναρξης και η τιμή του attribute πρέπει να είναι μέσα σε " ", για παράδειγμα <person name="Giorgos">. Εδώ έχουμε το element person με το attribute name και τιμή του attribute Giorgos. Τα attributes χρησιμοποιούνται για να δώσουν μια ιδιότητα σε αυτά και να ξεχωρίζουν από τα υπόλοιπα elements με βάση αυτό.

Σε γενικές γραμμές η XML είναι μια πολύ χρήσιμη γλώσσα σήμανσης για την οργάνωση δεδομένων σε απλή και χρήσιμη μορφή και θα τη χρησιμοποιήσουμε στο επόμενο κεφάλαιο της υλοποίησης για τη διευκόλυνση επεξεργασίας των δεδομένων.

2.7.1 Διαφορές XML και HTML

Τόσο η XML όσο και η HTML αποτελούν ίσως τις δύο πιο δημοφιλείς και γνωστές γλώσσες σήμανσης, ωστόσο έχουν κάποιες βασικές διαφορές. Μερικές από αυτές είναι:

- Αρχικά, η XML έχει ως κύριο μέλημα τη σωστή δομή και οργάνωση των δεδομένων και επικεντρώνεται στη σημασία τους και το πως σχετίζονται ιεραρχικά. Αντίθετα η HTML αφορά καθαρά την παρουσίαση των δεδομένων και το πως αυτά μπορούν να αναπαρασταθούν.
- Τα XML tags μπορούν να έχουν οποιαδήποτε τιμή ή ονομασία, αντίθετα με την HTML που τα tags έχουν προκαθορισμένες ονομασίες και συγκεκριμένο σκοπό το καθένα από αυτά. Στην XML το σκοπό και την ονομασία των tags, τα δίνει ο συντάκτης, ενώ στην HTML μπορεί απλά να χρησιμοποιήσει τις ετικέτες ανάλογα με το ρόλο που έχει η καθεμία.
- Αντίστοιχα με τη δεύτερη περίπτωση, η δομή του XML εγγράφου είναι καθαρά στα χέρια και την κρίση του συντάκτη η οποία δομή μπορεί να επεκταθεί εξίσου θεωρητικά άπειρα, ενώ στην HTML υπάρχει συγκεκριμένη δομή που πρέπει να ακολουθηθεί (Το έγγραφο χωρίζεται σε <head> και <body> και το καθένα παίρνει συγκεκριμένα tags με διαφορετικό ρόλο).
- Άλλη μια διαφορά είναι ότι τα XML tags είναι case sensitive, που σημαίνει ότι μια ετικέτα έναρξης πρέπει να είναι ακριβώς ίδια με την ετικέτα λήξης της, π.χ. εάν μια ετικέτα έναρξης είναι η <hello>, η ετικέτα λήξης της πρέπει να είναι ακριβώς </hello> με τα γράμματα αν είναι μικρά να παραμείνουν μικρά και αντίστροφα αν υπάρχουν κεφαλαία γράμματα να παραμείνουν κεφαλαία. Αν είχαμε οτιδήποτε άλλο π.χ. <hello> και </HELLO> τότε προκύπτει συντακτικό λάθος. Απ' την άλλη η HTML είναι case insensitive, δηλαδή δεν την ενδιαφέρει αν τα γράμματα ενός ζευγαριού ετικετών έναρξης και λήξης είναι κεφαλαία ή μικρά.
- Επίσης στην XML κάθε tag που ανοίγει πρέπει αναγκαστικά να κλείνει. Στην HTML δεν ισχύει απαραίτητα αυτό, μερικές φορές κάποια tags μπορεί να μην έχουν ετικέτα λήξης χωρίς να παρουσιάζεται κάποιο πρόβλημα.
- Η XML είναι δυναμική επειδή βασίζεται στη μεταφορά δεδομένων και όχι στην παρουσίαση, ενώ η HTML είναι απ' τη φύση της στατική.
- Η XML διατηρεί white space (κενό διάστημα το οποίο μπορεί να είναι απλό space, tabs ή line breaks δηλαδή αλλαγή γραμμής). Αντίθετα η HTML δεν διατηρεί white space.
- Τα XML αρχεία έχουν επέκταση ονόματος αρχείου (filename extension) .xml, ενώ τα HTML αρχεία έχουν .html ή .htm.

2.7.2 XML Schema

Ένα XML schema είναι μια γλώσσα που περιγράφει τον τύπο ενός XML εγγράφου, δηλαδή είναι ένα σύνολο κανόνων και περιορισμών πέρα από τους βασικούς περιορισμούς που επιβάλλει η ίδια η XML και "σχηματίζουν" το πως πρέπει να είναι ένα XML έγγραφο (εξού και η ονομασία). Υπάρχουν διαφορετικές schema γλώσσες που χρησιμοποιούνται για την επιβολή περιορισμών στα έγγραφα με τις κυριότερες να είναι το Document Type Definition (DTD), το XML Schema Definition (XSD) και το RELAX NG. Σε αυτή την ενότητα θα μιλήσουμε για τα δύο πρώτα. Ο τρόπος συσχέτισης ενός XML εγγράφου με ένα schema διαφέρει ανάλογα με το schema και μπορεί να επιτευχθεί είτε με τη χρήση εσωτερικής σήμανσης μέσα στο έγγραφο ή με χρήση άλλων εξωτερικών μέσων.

Ένα XML schema μπορεί να είναι πολύ χρήσιμο όταν θέλουμε να επεξεργαστούμε και να συγκρίνουμε τα δεδομένα διαφορετικών εγγράφων, ιδίως αν αυτά ακολουθούν κάποιους συγκεκριμένους κανόνες. Για παράδειγμα μπορούμε να δούμε τι τύπος στοιχείων επιτρέπονται όπως και τι είδους σειρά μπορούν να έχουν αυτά, όπως επίσης μας δίνει τη δυνατότητα να βρούμε εύκολα στοιχεία εφόσον γνωρίζουμε τη δομή του εγγράφου, το οποίο μπορεί να μας γλυτώσει χρόνο και κόπο εάν το έγγραφο είναι τεράστιο και διαθέτει χιλιάδες γραμμές. Επίσης βοηθάει τους υπολογιστές στην ανταλλαγή δεδομένων, για παράδειγμα αν έχουμε δύο βάσεις δεδομένων οι οποίες επιθυμούν να ανταλλάξουν δεδομένα μέσα από XML έγγραφα και να τα εισάγουν στη βάση τους, τα δεδομένα αυτά θα πρέπει να είναι κατάλληλα δομημένα και οργανωμένα για να εκτελεστεί σωστά αυτή η εισαγωγή. Γι' αυτό το λόγο, αρκετές φορές δεν αρκεί ένα έγγραφο να είναι συντακτικά σωστό (well formed), αλλά πρέπει να είναι ταυτόχρονα και έγκυρο (valid).

Η διαδικασία ελέγχου ενός XML εγγράφου για το αν ακολουθεί τους κανόνες ενός XML schema ονομάζεται επικύρωση (validation). Η επικύρωση προσθέτει νέους κανόνες και περιορισμούς πέρα από τους απλούς κανόνες σύνταξης της XML όπως αναφέραμε. Κάθε έγγραφο οφείλει να είναι well formed ωστόσο δεν είναι υποχρεωτικό να είναι έγκυρο, εκτός και αν το επιβάλουν οι συνθήκες για τις οποίες θα χρησιμοποιηθεί, οπότε ελέγχεται με ένα schema για την εγκυρότητά του.

2.7.3 Document Type Definition (DTD)

Το Document Type Definition είναι μια απ' τις πιο γνωστές και υποστηριζόμενες γλώσσες σχήματος για XML. Σκοπός του είναι ο ορισμός της δομής και των στοιχείων που επιτρέπονται σε ένα XML έγγραφο, έτσι ώστε ανεξάρτητες ομάδες ανθρώπων να μπορούν να συμφωνήσουν σε ένα συγκεκριμένο πρότυπο για ανταλλαγή δεδομένων. [11] Ένα έγγραφο που ελέγχεται με ένα DTD είναι σωστό συντακτικά (well formed) και είναι και έγκυρο (valid). Ένα DTD μπορεί να δηλωθεί εσωτερικά σε ένα XML έγγραφο ή να υπάρχει σε ένα εξωτερικό αρχείο το οποίο να δηλώνεται στο έγγραφο.

Ας υποθέσουμε ότι έχουμε ένα XML έγγραφο με το όνομα people.xml. Για να δηλώσουμε ένα DTD μέσα στο έγγραφο τότε αυτό δηλώνεται με το στοιχείο <!DOCTYPE> που τοποθετείται κάτω από το πρώτο element της XML. Επίσης δίπλα στο DOCTYPE υπάρχει το στοιχείο ρίζα (root) του εγγράφου και μέσα σε αγκύλες [ ] γράφονται οι κανόνες του DTD που πρέπει να ακολουθεί το έγγραφο. Δηλαδή η δήλωση του DTD πρέπει να έχει αυτή τη μορφή: <!DOCTYPE people [ ]>.

Αν το DTD είναι σε εξωτερικό αρχείο, τότε γίνεται αναφορά πάλι με το <!DOCTYPE>, το όνομα του root Element, τη λέξη SYSTEM και τέλος το όνομα του DTD αρχείου μέσα σε " " με την κατάληξη .dtd. Δηλαδή η αναφορά στο DTD πρέπει να έχει τη μορφή: <!DOCTYPE people SYSTEM "DTDname.dtd">.

Το κυρίως σώμα του DTD μπορεί να περιέχει τα 5 εξής στοιχεία: Elements, Attributes, Entities, PCDATA και CDATA. Τα Elements αντιπροσωπεύουν τα XML elements και τα Attributes αντίστοιχα αντιπροσωπεύουν τα χαρακτηριστικά των XML elements που διαθέτουν attributes. Τα Entities αποτελούν ειδικούς χαρακτήρες όπως < που ισοδυναμεί με το σύμβολο μικρότερο "<", το > που ισοδυναμεί με το σύμβολο μεγαλύτερο ">", ο χαρακτήρας & που είναι το σύμβολο "&" και άλλοι παρόμοιοι ειδικοί χαρακτήρες. Το PCDATA σημαίνει parsed character data και είναι το κείμενο που θα αναλυθεί από έναν parser ή εναλλακτικά είναι το κείμενο που υπάρχει μέσα στα tags. Το CDATA που σημαίνει character data είναι το κείμενο που δεν θα αναλυθεί από έναν parser.

Σχετικά με τη δήλωση δεδομένων, τα Elements δηλώνονται με το <!ELEMENT> στοιχείο και δίπλα το όνομα του στοιχείου στο οποίο αναφερόμαστε. Αν το element έχει παιδιά τα δηλώνουμε με το όνομά τους μέσα σε μια παρένθεση, αλλιώς αν το element δεν έχει στοιχεία παιδιά βάζουμε στην παρένθεση #PCDATA, δηλαδή το περιεχόμενό του. Εάν υπάρχουν περισσότερα από ένα elements με το ίδιο όνομα το δηλώνουμε με τον αστερίσκο "*". Για παράδειγμα μπορούμε να δηλώσουμε ένα element που έχει παιδιά ως <!ELEMENT people(person)*> και ένα element χωρίς παιδιά ως <!ELEMENT person(#PCDATA)*>. Τα Attributes δηλώνονται με το <!ATTLIST> στοιχείο και πρέπει να περιέχει το όνομα του element, το όνομα του attribute, τον τύπο του attribute και την τιμή του attribute. Δηλαδή για παράδειγμα, για το XML element <person name="Nikos"> έχουμε αντίστοιχα <!ATTLIST person name CDATA "Nikos">.

2.7.4 XML Schema Definition (XSD)

Το XSD ή XML Schema Definition είναι αντίστοιχα με το DTD ένας τρόπος περιγραφής της δομής των στοιχείων ενός XML εγγράφου. [11] Το XSD μπορεί να μοιάζει αρκετά και να έχει την ίδια λειτουργία με το DTD, έχει όμως περισσότερες δυνατότητες καθώς ορίζει τα elements και τα attributes που πρέπει να υπάρχουν στο έγγραφο, τον αριθμό και τη σειρά με την οποία εμφανίζονται και ορίζει τον τύπο των elements και αλλά και των attributes.

Ο λόγος που χρειαζόμαστε τα XSD παραμένει ο ίδιος με την περίπτωση των DTD, δηλαδή αποτελεί ιδιαίτερα χρήσιμο εργαλείο για τη διαμόρφωση ενός σταθερού προτύπου για XML έγγραφα. Επίσης ένα άλλο πλεονέκτημα που έχει το XSD είναι ότι μπορούμε να το προσπελάσουμε μέσω του Document Object Model (DOM), δηλαδή όπως θα κάναμε και για ένα απλό XML έγγραφο. Επειδή το XSD είναι γραμμένο σε XML, μπορεί να "μεταμορφωθεί" με μια γλώσσα που ονομάζεται XSLT. Το XSLT είναι μια styling γλώσσα που μετατρέπει XML έγγραφα σε άλλου είδους αρχεία, για παράδειγμα σε HTML, CSV αρχεία, κλπ. Ένα πλεονέκτημα ακόμα είναι ότι το ίδιο schema μπορεί να ξαναχρησιμοποιηθεί σε πολλά διαφορετικά XML έγγραφα.

Ένα XSD αποτελεί αυτοτελές, ξεχωριστό XML αρχείο και μπορεί να αναφερθεί μέσα στο XML έγγραφο που πρόκειται να ελεγχθεί. Το πρώτο στοιχείο σε ένα XSD είναι πάντα το <schema> το οποίο περιέχει κάποια attributes. Γενικά το πρώτο από αυτά που συναντάμε είναι το xmlns:xs="http://www.w3.org/2001/XMLSchema" το οποίο δηλώνει ότι τα στοχεία που περιέχονται στο έγγραφο αλλά και οι τύποι των δεδομένων δηλώνονται στο χώρο ονομάτων (namespace) με το όνομα μέσα στα εισαγωγικά και ότι πρέπει να έχουν το xs: πρόθεμα μπροστά. Ένας χώρος ονομάτων (namespace) είναι ένα σύνολο από ονόματα που χρησιμοποιούνται για την αναφορά σε αντικείμενα που περιέχουν κάποιο από αυτά τα ονόματα.

Τώρα θα εξηγήσουμε κάποιους από τους βασικότερους κανόνες σύνταξης για ένα XSD. Όπως είπαμε κάθε στοιχείο έχει μπροστά το πρόθεμα "xs:". Μπορούμε να δηλώσουμε το όνομα ενός οποιουδήποτε element με το attribute name και δίπλα το όνομά του και να ορίσουμε τον τύπο που θέλουμε να έχει με το attribute type, όπου η τιμή είναι π.χ. string αν θέλουμε αυστηρά συμβολοσειρά, integer αν θέλουμε ακέραιο, decimal αν θέλουμε δεκαδικό, κ.ο.κ. Κάθε στοιχείο το οποίο έχει παιδιά (ή και attributes) είναι complexType και δηλώνεται ως <xs:complexType> ακριβώς κάτω από το στοιχείο στο οποίο αναφερόμαστε. Εάν υπάρχει αλληλουχία από elements μέσα σε ένα parent element τότε αυτό ορίζεται ως <xs:sequence> επίσης από κάτω. Το στοιχείο <xs:choice> δηλώνει ότι για τα elements που περιέχονται μέσα σε αυτό, επιτρέπεται μόνο ένα, δηλαδή λειτουργεί σαν επιλογέας. Τέλος, αν θέλουμε να ορίσουμε πόσες φορές επιτρέπεται να εμφανιστεί ένα element το ορίζουμε με το attribute

minOccurs=”0” αν το στοιχείο είναι προαιρετικό ως προς την εμφάνισή του και αντίστοιχα με το maxOccurs δηλώνουμε μέχρι πόσες φορές επιτρέπεται να εμφανιστεί. Με την τιμή unbounded μπορεί να εμφανιστεί άπειρες φορές.

2.8 Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)

Πριν μιλήσουμε για το Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) πρέπει πρώτα να αναφέρουμε τι είναι μια βάση δεδομένων. Μια βάση δεδομένων είναι μια συστηματική συλλογή δεδομένων με σκοπό να διευκολύνει την διαχείριση, την οργάνωση και ανάκτηση αυτών των δεδομένων. Ένα ΣΔΒΔ ή Database Management System (DBMS) είναι μια συλλογή από προγράμματα που επιτρέπει στους χρήστες να έχουν πρόσβαση και να διαχειρίζονται τα δεδομένα των βάσεων όπως επιθυμούν. Επίσης μπορούν να δώσουν δικαιώματα πρόσβασης και σε άλλους χρήστες. Οι περισσότερες επιχειρήσεις χρησιμοποιούν τέτοια ΣΔΒΔ προφανώς για να αποθηκεύουν αλλά και να ελέγχουν και να χειρίζονται δεδομένα που είναι απαραίτητα για την ομαλή λειτουργία της επιχείρησης, για παράδειγμα κρατούν δεδομένα των πελατών τους, λογαριασμούς, τηλέφωνα επικοινωνίας και άλλα πολλά.

Σύμφωνα με το βιβλίο Database Systems: The Complete Book [3] τα ΣΔΒΔ έχουν πολλές χρήσεις και λειτουργίες μερικές από τις οποίες είναι:

- Επιτρέπουν τη δημιουργία νέων βάσεων και ορίζουν το σχήμα που θα πάρουν, δηλαδή το πως θα συσχετιστούν τα δεδομένα μεταξύ τους.
- Δίνουν στους χρήστες τη δυνατότητα να εκτελούν εντολές ή ερωτήματα (queries) μέσω της γλώσσας SQL για την οποία θα μιλήσουμε αμέσως μετά.
- Έχουν μεγάλη χωρητικότητα και μπορούν να αποθηκεύσουν τόνους δεδομένων για μεγάλα χρονικά διαστήματα και ταυτόχρονα προσφέρουν γρήγορη προσπέλαση των δεδομένων.
- Συμβάλλουν στην ανθεκτικότητα των βάσεων και στην ανάκτηση των δεδομένων σε περίπτωση κάποιου προβλήματος ή βλάβης.
- Επιτρέπουν σε πολλαπλούς χρήστες να προσπελαύνουν τα ίδια δεδομένα με τις ενέργειες κάθε χρήστη να μην επηρεάζονται από τις ενέργειες των υπολοίπων και κάθε ενέργεια να θεωρείται ολοκληρωμένη.

Τα ΣΔΒΔ έχουν εξελιχθεί αρκετά με την πάροδο του χρόνου. Τα πρώτα ΣΔΒΔ υιοθετούσαν το λεγόμενο ιεραρχικό πρότυπο, δηλαδή τα δεδομένα ήταν οργανωμένα βάσει κάποιας ιεραρχίας και έχουν σχέση γονέα-παιδιού (όπως π.χ. στην XML). Αργότερα εμφανίστηκε το δικτυακό πρότυπο στο οποίο τα δεδομένα συνδέονται μεταξύ τους με πολλές διαφορετικές σχέσεις, σχηματίζοντας την εικόνα ενός δικτύου σχέσεων. Τα πρότυπα αυτά όμως παρουσίαζαν περιορισμούς ή ήταν ανεπαρκή σε κάποιες περιπτώσεις και έτσι δημιουργήθηκε η ανάγκη για ένα καλύτερο πρότυπο.

Σύμφωνα με το σχεσιακό πρότυπο, καλύτερα γνωστό ως Σχεσιακό Μοντέλο τα δεδομένα αναπαρίστανται με τη μορφή πινάκων οι οποίοι αποτελούν και τις σχέσεις μεταξύ τους. Σε αντίθεση με το δικτυακό πρότυπο, το σχεσιακό μοντέλο διαθέτει προκαθορισμένους τύπους δεδομένων που μπορούν να χρησιμοποιηθούν και παραμένει το κυρίαρχο πρότυπο μέχρι σήμερα.

Στο σχεσιακό μοντέλο τα δεδομένα αποθηκεύονται μέσα σε πίνακες οι οποίοι έχουν γραμμές και στήλες. Κάθε στήλη ενός πίνακα έχει ένα όνομα και περιέχει έναν συγκεκριμένο τύπο δεδομένων και κάθε γραμμή του πίνακα αποτελεί μια εγγραφή. Το σχεσιακό σχήμα (relational schema) της βάσης

είναι μια απεικόνιση των σχέσεων που έχουν οι πίνακες μεταξύ τους και αυτές οι σχέσεις αναπαρίστανται με βέλη ή γραμμές που ξεκινάνε από έναν πίνακα και τελειώνουν σ' έναν άλλον. Για τη δημιουργία αυτών των σχέσεων όμως, πρέπει κάθε πίνακας να διαθέτει ένα ξεχωριστικό χαρακτηριστικό ή ένα αναγνωριστικό έτσι ώστε να διακρίνεται από τους υπόλοιπους. Αυτό το αναγνωριστικό παίρνει τη μορφή μιας στήλης και ονομάζεται κύριο κλειδί (Primary Key). Το κύριο κλειδί μπορεί να είναι μια στήλη ή συνδυασμός στηλών του πίνακα στην οποία/οποίες κάθε εγγραφή είναι μοναδική και απαγορεύεται να είναι κενή. Επιπλέον κάθε πίνακας μπορεί να έχει μόνο ένα κύριο κλειδί, αλλά δεν είναι απαραίτητο να έχουν όλοι οι πίνακες κύριο κλειδί. Με βάση αυτό, οι πίνακες πλέον μπορούν να συσχετιστούν μεταξύ τους εφόσον υπάρχει η συγκεκριμένη στήλη-κλειδί και σε άλλους πίνακες αλλά δεν αποτελεί κύριο κλειδί για τους ίδιους. Αυτή η στήλη ονομάζεται ξένο κλειδί (Foreign Key) και συνδέει τους πίνακες μεταξύ τους. Αν μια στήλη έχει μοναδικές εγγραφές αλλά μπορεί να υπάρχουν και κενές εγγραφές, μπορεί να οριστεί ως μοναδικό κλειδί (Unique Key). Ένας πίνακας μπορεί να περιέχει πολλά μοναδικά κλειδιά αλλά μόνο ένα κύριο κλειδί.

Οι σχέσεις μεταξύ των πινάκων παριστάνονται με γραμμές όπως εξηγήσαμε αλλά μπορεί να είναι πιο πολύπλοκες απ' όσο νομίζουμε. Δύο πίνακες μπορεί να έχουν σχέση "ένα-προς-πολλά" (one-to-many) που σημαίνει ότι η μοναδική εγγραφή της στήλης του πρώτου πίνακα μπορεί να εμφανίζεται πολλές φορές στη στήλη του δεύτερου πίνακα. Για παράδειγμα μια εγγραφή id ενός πίνακα Συγγραφέας μπορεί να υπάρχει πολλές φορές σ' έναν πίνακα Έργα αφού ο ίδιος συγγραφέας μπορεί να έχει γράψει πολλά έργα. Αντίστοιχα υπάρχει και η σχέση "πολλά-προς-πολλά" όπου οι εγγραφές μπορούν να υπάρχουν πολλές φορές και στους δύο πίνακες. Η σχέση "ενα-προς-ένα" δείχνει ότι μόνο μια εγγραφή συνδέεται με μία άλλη εγγραφή. Αυτές οι σχέσεις ξεχωρίζουν μεταξύ τους μέσω κατάλληλων συμβόλων πάνω στις γραμμές που συνδέουν τους πίνακες και μας βοηθούν να κατανοήσουμε καλύτερα το πως "επικοινωνούν" μεταξύ τους αλλά και να αποφύγουμε λάθη κατά το χτίσιμο των πινάκων.

Το σχεσιακό μοντέλο δεδομένων έχει λύσει αρκετά λογικά προβλήματα και φαίνεται αποτελεσματικό, αλλά αυτό δε σημαίνει ότι τα ΣΔΒΔ έχουν σταματήσει να εξελίσσονται, αφού νέες ιδέες και προτάσεις παρουσιάζονται συνεχώς για περαιτέρω βελτίωση των δεδομένων και των σχέσεων μεταξύ τους. Το πιο πρόσφατο μοντέλο είναι το Αντικειμενοστρεφές Μοντέλο, όπου τα δεδομένα παριστάνουν αντικείμενα, όπως ακριβώς στον αντικειμενοστρεφή προγραμματισμό.

2.8.1 Structured Query Language (SQL)

Η γλώσσα SQL αποτελεί βασικό εργαλείο των σχεσιακών βάσεων δεδομένων. Με την SQL μπορούμε να εκτελέσουμε εντολές για ενέργειες όπως εισαγωγής, διαγραφής, ενημέρωσης και αναζήτησης δεδομένων σε πίνακες αλλά και ενέργειες διαχείρισης αυτών των πινάκων. Όλες οι σχεσιακές βάσεις χρησιμοποιούν την SQL για την προσπέλαση των δεδομένων και η σύνταξή της μπορεί να διαφέρει ελαφρά από βάση σε βάση.

Με την SQL μπορούμε να πραγματοποιήσουμε πολλές διαφορετικές ενέργειες τόσο σε απλές γραμμές ή στήλες πινάκων, όσο και σε πίνακες ή ολόκληρες βάσεις. Ακολουθούν μερικές από τις πιο χρήσιμες ενέργειες που μπορούμε να εκτελέσουμε με ερωτήματα SQL. (Σημειώνεται ότι στο τέλος κάθε ερωτήματος πρέπει να υπάρχει ο χαρακτήρας semicolon ";"):

- Επιλογή δεδομένων από μια βάση με την εντολή SELECT [επιλογή] FROM [όνομα πίνακα]. Στο πεδίο επιλογής μπορεί να μπει το όνομα μιας ή περισσότερων στηλών ή μπορούμε να επιλέξουμε όλα τα δεδομένα με τον ειδικό χαρακτήρα "*". Η επιλογη μπορεί να γίνει πιο σύνθετη με χρήση του WHERE [συνθήκη] όπου η συνθήκη πρέπει να ικανοποιείται. Το

WHERE μπορεί να ενισχυθεί με τους λογικούς τελεστές AND (και), OR (ή) και NOT (όχι) και συνδυάζοντάς τα κατάλληλα μπορούμε να κάνουμε πιο συγκεκριμένη την επιλογή. Επίσης σε μια βάση που υπάρχουν πολλές διπλότυπες τιμές, μπορούμε να τις ξεφορτωθούμε και να κρατήσουμε τις μοναδικές διαφορετικές τιμές με την αντίστοιχη εντολή SELECT DISTINCT.

- Εισαγωγή δεδομένων σε έναν ή πολλούς πίνακες με την εντολή INSERT INTO [όνομα πίνακα ή πινάκων] VALUES [τιμή, ή τιμές]. Αν έχουμε πολλαπλά δεδομένα, (δηλαδή πίνακες και πολλές τιμές), αυτά διαχωρίζονται με κόμμα.
- Διαγραφή δεδομένων από έναν πίνακα με την εντολή DELETE FROM [όνομα πίνακα] WHERE [συνθήκη]. Με την εντολή DELETE FROM [όνομα πίνακα], χωρίς συνθήκη δηλαδή, διαγράφονται όλες οι γραμμές του πίνακα χωρίς όμως να διαγραφεί ο ίδιος.
- Ενημέρωση εγγραφών σε έναν πίνακα με την εντολή UPDATE [όνομα πίνακα] SET [όνομα στήλης πίνακα='νέα τιμή ή τιμές'] WHERE [συνθήκη]. Εδώ θα ενημερωθεί το σημείο του πίνακα που ικανοποιεί τη συνθήκη (δηλαδή το WHERE) με τη νέα τιμή στο SET.
- Δημιουργία νέου πίνακα με την εντολή CREATE TABLE [όνομα πίνακα] ([όνομα στήλης τύπος στήλης]). Εάν υπάρχουν περισσότερες από μια στήλες τότε το όνομα και ο τύπος της καθεμιάς μπαίνουν με τη σειρά διαχωρισμένα με κόμμα.
- Διαγραφή πίνακα με την εντολή DROP [όνομα πίνακα].
- Επεξεργασία στηλών πίνακα με την εντολή ALTER TABLE [όνομα πίνακα]. Η προσθήκη στηλών γίνεται με το ADD [όνομα στήλης τύπος στήλης] αμέσως μετά το ALTER και ανάλογα η διαγραφή στήλης γίνεται με το DROP COLUMN [όνομα στήλης].

Πέρα από αυτές τις απλές εντολές η SQL έχει και άλλες δυνατότητες όπως της ένωσης πινάκων. Οι ενώσεις (JOINS) γίνονται με βάση κάποιο κοινό "κλειδί" ανάμεσα στους πίνακες. Το κλειδί αυτό είναι μια στήλη η οποία έχει το ίδιο όνομα στους συγκεκριμένους πίνακες. Υπάρχουν 4 είδη ενώσεων τα οποία είναι:

- INNER JOIN: Εμφανίζει μόνο τα κοινά στοιχεία μεταξύ των πινάκων με βάση το κύριο κλειδί.
- LEFT JOIN: Συγχωνεύει όλα τα περιεχόμενα του αριστερού πίνακα και τα κοινά στοιχεία μεταξύ των πινάκων.
- RIGHT JOIN: Συγχωνεύει όλα τα περιεχόμενα του δεξιού πίνακα και τα κοινά στοιχεία μεταξύ των πινάκων.
- OUTER JOIN: Συγχωνεύει τα περιεχόμενα των πινάκων εφόσον υπάρχουν κοινά στοιχεία.

Άλλες αξιοσημείωτες εντολές είναι το ORDER BY και το GROUP BY. Το ORDER BY τοποθετείται στο τέλος του SQL ερωτήματος και ταξινομεί το αποτέλεσμα με άυξουσα (ASC) ή φθίνουσα σειρά (DESC). Η default επιλογή είναι ASC οπότε άμα δεν γράψουμε τίποτα μετά το ORDER BY θεωρείται δεδομένη η αύξουσα ταξινόμηση. Το GROUP BY τοποθετείται επίσης στο τέλος του ερωτήματος και χρησιμοποιείται για την κατηγοριοποίηση δεδομένων βάσει μιας στήλης του πίνακα. Επίσης χρησιμοποιείται συχνά με μαθηματικές συναρτήσεις όπως SUM (άθροισμα συνόλου), MAX (εύρεση μεγίστου), MIN (εύρεση μικρότερου), COUNT (σύνολο) και AVG (υπολογισμός μέσης τιμής).

Έχοντας καταλάβει πλέον τις δυνατότητες της γλώσσας SQL θα μιλήσουμε για το ΣΔΒΔ που θα χρησιμοποιηθεί για τη δημιουργία βάσης δεδομένων και σχεσιακών πινάκων τους οποίους θα γεμίσουμε με θεατρικά έργα.

2.8.2 MySQL

Η MySQL είναι, αν όχι το πιο δημοφιλές, ένα από τα πιο δημοφιλή συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS) στον κόσμο [13]. Είναι λογισμικό ανοιχτού κώδικα (open source) που σημαίνει ότι μπορεί ο καθένας να το χρησιμοποιήσει, είναι cross-platform δηλαδή λειτουργεί σε όλες τις πλατφόρμες και είναι ιδανικό τόσο για μεγάλα όσο και μικρά project που χρησιμοποιούν βάσεις δεδομένων. Αναπτύχθηκε αρχικά από την εταιρία MySQL AB και σήμερα ανήκει και διανέμεται δωρεάν από την Oracle Corporation.

Η MySQL λειτουργεί σύμφωνα με το μοντέλο πελάτη-εξυπηρετητή (client-server). Αρχικά δημιουργεί μια βάση δεδομένων για την αποθήκευση και διαχείριση δεδομένων, ενώ παράλληλα ορίζει τις σχέσεις μεταξύ των πινάκων της βάσης. Ο client εκτελεί SQL ερωτήματα με τη μορφή αιτημάτων τα οποία στέλνονται στο server και αυτός αποκρίνεται εμφανίζοντας το αποτέλεσμα στον client. Η MySQL τρέχει μέσω της γραμμής εντολών του υπολογιστή, αλλά διαθέτει και δική της διεπαφή γραμμής εντολών, το MySQL shell για την εκτέλεση εντολών και λειτουργιών στο διαχειριστικό κομμάτι και υποστηρίζει τις γλώσσες SQL, Python και Javascript. Διαθέτει ακόμα γραφική διεπαφή χρήστη (GUI) το MySQL Workbench για τους χρήστες που δεν είναι εξοικειωμένοι με τη γραμμή εντολών. Με το MySQL Workbench μπορεί κανείς να δημιουργήσει σύνδεση με μια βάση δεδομένων, να διαχειριστεί τις βάσεις και τους πίνακες, να εκτελέσει SQL ερωτήματα και να παράξει διαγράμματα από το σχήμα της βάσης μέσω του reverse engineering. Τις παραπάνω ενέργειες θα εφαρμόσουμε στην πράξη στο επόμενο κεφάλαιο για τις ανάγκες της Π.Ε. χρησιμοποιώντας την τελευταία έκδοση του MySQL Workbench 8.0.

2.9 Tablesaw

Το Tablesaw είναι μια open source βιβλιοθήκη για ανάλυση δεδομένων σε Java και θα την χρησιμοποιήσουμε στην εφαρμογή της Π.Ε. για την ανάλυση των θεατρικών έργων και την παραγωγή οπτικών γραφημάτων. Απαραίτητη προϋπόθεση για το χειρισμό των δεδομένων είναι αυτά να περιέχονται σε μορφή στηλών ή πινάκων. Το Tablesaw ενσωματώνει τη βιβλιοθήκη Plot.ly της Javascript για την οπτικοποίηση των δεδομένων με γραφήματα όπως πίτες (pie charts), ραβδογράμματα (bar charts), θηκογράμματα (box plots), ιστογράμματα (histograms) και άλλα πολλά. Στη σελίδα της βιβλιοθήκης στο Github [14] αναφέρονται οι εξής δυνατότητες για χειρισμό και επεξεργασία των δεδομένων:

- Εισαγωγή δεδομένων από συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων RDBMS, CSV, Excel, HTML αρχεία ή απλά αρχεία κειμένου (text files) που είναι κατάλληλα διαμορφωμένα. Τα αρχεία αυτά μπορούν να βρίσκονται τοπικά στον υπολογιστή ή απομακρυσμένα.
- Εξαγωγή δεδομένων σε ανάλογα CSV, JSON, HTML και αρχεία κειμένου.
- Επεξεργασία γραμμών και στηλών των πινάκων με προσθήκες και διαγραφές.
- Συνδυασμός πινάκων αλλά και φιλτράρισμα με χρήση των κατάλληλων συνθηκών.
- Ταξινόμιση και ομαδοποίηση των αποτελεσμάτων σε πίνακες.

- Υπολογισμός περιγραφικών στατιστικών όπως μεγίστου (max), μικρότερου (min), μέσης τιμής (mean), διάμεσου (median), διακύμανσης (variance), τυπικής απόκλισης (standard deviation) και άλλα πολλά.
- Διαχείριση κενών τιμών σε εγγραφές πινάκων.

Επιπλέον στο Documentation διαθέτει Οδηγό Χρήσης [15] με λεπτομερή αναφορά των μεθόδων και των λειτουργιών τους και χρήσιμα παραδείγματα για όποιον ενδιαφέρεται να ασχοληθεί με το αντικείμενο.

2.10 Επίλογος

Σε αυτό το κεφάλαιο μιλήσαμε για τη γλώσσα προγραμματισμού Java η οποία αποτελεί το βασικό θεμέλιο για το χτίσιμο της εφαρμογής της Π.Ε. και αναλύσαμε τις τεχνολογίες που σε συνεργασία μεταξύ τους θα συνθέσουν το τελικό αποτέλεσμα της εφαρμογής και τις βασικές της λειτουργίες. Έχοντας κατανοήσει αυτές τις έννοιες θα προχωρήσουμε τώρα στο κομμάτι της υλοποίησης βήμα προς βήμα, εφαρμόζοντάς τες στην πράξη και θα παραθέσουμε σχετικά τμήματα κώδικα και εικόνες.

Κεφάλαιο 3ο: Υλοποίηση Εφαρμογής

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα ξεκαθαρίσουμε τις απαιτήσεις της εφαρμογής που θα αναπτυχθεί και θα αναλύσουμε τη μεθοδολογία που λαμβάνεται υπόψην για την υλοποίησή της. Παράλληλα θα εμβαθύνουμε στο κομμάτι της λειτουργικότητας και θα εξοικειωθούμε με τεχνικές λεπτομέρειες που αφορούν την υλοποίησή της.

3.2 Ανάλυση Απαιτήσεων

Αρχικά πριν ξεκινήσουμε να χτίζουμε την εφαρμογή, πρέπει να ορίσουμε τις δυνατότητες που θα έχει, το πώς θα λειτουργεί και πώς θα πετύχει τους στόχους που έχουμε θέσει. Στην περίπτωσή μας η εφαρμογή που θα υλοποιηθεί θα έχει τις εξής λειτουργίες:

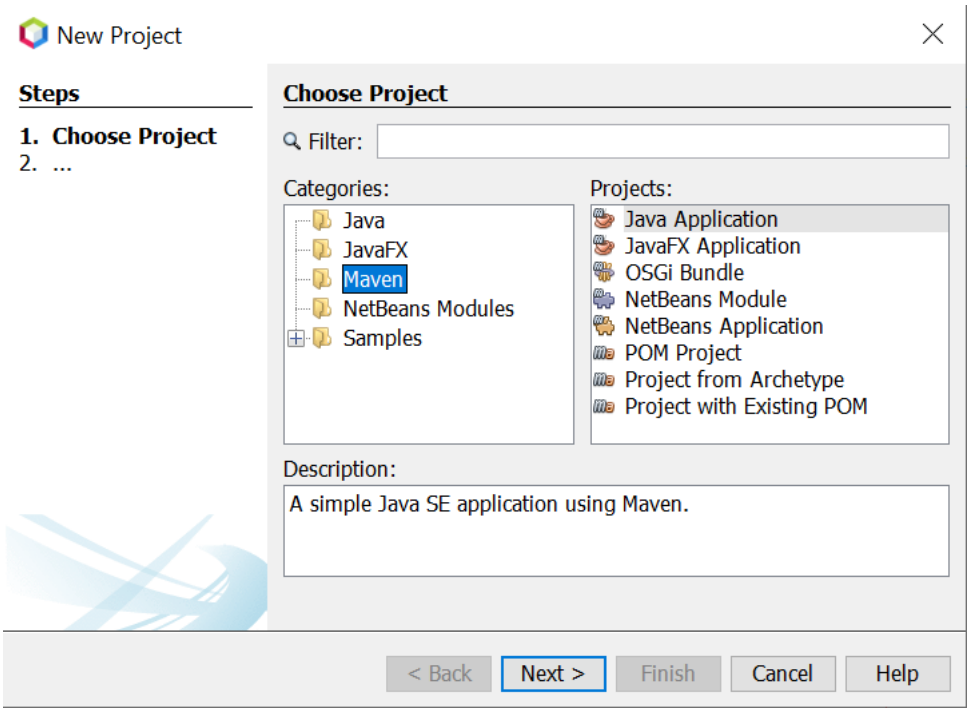
- Θα δέχεται ένα θεατρικό έργο σαν μια παράμετρο εισόδου
- Θα φορτώνει το έργο μέσα σε ένα πλαίσιο όπου ο χρήστης θα μπορεί να διαβάσει το έργο
- Θα επεξεργάζεται αυτό το έργο και θα κρατά συγκεκριμένα δεδομένα που μας ενδιαφέρουν
- Θα αποθηκεύει αυτά τα δεδομένα σε μια οργανωμένη μορφή η οποία θα προβάλλεται δίπλα στο έργο
- Θα πραγματοποιείται έλεγχος της συγκεκριμένης μορφής για την εγκυρότητα των δεδομένων
- Θα δίνεται η δυνατότητα αποθήκευσης της δομής
- Θα παρουσιάζονται βασικές πληροφορίες του έργου βάσει των δεδομένων που συλλέχθηκαν

- Τα έργα θα επεξεργάζονται κατάλληλα και θα εισάγονται σε μια βάση δεδομένων για αποθήκευση μέσα στους κατάλληλους πίνακες
- Θα πραγματοποιείται ανάλυση των δεδομένων των έργων βάσει των δεδομένων τους
- Τα αποτελέσματα της ανάλυσης θα παρουσιάζονται με τη μορφή γραφημάτων

Έχοντας ξεκαθαρίσει αυτές τις απαιτήσεις, μπορούμε να προχωρήσουμε στην υλοποίηση της εφαρμογής όπου θα συναντήσουμε κάθε μια από αυτές τις λειτουργίες.

3.3 Σχεδιασμός Εφαρμογής

Η εφαρμογή υλοποιείται σε Java Swing με το περιβάλλον ανάπτυξης NetBeans [16]. Για να δημιουργήσουμε ένα νέο project επιλέγουμε File -> New Project, επιλέγουμε τη Maven στα Categories και Java Application στα Projects. Η δημιουργία του νέου project φαίνεται στην Εικόνα 3.1:



Εικόνα 3.1: Δημιουργία νέου project με Maven στο NetBeans

Αφού πατήσουμε Next, θα μας ζητηθεί να δώσουμε όνομα στο project και να επιλέξουμε το φάκελο στον οποίο θα αποθηκευτεί. Δίνοντας το όνομα που θέλουμε πατάμε Finish για να δημιουργήσουμε το project.

Στα αριστερά του περιβάλλοντος μπορούμε να δούμε τα project που υπάρχουν στην αντίστοιχη καρτέλα με το όνομα Projects. Εκεί θα βρούμε το project που μόλις δημιουργήσαμε και μέσα στο φάκελό του βλέπουμε υποφακέλους όπως το Source Packages που περιέχει το ίδιο το project, τα Java Dependencies, δηλαδή εξαρτήσεις από άλλες κλάσεις και μέσα πρέπει να υπάρχει το Java Development Kit (JDK). Το JDK περιέχει όλα τα απαραίτητα εργαλεία για να τρέξουμε προγράμματα σε Java όπως το Java Runtime Environment JRE, τον διερμηνέα της Java, τον μεταγλωττιστή javac και άλλα. Τέλος,

δημιουργείται ο φάκελος Project Files που περιέχει το αρχείο pom.xml. Στην Εικόνα 3.2 φαίνεται το αρχείο pom.xml της εφαρμογής με τις βιβλιοθήκες (dependecies) που χρησιμοποιούνται για τις ανάγκες της εφαρμογής:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.LiteratureAnalyticsThesis</groupId>
    <artifactId>LiteratureAnalyticsThesis</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <name>LiteratureAnalyticsThesis</name>

    <dependencies>
        <dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>1.11.3</version>
        </dependency>
        <dependency>
            <groupId>tech.tablesaw</groupId>
            <artifactId>tablesaw-core</artifactId>
            <version>0.38.1</version>
        </dependency>
        <dependency>
            <groupId>tech.tablesaw</groupId>
            <artifactId>tablesaw-jsplot</artifactId>
            <version>0.38.1</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.25</version>
        </dependency>
        <dependency>
            <groupId>nz.ac.waikato.cms.weka.thirdparty</groupId>
            <artifactId>bounce</artifactId>
            <version>0.18</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Εικόνα 3.2: Το αρχείο pom.xml της εφαρμογής

Ανοίγοντας το φάκελο Source Packages, βρίσκουμε το project και κάνοντας δεξί κλικ σ' αυτό επιλέγουμε New -> JFrame Form και στη συνέχεια ονομάζουμε το όνομα της κλάσης του project και πατάμε Finish. Με αυτή την ενέργεια, δημιουργείται ένα JFrame μέσα στο οποίο θα προσθέσουμε τα Swing στοιχεία που θέλουμε.

Για αρχή μπορούμε να αλλάξουμε το μέγεθος του Frame τραβώντας το από την άκρη με το ποντίκι μέχρι να καταλήξουμε στο μέγεθος που μας συμφέρει. Μέσα στο Frame θα προσθέσουμε 4 JPanels όπου το καθένα θα περιέχει στοιχεία για διαφορετικές λειτουργίες. Έτσι η εφαρμογή μας θα χωριστεί σε αυτά τα 4 κομμάτια, με το καθένα να αντιστοιχεί σε ένα JPanel:

- TopPanel: Το Top Panel περιέχει 3 κουμπιά (JButtons), ένα για εισαγωγή έργων, ένα για αποθήκευση έργων σε XML μορφή και ένα για εισαγωγή τους στη Βάση Δεδομένων.

- LeftPanel: Το LeftPanel θα περιέχει ένα πλαίσιο (JEditorPane) μέσα στο οποίο θα φορτώνεται το θεατρικό έργο και θα διαθέτει μηχανισμό κύλισης (JScrollPane). Θα προσθέσουμε επίσης μια ετικέτα (JLabel) με τίτλο HTML για το Panel.
- RightPanel: Παρόμοια με το LeftPanel, θα περιέχεται ένα πλαίσιο στο οποίο θα εμφανίζεται ένα XML έγγραφο με τα δεδομένα του έργου και θα υπάρχει μηχανισμός κύλισης (JScrollPane). Αντίστοιχα θα προσθέσουμε μια ετικέτα (JLabel) με τον τίτλο XML.
- BottomPanel: Θα περιέχει ένα κουμπί ανάλυσης (JButton) και μερικές drop-down λίστες (JComboBox) και μία ετικέτα (JLabel) που θα δηλώνει το όνομα και κατ' επέκταση τη λειτουργία για κάθε λίστα. Τέλος θα υπάρχουν 2 κουτάκια επιλογής (JCheckBox) για επιπλέον χρηστικότητα.

Μπορούμε ακόμα να διαμορφώσουμε περαιτέρω τα στοιχεία μας κάνοντας δεξί κλικ πάνω τους και επιλέγοντας τα Properties. Εκεί περιέχονται όλες οι ιδιότητες για το κάθε στοιχείο και μπορούμε να εκτελέσουμε ενέργειες όπως να αλλάξουμε το όνομα του στοιχείου, να του αλλάξουμε χρώμα, κλπ. Για παράδειγμα θα προσθέσουμε ένα TitledBorder στο TopPanel από το πεδίο Border και θα δώσουμε τον τίτλο "Ανάλυση Δεδομένων Θεατρικών Έργων". Ανάμεσα στα LeftPanel και RightPanel θα προστεθεί ένας διαχωριστής (JSplitPane) ο οποίος θα μας δώσει τη δυνατότητα να αλλάζουμε διαστάσεις οριζόντια στα 2 αυτά πλαίσια. Έχοντας προσθέσει όλα αυτά τα στοιχεία, το επόμενο βήμα είναι να τους δώσουμε λειτουργικότητα για να μπορεί κάποιος να τα χρησιμοποιήσει κατάλληλα.

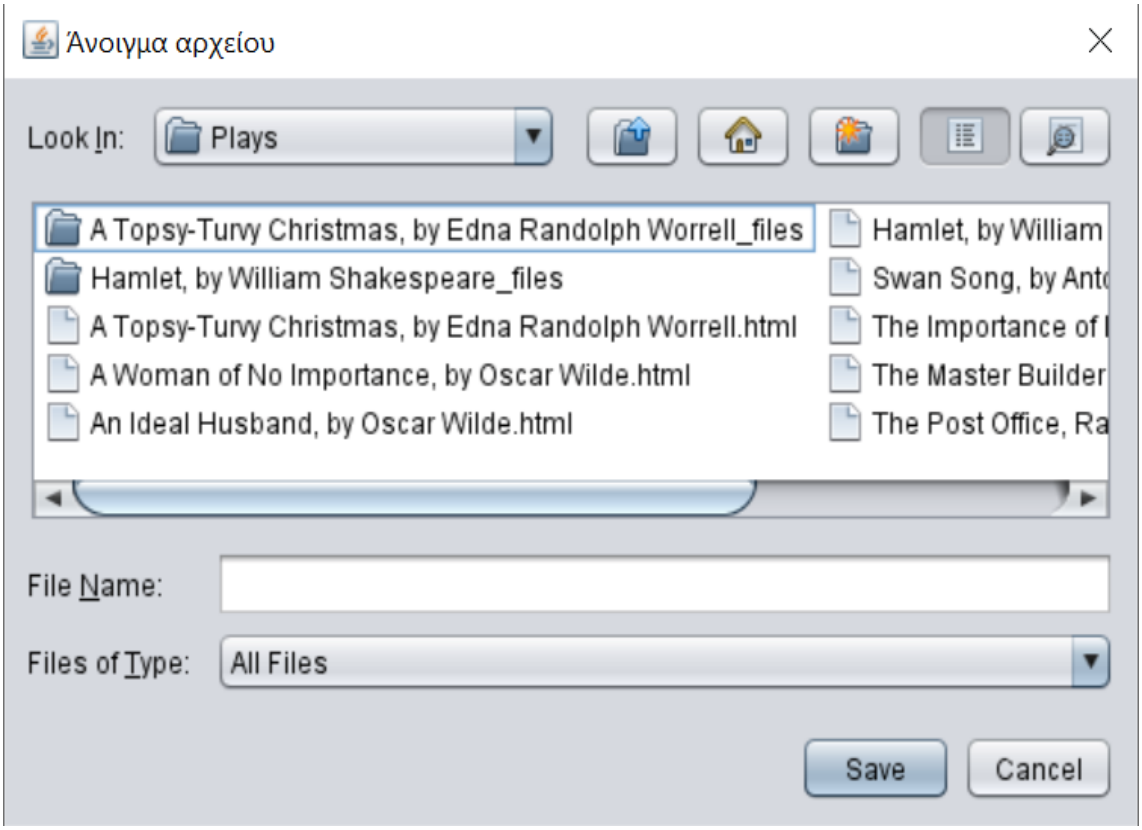
3.4 Είσοδος Θεατρικών Έργων

Όπως είπαμε η εφαρμογή θα δέχεται σαν είσοδο θεατρικά έργα τα οποία στη συνέχεια θα τα φορτώνει σε ένα πλαίσιο στα αριστερά της εφαρμογής. Τα έργα αυτά λαμβάνονται σε HTML από την ηλεκτρονική βιβλιοθήκη Project Gutenberg [17].

Για να κάνουμε ένα κουμπί λειτουργικό όταν το πατάμε πρέπει να το συνδέσουμε με ένα γεγονός (event). Εφόσον πρόκειται για πάτημα κουμπιού το γεγονός που μας ενδιαφέρει είναι το Action που σημαίνει ότι το κουμπί θα πατηθεί για να ενεργοποιηθεί το γεγονός. Έτσι λοιπόν, κάνουμε δεξί κλικ στο κουμπί Εισαγωγή, και επιλέγουμε Events -> Action -> actionPerformed. Με αυτή την ενέργεια δημιουργείται η μέθοδος ImportButtonActionPerformed() που δέχεται ως παράμετρο αυτό το γεγονός. Μέσα σ' αυτήν θα γράψουμε τον κώδικα για τις ενέργειες που γίνονται με το πάτημα αυτού του κουμπιού.

3.4.1 Παράθυρο Επιλογής Αρχείου

Το πρώτο πράγμα που συμβαίνει όταν πατηθεί το κουμπί εισόδου είναι να εμφανιστεί το Παράθυρο Επιλογής αρχείου. Με αυτό ο χρήστης μπορεί να πλοηγηθεί στο σύστημα αρχείων του υπολογιστή και να επιλέξει τοπικά το θεατρικό έργο. Το Παράθυρο Επιλογής Εισόδου υλοποιείται με το στοιχείο JFileChooser το οποίο παίρνει το αρχείο του έργου αλλά και τη διαδρομή του αρχείου. Στην Εικόνα 3.3 φαίνεται το Παράθυρο Επιλογής αρχείου:



Εικόνα 3.3: Παράθυρο Επιλογής αρχείου

Με το που επιλεχθεί ένα έργο εμφανίζεται ένα μήνυμα που ζητάει απ' το χρήστη να εισάγει το ID του έργου. Ο χρήστης πληκτρολογεί έναν ακέραιο αριθμό ο οποίος αποθηκεύεται σε μια μεταβλητή και θα χρησιμοποιηθεί σε επόμενο στάδιο.

3.4.2 HTML Scraping

Εφόσον επιλεχτεί ένα θεατρικό έργο ξεκινάει η διαδικασία της συλλογής δεδομένων (scraping) από το HTML αρχείο του. Η βιβλιοθήκη που χρησιμοποιείται για αυτή τη διαδικασία είναι το JSoup, το οποίο έχει προστεθεί στο pom.xml ως dependency. Αφού το προσθέσουμε, αποθηκεύουμε το project και εκτελούμε την ενέργεια Build Project του NetBeans. Με αυτή την ενέργεια, η Maven κατεβάζει και εγκαθιστά τη βιβλιοθήκη για να τη χρησιμοποιήσουμε.

Τώρα μπορούμε να ξεκινήσουμε τη συλλογή δεδομένων. Πριν ξεκινήσουμε όμως, πρέπει να μελετήσουμε τα HTML elements που υπάρχουν μέσα στο αρχείο του έργου. Με άλλα λόγια, να δούμε τι elements χρησιμοποιούνται για τον τίτλο του έργου, τον συγγραφέα, τους χαρακτήρες, τους διαλόγους, κλπ. Είναι απαραίτητο να γνωρίζουμε τι μορφή έχει αυτό που θέλουμε να τραβήξουμε και να συλλέξουμε. Τα έργα δεν χρησιμοποιούν όλα τα ίδια HTML elements για τα ίδια δεδομένα, (π.χ. οι διάλογοι σε ένα έργο μπορεί να είναι μέσα σε <p> elements και σε άλλο έργο να είναι σε) και γι'

αυτό το λόγο είναι σχεδόν αδύνατο το πρόγραμμα που θα γράψουμε να λειτουργεί σωστά για όλους τους διαφορετικούς συνδυασμούς των elements. Έτσι λοιπόν αυτή η διαδικασία μπορεί να χαρακτηριστεί ως ημι-αυτόματη γιατί μπορεί να δουλεύει σωστά τραβώντας συγκεκριμένα elements με συγκεκριμένη δομή αλλά σε άλλα να μπορεί μην δουλεύει.

Για να ξεκινήσουμε τη διαδικασία scraping από ένα αρχείο γράφουμε την εντολή Jsoup.parse() που δέχεται το αρχείο ως παράμετρο. Στην Εικόνα 3.4 βλέπουμε ένα απλό παράδειγμα scraping όπου τραβάμε τον τίτλο του έργου:

```
org.jsoup.nodes.Document scrape = Jsoup.parse(myFile, "UTF-8", "");
org.jsoup.nodes.Element title = scrape.select("h1").first();
System.out.println(title.text() + "\n");
```

Εικόνα 3.4: Παράδειγμα συλλογής Τίτλου θεατρικού έργου

Σε αυτό το παράδειγμα βλέπουμε ότι ο τίτλος είναι κεφαλίδα <h1> και τον συλλέγουμε τραβώντας το πρώτο <h1> element μέσα στο αρχείο. Αυτή είναι μια φαινομενικά πολύ απλή συνθήκη αλλά μερικές φορές τα elements, ιδίως στους διαλόγους, μπορεί να αποδειχθούν ιδιαίτερα δύσκολα για να τα συλλέξουμε.

3.4.3 Δημιουργία XML εγγράφου

Παράλληλα με τη συλλογή δεδομένων δημιουργούμε ένα XML έγγραφο μέσα στο οποίο θα αποθηκεύσουμε τα δεδομένα που τραβάμε από το scraping. Ο λόγος που γίνεται αυτό είναι για να έχουν τα δεδομένα μας μια οργανωμένη μορφή. Η δημιουργία του XML εγγράφου στη Java γίνεται με την κλάση DocumentBuilder. Το έγγραφο που θα δημιουργήσουμε θα το ονομάσουμε document. Αφού δημιουργήσουμε το document μπορούμε στη συνέχεια να ορίσουμε XML elements και να τα εισάγουμε στο document. Στην εικόνα 3.5 δίνεται παράδειγμα δημιουργίας του document όπως και του root element Έργα στο οποίο θα προστεθούν όλα τα υπόλοιπα στοιχεία:

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
document = documentBuilder.newDocument();

//Create Root element
rootElement = document.createElement("Εργα");
document.appendChild(rootElement);
```

Εικόνα 3.5: Δημιουργία document και root Element

Αφού δημιουργήσουμε το root Element, η διαδικασία που ακολουθείται είναι scraping για συλλογή, δημιουργία νέων child elements, εισαγωγή των συλλεγμένων δεδομένων στο child element και τελικά προσθήκη στο parent element.

Τα δεδομένα που συλλέγουμε και αποθηκεύουμε στο XML έγγραφο έχουν σαφώς μια ιεραρχία και γι' αυτό το λόγο το XML είναι ιδανικό σε αυτή την περίπτωση. Πιο αναλυτικά, τα δεδομένα που συλλέγουμε από το HTML έργο είναι:

- Τίτλος
- Συγγραφέας
- Περιεχόμενα
- Εισαγωγή
- Πράξεις
- Σκηνές
- Αποσπάσματα
- Χαρακτήρες
- Διάλογοι
- Σκηνικές Οδηγίες

Προφανώς δεν διαθέτουν όλα τα έργα το καθένα από αυτά τα δεδομένα π.χ. Περιεχόμενα, Εισαγωγή ή Σκηνές μπορεί να μην υπάρχουν, αλλά τα βασικά όπως Τίτλος, Συγγραφέας, Πράξεις, Χαρακτήρες και Διάλογοι υπάρχουν σε όλα και είναι απαραίτητα για την εφαρμογή. Όσον αφορά την ιεραρχία των δεδομένων στο XML document η ρίζα (root) είναι το Έργα. Κάτω από αυτό βρίσκεται το element Έργο (1ο επίπεδο). Μέσα στο Έργο βρίσκονται ο Τίτλος, ο Συγγραφέας, τα Περιεχόμενα, η Εισαγωγή και οι Πράξεις (2ο επίπεδο). Μέσα στις Πράξεις υπάρχουν οι Σκηνές (3ο επίπεδο) και μέσα στις Σκηνές υπάρχουν τα Αποσπάσματα και οι Χαρακτήρες (4ο επίπεδο). Τέλος, κάτω από τους Χαρακτήρες υπάρχουν οι Διάλογοι και οι Σκηνικές Οδηγίες (5ο επίπεδο). Στην εικόνα 3.6 έχουμε ένα πιο σύνθετο παράδειγμα με τη συλλογή ανάμεσα σε Αποσπάσματα (Excerpt) και Χαρακτήρες (Character) και αντίστοιχη δημιουργία element και έπειτα εισαγωγή του στο parent element Πράξη (Act):

```
//EXCERPT
if (act.tagName().equals("pre")) {
    System.out.println("Excerpt: " + act.text() + "\n");
    String ex = act.text().replaceAll("\\[", "").replaceAll("\\]", "").replaceAll("\r*\n*", "").replaceAll("       ", "");
    Excerpt = document.createElement("Απόσπασμα");
    Excerpt.appendChild(document.createTextNode(ex));
    Act.appendChild(Excerpt);
}
//CHARACTER
if (act.tagName().equals("p")) {
    if (act.text().contains("[")) {
        //Split the string into Stage Directions and Speech
        String[] parts = act.text().split("(?<=])|(?=\\[)");
        for (String s : parts) {
            //STAGE DIRECTIONS
            if (s.contains("[")) {
                s = s.trim();
                System.out.println("Stage Directions: " + s + "\n");
                StageDirections = document.createElement("Σκηνικές_Οδηγίες");
                StageDirections.appendChild(document.createTextNode(s.replaceAll("\\[", "").replaceAll("\\]", "")));
                Character.appendChild(StageDirections);
                Act.appendChild(Character);
            } else {
                //SPEECH
                s = s.replaceAll("—", "").trim();
                System.out.println("Speech: " + s + "\n");
                Speech = document.createElement("Λόγος");
                Speech.appendChild(document.createTextNode(s));
                Character.appendChild(Speech);
                Act.appendChild(Character);
            }
        }
```

Εικόνα 3.6: Scraping Αποσπαμάτων και Χαρακτήρων και εισαγωγή στην Πράξη

Εδώ βλέπουμε ότι τα Αποσπάσματα είναι μέσα σε <pre> elements και οι Χαρακτήρες είναι μέσα σε <p> elements τα οποία βρίσκονται μέσα σε μια Πράξη και τα διαχωρίζουμε με βάση αυτή τη συνθήκη. Μέσα στο κείμενο των Χαρακτήρων υπάρχει Διάλογος και μπορεί να υπάρχουν Σκηνικές Οδηγίες τις οποίες εντοπίζουμε με το χαρακτήρα "[". Αν υπάρχει αυτός ο χαρακτήρας τότε αυτό το κομμάτι του κειμένου ανήκει στις Σκηνικές Οδηγίες. Με ανάλογες τεχνικές συλλέγουμε και αποθηκεύουμε τα υπόλοιπα δεδομένα. Όταν φτάσουμε στο τέλος των δεδομένων προσθέτουμε το Έργο element μέσα στο rootElement δηλαδή τα Έργα που είναι το κορυφαίο στοιχείο.

Για να ολοκληρώσουμε ένα XML έγγραφο στη Java το μεταμορφώνουμε με την κλάση Transformer και στη συνέχεια μπορούμε να το μετατρέψουμε σε μία εύχρηστη μορφή όπως String. Επίσης με την κλάση Transformer μπορούμε να δώσουμε και κάποιες ιδιότητες στο έγγραφο. Στην Εικόνα 3.7 βλέπουμε τη μετατροπή του εγγράφου σε String:

```
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
//Xml encoding property
transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
//Xml indentation property
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
DOMSource source = new DOMSource(document);

//Convert to String
StringWriter writer = new StringWriter();
StreamResult result = new StreamResult(writer);
transformer.transform(source, result);
//Print output to string
String output = writer.toString().replaceAll("□", "-").replaceAll("&amp;", "");
```

Εικόνα 3.7: Μετατροπή document σε String

Έχοντας το XML έγγραφο σε μορφή String μπορούμε να το γράψουμε για λόγους ευχρηστίας σ' ένα απλό αρχείο με την κλάση PrintWriter. Στην Εικόνα 3.8 βλέπουμε σχετικό παράδειγμα:

```
//Write XML to file
PrintWriter pw = null;
try {
    //Clear file contents
    pw = new PrintWriter(new OutputStreamWriter(new FileOutputStream(myXMLFile, false), StandardCharsets.UTF_8));
    //Write to file
    pw = new PrintWriter(new OutputStreamWriter(new FileOutputStream(myXMLFile, true), StandardCharsets.UTF_8));
    pw.println(output);
} catch (FileNotFoundException ex) {
    ex.printStackTrace();
}
pw.flush();
pw.close();
```

Εικόνα 3.8: Εγγραφή XML document σε αρχείο

3.4.4 Φόρτωση Θεατρικού Έργου και XML Εγγράφου

Μόλις τελειώσει η διαδικασία συλλογής δεδομένων και ολοκληρωθεί το έγραφο, πρέπει να εμφανιστούν στα 2 πλαίσια της εφαρμογής το έργο στα αριστερά και το XML έγγραφο με τα συλλεγμένα δεδομένα του έργου στα δεξιά. Για να εμφανίσουμε περιεχόμενο HTML στο αριστερό πλαίσιο (HTMLEditorPane) όπως θα βλέπαμε HTML σε έναν περιηγητή, προσθέτουμε μέσα σ' αυτό το αντικείμενο HTMLEditorKit και έπειτα δημιουργούμε ένα αντικείμενο τύπου Document μέσα απ' το HTMLEditorKit. Μετά, παίρνουμε απλά την ωμή HTML με τη μέθοδο html() του Jsoup και την τοποθετούμε σε μια String μεταβλητή και τέλος χρησιμοποιούμε τη μέθοδο setText() του HTMLEditorPane για να εμφανίσουμε το έργο. Στην εικόνα 3.9 φαίνεται το αντίστοιχο τμήμα κώδικα:

```
HTMLEditorKit kit = new HTMLEditorKit();
HTMLEditorPane.setEditorKit(kit);
Document doc = kit.createDefaultDocument();
HTMLEditorPane.setDocument(doc);

//Set HTML on  HTMLEditorPane
String text = Jsoup.parse(myFile, "UTF-8", "").html();
HTMLEditorPane.setText(text);
HTMLEditorPane.setVisible(true);
```

Εικόνα 3.9: Εμφάνιση Θεατρικού Έργου στα αριστερά

Με παρόμοιο τρόπο φορτώνουμε το XML έγγραφο στο δεξιό πλαίσιο (XMLEditorPane). Δημιουργούμε ένα αντικείμενο XMLEditorKit και το προσθέτουμε στο XMLEditorPane. Στη συνέχεια, προσθέτουμε τη String μορφή του XML εγγράφου από την προηγούμενη ενότητα στο XMLEditorPane με τη μέθοδο setText(). Επιπλέον μπορούμε να δώσουμε ιδιότητες στο πλαίσιο π.χ. εάν υπάρχει συντακτικό λάθος στην XML και ένα element δεν έχει κλείσει κατάλληλα εμφανίζει μια κόκκινη γραμμή στο σχετικό σημείο για ένδειξη λάθους. Στην Εικόνα 3.10 βλέπουμε το αντίστοιχο τμήμα κώδικα για τη φόρτωση του XML εγγράφου:

```
//Create XMLEditorKit
XMLEditorKit xmlkit = new XMLEditorKit();
XMLEditorPane.setEditorKit(xmlkit);
//Load file on XMLEditorPane
XMLEditorPane.setText(output);
//Set the font style
XMLEditorPane.setFont(new Font("Courier", Font.PLAIN, 12));
//Set the tab size
XMLEditorPane.getDocument().putProperty(PlainDocument.tabSizeAttribute, new Integer(4));
//Enable auto indentation.
xmlkit.setAutoIndentation(true);
//Enable tag completion.
xmlkit.setTagCompletion(true);
//Enable error highlighting.
XMLEditorPane.getDocument().putProperty(XMLEditorKit.ERROR_HIGHLIGHTING_ATTRIBUTE, new Boolean(true));
//Set a style
xmlkit.setStyle(XMLStyleConstants.ATTRIBUTE_NAME, new Color(255, 0, 0), Font.BOLD);
RightPanel.add(new XMLFoldingMargin(XMLEditorPane), BorderLayout.EAST);
RightPanel.add(new LineNumberMargin(XMLEditorPane), BorderLayout.WEST);
```

Εικόνα 3.10: Φόρτωση XML εγγράφου στα δεξιά

Με τη φόρτωση του κειμένου του θεατρικού έργου στα αριστερά και του XML εγγράφου στα δεξιά έχει τελειώσει η διαδικασία της Εισαγωγής Θεατρικού Έργου.

3.5 Έλεγχος XML εγγράφου

Προκειμένου τα δεδομένα στο έγγραφό μας να είναι ορθά και να έχουν τη σωστή δομή θα χρησιμοποιήσουμε ένα XML schema ή XSD στο οποίο θα ορίσουμε ακριβώς πως πρέπει να είναι οργανωμένα τα δεδομένα. Ο έλεγχος αυτός γίνεται με το κουμπί με το αντίστοιχο όνομα Έλεγχος. Εφόσον το έγγραφο που δημιουργήθηκε είναι έγκυρο, ενεργοποιούνται οι υπόλοιπες δυνατότητες της εφαρμογής με τα αντίστοιχα κουμπιά να γίνονται enabled. Στην Εικόνα 3.11 παρουσιάζεται το XML schema που χρησιμοποιείται για την εγκυρότητα του εγγράφου:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Έργα">
    <xs:complexType>
    <xs:sequence>
        <xs:element name="Έργο">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Τίτλος" type="xs:string"/>
                <xs:element name="Συγγραφέας_ID" type="xs:integer"/>
                <xs:element name="Είδος" type="xs:string"/>
                <xs:element name="Ημερομηνία_Έκδοσης" type="xs:string"/>
                <xs:element name="Συγγραφέας">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Όνομα" type="xs:string" maxOccurs="1"/>
                            <xs:element name="Ημερομηνία_Γέννησης" type="xs:string" minOccurs="0"/>
                            <xs:element name="Βιογραφία" type="xs:string" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:integer"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Περιεχόμενα" type="xs:string"/>
                <xs:element name="Εισαγωγή" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:choice maxOccurs="unbounded">
                                <xs:element name="Σχόλιο" type="xs:string" maxOccurs="unbounded"/>
                                <xs:element name="Απόσπασμα" type="xs:string" maxOccurs="unbounded"/>
                            </xs:choice>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Χαρακτήρες" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="Πράξη" minOccurs="1" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:choice minOccurs="0" maxOccurs="unbounded">
                                <xs:element name="Απόσπασμα" type="xs:string" maxOccurs="unbounded"/>
                                <xs:element name="Χαρακτήρας" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:choice maxOccurs="unbounded">
                                                <xs:element name="Όνομα" type="xs:string" maxOccurs="1"/>
                                                <xs:element name="Σκηνικές_Οδηγίες" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                                <xs:element name="Λόγος" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                            </xs:choice>
                                        </xs:sequence>
                                        <xs:attribute name="id" type="xs:integer"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                            <xs:element name="Σκηνή" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:choice maxOccurs="unbounded">
                                            <xs:element name="Απόσπασμα" type="xs:string" maxOccurs="unbounded"/>
                                            <xs:element name="Χαρακτήρας" maxOccurs="unbounded">
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:choice maxOccurs="unbounded">
                                                            <xs:element name="Όνομα" type="xs:string" maxOccurs="1"/>
                                                            <xs:element name="Σκηνικές_Οδηγίες" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                                            <xs:element name="Λόγος" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                                        </xs:choice>
                                                    </xs:sequence>
                                                    <xs:attribute name="id" type="xs:integer"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:choice>
                                    </xs:sequence>
                                    <xs:attribute name="Νούμερο" type="xs:string"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="Νούμερο" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:integer"/>
        </xs:complexType>
        </xs:element>
    </xs:sequence>
    </xs:complexType>
    </xs:element>
</xs:schema>
```

Εικόνα 3.11: Το XML Schema για τον έλεγχο του XML εγγράφου

Όπως βλέπουμε στην εικόνα όλα τα elements είναι μέσα στο root Element Έργα. Κάθε element που έχει child elements ορίζεται ως complexType και αν ακολουθούν πολλά εσωτερικά child elements ορίζεται ως sequence. Σχεδόν όλα τα στοιχεία έχουν δηλωμένα τον τύπο τους εκτός από το Έργα και Έργο. Με το minOccurs με τιμή 0 δηλώνουμε ότι το element είναι προαιρετικό και μπορεί να μην εμφανίζεται (όπως π.χ. οι Σκηνικές Οδηγίες), με το maxOccurs με τιμή 1 ορίζουμε ότι πρέπει να υπάρχει 1 φορά ακριβώς (π.χ. το Όνομα Χαρακτήρα) και ως unbounded σημαίνει ότι το element μπορεί να εμφανιστεί όσες φορές θέλουμε (όπως π.χ. οι Διάλογοι).

3.6 Εισαγωγή στη Βάση Δεδομένων

Για να εισάγουμε και να αποθηκεύσουμε τα δεδομένα του έργου στη βάση μέσω του XML εγγράφου θα χρησιμοποιήσουμε το MySQL Workbench. Πριν ξεκινήσουμε το γέμισμα των πινάκων μέσω της εφαρμογής πρέπει να ορίσουμε τους ίδιους τους Πίνακες, τους τύπους των δεδομένων τους και τις σχέσεις μεταξύ τους.

Αρχικά θα πρέπει να δημιουργήσουμε μια σύνδεση με έναν διακομιστή εάν δεν υπάρχει ήδη. Στο κάτω μέρος της αρχικής σελίδας του MySQLWorkbench φαίνονται οι MySQL συνδέσεις και για να δημιουργήσουμε μια νέα σύνδεση επιλέγουμε το "+" σύμβολο δίπλα στο MySQL Connections. Μπορούμε να δοκιμάσουμε τη σύνδεση με το Test Connection και στη συνέχεια θα μας ζητηθεί να δώσουμε έναν κωδικό για να πραγματοποιηθεί η σύνδεση. Είναι απαραίτητο να γνωρίζουμε το Username και το Password για να συνδεθούμε στη Βάση.

Εφόσον δημιουργήσουμε μια σύνδεση, θα εμφανιστεί το περιβάλλον της εφαρμογής. Στα αριστερά βρίσκονται οι Βάσεις Δεδομένων και οι Πίνακες που περιέχουν. Για να δημιουργήσουμε μια νέα Βάση επιλέγουμε το εικονίδιο Create new schema και αφού δώσουμε όνομα στη Βάση δημιουργείται αυτόματα. Όπως είναι φυσικό δεν διαθέτει πίνακες γι' αυτό θα πρέπει να τους ορίσουμε κατάλληλα με βάση τα δεδομένα του XML εγγράφου που θα τους γεμίσουν. Θα δημιουργήσουμε 4 πίνακες τους Έργο, Διάλογοι, Συγγραφέας και Χαρακτήρες.

3.6.1 Πίνακας Έργο

Ο πίνακας Έργο σύμφωνα με το όνομά του περιέχει πληροφορίες για το έργο. Διαθέτει 5 στήλες οι οποίες είναι:

- Έργο (INT): Είναι ακέραιος αριθμός και αποτελεί το Κύριο Κλειδί του πίνακα που σημαίνει ότι είναι το αναγνωριστικό (id) του έργου. Κανένα άλλο έργο δεν μπορεί να έχει την ίδια τιμή.
- Τίτλος (VARCHAR(45)): Ο τύπος VARCHAR σημαίνει ότι αποτελείται από χαρακτήρες και συγκεκριμένα 45 από αυτούς. Επίσης είναι πεδίο NOT NULL που σημαίνει ότι είναι απαραίτητη η παρουσία του.
- Συγγραφέας (INT): Είναι ακέραιος αριθμός και αντιπροσωπεύει το αναγνωριστικό (id) του Συγγραφέα του έργου. Επίσης είναι Ξένο Κλειδί και συνδέεται με τον πίνακα Συγγραφέα.
- Ημερομηνία Έκδοσης (DATE): Είναι τύπου ημερομηνίας και αφορά την ημερομηνία έκδοσης του έργου.
- Είδος (VARCHAR (45)): Αφορά το είδος του έργου, π.χ. Κωμωδία.

3.6.2 Πίνακας Συγγραφέας

Ο πίνακας Συγγραφέας περιέχει τις πληροφορίες για αυτόν. Διαθέτει τις στήλες:

- Συγγραφέας (INT): Είναι το Κύριο Κλειδί του πίνακα και αποτελεί το αναγνωριστικό του Συγγραφέα. Ο πίνακας Έργο συνδέεται με αυτό.
- Όνομα (VARCHAR(45)): Το όνομα του Συγγραφέα. Επίσης είναι NOT NULL.

- Ημερομηνία Γέννησης (DATE): Η ημερομηνία γέννησης του συγγραφέα, είναι τύπου ημερομηνίας.
- Βιογραφία (VARCHAR(2000)): Βιογραφία για τη ζωή του Συγγραφέα.

3.6.3 Πίνακας Χαρακτήρες

Ο Πίνακας Χαρακτήρες περιέχει τους χαρακτήρες ενός έργου. Στο έργο εμφανίζονται πολλές φορές, αλλά εδώ κάθε χαρακτήρας του έργου αποτελεί μια μοναδική εγγραφή. Περιέχει τις στήλες:

- Έργο (INT): Είναι το αναγνωριστικό του έργου στο οποίο ανήκουν. Αποτελεί Ξένο Κλειδί και συνδέεται με τον πίνακα Έργο.
- Χαρακτήρας (INT): Είναι το αναγνωριστικό του κάθε χαρακτήρα βάσει της σειράς εμφάνισής του στο έργο. Για παράδειγμα ο πρώτος χαρακτήρας του έργου θα πάρει την τιμή 1.
- Όνομα (VARCHAR(50)): Το όνομα του χαρακτήρα.

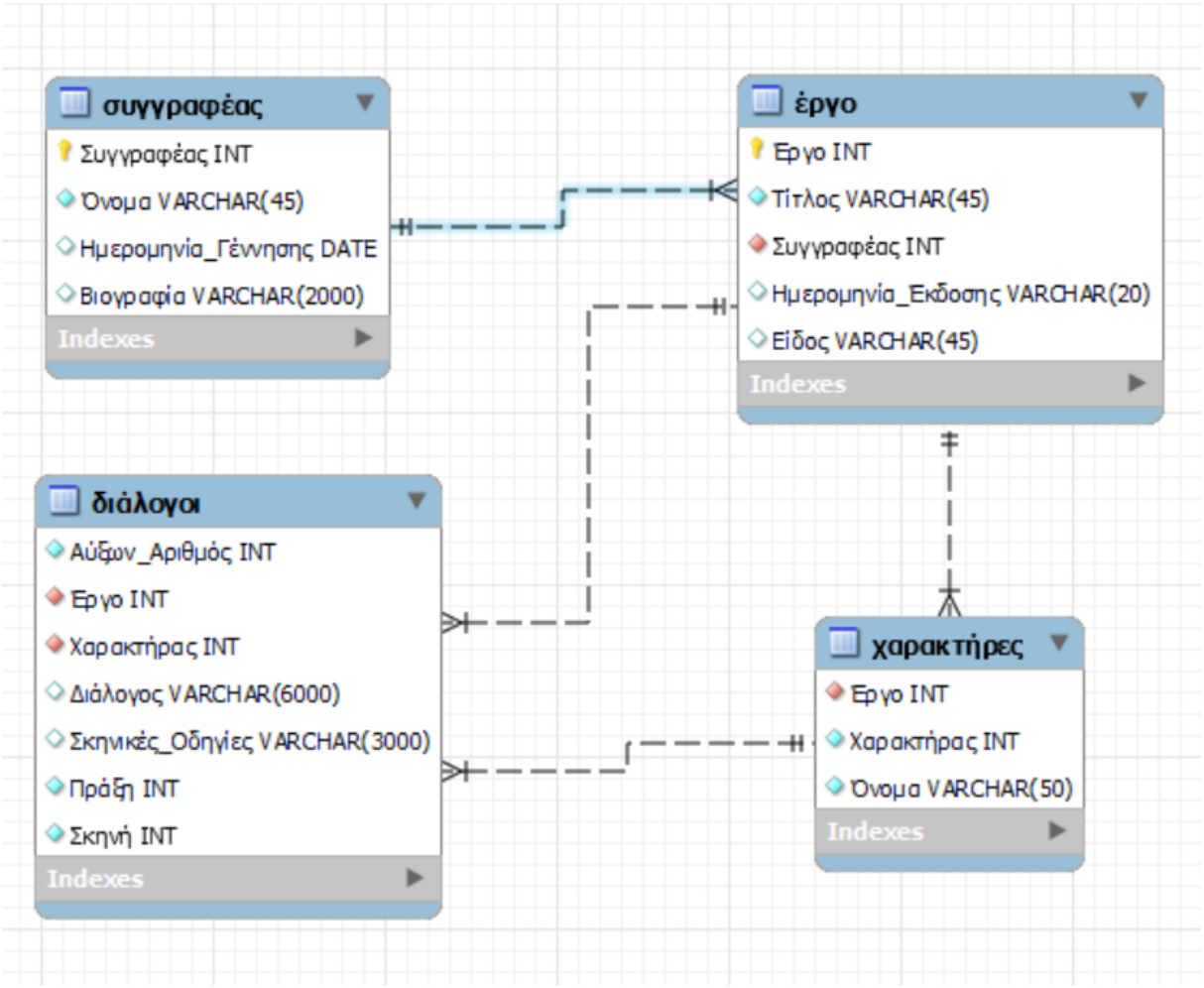
3.6.4 Πίνακας Διάλογοι

Ο πίνακας αυτός περιέχει τους διαλόγους των χαρακτήρων του έργου. Οι στήλες που περιέχονται είναι:

- Αύξων Αριθμός (INT): Είναι ακέραιος αριθμός για την καταμέτρηση των διαλόγων με τη σειρά. Έχει τιμή NOT NULL.
- Έργο (INT): Όπως και στον πίνακα Χαρακτήρες είναι το αναγνωριστικό του έργου και Ξένο Κλειδί που συνδέεται με τον πίνακα Έργο.
- Χαρακτήρας (INT): Το αναγνωριστικό του Χαρακτήρα που μιλάει. Είναι Ξένο Κλειδί και συνδέεται με τον πίνακα Χαρακτήρες.
- Διάλογος (VARCHAR(6000)): Πρόκειται για τα λόγια του κάθε χαρακτήρα.
- Σκηνικές Οδηγίες (VARCHAR(3000)): Αντίστοιχα, αφορά τις σκηνικές οδηγίες του έργου.
- Πράξη (INT): Ο αριθμός της Πράξης στην οποία ανήκει ο διάλογος.
- Σκηνή (INT): Ο αριθμός της Σκηνής στην οποία ανήκει ο διάλογος.

3.6.5 Διάγραμμα ER

Το διάγραμμα ER μαρτυρά τις σχέσεις που έχουν οι πίνακες μεταξύ τους και πως αυτοί σχετίζονται. Είναι πολύ χρήσιμο για την καλύτερη κατανόηση αυτών των σχέσεων. Για να δημιουργήσουμε ένα διάγραμμα, επιλέγουμε Database -> Reverse Engineer. Αμέσως μετά επιλέγουμε τη Βάση που μας ενδιαφέρει και τους πίνακες που θέλουμε να συμπεριλάβουμε στο διάγραμμα. Η έννοια του Reverse Engineer εκφράζει την ανάποδη τυπική διαδικασία, δηλαδή του να δημιουργήσουμε πίνακες από ένα ER διάγραμμα. Στην Εικόνα 3.12 βλέπουμε το σχετικό διάγραμμα με τις σχέσεις των πινάκων που θα τους αποδώσουμε:



Εικόνα 3.12: Το Διάγραμμα ER και οι σχέσεις των πινάκων

Με βάση την εικόνα βλέπουμε ότι η συνδέσεις έχουν κάποια σύμβολα στις άκρες. Ο πίνακας Συγγραφέας συνδέεται με τον πίνακα Έργο με τη σχέση "ένα-προς-πολλά", με άλλα λόγια ένας συγγραφέας μπορεί να γράψει πολλά έργα. Αντίστοιχα βλέπουμε ότι ο πίνακας Έργο συνδέεται με τον πίνακα Χαρακτήρες με την ίδια σχέση γιατί ένα έργο έχει πολλούς χαρακτήρες. Ομοίως για τον πίνακα Διάλογοι, ένα έργο όπως και οι χαρακτήρες του έργου μπορούν να έχουν πολλούς διαλόγους.

3.6.6 Γέμισμα Πινάκων

Έχοντας κατανοήσει τις σχέσεις των πινάκων θα προσπαθήσουμε να τους γεμίσουμε με τα δεδομένα του XML εγγράφου. Αυτή η ενέργεια εκτελείται με το κουμπί Εισαγωγή στη Βάση της εφαρμογής. Για να είναι λειτουργικό το κουμπί, πρέπει να του δώσουμε ένα γεγονός actionPerformed. Μέσα στη μέθοδο του γεγονότος θα γράψουμε τις κατάλληλες εντολές κώδικα.

Αρχικά πρέπει να δημιουργήσουμε σύνδεση με τη Βάση. Στην Εικόνα 3.13 φαίνεται ο κώδικας που το επιτυγχάνει αυτό:

```
String dbUrl = "jdbc:mysql://localhost/theatrical_plays?useUnicode=true&character_set_server=utf8mb4&characterEncoding=UTF-8";
String username = "root";
String password = "root123456";

try {
    //Create connection
    Connection conn = DriverManager.getConnection(dbUrl, username, password);
```

Εικόνα 3.13: Σύνδεση με τη Βάση Δεδομένων

Στην προκειμένη περίπτωση για να συνδεθούμε, χρειαζόμαστε ένα κατάλληλο URL, το Username και το Password. Εφόσον η σύνδεση είναι επιτυχής θα προσθέσουμε επιτέλους τα δεδομένα μας στους πίνακες.

Η εισαγωγή γίνεται μέσω PreparedStatements μέσα στα οποία θα γράψουμε SQL ερωτήματα και θα τα εκτελέσουμε με τη μέθοδο execute(). Για να πάρουμε τα σωστά δεδομένα από το XML έγγραφο και να τα βάλουμε στους πίνακες, πρέπει να πραγματοποιήσουμε κατάλληλη διέλευση του δέντρου του XML εγγράφου και να πάρουμε τα δεδομένα που μας ενδιαφέρουν. Στην Εικόνα 3.14 βλέπουμε το πώς παίρνουμε τα δεδομένα του πίνακα Έργο:

```
//Insert to Play table
PreparedStatement playStmt = conn.prepareStatement("INSERT INTO Έργο "
        + "(Έργο, Τίτλος, Συγγραφέας, Ημερομηνία_Έκδοσης, Είδος) "
        + "VALUES (?, ?, ?, ?, ?)");

//Get XML nodes
nodelist = rootElement.getChildNodes();
//Insert data to NodeList
for (int i = 0; i < nodelist.getLength(); i++) {
    Node node = nodelist.item(i);
    if (node.getNodeName().contains("Έργο")) {
        List<String> columns = Arrays.asList(getAttrValue(node, "id"),
                getTextContent(node, "Τίτλος"),
                getTextContent(node, "Συγγραφέας_ID"),
                getTextContent(node, "Ημερομηνία_Έκδοσης"),
                getTextContent(node, "Είδος"));
        System.out.println("ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΙΝΑΚΑ ΕΡΓΟ: ");
        for (int n = 0; n < columns.size(); n++) {
            System.out.println(n + 1 + " - " + columns.get(n));
            playStmt.setString(n + 1, columns.get(n));
        }
        playStmt.execute();
        System.out.println("\n");
    }
}
```

Εικόνα 3.14: Εισαγωγή στον πίνακα Έργο

Στην εικόνα που βλέπουμε χρησιμοποιούνται οι βοηθητικές μέθοδοι getAttrValue() για τη λήψη attribute των elements και αντίστοιχα η getTextContent() για τη λήψη του περιεχομένου των elements.

Με ανάλογο τρόπο γεμίζουμε και τους υπόλοιπους πίνακες και αποθηκεύουμε τα δεδομένα μας στη Βάση όπου και διατηρούνται οργανωμένα και ασφαλή.

3.7 Ανάλυση Δεδομένων Βάσης

Αυτό είναι το τελικό στάδιο της εφαρμογής, έχοντας ετοιμάσει και οργανώσει τα δεδομένα μας μπορούμε να τα αναλύσουμε. Για να πραγματοποιήσουμε ανάλυση στα δεδομένα θα χρησιμοποιήσουμε τη βιβλιοθήκη TableSaw και θα εισάγουμε το dependency στο pom.xml αρχείο.

Επειδή το Tablesaw λειτουργεί με τα δεδομένα σε μορφή πινάκων χρησιμοποιεί το αντικείμενο Table για το χειρισμό τους. Αρχικά πρέπει να εκτελέσουμε ένα SQL ερώτημα επιλέγοντας τα δεδομένα κάθε πίνακα, να συγκεντρώσουμε τα αποτελέσματα σε 4 διαφορετικά ResultSets και να τα διαβάσουμε με τη μέθοδο read().db() όπου μέσα θα έχουμε τα ResultSet ως παράμετρο. Κατ' αυτό τον τρόπο φτιάχνουμε 4 αντικείμενα τύπου Table με τα δεδομένα της Βάσης.

Το TableSaw διαθέτει μηχανισμό ώστε να μην λαμβάνουμε υπόψιν τις κενές εγγραφές και αυτό γίνεται με τη μέθοδο missingValueIndicator(). Αυτό βοηθάει στον καθαρισμό των δεδομένων μας για την καλύτερη και πιο αξιόπιστη ανάλυσή τους.

Για την επιλογή ανάλυσης έχουν προστεθεί λίστες με διαφορετικές λειτουργίες. Για να πραγματοποιήσουμε Ανάλυση με βάση κάποια από τις κατηγορίες των δεδομένων του έργου επιλέγουμε μια κατηγορία από τη λίστα με το όνομα Ανάλυση. Παράλληλα μπορούμε να επιλέξουμε να εμφανιστούν τα δεδομένα της συγκεκριμένης κατηγορίας με τη μορφή πίνακα, να τα ταξινομίσουμε κατά αύξουσα ή φθίνουσα σειρά και να επιλέξουμε είδος γραφικής παράστασης. Στην Εικόνα 3.15 φαίνεται ένα απόσπασμα του κώδικα για την ανάλυση των Χαρακτήρων:

```
//Characters analysis
case "Χαρακτήρας": {
    Table t = characterFilter.xTabCounts("Χαρακτήρας");
    Table t2 = data;
    //Pie plot
    if (PlotComboBox.getSelectedItem().equals("Πίτα")) {
        Layout layout = Layout.builder().title("ΧΑΡΑΚΤΗΡΕΣ ΕΡΓΟΥ").height(600).width(1200).build();
        PieTrace trace = PieTrace.builder(t.column("Category"), t.numberColumn("Count")).showLegend(true).build();
        Plot.show(new Figure(layout, trace));
        //Bar plot
    } else if (PlotComboBox.getSelectedItem().equals("Ραβδόγραμμα")) {
        Layout layout = Layout.builder().title("ΧΑΡΑΚΤΗΡΕΣ ΕΡΓΟΥ").height(600).width(1200).build();
        BarTrace trace = BarTrace.builder(t.categoricalColumn("Category"), t.numberColumn("Count")).showLegend(true).build();
        Plot.show(new Figure(layout, trace));
        //Box plot
    } else if (PlotComboBox.getSelectedItem().equals("Θηκόγραμμα")) {
        Plot.show(BoxPlot.create("ΧΑΡΑΚΤΗΡΕΣ ΑΝΑ ΠΡΑΞΗ", characterFilter, "Πράξη", "Χαρακτήρας"));
    }
    //Table Sorting
    if (SortingComboBox.getSelectedItem().equals("Αύξουσα")) {
        t = t.sortAscendingOn("Category");
    } else if (SortingComboBox.getSelectedItem().equals("Φθίνουσα")) {
        t = t.sortDescendingOn("Category");
    }
    //Table message
    if (TableCheckBox.isSelected()) {
        String info = "ΧΑΡΑΚΤΗΡΕΣ ΕΡΓΟΥ";
        String message = t.toString();
        JTextArea Area = new JTextArea(message);
        //Set Font to get Table aligned content
        Area.setFont(new Font("monospaced", Font.PLAIN, 12));
        Area.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(Area);
        scrollPane.setPreferredSize(new Dimension(350, 400));
        JOptionPane.showMessageDialog(null, scrollPane, info, JOptionPane.INFORMATION_MESSAGE);
    }
```

Εικόνα 3.15: Ανάλυση κατηγορίας Χαρακτήρων

Άλλες εφαρμογές ανάλυσης είναι ο υπολογισμός των εμφανίσεων ανάμεσα σε 2 κατηγορίες, για παράδειγμα οι εμφανίσεις των χαρακτήρων ανά σκηνή ή οι διάλογοι ανά χαρακτήρα. Αυτές οι εμφανίσεις υπολογίζονται και με τη μορφή ποσοστών (%). Επίσης προσφέρεται ο υπολογισμός περιγραφικών στατιστικών όπως ο μέγιστος και ο ελάχιστος αριθμός εμφανίσεων, ο μέσος όρος εμφανίσεων, κλπ. Τέλος δίνεται η δυνατότητα ανάλυσης βάσει φιλτραρίσματος ενός συνόλου δεδομένων για παράδειγμα, να γίνει ανάλυση δεδομένων μόνο για τους διαλόγους της πρώτης πράξης, τις σκηνικές πράξεις αποκλειστικά της δεύτερης σκηνής, κλπ. Αυτό επιτυγχάνεται με χρήση της μεθόδου where() στην οποία εφαρμόζεται κάποια συνθήκη.

3.8 Επίλογος

Σε αυτό το κεφάλαιο έγινε λεπτομερή περιγραφή της μεθοδολογίας που ακολουθείται και των λειτουργιών που διαθέτει η εφαρμογή, με παραθέματα κώδικα και επεξήγησή τους. Σκοπός του κεφαλαίου είναι η κατανόηση σε βασικό επίπεδο της λειτουργικότητας της εφαρμογής, πράγμα που συμβάλλει σημαντικά στη σωστή χρήση της.

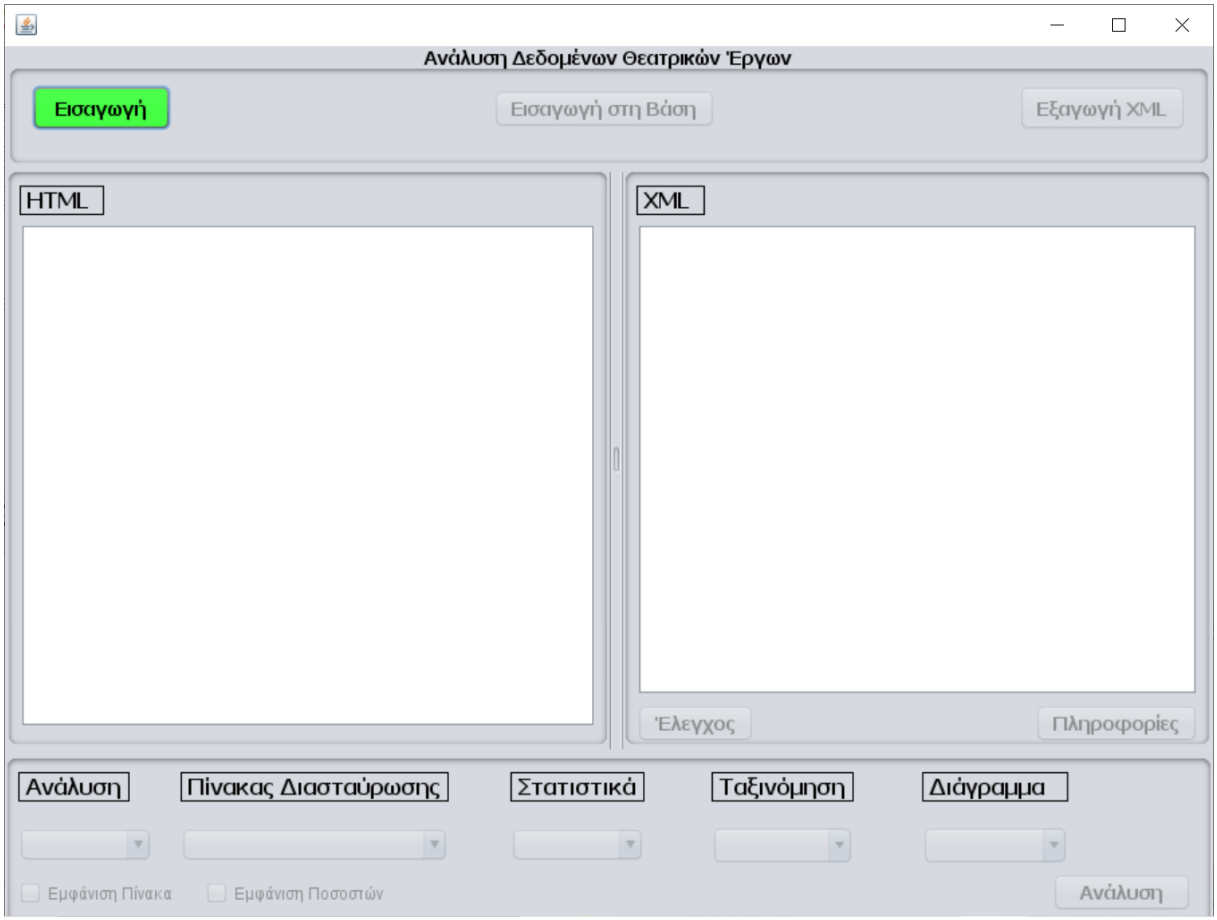
Κεφάλαιο 4ο: Οδηγός Χρήσης

4.1 Εισαγωγή

Το συγκεκριμένο κεφάλαιο αφορά καθαρά τη χρήση της εφαρμογής, παρουσιάζοντας τις δυνατότητες και ιδιαιτερότητές της. Βασικός στόχος αποτελεί η εξαγωγή αποτελεσμάτων και ορθών συμπερασμάτων από την ανάλυση δεδομένων θεατρικών έργων.

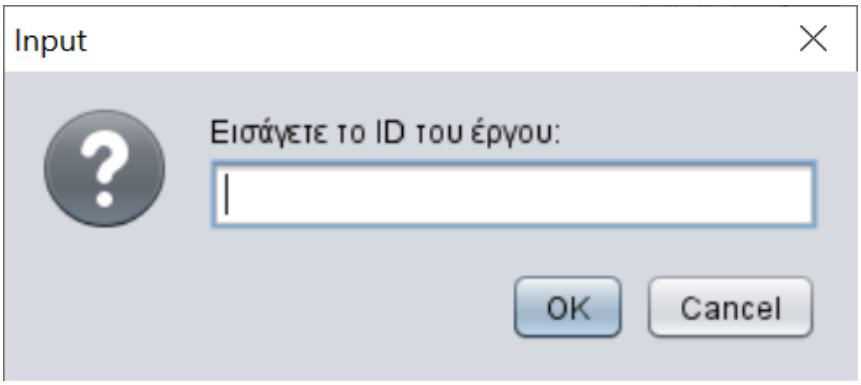
4.2 Είσοδος Έργου

Πριν χρησιμοποιήσουμε την εφαρμογή ας ρίξουμε μια ματιά στο γραφικό της περιβάλλον. Στην Εικόνα 4.1 παρουσιάζεται το περιβάλλον προς χρήση:



Εικόνα 4.1: Το γραφικό περιβάλλον της εφαρμογής

Αρχικά όλα τα κουμπιά είναι απενεργοποιημένα, οπότε η μόνη ενέργεια που μπορούμε να εκτελέσουμε είναι η εισαγωγή ενός θεατρικού έργου. Πατάμε λοιπόν το πράσινο κουμπί Εισαγωγής και επιλέγουμε ένα θέατρο προς ανάλυση από το Παράθυρο Επιλογής. Αμέσως μετά εμφανίζεται το μήνυμα εισαγωγής του αναγνωριστικού ID του έργου. Εισάγουμε έναν ακέραιο αριθμό π.χ. 8, ο οποίος θα αντιστοιχιστεί με το ID του έργου και μπορούμε να το δούμε στο XML έγγραφο δεξιά όταν φορτωθεί. Στην Εικόνα 4.2 φαίνεται το μήνυμα προτροπής:



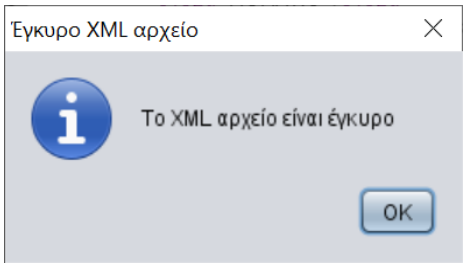
Εικόνα 4.2: Μήνυμα εισαγωγής ID έργου

Αμέσως μετά φορτώνεται το έργο αριστερά και το XML έγγραφο δεξιά εφόσον το πρόγραμμα αναγνώρισε όλα τα μοτίβα HTML στο έργο και τα τράβηξε με επιτυχία. Σε περίπτωση που δεν τραβήξει όλα τα στοιχεία μπορεί ο χρήστης να συμπληρώσει στο XML τις τιμές που χρειάζονται ή να διορθώσει δεδομένα. Στην Εικόνα 4.3 βλέπουμε το έργο μαζί με το XML έγγραφο που δημιουργήθηκε και περιέχει τα δεδομένα του.

Εικόνα 4.3: Εμφάνιση έργου και XML εγγράφου

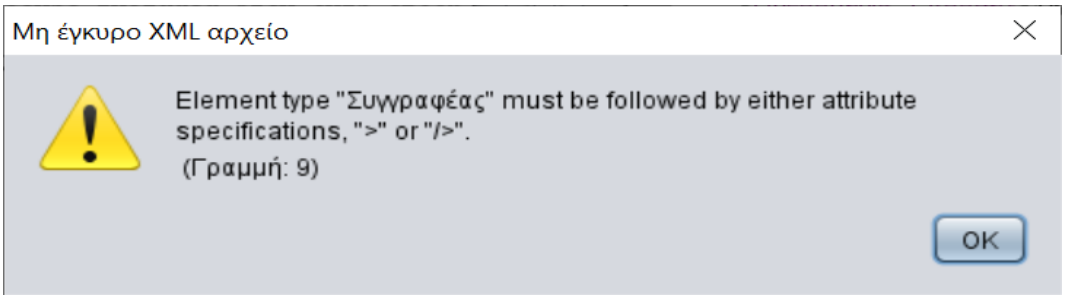
4.3 Έλεγχος Εγκυρότητας Εγγράφου

Αφού εμφανιστούν το κείμενο και το έγγραφο ενεργοποιείται το κουμπί Ελέγχου κάτω από το έγγραφο. Πατώντας το, γίνεται έλεγχος της εγκυρότητας του εγγράφου βάσει του XSD της εφαρμογής. Στην Εικόνα 4.4 φαίνεται το μήνυμα ότι το έγγραφο είναι έγκυρο:



Εικόνα 4.4: Μήνυμα έγκυρου XML εγγράφου

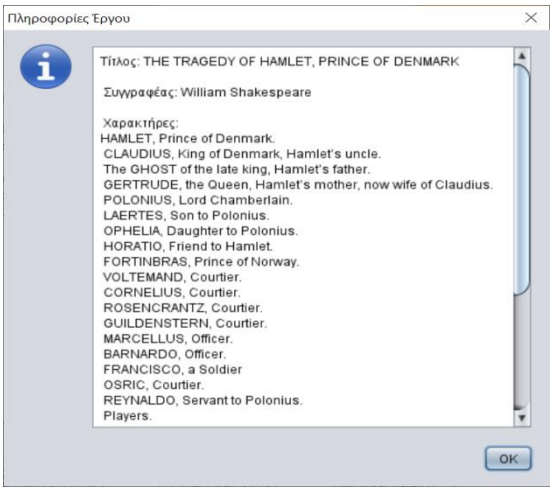
Αντίστοιχα, αν το έγγραφο δεν περιέχει τα δεδομένα όπως έχουν οριστεί στο XSD ή περιέχει κάποιου είδους συντακτικό λάθος π.χ. αν σβήσουμε την αγκύλη από το element Συγγραφέας θα μας βγάλει το μήνυμα της εικόνας 4.5 μαζί με τη γραμμή στην οποία βρίσκεται το πρόβλημα.



Εικόνα 4.5: Μήνυμα μη έγκυρου εγγράφου

4.4 Πληροφορίες Έργου

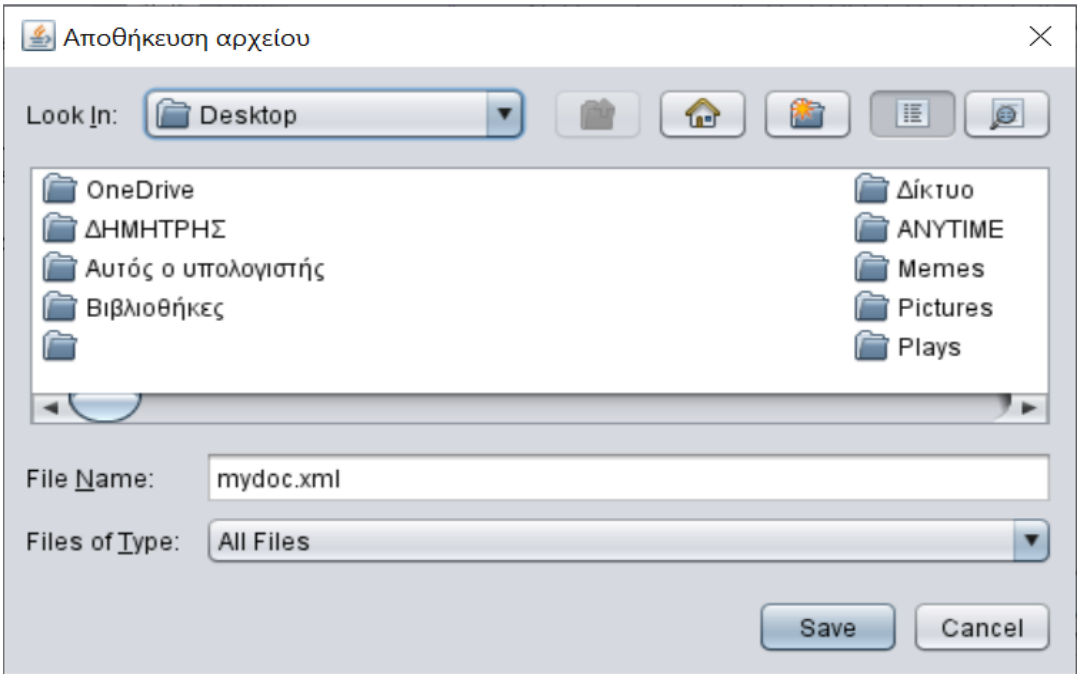
Εφόσον και αν το έγγραφο είναι έγκυρο, ενεργοποιούνται όλες οι υπόλοιπες δυνατότητες της εφαρμογής, για παράδειγμα με το κίτρινο κουμπί Πληροφορίες μπορεί κάποιος να δει βασικές πληροφορίες για το συγκεκριμένο έργο, όπως φαίνεται στην Εικόνα 4.6:



Εικόνα 4.6: Βασικές πληροφορίες έργου

4.5 Αποθήκευση Έγγράφου

Επίσης με το πορτοκαλί κουμπί Εξαγωγή XML πάνω δεξιά μπορεί να αποθηκεύσει κάποιος το XML έγγραφο με τα δεδομένα του έργου καθώς εμφανίζεται ένα παράθυρο αποθήκευσης. Για παράδειγμα στην Εικόνα 4.7 φαίνεται η αποθήκευση του εγγράφου ως mydoc.xml στην Επιφάνεια Εργασίας:



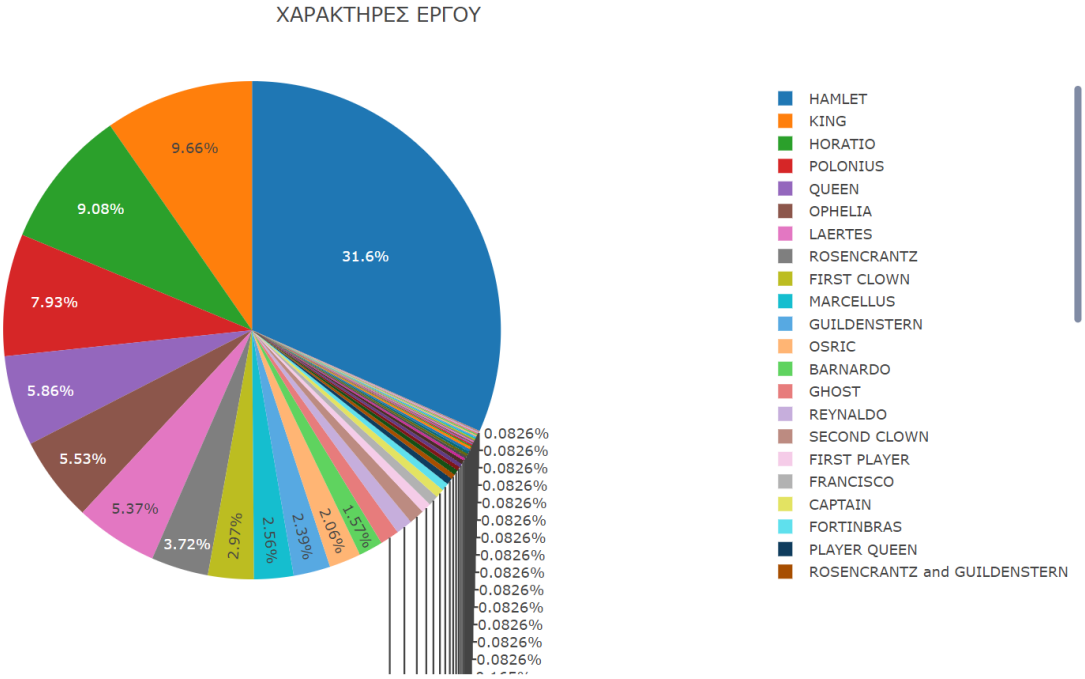
Εικόνα 4.7: Παράθυρο αποθήκευσης εγγράφου

4.6 Είσοδος δεδομένων στη Βάση και Έλεγχος

Πέρα από την απλή αποθήκευση, με το κουμπί Εισαγωγή στη Βάση εισάγονται τα δεδομένα στους Πίνακες της Βάσης, όπως περιγράψαμε στο προηγούμενο κεφάλαιο. Σε περίπτωση που πάει να εισαχθεί έργο με id έργου που υπάρχει ήδη στη Βάση, εμφανίζεται κατάλληλο μήνυμα λάθους. Έτσι ο χρήστης μπορεί να αλλάξει το id γράφοντας μια άλλη τιμή μέσα στο πλαίσιο του XML και μπορεί να ξαναδοκιμάσει την εισαγωγή.

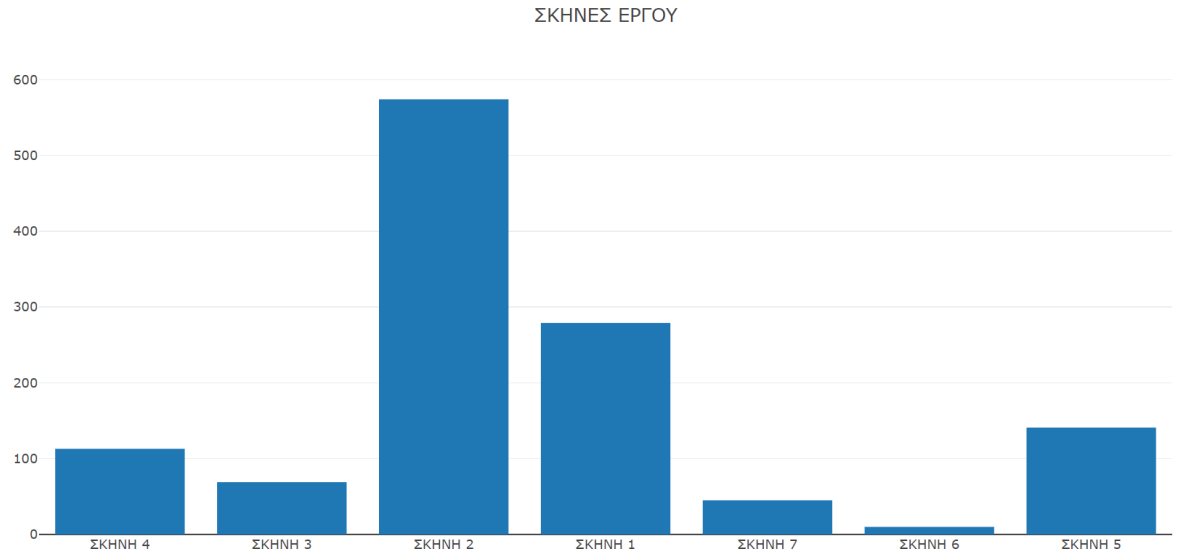
4.7 Ανάλυση

Με την πρώτη λίστα της Ανάλυσης στα αριστερά μπορούμε να αναλύσουμε δεδομένα του έργου με βάση τις κατηγορίες Πράξη, Σκηνή, Χαρακτήρας, Διάλογοι και Σκηνικές Οδηγίες. Επιλέγοντας π.χ. τους Χαρακτήρες και στη λίστα Διαγράμματος την επιλογή Πίτα, το αποτέλεσμα της ανάλυσης φαίνεται στην παρακάτω Εικόνα 4.8:



Εικόνα 4.8: Οι Χαρακτήρες του έργου σε μορφή πίτας

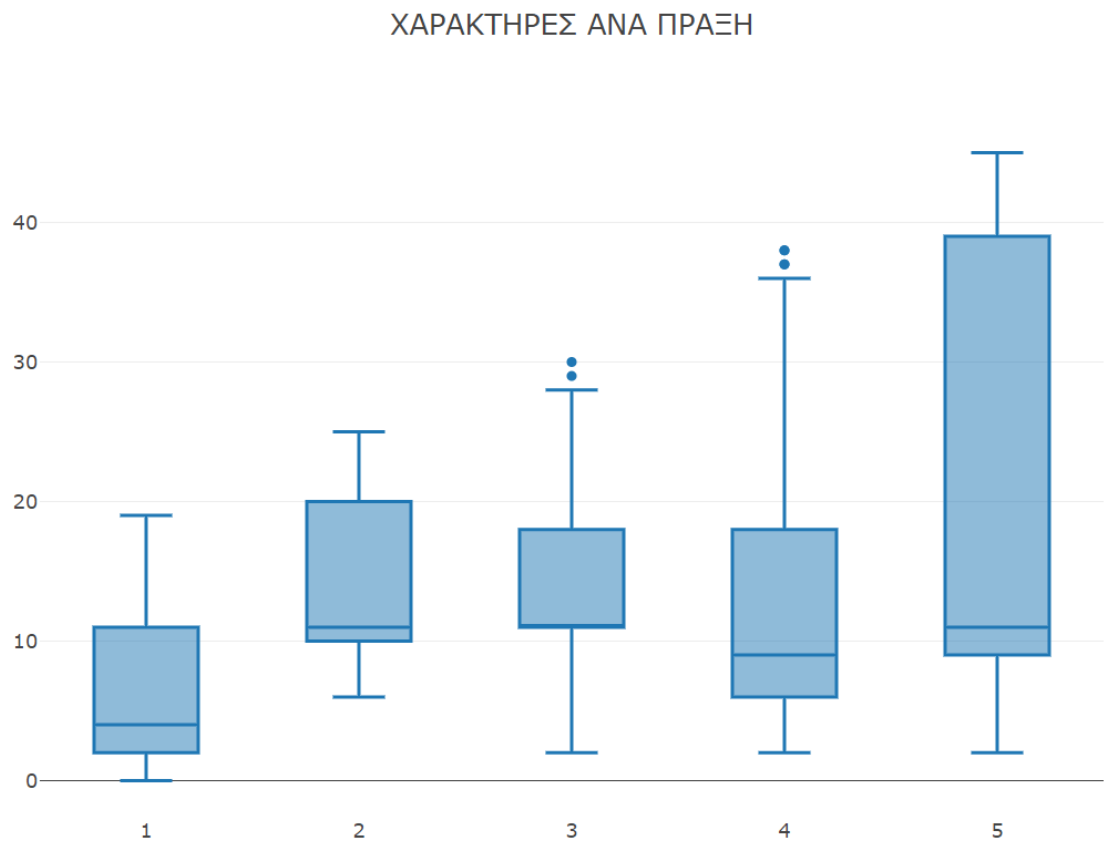
Κρίνοντας από το σχήμα της πίτας, πολύ εύκολα συμπεραίνουμε ποιοι χαρακτήρες έχουν τον πρωταγωνιστικό ρόλο στο έργο και ποιοι έχουν τα λιγότερα λόγια. Αντίστοιχα μπορούμε να επιλέξουμε να αναπαραστήσουμε τα δεδομένα με ράβδους, για παράδειγμα στην περίπτωση των Σκηνών του έργου βλέπουμε την Εικόνα 4.9:



Εικόνα 4.9: Ανάλυση Σκηνών του έργου με ραβδόγραμμα

Από το ραβδόγραμμα αυτό, καταλαβαίνουμε ότι το μεγαλύτερο μέρος του έργου λαμβάνει χώρα κατά την 2η σκηνή του έργου συνολικά. Επομένως είναι πιθανό τα σημαντικότερα γεγονότα του έργου να συμβαίνουν κατά τη 2η σκηνή κάποιας πράξης.

Επιπλέον μπορούμε να συγκρίνουμε δεδομένα μεταξύ τους όπως για παράδειγμα στην Εικόνα 4.10 με αναπαράσταση θηκογράμματος βλέπουμε το μέσο όρο και τη διασπορά των Χαρακτήρων ανά Πράξη του έργου:



Εικόνα 4.10: Μέσος όρος και διασπορά Χαρακτήρων ανά Πράξη

Η εικόνα αυτή μας λέει ότι ο μέσος όρος των Χαρακτήρων ανά Πράξη είναι στα ίδια επίπεδα και αναπαρίσταται με τη μπλε γραμμή μέσα στα κουτιά, αλλά παρατηρούμε ότι στην 5η Πράξη υπάρχει πολύ μεγάλη διασπορά στους χαρακτήρες σε σχέση με τις άλλες Πράξεις. Αυτό δηλώνει μια μορφή ανωμαλίας, που σημαίνει ότι στην 5η Πράξη εμφανίζονται πολύ περισσότεροι χαρακτήρες αναλογικά με τις υπόλοιπες.

Η εφαρμογή προσφέρει και άλλες λειτουργίες με την εμφάνιση πινάκων και συχνοτήτων εμφάνισης των δεδομένων, ωστόσο αυτές οι μέθοδοι ανάλυσης δεν είναι τόσο αποτελεσματικές όσο η οπτική απεικόνιση των δεδομένων για την εξαγωγή συμπερασμάτων γι' αυτό επικεντρωνόμαστε κυρίως στις γραφικές παραστάσεις.

4.8 Επίλογος

Σε αυτό το κεφάλαιο εξερευνήσαμε τις λειτουργίες της εφαρμογής και της δυνατότητές της και αντλήσαμε πληροφορίες μέσα από τις γραφικές παραστάσεις των δεδομένων. Στο τελικό κεφάλαιο θα επιχειρήσουμε να βγάλουμε συμπεράσματα από αυτές τις πληροφορίες και να απαντήσουμε στο πώς μπορούν να χρησιμοποιηθούν και να αξιοποιηθούν αυτές οι γνώσεις.

Κεφάλαιο 5ο: Επίλογος

5.1 Συμπεράσματα

Με βάση τα αποτελέσματα και τις πληροφορίες που αντλήσαμε από τα γραφήματα του προηγούμενου κεφαλαίου μπορούμε να συμπεράνουμε ότι η γνώση παραμέτρων όπως το πόσο μεγάλο σε έκταση είναι ένα θεατρικό έργο, πόσες πράξεις και σκηνές και πόσους χαρακτήρες έχει, μπορεί να είναι πολύ σημαντική για τους ενδιαφερόμενους χρήστες. Αυτοί οι χρήστες, όπως ηθοποιοί, σκηνοθέτες, παραγωγοί, σκηνογράφοι, σπουδαστές θεατρικών έργων, κ.α., έχουν τη δυνατότητα να επιλέξουν και να προγραμματίσουν τις ενέργειες και δράσεις τους ανάλογα με τα οικονομικά και καλλιτεχνικά τους κριτήρια. Για παράδειγμα, οι θεατρικοί επιχειρηματίες-παραγωγοί μπορούν να υπολογίσουν σε σύντομο χρονικό διάστημα το κόστος παραστάσεων (στο οποίο συμπεριλαμβάνονται αμοιβές, σκηνικά, κουστούμια, κλπ.) και να αποφασίσουν εάν επιθυμούν να μεταφέρουν το έργο στη μεγάλη σκηνή, να το απορρίψουν ή να υποδείξουν διορθωτικούς χειρισμούς, με απώτερο σκοπό την αποκόμιση κερδών.

Παρόμοια, εξετάζοντας αυτά τα δεδομένα μπορούν και οι δημιουργοί έργων να προβούν σε παρεμβάσεις ή διορθώσεις στα δικά τους έργα και να πάρουν αποφάσεις όπως την περαιτέρω ανάπτυξη ή αφαίρεση διαλόγων, σκηνών ή χαρακτήρων εάν κρίνουν ότι κάποιο από αυτά χρειάζεται επιπλέον βελτίωση. Αυτό μπορεί να γίνει μόνο κατά τη φάση της συγγραφής του έργου διότι μετά τη δημοσιοποίησή του και την κατοχύρωση των πνευματικών δικαιωμάτων, βάσει της νομοθεσίας απαγορεύεται οποιαδήποτε τροποποίηση ή μεταβολή του.

Έτσι στα πλαίσια της επιχειρησιακής στρατηγικής με τον θεατρικό προγραμματισμό γίνεται σωστότερη κατανομή των διαφόρων παραγωγών σε μια θεατρική σεζόν και όλοι οι εμπλεκόμενοι φορείς (ηθοποιοί, σκηνοθέτες, σκηνογράφοι, ενδυματολόγοι, κ.α.) μπορούν να αποφασίσουν αν θέλουν να συμμετέχουν, ενώ παράλληλα απασχολούνται και σε άλλα αντικείμενα (τηλεόραση, κινηματογράφο, θεατρικές σχολές, κλπ.).

Όλα αυτά συνεπάγονται εξοικονόμιση χρόνου, περισσότερες δυνατότητες απασχόλησης, ιεράρχιση αναγκών από τον κάθε ενδιαφερόμενο και λιγότερο στρες για τον σκληρά πληττόμενο από την ανεργία χώρο του θεάτρου.

5.2 Προτάσεις Βελτίωσης

Στα πλαίσια βελτίωσης της εφαρμογής ανάλυσης θεατρικών έργων παραθέτονται οι παρακάτω πιθανές λειτουργίες. Αρχικά η εφαρμογή θα μπορούσε να χρησιμοποιήσει πιο σύνθετο κώδικα για την ορθή συλλογή ακόμα περισσότερων δεδομένων από διαφορετικά HTML αρχεία έργων και επακόλουθα λιγότερες διορθώσεις από τη μεριά του χρήστη. Μια άλλη ιδέα είναι η χρήση ενός πεδίου τύπου TextField το οποίο να δέχεται URL θεατρικών έργων κατευθείαν από τη βιβλιοθήκη Project Gutenberg. Επίσης θα μπορούσε να είναι χρήσιμη η εμφάνιση ενός πίνακα που να υποδεικνύει την κατάσταση της βάσης δεδομένων, δηλαδή πόσα έργα υπάρχουν στους πίνακες και ποια είναι αυτά και ένα κουμπί ανανέωσης σε περίπτωση που τα δεδομένα της μεταβληθούν. Τέλος η εφαρμογή μπορεί να ενσωματώσει ακόμα περισσότερα γραφήματα της βιβλιοθήκης Tablesaw για πλουσιότερη ανάλυση των δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] Μπώκος, Γ. (2001). Εισαγωγή στην Επιστήμη της Πληροφόρησης, Αθήνα: Παπασωτηρίου

[2] Java Book, Εισαγωγή στη Γλώσσα Προγραμματισμού Java**,** Εργαστήριο Τεχνολογίας Πολυμέσων ΕΜΠ

[3] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2008. Database Systems: The Complete Book (2nd. ed.). Prentice Hall Press, USA.

Papers

[4] Clarisse Bardiot. Measuring Merce Cunningham : a theatre analytics research. DH 2018, Jun 2018, Mexico, Mexico. hal-02337153

Internet Sites

[5] https://en.wikipedia.org/

[6] https://repository.kallipos.gr/

[7] https://zetcode.com/

[8] https://maven.apache.org/

[9] https://mvnrepository.com/

[10] https://jsoup.org/

[11] https://www.w3schools.com

[12] https://www.w3.org

[13] https://www.mysql.com/

[14] https://github.com/jtablesaw/tablesaw

[15] https://jtablesaw.github.io/tablesaw/userguide/toc

[16] https://netbeans.apache.org/

[17] https://www.gutenberg.org/

Βοηθήματα

[18] https://stackoverflow.com/

[19] https://www.regexpal.com/