ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2025-2026
Υποχρεωτική εργασία

Τα τελευταία χρόνια έχει παρατηρηθεί μεγάλη αύξηση των εφαρμογών online τυχερών παιχνιδιών. Αυτά τα συστήματα συνήθως αποτελούνται από μια mobile frontend εφαρμογή που επιτρέπει στους παίκτες την προβολή των διαθέσιμων παιχνιδιών , των κατηγοριών τους και το ποντάρισμα σε αυτά, ενώ επιτρέπει στους διαχειριστές (managers) τη διαχείριση των παιχνιδιών τους όπως προσθήκη νέων παιχνιδιών, επεξεργασία αυτών και ανάλυση και αποθήκευση των ιστορικών στατιστιστικών τους.

Στα πλαίσια της εργασίας του μαθήματος καλείστε να δημιουργήσετε ένα απλό τέτοιο σύστημα online παιχνιδιών στο οποίο οι χρήστες θα μπορούν να εκτελούν λειτουργίες manager και παίκτη:

Στην λειτουργία manager θα πρέπει να μπορούν:

- Να προσθέτουν/αφαιρούν διαθέσιμα παιχνίδια.
- Να τροποποιούν τα διαθέσιμα παιχνίδια.
- Να εμφανίζουν τα συνολικά κέρδη/ζημιές ανά παιχνίδι.
- Να εμφανίζουν τα κέρδη/ζημιές ανά παίκτη.
- Η διαχείριση μπορεί να γίνεται μέσω console application.

Στη λειτουργία παίκτη θα πρέπει να μπορούν:

- Να προβάλλονται τα παιχνίδια που είναι διαθέσιμα στην πλατφόρμα.
- Να φιλτράρουν τα παιχνίδια ανάλογα με τις κατηγορίες που τους ενδιαφέρουν
    - αστέρια (δημοτικότητα παιχνιδιού).
    - όρια ποντάρίσματος (3 κατηγορίες -> $, $$, $$$).
    - επίπεδο ρίσκου (3 κατηγορίες -> low, medium, high).
- Να παίζουν στα παιχνίδια που τους ενδιαφέρουν.
- Να βαθμολογούν τα παιχνίδια 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 (π.χ. ID παίκτη, κατηγορίες παιχνιδιών, αστέρια, επίπεδο ρίσκου και όρια ποντάρισματος) ασύγχρονα και θα εμφανίζει στην οθόνη της εφαρμογής τα αποτελέσματα της αναζήτησης.
- Μέσω μιας συνάρτησης play() θα μπορεί ο χρήστης να παίζει σε ένα από τα παιχνίδια που έχει επιστρέψει η search() αφού βάλει το ποντάρισμα που θέλει.
- Μέσω μιας συνάρτησης addBalance() θα μπορεί να προσθέσει tokens

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 παιχνίδι στην πλατφόρμα, θα πρέπει να δώσει την περιγραφή του παιχνιδιού 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 παιχνιδιού (1-5).

```
{
"GameName":"gameName",
"ProviderName": "providerId",
"Stars": 3,
"NoOfVotes": 15,
"GameLogo": "/usr/bin/images/gameLogo.png",
"MinBet": 0.1,
"MaxBet": 10,
"RiskLevel": "low",
“HashKey”: ”key”
}
```

Η κατηγορία πονταρίσματος που έχει ένα παιχνίδι θα εξαρτάται από τον ελάχιστο ποντάρισμα και θα προκύπτει ως εξής:

1. Ελάχιστο ποντάρισμα 0.1 FUN -> $
2. Ελάχιστο ποντάρισμα 1 FUN -> $$
3. Ελάχιστο ποντάρισμα 5 FUN-> $$$

Η κατηγορία πονταρίσματος, καθώς και το Jackpot του παιχνιδιού, **ΔΕΝ** θα υπάρχουν στο json, αλλά θα υπολογίζονται και θα ορίζονται αυτόματα από το σύστημα.

Ο manager θα μπορεί να προσθέσει ή να αφαιρέσει παιχνίδια σε υπάρχοντες providers. Σε περίπτωση αφαίρεσης παιχνιδιού, το παιχνίδι απλά δεν θα εμφανίζεται στον παίκτη, αλλά τα στατιστικά του (κέρδη/ζημιές) θα εμφανίζονται κανονικά στα manager queries. Επιπλέον, θα μπορεί να αλλάζει το επίπεδο ρίκου σε υπάρχοντα παιχνίδια.

Η αρχική επικοινωνία του console app του manager θα πρέπει να γίνεται με τον Master, μέσω της οποίας θα αποστέλλονται όλα τα στοιχεία του παιχνιδιού. Ο Master, αφού λάβει τα στοιχεία, μέσω μιας hash συνάρτησης H(GameName) θα πρέπει να επιλέξει τον worker node στον οποίο θα αποθηκευτεί το παιχνίδι. Π.χ. NodeId = H(GameName) 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 τα λογότυπα των παιχνιδιών αν επιθυμείτε).

Διαδικασία Ποντάρίσματος και Υπολογισμός Κέρδους

Όταν ένας χρήστης επιθυμεί να παίξει, αρχικά μέσω της εφαρμογής θα πρέπει να επιλέξει τα κατάλληλα φίλτρα για τα παιχνίδια 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 παιχνίδια που ικανοποιούν τα κριτήρια.

Αφού προβληθούν στον χρήστη, εκείνος θα μπορεί να δει αναλυτικά τα παιχνίδια, τα όρια ποντάρίσματος, το επίπεδο ρίσκου και το Jackpot, και να προβεί σε ποντάρισμα εάν το επιθυμεί.

Όταν ο χρήστης πραγματοποιήσει ποντάρισμα, στέλνει ένα request παιχνιδιού στον Master για το συγκεκριμένο παιχνίδι και ο Master με τη σειρά του πρέπει να ενημερώσει κατάλληλα τον Worker που διαχειρίζεται το παιχνίδι για την ενέργεια.

Σχετικά με τα επίπεδα ρίσκου (Low, Medium, High) και τον υπολογισμό του κέρδους ανά ποντάρισμα, το σύστημα χρησιμοποιεί τρεις προκαθορισμένους πίνακες πολλαπλασιαστών και Jackpot. Κάθε παιχνίδι, ανάλογα με το ρίσκο του, αντιστοιχεί σε έναν από τους παρακάτω πίνακες:

- **Low:** [0.0 , 0.0 , 0.0 , 0.1 , 0.5 , 1.0 , 1.1 , 1.3 , 2.0 , 2.5] (Jackpot: 10)
- **Medium:** [0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.5 , 1.0 , 1.5 , 2.5 , 3.5] (Jackpot: 20)
- **High:** [0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 2.0 , 6.5] (Jackpot: 40)

Κατά την εκτέλεση ενός ποντάρίσματος, ο εκάστοτε Worker καλεί μια ασφαλή γεννήτρια τυχαίων αριθμών (Secured Random Generator) η οποία εκτελείται σε ένα ξεχωριστό μηχάνημα και η επικοινωνία γίνεται μέσω TCP. Κάθε παιχνίδι έχει μια δική του γεννήτρια τυχαίων αριθμών. Επειδή κάθε φορά που δημιουργείται ένας τυχαίος αριθμός υπάρχει μια μικρή καθυστέρηση, για αυτό το λόγο καλείστε να υλοποιήσετε ένα μοντέλο producer-consumer. Σε αυτό το μοντέλο, ο producer (γεννήτρια τυχαίων αριθμών) δημιουργεί συνεχώς τυχαίους αριθμούς και τους αποθηκεύει σε ένα buffer(πχ. Queue) μέχρις ότου ο buffer δεν έχει άλλο διαθέσιμο χώρο. Στη συνέχεια ο consumer (παιχνίδι) κάνει polling από τον buffer τον πρώτο διαθέσιμο αριθμό. Προκειμένου η επικοινωνία να είναι ασφαλής (μεταξύ του game και του server), ο Worker και η γεννήτρια μοιράζονται ένα κοινό secret S το οποίο δηλώνεται μέσα στο json εγγραφής του παιχνιδιού . Η γεννήτρια μαζί με τον τυχαίο αριθμό στέλνει και το sha256(αριθμός+secret). Στη συνέχεια ο Worker αφού λάβει τον αριθμό και το hash επαναλαμβάνει την ίδια πράξη (sha256) και ελέγχει αν τα δύο hashes είναι ίδια.

Μόλις ο Worker λάβει τον τυχαίο ακέραιο αριθμό, υπολογίζει αρχικά το υπόλοιπο της διαίρεσής του με το 100. Εάν το αποτέλεσμα είναι ακριβώς 0, ο παίκτης κερδίζει το

Jackpot που έχει οριστεί για το παιχνίδι πολλαπλασιάζοντας το ποσό του πονταρίσματός του με την τιμή αυτή. Σε κάθε άλλη περίπτωση (εάν δεν είναι 0), ο Worker υπολογίζει το υπόλοιπο της διαίρεσής του με το 10. Το κέρδος/ζημιά του παίκτη σε αυτή την περίπτωση προκύπτει πολλαπλασιάζοντας το ποσό του πονταρίσματός του με τον συντελεστή που βρίσκεται στη θέση αυτού του δείκτη, στον πίνακα ρίσκου του παιχνιδιού.

Στη συνέχεια, θα πρέπει να καταγραφεί το ποντάρισμα και να ενημερωθούν τα συνολικά έσοδα/ζημιές του συστήματος από το παιχνίδι αυτό. **Προσοχή:** Θα πρέπει να ληφθούν μέτρα ώστε να καταμετρώνται σωστά πιθανά ταυτόχρονα πονταρίσματα στο ίδιο παιχνίδι από διαφορετικούς παίκτες και να γίνεται σωστή ενημέρωση των υπολοίπων.

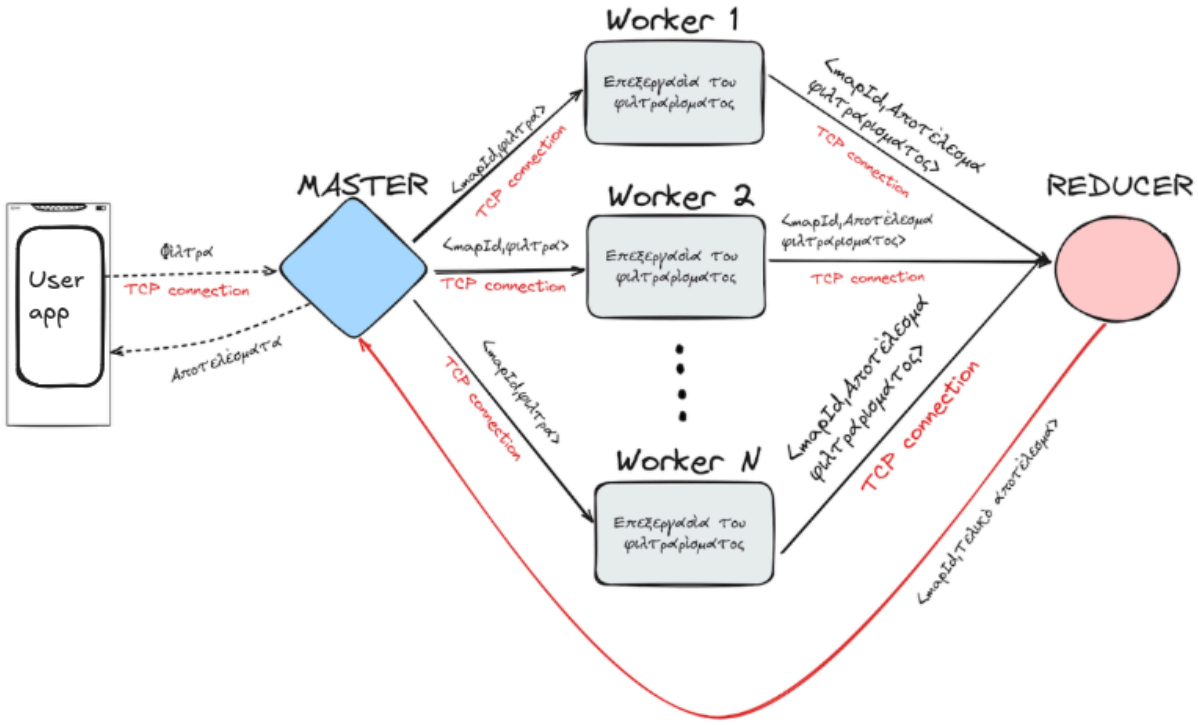
Τέλος, στην λειτουργία manager, θα πρέπει να μπορούν να εμφανιστούν τα συνολικά κέρδη/ζημιές ανά πάροχο και τα συνολικά κέρδη/ζημιές ανά παίκτη . Αυτοί οι υπολογισμοί θα πρέπει να υλοποιηθούν υποχρεωτικά με τη χρήση της διαδικασίας **MapReduce**.

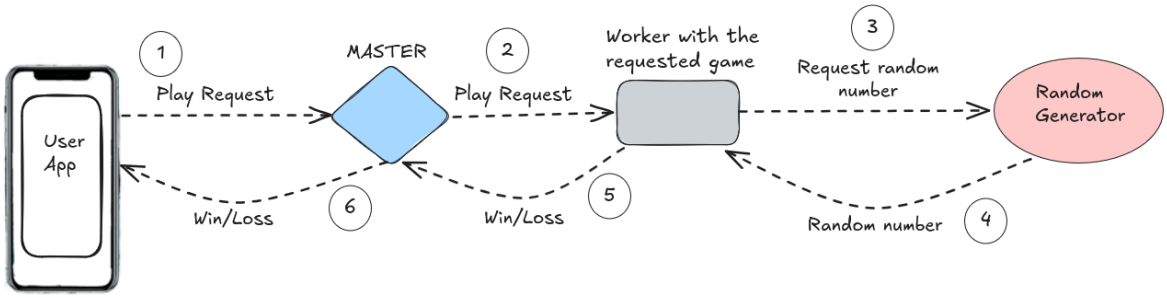
- Π.χ. με input για Provider: "provider1" -> output: "game1": +1000, "game2": -50, "Total": +950 FUN
- Π.χ. με input για Player: "user123" -> output: "Total Profit/Loss": -200 FUN

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.
- Η Ασφαλής Γεννήτρια Τυχαίων Αριθμών πρέπει να υλοποιηθεί σε Java ως ένας ξεχωριστός, πολυνηματικός TCP Server, ο οποίος θα μπορεί να εξυπηρετεί πολλούς Workers ταυτόχρονα. Κάθε φορά που απαιτείται ο υπολογισμός ενός αποτελέσματος πονταρίσματος, ο εκάστοτε Worker θα επικοινωνεί με τον SRG server αποκλειστικά μέσω TCP Sockets για να λάβει τον τυχαίο αριθμό.
- 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.

- Ο Reducer πρέπει να υλοποιηθεί σε Java και να είναι πολυνηματικός για να εκτελεί παράλληλα πολλά requests από τους Worker.
- Η επικοινωνία Master / Worker, Workers/Reducer, Reducer/Master και Worker/Random Generator να υλοποιείται 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.
- 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 ποντάρισμα.
- 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.

Bonus

Μια γνωστή τεχνική προκειμένου η πλατφόρμα να είναι ανθεκτική σε σφάλματα είναι το active replication. Σε αυτή την τεχνική τα δεδομένα των παιχνιδιών 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 με μέγιστο δυνατό βαθμό εργασίας 10 και είναι υποχρεωτικό για ομάδες των 4 ατόμων.

Παραδοτέα εργασίας

Το project θα παραδοθεί σε δύο φάσεις:

- **Παραδοτέο Α: (Ημερομηνία παράδοσης: 03/04/2026)** 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 παιχνιδιών, ενημέρωση στατιστικών). Για τον παίκτη αντί για android application θα έχετε μια dummy εφαρμογή όπου θα στέλνει τα φίλτρα στο Master και θα

λαμβάνει τα αντίστοιχα αποτελέσματα. Επίσης θα πρέπει ο χρήστης να μπορεί να κάνει ποντάρισμα από την dummy εφαρμογή.

- **Παραδοτέο Β: (Ημερομηνία παράδοσης: 10/05/2026)** 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 τα συνολικά κέρδη ανά Πάροχο και ανά Παίκτη .

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