ΤΕΚΜΗΡΙΩΣΗ ΤΕΛΙΚΗΣ ΕΡΓΑΣΙΑΣ ΘΕΩΡΙΑΣ/ΕΡΓΑΣΤΗΡΙΟΥ

Αυγουστής Αριστείδης 101049

Κωνσταντίνος Χρήστου 101040

Λίγα λόγια για τη βάση… Έχουμε…

- Ένα table foitites με AM,LastName,FirstName,Semester δηλαδή personal info φοιτητών
- Ένα table mathimata το οποίο περιέχει κωδικούς μαθημάτων και τα ονόματα τους
- Ένα table vathmoi το οποίο περιέχει ποια μαθήματα έχουν περάσει ποιοι φοιτητές και με τι βαθμό
- Ένα table παρακολουθούν για έξτρα λειτουργίες που δεν γίναν Implement εν τέλει, αυτό το table περιέχει ποιοι φοιτητές currently παρακολουθούν ποια μαθήματα
- Ένα table roles που περιέχει τους ρόλους και τα permission (Role 1 – Admin , Role 2 – Γραμματεία, Role 3 Student)
- Ένα table users που περιέχει τα credentials των user καθώς και τους ρόλους τους (άρα σε τι έχουν access)
- Ένα table leitourgeies που περιέχει πληροφορία για τα permissions των ρόλων.
- Για κάποια webService βρίσκουμε τα επιθυμητά αποτελέσματα κάνοντας INNER JOIN
- Για convenience όλα τα στοιχεία στο DB είναι σε VARCHAR format.

Η multi-tier εφαρμογή μας περιέχει 3 Project.

- Services το οποίο περιέχει όλα τα WebServices και οποιαδήποτε σύνδεση με την Database γίνεται μόνο μέσα σε αυτά
- WebApp που περιέχει σελίδες JSP. Συγκεκριμένα μόνο accessible από φοιτητές και από εδώ καλείται μόνο το Login Web Service και το WS που είναι υπεύθυνο για την προβολή προσωπικών στοιχείων ενός φοιτητή και των βαθμών του.
- DesktopApp – accessible μόνο από γραμματεία και Admin και εδώ γίνονται όλες οι υπόλοιπες λειτουργίες μέσω WS (εκτός εισαγωγή νέου χρήστη για γραμματεία)

Επιλέξαμε JSON format για να μεταφέρουμε τα δεδομένα μας.

Ανάλυση WebApp

Έχουμε μια login.html. Από αυτή δίνουμε String username + password και αυτά μεταφέρονται στην login_check.jsp.

Στη login_check.jsp αποθηκεύονται τα 2 String, μετατρέπονται σε JSON format , φτιάχνουμε ένα αντικείμενο της URL που περιέχει το @Path("LoginService") του Login WS και δίνουμε σαν @QueryParam το JSON αρχείο μας. Στο JSON αρχείο έχουμε βάλει και ένα param με όνομα from το οποίο παίρνει καρφωτά τιμή web. Αυτό συμβαίνει στο Login & ShowInfo WebService για να ξέρουμε από ποιο app γίνεται access. Στο Desktop app το from σε αυτά τα 2 WS παίρνει τιμή desktop.

Κάνουμε openConnection(); και γίνεται ένα http @GET request στο LoginService.

Στο @GET που ερχόμαστε λοιπόν έχουμε κώδικα ο οποίος κάνει access στη Database, κάνουμε Select μέσω sqlquery για να δούμε Username και Role στον πίνακα users με τα credentials που έρχονται από το JSON. Αν υπάρχει και ανάλογα με το role και από πιο app έχουμε κάνει connect βάζουμε σε ένα άλλο JSON αρχείο την κατάλληλη απάντηση (άμα υπάρχει error αντίστοιχο μήνυμα αλλιώς γυρνάμε access με json.toJSONString)

Το login_check στη συνέχεια μέσω ενός BufferedReader διαβάζει το String το μετατρέπει σε JSON Object και εμφανίζουμε κατάλληλα μηνύματα ζητώντας τα περιεχόμενα του JSON.

Σε successful login ανοίγουμε ένα session στο οποίο εκχωρούμε στοιχεία του JSON τα οποία γίνονται retrieve στην show_degrees.jsp και κάνουμε redirect με hyperlink σε αυτή.

Στο show_degrees με την ίδια διαδικασία όπως στην login_check.jsp κάνουμε @GET request στο ShowInfo WebService και αφού πάρουμε πίσω τα στοιχεία από το WS ανοίγουμε άλλο ένα session , βάζουμε μέσα ότι στοιχείο χρειαστούμε (όλα σε JSON) και τα κάνουμε retrieve στο home.jsp στο οποίο τα εμφανίζουμε (στοιχεία φοιτητή)

Ανάλυση DesktopApp

Πάνω κάτω το DesktopApp είναι παρόμοιο. Περιέχει κλάσεις που καλούνε τα webServices και έχουνε μεθόδους που μεταφέρουνε δεδομένα όπου αυτά χρειάζονται (πάντα σε JSON).

Είναι όμως πιο περίπλοκο λόγο δημιουργίας Interface με JFRAME.

Η Εφαρμογή ξεκινάει στη Login.java , η μόνο κλάση που περιέχει main μέθοδο.

Σε αυτή μέσω Constructor που καλείται στην main δημιουργούνται τα κατάλληλα Buttons και JTextFields για να δώσουμε credentials (Login / Password). Αυτή η εφαρμογή είναι μόνο accessible από Admin (Role 1) & Γραμματεία (Role 2).

Στον ActionListener του login button καλούμε την μέθοδο από την κλάση της Login_Check η οποία κάνει σχεδόν την ίδια δουλειά με την login_check.jsp σελίδα του WebApp. Φυσικά υπάρχουνε αλλαγές όπως π.χ ότι δεν εμφανίζουμε σε αυτή στοιχεία και γυρνάμε το json και άλλα π.χ το from δίνεται ως Desktop για να ξέρει το LOGINWS ποιος κάνει access. Εμφανίζονται κατάλληλα μηνύματα για έγκυρο η μη έγκυρο login. Τα errors εδώ όπως και σε αρκετά άλλα σημεία είναι πολυπληθές , εμφανίζεται δηλαδή άμα προσπαθεί να κάνει Login κάποιος φοιτητής μέσω αυτού του app.

Σε περίπτωση που το γίνει success στο Login εξαφανίζουμε όλες τις λειτουργίες για καινούριο Login, εμφανίζουμε ένα Logout button με το οποίο κάνουμε reset για καινούριο Login και εμφανίζουμε επίσης ένα actions Button το οποίο άμα πατηθεί ανοίγουμε τρέχοντας την μέθοδο openActions της κλάσης Menu.java και τις περνάμε σαν παράμετρο τον ρόλο σε JSON format φυσικά.

Αυτή η μέθοδο καλεί τον Constructor της Menu.java που δημιουργεί ένα καινούριο JFrame στο οποίο έχουμε ένα DropDown μενού για το πια λειτουργεία θέλουμε να κάνουμε (Αυτές παίρνουν στοιχεία μέσω JTextField τα οποία δίνονται ως ορίσματα σε μεθόδους κλάσεων που καλούνε τα WebServices)

Τα JSON data που γυρνάμε από αυτά τα WebServices επειδή πρόκειται για INSERT & UPDATE λειτουργίες, είναι Info για το άμα έγινε successful το action η όχι (π.χ υπάρχει ήδη ο φοιτητής/user άρα δεν μπορούμε να έχουμε duplicate entry). Στο DB-end όλα δουλεύουνε αψεγάδιαστα εκτός από όταν προσπαθούμε να κάνουμε INSERT νέο student/user και δώσουμε ελληνικούς χαρακτήρες. Ενώ λύσαμε τέτοια προβλήματα όταν παίρναμε από την database ελληνικούς χαρακτήρες έχουμε πρόβλημα όταν βάζουμε. Δεν έχει βρεθεί λύση για να κάνουμε parse με JSONParser σε UTF-8 μετά από πολύ ψάξιμο. Κανονικά ο Parser σύμφωνα με documentation περνάει σε UTF-8 οπότε τι να πούμε.

Φυσικά στο Frame αυτό ανάλογα με το ποια λειτουργεία επιλέξουμε εξαφανίζονται τα Fields και τα Buttons για τις υπόλοιπες λειτουργίες ενώ εμφανίζονται αυτά που χρειαζόμαστε. Συγκεκριμένα στο user Insert άμα έχουμε κάνει Login ως γραμματεία γυρνάει error mssg. Αρκετές ιδέες τύπου να υπάρχει dropdown μενού με το ποιους φοιτητές έχουμε στη βάση και επιλογή ενός από αυτούς για να δείξουμε τα στοιχεία του ήταν στο πρόγραμμα αλλά δεν τις καταφέραμε εν τέλει.

Other things

Πιστεύουμε ότι ενώ υπάρχει αρκετό error referencing για το τι πρόβλημα εμφανίστηκε στο DesktopApp όταν κάτι δεν γίνεται σωστά, είναι λιγότερο solid από το WebAPP, τουλάχιστον το JFrame με τις λειτουργίες και τα request στα WebServices γιατι το Login JFRAME tight.

Στην τεκμηρίωση αυτή δεν έχουν σημειωθεί πολλά details π.χ στον LoginService δίνουμε ΑΜ π.χ 101049 για να πάρουμε τα στοιχεία του φοιτητή, κάνουμε select από την βάση users που μέσα αυτός είναι γραμμένος ως cs101049 και στο JSON βάζουμε ένα String από το οποίο έχουν κοπεί οι 2 πρώτοι chars (cs) με substring(2) ώστε να έχουμε επιθυμητό αποτέλεσμα και σωστή λειτουργεία. Στη λειτουργεία για εισαγωγή/τροποποίηση του βαθμού ενός φοιτητή σε ένα μάθημα κοιτάμε στη βάση αν υπάρχει βαθμός στο μάθημα και αν υπάρχει κάνουμε Update αλλιώς Insert. Παίζουν πολλά τέτοια γενικά σε κάθε γωνία του κώδικα για να είναι optimized και άμα τεκμηριώναμε όλα αυτά σε κείμενο θα έπιανε πολλές σελίδες οπότε πολλά από αυτά είναι σημειωμένα με comments μέσα στον κώδικα.

Ελπίζουμε πολλές άπο αυτές τις λεπτομέρειες να σας τις εξηγήσουμε όσο το δυνατό καλύτερο στην εξέταση.

Προσπαθήσαμε γενικά σε ότι αναφορά JSON και Web Services να μεταφέρουμε και να χρησιμοποιούμε δεδομένα στην πιο ατόφια μορφή τους πάντα καταλήγοντας σε user friendly μορφή για τον Χρήστη.

Το DesktopApp έχει μέρη που γίναν optimized (π.χ μέθοδοι για την εμφάνιση & εξαφάνιση buttons/πεδίων) - έχουμε πολύ καλό και solid functionality (error documentation etc) αλλά λιτό User Interface και σε μερικά σημεία δυσανάγνωστο κώδικα.