



**ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΙΡΑΙΩΣ**

**ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ
ΣΥΣΤΗΜΑΤΩΝ**

REPAIRLOG

**KEEP YOUR REPAIRS
IN A CLEVER WAY**

**ΤΣΟΤΖΟΛΑΣ
ΓΕΩΡΓΙΟΣ**

ΜΕ 1627

ΠΑΝΕΠΙΣΤΗΜΕΙΟ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ



Εργασία Android

Τεκμηρίωση

Τσότηζολας Γεώργιος (ΜΕ1627)

Μάθημα: Κινητή Υπολογιστική και Εφαρμογές

Διδάσκων Καθηγητής: Μενύχτας Ανδρέας

ΗΜΕΡΟΜΗΝΙΑ
ΙΟΥΝ 2017



ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. ΕΙΣΑΓΩΓΗ.....	3
2. ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ	3
3. ΑΝΑΛΥΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ.....	3
3.2. ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ	3
3.3. ΔΟΜΗ ΤΟΥ ΚΩΔΙΚΑ	16
4. ΣΗΜΕΙΑ ΙΔΙΑΙΤΕΡΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟΥ ΕΝΔΙΑΦΕΡΟΝΤΟΣ	21
4.1. ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ	21
4.2. ΔΕΔΟΜΕΝΑ ΑΥΤΟΚΙΝΗΤΩΝ	24
4.3. ΔΟΚΙΜΕΣ ΕΦΑΡΜΟΓΗΣ.....	25



1. ΕΙΣΑΓΩΓΗ

Το παρών έντυπο αποτελεί την τεκμηρίωση της απαλλακτικής άσκησης για το μάθημα “Κινητή Υπολογιστική και Εφαρμογές” με διδάσκον καθηγητή τον κ. Μενύχτα Ανδρέα.

2. ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ

Το αντικείμενο της εργασίας ήταν να φτιάξουμε μια εφαρμογή σε περιβάλλον Android με αντικείμενο της εφαρμογής της δικής μας επιλογής. Το αντικείμενο της εφαρμογής το οποίο αποφάσισα να υλοποιήσω ήταν για μία εφαρμογή όπου θα μπορεί ο χρήστης να καταχωρεί τα οχήματα τα οποία έχει καθώς και τις επισκευές που έχει κάνει σε αυτά.

3. ΑΝΑΛΥΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

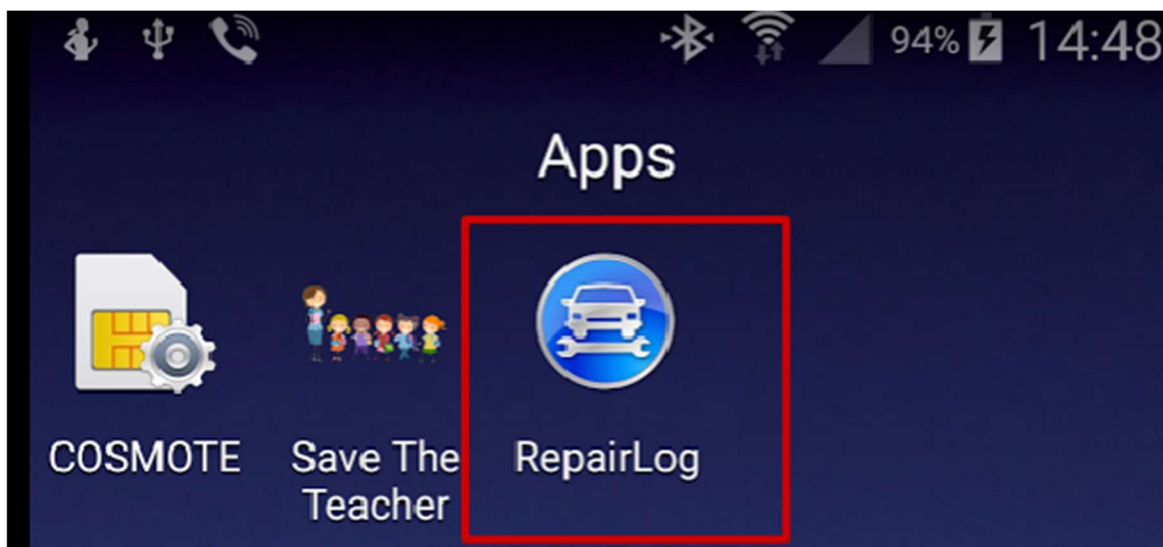
Ο κώδικας της εφαρμογής βρίσκεται στο [Github](https://github.com/tsotzolas/RepairLog) στον παρακάτω σύνδεσμο
<https://github.com/tsotzolas/RepairLog>

3.1. ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε [Android Studio](#) που είναι και το επίσημο εργαλείο ανάπτυξης που προτείνει και η Google για την ανάπτυξη εφαρμογών σε περιβάλλον Android.

3.2. ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

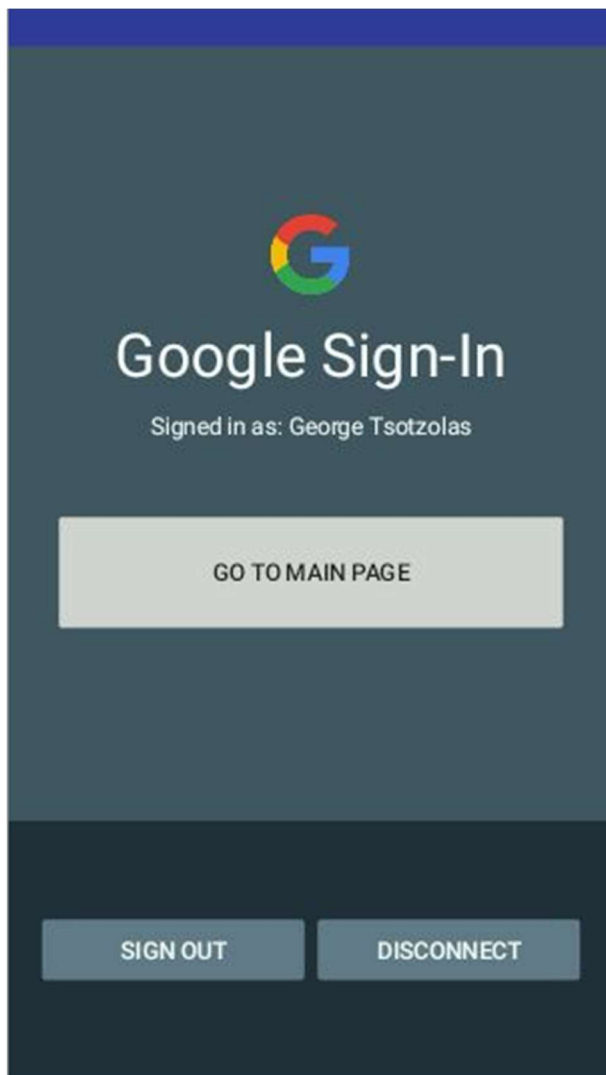
Το λογότυπο της εφαρμογής καθώς και το όνομά της όταν κάποιος την κάνει εγκατάσταση στη φορητή του συσκευή φαίνονται στη παρακάτω φωτογραφία.



Με το που την ανοίγει ο χρήστης την εφαρμογή τον μεταφέρει στην σελίδα να κάνει Google Sign In .

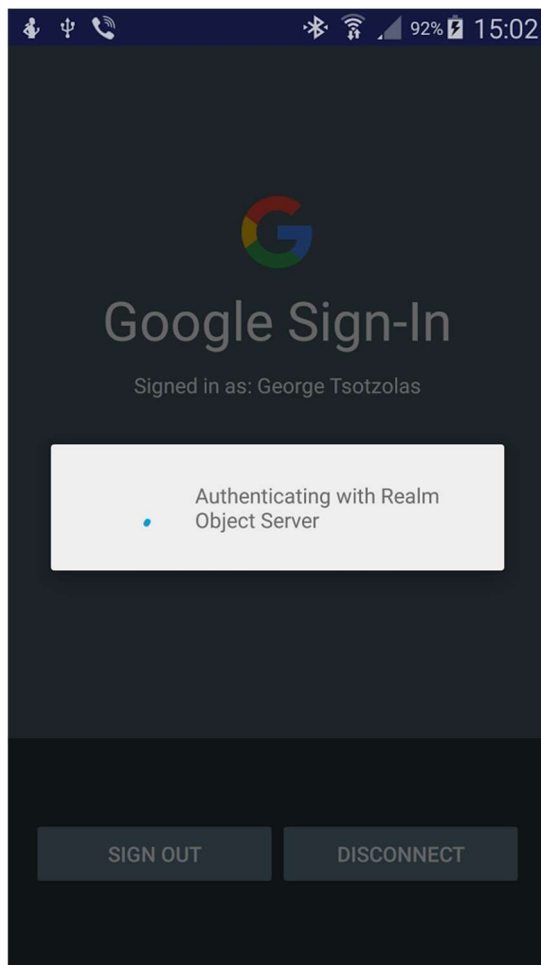


Και στη συνέχεια μόλις ολοκληρώσει το Google Sign In θα του εμφανίσει τις επιλογές να μεταβεί στην αρχική σελίδα της εφαρμογής



Εδώ βλέπουμε ότι του δίνεται και η δυνατότητα να κάνει κάνει και Google Sign Out καθώς και Disconnect.

Με το που πατήσει να μεταβεί ο χρήστης στην αρχική σελίδα γίνεται και ο συγχρονισμός των δεδομένων της εφαρμογής που βρίσκονται στην συσκευή του χρήστη με τα δεδομένα που είναι αποθηκευμένα στο cloud.



Στη συνέχεια εμφανίζεται η αρχική σελίδα της εφαρμογής




Να σημειωθεί σε αυτό το σημείο ότι η εφαρμογή είναι δίγλωσση , και αναγνωρίζει αν ο χρήστης έχει επιλεγμένη γλώσσα στο κινητό του τα Αγγλικά ή τα Ελληνικά και του εμφανίζει αντίστοιχα την γλώσσα. Φυσικά ο χρήστης μπορεί να αλλάξει και

αυτός την γλώσσα της εφαρμογής επιλέγοντας πάνω δεξιά τις  και στη

συνέχεια επιλέγοντας της επιλογή



Αρχικά ο χρήστης, από την αρχική οθόνη, θα πρέπει να επιλέξει  ώστε να μπορέσει να προσθέσει κάποιο όχημα. Πατώντας την επιλογή αυτή θα πρέπει να επιλέξει αν θέλει να καταχωρήσει κάποιο αυτοκίνητο ή κάποια μηχανή



Choose what do you want to insert car/Moto

CAR

MOTO

Έστω ότι θέλει να προσθέσει ένα αυτοκίνητο. Τότε θα του εμφανίσει να συμπληρώσει τα στοιχεία του οχήματος.



Please fill your's car details

Year ▼

Make ▼

Model ▼

CC ▼

ADD PHOTO SAVE


Θα πρέπει να επιλέξει πρώτα την χρονολογία, μετά τη μάρκα του αυτοκινήτου , στη συνέχεια το μοντέλο και τέλος τα κυβικά του αυτοκινήτου. Ακόμα προαιρετικά μπορεί ο χρήστης να βάλει και μια φωτογραφία του αυτοκινήτου του , ειδάλλως θα μπει μια προεπιλεγμένη φωτογραφία. Να σημειωθεί ότι τα δεδομένα των αυτοκινήτων δεν ορίζονται από την εφαρμογή αλλά από εξωτερικό Rest API . Οπότε αν κάποιες καταχωρήσεις λείπουν δεν είναι από δική μου υπαιτιότητα.

Στην παρακάτω εικόνα φαίνεται πώς θα είναι όταν θα έχει συμπληρώσει όλα τα στοιχεία.



Please fill your's car details

Year	2002	▼
Make	Honda	▼
Model	Civic	▼
CC	2000	▼

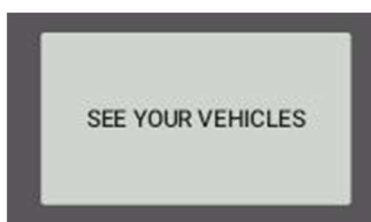


Να σημειωθεί ότι η εφαρμογή δεν αφήνει τον χρήστη να αφήσει κάποια από τα πεδία κενά και τον προτρέπει να τα συμπληρώσει.

Στη συνέχεια πρέπει να επιλέξει να αποθηκεύσει την καταχώρηση.





Μετά την αποθήκευση ο χρήστης μεταφέρεται στην αρχική σελίδα της εφαρμογής.

Επιλέγοντας την επιλογή να δει τα οχήματά του



μεταβαίνει στην λίστα με τα οχήματα.



Vehicle List View				
	Make	Audi	Model	80
	Year	1977	cc	1100
	Make	Honda	Model	Civic
	Year	2002	cc	2000
	Make	Fiat	Model	Tipo
	Year	1988	cc	1100
	Make	Ktm	Model	300 exc
	Year	2016	cc	300

Επιλέγοντας πάνω σε κάποιο από αυτά βλέπει την λίστα με τις εργασίες που έχουν γίνει σε αυτό.

Repair List View	
<div>ADD NEW REPAIR</div> <div>DELETE VEHICLE</div>	
Date	15/6/2017
Km	10000
cost	50
Description	λάδια

Εκεί υπάρχουν και οι επιλογές άμα θέλει ο χρήστης να διαγράψει το όχημα ή να προσθέσει κάποια ακόμα επισκευή.

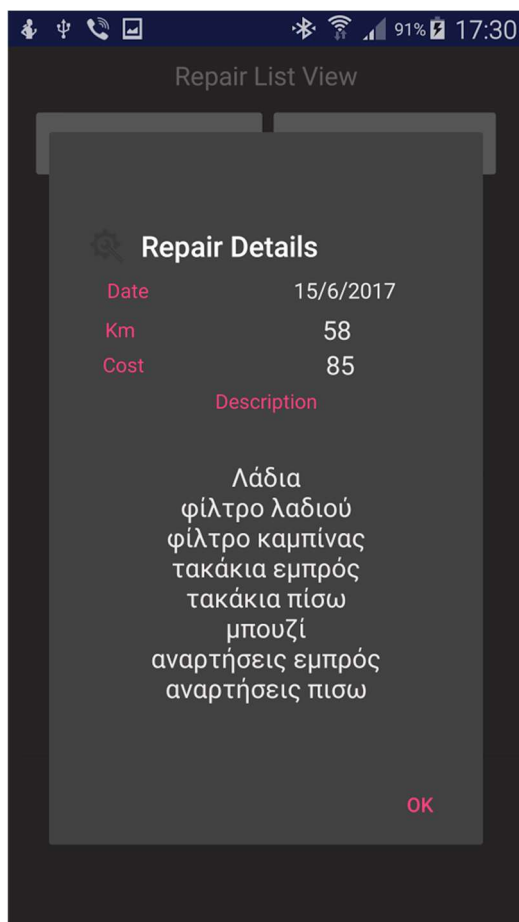


Κινητή Υπολογιστική και Εφαρμογές

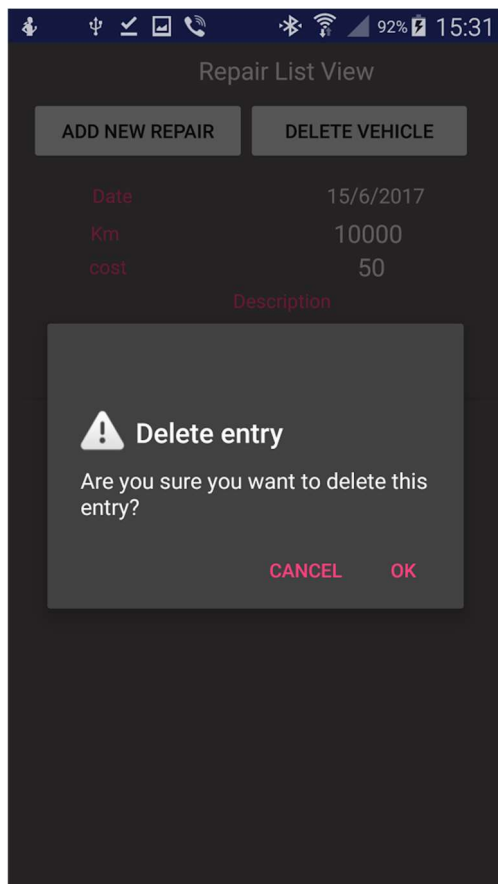
Στην περίπτωση που θέλει να διαγράψει το όχημα θα ερωτηθεί άμα θέλει να διαγράψει και θα προχωρήσει στην διαγραφή του οχήματος.

Στην περίπτωση που θέλει να προσθέσει μια νέα εργασία θα μεταβεί στην οθόνη της προσθήκης νέας εργασίας.

Στην περίπτωση που θέλει να δει την επισκευή την επιλέγει και μπορεί να την δει ολόκληρη



Στην περίπτωση όπου θέλει ο χρήστης να διαγράψει μια εργασία θα πρέπει να πατήσει παρατεταμένα πάνω στην εργασία και τότε θα ερωτηθεί άμα θέλει να διαγράψει την συγκεκριμένη καταχώρηση.



Η σελίδα εισαγωγής μιας νέας εργασίας είναι όπως παρακάτω.

Date

Km

Cost

Description

SAVE



Κινητή Υπολογιστική και Εφαρμογές

Θα πρέπει ο χρήστης να επιλέξει την ημερομηνία που έγινε η καταχώρηση, τα χιλιόμετρα του οχήματος καθώς και το κόστος και την περιγραφή της εργασίας.

The screenshot shows a mobile application interface. On the left, a date picker is open, displaying 'Wednesday 21 JUN 2017'. The calendar shows June 2017 with the 21st selected. On the right, a form is visible with the following fields: 'Date' (21/6/2017), 'Km' (15852), 'cost' (25), and 'Description' (Αλλαγή τακάκια εμπρός). A 'SAVE' button is at the bottom right.

Και στο τέλος να αποθηκεύσει την καταχώρηση.

The screenshot shows the 'Repair List View' of the application. It has two buttons at the top: 'ADD NEW REPAIR' and 'DELETE VEHICLE'. Below, there is a list of repairs. The first repair has a date of 15/6/2017, 10000 km, cost of 50, and description 'λάδια'. The second repair has a date of 21/6/2017, 15852 km, cost of 25, and description 'Αλλαγή τακάκια εμπρός'.



Η ίδια διαδικασία γίνεται αν θέλει ο χρήστης να κάνει εισαγωγή κάποια μηχανή απλά στην οθόνη που πρέπει να επιλέξει αν θέλει να δημιουργήσει μια εισαγωγή για αυτοκίνητο ή μηχανή πρέπει να επιλέξει τη μηχανή.

Choose what do you want to insert car/Moto

CAR MOTO

Please fill your's moto details

Year 2016

Make Ktm

Model 300exc

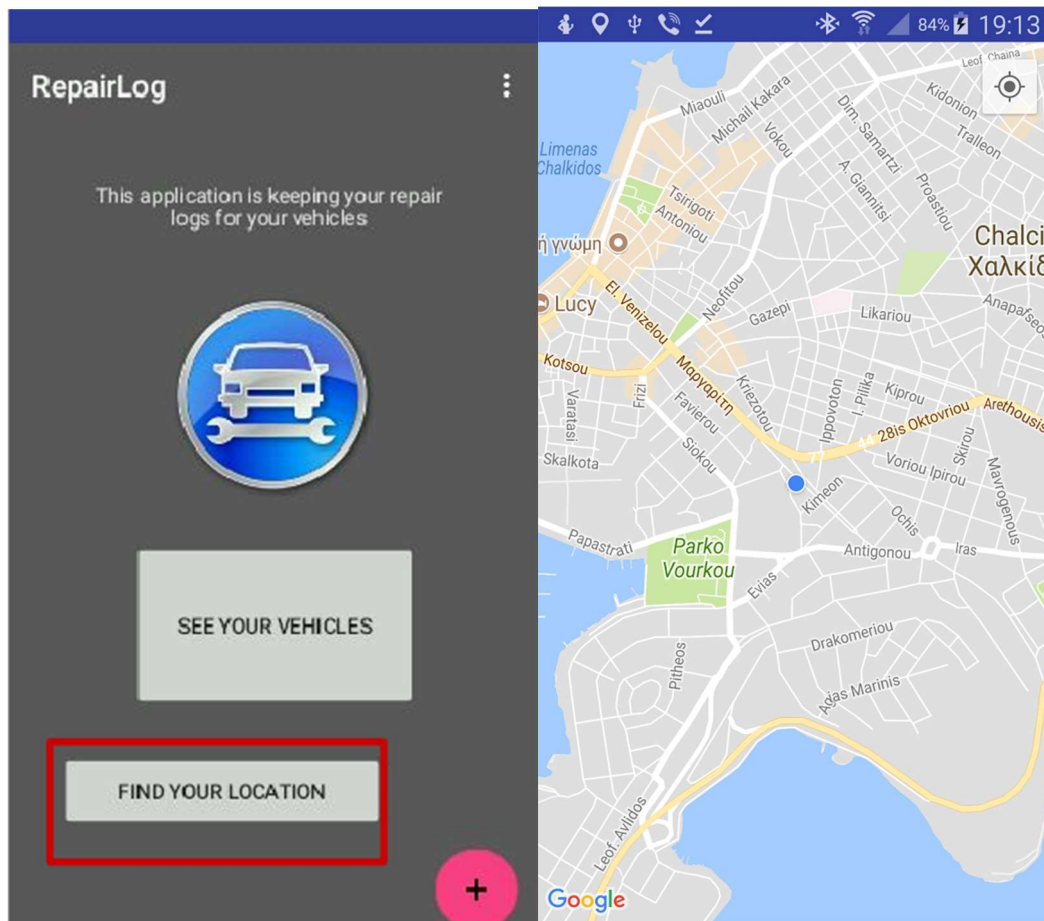
CC 300

ADD PHOTO

SAVE

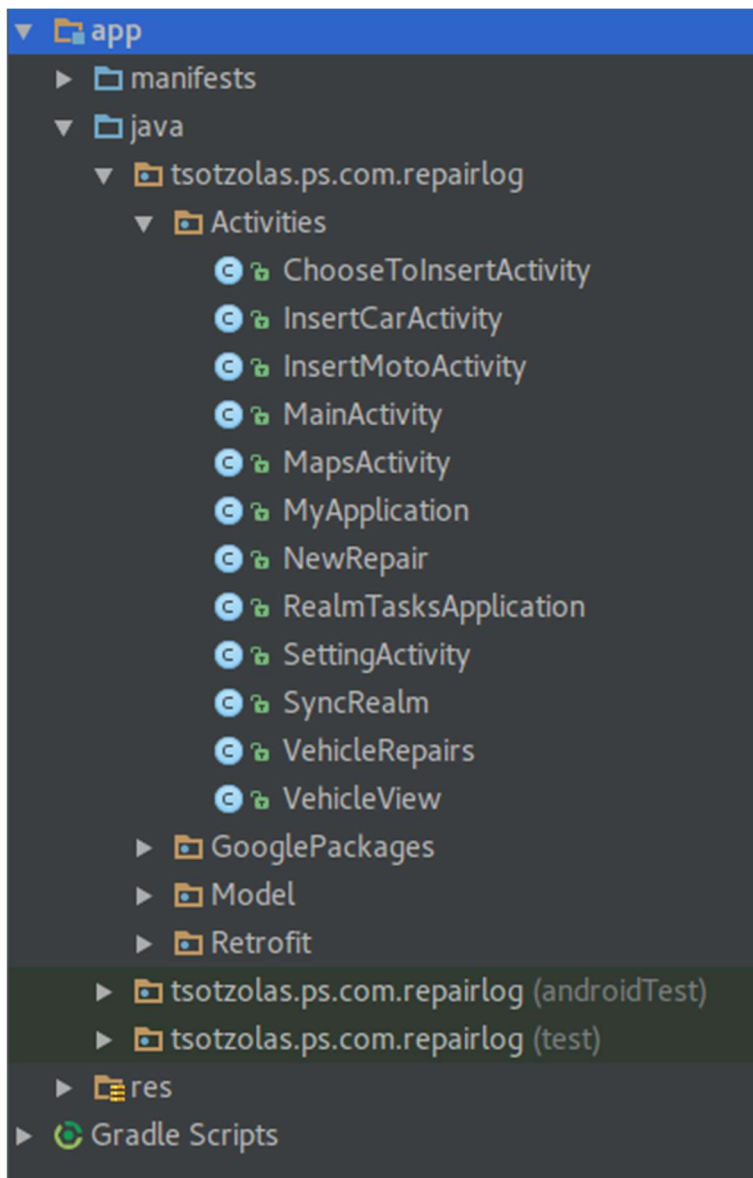
Μια διαφορά σε σχέση με το αμάξι είναι ότι ο χρήστης θα πρέπει να πληκτρολογήσει όλα τα πεδία για την καταχώρηση. Δεν βρήκα κάποιο αντίστοιχο REST API για της μηχανές.

Τέλος από την αρχική οθόνη ο χρήστης μπορεί να δει την τοποθεσία του στο χάρτη , αφού έχει ενεργοποιήσει πριν το GPS της φορητής του συσκευής , πατώντας στην επιλογή



3.3. ΔΟΜΗ ΤΟΥ ΚΩΔΙΚΑ

Στο σημείο αυτό κρίνεται σκόπιμο να αναλυθεί η δομή του κώδικα, ώστε να είναι πιο οικείο σε κάποιον να προηγηθεί σε αυτόν



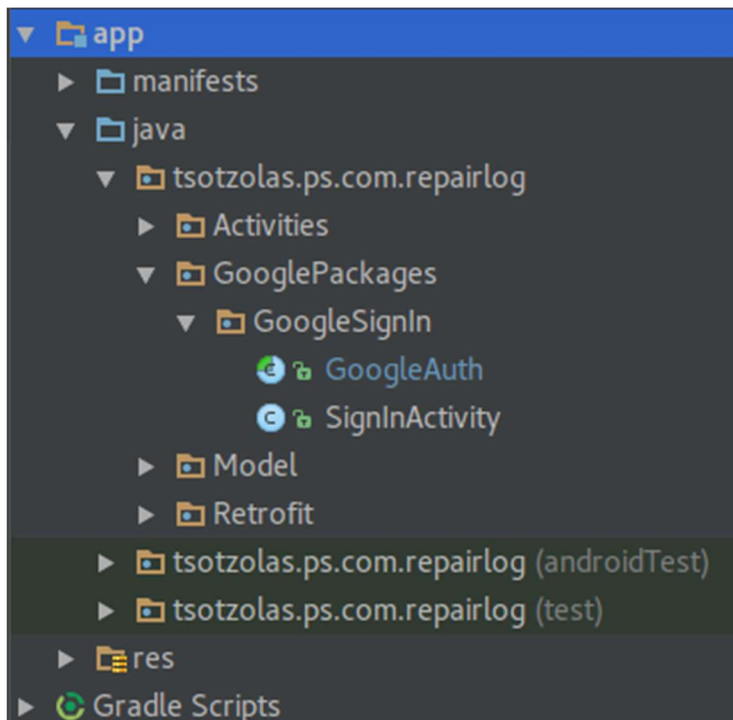
Για τον Java κώδικα έχω χωρίσει τα packages με τέτοιο τρόπο ώστε να είναι πιο ευδιάκριτα στο χρήστη. Γενικά στον κώδικα υπάρχουν αρκετά σχόλια ώστε να μπορεί κάποιος να καταλάβει τι γίνεται κάθε φορά.

Package Activities

Στο package αυτό έχουν μπει σχεδόν όλα τα activities που αφορούν τις κύριες λειτουργίες της εφαρμογής. Η ονομαστιά τους είναι χαρακτηριστική και μπορεί κάποιος να καταλάβει τι κάνει το καθένα.

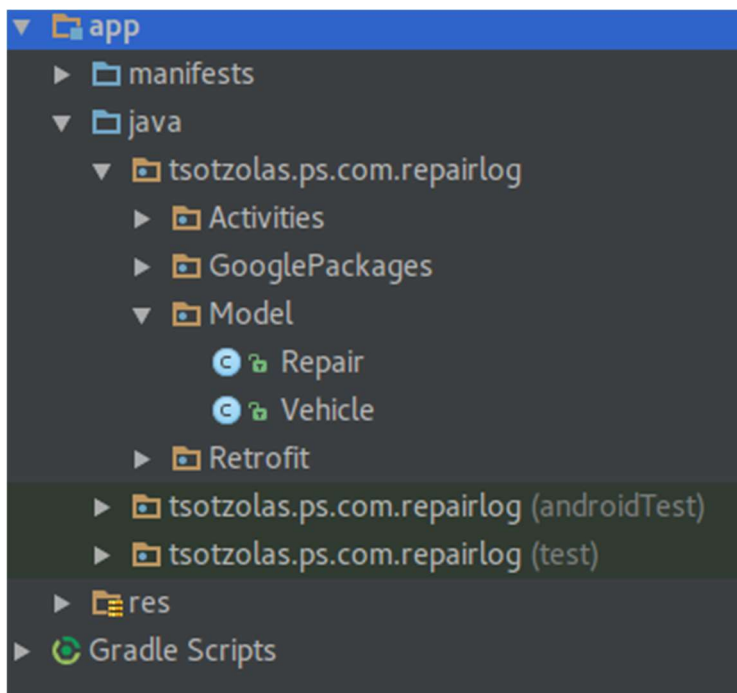


Package GooglePackages



Στο package αυτό έχει μπει ο κώδικας ο οποίο είναι για το Google Sign In.

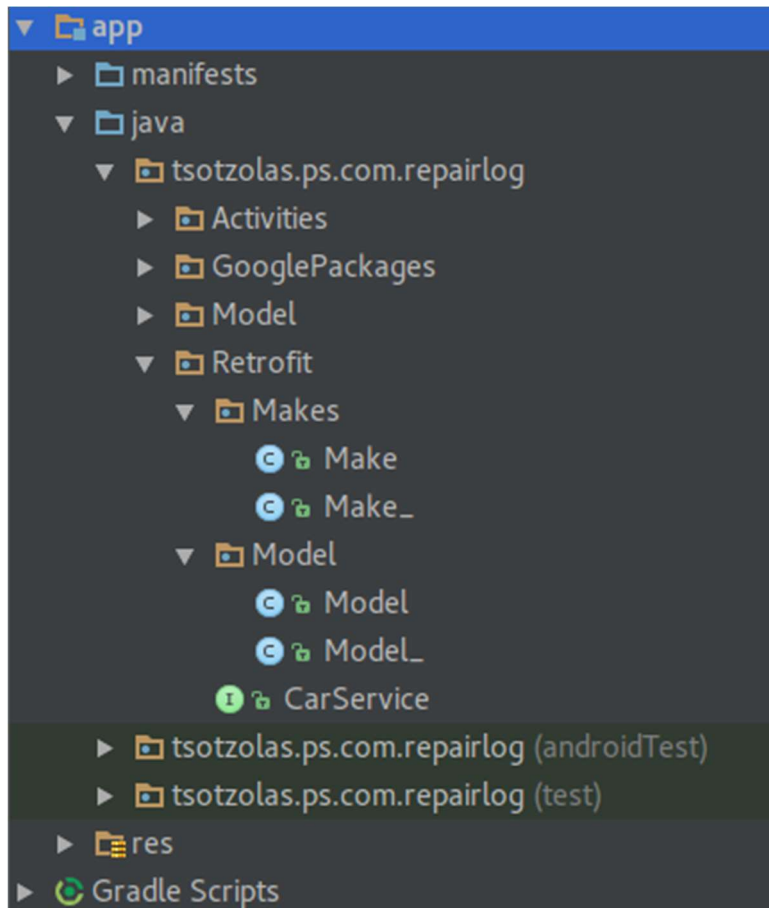
Package Model





Στο package αυτό βρίσκονται τα μοντέλα των δεδομένων της εφαρμογής μας που είναι τα οχήματα (Vehicles) και οι επισκευές (Repair).

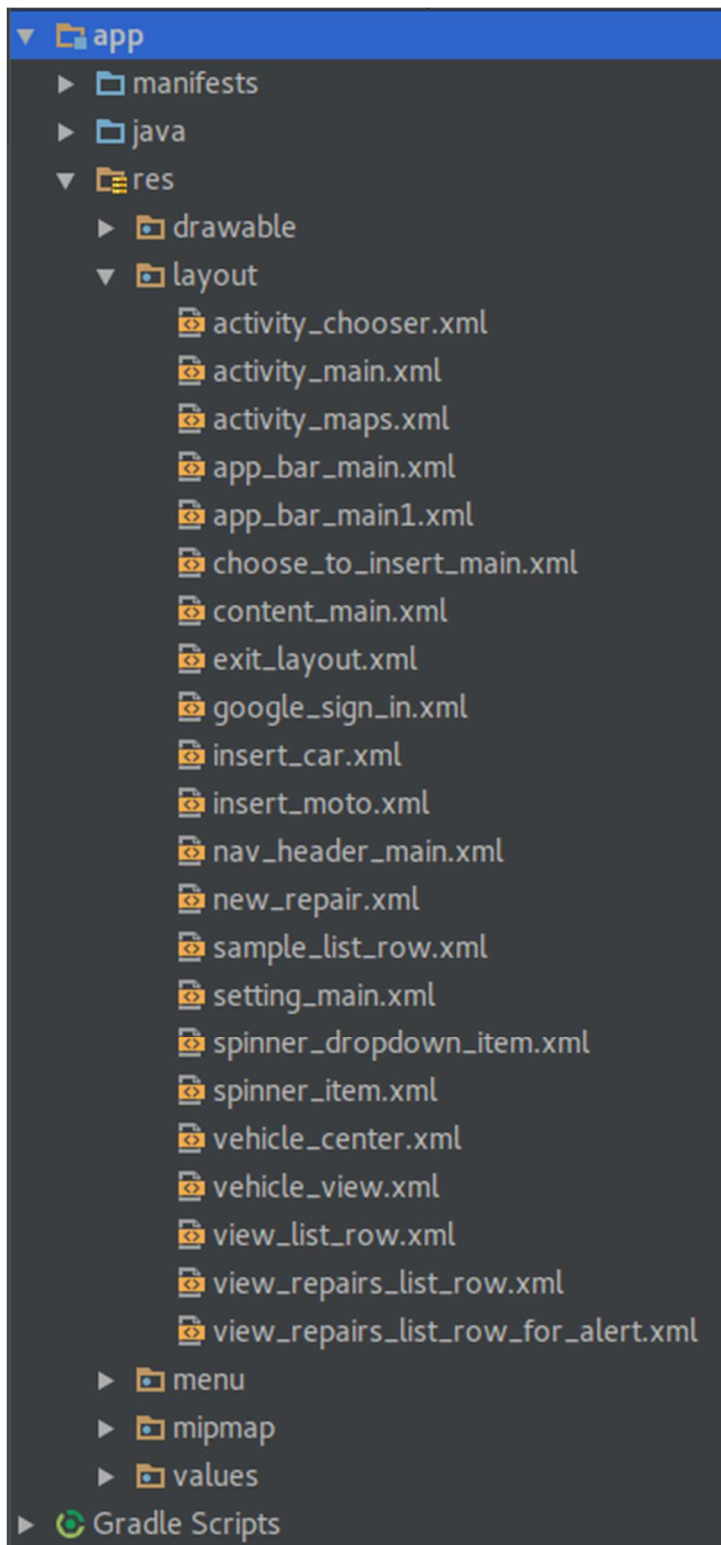
Package Retrofit



Στο package αυτό βρίσκονται τα αρχεία για το Retrofit που χρησιμοποιούμε για να φέρουμε τις μάρκες και τα μοντέλα των αυτοκινήτων από το REST API.



Φάκελος Layout



Στον φάκελο αυτό βρίσκονται όλα τα xml τα οποία χρησιμοποιήθηκαν για το UI της εφαρμογής.



4. ΣΗΜΕΙΑ ΙΔΙΑΙΤΕΡΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟΥ ΕΝΔΙΑΦΕΡΟΝΤΟΣ

Στην παράγραφο αυτή θα ήθελα να αναφέρουμε κάποια σημεία με ιδιαίτερο προγραμματιστικό ενδιαφέρον που έχει η εφαρμογή.

4.1. ΑΠΟΘΗΚΕΥΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ

Για την αποθήκευση των δεδομένων έχει χρησιμοποιηθεί για βάση το [Realm](#). Η βάση είναι τοπικά και μπορεί να λειτουργήσει και χωρίς τη χρήση του διαδικτύου. Αυτό όμως που έχει ιδιαίτερο ενδιαφέρον είναι ότι έχουμε υλοποιήσει συγχρονισμό των δεδομένων μας με τη χρήση του [Realm Object Server](#).



Το σχήμα δείχνει ακριβώς πώς λειτουργεί ή όλη διαδικασία και είναι μια Real Time Database. Ο Realm Object Server που έχει στηθεί, βρίσκεται σε ένα VM στον [Okeanos](#) σε ένα Ubuntu 16.04 λειτουργικό σύστημα .





Το διαχειριστικό UI του Real Object Server φαίνεται όπως παρακάτω



Μπορείτε να το δει κάποιος στο παρακάτω URL

<http://83.212.105.36:9080/>

Με username: amenychtas@unipi.gr

Και password: amenychtas@unipi.gr

Για την υλοποίηση αυτό χρειάζεται να δημιουργηθεί ένας χρήστης στον Realm Object Server.

Αυτό γίνεται στον κώδικα αμέσως μετά το Google Sign Login του χρήστη . Αφού γίνει το Google Sign Login , χρησιμοποιούμε το Google email του χρήστη σας username και το Google Id για password και φτιάχνουμε τον χρήστη

```
private void handleSignInResult(GoogleSignInResult result) {
    Log.d(TAG, "handleSignInResult:" + result.isSuccess());
    if (result.isSuccess()) {
        // Signed in successfully, show authenticated UI.
        acct = result.getSignInAccount();
        String username = "";
        String password = "";
        //Κάνουμε έναν έλεγχο άμα ο χρήστης έχει το συγκεκριμένο
        email να του το αλλάξουμε
        // γιατί με το email αυτό είναι ο λογαριασμός του Database
        Administrator στο Realm Object Server
        if (acct != null) {
            if ("tsotzolas@gmail.com".equals(acct.getEmail())) {
                username = "tsotzol@gmail.com";
            } else {
                //σαν username στο Realm βάζουμε το email του χρήστη
                username = acct.getEmail();
            }
            //Σαν password στο Realm βάζουμε το google id
            password = acct.getId();
        }
    }
}
```



```
}

mStatusTextView.setText(getString(R.string.signed_in_fmt,
acct.getDisplayName()));
Realm.init(getApplicationContext());
//Αφού ο χρήστης κάνει google Sign In μετα φτιάχνουμε χρήστη
στο Realm Object Server

SyncUser.loginAsync(SyncCredentials.usernamePassword(username, password,
true), AUTH_URL, new SyncUser.Callback() {
    @Override
    public void onSuccess(SyncUser user) {
//        registrationComplete(user);
        Toast.makeText(SignInActivity.this, "Create User in
Realm Object Server", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onError(ObjectServerError error) {
        String errorMsg;
        switch (error.getErrorCode()) {
            case EXISTING_ACCOUNT:
                errorMsg = "Account already exists";
                break;
            default:
                errorMsg = error.toString();
        }
    }
});
updateUI(true);
```

Στη συνέχεια όταν ο χρήστης πατήσει να πάει στην αρχική οθόνη κάνουμε login του χρηστη στον Realm Object Server. Σε αυτό το σημείο έχουμε βάλει μια χρονοκαθυστέρηση των 5 sec ώστε να προλάβει η βάση του Realm να συγχρονίσει με τον Realm Object Server.

```
public void gotoMain(View view) {
    showProgressDialog1();

    //Κάνουμε Συγχρονισμό του Realm με τον Realm Object Server
    SyncRealm.realmSync();

    //Καθυστερούμε την όλη διαδικασία για να προλάβει να κάνει
    συγχρονισμό το Realm
    Handler handler = new Handler();
    handler.postDelayed(new Runnable() {

        @Override
        public void run() {
            hideProgressDialog1();
            Intent ki = new Intent(SignInActivity.this,
MainActivity.class);
            startActivity(ki);
        }
    }, 5000);
}
```




```
}, 5000); // 5000ms delay  
}
```

4.2. ΔΕΔΟΜΕΝΑ ΑΥΤΟΚΙΝΗΤΩΝ

Για να μπορέσει να συμπληρώσει ο χρήστης τα δεδομένα των αυτοκινήτων του χρησιμοποιήσα ένα REST API που βρήκα στο διαδίκτυο και βρίσκεται στην παρακάτω διεύθυνση <https://www.carqueryapi.com/>

Για να μπορέσουμε να πάρουμε τα δεδομένα χρησιμοποιήσαμε το [Retrofit](#) για να μπορέσουμε να κάνουμε επικοινωνήσουμε με το API. Ενδεικτικά παραθέτω τον κώδικα όπου καλούμε και παίρνω τα δεδομένα από τις μάρκες των αυτοκινήτων.

```
//Καλεί για να πάρει απο το API τα δεδομένα σύμφωνα με το έτος που  
έχουμε επιλέξει  
CarService carService = retrofit.create(CarService.class);  
carService.getMake("getMakes", year, "0").enqueue(new  
Callback<Make>() {  
    @Override  
    public void onResponse(Call<Make> call, Response<Make> response)  
    {  
        if (response.isSuccessful()) {  
            makeList = response.body();  
            makeListString1 = new ArrayList<String>();  
            //Βάζουμε αυτό στην αρχή για να ξέρουμε αν έχει επιλέξει  
            κάτι ο χρήστης ή όχι  
            if (!makeListString1.contains(".....")) {  
                makeListString1.add(".....");  
            }  
            //Γεμίζουμε τη list με τα makes  
            for (int i = 0; i < makeList.getMakes().size(); i++) {  
                makeListString1.add(makeList.getMakes().get(i).getMakeDisplay());  
            }  
  
            //Καλούμε για να γεμίσουμε τις makes  
            fillListMakes();  
            hideProgressDialog();  
        } else {  
            Log.e(TAG, "Failed. Status: " + response.code());  
            Log.i(TAG, "----->" + call.request().url());  
        }  
    }  
}
```



4.3. ΔΟΚΙΜΕΣ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή δόθηκε για δοκιμή σε άλλους προγραμματιστές αλλά και σε απλούς χρήστες οι οποίοι μας επισήμαναν διάφορα λειτουργικά προβλήματα καθώς και λεκτικά σφάλματα που είχα. Ακόμα διαπιστώθηκαν διάφορες δυσλειτουργίες που είχε η εφαρμογή σε διαφορετικές συσκευές. Τα σφάλματα που προέκυπταν έρχονται και στο crash report του [Firebase](#)

