ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Χρήση της βάσης δεδομένων γραφημάτων neo4j για τη σύσταση επιχειρήσεων βάσει φωτογραφιών
Photo-based Recommendations

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Δήμητρας Σιμορέλη

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθηγητής ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Ιούνιος 2021

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Χρήση της βάσης δεδομένων γραφημάτων neo4j για τη σύσταση επιχειρήσεων βάσει φωτογραφιών
Photo-based Recommendations

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΔΗΜΗΤΡΑΣ ΣΙΜΟΡΕΛΗ

Επιβλέπων : Κωνσταντίνος Διαμαντάρας
Καθηγητής ΔΙ.ΠΑ.Ε.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	'Ονομα Επώνυμο
Choose an item. ΔΙ.ΠΑ.Ε.	Choose an item. ΔΙ.ΠΑ.Ε.	Choose an item. ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Ιούνιος 2021

(Υπογραφή)

.....

Click here to enter text.

Click here to enter text.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κο Κωνσταντίνο Διαμαντάρα και την κα Μαρίνα Δελιανίδη για την επιστημονική καθοδήγηση που μου πρόσφεραν αλλά και για την εμπιστοσύνη, την εμψύχωση και τον χρόνο που απλόχερα διέθεσαν.

Ένα ακόμη μεγάλο ευχαριστώ στον Νικόλα μου για την υπομονή που έδειξε τρία χρόνια τώρα και τη ζωντάνια που με γεμίζει.

Περίληψη

Με την παρούσα διπλωματική εργασία στοχεύουμε στη δημιουργία μιας εφαρμογής προσωποποιημένων συστάσεων για επιχειρήσεις βάσει φωτογραφιών, χρησιμοποιώντας την πιο γνωστή βάση δεδομένων γράφων neo4j και τους αλγορίθμους γράφων που έχουν αναπτυχθεί γι' αυτή. Χρησιμοποιήσαμε το δημόσιο σύνολο δεδομένων του Yelp, μιας δημοφιλούς μηχανής αναζήτησης υπηρεσιών. Βάσει του συνόλου δεδομένων που χρησιμοποιήσαμε έχουμε χρήστες οι οποίοι συνδέονται μεταξύ τους με σχέσεις φιλίας και κάνουν κριτικές και αξιολογήσεις σε επιχειρήσεις. Κάθε επιχείρηση συνδέεται με φωτογραφίες οι οποίες αντιπροσωπεύουν το είδος των υπηρεσιών και προϊόντων που προσφέρουν.

Στην εφαρμογή μας ο χρήστης μπορεί να κάνει μια προσωποποιημένη βελτιωμένη αναζήτηση επιχειρήσεων βάσει κατηγορίας επιχείρησης και ενός κειμένου. Του επιστρέφονται οι πιο σχετικές ως προς αυτόν επιχειρήσεις υπολογίζοντας την ομοιότητα μεταξύ των χρηστών βάσει των κοινών επιχειρήσεων στις οποίες έχουν κάνει αξιολογήσεις. Για κάθε μια από τις επιχειρήσεις εμφανίζονται οι κριτικές που έχουν γίνει από "έμπιστους" χρήστες του τρέχοντα χρήστη.

Του δίνεται επίσης η δυνατότητα σε άλλη επιλογή του μενού της εφαρμογής, να επιλέξει πέντε φωτογραφίες (όποιες θεωρεί εκείνος πιο ελκυστικές) από ένα τυχαίο σύνολο 200 φωτογραφιών που του παρουσιάζουμε σε μορφή πλέγματος και βάσει αυτής του της επιλογής χρησιμοποιώντας κατάλληλους αλγορίθμους γράφων του γίνονται συστάσεις επιχειρήσεων βάσει της ομοιότητας που υπάρχει μεταξύ των φωτογραφιών. Χρησιμοποιούμε τον αλγόριθμο Jaccard Similarity για να εντοπίσουμε ποιες φωτογραφίες είναι όμοιες με αυτές που επέλεξε ο χρήστης. Ακολουθεί ομαδοποίηση των όμοιων φωτογραφιών σε κοινότητες χρησιμοποιώντας τον αλγόριθμο Label Propagation. Καταλήγουμε στη σύσταση εκείνων των επιχειρήσεων των οποίων οι φωτογραφίες συνδέονται με φωτογραφίες της ίδιας κοινότητας, διασχίζοντας τον γράφο μας.

Πρόκειται για ένα σύστημα συστάσεων που βασίζεται στο περιεχόμενο και πιο συγκεκριμένα στις ετικέτες των φωτογραφιών οι οποίες έχουν εντοπιστεί με τη χρήση μηχανικής μάθησης και του εργαλείου Google Vision AI. Με τη χρήση των παραπάνω τεχνολογιών, η προτεινόμενη προσέγγιση συνεισφέρει στην δημιουργία ενός απλού, ευχάριστου και γραφικού συστήματος προσωποποιημένων συστάσεων που βασίζεται στην

απλή επιλογή φωτογραφιών και όχι σε πολύπλοκους μηχανισμούς και στην εισαγωγή επιπλέον ιδιωτικών στοιχείων του χρήστη.

Στην ερώτηση "Πώς οι χρήστες χρησιμοποιούν την εφαρμογή μας;" θα λέγαμε : *Αναζητούν για να βρουν και εξερευνούν για να ανακαλύψουν…*

Λέξεις Κλειδιά: Συστάσεις βάσει φωτογραφιών, βάσεις δεδομένων γράφων, αλγόριθμοι γράφων, ανίχνευση ετικετών, ομοιότητα, Label Propagation

Abstract

In this dissertation we aim to create an application of personalized, photo-based recommendations for businesses, using the most well-known neo4j graph database and graph algorithms developed for it. We used the public dataset of Yelp, a popular service search engine. Based on the data set we used we have users who connect to each other through friendships while they review and rate companies. Each business is associated with photos that represent the type of services and products it offers.

In our application the user can do an improved personalized business search based on business category and some text. The most relevant companies are returned to him by calculating the similarity between the users based on the joint companies for which they have made ratings. For each of the companies the reviews that have been made by "trusted" users of the current user are displayed.

The user is also given the opportunity in another option of the application menu, to select five photos (whichever he finds most attractive) from a random set of 200 photos that we present to him in a grid format and based on this selection using appropriate graph algorithms business recommendations are made based on the similarity between the photos. We use the Jaccard Similarity algorithm to identify which photos are similar to those selected by the user. Grouping of similar photos in communities then follows using the Label Propagation algorithm. We end up recommending those companies the photos of which are linked to photos of the same community, throughout our graph.

It is a system of recommendations based on content and more specifically on photo tags that have been detected using machine learning and the Google Vision AI tool. Using the above technologies, the proposed approach contributes to the creation of a simple, enjoyable and graphical system of personalized recommendations based on the mere selection of photos and not on complex mechanisms and the introduction of user's additional private data.

To the question "How do users use our application?" we would say: "They seek to find and explore to discover…".

Keywords: Photo-based Recommendations, graph databases, graph algorithms, label detection, similarity, label Propagation

Πίνακας περιεχομένων

Πίνακας Εικόνων

1

Εισαγωγή

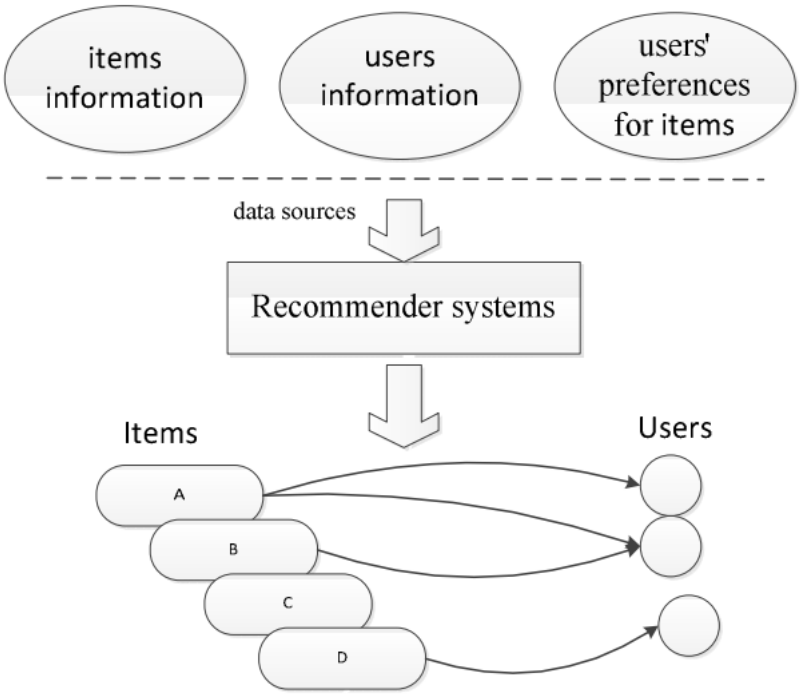
1.1 *Συστήματα εξατομικευμένων συστάσεων*

Η ανάγκη για κατηγοριοποίηση και ταξινόμηση των δεδομένων αναγνωρίστηκε από πολύ νωρίς. Ο όγκος των δεδομένων αυξάνεται καθημερινά, η δυνατότητα επιλογών και συνδυασμών επίσης αυξάνεται και αυτό έχει σαν αποτέλεσμα να ερχόμαστε μπροστά στο ερώτημα - τι να επιλέξουμε. Ο Barry Schwartz, ψυχολόγος, ισχυρίζεται ότι όσο περισσότερες επιλογές έχουμε τόσο χειρότερη και πιο δύσκολη είναι η επιλογή. Αν και η θέση του Schwartz δεν επικυρώθηκε ευρέως, η θεωρία του μπορεί να επιβεβαιωθεί από την προσωπική μας εμπειρία. Ο τεράστιος αριθμός δυνατοτήτων και επιλογών που δίνονται στους ανθρώπους, τους κάνουν πιο νευρικούς και τους φέρνουν στη δύσκολη κατάσταση του να αποφασίσουν τι θα επιλέξουν [1]. Τη λύση έρχονται να δώσουν τα συστήματα εξατομικευμένων συστάσεων (Personalized Recommendations Systems) τα οποία λόγω του μεγάλου όγκου πληροφοριών στο Διαδίκτυο, αποτελούν ένα από τα πιο πολύτιμα εργαλεία στις μέρες μας.

Τα " Συστήματα Συστάσεων " είναι ουσιαστικά αλγόριθμοι οι οποίοι είναι σχεδιασμένοι ώστε να βελτιώνουν την εμπειρία του χρήστη στα πλαίσια μιας πιθανής επιλογής. Ο στόχος τους είναι να δημιουργήσουν σημαντικές προτάσεις σε μια ομάδα χρηστών για αντικείμενα ή προϊόντα που μπορεί να τους ενδιαφέρουν [2]. Αναπτύχθηκαν για να κάνουν τις καθημερινές αποφάσεις των χρηστών απλούστερες. Ο χρήστης είναι στην ευχάριστη θέση να του γίνονται

συστάσεις σχετικές με τα ενδιαφέροντά του. Οι συστάσεις είναι για το τι αγορές να γίνουν, τι ανάγνωση ειδήσεων να γίνει, τι συνδέσεις κοινωνικής δικτύωσης να γίνουν και τι ταινίες να παρακολουθήσει ο χρήστης μεταξύ πολλών άλλων με πρωταρχικό στόχο τη διευκόλυνση του στη διαδικασία της αναζήτησης. Τα συστήματα συστάσεων είναι δημοφιλή κυρίως σε τομείς που έχουν να κάνουν με ταινίες, μουσική, βιβλία, προϊόντα (γενικά), άτομα (κοινωνικά δίκτυα) και υπηρεσίες (εστιατόρια, καταλύματα). Ανάμεσα στις πιο δημοφιλείς ιστοσελίδες που χρησιμοποιούν εξατομικευμένα συστήματα συστάσεων είναι η Amazon.com, η οποία παρέχει μια εξατομικευμένη ιστοσελίδα για κάθε μεμονωμένο χρήστη. Το Netflix είναι ένα άλλο παράδειγμα ιστοσελίδας που χρησιμοποιεί εξατομικευμένα συστήματα συστάσεων για να προταθούν ταινίες και τηλεοπτικές εκπομπές.

Το παρακάτω σχήμα δείχνει το διάγραμμα λειτουργίας ενός συστήματος συστάσεων :



Εικόνα 1.1 : Διάγραμμα συστήματος συστάσεων (Πηγή [3])

Θα πρέπει να σημειωθεί ότι πρόκειται για μια πολύ απαιτητική διαδικασία: η σχεδίαση και ανάπτυξη ενός τέτοιου συστήματος απαιτεί συνδυασμό γνώσεων και δεξιοτήτων από διαφορετικούς τομείς της επιστήμης των υπολογιστών [4]. Εξαρτάται από τον τομέα και από τα ιδιαίτερα χαρακτηριστικά των δεδομένων που έχουμε στη διάθεσή μας. Υπάρχουν πολλοί αλγόριθμοι που έχουν αναπτυχθεί και μπορούν να χρησιμοποιηθούν για τη δημιουργία συστημάτων προσωποποιημένων συστάσεων.

Οι δύο πιο σημαντικοί και ευρέως χρησιμοποιούμενοι αλγόριθμοι είναι οι εξής [2] :

Φιλτράρισμα βάσει περιεχομένου (Content-based filtering)

Ο συγκεκριμένος αλγόριθμος δε χρησιμοποιεί στοιχεία που αφορούν άλλους χρήστες πέραν του τρέχοντος χρήστη για τον οποίο παράγονται οι συστάσεις. Εντοπίζονται και συστήνονται στοιχεία τα οποία έχουν ιδιότητες παρόμοιες με αυτές που ένας χρήστης προτίμησε στο παρελθόν. Βασίζεται στη σύγκριση χαρακτηριστικών μεταξύ στοιχείων που ενδιαφέρουν τον χρήστη. Η εξατομίκευση βάσει περιεχομένου είναι αποτελεσματική σε συστήματα όπου υπάρχει προφανής κατηγοριοποίηση στοιχείων, όπως για παράδειγμα τα καταστήματα βιβλίων. Η μέτρηση και ο υπολογισμός της ομοιότητας μεταξύ στοιχείων μπορεί να είναι ιδιαίτερα χρήσιμα γι' αυτή την προσέγγιση.

Συνεργατικό φιλτράρισμα (Collaborative Filtering)

Η βασική ιδέα είναι να βρούμε ποιοι χρήστες μοιράζονται τα ίδια ενδιαφέροντα με τον ενδιαφερόμενο χρήστη στο παρελθόν. Η κατηγορία αυτή αλγορίθμων εξατομίκευσης στηρίζεται στο γεγονός ότι οι χρήστες οι οποίοι έχουν παρόμοιες προτιμήσεις, βαθμολογούν και αξιολογούν με παρόμοιο τρόπο. Οι τεχνικές αυτές παράγουν προβλέψεις σχετικά με το τι χρειάζεται ένας χρήστης, βασιζόμενες στους πιο κοντινούς (ως προς τις προτιμήσεις) σε αυτόν χρήστες [5]. Οι αλγόριθμοι ομοιότητας Pearson και Cosine χρησιμοποιούνται ευρέως σε αυτούς τους αλγορίθμους.

Η αρχιτεκτονική των συστημάτων συστάσεων και η αξιολόγηση τους σε προβλήματα στον πραγματικό κόσμο αποτελούν έναν ενεργό τομέα έρευνας. Αν και τα συστήματα συστάσεων και εξατομίκευσης αποτελούν σχετικά ένα νέο πεδίο μελέτης στη βιβλιογραφία, οι τεχνικές τους έχουν υιοθετηθεί ευρέως και έχουν λύσει ως ένα βαθμό το πρόβλημα της υπερφόρτωσης πληροφοριών [6]. Να σημειωθεί ότι η χρήση των συστημάτων συστάσεων είναι απαραίτητη όχι μόνο για τους χρήστες αλλά και για τους παρόχους υπηρεσιών [7]. Και αυτό γιατί οι ηλεκτρονικές επιχειρήσεις θα αυξήσουν τα κέρδη τους προωθώντας προϊόντα από μια ευρύτερη γκάμα προϊόντων και προτείνοντας στον χρήστη προϊόντα που τον ενδιαφέρουν. Ο χρήστης είναι περισσότερο ικανοποιημένος από τη συνολική υπηρεσία που του προσφέρεται και είναι πιθανόν να την προτείνει και σε άλλους, και φυσικά να ξανά επισκεφτεί την ιστοσελίδα ή να ξανά χρησιμοποιήσει την εφαρμογή.

1.2 *Αντικείμενο διπλωματικής*

Οι συστάσεις προσφέρουν λύσεις για δύο ανάγκες των χρηστών: την εξερεύνηση και τη βελτιωμένη αναζήτηση. Ξεκινώντας με τη βελτιωμένη αναζήτηση, οι στρατηγικές προτάσεων μπορούν να χρησιμοποιηθούν σε περιπτώσεις όπου οι χρήστες αναζητούν κάτι συγκεκριμένο. Εξετάζοντας πληροφορίες σχετικές με την κοινωνική σύνδεση ενός χρήστη, μπορεί το σύστημα να παρέχει καλύτερη κατάταξη των προτεινόμενων αποτελεσμάτων αναζήτησης, βελτιώνοντας έτσι τη στρατηγική της μηχανής αναζήτησης.

Με την παρούσα διπλωματική προσπαθούμε να καλύψουμε και τις δύο αυτές ανάγκες των χρηστών, δημιουργώντας ένα σύστημα το οποίο παρέχει μια προσωποποιημένη και βελτιωμένη αναζήτηση επιχειρήσεων βάσει κατηγορίας και ενός κειμένου, και την προβολή κριτικών που έχουν γραφτεί γι' αυτές, καθώς και την ανακάλυψη νέων επιχειρήσεων που προκύπτουν από το σύστημα ως προτάσεις μετά την τυχαία επιλογή φωτογραφιών από τον τρέχοντα χρήστη της εφαρμογής.

Ο χρήστης επιθυμεί να του γίνονται συστάσεις με τρόπο αποτελεσματικό ώστε να εξοικονομεί χρόνο και κόπο. Επίσης πολύ σημαντικό ρόλο για τον χρήστη παίζει και η διασφάλιση των ιδιωτικών του στοιχείων κατά τη χρήση τέτοιων συστημάτων. Είναι βέβαια ζωτικής σημασίας για το σύστημα να χρησιμοποιηθούν κάποια ιδιωτικά δεδομένα προκειμένου να παραχθούν ακριβείς συστάσεις. Ωστόσο, θα πρέπει να ληφθεί υπόψη ότι η προστασία της ιδιωτικής ζωής είναι ένα τεράστιο ζήτημα των συστημάτων αυτών γιατί η άρνηση των χρηστών να συμβάλλουν δίνοντας δεδομένα, καθιστά τα εξατομικευμένα περιβάλλοντα μη χρήσιμα [7].

Μελετήσαμε τους αλγορίθμους γράφων και πως αυτοί βοηθούν στην δημιουργία ενός συστήματος προσωποποιημένων συστάσεων βάσει φωτογραφιών, καθώς επίσης και πως ενισχύουν την λειτουργικότητα μιας διαδικτυακής εφαρμογής. Για το λόγο αυτό και με τη χρήση τεχνολογιών state-of-the-art, όπως οι Graph Databases – neo4j, η React και η μηχανική μάθηση (Google Vision AI), αναπτύχτηκε μια διαδικτυακή εφαρμογή προσωποποιημένων συστάσεων για επιχειρήσεις βάσει φωτογραφιών.

Η εφαρμογή μας δίνει τη δυνατότητα στον χρήστη να κάνει αναζήτηση επιχειρήσεων με τη χρήση ενός ελεύθερου κειμένου και μιας κατηγορίας στην οποία ανήκει μια επιχείρηση. Τα αποτελέσματα που του επιστρέφονται είναι προσωποποιημένα αφού του εμφανίζονται επιχειρήσεις και κριτικές που έχουν γίνει από "έμπιστους" φίλους του (ακολουθήθηκαν αλγόριθμοι ομοιότητας, PageRank, Pearson Similarity, Cosine Similarity).

Επιπλέον ζητείται από τον χρήστη να επιλέξει τυχαία πέντε φωτογραφίες, από ένα σύνολο φωτογραφιών, και του γίνονται προτάσεις επιχειρήσεων βάσει αυτής του της επιλογής. Εδώ ακολουθούμε μια προσέγγιση που στηρίζεται στο φιλτράρισμα βάσει περιεχομένου (Content-based filtering), χρησιμοποιώντας τις ετικέτες των φωτογραφιών, για να κάνουμε εξατομικευμένες συστάσεις με τη χρήση των αλγορίθμων γράφων του neo4j. Βρίσκουμε τον βαθμό ομοιότητας ανάμεσα στις φωτογραφίες που επιλέγει ο χρήστης και στις φωτογραφίες του συνόλου δεδομένων μας, χρησιμοποιώντας τον αλγορίθμο Jaccard Similarity. Κάνουμε ομαδοποίηση των όμοιων φωτογραφιών που βρέθηκαν με τον αλγόριθμο Label Propagation και οι επιχειρήσεις των οποίων οι φωτογραφίες ανήκουν στις ίδιες κοινότητες με τις όμοιες φωτογραφίες έρχονται ως σύσταση προς τον χρήστη μας.

Η συγκεκριμένη εργασία είχε ως στόχο τον σχεδιασμό ενός απλού και αποτελεσματικού συστήματος συστάσεων. Πιστεύουμε ότι ο πιο διαισθητικός τρόπος για να περιγράψουμε μια επιχείρηση ή ένα μέρος ή οτιδήποτε άλλο είναι να δείξουμε εικόνες στους χρήστες έτσι ώστε να γνωρίζουν αν θα ήθελαν ή όχι να επιλέξουν κάτι. Χρησιμοποιούμε εικόνες για να δείξουμε διαφορετικά είδη επιχειρήσεων και να βοηθήσουμε τους χρήστες να βρουν επιχειρήσεις που ταιριάζουν καλύτερα στις προτιμήσεις τους, χωρίς πολύπλοκους μηχανισμούς και χωρίς την απαίτηση για συμπλήρωση επιπλέον ιδιωτικών στοιχείων.

Στην ερώτηση “Πώς οι χρήστες χρησιμοποιούν την εφαρμογή μας;” θα λέγαμε : *Αναζητούν για να βρουν και εξερευνούν για να ανακαλύψουν…*

1.3 *Σχετικές εργασίες*

Υπάρχουν πολλές μελέτες σχετικές με συστήματα προσωποποιημένων συστάσεων βάσει φωτογραφιών. Άλλα χρησιμοποιούν σύνολα φωτογραφιών με γεωγραφικές συντεταγμένες (geotagged photos), άλλα φωτογραφίες από τις οποίες κάνουν εξόρυξη λέξεων-κλειδιών και αντικειμένων και άλλα τα οποία ελέγχουν την ομοιότητα ανάμεσα σε φωτογραφίες βάσει περιεχομένου. Τέτοιου είδους συστήματα χρησιμοποιούνται ευρέως στον τομέα του τουρισμού.

Στην εργασία τους οι Risa Kitamura και Takayuki Itoh με τίτλο “ Tourist Spot Recommendation Applying Generic Object Recognition with Travel Photos ” [8] δημιούργησαν ένα σύστημα συστάσεων ταξιδιωτικών προορισμών, βασιζόμενοι σε φωτογραφίες παλαιότερων ταξιδιών του χρήστη από τις οποίες γίνεται εξόρυξη λέξεων κλειδιών και αντικειμένων και με τη χρήση ενός διαδραστικού μηχανισμού ο χρήστης επιλέγει ελεύθερα ανάμεσα σε ένα σύνολο φωτογραφιών το οποίο έχει κατηγοριοποιηθεί

βάσει των λέξεων κλειδιών και του γίνεται η σύσταση τουριστικών προορισμών. Η εφαρμογή http://www.pixmeaway.com/ δίνει τη δυνατότητα στον χρήστη να επιλέγει τυχαία 7 φωτογραφίες και με βάση αυτή του την επιλογή του προτείνονται επίσης τουριστικοί προορισμοί.

Ένα σύστημα συστάσεων που χρησιμοποιεί φωτογραφίες με γεωγραφικές συντεταγμένες είναι και αυτό που αναπτύσσουν στην εργασία τους οι Liangliang Cao, JieboLuo, Andrew Gallagher,XinJin , JiaweiHan, Thomas S. Huang με τίτλο "A worldwide tourism recommendation systembased on geotagged web photos" [9]. Συστήνουν την ομαδοποίηση των geotagged φωτογραφιών σε κοινότητες, και στη συνέχεια βρίσκουν σε κάθε μια κοινότητα τις πιο αντιπροσωπευτικές φωτογραφίες. Οι συστάσεις παράγονται από τη σύγκριση των φωτογραφιών που επιλέγει ο χρήστης ή των λέξεων κλειδιών που εισάγει στην εφαρμογή, με τις αντιπροσωπευτικές φωτογραφίες κάθε κοινότητας, κάτω από την προϋπόθεση του " Αν σας αρέσει αυτό το μέρος, τότε πιθανόν να σας αρέσουν και εκείνα που του μοιάζουν".

Βασιζόμενοι λοιπόν κι εμείς στη γενική τάση της δημιουργίας συστημάτων συστάσεων βάσει φωτογραφιών, προχωρήσαμε στη δημιουργία απλούστερων εξατομικευμένων συστάσεων επιχειρήσεων βάσει φωτογραφιών. Οι συστάσεις βασίζονται στο περιεχόμενο (content based filtering) και την ομοιότητα που υπάρχει ανάμεσα σε φωτογραφίες βάσει των ετικετών που έχουν εντοπιστεί σε αυτές με τη χρήση μηχανικής μάθησης.

Τη συγκεκριμένη προσέγγιση για τη δημιουργία εξατομικευμένων συστάσεων βάσει φωτογραφιών με τη χρήση του neo4j που ακολουθήσαμε, την έχει αναπτύξει και η ομάδα του neo4j στα πλαίσια σεμιναρίου εκμάθησης του πώς χρησιμοποιούνται οι αλγόριθμοι γράφων στο neo4j για την ενίσχυση της λειτουργικότητας μιας εφαρμογής [10].

Δεν έχουμε εντοπίσει εμπορική εφαρμογή που δημιουργεί συστάσεις για επιχειρήσεις βάσει φωτογραφιών. Αντιθέτως, υπάρχουν πολλές εμπορικές εφαρμογές που συστήνουν προϊόντα και τουριστικούς προορισμούς στηρίζοντας τις συστάσεις τους στην απλή επιλογή φωτογραφιών και στην ομοιότητα περιεχομένου τους.

1.4 *Διάρθρωση της εργασίας*

Η διάρθρωση της εργασίας μας έχει ως εξής:

Στο Κεφάλαιο 1 περιγράφουμε το τι είναι "Συστήματα Συστάσεων" , καθώς και τις κατηγορίες στις οποίες διακρίνονται. Αναλύεται ο στόχος της συγκεκριμένης διπλωματικής

και παραθέτονται σχετικές εργασίες που έχουν γίνει στη δημιουργία συστημάτων συστάσεων βάσει φωτογραφιών.

Στο Κεφάλαιο 2 εισάγουμε και αναλύουμε την έννοια των βάσεων δεδομένων γράφων (Graph Databases) και πως φτάσαμε ως εκεί. Περιγράφουμε την πιο γνωστή Graph Database, τη Neo4j, αναλύοντας την αρχιτεκτονική της, καθώς και την ανάγκη χρήσης τέτοιου είδους NoSql βάσεων δεδομένων, κάτι που προκύπτει από τη σύγκριση της με τις απλές σχεσιακές βάσεις δεδομένων RDBMS. Στη συνέχεια αναφερόμαστε στη γλώσσα ερωτημάτων Cypher που χρησιμοποιεί η neo4j και τέλος παρουσιάζουμε τους τύπους αλγορίθμων γράφων (Centrality, Community Detection, Similarity) που χρησιμοποιούνται για την υλοποίηση της εφαρμογής μας.

Στο Κεφάλαιο 3 αναλύουμε το πρόβλημα και το σύνολο δεδομένων που χρησιμοποιήσαμε. Γίνεται αναφορά στα τεχνολογικά και προγραμματιστικά εργαλεία που χρησιμοποιήσαμε για τη δημιουργία της βάσης και της παραμετροποίησης της, την εισαγωγή των δεδομένων στη βάση και την αποκατάσταση της επικοινωνίας της βάσης με την διαδικτυακή εφαρμογή που αναπτύχτηκε σε React.

Στο Κεφάλαιο 4 παρουσιάζουμε και αναλύουμε τη γραφική διεπαφή της διαδικτυακής εφαρμογής συστάσεων επιχειρήσεων βάσει φωτογραφιών. Βασιζόμενοι στη θεωρητική ανάλυση των αλγορίθμων γράφων που έγινε στο κεφάλαιο 2, αναπτύξαμε αποδοτικά ερωτήματα σε γλώσσα cypher για να αποδείξουμε την ομοιότητα ανάμεσα σε χρήστες και κατηγορίες επιχειρήσεων, καθώς και ανάμεσα σε φωτογραφίες οι οποίες συνδέονται με επιχειρήσεις. Επίσης αναπτύχθηκαν ερωτήματα για την εμφάνιση των πιο σχετικών κριτικών για τις επιχειρήσεις που ενδιαφέρουν τον χρήστη μας. Για τη δημιουργία των συστάσεων βάσει φωτογραφιών χρησιμοποιήθηκε το εργαλείο Google Vision AI το οποίο χρησιμοποιεί μηχανική μάθηση για την ανίχνευση ετικετών σε φωτογραφίες (Label Detection). Ο αλγόριθμος ομοιότητας Jaccard χρησιμοποιήθηκε για τον προσδιορισμό παρόμοιων φωτογραφιών και με τον αλγόριθμο Label Propagation κάναμε την ομαδοποίηση των παρόμοιων φωτογραφιών σε κοινότητες (clusters).

Στο Κεφάλαιο 5 κλείνουμε με τα συμπεράσματα που πρόεκυψαν και αναφέρουμε μελλοντικές επεκτάσεις της παρούσας διπλωματικής εργασίας.

2

Βάσεις δεδομένων γράφων και Neo4j

2.1 *Η εξέλιξη των βάσεων δεδομένων*

Μια βάση δεδομένων είναι μια οργανωμένη συλλογή σχετιζόμενων δεδομένων, η οποία αποθηκεύεται και προσπελαύνεται ηλεκτρονικά από ένα υπολογιστικό σύστημα [11].

Στα μέσα της δεκαετίας του ’60 με την εμφάνιση των ηλεκτρονικών υπολογιστών και τη χρήση τους από τις επιχειρήσεις είχαμε την ανάπτυξη εφαρμογών οι οποίες δημιούργησαν την ανάγκη για αύξηση της χωρητικότητας αποθήκευσης των δεδομένων τους. Εκείνη την εποχή τυποποιήθηκαν οι δικτυακές βάσεις δεδομένων (network databases), οι οποίες είναι γνωστές και με το όνομα βάσεις δεδομένων CODASYL (IntegratedDataStore(IDS) /Codasyl) και έχουμε και την πρωτοεμφάνιση των Συστημάτων Διαχείρισης Πληροφορίας - IMS, το πιο γνωστό σύστημα διαχείρισης ιεραρχικών βάσεων δεδομένων (hierarchical DBMS) [12].

Στα 1970 ο E. F. Codd εισήγαγε το Σχεσιακό Μοντέλο Βάσεων Δεδομένων (RDBMS) το οποίο έφερε την επανάσταση στον χώρο των βάσεων δεδομένων εκείνης της εποχής, μιας και χαρακτηριζόταν από τη δυνατότητα μαζικής επεξεργασίας των δεδομένων και από την ανεξαρτησία των εφαρμογών από τη φυσική υλοποίηση. Στο σχεσιακό μοντέλο βάσεων δεδομένων έχουμε την οργάνωση των δεδομένων σε πίνακες [12].

Στις αρχές της δεκαετίας του 1980, ξεκινά μια περίοδος που σηματοδοτείται από την έλευση των σχεσιακών βάσεων δεδομένων (Relational Database Systems) με χρήση της SQL, μια γλώσσα δομημένων ερωτημάτων.

Οι βάσεις δεδομένων σχεδιάστηκαν αρχικά για την αποθήκευση και την ανάκτηση δεδομένων. Στην περίπτωση όμως των ισχυρά διασυνδεδεμένων δεδομένων η απόδοση των κλασσικών σχεσιακών βάσεων δεδομένων δεν ήταν τόσο μεγάλη μιας και τα δεδομένα σ ΄ αυτές οργανώνονται σε πίνακες που συνδυάζονται μεταξύ τους με χρήση εντολών JOIN. Όσο μεγαλύτεροι γίνονται οι πίνακες δεδομένων, τα ερωτήματα ανάκτησης στην βάση, που περιέχουν εντολές JOIN, επιβαρύνουν αισθητά το χρόνο εκτέλεσης επομένως και απόκρισης των ερωτημάτων αυτών.

Με το πέρασμα των χρόνων και την εμφάνιση του πρωτοκόλλου Web 2.0 καθώς και των Διαδικτύου των πραγμάτων (Internet of Things – IoT), Υπολογιστικού νέφους (Cloud Computing) και των Μεγάλων δεδομένων (Big Data), τα δεδομένα συσχετίζονταν και συνδέονταν μεταξύ τους ολοένα και περισσότερο, οπότε τα ερωτήματα JOIN έγιναν ακόμα πιο πολύπλοκα, οδηγώντας στο να φτάσουμε σε μεγάλη μείωση της απόδοσης των κλασσικών σχεσιακών βάσεων δεδομένων. Έγιναν αρκετές προσπάθειες ώστε να βελτιωθούν, να είναι πιο αποτελεσματικές στην περίπτωση των ισχυρά διασυνδεδεμένων δεδομένων και στα τέλη της δεκαετίας του ’90 έγινε σαφές πως εξίσου σημαντικές με τα δεδομένα προς αποθήκευση και την ανάκτησή τους είναι και οι συνδέσεις μεταξύ τους. Έτσι σχεδιάστηκαν οι μη-σχεσιακές βάσεις δεδομένων (NoSQL databases).

Οι μη-σχεσιακές βάσεις δεδομένων (NoSQL databases) σε αντίθεση με τις σχεσιακές (SQL databases) δεν απαιτούν την οργάνωση των δεδομένων σε μεγάλους πίνακες, αλλά μπορούν να αποθηκεύσουν τα δεδομένα με ανομοιόμορφο ή μη κανονικοποιημένο τρόπο στην μνήμη αποφεύγοντας την χρήση της εντολής JOIN σε μεγάλους πίνακες, πετυχαίνοντας έτσι έναν σχετικά σταθερό χρόνο εκτέλεσης των εφαρμογών.

Στην αρχή της δεκαετίας του 2000, εμφανίστηκε μια νέα κατηγορία των No-SQL βάσεων δεδομένων, οι λεγόμενες Graph Databases (δηλαδή βάσεις δεδομένων με οργάνωση γράφου), με στόχο την αποτελεσματικότερη λύση θεμάτων μοντελοποίησης επιστημονικών προβλημάτων, σε σχέση με τη μοντελοποίηση με χρήση πινάκων από τις κλασσικές βάσεις δεδομένων.

Οι No-SQL βάσεις δεδομένων γράφων δεν αποθηκεύουν τα δεδομένα σε διαφορετικούς πίνακες ή σε διαφορετικά έγγραφα, αλλά σε μία μοναδική δομή δεδομένων, τον γράφο. Είναι σχεδιασμένες να χειρίζονται τις σχέσεις μεταξύ των δεδομένων εξίσου σημαντικά με τα ίδια τα δεδομένα.

Αξίζει να σημειωθεί πως σε μεγάλους οργανισμούς χρησιμοποιούνται ευρέως ακόμη οι σχεσιακές βάσεις δεδομένων και όχι οι No-SQL βάσεις. Και αυτό γιατί φοβούνται και δυσκολεύονται να αντικαταστήσουν τα συστήματά τους, λόγω του υψηλού κόστους αναβάθμισης (πολύ μεγάλος όγκος δεδομένων), αλλά και λόγω του ότι οι βάσεις αυτές είναι

αξιόπιστες και καλά δοκιμασμένες. Σε κάθε περίπτωση, οι ειδικοί θεωρούν τις No-SQL βάσεις δεδομένων ως την τελευταία λέξη της τεχνολογίας.

Είναι και δική μας άποψη πως οι βάσεις δεδομένων γράφων (Graph Databases) αποτελούν μια απ 'τις καλύτερες επιλογές, όσον αφορά την διαχείριση δεδομένων και την οπτικοποίηση τους. Τα τελευταία χρόνια τόσο πανεπιστήμια όσο και επιχειρήσεις αναγνωρίζουν πόσο σημαντική είναι η γνώση που παράγεται από τη σύνδεση των δεδομένων που συλλέγονται στο πέρασμα του χρόνου.

Οι Graph Databases χρησιμοποιούνται σε διάφορες εφαρμογές, μια από τις οποίες είναι και η δημιουργία real-time συστημάτων εξατομικευμένων συστάσεων (Real-time personalized recommendation systems), όπως είναι και η περίπτωση της εργασίας μας.

2.2 *Γιατί Βάσεις Δεδομένων Γράφων*

Οι μεγάλοι οργανισμοί και επιχειρήσεις δεν χρειάζεται μόνο να διαχειριστούν τα Big Data, αλλά πρέπει να δημιουργήσουν και επιπλέον γνώση από τα δεδομένα αυτά. Στην περίπτωση αυτή οι σχέσεις μεταξύ των δεδομένων είναι πολύ πιο πολύτιμες από τα ίδια τα δεδομένα. Προκειμένου λοιπόν να αξιοποιήσουν τις σχέσεις αυτές, χρειάζονται και μια τεχνολογία βάσεων δεδομένων που να αποθηκεύει πληροφορίες γι' αυτές τις σχέσεις. Αυτή η τεχνολογία είναι μια βάση δεδομένων γράφων (Graph Database).

Επομένως αν θα θέλαμε να δώσουμε μια απάντηση στο ερώτημα " Γιατί Graph Databases ; " θα λέγαμε πως πέρα από την υψηλή απόδοση και τη δυνατότητα να αποθηκεύουν τις σχέσεις μεταξύ των δεδομένων, παίζει ρόλο και η μεγάλη ευελιξία που παρουσιάζουν όταν επεκτείνεται το μοντέλο δεδομένων ή οι ανάγκες της επιχείρησης, χωρίς να μπαίνει σε κίνδυνο η λειτουργικότητα των εφαρμογών τους. Ένας γράφος μπορεί να επεκταθεί σε μέγεθος και πολυπλοκότητα καθώς η εφαρμογή εξελίσσεται, με μικρό αντίκτυπο στις επιδόσεις της. Είτε ξεκινά η ανάπτυξη νέων εφαρμογών, είτε αυξάνεται η υπάρχουσα λειτουργικότητα, οι Graph Databases περιορίζονται μόνο από το φυσικό υλικό.

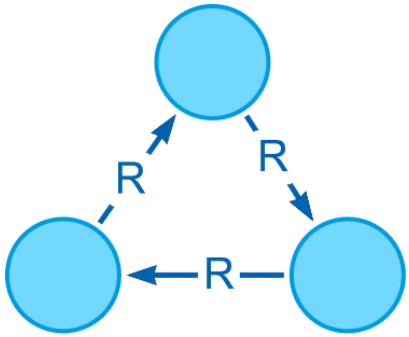
Όταν μιλάμε για Graph Database μιλάμε για ένα ηλεκτρονικό σύστημα διαχείρισης βάσεων δεδομένων το οποίο υποστηρίζει τις λειτουργίες Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή (CRUD) και εργάζεται σε ένα μοντέλο δεδομένων γράφου [13]. Μία graph βάση δεδομένων περιέχει κόμβους (nodes) και σχέσεις (relationships). Οι σχέσεις οργανώνουν τον γράφο, με τους κόμβους να είναι οργανωμένοι σε αυθαίρετες δομές, επιτρέποντας ένα γράφημα να αποτελεί μια λίστα, ένα δέντρο, ένα χάρτη ή μια οργανωμένη οντότητα, οι οποίες μπορούν να συνδυαστούν δημιουργώντας πιο περίπλοκες και διασυνδεδεμένες δομές.

Ο όρος γράφος που είναι ομώνυμος με τον αντίστοιχο όρο από την θεωρία γράφων στα μαθηματικά, αναφέρεται ως:

“Ένας γράφος είναι ένας αφηρημένος τύπος δεδομένων που αποτελείται από ένα πεπερασμένο σύνολο κόμβων και ακμών τα οποία γραφικά απεικονίζονται ως κύκλοι και βέλη αντίστοιχα [14].”

Ως κόμβοι μπορεί να είναι οντότητες όπως, για παράδειγμα, άνθρωποι, αντικείμενα, τοποθεσίες, κατηγορίες κλπ και οι ακμές μεταξύ αυτών των κόμβων είναι οι σχέσεις μεταξύ αυτών των οντοτήτων. Οι γράφοι μπορούν να χρησιμοποιηθούν για να μοντελοποιήσουν μια ευρεία γκάμα οντοτήτων, όπως ένα οδικό δίκτυο, ένα κοινωνικό δίκτυο, μια ιστοσελίδα ή οποιαδήποτε άλλη συλλογή αντικειμένων που μπορούν να συνδεθούν μεταξύ τους [15].

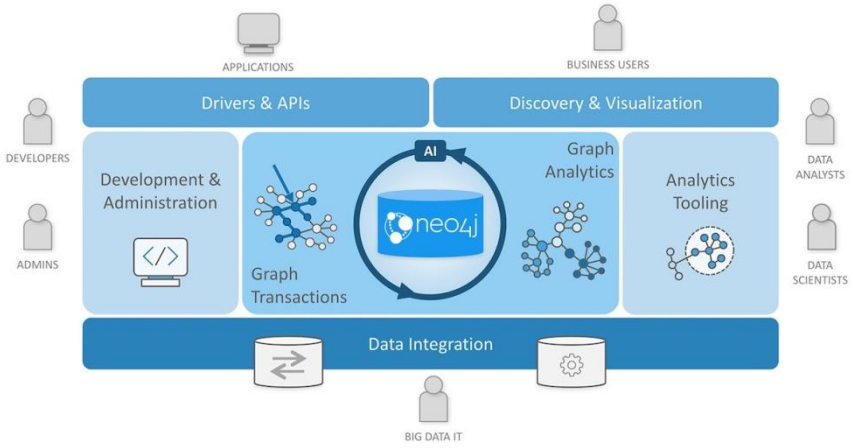
Στην παρακάτω Εικόνα 2.1 φαίνεται το παράδειγμα ενός γράφου, και πιο συγκεκριμένα ενός κατευθυνόμενου γράφου με τρεις κόμβους και τρεις ακμές οι οποίες έχουν την ετικέτα R.



Εικόνα 2.1: Παράδειγμα γράφου

2.3 *Το Neo4j ως Βάση Δεδομένων Γράφων*

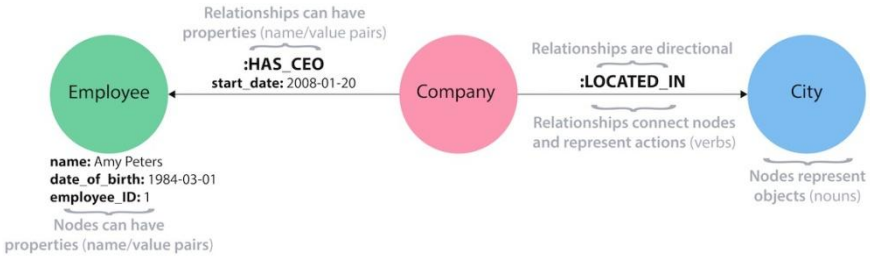
Το Neo4j είναι μια βάση δεδομένων για γράφους η οποία ακολουθεί την αρχή του ACID (*Atomicity, Consistency, Isolation, Durability*), χρησιμοποιώντας ένα σύστημα που χτίζεται με το μοντέλο του γράφου για την αποθήκευση και επεξεργασία των δεδομένων [13]. Η αρχή ACID (*ατομικότητα, συνέπεια, απομόνωση, μονιμότητα*) είναι ένα σύνολο ιδιοτήτων το οποίο εγγυάται την αξιόπιστη λειτουργία της βάσης δεδομένων [16]. Στην Εικόνα 2.2 φαίνεται η ολιστική αρχιτεκτονική που ακολουθεί το Neo4j .



Εικόνα 2.2 : Αρχιτεκτονική του neo4j (Πηγή [17])

Είναι ανοιχτού κώδικα λογισμικό που δημιουργήθηκε σε γλώσσα Java, ενώ η γλώσσα που χρησιμοποιεί για την προσπέλαση των δεδομένων είναι η γλώσσα ερωτημάτων Cypher, για την οποία θα μιλήσουμε πιο αναλυτικά σε επόμενη ενότητα.

Ακολουθεί το μοντέλο native property graph σύμφωνα με το οποίο το γράφημα περιέχει κόμβους (οντότητες) οι οποίοι συνδέονται μεταξύ τους με ακμές (σχέσεις) . Οι γράφοι αυτού του τύπου επιτρέπουν την ανάθεση ετικετών (labels) και ιδιοτήτων (properties) στους κόμβους και στις ακμές του γράφου. Επιπλέον, κάθε κόμβος και ακμή έχουν ένα μοναδικό αναγνωριστικό το οποίο μπορεί να χρησιμοποιηθεί ως σημείο αναφοράς για την συσχέτιση επιπλέον μεταδεδομένων σε κάθε κόμβο, με την μορφή ζευγών κλειδιών-τιμών [18]. Στην παρακάτω Εικόνα 2.3 φαίνεται ένα τέτοιο μοντέλο.



Εικόνα 2.3 : Label Property Graph (Πηγή[19])

Όπως βλέπουμε το μοντέλο περιέχει κόμβους, σχέσεις, ιδιότητες και ετικέτες:

- οι κόμβοι είναι τα κύρια στοιχεία και συνδέονται μέσω σχέσεων

- οι κόμβοι μπορούν να έχουν μία ή περισσότερες ιδιότητες (δηλ., ζεύγη κλειδιών / τιμών) και ετικέτες
- οι σχέσεις είναι κατευθυνόμενες και έχουν μία ή περισσότερες ιδιότητες
- οι ιδιότητες ονομάζονται τιμές, όπου το όνομα (κλειδί) είναι μια συμβολοσειρά
- οι ετικέτες χρησιμοποιούνται για την ομαδοποίηση των κόμβων σε σύνολα (ομάδες)

Το neo4j υποστηρίζει την εξαγωγή δεδομένων μέσω ερωτημάτων σε μορφή json και xls. Λόγω του REST API που διαθέτει μπορεί να χρησιμοποιηθεί με οποιαδήποτε γλώσσα προγραμματισμού. Μέσω των neo4j javascript drivers και της Java Script μπορεί να είναι προσβάσιμη από οποιοδήποτε UI MVC Framework όπως είναι η Node JS και την οποία έχουμε χρησιμοποιήσει στην εργασία μας. Το πρόγραμμα περιήγησης (neo4j browser) που διαθέτει ως ενσωματωμένη web εφαρμογή, μπορεί να χρησιμοποιηθεί για την ανάκτηση και τη δημιουργία των δεδομένων του γράφου.

Το Neo4j είναι η πιο δημοφιλής βάση δεδομένων γράφων σήμερα [20]. Σχεδιάστηκε και υλοποιήθηκε ώστε να είναι μια αξιόπιστη βάση δεδομένων, βελτιστοποιημένη για δομές γράφων, αντί για πίνακες. Δουλεύοντας με το Neo4j, μία εφαρμογή παίρνει όλη την εκφραστικότητα ενός γράφου, με όλη την αξιοπιστία μιας βάσης δεδομένων.

2.4 *Σύγκριση RDBMS και Graph Data Bases (Neo4J)*

Παραπάνω αναφερθήκαμε σε χαρακτηριστικά και ιδιότητες και των δύο ειδών βάσεων δεδομένων, των σχεσιακών βάσεων και των βάσεων δεδομένων γράφων (συγκεκριμένα της Neo4j). Θα τα παρουσιάσουμε συγκεντρωτικά στον ακόλουθο πίνακα [15] [21] :

Χαρακτηριστικά	RDBMS	Neo4j
Αποθήκευση δεδομένων	Αποθήκευση δεδομένων σε πίνακες	Αποθήκευση δεδομένων σε γράφους
Επιδόσεις	Έχουν μειωμένη απόδοση όσο μεγαλύτεροι γίνονται οι πίνακες δεδομένων. Τα ερωτήματα ανάκτησης στην βάση που περιέχουν JOIN	Έχει μεγάλη απόδοση, δεν παρουσιάζει καθυστερήσεις στην εκτέλεση των ερωτημάτων όποιο κι αν είναι το βάθος των σχέσεων

	επιβαρύνουν αισθητά το χρόνο εκτέλεσης (και κατ ’επέκταση απόκρισης) των ερωτημάτων αυτών.	των δεδομένων ή ο αριθμός τους . Διαθέτει τους κατάλληλους μηχανισμούς ώστε η πιο σημαντική αποστολή του συστήματος να είναι η βάση πάντα διαθέσιμη και αξιόπιστη.
Γλώσσα Ερωτημάτων	SQL	Cypher
Επεξεργασία σε μεγάλη κλίμακα	Οποιαδήποτε αλλαγή ή επέκταση του μοντέλου της βάσης, αυξάνει το κόστος. Αύξηση πολυπλοκότητας => Αύξηση κόστους	Έχει σχεδιαστεί για τη διασύνδεση δεδομένων ακόμη και μεγάλης κλίμακας
Μοντελοποίηση	Schema based	Graph data model
Υποστήριξη ACID	Πλήρης υποστήριξη	Πλήρης υποστήριξη
Εξαγωγή Δεδομένων	Εξάγουν δεδομένα σε διάφορους τύπους	Εξάγει δεδομένα σε μορφή JSON και xls (csv)

Πίνακας 2.1 : Σύγκριση RDBMS και Neo4j

2.5 *Η γλώσσα ερωτημάτων Cypher*

Η Cypher είναι μία δηλωτική γλώσσα υποβολής ερωτημάτων για το property graph μοντέλο, και είναι η βασική γλώσσα που υποστηρίζει η neo4j. Δίνει τη δυνατότητα υποβολής ερωτημάτων τόσο για τη διαχείριση των δεδομένων σε ένα property graph, όσο και για την ενημέρωση της δομής του γράφου [22]. Είναι μια ανθρώπινη γλώσσα ερωτημάτων (human readable) και έχει σχεδιαστεί ώστε να είναι απλή και ταυτόχρονα ισχυρή, κατάλληλη τόσο για τους προγραμματιστές όσο και για τους επιχειρηματίες οι οποίοι θέλουν να εκμεταλλευτούν όλες τις δυνατότητες που τους παρέχονται και να εκτελούν ad-hoc ερωτήματα στη βάση δεδομένων [23]. Με τον όρο ad-hoc εννοείται ότι μπορούν να υπάρχουν έτοιμα ερωτήματα σε κάποιες εφαρμογές που αλλάζουν τα ορίσματα δυναμικά.

Η κεντρική ιδέα της γλώσσας Cypher είναι η αναζήτηση μοτίβων, τα οποία αναπαρίστανται ως εξής : (a) - [r] -> (b) όπου a,b μέσα σε παρενθέσεις είναι οι κόμβοι, r είναι η σχέση

μεταξύ αυτών των κόμβων και το βέλος δείχνει την κατεύθυνση της σχέσης, και είναι προαιρετικό. Με τη βοήθεια της εντολής MATCH η Cypher αναζητά τέτοιου είδους μοτίβα μέσα στον γράφο. Υπάρχουν κι άλλες εντολές με τις οποίες μπορούμε να αλλάξουμε τη δομή του γράφου. Η εντολή CREATE χρησιμοποιείται για τη δημιουργία νέου κόμβου ή νέας σχέσης, το DELETE για την διαγραφή κόμβων ή σχέσεων και το SET για την αλλαγή των ιδιοτήτων κόμβων ή σχέσεων. Υπάρχει και ο όρος MERGE ο οποίος αναζητά μοτίβα στον γράφο όπως το MATCH, με την διαφορά όμως ότι στην περίπτωση που το μοτίβο δεν υπάρχει, το δημιουργεί [22].

Μερικές ακόμη εντολές στη Cypher είναι οι :

- WHERE: Κριτήρια φιλτραρίσματος.
- RETURN: Τι θα επιστραφεί από το ερώτημα.
- FOREACH: Εκτελεί ενημερώσεις για κάθε στοιχείο της λίστας.
- WITH: Διαιρεί ένα ερώτημα σε πολλαπλά και διακριτά μέρη.

Ένα παράδειγμα ερωτήματος σε Cypher φαίνεται παρακάτω:

```
MATCH (user)-[:FRIEND]->(friend)
WITH user, count(friend) as friends
WHERE friends > 15
RETURN user
```

Με το παραπάνω παράδειγμα βρίσκουμε τους κόμβους `user` που συνδέονται με τη σχέση `FRIEND` με τους κόμβους `friend`, τους οποίους καταμετρούμε και αν αυτοί είναι περισσότεροι από 15 επιστρέφουμε τους `user`.

2.6 *Τύποι αλγορίθμων γράφων*

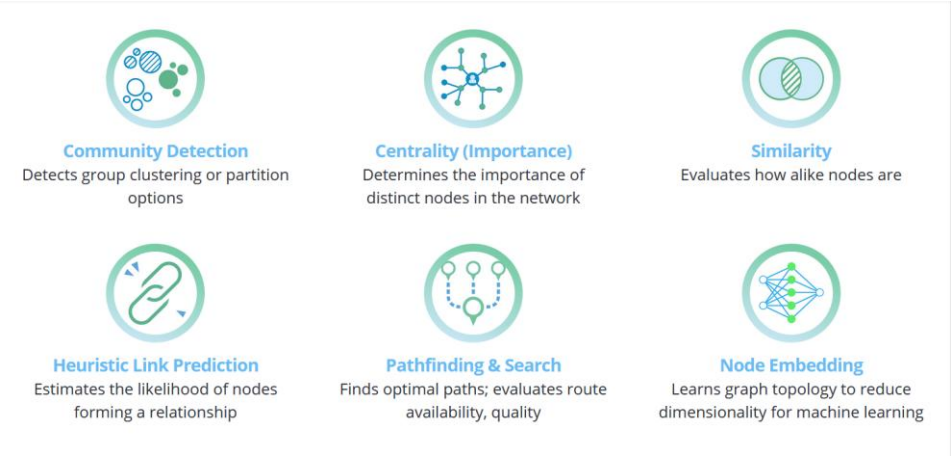
Οι αλγόριθμοι γράφων (Graphs Algorithms) παρέχουν μία από τις πιο ισχυρές προσεγγίσεις για την ανάλυση διασυνδεδεμένων δεδομένων. Χρησιμοποιούν τις συνδέσεις μεταξύ των δεδομένων για να αξιολογήσουν και να συμπεράνουν την οργάνωση και τη δυναμική σύνθετων συστημάτων και να φέρουν στο φως πολύτιμες πληροφορίες που είναι κρυμμένες

στις σχέσεις μεταξύ των δεδομένων. Μπορούν να παρέχουν πληροφορίες σχετικά με τις οντότητες του γράφου (τις τάξεις, την κατάταξη) ή τις εγγενείς δομές όπως οι κοινότητες (ανίχνευση κοινοτήτων, διαχωρισμό γραφημάτων, ομαδοποίηση). Είναι εύκολοι στη χρήση, εκτελούνται γρήγορα και παράγουν ισχυρά αποτελέσματα [24].

Το Neo4j προσφέρει μια αναπτυσσόμενη ανοικτή βιβλιοθήκη αλγορίθμων, που βελτιστοποιείται συνεχώς. Οι αλγόριθμοι υπάρχουν ως διαδικασίες, οι οποίες καλούνται άμεσα χρησιμοποιώντας ερωτήματα Cypher στο πρόγραμμα περιήγησης του Neo4j.

Οι αλγόριθμοι γράφων ομαδοποιούνται στις εξής κατηγορίες στο Neo4j [18]:

- Centrality
- Community Detection
- Similarity
- Link Prediction
- Pathfinding & Search
- Node Embedding

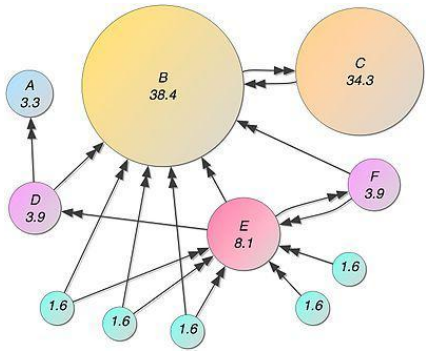


Εικόνα 2.4 : Αλγόριθμοι Γράφων (Πηγή [18])

Στην εργασία μας χρησιμοποιήθηκαν αλγόριθμοι από τις κατηγορίες Community Detection, Centrality και Similarity.

2.6.1 *Centrality Algorithms*

Οι αλγόριθμοι κεντρότητας (Centrality Algorithms) χρησιμοποιούνται για την κατανόηση των ρόλων συγκεκριμένων κόμβων σε ένα γράφημα και τον αντίκτυπό τους στο δίκτυο. Είναι ένα εξαιρετικό εργαλείο για τον εντοπισμό των σημαντικότερων κόμβων μέσα σε ένα δίκτυο. Μπορούμε να εντοπίσουμε τα χαρακτηριστικά με τη μεγαλύτερη επιρροή μέσα στα μοντέλα μας, εξαλείφοντας έτσι τα λιγότερο σημαντικά χαρακτηριστικά και μειώνοντας το overfiting. Δημιουργήθηκαν για τη μέτρηση διαφορετικών πραγμάτων, όπως για παράδειγμα τον υπολογισμό του πόσο γρήγορα μπορεί να διαδοθεί μια πληροφορία μεταξύ διαφορετικών ομάδων ή για το πόσο αξιόπιστος είναι ένας κόμβος. Ένας βασικός αλγόριθμος κεντρότητας είναι ο PageRank [25].



Εικόνα 2.5 : PageRank Αλγόριθμος

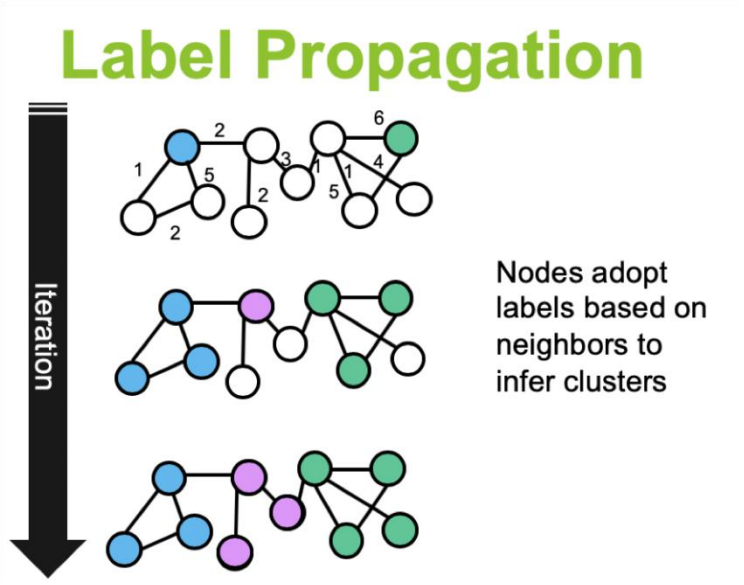
Αλγόριθμος	Τι κάνει	Παράδειγμα Χρήσης
PageRank	Εκτιμά τη σημαντικότητα ενός κόμβου. Η κατάταξη ενός κόμβου προέρχεται από τον αριθμό και την ποιότητα των συνδέσεων του με έναν κόμβο υψηλής επιρροής.	Στα μέσα κοινωνικής δικτύωσης για να σου προτείνονται χρήστες για να τους ακολουθήσεις . Επίσης χρησιμοποιείται και στη μηχανική μάθηση για την εύρεση των πιο σημαντικών χαρακτηριστικών που θα χρησιμοποιηθούν στα μοντέλα εκπαίδευσης.

2.6.2 *Community Detection Algorithms*

Ο σχηματισμός κοινοτήτων είναι ίδιος σε όλους τους τύπους των δικτύων και ο προσδιορισμός τους είναι απαραίτητος στην αξιολόγηση της συμπεριφοράς μιας κοινότητας -

ομάδας. Η γενική αρχή στην εύρεση κοινοτήτων είναι πως τα μέλη της κοινότητας έχουν περισσότερες σχέσεις με κόμβους εντός της κοινότητας παρά με κόμβους εκτός αυτής. Δημιουργούνται συστάδες κόμβων ή και απομονωμένοι κόμβοι, με κοινά χαρακτηριστικά και συμπεριφορές, που μας βοηθούν να κάνουμε περεταίρω αναλύσεις. Οι αλγόριθμοι σχηματισμού κοινοτήτων χρησιμοποιούνται για την οπτικοποίηση δικτύων και την παρακολούθηση τους [26]. Στην κατηγορία αυτή αλγορίθμων ανήκει ο αλγόριθμος Label Propagation.

Αλγόριθμος	Τι κάνει	Παράδειγμα Χρήσης
Label Propagation	Εκχωρεί ετικέτες σε κόμβους και δημιουργεί κοινότητες βάσει αυτών των ετικετών. Είναι ένας γρήγορος αλγόριθμος εξεύρεσης κοινοτήτων σε έναν γράφο και αποτελεί συχνά πρωταρχικό βήμα για άλλες αναλύσεις..	Έχει ποικίλες εφαρμογές, από την κατανόηση του σχηματισμού συναινετικών δομών σε κοινωνικά σύνολα έως τον εντοπισμό ομάδων πρωτεϊνών που συμμετέχουν μαζί σε μια διαδικασία βιοχημικών δικτύων. Χρησιμοποιείται επίσης στη μηχανική μάθηση με μερική ή και χωρίς επίβλεψη εκπαίδευση των μοντέλων ως αρχικό βήμα προεπεξεργασίας.

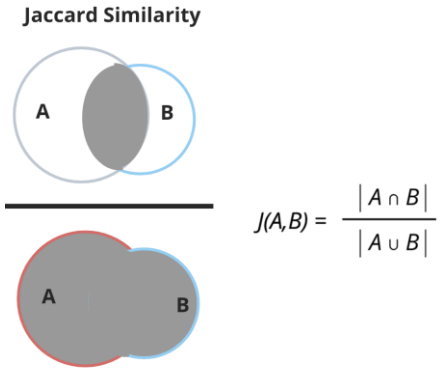


Εικόνα 2.6 : Label Propagation (Πηγή [13])

2.6.3 Similarity Algorithms

Οι αλγόριθμοι ομοιότητας (Similarity Algorithms) αξιολογούν κατά πόσο είναι παρόμοιοι κόμβοι μεταξύ τους, βάση των ιδιοτήτων τους ή με βάση τους γειτονικούς τους κόμβους ή τις ιδιότητες των σχέσεων μεταξύ τους [27].

Ο αλγόριθμος ομοιότητας **Jaccard** (Jaccard Similarity) μετρά πόσο όμοια είναι δύο σύνολα μεταξύ τους. Πρόκειται για έναν αλγόριθμο σύγκρισης συνόλων και υπολογίζεται διαιρώντας το μέγεθος της τομής των δύο συνόλων με το μέγεθος της ένωσής τους.

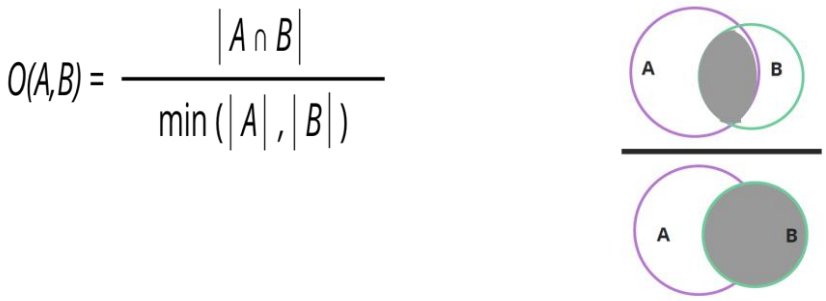


Εικόνα 2.7 : Αλγόριθμος ομοιότητας Jaccard

Δύο κόμβοι θεωρούνται παρόμοιοι εάν μοιράζονται πολλούς από τους ίδιους γείτονες. Μετά τον υπολογισμό της ομοιότητας μεταξύ δύο συνόλων θα μπορούσαμε να τη χρησιμοποιήσουμε σαν μέρος ενός ερωτήματος για τη δημιουργία συστάσεων.

Ο αλγόριθμος **Overlap** μετρά την υπερκάλυψη που υπάρχει ανάμεσα σε δύο σύνολα δεδομένων. Ορίζεται ως το μέγεθος της τομής των δύο συνόλων που διαιρείται με το μικρότερο σε μέγεθος σύνολο:

Overlap Similarity



Εικόνα 2.8 : Αλγόριθμος ομοιότητας Overlap

Μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο ομοιότητας Overlap για να βρούμε υποσύνολα μεγαλύτερων συνόλων. Είναι ιδανική επιλογή για την εύρεση ιεραρχίας στα δεδομένα και την ανάπτυξη υπερ και υποκατηγοριών. Ο συντελεστής ομοιότητας Overlap αντιπροσωπεύει τα κοινά αντικείμενα μεταξύ ομάδων.

Ο αλγόριθμος **Pearson** υπολογίζει την συσχέτιση που υπάρχει ανάμεσα σε δύο πράγματα. Για παράδειγμα μπορεί να υπολογίσει την ομοιότητα ανάμεσα σε δύο λίστες αριθμών. Στη συνέχει η υπολογιζόμενη ομοιότητα μπορεί να χρησιμοποιηθεί στη δημιουργία εξατομικευμένων συστάσεων. Υπολογίζεται με βάση τον ακόλουθο τύπο :

$$similarity(A,B) = \frac{\text{cov}(A,B)}{\sigma_A \sigma_B} = \frac{\sum_{i=1}^{n}(A_i - \overline{A})(B_i - \overline{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \overline{A})^2 (B_i - \overline{B})^2}}$$

Εικόνα 2.9 : Pearson Similarity

Οι τιμές κυμαίνονται μεταξύ -1 και 1, με το -1 να σημαίνει πως τα αντικείμενα είναι απολύτως ανόμοια και το 1 να σημαίνει πως τα αντικείμενα είναι απόλυτα όμοια μεταξύ τους [23].

Αλγόριθμος	Τι κάνει	Παράδειγμα Χρήσης
Jaccard Similarity	Βρίσκει την ομοιότητα ανάμεσα σε δύο σύνολα (π.χ. ετικέτες επισυναπτόμενες σε φωτογραφίες).	• Για την εύρεση και δημιουργία συστάσεων ίδιων αντικειμένων • Ως μέρος της ανάλυσης στην πρόβλεψη συνδέσεων (Link Prediction)
Overlap Similarity	Βρίσκει αν ένα σύνολο δεδομένων υπερκαλύπτετε από ένα άλλο.	Είναι ιδανική επιλογή για την εύρεση ιεραρχίας σε δεδομένα και την ανάπτυξη υπερκατηγοριών και υποκατηγοριών.
Pearson Similarity	Βρίσκει την ομοιότητα μεταξύ δύο πραγμάτων.	Η υπολογιζόμενη ομοιότητα μπορεί να χρησιμοποιηθεί στη δημιουργία rating-based recommendations.

3

Σχεδιασμός και ανάπτυξη εφαρμογής

3.1 *Ανάλυση προβλήματος*

Η παρούσα διπλωματική εργασία εστιάζει στην χρήση των αλγορίθμων γράφων, σε ένα κοινωνικό δίκτυο φίλων και επιχειρήσεων, και πως αυτοί αξιοποιούνται στη δημιουργία ενός συστήματος προσωποποιημένων συστάσεων βάσει φωτογραφιών. Για τον λόγο αυτό έχει αναπτυχθεί μια πιλοτική διαδικτυακή εφαρμογή η οποία διαχειρίζεται κοινωνικά δεδομένα και πιο συγκεκριμένα το δημόσιο σύνολο δεδομένων του Yelp, η περιγραφή του οποίου γίνεται στην επόμενη ενότητα. Η βασική λειτουργικότητα της εφαρμογής επιτρέπει στο χρήστη να αναζητά επιχειρήσεις και να προβάλλει κριτικές για τις επιχειρήσεις αυτές, καθώς επίσης του γίνονται και συστάσεις επιχειρήσεων μετά την επιλογή πέντε τυχαίων φωτογραφιών.

Πιο συγκεκριμένα η εφαρμογή μας έχει τις εξής λειτουργίες :

- ***Επιλογή Χρήστη*** : Μπορούμε να επιλέξουμε χρήστη, κάτι που ισοδυναμεί με log in. Με την επιλογή του χρήστη θα εμφανίζεται ο αριθμός των σχολίων που έχει κάνει ο χρήστης σε επιχειρήσεις , ο μέσος όρος των αξιολογήσεων του, πέντε κατηγορίες επιχειρήσεων στις οποίες έχει κάνει κριτική και το σύνολο των φίλων του.
- ***Αναζήτηση Επιχείρησης :*** Αναζητούμε επιχειρήσεις με βάση την κατηγορία της επιχείρησης και ένα ελεύθερο κείμενο. Με την αναζήτηση θα εμφανίζονται οι πιο σχετικές επιχειρήσεις στην κορυφή, με φωτογραφία και την ονομασία τους.

Πατώντας πάνω στη φωτογραφία θα εμφανίζονται τα σχόλια που έχουν γίνει από τους όμοιους "έμπιστους" φίλους του χρήστη οι οποίοι έχουν υψηλό PageRank. Σε κάθε σχόλιο φαίνονται τα αστέρια αξιολόγησης, το κείμενο-σχολιασμός, το όνομα του χρήστη που έγραψε το σχόλιο και η ημερομηνία σχολιασμού. Τα σχόλια είναι ταξινομημένα και χρονολογικά.

- ***Προτάσεις Επιχειρήσεων :*** Μέσω αυτής της επιλογής εμφανίζονται τυχαία στον χρήστη 200 φωτογραφίες, από ένα σύνολο φωτογραφιών που διαθέτουμε, του ζητείται να επιλέξει 5 (οποιεσδήποτε θεωρεί εκείνος ελκυστικές). Μόλις γίνει αυτό, η εφαρμογή χρησιμοποιώντας τους κατάλληλους αλγορίθμους γράφων και εκτελώντας τα κατάλληλα ερωτήματα σε γλώσσα Cypher βρίσκει παρόμοιες φωτογραφίες, με αυτές της αρχικής του επιλογής. Ομαδοποιεί τις όμοιες φωτογραφίες σε κοινότητες και οι επιχειρήσεις των οποίων οι φωτογραφίες ανήκουν στην ίδια κοινότητα με αυτές, παρουσιάζονται ως πρόταση προς τον χρήστη από την εφαρμογή μας.

3.2 *Περιγραφή data set*

Για τους σκοπούς της συγκεκριμένης εργασίας έχει χρησιμοποιηθεί το ανοιχτού κώδικα σύνολο δεδομένων του Yelp. Πρόκειται για ένα υποσύνολο επιχειρήσεων, αξιολογήσεων και δεδομένων χρηστών το οποίο μπορεί να χρησιμοποιηθεί για προσωπικούς, εκπαιδευτικούς και ακαδημαϊκούς σκοπούς. Διατίθεται ως αρχεία JSON. Από τα αρχεία αυτά εμείς στην εργασία μας χρησιμοποιήσαμε τα τέσσερα (***bussines.json, review.json, user.json, photo.json***)

Αρχείο	**Περιγραφή**
Bussines.json	Περιλαμβάνει τα στοιχεία των επιχειρήσεων συμπεριλαμβανομένων την τοποθεσία τους, τις κατηγορίες στις οποίες ανήκει και άλλα χαρακτηριστικά.
Review.json	Περιλαμβάνει κριτικές που έχουν γραφεί για συγκεκριμένες επιχειρήσεις από συγκεκριμένους χρήστες .
User.json	Περιλαμβάνει δεδομένα χρηστών, συμπεριλαμβανομένων και των φίλων κάθε χρήστη και όλα τα μεταδεδομένα που σχετίζονται με το χρήστη.

Photo.json	Περιέχει δεδομένα φωτογραφιών, συμπεριλαμβανομένων της λεζάντας και της ταξινόμησης τους σε μια κατηγορία από τις "food", "drink", "menu", "inside" ή "outside".

Πίνακας 3.1: JSON αρχεία του dataset

Παρακάτω φαίνεται η περιγραφή των πεδίων καθενός από τα αρχεία χρησιμοποιώντας μια τυχαία εγγραφή.

Bussines.json

```
{
    // string, 22 character unique string business id
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",
    // string, the business's name
    "name": "Garaje",
    // string, the full address of the business
    "address": "475 3rd St",
    // string, the city
    "city": "San Francisco",
    // string, 2 character state code, if applicable
    "state": "CA",
    // string, the postal code
    "postal code": "94107",
    // float, latitude
    "latitude": 37.7817529521,
    // float, longitude
    "longitude": -122.39612197,
    // float, star rating, rounded to half-stars
    "stars": 4.5,
    // integer, number of reviews
    "review_count": 1198,
    // integer, 0 or 1 for closed or open, respectively
    "is_open": 1,
    // object, business attributes to values. note: some attribute values might be objects
    "attributes": {
        "RestaurantsTakeOut": true,
        "BusinessParking": {
            "garage": false,
            "street": true,
            "validated": false,
            "lot": false,
            "valet": false
        },
    },
    // an array of strings of business categories
    "categories": [
        "Mexican",
        "Burgers",
        "Gastropubs"
    ],
    // an object of key day to value hours, hours are using a 24hr clock
    "hours": {
        "Monday": "10:00-21:00",
        "Tuesday": "10:00-21:00",
        "Friday": "10:00-21:00",
        "Wednesday": "10:00-21:00",
        "Thursday": "10:00-21:00",
        "Sunday": "11:00-18:00",
        "Saturday": "10:00-21:00"
    }
}
```

Review.json

```
{
    // string, 22 character unique review id
    "review_id": "zdSx_SD6obEhz9VrW9uAWA",
    // string, 22 character unique user id, maps to the user in user.json
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",
    // string, 22 character business id, maps to business in business.json
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",
    // integer, star rating
    "stars": 4,
    // string, date formatted YYYY-MM-DD
    "date": "2016-03-09",
    // string, the review itself
    "text":
           "Great place to hang out after work: the prices are decent, and the ambience is
           fun. It's a bit loud, but very lively. The staff is friendly, and the food is good.
           They have a good selection of drinks.",
    // integer, number of useful votes received
    "useful": 0,
    // integer, number of funny votes received
    "funny": 0,
    // integer, number of cool votes received
    "cool": 0
}
```

User.json

```
{
    // string, 22 character unique user id, maps to the user in user.json
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",
    // string, the user's first name
    "name": "Sebastien",
    // integer, the number of reviews they've written
    "review_count": 56,
    // string, when the user joined Yelp, formatted like YYYY-MM-DD
    "yelping_since": "2011-01-01",
    // array of strings, an array of the user's friend as user_ids
    "friends": [
        "wqoXYLWmpkEH0YvTmHBsJQ",
        "KUXLLiJGrjtSsapmxmpvTA",
        "6e9rJKQC3n0RSKyHLViL-Q"
    ],
    // integer, number of useful votes sent by the user
    "useful": 21,
    // integer, number of funny votes sent by the user
    "funny": 88,
    // integer, number of cool votes sent by the user
    "cool": 15,    // integer, number of fans the user has
    "fans": 1032,
    // array of integers, the years the user was elite
    "elite": [
        2012,
        2013
    ],
    // float, average rating of all reviews
    "average_stars": 4.31,
    // integer, number of hot compliments received by the user
    "compliment_hot": 339,
    // integer, number of more compliments received by the user
    "compliment_more": 668,
    // integer, number of profile compliments received by the user
    "compliment_profile": 42,
    // integer, number of cute compliments received by the user
    "compliment_cute": 62,
    // integer, number of list compliments received by the user
    "compliment_list": 37,
    // integer, number of note compliments received by the user
    "compliment_note": 356,
    // integer, number of plain compliments received by the user
    "compliment_plain": 68,
    // integer, number of cool compliments received by the user
```

```
    "compliment_cool": 91,
    // integer, number of funny compliments received by the user
    "compliment_funny": 99,
    // integer, number of writer compliments received by the user
    "compliment_writer": 95,
    // integer, number of photo compliments received by the user
    "compliment_photos": 50
}
```

Photo.json

```
{
    // string, 22 character unique photo id
    "photo_id": "_nN_DhLXkfwEkwPNxne9hw",
    // string, 22 character business id, maps to business in business.json
    "business_id" : "tnhfDv5Il8EaGSXZGiuQGg",
    // string, the photo caption, if any
    "caption" : "carne asada fries",
    // string, the category the photo belongs to, if any
    "label" : "food"
}
```

Να σημειωθεί πως για τις ανάγκες της εργασίας μας δεν έχουν χρησιμοποιηθεί όλα τα μεταδεδομένα των αρχείων. Επίσης λόγω του ότι τα δεδομένα θα εισαχθούν στη βάση χρησιμοποιώντας το εργαλείο ***neo4j-admin import tool*** (ενότητα 3.4.3.1) , διαδικασία η οποία απαιτεί να είναι τα αρχεία σε CSV μορφή, γράψαμε κώδικα σε γλώσσα Python με τον οποίο μετατρέψαμε τα JSON αρχεία σε CSV, παίρνοντας ταυτόχρονα μόνο τα μεταδεδομένα που χρειαζόμαστε από κάθε JSON αρχείο.

3.3 *Πλατφόρμες και προγραμματιστικά εργαλεία*

Για να είναι κάποιος πετυχημένος και αποτελεσματικός στη βιομηχανία ιστού είναι απαραίτητο να συμβαδίζει με τις σύγχρονες τεχνολογίες. Δεν υπάρχει απόλυτη απάντηση στην ερώτηση ποιο εργαλείο πρέπει να χρησιμοποιηθεί. Αυτό εξαρτάται πάντα από τους στόχους που θέτουμε. Στη συγκεκριμένη εργασία έγινε προσπάθεια να χρησιμοποιηθούν τεχνολογίες state-of-the-art με τις οποίες υπάρχει μια σχετική εξοικείωση.

Η βάση δεδομένων που χρησιμοποιήσαμε είναι η neo4j, η κορυφαία αυτή τη στιγμή βάση δεδομένων γραφημάτων [20]. Για την εξόρυξη πληροφοριών από τη βάση χρησιμοποιούμε τη γλώσσα Cypher, μια γλώσσα ερωτημάτων για τη neo4j. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την επεξεργασία και μετατροπή των αρχείων JSON σε CSV είναι η Python. Εκτός από αυτά χρησιμοποιούνται και τα ακόλουθα εργαλεία.

3.3.1 Node.js

Η Node.js είναι μια πλατφόρμα φτιαγμένη στο JavaScript runtime του Chrome και χρησιμοποιείται για την εύκολη δημιουργία γρήγορων, κλιμακωτών εφαρμογών δικτύων. Βασικό χαρακτηριστικό της είναι ότι αντίθετα με άλλες πλατφόρμες ανάπτυξης διαδικτυακών εφαρμογών που στηρίζονται στην πολυνηματική επεξεργασία, η Node.js χρησιμοποιεί ένα non-blocking I/O μοντέλο βασισμένο σε γεγονότα (events). Αυτό την καθιστά εξαιρετικά αποτελεσματική για real-time εφαρμογές μεγάλης κλίμακας οι οποίες εκτείνονται σε πολλές κατανεμημένες συσκευές. Κάθε λειτουργία στην Node.js εκτελείται ασύγχρονα, δηλαδή εμποδίζει την εφαρμογή από το να σταματάει κάθε φορά που περιμένει την εκτέλεση μιας λειτουργίας εισόδου/εξόδου [28].

3.3.2 NPM

Το npm (*Node Package Manager*) είναι ένα σύστημα διαχείρισης πακέτων το οποίο έρχεται προεγκατεστημένο με την Node.js . Το npm βοηθάει στην εύκολη εγκατάσταση πακέτων και plugins, τόσο για το front-end όσο και για το back-end κομμάτι μιας εφαρμογής, τα οποία έχουν αναπτυχθεί από τις κοινότητες προγραμματιστών της Node.js .Πρόκειται για το μεγαλύτερο μητρώο λογισμικού παγκοσμίως. Προγραμματιστές ανοιχτού κώδικα από κάθε ήπειρο χρησιμοποιούν το npm για να μοιραστούν και να δανειστούν πακέτα. Μπορούμε λοιπόν να βρούμε έτοιμα πακέτα κώδικα και να τα χρησιμοποιήσουμε ή να τα προσαρμόσουμε στην εφαρμογή μας με μεγάλη ευκολία. Επίσης παρέχει το αρχείο package-lock.json το οποίο μας δείχνει σε μορφή δέντρου όλα τα πακέτα με τις εκδόσεις τους που έχουμε εγκαταστήσει στο έργο μας και μπορούμε πολύ εύκολα να κάνουμε ενημερώσεις όταν αυτό απαιτείται [29].

3.3.3 React.js

Η React.js είναι μια από τις πιο δημοφιλείς βιβλιοθήκες ανοιχτού κώδικα της Javascript, για την δημιουργία διεπαφών χρήστη (User Interface - UI). Αποτελεί ένα Javascript εργαλείο το οποίο επιτρέπει την δημιουργία πλούσιων και διαδραστικών front-end εφαρμογών ιστού και κινητών για όσους έχουν γνώσεις προγραμματισμού σε Javascript.

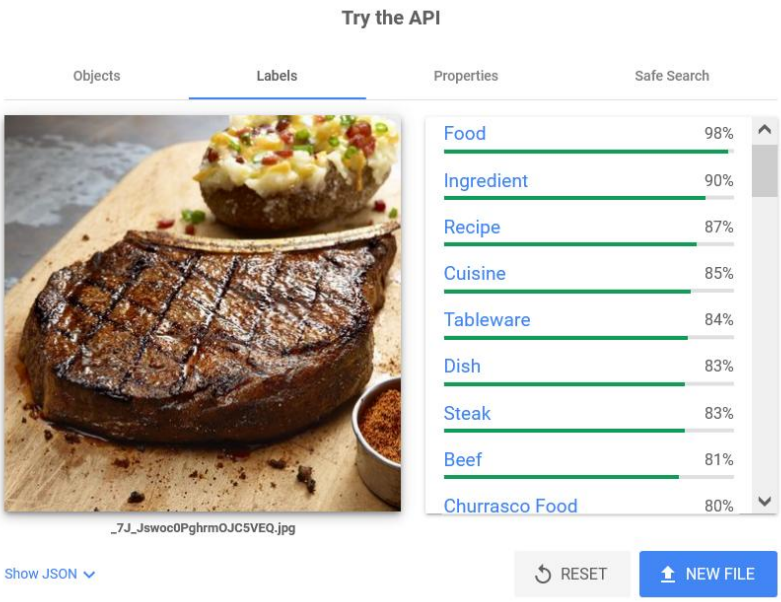
Σε αντίθεση με έναν σύνθετο κώδικα μιας δυναμικής διαδικτυακής εφαρμογής, η react js παρέχει λιγότερη κωδικοποίηση και δίνει περισσότερη λειτουργικότητα, υποστηρίζοντας επίσης την κατασκευή κώδικα αναγνώσιμου από μηχανές.

Επιτρέπει την εύκολη δημιουργία επαναχρησιμοποιήσιμων UI components δίνοντας στους προγραμματιστές τη δυνατότητα να δημιουργούν μεγάλες εφαρμογές ιστού στις οποίες μπορούν να αλλάζουν τα δεδομένα, χωρίς επαναφόρτωση της σελίδας. Αυτό γίνεται γιατί το React.js χρησιμοποιεί ένα εικονικό DOM (Document Object Model – Μοντέλο Αντικειμένου Εγγράφου) και όχι ένα απλό DOM[30][31].

Το DOM είναι ένα αντικείμενο το οποίο δημιουργείται κάθε φορά που μια σελίδα φορτώνεται στον φυλλομετρητή. Όταν γίνονται λοιπόν αλλαγές στα δεδομένα μας στο back-end κομμάτι της εφαρμογής μας, αυτά προκαλούν αλλαγές και δημιουργούν ένα νέο DOM για τη σελίδα που ξανά φορτώνεται στον φυλλομετρητή, κάτι που επιβαρύνει την εφαρμογή μας. Το React.js όμως χρησιμοποιώντας ένα εικονικό DOM, κάνει τις αλλαγές στο εικονικό DOM, το οποίο βρίσκεται εξολοκλήρου στη μνήμη, και μόνο στα components που έχουν αλλαγές, και στη συνέχεια αλλάζει μόνο τα μεμονωμένα στοιχεία DOM αντί να κάνει επαναφόρτωση του πλήρους DOM στον φυλλομετρητή κάθε φορά. Αυτό κάνει την εφαρμογή μας ελαφριά και άμεσα ανταποκρίσιμη [30][31].

3.3.4 Google Vision AI

Το Google Vision AI ενσωματώνει ισχυρά μοντέλα μηχανικής μάθησης σε ένα εύκολο στη χρήση REST API , επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τη δύναμη της μηχανικής μάθησης χωρίς να χρειάζεται να εκπαιδεύσουν μοντέλα δικά τους [32]. Μας δίνει τη δυνατότητα να ανιχνεύουμε σε μια εικόνα αντικείμενα, ετικέτες και πρόσωπα, να εντοπίζουμε αυτόματα τα λογότυπα προϊόντων, να ανιχνεύουμε κείμενο και άλλα.

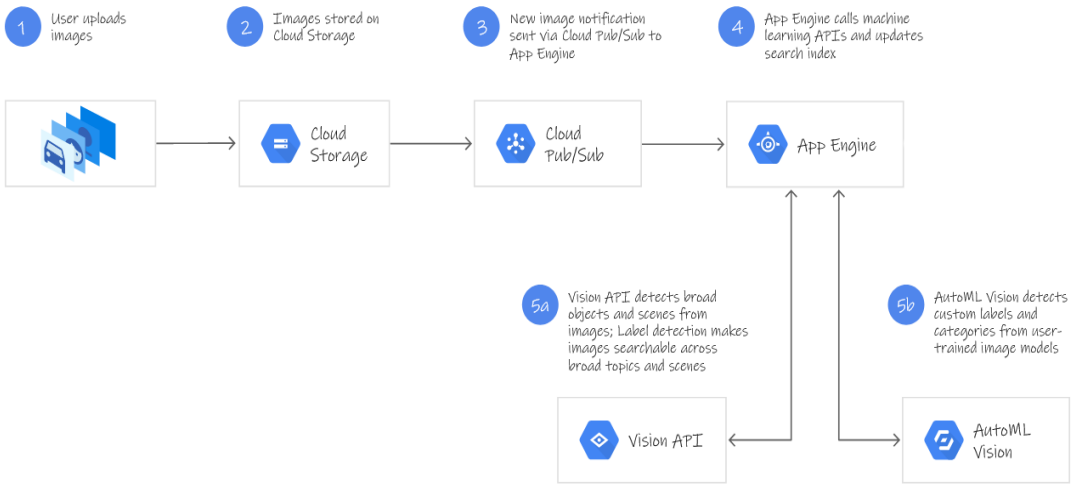


Εικόνα 3.1 : Ανίχνευση ετικετών με τη χρήση του Google Vision AI

Μπορούμε να αναλύσουμε το περιεχόμενο μιας εικόνας από τα μεταδεδομένα που μας επιστρέφει, τα οποία είναι σε μορφή JSON. Στην περίπτωση της ανίχνευσης ετικετών σε μια φωτογραφία το api επιστρέφει :


```
{
          "responses": [
          {
               "labelAnnotations": [

               {
               "description": "Food",
               "mid": "/m/02wbm",
               "score": 0.9829857,
               "topicality": 0.9829857
               },

               { "description": "Ingredient",
               "mid": "/m/07xgrh",
               "score": 0.90268815,
               "topicality": 0.90268815
               }
          ,…
     ]
 }
```


Είναι αρκετά εύκολο στη χρήση και την παραμετροποίηση του, έχει υψηλή ακρίβεια και πολύ καλό χρόνο απόκρισης. Το Google Vision AI στην συγκεκριμένη εργασία χρησιμοποιήθηκε για να γίνει ανίχνευση ετικετών σε φωτογραφίες (Label Detection).



Εικόνα 3.2 : Λειτουργία Google Vision AI (Πηγή[32])

3.4 *Λεπτομέρειες υλοποίησης*

Χρησιμοποιώντας τα παραπάνω τεχνολογικά εργαλεία δημιουργήθηκε μια διαδικτυακή εφαρμογή η οποία αποτελείται από τρία τμήματα :

- την βάση δεδομένων, σε περιβάλλον Neo4j, στην οποία εισάγουμε τα δεδομένα του dataset που χρησιμοποιούμε και το οποίο το αναλύσαμε στην Ενότητα 3.2 .
- το back-end κομμάτι το οποίο είναι υπεύθυνο για την επικοινωνία με τη βάση
- το front-end (client) κομμάτι το οποίο είναι υπεύθυνο για την ανάκτηση των δεδομένων από τη βάση και την παρουσίασή τους στον χρήστη.

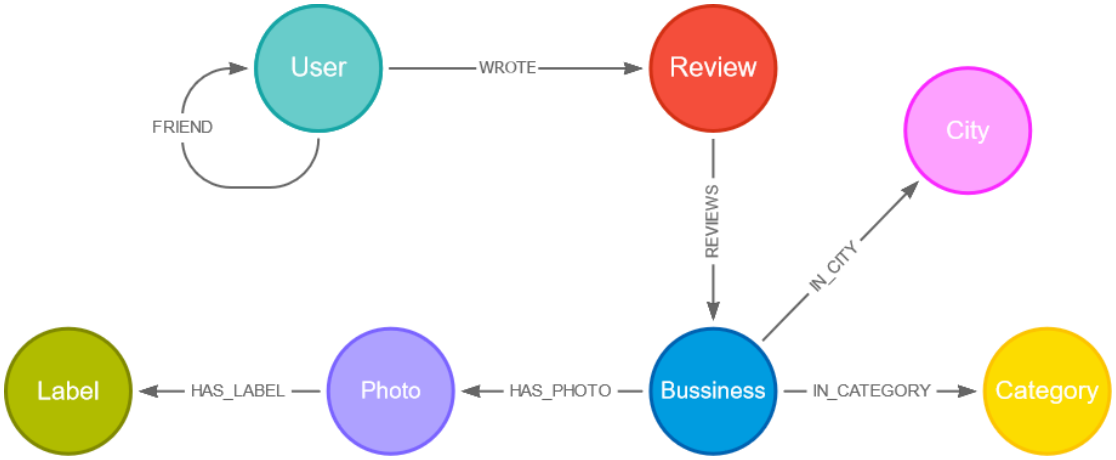
Αρχικά θα πρέπει να δημιουργηθεί η βάση δεδομένων σύμφωνα με το πρότυπο Label Property Graph (Ενότητα 2.3) . Θα πρέπει δηλαδή να προσδιοριστούν ποιες οντότητες είναι απαραίτητο να υπάρχουν στο native property graph, ποιες σχέσεις υπάρχουν μεταξύ αυτών των οντοτήτων και τι ερωτήματα θα εφαρμοστούν ώστε να καλυφθούν οι ανάγκες της εφαρμογής μας όπως αυτές περιγράφηκαν στην ενότητα 3.1. Στη συνέχεια με τη χρήση της JavaScript βιβλιοθήκης React.js δημιουργείται ένα API το οποίο συνδέεται με τη βάση neo4j χρησιμοποιώντας τους neo4j-java script drivers και μέσω Http Request του client προς το API εκτελούνται κατάλληλα ερωτήματα στη βάση ώστε να επιστραφούν στον client τα δεδομένα που ζήτησε. Για το front-end κομμάτι χρησιμοποιείτε και το Material-UI που είναι ένα React UI Framework για την εύκολη και γρήγορη δημιουργία επαναχρησιμοποιήσιμων UI (User Interface) components.

3.4.1 Το Graph Data Model

Για τη συγκεκριμένη εργασία το property graph που υλοποιήθηκε περιέχει τους εξής κόμβους :

1. User : Οι χρήστες
2. Business: Οι επιχειρήσεις
3. Category: Οι κατηγορίες στις οποίες ανήκουν οι επιχειρήσεις
4. City: Οι πόλεις στις οποίες βρίσκονται οι επιχειρήσεις
5. Review: Τα σχόλια που έχουν κάνει οι χρήστες για συγκεκριμένες επιχειρήσεις
6. Photo: Οι φωτογραφίες κάθε επιχείρησης
7. Label: Οι ετικέτες κάθε φωτογραφίας

Το τελικό μοντέλο δεδομένων με όλους τους κόμβους και τις σχέσεις που τους συνδέουν φαίνεται στην ακόλουθη εικόνα:



Εικόνα 3.3 : Label Property Graph

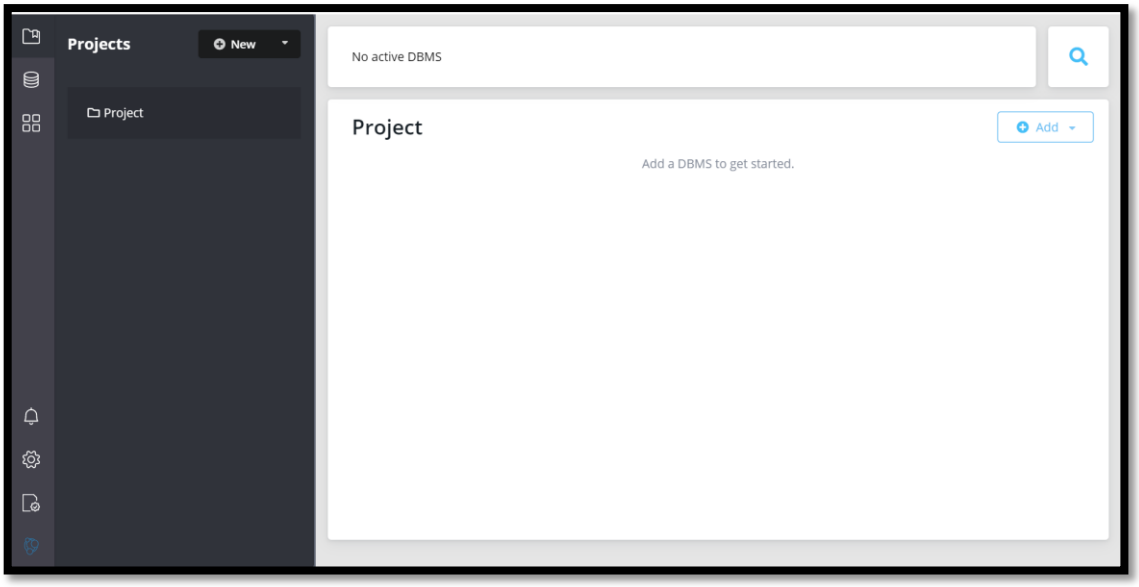
Το γράφημά μας περιέχει κόμβους με ετικέτα (`User`), οι οποίοι έχουν μια (`FRIEND`) σχέση φιλίας με άλλους χρήστες. Οι χρήστες (`User`) επίσης γράφουν (`WROTE`) αναφορές (`REVIEW`) οι οποίες αναφέρονται (`REVIEWS`) σε συγκεκριμένες επιχειρήσεις (`Business`). Κάθε επιχείρηση (`Business`) ανήκει (`IN_CATEGORY`) σε κάποια κατηγορία (`Category`) και βρίσκεται (`IN_CITY`) σε μια πόλη (`City`). Επίσης κάθε επιχείρηση (`Business`) έχει (`HAS_PHOTO`) μια φωτογραφία (`Photo`). Κάθε φωτογραφία (`Photo`) έχει (`HAS_LABEL`) ετικέτες (`Label`) .

Οι κόμβοι και οι σχέσεις που δημιουργούνται είναι οι ακόλουθες :

```
( User ) - [ : FRIEND ] -> ( User )
( User ) - [ : WROTE ] -> ( Review )
( Review ) - [ : REVIEWS ]  -> ( Business )
( Business ) - [ : IN_CATEGORY ] -> ( Category )
( Business ) - [ : IN_CITY ] -> ( City )
( Business ) - [ : HAS_PHOTO ] -> ( Photo )
( Photo ) - [ : HAS_LABEL ] -> ( Label )
```

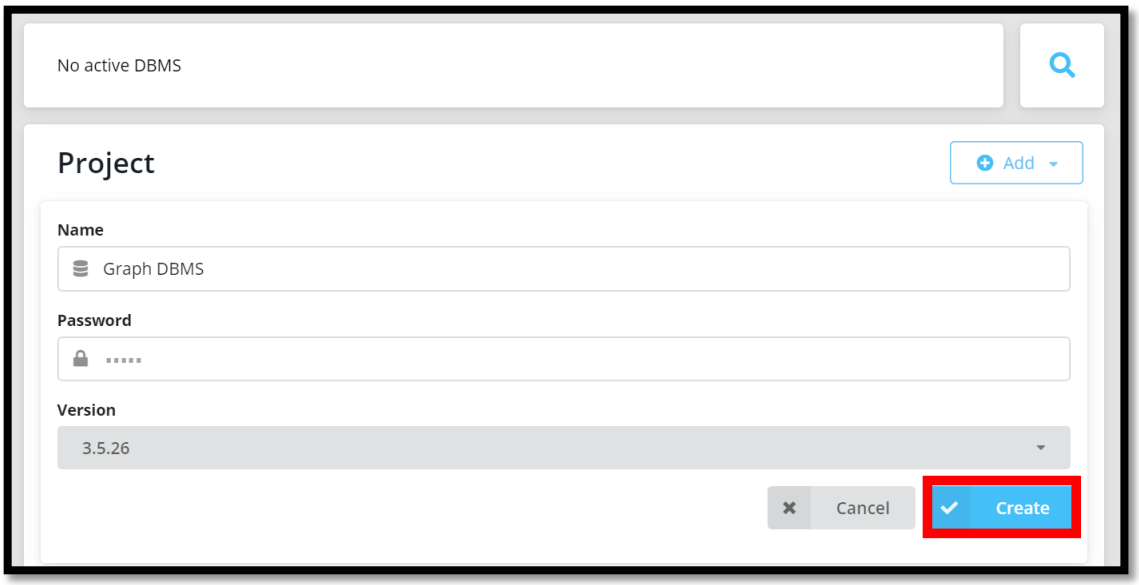
3.4.2 Δημιουργία βάσης στο Neo4j

Για να κατεβάσει κανείς το Neo4j Desktop αρκεί να επισκεφτεί τη σελίδα https://neo4j.com/download/ . Αφού ολοκληρωθεί η εγκατάσταση η πρώτη οθόνη που αντικρίζει ο χρήστης είναι η ακόλουθη:



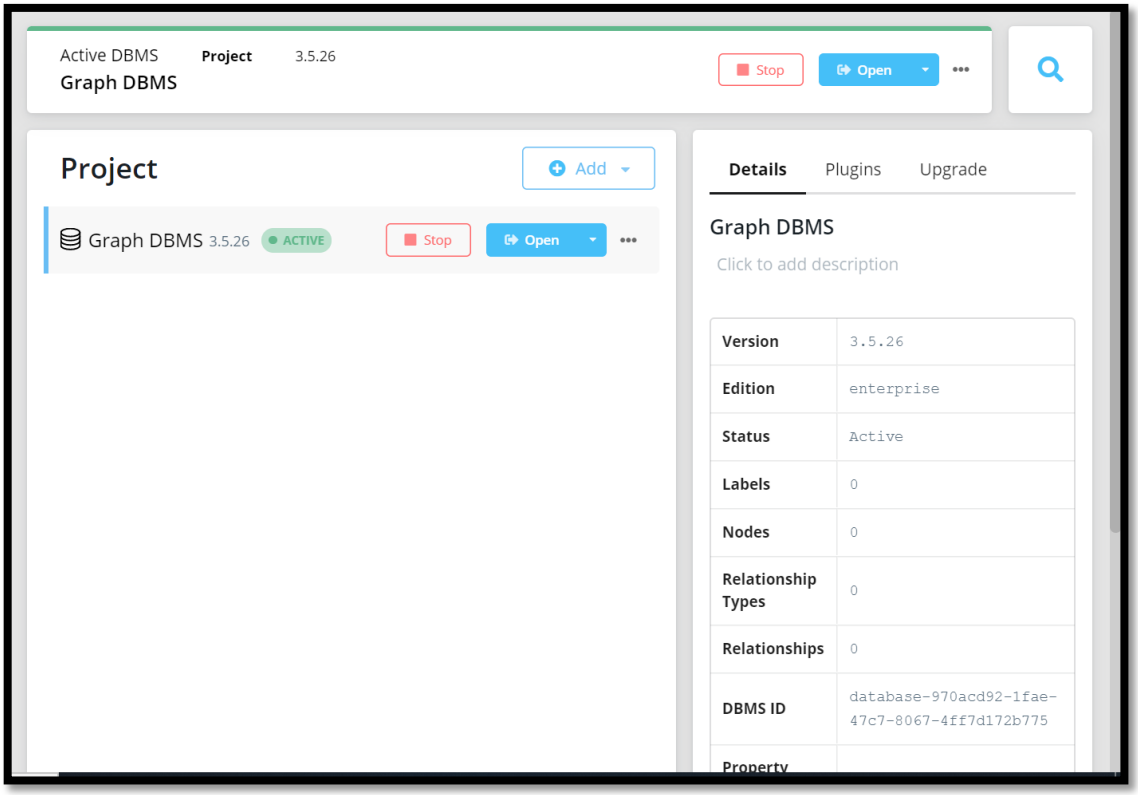
Εικόνα 3.4 : Περιβάλλον Neo4j Desktop

Από την επιλογή **+Add** προσθέτουμε μια τοπική βάση δεδομένων (Local DBMS), δίνοντας της ένα όνομα, κωδικό και επιλέγοντας της έκδοσή της. Στη συγκεκριμένη εργασία έχει επιλεγεί η έκδοση 3.5.26 .



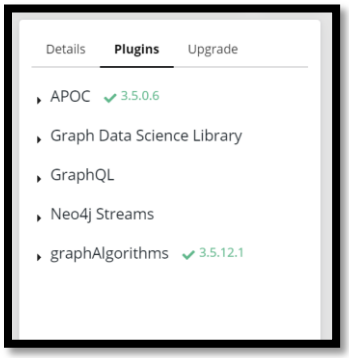
Εικόνα 3.5 : Δημιουργία βάσης στο Neo4j-Desktop

Μόλις ξεκινήσουμε τη βάση τότε θα δούμε την επόμενη οθόνη :



Εικόνα 3.6 : Neo4j Desktop DataBase

Μετά τη δημιουργία της βάσης θα πρέπει να προσθέσουμε και τις κατάλληλες βιβλιοθήκες (Plugins) που θα χρησιμοποιήσουμε. Τα plugins είναι πρόσθετες βιβλιοθήκες που μπορούν να βελτιώσουν τη λειτουργικότητα της βάσης. Για τις ανάγκες της εργασίας ενεργοποιήσαμε την βιβλιοθήκη APOC και graphAlgorithms (algo). Η βιβλιοθήκη algo περιέχει τους αλγόριθμους που αναφέραμε στην ενότητα 2.6, καθώς και άλλους, και θα χρησιμοποιήσουμε στην εργασία μας.



Εικόνα 3.7 : Plugins

3.4.2.1 *Παραμετροποίηση Neo4j*

Το αρχείο παραμετροποίησης μπορούμε να το δούμε από το drop-down μενού δίπλα στην επιλογή Open και επιλέγοντας Settings.

Εικόνα 3.8 : Settings of Neo4j

Εικόνα 3.9 : Edit Settings of Neo4j

Στο αρχείο παραμετροποίησης του Neo4j θα πρέπει να γίνουν κάποιες αλλαγές για τη σωστή λειτουργία της βάσης. Οι αλλαγές αυτές είναι:

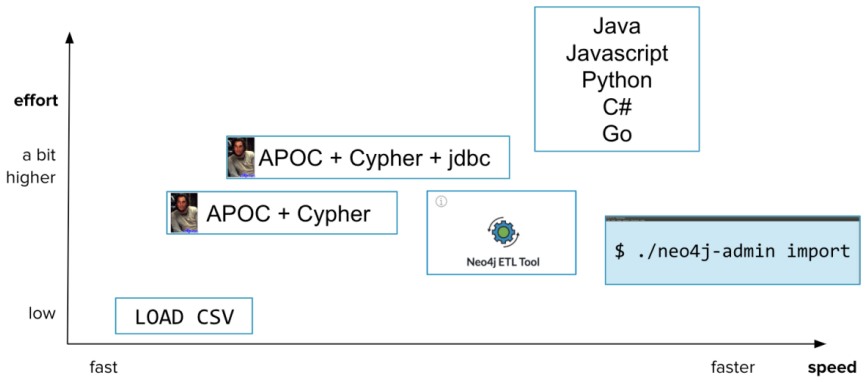
```
dbms.memory.heap.initial_size=3100m
dbms.memory.heap.max_size=3100m
dbms.memory.pagecache.size=1013m

dbms.security.procedures.unrestricted=apoc.*,algo.*
```

Οι τρεις πρώτες γραμμές έχουν να κάνουν με τη μνήμη που έχει στη διάθεση του ο υπολογιστής και πόση από αυτή μπορεί να διαθέσει στο Neo4j για τις λειτουργίες του. Για να έχουμε καλό έλεγχο της συμπεριφοράς του συστήματος, συνιστάται να ορίζονται πάντα οι παράμετροι μεγέθους μνήμης (heap memory και pagecache) στο neo4j.conf. μιας και το neo4j κατά την εκκίνηση του κάνει κάποιους ελέγχους βάσει των διαθέσιμων πόρων του συστήματος. Αυτές οι τιμές είναι οι ιδανικές για το δικό μας σύστημα. Με την τελευταία γραμμή δίνουμε πλήρη πρόσβαση στο neo4j σε όλες τις διαδικασίες και συναρτήσεις των βιβλιοθηκών apoc και algo.

3.4.3 Εισαγωγή δεδομένων στη βάση του Neo4j

Έχουμε αρκετές επιλογές στον τρόπο με τον οποίο θα εισάγουμε τα δεδομένα στο Neo4j. Το ποιόν θα επιλέξουμε εξαρτάται από το πόσα δεδομένα έχουμε, με ποια εργαλεία είμαστε άνετοι στη χρήση τους, καθώς επίσης και πόσο χρόνο έχουμε για την εκτέλεση της εισαγωγής των δεδομένων.

Οι επιλογές που έχουμε για την εισαγωγή δεδομένων στο Neo4j φαίνονται στην ακόλουθη εικόνα:



Εικόνα 3.10 : Επιλογές εισαγωγής δεδομένων στο Neo4j (Πηγή[33])

Στα πλαίσια της εργασίας μας δοκιμάσαμε τους τρεις παρακάτω τρόπους, καταλήγοντας σε έναν από αυτούς.

- **Load JSON :**

  Αφού τοποθετήσουμε όλα τα json αρχεία του Yelp dataset μέσα στον φάκελο Import της εγκατάστασής του Neo4j, μπορούμε να χρησιμοποιήσουμε την εντολή Load και τη βιβλιοθήκη APOC για να εισάγουμε τα δεδομένα. Θα πρέπει βέβαια να έχουμε δημιουργήσει από πριν μια βάση δεδομένων στο Neo4j.

  Ακολουθούν οι εντολές σε Cypher για την εισαγωγή των αρχείων **business.json** και **user.json** :

  Load business.json

```
CALL apoc.periodic.iterate("
CALL apoc.load.json('file:///dataset/user.json')
YIELD value RETURN value
","
MERGE (u:User{id:value.user_id})
SET u += apoc.map.clean(value, ['friends','user_id'],[0])
WITH u,value.friends as friends
UNWIND friends as friend
MERGE (u1:User{id:friend})
MERGE (u)-[:FRIEND]-(u1)
",{batchSize: 100, iterateList: true});
```

  Load user.json

```
CALL apoc.periodic.iterate("
CALL apoc.load.json('file:///dataset/business.json') YIELD value RETURN value
","
MERGE (b:Business{id:value.business_id})
SET b += apoc.map.clean(value,
['attributes','hours','business_id','categories','address','postal_code'],[])
WITH b,value.categories as categories
UNWIND categories as category
MERGE (c:Category{id:category})
MERGE (b)-[:IN_CATEGORY]->(c)
",{batchSize: 10000, iterateList: true});
```

  Ο συγκεκριμένος τρόπος εισαγωγής απορρίφτηκε λόγω του μεγάλου όγκου δεδομένων που είχαμε να εισάγουμε και της μεγάλης καθυστέρησης που παρατηρήθηκε κατά την εκτέλεση των ερωτημάτων. Κάθε ερώτημα χρειαζόταν πάρα πολύ χρόνο για την εκτέλεση του. Καταφέραμε να εισάγουμε μόνο το αρχείο business.json μετά από 8 ώρες και για το αρχείο user.json το ερώτημα μετά από μια

μέρα εκτέλεσης του, προκάλεσε την κατάρρευση του συστήματός μας. Έτσι ο τρόπος αυτός απορρίφθηκε.

- **Load CSV :**

  Για τον τρόπο αυτό θα έπρεπε να έχουμε στη διάθεσή μας τα αρχεία δεδομένων σε μορφή CSV και βέβαια να έχουμε δημιουργήσει από πριν μια βάση δεδομένων στο Neo4j. Γι ‘ αυτό το λόγο γράφτηκε κώδικας σε γλώσσα Python με τον οποίο δημιουργήσαμε τα CSV αρχεία. Και εδώ όμως παρατηρήθηκε πως ο χρόνος εισαγωγής του dataset ήταν μεγάλος και οδηγούσε σε προβλήματα μεγάλης καθυστέρησης ή κατάρρευσης του συστήματός μας. Απορρίψαμε και αυτόν τον τρόπο.

- **Neo4j-admin Import Tool :**

  Το neo4j–admin είναι ένα εργαλείο που μπορούμε να χρησιμοποιήσουμε στην περίπτωση που χρησιμοποιούμε το neo4j-desktop ή το neo4j server για να εισάγουμε δεδομένα στη βάση. Περιέχεται μέσα στον φάκελο bin της εγκατάστασης του Neo4j. Είναι ένα ιδανικό εργαλείο για την εισαγωγή ενός αρχικού μεγάλου συνόλου δεδομένων. Στη συνέχεια, καθώς η εφαρμογή εξελίσσεται, το μοντέλο δεδομένων γραφήματος μπορεί να αλλάξει. Ο χρόνος ολοκλήρωσης της εισαγωγής είναι πάρα πολύ μικρός, σε σχέση με τις άλλες δύο μεθόδους που αναφέραμε. Για το λόγο αυτό ήταν και η μέθοδος που επιλέξαμε για να εισάγουμε το σύνολο δεδομένων Yelp στο Neo4j.

3.4.3.1 Εισαγωγή δεδομένων με τη χρήση του Neo4j-admin Import Tool

Είναι σημαντικό να κατανοήσουμε πως για τη χρήση του συγκεκριμένου εργαλείου θα πρέπει να έχουμε τα CSV αρχεία στη μορφή που απαιτείται. Σε αντίθεση με την περίπτωση της εισαγωγής εκτελώντας ένα ερώτημα Cypher, εδώ δεν έχουμε την ευελιξία να μετατρέψουμε τα δεδομένα κατά την εισαγωγή τους. Πρέπει να είμαστε σίγουροι ότι τα δεδομένα μας είναι έτοιμα. Επίσης η διαδικασία μπορεί να γίνει μόνο μια φορά και η δημιουργία της βάσης γίνεται με τη διαδικασία της εισαγωγής ή στην περίπτωση που προϋπάρχει θα πρέπει να είναι κενή.

Πρέπει λοιπόν να προσέξουμε τα εξής στη μορφή των CSV αρχείων:

- Τόσο στα αρχεία των κόμβων όσο και στα αρχεία των σχέσεων, οι πληροφορίες κεφαλίδας πρέπει να σχετίζονται με τα δεδομένα.

- Οι πληροφορίες κεφαλίδας περιέχουν ένα ID το οποίο προσδιορίζει μοναδικά κάθε εγγραφή, τις προαιρετικές ετικέτες κόμβων ή τους τύπους των σχέσεων και ονόματα για τις ιδιότητες που αντιπροσωπεύουν τα εισαγόμενα δεδομένα.
- Ένα CSV αρχείο μπορεί να έχει μια σειρά κεφαλίδων ή μπορούμε να τοποθετήσουμε τις πληροφορίες κεφαλίδας σε ένα ξεχωριστό αρχείο.

Η μορφή του αρχείου business.csv είναι η εξής:

```
"f9NumwFMBDn751xgFiRbNA","The Range At Lake Norman","10913 Bailey Rd","Cornelius","NC"

"Yzvjg0SayhoZgCljUJRF9Q","Carlos Santo, NMD","8880 E Via Linda, Ste 107","Scottsdale","AZ"

"XNoUzKckATkOD1hP6vghZg","Felinus","3554 Rue Notre-Dame O","Montreal","QC"

"6OAZjbxqM5ol29BuHsil3w","Nevada House of Hose","1015 Sharp Cir","North Las Vegas","NV"
...
```

Το αρχείο business_header.csv που περιέχει τις πληροφορίες κεφαλίδας είναι το εξής:

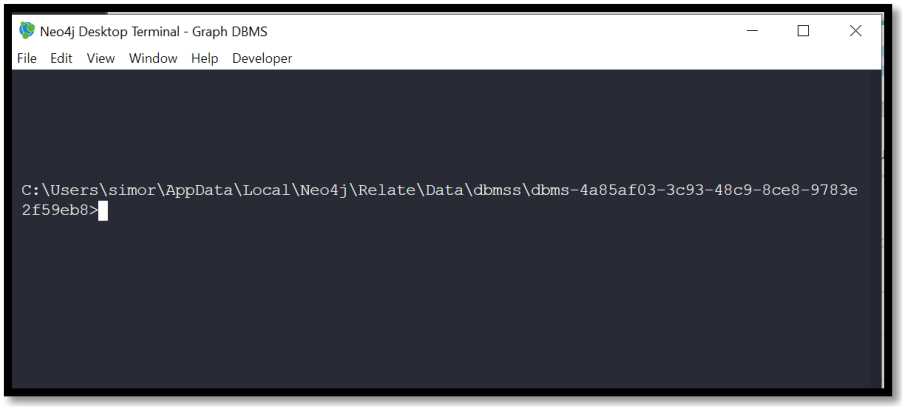
```
id:ID(Business),name,address,city,state
```

Όπως βλέπουμε έχει ένα ID που προσδιορίζει μονοσήμαντα κάθε εγγραφή του αρχείου business.csv και τα name (όνομα της επιχείρησης), address (διεύθυνση της επιχείρησης), city (πόλη στην οποία βρίσκεται η επιχείρηση) και state (η πολιτεία).

Στην περίπτωση των αρχείων κεφαλίδων που δηλώνουν τις σχέσεις μεταξύ των κόμβων, για παράδειγμα το business_HAS_PHOTO_header.csv, έχει την εξής μορφή :

```
:START_ID(Business),:END_ID(Photo)
```

Αφού δημιουργήσουμε όλα τα απαραίτητα CSV αρχεία για το σχηματισμό του property graph που επιθυμούμε, τα εισάγουμε στον φάκελο import της εγκατάστασης του Neo4j. Στη συνέχεια ανοίγουμε το terminal του neo4j.



Εικόνα 3.11 : Terminal Neo4j Desktop

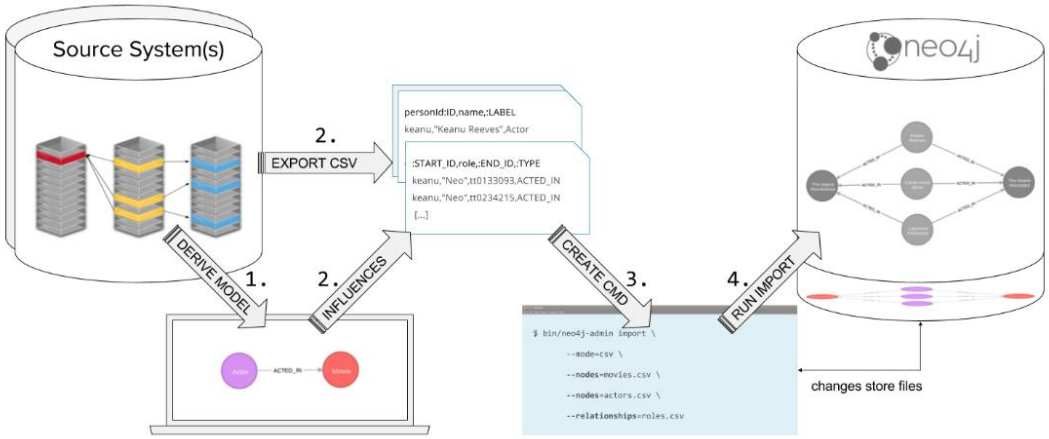
Δίνουμε την εντολή *SET DATA=C:\Users\simor\AppData\Local\Neo4j\Relate\Data\dbmss\dbms-ad390a3a-3ea4-4c0b-b15c-58eab302de2a\import* ώστε να προσδιορίσουμε που βρίσκεται ο φάκελος import, ο οποίος περιέχει όλα τα CSV αρχεία. Στη συνέχεια μετακινούμαστε στο bin directory της εγκατάστασης του Neo4j και δίνουμε την εξής εντολή με την οποία θα δημιουργήσουμε τη βάση yelp.db και θα εισάγουμε σε αυτή τα δεδομένα μας. Η εντολή πρέπει να είναι σε μια γραμμή για να εκτελεστεί.

```
bin\neo4j-admin.bat import
--database=yelp.db
--skip-bad-relationships
--nodes=Business=%DATA%\business_header.csv,%DATA%\business.csv
--nodes=User=%DATA%\user_header.csv,%DATA%\user.csv
--nodes=Review=%DATA%\review_header.csv,%DATA%\review.csv
--nodes=City=%DATA%\city_header.csv,%DATA%\city.csv
--nodes=Category=%DATA%\category_header.csv,%DATA%\category.csv
--nodes=Photo=%DATA%\photo_header.csv,%DATA%\photo.csv
--nodes=Label=%DATA%\label_header.csv,%DATA%\label.csv
--
relationships=FRIENDS=%DATA%\user_FRIENDS_user_header.csv,%DATA%\user_FRIENDS_user.csv
--relationships=WROTE=%DATA%\user_WROTE_review_header.csv,%DATA%\user_WROTE_review.csv
--
relationships=REVIEWS=%DATA%\review_REVIEWS_business_header.csv,%DATA%\review_REVIEWS_
business.csv
--
relationships=IN_CITY=%DATA%\business_IN_CITY_city_header.csv,%DATA%\business_IN_CITY_
city.csv
--
relationships=IN_CATEGORY=%DATA%\business_IN_CATEGORY_category_header.csv,%DATA%\busin
ess_IN_CATEGORY_category.csv
--
relationships=HAS_PHOTO=%DATA%\business_HAS_PHOTO_photo_header.csv,%DATA%\business_HAS
_PHOTO_photo.csv
--
relationships=HAS_LABEL=%DATA%\photo_HAS_LABEL_label_header.csv,%DATA%\photo_HAS_LABEL
_label.csv
--ignore-empty-strings
--multiline-fields=true

Neo4j version: 4.2.2
Importing the contents of these files into
C:\Users\simor\AppData\Local\Neo4j\Relate\Da
ta\dbmss\dbms-b1da6848-63e6-4ec3-9836-beeb20fe4a1d\data\databases\yelp.db:
```

```
...


IMPORT DONE in 13m 58s 66ms.
Imported:
  10401806 nodes
  35702605 relationships
  37555345 properties
Peak memory usage: 1.128GiB
```

Βλέπουμε πως ο χρόνος εισαγωγής ήταν 14 λεπτά περίπου, χρόνος εκπληκτικά μικρός για ένα τόσο μεγάλο σύνολο δεδομένων.

Στη συνέχεια θα πρέπει να ενημερώσουμε το αρχείο neo4j.conf ώστε να περιέχει την εξής γραμμή : *dbms.active_database=yelp.db* . Όταν πατήσουμε εφαρμογή, θα ζητηθεί να κάνουμε επανεκκίνηση της βάσης δεδομένων και μόλις το κάνουμε θα είμαστε έτοιμοι να εξερευνήσουμε το σύνολο δεδομένων yelp.



Εικόνα 3.12 : Λειτουργία neo4j-admin import tool (Πηγή [33])

3.4.4 Σύνδεση της βάσης Neo4j με το React API

Για την επικοινωνία του server με την βάση δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη neo4j-driver η οποία παρέχει μηχανισμούς που επιτρέπουν την ενσωμάτωση Cypher ερωτημάτων μέσα στον JavaScript κώδικα. Δίνοντας σε command line την εντολή `npm install --save neo4j-driver` εγκαθιστούμε τη βιβλιοθήκη.

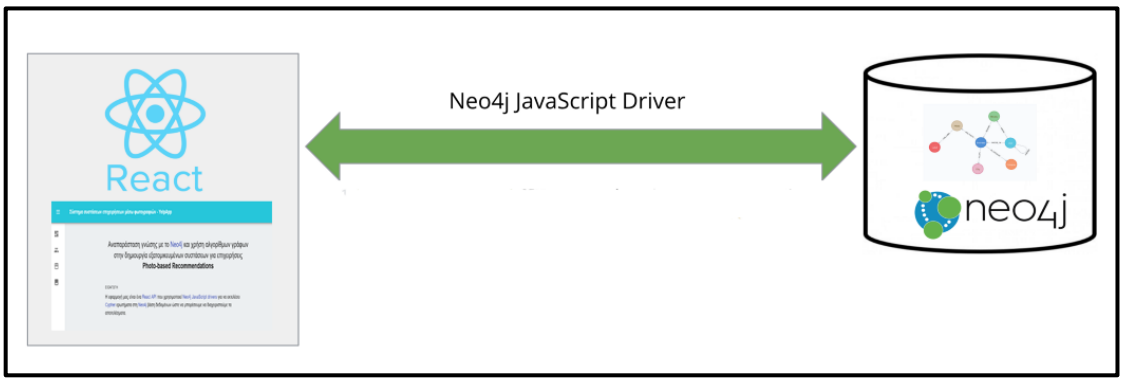
Το αρχείο `.env` είναι ένα αρχείο το οποίο βρίσκεται στη ρίζα του project μας και μέσα σ' αυτό γίνεται ο ορισμός των μόνιμων μεταβλητών του περιβάλλοντος :

```
// Αρχείο .env
REACT_APP_NEO4J_URI= http://localhost:7474/browser/
REACT_APP_NEO4J_USER=neo4j
REACT_APP_NEO4J_PASSWORD=12345
```

Η πρώτη μεταβλητή έχει να κάνει με τη διεύθυνση στην οποία απαντάει η βάση του neo4j και οι άλλες δύο είναι το όνομα χρήστη και ο κωδικός της βάσης.

Παρακάτω φαίνεται ο κώδικας ο οποίος τοποθετείτε σε ένα χωριστό και αυτόνομο αρχείο, με όνομα neo4j.js , και χρησιμοποιείται για τη σύνδεση της εφαρμογής με τη βάση δεδομένων.

```
//Αρχείο neo4j.js
import neo4j from "neo4j-driver/lib/browser/neo4j-web";
export const driver = neo4j.driver(
    process.env.REACT_APP_NEO4J_URI,
    neo4j.auth.basic(
        process.env.REACT_APP_NEO4J_USER,
        process.env.REACT_APP_NEO4J_PASSWORD
    ),
    {encrypted: true}
)
```



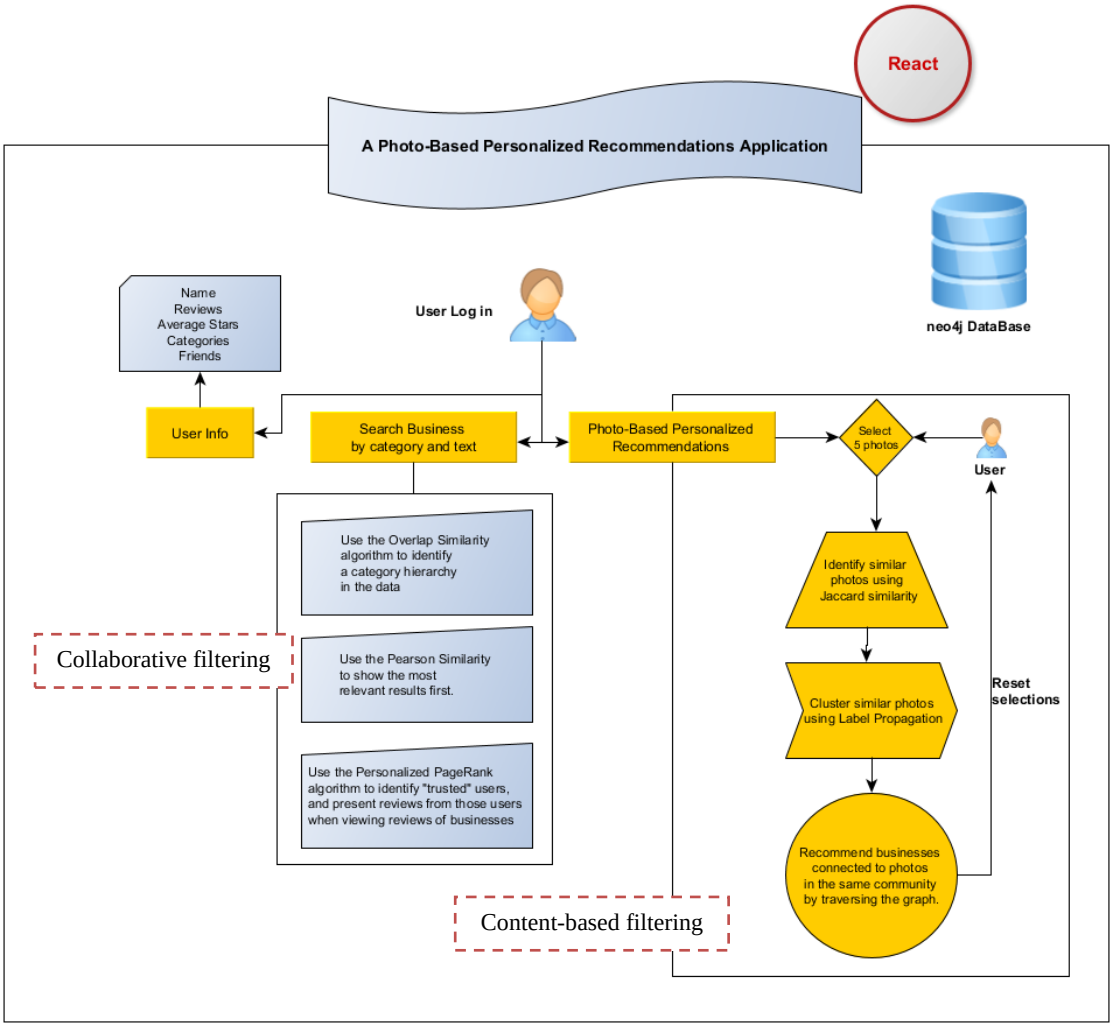
Εικόνα 3.13 : Σύνδεση Neo4j και React μέσω Neo4j JavaScript Driver

4

Γραφική Διεπαφή Εφαρμογής

Στο κεφάλαιο αυτό θα παρουσιάσουμε τη γραφική διεπαφή της εφαρμογή μας. Θα παρουσιάσουμε αναλυτικά τις επιλογές που υπάρχουν στο μενού της εφαρμογής, τη λειτουργικότητά τους καθώς και τα ερωτήματα cypher και τους αλγορίθμους γράφων που μας βοήθησαν στην υλοποίηση των στόχων μας.

Στην Εικόνα 4.1 φαίνεται η αρχιτεκτονική της εφαρμογής μας η οποία χρησιμοποιεί τη Neo4j, τους αλγορίθμους γράφων, το δημόσιο σύνολο δεδομένων του Yelp και τη React, για τη δημιουργία μιας εφαρμογής προσωποποιημένων συστάσεων για επιχειρήσεις βάσει φωτογραφιών.



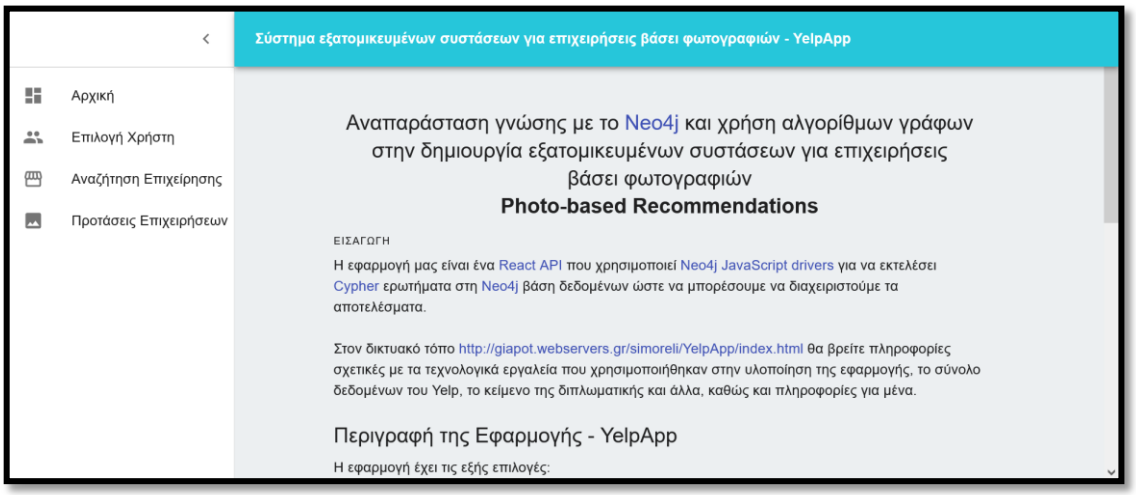
Εικόνα 4.1 : Διάγραμμα εφαρμογής

4.1 *Αρχική σελίδα*

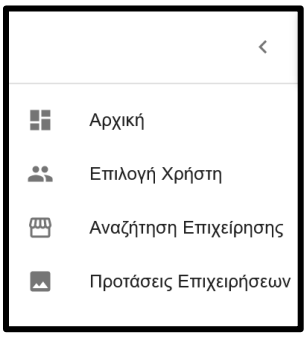
Αφού δώσουμε από `command line` την εντολή `npm start`, στον φάκελο που βρίσκονται τα αρχεία μας, ανοίγει η εφαρμογή μας στον browser στη διεύθυνση http://localhost:3000/ .

Εικόνα 4.2 : Εκκίνηση εφαρμογής

Με την είσοδο μας στο περιβάλλον της εφαρμογής βλέπουμε την οθόνη της Εικόνας 4.3 . Στην κορυφή είναι η κύρια μπάρα της εφαρμογής που φέρει το όνομα της εφαρμογής και το εικονίδιο του μενού το οποίο όταν πατηθεί αναπτύσσει ακριβώς από κάτω και αριστερά της οθόνης το μενού. Στα δεξιά του μενού γίνεται μια γενική περιγραφή της εφαρμογής μας ως εισαγωγή.



Εικόνα 4.3 : Αρχική οθόνη εφαρμογής

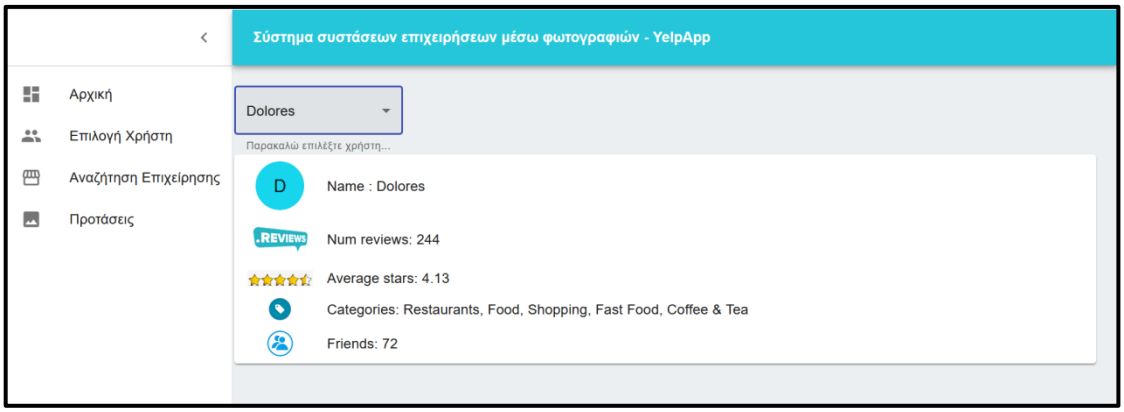


Εικόνα 4.4 : Μενού εφαρμογής

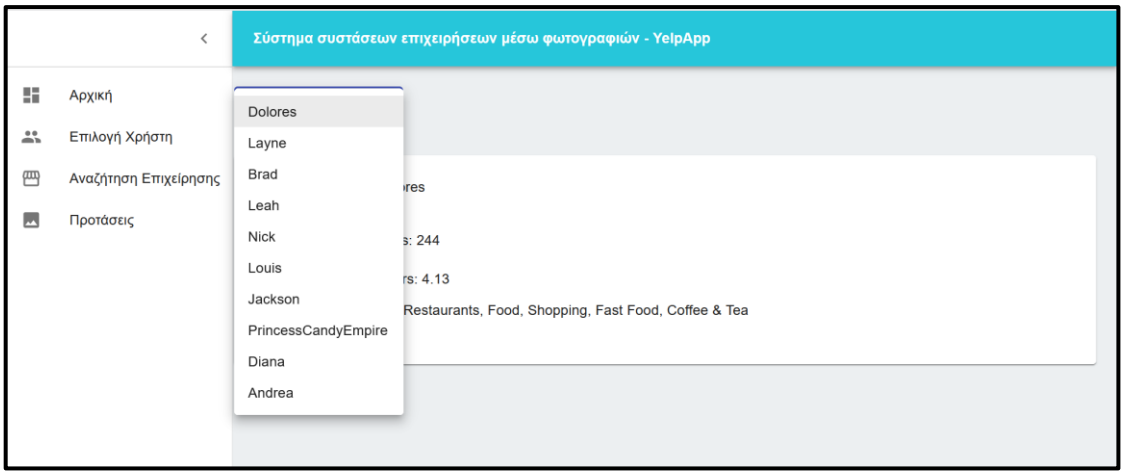
4.2 *Επιλογή Χρήστη*

Όταν από το μενού επιλέξουμε "*Επιλογή Χρήστη*" εμφανίζεται η οθόνη της Εικόνας 4.5 . Από τη drop-down λίστα που υπάρχει (εικόνα 4.6) εμφανίζονται δέκα ονόματα χρηστών που μπορούμε να επιλέξουμε. Η επιλογή αυτή λειτουργεί ουσιαστικά ως είσοδος του χρήστη στην εφαρμογή. Το id του χρήστη που επιλέγουμε θα χρησιμοποιείται στα ερωτήματα cypher όσο

περιηγούμαστε στην εφαρμογή. Επιλέχτηκαν τυχαία μόνο δέκα χρήστες αφού είναι περασμένοι στη βάση δεδομένων μας πάρα πολλοί χρήστες.



Εικόνα 4.5 : Επιλογή Χρήστη



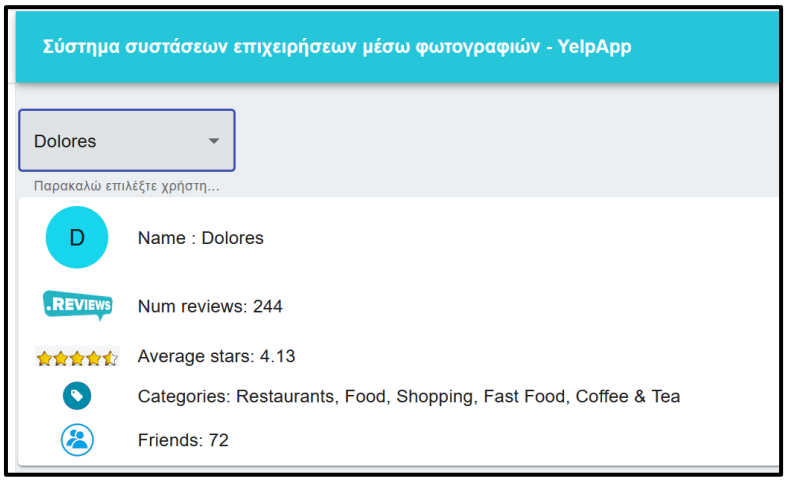
Εικόνα 4.6 : Λίστα επιλογής χρηστών

Με την επιλογή χρήστη εμφανίζονται κάποια στοιχεία που τον αφορούν. Εμφανίζεται το όνομα του, το σύνολο των Reviews που έχει κάνει σε επιχειρήσεις, ο μέσος όρος αστεριών που έχει δώσει , πέντε κατηγορίες επιχειρήσεων που προτιμά και τέλος το σύνολο των φίλων του. Το ερώτημα που εκτελείται με την επιλογή του χρήστη από τη λίστα είναι το ακόλουθο :

```
MATCH (u:User {id: $userId})
MATCH (u:User)-[:WROTE]→(r:Review)
WITH u, avg(r.stars) AS averageStars
ORDER BY averageStars DESC LIMIT 1000
MATCH (u)-[:WROTE]→(:Review)-[:REVIEWS]→(:Business)-[:IN_CATEGORY]-(c:Category)
WITH u, averageStars,c.name as category, count(*) AS count ORDER BY u, count DESC
WHERE count > 5
RETURN u {.*,
          numReviews: toFloat(SIZE((u)-[:WROTE]→(:Review))),
          categories: COLLECT(category)[..5],
          friends: max(apoc.node.degree(u,'FRIENDS')),
          averageStars} AS userInfo
```

Όπου `$userId` είναι η παράμετρος στην οποία αποθηκεύεται ο κωδικός του χρήστη (π.χ. για την `Dolores` είναι `$userId= "7C4B2Skmh4X9f8xJDo9O4w"`).

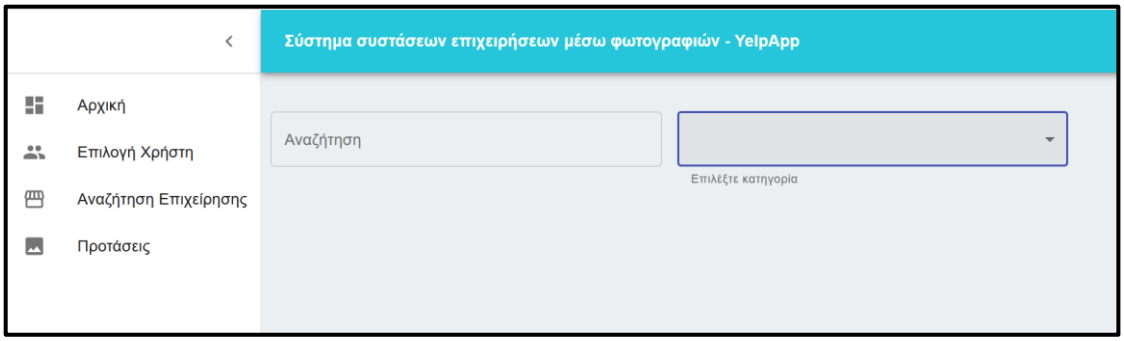
Για το χρήστη `Dolores` βλέπουμε στην Εικόνα 4.7 πως έχει κάνει 244 Reviews, ο μέσος όρος αστεριών που έχει δώσει είναι 4.13 , προτιμά επιχειρήσεις που ανήκουν στις κατηγορίες `Restaurants, Food, Shopping, Fast Food, Coffee & Tea` και έχει 72 φίλους.



Εικόνα 4.7 : Πληροφορίες Χρήστη

4.3 *Αναζήτηση Επιχείρησης*

Στην επιλογή *"Αναζήτηση Επιχείρησης"* μπορούμε να αναζητήσουμε επιχειρήσεις με βάση ένα όνομα και μια κατηγορία.



Εικόνα 4.8 : Αναζήτηση Επιχείρησης

4.3.1 *Ιεράρχηση Κατηγοριών*

Ανοίγοντας την επιλογή της κατηγορίας βλέπουμε πως έχουμε να επιλέξουμε πάνω από 100 κατηγορίες επιχειρήσεων. Ένας τρόπος με τον οποίο μπορούμε να κάνουμε τα πράγματα πιο εύκολα για τον χρήστη είναι να του δώσουμε τη δυνατότητα να πλοηγηθεί σε μια ταξινόμηση των κατηγοριών, ξεκινώντας από το ανώτατο επίπεδο. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε τον αλγόριθμο ομοιότητας επικάλυψης (Overlap Similarity) ώστε να δημιουργήσουμε μια ιεραρχία στις κατηγορίες των επιχειρήσεων.

Ακολουθούν το ερώτημα για τη δημιουργία της ιεραρχίας στις κατηγορίες και το τι επιστρέφει το neo4j :

```
MATCH (category:Category)←[:IN_CATEGORY]-(business)
WITH {item:id(category), categories: collect(id(business))} AS userData
WITH collect(userData) as data
CALL algo.similarity.overlap(data,{ write: true, similarityCutoff: 0.75 })
YIELD nodes, similarityPairs, p50, p75, p90, p99
RETURN nodes, similarityPairs, p50, p75, p90, p99
```

Η παράμετρος `similarityCutoff : 0.75` ορίζει το κατώτατο όριο ομοιότητας στο 75%.

Εικόνα 4.9 : Αποτελέσματα του αλγορίθμου Overlap Similarity

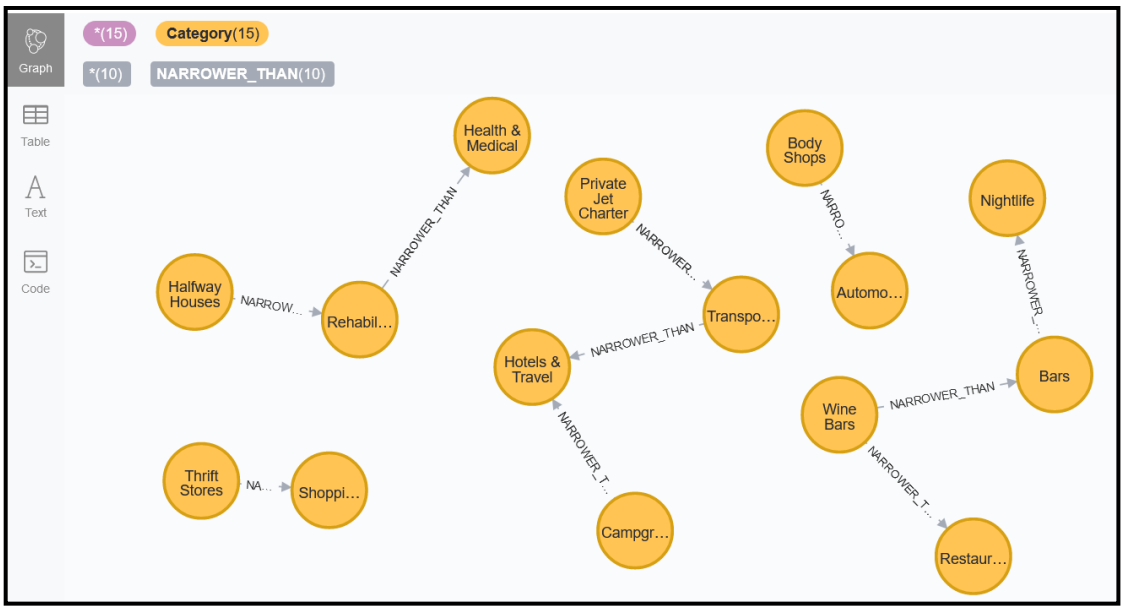
Ο αλγόριθμος `Overlap Similarity` έχει βρει 2.367 παρόμοια ζευγάρια και δημιουργεί τη σχέση `NARROWER_THAN` μεταξύ των κόμβων με ετικέτα `Category`. Θα αφαιρέσουμε τις μεταβατικές σχέσεις με την εκτέλεση του ακόλουθου ερωτήματος ώστε να βρούμε τις κορυφαίες κατηγορίες στην ιεραρχία:

```
MATCH (g1:Category)-[:NARROWER_THAN*2..]→(g3:Category),
(g1)-[d:NARROWER_THAN]→(g3)
DELETE d;
```

Για να δούμε την ιεραρχία που δημιουργήσαμε εκτελούμε το ερώτημα το οποίο μας επιστρέφει 10 σχέσεις :

```
MATCH path = (category:Category)-[:NARROWER_THAN*]→(superCategory:Category)
RETURN path
LIMIT 10
```

Από την Εικόνα 4.10 βλέπουμε πως η κατηγορία `Wine Bars` είναι υποκατηγορία των `Bars` και `Restaurants` και η κατηγορία `Bars` είναι υποκατηγορία της κατηγορίας `Nightlife`.

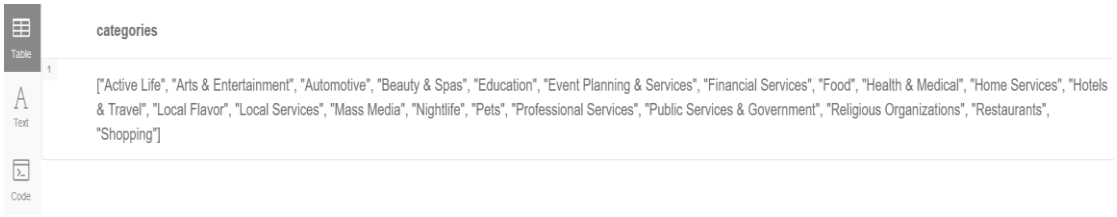


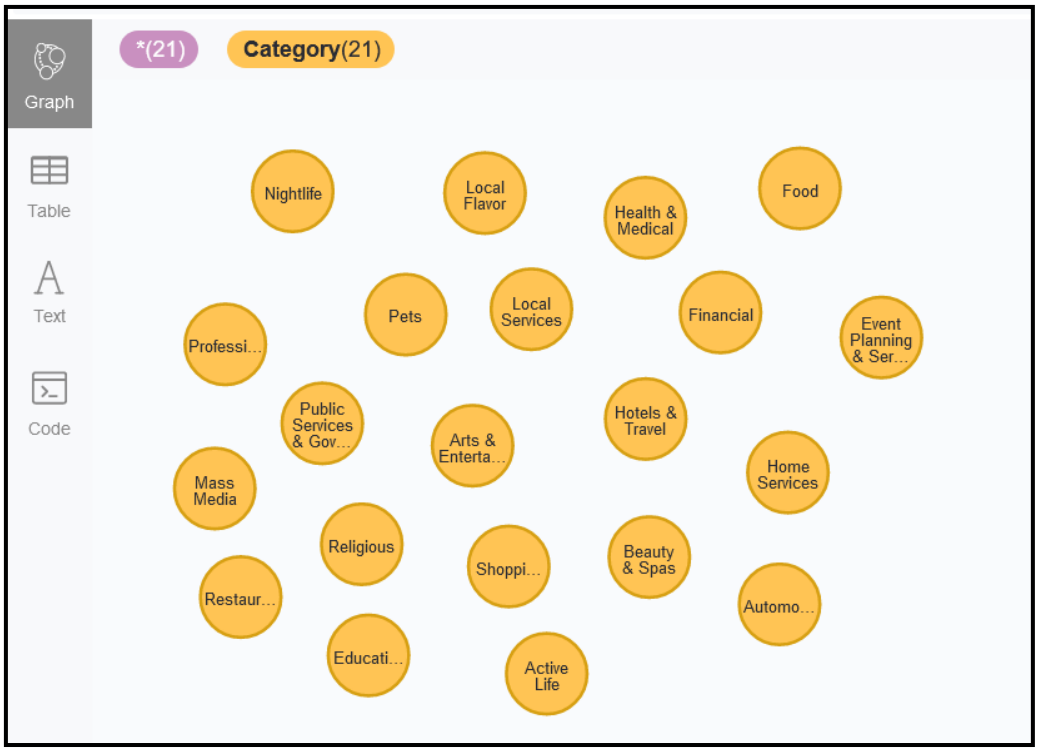
Εικόνα 4.10 : Αποτέλεσμα εκτέλεσης ερωτήματος ιεραρχίας

Μέσα στον κώδικα της javascript η εκτέλεση του ερωτήματος cypher που θα μας εμφανίσει την ιεραρχία των κατηγοριών των επιχειρήσεων γίνεται με την ακόλουθη σύνταξη :

```
export const FETCH_CATEGORIES_QUERY =
`
MATCH (c:Category)
WHERE NOT EXISTS ((c)-[:NARROWER_THAN]->())
WITH c ORDER BY c.name LIMIT 200
RETURN COLLECT(c.name) AS categories
`;
```

Αποτελέσματα ερωτήματος :





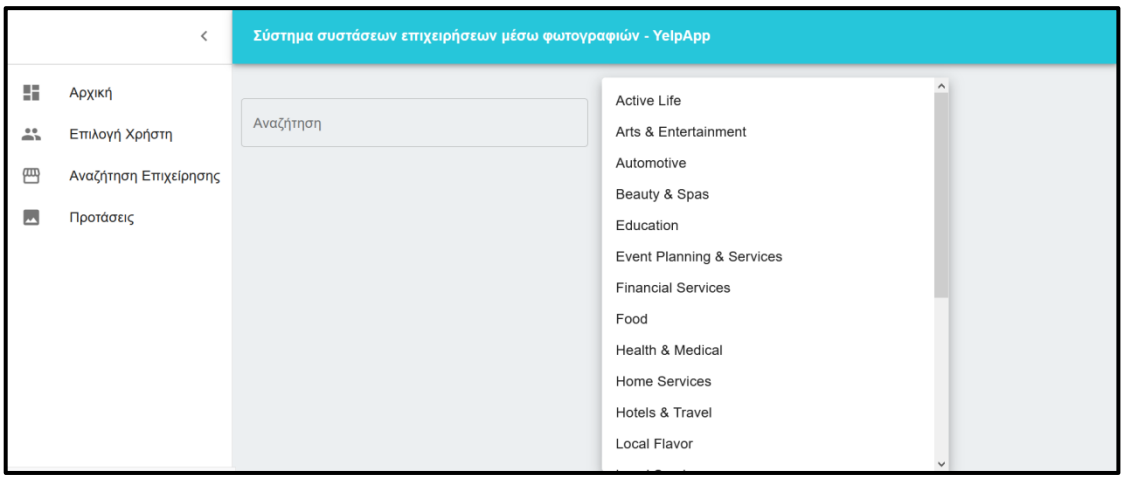
Εικόνα 4.11 : Ιεραρχία Κατηγοριών

Βρίσκουμε με αυτόν τον τρόπο τις κατηγορίες στις οποίες δεν υπάρχει η σχέση `NARROWER_THAN` και είναι οι κορυφαίες κατηγορίες στην ιεραρχία. Τώρα ο χρήστης έχει να επιλέξει ανάμεσα σε 21 κατηγορίες και όχι 100 όπως είχε αρχικά. Στην Εικόνα 4.12 και στον Πίνακα 4.1 βλέπουμε τις κατηγορίες.

Κατηγορίες		
Active Life	Food	Nightlife
Arts & Entertainment	Health & Medical	Pets
Automotive	Home Services	Professional Services
Beauty & Spas	Hotels & Travel	Public Services & Government
Education	Local Flavor	Religious Organizations

Event Planning & Services	Local Services	Restaurants
Financial Services	Mass Media	Shopping

Πίνακας 4.1 : Κορυφαίες κατηγορίες επιχειρήσεων



Εικόνα 4.12 : Επιλογή Κατηγορίας

4.3.2 Ταξινόμηση Αποτελεσμάτων Αναζήτησης

Όταν ένας χρήστης αναζητά επιχειρήσεις θέλουμε να ταξινομούνται τα αποτελέσματα με τέτοιον τρόπο ώστε να δείχνουμε πρώτα στην κορυφή της λίστας τις πιο σχετικές επιχειρήσεις ως προς εκείνον. Θα μπορούσαμε να του επιστρέψουμε ως αποτέλεσμα τις επιχειρήσεις που πληρούν τα κριτήρια της αναζήτησης και έχουν αξιολογήσεις από οποιονδήποτε χρήστη, αλλά θέλουμε να ανακαλύψουμε τους χρήστες που είναι παρόμοιοι και να επιστρέψουμε τις πιο σχετικές επιχειρήσεις στην κορυφή. Αυτό για να το κάνουμε θα χρησιμοποιήσουμε τον αλγόριθμο ομοιότητας Pearson.

Ο αλγόριθμος ομοιότητας Pearson λαμβάνει υπόψη όχι μόνο τη συνολική ομοιότητα μεταξύ των χρηστών, αλλά και τη σχετική ομοιότητά τους. Το ερώτημα παρακάτω παρέχει ένα πρότυπο για τον υπολογισμό της ομοιότητας Pearson των χρηστών με βάση τις επιχειρήσεις στις οποίες έχουν κάνει αξιολογήσεις.

```
CALL algo.similarity.pearson(
"MATCH (u:User) WHERE size((u)-[:WROTE]→()) > 15
 MATCH (u)-[:WROTE]→(review)-[:REVIEWS]→(b)
 RETURN id(u) AS item, id(b) AS category, review.stars as weight",
 {graph: "cypher", topK: 5, skipValue: algo.NaN(), similarityCutoff: 0.1, write:
  true})

```

Το ερώτημα υπολογίζει την ομοιότητα μεταξύ των χρηστών με βάση τις κοινές επιχειρήσεις στις οποίες έχουν κάνει αξιολογήσεις (θα χρειαστεί να χρησιμοποιήσουμε τις σχέσεις `WROTE` και `REVIEWS`). Με την παράμετρο `topK:5` βρίσκουμε τους 5 πιο όμοιους χρήστες με τον χρήστη μας και με την παράμετρο `similarityCutoff: 0.1` ορίζουμε το κατώτατο όριο ομοιότητας. Θέλουμε να χρησιμοποιήσουμε αυτό το γράφημα ομοιότητας για να βελτιώσουμε τις αξιολογήσεις που παρουσιάζουμε στην εφαρμογή μας. Μπορούμε να βρούμε τους παρόμοιους χρήστες ακολουθώντας την σχέση `SIMILARITY` από έναν χρήστη:

```
MATCH (me:User {id: $userId})-[similarity:SIMILAR]→(other)
RETURN other, similarity.score AS similarity
ORDER BY similarity DESC
```

Όπου `$userId` είναι η παράμετρος που κρατάει το `id` του χρήστη.

Όταν λοιπόν ο χρήστης επιλέξει κατηγορία επιχείρησης και γράψει ένα κείμενο στο πεδίο της αναζήτησης, εκτελείται στην JavaScript το ερώτημα με την εξής μορφή:

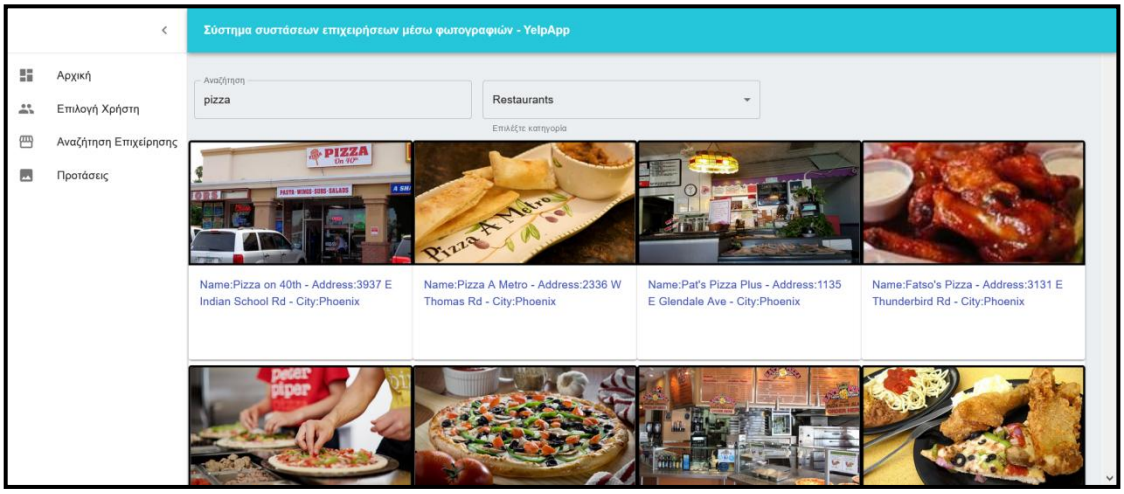
```
1  export const FETCH_BUSINESSES_QUERY =
2  `
3  CALL db.index.fulltext.queryNodes("BusinessNameIndex", $searchText + "~0.75")
4  YIELD node, score
5  WITH node AS b ORDER BY score DESC LIMIT 200
6  MATCH (b)-[:IN_CATEGORY]->(c:Category {name: $category})
7  OPTIONAL MATCH (b)-[:HAS_PHOTO]->(p:Photo)
8  WITH b, COLLECT(p)[0] AS p
9  MATCH (me:User {id: $userId})
10 OPTIONAL MATCH (me)-[similarity:SIMILAR]-(other)-[:WROTE]-(:Review)-[:REVIEWS]->(b)
11 WITH DISTINCT b,p, avg(coalesce(similarity.score,0)) AS num ORDER BY num DESC LIMIT 1000
12 RETURN COLLECT(b {.*, photo: p.id, score: num}) AS businesses
13 `;
14
```

Αφού δημιουργήσουμε έναν `fulltext index` για την ιδιότητα `name` του κόμβου `Business`: `CALL db.index.fulltext.createNodeIndex("BusinessNameIndex",["Business"],["name"])`

στη γραμμή 3 καλούμε αυτόν τον index χρησιμοποιώντας σαν παραμέτρους το `$searchText`, που είναι το κείμενο προς αναζήτηση χωρίς αυτό να είναι case sensitive, και το "~0.75" το οποίο ταιριάζει το κείμενο της αναζήτησης κατά 75% στα αποτελέσματα που θα φέρει πίσω. Δηλαδή αν κάποιος θέλει να φάει `pizza` θα επιλέξει την κατηγορία `Restaurants` και σαν κείμενο αναζήτησης είτε βάλει `Pizza` ή `pizza` ή κατά λάθος βάλει `pizzza` θα του εμφανίσει τα ίδια αποτελέσματα.

Το ερώτημα επιστρέφει τις επιχειρήσεις που ανήκουν στην κατηγορία που επέλεξε ο τρέχον χρήστης, επιθυμούμε να έχουν φωτογραφία (γραμμή 7) και στις οποίες έχουν κάνει αξιολογήσεις παρόμοιοι χρήστες (γραμμή 9-10-11).

Ακολουθεί η Εικόνα 4.13 που μας δείχνει πως εμφανίζονται τα αποτελέσματα στην εφαρμογή μας ψάχνοντας στην κατηγορία `Restaurants` με κείμενο αναζήτησης `pizza`.



Εικόνα 4.13 : Αποτελέσματα αναζήτησης επιχείρησης

4.3.3 Εύρεση σχετικών αξιολογήσεων

Αφού μας εμφανιστούν οι επιχειρήσεις με εικόνα, όνομα, διεύθυνση και πόλη, όπως βλέπουμε στην Εικόνα 4.13, κάνοντας κλικ πάνω στην εικόνα εμφανίζονται σε νέο παράθυρο οι αξιολογήσεις που έχουν γίνει για την συγκεκριμένη επιχείρηση.

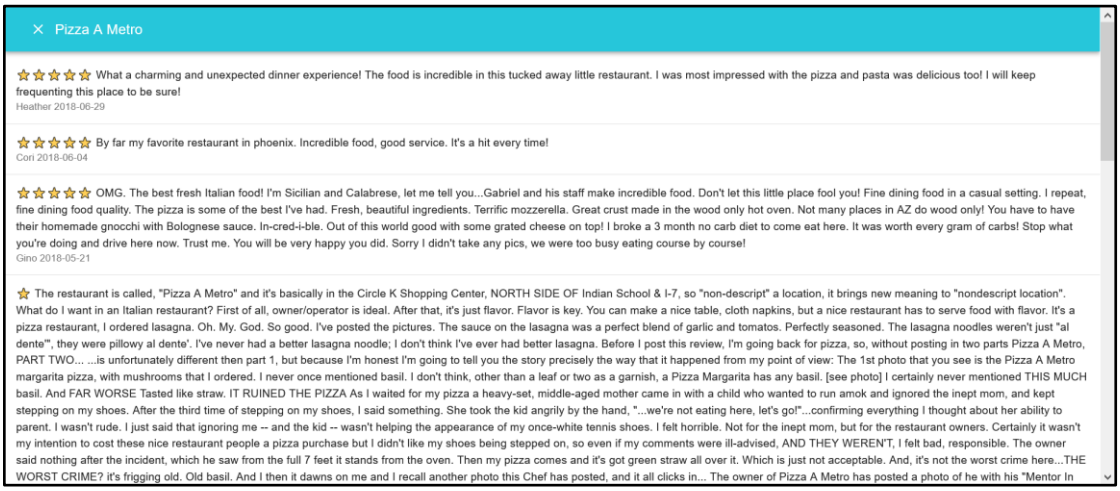
Επειδή οι χρήστες οι οποίοι έχουν γράψει κριτικές για τις επιχειρήσεις είναι πολλοί, θα χρησιμοποιήσουμε τον αλγόριθμο `PageRank` για να προσδιορίσουμε τους "έμπιστους" χρήστες και να παρουσιάσουμε τις δικές τους κριτικές κατά την προβολή των αξιολογήσεων για τις επιχειρήσεις.

Θα τρέξουμε τον αλγόριθμο `PageRank` στον γράφο `review-graph` που θα δημιουργηθεί από χρήστες οι οποίοι έχουν γράψει αξιολογήσεις για τις ίδιες επιχειρήσεις με τον χρήστη μας, καθορίζοντας τους κόμβους προέλευσης `sourceNodes`. Από την εκτέλεση του αλγορίθμου επιστρέφουμε το nodeId και το score κάθε κόμβου και φιλτράρουμε το score ώστε να είναι μεγαλύτερο από 0.15 (score > 0.15) κρατώντας τους 50 πρώτους χρήστες και δημιουργώντας τη σχέση TRUSTS με τον χρήστη μας. Ακολουθεί το ερώτημα :

```
// Compute source nodes
MATCH (u:User {id: $userId})-[:WROTE]→()-[:REVIEWS]→()←[:REVIEWS]-()←[:WROTE]-(other)
WITH u, other, count(*) AS count
WHERE count > 1
WITH u, collect(other) AS sourceNodes

// Execute the PageRank algorithm
CALL algo.pageRank.stream(null, null, {
  iterations:5, direction: "BOTH",
  graph: "review-graph", sourceNodes: sourceNodes
})

// Only keep users that have a PageRank score bigger than the default
YIELD nodeId, score
WITH u, algo.getNodeById(nodeId) AS node, score
WHERE score > 0.15 AND node <> u

// Keep up to 50 users
WITH u, node, score
ORDER BY score DESC
LIMIT 50

// Create a relationship between our user (u) and the influential users (node)
MERGE (u)-[trust:TRUSTS]→(node)
SET trust.score = score
```

Από τη στιγμή που έχουμε βρει ποιοι είναι οι «έμπιστοι» χρήστες του τρέχοντα χρήστη μέσω της σχέσης `TRUSTS`, το ερώτημα που θα εκτελέσουμε ώστε να εμφανίσουμε στην εφαρμογή μας τις πιο σχετικές κριτικές πρώτες, όπως φαίνονται στην Εικόνα 4.14, είναι το εξής :

```
export const FETCH_REVIEWS_QUERY =
`
MATCH (b:Business {id: $businessId}), (me:User {id: $userId})
MATCH (b)<-[:REVIEWS]-(r:Review)<-[:WROTE]-(u:User)
OPTIONAL MATCH (u)<-[t:TRUSTS]-(me)
WITH r {.text, stars: toFloat(r.stars), name: u.name, date: toString(r.date)} AS review
ORDER BY coalesce(t.score, 0.0) DESC, r.date DESC LIMIT 1000
RETURN COLLECT(review) AS reviews
`
```



Εικόνα 4.14 : Αξιολογήσεις επιχείρησης

Εκτός του ότι εμφανίζονται οι πιο σχετικές κριτικές πρώτες, είναι ταξινομημένες και χρονολογικά.

4.4 *Συστάσεις Επιχειρήσεων βάσει φωτογραφιών*

4.4.1 Εισαγωγή

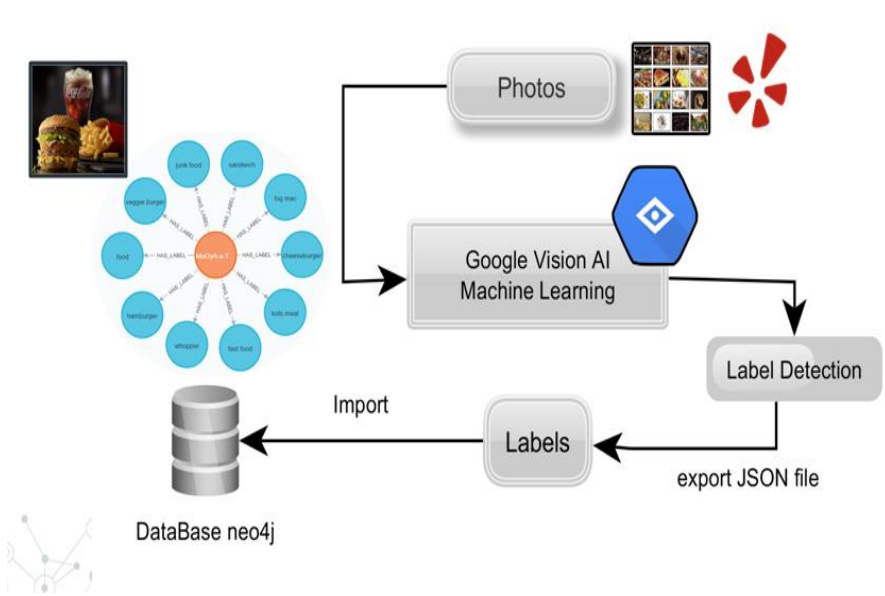
Όπως αναφέρθηκε και στην Ενότητα 3.1 η εφαρμογή μας από την επιλογή "Προτάσεις Επιχειρήσεων" εμφανίζει τυχαία 200 φωτογραφίες στον χρήστη, επιτρέποντάς του να επιλέξει φωτογραφίες που του αρέσουν. Με τη χρήση των αλγορίθμων Jaccard Similarity και Label Propagation βρίσκουμε παρόμοιες φωτογραφίες σε αυτές που άρεσαν στον χρήστη, και οι επιχειρήσεις που συνδέονται με αυτές τις φωτογραφίες συστήνονται στον χρήστη για να τις επιλέξει. Αποτελούν δηλαδή μια προσωποποιημένη σύσταση προς τον χρήστη της εφαρμογής.

Συγκεκριμένα, τα βήματα που ακολουθήθηκαν είναι :

- Προσδιορισμός παρόμοιων φωτογραφιών με τη χρήση του αλγορίθμου ομοιότητας Jaccard.
- Ομαδοποίηση παρόμοιων φωτογραφιών με τη χρήση του αλγορίθμου Label Propagation.
- Πρόταση επιχειρήσεων που είναι συνδεδεμένες με όμοιες φωτογραφίες της ίδιας κοινότητας, διασχίζοντας το γράφημα.

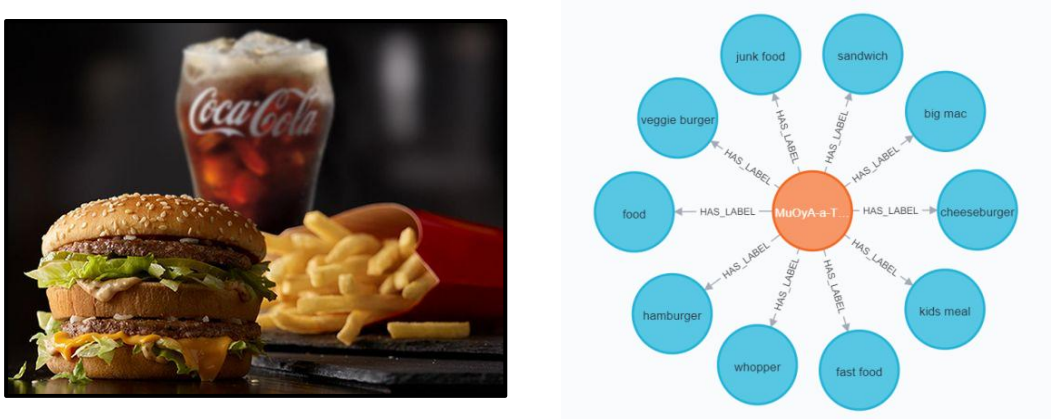
4.4.2 Προσδιορισμός παρόμοιων φωτογραφιών χρησιμοποιώντας την ομοιότητα Jaccard

Το Yelp σύνολο δεδομένων που χρησιμοποιούμαι στην παρούσα εργασία έχει ελάχιστα μεταδεδομένα για κάθε φωτογραφία. Εμείς θέλοντας να προσδιορίσουμε ετικέτες για κάθε μια φωτογραφία που διαθέτουμε, χρησιμοποιήσαμε το Google Vision AI το οποίο χρησιμοποιεί ισχυρά μοντέλα μηχανικής μάθησης για την ανίχνευση ετικετών σε φωτογραφίες (Label Detection). Χρησιμοποιώντας λοιπόν το Google Vision AI σε συνδυασμό με ένα script που γράψαμε σε γλώσσα Python, επιστρέφουμε σε JSON μορφή, για κάθε φωτογραφία, τις ετικέτες που εντοπίστηκαν δημιουργώντας έτσι τους κόμβους με ετικέτα `Label` και τη σχέση `(Photo)-[:HAS_LABEL]->(Label)` στο γράφημα μας .



Εικόνα 4.15 : Αναγνώριση ετικετών σε φωτογραφίες με το Google Vision AI

Κάθε `Photo` στο γράφημα έχει `Labels` όπως φαίνεται στην Εικόνα 4.16.

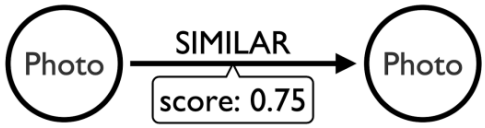


Εικόνα 4.16 : Προσθήκη ετικετών φωτογραφιών στον γράφο

Χρησιμοποιώντας τον αλγόριθμο ομοιότητας Jaccard υπολογίζουμε πόσο όμοιο είναι ένα ζευγάρι φωτογραφιών. Ο αλγόριθμος Jaccard μετρά την ομοιότητα μεταξύ συνόλων (στην περίπτωσή μας ετικέτες που επισυνάπτονται σε φωτογραφίες). Πρόκειται για έναν αλγόριθμο σύγκρισης συνόλων και υπολογίζεται διαιρώντας το μέγεθος της τομής των δύο συνόλων με το μέγεθος της ένωσής τους όπως προαναφέραμε στην Ενότητα 2.6.3. Εδώ τον χρησιμοποιούμε σε ένα ερώτημα της cypher, υπολογίζοντας την ομοιότητα όλων των ζευγών φωτογραφιών.

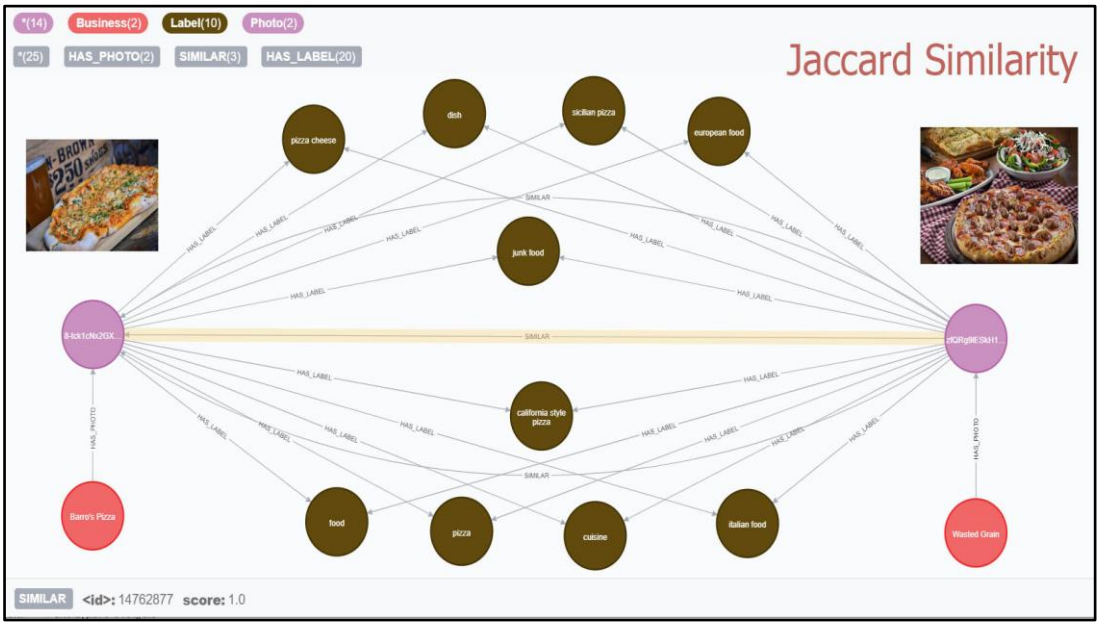
```
MATCH (p:Photo)-[:HAS_LABEL]→(label)
WITH {item:id(p), categories: collect(id(label))} as userData
WITH collect(userData) as data
CALL algo.similarity.jaccard(data, {topK: 3, similarityCutoff: 0.9, write: true})
YIELD p25, p50, p90, p99, p999, p100,  write
RETURN p25, p50, p90, p99, p999, p100, write
```

Χρησιμοποιώντας την παράμετρο `similarityCutoff` προσπαθούμε να αποκλείσουμε όλα τα ζευγάρια που δεν παρουσιάζουν κάποια ομοιότητα. Κρατάμε μόνο εκείνα τα ζευγάρια που έχουν `similarity` τουλάχιστον 0.9 . Επίσης με την παράμετρο `topK: 3` βρίσκουμε τις 3 πιο όμοιες φωτογραφίες. Το παραπάνω ερώτημα δημιουργεί μια σχέση `SIMILAR` με ιδιότητα ένα `score`.



Εικόνα 4.17 : Δημιουργία σχέσης SIMILAR ανάμεσα σε φωτογραφίες

Ας ρίξουμε μια ματιά σε δύο φωτογραφίες με επικαλυπτόμενες ετικέτες και να δούμε πώς υπολογίζεται το `score` στον αλγόριθμο Jaccard Similarity. Στην Εικόνα 4.18, οι δύο φωτογραφίες έχουν 10 επικαλυπτόμενες ετικέτες. Γι' αυτό και το `score` της ομοιότητας Jaccard είναι 10/10 ή 1.0 .



Εικόνα 4.18 : Jaccard Similarity

4.4.3 Ομαδοποίηση παρόμοιων φωτογραφιών με τη χρήση Label Propagation

Έχουμε βρει λοιπόν ποιες φωτογραφίες είναι όμοιες μεταξύ τους, κάτι που προσδιορίζεται από τη σχέση `SIMILAR` που δημιουργήθηκε εφαρμόζοντας το αλγόριθμο Jaccard Similarity στην προηγούμενη ενότητα. Το ερώτημα με το οποίο μπορούμε να δούμε τις φωτογραφίες που συνδέονται με τη σχέση `SIMILAR` είναι :

```
MATCH path = (p1:Photo)-[r:SIMILAR]→(p2:Photo)
RETURN path
LIMIT 20
```

Τώρα θα ομαδοποιήσουμε τις φωτογραφίες βασιζόμενοι στη σχέση `SIMILAR`. Για να το κάνουμε αυτό θα τρέξουμε τον αλγόριθμο Label Propagation στον υπογράφο που σχηματίζεται από τη σχέση `SIMILAR`, ώστε να φτιάξουμε κοινότητες φωτογραφιών.

```
$ CALL algo.beta.labelPropagation('Photo','SIMILAR', {direction: "BOTH"})
```

Η εκτέλεση αυτής της διαδικασίας θα προσθέσει την ιδιότητα `partition` σε κάθε φωτογραφία, η τιμή της οποίας καθορίζει σε ποια κοινότητα φωτογραφιών ανήκει.

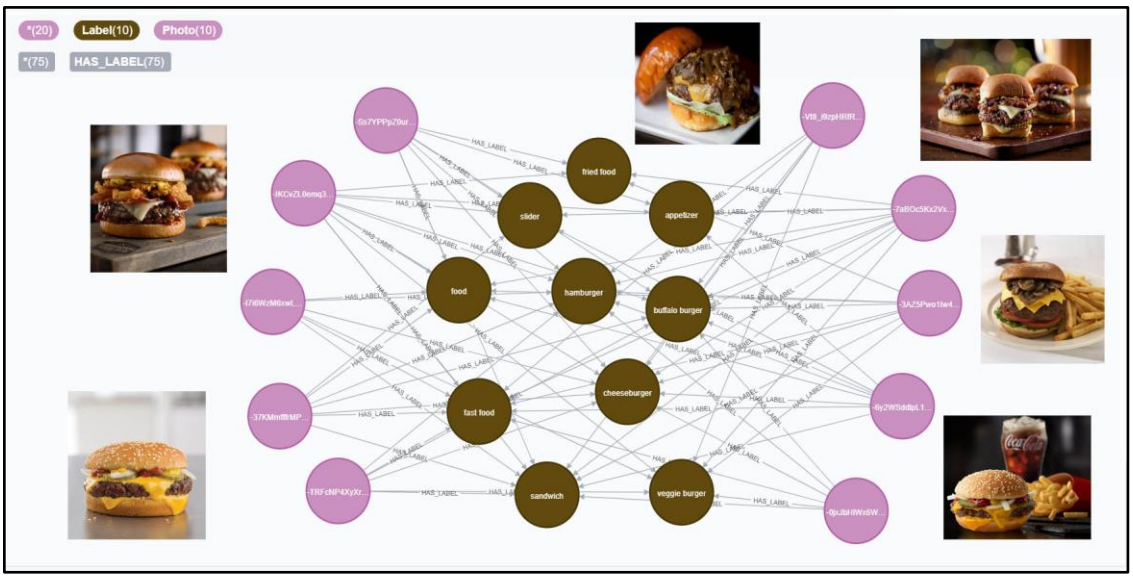
Ας δούμε πόσες φωτογραφίες ανήκουν σε κάθε κοινότητα με το ερώτημα που ακολουθεί:

```
MATCH (p:Photo)
RETURN p.partition AS community, count(*) AS count
ORDER BY count DESC
LIMIT 10
```

	community	count
1	758	109
2	478	104
3	11617	79
4	819	77
5	5546	71
6	9744	71
7	11315	71

Εικόνα 4.19 : Αριθμός φωτογραφιών σε κάθε κοινότητα

Αν επιλέξουμε ένα `community` μπορούμε να δούμε και να επικυρώσουμε κατά πόσο οι φωτογραφίες οι οποίες ανήκουν σε αυτό είναι σωστές .

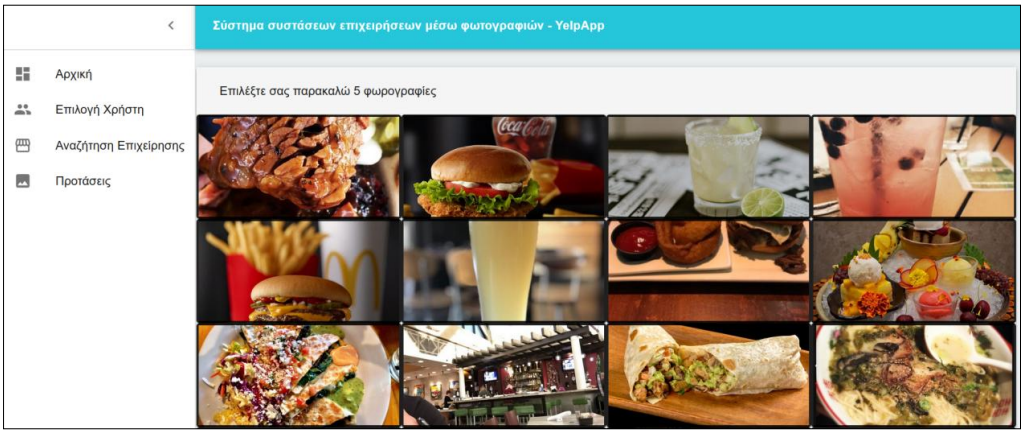


Εικόνα 4.20 : Εμφάνιση community

Παρατηρώντας λοιπόν την Εικόνα 4.20 βλέπουμε πως όλες είναι εικόνες από hamburger, οπότε προφανώς είμαστε σε ένα "hamburger" cluster. Επομένως ο αλγόριθμος δουλεύει σωστά.

4.4.4 Πρόταση επιχειρήσεων βάσει φωτογραφιών

Στην επιλογή "Προτάσεις Επιχειρήσεων" της εφαρμογής μας, όπως έχει ήδη αναφερθεί, εμφανίζονται στον χρήστη τυχαία 200 φωτογραφίες οι οποίες συνδέονται με κάποια επιχείρηση σε μορφή πλέγματος όπως φαίνεται στην Εικόνα 4.21.



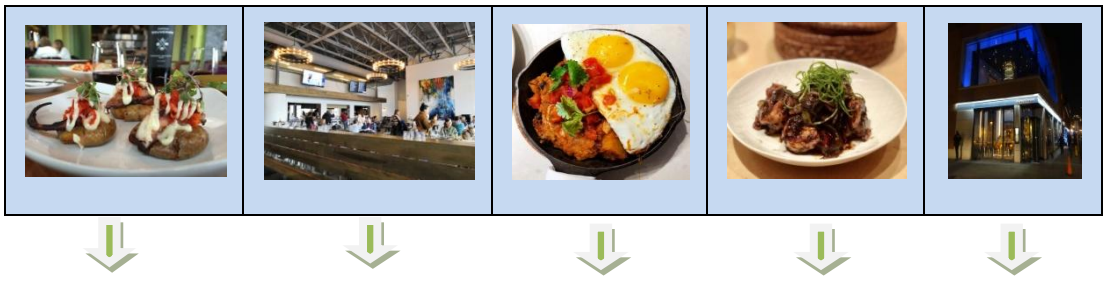
Εικόνα 4.21 : Εμφάνιση φωτογραφιών για επιλογή

Το ερώτημα `cypher` που εκτελείται από τη `JavaScript` για την τυχαία εμφάνιση φωτογραφιών είναι το ακόλουθο :

```
export const FETCH_RECOMMENDED_PHOTOS_QUERY =
`
MATCH (b:Business)-[:HAS_PHOTO]->(p:Photo)
WITH b, p SKIP toInteger(rand() * 30000)-1000 LIMIT 10000
WITH COLLECT(p{photoId: p.id, businessId: b.id}) AS photos
RETURN apoc.coll.shuffle(photos)[..199] AS photos
`;
```

Τώρα που έχουμε ομαδοποιήσει τις φωτογραφίες σε ομάδες χρησιμοποιώντας τον αλγόριθμο Label Propagation, είμαστε έτοιμοι να δημιουργήσουμε τις εξατομικευμένες συστάσεις για επιχειρήσεις για τον χρήστη μας.

Επιλέγοντας ο χρήστης 5 φωτογραφίες, βρίσκουμε παρόμοιες φωτογραφίες με αυτές που εκείνος επέλεξε, κάνουμε ομαδοποίηση των παρόμοιων φωτογραφιών, και οι επιχειρήσεις των οποίων οι φωτογραφίες ανήκουν στην ίδια ομάδα με τις όμοιες φωτογραφίες αποτελούν τις συστάσεις. Έτσι μπορούμε να προτείνουμε τις συγκεκριμένες επιχειρήσεις στον χρήστη της εφαρμογής. Αυτό γίνεται με την εκτέλεση του ακόλουθου ερωτήματος σε cypher από τη JavaScript:



```
export const FETCH_PHOTO_RECOMMENDATIONS =
`
UNWIND $selectedPhotos AS photoId
MATCH (p:Photo {id: photoId})
MATCH (recPhoto:Photo {partition: p.partition})<-[:HAS_PHOTO]-(b:Business)
WITH b, COUNT(*) AS num, COLLECT(recPhoto)[0] AS businessPhoto ORDER BY num DESC LIMIT 100
RETURN COLLECT(b {.*, photo: businessPhoto.id}) AS recommendations
`;
```

Πρόταση επιχείρησης J. Alexander's

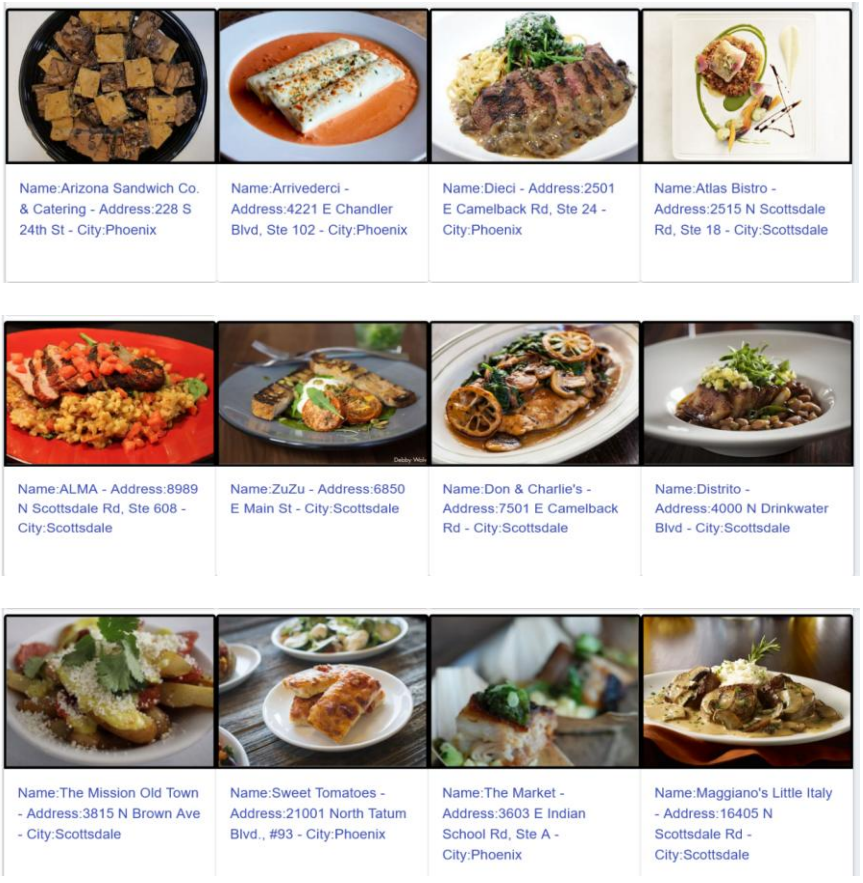
Το ερώτημα μας επιστρέφει επιχειρήσεις που σχετίζονται με τις φωτογραφίες που έχει αρχικά επιλέξει ο χρήστης της εφαρμογής.

Ακολουθούν τέσσερα παραδείγματα επιλογής φωτογραφιών και συστάσεων επιχειρήσεων.

Παράδειγμα 1:

Φωτογραφίες που επέλεξε ο χρήστης μας

Μερικές από τις επιχειρήσεις που του προτάθηκαν



Name:Pa'La - Address:2107 N 24th St - City:Phoenix	Name:Sweet Tomatoes - Address:9029 E Talking Stick Way - City:Scottsdale	Name:Voila French Bistro and Wine Bar - Address:10135 E Via Linda, Ste C120 - City:Scottsdale	Name:Alexi's Grill - Address:3550 N Central Ave, Ste 120 - City:Phoenix

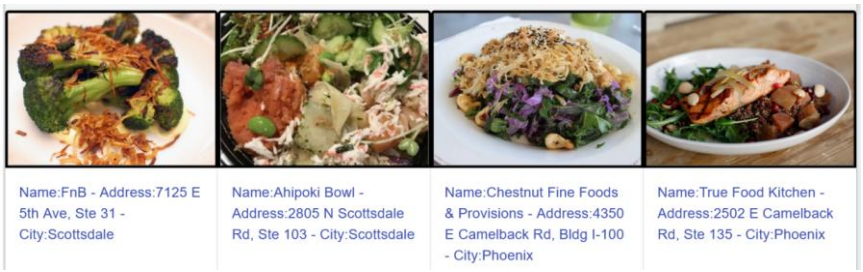
Name:La Locanda Ristorante Italiano - Address:6830 E 5th Ave, Ste 108 - City:Scottsdale	Name:Sweet Tomatoes - Address:4723 E Ray Road - City:Phoenix	Name:North Italia - Address:North Arcadia, 4925 N 40th St - City:Phoenix	Name:stonegrill - Address:5350 East Marriott Drive - City:Phoenix

Name:Big Earl's BBQ - Address:7213 E 1st Ave - City:Scottsdale	Name:Bootleggers Modern American Smokehouse - Address:3375 E Shea Blvd - City:Phoenix	Name:Clever Koi - Address:4236 N Central Ave, Ste 100 - City:Phoenix	Name:The Gladly - Address:2201 E Camelback Rd, Ste 106 - City:Phoenix

Παράδειγμα 2:

Φωτογραφίες που επέλεξε ο χρήστης μας

Μερικές από τις επιχειρήσεις που του προτάθηκαν

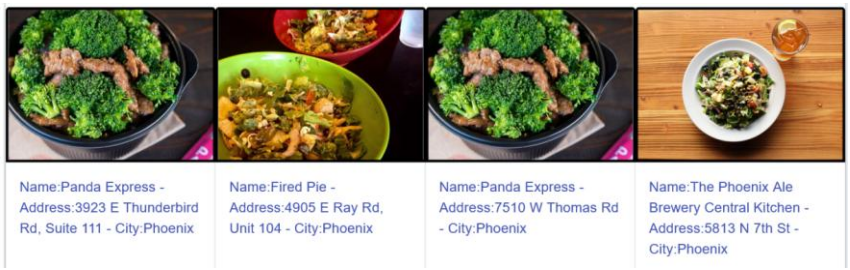


Name:FnB - Address:7125 E 5th Ave, Ste 31 - City:Scottsdale

Name:Ahipoki Bowl - Address:2805 N Scottsdale Rd, Ste 103 - City:Scottsdale

Name:Chestnut Fine Foods & Provisions - Address:4350 E Camelback Rd, Bldg I-100 - City:Phoenix

Name:True Food Kitchen - Address:2502 E Camelback Rd, Ste 135 - City:Phoenix

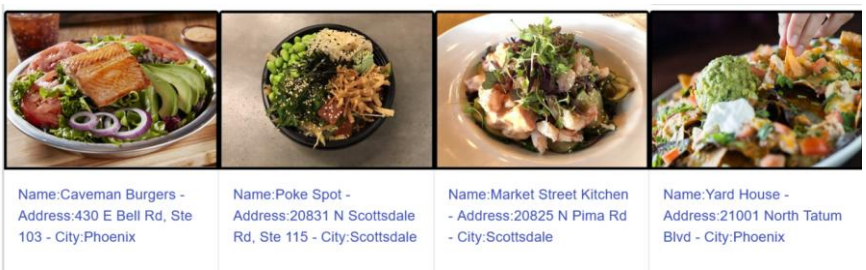


Name:Panda Express - Address:3923 E Thunderbird Rd, Suite 111 - City:Phoenix

Name:Fired Pie - Address:4905 E Ray Rd, Unit 104 - City:Phoenix

Name:Panda Express - Address:7510 W Thomas Rd - City:Phoenix

Name:The Phoenix Ale Brewery Central Kitchen - Address:5813 N 7th St - City:Phoenix

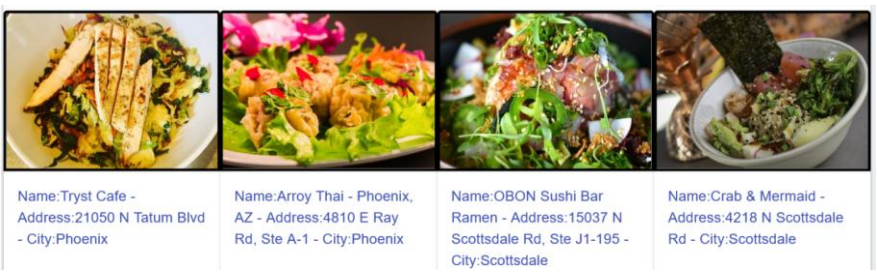


Name:Caveman Burgers - Address:430 E Bell Rd, Ste 103 - City:Phoenix

Name:Poke Spot - Address:20831 N Scottsdale Rd, Ste 115 - City:Scottsdale

Name:Market Street Kitchen - Address:20825 N Pima Rd - City:Scottsdale

Name:Yard House - Address:21001 North Tatum Blvd - City:Phoenix



Name:Tryst Cafe - Address:21050 N Tatum Blvd - City:Phoenix

Name:Arroy Thai - Phoenix, AZ - Address:4810 E Ray Rd, Ste A-1 - City:Phoenix

Name:OBON Sushi Bar Ramen - Address:15037 N Scottsdale Rd, Ste J1-195 - City:Scottsdale

Name:Crab & Mermaid - Address:4218 N Scottsdale Rd - City:Scottsdale

Name:Coconut's Fish Cafe - Address:7366 E Shea Blvd, Ste 110 - City:Scottsdale	Name:Poké Catcher - Address:49 W Thomas Rd - City:Phoenix	Name:Panda Express - Address:3013 W Agua Fria Freeway C1-a, Ste 4 - City:Phoenix	Name:Panda Express - Address:2501 W Happy Valley Rd - City:Phoenix

Name:Mora Italian - Address:5651 N 7th St - City:Phoenix	Name:The Vig McDowell Mountain - Address:10199 E Bell Rd - City:Scottsdale	Name:Citizen Public House - Address:7111 E 5th Ave, Ste E - City:Scottsdale	Name:Sonata's Restaurant - Address:10050 N Scottsdale Rd, Ste 127 - City:Scottsdale

Name:Panda Express - Address:2434 E Baseline Rd, Ste 101 - City:Phoenix	Name:Yard House - Address:7014 East Camelback Rd - City:Scottsdale	Name:Modern Market Eatery - Address:4821 N Scottsdale Rd, Ste 100 - City:Scottsdale	Name:Modern Market Eatery - Address:16203 N Scottsdale Rd - City:Scottsdale

Name:Copper Blues Rock Pub & Kitchen - Phoenix - Address:50 W Jefferson St, Ste 200 - City:Phoenix	Name:J & G Steakhouse - Address:6000 E Camelback Rd - City:Scottsdale	Name:Kale & Clover - Address:20511 N Hayden Rd - City:Scottsdale	Name:Slanted Rice Vietnamese Bistro - Address:6149 N Scottsdale Rd - City:Scottsdale

Παράδειγμα 3:

Φωτογραφίες που επέλεξε ο χρήστης μας

Μερικές από τις επιχειρήσεις που του προτάθηκαν

Name:Mancuso's Restaurant - Address:201 E Washington St - City:Phoenix

Name:Streets of New York - Address:16838 N 7th St - City:Phoenix

Name:The Capital Grille - Address:2502 E Camelback Rd - City:Phoenix

Name:Dominick's Steakhouse - Address:15169 N Scottsdale Rd - City:Scottsdale

Name:Mastro's City Hall - Address:6991 E Camelback Rd - City:Scottsdale

Name:The Grill - Address:17020 N Hayden Rd - City:Scottsdale

Name:Uncle Sal's Italian Restaurant - Address:3370 N Hayden Rd, Ste 120 - City:Scottsdale

Name:Garden Court Restaurant Scottsdale Plaza Resort - Address:7200 N Scottsdale Rd - City:Scottsdale

Name:Jewel of the Crown - Address:7373 E Scottsdale Mall, Ste 1 - City:Scottsdale

Name:Macayo's Mexican Table - Address:12637 S 48th St - City:Phoenix

Name:Tapas Papa Frita - Address:7114 E Stetson Dr, Ste 210 - City:Scottsdale

Name:Flavors of India - Address:4515 N 16th St - City:Phoenix

Name:Artizen - Address:2401 E Camelback Rd - City:Phoenix

Name:Eddie V's Prime Seafood - Address:15323 N Scottsdale Rd - City:Scottsdale

Name:Dubliner - Address:3841 E Thunderbird Rd, Ste 111 - City:Phoenix

Name:The Henry - Address:4455 E Camelback Rd - City:Phoenix

Name:Sushi Roku - Address:W Scottsdale, 7277 E Camelback Rd - City:Scottsdale

Name:Bourbon Steak - Address:7575 East Princess Drive - City:Scottsdale

Name:Squash Blossom - Address:705 N 1st St - City:Phoenix

Name:Daily Dose - Address:4020 N Scottsdale Rd - City:Scottsdale

Name:Rehab Burger Therapy - Address:7210 E 2nd St - City:Scottsdale

Name:Bitter & Twisted Cocktail Parlour - Address:1 W Jefferson St - City:Phoenix

Name:Steak 44 - Address:5101 N 44th St - City:Phoenix

Παράδειγμα 4 :

Φωτογραφίες που επέλεξε ο χρήστης μας

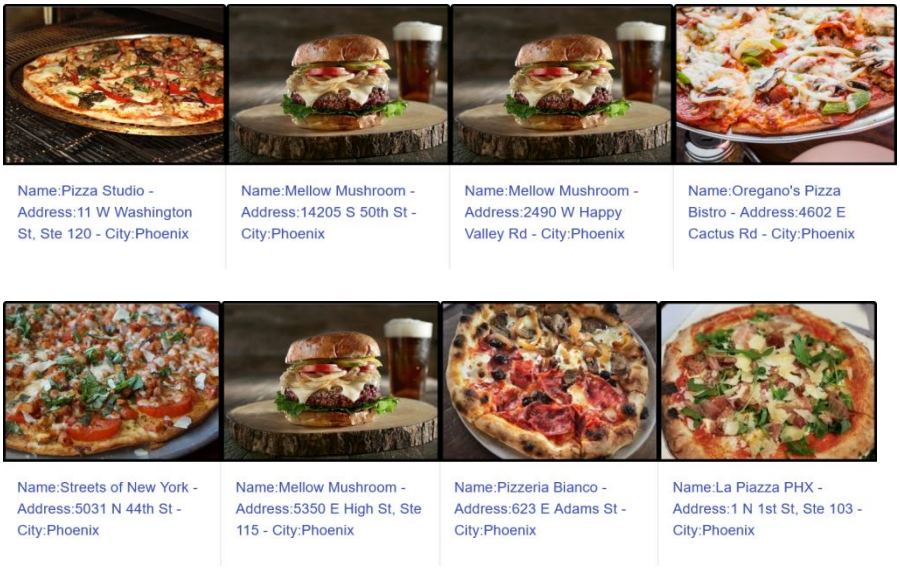
Μερικές από τις επιχειρήσεις που του προτάθηκαν

Name:My Pie Pizza - Address:12601 N Tatum Blvd - City:Phoenix	Name:1000 Degrees Neapolitan Pizzeria - Address:7000 E Mayo Blvd - City:Phoenix	Name:Doughbird - Address:4385 E Indian School Rd - City:Phoenix	Name:Cibo - Address:603 N 5th Ave - City:Phoenix

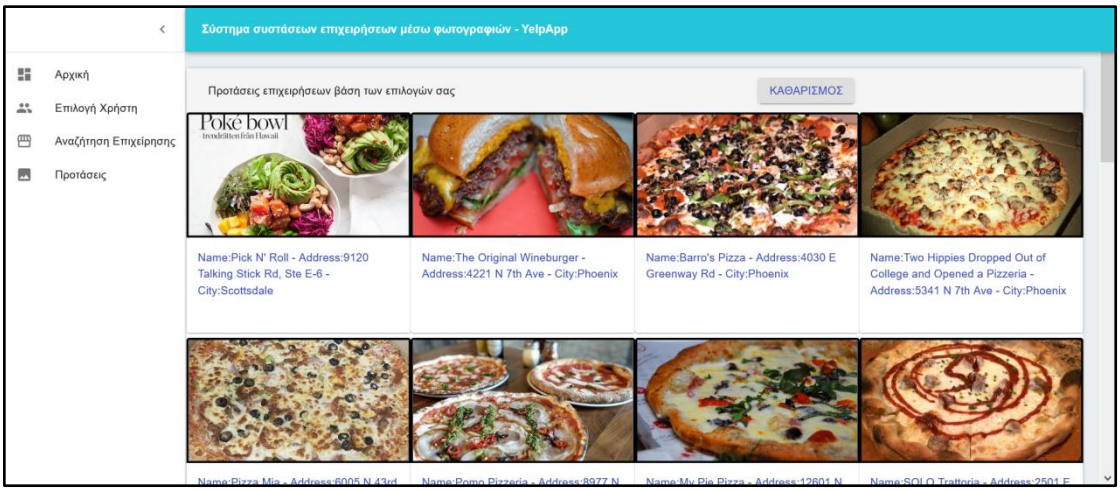
Name:McDonald's - Address:8849 N 7th St - City:Phoenix	Name:McDonald's - Address:2911 S 99th Ave - City:Phoenix	Name:Streets of New York - Address:4757 E Greenway Rd - City:Phoenix	Name:McDonald's - Address:2427 W Thomas Rd - City:Phoenix

Name:McDonald's - Address:4505 E Cactus Rd - City:Phoenix	Name:McDonald's - Address:5120 W Baseline Rd - City:Phoenix	Name:The Parlor - Address:1916 E Camelback Rd - City:Phoenix	Name:McDonald's - Address:4019 E Chandler - City:Phoenix

Name:McDonald's - Address:5750 W Thomas - City:Phoenix	Name:Streets of New York - Address:16838 N 7th St - City:Phoenix	Name:Bottled Blonde - Address:7340 E Indian Plz - City:Scottsdale	Name:Streets of New York - Address:2805 West Carefree Highway - City:Phoenix



Στην Εικόνα 4.22 φαίνεται πως παρουσιάζονται οι συστάσεις στην εφαρμογή μας.

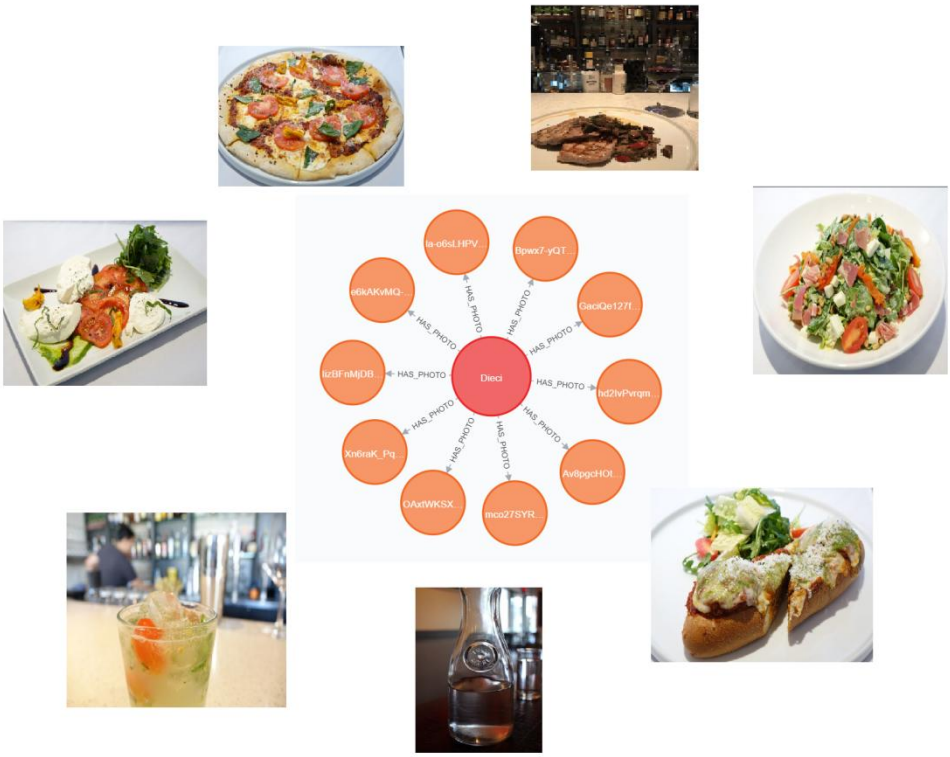


Εικόνα 4.22 : Συστάσεις επιχειρήσεων βάσει φωτογραφιών

Πατώντας το κουμπί “ΚΑΘΑΡΙΣΜΟΣ” εμφανίζονται στον χρήστη και πάλι 200 φωτογραφίες τυχαία και μπορεί να κάνει νέα επιλογή.

Θα πρέπει να σημειωθεί πως κάθε επιχείρηση συνδέεται με περισσότερες από μια διαφορετικές φωτογραφίες, καθώς και ίδιες φωτογραφίες υπάρχουν στη βάση με διαφορετικό id και ανήκουν σε διαφορετικές επιχειρήσεις. Γι’ αυτό παρατηρούμε και την επανάληψη πολλών ίδιων φωτογραφιών.

Στην Εικόνα 4.23 φαίνονται κάποιες από τις εικόνες της επιχείρησης με όνομα “Dieci”. Όπως αναφέραμε κάθε επιχείρηση έχει περισσότερες από μια φωτογραφίες. Η συγκεκριμένη επιχείρηση έχει συνολικά 25 φωτογραφίες.



Εικόνα 4.23 : Φωτογραφίες επιχείρησης

5

Επίλογος

5.1 *Σύνοψη και συμπεράσματα*

Στα πλαίσια της διπλωματικής μας εργασίας δημιουργήσαμε ένα σύστημα εξατομικευμένων συστάσεων για επιχειρήσεις βάσει φωτογραφιών, με μια εύκολη, ελκυστική και γραφική διεπαφή. Δεν εμπλέκουμε τον χρήστη σε πολύπλοκους μηχανισμούς, ούτε τον υποχρεώνουμε να δηλώσει επιπλέον ιδιωτικά στοιχεία στην εφαρμογή.

Μελετήσαμε έννοιες των συστημάτων συστάσεων και σύγχρονες μεθοδολογίες για τη δημιουργία προσωποποιημένων συστάσεων καθώς και την εξέλιξη των βάσεων δεδομένων και πως φτάσαμε στις γραφικές βάσεις δεδομένων, εξετάζοντας τα σαφή πλεονεκτήματα τους έναντι των απλών σχεσιακών βάσεων δεδομένων. Αναλύσαμε τη neo4j, την κορυφαία αυτή τη στιγμή βάση δεδομένων γράφων [19][20], και πως αυτή βοηθάει στην υλοποίηση σύγχρονων εφαρμογών οι οποίες απαιτούν μεγάλο όγκο ισχυρά διασυνδεδεμένων δεδομένων.

Μελετήσαμε θεωρητικά τους αλγορίθμους γράφων του neo4j που έχουν να κάνουν με κριτήρια ομοιότητα, με εύρεση κεντρότητας και ομαδοποίηση στοιχείων. Αναπτύχθηκε διαδικτυακή εφαρμογή η οποία χρησιμοποιώντας state-of-the-art τεχνολογίες (neo4j, react, cypher, machine learning) και βασικά κριτήρια ομοιότητας, δημιουργεί προσωποποιημένες συστάσεις για επιχειρήσεις βάσει φωτογραφιών. Χρησιμοποιήσαμε το δημόσιο σύνολο δεδομένων του Yelp, το οποίο επεξεργαστήκαμε γράφοντας κώδικα σε python και το φορτώσαμε στη βάση χρησιμοποιώντας το εργαλείο admin import tool του neo4j. Πρόκειται

για ένα εργαλείο με το οποίο μπορούμε να εισάγουμε πολύ μεγάλα σύνολα δεδομένων στη βάση σε πολύ λίγο χρόνο.

Με το εργαλείο Google Vision AI, το οποίο διαθέτει μηχανική μάθηση, κάναμε αναγνώριση ετικετών (Label detection) στις φωτογραφίες του συνόλου δεδομένων, εισάγοντάς τες στη βάση και δημιουργώντας και την αντίστοιχη σχέση ανάμεσα στις φωτογραφίες και τις ετικέτες. Αναπτύξαμε αποδοτικά ερωτήματα σε γλώσσα cypher η εκτέλεση των οποίων επιστρέφει και εμφανίζει τα αποτελέσματα στο front-end της εφαρμογής μας. Μελετήσαμε και ενσωματώσαμε στα ερωτήματα γνωστά κριτήρια ομοιότητας Pearson, Cosine, Jaccard, καθώς και τους αλγορίθμους Label Propagation, PageRank.

Είναι βασικό να σημειωθεί πως τα αποτελέσματα της εφαρμογής μας είναι ποιοτικά, η προσέγγιση μας επικεντρώνεται περισσότερο στην ποιότητα, ενώ θα έπρεπε να ληφθούν υπόψη και άλλοι παράγοντες όπως πχ. το τι τελικά επιλέγει ο χρήστης. Δεν υπάρχει δυνατότητα διατήρησης ιστορικού των επιλογών ενός χρήστη μετά τη σύσταση επιχειρήσεων που του γίνεται. Και αυτό γιατί δεν έχουμε πραγματικούς χρήστες στην εφαρμογή μας.

Στην εργασία εντοπίσαμε επίσης και κάποια ιδιαίτερα σημεία που έχριζαν σωστής αντιμετώπισης. Ένα από αυτά έχει να κάνει με την έκδοση της βάσης του neo4j που χρησιμοποιήσαμε και ποιες βιβλιοθήκες αλγορίθμων αυτή υποστηρίζει. Όπως αναφέραμε και στην Ενότητα 3.4.2 χρησιμοποιήσαμε την έκδοση 3.5.26 η οποία υποστηρίζει τη βιβλιοθήκη algo. Αυτή τη στιγμή βρισκόμαστε στην έκδοση 4.2 στην οποία η βιβλιοθήκη algo έχει αντικατασταθεί από τη Graph Data Science – GDS η οποία έρχεται να βελτιώσει την προηγούμενη. Έπρεπε λοιπόν να εισάγουμε χειροκίνητα τη βιβλιοθήκη algo στη βάση μας.

Ένα ακόμη ιδιαίτερο σημείο ήταν η μορφή των δεδομένων και το πολύ μεγάλο μέγεθος τους. Τα JSON αρχεία του συνόλου δεδομένων τα επεξεργαστήκαμε και τα μετατρέψαμε σε CSV μορφή κρατώντας μόνο εκείνα τα στοιχεία που μας ενδιέφεραν γράφοντας κώδικα σε python. Και αυτό το κάναμε γιατί χρησιμοποιήσαμε το εργαλείο admin import tool για την εισαγωγή του συνόλου δεδομένων στη βάση που απαιτεί τα αρχεία να είναι σε μορφή CSV. Επιλέξαμε αυτόν τον τρόπο μιας και με τη χρήση των εντολών LOAD JSON ή LOAD CSV με periodic.commit, οδηγηθήκαμε σε κατάρρευση του συστήματος. Καταφέραμε όμως με τη χρήση του admin import tool του neo4j να φορτώσουμε όλα τα δεδομένα στη βάση μόλις σε 14 λεπτά.

Καταλήγουμε πως είναι πολύ σημαντικό να διαθέτουμε ένα καλό, καθαρό και καλά δομημένο σύνολο δεδομένων έτσι ώστε να είναι εύκολα διαχειρίσημο και να μπορούμε από τα δεδομένα να αντλήσουμε σωστά αξιολογήσιμα αποτελέσματα.

Συμπερασματικά μπορούμε να πούμε πως όσο μεγαλώνει ένα σύνολο δεδομένων, τόσο πιο απαραίτητη γίνεται η χρήση μιας Graph Database, για λόγους οπτικοποίησης των δεδομένων

και παραγωγής γνώσης από αυτήν. Η αμεσότητα εκτέλεσης αποδοτικών ερωτημάτων cypher από την εφαρμογής και η επιστροφή και εμφάνιση των αποτελεσμάτων δίνει τη δυνατότητα δημιουργίας ισχυρών real time συστημάτων προσωποποιημένων συστάσεων (Real Time Personalized Recommendation Systems).

5.2 *Μελλοντικές επεκτάσεις*

Η συγκεκριμένη εφαρμογή που αναπτύχτηκε στα πλαίσια αυτής της διπλωματικής εργασίας μπορεί να χρησιμοποιηθεί από τον εμπορικό σύλλογο μιας μικρής πόλης ή σαν ένα επιπλέον εργαλείο σε έναν τουριστικό οδηγό της πόλης, για τη σύσταση επιχειρήσεων σε κάποιον επισκέπτη ο οποίος δεν γνωρίζει τα καταστήματα της πόλης. Αυτό γίνεται με έναν έξυπνο, εύκολο και γραφικό τρόπο μέσω μιας φιλικής και εύχρηστης διεπαφής. Ο επισκέπτης επιλέγει τυχαία κάποιες φωτογραφίες που του φαίνονται πιο ελκυστικές και η εφαρμογή του προτείνει επιχειρήσεις βάσει της επιλογής που έκανε. Επίσης του δίνεται και η δυνατότητα μιας πιο προσωποποιημένης αναζήτησης επιχειρήσεων βάσει κειμένου και κατηγορίας επιχείρησης.

Όπως αναφέραμε σε προηγούμενη ενότητα, τα αποτελέσματα των συστάσεων που κάνει η εφαρμογή μας είναι ποιοτικά και όχι ποσοτικά. Επειδή κάθε τι που δεν είναι μετρίσιμο δεν μπορεί και να αξιολογηθεί σωστά, μελλοντικά θα μπορούσε να γίνει τροποποίηση της εφαρμογής ώστε να έχουμε και μετρίσιμα αποτελέσματα από τις συστάσεις που γίνονται με τη διατήρηση ιστορικού των επιλογών που κάνει ο χρήστης μετά τις συστάσεις που του γίνονται. Να έχουμε δηλαδή πραγματικούς χρήστες στην εφαρμογή.

Επίσης σαν μελλοντική επέκταση της εφαρμογής, προτείνουμε την προσθήκη επιλογής εμφάνισης της κορυφαίας επιχείρησης ανά κατηγορία σε σχόλια και αστέρια αξιολόγησης. Η εφαρμογή μπορεί να αναφέρεται και μόνο σε μια κατηγορία επιχείρησης όπως για παράδειγμα ξενοδοχεία.

Αποτελεί ενδιαφέρον να δούμε και την αποδοτικότητα της εφαρμογής με τη χρήση νεότερης έκδοσης βάσης του neo4j και της νέας βιβλιοθήκης αλγορίθμων γράφων GDS – Graph Data Science η οποία έρχεται να βελτιώσει προβλήματα της προηγούμενης βιβλιοθήκης γράφων της algo .

Η μεταφορά ενός τέτοιου συστήματος συστάσεων στον τομέα τον κινητών συσκευών θα ήταν πιθανόν μια μελλοντική επέκταση, μιας και το πρόβλημα πρόσβασης στην πληροφορία γίνεται ακόμη πιο δύσκολο λόγω των δυσκολιών που οφείλονται σε περιορισμούς του υλικού. Είναι σημαντικό να σημειωθεί ότι οι αλγόριθμοι που εφαρμόζονται σε συστήματα που βασίζονται στον παγκόσμιο ιστό δεν μπορούν να μεταφερθούν αυτούσια σε συστήματα που

προορίζονται για μια κινητή συσκευή, αφού υπάρχουν διαφορετικές ανάγκες, χαρακτηριστικά και περιορισμοί.

6

Βιβλιογραφία

[1] B. Schwartz, *The Paradox of Choice*, vol. 30. HarperCollins Publishers (Australia) Pty. Ltd., 2004.

[2] P. Melville and V. Sindhwani, "Recommender Systems,"IBMT.J. Watson Research Center,Yorktown Heights.

[3] N. Yi, C. Li, X. Feng, and M. Shi, "Design and implementation of movie recommender system based on graph database," *Proc. - 2017 14th Web Inf. Syst. Appl. Conf. WISA 2017*, vol. 2018-Janua, 2018.

[4] J. A. Konstan and J. Riedl, "Recommender systems: From algorithms to user experience," *User Model. User-adapt. Interact.*, vol. 22, no. 1–2, pp. 101–123, 2012.

[5] D. Jannach , M. Zanker and A. Felfernig, " Recommender Systems: An Introduction", *Cambridge University Press, G. (2010).*

[6] A. Oulasvirta, T. Rattenbury, L. Ma, and E. Raita, "Habits make smartphone use more pervasive," *Pers. Ubiquitous Comput.*, vol. 16, no. 1, pp. 105–114, 2012.

[7] N. Polatidis and C. K. Georgiadis, "Mobile recommender systems: An overview of technologies and challenges," *2013 2nd Int. Conf. Informatics Appl. ICIA 2013*, pp. 282–287, 2013.

[8] R. Kitamura and T. Itoh, "Tourist spot recommendation applying generic object recognition with travel photos," *Inf. Vis. - Biomed. Vis. Vis. Built Rural Environ. Geom. Model. Imaging, IV 2018*, pp. 1–5, 2018.

[9] L. Cao, J. Luo, A. Gallagher, X. Jin, J. Han, and T. S. Huang, “A worldwide tourism recommendation system based on geotagged web photos,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, 2010.

[10] “Applied Graph Algorithms with Neo4j 3.5 - Applied Graph Algorithms with Neo4j 3.5,” *Neo4j Graph Database Platform*. [Online]. Available: https://neo4j.com/graphacademy/training-applied-algos/enrollment/. [Accessed: 24-May-2021].

[11] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*. United States of America.: Addison-Wesley Publishing Company, 2016.

[12] K. L. Berg, T. Seymour, and R. Goel, “History Of Databases,” Int. J. Manag. Inf. Syst., vol. 17, no. 1, pp. 29–36, 2012.

[13] E. Robinson, Ian Webber, Jim Eifrem, *Graph Databases - New opportunities for connected data*, 2nd ed. United States of America.: O’REILLY, 2015.

[14] Y. Chen, “Comparison of Graph Databases and Relational Databases When Handling Large-Scale Social Data,” University of Saskatchewan, 2016.

[15] “10 Graph Algorithms Visually Explained | by Vijini Mallawaarachchi | Towards Data Science.” [Online]. Available: https://towardsdatascience.com/10-graph-algorithms-visually-explained-e57faa1336f3. [Accessed: 24-May-2021].

[16] “ACID.” [Online]. Available: https://el.wikipedia.org/wiki/ACID.

[17] Neo4j, “Neo4j Graph Platform - Developer Guides.” [Online]. Available: https://neo4j.com/developer/graph-platform/. [Accessed: 24-May-2021].

[18] Neo4j, “Graph Modeling Guidelines - Developer Guides.” [Online]. Available: https://neo4j.com/developer/guide-data-modeling/. [Accessed: 24-May-2021].

[19] Neo4j Inc, “Graph Data Science Algorithms | Graph Data Analysis | Neo4j.” [Online]. Available: https://neo4j.com/product/graph-data-science-library/?ref=pdf-ebook-graph-algo. [Accessed: 24-May-2021].

[20] DB-Engines, “DB-Engines Ranking of Graph DBMS,” 2016. [Online]. Available: https://db-engines.com/en/ranking/graph+dbms. [Accessed: 24-May-2021].

[21] JavaTpoint, “Graphdb vs Rdbms - javatpoint.” [Online]. Available: https://www.javatpoint.com/graphdb-vs-rdbms. [Accessed: 24-May-2021].

[22] N. Francis *et al.*, “Cypher : An Evolving Query Language for Property Graphs,” Houston, United States., 2018.

[23] Neo4j Inc., “Introduction - Neo4j Cypher Manual,” 2020. [Online]. Available: https://neo4j.com/docs/cypher-manual/current/introduction/. [Accessed: 24-May-

2021].

[24] A. E. Hodler, “An Overview of Graph Algorithms in Neo4j Use the Power of Connections to Drive Discovery,” 2018.

[25] M. Needham and A. E. Hodler, *A Comprehensive Guide to Graph Algorithms in Neo4j*. 2020.

[26] M. Needham and A. E. Hodler, *Graph Algorithms - Practical Examples in Apache Spark & Neo4j*. United States of America.: O’REILLY, 2019.

[27] Neo4j Inc., “Introduction to Graph Algorithms in Neo4j 4.x - Neo4j Graph Database Platform,” 2021. [Online]. Available: https://neo4j.com/graphacademy/training-iga-40/enrollment/. [Accessed: 24-May-2021].

[28] T. Point, “Node.js - Introduction - Tutorialspoint,” *https://www.tutorialspoint.com/*, 2019. [Online]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Accessed: 24-May-2021].

[29] I. Nmp, “About NPM,” *npmjs.com*, 2019. [Online]. Available: https://docs.npmjs.com/about-npm. [Accessed: 24-May-2021].

[30] H. Patel, “React JS: What and Why?. React.js is the most popular front-end… | by Harsh Patel | Medium.” [Online]. Available: https://harsh-patel.medium.com/react-js-what-and-why-e6cad2dfb4c3. [Accessed: 24-May-2021].

[31] JavaTpoint, “ReactJS Introduction.” [Online]. Available: https://www.javatpoint.com/react-introduction. [Accessed: 24-May-2021].

[32] Google, “Vision AI | Derive Image Insights via ML | Cloud Vision API,” 2020. [Online]. Available: https://cloud.google.com/vision. [Accessed: 24-May-2021].

[33] Neo4j Inc., “Importing Data with Neo4j 4.x - Importing Data with Neo4j 4.x.” [Online]. Available: https://neo4j.com/graphacademy/training-importing-data-40/00-import-40-about/. [Accessed: 24-May-2021].

7

Παράρτημα

Στον δικτυακό τόπο http://giapot.webservers.gr/simoreli/YelpApp/index.html θα βρείτε πληροφορίες σχετικές με τα τεχνολογικά εργαλεία που χρησιμοποιήθηκαν στην εφαρμογή μας, το σύνολο δεδομένων του Yelp, το κείμενο της διπλωματικής, κώδικας, καθώς και άλλες πληροφορίες. Είναι ένας ιστότοπος που αναπτύχθηκε κατά τη διάρκεια εκπόνησης της διπλωματικής για να συγκεντρώσουμε όλες τις πληροφορίες για εύκολη και γρήγορη πρόσβαση.

Ακολουθούν τα βασικά ερωτήματα σε cypher που χρησιμοποιήθηκαν στην εφαρμογή μας:

ΕΡΩΤΗΜΑ 1 :

Επιστρέφει για τον τρέχοντα χρήστη το όνομά του, πόσες κριτικές έχει κάνει, τον μέσο όρο των αξιολογήσεων του, πέντε κατηγορίες στις οποίες έχει κάνει κριτική και το σύνολο των φίλων του.

```
MATCH (u:User {id: $userId})
MATCH (u:User)-[:WROTE]->(r:Review)
WITH u, avg(r.stars) AS averageStars
ORDER BY averageStars DESC
LIMIT 1000
MATCH (u)-[:WROTE]->(:Review)-[:REVIEWS]->(:Business)-[:IN_CATEGORY]-(c:Category)
WITH u, averageStars,c.name as category, count(*) AS count ORDER BY u, count DESC
WHERE count > 5
RETURN u {.*,
          numReviews: toFloat(SIZE((u)-[:WROTE]->(:Review))),
          categories: COLLECT(category)[..5],
                  friends: max(apoc.node.degree(u,'FRIENDS')),
          averageStars} AS userInfo
```

ΕΡΩΤΗΜΑ 2

Εμφανίζει τις κορυφαίες κατηγορίες επιχειρήσεων μετά την ιεράρχηση τους.

```
MATCH (c:Category)
WHERE NOT EXISTS ((c)-[:NARROWER_THAN]->())
WITH c ORDER BY c.name LIMIT 200
RETURN COLLECT(c.name) AS categories
```

ΕΡΩΤΗΜΑ 3

Το ερώτημα αυτό δέχεται ως είσοδο την κατηγορία της επιχείρησης και το ελεύθερο κείμενο που εισάγει ο χρήστης για να αναζητήσει επιχειρήσεις. Μας επιστρέφει επιχειρήσεις στις οποίες έχουν κάνει κριτική άλλοι όμοιοι χρήστες με τον τρέχον χρήστη.

```
CALL db.index.fulltext.queryNodes("BusinessNameIndex", $searchText + "~0.75")
YIELD node, score
```

```
WITH node AS b ORDER BY score DESC LIMIT 200
MATCH (b)-[:IN_CATEGORY]->(c:Category {name: $category})
OPTIONAL MATCH (b)-[:HAS_PHOTO]->(p:Photo)
WITH b, COLLECT(p)[0] AS p
MATCH (me:User {id: $userId})
OPTIONAL MATCH (me)-[similarity:SIMILAR]-(other)-[:WROTE]-(:Review)-[:REVIEWS]->(b)
WITH DISTINCT b,p, avg(coalesce(similarity.score,0)) AS num ORDER BY num DESC LIMIT 1000
RETURN COLLECT(b {.*, photo: p.id, score: num}) AS businesses
```

ΕΡΩΤΗΜΑ 4

Για την επιχείρηση που επιλέγει ο χρήστης να δει τις κριτικές, του επιστρέφονται οι πιο σχετικές κριτικές που έγιναν από τους "έμπιστους" φίλους χρήστες του τρέχοντα χρήστη. Για κάθε κριτική εμφανίζεται το κείμενο της κριτικής, τα αστέρια αξιολόγησης, το όνομα του χρήστη που έκανε την κριτική και την ημερομηνία.

```
MATCH (b:Business {id: $businessId}), (me:User {id: $userId})
MATCH (b)<-[:REVIEWS]-(r:Review)<-[:WROTE]-(u:User)
OPTIONAL MATCH (u)<-[t:TRUSTS]-(me)
WITH r {.text, stars: toFloat(r.stars), name: u.name, date: toString(r.date)} AS review
ORDER BY coalesce(t.score, 0.0) DESC, r.date DESC LIMIT 1000
RETURN COLLECT(review) AS reviews
```

ΕΡΩΤΗΜΑ 5

Επιστρέφει τις 200 φωτογραφίες που εμφανίζονται τυχαία σε μορφή πλέγματος στην εφαρμογή μας. Από αυτές ο χρήστης καλείται να επιλέξει 5 φωτογραφίες για να γίνει το Photo-based recommendation.

```
MATCH (b:Business)-[:HAS_PHOTO]->(p:Photo)
WITH b, p SKIP toInteger(rand() * 30000)-1000 LIMIT 10000
WITH COLLECT(p{photoId: p.id, businessId: b.id}) AS photos
RETURN apoc.coll.shuffle(photos)[..199] AS photos
```

Το ερώτημα παίρνει ως παράμετρο τα ids των φωτογραφιών που επέλεξε ο χρήστης και επιστρέφει τις επιχειρήσεις των οποίων οι φωτογραφίες ανήκουν στο ίδιο partition με τις άλλες παρόμοιες φωτογραφίες που βρέθηκαν.

```
UNWIND $selectedPhotos AS photoId
MATCH (p:Photo {id: photoId})
MATCH (recPhoto:Photo {partition: p.partition})<-[:HAS_PHOTO]-(b:Business)
WITH b, COUNT(*) AS num, COLLECT(recPhoto)[0] AS businessPhoto ORDER BY num DESC LIMIT 100
RETURN COLLECT(b {.*, photo: businessPhoto.id}) AS recommendations
```