ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2019-2020

Υποχρεωτική εργασία

Τα τελευταία χρόνια, η εξέλιξη των μέσων αναπαραγωγής μουσικής έχει οδηγήσει στην ανάπτυξη υπηρεσιών συνδρομητικού πολυμεσικού περιεχομένου. Υπηρεσίες όπως το Spotify, το i-tunes, το Amazon Music, το Google Music, διάφορα pocasts, κλπ. έχουν κάνει την εμφάνισή τους, παρέχοντας τη δυνατότητα σε μεγάλο πλήθος χρηστών να ακούν τα αγαπημένα τους κομμάτια, για ένα συγκεκριμένο συνδρομητικό αντίτιμο. Ένα σημαντικό χαρακτηριστικό και πλεονέκτημα αυτών των συστημάτων είναι η δυνατότητα ακρόασης της μουσικής, είτε σε πραγματικό χρόνο (real-time streaming) είτε σε offline mode (στην περίπτωση μη διαθεσιμότητας δικτύου). Ως γνωστόν, τα περισσότερα από αυτά τα συστήματα διαθέτουν δύο τύπους λειτουργίας, έναν ελεύθερο και έναν υπό συνδρομή.

Τα συστήματα αυτά, έχουν την ικανότητα να εξυπηρετούν ένα μεγάλο αριθμό χρηστών. Αυτό σημαίνει ότι κάθε καλλιτέχνης διατηρεί ένα “κανάλι”, δηλαδή, ένα χώρο όπου μπορεί να προσθέτει την δουλειά του (album, cd single, κλπ.) τα οποία με τη σειρά τους μπορούν να ακούσουν διάφοροι ακροατές μέσω του δικτύου. Έτσι, κάθε φορά που ο εκάστοτε καλλιτέχνης ανεβάζει μια νέα δουλειά του στην πλατφορμα, αυτή γίνεται διαθέσιμη σε όλους τους χρήστες που ακολουθούν τον συγκεκριμένο καλλιτέχνη, είτε για να την ακούσουν κατευθείαν online είτε για να την αποθηκεύσουν σε μια offline playlist που επιτρέπει την ακρόαση της δουλειάς του αργότερα.

Στην παρούσα εργασία καλείστε να φτιάξετε ένα τέτοιο music streaming on demand σύστημα με χρήση της java 8, στο οποίο η πληροφορία που θέλουμε να μεταδώσουμε είναι μουσική από έναν ή παραπάνω καλλιτέχνες σε ένα πλήθος από πολλούς ακροατές. Λόγω του δεδομένου μεγάλου αριθμού των ακροατών που θέλουμε να εξυπηρετήσουμε, χρειάζεται να υλοποιήσουμε ένα έξυπνο σύστημα στο οποίο, δεδομένου ενός συνόλου από καλλιτέχνες που τους ενδιαφέρουν, θα τους δίνει την δυνατότητα να ακούσουν σε πραγματικό χρόνο είτε offline τη μουσική. Η πληροφορία αυτή θα μεταφέρεται κατάλληλα από το σύστημα που καλείστε να υλοποιήσετε έτσι ώστε οι χρήστες να μπορούν να ακούσουν κομμάτια σε πραγματικό χρόνο αλλά και να επιτρέπεται η αποθήκευσή της στο κινητό των χρηστών που τη ζητούν στην περίπτωση

μιας offline λειτουργίας, επιτρέποντας τους να έχουν μια playlist από κομμάτια τα οποία θα μπορούν να ακούσουν.

Η ανάπτυξη του συστήματος θα σας επιτρέψει να εξοικειωθείτε τόσο με την έννοια ενός τέτοιου music streaming on demand συστήματος, όσο και με το να γνωρίσετε το περιβάλλον Android. Για την διευκόλυνσή σας, η ανάπτυξη αυτού του συστήματος θα γίνει σε δύο φάσεις:

- Η πρώτη φάση αφορά ***την υλοποίηση του music streaming Framework (Event Delivery System) που θα είναι υπεύθυνο να υποστηρίξει την προώθηση και λήψη (streaming) των μουσικών δεδομένων***.
- Η δεύτερη φάση αφορά ***τη δημιουργία μιας εφαρμογής Android, η οποία μέσω των κατάλληλων διεπαφών που θα έχετε φροντίσει να υλοποιήσετε στην πρώτη φάση, θα μπορεί να αξιοποιήσει το framework, έτσι ώστε να εκτελείται ο αλγόριθμος που θα μεταφέρει την πληροφορία από τους καλλιτέχνες προς τους αντίστοιχους ακροατές, και στη συνέχεια τα μουσικά κομμάτια να μπορούν να αναπαραχθούν στο κινητό***.

Παρακάτω ακολουθούν λεπτομερείς οδηγίες για την κάθε φάση και τι καλείστε να υλοποιήσετε σε αυτές.

Event Delivery System

Το Event Delivery System μοντέλο είναι ένα προγραμματιστικό framework που επιτρέπει την αποστολή και λήψη δεδομένων που πληρούν συγκεκριμένα κριτήρια. Το πλεονέκτημα του Event Delivery System συστήματος είναι ότι επιτρέπει την άμεση αποστολή και προώθηση των δεδομένων σε πραγματικό χρόνο μέσω δύο βασικών συναρτήσεων, την “push” και την “pull”. Αυτές οι δύο συναρτήσεις είναι ανεξάρτητες η μια με την άλλη. Σε κάθε κλήση της “push” συνάρτησης, θα πρέπει να υπάρξει κατάλληλη μέριμνα ώστε ο ενδιάμεσος κόμβος στο σύστημα, που ονομάζεται broker και μπορεί να δέχεται ταυτόχρονα δεδομένα από διαφορετικούς publishers (στην περίπτωσή μας καλλιτέχνες), να μπορεί να μεταφέρει κατάλληλα τα αποτελέσματα στους τελικούς ακροατές (οι οποίοι ονομάζονται και consumers, λόγω του ότι “καταναλώνουν” την πληροφορία). Επίσης, ο παραλληλισμός είναι απαραίτητος γιατί το σύστημα **επιβάλλεται** να προσφέρει δυνατότητα ταυτόχρονης αποστολής των δεδομένων τόσο από τους καλλιτέχνες στους ενδιάμεσους κόμβους, όσο και από τους ενδιάμεσους κόμβους στους ακροατές.

Το Event Delivery System μοντέλο, όπως προαναφέρθηκε, στηρίζεται στη χρήση δύο συναρτήσεων:

push(topic,value) -> [broker]
pull(topic,[broker]) -> [topic,value]

Παρακάτω ακολουθεί μια γενική περιγραφή αυτών των δύο συναρτήσεων:

● "Push" συνάρτηση: Ο ρόλος της “push()” συνάρτησης είναι να προωθήσει στον ενδιάμεσο κόμβο ένα ζεύγος key(ή αλλιώς topic)/value το οποίο και αποθηκεύεται σε κατάλληλη ενδιάμεση δομή(πχ. queue) ώστε να μπορεί να αναζητηθεί. Στην περίπτωσή μας, η συνάρτηση “push” μπορεί να πάρει σαν είσοδο ό,τι πληροφορία απαιτείται για να μεταφερθεί από το κανάλι του καλλιτέχνη (πχ. πληροφορίες του album, ονομα καλλιτέχνη, τίτλος τραγουδιού το οποίο γίνεται streamed κ.οκ.), καθώς και τα αντίστοιχα ηχητικά δεδομένα (πχ. το κομμάτι του τραγουδιού το οποίο γίνεται streamed). Μια πολύ σημαντική παράμετρος που πρέπει να λάβουμε υπόψιν μας κατά την ανάπτυξη του συστήματος είναι: Ένα κομμάτι το οποίο γίνεται streamed από το framework ΠΟΤΕ δεν αποστέλλεται ολόκληρο, αλλά σε μικρά ισομεγέθη κομμάτια, ώστε να γίνεται αποτελεσματικά η μεταφορά των δεδομένων (communication efficiency), και αυτό είναι μια σημαντική λειτουργικότητα που θα πρέπει να εξασφαλίσετε κατά την ανάπτυξη του project. Επιπλέον, η push συνάρτηση θα πρέπει να μπορεί να εκτελεστεί παράλληλα, παρέχοντας ουσιαστικά τη δυνατότητα σε διαφορετικούς χρήστες να ακούσουν διαφορετικά κομμάτια από τον ίδιο καλλιτέχνη. Ο βαθμός παραλληλίας είναι μια απαραίτητη παράμετρος του streaming framework για να μπορεί να εξυπηρετήσει ταυτόχρονα τον φόρτο (“load”) από πολλαπλούς χρήστες και άρα θα πρέπει το σύστημα να μπορεί να ανταποκριθεί κατάλληλα.
● "Pull" συνάρτηση: Ο ρόλος της συνάρτησης pull είναι να πάρει ότι πληροφορία χρειάζεται για ένα συγκεριμένο κλειδί από έναν ενδιάμεσο κόμβο. Αυτό που κάνει είναι να λαμβάνει τα values που σχετίζονται με ένα συγκεκριμένο κλειδί (πχ. το όνομα του τραγουδιού το οποίο ακούει ένας χρήστης του framework) και να το προωθήσει κατάλληλα στον ίδιο. Η συνάρτηση pull λειτουργεί ανεξάρτητα από τον τρόπο επιλογής της ακρόασης. Πιο συγκεκριμένα, είτε αν ο χρήστης επιλέξει να ακούει σε πραγματικό χρόνο το τραγούδι είτε επιλέξει να το ακούσει μεταγενέστερα, η συνάρτηση pull λειτουργεί με τον ίδιο ακριβώς τρόπο. Για ένα συγκεκριμένο κλειδί δημιουργείται μια λίστα από τις τιμές που αντιστοιχούν σε αυτό το κλειδί. Αυτή η λίστα με values στέλνεται στον ενδιάμεσο κόμβο όπου μπαίνουν σε μια ενδιάμεση δομή δεδομένων και στη συνέχεια, αυτά στέλνονται σε όλους τους ακροατές που ενδιαφέρονται για το ίδιο κλειδί.

Για τις ανάγκες του framework χρειάζεται να κατασκευάσουμε τρία(3) βασικά components: *τον publisher node, τους broker nodes* και *τον consumer node*. Παρακάτω

ακολουθεί η περιγραφή του καθενός από τα components που θα πρέπει να υλοποιήσετε στα πλαίσια της εργασίας:

1. **Publisher node:** Ο συγκεκριμένος κόμβος είναι απαραίτητος για την αποθήκευση, αναζήτηση και εξαγωγή όλων των μουσικών κομματιών τα οποία θα επιθυμούν να ακούσουν οι ακροατές. Είναι ουσιαστικά η πηγή μας (source) από την οποία θα διαβάζουμε τα δεδομένα και ο κάθε κόμβος θα είναι υπεύθυνος για ένα υποσύνολο από καλλιτέχνες. Ο κόμβος αυτός λοιπόν, με κατάλληλο τρόπο και σε κατάλληλο χρονικό διάστημα, στέλνει τα δεδομένα του στους ενδιάμεσους κόμβους. Αυτό σημαίνει ότι θα πρέπει να μπορεί να είναι σε θέση να εξυπηρετεί πολλαπλά requests για διαφορετικά κομμάτια του ίδιου καλλιτέχνη ή και για διαφορετικούς καλλιτέχνες, όπως συμβαίνει σε ένα πραγματικό σύστημα. Επιπλέον, το μουσικό αρχείο που θα μεταφέρεται, ***θα πρέπει να μεταφέρεται κατά μικρά κομμάτια("chunks") και όχι αυτούσιο.*** Η βασική, λοιπόν, λειτουργία του Publisher Node, είναι να κάνει push τα δεδομένα για τα κλειδιά για τα οποία είναι υπεύθυνος, στους brokers που είναι υπεύθυνοι να εξυπηρετήσουν τα εν λόγω κλειδιά. Ο publisher node ως υπεύθυνος για το υποσύνολο των καναλιών των καλλιτεχνών, θα πρέπει κατά την έναρξη της λειτουργίας του και την αρχικοποίησή του, να γνωρίζει το σύνολο της πληροφορίας για την οποία είναι υπεύθυνος. Πρέπει ουσιαστικά να αρχικοποιήσει τις κατάλληλες δομές που θα έχει ώστε να γνωρίζει πχ. τα κομμάτια τα οποία είναι διαθέσιμα, για ποιους καλλιτέχνες είναι υπεύθυνος κλπ., έτσι ώστε να μπορεί να στείλει όλη την απαραίτητη πληροφορία τους στους ενδιάμεσους κόμβους. Κάθε κόμβος που είναι υπεύθυνος για ένα σύνολο από καλλιτέχνες, θα πρέπει να γνωρίζει σε ποιον ενδιάμεσο κόμβο (broker) θα αποστείλει τα δεδομένα. Υπεύθυνος γι'αυτή τη λειτουργικότητα θα είναι ο broker όπως θα δούμε παρακάτω. Γι'αυτό το λόγο όμως ***είναι απαραίτητο να υπάρχει κάποιας μορφής συγχρονισμός.*** Πιο συγκεκριμένα, με κατάλληλες τεχνικές να ενημερωθούν δομές διαμοιραζόμενες στους brokers ώστε να ξέρουν ποιος publisher node είναι υπεύθυνος για ποιον καλλιτέχνη. ***Έτσι, όταν γίνει μεταφορά των δεδομένων, τότε αυτά θα μεταφερθούν στους αντίστοιχους brokers και όχι σε όλους.*** Αν για τον οποιονδήποτε λόγο ο publisher node δεν στέλνει αποτελέσματα (πχ. γιατί δεν υπάρχει το συγκεκριμένο κομμάτι που έχει ζητήσει ένας user), τότε ο publisher node επιστρέφει κατάλληλο μήνυμα το οποίο θα πρέπει να μεταφερθεί στον broker και αφού γίνει κατάλληλα handle να προωθηθεί στους consumer nodes.

2. **Broker Nodes:** Οι συγκεκριμένοι κόμβοι είναι υπεύθυνοι για ένα εύρος από topics, όπως ονομάζονται, που στην περίπτωσή μας είναι οι καλλιτέχνες. Ένα απαραίτητο χαρακτηριστικό τέτοιων συστημάτων είναι ότι χρειάζεται να γίνει ισοκατανομή μεταξύ των brokers. Αυτό πρακτικά σημαίνει ότι περίπου κάθε broker θα πρέπει να εξυπηρετεί (περίπου) το ίδιο μέγεθος από καλλιτέχνες. Προκειμένου, λοιπόν να γίνει μια ισοκατανομή στο πόσα topics αναλαμβάνει κάθε broker node,

χρειάζεται να γίνει κάποιας μορφής hashing. Για να γίνει αυτό χρησιμοποιούμε μια hash function (π.χ. SHA1 ή MD5) και παίρνουμε το hash του String ArtistName και το hash του IP+Port του broker. Έτσι ο κάθε broker θα είναι υπεύθυνος για όσα hashes εγγραφών είναι μικρότερα από το hash του IP+Port του. (Προσοχή για το ποια hashes αντιστοιχούν στον broker με το μικρότερο hash, θα χρειαστεί η χρήση mod). Αυτοί οι κόμβοι ενημερώνουν κατάλληλα τους publisher nodes για το ποια κλειδιά είναι υπεύθυνοι και στην συνέχεια ανοίγουν επικοινωνία με αυτούς (με τους publisher nodes) έτσι ώστε να είναι σε θέση να λάβουν την πληροφορία όταν αυτή γίνει διαθέσιμη και να την προωθήσουν στους κατάλληλους consumers που έχουν γίνει register επάνω τους. Επίσης, σε κάθε νέα σύνδεση κάποιου νέου consumer τον ενημερώνουν κατάλληλα για το ποιοι ειναι οι υπολοιποι brokers και για ποια topics είναι υπεύθυνοι. Ένα πολύ σημαντικό στοιχείο των brokers είναι ότι η πληροφορία την οποία αποστέλλουν στους consumer nodes πρέπει να την στείλουν ταυτόχρονα σε όλους. Γι'αυτό το λόγο, επειδή θα πρέπει ταυτόχρονα να γίνεται μετάδοση της μουσικής στους ακροατές, θα χρειαστεί να χρησιμοποιήσετε αρχές πολυνηματικού προγραμματισμού, ώστε να είναι εφικτή η πολλαπλή αποστολή σε πραγματικό χρόνο στους ακροατές.

3. **Consumer Node:** Ο συγκεκριμένος κόμβος είναι υπεύθυνος για να δέχεται την απαιτούμενη πληροφορία από το σύστημα. Πρακτικά, θα είναι το κινητό σας το οποίο θα πρέπει να είναι σε θέση να αναπαράγει όλη τη μουσική πληροφορία την οποία θα λαμβάνει. Αυτό που ουσιαστικά δέχεται από τους broker nodes είναι tuples της μορφής: <ArtistName, SongName, MusicFileInfo>. Ο Consume Node έχει δύο δυνατές επιλογές: είτε να είναι μόνιμα συνδεδεμένος με τον broker που είναι υπεύθυνος για έναν καλλιτέχνη, είτε να βρεθεί και offline. Γι'αυτο το λόγο θα πρέπει να ρωτάει κατάλληλα τον πρώτο τυχαίο broker για το ποιοι ειναι οι διαθέσιμοι broker που είναι υπεύθυνοι για κάποιον καλλιτέχνη κατά την πρώτη επικοινωνία του consumer με το Event Delivery System. Ουσιαστικά επιστρέφεται ένα αντικείμενο της μορφής Info {ListOfBrokers {IP,Port} , < BrokerId, ArtistName>}. Με βάση αυτό το object, λοιπόν, ο consumer στην συνέχεια, είναι εύκολο να επιλέξει τον κατάλληλο broker ο οποίος θα του επιστρέψει την αντίστοιχη πληροφορία για έναν δεδομένο καλλιτέχνη.

Υλοποίηση Event Delivery System με Java 8 [3]

Ο τρόπος λειτουργίας του Event Delivery System θα γίνεται ως εξής:

1. Κάθε ένας από τους παραπάνω κόμβους είναι ένα ξεχωριστό process (μια διαφορετική main που εκτελείται κάθε φορά).

2. Σε κάθε broker κόμβο θα υπάρχει ένα instance της εφαρμογής που θα προωθεί κατάλληλα τα δεδομένα.

3. Κάθε instance (publisher node, broker nodes) θα ακούει σε κάποιο προκαθορισμένο port για συνδέσεις.

4. Όταν λάβει ο broker node ένα query από τον χρήστη, αρχικά κοιτά αν είναι ήδη συνδεδεμένος μαζί του (αν έχει γίνει register ο consumer node στον συγκεκριμένο broker και αν έχει κάνει login). Αν υπάρχει ήδη σύνδεση, τότε επιτρέπει στον χρήστη να ζητήσει και να κάνει pull το κατάλληλο κομμάτι. Αν όχι, τότε δημιουργείται μια νέα σύνδεση και παράλληλα επιστρέφεται στον consumer η λίστα με τους υπόλοιπους brokers και τα κλειδιά για τα οποία είναι υπεύθυνοι. Όταν βρεθεί το κομμάτι, τότε ο broker το στέλνει κατάλληλα στον consumer. Αν δεν υπάρχει, επιστρέφει κατάλληλο μήνυμα με το οποίο ενημερώνει τον χρήστη αν επιθυμεί να ακούσει κάποιο άλλο διαθέσιμο.

5. Κάθε broker θα προωθεί τα δεδομένα για το συγκεκριμένο εύρος δεδομένων για το οποίο είναι υπεύθυνος (ουσιαστικά για τους καλλιτέχνες για τους οποίους είναι υπεύθυνος).

6. Όταν ο consumer node παραλάβει την πληροφορία την αναπαράγει κατάλληλα. **Προσοχή:** Για τις ανάγκες του **πρώτου(Α')** παραδοτέου, **δεν είναι απαραίτητο** να φτιάξετε κάποιον player. Αρκεί τα chunks από τα μουσικά αρχεία να μπορούν να αναπαραχθούν αν ανοιχτούν με έναν player του λειτουργικού συστήματος. Με άλλα λόγια, θα πρέπει consumer να τα αποθηκεύει τοπικά. Στο δεύτερο παραδοτέο, αυτά τα chunks, θα αναπαράγονται με τις κατάλληλες βιβλιοθήκες του Android Framework.

7. Θα πρέπει να δίνεται η επιλογή στον χρήστη είτε να ακούσει σε πραγματικό χρόνο είτε να αποθηκεύσει τοπικά τη μουσική για μεταγενέστερα. Και στις δύο περιπτώσεις για το πρώτο παραδοτέο, τα chunks των κομματιών θα αποθηκεύονται τοπικά στον consumer.

Android Application

Θα αναπτύξετε μια εφαρμογή που θα εκτελείται σε συσκευές με λειτουργικό Android και είναι ικανή να αναπαράγει σε πραγματικό χρόνο τη μουσική η οποία δημοσιεύεται στο κανάλι του καλλιτέχνη. Η εφαρμογή θα υλοποιηθεί στην πλατφόρμα Android και θα επωφελείται από το music streaming framework το οποίο θα τρέχει ανεξάρτητα. Η εφαρμογή Android θα εκτελείται ώς εξής:

1. Η βασική οθόνη της εφαρμογής θα περιλαμβάνει μια λίστα από καλλιτέχνες απ' όπου ο χρήστης θα μπορεί να επιλέξει αυτόν ή αυτούς για τους οποίους επιθυμεί να ακούσει τραγούδια (ή ακόμα και το είδος της μουσικής).

2. Στην περίπτωση που η αρχική λίστα είναι κενή, η εφαρμογή θα απευθύνεται σε έναν broker node ζητώντας την απαραίτητη πληροφορία αν υπάρχει, ειδάλλως θα περιμένει για κατάλληλο χρονικό διάστημα μέχρι αυτή να είναι διαθέσιμη.
3. Ακολουθείται η διαδικασία που περιγράψαμε στο κομμάτι της υλοποίησης της αποστολής και διαχείρισης των δεδομένων του publisher από τους brokers. Αν τυχόν δεν υπάρχει διαθέσιμη πληροφορία εκείνη τη στιγμή και έχει περάσει ένα εύλογο χρονικό διάστημα (κάποιων δευτερολέπτων), τότε επιστρέφεται κατάλληλο μήνυμα («ότι δεν βρέθηκε») σχετικά με τον συγκεκριμένο καλλιτέχνη, δίνοντας παράλληλα την δυνατότητα να διαλέξει από άλλα κομμάτια του ίδιου καλλιτέχνη.
4. Με το που λάβει την πληροφορία από τον broker node πρέπει να μπορεί να γίνει αναπαραγωγή της λαμβανόμενης πληροφορίας από τον πολύ απλό player που θα πρέπει να φτιάξετε με χρήση του Android Framework.

Παραδοτέα εργασίας: Το project θα παραδοθεί σε δύο φάσεις:

Παραδοτέο Α: (Ημερομηνία παράδοσης: 5/4/2020)

Στο παραδοτέο αυτό, θα πρέπει να έχετε ολοκληρώσει εντελώς το music streaming σύστημα, όπως ακριβώς σας έχει ζητηθεί, έτσι ώστε να μπορεί να χρησιμοποιηθεί στην επόμενη φάση της εργασίας του μαθήματος.

Παραδοτέο Β: (Ημερομηνία παράδοσης: 24/5/2020)

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.

Ομάδες: Όλοι οι φοιτητές θα πρέπει να σχηματίσουν ομάδες των τριών (3) ή τεσσάρων (4) 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/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