

ΤΕΧΝΟΛΟΓΙΕΣ ΕΦΑΡΜΟΓΩΝ ΔΙΑΔΙΚΤΥΟΥ

PROJECT ΜΑΘΗΜΑΤΟΣ - PRONET

Το project αυτό υλοποιήθηκε χρησιμοποιώντας τις ακόλουθες εκδόσεις: **Tomcat 8.5**, **JDK 1.8.0**, **MySQL 8.0.12**.





Η εφαρμογή αυτή είναι υλοποιημένη σε JAVA (dynamic web project στο Eclipse IDE) και ακολουθεί το **MVC** μοντέλο χρησιμοποιώντας JAVA servlets ως μέρος του *Control*, JSP pages (και plain html αρχεία) ως μέρος του *VIEW* και δικές μας JAVA κλάσεις ως μέρος του *Model*, που υλοποιούν το business logic της εφαρμογής. Κάθε HTTP request περνάει πρώτα από κάποιο απο τα servlet (και πιθανόν κάποιου φίλτρου) του *Control* ανάλογα με το URL του. Το servlet αυτό επεξεργάζεται το αίτημα κατάλληλα και είτε το προωθεί στο κατάλληλο JSP του *View*, το οποίο θα δημιουργήσει δυναμικά την σελίδα που θα σταλεί στο HTTP Response του server καλώντας λειτουργίες του *Model*, είτε ενημερώνει απλά το *Model* αν το αίτημα δεν χρήζει απάντησης (πχ ενημέρωση του backend με AJAX).

Για το front-end κομμάτι αξιοποιήσαμε τις βιβλιοθήκες [Bootstrap 4.1](#) (CSS και JavaScript) και [jQuery 3.3.1](#) (JavaScript).

Για την επικοινωνία με την βάση δεδομένων μας σε JAVA χρησιμοποιήσαμε απευθείας το **JDBC**.

Για την ίδια την βάση δεδομένων μας χρησιμοποιήσαμε την **MySQL** (8.0.12).

Στο παραδοτέο υπάρχουν οι ακόλουθοι φάκελοι/αρχεία:

-  DataBase/: ο φάκελος αυτός περιέχει το schematic *TED.mwb* της βάσης δεδομένων της εφαρμογής για το MySQL Workbench και ένα SQL script που δημιουργεί την βάση δεδομένων μαζί με έναν administrator με στοιχεία: (email: "admin@uoa.gr", password: "admin").
-  TEDProject/: ο φάκελος αυτός περιέχει τον πηγαίο κώδικα της εφαρμογής. Στον υποφάκελο "src/" υπάρχουν οι JAVA κλάσεις του *Control* και του *Model*, ενώ στον υποφάκελο "WebContent/" βρίσκεται τα αρχεία του front-end (html, css, js, file resources) και ο φάκελος *WEB-INF* που περιέχει τα JSP pages του *View*.
-  TomcatConfiguration/: ο φάκελος αυτός περιέχει το αρχείο *server.xml* του Tomcat που πειράξαμε για τους λόγους της εφαρμογής και το αρχείο *.keystore* που χρησιμοποιείται για το SSL/TLS encryption της επικοινωνίας με τον server (HTTPS).
-  TEDProject.war: το WAR αρχείο της εφαρμογής (που επίσης περιλαμβάνει τα source files για λόγους ευκολίας).

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΕΦΑΡΜΟΓΗΣ

Το αρχείο *server.xml* του Tomcat που περιλαμβάνεται στον φάκελο “TomcatConfiguration/” πρέπει να αντικαταστήσει το ομώνυμο αρχείο στον φάκελο “conf/” του Tomcat.

Το αρχείο *.keystore* που επίσης περιλαμβάνεται στον ίδιο φάκελο θα πρέπει να τοποθετηθεί στον φάκελο $\$ \{user.home\}$ (π.χ. “C:/Users/username/”).

Επίσης, πολλές από τις παραμέτρους της εφαρμογής έχουν ομαδοποιηθεί σε ένα αρχείο, το ***config.properties***. Κατά την εκκίνηση του server - για την ακρίβεια όταν φορτώνεται η κλάση *PropertiesManager.java* – εκτελείται ένα κομμάτι κώδικα (static initialization της κλάσης αυτής) που διαβάζει τις παραμέτρους αυτές οι οποίες χρησιμοποιούνται από την εφαρμογή. Τα properties του *config.properties* μεταξύ άλλων περιλαμβάνουν:

- Τα credentials που χρειάζονται για την σύνδεση με την βάση δεδομένων
- Το όνομα του host (πχ: localhost), το port και το πρωτόκολλο (πχ: https) πάνω στα οποία τρέχει ο server.
- Το *saveDir* το οποίο είναι ο root φάκελος κάτω από τον οποίο αποθηκεύονται τα αρχεία που αποθηκεύει η εφαρμογή (θα δημιουργηθεί αν δεν υπάρχει ήδη).
- Το χρονικό όριο λήξης ενός inactive session
- Το default format εμφάνισης ημερομηνιών

ΠΑΡΑΔΟΧΕΣ

- Η εφαρμογή αυτή είναι στα Αγγλικά και είναι σχεδιασμένη να δέχεται και να επεξεργάζεται μόνο Αγγλικά (λατινικούς χαρακτήρες). Δεν υποστηρίζονται ελληνικοί χαρακτήρες (θα φαίνονται «σκουπίδια» αντί για ελληνικά σύμβολα λόγω του *Markdown*) σε inputs που χρησιμοποιούν το *Markdown* (ενώ πχ στα Messages δουλεύουν και τα ελληνικά) και πολλές από τις λειτουργίες όπως ο KNN στα work ads είναι σχεδιασμένες για λατινικούς χαρακτήρες.

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Ακολουθούν ορισμένες παρατηρήσεις για σχεδιαστικές επιλογές για την βάση δεδομένων της εφαρμογής:

- Αντί να αποθηκεύουμε ολοκληρά αρχεία μέσα στην βάση, επιλέξαμε να αποθηκεύουμε τα file paths τους instead. Για την ακρίβεια, αποθηκεύουμε το URI τους στην βάση το οποίο η εφαρμογή μπορεί να χρησιμοποιήσει i) για να παρουσιάσει το αρχείο στο View και ii) για να πάρει το όνομα του αρχείου και να το αναζητήσει στο κατάλληλο φάκελο μέσα στο *saveDir* (ορίζεται στο *config.properties*).
- Κατά το delete εγγραφών από την βάση χρησιμοποιούμε την επιλογή «cascade». Αυτό σημαίνει ότι αν διαγραφεί μια εγγραφή της οποίας το primary key υπάρχει ως τιμή ξένου κλειδιού σε άλλη(ες) εγγραφή(ές) τότε θα διαγραφθεί(ούν) και αυτή(ές). Κάτι τέτοιο είναι επιθυμητό καθώς όλα τα ξένα κλειδιά στην βάση μας είναι “NN” (Not Null) και επομένως εγγραφές με μη υπάρχοντα ξένα κλειδιά δεν έχουν νόημα.

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Παίρνοντας σε σειρά ένα-προς-ένα τα ζητούμενα της εκφώνησης, παρακάτω περιγράφουμε την προσέγγισή μας πάνω σε αυτά:

1. Στη σελίδα καλωσορίσματος (*index.html*) υπάρχει στην μπάρα πλοήγησης φόρμα εισαγωγής credentials για είσοδο στην εφαρμογή ενώ για σελίδα εγγραφής επιλέξαμε να χρησιμοποιήσουμε την ίδια τη σελίδα καλωσορίσματος. Σε ότι αφορά την ασφάλεια των αιτήσεων προς την εφαρμογή μας αρχικά κατασκευάσαμε ένα keystore αρχείο χρησιμοποιώντας το εργαλείο keytool που περιλαμβάνεται στο JDK. Έπειτα ρυθμίσαμε κατάλληλα τον Tomcat από το *server.xml* ώστε να εντοπίζει αυτό το keystore και να κάνει χρήση του πρωτοκόλλου SSL/TLS. Περαιτέρω προσθέσαμε ένα `<security-constraint>` στο *web.xml* της εφαρμογής μας έτσι ώστε ακόμα και οι αιτήσεις http (που γίνονται στο port 8080) να ανακατευθύνονται αυτόματα στο port 8443 (όπου γίνεται χρήση https).
2. Η σελίδα εγγραφής χρήστη στην εφαρμογή είναι η *index.html* στην οποία υπάρχει αντίστοιχη φόρμα. Όλα τα πεδία αυτής της φόρμας είναι υποχρεωτικά εκτός από την εισαγωγή εικόνας προφίλ που, ενώ υπάρχει φυσικά η δυνατότητα να δωθεί, αποφασίσαμε να είμαστε λιγότερο απαιτητικοί και να την κάνουμε προαιρετική. Χρήστες που κάνουν εγγραφή χωρίς να ανεβάσουν δικιά τους εικόνα προφίλ έχουν ως εικόνα προφίλ μια default εικόνα που έχουμε επιλέξει εμείς.
3. Στην εφαρμογή μας κάθε διαχειριστής έχει την ίδια ακριβώς λειτουργικότητα με όλους τους άλλους και έτσι δεν διακρίνονται μεταξύ τους, από την στιγμή που έχουν κάνει login. Κατά την δημιουργία της βάσης με το σχετικό SQL script εισάγουμε έναν admin με email "admin@uoa.gr" και password "admin".
4. Ο διαχειριστής, αφού πατήσει το σχετικό κουμπί στη σελίδα διαχείρισης, μπορεί να επιλέξει οσουσδήποτε ή και όλους (για ευκολία προσφέρεται και επιλογή "Select All") τους Professionals για να εξάγει τα δεδομένα τους σε αρχείο XML το οποίο προσφέρεται για download. Στο αρχείο αυτό για κάθε Professional παρατίθενται με την εξής σειρά τα παρακάτω:
 - Τα προσωπικά του στοιχεία [*ProfID*, ονοματεπώνυμο, επαγγελματική εμπειρία, δεξιότητες, κλπ εκτός του κωδικού του για λόγους ασφαλείας]
 - Τα άρθρα που έχει αναρτήσει [*ArticleID*, περιεχόμενο, ημερομηνία ανάρτησης και paths των αρχείων που περιέχει (αν περιέχει) για το καθένα]
 - Τις αγγελίες που έχει αναρτήσει [*WorkAdID*, τίτλος, περιεχόμενο και ημερομηνία ανάρτησης για το καθένα]
 - Τα σχόλια που έχει αναρτήσει [*CommentID*, *ArticleID* του άρθρου στο οποίο αντιστοιχεί, περιεχόμενο και ημερομηνία σχολιασμού για το καθένα]
 - Τις σημειώσεις ενδιαφέροντος που έχει κάνει [*ArticleID* για κάθε τέτοιο άρθρο]
 - Το δίκτυο των συνδεδεμένων με αυτόν επαγγελματιών [*ProfID* για τον καθένα]

Επιπρόσθετα, περιηγούμενος στο προφίλ κάποιου Professional ο διαχειριστής έχει την επιλογή "Export XML" ώστε να εξάγει τα παραπάνω δεδομένα αποκλειστικά για αυτόν.

Για το ζητούμενο αυτό έγινε χρήση της βιβλιοθήκης **JAXB**.

5. Η σελίδα «Home Page» έχει το ζητούμενο από την εκφώνηση layout:

Στα αριστερά υπάρχουν δύο sticky side tabs: ένα με τις βασικότερες πληροφορίες για τον χρήστη και link στο προφίλ του και ένα που περιέχει έως 6 (έτσι ώστε να χωράνε σε ένα 2x3 grid) από τους Κ «πλησιέστερους» συνδεδεμένους επαγγελματίες (είναι ταξινομημένοι από τον πιο κοντινό στον πιο μακρινό), που βρέθηκαν με τον KNN (για τον KNN δείτε το 12), σε μορφή clickable εικονιδίου, και ένα link προς την σελίδα «Network», όπου υπάρχουν σε grid όλοι.

Στο πάνω δεξιά μέρος υπάρχει μια φόρμα για το ποστάρισμα ενός νέου άρθρου. Το νέο άρθρο αυτό μπορεί να περιέχει κείμενο μορφοποιημένο με τον *Markdown Editor* καθώς και οσοδήποτε αριθμό από πολυμέσα (εικόνες, βίντεο και αρχεία ήχου).

Στο δεξί μέρος, κάτω από την παραπάνω φόρμα, εμφανίζονται τα ζητούμενα άρθρα για τον χρήστη σε βελτιστοποιημένη από τον KNN σειρά (δείτε το 12). Όμως, δεν στέλνονται όλα τα άρθρα από τον server στον client κατευθείαν - αν και στέλνονται τα article IDs τους (ο KNN ταξινομεί article IDs). Αρχικά, ο server θα στείλει μία σελίδα που περιλαμβάνει έναν fixed αριθμό από αυτά και κάθε φορά που ο χρήστης σκρολάρει στο τέλος της σελίδας στέλνεται το επόμενο με AJAX (κάτι που συχνά αναφέρεται ως *infinite scrolling*).

Κάθε άρθρο έχει δομή παρόμοια με αυτή γνωστών μέσων κοινωνικής δικτύωσης. Στο πάνω αριστερά μέρος του εμφανίζεται η φωτογραφία και το ονοματεπώνυμο του χρήστη που το ανάρτησε όπως και το πριν από πόση ώρα αναρτήθηκε. Κάνοντας κλικ στην ώρα αυτή ο χρήστης οδηγείται σε μια ξεχωριστή σελίδα για το άρθρο αυτό (η λειτουργία προσφέρει ένα μοναδικό link για κάθε άρθρο το οποίο διευκολύνει την κοινοποίησή του). Κάτω από αυτά παρατίθεται το κείμενο του άρθρου (αν υπάρχει) κι έπειτα τα πολυμέσα (φωτογραφίες, βίντεο και αρχεία ήχου). Οι εικόνες παρουσιάζονται σε μικρογραφίες, δίνεται ωστόσο η δυνατότητα μεγέθυνσής τους επιλέγοντάς τις. Ακολουθούν τα κουμπιά εκδήλωσης ενδιαφέροντος και σχολιασμού, το δεύτερο εκ των οποίων ανοίγει δυναμικά μια φόρμα εισαγωγής για το σχόλιο. Αναγράφεται επίσης το πλήθος των Professionals που έχουν εκδηλώσει ενδιαφέρον ή έχουν σχολιάσει. Κάνοντας κλικ στον πρώτο αριθμό εμφανίζονται ακριβώς οι Professionals αυτοί σε ένα modal ενώ τα σχόλια παρατίθενται ολόκληρα παρακάτω, με την εικόνα και το ονοματεπώνυμο του Professional που τα έχει κάνει καθώς και την ημερομηνία σχολιασμού. Τέλος, για τα άρθρα και σχόλια που ανήκουν στον logged-in Professional δίνεται η δυνατότητα διαγραφής τους ενώ συγκεκριμένα για τα άρθρα υπάρχει περαιτέρω η δυνατότητα επεξεργασίας του κειμένου τους.

6. Οσον αφορά την καρτέλα «Network» υλοποιήσαμε ακριβώς αυτά που ζητάει η εκφώνηση. Η αναζήτηση επαγγελματία (ανάμεσα σε όλους τους registered επαγγελματίες) γίνεται με μια φόρμα που υποβάλλεται με AJAX. Τα αποτελέσματα έτσι εμφανίζονται δυναμικά με μορφή grid κάτω από το search bar χωρίς να χρειαστεί να ξαναληφθεί ολόκληρη η σελίδα από τον client. Η αναζήτηση γίνεται βάση ονοματεπωνύμου εκμεταλεύοντας το «LIKE» της MySQL. Με κενό input επιστρέφονται όλοι οι επαγγελματίες.

7. Στην καρτέλα «Work Ads» υπάρχουν 4 διακριτές κατηγορίες για τα παρακάτω:

- i) Αγγελίες από συνδεδεμένους Professionals
- ii) Αγγελίες από μη συνδεδεμένους Professionals
- iii) Αγγελίες του logged in Professional
- iv) Applications του logged in Professional

Στην καρτέλα αυτή υπάρχει επίσης το κουμπί για δημιουργία καινούργιας αγγελίας με τη χρήση του *Markdown Editor*.

Πατώντας οποιαδήποτε αγγελία αυτή ανοίγει σε καινούργια σελίδα όπου υπάρχει η δυνατότητα υποβολής αίτησης επίσης με χρήση του *Markdown Editor* (για τις αγγελίες των i) και ii)) ή η δυνατότητα επεξεργασίας ή διαγραφής της αγγελίας (για αυτές του iii)). Στις σελίδες των αγγελιών που ανήκουν στον logged in Professional υπάρχει επίσης το κάτω μέρος τους μια περιοχή για όλα τα Applications που έχουν γίνει στην εκάστοτε αγγελία με τη μορφή *accordion* όπου αυτός μπορεί εύκολα να δει ποιοι και πότε τα έχουν υποβάλει καθώς και το κείμενο που περιέχουν.

Τα Applications του ίδιου του logged in Professional δεν ανοίγουν σε ξεχωριστή σελίδα αλλά παρατίθενται κι αυτά με τη μορφή *accordion* στο iv). Κάνοντας κλικ πάνω σε κάποιο ο χρήστης μπορεί να δει το κείμενο που το συνοδεύει ενώ υπάρχει επίσης κουμπί για την ακύρωσή του.

Στα i) και ii) δεν εμφανίζονται αγγελίες στις οποίες ο logged-in Professional έχει ήδη υποβάλει αίτηση, ενώ η ταξινόμησή τους γίνεται βάσει του αθροίσματος δύο ξεχωριστών score: ένα σκορ κανονικοποιημένο στο $[0,1]$ βάσει του πόσο ταιριάζει η αγγελία στα skills που έχει αναρτήσει ο επαγγελματίας και ένα σκορ επίσης κανονικοποιημένο στο $[0,1]$ που παίρνει από τον αλγόριθμο *K Nearest Neighbors* και αντιπροσωπεύει το πόσο πολύ μοιάζει αυτή η αγγελία με αγγελίες στις οποίες ο επαγγελματίας έχει κάνει ήδη apply. Έτσι, θεωρητικά το βάρος των δύο αυτών bonuses εξισορροπείται.

Για τον υπολογισμό του σκορ που βασίζεται στα skills, χρησιμοποιείται το *SkillRelevanceEvaluator.java*. Εκεί υλοποιείται η συνάρτηση *getStemmedList()* η οποία παίρνει είσοδο ένα String, αφαιρεί πρώτα τα stop words (λέξεις που δεν βοηθούν στην αξιολόγηση των κειμένων λόγω μεγάλης συχνότητάς εμφάνισής τους σε αυτά) και έπειτα, χρησιμοποιώντας το [Snowball](#) το οποίο ενσωματώσαμε στην εφαρμογή μας, πραγματοποιεί **stemming** στις λέξεις αυτού για βελτιστοποίηση της ακρίβειας των συγκρίσεών μας και επιστρέφει μια λίστα με αυτές. Αρχικά δίνουμε στη συνάρτηση αυτή τα skills του logged in Professional και έπειτα, για κάθε αγγελία υπολογίζουμε ένα σκορ με τον εξής τρόπο: Περνάμε τα Strings που περιέχουν τον τίτλο και το περιεχόμενο της αγγελίας (ξεχωριστά) από αυτή τη συνάρτηση και στη συνέχεια αν μια λέξη των skills εμφανίζεται στις λέξεις του τίτλου προσθέτουμε 3 στο τελικό σκορ και αν μια λέξη εμφανίζεται στο περιεχόμενο προσθέτουμε $1+\log(n)$ όπου n το πλήθος των φορών που εμφανίζεται εκεί.

Όσον αφορά το σκορ που υπολογίζεται με **KNN**, αυτό είναι ανάλογο με το πόσο μοιάζει το description της αγγελίας με το description αγγελιών στις οποίες ο επαγγελματίας έχει κάνει ήδη apply. Ο KNN, δηλαδή, για κάθε αγγελία που είναι να ταξινομήσει θα βρει τις K πιο όμοιες (ως προς το description) αγγελίες στις οποίες ο επαγγελματίας έχει κάνει ήδη apply και θα υπολογίσει για την πρώτη ένα σκορ βάσει της τιμής ομοιότητάς της με αυτές τις K αγγελίες. Πειραματικά συνειδητοποιήσαμε ότι η καλύτερη τιμή για το K είναι το μέγεθος των ήδη applied work ads (η μέγιστη τιμή της δηλαδή), καθώς

όσο μεγαλύτερο K τόσο καλύτερη ταξινόμηση παίρνουμε, επειδή εξετάζονται περισσότερα applied ads, και είναι όντως φθηνότερο να υπολογίσουμε και να χρησιμοποιήσουμε όλες τις ομοιότητες από το να τις υπολογίσουμε όλες, να βρούμε τις top-K και να χρησιμοποιήσουμε μόνο αυτές (υποστηρίζεται βέβαια και μικρότερο K στον κώδικα αν αλλάξουμε την παράμετρο K).

Προκειμένου να συγκρίνουμε ομοιότητα ανάμεσα σε κείμενο (το work ad description), κατασκευάζουμε για κάθε διάνυσμα ένα Map (συγκεκριμένα *HashMap<String, Double>*) που αντιστοιχεί λέξεις (terms) του κειμένου στην συχνότητα τους (term frequency) μέσα στο κείμενο, αφού πρώτα αφαιρέσουμε τα *engStopWords.csv* – λέξεις που δεν έχουν καμία σημασία για την ομοιότητα δύο work ads πχ: 'is', 'are', 'my', 'not', etc - (**stemming**) με την χρήση του [Snowball](#). Η δημιουργία του «πραγματικού» διανύσματος (*double[]*) από αυτό το Map καθυστερείται μέχρι να έρθει η ώρα να υπολογιστεί η ομοιότητα μεταξύ δύο Maps (στην συνάρτηση "*documentSimilarity()*"). Εκεί κρατάμε το term frequency όλων των λέξεων που υπάρχουν και στα δύο Maps (καθώς οι υπόλοιπες λέξεις προσφέρουν μηδέν στην συνολική ομοιότητα) για να δημιουργήσουμε τα τελικά διανύσματα και έπειτα, ως similarity, επιστρέφουμε την *cosineSimilarity* των διανυσμάτων αυτών, η οποία χρησιμοποιείται ευρέως για την ομοιότητα κειμένων.

8. Στην σελίδα «Messages» υπάρχει ένα παράθυρο “σε στυλ messenger”, όπου αριστερά εμφανίζονται οι εικόνες profile και τα ονόματα των επαγγελματιών με τους οποίους ο χρήστης έχει ξεκινήσει συνομιλία, ταξινομημένες κατά φθίνουσα σειρά (πρόσφατο - > παλιό) του χρόνου που στάλθηκε το τελευταίο μήνυμα της. Επιλέγοντας την εικόνα προφίλ ο χρήστης μεταφέρεται στην σελίδα με τα Personal Information του χρήστη αυτού, ενώ επιλέγοντας οπουδήποτε αλλού μέσα στο «κουτί» ο χρήστης «ενεργοποιεί» την συνομιλία αυτή.

Κατά την «ενεργοποίηση» μιας συνομιλίας, φορτώνονται τα μηνύματα της συνομιλίας αυτής σε κατάλληλη σειρά (παλιά->πρόσφατα) με AJAX στο πάνω δεξί μέρος του παραθύρου. Ανα πάσα στιγμή, μόνο μία συνομιλία μπορεί να είναι «ενεργή» και να εμφανίζονται τα μηνύματά της στο δεξί μέρος. Έτσι, αν ο χρήστης κλικάρει σε άλλη συνομιλία θα φορτωθούν όλα τα μηνύματα της νέας συνομιλίας με AJAX.

Προκειμένου να στείλει μήνυμα ο χρήστης σε μια υπάρχουσα συνομιλία μπορεί να την «ενεργοποιήσει» και να χρησιμοποιήσει την φόρμα που υπάρχει στο κάτω δεξιά μέρος του παραθύρου. Για να στείλει μήνυμα σε επαγγελματία με τον οποίο δεν έχει ήδη συνομιλία θα πρέπει να πατήσει το σχετικό κουμπί στην σελίδα των Personal Information του επαγγελματία αυτού. Το κουμπί αυτό κάνει redirect στην σελίδα «Messages» με «ενεργή» την νέα συνομιλία του χρήστη με τον επαγγελματία (η οποία θα δημιουργηθεί μόνο αφού σταλεί το νέο μήνυμα).

Να σημειωθεί ότι παρέχεται η δυνατότητα ένας επαγγελματίας να επικοινωνεί σε real time με έναν άλλο χωρίς να χρειάζεται να κάνει refresh την σελίδα. Αυτό γίνεται εφικτό, ελέγχοντας ανά τακτά χρονικά διαστήματα (2 sec) με AJAX αν υπάρχουν νέα μηνύματα και κάνοντάς τα append στην υπάρχουσα συνομιλία στην περίπτωση που υπάρχουν. Κάτι τέτοιο φαίνεται απαραίτητο εφόσον στο client-server HTTP πρωτόκολλο, δεν μπορεί ο server να ξεκινήσει την επικοινωνία όταν «αντιληφθεί» νέα μηνύματα, αλλά πρέπει ο client να «ρωτήσει» πρώτα.

9. Στην καρτέλλα «Notifications» εμφανίζονται οι αιτήσεις σύνδεσης και ειδοποιήσεις για interest και comments σε articles του χρήστη, καθώς επίσης και ειδοποιήσεις για applications που έγιναν σε αγγελίες του χρήστη. Επιπλέον, ο αριθμός των notifications υπολογίζεται κατά το «φόρτωμα» κάθε σελίδας (για την ακρίβεια του navbar) και εμφανίζεται (εφόσον είναι μη μηδενικός) με την μορφή *badge* στο navbar.

10. Στην σελίδα «Personal Information» γίνεται ανασκόπηση του προφίλ του τρέχοντος χρήστη με όλες τις πληροφορίες που αυτός έχει αναρτήσει. Στη σελίδα αυτή υπάρχει κουμπί “Edit Profile” πατώντας το οποίο ο χρήστης οδηγείται σε μια σελίδα όπου μπορεί να επεξεργαστεί όλες του τις πληροφορίες (εκτός από email και κωδικό τα οποία αλλάζουν από τα Settings). Για την εισαγωγή επαγγελματικής εμπειρίας, πληροφοριών εκπαίδευσης και δεξιοτήτων γίνεται χρήση του *Markdown Editor*.

Κάνοντας από οποιαδήποτε σελίδα κλικ σε κάποιον άλλο χρήστη οδηγείται στο προφίλ αυτού, όπου όμως δεν θα μπορεί να δει πληροφορίες που έχουν δηλωθεί ως ιδιωτικές αν δεν είναι συνδεδεμένος μαζί του. Στην σελίδα προφίλ οποιουδήποτε άλλου χρήστη υπάρχει κουμπί για εκκίνηση συνομιλίας με αυτόν ενώ επίσης υπάρχει κουμπί για να στείλουν αίτημα σύνδεσης προς αυτόν, να αποδεχτούν/απορρίψουν εισερχόμενο αίτημα σύνδεσης ή να τον αφαιρέσουν από τις συνδέσεις τους (όποιο από αυτά εφαρμόζεται).

11. Στην καρτέλλα «Settings», πέρα από την επιλογή αλλαγής email και password, προσθέσαμε επιλογή για διαγραφή του account ενός Professional. Για λόγους ασφαλείας οποιαδήποτε αλλαγή αυτών των ρυθμίσεων απαιτεί επιβεβαίωση με την εισαγωγή του κωδικού του χρήστη.

12. [BONUS] Όπως αναφέρθηκε στο 5 τα articles που εμφανίζονται στο «Home Page» ταξινομούνται, όχι απλά σε χρονική σειρά, αλλά βάσει διαφόρων bonuses που τους δίνουμε στο αρχείο *KNNArticles.java*. Το βασικότερο bonus που παίρνουν και που ζητείται από την εκφώνηση λαμβάνεται χρησιμοποιώντας τον αλγόριθμο *K Nearest Neighbors (KNN)*, ενώ προσθέσαμε επιπλέον προαιρετικά bonuses (μπορούν να απενεργοποιηθούν αλλάζοντας τις παραμέτρους στην κλήση της “*reorderArticleIds()*”) για άρθρα i) που είναι πρόσφατα, ii) που είναι δημοφιλή (aka έχουν πολλά Interests) και iii) για τα οποία ο χρήστης δεν έχει δηλώσει ενδιαφέρον (ακόμα).

Όσον αφορά τον KNN, θεωρούμε ένα διανύσμα για τον logged in χρήστη και ένα διάνυσμα για κάθε συνδεδεμένο σε αυτόν χρήστη. Τα διανύσματα αυτά κατασκευάζονται κατά το “*fit()*” του αλγορίθμου και έχουν χαρακτηριστικά, ένα για κάθε άρθρο που προβάλεται στο “Home Page” (μιας και αυτά είναι τα μόνα άρθρα τα οποία βλέπει και μπορεί να αντιδράσει ο logged in χρήστης), μία ακέραια τιμή που αναπαριστά πόσο πολύ ο επαγγελματίας έχει αλληλεπιδράσει με το άρθρο (δηλαδή αν έχει κάνει «interest» και πόσα comments έχει κάνει σε αυτό). Αν ο επαγγελματίας λ.χ. δεν έχει αντιδράσει καθόλου σε ένα από αυτά τα άρθρα η τιμή του αντίστοιχου χαρακτηριστικού θα είναι 0, ενώ αν έχει κάνει interest θα είναι πχ 3. Κατά το reordering των άρθρων βάσει των bonuses τους (“*reorderArticleIds()*”), βρίσκονται τα *K* πλησιέστερα διανύσματα συνδεδεμένων επαγγελματιών στο διάνυσμα του logged in χρήστη, χρησιμοποιώντας ως συνάρτηση εγγύτητας μία δικιά μας συνάρτηση “*similarity()*”. Η συνάρτηση αυτή υπολογίζει την ομοιότητα μεταξύ διανυσμάτων λαμβάνοντας υπόψη την ασυμμετρία που υπάρχει στα χαρακτηριστικά τους (= ότι δηλαδή μας ενδιαφέρουν μόνο

οι μη μηδενικές τιμές). Αφού βρεθούν τα K πλησιέστερα διανύσματα αυτά, γνωρίζοντας σε ποιους επαγγελματίες αντιστοιχούν, δίνουμε ένα bonus, ανάλογο με την τιμή της similarity των K επαγγελματιών αυτών με τον logged in χρήστη, στα άρθρα εκείνα στο «Home Page» τα οποία έχουν είτε ποσταριστεί από αυτούς είτε γίνει “Interested” από αυτούς. Το αποτέλεσμα είναι μία ταξινόμηση άρθρων στην οποία τα πιο σχετικά, με την έννοια ότι εμφανίζονται εξαιτίας επαγγελματιών με τους οποίους μοιάζει περισσότερο ως προς τα άρθρα, για τον logged in επαγγελματία άρθρα εμφανίζονται πρώτα.

Στο αρχείο *KNNArticles.java* θα βρείτε πολύ λεπτομερή σχόλια που περιγράφουν πιο συγκεκριμένα την όλη διαδικασία και εξηγούν και συγκεκριμένες επιλογές. Γενικά, προσπαθήσαμε πειραματικά και διαισθητικά να εξισταθμίσουμε όλα τα διαφορετικά bonuses για να έχουμε ένα καλό αποτέλεσμα.

ΕΠΙΠΛΕΟΝ ΖΗΤΗΜΑΤΑ

➤ Φίλτρα και sessions:

Όταν οποιοσδήποτε χρήστης συνδέεται στην εφαρμογή μας, δημιουργείται ένα session. Στην περίπτωση που συνδέθηκε Professional δίνουμε στο session ένα int attribute με το ProfID του, ενώ για admin ένα boolean attribute “isAdmin” με τιμή true. Ο τρόπος αυτός αναγνώρισης του χρήστη είναι ασφαλής (αντίθετα, π.χ., με τη χρήση cookies για αυτόν το σκοπό) επειδή τα session αποθηκεύονται server-side και άρα δεν υπάρχει κίνδυνος να πειραχτούν από τον client.

Οι σελίδες που είναι προσβάσιμες μόνο από Professionals ή μόνο από Admins έχουν ως πρόθεμα στο URL τους “/prof” ή “/admin” αντίστοιχα. Για να εξασφαλιστεί ότι μόνο οι κατάλληλοι χρήστες θα έχουν πρόσβαση στην εκάστοτε σελίδα γίνεται η χρήση φίλτρων, συγκεκριμένα τα ProfAuthenticationFilter.java και AdminAuthenticationFilter.java αντίστοιχα.

Οι επισκέπτες στην εφαρμογή μας (οι μη έχοντες δηλαδή κάποιο ενεργό session) μπορούν εκτός, προφανώς, από το index.html να δουν το προφίλ κάποιου Professional ή τη σελίδα κάποιου Work Ad εφόσον έχουν το σχετικό link – οι σελίδες αυτές δεν περνούν από κάποιο φίλτρο. Στην περίπτωση αυτή βλέπουν μόνο όσα στοιχεία ο Professional έχει καταχωρήσει ως δημόσια και δεν έχουν την δυνατότητα να υποβάλουν Applications. Επίσης, στην μπάρα πλοήγησης έχουν τη δυνατότητα να κάνουν Sign Up ή να εισάγουν τα στοιχεία τους για να κάνουν Login.

➤ Password hashing:

Θέλοντας να μιμηθούμε μια πραγματική εφαρμογή διαδικτύου, θα ήταν χερίστη πρακτική να αποθηκεύαμε σε plaintext τους κωδικούς των χρηστών μας στη βάση. Για το λόγο αυτό επιλέξαμε να υλοποιήσουμε μια συνηθισμένη μέθοδο ασφάλειας των κωδικών αυτών γνωστή ως hashing.

Όταν ο χρήστης εγγράφεται στην εφαρμογή μας παίρνουμε τον κωδικό που εισήγαγε, του προσθέτουμε στο τέλος ένα δικό μας προκαθορισμένο String (salt) και έπειτα περνάμε το καινούργιο αυτό String που δημιουργήσαμε από έναν hashing αλγόριθμο – χρησιμοποιήσαμε για αυτό τον αλγόριθμο SHA-256. Το αποτέλεσμα (hash) είναι ένας δεξαεκαδικός αριθμός μεγέθους 64 χαρακτήρων τον οποίο αποθηκεύουμε στη βάση και ο οποίος δεν είναι ρεαλιστικά εφικτό να αντιστοιχηθεί με τον πραγματικό κωδικό του χρήστη.

Όταν έπειτα ο χρήστης επιθυμεί να συνδεθεί στην εφαρμογή περνάμε τον κωδικό που εισάγει από την ίδια διαδικασία και συγκρίνουμε το αποτέλεσμα με το hash που έχουμε αποθηκευμένο στη βάση για αυτόν. Ο αλγόριθμος λειτουργεί με τέτοιο τρόπο έτσι ώστε αν εισήγαγε τον ίδιο κωδικό τότε τα hashes αυτά θα είναι ίδια και στην οποία περίπτωση του επιτρέπουμε την πρόσβαση.

➤ HTML και SQL Injection:

Η Βάση Δεδομένων μας είναι ασφαλής από SQL Injection καθώς χρησιμοποιούμε Prepared (SQL) Statements στο JDBC, οι παράμετροι των οποίων (οι οποίες προκύπτουν συνήθως από user input) δεν θεωρούνται μέρος του SQL Statement – εμποδίζοντας έτσι έναν κακόβουλο χρήστη από το να τρέξει κάποια δική του SQL εντολή στην βάση μας. Πουθενά δεν ενώνουμε ως Strings user input και εντολές SQL.

Επιπλέον, έχουμε προνοήσει και η εφαρμογή μας δεν αποδέχεται user input (πχ σε φόρμες) που περιέχει html tags (“<”, “>”), καθώς ένας κακόβουλος χρήστης θα μπορούσε κατέστρεψει το html structure των σελίδων που παράγει η εφαρμογή (πχ εισάγοντας “</html>”).

➤ Timestamps:

Όπως είναι αναμενόμενο, σε πολλά σημεία της εφαρμογής είναι αναγκαία η αποθήκευση και προβολή ημερομηνίας και ώρας (timestamp). Προκειμένου να αποφευχθεί σύγχυση με τα timezones, τα timestamps αποθηκεύονται πάντα στη βάση ως UTC. Όπου όμως κάποιο timestamp χρειάζεται να παρουσιαστεί στον χρήστη αυτό παρουσιάζεται στην δική του ζώνη ώρας, την οποία αποκτούμε από αυτόν μέσω JavaScript. Σημειώνεται επίσης ότι κάνοντας hover πάνω από οποιοδήποτε κείμενο της μορφής “X time ago” εμφανίζεται ένα μικρό tooltip με ολόκληρο το timestamp, μια πρακτική συνηθισμένη σε πολλές ιστοσελίδες.

➤ Markdown Editor:

Στα σημεία της εφαρμογής μας όπου ζητείται η εισαγωγή εκτενούς κειμένου (άρθρα, αγγελίες, αιτήσεις και ορισμένα πεδία του προφίλ) θελήσαμε να δώσουμε στον χρήστη τη δυνατότητα μορφοποίησης του κειμένου του. Για το λόγο αυτό επιλέξαμε να κάνουμε χρήση Markdown ενσωματώνοντας για αυτό τον [SimpleMDE](#).