

**Τεχνολογίες Εφαρμογών Διαδικτύου
Θερινή Εργασία**

Εφαρμογή Επαγγελματικής Δικτύωσης

**ΘΕΟΦΑΝΗΣ ΜΑΡΑΝΤΙΔΗΣ - 1115 2018 00106
ΚΟΥΚΟΥΛΑΡΗΣ ΕΜΜΑΝΟΥΗΛ - 1115 2017 00262**

Πίνακας περιεχομένων

1. Install and deploy
 - 1.1. front-end install
 - 1.2. back-end install
 - 1.2.1. Με docker
 - 1.2.2. Χωρίς docker
2. Front-end
3. Back-end
 - 3.1. Γενικά
 - 3.2. Αλγόριθμος Recommendation System
 - 3.3. Dataset
 - 3.4. Τρόπος επιλογής αποτελεσμάτων για job posts και posts

Εισαγωγή

Η εργασία καλύπτει τα ζητούμενα της εκφώνησης όπως αυτά περιγράφονται.

Συγκεκριμένα η εφαρμογή (τύπου Linkedin) στο νωτιαίο άκρο υλοποιήθηκε βάση της διεπαφής υπηρεσιών REST API με την βοήθεια του Spring Boot.

Στο front-end αντίστοιχα η υλοποίηση βασίστηκε στο Model-View-Controller μοντέλο, με χρήση του AngularJS framework, κατά πλήρη αντιστοιχία με το back-end.

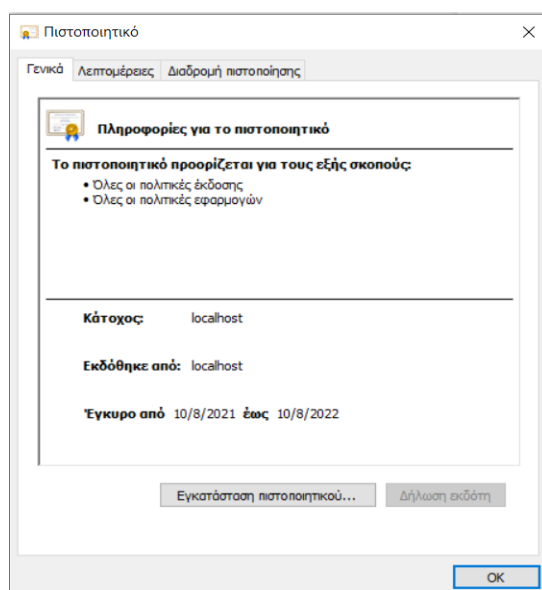
Όσον αφορά την βάση για την αποθήκευση και την διαχείριση των δεδομένων της εφαρμογής επιλέξαμε να χρησιμοποιήσουμε την MySQL.

Οι αιτήσεις γίνονται μέσω του SSL / TLS πρωτοκόλλου (στο classpath βρίσκονται τα αντίστοιχα certificates που χρειάζονται).Επίσης έχουμε προσθέσει και ένα self-signed certificate το οποίο θα χρειαστεί να γίνει χειροκίνητα import ώστε να έχουμε το οπτικό αποτέλεσμα της ασφαλούς σύνδεσης μέσω του https πρωτοκόλλου.

Για την διασφάλιση της επικοινωνίας και της πιστοποίησης του χρήστη σε κάθε αίτηση συμπεριλαμβάνουμε το JSON Web Token το οποίο λαμβάνει ο κάθε χρήστης κατά την σύνδεσή του στην εφαρμογή και το οποίο ανανεώνεται αφότου περάσει το expiration date του.Έτσι είμαστε σίγουροι ότι οι χρήστες έχουν πρόσβαση μόνο εκεί που τους επιτρέπεται να έχουν.Κάθε μη εξουσιοδοτημένος χρήστης δεν μπορεί να έχει πρόσβαση σε όλες τις σελίδες της εφαρμογής παρά μόνο στις σελίδες εγγραφής και σύνδεσης αντίστοιχα.

Για την εγκατάσταση του self-signed certificate κάνουμε διπλό κλικ στο αρχείο **front-end/ssl/server.crt**

Στην συνέχεια πατάμε **εγκατάσταση πιστοποιητικού**



Επιλέγουμε ως **Θέση Αποθήκευσης** τον τρέχον χρήστη και πατάμε επόμενο

← Οδηγός εισαγωγής πιστοποιητικού

Καλώς ορίσατε στον "Οδηγό εισαγωγής πιστοποιητικού"

Αυτός ο οδηγός σας βοηθάει να αντιγράψετε πιστοποιητικά, λίστες αξιοπιστίας πιστοποιητικών και λίστες ανάκλησης πιστοποιητικών από το δίσκο σας στο χώρο αποθήκευσης πιστοποιητικών.

Ένα πιστοποιητικό, το οποίο εκδίδεται από μια αρχή έκδοσης πιστοποιητικών, είναι η επιβεβαίωση της ταυτότητάς σας και περιλαμβάνει πληροφορίες που χρησιμοποιούνται για να προστατέψουν δεδομένα ή για τη δημιουργία ασφαλών συνδέσεων δικτύου. Ένας χώρος αποθήκευσης πιστοποιητικών είναι μια περιοχή συστήματος στην οποία αποθηκεύονται πιστοποιητικά.

Θέση αποθήκευσης

☒ Τρέχων χρήστης

☐ Τοπικός υπολογιστής

Για να συνεχίσετε, κάντε κλικ στο κουμπί "Επόμενο".

Επόμενο Άκυρο

Στον χώρο αποθήκευσης πιστοποιητικών επιλέγουμε **Τοποθέτηση όλων των πιστοποιητικών** στον παρακάτω χώρο αποθήκευσης και στην **Αναζήτηση** κλικάρουμε την επιλογή **Αξιόπιστες κεντρικές αρχές έκδοσης πιστοποιητικών**

← Οδηγός εισαγωγής πιστοποιητικού

Χώρος αποθήκευσης πιστοποιητικών

Οι χώροι αποθήκευσης πιστοποιητικών είναι χώροι του συστήματος στους οποίους αποθηκεύονται τα πιστοποιητικά.

Είναι δυνατό τα Windows να επιλέξουν αυτόματα ένα χώρο αποθήκευσης πιστοποιητικών ή μπορείτε να καθορίσετε εσείς το χώρο αποθήκευσης του πιστοποιητικού.

☐ Αυτόματη επιλογή του χώρου αποθήκευσης ανάλογα με τον τύπο του πιστοποιητικού

☒ Τοποθέτηση όλων των πιστοποιητικών στον παρακάτω χώρο αποθήκευσης

Χώρος αποθήκευσης πιστοποιητικών:

Αναζήτηση...

Επόμενο Άκυρο

Επιλογή χώρου αποθήκευσης πιστοποιητικών

Επιλέξτε ένα χώρο αποθήκευσης πιστοποιητικών που θέλετε να χρησιμοποιήσετε

Προσωπικός χώρος αποθήκευσης

Αξιόπιστες κεντρικές αρχές έκδοσης πιστοποιητικών

Εταιρική αξιοπιστία

Ενδιάμεσες αρχές έκδοσης πιστοποιητικών

Αξιόπιστοι εκδότες

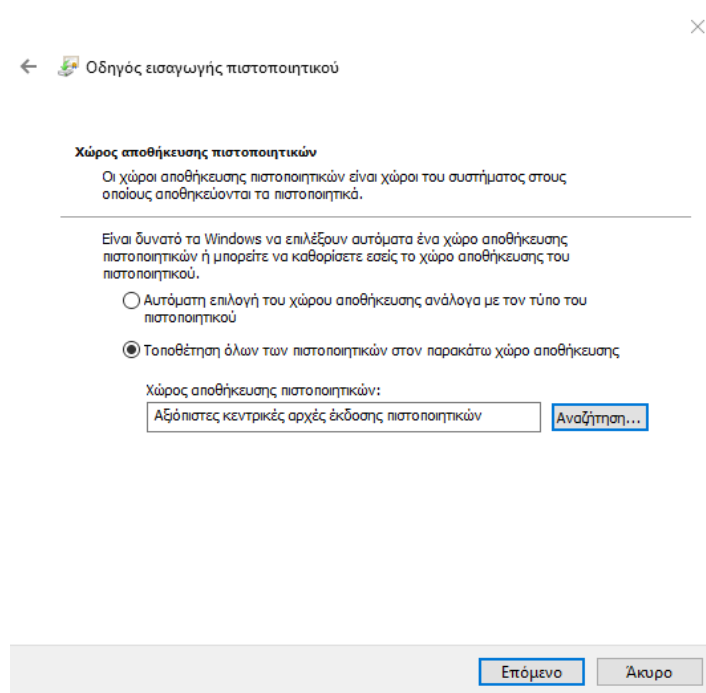
Μη αξιόπιστα πιστοποιητικά

Ανεξέχνη έκδοση πιστοποιητικών λίζας άλλων εταιρειών

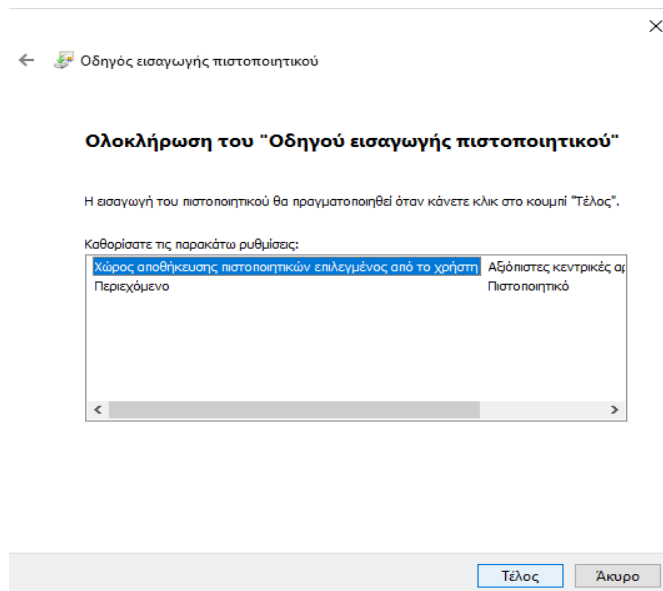
☐ Εμφάνιση φυσικών χώρων αποθήκευσης

OK Άκυρο

Επιλέγουμε λοιπόν το **OK** στο τρέχον παράθυρο , στην συνέχεια το **Επόμενο**



και μετά την επιλογή **Τέλος**.



Έχουμε λοιπόν εγκαταστήσει το certificate στο τοπικό μας μηχάνημα με αποτέλεσμα οι φυλλομετρητές μας να το εμπιστεύονται.

1. Install and deploy

1.1. front-end install

Για την εγκατάσταση των node modules και μεταγλώττιση του angular project, εκτελούμε μία φορά:

```
npm install
```

Για την εκκίνηση του frontend στην port 4200, εκτελούμε:

```
npm run start
```

1.2. back-end install

1.2.1. Με docker (συνιστάται)

Εγκατάσταση μόνο της βάσης σε docker και έναρξη του spring boot τοπικά

```
cd back-end/docker-sql  
docker-compose up -d
```

Με την εκτέλεση των παραπάνω, θα δημιουργηθεί ένα container που θα περιμένει για αιτήματα στην πορτα 3300

Δεν απαιτούνται αλλαγές στο application.properties του spring boot application για την σύνδεση με τη βάση.

Για τον τερματισμό του mysql container, στον κατάλογο *back-end/docker-sql* εκτελούμε:

```
docker-compose down
```

1.2.2. Χωρίς docker

Στην περίπτωση που δεν επιθυμείτε να χρησιμοποιήσετε docker, μπορείτε να τροποποιήσετε τα στοιχεία για την mysql βάση στο αρχείο *application.properties* του spring boot application, και αφού εκκινήσετε την βάση θα πρέπει να κάνετε import το κατάλληλο dataset που βρίσκεται στον κατάλογο *back-end/docker-sql/data*. Εάν η βάση mysql είναι εγκατεστημένη σε σύστημα Win64 (x86_64), τότε χρησιμοποιείτε το αρχείο *linkedit.sql* , αλλιώς εάν η βάση είναι εγκατεστημένη σε Linux (x86_64) , τότε κάντε import το αρχείο *dumpfile.sql*

Αφού ακολουθήσατε έναν από τους δύο τρόπους αρχικοποίησης της βάσης, τότε μπορείτε να εκκινήσετε το spring boot application

Η έναρξη του spring boot application μπορεί να γίνει μέσω intelliJ IDEA πατώντας Build και έπειτα επιλέγοντας το πράσινο start κουμπί στην `main()`

Αφού ακολουθήσετε τα παραπάνω βήματα χωρίς πρόβλημα, πατήστε στο λίνκ:

<https://localhost:4200>

2. Front-end

Η οργάνωση του project έχει γίνει σε components με τρόπο που κρίναμε εμείς πιο εύχρηστο και πιο εύκολο στην οργάνωση.

Για την εναλλαγή και την χρήση πολλαπλών views/pages κάνουμε χρήση του `<router-outlet></router-outlet>` στο app Root Component. Έχουμε την δυνατότητα λοιπόν του mapping ενός component με το εκάστοτε URL Path ή router-link που επιθυμούμε (δεδομένου ότι έχουμε ορίσει τα routes που θα χρησιμοποιήσουμε στο route array το οποίο βρίσκεται στο `app.module.ts`).

Routes

Με την προσθήκη του canActivate Guard στα routes μπορούμε να περιορίσουμε την πρόσβαση σε μη εξουσιοδοτημένους χρήστες στα εκάστοτε path που ορίζουμε το Guard προκειμένου να μην έχουν πλήρη λειτουργικότητα του ιστοτόπου.

Επίσης γίνεται έλεγχος σε περίπτωση που ο χρήστης που συνδέεται είναι ο admin έτσι ώστε να γίνει το navigate στο admin panel. (Έχουμε προεγκατεστημένο τον admin με username

admin και κωδικό admin. Υπάρχει αυστηρά μόνο ένας διαχειριστής της εφαρμογής μας).

Το activate των routes πραγματοποιείται αφού γίνει το authentication του χρήστη.

Μορφοποίηση του site

Για την μορφοποίηση του site εκτος από css καναμε χρηση τοσο του bootstrap όσο και components του material design.

Οι γραμματοσειρές που χρησιμοποιήθηκαν έγιναν import από το <https://fonts.google.com/> ενώ τα εικονίδια (icons) από το <https://fontawesome.com/v4.7/icons/>

Στα forms του sign-up, login page έχουμε τους κατάλληλους validators για τον έλεγχο της μορφής του κάθε πεδίου

Στην home σελίδα, όταν ο χρήστης κάνει scroll down στο τέλος, θα σταλούν περισσότερα post.

Polling

Με την διαδικασία του polling επιτυγχάνουμε την άμεση ανανέωση του feed των μηνυμάτων μεταξύ των χρηστών. Με την χρήση της timer ανανεώνουμε ανά 3 δευτερόλεπτα την επιλεγμένη μας συζήτηση εφόσον υπάρχουν νέα μηνύματα σε αυτή.

Η σελίδα των μηνυμάτων χωρίζεται σε 2 μέρη. Στο αριστερό και στο δεξί μέρος. Στο πρώτο εμφανίζεται η λίστα όλων των συνδεδεμένων χρηστών με τον τρέχον χρήστη. Στο δεξί μέρος εμφανίζονται τα μηνύματα που έχει ανταλλάξει ο συνδεδεμένος χρήστης με τον επιλεγμένο από το αριστερά μέρος “φίλο” του.

Files

Για την επιλογή φωτογραφίας προφίλ, ο χρήστης πρέπει να περιηγηθεί στη σελίδα των ρυθμίσεων όπου υπάρχει επιλογή για το ανέβασμα της. Έχουμε υλοποιήσει τόσο έλεγχο τύπου όσο και έλεγχο μεγέθους του αρχείου που επιλέγεται.

(Επιτρέπονται μόνο .jpg, .png αρχεία με μέγιστο μέγεθος τα 4MB)

Από την μεριά του back-end οι φωτογραφίες αποθηκεύονται στο directory profile_images/{image_filename} και αντίστοιχα στην βάση μας γίνεται η αποθήκευση του ονόματος του αρχείου για τον εκάστοτε χρήστη.

Για να πάρουμε τις φωτογραφίες προφίλ των χρηστών κάνουμε encode σε base64 string την φωτογραφία και την στέλνουμε στο front από όπου γίνεται η εμφάνιση της μέσω του data URI `data:image/jpeg;base64,{base64_img_string}`.

Με αυτόν τον τρόπο επιτυγχάνουμε την εξοικονόμηση χρόνου διότι δεν χρειάζονται HTTP Request και Header Traffic για embedded δεδομένα οπότε καταναλώνουμε λιγότερο

Bandwidth.

Cloud

Για την αποδοτικότερη διαχείριση του load balancing και του χώρου αποθήκευσης αποφασίσαμε να χρησιμοποιήσουμε το google firebase storage για την αποθήκευση όλων των φωτογραφιών / βίντεο / αρχείων ήχου που επιλέγει να ανεβάσει ο χρήστης στο post region.

Επομένως για κάθε post αποθηκεύεται στην βάση και το url του αρχείου που το συνοδεύει εφόσον αυτό υπάρχει (σε διαφορετική περίπτωση είναι null).

Όλες οι σελίδες/καρτέλες υλοποιήθηκαν βάση των οδηγιών της εκφώνησης.

3. Backend

3.1. Γενικά

Με την εκκίνηση του `mysql docker container`, αρχικοποιείται η βάση με ένα έτοιμο dataset το οποίο βρίσκεται στο αρχείο `back-end/docker-sql/data/dumpfile.sql`. Το `mysql container` έχει ανοιχτή την πόρτα 3306 για να περιμένει requests, η οποία γίνεται map με την 3300 του `host machine`. Άρα στο `src/main/resources/application.properties` η default πόρτα για τη `mysql` βάση θα είναι η 3300.

3.2. Αλγόριθμος Recommendation System [Bonus]

Αλγορίθμος εδώ

Εάν το πλήθος των χρηστών που εξετάζουμε είναι $|U|$ και το πλήθος των posts είναι $|Q|$, τότε ξεκινάμε με έναν πίνακα R μεγέθους $|U| \times |Q|$, όπου ο πίνακας U είναι μεγέθους $(|U| \times K)$ και $(K \times |Q|)$ ο πίνακας Q . Ο πίνακας R έχει την παρακάτω μορφή

	Post1	Post2	Post3	Post4	Post5
user1	1	4	5	5	1
user2	1	0	3	0	2
user3	1	5	0	4	0

Με την εκτέλεση του αλγορίθμου, θα πάρουμε μία καλή προσέγγιση για το ποιά θα ήταν η πιο πιθανή αξιολόγηση (views) που θα έδινε ο χρήστης `user3` στα posts όπου δεν έχει παρακολουθήσει (δηλαδή views ίσο με 0). Οι χρήστες που θα χρησιμοποιηθούν για την εκτέλεση του αλγορίθμου θα είναι προφανώς ο δικός μας χρήστης (`user3`) και όλοι οι φίλοι του. Ως στήλες (posts) θα χρησιμοποιήσουμε όλα τα post τα οποία έχουν παρακολουθήσει τουλάχιστον μία φορά όλοι οι χρήστες. Στο τέλος, θα αποφασίσουμε να προτείνουμε τα post

με τις καλύτερες αξιολογήσεις (views) που θα έδινε ο χρήστης user3, όποι η αρχική τιμή ήταν 0.

Ο αλγόριθμος matrix factorization έχει σαν στόχο να παράξει δύο πίνακες P και Q με κατάλληλο K (αριθμός από features), τέτοιοι ώστε

$$R \approx P \times Q^T = \hat{R}$$

Με τον πολλαπλασιασμό των πινάκων θα ισχύει ότι:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^K p_{ik} q_{kj}$$

Αφού πολλαπλασιαστούν οι πίνακες, θα προσπαθήσουμε να μειώσουμε τη διαφορά (σφάλμα της μεθόδου) μεταξύ των πινάκων \hat{R} και R σε κάθε επανάληψη.

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2$$

Ο υπολογισμός του νέου p_{ik} θα προκύπτει από την:

$$p_{ik} = p_{ik} + a \frac{\theta}{\theta p_{ik}} e_{ij}^2 = p_{ik} + 2a e_{ij} q_{kj}$$

διότι

$$\frac{\theta}{\theta p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij}) q_{kj} = -2e_{ij} q_{kj}$$

όμοια για q_{kj}

Σύμφωνα με τα παραπάνω, ο αλγόριθμος θα μπορούσε να γραφεί απλά:

```

matrix_factorization(R, P, Q, K) {
    Q = Ανάστροφος(Q)
    a = παράμετρος ρυθμού εκμάθησης
    iterations = 5000
    Επανάλαβε iterations φορές {
        Για i = 1 έως |P|
            Για j = 1 έως |Q|
                Αν R[i][j] > 0
                    eij = R[i][j] - (σειρά i του P) * (στήλη j του Q)

                    Για k = 1 έως K
                        P[i][k] = P[i][k] + 2 * a * eij * Q[k][j]
                        Q[k][j] = Q[k][j] + 2 * a * eij * P[i][k]

eR = P * Q
e = 0
        Για i = 1 έως |P|
            Για j = 1 έως |Q|
                Αν R[i][j] > 0
                    eij = R[i][j] - (σειρά i του P) * (στήλη j του Q)
                    e = e + R[i][j] - (eij)^2

                    Για k = 1 έως K
                        e = e + (P[i][k])^2 + (Q[k][j])^2

        Αν e < 0.001:
            break
    }
    επέστρεψε P, Q
}

```

Η υλοποίηση του αλγορίθμου `matrix_factorization` βρίσκεται στο αρχείο `back-end/src/main/service/JobPostService.java`

3.3. Dataset [Bonus]

Για τη δημιουργία του dataset θα θεωρήσουμε 4 τομείς (fields): cs, finance, math και biology.

Ο κάθε τομέας έχει 5 κατηγορίες (categories). Το πλήθος των χρηστών ανα τομέα είναι ίσο με το 25% του συνολικού πλήθους των χρηστών.

cs-categories: kernel, web-dev, software-engineering, IT, hardware

fin-categories: finance, logistics, marketing, accountant, broker

math-categories: algebra, number-theory, combinatorics, geometry, mathematical-analysis

bio-categories: biochemistry, biophysics, genetics, biotechnology, anatomy

Ο κάθε χρήστης ανήκει σε ένα μόνο τομέα, και έχει επιλέξει μόνο μία κατηγορία. Ο δικός μας χρήστης (*myuser_201*) ανήκει στον τομέα CS και έχει επιλέξει την κατηγορία web.

Ο κάθε τομέας επίσης έχει και από έναν “γειτονικό” τομέα. Ορίζονται παρακάτω οι γειτονικοί τομείς:

Γειτονικός τομέας του CS είναι το math

Γειτονικός τομέας του math είναι ο CS

Γειτονικός τομέας του fin είναι ο math

Γειτονικός τομέας του biology είναι ο math

Ο κάθε χρήστης κάνει μερικούς φίλους από τον τομέα του και λίγους από τον γειτονικό του τομέα.

Εκτός από χρήστες, δημιουργούνται Job posts, όπου το κάθε ένα ανήκει σε μία κατηγορία.

Ο κάθε χρήστης έχει παρακολουθήσει μόνο Job posts που ανήκουν στην κατηγορία του.

Administrator χρήστης:

username: admin

password: admin

Χρήστης για έλεγχο λειτουργίας του αλγορίθμου matrix-factorization:

username: myuser_201

password: myuser

ανήκει στον τομέα CS και στην κατηγορία web. Για την δοκιμή του αλγορίθμου μπορείτε να δοκιμάσετε και άλλους χρήστες αρκεί όμως να μάθετε την κατηγορία στην οποία ανήκει, μπαίνοντας στο λογαριασμό του. Όλοι οι χρήστες έχουν κωδικό “myuser”

Το γεγονός ότι ο αλγόριθμος βγάζει σωστές προτιμήσεις, αποδεικνύεται όταν για πρώτη φορά πατήσουμε την καρτέλα “Jobs” για τον χρήστη “myuser_201” θα εμφανιστούν και job posts από γειτονικό τομέα, τον math συγκεκριμένα, επειδή έχουν προβληθεί από κάποιους φίλους του (οι οποίοι ανήκουν στον τομέα math, και άρα έχουν παρακολουθήσει job posts που ανήκουν στον τομέα math όπως αναφέραμε).

JobPost

Το πεδίο *requiredSkills* της κλάσης *JobPost* αντιστοιχίζεται με το πεδίο *abilities_desc* της κλάσης *User* και έχει τη μορφή “*ability1,ability2,...,abilityN*”

3.4. Τρόπος επιλογής αποτελεσμάτων για job posts και posts

Για την παραγωγή προτάσεων για τα job posts γίνεται από την συνάρτηση *getSuggestions()* που βρίσκεται στο αρχείο *back-end/src/main/java/com/example.tedi_app/service/JobPostService.java*. Πιο συγκεκριμένα θα επιλεγούν όλα τα job posts τα οποία προτείνει ο αλγόριθμος με βάση τους φίλους και τις προβολές, καθώς και άλλα 5 job post χωρίς την χρήση του αλγορίθμου προτιμήσεων, αλλά κάνοντας χρήση των πεδίων *abilites_desc* από τους χρήστες και *requiredSkills* από τα job posts, βρίσκοντας έτσι αγγελίες που αντιστοιχούν αυστηρά σε κάποιες από τις δεξιότητες του χρήστη.

Για την επιλογή των posts, η συνάρτηση *get_all_post_suggestions()*, στο αρχείο *back-end/src/main/java/com/example.tedi_app/service/PostRecommendationService.java*, θα αρχικοποιήσει μία λίστα με τα 6 πιο πρόσφατα ποστ, θα προσθέσει μερικά posts στα οποία έχουν σημειώσει likes οι φίλοι του χρήστη σε μία λίστα, έπειτα θα προστεθούν post όπου έχουν σχολιάσει οι φίλοι και τέλος θα αξιοποιηθούν οι προβολές ως μέτρο προτίμησης.

Βήματα Υλοποίησης Εφαρμογής

Αρχικά έγινε ο σχεδιασμός της Βάσης δεδομένων από την ομάδα μας, έτσι ώστε να πληρεί τις προϋποθέσεις της εκφώνησης. Στην συνέχεια υλοποιήσαμε τις βασικές λειτουργίες του backend (CRUD) και έγινε ο έλεγχος του API με την βοήθεια του POSTMAN. Έπειτα ξεκινήσαμε την υλοποίηση του front-end και την παράλληλη συμπλήρωση των απαιτούμενων λειτουργιών στο backend.

Μεγαλύτερο βαθμό δυσκολίας βρήκαμε στην υλοποίηση και στην δοκιμή του αλγορίθμου Matrix Factorization Collaborative Filtering.