

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2021-2022

Υποχρεωτική εργασία

Τα τελευταία χρόνια έχει παρατηρηθεί μεγάλη αύξηση των εφαρμογών διαμοιρασμού πολυμεσικού περιεχομένου σε χρήστες στα πλαίσια της online κοινωνικής δικτύωσης. Στα πλαίσια της εργασίας του μαθήματος καλείστε να δημιουργήσετε ένα απλό τέτοιο σύστημα διαμοιρασμού κειμένου και πολυμεσικού περιεχομένου. Μια τέτοια γνωστή υπηρεσία που πιθανόν να έχετε χρησιμοποιήσει είναι το Whatsapp, το Slack κλπ, που παρέχουν τη δυνατότητα σε μεγάλο πλήθος χρηστών να διαμοιράζονται πολυμεσικό περιεχόμενο μεταξύ τους.

Τα συστήματα αυτά, έχουν την ικανότητα να εξυπηρετούν ένα μεγάλο αριθμό χρηστών. Αυτό σημαίνει ότι κάθε user διατηρεί ένα προσωπικό “προφίλ”, δηλαδή, ένα χώρο όπου μπορεί να προσθέτει το περιεχόμενό του (video, κείμενο, φωτογραφίες) το οποίο με τη σειρά τους μπορούν να το δουν διάφοροι φίλοι του μέσω του δικτύου καθώς έρχεται σε επικοινωνία μαζί τους. Οι χρήστες που είναι φίλοι μεταξύ τους μπορούν να στείλουν μηνύματα μεταξύ τους αλλά και να φτιάξουν διάφορες ομαδικές συνομιλίες (συνήθως αυτές ονομάζονται κανάλια) ώστε να διαμοιράζονται περιεχόμενο μεταξύ τους. Κάθε χρήστης μπορεί είτε να τραβάει ένα βίντεο και να το ανεβάζει εκείνη τη στιγμή, είτε να ανεβάζει ένα βίντεο που έχει τραβήξει προγενέστερα, να τραβάει μια φωτογραφία εκείνη τη στιγμή είτε μια φωτογραφία που έχει τραβήξει προγενέστερα καθώς και να στέλνει μηνύματα σε μορφή κειμένου. Κάθε φορά που ένα περιεχόμενο ανεβαίνει στην πλατφόρμα, αυτό γίνεται διαθέσιμο σε όλους τους χρήστες που βρίσκονται στη συγκεκριμένη συνομιλία-κανάλι. Σ’ ένα τέτοιο σύστημα, το περιεχόμενο ποτε δεν αποστέλλεται εξ’ολοκλήρου, αλλά γίνεται, όπως λέμε, streamed στον τελικό αποδέκτη, όπως συμβαίνει και σε μια φωτογραφία ή ένα βίντεο.

Στην εργασία καλείστε να φτιάξετε ένα τέτοιο σύστημα επικοινωνίας με υποστήριξη για πολυμεσικό περιεχόμενο με χρήση της java 8, στο οποίο η πληροφορία που θέλουμε να μεταδώσουμε είναι είτε βίντεο, είτε φωτογραφία, είτε κείμενο από έναν ή παραπάνω χρήστες σε ένα πλήθος από πολλούς συνδρομητές. Λόγω του μεγάλου αριθμού των χρηστών που θέλουμε να εξυπηρετήσουμε, χρειάζεται να υλοποιήσουμε ένα έξυπνο σύστημα, το οποίο θα μπορεί να μεταφέρει κατάλληλα και στους κατάλληλους αποδέκτες το σχετικό περιεχόμενο. Για να μοιράσουμε στους χρήστες αυτήν την πληροφορία, χρειάζεται να ξέρουμε ποιοι είναι αυτοί που ενδιαφέρονται γι’αυτή (**subscribers**), πώς μπορούν να εκδηλώσουν ενδιαφέρον γι’αυτή (**topic subscription**) και πώς μπορούν να την πάρουν. Η πληροφορία αυτή θα μεταφέρεται κατάλληλα από το σύστημα που καλείστε να υλοποιήσετε έτσι ώστε οι χρήστες να μπορούν να δουν το περιεχόμενο μόλις αυτό γίνεται διαθέσιμο στο σύστημα.

Η ανάπτυξη του συστήματος θα σας επιτρέψει να εξοικειωθείτε τόσο με την έννοια ενός τέτοιου συστήματος Instant Messaging, 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 multimedia streaming Framework (Event Delivery System) που θα είναι υπεύθυνο να υποστηρίξει την προώθηση και λήψη (streaming) των πολυμεσικών δεδομένων (video, φωτογραφίες και κείμενο).*
- 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 μεταξύ των users μιας ομαδικής συνομιλίας, και στη συνέχεια το περιεχόμενο να μπορεί να αναπαραχθεί στο κινητό (είτε να εμφανιστεί αν είναι φωτογραφία ή κείμενο).*

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 ενδιαμέσος κόμβος στο σύστημα, που ονομάζεται **broker** και μπορεί να δέχεται ταυτόχρονα δεδομένα από διαφορετικούς **publishers** (στην περίπτωση μας users), να μπορεί να μεταφέρει κατάλληλα τα αποτελέσματα στους τελικούς **subscribers** (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 users στους ενδιαμέσους κόμβους, όσο και από τους ενδιαμέσους κόμβους στους subscribers, γιατί θα πρέπει όλοι οι εγγεγραμμένοι χρήστες να λάβουν ταυτόχρονα το ίδιο περιεχόμενο.

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 ενδιαμέσο κόμβο ένα ζεύγος key(ή αλλιώς topic)/value το οποίο και αποθηκεύεται σε κατάλληλη ενδιάμεση δομή(πχ. queue) ώστε να μπορεί να προωθηθεί στη συνέχεια. Ένας επιπλέον λόγος που χρειάζεται αυτή η ενδιάμεση δομή είναι γιατί όταν μπει ένας νέος χρήστης στην ομαδική συνομιλία θα πρέπει να μπορεί να δει όλο το ιστορικό της συνομιλίας. Στην περίπτωση μας, η συνάρτηση “push” μπορεί να πάρει σαν είσοδο ό,τι πληροφορία απαιτείται για να μεταφερθεί από το προφίλ του user (πχ. ονομα user, τίτλος βίντεο το οποίο γίνεται streamed ή φωτογραφίες κ.οκ.), καθώς και τα αντίστοιχα δεδομένα (πχ. το βίντεο ή η φωτογραφία ή το κείμενο το οποίο γίνεται streamed). Μια πολύ σημαντική παράμετρος που πρέπει να λάβουμε υπόψιν μας κατά την ανάπτυξη

του συστήματος είναι: Ένα video ή φωτογραφία 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 δουν την πληροφορία που κοινοποίησε ο ίδιος χρήστης. 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 σύστημά σας να μπορεί να ανταποκριθεί κατάλληλα.

- "Pull" συνάρτηση: Ο ρόλος της συνάρτησης pull είναι να πάρει ότι πληροφορία χρειάζεται για ένα συγκεκριμένο κλειδί από έναν ενδιάμεσο κόμβο. Αυτό που κάνει είναι να λαμβάνει τα values που σχετίζονται με ένα συγκεκριμένο κλειδί (πχ. το όνομα της ομαδικής συνομιλίας) και να το προωθήσει κατάλληλα στον ίδιο. 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* (προφίλ του χρήστη), τους *broker nodes*(ενδιάμεσοι κόμβοι) και τον *consumer* (τη λειτουργία του subscriber). Παρακάτω ακολουθεί η περιγραφή του καθενός από τα components που θα πρέπει να υλοποιήσετε στα πλαίσια της εργασίας:

1. **Publisher:** Το συγκεκριμένο component είναι αρμόδιο για την αποθήκευση, αναζήτηση και εξαγωγή όλων των πολυμέσων ενός χρήστη τα οποία θέλει να στείλει σε μια συνομιλία. Είναι ουσιαστικά η πηγή μας (source) από την οποία θα διαβάζουμε τα δεδομένα και κάθε τέτοιο component θα είναι υπεύθυνο για το προφίλ ενός χρήστη. Ο publisher, με κατάλληλο τρόπο και σε κατάλληλο χρονικό διάστημα, στέλνει τα δεδομένα του στους ενδιάμεσους κόμβους. Αυτό στην πράξη σημαίνει ότι με κάποιο τρόπο κάνει notify τους ενδιάμεσους κόμβους ότι έχει κάποιο νέο υλικό και στη συνέχεια τους το στέλνει. Συνεπώς, θα πρέπει να είναι σε θέση να εξυπηρετεί πολλαπλά requests από διαφορετικούς ενδιάμεσους κόμβους, όπως συμβαίνει σε ένα πραγματικό σύστημα. Επιπλέον, το περιεχόμενο που θα μεταφέρεται, **θα πρέπει να μεταφέρεται κατά μικρά κομμάτια("chunks") και όχι αυτούσιο**. Ένας τέτοιος κόμβος, συνήθως παράγει δεδομένα για ένα ή περισσότερα κλειδιά (ιδανικά δηλαδή θα πρέπει ένας χρήστης να μπορεί να στείλει και να λάβει μηνύματα από παραπάνω από μια συνομιλίες). Γι' αυτό, λοιπόν, η βασική λειτουργία του Publisher είναι να κάνει push δεδομένα στους brokers που είναι υπεύθυνοι να εξυπηρετήσουν τα εν λόγω κλειδιά. Κάθε publisher που θέλει να στείλει μήνυμα σε μια συνομιλία(στο αντίστοιχο κλειδί δηλαδή), 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 κάποιος μορφής συγχρονισμός**. Επίσης, με κατάλληλες τεχνικές πρέπει να υπάρχουν δομές διαμοιραζόμενες στους brokers ώστε να ξέρουν ποιος publisher είναι subscribed σε ποια κλειδιά. **Έτσι, όταν γίνει μεταφορά των δεδομένων, τότε αυτά θα μεταφερθούν στους αντίστοιχους brokers και όχι σε όλους.**

2. Broker Nodes: Οι συγκεκριμένοι κόμβοι είναι υπεύθυνοι για ένα εύρος από κλειδιά(topics ονομάζονται) που στην περίπτωση μας είναι το όνομα μιας συνομιλίας(το όνομα του καναλιού πχ. “Κατανεμημένα_Συστήματα”). 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 συνομιλίες. 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 ChannelName 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 για το ποια κλειδιά είναι υπεύθυνοι και στην συνέχεια ανοίγουν επικοινωνία με αυτούς (με τους publisher) 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. Αυτό γίνεται κατά τη διάρκεια της αναζήτησης για μια συνομιλία. Επίσης, σε κάθε νέα σύνδεση κάποιου νέου consumer τον ενημερώνουν κατάλληλα για το ποιοι είναι οι υπολοίποι brokers και για ποια topics-συνομιλίες είναι υπεύθυνοι. Ένα πολύ σημαντικό στοιχείο των brokers είναι ότι η πληροφορία την οποία αποστέλλουν στους consumer πρέπει να την στείλουν ταυτόχρονα σε όλους. Γι’αυτό το λόγο, επειδή θα πρέπει ταυτόχρονα να γίνεται μετάδοση του πολυμεσικού περιεχομένου στους subscribers, θα χρειαστεί να χρησιμοποιήσετε αρχές πολυνηματικού προγραμματισμού, ώστε να είναι εφικτή η πολλαπλή αποστολή σε πραγματικό χρόνο στους subscribed χρήστες της συνομιλίας.

3. Consumer: Το συγκεκριμένο component είναι υπεύθυνο για να δέχεται την απαιτούμενη πληροφορία από το σύστημα. Πρακτικά, θα είναι μέρος της εφαρμογής του κινητού σας το οποίο θα πρέπει να είναι σε θέση να αναπαράγει το πολυμεσικό περιεχόμενο το οποίο θα λαμβάνει. Αυτό που ουσιαστικά δέχεται από τους broker nodes είναι tuples της μορφής: <ProfileName, ObjectName, ObjectFileInfo>. Ο consumer θα πρέπει να ρωτάει κατάλληλα τον πρώτο τυχαίο broker για το ποιοι είναι οι διαθέσιμοι broker που είναι υπεύθυνοι για τα διάφορα topic-συνομιλίες κατά την πρώτη επικοινωνία του consumer με το Event Delivery System. Ουσιαστικά επιστρέφεται ένα αντικείμενο της μορφής Info {ListOfBrokers {IP,Port} , < BrokerId, ProfileName, {Topics-Channels}>}. 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 ένα προφίλ χρήστη.

Υλοποίηση Event Delivery System με Java 8 [3]

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

1. Κάθε broker που εκτελείται είναι ένα ξεχωριστό process (μια διαφορετική main που εκτελείται κάθε φορά). **Ο producer και ο consumer εκτελούνται μέσα από το ίδιο process (aka UserNode) χρησιμοποιώντας πολυνηματικό προγραμματισμό.**

2. Κάθε broker κόμβος είναι ένα instance της εφαρμογής που θα προωθεί κατάλληλα

τα δεδομένα. Για τις ανάγκες της εργασίας θα πρέπει να έχετε τουλάχιστον $n \geq 3$ αριθμό από broker κόμβους και τουλάχιστον $m \geq 3$ UserNodes να τρέχουν.

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

4. Όταν λάβει ο broker node ένα event από τον χρήστη, αρχικά κοιτά αν είναι ήδη συνδεδεμένος μαζί του (αν έχει γίνει register ο consumer node στον συγκεκριμένο broker). Αν υπάρχει ήδη σύνδεση, τότε επιτρέπει στον χρήστη στείλει υλικό στη συνομιλία. Αν όχι, τότε δημιουργείται μια νέα σύνδεση και παράλληλα επιστρέφεται στον consumer η λίστα με τους υπόλοιπους brokers και τα topics-συνομιλίες για τα οποία είναι υπεύθυνοι. Όταν βρεθεί η συνομιλία, τότε ο broker το στέλνει κατάλληλα στον consumer. Αν δεν υπάρχει, επιστρέφει κατάλληλο μήνυμα με το οποίο ενημερώνει τον χρήστη αν επιθυμεί να ψάξει για κάποιο άλλη διαθέσιμη συνομιλία.

5. Κάθε broker θα προωθεί τα δεδομένα για το συγκεκριμένο εύρος δεδομένων για το οποίο είναι υπεύθυνος (ουσιαστικά για τα topics για τα οποία είναι υπεύθυνος).

6. Όταν ο consumer παραλάβει την πληροφορία την αναπαράγει κατάλληλα.

Προσοχή: Για τις ανάγκες του **πρώτου(Α')** παραδοτέου, **δεν είναι απαραίτητο** να φτιάξετε κάποιον video player (πχ με java swift). Αρκεί τα chunks από τα βίντεο, από τις φωτογραφίες ή το κείμενο να μπορούν να αναπαραχθούν, μόλις συντεθούν σε ένα ενιαίο αρχείο βίντεο, είτε φωτογραφία είτε σε αρχείο κειμένου στον consumer. Με άλλα λόγια, θα πρέπει ο consumer να τα αποθηκεύει τοπικά στο μηχάνημά σας και να μπορούν να ανοίξουν μέσα από τον file explorer του λειτουργικού σας. Στο δεύτερο παραδοτέο, αυτά τα chunks, θα αναπαράγονται με τις κατάλληλες βιβλιοθήκες του Android Framework.

7. Σε ότι αφορά το πρώτο παραδοτέο, θα πρέπει να υποστηρίζεται η λειτουργικότητα ανεβάσματος βίντεο στο σύστημα και της αποστολής κειμένου και φωτογραφίας. Αυτό μπορεί να γίνει με τη χρήση ενός απλού .mp4 αρχείου βίντεο (διάρκειας τουλάχιστον ίσης με 15 second), χωρίς την ανάγκη να υλοποιήσετε κάποιο video capture σύστημα (που θα συμβεί στο δεύτερο παραδοτέο).

8. Θα πρέπει να δίνεται η επιλογή στον χρήστη είτε να δει το βίντεο σε πραγματικό χρόνο είτε να αποθηκεύσει τοπικά το βίντεο για μεταγενέστερα. Και στις δύο περιπτώσεις για το πρώτο παραδοτέο, τα chunks του βίντεο θα αποθηκεύονται τοπικά στον consumer και θα ανασυντίθενται σε ένα ενιαίο το οποίο θα πρέπει να μπορεί να αναπαραχθεί.

9. Για δική σας βοήθεια, επειδή το UserNode θα αξιοποιεί πολυνηματικό προγραμματισμό, μπορείτε να χρησιμοποιήσετε δύο ξεχωριστά αρχεία για να εκτυπώνετε το output σε διάφορα στάδια της εκτέλεσης αλλά και για να μπορείτε να κάνετε debug για την ορθή λειτουργία.

Android Application

Θα αναπτύξετε μια εφαρμογή που θα εκτελείται σε συσκευές με λειτουργικό Android και είναι ικανή να μπορεί να στέλνει μηνύματα σε πραγματικό χρόνο από το προφίλ ενός χρήστη. Η εφαρμογή θα υλοποιηθεί στην πλατφόρμα Android και θα επωφελείται από το multimedia streaming framework το οποίο θα τρέχει ανεξάρτητα. Η εφαρμογή Android θα εκτελείται ως εξής:

1. Η βασική οθόνη της εφαρμογής θα περιλαμβάνει μια λίστα από συνομιλίες απ'

όπου ο χρήστης θα μπορεί να επιλέξει για να στείλει πολυμεσικό περιεχόμενο.

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.

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: 1/5/2022)

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

Παραδοτέο Β: (Ημερομηνία παράδοσης: 12/6/2022)

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) 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.

Bonus: Extra features όπως υποστήριξη για stories, fault tolerance αν πέσει κάποιος broker κλπ. θα έχουν βαθμολογικό bonus έως 10% στο βαθμό του project αναλόγως της δυσκολίας υλοποίησης.

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>