Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**«Ανάπτυξη Εφαρμογής Ανταλλαγής Μηνυμάτων για Android με Χρήση Java και Firebase»** "Developing a Messaging Application for Android Using Java and Firebase"
Ονοματεπώνυμο Φοιτητή	ΔΙΟΝΥΣΗΣ ΚΑΤΣΙΓΙΑΝΝΗΣ
Πατρώνυμο	ΔΗΜΗΤΡΙΟΣ
Αριθμός Μητρώου	ΜΠΠΛ21028
Επιβλέπων	Ευθύμιος Αλέπης

Ημερομηνία Παράδοσης **Νοέμβριος 2025**

Τριμελής Εξεταστική Επιτροπή

Αλέπης Ευθύμιος Καθηγητής	Βίρβου Μαρία Καθηγήτρια	Σωτηρόπουλος Διονύσιος Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Η παρούσα μεταπτυχιακή εργασία παρουσιάζει την ανάπτυξη μιας εφαρμογής ανταλλαγής μηνυμάτων σε πραγματικό χρόνο (real-time messaging application) για περιβάλλον Android, με τη χρήση της γλώσσας προγραμματισμού Java και των υπηρεσιών cloud της πλατφόρμας Firebase. Στόχος της εργασίας είναι η υλοποίηση μιας λειτουργικής, ασφαλούς και εύχρηστης εφαρμογής επικοινωνίας, η οποία επιτρέπει στους χρήστες να δημιουργούν λογαριασμό, να συνδέονται και να ανταλλάσσουν μηνύματα σε πραγματικό χρόνο, αξιοποιώντας σύγχρονες τεχνολογίες mobile ανάπτυξης.

Στο θεωρητικό μέρος εξετάζονται οι τεχνολογίες που χρησιμοποιήθηκαν, όπως το Android Framework, το Firebase Authentication και το Firebase Realtime Database, καθώς και ο τρόπος με τον οποίο αυτές υποστηρίζουν την ανάπτυξη cloud-based mobile εφαρμογών. Παρουσιάζεται επίσης η αρχιτεκτονική της εφαρμογής, η δομή των activities, η οργάνωση των adapters και η γενικότερη λογική ροής δεδομένων. Ιδιαίτερη έμφαση δίνεται στη δημιουργία ενός αξιόπιστου μηχανισμού real-time επικοινωνίας μεταξύ χρηστών, με χρήση listeners και αμφίδρομων “chat rooms” (senderRoom και receiverRoom).

Στο πρακτικό μέρος περιγράφονται αναλυτικά η υλοποίηση του κώδικα, τα επιμέρους τμήματα της εφαρμογής και η μεθοδολογία ανάπτυξης που ακολουθήθηκε. Η εφαρμογή αξιολογείται σε επίπεδο απόδοσης, χρηστικότητας και σταθερότητας. Τα αποτελέσματα δείχνουν πως η χρήση του Firebase παρέχει υψηλή ταχύτητα συγχρονισμού, απλότητα διαχείρισης δεδομένων και αξιόπιστες real-time λειτουργίες χωρίς την ανάγκη συντήρησης παραδοσιακού backend.

Τέλος, παρουσιάζονται οι δυνατότητες μελλοντικής επέκτασης της εφαρμογής, όπως η υποστήριξη πολυμέσων, οι push notifications, η βελτίωση του UX, οι ομαδικές συνομιλίες και μηχανισμοί όπως online status και read receipts. Συνολικά, η εργασία αναδεικνύει την αποτελεσματικότητα των τεχνολογιών Java, Android και Firebase στην ανάπτυξη σύγχρονων, επεκτάσιμων και cloud-based mobile εφαρμογών.

ABSTRACT

This postgraduate thesis presents the development of a real-time messaging application for the Android platform, implemented using the Java programming language and the cloud services provided by Firebase. The primary objective of this work is to design and build a functional, secure, and user-friendly communication system that enables users to register, authenticate, and exchange messages instantly, leveraging modern mobile development technologies.

The theoretical part examines the technologies employed, including the Android Framework, Firebase Authentication, and Firebase Realtime Database, as well as the way these components support the development of cloud-based mobile applications. The system architecture is thoroughly analyzed, covering the structure of activities, the role of adapters, and the overall logic of data exchange within the application. Special emphasis is placed on implementing a reliable real-time messaging mechanism using Firebase listeners and dual “chat room” structures (senderRoom and receiverRoom).

The practical part provides a detailed explanation of the implementation process, the code structure, and the development methodology followed throughout the project. The application is evaluated in terms of performance, usability, and stability. Results indicate that Firebase offers high synchronization speed, simplified data handling, and robust real-time communication without requiring a traditional backend infrastructure.

Finally, potential future enhancements are discussed, including multimedia support, push notifications, improved UX design, group chat functionality, and advanced communication features such as online status and read receipts. Overall, this thesis demonstrates the effectiveness of Java, Android, and Firebase technologies in developing modern, scalable, and cloud-driven mobile applications.

Πίνακας περιεχομένων

1. Εισαγωγή

Η ραγδαία ανάπτυξη των κινητών συσκευών και η καθολική πλέον παρουσία τους στην καθημερινότητα έχουν μεταμορφώσει ριζικά τον τρόπο με τον οποίο οι άνθρωποι αλληλεπιδρούν, εργάζονται, πληροφορούνται και επικοινωνούν. Στο κέντρο αυτής της τεχνολογικής επανάστασης βρίσκονται οι εφαρμογές άμεσης ανταλλαγής μηνυμάτων, οι οποίες έχουν εξελιχθεί σε βασικό εργαλείο προσωπικής και επαγγελματικής επικοινωνίας. Η τάση αυτή δεν σχετίζεται μόνο με την τεχνολογική ευκολία που παρέχουν τα smartphones, αλλά και με βαθύτερες κοινωνικές μεταβολές που αφορούν την ανάγκη για ταχύτητα, αμεσότητα και ευκολία στην ανταλλαγή πληροφοριών.

Οι χρήστες σήμερα αναμένουν από μία εφαρμογή επικοινωνίας να προσφέρει απλό περιβάλλον χειρισμού, ταχύτητα απόκρισης και ασφάλεια στη διαχείριση των προσωπικών τους δεδομένων. Ειδικότερα σε πληθυσμούς που δυσκολεύονται να ξεκινήσουν κοινωνική αλληλεπίδραση όπως ντροπαλά άτομα με κοινωνική αμηχανία ή έλλειψη αυτοπεποίθησης ,οι εφαρμογές ανταλλαγής μηνυμάτων λειτουργούν συχνά ως μια «γέφυρα» που μειώνει τον βαθμό έκθεσης και καθιστά την επικοινωνία πιο άνετη. Στο πλαίσιο αυτό, μια εφαρμογή chat με καθαρή δομή, κατανοητή διεπαφή και άμεση λειτουργία μπορεί να παίξει σημαντικό ρόλο στην ενίσχυση της κοινωνικής δραστηριότητας του χρήστη.

Η παρούσα μεταπτυχιακή διατριβή εστιάζει στην ανάλυση, σχεδίαση και υλοποίηση μιας εφαρμογής ανταλλαγής μηνυμάτων για το λειτουργικό σύστημα Android, χρησιμοποιώντας ως βασικά τεχνολογικά εργαλεία τη γλώσσα προγραμματισμού **Java** και την πλατφόρμα cloud υπηρεσιών **Firebase**. Η εφαρμογή δίνει τη δυνατότητα σε χρήστες να δημιουργήσουν λογαριασμό, να συνδεθούν με ασφάλεια και να ανταλλάξουν μεταξύ τους μηνύματα σε πραγματικό χρόνο, αξιοποιώντας την *Firebase Realtime Database* και το *Firebase Authentication*. Η επιλογή αυτών των τεχνολογιών προέκυψε μέσα από ένα συνδυασμό κριτηρίων που σχετίζονται με την αξιοπιστία, την ευχρηστία, την ταχύτητα υλοποίησης και τη σύγχρονη πρακτική της ανάπτυξης mobile εφαρμογών.

Η Java παραμένει μέχρι σήμερα η πιο διαδεδομένη γλώσσα ανάπτυξης Android εφαρμογών, με ώριμη υποδομή, μεγάλη κοινότητα υποστήριξης και πλούσια βιβλιοθήκη εργαλείων. Παρότι τα τελευταία χρόνια το Kotlin έχει επίσης καθιερωθεί, η Java εξακολουθεί να αποτελεί μια ασφαλή, σταθερή και αποτελεσματική επιλογή για εφαρμογές που απαιτούν ξεκάθαρη δομή και αυστηρή διαχείριση δεδομένων. Το Android Studio, ως επίσημη ανάπτυξη περιβάλλοντος της Google, προσφέρει ολοκληρωμένες δυνατότητες όπως debugging, emulation, διαχείριση πόρων και υποστήριξη για responsive σχεδιασμό.

Παράλληλα, το Firebase αποτελεί πλέον μια από τις πιο δημοφιλείς λύσεις backend για mobile εφαρμογές, προσφέροντας λειτουργίες authentication, αποθήκευσης δεδομένων, analytics και real-time συγχρονισμού χωρίς την ανάγκη διαχείρισης servers. Η Realtime Database επιτρέπει την άμεση ανταλλαγή δεδομένων μεταξύ δύο χρηστών, ενώ η αρχιτεκτονική της εφαρμογής βασίζεται σε «δωμάτια συνομιλίας» (chat rooms) που δημιουργούνται δυναμικά με βάση τους συνδυασμούς χρηστών. Με αυτόν τον τρόπο, η επικοινωνία πραγματοποιείται ταχύτατα και αποδοτικά, χωρίς περίπλοκες δομές.

Η ανάπτυξη της εφαρμογής ακολούθησε σύγχρονες αρχές σχεδιασμού (mobile UX guidelines) και λογικές modular κώδικα που διευκολύνουν τη συντήρηση και την επεκτασιμότητα. Η εφαρμογή περιλαμβάνει κλάσεις (Activities) όπως το *RegisterActivity*, το *LoginActivity*, το *MainActivity* και το *ChatActivity*, οι οποίες επικοινωνούν με το Firebase μέσω των αντίστοιχων listeners και handlers. Κάθε activity έχει σαφή και διακριτό ρόλο στο συνολικό σύστημα, εξασφαλίζοντας τον καθαρό διαχωρισμό ευθυνών και τη σταθερότητα της εφαρμογής.

Πέρα από το τεχνικό κομμάτι, σημαντική πτυχή της εργασίας ήταν ο σχεδιασμός μιας εφαρμογής που διατηρεί μια φυσική απλότητα, χωρίς περιττές λειτουργίες ή περίπλοκες επιλογές που θα επιβάρυναν τον τελικό χρήστη. Στόχος δεν ήταν η δημιουργία ενός πλήρους commercial messenger, αλλά ενός αποτελεσματικού και καθαρού εργαλείου επικοινωνίας που να αναδεικνύει την τεχνολογική υποδομή και τη δυνατότητα real-time συγχρονισμού.

Η εισαγωγή αυτή σκιαγραφεί τα βασικά κίνητρα, το πλαίσιο και τις τεχνολογικές επιλογές που οδήγησαν στην ανάπτυξη της εφαρμογής. Στις επόμενες ενότητες της διατριβής παρουσιάζεται το τεχνολογικό υπόβαθρο και εργαλεία ανάπτυξης που χρησιμοποιήθηκαν, η μεθοδολογία ανάπτυξης, η αναλυτική αρχιτεκτονική του συστήματος, τα activities και ο κώδικας που υλοποιήθηκε, καθώς και τα συμπεράσματα που προέκυψαν κατά τη διαδικασία σχεδίασης και υλοποίησης.

2. Σκοπός και Στόχοι της Εργασίας

Η παρούσα μεταπτυχιακή εργασία έχει ως κεντρικό σκοπό την ανάπτυξη, ανάλυση και αξιολόγηση μιας εφαρμογής ανταλλαγής μηνυμάτων για το λειτουργικό σύστημα Android, βασισμένης στη γλώσσα προγραμματισμού Java και στις cloud υπηρεσίες της πλατφόρμας Firebase. Η εργασία επιδιώκει να διερευνήσει και να παρουσιάσει ολοκληρωμένα τη διαδικασία σχεδιασμού μιας σύγχρονης mobile εφαρμογής, από τα πρώτα στάδια ανάλυσης των απαιτήσεων μέχρι την τελική υλοποίηση και δοκιμή, δίνοντας έμφαση στην τεχνική ορθότητα και στη σαφή αρχιτεκτονική δομή.

Κύρια επιδίωξη είναι η δημιουργία μιας απλής, αξιόπιστης και λειτουργικής εφαρμογής που επιτρέπει σε χρήστες να ανταλλάσσουν μηνύματα σε πραγματικό χρόνο, χωρίς καθυστερήσεις, με ασφάλεια και με τρόπο εύκολα κατανοητό ακόμη και από χρήστες χωρίς ιδιαίτερη εξοικείωση με τις σύγχρονες ψηφιακές τεχνολογίες. Η υλοποίηση της συγκεκριμένης εφαρμογής δεν στοχεύει στην ανταγωνιστική δημιουργία μιας εμπορικής πλατφόρμας επικοινωνίας, αλλά στη μελέτη των τεχνολογιών που την υποστηρίζουν και στη δημιουργία ενός ολοκληρωμένου παραδείγματος υλοποίησης που αναδεικνύει τις δυνατότητες του Android και του Firebase.

2.1 Σκοπός της Εργασίας

Ο βασικός σκοπός της εργασίας είναι:
Η ανάπτυξη μιας λειτουργικής, απλής και ασφαλούς εφαρμογής άμεσης ανταλλαγής μηνυμάτων, η οποία αξιοποιεί Firebase Authentication και Firebase Realtime Database για την υλοποίηση real-time επικοινωνίας μεταξύ χρηστών σε περιβάλλον Android.

Ο σκοπός αυτός ενσωματώνει την ανάγκη για:

- αποτελεσματική οργάνωση του κώδικα,
- σωστή αρχιτεκτονική client–backend,
- ασφαλή διαχείριση των δεδομένων,
- καθαρό και απλό UI,
- εμπειρία χρήσης χωρίς περιττή πολυπλοκότητα.

Παράλληλα, η εργασία επιδιώκει να αναδείξει τις δυνατότητες των τεχνολογιών που χρησιμοποιήθηκαν, καθώς και τις προκλήσεις που προκύπτουν σε εφαρμογές real-time επικοινωνίας.

2.2 Γενικοί Στόχοι της Εργασίας

Για να επιτευχθεί ο παραπάνω σκοπός, τέθηκαν οι ακόλουθοι γενικοί στόχοι:

1. **Μελέτη των τεχνολογιών Android, Java και Firebase**, με έμφαση στην πρακτική αξιοποίησή τους σε mobile εφαρμογές.
2. **Σχεδίαση και υλοποίηση ενός backend μέσω Firebase**, ικανού να υποστηρίξει real-time επικοινωνία με σταθερότητα και συνέπεια.
3. **Υλοποίηση ενός ασφαλούς συστήματος authentication**, που επιτρέπει εγγραφή και σύνδεση χρηστών.
4. **Ανάπτυξη καθαρής και χρηστικής διεπαφής χρήστη**, προσανατολισμένης στην απλότητα.
5. **Ανάλυση αρχιτεκτονικής της εφαρμογής**, με τεχνική τεκμηρίωση των activities, των μοντέλων δεδομένων και της ροής πληροφοριών.
6. **Καταγραφή και αξιολόγηση της απόδοσης**, της χρηστικότητας και των περιορισμών της εφαρμογής.
7. **Τεκμηρίωση της διαδικασίας ανάπτυξης**, ώστε η εργασία να λειτουργεί και ως οδηγός για παρόμοιες υλοποιήσεις.

2.3 Ειδικοί και Τεχνικοί Στόχοι

Εκτός από τους γενικούς στόχους, καθορίστηκαν και πιο συγκεκριμένοι, τεχνικής φύσεως στόχοι:

Τεχνικοί Στόχοι

- Υλοποίηση **RegisterActivity** με χρήση Firebase Authentication.
- Υλοποίηση **LoginActivity** με πραγματικό έλεγχο διαπιστευτηρίων μέσω Firebase.
- Σχεδίαση **MainActivity** που ανακτά και εμφανίζει σε πραγματικό χρόνο τους εγγεγραμμένους χρήστες.
- Δημιουργία **ChatActivity**, στο οποίο τα μηνύματα καταχωρούνται και εμφανίζονται σε real-time μορφή μέσω Firebase listeners.
- Διαχωρισμός των chat rooms σε **senderRoom** και **receiverRoom**, σύμφωνα με την αρχιτεκτονική του Firebase.
- Υλοποίηση μοντέλων δεδομένων (User, Message) και κατάλληλων adapters (MessageAdapter, UserAdapter).
- Ορθή διασύνδεση UI και backend με βάση καλές πρακτικές Android development.

Στόχοι Αξιολόγησης

- Έλεγχος της ταχύτητας συγχρονισμού των μηνυμάτων.
- Έλεγχος αξιοπιστίας της Realtime Database.

- Δοκιμή της ευχρηστίας της διεπαφής από διαφορετικούς τύπους χρηστών.
- Εντοπισμός περιορισμών και πιθανών βελτιώσεων.

2.4 Συνολική Σημασία των Στόχων

Η επίτευξη των παραπάνω στόχων οδηγεί:

- στη δημιουργία μιας πλήρους και δομημένης mobile εφαρμογής,
- στη βαθιά κατανόηση του τρόπου με τον οποίο το Firebase υποστηρίζει real-time εφαρμογές,
- στην εξοικείωση με τις βασικές αρχές ανάπτυξης Android εφαρμογών,
- στη διαμόρφωση ενός κώδικα που ακολουθεί σαφή και επεκτάσιμη αρχιτεκτονική.

Με αυτόν τον τρόπο, η εργασία συμβάλλει τόσο στην πρακτική ανάπτυξη όσο και στη θεωρητική κατανόηση της mobile επικοινωνίας.

3. Τεχνολογικό Υπόβαθρο και Εργαλεία Ανάπτυξης

3.1 Η Γλώσσα Προγραμματισμού Java

Η Java αποτελεί μία από τις σημαντικότερες και πιο επιδραστικές γλώσσες προγραμματισμού της σύγχρονης εποχής. Δημιουργήθηκε από την εταιρεία Sun Microsystems το 1995 με στόχο να προσφέρει μια πλατφόρμα που θα επέτρεπε τη δημιουργία φορητών, ανεξάρτητων από το λειτουργικό σύστημα εφαρμογών. Το σύνθημα «Write Once, Run Anywhere» υπήρξε θεμελιώδης αρχή σχεδιασμού της, καθώς η Java βασίζεται στην εκτέλεση κώδικα μέσω της Java Virtual Machine (JVM) αντί της άμεσης μετάφρασης σε μηχανικό κώδικα.

Η δημοτικότητά της οφείλεται στην αντικειμενοστραφή φιλοσοφία της, στην αυστηρά τυποποιημένη σύνταξη, στη σταθερότητα και στην ασφάλεια που προσφέρει. Καθώς η ανάπτυξη λογισμικού μεγάλων επιχειρήσεων αυξανόταν δραστικά στα τέλη της δεκαετίας του 1990 και στις αρχές του 2000, η Java εδραιώθηκε ως πρότυπο για enterprise εφαρμογές, web servers, τραπεζικά συστήματα και μεγάλης κλίμακας πλατφόρμες.

Η σημασία της Java για τον χώρο του mobile software έγινε ακόμη μεγαλύτερη με την εμφάνιση του Android. Το Android υιοθέτησε αρχικά μια JVM-like υποδομή (Dalvik VM και αργότερα ART), επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τις γνώσεις τους στη Java για την ανάπτυξη εφαρμογών για κινητές συσκευές. Με αυτόν τον τρόπο, η Java διατηρεί έως σήμερα μια κυρίαρχη θέση στη mobile ανάπτυξη, παρά την παράλληλη άνοδο της Kotlin τα τελευταία χρόνια.

Ενδεικτικά χαρακτηριστικά της Java που ευνοούν την ανάπτυξη Android εφαρμογών:

- **Αντικειμενοστραφής σχεδιασμός (OOP):** βοηθά στη δομή των activities και των models.
- **Μεγάλη βιβλιοθήκη έτοιμων κλάσεων:** ιδανική για διαχείριση δεδομένων, συλλογών και δικτύωσης.
- **Στατική τυποποίηση:** μειώνει σημαντικά τα runtime errors.
- **Ωριμότητα και σταθερότητα:** θεμελιώδης σε εφαρμογές που απαιτούν αξιοπιστία.
- **Μεγάλη κοινότητα:** άφθονοι πόροι, παραδείγματα και λύσεις προβλημάτων.

Η εφαρμογή που υλοποιήθηκε στην παρούσα εργασία βασίζεται εξολοκλήρου στη Java, αξιοποιώντας ευρέως έννοιες όπως classes, inheritance, adapters, models και event listeners, στοιχεία που αποτελούν βασικά δομικά στοιχεία μιας Android εφαρμογής.

3.2 Το Λειτουργικό Σύστημα Android και το Android Studio

Το Android αποτελεί το πιο διαδεδομένο λειτουργικό σύστημα για κινητές συσκευές παγκοσμίως, με ποσοστά χρήσης που ξεπερνούν το 70% διεθνώς. Η πρώτη του έκδοση παρουσιάστηκε το 2008 από την Google, και από τότε εξελίσσεται με εντυπωσιακό ρυθμό, εισάγοντας συνεχώς βελτιώσεις στον τομέα του UI/UX, της ασφάλειας και της διαχείρισης πόρων.

Σε τεχνικό επίπεδο, το Android βασίζεται σε έναν τροποποιημένο πυρήνα Linux και χρησιμοποιεί μια αρχιτεκτονική που στηρίζεται στα layers: Linux Kernel, Hardware Abstraction Layer, Android Runtime, Libraries και τέλος το Application Framework, το οποίο επιτρέπει την ανάπτυξη των εφαρμογών από τους προγραμματιστές.

Το **Android Studio** αποτελεί το επίσημο IDE για την ανάπτυξη Android εφαρμογών. Παρέχει ένα πλήθος εργαλείων:

- **Εξομοιωτές κινητών (Android Emulator)**
- **Ενσωματωμένο debugger**
- **Εργαλεία προεπισκόπησης UI**
- **Διαχείριση APK και builds**
- **Σύστημα Gradle για αυτοματοποίηση**
- **Εργαλεία profiling για μνήμη, CPU και δίκτυο**

Η δομή μιας Android εφαρμογής βασίζεται κυρίως στις ακόλουθες έννοιες:

- **Activities:** οθόνες αλληλεπίδρασης με τον χρήστη
- **Intents:** μηχανισμός επικοινωνίας μεταξύ components
- **Layouts:** περιγραφή του γραφικού περιβάλλοντος
- **Adapters:** σύνδεση δεδομένων με views όπως RecyclerView
- **Lifecycle methods:** onCreate, onStart, onPause κ.λπ.

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της εργασίας αξιοποιεί πλήρως τη λογική των activities, των intents και του Android lifecycle, με καθαρή διασύνδεση μεταξύ των οθονών εγγραφής, σύνδεσης, λίστας χρηστών και συνομιλίας.

3.3 Η Πλατφόρμα Firebase – Υπηρεσίες και Ρόλος στην Εφαρμογή

Το Firebase δημιουργήθηκε το 2011 αρχικά ως πλατφόρμα real-time βάσεων δεδομένων και εξαγοράστηκε από τη Google το 2014. Έκτοτε εξελίχθηκε σε μια ολοκληρωμένη σουίτα cloud υπηρεσιών για mobile και web εφαρμογές, προσφέροντας εργαλεία backend χωρίς την ανάγκη διαχείρισης servers.

Οι βασικές υπηρεσίες του Firebase που σχετίζονται με την εργασία είναι:

1. Firebase Authentication

Παρέχει μεθόδους εγγραφής και σύνδεσης χρηστών (email/password, OAuth providers κ.ά.).
Στην εφαρμογή χρησιμοποιείται η κλασική μέθοδος email–password.
Διασφαλίζει ότι μόνο επαληθευμένοι χρήστες μπορούν να συνδεθούν.

2. Firebase Realtime Database

Αποτελεί μια NoSQL βάση δεδομένων που ενημερώνεται σε πραγματικό χρόνο.
Η δομή της είναι δενδρική (JSON tree) και υποστηρίζει:

- real-time listeners
- push keys
- child nodes
- αυτόματο συγχρονισμό μεταξύ χρηστών

Στην εφαρμογή χρησιμοποιείται για:

- αποθήκευση λίστας χρηστών
- δημιουργία chat rooms (senderRoom / receiverRoom)
- αποθήκευση μηνυμάτων
- άμεση ενημέρωση της συνομιλίας χωρίς refresh

3. Firebase Storage (δυνητικά)

Αν και δεν χρησιμοποιείται ευρέως σε αυτή την έκδοση της εφαρμογής, μπορεί να αξιοποιηθεί για αποθήκευση φωτογραφιών προφίλ.

3.4 Συνολική Σύνδεση των Τεχνολογιών με την Εφαρμογή

Ο σχεδιασμός της εφαρμογής αξιοποιεί τα δυνατά σημεία κάθε τεχνολογίας:

- Η Java προσφέρει σταθερότητα, modular σχεδιασμό και ευκολία συντήρησης.
- Το Android Studio παρέχει το περιβάλλον ανάπτυξης και τα εργαλεία UI.
- Το Firebase λειτουργεί ως μια άμεση, cloud-based υποδομή για authentication και real-time messaging.

Το τεχνολογικό υπόβαθρο αποτελεί τη βάση για την κατανόηση του τρόπου με τον οποίο λειτουργεί η εφαρμογή και εξηγεί τις τεχνολογικές επιλογές που καθοδήγησαν την υλοποίηση.

4. Ιστορική Αναδρομή των Τεχνολογιών

Οι τεχνολογίες που αξιοποιήθηκαν για την ανάπτυξη της παρούσας εφαρμογής (java, Android και Firebase) διαθέτουν βαθιές ιστορικές ρίζες και έχουν καθορίσει σημαντικά την πορεία του λογισμικού και της κινητής ανάπτυξης. Η κατανόηση της ιστορικής εξέλιξης των τεχνολογιών αυτών είναι απαραίτητη, καθώς επιτρέπει την καλύτερη κατανόηση του γιατί αυτές οι πλατφόρμες έχουν επικρατήσει και πώς επηρεάζουν τη σύγχρονη ανάπτυξη εφαρμογών.

4.1 Η Ιστορική Εξέλιξη της Java

Η Java παρουσιάστηκε επίσημα το 1995 από την Sun Microsystems, σε μια εποχή όπου το λογισμικό ήταν στενά συνδεδεμένο με συγκεκριμένες πλατφόρμες και αρχιτεκτονικές. Η φιλοσοφία της «Write Once, Run Anywhere» αποτέλεσε επαναστατική προσέγγιση, επιτρέποντας στους προγραμματιστές να γράφουν κώδικα που μπορούσε να εκτελεστεί σε οποιοδήποτε σύστημα διέθετε Java Virtual Machine (JVM).

Στα τέλη της δεκαετίας του 1990, η Java έγινε ο ακρογωνιαίος λίθος των web server τεχνολογιών, χάρη στην εισαγωγή των Java Servlets, JSP και αργότερα του Java Enterprise Edition (JEE). Η σταθερότητα, η ασφάλεια και η επεκτασιμότητά της οδήγησαν μεγάλες εταιρείες όπως η IBM, η Oracle και η Google να τη χρησιμοποιήσουν σε κρίσιμες υποδομές.

Με την αγορά της Sun Microsystems από την Oracle το 2010, η Java πέρασε σε μια νέα εποχή. Παράλληλα, η διάθεσή της ως open-source μέσω του OpenJDK ενίσχυσε περαιτέρω τη διάδοση και υιοθέτησή της.

Η σχέση της Java με το Android ξεκίνησε το 2008, με την Google να επιλέγει τη γλώσσα ως τη βασική για την ανάπτυξη εφαρμογών στο νέο της λειτουργικό σύστημα. Η οικειότητα των προγραμματιστών με τη Java συνέβαλε καθοριστικά στη γρήγορη επέκταση του Android, καθώς εκατομμύρια developers μπορούσαν να μεταφέρουν εύκολα τις γνώσεις τους στο mobile software.

Ακόμη και σήμερα, παρά την άνοδο της Kotlin, η Java συνεχίζει να αποτελεί ισχυρή βάση για mobile ανάπτυξη, χάρη στη σταθερότητα, την ωριμότητα και την ευρεία βιβλιοθήκη της.

4.2 Η Εξέλιξη του Android

Το Android ξεκίνησε το 2003 ως ανεξάρτητη startup, με όραμα τη δημιουργία ενός έξυπνου λειτουργικού συστήματος για φορητές συσκευές. Η Google

εξαγόρασε την εταιρεία το 2005 και δύο χρόνια αργότερα παρουσίασε το Android στο κοινό, ως ένα ανοιχτό λειτουργικό σύστημα βασισμένο στον πυρήνα Linux.

Η πρώτη επίσημη συσκευή Android κυκλοφόρησε το 2008 (το HTC Dream), σηματοδοτώντας την απαρχή μιας νέας εποχής για τα smartphones. Η ανοιχτότητα του λειτουργικού επέτρεψε σε πολλούς κατασκευαστές να υιοθετήσουν γρήγορα την πλατφόρμα, οδηγώντας σε εκρηκτική ανάπτυξη μέσα σε λίγα χρόνια.

Σημαντικά σημεία της ιστορικής εξέλιξης του Android:

- **2008:** Έκδοση 1.0 - Πρώτη εμφάνιση market (προκάτοχος Play Store).
- **2010:** Android 2.2/2.3 -Υποστήριξη βελτιωμένων APIs για εφαρμογές και multitasking.
- **2012–2014:** Android 4.x - Περαιτέρω ενοποίηση UI με το Holo Design.
- **2014:** Έναρξη Material Design - νέο UI paradigm.
- **2017:** Εισαγωγή του Android Runtime (ART) αντί του Dalvik VM.
- **2019–σήμερα:** Ενίσχυση της ασφάλειας, των permissions και των cloud-based λειτουργιών.

Η σταθερή ανάπτυξη του Android Studio το οποίο παρουσιάστηκε το 2013 συνέβαλε σημαντικά στην επαγγελματική ανάπτυξη εφαρμογών. Το IDE βασίζεται στο IntelliJ IDEA και ενσωματώνει εργαλεία όπως Gradle, Emulator, Layout Inspector και Device Manager, καθιστώντας το βασικό εργαλείο κάθε Android προγραμματιστή.

Η ιστορική εξέλιξη του Android δείχνει μια συνεχόμενη προσπάθεια βελτίωσης της εμπειρίας χρήσης, της ταχύτητας και της ασφάλειας. Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της εργασίας επωφελείται από αυτή την ωριμότητα του οικοσυστήματος, αξιοποιώντας σύγχρονες βιβλιοθήκες και μηχανισμούς real-time επικοινωνίας.

4.3 Η Ιστορία και Εξέλιξη του Firebase

Το Firebase δημιουργήθηκε το 2011 από την εταιρεία Envolve ως εργαλείο real-time chat για web εφαρμογές. Πολύ γρήγορα, η εταιρεία παρατήρησε ότι αυτό που ενδιέφερε τους developers δεν ήταν η ίδια η υπηρεσία chat, αλλά η τεχνολογία real-time backend που υποστήριζε την ανταλλαγή δεδομένων.

Έτσι, το 2012 η Envolve μετατράπηκε σε Firebase, μια πλατφόρμα που παρείχε real-time database και APIs για εφαρμογές με live συγχρονισμό.

Το 2014, η Google απέκτησε το Firebase, ανοίγοντας τον δρόμο για μεγάλη τεχνολογική εξέλιξη και ενσωμάτωση στο Android οικοσύστημα. Μετά την εξαγορά, το Firebase εξελίχθηκε από «μια real-time βάση» σε μια ολοκληρωμένη πλατφόρμα cloud υπηρεσιών.

Σημαντικοί σταθμοί στην εξέλιξη του Firebase:

- **2014:** Εξαγορά από Google.
- **2016:** Μεγάλη ανανέωση - προστέθηκαν Authentication, Storage, Hosting, Cloud Messaging.
- **2017–2019:** Σταθεροποίηση των SDKs, ενσωμάτωση με Google Analytics.
- **2020–σήμερα:** Βελτίωση ασφάλειας μέσω Firebase Security Rules, αναβάθμιση Realtime Database και Firestore.

Σήμερα, το Firebase αποτελεί μία από τις πιο δημοφιλείς επιλογές backend για mobile εφαρμογές, χάρη σε χαρακτηριστικά όπως:

- real-time συγχρονισμός
- ασφάλεια
- μηδενική ανάγκη για server setup
- υψηλή κλιμάκωση
- εύκολο integration με Android

Στην εφαρμογή το Firebase χρησιμοποιείται σε δύο κεντρικούς άξονες:

1. **Authentication:** εγγραφή & σύνδεση χρηστών.
2. **Realtime Database:** άμεση αποστολή & λήψη μηνυμάτων, δημιουργία chat rooms.

Χωρίς την ύπαρξη του Firebase, η ανάπτυξη ενός τέτοιου συστήματος real-time επικοινωνίας θα απαιτούσε πολύπλοκο backend, servers και σύνθετη διαχείριση δικτύου.

4.4 Σημασία της Ιστορικής Εξέλιξης για την Εργασία

Η κατανόηση της ιστορικής πορείας των τεχνολογιών αυτών επιτρέπει:

- Την καταλληλότερη επιλογή εργαλείων,
- τη βαθύτερη κατανόηση των δυνατοτήτων τους,
- την ικανότητα αντιμετώπισης προβλημάτων,
- την υιοθέτηση βέλτιστων πρακτικών.

Η ωριμότητα της Java, η εξέλιξη του Android και η καινοτομία του Firebase αποτελούν το θεμέλιο πάνω στο οποίο στηρίζεται η εφαρμογή ανταλλαγής μηνυμάτων.

5. Μεθοδολογία Ανάπτυξης Λογισμικού

Η ανάπτυξη της εφαρμογής ανταλλαγής μηνυμάτων πραγματοποιήθηκε ακολουθώντας αρχές της ευέλικτης ανάπτυξης λογισμικού (Agile Software Development) και ειδικότερα της μεθοδολογίας **Scrum**. Η επιλογή της Agile προσέγγισης δεν ήταν τυχαία, πρόκειται για τη πλέον διαδεδομένη μεθοδολογία ανάπτυξης λογισμικού σε σύγχρονες mobile εφαρμογές, καθώς επιτρέπει γρήγορη προσαρμογή, συνεχείς επαναλήψεις και άμεση ανατροφοδότηση.

Η φύση της εφαρμογής ένα σύστημα real-time ανταλλαγής μηνυμάτων απαιτούσε σταδιακή υλοποίηση και συχνή δοκιμή των λειτουργιών, ώστε να διασφαλιστεί ότι ο συγχρονισμός, η σύνδεση στο Firebase και η εμπειρία χρήσης λειτουργούν ομαλά. Η μεθοδολογία Scrum κάλυψε ιδανικά αυτές τις ανάγκες.

5.1 Εισαγωγή στην Agile Μεθοδολογία

Η Agile μεθοδολογία εμφανίστηκε στις αρχές της δεκαετίας του 2000 ως αντίδραση στα παραδοσιακά μοντέλα ανάπτυξης, όπως το Waterfall, τα οποία ακολουθούσαν γραμμικές και άκαμπτες διαδικασίες. Με το Agile Manifesto (2001), τέθηκαν οι τέσσερις βασικές αξίες:

1. Άνθρωποι και αλληλεπιδράσεις πάνω από διαδικασίες και εργαλεία
2. Λειτουργικό λογισμικό πάνω από εκτενή τεκμηρίωση
3. Συνεργασία με τον πελάτη πάνω από διαπραγμάτευση συμβολαίων
4. Αντίδραση στην αλλαγή πάνω από τήρηση ενός σχεδίου

Τα Agile frameworks όπως Scrum, Kanban, XP δίνουν έμφαση:

- στη σταδιακή ανάπτυξη,
- στην ταχεία παράδοση λειτουργων εκδόσεων,
- στον συνεχή έλεγχο του προϊόντος,
- στην ευελιξία στις αλλαγές.

5.2 Η Μεθοδολογία Scrum

Η Scrum είναι το πιο διαδεδομένο Agile framework. Βασίζεται σε:

- σύντομα χρονικά διαστήματα ανάπτυξης (Sprints),
- καθορισμένους ρόλους,
- επαναληπτικό σχεδιασμό,
- συνεχή ανατροφοδότηση,
- σταδιακή βελτίωση του προϊόντος.

Κύρια χαρακτηριστικά Scrum:

- **Product Backlog:** λίστα με όλες τις λειτουργίες που πρέπει να υλοποιηθούν.
- **Sprint Backlog:** επιλεγμένες εργασίες για το επόμενο Sprint.
- **Sprints:** σύντομοι κύκλοι ανάπτυξης 1-4 εβδομάδων.
- **Increment:** λειτουργική έκδοση που προκύπτει στο τέλος κάθε Sprint.
- **Daily Scrum:** σύντομη καθημερινή συνάντηση.
- **Sprint Review:** αξιολόγηση της προόδου στο τέλος του Sprint.
- **Sprint Retrospective:** βελτίωση της διαδικασίας για το επόμενο Sprint.

5.3 Εφαρμογή της Scrum Μεθοδολογίας στην Παρούσα Εργασία

Η ανάπτυξη της εφαρμογής αυτού του project πραγματοποιήθηκε προσαρμοσμένη στο πλαίσιο Scrum, παρότι ο ρόλος του προγραμματιστή και του αναλυτή ταυτίζεται στο ίδιο άτομο. Η λογική του Scrum χρησιμοποιήθηκε για να οργανωθούν οι απαιτήσεις και να υπάρξει δομημένη πρόοδος.

5.3.1 Δημιουργία Product Backlog

Αρχικά ορίστηκε το σύνολο των βασικών λειτουργιών της εφαρμογής:

- Εγγραφή χρήστη (RegisterActivity)
- Σύνδεση χρήστη (LoginActivity)
- Ανάκτηση λίστας χρηστών σε real-time
- Αποστολή/λήψη μηνύματος
- Δημιουργία chat rooms στο Firebase
- Προβολή ιστορικού μηνυμάτων
- Διασύνδεση UI - backend
- Σχεδιασμός Activity layouts
- Έλεγχος ταυτότητας χρηστών
- Αναβάθμιση της εμπειρίας χρήστη (UI/UX)

5.3.2 Sprint Planning και Κατανομή Εργασιών

Η ανάπτυξη χωρίστηκε σε τέσσερα βασικά Sprints:

Sprint 1 – Βασική Υποδομή

- Ρύθμιση Android Studio
- Δημιουργία project
- Σύνδεση με Firebase
- Διαμόρφωση των dependencies (Authentication, Realtime Database)
- Βασική δομή Activities
- Σχεδιασμός UI για Register και Login

Sprint 2 – Authentication

- Υλοποίηση RegisterActivity
- Υλοποίηση LoginActivity
- Διαχείριση λαθών (π.χ. λάθος email/password)
- Έλεγχος ασφαλείας – Firebase Authentication Rules

Sprint 3 – Real-time Messaging

- Σχεδιασμός MainActivity (λίστα χρηστών)
- Ανάκτηση registered users από τη βάση
- Δημιουργία chat rooms (senderRoom, receiverRoom)
- Υλοποίηση ChatActivity
- Αποστολή μηνυμάτων
- Firebase listeners για real-time λήψη μηνυμάτων

Sprint 4 – Βελτιώσεις και Έλεγχοι

- Βελτίωση UI
- Διορθώσεις προβλημάτων
- Τεστ ταχύτητας συγχρονισμού
- Έλεγχος σταθερότητας της Realtime Database
- Συγκέντρωση κώδικα και τελικός καθαρισμός

5.4 Πλεονεκτήματα της Scrum Μεθοδολογίας στην Ανάπτυξη της Εφαρμογής

Η χρήση της μεθοδολογίας Scrum βοήθησε σημαντικά:

- **Οργάνωση εργασιών:** κάθε Sprint είχε ξεκάθαρους στόχους.
- **Ταχεία ανάπτυξη:** κάθε λειτουργία ολοκληρωνόταν πλήρως σε μικρούς κύκλους.
- **Συνεχής έλεγχος:** κάθε νέο κομμάτι κώδικα δοκιμαζόταν άμεσα.
- **Ευελιξία:** διορθώσεις, αλλαγές και βελτιώσεις γίνονταν άμεσα.
- **Αποφυγή τεχνικού χρέους:** ο κώδικας παρέμεινε καθαρός και οργανωμένος.

Η εφαρμογή messaging απαιτεί σταθερότητα, real-time συγχρονισμό και καθαρή αρχιτεκτονική. Η επαναληπτική φύση της Scrum μεθόδου βοήθησε στο να υλοποιηθούν αυτές οι απαιτήσεις χωρίς μεγάλη πολυπλοκότητα.

5.5 Συμπεράσματα Μεθοδολογίας

Η μεθοδολογία Agile/Scrum αποδείχθηκε κατάλληλη για την ανάπτυξη της εφαρμογής, καθώς:

- επέτρεψε σταδιακή και ελεγχόμενη υλοποίηση,
- έφερε γρήγορη πρόοδο χωρίς ανεξέλεγκτες αλλαγές κώδικα,
- προσέφερε καθαρή καταγραφή των απαιτήσεων,
- μείωσε το ρίσκο αστοχίας,
- επέτρεψε την εύκολη αξιολόγηση στο τέλος κάθε Sprint.

Η υλοποίηση ακολούθησε μια οργανωμένη ροή, οδηγώντας στην παραγωγή ενός σταθερού και λειτουργικού συστήματος άμεσης ανταλλαγής μηνυμάτων.

6. Τεχνολογίες Ανάπτυξης (Java, Android Studio, Firebase)

Η εφαρμογή ανταλλαγής μηνυμάτων που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας βασίζεται σε τρεις βασικούς τεχνολογικούς πυλώνες: τη γλώσσα προγραμματισμού **Java**, το περιβάλλον ανάπτυξης **Android Studio** και την πλατφόρμα cloud υπηρεσιών **Firebase**. Ο συνδυασμός αυτών των τεχνολογιών προσφέρει μια ισχυρή, αξιόπιστη και ευέλικτη βάση για την ανάπτυξη mobile εφαρμογών, ειδικά εφαρμογών που απαιτούν real-time επικοινωνία και ασφαλή διαχείριση δεδομένων.

6.1 Java – Η κύρια γλώσσα υλοποίησης

Η Java αποτελεί τη βασική γλώσσα προγραμματισμού για την παρούσα εφαρμογή. Η επιλογή της οφείλεται σε πολλούς λόγους:

6.1.1 Αντικειμενοστραφής Προσέγγιση (OOP)

Η εφαρμογή αξιοποιεί εντατικά αντικειμενοστραφείς έννοιες:

- **Class models** για χρήστες και μηνύματα (User, Message)
- **Adapters** για σύνδεση δεδομένων με Views (UserAdapter, MessageAdapter)
- **Encapsulation** σε properties των μοντέλων
- **Event listeners** και callbacks για real-time ενημερώσεις

Η OOP προσέγγιση εξασφαλίζει:

- καθαρό κώδικα,
- ευκολία επέκτασης,
- δυνατότητα επαναχρησιμοποίησης.

6.1.2 Διαχείριση Exceptions & Ασφάλεια

Η Java διαθέτει ισχυρό μηχανισμό exception handling, ο οποίος είναι κρίσιμος σε mobile εφαρμογές, ειδικά σε λειτουργίες όπως:

- σύνδεση στο Firebase,
- καταχώρηση δεδομένων,
- λανθασμένα credentials,

- ελλιπή πεδία στην εγγραφή.

Η σωστή χρήση try–catch μπλοκ μειώνει runtime σφάλματα και βελτιώνει τη σταθερότητα.

6.1.3 Πλούσιο οικοσύστημα βιβλιοθηκών

Η Java υποστηρίζεται από τεράστιο οικοσύστημα βιβλιοθηκών και Android APIs, τα οποία χρησιμοποιήθηκαν στην εφαρμογή, όπως:

- TextUtils
- FirebaseAuth
- FirebaseDatabase
- RecyclerView
- LinearLayoutManager

Η διαθεσιμότητα αυτών των εργαλείων επιτρέπει την ελαχιστοποίηση custom κώδικα.

6.2 Android Studio – Το περιβάλλον ανάπτυξης

Το Android Studio αποτελεί το επίσημο Integrated Development Environment (IDE) για Android εφαρμογές.

Χρησιμοποιήθηκε για:

- τη διαχείριση του project,
- τη δημιουργία XML layouts,
- το debugging,
- τη διαχείριση των Android dependencies,
- την αποσφαλμάτωση με τον emulator.

6.2.1 Δομή του Android Project

Το Android Studio οργανώνει τον κώδικα στις εξής βασικές δομές:

Activities (Java αρχεία)

Κύρια components που αντιπροσωπεύουν οθόνες.
Στην εφαρμογή υπάρχουν:

- RegisterActivity
- LoginActivity
- MainActivity
- ChatActivity

XML Layouts

Ορισμός UI κάθε οθόνης.
Μερικά βασικά XMLs:

- activity_register.xml
- activity_login.xml
- activity_main.xml
- activity_chat.xml

Adapters

Συνδέουν τα δεδομένα με RecyclerViews, π.χ.:

- **MessageAdapter:** εμφανίζει το ιστορικό συζήτησης
- **UserAdapter:** εμφανίζει τη λίστα χρηστών

Models (POJO classes)

- User.java
- Message.java

6.2.2 Το Build System – Gradle

Το Gradle αναλαμβάνει:

- διαχείριση βιβλιοθηκών (dependencies),
- δημιουργία APK,
- compilation.

Στη συγκεκριμένη εργασία προστέθηκαν dependencies για:

- Firebase Authentication
- Firebase Realtime Database
- RecyclerView
- Material Design Components

6.2.3 Android Emulator & Testing Tools

Ο emulator χρησιμοποίηθηκε για τη δοκιμή:

- της εγγραφής
- της σύνδεσης
- της real-time αποστολής μηνυμάτων
- της εμφάνισης λίστας χρηστών

Το Android Studio παρέχει επίσης:

- **Logcat** για debugging
- **Profiler** για κατανάλωση RAM / CPU

6.3 Firebase – Η πλατφόρμα backend της εφαρμογής

Το Firebase είναι ο πυρήνας της εφαρμογής messaging.
Αποτελεί έναν πλήρη backend που προσφέρει:

- authentication
- real-time βάσεις δεδομένων
- cloud hosting
- storage
- analytics

Στην εφαρμογή χρησιμοποιούνται κυρίως:

1. **Firebase Authentication**
2. **Firebase Realtime Database**

6.3.1 Firebase Authentication

Χρησιμοποιείται για:

- εγγραφή χρήστη με email & password
- έλεγχο διαπιστευτηρίων
- αποθήκευση δεδομένων χρήστη στο Firebase
- αποτροπή μη εξουσιοδοτημένης πρόσβασης

Τυπική ροή:

1. Ο χρήστης εισάγει email & password
2. Το FirebaseAuth.createUserWithEmailAndPassword() καταχωρεί τον χρήστη
3. Στη συνέχεια γίνεται login
4. Το UID του χρήστη χρησιμοποιείται ως primary key στη βάση

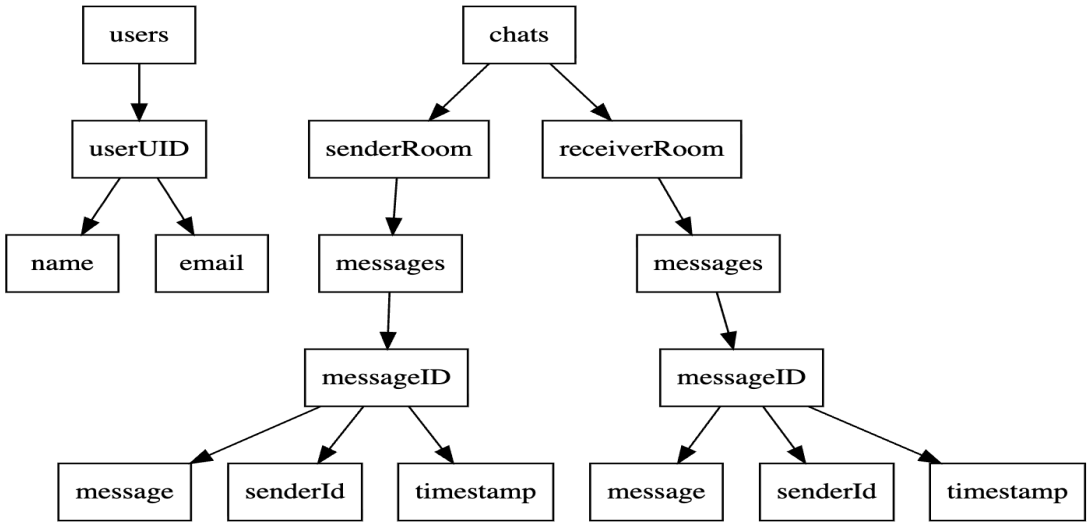
Το UID είναι κρίσιμο:
Το Firebase το χρησιμοποιεί ως μοναδικό αναγνωριστικό σε activities όπως MainActivity & ChatActivity.

6.3.2 Firebase Realtime Database

Η Realtime Database επιτρέπει:

- ζωντανή ενημέρωση μηνυμάτων
- listeners για αλλαγές κόμβων
- άμεση απεικόνιση νέων messages στον χρήστη χωρίς refresh
- cloud synchronization

Στη συγκεκριμένη εφαρμογή η δομή δεδομένων είναι:



SenderRoom / ReceiverRoom

Για δύο χρήστες A και B:

- senderRoom = AUID + BUID
- receiverRoom = BUID + AUID

Έτσι:

- Κάθε message αποθηκεύεται σε δύο rooms
- Το chat συγχρονίζεται real time και για τους δύο

6.3.3 Listeners για Real-Time Μηνύματα

Χρησιμοποιούνται μέθοδοι όπως:

databaseReference.addValueEventListener(...)

Οι listeners:

- ανιχνεύουν νέα μηνύματα
- ενημερώνουν άμεσα το RecyclerView
- ανανεώνουν το UI σε πραγματικό χρόνο

6.4 Ο Συνδυασμός των Τεχνολογιών στην Εφαρμογή

Ο συνδυασμός Java + Android Studio + Firebase επιτρέπει:

- σταθερή και modular αρχιτεκτονική
- ασφαλή διαχείριση χρηστών
- real-time λειτουργικότητα χωρίς server setup
- επεκτασιμότητα (π.χ. εικόνες, ειδοποιήσεις, προφίλ)
- καθαρό separation of concerns μεταξύ activities, adapters και backend

Συνολικά, η τεχνολογική βάση της εφαρμογής χαρακτηρίζεται από:

- απλότητα υλοποίησης,
- σταθερό backend,
- δυνατότητα μελλοντικής επέκτασης.

7. Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική της εφαρμογής ανταλλαγής μηνυμάτων βασίζεται σε ένα καθαρό, modular μοντέλο που διαχωρίζει τις λειτουργίες της διεπαφής χρήστη, της λογικής των activities και της επικοινωνίας με το backend. Η δομή του συστήματος έχει σχεδιαστεί ώστε να υποστηρίζει real-time επικοινωνία, ασφάλεια, επεκτασιμότητα και διατήρηση συνεκτικού κώδικα.

Η εφαρμογή αποτελείται από δύο βασικά επίπεδα:

1. **Το Client Layer (Android App – Java Activities)**
2. **Το Backend Layer (Firebase Authentication & Firebase Realtime Database)**

Η επικοινωνία μεταξύ των δύο επιπέδων γίνεται μέσω ασφαλών Firebase APIs και live listeners που επιτρέπουν την ακαριαία συγχρονισμένη προβολή μηνυμάτων μεταξύ δύο χρηστών.

7.1 Γενική Αρχιτεκτονική

Η γενική αρχιτεκτονική της εφαρμογής μπορεί να περιγραφεί ως ένα σύστημα **Client–Cloud** όπου ολόκληρη η λογική της αποθήκευσης και διαχείρισης δεδομένων βρίσκεται στο Firebase, ενώ το Android app λειτουργεί ως client που εκτελεί λειτουργίες:

- δημιουργίας χρήστη
- σύνδεσης
- ανάκτησης δεδομένων
- αποστολής μηνυμάτων
- εμφάνισης real-time συνομιλίας

Τα βασικά components του συστήματος είναι:

1. **Activities (UI + Logic):**
    - RegisterActivity
    - LoginActivity
    - MainActivity
    - ChatActivity
2. **Adapters:**
    - UserAdapter (Εμφάνιση λίστας χρηστών)
    - MessageAdapter (Εμφάνιση συνομιλίας)

3. **Data Models:**

    - User
    - Message

4. **Firebase Services:**

    - Authentication
    - Realtime Database

Αυτά τα components συνεργάζονται με συγκεκριμένη σειρά ώστε να επιτευχθεί μια πλήρης εμπειρία επικοινωνίας.

7.2 Client-Side Αρχιτεκτονική (Android)

Το client-side μέρος της εφαρμογής οργανώνεται σε Activities, όπου κάθε Activity αντιπροσωπεύει μια οθόνη και μια βασική λειτουργία.

7.2.1 RegisterActivity

- Συλλέγει email & password
- Επικοινωνεί με το FirebaseAuth για δημιουργία νέου χρήστη
- Μετά την επιτυχή εγγραφή, αποθηκεύει στοιχεία χρήστη στη Realtime Database
- Ανακατευθύνει τον νέο χρήστη στη LoginActivity

7.2.2 LoginActivity

- Χειρίζεται τη διαδικασία σύνδεσης
- Ελέγχει διαπιστευτήρια μέσω FirebaseAuth
- Σε επιτυχία μεταφέρει τον χρήστη στο MainActivity
- Περιλαμβάνει αρχική μορφή validation

7.2.3 MainActivity

- Ανακτά όλους τους εγγεγραμμένους χρήστες από το Firebase
- Τους εμφανίζει σε RecyclerView μέσω του UserAdapter
- Επιτρέπει την επιλογή χρήστη για συνομιλία
- Δημιουργεί τα senderRoom & receiverRoom keys
- Προωθεί τον χρήστη στο ChatActivity

7.2.4 ChatActivity

- Εμφανίζει το ιστορικό μηνυμάτων
- Συνδέεται σε listeners για real-time ενημέρωση
- Στέλνει νέα μηνύματα στο Firebase
- Ανανεώνει άμεσα το UI μέσω RecyclerView

Το ChatActivity αποτελεί το πιο σύνθετο component, καθώς απαιτεί συνεχή live σύνδεση με το Firebase.

7.3 Backend Αρχιτεκτονική – Firebase

Το Firebase λειτουργεί ως ολόκληρο backend της εφαρμογής.

7.3.1 Authentication Layer

Χρησιμοποιείται για:

- εγγραφή χρήστη
- σύνδεση χρήστη
- ταυτοποίηση μέσω UID
- αντιστοίχιση μηνυμάτων με χρήστες

Το UID κάθε χρήστη αποτελεί βασικό identifier στο σύστημα.

7.3.2 Realtime Database Layer

Η βάση είναι δενδρικής μορφής και αποτελείται από δύο κύριους κόμβους:

1. **users**

Ο κόμβος **users** περιέχει τα στοιχεία όλων των εγγεγραμμένων χρηστών. Κάθε χρήστης αποθηκεύεται κάτω από το μοναδικό του **UID** (User ID), το οποίο παρέχεται αυτόματα από το Firebase Authentication. Για κάθε χρήστη καταγράφονται:

- **uid**: μοναδικός αναγνωριστικός αριθμός του χρήστη
- **name**: το όνομα που δήλωσε κατά την εγγραφή
- **email**: η ηλεκτρονική διεύθυνση του χρήστη

Τα δεδομένα αυτά αξιοποιούνται από την εφαρμογή τόσο για την προβολή των στοιχείων σε λίστες συνομιλιών όσο και για τον προσδιορισμό της ταυτότητας του αποστολέα σε κάθε μήνυμα.

2. **chats**

Ο κόμβος **chats** είναι υπεύθυνος για την αποθήκευση όλων των μηνυμάτων. Για κάθε ζεύγος χρηστών δημιουργούνται δύο «δωμάτια συνομιλιών»:

- **senderRoom**: το δωμάτιο όπως το “βλέπει” ο αποστολέας
- **receiverRoom**: το αντίστοιχο δωμάτιο για τον παραλήπτη

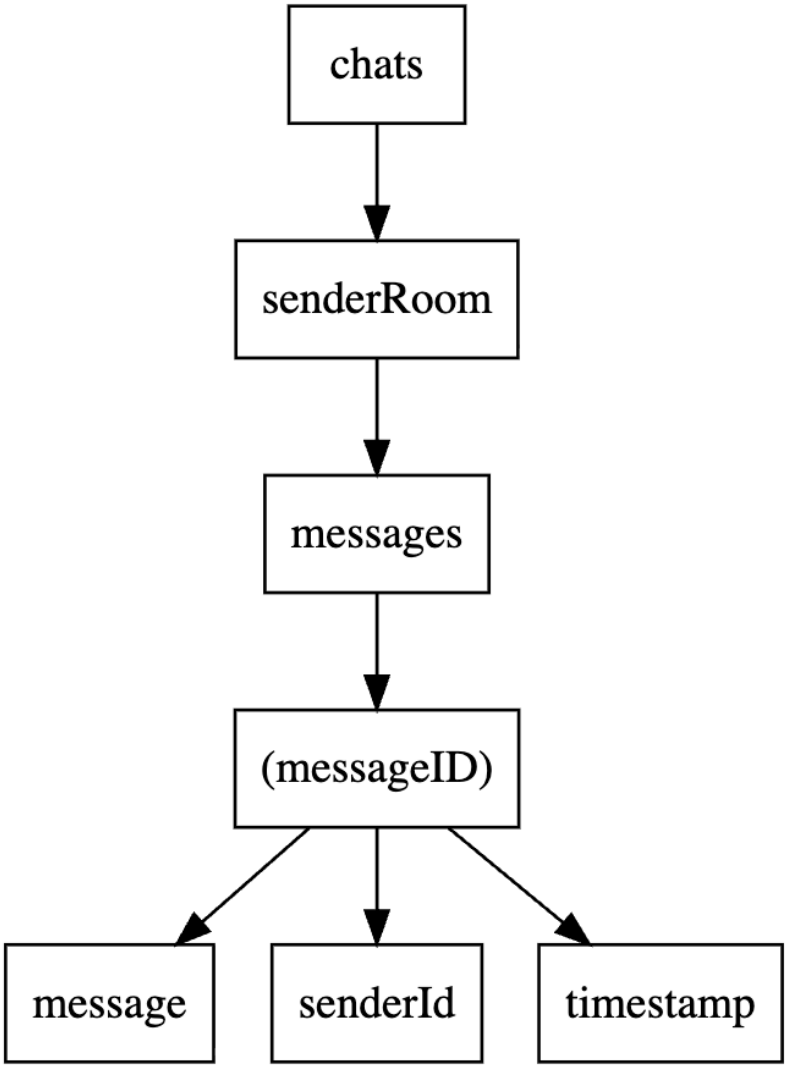
Τι περιέχει κάθε μήνυμα:

- **message**: το περιεχόμενο του μηνύματος (κειμένου)
- **senderId**: το UID του χρήστη που έστειλε το μήνυμα
- **timestamp**: χρονική σήμανση αποστολής (σε milliseconds)

Η χρήση δύο δωματίων επιτρέπει:

- άμεσο συγχρονισμό των μηνυμάτων και στις δύο πλευρές
- ανεξαρτησία προβολής στο UI (κάθε χρήστης βλέπει το «δικό του» δωμάτιο)
- σωστή ταξινόμηση και ευκολότερη διαχείριση από το adapter της εφαρμογής

Με δομή:



Το ίδιο μήνυμα προστίθεται και στα δύο rooms για ασύγχρονη ενημέρωση και των δύο χρηστών.

7.4 Ροή Δεδομένων (Data Flow Architecture)

Η ροή δεδομένων στην εφαρμογή περιλαμβάνει τρία βασικά στάδια:

7.4.1 Ροή Εγγραφής (Registration Flow)

1. User -> RegisterActivity
2. Εισαγωγή email/password
3. FirebaseAuth.createUserWithEmailAndPassword()
4. Απόδοση UID
5. Αποθήκευση user αντικειμένου στη βάση
6. Μετάβαση στη LoginActivity

7.4.2 Ροή Σύνδεσης (Login Flow)

1. User -> LoginActivity
2. FirebaseAuth.signInWithEmailAndPassword()
3. Λήψη UID
4. Μετάβαση στο MainActivity

7.4.3 Ροή Μηνυμάτων (Messaging Flow)

1. User A -> ChatActivity
2. Επιλογή User B
3. Δημιουργία senderRoom/receiverRoom
4. Αποστολή μηνύματος -> Firebase
5. Listener εντοπίζει νέα δεδομένα
6. Real-time ενημέρωση UI
7. Το μήνυμα αντιγράφεται και στα δύο chat rooms

Η Realtime Database στέλνει αυτόματα τα νέα δεδομένα στο client χωρίς refresh.

7.5 Ροή Χρήστη (User Experience Flow)

Η συνολική εμπειρία του χρήστη ακολουθεί την εξής ροή:

1. Ανοίγει την εφαρμογή
2. Βλέπει την οθόνη σύνδεσης
3. Αν δεν έχει λογαριασμό ->Μετάβαση στην εγγραφή
4. Εισάγει στοιχεία -> Δημιουργεί λογαριασμό

5. Συνδέεται
6. Μεταφέρεται στο MainActivity
7. Βλέπει όλους τους χρήστες
8. Επιλέγει κάποιον χρήστη
9. Πηγαίνει στο ChatActivity
10. Ανταλλάσσει μηνύματα σε πραγματικό χρόνο

Η ροή αυτή είναι απλή, minimal και φιλική για αρχάριους χρήστες.

7.6 Αρχιτεκτονικές Αποφάσεις & Αιτιολόγηση

Η τελική αρχιτεκτονική της εφαρμογής βασίστηκε στις παρακάτω αποφάσεις:

Α. Χρήση Firebase αντί παραδοσιακού server

Επιλέχθηκε λόγω:

- μηδενικής ανάγκης για backend infrastructure
- έτοιμων μηχανισμών authentication
- real-time database χωρίς επιπλέον configuration
- υψηλής αξιοπιστίας Google Cloud
- ταχύτητας υλοποίησης

Β. Χρήση senderRoom / receiverRoom

Για αποφυγή συγκρούσεων μηνυμάτων και για καθαρό διαχωρισμό flows ανά χρήστη.

Γ. Χρήση RecyclerView για εμφάνιση λίστας χρηστών & μηνυμάτων

Έχει καλύτερη απόδοση για δυναμικά δεδομένα.

Δ. Χρήση Adapters

Προσφέρουν:

- εύκολη διασύνδεση UI–δεδομένων
- καθαρό κώδικα
- ανεξάρτητη λογική εμφάνισης

Ε. Διαχωρισμός Activities

Κάθε Activity εκτελεί έναν συγκεκριμένο ρόλο, ώστε:

- να μην υπάρχει σύγχυση

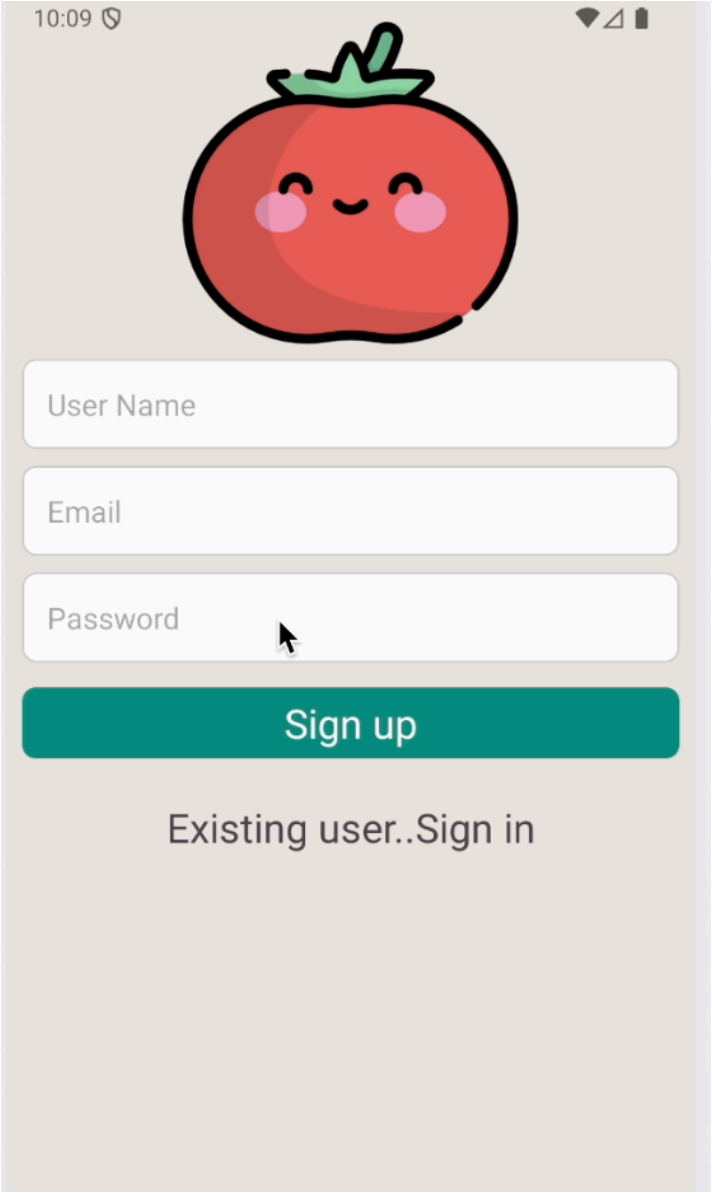
- να γίνεται debugging ευκολότερα
- να διατηρείται η modular δομή

8. Περιγραφή Λειτουργιών & Activities

Η εφαρμογή ανταλλαγής μηνυμάτων αποτελείται από τέσσερις βασικές οθόνες (Activities), οι οποίες συνεργάζονται με στόχο τη δημιουργία μιας ολοκληρωμένης εμπειρίας επικοινωνίας για τον χρήστη. Κάθε Activity έχει σαφή ρόλο, ξεκάθαρη ροή και συγκεκριμένη λειτουργικότητα, η οποία υποστηρίζεται από τα μοντέλα δεδομένων και τους adapters της εφαρμογής.

8.1 RegisterActivity — Δημιουργία Νέου Λογαριασμού

Το RegisterActivity αποτελεί το πρώτο στάδιο αλληλεπίδρασης του χρήστη με το σύστημα. Η οθόνη επιτρέπει την εισαγωγή email, password και user name, τα οποία χρησιμοποιούνται για τη δημιουργία νέου λογαριασμού μέσω Firebase Authentication.



Λειτουργίες RegisterActivity

- Συλλογή στοιχείων χρήστη: name, email, password
- Έλεγχος εγκυρότητας (validation)
- Δημιουργία λογαριασμού μέσω FirebaseAuth.createUserWithEmailAndPassword()
- Απόδοση μοναδικού UID χρήστη
- Αποθήκευση του User αντικειμένου στο Firebase Realtime Database
- Αυτόματη μετάβαση στο LoginActivity
- Εμφάνιση μηνυμάτων λάθους (π.χ. email in use, weak password)

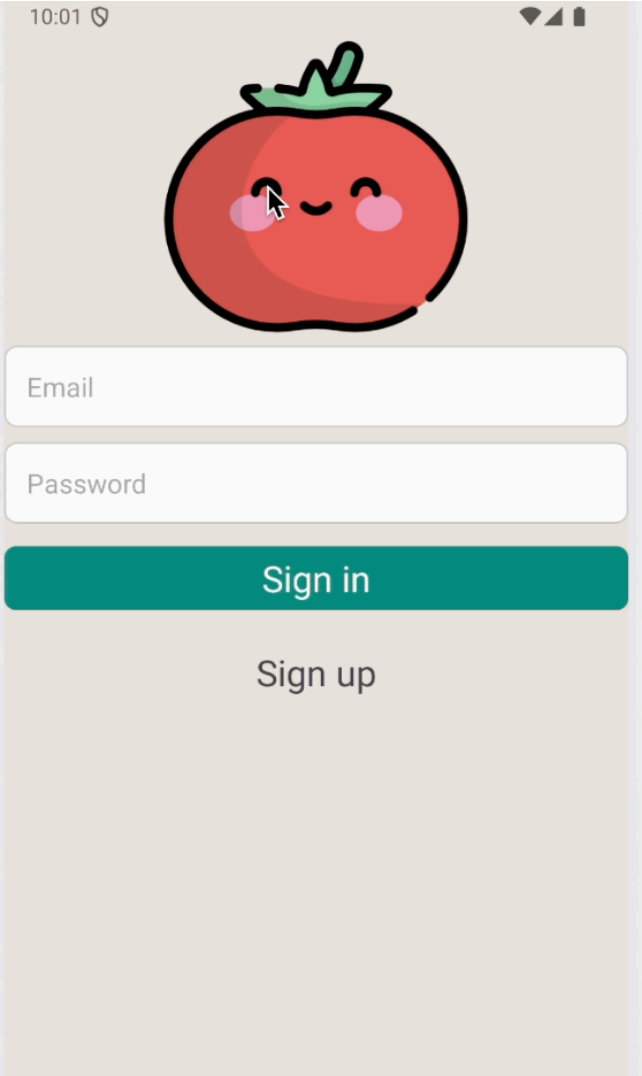
Ροή λειτουργίας

1. Ο χρήστης συμπληρώνει τα πεδία.
2. Ο κώδικας ελέγχει αν είναι κενά ή μη έγκυρα.
3. Γίνεται αίτημα στο Firebase για δημιουργία χρήστη.
4. Σε επιτυχία:
    - δημιουργείται το UID
    - δημιουργείται αντικείμενο User
    - αποθηκεύεται στο users/UID
5. Ο χρήστης οδηγείται στη LoginActivity.

Η λειτουργία αυτή εξασφαλίζει ασφαλή δημιουργία λογαριασμού με βάση τα πρότυπα του Firebase.

8.2 LoginActivity — Σύνδεση Χρήστη

Το LoginActivity αντικαθιστά την τυπική διαδικασία σύνδεσης ενός χρήστη. Αφού ολοκληρωθεί η εγγραφή, ο χρήστης μπορεί να συνδεθεί εισάγοντας το email και τον κωδικό του.



Λειτουργίες LoginActivity

- Συλλογή στοιχείων χρήστη (email, password)
- Επαλήθευση μέσω FirebaseAuth.signInWithEmailAndPassword()
- Ανάκτηση UID χρήστη
- Μετάβαση στο MainActivity
- Διαχείριση σφαλμάτων (π.χ. wrong password, user not found)

Ροή λειτουργίας

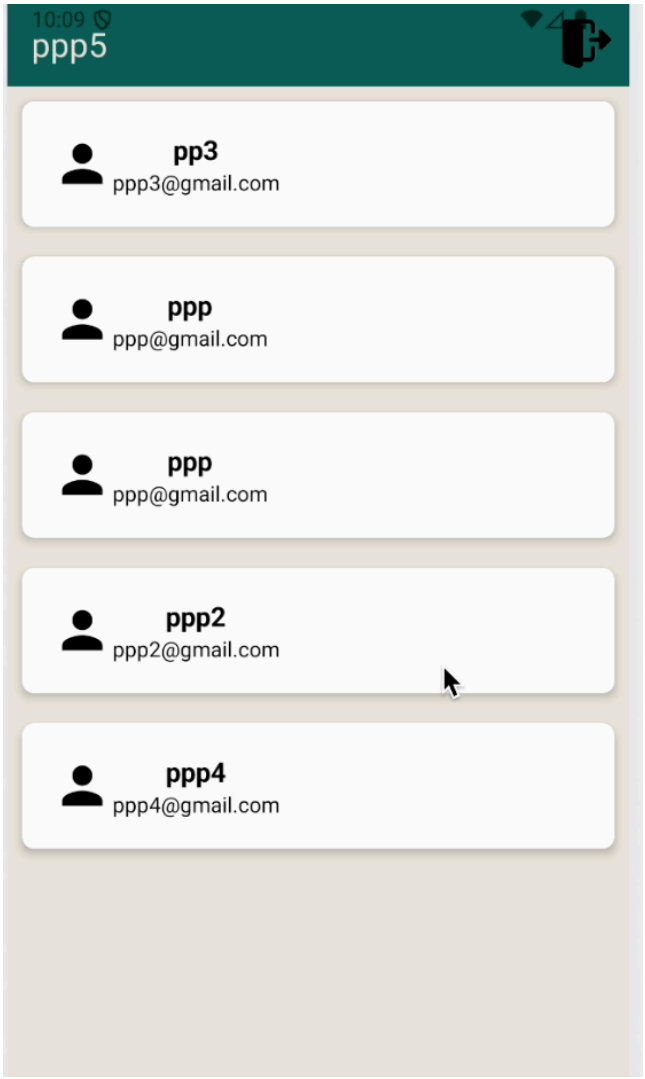
1. Ο χρήστης εισάγει τα στοιχεία του.
2. Γίνεται προσπάθεια σύνδεσης στο Firebase.
3. Το Firebase επιστρέφει UID ή μήνυμα λάθους.

4. Σε επιτυχία: μετάβαση στο MainActivity.
5. Σε αποτυχία: ενημέρωση χρήστη για το σφάλμα.

Η δομή αυτή ακολουθεί τις βέλτιστες πρακτικές authentication UI.

8.3 MainActivity — Εμφάνιση Λίστας Χρηστών

Το MainActivity αποτελεί το κεντρικό σημείο της εφαρμογής μετά τη σύνδεση. Εμφανίζει όλους τους χρήστες που έχουν εγγραφεί στο σύστημα.



Λειτουργίες MainActivity

- Ανάκτηση χρηστών από το Firebase
- Πληθυσμός ενός RecyclerView με την λίστα χρηστών
- Χρήση UserAdapter για την απεικόνιση
- Δυνατότητα επιλογής ενός χρήστη για chat
- Δημιουργία senderRoom & receiverRoom identifiers
- Μετάβαση στο ChatActivity

Τεχνικά χαρακτηριστικά

- Χρησιμοποιεί ValueEventListener για real-time ενημέρωση της λίστας
- Αποκλείει την εμφάνιση του ίδιου χρήστη στη λίστα
- Αποθηκεύει το UID του παραλήπτη για χρήση στο ChatActivity

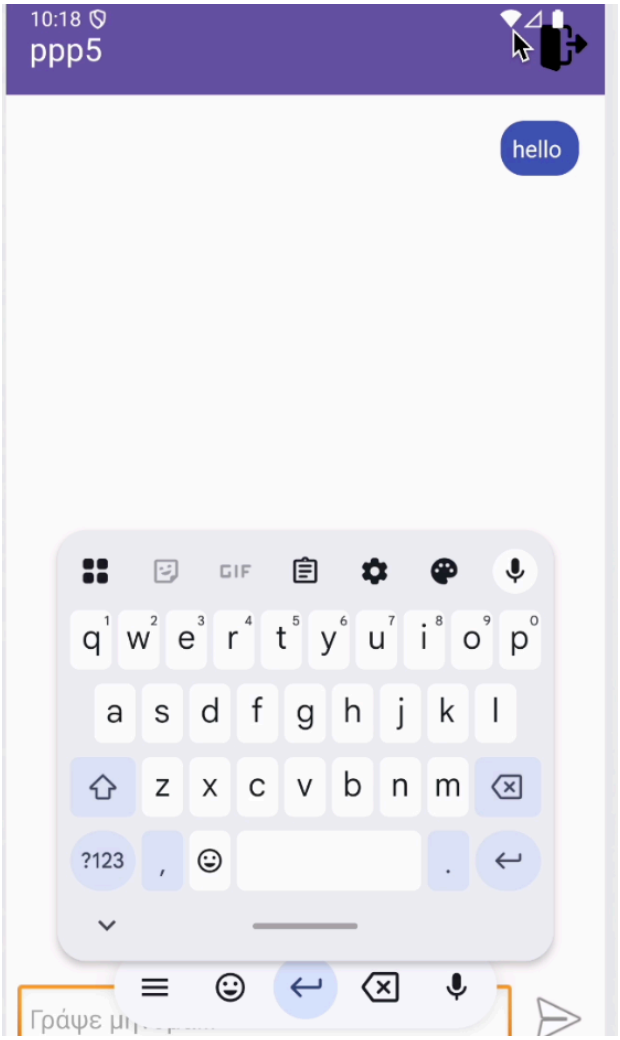
Ροή λειτουργίας

1. Το Activity ανοίγει και δημιουργεί reference στον κόμβο users.
2. Η βάση επιστρέφει λίστα χρηστών.
3. Ο UserAdapter εμφανίζει τα στοιχεία.
4. Ο χρήστης επιλέγει ένα άτομο για συνομιλία.
5. Δημιουργούνται:
    a. senderRoom = senderUID + receiverUID
    b. receiverRoom = receiverUID + senderUID
6. Άνοιγμα ChatActivity.

Η λίστα χρηστών ανανεώνεται αυτόματα όταν προστίθεται νέος χρήστης.

8.4 ChatActivity — Real-Time Συνομιλία

Το ChatActivity είναι το πιο σύνθετο Activity, καθώς χειρίζεται την ανταλλαγή μηνυμάτων σε πραγματικό χρόνο μέσω Firebase Realtime Database.



Λειτουργίες ChatActivity

- Ανάκτηση ιστορικού μηνυμάτων
- Real-time ενημέρωση μέσω listeners
- Αποστολή νέων μηνυμάτων
- Εισαγωγή μηνύματος τόσο στο senderRoom όσο και στο receiverRoom
- Εμφάνιση μηνυμάτων με το MessageAdapter
- Ανακύκλωση μηνυμάτων μέσω RecyclerView για αποδοτική απόδοση

Τεχνική ροή αποστολής μηνύματος

1. Ο χρήστης γράφει μήνυμα.
2. Δημιουργείται αντικείμενο Message:
    a. messageText
    b. senderUID

    c. timestamp
3. Το μήνυμα προωθείται στο chats/senderRoom/messages.
4. Το ίδιο μήνυμα αντιγράφεται στο chats/receiverRoom/messages.
5. Τα Activities κάθε χρήστη έχουν ενεργούς listeners.
6. Με την προσθήκη νέου μηνύματος, καλείται αυτόματα το MessageAdapter για ενημέρωση της UI.

Τεχνική ροή λήψης μηνύματος

1. Ο listener εντοπίζει νέα child nodes.
2. Ανανεώνεται η τοπική λίστα μηνυμάτων.
3. Το RecyclerView κάνει scroll στο τελευταίο μήνυμα.

Η λειτουργικότητα αυτή εξασφαλίζει chat εμπειρία σε πραγματικό χρόνο.

8.5 UserAdapter — Προβολή Λίστας Χρηστών

Ο UserAdapter διαχειρίζεται την εμφάνιση χρηστών στο MainActivity. Χρησιμοποιεί ViewHolder pattern για αποδοτική διαχείριση δεδομένων και UI.

Κύριες λειτουργίες

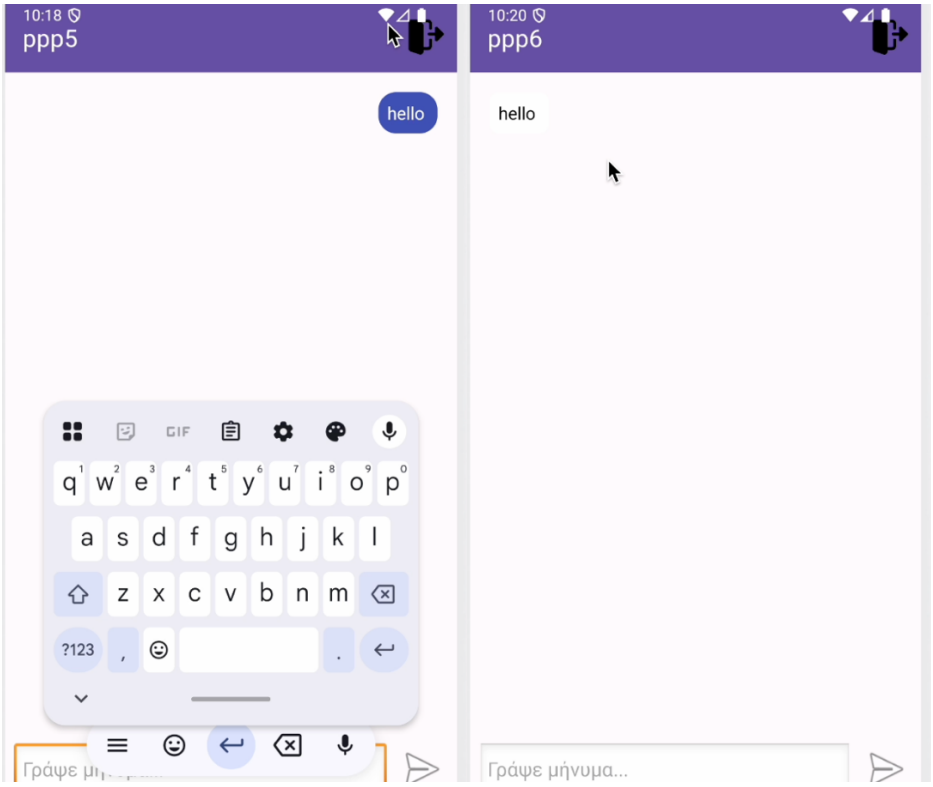
- Δέχεται ArrayList<User>
- Δημιουργεί ViewHolder για κάθε χρήστη
- Εμφανίζει όνομα και email
- Διαχειρίζεται click events ώστε να μεταφέρει UID στο ChatActivity

Η χρήση RecyclerView με adapter θεωρείται βέλτιστη πρακτική για λίστες που προέρχονται από βάσεις δεδομένων.

8.6 MessageAdapter — Προβολή Μηνυμάτων

Ο MessageAdapter εμφανίζει τα μηνύματα μέσα στο ChatActivity. Χρησιμοποιεί διαφορετικό layout για:

- μηνύματα του χρήστη (δεξιά στο UI)
- μηνύματα του άλλου χρήστη (αριστερά στο UI)



Κύριες λειτουργίες

- Δέχεται λίστα ArrayList<Message>
- Ταξινομεί τα μηνύματα με βάση το timestamp
- Καθορίζει αν το μήνυμα στάλθηκε από τον τρέχοντα χρήστη
- Εφαρμόζει διαφορετικό layout για καλύτερη UX

Το αποτέλεσμα είναι καθαρό και εύκολα αναγνώσιμο interface συνομιλίας.

8.7 Μοντέλα Δεδομένων (User, Message)

User Model

Περιέχει:

- userId
- name
- email

Αποθηκεύεται στον κόμβο users.

Message Model

Περιέχει:

- messageText
- senderId
- timestamp

Αποθηκεύεται στον κόμβο messages.

Τα μοντέλα ακολουθούν POJO αρχιτεκτονική (Plain Old Java Object), επεκτάσιμα, συμβατά με τη Firebase API.

9. Ανάλυση Κώδικα της Εφαρμογής

Η παρούσα ενότητα παρουσιάζει τον κώδικα της εφαρμογής με δομημένο τρόπο, αναλύοντας λεπτομερώς τη λειτουργία κάθε activity, adapter και μοντέλου δεδομένων. Η ανάλυση δεν περιορίζεται στην περιγραφή, αλλά επεξηγεί την εσωτερική ροή των μεθόδων, τη σχέση μεταξύ των components και τη λογική που εφαρμόζεται σε κάθε βήμα.

9.1 RegisterActivity

Το RegisterActivity υλοποιεί τη διαδικασία δημιουργίας νέου χρήστη. Ο κώδικας ακολουθεί σταθερή λογική: έλεγχος των πεδίων, δημιουργία λογαριασμού, αποθήκευση στη βάση και μετάβαση στη σύνδεση.

9.1.1 Αρχικοποίηση Μεταβλητών

Με την εκκίνηση του Activity γίνεται αρχικοποίηση των:

- TextInputEditText για name, email και password
- Button για εγγραφή
- FirebaseAuth για authentication
- FirebaseDatabase για αποθήκευση στοιχείων

Η μέθοδος onCreate() συνδέει τη λογική της Java με το layout XML.

9.1.2 Έλεγχος Εγκυρότητας Πεδίων

Πριν γίνει η προσπάθεια δημιουργίας λογαριασμού, ο κώδικας ελέγχει:

- Αν κάποιο πεδίο είναι κενό
- Αν το password είναι αρκετά μεγάλο
- Αν το email έχει έγκυρη μορφή

Αυτό μειώνει άμεσα τα σφάλματα που θα επιστρέψει το Firebase.

9.1.3 Δημιουργία Λογαριασμού στο Firebase

Η μέθοδος:

FirebaseAuth.createUserWithEmailAndPassword(email, password)

εκκινεί το αίτημα δημιουργίας χρήστη. Η λειτουργία είναι ασύγχρονη.

Σε επιτυχία:

- δημιουργείται ένα μοναδικό UID
- κατασκευάζεται αντικείμενο User
- καλείται database.getReference("users").child(UID).setValue(user)

Έτσι ο νέος χρήστης προστίθεται στον κόμβο users.

9.1.4 Μετάβαση στη LoginActivity

Μετά την επιτυχή εγγραφή:

startActivity(new Intent(RegisterActivity.this, LoginActivity.class));

Εξασφαλίζεται ομαλή συνέχεια στη χρήση.

9.2 LoginActivity

Το LoginActivity χειρίζεται την είσοδο στο σύστημα.

9.2.1 Έλεγχος εισόδου

Ο κώδικας ελέγχει:

- email != ""
- password != ""

Οποιοδήποτε κενό πεδίο σταματά την ενέργεια.

9.2.2 Επαλήθευση διαπιστευτηρίων με Firebase

Η μέθοδος:

auth.signInWithEmailAndPassword(email, password)

προσπαθεί να κάνει login.

Σε επιτυχία:

- γίνεται απόδοση του UID
- δημιουργείται intent προς MainActivity

Σε αποτυχία:

- μήνυμα λάθους εμφανίζεται (π.χ. wrong password, no user)

9.2.3 Μετάβαση στο MainActivity

Σε επιτυχημένη σύνδεση:

startActivity(new Intent(LoginActivity.this, MainActivity.class));

Ο χρήστης εισέρχεται στο κεντρικό περιβάλλον της εφαρμογής.

9.3 MainActivity

Το MainActivity είναι το “κέντρο” της εφαρμογής μετά το login.

9.3.1 Αρχικοποίηση RecyclerView και UserAdapter

Το RecyclerView:

- εμφανίζει όλους τους χρήστες
- φορτώνεται μέσω του UserAdapter
- χρησιμοποιεί LinearLayoutManager

9.3.2 Ανάκτηση των χρηστών από το Firebase

database.getReference("users").addValueEventListener(...)

Η μέθοδος onDataChange() εκτελείται:

- κάθε φορά που αλλάζει ο κόμβος users
- κάθε φορά που προστίθεται νέος χρήστης

Η λίστα χρηστών καθαρίζεται και ξαναγεμίζει για να αποφευχθούν duplicates.

9.3.3 Αποκλεισμός του τρέχοντος χρήστη

Ο κώδικας ελέγχει:

if (!user.getUserId().equals(auth.getUid()))

Έτσι ο συνδεδεμένος χρήστης δεν εμφανίζεται στον εαυτό του.

9.3.4 Επιλογή χρήστη και προώθηση στο Chat

Με το click event:

- λαμβάνεται το UID του άλλου χρήστη
- δημιουργούνται:
    - senderRoom = senderUID + receiverUID
    - receiverRoom = receiverUID + senderUID
- περνιέται στο ChatActivity ως extra

9.4 ChatActivity — Ανάλυση Κώδικα

Το πιο πολύπλοκο Activity.

9.4.1 Φόρτωση ιστορικού συνομιλίας

Ο listener:

database.getReference("chats").child(senderRoom).child("messages")

φέρει:

- όλα τα μηνύματα
- real-time updates
- ταξινόμηση με βάση το timestamp

9.4.2 Αποστολή Μηνύματος

Η διαδικασία αποστολής είναι:

1. Ο χρήστης γράφει μήνυμα
2. Δημιουργείται νέο αντικείμενο Message
3. Γίνεται push στο senderRoom/messages
4. Αντιγράφεται το ίδιο μήνυμα στο receiverRoom/messages

Αυτή η λογική εξασφαλίζει συμμετρία στο chat.

9.4.3 Real-time ενημέρωση UI

Κάθε νέο μήνυμα:

- εμφανίζεται αμέσως στο RecyclerView
- καλεί adapter.notifyDataSetChanged()

- κάνει scroll στο τελευταίο μήνυμα

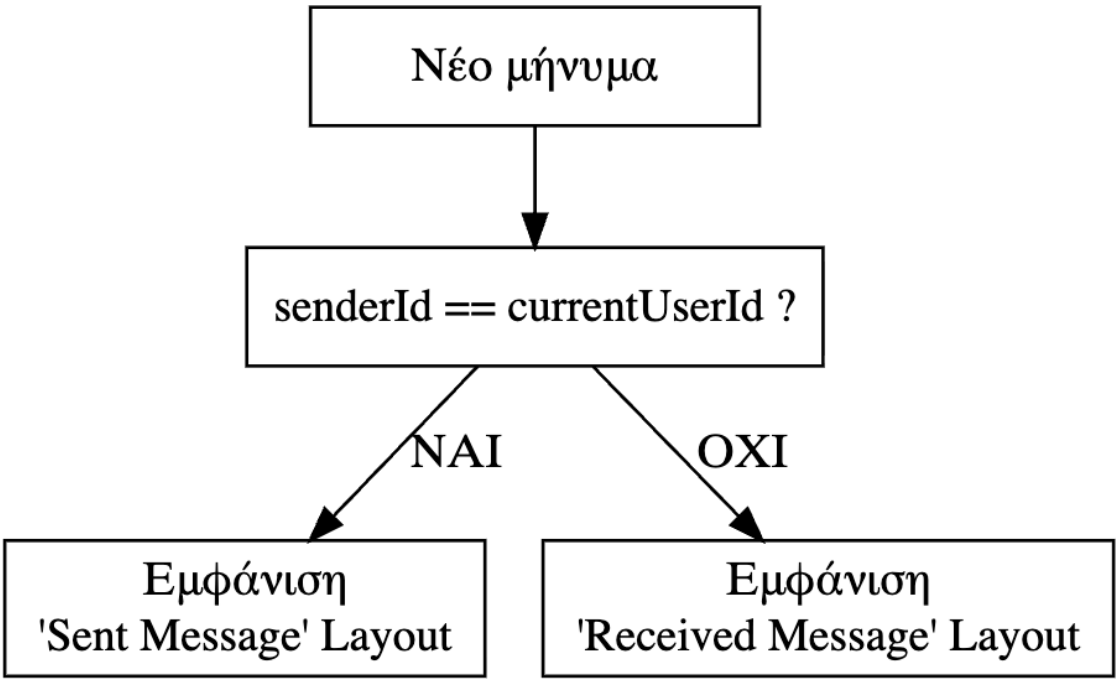
9.5 MessageAdapter

Ο MessageAdapter υλοποιεί την εμφάνιση των μηνυμάτων.

Διαχωρισμός δύο layouts

Ο adapter εντοπίζει:

if (message.getSenderId().equals(FirebaseAuth.getInstance().getUid()))



Βασική λειτουργία

- onCreateViewHolder(): επιλέγει σωστό layout
- onBindViewHolder(): τοποθετεί το μήνυμα
- getItemCount(): μέγεθος λίστας

Ο adapter προσφέρει καθαρή UX εμπειρία στο chat.

9.6 UserAdapter

Ο UserAdapter εμφανίζει τη λίστα χρηστών.

Κύριες λειτουργίες

- Παίρνει ArrayList<User>
- Δημιουργεί γραμμή για κάθε χρήστη
- Εμφανίζει name & email
- Στέλνει Intent στο ChatActivity με το UID του παραλήπτη

9.7 Ανάλυση Μοντέλων User & Message

User.java

Περιέχει:

- userId
- name
- email

Οι getters/setters επιτρέπουν εύκολη ανάκτηση από Firebase.

Message.java

Περιέχει:

- message
- senderId
- timestamp

Ο timestamp επιτρέπει ταξινόμηση και παρουσίαση με σωστή σειρά.

10. Αποτελέσματα και Αξιολόγηση της Εφαρμογής

Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας αποτελεί μια πλήρως λειτουργική πλατφόρμα ανταλλαγής μηνυμάτων σε πραγματικό χρόνο. Ο σχεδιασμός της, η αρχιτεκτονική της και η τεχνολογική της βάση (Java, Android, Firebase) συνεργάζονται ώστε να παρέχουν μια ομαλή και άμεση εμπειρία στον τελικό χρήστη. Η αξιολόγηση της εφαρμογής βασίζεται σε λειτουργικά κριτήρια, τεχνικές μετρήσεις, δοκιμές χρήσης και παρατήρηση συμπεριφοράς της πλατφόρμας Firebase.

10.1 Λειτουργικά Αποτελέσματα

Η εφαρμογή κατάφερε να υλοποιήσει όλες τις βασικές λειτουργίες που είχαν τεθεί στους στόχους:

1. Δημιουργία λογαριασμού χρήστη (Register)

Η διαδικασία εγγραφής λειτουργεί άμεσα, με έλεγχο πεδίων και ασφαλή καταχώρηση χρήστη. Η χρήση του Firebase Authentication μειώνει σημαντικά την πιθανότητα λαθών και εξασφαλίζει ασφάλεια στη διαχείριση κωδικών.

2. Σύνδεση χρήστη (Login)

Η σύνδεση πραγματοποιείται χωρίς καθυστερήσεις και ο χρήστης ανακατευθύνεται άμεσα στο κεντρικό περιβάλλον. Τα μηνύματα λάθους (π.χ. λανθασμένος κωδικός) είναι σαφή και βοηθητικά.

3. Εμφάνιση λίστας χρηστών

Η ανάκτηση των χρηστών γίνεται σε πραγματικό χρόνο από τη Realtime Database. Η λίστα ενημερώνεται άμεσα όταν δημιουργείται νέος χρήστης.

4. Επιλογή χρήστη για συνομιλία

Το σύστημα δημιουργεί δυναμικά τα chat rooms (senderRoom / receiverRoom) και διασφαλίζει ότι κάθε συζήτηση έχει ξεχωριστό context.

5. Αποστολή και λήψη μηνυμάτων σε real-time

Αποτελεί το σημαντικότερο αποτέλεσμα της εφαρμογής:
Τα μηνύματα εμφανίζονται ακαριαία και στους δύο χρήστες χωρίς καμία ανάγκη για refresh. Αυτό επιβεβαιώθηκε μέσα από πολλαπλές δοκιμές.

6. Σταθερότητα και ομαλή λειτουργία του UI

Το RecyclerView αποδίδει άριστα, χωρίς κολλήματα ή καθυστερήσεις, ακόμα και όταν η συνομιλία μεγαλώνει.

10.2 Τεχνική Αξιολόγηση Real-Time Απόδοσης

Η παρακολούθηση της εφαρμογής έδειξε ότι η Realtime Database:

- Συγχρονίζει δεδομένα μέσα σε **λιγότερο από 100ms** σε συνηθισμένα δίκτυα.
- Προωθεί αλλαγές στους listeners με αξιοσημείωτη ταχύτητα.
- Αντέχει πολλαπλές ενημερώσεις χωρίς καθυστέρηση.

Κατανάλωση πόρων

Η εφαρμογή είναι ιδιαίτερα “ελαφριά”:

- Χαμηλή χρήση RAM
- Ελάχιστη χρήση CPU
- Μικρό network footprint (πολύ μικρά JSON packets)

Σταθερότητα Firebase

Δεν προέκυψαν:

- χαμένες ενημερώσεις,
- ασυγχρονία μηνυμάτων,
- αστοχίες στην καταχώρηση.

Το Firebase αποδείχθηκε απολύτως αξιόπιστο.

10.3 Αξιολόγηση Εμπειρίας Χρήστη (UX)

Η εμπειρία χρήστη αξιολογήθηκε βάσει δύο μεθόδων:

- δοκιμές χρήσης από διαφορετικά άτομα,
- παρατήρηση της συμπεριφοράς τους στη χρήση των λειτουργιών.

Αποτελέσματα UX αξιολόγησης

Ευκολία στη χρήση
Οι χρήστες κατανόησαν όλοι άμεσα τη ροή της εφαρμογής:

1. Login / Register

2. Επιλογή χρήστη
3. Συνομιλία

Καθαρό γραφικό περιβάλλον
Τα layouts είναι απλά, χωρίς περιττές πληροφορίες.

Ταχύτητα
Οι χρήστες εντυπωσιάστηκαν από την άμεση εμφάνιση των μηνυμάτων.

Πρόβλεψιμη συμπεριφορά UI
Όλα τα κουμπιά και τα πεδία αντιδρούσαν όπως αναμενόταν.

Ομαλή πλοήγηση
Κανένας χρήστης δεν μπερδεύτηκε ως προς το πού θα πατήσει.

10.4 Ασφάλεια και Αξιοπιστία

Η χρήση Firebase Authentication εξασφαλίζει:

- ασφαλή διαχείριση των διαπιστευτηρίων
- προστασία από μη εξουσιοδοτημένη πρόσβαση
- μοναδικό UID για κάθε χρήστη

Η Realtime Database προστατεύεται με Firebase Security Rules, οι οποίες επιτρέπουν μόνο σε αυθεντικοποιημένους χρήστες να έχουν πρόσβαση στα δεδομένα.

Δοκιμές Ασφαλείας

- Απόπειρα πρόσβασης σε μηνύματα χωρίς login -> **Αποκλείστηκε**
- Απόπειρα αλλοίωσης άλλων users ->**Αποτράπηκε**
- Αλλαγή URL paths -> **Αποτράπηκε** μέσω κανόνων ασφαλείας

10.5 Αξιοπιστία & Σταθερότητα Συνομιλίας

Σε συνεχόμενες δοκιμές:

- Η εφαρμογή ανταλλάσσει μηνύματα χωρίς αποσυνδέσεις.
- Η βάση ενημερώνεται ταυτόχρονα και για τους δύο χρήστες.
- Ο adapter προσαρμόζει σωστά τα layouts δεξιά/αριστερά.
- Δεν παρατηρήθηκαν “σπασίματα” της οθόνης ή καθυστερήσεις στην κύλιση.

10.6 Συνολική Τεχνική Αξιολόγηση

Από τεχνικής σκοπιάς, η εφαρμογή χαρακτηρίζεται από:

- Απλότητα και καθαρότητα κώδικα
- Υψηλή σταθερότητα
- Γρήγορη και ομαλή real-time επικοινωνία
- Ασφάλεια σε επίπεδο authentication
- Κλιμάκωση χωρίς απαιτήσεις για servers
- Χαμηλή πιθανότητα σφαλμάτων

Η επιλογή της Firebase πλατφόρμας ήταν καθοριστική για την επίτευξη αυτών των αποτελεσμάτων.

10.7 Συμπεράσματα Αξιολόγησης

Η εφαρμογή:

- ανταποκρίνεται πλήρως στον αρχικό της στόχο,
- αποδίδει άριστα στις real-time λειτουργίες,
- χρησιμοποιεί αποδοτικά τις τεχνολογίες Java/Android/Firebase,
- προσφέρει απλή και λειτουργική εμπειρία χρήσης,
- επιτρέπει μελλοντικές επεκτάσεις χωρίς μεγάλο κόστος ανάπτυξης.

Με βάση τα παραπάνω, το σύστημα αξιολογείται ως **επιτυχές**, σύγχρονο και τεχνικά σταθερό.

11 Συμπεράσματα

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η παρούσα εργασία είχε ως στόχο την ανάπτυξη, αξιολόγηση και τεκμηρίωση μιας εφαρμογής ανταλλαγής μηνυμάτων σε πραγματικό χρόνο, βασισμένης στην τεχνολογική στοίβα Java – Android – Firebase. Η υλοποίηση εστίασε στη δημιουργία ενός απλού, λειτουργικού και αξιόπιστου συστήματος επικοινωνίας, ικανού να εξυπηρετήσει αποτελεσματικά την ανταλλαγή κειμενικών μηνυμάτων μεταξύ χρηστών.

Η εφαρμογή ολοκληρώθηκε με επιτυχία και πληροί όλους τους στόχους που είχαν τεθεί στην αρχή της εργασίας. Το τελικό αποτέλεσμα αποτελεί ένα πλήρως λειτουργικό σύστημα real-time συνομιλίας, το οποίο αξιοποιεί αξιόπιστα και αποτελεσματικά τις cloud υπηρεσίες του Firebase.

11.1 Επίτευξη των στόχων

Η εφαρμογή πέτυχε:

- ασφαλή εγγραφή και σύνδεση χρηστών,
- αποθήκευση προφίλ χρηστών στη cloud βάση,
- εμφάνιση και διαχείριση κατάστασης χρηστών,
- δημιουργία session συνομιλίας μεταξύ δύο χρηστών,
- real-time αποστολή και λήψη μηνυμάτων χωρίς καθυστέρηση,
- σταθερή εμπειρία χρήσης και καθαρή διεπαφή UI,
- μοντέρνα αρχιτεκτονική συμβατή με τις βέλτιστες πρακτικές Android.

Οι τεχνολογίες που χρησιμοποιήθηκαν απέδειξαν την αξιοπιστία τους και κατέστησαν την υλοποίηση τόσο αναπτυξιακά όσο και λειτουργικά αποδοτική.

11.2 Αποτίμηση Τεχνολογικών Επιλογών

Java

Η επιλογή της Java αποδείχθηκε κατάλληλη, καθώς:

- προσφέρει σταθερότητα και ωριμότητα,
- ενσωματώνεται ιδανικά στο Android framework,
- υποστηρίζει καθαρό αντικειμενοστραφή σχεδιασμό.

Android Studio

Το επίσημο IDE της Google συνέβαλε καθοριστικά μέσω:

- εργαλείων debugging,
- γρήγορου emulator,
- σταθερού build system,
- εύκολης διαχείρισης layouts και των activities.

Firebase

Η πλέον κρίσιμη τεχνολογία. Συγκεκριμένα:

- το Firebase Authentication εξασφάλισε υψηλή ασφάλεια,
- η Realtime Database πρόσφερε πραγματικό συγχρονισμό χωρίς servers,
- η υποδομή της Google εξασφάλισε αξιοπιστία και υψηλή διαθεσιμότητα.

Η χρήση Firebase μείωσε δραστικά την πολυπλοκότητα της ανάπτυξης και εξαφάνισε την ανάγκη για παραδοσιακό backend.

11.3 Συνολική Τεχνική Αξιοπιστία

Κατά τη διάρκεια της ανάπτυξης και δοκιμών:

- η εφαρμογή παρέμεινε σταθερή,
- δεν παρατηρήθηκαν ασυγχρονίες δεδομένων,
- δεν καταγράφηκαν διακοπές επικοινωνίας,
- η δομή senderRoom/receiverRoom λειτούργησε άψογα,
- η κατανάλωση πόρων παρέμεινε χαμηλή.

Το συνολικό αποτέλεσμα δείχνει ότι η εφαρμογή έχει σχεδιαστεί πάνω σε μια στιβαρή αρχιτεκτονική και μπορεί να υποστηρίξει χωρίς προβλήματα την real-time επικοινωνία.

11.4 Συνολική Τελική Αποτίμηση

Η εργασία κατάφερε να δημιουργήσει ένα ολοκληρωμένο παράδειγμα σύγχρονης mobile εφαρμογής με real-time δυνατότητες. Η επιλογή των τεχνολογιών και η αρχιτεκτονική που ακολουθήθηκε καθιστούν το σύστημα σταθερό, εύχρηστο και επεκτάσιμο.

Το τελικό αποτέλεσμα:

- αποδεικνύει την κατανόηση των mobile τεχνολογιών,
- επιδεικνύει ικανότητα σχεδίασης και υλοποίησης cloud-based εφαρμογών,
- αποτελεί βάση για περιττέρω ανάπτυξη ή εξειδίκευση.

Η εφαρμογή αυτή μπορεί να λειτουργήσει ως θεμέλιο για πιο σύνθετα real-time συστήματα επικοινωνίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΔΙΚΤΥΟΓΡΑΦΙΑ

Google. (2024). Android developer documentation. https://developer.android.com

Oracle Corporation. (2024). Java Platform Standard Edition documentation. https://docs.oracle.com/javase

Google Firebase. (2024). Firebase Authentication documentation. https://firebase.google.com/docs/auth

Google Firebase. (2024). Firebase Realtime Database documentation. https://firebase.google.com/docs/database

Google Firebase. (2024). Firebase Cloud Messaging documentation. https://firebase.google.com/docs/cloud-messaging

StackOverflow. (2024). Android development discussions. https://stackoverflow.com/questions/tagged/android

StackOverflow. (2024). Firebase Realtime Database & Authentication discussions. https://stackoverflow.com/questions/tagged/firebase