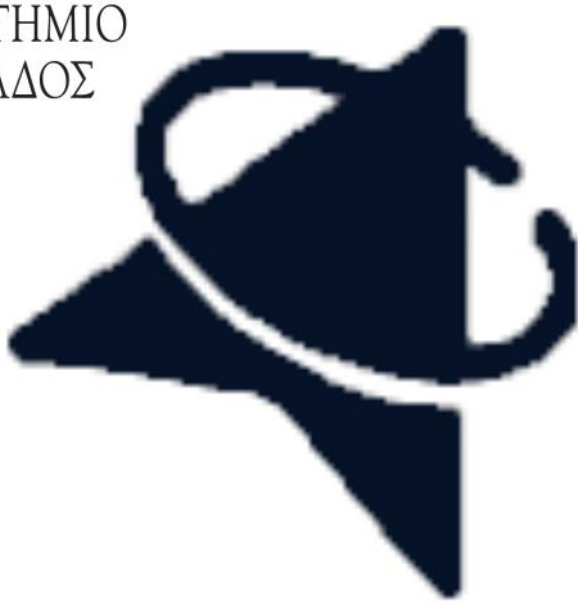
ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ



ΔΙΑΡΚΕΙΑ: **1/11-17/1**

ΜΑΘΗΜΑ: **ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ**

Μέλη Ομάδας:

Ντούλο Γιώργος 4680
Γκαρακλίδης Χρήστος 4669
Ιγνατιάδης Μιχάλης 4522
Λιόγκας Θάνος 4313
Βαρκιανός Βασίλης 4729

GitHub: https://github.com/XGkara/Tex_Logismikou

Στόχος Project:

Το project το οποίο μας ανατέθηκε είναι η υλοποίηση μιας πλήρως λειτουργικής ταξιδιωτικής εφαρμογής, χρησιμοποιώντας Android Studio, σε γλώσσα προγραμματισμού Java. Αυτή η εφαρμογή μπορεί να βρει την ακριβή γεωγραφική τοποθεσία του χρήστη και να εμφανίζει αξιοθέατα, πληροφορίες, καθώς και οδηγίες προσέγγισης σε αυτά.

Project Plan:

1. Εύρεση Τοποθεσίας
2. Εμφάνιση θέσης στο χάρτη
3. Εύρεση αξιοθέατα
4. Τοποθέτηση αξιοθέατα στο χάρτη
5. Υπολογισμός Απόστασης
6. Οδηγίες Προσέγγισης
7. Πληροφορίες για το αξιοθέατο

Εργαλεία:

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον για ανάπτυξη εφαρμογών στην πλατφόρμα Android.

Το Google Firebase είναι ένα σύνολο εργαλείων ανάπτυξης που βασίζονται σε cloud βοηθά τους προγραμματιστές εφαρμογών για κινητά να δημιουργήσουν, να αναπτύξουν και να κλιμακώσουν τις εφαρμογές τους.

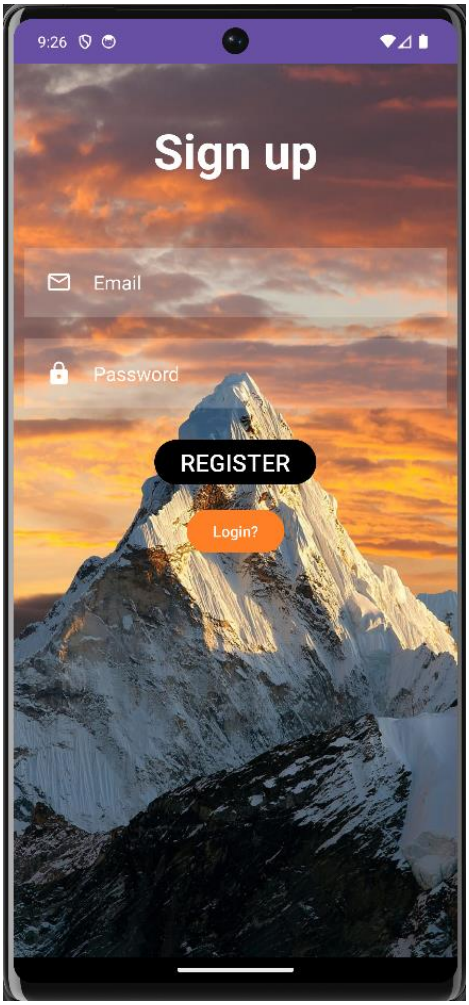
Το **GitHub,** είναι μια πλατφόρμα προγραμματιστών που υποστηρίζεται από AI που επιτρέπει στους προγραμματιστές να δημιουργούν, να αποθηκεύουν και να διαχειρίζονται τον κώδικά τους.

Το GitHub Desktop είναι μια δωρεάν εφαρμογή ανοιχτού κώδικα που σας βοηθά να εργάζεστε με κώδικα που φιλοξενείται στο **GitHub** ή σε άλλες υπηρεσίες φιλοξενίας Git.

Το Application

- **Sign Up:**

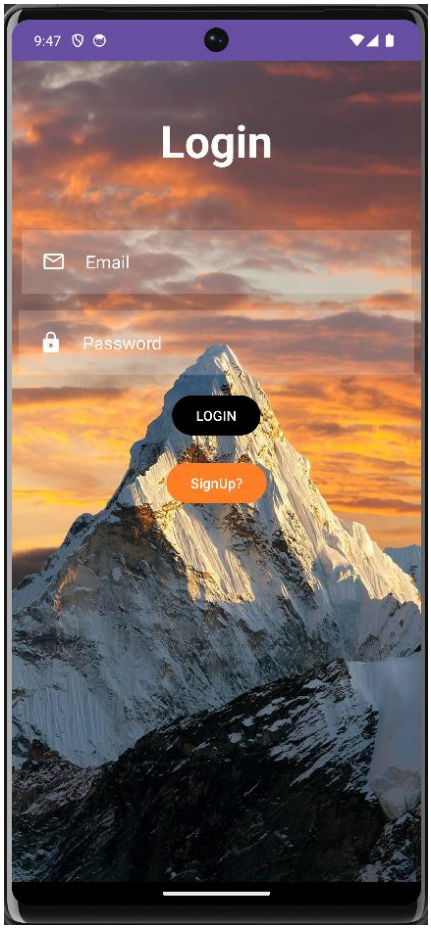
Το πρώτο πράγμα που συναντά ο χρήστης ανοίγοντας την εφαρμογή μας είναι το πεδίο **Sign Up**, όπου τον προτρέπει να δημιουργήσει έναν λογαριασμό ώστε να μπορέσει να την χρησιμοποιήσει. Στην φόρμα Sign up, υπάρχουν δύο πεδία κειμένου στα οποία ζητείται από τον χρήστη να εισάγει τα στοιχεία όπου θα χρησιμοποιεί για την είσοδό του στην εφαρμογή και δύο κουμπιά όπου με το ένα κάνει εγγραφή και μεταφέρεται στην εφαρμογή , ενώ , το άλλο προωθεί τον χρήστη στη φόρμα Login. Η μορφή του δημιουργήθηκε μέσω ενός αρχείο τύπου .xml οπού κάναμε τις απαραίτητες τροποποιήσεις του κώδικα για να έχουμε το επιθυμητό αποτέλεσμα. Επιπλέον, κάθε αντικείμενο που δημιουργούμε σε αυτά τα αρχεία δίνεται και ένα id, για να μπορούμε να τα καλέσουμε ή να τα αναφέρουμε στις κλάσεις .Java για να τροποποιήσουμε τις λειτουργίες τους.



- **Login:**

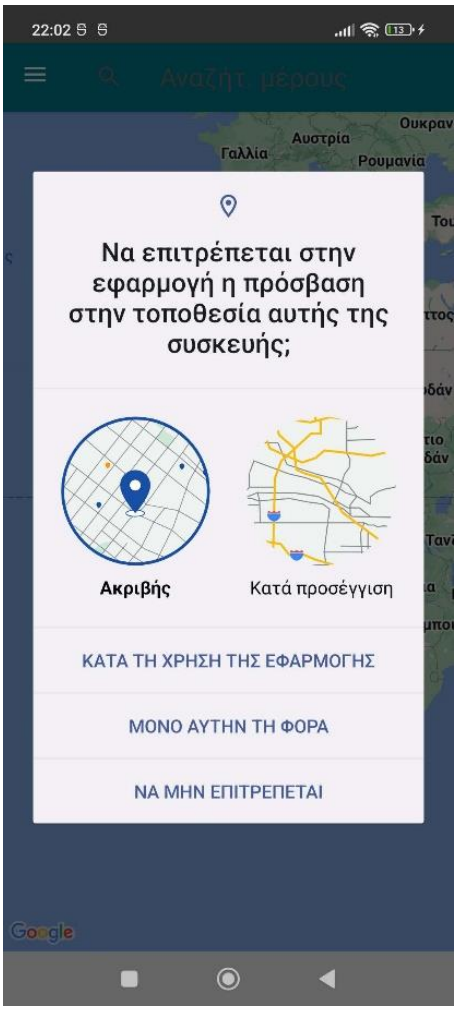
Μετά από κάθε Sign Up υπάρχει και ένα Login.

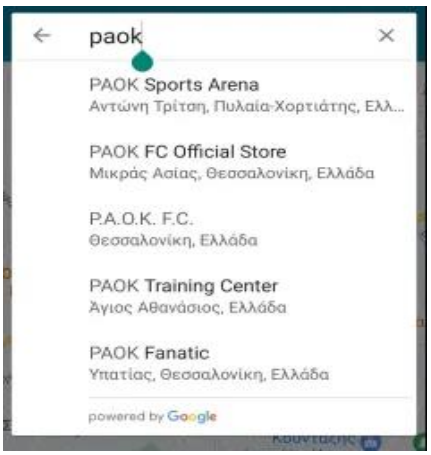
Η φόρμα Login, διαθέτει δύο πεδία κειμένου, για το “username” και το “password” αντίστοιχα, όπως και δύο κουμπιά «LOGIN» και «Sign up?». Το «LOGIN» ελέγχει αν τα στοιχεία είναι σωστά και ανοίγει την εφαρμογή, διαφορετικά, εμφανίζει μήνυμα σφάλματος. Το «Sign up?», μας μεταφέρει πίσω στην αρχική φόρμα Sign Up. Η μορφή του δημιουργήθηκε ,επίσης, μέσω ενός αρχείο τύπου .xml οπού κάναμε τις απαραίτητες τροποποιήσεις του κώδικα για να έχουμε το επιθυμητό αποτέλεσμα.

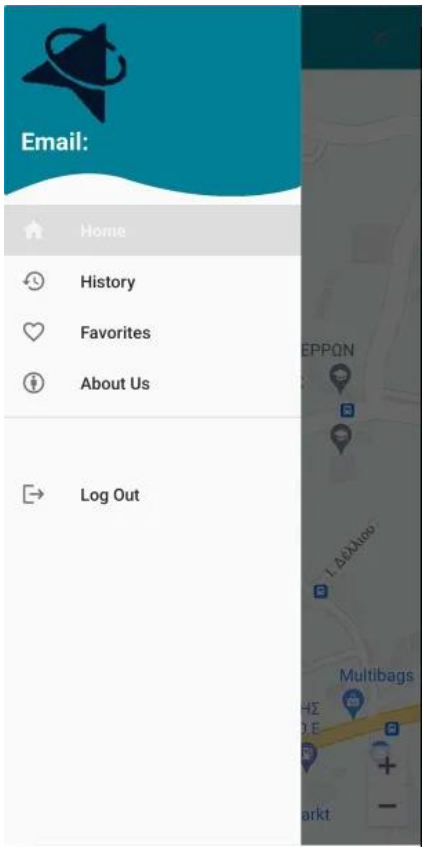


- **Home Page:**

Το πρώτο πράγμα που συμβαίνει με την είσοδο του χρήστη στην εφαρμογή, είναι να ζητάει άδεια χρήσης της τοποθεσίας του. Εφόσον δοθεί η άδεια, εμφανίζεται η ακριβής τοποθεσία του χρήστη στο χάρτη με μία κόκκινη πινέζα και όλων των «καταστημάτων» γύρω του. Εάν θέλει να αναζητήσει κάτι συγκεκριμένο, υπάρχει ένα Search Bar στο πάνω μέρος της οθόνης με autocomplete Assistance.



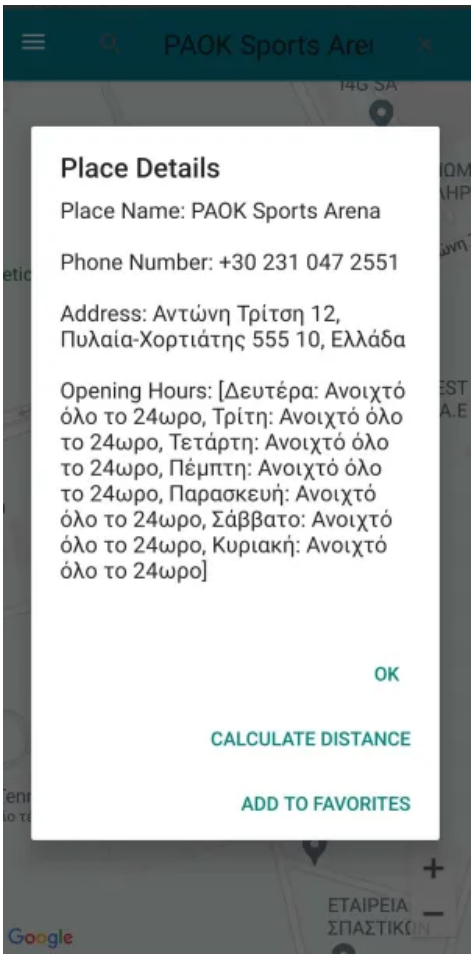




Επίσης, πάνω αριστερά υπάρχει ένα κουμπί  , όπου ανοίγει το Navigation View, για την ευκολότερη μετακίνηση του χρήστη εντός της εφαρμογής. Όπως και το κουμπί “Logout”, όπου αποσυνδέει τον χρήστη και τον μεταφέρει πίσω στη φόρμα Login.

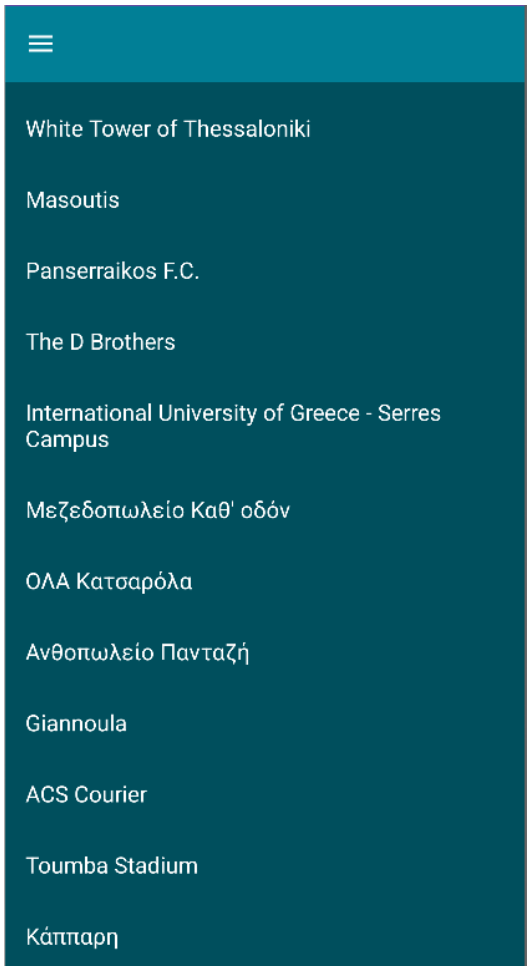
- **Search Bar:**

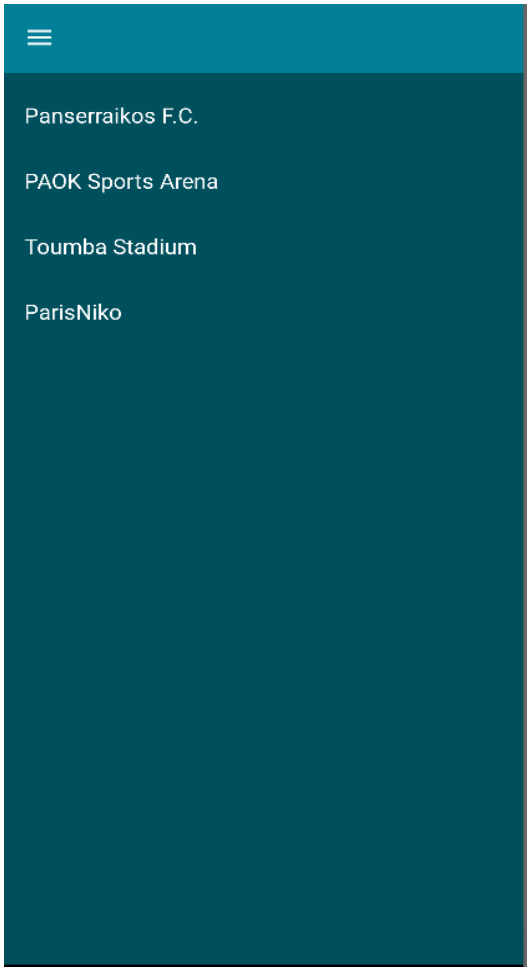
  Μετά την αναζήτηση μιας τοποθεσίας η εφαρμογή εμφανίζει πληροφορίες. Όταν δεν τις χρειάζεται πλέον πατάει το OK και εμφανίζεται η ακριβής τοποθεσία στο χάρτη και με την επιλογή Calculate Distance υπολογίζει την απόσταση από την τοποθεσία που βρίσκεται ο χρήστης.



- **History:** Το History εμφανίζει στον χρήστη τις τελευταίες του αναζητήσεις.

- **Favorites:** Το Favorites εμφανίζει στον χρήστη τις τοποθεσίες τις οποίες έχει επιλέξει ως αγαπημένες του.





About Us:

Το About Us είναι μία σελίδα όπου λέει πως η εφαρμογή μας είναι project, δείχνει τα μέλη της ομάδας μας και διαθέτει ένα Image Button όπου μας μεταφέρει στο link του GitHub repository μας.

- **History:**
  Δυστυχώς δεν καταφέραμε να υλοποιήσουμε τις λειτουργείες αυτών των δύο σελίδων.

ΚΩΔΙΚΑΣ

- **Sign Up:** Το .xml αρχείο της Sign Up περιέχει ένα TextView το οποίο χρησιμοποιούμε ως τίτλο στο πάνω μέρος της οθόνης, δύο EditText και δύο Buttons τα οποία μας βοηθάνε στο να πάρουμε τα στοιχεία του χρήστη και να τον μεταφέρουμε στις υπόλοιπες δραστηριότητες της εφαρμογής μας.

```
<TextView
    android:id="@+id/signuptitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="50dp"
    android:layout_marginTop="50dp"
    android:layout_marginEnd="50dp"
    android:layout_marginBottom="50dp"
    android:gravity="center"
    android:text="Sign up"
    android:textColor="@color/white"
    android:textSize="45sp"
    android:textStyle="bold" />



<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/email"
    android:layout_below="@id/signuptitle"
    android:background="#30ffffff"
    android:hint="Email"
    android:textColorHint="@color/white"
    android:textColor="@color/white"
    android:layout_margin="10dp"
    android:padding="20dp"
    android:drawableLeft="@drawable/baseline_mail_outline_24"
    android:drawablePadding="20dp"/>
```

Στην κλάση Signup.java καλούμε και δηλώνουμε ότι φτιάξαμε στο αντίστοιχο .xml αρχείο και τα δίνουμε κάποιες λειτουργείες για να επιτύχουμε το επιθυμητό μας αποτέλεσμα. Αναλυτικότερα πατώντας το κουμπί Login?, μεταφέρουμε τον χρήστη κατευθείαν στην δραστηριότητα Login για τις περιπτώσεις όπου ο χρήστης έχει ήδη δημιουργήσει λογαριασμό στο παρελθόν. Αλλιώς παίρνουμε τα στοιχεία που βάζει στα δύο πλαίσια και αυτά αποθηκεύονται αυτομάτως στην Firebase Firestore, δηλαδή την βάση δεδομένων μας, εφόσον τα στοιχεία είναι έγκυρα με το κουμπί REGISTER.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    setContentView(R.layout.activity_signup);
    mAuth= FirebaseAuth.getInstance();
    editTextEmail = findViewById(R.id.email);
    editTextPassword = findViewById(R.id.password);
    buttonSignUp = findViewById(R.id.signupbtn);
    progressBar = findViewById(R.id.progressBar);
    buttonqLogIn = findViewById(R.id.qLogIn);
    Lowzy2k +2 *
    buttonqLogIn.setOnClickListener(new View.OnClickListener() {
        Lowzy2k +2 *
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(getApplicationContext(),Login.class);
            startActivity(intent);
            finish();
        }
    });
```

```
buttonSignUp.setOnClickListener(new View.OnClickListener() {
    Lowzy2k +1 *
    @Override
    public void onClick(View v) {
        progressBar.setVisibility(View.VISIBLE);
        passwords = String.valueOf(editTextPassword.getText());
        emails = String.valueOf(editTextEmail.getText());

        Map<String, Object> user = new HashMap<>();
        user.put("email", emails);
        user.put("password", passwords);


        if (TextUtils.isEmpty(emails)){
            Toast.makeText( context: Signup.this, text: "Enter email", Toast.LENGTH_SHORT).show();
            return;
        }

        if (TextUtils.isEmpty(passwords)){
            Toast.makeText( context: Signup.this, text: "Enter password", Toast.LENGTH_SHORT).show();
            return;
        }
```

```
mAuth.createUserWithEmailAndPassword(emails, passwords)

        Lowzy2k +1 *
        .addOnCompleteListener( new OnCompleteListener<AuthResult>() {

            Lowzy2k +1
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                progressBar.setVisibility(View.GONE);

                db.collection( collectionPath: "users") CollectionReference
                        .add(user) Task<DocumentReference>
                        Lowzy2k
                        .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                            Lowzy2k
                            @Override
                            public void onSuccess(DocumentReference documentReference) {
                                Log.d(MotionEffect.TAG, msg: "DocumentSnapshot added with ID: " + documentReference.getId());
                            }
                        })
                        Lowzy2k
                        .addOnFailureListener(new OnFailureListener() {
                            Lowzy2k
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Log.w(MotionEffect.TAG, msg: "Error adding document", e);
                            }
                        });
                if (task.isSuccessful()) {
                    Toast.makeText( context: Signup.this, text: "Account created.",
                            Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(),Login.class);
                    startActivity(intent);
                    finish();
```

- **Login:** Το .xml αρχείο της Login είναι ίδιο με αυτό του Sign Up. Οι λειτουργείες της Login.java διαφέρουν ως προς την λογική τους από την αντίστοιχη SignUp.java, καθώς η μία λαμβάνει και αποθηκεύει ενώ η άλλη διαβάζει τα στοιχεία και κάνει έλεγχο εγκυρότητας ώστε να μπορέσει να συνδεθεί ο χρήστης με τα σωστά στοιχεία.

```
buttonLogIn.setOnClickListener(new View.OnClickListener() {
    Lowzy2k +1 *
    @Override
    public void onClick(View v) {
        progressBar.setVisibility(View.VISIBLE);
        String email, password;
        password = String.valueOf(editTextPassword.getText());
        email = String.valueOf(editTextEmail.getText());

        if (TextUtils.isEmpty(email)){
            Toast.makeText( context: Login.this, text: "Enter email", Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(password)){
            Toast.makeText( context: Login.this, text: "Enter password", Toast.LENGTH_SHORT).show();
            return;
        }
        mAuth.signInWithEmailAndPassword(email, password)
                Lowzy2k +1 *
                .addOnCompleteListener( new OnCompleteListener<AuthResult>() {
                    Lowzy2k +1 *
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        progressBar.setVisibility(View.GONE);
                        if (task.isSuccessful()) {
                            Toast.makeText(getApplicationContext(), text: "Login Successful",Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                            startActivity(intent);
                            finish();
                        } else {
                            Toast.makeText( context: Login.this,  text: "Authentication failed.",
                                    Toast.LENGTH_SHORT).show();
                        }
                    }
                });
```

- **Main Activity:** Το .xml αρχείο της MainActivity αποτελείτε από τα εξής κομμάτια:

1. Το MapFragment, στην ουσία ο χάρτης μας όπου όλες οι λειτουργείες του app αναδεικνύονται πάνω του.

2. Το autoCompleteFragment, όπου είναι ένα SearchBar με αυτόματη συμπλήρωση λέξεων και μαντεύοντας βοηθάει την εμπειρία του χρήστη αλλά και την ορθότητα της αναζήτησης, που βεβαιώνει την αποφυγή λαθών.

3. Ένα NavigationViewer, όπου ανοίγει πατώντας στο κουμπί πάνω αριστερά στην οθόνη. Από το οποίο θα μπορούσαμε να περιηγηθούμε στην εφαρμογή με μεγαλύτερη άνεση και φυσικά να αποσυνδεθούμε.

4. Τέλος, υπάρχει ένα Toolbar στο πάνω μέρος την οθόνης όπου πλαισιώνεται από το κουμπί,του NavigationViewer και του SearchBar «autoCompleteFragment».

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/autocomplete_fragment"
    android:name="com.google.android.libraries.places.widget.AutocompleteSupportFragment"
    android:layout_width="318dp"
    android:layout_height="38dp"
    android:layout_margin="10dp"
    android:layout_marginTop="5dp"
    android:elevation="5dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<androidx.appcompat.widget.Toolbar
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:id="@+id/toolbar"
    android:elevation="4dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:background="@color/blue"/>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/fragment_container">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
         />
</FrameLayout>
```

```
<com.google.android.material.navigation.NavigationView
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:id="@+id/nav_view"
    android:layout_gravity="start"
    app:headerLayout="@layout/nav_header"
    app:menu="@menu/nav_menu"/>
```

Στη κλάση MainActivity.java ξεκινάμε κάνοντας import όλες τις βιβλιοθήκες όπου μας χρειάστηκαν με βασικότερες αυτές τις Google και της Firebase. Στη συνέχεια, κάνουμε δήλωση μεταβλητών και ξεκινάμε με τη βασικότερη μέθοδο του application, την onCreate η οποία καλεί και δημιουργεί όλα τα στοιχεία της activity_main.xml.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Places.initialize(getApplicationContext(), apiKey: "AIzaSyAkN5S8_mhBiljsTKC7LuvT_eCt1Z8DQFI");
    AutocompleteSupportFragment autocompleteFragment = (AutocompleteSupportFragment)
            getSupportFragmentManager().findFragmentById(R.id.autocomplete_fragment);

    autocompleteFragment.setLocationBias(RectangularBounds.newInstance(
            new LatLng( latitude: 41.07670157862302,  longitude: 23.554400400271827),
            new LatLng( latitude: 41.091226420839696,  longitude: 23.54935511484131)));
    autocompleteFragment.setCountries("GR");

    autocompleteFragment.setPlaceFields(Arrays.asList(Place.Field.ID, Place.Field.NAME, Place.Field.LAT_LNG, Place.Field.OPENING_HOURS, Place.Field.WEBSITE_URI
    auth = FirebaseAuth.getInstance();
    user = FirebaseAuth.getInstance().getCurrentUser();

    reference = FirebaseDatabase.getInstance().getReference();
    userID = user.getUid();
    if (user == null) {
        Intent intent = new Intent(getApplicationContext(), Login.class);
        startActivity(intent);
        finish();
    }

    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    drawerLayout = findViewById(R.id.drawer_layout);
    NavigationView navigationView = findViewById(R.id.nav_view);
```

Όπως είπαμε παραπάνω η εφαρμογή μας έχει την δυνατότητα παρουσίασης της φυσικής θέσης του χρήστη πάνω στο χάρτη, αυτό συμβαίνει στην onCreate με την βοήθεια της κλάσης LocationHelper.java όπου καλείτε εντός της onCreate.
Επίσης, η λειτουργεία εμφάνισης μίας τοποθεσίας στο χάρτη και ανάδειξη των κατάλληλων πληροφοριών γίνεται στην onCreate με τη μέθοδο onPlaceSelected όπου έχουμε επιλέξει τις ακριβείς πληροφορίες που χρειαζόμαστε. Να σημειώσουμε ότι εντός της onPlaceSelected καλείτε η μέθοδος showPlaceDetailsDialog,

όπου εμφανίζει το χαρακτηριστικό pop με τις πληροφορίες και την επιλογή της δρομολόγησης προς την τοποθεσία που επέλεξε. Η απόσταση υπολογίζεται μέσω της calculateDistance και μέσω της showModeSelectionDialog επιτρέπει στον χρήστη να έχει την επιλογή του να κατευθυνθεί με αυτοκίνητο ή με τα πόδια.

```
autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {

    no usages
    List<Address> addressList = null;

    no usages  Lowzy2k +3 *
    @Override
    public void onPlaceSelected(@NonNull Place place) {
        StringBuilder placeInfo = new StringBuilder();
        placeInfo.append("Place Name: ").append(place.getName()).append("\n\n");

        if (place.getLatLng() != null) {

            myMap.moveCamera(CameraUpdateFactory.newLatLngZoom(place.getLatLng(),  zoom: 17.0f));

            myMap.addMarker(new MarkerOptions().position(place.getLatLng()).title(place.getName()));

            addPlaceToFirestore(place);

        } else {
            placeInfo.append("Place LatLng is null").append("\n");

        }

        if (place.getPhoneNumber() != null) {
            placeInfo.append("Phone Number: ").append(place.getPhoneNumber()).append("\n\n");
        }

        if (place.getAddress() != null) {
            placeInfo.append("Address: ").append(place.getAddress()).append("\n\n");
        }


        if (place.getOpeningHours() != null) {
            placeInfo.append("Opening Hours: ").append(place.getOpeningHours().getWeekdayText()).append("\n\n");
        }

        showPlaceDetailsDialog(placeInfo.toString(), place);
    }
```

```
private void showPlaceDetailsDialog(String placeDetails, Place place) {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Place Details")
            .setMessage(placeDetails)
            .setPositiveButton( text: "OK", (dialog, which) -> dialog.dismiss())
            .setNegativeButton( text: "Calculate Distance", (dialog, which) -> {
                if (place != null && place.getLatLng() != null && currentLocation != null) {
                    clearPolylines();
                    showModeSelectionDialog(place);
                }
            })
            .show();
}
1 usage  Lowzy2k
private void showModeSelectionDialog(Place place) {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Select Mode")
            .setPositiveButton( text: "WalKing", (dialog, which) -> {
                dialog.dismiss();
                calculateDistance(place, AbstractRouting.TravelMode.WALKING);
            })
            .setNegativeButton( text: "Driving", (dialog, which) -> {
                dialog.dismiss();
                calculateDistance(place, AbstractRouting.TravelMode.DRIVING);
            })
            .show();
}

2 usages  Lowzy2k
private void calculateDistance(Place place, AbstractRouting.TravelMode mode) {
    if (place != null && place.getLatLng() != null && currentLocation != null) {
        clearPolylines();
        start = new LatLng(currentLocation.getLatitude(), currentLocation.getLongitude());
        end = place.getLatLng();
        Findroutes(start, end, mode);
    }
}
```

Μετά από κάθε αναζήτηση του χρήστη μέσω της addPlacetoFirestore, αποθηκεύουμε την τοποθεσία στη βάση δεδομένων.

```
1 usage   michalisign
private void addPlaceToFirestore(Place place) {
    // Create a reference to the "places" collection
    CollectionReference placesRef = db.collection( collectionPath: "places");


    // Create a document with a unique ID
    String documentId = placesRef.document().getId();

    // Create a Place object with the required information

    PlaceData placeData = new PlaceData(
            place.getId(),
            place.getName(),
            place.getLatLng().latitude,
            place.getLatLng().longitude
    );

    // Add the place data to Firestore
    placesRef.document(documentId).set(placeData);
}
```

Πέραν της onCreate, έχουμε την μέθοδο onMapReady, όπου για αρχή καλεί τον χάρτη και δεδομένου της άδειας χρήσης τοποθεσίας διαβάζει τα δεδομένα και εμφανίζει την θέση του χρήστη στο χάρτη, αλλιώς, ένα μήνυμα αποτυχίας εύρεσης τοποθεσίας. Μέσα της επίσης, υπάρχει η εντολή setOnMapClickListener, η οποία με την χρήση της μεθόδου onMapClick επιτρέπει στο χρήστη να πατήσει οπουδήποτε πάνω στο χάρτη χωρίς αυτό να είναι κάποιος χώρος ή αξιοθέατο και να εμφανιστεί βάδην δρομολόγηση. Αν ξαναπατήσει κάπου ο χρήστης η διαδρομή μεταφέρετε προς τη νέα κατεύθυνση.

```java
@Override
public void onMapReady(@NonNull GoogleMap googleMap) {

    myMap = googleMap;

    if (currentLocation != null) {
        LatLng lcn = new LatLng(currentLocation.getLatitude(), currentLocation.getLongitude());
        myMap.addMarker(new MarkerOptions().position(lcn).title("My Location"));
        myMap.moveCamera(CameraUpdateFactory.newLatLngZoom(lcn, zoom: 15f));
    } else {
        Toast.makeText( context: this, text: "Unable to get current location", Toast.LENGTH_SHORT).show();

    }

    thanoslgks +1 *
    myMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
        no usages  thanoslgks +1 *
        @Override
        public void onMapClick(LatLng latLng) {

            end=latLng;
            myMap.clear();

            start=new LatLng(currentLocation.getLatitude(),currentLocation.getLongitude());
            //start route finding
            Findroutes(start,end, AbstractRouting.TravelMode.WALKING);

            //calculate distance
            float[] results = new float[1];
            Location.distanceBetween(
                    currentLocation.getLatitude(),
                    currentLocation.getLongitude(),
                    end.latitude,
                    end.longitude,
                    results);

            Toast.makeText( context: MainActivity.this, text: "Απόσταση: " + results[0] + " meters", Toast.LENGTH_SHORT).show();
        }
    });
```

Τέλος, μέσα στην onMapReady έχουμε τοποθετήσει και 3 πινέζες όπου μας ζητήθηκαν με την εντολή myMap.addMarker.

```java
        LatLng serres = new LatLng( latitude: 41.07670157862302,  longitude: 23.554400400271827);
        myMap.addMarker(new MarkerOptions().position(serres).title("serres")
                .icon(bitmapDescriptor(getApplicationContext(), R.drawable.pin)));
        LatLng hotel = new LatLng( latitude: 41.10409809049689,  longitude: 23.549068805161838);
        myMap.addMarker(new MarkerOptions().position(hotel).title("hotel")
                .icon(bitmapDescriptor(getApplicationContext(), R.drawable.pin)));


        LatLng mouseio_serres = new LatLng( latitude: 41.091226420839696,  longitude: 23.54935511484131);
        myMap.addMarker(new MarkerOptions().position(mouseio_serres).title("mouseio_serres")
                .icon(bitmapDescriptor(getApplicationContext(), R.drawable.pin)));


        myMap.getUiSettings().setZoomControlsEnabled(true);
        myMap.getUiSettings().setCompassEnabled(true);
```

Έπειτα έχουμε την μέθοδο Findroutes η οποία με την χρήση του API key μας και του Routing βρίσκει και εμφανίζει τη διαδρομή του χρήστη προς την τοποθεσία που αναζητήθηκε. Αν η τοποθεσία δε βρεθεί εμφανίζεται ανάλογο μήνυμα.

```
4 usages  thanoslgks + 1
private void Findroutes(LatLng start, LatLng end, AbstractRouting.TravelMode mode) {

    if(start==null || end==null) {
        Toast.makeText( context: MainActivity.this, text: "Unable to get location", Toast.LENGTH_LONG).show();
    }
    else
    {

        Routing routing = new Routing.Builder()
                .travelMode(mode)
                .withListener(this)
                .alternativeRoutes(true)
                .waypoints(start, end)
                .key("AIzaSyAkN5S8_mhBiljsTKC7LuvT_eCt1Z8DQFI") // also define your api key here.
                .build();
        routing.execute();
    }
}

no usages  thanoslgks
public void onRoutingFailure(RouteException e) {
    View parentLayout = findViewById(android.R.id.content);
    Snackbar snackbar= Snackbar.make(parentLayout, e.toString(), Snackbar.LENGTH_LONG);
    snackbar.show();
      Findroutes(start,end);
}
```

Μετά την επιτυχία αναζήτησης της τοποθεσίας τα παραπάνω γίνονται με εξής μεθόδους :

1. OnRoutingStart εμφανίζει μήνυμα «Εύρεση Διαδρομής»
2. OnRoutingSuccess Υπολογίζει την απόσταση και κάνει το ανάλογο zoom - ξezoom ώστε να φαίνονται τα δύο σημεία στο χάρτη. Επίσης, μέσα σε αυτήν γίνεται και η τοποθέτηση pin επάνω στο χάρτη.

```
public void onRoutingStart() {
    Toast.makeText( context: MainActivity.this, text: "Εύρεση Διαδρομής...",Toast.LENGTH_LONG).show();
}

no usages  thanosIgks +2 *
public void onRoutingSuccess(ArrayList<Route> route, int shortestRouteIndex) {

    myMap.clear();
    CameraUpdate center = CameraUpdateFactory.newLatLng(start);
    CameraUpdate zoom = CameraUpdateFactory.zoomTo(16);
    if(polylines!=null) {...}
    PolylineOptions polyOptions = new PolylineOptions();
    LatLng polylineStartLatLng=null;
    LatLng polylineEndLatLng=null;
    float totalDistance = 0;


    polylines = new ArrayList<>();
    //add route(s) to the map using polyline
    for (int i = 0; i <route.size(); i++) {

        if(i==shortestRouteIndex)
        {
            polyOptions.color(ContextCompat.getColor( context: this, R.color.colorPrimary));
            polyOptions.width(7);
            polyOptions.addAll(route.get(shortestRouteIndex).getPoints());

            Polyline polyline = myMap.addPolyline(polyOptions);
            polylineStartLatLng=polyline.getPoints().get(0);
            int k=polyline.getPoints().size();
            polylineEndLatLng=polyline.getPoints().get(k-1);
            polylines.add(polyline);

            //upologismos apostashs
            totalDistance += route.get(shortestRouteIndex).getDistanceValue();

        }
    }

    //Add Marker on route starting position
    MarkerOptions startMarker = new MarkerOptions();
    startMarker.position(polylineStartLatLng);
    startMarker.title("My Location");
    myMap.addMarker(startMarker);

    //Add Marker on route ending position
    MarkerOptions endMarker = new MarkerOptions();
    endMarker.position(polylineEndLatLng);
    endMarker.title("Destination");
    myMap.addMarker(endMarker);

    View parentLayout = findViewById(android.R.id.content);
    Snackbar snackbar = Snackbar.make(parentLayout,  text: "Απόσταση: " + formatDistance(totalDistance), Snackbar.LENGTH_LONG);
    snackbar.setDuration(6000);
    snackbar.show();
```

Αξιοσημείωτη αναφορά είναι και η μέθοδος formatDistance όπου η μία θέτει την μονάδα απόστασης ανάλογα με την απόσταση που απέχει ο χρήστης από το προορισμό του.

```
private String formatDistance(float distance) {
    if (distance < 1000) {
        return String.format("%.0f meters", distance);
    } else {
        return String.format("%.2f km", distance / 1000);
    }
}
```

Όπως και η μέθοδος bitmapDescriptor, όπου μετατρέπει έναν πόρο με δυνατότητα σχεδίασης διανυσμάτων σε BitmapDescriptor, ο οποίος μπορεί να χρησιμοποιηθεί ως προσαρμοσμένο εικονίδιο δείκτη σε έναν Χάρτη Google.

```
private BitmapDescriptor bitmapDescriptor(Context context, int vectorResId){
    Drawable vectorDrawable= ContextCompat.getDrawable(context,vectorResId);
    vectorDrawable.setBounds( left: 0, top: 0, vectorDrawable.getIntrinsicWidth(),vectorDrawable.getIntrinsicHeight())
    Bitmap bitmap=Bitmap.createBitmap(vectorDrawable.getIntrinsicWidth(),
            vectorDrawable.getIntrinsicHeight(),Bitmap.Config.ARGB_8888);
    Canvas canvas=new Canvas(bitmap);
    vectorDrawable.draw(canvas);
    return BitmapDescriptorFactory.fromBitmap(bitmap);
}
```

Τέλος, η μέθοδος onNavigationItemSelected, συνοπτικά, αυτή η μέθοδος χρησιμοποιείται για το χειρισμό των επιλογών στοιχείων πλοήγησης, όπου κάθε στοιχείο αντιστοιχεί σε μια συγκεκριμένη ενέργεια ή δραστηριότητα που πρόκειται να ξεκινήσει.

- **Location Helper:** Η κλάση LocationHelper.java είναι αυτή η οποία ευθύνεται για την παράκληση προς τον χρήστη να δώσει την άδεια του για την αναζήτηση της τοποθεσίας του και καλείτε στη MainActivity.java

```
public class LocationHelper {
    1 usage
    private final int FINE_PERMISSION_CODE = 1;
    5 usages
    private final Activity context;
    2 usages
    private final FusedLocationProviderClient fusedLocationProviderClient;

    1 usage  Lowzy2k
    public LocationHelper(Activity context, FusedLocationProviderClient fusedLocationProviderClient) {
        this.context = context;
        this.fusedLocationProviderClient = fusedLocationProviderClient;
    }
    2 usages  Lowzy2k *
    public void getLastLocation(OnSuccessListener<Location> onSuccessListener) {
        if (context == null) {
            return;
        }

        if (ActivityCompat.checkSelfPermission(context, android.Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(context, android.Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(context, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, FINE_PERMISSION_CODE);
            return;
        }

        Task<Location> task = fusedLocationProviderClient.getLastLocation();
        task.addOnSuccessListener(onSuccessListener);
    }

}
```

- **Place Data:** Η κλάση PlaceData.java έχει τα getters και τα setters των τοποθεσιών που επιλέγει ο χρήστης μέσα από το SearchBar, τα οποία είναι:

1. placeID **2.placeName** **3.location**

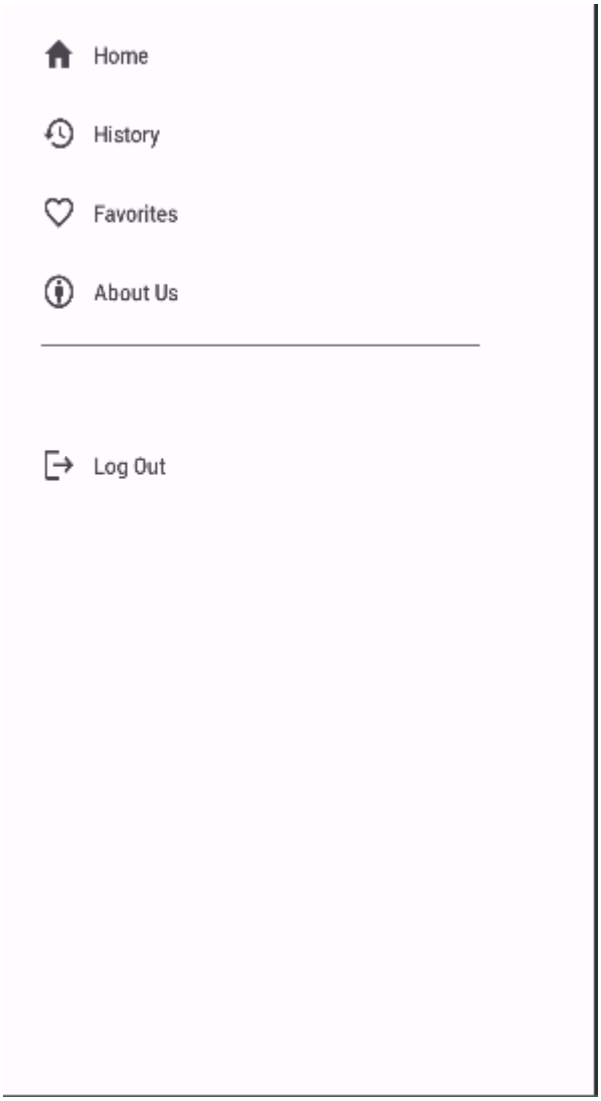
- **History:** Η κλάση History.java εμπεριέχει ένα RecyclerView, το οποίο επιτρέπει στο χρήστη να δει ότι τοποθεσία έχει αναζητήσει μέχρι εκείνη τη στιγμή από το SearchBar, λαμβάνοντας τα δεδομένα αυτά μέσω της βάσης.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_history);

    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    drawerLayout = findViewById(R.id.drawer_layout2);
    recyclerView = findViewById(R.id.recyclerView);

    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, R.string.open_nav,
            R.string.close_nav);
    drawerLayout.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);

    placeAdapter = new PlaceAdapter( context: this, placeNames, onItemClickListener: this);
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    recyclerView.setAdapter(placeAdapter);

    db.collection( collectionPath: "places") CollectionReference
            .get() Task<QuerySnapshot>
            Lowzy2k +1
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                Lowzy2k +1
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            if (document.contains("placeName")) {
                                String placeName = document.getString( field: "placeName");
                                placeNames.add(placeName);
                            }
                        }

                        Lowzy2k
                        runOnUiThread(new Runnable() {
                            Lowzy2k
                            @Override
                            public void run() { placeAdapter.notifyDataSetChanged(); }
                        });
                    } else {
                        Log.d(TAG, msg: "Error getting documents: ", task.getException());
                    }
```

- **Favorites:** Η κλάση Favorites.java, όταν εκτελείται, συμπληρώνει το RecyclerView με τα ονόματα των αγαπημένων τοποθεσιών που ανακτήθηκαν από τη βάση δεδομένων Firestore.

```
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

drawerLayout = findViewById(R.id.drawer_layout2);
recyclerView = findViewById(R.id.recyclerView);

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, "Open Navigation Drawer",
        R.string.close_nav);
drawerLayout.addDrawerListener(toggle);
toggle.syncState();

NavigationView navigationView = findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);

placeAdapter = new PlaceAdapter( context: this, placeNames, onItemClickListener: this);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
recyclerView.setAdapter(placeAdapter);

db.collection( collectionPath: "Favorites") CollectionReference
        .get() Task<QuerySnapshot>
        Lowzy2k
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            Lowzy2k
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        if (document.contains("placeName")) {
                            String placeName = document.getString( field: "placeName");
                            placeNames.add(placeName);
                        }
                    }

                    Lowzy2k
                    runOnUiThread(new Runnable() {
                        Lowzy2k
                        @Override
                        public void run() { placeAdapter.notifyDataSetChanged(); }
                    });
                } else {
                    Log.d(TAG, msg: "Error getting documents: ", task.getException());
                }
```

- **Fragments:** Οι κλάσεις των fragments μας βοηθάνε να μεταφέρουμε τον χρήστη σε όποια δραστηριότητα αυτός επιθυμεί.



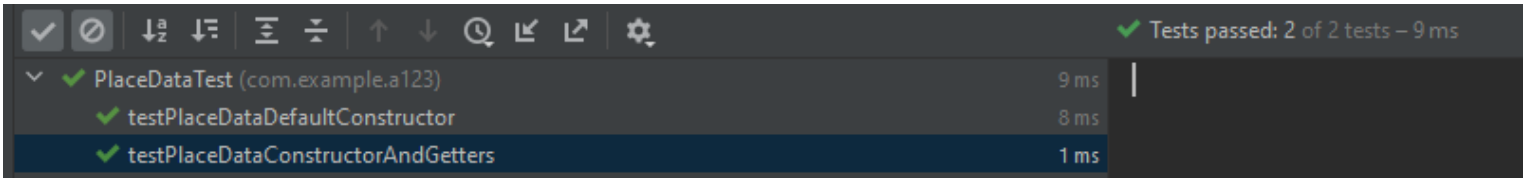

```
<item
    android:id="@+id/nav_home"
    android:icon="@drawable/nav_home"
    android:title="Home"/>

<item
    android:id="@+id/nav_history"
    android:icon="@drawable/baseline_history_24"
    android:title="History"/>

<item
    android:id="@+id/nav_favorites"
    android:icon="@drawable/baseline_favorite_border_24"
    android:title="Favorites"/>

<item
    android:id="@+id/nav_aboutus"
    android:icon="@drawable/baseline_aboutus_24"
    android:title="About Us"/>
```

```
public class HistoryFragment extends Fragment {

    XGkara
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.activity_history, container, attachToRoot: false);
    }
}
```

Testing: Ξεκινήσαμε κάνοντας μερικά τεστ για τον έλεγχο εγκυρότητας των κλάσεων μας και για την επαλήθευση των λειτουργειών τους.Αυτο το καταφέραμε χρησιμοποιώντας JUnit, Mockito και Robolectric.

```
public class PlaceDataTest {
    no usages  Lowzy2k *
    @Test
    public void testPlaceDataConstructorAndGetters() {
        String placeId = "123";
        String placeName = "Test Place";
        double latitude = 12.345;
        double longitude = 67.890;

        PlaceData placeData = new PlaceData(placeId, placeName, latitude, longitude);

        assertEquals(placeId, placeData.getPlaceId());
        assertEquals(placeName, placeData.getPlaceName());
        GeoPoint location = placeData.getLocation();
        assertEquals(latitude, location.getLatitude(), delta: 0.001);
        assertEquals(longitude, location.getLongitude(), delta: 0.001);
    }

    no usages  Lowzy2k *
    @Test
    public void testPlaceDataDefaultConstructor() {

        PlaceData placeData = new PlaceData();

        assertNotNull(placeData);
    }
}
```

Παρακάτω φαίνεται η εγκυρότητα του συγκεκριμένου παραδείγματος:



Η Εξέλιξη της Βάσης Δεδομένων

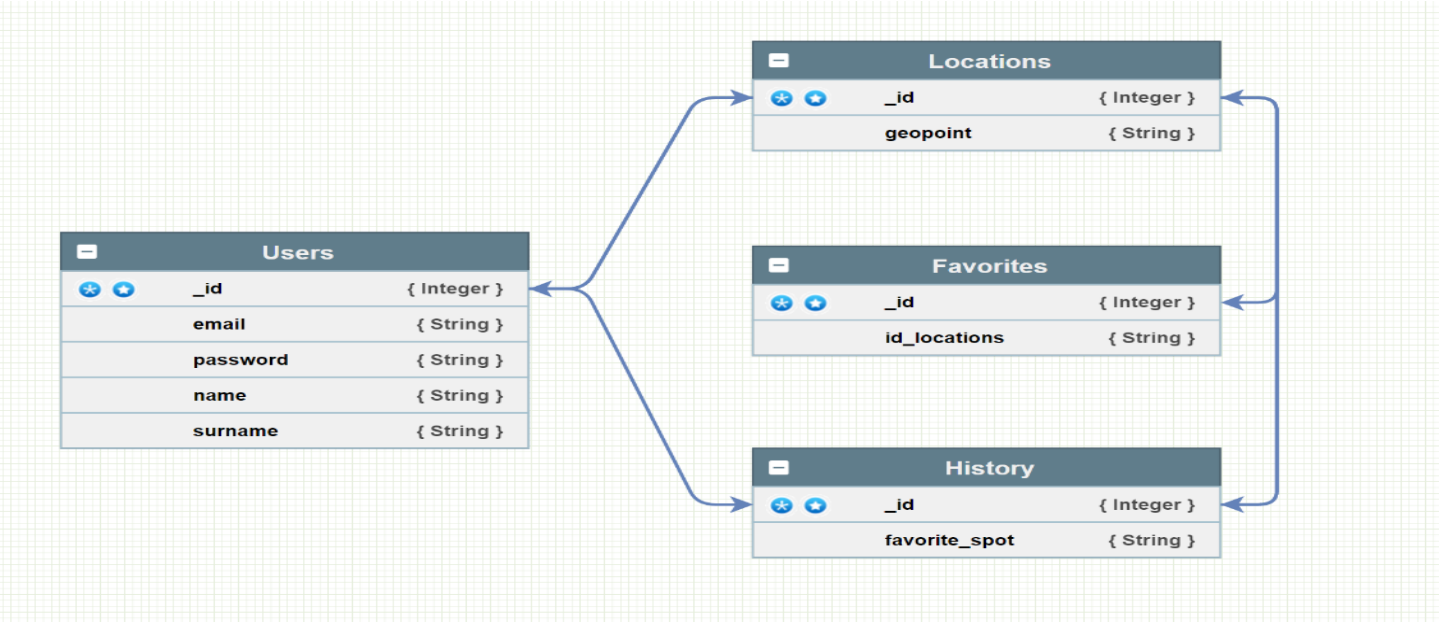
Αρχικώς , το πλάνο που είχε αποφασιστεί από την ομάδα ότι θα υλοποιηθεί για την βάση δεδομένων ήταν η χρήση της MySQL, χρησιμοποιώντας το panel του phpMyAdmin.

- Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα. Το όνομά της είναι ένας συνδυασμός του “My” , το όνομα της κόρης του συνιδρυτή Michael Widenius, My, και του ‘’SQL’’, το ακρωνύμιο για τη γλώσσα δομημένων ερωτημάτων.
- Το phpMyAdmin είναι ένα δωρεάν εργαλείο διαχείρισης ανοιχτού κώδικα για MySQL και MariaDB. Ως φορητή εφαρμογή ιστού γραμμένη κυρίως σε PHP, έχει γίνει ένα από τα πιο δημοφιλή εργαλεία διαχείρισης βάσεων MySQL,ειδικά για υπηρεσίες φιλοξενίας ιστοσελίδων
- Το XAMPP είναι ένα δωρεάν και ανοιχτού κώδικα πακέτο στοίβας λύσεων διακομιστή ιστού πολλαπλών πλατφορμών που αναπτύχθηκε από την Apache Friends,αποτελούμενο κυρίως από τον διακομιστή HTTP Apache, τη βάση δεδομένων MariaDB και διερμηνείς για σενάρια γραμμένα στις γλώσσες προγραμματισμού PHP και Perl.
- Το PhpStorm είναι γνωστό για το Visual Debugger μηδενικής διαμόρφωσης, παρέχοντας εξαιρετική εικόνα για το τι συμβαίνει στην εφαρμογή σε κάθε βήμα. Λειτουργεί με Xdebug και Zend Debugger και μπορεί να χρησιμοποιηθεί τόσο τοπικά όσο και απομακρυσμένα.

Χρησιμοποιώντας τα παραπάνω εργαλεία και ένα πρόχειρο ER που είχε φτιαχτεί , σχεδιάστηκε το αρχικό στάδιο της βάσεως δεδομένων, η οποία έτρεχε locally με το XAMPP Control Panel.
Η βάση αποτελούταν από ένα table που το είχαμε ονομάσει users1. Μέσα σε αυτό, υπήρχαν τα εξής 5 πεδία:

1. **id**
2. **name**
3. **Email**
4. **Password**
5. **apiKey**

Το ***id*** είχε τύπο int και θα χρησιμοποιούταν για να επιτευχθεί η σύζευξη μεταξύ των tables που θα χρησιμοποιούσαμε(Είχαμε ως σκοπό να φτιάξουμε 4 tables συνολικά μαζί με το Users : το **Locations** ,το οποίο θα χρησιμοποιούταν για τις εγγραφές για τις τοποθεσίες που θα επισκεπτόταν ο χρήστης ,το Favourites, το



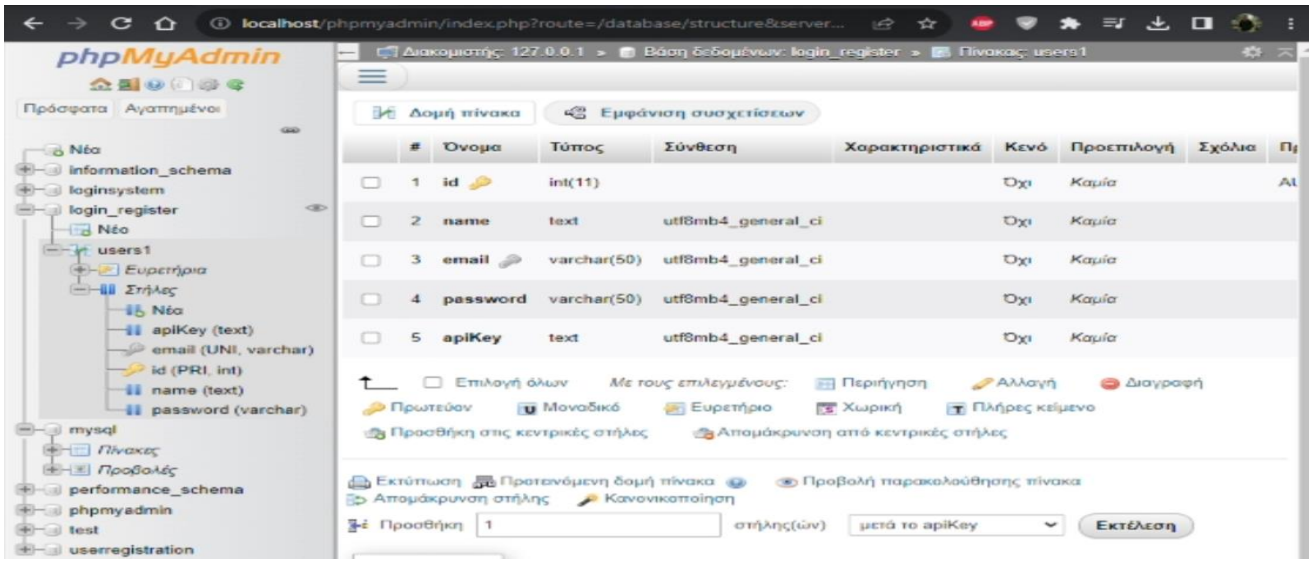
οποίο θα έδινε την δυνατότητα αποθήκευσης ενός αγαπημένου μέρους του χρήστη και το History, το οποίο θα εμφάνιζε τις εγγραφές του ιστορικού αναζήτησης του χρήστη.Το ***name*** είχε τύπο text και θα χρησιμοποιούταν για να αποθηκεύει το όνομα και το επίθετο του χρήστη.

Το ***email*** είχε τύπο varchar(50) και θα χρησιμοποιούταν για να αποθηκεύει το email και το επίθετο του χρήστη.

Ο ***κωδικός*** είχε τύπο varchar(50), με την προοπτική ότι αργότερα θα προσθέταμε και κρυπτογράφηση, για να παραμένουν ασφαλή τα προσωπικά δεδομένα του χρήστη.

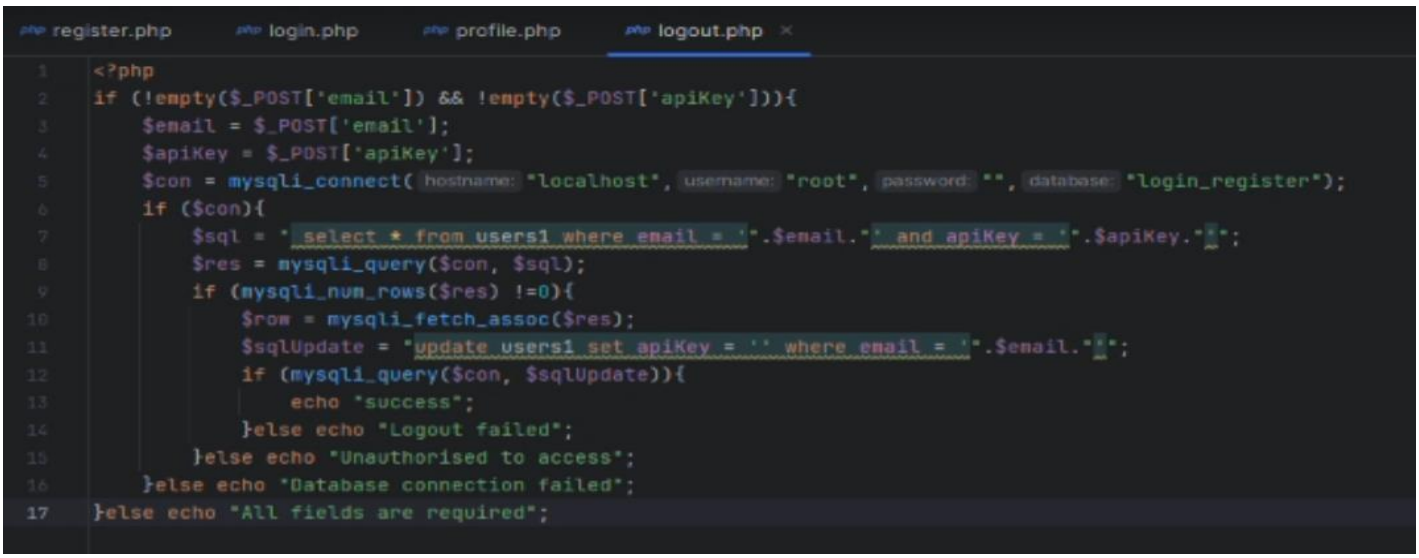
Το ***apiKey*** είχε τύπο text και θα χρησιμοποιούταν για την άντληση των δεδομένων των api της google μέσω json αρχείου.

Έπειτα , είχαμε ορίσει ως primary key το πεδίο id (του χρήστη) , ώστε να μπορούσε να πραγματοποιηθεί η σύνδεση μεταξύ των tables.



Συμπληρωματικά με τα προαναφερθέντα, είχαν γραφτεί και 4 php scripts ,χρησιμοποιώντας το PhpStorm, τα οποία θα είχαν ως σκοπό την αλληλομεταφορά πληροφορίας μεταξύ της βάσης δεδομένων και των φορμών που είχαμε δημιουργήσει για την εφαρμογή μας.

- Ένα για το logout



Ανάλυση κώδικα:

Ο κώδικας logout.php είχε ως σκοπό την ανάλυση και την υλοποίηση της λειτουργίας ενός συστήματος αποσύνδεσης για τους χρήστες της εφαρμογής μας. Αναλυτικότερα, ο κώδικας ξεκινά ελέγχοντας ένα οι παράμετροι ‘’email’’ και ‘’apiKey’’ έχουν valid τιμές και δεν είναι κενά πεδία. Εάν λείπουν , εκτυπώνεται το μήνυμα λάθους ‘’All fields are required’’. Μετά, εάν ο έλεγχος είναι επιτυχής, τα στοιχεία ‘’email’’ και ‘’apiKey’’ αποθηκεύονται σαν μεταβλητές. Έπειτα, δημιουργεί μια σύνδεση με τη βάση δεδομένων ‘’login_register’’, κατασκευάζει και τρέχει την SQL της

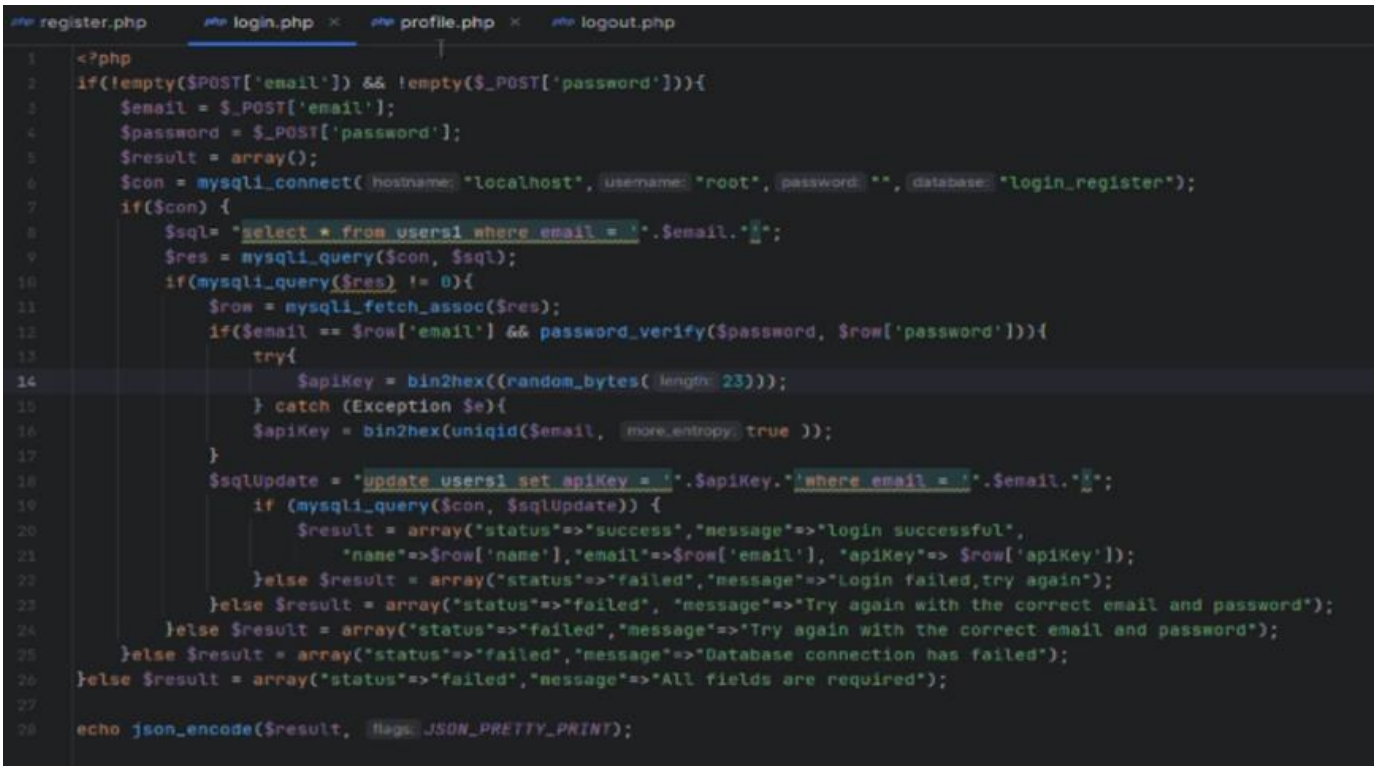
επιλογής μας ,ώστε να ελέγξει την ύπαρξη χρήστη με συγκεκριμένο email και API key. Εάν υπάρχουν αποτελέσματα, σημαίνει ότι ο χρήστης είναι εγγεγραμμένος και το API key του ενημερώνεται σε κενό, σηματοδοτώντας την αποσύνδεση του.Επίσης, εκτυπώνει το μήνυμα success αν η ενημέρωση της βάσης δεδομένων είναι επιτυχής, ειδάλλως εκτυπώνει το μήνυμα λάθους ‘’Logout failed’’. Τέλος , έχουν προστεθεί και μηνύματα λάθους για τις εξής περιπτώσεις:
‘’Unauthorised to access’’ -> Στη περίπτωση που δεν βρεθεί ο χρήστης στο σύστημα
‘’Database connection failed’’-> Στη περίπτωση που αποτύχει η σύνδεση στη βάση δεδομένων
‘’All fields are required’’-> Στη περίπτωση που λείπουν στοιχεία εισόδου.

- Ένα για το register

Ανάλυση κώδικα:

Ο κώδικας register.php είχε ως σκοπό την ανάλυση και την υλοποίηση της λειτουργίας ενός συστήματος εγγραφής χρήστη για ένα σύστημα με χρήση PHP και MySQL. Αρχικά, ο κώδικας εκτελεί έναν έλεγχο για την πληρότητα των πεδίων εισόδου σε μια φόρμα, ελέγχοντας τα πεδία ονόματος (name), email (email), και κωδικού (password). Αν κάποιο από αυτά τα πεδία δεν έχει συμπληρωθεί, ο κώδικας εκτυπώνει το μήνυμα "All fields are required to be filled". Στη συνέχεια, ο κώδικας δημιουργεί μια σύνδεση με μια βάση δεδομένων MySQL, χρησιμοποιώντας τις παραμέτρους "localhost" (διεύθυνση του διακομιστή), "root" (όνομα χρήστη), κενό κωδικό (καθώς πρόκειται για τοπική δοκιμαστική βάση), και "login_register" (όνομα της βάσης δεδομένων). Αν η σύνδεση επιτύχει, ο κώδικας προχωρά στο επόμενο στάδιο. Στο επόμενο στάδιο, ο κώδικας ανακτά τα δεδομένα που εισήγαγε ο χρήστης στα πεδία της φόρμας, δηλαδή το όνομα, το email και τον κωδικό. Τα δεδομένα αυτά αποθηκεύονται σε μεταβλητές ($name, $email, $password). Στη συνέχεια, ο κώδικας δημιουργεί και εκτελεί μια SQL εντολή που εισάγει τα παραπάνω δεδομένα του χρήστη στον πίνακα "users1" της βάσης δεδομένων. Εάν η εισαγωγή είναι επιτυχής, εκτυπώνεται το μήνυμα "successful connection". Αν, ωστόσο, η εισαγωγή αποτύχει, εκτυπώνεται το μήνυμα "Registration failed". Τέλος, σε περίπτωση που κάποιο από τα πεδία της φόρμας δεν έχει συμπληρωθεί, εκτυπώνεται το μήνυμα "All fields are required to be filled".

- Ένα για το login:



Ανάλυση κώδικα:

Ο κώδικας login.php είχε ως σκοπό να παίρνει HTTP POST αιτήσεις και ελέγχει τα δεδομένα που παρέχονται. Καταρχάς, ελέγχει αν τα πεδία 'email' και 'apiKey' υπάρχουν στο σύνολο δεδομένων που λαμβάνει μέσω HTTP POST. Αν είναι παρόντα, αναθέτει τα αντίστοιχα δεδομένα σε μεταβλητές (email και apiKey). Στη συνέχεια, δημιουργεί μια σύνδεση προς έναν τοπικό διακομιστή MySQL χρησιμοποιώντας το όνομα χρήστη 'root', κενό κωδικό πρόσβασης και τη βάση δεδομένων 'login_register'. Έπειτα, δημιουργεί και εκτελεί ένα SQL ερώτημα που ελέγχει την ύπαρξη συγκεκριμένων εγγραφών στον πίνακα 'users1', όπου το email και το apiKey ταιριάζουν με τα παρεχόμενα δεδομένα. Τέλος, αν το

ερώτημα επιστρέφει εγγραφές, δημιουργεί ένα JSON αντικείμενο που περιέχει δεδομένα χρήστη όπως το όνομα, το email και το apiKey, μαζί με ένα μήνυμα επιτυχίας. Σε αντίθετη περίπτωση, επιστρέφει ένα JSON αντικείμενο που περιέχει ένα μήνυμα αποτυχίας. Ανεξαρτήτως του αποτελέσματος, κλείνει τη σύνδεση προς τη βάση δεδομένων.

- Και ένα για το profile:

register.php | login.php | profile.php | logout.php

```php
<?php
if (!empty($_POST['email']) && !empty($POST['apiKey'])){
    $email = $POST['email'];
    $apiKey = $POST['apiKey'];
    $result = array();
    $con = mysqli_connect( hostname: "localhost", username: "root", password: "", database: "login_register");
    if($con){
        $sql = "select * from users1 where email ='".$email."'and apiKey = '".$apiKey."'";
        $res = mysqli_query($con, $sql);
        if (mysqli_num_rows($res) != 0){
            $row = mysqli_fetch_assoc($res);
            $result = array("status"=>"success","message"=>"data fetched successfully",
                "name"=>$row['name'],"email"=>$row['email'], "apiKey"=> $row['apiKey']);
        }else $result = array("status"=>"failed","message"=>"Unauthorised access");
    }else $result = array("status"=>"failed","message"=>"Database connection has failed");
}else $result = array("status"=>"failed","message"=>"All fields are required");

echo json_encode($result, flags: JSON_PRETTY_PRINT);
```

Ανάλυση κώδικα:

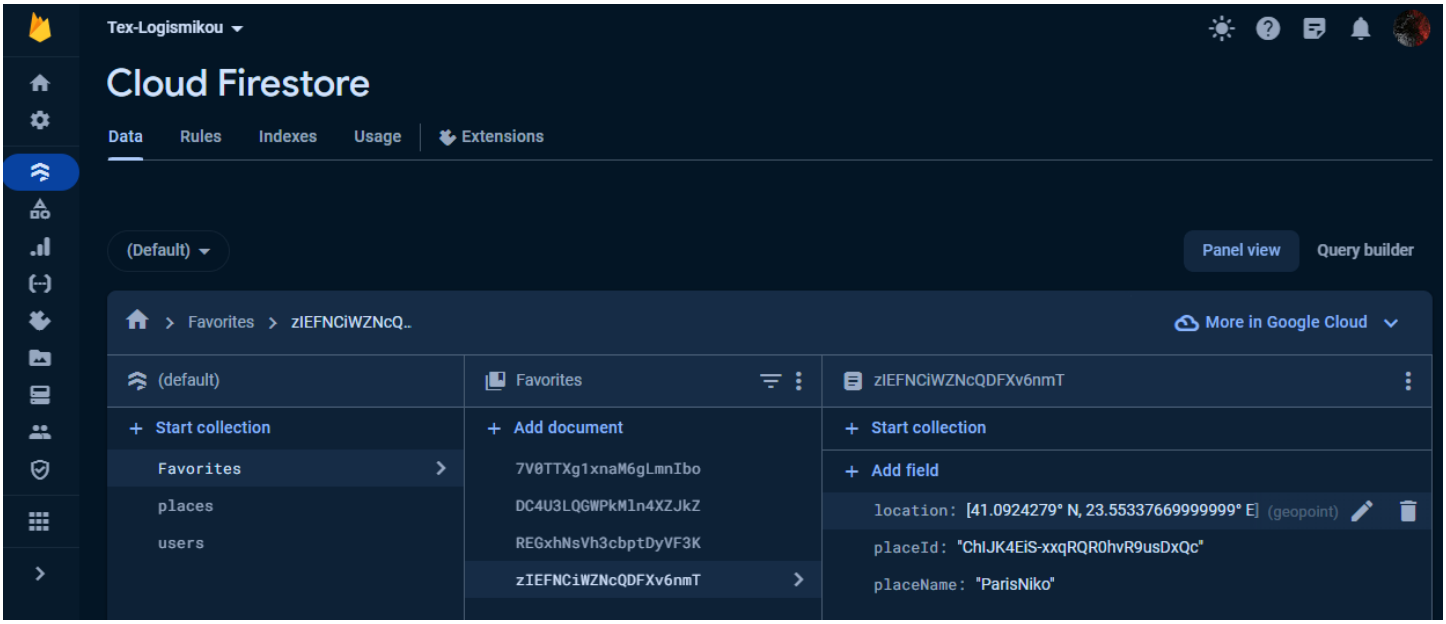
Ο κώδικας profile.php είχε ως σκοπό να λειτουργεί ως μηχανισμός ελέγχου ταυτότητας χρήστη. Αφού λαμβάνει τα δεδομένα (email και apiKey) μέσω αιτήματος POST, ελέγχει τη σύνδεση με τη βάση δεδομένων MySQL. Σε περίπτωση επιτυχίας, εκτελεί ένα ερώτημα SQL για επαλήθευση των παρεχόμενων στοιχείων. Εάν υπάρχει αντιστοίχιση, επιστρέφει τα στοιχεία του χρήστη, διαφορετικά επιστρέφει ένα μήνυμα αποτυχίας. Σε περίπτωση προβλήματος με τη σύνδεση στη βάση δεδομένων ή κενών πεδίων POST, επιστρέφει αντίστοιχα μηνύματα αποτυχίας. Τα αποτελέσματα επιστρέφονται σε μορφή JSON για απλή επεξεργασία από τον client.

Μετάβαση από MySQL σε Firebase Firestore

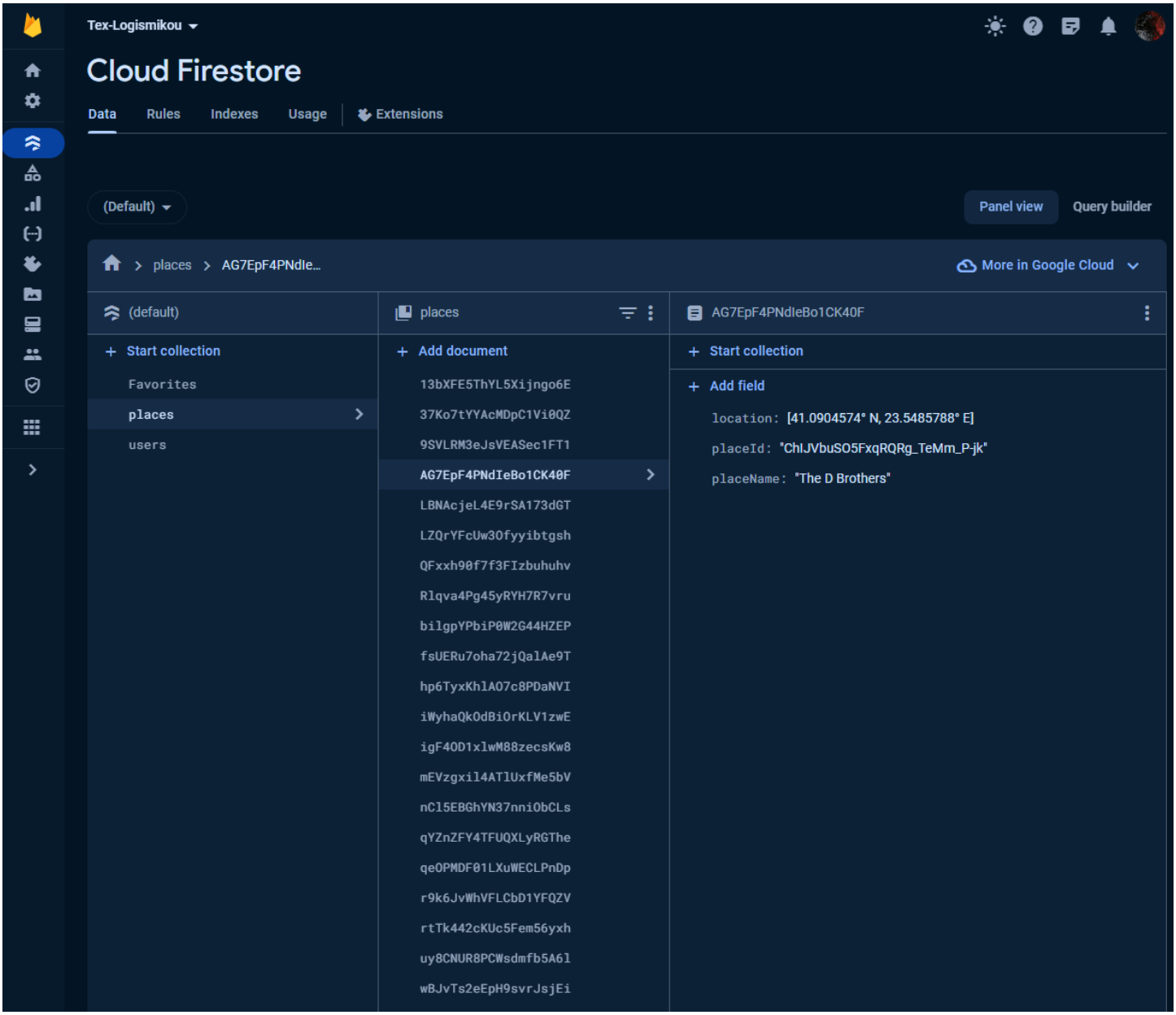
Οι λόγοι για τους οποίους μεταβήκαμε από μια βάση δεδομένων MySQL σε βάση δεδομένων Firebase Firestore, είναι ότι υπάρχουν πλεονεκτήματα τα οποία έκαναν την χρήση της πιο κατάλληλη για την ανάπτυξη της εφαρμογής μας. Στην Firebase Firestore, ως NoSQL βάση δεδομένων, ένα από τα κυριότερα πλεονεκτήματα είναι η δυνατότητα πραγματικού χρόνου συγχρονισμού δεδομένων, το οποίο μας φάνηκε πιο εύκολα διαχειρίσιμο για τον τελικό στόχο που θέλαμε να επιτύχουμε στην εφαρμογή μας. Τέλος, η Firebase Firestore είναι ευέλικτη όσον αφορά τη δομή των δεδομένων, επιτρέποντας την αποθήκευση σύνθετων δεδομένων χωρίς την ανάγκη για σχηματοποίηση (χωρίς schema). Αυτό επιτρέπει στην εύκολη προσαρμογή στις ανάγκες ανάπτυξης χωρίς περιορισμούς που μπορεί να υπάρχουν σε πιο συμβατικές σχεσιακές βάσεις δεδομένων.

Το περιβάλλον της Firebase Firestore βάσης μας:

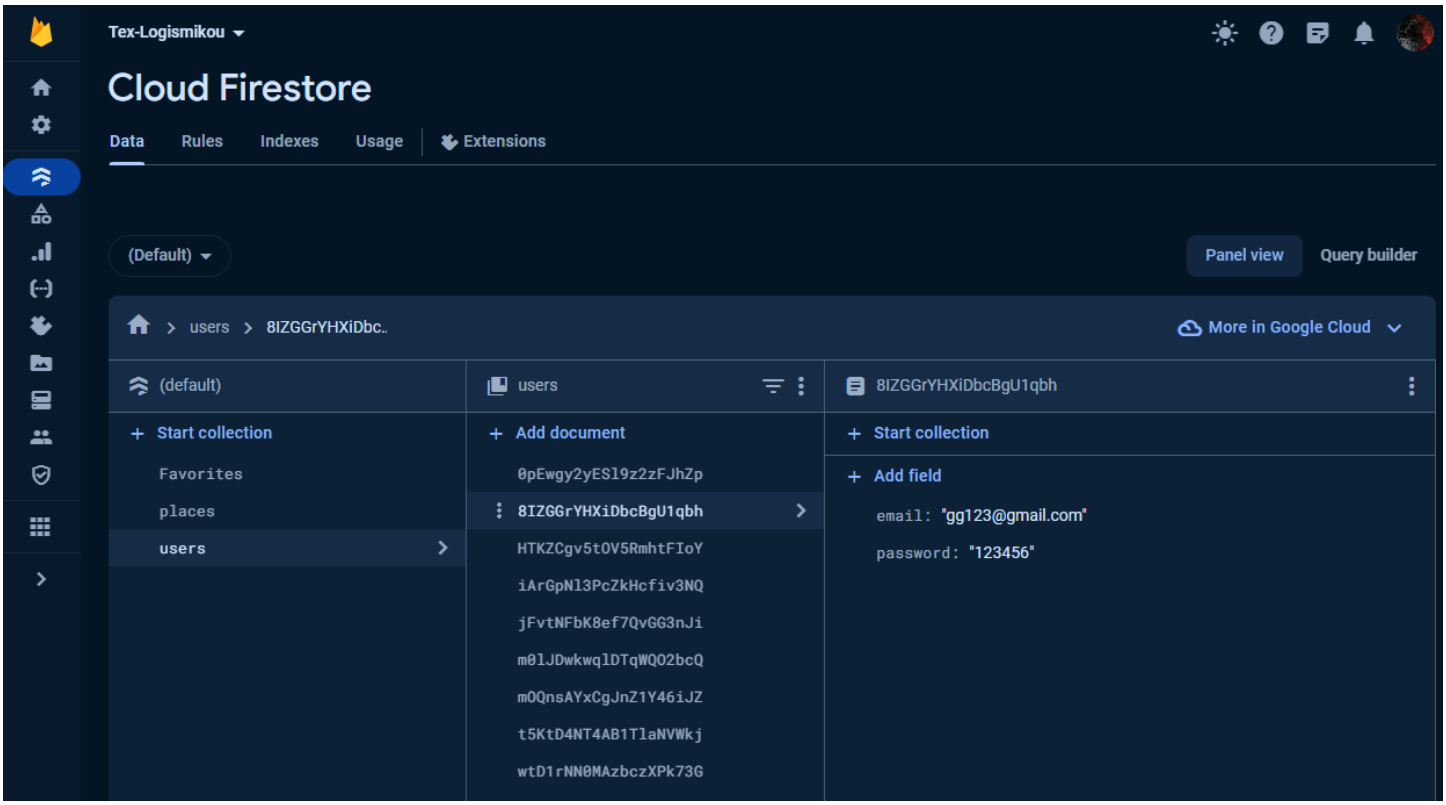
- Favorites Collection:



Ανάλυση: Αφού αναζητήσει κάτι ο χρήστης, θα εμφανιστούν οι λεπτομέρειες της τοποθεσίας σε ένα popup πλαίσιο. Μέσα σε αυτό υπάρχουν 3 κουμπιά. Μόλις πατηθεί το κουμπί “Add to Favorites”, τα αντίστοιχα πεδία location, placeId και placeName αποθηκεύονται μέσα στο collection «Favorites» και εμφανίζονται όταν ο χρήστης πατήσει το αντίστοιχο fragment μέσα από το drawer της φόρμας της εφαρμογής μας.



Ανάλυση: Μόλις ο χρήστης πληκτρολογήσει μια τοποθεσία, θα εμφανιστούν προτεινόμενα αποτελέσματα. Αφού πατήσει σε αυτό που επιθυμεί και εμφανιστούν οι πληροφορίες του, τα αντίστοιχα πεδία location (lat. , long.), placeId και placeName αποθηκεύονται μέσα στο collection «places», και έτσι δημιουργείται μια καταχώρηση για κάθε αναζήτηση.



Ανάλυση: Είναι το collection στο οποίο αποστέλλονται και αποθηκεύονται τα διαπιστευτήρια που έχει ορίσει ο χρήστης (email και password) κατά την εγγραφή του στην εφαρμογή.

ΒΙΒΛΙΟΘΗΚΕΣ

```
dependencies { this: DependencyHandlerScope

    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("com.google.firebase:firebase-auth:22.2.0")
    implementation ("com.firebase:geofire-android-common:3.1.0")
    implementation("com.google.firebase:firebase-firestore:24.9.1")
    implementation("com.google.firebase:firebase-database:20.3.0")
    implementation("com.google.android.libraries.places:places:3.3.0")
    implementation("junit:junit:4.12")
    implementation("junit:junit:4.13.2")
    implementation("androidx.test.ext:junit:1.1.5")
    implementation("org.mockito:mockito-core:5.8.0")
    implementation("androidx.test:core:1.5.0")
    implementation("org.robolectric:robolectric:4.6.1")
    testImplementation ("org.robolectric:robolectric:4.6.1")
    testImplementation("junit:junit:4.13.2")
    testImplementation ("org.mockito:mockito-core:5.8.0")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
    implementation("com.google.android.gms:play-services-maps:18.2.0")
    implementation("com.google.android.gms:play-services-location:21.0.1")
    implementation ("com.github.jd-alexander:library:1.1.0")
}
```

ΠΗΓΕΣ

- https://developers.google.com/maps/documentation/android-sdk/start#maps_android_mapsactivity-java
- https://stackoverflow.com/questions/23904281/circular-dependencies-cannot-exist-in-relativelayout-android
- https://stackoverflow.com/questions/47866062/why-android-studio-show-error-of-missing-constraints-in-constraintlayout
- https://firebase.google.com/docs/firestore/manage-data/add-data?utm_source=studio#java
- https://developers.google.com/maps/documentation/places/web-service/place-id#places-api
- https://developers.google.com/maps/documentation/routes/overview#v2-enhancements
- https://developers.google.com/maps/documentation/directions/get-directions
- https://developers.google.com/maps/documentation/javascript/examples/places-placeid-finder - maps_places_placeid_finder-javascript
- https://developers.google.com/maps/documentation/places/android-sdk/place-id
- https://developers.google.com/maps/documentation/places/android-sdk/autocomplete
- https://firebase.google.com/docs/reference/kotlin/com/google/firebase/database/DatabaseReference
- https://firebase.google.com/docs/firestore/query-data/get-data
- https://developers.google.com/maps/documentation/places/android-sdk/autocomplete
- https://developers.google.com/maps/documentation/places/android-sdk/reference/com/google/android/libraries/places/widget/Autocomplete.IntentBuilder

- https://developers.google.com/maps/documentation/directions/overview
- https://developers.google.com/maps/documentation/directions/client-library