

Προγραμματστική εργασία

Αθανασία Καρανίκα 2530

ΠΡΟΧΩΡΗΜΕΝΗ ΔΙΑΧΕΙΡΙΣΗ ΔΕΔΟΜΕΝΩΝ

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας, Βόλος
atkaranika@uth.gr

1 Εισαγωγή.

Τα αρχεία που παραδίδονται είναι :

- Για τα δεδομένα:
4 seasons bio.csv Bioshop.csv Bioway.csv Βιολογικό χωριό.csv ΓΩΝΙΑ.csv ΔΗ-ΜΗΤΡΑ.csv Ήταν καιρός....csv
- Για την εισαγωγή των δεδομένων στην βάση :
main.py
- Για να τρέξει το πρόγραμμα:
windows.py
- Εικόνες που χρησιμοποιήθηκαν:
background.jpg plant.gif

μελλοντικές επεκτάσεις και προσθήκες:

Μια καλή ιδέα για την εφαρμογή θα ήταν να υπήρχε ένας administrator που θα διέγραφε θα πρόσθετε και θα ανανέωνε μαγαζιά και προιόντα, επίσης θα μπορούσα να βρω διαθέσιμα On-line δεδομένα και να πάιρνω τα ωράρια των μαγαζιών σε πραγματικό χρόνο καθώς και την διαθεσιμότητα σε προιόντα. Επίσης η εφαρμογή μου περιορίζεται σε περιοχές της Αθήνας γεγονός που θα μπορούσε να επεκταθεί σε μεγαλύτερο εύρος.

2 Επιλογή θέματος

Εφαρμογή ένυρεσης κοντινότερων καταστημάτων

3 Περιγραφή και προδιαγραφές του θέματος.

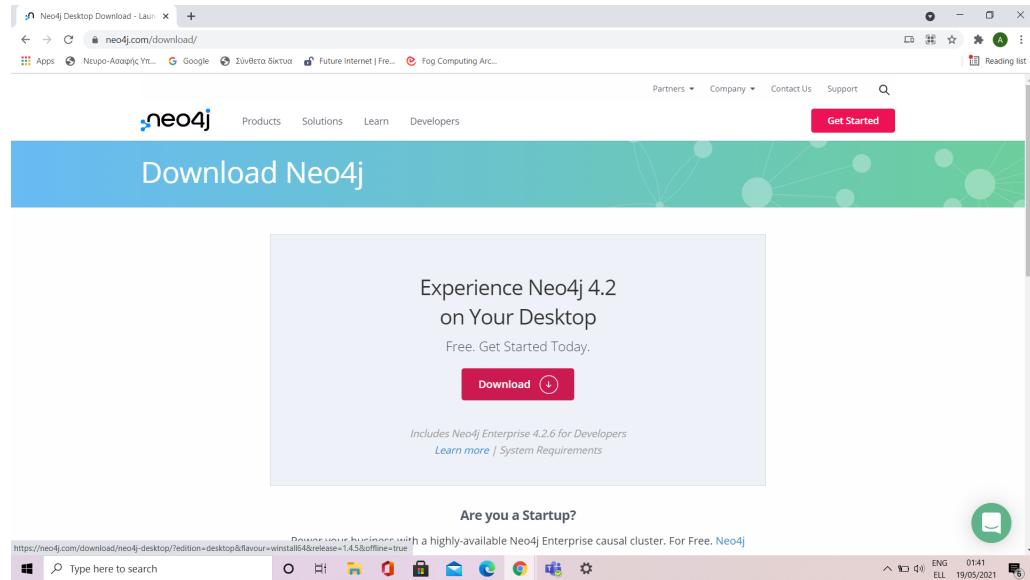
Στόχος της εφαρμογής είναι να προσφέρει σε πιθανούς πελάτες το κοντινότερο σημείο πώλησης του βιολογικού-χορτοφαγικού προϊόντος που ψάχνουν. Η εφαρμογή έχει την δυνατότητα να βρίσκει το πιο κοντινό σε απόσταση κατάστημα που διαθέτει το ζητούμενο προϊόν. Παράλληλα θα υπάρχει ένα είδος rank που θα βασίζεται στην συγνότητα επιλογής ενός προϊόντος από έναν χρήστη. Η δεύτερη δυνατότητα βασίζεται σε επισκέπτες που έχουν κάνει login. Για αυτούς που δεν έχουν κάνει σύνδεση ή δεν έχουν λογαριασμό δεν υπαρχει δυνατότητα να δουν το πιθανό αγαπημένο τους προιόν.

4 Επιλογή Συστήματος NoSQL ΒΔ κατάλληλου για το παραπάνω θέμα.

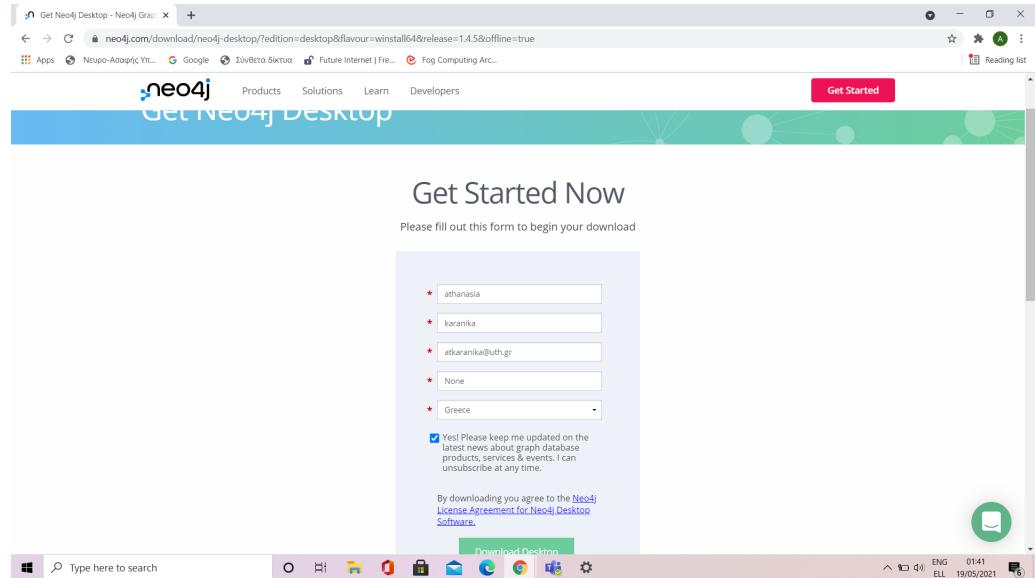
Η βάση που χρησιμοποίησα είναι η neo4j NoSql database διότι τα καταστήματα προσομοιώνονται καλύτερα ώς κόμβοι που έχουν συνδέσεις με προιόντα-κόμβους και αξιοποιούνται τα χαρακτηριστικά των ακμών τους. Ακόμη οι χρήστες είναι κόμβοι που συνδέονται με τα προιόντα που είχαν επιλέξει στο παρελθόν και το αξιοποιείται το χαρακτηριστικό της ακμής τους(rank).

5 Εγκατάσταση του Συστήματος NoSQL ΒΔ.

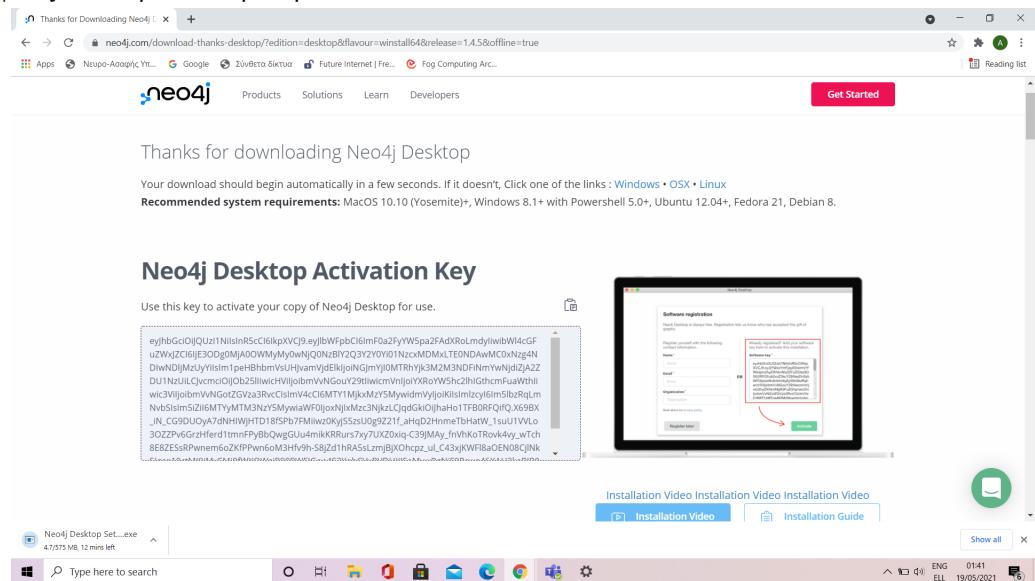
Από το link <https://neo4j.com/download/> πατάμε το download <https://neo4j.com/download/>



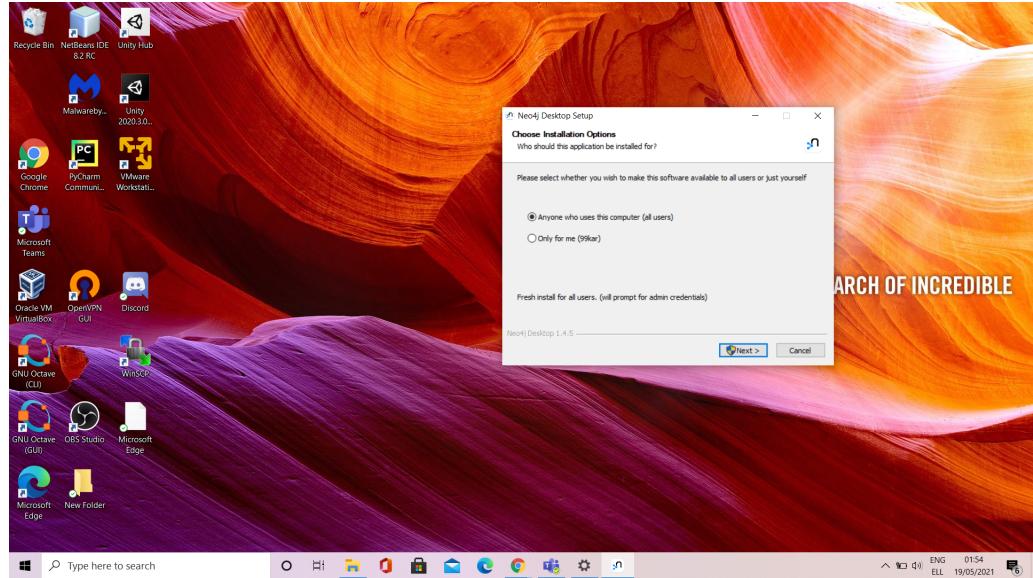
Μας εμφανίζει το παρακάτω παράθυρο στο οποίο βάζουμε τα στοιχεία από το πανεπιστήμιο θεσσαλίας.



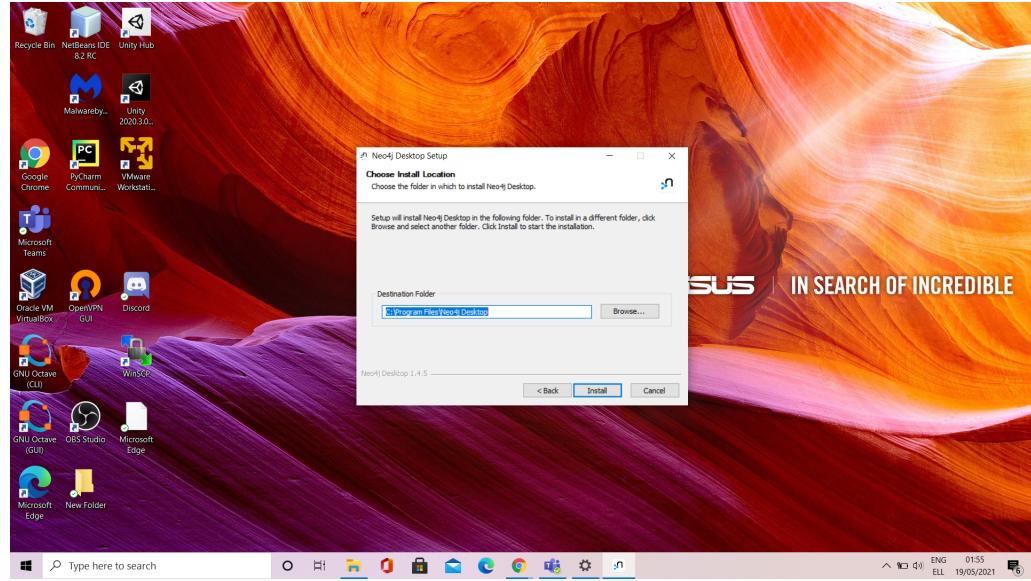
Αφού βάλουμε τα στοιχεία μας και πατήσουμε το πράσινο κουμπί download μας εμφανίζει το παρόντα παράθυρο.



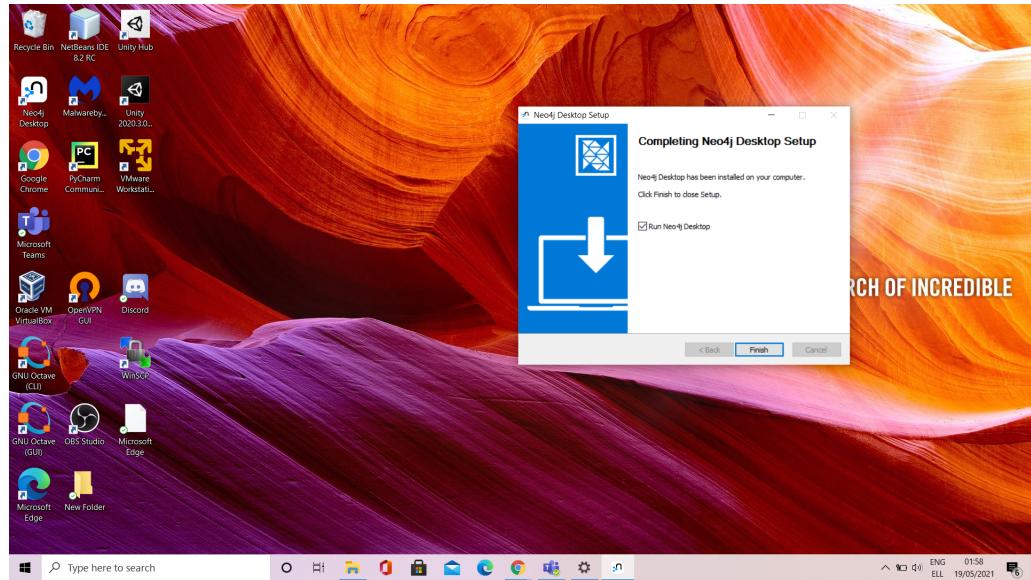
Από αυτό το σημείο ξεκινάει η αποθήκευση και πρέπει να κανουμε copy το κλειδί που δίνεται στο παράθυρο για να το χρησιμοποιήσουμε αργότερα. Επιλέγουμε την πρώτη επιλογή(παρ’όλο που η deafault επιλογή είναι η 2η) δηλαδη to anyone who uses this computer



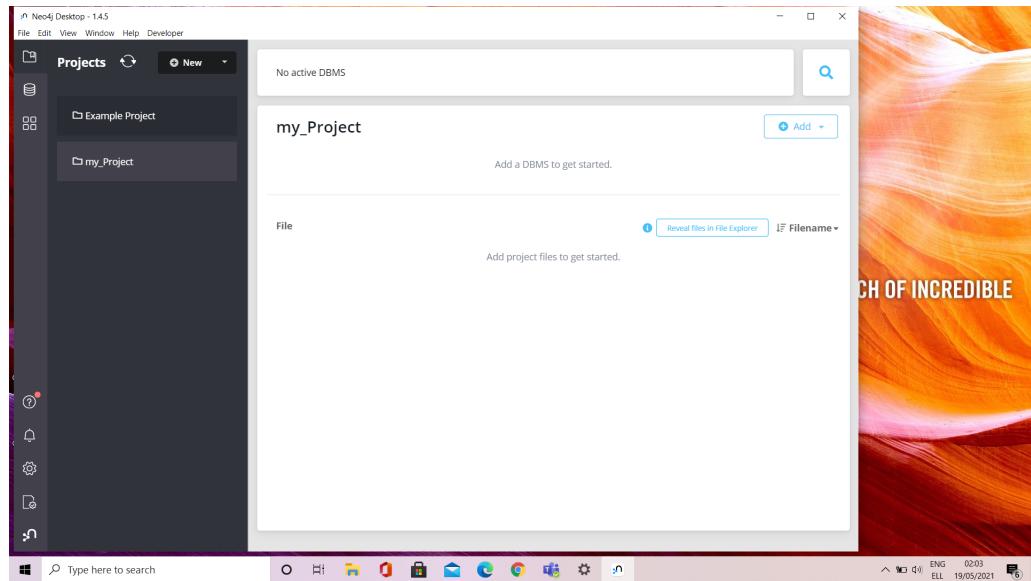
Στο επόμενο βήμα πατάμε install



Στοεπόμενο παράθυρο πατάμε finish



και μας εμφανίζεται η παρακάτω εικόνα



Στο σημείο αυτό πρέπει αρχικά να δημιουργήσουμε την βάση παταμε στο add(local dbms) και εισαγούμε κωδικό για την βάση το 1234 για να μπορέσει μετά να τρέξει η εφαρμογή. Αφού δημιουργηθεί η βάση πατάμε το start(πρασινό κουμπί) για να μπορέσει το πρόγραμμα να συνδεθεί και όταν εμφανιστεί η ειδοποίηση ότι η βάση ξεκίνησε τότε μπορούμε να προχωρήσουμε. Στην συνέχεια για να βάλουμε τα δεδομένα πάμε στο

πρόγραμμα main.py και εφόσον έχουμε τα υπόλοιπα αρχεία για τα δεδομένα μέσα στον ίδιο φάκελο θα γεμίζει η βάση με τα δεδομένα της εφαρμογής. Για ναν δημιουργηθεί επιτυχώς η βάση θα πρέπει να δημιουργήσουμε ένα φάκελο shops και εκεί να βάλουμε όλα τα .csv για κάθε μαγαζί το κεντρικό αρχείο shops.csv πρέπει αν βρίσκεται στον ίδιο φάκελο με την main.py και windows.py

6 Προσδιορισμός χρήσιμων ερωτημάτων για την εφαρμογή.

Τα ερωτήματα που γίνονται στην βάση είναι

- Εύρεση κοντινότερης απόστασης .

Στο ερώτημα αυτό η o server της βάσης βρίσκει όλα τα μαγαζιά που διαθέτουν το πριόν και βρίσκει τις αποστάσεις τους μέσω γεωγραφικού μήκους και πλάτους από τον η συνάρτηση που χρησιμοποιείται έιναι η geodesic(loc1, loc2) όπου loc1 loc2 το γεωγραφικό μήκος και πλάτος της διεύθυνσης που έδωσε ο χρήστης. Άμα δεν υπάρχει η διεύθυνση το ανακαλύπτει το πρόγραμμα και εμφανίζει ένα modal window.

- Εύρεση αγαπημένου προιόντος.

Στο ερώτημα αυτό η o server της βάσης βρίσκει όλα τα προιόντα που έχει επιλέξει στο παρελθόν ο χρήστης και κάνει ένα είδους rank ουσιαστικα ανάμεσα στην σχέση πελάτης-προιόν κρατάμε σαν χαρακτηριστικό το rank το οποίο αυξάνεται κατα 1 κάθε φορά που ο χρήστης αναζητά κάποιο προιόν. Κατά την είσοδο του χρήστη άμα το όνομα που εισήγαγε δεν υπάρχει εμφανίζεται μήνυμα λάθους καθώς και αν εισάγει λανθασμένο κωδικό.

Πιο σιγκεκριμένα:

Αρχικά σύνδεση στην βάση :

```
class graph:
    def __init__(self, uri, user, password):
        self.driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "1234"))
```

Μετά αφου ο χρήστης βάζει το προιόν και μια διεύθυνση ελέγχεται αν η διεύθυνση είναι σωστή, με την ακόλουθη συνάρτηση.

```
typed = self.entry.get()
address = self.entry_address.get()
geolocator = Nominatim(user_agent="main.py", timeout=50)
location = geolocator.geocode(address)

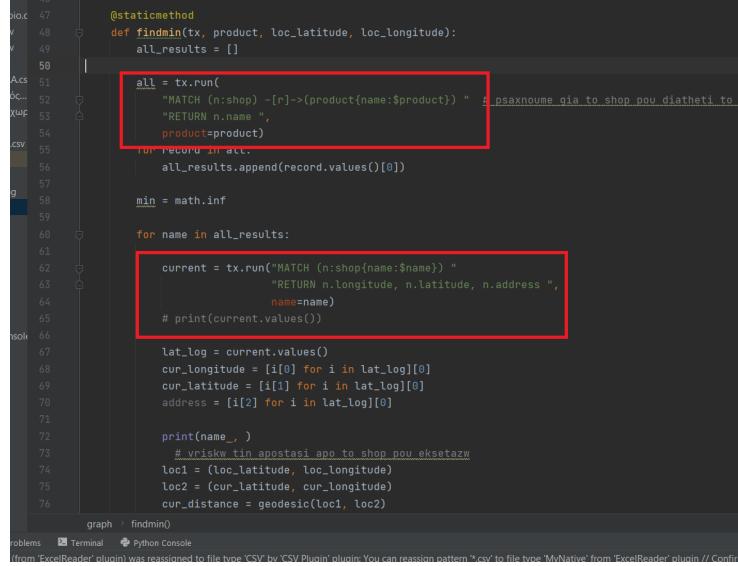
if typed not in self.list:
    messagebox.showerror("Error", "το προϊόν δεν είναι διαθέσιμο")
    return

if type(location) is type(None):
    messagebox.showerror("Error", "διεύθυνση δεν υπάρχει")
    return

self.dbs = graph("bolt://localhost:7687", "neo4j", "1234")
```

Εφόσον ελέγχεθεί ότι η διεύθυνση είναι έγκυρη γίνεται ένα query στην βάση που ψάχνει αρχικά όλα τα μαγαζιά που διαθέτουν το προϊόν και στην συνέχεια βρίσκει

όλες τις αποστάσεις από την τρέχουσα τοποθεσία του χρήστη και τελικά επιστρέφει το κατάστημα με την μικρότερη απόσταση.



```

@staticmethod
def findmin(tx, product, loc_latitude, loc_longitude):
    all_results = []
    all = tx.run(
        "MATCH (n:shop) -[r]-(product{name:$product}) "
        "+RETURN n.name",
        product=product)
    for record in all:
        all_results.append(record.values()[0])
    min = math.inf
    for name in all_results:
        current = tx.run("MATCH (n:shop{name:$name}) "
                        "+RETURN n.longitude, n.latitude, n.address ",
                        name=name)
        # print(current.values())
        lat_log = current.values()
        cur_longitude = [i[0] for i in lat_log][0]
        cur_latitude = [i[1] for i in lat_log][0]
        address = [i[2] for i in lat_log][0]
        print(name, )
        # vriskw tin apostasi apo to shop pou eksetazw
        loc1 = (loc_latitude, loc_longitude)
        loc2 = (cur_latitude, cur_longitude)
        cur_distance = geodesic(loc1, loc2)
        if cur_distance < min:
            min = cur_distance
    print(min)
    graph > findmin0

```

Στην συνέχεια αν ο χρήστης επιλέξει να κάνει δημιουργία λογαριασμού γίνεται ένα ερώτημα στην βάση που αφορά στο άμα υπάρχει άλλος χρήστης με το ίδιο όνομα.

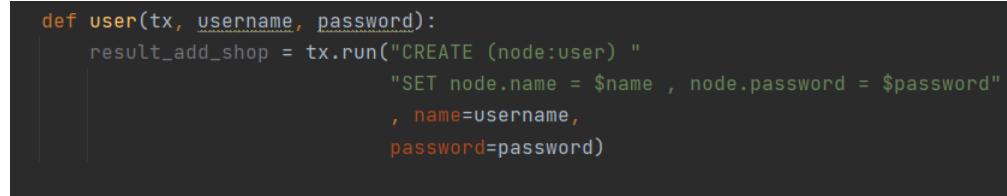


```

@staticmethod
def check_user_exists(tx, username, password):
    user = []
    all = tx.run("MATCH (n:user{name : $name, password : $password}) "
                "+RETURN n.name", name=username, password=password)
    for record in all:
        user.append(record.values()[0])
    if len(user) == 0:
        return 0
    else:
        return 1

```

εΦ'όσον η συνάρτηση αυτή επιστρέψει ότι δεν υπάρχει χρήστης με τέτοιο όνομα καλούμε την δημιουργία κόμβου με username το όνομα που πληκτρολόγησε ο χρήστης και κωδικό τον κεντρικό που πληκτρολόγησε.



```

def user(tx, username, password):
    result_add_shop = tx.run("CREATE (node:user) "
                            "SET node.name = $name , node.password = $password"
                            ", name=username, password=password")

```

Άμα ο χρήστης κάποια στιγμή αποφασίσει σε ένα επόμενο τρέξιμο του προγράμματος να κάνει σύνδεση θα πρέπει να εισάγει τον σωστό κωδικό για το username που πληκτρολόγησε, αν το όνομα δεν υπάρχει(η προηγούμενη συνάρτηση επιστρέψει

0) τότε εμφανίζεται ένα μήνυμα λάθους στον χρήστη. Το ίδιο και αν δεν πληκτρολογήθηκε ο σωστός κωδικός. Και εφόσον ο χρήστης συνδεθεί γίνεται το επόμενο query στην βάση για βρεθούν αγαπημένα προϊόντα του χρήστη. Παράλληλα για κάθε νέα αναζήτηση του χρήστη για κάποιο προϊόν το rank που έχει με αυτό το προϊόν αυξάνεται κατα 1 εφόσον το έχει ξαναεπιλέξει στο παρελθόν ή άμα είναι η πρώτη φορά που το επιλέγει δημιουργείται η ακμή. Αυτή η λειτουργία επιτυγχάνεται με τα ακόλουθα query.

```
@staticmethod
def find_user_products(tx, username, product):
    all = tx.run("RETURN EXISTS(:user {name: $name})-[:selected]-(:product {name:$product})", name=username, product=product)
    out = all.values()[0]

    if not out[0]:
        relationship = tx.run("MATCH (a:user {name: $name}),(b:product {name:$product}) "
                             "WHERE a.name = $name AND b.name = $product"
                             "CREATE(a)-[r:selected {rank : $rank}]->(b) "
                             "RETURN r.rank", name=username, product=product, rank=1)
    else:
        relationship = tx.run("MATCH ((:user {name: $name})-[r:selected]-(p:product {name:$product})) "
                             "SET r.rank = r.rank+1"
                             "RETURN r.rank"
                             ", name=username, product=product",
                             name=username, product=product)

    #print(relationship)
    return out[0]
```

Κοιτάμε αν υπάρχει η συγκεκριμένη σχέση
← οχι
Ναι

και πλέον στο μενού του χρήστη δίνεται και η επιλογή να διαγράψει τον λογαριασμό του και γίνεται το εξής query

```
@staticmethod
def delete(tx):
    global username
    out = tx.run("MATCH (user {name: $name}) "
                "DETACH DELETE user"
                ", name=username)",
                name=username)
```

7 Δημιουργία δεδομένων με χρήση data generator ή ανεύρεση έτοιμων δεδομένων.

Για την δημιουργία δεδομένων χρησιμοποίησα csv αρχεία που δημητριάσα εγώ διότι δεν βρήκα κάποιον αξιόλογο data generator για τις ανάγκες της εφαρμογής μου. Έχω ένα κεντρικό csv που περιέχει τα ονόματα όλων των διαθέσιμων καταστημάτων(shops.csv) καθώς και τις διευθύνσεις τους, και για κάθε κατάστημα μέσα στο shops.csv έχω ένα ακόμα csv που ονομάζεται ”όνομα καταστήματος”.csv και περιέχει όλα τα διαθέσιμα πριόντα του εκάστοτε καταστήματος.

8 Σχεδιασμός του μοντέλου της ΒΔ και υλοποίησή του στο Σύστημα NoSQL ΒΔ..

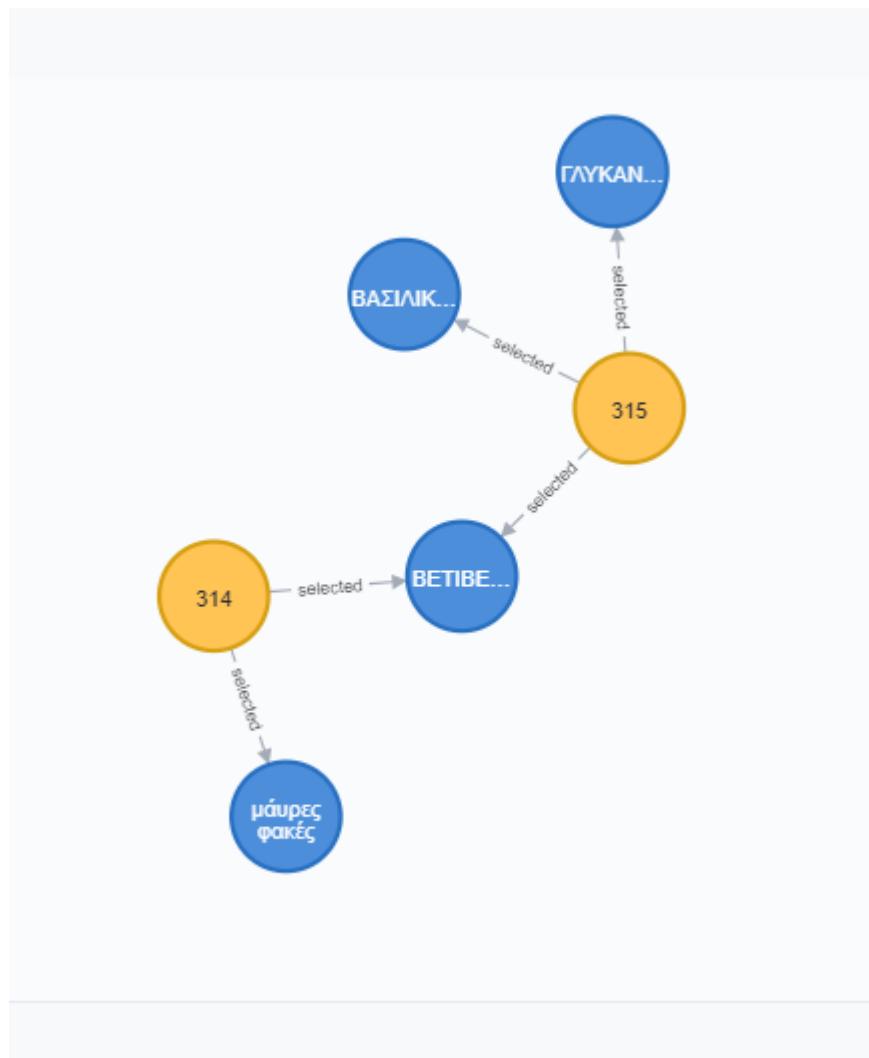
Το μοντέλο έχει ως εξής: Για κάθε κατάστημα βλέπουμε όλα τα διαθέσιμα προϊόντα και δημιουργούμε έναν κόμβο για το κατάστημα και ένα για το προϊόν, οι δύο κόμβοι συνδέονται μεταξύ τους με μια ακμή, ένα παράδειγμα της υλοποίησης είναι το εξής :



Στην εικόνα παρατηρούμε ότι οι δύο κόμβοι μαγαζών(βιολογικό χωριό και 4 seasons bio) μπορεί να έχουν κάποια κοινά προϊόντα αλλά αυτά προστίθονται μία φορά στην βάση.⁷

Παράλληλα για τους χρήστες έχουμε την δημιουργία χρήστη να έχει ώς αποτέλεσμα την δημιουργία ενός κόμβου χρήστη και από εκεί και πέρα όσες φορές συνδεθεί με αυτό το username και password θα δημιουργείται μια ακμή για κάθε καινούριο προϊόν που

εισάγεται και άμα ο χρήστης στο παρελθόν έχει ξαναεπιλέξει το προιόν απλά αυξάνεται το rank. Προύποθεση για να δημιουργηθεί νέος λογαριασμός είναι να μην υπάρχει μέσα στην βάση το ίδιο username αλλιώς εμφανίζεται μήνυμα λάθους και ο χρήστης πρέπει να ξαναπροσπαθήσει.



Παρατηρούμε ότι γενικά τα προιόντα δεν δημιουργούνται δεύτερη φορά απλά υπάρχει διαφορετικό rank για τον κάθε χρήστη.

9 Εκτέλεση των ερωτημάτων και εξέταση της ορθότητας των αποτελεσμάτων.

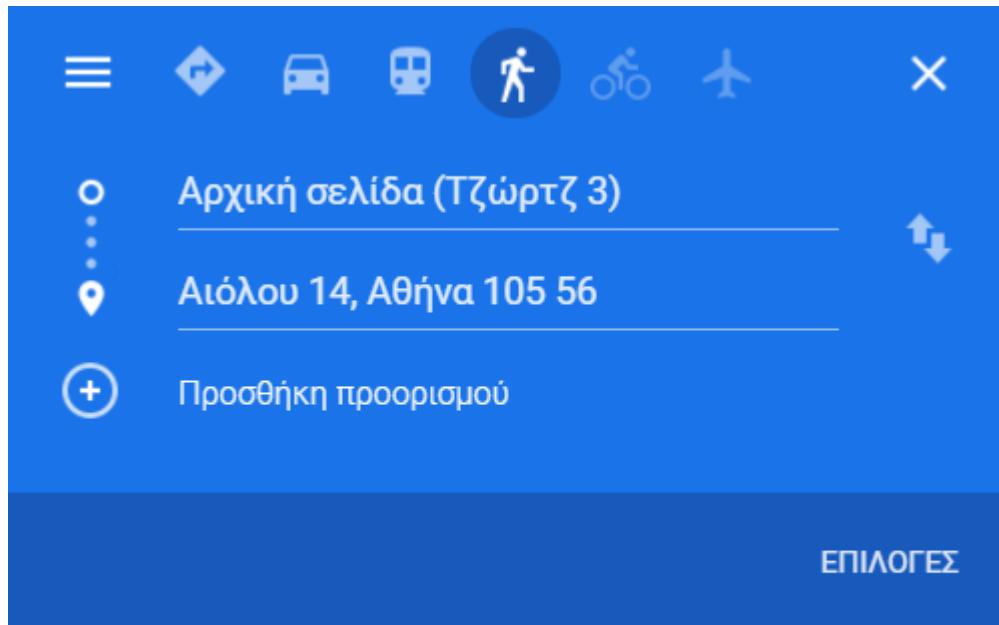
Στο παρακάτω παράδειγμα ζητάμε χωρίς να έχουμε κάνει login να μας βρεί η βάση το κοντινότερο μαγαζί που να έχει tofu.

The screenshot shows a search interface for "BIO products". The search term "tofu" is entered in the search bar. The results list includes various items such as "διατροφική μαγιά tofu", "mix ξηρών καρπών", "κόκκινες φακές", "μύρες φακές", "ΑΡΚΕΥΘΟΣ ΚΑΡΠΟΣ", "ΓΛΥΚΑΝΙΣΟ", "ΒΑΛΣΑΜΟ", "ΒΑΝΙΛΙΑ αρωματική ύλη", "ΒΑΣΙΛΙΚΟΣ", "ΒΕΤΙΒΕΡΙΑ", "ΒΙΟΛΕΤΑ αρωματικό έλαιο", "ΓΑΡΔΕΝΙΑ αρωματικό έλαιο", "ΓΑΡΥΦΑΛΛΟ", "ΓΕΡΑΝΙΟ αρωματικό έλαιο", "ΓΙΑΣΕΜΙ αρωματικό έλαιο", "ΓΚΡΕΪΦΡΟΥΤ", "ΔΑΦΝΗ", "ΔΕΝΔΡΟΛΙΒΑΝΟ", and "ΔΥΟΣΜΟΣ". To the right of the results, there is a button labeled "διέυθυνση" and a box labeled "τζωρτζ 3" containing the text "Βιολογικό χωριό, Αιώλου 14, Αθήνα Απόσταση : 1.26 km.". A "Submit" button is also visible. The background of the interface features a landscape with a single tree in a green field under a blue sky with white clouds.

Δύο ήταν τα μαγαζία που διέθεταν το προϊόν και οι αποστάσεις φαίνονται παρακάτω.

Βιολογικό χωριό
Βιολογικό χωριό 1.2600317761853925 km
4 seasons bio
4 seasons bio 1.3988759883373736 km
MIN telika-----> Βιολογικό χωριό, Αιώλου 14, Αθήνα
Απόσταση : 1.26 km.

Το πρόγραμμα επιλέγει την μικρότερη απόσταση και την εμφανίζει ως αποτέλεσμα



Αποστολή οδηγιών στο τηλέφωνό σας



μέσω Αιόλου

17 λεπτ.

[ΛΕΠΤΟΜΕΡΕΙΕΣ](#)

1,4 χλμ.



μέσω Εμμανουήλ Μπενάκη και
Αιόλου

19 λεπτ.

1,5 χλμ.



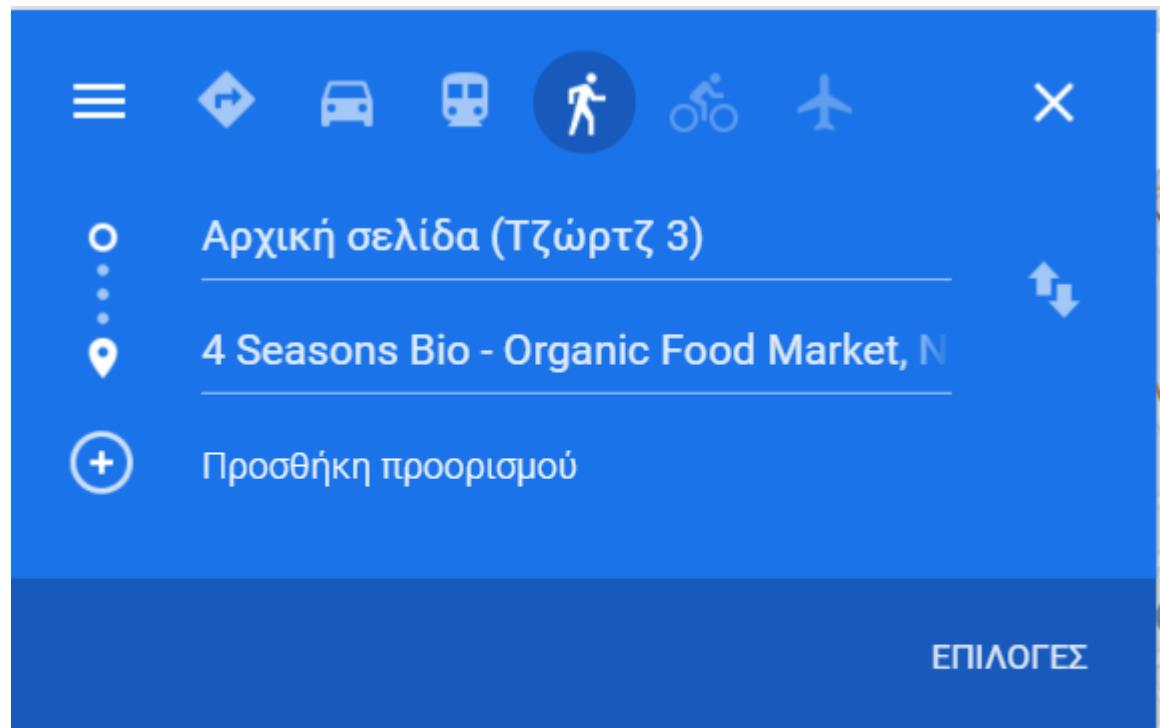
μέσω Καποδιστρίου και Αιόλου

20 λεπτ.

1,6 χλμ.

Όλες οι διαδρομές είναι κυρίως επίπεδες





	Αποστολή οδηγιών στο τηλέφωνό σας	
	μέσω Σταδίου	23 λεπτ.
	ΛΕΠΤΟΜΕΡΕΙΕΣ	1,8 χλμ.
	μέσω Ακαδημίας	23 λεπτ.
		1,8 χλμ.
	μέσω Ακαδημίας και Σταδίου	23 λεπτ.
		1,7 χλμ.
Όλες οι διαδρομές είναι κυρίως επίπεδες		

παρατηρούμε οτι η μικρή απόκλιση από το σωστό αποτέλεσμα δεν επηρεάζει το αποτέλεσμα. Στην συνέχεια βλέπουμε ένα παράδειγμα που ο χρήστης έχει συνδεθεί και του προτείνεται ένα προιόν

BIO products

Λογοτασμός:

Προϊόν | Διατροφική μαγιά
tofu
τούρι ξηρών καρπών
κόκκινες φραές
μάνικες φραές
ΑΡΚΕΥΘΩΣ ΚΑΡΠΟΣ
ΓΛΥΚΑΝΙΣΟ
ΒΑΛΣΑΜΟ
ΒΑΝΙΛΙΑ αρωματική θήλη
ΒΑΣΙΛΙΚΟΣ
ΒΕΤΙΒΕΡΙΑ
ΒΙΟΛΕΤΑ αρωματικό έλαιο
ΓΑΡΔΕΝΙΑ αρωματικό έλαιο
ΓΑΡΥΦΑΛΛΟ
ΓΕΡΑΝΙΟ αρωματικό έλαιο
ΓΙΑΣΕΜΙ αρωματικό έλαιο
ΓΚΡΕΙΠΦΡΟΥΤ
ΔΑΦΝΗ
ΔΕΝΔΡΟΛΙΒΑΝΟ
ΔΥΟΣΜΟΣ
ΕΛΑΤΟΥ ΒΕΛΟΝΑ
ΕΥΚΑΛΥΠΤΟΣ
ΘΥΜΑΡΙ
ΚΑΜΦΟΡΑ
ΚΑΝΕΛΛΑ αρωματικό έλαιο
ΚΑΡΔΑΜΟ

διένυθνυση Submit

Προτείνεται για εσάς

