Πανεπιστήμιο Πελοποννήσου

Σχολή Οικονομίας και Τεχνολογίας

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Π.Μ.Σ. στην Επιστήμη Υπολογιστών

Διπλωματική Εργασία

«Σχεδιασμός και ανάπτυξη εγγενούς Android εφαρμογής σε Kotlin για τον εντοπισμό και την αποφυγή ανεπιθύμητων εισερχομένων τηλεφωνικών κλήσεων»

Αναστάσιος Φόρτης

ΑΜ: 2022202002027 – Email: dit2027cst@go.uop.gr

Υπεύθυνος Καθηγητής: κ. Νικόλαος Τσελίκας

Τρίπολη, Μάρτιος 2022

Περιεχόμενα

Ευχαριστίες

Με την ολοκλήρωση της παρούσας εργασίας θα ήθελα να ευχαριστήσω τον καθηγητή κ. Νικόλαο Τσελίκα για τις πολύτιμες γνώσεις και συμβουλές που μου παρείχε καθ' όλη τη διάρκεια δημιουργίας της διπλωματικής μου εργασίας. Επίσης, θα ήθελα να ευχαριστήσω όλους τους δασκάλους και καθηγητές που με τις διδαχές τους μου έδωσαν όλα τα απαραίτητα εφόδια ώστε να ενταχθώ και να πορευθώ σαν πολύτιμο, θέλω να πιστεύω, μέλος της κοινωνίας μας.

Θα ήθελα ακόμη να ευχαριστήσω την οικογένειά μου, τους γονείς μου και τον αδερφό μου, καθώς χωρίς τη δική τους στήριξη και βοήθεια σε όλη τη διάρκεια των σπουδών μου δεν θα βρισκόμουν σήμερα εδώ.

Επιπλέον, δεν μπορώ να παραλείψω να ευχαριστήσω την σύντροφό μου και όλους τους φίλους μου οι οποίοι με επηρέασαν θετικά και βρέθηκαν δίπλα μου όποτε τους χρειάστηκα δίνοντάς μου ιδέες και στήριξη.

Περίληψη

Το παρόν έγγραφο αποτελεί την ανάλυση της εφαρμογής με τίτλο “NBlockker”. Λειτουργεί ως εγχειρίδιο για τους πιθανούς χρήστες της εφαρμογής και περιγράφει όλα τα βασικά στοιχεία αναφορικά με τον σχεδιασμό και την υλοποίησή της. Η εφαρμογή «NBlockker» έχει ως σκοπό να βοηθήσει τους χρήστες να μπορέσουν να αποφύγουν τις ενοχλητικές κλήσεις, οι οποίες κατακλύζουν αυτή την περίοδο τον κόσμο. Στα επόμενα κεφάλαια θα γίνει περιγραφή των προβλημάτων που προσπαθεί να λύση η εφαρμογή. Θα αναφερθούν οι γλώσσες προγραμματισμού, αλλά και οι τεχνολογίες που χρησιμοποιήθηκαν για να δημιουργηθεί η εφαρμογή. Στο 1ο κεφάλαιο θα αναλύσουμε το πρόβλημα το οποίο εντοπίσαμε και προσπαθήσαμε να λύσουμε. Στο 2ο κεφάλαιο περιγράφουμε τη λειτουργικότητα της εφαρμογής. Στο 3ο κεφάλαιο γίνεται μια αναφορά στις τεχνολογίες ανάπτυξης. Στο 4ο κεφάλαιο περιγράφονται οι λειτουργικές απαιτήσεις της εφαρμογής καθώς στο 5ο και 6ο κεφάλαιο αναφέρονται βασικά κομμάτια της υλοποίησης με περιγραφές κώδικα και τέλος στο 7ο κεφάλαιο αναφέρονται μελλοντικές επεκτάσεις της εφαρμογής.

Λέξεις κλειδιά

Ανάπτυξη εφαρμογών κινητών συσκευών, Kotlin, Jetpack Compose, Hilt, Native Development, Android Studio, IaaS, Google Firebase

Abstract

This document is an analysis of the application entitled "NBlockker". It functions as a manual for potential users of the application and describes all the basic elements regarding its design and construction. The "NBlockker" application aims to help users to avoid annoying calls that are currently flooding the people. The following chapters will describe the problems that the application tries to solve. The programming languages used and the technologies used to create the application will be mentioned. In the first chapter we will analyze the problem that we identified and tried to solve. In the 2nd chapter we describe the functionality of the application. Chapter 3 provides an overview of development technologies. Chapter 4 describes the operational requirements of the application as Chapter 5 and Chapter 6 list key parts of the implementation with code descriptions and finally Chapter 7 lists future extensions of the application.

Keywords

Kotlin, Jetpack Compose, Hilt, Native Development, Android Studio, IaaS, Google Firebase

1. Εισαγωγή

Το πρώτο κεφάλαιο της αναφοράς αναφέρεται στον σκοπό της δημιουργίας της εργασίας και σε ποιο πρόβλημα προσπαθεί να λύσει.

1.1 Το πρόβλημα

Στις μέρες μας οι κινητές τηλεφωνικές συσκευές έχουν γίνει αναπόσπαστο κομμάτι της ζωής μας. Καθημερινά ξοδεύουμε αρκετό χρόνο χρησιμοποιώντας τις κινητές τηλεφωνικές συσκευές για επικοινωνία με άλλους ανθρώπους. Ήδη από την εμφάνιση των πρώτων κινητών συσκευών οι άνθρωποι κατάλαβαν ότι λύθηκε, εν μέρει, το πρόβλημα που υπήρχε παλαιότερα στο να επικοινωνήσουν οι άνθρωποι μεταξύ τους σε όλα τα μέρη της γης. Οι κινητές συσκευές όχι απλά έλυσαν αυτό το πρόβλημα, αλλά εξελίχθηκαν με αποτέλεσμα να έχουν γίνει πλέον ψηφιακοί βοηθοί στην ζωή του ανθρώπου.

Το γεγονός αυτό εκμεταλλεύονται αρκετές επιχειρήσεις για να κερδίσουν χρήματα, χωρίς κατ' ανάγκη κάτι τέτοιο να θεωρείται κακό ή κακόβουλο. Αποτελεί φυσικό επακόλουθο έτσι ώστε για να μπορέσει η κοινωνία να προχωρήσει μπροστά και να εξελιχθεί.

Όμως, πολλές φορές οι εταιρίες για να αποφέρουν κέρδη στα ταμεία τους, καταχράζονται την επικοινωνία που τους προσφέρουν οι κινητές συσκευές και γίνονται ενοχλητικές (Spam calls).

1.2 Spam

Η πλέον οικονομική και πολυπληθή προώθηση προϊόντων και ιδεών γίνεται ευκολότερα με μαζική αποστολή ηλεκτρονικών μηνυμάτων, τα ονομαζόμενα Σπαμ (Spam). Σε έρευνα του 2011 έχουν αναφερθεί 7 τρισεκατομμύρια Spam μηνύματα με αποτέλεσμα πολλές χώρες να έχουν ποινικοποιήσει την αποστολή Spam. Για την ιστορία η λέξη Spam πρωτοεμφανίζεται τη δεκαετία του 1960 σε αμερικανικό προϊόν κονσέρβα τυποποιημένου κρέατος που εισαγόταν σε μεγάλες ποσότητες στο Ην. Βασίλειο. Ευρέως γνωστό έγινε το 1970 όταν οι Μόντυ- Πάιθονς στην εκπομπή «Το ιπτάμενο τσίρκο» μια ομάδα Βίκινγκς που τραγουδούσαν «Spam,Spam… ωραίο Spam… εξαιρετικό Spam».

Όμως από το 1980 χρησιμοποιείται η λέξη Spam σαν ακρώνυμο του Sales, Promotion and Marketing. Τέλος το 1998 καταχωρείται στο New Oxford Dictionary of English με την έννοια

«άσχετα ή μη αποδεκτά μηνύματα που στέλνονται στο ιντερνέτ σε μεγάλο αριθμό ομάδων, ειδήσεων ή χρηστών».

Ο όρος Spam αναφέρεται πλέον σε ότι ενοχλητικό γεγονός συμβαίνει στα ηλεκτρονικά μέσα. Για παράδειγμα Spam Emails και Spam Calls.

Η εφαρμογή που υλοποιήθηκε προσπαθεί να λύσει το πρόβλημα των Spam Calls.

2. Σκοπός και Περιγραφή Εφαρμογής

Σκοπός της εργασίας αυτής, όπως ήδη αναφέρθηκε, είναι η ανάπτυξη μιας εφαρμογής για κινητές συσκευές με όνομα «NBlockker». Στο κεφάλαιο αυτό αρχικά θα περιγραφεί ο λόγος για τον οποίο επιλέχθηκε να αναπτυχθεί αυτή η εφαρμογή. θα μιλήσουμε δηλαδή για το πρόβλημα που εντοπίσαμε στην κοινωνία, τον τρόπο που θέλουμε να βοηθήσουμε να μειωθεί η έκταση αυτού του προβλήματος με τη δημιουργία της εφαρμογής «Nblockker», και τέλος θα αναφερθούμε πιο αναλυτικά στις λειτουργίες που μπορούν να εκτελεστούν με την εφαρμογή μας.

2.1 Εντοπισμός του προβλήματος

Για να δημιουργηθεί η εφαρμογή αυτή λήφθηκαν ερεθίσματα από την σύγχρονη κοινωνία μας. Πάρα πολλές εταιρίες με δόλιο τρόπο ενοχλούν τον κόσμο πλέον μέσω των κινητών συσκευών για να προωθήσουν δικά τους προϊόντα.

Το φαινόμενο αυτό παρατηρείται πλέον σε παγκόσμια κλίμακα. Η κατάσταση πλέον έχει ξεφύγει φτάνοντας το 2020 τις 31.3 δισεκατομμύρια Spam κλήσεις το οποίο είναι αυξημένο κατά 18% συγκριτικά με το 2019.

2.2 Λύση στο πρόβλημα των Spam Calls

Η λύση στο πρόβλημα των Spam Calls είναι να δημιουργηθεί ένα δίκτυο – μια κοινή βάση δεδομένων με αριθμούς spam. Το βασικό πρόβλημα σε αυτήν την προσέγγιση είναι ο τρόπος με τον οποίο θα γεμίσει η βάση δεδομένων. Και η λύση του προβλήματος είναι να «γεμίσει» η βάση δεδομένων είναι να την «γεμίσουν» οι ίδιοι οι χρήστες. Για να δημιουργηθεί η βάση δεδομένων χρειάζεται η καταχώρηση των Spam αριθμών να γίνει από τους ίδιος τους χρήστες έτσι ώστε οι αριθμοί να είναι διαθέσιμοι στο δίκτυο από όλους όσους κάνουν αναζητήσεις μέσω αυτού.

Ψευδής δήλωση αριθμού spam από τους χρήστες

Για να αντιμετωπιστεί το φαινόμενο οι χρήστες να δηλώνουν ψευδείς αριθμούς ως spam, δηλαδή το αντίστροφο φαινόμενο από το spam, θα χρησιμοποιείται ένα κατώφλι των 100 αναφορών ανά δηλωθέν αριθμό για μπορέσει μετά να είναι διαθέσιμο στους υπόλοιπους χρήστες.

2.3 Περιγραφή λειτουργίας Εφαρμογής

Η εφαρμογή αναπτύχθηκε να είναι προσιτή, κατανοητη και εύχρηστη για όλους τους χρήστες. Παρακάτω παρατίθενται οι διαδικασίες που μπορεί να εκτελέσει ο χρήστης και οι δυνατότητες ειδοποίησης του χρήστη από την ίδια την εφαρμογή

Ο χρήστης αρχικά θα πρέπει να εγγραφή στο δίκτυο δημιουργώντας έναν λογαριασμό δηλώνοντας ένα email ή μέσω του google account. Αφού δημιουργήσει τον λογαριασμό θα έχει την δυνατότητα να δηλώσει αριθμούς ως spam.

Έτσι εμπλουτίζεται η βάση δεδομένων με αριθμούς οι οποίο ενοχλούν συστηματικά τους χρήστες.

Ακόμα για να αποφευχθεί η εσκεμμένη εισαγωγή αριθμών ως σπαμ, δηλαδή να εκμεταλλευτεί κάποιος την εφαρμογή ώστε να μπλοκάρει κάποιους αριθμούς για δικό του όφελος, έχει μπει ένα όριο των 30 δηλώσεων ανά αριθμό και μετά από αυτό το όριο ο αριθμός θα θεωρείται ως spam και θα μπορεί να χρησιμοποιηθεί από την εφαρμογή.

Αυτό το όριο μπορεί είναι στατικό αλλά μπορεί να μεταβληθεί κατά το δοκούν από τον πηγαίο κώδικα της εφαρμογής.

Έπειτα όταν ένας χρήστης δέχεται μια κλίση από έναν τέτοιο αριθμό τότε την ώρα που τον καλεί αυτός ο αριθμός εμφανίζεται ένα Toast message το οποίο αναφέρει τον τύπο spam του καλούντος. Περισσότερες λεπτομέρειες μαζί με εικόνες στα επόμενα κεφάλαια.

2.4 Περιγραφή τεχνολογιών της εφαρμογής

Η λειτουργία της εφαρμογής χωρίζεται σε 2 κομμάτια το κομμάτι της εφαρμογής η οποία τρέχει στην κινητή συσκευή και το κομμάτι που θα τρέχει στον εξυπηρετητή. Και τα δυο κομμάτια είναι απαραίτητα ώστε να δημιουργηθεί μια διαδικτυακή εφαρμογή. Ο εξυπηρετητής είναι υπεύθυνος για την κεντρική διαχείριση των δεδομένων είτε είναι εισαγωγή είτε ανάγνωση δεδομένων είτε αυθεντικοποίηση - πιστοποίηση των χρηστών.

2.4.1 Περιγραφή Εξυπηρετητή - Βάσης δεδομένων (Server – Database)

Για την υλοποίηση του εξυπηρετητή χρησιμοποιήθηκε η τεχνολογία της Google, το Google Firebase.

Εικόνα 1 Firebase logo

Το Firebase είναι ένα Back-end as a Service περιλαμβάνει μια σουίτα Cloud προϊόντων με τα οποία μπορούν να αναπτυχθούν πολύ γρήγορα αλλά και ασφαλή πληροφοριακά συστήματα. Διαθέτει μια ευρεία γκάμα από συστήματα Authentication μέχρι Analytics.

Στην εικόνα μπορούμε να δούμε τις δυνατότητές του.

Εικόνα 2 Firebase functionality

2.4.2 Δυνατότητες του συστήματος Google Firebase που χρησιμοποιήθηκαν

Firebase Authentication

Είναι το σύστημα το οποίο ευθύνεται για την αυθεντικοποίηση των χρηστών και την διαχείριση αυτών. Μέσω των διαθέσιμων SDKs σου δίνει την δυνατότητα να εγγραφούν και να πιστοποιηθούν οι χρήστες που χρησιμοποιούν την εφαρμογή.

Στην εφαρμογή χρησιμοποιήθηκαν 2 τρόποι αυθεντικοποίησης χρηστών.

1. Μέσω Google Account
2. Μέσω E-mail

Για την κάθε περίπτωση χρησιμοποιήθηκε αρχικά η δυνατότητα εγγραφής των χρηστών και σε δεύτερο χρόνο το σύστημα αυθεντικοποίησης.

Ότι περιγράφεται παραπάνω υλοποιήθηκε μέσω πηγαίου κώδικα της εφαρμογής που τρέχει στην κινητή συσκευή και αυτό ακριβώς είναι το τεράστιο πλεονέκτημα που προσφέρει το παραπάνω σύστημα.

Firebase Firestore

Το παραπάνω σύστημα είναι μια cloud υπηρεσία βάσης δεδομένων. Είναι μια NoSQL βάση δεδομένων και αποτελείται από Documents. Παρακάτω παρατίθεται η ακριβής περιγραφή στην αγγλική γλώσσα από την Google.

Cloud Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, you store data in documents, which are organized into collections.

Each document contains a set of key-value pairs. Cloud Firestore is optimized for storing large collections of small documents.

All documents must be stored in collections. Documents can contain subcollections and nested objects, both of which can include primitive fields like strings or complex objects like lists.

Collections and documents are created implicitly in Cloud Firestore. Simply assign data to a document within a collection. If either the collection or document does not exist, Cloud Firestore creates it.

2.4.3 Περιγραφή τεχνολογιών εφαρμογή κινητής συσκευής(Mobile Application)

Η ανάπτυξη της εφαρμογής της κινητής συσκευής έγιναν με τεχνολογίες οι οποίες μπορούν να εκμεταλλευτούν απόλυτα τις δυνατότητες της εφαρμογής και να προσφέρουν το καλύτερο performance. Για αυτό το λόγο επιλέχθηκε η γλώσσα προγραμματισμού Kotlin η οποία έχει άμεση επικοινωνία με το Android SDK και Hardware SDK. Ακόμα έχει την δυνατότητα να επικοινωνεί με το Firebase SDK εγγενές το οποίο δίνει το καλύτερο performance από πλευράς διαχείρισης των δεδομένων και επικοινωνίας.

3. Τεχνολογίες Ανάπτυξης

Σε αυτό το κεφάλαιο θα γίνει μια πιο εκτενής ανάπτυξη και περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής. Θα περιγράφουν στα επόμενα κεφάλαια η γλώσσα προγραμματισμού που χρησιμοποιήθηκε, αλλά και του συστήματος Google Firebase.

Από τους διάφορους τύπους κινητών εφαρμογών επιλέχθηκε να δημιουργηθεί μια εγγενής εφαρμογή (Native App), διότι μέσω αυτών των τεχνολογιών προσφέρεται η μεγαλύτερη πρόσβαση στο SDK της συσκευής, το οποίο είναι απαραίτητο για τη λειτουργικότητα που έπρεπε να επιτευχθεί.

Εγγενείς Εφαρμογές (Native App)

Ο πλέον συνηθισμένος τύπος εφαρμογής είναι οι εγγενείς εφαρμογές. Οι εγγενείς εφαρμογές είναι γραμμένες σε γλώσσες προγραμματισμού που υποστηρίζει η εκάστοτε συσκευή ή πλατφόρμα. Σαν βοήθεια στους προγραμματιστές η Apple και η Google δίνουν τα δικά τους εργαλεία ανάπτυξης, στοιχεία διεπαφής και SDK. Το πλεονέκτημα της ανάπτυξης των εγγενών εφαρμογών σε σύγκριση με άλλου τύπου εφαρμογές οδήγησαν τις περισσότερες εταιρίες να επενδύσουν σε αυτό. Στα πλεονεκτήματα μπορούμε να αναφέρουμε

a) Την ταχύτητα
b) Την καλή απόκριση εφόσον είναι κατασκευασμένες για τη συγκεκριμένη πλατφόρμα
c) Την αξιοποίηση όλου του API και
d) Την προσθήκη βιβλιοθηκών έτσι ώστε οι προγραμματιστές να μπορούν να χρησιμοποιούν τις λειτουργίες της πλατφόρμας χωρίς να χρειάζεται να ανατρέχουν στο διαδίκτυο.

Σαν μειονέκτημα βέβαια είναι η δύσκολη ανάπτυξη από πλευράς κώδικα και η μη δυνατή χρήση τους σε άλλες πλατφόρμες.

Λειτουργικό Σύστημα Android

Ένα ελεύθερο λειτουργικό σύστημα που βασίζεται στον πυρήνα του λειτουργικού Linux αφού έχει υποστεί ορισμένες τροποποιήσεις, είναι το android. Σαν αρχή χρησιμοποιήθηκε στις κινητές συσκευές με οθόνη αφής, αλλά πλέον βρίσκεται σε έξυπνες συσκευές όπως, ρολόγια,

τηλεοράσεις, αυτοκίνητα, tablets. Η Android Inc αναπτύσσει αρχικά το λειτουργικό. Η Google μετά την έξαρση που παρουσιάζει η παρουσία του iPhone από την Apple το 2007 προχωράει στην αγορά της Android Inc. Η Google αφού ανέπτυξε το Android συνεργάστηκε με την Open Handset Alliance, μια κοινοπραξία από εταιρίες που ασχολούνται με τη δημιουργία λογισμικού, hardware και τηλεπικοινωνίων.

Η ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance στις 5 Νοεμβρίου 2007 είναι και η ημέρα της επίσημης παρουσίασης του λειτουργικού με τους προγραμματιστές να παίρνουν από την Google το μεγαλύτερο μέρος του κώδικα.

Στις 22 Οκτωβρίου 2008 το HTC Dream είναι η πρώτη συσκευή που παρουσιάζεται με διαθέσιμο το λειτουργικό σύστημα Android.

Εξαιτίας του ανοιχτού λογισμικού το λειτουργικό παρουσιάζει έξαρση χρήσης. Η πληθώρα των κατασκευαστών παγκοσμίως και δη των μικρότερων κατασκευαστών μπορούν πλέον να παράγουν συσκευές μικρού κόστους με λογισμικού ανοιχτού κώδικα που εύκολα προσαρμόζεται στα μετρά τους. Αυτό έχει σαν αποτέλεσμα αρκετοί προγραμματιστές να στραφούν σε αυτό το λειτουργικό.

Η γλώσσα προγραμματισμού που αξιοποιείται είναι η Java/Kotlin. Ο προγραμματιστής έχει την δυνατότητα δημιουργίας σύγχρονων εφαρμογών χρησιμοποιώντας το IDE Android Studio και το λογισμικό ανάπτυξης (Android SDK).

Φτάνοντας σήμερα στην δωδέκατη έκδοσή του βλέπουμε ραγδαία εξέλιξη με σημαντικές βελτιώσεις και εμπλουτισμό λειτουργιών.

3.1 Γλώσσα Προγραμματισμού Kotlin

Εικόνα 3 Kotlin logo

Η Kotlin είναι μία μοντέρνα, στατικού τύπου (statically typed), αντικειμενοστραφής, συμβατή με το λειτουργικό Android γλώσσα προγραμματισμού, η οποία έρχεται να διορθώσει αρκετά προβλήματα της γλώσσας προγραμματισμού Java όπως τις εξαιρέσεις τύπου NullPointer και τις υπερβολικές γραμμές κώδικα. Εκτελείται στην εικονική μηχανή της Java (JVM) και έχει τη δυνατότητα να χρησιμοποιήσει κλάσεις της Java άμεσα, γεγονός που αποτελεί μία από τις πλέον σύγχρονες τάσεις στις γλώσσες προγραμματισμού, δηλαδή γλώσσες που στηρίζονται σε JVM και συνεργάζονται με Java. Μπορεί να χρησιμοποιηθεί στην ανάπτυξη εφαρμογών διαφόρων πεδίων. Επιπλέον μπορεί να μεταγλωττιστεί στην JavaScript, με αποτέλεσμα να αποτελεί μία εξίσου δυνατή επιλογή και στην ανάπτυξη web εφαρμογών.

Η Kotlin σχεδιάστηκε από τους προγραμματιστές της JetBrains, μία εταιρεία που έδρευε στην Αγία Πετρούπολη της Ρωσίας και πλέον εδρεύει στην Τσεχία, η οποία είναι και η δημιουργός του πολύ γνωστού περιβάλλοντος ανάπτυξης λογισμικού (IDE), IntelliJ. Πήρε το όνομά της από το νησί Kotlin, που βρίσκεται κοντά στην Αγία Πετρούπολη. Για πρώτη φορά εμφανίστηκε το 2011 αλλά επίσημα κυκλοφόρησε το 2016 ως γλώσσα ανοικτού κώδικα. Συντακτικά μοιάζει αρκετά με την Java. Για την ακρίβεια σχεδιάστηκε με σκοπό να αποτελέσει μία καλύτερη εναλλακτική της Java. Ως αποτέλεσμα, υπάρχουν πολλά σημαντικά πλεονεκτήματα της χρήσης της Kotlin έναντι της Java στην ανάπτυξη λογισμικού.

Η Kotlin είναι εμπνευσμένη από γλώσσες όπως Swift, Scala, Groovy, C# και πολλές άλλες. Πραγματοποιήθηκε ενδελεχής ανάλυση πολλών γλωσσών ώστε να αποφευχθούν τα λάθη τους αλλά και να χρησιμοποιηθούν τα δυνατά χαρακτηριστικά τους. Έτσι δημιουργήθηκε μία γλώσσα ώριμη και καλά σχεδιασμένη, που φέρνει την ανάπτυξη εφαρμογών σε εντελώς νέο επίπεδο με την βελτίωση της ποιότητας του κώδικα, την ασφάλεια που παρέχει αλλά και την ενίσχυση της απόδοσης του προγραμματιστή. Διαθέτει πολύ ενεργή προγραμματιστική κοινότητα ενώ η υιοθέτησή της από την πλατφόρμα Android αναπτύσσεται ταχύτατα.

Γενικά, θα μπορούσε να περιγραφεί ως μία γλώσσα ασφαλής, εκφραστική, περιεκτική, ευέλικτη και φιλική στη χρήση εργαλείων, που έχει πολύ καλή διαπερατότητα με τις γλώσσες Java και JavaScript. Τον Μάιο του 2019 η Google, ανακοίνωσε την Kotlin ως προτιμώμενη γλώσσα για ανάπτυξη εφαρμογών Android. Το παραπάνω γεγονός σε συνδυασμό με τα προηγμένα χαρακτηριστικά της, έδωσαν ισχυρή ώθηση στην δημοτικότητά της.

3.2 Google Firebase

Εικόνα 4 Firebase logo

Η πλατφόρμα Firebase αναπτύχθηκε από την Google για την δημιουργία κινητών και web εφαρμογών με σκοπό να βοηθήσει τους προγραμματιστές να παραδίδουν πλουσιότερες εμπειρίες εφαρμογών. Η Firebase παρέχει ένα σύνολο εξαιρετικών εργαλείων για την απλοποίηση της εργασίας των προγραμματιστών. Τα εργαλεία αυτά είναι διασυνδεδεμένα, επεκτάσιμα και μπορούν να ενσωματωθούν με λογισμικά

τρίτων, ώστε να μπορούν να ξεπεράσουν ακόμη και τις πιο πολύπλοκες προκλήσεις.

Η πλατφόρμα Firebase αποτελείται από ένα μεγάλο σύνολο εργαλείων ανάπτυξης που αφορούν την δημιουργία λογισμικού (Building), τον έλεγχο ποιότητας (Quality) και την ενίσχυση των επιχειρήσεων (Business) που έχουν ως εξής:

- Building: Realtime Database, Cloud Firestore, Cloud Functions, Authentication, Hosting, Cloud Storage, ML Kit.
- Quality: Crashlytics, Performance Monitoring, Test Lab, App Distribution
- Business: Google Analytics, Predictions, In-App Messaging, A/B Testing, Cloud Messaging, Remote Config, Dynamic Links

Η Realtime Database και το Cloud Firestore μπορούν να αποθηκεύουν δεδομένα σε μορφή εγγράφου και να συγχρονίζουν τις αντίστοιχες εφαρμογές σε χιλιοστά του δευτερολέπτου, ώστε κάθε φορά που πραγματοποιείται κάποια αλλαγή στα δεδομένα, αυτές να ενημερώνονται άμεσα. Κάτι τέτοιο σημαίνει ότι τόσο η εφαρμογή όσο και η βάση δεδομένων “ακούνε” η μία την άλλη παρέχοντας στους χρήστες των εφαρμογών διαδραστικές εμπειρίες. Μάλιστα, οι Cloud Functions της Firebase επεκτείνουν ακόμη περισσότερο αυτή τη λειτουργικότητα, καθώς επιτρέπουν τον προγραμματιστή να γράψει κώδικα που παραδοσιακά θα γραφόταν σε ένα πρόγραμμα που θα έτρεχε στο εξυπηρετητή, χωρίς να απαιτείται η εμπλοκή του με οποιουδήποτε είδους εξυπηρετητή (server). Για παράδειγμα, μία Cloud Function μπορεί να στέλνει μία ειδοποίηση στην εφαρμογή κάθε φορά που κάτι συμβαίνει στην βάση. Με την Authentication της Firebase μπορεί να γίνει ταυτοποίηση χρηστών τόσο μέσα από τα κοινωνικά δίκτυα όσο και από λογαριασμούς ηλεκτρονικού ταχυδρομείου αλλά και τηλεφωνικούς αριθμούς, για την ασφαλέστερη διαχείρισή τους. Το Firebase Hosting μπορεί να αναπτύξει τόσο στατικό όσο και δυναμικό περιεχόμενο σε web εφαρμογές με ταχύτητα και ασφάλεια ενώ το Cloud Storage μπορεί να συγκεντρώσει και να εξυπηρετήσει τεράστιο αριθμό αρχείων. Επιπλέον το ML Kit φέρνει στις εφαρμογές το machine learning με έτοιμα προς χρήση APIs και προσαρμοσμένα μοντέλα, χρησιμοποιώντας την TensorFlow Lite, μια machine learning πλατφόρμα ανοιχτού κώδικα της Google.

Η Firebase έχει επιπλέον τη δυνατότητα να βελτιώσει την απόδοση της εφαρμογής όταν ολοκληρωθεί η ανάπτυξή της. Η ενότητα Crashlytics δίνει μία επισκόπηση σε πραγματικό χρόνο, των ζητημάτων που ενδέχεται να συναντήσουν οι χρήστες. Η απόδοση της εφαρμογής μπορεί επίσης να παρακολουθείται με την χρήση των Performance Monitoring και Test Lab εργαλείων, που δοκιμάζουν την εφαρμογή σε μία γκάμα συσκευών για να εξασφαλίσουν ότι όλοι οι χρήστες

θα έχουν την ίδια εμπειρία από την εφαρμογή. Τέλος, η App Distribution διανέμει προς έλεγχο σε έμπιστους χρήστες, νέες εκδόσεις της εφαρμογής πριν από την κυκλοφορία τους.

Ως προς την ανάπτυξη των επιχειρήσεων η πλατφόρμα διαθέτει τα δικά της εργαλεία ανάλυσης (Google Analytics) και πρόβλεψης (Predictions) για την καλύτερη κατανόηση της συμπεριφοράς χρηστών και την στοχοποίηση συγκεκριμένου κοινού με τη χρήση του In-App Messaging. Το εργαλείο A/B Testing χρησιμοποιείται για τον πειραματισμό με νέες λειτουργικότητες, το Cloud Messaging αποστέλλει στοχευμένα μηνύματα και ειδοποιήσεις ενώ το Remote Config τροποποιεί την εφαρμογή χωρίς την ανάπτυξη νέας έκδοσης. Τέλος, τα Dynamic Links είναι έξυπνα URLs που επιτρέπουν την μεταφορά των υπαρχόντων και μελλοντικών χρηστών σε οποιαδήποτε τοποθεσία εντός της εφαρμογής.

4. Λειτουργίες Εφαρμογής και Περιπτώσεις Χρήσης

Δεδομένα

Δημιουργία εφαρμογής σχεδιασμός και ανάπτυξη εγγενούς Android εφαρμογής σε Kotlin για τον εντοπισμό και την αποφυγή ανεπιθύμητων εισερχομένων τηλεφωνικών κλήσεων.

Ζητούμενα

1. Εγγραφή νέου χρήστη στην εφαρμογή
2. Σύνδεση στο σύστημα
    a. Σύνδεση με email
    b. Σύνδεση μέσω Google Authentication
3. Επιβεβαίωση email
4. Εισαγωγή αριθμού ως ανεπιθύμητου αριθμού στην βάση δεδομένων.
    a. Από την λίστα κλήσεων της συσκευής
    b. Μέσω ειδοποιήσεις του συστήματος
5. Εμφάνιση των δηλωθέντων αριθμών του χρήστη στην βάση δεδομένων.

4.1 Λειτουργίες Εφαρμογής

1. Εγγραφή νέου χρήστη στην εφαρμογή

Περιγραφή λειτουργίας

Με αυτή την λειτουργία ο χρήστης θα εγγράφεται στη βάση δεδομένων και έπειτα θα μπορεί να καταχωρεί δεδομένα. Η εγγραφή είναι αναγκαία για να είναι ελεγχόμενη η εισαγωγή των ενοχλητικών αριθμών στην βάση δεδομένων. Τα στοιχεία του χρήστη που χρειάζονται είναι είτε ένας λογαριασμός Google ή ένα email από άλλο πάροχο.

Ακολουθίες ερεθίσματος/απόκρισης

Ο χρήστης θα εισάγει τα δεδομένα στην φόρμα και θα επιλέγει “Register” ή θα μπορεί να επιλέγει “Google Sign In”.

Ανάλυση σε λειτουργικές Απαιτήσεις

Όλα τα πεδία πρέπει να έχουν πληροφορίες, δηλαδή δεν θα πρέπει να υπάρχουν κενά κελιά.

2. Σύνδεση στο σύστημα

Περιγραφή Λειτουργίας

Όπως και η εγγραφή, έτσι και η σύνδεση στο σύστημα είναι απαραίτητη για τον ίδιο λόγο.

Ακολουθίες ερεθίσματος / απόκρισης

Ο χρήστης εισάγει το username και τον κωδικό πρόσβασης και επιλέγει «Login». Ακόμα μπορεί να επιλέξει την επιλογή “Google Sign in”

Ανάλυση σε λειτουργικές απαιτήσεις

Ο χρήστης πρέπει να πληκτρολογήσει σωστά το όνομα χρήστη ή το email του και τον κωδικό του, έτσι ώστε να εισέλθει στο σύστημα.

Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3. Επιβεβαίωση email

Περιγραφή λειτουργίας

Αφού ολοκληρώσει τη διαδικασία της εγγραφής θα πρέπει να επιβεβαιώσει ότι είναι κάτοχος του δηλωθέντος email, μέσω αποστολής αυτοματοποιημένου email.

Ανάλυση σε λειτουργικές απαιτήσεις

Απαραίτητη προϋπόθεση ο χρήστης να έχει κάνει εγγραφή με ένα «ενεργό» email.

Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

4. Εισαγωγή αριθμού ως ανεπιθύμητου στη βάση δεδομένων

Περιγραφή λειτουργίας

Ο χρήστης μπορεί να επιλέξει από τη λίστα «Recent Calls» τον αριθμό που θέλει να δηλώσει ως ανεπιθύμητο και αφού εισάγει το πεδίο «Comment» και επιλέξει και τον τύπο του καλούντος από τη διαθέσιμη λίστα.

Αυτή η λειτουργία μπορεί να εκτελεστεί είτε μέσω από την εφαρμογή, είτε μέσω ειδοποιήσεις της εφαρμογής.

Ανάλυση σε λειτουργικές απαιτήσεις

Ο χρήστης μπορεί να επιλέξει όσες αριθμούς επιθυμεί και να τους εισάγει. Δεν υπάρχει κάποιος περιορισμός στους πόσους αριθμούς θα εισάγει ο χρήστης.

Ανάλυση σε μη λειτουργικές απαιτήσεις

Το πεδίο «Comments» δεν είναι απαραίτητο για τη δήλωση του αριθμού στη βάση δεδομένων.

5. Εμφάνιση των δηλωθέντων αριθμών του χρήστη στη βάση δεδομένων.

Περιγραφή λειτουργίας

Ο χρήστης μπορεί να δει τη λίστα με τους αριθμούς τους οποίους έχει δηλώσει στη βάση δεδομένων.

Ανάλυση σε λειτουργικές απαιτήσεις

Ο χρήστης θα πρέπει να έχει κάνει log-in με κατάλληλο email για να μπορέσει να δει την λίστα

με τα μηνύματα.

Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

5. Υλοποίηση Εφαρμογής

Σε αυτό το κεφάλαιο θα παρουσιαστούν κομμάτια κώδικα από την υλοποίηση της εφαρμογής, αλλά και βασικές προσεγγίσεις για τη δημιουργία ένα UI εν έτη 2022. Μετά από χρόνια ανάπτυξης πολλών εφαρμογών και πολλές συζητήσεις μεταξύ των 2 επικρατέστερων φιλοσοφιών ανάπτυξης, οι οποίες είναι η Imperative και η Declarative, έχει επικρατήσει πλέον η Declarative προσέγγιση.

Θα περιγράφουν αυτές οι προσεγγίσεις στο πεδία ανάπτυξης ενός UI. Αλλά γενικότερα αυτές οι 2 φιλοσοφίες ανάπτυξης μπορούν να εφαρμοστούν σε όλο το ευρύ φάσμα του προγραμματισμού και όχι απαραίτητα στην ανάπτυξη ενός UI.

5.1 Imperative vs Declarative Programming

Ακολουθεί ένα παράδειγμα για να μπορέσουμε να εξηγήσουμε την διαφορά των 2 προσεγγίσεων.

Σενάριο

«Είμαι στα Wal-Mart. Πώς να έρθω σπίτι σου;»

Imperative απάντηση (ΠΩΣ): «Βγες από τη βόρεια έξοδο του πάρκινγκ και στρίψε αριστερά. Ακολούθησε τον A-15 αυτοκινητόδρομο με πορεία προς τον Βορά. Βγες δεξιά στην έξοδο για τα ΙΚΕΑ. Συνεχίστε ευθεία και στο πρώτο φανάρι στρίψτε αριστερά και στο επόμενο φανάρι στρίψτε δεξιά. Το σπίτι μου είναι το νούμερα 10»

Declarative απάντηση (ΤΙ): «Η διεύθυνση μου είναι Παπαδοπούλου 10, Παγκράτι, Αθήνα Αττική».

Παρατήρηση

Όπως πιθανόν να συμπεράνατε στην declarative απάντηση υποθέσαμε ότι υπάρχει κάποια συσκευή gps που ξέρει τα imperative βήματα για να πάει στο σπίτι.

Πολλές, εάν όχι όλες οι Declarative προσεγγίσεις, έχουν εσωτερικά κάποιες imperative περιγραφές, ώστε να κάνουν τα βήματα που χρειάζονται.

Χρησιμοποιώντας την Declarative προσέγγιση αναπτύχθηκε η εφαρμογή.

5.1.1 Composables

Τα Composables είναι δομικά στοιχεία του καινούργιου Android framework, που είναι υπεύθυνο για τη δημιουργία οπτικών στοιχείων στις κινητές συσκευές, Jetpack Compose.

Κάθε Composable είναι μια μέθοδος η οποία αντιπροσωπεύει ένα οπτικό αντικείμενο στην οθόνη. Είτε αυτό είναι ένα Button είτε μια συλλογή από Buttons, Labels, Images κ.α. Τα Composables δημιουργούνται με Declarative προσέγγιση, γι' αυτό το λόγο αναφέρθηκαν στο προηγούμενο υπο-κεφάλαιο οι όροι Imperative και Declarative.

Μια οθόνη συντίθεται από πολλά Composables τα οποία επικοινωνούν μεταξύ τους, αλλά μέσω του Core κώδικα της εφαρμογής, ο οποίος είναι υπεύθυνος για την επικοινωνία με το Firebase.

Η λειτουργία των Composable είναι πολύ απλή και δομημένη. Κάθε Composable έχει ένα state και μόλις γίνει μια αλλαγή σε αυτό το state τότε «ξανά-ζωγραφίζεται» στην οθόνη με βάση το καινούργιο state.

Όταν θέλει ένα composable να ενημερώσει το state ενός άλλου composable τότε αυτό γίνεται μέσω Callbacks ή/και αλλαγής των παραμέτρων του 2^{ου} Composable.

Παρακάτω θα παρατεθούν κάποια κομμάτια κώδικα στα οποία θα περιγράφονται αυτές οι βασικές λειτουργίες. Θα περιγράφουν τα βασικότερα Composables τα οποία χρησιμοποιούνται για να δημιουργηθεί μια οθόνη. Θα γίνει μια απλή και όχι εκτενής ανάλυση της λειτουργίας τους, διότι η περεταίρω ανάλυση θα ξεφευγε από τα όρια της συγκεκριμένης αναφοράς.

Composable (OutlineTextField)

```
OutlinedTextField(
    modifier = Modifier.fillMaxWidth(),
    visualTransformation = PasswordVisualTransformation(),
    value = userPassword,
    label = {
        Text(text = "Password", style= TextStyle(color =
MaterialTheme.colors.primaryVariant))
    },
    onValueChange = {
        userPassword = it
    }
)
```

Εικόνα 5 Πεδίο εισαγωγής κωδικού

Το παραπάνω composable περιγράφει ένα Text field πεδίο.

Παράμετροι

Modifier

Περιγράφεται η οπτική κατάσταση του composable. Θα μπορούσαμε να το παρομοιάσουμε με το CSS στις web τεχνολογίες.

VisualTransformation

Animation το οποίο θα τρέξει όταν ο χρήστης πατήσει λάθος κωδικό (CSS Animations).

Value, Label

Data τα οποία καθορίζουν το περιεχόμενο του composable, στην προκειμένη περίπτωση αντιπροσωπεύουν την τιμή του πεδίου άλλα και την ετικέτα του.

Ο πιο παρατηρητικός αναγνώστης θα παρατηρήσει ότι η παράμετρος Label εμπεριέχει ένα composable. Αυτό είναι χαρακτηριστικό των Composables. Το κάθε composable μπορεί να πάρει σαν παράμετρο άλλο composable, το οποίο και μπορεί να χρησιμοποιήσει αναλόγως.

onValueChange

Παράμετρος, τύπου callback, η οποία μπορεί να περιέχει μια αναφορά σε μια callback μέθοδο ή ακόμα και την ίδια την υλοποίηση της μεθόδου. Στην προκειμένη περίπτωση περιέχεται η υλοποίηση της μεθόδου και όχι μια αναφορά σε μία callback μέθοδο.

Composable (LazyColumn)

```
LazyColumn(
    verticalArrangement = Arrangement.Top,
    horizontalAlignment = Alignment.CenterHorizontally,
    modifier = Modifier.fillMaxSize()
) {
    items(calls) {it ->
        ContactItem(call = it, homePageViewModel = homePageViewModel)
    }
}
```

Το παραπάνω composable αντιπροσωπεύει μια λίστα από άλλα Composables τύπου ContactItem. Η ιδιαιτερότητα αυτού του composable είναι ότι τα items που «ζωγραφίζει» τα δημιουργεί με lazy τρόπο. Δηλαδή εμφανίζει όσο Composables είναι εμφανίσιμα στην οθόνη. Αυτό επιτυγχάνει το καλύτερο performance για τη συσκευή διότι δεν σπαταλούνται resources για «αχρείαστα» στοιχεία.

```
verticalArrangement=Arrangement.Top
```

Το Composable στοιχίζει στον κάθετο άξονα τα «παιδιά» του, στο πάνω μέρος.

```
horizontalAlignment=Alignment.CenterHorizontally
```

Το Composable στοιχίζει οριζόντιο άξονα τα «παιδία» του, στο κέντρο.

```
modifier=Modifier.fillMaxSize()
```

Δηλώνει το Composable να καταλάβει όλον τον διαθέσιμο χώρο και σε πλάτος και σε ύψος.

Composable (Button)

```
Button(
    border = ButtonDefaults.outlinedBorder.copy(width = 1.dp),
    modifier = Modifier
        .fillMaxWidth()
        .height(50.dp),
    shape = RoundedCornerShape(50),
    onClick = {
        val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(token)
            .requestEmail()
            .build()

        val googleSignInClient = GoogleSignIn.getClient(context, gso)
        launcher.launch(googleSignInClient.signInIntent)
    },
    content = {
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically,
            content = {
                Icon(
                    tint = Color.Unspecified,
                    painter = painterResource(id =
R.drawable.googleg_standard_color_18),
                    contentDescription = null,
                )
                Text(
                    style = MaterialTheme.typography.button,
                    color = MaterialTheme.colors.onPrimary,
                    text = "Google"
                )
                Icon(
                    tint = Color.Transparent,
                    imageVector = Icons.Default.MailOutline,
                    contentDescription = null,
                )
            }
        )
    }
)
```

Εικόνα 6 Button

Στο παραπάνω composable φαίνεται η δυναμική των Composables. Όπως παρατηρείται στο παραπάνω κουμπί έχει γίνει μια σύνδεση από πολλά Composables, ώστε να δημιουργηθεί ένα

πιο προηγμένο κουμπί. Αμέσως παρακάτω θα αναλυθούν κάθε μια από τις παραμέτρους και τι πετυχαίνουν.

```
border = ButtonDefaults.outlinedBorder.copy(width = 1.dp)
```

Γραμμή στο περιθώριο του κουμπιού με πάχος γραμμής 1dp (density pixels).

```
modifier=Modifier.fillMaxWidth().height(50.dp)
```

Δηλώνει στο κουμπί να καταλάβει το όλο το διαθέσιμο πλάτος και να έχει ύψος 50 dp.

```
shape=RoundedCornerShape(50)
```

```
onClick={
        val gso =
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(token)
            .requestEmail()
            .build()

        val googleSignInClient = GoogleSignIn.getClient(context, gso)
        launcher.launch(googleSignInClient.signInIntent)
    },
```

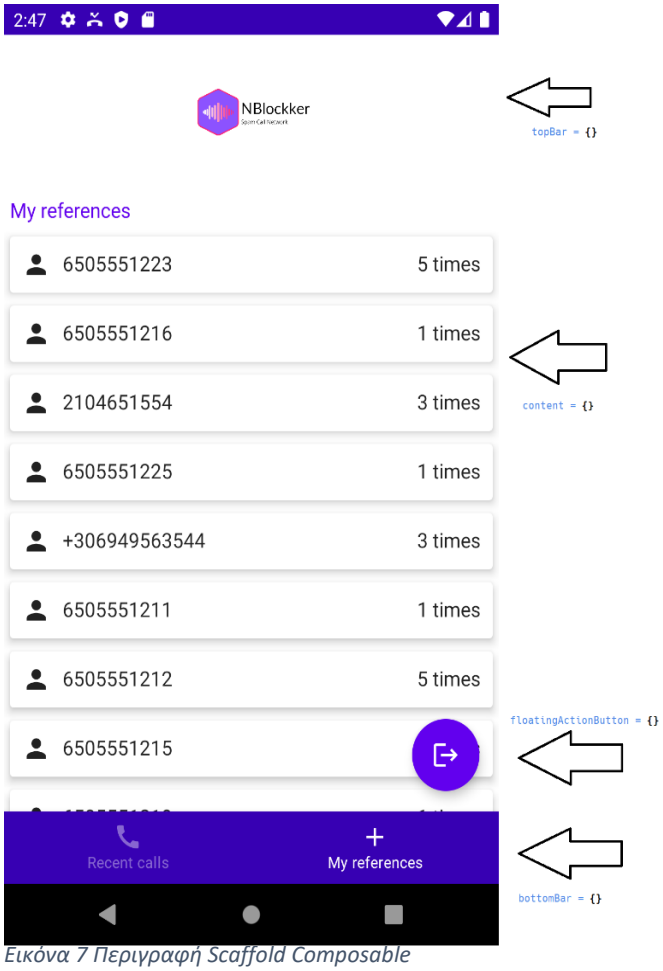
Εμπεριέχει την υλοποίηση της ρουτίνας η οποία θα τρέξει όταν ο χρήστης πατήσει το κουμπί.

```
content = {
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically,
            content = {
                Icon(
                    tint = Color.Unspecified,
                    painter = painterResource(id =
R.drawable.googleg_standard_color_18),
                    contentDescription = null,
                )
                Text(
                    style = MaterialTheme.typography.button,
                    color = MaterialTheme.colors.onPrimary,
                    text = "Google"
                )
                Icon(
                    tint = Color.Transparent,
                    imageVector = Icons.Default.MailOutline,
                    contentDescription = null,
                )
            }
        )
```

Δηλώνεται το σχήμα του κουμπιού ως Ορθογώνιο με καμπυλωμένες γωνίες. Η παραπάνω παράμετρος εμπεριέχει μια σειρά από Composables τα οποία συνθέτουν τη δομή και το περιεχόμενο του κουμπιού.

Composable(Scaffold)

```
Scaffold(topBar = {},
        floatingActionButtonPosition = FabPosition.End,
        floatingActionButton = {},
        drawerContent = {},
        content = {},
        bottomBar = {})
}
```



Εικόνα 7 Περιγραφή Scaffold Composable

Το παραπάνω composable περιγράφει τη βασική δομή μια σελίδας. Σε κάθε μία από τις παραπάνω παραμέτρους τοποθετούνται άλλα Composables, τα οποία θα συνθέσουν την οθόνη.

Με το παραπάνω composable μπορούν να δημιουργηθούν εύκολα καινούργιες οθόνες.

Παρέχεται η βασική δομή των Android οθονών και έτσι μπορούν οι προγραμματιστές να δημιουργήσουν πολλές διαφορετικές οθόνες αλλά κρατώντας μια ομοιογένεια σε όλη την εφαρμογή.

Αυτό είναι πολύ χρήσιμο διότι η εφαρμογή κρατάει μια οπτική δομή η οποία είναι απαραίτητη για την εμπορική επιτυχία μια εφαρμογής.

5.2 Γλώσσα προγραμματισμού Kotlin

Στο παρακάτω κεφάλαιο θα περιγράφουν βασικά κομμάτια της γλώσσας προγραμματισμού Kotlin, για να ενημερωθεί ο αναγνώστης για τη γλώσσα. Θα γίνει μια πολύ απλή εισαγωγή σε πάρα πολύ βασικά κομμάτια της γλώσσας. Δεν θα γίνει εκτενής ανάπτυξη, διότι αυτό το κεφάλαιο μπορεί να αποτελέσει από μόνο του ένα βιβλίο. Όποιος αναγνώστης ενδιαφέρεται να εντρυφήσει περεταίρω στο κομμάτι της γλώσσας υπάρχουν αρκετά βιβλία αλλά και πηγές στο διαδίκτυο για να αναπτύξει τις γνώσεις του.

5.2.1 Hello world

Η γενική μορφή ενός απλού προγράμματος Kotlin είναι η ακόλουθη:

```
fun main (args: Array) {
      println(“Hello World!”)
}
```

Η μέθοδος main δέχεται μία παράμετρο που ονομάζεται «args» και είναι τύπου Πίνακα από συμβολοσειρές (Array). Το σώμα της μεθόδου βρίσκεται εντός των συμβόλων «{}». Ο κώδικας μπορεί να σχολιαστεί με τη χρήστη των «//» όταν πρόκειται για μία γραμμή σχολίου, ή των «/* ..... */» όταν πρόκειται για πολλές. Τα αρχεία στα οποία γράφεται ο κώδικας Kotlin έχει επέκταση .kt.

Σημειώνεται ότι από την έκδοση 1.3 της Kotlin και μετά, η μέθοδος main μπορεί να γραφτεί και χωρίς τα ορίσματα ως εξής:

```
fun main () {
      println(“Hello World!”)
}
```

Η εκτέλεση του παραπάνω προγράμματος και στις δύο εκδοχές του, θα δώσει το αποτέλεσμα: Hello World!

5.2.2 Μεταβλητές(Variables)

Η Kotlin χρησιμοποιεί δύο διαφορετικά keywords για τη δήλωση μεταβλητών, var και val. Η διαφορά τους βρίσκεται στο ότι η λέξη var χρησιμοποιείται για να δηλώσει μία μεταβλητή της οποίας η τιμή μπορεί να αλλάξει, ενώ η λέξη val χρησιμοποιείται για τη δήλωση μεταβλητής στην οποία δεν μπορεί να ξαναγίνει ανάθεση τιμής (ουσιαστικά πρόκειται για σταθερά).

Στο παράδειγμα που ακολουθεί γίνεται ανάθεση τιμών σε δύο μεταβλητές age και name, οι οποίες δηλώνονται ως var και val αντιστοίχως. Στη συνέχεια γίνεται προσπάθεια να δοθούν νέες τιμές στις μεταβλητές αυτές με αποτέλεσμα η ανάθεση νέας τιμής στην πρώτη να πραγματοποιείται χωρίς πρόβλημα ενώ στην δεύτερη να αποτυγχάνει.

```
fun main (args: Array<String>) {
      var age = 28
      age = 30
      val name = "Tasos"
      name = "Nikos" //error
 }
```

Όπως παρατηρείτε ο τύπος δεδομένων στο παραπάνω παράδειγμα δεν αναγράφεται στην αρχικοποίηση των μεταβλητών. Σε αντίθεση με την Java, στην Kotlin δεν είμαστε υποχρεωμένοι να δηλώνουμε ακριβώς τον τύπο δεδομένων. Ο μεταγλωττιστής (compiler) μπορεί να τον «καταλάβει» από τις πληροφορίες που παρέχονται στην έκφραση. Παρόλα αυτά, αν θέλουμε να δηλώσουμε τον τύπο δεδομένων χρησιμοποιείται το σύμβολο «:» αμέσως μετά το όνομα της μεταβλητής και γράφουμε ακολούθως τον τύπο. Έτσι οι μεταβλητές στο παραπάνω παράδειγμα μπορούν να δηλωθούν ως εξής**: var age: Int = 28** και **val name: String = "Tasos"**.

5.2.3 Τύποι δεδομένων (Data types)

Μια από τις σημαντικότερες διαφορές στην Kotlin σε σχέση με την Java είναι ότι στην Kotlin όλα είναι αντικείμενα. Στη Java υπάρχουν οι βασικοί τύποι (primitive basic types) που δεν μπορούν να χρησιμοποιηθούν ως γενικοί τύποι (generic types), δεν υποστηρίζουν κλήση μεθόδων και δεν μπορούν να τους ανατεθούν τιμές null, όπως για παράδειγμα ο τύπος Boolean. Αυτό δεν ισχύει στην Kotlin καθώς όλοι οι βασικοί τύποι προωθούνται ως αντικείμενα. Ακολουθεί περιγραφή των βασικών τύπων δεδομένων.

Υπάρχουν 4 διαφορετικοί τύποι δεδομένων.

1. Integer Data type
2. Floating-point Data Type
3. Boolean Data Type
4. Character Data Type

Integer Data Types

Data Type	Bits	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32768	32767
int	32 bits	-2147483648	2147483647
long	64 bits	-9223372036854775808	9223372036854775807

Floating-Point Data Type:

Data Type	Bits	Min Value	Max Value
float	32 bits	1.40129846432481707e-45	3.40282346638528860e+38
double	64 bits	4.94065645841246544e-324	1.79769313486231570e+308

Boolean Data Type:

Data Type	Bits	Data Range
boolean	1 bit	true or false

Character Data Type:

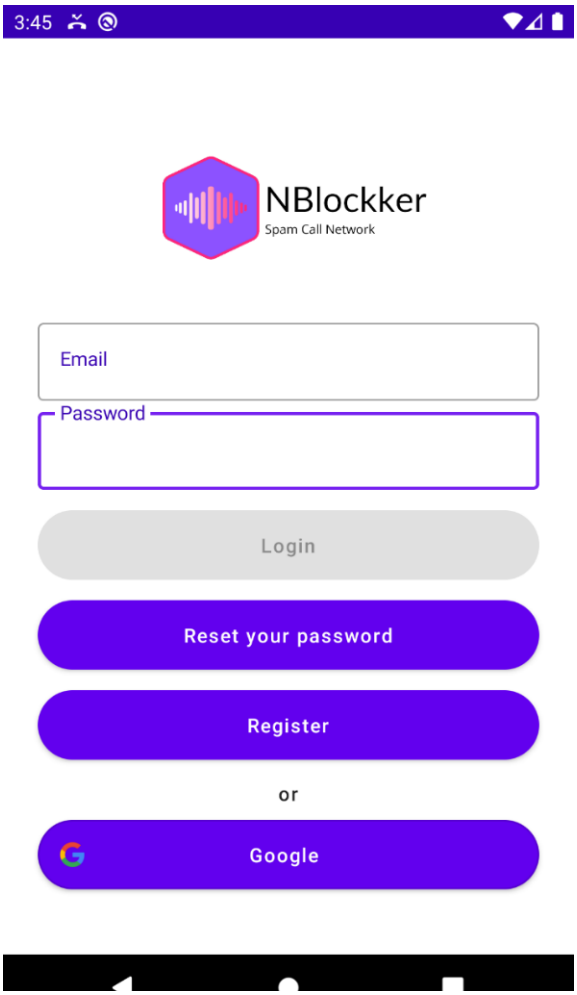
Data Type	Bits	Min Value	Max Value
Char	8 bits	-128	127

6. Εγχειρίδιο χρήσης της εφαρμογής

Σε αυτό το υποκεφάλαιο θα περιγραφούν οι περιπτώσεις χρήσης της εφαρμογής με εικόνες, ώστε να γίνει πιο κατανοητή η χρήση της εφαρμογής από τον αναγνώστη της αναφοράς.

Αρχική οθόνη

Παρακάτω παρουσιάζεται η αρχική οθόνη της εφαρμογής η οποία λειτουργεί ως αφετηρία για την εφαρμογή και παρέχει τις βασικές λειτουργίας της εγγραφής, εισόδου και επαναφοράς κωδικού του χρήστη.

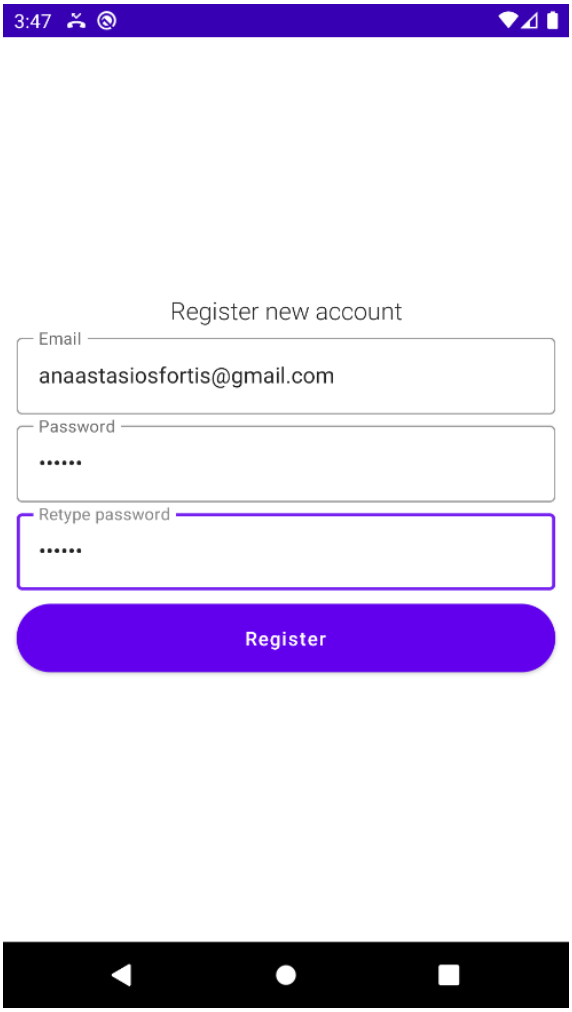


Εικόνα 8 Application Αρχική Οθόνη

Ο χρήστης εισάγοντας τα στοιχεία σύνδεσής του μπορεί να εισέλθει στην εφαρμογή. Ακόμα, εάν ο χρήστης έχει ήδη συνδεθεί, δεν χρειάζεται να επαναλάβει τη διαδικασία της σύνδεσης.

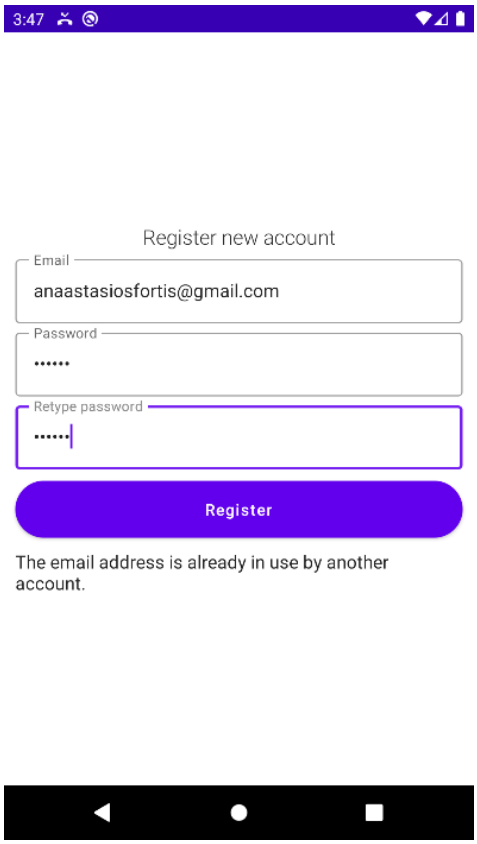
Εγγραφή νέου χρήστη

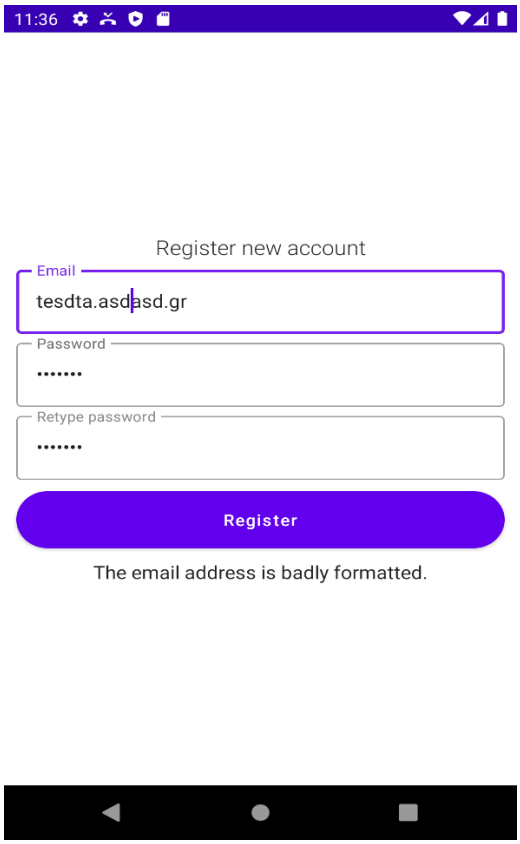
Από την προηγούμενη οθόνη πατώντας το κουμπί «Register» ο χρήστης ανακατευθύνεται στην παρακάτω οθόνη, στην οποία μπορεί να εισάγει ένα email και ένα κωδικό πρόσβασης για να κάνει εγγραφή.



Εικόνα 9 Application Εγγραφή νέου χρήστη

Εισάγοντας ένα email που έχει ήδη εγγραφεί στην εφαρμογή ή έχει λανθασμένο format, εμφανίζονται τα κατάλληλα μηνύματα, όπως φαίνονται στις παρακάτω εικόνες.





Εικόνα 10 Application Εγγραφή νέου χρήστη 1

Εικόνα 11 Application Εγγραφή νέου χρήστη 2

Είσοδος χρήστη

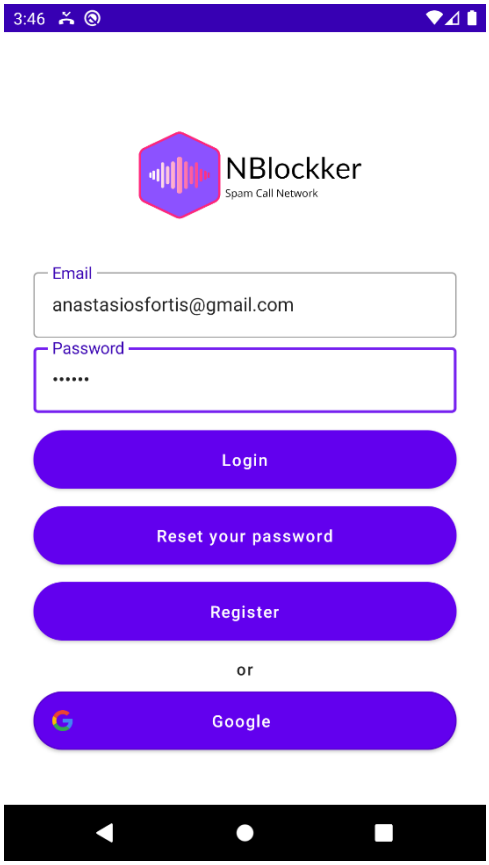
Στην παρακάτω οθόνη φαίνεται ξεκάθαρα ο τρόπος εισαγωγής των στοιχείων για είσοδο στην εφαρμογή.

Πατώντας το κουμπί «Login», μπορεί να ακολουθήσουν 3 σενάρια.

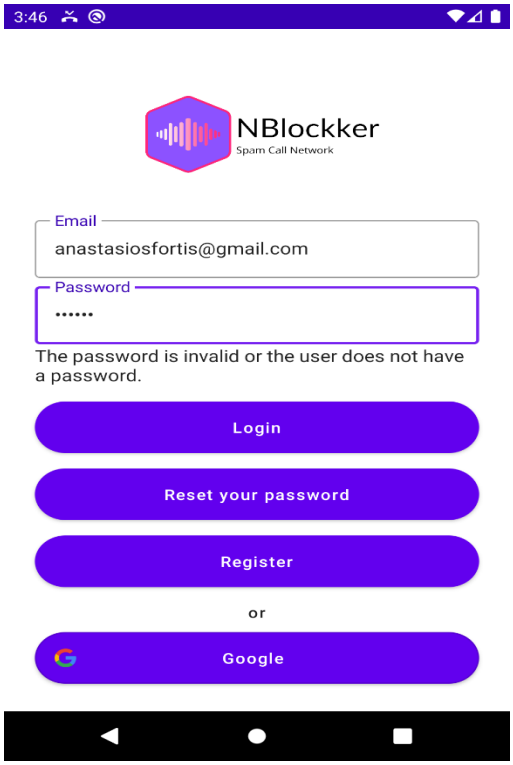
1. Λάθος κωδικός πρόσβασης, οπότε η εφαρμογή θα δείξει το κατάλληλο μήνυμα.

2. Να γίνει είσοδος στην εφαρμογή

3. Να θελήσει να επαναφέρει τον κωδικό πρόσβασης μέσω email.

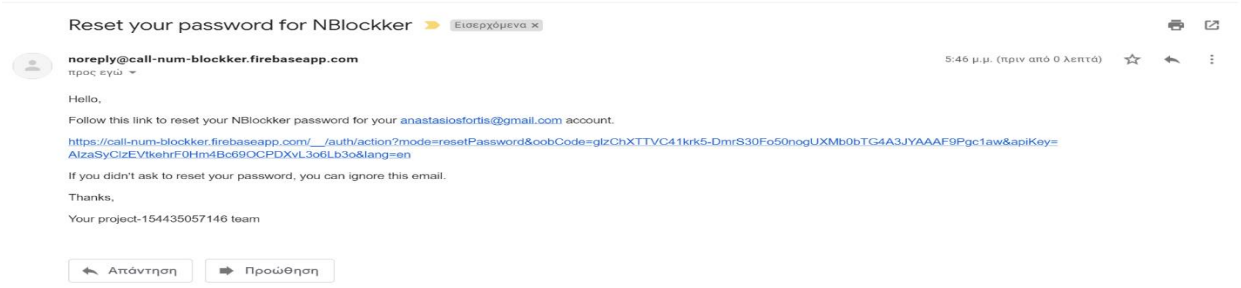


Εικόνα 12 Application Είσοδος χρήστη



Εικόνα 13 Application Λάθος κωδικός

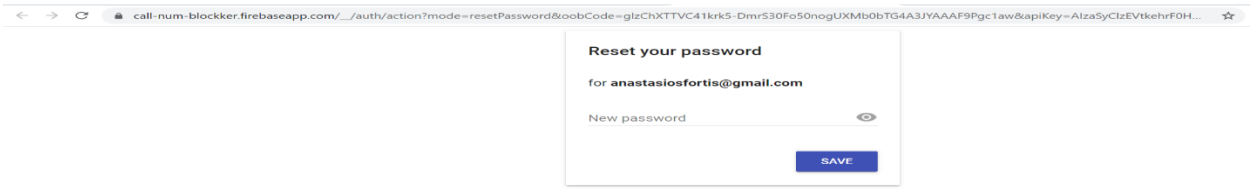
Εάν πατήσει το κουμπί «Reset your password» τότε θα σταλεί email μέσω του συστήματος Google Firebase Authentication, το οποίο θα περιέχει το κατάλληλο link για την επαναφορά του κωδικού πρόσβασης.



Εικόνα 14 Application Επαναφορά κωδικού

Παραπάνω φαίνεται το πρότυπο του email όπως το λαμβάνει ο χρήστης για την επαναφορά του κωδικού.

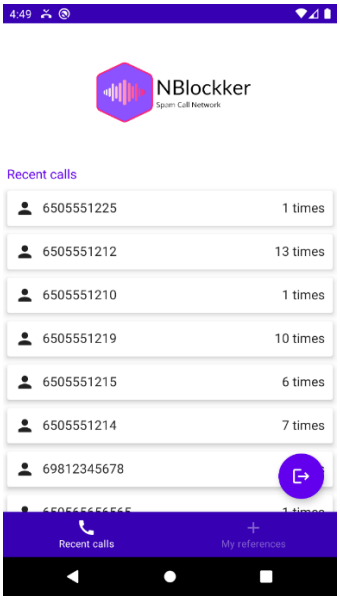
Το μόνο που έχει να κάνει ο χρήστης είναι να πατήσει πάνω στο link που περιέχει το μήνυμα και να δημιουργήσει καινούργιο κωδικό πρόσβασης. Όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 15 Application Επαναφορά κωδικού 2

Κύρια Οθόνη

Ακολουθεί η κύρια οθόνη από την οποία περιέχονται οι λειτουργίες της εφαρμογής.



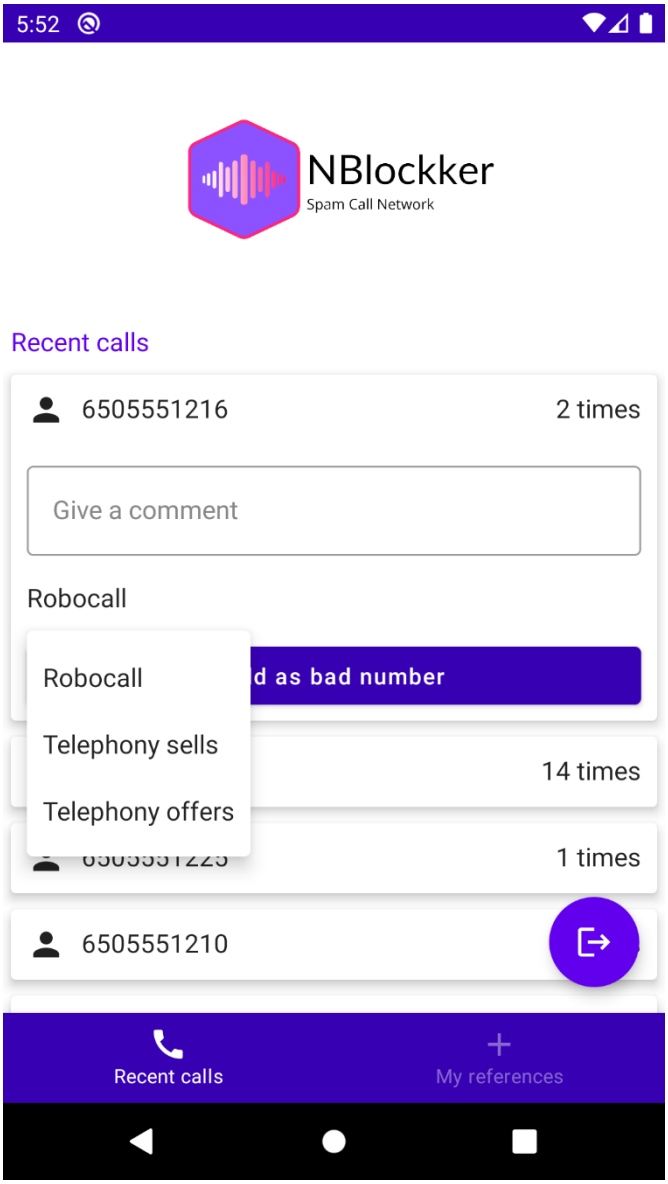
Εικόνα 16 Application Κύρια Οθόνη

Η κύρια οθόνη περιέχει 2 καρτέλες «Recent calls» και «My references».

Recent calls

Η καρτέλα αυτή διαβάζει τα πρόσφατες κλήσεις που έχει δεχθεί η εφαρμογή και τις εμφανίζει σε μια λίστα για να μπορέσει ο χρήστης να δηλώσει κάποιον αριθμό ως ενοχλητικό.

My references



Εικόνα 17 Application Κύρια Οθόνη 3

Η συγκεκριμένη καρτέλα περιέχει τις δικές του αναφορές ενοχλητικών αριθμών που έχει κάνει ο χρήστης στην εφαρμογή-σύστημα, μπορεί να δει λοιπόν τι αριθμούς έχει δηλώσει στη βάση δεδομένων.

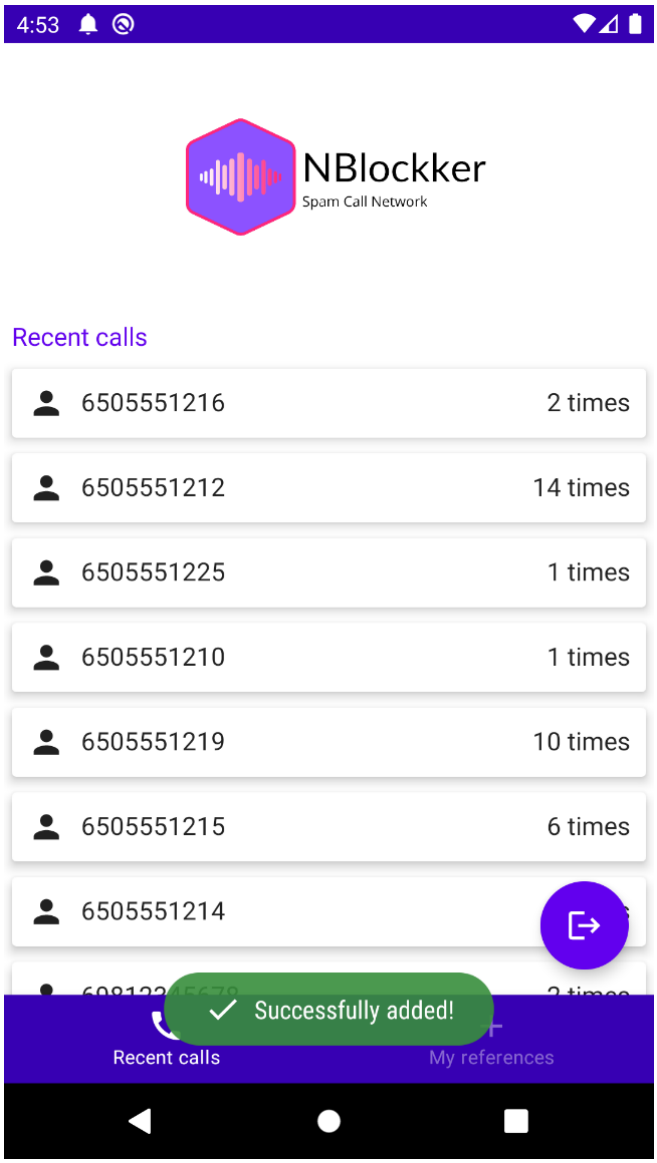
Δήλωση αριθμού ως ενοχλητικού

Ο χρήστης πατώντας πάνω σε έναν αριθμό εμφανίζεται το παρακάτω μενού επιλογών που μπορεί να δηλώσει.

Αρχικά μπορεί να δώσει ένα comment και μετά μπορεί να δηλώσει έναν τύπο ενοχλητικού αριθμού.

Σαν σχόλιο μπορεί να βάλει ό,τι επιθυμεί ο ίδιος, ώστε να είναι πιο περιγραφικός για το τι είδους αριθμός είναι. Αυτό μπορεί να αξιοποιηθεί στο μέλλον για περισσότερες δυνατότητες της εφαρμογής.

Μόλις ο χρήστης πατήσει «Add as bad number» ο αριθμός εισάγεται στη βάση δεδομένων και είναι αξιοποιήσιμος από όλο το δίκτυο της εφαρμογής.

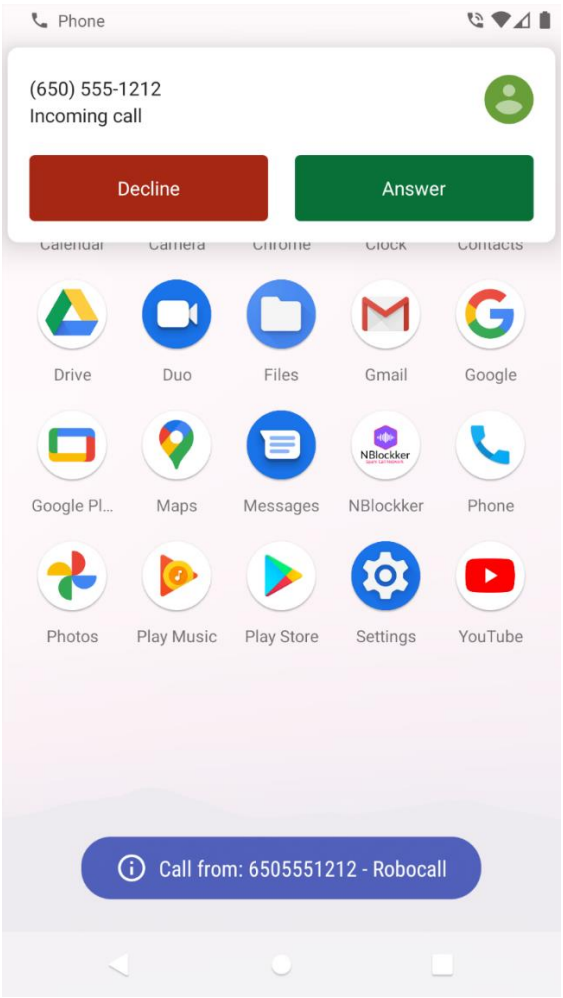


Εικόνα 18 Application Κύρια Οθόνη 4

Κλήση από ενοχλητικό αριθμό

Όταν κάποιος αριθμός καλεί τη συσκευή, τότε η εφαρμογή ελέγχει στη βάση δεδομένων εάν αυτός ο αριθμός έχει αναφερθεί από τους χρήστες ως ενοχλητικός.

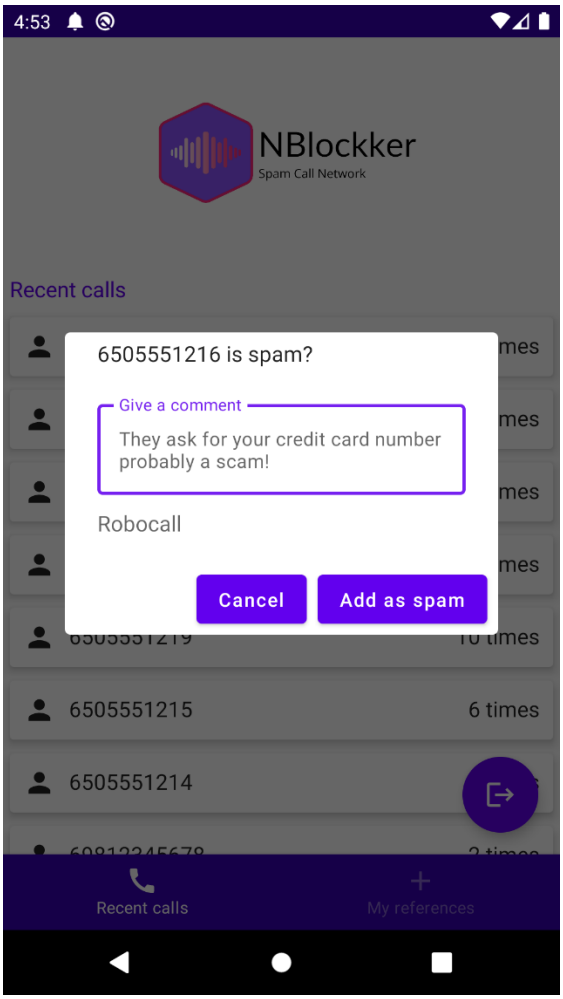
Εάν ο αριθμός έχει χαρακτηριστεί ως ενοχλητικός, τότε την ώρα που καλεί τη συσκευή εμφανίζεται ένα κατάλληλο μήνυμα που ειδοποιεί τον χρήστη.



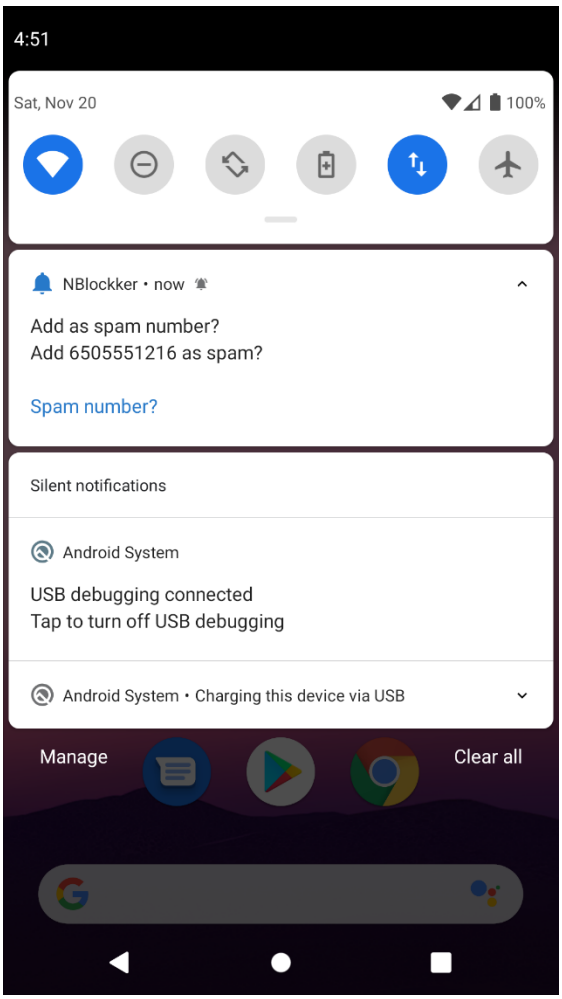
Εικόνα 19 Application Κλήση από ενοχλητικό αριθμό

Κλήση από πιθανό ενοχλητικό αριθμό

Επίσης, υπάρχει το σενάριο να καλέσει κάποιος αριθμός που δεν βρίσκεται στη βάση δεδομένων, γι' αυτό κάθε φορά που καλεί ένας άγνωστος αριθμός η εφαρμογή ειδοποιεί τον χρήστη με μία ειδοποίηση (Notification) και μέσω αυτής της ειδοποίησης μπορεί ο χρήστης να δηλώσει τον αριθμό ως ενοχλητικό.



Εικόνα 20 Εισαγωγή Spam αριθμού στην βάση δεδομένων



Εικόνα 21 Ειδοποίηση για πιθανό Spam αριθμό

7. Συμπεράσματα και Μελλοντικές επεκτάσεις

7.1 Συμπεράσματα

Η ανάπτυξη της εφαρμογής οδήγησε σε πολλά συμπεράσματα ως προς την ανάπτυξη μιας εφαρμογής για κινητές συσκευές. Οι τεχνολογίες και τα εργαλεία ανάπτυξης έχουν ωριμάσει και ωριμάζουν όσο περνάει ο χρόνος.

Πριν από 10 χρόνια η ανάπτυξη μια παρόμοιας εφαρμογής θα απαιτούσε πάρα πολύ βαθιά τεχνική γνώση πάνω στο SKD του Android λειτουργικού, αλλά και χρόνο.

Η επιλογή της google να ορίσει την Kotlin σαν κύρια γλώσσα ανάπτυξης για το λειτουργικό σύστημα Android ήταν ένα παράγοντας κλειδί, ώστε οι προγραμματιστές να αναπτύσσουν κώδικα ταχύτατα και με λιγότερα λάθη. Ακόμη, το μέγεθος του πηγαίου κώδικα της εφαρμογής έχει συρρικνωθεί σε πάρα πολύ μεγάλο βαθμό συγκριτικά με τον κώδικα που θα χρειαζόταν να αναπτυχθεί η εφαρμογή σε γλώσσα JAVA.

Οι ενοχλητικές κλήσεις είναι και θα είναι ένα μεγάλο πρόβλημα στην καθημερινότητα των ανθρώπων. Με αυτήν την εφαρμογή έγινε μια προσπάθεια ώστε να βρεθεί μια λύση στο πρόβλημα αυτό. Σαφώς και δεν αποτελεί την “De-facto” λύση πάνω σε αυτό το πρόβλημα, αλλά μέσω της συνεργασίας των ανθρώπων ίσως μειωθεί το συγκεκριμένο φαινόμενο.

7.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή μπορεί και πρέπει να αναπτυχθεί περεταίρω. Ενδεικτικά θα αναφερθούν κάποιες βελτιώσεις που μπορούν να γίνουν σε επίπεδο κώδικα και User-Interface, αλλά και σε γενικότερο κομμάτι ως προς την επιχειρηματική ανάπτυξη της εφαρμογής.

- Καλύτερο User-Interface / User experience
- Ανάπτυξη της εφαρμογής για το λειτουργικό σύστημα iOS, για μεγαλύτερη κάλυψη των χρηστών αλλά και μεγαλύτερη πηγή ενοχλητικών αριθμών.
- Ανάπτυξη ξεχωριστού server application για την υποστήριξη του back-end κομματιού αλλά και απεξάρτηση από το Google Firebase.
- Νέο λογότυπο το οποίο να αντιπροσωπεύει την εφαρμογή με όρους design.

8. Βιβλιογραφία

Polara, Vishal & Mahant, Chintan & Dalwadi, Bijal & patel, nikita. (2018). Kotlin The new android programming language. International Journal of Advanced Scientific Research & Development (IJASRD). 7.

Späth, Peter. (2019). Learn Kotlin for Android Development: The Next Generation Language for Modern Android Apps Programming. 10.1007/978-1-4842-4467-8.

Phiri, Hazael & Kunda, Douglas. (2017). A Comparative Study of NoSQL and Relational Database. Zambia ICT Journal. 1. 1-4. 10.33260/zictjournal.v1i1.8.

https://el.wikipedia.org/wiki/%CE%A3%CF%80%CE%B1%CE%BC

https://kotlinlang.org/

https://www.atomickotlin.com/atomickotlin/

https://www.oreilly.com/library/view/head-first-kotlin/9781491996683/

https://play.kotlinlang.org/

https://firebase.google.com/

https://developer.android.com/jetpack/compose

https://developer.android.com/

https://apothesis.eap.gr/handle/repo/49121

https://firebase.google.com/docs/android/setup

https://www.comparitech.com/blog/information-security/phone-spam-statistics/

https://kotlinlang.org/

https://gradle.org/

https://fonts.google.com/icons

https://www.tutorialspoint.com/kotlin/index.htm