ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2024-2025
Υποχρεωτική εργασία

Τα τελευταία χρόνια έχει παρατηρηθεί μεγάλη αύξηση των εφαρμογών online food delivery . Αυτά τα συστήματα συνήθως αποτελούνται από μια mobile frontend εφαρμογή που επιτρέπει στους πελάτες την προβολή των καταστημάτων, των προϊόντων που παρέχουν και την αγορά αυτών και επιτρέπει στους ιδιοκτήτες των καταστημάτων τη διαχείριση των καταστημάτων τους όπως προσθήκη νέων προϊόντων, αλλαγή τιμών και ορισμό προσφορών και εκπτώσεων. Επιπλέον, υπάρχει και ενα backend σύστημα το οποίο αναλαμβάνει την ανάλυση, επεξεργασία και αποθήκευση των δεδομένων. Γνωστές υπηρεσίες που υπάρχουν διαθέσιμες online είναι π.χ. το efood, wolt και BOX. Στα πλαίσια της εργασίας του μαθήματος καλείστε να δημιουργήσετε ένα απλό τέτοιο σύστημα online food delivery στο οποίο οι χρήστες θα μπορούν να εκτελούν λειτουργίες manager και πελάτη:

Στην λειτουργία manager θα πρέπει να μπορούν:

- Να προσθέτουν καταστήματα
- Να προσθέτουν/αφαιρούν διαθέσιμα προϊόντα
- Να προσθέτουν νέα προϊόντα ή να αφαιρούν παλιά
- Να εμφανίζουν τις συνολικές πωλήσεις ανα προϊόν
- Η διαχείριση μπορεί να γίνεται μέσω console application

Στη λειτουργία πελάτη θα πρέπει να μπορούν:

- Να προβάλλονται τα καταστήματα που είναι σε ακτίνα 5 km
- Να φιλτράρουν τα μαγαζιά ανάλογα
    - με τις κατηγορίες φαγητών που τους ενδιαφέρει
    - αστέρια
    - τιμές (3 κατηγορίες $,$$,$$$)
- Να πραγματοποιούν αγορά για τα προϊόντα που τους ενδιαφέρουν
- Να βαθμολογούν τα καταστήματα με αστέρια (1-5)
- Μέσω ενός UI σε Android να μπορεί να πραγματοποιήσει τις παραπάνω ενέργειες
    - Μια συνάρτηση search() θα στέλνει τα φίλτρα στο Master(συντεταγμένες πελάτη,κατηγορίες φαγητών,αστέρια και τιμες) ασύγχρονα και θα εμφανίζει στην οθόνη της εφαρμογής τα αποτελέσματα της αναζήτησης.
    - Μέσω μιας συνάρτησης buy() θα μπορεί ο χρήστης να πραγματοποιήσει αγορά για ένα ή περισσότερα προϊόντα από τα καταστήματα που έχει επιστρέψει η search().

Προκειμένου το σύστημα να μπορεί να διαχειριστεί τον όγκο των δεδομένων, θα πρέπει να μπορεί να τρέξει κατανεμημένα πάνω από ένα σύνολο μηχανημάτων.

Το MapReduce framework είναι ένα προγραμματιστικό μοντέλο που επιτρέπει την παράλληλη επεξεργασία μεγάλων όγκων δεδομένων.

Το MapReduce στηρίζεται στη χρήση δύο συναρτήσεων:

map(key,value) -> [(key2, value2)]

reduce(key2,[value2]) -> [final_value]

- "Map" συνάρτηση: επεξεργάζεται ένα ζεύγος key/value και παράγει ένα ενδιάμεσο ζεύγος key/value. Η είσοδος στη συνάρτηση map μπορεί να είναι γραμμές ενός αρχείου κλπ., και έχουν τη μορφή (κλειδί, τιμή). Η συνάρτηση map μετατρέπει κάθε τέτοιο ζευγάρι σε ένα άλλο ζευγάρι (κλειδί2, τιμή2). Η map συνάρτηση μπορεί να εκτελείται παράλληλα, πάνω σε διαφορετική είσοδο δεδομένων και σε διαφορετικούς κόμβους. Ο βαθμός παραλληλίας εξαρτάται από την εφαρμογή και μπορεί να την ορίσει ο χρήστης.
- "Reduce" συνάρτηση: συγχωνεύει όλα τα ενδιάμεσα values που σχετίζονται με το ίδιο κλειδί και παράγει τα τελικά αποτελέσματα. Για κάθε ξεχωριστό κλειδί δημιουργείται μια λίστα από τις τιμές που αντιστοιχούν σε αυτό το κλειδί. Η συνάρτηση αυτή υπολογίζει μια τελική τιμή για το κλειδί, επεξεργάζοντας τη λίστα των τιμών που αντιστοιχούν σε αυτό το κλειδί. Η επεξεργασία της συνάρτησης reduce γίνεται αφού έχει τελειώσει η επεξεργασία όλων των map συναρτήσεων.

Αρχικά, όταν ένας manager επιθυμεί να προσθέσει ένα κατάστημα στην πλατφόρμα, θα πρέπει να δώσει την περιγραφή του καταστήματος με μορφή αρχείου json, όπως επισυνάπτεται, όπως επίσης ένα λογότυπο. Μπορείτε να έχετε ένα φάκελο που να περιέχει το json και το λογότυπο. Το path του λογότυπου θα περιέχεται μέσα στο json. Το json επίσης θα περιέχει ένα αριθμό από review και τη βαθμολογία του καταστήματος (1-5). Τέλος θα περιέχει και έναν πίνακα από τα προϊόντα που παρέχει και τις τιμές τους. Η κατηγορία της ακρίβειας που είναι το κατάστημα θα εξαρτάται από τον μέσο όρο των τιμών των προϊόντων και θα προκύπτει ως εξής:

1. Μέσος όρος τιμών μέχρι και 5€ -> $
2. Μέσος όρος τιμών μέχρι και 15€ -> $$
3. Μέσος όρος τιμών πάνω από 15€ -> $$$

Η κατηγορία της ακρίβειας **ΔΕΝ** θα υπάρχει στο json και θα υπολογίζεται από το σύστημα.
Ο manager θα μπορεί να προσθέσει ή να αφαιρέσει προϊόντα σε υπάρχοντα καταστήματα. Σε περίπτωση αφαίρεσης προϊόντος, το προϊόν απλά δεν θα εμφανιζεται στον πελάτη αλλά τα προΪόντα θα εμφανίζονται κανονικά στα manager queries.
Επιπλέον θα μπορεί να αυξάνει ή να μειώνει το απόθεμα από υπάρχοντα προϊόντα.

Η αρχική επικοινωνία του console app του manager θα πρέπει να γίνεται με τον Master μέσω της οποίας θα αποστέλλονται όλα τα στοιχεία του καταστήματος. Ο Master αφού λάβει τα στοιχεία, μέσω μιας hash συνάρτησης H(storeName) θα πρέπει να επιλέξει τον worker node στον οποίο θα αποθηκευτεί το κατάστημα. Π.χ. NodeId = H(storeName) mod NumberOfNodes. Αφού επιλέξει τον κόμβο αποστέλλει τις πληροφορίες σε αυτόν. **Για τα πλαίσια της εργασίας ο Worker θα αποθηκεύει τις πληροφορίες στις κατάλληλες**

δομές δεδομένων στην μνήμη του. *Δεν επιτρέπεται η αποθήκευση στο δίσκο* (εκτός από τις φωτογραφίες αν επιθυμείτε).

Όταν ένας χρήστης επιθυμεί να αγοράσει προϊόντα αρχικά μέσω της εφαρμογής θα πρέπει να επιλέξει τα κατάλληλα φίλτρα για τα καταστήματα που επιθυμεί. Τότε αποστέλλεται ένα request προς τον Master που περιέχει τα φίλτρα. **Ο Master θα πρέπει με διαδικασία MapReduce να επιστρέψει στον χρήστη τα καταστήματα που ικανοποιούν τα κριτήρια.**

Αφού προβληθούν στον χρήστη, εκείνος θα μπορεί να δει αναλυτικά τα καταστήματα, τα προϊόντα που παρέχουν, τις τιμές και να προβεί σε αγορά εάν το επιθυμεί.

Όταν ο χρήστης πραγματοποιήσει αγορά, στέλνει ένα request αγοράς στον Master για το συγκεκριμένο κατάστημα και ο Master με τη σειρά του πρέπει να ενημερώσει κατάλληλα τον Worker που διαχειρίζεται το κατάστημα για την αγορά. Θα πρέπει να προσθέσει στο κατάστημα αυτό τα προϊόντα και την ποσότητα που αγόρασε, και το συνολικά έσοδα από την αγορά αυτή. **Προσοχή θα πρέπει να ληφθούν μέτρα ώστε να καταμετρώνται σωστά πιθανές ταυτόχρονες αγορές ιδίων προϊόντων στο ίδιο κατάστημα και να γίνεται σωστή ενημέρωση των εσόδων.**


```
{
 "StoreName": "storeName",
 "Latitude": 37.9932963,
 "Longitude": 23.733413,
 "FoodCategory": "pizzeria",
 "Stars": 3,
 "NoOfVotes": 15,
 "StoreLogo": "/usr/bin/images/storeLogo.png",
 "Products": [
        {
        "ProductName": "margarita",
        "ProductType": "pizza",
        "Available Amount": 5000,
        "Price": 9.2
        },
        {
        "ProductName": "special",
        "ProductType": "pizza",
        "Available Amount": 1000,
        "Price": 12
        },
        {
        "ProductName": "chef's Salad",
        "ProductType": "salad",
        "Available Amount": 100,
        "Price": 5
        }
```

]
}

Τέλος στην λειτουργία manager, θα πρέπει να μπορούν να εμφανιστούν οι συνολικές πωλήσεις ανα τύπο καταστημάτων και συνολικές πωλήσεις ανα κατηγορία προϊόντος

Π.χ με input για FoodCategory: pizzeria
output: "Pizza Fun": 100,
"Pizza Hat": 50,
"total": 150

Π.χ με input για ProductCategory: salad
output: "Pizza Fun": 10,
"Pizza Hat": 5,
"salad minus": 75,
"total": 90

Απαιτήσεις υλοποίησης Backend:

- Ο **Master** πρέπει να υλοποιηθεί σε **Java** και να υλοποιεί **TCP Server**. **Δεν επιτρέπεται** η χρήση έτοιμων libraries πέρα των default ServerSocket της Java ή HTTP πρωτοκόλλου με τη χρήση έτοιμου server, όπως της Java ή Apache.

- Ο **Master** πρέπει να είναι **πολυνηματικός** και να μπορεί να εξυπηρετεί πολλούς χρήστες ταυτόχρονα και να επικοινωνεί ταυτόχρονα με τους workers.

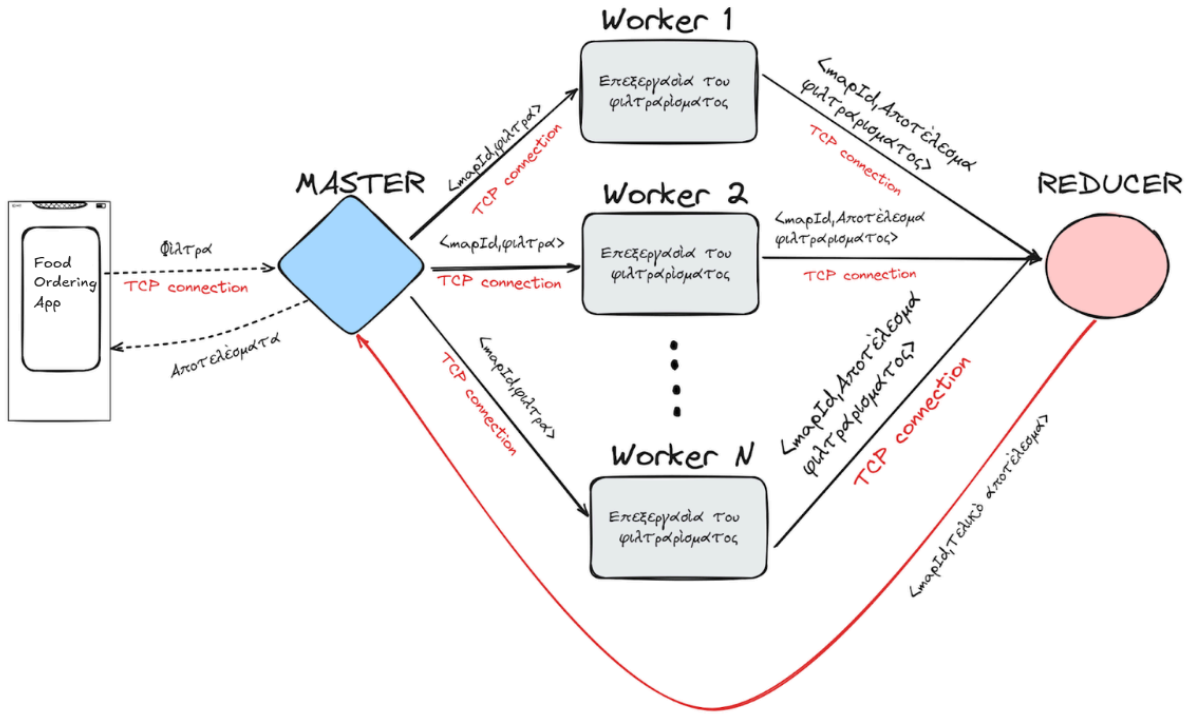
- Οι **Workers** πρέπει να υλοποιηθούν σε **Java** και να είναι **πολυνηματικοί** για να εκτελούν παράλληλα πολλά requests από τον Master.

- Οι **Workers** θα πρέπει να ορίζονται δυναμικά κατά το initialization του Master (από τα arguments ή config file) και ο αριθμός τους θα μπορεί να είναι αυθαίρετος.

- Η επικοινωνία Master / Worker να υλοποιείται και αυτή **αποκλειστικά μέσω TCP sockets**.

- Πρέπει να υπάρχει συγχρονισμός στα σημεία που κρίνετε απαραίτητο. Ο συγχρονισμός πρέπει να γίνει **αποκλειστικά** χρησιμοποιώντας τεχνικές **synchronized, wait - notify** και **όχι με τη χρήση έτοιμων εργαλείων της βιβλιοθήκης *java.util.concurrent* ή άλλων έτοιμων εργαλείων.**

- Οι δομές δεδομένων πρέπει να είναι αποθηκευμένες στην μνήμη. **Απαγορεύεται η χρήση βάσης δεδομένων.**



Απαιτήσεις υλοποίησης Frontend:

Θα αναπτύξετε μια εφαρμογή που θα εκτελείται σε συσκευές με λειτουργικό Android και θα αποτελεί interface για το σύστημα. Μέσω αυτής ο χρήστης:

- Η επικοινωνία του Application με τον Master θα πρέπει να γίνεται αποκλειστικά με τη χρήση **TCP Sockets**. Το Application θα πρέπει **να συνδέεται με TCP Socket στον Master**. Μέσω αυτού του Socket γίνεται η αποστολή των φίλτρων και της αίτησης για αγορά. Ο Master αποστέλλει πίσω τα αποτελέσματα της επεξεργασίας **μέσω του ίδιου Socket** το οποίο παραμένει ανοιχτό έως ότου αυτά ληφθούν. **Αυτή η διαδικασία θα πρέπει να υλοποιηθεί με τη χρήση Threads, ώστε η εφαρμογή να παραμένει διαδραστική μέχρι να ληφθούν τα αποτελέσματα**, **δηλαδή η αποστολή γίνεται ασύγχρονα**.

<u>Bonus</u>

Μια γνωστή τεχνική προκειμένου η πλατφόρμα να είναι ανθεκτική σε σφάλματα είναι το active replication. Σε αυτή την τεχνική τα δεδομένα των καταστημάτων θα πρέπει να βρίσκονται ταυτόχρονα σε πολλαπλούς κόμβους και να παραμένουν απολύτως συγχρονισμένα. Σε περίπτωση που κάποιος κόμβος worker πέσει δεν θα πρέπει να χαθεί η πρόσβαση στα δεδομένα που διαχειρίζεται (θα πρέπει να γίνει δρομολόγηση του request στο replica του). Επίσης τα back up δεδομένα θα πρέπει να χρησιμοποιούνται μόνο όταν υπάρχει σφάλμα στον κανονικό κόμβο.

Το Bonus μετράει +20% στον βαθμό της εργασίας <u>με μέγιστο δυνατό βαθμό εργασίας 10</u> και είναι υποχρεωτικό για ομάδες των 4 ατόμων.

Παραδοτέα εργασίας

Το project θα παραδοθεί σε **δύο** φάσεις:

Παραδοτέο Α: (Ημερομηνία παράδοσης: 28/04/2024)

Στο παραδοτέο αυτό, θα πρέπει να έχετε ολοκληρώσει εντελώς το backend σύστημα και το manager console app, όπως ακριβώς σας έχει ζητηθεί, έτσι ώστε να μπορεί να χρησιμοποιηθεί στην επόμενη φάση της εργασίας του μαθήματος (προσθήκη καταστημάτων, ενημέρωση καταστημάτων). Για τον πελάτη αντί για android application θα έχετε μια dummy εφαρμογή όπου θα στέλνει το τα φίλτρα στο Master και θα λαμβάνει τα αντίστοιχα αποτελέσματα. Επίσης θα πρέπει ο χρήστης να μπορεί να κάνει αγορά από την dummy εφαρμογή.

Παραδοτέο Β: (Ημερομηνία παράδοσης: 26/05/2024)

Το παραδοτέο αυτό αποτελεί το Android application, που περιγράφηκε παραπάνω. Στη φάση αυτή το σύστημα θα πρέπει να είναι πλήρως λειτουργικό και ολοκληρωμένο, με όλα τα components του να λειτουργούν σωστά. Σε αυτή τη φάση το manager console app θα πρέπει να μπορεί να εκτελέσει και τα aggregation queries για τις **συνολικές πωλήσεις ανά τύπο καταστημάτων και συνολικές πωλήσεις ανά κατηγορία προϊόντος.**

Ομάδες: Όλοι οι φοιτητές θα πρέπει να σχηματίσουν ομάδες των τριών (3) ατόμων προκειμένου να εκπονήσουν την προγραμματιστική τους εργασία. Γλώσσα προγραμματισμού θα είναι η Java, στην οποία και θα παρέχεται υποστήριξη από τους βοηθούς του μαθήματος.

Αναφορές – Χρήσιμοι Σύνδεσμοι:

[1] https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html
[2] Android. URL : http://code.google.com/android/
[3] Android SDK: http://developer.android.com/sdk/index.html
[4] Android Studio http://developer.android.com/sdk/index.html