Εργασία στο μάθημα «Σύγχρονα θέματα Τεχνολογίας Λογισμικού»

Ακαδημαϊκό έτος 2023-2024

Μέλη της εργασίας:
Αθηνά Αρμένη Π20025

Αλέξανδρος Βασίλειος Παναγόπουλος Π20150

Σωτήρης Χατζηκυριάκου Π20011

Επιβλέποντες Καθηγητές : Μανουσόπουλος Χρήστος, Αλέπης Ευθύμιος, Μαρία Βίρβου

ΠΕΙΡΑΙΑΣ

ΙΑΝΟΥΑΡΙΟΣ 2024

Περιεχόμενα:

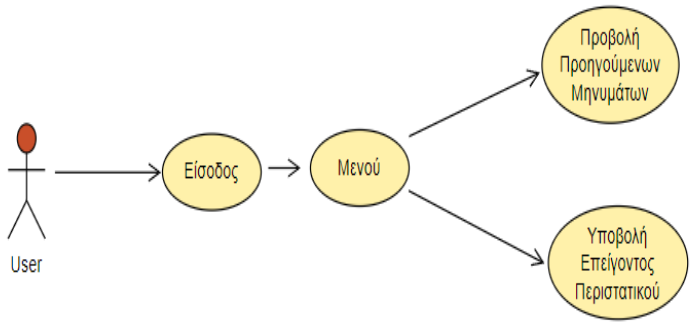
Συνοπτική επεξήγηση της εργασίας :

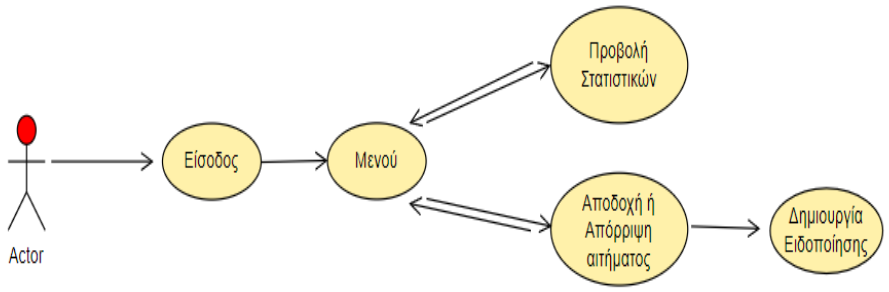
Σε αυτήν την εργασία καλούμαστε να υλοποιήσουμε μια εφαρμογή στο Android Studio την “SmartAlert” της οποίας ο σκοπός είναι η έγκαιρη ειδοποίηση των πολιτών σε έκτακτες περιστάσεις υψηλού κινδύνου. Στην ουσία καλείται να ενημερώνει τα 2 είδη χρηστών , οι οποίοι είναι οι εγγεγραμμένοι χρήστες και ο υπάλληλος πολιτικής προστασίας(παρακάτω θα εξηγήσουμε τους υπάρχοντες ρόλους), για καταστάσεις υψηλού κινδύνου(Πλημμύρες, Πυρκαγιές, Σεισμοί, Καταιγίδες και Ανεμοστρόβιλοι).

- Εγγεγραμμένος Χρήστης: Αρχικά με την έναρξη της εφαρμογής είναι απαραίτητο να δημιουργηθεί πρώτα ο χρήστης έτσι ώστε να συνδεθεί σε αυτήν . Με την ολοκλήρωση της εγγραφής εμφανίζεται κατάλληλο μήνυμα ότι ο χρήστης δημιουργήθηκε με επιτυχία , αποθηκεύονται τα στοιχεία του στη βάση δεδομένων και έτσι μπορεί να κάνει sign in χωρίς κανένα πρόβλημα. Με την είσοδο του στην εφαρμογή ο χρήστης μπορεί να δει το ιστορικό των μηνυμάτων που έχει στείλει ο ίδιος (όσον αφορά τα ακραία καιρικά φαινόμενα) ή να στείλει ένα καινούργιο μήνυμα ώστε να ειδοποιήσει και τους υπόλοιπους χρήστες άλλα και τον admin του SmartAlert.
- Υπάλληλος πολιτικής προστασίας: Ο υπάλληλος προστασίας δεν μπορεί να κάνει εγγραφή όπως ο χρήστης επειδή είναι μόνο ένας και έχει δημιουργηθεί στην βάση δεδομένων που έχουμε. Ο σκοπός του admin είναι να ελέγχει όλα τα μηνύματα των χρηστών που στέλνονται στον ίδιο για να αναφέρουν κάποιο περιστατικό και καλείται , ανάλογα με τον αριθμό των χρηστών που έχουν στείλει το ίδιο περιστατικό , την επικινδυνότητα του και το χρονικό διάστημα ανάμεσα στους χρήστες , να επιλέξει αν όντως είναι υψηλής επικινδυνότητας και ότι πρέπει να στείλει ειδοποίηση σε όλους τους χρήστες της εφαρμογής ή δεν είναι κάποιο σοβαρό συμβάν . Εκτός αυτού μπορεί να έχει πρόσβαση στα στατιστικά των φαινομένων που στέλνονται από τους χρήστες ώστε να

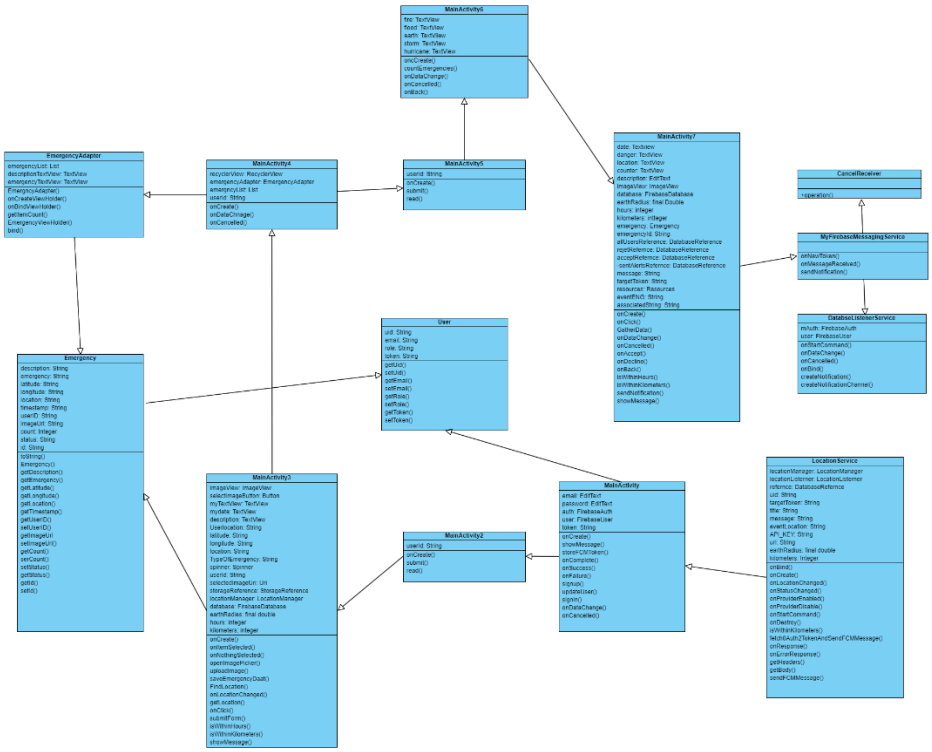
έχει μια πλήρη εικόνα για το ποιο καιρικό φαινόμενο είναι πιο συχνό , επομένως και το πιο επικίνδυνο .

Παρακάτω θα σας δείξουμε τα διαγράμματα τάξεων και περιπτώσεων χρήσης

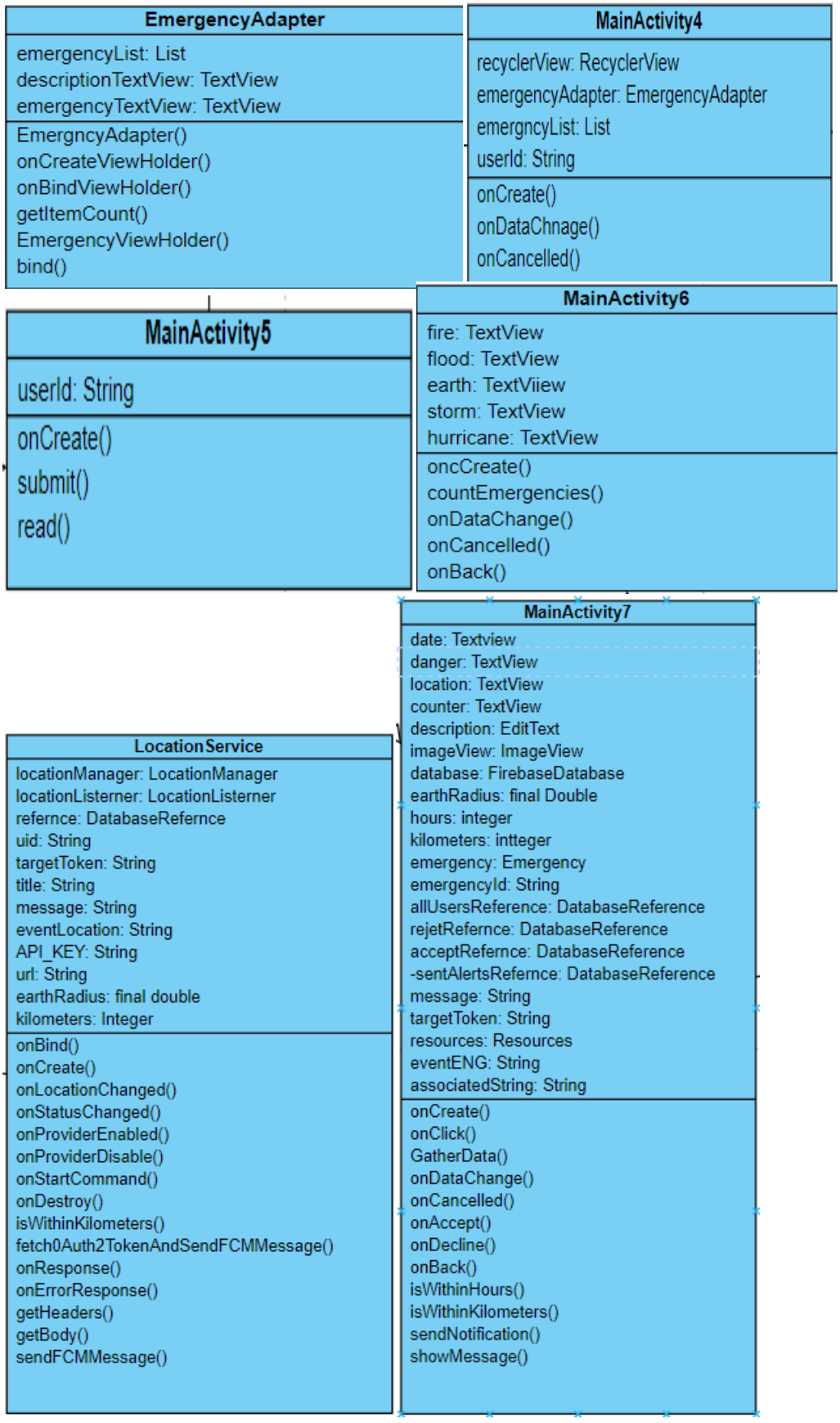


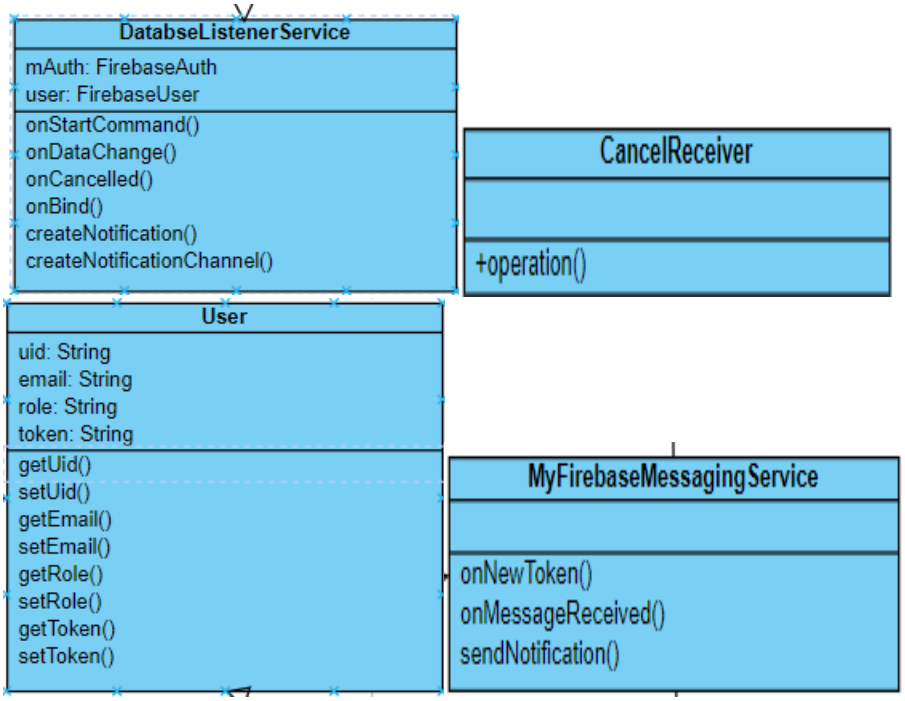


Διάγραμμα Περιπτώσεων χρήστης



Διάγραμμα τάξεως


EmergencyAdapter
emergencyList: List
descriptionTextView: TextView
emergencyTextView: TextView
EmergncyAdapter()
onCreateViewHolder()
onBindViewHolder()
getItemCount()
EmergencyViewHolder()
bind()
MainActivity4
recyclerView: RecyclerView
emergencyAdapter: EmergencyAdapter
emergncyList: List
userId: String
onCreate()
onDataChnage()
onCancelled()
MainActivity5
userId: String
onCreate()
submit()
read()
MainActivity6
fire: TextView
flood: TextView
earth: TextViiew
storm: TextView
hurricane: TextView
oncCreate()
countEmergencies()
onDataChange()
onCancelled()
onBack()
MainActivity7
date: Textview
danger: TextView
location: TextView
counter: TextView
description: EditText
imageView: ImageView
database: FirebaseDatabase
earthRadius: final Double
hours: integer
kilometers: intteger
emergency: Emergency
emergencyId: String
allUsersReference: DatabaseReference
rejetRefernce: DatabaseReference
acceptRefernce: DatabaseReference
-sentAlertsRefernce: DatabaseReference
message: String
targetToken: String
resources: Resources
eventENG: String
associatedString: String
onCreate()
onClick()
GatherData()
onDataChange()
onCancelled()
onAccept()
onDecline()
onBack()
isWithinHours()
isWithinKilometers()
sendNotification()
showMessage()
LocationService
locationManager: LocationManager
locationListerner: LocationListerner
refernce: DatabaseRefernce
uid: String
targetToken: String
title: String
message: String
eventLocation: String
API_KEY: String
url: String
earthRadius: final double
kilometers: Integer
onBind()
onCreate()
onLocationChanged()
onStatusChanged()
onProviderEnabled()
onProviderDisable()
onStartCommand()
onDestroy()
isWithinKilometers()
fetch0Auth2TokenAndSendFCMMessage()
onResponse()
onErrorResponse()
getHeaders()
getBody()
sendFCMMessage()


DatabseListenerService
mAuth: FirebaseAuth
user: FirebaseUser
onStartCommand()
onDataChange()
onCancelled()
onBind()
createNotification()
createNotificationChannel()
CancelReceiver
+operation()
User
uid: String
email: String
role: String
token: String
getUid()
setUid()
getEmail()
setEmail()
getRole()
setRole()
getToken()
setToken()
MyFirebaseMessagingService
onNewToken()
onMessageReceived()
sendNotification()

Επεξήγηση του κώδικα :

Κατά την έναρξη της εφαρμογής ο οποιοδήποτε χρήστης ξεκινάει από το αρχικό activity όπου μπορούν να δημιουργηθεί ο κάθε user και να συνδεθούν

ως χρήστες ή ως admin της εφαρμογής . Για την εγγραφή του κάθε χρήστη χρησιμοποιείται η μέθοδος signup

```
public void signup(View view) {
    if (!email.getText().toString().isEmpty() &&
            !password.getText().toString().isEmpty()) {
        auth.createUserWithEmailAndPass-
word(email.getText().toString().trim(), password.getText().toString())
                .addOnCompleteListener(new OnCompleteListener<AuthResult>()
{
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task)
{
                        if (task.isSuccessful()) {
                            user = auth.getCurrentUser();
                            updateUser(user);
                            showMessage("Success", "User profile cre-
ated!");

                            // Store FCM token after user signup
                            storeFCMToken();

                            // Write user data to Firebase Realtime Data-
base
                            if (user != null) {
                                DatabaseReference usersRef = FirebaseData-
base.getInstance().getReference("User");
                                // Create a new User object with the neces-
sary data including role
                                User newUser = new User(user.getUid(),
email.getText().toString(), "User");

                                usersRef.child(user.getUid()).set-
Value(newUser)
                                        .addOnSuccessListener(aVoid -> {
                                            // Handle success
                                            showMessage("Success", "User
data saved successfully!");
                                        })
                                        .addOnFailureListener(e -> {
                                            // Handle failure
                                            showMessage("Error", "Failed to
save user data!");
                                        });
                            }
                        } else {
                            showMessage("Error", task.getException().get-
LocalizedMessage());
                        }
                    }
                });
    } else {
        showMessage("Error", "Please provide all Info!");
    }
}
```

Αρχικά η μέθοδος καλείται όταν πατιέται το κουμπί Sign Up . Πρώτα ελέγχει αν τα πεδία email και password δεν είναι κενα. Αν τα πεδία δεν είναι κενά, τότε καλεί τη μέθοδο createUserWithEmailAndPassword() του αντικειμένου auth

(FirebaseAuth) για να δημιουργήσει ένα νέο λογαριασμό χρήστη χρησιμοποιώντας το email και τον κωδικό που έχουν εισαχθεί από τον χρήστη αποθηκεύοντάς τον στη realtime database του firebase(Κατά την επιτυχή εγγραφή του γίνεται το κατάλληλο authentication του χρήστη και τα δεδομένα του πηγαίνουν στο realtime database). Επισυνάπτει έναν ακροατή (listener) OnCompleteListener στην προηγούμενη ενέργεια, που θα εκτελεστεί όταν η εργασία ολοκληρωθεί (είτε επιτυχώς είτε ανεπιτυχώς). Εάν η εργασία ολοκληρώθηκε επιτυχώς, τότε ανανεώνει τον τρέχοντα χρήστη. Καλεί τη μέθοδο updateUser(user) για να ενημερώσει το προφίλ του χρήστη. Εμφανίζει ένα μήνυμα επιτυχίας της δημιουργίας του χρήστη χρησιμοποιώντας τη μέθοδο showMessage(). Ύστερα αποθηκεύει το FCM token χρησιμοποιώντας τη μέθοδο storeFCMToken(). Εάν η εργασία δεν ολοκληρωθεί με επιτυχία, τότε εμφανίζει ένα μήνυμα λάθους. Εάν τα πεδία email και password είναι κενά, τότε εμφανίζει ένα μήνυμα που ζητά από τον χρήστη να παρέχει όλες τις πληροφορίες εφόσον τουλάχιστον ένα από τα 2 πεδία είναι κενά και πρέπει να συμπληρώσει και τα 2 πεδία για να δημιουργηθεί ο χρήστης επιτυχώς. Εννοείται πως πριν δημιουργηθεί ο χρήστης γίνεται έλεγχος στην βάση δεδομένων για το αν υπάρχει ήδη ο χρήστης με τα ίδια στοιχεία .

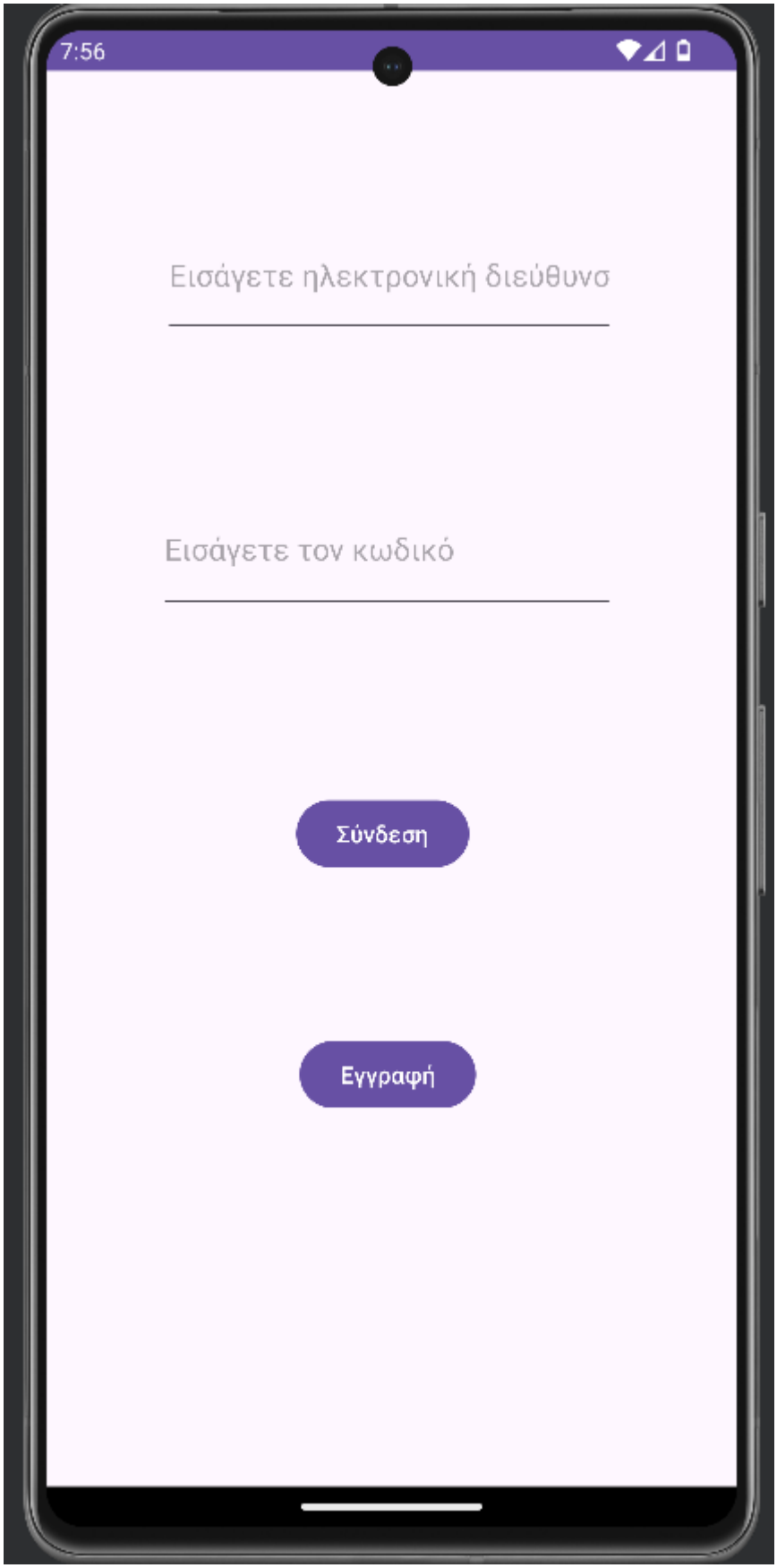
Από την στιγμή που έχει αποθηκευτεί ο χρήστης στην βάση μας τότε μπορούμε να προχωρήσουμε στην διαδικασία της σύνδεσης του χρήστη(η ακόμα και του admin) στην εφαρμογή μας καλώντας την μέθοδο signin μέσω του πατήματος του κουμπιού Sign In.

```
public void signin(View view) {
    String userEmail = email.getText().toString().trim();
    String userPassword = password.getText().toString();

    if (!userEmail.isEmpty() && !userPassword.isEmpty()) {
        auth.signInWithEmailAndPassword(userEmail, userPassword)
                .addOnCompleteListener(new OnCompleteListener<AuthResult>()
{
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task)
{
                        if (task.isSuccessful()) {
                            FirebaseUser user = auth.getCurrentUser();
                            DatabaseReference usersRef = FirebaseData-
base.getInstance().getReference("User");
                            usersRef.child(user.getUid()).addListener-
ForSingleValueEvent(new ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull DataSnap-
shot snapshot) {
                                    if (snapshot.exists()) {
```

```
                                        User userData = snapshot.get-
Value(User.class);
                                        if (userData != null) {
                                            String role =
userData.getRole();
                                            if (role.equals("User")) {
                                                // Go to MainActivity2
                                                Intent intent = new In-
tent(getApplicationContext(), MainActivity2.class);
                                                intent.putExtra("userId",
user.getUid());
                                                startActivity(intent);
                                            } else if (role.equals("Ad-
min")) {
                                                Intent intent = new In-
tent(getApplicationContext(), MainActivity5.class);
                                                intent.putExtra("userId",
user.getUid());
                                                startActivity(intent);
                                            } else {
                                                showMessage("Error", "Un-
known role!");
                                            }
                                        } else {
                                            showMessage("Error", "User data
is null!");
                                        }
                                    } else {
                                        showMessage("Error", "User data not
found!");
                                    }
                                }

                                @Override
                                public void onCancelled(@NonNull Data-
baseError error) {
                                    showMessage("Error", "Database error: "
+ error.getMessage());
                                }
                            });

                            // Store FCM token after user signin
                            storeFCMToken(); // Update the token every time
the user signs in
                        } else {
                            showMessage("Error", "Authentication failed: "
+ task.getException().getMessage());
                        }
                    }
                });
        } else {
            showMessage("Error", "Please provide email and password!");
        }
}
```

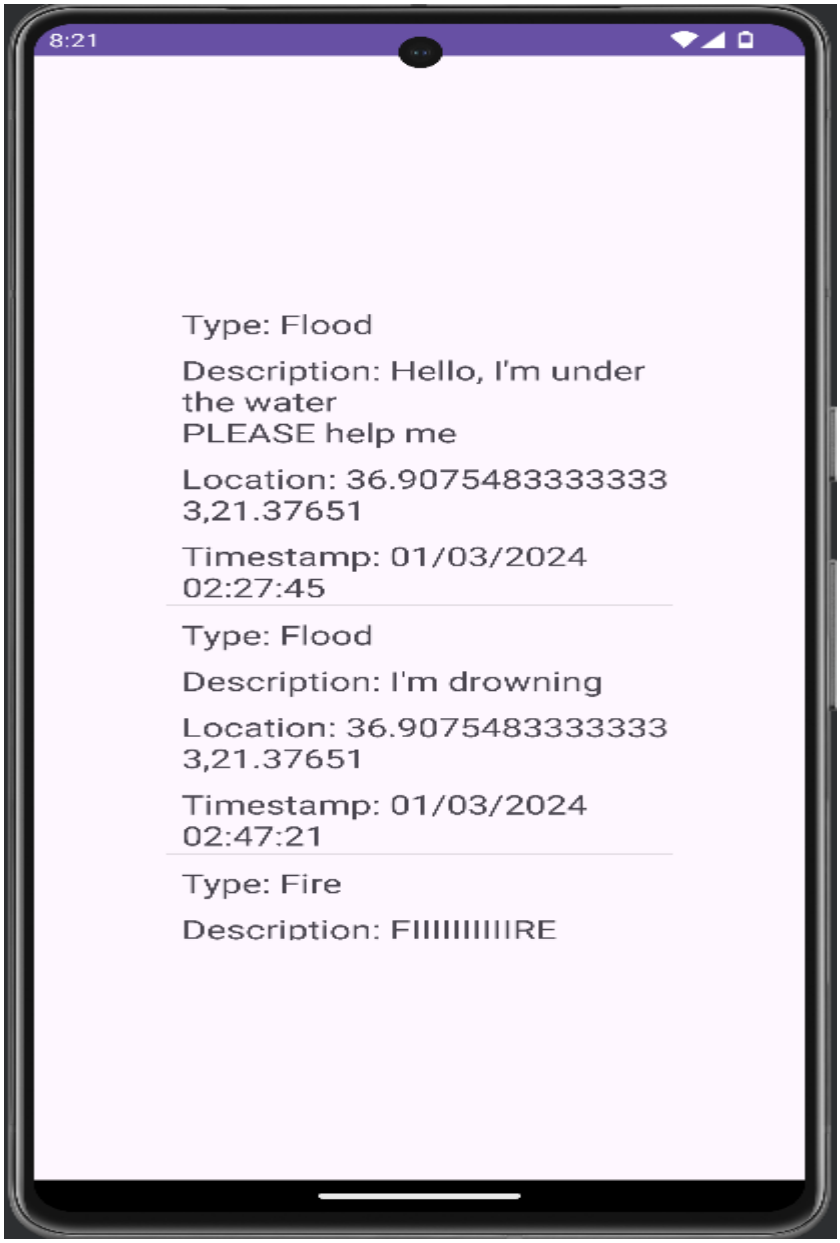
Αρχικά ελέγχεται αν τα πεδία των email και του κωδικού δεν είναι κενά . Αν τα πεδία δεν είναι κενά, τότε καλείται η μέθοδος signInWithEmailAndPassword() του αντικειμένου auth (FirebaseAuth) για να

συνδεθεί ο χρήστης ή ο admin χρησιμοποιώντας το email και τον κωδικό του. Επισυνάπτει έναν ακροατή OnCompleteListener στην προηγούμενη ενέργεια, που θα εκτελεστεί όταν η εργασία ολοκληρωθεί (είτε επιτυχώς είτε ανεπιτυχώς).Μέσα στην onComplete(), ελέγχει εάν η εργασία ολοκληρώθηκε επιτυχώς . Αν η εργασία ολοκληρώθηκε επιτυχώς, τότε ανακτά τον τρέχοντα χρήστη. Ανακτά τα δεδομένα του χρήστη από τη βάση δεδομένων του Firebase χρησιμοποιώντας έναν ακροατή για μια μόνο τιμή (addListenerForSingleValueEvent). Αυτός ο ακροατής θα εκτελεστεί μόνο μία φορά και θα ανακτήσει τα δεδομένα χρήστη από τη βάση δεδομένων. Μέσα στην onDataChange(), ελέγχει εάν υπάρχουν δεδομένα για τον χρήστη στη βάση δεδομένων. Αν υπάρχουν δεδομένα, ανακτά τα δεδομένα του χρήστη. Εάν τα δεδομένα είναι διαθέσιμα, ανακτά τον ρόλο του χρήστη και ανακατευθύνει τον χρήστη στην κατάλληλη δραστηριότητα (activity) ανάλογα με τον ρόλο του. Εάν τα δεδομένα δεν είναι διαθέσιμα, εμφανίζει ένα μήνυμα λάθους. Αν τα πεδία email και password είναι κενά, τότε εμφανίζει ένα μήνυμα που ζητά από τον χρήστη να παρέχει το email και τον κωδικό του.



Στη περίπτωση που συνδέεται ως χρήστης της εφαρμογής , ανακατευθύνεται στο MainActivity2 όπου έχει την δυνατότητα είτε να δει όλα τα μηνύματα που

έχει στείλει για ένα επικίνδυνο φαινόμενο και έχουν εγκριθεί ως επικίνδυνα από τον admin αλλά και να στείλει καινούργιο μήνυμα για να στείλει το καινούργιο emergency. Πατώντας το κουμπί Read last urgent messages καλείται η μέθοδος submit και ανακατευθύνεται στη πλατφόρμα των μηνυμάτων. Πατώντας το κουμπί submit an emergency incident ανακατευθύνεται στη πλατφόρμα για να δημιουργήσει το νέο emergency . Και στα 2 activities το πρόγραμμα παίρνει ως έξτρα στοιχείο το userId για να έχουμε αποθηκευμένο το id του συγκεκριμένου χρήστη που είναι συνδεδεμένος στην εφαρμογή.



Μπαίνοντας στην πλατφόρμα της περιγραφής νέου φαινομένου , ο χρήστης καλείται να συμπληρώσει όλα τα απαραίτητα πεδία για να τα στείλει στον admin να ελέγξει την σοβαρότητα αυτού του φαινομένου. Μέσα σε αυτά τα πεδία υπάρχουν και 2 κουμπιά τα οποία έχουν να κάνουν με την τοποθεσία στην οποία βρίσκεται ο χρήστης την εκάστοτε στιγμή και για την λήψη φωτογραφίας για την απεικόνιση του φαινομένου που συμβαίνει στην περιοχή του.

```
public void FindLocation(android.view.View view) {
    if (ActivityCompat.checkSelfPermission(this, android.Manifest.permis-
sion.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.per-
mission.ACCESS_FINE_LOCATION}, 123);
```

```
        return;
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
0, this);
}

@Override
public void onLocationChanged(Location location) {
    latitude = String.valueOf(location.getLatitude());
    longitude = String.valueOf(location.getLongitude());
    Userlocation = latitude + "\n" + longitude + "\n";
    myTextView.setText(Userlocation);
}
public void getLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.AC-
CESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.per-
mission.ACCESS_FINE_LOCATION}, 123);
        return;
    }

    if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
            !locationManager.isProviderEnabled(LocationManager.NETWORK_PRO-
VIDER)) {
        // Build the alert dialog
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Location Services Not Enabled");
        builder.setMessage("Please enable Location Services");
        builder.setPositiveButton("OK", new DialogInterface.OnClick-
Listener() {
            public void onClick(DialogInterface dialogInterface, int i) {
                Intent intent = new Intent(Settings.ACTION_LOCA-
TION_SOURCE_SETTINGS);
                startActivity(intent);
            }
        });
        Dialog alertDialog = builder.create();
        alertDialog.setCanceledOnTouchOutside(true);
        alertDialog.show();
    } else {
        LocationListener locationListener = new LocationListener() {
            @Override
            public void onLocationChanged(@NonNull Location location) {
                Log.d("Location", "Location changed: " + location.getLati-
tude() + ", " + location.getLongitude());
                // Save the location to a global variable or use it di-
rectly where needed
                MainActivity3.this.location = location.getLatitude() + ","
+ location.getLongitude();
                // Once you get the location, you can proceed with further
actions, such as uploading data to Firebase
            }
        };
        locationManager.requestLocationUpdates(LocationManager.GPS_PRO-
VIDER, 0, 0, locationListener);
        Location loc = locationManager.getLastKnownLocation(LocationMan-
ager.GPS_PROVIDER);

        if (loc != null) {
            location = loc.getLatitude() + "," + loc.getLongitude();
            Log.d("Location", "Last known location: " + location);
```

```
            // If you need to use the last known location immediately, you
can handle it here
        }
    }
}
```

Όσον αφορά την ανάκτηση της τοποθεσίας , πατώντας το κουμπί Find Your Location καλούνται 3 μεθόδοι (FindLocation , getLocation , onLocationChanged). Η μέθοδος FindLocation() καλείται όταν ο χρήστης πατάει το κουμπί για να βρει την τοποθεσία του. Ελέγχει αν η εφαρμογή έχει την άδεια για πρόσβαση στην τοποθεσία. Αν δεν έχει την άδεια, ζητείται η άδεια από τον χρήστη. Αν υπάρχει ήδη άδεια, ο LocationManager ζητάει ενημερώσεις της τοποθεσίας από τον πάροχο GPS.

Η μέθοδος onLocationChanged() υλοποιεί τη διεπαφή LocationListener. Καλείται κάθε φορά που αλλάζει η τοποθεσία. Εδώ, η τοποθεσία του χρήστη εξάγεται (για παράδειγμα, γεωγραφικό πλάτος και μήκος) και εμφανίζεται σε ένα TextView.

Η μέθοδος getLocation() χρησιμοποιείται για να πάρει την τρέχουσα τοποθεσία του χρήστη. Αυτό είναι ένας συνδυασμός του FindLocation() και του onLocationChanged(). Εάν δεν υπάρχει άδεια για πρόσβαση στην τοποθεσία, ζητείται από τον χρήστη. Αν οι πάροχοι τοποθεσίας (GPS ή δίκτυο) δεν είναι ενεργοποιημένοι, εμφανίζεται ένας διάλογος για την ενεργοποίησή τους. Αν όλα είναι εντάξει, ο LocationManager ζητά ενημερώσεις της τοποθεσίας από τον πάροχο GPS και αναλαμβάνει την ενημέρωση της τοποθεσίας όταν αυτή αλλάζει. Επίσης, προσπαθεί να πάρει την τελευταία γνωστή τοποθεσία από τον πάροχο GPS. Ταυτόχρονα, η τελευταία τοποθεσία εμφανίζεται σε ένα Log και μπορεί να χρησιμοποιηθεί για περαιτέρω επεξεργασία, όπως η αποστολή δεδομένων στο Firebase.

```
private void openImagePicker() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    resultLauncher.launch(Intent.createChooser(intent, "Select Picture"));
}

// Upload the image to Firebase Storage
private void uploadImage(Uri imageUri) {
    if (imageUri != null) {
        String imageName = UUID.randomUUID().toString();
        StorageReference imageRef = storageReference.child("images/" +
imageName);

        imageRef.putFile(imageUri)
```

```
                .addOnSuccessListener(taskSnapshot -> imageRef.getDown-
loadUrl()
                        .addOnSuccessListener(uri -> saveEmergency-
Data(uri.toString()))
                        .addOnFailureListener(e -> showMessage("Error",
"Failed to get image URL: " + e.getMessage())))
                .addOnFailureListener(e -> showMessage("Error", "Failed to
upload image: " + e.getMessage()));
    } else {
        showMessage("Error", "No image selected");
    }
}
```

Όσον αφορά την λήψη της φωτογραφίας καλείται η μέθοδος UploadImage και OpenImagePicker των οποίων η λειτουργία είναι η εξής : Στο πάτημα του κουμπιού Select Image γίνεται το κάλεσμα της μεθόδου OpenImagePicker και χρησιμοποιείται για να ανοίξει το παράθυρο επιλογής εικόνας του συστήματος. Δημιουργεί ένα αίτημα Intent με δράση ACTION_GET_CONTENT, προκειμένου να πάρει περιεχόμενο (σε αυτή την περίπτωση εικόνες), και ορίζει τον τύπο του περιεχομένου ως "image/*". Στη συνέχεια, εκκινεί τη δραστηριότητα με το αίτημα αυτό, χρησιμοποιώντας τη μέθοδο launch() του ActivityResultLauncher. Μετά την επιλογή της εικόνας που θέλει ο χρήστης , γίνεται η αποθήκευσή της στον αποθηκευτικό χώρο της firebase. Συγκεκριμένα η μέθοδος uploadImage(Uri imageUri) χρησιμοποιείται για να ανεβάσει την εικόνα που επιλέχθηκε στο Firebase Storage. Αρχικά, ελέγχει αν η imageUri περιέχει μια έγκυρη τοποθεσία εικόνας. Αν ναι, δημιουργεί ένα μοναδικό όνομα για την εικόνα χρησιμοποιώντας τη UUID.randomUUID(). Στη συνέχεια, δημιουργεί μια αναφορά στο Firebase Storage για να αποθηκεύσει την εικόνα στη διαδρομή "images/". Χρησιμοποιώντας τη μέθοδο putFile(imageUri), ανεβάζει την εικόνα στο Firebase Storage. Εάν η αποθήκευση είναι επιτυχής, παίρνει τη διεύθυνση URL της αποθηκευμένης εικόνας και καλεί τη μέθοδο saveEmergencyData() για να αποθηκεύσει τα δεδομένα της έκτακτης ανάγκης στη βάση δεδομένων. Αν αποτύχει η αποθήκευση του URL της εικόνας, εμφανίζει ένα μήνυμα σφάλματος. Εάν η αποθήκευση της εικόνας αποτύχει, εμφανίζει ένα μήνυμα σφάλματος που αναφέρει τον λόγο της αποτυχίας. Μέσα στην ίδια μέθοδο καλείται και η saveEmergencyData.

```
private void saveEmergencyData(String imageUrl) {
    String stringdesc = description.getText().toString();
    DatabaseReference dbEmergency = FirebaseDatabase.get-
Instance().getReference("Emergencies");
    String timestamp = DateTimeFormatter.ofPattern("dd/MM/yyyy
HH:mm:ss").format(LocalDateTime.now());
    DatabaseReference reference = database.getReference("Emergencies");
    String emergencyId = reference.push().getKey();
```

```
    Emergency emergency = new Emergency(emergencyId,stringdesc, TypeOfEmer-
gency, latitude, longitude, location, timestamp, userId, imageUrl);
    reference.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (DataSnapshot alertSnapshot :
task.getResult().getChildren()) {
                String eventType = alertSnapshot.child("emergency").get-
Value(String.class);
                if (eventType != null && eventType.equals(emer-
gency.getEmergency())) {
                    hours = 0;
                    kilometers = 0;
                    switch (emergency.getEmergency()) {
                        case "Earthquake":
                            hours = 2;
                            kilometers = 150;
                            break;
                        case "Flood":
                            hours = 12;
                            kilometers = 100;
                            break;
                        case "Hurricane":
                            hours = 24;
                            kilometers = 80;
                            break;
                        case "Fire":
                            hours = 48;
                            kilometers = 200;
                            break;
                        case "Storm":
                            hours = 5;
                            kilometers = 50;
                            break;
                    }
                    if (isWithinHours(alertSnapshot.child("timestamp").get-
Value(String.class), emergency.getTimestamp(), hours) &&
                            isWithinKilometers(alertSnapshot.child("loca-
tion").getValue(String.class), emergency.getLocation(), kilometers)) {
                        emergency.setCount(emergency.getCount() + 1);
                        reference.child(alertSnapshot.get-
Key()).child("count").setValue(alertSnapshot.child("count").getValue(Inte-
ger.class) + 1);
                    }
                }
            }
            dbEmergency.child(emergencyId).setValue(emergency)
                    .addOnSuccessListener(aVoid -> {
                        Toast.makeText(MainActivity3.this, "Emergency data
saved successfully!", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(MainActivity3.this,
MainActivity2.class);
                        startActivity(intent);
                    })
                    .addOnFailureListener(e -> Toast.makeText(MainActiv-
ity3.this, "Failed to save emergency data!", Toast.LENGTH_SHORT).show());



        } else {
            showMessage("Error", "Failed to retrieve data: " + task.getEx-
ception().getMessage());
```

```
            }
        });

}
```

Η μέθοδος αρχικά παίρνει την περιγραφή της έκτακτης ανάγκης από το πεδίο κειμένου description και την αποθηκεύει σε μια μεταβλητή με το όνομα stringdesc. Δημιουργεί μια αναφορά στη βάση δεδομένων Firebase με το όνομα "Emergencies" χρησιμοποιώντας τη μέθοδο FirebaseDatabase.getInstance().getReference("Emergencies") και την αποθηκεύει στη μεταβλητή dbEmergency. Φτιάχνει ένα μορφοποιημένο χρονικό σήμα (timestamp) χρησιμοποιώντας την κλάση DateTimeFormatter για την τρέχουσα ημερομηνία και ώρα. Δημιουργεί μια αναφορά στη βάση δεδομένων Firebase με το όνομα "Emergencies" χρησιμοποιώντας τη μέθοδο database.getReference("Emergencies") και την αποθηκεύει στη μεταβλητή reference.

Δημιουργεί ένα μοναδικό αναγνωριστικό για την έκτακτη ανάγκη χρησιμοποιώντας τη μέθοδο push().getKey() και το αποθηκεύει στη μεταβλητή emergencyId. Δημιουργεί ένα νέο αντικείμενο Emergency με τα δεδομένα της έκτακτης ανάγκης που πρέπει να αποθηκευτούν. Τα δεδομένα αυτά περιλαμβάνουν το αναγνωριστικό της έκτακτης ανάγκης, την περιγραφή, τον τύπο, τις συντεταγμένες, την τοποθεσία, το χρονικό σήμα, τον χρήστη και τη διεύθυνση URL της εικόνας. Κάνει μια αναζήτηση στη βάση δεδομένων Firebase για να πάρει τις προηγούμενες έκτακτες ανάγκες μέσω της μεθόδου getReference().get(). Σε περίπτωση που η αναζήτηση είναι επιτυχής, πραγματοποιεί μια λούπα μέσω των προηγούμενων έκτακτων αναγκών και ελέγχει αν υπάρχει προηγούμενη έκτακτη ανάγκη με τον ίδιο τύπο με την τρέχουσα. Εάν υπάρχει προηγούμενη έκτακτη ανάγκη με τον ίδιο τύπο, ελέγχει αν η προηγούμενη έκτακτη ανάγκη είναι εντός ενός συγκεκριμένου χρονικού διαστήματος και απόστασης από την τρέχουσα. Αν ισχύουν τα παραπάνω, αυξάνει τον αριθμό εμφανίσεων της προηγούμενης έκτακτης ανάγκης και ενημερώνει τη βάση δεδομένων Firebase. Τέλος, αποθηκεύει τα δεδομένα της τρέχουσας έκτακτης ανάγκης στη βάση δεδομένων Firebase και εμφανίζει αντίστοιχα μηνύματα επιτυχίας ή αποτυχίας στην οθόνη.

Συνοψίζοντας, η μέθοδος αυτή χρησιμοποιείται για να αποθηκεύσει τα δεδομένα μιας νέας έκτακτης ανάγκης στη βάση δεδομένων Firebase και να ελέγξει εάν υπάρχουν παρόμοιες προηγούμενες ανάγκες.

Για να γίνει η αποθήκευση των στοιχείων τύπου emergency στο realtime database γίνεται με το πάτημα του κουμπιού Submit και καλείται η μέθοδος

SubmitForm. Γίνεται απαραίτητος έλεγχος αν έχουν συμπληρωθεί όλα τα στοιχεία της φόρμας για να γίνει η αποθήκευση τους. Σε περιπτωσή που λείπει έστω και ένα πεδίο εμφανίζαται κατάλληλο μήνυμα ανάλογα με το ποι πεδίο δεν έχει συμπληρωθεί. Ειδάλλως καλείται η UploadImage που εξηγήσαμε παραπάνω.

```java
package com.example.smartalert;

import android.os.Bundle;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DividerItemDecoration;
```

```
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class MainActivity4 extends AppCompatActivity {

    RecyclerView recyclerView;
    EmergencyAdapter emergencyAdapter;
    List<Emergency> emergencyList;
    String userId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);
        userId = getIntent().getStringExtra("userId");

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        // Adding item decoration to display dividers between items
        DividerItemDecoration dividerItemDecoration = new DividerItemDeco-
ration(recyclerView.getContext(), DividerItemDecoration.VERTICAL);
        recyclerView.addItemDecoration(dividerItemDecoration);

        emergencyList = new ArrayList<>();
        emergencyAdapter = new EmergencyAdapter(emergencyList);
        recyclerView.setAdapter(emergencyAdapter);

        // Get reference to the "Emergencies" node in the database
        DatabaseReference dbEmergency = FirebaseDatabase.get-
Instance().getReference("Emergencies");

        dbEmergency.addListenerForSingleValueEvent(new ValueEventListener()
{
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    // Retrieve emergency data from the snapshot
                    Emergency emergency = snapshot.getValue(Emer-
gency.class);
                    if (emergency != null && emergency.getUse-
rID().equals(userId)) {
                        // Add emergency to the list
                        emergencyList.add(emergency);
                    }
                }
                // Notify the adapter that data set has changed
                emergencyAdapter.notifyDataSetChanged();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
```

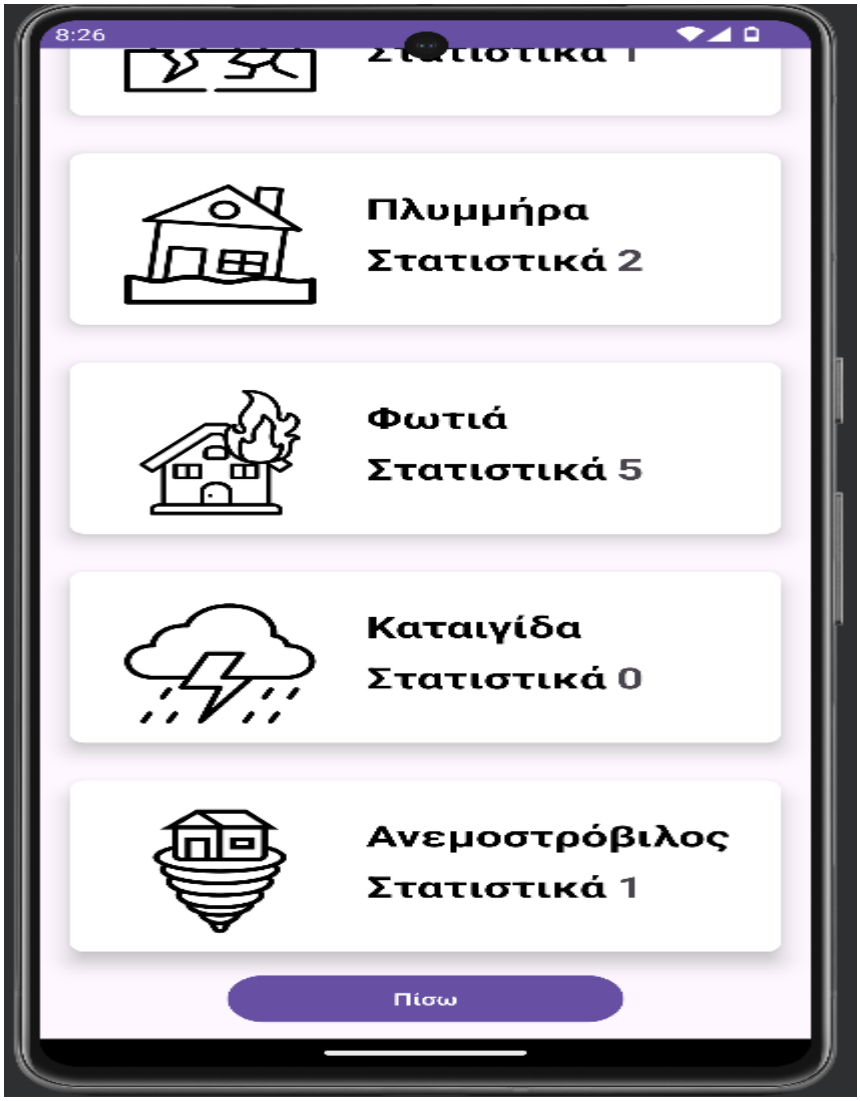
```
                // Handle database error
            }
        });
    }

}
```

Στο MainActivity4 ο χρήστης βλέπει τα emergencies που έχει αποδεχτεί ο admin εφόσον τα έγκρινε ως σοβαρά περιστατικά. Τα emergencies του εμφανίζονται με βάση το userId του . Στην μέθοδο onCreate() αρχικοποιεί το layout της δραστηριότητας και τις μεταβλητές που θα χρησιμοποιηθούν, όπως η λίστα emergencyList, ο διακομιστής recyclerView, και ο προσαρμογέας emergencyAdapter. Ορίζεται ο διαχειριστής της λίστας με γραμμές και προστίθενται διαχωριστικές γραμμές μεταξύ των αντικειμένων της λίστας. Δημιουργείται μια νέα λίστα emergencyList για την αποθήκευση των εκτάκτων αναγκών. Δημιουργείται ένας νέος προσαρμογέας EmergencyAdapter και ορίζεται ως προσαρμογέας του recyclerView. Στη συνέχεια, γίνεται αναφορά στον κόμβο "Emergencies" στη βάση δεδομένων Firebase. Μετά ορίζεται ένας ValueEventListener για να ακούει για αλλαγές στα δεδομένα του κόμβου "Emergencies". Μέσα στη μέθοδο onDataChange, ο κώδικας περνάει από κάθε παιδί του κόμβου "Emergencies" και αποκτά τα δεδομένα των εκτάκτων αναγκών που αποθηκεύονται στο Firebase. Αν το αντικείμενο έκτακτης ανάγκης ανήκει στον τρέχοντα χρήστη (με βάση το ID του χρήστη), τότε προστίθεται στη λίστα emergencyList. Τέλος, ο προσαρμογέας ειδοποιείται για τις αλλαγές στα δεδομένα με τη μέθοδο notifyDataSetChanged(), ώστε να ανανεωθεί η εμφάνιση του recyclerView με τα νέα δεδομένα. Αυτός ο κώδικας επιτρέπει στην εφαρμογή να ανακτήσει και να εμφανίσει όλες τις έκτακτες ανάγκες που ανήκουν στον συγκεκριμένο χρήστη στο recyclerView.

Τώρα προχωράμε στην λειτουργικότητα του admin στην εφαρμογή μας. Όταν γίνεται σύνδεση ως admin , όπως και στους users , έτσι και ο ίδιος έχει τις εξής δυνατοτητες: Να διαβάσει τα στατιστικά των περιστατικών και να αποδεχτεί ή να απορρίψει τις ειδοποιήσεις των χρηστών για περιστατικά υψηλού κινδύνου.

Με το πάτημα του κουμπιού Read Statistics ανακατευθυνόμαστε στη πλατφόρμα των στατιστικών της οποίας μπορεί να δει τα στατιστικά του κάθε επικίνδυνου περιστατικού που συμβαίνει


8:26
Πλυμμήρα
Στατιστικά 2
Φωτιά
Στατιστικά 5
Καταιγίδα
Στατιστικά 0
Ανεμοστρόβιλος
Στατιστικά 1
Πίσω



Με το πάτημα του κουμπιού Accept or Decline New Message πηγαίνει στην πλατφόρμα του για να διαχειριστεί τις ειδοποιήσεις από τους users του.

```
private void GatherData() {
    DatabaseReference dbEmergency = FirebaseDatabase.get-
Instance().getReference("Emergencies");
    Geocoder geocoder = new Geocoder(this, Locale.getDefault());
    dbEmergency.orderByChild("count").limitToLast(1).addListener-
ForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) {
                boolean pendingEmergencyFound = false;
                for (DataSnapshot alertSnapshot : dataSnap-
shot.getChildren()) {
                    // Retrieve emergency data from the snapshot
                    emergency = alertSnapshot.getValue(Emergency.class);
                    if (emergency != null && emergency.getSta-
tus().equals("pending")) {
                        pendingEmergencyFound = true;
                        // Add emergency to the list
                        date.setText(emergency.getTimestamp());
                        danger.setText(emergency.getEmergency());
```

```
                        location.setText(emergency.getLocation());
                        String loc = alertSnapshot.child("location").get-
Value(String.class);
                        try {
                            String city = geocoder.getFromLoca-
tion(parseDouble(loc.substring(0,loc.indexOf(","))),parseDouble(loc.sub-
string(loc.indexOf(",")+1,loc.length())),1).get(0).getAddressLine(0);
                            location.setText(city);
                            System.out.println(city);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        }
                        description.setText(emergency.getDescription());
                        counter.setText("Total Reports:"+emer-
gency.getCount());
                        Picasso.get().load(emer-
gency.getImageUrl()).into(imageView);
                        emergencyId=alertSnapshot.getKey();
                    }
                }
                if (!pendingEmergencyFound) {
                    showMessage("Warning", "No Pending Emergencies
Found!");

                    // Wait for 3 seconds before redirecting to MainActiv-
ity2
                    new android.os.Handler().postDelayed(() -> {
                        Intent intent = new Intent(MainActivity7.this,
MainActivity5.class);
                        startActivity(intent);
                        finish(); // Optional, if you want to close the
current activity
                    }, 3000); // 3000 milliseconds delay (adjust as needed)
                }
            } else {

            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            // Handle errors
        }
    });
}
```

Κατά εκκίνηση του MainActivity7 αρχικά καλείται η μέθοδος GatherData η οποία έχει την εξής λειτουργικότητα. Δημιουργεί μια αναφορά στη βάση δεδομένων Firebase με τον τίτλο "Emergencies". Χρησιμοποιεί τη μέθοδο orderByChild("count").limitToLast(1) για να επιστρέψει την τελευταία έκτακτη ανάγκη με βάση τον αριθμό των αναφορών. Προσθέτει έναν ValueEventListener στην αναφορά που ακούει για αλλαγές στα δεδομένα. Στη μέθοδο onDataChange(), ελέγχει αν υπάρχουν δεδομένα στο στιγμιότυπο του DataSnapshot. Αν υπάρχουν, επεξεργάζεται το πρώτο στιγμιότυπο που επιστρέφεται (το οποίο είναι η τελευταία έκτακτη ανάγκη με βάση τον αριθμό

των αναφορών). Αν η έκτακτη ανάγκη είναι σε κατάσταση "pending", τότε ενημερώνει το UI με τα σχετικά δεδομένα της ανάγκης, όπως η ημερομηνία, ο τύπος, η τοποθεσία, η περιγραφή, ο αριθμός των αναφορών και η εικόνα. Αν δεν υπάρχει καμία εκκρεμής έκτακτη ανάγκη, τότε εμφανίζει ένα προειδοποιητικό μήνυμα και μεταφέρει τον χρήστη σε μια άλλη δραστηριότητα μετά από μια σύντομη καθυστέρηση. Συνολικά, αυτή η μέθοδος εκτελεί τη διαδικασία ενημέρωσης του UI με την τελευταία έκτακτη ανάγκη από τη βάση δεδομένων Firebase. Όσο υπάρχουν αναφορές ο Υπάλληλος πολιτικής προστασίας έχει 2 επιλογές . Να αποδεχτεί ή να απορρίψει την αναφορά του χρήστη που την έστειλε. Για κάθε ενέργεια καλείται και η αντίστοιχη μέθοδος(onAccept για αποδοχή και onDecline για απόρριψη).

Ας δούμε κάθε μέθοδο ξεχωριστά:

onAccept()

Αυτή η μέθοδος εκτελείται όταν ο χρήστης αποδέχεται μια έκτακτη ανάγκη. Εκτελείται η παρακάτω λογική: Δημιουργεί αναφορές στη βάση δεδομένων Firebase για τις κατηγορίες "emergencies", "accepted" και "sent_alerts". Λαμβάνει τα δεδομένα όλων των έκτακτων αναγκών από τη βάση δεδομένων. Για κάθε έκτακτη ανάγκη, ελέγχει αν η ανάγκη είναι εντός των ορίων χρόνου και απόστασης και αν η κατάστασή της είναι "pending". Αν ναι, μεταφέρει την ανάγκη στην κατηγορία "accepted" και την αφαιρεί από την κατηγορία "emergencies". Στη συνέχεια, ενημερώνει την κατηγορία "sent_alerts" με τον αριθμό των αναγκών που έχουν αποδεχτεί για κάθε τύπο έκτακτης ανάγκης. Εκτελεί τη μέθοδο sendNotification() για να στείλει ειδοποιήσεις σε χρήστες που σχετίζονται με την έκτακτη ανάγκη. Επιστρέφει τα δεδομένα ανάγκης μέσω της μεθόδου GatherData().

onDecline()

Αυτή η μέθοδος εκτελείται όταν ο χρήστης απορρίπτει μια έκτακτη ανάγκη. Εκτελείται η παρακάτω λογική: Δημιουργεί αναφορές στη βάση δεδομένων Firebase για τις κατηγορίες "alerts" και "rejected". Λαμβάνει τα δεδομένα όλων των έκτακτων αναγκών από τη βάση δεδομένων. Για κάθε έκτακτη ανάγκη, ελέγχει αν η ανάγκη είναι εντός των ορίων χρόνου και απόστασης και αν η κατάστασή της είναι "pending". Αν ναι, μεταφέρει την ανάγκη στην κατηγορία "rejected" και μειώνει τον αριθμό των αναφορών κατά 1 στην κατηγορία "alerts". Επιστρέφει τα δεδομένα ανάγκης μέσω της μεθόδου GatherData(). Συνολικά, οι δύο μέθοδοι εκτελούν λογική για την αποδοχή ή την απόρριψη μιας έκτακτης ανάγκης, εκτελώντας αντίστοιχες ενέργειες στη βάση δεδομένων Firebase και επικοινωνώντας με τους χρήστες μέσω ειδοποιήσεων.