Σημειώσεις για το μάθημα "Τεχνολογία και Προγραμματισμός Κινητών Συσκευών" χρησιμοποιώντας το Android Studio.

Σημειώσεις Android

Ιωάννης Έλληνας

Athens 2018

Author Name

Σημειώσεις Android

Ιωάννης Έλληνας

Πρώτη Έκδοση: Απρίλιος 2018

Περιεχόμενα

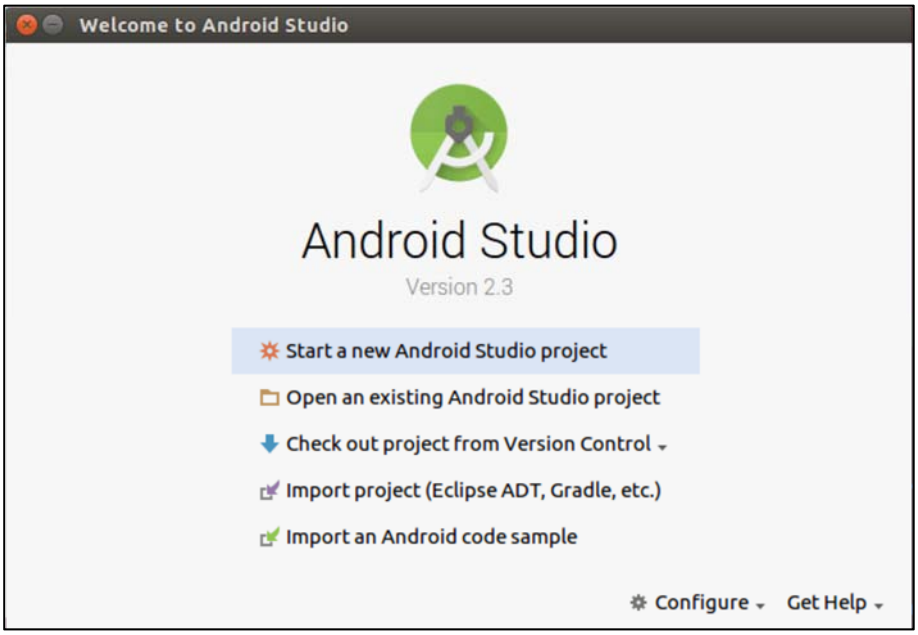
Android Studio

Εγκατάσταση

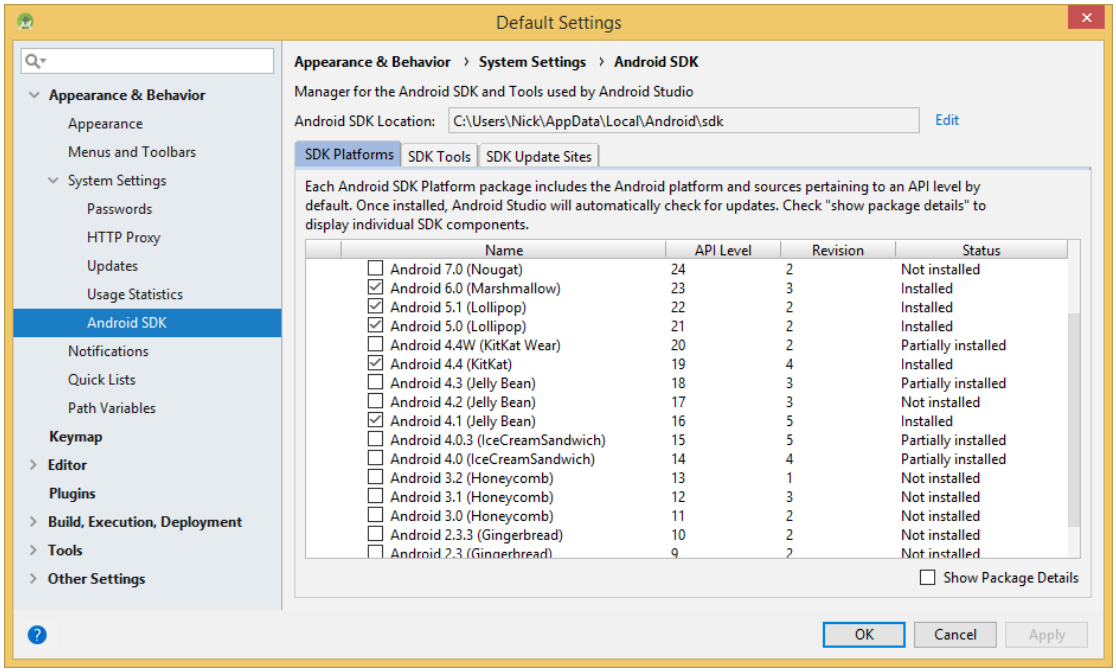
- Κατεβάστε και εγκαταστήστε το Android Studio:

  http://developer.android.com/sdk/index.html

- Η οθόνη αμέσως μετά την εγκατάσταση είναι:



- Στο κάτω μέρος της οθόνης πατήστε το Configure → Check for Updates και αν υπάρχουν ενημερώσεις προχωρήστε σε αυτές.
- Κάθε έκδοση Android έχει το δικό της SDK (Software Development Kit), το οποίο επιτρέπει την εγκατάσταση του αντίστοιχου εξομοιωτή AVD (Android Virtual Device), ο οποίος μπορεί να είναι μα κινητή συσκευή, ένα tablet, μια τηλεόραση, και στον οποίο θα τρέξει η εφαρμογή μας. Για την ενσωμάτωση διαφόρων εκδόσεων Android πατάμε **Configure → SDK Manager** και εμφανίζονται όλες οι δυνατές εκδόσεις και ποιες από αυτές είναι ήδη εγκατεστημένες. Καλό είναι να εγκαταστήσετε την τελευταία έκδοση καθώς επίσης και την έκδοση που έχει το κινητό σας επειδή σε αυτό θα τρέχουν οι εφαρμογές που θα αναπτύσσονται. Αν για παράδειγμα το κινητό σας έχει την έκδοση Android 5.1 (Lollipop ή API 22), τότε τσεκάρετε το αντίστοιχο κουτάκι και πατάτε το πλήκτρο "OK" κάτω δεξιά.



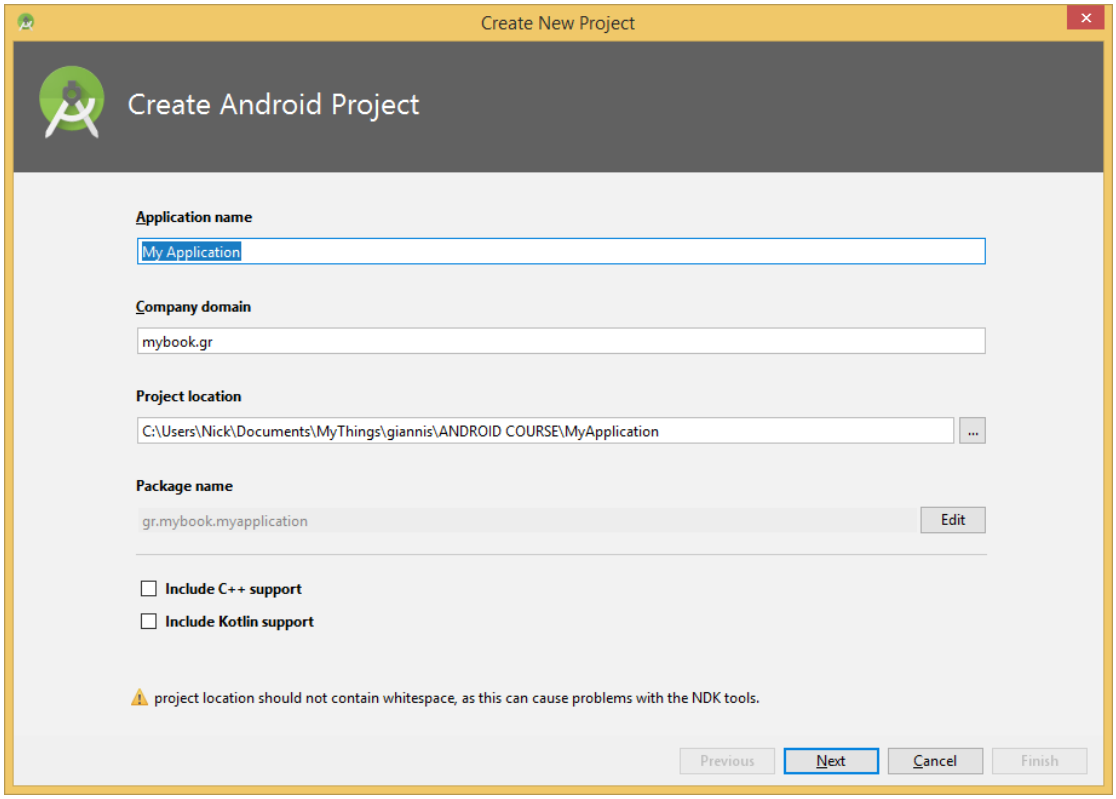
Δημιουργία νέου project

Τα βήματα δημιουργίας ενός νέου project είναι τα εξής:

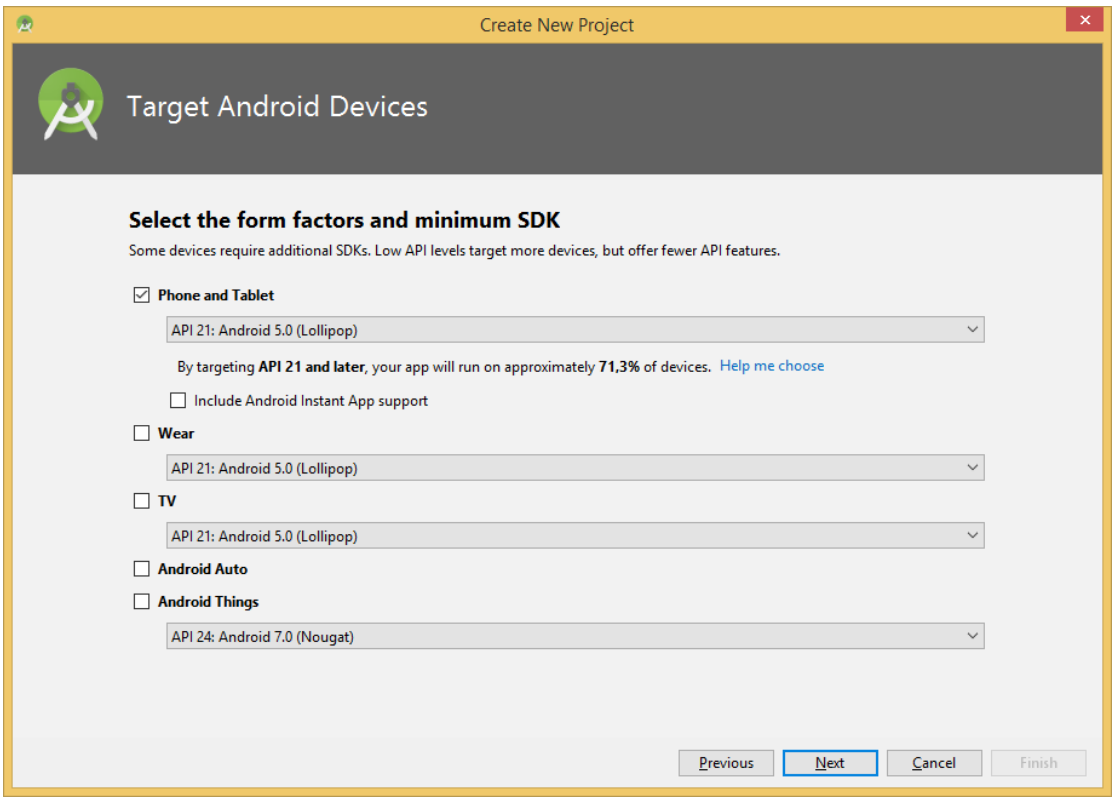
- Από την οθόνη εισόδου του Android Studio επιλέγουμε:

  Start a new Android Studio Project

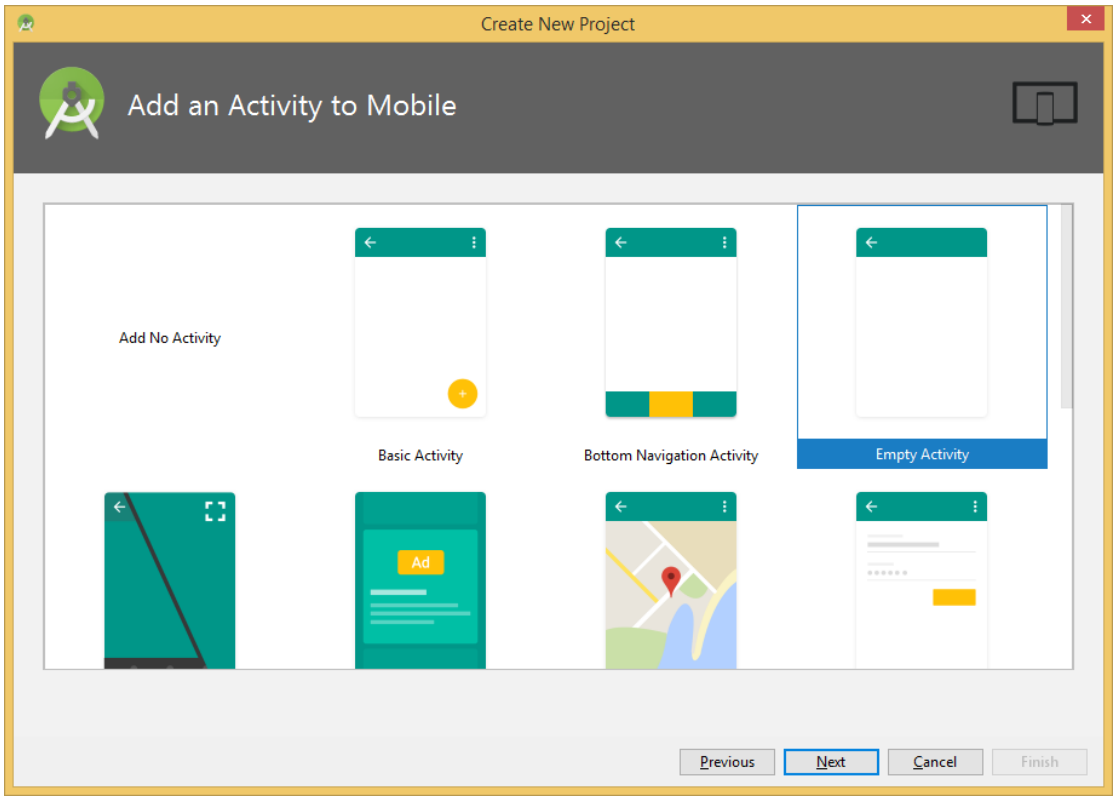
- Αν δεν υπάρχει η οθόνη εισόδου επειδή έχετε ήδη ανοικτό κάποιο project, τότε επιλέγετε **File → New → New Project**.
- Τοποθετούμε το όνομα της εφαρμογής:
  Application name: MyFirstProject
  Company domain: mybook.gr
- Τοποθετούμε το workspace οπουδήποτε στο δίσκο μας:
  Project location: C:\Users\……



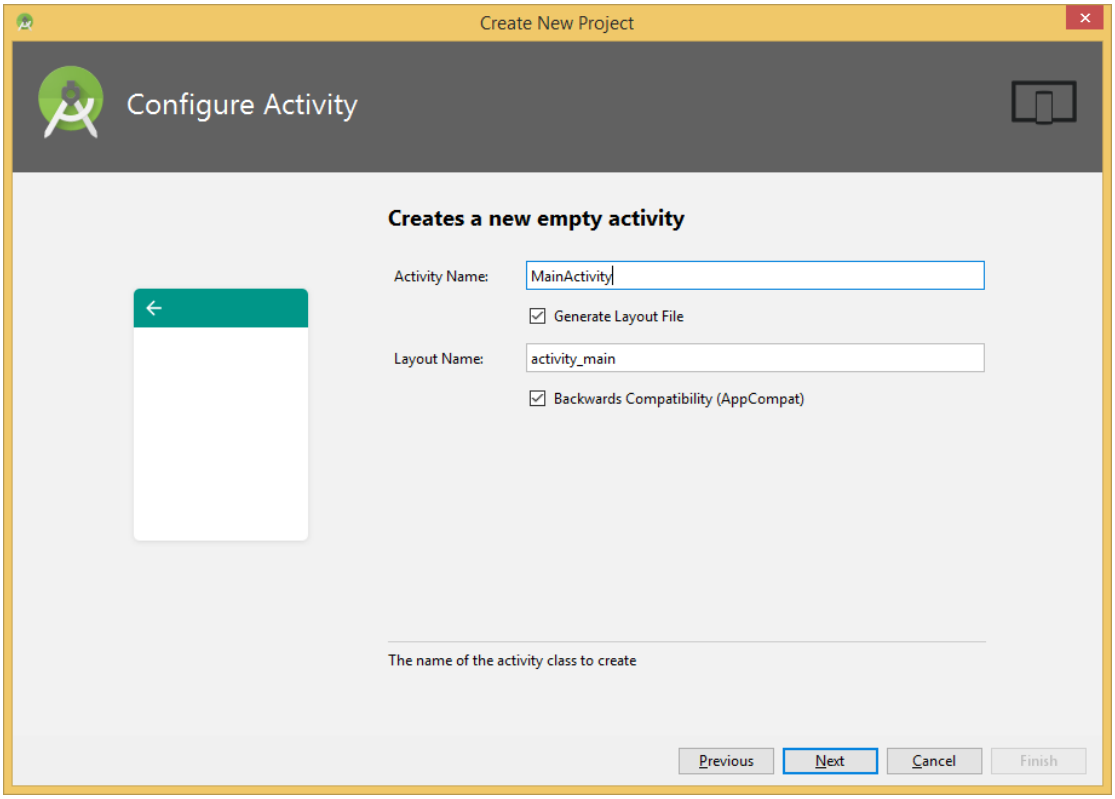
- Στην επόμενη οθόνη καθορίζουμε το ελάχιστο API που θα τρέχει η εφαρμογή μας. Θέτοντας το API 21 παρατηρούμε ότι θα τρέχει στο 71,3% των συσκευών.



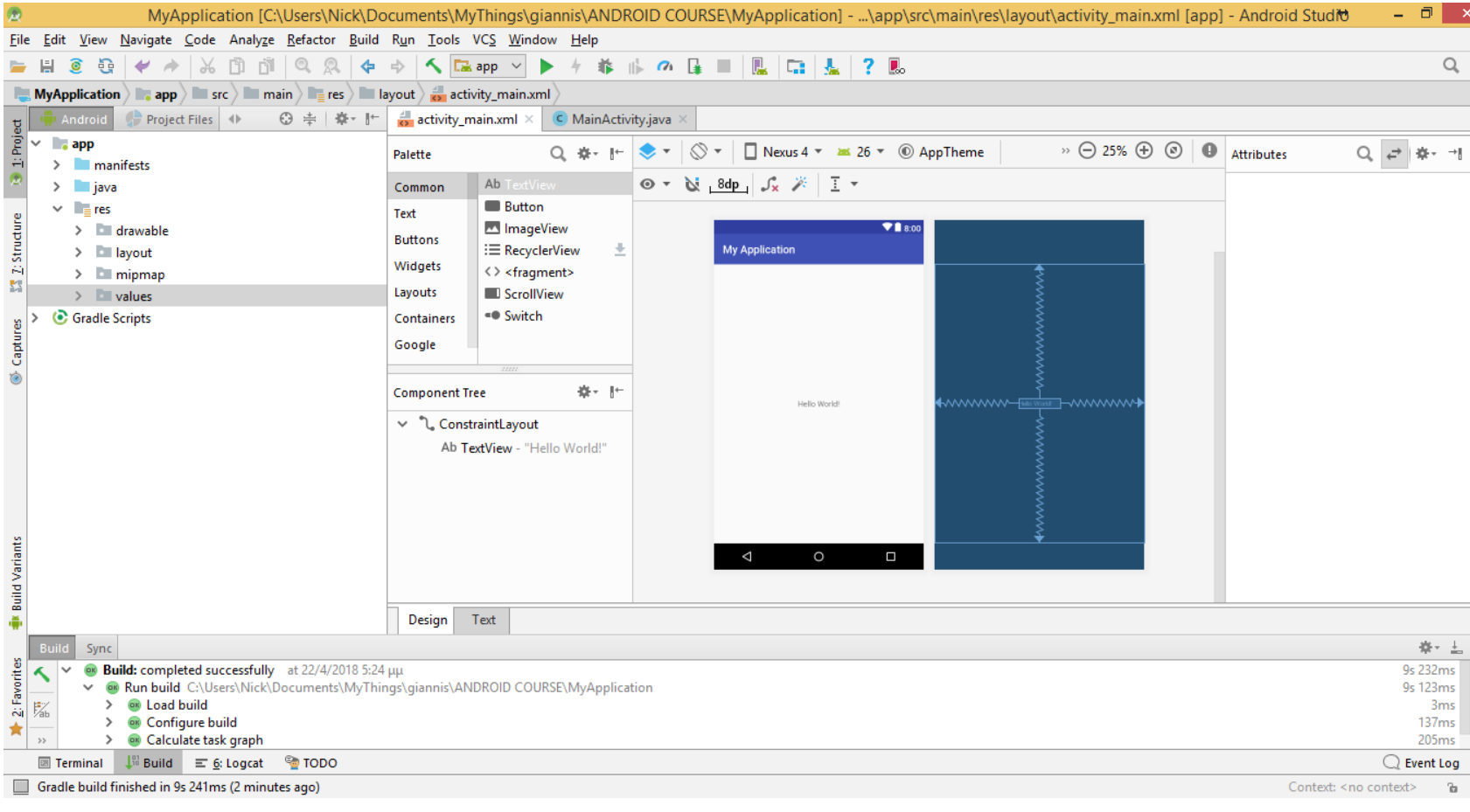
- Στην επόμενη οθόνη καθορίζουμε τη μορφή της δραστηριότητας με την οποία θα ξεκινήσει η εφαρμογή (ουσιαστικά τη μορφή που θα παρουσιάσει η οθόνη μας). Συνήθως επιλέγουμε το **Empty Activity**, η οποία είναι μια κενή δραστηριότητα στην οποία εμείς καθορίζουμε την εμφάνιση με κάποιο αρχείο xml.



- Στην επόμενη οθόνη παρουσιάζονται οι εξής προ-τοποθετημένες τιμές:
  Activity Name: MainActivity → Το όνομα της υποκλάσης με την οποία θα ξεκινήσει η εκτέλεση της εφαρμογής.
  Layout Name: activity_main → Το όνομα του xml αρχείου το οποίο καθορίζει την εμφάνιση της εφαρμογής στην οθόνη.



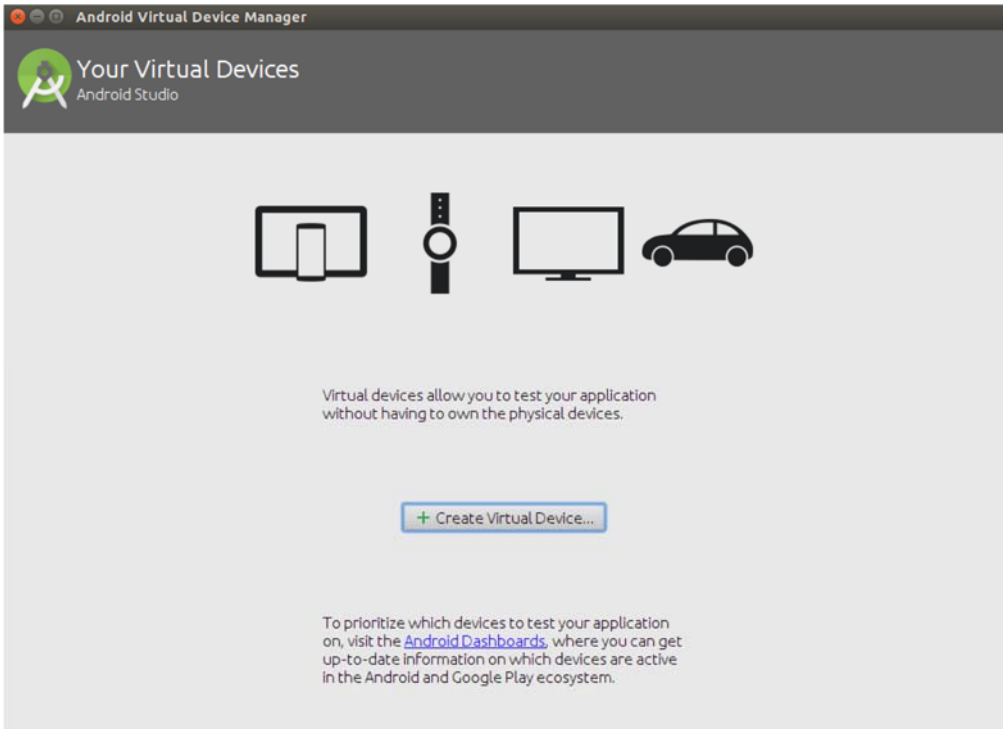
- Με το πλήκτρο **Next** και στη συνέχεια **Finish** ολοκληρώνεται η δημιουργία νέου project και στην οθόνη μας έρχεται η οθόνη ανάπτυξης των εφαρμογών.


MyApplication [C:\Users\Nick\Documents\MyThings\giannis\ANDROID COURSE\MyApplication] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication app src main res layout activity_main.xml
Android
Project Files
app
manifests
java
res
drawable
layout
mipmap
values
Gradle Scripts
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Component Tree
ConstraintLayout
TextView - "Hello World!"
Nexus 4
26
AppTheme
25%
8dp
My Application
Hello World!
Attributes
Design
Text
Build
Sync
Build: completed successfully at 22/4/2018 5:24 μμ
Run build C:\Users\Nick\Documents\MyThings\giannis\ANDROID COURSE\MyApplication
Load build
Configure build
Calculate task graph
9s 232ms
9s 123ms
3ms
137ms
205ms
Terminal
Build
6: Logcat
TODO
Event Log
Gradle build finished in 9s 241ms (2 minutes ago)
Context: <no context>
1: Project
7: Structure
Captures
Build Variants
2: Favorites
Gradle
Device File Explorer

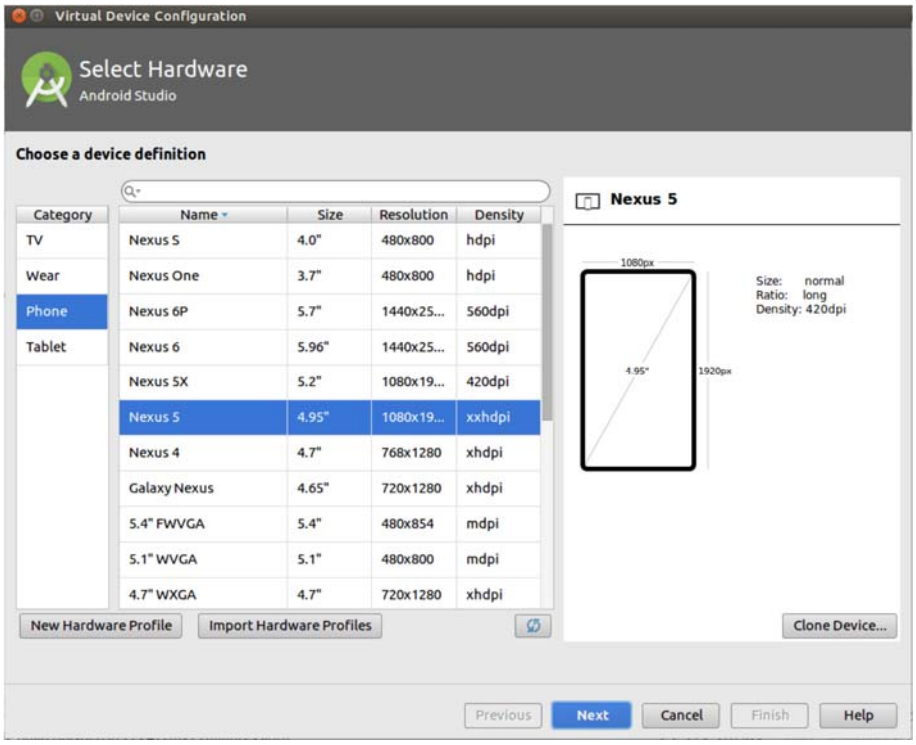
Στο αριστερό μέρος παρουσιάζεται η δομή της εφαρμογής μας. Στο δεξιό μέρος παρουσιάζονται τα MainActivity.java και το περιγραφικό αρχείο activity_main.xml. Στη δεξιά άκρη εμφανίζονται οι ιδιότητες των widgets του περιγραφικού αρχείου.
Αν για παράδειγμα πατήσουμε πάνω στη λέξη "Hello World", θα δούμε δεξιά ότι είναι ένα πεδίο κειμένου (Text View) το οποίο μπορεί να βρίσκεται είτε στο activity_main.xml ή στο αρχείο strings.xml.

AVD Manager

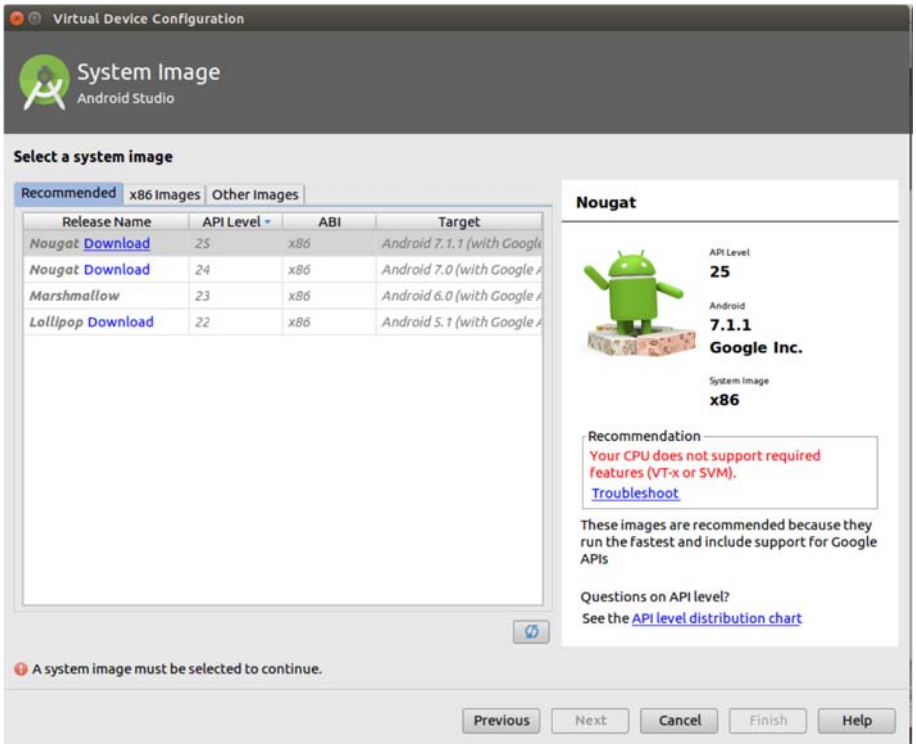
- Με την επιλογή **Tools→ AVD Manager** παράγεται η οθόνη δημιουργίας του εξομοιωτή της κινητής συσκευής.



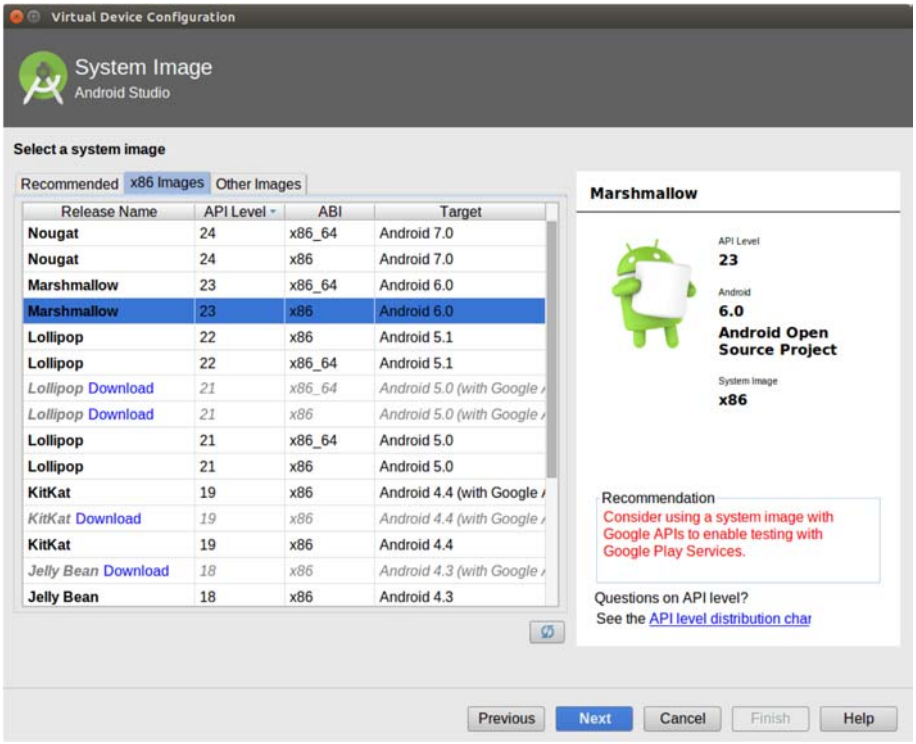
- Επιλέγουμε "Create Virtual Device".



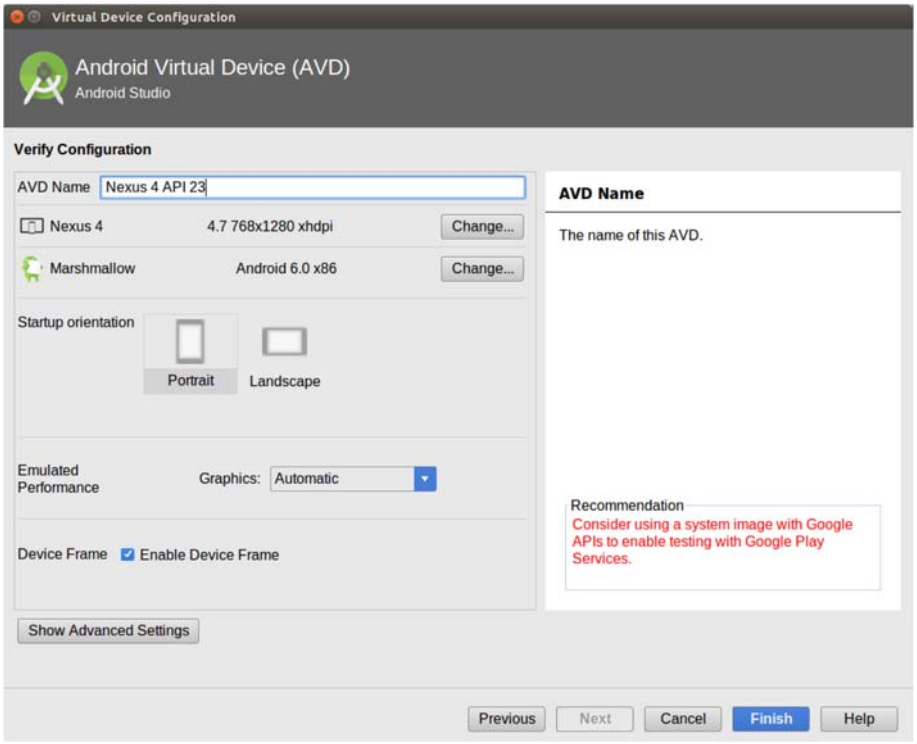
- Επιλέγουμε "Phone" και μια έτοιμη συσκευή ή δημιουργούμε μια δική μας με το "New Hardware Profile".
- Επειδή η ταχύτητα εξαρτάται από την ανάλυση της οθόνης του εξομοιωτή, επιλέγουμε μια μεσαία ανάλυση όπως 800X480 ή 1280X768.



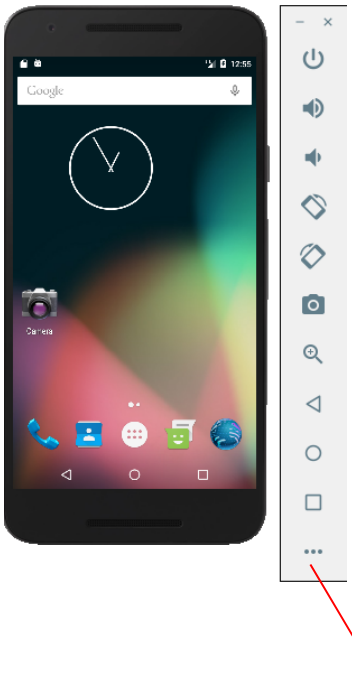
- Επιλέγουμε κάποιο λειτουργικό με API level από 15 και επάνω. Αν υπάρχει η ένδειξη "Download" δίπλα σε αυτό που θα επιλέξουμε τότε κατεβάζουμε το συγκεκριμένο API.

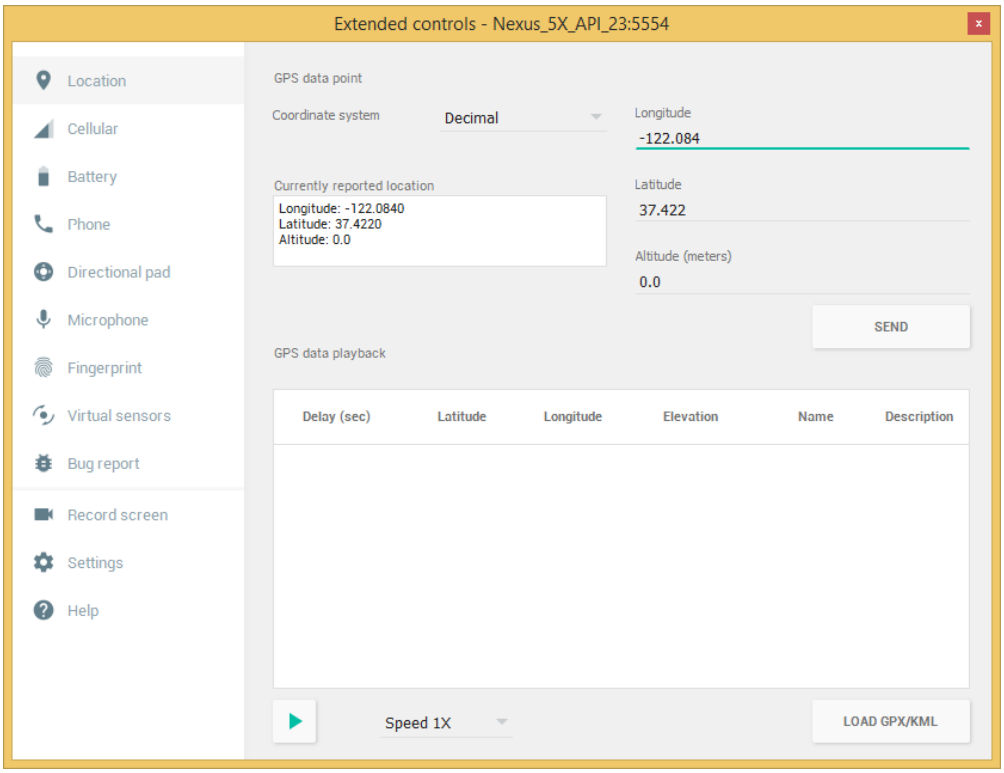


- Στη συνέχεια δημιουργούμε ένα όνομα για αυτόν τον εξομοιωτή.



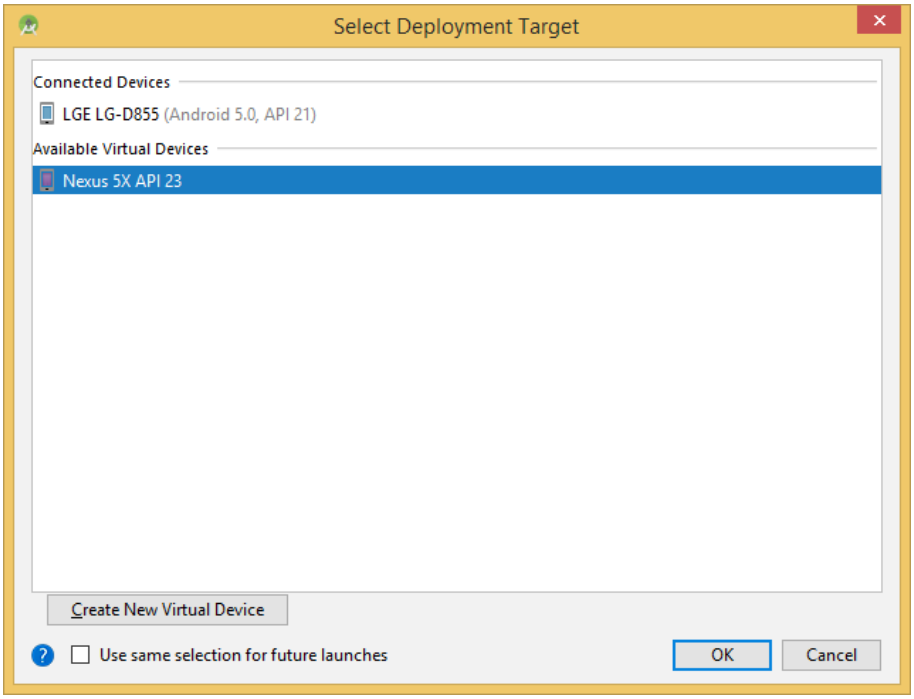
- Εκκινώντας αυτόν τον εξομοιωτή, παρουσιάζεται στην οθόνη μας:




Extended controls - Nexus_5X_API_23:5554
Location
Cellular
Battery
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
Bug report
Record screen
Settings
Help
GPS data point
Coordinate system
Decimal
Longitude
-122.084
Currently reported location
Longitude: -122.0840
Latitude: 37.4220
Altitude: 0.0
Latitude
37.422
Altitude (meters)
0.0
SEND
GPS data playback
Delay (sec)
Latitude
Longitude
Elevation
Name
Description
Speed 1X
LOAD GPX/KML

Εκτέλεση

Κατά την εκτέλεση μιας εφαρμογής πρέπει να επιλέξουμε τον εξομοιωτή που θα τρέξει η εφαρμογή (όπως αυτός που δημιουργήθηκε με τον AVD Manager) ή κάποιο κινητό τηλέφωνο που είναι συνδεδεμένο στον υπολογιστή μας. Για να βλέπουμε την οθόνη του κινητού μας στην επιφάνεια εργασίας, κατεβάστε και τρέξτε την εφαρμογή "Vysor" η οποία είναι μια επέκταση του Google Chrome. Προτιμούμε τον τελευταίο τρόπο εκτέλεσης των προγραμμάτων για να αποφύγουμε τις καθυστερήσεις του εξομοιωτή.
Επίσης, μπορούμε να δημιουργήσουμε πιο γρήγορους εξομοιωτές που να μοιάζουν με το κινητό μας χρησιμοποιώντας την εφαρμογή Genymotion.



Μετά την επιλογή του εξομοιωτή, ο εξομοιωτής ανοίγει και τρέχει η εφαρμογή μας.

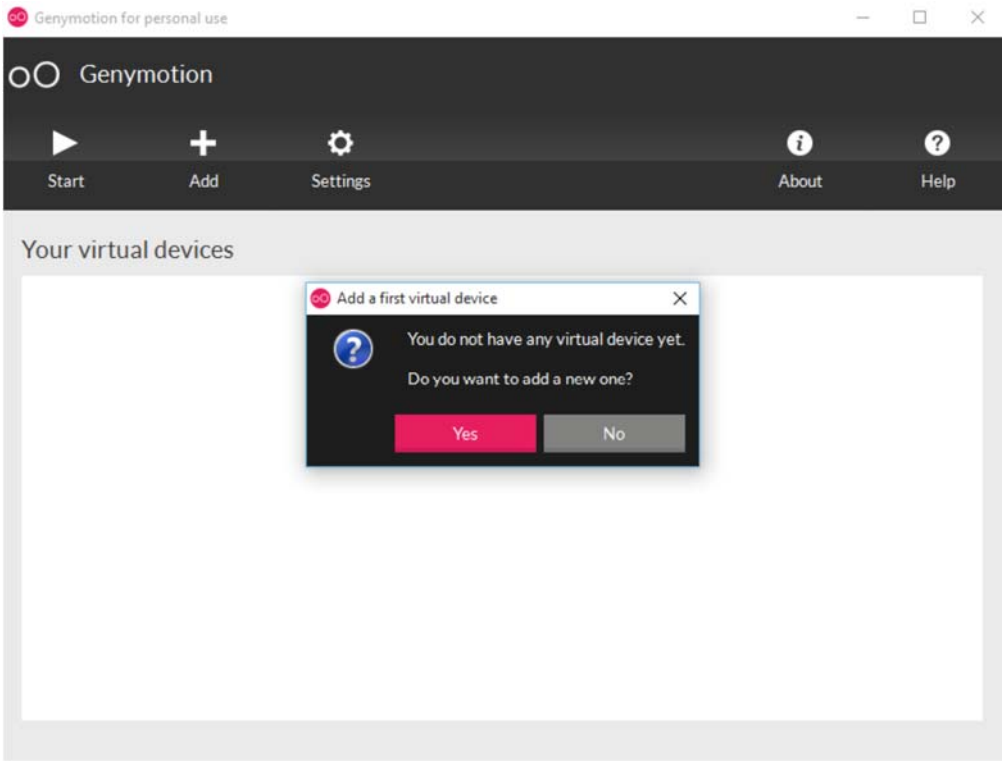
Παρατηρούμε ότι το όνομα της εφαρμογής στη μπάρα είναι **MyFirstProject** και το πεδίο κειμένου στην οθόνη της κινητής συσκευής είναι **Hello world!**

Φόρτωμα ήδη υπάρχουσας εφαρμογής

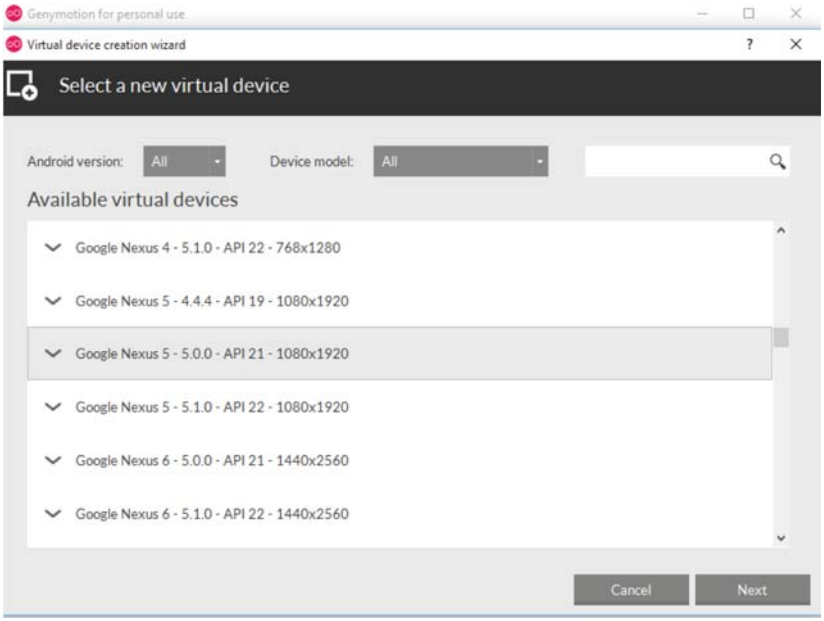
Εάν έχουμε μια εφαρμογή η οποία είχε δημιουργηθεί με τον προηγούμενο σύστημα ανάπτυξης (Eclipse-adt bundle), την κάνουμε Import στο Android Studio.
Από την αρχική οθόνη επιλέγοντας **Import project(Eclipse ADT, Gradle)** ή από μέσα επιλέγοντας **Import Ptroject**.

Εγκατάσταση του Genymotion

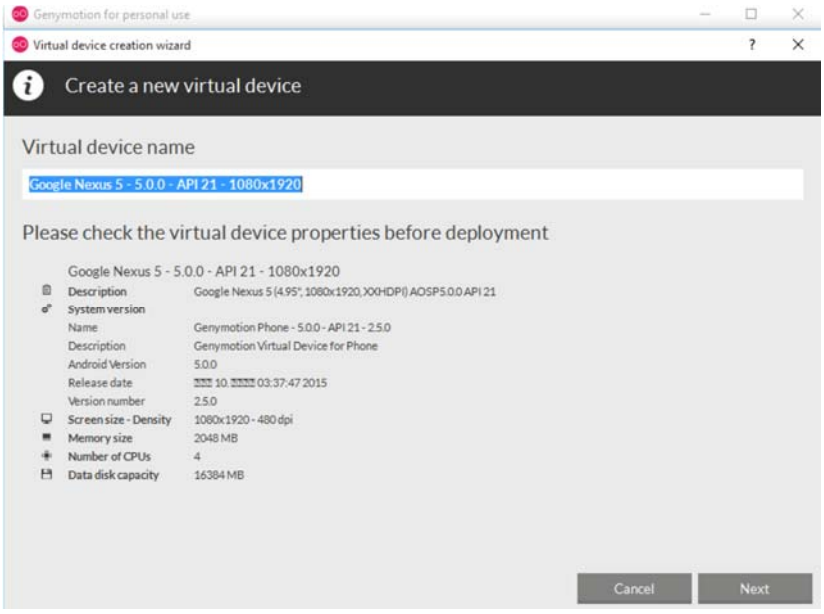
- Κατέβασμα από: https://www.genymotion.com/download/
- Μετά την εγκατάσταση, εάν ζητηθεί License Key, θα πρέπει να δηλώσετε ότι πρόκειται για Personal Use.
- Στη συνέχεια μπορούμε να δημιουργήσουμε κάποιο virtual device με το "Add".



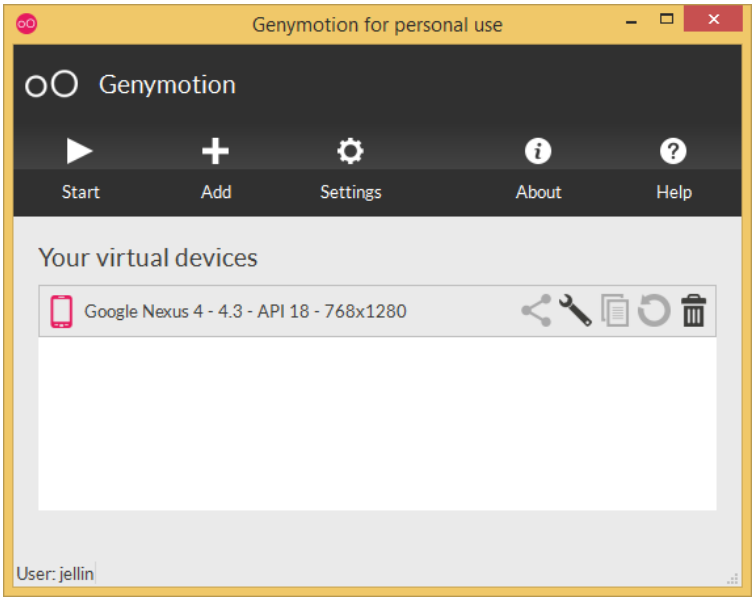
- Από τη λίστα που παρέχεται επιλέγουμε αυτό που θέλουμε (μην ξεχνάμε ότι για λόγους ταχύτητας η οθόνη δεν πρέπει να έχει μεγάλη ανάλυση).



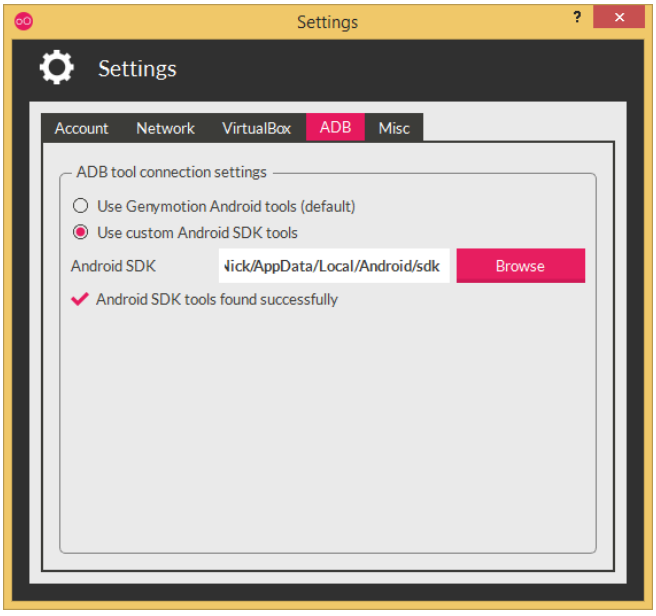
- Ολοκληρώνουμε τη δημιουργία της εικονικής συσκευής.



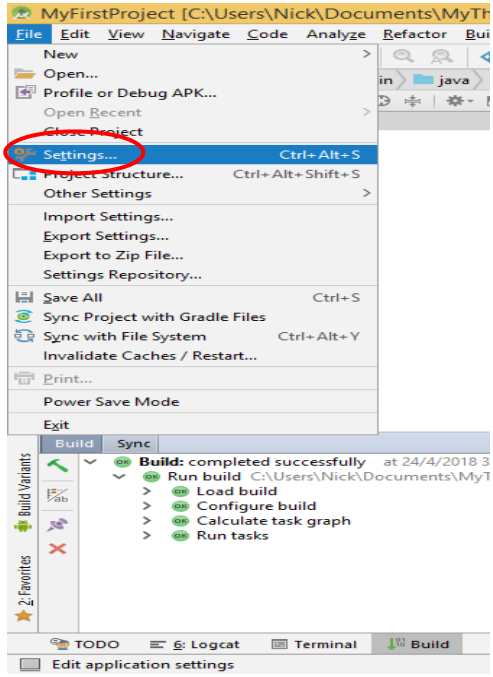
- Η εικονική συσκευή φορτώνεται και θα εμφανίζεται πλέον στην κεντρική οθόνη του Genymotion.



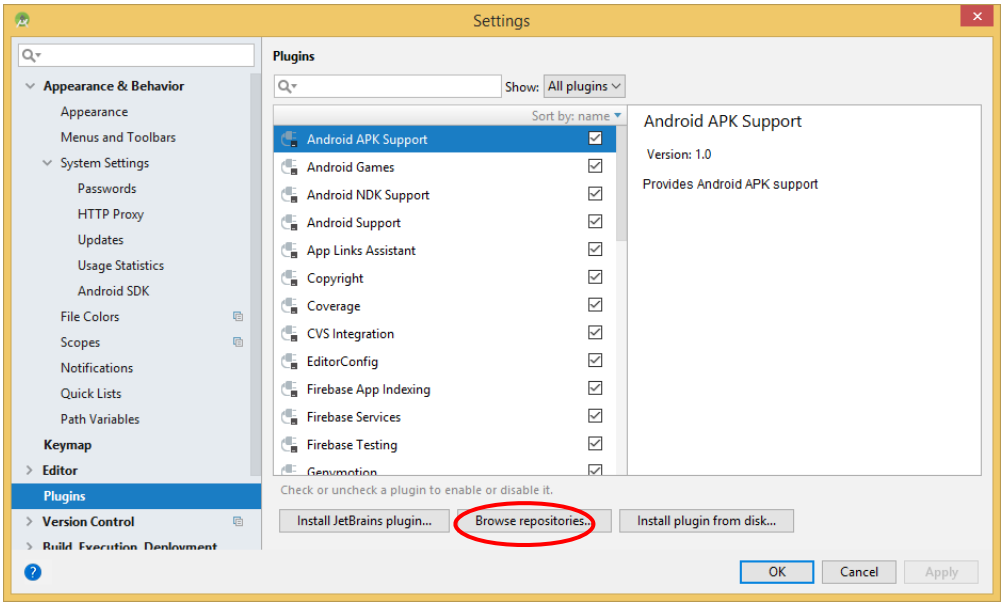
- Επιλέγουμε το πλήκτρο "Settings" → ADB → Use custom Android SDK tools, όπου πρέπει να ορίσουμε το path του Android SDK σε μορφή "C:\Users\Nick\AppData\Local\Android\sdk"



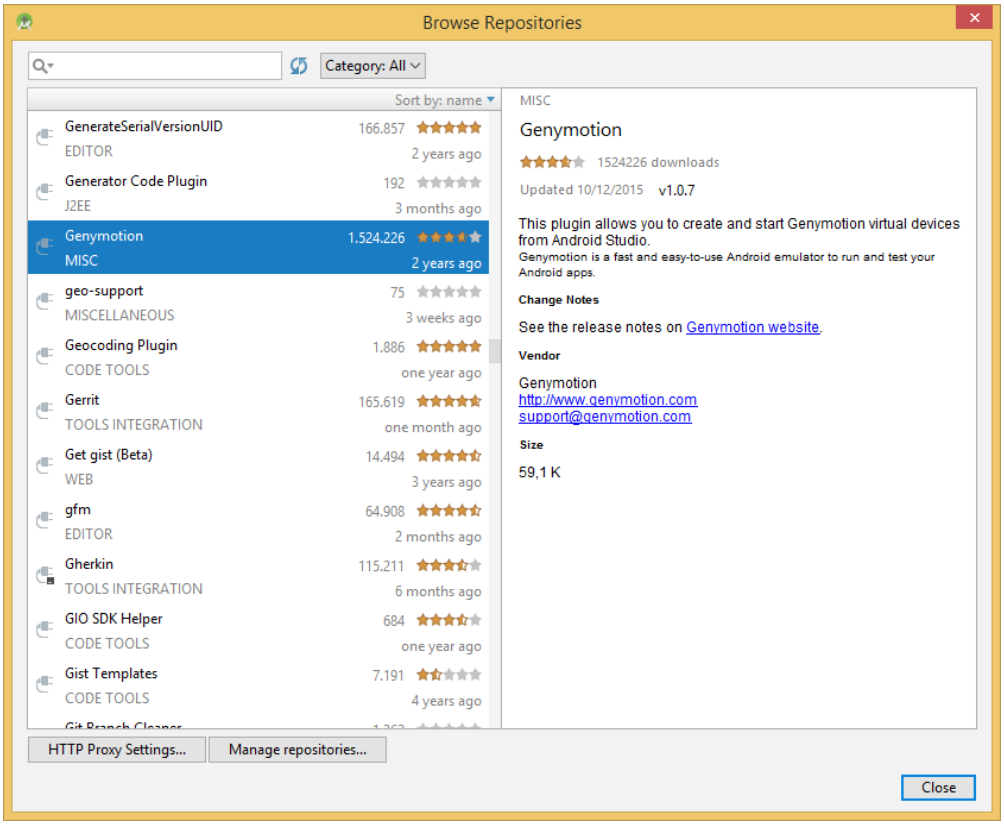
- Ανοίγουμε το Android Studio και επιλέγουμε File → Settings.



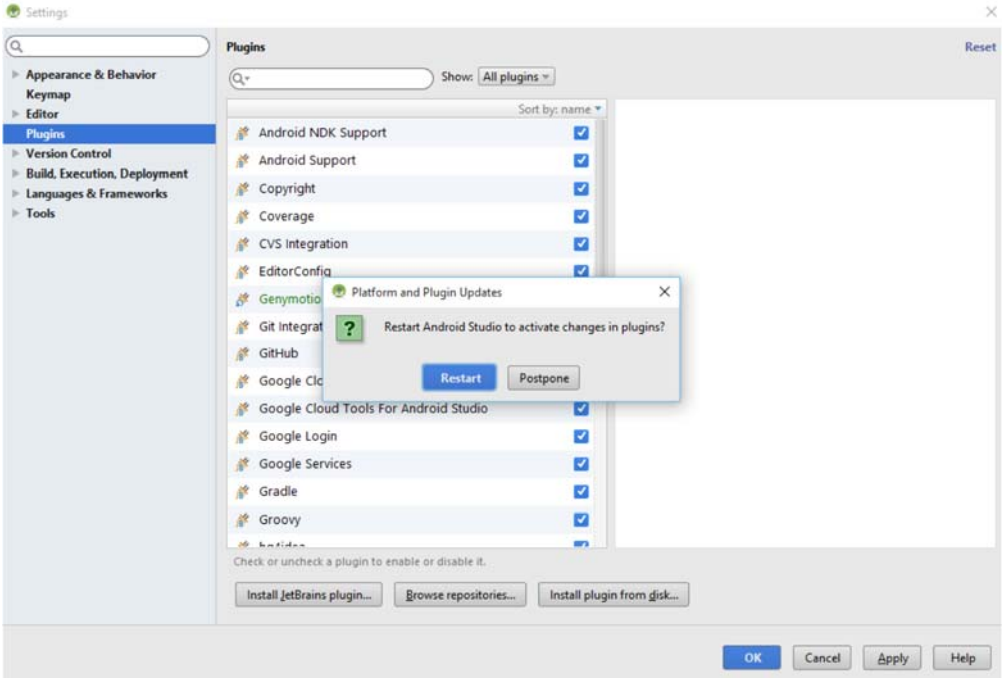
- Στο αναδυόμενο παράθυρο επιλέγουμε Plugins → Android NDK Support → Browse repositories.



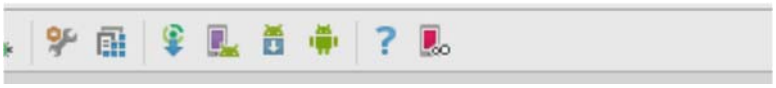
- Στη λίστα που ανοίγει επιλέγουμε το Genymotion και με δεξί κλικ κάνουμε "Download and Install".



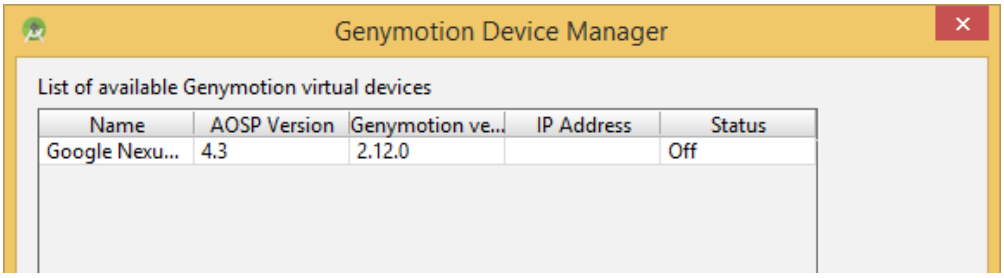
- Κάνουμε επανεκκίνηση του Android Studio.



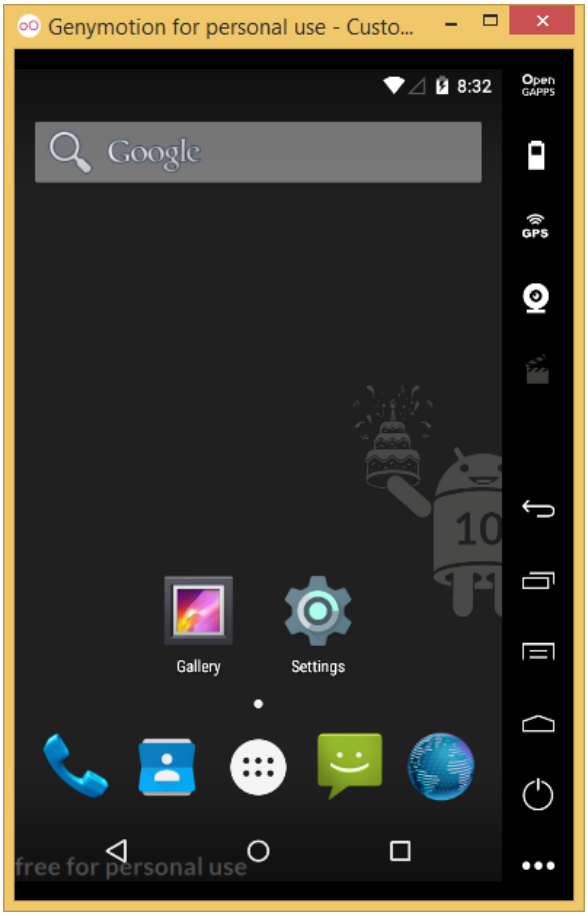
- Παρατηρούμε στη μπάρα συντομεύσεων το εικονίδιο του Genymotion.



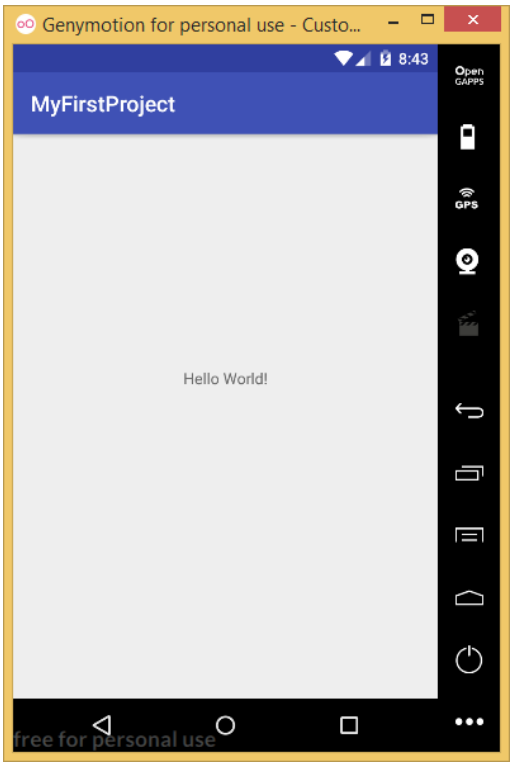
- Επιλέγουμε το εικονίδιο και μας ζητά να εισάγουμε το path του Genymotion, το οποίο πρέπει να είναι της μορφής: "C:\Program Files\Genymobile\Genymotion".
- Επιλέγοντας ξανά το ίδιο εικονίδιο πρέπει να εμφανίζεται η εικονική συσκευή.



- Την επιλέγουμε και πατάμε το πλήκτρο "Start".



- Η συσκευή αυτή θα φαίνεται σαν επιλογή κάθε φορά που θα τρέχετε μια εφαρμογή. Οι άλλες επιλογές είναι ο emulator του Android Studio και η κινητή μας συσκευή που είναι συνδεδεμένη στο USB του υπολογιστή.
- Εάν εκτελέσουμε την εφαρμογή μας πατώντας το "Run app" στη γραμμή εργαλείων του Android Studio, θα εμφανιστεί στον Genymotion emulator:


Genymotion for personal use - Custo...
8:43
MyFirstProject
Hello World!
Open GAPPS
GPS
free for personal use

Περιήγηση στο Android Studio

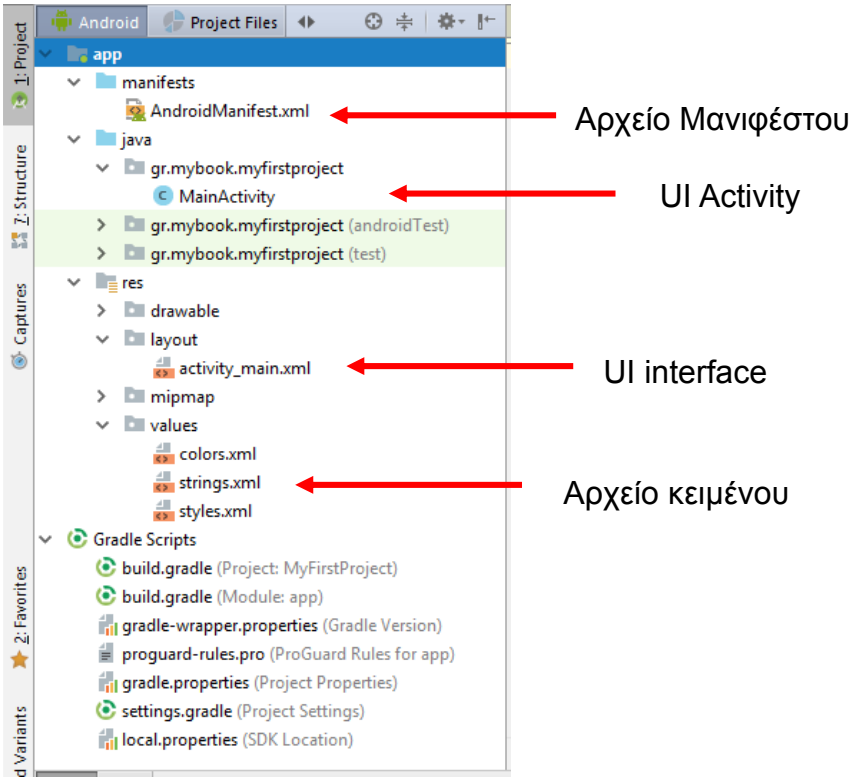
Είδαμε την οθόνη του Android Studio όταν επιλέξουμε να ανοίξουμε μια υπάρχουσα εφαρμογή ή μια νέα εφαρμογή. Ας περιηγηθούμε στο Android Studio.

Project Manager

Ας αναλύσουμε την εφαρμογή εξετάζοντας τον Project Manager που εμφανίζεται στο επάνω αριστερό μέρος.

- Το AndroidManifest.xml είναι το κεντρικό αρχείο xml των εφαρμογών μας επειδή περιέχει βασικές ρυθμίσεις και λειτουργίες. Είναι ένα περιγραφικό αρχείο xml το οποίο περιέχει τα συστατικά στοιχεία της εφαρμογής, όπως είναι οι δραστηριότητες, οι υπηρεσίες παρασκηνίου, οι προθέσεις, τα φίλτρα προθέσεων, οι άδειες χρήσης άλλων εφαρμογών, κλπ.
- Ο φάκελος "java" περιέχει το αρχείο MainActivity.java που αποτελεί την κύρια δραστηριότητα (activity) του project μας. Σε αυτόν τον φάκελο θα προστίθενται και νεότερα τμήματα κώδικα που ενδεχομένως θα αναπτυχθούν στη διάρκεια του μαθήματος.
- Ο φάκελος "res" περιέχει τους διαφόρους πόρους (resources), όπως εικονίδια, GUI layouts.
- Ο φάκελος "res → drawable" περιέχει τις εικόνες (PNG, JPEG, etc.) που χρησιμοποιούνται στην εφαρμογή σε διάφορες αναλύσεις (mdpi, hdpi, xhdpi, κλπ).

- Ο φάκελος "res → layout" περιέχει τα xml αρχεία τα οποία είναι υπεύθυνα για την εμφάνιση κάθε οθόνης της εφαρμογής μας. Στην συγκεκριμένη περίπτωση το activity_main.xml που αντιστοιχίζεται στην MainActivity.java περιέχει πληροφορίες για την εμφάνιση της οθόνης της κεντρικής μας δραστηριότητας.
- Ο φάκελος "res → mipmap" περιέχει το εικονίδιο της εφαρμογής (launcher icon) σε διάφορες αναλύσεις συσκευών.
- Ο φάκελος "res → values" είναι πολύ σημαντικός καθώς εκεί ορίζουμε όλες τις μεταβλητές που θα κάνει χρήση η εφαρμογή μας. Για κάθε τύπο μεταβλητής δημιουργείται ένα .xml αρχείο που περιέχει τα ονόματα και τις τιμές των αντίστοιχων μεταβλητών. Για παράδειγμα σε ένα project το strings.xml περιέχει όσες μεταβλητές τύπου string χρησιμοποιούνται στον κώδικα. Άλλα τέτοια αρχεία μπορεί να είναι το "colors.xml" για τα χρώματα, το "styles.xml" για το στυλ εμφάνισης.
- Μπορούμε να δημιουργήσουμε άλλα δικά μας αρχεία xml όπως το "res/menu/menu.xml" για την εμφάνιση ενός μενού για την εφαρμογή μας, το "res/raw/raw.xml" για αρχεία γενικού σκοπού (π.χ. ένα αρχείο CSV με διάφορες πληροφορίες).



- **Gradle Scripts** – Google user friendly development and build system.

Η δομή ενός αρχείου xml είναι:

```
<κεντρικό αντικείμενο εμφάνισης – root element >
        <τύπος αντικειμένου
                Ιδιότητα αντικειμένου = "τιμή ιδιότητας"
                …….
                /> <!-- κλείσιμο του tag του αντικειμένου -->
</ root element ><!-- κλείσιμο του tag του root element -->
<! -- με αυτό τον τρόπο γράφουμε σχόλια -->
```

Οι πόροι βρίσκονται στον φάκελο "res/" του έργου και περιλαμβάνουν διάφορα χαρακτηριστικά της εφαρμογής, όπως συμβολοσειρές, χρώματα, διαστάσεις, εικονίδια, διατάξεις οθόνης για τις δραστηριότητες, κλπ. Χαρακτηριστικό στοιχείο των αρχείων που διαχειρίζονται πόρους είναι ότι έχουν επέκταση xml.

- Ανάκτηση του πόρου "hello_world" από το αρχείο strings.xml:

```
String mName = getResources().getString(R.string.hello_world);
```

- Αναφορά σε έναν πόρο από κάποιο άλλο αρχείο πόρων:

```
@[resource type]/[resource name]
```

  π.χ. @string/hello_world

- Ανάκτηση μιας συμβολοσειράς του συστήματος:

```
String okay =
Resources.getSystem().getString(android.R.string.ok);
```

- Αναφορά σε έναν πόρο του συστήματος:

```
@android:[resource type]/[resource name]
```

  π.χ. @android:string/ok

Ανάλυση πόρων

1. Αρχείο μανιφέστου

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="gr.mybook.myfirstproject">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Οι βασικές του λειτουργίες είναι:

- Ονομάζει το πακέτο Java της εφαρμογής το οποίο λειτουργεί ως αναγνωριστικό ID για την εφαρμογή (**"gr.mybook.myfirstproject")**.
- Περιγράφει τις δραστηριότητες , τις υπηρεσίες, τους broadcast receivers.
- Ονοματίζει τις κλάσεις που υλοποιεί καθένα από τα στοιχεία.
- Καθορίζει ποιες διαδικασίες θα υλοποιήσουν τα στοιχεία της εφαρμογής.
- Καθορίζει τα δικαιώματα που πρέπει να αιτηθεί η εφαρμογή για να αποκτήσει πρόσβαση σε προστατευμένα μέρη του API .
- Καθορίζει τα δικαιώματα που πρέπει να έχουν οι άλλοι για να αλληλεπιδρούν με την εφαρμογή.
- Δηλώνει τη μικρότερη έκδοση Android API που χρειάζεται η εφαρμογή.
- Εμπερικλείει την κλήση των βιβλιοθηκών με τις οποίες συνδέεται η εφαρμογή.

Στο παραπάνω αρχείο, μερικά χαρακτηριστικά είναι:

- **<application>:** Ορίζει τα χαρακτηριστικά της εφαρμογής μας.
- **android:icon:** Το εικονίδιο της εφαρμογής μας.
- **android:label:** Το όνομα της εφαρμογής που θα εμφανίζεται στο χρήστη. Αυτό μπορεί να δίνεται με άμεσο τρόπο (**android:label="MyFirstProject"**) ή με έμμεσο τρόπο στο αρχείο "strings.xml" όπως παραπάνω.
- **<activity>:** Δηλώνει μια ενέργεια (activity) και στην προκειμένη περίπτωση αυτήν που είχαμε ήδη κατασκευάσει. Όλες οι ενέργειες πρέπει να είναι δηλωμένες διαφορετικά δεν θα είναι εκτελέσιμες.

2. Αρχείο κώδικα

Η MainActivity.java αποτελεί την κεντρική κλάση της εφαρμογής μας.

Δημιουργία εφαρμογής

```
package gr.mybook.myfirstproject;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Διάταξη οθόνης

- **package** gr.mybook.myfirstproject → Το όνομα του πακέτου που δημιουργήθηκε αυτόματα όταν δημιουργήθηκε το project.
- **import** android.os.Bundle → Βιβλιοθήκη που θα γίνει χρήση της στην κύρια κλάση.
- **import** android.support.v7.app.AppCompatActivity → Οι βιβλιοθήκες υποστήριξης επιτρέπουν σε εφαρμογές που εκτελούνται σε παλαιότερες εκδόσεις της πλατφόρμας Android να υποστηρίζουν λειτουργίες διαθέσιμες σε νεότερες εκδόσεις της πλατφόρμας.

- **public class** MainActivity **extends** AppCompatActivity → Δήλωση της κύριας κλάσης η οποία είναι υποκλάση της " AppCompatActivity".
- **@Override** → Δηλώνει ότι η συνάρτηση που επακολουθεί θα επικαλύψει την αντίστοιχη συνάρτηση που υπάρχει στην υπερκλάση που δηλώνεται με "super".
- **protected void** onCreate(Bundle savedInstanceState) → Κλήση της συνάρτησης "onCreate()". Οι δραστηριότητες έχουν τη δυνατότητα να αποκατασταθούν σε προηγούμενη κατάσταση χρησιμοποιώντας τα δεδομένα που είναι αποθηκευμένα σε αυτή τη δέσμη (π.χ. μετά από τερματισμό λόγω περιστροφής της συσκευής).
- setContentView(R.layout.***activity_main***) → Χρήση του αρχείου "activity_main.xml" σαν UI στην εφαρμογή.

3. Φάκελος εικονιδίων

- Στον φάκελο res/drawable υπάρχουν τα αρχεία εικονιδίων σε διάφορες αναλύσεις hdpi, -ldpi, -mdpi, -xhdpi, -xxhdpi.

Η ανάκληση της εικόνας /res/drawable/name.png γίνεται ως εξής:

```
BitmapDrawable myImage = (BitmapDrawable)
            getResources().getDrawable(R.drawable.name);
```

- Απόδοση της εικόνας "name.png" στο πλήκτρο "ImageView1" της διάταξης οθόνης γίνεται ως εξής:

```
ImageView myView = (ImageView) findViewById(R.id.ImageView1);
myView.setImageResource(R.drawable.name);
```

4. Αρχείο UI

Το activity_main.xml που βρίσκεται στον φάκελο "res → layout" περιέχει όλα εκείνα τα αντικείμενα τα οποία εμφανίζονται στην κεντρική οθόνη της εφαρμογής. Κάθε εντολή αντιπροσωπεύει ένα αντικείμενο ή κάποια ιδιότητά του.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="76dp"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.43" />

</android.support.constraint.ConstraintLayout>
```

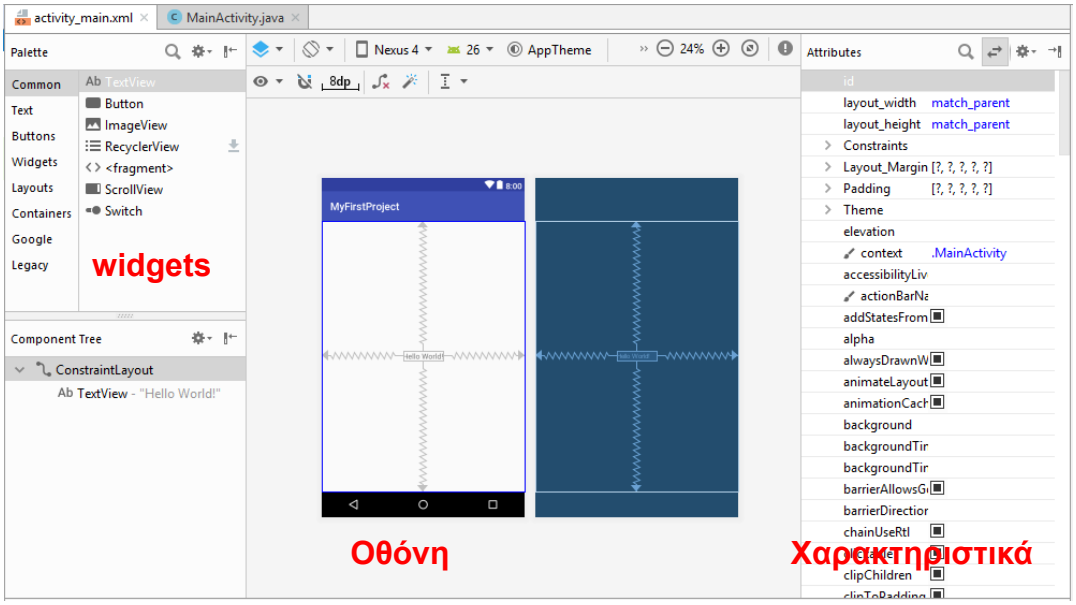
- Το Root Element είναι το "ConstraintLayout" μέσα στο οποίο υπάρχουν όλα τα αντικείμενα που εμφανίζονται στην οθόνη. Τα layouts καθορίζουν τον τρόπο με τον οποίο τοποθετούνται τα αντικείμενα στην οθόνη.
- Ορίζεται το πλάτος και το ύψος του layout να είναι στο μέγεθος της οθόνης της συσκευής.
- Ακολουθεί ένα αντικείμενο τύπου "TextView" το οποίο καλείται πεδίο κειμένου. Στη συνέχεια ορίζεται η ταυτότητά του (android:id = "@+id/text1"). Το "@+" ορίζει νέα μεταβλητή, ενώ το "@" ψάχνει να βρει ήδη δηλωθείσα μεταβλητή. Ορίζονται οι διαστάσεις του, με το πλάτος 76 dp (density independent pixels) και το ύψος με τη δήλωση "wrap_content" που σημαίνει όσο ορίζει το ύψος του κειμένου. Στη συνέχεια δηλώνεται το κείμενο με άμεσο ή έμμεσο τρόπο. Οι υπόλοιπες δηλώσεις αφορούν τη θέση του πεδίου κειμένου στην οθόνη της συσκευής.
- Αντί του άμεσου τρόπου απόδοσης του κειμένου προτιμάται ο έμμεσος τρόπος δίνοντας την εντολή:

android:text="@string/str_1"

Στο αρχείο "strings.xml" θα πρέπει να προσθέσουμε το "str_1":

```
<string name="str_1">Hello World!</string>
```

Η γραφική όψη του xml αρχείου είναι όπως φαίνεται στην παρακάτω εικόνα:



5. Αρχείο μενού

Δημιουργούμε τον φάκελο *res/menu* και μέσα σε αυτόν το αρχείο *main.xml* το οποίο δημιουργεί στην οθόνη της συσκευής το μενού επιλογών (Options Menu) όταν πατηθεί το πλήκτρο MENU. Στο συγκεκριμένο το περιεχόμενο του μενού είναι οι επιλογές: "Settings", "Help", "Info".

```
<menu
xmlns:android = "http://schemas.android.com/apk/res/android" >
  <item
     android:id = "@+id/action_settings"
     android:orderInCategory = "100"
     android:showAsAction ="never"
     android:title = "@string/action_settings"/>
  <item android:id = "@+id/info"
     android:title = "Info" />
  <item android:id = "@+id/help"
     android:title = "Help" />
</menu>
```

Ο κώδικας που πρέπει να υπάρχει στην κύρια δραστηριότητα για την κλήση του μενού είναι:

```
@Override
  public boolean onCreateOptionsMenu(Menu menu) {
         // Inflate the menu; this adds items to the action bar
            if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
```

Εάν θέλουμε η επιλογή ενός στοιχείου του μενού να πραγματοποιεί κάποια ενέργεια, θα πρέπει να προστεθεί στον κώδικα της κύριας δραστηριότητας:

```
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  case R.id.info:
    startActivity(new Intent(this, Info.class));
    return true;
  case R.id.help:
    startActivity(new Intent(this, Help.class));
    return true;
  case R.id.action_settings:
    startActivity(new Intent(this, ActionSettings.class));
    return true;
    default:
    return super.onOptionsItemSelected(item);
  }
}
```

Κάθε επιλογή οδηγεί σε μια νέα δραστηριότητα (νέα κλάση).

6. Αρχείο στυλ

Στον φάκελο *res/values* υπάρχει το αρχείο *styles.xml* το οποίο παρέχει το στυλ μιας όψης ή μιας δραστηριότητας. Όταν αφορά μια όψη αναφέρεται με το *style*, ενώ όταν αφορά τη δραστηριότητα αναφέρεται με το *theme*.

7. Αρχείο χρωμάτων

Στον φάκελο *res/values* μπορεί να δημιουργηθεί το αρχείο χρωμάτων *colors.xml*.

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
   <color name = ”background_color”>#ffe4e1</color>
   <color name = ”myView_color”>#cdc0b0</color>
</resources>
```

Η ανάκληση ενός χρώματος με κώδικα γίνεται ως εξής:

```
int myColor = getResources().getColor(R.color.name);
```

Αποσφαλμάτωση

Στο παράθυρο Logcat εμφανίζονται μηνύματα κατά την εκτέλεση μιας εφαρμογής. Με αυτόν τον τρόπο μπορούμε να ελέγχουμε την ορθή εκτέλεση του κώδικά μας προκαλώντας εμφανίσεις μηνυμάτων. Οι κατηγορίες είναι:

Log.e() - Καταγραφή σφαλμάτων,
Log.w() - Καταγραφή προειδοποιήσεων
Log.i() - Καταγραφή πληροφοριών,
Log.d() - Καταγραφή μηνυμάτων αποσφαλμάτωσης
Log.v() - Καταγραφή όλων των περιπτώσεων

- Η δήλωση στον κώδικα γίνεται ως εξής:

```
private static final String TAG = "gr.mybook.myfirstproject";
```

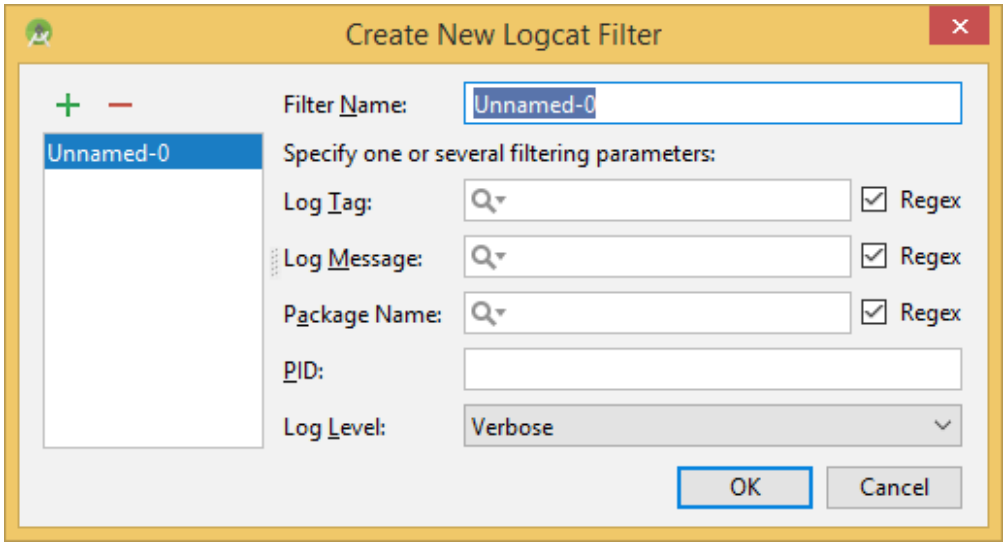
- Η χρήση στον κώδικα γίνεται ως εξής:

```
Log.i(TAG, "We are within the onCreate method");
```

Για παράδειγμα μπορούμε να καταλάβουμε ότι έχει εκτελεστεί η δραστηριότητα MainActivity με τον εξής τρόπο:

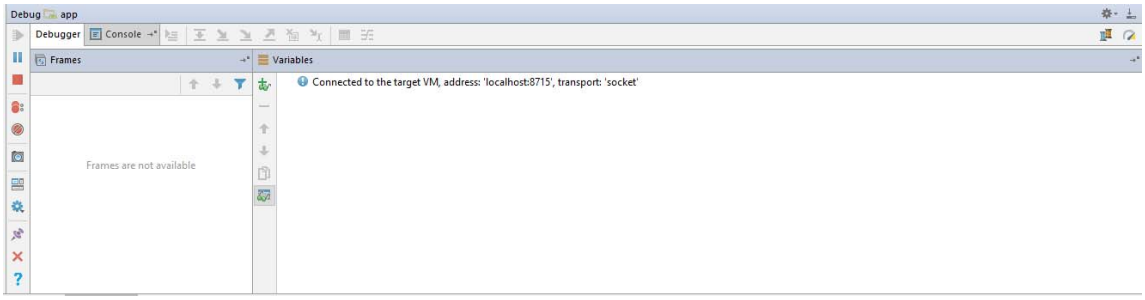
```
import android.util.Log;

public class MainActivity extends Activity {
  private static final String TAG = "gr.mybook.myfirstproject";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.i(TAG, "We are within the onCreate method");
  }
}
```

Μπορούμε να δημιουργήσουμε ένα φίλτρο στην οθόνη Logcat έτσι ώστε να παρουσιάζονται μόνο τα μηνύματα που επιθυμούμε.



Regex = Regular Expressions

Μπορούμε να πραγματοποιήσουμε αποσφαλμάτωση (Debbugging) σε μια εφαρμογή μας αν για την εκτέλεση της εφαρμογής επιλέξουμε το "Debug 'app'" από τη γραμμή εργαλείων του Android Studio. Η οθόνη αποσφαλμάτωσης είναι:

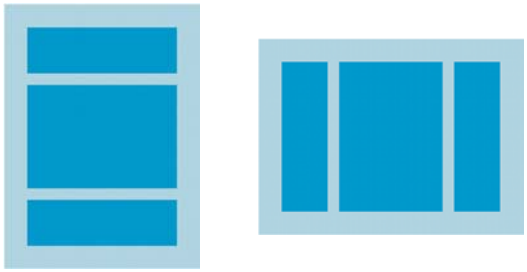


Μπορούμε σε διάφορα σημεία της εφαρμογής μας να τοποθετήσουμε "breakpoints" για να εξετάζουμε κατά τμήματα την ορθή εκτέλεση του προγράμματος. Στην οθόνη αποσφαλμάτωσης μπορούμε να δούμε διαδικασίες, μεταβλητές, κλπ.

Διατάξεις οθόνης (Layouts)

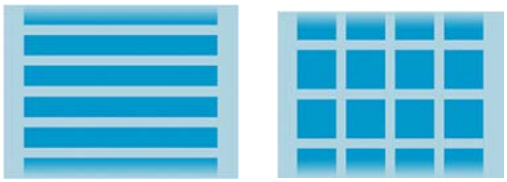
Τα συνηθισμένα layouts, δηλαδή οι διατάξεις οθόνης μέσα στις οποίες θα τοποθετηθούν αντικείμενα, είναι LinearLayout και RelativeLayout. Το Android Studio περιέχει ένα νέο τύπο layout το οποίο καλείται ConstraintLayout. Είναι μια δυναμική νέα κλάση η οποία μας επιτρέπει να διατάξουμε τα child views χρησιμοποιώντας 'περιορισμούς' σχετικά με τις θέσεις μεταξύ διαφορετικών όψεων που βρίσκονται στη διάταξή μας. Η συνεισφορά του είναι να απλοποιηθεί η σχεδίαση σύνθετων όψεων οι οποίες να εκτελούνται πιο γρήγορα.

Το LinearLayout είναι μια διάταξη που ευθυγραμμίζει όλα τα "child views" σε μια κατεύθυνση, οριζόντια ή κάθετη (ιδιότητα android:orientation).Οι όψεις τοποθετούνται η μία μετά την άλλη καταλαμβάνοντας ολόκληρη τη γραμμή ή στήλη ενώ παράλληλα αναγνωρίζει τα περιθώρια μεταξύ τους και τη θέση τους (αριστερά, κέντρο, δεξιά).



Το RelativeLayout είναι μια διάταξη που τοποθετεί μια όψη σε σχέση με τη θέση μιας άλλης ή σε σχέση με την όλη διάταξη. Επομένως, μπορεί να αποφεύγεται η χρήση εμφωλευμένων διατάξεων LinearLayout για την υλοποίηση μιας σύνθετης διάταξης, απλοποιώντας τη σχεδίαση και κάνοντας την εφαρμογή πιο γρήγορη.

Όταν το περιεχόμενο μιας διάταξης οθόνης είναι δυναμικό και όχι εκ των προτέρων γνωστό, μπορεί να χρησιμοποιηθεί μια υποκλάση της διάταξης AdapterView ώστε να δημιουργούνται όψεις κατά την εκτέλεση της εφαρμογής. Το αντικείμενο Adapter αντλεί τα δεδομένα που πρόκειται να εμφανιστούν στην οθόνη και τα μετατρέπει σε όψεις για τη διάταξη οθόνης. Τέτοιες διατάξεις οθόνης είναι η ListView και η GridView.



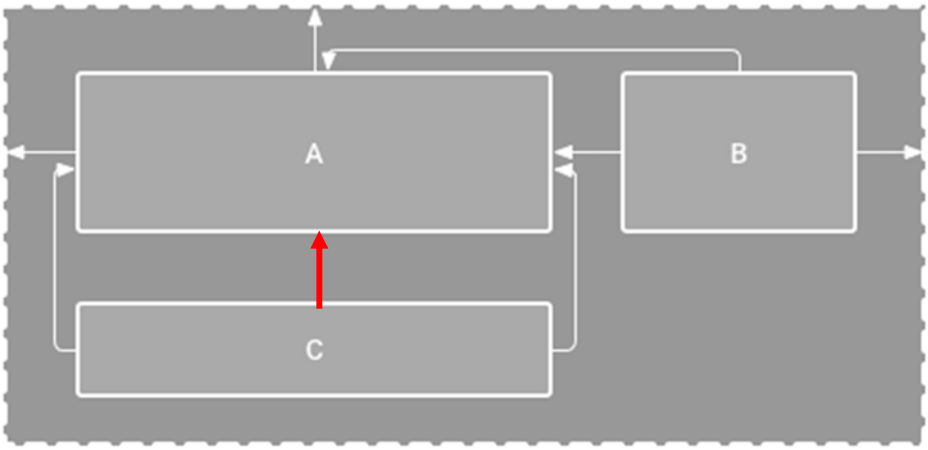
Ο ArrayAdapter μετατρέπει κάθε στοιχείο ενός array σε ένα TextView με την κλήση της συνάρτησης "toString()". Έτσι αν έχουμε έναν πίνακα με αλφαριθμητικά τα οποία πρόκειται να εμφανιστούν σε ένα ListView:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

ConstraintLayout

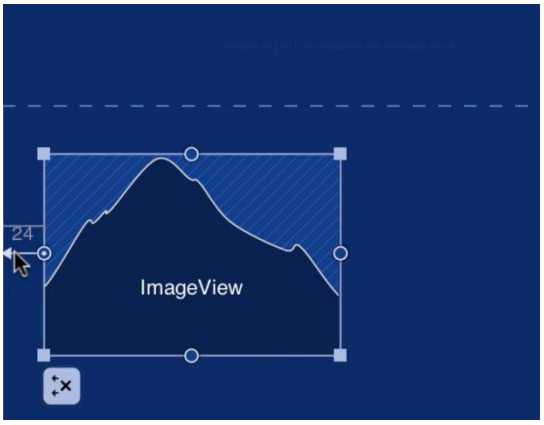
Αυτή η διάταξη οθόνης μας επιτρέπει να δημιουργούμε σύνθετες και μεγάλες διατάξεις οθόνης χωρίς εμφωλευμένες διατάξεις. Είναι παρόμοια με την RelativeLayout ως προς τη διάταξη των όψεων σε σχέση με τη διάταξη που ανήκουν και τις άλλες όψεις αλλά είναι περισσότερο ευέλικτη. Τα χαρακτηριστικά της είναι τα εξής:

- Η θέση μιας όψης στη διάταξη καθορίζεται από τουλάχιστον έναν περιορισμό (constraint) οριζόντια και κάθετα. Κάθε περιορισμός αντιπροσωπεύει μια σύνδεση ή ευθυγράμμιση με μια άλλη όψη, τη γονική διάταξη ή μια αόρατη κατευθυντήρια γραμμή. Εάν μια όψη δεν έχει περιορισμούς εμφανίζεται επάνω και αριστερά στην οθόνη. Για παράδειγμα, η όψη C δεν έχει κάθετο περιορισμό και επομένως θα εμφανιστεί ευθυγραμμισμένη με την A αλλά στο επάνω μέρος της οθόνης.



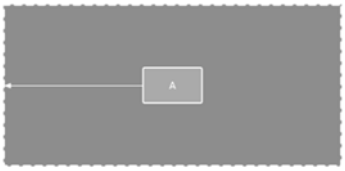
Ο Layout Editor δείχνει τους μη υπάρχοντες περιορισμούς στη γραμμή εργαλείων. Για να δούμε τα σφάλματα πρέπει να πατήσουμε **Show Warnings and Errors** . Επίσης μπορεί να τοποθετήσει αυτόματα τους περιορισμούς με το **Autoconnect and infer constraints**.

- Τοποθετούμε μια όψη μέσα στη διάταξη οθόνης σύροντάς την από την παλέτα όψεων.

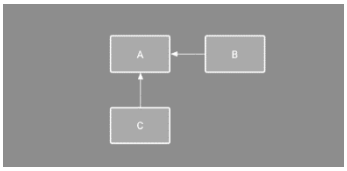


Παρουσιάζεται σαν ένα ορθογώνιο με τετράγωνες λαβές αλλαγής μεγέθους (resizing handles) στις κορυφές και στρογγυλές λαβές περιορισμού (constraint handles) στις πλευρές. Σύρουμε την όψη από τα constraint handles και το τοποθετούμε σε σχέση με τη διάταξη οθόνης ή με άλλη όψη. Τέλος, πατάμε "**Create a connection**" . Όταν τοποθετηθεί, ο Editor δίνει ένα προκαθορισμένο περιθώριο (margin) για το διαχωρισμό των δύο οντοτήτων.

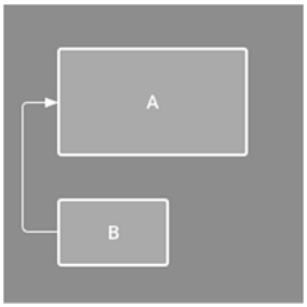
- Για την κατάργηση ενός περιορισμού επιλέγουμε την όψη και πατάμε το "constraint handle". Για να απομακρύνουμε όλους τους περιορισμούς πατάμε "**Delete constraints**" .



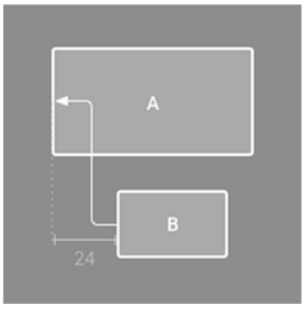
Horizontal constraint to the parent.
Ορισμός της απόστασης με το περιθώριο (margin).



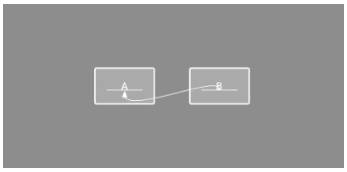
Περιορισμός σε σχέση με άλλη όψη. Οι B και C έχουν μόνο έναν περιορισμό.



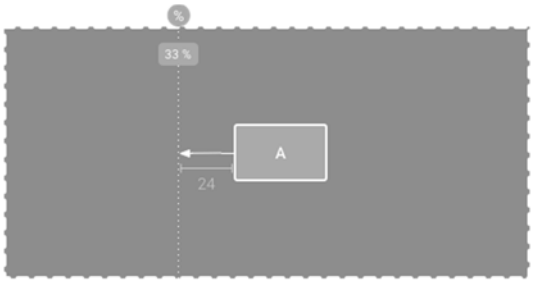
Η όψη B ευθυγραμμίζεται με την A αριστερά. Χρησιμοποιούμε και το δεξί handle αν θέλουμε να κεντραριστούν.



Μπορούμε να δημιουργήσουμε μια απόσταση από την αριστερή πλευρά της A επιλέγοντας τιμή για το περιθώριο.

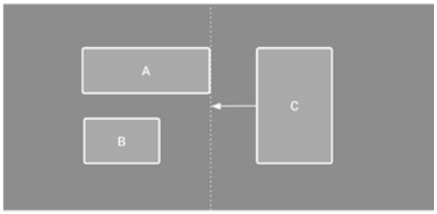


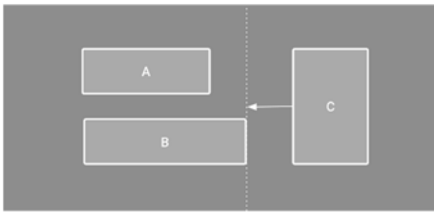
Ευθυγράμμιση του κειμένου δύο όψεων. Για τη δημιουργία ενός περιορισμού κειμένου (baseline constraint) επιλέγουμε την όψη και **"Edit Baseline"** . Στη συνέχεια συνδέουμε τις 2 baselines.



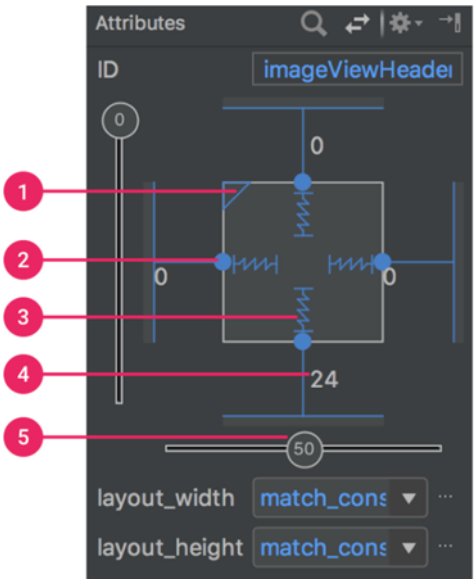
Μπορούμε να τοποθετήσουμε έναν οδηγό (guideline) και ως προς αυτόν μια όψη καθορίζοντας το περιθώριο. Η θέση του οδηγού μπορεί να είναι ως προς τη διάταξη σε dp ή %. Πατάμε "**Guidelines**" και μετά "**Add Vertical Guideline**" ή "**Add Horizontal Guideline**". Τη σύρουμε στην επιθυμητή θέση και πατάμε στον κύκλο για να ορίσουμε τη μονάδα μέτρησης.

Μπορούμε να τοποθετήσουμε ένα φράγμα (barrier) και μέσα σε αυτό όψεις. Το barrier αλλάζει θέση ανάλογα με τις όψεις που βρίσκονται σε αυτό. Χρησιμοποιείται για να ορίσουμε περιορισμό σε μια όψη (π.χ. την C) ως προς μια ομάδα όψεων. Πατάμε "**Guidelines**" → "**Add Vertical/Horizontal Barrier**". Στο παράθυρο "**Component Tree**" επιλέγουμε τις όψεις και τις σύρουμε στο barrier. Επιλέγουμε το barrier από το "**Component Tree**", ανοίγουμε το παράθυρο "**Attributes**" και επιλέγουμε το "**barrierDirection**". Μπορούμε να τοποθετήσουμε περιορισμούς για τις όψεις μέσα στο barrier ως προς αυτό όπως επίσης και Guideline για την ελάχιστη απόσταση των όψεων από αυτό.





- Όταν τοποθετούμε περιορισμούς και στις δύο πλευρές μιας όψης (και το μέγεθος της όψης είναι "fixed" ή "wrap content"), η όψη κεντράρεται ως προς το μέσον δηλαδή αποκτά bias 50%. Το bias μεταβάλλεται κινώντας την όψη ή σύροντας το slider στο παράθυρο των Ιδιοτήτων (Attributes).



1. **Size ratio.**
2. **Delete constraint**
3. **Height/Width mode**
4. **Margins**
5. **Constraint bias**

Τα σύμβολα στο Height/Width mode είναι:

 Fixed = Συγκεκριμένες διαστάσεις ή αλλαγή μεγέθους (view resizing).

 Wrap Content = Η διάσταση είναι όσο και το περιεχόμενο.

 Match Constraints = Η όψη περιορίζεται από τα ορισθέντα constraints λαμβάνοντας υπόψη τα περιθώρια. Σε αυτήν την περίπτωση μπορεί να υπάρχουν οι εξής ιδιότητες:

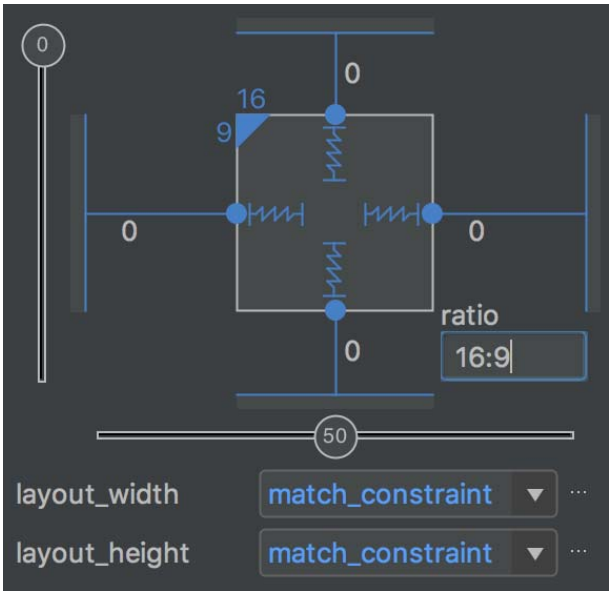
- **layout_constraintWidth_default**
  - **spread**: Επεκτείνει την όψη όσο επιτρέπουν τα constraints (default).

- **wrap**: Επεκτείνει την όψη όσο τα περιεχόμενα της όψης αλλά επιτρέπει στην όψη να είναι μικρότερη εάν το επιτρέπουν οι περιορισμοί, σε αντίθεση με την επιλογή "**Wrap Content**".

➢ **layout_constraintWidth_min**
Διάσταση σε dp για το ελάχιστο πλάτος της όψης.

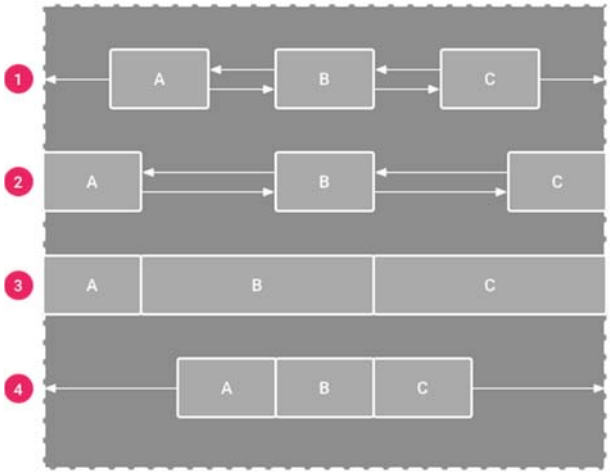
➢ **layout_constraintWidth_max**
Διάσταση σε dp για το μέγιστο πλάτος της όψης.

- Το μέγεθος μιας όψης μπορεί να είναι αναλογία (π.χ. 16:9) εάν μια τουλάχιστον διάσταση τεθεί σαν "match constraints" (0 dp). Πατάμε "**Toggle Aspect Ratio Constraint**" (επιλογή 1 στην παραπάνω εικόνα των ιδιοτήτων) και επιλέγουμε την αναλογία.



- Για τον καθορισμό του περιθωρίου στις όψεις πατάμε "Margin" 8 . Από τις ιδιότητες κάθε όψης μπορεί να τοποθετηθεί το περιθώριο της.

- Σε μια ομάδα όψεων μπορούμε να δημιουργήσουμε αλυσίδα (chain).



1 = Spread – Ομοιόμορφη κατανομή μετά τον υπολογισμό των περιθωρίων.
2 = Spread Inside – η πρώτη και τελευταία όψη ακολουθούν τους περιορισμούς και οι ενδιάμεσες κατανέμονται ομοιόμορφα.
3 = Weighted – Όταν η αλυσίδα είναι σε μία από τις παραπάνω μορφές, οι ενδιάμεσες αποστάσεις καλύπτονται θέτοντας τις όψεις σε "match constraints" (0 dp). Η κάλυψη μπορεί να είναι ομοιόμορφη ή με βάρος χρησιμοποιώντας τα "**layout_constraintHorizontal_weight**" ή "**layout_constraintVertical_weight**".
4 = Packed – Οι όψεις γίνονται διαδοχικές αφού υπολογιστούν τα περιθώρια.

Η αλυσίδα γίνεται επιλέγοντας τις όψεις σύροντας το ποντίκι γύρω από αυτές, στη συνέχεια πάνω σε μία όψη με δεξί κλικ επιλέγουμε "**Center Horizontally**" ή "**Center Vertically**" και τέλος με το  . Τα modes επιλέγονται πατώντας σε μια όψη διαδοχικά την αλυσίδα.

- Για τη δημιουργία αυτόματων περιορισμών σύρουμε τις όψεις στο σημείο που επιθυμούμε και πατάμε το "**Infer Constraints**"  .
- Το "**Autoconnect**"  δημιουργεί τουλάχιστον δύο περιορισμούς σε κάθε όψη που επιλέγεται σε σχέση μόνο με τη γονική διάταξη, όταν είναι ON.

<u>Πρακτική εφαρμογή</u>: Ανοίξτε την εφαρμογή "constraint-layout-start" και ακολουθήστε το παράδειγμα που υπάρχει στον ιστότοπο:

https://codelabs.developers.google.com/codelabs/constraint-layout/#0

Google Firebase

Η Google Firebase είναι μια πλατφόρμα ανάπτυξης mobile και web εφαρμογών, με εργαλεία και υποδομές τα οποία βοηθούν τους προγραμματιστές. Αρχικά, η Firebase ήταν μια βάση δεδομένων πραγματικού χρόνου (real time database), η οποία παρείχε ένα API που επέτρεπε στους προγραμματιστές να αποθηκεύουν και να συγχρονίζουν δεδομένα μεταξύ πολλαπλών clients. Σήμερα έχει γίνει πλέον μια πλήρης σουίτα για ανάπτυξη εφαρμογών [1]:

Analytics

Firebase Analytics: Παρέχει μετρήσεις και στατιστικά σχετικά με τη χρησιμοποίηση της εφαρμογής και τους μηνιαία ενεργούς χρήστες ανάλογα με το φύλο, την ηλικία και την περιοχή.

Develop

Authentication: Καθιστά δυνατή την αυθεντικοποίηση των χρηστών μόνο με client-side κώδικα, έχοντας το back-end ήδη υλοποιημένο. Επιτρέπει το login, εκτός από e-mail και κωδικό, και με εξωτερικούς παρόχους login (Facebook, Github, Twitter, Google).
Realtime Database: Είναι μία noSQL βάση δεδομένων αποθηκευμένη στο Cloud σε μορφή JSON. Αντί για κλασσικά HTTP Requests χρησιμοποιεί συγχρονισμό δεδομένων, επιτρέποντας την ενημέρωση οποιασδήποτε συσκευής που τρέχει την εκάστοτε εφαρμογή μέσα σε milliseconds. Η πρόσβαση στη βάση μπορεί να περιοριστεί από τον προγραμματιστή μέσω κανόνων ασφάλειας (Security Rules).

[1] Ελπίδα Ρουκά, «Σχεδίαση και Υλοποίηση κοινωνικού δικτύου που επιτρέπει στους χρήστες να μοιράζονται την τοποθεσία τους με άλλους χρήστες σε περιπτώσεις κινδύνου», Διπλωματική εργασία, ΕΜΠ, Ιούλιος 2017.

Storage: Προσφέρει ασφαλή μεταφόρτωση και λήψη αρχείων για τις εφαρμογές του Firebase, ανεξάρτητα της ποιότητας δικτύου. Χρησιμοποιείται για αποθήκευση αρχείων εικόνας, ήχου και βίντεο των χρηστών.
Hosting: Χρησιμοποείται σε Web εφαρμογές, για την ανάπτυξη και κυκλοφορία στατικού περιεχομένου (π.χ. αρχεία CSS, HTML, JavaScript) σε ένα δίκτυο διανομής περιεχομένου (ContentDelivery Network, CDN) μέσω HTTPS/SSL.
Cloud Functions: Δίνει τη δυνατότητα στον προγραμματιστή να γράφει back-end κώδικα που ενεργοποιείται αυτόματα όταν συμβαίνουν συγκεκριμένες αλλαγές ή αιτήσεις στη βάση. Ο κώδικας αποθηκεύεται στο Google Cloud.
Test Lab: Παρέχει μία υποδομή δοκιμών (testing) για εφαρμογές Android. Τα αποτελέσματα των δοκιμών (screenshots, logs, videos) γίνονται διαθέσιμα στον προγραμματιστή στην κονσόλα του Firebase. Βοηθά ιδιαίτερα στην ανεύρεση προβλημάτων που οφείλονται σε κάποιο συγκεκριμένο συνδυασμό συσκευής – λογισμικού.
Crash Reporting: Δημιουργεί λεπτομερείς αναφορές (reports) για σφάλματα που συμβαίνουν στο πρόγραμμα.
Performance Monitoring: Βοηθά τον προγραμματιστή να διεισδύσει στα προβλήματα επίδοσης της εφαρμογής του. Παρέχει ένα SDK συλλογής δεδομένων επίδοσης τα οποία έπειτα παρουσιάζονται και αναλύονται στην κονσόλα.
Firebase Cloud Messaging (FCM): Εργαλείο για αποστολή μηνυμάτων μεταξύ των χρηστών, που διέρχονται από τον server της εφαρμογής, καθώς και απομακρυσμένων ειδοποιήσεων (remote notifications) ως απόκριση σε ορισμένα γεγονότα.

Grow

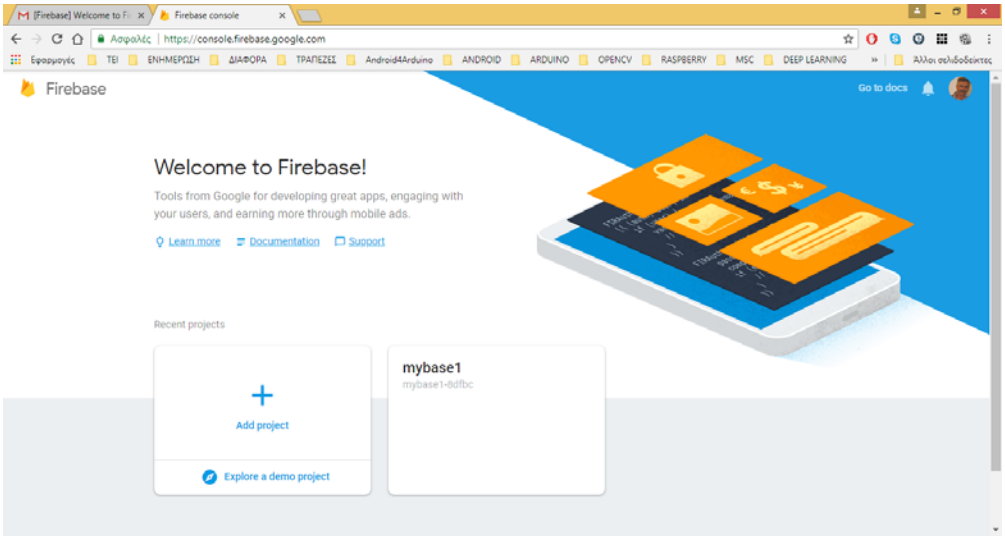
Notifications: Δωρεάν υπηρεσία αποστολής μηνυμάτων και ειδοποιήσεων από τον προγραμματιστή σε συγκεκριμένες συσκευές με σκοπό το testing.
Remote Configuration: Υπηρεσία που καθιστά ικανούς τους προγραμματιστές να αλλάξουν τη συμπεριφορά και τα γραφικά της εφαρμογής χωρίς να απαιτηθεί από το χρήστη να εκτελέσει κάποιο update.
Dynamic Links: Σύνδεσμοι προς το περιβάλλον της εφαρμογής που συμπεριφέρονται ανάλογα με τη συσκευή στην οποία βρίσκεται ο χρήστης, για παράδειγμα Mobile View ή Desktop View της εφαρμογής. Ακόμη, σε περίπτωση που η εφαρμογή δεν είναι εγκατεστημένη, μπορούν να παραπέμπουν το χρήστη στο App Store.

Earn

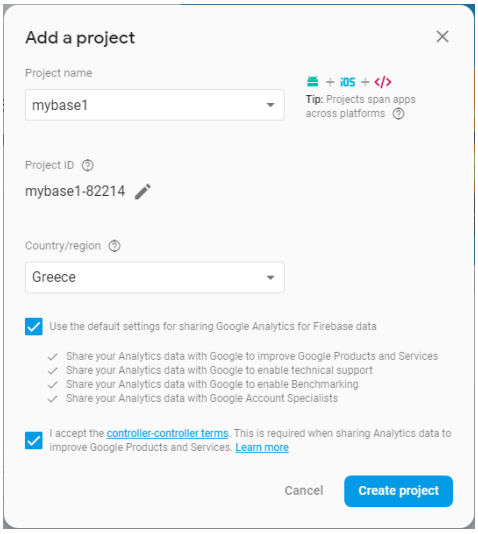
AdMob: Πλατφόρμα τοποθέτησης διαφημίσεων εντός της εφαρμογής που καθιστά τον προγραμματιστή ικανό να αποκτήσει εισόδημα μέσω αυτής.

Χρήση της βάσης δεδομένων με κινητή συσκευή

- Συνδεόμαστε με την Firebase: https://console.firebase.google.com.



- Δημιουργούμε ένα νέο project (π.χ. mybase1) με το Add project.

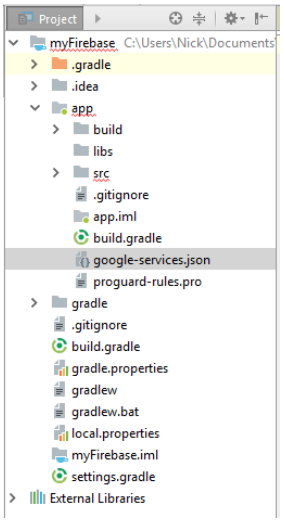


- Δημιουργούμε στο Android Studio μια εφαρμογή για να δοκιμάσουμε τη λειτουργία της βάσης δεδομένων Firebase. Ονομάζουμε την εφαρμογή "myFirebase" και έχει το παρακάτω UI. Πατώντας "WRITE", ότι υπάρχει στο πεδίο κειμένου θα αποθηκεύεται στη βάση δεδομένων ενώ πατώντας "READ" ότι υπάρχει στη βάση δεδομένων (στο συγκεκριμένο κλειδί) εμφανίζεται στην ετικέτα κειμένου.

- Στην οθόνη του project που εμφανίζεται πατάμε "Add Firebase to you Android App". Παρουσιάζονται τα βήματα για τη σύνδεση της βάσης δεδομένων με την εφαρμογή μας.
  1. **Register App:** Τοποθετούμε το package name της εφαρμογής μας, δηλαδή το gr.mybook.myfirebase. Πατάμε το πλήκτρο "Register app".
  2. **Download config file:** Κατεβάζουμε το "google-services.json" και το τοποθετούμε στο root της εφαρμογής μας (Οθόνη με επιλογή του Project). Πατάμε "Next".
  3. **Add Firebase SDK:** Προσθέτουμε στο αρχείο "build.gradle (Project: myFirebase)" (που υπάρχει στην οθόνη με ένδειξη Andoid) το: classpath 'com.google.gms:google-services:4.0.0' και στο αρχείο "build.gradle (Module:

app)" τα: implementation 'com.google.firebase:firebase-core:16.0.0' και implementation 'com.google.firebase:firebase-database:16.0.1'

4. Στο τέλος του αρχείου προσθέτουμε το: apply plugin: 'com.google.gms.google-services'.



Project-level build.gradle (<project>/build.gradle):

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:4.0.0'
  }
}
```

App-level build.gradle (<project>/<app-module>/build.gradle):

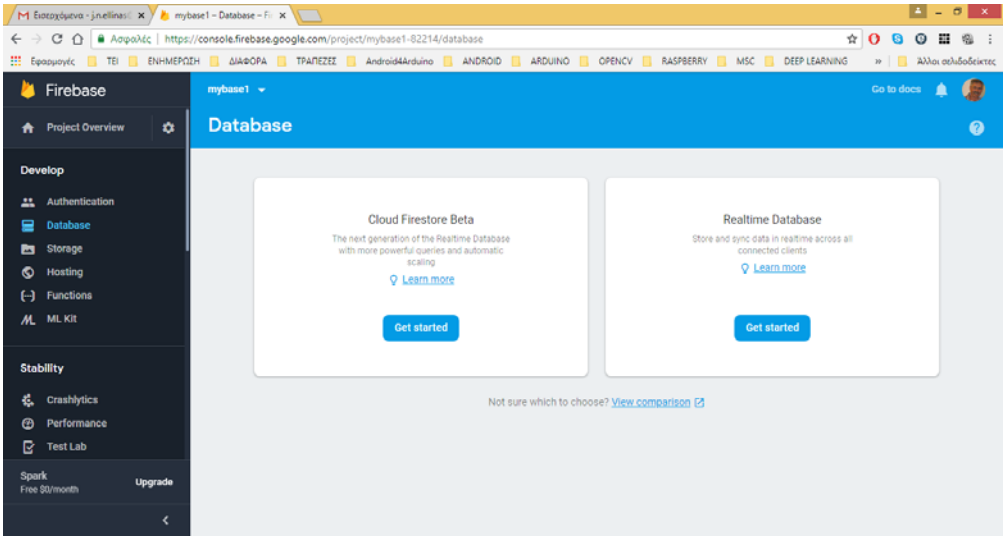
```
dependencies {
  // Add this line
  compile 'com.google.firebase:firebase-core:16.0.0'
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Includes Analytics by default

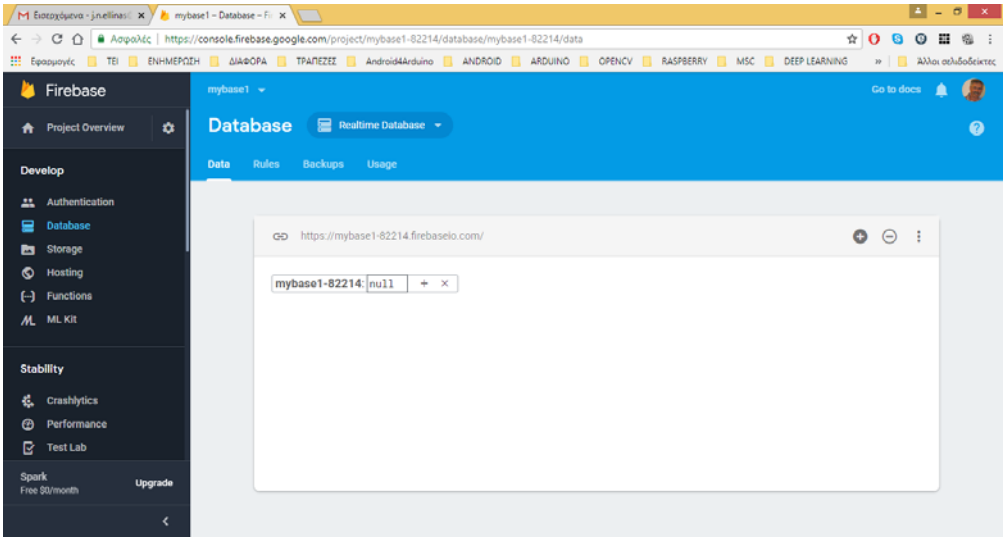
Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed sir Sync now

5. Στη συνέχεια πατάμε "Sync Now" στο Android Studio και παρατηρούμε αν υπάρχουν σφάλματα. Αν υπάρχουν σφάλματα δοκιμάζουμε προηγούμενες εκδόσεις, όπως 'com.google.gms:google-services:3.1.0' ή 'com.google.firebase:firebase-core:10.2.6'. Πατάμε "Next".
6. **Run your app to verify installation.** Κάνουμε skip.
7. **Συμπληρώνουμε τον κώδικα στην εφαρμογή μας με την ενέργεια των πλήκτρων. Πρέπει από τον SDK Manager και τα SDK Tools να έχουμε κατεβάσει τα "Google Play services".**
8. Στην κονσόλα firebase μετά τη δημιουργία της βάσης και τη σύνδεση με κάποια συγκεκριμένη εφαρμογή Android:



Πατάμε "Realtime Database"→ Get started → Start in test mode → Enable. Εμφανίζεται η βάση δεδομένων στην οποία θα παρατηρούμε τα στοιχεία της.



9. Ο κώδικας της εφαρμογής:

```
package gr.mybook.myfirebase;

import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
    TextView textView1;
    EditText editText1;
    Button btn1, btn2;

    FirebaseDatabase fbdatabase;
    DatabaseReference fbref;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textView1=(TextView)findViewById(R.id.textView1);
        editText1=(EditText)findViewById(R.id.editText1);
        fbdatabase=FirebaseDatabase.getInstance();
        fbref=fbdatabase.getReference("mykey1");
    }
```

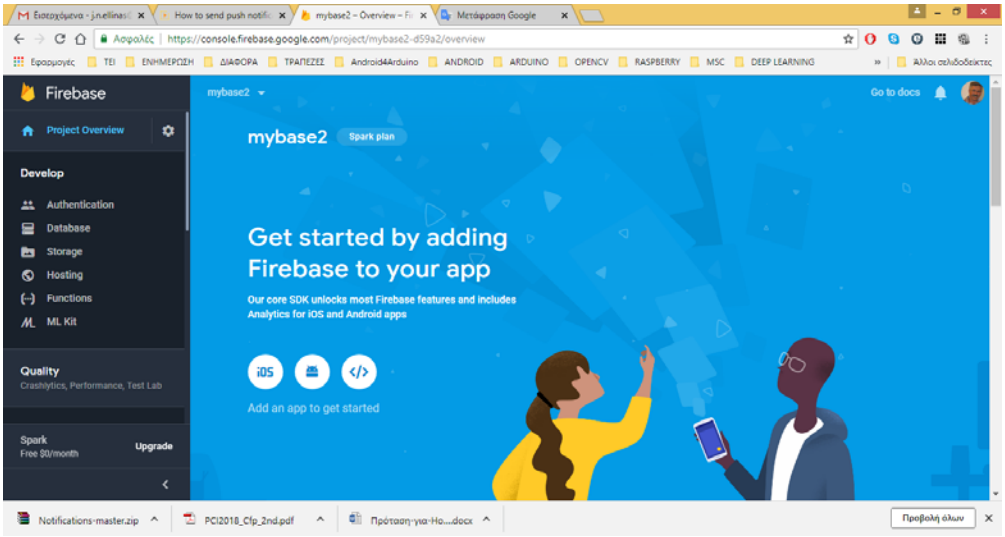
```
    public void read(View view){
        fbref.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
       textView1.setText(dataSnapshot.getValue().toString());
            }

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

            }
        });
    }
    public void write(View view){
        fbref.setValue(editText1.getText().toString());
    }
}
```

- Οι κλάσεις που χρησιμοποιούνται για τη βάση δεδομένων:
  ```
  FirebaseDatabase fbdatabase;
  DatabaseReference fbref;
  ```
- Ορισμός των αντικειμένων τους:
  ```
  fbdatabase=FirebaseDatabase.getInstance();
  fbref=fbdatabase.getReference("mykey1");
  ```
  Το "mykey1" είναι το κλειδί χαρακτηρισμού στη βάση.
- Ο ακροατής γεγονότων (αν κάτι αλλάξει στη βάση) χρησιμοποιείται στη μέθοδο ανάγνωσης, όπου η τιμή από τη βάση δεδομένων καταχωρείται στο "TextView".
- Στη μέθοδο εγγραφής, διαβάζεται το πεδίο κειμένου και καταχωρείται στη βάση.

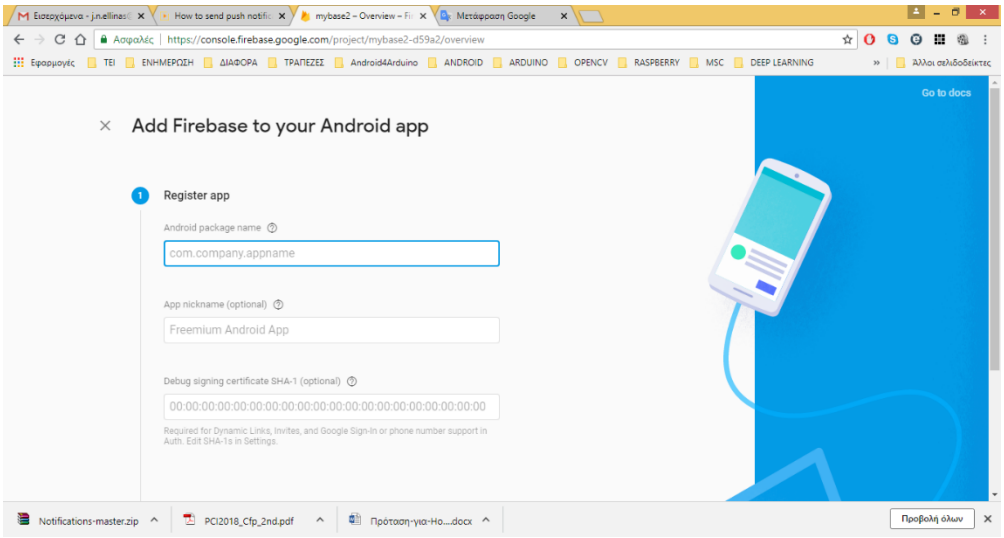
Push Notifications – FCM

Η Firebase, μέσω του FCM (Firebase Cloud Messaging) λειτουργεί σαν μονάδα μεταξύ του διακομιστή σας και των συσκευών που θα λαμβάνουν τις ειδοποιήσεις push που δημιουργείτε. Ο διακομιστής σας ενημερώνει το Firebase ότι πρέπει να αποσταλεί μια ειδοποίηση. Στη συνέχεια, η Firebase πραγματοποιεί τη λειτουργία αυτή για να δημοσιεύσει την ειδοποίηση.

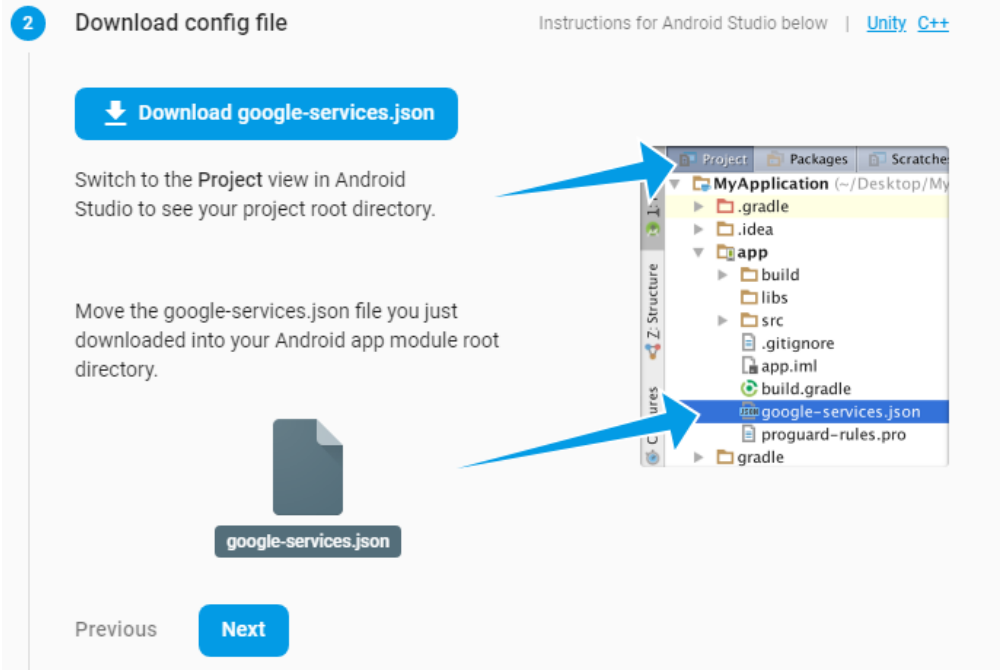
- Δημιουργούμε μια εφαρμογή στο Android Studio: MyFCM.
- Πηγαίνουμε στη Firebase Console: https://console.firebase.google.com/
- Επιλέγουμε το "Add project" (π.χ. mybase2) και το δημιουργούμε.



- Συνδέουμε τη Firebase με την εφαρμογή μας. Επιλέγουμε το εικονίδιο Android και ακολουθούμε τα εξής βήματα:



1. **Register App:** Εγγράφουμε το package name: "gr.mybook.myfcm" και επιλέγουμε "Register app".
2. **Download Config Files:** Επιλέγουμε το κουμπί 'Download google-services.json' για να κατεβάσουμε ένα JSON αρχείο. Το αρχείο αυτό αποθηκεύεται στον φάκελο "app" της εφαρμογής μας. Επιλέγουμε **Next**.



3. **Add Firebase SDK:** Ανοίγουμε το αρχείο <project> build.gradle και ενσωματώνουμε τον κώδικα:

classpath 'com.google.gms:google-services:4.1.0'

Στη συνέχεια ανοίγουμε το αρχείο <app> build.gradle και ενσωματώνουμε τον κώδικα:

```
dependencies {
  // Add this line
  implementation 'com.google.firebase:firebase-core:16.0.3'
//Firebase Messaging library for Sending push Notifications
```

```
implementation 'com.google.firebase:firebase-
messaging:17.3.0'
```

}

Στο τέλος του αρχείου προσθέτουμε τον κώδικα:

```
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

4. Ολοκληρώνοντας κάνουμε Sync την εφαρμογή.

- Επιστρέφουμε στην Firebase Console και επιλέγουμε το "Cloud messaging". Πατάμε το "SEND YOUR FIRST MESSAGE".
- Γράφουμε ένα μήνυμα και επιλέγουμε σαν "app" την εφαρμογή μας "gr.mybook.myfcm".
- Με το "Send Message" ο server της Firebase στέλνει το push notification στο κινητό μας στο οποίο πρέπει να υπάρχει η εφαρμογή που δημιουργήσαμε. Μόλις πατήσουμε το push notification, τρέχει η εφαρμογή μας. Προφανώς πρέπει το κινητό μας να έχει πρόσβαση στο διαδίκτυο.

Υπηρεσία εντοπισμού θέσης (GPS)

Το γνωστό σε όλους GPS (Global Positioning System) μιας κινητής συσκευής χρησιμοποιείται σε συνδυασμό με τους χάρτες της Google για την παροχή υπηρεσιών προσδιορισμού θέσης (LBS-Location Based Services). Οι υπηρεσίες αυτές μπορούν να προσδιορίσουν τη θέση μας, να χαράξουν πορεία για έναν προορισμό, να παρέχουν πληροφορίες για σημεία ενδιαφέροντος που είναι κοντά μας, κλπ.

- Η κύρια συνιστώσα της υπηρεσίας είναι ο LocationManager ο οποίος παρέχει APIs για τον καθορισμό της θέσης και όχι μόνο, όπως:
  - Ερώτημα για τη λίστα όλων των παρόχων τοποθεσίας για την τελευταία γνωστή τοποθεσία χρήστη.
  - Εγγραφή/Ακύρωση εγγραφής για περιοδικές ενημερώσεις της τρέχουσας τοποθεσίας του χρήστη από έναν πάροχο τοποθεσίας.
  - Εγγραφή/Ακύρωση εγγραφής για την ενεργοποίηση πρόθεσης όταν η συσκευή βρίσκεται κοντά (καθορίζεται από την ακτίνα σε μέτρα) σε μια δεδομένη θέση lat/long.
- Η ενεργοποίηση γίνεται με τη μέθοδο *getSystemService(Context.LOCATION_SERVICE)*, η οποία παρέχει ένα "handle" για την υπηρεσία.
- Το αντικείμενο **Location** αντιπροσωπεύει μια γεωγραφική θέση η οποία μπορεί να αποτελείται από γεωγραφικό πλάτος, γεωγραφικό μήκος, χρονική σφραγίδα και άλλες πληροφορίες όπως εκτροπή πορείας, υψόμετρο και ταχύτητα. Υπάρχουν οι

ακόλουθες σημαντικές μέθοδοι που μπορείτε να χρησιμοποιήσετε με το αντικείμενο **Location** για να λάβετε πληροφορίες σχετικά με τη θέση:

- **float distanceTo(Location dest) -** Επιστρέφει την κατά προσέγγιση απόσταση σε μέτρα μεταξύ της τοποθεσίας dest και της δεδομένης θέσης.
- **float getAccuracy() –** Ακρίβεια θέσης σε μέτρα.
- **double getAltitude() –** Ύψος σε μέτρα.
- **float getBearing() –** Εκτροπή πορείας σε μοίρες.
- **double getLatitude() –** Γεωγραφικό πλάτος σε μοίρες.
- **double getLongitude() –** Γεωργικό μήκος σε μοίρες.
- **float getSpeed() –** Ταχύτητα σε m/s.
- **boolean hasAccuracy() –** True εάν η θέση έχει ακρίβεια.
- **boolean hasAltitude() -** True εάν η θέση έχει ύψος.
- **boolean hasBearing() -** True εάν η θέση έχει εκτροπή.
- **boolean hasSpeed() -** True εάν η θέση έχει ταχύτητα.
- **void reset() –** Καθαρίζει τα περιεχόμενα της θέσης.
- **void setAccuracy(float accuracy) –** Καθορίζει την ακρίβεια της θέσης σε μέτρα.
- **void setAltitude(double altitude) –** Καθορίζει το ύψος σε μέτρα.
- **void setBearing(float bearing) -** Καθορίζει την εκτροπή πορείας σε μέτρα.
- **void setLatitude(double latitude) –** Καθορίζει το γ.π. σε μοίρες.
- **void setLongitude(double longitude) -** Καθορίζει το γ.μ. σε μοίρες.
- **void setSpeed(float speed) -** Καθορίζει την ταχύτητα σε m/s.
- **String toString() -** Επιστρέφει μια συμβολοσειρά με περιγραφή αυτού του αντικειμένου.

- Για τη λήψη της τρέχουσας θέσης δημιουργούμε το αντικείμενο **LocationClient,** το οποίο συνδέουμε με την υπηρεσία προσδιορισμού θέσης με τη μέθοδο **connect()** και στη συνέχεια καλούμε τη μέθοδο **getLastLocation().** Αυτή η μέθοδος επιστρέφει την πιο πρόσφατη θέση με τη μορφή αντικειμένου **Location** το οποίο περιέχει συντεταγμένες γεωγραφικού πλάτους και μήκους και άλλες πληροφορίες. Για να υπάρχει λειτουργικότητα στη δραστηριότητά μας με βάση την τοποθεσία, θα πρέπει να εφαρμόσουμε δύο διεπαφές (interfaces):

  ```
  GooglePlayServicesClient.ConnectionCallbacks
  GooglePlayServicesClient.OnConnectionFailedListener
  ```

Αυτές οι διεπαφές παρέχουν τις εξής μεθόδους επανάκλησης:

- **abstract void onConnected(Bundle connectionHint) –** Καλείται όταν η υπηρεσία συνδέεται επιτυχώς με τον **LocationClient.** Για σύνδεση με τον client χρησιμοποιούμε τη μέθοδο **connect().**
- **abstract void onDisconnected() –** Καλείται όταν ο client αποσυνδέεται από την υπηρεσία. Για αποσύνδεση από τον client χρησιμοποιούμε τη μέθοδο **disconnect()**.
- **abstract void onConnectionFailed(ConnectionResult result) –** καλείται όταν προκύπτει σφάλμα κατά τη σύνδεση του client με την υπηρεσία.

Πρέπει να δημιουργήσουμε τον location client στη μέθοδο onCreate() της δραστηριότητάς και, στη συνέχεια, να τον συνδέσουμε στην onStart(), έτσι ώστε οι υπηρεσίες θέσης να διατηρούν την τρέχουσα θέση ενώ η δραστηριότητα είναι πλήρως ορατή. Θα πρέπει να αποσυνδέσουμε τον location client στη μέθοδο onStop(), έτσι ώστε όταν η εφαρμογή δεν είναι ορατή, οι υπηρεσίες θέσης δεν διατηρούν την τρέχουσα θέση. Αυτό βοηθά στην εξοικονόμηση ενέργειας μπαταρίας.

- Για την ανανέωση του εντοπισμού της θέσης, εκτός από τις παραπάνω διεπαφές, χρειάζεται επίσης η διεπαφή **LocationListener.** Αυτή η διεπαφή παρέχει την εξής μέθοδο επανάκλησης:
  - **abstract void onLocationChanged(Location location) -** Χρησιμοποιείται για τη λήψη ειδοποιήσεων από τον **LocationClient** όταν η θέση έχει αλλάξει.
- Το αντικείμενο **LocationRequest** χρησιμοποιείται για τη ρύθμιση παραμέτρων στην ανανέωση του εντοπισμού θέσης από τον **LocationClient.**
  - **setExpirationDuration(long millis) –** Ορίζεται η διάρκεια αυτού του αιτήματος, σε χιλιοστά του δευτερολέπτου.
  - **setExpirationTime(long millis) –** Ορίζεται ο χρόνος λήξης του αιτήματος, σε χιλιοστά του δευτερολέπτου από την εκκίνηση.
  - **setFastestInterval(long millis) –** Ορίζεται το ταχύτερο διάστημα για ενημερώσεις τοποθεσίας, σε χιλιοστά του δευτερολέπτου.
  - **setInterval(long millis) –** Ορίζεται το επιθυμητό διάστημα για ενημερώσεις ενεργών τοποθεσιών σε χιλιοστά του δευτερολέπτου.
  - **setNumUpdates(int numUpdates) –** Ορίζεται ο αριθμός των ενημερώσεων τοποθεσίας.
  - **setPriority(int priority) -** Ορίζεται η προτεραιότητα του αιτήματος.

Για παράδειγμα, αν μια εφαρμογή θέλει υψηλή ακρίβεια θέσης, θα πρέπει να δημιουργήσει μια αίτηση θέσης με setPriority(int) με PRIORITY_HIGH_ACCURACY και setInterval(long) σε 5 δευτερόλεπτα. Μπορούμε επίσης να χρησιμοποιήσουμε μεγαλύτερο χρονικό διάστημα και/ή άλλες προτεραιότητες όπως PRIORITY_LOW_POWER για να ζητήσουμε ακρίβεια σε επίπεδο πόλης.

- Όταν έχουμε το αντικείμενο **Location**, μπορούμε να χρησιμοποιήσουμε τη μέθοδο **Geocoder.getFromLocation()** για να λάβουμε μια διεύθυνση για ένα δεδομένο γεωγραφικό πλάτος και μήκος. Αυτή η μέθοδος είναι σύγχρονη και μπορεί να πάρει πολύ χρόνο για εκτέλεση, γι 'αυτό πρέπει να καλέσουμε τη μέθοδο από τη μέθοδο **doInBackground()** μιας κλάσης **AsyncTask**. Η AsyncTask πρέπει να χρησιμοποιηθεί ως υποκλάση η οποία θα αντικαταστήσει τη μέθοδο **doInBackground(Params ...)** για να εκτελέσει μια εργασία στο παρασκήνιο. Η μέθοδος **onPostExecute(Result)** ενεργοποιείται στο νήμα UI αφού τελειώσει ο υπολογισμός στο παρασκήνιο και τη στιγμή εμφάνισης του αποτελέσματος. Υπάρχει μια ακόμα σημαντική μέθοδος διαθέσιμη στο AyncTask, η **execute(Params ... params)**, η οποία εκτελεί την εργασία με τις καθορισμένες παραμέτρους.

- Για τη δημιουργία μιας εφαρμογής υπηρεσίας εντοπισμού θέσης πρέπει να λάβουμε υπ' όψη τα εξής:

  Ενεργοποίηση του αντικειμένου χειρισμού της υπηρεσίας (LocationManager).

```
// Reference to the system Location Manager
LocationManager locationManager = (LocationManager)
      this.getSystemService(Context.LOCATION_SERVICE);
```

Ενεργοποίηση του ακροατή γεγονότων σχετικά με την ανανέωση θέσης.

```
// Define a listener that responds to location updates
LocationListener locationListener = new
         LocationListener() {
    public void onLocationChanged(Location location) {
      // Called when a new location is found by provider.
      double lat=(double)(location.getLatitude());
      double lng=(double)(location.getLongitude());
      latText.setText(String.valueOf(lat));
      longText.setText(String.valueOf(lng));
    }

    public void onStatusChanged(String provider, int
         status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
  };

// Register the listener with the Location Manager
// Peceive location updates
locationManager.requestLocationUpdates(LocationManager.NE
TWORK_PROVIDER, 0, 0, locationListener);
```

Λήψη της τελευταίας γνωστής θέσης από κάποιον πάροχο.

```
String locationProvider =
LocationManager.NETWORK_PROVIDER;
// Or use LocationManager.GPS_PROVIDER

Location lastKnownLocation =
locationManager.getLastKnownLocation(locationProvider);
```

Τερματισμός των ανανεώσεων ενημέρωσης θέσης.

```
// Remove the listener you previously added
locationManager.removeUpdates(locationListener);
```

- Στο αρχείο μανιφέστου πρέπει να προστεθούν οι αιτήσεις έγκρισης για δίκτυο και υπηρεσίας εντοπισμού θέσης.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Για λειτουργικό σύστημα μεγαλύτερο του 6 (Marshmallow) υπάρχουν και τα runtime permissions, τα οποία επιβεβαιώνουν την άδεια χρήσης η οποία έχει χορηγηθεί στο αρχείο μανιφέστου ή τη δημιουργούν αν δεν υπάρχει. Ο έλεγχος γίνεται με:

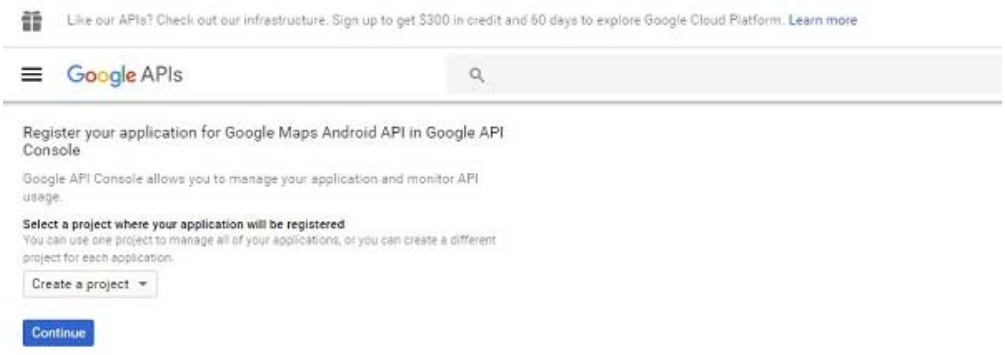
```
if (ContextCompat.checkSelfPermission(thisActivity,
            Manifest.permission.ACCESS_FINE_LOCATION)
   != PackageManager.PERMISSION_GRANTED) {
    // Permission is not granted
}
```

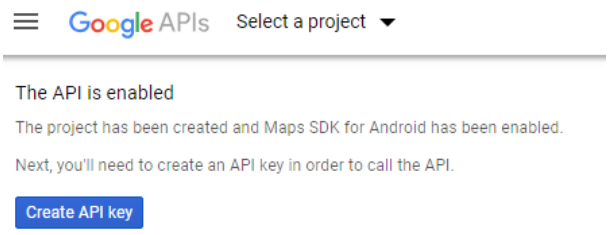
Η εφαρμογή myLBS προσδιορίζει την τρέχουσα θέση της συσκευής όταν είναι ενεργοποιημένο το GPS. Το UI είναι:

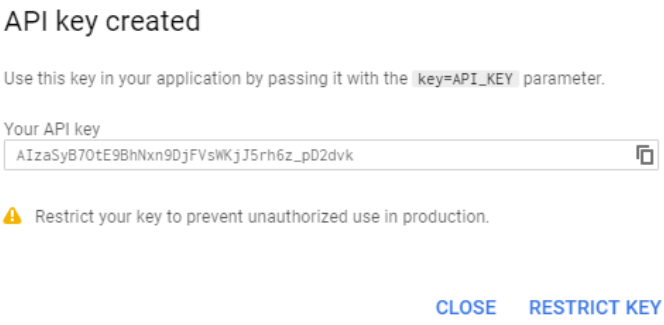
Χάρτες Google

Μπορούμε σε μια εφαρμογή μας να ενσωματώσουμε τους χάρτες Google και να κάνουμε διάφορους χειρισμούς (μεγέθυνση, σμίκρυνση, τοποθέτηση δεικτών, κλπ). Δημιουργούμε την εφαρμογή "myMap" για να δούμε τον τρόπο χρήσης των χαρτών αλλά και για εξάσκηση με τους χάρτες Google.

- Δημιουργούμε το project με το παραπάνω όνομα επιλέγοντας εφαρμογή με χάρτες Google (Google Maps Activity). Δημιουργούνται τα αρχεία "MapsActivity.java", "activity_maps.xml" και "google_maps_api.xml".
  - Στο περιγραφικό αρχείο "google_maps_api.xml" δίνεται μια διεύθυνση για την απόκτηση ενός API key για χρήση των χαρτών (https://console.developers.google.com/flows/.....). Πατάμε "Continue" και στη συνέχεια "Create API key".







- Τοποθετούμε το κλειδί στην κατάλληλη θέση του περιγραφικού αρχείου "google_maps_API.xml".
- Παρατηρήστε τη μορφή του "activity_maps.xml" το οποίο είναι ένα fragment δίνοντας όλη την οθόνη για απεικόνιση των χαρτών.
- Παρατηρήστε το αρχείο μανιφέστου. Το API key έχει περάσει στο αρχείο αυτό και έχει δοθεί άδεια για "ACCESS_FINE_LOCATION".
- Παρατηρήστε τον κώδικα στο αρχείο "MapsActivity.java", όπου τοποθετεί έναν marker σε δεδομένο (lat, long). Βρείτε το αντίστοιχο (lat, long) της Ακρόπολης και σημειώστε "ACROPOLIS OF ATHENS". Θα εμφανιστεί όταν πατηθεί ο marker. Οι συντεταγμένες φαίνονται στο url. Η μέθοδος που χρησιμοποιείται είναι η:

  ```
  newLatLng(lat, long)
  ```

  Χρησιμοποιήστε επίσης τη μέθοδο:

  ```
  newLatLngZoom(lat, long, 10)
  ```

  και παρατηρήστε τα αποτελέσματα.