ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Πανεπιστήμιο Πειραιώς - Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης

| | |
|---|---|
| Τίτλος Διατριβής: | Ανάπτυξη Android Παιχνιδιού Πραγματικού Χρόνου με την Χρήση Firebase και της Ασαφούς Λογικής.<br><br>Real-time Android Game Development Using Firebase And Fuzzy Logic. |
| Ονοματεπώνυμο Φοιτητή: | Τσελεπατιώτης Μιχαήλ |
| Πατρώνυμο: | Κωνσταντίνος |
| Αριθμός Μητρώου: | ΜΠΣΠ20053 |
| Επιβλέπων: | Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής |

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

**Τριμελής Εξεταστική Επιτροπή**

| | | |
|---|---|---|
| **Ευθύμιος Αλέπης** | **Μαρία Βίρβου** | **Κωνσταντίνος Πατσάκης** |
| **Αναπληρωτής Καθηγητής** | **Καθηγήτρια** | **Αναπληρωτής Καθηγητής** |

# Ευχαριστίες

Για την υλοποίηση της παρούσας διπλωματικής εργασίας, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Ευθύμιο Αλέπη, για την εμπιστοσύνη του, τον υποψήφιο διδάκτορα κ. Σπύρο Παπαδημητρίου για τις πολύτιμες συμβουλές του πάνω στην ασαφή λογική, καθώς και τον Επίκουρο Καθηγητή κ. Σωτηρόπουλο Διονύσιο.

Επίσης, θέλω να ευχαριστήσω θερμά τις συμφοιτήτριες μου Ιωάννα Δαβλιάνη και Ηλιάνα Παπαδημητρίου, για την άψογη συνεργασία που είχαμε καθ' όλη την διάρκεια του ΠΜΣ, με τις οποίες συνεργαστήκαμε για την ολοκλήρωση των εργασιών εξαμήνων.

Τέλος, θέλω να ευχαριστήσω την Ελένη Κατραμάδου για την βοήθεια της πάνω στο γραφιστικό κομμάτι της εφαρμογής, καθώς οικογένεια μου, για την υποστήριξη της, σε όλη τη διάρκεια του εκπαιδευτικού μου βίου.

# Πίνακας Περιεχομένων

# Περίληψη

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών "Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης", και έχει ως στόχο την υλοποίηση μιας εφαρμογής Android και συγκεκριμένα ενός παιχνιδιού λέξεων, σε πραγματικό χρόνο και για πολλαπλούς παίκτες με την χρήση του Firebase. Το παιχνίδι αυτό αφορά τον ανταγωνισμό 2 παικτών ως προς την δημιουργία λέξεων. Συγκεκριμένα, στους 2 παίκτες μοιράζεται ένας αριθμός ιδίων γραμμάτων, και οι παίκτες καλούνται να δημιουργήσουν όσες περισσότερες λέξεις μπορούν με τα γράμματα, στην διάρκεια ενός γύρου. Η κάθε παρτίδα περιλαμβάνει 3 γύρους και νικητής είναι αυτός που θα έχει κερδίσει τους περισσότερους γύρους. Ο κάθε χρήστης συνδέεται στην εφαρμογή, κάνοντας Login μέσω του προσωπικού του λογαριασμού που μπορεί να διαθέτει (Facebook ή Google). Επιπλέον παρέχεται στον χρήστη η δυνατότητα της προσθήκης άλλων χρηστών στην λίστα φίλων του, η δυνατότητα να συνομιλήσει (chat) με τους φίλους του ή και με τους τυχαίους αντιπάλους που μπορεί να συναντήσει, η δυνατότητα να προσκαλεί και να προκαλέσει τους φίλους σε παιχνίδι (αγώνα παίκτης εναντίον παίκτη) καθώς και η δυνατότητα να μπορεί να παίξει με τυχαίο αντίπαλο εκτός της λίστας φίλων του. Για την τελευταία, κρίθηκε απαραίτητη η χρήση ενός μηχανισμού AI (bot) για εκείνες τις χρονικές στιγμές που μπορεί να μην υπάρχουν συνδεδεμένοι και διαθέσιμοι χρήστες ως τυχαίοι αντίπαλοι. Το bot προσαρμόζεται στο επίπεδο του χρήστη και έτσι δημιουργείται στον χρήστη η ψευδαίσθηση ότι παίζει με αληθινό αντίπαλο. Φυσικά όταν υπάρχουν συνδεδεμένοι χρήστες και είναι διαθέσιμοι ως τυχαίοι αντίπαλοι, ο μηχανισμός του bot δεν χρησιμοποιείται. Η υλοποίηση του παραπάνω μηχανισμού γίνεται με την χρήση μιας δομής δεδομένων Prefix Tree (Trie) και της ασαφής λογικής με την οποία το bot αποφασίζει πότε να παίξει και πόσες λέξεις να δημιουργήσει. Επιπλέον, η εφαρμογή παρέχει ένα σύστημα επιπέδου (Level System) καθώς με αυτόν τον τρόπο γίνεται η επιβράβευση αλλά και η αξιολόγηση του επιπέδου του χρήστη η οποία είναι ανάλογη των παιχνιδιών που έχει παίξει και των αντίστοιχων αποτελεσμάτων τους (νίκες, ήττες, ισοπαλίες). Επιπρόσθετα η εφαρμογή διαθέτει και άλλους τρόπους επιβράβευσης, όπως, επιτεύγματα (achievements), τα οποία προκύπτουν από τα κατορθώματα του χρήστη κατά την διάρκεια του παιχνιδιού, πίνακας με τους καλύτερους παίκτες (Top Players), ο οποίος βρίσκεται στην αρχική οθόνη της εφαρμογής, και είναι ένας τρόπος όπου υποσυνείδητα δημιουργεί την ανάγκη στον χρήστη να εξελιχθεί και να γίνει καλύτερος ώστε, κάποια μέρα, να υπάρξει και το δικό του όνομα στον πίνακα με τους καλυτέρους παίκτες. Τέλος, η εφαρμογή διαθέτει επιβράβευση με καθημερινά δώρα, όπως δωρεάν εισιτήρια (για

προσκλήσεις φίλων σε παιχνίδι), βοήθειες για μέσα στο παιχνίδι, εμπειρίες επιπέδου (Level XP) καθώς και αλλά πολλά. Τέλος, όπως αναφέρθηκε και παραπάνω, η εφαρμογή διαθέτει έναν μηχανισμό συνομιλίας (chat) ώστε οι παίκτες να μπορούν να επικοινωνούν και να ανταλλάζουν μηνύματα. Για την καλύτερη ασφάλεια τόσο των χρηστών όσο και των προσωπικών τους δεδομένων, (ιδιωτικές πληροφορίες, ευαίσθητα δεδομένα κ.α) έχει υλοποιηθεί ένας μηχανισμός κρυπτογράφησης από άκρο σε άκρο (End-to-end encryption) όπου μόνο οι χρήστες που επικοινωνούν μπορούν να διαβάσουν τα μηνύματα. Έτσι τα μηνύματα και οι επιπλέον πληροφορίες διαχείρισης που τα συνοδεύουν αποθηκεύονται και ανακτώνται σε κρυπτογραφημένη μορφή, και ο κάθε client αντίστοιχα τα αποκωδικοποιεί. Παρόλο που ο πιο σύνηθες αλγόριθμος κρυπτογράφησης εφαρμογών συνομιλίας είναι ο RSA (ασύμμετρου κλειδιού) εντούτοις στην παρούσα διπλωματική υλοποιήθηκε ο AES (Συμμετρικού Κλειδιού)

# Abstract

This diploma thesis was conducted as a component of the Postgraduate Program called "Advanced Information Systems - Software Development and Artificial Intelligence." The objective of this thesis was to utilize Firebase for the development of an Android application, specifically a word game. The game enables real-time interaction among users, where two players compete against each other by creating words. Each player receives a predetermined set of identical letters and must utilize them to form as many words as possible within a timed round. The game consists of three rounds, and the player who wins the majority of rounds is declared the overall winner. To access the application, users are required to log in using their personal accounts, such as Facebook or Google. Furthermore, the application provides users with various additional features. Users have the capability to add other users to their list of friends, allowing them to establish connections and interact with them. This includes the ability to engage in chat conversations with both friends and random opponents they encounter. Additionally, users can send invitations and challenges to their friends for a player versus player match, enhancing the competitive aspect of the game. Moreover, the application offers the option to play with a random opponent who is not on the user's friends list. To ensure uninterrupted gameplay even when there are no logged in and available users as random opponents, an AI mechanism, commonly referred to as a "bot," has been incorporated. The bot is designed to adapt to the user's skill level, creating the impression that the user is competing against a real opponent. It is important to note that when there are logged in users who are available as random opponents, the bot's mechanism is not utilized. The implementation of the aforementioned mechanism incorporates a prefix tree (Trie) data structure and fuzzy logic. This combination enables the bot to make decisions regarding when to participate in a game and how many words to generate. Furthermore, the application introduces a Level System to evaluate and reward the user's progress. The user's level is determined based on the number of games played and their outcomes, including wins, losses, and draws. Additionally, the application offers various other forms of rewards. Users can earn achievements based on their in-game accomplishments. The home screen of the application displays a table featuring the top players, subtly motivating users to strive for improvement and aspire to see their own name among the best players. Moreover, the app provides daily gifts as rewards, such as free tickets for inviting friends to play, in-game assistance, experience points to level up, and more. Lastly, as previously discussed, the application incorporates a chat feature to facilitate communication and message exchange among players. To ensure enhanced security for both users and their

personal data, including private information and sensitive data, an end-to-end encryption mechanism has been implemented. This mechanism allows only the intended users involved in the communication to read the messages. Consequently, both the messages themselves and the associated management information are stored and retrieved in an encrypted format, with each client responsible for decoding them. While the widely used RSA algorithm (asymmetric key) is commonly employed for encrypting chat applications, this thesis has implemented the AES algorithm (Symmetric Key) for this purpose.

# Κεφάλαιο 1 - Βασικές Εννοείς

## 1.1 Προγραμματισμός Υπολογιστών

Ο προγραμματισμός υπολογιστών είναι η διαδικασία σχεδιασμού και δημιουργίας ενός εκτελέσιμου προγράμματος υπολογιστή με στόχο την επίλυση ενός συγκεκριμένου υπολογιστικού προβλήματος ή την εκτέλεση μιας συγκεκριμένης διαδικασίας. Ο προγραμματισμός περιλαμβάνει όλες τις διαδικασίες εκείνες που φέρνουν εις πέρας το επιθυμητό αποτέλεσμα όπως: η κατανόηση και η ανάλυση του προβλήματος, ο σχεδιασμός και η δημιουργία αλγορίθμων καθώς και την εφαρμογή των αλγορίθμων αυτών σε μια επιλεγμένη γλώσσα προγραμματισμού (κοινώς αναφέρεται ως κωδικοποίηση). Η κωδικοποίηση αυτή που συχνά αναφέρεται και ως “κώδικας”, είναι κατανοητή από τον προγραμματιστή και όχι από τον υπολογιστή. Ο κώδικας αυτός ονομάζεται πηγαίος κώδικας ενός προγράμματος και είναι γραμμένος σε μία ή περισσότερες γλώσσες προγραμματισμού. Ο σκοπός του προγραμματισμού είναι να δημιουργήσει μια σειρά διαφόρων οδηγιών προς τον υπολογιστή με σκοπό να βελτιστοποιηθεί ή αυτοματοποιηθεί η απόδοση μιας οποιαδήποτε εργασίας, από έναν απλό υπολογισμό μιας μαθηματικής πράξης έως την υλοποίηση ενός λειτουργικού συστήματος, συχνότερα μεν για την επίλυση ενός συγκεκριμένου προβλήματος. Ο προγραμματισμός απαιτεί συχνά εμπειρία και γνώση σε διάφορα θέματα, συμπεριλαμβανομένης της γνώσης του τομέα εφαρμογή των εξειδικευμένων αλγορίθμων, των μαθηματικών καθώς και γνώση διάφορων άλλων επιστημών.

## 1.2 Γλώσσα Προγραμματισμού

Μια γλώσσα προγραμματισμού είναι ένας είδος ένα είδος γλώσσας υπολογιστή, δηλαδή είναι ένας τρόπος επικοινωνίας ανθρώπου και υπολογιστή και περιλαμβάνει ένα σύνολο συμβολοσειρών που παράγουν διάφορα είδη εξόδου κώδικα μηχανής. Οι γλώσσες προγραμματισμού χρησιμοποιούνται στον προγραμματισμό υπολογιστών για την εφαρμογή διαφόρων αλγορίθμων για την επίλυση προβλημάτων ή την ικανοποίηση διαφόρων ψηφιακών αναγκών.

Οι περισσότερες γλώσσες προγραμματισμού αποτελούν οδηγίες ειδικά διαμορφωμένες για τους υπολογιστές και γενικότερα για ένα μεγάλο φάσμα ηλεκτρονικών συσκευών όπως κινητές συσκευές κτλ. Υπάρχουν προγραμματισιμά μηχανήματα που χρησιμοποιούν ένα

σύνολο συγκεκριμένων οδηγιών και όχι γλώσσες γενικού προγραμματισμού. Από τις αρχές του 1800, τα προγράμματα χρησιμοποιούνται για να κατευθύνουν τη συμπεριφορά μηχανών όπως οι αργαλειοί Jacquard, τα μουσικά κουτιά και τα πιάνα παικτών. Τα προγράμματα για αυτές τις μηχανές (όπως οι κύλινδροι ενός πιάνου παικτών) δεν παρήγαγε τη διαφορετική συμπεριφορά ως απάντηση στις διαφορετικές εισροές ή τις συνθήκες.

Οι γλώσσες προγραμματισμού δεν μπορούν να κατηγοριοποιηθούν με ευκολία. Αυτό οφείλεται συνήθως στο γεγονός ότι κάθε γλώσσα προγραμματισμού περιέχει στοιχεία από πολλές προηγούμενες γλώσσες, συνδυάζοντας μεν τα θετικά στοιχεία και προσθέτοντας δε, νέα στοιχεία. Όταν μια γλώσσα εμφανίζει χαρακτηριστικά που έχουν θετική αποδοχή ή είναι ευρέως χρησιμοποιούμενα από πολλούς προγραμματιστές, συνήθως υιοθετούνται και από άλλες γλώσσες μεταγενέστερες και μη, ακόμα και αν αυτές οι γλώσσες ανήκουν σε διαφορετική κατηγορία.

Η κατηγοριοποίηση συναντά περισσότερες δυσκολίες και γίνεται ακόμα πιο περίπλοκη για το λόγο ότι πολλές γλώσσες συνήθως ανήκουν σε παραπάνω από μία κατηγορίες. Για παράδειγμα, η C# είναι τόσο αντικειμενοστραφής (object-oriented) όσο και παράλληλη γλώσσα (multithreading), δεδομένου ότι υποστηρίζει την οργάνωση των δεδομένων και υπολογισμών σε κλάσεις και αντικείμενα, αλλά επιτρέπει επίσης και την δημιουργία προγραμμάτων με ταυτόχρονα νήματα (threads) που εκτελούνται παράλληλα.

Δεδομένης της δυσκολίας στην κατηγοριοποίηση που αναφέραμε, οι γλώσσες προγραμματισμού κατηγοριοποιούνται με διάφορους τρόπους. Οι συνηθέστεροι τρόποι είναι:

- με βάση τον τρόπο οργάνωσης των δεδομένων.
- με βάση τον στόχο που έχει η κάθε γλώσσα.
- με βάση τον τρόπο που περιγράφουν το ζητούμενο αποτέλεσμα.

Υπάρχουν σχεδόν 50 κατηγορίες γλωσσών προγραμματισμού, που όπως αναφέραμε παραπάνω πολλές γλώσσες προγραμματισμού ανήκουν σε περισσότερες από μια κατηγορίες. Μερικές από αυτές τις κατηγορίες είναι :

- Διαδικαστικός Προγραμματισμός (Procedural Programming)
- Λειτουργικός Προγραμματισμός (Functional Programming)
- Αντικειμενοστραφής Προγραμματισμός (Object Oriented Programming)
- Προγραμματισμός σεναρίων (Scripting Programming)
- Λογικός Προγραμματισμός

### 1.2.1 Διαδικαστικός Προγραμματισμός (Procedural Programming)

Ο διαδικαστικός προγραμματισμός είναι μια προσέγγιση προγραμματισμού, που προέρχεται από τον επιτακτικό προγραμματισμό, βασισμένος στην έννοια της κλήσης διαδικασίας. Οι διαδικασίες αυτές συχνά αναφέρονται και ως ρουτίνα, υπορουτίνα, μέθοδος ή και συνάρτηση περιέχουν απλώς μια σειρά από προγραμματιστικές εντολές που πρέπει να εκτελεστούν. Κατά τη διάρκεια της εκτέλεσης ενός προγράμματος οποιαδήποτε ρουτίνα μπορεί να κληθεί σε οποιοδήποτε σημείο. Οι γλώσσες προγραμματισμού Fortran, ALGOL, COBOL, PL/I και BASIC που έκαναν την εμφάνιση τους περίπου το 1957-1964 καθώς και οι Pascal και C που δημοσιεύθηκαν περίπου 1970-1972, ακολουθήσαν την προσέγγιση του διαδικαστικού προγραμματισμού.

### 1.2.2 Συναρτησιακός Προγραμματισμός (Functional Programming)

Ο συναρτησιακού προγραμματισμός είναι μια προσέγγιση προγραμματισμού στην οποία προσπαθούμε να εκφράσουμε τα πάντα σε λειτουργίες καθαρού μαθηματικού στυλ. Είναι ένας δηλωτικός τύπος (declarative) στυλ προγραμματισμού. Η κύρια εστίασή του είναι στο "τι να λύσει" σε αντίθεση με ένα επιτακτικό (imperative) στυλ όπου η κύρια εστίαση είναι "πώς να λύσει". Η διαφορά μεταξύ μιας μαθηματικής συνάρτησης και της έννοιας της "συνάρτησης" που χρησιμοποιείται στον προστακτικό προγραμματισμό είναι ότι οι προστακτικές συναρτήσεις μπορούν να έχουν και κάποιες παρενέργειες, δηλαδή εκτός από την τιμή που μπορεί να επιστρέψει μια συνάρτηση μπορούν να τροποποιήσουν και τιμές άλλων μεταβλητών ή διαφορών άλλων προγραμματιστικών στοιχείων. Παράδειγμα συναρτησιακού προγραμματισμού στην γλώσσα Java :

```
public int plusOne(int x) {
    return x+1;
}
```

### 1.2.3 Αντικειμενοστραφής Προγραμματισμός (Object Oriented Programming)

Ο αντικειμενοστραφής προγραμματισμός είναι μια προσέγγιση προγραμματισμού όπου ο χειρισμός σχετιζόμενων τα δεδομένα και οι διαδικασίες που επιδρούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περικλείει ως αυτόνομη οντότητα με δικιά της ταυτότητα και χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο της κλάσης και συνθέτει ένα πραγματικό στιγμιότυπο στη μνήμη του υπολογιστή, συγκεκριμένου τύπου δεδομένων που ονομάζεται *κλάση* (class). Η κλάση μπορεί να είναι μια σύνθετη ή πιο απλή δομή δεδομένων, και πολλές φορές οριζόμενη από τον προγραμματιστή ή από την ίδια την γλώσσα. Η κλάση καθορίζει με συγκεκριμένα χαρακτηριστικά τόσο τα δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν πάνω σε αυτά. Εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό την παραδοσιακή προσέγγιση του δομημένου προγραμματισμού. Στην παρούσα διπλωματική εργασία για την δημιουργία του παιχνιδιού σε android περιβάλλον γίνεται χρήση της αντικειμενοστραφής γλώσσας προγραμματισμού Java.

### 1.2.4 Προγραμματισμός σεναρίων (Scripting Programming)

Ο προγραμματισμός σεναρίων (γνωστή και ως scripting ή script language) είναι μια αλληλουχία από εντολές κώδικα που μπορούν να εκτελεστούν χωρίς την ανάγκη της μεταγλώττισης αλλά ερμηνεύονται και μεταφράζονται κατά τον χρόνο εκτέλεσης τους σε γλώσσα μηχανής απευθείας από τον πηγαίο κώδικα χρησιμοποιώντας ένα πρόγραμμα που ονομάζεται διερμηνέας. Ο προγραμματισμός σεναρίων αυτοματοποιεί πολλές από τις προγραμματιστικές εργασίες που είναι αναγκαίες και επικρατούν σε άλλες γλώσσες προγραμματισμού και απαλλάσσει τον προγραμματιστή από αυτές. Τέτοιες εργασίες συνήθως μπορεί να είναι οι δηλώσεις τύπων μεταβλητών, η δέσμευση μνήμης για την δημιουργία αντικειμένων, διαφορά συντακτικά σύμβολα καθώς και αλλά πολλά.

### 1.2.5 Λογικός Προγραμματισμός

Ο λογικός προγραμματισμός είναι μια προσέγγιση προγραμματισμού που βασίζεται στη λογική. Αυτό σημαίνει ότι μια γλώσσα προγραμματισμού λογικής εκφράζει τα γεγονότα και τους κανόνες με προτάσεις που πρεσβεύουν την λογική. Ο λογικός προγραμματισμός εξάγει από τα διαθέσιμα δεδομένα και τις διαθέσιμες καταστάσεις λογικά συμπεράσματα. Προκειμένου τα προγράμματα υπολογιστών να κάνουν χρήση του λογικού προγραμματισμού, πρέπει να υπάρχει μια βάση υπάρχουσας λογικής, που ονομάζεται κατηγόρημα. Από τα κατηγορήματα κατασκευάζονται ατομικοί τύποι ή άτομα, τα οποία δηλώνουν πραγματικά γεγονότα.

### 1.2.6 Επιτακτική και Δηλωτική μορφή (Imperative and Declarative form)

Υπάρχουν χιλιάδες διαφορετικές γλώσσες προγραμματισμού και κάθε χρόνο δημιουργούνται όλο και περισσότερες. Εκτός από τις κατηγορίες που αναφέραμε οι γλώσσες προγραμματισμού διακρίνονται σε δυο κατηγορίες ανάλογα με την μορφή που συντάζονται, την επιτακτική μορφή και την δηλωτική. Μια γλώσσα προγραμματισμού μπορεί να υποστηρίζει και τις 2 μορφές σύνταξης όπως η γλώσσα προγραμματισμού C#. Η επιτακτική μορφή καθορίζει την ακολουθία λειτουργιών για την εκτέλεση καθώς και τρόπο με τον οποίο θα επιτευχθεί το επιθυμητό αποτέλεσμα. Η δηλωτική μορφή απο την άλλη καθορίζεται το επιθυμητό αποτέλεσμα, αλλά δεν καθορίζει το πως αυτό θα επιτευχθεί. Στο παρακάτω παράδειγμα σε γλώσσα προγραμματισμού C# υλοποιούμε το ίδιο αποτέλεσμα αλλά με δύο διαφορετικούς τρόπους. Σε μια συλλογή αποθήκευσης δεδομένων, συγκεκριμένα μιας λίστας με το όνομα collection όπου έχουμε δημιουργήσει και αποθηκεύσει αριθμούς από το 1 έως 5, επιθυμούμε να δημιουργήσουμε μια δεύτερη συλλογή αριθμών, δηλαδή μια νέα λίστα, οπού θα περιέχει μόνο τους ζυγούς αριθμούς που υπάρχουν στην πρώτη λίστα, δηλαδή τους αριθμούς 2 και 4.

```
List<int> collection = new List<int> { 1, 2, 3, 4, 5 };
```

Στην πρώτη περίπτωση θα αναλύσουμε την επιτακτική μορφή. Η λογική είναι εξής: κάθε στοιχείο-αριθμό της λίστας (εντολή foreach) θα ελεχθεί για το αν είναι περιττός η άρτιος (εντολή if). Αν ο αριθμός είναι άρτιος θα τον προσθέσουμε (με την μέθοδο Add) στην δεύτερη άδεια λίστα που δημιουργήσαμε με όνομα even. Εδώ είναι προφανές πως καθορίζουμε και το τι θα γίνει (θέλουμε μόνο τους άρτιους αριθμούς) αλλά και το πως θα γίνει (έλεγχος κάθε στοιχειού της λίστας και αν ικανοποιείται η αρχική μας συνθήκη τότε προσθέτουμε τον αριθμό αυτόν στην νέα λίστα). Το σύμβολο % στον προγραμματισμό δηλώνει το υπόλοιπο της διαίρεσης, ενώ το σύμβολο != σημαίνει όχι ίσο (διάφορο).

```
List<int> even = new List<int>();
foreach(var num in collection){
    if (num % 2 == 0)
        even.Add(num);
}
```

Αντίθετα στην δηλωτική μορφή κάνουμε χρήση κατάλληλων εντολών και ουσιαστικά καθορίζουμε μόνο το επιθυμητό. Η χρήση της εντολής *Where number % 2* δηλώνει το "τι" αλλά όχι το "πως".

```
var even = collection.Where(number => number % 2 == 0);
```

Η περιγραφή μιας γλώσσας προγραμματισμού συνήθως χωρίζεται στα δύο στοιχεία της σύνταξης (μορφή) και της σημασιολογικής (έννοια). Ορισμένες γλώσσες ορίζονται από ένα έγγραφο προδιαγραφών (για παράδειγμα, η γλώσσα προγραμματισμού C καθορίζεται από ένα πρότυπο ISO), ενώ άλλες γλώσσες (όπως το Perl) έχουν μια κυρίαρχη εφαρμογή που αντιμετωπίζεται ως αναφορά. Ορισμένες γλώσσες έχουν και οι δύο, με τη βασική γλώσσα που ορίζεται από ένα πρότυπο και τις επεκτάσεις που λαμβάνονται από την κυρίαρχη εφαρμογή να είναι κοινές.

Η θεωρία γλώσσας προγραμματισμού είναι ένα υποπεδίο της επιστήμης των υπολογιστών που ασχολείται με το σχεδιασμό, την εφαρμογή, την ανάλυση, το χαρακτηρισμό και την ταξινόμηση των γλωσσών προγραμματισμού.

## 1.3 JAVA

Η JAVA είναι μια ιδιαίτερα διαδεδομένη αντικειμενστρεφής γλώσσα προγραμματισμού. Με την χρήση της οποίας διασφαλίζεται η δυνατότητα εκτέλεσης των java προγραμμάτων ανεξαρτήτως πλατφόρμας υλικού και λογισμικού. Τα προγράμματα που είναι γραμμένα σε JAVA τρέχουν ακριβώς το ίδιο σε windows, linux, unix και macintosh χωρίς να χρειαστεί να γίνει πάλι μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα που καλείται να τρέξει. Δημιουργήθηκε από την εταιρεία πληροφορικής Sun Microsystems το 1995 και πολύ σύντομα κυριάρχησε στον κόσμο των υπολογιστών συνοδευόμενη από την αρχή «Write once, Run anywhere». Βασικά χαρακτηριστικά της οποίας είναι η αντικειμενοστρέφεια και η δημιούργια ανεξάρτητων εφαρμογών και applets. Το περιβάλλον της Java ως ένα αντικειμενοστρεφές περιβάλλον θεωρεί ότι το πρόγραμμα αποτελείται από αντικείμενα1 , τα οποία ‘συνεργάζονται’ μεταξύ τους για την παροχή των εξυπηρετήσεων τους. Επιγραμματικά υπάρχουν παρακάπω οι βασικές αρχες αντικειμενοστρεφούς προγραμματισμου:

- Κληρονομικότητα
- Πολυμορφισμός
- Υπέρβαση (Overriding)
- Υπερφορτωση (Overloading)

**Κληρονομικότητα:** Αν θεωρήσουμε ένα δέντρο κλάσεων όπου κάθε κλάση (υποκλάση) κληρομονομεί (inherits) τις ιδιότητες και μεθόδους της μητρική της κλάση (υπερκλάση). Με αποτέλεσμα τη δημιουργία αντικειμένων με περισσότερα χαρακτηριστικά και λειτουργίες.

**Πολυμορφισμός:** Η δυνατότητα των αντικειμένων, χρησιμοποιώντας το ίδιο όνομα μεθόδου, να προκαλεί την διαφορετική συμπεριφοράς ανάλογα με το τύπο του αντικειμένου στο οποίο καλείται.

**Υπέρβαση (Overriding):** Μια μέθοδος που ορίζεται στην υπερκλάση μπορούμε να την ξανα ορίσουμε στην υποκλάση με διαφορετικό τρόπο.

**Υπερφορτωση (Overloading):** Η Δυνατότητα που μας δίνει ο εντικειμενστραφής προγραμματισμός να ορίσουμε σε μια κλάση δυο μεθόδους με το ίδιο όνομα αλλά με διαφορετικά ορίσματα.

Το μεγάλο πλεονέκτημα της Java είναι πως ο πηγαίος κώδικας μπορεί να εκτελεστεί από διαφορετικά λειτουργικά συστήματα. Ο κώδικας της Java μετατρέπεται σε ένα object αρχείο το οποίο αποτελείται από byte codes. Με τη χρήση ενός εικονικού μηχανήματος το Java Virtual Machine (JVM) που μεταρέπουν τα byte codes σε οδηγίες για το αντίστοιχο λειτουργικό που θα τα χρησιμοποιεί. Το κύριο πρόγραμμα του JVM που αναλαμβάνει την παραπάνω λειτουργία είναι ο Java Interpreter όπου είναι ενσωματωμένος σε διάφορους web browsers, έτσι ώστε να είναι δυνατή η εκτέλεση Java κώδικα σε μορφή applet.

## 1.4 Λογισμικό

Με τον όρο λογισμικό ή λογισμικό υπολογιστών περιγράφουμε μια συλλογή από εκτελέσιμα προγράμματα και εφαρμογές υπολογιστών και διαδικασιών που εκτελούν ορισμένες εργασίες σε ένα υπολογιστικό σύστημα. Επομένως μπορούμε να πούμε ότι το λογισμικό απαρτίζει τα άυλα μέρη του υπολογιστή. Ένα ευρύτατο φάσμα διαφόρων ηλεκτρονικών προϊόντων και τεχνολογιών καλύπτεται από το λογισμικό. Το λογισμικό αναπτύσσεται με χρήση διαφορετικών ειδών προγραμματισμού, όπως οι γλώσσες προγραμματισμού, οι γλώσσες μορφοποίησης κα. Το λογισμικό εκτελείται συνήθως πάνω από ένα λειτουργικό σύστημα (που είναι επίσης λογισμικό), όπως τα Microsoft Windows και το Linux. Τα λογισμικά που εχουν κατασκευαστεί και προορίζονται για μία αρχιτεκτονική λογισμικού ή μια πλατφόρμα συνήθως δεν μπορεί να λειτουργεί άμεσα και σε άλλες πλατφόρμες ή αρχιτεκτονικές λογισμικού, για παράδειγμα, οι εφαρμογές των Microsoft Windows δεν μπορούν να λειτουργήσουν σε λειτουργικό σύστημα Linux, λόγω των διαφορών που παρατηρούνται στις πλατφόρμες και στα πρότυπά σχεδιασμού τους. Τα λογισμικά μπορούν να λειτουργήσουν σε άλλες αρχιτεκτονική λογισμικού μόνο εάν μεταφερθούν, χρησιμοποιώντας ένα διερμηνέα ή με μεταφορά (port) του πηγαίου κώδικα στην επιθυμητή πλατφόρμα. Οι διαφορετικοί τύποι λογισμικού περιλαμβάνουν τις ιστοσελίδες, τις διάφορες εφαρμογές που υπάρχουν προ εγκατεστημένες στον υπολογιστή μας όπως η αριθμομηχανή και το σημειωματάριο, τα προγράμματα επεξεργασίας κειμένου όπως το Microsoft Word και το OpenOffice καθώς και τα βιντεοπαιχνίδια είτε για προσωπικούς υπολογιστές, είτε για

κονσόλες βιντεοπαιχνιδιών ή και ακόμα για κινητές συσκευές. Στην παρούσα διπλωματική παρουσιάζεται ένα λογισμικό ψυχαγωγίας, δηλαδή ένα βιντεοπαιχνίδι λέξεων για κινητές συσκευές που μπορούν να παίζουν ταυτόχρονα 2 παίχτες ως αντίπαλοι.

## 1.5 Λειτουργικό Σύστημα

Ένα λειτουργικό σύστημα (OS) είναι ένα λογισμικό συστήματος που διαχειρίζεται το υλικό και τους πόρους του υπολογιστή, και παρέχει κοινές υπηρεσίες στα προγράμματα των υπολογιστών. Το λειτουργικό σύστημα λειτουργεί ως ενδιάμεσος υποστηρικτής μεταξύ των προγραμμάτων που τρέχουν σε αυτό αλλά και του υλικού του υπολογιστή και υποστηρίζει εργασίες και λειτουργίες υλικού όπως είσοδος, έξοδος δεδομένων καθώς και δέσμευσης και αποδέσμευσης μνήμης. Τα προγράμματα μπορεί να πραγματοποιήσουν διάφορες κλήσεις συστήματος καθώς όμως το λειτουργικό σύστημα είναι υπεύθυνο για την διαχείριση των προγραμμάτων μπορεί να διακόψει την λειτουργία τους όπου αυτό κρίνει χρήσιμο ώστε να αποτρέψει άλλες δυσλειτουργίες που θα επηρεάσουν ενδεχομένως την ομαλή λειτουργία του υπολογιστή. Τα λειτουργικά συστήματα είναι χρήσιμα και υπάρχουν σε πολλές συσκευές που περιέχουν υλικό υπολογιστή – από σταθερούς και φορητούς υπολογιστές, κινητά τηλέφωνα και κονσόλες βιντεοπαιχνιδιών έως διακομιστές ιστού και υπερπολογιστές. Το λειτουργικό σύστημα που κυριαρχεί στις μέρες μας σε σταθερούς και φορητούς υπολογιστές γενικής χρήσης (προσωπικής ή επαγγελματικής) είναι τα είναι τα Microsoft Windows με μερίδιο αγοράς περίπου 76,45%. Ακολουθεί στην δεύτερη το λειτουργικό σύστημα της macOS της Apple Inc. με μερίδιο αγοράς περίπου (17,72%) και στην τρίτη θέση είναι οι διάφορες ποικιλίες του ανοιχτού κώδικα λογισμικού Linux με μερίδιο αγοράς (1,73%). Στον τομέα τον κινητών συσκευών όπως τα κινητά τηλέφωνα, tablet, smartwatch και μη όπως smart Tv τα λειτουργικά συστήματα που επικρατούν το 2021 είναι το Android με ποσοστό μεριδίου αγοράς (72,44%) ενώ στην δεύτερη θέση ακολουθεί το λειτουργικό σύστημα iOS της Apple Inc. με μερίδιο αγοράς (26,75%) ενώ άλλα λειτουργικά συστήματα ανέρχονται σε μόλις 0,74 τοις εκατό.

## 1.6 Βάση Δεδομένων

Μια Βάση Δεδομένων (DataBase) είναι μία οργανωμένη αποθήκη ή συλλογή δεδομένων που η αποθήκευση και η πρόσβαση γίνεται ηλεκτρονικά από ένα υπολογιστικό

σύστημα μέσω ενός λογισμικού γνωστό και ως Σύστημα διαχείρισης βάσεων δεδομένων (DBMS).Το σύστημα διαχείρισης βάσεων δεδομένων (DBMS) είναι το λογισμικό διαχείρισης που αλληλοεπιδρά με τους χρήστες, τις εφαρμογές και την ίδια την βάση δεδομένων για την εισαγωγή, την ενημέρωση, την διαγραφή, την λήψη καθώς και την ανάλυση των δεδομένων. Το λογισμικό DBMS συνεπώς περιλαμβάνει όλα τα εργαλεία εκείνα καθώς τις βασικές εγκαταστάσεις για τη διαχείριση της βάσης δεδομένων. Τα δεδομένα που αποθηκεύονται συνήθως αποτελούν τρόπους αναπαράστασης εννοιών και γεγονότων όπως παραδείγματος χάρη το όνομα και η φωτογραφία προφίλ ενός χρήστη μιας εφαρμογής κοινωνικών δικτυών όπως είναι το Facebook ή το Instagram. Ειδικές τεχνικές σχεδιασμού και μοντελοποίησης χρησιμοποιούνται όταν οι βάσεις δεδομένων γίνονται πιο περίπλοκες και συνθέτες. Λόγω της αυξημένης χρήσης της τεχνολογίας καθώς και του τεράστιου όγκου των δεδομένων στις μέρες μας, τα δεδομένα πρέπει να συντονιστούν με βέλτιστο τρόπο έτσι ώστε να εντοπίζονται και να αξιοποιούνται εύκολα, γρήγορα και τη στιγμή ακριβώς όπου τα χρειαζόμαστε. Υπάρχουν αρκετοί διαφορετικοί τύποι βάσεων δεδομένων που χρησιμοποιούνται σήμερα. Μερικοί από τους πιο συχνά χρησιμοποιούμενους τύπους συστήματος διαχείρισης βάσεων δεδομένων (DBMS) είναι οι εξής:

- Ιεραρχικές βάσεις δεδομένων
- Σχεσιακές βάσεις δεδομένων

### 1.6.1 Ιεραρχικές βάσεις δεδομένων

Οι ιεραρχικές βάσεις δεδομένων αναπτύχθηκαν στη δεκαετία του 1960 και ακολουθούν την λογική ενός οικογενειακού δέντρου. Ένα μόνο αντικείμενο από την ιεραρχία, το "γονικό" έχει ένα ή περισσότερα αντικείμενα κάτω από αυτό ιεραρχικά, τα "παιδία". Οι θυγατρικές εγγραφές έχουν το πολύ έναν γονέα. Οι ιεραρχικές βάσεις δεδομένων είναι μια σύνθετη δομή πλοήγησης οι οποίες παρέχουν υψηλή απόδοση, καθώς υπάρχει ένας γρήγορος χρόνος ανταπόκρισης και υποβολής ερωτημάτων.

### 1.6.2 Σχεσιακές βάσεις δεδομένων

Οι σχεσιακές βάσεις δεδομένων αναπτυχθήκαν και σχεδιάστηκαν τη δεκαετία του 1970. Είναι οι πιο πολυχρησιμοποιημένες και αρκετά δημοφιλείς βάσεις δεομένων στις μέρες μας. Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν συνήθως τη Δομημένη γλώσσα

ερωτήματος (SQL) για τις διάφορες λειτουργίες και αλληλεπιδράσεις των χρηστών με τα δεδομένα όπως η δημιουργία, η ανάγνωση, η ενημέρωση και η διαγραφή δεδομένων. Οι σχεσιακές βάσεις δεδομένων αποθηκεύουν τις πληροφορίες σε διακριτούς πίνακες, όπου κάθε πίνακας μπορεί να έχει από ένα κελί έως 1024 κελιά, και το πολύ ένα πρωτεύον κλειδί (Primary Key). Το πρωτεύον κλειδί ενός πίνακα είναι η ειδική στήλη εκείνη ή ο συνδυασμός στηλών που έχει σχεδιαστεί για να αναγνωρίζει μοναδικά κάθε εγγραφή πίνακα. Οι σχεσιακές βάσεις δεδομένων επιτρέπουν το συσχετισμό των πινάκων μεταξύ τους μέσω των ξένων κλειδιών (Foreign Keys). Ένα ξένο κλειδί είναι μια στήλη ή μια ομάδα στηλών σε έναν πίνακα σχεσιακής βάσης δεδομένων που παρέχει μια σύνδεση μεταξύ δεδομένων σε δύο πίνακες.

### 1.6.3 Μη σχεσιακές βάσεις δεδομένων

Τα μη σχεσιακή συστήματα διαχείρισης βάσεων δεδομένων αναφέρονται συχνά ως NoSQL. Οι αυξημένες απαιτήσεις των σύγχρονων διαδικτυακών εφαρμογών έφεραν την ανάγκη δημιουργίας και χρήσης μη σχεσιακών μοντέλων διαχείρισης βάσεων δεδομένων. Την τελευταία δεκαετία έχουν παρουσιαστεί διάφορες ποικιλίες τέτοιων μοντέλων βάσεων δεδομένων και κάθε ένα από αυτά αποθηκεύει τις πληροφορίες με διαφορετικό τρόπο.

Τέτοιοι τρόποι αποθήκευσης μπορεί είναι:

- Τα δεδομένα μπορούν να αποθηκευτούν σε ένα δομημένο έγγραφο, παρόμοιο με το αρχείο δομής JSON.
- Τα δεδομένα μπορεί να είναι σε μορφή τιμής - κλειδιού που αντιστοιχίζει ένα μόνο χαρακτηριστικό (το "κλειδί") στην τιμή του.
- Τα δεδομένα μπορούν να αποθηκευτούν σε ένα αρχείο δεδομένων γραφήματος όπου χρησιμοποιούνται κόμβοι για την αναπαράσταση αντικειμένων και άκρων για να γίνει περιγραφή της σχέση μεταξύ τους.

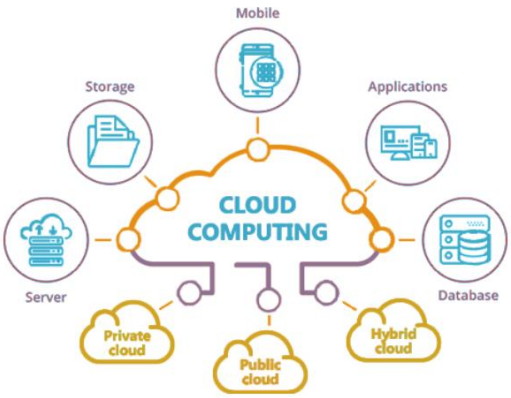
Στην παρούσα διπλωματική για τις ανάγκες του παιχνιδιού θα χρησιμοποιήσουμε μια μη σχεσιακή βάση δεδομένων πραγματικού χρόνου, το Firebase.

## 1.7 Υπολογιστική Νέφους

Τι είναι λοιπόν σύννεφο και κατ' επέκταση cloud; Αρχικά είναι εμφανές πως πρόκειται για μια μεταφορική έννοια η οποία αναφέρεται στους διακομιστές. Στους διακομιστές αυτούς έχουμε πρόσβαση μέσω του Διαδικτύου. Επίσης, αναφερόμαστε στις βάσεις δεδομένων αλλά και στο λογισμικό στα οποία αυτά δύο έχουμε πρόσβαση μέσω αυτών των διακομιστών. Που λοιπόν βρίσκονται όλοι αυτοί οι διακομιστές; Η απάντηση είναι πως βρίσκονται σε όλο τον κόσμο. Για αυτό το λόγο, μέσω του cloud αλλά και του cloud computing, οι χρήστες οι οποίοι μπορεί να είναι είτε ένα μεμονωμένο άτομο, είτε μεγάλες εταιρίες, είτε προγραμματιστές, έχουν την δυνατότητα να μην χειρίζονται οι ίδιοι φυσικούς διακομιστές και λειτουργικό αλλά και να μην υλοποιούν εφαρμογές στα δικά τους συστήματα.

Το cloud παρέχει την δυνατότητα στον εκάστοτε χρήστη να έχει πρόσβαση στα ίδια αρχεία, στις ίδιες εφαρμογές από όποια συσκευή θελήσει. Αυτό συμβαίνει γιατί η αποθήκευση αυτών των εφαρμογών και αρχείων δεν γίνεται στο τοπικό μηχάνημα του χρήστη αλλά σε διακομιστές σε ένα κέντρο δεδομένων. Ένα παράδειγμα προκειμένου να καταλάβουμε στην πράξη τον όρο είναι το Dropbox, όπου το upload των φακέλων, φωτογραφιών, αρχείων , αλλά και οποιοδήποτε άλλου αρχείου μπορεί να γίνει από οπουδήποτε. Αυτό είναι πολύ σημαντικό γιατί κάποιος που δεν έχει την δυνατότητα να σπαταλήσει ένα αρκετά μεγάλο κεφάλαιο για την αγορά μιας εσωτερικής υποδομής, όπως για παράδειγμα μία μικρή επιχείρηση, μπορεί να αναθέσει σε κάποιον άλλον το χτίσιμο της υποδομής αυτής, χωρίς ο ίδιος να σπαταλήσει πόρους. Επίσης, μπορεί ο χρήστης να έχει πρόσβαση από οποιοδήποτε σημείο στον πλανήτη, πράγμα που οδηγεί σε διεθνή αναγνώριση τόσο των εταιριών όσο και των απλών χρηστών.

Στη συνέχεια, είναι άκρως απαραίτητο να κατανοήσουμε πως ακριβώς λειτουργεί το cloud. Ουσιαστικά δουλεύει μέσα από μία υπηρεσία που ονομάζεται εικονικοποίηση ή αλλιώς virtualization. Πρόκειται για έναν νοητό υπολογιστή ακριβώς μέσα σε υπολογιστή! Έχει την δυνατότητα να συμπεριφέρεται όπως ακριβώς ένας υπολογιστής. Για αυτό ακριβώς ονομάζεται εικονική μηχανή. Όταν έχουμε πολλές εικονικές μηχανές οι οποίες λειτουργούν μαζί και εφαρμόζονται σωστά, δεν αλληλοεπιδρούν μεταξύ τους, πράγμα που σημαίνει ότι τα αρχεία, οι φάκελοι και οι εφαρμογές δεν είναι ορατά σε άλλες εικονικές μηχανές και ας βρίσκονται στην ίδια φυσική μηχανή.




Εικόνα 1.7.α - Cloud Computing

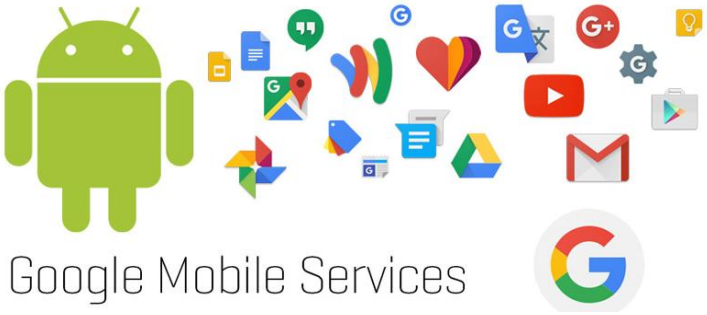
# Κεφάλαιο 2 - Λειτουργικό Σύστημα Android

## 2.1 Τι είναι το Android

Το Android είναι ένα ανοιχτού κώδικα λειτουργικό σύστημα που βασίζεται σε μια τροποποιημένη έκδοση του πυρήνα Linux. Είναι σχεδιασμένο κυρίως για κινητές συσκευές με ή χωρίς οθόνη αφής, όπως smartphone, tablet, smart Tv, smartwatches καθώς και άλλες συσκευές. Το Android αναπτύχθηκε από μια κοινοπραξία διαφόρων εταιριών γνωστή ως Open Handset Alliance και τα εμπορικά δικαιώματα ανήκουν στην Google. Παρουσιάστηκε τον Νοέμβριο του 2007, με την πρώτη εμπορική συσκευή Android, το HTC Dream, να κυκλοφόρησε τον Σεπτέμβριο του 2008.

Εικόνα 2.1.α HTC Dream - wiki

Είναι δωρεάν λογισμικό ανοιχτού κώδικα με άδεια χρήσης βάσει της άδειας Apache. Ο πηγαίος κώδικας του είναι γνωστός ως Android Open Source Project (AOSP). Λόγω της ιδιότητας του λογισμικού, που είναι ανοιχτού κώδικα, πολλές γνωστές εμπορικές εταιρείες τροποποιούν μικρά κομμάτια του λειτουργικού προσαρμόζοντας τα στις δικές τους ανάγκες. Ετσι, οι περισσότερες συσκευές Android αποστέλλονται με προεγκατεστημένο πρόσθετο ιδιόκτητο λογισμικό, με το συνηθέστερο απο όλα το Google Mobile Services (GMS) που περιλαμβάνει βασικές εφαρμογές όπως το Google Chrome, την πλατφόρμα ψηφιακής διανομής εφαρμογών, ταινιων και άλλων ψηφιακων αγαθων, το Google Play, την πλατφόρμα ψηφιακής χαρτογράφησης Google Maps καθώς και όλη την πλατφόρμα ανάπτυξης Google Play Services.




Εικόνα 2.1.β Google Mobile Services

Πάνω από το 70 τοις εκατό των Android smartphone τρέχουν τις υπηρεσίες της Google (Google Play Services). Ορισμένα με προσαρμοσμένο από τον προμηθευτή περιβάλλον εργασίας χρήστη και σουίτα λογισμικού, όπως το TouchWiz και αργότερα OneUI, ένα περιβάλλον εργασίας χρήστη από τη Samsung και το HTC Sense. Ανταγωνιστικά οικοσυστήματα Android και τροποποιημένες διανομές περιλαμβάνουν το Fire OS (που αναπτύχθηκε από την Amazon) ή το LineageOS (πρώην Cyanogenmod). Ωστόσο, το όνομα και το λογότυπο "Android" είναι εμπορικά σήματα της Google που επιβάλλουν πρότυπα για τον περιορισμό των "μη πιστοποιημένων" συσκευών εκτός του οικοσυστήματός τους για τη χρήση επωνυμίας Android. Ο πηγαίος κώδικας έχει χρησιμοποιηθεί για την ανάπτυξη παραλλαγών του Android σε μια σειρά άλλων ηλεκτρονικών ειδών, όπως κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, φορητές συσκευές αναπαραγωγής πολυμέσων, υπολογιστές και άλλα, το καθένα με μια εξειδικευμένη διεπαφή χρήστη. Ορισμένα γνωστά

παράγωγα περιλαμβάνουν το Android TV για τηλεοράσεις και το Wear OS για φορητές συσκευές, που αναπτύχθηκαν και τα δύο από την Google. Τα πακέτα λογισμικού στο Android τα οποία χρησιμοποιούν τη μορφή APK, διανέμονται γενικά μέσω ιδιόκτητων καταστημάτων εφαρμογών όπως το Google Play Store, το Samsung Galaxy Store, το Huawei AppGallery, το Cafe Bazaar και το GetJar ή από πλατφόρμες ανοιχτού κώδικα όπως το Aptoide ή το F-Droid.Το Android είναι το λειτουργικό σύστημα με τις περισσότερες πωλήσεις παγκοσμίως σε smartphone από το 2011 και σε tablet από το 2013. Από τον Μάιο του 2021, έχει πάνω από τρία δισεκατομμύρια ενεργούς χρήστες μηνιαίως, τη μεγαλύτερη εγκατεστημένη βάση οποιουδήποτε λειτουργικού συστήματος και από τον Ιανουάριο του 2021, η Google To Play Store διαθέτει περισσότερες από 3 εκατομμύρια εφαρμογές. Το Android 12, που κυκλοφόρησε στις 4 Οκτωβρίου 2021, είναι η πιο πρόσφατη έκδοση.

## 2.2 Ιστορική Αναδρομή

Το λειτουργικό σύστημα Android (Android Inc) ιδρύθηκε τον Οκτώβριο 2003 στο Palo Alto της Καλιφόρνια από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Η βασική ιδέα της ήταν η ανάπτυξη ενός προηγμένου λειτουργικού συστηματος για ψηφιακές φωτογραφικές μηχανές, ωστόσο η αγορά των φωτογραφικών μηχανών δεν ήταν αρκετά μεγάλη για τους στόχους τους των ιδρυτών και πέντε μήνες αργότερα είχαν εκτρέψει τις προσπάθειές τους και προωθούσαν το Android ως ένα λειτουργικό σύστημα συσκευών χειρός που θα ανταγωνιζόταν ισάξια τα δημοφιλή τότε λειτουργικά συστήματα όπως το Symbian και το Microsoft Windows Mobile.


Εικονα 2.2.α Το λογότυπο του **Android** από το 2007 έως το 2014.

Δυο χρόνια αργότερα τον Ιούλιο του 2005, η Google εξαγόρασε την επιχείρηση Android Inc. για τουλάχιστον 50 εκατομμυριων δολαριων. Σύμφωνα με τον τότε αντιπρόεδρο

εταιρικής ανάπτυξης της Google αυτή ήταν η «καλύτερη συμφωνία ποτέ» όπως εξομολογήθηκε το 2010. Οι βασικοί υπάλληλοι της Android Inc. συμπεριλαμβανομένων και των ιδρυτών της, εντάχθηκαν στην Google ως μέρος της εξαγοράς. Μετά την εξαγορά, η Google, με επικεφαλής τον Rubin ανέπτυξε μια πλατφόρμα κινητής συσκευής που τροφοδοτείται από τον πυρήνα Linux. Εκείνη την εποχή η Google δεν έχει δώσει ξεκάθαρες πληροφορίες σχετικά με το Android παρά μόνο οτι πρόκειται για ένα λογισμικό κινητών συσκευών. Η ανακοίνωση του Apple iPhone τον Ιανουάριο του 2007 ανάγκασε την Google να τροποποιήσει τα έγραφα των προδιαγραφών του Android και να δηλώσει ότι πλέον θα υποστηρίζει και οθόνες αφής μαζί με την παρουσία διακριτών φυσικών κουμπιών. Ο προσανατολισμός για την υποστήριξη του Android μόνο για οθόνες αφής ήρθε το 2008.

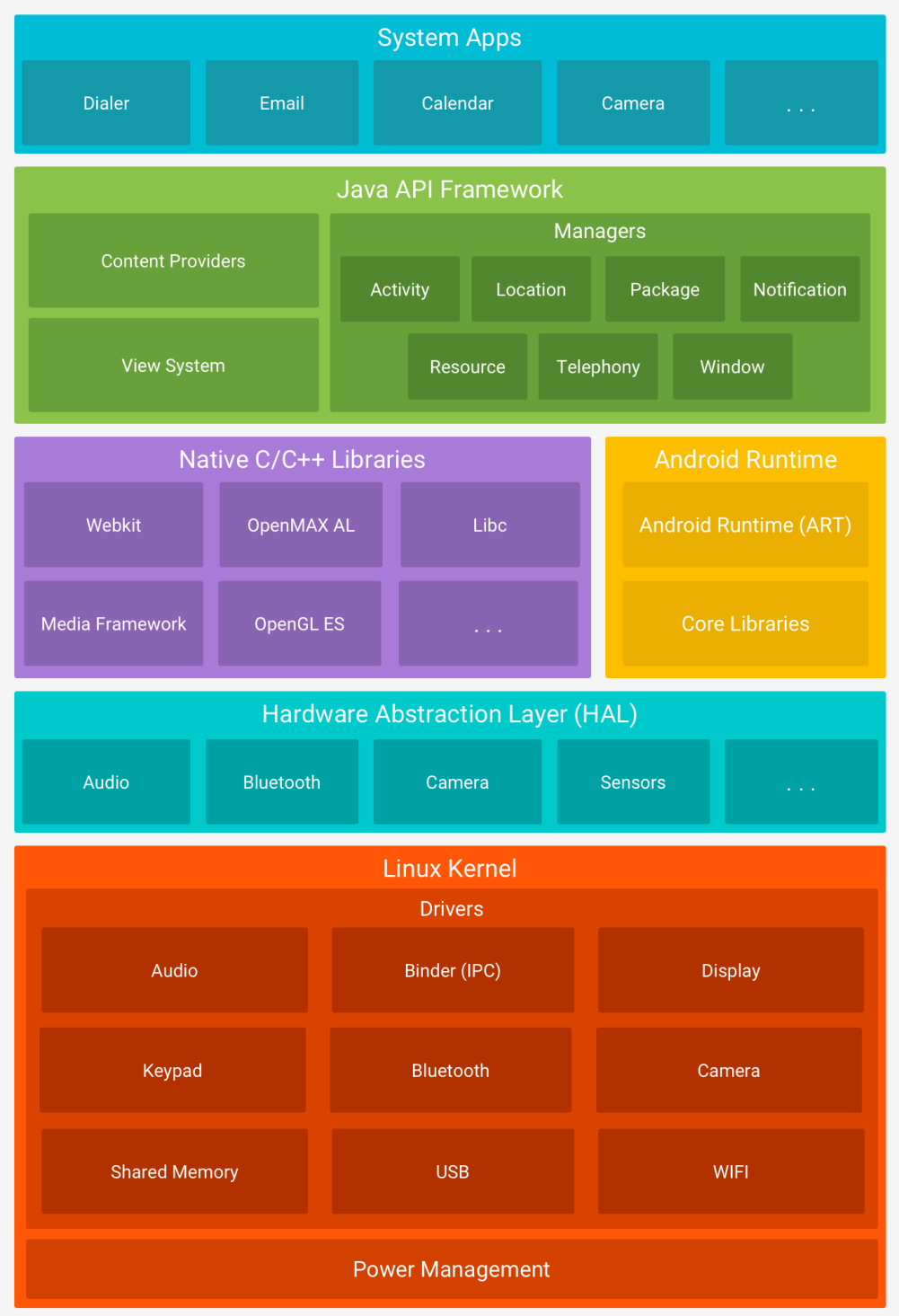
Στις 5 Νοεμβρίου 2007, μια κοινοπραξία εταιρειών τεχνολογίας, μεταξύ άλλων της Google, κατασκευαστών συσκευών όπως η HTC, η Motorola και η Samsung, εταιρείες τηλεπικοινωνίων όπως η Sprint και η T-Mobile, και κατασκευαστές ολοκληρωμένων κυκλωμάτων (chipset) όπως η Qualcomm και η Texas Instruments, συμμάχησαν με κοινό στόχο την δημιουργία της πρώτης ανοικτής και ολοκληρωμένης πλατφόρμας για κινητές συσκευές, και δημιουργήσαν την Open Handset Alliance.

Από το 2008, το Android έχει βγάλει αρκετές αναβαθμίσεις, οι οποίες έχουν τροποποιήσει σταδιακά το λειτουργικό σύστημα, βελτιώνοντας και προσθέτοντας νέα χαρακτηριστικά και διορθώνοντας σφάλματα σε προηγούμενες κυκλοφορίες. Κάθε νέα κυκλοφορία φέρνει το όνομα ενός γλυκού ή επιδορπίου ή και ζαχαρούχου σκευάσματος αλλά διατηρώντας αλφαβητική σειρά από το προηγούμενο του, με τις πρώτες εκδόσεις του Android να ονομάζονται με αυτή τη σειρά "Cupcake", "Donut", "Eclair" και "Froyo". Κατά την ανακοίνωση του Android KitKat το 2013, η Google εξήγησε ότι "Δεδομένου ότι οι κινητές συσκευές κάνουν τη ζωή μας τόσο γλυκιά, κάθε έκδοση Android φέρει και το όνομα ενός επιδορπίου".

## 2.3 Αρχιτεκτονική Πλατφόρμας

Το Android αποτελείται από 5 επίπεδα που αποτελούν τα κύρια συστατικά του λειτουργικού συστήματος. Στο υψηλότερο επίπεδο είναι οι εφαρμογές, αμέσως πιο κάτω είναι το πλαίσιο των εφαρμογών, ακολουθούν οι βιβλιοθήκες και το περιβάλλον χρόνου εκτέλεσης και τέλος στο χαμηλότερο επίπεδο βρίσκεται ο πυρήνας του Linux. Το ακόλουθο διάγραμμα

παρουσιάζει την αρχιτεκτονική του Android ενώ στη συνέχεια αναλύεται κάθε επίπεδο ξεχωριστά. Παρακάτω αναλύουμε τα βασικά επίπεδα του λειτουργικού συστήματος από κάτω προς τα πάνω.




Εικόνα 2.3.α Αρχιτεκτονική Πλατφόρμας

### 2.3.1 Ο πυρήνας Linux

Η βάση του λειτουργικού συστήματος Android είναι ο πυρήνας Linux, στον οποίο εξαρτάται ο χρόνος εκτέλεσης Android (Android Run Time) για τις διάφορες υποκείμενες λειτουργίες, όπως τα νήματα και η διαχείριση μνήμης χαμηλού επιπέδου. Το Android σε επίπεδο λειτουργικού συστήματος, παρέχει την ασφάλεια του πυρήνα Linux, καθώς και την δυνατότητα για μια ασφαλή επικοινωνία μεταξύ διεργασιών (IPC) για να επιτρέψει την ασφαλή επικοινωνία μεταξύ εφαρμογών που εκτελούνται σε διαφορετικές διεργασίες. Επιπλέον από πλευράς υλικού και κατασκευής επιτρέπει στους κατασκευαστές κινητών συσκευών να αναπτύξουν προγράμματα οδήγησης υλικού για έναν γνωστό πυρήνα, αυτού του Linux.

### 2.3.2 Επίπεδο αφαίρεσης υλικού (HAL)

Το επίπεδο αφαίρεσης υλικού (HAL) κατέχει όλες τις διεπαφές που παρέχουν τις δυνατότητες υλικού της συσκευής στο υψηλότερο επίπεδο, του πλαισίου Java API που τις αιτήθηκε. Το HAL αποτελείται από πολλαπλές μονάδες βιβλιοθήκης, καθεμία από τις οποίες υλοποιεί μια διεπαφή για έναν συγκεκριμένο τύπο στοιχείου υλικού, όπως η φωτογραφική μηχανή, η μονάδα bluetooth ή οι αισθητήρες. Όταν ένα Java API πλαισίου κάνει μια κλήση για πρόσβαση στο υλικό της συσκευής, το σύστημα Android φορτώνει τη λειτουργική μονάδα βιβλιοθήκης για αυτό το στοιχείο υλικού και με την σειρά του, παρέχει την βιβλιοθήκη αυτή στο Java API που την αιτήθηκε.

### 2.3.3 Χρόνος εκτέλεσης Android

Για συσκευές που εκτελούν Android έκδοση 5.0 (επίπεδο API 21) ή νεότερη, κάθε εφαρμογή εκτελείται με τη δική της διαδικασία και με τη δική της παρουσία του χρόνου εκτέλεσης Android (ART). Το ART εγγράφεται για την εκτέλεση πολλαπλών εικονικών μηχανών σε συσκευές χαμηλής μνήμης εκτελώντας αρχεία DEX, μια μορφή bytecode σχεδιασμένη ειδικά για Android που έχει βελτιστοποιηθεί για ελάχιστο αποτύπωμα μνήμης. Δημιουργήστε εργαλεία, όπως το d8, μεταγλωττίστε πηγές Java σε DEX bytecode, τα οποία

μπορούν να εκτελούνται στην πλατφόρμα Android.Μερικά από τα σημαντικότερα χαρακτηριστικά του ART περιλαμβάνουν τα εξής:

- Εκ των προτέρων (Ahead of Time) και πάνω-στην-ώρα(Just In Time) συλλογή
- Βελτιστοποιημένη συλλογή απορριμμάτων (GC)
- Σε Android 9 (επίπεδο API 28) και υψηλότερο, μετατροπή των αρχείων εκτελέσιμης μορφής Dalvik (DEX) ενός πακέτου εφαρμογών σε πιο συμπαγή κώδικα μηχανής (machine code).
- Καλύτερη υποστήριξη εντοπισμού σφαλμάτων, συμπεριλαμβανομένου ενός ειδικού προφίλ δειγματοληψίας, λεπτομερών διαγνωστικών εξαιρέσεων και αναφορών σφαλμάτων και της δυνατότητας ρύθμισης σημείων παρακολούθησης για την παρακολούθηση συγκεκριμένων πεδίων

Πριν από την έκδοση Android 5.0 (επίπεδο API 21), το Dalvik ήταν ο χρόνος εκτέλεσης του Android. Εάν μια εφαρμογή λειτουργεί καλά στο ART, τότε θα πρέπει να λειτουργεί και στο Dalvik, αλλά το αντίστροφο μπορεί να μην ισχύει πάντοτε. Το Android περιλαμβάνει επίσης ένα σύνολο βασικών βιβλιοθηκών χρόνου εκτέλεσης που παρέχουν το μεγαλύτερο μέρος της λειτουργικότητας της γλώσσας προγραμματισμού Java, συμπεριλαμβανομένων ορισμένων λειτουργιών γλώσσας Java 8, που χρησιμοποιεί το πλαίσιο Java API.

### 2.3.4 Εγγενείς βιβλιοθήκες C/C++

Πολλά βασικά στοιχεία και υπηρεσίες συστήματος Android, όπως το ART και το HAL, είναι κατασκευασμένα από εγγενή κώδικα που απαιτούν εγγενείς βιβλιοθήκες γραμμένες σε C και C ++. Η πλατφόρμα Android παρέχει API πλαισίου Java για να εκθέσει τη λειτουργικότητα ορισμένων από αυτές τις εγγενείς βιβλιοθήκες σε εφαρμογές. Για παράδειγμα, υπάρχει η δυνατότητα πρόσβασης στο OpenGL ES μέσω του Java OpenGL API του πλαισίου Android για την προσθήκη της υποστήριξης για σχεδίαση και χειρισμό γραφικών 2D και 3D. Επιπλέον δίνεται η δυνατότητα στους προγραμματιστές και όπου απαιτείται κώδικας C ή C++, να χρησιμοποιήσουν το Android NDK για να αποκτήσουν πρόσβαση σε ορισμένες από αυτές τις εγγενείς βιβλιοθήκες πλατφορμών απευθείας από τον native κώδικα.

### 2.3.5 Πλαίσιο API Java

Ολόκληρο το σύνολο δυνατοτήτων του λειτουργικού συστήματος Android είναι διαθέσιμο στους προγραμματιστές μέσω API που είναι γραμμένα στη γλώσσα προγραμματισμού Java. Αυτά τα API αποτελούν τα δομικά στοιχεία που χρειάζονται για την δημιουργία εφαρμογών Android απλοποιώντας την επαναχρησιμοποίηση βασικών, αρθρωτών στοιχείων και υπηρεσιών συστήματος, τα οποία περιλαμβάνουν τα εξής:

- Ένα πλούσιο και επεκτάσιμο σύστημα προβολής για τη δημιουργία του περιβάλλοντος εργασίας χρήστη μιας εφαρμογής, συμπεριλαμβανομένων λιστών, πλέγματα, πλαισίων κειμένου, κουμπιών, ακόμη και ενός ενσωματωμένου προγράμματος περιήγησης ιστού.
- Μια Διαχείριση πόρων, η οποία παρέχει πρόσβαση σε πόρους που δεν είναι κώδικας, όπως μεταφρασμένες συμβολοσειρές, γραφικά και αρχεία διάταξης.
- Μια Διαχείριση ειδοποιήσεων που επιτρέπει σε όλες τις εφαρμογές να εμφανίζουν προσαρμοσμένες ειδοποιήσεις στη γραμμή κατάστασης (status bar).
- Μια Διαχείριση δραστηριότητας που διαχειρίζεται τον κύκλο ζωής των εφαρμογών και παρέχει μια κοινή στοίβα περιήγησης πίσω.
- Υπηρεσίες παροχής περιεχομένου που επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε δεδομένα από άλλες εφαρμογές, όπως η εφαρμογή Επαφές, ή να μοιράζονται τα δικά τους δεδομένα.

Οι προγραμματιστές έχουν πλήρη πρόσβαση στα ίδια API πλαισίου που χρησιμοποιούν οι εφαρμογές συστήματος Android.

### 2.3.6 Εφαρμογές συστήματος

Το Android διαθέτει ένα σύνολο βασικών εφαρμογών για μηνύματα ηλεκτρονικού ταχυδρομείου, μηνύματα SMS, ημερολόγια, περιήγηση στο διαδίκτυο, επαφές και πολλά άλλα. Οι εφαρμογές που περιλαμβάνονται στην πλατφόρμα δεν έχουν ειδική κατάσταση μεταξύ των εφαρμογών που επιλέγει να εγκαταστήσει ο χρήστης. Έτσι, μια εφαρμογή τρίτου μέρους μπορεί να γίνει το προεπιλεγμένο πρόγραμμα περιήγησης ιστού του χρήστη, το SMS messenger ή ακόμα και το προεπιλεγμένο πληκτρολόγιο (ισχύουν ορισμένες εξαιρέσεις, όπως η εφαρμογή Ρυθμίσεις του συστήματος).

Οι εφαρμογές συστήματος λειτουργούν τόσο ως εφαρμογές για τους χρήστες όσο και για την παροχή βασικών δυνατοτήτων στις τις οποίο μπορούν να έχουν πρόσβαση οι προγραμματιστές από τη δική τους εφαρμογή. Για παράδειγμα, αν η εφαρμογή σας θέλει να παραδώσει ένα μήνυμα SMS, δεν χρειάζεται να δημιουργήσετε μόνοι σας αυτήν τη λειτουργία— μπορείτε αντ 'αυτού να αναφέρετε όποια εφαρμογή SMS είναι ήδη εγκατεστημένη για να παραδώσετε ένα μήνυμα στον παραλήπτη που καθορίζετε.

## 2.4 Android Studio

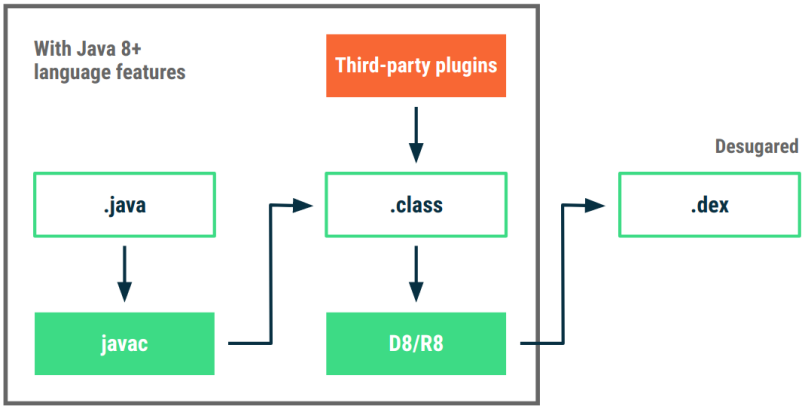
Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών λειτουργικού συστήματος Android. Το Android Studio δίνει τη δυνατότητα στον προγραμματιστή να αναπτύξει το δικό του λογισμικό για λειτουργικό σύστημα Android εντελώς δωρεάν καθώς δεν απαιτείται κάποιο χρηματικό ποσό για την απόκτηση του. Η κυκλοφορία του ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο Google I/O από την Google Product Manager, Katherine Chou. Το Android Studio είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0. Βασισμένο στο λογισμικό IntelliJ IDEA της JetBrains', το Android Studio σχεδιάστηκε αποκλειστικά για την δημιουργία εφαρμογών Android με την χρήση προγραμματισμού και συγκεκριμένα την της γλώσσας προγραμματισμού Java και από τις 7 Μαΐου 2019 υποστηρίζει και την γλώσσα προγραμματισμού Kotlin. Είναι διαθέσιμο για αρκετά λειτουργικά συστήματα όπως Windows, Mac OS X και Linux, και αντικατέστησε τα έως τότε προτεινόμενα εργαλεία Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

## 2.5 Βασικά στοιχεία μιας εφαρμογής Android

Η ανάπτυξη λογισμικού Android είναι η διαδικασία με την οποία δημιουργούνται εφαρμογές για συσκευές που εκτελούν το λειτουργικό σύστημα Android. Σχεδόν όλες οι εφαρμογές Android είναι προγραμματισμένες σε γλώσσα προγραμματισμού Java ή Kotlin. Τα εργαλεία ανάπτυξης λογισμικού Android (SDK) μεταγλωττίζουν τον πηγαίο κώδικα και όποια άλλα δεδομένα και πόρους (resources) χρησιμοποιεί η εφαρμογή και δημιουργούν ένα τελικό αρχείο με κατάληξη .apk. Η κατάληξη apk είναι τα αρχικά της λέξης Android package.

Το εργαλείο d8 που αντικατέστησε το dx (Dex compiler) στην έκδοση του Android Studio 3.1, είναι ένα εργαλείο γραμμής εντολών που χρησιμοποιούν το Android Studio και το Android Gradle Plugin για να μεταγλωττίσουν τον bytecode Java, δηλαδή .class αρχεία που έχουν μεταγλωττιστεί από τον Javac (Java Compiler) σε bytecode DEX που εκτελείται σε συσκευές Android και επιτρέπει στους προγραμματιστές να χρησιμοποιούν τις νέες και των πολλά υποσχόμενων νέων δυνατότητων της έκδοσης 8 της γλώσσας Java στον κώδικα των εφαρμογών.

Τα Android Package είναι η μορφή αρχείου πακέτου εφαρμογών Android που χρησιμοποιείται από το λειτουργικό σύστημα Android και μια σειρά από άλλα λειτουργικά συστήματα που βασίζονται στο λειτουργικό του Android για τη διανομή και εγκατάσταση εφαρμογών, παιχνιδιών καθώς και middleware.




Εικόνα 2.5.α Διαδικασία μεταγλώττισης και παραγωγής εκτελέσιμού αρχείου android.

## 2.6 Συστατικά μέρη μιας Εφαρμογής Android

Όλες οι εφαρμογές Android αποτελούνται από ένα συνδυασμό βασικών δομικών στοιχείων γνωστά και ως συστατικά (components) τα οποία μπορούν να χρησιμοποιηθούν και να κληθούν ξεχωριστά ή και ανεξάρτητα το ένα από το άλλο. Υπάρχουν τέσσερα διαφορετικά είδη συστατικών μιας εφαρμογής. Αυτά είναι :

- Δραστηριότητες (Activities).
- Υπηρεσίες (Services).
- Πάροχοι Περιεχομένων (Content Providers).

- Δέκτες Μετάδοσης (Broadcast Receivers).

Μια εφαρμογή android πρέπει να περιλαμβάνει τουλάχιστον μια δραστηριότητα (Activity). Συνεπώς η Δραστηριότητα είναι το πρώτο και βασικότερο δομικό συστατικό που συναντάει κάθε προγραμματιστής android και αυτό δημιουργείται αυτόματα με την δημιουργία ενός νέου android project (έργου). Η πρώτη δραστηριότητα από προεπιλογή ονομάζεται main Activity (κύρια Δραστηριότητα).

### 2.6.1 Επικοινωνία μεταξύ των συστατικών μιας εφαρμογής.

Ο σχεδιασμός του λειτουργικού συστήματος του Android έχει γίνει με τέτοιο τρόπο ώστε να επιτρέπει σε οποιαδήποτε εφαρμογή να εκκινήσει μια άλλη εφαρμογή ή και να καλέσει κάποιο συστατικό είτε δικό της είτε μιας άλλης. Επίσης είναι λογικό ενα συστατικό (component) να μπορεί να καλέσει ή να εκκινήσει ένα άλλο συστατικό. Για την εκκίνηση αυτή χρειάζεται να χρησιμοποιηθεί μια πρόθεση (Intent). Μια πρόθεση είναι ένα αντικείμενο ανταλλαγής μηνυμάτων που μπορεί να χρησιμοποιηθεί για να επικαλεσθεί μια ενέργεια από ένα άλλο στοιχείο εφαρμογής. Οι Προθέσεις (Intents) χρησιμοποιούνται για την επικοινωνία μεταξύ των συστατικών (components) με διάφορους τρόπους, ωστόσο υπάρχουν τρεις βασικές περιπτώσεις χρήσης:

- Εκκίνηση μιας Δραστηριότητας (Starting an Activity)
- Εκκίνηση μιας Υπηρεσίας (Starting a Service)
- Παροχή μετάδοσης (Delivering a broadcast)

### 2.6.2 Είδη Προθέσεων (Intents Types)

Στο λειτουργικό σύστημα Android υπάρχουν δύο είδη προθέσεων:

- Ρητές Προθέσεις (Explicit intents)
- Έμμεσες Προθέσεις (Implicit intents)

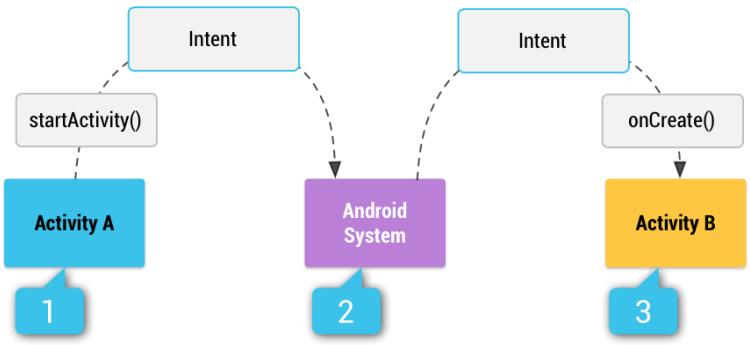
### 2.6.3 Ρητές Προθέσεις (Explicit intents)

Οι ρητές προθέσεις καθορίζουν ποια εφαρμογή ή ποια κλάση θα πραγματοποιήσει την πρόθεση, προσδιορίζοντας είτε το όνομα του πακέτου της εφαρμογής προορισμού είτε ένα πλήρως προσδιορισμένο όνομα κλάσης στοιχείου. Παράδειγμα ρητής πρόθεσης είναι η εκκίνηση μιας υπηρεσίας (service).

### 2.6.4 Έμμεσες Προθέσεις (Implicit intents)

Οι έμμεσες προθέσεις, σε αντίθεση με τις ρητές δεν χρησιμοποιούν το όνομα του πακέτου ή το όνομα της κλάσης ενός συγκεκριμένου συστατικού, αλλά αντίθετα δηλώνουν μια γενική ενέργεια που πρέπει να πραγματοποιηθεί, η οποία επιτρέπει σε ένα συστατικό από μια άλλη εφαρμογή να το χειριστεί. Παράδειγμα έμμεσης πρόθεσης είναι η εμφάνιση στον χρήστη μιας τοποθεσίας του χάρτη.

Στην παρακάτω εικόνα παρουσιάζεται η διαδικασία για το πώς μια σιωπηρή πρόθεση παραδίδεται μέσω του συστήματος για να ξεκινήσει μια άλλη δραστηριότητα. Στο βήμα 1 η δραστηριότητα Α δημιουργεί μια πρόθεση μαζί ένα φίλτρο περιγραφής ενέργειας και την μεταβιβάζει στην μέθοδο startActivity(). Στο βήμα 2 το σύστημα Android αναζητά όλες τις εφαρμογές και προσπαθεί να ταιριάξει την πρόθεση σύμφωνα με το φίλτρο περιγραφής ενέργειας. Όταν υπάρξει αντιστοιχία μεταβαίνουμε στο βήμα 3 το οποίο είναι η εκκίνηση της δραστηριότητας Β από το σύστημα και πυροδοτει την μέθοδο επιστροφής κλήσης onCreate() της δραστηριότητας Β και μεταβιβάζει την πρόθεση.




Εικόνα 2.6.4.α Μεταφορά μηνύματος από μια σιωπηρή πρόθεση

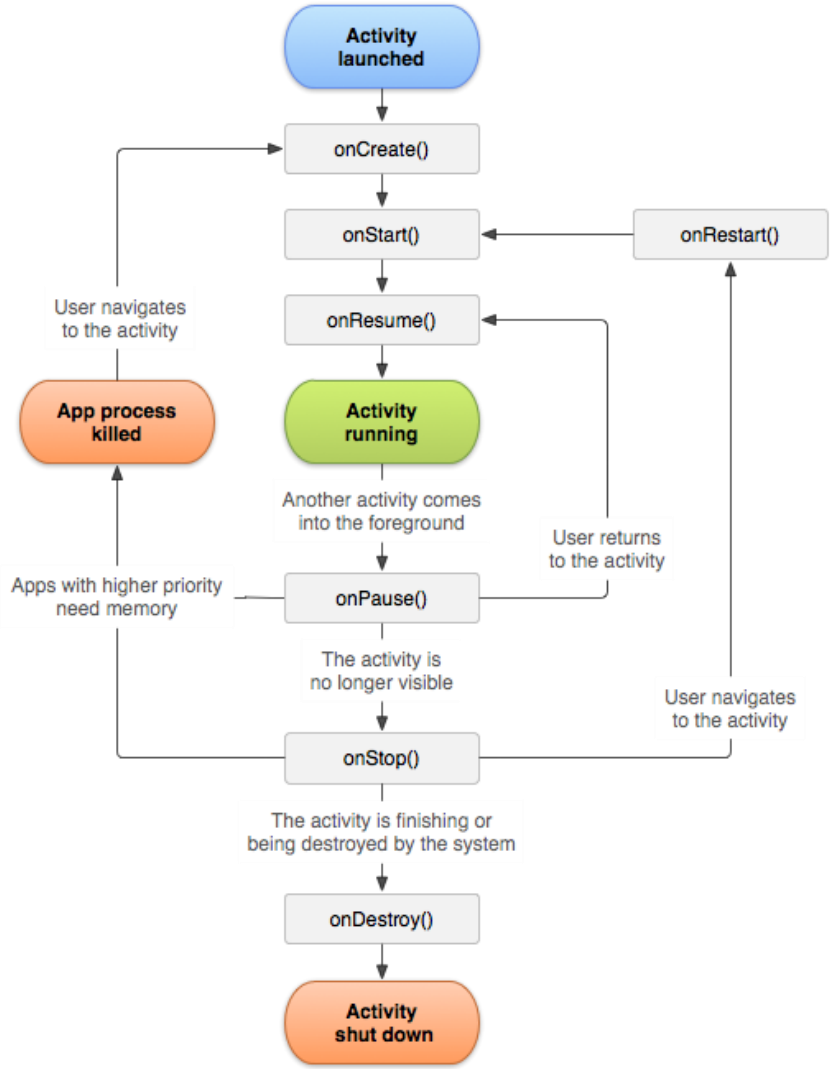
## 2.7 Ανάλυση των βασικών δομικών συστατικών μιας εφαρμογής.

### 2.7.1 Δραστηριότητες

Μια δραστηριότητα (activity) παρέχει μια μοναδική, εστιασμένη οθόνη με την οποία ο χρήστης μπορεί να αλληλεπιδράσει ώστε να πραγματοποιήσει ορισμένες ενέργειες, όπως το πάτημα ενός κουμπιού, η πληκτρολόγηση ενός κειμένου, η πλοήγηση σε έναν χαρτί καθώς και άλλες πολλές ενέργειες. Συνεπώς μπορούμε να ορίσουμε πως μια δραστηριότητα είναι ένα παράθυρο οθόνης που αποτελεί την διεπαφη του χρήστη (user interface) και ορίζει τις ενέργειες εκείνες που πρέπει να πραγματοποιηθούν έπειτα από απαίτηση του χρήστη, όπως παραδείγματος χάρη ποια ενέργεια πρέπει να γίνει έπειτα από το πάτημα ενός κουμπιού.

Η πρώτη δραστηριότητα που εμφανίζεται στον χρήστη κατά το άνοιγμα μιας εφαρμογής τυπικά φέρει το όνομα “κύρια Εφαρμογή” (Main Activity). Ανάλογα το σχεδιασμό και της απαιτήσεις κάθε εφαρμογής μπορεί να υπάρχουν πολλές δραστηριότητες οι οποίες να εναλλάσσονται μεταξύ τους καθ' όλη την διάρκεια χρήσης της εφαρμογής από την χρήστη. Το λειτουργικό σύστημα Android διατηρεί μια στοίβα ιστορικού για τις δραστηριότητες μιας εφαρμογής. Έτσι όταν μια δραστηριότητα εκκινεί μια νέα δραστηριότητα τότε η τρέχων τοποθετείται στην κορυφή της στοίβας και όταν ο χρήστης επιθυμεί να επιστρέψει στην προηγουμένη σελίδα τότε η δραστηριότητα που βρίσκεται στην κορυφή της στοίβας παίρνει την θέση της τρέχοντος.

Κάθε δραστηριότητα έχει ένα κύκλο ζωής όπως παρουσιάζεται στο παρακάτω διάγραμμα ανάλογα με την κατάσταση που βρίσκεται.




Εικόνα 2.7.1.α Κύκλος ζωής Δραστηριότητας

Οι καταστάσεις που μπορεί να βρεθεί μια δραστηριότητα είναι έξι και αυτές είναι:

- Δημιουργίας (Create)
- Εκκίνησης (Start)
- Συνέχειας (Resume)
- Παύσης (Pause)
- Διακοπής (Stop)
- Καταστροφής (Destroy)

Η κλάση Activity παρέχει ένα βασικό σύνολο έξι επιστροφών κλήσης (callbacks), μια για κάθε κατάσταση στην οποία μπορεί να βρεθεί μια δραστηριότητα σε όλο τον κύκλο ζωής της. Αυτές οι επιστροφές κλήσης (callbacks) φέρουν το αντίστοιχο όνομα της κατάστασης. Οι έξι μέθοδοι επιστροφές κλήσης είναι:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()

### 2.7.2 onCreate

Αυτή η μέθοδος επιστροφή κλήσης, πυροδοτείτε όταν το σύστημα δημιουργεί για πρώτη φορά την δραστηριότητα και η δραστηριότητα εισέρχεται στην κατάσταση "Δημιουργίας". Στη μέθοδο onCreate(), συνήθως εκτελείται βασική λογική και ενέργειες εκκίνησης της δραστηριότητας, όπως αρχικοποίηση διάφορων μεταβλητών ή τοποθέτηση διαφορών δεδομένων σε λίστες καθώς και άλλα πολλά. Αυτές οι ενέργειες πρέπει να συμβούν μόνο μία φορά για ολόκληρη τη διάρκεια ζωής της δραστηριότητας.

### 2.7.3 onStart

Αυτή η μέθοδος επιστροφή κλήσης, καλείτε από το σύστημα όταν η δραστηριότητα έχει φύγει από την κατάσταση της “Δημιουργίας”, και συγκείμενα μετά από την κλήση της μεθόδου, onCreate. Οταν η δραστηριότητα εισέρχεται στην κατάσταση “Έναρξης” γίνεται ορατή στον χρήστη. Η κλήση της μεθόδου onStart() προετοιμάζεται τη δραστηριότητα να εισέλθει στο προσκήνιο και να γίνει διαδραστική, έτοιμη για την αλληλεπίδραση της με τον χρήστη. Η μέθοδος onStart() ολοκληρώνεται πολύ γρηγορά όπως και η κατάσταση της “Δημιουργίας”.

### 2.7.4 onResume

Αυτή η μέθοδος επιστροφή κλήσης, πυροδοτείτε από το σύστημα όταν η δραστηριότητα εισέλθει στην κατάσταση "Συνέχισης". Στην κατάσταση αυτή, η δραστηριότητα έρχεται στο προσκήνιο και η εφαρμογή αλληλοεπιδρά με τον χρήστη. Η εφαρμογή παραμένει σε αυτήν την κατάσταση μέχρι να συμβεί κάτι που θα αλλάξει την κατάσταση της δραστηριότητας και θα απομακρύνει την εστίαση από αυτήν. Ένα τέτοιο συμβάν μπορεί να είναι, για παράδειγμα, μια τηλεφωνική κλήση, ή η εκκίνηση και η πλοήγηση του χρήστη σε μια άλλη δραστηριότητα.

### 2.7.5 onPause

Η μέθοδος αυτή, καλείτε από το σύστημα με την πρώτη ένδειξη ότι ο χρήστης έχει εγκαταλείψει την εφαρμογή και ταυτόχρονα την τρέχων δραστηριότητά. Τις περισσότερες φορές δεν σημαίνει ότι η εφαρμογή εγκαταλείπεται ή ότι δραστηριότητα καταστρέφεται, υποδεικνύει όμως ότι η δραστηριότητα δεν βρίσκεται πλέον στο προσκήνιο και στο επίκεντρο της προσοχής του χρήστη. Το λειτουργικό σύστημα του Android υποστηρίζει την λειτουργία των πολλαπλών παράθυρων και έτσι είναι λογικό μια δραστηριότητα να εισέλθει στην κατάσταση 'Παύσης". Η μέθοδος onPause() χρησιμοποιείται συχνά για την διακοπή ή για την ρύθμιση κάποιων λειτουργιών ή ενεργειών που ήταν ενεργείς καθώς ήταν ενεργή και η δραστηριότητα.

### 2.7.6 onStop

Αυτή η μέθοδος επιστροφή κλήσης, πυροδοτείτε από το σύστημα όταν η δραστηριότητά δεν είναι πλέον ορατή στο χρήστη, και έχει εισέλθει στην κατάσταση "Διακοπή". Αυτό μπορεί να συμβεί, για παράδειγμα, όταν μια νέα δραστηριότητα που ξεκίνησε πρόσφατα καλύπτει ολόκληρη την οθόνη και βρίσκεται στο επίκεντρο του χρήστη. Το σύστημα μπορεί επίσης να καλέσει την μέθοδο onStop() όταν η δραστηριότητα έχει ολοκληρωθεί και πρόκειται να τερματιστεί.

### 2.7.7 onDestroy

Αυτή η μέθοδος επιστροφή κλήσης, καλείτε από το σύστημα πριν από την καταστροφή της δραστηριότητας. Οι λόγοι που το σύστημα ενεργοποιεί αυτήν την επιστροφή κλήσης συνήθως είναι:

- Η δραστηριότητα έχει ολοκληρωθεί και πλέον δεν βρίσκεται στο παρασκήνιο και συνεπώς στο επίκεντρο της χρήσης από τον χρήστη.
- Το σύστημα καταστρέφει προσωρινά τη δραστηριότητα λόγω αλλαγής ρύθμισης παραμέτρων όπως η περιστροφή της συσκευής ή λειτουργία ταυτοχρόνων παραθύρων.

## 2.8 Υπηρεσίες (Services)

Μια υπηρεσία είναι ένα από τα κύρια δομικά συστατικά (components) μιας εφαρμογής που μπορεί να εκτελεί μακροχρόνιες εργασίες στο παρασκήνιο ανεξάρτητα από το αν η εφαρμογή είναι ενεργή ή όχι. Μια υπηρεσία δεν διαθέτει περιβάλλον εργασίας χρήστη (user interface). Η υπηρεσία ενδέχεται να συνεχίσει να εκτελείται για κάποιο χρονικό διάστημα, ακόμη και μετά τη μετάβαση του χρήστη σε άλλη εφαρμογή ή το κλείσιμο της ιδίας της εφαρμογής. Μια υπηρεσία εκτελείται στο κύριο νήμα της εφαρμογής, επομένως από προεπιλογή δεν δημιουργεί δικό της νήμα. Είναι στο χέρι του προγραμματιστή και της εμπειρίας που διαθέτει για το αν θα δημιουργήσει ένα νέο νήμα εντός της υπηρεσίας για την εκτέλεση εντατικών λειτουργιών ή λειτουργίες αποκλεισμού. Επίσης είναι ευθύνη του προγραμματιστή, να σταματήσει την υπηρεσία όταν ολοκληρώσει την εργασίας της καλώντας είτε τη μέθοδο stopSelf() είτε τη μέθοδο stopService().

### 2.8.1 Είδη Υπηρεσιών

Στο λειτουργικό σύστημα Android υπάρχουν τριών ειδών υπηρεσιών:

- Υπηρεσίες προσκήνιου (Foreground Services)
- Υπηρεσίες παρασκηνίου (Background Services)
- Υπηρεσίες Δέσμευσης (Bound Services)

### 2.8.2 Υπηρεσίες προσκήνιου (Foreground Services)

Μια υπηρεσία προσκήνιου εκτελεί εργασίες που είναι αισθητές στο χρήστη. Για παράδειγμα, μια εφαρμογή αναπαραγωγής ήχου χρησιμοποιεί μια υπηρεσία προσκηνίου για την αναπαραγωγή ενός κομματιού ήχου. Οι υπηρεσίες προσκηνίου συνεχίζουν να εκτελούνται ακόμα και όταν ο χρήστης δεν χρησιμοποιεί με την εφαρμογή. Για τον λόγο αυτό οι υπηρεσίες προσκηνίου πρέπει να εμφανίζουν μια ειδοποίηση ώστε να γνωστοποιείται στον χρήστη ότι η υπηρεσία είναι ενεργή και εκτελείται. Η ειδοποίηση αυτή δεν μπορεί να απορριφθεί ή να αφαιρεθεί, εκτός εάν η υπηρεσία διακοπεί ή αφαιρεθεί από το προσκήνιο.

### 2.8.3 Υπηρεσίες παρασκήνιου (Background Services)

Μια υπηρεσία παρασκήνιου εκτελεί εργασίες που δεν είναι αισθητές στον χρήστη. Για παράδειγμα, εάν μια εφαρμογή χρησιμοποιούσε μια υπηρεσία για τη συμπύκνωση του χώρου αποθήκευσης, αυτή η υπηρεσία θα ήταν συνήθως μια υπηρεσία παρασκηνίου. Από προεπιλογή οι υπηρεσίες που δημιουργούνται είναι υπηρεσίες παρασκήνιου, πράγμα που σημαίνει ότι εάν το σύστημα χρειάζεται να τις τερματίσει για να ανακτήσει περισσότερη μνήμη όπως για να εμφανιστεί μια μεγάλη σελίδα σε ένα πρόγραμμα περιήγησης ιστού), μπορούν να τερματιστούν χωρίς μεγάλο κόστος.

### 2.8.4 Υπηρεσίες Δέσμευσης (Bound Services)

Μια υπηρεσία δεσμεύεται όταν ένα οποιοδήποτε συστατικό εφαρμογής συνδέεται σε αυτήν καλώντας την μέθοδο bindService(). Μια δεσμευμένη υπηρεσία προσφέρει ένα περιβάλλον εργασίας που ακολουθεί το μοντέλο αρχιτεκτονικής πελάτη - διακομιστή το οποίο επιτρέπει στα συστατικά να αλληλοεπιδρούν με την υπηρεσία, να στέλνουν αιτήσεις, να λαμβάνουν αποτελέσματα και ακόμη να πραγματοποιούν τις εργασίες αυτές μέσου του συνόλου μηχανισμών, γνωστό ως “αλληλεπίδραση επικοινωνίας “ (IPC). Η εκτέλεση μιας δεσμευμένης υπηρεσίας πραγματοποιείται μόνο όταν ένα άλλο συστατικό εφαρμογής είναι συνδεδεμένο σε αυτό. Πολλά στοιχεία εφαρμογής μπορούν να συνδεθούν με την υπηρεσία

ταυτόχρονα, αλλά όταν όλα αυτά αποδεσμευτούν από την υπηρεσία, τότε η υπηρεσία καταστρέφεται.

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

Εικόνα 2.8.4.α Δήλωση Service στο αρχείο manifest.XML

### 2.8.5 Κύκλος ζωής και μέθοδοι υπηρεσιών.

Μια υπηρεσία (service) διαχειρίζεται τον δικό της κύκλο ζωής. Το λειτουργικό σύστημα Android δεν σταματά ή καταστρέφει μια υπηρεσία εκτός εάν πρέπει να ανακτήσει μέρος της μνήμη του συστήματος. Ο κύκλος ζωής μιας υπηρεσίας περιλαμβάνει κάποιες μεθόδους. Οι πιο σημαντικές μέθοδοι επιστροφής κλήσης που ένας προγραμματιστής πρέπει να υλοποιήσει όταν δημιουργεί μια υπηρεσία είναι:

- onStartCommand()
- onBind()
- onCreate()
- onDestroy()

### 2.8.5.1 onStartCommand()

Το σύστημα καλεί τη μέθοδο επιστροφής κλήσης onStartCommand() όταν ένα άλλο συστατικό εφαρμογής (όπως μια δραστηριότητα) ζητήσει την εκκίνηση μιας υπηρεσίας (service) μέσω της κλήσης της μεθόδου startService(). Όταν εκτελείται αυτή η μέθοδος, η υπηρεσία ξεκινά και μπορεί να εκτελείται στο παρασκήνιο επ' αόριστόν.

### 2.8.5.2 onBind()

Το σύστημα καλεί αυτήν τη μέθοδο επιστροφής κλήσης onBind() όταν ένα άλλο συστατικό εφαρμογής θέλει να συνδεθεί με την υπηρεσία καλώντας την μέθοδο bindService(). Για την δημιουργία μιας δεσμευμένης υπηρεσίας πρέπει να οριστεί από τον προγραμματιστή μια διεπαφή που θα καθορίζει τον τρόπο επικοινωνίας του πελάτη (client) με την υπηρεσία αυτή. Έτσι κατά την κλήση της μεθόδου bindService() πρέπει να γίνει μια επιστροφή απόδοσης της διεπαφης IBinder. Εάν ένα συστατικό εφαρμογής καλέσει τη bindService() για να δημιουργήσει την υπηρεσία και η onStartCommand() δεν κληθεί, η υπηρεσία εκτελείται μόνο εφόσον το στοιχείο είναι συνδεδεμένο σε αυτήν. Αφού η υπηρεσία αποδεσμευτεί από όλα τα συστατικά που είχαν συνδεθεί σε αυτήν, το σύστημα καταστρέφει την υπηρεσία (service).

### 2.8.5.3 onCreate()

Το σύστημα καλεί αυτήν τη μέθοδο επιστροφής κλήσης για να εκτελέσει διαδικασίες εγκατάστασης μίας χρήσης κατά την αρχική δημιουργία της υπηρεσίας (πριν καλέσει είτε την onStartCommand() είτε την onBind()). Εάν η υπηρεσία εκτελείται ήδη, αυτή η μέθοδος δεν καλείται.

### 2.8.5.5 onDestroy()

Το σύστημα καλεί αυτήν τη μέθοδο όταν η υπηρεσία δεν χρησιμοποιείται πλέον και καταστρέφεται ή η ίδια υπηρεσία καλέσει την μέθοδο stopSelf() ή ένα κάποιο άλλο συστατικό τη σταματήσει καλώντας την μέθοδο stopService(). Οι προγραμματιστές συνήθως παραγράφουν (override) την μέθοδο onDestroy για να καθαρίσουν τυχόν πόρους, όπως νήματα, εγγεγραμμένους ακροατές (listeners) ή δέκτες (receivers).

## 2.9 Πάροχοι Περιεχομένων (Content Providers).

Ο πάροχος περιεχομένου (content provider) ελέγχει ένα κοινό σύνολο δεδομένων μιας εφαρμογής οπου μπορεί να αποθηκεύσει δεδομενα σε μια βάση δεδομένων SQLite, στο

σύστημα αρχείων του λειτουργικού, στον ιστό ή σε οποιαδήποτε άλλη μόνιμη τοποθεσία αποθήκευσης στην οποία μπορεί να έχει πρόσβαση μια εφαρμογή.

Μέσω του παρόχου περιεχομένου, άλλες εφαρμογές μπορούν να έχουν πρόσβαση στα δεδομένα όπως να υποβάλουν SQL ερωτήματα ή να επεξεργαστούν τα δεδομένα αυτά εάν φυσικά το επιτρέπει ο ίδιος ο πάροχος περιεχομένου. Για παράδειγμα, το λειτουργικό σύστημα Android παρέχει έναν πάροχο περιεχομένου που διαχειρίζεται τα στοιχεία επικοινωνίας του χρήστη. Αυτό έχει σαν αποτέλεσμα οποιαδήποτε εφαρμογή με τα κατάλληλα δικαιώματα να μπορεί να ζητήσει από τον πάροχο περιεχομένου, όπως για παράδειγμα το ContactsContract.Data, να προσπελάσει και να επεξεργαστεί πληροφορίες για ένα συγκεκριμένο χρήστη.

Είναι παραπλανητικό να σκεφτόμαστε έναν πάροχο περιεχομένου ως ένα αφηρημένο μοντέλο σχεδίασης για μια βάση δεδομένων, επειδή υπάρχουν πολλές διεπαφές προγραμματισμού εφαρμογών (APIs) που υποστηρίζουν τις αλληλεπιδράσεις με τις βάσεις δεδομένων και έχουν ενσωματωθεί σε αυτήν την κοινή περίπτωση. Από την οπτική του σχεδιασμού του συστήματος οι πάροχοι περιεχομένου έχουν διαφορετικό σκοπό . Για το σύστημα, ένας πάροχος περιεχομένου είναι το σημείο εισόδου της εφαρμογή για τη διαχείριση δεδομένων που προσδιορίζονται από ένα σχήμα ενιαίο αναγνωριστικό πόρων (Uniform Resource Identifier - URI).

Έτσι, μια εφαρμογή μπορεί να δώσει πρόσβαση σε δικά της δεδομένα μέσω των πάροχων περιεχομένου και των URI, διανέμοντας αυτά τα URI σε άλλες οντότητες που μπορούν με τη σειρά τους να διαχειριστούν δεδομένα. Υπάρχουν μερικά συγκεκριμένα πράγματα που επιτρέπουν στο σύστημα να κάνει στη διαχείριση μιας εφαρμογής:

- Η εκχώρηση ενός URI δεν απαιτεί η εφαρμογή να συνεχίσει να εκτελείται, επομένως τα URI μπορούν να διατηρηθούν μετά την έξοδο από τις εφαρμογές που κατέχουν. Το σύστημα χρειάζεται μόνο να βεβαιωθεί ότι μια εφαρμογή που κατέχει εξακολουθεί να εκτελείται όταν πρέπει να ανακτήσει τα δεδομένα της εφαρμογής από το αντίστοιχο URI.
- Αυτά τα URI παρέχουν επίσης ένα σημαντικό λεπτομερές μοντέλο ασφάλειας. Για παράδειγμα, μια εφαρμογή μπορεί να προσθέσει (add) το URI για μια εικόνα που έχει στο πρόχειρο (clipboard), αλλά να αφήσει τον πάροχο περιεχομένου της κλειδωμένο, έτσι ώστε άλλες εφαρμογές να μην μπορούν να έχουν ελεύθερη πρόσβαση σε αυτό. Όταν μια δεύτερη εφαρμογή επιχειρεί να αποκτήσει πρόσβαση σε αυτό το URI από το

πρόχειρο, το σύστημα μπορεί να επιτρέψει σε αυτήν την εφαρμογή να έχει πρόσβαση στα δεδομένα μέσω μιας προσωρινής παραχώρησης άδειας URI, έτσι ώστε να επιτρέπεται η πρόσβαση στα δεδομένα μόνο υπό αυτό το URI, αλλά καμία άλλη πρόσβαση στη δεύτερη εφαρμογή .

Οι πάροχοι περιεχομένου είναι επίσης χρήσιμοι για την ανάγνωση και τη σύνταξη δεδομένων που είναι ιδιωτικά σε μια εφαρμογή και δεν είναι κοινά.

## 2.10 Δέκτες Μετάδοσης (Broadcast Receivers).

Ο δέκτης εκπομπής (broadcast receivers) είναι ένα συστατικό στοιχείο εφαρμογής που επιτρέπει στο λειτουργικό σύστημα να παραδίδει συμβάντα, γεγονότα, ή ειδοποιήσεις στην εφαρμογή εκτός μιας κανονικής ροής χρήστη, επιτρέποντας στην εφαρμογή να υπακούει σε ανακοινώσεις εκπομπής σε όλο φάσμα του λειτουργικού σύστημα, όταν για παράδειγμα η συσκευή βρίσκεται σε αναμονή (lock screen).

Επειδή οι δέκτες εκπομπής είναι μια άλλη καλά σχεδιασμένη είσοδος στην εφαρμογή, το σύστημα μπορεί να ανταποκρίνεται σε ανακοινώσεις εκπομπών ακόμη και σε εφαρμογές που δεν είναι σε λειτουργία. Έτσι, για παράδειγμα, μια εφαρμογή που εμφανίζει στον χρήστη ειδοποιήσεις για επερχόμενα προγραμματισμένα γεγονότα ή συμβάντα, όπως για παράδειγμα μια εφαρμογή ξυπνητήρι ή μια εφαρμογή ατζέντα, μετά την παράδοση αυτού του συναγερμού σε έναν BroadcastReceiver της εφαρμογής, δεν χρειάζεται η εφαρμογή να παραμείνει σε λειτουργία μέχρι ο συναγερμός να χτυπήσει και να ειδοποιηθεί ο χρήστης.

Πολλές εκπομπές προέρχονται από το ίδιο το σύστημα ή από εφαρμογές του ίδιου του συστήματος για παράδειγμα, μια εκπομπή που ανακοινώνει ότι η οθόνη έχει απενεργοποιηθεί, η μπαταρία είναι χαμηλή ή έχει τραβήξει μια φωτογραφία. Οι εφαρμογές μπορούν επίσης να ξεκινήσουν εκπομπές—για παράδειγμα, για να ενημερώσουν άλλες εφαρμογές ότι ορισμένα δεδομένα έχουν ληφθεί στη συσκευή και είναι διαθέσιμα για χρήση. Αν και οι δέκτες εκπομπής δεν εμφανίζουν διεπαφή χρήστη, ενδέχεται να δημιουργήσουν μια ειδοποίηση γραμμής κατάστασης για να ειδοποιούν τον χρήστη όταν συμβαίνει ένα συμβάν μετάδοσης. Συνηθέστερα, ωστόσο, ένας δέκτης εκπομπής είναι απλώς μια πύλη σε άλλα εξαρτήματα και προορίζεται να κάνει μια πολύ ελάχιστη ποσότητα εργασίας. Για παράδειγμα, μπορεί να προγραμματίσει ένα JobService για να εκτελέσει κάποια εργασία με βάση το συμβάν με το JobScheduler

Ένας δέκτης εκπομπής υλοποιείται ως υποκατηγορία του BroadcastReceiver και κάθε εκπομπή παραδίδεται ως αντικείμενο Intent. Για περισσότερες πληροφορίες, ανατρέξτε στην κατηγορία BroadcastReceiver.

## 2.11 Android Layouts

Ένα σχέδιο (Layout) ορίζει τη δομή για μια διεπαφή χρήστη (User Interface) σε μια εφαρμογή, όπως σε μια δραστηριότητα. Όλα τα στοιχεία στο σχέδιο δημιουργούνται και χρησιμοποιούνται ακολουθώντας μια ιεραρχία αντικειμένων View και ViewGroup. Μια προβολή (View) είναι η βασική κλάση για γραφικά στοιχεία όπου σχεδιάζει ή δημιουργεί κάτι με το οποίο ο χρήστης μπορεί να δει και να αλληλεπιδράσει. Συνήθως καταλαμβάνει μια ορθογώνια περιοχή στην οθόνη και είναι υπεύθυνη για διάφορα γραφικά στοιχεία, τα οποία χρησιμοποιούνται για τη δημιουργία διαδραστικών στοιχείων διεπαφής χρήστη. Αντίθετα ένα ViewGroup είναι ένα αόρατο κοντέινερ ή μια ομάδα από προβολές που ορίζει τη δομή διάταξης για όλες τις προβολές καθώς άλλα αντικείμενα της κλάσης ViewGroup που περιλαμβάνει.

Τα αντικείμενα Προβολής (View Objects) συνήθως ονομάζονται "γραφικά στοιχεία" και μπορεί να είναι μία από τις πολλές υποκλάσεις, όπως ένα κουμπί (Button) ή ένα πεδίο κειμένου (TextView). Τα αντικείμενα ViewGroup ονομάζονται συνήθως "διατάξεις" και μπορεί να είναι ένας από τους πολλούς τύπους που παρέχουν διαφορετική δομή διάταξης, όπως μια γραμμική διάταξη (LinearLayout) .

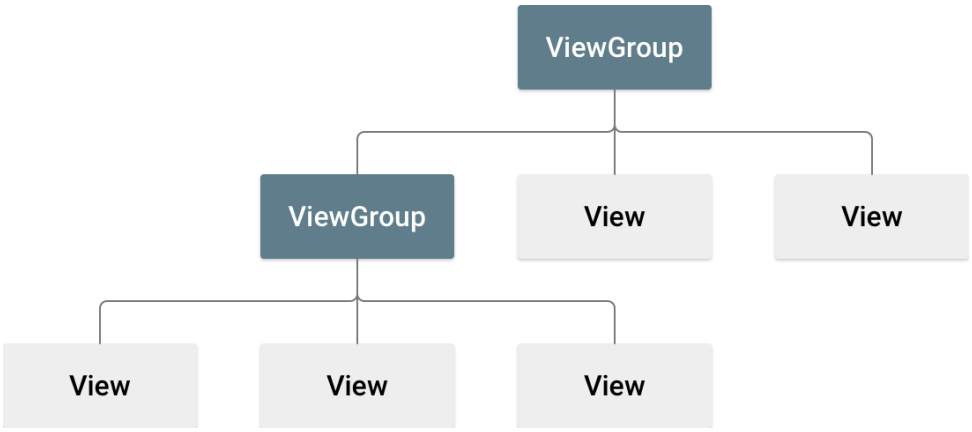
Ο προγραμματιστής έχει την δυνατότητα να δηλώσει ένα σχέδιο (Layout) με δυο τρόπους:

- Δήλωση στοιχείων διεπαφής χρήστη (UI Elements) σε αρχείο επεκτάσιμης γλώσσα σήμανσης (Extensible Markup Language - XML). Το Android παρέχει ένα απλό λεξιλόγιο XML που αντιστοιχεί στις τάξεις και τις υποκλάσεις της Προβολής (View Class), όπως αυτές που είναι για γραφικά στοιχεία και διατάξεις.
- Δημιουργίας στιγμιαίων στοιχείων διάταξης (Instantiate layout elements) κατά το χρόνο εκτέλεσης. Η εφαρμογή σας μπορεί να δημιουργήσει αντικείμενα View και ViewGroup (και να χειριστεί τις ιδιότητές τους) μέσω προγραμματισμού.

Η δήλωση της διεπαφής χρήστη σε XML επιτρέπει στον προγραμματιστή να διαχωρίσει την παρουσίαση της εφαρμογής σας από τον κώδικα που ελέγχει τη συμπεριφορά

της. Η χρήση αρχείων XML διευκολύνει επίσης την παροχή διαφορετικών διατάξεων για διαφορετικά μεγέθη και προσανατολισμούς οθόνης όπως για παράδειγμα η παροχή διαφορετικών Layouts σε κινητά ή tablets.

Το πλαίσιο Android δίνει την ευελιξία στον προγραμματιστή να χρησιμοποιήσει μία ή και τις δύο μεθόδους για να δημιουργήσει διεπαφή χρήστη για την εφαρμογής του. Για παράδειγμα, μπορεί να δηλωθούν οι προεπιλεγμένες διατάξεις της εφαρμογής σε XML αρχείο και, στη συνέχεια, να τροποποιηθούν οι διατάξεις προγραμματιστικά κατά το χρόνο εκτέλεσης.




Εικόνα 2.12.α Απεικόνιση μιας ιεραρχίας προβολής

## 2.12 Επιλογή και δημιουργία Layout

Η δημιουργία ενός Σχεδίου (Layout) γίνεται με την κατάλληλη επιλογή του σχετικού κανόνα διάταξης που επιθυμούμε να ακολουθήσουν όλες οι προβολές και οι ομάδες προβολών που θα περιλαμβάνονται στο layout προς δημιουργία. Ο σχετικός κανόνας αυτός ή View Group συνήθως τίθεται ως πατέρας - αρχικός κόμβος ή κοντέινερ.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

   <!-- More GUI components go here  -->

</LinearLayout>
```

Εικόνα 2.12.α Δημιουργία Layout από XML αρχείο που ακολουθεί την γραμμική διάταξη.

Υπάρχουν συνολικά 10 είδη Layout ή ViewGroups, όπου το κάθε ένα παρέχει έναν διαφορετικό κανόνα διάταξης στα γραφικά στοιχεία που περιέχει. Για την κατάλληλη επιλογή ενος Layout λαμβάνεται πάντα υπόψιν ο σχετικός κανόνας διάταξης που αντιπροσωπεύει καλυτέρα την διεπαφή χρήστη (User Interface - UI) που θέλουμε να δημιουργήσουμε. Κάποιες φορές μπορεί να χρειαστεί να γίνει ένας συνδυασμός των layout, φυσικά κάτω από ένα Layout - γονέα. Παράδειγμα μπορεί ένα Layout να ακολουθεί την Σχετική Διάταξη (Relative Layout) και να έχει 2 παιδία - ViewGroups που να παρέχουν στα δικά τους παιδιά την γραμμική διάταξη (Linear Layout). Όταν δημιουργούμε μια νέα δραστηριότητα στο Android Studio, αυτομάτως δημιουργείται και το Layout που συνοδεύει την δραστηριότητα αυτή. Από προεπιλογή η διάταξη που ακολουθεί το Layout αυτό είναι η Διάταξη Περιορισμών (Constraint Layout). Τα 10 είδη Layout είναι:

- Linear Layout
- Relative Layout
- Constraint Layout
- Table Layout
- Frame Layout
- List View
- Grid View
- Absolute Layout

- WebView
- ScrollView

Εκ των οποίων τα πιο συνηθισμένα Layout που χρησιμοποιουνται είναι τα

- Linear Layout
- Relative Layout
- Frame Layout
- Grid View

### 2.12.1 Linear Layout

Η γραμμική διάταξη είναι μια ομάδα προβολής που ευθυγραμμίζει όλα τα παιδιά σε μια ενιαία κατεύθυνση, κάθετα ή οριζόντια. Η κατεύθυνση της διάταξης μπορεί να οριστεί με το χαρακτηριστικό android:orientation στο XML αρχείο. Όλα τα παιδιά ενός LinearLayout στοιβάζονται το ένα μετά το άλλο, επομένως μια κατακόρυφη λίστα θα έχει μόνο ένα παιδί ανά σειρά, ανεξάρτητα από το πόσο πλατιά είναι, και μια οριζόντια λίστα θα έχει ύψος μόνο μία σειρά (το ύψος του πιο ψηλού παιδιού, συν υλικό παραγεμίσματος). Το LinearLayout σέβεται τα περιθώρια μεταξύ των παιδιών και τη βαρύτητα (δεξιά, κεντρική ή αριστερή στοίχιση) κάθε παιδιού.

### 2.12.2 Relative Layout

Η σχετική διάταξη (relative layout) είναι μια ομάδα προβολών που εμφανίζει τις θυγατρικές προβολές σε σχετικές θέσεις. Η θέση κάθε προβολής μπορεί να καθοριστεί ως προς τα αδερφικά στοιχεία (όπως στα αριστερά, δεξιά, πάνω ή κάτω από μια άλλη προβολή) ή σε θέσεις σε σχέση με τη γονική περιοχή RelativeLayout (όπως στοιχισμένη προς τα κάτω, αριστερά ή στο κέντρο). Η σχετική διάταξη είναι ένα πολύ ισχυρό εργαλείο για το σχεδιασμό μιας διεπαφής χρήστη (UI), επειδή μπορεί να εξαλείψει τις ένθετες ομάδες προβολής και να διατηρήσει επίπεδη την ιεραρχία της διάταξης, γεγονός που βελτιώνει την απόδοση.

### 2.12.3 Frame Layout

Αυτή η διάταξη έχει σχεδιαστεί για να αποκλείει μια περιοχή στην οθόνη με σκοπό την εμφάνιση ενός μεμονωμένου στοιχείου. Γενικά, το FrameLayout θα πρέπει να χρησιμοποιείται για τη διατήρηση μιας μεμονωμένης θυγατρικής προβολής, επειδή μπορεί να είναι δύσκολη η οργάνωση των θυγατρικών προβολών με τρόπο που να μπορεί να κλιμακωθεί σε διαφορετικά μεγέθη οθόνης χωρίς τα πεδία να επικαλύπτονται μεταξύ τους. Υπάρχει, ωστόσο, η δυνατότητα προσθήκης πολλών πεδίων σε ένα FrameLayout ώστε να ελέγχεται η θέση τους μέσα στο FrameLayout εκχωρώντας βαρύτητα σε κάθε παιδί, χρησιμοποιώντας το χαρακτηριστικό android:layout_gravity στο XML αρχείο.

### 2.12.4 Grid View

Η Προβολή πλέγματος (Grid View) αποτελεί μια διάταξη που τοποθετεί τα παιδιά της σε ένα ορθογώνιο (αόρατο) πλέγμα. Το πλέγμα αποτελείται από ένα σύνολο απείρως λεπτών γραμμών που χωρίζουν την περιοχή προβολής σε κελιά. Σε όλο το API, οι γραμμές πλέγματος αναφέρονται με δείκτες πλέγματος. Ένα πλέγμα με N στήλες έχει N + 1 δείκτες πλέγματος που εκτείνονται από το 0 έως το N συμπεριλαμβανομένων. Ανεξάρτητα από τον τρόπο διαμόρφωσης του GridLayout, ο δείκτης πλέγματος 0 είναι στερεωμένος στην μπροστινή άκρη του κοντέινερ και ο δείκτης πλέγματος N είναι στερεωμένος στο πίσω άκρο του (αφού ληφθεί υπόψη η επένδυση).

## 2.13 Επιλογή Κατάλληλων Γραφικών Στοιχείων

Το λειτουργικό σύστημα Android παρέχει μια μεγάλη ποικιλία από γραφικά στοιχεία ή προβολές (Views), όπου η κάθε μια προβολή εξυπηρετεί ένα ή πολλούς σκοπούς (ή λειτουργίες) ανάλογα με τις ανάγκες που έχουν καθοριστεί από το σχεδιασμό της εκάστοτε εφαρμογής. Για παράδειγμα το πάτημα ενός κουμπιού μπορεί να εξυπηρετεί μια ή πολλές λειτουργίες ταυτόχρονα όπως η αναπαραγωγή ενός τραγουδιού ή σε συνδυασμό με την αλλαγή του κυρίου φόντου της εφαρμογής με το καλλιτέχνη του τραγουδιού. Τα πιο συνηθισμένα Views είναι:

- TextView

- EditText
- Button
- Image Button
- Date Picker
- RadioButton
- CheckBox buttons
- Image View

Μια προβολή (view) έχει τα δικά της χαρακτηριστικά όπως ύψος, πλάτος, ταυτότητα (id) καθώς και αλλά πολλά. Κάθε προβολή κληρονομεί από την κλάση View κάποιες μεθόδους οι οποίες επιτρέπουν στον προγραμματιστή να δώσει διαφορετική συμπεριφορά όπως πχ στο πάτημα (on Click) ή στο άγγιγμα (on Touch) της εν λόγω προβολής.

## 2.14 Android Manifest

Κάθε έργο εφαρμογής πρέπει να έχει ένα αρχείο AndroidManifest.xml (με αυτό ακριβώς το όνομα) στη ρίζα του συνόλου προέλευσης του έργου. Το αρχείο δήλωσης περιγράφει βασικές πληροφορίες σχετικά την εφαρμογή στα εργαλεία κατασκευής Android, στο λειτουργικό σύστημα Android και στο Google Play.

Μεταξύ πολλών άλλων πραγμάτων, το αρχείο δήλωσης απαιτείται να δηλώνει τα εξής:

- Το όνομα πακέτου της εφαρμογής, το οποίο συνήθως ταιριάζει με τον χώρο ονομάτων του κώδικά της εφαρμογής. Τα εργαλεία κατασκευής Android το χρησιμοποιούν για να προσδιορίσουν τη θέση των οντοτήτων κώδικα κατά την κατασκευή του έργου της εφαρμογής. Κατά τη διαδικασία packaging της εφαρμογής, τα εργαλεία δημιουργίας αντικαθιστούν αυτήν την τιμή με το αναγνωριστικό εφαρμογής από τα αρχεία δημιουργίας Gradle, το οποίο χρησιμοποιείται ως το μοναδικό αναγνωριστικό εφαρμογής στο σύστημα και στο Google Play.
- Όλα τα συστατικά στοιχεία της εφαρμογής, τα οποία περιλαμβάνουν όλες τις δραστηριότητες, τις υπηρεσίες, τους δέκτες εκπομπής και τους παρόχους περιεχομένου. Κάθε στοιχείο πρέπει να ορίζει βασικές ιδιότητες, όπως το όνομα της κλάσης Kotlin ή Java. Μπορεί επίσης να δηλώσει δυνατότητες όπως ποιες

διαμορφώσεις συσκευής μπορεί να χειριστεί και φίλτρα πρόθεσης (Intent Filters) που περιγράφουν πώς μπορεί να ξεκινήσει το κάθε συστατικό στοιχείο.

- Τα δικαιώματα που χρειάζεται η εφαρμογή για πρόσβαση σε προστατευμένα μέρη του συστήματος ή άλλων εφαρμογών. Το αρχείο δήλωσης δηλώνει επίσης τυχόν δικαιώματα που πρέπει να έχουν άλλες εφαρμογές εάν θέλουν να έχουν πρόσβαση σε περιεχόμενο από την τρέχον εφαρμογή.
- Τα χαρακτηριστικά υλικού και λογισμικού που απαιτεί η εφαρμογή, τα οποία επηρεάζουν τις συσκευές που μπορούν να εγκαταστήσουν την εφαρμογή από το Google Play.

Το Android Studio δημιουργεί αυτόματα το αρχείο δήλωσης κάθε φορά που δημιουργείται ένα καινούργιο Android έργο (Android Project) μαζί με τα περισσότερα από τα βασικά στοιχεία που χρειάζεται το αρχείο δήλωσης.

# Κεφάλαιο 3 - Ανάλυση Εννοιών και μεθόδων χρήσιμα για την υλοποίηση του έργου.

## 3.1 Εισαγωγή

Η κυριά ιδέα ήταν η ανάπτυξη ενός παιχνιδιού λέξεων για κινητές συσκευές όπου ένας χρήστης θα μπορεί να παίξει αντίπαλος με έναν άλλον χρήστη. Δοθέντος λοιπόν τυχαίων ίδιων γραμμάτων και στους δυο χρήστες, οι χρήστες θα ανταγωνίζονται στην δημιουργία λέξεων μέσα σε ένα χρονικό όριο, ώστε να συγκεντρώσουν πόντους από κάθε λέξη που βρίσκουν και δημιουργούν. Οι πόντοι μιας λέξης θα προκύπτουν από την συνάθροιση των πόντων που θα έχει κάθε γράμμα της λέξης αυτής, όπως και στο πολύ γνωστό σε όλους μας παιχνίδι Scrabble. Κάθε παιχνίδι θα έχει 3 γύρους, όπου ο κάθε γύρος θα έχει ένα χρονικό όριο που θα ορίζεται από τους χρήστες. Ο αριθμός των λέξεων θα ορίζεται και αυτός από τους χρήστες και θα κυμαίνεται από 8 έως 13 γράμματα. Ο κάθε χρήστης θα πρέπει να διαθέτει έναν λογαριασμό, ώστε να αποθηκεύεται η πρόοδος του και διάφορα αλλά δεδομένα παιχνιδιού όπως η λίστα φίλων του, να μπορεί να βρίσκει άλλους παίχτες και να τους προσθέτει στην λίστα φίλων του, να μπορεί να προσκαλέσει φίλούς του σε παιχνίδι, να μπορεί να παίξει με τυχαίους αντίπαλους, να μπορεί να ανέβει στην συνολική κατάταξη των παιχτών, στους κορυφαίους 20, καθώς και άλλες δυνατότητες. Οι ιδέες αυτές γίνονται απαιτήσεις του παιχνιδιού και συνεπώς γεννιούνται

αρκετά ερωτήματα και προβληματισμοί υλοποιήσης που μπορούν να εκφραστούν με την μορφή ερωτημάτων, μεταξύ αυτών:

- Για ποιο λειτουργικό σύστημα κινητών συσκευών θα αναπτυχθεί το παιχνίδι αυτό;
- Ποια βάση δεδομένων θα χρησιμοποιήσουμε, δεδομένου ότι όλες οι αλλαγές και ενημερώσεις θα πρέπει να έχουν απήχηση σε πραγματικό χρόνο;
- Πως οι χρήστες θα δημιουργούν λογαριασμό; Μήπως θα ήταν καλύτερη επιλογή η χρήση ενός μηχανισμού αυθεντικοποίησης μέσω κάποιου γνωστού παρόχου όπως η Google και το Facebook;
- Πόσες γλώσσες θα υποστηρίζει η εφαρμογή;
- Που θα αποθηκευτεί το λεξικό ή τα λεξικά των γλωσσών, δεδομένου ότι ένα πλήρης λεξικό συνήθως περιέχει πάνω από 350.000 λέξεις.
- Πόσο χρόνο θα χρειάζεται η αναζήτηση μιας λέξης σε ένα λεξικό εκατοντάδων λέξεων, ώστε να μπορεί να επικυρωθεί μια λέξη κατά την διάρκεια του παιχνιδιού;
- Υπάρχει κάποιος μηχανισμός ώστε να μπορεί η συσκευή να δημιουργεί λέξεις με συγκεκριμένα γράμματα;
- Με ποιον τρόπο θα μπορούν δυο χρήστες να συγχρονιστούν και να οδηγηθούν ταυτόχρονα στο ίδιο δωμάτιο παιχνιδιού;
- Πως οι χρήστες θα μπορούν να παίζουν με τυχαίους αντίπαλους και πως θα αντιμετωπιστεί η περίπτωση της μη διαθεσιμότητας ενεργών χρηστών, σε μια τυχαία χρονική στιγμή;
- Πως θα δημιουργηθεί ένας μηχανισμός εικονικού αντιπάλου (bot), με ρεαλιστική συμπεριφορά που θα προσαρμόζεται στο επιπέδου του χρήστη;
- Η εφαρμογή θα υποστηρίζει την συνομιλία (chat) μεταξύ των χρηστών; Αν ναι, πως θα δημιουργηθεί ένας μηχανισμός κρυπτογράφησης των δεδομένων αυτών, που ανταλλάζουν οι χρήστες μεταξύ τους;

## 3.2 Βάση Δεδομένων πραγματικού χρόνου

Μια βάση δεδομένων σε πραγματικό χρόνο είναι ένα σύστημα βάσης δεδομένων που χρησιμοποιεί επεξεργασία σε πραγματικό χρόνο για να χειριστεί φόρτους εργασίας των οποίων η κατάσταση αλλάζει συνεχώς. Οι βάσεις δεδομένων διαφέρουν από τις παραδοσιακές βάσεις δεδομένων που περιέχουν μόνιμα δεδομένα, ως επί το πλείστων ανεπηρέαστα από το χρόνο.

Χρησιμοποιούν χρονικούς περιορισμούς που αντιπροσωπεύουν ένα συγκεκριμένο εύρος τιμών για τις οποίες ισχύουν τα δεδομένα. Αυτό το εύρος ονομάζεται χρονική εγκυρότητα. Μια συμβατική βάση δεδομένων δεν μπορεί να λειτουργήσει υπό αυτές τις συνθήκες, επειδή οι ασυνέπειες μεταξύ των αντικειμένων του πραγματικού κόσμου και των δεδομένων που τα αντιπροσωπεύουν είναι πολύ σοβαρές για απλές τροποποιήσεις. Ένα αποτελεσματικό σύστημα πρέπει να μπορεί να χειρίζεται ερωτήματα ευαίσθητα στον χρόνο, να επιστρέφει μόνο προσωρινά έγκυρα δεδομένα και να υποστηρίζει τον προγραμματισμό προτεραιότητας. Για την εισαγωγή των δεδομένων στα αρχεία, συχνά ένας αισθητήρας ή μια συσκευή εισόδου παρακολουθεί την κατάσταση του φυσικού συστήματος και ενημερώνει τη βάση δεδομένων με νέες πληροφορίες για να αντικατοπτρίζει το φυσικό σύστημα με μεγαλύτερη ακρίβεια. Όταν σχεδιάζεται ένα σύστημα βάσης δεδομένων σε πραγματικό χρόνο, θα πρέπει να γίνει η σκέψη της αναπαράστασης του έγκυρου χρόνου αλλά και πώς συνδέονται τα γεγονότα με το σύστημα σε πραγματικό χρόνο. Επίσης, πρέπει να γίνει η σωστή αναπαράσταση των τιμών των χαρακτηριστικών στη βάση δεδομένων, έτσι ώστε οι συναλλαγές διεργασιών και η συνέπεια των δεδομένων να μην έχουν παραβιάσεις.

Κατά το σχεδιασμό ενός συστήματος, είναι σημαντικό να ληφθεί υπόψη τι πρέπει να κάνει το σύστημα όταν δεν τηρούνται οι προθεσμίες. Για παράδειγμα, ένα σύστημα ελέγχου εναέριας κυκλοφορίας παρακολουθεί συνεχώς εκατοντάδες αεροσκάφη και λαμβάνει αποφάσεις σχετικά με τις εισερχόμενες διαδρομές πτήσης και καθορίζει τη σειρά με την οποία πρέπει να προσγειωθεί το αεροσκάφος με βάση δεδομένα όπως καύσιμα, υψόμετρο και ταχύτητα. Εάν κάποια από αυτές τις πληροφορίες καθυστερήσει, το αποτέλεσμα θα μπορούσε να είναι καταστροφικό. Για την αντιμετώπιση προβλημάτων απαρχαιωμένων δεδομένων, η χρονική σήμανση μπορεί να υποστηρίξει συναλλαγές παρέχοντας σαφείς χρονικές αναφορές.

### 3.2.1 Firebase

Το Firebase είναι μια βάση δεδομένων σε πραγματικό χρόνο που φιλοξενείται στο cloud. Τα δεδομένα αποθηκεύονται ως αρχεία JSON και συγχρονίζονται σε πραγματικό χρόνο με κάθε συνδεδεμένο πελάτη (client). Όταν πρόκειται για εφαρμογές που τρεχουν σε πολλαπλές πλατφόρμες όπως οι πλατφόρμες της Apple, του Android καθώς και τα SDK JavaScript, όλοι οι πελάτες έχουν πρόσβαση και μοιράζονται το ίδιο στιγμιότυπο βάσης

δεδομένων σε πραγματικό χρόνο και λαμβάνουν αυτόματα ενημερώσεις με τα νεότερα δεδομένα.

Αντί για τυπικά αιτήματα του πρωτοκόλλου HTTP, το Firebase χρησιμοποιεί συγχρονισμό δεδομένων κάθε φορά που αλλάζουν δεδομένα και οποιαδήποτε συνδεδεμένη συσκευή λαμβάνει αυτήν την ενημέρωση αλλαγής δεδομένων μέσα σε χιλιοστά του δευτερολέπτου. Το Firebase παρέχετε συνεργατικές και καθηλωτικές εμπειρίες χωρίς να προβληματίζει τον προγραμματιστή σχετικά με τον κώδικα δικτύωσης από την πλευρά του διακομιστή (server).

Οι εφαρμογές Firebase εξακολουθούν να αποκρίνονται ακόμα και όταν βρίσκονται εκτός σύνδεσης, επειδή το SDK της βάσης δεδομένων σε πραγματικό χρόνο του Firebase διατηρεί τα δεδομένα τοπικά στον σκληρό δίσκο της εκάστοτε συσκευής. Τα συμβάντα σε πραγματικό χρόνο συνεχίζουν να ενεργοποιούνται, δίνοντας στον τελικό χρήστη μια εμπειρία απόκρισης. Όταν η συσκευή επανακτήσει τη σύνδεση με το διαδίκτυο, η βάση συγχρονίζει τις αλλαγές τοπικών δεδομένων με τις απομακρυσμένες ενημερώσεις που πραγματοποιήθηκαν ενώ ο πελάτης ήταν εκτός σύνδεσης, συγχωνεύοντας αυτόματα τυχόν διενέξεις.

Η πρόσβαση στη Βάση Δεδομένων σε πραγματικό χρόνο του Firebase μπορεί να γίνει απευθείας από μια κινητή συσκευή ή ένα πρόγραμμα περιήγησης ιστού (Browser). Με το Firebase κάθε εφαρμογή δεν χρειάζεται να έχει τον δικό της διακομιστή εφαρμογών (Application Server). Το Firebase επιτρέπει την δημιουργία συνεργατικών εφαρμογών, παρέχοντας την ασφαλή πρόσβαση στη βάση δεδομένων απευθείας από τον κώδικα από την πλευρά του πελάτη (client side).

Το Firebase παρέχει μια ευέλικτη γλώσσα κανόνων που βασίζεται σε εκφράσεις, που ονομάζεται Κανόνες ασφαλείας της βάσης. Οι κανόνες αυτοί εκτελούνται κατά την ανάγνωση ή την εγγραφή των δεδομένων στην βάση. Η ασφάλεια και η επικύρωση των δεδομένων μέσω των κανόνων ασφαλείας τίθενται από τον προγραμματιστή κατά την δημιουργία ενός στιγμιότυπου έργου βάσης (New Firebase Project) και μπορούν να ορίσουν ποιος έχει πρόσβαση σε ποια δεδομένα και πώς μπορεί να γίνεται η πρόσβαση σε αυτά.

Επιπλέον το firebase είναι μια βάση δεδομένων NoSQL και ως εκ τούτου έχει διαφορετικές βελτιστοποιήσεις και λειτουργικότητα σε σύγκριση με μια σχεσιακή βάση δεδομένων. Το API του firebase έχει σχεδιαστεί για να επιτρέπει μόνο λειτουργίες που μπορούν να εκτελεστούν γρήγορα. Αυτό δίνει τη δυνατότητα της δημιουργίας μιας εξαιρετικής εμπειρία σε πραγματικό χρόνο που μπορεί να εξυπηρετήσει εκατομμύρια χρήστες χωρίς

συμβιβασμούς στην ανταπόκριση. Εξαιτίας αυτού, είναι σημαντικό οι προγραμματιστές να σκεφτούν πώς οι χρήστες τους θα πρέπει να έχουν πρόσβαση στα δεδομένα ώστε να γίνει η κατάλληλη δόμηση αυτών.

Τέλος το firebase παρέχει την δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν οποιαδήποτε διεύθυνση URL έργου βάσης δεδομένων Firebase (Firebase Project) σε πραγματικό χρόνο ως τελικό σημείο REST αρχιτεκτονικής. Το μόνο που χρειάζεται είναι η προσθήκη της κατάληξης .json στο τέλος της διεύθυνσης URL ενός έργου βάσης firebase και να σταλθεί οποιοδήποτε αίτημα πρωτοκόλλου HTTPS.

Η Google παρουσίασε μια νέα έκδοση του Firebase στο I/O 2016, η πλατφόρμα που προσφέρει μια ολοκληρωμένη λύση για τη δημιουργία μιας υποδομής back-end για κινητά (iOS και Android) καθώς και εφαρμογές ιστού.

Από την αρχική υποδομή Mobile-Back-end-as-a-Service (MBaaS), η Google έχει μετατρέψει το Firebase σε μια ολοκληρωμένη λύση back-end τόσο για κινητά όσο και για εφαρμογές ιστού. Το Firebase διαθέτει ένα SDK και μία κονσόλα για τη δημιουργία και τη διαχείριση εφαρμογών σε Android, iOS και ιστού. Το Firebase διαθέτει τις ακόλουθες υπηρεσίες:

- **AdMob:** Ενσωμάτωση με το Google AdMob (διεπαφή για προβολή διαφημίσεων).
- **AdWords:** Ενοποίηση με το Google AdWords (υπηρεσία για διαφήμιση ηλεκτρονικών προϊόντων).
- **Analytics:** Ένας πίνακας ελέγχου για την παρακολούθηση της συμπεριφοράς των χρηστών, της δημογραφικής ανάλυσης χρήσης μιας εφαρμογής και της απόδοσης της καμπάνιας (συνδυασμός με AdWords).
- **Authentication:** Υποστήριξη για έλεγχο ταυτότητας μέσω παρόχων όπως email, Facebook, GitHub, Google Sign-In και Twitter.
- **Crash Reporting:** Παρακολούθηση σφαλμάτων εφαρμογής σε όλες τις συσκευές. Είναι ενσωματωμένο με την υπηρεσία Analytics για την αξιολόγηση της συμπεριφοράς των χρηστών σε περίπτωση σφάλματος εφαρμογής.
- **Database:** Μια βάση δεδομένων NoSQL που χρησιμοποιείται για την αποθήκευση δεδομένων JSON.
- **Dynamic links:** Σύνδεσμοι σε βάθος που οδηγούν έναν χρήστη στην επιθυμητή σελίδα μέσα σε μια εφαρμογή.
- **Host:** Μπορεί να φιλοξενήσει οποιαδήποτε εφαρμογή Ιστού.

- **Index:** Χρησιμοποιείται για την ευρετηρίαση μιας εφαρμογής για την Αναζήτηση Google.
- **Invitations:** Επιτρέπει στους χρήστες να μοιράζονται πληροφορίες σχετικά με μια εφαρμογή με άλλους χρήστες.
- **Messages:** Η πρώην υπηρεσία Google Cloud Messaging (GCM) που έχει μετονομαστεί σε Firebase Cloud Messaging (FCM)
- **Notifications:** Διαχείριση των ειδοποιήσεων που αποστέλλονται στους χρήστες.
- **Offline:** Επιτρέπει στις εφαρμογές να αποθηκεύουν δεδομένα σε μια τοπική προσωρινή μνήμη, ώστε να μπορούν να συνεχίσουν να εκτελούνται ενώ βρίσκονται εκτός σύνδεσης.
- **Realtime:** Τα δεδομένα αποθηκεύονται στη βάση δεδομένων cloud σε πραγματικό χρόνο.
- **Remote Config:** Επιτρέπει στους προγραμματιστές να τροποποιούν τη συμπεριφορά ή την εμφάνιση μιας εφαρμογής χωρίς να απαιτείται από τους χρήστες να κάνουν λήψη μιας νέας έκδοσης της εφαρμογής. Αυτή η δυνατότητα χρησιμοποιείται για την αλλαγή του οπτικού θέματος της εφαρμογής, την αντιμετώπιση διαφορετικών δημογραφικών στοιχείων χρηστών, την εκτέλεση δοκιμών A/B κ.λπ.
- **Save:** Αποθήκευση ήχου, εικόνων και βίντεο χρήστη.
- **Synchronization:** Όταν τα δεδομένα αλλάζουν σε μια συσκευή, προωθούνται στο Firebase και μετά πίσω σε όλες τις σχετικές συσκευές. Επίσης, μια συσκευή ενημερώνεται αυτόματα με το πιο πρόσφατο στιγμιότυπο όταν έρχεται στο διαδίκτυο αφού είναι εκτός σύνδεσης.
- **Test Lab:** Δοκιμή μιας εφαρμογής σε πραγματικές συσκευές.

Όλες οι παραπάνω υπηρεσίες είναι διαθέσιμες για ανάπτυξη Android και iOS, εκτός από το Test Lab που δεν υποστηρίζεται για συσκευές iOS. Ορισμένες από τις λειτουργίες δεν υποστηρίζονται ακόμη για εφαρμογές web.

Το Firebase SDK υποστηρίζει προγραμματισμό σε C++, Java, JavaScript, JavaScript/Node.js, Objective-C και Swift. Τα Framework όπως Angular, Backbone, Ember και React υποστηρίζονται μέσω δεσμεύσεων στη βάση δεδομένων. Η Google πρόσθεσε μια σειρά από βοηθητικές βιβλιοθήκες: FirebaseUI, Geofire, Firebase Queue, FirebaseJobDispatcher. Το όνομά τους δείχνει ποιος είναι ο σκοπός τους. Το Firebase

υποστηρίζει επίσης την εισαγωγή μεγάλων συνόλων δεδομένων JSON και την ενσωμάτωση με το ElasticSearch.

### 3.2.2 Διαχείριση Δεδομένων στο Firebase

Όλα τα δεδομένα της βάσης δεδομένων σε πραγματικό χρόνο του Firebase αποθηκεύονται ως αντικείμενα JSON. Μπορούμε να παρομοιάσουμε τη βάση δεδομένων αυτή ως ένα δέντρο JSON που φιλοξενείται στο cloud. Σε αντίθεση με μια βάση δεδομένων SQL, δεν υπάρχουν πίνακες ή εγγραφές. Όταν προστεθούν δεδομένα στο δέντρο JSON, δημιουργείται ένας νέος κόμβος στην υπάρχουσα δομή του δέντρου JSON με ένα συσχετισμένο κλειδί ως ζευγάρι, γνωστό και ως κλειδί - τιμή (Key-Value). Οι προγραμματιστές μπορούν να παρέχουν δικά τους κλειδιά για κάθε δεδομένο ή σύνολο δεδομένων, όπως αναγνωριστικά χρήστη (User ID) ή σημασιολογικά ονόματα, ή μπορούν να παρέχονται από την ίδια την βάση όταν προσθέτουμε νέα δεδομένα στην βάση. Ένα κλειδί δεδομένων πρέπει να είναι κωδικοποιημένο σε μορφή κειμένου UTF-8 και το μέγεθος του δεν πρέπει να ξεπερνάει τα 768 byte. Επιπλέον ένα κλειδί δεν μπορεί να περιέχει χαρακτήρες ελέγχου ***.***, ***$***, ***#***, ***[***, ***]***, ***/*** ή ASCII 0-31 ή 127 και δεν μπορούν να χρησιμοποιηθούν χαρακτήρες ελέγχου ASCII στις ίδιες τις τιμές των δεδομένων.

```
{
  "users": {
    "alanisawesome": {
      "date_of_birth": "June 23, 1912",
      "full_name": "Alan Turing",
      "nickname": "Alan The Machine"
    },
    "gracehop": {
      "date_of_birth": "December 9, 1906",
      "full_name": "Grace Hopper",
      "nickname": "Amazing Grace"
    }
  }
}
```

Εικόνα 3.2.2.α Μορφή αποθηκευμένων δεδομένων με μοναδικό αναγνωριστικό δεδομένων ένα σημασιολογικό όνομα.

Το firebase υποστηρίζει την αποθήκευση εμφωλευμένων δεδομένων σε βάθος των 32 επίπεδων. Ωστόσο η διατήρηση των δεδομένων σε μεγάλο βάθος επίπεδων δεν θεωρείται η καλύτερη πρακτική και συνίσταται στους προγραμματιστές η αποθήκευση των δεδομένων σε όσο τον δυνατόν λιγότερα επίπεδα. Ο λόγος είναι γιατί όταν γίνεται η ανάκτηση ενός μονοπατιού (Data Path) της βάσης δεδομένων γίνεται ανάκτηση όλων των θυγατρικών κόμβων του.

### 3.2.3 Εγγραφή και Ανάγνωση Δεδομένων στο Firebase για Android

Για να γίνει ανάγνωση και εγγραφή δεδομένων από τη βάση δεδομένων, είναι απαραίτητο ένα reference (μια αναφορά) της Singleton τύπου κλάση DatabaseReference η οποία κληρονομεί τα χαρακτηριστικά της κλάσης Query.

Για τις βασικές λειτουργίες εγγραφής στην βάση δεδομένων του Firebase, χρησιμοποιούμε την μέθοδο setValue() της κλάσης DatabaseReference. Η μέθοδος αυτή αποθηκεύει τα δεδομένα σε μια καθορισμένη διαδρομή, αντικαθιστώντας τυχόν υπάρχοντα δεδομένα σε αυτήν τη διαδρομή. Η μέθοδος αυτή μπορεί να αποθηκεύσει στην βάση όλους τους διαθέσιμους τύπους δεδομένων που υποστηρίζονται από το αρχείο JSON. Επίσης υπάρχει η δυνατότητα να αποθηκεύσουμε ένα προσαρμοσμένο αντικείμενο Java, εάν φυσικά η κλάση του αντικειμένου που το ορίζει έχει έναν προεπιλεγμένο κατασκευαστή (Default Constructor) που δεν δέχεται ορίσματα και έχει ορίσει όλα τα πεδία της (fields/attributes) δημόσιας πρόσβασης (public). Τα περιεχόμενα του αντικειμένου αυτού αντιστοιχίζονται αυτόματα ως κλειδί τιμή (Key Value), οπού το κλειδί είναι το όνομα του πεδίου και η τιμή είναι η τιμή που έχει οριστεί σε αυτό το πεδίο. Το αντικείμενο αυτό αποθηκεύεται στην δηλωμένη διαδρομή ως παιδιά της. Οι τύποι αυτοί είναι :

- String
- Long
- Double
- Boolean
- Map<String, Object>
- List<Object>

```
private DatabaseReference mDatabase;
// ...
mDatabase = FirebaseDatabase.getInstance().getReference();
// ...
// ...
public void writeNewUser(String userId, String name, String email) {
    User user = new User(name, email);

    mDatabase.child("users").child(userId).setValue(user);
}
```

Εικόνα 3.2.3.α Αρχικοποίηση μιας αναφοράς της κλάση DatabaseReference και αποθήκευση ενός προσαρμοσμένου αντικειμένου Java.

Για την ανάγνωση δεδομένων από την βάση δεδομένων Firebase μιας συγκεκριμένης διαδρομής χρησιμοποιούμε μια μέθοδο ακροατή (Listener) η οποία πυροδοτείτε όταν συμβαίνουν συγκεκριμένες αλλαγές στα δεδομένα της διαδρομής αυτή. Ανάλογα με τις αλλαγές των δεδομένων που θέλουμε να έχουμε ενημέρωση, όπως προσθήκη νέων δεδομένων, διαγραφή δεδομένων ή ενημέρωση υπάρχοντών δεδομένων επιλέγουμε τους κατάλληλους Listeners. Ένας Listener όταν κληθεί την πρώτη φορά από ένα συγκεκριμένο σημείο του κώδικα της εφαρμογής, μένει ενεργός και υπακούει στα αντίστοιχα συμβάντα αλλαγών των δεδομένων. Συνεπώς η σωστή χρήση και διαχείριση των Listeners είναι ευθύνη των προγραμματιστών. Οι Listeners δουλεύουν ασύγχρονα, δηλαδή δουλεύουν σε ξεχωριστό νήμα από το κύριο νήμα της εφαρμογής και έτσι δεν παγώνει η ροή του κυρίου νήματος όσο χρονικό διαστήματα χρειάζονται για να διαβάσουν τα δεδομένα και να εκτελέσουν τις εργασίες που τους έχουν αναθέσει οι προγραμματιστές της εφαρμογής. Οι Listeners προστίθενται ή αφαιρούνται στο reference της κλάσης DatabaseReference. Οι πιο συνηθισμένοι μέθοδοι της κλάσης DatabaseReference για την διαχείριση της βάσης όπως των προσθηκών νέων δεδομένων, ενημερώσεων ή αλλαγών είναι οι εξής:

- child(String pathString)
- getDatabase()
- goOffline()
- goOnline()

- push()
- removeValue()
- setValue(Object value)
- updateChildren(Map<String, Object> update)

### 3.2.3.α Μεθοδος child(String pathString)

Η μέθοδος child() επιστρέφει έναν τύπο DatabaseReference σε μια διαδρομή τοποθεσίας. Ουσιαστικά είναι υπεύθυνη για τον καθορισμό ενός directory σε μια διαδρομή. Το όνομα του directory δίνεται ως δεδομένο String μέσω της μοναδικής της παραμέτρου. Η διαδρομή αυτή είτε πρόκειται να δημιουργηθεί είτε να προσπελασθεί.

### 3.2.3.β Μεθοδος getDatabase()

Η μέθοδος getDatabase() επιστρέφει ένα instance της βάσης δεδομένων που σχετίζεται με αυτήν την αναφορά και η οποία είναι μοναδική και καθολική σε όλο το εύρος του project.

### 3.2.3.γ Μεθοδος goOffline()

Η κλήση της μεθόδου goOffline() επιτρέπει την χειροκίνητη αποσύνδεση του πελάτη (client) από τον διακομιστή της βάσης δεδομένων Firebase και απενεργοποιεί την αυτόματη επανασύνδεση. Η κλήση αυτής της μεθόδου επηρεάζει όλες τις συνδέσεις της βάσης δεδομένων Firebase που υπάρχουν σε μια εφαρμογή.

### 3.2.3.δ Μεθοδος goOnline()

Η κλήση της μεθόδου goOffline() επιτρέπει την χειροκίνητη σύνδεση του πελάτη (client) με τον διακομιστή της βάσης δεδομένων Firebase και ταυτόχρονα ενεργοποιεί την αυτόματη επανασύνδεση. Όπως και στην κλήση της μεθόδου goOffline, που περιεγράφηκε προηγουμένως, η κλήση αυτής της μεθόδου επηρεάζει όλες τις συνδέσεις της βάσης δεδομένων Firebase που υπάρχουν σε μια εφαρμογή.

### 3.2.3.γ Μεθοδος Push()

Συνενώνει σε μια νέα διαδρομή δεδομένων που πρόκειται να δημιουργηθεί, ένα θυγατρικό κλειδί - χρονικής διάταξης (directory). Ουσιαστικά με αυτήν την μέθοδο, εισάγονται και αποθηκεύονται νέα δεδομένα με την διάταξη που έχει ορισθεί από το θυγατρικό κλειδί. Το θυγατρικό κλειδί δημιουργείται από την πλευρά του πελάτη και ενσωματώνει μια εκτίμηση του χρόνου του διακομιστή για σκοπούς ταξινόμησης. Η εκτίμηση αυτή είναι μια αλληλουχία αλφαριθμητικών χαρακτήρων και έχει την εξής μορφή: *-NLlSFSqmG2D5PHniU3q*. Οι τοποθεσίες που δημιουργούνται σε έναν μεμονωμένο πελάτη θα ταξινομηθούν με τη σειρά που δημιουργούνται και θα ταξινομηθούν κατά προσέγγιση βάση του εκάστοτε θυγατρικού κλειδιού, και θα είναι ορατές με την ίδια διάταξη σε όλους τους πελάτες που συνδέονται στην βάση.

### 3.2.3.ε Μέθοδος removeValue()

Η μέθοδος αυτή, θέτει το περιεχόμενο μιας τοποθεσίας της βάσης στην τιμή null και ουσιαστικά διαγράφει ολόκληρη την διαδρομή δεδομένων (το path) καθώς και τα δεδομένα που περιέχει. Ουσιαστικά αυτή η μέθοδος διαγράφει δεδομένά από την βάση δεδομένων, βάση του επιθυμητού directory path των δεδομένων.

### 3.2.3.στ Μέθοδος setValue(Object value)

Αποθηκεύει τα δεδομένα στην βάση. Αν προηγουμένως έχει ορισθεί συγκεκριμένη διαδρομή με την χρήση της μεθόδου child, τότε αποθηκεύει τα δεδομένα ως παιδιά της διαδρομής αυτής. Η μεταβίβαση της τιμής null ως παράμετρο στη μέθοδο αυτή θα διαγράψει τα δεδομένα στην καθορισμένη τοποθεσία (directory path) αν φυσικά υπάρχουν δεδομένα σε αυτήν.

### 3.2.3.ζ Μέθοδος updateChildren(Map<String, Object> update)

Η μέθοδος αυτή ενημερώνει τα δεδομένα στις καθορισμένες τιμές τα οποία βρίσκονται στην καθορισμένη διαδρομή (θυγατρικά κλειδιά). Η μέθοδος αυτή μπορεί να δεχτεί την τιμή null η οποία όμως έχει ως αποτέλεσμα την κατάργηση της τιμής στην καθορισμένη τοποθεσία (διαγραφή δεδομένων τοποθεσίας).

## 3.2.4 Διεπαφές και Μέθοδοί Προσθήκης γεγονότων Ακροατών (addEventListeners)

Το firebase παρέχει δυο βασικές διεπαφές (interfaces) που η κάθε μια προσανατολίζεται διαφορετικά ως προς τον χειρισμό των πυροδοτήσεων των αλλαγών που σημειώνονται σε ένα στιγμιότυπο βάσης δεδομένων firebase. Έτσι ένα στιγμιότυπο της κλάσης DatabaseReference παρέχει μεθόδους προσθήκης event listeners δηλαδή γεγονότων ακροατών (addEventListeners) που η κάθε μια μέθοδο δέχεται ως παράμετρο έναν από τους δυο τύπους διεπαφών, αφού η ίδια έχει κληρονομήσει τις μεθόδους αυτές από την κλάση Query.

Οι δυο διεπαφές αυτές είναι:

- ValueEventListener
- ChildEventListener

### 3.2.4.α Διεπαφή ChildEventListener

Τα θυγατρικά συμβάντα (Child Events) ενεργοποιούνται ως απόκριση σε συγκεκριμένες λειτουργίες που συμβαίνουν στα παιδιά ενός κόμβου από μια λειτουργία όπως ένα νέο παιδί που προστίθεται μέσω της μεθόδου push() ή ένα παιδί που ενημερώνεται μέσω της μεθόδου updateChildren(). Καθένα από αυτά μαζί μπορεί να είναι χρήσιμο για την ακρόαση αλλαγών σε έναν συγκεκριμένο κόμβο σε μια βάση δεδομένων. Η διεπαφή αυτή δηλώνει τέσσερεις μεθόδους, όπου η κάθε μια μέθοδο αντιστοιχεί σε διαφορετικά είδη πυροδότησης αλλαγών γεγονότων, όπως, προσθήκης, ενημέρωσης, διαγράφης και μετακίνησης παιδιών ενός κόμβου. Έτσι μπορούμε να διαχειριστούμε μεμονωμένα οποιοδήποτε είδος αλλαγών σε μια συγκεκριμένη διαδρομή δεδομένων ενός στιγμιότυπου βάσης, όπως ακριβώς αναφέρθηκε.

Οι μέθοδοι που παρέχονται στην διεπαφή ChildEventListener είναι:

- **onChildAdded()** Η οποία ανακτάει λίστες αντικειμένων ή πυροδοτείτε όταν υπάρξουν νέες προσθήκες σε μια λίστα αντικειμένων σε μια καθορισμένη διαδρομή. Αυτή η μέθοδος επανάκλησης (callback method) πυροδοτείτε μία φορά για κάθε υπάρχον παιδί και στη συνέχεια κάθε φορά που προστίθεται ένα νέο παιδί στην διαδρομή αυτή. Το στιγμιότυπο δεδομένων (DataSnapshot) που παρέχεται από την παράμετρο αυτής της μεθόδου περιέχει τα δεδομένα του νέου παιδιού.
- **onChildChanged()** η οποία υπακούει για αλλαγές στα στοιχεία μιας λίστας. Αυτό το συμβάν πυροδοτείτε κάθε φορά που τροποποιείται ένας θυγατρικός κόμβος, συμπεριλαμβανομένων τυχόν τροποποιήσεων σε απογόνους του κόμβου αυτού. Το στιγμιότυπο δεδομένων που διαβιβάστηκε στην μέθοδο μέσω της παραμέτρου της περιέχει τα ενημερωμένα δεδομένα για το παιδί-κόμβος που πραγματοποιήθηκαν οι αλλαγές.
- **onChildRemoved()** η οποία μέθοδος υπακούει για αντικείμενα που αφαιρούνται από μια λίστα. Το στιγμιότυπο δεδομένων που μεταβιβάστηκε στην μέθοδο επανάκλησης αυτή μέσω της παραμέτρου της περιέχει τα δεδομένα για το παιδί που αφαιρέθηκε.
- **onChildMoved()** η οποία μέθοδος υπακούει όταν υπάρξουν αλλαγές στη σειρά των στοιχείων σε μια ταξινομημένη λίστα. Αυτό το συμβάν ενεργοποιείται κάθε φορά που ενεργοποιείται η μέθοδος επανάκλησης onChildChanged() από μια ενημέρωση που προκαλεί αναδιάταξη των στοιχείων ενός παιδιού. Χρησιμοποιείται με δεδομένα που έχουν ταξινομηθεί με την χρήση των μεθόδων orderByChild ή orderByValue.

### 3.2.4.β Διεπαφή ValueEventListener

Καθώς η χρήση ενός ChildEventListener είναι ο προτεινόμενος τρόπος ανάγνωσης λιστών δεδομένων, μιας και κατηγοριοποιεί της αλλαγές που συμβαίνουν στα δεδομένα μέσω μεθόδων, υπάρχουν περιπτώσεις όπου η επισύναψη ενός ValueEventListener σε μια αναφορά λίστας είναι χρήσιμη. Μια διαφορά με ένα ChildEventListener είναι ότι η επισύναψη ενός ValueEventListener σε μια λίστα δεδομένων θα επιστρέψει ολόκληρη τη λίστα δεδομένων ως ένα ενιαίο DataSnapshot, το οποίο δίνει την δυνατότητα στον προγραμματιστή να αποκτήσει πρόσβαση σε μεμονωμένα παιδιά μέσω δομών επαναλήψεων (loop over). Ακόμη και όταν υπάρχει μόνο μία αντιστοίχιση για το ερώτημα πάνω σε δεδομένα (query), το στιγμιότυπο

εξακολουθεί να είναι μια λίστα. Επίσης μια δεύτερη διαφορά του ValueEventListener με το ChildEventListener είναι ότι το πρώτο υπακούει σε οποιαδήποτε αλλαγή παρατηρηθεί στα δεδομένα χωρίς να μπορεί να γίνει διάκριση αυτής της αλλαγής, όπως πχ αν υπήρξε ενημέρωση κάποιου κόμβου ή διαγραφή παιδιού -κόμβου.

Οι μέθοδοι που παρέχονται στην διεπαφή ValueEventListener είναι:

- **onDataChange()** η μέθοδος αυτή παρέχει ένα στατικό στιγμιότυπο των δεδομένων σε μια συγκεκριμένη διαδρομή, όπως ακριβώς υπήρχαν τα δεδομένα τη στιγμή του συμβάντος. Αυτή η μέθοδος ενεργοποιείται μία φορά όταν ο Listener συνδέεται για πρώτη φορά και πυροδοτείτε ξανά κάθε φορά που αλλάζουν τα δεδομένα, συμπεριλαμβανομένων και των παιδιών - κόμβων. Το στιγμιότυπο δεδομένων (DataSnapshot) που παρέχεται από την παράμετρο αυτής της μεθόδου φέρνει όλα τα δεδομένα από την διαδρομή, συμπεριλαμβανομένων των θυγατρικών δεδομένων. Εάν δεν υπάρχουν δεδομένα, το στιγμιότυπο θα επιστρέψει false όταν κληθεί η μέθοδος exists() και null όταν κληθεί η getValue().
- **onCancelled()** Αυτή η μέθοδος θα ενεργοποιηθεί σε περίπτωση που υπάρξει αποτυχία σύνδεσης με τον διακομιστή είτε αφαιρεθεί ως αποτέλεσμα των κανόνων ασφάλειας και βάσης δεδομένων Firebase (όπως για παράδειγμα μη ύπαρξη αδείας του client για πρόσβαση στα δεδομένα).

### 3.2.4.γ Μεθοδοι προσθήκης γεγονότων Διεπαφων (addEventListeners)

Όπως προαναφέραμε ένα στιγμιότυπο της κλάσης DatabaseReference παρέχει μεθόδους προσθήκης γεγονότων ακροατών (event listener - addEventListeners), αφού η ίδια έχει κληρονομήσει τις μεθόδους αυτές από την κλάση Query. Οι ακροατές αυτοί (listeners) δέχονται ως παράμετρο έναν από τους δυο τύπους διεπαφών που αναλύσαμε προηγουμένως και με την υλοποίηση που δίνει ο κάθε προγραμματιστής, επιτυγχάνεται το ανάλογο αποτέλεσμα ως προς τις πυροδοτήσεις για τις αλλαγές των δεδομένων που πραγματοποιούνται στην βάση.

Οι μέθοδοι προσθήκης γεγονότων ακροατών είναι οι:

- **addChildEventListener()** προσθέτει σε ένα στιγμιότυπο βάσης, έναν listener, για συμβάντα που λαμβάνουν χώρα σε κόμβους - παιδιά σε ένα συγκεκριμενο μονοπάτι.

Όταν προστίθενται, αφαιρούνται, αλλάζουν ή μετακινούνται θυγατρικές τοποθεσίες, ο listener θα ενεργοποιείται για το αντίστοιχο συμβάν.

- **addValueEventListener()** προσθέτει σε ένα στιγμιότυπο βάσης, έναν listener που ακούει στις αλλαγές των δεδομένων που πραγματοποιούνται σε αυτήν την τοποθεσία. Κάθε φορά που αλλάζουν τα δεδομένα, ο listener θα καλείται παρέχοντας ένα αμετάβλητο (immutable) στιγμιότυπο των δεδομένων.
- **addListenerForSingleValueEvent()** προσθέτει σε ένα στιγμιότυπο βάσης, έναν listener που ακούει για μία μόνο αλλαγή στα δεδομένα σε αυτήν την τοποθεσία. Δέχεται μια παράμετρο τύπου ValueEventListener. Αυτός ο listener θα ενεργοποιηθεί μία φορά και καμία άλλη, παρέχοντας ένα αμετάβλητο (immutable) στιγμιότυπο των δεδομένων.

## 3.3 Έλεγχος Ταυτότητας - Αυθεντικοποίηση

Ο έλεγχος ταυτότητας είναι η διαδικασία προσδιορισμού του εάν κάποιος ή κάτι είναι, στην πραγματικότητα, ποιος ή τι λέει ότι είναι. Η τεχνολογία ελέγχου ταυτότητας παρέχει έλεγχο πρόσβασης για συστήματα ελέγχοντας εάν τα διαπιστευτήρια ενός χρήστη ταιριάζουν με τα διαπιστευτήρια σε μια βάση δεδομένων εξουσιοδοτημένων χρηστών ή σε έναν διακομιστή ελέγχου ταυτότητας δεδομένων. Με αυτόν τον τρόπο, ο έλεγχος ταυτότητας διασφαλίζει ασφαλή συστήματα, ασφαλείς διαδικασίες και ασφάλεια εταιρικών πληροφοριών και προσωπικών δεδομένων.

Ο παραδοσιακός έλεγχος ταυτότητας εξαρτάται από τη χρήση ενός αρχείου κωδικού πρόσβασης, στο οποίο αποθηκεύονται τα αναγνωριστικά χρήστη μαζί με τους κατακερματισμούς των κωδικών πρόσβασης που σχετίζονται με κάθε χρήστη. Κατά τη σύνδεση, ο κωδικός πρόσβασης που υποβάλλεται από τον χρήστη κατακερματίζεται και συγκρίνεται με την τιμή στο αρχείο κωδικού πρόσβασης. Εάν οι δύο κατακερματισμοί ταιριάζουν, ο χρήστης επαληθεύεται.

Αυτή η προσέγγιση στον έλεγχο ταυτότητας έχει πολλά μειονεκτήματα, ιδιαίτερα για πόρους που αναπτύσσονται σε διαφορετικά συστήματα. Πρώτον, οι εισβολείς που μπορούν να αποκτήσουν πρόσβαση στο αρχείο κωδικών πρόσβασης για ένα σύστημα μπορούν να χρησιμοποιήσουν επιθέσεις brute-force εναντίον των κατακερματισμένων κωδικών

πρόσβασης για να εξαγάγουν τους κωδικούς πρόσβασης. Επιπλέον, αυτή η μέθοδος θα απαιτούσε πολλαπλούς ελέγχους ταυτότητας για σύγχρονες εφαρμογές που έχουν πρόσβαση σε πόρους σε πολλαπλά συστήματα.

Οι αδυναμίες ελέγχου ταυτότητας βάσει κωδικών πρόσβασης μπορούν να αντιμετωπιστούν σε κάποιο βαθμό με πιο έξυπνα ονόματα χρήστη και κωδικούς πρόσβασης που βασίζονται σε κανόνες όπως το ελάχιστο μήκος και η πολυπλοκότητα χρησιμοποιώντας κεφαλαία γράμματα και σύμβολα. Ωστόσο, ο έλεγχος ταυτότητας βάσει κωδικού πρόσβασης και ο έλεγχος ταυτότητας βάσει γνώσης είναι πιο ευάλωτοι από τα συστήματα που απαιτούν πολλαπλές ανεξάρτητες μεθόδους.

Οι μέθοδοι ελέγχου ταυτότητας περιλαμβάνουν είναι οι ακόλουθοι:

- 2FA
- MFA
- OTP
- Three-Factor Authentication
- Biometrics
- Mobile Authentication
- Continuous Authentication
- Application Programming Interface (API) Authentication

### 3.3.1 2FA

Αυτός ο τύπος ελέγχου ταυτότητας προσθέτει ένα επιπλέον επίπεδο προστασίας στη διαδικασία, απαιτώντας από τους χρήστες να παρέχουν έναν δεύτερο παράγοντα ελέγχου ταυτότητας εκτός από τον κωδικό πρόσβασης. Τα συστήματα 2FA συχνά απαιτούν από τον χρήστη να εισαγάγει έναν κωδικό επαλήθευσης που έλαβε μέσω μηνύματος κειμένου σε ένα προεγγεγραμμένο κινητό τηλέφωνο ή κινητή συσκευή ή έναν κωδικό που δημιουργείται από μια εφαρμογή ελέγχου ταυτότητας.

### 3.3.2 MFA

Αυτός ο τύπος ελέγχου ταυτότητας απαιτεί από τους χρήστες να πραγματοποιούν έλεγχο ταυτότητας με περισσότερους από έναν παράγοντες ελέγχου ταυτότητας, συμπεριλαμβανομένου ενός βιομετρικού παράγοντα, όπως δακτυλικό αποτύπωμα ή αναγνώριση προσώπου. παράγοντα κατοχής, όπως κλειδί ασφαλείας. ή ένα διακριτικό που δημιουργήθηκε από μια εφαρμογή ελέγχου ταυτότητας.

### 3.3.3 OTP

Το OTP είναι μια αριθμητική ή αλφαριθμητική συμβολοσειρά χαρακτήρων που δημιουργείται αυτόματα και επαληθεύει την ταυτότητα ενός χρήστη. Αυτός ο κωδικός πρόσβασης είναι έγκυρος μόνο για μία περίοδο σύνδεσης ή συναλλαγή και συνήθως χρησιμοποιείται για νέους χρήστες ή για χρήστες που έχασαν τους κωδικούς πρόσβασής τους και τους δίνεται ένα OTP για να συνδεθούν και να αλλάξουν σε νέο κωδικό πρόσβασης.

### 3.3.4 Three-Factor Authentication

Αυτός ο τύπος MFA χρησιμοποιεί τρεις παράγοντες ελέγχου ταυτότητας -- συνήθως, έναν παράγοντα γνώσης, όπως έναν κωδικό πρόσβασης, σε συνδυασμό με έναν παράγοντα κατοχής, όπως ένα διακριτικό ασφαλείας, και έναν εγγενή παράγοντα, όπως ένα βιομετρικό στοιχείο.

### 3.3.5 Biometrics

Ενώ ορισμένα συστήματα ελέγχου ταυτότητας εξαρτώνται αποκλειστικά από τη βιομετρική ταυτοποίηση, τα βιομετρικά στοιχεία χρησιμοποιούνται συνήθως ως δεύτερος ή τρίτος παράγοντας ελέγχου ταυτότητας. Οι πιο συνηθισμένοι τύποι βιομετρικού ελέγχου ταυτότητας που είναι διαθέσιμοι περιλαμβάνουν σαρώσεις δακτυλικών αποτυπωμάτων, σαρώσεις προσώπου ή αμφιβληστροειδούς και φωνητική αναγνώριση.

### 3.3.6 Mobile Authentication

Ο έλεγχος ταυτότητας μέσω κινητού τηλεφώνου είναι η διαδικασία επαλήθευσης των χρηστών μέσω των συσκευών τους ή επαλήθευσης των ίδιων των συσκευών. Αυτό επιτρέπει στους χρήστες να συνδέονται σε ασφαλείς τοποθεσίες και πόρους από οπουδήποτε. Η διαδικασία ελέγχου ταυτότητας για κινητά περιλαμβάνει MFA που μπορεί να περιλαμβάνει OTP, βιομετρικό έλεγχο ταυτότητας ή κωδικό γρήγορης απόκρισης

### 3.3.7 Continuous Authentication

Με συνεχή έλεγχο ταυτότητας, αντί ο χρήστης να είναι συνδεδεμένος ή εκτός σύνδεσης, η εφαρμογή μιας εταιρείας υπολογίζει συνεχώς μια βαθμολογία ελέγχου ταυτότητας που μετρά πόσο σίγουρος είναι ότι ο κάτοχος του λογαριασμού είναι το άτομο που χρησιμοποιεί τη συσκευή.

### 3.3.8 Application Programming Interface (API) Authentication

Οι τυπικές μέθοδοι διαχείρισης του ελέγχου ταυτότητας API είναι ο βασικός έλεγχος ταυτότητας HTTP, τα κλειδιά API και η Ανοιχτή Εξουσιοδότηση (OAuth).

## 3.4 Αυθεντικοποιηση με την χρήση του Firebase

Οι περισσότερες εφαρμογές πρέπει να γνωρίζουν την ταυτότητα ενός χρήστη. Η γνώση της ταυτότητας ενός χρήστη επιτρέπει σε μια εφαρμογή να αποθηκεύει με ασφάλεια δεδομένα χρήστη στο cloud ή στην βάση δεδομένων της και να παρέχει την ίδια εξατομικευμένη εμπειρία σε όλες τις συσκευές του χρήστη. Όπως αναφέραμε και προηγουμένως για την υλοποίηση του παιχνιδιού λέξεων της παρούσας διπλωματικής είναι αναγκαίο να υπάρχει ένα σύστημα αυθεντικοποίησης για τους χρήστες μας.

Το Firebase Authentication παρέχει υπηρεσίες υποστήριξης, εύχρηστα SDK και έτοιμες βιβλιοθήκες διεπαφής χρήστη για τον έλεγχο ταυτότητας των χρηστών μιας εφαρμογή. Υποστηρίζει τον έλεγχο της ταυτότητας ενός χρήστη (αυθεντικοποίηση) χρησιμοποιώντας

κωδικούς πρόσβασης, αριθμούς τηλεφώνου, δημοφιλείς παρόχους ταυτότητας όπως το Google, το Facebook και το Twitter και πολλά άλλα.

Το Firebase Authentication ενσωματώνεται στενά με άλλες υπηρεσίες Firebase και αξιοποιεί τα βιομηχανικά πρότυπα όπως το OAuth 2.0 και το OpenID Connect, ώστε να μπορεί εύκολα να ενσωματωθεί με το προσαρμοσμένο backend της εκάστοτε εφαρμογής.

## 3.5 Η Αυθεντικοποίηση μέσω Firebase

Για να συνδεθεί ένας χρήστης σε μια εφαρμογή, η εφαρμογή λαμβάνει πρώτα διαπιστευτήρια ελέγχου ταυτότητας από τον χρήστη. Αυτά τα διαπιστευτήρια μπορεί να είναι η διεύθυνση email και ο κωδικός πρόσβασης του χρήστη ή ένα OAuth τεκμήριο (OAuth token) από έναν ομοσπονδιακό πάροχο ταυτότητας, όπως η Google, το Facebook ή και άλλοι γνωστοί πάροχοι . Στη συνέχεια, μεταβιβάζετε αυτά τα διαπιστευτήρια στο SDK ελέγχου ταυτότητας Firebase. Στη συνέχεια, οι υπηρεσίες υποστήριξης θα επαληθεύσουν αυτά τα διαπιστευτήρια και θα επιστρέψουν μια απάντηση στον πελάτη(client) δηλαδή την συσκευή του χρήστη.

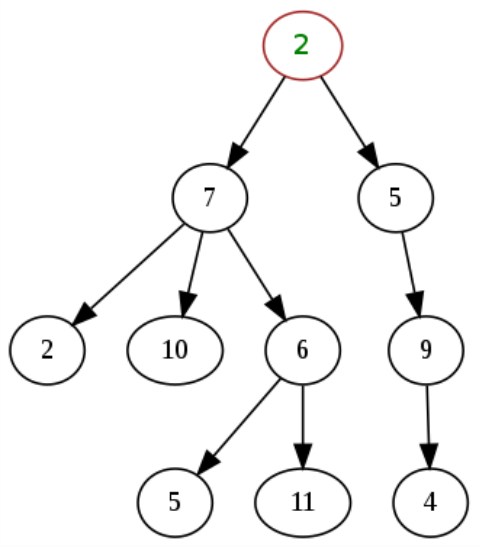
Μετά από μια επιτυχημένη σύνδεση, η εφαρμογή μπορεί να αποκτήσετε πρόσβαση στις βασικές πληροφορίες προφίλ του χρήστη και να ελέγξετε την πρόσβαση του χρήστη σε δεδομένα που είναι αποθηκευμένα σε άλλα προϊόντα Firebase. Η εφαρμογή μπορεί επίσης να χρησιμοποιήσει το παρεχόμενο τεκμήριο ελέγχου ταυτότητας (authedication token) για να επαληθεύσει την ταυτότητα των χρηστών στις δικές της υπηρεσίες (backend services).


Εικόνα 3.5.α Firebase Authedication

## 3.6 Τι ονομάζουμε δέντρο στην επιστήμη των υπολογιστών

Στην επιστήμη των υπολογιστών, ένα δέντρο είναι ένας ευρέως χρησιμοποιούμενος αφηρημένος τύπος δεδομένων που αναπαριστά μια ιεραρχική δομή δέντρου, με μια τιμή ρίζας και παιδιών ως υποδέντρα με έναν γονικό κόμβο, που αναπαρίσταται ως ένα σύνολο συνδεδεμένων κόμβων. Μια δομή δεδομένων δέντρου μπορεί να οριστεί αναδρομικά ως μια συλλογή κόμβων, όπου κάθε κόμβος είναι μια δομή δεδομένων που αποτελείται από μια τιμή και μια λίστα αναφορών σε κόμβους. Η αρχή του δέντρου ονομάζεται "κόμβος ή ρίζα" και οι κόμβοι αναφοράς είναι τα "παιδιά". Καμία αναφορά δεν είναι διπλή και καμία δεν δείχνει τη ρίζα, δηλαδή ενώ υπάρχει διαδρομή από έναν κόμβο "πατερά" προς έναν κόμβο "παιδί" το αντίθετο δεν ισχύει.

Εναλλακτικός ορισμός που συναντάμε στις βιβλιογραφίες και εκφράζει την μαθηματική οπτική, είναι πως ένα δέντρο μπορεί να οριστεί αφηρημένα ως ένα σύνολο (καθολικά) ή ως ταξινομημένο δέντρο, με μια τιμή να εκχωρείται σε κάθε κόμβο. Και οι δύο αυτές προοπτικές είναι χρήσιμες: ενώ ένα δέντρο μπορεί να αναλυθεί μαθηματικά ως σύνολο, όταν στην πραγματικότητα αναπαρίσταται ως δομή δεδομένων, συνήθως αναπαρίσταται και δουλεύεται ξεχωριστά ανά κόμβο (αντί ως σύνολο κόμβων και λίστα γειτνίασης ακμών μεταξύ κόμβων , όπως μπορεί κανείς να αντιπροσωπεύει ένα δίγραμμα, για παράδειγμα). Για παράδειγμα, κοιτάζοντας ένα δέντρο στο σύνολό του, μπορεί κανείς να μιλήσει για "τον γονικό κόμβο" ενός συγκεκριμένου κόμβου, αλλά γενικά, ως δομή δεδομένων, ένας συγκεκριμένος κόμβος περιέχει μόνο τη λίστα των παιδιών του, αλλά δεν περιέχει αναφορά στον γονέα του (εάν υπάρχει).




Εικόνα 3.6.α Ένα αταξινόμητο δέντρο.

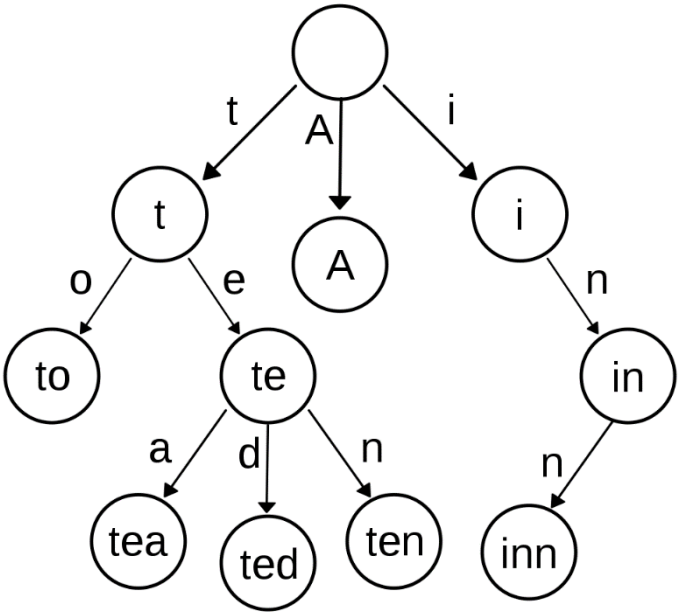
## 3.7 Δέντρο προθέματος - Prefix Tree (Trie)

Στην επιστήμη των υπολογιστών, ένα Prefix Tree (γνωστό και ως “trie”), που ονομάζεται επίσης ψηφιακό δέντρο ή δέντρο προθέματος, είναι ένας τύπος δέντρου αναζήτησης, μια δομή δεδομένων δέντρου που χρησιμοποιείται για τον εντοπισμό συγκεκριμένων κλειδιών μέσα από ένα σύνολο. Αυτά τα κλειδιά είναι συνήθως συμβολοσειρές, με συνδέσμους μεταξύ κόμβων που δεν ορίζονται από ολόκληρο το κλειδί, αλλά από μεμονωμένους χαρακτήρες. Για να αποκτήσει κάποιος πρόσβαση σε ένα κλειδί (για την ανάκτηση της τιμή του, την αλλαγή ή την αφαίρεση), το δέντρο προθέματος διασχίζεται πρώτα σε βάθος (Depth-First Search), ακολουθώντας τους συνδέσμους μεταξύ των κόμβων, οι οποίοι αντιπροσωπεύουν κάθε χαρακτήρα του κλειδιού.

Σε αντίθεση με ένα δυαδικό δέντρο αναζήτησης, οι κόμβοι του δέντρου προθέματος δεν αποθηκεύουν το συσχετισμένο κλειδί τους. Αντίθετα, η θέση ενός κόμβου στο δέντρο αυτό ορίζει το κλειδί με το οποίο συσχετίζεται. Αυτό κατανέμει την τιμή κάθε κλειδιού στη δομή δεδομένων και σημαίνει ότι κάθε κόμβος δεν έχει απαραίτητα μια σχετική τιμή.

Όλα τα παιδιά ενός κόμβου έχουν ένα κοινό πρόθεμα της συμβολοσειράς που σχετίζεται με αυτόν τον γονικό κόμβο και η ρίζα σχετίζεται με την κενή συμβολοσειρά. Αυτή η αποστολή της αποθήκευσης δεδομένων που είναι προσβάσιμα από το πρόθεμά τους μπορεί να επιτευχθεί με τρόπο βελτιστοποιημένο για τη μνήμη χρησιμοποιώντας ένα δέντρο ρίζας.

Αν και οι προσπάθειες μπορούν να πληκτρολογηθούν με συμβολοσειρές χαρακτήρων, δεν χρειάζεται να είναι. Οι ίδιοι αλγόριθμοι μπορούν να προσαρμοστούν για ταξινομημένες λίστες οποιουδήποτε υποκείμενου τύπου, π.χ. μεταθέσεις ψηφίων ή σχημάτων. Συγκεκριμένα, μια δοκιμή bitwise πληκτρολογείται στα μεμονωμένα bit που αποτελούν ένα κομμάτι δυαδικών δεδομένων σταθερού μήκους, όπως ένας ακέραιος αριθμός ή μια διεύθυνση μνήμης.




Εικόνα 3.7 Ένα δέντρο προθέματος.

## 3.8 Ασαφής Λογική (Fuzzy Logic)

Η ασαφής λογική είναι μια μορφή λογικής πολλών τιμών στην οποία η τιμή αλήθειας των μεταβλητών μπορεί να είναι οποιοσδήποτε πραγματικός αριθμός μεταξύ 0 και 1. Χρησιμοποιείται για να χειριστεί την έννοια της μερικής αλήθειας, όπου η τιμή αλήθειας μπορεί να κυμαίνεται μεταξύ εντελώς αληθής και εντελώς ψευδής. Μια πρόταση μπορεί να είναι αληθής «με κάποιο βαθμό αληθείας», και όχι απλά αληθής ή ψευδής. Η ασαφής λογική είναι μια επέκταση της κλασσικής αριστοτέλειας λογικής. Αντίθετα, στη λογική Boole, οι τιμές αλήθειας των μεταβλητών μπορεί να είναι μόνο οι ακέραιες τιμές 0 ή 1.

Ο όρος ασαφής λογική εισήχθη με την πρόταση του 1965 για τη θεωρία ασαφών συνόλων από τον επιστήμονα Lotfi Zadeh. Η ασαφής λογική είχε, ωστόσο, μελετηθεί από τη δεκαετία του 1920, ως λογική απεριόριστης αξίας ιδίως από τους Łukasiewicz και Tarski.

Η ασαφής λογική βασίζεται στην παρατήρηση ότι οι άνθρωποι λαμβάνουν αποφάσεις με βάση ανακριβείς και μη αριθμητικές πληροφορίες. Τα ασαφή μοντέλα ή σύνολα είναι μαθηματικά μέσα για την αναπαράσταση της ασάφειας και των ανακριβών πληροφοριών (εξ ου και ο όρος ασαφής). Αυτά τα μοντέλα έχουν την ικανότητα αναγνώρισης, αναπαράστασης, χειρισμού, ερμηνείας και χρήσης δεδομένων και πληροφοριών που είναι ασαφή και στερούνται βεβαιότητας. Η ασαφής λογική έχει εφαρμοστεί σε πολλά πεδία, από τη θεωρία ελέγχου έως την τεχνητή νοημοσύνη.

Τα ασαφή συστήματα δύναται να λειτουργούν σε περιβάλλον ασάφειας και αβεβαιότητας και αποδίδουν αποτελέσματα με νόημα για τον άνθρωπο. Προσεγγίζουν δηλαδή την ανθρώπινη λογική. Είναι ένα κοινώς εξιδανικευμένο εργαλείο για την λήψη αποφάσεων. Χαρακτηριστικό προτέρημα της ασαφούς λογικής είναι ότι έχει την ικανότητα να λειτουργεί, αλλά και αναλύει, συστήματα μεγάλης πολυπλοκότητας.

Η Ασαφής λογική περιλαμβάνει:

- Ασαφής Προτάσεις
- Ασαφής Σύνολα
- Γλωσσικές Μεταβλητές
- Ασαφείς Αριθμοί
- Ασαφείς Κανόνες
- Τελεστές Ασαφούς Λογικής
- Πράξεις με Ασαφή Σύνολα

## 3.8.1 Ασαφείς Προτάσεις

Ασαφής πρόταση είναι αυτή που θέτει μια τιμή σε μια γλωσσική μεταβλητή. Για παράδειγμα στην ασαφή πρόταση το BOT θα σχηματίζει «ΠΟΛΥ ΛΙΓΕΣ» λέξεις, η λέξη «σχηματίζει» είναι γλωσσική μεταβλητή και η φράση «ΠΟΛΥ ΛΙΓΕΣ» είναι η ασαφής τιμή της. Ο συνδυασμός μιας γλωσσικής μεταβλητής με την ασαφή τιμή της (ασαφής πρόταση) ορίζουν ένα ασαφές σύνολο.

### 3.8.2 Ασαφείς Σύνολα

Ένα σύνολο διατεταγμένων ζευγών $(x, u_A(x))$ όπου $x \in X$ και $u_A(x) \in [0,1]$ ονομάζεται ασαφές σύνολο A (fuzzy set). Το σύνολο X πρόκειται για ένα εκτεταμένο πεδίο ορισμού που περιέχει κάθε αντικείμενο που μπορεί να αναφερθεί. Η τιμή $u_A(x)$ ονομάζεται βαθμός συμμετοχής, συμβολίζει το βαθμό της συμμετοχής του x στο σύνολο A και λαμβάνει τιμές στο κλειστό διάστημα [0,1]. Η συνάρτηση $u_A$ λέγεται συνάρτηση συμμετοχής και πρακτικά μπορεί να προέρχεται από:

- Φυσικές μετρήσεις
- Υποκειμενικές εκτιμήσεις
- Διαδικασίες μάθησης και προσαρμογής
- Συχνότητες εμφανίσεων και πιθανότητες

### 3.8.3 Γλωσσικές Μεταβλητές

Μια μεταβλητή της οποίας οι τιμές ορίζονται με ασαφή σύνολα ονομάζεται γλωσσική - ασαφής μεταβλητή (fuzzy variable). Γενικότερα, οι τιμές μιας ασαφούς μεταβλητής μπορεί να είναι προτάσεις σε κάποια προδιαγεγραμμένη γλώσσα με συνδυασμό ασαφών μεταβλητών, λεκτικών περιγραμμάτων και υπεκφυγών. Οι τιμές της μεταβλητής ΠΑΙΖΕΙ του παραδείγματος μπορούν έτσι να εκφραστούν ως:

«ΛΙΓΟΤΕΡΟ ΣΥΧΝΑ», «ΚΑΝΟΝΙΚΑ», «ΣΥΧΝΑ», «ΑΡΚΕΤΑ ΣΥΧΝΑ» δηλαδή με προτάσεις αποτελούμενες από την ετικέτα ΣΥΧΝΑ, την άρνηση όχι, τα συνδετικά και άλλα καθώς και τα περιγράμματα ΛΙΓΟΤΕΡΟ, ΑΡΚΕΤΑ κ.ά. Κατά τον τρόπο αυτό η μεταβλητή ΠΑΙΖΕΙ είναι μια λεκτική μεταβλητή.

### 3.8.4 Ασαφείς Αριθμοί

Ένας ασαφής αριθμός είναι μια γενίκευση ενός κανονικού, πραγματικού αριθμού με την έννοια ότι δεν αναφέρεται σε μια μεμονωμένη τιμή αλλά μάλλον σε ένα συνδεδεμένο σύνολο πιθανών τιμών, όπου κάθε δυνατή τιμή έχει το δικό της βάρος μεταξύ 0 και 1. Αυτό το

βάρος ονομάζεται συνάρτηση μέλους. Ένας ασαφής αριθμός είναι επομένως μια ειδική περίπτωση ενός κυρτού, κανονικοποιημένου ασαφούς συνόλου της πραγματικής γραμμής. Ακριβώς όπως η ασαφής λογική είναι μια επέκταση της λογικής Boole (η οποία χρησιμοποιεί μόνο την απόλυτη αλήθεια και το ψέμα, και τίποτα ενδιάμεσο), οι ασαφείς αριθμοί είναι μια επέκταση των πραγματικών αριθμών.

### 3.8.5 Ασαφείς Κανόνες

Μία υπό συνθήκη έκφραση που συσχετίζει δύο ή περισσότερες ασαφείς προτάσεις ονομάζεται ασαφής κανόνας (fuzzy rule). Στην πιο απλή εκδοχή «if x is R then y is T» Παραδείγματος χάριν «Εάν ο χρήστης συγκέντρωσε ΠΟΛΛΟΥΣ πόντους τότε το BOT θα παίζει ΑΡΚΕΤΑ ΣΥΧΝΑ» τα «συγκέντρωσε» και «παίζει» είναι οι ασαφείς μεταβλητές ενώ τα «ΠΟΛΛΟΥΣ» και «ΑΡΚΕΤΑ ΣΥΧΝΑ» είναι οι τιμές των ασαφών μεταβλητών «συγκέντρωσε» και «παίζει» αντίστοιχα.

### 3.8.6 Τελεστές Ασαφούς Λογικής

Οι τελεστές δύο στοιχείων α και β που συναντάμε στην ασαφή λογική είναι οι min και max ορίζονται ως:

- $\alpha \wedge \beta = \min(\alpha,\beta) = \alpha$ Αν $\alpha \le \beta$
- $\alpha \wedge \beta = \min(\alpha,\beta) = \beta$ Αν $\alpha > \beta$
- $\alpha \vee \beta = \max(\alpha,\beta) = \alpha$ Αν $\alpha \ge \beta$
- $\alpha \vee \beta = \max(\alpha,\beta) = \beta$ Αν $\alpha < \beta$

Οι λογικές πύλες Ή (OR) και ΚΑΙ (AND) αποτελούν παραδείγματα των παραπάνω τελεστών.

### 3.8.7 Πράξεις με Ασαφή σύνολα

Οι πράξεις που μπορούν να γίνουν μεταξύ δύο ασαφών συνόλων Α και Β είναι οι εξής:

- **Το υποσύνολο (subset)** $\mu_{B \subseteq A}(x) = \mu_B(x) \le \mu_A(x) \quad \forall x \in X$
- **Η ένωση (union)** και ορίζεται ως: $\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) \quad \forall x \in X$

- **Η τομή (intersection)** και ορίζεται ως: $\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) \quad \forall\ x \in X$
- **Το γινόμενο (product)** και ορίζεται ως: $\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x) \quad \forall\ x \in X$

Επιπλέον ένα ασαφές σύνολο A του X θεωρείται κενό (null) αν η συνάρτηση συμμετοχής του είναι μηδενική παντού, δηλαδή:

$$A = \emptyset \quad \text{Αν } \mu_A(x) = 0 \ \forall\ x \in X.$$

Τέλος το συμπλήρωμα (complement) \ overline {A} ενός ασαφούς συνόλου ορίζεται ως:

$$\mu_{\overline{A}} = 1 - \mu_A(x) \ \forall\ x \in X$$

## 3.8.8 Αρχιτεκτονική Ασαφούς Λογικής

Η αρχιτεκτονική ενός συστήματος ασαφούς λογικής περιλαμβάνει 4 συστατικά.

- Βάση Κανόνων
- Ασαφοποίηση
- Συνάθροιση Εξόδων
- Αποασαφοποίηση

## 3.8.9 Βάση Κανόνων

Περιλαμβάνει το σύνολο των κανόνων και τις προϋποθέσεις AN-TOTE που παρέχονται από τους ειδικούς για να διέπουν το σύστημα λήψης αποφάσεων, βάσει γλωσσικών πληροφοριών. Οι πρόσφατες εξελίξεις στην ασαφή θεωρία προσφέρουν αρκετές αποτελεσματικές μεθόδους για το σχεδιασμό και τον συντονισμό των ασαφών ελεγκτών. Οι περισσότερες από αυτές τις εξελίξεις μειώνουν τον αριθμό των ασαφών κανόνων.

## 3.8.10 Ασαφοποίηση

Χρησιμοποιείται για τη μετατροπή εισόδων, δηλαδή ευκρινών αριθμών σε ασαφή σύνολα. Οι καθαρές είσοδοι είναι βασικά οι ακριβείς είσοδοι που μετρώνται από αισθητήρες και περνούν στο σύστημα ελέγχου για επεξεργασία, όπως θερμοκρασία, πίεση, στροφές κ.λπ.

### 3.8.12 Συνάθροιση Εξόδων

Συνδυασμός των παραχθέντων συναρτήσεων συμμέτοχης των εμπλεκομένων κανόνων σ ένα ασαφές σύνολο. Επιπλέον καθορίζει τον βαθμό αντιστοίχισης της τρέχουσας ασαφούς εισόδου σε σχέση με κάθε κανόνα και αποφασίζει ποιοι κανόνες θα ενεργοποιηθούν σύμφωνα με το πεδίο εισαγωγής. Στη συνέχεια, οι κανόνες πυροδότησης συνδυάζονται για να σχηματίσουν τις ενέργειες ελέγχου.

### 3.8.13 Αποασαφοποίηση

Η αποασαφοποίηση είναι η διαδικασία παραγωγής ενός μετρήσιμου αποτελέσματος σε καθαρή λογική, δεδομένων ασαφών συνόλων και αντίστοιχων βαθμών συμμετοχής. Είναι η διαδικασία που αντιστοιχίζει ένα ασαφές σύνολο σε ένα ευκρινές σύνολο. Συνήθως απαιτείται σε συστήματα ασαφούς ελέγχου. Αυτά τα συστήματα θα έχουν έναν αριθμό κανόνων που μετατρέπουν έναν αριθμό μεταβλητών σε ένα ασαφές αποτέλεσμα, δηλαδή το αποτέλεσμα περιγράφεται με όρους συμμετοχής σε ασαφή σύνολα. Για παράδειγμα, κανόνες που έχουν σχεδιαστεί για να αποφασίσουν πόση πίεση θα ασκηθεί μπορεί να οδηγήσουν σε "Μείωση πίεσης (15%), Διατήρηση πίεσης (34%), Αύξηση πίεσης (72%). Η αποασαφοποίηση είναι η ερμηνεία των βαθμών συμμετοχής των ασαφών συνόλων σε μια συγκεκριμένη απόφαση ή πραγματική τιμή. Η διαδικασία της αποσαφοποίησης και γενικότερα ένας αποασαφοποιητής ορίζεται ως μια απεικόνιση του ασαφούς συνόλου B΄, το οποίο είναι η έξοδος της μηχανής ασαφούς συμπεράσματος, σε ένα σαφές σημείο □□∗ ∈ □□. Αυτή η λειτουργία είναι όμοια με τη μέση τιμή μιας τυχαίας μεταβλητής. Τρία βασικά κριτήρια υπάρχουν για την κατάλληλη επιλογή της μεθόδου αποασαφοποίησης. Τα κριτήρια αυτά είναι:

1. η υπολογιστική απλότητα που είναι ένας σημαντικός παράγοντας για ασαφείς ελεγκτές που δουλεύουν σε πραγματικό χρόνο,
2. σε μια μικρή αλλαγή του συνόλου B΄ η μέθοδος αποασαφοποίησης δεν πρέπει να επιφέρει μεγάλη αλλαγή στην τιμή του y* και
3. η τελική τιμή y* να είναι περίπου στο κέντρο του συνόλου στήριξης του B΄ ή να έχει έναν υψηλό βαθμό συμμετοχής στο σύνολο B΄.

Οι κυριότερες μεθοδολογίες αποασαφοποίησης είναι οι εξής:

- Μέθοδος κέντρου βάρους (Center of Gravity ή Centroid, COG ή COA)
- Μέθοδος των υψών ή σταθμισμένος μέσος όρος (Height method ή weighted average method)
- Μέθοδος Αποασαφοποίησης μεγίστου (Maximum defuzzifier)
- Μέθοδος Αποασαφοποίησης με μέσο όρο των μεγίστων (Mean of Maxima ή MOM defuzzifier)

Μέθοδος κέντρου βάρους Center of Gravity (CoG):

Στη μέθοδο αυτή η τιμή y* δίνεται από την σχέση:

$$y* = \frac{\int_v y\mu_{B'}(y)dy}{\int_v \mu_{B'}(y)dy}$$

ή εάν το σύνολο B΄ είναι διακριτό από τη σχέση:

$$y* = \frac{\sum_i y_i \mu_{B'}(y)dy}{\sum_i \mu_{B'}(y)dy}$$

Το μειονέκτημα αυτής της μεθόδου είναι το υπολογιστικό φορτίο που υπάρχει όταν το σύνολο B΄ είναι ακανόνιστο. Η δεύτερη μέθοδος αποασαφοποίησης έχει απλούστερο τύπο και υπολογιστική απλότητα σε σχέση με την CoG.

Μέθοδος των υψών ή σταθμισμένος μέσος όρος (Height method ή weighted average method):

Το ασαφές σύνολο B΄ είναι η ένωση ή η τομή των M ασαφών συνόλων που είναι οι έξοδοι των ενεργοποιημένων κανόνων. Μια καλή προσέγγιση της προηγούμενης μεθόδου και υπολογιστική απλότητα είναι ο σταθμισμένος μέσος όρος των κέντρων βαρύτητας $\overline{y}^l$ των M ασαφών συνόλων. Οι συντελεστές βαρύτητας είναι τα ύψη των αντίστοιχων ασαφών συνόλων

$$W^l = \mu_{B^l}(\overline{y}^l)$$

Χωρίς να μας ενδιαφέρει εάν χρησιμοποιούμε «και» (min) ή αλγεβρικό γινόμενο ως ασαφή συνεπαγωγή το κέντρο βαρύτητας $\overline{y}^l$ των ασαφών συνόλων $B^l$ για συμμετρικά σχήματα συναρτήσεων συμμετοχής είναι: η τιμή που αντιστοιχεί στην κορυφή του ισοσκελούς ή ισόπλευρου τριγώνου, η κεντρική τιμή της Γκαουσιανής συνάρτησης και το μέσον του

συνόλου στήριξης ενός συμμετρικού τραπεζίου. Στη μέθοδο αυτή η τιμή $y_h^*$ δίνεται από την σχέση:

$$y_h^* = \frac{\sum_{i=1}^{M} \overline{y}^l \, w^l}{\sum_{i=1}^{M} w^l}$$

Η μέθοδος των υψών χρησιμοποιείται ευρύτατα σε ασαφή συστήματα και στον ασαφή έλεγχο.

Μέθοδος Αποασαφοποίησης μεγίστου (Maximum defuzzifier):

Η μέθοδος αυτή επιλέγει το y* για την οποία το □□(□) έχει την μεγαλύτερη τιμή του. Αυτή η τεχνική είναι υπολογιστικά πολύ απλή, αλλά τις περισσότερες φορές δε δίνει ικανοποιητικά αποτελέσματα. Ειδικά, όταν υπάρχουν πολλαπλά μέγιστα, η επιλογή της γίνεται τυχαία μεταξύ των μεγίστων. Υπάρχουν τρεις παραλλαγές της μεθόδου που επιλύουν αυτό το πρόβλημα,

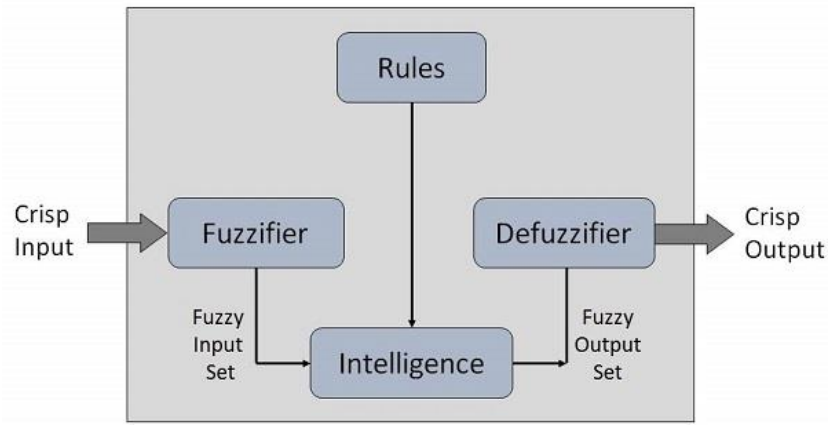
Αποασαφοποίηση του μικρότερου από τα μέγιστα (Smallest of axima ή SOM defuzzifier). Στην τεχνική αυτή, από όλα τα μέγιστα, επιλέγεται το μικρότερο και η έξοδος του συστήματος □*$_{som}$ είναι:

$$y_{SOM}^* = \min\left(\overline{y}^1, \overline{y}^2, \ldots, \overline{y}^m\right)$$

Όπου m ο αριθμός όλων των μεγίστων.

Αποασαφοποίηση του μεγαλύτερου από τα μέγιστα (Largest of Maxima ή LOM defuzzifier), όπου η έξοδος του συστήματος □*$_{LOM}$ είναι:

$$y_{LOM}^* = \max\left(\overline{y}^1, \overline{y}^2, \ldots, \overline{y}^m\right)$$



Εικόνα 3.8.13 Η διαδικασία της ασαφούς λογικής.

## 3.9 Διαδικτυακά παιχνίδια και συνομιλία

Η έλευση των διαδικτυακών παιχνιδιών άλλαξε τον τρόπο που παίζονται τα βιντεοπαιχνίδια. Από μια μεμονωμένη δραστηριότητα όπου συμμετείχαν ένας ή δύο παίκτες, μέχρι πολυμερή διαδικτυακά παιχνίδια που πλημυρίζουν από αλληλεπίδραση, οικοδόμηση κοινότητας, συντροφικότητα παικτών και πολλά άλλα.

Όλα αυτά είναι δυνατά με τη συνομιλία εντός παιχνιδιού, η οποία έχει μεταμορφώσει την εμπειρία του βιντεοπαιχνιδιού. Για τους παίκτες, η συνομιλία στο παιχνίδι κάνει το παιχνίδι πιο ευχάριστο και συναρπαστικό. Για τους προγραμματιστές παιχνιδιών, η συνομιλία παιχνιδιών βελτιώνει την απόκτηση και τη διατήρηση των χρηστών, δημιουργώντας υψηλότερες αποδόσεις.

Στα παιχνίδια για πολλούς παίκτες, η συνομιλία εντός του παιχνιδιού επιτρέπει στους παίκτες να επικοινωνούν με άλλους παίκτες, σε πραγματικό χρόνο, κάνοντας τις αλληλεπιδράσεις απρόσκοπτες. Ο τρόπος επικοινωνίας μπορεί να είναι κείμενο, φωνή ή βίντεο. Οι προγραμματιστές και οι σχεδιαστές παιχνιδιών μπορούν επίσης να παρέχουν πλούσιες σε χαρακτηριστικά συνομιλίες στο παιχνίδι για τους παίκτες τους, για παράδειγμα, διευκολύνοντας τις ομαδικές συνομιλίες, τις χειρονομίες (emoji) αντίδρασης ή τα ηχητικά εφέ για να κάνουν το παιχνίδι ακόμα πιο καθηλωτικό.

Οι προγραμματιστές παιχνιδιών μπορούν είτε να αναπτύξουν δικά τους παιχνίδια συνομιλίας είτε να ενσωματώσουν απρόσκοπτα ένα API συνομιλίας παιχνιδιών ή SDK χωρίς προβλήματα στις πλατφόρμες τους. Ενώ η συνομιλία εντός παιχνιδιού γίνεται όλο και πιο δημοφιλής, υπάρχει το ζήτημα των εφαρμογών συνομιλίας τρίτων που προσφέρουν εναλλακτικά κανάλια επικοινωνίας.

### 3.9.1 Η συνομιλία εντός παιχνιδιού και οι συνομιλίες τρίτων.

Η συνομιλία στο παιχνίδι ξεκίνησε με μια ιστορία δυσκίνητης και χωρίς χαρακτηριστικά. Κατά τη διάρκεια αυτής της περιόδου, πολλοί παίκτες στράφηκαν σε

εφαρμογές συνομιλίας τρίτων, όπως το Discord, το TeamSpeak, το Mumble και το Twitch, οι οποίες έγιναν όλο και πιο δημοφιλείς με τις ποικίλες προσφορές λειτουργιών τους.

Αυτές οι εφαρμογές παρέχουν επικοινωνία με αρκετά προνόμια μεταξύ των παικτών σε όλες τις κονσόλες. Ενώ οι περισσότερες παρέχουν βασικές λειτουργίες συνομιλίας, όπως φωνή και κείμενο, μερικά προσφέρουν μοναδικές δυνατότητες όπως κοινή χρήση αρχείων, ιδιωτικές συνομιλίες και δημιουργία κοινότητας. Ένα πλεονέκτημα των εφαρμογών συνομιλίας τρίτων είναι ότι παρέχουν συνέχεια επικοινωνίας εκτός παιχνιδιού, κάτι που απολαμβάνουν πολλοί παίκτες. Στους παίκτες αρέσει να συμμετέχουν σε μια συζήτηση εκτός παιχνιδιού για να μοιραστούν κόλπα και συμβουλές, να μάθουν για το παιχνίδι, να δημιουργούν ομάδες και κοινότητες, καθώς και πολλά άλλα.

Ωστόσο, η ταλαιπωρία της μετάβασης σε άλλες εφαρμογές εκτός του παιχνιδιού είναι επίσης προβληματική για πολλούς παίκτες. Η συνομιλία μέσα σε ένα παιχνίδι προσφέρει αξιοσημείωτη χρήση και ευκολία. Επιπλέον, οι εφαρμογές τρίτων συνήθως προσφέρουν μόνο τις βασικές λειτουργίες δωρεάν, χρεώνοντας ένα κόστος συνδρομής για πιο προηγμένες λειτουργίες, τις οποίες πολλοί νεότεροι παίκτες δεν μπορούν να αντέξουν οικονομικά. Πιάνοντας τη δημοτικότητα της συνομιλίας παράλληλα με το όρο «Gaming», οι προγραμματιστές επενδύουν τώρα περισσότερο χρόνο και πόρους για τη δημιουργία πιο συναρπαστικών και εξελιγμένων λειτουργιών συνομιλίας εντός του παιχνιδιού, αποτρέποντας τους χρήστες να απευθυνθούν σε εφαρμογές συνομιλίας τρίτων.

### 3.9.2 Τύποι συνομιλίας εντός παιχνιδιού.

Κάθε παιχνίδι είναι διαφορετικό και συνεπώς η συνομιλία μεταξύ χρηστών γίνεται διαφορετικά. Υπάρχουν διάφοροι τύποι συνομιλίας εντός παιχνιδιού.

- Συνομιλία βάσει κειμένου.
- Συνομιλία με φωνή και βίντεο.

<u>Συνομιλία βάσει κειμένου:</u>

Η συνομιλία βάσει κειμένου είναι η πιο βασική δυνατότητα ανταλλαγής μηνυμάτων σε οποιοδήποτε παιχνίδι. Περιλαμβάνει λιγότερη απαίτηση εύρους ζώνης και μπορεί να έχει μια

ελκυστική διεπαφή. Μια λύση συνομιλίας που βασίζεται σε κείμενο θα μπορούσε να έχει επιλογή κοινής χρήσης αρχείων, λειτουργίες γρήγορης συνομιλίας με καθορισμένες φράσεις, σύστημα ping για αντίδραση μέσω μη λεκτικών χειρονομιών, όπως είναι τα emoticons και πολλά άλλα. Στην παρούσα διπλωματική και για τις ανάγκες του παιχνιδιού που υλοποιήθηκε, δημιουργήθηκε μηχανισμός για την υποστήριξη συνομιλίας μεταξύ των χρηστών της εφαρμογής, η οποία είναι συνομιλία βάσει κειμένου χωρίς όμως περεταίρω δυνατότητες όπως κοινής χρήσης αρχείων ή μη λεκτικών χειρονομιών (emoticons)

Συνομιλία με φωνή και βίντεο:

Τα παιχνίδια φωνητικής συνομιλίας είναι πιο δημοφιλή μεταξύ των παικτών. Η αλληλεπίδραση σε πραγματικό χρόνο με άλλους παίκτες χωρίς διακοπή του παιχνιδιού για πληκτρολόγηση μηνύματος, καθώς η πληκτρολόγηση αποσπά την προσοχή του παίκτη από το παιχνίδι κάνει τη φωνητική συνομιλία την καλύτερη λύση συνομιλίας στο παιχνίδι. Η φωνητική συνομιλία, αφήνοντας τα χέρια ελεύθερα για τον χειρισμό μιας κονσόλας παιχνιδιών, παρέχει την πιο καθηλωτική εμπειρία καθώς οι παίκτες παίζουν με τους φίλους τους. Το στοιχείο βίντεο της συνομιλίας εντός παιχνιδιού είναι διάσημο μεταξύ των περιστασιακών ειδών παιχνιδιών που απαιτούν πολλούς παίκτες.

### 3.9.3 Η δημοτικότητα της συνομιλίας εντός παιχνιδιού.

Με τη βοήθεια μιας συνομιλίας εντός του παιχνιδιού, τα διαδικτυακά παιχνίδια έχουν γίνει μια κοινωνική δραστηριότητα όπου οι παίκτες συνεργάζονται και ενθαρρύνουν τους άλλους να εργαστούν για έναν κοινό στόχο. Οι παίκτες μπορούν να συντονιστούν με τα μέλη της ομάδας τους ή με άλλους, να προκαλέσουν ο ένας τον άλλον και να απολαύσουν ευχάριστες στιγμές κατά τη διάρκεια του παιχνιδιού. Τα φόρουμ συζήτησης για παιχνίδια είναι γεμάτα με ιστορίες για το πώς οι άνθρωποι βρήκαν καλούς φίλους ή και το άλλο τους μισό, μέσω συνομιλιών παιχνιδιών.

Η επικοινωνία φωνής/βίντεο σε πραγματικό χρόνο σε ένα παιχνίδι για πολλούς παίκτες έχει αυξήσει την αίσθηση της κοινωνικής παρουσίας κάποιου. Από απομονωμένες ζωές, οι άνθρωποι συνδέονται πλέον και αλληλοεπιδρούν με άλλους μέσω διαδικτυακών παιχνιδιών, μετατρέποντας έτσι τα διαδικτυακά παιχνίδια σε κοινωνικά περιβάλλοντα.

Το φαινόμενο εντάθηκε κατά τη διάρκεια των ημερών καραντίνας της πανδημίας του 2020, καθώς περισσότεροι άνθρωποι σε όλο τον κόσμο άρχισαν να αφιερώνουν περισσότερο χρόνο σε βιντεοπαιχνίδια ως πηγή ψυχαγωγίας και σύνδεσης με άλλους. Σε μια πρόσφατη παγκόσμια έρευνα, οι ερωτηθέντες ανέφεραν ότι είχαν αυξήσει το χρόνο τους παίζοντας βιντεοπαιχνίδια κατά 39%. Οι άνθρωποι, σωματικά απομονωμένοι και βαριούνται τη μονοτονία της παρέας σε εσωτερικούς χώρους, συνδέονται με παιχνίδια συνομιλίας όπως το Ludo King, τα διαδικτυακά καζίνο και το Facebook Gaming. Επίσης, τα παιχνίδια για πολλούς παίκτες όπως το Call of Duty, το PUBG και το Free Fire έχουν αυξήσει τη δημοτικότητα των συνομιλιών εντός του παιχνιδιού.

### 3.9.4 Η αναγκαιότητα των συνομιλιών βιντεοπαιχνιδιών.

Οι δημιουργοί διαδικτυακών παιχνιδιών, και γενικότερα οι προγραμματιστές παιχνιδιών, δεν μπορούν να αγνοήσουν και να χάσουν τις ευκαιρίες που υπάρχουν στις συνομιλίες εντός των παιχνιδιών. Το κόστος απόκτησης πελατών για ένα παιχνίδι για κινητά αυξάνεται κάθε χρόνο. Σύμφωνα με έρευνα, οι εφαρμογές παιχνιδιών ξόδεψαν 14,5 δισεκατομμύρια δολάρια το 2021 για την απόκτηση χρηστών. Το υψηλό κόστος απαιτεί την επαναστρατηγική του μάρκετινγκ παιχνιδιών.

Η συνομιλία εντός παιχνιδιού έχει κάνει τα διαδικτυακά παιχνίδια συναρπαστικά και ελκυστικά με όλη την παιχνιδιάρικη συνομιλία και την ανταλλαγή πληροφοριών. Έχει γίνει ένα ουσιαστικό χαρακτηριστικό που πρέπει να ενσωματώσουν οι προγραμματιστές παιχνιδιών εάν επιθυμούν να αποκτήσουν νέους χρήστες. Η συνομιλία εντός παιχνιδιού συμβάλλει στη μείωση του κόστους απόκτησης πελατών για εταιρείες παραγωγής τυχερών παιχνιδιών.

Ένα άλλο πλεονέκτημα για τους προγραμματιστές είναι ότι τα παιχνίδια φωνητικής συνομιλίας βοηθούν στη διατήρηση των χρηστών επειδή κρατούν τους παίκτες αφοσιωμένους. Όσο περισσότερο χρόνο αφιερώνει ένας χρήστης παίζοντας, τόσο περισσότερα χρήματα μπορεί να κερδίσει μια εταιρεία τυχερών παιχνιδιών από αυτόν τον παίκτη. Με τα παιχνίδια φωνητικής συνομιλίας, ο χρόνος περιόδου σύνδεσης ανά χρήστη αυξάνεται, αυξάνοντας έτσι τη μακροπρόθεσμη αξία του χρήστη (LTV) - μια βασική μέτρηση για μια εταιρεία τυχερών παιχνιδιών.

Υπάρχουν και άλλες ευκαιρίες που προσφέρουν τα παιχνίδια συνομιλίας. Οι συνομιλίες βιντεοπαιχνιδιών δημιουργούν πληθώρα δεδομένων, παρέχοντας ευκαιρίες στους

προγραμματιστές παιχνιδιών. Με εφαρμογές τρίτων, τα δεδομένα αποθηκεύονται σε διακομιστές στους οποίους δεν μπορούν να έχουν πρόσβαση οι προγραμματιστές. Ωστόσο, μια συνομιλία εντός του παιχνιδιού θα επιτρέψει στους προγραμματιστές να έχουν πρόσβαση στα δεδομένα και να τα χρησιμοποιήσουν για να ελέγξουν την εμπειρία του χρήστη, να δημιουργήσουν νέες δυνατότητες και να κάνουν το παιχνίδι πιο ελκυστικό.

## 3.10 Τι είναι η κρυπτογράφηση;

Η κρυπτογράφηση είναι ένας τρόπος παραμόρφωσης δεδομένων, έτσι ώστε μόνο εξουσιοδοτημένα μέρη να μπορούν να κατανοήσουν τις πληροφορίες. Σε τεχνικούς όρους, είναι η διαδικασία μετατροπής απλού κειμένου αναγνώσιμου από τον άνθρωπο σε ακατανόητο κείμενο, γνωστό και ως κρυπτογραφημένο κείμενο. Με απλούστερους όρους, η κρυπτογράφηση παίρνει αναγνώσιμα δεδομένα και τα μεταβάλλει έτσι ώστε να εμφανίζονται τυχαία. Η κρυπτογράφηση απαιτεί τη χρήση ενός κρυπτογραφικού κλειδιού: ένα σύνολο μαθηματικών τιμών στις οποίες συμφωνούν να χρησιμοποιήσουν τόσο ο αποστολέας όσο και ο παραλήπτης ενός κρυπτογραφημένου μηνύματος.

Αν και τα κρυπτογραφημένα δεδομένα εμφανίζονται τυχαία, η κρυπτογράφηση προχωρά με λογικό, προβλέψιμο τρόπο, επιτρέποντας σε ένα μέρος που λαμβάνει τα κρυπτογραφημένα δεδομένα και διαθέτει το σωστό κλειδί να αποκρυπτογραφήσει τα δεδομένα, μετατρέποντάς τα ξανά σε απλό κείμενο. Η πραγματικά ασφαλής κρυπτογράφηση θα χρησιμοποιεί κλειδιά αρκετά περίπλοκα ώστε σε ένα τρίτο μέρος να είναι πολύ απίθανο να τα αποκρυπτογραφήσει ή να σπάσει το κρυπτογραφημένο κείμενο με ωμή βία, χωρίς να έχει δικαίωμα πρόσβασης με άλλα λόγια, μαντεύοντας το κλειδί. Τα δεδομένα μπορούν να κρυπτογραφηθούν “σε ηρεμία”, όταν αποθηκεύονται ή “σε μεταφορά”, ενώ μεταδίδονται κάπου αλλού. Στην παρούσα διπλωματική γίνεται κρυπτογράφηση των δεδομένων που πρόκειται να σταλθούν κατά την συνομιλία χρηστών πριν την αποστολή τους στην βάση δεδομένων.


Εικόνα 3.10: Διαδικασία κρυπτογράφησης.

### 3.10.1 Τι είναι το κλειδί στην κρυπτογραφία;

Ένα κρυπτογραφικό κλειδί είναι μια συμβολοσειρά χαρακτήρων που χρησιμοποιούνται σε έναν αλγόριθμο κρυπτογράφησης για την αλλαγή δεδομένων έτσι ώστε να εμφανίζονται τυχαία. Όπως ένα φυσικό κλειδί, κλειδώνει (κρυπτογραφεί) δεδομένα έτσι ώστε μόνο κάποιος με το σωστό κλειδί μπορεί να τα ξεκλειδώσει (αποκρυπτογραφήσει).

### 3.10.2 Ποιοι είναι οι διαφορετικοί τύποι κρυπτογράφησης;

Τα δύο κύρια είδη κρυπτογράφησης είναι η συμμετρική κρυπτογράφηση και η ασύμμετρη κρυπτογράφηση. Η ασύμμετρη κρυπτογράφηση είναι επίσης γνωστή ως κρυπτογράφηση δημόσιου κλειδιού.

Στη συμμετρική κρυπτογράφηση, υπάρχει μόνο ένα κλειδί και όλα τα μέρη που επικοινωνούν χρησιμοποιούν το ίδιο (μυστικό) κλειδί τόσο για κρυπτογράφηση όσο και για αποκρυπτογράφηση. Στην κρυπτογράφηση ασύμμετρου ή δημόσιου κλειδιού, υπάρχουν δύο κλειδιά: ένα κλειδί χρησιμοποιείται για κρυπτογράφηση και ένα διαφορετικό κλειδί χρησιμοποιείται για αποκρυπτογράφηση. Το κλειδί αποκρυπτογράφησης διατηρείται ιδιωτικό (εξ ου και το όνομα "ιδιωτικού κλειδιού"), ενώ το κλειδί κρυπτογράφησης μοιράζεται δημόσια, για να το χρησιμοποιήσει ο καθένας (εξ ου και το όνομα "δημόσιο κλειδί"). Η ασύμμετρη κρυπτογράφηση είναι μια βασική τεχνολογία για το TLS (συχνά ονομάζεται SSL). Στην παρούσα διπλωματική, για την κρυπτογράφηση των δεδομένων που στέλνονται κατά την συνομιλία των χρηστών γίνεται χρήσης συμμετρικής κρυπτογράφησης.

### 3.10.3 Γιατί είναι απαραίτητη η κρυπτογράφηση δεδομένων;

Η κρυπτογράφηση δεδομένων είναι σημαντική για 4 βασικούς λόγους:

- Απόρρητο
- Ασφάλεια
- Ακεραιότητα δεδομένων
- Κανονισμοί

**Απόρρητο:** Η κρυπτογράφηση διασφαλίζει ότι κανείς δεν μπορεί να διαβάσει επικοινωνίες ή δεδομένα σε κατάσταση ηρεμίας, εκτός από τον προβλεπόμενο παραλήπτη ή τον νόμιμο κάτοχο δεδομένων. Αυτό αποτρέπει τους εισβολείς, τα δίκτυα διαφημίσεων, τους παρόχους υπηρεσιών Διαδικτύου και, σε ορισμένες περιπτώσεις, τις κυβερνήσεις από την υποκλοπή και την ανάγνωση ευαίσθητων δεδομένων, προστατεύοντας το απόρρητο των χρηστών.

**Ασφάλεια:** Η κρυπτογράφηση συμβάλλει στην αποφυγή παραβιάσεων δεδομένων, είτε τα δεδομένα διαβιβάζονται είτε βρίσκονται σε κατάσταση ηρεμίας. Εάν χαθεί ή κλαπεί μια εταιρική συσκευή και ο σκληρός δίσκος είναι σωστά κρυπτογραφημένος, τα δεδομένα σε αυτήν τη συσκευή θα εξακολουθούν να είναι ασφαλή. Ομοίως, οι κρυπτογραφημένες επικοινωνίες επιτρέπουν στα επικοινωνούντα μέρη να ανταλλάσσουν ευαίσθητα δεδομένα χωρίς να διαρρέουν τα δεδομένα.

**Ακεραιότητα δεδομένων:** Η κρυπτογράφηση βοηθά επίσης στην αποτροπή κακόβουλης συμπεριφοράς, όπως επιθέσεις εντός διαδρομής. Όταν τα δεδομένα μεταδίδονται μέσω του Διαδικτύου, η κρυπτογράφηση διασφαλίζει ότι αυτό που λαμβάνει ο παραλήπτης δεν έχει προβληθεί ή παραποιηθεί καθ' οδόν.

**Κανονισμοί:** Για όλους αυτούς τους λόγους, πολλοί βιομηχανικοί και κυβερνητικοί κανονισμοί απαιτούν από τις εταιρείες που χειρίζονται δεδομένα χρηστών να διατηρούν κρυπτογραφημένα αυτά τα δεδομένα. Παραδείγματα ρυθμιστικών προτύπων και προτύπων συμμόρφωσης που απαιτούν κρυπτογράφηση περιλαμβάνουν το HIPAA, το PCI-DSS και το GDPR.

### 3.10.4. Ο αλγόριθμος κρυπτογράφησης AES

Ο αλγόριθμος κρυπτογράφησης AES (γνωστός και ως αλγόριθμος Rijndael) είναι ένας συμμετρικός αλγόριθμος κρυπτογράφησης μπλοκ με μέγεθος μπλοκ/τεμαχίου 128 bit. Μετατρέπει αυτά τα μεμονωμένα μπλοκ χρησιμοποιώντας κλειδιά των 128, 192 και 256 bit. Μόλις κρυπτογραφήσει αυτά τα μπλοκ, τα ενώνει για να σχηματίσει το κρυπτογραφημένο κείμενο. Βασίζεται σε ένα δίκτυο υποκατάστασης-μετάθεσης, γνωστό και ως δίκτυο SP. Αποτελείται από μια σειρά συνδεδεμένων λειτουργιών, συμπεριλαμβανομένης της αντικατάστασης των εισόδων με συγκεκριμένες εξόδους (αντικαταστάσεις) και άλλων που περιλαμβάνουν ανακάτεμα bit (μεταθέσεις).

Χαρακτηριστικά του AES:

- Δίκτυο SP
- Επέκταση κλειδιού
- Byte Data
- Μήκος κλειδιού

**Δίκτυο SP**: Λειτουργεί σε μια δομή δικτύου SP αντί για μια δομή κρυπτογράφησης Feistel, όπως φαίνεται στην περίπτωση του αλγόριθμου DES.

**Επέκταση κλειδιού**: Απαιτείται ένα μόνο κλειδί κατά τη διάρκεια του πρώτου σταδίου, το οποίο αργότερα επεκτείνεται σε πολλά πλήκτρα που χρησιμοποιούνται σε μεμονωμένους γύρους.

**Byte Data**: Ο αλγόριθμος κρυπτογράφησης AES εκτελεί λειτουργίες σε δεδομένα byte αντί για δεδομένα bit. Επομένως, αντιμετωπίζει το μέγεθος μπλοκ 128-bit ως 16 byte κατά τη διαδικασία κρυπτογράφησης.

**Μήκος κλειδιού**: Ο αριθμός των γύρων που πρέπει να πραγματοποιηθούν εξαρτάται από το μήκος του κλειδιού που χρησιμοποιείται για την κρυπτογράφηση δεδομένων. Το μέγεθος κλειδιού 128 bit έχει δέκα γύρους, το μέγεθος κλειδιού 192 bit έχει 12 γύρους και το μέγεθος κλειδιού 256 bit έχει 14 γύρους.

Ο πίνακας που φαίνεται στην παρακάτω εικόνα είναι γνωστός ως πίνακας καταστάσεων. Ομοίως, το κλειδί που χρησιμοποιείται αρχικά επεκτείνεται σε (n+1) κλειδιά, με το n να είναι ο αριθμός των γύρων που πρέπει να ακολουθηθούν στη διαδικασία

κρυπτογράφησης. Έτσι, για ένα κλειδί 128-bit, ο αριθμός των γύρων είναι 16, με αρ. των κλειδιών που θα δημιουργηθούν είναι 10+1, που είναι συνολικά 11 κλειδιά.

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

Εικόνα 3.10.4.α πίνακας καταστάσεων.

Τα παρακάτω βήματα πρέπει να ακολουθούνται για κάθε μπλοκ διαδοχικά. Μετά την επιτυχή κρυπτογράφηση των μεμονωμένων μπλοκ, ο αλγόριθμος τα ενώνει μεταξύ τους για να σχηματίσει το τελικό κρυπτογραφημένο κείμενο. Τα βήματα είναι τα εξής:

- Προσθήκη κλειδιού γύρου
- Διαχωρισμός bytes
- Μετατόπιση σειρών
- Μίξη στηλών
- Προσθήκη κλειδιού γύρου

**Προσθήκη κλειδιού γύρου:** Μεταβιβάζετε τα δεδομένα μπλοκ που είναι αποθηκευμένα στον πίνακα κατάστασης μέσω μιας συνάρτησης XOR με το πρώτο κλειδί που δημιουργείται (K0). Μεταβιβάζει τον πίνακα καταστάσεων που προκύπτει ως είσοδο στο επόμενο βήμα.

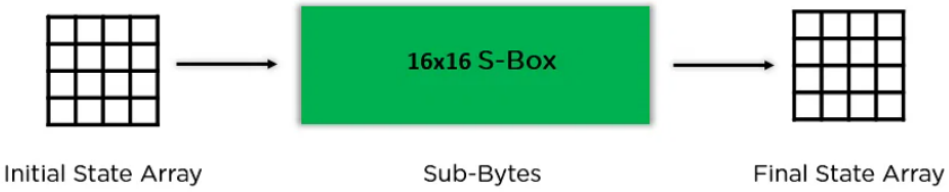
| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 6 | 7 | 8 | 5 |
| 11 | 12 | 9 | 10 |
| 16 | 13 | 14 | 15 |

XOR

| | | | |
|---|---|---|---|
| $K_0$ | $K_1$ | $K_2$ | $K_3$ |
| $K_4$ | $K_5$ | $K_6$ | $K_7$ |
| $K_8$ | $K_9$ | $K_{10}$ | $K_{11}$ |
| $K_{12}$ | $K_{13}$ | $K_{14}$ | $K_{15}$ |

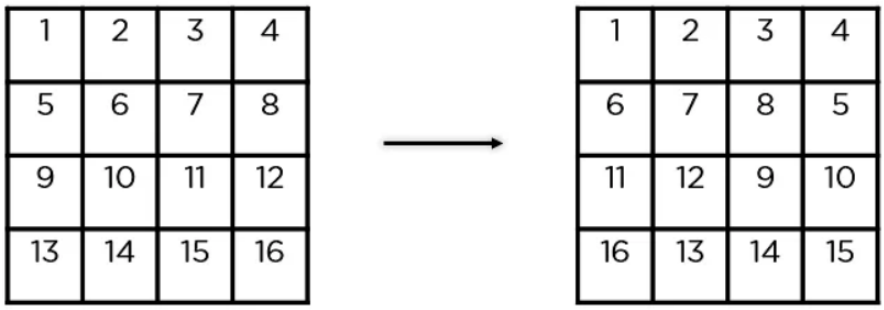
Εικόνα 3.10.4.β Προσθήκη κλειδιού γύρου.

**Διαχωρισμός bytes:** Σε αυτό το βήμα, μετατρέπει κάθε byte του πίνακα καταστάσεων σε δεκαεξαδικό, χωρισμένο σε δύο ίσα μέρη. Αυτά τα μέρη είναι οι γραμμές και οι στήλες, που αντιστοιχίζονται με ένα πλαίσιο αντικατάστασης (S-Box) για τη δημιουργία νέων τιμών για τον πίνακα τελικής κατάστασης.




Εικόνα 3.10.4.γ Διαχωρισμός bytes.

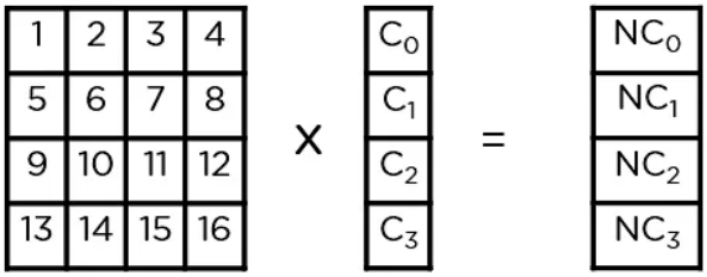
**Μετατόπιση σειρών:** Εναλλάσσει τα στοιχεία της σειράς μεταξύ τους. Παραλείπει την πρώτη σειρά. Μετατοπίζει τα στοιχεία στη δεύτερη σειρά, μια θέση προς τα αριστερά. Μετατοπίζει επίσης τα στοιχεία από την τρίτη σειρά δύο διαδοχικές θέσεις προς τα αριστερά και την τελευταία σειρά τρεις θέσεις προς τα αριστερά.




Εικόνα 3.10.4.δ Μετατόπιση σειρών.

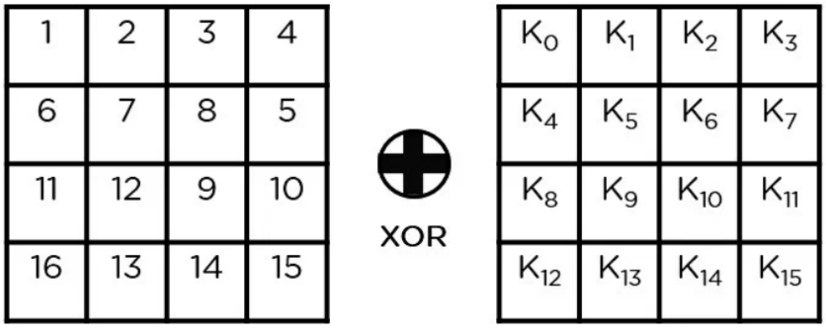
**Μίξη στηλών:** Πολλαπλασιάζει μια σταθερή μήτρα με κάθε στήλη στον πίνακα κατάστασης για να πάρει μια νέα στήλη για τον επόμενο πίνακα καταστάσεων. Μόλις πολλαπλασιαστούν

όλες οι στήλες με τον ίδιο σταθερό πίνακα, δημιουργείται ο πίνακας καταστάσεων για το επόμενο βήμα. Το συγκεκριμένο βήμα δεν περιλαμβάνεται στον τελευταίο γύρο.




Εικόνα 3.10.4.ε Μίξη στηλών.

**Προσθήκη κλειδιού γύρου**: Το αντίστοιχο κλειδί για τον γύρο έχει γίνει XOR με τον πίνακα κατάστασης που προκύπτει στο προηγούμενο βήμα. Εάν αυτός είναι ο τελευταίος γύρος, ο πίνακας καταστάσεων που προκύπτει γίνεται το κρυπτογραφημένο κείμενο για το συγκεκριμένο μπλοκ. Διαφορετικά, περνά ως είσοδος πίνακα νέας κατάστασης για τον επόμενο γύρο.




Εικόνα 3.10.4.στ Προσθήκη κλειδιού γύρου.

Στην παρούσα διπλωματική υλοποιείται ο αλγόριθμος AES με μέγεθος κλειδιού 128 bit.

# Κεφάλαιο 4

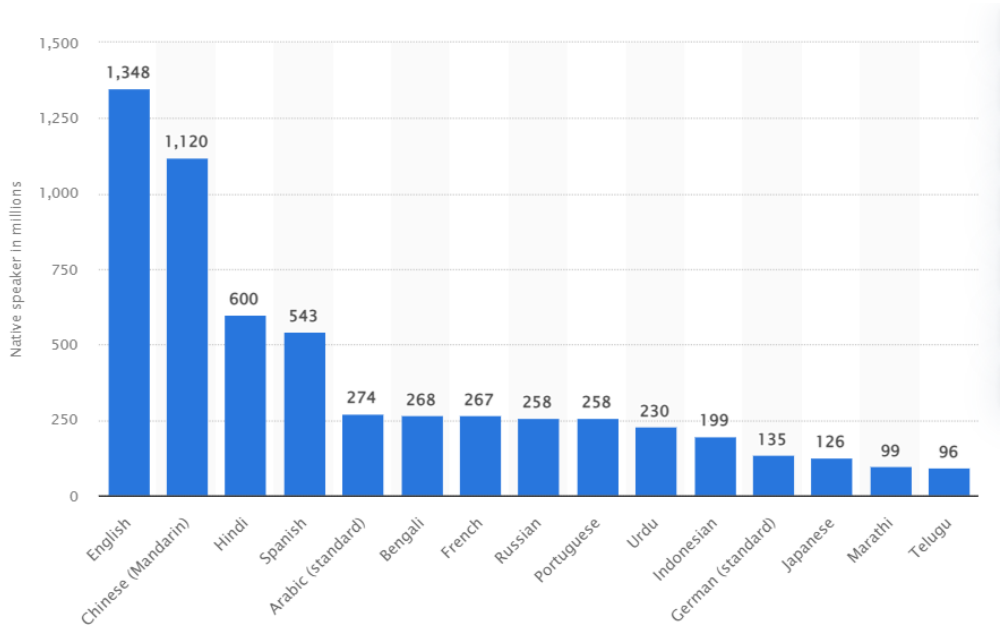
## 4.1 Επιλογή Γλώσσας

Ο 21ος αιώνας είναι ο αιώνας της διαφορετικότητας. Χώρες όπως οι ΗΠΑ, το Ηνωμένο Βασίλειο, η Νότια Αφρική, η Ινδία, η Αυστραλία κ.λπ. είναι πολύγλωσσες κοινωνίες. Αυτές οι χώρες έχουν τα αγγλικά ως επίσημη γλώσσα τους, αλλά ένας σημαντικός αριθμός ανθρώπων συνομιλεί στις μητρικές τους γλώσσες. Σχεδόν κάθε άλλη εξέχουσα πόλη διαθέτει

πολυγλωσσική κουλτούρα. Ως εκ τούτου, είναι λογικό οι εφαρμογές για κινητά να ανταποκρίνονται ανεξάρτητα από τη χώρα ή την πόλη.

Η επίτευξη παγκόσμιας παρουσίας είναι ζωτικής σημασίας τόσο για τις επιχειρήσεις όσο και για τις εφαρμογές ώστε να είναι επιτυχημένες σε αυτήν την εποχή. Οι εφαρμογές για κινητά που στοχεύουν σε διεθνές κοινό μέσω πολύγλωσσων εφαρμογών όχι μόνο αυξάνουν τα ποσοστά μετατροπών αλλά δημιουργούν μια μακροχρόνια σχέση με τους τελικούς χρήστες. Οι πολύγλωσσες εφαρμογές είναι πιο ελκυστικές και έχουν την τάση να κοινοποιούνται, γεγονός που συμβάλλει στη βελτίωση της προβολής της επωνυμίας της εφαρμογής. Όσο περισσότεροι χρήστες, τόσο πιο επιτυχημένη θα είναι η εφαρμογή. Για τον λόγο αυτό πρέπει να επιλέξουμε ποιες γλώσσες μπορούμε να υποστηρίξουμε μέσα στην εφαρμογή μας. Σύμφωνα με την ιστοσελίδα www.statista.com προκύπτει πως οι πέντε πιο ομιλούμενες γλώσσες σε όλον τον κόσμο είναι:

1. Αγγλικά με 1.348.000.000 φυσικούς ομιλητές
2. Μανδαρινικά (Κινέζικα) με 1.120.000.000 φυσικούς ομιλητές
3. Χίντι (Ινδικά) με 600.000.000 φυσικούς ομιλητές
4. Ισπανικά με 543.000.000 φυσικούς ομιλητές
5. Αραβικά με 274.000.000 φυσικούς ομιλητές




Εικόνα 4.1: Οι περισσότερες ομιλούμενες γλώσσες το έτος 2021.

Με βάση τα παραπάνω στατιστικά και τις μελέτες που έγιναν στο αλφάβητο κάθε μιας από τις πέντε πιο ομιλούμενες γλώσσες ανά το κόσμο, καταλήξαμε ότι η εφαρμογή μας μπορεί να υποστηρίξει τα Ελληνικά, τα Αγγλικά και τα Ισπανικά. Στο παιχνίδι που θα υλοποιηθεί θα γίνεται χρήση κεφαλαίων γραμμάτων χωρίς τόνους, ειδικά σύμβολα ή σημεία στίξης και λόγος είναι γιατί τα γράμματα θα δίνονται σε πλαίσια (κουτάκια) μέσα στο πλαίσιο της οθόνης, δηλαδή δεν θα γίνεται χρήση του πληκτρολογίου που διαθέτει το κινητό, ελαχιστοποιώντας έτσι τα γράμματα που θα πρέπει να μοιράζονται και να διαχειρίζεται ο κάθε παίχτης καθώς διαφορετικός ASCII χαρακτήρας είναι το *Ε* (έψιλον) από το *Έ* (έψιλον με τόνο).

Κάθε νέο έργο (project) που ξεκινάει με την χρήση του εργαλείου Android Studio, δημιουργείτε και ένας κατάλογος res/ στο ανώτερο επίπεδο του έργου. Μέσα σε αυτόν τον κατάλογο *res/* υπάρχουν υποκατάλογοι για διάφορους τύπους πόρων. Υπάρχουν επίσης μερικά προεπιλεγμένα αρχεία, όπως το *res/values/strings.xml*, το οποίο διατηρεί τις τιμές για τις συμβολοσειρές που θα χρησιμοποιηθούν σε μια εφαρμογή. Έτσι εμείς στην εφαρμογής μας θα δημιουργήσουμε τρία αρχεία string.xml, ένα για κάθε γλώσσα που θα υποστηρίξουμε μέσα στην εφαρμογή μας, όπου εκεί θα αποθηκεύσουμε τα κείμενα που θα περιέχει το παιχνίδι μας, αυτά αρχεία θα έχουν τίτλο:

- string.xml
- string.xml (el-rGR)
- string.xml (es-rES)

## 4.2 Διαχείριση λέξεων από το λεξικό

Η κυριά φιλοσοφία του παιχνιδιού είναι οι χρήστες να μπορούν να δημιουργούν λέξεις με την χρήση των τυχαίων γραμμάτων που θα τους έχουν δοθεί και θα έχουν στην διάθεση τους κατά την διάρκεια ενός γύρου. Συνεπώς το παιχνίδι θα πρέπει να μπορεί να αναγνωρίσει αν μια λέξη υπάρχει η όχι σε μια γλώσσα. Για τον λόγο αυτό θα χρειαστούμε τα αντίστοιχα λεξικά λέξεων των γλωσσών που επιλέξαμε ότι θα υποστηρίζονται μέσα στην εφαρμογή - παιχνίδι μας. Τα λεξικά αυτά θα πρέπει να είναι πλήρες και να μην βασίζονται σε στατιστικές για τις λέξεις που χρησιμοποιούνται περισσότερο στην καθημερινή μας ζωή ή σε λέξεις που χρησιμοποιούν οι χρήστες σε παιχνίδια λέξεων, για τον λόγο ότι δεν θα θέλαμε να μείνει κάποιος παίχτης δυσαρεστημένος από την μη βαθμολόγηση μιας, ναι μεν υπαρκτής λέξης σε μια γλώσσα αλλά όχι υπαρκτής στο λεξικό του παιχνιδιού.

Κατά συνέπεια εφόσον τα λεξικά θα είναι πλήρης, τότε τα αρχεία που θα τα περιέχουν θα είναι μεγάλου μεγέθους και με μια πρώτη σκέψη, δεν θα μπορούν να είναι κομμάτι του έργου της εφαρμογής μας, δηλαδή να βρίσκονται μαζί με τα αρχεία της εφαρμογής (αρχεία κώδικα, εικόνες, ήχοι κ.α). Θα πρέπει να βρίσκονται ξεχωριστά τα αρχεία αυτά, σε μια βάση δεδομένων. Μιας και το Firebase επιλέχθηκε ως βάση δεδομένων για τις ανάγκες της εφαρμογής μας, θα αποθηκεύσουμε τα τρία αυτά λεξικά εκει. Όπως αναφέρθηκε παραπάνω το Firebase αποθηκεύει τα δεδομένα με την μορφή κλειδί - τιμή (Key - Value).

Συνεπώς θα πρέπει να διαμορφωθεί το αρχείο έτσι ώστε ως κλειδί να τίθεται η κάθε λέξη και ως τιμή μια ασήμαντη τιμή μιας και δεν θα χρησιμοποιηθεί η τιμή. Η μορφή δηλαδή που θα αποθηκευτούν τα δεδομένα, η κάθε λέξη των λεξικών δηλαδή, θα είναι:

```
" GreekGrammar" {
    "AN" : "1",
    "ANA" : "1"
        ...
},
"EnglishGrammar "{
        ...
},
        ...
```

Έτσι όταν ο χρήστης θα βρίσκει μια λέξη το μόνο που πρέπει να κάνουμε είναι να προσπαθήσουμε να διαβάσουμε το περιεχόμενο της διαδρομής */GreekGrammar/λέξη-που-βρήκε.* Αν δεν υπάρχει περιεχόμενο στην διαδρομή αυτή τότε η λέξη που σχημάτισε ο χρήστης δεν υπάρχει στο λεξικό. Το Firebase έχει πολύ γρήγορη ανταπόκριση όταν η διαδρομή των ζητουμένων δεδομένων είναι γνωστή, θα μπορούσαμε να πούμε ότι ο χρονική πολυπλοκότητα είναι O(1). Με αυτόν τον τρόπο λύθηκε ένα πολύ σημαντικό πρόβλημα που αναφέρθηκε στην αρχή του κεφαλαίου αυτού, για τον χρόνο αναζήτησης σε ένα μεγάλο λεξικό.

Ταυτόχρονα με την λύση του προβλήματος της αναζήτησης λέξης σε ένα μεγάλο λεξικό, τίθεται και ένα ζήτημα το οποίο είναι, η συνεχής αλληλεπίδραση του πελάτη (client) με την βάση. Καθώς το Firebase είναι δωρεάν μέχρι έναν συγκεκριμένο αριθμό ανάγνωσης και εγγραφής (reads και writes) δεδομένων και έναν αριθμό ταυτοχρόνων συνδεδεμένων

clients, θέλουμε να ελαχιστοποιήσουμε τις αναγνώσεις, πέραν των βασικών αναγκών του μηχανισμού του παιχνιδιού, όπως δηλαδή η επικοινωνία με τον αντίπαλο κ.α.

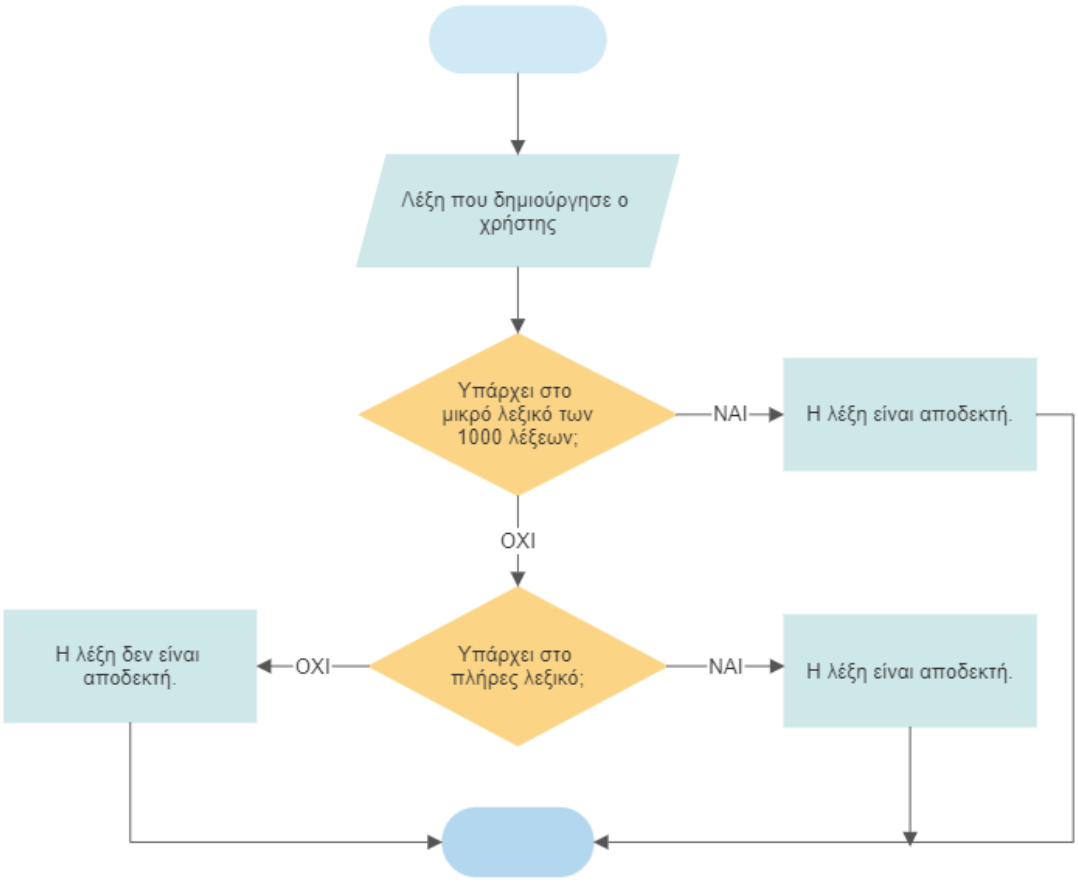
Πραγματοποιήθηκε μια μικρή έρευνα γύρω από την ψυχολογία του παίχτη και την δημιουργία λέξεων. Συγκεκριμένα:

- Το 45% είχε την τάση να δημιουργεί μικρές και πολλές λέξεις, όπως *εγώ, ναι, και, εσύ, αν* κ.α
- Το 25% είχε την τάση να δημιουργεί μεσαίου μεγέθους καθημερινές λέξεις όπως *αμάξι, παίζω, κέντρο* κ.α
- Το 30% είχε την τάση να δημιουργεί μεγάλου μεγέθους λέξεις όπως παραπάνω, *μπανάνα, επιλογή* κ.α


Εικόνα 4.2.α: Αποτελέσματα έρευνας ψυχολογία παίχτη.

Τα αποτελέσματα της ερευνάς ήταν πολύ σημαντικά καθώς προκύπτει το συμπέρασμα πως το 70% των παιχτών τείνουν να δημιουργούν λέξεις που χρησιμοποιούμε στην καθημερινότητα μας, δηλαδή της πιο συνηθισμένες λέξεις μια γλώσσας. Δημιουργήθηκε λοιπόν στην δημιουργία ενός μικρού λεξικού, για κάθε γλώσσα, με τις 1000 πιο συνηθισμένες λέξεις που χρησιμοποιούν οι άνθρωποι στην καθημερινότητα τους. Έτσι αποθηκεύσαμε αυτές τις 1000 λέξεις για κάθε γλωσσα σε μια λίστα σε αρχείο .java και το μόνο που έχουμε να

κάνουμε είναι να ελέγχουμε αν η λέξη που δημιούργησε ο παίχτης υπάρχει σε αυτό το μικρό λεξικό, αν δεν υπάρχει τότε θα ελέγξουμε στο πλήρες λεξικό που έχουμε αποθηκεύσει στο Firebase. Αν η λέξη υπάρχει στο πλήρες λεξικό τότε θα την κρατήσουμε σε μια λίστα για μελλοντική χρήση, με στόχο, αν ο χρήστης την δημιουργήσει πάλι σε κάποιο επόμενο γύρο να μην ψάξουμε πάλι στην βάση, μειώνοντας έτσι τις αναγνώσεις (reads) που γίνονται σε αυτήν. Αν δεν υπάρχει ούτε στο πλήρες λεξικό τότε αυτό σημαίνει πως η λέξη που δημιούργησε ο χρήστης δεν υπάρχει στην αντίστοιχη επιλεγμένη γλώσσα παιχνιδιού. Η σκέψη αυτού του αλγορίθμου που μόλις παρουσιάσαμε με ελεύθερο κείμενο οπτικοποιείται στο παρακάτω διάγραμμα ροής:




Εικόνα 4.2.β: Διάγραμμα ροής διαδικασίας ελέγχου ύπαρξης λέξης.

## 4.3 Μηχανισμός δημιουργίας λέξης με συγκεκριμένα γράμματα

Σε πολλά παιχνίδια λέξεων υπάρχει η δυνατότητα παροχής βοήθειας στον χρήστη, είτε δίνοντας του ένα στοιχείο για την λέξη που ψάχνει, εάν παραδείγματος χάριν πρόκειται για παιχνίδι λέξεων αναγραμματισμού ή και ακόμα παρέχοντας ολόκληρη την λέξη. Αυτή η παροχή βοήθειας παρόλο που με μια γρήγορη σκέψη φαίνεται να είναι άδικη ως προς τον αντίπαλο παίχτη ή ότι επηρεάζει την ροή εξέλιξης του παιχνιδιού εντούτοις είναι πολύ χρήσιμη, σημαντική ως προς την ψυχολογία του παίχτη καθώς υπάρχει η ανακούφιση πως ακόμα και εάν ο παίχτης "κολλήσει" υπάρχει τρόπος να συνεχίσει το παιχνίδι και φυσικά περιορισμένη με έναν αριθμό φορών που μπορεί να ζητηθεί. Συνήθως τα παιχνίδια περιορίζουν το πόσες φορές ο χρήστης μπορεί να ζητήσει βοήθεια σε έναν γύρο ή κατά την διάρκεια μιας μέρας. Τα παιχνίδια μπορεί να παρέχουν έναν αριθμό βοηθειών, είτε ως καθημερινή αμοιβή για τον χρήστη, είτε μέσω αμοιβής παρακολούθησης κάποιας διαφήμισης είτε ως ηλεκτρονικό αγαθό που μπορεί ο χρήστης να αγοράσει. Φυσικά η πολιτική για τις βοήθειες μπορεί να διαφέρει από παιχνίδι σε παιχνίδι.

Είναι λοιπόν προφανές πως το παιχνίδι πρέπει να παρέχει βοήθεια στον παίχτη στην προσπάθεια του να δημιουργήσει λέξεις στην διάρκεια ενός γύρου και γενικότερα του παιχνιδιού. Έτσι δημιουργείται η ανάγκη ύπαρξης κάποιου μηχανισμού που δεδομένου τα γράμματα ενός γύρου να είναι ικανός να δημιουργεί υπαρκτές λέξεις, που φυσικά υπάρχουν στο αντίστοιχο λεξικό της επιλεγμένης γλώσσας. Με μια γρήγορη σκέψη το πρόβλημα αυτό δείχνει να είναι πρόβλημα μηχανικής μάθησης και συγκεκριμένα εκπαίδευσης κάποιου νευρωνικού δικτύου ως προς την εκμάθηση δημιουργίας λέξεων. Μελετώντας όμως καλυτέρα το εν λόγο πρόβλημα γίνεται ξεκάθαρο πως πρόκειται για πρόβλημα αναζήτησης και κατάλληλης αναπαράστασης της βάσης δεδομένων (λεξικογραφικής ή μέσω κάποιου δένδρου).

### 4.3.1 Χρήση δέντρου προθέματος

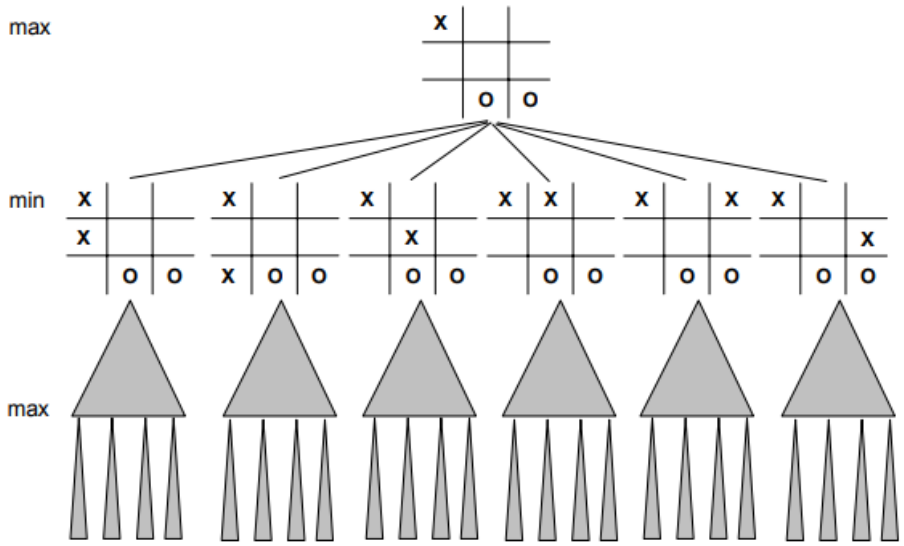
Η χρήση δέντρου προθέματος για την ανάπτυξη μηχανισμού δημιουργίας λέξεων με σκοπό την παροχή βοήθειας προς τους παίχτες, είναι ιδανική καθώς λύνει στοχευμένα το πρόβλημα αυτό καθώς μπορεί να αποθηκεύσει τις λέξεις χωρίζοντας την σε επίπεδα. Για παράδειγμα το γράμμα “Α” θα αποθηκευτεί σε κάποιον κόμβο στο πρώτο επίπεδο του δέντρου.

Η διαδικασία αποθήκευσης ξεκινάει από το πρώτο γράμμα της λέξης, τοποθετώντας το στο αντίστοιχο επίπεδο (πρώτο επίπεδο) και συνεχίζοντας με το δεύτερο γράμμα της λέξης αποθηκεύοντας όμως το πρώτο και το δεύτερο γράμμα ενιαία στο ακριβώς από κάτω επίπεδο του δέντρου. Η διαδικασία αυτή συνεχίζεται για όλα τα γράμματα της λέξης που έχουν απομείνει. Για παράδειγμα το γράμμα "Α" της λέξης "ΑΝΑ", θα βρίσκεται στο πρώτο επίπεδο του δέντρου. Στο παιχνίδι μας δημιουργούμε μια δομή δεδομένων δέντρου προθέματος που περιέχει τις χίλιες λέξεις του μικρού λεξικού που διαθέτουμε για κάθε γλώσσα, ακριβώς κάθε φορά που ο παίχτης ανοίγει την εφαρμογή, ώστε να υπάρχει για όλο το χρονικό διάστημα που η εφαρμογή παραμένει ανοιχτή και σε χρήση. Η δημιουργία του δέντρου προθέματος γίνεται με την βοήθεια και χρήση ξεχωριστού νήματος από το κύριο της εφαρμογής για μια πιο ομαλή εμπειρία χρήστη (UX).

## 4.4 Δημιουργία Μηχανισμού BOT

Σε αρκετά παιχνίδια πραγματικού χρόνου πολλαπλών παικτών (multiplayer), και όχι μόνο, γίνεται χρήση bot, δηλαδή, ενός χαρακτήρα που ελέγχεται από έναν υπολογιστή. Οι πολύ μεγάλες ανάγκες που υπάρχουν τόσο από την πλευρά των παιχνιδιών όσο και από την πλευρά των χρηστών, καθιστούν την χρήση bot αναγκαία. Η χρήση bot γίνεται για ποικίλους λόγους αλλά ο κυριότερος είναι η ψευδαίσθηση που δίνεται στον χρήστη ότι παίζει εναντίων πραγματικού χρήστη. Στα παιχνίδια όπου οι χρήστες μπορούν να συνδέονται και να παίζουν με άλλους χρήστες, η ύπαρξη διαθέσιμων χρηστών (Online Users) είναι απαραίτητη καθώς χωρίς αυτούς δεν νοείται και ο όρος «για πολλαπλούς παίκτες» (multiplayer). Φυσικά μια τέτοια περίπτωση είναι δυνατόν να συμβεί όταν το παιχνίδι είναι νέο στην αγορά και δεν έχει αποκτήσει όγκο σε χρήστες ή οι χρήστες που έχει δεν επαρκούν ώστε να καλύπτονται οι ανάγκες του παιχνιδιού σε εικοσιτετράωρη βάση. Ένα κοινό πρόβλημα που παρατηρείται σε νέα παιχνίδια ή παιχνίδια με μικρό κοινό είναι η μεγάλη αναμονή στο «δωμάτιο εύρεσης αντιπάλου». Αυτή η μεγάλη αναμονή δημιουργεί δυσαρέσκεια και εκνευρισμό στους χρήστες με αποτέλεσμα οι περισσότεροι να απομακρύνονται από το εν λόγω παιχνίδι ή να το βαθμολογούν αρνητικά, επηρεάζοντας έτσι σημαντικά το «User Growth» του. Για αυτό τον λόγο πολλά παιχνίδια πραγματικού χρόνου χρησιμοποιούν bots, με σκοπό να περιορίσουν τα προβλήματα αυτά. Πως λειτουργεί λοιπόν ένας τέτοιος μηχανισμός; Είναι ένας μηχανισμός που χρησιμοποιεί τεχνητή νοημοσύνη με σκοπό να εκπαιδευτεί πάνω στον σκοπό του παιχνιδιού ώστε να μπορεί να λαμβάνει αποφάσεις αλλά και να αποκτήσει ρεαλιστική

συμπεριφορά που να θυμίζει αληθινό χρήστη. Ένα παράδειγμα ενός τέτοιου μηχανισμού είναι ένα bot που παίζει τρίλιζα ενάντιων ενός ανθρώπου. Ο υπολογιστής κάνει χρήση του γνωστού αλγορίθμου στον χώρο της τεχνητής νοημοσύνης, τον Min-Max, ενός αλγορίθμου αναζήτησης με αντιπαλότητα, με σκοπό να βρεθεί η βέλτιστη κίνηση για τον εκάστοτε γύρο.




Εικόνα 4.4: Οι πιθανές κινήσεις που αναζητά ο αλγόριθμος Min-Max σε ένα παιχνίδι τρίλιζας.

Στο παιχνίδι που αναπτύχθηκε για τις ανάγκες τις παρούσας διπλωματικής, έπρεπε να δημιουργηθεί ένα bot όπου όχι μόνο να βρίσκει λέξεις, αλλά να είναι σε θέση να αποφασίζει κάθε πότε πρέπει να παίζει, να έχει δηλαδή μια ρεαλιστική συμπεριφορά. Θα μπορούσε να είχε μια συμπεριφορά που θα βασιζόταν στην τύχη, όμως αυτό ίσως ευνοούσε ή αδικούσε τους παίκτες γιατί δεν θα λαμβάνονταν υπόψιν το επίπεδο του κάθε παίκτη. Έτσι λοιπόν αποφασίστηκε πως η κατάλληλη συμπεριφορά που θα πρέπει να έχει το bot θα πρέπει να προσαρμόζεται με τον εκάστοτε παίκτη-χρήστη που έχει να αντιμετωπίσει. Μερικά από τα δεδομένα που μπορούμε να συλλέξουμε ώστε να συμβάλουν στην δημιουργία του μηχανισμού του bot είναι:

1) Πόσες λέξεις βρήκε ο χρήστης στην διάρκεια ενός γύρου;

2) Πόσους πόντους μάζεψε ο χρήστης από κάθε λέξη ή συνολικά από όλες τις λέξεις;

3) Το μήκος των λέξεων (ποιο εύκολα βρίσκουμε και σχηματίζουμε την λέξη ΕΓΩ σε σχέση με την λέξη ΕΓΩΙΣΤΗΣ)

Το bot θα έπρεπε να είναι θέση να αποφασίσει πόσες λέξεις θα σχηματίσει και ποσό συχνά θα παίζει. Δηλαδή θα σχηματίζει «ΛΙΓΕΣ» λέξεις και θα παίζει «ΛΙΓΟΤΕΡΟ ΣΥΧΝΑ», «ΜΕΤΡΙΕΣ» λέξεις και θα παίζει «ΣΥΧΝΑ» ή «ΠΟΛΛΕΣ» λέξεις και θα παίζει «ΑΡΚΕΤΑ ΣΥΧΝΑ»; Σε αυτά τους είδους τα ερωτήματα παρατηρούμε πως υπάρχει «ασάφεια», η έννοια που συσχετίζεται με την ποσοτικοποίηση της πληροφορίας σε μη-ακριβή δεδομένα, όπως είναι οι λέξεις που προαναφέρθηκαν «ΛΙΓΕΣ», «ΜΕΤΡΙΕΣ», «ΑΡΚΕΤΑ ΣΥΧΝΑ» κτλ. Τέτοιου είδους πληροφορίες που περιέχουν ασάφεια μπορούμε να τις διαχειριστούμε με μεγάλη επιτυχία με την χρήση της Ασαφής λογικής (Fuzzy Logic) όπως ακριβώς επιγράφτηκε στο κεφάλαιο 3.

## 4.5 Εφαρμογή Ασαφούς Λογικής

Όπως αναφέρθηκε και προηγούμενος, στην πρόταση, το BOT θα σχηματίζει «ΛΙΓΕΣ» λέξεις και θα παίζει «ΑΡΚΕΤΑ ΣΥΧΝΑ» δεν υπάρχει μια σαφής ποσοτικοποίησή της. Δηλαδή δεν μπορούμε να πούμε ότι η φράση «ΛΙΓΕΣ» και η φράση «ΑΡΚΕΤΑ ΣΥΧΝΑ» είναι αριθμοί, παραδείγματος χάριν το 3 και 7 αντίστοιχα. Η ασαφής λογική όμως μπορεί να κάνει κάτι τέτοιο εφόσον χρησιμοποιεί λεκτικές μεταβλητές, οι οποίες διαχωρίζονται στο χώρο ορισμού τους. Κάποιες φορές δεν έχει σημασία λοιπόν η ακριβής τιμή, αλλά ένας ποιοτικός της χαρακτηρισμός. Παρακάτω εξηγείται αυτή η διαδικασία:

Έστω η προηγούμενη πρόταση, και η μεταβλητή «παίζει» που θέλουμε να αποδώσουμε. Η ασαφής λογική δημιουργεί την λεκτική μεταβλητή «παίζει» την οποία και διαχωρίζει σε κάποιες κατηγορίες-ετικέτες όπως:

- «ΛΙΓΕΣ»,
- «ΜΕΤΡΙΕΣ»,
- «ΠΟΛΛΕΣ»,
- «ΠΑΡΑ ΠΟΛΛΕΣ»

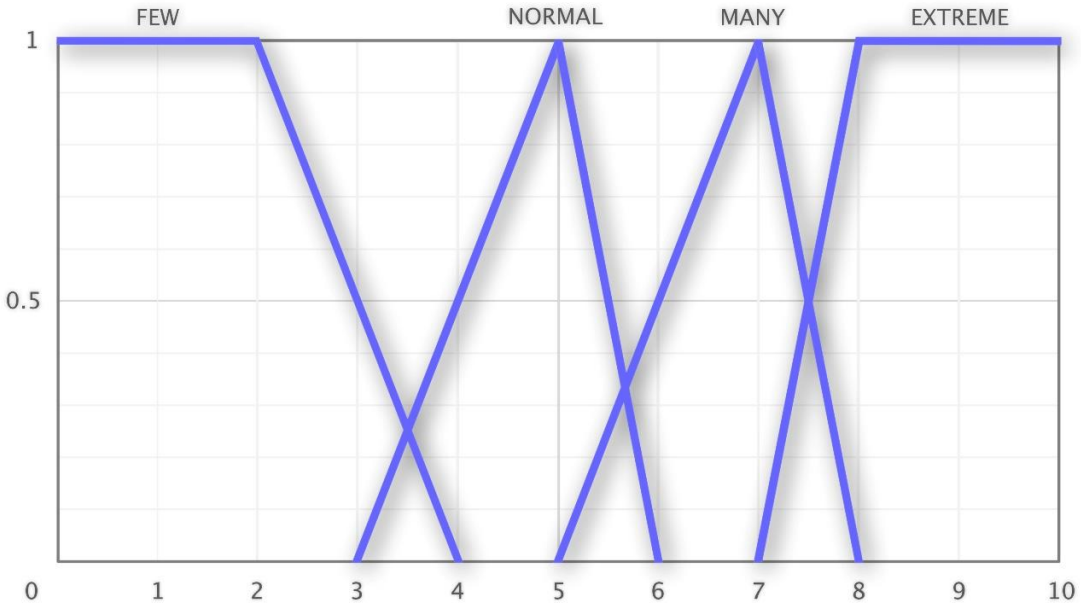
Κάθε κατηγορία-ετικέτα είναι ένα «ασαφές σύνολο». Πόσες φορές θα παίξει το bot στην διάρκεια ενός γύρου; Η ασαφής λογική απαντά ότι κατά 30% θα παίξει «ΛΙΓΕΣ» φορές

και κατά 5% θα παίξει «ΜΕΤΡΙΕΣ» φορές. Έτσι λοιπόν δημιουργείται η ανάγκη της απόδοσης διακριτών τιμών και των αντίστοιχων ετικετών τους στα παραπάνω ασαφής σύνολα. Όπως προαναφέρθηκε σε προηγουμένη παράγραφο, το bot θα πρέπει να προσαρμόζεται ανάλογα με τον αντίπαλο - χρήστη που έχει να αντιμετωπίσει. Για τον λόγο αυτό, το bot, θα πρέπει να έχει την ικανότητα να αποφασίζει πόσες λέξεις θα σχηματίζει και σε τι συχνότητα θα παίζει αυτές τις λέξεις. Η προσαρμογή αυτή θα διεξάγεται με την συλλογή δεδομένων από τον εκάστοτε χρήστη. Συγκεκριμένα θα αποθηκεύεται στην βάση δεδομένων του παιχνιδιού, οι συνολικοί πόντοι του χρήστη που συγκέντρωσε στον τελευταίο γύρο που έπαιξε και θα λαμβάνεται υπόψιν ο μέσος όρος των 5 τελευταίων γυρών. Με την πληροφορία αυτή, και με την εφαρμογή της ασαφούς λογικής, θα είμαστε σε θέση να δώσουμε απάντηση στο ερώτημα «Πόσες λέξεις θα σχηματίζει το bot στην διάρκεια ενός γύρου και σε τι συχνότητα θα παίζει αυτές τις λέξεις;» Έτσι αν θεωρήσουμε ότι ένας γύρος διαρκεί κατά μέσο όρο 1 λεπτό, τότε θα πρέπει να αποδοθούν διακριτές τιμές για:

1. Τον αριθμό των λέξεων που σχημάτισε ο χρήστης στην διάρκεια ενός γύρου.
2. Τους συνολικούς πόντους που συγκέντρωσε ο χρήστης στην διάρκεια ενός γύρου.
3. Την συχνότητα εκφρασμένη σε δευτερόλεπτα με την οποία θα παίζει το bot.

Αποτίμηση διακριτών τιμών και ετικετών για τον αριθμό των λέξεων:

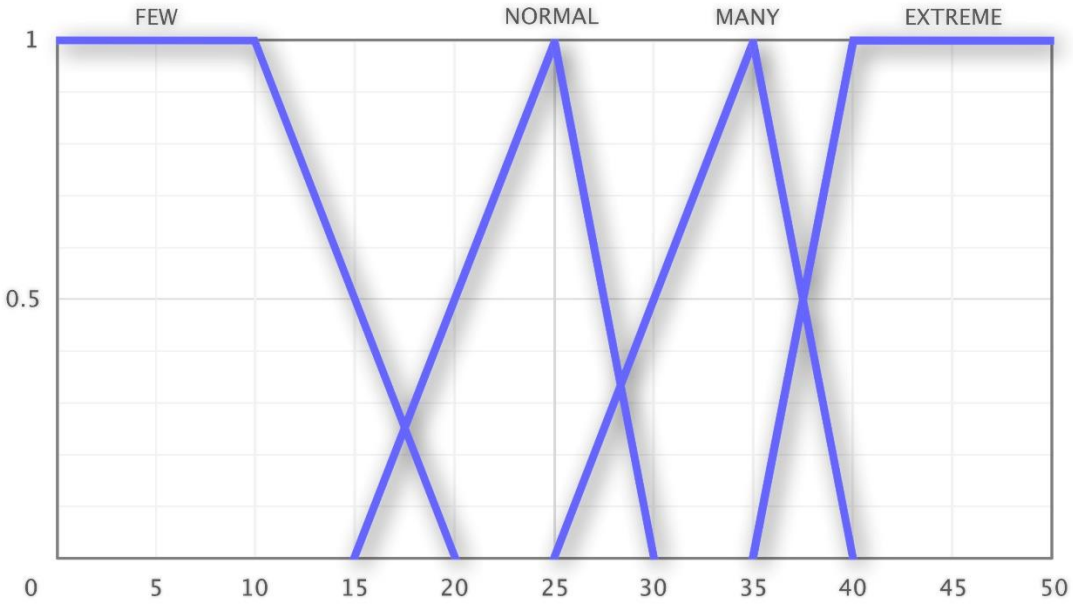
- Από 0 έως 4 λέξεις, ετικέτα: «ΛΙΓΕΣ»
- Από 3 έως 6 λέξεις, ετικέτα: «ΜΕΤΡΙΕΣ»
- Από 5 έως 8 λέξεις, ετικέτα: «ΠΟΛΛΕΣ»
- Από 7 λέξεις και πάνω, ετικέτα: «ΠΑΡΑ ΠΟΛΛΕΣ»




Εικόνα 4.5.α: Συνάρτηση συμμετοχής ασαφών συνόλων για τον αριθμό λέξεων.

Αποτίμηση διακριτών τιμών και ετικετών για τους συνολικούς πόντους:

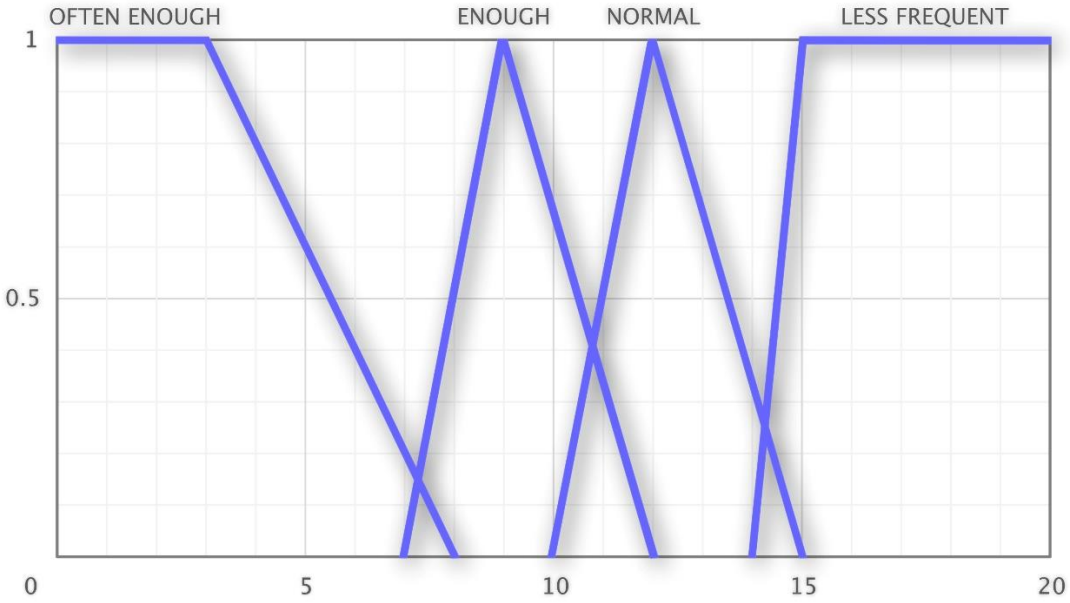
- Από 0 έως 20 πόντοι, ετικέτα: «ΛΙΓΟΙ»
- Από 15 έως 30 πόντοι, ετικέτα: «ΜΕΤΡΙΟΙ»
- Από 25 έως 40 πόντοι, ετικέτα: «ΠΟΛΛΟΙ»
- Από 35 πόντοι και πάνω, ετικέτα: «ΠΑΡΑ ΠΟΛΛΟΙ»




Εικόνα 4.5.β: Συνάρτηση συμμετοχής ασαφών συνόλων για τους συνολικούς πόντους.

Αποτίμηση διακριτών τιμών και ετικετών για την συχνότητα με την οποία θα παίζει το BOT:

- Από 0 έως 8 δευτερόλεπτα, ετικέτα: «ΑΡΚΕΤΑ ΣΥΧΝΑ»
- Από 7 έως 12 δευτερόλεπτα, ετικέτα: «ΣΥΧΝΑ»
- Από 10 έως 15 δευτερόλεπτα, ετικέτα: «ΚΑΝΟΝΙΚΑ»
- Από 14 δευτερόλεπτα και πάνω, ετικέτα: «ΛΙΓΟΤΕΡΟ ΣΥΧΝΑ»




Εικόνα 4.5.γ: Συνάρτηση συμμετοχής ασαφών συνόλων για την συχνότητα με την οποία θα παίζει το bot

Εφόσον ορίσθηκαν οι ετικέτες με τις αντίστοιχες διακριτές τιμές τους πλέον μπορεί να πραγματοποιηθεί η ασαφής εξαγωγή συμπερασμάτων μέσα από 3 εξής στάδια:

1. Ασαφοποίηση
2. Εκτίμηση κανόνων και Συνάθροιση εξόδων
3. Αποσαφοποίηση

### 4.4.1 Εφαρμογή Ασαφοποίησης

Όπως έχει αναφερθεί και στο προηγούμενο κεφάλαιο, σε αυτό το στάδιο γίνεται η μετατροπή των "ξερών" εισόδων σε ασαφείς, δηλαδή στους αντίστοιχους βαθμούς συμμέτοχης. Για τον λόγο αυτόν είναι απαραίτητο να ορισθούν και οι ασαφείς κανόνες οι οποίοι αντιπροσωπεύουν καλυτέρα το ζητούμενο αποτέλεσμα, δηλαδή, πόσες λέξεις θα δημιουργήσει το bot και με την συχνότητα θα παίζει αυτές τις λέξεις. Οι κανόνες που έχουν

ορισθεί δεν είναι σύνθετοι καθώς δεν γίνεται ούτε χρήση των τελεστών ΚΑΙ (AND) ή Ή (OR) στις συνθήκες αυτών αλλά ούτε γίνεται χρήση περισσότερων κανόνων του ενός συμπερασμάτων.

Κανόνες ως προς των αριθμό των λέξεων:

- Κανόνας 1ος: Αν ο χρήστης έχει συγκεντρώσει «ΛΙΓΟΥΣ» πόντους τότε το bot θα δημιουργήσει «ΛΙΓΕΣ» λέξεις.
- Κανόνας 2ος: Αν ο χρήστης έχει συγκεντρώσει « ΜΕΤΡΙΟΥΣ» πόντους τότε το bot θα δημιουργήσει «ΜΕΤΡΙΕΣ» λέξεις.
- Κανόνας 3ος: Αν ο χρήστης έχει συγκεντρώσει «ΠΟΛΛΟΥΣ» πόντους τότε το bot θα δημιουργήσει «ΠΟΛΛΕΣ» λέξεις.
- Κανόνας 4ος: Αν ο χρήστης έχει συγκεντρώσει «ΠΑΡΑ ΠΟΛΛΟΥΣ» πόντους τότε το bot θα δημιουργήσει «ΠΑΡΑ ΠΟΛΛΕΣ» λέξεις.

Κανόνες ως προς την συχνότητα με την οποία θα παίζει το bot:

- Κανόνας 1ος: Αν ο χρήστης έχει συγκεντρώσει «ΛΙΓΟΥΣ» πόντους τότε το bot θα παίζει «ΛΙΓΟΤΕΡΟ ΣΥΧΝΑ».
- Κανόνας 2ος: Αν ο χρήστης έχει συγκεντρώσει «ΜΕΤΡΙΟΥΣ» πόντους τότε το bot θα παίζει «ΚΑΝΟΝΙΚΑ».
- Κανόνας 3ος: Αν ο χρήστης έχει συγκεντρώσει «ΠΟΛΛΟΥΣ» πόντους τότε το bot θα παίζει «ΣΥΧΝΑ».
- Κανόνας 4ος: Αν ο χρήστης έχει συγκεντρώσει «ΠΑΡΑ ΠΟΛΛΟΥΣ» πόντους τότε το bot θα παίζει «ΑΡΚΕΤΑ ΣΥΧΝΑ».

Ακολουθώντας τους παραπάνω ασαφείς κανόνες αλλά και την συνάρτηση συμμετοχής ασαφών συνόλων για τον αριθμό των λέξεων και με βάση την συνάρτηση της εικόνα 3.8.3β οι 16 πόντοι για παράδειγμα είναι 0.4 FEW, 0.1 NORMAL, 0 MANY και 0 EXTREME.

### 4.4.2 Εκτίμηση κανόνων και Συνάθροιση εξόδων

Στο στάδιο αυτό, γίνεται ο υπολογισμός του συνδυασμένου βαθμού συμμέτοχης και η εφαρμογή του πάνω στην συνάρτηση συμμέτοχης του συμπεράσματος. Λόγω της απλότητας των ασαφών κανόνων, το στάδιο αυτό έχει ενοποιηθεί με το στάδιο της αποσαφοποίησης.

Ο παρακάτω πίνακας είναι το αποτέλεσμα του Fuzzy Logic συστήματος για πόντους από 0 έως 50. Στην πρώτη στήλη έχουμε τους πόντους που έχει σχηματίσει ο χρήστης στο πιο πρόσφατο γύρο που έχει παίξει ή από τον μέσο όρο των πόντων των 5 τελευταίων γύρων. Στην δεύτερη στήλη έχουμε τον αριθμό λέξεων που πρέπει να σχηματίσει το bot και στην τρίτη στήλη έχουμε την συχνότητα (σε δευτερόλεπτα) με την οποία θα τις σχηματίζει.

| Πόντοι | Λέξεις | Συχνότητα (sec) |
|---|---|---|
| 0 | 2.5 | 17 |
| 1 | 2.5 | 17 |
| 2 | 2.5 | 17 |
| 3 | 2.5 | 17 |
| 4 | 2.5 | 17 |
| 5 | 2.5 | 17 |
| 6 | 2.5 | 17 |
| 7 | 2.5 | 17 |
| 8 | 2.5 | 17 |
| 9 | 2.5 | 17 |
| 10 | 2.5 | 17 |
| 11 | 2.5 | 17 |
| 12 | 2.5 | 17 |
| 13 | 2.5 | 17 |
| 14 | 2.5 | 17 |
| 15 | 2.5 | 17 |
| 16 | 2.9 | 16.2 |
| 17 | 3.3 | 15.3 |
| 18 | 3.7 | 14.4 |
| 19 | 4.1 | 13.5 |
| 20 | 4.5 | 12.5 |
| 21 | 4.5 | 12.5 |

| | | |
|---|---|---|
| 22 | 4.5 | 12.5 |
| 23 | 4.5 | 12.5 |
| 24 | 4.5 | 12.5 |
| 25 | 4.5 | 12.5 |
| 26 | 4.7 | 12.2 |
| 27 | 4.9 | 11.9 |
| 28 | 5.1 | 11.6 |
| 29 | 5.3 | 11.3 |
| 30 | 6.5 | 9.5 |
| 31 | 6.5 | 9.5 |
| 32 | 6.5 | 9.5 |
| 33 | 6.5 | 9.5 |
| 34 | 6.5 | 9.5 |
| 35 | 6.5 | 9.5 |
| 36 | 6.7 | 8.7 |
| 37 | 6.9 | 8 |
| 38 | 7.1 | 7.3 |
| 39 | 7.3 | 6.7 |
| 40 | 8.5 | 4 |
| 41 | 8.5 | 4 |
| 42 | 8.5 | 4 |
| 43 | 8.5 | 4 |
| 44 | 8.5 | 4 |
| 45 | 8.5 | 4 |
| 46 | 8.5 | 4 |
| 47 | 8.5 | 4 |
| 48 | 8.5 | 4 |
| 49 | 8.5 | 4 |
| 50 | 8.5 | 4 |

Πίνακας 4.4.2.α: Οι λέξεις και η συχνότητα που πρέπει να τις δημιουργήσει το bot δεδομένου των πόντων του χρήστη.

## 4.5 Σύστημα Επιπέδου (Level System)

Στα βιντεοπαιχνίδια, ένα επίπεδο είναι κάθε χώρος που είναι διαθέσιμος στον παίκτη κατά τη διάρκεια της ολοκλήρωσης ενός στόχου. Τα επίπεδα των βιντεοπαιχνιδιών έχουν γενικά προοδευτικά αυξανόμενη δυσκολία να προσελκύσουν παίκτες με διαφορετικά επίπεδα δεξιοτήτων. Κάθε επίπεδο μπορεί να παρουσιάζει νέες έννοιες και προκλήσεις με σκοπό να διατηρήσει το ενδιαφέρον ενός παίκτη για το εν λόγο παιχνίδι.

Σε παιχνίδια ανοιχτού κόσμου (open world) τα επίπεδα είναι περιοχές ενός μεγαλύτερου κόσμου. Τα παιχνίδια μπορεί επίσης να διαθέτουν διασυνδεδεμένα επίπεδα, που αντιπροσωπεύουν τοποθεσίες. Αν και η πρόκληση σε ένα παιχνίδι είναι συχνά να νικήσεις κάποιου είδους χαρακτήρα, τα επίπεδα μερικές φορές σχεδιάζονται με μια πρόκληση κίνησης, όπως η επίτευξη αλμάτων σε μια διαδρομή με εμπόδια. Οι παίκτες πρέπει να κρίνουν την απόσταση μεταξύ των πλατφορμών ή των προεξοχών και να πηδήξουν με ασφάλεια για να φτάσουν στην επόμενη περιοχή. Αντίθετα σε παιχνίδια παζλ ή λέξεων, τα επίπεδα εκτός από την εισαγωγή νέων χαρακτηριστικών και προκλήσεων, μπορεί να περιγράφουν την εμπειρία που διαθέτει ο παίχτης και τις δεξιότητες του στο εν λόγο παιχνίδι. Σε παιχνίδια πολλαπλών παιχτών, το επίπεδο ενός παίχτη μπορεί να χρησιμοποιηθεί για την εύρεση κατάλληλου αντίπαλου, κοντά στο επίπεδο του παίχτη.

Όπως έχει ήδη αναφερθεί, η αντιστοίχιση παιχτών στο δωμάτιο με τους τυχαίους αντίπαλους γίνεται με βάση τον μέσο ορό των πόντων των λέξεων που έχει συγκεντρώσει ο χρήστης στα 5 τελευταία του παιχνίδια. Έτσι, στην δικιά μας περίπτωση, το επίπεδο του κάθε παίχτη δεν θα χρησιμοποιείται για την εύρεση κατάλληλου αντίπαλου αλλά ως μονάδα μέτρησης των δεξιοτήτων και ικανοτήτων καθώς και του χρόνου στον οποίο ο χρήστης παίζει το παιχνίδι.

Βασικό συστατικό των επίπεδων είναι οι πόντοι εμπειρίας (συχνά συντομεύεται ως exp ή XP). Αυτή η μονάδα μέτρησης χρησιμοποιείται για την ποσοτικοποίηση της εμπειρίας ζωής και της εξέλιξης ενός χαρακτήρα παίκτη στο παιχνίδι. Έτσι αν ο χρήστης συγκεντρώσει κάποιο προκαθορισμένο αριθμό πόντων εμπειρίας αλλάζει επίπεδο του και πηγαίνει στο επόμενο. Για παράδειγμα αν ο χρήστης βρίσκεται στο επίπεδο 1 και συγκεντρώσει παραπάνω από 154 πόντους εμπειρίας τότε το επίπεδο του θα αλλάξει και θα πάει στο επίπεδο 2. Συνηθίζεται δε οι ανάλογοι αριθμοί των πόντων εμπειρίας που πρέπει να συγκεντρωθούν ώστε ο χρήστης να αλλάξει επίπεδο και να πάει στο επόμενο, να αυξάνονται εκθετικά. Παρακάτω παρουσιάζεται

ο αλγόριθμος που δημιουργήθηκε για τις ανάγκες τις παρούσας διπλωματικής με τίτλο «Exponential XP» που υπολογίζει των αριθμό των πόντων εμπειρίας που αναλογούν στο αντίστοιχο επίπεδο.

**Algorithm** Exponential XP
**Input:** Current level of user.
**Output**: The number of experience points corresponding to the respective level.
1: *xp* ← 150
2: *factor* ← 0.03
3: **for** *i* ∈ 1, ..., *level* **do**
4: **if** *i* > 50 *factor* ← 0.01
5: *xp* ← *xp* + (*factor* * *xp*)
6: **return** *xp*

Ο παραπάνω αλγόριθμος δέχεται ως είσοδο τον αριθμό του επιπέδου που βρίσκεται ο χρήστης και δίνει ως έξοδο τον αριθμό των πόντων εμπειρίας που πρέπει να συγκεντρωθούν ώστε ο χρήστης να αλλάξει επίπεδο. Παρατηρούμε πως υπάρχουν δυο μεταβλητές, η πρώτη είναι ο αριθμός των πόντων εμπειρίας ζωής και αρχικοποιείται στην τιμή 150 ενώ η δεύτερη μεταβλητή είναι ο παράγοντας της αύξησης της πρώτης ανά επίπεδο, με αρχική τιμή 0.03 που δηλώνει, ποσοστιαία αύξηση ανά επίπεδο των πόντων εμπειρίας της τάξης 3%. Επιπλέον στον αλγόριθμο υπάρχει μια επαναληπτική διαδικασία όπου υπολογίζει μέχρι και τον αριθμό του επιπέδου του χρήστη που δόθηκε ως είσοδο, τον αντίστοιχο αριθμό των πόντων εμπειρίας. Ο παράγοντας αύξησης αλλάζει μετά το επίπεδο 50 και γίνεται 1% και ο λόγος είναι για την αποτροπή της ακραίας αύξησης των απαιτούμενων πόντων εμπειρίας σε κλίμακα επιπέδου 1 έως 999.

Η εκθετική συνάρτηση που πρέπει να υπολογίσουμε και περιγράφει τον παραπάνω αλγόριθμο έχει τη μορφή:

$$f(t)=A_0 e^{kt}$$

Εκτελώντας τον παραπάνω αλγόριθμο μέχρι και το επίπεδο 50 προκύπτουν οι παρακάτω διακριτές τιμές:

| Επίπεδο | Πόντοι Εμπειρίας |
|---|---|
| 1 | 154 |

| | |
|---|---|
| 2 | 159 |
| 3 | 163 |
| 4 | 168 |
| 5 | 173 |
| 6 | 179 |
| 7 | 184 |
| 8 | 190 |
| 9 | 195 |
| 10 | 201 |
| 11 | 207 |
| 12 | 213 |
| 13 | 220 |
| 14 | 226 |
| 15 | 233 |
| 16 | 240 |
| 17 | 247 |
| 18 | 255 |
| 19 | 263 |
| 20 | 270 |
| 21 | 279 |
| 22 | 287 |
| 23 | 296 |
| 24 | 304 |
| 25 | 314 |
| 26 | 323 |
| 27 | 333 |
| 28 | 343 |
| 29 | 353 |
| 30 | 364 |
| 31 | 375 |
| 32 | 386 |
| 33 | 397 |
| 34 | 409 |
| 35 | 422 |

| | |
|---|---|
| 36 | 434 |
| 37 | 447 |
| 38 | 461 |
| 39 | 475 |
| 40 | 489 |
| 41 | 503 |
| 42 | 519 |
| 43 | 534 |
| 44 | 550 |
| 45 | 567 |
| 46 | 584 |
| 47 | 601 |
| 48 | 619 |
| 49 | 638 |
| 50 | 657 |

Πίνακας 4.5.α: Τα Επίπεδα και οι αντίστοιχοι πόντοι εμπειρίας τους, με ρυθμό αύξησης 3%.

Για να λύσουμε για $A_0$ και $k$, πρέπει να λυθούν οι ακόλουθες δύο ταυτόχρονες εξισώσεις:

$$y_1 = A_0 e^{kt_1}$$

$$y_2 = A_0 e^{kt_2}$$

Βήμα 1: Το πρώτο βήμα αποτελείται από τη διαίρεση και των δύο πλευρών της ισότητας για τις εξισώσεις 1 και 2 παραπάνω προκειμένου να ακυρωθεί το $A_0$, οπότε παίρνουμε:

$$\frac{y1}{y2} = \frac{e^{kt1}}{e^{kt2}}$$

Βήμα 2ο: Διαιρώντας και τις δύο πλευρές και των δύο εξισώσεων έχουμε εξαλείψει το $A_0$, και τώρα μπορούμε να πάρουμε την παραπάνω εξίσωση και να λύσουμε για $k$:

$$k = \frac{1}{t1 - t2} \ln\left(\frac{y1}{y2}\right)$$

Βήμα 3ο: Τώρα που βρήκαμε $k$ μπορούμε να επιστρέψουμε σε οποιαδήποτε από τις αρχικές εξισώσεις και να συνδέσουμε το $k$ που βρήκαμε και στη συνέχεια να λύσουμε για $A_0$:

$$A_0 = y2e^{-kt2}$$

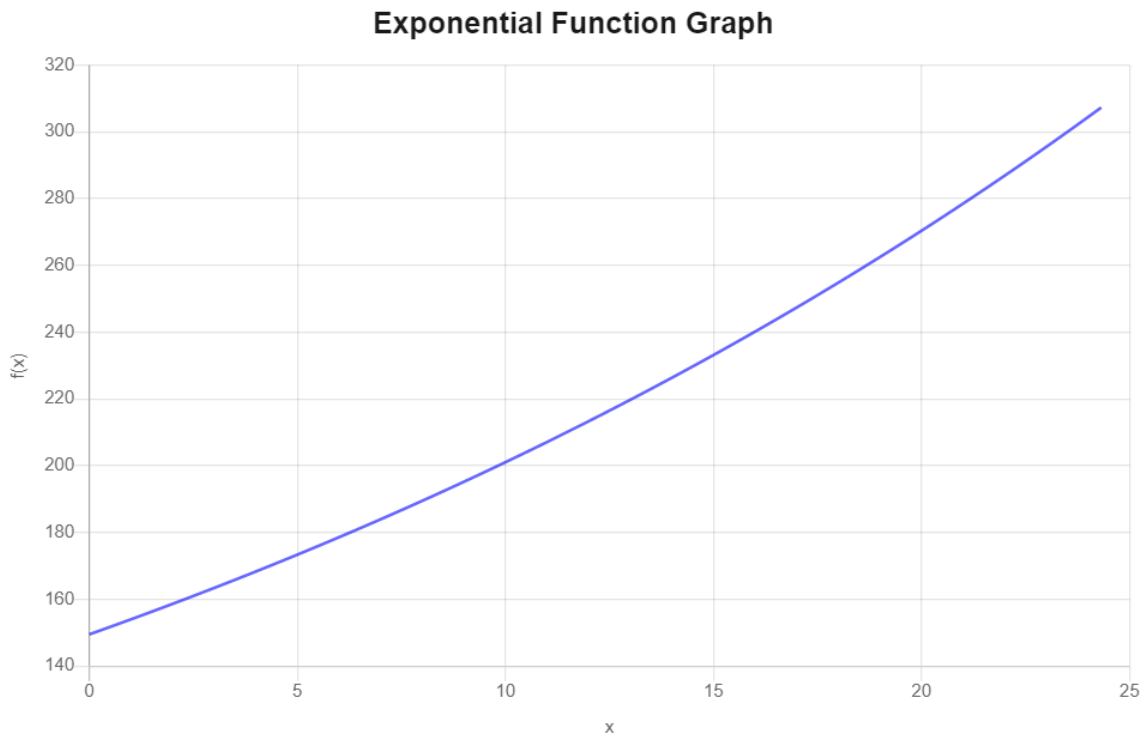
Βήμα 4ο: Τώρα που έχουμε έναν τύπο για $k$ και $A_0$ σε όρους $t_1$, $t_2$, $y_1$, $y_2$, πρέπει να συνδέσουμε τις δεδομένες τιμές για να βρούμε τις πραγματικές τιμές των $k$ και $A_0$. Από το παραπάνω πίνακα, παίρνω 2 σημεία συγκεκριμένα το (1, 154) και (50, 657) οπότε για $t_1 = 1$, $t_2 = 50$, $y_1 = 154$ και $y_2 = 657$ έχω:

$$k = \frac{1}{t1 - t2}\ln\left(\frac{y_1}{y_2}\right) = \frac{1}{1 - 50}\ln\left(\frac{154}{657}\right) = 0.0296$$

$$A_0 = y2e^{-kt2} = 657e^{-0.0296\times 50} = 149.5074$$

Επομένως, η εκθετική συνάρτηση που διέρχεται από τα σημεία (1, 154) και (50, 657) είναι:

$$f(t) = 149.5074e^{0.0296t}$$




Εικόνα 4.5.α: Γραφική απεικόνιση της συνάρτησης $f(t) = 149.5074e^{0.0296t}$

Επαναλαμβάνοντας την παραπάνω διαδικασία και εκτελώντας τον παραπάνω αλγόριθμο από το επίπεδο 51 έως και το επίπεδο 999, απ' όπου και αλλάζει ο παράγοντας αύξησης και γίνεται 1% προκύπτουν οι παρακάτω διακριτές τιμές:

| Επίπεδο | Πόντοι Εμπερίας |
|---|---|
| 51 | 664 |
| 52 | 670 |
| 53 | 677 |
| . | . |
| . | . |
| 100 | 1081 |
| 101 | 1092 |
| 102 | 1103 |
| 103 | 1114 |

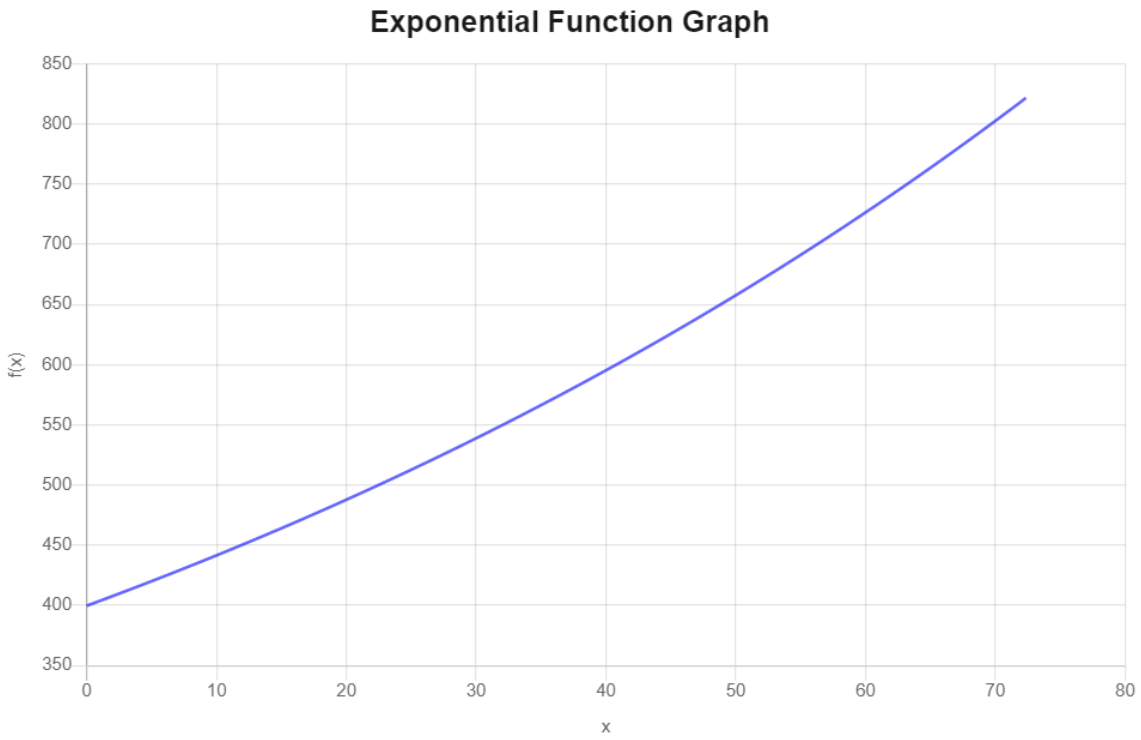
| | |
|---|---|
| 104 | 1125 |
| 105 | 1136 |
| . | . |
| . | . |
| 200 | 2925 |
| 201 | 2954 |
| 202 | 2984 |
| 203 | 3013 |
| 204 | 3043 |
| 205 | 3074 |
| . | . |
| . | . |
| 300 | 7912 |
| 300 | 7991 |
| 300 | 8071 |
| 300 | 8151 |
| . | . |
| . | . |
| . | . |
| 501 | 58464 |
| 501 | 59049 |
| 501 | 59639 |
| 501 | 60236 |
| . | . |
| . | . |
| . | . |
| 699 | 419299 |
| 700 | 423492 |
| 701 | 427727 |
| 702 | 432004 |
| . | . |
| . | . |
| 800 | 1145467 |
| 801 | 1156922 |

| | |
|---|---|
| . | . |
| . | . |
| 900 | 3098276 |
| 901 | 3129259 |
| . | . |
| . | . |
| 998 | 8215137 |
| 999 | 8297288 |

Πίνακας 4.5.β: Τα Επίπεδα και οι αντίστοιχοι πόντοι εμπειρίας τους, με ρυθμό αύξησης 1%.

Επομένως, η εκθετική συνάρτηση που διέρχεται από τα σημεία (51, 664) και (999, 8297288) είναι:

$$f(t) = 399.7351e^{0.01t}$$




Εικόνα 4.5.β: Γραφική απεικόνιση της συνάρτησης $f(t) = 399.7351e^{0.01t}$

## 4.6 Μηχανισμός H.G. ACK (Host - Guest – ACK)

Σχεδόν σε κάθε παιχνίδι πραγματικού χρόνου πολλαπλών παικτών (real time mutliplayer game), ο κάθε χρήστης έχει την δικιά του λίστα φίλων, ώστε να μπορεί να συνομιλεί, να προσκαλεί φίλους σε παιχνίδι ή να δέχεται προσκλήσεις για παιχνίδι. Επιπλέον μπορεί να παίζει και με τυχαίους παίκτες οι οποίοι μπορεί να μην βρίσκονται στην λίστα φίλων τους. Έτσι λοιπόν πρέπει να υπάρξει ένας μηχανισμός όπου 2 παίκτες θα μπορούν να συγχρονιστούν και να παίξουν μαζί. Ο μηχανισμός που υλοποιήθηκε ονομάστηκε «H.G. ACK» και είναι τα αρχικά των Host – Guest – ACK. Ο μηχανισμός αυτός περιγράφεται αναλυτικά παρακάτω:

Έστω ότι ο χρήστης Α επιθυμεί να ξεκινήσει παιχνίδι με τον χρήστη Β. Ο χρήστης Α έχει την ιδιότητα του «host», δηλαδή αυτός που αποστέλλει αίτημα παιχνιδιού και αντίστοιχα ο χρήστης Β είναι ο guest, αυτός που δέχεται πρόσκληση να παίξει. Η διαδικασία περιγράφεται παρακάτω ως εξής:

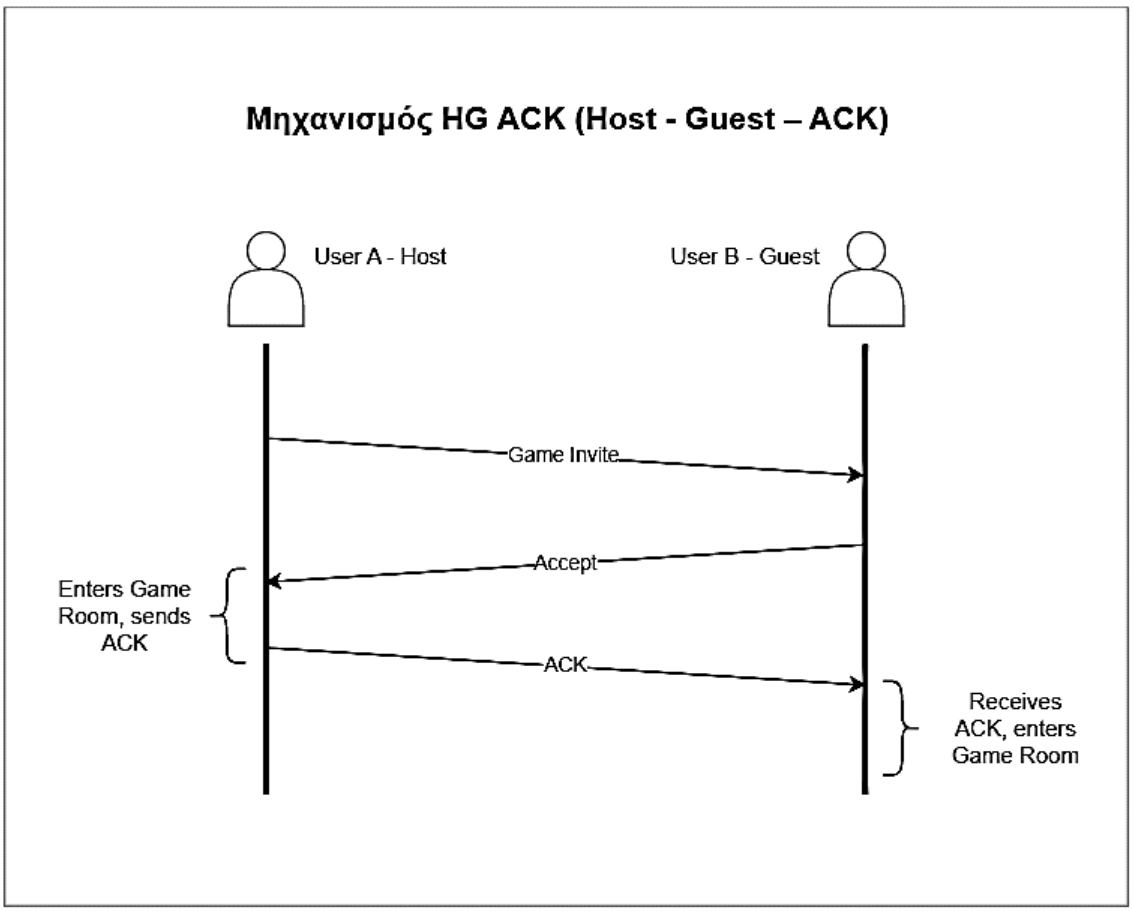
1. Ο χρήστης Α (Host) αποστέλλει ένα αίτημα παιχνιδιού (Invite) στον χρήστη Β (Guest) και περιμένει μια απάντηση επιβεβαίωσης (Accept).

2. Ο Χρήστης Β (Guest) καλείται είτε να απόδεχτεί την πρόσκληση ή να την απορρίψει έχοντας ένα χρονικό περιθώριο μερικών δευτερολέπτων (Time Gap) πριν το αίτημα παιχνιδιού λήξει και απορριφθεί αυτομάτως από το σύστημα.
3. Αν ο χρήστης Β (Guest) αποδεχτεί το αίτημα παιχνιδιού, ο χρήστης Α (Host) ενημερώνεται πως έχει γίνει αποδοχή του αιτήματος παιχνιδιού, με μια απάντηση «Accept» και αυτός με τη σειρά του αποστέλλει επιβεβαίωση «ACK»
4. Αν ο χρήστης Α (Host), μέσα σε αυτά τα δευτερόλεπτα (Time Gap) παραμένει σε κατάσταση διαθεσιμότητας και δεν έχει προηγηθεί κάποιο άλλο γεγονός όπως κλείσιμο της εφαρμογής, απώλεια σύνδεσης διαδικτύου, ή αποδοχή κάποιας άλλης πρόσκλησης παιχνιδιού από κάποιον άλλον χρήστη ενδεχόμενος κτλ, τότε στέλνει ένα μήνυμα επιβεβαίωσης «ACK» που σηματοδοτεί τον άρτιο συγχρονισμό τους και μεταβαίνει στο Λόμπι του παιχνιδιού (Game Lobby) ή αλλιώς δωμάτιο παιχνιδιού (Game Room)
5. Όμοια, αν ο χρήστης Β (Guest) είναι και εκείνος διαθέσιμος, λαμβάνει το μήνυμα επιβεβαίωσης (ACK) και μεταβαίνει και αυτός στο Λόμπι παιχνιδιού με τον χρήστη Α (Host)

Ένα αίτημα παιχνιδιού «Invite» περιλαμβάνει τις εξής διαχειριστικές πληροφορίες

1. Ποιος είναι ο Host του παιχνιδιού (ID χρήστη που έστειλε το αίτημα)
2. Ποιος είναι ο Guest του παιχνιδιού (ID χρήστη που θα στείλει απάντηση «Accept»)
3. Μοναδικό ID δωματίου παιχνιδιού (GUID)
4. Τα γράμματα στα οποία θα διαγωνιστούν οι παίκτες (Game Letters)
5. Το χρονική διάρκεια (Timer) κάθε γύρου.

Ο μηχανισμός «H.G. ACK» εξασφαλίζει πως και οι 2 χρήστες (Host και Guest) είτε θα μπουν ταυτόχρονα στο δωμάτιου παιχνιδιού, είτε καθόλου, αποτρέποντας έτσι την περίπτωση κακού συγχρονισμού το οποίο ενδεχομένως θα οδηγούσε κάποιον από τους 2 παίκτες στο δωμάτιο παιχνιδιού χωρίς την συνοδεία του άλλου.




Εικόνα 4.6 Μηχανισμός H.G. ACK (Host – Guest - ACK)

## 4.7 Μηχανισμός R.R. Queue (Random Room Queue)

Η κύρια φιλοσοφία των παιχνιδιών πραγματικού χρόνου πολλαπλών παικτών είναι πως οι χρήστες τους μπορούν να συνδέονται και να αλληλοεπιδρούν. Εκτός όμως από την αλληλεπίδραση αυτή με χρήστες που βρίσκονται ήδη στη λίστα φίλων τους, είναι σημαντικό, να τους δίνεται η δυνατότητα να αλληλοεπιδράσουν και με άλλους, τυχαίους χρήστες αν το επιθυμήσουν. Οι λόγοι είναι πολλοί. Τα παιχνίδια αυτά βασίζονται εν μέρει στην ιδέα των κοινωνικών δικτυών, δηλαδή στην αλληλεπίδραση και στην δημιουργία κοινωνικών σχέσεων μεταξύ των ανθρώπων, που συνήθως αποτελούν ενεργά μέλη ενός κοινωνικού δικτύου, με κοινά ενδιαφέροντα ή δραστηριότητες. Όσο περισσότερους ανθρώπους περιλαμβάνει ένα κοινωνικό δίκτυο, τόσο περισσότερο είναι το ενδιαφέρον των ανθρώπων για αυτό. Αντίστοιχα, σε ένα παιχνίδι πραγματικού χρόνου, το κοινό ενδιαφέρον των χρηστών του, είναι το ίδιο το παιχνίδι, και σκοπός του, είναι η συνεχής παροχή παικτών, τυχαίων ή μη, (λίστα φίλων) ώστε

να υπάρχει όλο και περισσότερη επαφή των χρηστών με το παιχνίδι, γνωρίζοντας και αντιμετωπίζοντας κάθε φορά νέους αντίπαλους. Επομένως, είναι αναγκαίος ένας μηχανισμός όπου ένας χρήστης, θα μπορεί να παίξει ανά πάσα στιγμή με έναν τυχαίο αντίπαλο. Ο μηχανισμός που υλοποιήθηκε, ονομάστηκε «R.R. Queue» (Random Room Queue) και όπως το όνομα αναφέρει, βασίζεται σε μια δομή δεδομένων με τη μορφή παρατεταμένης συλλογής, γνωστή και ως ουράς. Ο μηχανισμός αυτός χωρίζει τους ενδιαφερόμενους χρήστες σε Hosts και Guests, όπου αντίστοιχα περιλαμβάνει και 2 ουρές (Host Queue και Guest Queue). Στην συνέχεια παρουσιάζονται τα δομικά συστατικά του και αναλύεται ο μηχανισμός.

Ο Χρήστης Α επιθυμεί να παίξει με τυχαίο αντίπαλο.

Το σύστημα ελέγχει:

1. Αν οι 2 ουρές είναι άδειες ή οι 2 ουρές είναι γεμάτες. Στην περίπτωση αυτή του δίνεται ένας ρόλος τυχαία από το σύστημα. Host (id = 1) ή Guest (id = 2) και εισάγεται στην αντίστοιχη ουρά.
2. Αν μια από τις 2 ουρές είναι άδεια. Στην περίπτωση αυτή ο χρήστης εισάγεται στην άδεια ουρά και του δίνεται ο αντίστοιχος ρόλος της. (πχ αν η Hosts Queue έιναι άδεια τότε ο χρήστης εισάγεται σε αυτήν με ρόλο Host)

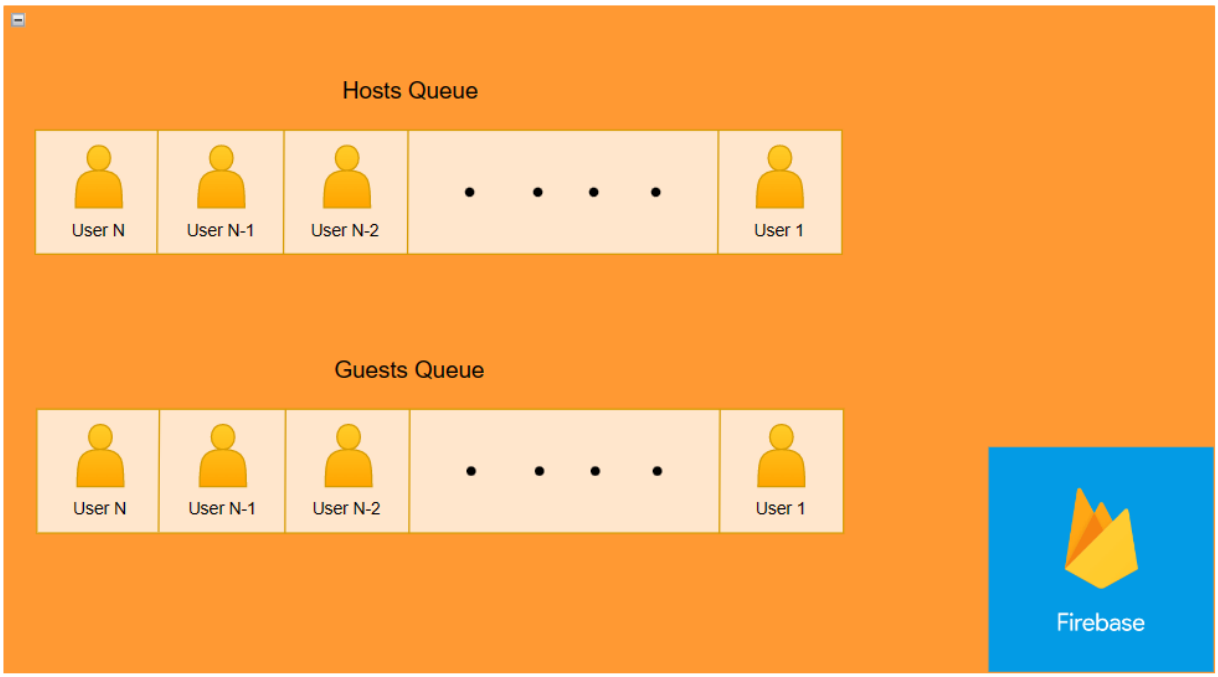
Εφόσον ο client (χρήστης) αποκτήσει ρόλο (Host ή Guest) και εισαχθεί στην αντίστοιχη ουρά, το σύστημα θέτει ένα χρονικό όριο (Time Gap) στο οποίο αν δεν βρεθεί αντίπαλος για τον χρήστη, τότε ο χρήστης θα αντιμετωπίσει έναν εικονικό αντίπαλο, το μηχανισμό bot που διαθέτει το παιχνίδι και περιγράφεται αναλυτικά παρακάτω.

Όπως έχει αναφερθεί, το σύστημα του παιχνιδιού χρησιμοποιεί μια βάση πραγματικού χρόνου και συγκεκριμένα το Firebase. Ολες οι αλλαγές που πραγματοποιούνται σε αυτήν (εισαγωγή, διαγραφή, ενημέρωση) έχουν ως αποτέλεσμα την άμεση ειδοποίηση και ενημέρωση των clients που είναι συνδεδεμένοι και εγγεγραμμένοι στα αντίστοιχα γεγονότα (Events) . Επομένως ένας client (χρήστης) που βρίσκεται είτε στην ουρά των Hosts είτε στην ουρά των Guests, ενημερώνεται άμεσα για νέες εισαγωγές ή διαγραφές στις 2 ουρές. Επομένως, η εύρεση, και το «ζευγάρωμα», 2 clients, host και guest γίνεται από την πλευρά των ίδιων των clients και όχι από πλευράς server δηλαδή μέσω κάποιου cloud function του Firebase.

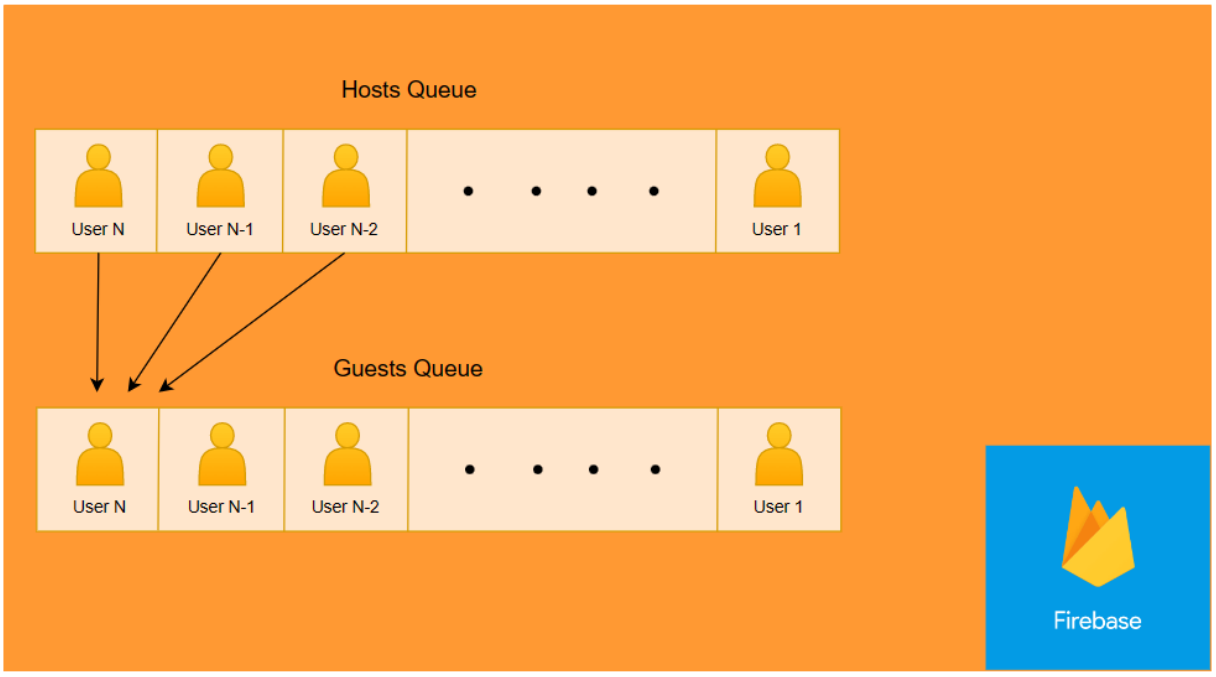
Ο μηχανισμός του ζευγαρώματος 2 clients (Host και Guest) βασίζεται στον μηχανισμό «H.G. ACK» που περιγράφηκε παραπάνω. Η λογική είναι τύπου Master – Slave, δηλαδή, ένας Host αναζητεί πάντα έναν Guest για να ζευγαρώσει.

1. Ένας Host στέλνει ένα αίτημα παιχνιδιού «Invite From Random» στον Guest που βρίσκεται στην αρχή της ουράς (Head). Ενδεχομένως, λόγω των γρήγορων και ταυτόχρονων εισαγωγών στις ουρές, πολλοί Hosts ενδέχεται να στείλουν αίτημα στον ίδιο Guest.
2. Ο Guest που βρίσκεται στην αρχή της ουράς, στέλνει μια απάντηση «Accept» στον Host. Αν όπως αναφέρθηκε, δεχτεί ταυτόχρονα, αιτήματα από πολλούς Hosts, τότε στέλνει απάντηση «Accept» στον Host εκείνον που έστειλε πρώτος το αίτημα «Invite From Random».
3. Όμοια με τον μηχανισμό «H.G. ACK», ο Host στέλνει μήνυμα επιβεβαίωσης ACK στον Guest με τον οποίο ζευγάρωσε, και θα μπουν μαζί στο δωμάτιο παιχνιδιού.
4. Τόσο ο Guest, όσο και ο Host βγαίνουν από την ουρά στην οποία είχαν εισαχθεί και οι υπόλοιποι clients ενημερώνονται για τις αλλαγές αυτές στις 2 αντίστοιχες ουρές και όμοια η διαδικασία συνεχίζεται για τους εναπομείναντες.

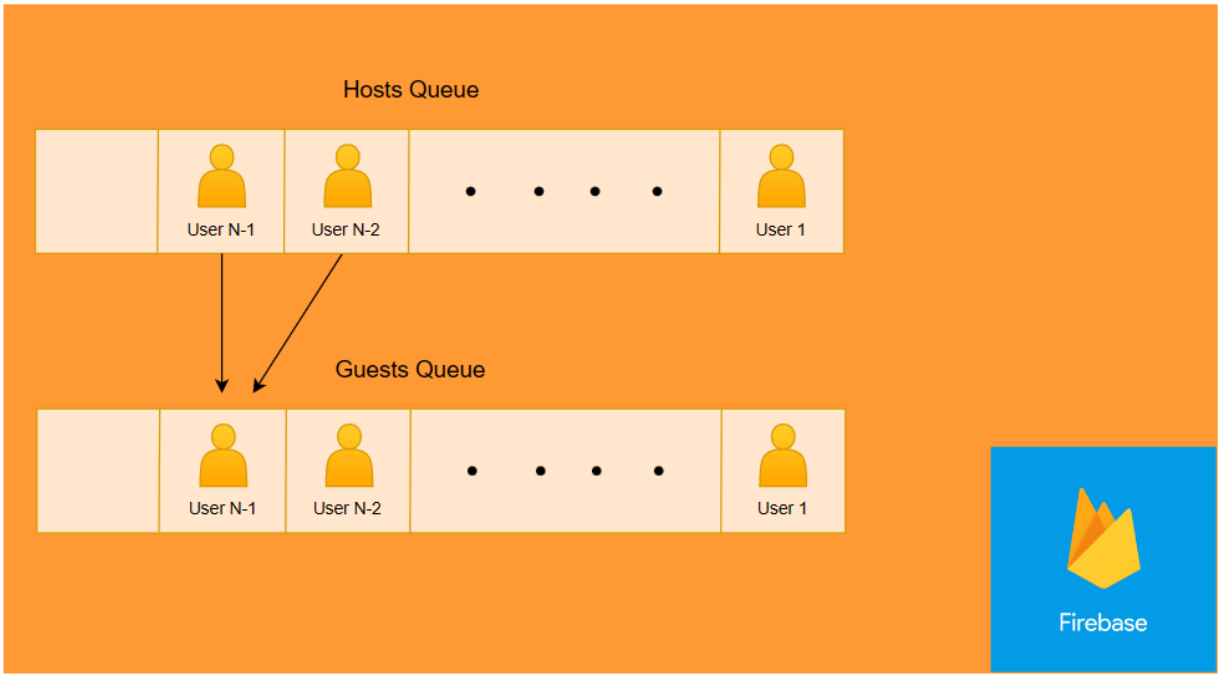
Ένα αίτημα «Invite From Random» περιλαμβάνει τι ίδιες διαχειριστικές πληροφορίες με ένα αίτημα Invite του μηχανισμού «H.G. ACK». Ο μηχανισμός «R.R. Queue» δεν είναι βέλτιστος αλλά παρέχει αξιοπιστία, εξασφαλίζει συγχρονισμό, και τέλος, εξασφαλίζει πως όλοι οι clients θα βρουν αντίπαλο είτε εικονικό είτε πραγματικό. Στην παρούσα διπλωματική, το παρόν παιχνίδι που υλοποιήθηκε υποστηρίζει 3 γλώσσες. Τα ελληνικά, τα αγγλικά και τα ισπανικά. Ο σκοπός του παιχνιδιού είναι ο διαγωνισμός των παικτών στην δημιουργία λέξεων με τυχαία γράμματα. Συνεπώς ένας χρήστης που έχει επιλέξει την ισπανική γλώσσα και επιθυμεί την εύρεση αντιπάλου, δεν θα πρέπει να τοποθετηθεί στην ίδια ουρά με έναν χρήστη που έχει επιλέξει ως γλώσσα τα ελληνικά. Για τον λόγο αυτό, το σύστημα διαθέτει διαφορετικές ουρές για τις 3 διαφορετικές γλώσσες που υποστηρίζει. Επιπλέον γίνεται εξατομίκευση ως προς το επίπεδο των χρηστών. Συγκεκριμένα δημιουργούνται ουρές τόσο για τους Hosts όσο και για τους Guests με βάση το επίπεδο τους.




Εικόνα 4.7.α: Οι ουρές των Hosts και των Quests, σε μια τυχαία χρονική στιγμή t.




Εικόνα 4.7.β: Ένας Host στέλνει πάντα αίτημα παιχνιδιού σε χρήστη που βρίσκεται στην αρχή της ουράς των Guests.




Εικόνα 4.7.γ: Εφόσον υπήρξε «ζευγάρωμα» 2 χρηστών, η διαδικασία συνεχίζεται.

## 4.8 Μηχανισμός Παιχνιδιού

Όπως αναλύθηκε προηγουμένως, τόσο ο μηχανισμός «H.G. ACK» όσο και ο μηχανισμός «R.R. Queue» επιτυγχάνουν συγχρονισμό αναμεσά σε 2 clinets (χρήστες) τοποθετώντας τους ταυτόχρονα στο ίδιο δωμάτιο παιχνιδιού, έχοντας μαζί τους τις απαραίτητες διαχειριστικές πληροφορίες που είναι χρήσιμες για την ομαλή διεξαγωγή του «αγώνα». Οι πληροφορίες αυτές όπως έχουν αναφερθεί είναι οι εξής:

1. Id του Host χρήστη.
2. Id του Guest χρήστη.
3. Id δωματίου παιχνιδιού.
4. Τα γράμματα.
5. Η διάρκεια του γύρου.

Έτσι, και οι 2 χρήστες (clients) που βρίσκονται στο ίδιο δωμάτιο μπορούν να «συνομιλούν» ανταλλάζοντας όλα εκείνα τα απαραίτητα δεδομένα κατά την διάρκεια του παιχνιδιού, όπως: οι λέξεις που σχηματίστηκαν, τα νέα γράμματα για τον νέο γύρο, αν αποχώρησε ένας από τους 2 χρήστες, αν επιθυμούν να παίξουν ξανά και φυσικά να συνομιλήσουν, μέσω του chat που παρέχει το δωμάτιο.

Ο μηχανισμός, βασίζεται στην ανταλλαγή δεδομένων και στον συγχρονισμό του χρονομέτρου πριν από κάθε γύρο. Βέβαια το «πάνω χέρι» στον συντονισμό έχει η σύνδεση του εκάστοτε χρήστη (client) διότι επηρεάζεται από τον χρόνο αποστολής και λήψης δεδομένων. Ο μηχανισμός θέτει σημεία ελέγχου (Check-Points) ώστε ακόμα και αν παρατηρηθεί μια μικρή διαφορά δευτερολέπτων (1 έως 2 το πολύ) σε έναν γύρο, τότε, να γίνει προσπάθεια συγχρονισμού στον επόμενο. Έτσι κατά την διάρκεια ενός «αγώνα» (παιχνιδιού) θα πραγματοποιηθούν 3 σημεία ελέγχου - συγχρονισμοί. Ο συγχρονισμός για τον πρώτο γύρο διαφέρει από τους υπόλοιπους δύο (δεύτερο και τρίτο γύρο). Στον πρώτο συγχρονισμό, ο οποίος είναι και ταυτόχρονα το «εναρκτήριο λάκτισμα» για την εκκίνηση του παιχνιδιού, οι χρήστες πρέπει να πατήσουν το κουμπί «Έτοιμος». Αντίθετα, αν ένας από τους δυο χρήστες ή και οι δύο, δεν επιθυμούν την εκκίνηση του αγώνα μπορούν να πατήσουν το κουμπί «Έξοδος». Αναλυτικά η διαδικασία παρακάτω:

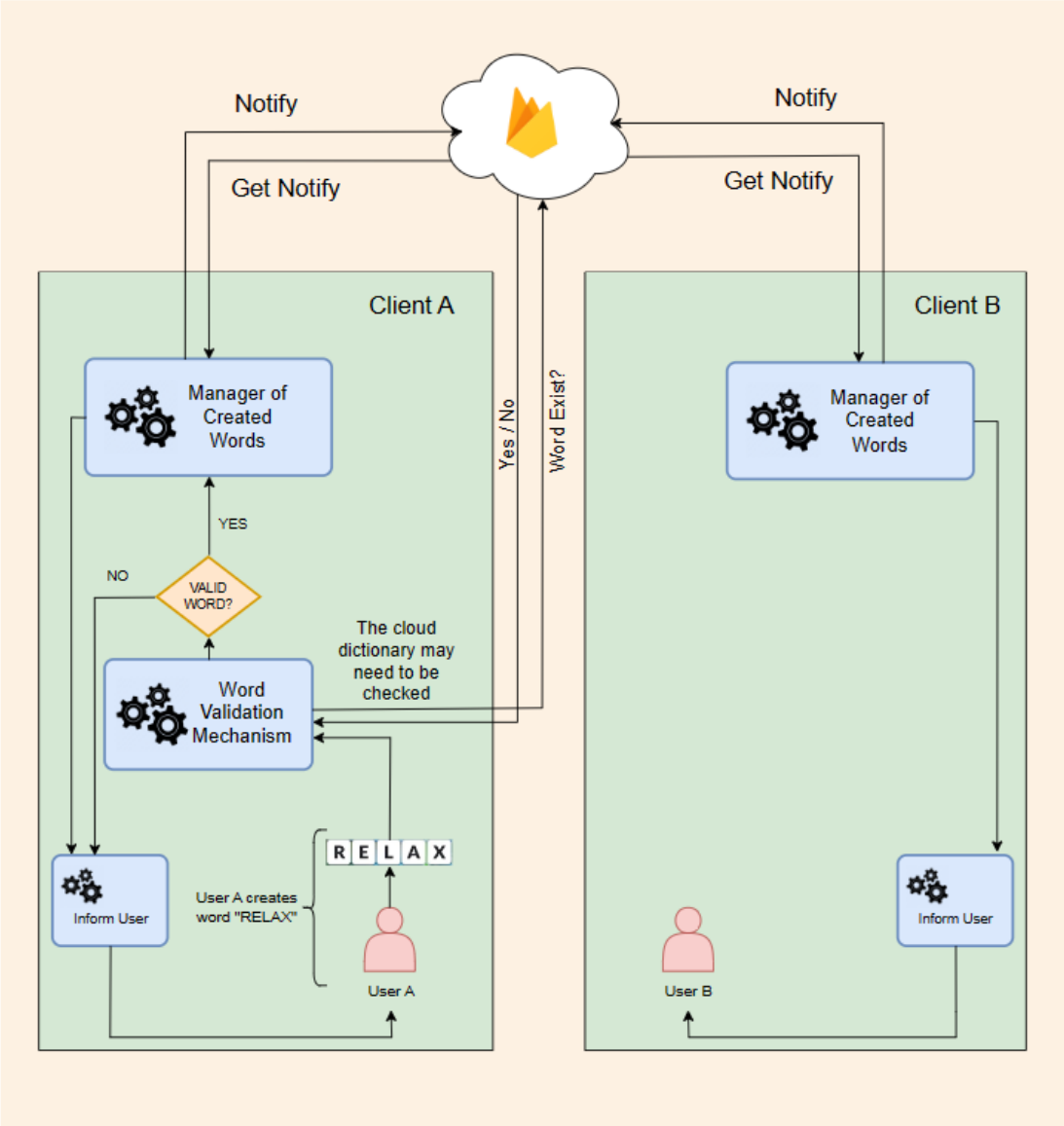
1. Ο χρήστης Α πατάει το κουμπί «Έτοιμος» και στέλνει ένα αίτημα «ReadyToPlay» στον χρήστη Α και περιμένει να λάβει το ίδιο αίτημα από το χρήστη Β.
2. Ο χρήστης Β δέχεται το αίτημα «ReadyToPlay» από τον χρήστη Α και με την σειρά του μπορεί να πατήσει το κουμπί «Έτοιμος» για την εκκίνηση του «αγώνα» ή του κουμπιού «Έξοδος». Ο χρήστης β πατάει το κουμπί «Έτοιμος»
3. Εφόσον και οι 2 χρήστες έχουν στείλει και έχουν λάβει τα αιτήματα «ReadyToPlay», το παιχνίδι ξεκινάει, εμφανίζονται τα γράμματα του γύρου, το χρονόμετρο του γύρου καθώς και άλλες επιλογές για τον γύρο.

Φυσικά και οι δύο χρήστες ενδέχεται να πατήσουν ταυτόχρονα το κουμπί «Έτοιμος» όπου και σε αυτήν την περίπτωση το αποτέλεσμα είναι το ίδιο. Μόλις ο πρώτος γύρος ολοκληρωθεί, εμφανίζεται το μήνυμα «Ο νέος γύρος ξεκινά σύντομα» Σε αυτό το σημείο υπάρχει ένα χρονικό κενό (Time Gap) στο οποίο και οι δύο χρήστες πρέπει να στείλουν και να λάβουν ένα αίτημα «ReadyNextRound». Με αυτόν τον τρόπο, γίνεται προσπάθεια συγχρονισμού εκ νέου των δυο χρηστών για την διόρθωση ή μη τυχόν διαφορών που παρατηρήθηκαν στο χρονόμετρο των δυο χρηστών. Επιπλέον, σε αυτό το χρονικό σημείο, ο χρήστης εκείνος που έχει ρόλο Host, δημιουργεί νέα τυχαία γράμματα για τον επερχόμενο γύρο και τα στέλνει στον Guest μέσω του αιτήματος «newRoundInfo». Μόλις και οι δύο χρήστες στείλουν και λάβουν το αίτημα «ReadyNextRound» ξεκινάει η αντίστροφη μέτρηση για την εκκίνηση του νέου γύρου. Τέλος, αν ένας από τους χρήστες αποχωρήσει κατά την διάρκεια του αγώνα, στέλνει ένα αίτημα «Exit» στον άλλον χρήστη ώστε να τον ειδοποιήσει για την αποχώρηση του.

Ξέχωρα όμως από τα μηνύματα συγχρονισμού, κατά την διάρκεια των γυρών στέλνονται και μηνύματα ειδοποίησης. Τέτοια μηνύματα αφορούν τον σχηματισμό των λέξεων. Για παράδειγμα αν ο χρήστης Α σχηματίσει την λέξη «ΓΑΤΑ» και εφόσον δεν έχει σχηματιστεί από τον Χρήστη Β, ο χρήστης Β ενημερώνεται.

Η διαδικασία αναλυτικά:

Ο χρήστης Α σχηματίζει μια λέξη. Αν η λέξη δεν έχει σχηματιστεί πάλι είτε από τον ίδιο είτε από τον αντίπαλο τότε ελέγχεται αν είναι έγκυρη, δηλαδή αν υπάρχει σε ένα από τα δυο λεξικά του παιχνιδιού. Αν η λέξη είναι έγκυρη τότε ο χρήστης Α στέλνει ένα αίτημα «GameInfo» στον Χρήστη Β με το GUID του παιχνιδιού και πληροφορίες όπως την λέξη που σχηματίστηκε και αν υπήρξε χρήση της ικανότητας «x2 Boost» ώστε να διπλασιαστούν οι πόντοι της λέξης.




Εικόνα 4.8 Ο μηχανισμός του παιχνιδιού όταν δημιουργείται μια λέξη.

## 4.9 Μηχανισμός Chat

Όπως αναφέρθηκε και στο κεφάλαιο 3, τα περισσότερα παιχνίδια πραγματικού χρόνου και πολλαπλών χρηστών (multiplayer), διαθέτουν μηχανισμούς συνομιλίας (chat) ώστε οι χρήστες να μπορούν να συνομιλούν μεταξύ τους με σκοπό την ανταλλαγή μηνυμάτων σχετικά με το παιχνίδι, να κοινωνικοποιούνται αλλά και για άλλους λόγους οι οποίοι αναλυθήκαν στο προηγούμενο κεφάλαιο. Στην παρούσα διπλωματική εργασία, όπως έχει αναφερθεί, γίνεται υλοποίηση συνομιλίας χρηστών εντός του παιχνιδιού.

Ο μηχανισμός της συνομιλίας, υλοποιείται με την χρήση του Firebase, που όπως έχει αναφερθεί και σε προηγούμενα κεφάλαια, αποτελεί μια βάση δεδομένων πραγματικού χρόνου, όπου όλες οι αλλαγές, όπως, προσθήκες, διάγραφες, ενημερώσεις των δεδομένων είναι ορατές, ταυτόχρονα στους πελάτες (clients) που είναι συνδεδεμένοι σε αυτήν. Επειδή η ανταλλαγή μηνυμάτων σε μια συνομιλία 2 ή και παραπάνω χρηστών απαιτεί την άμεση αποστολή και εμφάνιση των αποσταλμένων μνημάτων στον παραλήπτη, το Firebase εξυπηρετεί και με το παραπάνω αυτή την ανάγκη.

Το Firebase είναι μια βάση δεδομένων NoSQL, όπου τα δεδομένα αποθηκεύονται ως αντικείμενα JSON, και συγκεκριμένα ως δέντρα JSON. Έτσι όλα τα δεδομένα των συνομιλίων, θα πρέπει να βρίσκονται στον γονικό κατάλογο «\Chats». Το επόμενο βήμα είναι να ορισθεί ένας θυγατρικός κατάλογος, κάτω από τον γονικό όπου οι 2 χρήστες της συνομιλίας θα υπακούν σε αυτόν, δηλαδή θα στέλνουν και θα λαμβάνουν τα μηνύματα της συνομιλίας μαζί με τις διαχειριστικές και συνοδεύτηκες πληροφορίες αυτών, στον θυγατρικό κατάλογο.

Ο θυγατρικός κατάλογος δημιουργείτε από την διαδικασία αποκλειστικής διάζευξης (xor) του κάθε χαρακτήρα της συμβολοσειράς του Firebase ID του ενός χρήστη, με τον αντίστοιχο χαρακτήρα της συμβολοσειράς του Firebase ID του άλλου χρήστη. Έτσι για παράδειγμα αν έχουμε τον χρήστη A με Firebase ID: *5QvhwwXUhrcQ5XeYHPvTnOWvcdC3*, και τον χρήστη B με Firebase ID: *C7oqJjK7P7VbNPOa0RZ5IN9zDkc2*, τότε η διαδικασία αποκλειστικής διάζευξης, λαμβάνει αρχικά τον 1ο χαρακτήρα του 1ου id, δηλαδή τον χαρακτήρα *5* και τον αντίστοιχο 1ο χαρακτήρα του 2ου ID, δηλαδή τον χαρακτήρα *C* και παράγει έναν αριθμό που είναι το αποτέλεσμα της αποκλειστικής διάζευξης των 2 αυτών. Έτσι το αποτέλεσμα της αποκλειστικής διάζευξης των 2 αναφερθέντων χαρακτήρων, μας δίνει το αποτέλεσμα:

$$5 \oplus C = 118$$

Επαναλαμβάνοντας την διαδικασία για όλους τους εναπομείναντες χαρακτήρες των firebase id των 2 χρηστών, λαμβάνουμε το αποτέλεσμα:

*118102252561291998566953511238425612024497391110123915321.*

Έτσι τα δεδομένα που θα ανταλλάζουν οι 2 χρήστες θα βρίσκονται στην τοποθεσία:

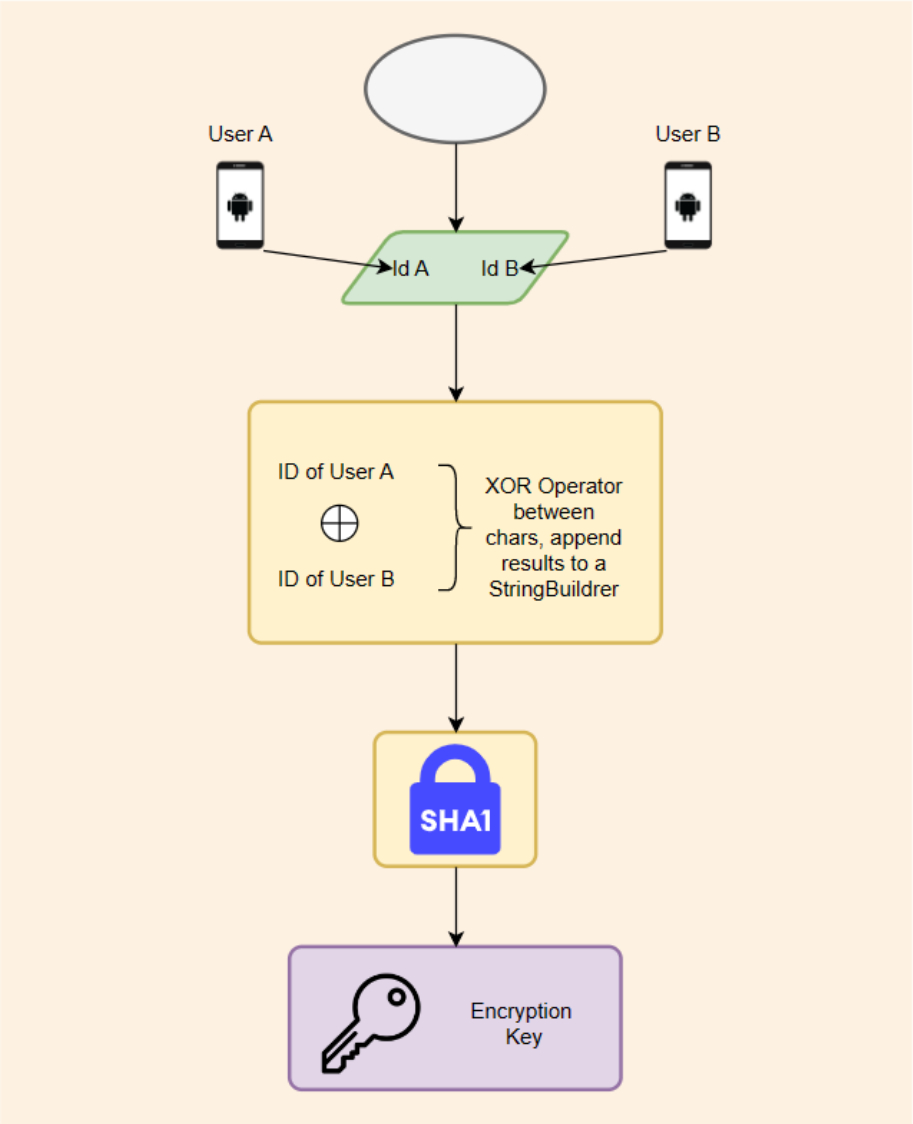
Chats\118102252561291998566953511238425612024497391110123915321.

Επιπλέον, το μήνυμα που στέλνεται από έναν χρήστη σε έναν άλλον, κατά την διάρκεια της συνομιλίας τους, κρυπτογραφείται, με κλειδί που προκύπτει από την παραπάνω αναφερθείσα διαδικασία, αυτής της αποκλειστικής διάζευξης και την προσθήκη επιπλέον μιας συνάρτησης κατακερματισμού, την SHA-1 η οποία δέχεται ως είσοδο το αποτέλεσμα από την αποκλειστική διάζευξη και παράγει μια τιμή κατακερματισμού, δηλαδή το τελικό κλειδί κρυπτογράφησης και αποκρυπτογράφησης του αλγορίθμου AES.

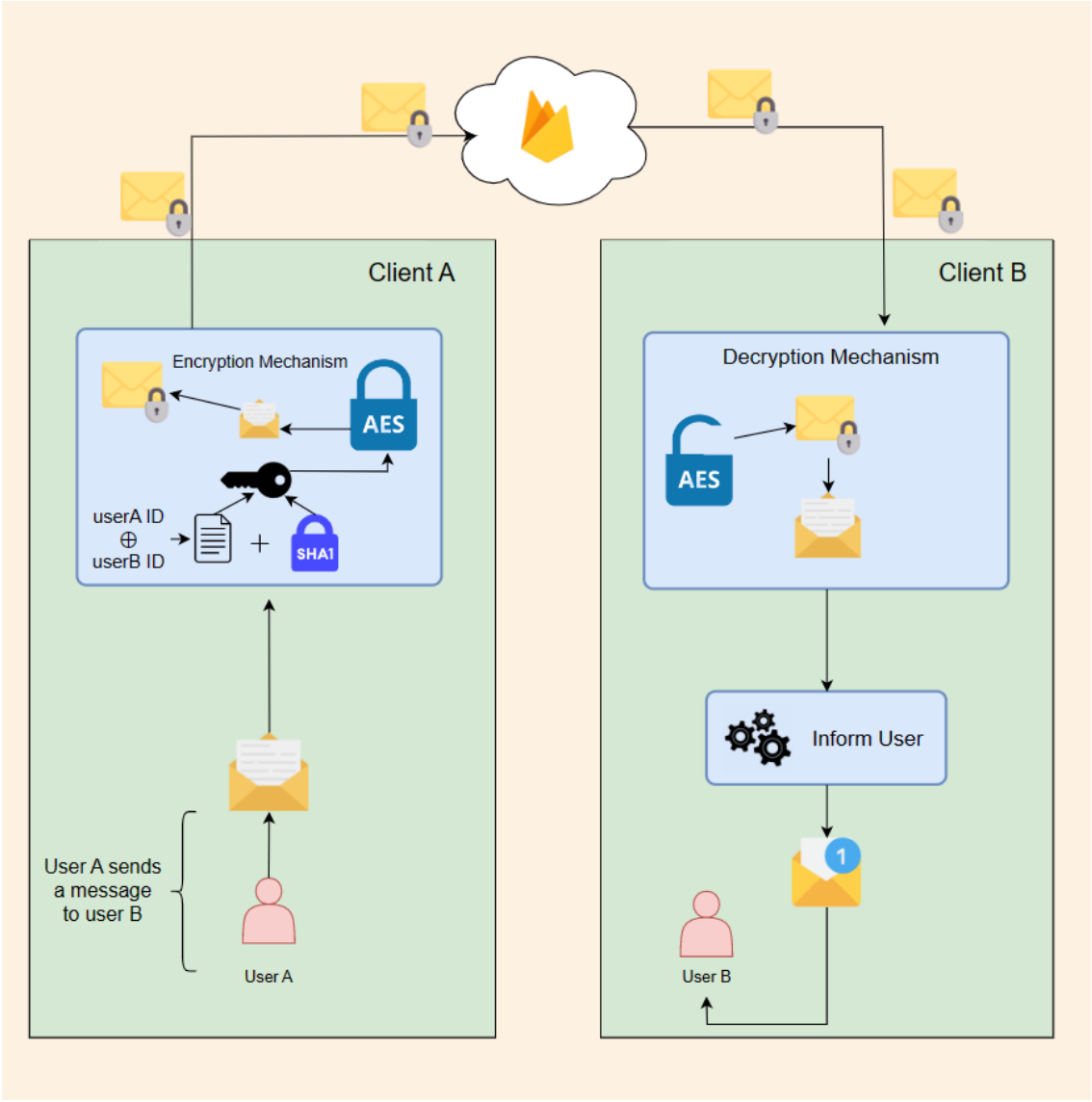
Οι συνοδευτικές διαχειριστικές πληροφορίες ενός μηνύματος είναι εξής:

- **message**: Όπου αποτελεί το μήνυμα που πρέπει να παραδοθεί στον παραλήπτη.
- **receiver**: Ο παραλήπτης του μηνύματος, ως παραλήπτης αναγράφεται το Firebase ID του χρήστη
- **sender**: Ο αποστολέας του μηνύματος, ως αποστολέας αναγράφεται το Firebase id του χρήστη
- **seen**: Λαμβάνει την τιμή 1 ή 0, ώστε να εμφανίζονται τα κατάλληλα notifications στον παραλήπτη
- **timestamp**: Ο χρόνος αποστολής του μηνύματος εκφρασμένος σε χιλιοστά του δευτερολέπτου (milliseconds)

Από τις παραπάνω συνοδευτικές και διαχειριστικές πληροφορίες, το μήνυμα, ο αποστολέας και ο παραλήπτης αποθηκεύονται κρυπτογραφημένα, αλλάζοντας μορφή στα δεδομένα που αναπαριστούν, κάνοντας την συνομιλία πιο ασφαλή, ακολουθώντας την προστασία των προσωπικών δεδομένων των χρηστών, ακόμα και αν η βάση δεδομένων «πέσει» σε λάθος χεριά.




Εικόνα 4.9.α Η διαδικασία δημιουργίας κλειδιού κρυπτογράφησης για τον αλγόριθμο AES.



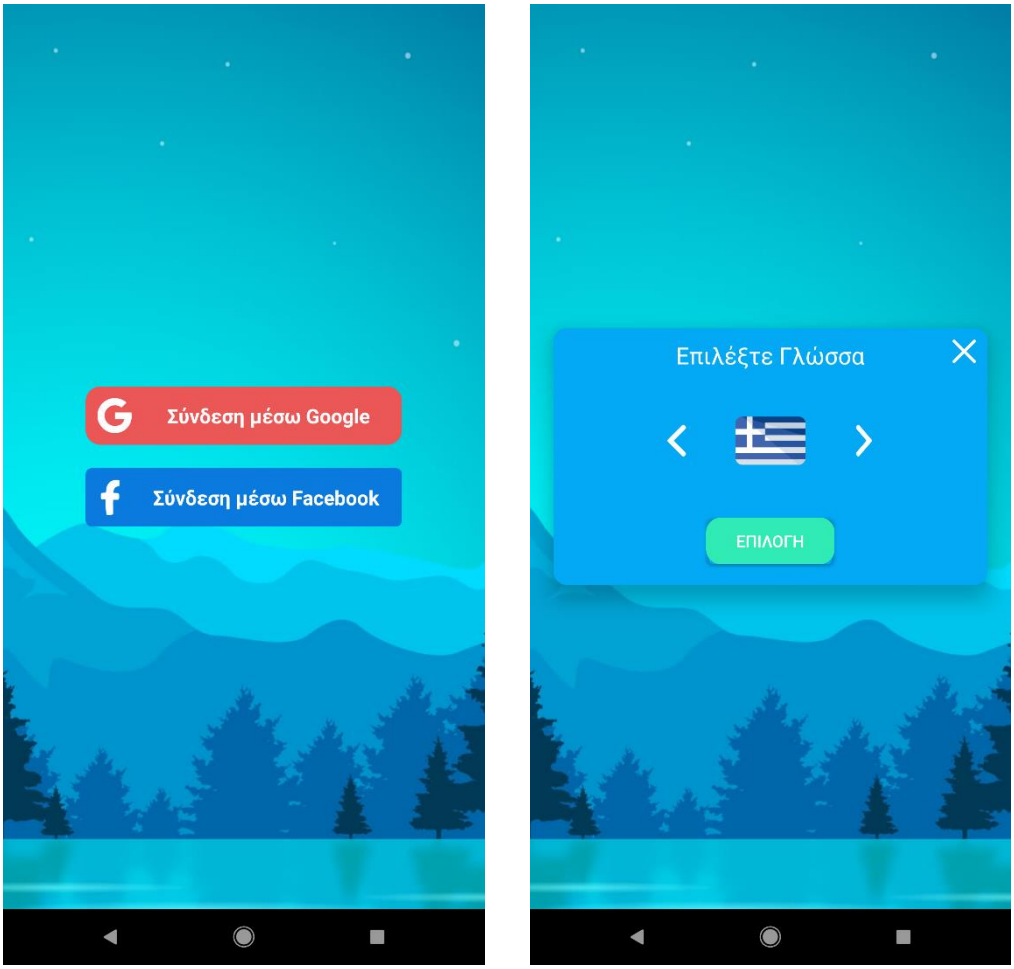

Εικόνα 4.9.β Η διαδικασία της αποστολής μηνύματος από τον χρήστη Α στον χρήστη Β.


Εικόνα 4.9: Η Δομή ενός μηνύματος που αποθηκεύεται στην βάση με κρυπτογραφημένα τον αποστολέα, τον παραλήπτη καθώς και το ίδιο το μήνυμα.

# Κεφάλαιο 5 - Παρουσίαση Λογισμικού

## 5.1 Σύνδεση χρήστη.

Κατά την πρώτη εκκίνηση της εφαρμογής σε μια συσκευή, ο χρήστης υποχρεωτικά θα πρέπει να συνδεθεί σε αυτήν, μέσω των διαθέσιμων παροχών αυθεντικοποίησης, Facebook και Google. Έτσι το σύστημα ελέγχει αν πρόκειται για νέο χρήστη ή υπάρχον. Στην περίπτωση του νέου χρήστη, ο χρήστης πρέπει να επιλέξει γλώσσα παιχνιδιού. Το σύστημα στην συνέχεια προετοιμάζει και αποθηκεύει τα απαραίτητα δεδομένα, όπως όνομα χρήστη, Firebase ID του χρήστη, URL εικόνας προφίλ χρήστη καθώς και άλλα πολλά. Στην αντίθετη περίπτωση, όπου ο χρήστης είναι ήδη εγγεγραμμένος, το σύστημα αντλεί τα δεδομένα χρήστη και τον οδηγεί στο Dashboard του παιχνιδιού.




Εικόνα 5.1.1: Οθόνη σύνδεσης χρήστη, μέσω Google ή Facebook και επιλογή γλώσσας.


5.1.2: Στοιχεία νέου χρήστη.

## 5.2 Αρχική Οθόνη και περιήγηση στην εφαρμογή.

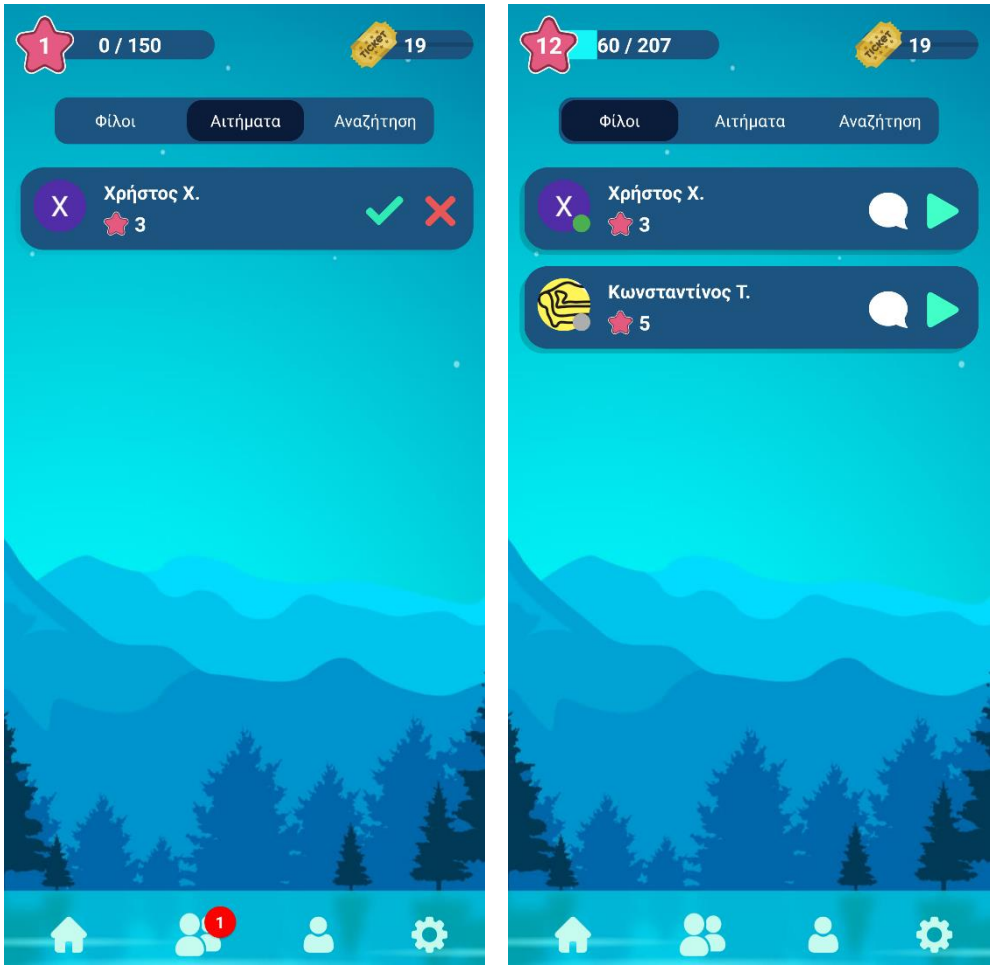
### 5.2.1 Αρχική Οθόνη

Αφού ο χρήστης συνδεθεί στην εφαρμογή, οδηγείται στην οθόνη Dashboard. Στην οθόνη αυτή, ο χρήστης μπορεί να επιλέξει να παίξει με τυχαίο αντίπαλο, να λάβει το καθημερινό του δώρο, καθώς και να δει τους κορυφαίους 20 παίκτες, (TOP 20). Επιπλέον, στην οθόνη αυτή, προβάλλονται, όλες οι εισερχόμενες προσκλήσεις για παιχνίδι, που μπορεί να λάβει ο χρήστης από τους φίλους που διαθέτει.

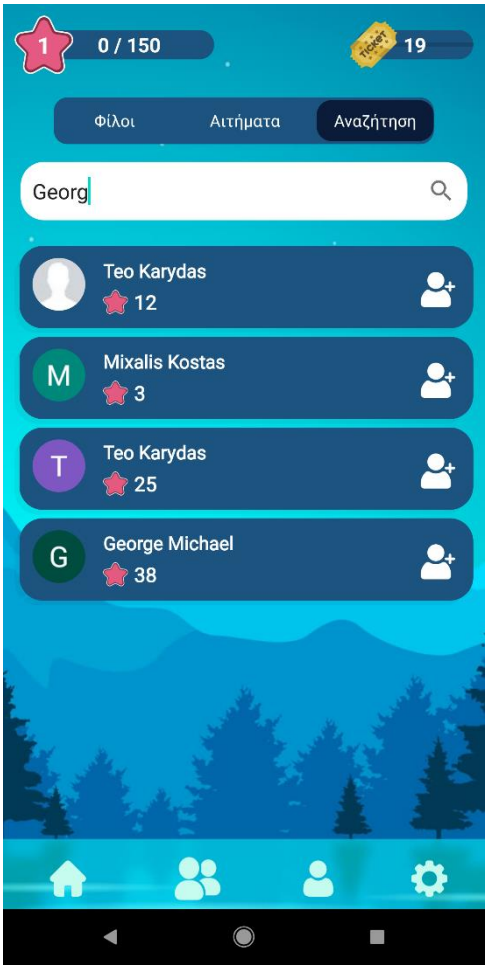

Εικόνα 5.1.3: Αρχική οθόνη εφαρμογής.

### 5.2.2 Οθόνη φίλων

Στην οθόνη αυτή, ο χρήστης μπορεί να δει και να διαχειριστεί την λίστα φίλων του, να δει εισερχόμενα αιτήματα φίλιας του, και να αναζητήσει φίλους νέους φίλους, πληκτρολογώντας είτε το όνομα τους, είτε το επίθετο τους, είτε ολόκληρο το ονοματεπώνυμο τους. Επιπλέον μπορεί να στείλει αιτήματα παιχνιδιού σε χρήστη/ες από την λίστα φίλων του, καθώς και να συνομιλήσει με αυτούς.



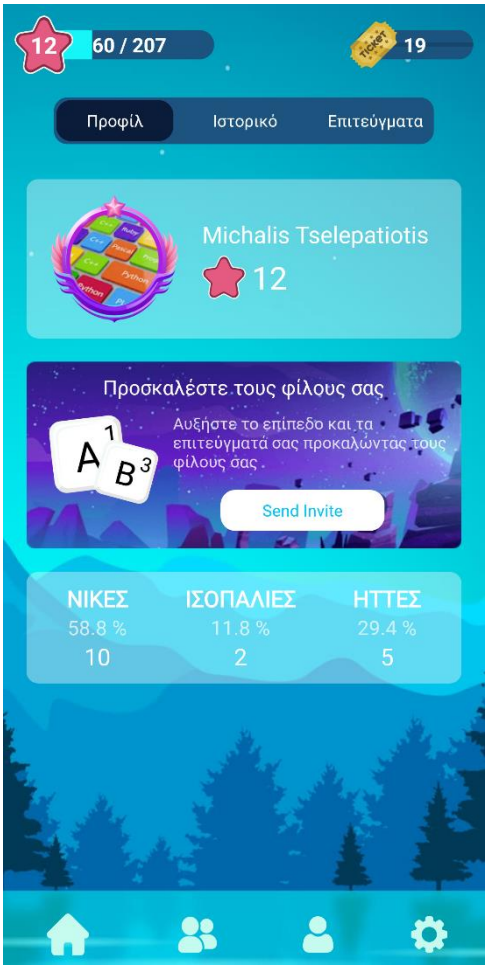

Εικόνα 5.2.2.α: Ειδοποίηση εισερχομένου αιτήματος φίλιας και η λίστα φίλων, μετά την αποδοχή του αιτήματος.



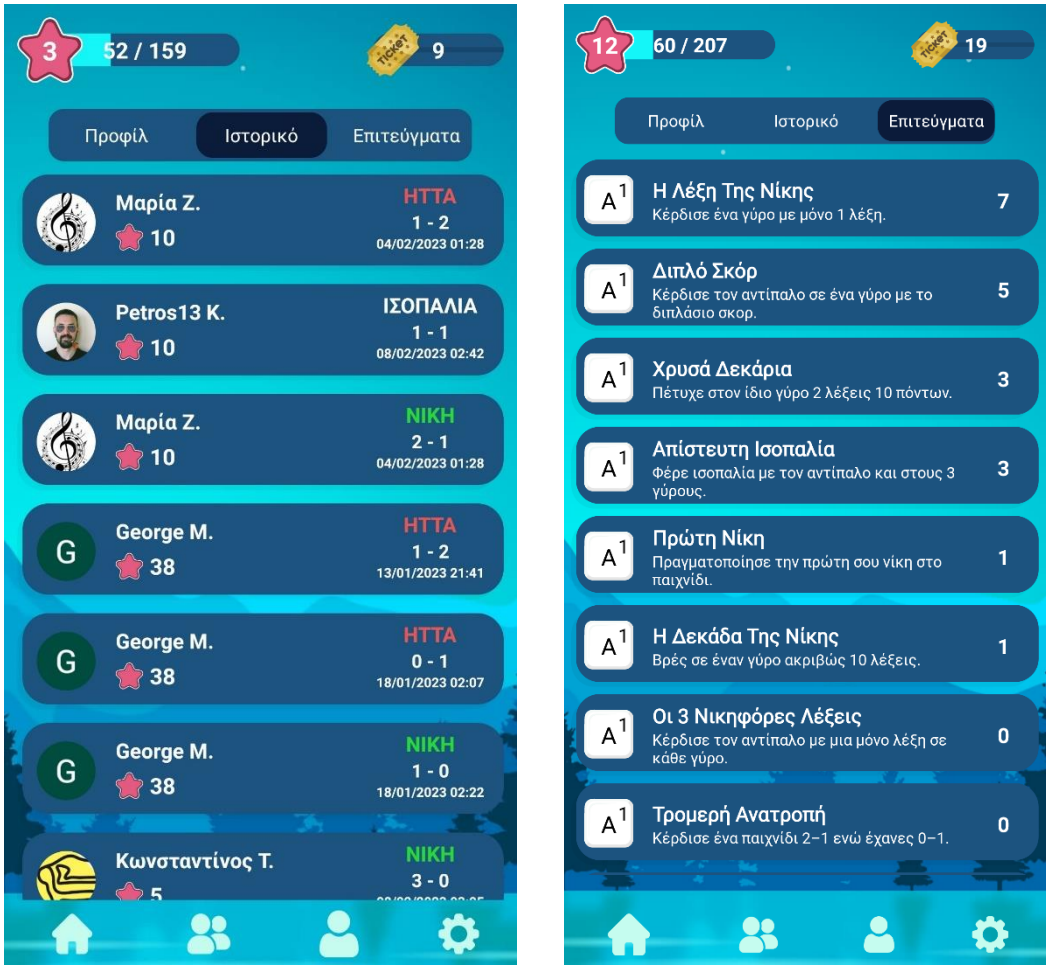

Εικόνα 5.2.2.β: Αναζήτηση φίλων με βάση το όνομα τους.

### 5.2.3 Οθόνη προφίλ χρήστη

Ο χρήστης, σε αυτήν την οθόνη μπορεί να δει πληροφορίες που συσχετίζονται με το προφίλ του, όπως το όνομα του, το επίπεδο του, την εικόνα προφίλ του, καθώς και τα στατιστικά του, που βασίζονται στα παιχνίδια που έχει παίξει συνολικά, όπως, συνολικός αριθμός καθώς και ποσοστά για τις ήττες, τις νίκες και τις ισοπαλίες που έχει πέτυχει. Επιπλέον, ο χρήστης μπορεί να δει το ιστορικό των παιχνιδιών που έχει παίξει, το οποίο περιλαμβάνει ημερομηνία και ώρα παιχνιδιού, σκορ, αποτέλεσμα , όνομα, επίπεδο και εικόνα προφίλ αντίπαλου. Τέλος, η οθόνη αυτή παρέχει και την δυνατότητας της προβολής των επιτευγμάτων του, τα οποία ουσιαστικά είναι, αυθαίρετες προκλήσεις που ορίζονται από τον προγραμματιστή και πρέπει να ανταποκριθούν στον παίκτη.




Εικόνα 5.2.3.α: Προφίλ χρήστη.




Εικόνα 5.2.3.β: Ιστορικό παιχνιδιών και επιτεύγματα.

## 5.3 Παιχνίδι εναντίον αντίπαλου

Η βασική ιδέα της εφαρμογής που υλοποιήθηκε ήταν πως οι χρήστες να μπορούν να συνδεθούν και να παίξουν με άλλους χρήστες, διαδικτυακά, σε πραγματικό χρόνο. Έτσι λοιπόν, μεταξύ άλλων, η εφαρμογή διαθέτει και ένα μηχανισμό για αυτόν τον σκοπό, ο οποίος είναι υπεύθυνος για την αποστολή και λήψη αιτημάτων παιχνιδιού, συγχρονισμό και συντονισμό χρηστών με σκοπό την ομαλή διεξαγωγή των παιχνιδιών μεταξύ των χρηστών. η εφαρμογή, υποστηρίζει:

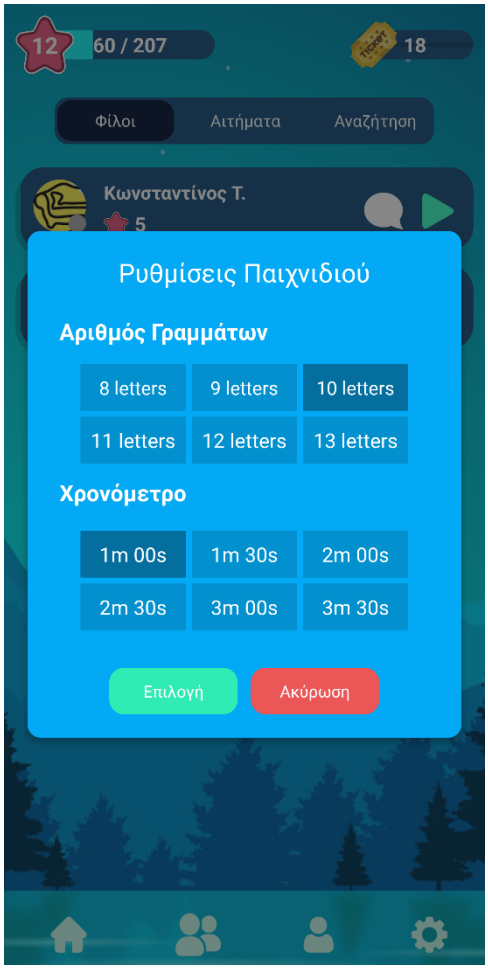
- Χρήστες να μπορούν να παίξουν με χρήστες που βρίσκονται στην λίστα φίλων τους.
- Χρήστες να μπορούν να παίξουν με τυχαίους αντιπάλους.

Όπου, στην τελευταία περίπτωση, περιλαμβάνεται και η δυνατότητα της εφαρμογής να παρέχει εικονικούς χρήστες ως τυχαίους αντιπάλους, μέσω της παροχής ενός μηχανισμού bot, όπου οι

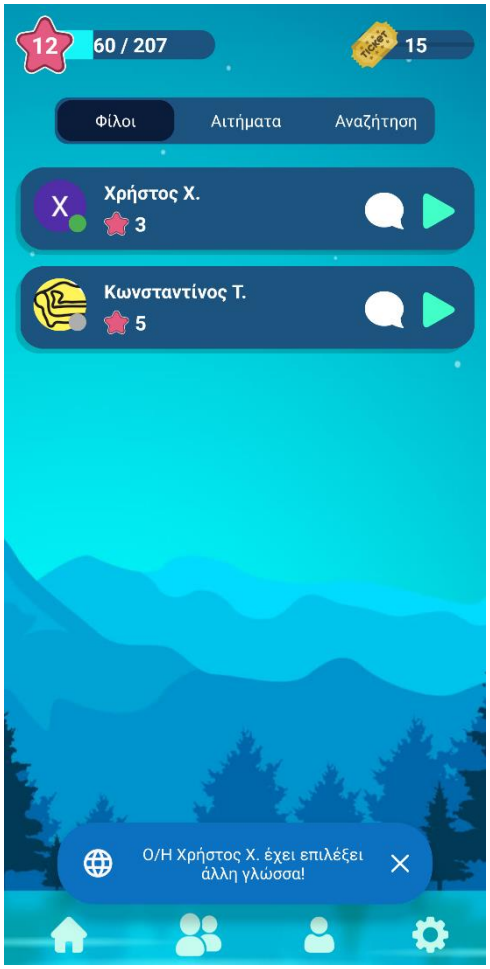
χρήστες παίζουν ενάντιων AI bot, και αντιμετωπίζει την περίπτωση που δεν θα υπάρχουν διαθέσιμοι αντίπαλοι, σε μια δεδομένη χρονική στιγμή.

### 5.3.1 Πρόσκληση παιχνιδιού σε χρήστη από την λίστα φίλων

Στην περίπτωση αυτή, οι χρήστες της εφαρμογής μπορούν να παίξουν με χρήστες που βρίσκονται στην λίστα φίλων τους, στέλνοντας ή λαμβάνοντας πρόσκληση (αίτημα) παιχνιδιού (Game Invite). Για την αποστολή ενός αιτήματος - πρόσκλησης παιχνιδιού, ο χρήστης πρέπει να μεταβεί στην λίστα φίλων, να επιλέξει το πράσινο τριγωνικό εικονίδιο (Play Icon), όπου στην συνέχεια, του εμφανίζεται ένα μενού, με τις ρυθμίσεις του παιχνιδιού, που περιλαμβάνει την επιλογή γραμμάτων και την διάρκεια του γύρου. Έτσι αν ο χρήστης που λαμβάνει την πρόσκληση, την αποδεχτεί, το παιχνίδι αυτό, θα περιέχει τις ρυθμίσεις που έχει επιλέξει ο αποστολέας.

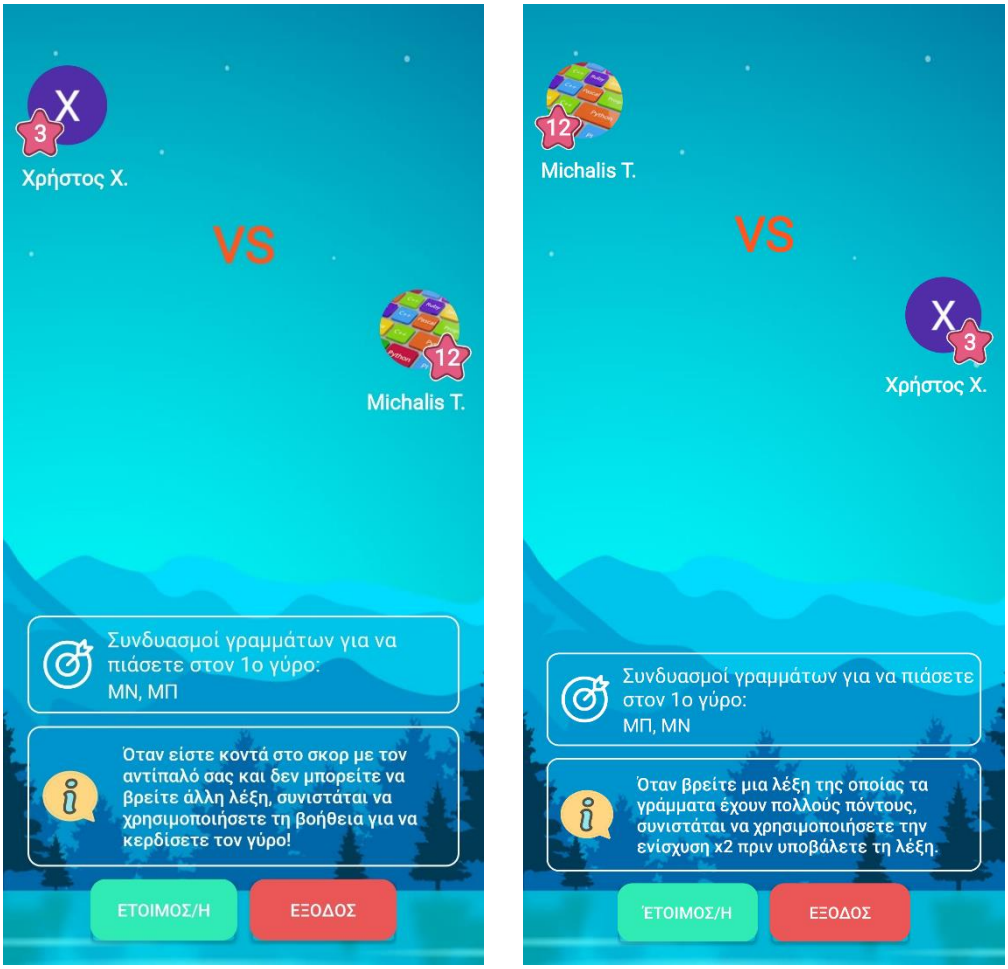



Εικόνα 5.3.1.α: Αποστολή και λήψη, αιτήματος παιχνιδιού.

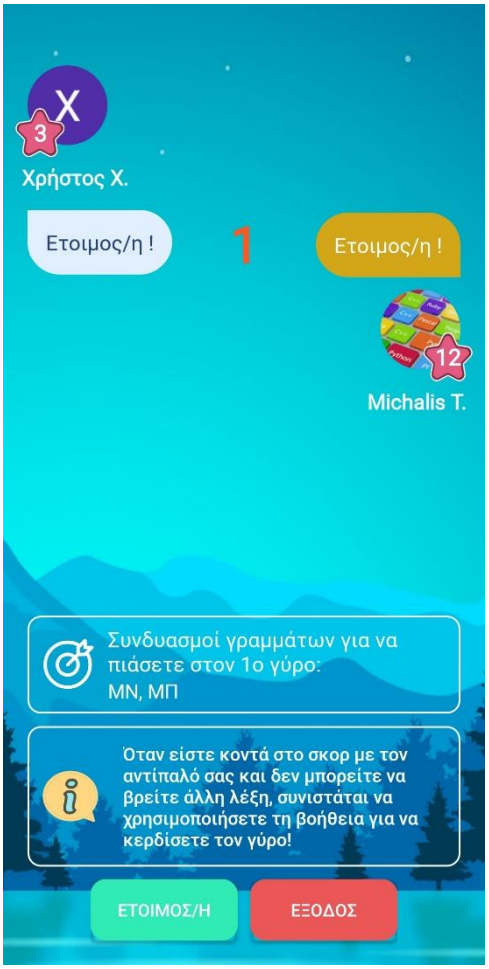



Εικόνα 5.3.1.β: Οι δύο χρήστες πρέπει να έχουν την ίδια γλώσσα παιχνιδιού.

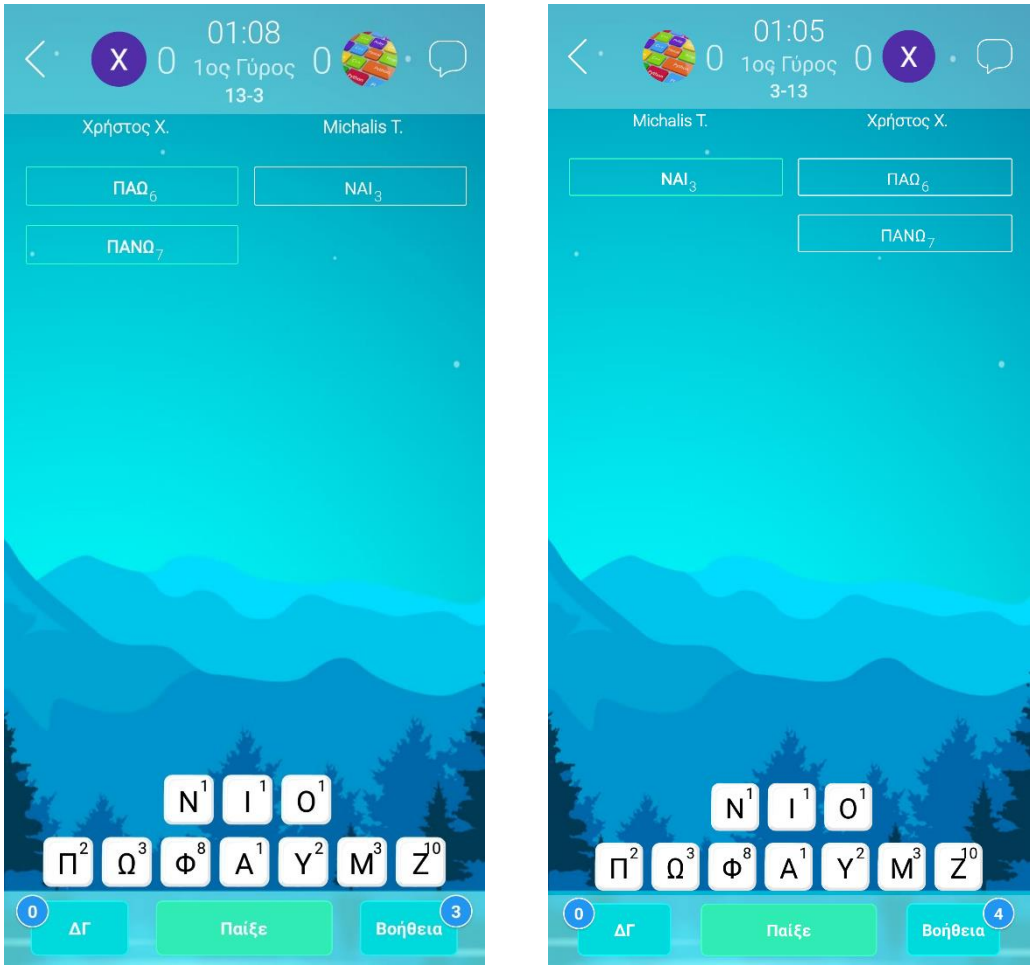
Εφόσον ο χρήστης που λαμβάνει το αίτημα παιχνιδιού, το αποδεχτεί, και εφόσον είναι διαθέσιμος και αποστολές, οι δυο χρήστες οδηγούνται στο ίδιο δωμάτιο παιχνιδιού (Game Lobby). Για την έναρξη του «αγώνα», θα πρέπει και οι δύο χρήστες να πατήσουν το κουμπί «Έτοιμος». Με αυτόν τον τρόπο, όπως αναφέρθηκε και στο Κεφάλαιο 4, παράγραφος 4.8, γίνεται συγχρονισμός των δυο αντίπαλων και ξεκινάει η αντιστροφή μέτρηση για την έναρξη του «αγώνα».



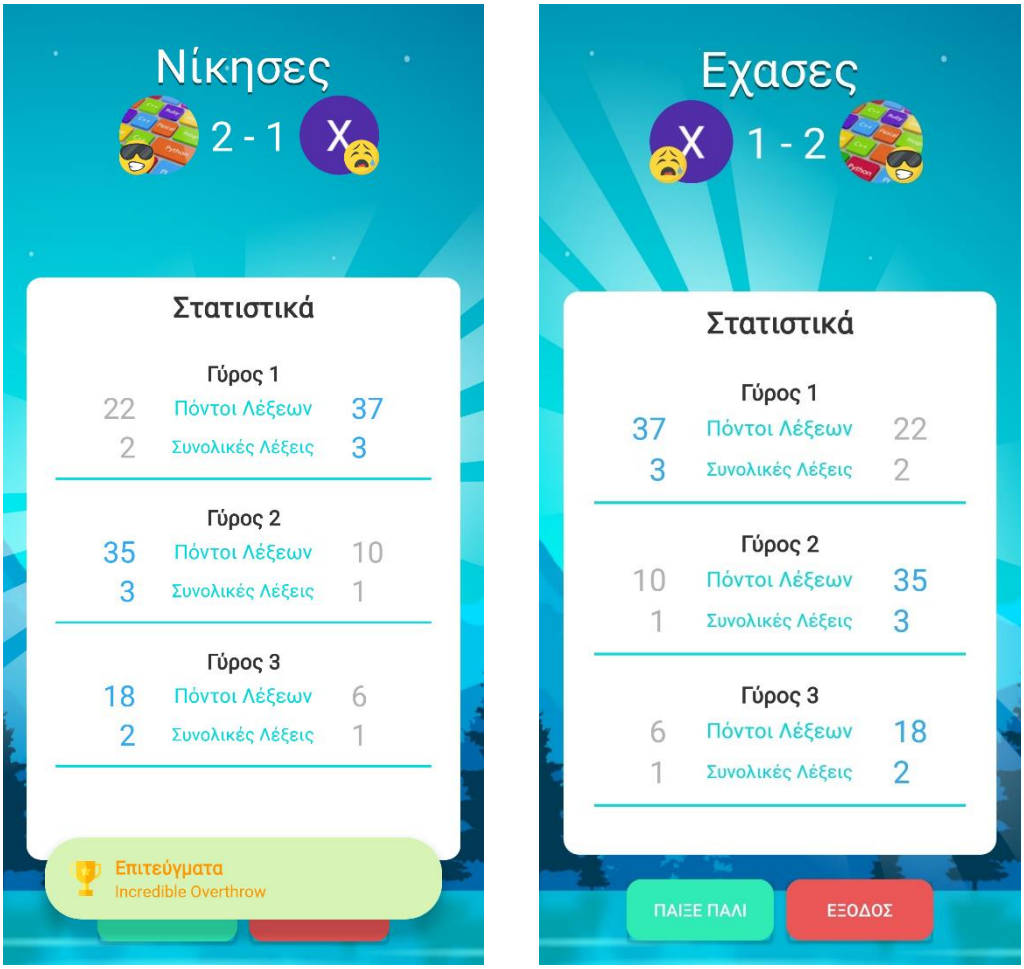

Εικόνα 5.3.1.γ: Το δωμάτιο παιχνιδιού (Game Looby)




Εικόνα 5.3.1.δ: Οι δύο χρήστες είναι έτοιμοι και έχει ξεκινήσει η αντίστροφη μέτρηση.

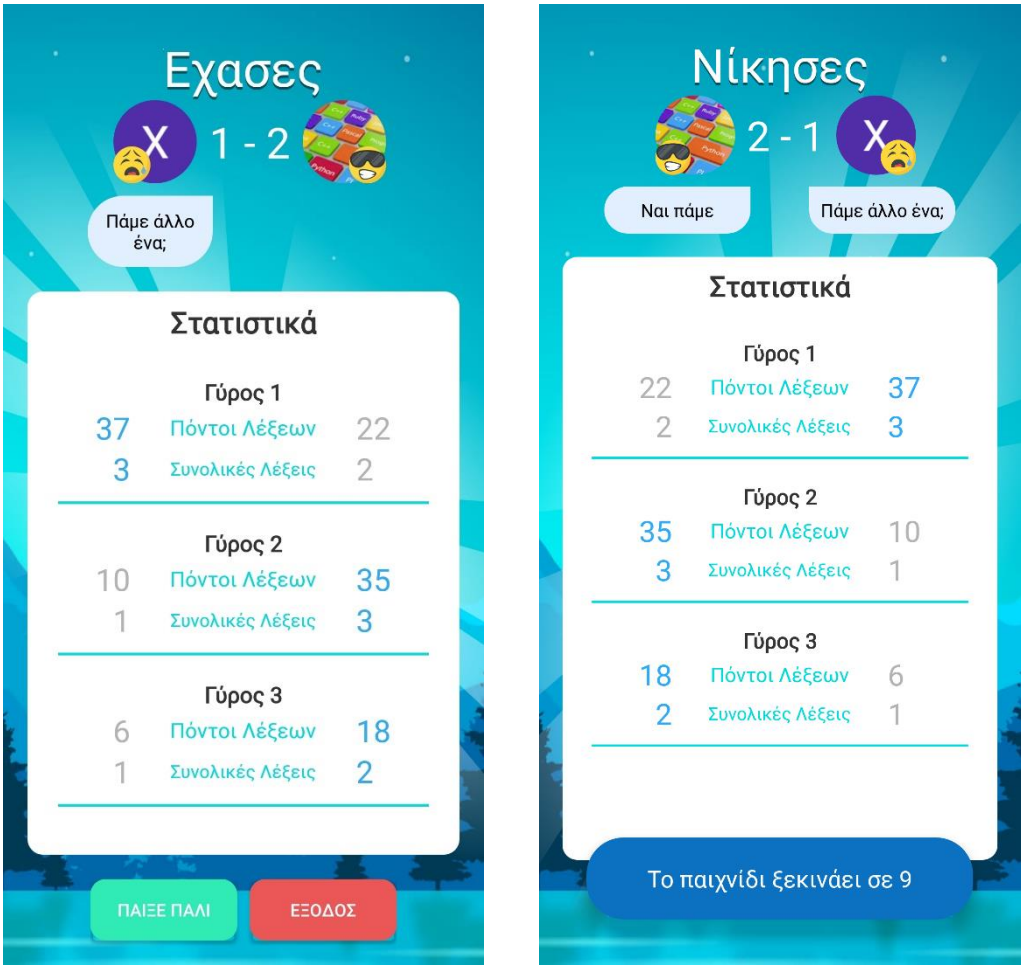



Εικόνα 5.3.1.ε: Οι λέξεις που έχουν σχηματίσει οι δύο αντίπαλοι κατά την διάρκεια του 1ου γύρου.

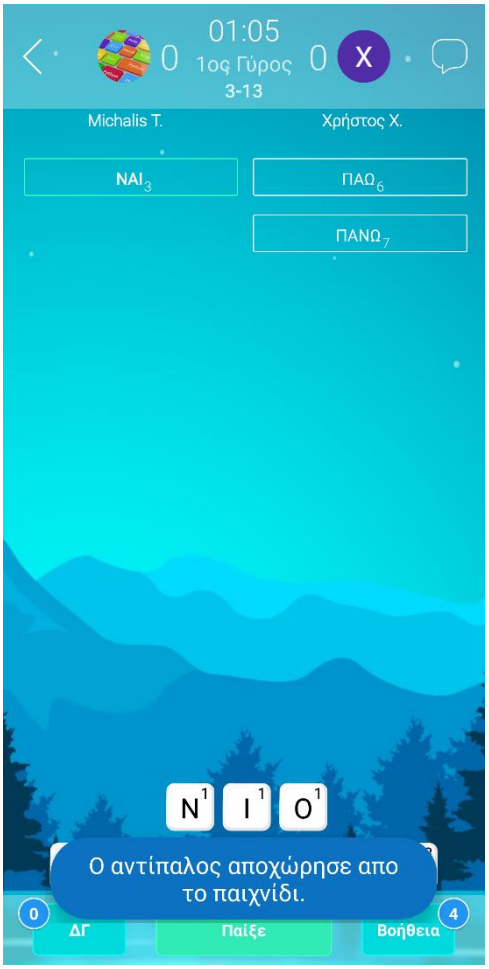



Εικόνα 5.3.1.στ: Στατιστικά παιχνιδιού μετά την λήξη του «αγώνα».


Εικόνα 5.3.1.ζ: Ο χρήστης κερδίζει εμπειρία επιπέδου. (Level XP)




Εικόνα 5.3.1.η: Αίτημα και αποδοχή ρεβάνς.

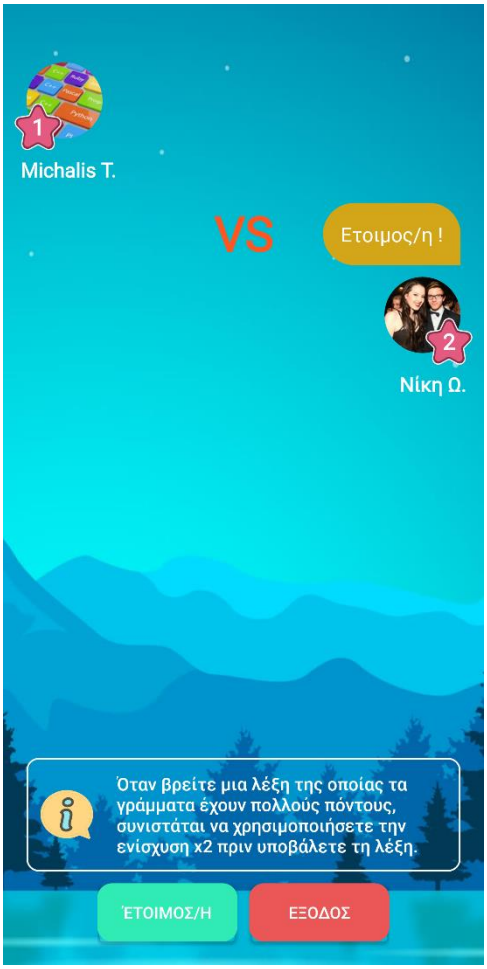


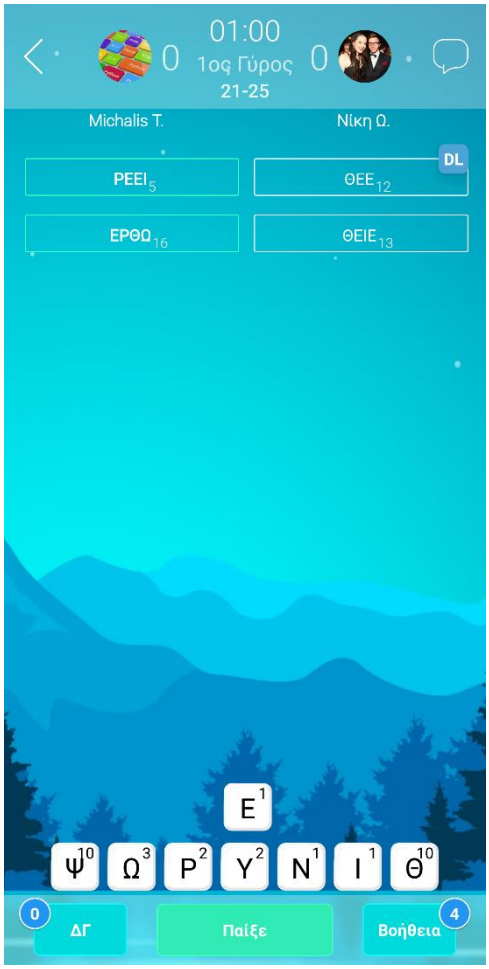

Εικόνα 5.3.1.θ: Ειδοποίηση χρήστη σε περίπτωση αποχώρησης του αντίπαλου.

## 5.3.2 Παιχνίδι με τυχαίο αντίπαλο

Σε προηγούμενα κεφάλαια τονίστηκε και αναλύθηκε το ποσό σημαντικό ρολό έχει η δυνατότητα των παιχνιδιών πραγματικού χρόνου, να παρέχουν στους χρήστες άλλους χρήστες ως τυχαίους αντίπαλους, και όχι να περιορίζονται μόνο στην λίστα φίλων τους. Στην εφαρμογή της παρούσας διπλωματικής, ο χρήστης, έχει την δυνατότητα να παίξει με τυχαίο αντίπαλο που βρίσκεται κοντά στο ίδιο επίπεδο με αυτόν. Για να επιτευχθεί αυτό, πρέπει απλά να γίνει κλικ στο κουμπί «Παίξε με τυχαίο αντίπαλο» που βρίσκεται στο επάνω μέρος της αρχικής οθόνης και να επιλέξει των αριθμό γραμμάτων και την διάρκεια του γύρου που επιθυμεί. Βεβαία, η εύρεση ενός τυχαίου αντιπάλου, με το ίδιο επίπεδο χρήστη και τις ίδιες ρυθμίσεις παιχνιδιού (αριθμός γραμμάτων και χρονόμετρο) σε μια τυχαία χρονική στιγμή, μπορεί να είναι χρονοβόρα ή ανέφικτη, και ιδιαίτερα, όταν η εφαρμογή δεν έχει μεγάλο κοινό και συνάμα αρκετούς ενεργούς χρήστες. Για τον λόγο αυτό, η εύρεση τυχαίου αντιπάλου περιορίζεται μόνο

στη εύρεση αντιπάλου σε κοντινό ή στο ίδιο επίπεδο με τον χρήστη. Έτσι, αν υπάρξει ζευγάρωμα των τυχαίων αντίπαλων, στο παιχνίδι που θα παίξουν, θα περιλαμβάνει την μια από τις δυο επιλογές (αριθμός γραμμάτων ή χρονόμετρο) του κάθε χρήστη. Τέλος, όπως αναφέρθηκε και στο κεφάλαιο 4, παράγραφος 4.4, αν μέσα σε ένα χρονικό διάστημα δεν υπάρξει ζευγάρωμα με τυχαίο αντίπαλο, τότε ως τυχαίος αντίπαλος ορίζεται ένα AI bot, ανάλογου επιπέδου του χρήστη.



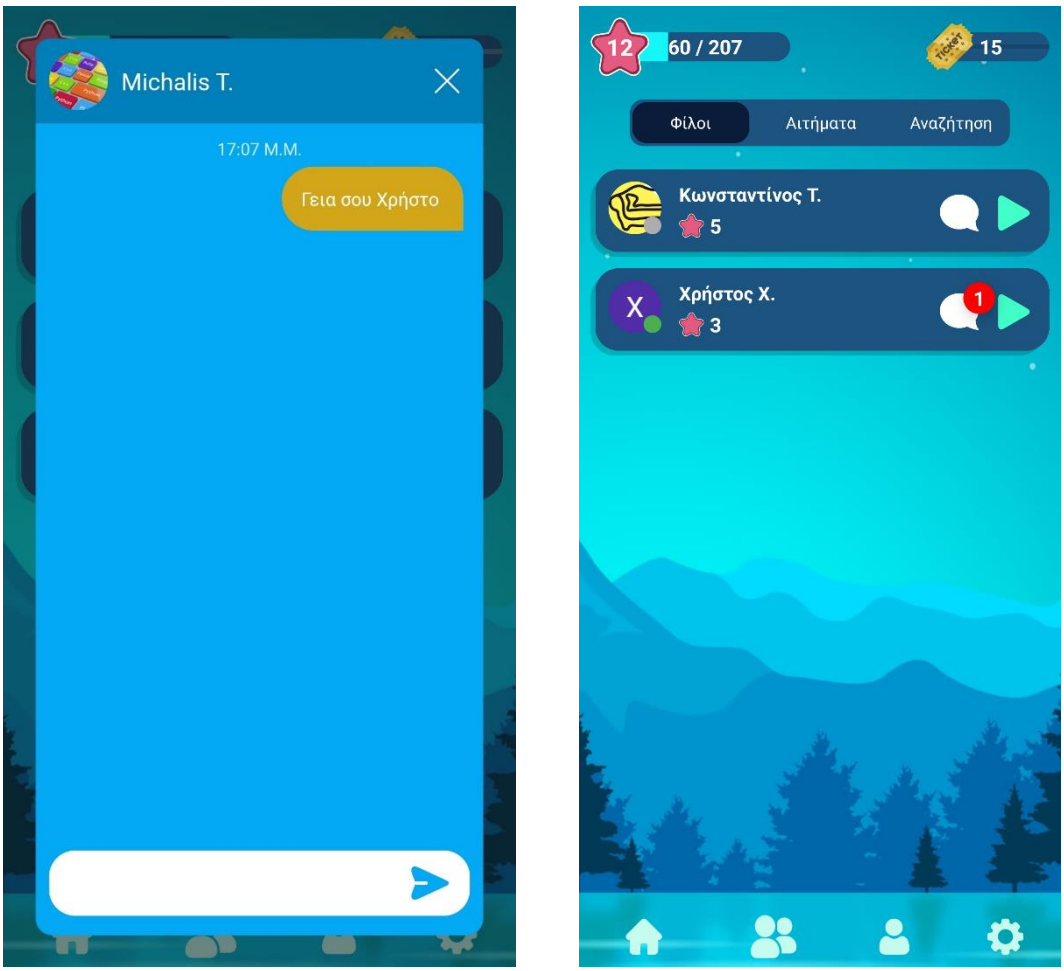

Εικόνα 5.3.2 Δημιουργία bot, ως τυχαίος αντίπαλος, με ρεαλιστικό όνομα, εικόνα προφίλ, και επίπεδο, το οποίο μπορεί και δημιουργεί λέξεις και προσαρμόζεται στο επίπεδο του χρήστη με την βοήθεια της ασαφούς λογικής.

## 5.4 Συνομιλία Χρηστών

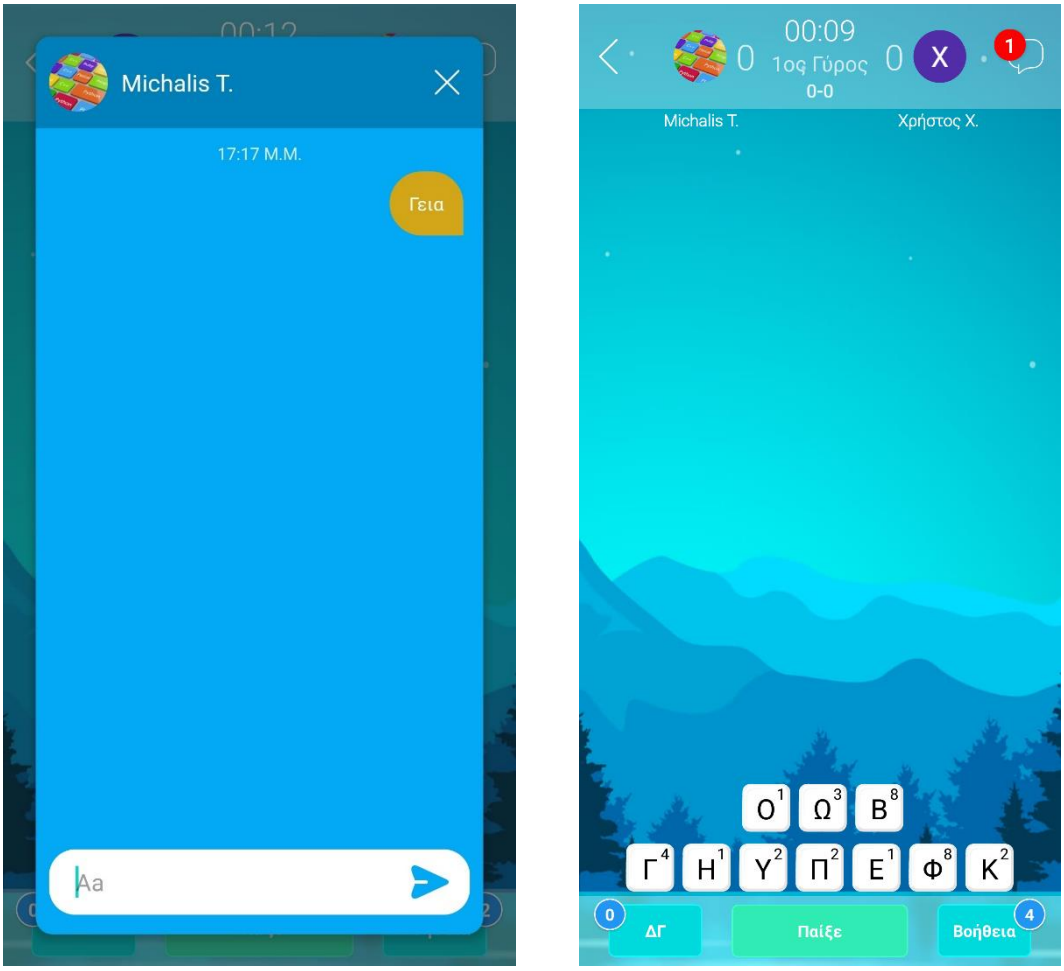
Μια εξίσου σημαντική λειτουργία ενός παιχνιδιού πραγματικού χρόνου, όπως αυτό έχει αναλυθεί, είναι η δυνατότητα της συνομιλίας μεταξύ των χρηστών της εφαρμογής. Επιπλέον, εφαρμογές που παρέχουν υπηρεσίες συνομιλίας, οφείλουν να ακολουθούν τους κανόνες προστασίας προσωπικών δεδομένων και να κρυπτογραφούν τα δεδομένα (μηνύματα) που ανταλλάζουν μεταξύ τους οι χρήστες. Στην παρούσα εφαρμογή που αναπτύχθηκε και παρουσιάζεται, ακολουθούνται οι κανόνες προσωπικών δεδομένων και για τον λόγο αυτό εφαρμόστηκε ο αλγόριθμος κρυπτογράφησης συμμετρικού κλειδιού AES. Η εφαρμογή υποστηρίζει την συνομιλία:

Με χρήστη που βρίσκεται στην λίστα φίλων.

Κατά την διάρκεια του «αγώνα» είτε με τυχαίο αντίπαλο, είτε με χρήστη από την λίστα φίλων.




Εικόνα 5.4.α Αποστολή και λήψη μηνύματος σε χρήστη που βρίσκεται στην λίστα φίλων.




Εικόνα 5.4.β Αποστολή και λήψη μηνύματος κατά την διάρκεια του «αγώνα».

# Κεφάλαιο 6 - Μελλοντικές Βελτιώσεις και Επεκτάσεις

## 6.1 Βελτίωση υπηρεσίας back-end της εφαρμογής

Πολλές εμπορικές και μη εφαρμογές, χρησιμοποιούν το μοντέλο αρχιτεκτονικής λογισμικού Πελάτη - Εξυπηρετητή (Client – Server), δηλαδή τα δεδομένα των χρηστών δεν παραμένουν local (τοπικά) στην συσκευή (Client), αλλά αποθηκεύονται σε έναν σέρβερ. (Server) Στο μοντέλο αυτό, ο πελάτης (client) «ζητά κάτι» και ένα άλλο τμήμα του λογισμικού, ο διακομιστής, του το επιστρέφει, έχουμε δηλαδή την σχέση request – response. Για μια εφαρμογή, το στήσιμο και η συντήρηση μιας υπηρεσίας back-end η οποία θα τρέχει σε έναν

server, και στην οποία πολλοί πελάτες θα συνδέονται και θα αλληλοεπιδρούν με αυτήν, αποτελεί έναν «πονοκέφαλο», που απαιτεί πόρους ανθρώπινους και οικονομικούς καθώς πολλοί παράμετροι και τεχνικά ζητήματα πρέπει να ληφθούν υπόψιν, ώστε να υλοποιηθεί αυτή η υπηρεσία back – end, σωστά και να έχει μια άρτια συμπεριφορά και λειτουργία ώστε να εξυπηρετεί σωστά τους πελάτες. Έτσι, αρκετοί πάροχοι, προσφέρουν υπηρεσίες cloud, δωρεάν έως ένα προκαθορισμένο όριο χρήσης, και στην συνέχεια με την φιλοσοφία του «pay as you go», ώστε να απαλλάξουν τις εφαρμογές και τους προγραμματιστές από τα αναφερθέν κωλύματα όπως στήσιμο και συντήρηση. Μικρές εφαρμογές και μη, και ιδιαίτερα, μια νεοφυής επιχείρηση (Start-Up), στρέφονται προς αυτές τις βολικές λύσεις, όσο το κόστος τους το επιτρέπει. Μια από αυτές τις υπηρεσίες cloud, είναι και το Firebase, το οποίο χρησιμοποιήθηκε ως back – end υπηρεσία στην παρούσα διπλωματική.

Η εφαρμογή που υλοποιήθηκε, βασίζεται σε μηνύματα που ανταλλάζουν οι χρήστες μεταξύ τους, καθώς είναι μια εφαρμογή που αποτελεί ένα παιχνίδι πραγματικού χρόνου, και όλες οι ενέργειες που πραγματοποιεί ο ένας χρήστης πρέπει να είναι ορατές και στον άλλον. Επιπλέον τα πλήρη λεξικά του παιχνιδιού, για τις γλώσσες, Αγγλικά, Ελληνικά, Ισπανικά, διατηρούνται στο Firebase και επομένως τα reads που θα γίνονται ανά γύρο και ανά χρήστη, σε συνδυασμό με τα μηνύματα συγχρονισμού στα οποία βασίζεται η αρχιτεκτονική της εφαρμογής, θα αυξήσουν κατακόρυφα τις εγγραφές και τις προσπελάσεις των δεδομένων που γίνονται στην βάση, σε μια ίσως, ενδεχομένη αύξηση των χρηστών της εφαρμογής που θα την χρησιμοποιούν καθημερινά. Επομένως, αν η εφαρμογή αποκτήσει μεγάλο κοινό χρηστών, θα υπάρξει και ένα κόστος μηνιαίως, το οποίο μπορεί να είναι υψηλότερο, από ότι θα ήταν, αν η εφαρμογή διατηρούσε δικιά της back-end υπηρεσία, όπως αναφέρθηκε προηγουμένως. Για αυτήν την περίπτωση υπάρχουν 2 πιθανές λύσεις:

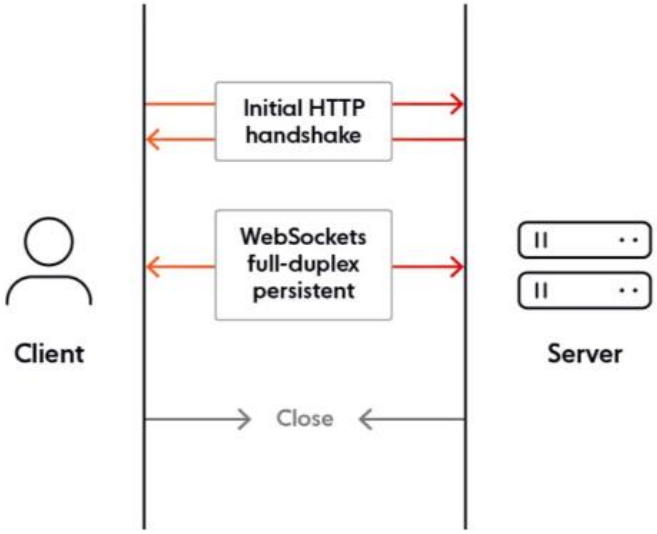
1. Βελτιστοποίηση των προσπελάσεων και των εγγραφών που γίνονται από τους clients προς την βάση (Firebase).
2. Δημιουργία back-end υπηρεσία, η οποία θα βασίζεται στο πρωτόκολλο επικοινωνίας WebSocket.

Η βελτιστοποίηση αυτή μπορεί να στραφεί προς τις προσπελάσεις που γίνονται στο πλήρες λεξικό που βρίσκεται αποθηκευμένο στην βάση. Όπως έχει αναλυθεί και στο κεφάλαιο 4 παράγραφος 2, η εφαρμογή διαθέτει ένα μικρό λεξικό, τοπικά αποθηκευμένο, το οποίο χρησιμοποιείται, όταν ο χρήστης δημιουργεί μια λέξη. Αν η λέξη δεν υπάρχει στο μικρό λεξικό, τότε ελέγχεται το πλήρες λεξικό που υπάρχει αποθηκευμένο στην βάση. Κατά

συνέπεια, τα reads που θα γίνονται στην βάση, ενδέχεται να είναι πολυάριθμα, για μεγάλο αριθμό χρηστών. Η λύση είναι, να ενισχυθεί το τοπικά αποθηκευμένο μικρό λεξικό, συλλέγοντας με την βοήθεια του Firebase, τις Ν συχνότερες λέξεις που οι χρήστες βρίσκουν και δημιουργούν. Ο αριθμός Ν μπορεί να καθοριστεί αναλόγως του μεγέθους που θέλουμε να έχει το μικρό λεξικό.

## 6.2 Πρωτόκολλο Επικοινωνίας WebSocket

Το WebSocket είναι ένα πρωτόκολλο επικοινωνίας υπολογιστή, που παρέχει κανάλια επικοινωνίας πλήρους αμφίδρομης λειτουργίας μέσω μιας μόνο σύνδεσης TCP. Το πρωτόκολλο WebSocket τυποποιήθηκε από την IETF ως RFC 6455 το 2011. Η τρέχουσα προδιαγραφή API που επιτρέπει στις εφαρμογές Ιστού να χρησιμοποιούν αυτό το πρωτόκολλο είναι γνωστή ως WebSockets. Το WebSocket διαφέρει από το HTTP. Και τα δύο πρωτόκολλα βρίσκονται στο επίπεδο 7 του μοντέλου OSI και εξαρτώνται από το TCP στο επίπεδο 4. Αν και είναι διαφορετικά, το έγγραφο RFC 6455 δηλώνει ότι το WebSocket «έχει σχεδιαστεί για να λειτουργεί πάνω από τις θύρες HTTP 443 και 80 καθώς και να υποστηρίζει διακομιστές μεσολάβησης και μεσάζοντες HTTP», καθιστώντας το έτσι συμβατό με HTTP. Μια σύνδεση WebSocket ξεκινά ως χειραψία αιτήματος/απόκρισης HTTP μεταξύ του πελάτη και του διακομιστή. Ο πελάτης ξεκινά πάντα τη χειραψία. στέλνει ένα αίτημα GET στον διακομιστή, υποδεικνύοντας ότι θέλει να αναβαθμίσει τη σύνδεση από HTTP σε WebSockets. Ο διακομιστής πρέπει να επιστρέψει έναν κωδικό απόκρισης HTTP 101 Switching Protocols για να δημιουργηθεί η σύνδεση WebSocket. Μόλις η αναβάθμιση της σύνδεσης είναι επιτυχής και αλλάξει από HTTP σε WebSocket, ο πελάτης και ο διακομιστής μπορούν ελεύθερα να ανταλλάσσουν μηνύματα χαμηλής καθυστέρησης μέσω της σύνδεσης όπως και όταν χρειάζεται. Αφού η σύνδεση WebSocket εξυπηρετήσει το σκοπό της, μπορεί να τερματιστεί μέσω χειραψίας κλεισίματος (τόσο ο πελάτης όσο και ο διακομιστής μπορούν να την εκκινήσουν).




Εικόνα 6.2.α Πρωτόκολλο Επικοινωνίας WebSocket

Συνεπώς, η χρήση μιας back-end υπηρεσίας, βασισμένη στο πρωτόκολλο επικοινωνίας WebSocket, είναι ιδανική για εφαρμογές που απαιτούν δεδομένα και αλληλεπιδράσεις σε πραγματικό χρόνο, όπως είναι η εφαρμογή που υλοποιήθηκε για τις ανάγκες τις παρούσας διπλωματικής εργασίας. Το Firebase χρησιμοποιεί WebSockets για συγχρονισμό δεδομένων με πελάτες. Βέβαια η μετάβαση από την χρήση του Firebase, σε μια WebSockets υπηρεσία, δεδομένου οτι η εφαρμογή έχει στηθεί πάνω στο Firebaase, είναι επίπονη, απαιτεί εμπειρία τόσο των τεχνολογιών web, όσο και του ίδιου του πρωτοκόλλου, και απαιτεί πολύ καλή μελέτη των αναγκών της εφαρμογής και των μεταβάσεων που θα πρέπει να γίνουν, καθώς, πολλές αυτοματοποιημένες διαδικασίες που θεωρούνται αυτονόητες με το Firebase, πλέον θα πρέπει να υλοποιηθούν από την υπηρεσία.

## 6.3 Βελτίωση αλγορίθμου κρυπτογράφησης των μηνυμάτων συνομιλίας

Στο κεφάλαιο 3 παράγραφος 10.4, παρουσιάστηκε ο αλγόριθμος AES, ένας συμμετρικός αλγόριθμος με μέγεθος μπλοκ/τεμαχίου 128 bit. Ο αλγόριθμος αυτός μετατρέπει τα μεμονωμένα μπλοκ χρησιμοποιώντας κλειδιά των 128, 192 και 256 bit. Στην υλοποίηση του αλγόριθμου αυτού για τις ανάγκες της παρούσας διπλωματικής χρησιμοποιήθηκε μέγεθος κλειδιού 128bit. Παρόλο που ο αλγόριθμος AES χρησιμοποιείται σε πληθώρα εφαρμογών και

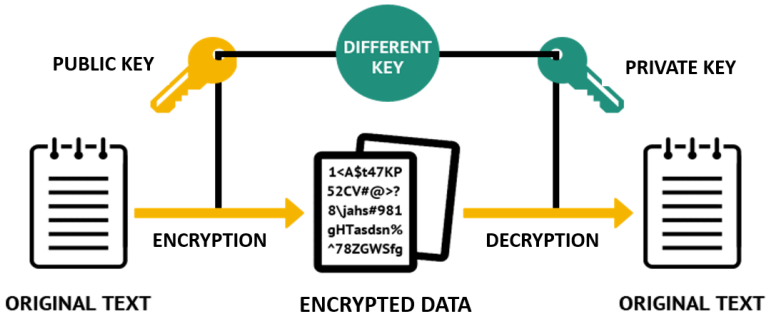
συστήματα, εντούτοις οι πιο συνηθισμένες περιπτώσεις εφαρμογές του αλγορίθμου τις συναντάμε σε:

- Wireless Security
- Encrypted Browsing
- General File Encryption
- Mobile applications
- Processor Security
- OS system components
- Password managers

Αυτό όμως δεν σημαίνει πως η υλοποίηση του αλγόριθμου AES, δεν ενδείκνυται και για περιπτώσεις κρυπτογράφησης μηνυμάτων συνομιλίων, όπως ακριβώς παρουσιάστηκε και υλοποιήθηκε στην εφαρμογή της παρούσας διπλωματικής. Παρακάτω παρουσιάζεται ένας αλγόριθμος κρυπτογράφησης ασύμμετρου κλειδιού, καθώς επίσης περιγράφεται και η δυνατότητα συνδυασμού ενός συστήματος κρυπτογράφησης ασύμμετρου κλειδιού και ενός συστήματος συμμετρικού κλειδιού, για μια πιο ολοκληρωμένη προστασία των δεδομένων.

### 6.3.1 Ασύμμετρη κρυπτογράφηση.

Στους αλγόριθμους ασύμμετρης κρυπτογράφησης, χρησιμοποιείτε δύο διαφορετικά κλειδιά, το ένα για κρυπτογράφηση και το άλλο για αποκρυπτογράφηση. Το κλειδί που χρησιμοποιείται για την κρυπτογράφηση είναι το δημόσιο κλειδί και το κλειδί που χρησιμοποιείται για την αποκρυπτογράφηση είναι το ιδιωτικό κλειδί. Αλλά, φυσικά, και τα δύο κλειδιά πρέπει να ανήκουν στον δέκτη.

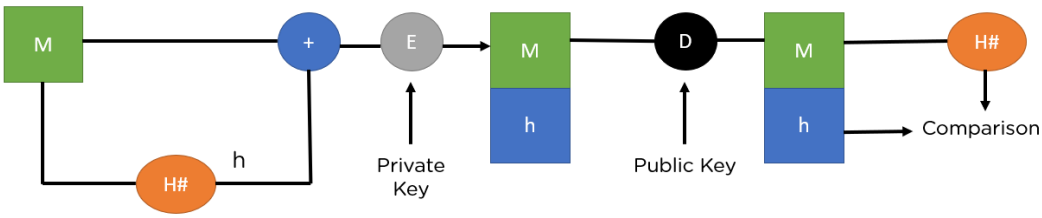


Εικόνα 6.3.1.α Η διαδικασία ασύμμετρης κρυπτογράφησης.

Όπως φαίνεται στην παραπάνω εικόνα, η χρήση διαφορετικών κλειδιών για κρυπτογράφηση και αποκρυπτογράφηση έχει βοηθήσει στην αποφυγή ανταλλαγής κλειδιών, όπως φαίνεται στη συμμετρική κρυπτογράφηση.

### 6.3.2 Ψηφιακές υπογραφές.

Οι ψηφιακές υπογραφές εξυπηρετούν τον έλεγχο ταυτότητας και επαλήθευση εγγράφων και αρχείων. Αυτό είναι ζωτικής σημασίας για την αποφυγή παραβίασης κατά τη μετάδοση των επίσημων εγγράφων και την αποφυγή ψηφιακού χειρισμού ή πλαστογραφίας. Εργάζονται στην αρχιτεκτονική κρυπτογραφίας δημόσιου κλειδιού, αποκλείοντας μια μικρή προειδοποίηση. Συνήθως, το σύστημα ασύμμετρου κλειδιού χρησιμοποιεί ένα δημόσιο κλειδί για κρυπτογράφηση και ένα ιδιωτικό κλειδί για αποκρυπτογράφηση. Ωστόσο, όταν έχουμε να κάνουμε με ψηφιακές υπογραφές, συμβαίνει το αντίθετο. Το ιδιωτικό κλειδί χρησιμοποιείται για την κρυπτογράφηση της υπογραφής και το δημόσιο κλειδί για την αποκρυπτογράφηση της. Δεδομένου ότι τα κλειδιά λειτουργούν παράλληλα μεταξύ τους, η αποκρυπτογράφηση του με το δημόσιο κλειδί σημαίνει ότι χρησιμοποίησε το σωστό ιδιωτικό κλειδί για να υπογράψει το έγγραφο, επαληθεύοντας έτσι την προέλευση της υπογραφής.




Εικόνα 6.3.2.α Η διαδικασία της ψηφιακής υπογραφής.

Όπου:

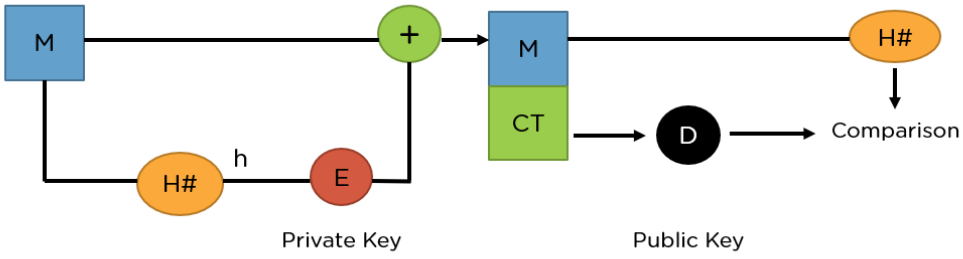
- M – Απλό κείμενο (Plaintext)
- H - Συνάρτηση κατακερματισμού (Hash function)

- h - Η έξοδος της συνάρτησης κατακερματισμού (Hash digest)
- '+' - Η ομαδοποιηση του απλού κειμένου καθώς και της εξόδου της συνάρτησης κατακερματισμού (Bundle both plaintext and digest)
- E - Κρυπτογράφηση (Encryption)
- D - Αποκρυπτογράφηση (Decryption)

Η παραπάνω εικόνα δείχνει όλη τη διαδικασία, από την υπογραφή του κλειδιού μέχρι την επαλήθευση του.

### 6.3.3 Ο αλγόριθμος RSA.

Ο αλγόριθμος RSA είναι ένας αλγόριθμος υπογραφής δημόσιου κλειδιού που αναπτύχθηκε από τους Ron Rivest, Adi Shamir και Leonard Adleman. Το έγγραφό τους δημοσιεύτηκε για πρώτη φορά το 1977 και ο αλγόριθμος χρησιμοποιεί λογαριθμικές συναρτήσεις για να διατηρεί το σύνθετο έργο αρκετά ώστε να αντέχει την ωμή βία και αρκετά βελτιωμένο ώστε να είναι γρήγορο μετά την ανάπτυξη. Η παρακάτω εικόνα δείχνει ότι επαληθεύει τις ψηφιακές υπογραφές χρησιμοποιώντας τη μεθοδολογία RSA.




Εικόνα 6.3.3.α Η διαδικασία κρυπτογράφησης του αλγορίθμου RSA.

Ο αλγόριθμος RSA μπορεί επίσης να κρυπτογραφήσει και να αποκρυπτογραφήσει γενικές πληροφορίες για την ασφαλή ανταλλαγή δεδομένων μαζί με τον χειρισμό της επαλήθευσης ψηφιακής υπογραφής. Η παραπάνω εικόνα δείχνει ολόκληρη τη διαδικασία του αλγόριθμου RSA.

### 6.3.4 Τα βήματα του αλγορίθμου RSA

Όταν χρησιμοποιείται RSA για κρυπτογράφηση και αποκρυπτογράφηση γενικών δεδομένων, αντιστρέφει τη χρήση του συνόλου κλειδιών. Σε αντίθεση με την επαλήθευση υπογραφής, χρησιμοποιεί το δημόσιο κλειδί του παραλήπτη για την κρυπτογράφηση των δεδομένων και χρησιμοποιεί το ιδιωτικό κλειδί του παραλήπτη για την αποκρυπτογράφηση των δεδομένων. Έτσι, δεν υπάρχει ανάγκη ανταλλαγής κλειδιών σε αυτό το σενάριο. Υπάρχουν δύο γενικές συνιστώσες όσον αφορά την κρυπτογραφία RSA, είναι:

- **Δημιουργία κλειδιών:** Δημιουργία κλειδιών που θα χρησιμοποιηθούν για την κρυπτογράφηση και την αποκρυπτογράφηση των δεδομένων που θα ανταλλάσσονται.
- **Λειτουργία κρυπτογράφησης/αποκρυπτογράφησης:** Τα βήματα που πρέπει να εκτελεστούν κατά την κωδικοποίηση και την ανάκτηση των δεδομένων.

#### 6.3.4.1 Δημιουργία κλειδιών

Πρέπει να δημιουργηθούν δημόσια και ιδιωτικά κλειδιά πριν την εκτέλεση της συνάρτησης κατακερματισμού, για τη δημιουργία κρυπτογραφημένου κειμένου και απλού κειμένου. Χρησιμοποιούν ορισμένες μεταβλητές και παραμέτρους, οι οποίες εξηγούνται όλες παρακάτω:

- Επιλογή δύο μεγάλων πρώτων αριθμων (***p*** και ***q***)
- Υπολογισμός των $n = p*q$ και $\varphi(n) = (p-1)(q-1)$
- Επιλογή ενός αριθμού ***e***, όπου $1 < e < z$
- Υπολογισμός της $d = e^{-1} \bmod \varphi(n)$
- Ομαδοποίηση των ζευγών ιδιωτικών κλειδιών ως (***n***, ***d***)
- Ομαδοποίηση του ζεύγος δημόσιου κλειδιού ως (***n***, ***e***)

#### 6.3.4.2 Συνάρτηση κρυπτογράφησης/αποκρυπτογράφησης

Μόλις δημιουργηθούν τα κλειδιά, μεταβιβάζονται στις παραμέτρους των συναρτήσεων κατακερματισμού (που υπολογίζουν το κρυπτογραφημένο κείμενο και το απλό κείμενο) χρησιμοποιώντας το αντίστοιχο κλειδί.

Εάν το απλό κείμενο είναι ***m***, τότε το κρυπτογραφημένο κείμενο (encrypt) είναι:

$$c = m^e \bmod n$$

Εάν το κρυπτογραφημένο κείμενο είναι ***c***, τότε το απλό κείμενο (decrypt) είναι:

$$m = c^d \bmod n$$

Για παράδειγμα αν θέσουμε ως ***p*** = 17 και ***q*** = 13, τότε η τιμή του e μπορεί να είναι 5 καθώς ικανοποιεί τη συνθήκη $1 < e < (p-1)(q-1)$. Επιπλέον ισχυέι:

$$N = p * q = 221$$

καθώς και

$$D = e^{-1} \bmod \varphi(n) = 29$$

Τέλος, το ζεύγος δημόσιου κλειδιού είναι (221, 5) καθώς και το ζεύγος ιδιωτικού κλειδιού είναι (221, 29)

Εάν η τιμή του απλού κειμένου (***m***) είναι 10, μπορεί να κρυπτογραφηθεί χρησιμοποιώντας τον τύπο:

$$m^e \bmod n = 82.$$

Για την αποκρυπτογράφηση του κρυπτογραφημένου κειμένου (***c***) πίσω στα αρχικά δεδομένα, πρέπει να χρησιμοποιηθέι ο τύπος:

$$c^d \bmod n = 29.$$

## 6.3.5 Υβριδική κρυπτογράφιση

Στην κρυπτογραφία, μια υβριδική κρυπτογράφηση είναι ο συνδυασμός ενός κρυπτοσυστήματος ασύμμετρου κλειδιού με ένα κρυπτοσύστημα συμμετρικού κλειδιού. Τα κρυπτοσυστήματα ασύμμετρου κλειδιού είναι βολικά καθώς δεν απαιτούν από τον αποστολέα και τον παραλήπτη να μοιράζονται ένα κοινό μυστικό προκειμένου να επικοινωνούν με ασφάλεια. Ωστόσο, συχνά βασίζονται σε πολύπλοκους μαθηματικούς υπολογισμούς και επομένως είναι γενικά πολύ πιο αναποτελεσματικοί από συγκρίσιμα κρυπτοσυστήματα συμμετρικού κλειδιού. Σε πολλές εφαρμογές, το υψηλό κόστος της κρυπτογράφησης μεγάλων

μηνυμάτων σε ένα κρυπτοσύστημα δημόσιου κλειδιού μπορεί να είναι απαγορευτικό. Αυτό αντιμετωπίζεται από τα υβριδικά συστήματα χρησιμοποιώντας έναν συνδυασμό και των δύο. Σε αυτήν την προσέγγιση στην κρυπτογραφία, ο αποστολέας δημιουργεί ένα ιδιωτικό κλειδί, κρυπτογραφεί το κλειδί χρησιμοποιώντας έναν αλγόριθμο δημόσιου κλειδιού και στη συνέχεια κρυπτογραφεί ολόκληρο το μήνυμα (συμπεριλαμβανομένου του ήδη κρυπτογραφημένου ιδιωτικού κλειδιού) με το αρχικό συμμετρικό κλειδί. Συνεπώς, μια προτεινόμενη βελτίωση του συστήματος κρυπτογράφησης, που έχει υλοποιηθεί στην παρούσα διπλωματική, είναι η εφαρμογή υβριδικής κρυπτογράφησης για την κρυπτογράφηση των μηνυμάτων των συνομιλίων μεταξύ των χρηστών. Επιπλέον, για μια πιο ολοκληρωμένη προστασία των δεδομένων που βρίσκονται αποθηκευμένα στο firebase, μια επιπλέον πρόταση βελτίωσης, είναι, η κωδικοποίηση και άλλων δεδομένων, όπως ονόματα χρηστών, url των εικόνων προφίλ τους κτλ.

## 6.4 Λοιπές Βελτιώσεις

Όπως αναλύθηκε στις προηγούμενες παραγράφους, οι προαναφερθέντες αυτές βελτιώσεις, δύναται να λύσουν τόσο το οικονομικό κόστος, που όπως αναλύσαμε παραπάνω, μπορεί να εκτοξευτεί για μεγάλο αριθμό χρηστών, όσο, και το κομμάτι της ασφάλειας των δεδομένων των χρηστών που αποθηκεύονται στο firebase. Εκτός όμως από αυτές τις προτεινόμενες βελτιώσεις, παρακάτω γίνεται μια αναφορά και σε άλλες πιθανές βελτιώσεις στην εφαρμογή, που θα μπορούσαν να πραγματοποιηθούν μελλοντικά, χωρίς όμως να αναλύονται περεταίρω. Οι βελτιώσεις αυτές, επικεντρώνονται κυρίως στο κομμάτι της εμπειρίας του χρήστη (UX) και είναι οι εξής:

- **Βελτίωση του μηχανισμού bot,** που είναι υπεύθυνο για την συμπεριφορά του bot απέναντι στον χρήστη. Συγκεκριμένα, μπορούμε να εισάγουμε στο σύστημα της ασαφής λογικής, και άλλα δεδομένα ως παράγοντες, όπως ο υπολογισμός των χρόνων που χρειάζεται ο χρήστης για να δημιουργεί μεγάλες και μικρές λέξεις, την ευκολία που δημιουργεί τις λέξεις αυτές, την συχνότητα δημιουργίας σπάνιων ή συνηθισμένων λέξεων, καθώς και άλλα πολλά.
- **Βελτίωση του μηχανισμού ζευγαρώματος τυχαίων αντιπάλων**. Όπως έχει αναλυθεί και στο κεφάλαιο 4 παράγραφος 4.7, για το ζευγάρωμα δυο τυχαίων χρηστών, το σύστημα λαμβάνει υπόψιν το επίπεδο τους. Έτσι μια μελλοντική βελτίωση ως προς το

ζευγάρωμα των αντίπαλων, θα επικεντρωνόταν και σε άλλα χαρακτηριστικά των δυο χρηστών, τα οποία θα αντλούνται από το ιστορικό των παιχνιδιών τους, όπως μέσος όρος πόντων, μέσος όρος λέξεων, μέσος όρος δημιουργίας λέξεων, φυσικά, με την βοήθεια ασαφής λογικής.

- **Παροχή μηχανισμού πρότασης φίλου. (Friend Suggestion)** Οι αλγόριθμοι πρότασης φίλων αποτελούν ένα πολύπλοκο σύστημα το οποίο χρησιμοποιεί διάφορους παράγοντες για να καθορίσει ποιους φίλους θα εμφανίσει στον χρήστη. Οι αλγόριθμοι αυτοί, είναι άκρως σημαντικοί σε εφαρμογές κοινωνικών δικτυών, αλλά τελευταία παρατηρείται χρήση τους και σε παιχνίδια πολλαπλών χρηστών και πραγματικού χρόνου, όπως είναι η εφαρμογή της παρούσας διπλωματικής.

# Βιβλιογραφία

Trie - Wikipedia

SHA-1 - Wikipedia

Database - Wikipedia

Software - Wikipedia

WebSocket - Wikipedia

Fuzzy logic - Wikipedia

Defuzzification - Wikipedia

Cloud computing - Wikipedia

Scripting language - Wikipedia

RSA (cryptosystem) - Wikipedia

Hybrid cryptosystem - Wikipedia

Procedural programming - Wikipedia

Functional programming - Wikipedia

Java (programming language) - Wikipedia

Object-oriented programming - Wikipedia

Advanced Encryption Standard - Wikipedia

https://developer.android.com/guide/platform

https://developer.android.com/guide/components/services

https://developer.android.com/guide/topics/ui/layout/linear

https://developer.android.com/guide/topics/ui/layout/relative

https://developer.android.com/reference/android/app/Activity

https://developer.android.com/guide/components/intents-filters

https://developer.android.com/guide/components/fundamentals

https://developer.android.com/guide/topics/manifest/manifest-intro

https://developer.android.com/reference/android/widget/GridLayout

https://developer.android.com/guide/components/foreground-services

https://developer.android.com/reference/android/widget/FrameLayout

https://developer.android.com/guide/components/activities/activity-lifecycle

https://firebase.google.com/docs/auth

https://firebase.google.com/docs/database

https://firebase.google.com/docs/database/rest/start

https://firebase.google.com/docs/functions/http-events

https://firebase.google.com/docs/database/rest/save-data

https://firebase.google.com/docs/database/rest/structure-data

https://firebase.google.com/docs/database/android/lists-of-data

https://firebase.google.com/docs/database/android/structure-data

https://firebase.google.com/docs/database/android/read-and-write

https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference

https://firebase.google.com/docs/reference/android/com/google/firebase/database/DatabaseReference#push()

https://firebase.google.com/docs/reference/android/com/google/firebase/database/Query#addChildEventListener(com.google.firebase.database.ChildEventListener)

AIBook (uoi.gr)

ΑΣΑΦΗΣ ΛΟΓΙΚΗ (upatras.gr)

Παρουσίαση του PowerPoint (teipir.gr)

What is Authentication? (techtarget.com)

Android - UI Layouts (tutorialspoint.com)

Στοιχεία της Ασαφούς Λογικής (upatras.gr)

Fuzzy Logic | Introduction - GeeksforGeeks

Declarative vs imperative - DEV Community

What Is AES Encryption and How Does It Work? - Simplilearn

In-Game Chat and Its Importance for Online Gaming • QuickBlox

Level systems and character growth in RPG games - Pav Creations

In-Game Chat 101 - Text, Voice, & Video Messaging (getstream.io)

Benefits of Multilingual App and How to Create Them | CustomerThink

https://repository.kallipos.gr/bitstream/11419/5956/2/02_chapter_02.pdf

Firebase vs WebSocket: Differences and how they work together | Ably Realtime

Google Firebase - a Complete Back-End Solution for Mobile and Web (infoq.com)

Encrypted Messaging – What Is It, And Why Should You Use It? (pixelprivacy.com)