ΑΝΑΦΟΡΑ ΤΕΛΙΚΟΥ ΠΑΡΑΔΟΤΕΟΥ ΑΝΑΛΥΣΗΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ομάδα Εργασίας:

03116075 Δόλογλου Δημήτριος

03116661 Λώλης Ηλίας

03116610 Αλεξάκης Γεώργιος

1. Εισαγωγή

Ως εργασία τελικού παραδοτέου για το μάθημα της ανάλυσης πληροφοριακών συστημάτων, η ομάδα μας αποφάσισε την υλοποίηση της θεματικής με αριθμό 3. Συγκεκριμένα, η θεματική αυτή αφορά το Θέμα της Ανάπτυξης του Ψηφιακού Μητρώου για τις Δικτυακές Υποδομές της χώρας. Σκοπός της εργασίας μας είναι η δημιουργία παραδειγμάτων υποβολής δεδομένων από παρόχους και παραδειγμάτων κλήσεων για λήψη δεδομένων από άλλες δημόσιες υπηρεσίες, με τη χρήση RESTful web services. Καθώς το JBoss είναι αρκετά παλαιότερο στη χρήση του, προτιμήσαμε να κάνουμε χρήση του αρκετά δημοφιλέστερου framework, ονόματι Spring Boot. Το framework αυτό χρησιμοποιεί Tomcat ως web server για την εφαρμογή μας. Όμως, πέρα από την σύνδεση με μια βάση δεδομένων και την κλήση/υποβολή των δεδομένων αυτής, η εκφώνηση της θεματικής απαιτεί και την υλοποίηση ασύγχρονων μηνυμάτων σφάλματος κατά την υποβολή των δεδομένων αυτών. Για αυτόν τον λόγο, η χρήση του Spring Boot είναι ακόμη πιο σημαντική, καθώς μας προσφέρει ασύγχρονους executors για να υλοποιήσουμε την τεχνική υποβολής και ελέγχου που μας ζητείται. Τέλος, καθώς τα δεδομένα της βάσης αφορούν δίκτυα, τα οποία έχουν τοπολογικά χαρακτηριστικά, μας ζητήθηκε η υλοποίηση της μετατροπής spatial data, δηλαδή shapefiles, σε .json αρχεία (συγκεκριμένα geojson). Αυτό το καταφέραμε με την χρήση της βιβλιοθήκης GDAL. Στο υπόλοιπο της αναφοράς θα βρείτε την αναλυτική περιγραφή της διαδικασίας ανάπτυξης των ζητουμένων αυτών. Το σύστημα τρέχει με την εντολή *gradlew bootRun.*

2. Ανάπτυξη συστήματος δεδομένων

Για την ανάπτυξη του συστήματος δεδομένων μας, αρχικά υλοποιήσαμε μια απλή βάση σε MySQL. Καθώς από την εκφώνηση της εργασίας μας ζητείται η βάση αυτή να χειρίζεται απλούς τύπους δεδομένων, εμείς δημιουργήσαμε 3 Tables για την κάλυψη των απαιτήσεων της διακήρυξης, δηλαδή τα **Network** (Table για την περιγραφή των χαρακτηριστικών των δικτύων μας), **Provider** (Table για την περιγραφή των χαρακτηριστικών των παρόχων) και **Systems** (Table για την περιγραφή των υπηρεσιών που προσφέρονται από το σύστημα). Η σύνδεση με την βάση δεδομένων μας γίνεται από το αρχείο application.properties, στον φάκελο resources. Για την δημιουργία της βάσης μας και των Tables που αναφέρθηκαν, κάναμε χρήση του Hibernate. Συγκεκριμένα, χωρίς καθόλου κώδικα SQL από εμάς και με τη

βοήθεια του Hibernate, στο directory Models υλοποιούμε τα μοντέλα της βάσης μας σύμφωνα με το MVC πρότυπο. Αναλυτικότερα, κάθε μοντέλο (π.χ. Network.java) θα δημιουργήσει με την βοήθεια του Hibernate και του spring boot (@Entity), τα tables της βάσης μας. Όπως παρατηρείται, έχουμε υλοποιήσει και μια βάσικη μορφή validation, ώστε τα δεδομένα προς υποβολή να μην είναι κενά. Στη συνέχεια, υλοποιήσαμε τα Repositories και τους Controllers για την διαχείριση των δεδομένων μας και την δημιουργία των RESTful APIs μας. Σε αυτό το στάδιο υλοποιήθηκαν 2 βασικά API για κάθε μοντέλο δεδομένων. Για παράδειγμα για το μοντέλο των Networks υπάρχει ένα API για την κλήση όλων των δεδομένων της βάσης, για το table Networks και την εμφάνιση τους σε μορφή json (localhost:8080/network/all), καθώς και άλλο ένα για την υποβολή στη βάση δεδομένων (localhost:8080/network/add).

3. Ασύγχρονα μηνύματα

Για να υλοποιήσουμε την διαδικασία εμφάνισης ασύγχρονων μηνυμάτων σφάλματος κατά την υποβολή δεδομένων από τον χρήστη είναι απαραίτητη η μαζική υποβολή τέτοιων δεδομένων. Αυτό δεν γίνεται με τη χρήση του API που αναφέρθηκε προηγουμένως. Συνεπώς, υλοποιήσαμε την ανάγνωση από csv αρχείο. Με καινούριους RESTful Controllers δημιουργήσαμε νέα APIs, τα οποία όταν κληθούν υποβάλουν ένα αρχείο CSV με δεδομένα στη βάση. Αυτό γίνεται με την υλοποίηση συναρτήσεων που υλοποιούνται ασύγχρονα (σύμφωνα με το @Async) και βρίσκονται στο directory Services. Επειδή όμως κάποια δεδομένα είναι κενά και επειδή έχουμε καλέσει validators στα μοντέλα μας, κάποιες από τις υποβολές δεν θα μπορούν να περάσουν. Για αυτό υλοποιήσαμε τον Async Executor. Συγκεκριμένα τα δεδομένα του CSV υποβάλλονται ασύγχρονα σε 5άδες και μετά την υποβολή, επιστρέφονται ασύγχρονα μηνύματα σφάλματος προς τον χρήστη για τα δεδομένα που δεν πέρασαν την υποβολή. Αυτό συμβαίνει χάρις τον Async Exception Handler, ο οποίος θα “πιάσει” το exception από τον validator (για τις κενές υποβολές) και θα εμφανίσει το μήνυμα σφάλματος, όπως έχουμε ορίσει.

4. Shapefiles

Στην εκφώνηση της θεματικής μας ζητήθηκε ακόμη η ανάγκη μετατροπής μορφότυπων shapefiles σε αρχεία json ή XML. Εμείς με την βοήθεια της βιβλιοθήκης GDAL υλοποιήσαμε την μετατροπή αυτή. Συγκεκριμένα, δημουργήσαμε ένα ακόμη RESTful API για το σύστημα μας, το οποίο όταν κληθεί κάνει μια παραδειγματική μετατροπή ενός shapefile σε json (Geojson) αρχείο. Με την βοήθεια του GDAL, ουσιαστικά καλούμε μέσω CMD τις εντολές που μας επιτρέπει και αποθηκεύουμε στο χώρο του BackEnd συστήματος το νέο μας αρχείο, στο όνομα το οποίο θέλουμε.

5. Συμπεράσματα

Η εργασία μας επέτρεψε να ασχοληθούμε με ένα πραγματικό έργο και να δημιουργήσουμε με τη χρήση σημαντικών εργαλείων ένα λειτουργικό back-end που υλοποιεί τις βασικότερες απαιτήσεις του, εξοικειώνοντας μας με έννοιες όπως οι μορφότυποι shapefile, RESTful web services, Asynchronous Execution, το πρότυπο MVC και πολλά ακόμη.