



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διαδικτυακός Προγραμματισμός
Δεύτερη Εργασία Εαρινού Εξαμήνου 2020 – 2021

Παναγιώτης Καραμολέγκος

ΑΜ : Ε17065

p.karamolegos@yahoo.gr

Πίνακας περιεχομένων

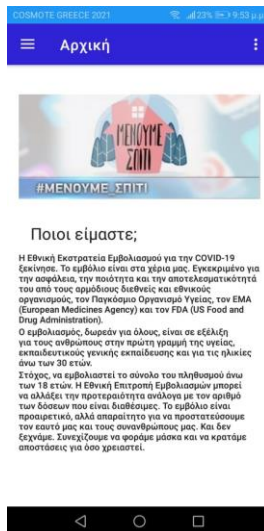
| | |
|---|----|
| 1. Εγχειρίδιο Χρήστη..... | 1 |
| 1.1. Αρχική..... | 1 |
| 1.2. Navigation Drawer | 1 |
| 1.3. Συλλογή | 2 |
| 1.4. Ραντεβού | 2 |
| 1.5. Στατιστικά | 3 |
| 1.6. Q&A..... | 4 |
| 1.7. Αλλαγή Γλώσσας | 5 |
| 1.8. Landscape mode | 5 |
| 2. Τεχνικό Εγχειρίδιο..... | 7 |
| 2.1. Τεχνολογίες που χρησιμοποιήθηκαν | 8 |
| 2.2. Main Activity..... | 9 |
| 2.3. Αρχική..... | 9 |
| 2.4. Συλλογή | 11 |
| 2.5. Ραντεβού | 13 |
| 2.6. Στατιστικά | 15 |
| 2.7. Ερωτήσεις και Απαντήσεις..... | 17 |
| 2.8. Υποστήριξη Δεύτερης Γλώσσας..... | 19 |
| 2.9. Android τροποποιήσεις | 20 |

1. Εγχειρίδιο Χρήστη

Στην ενότητα αυτήν, θα γίνει αναφορά επάνω στις διαφορετικές σελίδες που χρησιμοποιούνται εντός του Mobile App, το οποίο έχει παραδοθεί με αυτήν την αναφορά, ως προς τον τρόπο χρήσης τους από την μεριά του τελικού χρήστη.

1.1. Αρχική

Πατώντας επάνω στο εικονίδιο της εφαρμογής, ανοίγει η οθόνη που φαίνεται στην *Εικόνα 1*, μέσα σε αυτήν την οθόνη, υπάρχουν γενικές πληροφορίες για τον οργανισμό, καθώς επίσης μία σχετική εικόνα του hashtag «MENOYME_ΣΠΙΤΙ».



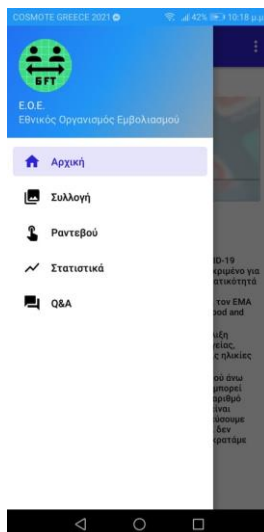
Από αυτό το σημείο, φαίνονται επάνω αριστερά, τρεις οριζόντιες παράλληλες γραμμές, από αυτές μπορούμε να κατευθυνθούμε στο Navigation Drawer και να το χρησιμοποιήσουμε για την μετακίνησή μας σε όλη την υπόλοιπη εφαρμογή. Αυτό θα αναλυθεί παραπάνω στο κεφάλαιο 1.2.

Ακόμα, πατώντας επάνω δεξιά, στις τρεις κάθετα συνεχόμενες τελείες, μπορούμε να αλλάξουμε την γλώσσα της εφαρμογής. Αυτό θα αναλυθεί παραπάνω στο κεφάλαιο 1.7.

Εικόνα 1 – Αρχική

1.2. Navigation Drawer

Στην *Εικόνα 2*, φαίνεται το Navigation Drawer το οποίο αναλαμβάνει τον ρόλο της κατεύθυνσης του χρήστη εντός της mobile εφαρμογής που έχει κατασκευαστεί.



Σε αυτό, φαίνονται οι 5 βασικές οθόνες της εφαρμογής. Αυτές αποτελούν:

1. Την Αρχική για την περιληπτική αναφορά του οργανισμού στον χρήστη.
2. Την Συλλογή, ώστε ο χρήστης να έχει στην διάθεση του μία γκάμα από φωτογραφίες και χρήσιμα σχετικά βίντεο.
3. Το Ραντεβού, από το οποίο ο Χρήστης μπορεί να κλείσει, και να ακυρώσει ένα ραντεβού του.
4. Τα Στατιστικά, στα οποία ο Χρήστης μπορεί να δει την διαδικασία των εμβολίων για τις ημερομηνίες που επιθυμεί ο ίδιος.
5. Τις Συχνές ερωτήσεις και απαντήσεις (Q&A) στις οποίες ο χρήστης μπορεί να λάβει σύντομες απαντήσεις σε πολύ πιθανές τους απορίες.

Εικόνα 2 – Navigation Drawer

1.3. Συλλογή

Ο χρήστης, ερχόμενος από το Navigation Drawer στην οθόνη της Συλλογής (Εικόνα 3), μπορεί να δει εικόνες σχετικές με τον οργανισμό, καθώς επίσης να πατήσει επάνω σε αυτές για να τις δει σε Fullscreen (Εικόνα 4). Στην συνέχεια, ξαναπατώντας τις, μπορεί να βγει από το Fullscreen mode. Για να δει όλες τις εικόνες, αρκεί να κάνει slide down επάνω τους και θα του εμφανιστούν όλες όσο κατεβαίνει ο χρήστης (Εικόνα 5). Με παρόμοιο τρόπο ο χρήστης, μπορεί να κάνει slide down για να δει όλα τα βίντεο τα οποία του παρουσιάζονται στο κάτω μέρος της οθόνης. Αυτά τα βίντεο έχουν controls του YouTube και συνεπώς ο χρήστης μπορεί να προσαρμόσει τα settings του βίντεο σαν να το βλέπει κατευθείαν από το site του YouTube.



Εικόνα 3 - Συλλογή



Εικόνα 4 - Fullscreen



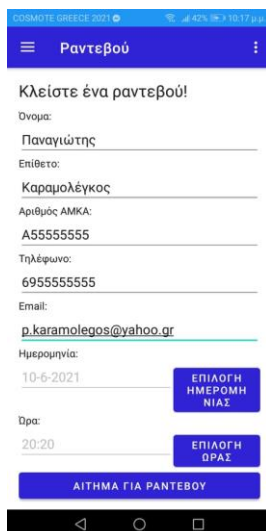
Εικόνα 5 – Scrolling

1.4. Ραντεβού

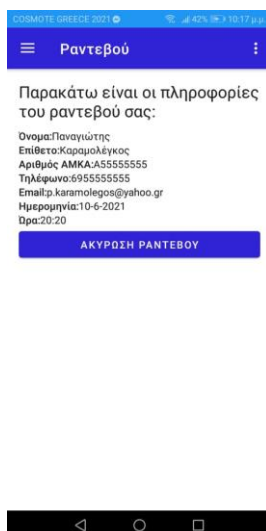
Στην οθόνη του Ραντεβού, ο χρήστης έχει την ικανότητα να δώσει τα στοιχεία του, να επιλέξει την ώρα και την ημερομηνία που επιθυμεί και να κλείσει ένα ραντεβού για να κάνει το εμβόλιό του. Στην Εικόνα 6, φαίνεται η αρχική οθόνη που εμφανίζεται στον χρήστη πριν συμπληρώσει οτιδήποτε.

Εικόνα 6 – Κενή φόρμα Ραντεβού

Να σημειωθεί ότι στα πεδία ημερομηνίας και ώρας, ο χρήστης, δεν μπορεί να γράψει ότι θέλει. Για να συμπληρώσει αυτά τα πεδία, πρέπει να πατήσει επάνω στα κουμπιά «Επιλογή Ημερομηνίας» και «Επιλογή Ώρας» αντίστοιχα, για να του εμφανιστεί ιδικό widget για την σωστή συμπλήρωση αυτών των πεδίων. Για την ημερομηνία, θα του εμφανιστεί ένα ημερολόγιο και για την ώρα ένα ρολόι.



Εικόνα 7 – Συμπληρωμένη Φόρμα Ραντεβού



Εικόνα 8 – Κλεισμένο Ραντεβού

1.5. Στατιστικά

Ο χρήστης, αφού εισαχθεί στην οθόνη «Στατιστικά» (Εικόνα 9), μπορεί να εισάγει δύο ημερομηνίες, με παρόμοιο widget που χρησιμοποιείται και στην οθόνη «Ραντεβού» για αυτές. Αφού τις εισάγει και αφού οι ημερομηνίες είναι σωστά τοποθετημένες (Δηλαδή για παράδειγμα η Από είναι Πριν της Έως) τότε μπορεί να πατήσει το κουμπί «Πάρτε τα Στατιστικά!» για να λάβει τα στατιστικά των δόσεων του εμβολίου για την παρών πανδημία για τις ημέρες που διάλεξε ο ίδιος.

Τα αποτελέσματα των Στατιστικών θα έρθουν σε έναν πίνακα που θα εμφανιστεί κάτω από το προαναφερθέν κουμπί (Εικόνα 10).



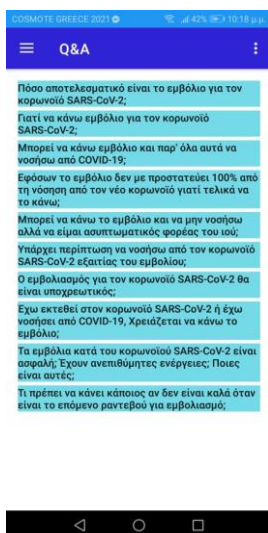
Εικόνα 9 – Στατιστικά



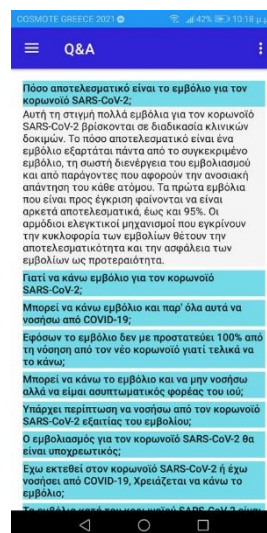
Εικόνα 10 – Συμπληρωμένη Φόρμα Στατιστικών

1.6. Q&A

Στην περίπτωση που ο χρήστης έχει κάποιες απορίες σχετικά με τον οργανισμό, τότε μπορεί να κατευθυνθεί στην οθόνη του «Q&A» που είναι η συντομία του «Questions & Answers». Εκεί θα βρει μία λίστα με ερωτήσεις (Εικόνα 11). Σε όποια ερώτηση από αυτές και αν πατήσει, θα εμφανιστεί από κάτω της μία σχετική απάντηση (Εικόνα 12).



Εικόνα 11 – Ερωτήσεις και Απαντήσεις



Εικόνα 12 – Απάντηση σε μία Ερώτηση

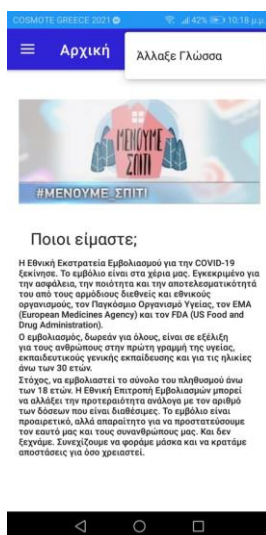
1.7. Αλλαγή Γλώσσας

Η εφαρμογή που έχει κατασκευαστεί υποστηρίζει δύο διαφορετικές γλώσσες:

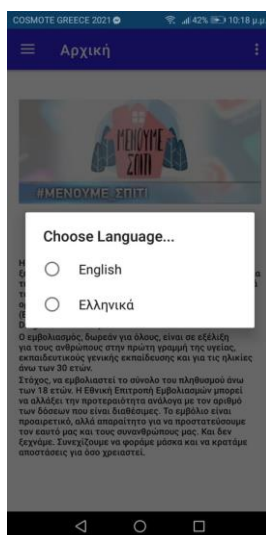
- Ελληνικά
- Αγγλικά

Η Γλώσσα με την οποία ξεκινάει η εφαρμογή να λειτουργεί, είναι η default γλώσσα που έχει επιλέξει ο χρήστης στο κινητό του.

Όμως, η εφαρμογή, έχει και build in τρόπο αλλαγής της γλώσσας της. Πατώντας οποτεδήποτε επάνω στις τρεις κάθετες συνεχόμενες τελείες επάνω δεξιά (Εικόνα 13), ο χρήστης έχει την ικανότητα να επιλέξει σε ποια γλώσσα θέλει να είναι η εφαρμογή του (Εικόνα 14). Από την στιγμή που θα κάνει την επιλογή του, τότε όλη η εφαρμογή θα αλλάξει στην επιλεγμένη γλώσσα (Εικόνα 15).



Εικόνα 13 – Αλλαξε Γλώσσα



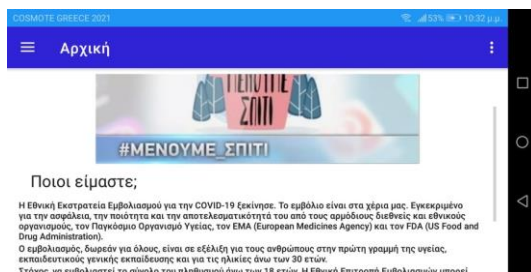
Εικόνα 14 – Επιλογή Γλώσσας



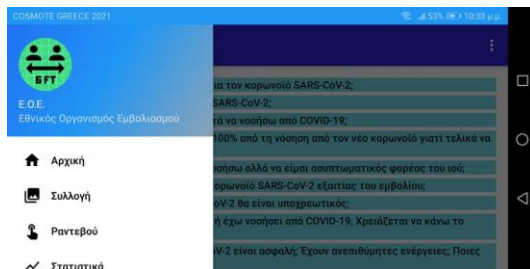
Εικόνα 15 - Αγγλικά

1.8. Landscape mode

Η εφαρμογή που έχει παραδοθεί, υποστηρίζει την λειτουργία της και σε Landscape mode. Συνεπώς, οποιαδήποτε στιγμή, ο χρήστης μπορεί να γυρίσει το κινητό του σε οριζόντια θέση και θα μπορεί να διαχειριστεί όλη την εφαρμογή κανονικά. Οι Εικόνες 16 – 21 παρουσιάζουν μερικά στιγμιότυπα αυτής της λειτουργικότητας.



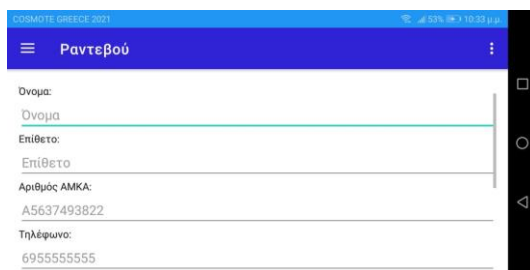
Εικόνα 16 – Landscape Αρχική



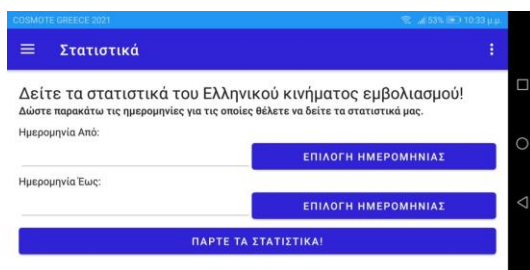
Εικόνα 17 – Landscape Navigation Drawer



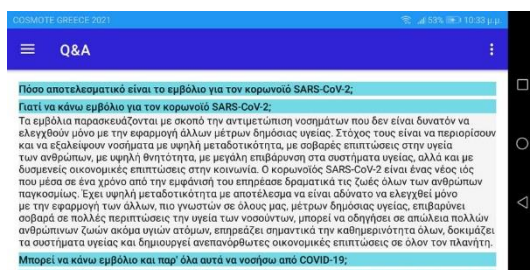
Εικόνα 18 – Landscape Συλλογή



Εικόνα 19 – Landscape Ραντεβού



Εικόνα 20 – Landscape Στατιστικά

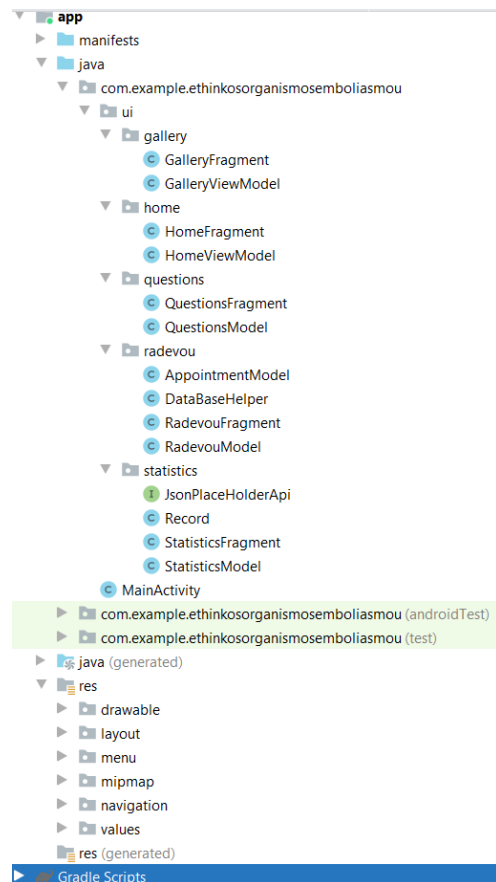


Εικόνα 21 – Landscape Q&A

2. Τεχνικό Εγχειρίδιο

Στην ενότητα αυτήν, θα γίνει αναφορά επάνω στον κώδικα των διαφορετικών οθονών που χρησιμοποιούνται εντός του Mobile App, το οποίο έχει παραδοθεί με αυτήν την αναφορά, ως προς τις λειτουργίες τους και τον τρόπο εκτέλεσής τους.

Η κατασκευή της εφαρμογής, έχει κατηγοριοποιηθεί με τον τρόπο που φαίνεται στην *Εικόνα 22*.



Ο κώδικας της εφαρμογής έχει τοποθετηθεί, όπως φαίνεται, εντός του πακέτου «ui» και μέσα στα υποπακέτα του (gallery, home, questions, radevou, statistics).

Η εφαρμογή λειτουργεί με ένα Main Activity (MainActivity.java) και καλεί Fragments για να κάνει navigate στις επιθυμητές οθόνες. Τα παρακάτω αρχεία υπάρχουν για debugging σκοπούς και δεν χρησιμοποιούνται, συνεπώς δεν θα εξηγηθούν στην αναφορά αυτήν:

- GalleryViewModel.java
- HomeViewModel.java
- QuestionsModel.java,
- RadevouModel.java
- StatisticsModel.java

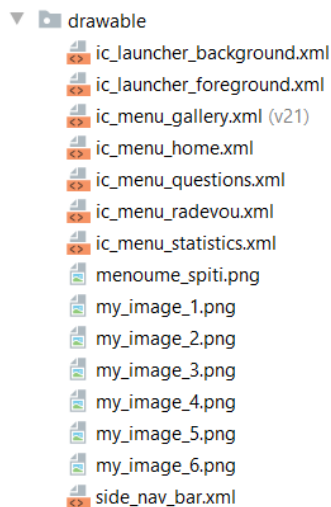
Τα αρχεία που αναφέρονται παρακάτω είναι τα fragments που χρησιμοποιούνται από το Main Activity:

- GalleryFragment.java
- HomeFragment.java
- QuestionsFragment.java,
- RadevouFragment.java
- StatisticsFragment.java

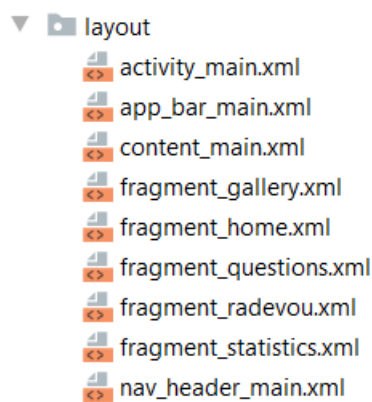
Εικόνα 22 – κατηγοριοποίηση περιεχομένου εφαρμογής

Μεγάλη σημασία, έχουν οι φάκελοι:

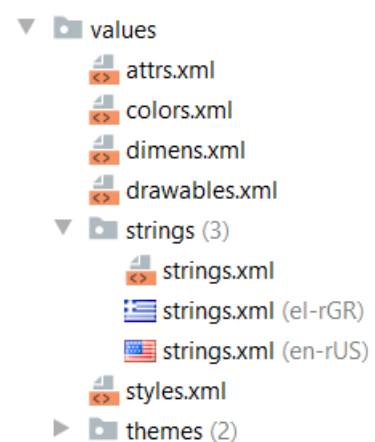
- res/drawable (Εικόνα 23)
- res/ layout (Εικόνα 24)
- res/values (Εικόνα 25)



[*Εικόνα 23 – drawable directory*](#)



[*Εικόνα 24 – layout directory*](#)



[*Εικόνα 25 – values directory*](#)

Ο φάκελος drawable είναι υπεύθυνος για τις εικόνες και τα διανυσματικά γραφικά που χρησιμοποιούνται για την επίτευξη μίας όμορφης αναπαράστασης των περιεχομένων της εφαρμογής στον χρήστη.

Ο φάκελος layout κατέχει τα xml αρχεία όλων των οθονών που παρουσιάζονται στον χρήστη, δηλαδή του navigation drawer, το βασικό activity, αλλά και τα fragments που χρησιμοποιούνται.

Ο φάκελος values περιέχει προκαθορισμένες τιμές για την χρήση τους από την εφαρμογή. Σε αυτόν περιέχονται τα χρώματα και τα μηνύματα τα οποία εμφανίζονται.

Στις υπόλοιπες ενότητες της αναφοράς αυτής (εκτός της **Ενότητας 2.1**) θα περιγραφεί ο κώδικας ως προς την λειτουργικότητά του με τρόπο ευκολοδιάβαστο. Συνιστάται να είναι ανοιχτός στην περίπτωση ανάγκης της ανάγνωσής του. Τα xml που αφορούν Android Layouts δεν θα εμφανίζονται με τον κώδικά τους, αλλά θα γίνεται η εμφάνισή τους με την χρήση του γραφικού τους Design.

2.1. Τεχνολογίες που χρησιμοποιήθηκαν

Για την κατασκευή της mobile εφαρμογής, χρησιμοποιήθηκαν οι εξής τεχνολογίες:

Android Studio: Αποτελεί ένα Integrated Development Environment (IDE) για την κατασκευή Android Εφαρμογών. Περιέχει κάποια έτοιμα Layouts, χρώματα και Activities, όπως επίσης και άλλες χρήσιμες λειτουργικότητες για την συγγραφή των τεχνολογιών που χρησιμοποιούνται μέσα σε αυτό.

Java: Προγραμματιστική Γλώσσα για την συγγραφή Αντικειμενοστραφούς Προγραμματισμού. Επιλέχθηκε η Java αντί για την Kotlin για λόγους οικειότητας.

XML: Γλώσσα για την κατασκευή Δομημένης Αναπαράστασης Δεδομένων. Χρησιμοποιήθηκε κατά κόρων για την δημιουργία των Android οθονών μέσω των Layout τους. Όπως επίσης χρησιμοποιήθηκε για την κατασκευή διανυσματικών γραφικών εντός του φακέλου drawable.

SQLite: Βάση Δεδομένων οι οποία αποθηκεύει records εντός της συσκευής τους χρήστη. Χρησιμοποιήθηκε για την αποθήκευση και την αλλαγή των ραντεβού κάθε χρήστη.

Retrofit: Java βιβλιοθήκη για την κλήση REST APIs με Android συσκευές.

HTML: Τεχνολογία κατασκευής ιστοσελίδων. Χρησιμοποιήθηκε ως input εντός μερικών Android WebViews για την εμφάνιση Video υλικού στον περιβάλλον της εφαρμογής κατευθείαν από το YouTube.

2.2. Main Activity

Στην *Εικόνα 26*, φαίνεται το JAVA περιεχόμενο του Main Activity το οποίο διαχειρίζεται το Navigation Drawer. Αυτό γίνεται με την κλήση του μέσω του R.id και στην συνέχεια την διαχείριση του NavigationView του, καλώντας το με παρόμοιο τρόπο. Για την ολοκλήρωση του Navigation Drawer τοποθετούνται εντός του Builder τα id των fragments τα οποία έχουν δοθεί στον φάκελο res/navigation.

```
DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
// Passing each menu ID as a set of Ids because each
// menu should be considered as top level destinations.
mAppBarConfiguration = new AppBarConfiguration.Builder(
    R.id.nav_home, R.id.nav_gallery, R.id.nav_radevou, R.id.nav_statistics, R.id.nav_questions)
    .setDrawerLayout(drawer)
    .build();
NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment);
NavigationUI.setupActionBarWithNavController( activity: this, navController, mAppBarConfiguration);
NavigationUI.setupWithNavController(navigationView, navController);
```

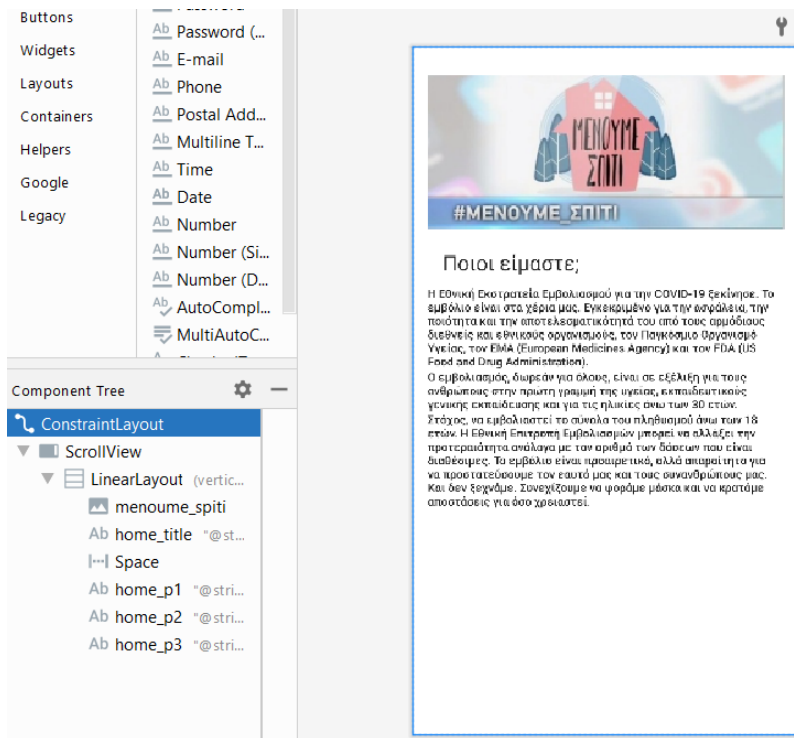
[Εικόνα 26 – Navigation Drawer Handling](#)

Εκτός από αυτό, το MainActivity.java διαχειρίζεται επίσης τα events με όνομα «getLanguages», αυτά όμως θα αναλυθούν παραπάνω στην **Ενότητα 2.8** για την Υποστήριξη Δεύτερης Γλώσσας εντός της εφαρμογής.

2.3. Αρχική

Η Αρχική οθόνη της εφαρμογής κατασκευάζεται με την χρήση ενός fragment το οποίο χρησιμοποιεί τα fragment_home.xml και HomeFragment.java για την υλοποίηση του UI του και την κωδικοποίησή του αντίστοιχα.

Στην *Εικόνα 27*, φαίνεται το Layout της προαναφερόμενης οθόνης και στην *Εικόνα 28*, βλέπουμε τον κώδικά της, ο οποίος υλοποιεί απλά το Fragment καθώς η οθόνη αυτή είναι στατική, όπως φαίνεται και από το περιεχόμενό της στην σχετική εικόνα.



[Εικόνα 27 – Layout Αρχικής οθόνης](#)

```
public class HomeFragment extends Fragment {

    private HomeViewModel homeViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

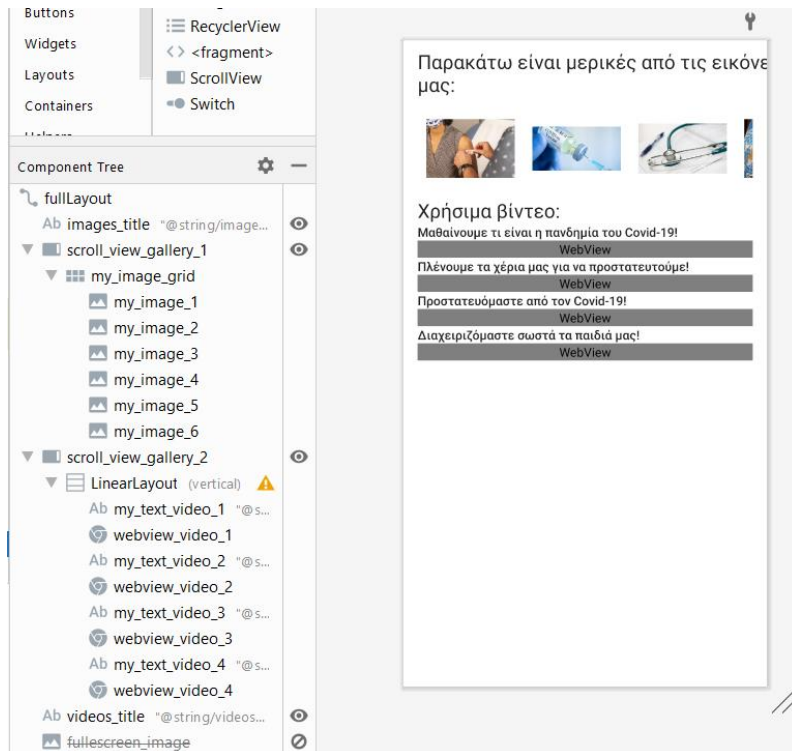
        homeViewModel =
            new ViewModelProvider( owner: this).get(HomeViewModel.class);
        // Get the homeFragment as a View
        View root = inflater.inflate(R.layout.fragment_home, container, attachToRoot: false);
        return root;
    }
}
```

[Εικόνα 28 – Κώδικας Αρχικής οθόνης](#)

2.4. Συλλογή

Το Layout της οθόνης της Συλλογής του οργανισμού (Εικόνα 29) έχει κατασκευαστεί να είναι πιο δυναμικό.

Όπως φαίνεται και στην αντίστοιχη Εικόνα, υπάρχει ένα View το οποίο είναι κρυμμένο. Αυτό το View αξιοποιείται στον κώδικα του Fragment για την εμφάνιση της εικόνας που θα πατήσει ο χρήστης. Ακόμα χρησιμοποιούνται WebViews τα οποία μέσω του κώδικα, ξανά του Fragment αυτού, εμφανίζουν στον χρήστη βίντεο από το YouTube.



Εικόνα 29 – Layout Συλλογής

Ο κώδικας αυτής της οθόνης ξεκινάει υπολογίζοντας τον αριθμό των στηλών στο GridView που υπάρχει για της εικόνες της εφαρμογής (Εικόνα 30)

```
// The below code will fix the column number for the grid of the images
final GridLayout gridLayout = root.findViewById(R.id.my_image_grid);
DisplayMetrics displayMetrics = getResources().getDisplayMetrics();
// float dpHeight = displayMetrics.heightPixels / displayMetrics.density;
float dpWidth = displayMetrics.widthPixels / displayMetrics.density;
// Log.d("MyHeight", dpHeight+"");
// Log.d("MyWidth", dpWidth+"");
gridLayout.setColumnCount((int)(dpWidth/120));
```

Εικόνα 30 – GridView column count

Στην συνέχεια γίνεται δημιουργία του html περιεχομένου το WebViews και αποθηκεύεται μέσα τους για την εμφάνισή τους στον χρήστη (Εικόνα 31).

```
// The below code is used to attach videos inside the Fragment
WebView myWebView = (WebView) root.findViewById(R.id.webview_video_1);
myWebView.getSettings().setJavaScriptEnabled(true);
String htmlContent = "<html><body>" +
    "<iframe width=\"auto\" height=\"auto\" src=\"https://www.youtube.com/watch?v=...\"></iframe>" +
    "</body></html>";
myWebView.loadData(htmlContent, mimeType: "text/html", encoding: null);

myWebView = (WebView) root.findViewById(R.id.webview_video_2);
myWebView.getSettings().setJavaScriptEnabled(true);
htmlContent = "<html><body>" +
    "<iframe width=\"auto\" height=\"auto\" src=\"https://www.youtube.com/watch?v=...\"></iframe>" +
    "</body></html>";
myWebView.loadData(htmlContent, mimeType: "text/html", encoding: null);

myWebView = (WebView) root.findViewById(R.id.webview_video_3);
myWebView.getSettings().setJavaScriptEnabled(true);
htmlContent = "<html><body>" +
    "<iframe width=\"auto\" height=\"auto\" src=\"https://www.youtube.com/watch?v=...\"></iframe>" +
    "</body></html>";
myWebView.loadData(htmlContent, mimeType: "text/html", encoding: null);

myWebView = (WebView) root.findViewById(R.id.webview_video_4);
myWebView.getSettings().setJavaScriptEnabled(true);
htmlContent = "<html><body>" +
    "<iframe width=\"auto\" height=\"auto\" src=\"https://www.youtube.com/watch?v=...\"></iframe>" +
    "</body></html>";
myWebView.loadData(htmlContent, mimeType: "text/html", encoding: null);
```

Εικόνα 31 – HTML WebViews

Η συνέχεια εμφανίζεται σε μαζεμένα code blocks (Εικόνα 32). Εντός αυτών γίνεται αρχικοποίηση των υπάρχοντων εικόνων της εφαρμογής και μετά δίνεται ένας Event Listener για την τοποθέτηση της κάθε εικόνας ως Fullscreen αφού κάνει click επάνω της ο χρήστης. Τελικά δημιουργείτε ένα Event Listener ακόμα, για την επιστροφή της οθόνης στην αρχική της κατάσταση στο μετά από ένα Fullscreen.

```
// My images array
ImageView[] images = {
    (ImageView) root.findViewById(R.id.my_image_1),
    (ImageView) root.findViewById(R.id.my_image_2),
    (ImageView) root.findViewById(R.id.my_image_3),
    (ImageView) root.findViewById(R.id.my_image_4),
    (ImageView) root.findViewById(R.id.my_image_5),
    (ImageView) root.findViewById(R.id.my_image_6)
};

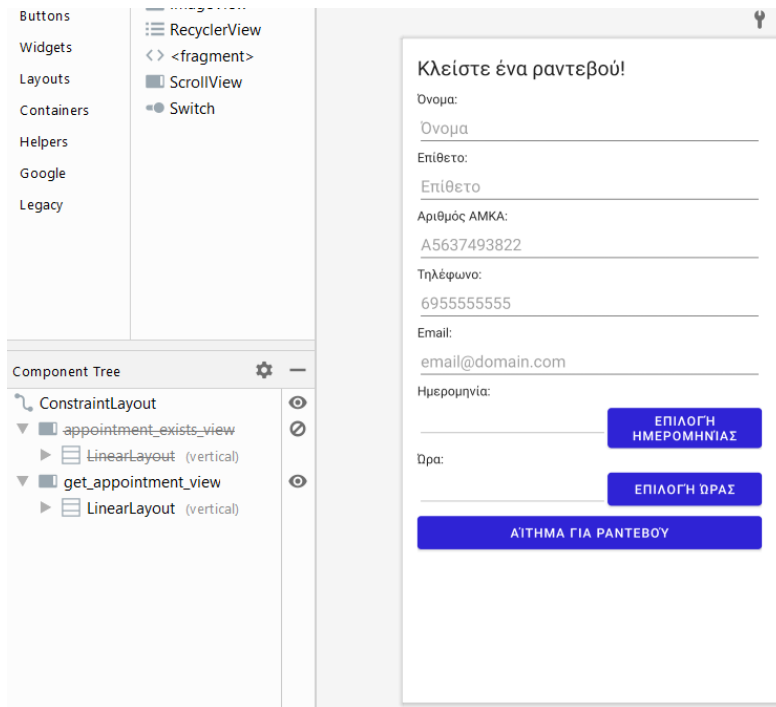
// Listeners to make images able to have fullscreen mode
for(int i=0; i<images.length; i++){
    ImageView imageView = images[i];
    imageView.setOnClickListener(new View.OnClickListener() {...});
}

// Listener to make content normal again after fullscreen mode
ImageView fullScreenImage = (ImageView) root.findViewById(R.id.fullscreen_image);
fullScreenImage.setOnClickListener(new View.OnClickListener() {...});
```

Εικόνα 32 – Fullscreen handling

2.5. Ραντεβού

Όπως φαίνεται και στην *Εικόνα 33*, η οθόνη αυτή έχει το Fragment της χωρισμένο σε δύο ScrollViews και ανάλογα με τα αποτελέσματά της στην Βάση Δεδομένων εμφανίζει το αντίστοιχο ScrollView στον χρήστη.



Εικόνα 33 – Ραντεβού Layout

Για την επίτευξη της οθόνης αυτής, έχει κατασκευαστεί αρχικά μία java κλάση για την αποτύπωση ενός αντικειμένου «Ραντεβού - Appointment» (AppointmentModel). Καθώς επίσης, έχει κατασκευαστεί η κλάση DataBaseHelper η οποία υλοποιεί ερωτήματα προς την SQLite βάση (Εικόνα 34).

```
public class DataBaseHelper extends SQLiteOpenHelper {  
  
    public DataBaseHelper(@Nullable Context context) {  
        super(context, "appointment.db", null, 1);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db){...}  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){  
  
    }  
  
    // Inserts an Appointment in APPOINTMENT_TABLE  
    public void addAppointment(AppointmentModel appointmentModel){...}  
  
    // Returns true if an appointments exists  
    // Returns false if there is no appointment  
    public boolean checkForAppointment(){...}  
  
    // Gets the first appointment  
    public AppointmentModel getAppointment(){...}  
  
    // Deletes an appointment  
    public void deleteAppointment(int id){...}  
}
```

Εικόνα 34 – DataBaseHelper

Το fragment ξεκινάει στον κώδικά του ορίζοντας DatePicker και TimePicker ώστε ο χρήστης να μπορεί να έχει ειδικά widget για τις αντίστοιχες επιλογές του (Εικόνα 35).

```
radevouModel =
    new ViewModelProvider( owner: this).get(RadevouModel.class);
View root = inflater.inflate(R.layout.fragment_radevou, container, attachToRoot: false);

btnDatePicker=(Button)root.findViewById(R.id.date_button);
btnTimePicker=(Button)root.findViewById(R.id.time_button);
txtDate=(EditText)root.findViewById(R.id.date_field);
txtTime=(EditText)root.findViewById(R.id.time_field);

// Listener to handle date picks by the user
btnDatePicker.setOnClickListener(new View.OnClickListener(){...});
// Listener to handle time picks by the user
btnTimePicker.setOnClickListener(new View.OnClickListener(){...});
```

Εικόνα 35 – Date και Time Picker

Έπειτα, η γίνεται χρήση της βοηθητικής κλάσης για τα ερωτήματα στην SQLite που αναφέρθηκε πιο πριν. Συνεχίζοντας γίνονται επιλογές ανάλογα τα αποτελέσματα των ερωτημάτων ως προς το ποια android views πρέπει να φαίνονται και ποια όχι, όπως και να επιστραφούν τα αποτελέσματα του ραντεβού στην περίπτωση που ήδη υπάρχει ένα. Η προαναφερθέντα διαδικασία αποτυπώνετε στην Εικόνα 36.

```
/** Below is the code to handle the SQLite DB**/
// Check if there is an appointment to show the right Views
DataBaseHelper dataBaseHelper = new DataBaseHelper(getActivity());
appointmentMade = dataBaseHelper.checkForAppointment();
if(appointmentMade){...}
else{...}

// If an Appointment is made then add it and change the showing views
Button submitAppointmentButton = root.findViewById(R.id.appointment_request_button);
submitAppointmentButton.setOnClickListener(new View.OnClickListener(){...});

// If an Appointment is deleted then delete it and change the showing views
Button deleteButton = root.findViewById(R.id.cancel_appointment_button);
deleteButton.setOnClickListener(new View.OnClickListener(){...});
```

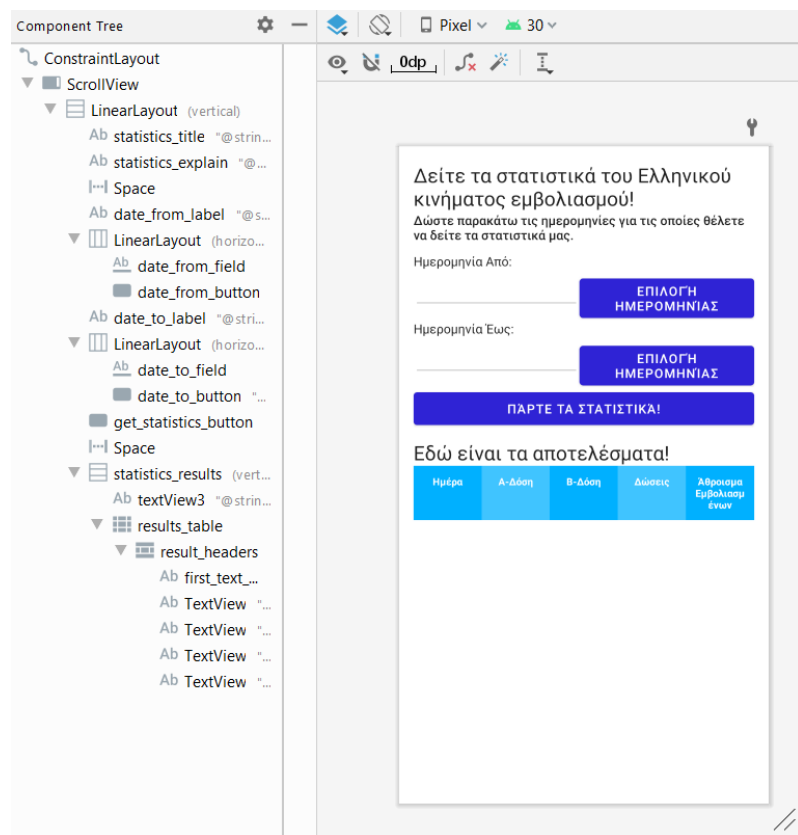
Εικόνα 36 – Διαχείριση Ραντεβού

Τέλος, κατασκευάζονται δύο Event Listeners για τα αντίστοιχα κουμπιά στο Layout που αναφέρθηκε, για να μπορεί ο χρήστης να κάνει submit ή cancel το Ραντεβού του και τελικά να πηγαίνει στην άλλη οθόνη δυναμικά.

2.6. Στατιστικά

Για την επιστροφή των στατιστικών στην αντίστοιχη οθόνη, κατασκευάστηκαν δύο βοηθητικά αρχεία JAVA. Το ένα με όνομα «Record.java» είναι το κάθε record που επιστρέφει το API των εμβολίων. Περιέχει όλα τα αντίστοιχα πεδία ενός record καθώς και getter μεθόδους για κάθε ένα από αυτά. Το δεύτερο, με όνομα «JsonPlaceHolderApi.java» είναι ένα java interface το οποίο χρησιμοποιεί την αναγκαία μορφοποίηση της Retrofit βιβλιοθήκης για την κλήση του REST API το οποίο έχει δοθεί.

Στην *Εικόνα 37*, φαίνεται το Layout της σχετικής οθόνης μαζί με τον πίνακα ο οποίος υπάρχει για να τοποθετηθεί επάνω του η απάντηση του API. Αυτός ο πίνακας θα είναι ορατός μόνο αν ο χρήστης εισάγει λογικές τιμές στις ημερομηνίες.



Εικόνα 37 – Layout Στατιστικών

Συνεχίζοντας στον κώδικα αυτής της οθόνης, αρχικά δημιουργούνται Date Pickers με παρόμοιο τρόπο όπως και στην **Ενότητα 2.5** (*Εικόνα 38*). Ύστερα, τοποθετείτε ένα Event Listener για την απόκτηση των Στατιστικών από το API επάνω στο αντίστοιχο κουμπί.

```

btnDatePickerFrom=(Button)root.findViewById(R.id.date_from_button);
txtDateFrom=(EditText)root.findViewById(R.id.date_from_field);

btnDatePickerTo=(Button)root.findViewById(R.id.date_to_button);
txtDateTo=(EditText)root.findViewById(R.id.date_to_field);

// Listener to handle date picks by the user for the Date From
btnDatePickerFrom.setOnClickListener(new View.OnClickListener(){...});

// Listener to handle date picks by the user for the Date To
btnDatePickerTo.setOnClickListener(new View.OnClickListener(){...});

// Listener to check the user input and get the final results
Button getStatisticsButton = (Button) root.findViewById(R.id.get_statistics_button);
getStatisticsButton.setOnClickListener(new View.OnClickListener(){...});

```

Εικόνα 38 – Κώδικας Στατιστικών

Στο μαζεμένο block που υπάρχει στην *Εικόνα 38*. Γίνεται αρχικά έλεγχος της σωστής τοποθέτησης των ημερομηνιών από τον χρήστη. Στην περίπτωση που υπάρχει κάποιο λάθος τότε απλά δεν επιστρέφονται αποτελέσματα και ο πίνακας παραμένει αόρατος. Όμως, αν ο χρήστης εισάγει σωστές ημερομηνίες τότε χρησιμοποιείται η Retrofit για την κλήση του API (*Εικόνα 39*), εμφανίζεται ο πίνακας και τοποθετούνται δυναμικά επάνω του τα αποτελέσματα.

```

// Getting ready to make a retrofit call
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://data.gov.gr/api/v1/query/")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

// I will use my own interface to get many records of the class Records to use them for my data
JsonPlaceholderApi jsonPlaceholderApi = retrofit.create(JsonPlaceholderApi.class);

SimpleDateFormat formatter = new SimpleDateFormat( pattern: "yyyy-MM-dd");
fromText = formatter.format(realDateFrom);
toText = formatter.format(realDateTo);

// I am giving the query variables
Call<List<Record>> call = jsonPlaceholderApi.getRecords(fromText,toText);

// I am making the call
Date finalRealDateTo = realDateTo;
Date finalRealDateFrom = realDateFrom;
call.enqueue(new Callback<List<Record>>() {...});

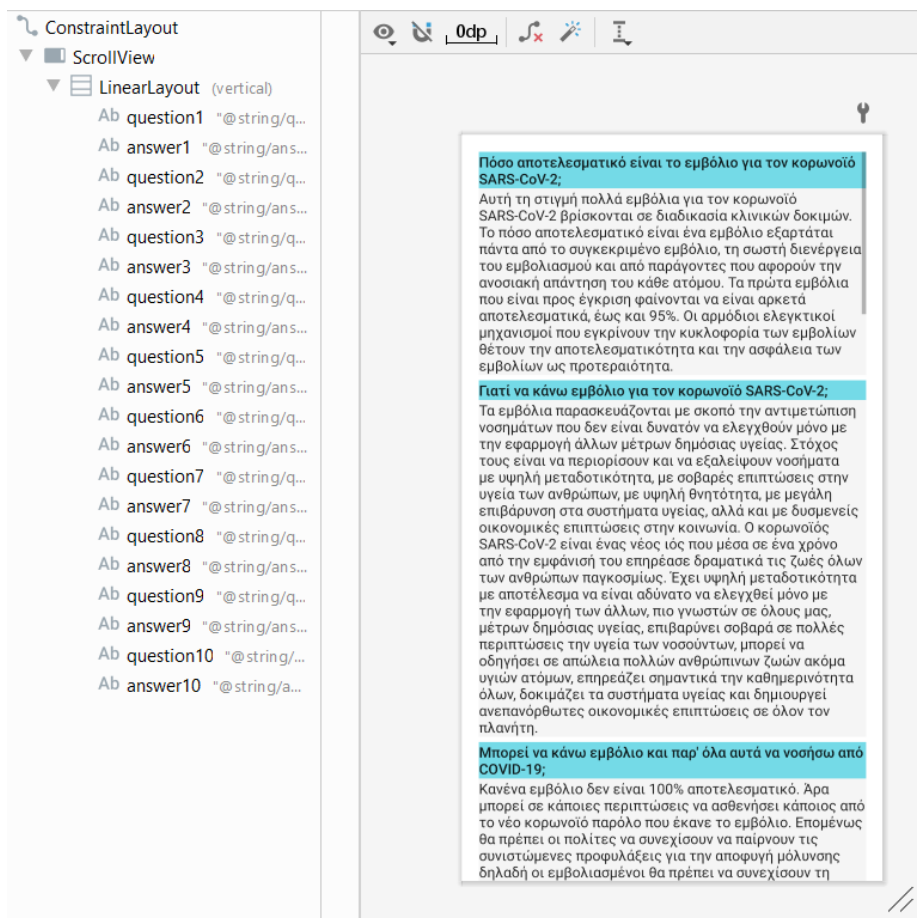
```

Εικόνα 39 – Κλήση API

Στην *Εικόνα 39*, στην γραμμή που χρησιμοποιείται η μέθοδος enqueue, γίνεται η προαναφερθέν δυναμική τοποθέτηση των δεδομένων στον πίνακα. Στην περίπτωση κάποιου σφάλματος, τότε εμφανίζονται debug messages τα οποία δεν φαίνονται στον χρήστη.

2.7. Ερωτήσεις και Απαντήσεις

Η οθόνη περί των Q&A του οργανισμού, περιέχει στο Layout της διαδοχικά TextViews τα οποία έχουν διαφορετικά χρώματα με σκοπό να ξεχωρίζουν στον χρήστη αυτά που είναι clickable με αυτά που δεν είναι (Εικόνα 40).



Εικόνα 40 – Q&A Layout

Στον κώδικα του Fragment αυτού, αρχικά γίνεται αρχικοποίηση όλων των αντίστοιχων ερωτήσεων και των απαντήσεων σε Java πίνακες (Εικόνα 41). Με την χρήση αυτών γίνεται μία επαναληπτική διαδικασία για κάθε ένα ζεύγος τους, με σκοπό το χτίσιμο μίας λειτουργικότητας τύπου Accordion (Εικόνα 42).

```

// An array holding all the questions
TextView[] questions = {
    (TextView) root.findViewById(R.id.question1),
    (TextView) root.findViewById(R.id.question2),
    (TextView) root.findViewById(R.id.question3),
    (TextView) root.findViewById(R.id.question4),
    (TextView) root.findViewById(R.id.question5),
    (TextView) root.findViewById(R.id.question6),
    (TextView) root.findViewById(R.id.question7),
    (TextView) root.findViewById(R.id.question8),
    (TextView) root.findViewById(R.id.question9),
    (TextView) root.findViewById(R.id.question10)
};

// An array holding all the answers
TextView[] answers = {
    (TextView) root.findViewById(R.id.answer1),
    (TextView) root.findViewById(R.id.answer2),
    (TextView) root.findViewById(R.id.answer3),
    (TextView) root.findViewById(R.id.answer4),
    (TextView) root.findViewById(R.id.answer5),
    (TextView) root.findViewById(R.id.answer6),
    (TextView) root.findViewById(R.id.answer7),
    (TextView) root.findViewById(R.id.answer8),
    (TextView) root.findViewById(R.id.answer9),
    (TextView) root.findViewById(R.id.answer10)
};

```

Εικόνα 41 – Java πίνακες για Ερωτήσεις και Απαντήσεις

```

// Remove the answers
for(int i=0; i<answers.length; i++){
    answers[i].setVisibility(View.GONE);
}

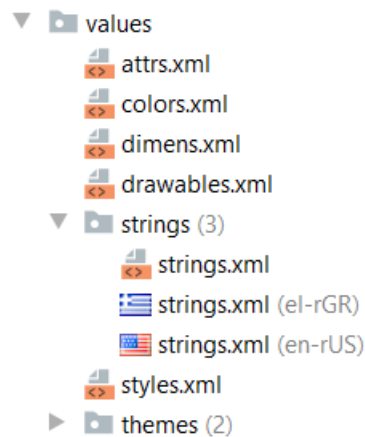
// for every question
for(int i=0; i<questions.length; i++){
    // use the position of the question as a final value to use it in the OnClickListener
    final int j = i;
    // Put an OnClickListener to remove or add the answer when clicked
    questions[i].setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v){
            if(answers[j].isShown()){
                answers[j].setVisibility(View.GONE);
            }
            else{
                answers[j].setVisibility(View.VISIBLE);
            }
        }
    });
}

```

Εικόνα 42 – Accordion Ερωτήσεων / Απαντήσεων

2.8. Υποστήριξη Δεύτερης Γλώσσας

Για την υποστήριξη δεύτερης γλώσσας η εφαρμογή χρησιμοποιεί επιπλέον string.xml αρχεία για διαφορετικές γλώσσες (*Εικόνα 43*)



Με αυτόν τον τρόπο, ο κάθε χρήστης της εφαρμογής, θα έχει μία δική του αντίστοιχη μετάφραση. Στην περίπτωση που ο χρήστης δεν έχει δική του μετάφραση (Αγγλικών ή Ελληνικών), τότε θα έχει τα αγγλικά σαν γλώσσα της εφαρμογής, καθώς είναι το default string.xml (το πάνω - πάνω) μεταφρασμένο στα αγγλικά.

Εικόνα 43– string.xml

Εκτός της παραπάνω λειτουργίας όμως, η εφαρμογή υποστηρίζει μέσω της MainActivity.java και λειτουργικότητα για την αλλαγή γλώσσας δυναμικά μέσα στην εφαρμογή κατά την ώρα της χρήσης της. Αυτό επιτυγχάνεται με τον κώδικα που εμφανίζεται στην *Εικόνα 44*.

```
// The code below will get called when the change languages button will get pressed
public void getLanguages(MenuItem item){
    final String[] listItems = {"English", "Ελληνικά"};
    AlertDialog.Builder mBuilder = new AlertDialog.Builder( context: MainActivity.this);
    mBuilder.setTitle("Choose Language...");
    mBuilder.setSingleChoiceItems(listItems, checkedItem: -1, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int i) {
            if(i == 0){
                // English
                setLocale("en");
                recreate();
            }
            if(i == 1){
                // Greek
                setLocale("el");
                recreate();
            }

            // dismiss alert dialog when language selected
            dialog.dismiss();
        }
    });

    AlertDialog mDialog = mBuilder.create();
    // show alert dialog
    mDialog.show();
}

private void setLocale(String lang){...}
```

Εικόνα 44 – Διαχείριση Αλλαγής Γλώσσας

Ο προαναφερθέν κώδικας, τρέχει όποτε πατηθεί από τον χρήστη το κουμπί «Αλλάξτε Γλώσσα» που εμφανίζεται αφού πατηθεί το κουμπί στο πάνω δεξιά μέρος της οθόνης της συσκευής με τις τρεις κάθετες συνεχόμενες τελείες. Αρχικά, εμφανίζονται όλες οι διαθέσιμες γλώσσες στον χρήστη. Στην συνέχεια αφού ο χρήστης κάνει την επιλογή του, τότε με την χρήση της υλοποιημένης συνάρτησης «setLocale» αλλάζει η περιοχή της εφαρμογής και συνεπώς και η γλώσσα της από τα string.xml

2.9. Android τροποποιήσεις

Για την κατασκευή της εφαρμογής αυτής, έπρεπε να γίνει χρήση των Retrofit dependencies, καθώς επίσης να ζητηθεί και η πρόσβαση του ίντερνερ για την επίτευξη της δημιουργίας των Webviews τα οποία καλούν το YouTube έξω από το δίκτυο της συσκευής του χρήστη, από το κάθε κινητό στο οποίο θα εκτελείτε. Η πρόσβαση στο ίντερνερ, υλοποιείται κατευθείαν μέσω του AndroidManifest.xml (Εικόνα 45) μέσω της γραμμής 6, όπου γίνεται έτοιμα του ίντερνερ από την συσκευή.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.ethinkosorganismosemboliasmou">
4
5     <!-- I am using internet for the web views-->
6     <uses-permission android:name="android.permission.INTERNET"/>
7
8     <application
9         android:allowBackup="true"
10        android:icon="@mipmap/ic_launcher"
11        android:label="Ethinkos Organismos Emboliasmou"
12        android:roundIcon="@mipmap/ic_launcher_round"
13        android:supportsRtl="true"
14        android:theme="@style/Theme.EthinkosOrganismosEmboliasmou">
15        <activity
16            android:name=".MainActivity"
17            android:label="Ethinkos Organismos Emboliasmou"
18            android:theme="@style/Theme.EthinkosOrganismosEmboliasmou.NoActionBar">
19            <intent-filter>
20                <action android:name="android.intent.action.MAIN" />
21
22                <category android:name="android.intent.category.LAUNCHER" />
23            </intent-filter>
24        </activity>
25    </application>
26
27 </manifest>
```

Εικόνα 45 – Έτοιμα πρόσβασης στο Internet

Για την επιτυχή χρήση της Retrofit στο αρχείο Gradle της εφαρμογής, έγινε προσθήκη των δύο παρακάτω dependencies, όπως φαίνεται και στην Εικόνα 46:

- implementation 'com.squareup.retrofit2:retrofit:2.4.0'
- implementation 'com.squareup.retrofit2:converter-gson:2.4.0'


```
31 ► dependencies {
32
33     implementation 'androidx.appcompat:appcompat:1.3.0'
34     implementation 'com.google.android.material:material:1.3.0'
35     implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
36     implementation 'androidx.navigation:navigation-fragment:2.3.5'
37     implementation 'androidx.navigation:navigation-ui:2.3.5'
38     implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
39     implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
40     implementation 'androidx.gridlayout:gridlayout:1.0.0'
41     implementation 'com.squareup.retrofit2:retrofit:2.4.0'
42     implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
43     testImplementation 'junit:junit:4.+'
44     androidTestImplementation 'androidx.test.ext:junit:1.1.2'
45     androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
46 }
```

[Εικόνα 46 – Gradle dependencies](#)