ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας	**(Ελληνικά)** Android εφαρμογή διαχείρισης υγείας κατοικίδιων και επικοινωνίας με κτηνίατρους **(Αγγλικά)** Android app for managing pet health and communicating with veterinarians
Ονοματεπώνυμο Φοιτητή	**Κατσικονούρης Βασίλειος**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**Π19066**
Επιβλέπων	**Ευθύμιος Αλέπης, Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2024**

Copyright ©

Πίνακας περιεχομένων

Περίληψη (Abstract)

English

This thesis focuses on the development of an Android application, which functions as an electronic health booklet for pets, using Android Studio and Java code, with Firebase as the database.

The application provides registered users the ability to add their pets by filling in the necessary information. Based on the user's postal code, it connects them with nearby veterinarians to facilitate collaboration. Additionally, users can communicate with veterinarians and other users via the chat feature.

The purpose of the application is to offer a comprehensive and user-friendly platform that helps pet owners find a veterinarian in their area to take care of their pets' health. At the same time, it promotes the creation of relationships among users for the exchange of knowledge and information, as well as to assist in finding lost pets.

Ελληνικά

Η παρούσα πτυχιακή επικεντρώνεται στην υλοποίηση μιας Android εφαρμογής , η οποία λειτουργεί ως ηλεκτρονικό βιβλιάριο υγείας για κατοικίδια ,με την χρήση του εργαλείου Android studio σε java κώδικα έχοντας για βάση δεδομένων την Firebase .

Η εφαρμογή παρέχει στους εγγεγραμμένους χρήστες της, τη δυνατότητα να εισάγουν τα κατοικίδια τους , συμπληρώνοντας τις απαραίτητες πληροφορίες για αυτά ,όπου με βάση τον ταχυδρομικό κώδικα του εκάστοτε χρήστη έρχεται σε επαφή με κοντινούς κτηνίατρους με σκοπό τη διευκόλυνση της συνεργασίας μεταξύ τους. Επιπλέον , οι χρήστες μπορούν να έρθουν σε επικοινωνία με κτηνίατρους αλλά και με άλλους χρήστες μέσω του chat

Ο σκοπός της εφαρμογής είναι να προσφέρει μια ολοκληρωμένη και εύχρηστη πλατφόρμα η οποία θα βοηθάει τους ιδιοκτήτες κατοικίδιων να βρουν κτηνίατρο στην περιοχή τους ο οποίος θα αναλάβει την υγεία των ζώων τους. Παράλληλα προωθεί την δημιουργία σχέσεων μεταξύ πολλών χρηστών με σκοπό την ανταλλαγή γνώσεων και πληροφοριών αλλά και την βοήθεια στο να βρεθεί κάποιο χαμένο κατοικίδιο

Εισαγωγή

Στην σημερινή εποχή , όπου το μεγαλύτερο ποσοστό των ανθρώπων έχουν ένα ή περισσότερα κατοικίδια , υπάρχει η ανάγκη για μια γρήγορη και εύχρηστη εφαρμογή η οποία θα εξοικονομεί πολύ χρόνο , βοηθώντας μας στην εύκολη και

άμεση εύρεση ενός κτηνίατρου αλλά και στην ταχύτερη αναζήτηση ενός εξαφανισμένου κατοικίδιου ,προσφέροντας αποτελεσματικές λύσεις .

Η παρούσα εφαρμογή στοχεύει να καλύψει τις ανάγκες παρέχοντας μια ολοκληρωμένη πλατφόρμα για τους χρήστες. Η εφαρμογή επιτρέπει στους χρήστες να εισάγουν τα κατοικίδια τους σε μορφή μιας ηλεκτρονικής ταυτότητας με σκοπό την εύρεση ενός κτηνίατρου σε κοντινή απόσταση αλλά και την πλήρη συνεργασία με αυτόν. Επίσης παρέχει στους χρήστες τη δυνατότητα επικοινωνίας μέσου του chat , τόσο με άλλους ιδιοκτήτες όσο και με κτηνιάτρους ανταλλάσσοντας χρήσιμες πληροφορίες και συμβουλές . Ακόμα οι χρήστες μπορούν να στείλουν ένα μήνυμα κινδύνου σε περίπτωση εξαφάνισης κατοικίδιου μέσα στο chat , παρέχοντας φωτογραφία και περιγραφή του ζώου τα οποία καθιστούν την αναγνώριση και την εύρεση πολύ ποιο εύκολη .

Η εφαρμογή έχει κατασκευαστεί χρησιμοποιώντας ένα συνδυασμό τεχνολογιών . Στην πλευρά του Back -End χρησιμοποιήθηκε Java σε συνδυασμό με το εργαλείο Android Studio του οποίου τα Components βοήθησαν στην υλοποίηση του Front End . Ως βάση δεδομένων έχουμε την Firebase καθώς παρέχει ένα εύχρηστο Ui και συγχρονισμό δεδομένων σε πραγματικό χρόνο . Επίσης η Firebase παρέχει ασφάλεια μέσω του ελέγχου ταυτότητας χρηστών (Firebase Authentication) και δυνατότητα αποθήκευσης και διαχείρισης μεγάλων αρχείων όπως εικόνες και βίντεο μέσο το Firebase Storage .

Ιδιαίτερη έμφαση δίνεται στην εμπειρία χρήστη και την προσβασιμότητα ,διασφαλίζοντας ότι κάθε χρήστης δεν θα δυσκολευτεί να χρησιμοποιήσει την εφαρμογή. Το pawprintcare παρέχει σε κάθε φόρμα ένα button βοήθειας το οποίο εξηγεί με αρκετά σαφή τρόπο το τη ακριβώς κάνει η εκάστοτε σελίδα αλλά και το πως μπορείς κάποιος να προηγηθεί μέσα σε αυτήν . Ακόμα η εφαρμογή υποστηρίζει δύο γλώσσες Αγγλικά – Ελληνικά έτσι ώστε να μπορεί να χρησιμοποιηθεί και από άτομα τα οποία δεν έχουν αρκετή εξοικείωση με τα Αγγλικά .

Επισκόπηση Αρχιτεκτονικής

Το Ui της εφαρμογής περιλαμβάνει Activities και Fragments με τα οποία οι χρήστες αλληλοεπιδρούν με αυτήν . Αυτά τα 2 στοιχεία είναι σχεδιασμένα σε XML αρχεία όπου τα JAVA Classes είναι υπεύθυνα για την λειτουργία τους.

Το Firebase SDK Integration ενσωματώνεται μέσω του Gradle στο build αρχείο έτσι ώστε να πραγματοποιηθεί η πρόσβαση σε υπηρεσίες της βάσης , όπως Firebase Realtime Database για την αποθήκευση δεδομένων , Firebase Authentication για τον έλεγχο ταυτότητας αλλά και Firebase Storage για την αποθήκευση εικόνων

Μέσα στην εφαρμογή υπάρχει μια κλάση FirebaseHelper η οποία είναι υπεύθυνη για την διαχείριση τον δεδομένων που αποστέλλονται και λαμβάνονται από την firebase . Η κλάση αυτή περιλαμβάνει τις εξής μεθόδους:

- retiveImgeUrlFromFistore () ❼ Η παίρνει μια εικόνα από το Firebase Storage και την προβάλει σε ένα συγκεκριμένο imageview
- updateOwnerField () ❼ Η οποία κάνει Update το collection Appoints
- location () ❼ Η οποία επιστρέφει το Location του χρήστη
- addAppoint () ❼ Η οποία προσθέτει ένα νέο appoint στην βάση
- acceptRequest ()❼ Η οποία χρησιμοποιείται όταν ο γιατρός αποδέχεται ένα αίτημα για συνεργασία
- findDetails () ❼ Η οποία επιστρέφει πληροφορίες για τους χρήστες
- findLocation ()❼ Με την οποία βρίσκουμε την τοποθεσία του χρήστη
- findMyVet () ❼ Η οποία βρίσκει τον κτηνίατρο του χρήστη, στην περίπτωση που συνεργάζεται με κάποιον
- findMyPets () ❼ Βρίσκει και επιστρέφει τα κατοικίδια του χρήστη
- setVaccineAppoint () ❼ Με την οποία ένας κτηνίατρος προτείνει ένα εμβόλιο στον πελάτη του.
- setVaccineValue () ❼ Η οποία χρησιμοποιείται από έναν ιδιοκτήτη ζώου και αλλάζει την τιμή vaccineIsOff σε true εάν δέχεται το ραντεβού ή σε false εάν όχι
- deletePet () ❼ Η οποία διαγράφει ένα κατοικίδιο από την βάση
- haveApoint () ❼ Με την οποία γίνεται έλεγχος εάν κάποιος χρήστης έχει στείλει έτοιμα σε έναν κτηνίατρο

(Ορισμένες λειτουργίες για την βάση έχουν γίνει από τάξεις του εκάστοτε activity για να κρατήσουμε όσο γίνεται μικρότερο το μέγεθος του κώδικα)

Στήσιμο της Βάσης Δεδομένων

Αρχικά ,προκειμένου να συνδέσουμε την Firebase με το project μας στο Android studio ακολουθούμε τα εξής βήματα :

1. Προσθέτουμε το Firebase sdk στο build gradle του project μας
2. Προσθέτουμε επίσης τα απαραίτητα dependencies στο build gradle applevel

```
dependencies {

    implementation 'com.google.android.material:material:1.9.0'

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'

    implementation 'com.makeramen:roundedimageview:2.3.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.android.gms:play-services-maps:18.2.0'
    implementation 'com.google.firebase:firebase-auth:22.3.1'
    implementation 'com.google.firebase:firebase-firestore:24.11.1'
    implementation 'com.google.firebase:firebase-storage:20.3.0'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

    //implementation "androidx.sqlite:sqlite:2.4.0"//for sqlite
```

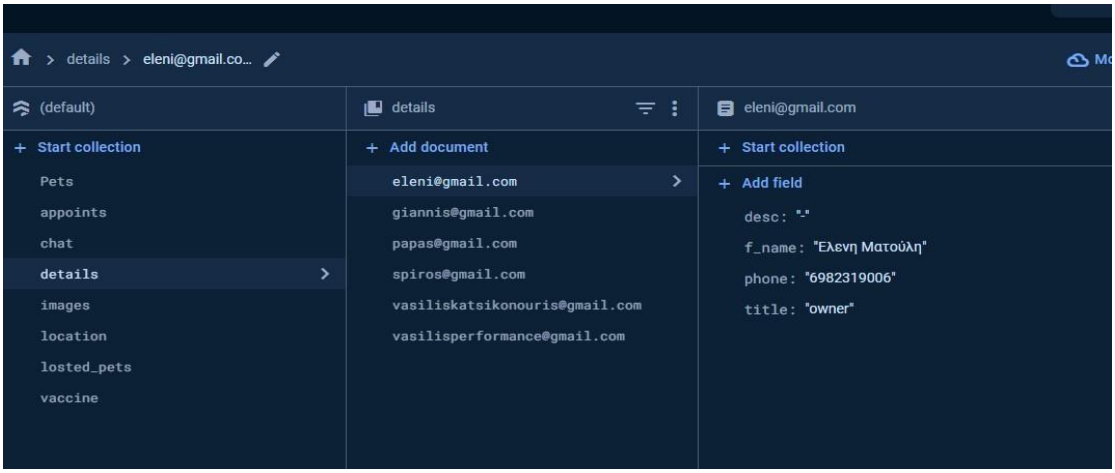
3. Έπειτα ανοίγουμε το Firebase και επιλέγουμε προσθήκη project και ακολουθούμε τα βήματα για να προσθέσουμε το project Μας

Αφού έχουμε κάνει όλα αυτά ,φτιάχνουμε την **sign up** η οποία αρχικά προσθέτει το email και το password του χρήστη(ιδιοκτήτη ή κτηνίατρου)στο Authentication της Firebase

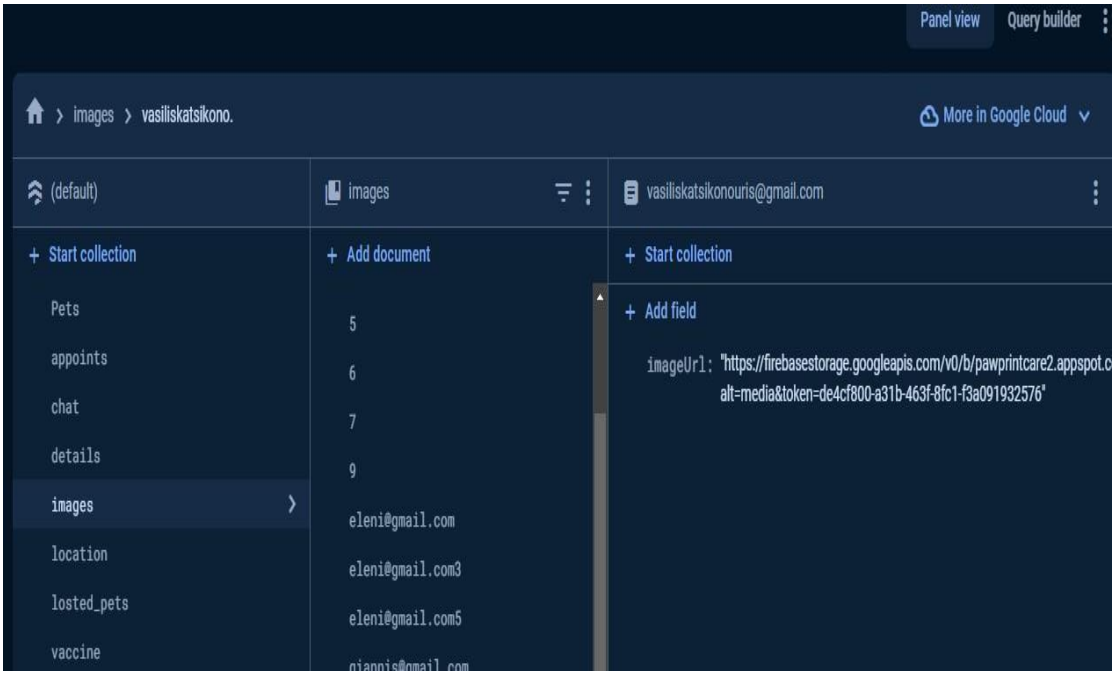
```
mAuth.createUserWithEmailAndPassword(email.getText().toString(), pass.getText().toString())
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // Sign up success, proceed to Firestore data storage
                FirebaseFirestore db = FirebaseFirestore.getInstance();
                CollectionReference usersCollection = db.collection( collectionPath: "details");
                // Create a user data map
                Map<String, Object> userData = new HashMap<>();
                userData.put("f_name", name.getText().toString());
                userData.put("title", type);
                userData.put("phone", phone.getText().toString());
                if(type.equals("vet"))userData.put("desc", desc);
                else userData.put("desc", "-");
                // Add user data to Firestor
                usersCollection.document(email.getText().toString()) DocumentReference
                        .set(userData, SetOptions.merge()) // Use merge to update existing or create new
                        .addOnSuccessListener(aVoid -> {
                            // User data added successfully

                            uploadImageToFirebase(selectedImageUri);//kanoyme kai sace eikona
                            // message("OK","User data added to Firestore successfully!");
                            Log.d(ContentValues.TAG,  msg: "User data added to Firestore successfully!");
                            // Proceed with any additional logic after signup
                        }) Task<Void>
                        .addOnFailureListener(e -> {
                            // Error adding user data
                            Log.w(ContentValues.TAG,  msg: "Error adding user data to Firestore", e);
                            // Handle failure scenario
                        });
            } else {
```

Στην συνέχεια , εφόσον η διαδικασία ολοκληρωθεί τότε αποθηκεύομε τις υπόλοιπες πληροφορίες που περιέχει το sign up ως εξής :

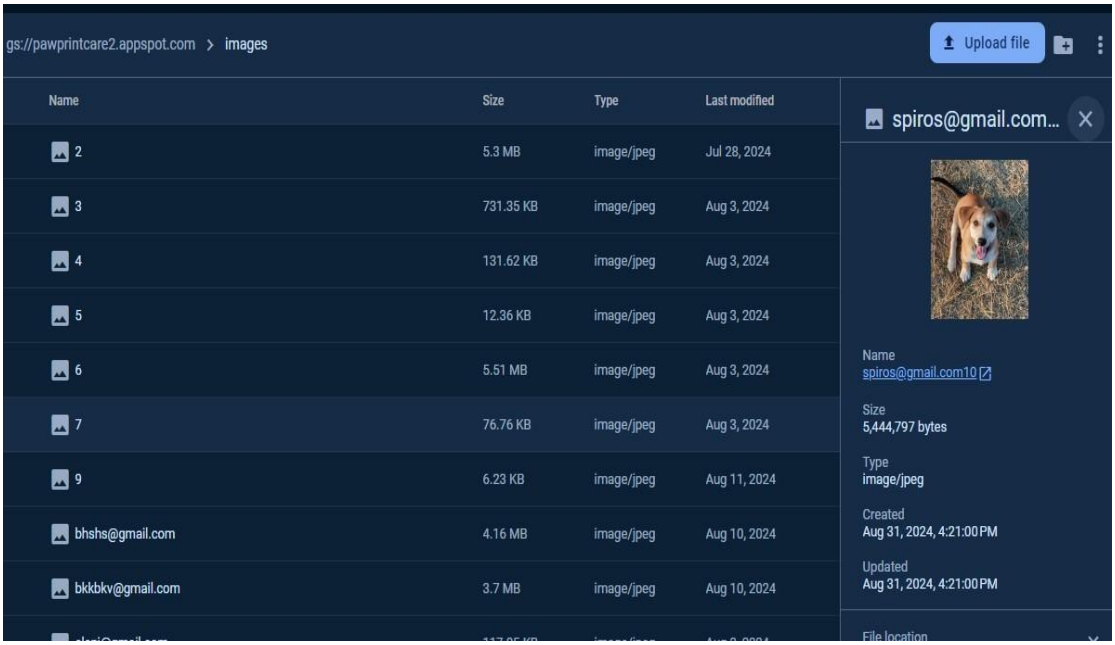
- Δημιουργούμε μια συλλογή details στο Firestore database η οποία περιλαμβάνει το πλήρες όνομα του χρήστη , το τηλέφωνό του, την ιδιότητά του (κτηνίατρος ή ιδιοκτήτης) και μια περιγραφή σε περίπτωση που είναι κτηνίατρος. Την γράφει ο χρήστης και έχει να κάνει με τις σπουδές του και την προϋπηρεσία που έχει σαν κτηνίατρος. Αυτό το έχω κλάνει έτσι ώστε να μπορεί ένας χρήστης να διαλέξει τον καλύτερο για αυτόν



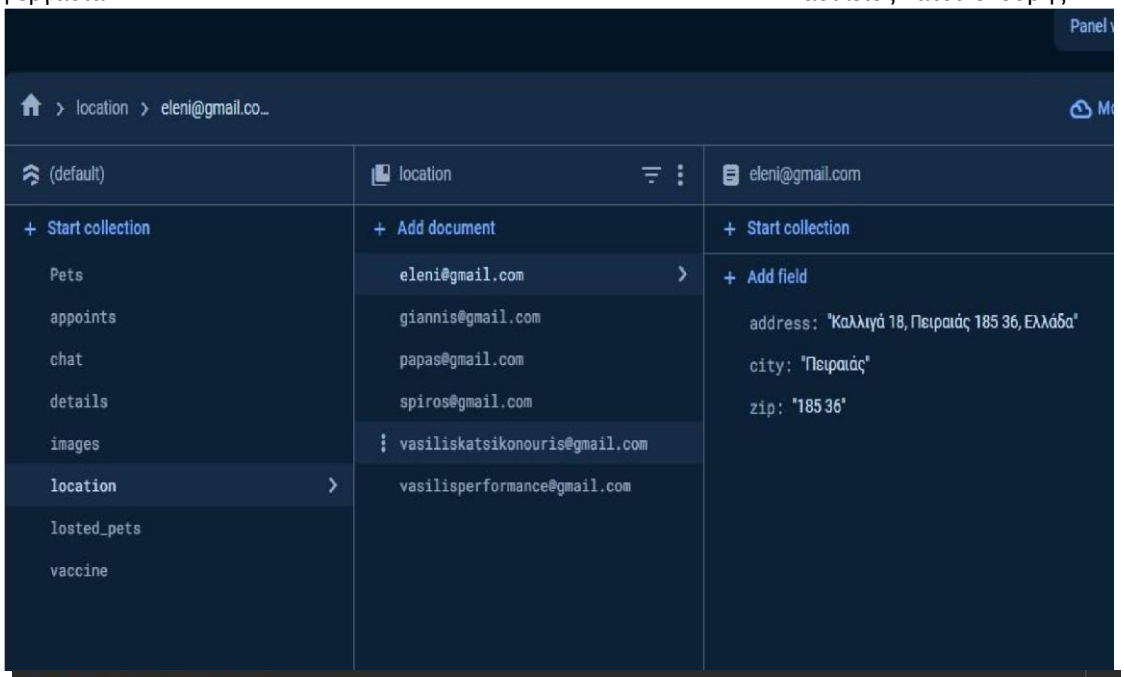
- Δημιουργούμε ένα collection Images στο οποίο αποθηκεύονται ως document το email του χρήστη ή το id του κατοικίδιου αντιστοιχίζοντας το με ένα imageUrl που περιέχει την εικόνα που έβαλε ο χρήστης



Αυτό το ImageUrl οδηγεί στο storage όπου αποθηκεύονται όλες οι εικόνες μας



- Τελευταία αποθήκευση που γίνεται είναι στο collection location όπου εκεί αποθηκεύομε την τοποθεσία του χρήστη



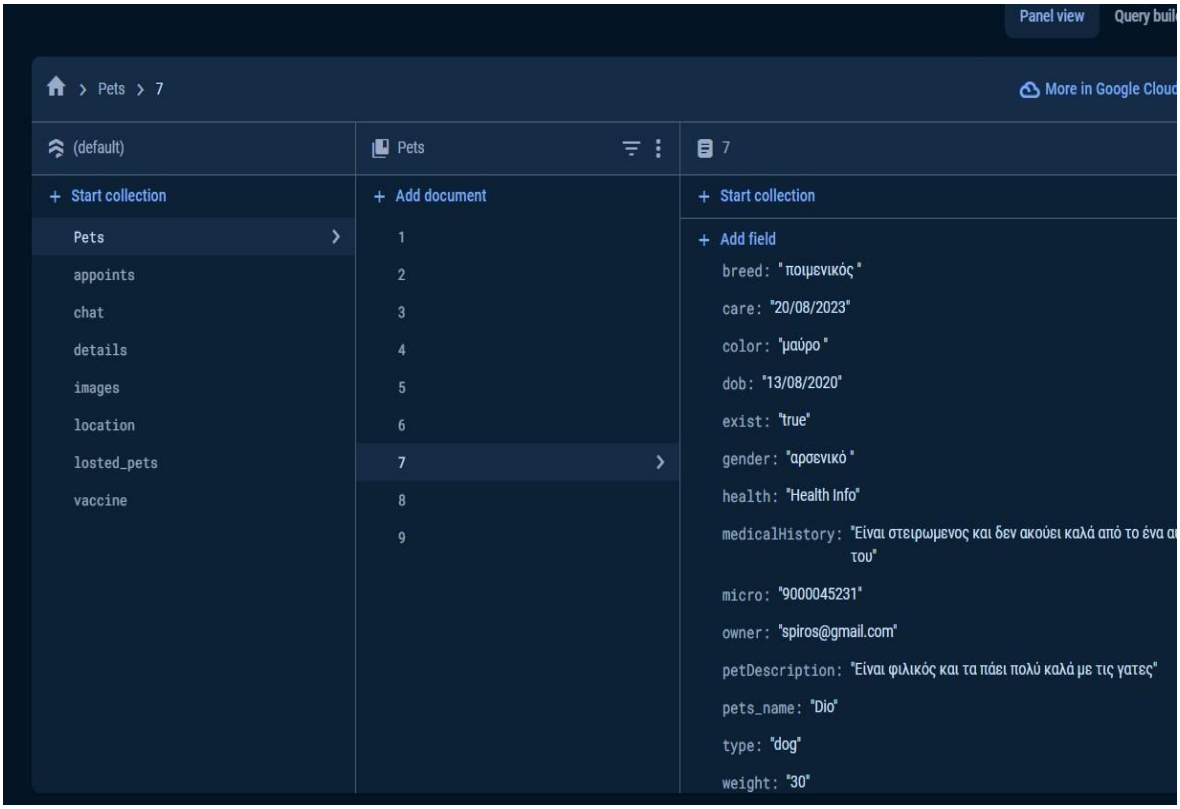

```
public void create_user(View v){
    CollectionReference petsCollection = db.collection( collectionPath: "location");
    petsCollection.get()
            .addOnSuccessListener(queryDocumentSnapshots -> {
                DocumentReference newDocumentRef = petsCollection.document(email);

                // Create a map with user data to be added to the new document
                Map<String, Object> userData = new HashMap<>();
                userData.put("city",city);
                userData.put("address", address);
                userData.put("zip", postalCode);

                newDocumentRef.set(userData, SetOptions.merge())
                        .addOnSuccessListener(aVoid -> {
                            Log.d( tag: "TAG",  msg: "User data added to Firestore successfully!");
                            goon( title: "Save location", message: "Location saved successfully");
                        })
                        .addOnFailureListener(e -> {
                            // Error adding user data
                            Log.w( tag: "TAG",  msg: "Error adding user data to Firestore", e);
                        });

            })
            .addOnFailureListener(e -> {
                // Error getting documents from "Pets" collection
                Log.w( tag: "TAG",  msg: "Error getting documents from 'Pets' collection", e);
                // Handle failure scenario (e.g., display an error message)
            });
}
```

Για images :

```
private void uploadImageToFirebase(Uri imageUri) {//WE NEED THIS
    // Create a reference to store the image in Firebase Storage
    //String userEmail = mAuth.getCurrentUser().getEmail();//αυτο για update image
    StorageReference imageRef = storageRef.child("images/" + email.getText().toString());

    // Upload the image to Firebase Storage
    imageRef.putFile(imageUri) UploadTask
            .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                    // Image uploaded successfully, get the download URL
                    imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                        @Override
                        public void onSuccess(Uri uri) {
                            String imageUrl = uri.toString();
                            // Save the image URL to Firestore
                            saveImageUrlToFirestore(imageUrl);
                        }
                    });
                }
            }) StorageTask<UploadTask.TaskSnapshot>
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    // Handle unsuccessful upload
                    Log.e(ContentValues.TAG,  msg: "Error uploading image to Firebase Storage: " + e.getMessage());
                }
```

```
}//we need this
private void saveImageUrlToFirestore(String imageUrl) {
    // Create a new document in the "images" collection in Firestore
    Map<String, Object> imageInfo = new HashMap<>();
    imageInfo.put("imageUrl", imageUrl);

    db.collection( collectionPath: "images").document(email.getText().toString()) DocumentReference
            .set(imageInfo) Task<Void>
            .addOnSuccessListener(documentReference -> {
                //message("OK","User data added to Firestore successfully!");
                goon( title: "finished",  message: "We are ready ");
                progressBar.setVisibility(View.INVISIBLE);
                editor.putString("type",type);
                editor.putString("email",email.getText().toString());
                editor.apply();
                System.out.println("Document written successfully!");
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    System.err.println("Error writing document: " + e.getMessage());
                }
            });
```

Μέσα στην Firebase υπάρχει και το collection Pets . Σε αυτό το collection αποθηκεύονται τα κατοικίδια του κάθε χρήστη έχοντας σαν document το id τoy κάθε ζώου και μέσα στο collection αποθηκεύονται τα εξής ,

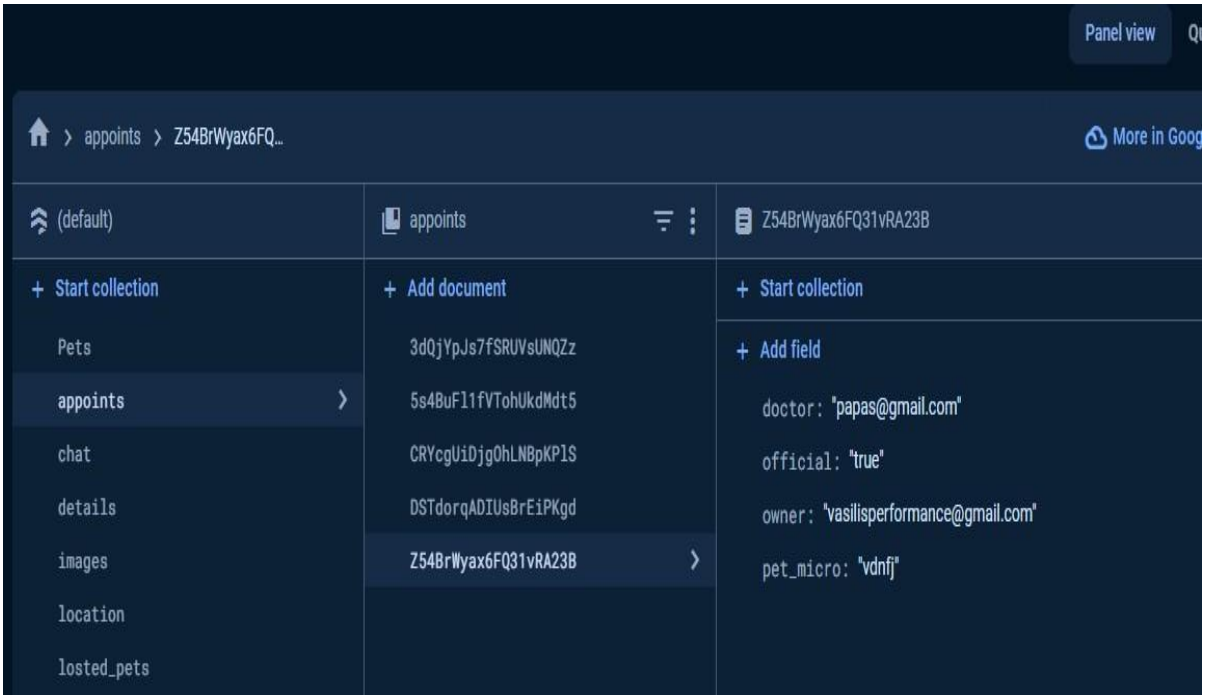


Μέσα στο collection υπάρχει και το email του χρήστη (σαν foreign key) για να αναγνωρίζουμε ποιου είναι το κάθε ζώο

Παράδειγμα εισαγωγής δεδομένων στο Pets

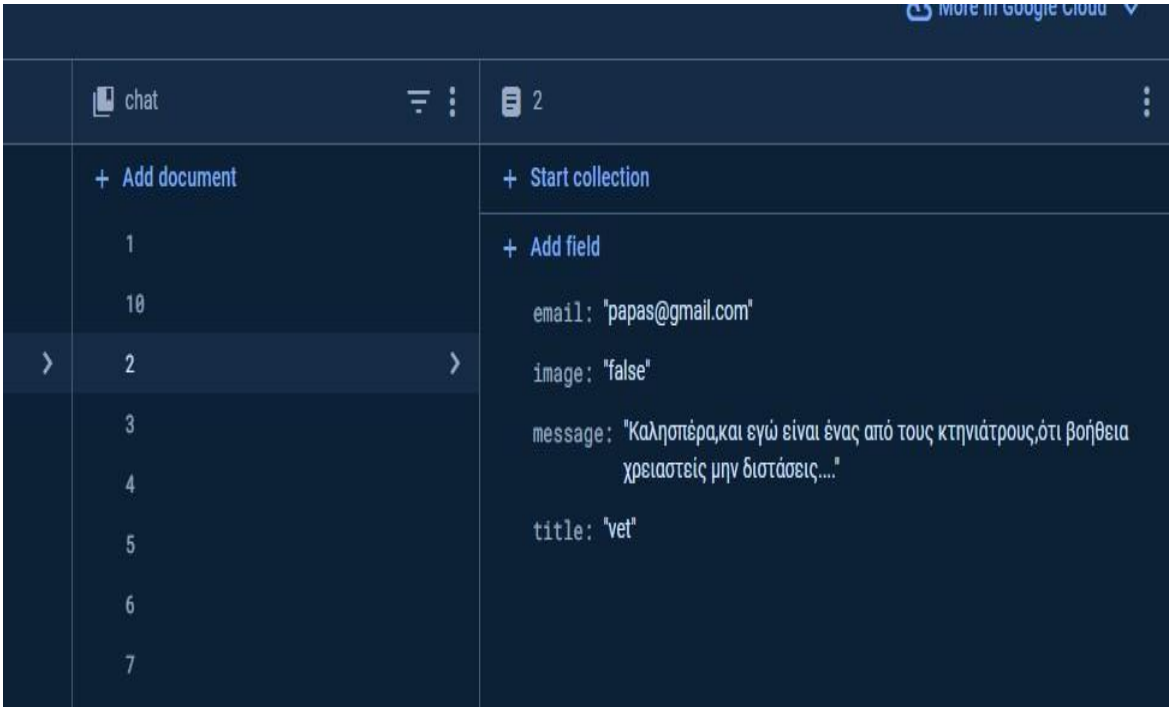
```
CollectionReference petsCollection = db.collection( collectionPath: "Pets");
petsCollection.get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            // Get the number of documents in the collection
            int numberOfDocuments = queryDocumentSnapshots.size();
            Log.d( tag: "TAG",  msg: "Number of documents in 'Pets' collection: " + numberOfDocuments);
            // Create a new document with a generated document ID
            newDocumentId = String.valueOf(numberOfDocuments + 1); // Generate a new ID
            DocumentReference newDocumentRef = petsCollection.document(newDocumentId);

            // Create a map with user data to be added to the new document
            Map<String, Object> userData = new HashMap<>();
            userData.put("Id",newDocumentId);
            userData.put("breed", breed.getText().toString());
            userData.put("dob", dob.getText().toString());
            userData.put("care", care.getText().toString());
            userData.put("color", color.getText().toString());
            userData.put("gender", gender.getText().toString());
            userData.put("exist", "true");
            userData.put("health", health.getText().toString());
            userData.put("micro", micro.getText().toString());
            userData.put("owner", logged_in_user);
            userData.put("pets_name", name.getText().toString());
            userData.put("weight", weight.getText().toString());
            userData.put("medicalHistory", medicalHistory.getText().toString());
            userData.put("petDescription", petDescription.getText().toString());
            userData.put("type", type.getText().toString());

            progressBar=new ProgressBar( context: this);
            progressBar.setPadding( left: 0, top: 0, right: 0, bottom: 10);
            form.removeView(save);
            //save.setVisibility(View.INVISIBLE);
            form.addView(progressBar);

            // Add user data to the new document in the "Pets" collection
            newDocumentRef.set(userData, SetOptions.merge())
                    .addOnSuccessListener(aVoid -> {
                        // User data added successfully
                        uploadImageToFirebase(pet_image); // Example: Upload image to Firebase
```

Ακόμα έχουμε το collection appoints μέσα στο οποίο αποθηκεύονται τα αιτήματα που κάνει ένας χρήστης σε κάποιον κτηνίατρο .Αρχικά , μόλις γίνει το request υπάρχει ένα πεδίο το οποίο λέγετε official . Εάν αυτό είναι false σημαίνει πως ο ιατρός δεν έχει δεκτή ακόμα το request αλλά όταν το δεκτή γίνεται true . Τα δεδομένα που αποθηκεύονται εδώ είναι το email του χρήστη, το email του ιατρού και η μεταβλητή official (η petmicro δεν χρησιμοποιείται γιατί ένας ιατρός προσέχει όλα τα κατοικίδια του χρήστη)



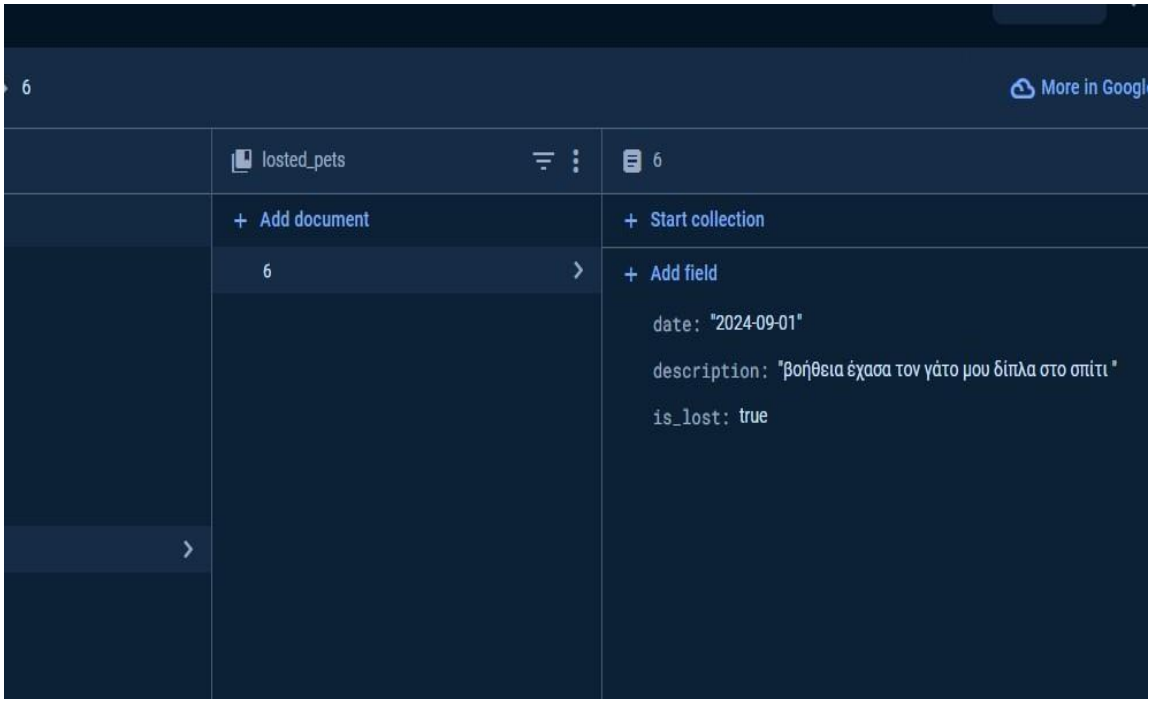

```
public void addAppoint(String ownerName,String doctorName,String petMicro){
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    // Create a new document with fields owner, official, doctor, and pet_micro
    Map<String, Object> appointmentData = new HashMap<>();
    appointmentData.put("owner", ownerName);
    appointmentData.put("official", "false");  // or boolean true if that's the intended type
    appointmentData.put("doctor", doctorName);
    appointmentData.put("pet_micro", petMicro);

    // Add the new document to the "appoints" collection
    db.collection( collectionPath: "appoints") CollectionReference
            .add(appointmentData) Task<DocumentReference>
            .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                @Override
                public void onSuccess(DocumentReference documentReference) {
                    // Document was successfully added
                    System.out.println("Document added with ID: " + documentReference.getId());
                }
            })
            .addOnFailureListener(e -> {
                // Handle failure
                System.err.println("Error adding document: " + e.getMessage());
```

Ακόμα , υπάρχει το chat collection μέσα στο οποίο αποθηκεύονται τα μηνύματα που στέλνει κάθε χρήστη παρέα με email και μια μεταβλητή image η οποία εάν είναι true σημαίνει πως αυτό το μήνυμα είναι φωτογραφία και πρέπει να γίνει η αντιστοιχεί λειτουργία με την αποθήκευση αυτής στο firebase storage



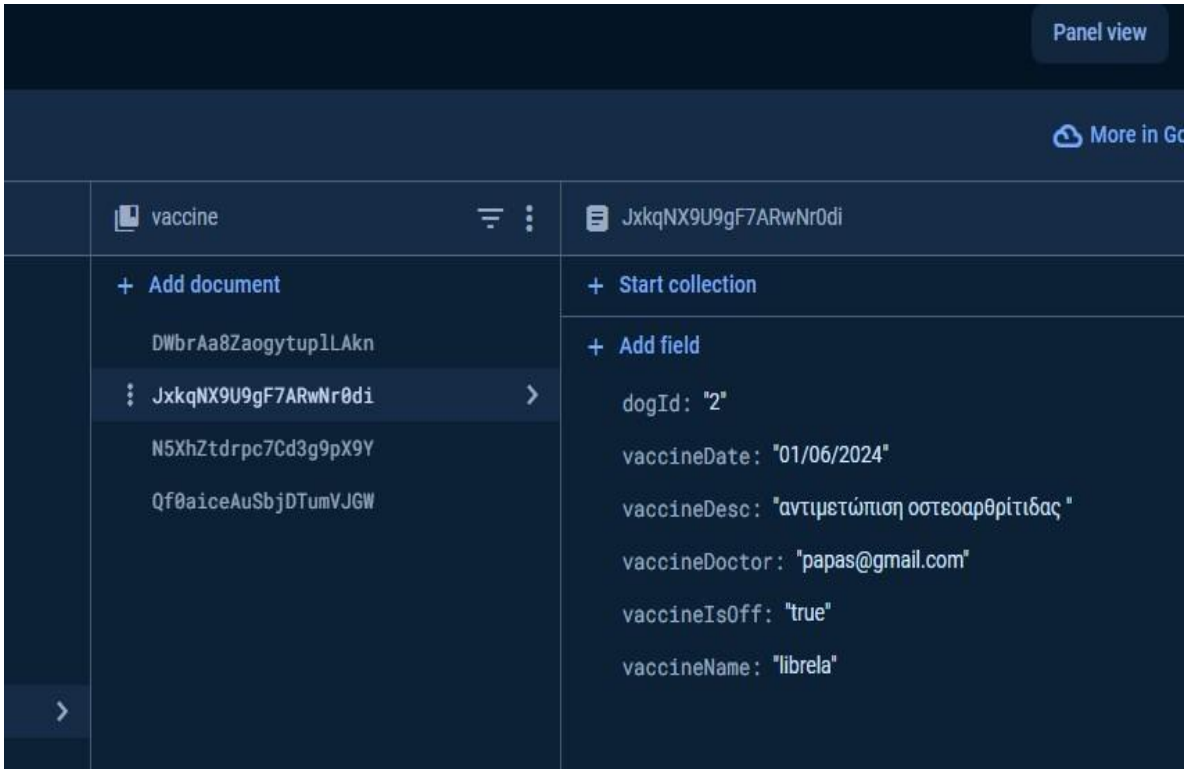

```
public void upload_message(String message,String is_image){
    CollectionReference petsCollection = firestore.collection( collectionPath: "chat");
    petsCollection.get()
            .addOnSuccessListener(queryDocumentSnapshots -> {
                // Get the number of documents in the collection
                int numberOfDocuments =  queryDocumentSnapshots.size();
                Log.d( tag: "TAG",  msg: "Number of documents in 'Pets' collection: " + numberOfDocuments);

                // Create a new document with a generated document ID
                newDocumentId = String.valueOf(numberOfDocuments + 1); // Generate a new ID
                DocumentReference newDocumentRef = petsCollection.document(newDocumentId);
                // Create a map with user data to be added to the new document
                Map<String, Object> userData = new HashMap<>();
                userData.put("email",logged_in_user);
                userData.put("image",is_image);
                userData.put("title", user_title);
                if(is_image.equals("false"))userData.put("message", message);
                else {userData.put("message", message+newDocumentId);
                    uploadImageToFirebase(send_image, imageid: message+newDocumentId);
                }
```

Ένα ακόμα collection που έχω δημιουργήσει είναι το Losted_pets . Σε αυτό το collection λοιπόν αποθηκεύονται τα δεδομένα ενός κατοικίδιου για το οποίο ο ιδιοκτήτης του έχει δηλώσει εξαφάνιση . Μέσα σε αυτό έχουμε την ημερομηνία εξαφάνισης , μια περιγραφή του ζώου αλλά και την μεταβλητή is_lost η οποία είναι true και όταν βρεθεί το ζώο γίνεται false. Αυτό το έχω κάνει έτσι ώστε να μην χρειαστεί να διαγράψω την εγγραφή αφού βρεθεί το ζώο και να την κρατήσω για αρχείο.




```
if(uplaod){
    Map<String, Object> petData = new HashMap<>();
    //LocalDate currentDate = LocalDate.now();
    petData.put("date", lost_date);
    petData.put("description", descr);
    petData.put("is_lost", true);
    // Add a new document with the specified ID
    firestore.collection( collectionPath: "losted_pets").document(id) DocumentReference
            .set(petData) Task<Void>
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Log.d( tag: "TAG",  msg: "DocumentSnapshot successfully written!");
                    sendAlert(descr,lost_city);
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.w( tag: "TAG",  msg: "Error writing document", e);
                }
            });
```

Το τελευταίο collection που δημιούργησα για την εφαρμογή μου είναι το vaccine το οποίο έχει να κάνει με τα εμβόλια που έχει κάνει ή πρόκειται να κάνει ένα ζώο . Μέσα σε αυτό το collection αποθηκεύω το id του ζώου , το vaccineDate ,μια περιγραφή για το εμβόλιο ,το email του ιατρού που θα το κάνει , το όνομα του εμβολίου και μια μεταβλητή vaccineIsOff η οποία είναι false και θα γίνει true εάν ο πελάτης αποδεχτεί αυτό το ραντεβού για εμβόλιο



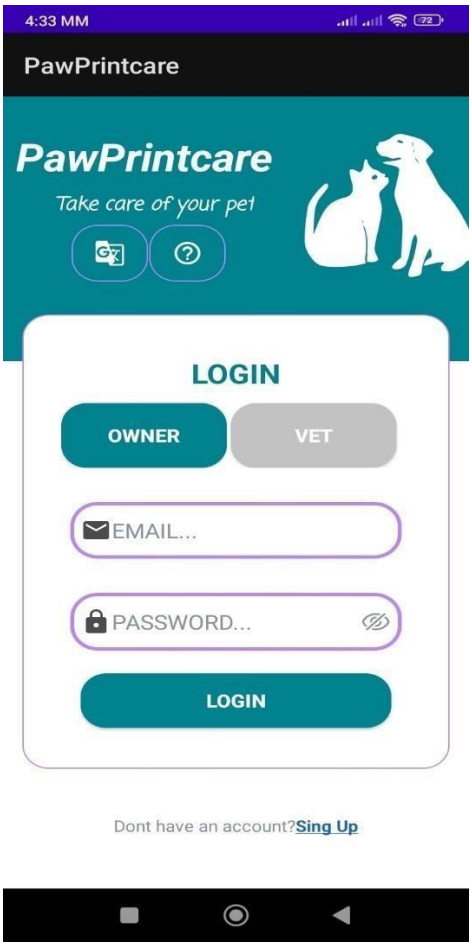

```java
public void setVaccineAppoint(String id,String vaccine,String vaccineDesc,String vaccineDate,String doctor){
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    // Create a new document with fields owner, official, doctor, and pet_micro
    Map<String, Object> appointmentData = new HashMap<>();
    appointmentData.put("dogId", id);
    appointmentData.put("vaccineDate", vaccineDate);  // or boolean true if that's the intended type
    appointmentData.put("vaccineDesc", vaccineDesc);
    appointmentData.put("vaccineDoctor", doctor);
    appointmentData.put("vaccineIsOff", "false");
    appointmentData.put("vaccineName", vaccine);

    // Add the new document to the "appoints" collection
    db.collection( collectionPath: "vaccine") CollectionReference
            .add(appointmentData) Task<DocumentReference>
            .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                @Override
                public void onSuccess(DocumentReference documentReference) {
                    // Document was successfully added
                    System.out.println("Document added with ID: " + documentReference.getId());
                }
            })
            .addOnFailureListener(e -> {
                // Handle failure
                System.err.println("Error adding document: " + e.getMessage());
            });
```

Στήσιμο του Front -end

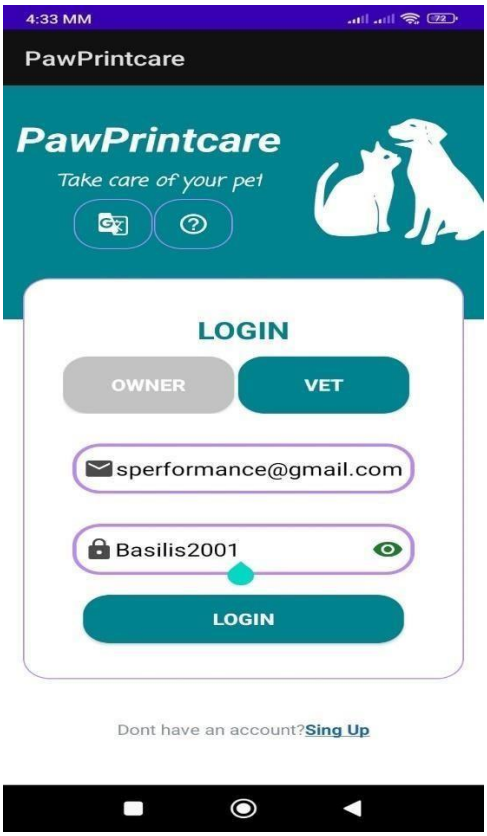
MainActivity

Με το που ανοίξει ο χρήστης την εφαρμογή ,θα οδηγηθεί στο Main activity , στο οποίο μπορεί να συνδεθεί ως απλός ιδιοκτήτης ή ως κτηνίατρος

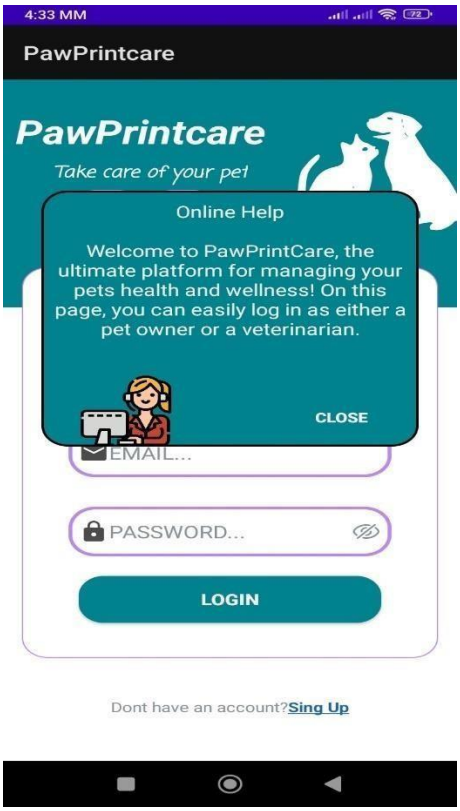
συμπληρώνοντας τα απαραίτητα στοιχεία τα οποία είναι το email του και ο κωδικός του.



Επίσης στο κάτω μέρος αυτού του activity υπάρχει το signup κουμπί μέσο του οποίου ένας νέος χρήστης μπορεί να δημιουργήσει καινούριο λογαριασμό και για τα της δύο κατηγορίες.



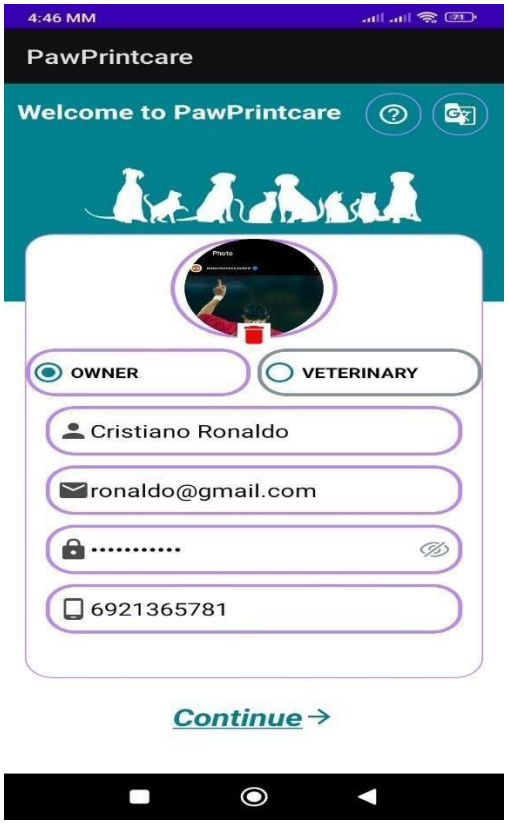
Ακόμα σε αυτό το activity πάνω αριστερά υπάρχουν δύο κουμπιά μέσο των οποίον ένας χρήστης μπορεί να αλλάξει την γλώσσα της εφαρμογής σε αγγλικά ή ελληνικά αλλά και να ζητήσει βοήθεια σχετικά με το τη πρέπει να κάνει στην παρόν σελίδα



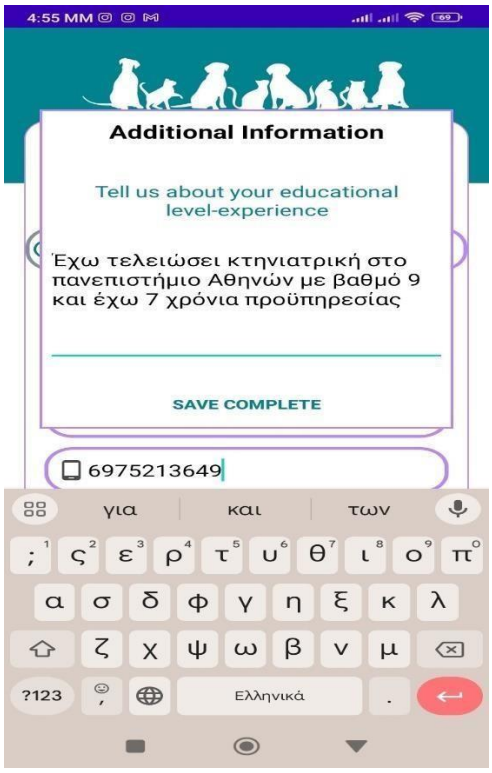
SignUpActivity

Εάν δεν έχουμε λογαριασμό και πατήσουμε το sign up button θα προηγηθούμε στο sign up activity . Σε αυτήν την σελίδα μπορούμε να δημιουργήσουμε έναν νέο λογαριασμό πολύ ευκολά τόσο για ένα απλό χρήστη όσο και για ένα νέο κτηνίατρο . Η διαδικασία αυτή απαιτεί τα εξής:

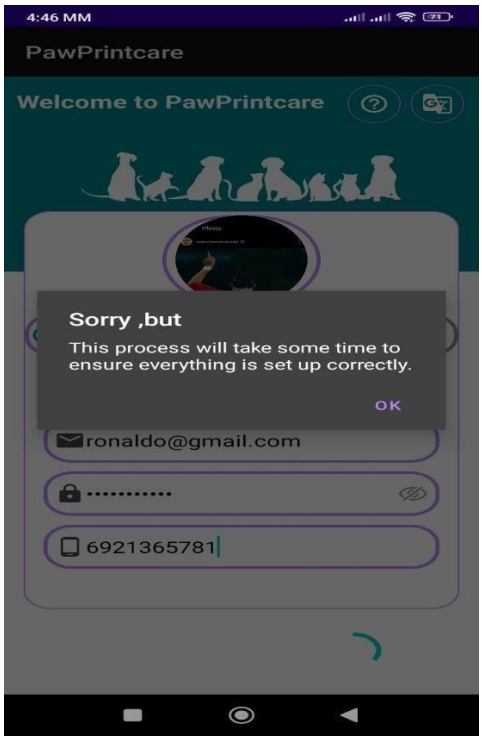
1. Το πλήρες όνομα του χρήστη (full name)
2. Το email του χρήστη (email)
3. Ένα έγκυρο τηλέφωνο (phone number)
4. Έναν κωδικό (password)
5. Μια φωτογραφία για το προφίλ του χρήστη (photo profile)



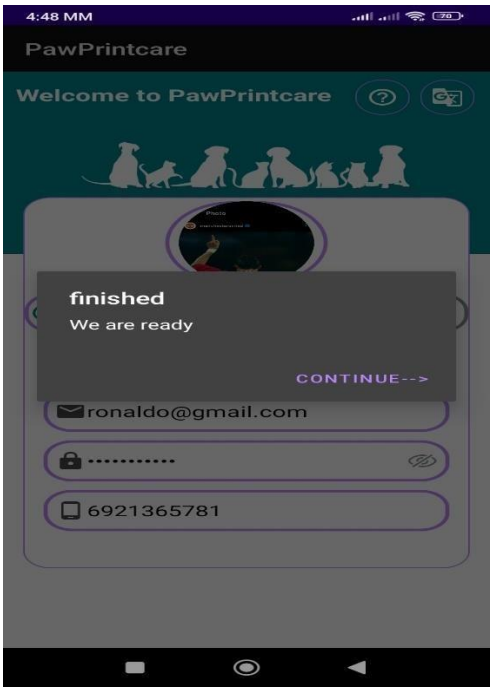
Στην περίπτωση όμως που ο νέος χρήστης είναι κτηνίατρος τότε θα εμφανιστεί ένα παράθυρο μέσα στο οποίο πρέπει να γράψει μια περίληψή για αυτόν , κατά προτιμήσει προϋπηρεσία και σπουδές



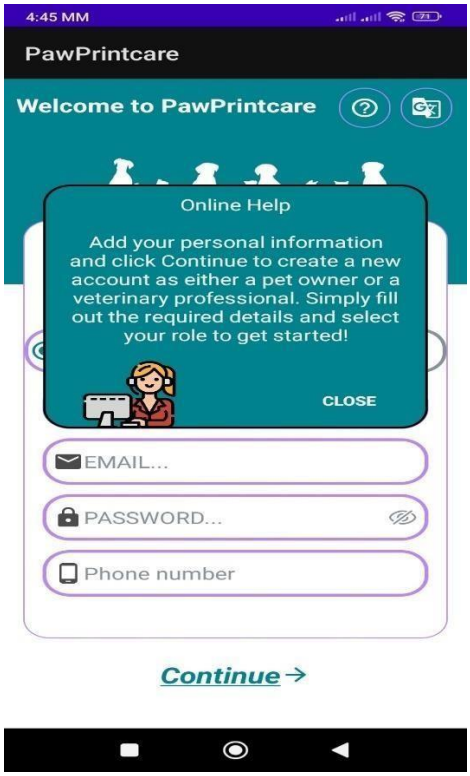
Εφόσον έχουμε συμπληρώσει όλα τα απαραίτητα στοιχεία , θα εμφανιστεί ένα μήνυμα το οποίο θα μας προειδοποιεί για την μια ολιγόλεπτη καθυστέρηση καθώς γίνεται έλεγχος εγκυρότητας των στοιχείων που δώσατε



Μετά τον επιτυχή έλεγχο – αποθήκευση των στοιχείων σας εμφανίζεται ένα μήνυμα το οποίο πατάτε συνέχεια έτσι ώστε να ολοκληρωθεί η διαδικασία εγγραφής σας



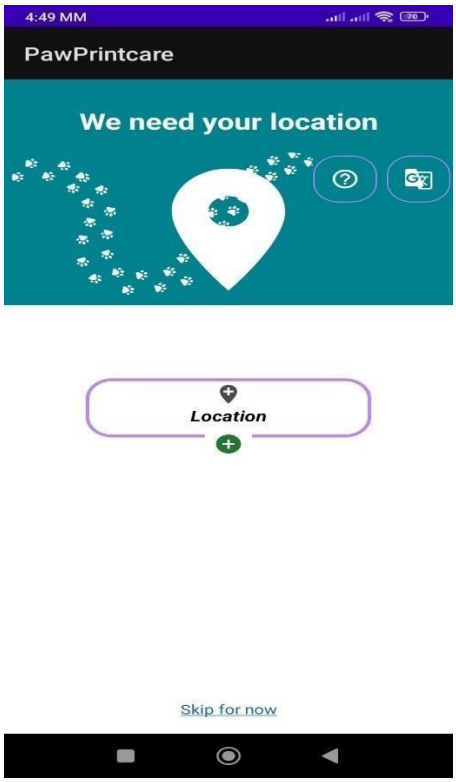
Και σε αυτό το activity υπάρχουν δύο κουμπιά όπου με το ένα γίνεται μετάφραση και με το άλλο εμφανίζεται μια online Βοήθεια για να κάνει ποιο εύκολη την χρήση της εφαρμογής .



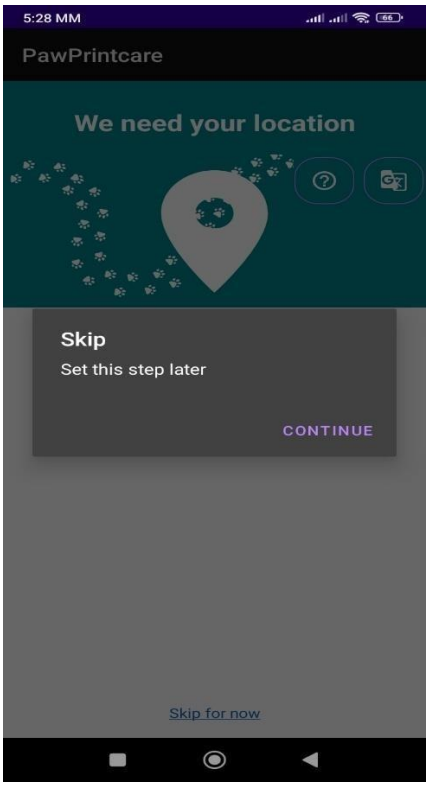
LocationActivity

Μετά την signup σελίδα κατευθυνόμαστε στην location activity, η οποία ζητάει από τον χρήστη να ανοίξει την τοποθεσία του έτσι ώστε να πάρει την τοποθεσία του . Αυτή η πληροφορία θα βοηθήσει την εφαρμογή έτσι ώστε να πρωτινή κτηνίατρους που βρίσκονται κοντά στον χρήστη αλλά και σε περίπτωση που

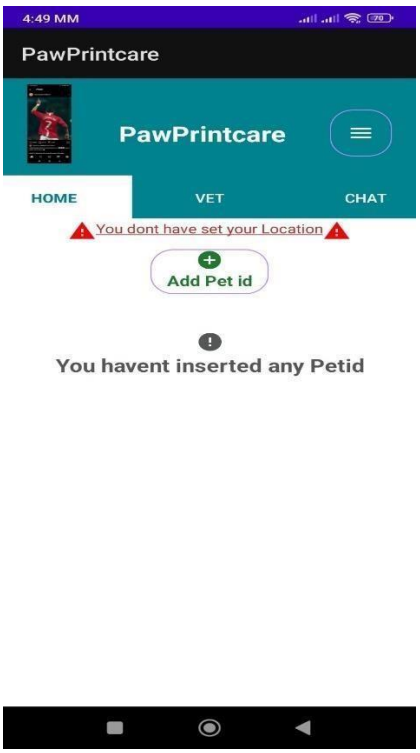
κάποιος χάσει το κατοικίδιο του να έχει την τοποθεσία του . Η τοποθεσία αυτή αφορά την μόνιμη κατοικία για τον ιδιοκτήτη και την τοποθεσία του καταστήματος του κτηνίατρου .



Γι’ αυτό εάν δεν βρίσκεται σε αυτήν την περιοχή την στιγμή που δημιουργεί τον λογαριασμό, υπάρχει ένα κουμπί στο κάτω μέρος της σελίδας με το οποίο μπορεί να κάνει skip προσωρινά αυτήν την λειτουργεία μέχρι ο χρήστης να είναι στην κατάλληλη τοποθεσία



Αυτό είναι το μήνυμα που εμφανίζεται κάθε φορά στο Home page του κάθε χρήστη όταν δεν έχουν θέσει την τοποθεσία τους



OwnerPageActivity

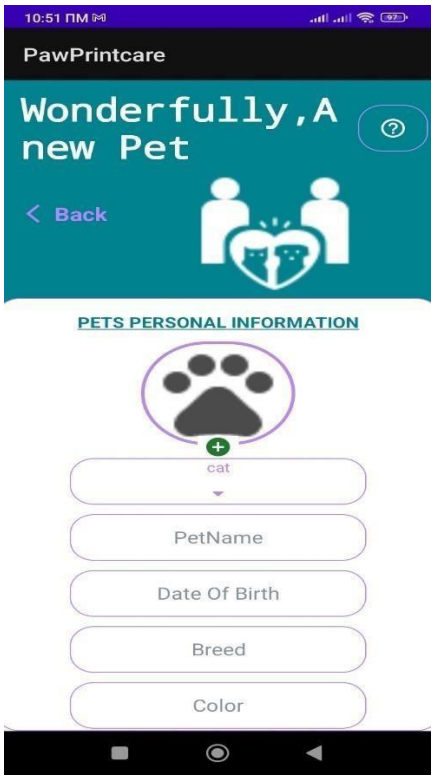
Εφόσον έχουμε βάλει ένα έγκυρο email και έναν σωστό κωδικό που να αντιστοιχεί σε ένα χρήστη ιδιοκτήτη θα προηγηθούμε στην owner page σελίδα . Μέσα σε αυτήν υπάρχουν 3 βασικά navigation button

Εάν πατήσουμε πάνω στην ταυτότητα του ζώου εμφανίζεται ένα menu μέσα στο οποίο βρίσκονται περισσότερες πληροφορίες για αυτό αλλά και δύο κουμπιά όπου με το ένα μπορείς να διαγράψεις αυτό το κατοικίδιο από την εφαρμογή και με το άλλο να ανοίξεις το βιβλιάριο υγείας και να δεις τα εμβόλια που έχεις κάνει αλλά αυτά που προτείνει ο κτηνίατρος . Σε αυτά τα εμβόλια που προτείνει ο κτηνίατρος υπάρχουν δύο κουμπιά , ένα για να κλείσεις το ραντεβού για την ημερομηνία που αναφέρετε και το άλλο για να το απορρίψεις

AppointActivity

Το home button οδηγεί στην αρχική σελίδα του χρήστη στην οποία υπάρχουν οι ταυτότητές των ζών που έχει πρόσθεση στην εφαρμογή αλλά και ένα κουμπί “Add Pet id ” όπου μπορεί να προσθέσει και άλλο κατοικίδιο

PetidActivity



Το vet κουμπί ανοίγει την σελίδα vet στην οποία εμφανίζεται ο κτηνίατρος με τον οποίο συνεργαζόμαστε . Στην σελίδα αυτοί υπάρχουν όλες οι χρήσιμες πληροφορίες για αυτόν όπως email , τηλέφωνο και οδός - αριθμός του καταστήματός του .όλες αυτές τις πληροφορίες μπορεί να τις πατήσει ο χρήστης και να κάνει τα εξής :

1. Όταν πατήσει ο χρήστης το τηλέφωνο του κτηνίατρου τότε ανοίγουν οι κλήσεις στο τηλέφωνό του με σκοπό να καλέσει τον κτηνίατρο

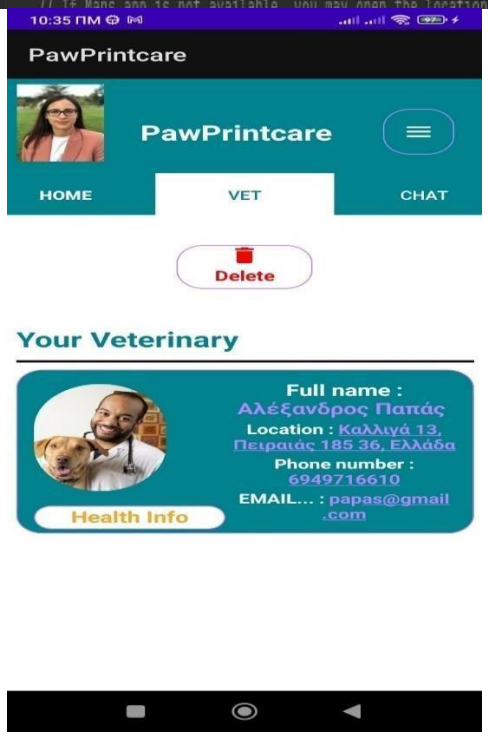
```
helper.findDetails(doctor)
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                DocumentSnapshot document = task.getResult();
                if (document != null) {
                    SpannableString spannableFullName = new SpannableString("Full name" + " : " + document.getString( field: "f_name"));
                    spannableFullName.setSpan(new ForegroundColorSpan(ContextCompat.getColor( context: this, R.color.purple_200)),
                            "Full name".length() + 3, // Start index of name text
                            spannableFullName.length(), // End index (exclusive) of name text
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);

                    // Set the SpannableString to fullName TextView
                    fullName.setText(spannableFullName);
                   // fullName.setText("-"+getString(R.string.name)+" : "+document.getString("f_name"));
                    SpannableString spannablePhone = new SpannableString("Phone number" + " : " + document.getString( field: "phone"));
                    spannablePhone.setSpan(new ForegroundColorSpan(ContextCompat.getColor( context: this, R.color.purple_200)),
                            "Phone number".length() + 3, // Start index of phone text
                            spannablePhone.length(), // End index (exclusive) of phone text
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                    spannablePhone.setSpan(new UnderlineSpan(),
                             start: 14, // Start index of the whole text
                            spannablePhone.length(), // End index of the whole text
                            Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                    phoneNumber.setText(spannablePhone);
                    phoneNumber.setOnClickListener(view -> {
                        String phone = phoneNumber.getText().toString().replaceAll( regex: "[^0-9]",  replacement: ""); // Extract phone number from TextView

                        // Create an Intent to initiate a phone call
                        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + phone));
                        startActivity(intent);
```

2. Όταν πατήσει το email τότε θα ανοίξει το email του τηλεφώνου του έτσι ώστε να στείλει email στον κτηνίατρο

```
                    });
                    SpannableString spannableEmail = new SpannableString("EMAIL..." + " : " + doctor);
                    spannableEmail.setSpan(new ForegroundColorSpan(ContextCompat.getColor( context: this, R.color.purple_200)),
                            "EMAIL...".length() + 3, // Start index of phone text
                            spannableEmail.length(), // End index (exclusive) of phone text
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                    spannableEmail.setSpan(new UnderlineSpan(),
                             start: 14, // Start index of the whole text
                            spannableEmail.length(), // End index of the whole text
                            Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                    emailDoctor.setText(spannableEmail);
                    emailDoctor.setOnClickListener(view -> {
                        String recipientEmail = doctor; // Replace with the doctor's email address
                        String subject = "Your Email Subject Here";
                        String body = "Hello Doctor, \n\n"; // Optional: You can pre-fill the email body if needed

                        // Create an Intent to initiate an email
                        Intent intent = new Intent(Intent.ACTION_SENDTO);
                        intent.setData(Uri.parse("mailto:")); // Only email apps should handle this
                        intent.putExtra(Intent.EXTRA_EMAIL, new String[]{recipientEmail});
                        intent.putExtra(Intent.EXTRA_SUBJECT, subject);
                        intent.putExtra(Intent.EXTRA_TEXT, body);
                        startActivity(intent);
                    });
                    //phoneNumber.setText("-"+getString(R.string.phone)+" : "+document.getString("phone"));
                }
                else{
                    fullName.setText("-"+"Full name"+" : "+"");
                    phoneNumber.setText("-"+"Phone number"+" : "+"");
                }
```

3. Όταν πατήσει το location τότε ανοίγουν οι χάρτες για πλοήγηση στο κατάστημα του κτηνίατρου

```
if (task.isSuccessful()) {
    DocumentSnapshot document = task.getResult();
    if (document != null) {
        // Create a SpannableString for dLocation with blue color and underline
        SpannableString spannableLocation = new SpannableString("Location" + " : " + document.getString( field: "address"));
        spannableLocation.setSpan(new ForegroundColorSpan(ContextCompat.getColor( context: this, R.color.purple_200)),
                "Location".length() + 3, // Start index of phone text
                spannableLocation.length(), // End index (exclusive) of phone text
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
        spannableLocation.setSpan(new UnderlineSpan(),
                start: 11, // Start index of the whole text
                spannableLocation.length(), // End index of the whole text
                Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
        dLocation.setText(spannableLocation);
        dLocation.setOnClickListener(view -> {
            String location = dLocation.getText().toString().substring(dLocation.getText().toString().indexOf(":") + 1).tr

            // Create an Intent to open the location in Maps
            Uri gmmIntentUri = Uri.parse("geo:0,0?q=" + Uri.encode(location));
            Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
            mapIntent.setPackage("com.google.android.apps.maps"); // Specify package to ensure only Maps handles this Inte
            if (mapIntent.resolveActivity(getPackageManager()) != null) {
                startActivity(mapIntent);
            } else {
                Log.e( tag: "MainActivity",  msg: "Google Maps app not found, opening in browser.");
```



Το κουμπί chat οδηγεί σε μια ομαδική συνομιλία στην οποία όλοι οι χρήστες της εφαρμογής μπορούν να μιλάνε μεταξύ τους και να στέλνουν φωτογραφίες

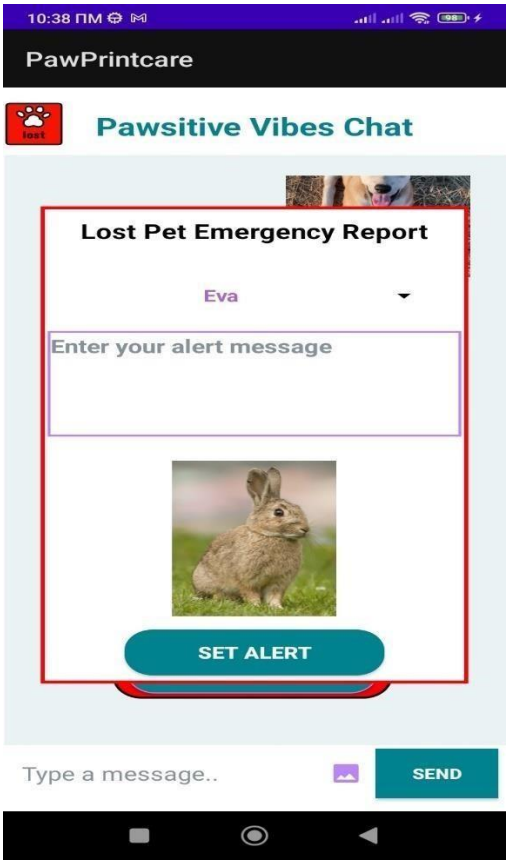
ChatActivity


```
public void addmessage(String title,String user,String message){
    Log.d( tag: "title"+title, msg: "user_title"+user_title+" "+message);
    if(!message.isEmpty()){
        TextView name=new TextView( context: this);
        name.setText(title+" : "+user); // Set user's name
        name.setTypeface( tf: null, Typeface.BOLD);
        name.setTextSize(14); // Set text size
        name.setGravity(Gravity.LEFT);

        chat.addView(name);

        TextView nameTextView = new TextView( context: this);
        nameTextView.setText(message); // Set user's name
        nameTextView.setTypeface( tf: null, Typeface.BOLD);
        nameTextView.setTextSize(18); // Set text size
        nameTextView.setBackgroundResource(R.drawable.custome_border);
        LinearLayout.LayoutParams nameParams2= new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT
        );
        LinearLayout.LayoutParams nameParams = new LinearLayout.LayoutParams(
                width: 500,
                LinearLayout.LayoutParams.WRAP_CONTENT
        );
        if(logged_in_user.equals(user)){
```

```
        if(logged_in_user.equals(user)){
            nameTextView.setBackgroundResource(R.drawable.button_corner);
            nameTextView.setTextColor(ContextCompat.getColor( context: Chat.this, R.color.white));
            name.setTextColor(ContextCompat.getColor( context: Chat.this, R.color.yellow));
            nameParams.setMargins( left: 5,  top: 0,  right: 5,  bottom: 25);
            nameParams2.setMargins( left: 2,  top: 0,  right: 5,  bottom: 0);
        }
        else{
            if(title.equals("vet"))name.setTextColor(ContextCompat.getColor( context: Chat.this, R.color.purple
            else name.setTextColor(ContextCompat.getColor( context: Chat.this, R.color.black));
            nameTextView.setTextColor(ContextCompat.getColor( context: Chat.this, R.color.black));
            nameParams.setMargins( left: 150,  top: 0,  right: 5,  bottom: 25);
            nameParams2.setMargins( left: 150,  top: 0,  right: 5,  bottom: 0);
            nameTextView.setBackgroundResource(R.drawable.custome_border);
        }
        name.setLayoutParams(nameParams2);
        nameTextView.setLayoutParams(nameParams);
        nameTextView.setPadding( left: 15, top: 15, right: 15, bottom: 15);
        chat.addView(nameTextView);
    }
}
```

```
private void retrieveImageUrlFromFirestore(String documentId, ImageView imageView) {
    // Get reference to the document in Firestore
    DocumentReference docRef = firestore.collection( collectionPath: "images").document(documentId);

    // Retrieve the document data asynchronously
    docRef.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            if (documentSnapshot.exists()) {
                // Document exists, retrieve imageUrl
                String imageUrl = documentSnapshot.getString( field: "imageUrl");

                // Load image into ImageView using Glide
                if (imageUrl != null && !imageUrl.isEmpty()) {
                    loadImageWithGlide(imageUrl, imageView);
                } else {
                    // Handle case where imageUrl is null or empty
                    imageView.setImageResource(R.drawable.ic_baseline_person_24);
                }
            } else {
                // Document does not exist
                imageView.setImageResource(R.drawable.ic_baseline_person_24);
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
```

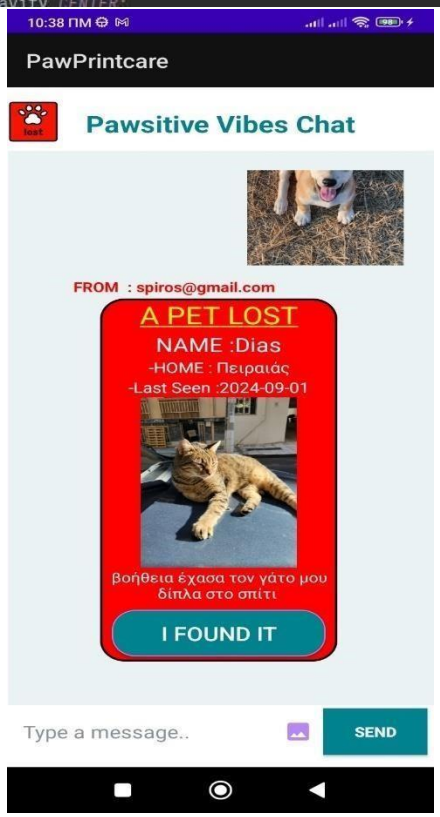
Ακόμα, στο πάνω αριστερά μέρος της σελίδας βρίσκεται ένα emergency button με το οποίο ο κάθε ιδιοκτήτης ζώου μπορεί να ανεβάσει ένα post στο chat το οποία θα έχει να κάνει με την εξαφάνιση κάποιου κατοικίδιου του . Αυτό το μήνυμα περιέχει χρήσιμες πληροφορίες για την αναζήτηση – αναγνώριση του ζώου αλλά και ένα κουμπί με το οποίο μπορεί κάποιος να επικοινωνήσει με τον ιδιοκτήτη του χαμένου ζώου με σκοπό είτε να του πει ότι το βρήκε είτε για να ζητήσει περισσότερες πληροφορίες για αυτό .


10:38 ΠΜ
PawPrintcare
Pawsitive Vibes Chat
Lost Pet Emergency Report
Eva
Enter your alert message
SET ALERT
Type a message..
SEND

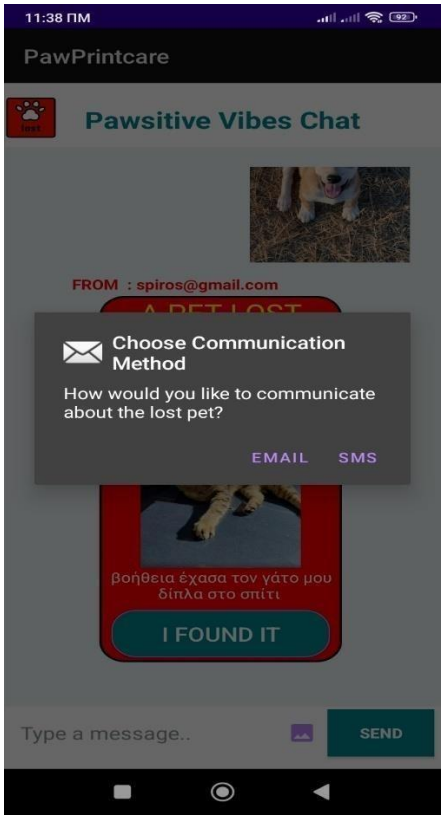
```
private void sendAlert(String desc, String city) {
    Context context = wrapper.getApplicationContext();

    // Create the notification channel if necessary (for Android 8.0 and above)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel(
                id: "alert_channel_id",
                name: "Alert Notifications",
                NotificationManager.IMPORTANCE_DEFAULT
        );
        channel.setDescription("Channel for alert notifications");

        // Register the channel with the system
        NotificationManager notificationManager = context.getSystemService(NotificationManager.class);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context,  channelId: "alert_channel_id")
            .setSmallIcon(R.drawable.launcher1) // Set your own icon here
            .setContentTitle("A pet lost in "+"  "+city)
            .setContentText(desc)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setAutoCancel(true); // Automatically remove the notification when tapped

    // Get the NotificationManager
    NotificationManager notificationManager = (NotificationManager) wrapper.getSystemService(Context.NOTIFICATION_SERVICE);

    // Send the notification with a unique ID (you can replace 1 with any ID)
    if (notificationManager != null) {
        notificationManager.notify( id: 1, builder.build());
    }
```

```
@SuppressLint({"SetTextI18n", "ResourceAsColor"})
void addalertmessage(String image_p, String name_p, String descr,String id, boolean uplaod,String lost_city,String lost_date,String alert_user ){

    LinearLayout.LayoutParams nameParams = new LinearLayout.LayoutParams(
            width: 550,
            LinearLayout.LayoutParams.WRAP_CONTENT
    );
    nameParams.setMargins( left: 70, top: 0, right: 0, bottom: 0);
    TextView details = new TextView( context: this);
    details.setId(View.generateViewId());
    details.setTextColor(Color.RED);
    details.setTypeface( tf: null, Typeface.BOLD);
    details.setGravity(Gravity.LEFT);
    details.setInputType(Typeface.BOLD);
    //details.setText(user_title+" : "+alert_user);
    details.setText("FROM "+" : "+alert_user);
    details.setLayoutParams(nameParams);
    details.setTextSize(14);
    chat.addView(details);

    LinearLayout linearLayout = new LinearLayout( context: this);
    LinearLayout.LayoutParams layout_params = new LinearLayout.LayoutParams(
            width: 400,
            LinearLayout.LayoutParams.WRAP_CONTENT
    );
    layout_params.setMargins( left: 115, top: 5, right: 0, bottom: 40);
    linearLayout.setLayoutParams(layout_params);
    linearLayout.setOrientation(LinearLayout.VERTICAL);
    linearLayout.setBackgroundResource(R.drawable.pet_lost);
```

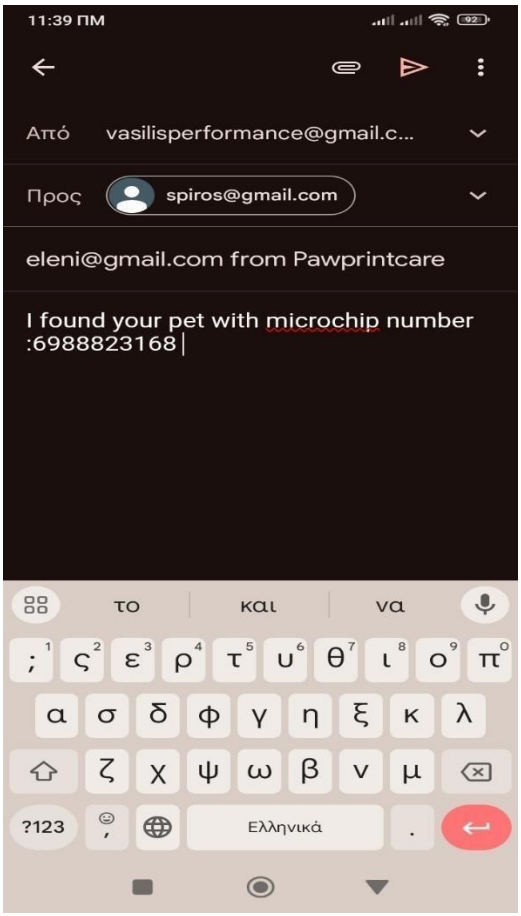
```
TextView textViewTitle = new TextView( context: this);
textViewTitle.setId(View.generateViewId());
textViewTitle.setTextColor(Color.WHITE);
textViewTitle.setGravity(Gravity.CENTER);
textViewTitle.setInputType(Typeface.BOLD);
textViewTitle.setText("NAME :"+""+name_p);
textViewTitle.setTextSize(20);
linearLayout.addView(textViewTitle);

TextView location = new TextView( context: this);
location.setId(View.generateViewId());
location.setTextColor(Color.WHITE);
location.setGravity(Gravity.CENTER);
location.setInputType(Typeface.ITALIC);
location.setText("-"+"HOME"+" : "+lost_city);
location.setTextSize(15);
location.setPadding( left: 5, top: 0, right: 5, bottom: 0);
linearLayout.addView(location);

TextView datatime = new TextView( context: this);
datatime.setId(View.generateViewId());
datatime.setTextColor(Color.WHITE);
datatime.setGravity(Gravity.CENTER);
datatime.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
datatime.setText("-"+"Last Seen :"+lost_date);
datatime.setTextSize(15);
datatime.setSingleLine(false);
datatime.setPadding( left: 5, top: 0, right: 5, bottom: 0);
datatime.setInputType(Typeface.ITALIC);
linearLayout.addView(datatime);

// Create the ImageView
ImageView imageView = new ImageView( context: this);
imageView.setId(View.generateViewId());
LinearLayout.LayoutParams nameParams3 = new LinearLayout.LayoutParams(
        width: 300,
        height: 350
);
nameParams3.gravity = Gravity.CENTER;
```



Όταν πατήσει κάποιος το κουμπί «I Found it» εμφανίζεται αυτό το menu από το οποίο επιλέγεις πως θες να μιλήσεις με αυτόν



Εάν πατήσουμε για παράδειγμα το email οδηγούμαστε εδώ



Τέλος εφόσον αυτός που έχει χάσει το κατοικίδιο το βρει μπορεί να πατήσει το ίδιο κουμπί και το Post να διαγραφεί

```
found.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(logged_in_user.equals(alert_user)){
            Log.d( tag: "to skuli sou", msg: "");
            firestore.collection( collectionPath: "losted_pets").document((String) v.getTag()) DocumentReference
                    .update( field: "is_lost",  value: false) Task<Void>
                    .addOnSuccessListener(aVoid -> Toast.makeText( context: Chat.this,  text: "Congratulations, this is wonderful news!", Toast.LENGTH_SHORT).show())
                    .addOnFailureListener(e -> Log.w( tag: "Firestore",  msg: "Error updating document", e));}
        //i have to change the bollean variable is_lost to false
        else{
            AlertDialog.Builder builder = new AlertDialog.Builder( context: Chat.this);
            builder.setTitle("Choose Communication Method")
                    .setMessage("How would you like to communicate about the lost pet?")
                    .setPositiveButton( text: "SMS", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            String tagString = (String) v.getTag();
```

Στο home page πάνω δεξιά βρίσκεται ένα menu button με το οποίο μπορεί ο χρήστης να κάνει μετάφραση την σελίδα , να αποσυνδεθεί από το PawPrintCare , να ζητήσει βοήθεια για το πώς χειρίζεται την παρών σελίδα αλλά και να ανοίξει την σελίδα με τα στατιστικά


```
switch (item.getItemId()) {
    case R.id.help:{
        LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
        View popupView = inflater.inflate(R.layout.onlinehelp,  root: null);
        // Find the Button and TextView in the inflated layout
        Button closeButton = popupView.findViewById(R.id.button6);
        TextView textView = popupView.findViewById(R.id.textView23);
        // Set up the PopupWindow
        PopupWindow popupWindow = new PopupWindow(popupView, ConstraintLayout.LayoutParams.WRAP_CONTENT, ConstraintLayout.LayoutParams.WRAP_CONTENT,  focusable
        // Set a click listener for the close button
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { popupWindow.dismiss(); }
        });
        // Optionally, you can modify the TextView or other views here
        textView.setText("Online Help"+"\n"+"\n"+"You can handle all your pet-related tasks here, includi...");
        // Show the popup window
        popupWindow.showAtLocation(view, Gravity.CENTER,  x: 0,  y: 10);
        return true;
    }
    case R.id.translate:
        showTranslateMenu(wrapper,view);
        return true;
    case R.id.statistic:{
        startActivity(new Intent( packageContext: OwnerPage.this, StatisticsActivity.class));
        return true;
    }
    case R.id.logout:{
        Intent intent = new Intent( packageContext: OwnerPage.this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
        finish();
        return true;
```

StatisticsActivity

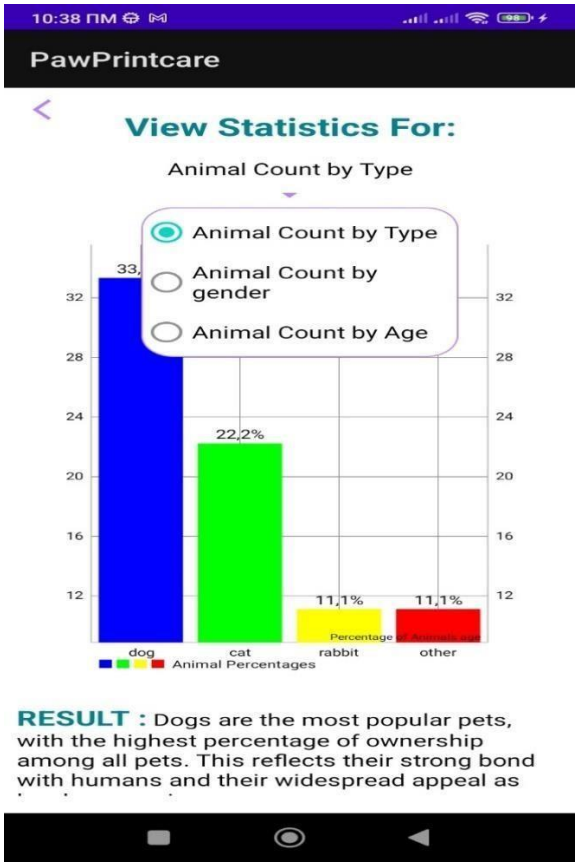
Η σελίδα statistics παρέχει διάφορα στατιστικά που αφορούν την εφαρμογή όπως :

1.Ποιός τύπος κατοικίδιου έχει το μεγαλύτερο ποσοστό , δηλαδή ποιο ζώο είναι το ζώο που έχουν οι ποιο πολύ χρήστες . Οι επιλογές που προβάλλονται στο διάγραμμα είναι σκύλος, γάτα , κουνέλι και άλλο (οτιδήποτε άλλο ζώο μπορεί να έχει κάποιος για κατοικίδιο)

2. Το φύλο του ζώου , δηλαδή εάν οι ιδιοκτήτες προτιμούν αρσενικό ή θηλυκό κατοικίδιο. Επίσης προβάλλεται και επιλογή άλλο

3 Η τελευταία επιλογή για στατιστικά είναι η προβολή ποσοστού ανάλογα με την ηλικία που έχει το κατοικίδιο . Τα ποσοστά χωρίζονται σε ηλικιακά group τα οποία είναι ηλικία <1 , 1> ηλικία <5 , 5> ηλικία <9 και ηλικία >10

Για κάθε μια από της 3 επιλογές στατιστικών εμφανίζεται το διάγραμμα που προβάλει αυτά σε μορφή στηλών αλλά και στο κάτω μέρος ποιο συμπέρασμα προκύπτει από αυτά τα αποτελέσματα



```
private void setupBarChart(double percentageOfDogs,double percentageOfCats ,double percentageOfRabb,double percentageOfOther,String type) {
    // Create list of entries for the bar chart
    List<BarEntry> entries = new ArrayList<>();
    entries.add(new BarEntry( x: 0f, (float) percentageOfDogs)); // 0f for dog
    entries.add(new BarEntry( x: 1f, (float) percentageOfCats)); // 1f for cat
    entries.add(new BarEntry( x: 2f, (float) percentageOfRabb)); // 2f for rabbit
    entries.add(new BarEntry( x: 3f, (float) percentageOfOther)); // 3f for other
    double maxPercentage = Math.max(Math.max(percentageOfDogs, percentageOfCats), Math.max(percentageOfRabb, percentageOfOther));
    String descr;
    if(type.equals("type")){
        if (maxPercentage == percentageOfDogs) {
            descr=getString(R.string.descr_dogs_max_percentage);
        } else if (maxPercentage == percentageOfCats) {
            descr=getString(R.string.descr_cats_max_percentage);
        } else if (maxPercentage == percentageOfRabb) {
            descr=getString(R.string.descr_rabbits_max_percentage);
        } else {
            descr=getString(R.string.descr_others_max_percentage);
        }
    }
    else{
        if (maxPercentage == percentageOfDogs) {
            descr=getString(R.string.descr_dogs_young_pets);
        } else if (maxPercentage == percentageOfCats) {
            descr=getString(R.string.descr_cats_2_to_5_years);
        } else if (maxPercentage == percentageOfRabb) {
            descr=getString(R.string.descr_rabbits_6_to_9_years);
        } else {
            descr=getString(R.string.descr_10_years_and_older);
        }
    }
```

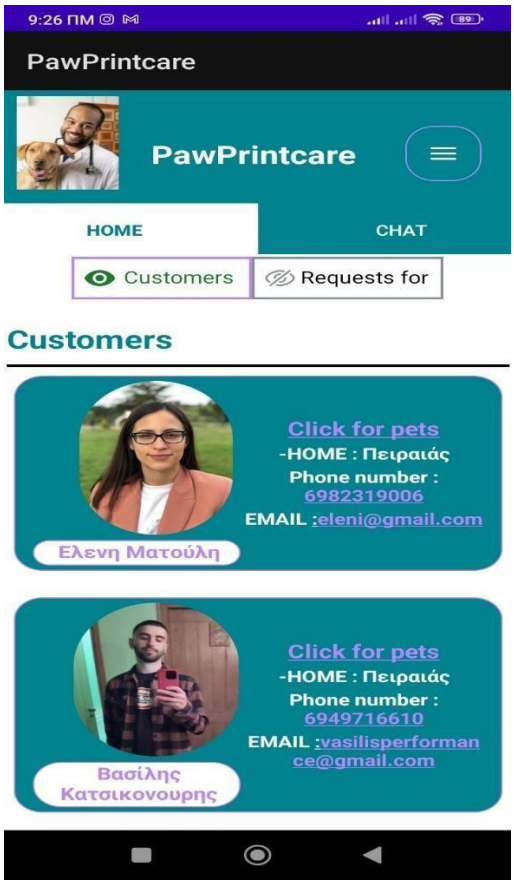
```
// Set up radio buttons and exit button
RadioGroup radioGroup = popupView.findViewById(R.id.radioGroup);
RadioButton radioButton1 = popupView.findViewById(R.id.radioButton1);
RadioButton radioButton2 = popupView.findViewById(R.id.radioButton2);
RadioButton radioButton3 = popupView.findViewById(R.id.radioButton3);
if (selectedRadioButtonId != -1) {
    radioGroup.check(selectedRadioButtonId);
}
radioButton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        statisticValue.setText("Animal Count by Type");
        popupWindow.dismiss();
        selectedRadioButtonId=R.id.radioButton1;
        getPetType();
    }
});
radioButton2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        statisticValue.setText("Animal Count by gender");
        popupWindow.dismiss();
        selectedRadioButtonId=R.id.radioButton2;
        getGender();
    }
});
radioButton3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        statisticValue.setText("Animal Count by Age");
        popupWindow.dismiss();
        selectedRadioButtonId=R.id.radioButton3;
        getAge();
```

Για να πετύχω αυτό το σχεδιάγραμμα για τα στατιστικά , έβαλα στο statistics activity barChart το οποίο του δίνεις τα δεδομένα που θες και αυτό παράγει τα σχεδιαγράμματα

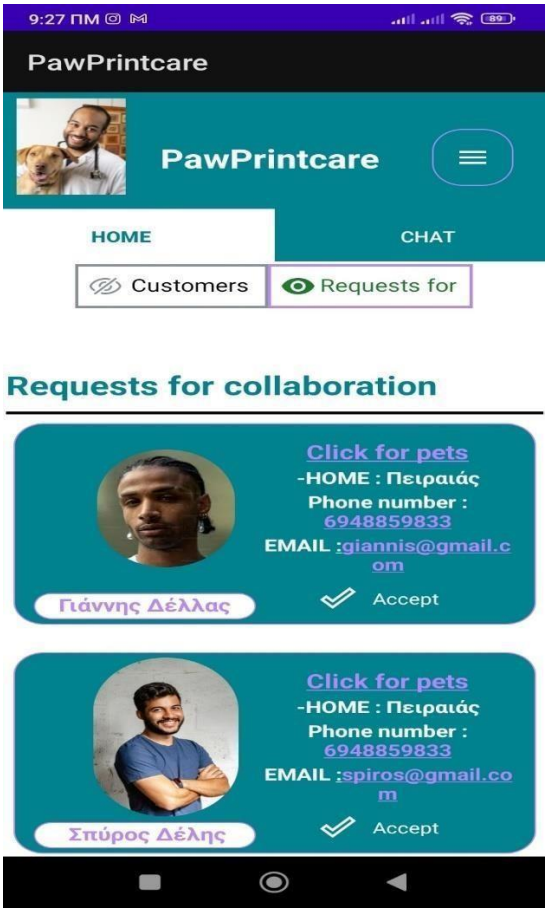
```
// Create BarData with the dataset
BarData barData = new BarData(dataSet);
barChart.setData(barData);

// Customize chart appearance
Description description = new Description();
description.setText("Percentage of Animals gender ");
barChart.setDescription(description);
```

VetpageActivity

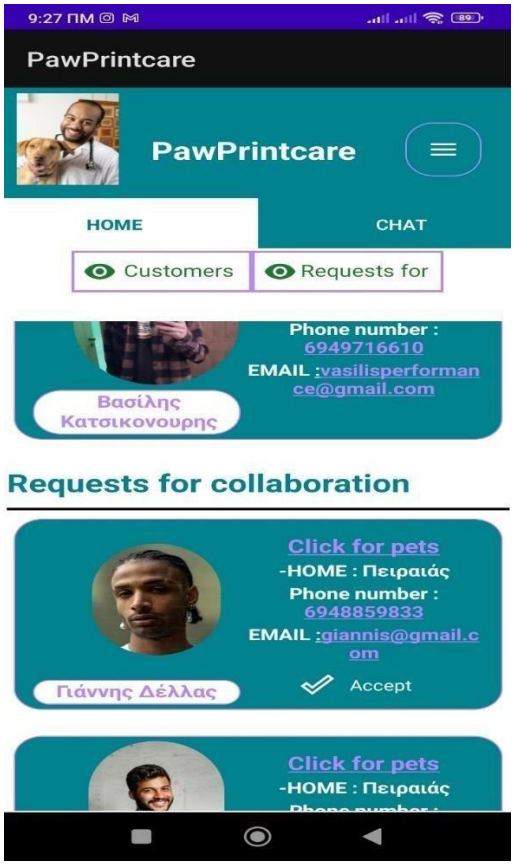
Εφόσον έχουμε βάλει ένα έγκυρο email και έναν σωστό κωδικό που να αντιστοιχεί σε ένα κτηνίατρο θα προηγηθούμε στην vet page σελίδα . Μέσα σε αυτήν υπάρχουν 2 βασικά navigation button ,όπου το ένα μας επιστρέφει στο homa page του κτηνίατρου και το άλλο ανοίγει το chat της εφαρμογής .

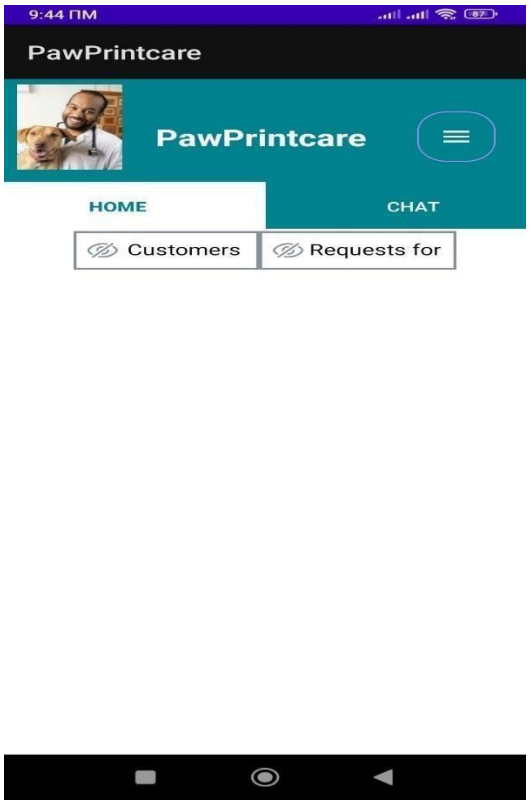


Μέσα στο homa page εμφανίζονται οι πελάτες του εκάστοτε κτηνίατρου και δύο ακόμα κουμπιά όπου το ένα εμφανίζει μόνο τους πελάτες και το άλλο εμφανίζει τα αιτήματα που έχει ο κτηνίατρος για συνεργασία .



Τα κουμπιά αυτά μπορούν αν είναι και τα δύο ενεργοποιημένα, δηλαδή να εμφανίζονται και οι πελάτες και τα αιτήματα , αλλά και τα δύο απενεργοποιημένα , δηλαδή να μην εμφανίζεται τίποτα .

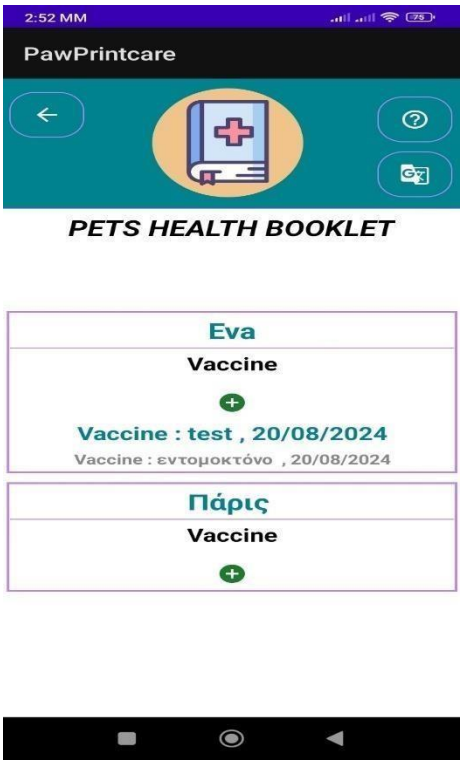


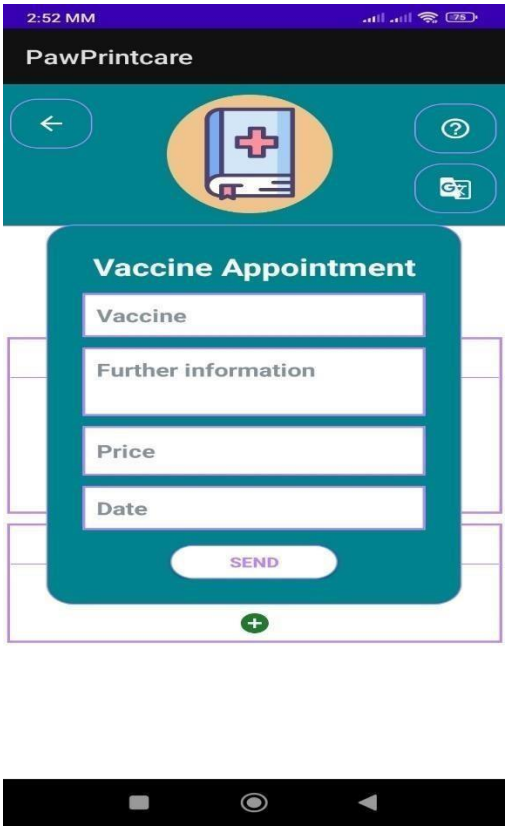


Εδώ είναι ο κώδικας για το πως έκανα την λειτουργία με τα κουμπιά που περιγράφεται παραπάνω

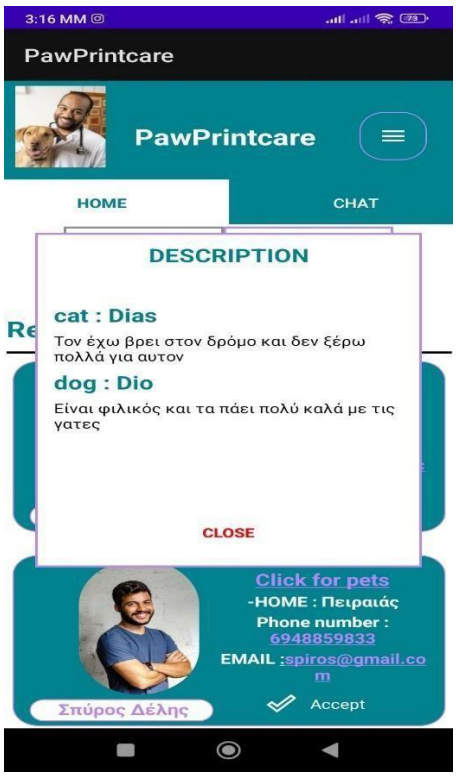
```
            countRequest=0;
        }
        // Handle document existence as needed
        if(onlyCustomers && onlyRequest){
            for (DocumentSnapshot document : officialTrueDocs){
                createCustomerCard(document.getString( field: "owner"),document.getString( field: "official"));
                Log.d( tag: "MainActivity",  msg: "Document with email " + document.getString( field: "official") + " exists.");
            }
            for (DocumentSnapshot document : officialFalseDocs){
                createCustomerCard(document.getString( field: "owner"),document.getString( field: "official"));
                Log.d( tag: "MainActivity",  msg: "Document with email " + document.getString( field: "official") + " exists.");
            }
        }
        else if(onlyCustomers){
            for (DocumentSnapshot document : officialTrueDocs){
            if(document.getString( field: "official").equals("true")){
                createCustomerCard(document.getString( field: "owner"),document.getString( field: "official"));
                Log.d( tag: "MainActivity",  msg: "Document with email " + document.getString( field: "official") + " exists.");
            }
          }
        }
        else if(onlyRequest){
            for (DocumentSnapshot document : officialFalseDocs){
            if(document.getString( field: "official").equals("false")){
```

Κάνοντας click στο «click for pets» που υπάρχει πάνω στην κάρτα του κάθε πελάτη ανοίγει το health booklet αυτού που περιέχει τα ζώα του και από εκεί πατώντας το σήμα + μπορεί ο κτηνίατρος να προσθέσει ένα εμβόλιο το οποίο πρέπει να γίνει σε κάποιο κατοικίδιο





Εάν όμως πατήσουμε το «click for pets »σε κάποιον που δεν είναι αι πελάτης τότε θα μας εμφανιστεί ένα παράθυρο που θα μας λέει πόσα κατοικίδια έχει ο συγκεκριμένος αλλά και μια περιγραφή για το κάθε ένα . Αυτή η λειτουργία έγινε έτσι ώστε κάποιος κτηνίατρος να μπορεί να ελέγξει τα ζώα που έχει κάποιος προτού τον κάνει πελάτη του



Μελλοντικές επεκτάσεις της εφαρμογής

Η τρέχουσα έκδοση της εφαρμογής προσφέρει ισχυρή λειτουργικότητα για την εύρεση κτηνιάτρων, τη διαχείριση της υγείας των κατοικίδιων και την αλληλεπίδραση με μια κοινότητα ιδιοκτητών κατοικίδιων. Ωστόσο, έχουν προγραμματιστεί αρκετές συναρπαστικές επεκτάσεις για το μέλλον, οι οποίες θα βελτιώσουν την εμπειρία του χρήστη και θα διευρύνουν τις δυνατότητες της πλατφόρμας.

Ένας από τους πρωταρχικούς τομείς επέκτασης θα είναι η προσθήκη προσωπικού chat. Οι χρήστες θα μπορούν να επικοινωνούν απευθείας μεταξύ τους για ανταλλαγή πληροφοριών και συμβουλών, αλλά και να συνομιλούν με τους κτηνιάτρους τους για να λαμβάνουν άμεσες συμβουλές. Επιπλέον, η δυνατότητα άμεσης επικοινωνίας μεταξύ κτηνιάτρων και πελατών θα διευκολύνει την καλύτερη παρακολούθηση της υγείας των κατοικίδιων.

Η δυνατότητα κρατήσεων ραντεβού θα ενσωματωθεί πλήρως στην πλατφόρμα, επιτρέποντας στους χρήστες να κλείνουν εύκολα ραντεβού με τον κτηνίατρό τους χωρίς την ανάγκη εξωτερικών εργαλείων(email). Οι χρήστες θα μπορούν να βλέπουν τις διαθέσιμες ώρες των κτηνιάτρων, να επιλέγουν την κατάλληλη για αυτούς και να διαχειρίζονται τις επισκέψεις τους μέσα από την εφαρμογή.

Η ενημέρωση του "health booklet" των κατοικίδιων θα παραμείνει κεντρική λειτουργία, ενώ σχεδιάζονται προσθήκες όπως υπενθυμίσεις για εμβολιασμούς, θεραπείες και άλλες ιατρικές ενέργειες. Αυτές οι υπενθυμίσεις θα βοηθήσουν τους ιδιοκτήτες να διασφαλίσουν την καλύτερη φροντίδα για τα κατοικίδια τους, σε συνεργασία με τους κτηνιάτρους.

Η διεπαφή χρήστη θα υποβάλλεται σε συνεχείς βελτιώσεις για να εξασφαλίσει ευκολία στη χρήση και καλύτερη εμπειρία πλοήγησης. Σχεδιάζεται η εισαγωγή δυνατοτήτων για εξατομίκευση της εμφάνισης της εφαρμογής, προσφέροντας στους χρήστες τη δυνατότητα να προσαρμόζουν τη διεπαφή στις προτιμήσεις τους.

Για να διευρυνθεί το κοινό της εφαρμογής, θα προστεθεί υποστήριξη πολλαπλών γλωσσών. Αυτό θα επιτρέψει στους χρήστες από διαφορετικές χώρες να χρησιμοποιούν την εφαρμογή στη γλώσσα της προτίμησής τους, καθιστώντας την πλατφόρμα προσιτή σε παγκόσμιο επίπεδο.

Με την ενσωμάτωση αυτών των επεκτάσεων, η εφαρμογή θα συνεχίσει να εξελίσσεται, προσφέροντας μεγαλύτερη αξία και μια πιο ολοκληρωμένη εμπειρία στους χρήστες της, διατηρώντας την ως ένα κορυφαίο εργαλείο για τη φροντίδα των κατοικίδιων και την επικοινωνία με τους κτηνιάτρους.

Συμπεράσματα

Η παρούσα εφαρμογή αποτελεί μια ολοκληρωμένη λύση για τους κατόχους κατοικίδιων οι οποίοι επιθυμούν να βρουν με πολύ γρήγορο και εύκολο τρόπο έναν

κτηνίατρο σε κοντινή απόσταση από αυτούς . Παράλληλα, τους παρέχει τη δυνατότητα να ανταλλάσσουν απόψεις με άλλους χρήστες, προσφέροντας λύσεις σε θέματα που αφορούν τη φροντίδα των κατοικίδιων τους

Με την φιλική προς τον χρήστη διεπαφή και την άμεση πρόσβαση σε έναν OnLine βοηθό, ο οποίος επεξηγεί κάθε σελίδα, οι χρήστες μπορούν να διαχειρίζονται αποτελεσματικά την υγεία και ευεξία των κατοικίδιων τους.

Η εφαρμογή δίνει στους κατόχους κατοικίδιων τη δυνατότητα να εισάγουν τα κατοικίδιά τους, προσθέτοντας όλες τις απαραίτητες πληροφορίες. Μέσω εύχρηστων φορμών, κάθε χρήστης μπορεί να αναζητήσει έναν κτηνίατρο στην περιοχή του και, αν τον ικανοποιούν τα προσόντα του, να στείλει αίτημα για συνεργασία. Επιπλέον, μέσω της λειτουργίας chat, οι χρήστες μπορούν να δημοσιεύσουν ανακοινώσεις για την εξαφάνιση του κατοικίδιου τους, ώστε να ενημερωθούν όσο το δυνατόν περισσότεροι άνθρωποι και να επιταχυνθεί η διαδικασία ανεύρεσης.

Η εφαρμογή προσφέρει στους κτηνιάτρους τη δυνατότητα να βρουν εύκολα νέους πελάτες. Το μόνο που χρειάζεται να κάνουν είναι να καταχωρίσουν τα στοιχεία τους, καθώς και μια σύντομη περιγραφή που να περιλαμβάνει τις σπουδές τους και την επαγγελματική τους εμπειρία στον κλάδο. Οι κτηνίατροι μπορούν επίσης να διαχειρίζονται το "health booklet" κάθε ζώου που αναλαμβάνουν, προσθέτοντας πληροφορίες για εμβολιασμούς και άλλες απαραίτητες ιατρικές ενέργειες. Στη συνέχεια, ο ιδιοκτήτης μπορεί να αποδεχτεί ή να απορρίψει τις προτάσεις αυτές, διατηρώντας τον έλεγχο της υγείας του κατοικίδιου.

Εν κατακλείδι, η εφαρμογή αποτελεί μια δυναμική και συνεχώς εξελισσόμενη πλατφόρμα, σχεδιασμένη να καλύπτει τις ποικίλες ανάγκες της κοινότητας των ιδιοκτητών κατοικίδιων. Τα τρέχοντα χαρακτηριστικά της προσφέρουν μια ισχυρή βάση για την εύκολη εύρεση κτηνιάτρων και τη διαχείριση της υγείας των κατοικίδιων, ενώ οι μελλοντικές επεκτάσεις υπόσχονται να ενισχύσουν ακόμη περισσότερο την αξία και τη χρηστικότητά της. Με τη συνεχή προσαρμογή στις ανάγκες των χρηστών και την ενσωμάτωση των τελευταίων τεχνολογιών, η εφαρμογή είναι έτοιμη να παραμείνει ένα κορυφαίο εργαλείο για την ευζωία των κατοικίδιων και την υποστήριξη των ιδιοκτητών τους.

Βιβλιογραφία

1. https://chatgpt.com/c/7c157ccc-faad-46d7-af4a-a786519a45d3
2. https://developer.android.com/studio/write/firebase
3. https://www.geeksforgeeks.org/adding-firebase-to-android-app/

4. https://developer.android.com/reference/android/text/SpannableStri ng
5. https://stackoverflow.com/questions/71006050/androidspannablestringwit h-correctly-formated-numbers
6. https://chatgpt.com/c/bc0610e0-ba0f-4dd1-bf21-61dd78787649
7. https://chatgpt.com/c/927306f5-69d8-42f0-862a-46f4407ea1ca
8. https://stackoverflow.com/questions/73927631/upload-image-andvideoon-firebase-storage-and-save-url-in-firestore-using-java
9. https://stackoverflow.com/questions/15208408/null-pointerexceptionwhile-implementing-touch-listener-in-android
10. https://medium.com/@abhinavv.singh/integrating-google-mapsinandroid-studio-using-java-a-step-by-step-guide-with-code5ba857dff344
11. https://mailtrap.io/blog/android-intent-send-email/
12. https://developer.android.com/develop/connectivity/telecom/selfMa naged
13. https://github.com/search?q=firebase+android+java&type=repositori es
14. Head First Android Development: A Brain-Friendly Guide" - by Dawn Griffiths and David Griffiths
15. Head First Java, 2nd Edition by Kathy Sierra and Bert Bates
16. Android Programming with Kotlin for Beginners: Build Android Apps by John Horton
17. Firebase Essentials - Android Edition by Mark Wickham https://www.empik.com/practical-androidwickhammark,p1169549663,ksiazka-p

Επιστημονικά άρθρα

1. ReckDroid: Detecting red packet fraud in Android apps https://www.sciencedirect.com/science/article/abs/pii/S016740482400422X?via%3Dihub

2. Runtime and Design Time Completeness Checking of Dangerous Android App Permissions Against GDPR https://ieeexplore.ieee.org/document/10373786

3. FIREBASE -OVERVIEW AND USAGE https://www.researchgate.net/publication/362539877_FIREBASE_OVERVIEW_AND_USAGE

4. A set of Role-Based Mobile Applications for Automatic Attendance Checking with UHF RFID Using Realtime Firebase and Face Recognition https://link.springer.com/chapter/10.1007/978-981-19-8069-5_29

5. Approximate XML structure validation based on document–grammar tree similarity
https://www.sciencedirect.com/science/article/abs/pii/S0020025514009 566

6. Learning Java https://link.springer.com/book/10.1007/978-3-031-66638-4

7. Connecting Communities: An Android Social Networking Application with Firebase & Java
https://www.researchgate.net/publication/381190663_Connecting_Communities_An_Android_Social_Networking_Application_with_Firebase_Jav a

8. Explore Java Language and Android Mobile Software Development
https://francis-press.com/papers/3678

9. The Java location API
https://www.researchgate.net/publication/292444078_The_Java_location_API