ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εφαρμογή για Android "Personal Trainer"

Android application "Personal Trainer"

Διπλωματική Εργασία

Λάλος Απόστολος

Επιβλέπων Καθηγητής

Ακρίτας Αλκιβιάδης

Δεύτερο μέλος Επιτροπής

Σταμούλης Γεώργιος

Βόλος, 2016

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας θα ήθελα αρχικά να ευχαριστήσω τον Καθηγητή, Κύριο Ακρίτα Αλκιβιάδη που μου έδωσε την ευκαιρία εκπόνησης μιας τέτοιου είδους ενδιαφέρουσας εργασίας η οποία αν μη τι άλλο με εξόπλισε με πολλές χρήσιμες γνώσεις.

Επίσης ένα μεγάλο ευχαριστώ από καρδιάς στην οικογένειά μου που μου συμπαραστάθηκε καθ' όλη την διάρκεια ολοκλήρωσής της, καθώς επίσης και σε όλους τους φίλους μου που όλα αυτά τα χρόνια με βοήθησαν ψυχικά και μου συμπαραστάθηκαν να ξεπεραστούν οι δυσκολίες της σχολής.

Περίληψη

Είναι γνωστό πως στην εποχή που ζούμε η τεχνολογία έχει διεισδύσει για τα καλά στην ζωή μας με σκοπό την διευκόλυνση και βελτίωσή της. Οι άνθρωποι προτιμούν να χρησιμοποιούν μία μόνο συσκευή για να συνδυάσουν δουλειά - επικοινωνία - διασκέδαση. Έτσι έχουμε την εμφάνιση των έξυπνων κινητών τηλεφώνων και των tablets, τις πανίσχυρες αυτές συσκευές με αμέτρητες εφαρμογές να διαλέξει και να χρησιμοποιήσει κανείς. Σε πολλούς ανθρώπους αρέσει να γυμνάζονται και αφιερώνουν ένα μεγάλο μέρος του χρόνου τους καθημερινά στα γυμναστήρια, ωστόσο το να θυμάται κανείς και να διαλέγει ποιές από όλες τις ασκήσεις να κάνει και ποιά μέρα ίσως για κάποιους να είναι δύσκολο και να τους μπερδεύει.

Στην παρούσα διπλωματική εργασία θα παρουσιαστεί η ανάπτυξη μιας τέτοιου είδους εφαρμογής για λειτουργικό Android, η οποία έχει σκοπό να παρέχει την δυνατότητα στον χρήστη να δημιουργεί και να οργανώνει το δικό του πρόγραμμα γυμναστικής, να το επεξεργάζεται και να το διαγράφει όπως αυτός θέλει, εύκολα και γρήγορα, καθώς και να του παρουσιάζει πως να ολοκληρώνει σωστά μια άσκηση γυμναστικής. Η εργασία δημιουργήθηκε πάνω στο περιβάλλον Android Studio.

Abstract

Nowadays, it is known that technology has entered in our lives for good to make them easier and better. People prefer to use one device to combine work - communication - entertainment. So we have the creation of smartphones and tablets, such a powerful small device with countless applications to pick and use. Many of the people like to workout and spend a good amount of their daily time at the gyms , though keeping track of all these exercises they have to perform each day of the week for someone may be confusing and or difficult.

In this thesis I 'm going to present the development of such an application for Android OS, to give the user the ability to create and manage his own Workout Plan(Schedule), to edit it and deleted it as he likes easily and quickly. He will also have the ability to preview how a specific exercise is done correctly. This project has been created on Android Studio environment.

Περιεχόμενα

1 Εισαγωγή στο Android Studio

Το Android Studio είναι ένα περιβάλλον (IDE) για ανάπτυξη εφαρμογών για την πλατφόρμα Android που πρωτο ανακοινώθηκε τον Μάιο του 2013 από την Google με την πρώτη σταθερή έκδοση τον Δεκέμβριο του 2014. Το Android Studio παρέχεται εντελώς δωρεάν σε όλους τους χρήστες.

Βασισμένο στο IntelliJ IDEA και γραμμένο σε Java, το Android Studio έχει σχεδιαστεί ειδικά για την ανάπτυξη android εφαρμογών. Διατίθεται για λειτουργικό Windows, Mac OS X και Linux .

Επιπλέον με όλες τις δυνατότητες που προσφέρει το IntelliJ IDEA, το Android Studio προσθέτει μερικές νέες όπως:

- Gradle-based σύστημα δημιουργίας.
- Εύκολη παραγωγή αρχείων apk.
- Χρήσιμα πρότυπα κώδικα για τις εφαρμογές.
- Εύχρηστο εργαλείο επεξεργασίας του γραφικού περιβάλλοντος (layout editor), με drag-and-drop δυνατότητα επεξεργασίας θεμάτων.
- Υποστήριξη για την πλατφόρμα Google Cloud
- Και πολλά άλλα

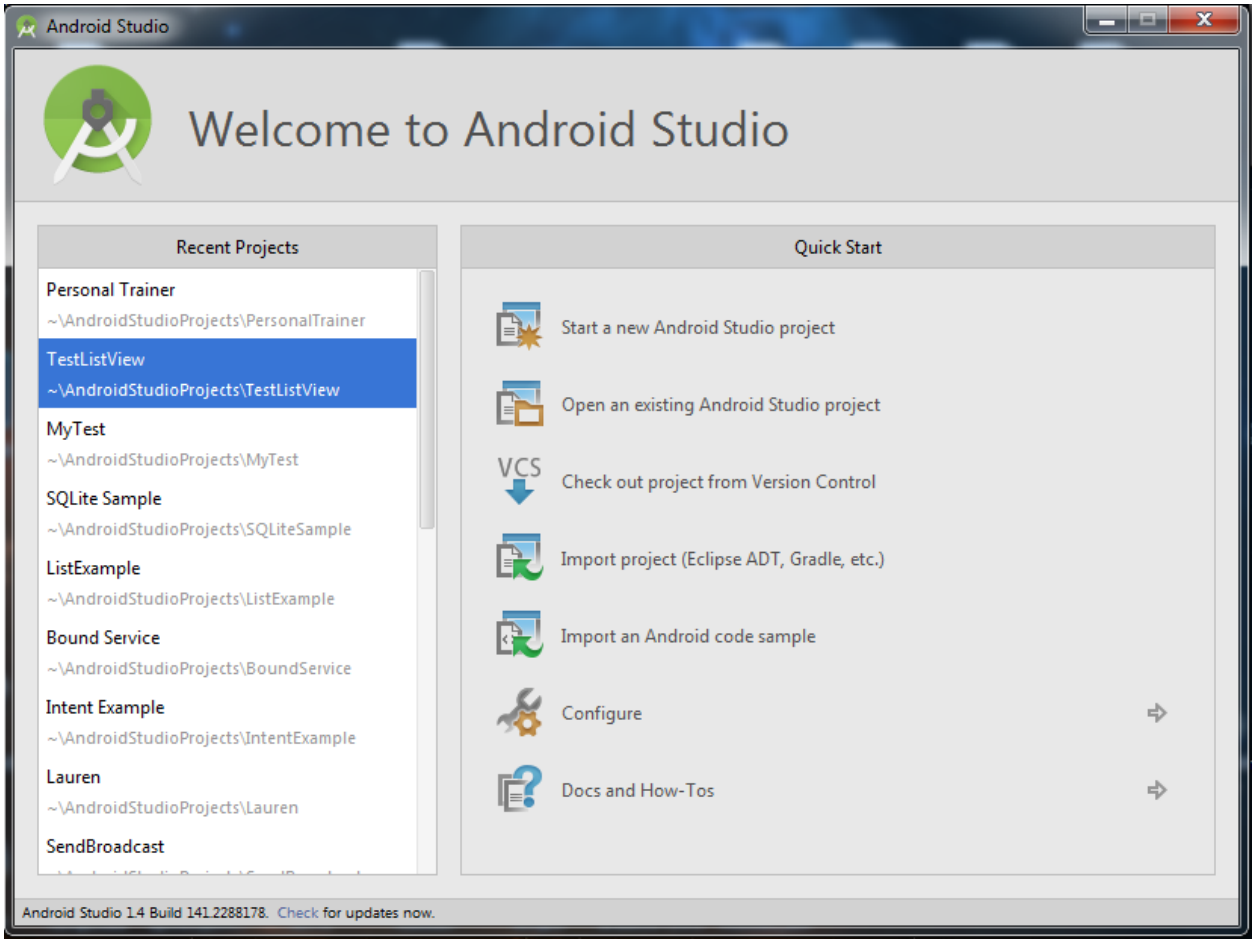
Το Android Studio μπορεί κανείς να το κατεβάσει από εδώ
Οδηγίες εγκατάστασης μπορεί να τις βρει εδώ (επίσης SDK packages)

Τα προαπαιτούμενα συστήματος για την εγκατάσταση του περιβάλλοντος είναι :

	Windows	OS X	Linux
OS version	Microsoft Windows 10/8.1/8/7/Vista/2003/XP (32 or 64 bit)	Mac OS X 10.8.5 or higher, up to 10.10 to up 10.10.2 up 10.10.3 on 10.10.5 (Yosemite)	GNOME or KDE or Unity desktop on Ubuntu or Fedora or GNU/Linux Debian
RAM	2 GB RAM minimum, 4 GB RAM recommended		
Disk space	500 MB disk space		
Space for Android SDK	At least 1 GB for Android SDK, emulator system images, and caches		
JDK version	Java Development Kit (JDK) 7 or higher		
Screen resolution	1280x800 minimum screen resolution		

1.1 Προετοιμασία

Εφόσον το Android Studio έχει εγκατασταθεί με επιτυχία ο χρήστης πρέπει να βεβαιωθεί ότι το περιβάλλον είναι ενημερωμένο και διασυνδεδεμένο με την σωστή έκδοση Java.

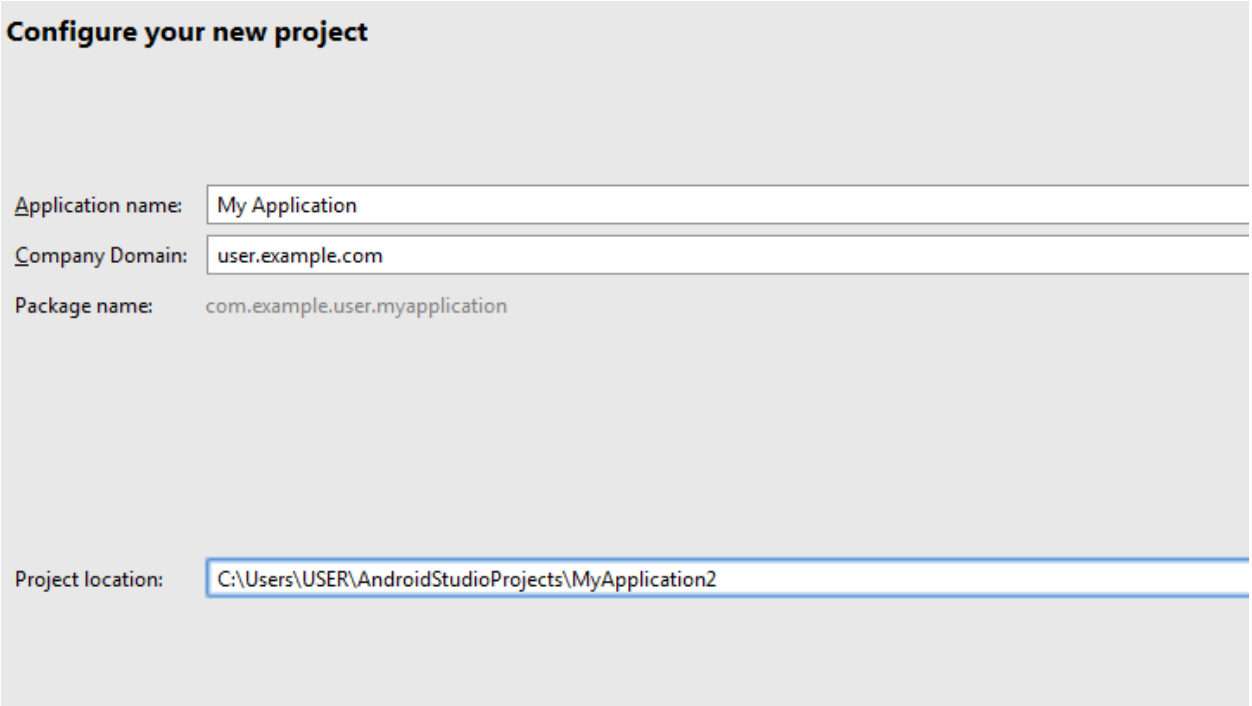


Εικόνα 1.1 Αρχικό μενού του περιβάλλοντος Android Studio.

Στην εικόνα 1.1 βλέπουμε την αρχική οθόνη του Android Studio όπου στα αριστερά έχουμε την λίστα των projects που έχουν δημιουργηθεί μέχρι τώρα, ενώ στα δεξιά το μενού επιλογών που προσφέρονται στον χρήστη. Για οποιαδήποτε αλλαγή επιλογών του περιβάλλοντος ο χρήστης πατάει στην επιλογή **Configure**.

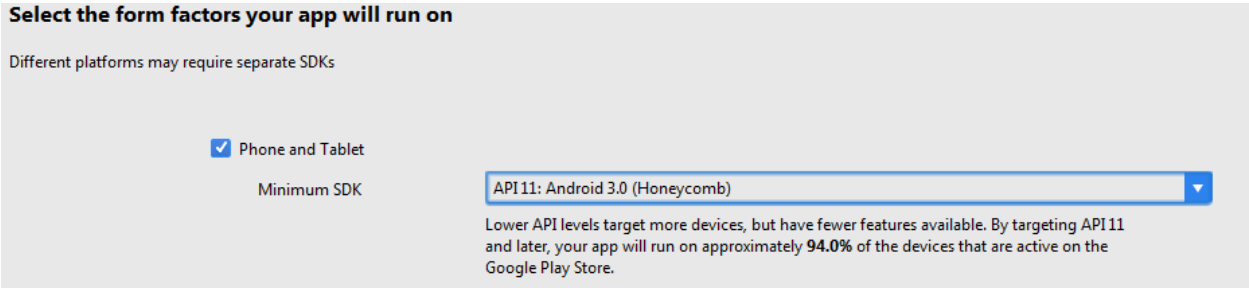
Για να δημιουργήσουμε ένα νέο project επιλέγουμε **Start a new Android Studio project** και μία νέα καρτέλα θα μας παρουσιαστεί στην οθόνη (βλ. εικόνα 1.2). Ο χρήστης καλείται να επιλέξει ένα όνομα για την εφαρμογή(**Application name**) καθώς και ένα όνομα πακέτου(**Domain**). Να σημειώσουμε ότι το όνομα εφαρμογής δεν είναι απλώς ένα προσωρινό όνομα αλλά αυτό που θα φαίνεται αν επιθυμούμε να την ανεβάσουμε και στο Google Play Store αργότερα.

Για το παρόν project επιλέχτηκε ως όνομα το **Personal Trainer.**



Εικόνα 1.2 Δημιουργία νέου project

Στην επόμενη καρτέλα(βλ. εικόνα 1.3) πρέπει να σκευτούμε για ποιές συσκευές προορίζεται η εφαρμογή μας. Μιας και προορίζουμε την εφαρμογή για συσκευές κινητών τηλεφώνων επιλέγουμε **Phone and Tablet** και στην συνέχεια επιλέγουμε ένα από τα API στην λίστα. Στην παρούσα εργασία επιλέγουμε API ίσο με 11 (Android 3.0 Honeycomb).



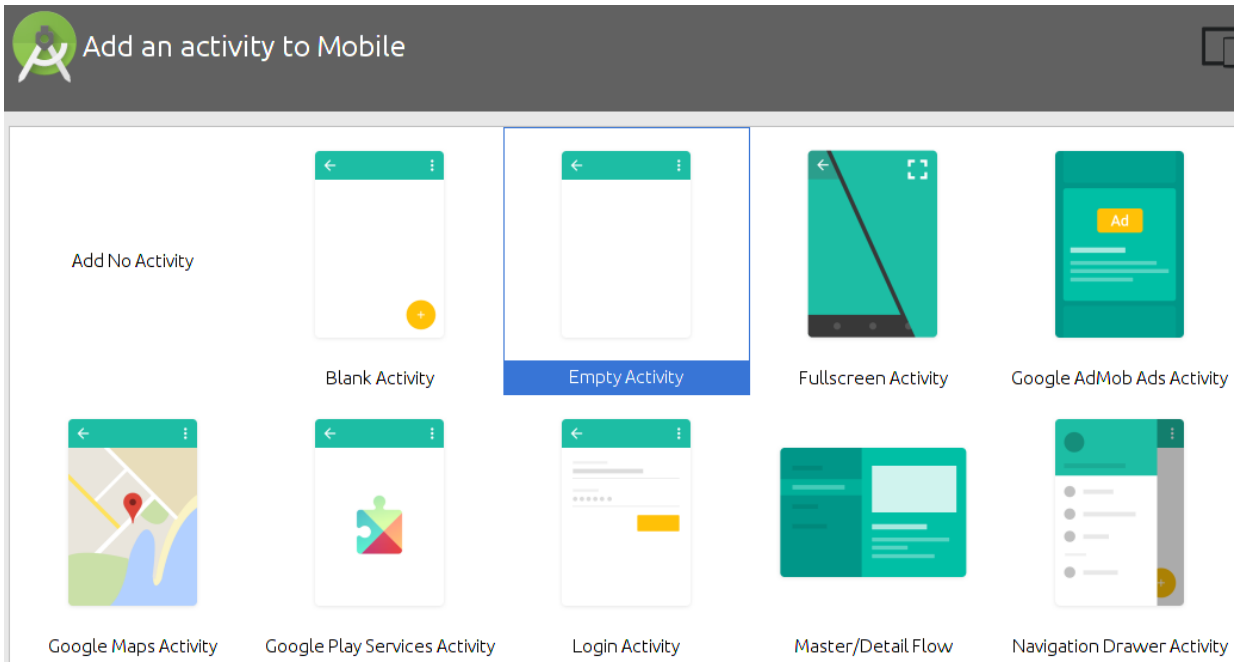
Εικόνα 1.3 Επιλογή κατάλληλου API

Σημείωση: Η επιλογή του κατάλληλου API είναι πολύ σημαντική και θέλει προσοχή γιατί αργότερα με τον τρόπο που θα δημιουργήσουμε την εφαρμογή μας κάποια στοιχεία μπορεί να μην είναι συμβατά με μικρότερο API από το προκαθορισμένο για αυτά, με αποτέλεσμα σε κάποιες συσκευές με παλαιότερη έκδοση λογισμικού να έχουν πρόβλημα συμβατότητας με την εφαρμογή.

Αφού επιλέξαμε και το API το επόμενο βήμα είναι να επιλέξουμε την βασική αρχική Activity(Δραστηριότητα) για την εφαρμογή μας.

Ένα από τα πολλά πλεονεκτήματα που προσφέρει το Android Studio είναι ότι παρέχει στον χρήστη μια μεγάλη γκάμα από προκαθορισμένα Activities για να διαλέξει ανάλογα με τον τύπο εφαρμογής που δημιουργεί (βλ. εικόνα 1.4).

Για την εργασία μας επιλέχτηκε το **Blank Activity**.

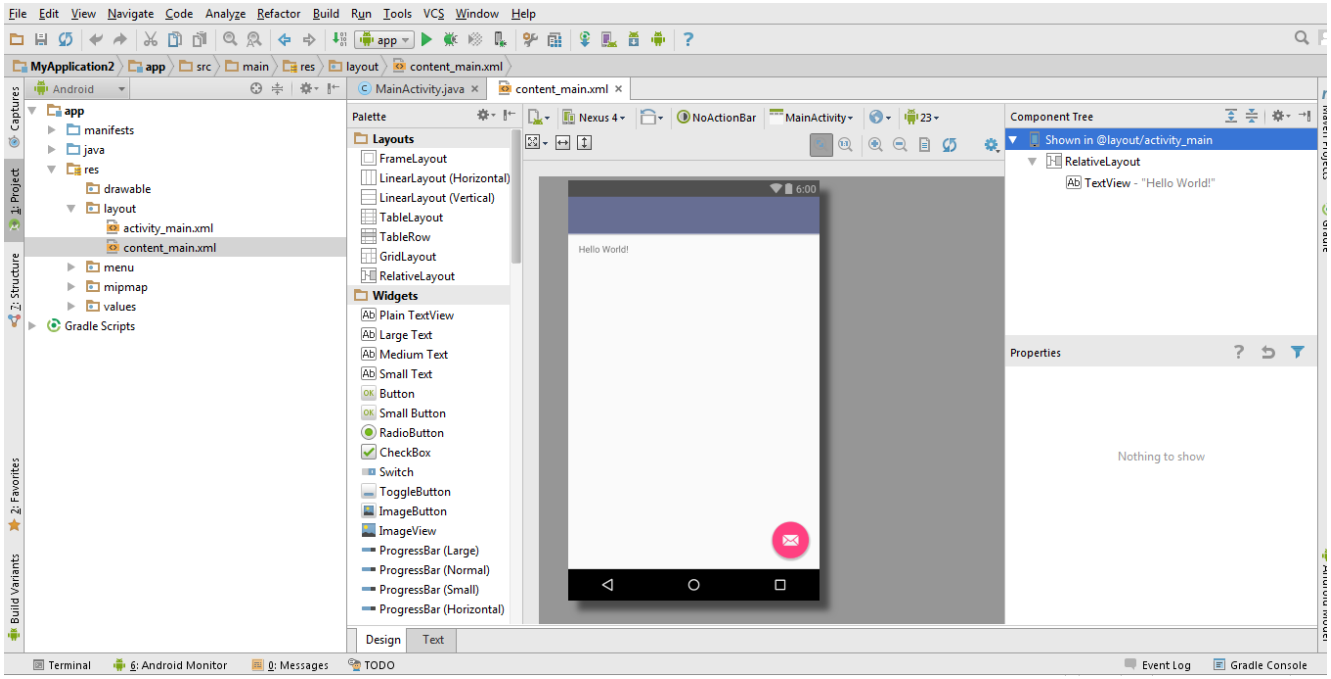


Εικόνα 1.4 Επιλογή του Activity(Δραστηριότητα)

Έτσι ολοκληρώσαμε την διαδικασία προετοιμασίας για το project μας. Στην συνέχεια ακολουθεί επεξήγηση του περιβάλλοντος με τα πιο κύρια σημεία του.

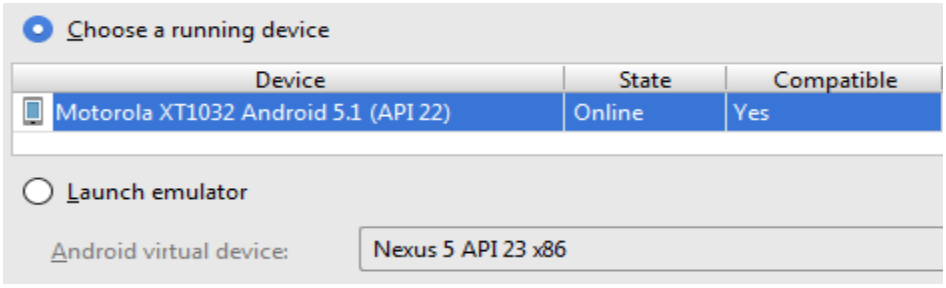
1.2 Το Περιβάλλον

Αφού το Android Studio ολοκληρώσει την διαδικασία τακτοποίησης του project μας θα έχουμε μια οθόνη σαν αυτή της εικόνας 1.5.



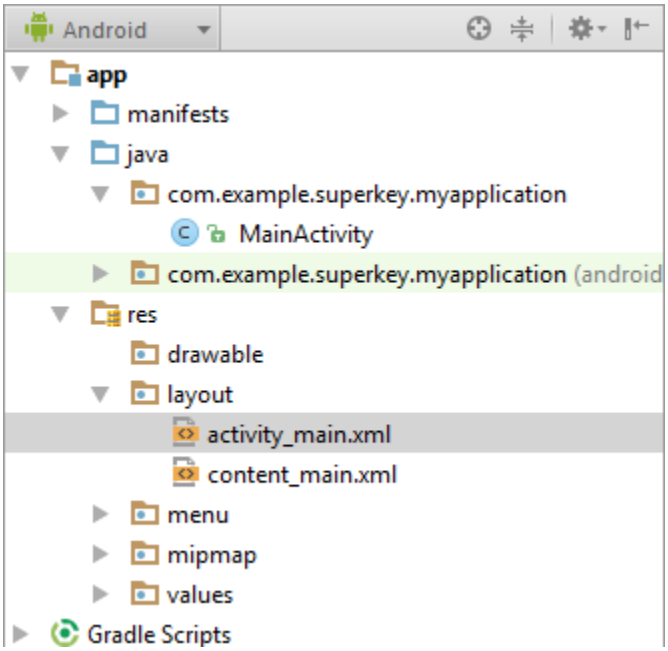
Εικόνα 1.5 Η κύρια οθόνη δημιουργίας της εφαρμογής

Για να τρέξουμε οποιαδήποτε στιγμή την εφαρμογή μας πατάμε το κουμπί  από το μενού με τα εργαλεία. Και επιλέγουμε είτε emulator είτε κάποια συνδεδεμένη συσκευή (καλώδιο USB).



Εικόνα 1.6 Τρέξιμο της εφαρμογής σε συσκευή

Στην αριστερή περιοχή έχουμε την ιεραρχική δομή όλων των αρχείων στο Project μας(βλ. εικόνα 1.7). Αρχικά έχουμε τα αρχεία **AndroidManifest.xml** τα οποία περιλαμβάνουν πολύ σημαντικές πληροφορίες για την εφαρμογή, επίσης περιέχει όλα τα ονόματα των κλάσεων που έχουμε δημιουργήσει και τις ιδιότητές τους.



Εικόνα 1.7 Δομή αρχείων στο project

Στο φάκελο **Java** βλέπουμε το πακέτο(package) που δημιουργήσαμε προηγουμένως και μέσα σε αυτό θα τοποθετούνται όλα τα **.java** αρχεία κλάσεων. Προς το παρόν μόνο το **MainActivity.java** βρίσκεται εκεί.

Να τονίσουμε ότι τα αρχεία java περιέχουν τον κώδικα για το πώς λειτουργεί η εφαρμογή. Όπως προαναφέρθηκε, μια εφαρμογή αποτελείται από πολλά Activities. Προς το παρόν ας σκευτούμε κάθε Activity σαν μια "οθόνη" στην συσκευή. Κάθε Activity μέσα στην εφαρμογή αποτελείται από δυο μέρη:

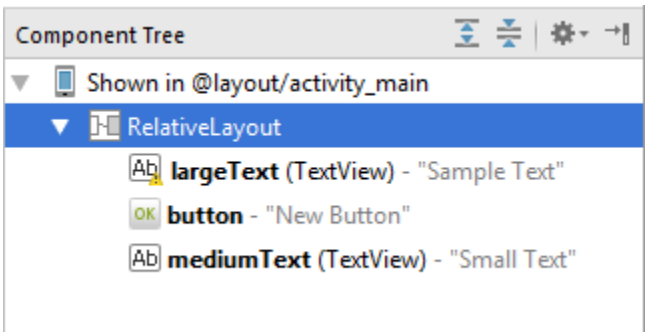
Το πρώτο μέρος είναι το γραφικό περιβάλλον όπως βλέπουμε στην εικόνα 1.8 το οποίο περιλαμβάνει τα στοιχεία που έχουν να κάνουν με το σχεδιαστικό μέρος και το βλέπουμε στην οθόνη της συσκευή πχ. χρώματα, εικόνες, κουμπιά, κείμενο κτλ.

Εικόνα 1.8 Οθόνη σχεδιασμού

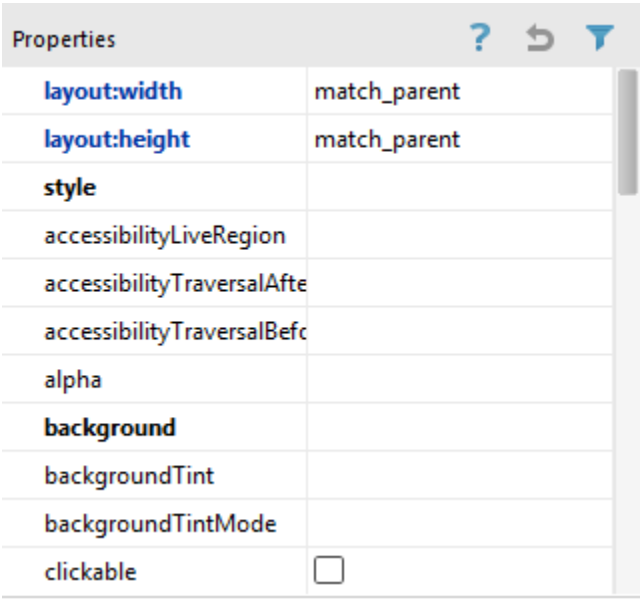
Τα αρχεία που είναι υπεύθυνα για αυτό είναι τα activity_main.xml και content_main.xml και βρίσκονται στον φάκελο **layout**(βλ. εικόνα 1.7). Να τονίσουμε ότι αυτά τα αρχεία **δεν** περιέχουν κώδικα με συναρτήσεις και το πως λειτουργεί το πρόγραμμα. Ο εγκέφαλος(παρασκήνιο) είναι το αντίστοιχο **.java** αρχείο MainActivity.java που καθορίζει την συμπεριφορά των Activities.

Οπότε κάθε φορά που δημιουργούμε ένα καινούργιο Activity δημιουργούμε αυτά τα τρία αρχεία **activity_name.xml , content_name.xml και NameActivity.java .**

Στο δεξί μέρος του περιβάλλοντος του Android Studio έχουμε την δόμηση μιας Activity όπως παρουσιάζεται στην παρακάτω εικόνα 1.9a και από κάτω έχουμε τις ιδιότητες και παραμέτρους για καθένα από αυτά τα στοιχεία(components) όπως στην εικόνα 1.9b.



Εικόνα 1.9a Δομημένη λίστα στοιχείων σε μια Activity



Εικόνα 1.9b Επεξεργασία παραμέτρων για κάθε στοιχείο στην Activity

Στα πλαίσια αυτής της εργασίας δεν θα γίνει περεταίρω αναλυτική περιγραφή σχετικά με τα αμέτρητα εργαλεία που προσφέρει αυτό το περιβάλλον, απλώς δόθηκε μια γενική αναφορά των πιο βασικών στοιχείων για να υπάρχει μια οικειότητα με αυτό.

Καθώς θα προχωράμε στην ανάπτυξη της εφαρμογής σταδιακά, όπου κρίνεται αναγκαίο θα γίνει και επιπλέον επεξήγηση του Android Studio περιβάλλοντος.

2 Τι είναι μια Δραστηριότητα(Activity)

Μια Δραστηριότητα(Activity) είναι μια "οθόνη" στην οποία μπορεί να κάνει κάτι ο χρήστης. Σχεδόν όλες οι Δραστηριότητες αλληλεπιδρούν με τον χρήστη, οπότε η Activity Class (κλάση) φροντίζει να δημιουργεί ένα παράθυρο μέσα στο οποίο και υλοποιούμε το γραφικό μας περιβάλλον(**UI**) και το εφαρμόζουμε με την εντολή:

```
setContentView(View);
```

Τα Activities παρόλο που τις περισσότερες φορές παρουσιάζονται ως πλήρης-οθόνης(**full-screen**) παράθυρα, μπορούν επίσης να χρησιμοποιηθούν και ως μετακινούμενα παράθυρα (**floating-window**) ή και ως μέρος μιας άλλης Δραστηριότητας (**ActivityGroup**).

Υπάρχουν δύο μέθοδοι που σχεδόν κάθε υποκλάση μιας Activity πρέπει να υλοποιεί:

- `onCreate(Bundle)`
- `onPause()`

Η πρώτη έχει να κάνει με την αρχικοποίηση της Δραστηριότητας διασυνδέοντάς τη προγραμματιστικά με το αντίστοιχο αρχείο για το γραφικό περιβάλλον(**UI**) που φτιάξαμε καλώντας την συνάρτηση:

```
setContentView(R.layout.myUI);
```

και ανακτώντας τα επιθυμητά widgets σε αυτό το περιβάλλον με την

```
findViewById(R.id.myWidget);
```

Η **onPause()** διαχειρίζεται την κατάσταση όταν ο χρήστης εγκαταλείψει προσωρινά την Δραστηριότητα. Όλες οι αλλαγές που έχουν γίνει στην τρέχουσα Δραστηριότητα αποθηκεύονται στον χώρο δεδομένων για αυτήν την Δραστηριότητα έως ότου γίνει **Resume**.

Κάθε κλάση Activity που δημιουργείται θα πρέπει να δηλώνεται στο αρχείο **AndroidManifest.xml.**

2.1 Activity Lifecycle (Κύκλος ζωής)

Το σύστημα διαχειρίζεται και τοποθετεί κάθε Activity σε μια **σορό(στοίβα)** από Δραστηριότητες (**Activity Stack**). Κάθε φορά που μια νέα Activity ξεκινάει, τοποθετείται στην κορυφή της **σορού(stack)** και ορίζεται ως η **τρέχουσα** Δραστηριότητα, με όλες της προηγούμενες να ακολουθούν με την σειρά που ξεκίνησαν από κάτω στην σορό.

Κάθε Activity έχει ουσιαστικά τέσσερεις καταστάσεις:

- Αν βρίσκεται στο προσκήνιο(κορυφή στοίβας) τότε λέμε ότι είναι **ενεργή**(**τρέχουσα**).
- Αν έχει χάσει την εστίαση(**focus**) αλλά εξακολουθεί να είναι μερικώς ορατή, τότε λέμε ότι είναι **σε παύση**(**paused**).
  Μια τέτοια Δραστηριότητα αν μη τι άλλο είναι εντελώς "ζωντανή" αλλά μπορεί σε μερικές περιπτώσεις χαμηλής διαθεσιμότητας σε

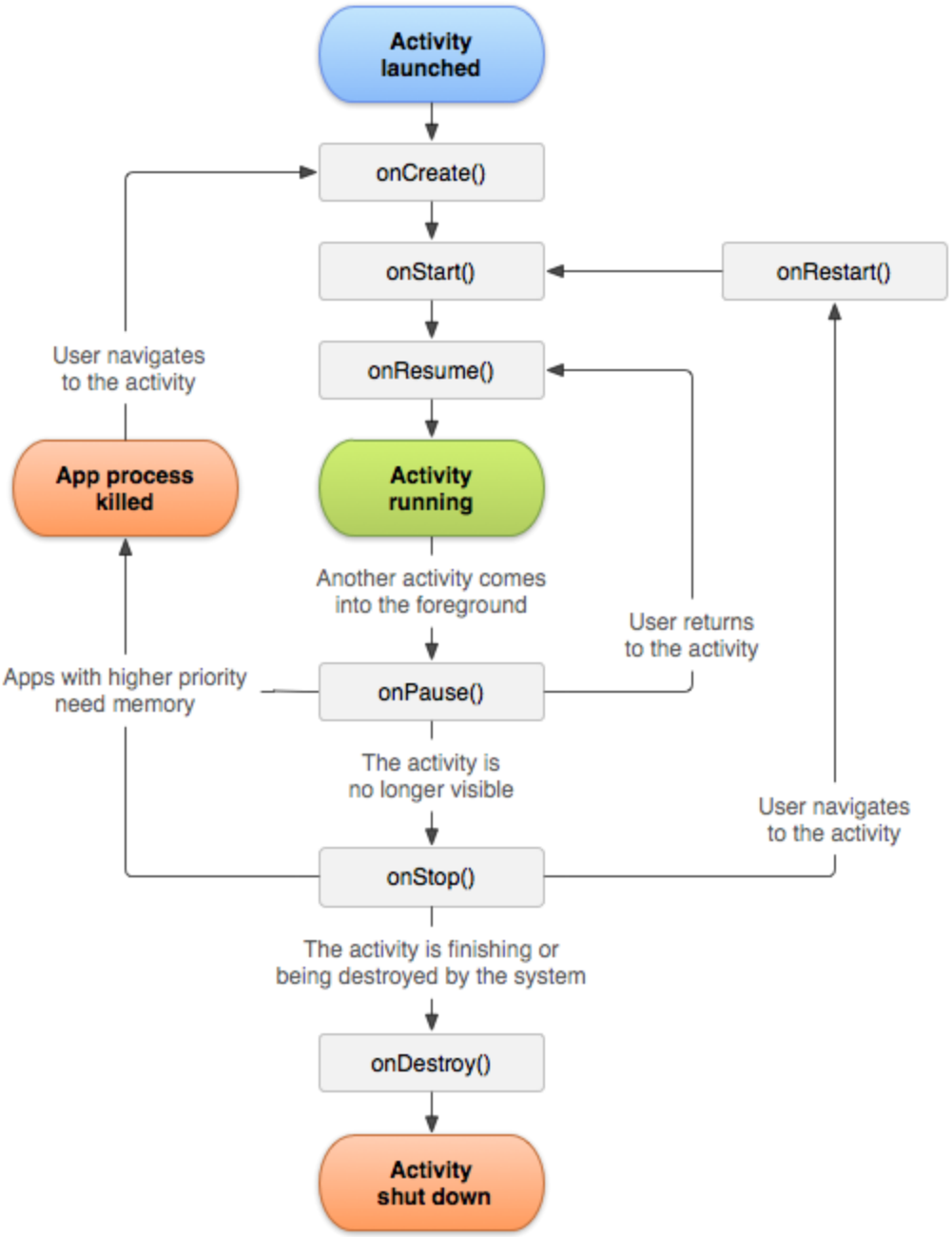
μνήμη το σύστημα να την τερματίσει για να απελευθερώσει πόρους.

- Αν έχει επισκιαστεί εντελώς από μία άλλη Δραστηριότητα τότε λέμε ότι είναι **σταματημένη**(**stopped**).
  Σε αυτήν την περίπτωση εξακολουθεί να διατηρεί την κατάσταση και τις πληροφορίες, ωστόσο, δεν είναι πλέον ορατή στο χρήστη.
- Στην περίπτωση που είναι **σε παύση**(**paused**) ή **σταματημένη**(**stopped**) το σύστημα μπορεί ανά πάσα στιγμή να την απελευθερώσει από την μνήμη αν κριθεί αναγκαίο. Σε αυτήν την περίπτωση πρέπει αυτή η Δραστηριότητα να επανεκκινηθεί εξαρχής και να επαναφερθεί στην προηγούμενή της κατάσταση.

Το παρακάτω διάγραμμα(**σχήμα 2**) παρουσιάζει το σημαντικό **μονοπάτι καταστάσεων.**

Με τα ορθογώνια κουτιά παρουσιάζονται οι μέθοδοι επανάκλησης(**callback methods**) που μπορούν να υλοποιηθούν για εκτέλεση ενεργειών όταν η Δραστηριότητα μεταβαίνει μεταξύ καταστάσεων.

Τα χρωματιστά Οβάλ κουτιά αντιπροσωπεύουν τις καταστάσεις που μπορεί να βρίσκεται μια Activity.



Σχήμα 2 Μονοπάτι καταστάσεων

Ολόκληρος ο **κύκλος ζωής** ορίζεται από τις παρακάτω μεθόδους, που θα πρέπει να καλούμε πάντα την Υπερκλάση(**superclass**) όταν τις υλοποιούμε.

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```

Για μια Δραστηριότητα:

- **onCreate()** καλείται στην **εκκίνηση(Αρχικοποίηση).**
- **onRestart()** καλείται αφού έχει γίνει **σταματημένη**(**stopped**).
- **onStart()** καλείται όταν γίνεται ορατή στον χρήστη.
- **onResume()** καλείται όταν ένα Activity αρχίζει να αλληλεπιδρά με τον χρήστη(επαναφορά στην κορυφή της stack).
- **onPause()** καλείται όταν το σύστημα πρόκειται να επαναφέρει μια προηγούμενη Δραστηριότητα. Σταματάει κάθε τρέχουσα ενέργεια και αποθηκεύει την κατάσταση δεδομένων.
- **onStop()** καλείται όταν δεν είναι ορατή στον χρήστη.
- **onDestroy()** αποτελεί την τελευταία κλήση προτού η Δραστηριότητα τερματιστεί.

3 MainActivity για την εφαρμογή Personal Trainer

Αρχικά η εφαρμογή αποτελείται από μόνο και μια δραστηριότητα (Activity) την **MainActivity** η οποία αποτελεί και την αρχική μας οθόνη όταν ο χρήστης εκκινεί την εφαρμογή.

Στόχος μας είναι να δημιουργήσουμε την αρχική οθόνη όσο πιο απλά γίνεται με μία λίστα επιλογών προς τον χρήστη του τύπου ενός panel με κουμπιά(buttons) τα οποία θα πιέζει και ανάλογα με την επιλογή του θα τον μεταφέρει στην αντίστοιχη νέα οθόνη(νέο Activity).

Πρόκειται να προσθέσουμε τέσσερα λειτουργικά κουμπιά(**buttons**) που θα αναλύσουμε στην συνέχεια:

- Schedule
- Workouts
- Instructions
- About

Τα αρχεία με τα οποία θα ασχοληθούμε και θα κάνουμε τις αλλαγές είναι τα **MainActivity.java, activity_main.xml** και **content_main.xml**.

3.1 Αρχικοποίηση της MainActivity

Να σημειώσουμε ότι κάθε Activity θα πρέπει να περιέχει το παρακάτω βασικό τμήμα κώδικα στο αρχείο **.java**

Αφού δουλεύουμε στο **MainActivity.java** πρέπει να έχουμε

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

}
```

Το παραπάνω κομμάτι κώδικα θα εκτελείται κάθε φορά που μια νέα Activity δημιουργείται και είναι υποχρεωτικό. Η συνάρτηση αυτή

```
protected void onCreate( )
```

αποτελεί την αρχικοποίηση της δραστηριότητας.

Το πιο σημαντικό εδώ είναι ότι καλούμε την συνάρτηση

```
setContentView(R.layout.activity_main);
```

για να τη διασυνδέσουμε με το UI (γραφικό περιβάλλον) ενός layout αρχείου.

Ο υπόλοιπος κώδικας υποδηλώνει την υποστήριξη γραμμής εργαλείων(toolbar) που θα βρίσκεται στο πάνω μέρος της οθόνης.

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

3.2 Ημερομηνία

Δημιουργώντας μια εφαρμογή η οποία περιλαμβάνει πρόγραμμα γυμναστικής θα ήταν πρακτικό και χρήσιμο να δείχνουμε στον χρήστη και την τρέχουσα ημερομηνία. Ένας εύκολος τρόπος είναι να φτιάξουμε μια συνάρτηση που θα λαμβάνει σήμα από εσωτερικά της συσκευής και θα το τυπώνει στην οθόνη.

Αφού δημιουργήσουμε ένα **TextView** στοιχείο στο αρχείο **content_main.xml** δίνοντάς του αναγνωριστικό(**id**) ίσο με **"dateText"** και υπολογίσουμε την τρέχουσα ημερομηνία μέσα στην συνάρτηση που ονομάσαμε **getCurrentDate()** αλλάζουμε το κείμενο του **dateText** με την ημερομηνία.

```
Date currDate = new Date();
DateFormat df = android.text.format.DateFormat.getDateFormat(getApplicationContext());
dateText.setText(today + " " + df.format(currDate));
```

Αυτό που κάναμε είναι ότι φτιάξαμε ένα **Date object** , πήραμε την τιμή από το σύστημα και στην συνέχεια την μετατρέψαμε σε μορφή κειμένου(**String**) και τη τοποθετήσαμε στο στοιχείο **dateText.** Τέλος για να έχει ισχύ η συνάρτησή μας την καλούμε μέσα στην **onCreate()** .

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //Display current Date
        getCurrentDate();

    }
```

Όπως εξηγήσαμε σε προηγούμενο κεφάλαιο θα πρέπει να λάβουμε υπόψη μας ότι η κατάσταση δεδομένων μιας Activity διατηρείται καθώς αυτή βρίσκεται στην **στοίβα**(**stack**). Σε μια τέτοια περίπτωση τα θέλαμε η Ημερομηνία να αλλάζει δυναμικά κάθε φορά που συμβαίνει **Επαναφορά**(**Resume**) στην κορυφή και είναι ξανά ορατή ώστε να μην μας προβάλει λανθασμένα αποτελέσματα.

Για παράδειγμα αν για κάποιον λόγο ο χρήστης **επαναφέρει** την Δραστηριότητα μετά από διαφορετική μέρα θα θέλαμε να δούμε την τρέχουσα και όχι αυτήν που είχε αποθηκευμένη στον χώρο δεδομένων.

Αυτό το πετυχαίνουμε κάνοντας **Override** την μέθοδο **onResume()**

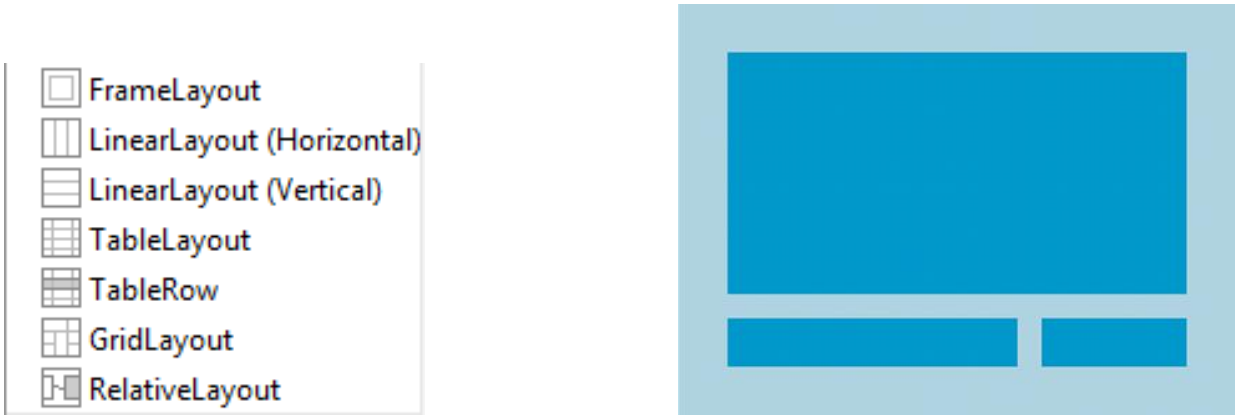
```
@Override
protected void onResume() {

	super.onResume();

	getCurrentDate();

}
```

Έτσι είμαστε σίγουροι ότι ποτέ δεν θα έχουμε θέματα ασυμφωνίας πληροφοριών.

3.2 Buttons και layout

Όπως προαναφέρθηκε στην αρχική οθόνη(home screen) μας θα υπάρχουν τέσσερα λειτουργικά κουμπιά(Buttons). Το Button είναι ένα widget το οποίο μπορεί κάποιος να το πατήσει μια φορά(click) ή να το κρατήσει πιεσμένο(hold press) και να προγραμματιστεί ανάλογα.

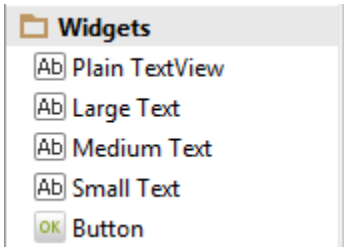
Το **layout** ορίζει την οπτική δομή του UI (περιβάλλοντος χρήστη). Επιλέγοντας το αρχείο **content_main.xml** , αρχικά πρέπει να επιλέξουμε σε τι **layout** θα τοποθετηθούν τα στοιχεία που πρόκειται να εισάγουμε. Από την παλέτα **layouts** (εικόνα 3.1) θα επιλέξουμε το **RelativeLayout**.



Εικόνα 3.1 RelativeLayout δομή

Το RelativeLayout αποτελεί μία ομάδα από στοιχεία προβολών(view group) το οποίο απεικονίζει διάφορα παιδιά-στοιχεία (child views) σε μια "σχετική θέση" στην οθόνη μας. Με άλλα λόγια, κάθε στοιχείο που εισάγεται σε αυτό το layout έχει μια σχετική θέση σε σχέση με τα υπόλοιπα συγγενικά του (δεξιά, πάνω από, κάτω από κτλ).

Αφού ορίσαμε το βασικό μας layout, από την παλέτα των **widgets**



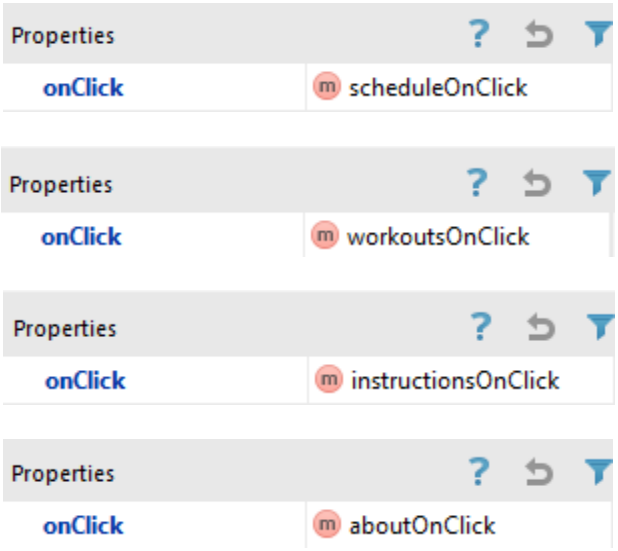
επιλέγουμε τέσσερα στοιχεία **Button** και τα προσαρμόζουμε στην οθόνη τοποθετώντας τα το ένα κάτω από το άλλο. Έπειτα δίνουμε κατάλληλα αναγνωριστικά (**Id**) σε κάθε ένα κουμπί ξεχωριστά, αφού αλλάξουμε το κείμενο(**Text**) ώστε να αντιπροσωπεύουν την λειτουργία τους, επιλέγουμε ως χρώμα **background** το background #4d90fe για το καθένα. Ήρθε λοιπόν η ώρα να τους δώσουμε λειτουργικότητα.

Στο αρχείο **MainActivity.java** θα δημιουργήσουμε τέσσερεις αντίστοιχες συναρτήσεις.

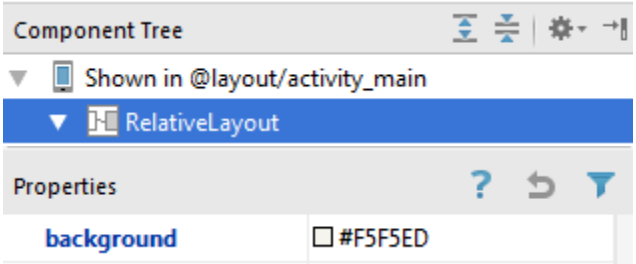
```
public void scheduleOnClick(View view){

	//Intent κώδικας

}

public void workoutsOnClick(View view){

	//Intent κώδικας

}

public void instructionsOnClick(View view){

	//Intent κώδικας

}

public void aboutOnClick(View view){

	//Intent κώδικας

}
```

Το τμήμα του κώδικα για το Intent εξετάζεται αναλυτικά σε άλλο κεφάλαιο.

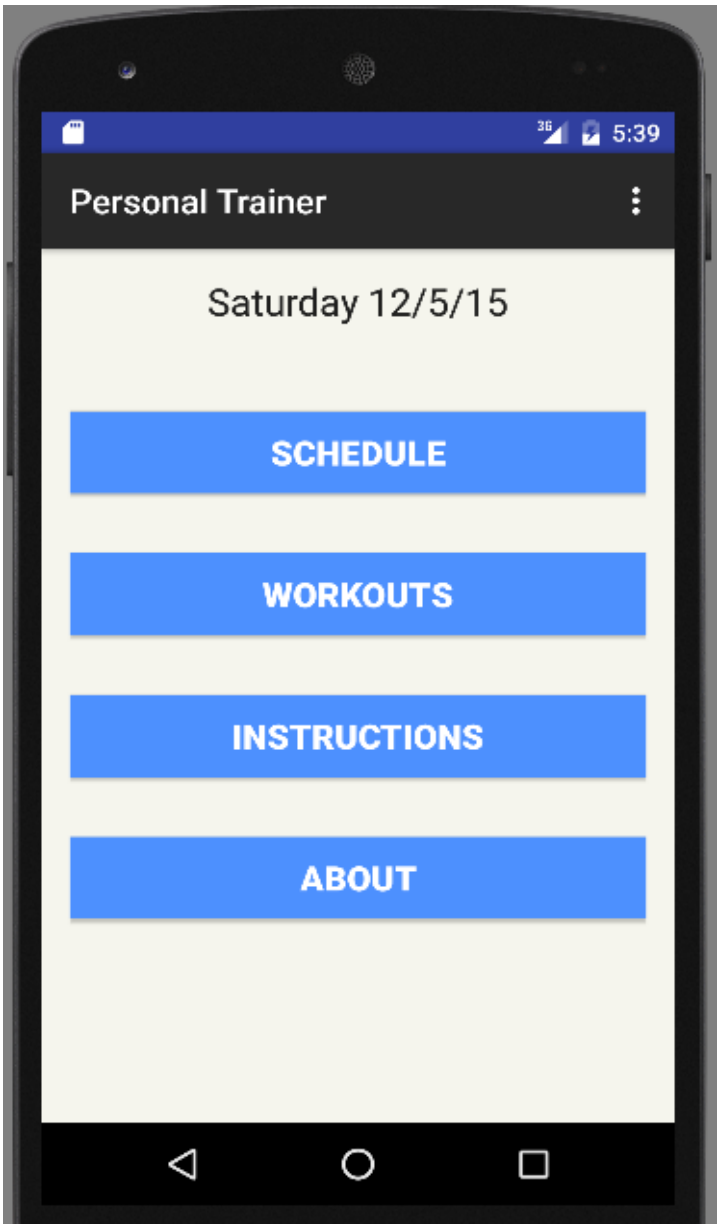
Επειδή όμως δεν υπάρχει άμεση σύνδεση των συναρτήσεων με την γραφική απεικόνιση των **buttons** πρέπει να δηλώσουμε στις ιδιότητες (properties), για καθένα ξεχωριστά, σε ποιό τμήμα του κώδικα να ανατρέξει όταν ο χρήστης πατήσει(**onClick**) το κάθε ένα κουμπί. Για αυτό τον λόγο προσθέτουμε στις ιδιότητες κάθε μία από τις παραπάνω συναρτήσεις που δημιουργήσαμε.



Ας δώσουμε και έναν διαφορετικό τόνο χρώματος στο background της εφαρμογής μας επιλέγοντας RelativeLayout και στα properties αλλάζοντας το χρώμα:



Τέλος τρέχοντας το project μας πατώντας το  θα έχουμε στην οθόνη του emulator(προσομοιωτή) :



Εικόνα 3.2 Στιγμιότυπο εφαρμογής σε Emulator Nexus 5 API 23 x86 Android 6.0

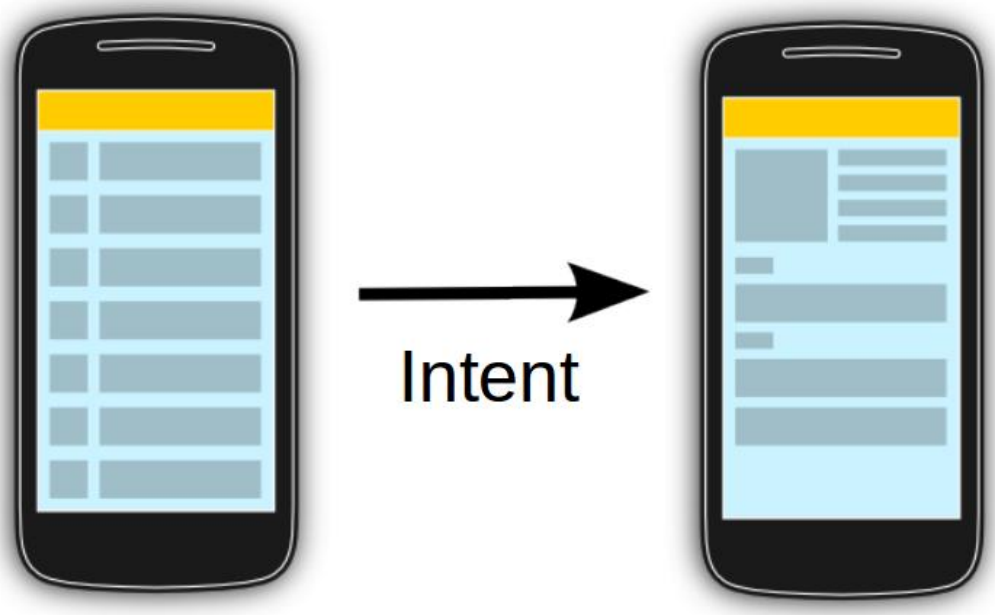
Σημείωση: Όλες οι δοκιμές γίνανε στον emulator(προσομοιωτή) Nexus 5 (API 23) Android 6.0 και σε φυσική συσκευή smartphone Motorola Moto G Android 5.1 Lollipop.

4 Intents

Τα **Intents** είναι ασύγχρονα μηνύματα τα οποία επιτρέπουν σε μέρη της εφαρμογής, όπως **Δραστηριότητες**(**Activities**), **Υπηρεσίες** (**Services**) κ.α., να επικοινωνούν και να ζητούν λειτουργικότητα από άλλα τέτοια μέρη, όχι μόνο της ίδιας της εφαρμογής αλλά και από άλλες.

Τα **Intents** είναι αντικείμενα τύπου ***android.content.Intent*** και μπορούμε να τα στείλουμε από τον κώδικά μας μέσω συστήματος στο μέρος "στόχο". Η διαδικασία ξεκινάει με την μέθοδο ***startActivity()***.Το παρακάτω κομμάτι κώδικα μας δείχνει πώς μέσω ενός intent ξεκινάει μια άλλη Activity.

```
Intent intent = new Intent(this,Activity2. class);

startActivity(intent);
```

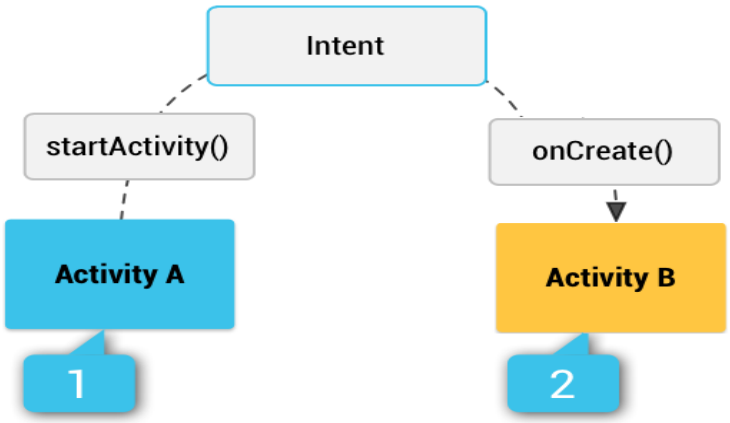


4.1 Τύποι των Intents

Υπάρχουν δύο διαφορετικοί τύποι :

Με ρητή δήλωση(Explicit)

Τα explicit intents ορίζουν επακριβώς τα Activities και Services που πρόκειται να καλέσουν μέσω του Android συστήματος, χρησιμοποιώντας την **Java κλάση** (.class) ως αναγνωριστικό.



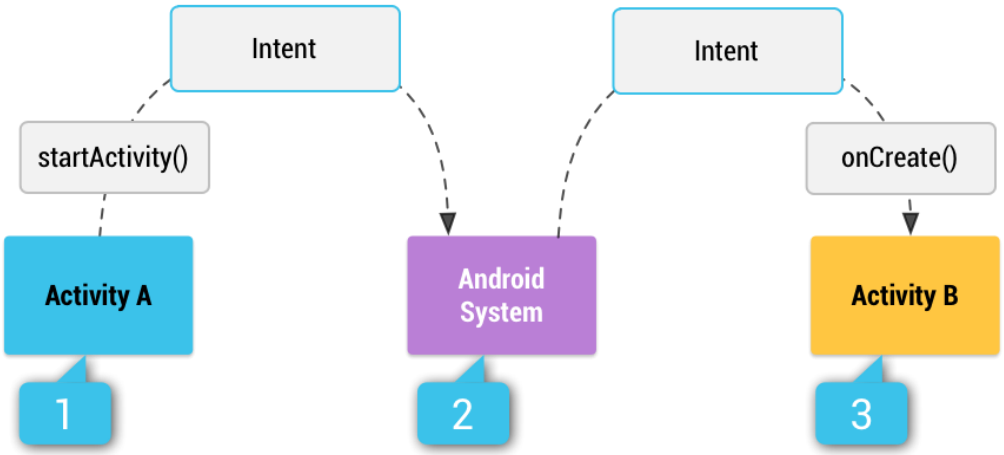
Το παρακάτω κομμάτι κώδικα δείχνει πως δημιουργούμε ένα τέτοιο **explicit intent** .

```
Intent intent = new Intent(this,Activity2. class);

intent.putExtra("Value1","first value for Activity2");

intent.putExtra("Value2","second value for Activity2");
```

Όπως καταλαβαίνουμε αυτού του είδους χρησιμοποιούνται τυπικά από τον προγραμματιστή που έχει πλήρη έλεγχο στις κλάσεις αυτές.

Χωρίς ρητή δήλωση(Implicit)

Τα implicit intents αντιθέτως χρησιμοποιούνται στην περίπτωση όπου ξεκινά ένα Activity χωρίς εμείς να γνωρίζουμε ακριβώς την κλάση(class) αλλά απευθυνόμαστε στο σύστημα.



Για παράδειγμα στο παρακάτω κομμάτι κώδικα λέμε στο σύστημα να μεταβεί στην διεύθυνση "http://developer.android.com" το οποίο θα πρέπει να αποφασίσει με ποιόν **περιηγητή**(**web-browser**), που είναι εγκατεστημένος στην συσκευή, θα την ανοίξει.

```
Intent intent = new Intent(Intent.ACTION_VIEW,
      Uri.parse("http://developer.android.com"));

startActivity(intent);
```

4.2 Δεδομένα μέσω Intents

Ένα intent εκτός από τα κύρια δεδομένα όπως τον τύπο, τη λειτουργία κ.α. μπορεί αν επιθυμούμε να περιέχει και επιπλέον δεδομένα, τα οποία μπορούμε να περάσουμε από μία Δραστηριότητα σε μια άλλη.

Έτσι βασισμένοι στην κλάση **Bundle**, μπορούμε να **προσθέτουμε δεδομένα(Extras**) μέσω της μεθόδου ***putExtra()*** ενός αντικειμένου (object) τύπου **Intent**. Αυτά τα **Extras** είναι ένα ζεύγος **Όνομα-Τιμή (key-value)** όπου το **Όνομα** είναι πάντοτε τύπου String. Η **Τιμή** μπορεί να είναι διάφοροι τύποι όπως int, float, string, bundle κτλ.

Για να λάβουμε τα δεδομένα που στείλαμε από την μία Activity σε μια άλλη μέσω intent χρησιμοποιούμε την μέθοδο ***getExtras()*** στο αντικείμενο intent που λάβαμε, δηλαδή ***getIntent().getExtras()***.

Το παρακάτω κομμάτι κώδικα μας δείχνει την βασική υλοποίηση όσων εξηγήσαμε παραπάνω.

```
// Activity1

Intent intent = new Intent(this, Activity2.class);

intent.putExtra("message", "Monday");

startActivity(intent);
```

```
// Activity2

Bundle extras = getIntent().getExtras();

if( extras == null) {

       return ;

}

String message = extras.getString("message");     // "Monday"
```

4.3 Τερματισμός Δραστηριότητας

Όπως εξηγήσαμε σε προηγούμενο κεφάλαιο (Κεφάλαιο 2) όταν από το μία Activity μεταβούμε σε μία άλλη η δεύτερη ανεβαίνει στην κορυφή της στοίβας ωστόσο η πρώτη δεν παύει να εξακολουθεί να υπάρχει μέσα στην στοίβα. Οπότε όταν δημιουργούμε ένα Intent πρέπει να σκευτούμε τί θέλουμε να κάνουμε με την προηγούμενη Activity.

Αν επιθυμούμε να την ξαναχρησιμοποιήσουμε στο μέλλον, μας αρκεί το ***startActivity()***.Η κατάστασή της θα εισέλθει στην στοίβα.

Αν επιθυμούμε η Activity στην οποία δημιουργούμε το intent και καλούμε την ***startActivity()*** να τερματιστεί, πρέπει να καλέσουμε την μέθοδο ***finish()***.

Σε περίπτωση που καλέσουμε τη ***finish()*** ουσιαστικά εκτελείται (Κεφ. 2) η μέθοδος **onDestroy()**. Η ***finish()*** δεν τερματίζει αμέσως την

Δραστηριότητα, απλώς ενημερώνει το σύστημα ότι η συγκεκριμένη είναι έτοιμη για απελευθέρωση από την μνήμη.

Παρακάτω βλέπουμε ένα τέτοιο στιγμιότυπο με χρήση της μεθόδου **finish()** :

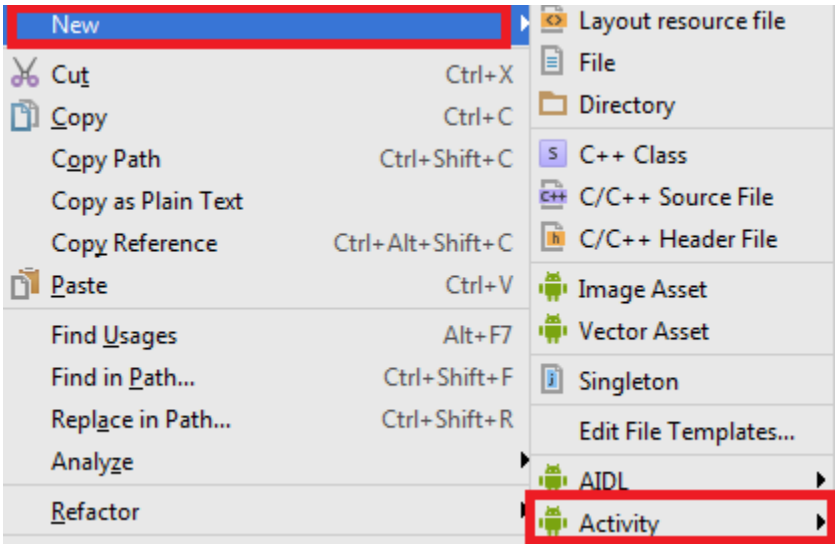
```
// Activity1

Intent intent = new Intent(this, Activity2.class);

intent.putExtra("message", "Monday");

startActivity(intent);

finish();
```

Η διαφορά εδώ είναι ότι από την στιγμή που γίνει η μετάβαση στην **Activity2** ο χρήστης δεν θα είναι πλέον σε θέση επιστρέψει στην **Activity1** , για παράδειγμα πιέζοντας το πλήκτρο "Επιστροφή".

5 Η κλάση(class) Schedule

Αφού ολοκληρώσαμε την κλάση **MainActivity.java** και περιγράψαμε τον τρόπο λειτουργίας των **Intents** θα προχωρήσουμε στην δημιουργία μιας άλλης σημαντικής κλάσης για την εφαρμογή που θα ονομάσουμε **Schedule.java.**

Επιλέγουμε από το μενού **New -> Activity -> Blank Activity**



Στην συνέχεια θα έχουμε τρία νέα αρχεία στα οποία και θα δουλέψουμε, τα **Schedule.java**, **activity_schedule.xml** και **content_schedule.xml .**

Ο σχεδιασμός και η υλοποίηση αυτής της κλάσης δεν διαφέρει σχεδόν καθόλου σε σχέση με την **MainActivity.**

Θα περιέχει και αυτή το κομμάτι κώδικα για την **onCreate()** , **onResume(), onBackPressed().**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //Display current Date
    getCurrentDate();


}

@Override
protected void onResume(){
    super.onResume();
    getCurrentDate();
}

@Override
public void onBackPressed(){

    Intent intent = new Intent(this, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    startActivity(intent);
    finish();
}
```

Επίσης επιλέγουμε να φαίνεται και σε αυτήν την οθόνη η ημερομηνία οπότε θα υλοποιήσουμε και την μέθοδο **getCurrentDate()** η οποία περιγράφτηκε στην **MainActivity.** Τα χρώματα στα buttons και στο background παραμένουν τα ίδια.

5.1 Κουμπιά (Buttons)

Σε αυτό το μενού θα έχουμε τρία διαφορετικά λειτουργικά κουμπιά.

- Schedule List
- New Schedule
- Delete Schedule

Πατώντας το κουμπί **Schedule List** ο χρήστης θα βλέπει μια πλήρη λίστα με όλα τα προγράμματα γυμναστικής που ο ίδιος έχει δημιουργήσει.

Πατώντας το κουμπί **New Schedule** ο χρήστης θα δημιουργεί ένα νέο πρόγραμμα γυμναστικής.

Και τέλος πατώντας το κουμπί **Delete Schedule** ο χρήστης θα διαγράφει ένα πρόγραμμα από την λίστα.

5.2 Flags και AlertDialog

Όπως εξηγήσαμε στο Κεφάλαιο 4 με τα Intents πολλές φορές θα χρειαστεί να στείλουμε δεδομένα μεταξύ διαφορετικών Activities. Σε αυτήν την περίπτωση θα χρειαστεί να περάσουμε ως παράμετρο στην μεταβλητή **flag** μέσω **Intent** στην ίδια Δραστηριότητα που θα δημιουργήσουμε αργότερα και θα ονομάσουμε **myScheduleList.class.**

Πιο συγκεκριμένα όταν ο χρήστης πατήσει το κουμπί **Schedule List** τότε θα περαστεί ως πληροφορία η μεταβλητή **flag** με τιμή **"show"** που υποδηλώνει "**παρουσίασε την λίστα**" και ο κώδικας που θα εκτελεστεί στην κλάση **mySchedule**List θα σχετίζεται μόνο με την εμφάνιση λίστας.

Αντίθετα, εάν ο χρήστης πατήσει **Delete Schedule** τότε θα περαστεί η μεταβλητή **flag** με τιμή "**delete**" που υποδηλώνει "**διέγραψε πρόγραμμα**" και ο κώδικας που θα εκτελεστεί θα σχετίζεται μόνο με την διαγραφή.

Την παραπάνω διαδικασία την ακολουθούμε γιατί μεταβαίνουμε στην ίδια Activity από διαφορετικά κουμπιά αλλά θέλουμε να εκτελέσουμε διαφορετικό κώδικα μέσα σε αυτή την Activity.

Όταν όμως πατήσει **New Schedule** τότε ένα αναδυόμενο παράθυρο θα εμφανιστεί ζητώντας από τον χρήστη να πληκτρολογήσει ένα όνομα για το καινούργιο πρόγραμμα γυμναστικής. Το όνομα αυτό πρέπει να πληροί κάποιες προϋποθέσεις:

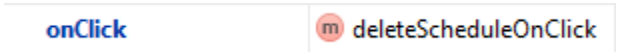
- Να περιέχει τουλάχιστον ένα γράμμα ή σύμβολο ή αριθμό, να μην είναι δηλαδή κενό (white-space characters only).
- Και να μην υπάρχει ήδη πρόγραμμα με το ίδιο όνομα στην **Βάση Δεδομένων**(**Database**).

Εφόσον ο χρήστης επιλέξει έγκυρο όνομα τότε στο παρασκήνιο θα πραγματοποιηθούν κάποιες αρχικοποιήσεις όπως:

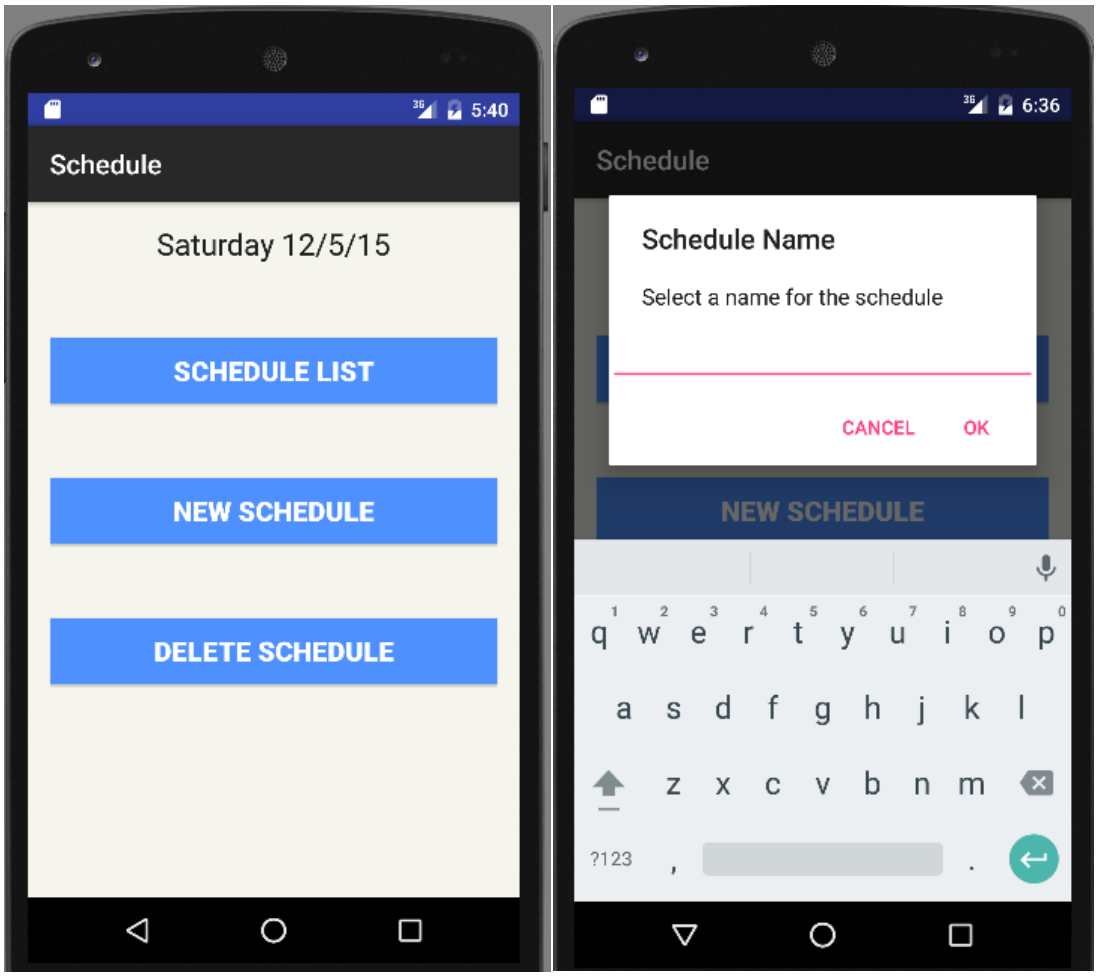
- Προετοιμασία για εγγραφή/ανάγνωση με την **Βάση Δεδομένων**.
- Καθαρισμός της **Βάσης Δεδομένων** από προσωρινά δεδομένα που δεν είναι έγκυρα.
- Προετοιμασία για εγγραφή/ανάγνωση από τα **SharedPreferences**.
- Αρχικοποίηση μεταβλητών των **SharedPreferences**.

Η Βάση Δεδομένων και τα SharedPreference αποτελούν τρόπους μόνιμης αποθήκευσης δεδομένων στο σύστημα και θα περιγραφούν αναλυτικά σε επόμενο κεφάλαιο.

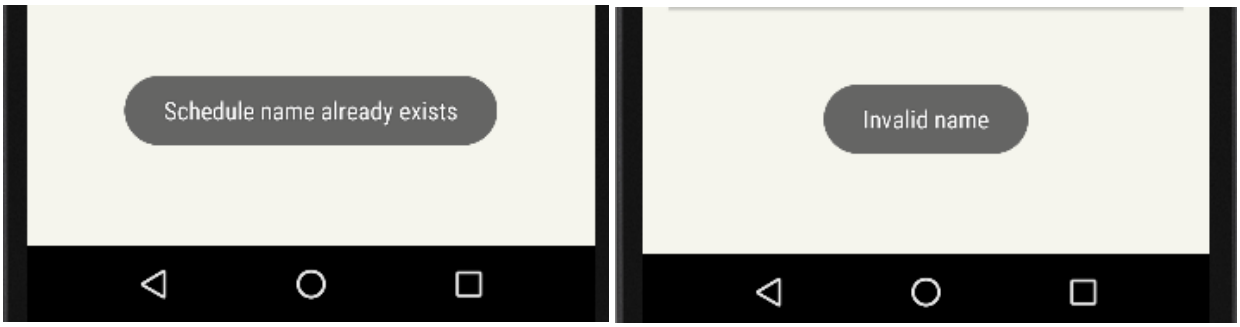
Τέλος, πρέπει να συνδέσουμε τον κώδικα με το γραφικό περιβάλλον από τα αρχεία **Schedule.java** και **content_schedule.xml .**



```
public void scheduleListOnClick(View view){
       Intent intent = new Intent(this,myScheduleList.class);
       intent.putExtra("Flag", "show");
       startActivity(intent);
}

public void newScheduleOnClick(View view){

       AlertDialog.Builder alert = new AlertDialog.Builder(this);

       alert.setTitle("Schedule Name");

       final EditText inputName = new EditText(this);

       alert.setView(inputName);

       alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int whichButton) {

                     if(//inputName is valid){

                            //Αρχικοποίηση Βάσης Δεδομένων - Database Init

                            //Κώδικας αρχικοποίησης των SharedPreferences

                            // Intent σε New_Schedule.class

                     }

                     else{

                            Toast.makeText(Schedule.this,"Invalid
name",Toast.LENGTH_LONG).show();

                     }

              }

       });

       alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int whichButton) {
                         //Canceled
                     }
              }
        );

       alert.show();

}


public void deleteScheduleOnClick(View view){

       Intent intent = new Intent(this,myScheduleList.class);
       intent.putExtra("Flag", "delete");
       startActivity(intent);
}
```



Εικόνα 5.1 Στιγμιότυπα κλάσης Schedule στον emulator



Εικόνα 5.2 Το όνομα υπάρχει ήδη στην Βάση Δεδομένων / Μη έγκυρο όνομα

6 Μόνιμη Αποθήκευση Δεδομένων

Το Android μας παρέχει διάφορους τρόπους να αποθηκεύουμε δεδομένα και πληροφορίες για μια εφαρμογή. Η επιλογή εξαρτάται από εμάς για το τι ακριβώς θέλουμε πραγματικά να κάνουμε με τα δεδομένα μας. Αν για παράδειγμα έχουμε μεγάλο όγκο δεδομένων με μαζική προσπέλαση θα χρησιμοποιήσουμε **Βάση Δεδομένων,** ενώ αν θέλουμε λίγα δεδομένα και να είναι προσπελάσιμα και από άλλες εφαρμογές θα χρησιμοποιήσουμε **SharedPreferences**.

Στην συγκεκριμένη εργασία θα χρησιμοποιήσουμε δύο από τους τρόπους που μας παρέχονται:

- **SQLite Βάση Δεδομένων(Database)**
  Βάση δεδομένων αποκλειστικά για την εφαρμογή μας.
- **Shared Preferences**
  Αποθήκευση σε μορφή ζεύγους "Όνομα" - "Τιμή" σε αρχείο.

6.1 Shared Preferences

Η κλάση **SharedPreferences** μας παρέχει δυνατότητα αποθήκευσης και ανάκτησης δεδομένων σε αρχείο της μορφής όνομα-τιμή οποιουδήποτε από τους τύπους **Integer**, **String**, **Boolean**, **Float** και **Long**.

Χρησιμοποιούμε δύο μεθόδους για να λάβουμε δεδομένα μέσω **SharedPreferences**:

- `getSharedPreferences()`
- `getPreferences()`

Η πρώτη χρησιμοποιείται εάν θέλουμε να έχουμε πολλαπλά αρχεία με preferences. Το όνομα του αρχείου μπαίνει ως πρώτη παράμετρος.

Η δεύτερη χρησιμοποιείται αν έχουμε μόνο ένα αρχείο, οπότε παραλείπεται το όνομα.

Για να γράψουμε στο αρχείο μια μεταβλητή ("όνομα-τιμή"):

- Καλούμε την μέθοδο **edit()** για να λάβουμε τον συντάκτη **SharedPreferences.Editor**
- Προσθέτουμε τις επιθυμητές μεταβλητές με τις μεθόδους **putString()**, **putBoolean()**, **putInt()** κτλ.
- Πραγματοποίησε εγγραφή στο αρχείο με την μέθοδο **commit()** ή **apply()**

Για διαβάσουμε από το αρχείο τις μεταβλητές χρησιμοποιούμε τις μεθόδους **getString()**, **getBoolean()** κτλ.

Παρακάτω παρουσιάζεται ένα παράδειγμα εγγραφής σε ένα αρχείο με όνομα **"MyFile".xml**

```
SharedPreferences sharedPreferences = getSharedPreferences("MyFile",
Context.MODE_PRIVATE);  // Μόνο η εφαρμογή μπορεί να βρει το αρχείο(Mode_Private).


SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putString("Day", "Monday");

editor.putString("Month", "December");

editor.commit();
```

Μετά το commit() στο σύστημα θα δημιουργηθεί αν δεν υπάρχει ήδη το αρχείο **"MyFile".xml** και θα περιέχει δυο μεταβλητές **Day** και **Month** με τιμές **Monday** και **December** αντίστοιχα.

Παρακάτω παρουσιάζεται ένα παράδειγμα ανάγνωσης από ένα αρχείο με όνομα **"MyFile".xml**

```
SharedPreferences sharedPreferences = getSharedPreferences("MyFile",
Context.MODE_PRIVATE);

String day  = sharedPreferences.getString("Day", "");

String month  = sharedPreferences.getString("Month", "");
```

Η πρώτη παράμετρος στην getString() είναι το όνομα της μεταβλητής που θέλουμε να διαβάσουμε **Day** και **Month** ενώ το **""** μας επιστρέφεται σε περίπτωση που η μεταβλητή δεν έχει βρεθεί για κάποιον λόγο.

Στην συγκεκριμένη εργασία εργαζόμαστε σε δύο αρχεία με ονόματα:

- TempData.xml
- DurationLog.xml

Στο πρώτο κρατάμε διάφορες "προσωρινές" μεταβλητές κατά την διάρκεια δημιουργίας ενός νέου προγράμματος γυμναστικής από τον χρήστη, προτού αποθηκευτούν μόνιμα στην SQLite βάση αργότερα.

Στο δεύτερο κρατάμε ένα καταστατικό με την διάρκεια ζωής των προγραμμάτων γυμναστικής που έχει δημιουργήσει ο χρήστης, για παράδειγμα πότε ήταν η τελευταία φορά που το χρησιμοποίησε κτλ.

6.2 SQLite Βάση Δεδομένων (Database)

Ένας άλλος τρόπος για μόνιμη αποθήκευση δεδομένων για μια εφαρμογή είναι η χρήση SQLite βάσης δεδομένων. Η χρήση αυτής της μεθόδου είναι η ιδανική όταν δουλεύουμε με πολλά δεδομένα τα οποία έχουν την ίδια δομή και επαναλαμβάνονται. Εδώ πρέπει να τονίσουμε ότι προϋποθέτει βασικές γνώσεις SQL για να την χρησιμοποιήσει κανείς.

Οτιδήποτε έχει να κάνει με επικοινωνία με την βάση δεδομένων υλοποιείται σε μια κλάση που στην παρούσα εργασία θα ονομάσουμε **MyDBHandler** σε ένα νέο αρχείο **MyDBHandler.java** και καλούμε τις μεθόδους δημιουργώντας και καλώντας αντικείμενα **MyDBHandler.**

Στην συνέχεια θα πρέπει να ορίσουμε το "θέμα" για την SQL βάση δεδομένων, δηλαδή πρόκειται για μια σαφή δήλωση για το πώς είναι οργανωμένη η βάση μας. Οι δηλώσεις που κάνουμε είναι:

- όνομα της βάσης δεδομένων
- ο αριθμός έκδοσης της βάσης (Database Version)
- ονόματα πινάκων (Tables)
- ονόματα στηλών (Columns)

```
public class MyDBHandler extends SQLiteOpenHelper{

  private static final int DATABASE_VERSION = 1;
  private static final String DATABASE_NAME = "exercisesdb.db";
  public static final String TABLE_EXERCISES = "exercises";
  public static final String COLUMN_ID = "_id";
  public static final String COLUMN_DAY = "day";
  public static final String COLUMN_EXERCISENAME = "exercisename";
```

Η παραπάνω κλάση λοιπόν είναι υπεύθυνη για οτιδήποτε θέλουμε να κάνουμε συγκεκριμένα με την βάση δεδομένων μας σε ένα αρχείο που θα αποθηκεύεται στην συσκευή του χρήστη. Οποτεδήποτε δημιουργείται ή ανανεώνεται μια βάση, προστίθενται ή διαγράφονται αντικείμενα στην βάση, ακόμη και μια απλή προσπέλαση/ανάγνωση κάποιου αντικειμένου που ανήκει στην βάση υλοποιείται μέσα σε αυτή την κλάση.

DATABASE_VERSION: Κάθε φορά που θέλουμε να ανανεώσουμε την βάση δεδομένων μας πρέπει να αλλάξουμε και τον αριθμό έκδοσής της.

DATABSE_NAME: Ουσιαστικά πρόκειται για το όνομα του αρχείου που θα αποθηκευτούν τα δεδομένα της βάσης.

TABLE_ και **COLUMN_**: Ονόματα πινάκων και στηλών αντίστοιχα.

Στην συνέχεια προσθέτουμε τον constructor:

```
public MyDBHandler(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
    super(context, DATABASE_NAME, factory, DATABASE_VERSION);
  }
```

και κάνουμε override τις δυο πιο βασικές μεθόδους **onCreate()** και **onUpgrade()**.

```
@Override
  public void onCreate(SQLiteDatabase db) {

          String query ="CREATE TABLE " + TABLE_EXERCISES + "(" +
          COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
          COLUMN_DAY + " TEXT, " +
          COLUMN_EXERCISENAME + " TEXT, " + ");";

          db.execSQL(query);

  }


@Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_EXERCISES);
    onCreate(db);
  }
```

Η **onCreate()** εκτελείται μια φορά στην αρχικοποίηση και δημιουργία της βάσης δεδομένων. Όπως και φαίνονται οι πίνακες με τις στήλες και τους τύπους των δεδομένων που θα αποθηκεύονται.

Η **onUpgrade()** εκτελείται κάθε φορά που ανανεώνεται η έκδοση της βάσης (**DATABASE_VERSION**). Διαγράφει και ξαναδημιουργεί από την αρχή.

Τα αντικείμενα που πρόκειται να καταχωρούμε στην βάση μας τα ορίζουμε με μια δομή που ονομάζουμε **Exercises.** Υλοποιούμε την δομή αυτή σε μια κλάση **Exercises.class** με τα πεδία όπως **_id**, **name**, **sets**, **reps** κ.α. και με τους αντίστοιχους **Getters** και **Setters** τους.

Στην συνέχεια στην αρχική κλάση **MyDBHandler** υλοποιούνται ένας μεγάλος αριθμός από μεθόδους που έχουν να κάνουν με την εισαγωγή και διαγραφή αντικειμένων **Exercises** προς και από την βάση δεδομένων, όπως επίσης και εκτύπωση όλης της βάσης και πολλές άλλες συναρτήσεις τις οποίες δεν θα παρουσιάσουμε με κώδικα.

Αναφορικά μερικές από αυτές είναι:

```
//Add a new row to the Database
 public void addExercise(Exercises exercise){

            //κώδικας

 }

//Print all the exercises in the Database

public String printExercises( ){

            //κώδικας

}

//Return Sets&Reps of a specific exercise by name

public String returnSetsReps(String exercise){

            //κώδικας

}
```

Για να καλέσουμε τις παραπάνω μεθόδους με τα κατάλληλα ορίσματα:

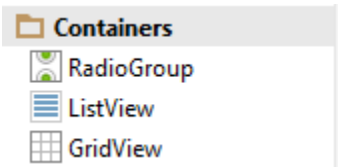
```
MyDBHandler myHandler = new MyDBHandler(this, null, null , 1);

myHandler.printExercises();
```

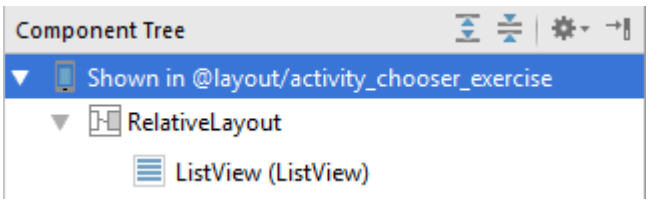
7 Λίστες και CustomAdapters

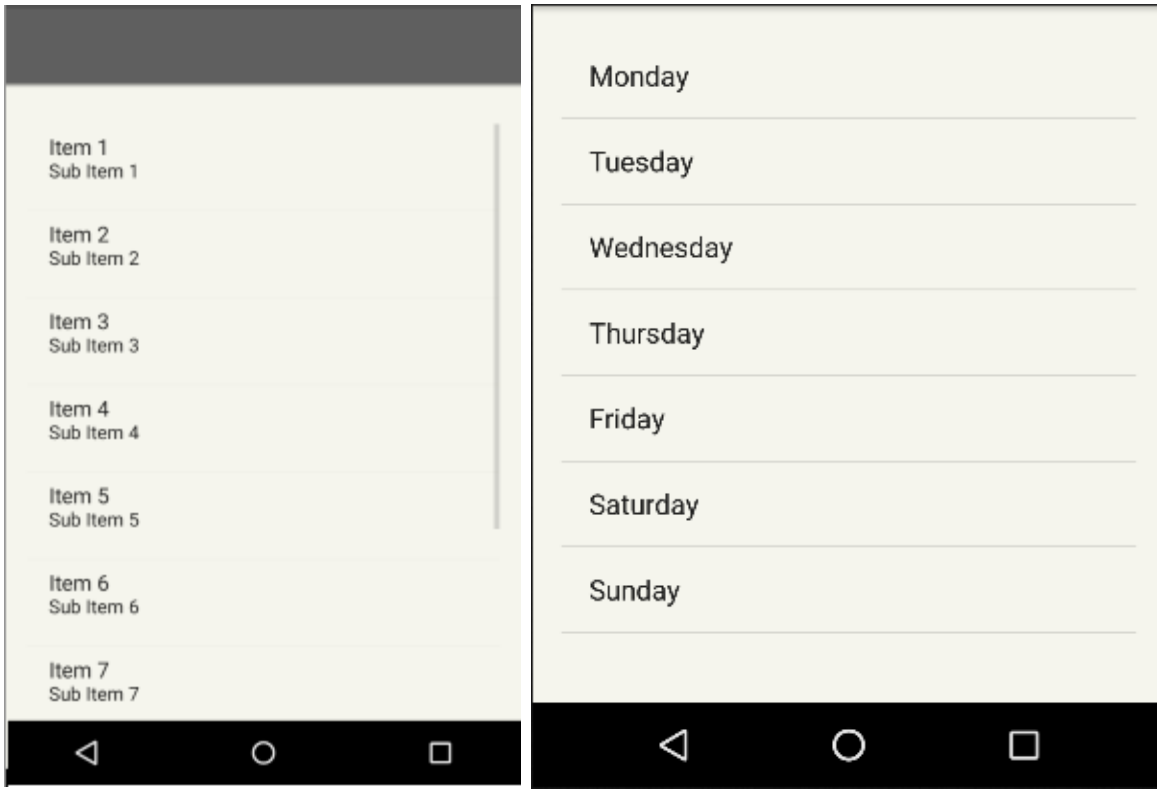
Σε προηγούμενο κεφάλαιο(Κεφ. 5) παρουσιάσαμε και συζητήσαμε το μενού από το οποίο ο χρήστης θα έχει την δυνατότητα να δημιουργεί και να διαγράφει προγράμματα γυμναστικής καθώς και να προβάλλει μια λίστα με όλα όσα έχει μέχρι τώρα δημιουργήσει πιέζοντας το κατάλληλο κουμπί. Στην συνέχεια θα περιγράψουμε τις νέες κλάσεις που μεταβαίνουμε μέσω Intents (πιέζοντας κάποιο κουμπί) οι οποίες έχουν όλες μια κοινή υλοποίηση, χρησιμοποιούνε **λίστες(listviews)**.

Λίστα είναι ένα σύνολο από στοιχεία/αντικείμενα ή συνδυασμός στοιχείων διατεταγμένα το ένα κάτω από το άλλο τα οποία ο χρήστης μπορεί να προβάλει κάνοντας scrolling στην οθόνη της συσκευής του. Τα αντικείμενα αυτά μπορεί να είναι οτιδήποτε περιλαμβάνει το μενού με τα widgets όπως **TextViews/ImageViews/Buttons/Checkboxes** κτλ. Μια λίστα μπορεί να είναι από πολύ απλή έως πολύ σύνθετη ως προς το τι αντικείμενα θα περιέχει σε κάθε κελί της.

Από το μενού των widgets επιλέγουμε το στοιχείο **ListView** και το τοποθετούμε σε ένα **RelativeLayout** ενός αρχείου **content_.xml**. Όλη η διαδικασία επεξεργασίας της λίστας γίνεται στο αντίστοιχο .Java αρχείο.







Εικόνα 7.1 Λίστα αντικειμένων

Τα αντικείμενα που πρόκειται να βάλουμε στην λίστα τα δημιουργούμε με κώδικα java στο αντίστοιχο αρχείο. Σε αυτή την εργασία μία από τις λίστες θα αποτελείται από ονόματα των ασκήσεων γυμναστικής οπότε τα ονόματα(Strings) δηλώνονται μέσω κώδικα. Την διασύνδεση κώδικα και **ListView** την κάνει το αρχείο που ονομάζουμε **Adapter**. Σε αυτό το αρχείο περιλαμβάνεται η δομή από κάθε ένα από τα αντικείμενα στην λίστα μας(βλ. εικόνα 7.1). Στην περίπτωση που δημιουργούμε δικούς μας **Adapters** ονομάζονται **CustomAdapters**.

Παρακάτω φαίνεται ο κώδικας σε java διασύνδεσης λίστας ονομάτων με το αντίστοιχο **ListView** στοιχείο:

```
String[] daysList = {"MONDAY", "TUESDAY", "WEDNSDAY", "THURSDAY", "FRIDAY", "SATURDAY", "SUNDAY"};

ListAdapter myAdapter = new ArrayAdapter<String>( this, android.R.layout.simple_list_item1, daysList );

ListView myListView = (ListView) findViewById(R.id. myListView );

myListView.setAdapter(myAdapter);
```

Έχουμε έναν **Adapter** με όνομα **myAdapter** ο οποίος και είναι υπεύθυνος για την μετατροπή της λίστας με Strings με τις ημέρες σε λίστα αντικειμένων(**ListAdapter**). Το " android.R.layout.*simple_list_item1*" υποδηλώνει την μορφή που θα έχει κάθε αντικείμενο στην λίστα και αποτελεί την πιο απλή μορφή αντικειμένου σε μια απλή λίστα στοιχείων κειμένου.

Στην παρούσα εργασία θα κάνουμε χρήση πιο σύνθετων μορφών αντικειμένων σε λίστα οπότε αντί του **ListAdapter** θα εισάγουμε τον **CustomAdapter** που θα παρουσιαστούν στις επόμενες υποενότητες.

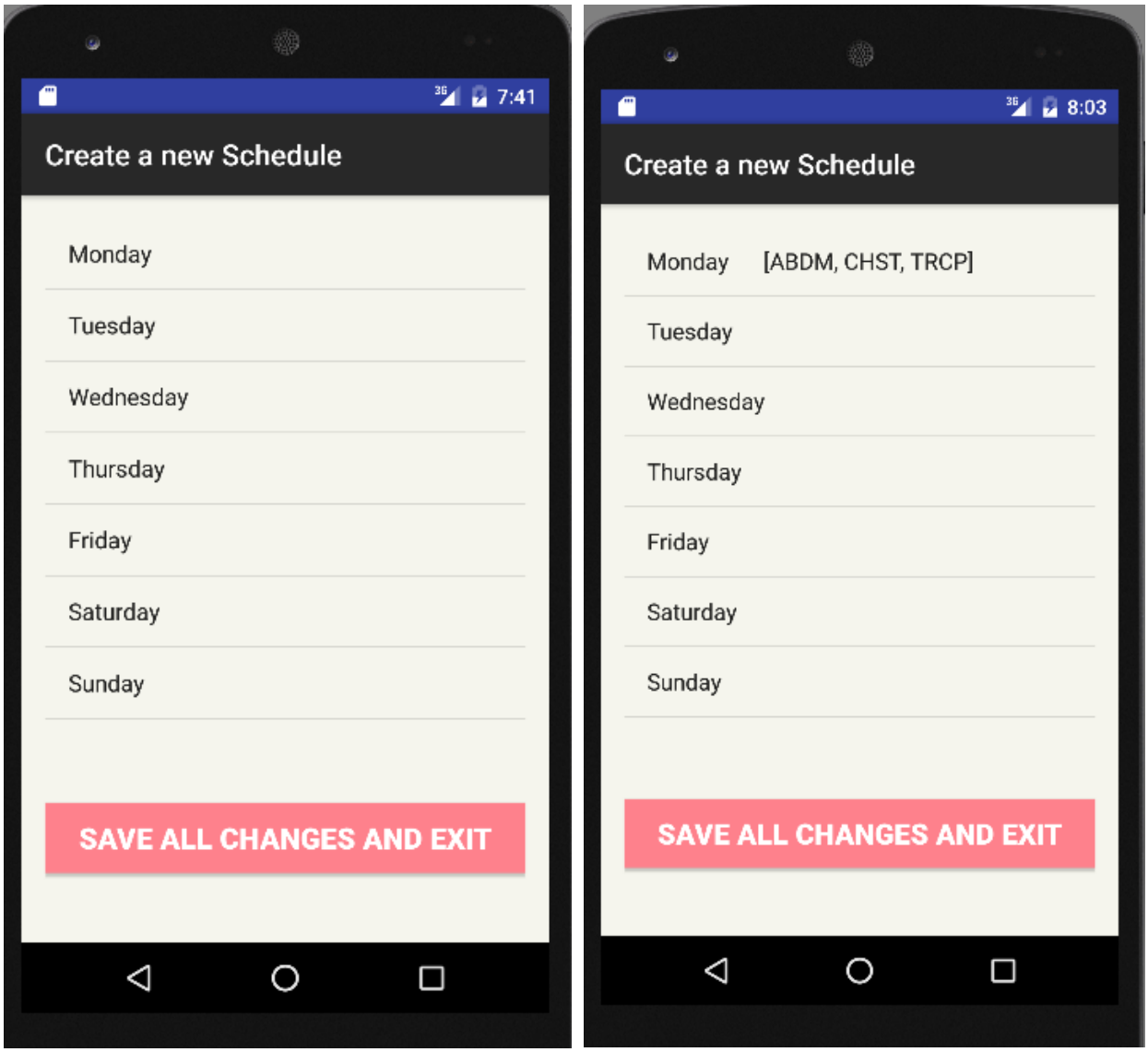
```
ArrayList<MyNewType> daysList = new ArrayList<MyNewType>();

daysList.add(new  MyNewType("MONDAY"));

.....

.....

daysList.add(new  MyNewType("SUNDAY"));


MyCustomAdapter myAdapter = new MyCustomAdapter(daysList , this );

ListView myListView = (ListView) findViewById(R.id. myListView );

myListView.setAdapter(myAdapter);
```

7.1 Δραστηριότητα(Activity) επιλογής ασκήσεων για κάθε ημέρα της εβδομάδας

Στην ενότητα 5.2 ο χρήστης αφού επιλέξει την δημιουργία ενός νέου προγράμματος γυμναστικής και πληκτρολογήσει κατάλληλο όνομα που θα πληροί κάποιες προϋποθέσεις. Πατώντας το πλήκτρο "OK" θα μεταβεί σε μία εκ νέου οθόνη μέσω Intent στην οποία καλείται να επιλέξει και να ορίσει μυϊκές ομάδες και ασκήσεις για κάθε μια μέρα της εβδομάδας ξεχωριστά όπως επιθυμεί ο ίδιος.

Αυτή η νέα Δραστηριότητα(Activity) περιλαμβάνει μια λίστα με τις ημέρες της εβδομάδας(βλ. εικόνα 7.2) καθώς και ένα πεδίο που θα εμφανίζει τις μυϊκές ομάδες που επέλεξε ο χρήστης έτσι ώστε να αποφευχθεί η σύγχυση στην περίπτωση που δεν θυμάται τί έχει επιλέξει και για ποιά ημέρα επ' ακριβώς. Τέλος η Δραστηριότητα αυτή θα περιλαμβάνει ένα πλήκτρο(κουμπί) για οριστική "Αποθήκευση όλων των αλλαγών και Έξοδος"("Save All Changes and Exit") .

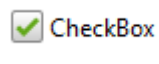
Σε αυτό το σημείο πρέπει να σημειώσουμε ότι οποιεσδήποτε επιλογές και αλλαγές κάνει ο χρήστης αποθηκεύονται στο σύστημα προσωρινά. Η μόνιμη εκχώρηση των δεδομένων στην Βάση θα γίνεται μόνο στην περίπτωση που πατήσει το κουμπί "Αποθήκευση". Με το που πατήσει το κουμπί αυτό τότε δημιουργείται το πρόγραμμα γυμναστικής, που μπορεί να ελέγξει οποιαδήποτε στιγμή από το μενού επιλογών, καθώς και αρχικοποιείται η διάρκεια(duration) σε 0 ημέρες.



Εικόνα 7.2 Μενού από το οποίο ο χρήστης ορίζει ασκήσεις για κάθε ημέρα

Όπως παρατηρούμε στην παραπάνω λίστα(ListView) ορίζουμε έναν απλό Adapter (Default Adapter) ο οποίος αποτελείται μόνο από ένα αντικείμενο κειμένου με την μέρα και ομάδα εκγύμνασης, καλώντας το "android.R.layout.*simple_list_item1*" ως όρισμα. Στις επόμενες Δραστηριότητες θα κάνουμε χρήση λιστών με επιπλέων αντικείμενα και δικούς μας CustomAdapters.

7.2 Δραστηριότητα(Activity) επιλογής μυϊκής ομάδας με Checkboxes σε λίστα

Συνεχίζουμε περιγράφοντας την οθόνη(intent σε νέο Activity) την οποία βλέπει πλέον ο χρήστης αφού επιλέξει μια ημέρα από το μενού που περιγράψαμε παραπάνω. Η νέα Δραστηριότητα περιλαμβάνει μια εκ νέου λίστα η οποία αποτελείται από ένα αντικείμενο κειμένου(**TextView**) καθώς και ένα αντικείμενο **Checkbox** 

το οποίο μας το προσφέρει το Android Studio ως widget. Το Checkbox widget πρόκειται για ένα "κουτί" το οποίο όταν πιέσει ο χρήστης στην οθόνη του ορίζεται ως επιλεγμένο. Σαν κείμενο στη παρούσα λίστα ορίζουμε τις μυϊκές ομάδες από τις οποίες ο χρήστης μπορεί να επιλέξει μία ή και περισσότερες για την ημέρα της εβδομάδας που επέλεξε σε προηγούμενο βήμα.

Όπως μιλήσαμε σε αυτήν την περίπτωση θα πρέπει να χρησιμοποιήσουμε έναν δικό μας **Custom Adapter** για να μπορέσουμε να συνθέσουμε μια λίστα που να περιέχει σύνθετα στοιχεία (Text+Check box) . Οπότε σε ένα νέο αρχείο για τον Custom Adapter μας ορίζουμε μια **νέα κλάση Selection** με δυο πεδία , μια μεταβλητή **String** για το όνομα της μυϊκής ομάδας και μια μεταβλητή **Boolean** που υποδηλώνει ότι το **widget checkbox** είναι **επιλεγμένο**(**true**) ή όχι(**false**) εισάγοντας τους αντίστοιχους **Getters** και **Setters**.

Επίσης ορίζουμε την κλάση για τον Custom Adapter η οποία παίρνει ως ορίσματα μια **λίστα τύπου Selection** που

δημιουργήσαμε παραπάνω και μιας τιμής **Context**.

```
public ExercisesCustomAdapter(List<Selection> myList, Context context)
```

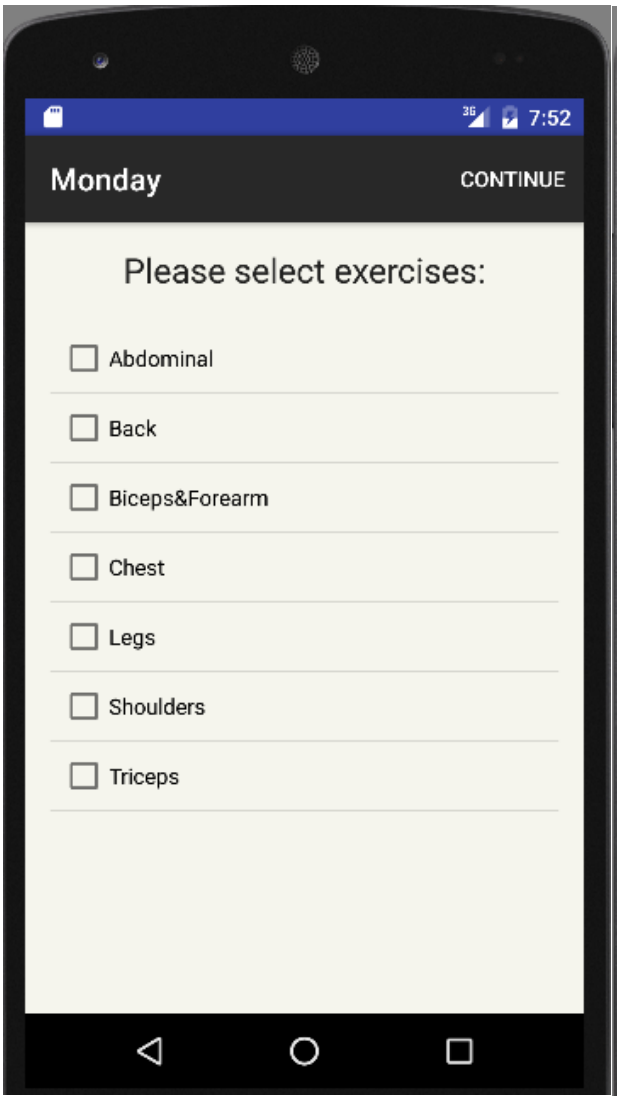
Υλοποιούμε επίσης και έναν **ViewHolder** αποτελούμενος από μια μεταβλητή τύπου **TextView** και μια τύπου **Checkbox** στις οποίες και θα διασυνδέσουμε με τα γραφικά Widget του **context.xml** αρχείου που υλοποιεί την λίστα(**ListView**).

```
private static class MyHolder {

        public TextView name;

        public Checkbox checkbox;

}
```

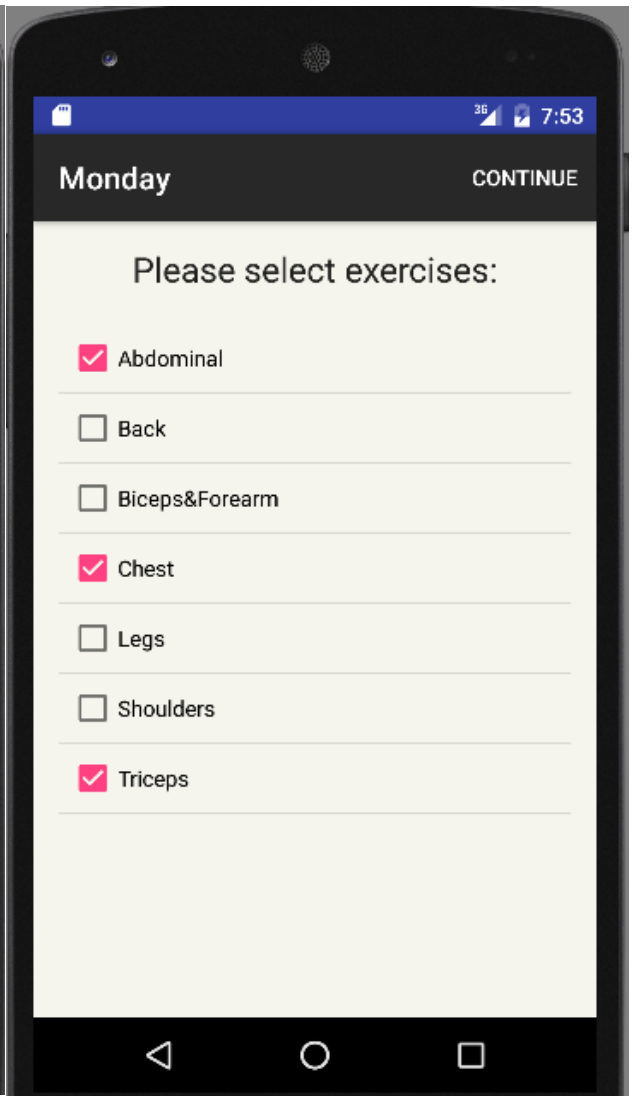
Τέλος στο αρχείο Java της Δραστηριότητας(Activity) για την οποία και υλοποιούμε την λίστα καλούμε τον Custom Adapter που μόλις φτιάξαμε:

```
ExercisesCustomAdapter myAdapter = new ExercisesCustomAdapter(selectionList , this );

ListView myListView = (ListView) findViewById(R.id. myListView );

myListView.setAdapter(myAdapter);
```

Στην επόμενη σελίδα φαίνεται η οθόνη με την παραπάνω λίστα (εικόνες 7.3 και 7.4).



Εικόνα 7.3 Λίστα με τις μυϊκές ομάδες



Εικόνα 7.4 Επιλογή τριών ομάδων στη λίστα

Αφού ο χρήστης αποφασίσει και ορίσει τις μυϊκές ομάδες πρέπει με κάποιον τρόπο να μεταβεί σε επόμενη Δραστηριότητα(Activity) όπου θα επιλέξει τις ασκήσεις. Για αυτό τον λόγο θα προσθέσουμε στο μενού της γραμμής εργαλείων(toolbar) ένα κουμπί "Continue" ("Συνέχεια") το οποίο και θα είναι υπεύθυνο για την εκτέλεση του Intent σε νέα οθόνη. Επίσης ο χρήστης δεν θα είναι σε θέση να συνεχίσει εάν δεν έχει επιλέξει καμία ομάδα από το μενού.

```
@Override
public boolean onCreateOptionsMenu(Menu menu){
      MenuInflater minflater = getMenuInflater();
      minflater.inflate(R.menu.continue_button, menu);
      return true;
}

   @Override
public boolean onOptionsItemSelected(MenuItem item){

     //code executed when Continue button is pressed

    //κώδικας που εκτελείται όταν πιέσουμε το κουμπί Continue

}
```

7.3 Δραστηριότητα(Activity) επιλογής ασκήσεων γυμναστικής και αναδυόμενο παράθυρο

Η συγκεκριμένη δραστηριότητα(βλ. εικόνα 7.5) δεν διαφέρει κατασκευαστικά πολύ από την προηγούμενη δεδομένου ότι χρησιμοποιούμε μια σύνθετη λίστα με ένα αντικείμενο κειμένου TextView και ένα αντικείμενο κουτιού επιλογής Checkbox, το μόνο που προσθέτουμε είναι ακόμη ένα αντικείμενο TextView που θα ονομάσουμε "**Preview**".

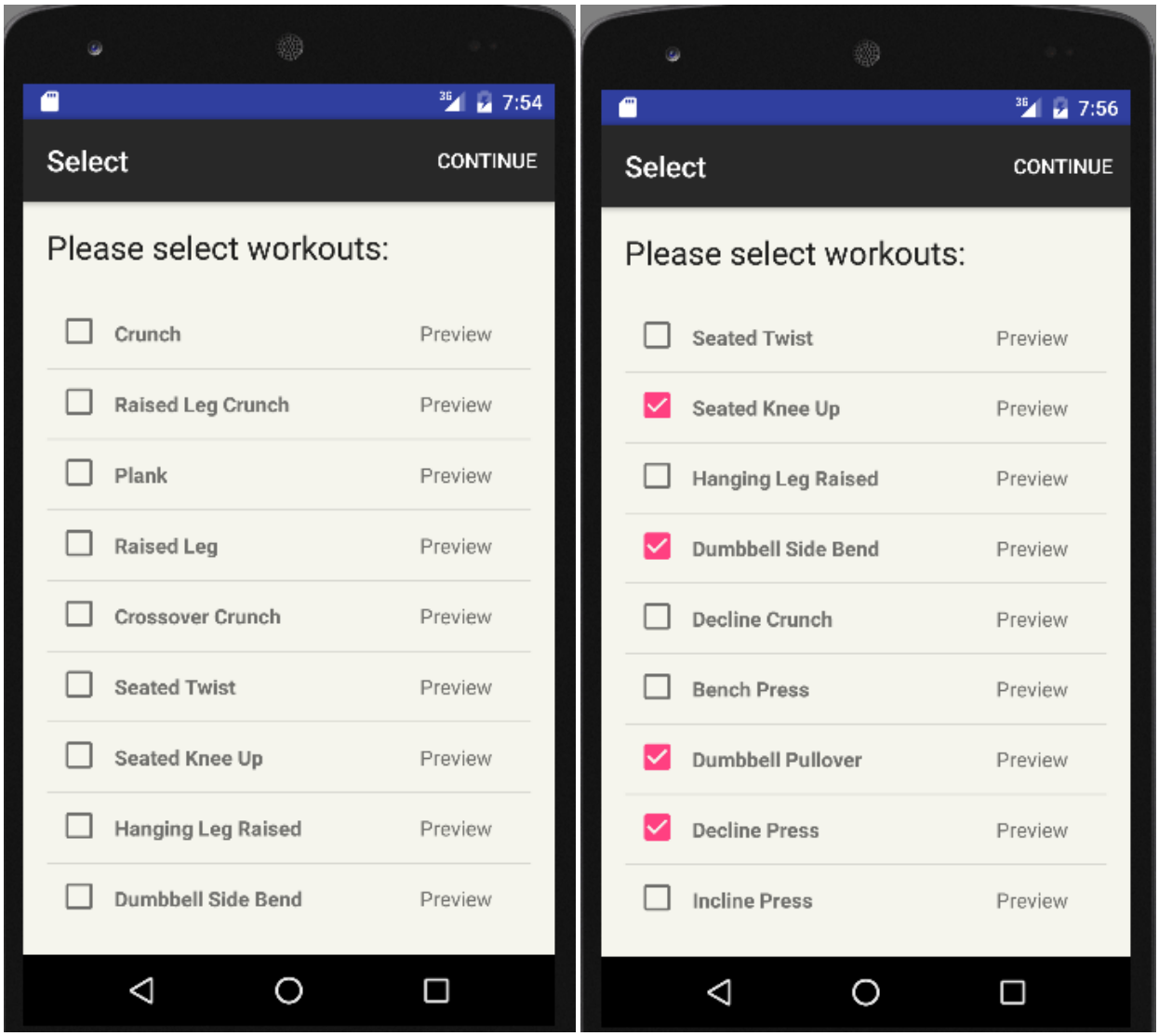
Ο χρήστης θα μπορεί να επιλέγει ασκήσεις που επιθυμεί από την λίστα κανονικά(με απλό άγγιγμα στην οθόνη) επιλέγοντας το κουτί δίπλα από το όνομα της άσκησης, με την μονή διαφορά ότι αν πατήσει το πεδίο "**Preview**" στην λίστα θα εμφανίζεται ένα **αναδυόμενο**

παράθυρο(**pop-up window**) που θα περιλαμβάνει την απεικόνιση εκτέλεσης της συγκεκριμένης άσκησης που επέλεξε.

Δημιουργούμε έναν παρόμοιο **Custom Adapter** όπως στην προηγούμενη ενότητα και περνάμε σαν όρισμα μια λίστα με όλες τις ασκήσεις που ανήκουν στις μυϊκές ομάδες που έχει επιλέξει στη προηγούμενη οθόνη.

```
ListView myListView = (ListView) findViewById(R.id. myListView );

myListView.setAdapter(myAdapter);



myListView.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

                  //intent to pop-up image

                  //Intent σε αναδυόμενη εικόνα

          }

        }

};
```

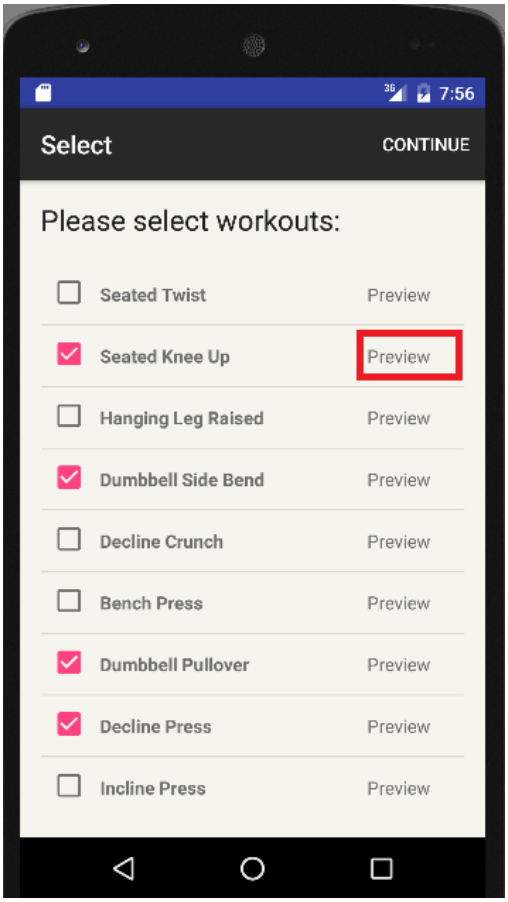
Το **αναδυόμενο παράθυρο**(**pop-up window**) δεν είναι τίποτα άλλο από μια νέα δραστηριότητα(activity) της οποίας ορίζουμε οι διαστάσεις της να μην καλύπτουν ολόκληρη την οθόνη(βλ. εικόνα 7.6) και ως **παρασκήνιο**(**background**) ορίζουμε την κατάλληλη εικόνα που έχουμε βάλει στον φάκελο **res/drawable**. Ο χρήστης κλείνει την εικόνα είτε πιέζοντας μια φορά στο μέρος εξωτερικά της εικόνας είτε πατώντας το πλήκτρο "επιστροφή" στο κάτω μέρος της οθόνης του.

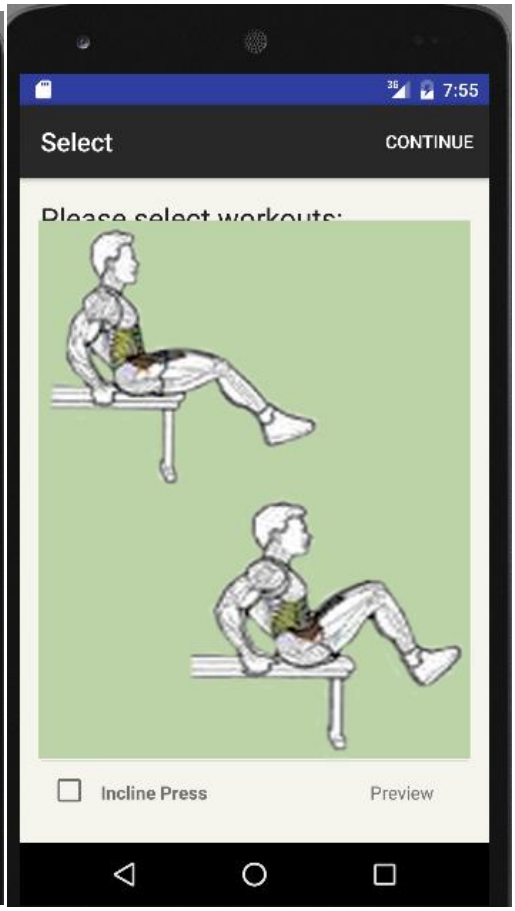


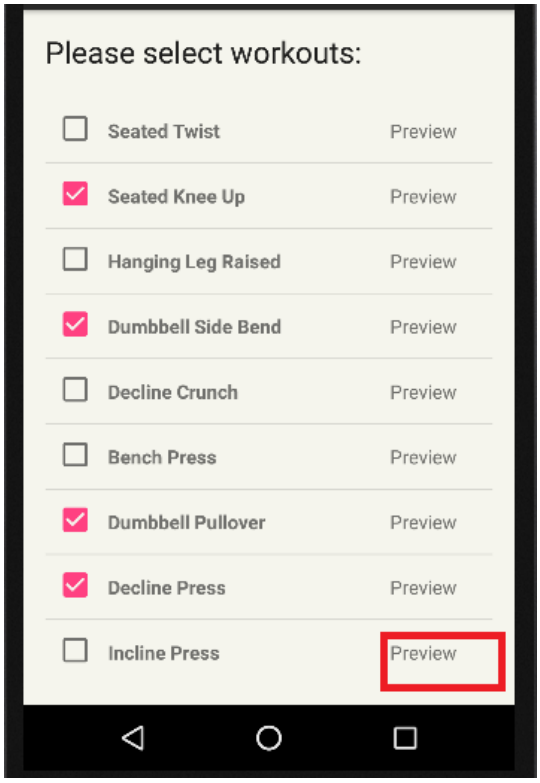
Εικόνα 7.5 Επιλογή ασκήσεων από την λίστα

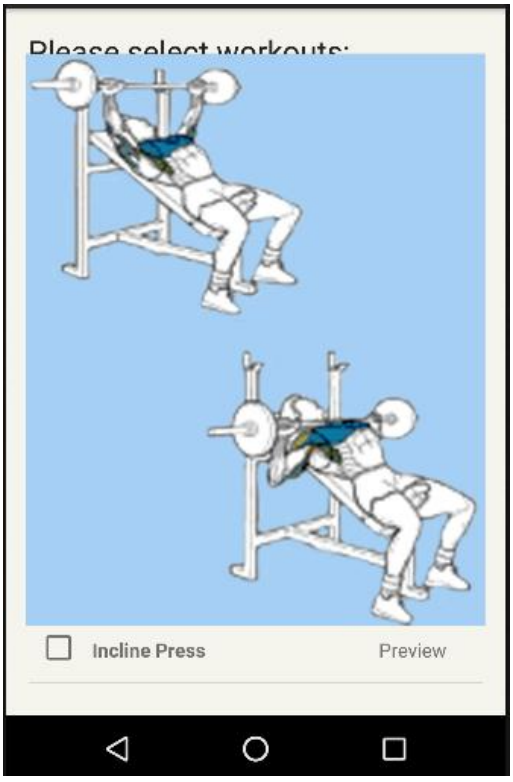
Αφού ο χρήστης επιλέξει ένα σύνολο ασκήσεων από την λίστα που τον ικανοποιούν για το εβδομαδιαίο πρόγραμμά του πατώντας το κουμπί "Continue" θα μεταφερθεί στην τελευταία οθόνη η οποία και περιγράφεται στην επόμενη ενότητα.

Εικόνα 7.6(κάτω) Απεικόνιση εκτέλεσης ασκήσεων σε αναδυόμενο παράθυρο


7:56
Select
CONTINUE
Please select workouts:
Seated Twist
Preview
Seated Knee Up
Preview
Hanging Leg Raised
Preview
Dumbbell Side Bend
Preview
Decline Crunch
Preview
Bench Press
Preview
Dumbbell Pullover
Preview
Decline Press
Preview
Incline Press
Preview


7:55
Select
CONTINUE
Incline Press
Preview


Please select workouts:
Seated Twist
Preview
Seated Knee Up
Preview
Hanging Leg Raised
Preview
Dumbbell Side Bend
Preview
Decline Crunch
Preview
Bench Press
Preview
Dumbbell Pullover
Preview
Decline Press
Preview
Incline Press
Preview


Incline Press
Preview

7.4 Δραστηριότητα(Activity) επιλογής Επαναλήψεων και Σετ

Η Δραστηριότητα αυτή είναι υπεύθυνη όχι μόνο για τον ορισμό των **επαναλήψεων** και των **σετ** στο πρόγραμμα που δημιουργεί ο χρήστης αλλά και για την προσωρινή αποθήκευση όλων αυτών που έχει επιλέξει έως τώρα(ασκήσεις, μυϊκές ομάδες, ημέρες). Την προσωρινή αυτή αποθήκευση την κάνουμε στην περίπτωση που ο χρήστης θελήσει να ακυρώσει την όλη διαδικασία δημιουργίας νέου προγράμματος γυμναστικής, οπότε δεν θα θέλαμε να μείνουν άκυρα δεδομένα στο σύστημα.

Για άλλη μια φορά η λειτουργικότητα αυτής της Activity βασίζεται σε μια **λίστα** με **Custom Adapter** και υπάρχει ισχυρή διασύνδεση πληροφοριών με τις προηγούμενες δραστηριότητες, δηλαδή μέσω των **Intents** θα λάβει τις ασκήσεις και μυϊκές ομάδες που όρισε ο χρήστης στα προηγούμενα βήματα.

Ο χρήστης βλέπει στην οθόνη του την λίστα των ασκήσεων που επέλεξε προηγουμένως(βλ. εικόνα 7.7) και καλείται σε αυτό το βήμα να ορίσει τα **σετ** και τις **επαναλήψεις** για κάθε μια ξεχωριστά. Επίσης θα μπορεί για άλλη μια φορά για λόγους πρακτικότητας να κάνει "**Preview**" των ασκήσεων (βλ. εικόνα 7.8) με τον μηχανισμό **αναδυόμενου παραθύρου**(**pop-up window**) που υλοποιήσαμε με την διαφορά ότι αντί για "απλό άγγιγμα"(**tap**) στην οθόνη ο χρήστης κάνει "πατημένο άγγιγμα" (**hold**).

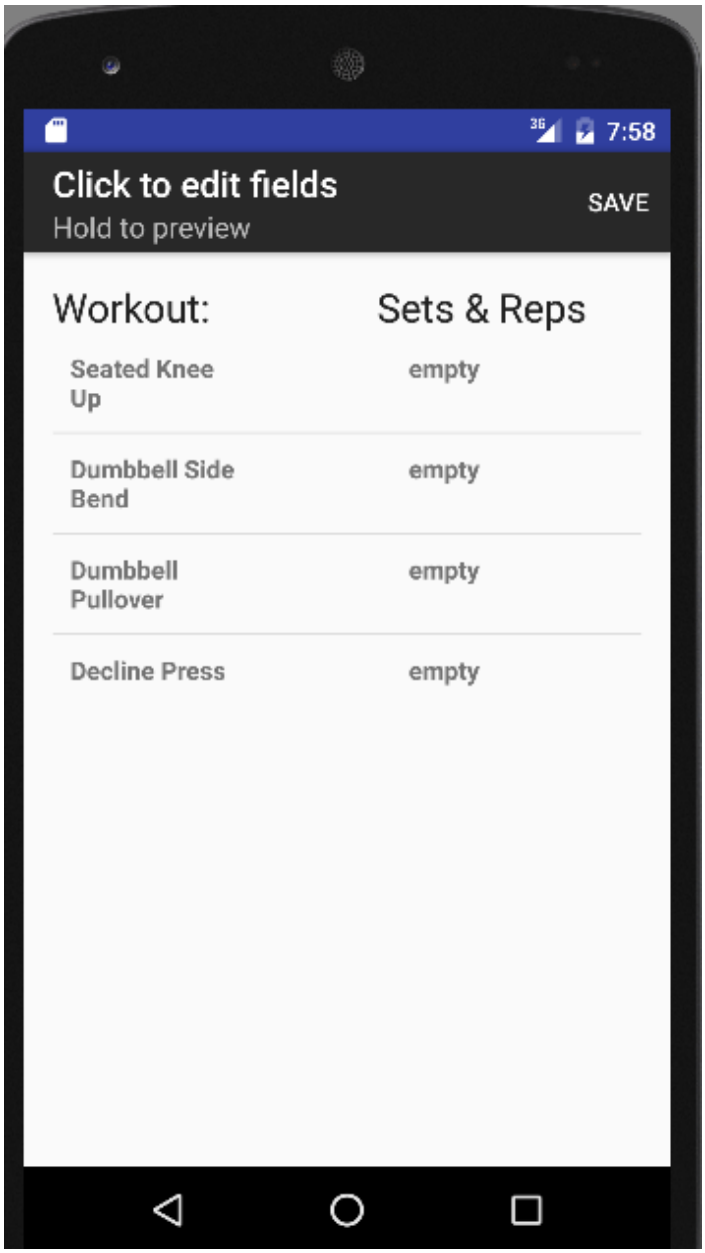
Ο **Custom Adapter** αυτή την φορά αποτελείται από δυο αντικείμενα κειμένου **TextViews ,** το ένα περιέχει το όνομα της άσκησης και το άλλο ένα πεδίο το οποίο ο χρήστης πατώντας το θα μπορεί να ορίσει τα σετ και τις επαναλήψεις από ένα αναδυόμενο παράθυρο. Για να ορίσει τα σετ ο χρήστης κάνει "απλό άγγιγμα"(**tap**) στο στοιχείο της λίστας που επιθυμεί. Επίσης έχει την δυνατότητα να μην ορίσει σετ και να το αφήσει με την προεπιλογή "**empty - άδειο**" .

OnItemClickListener - Κώδικας για "απλό άγγιγμα"(**tap**) στην οθόνη

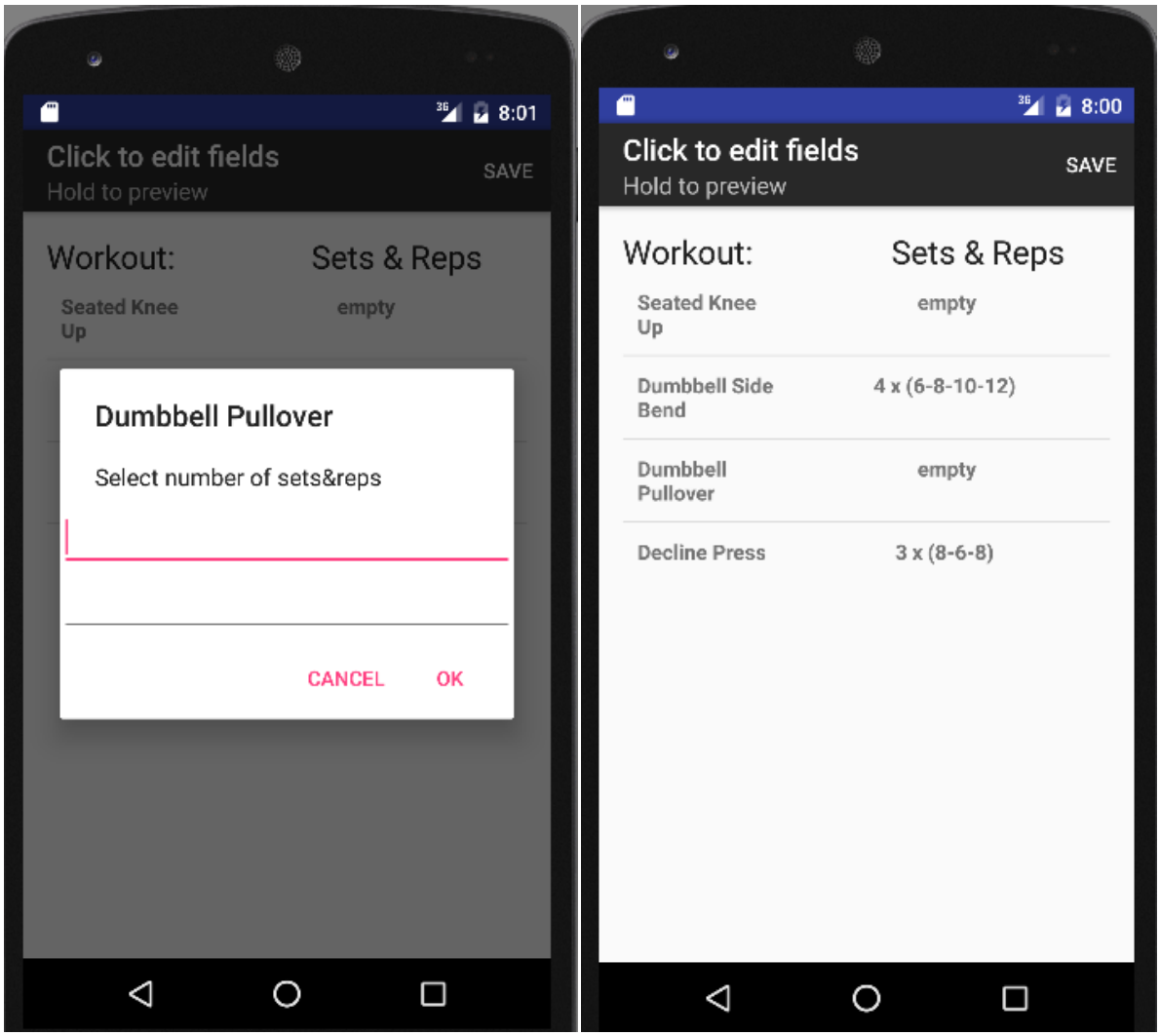
```
myListView.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view, final int position, long id) {

                  AlertDialog.Builder alert = new AlertDialog.Builder(this);

                  //Κώδικας για Παράθυρο Προειδοποίησης -  Alert dialog

          }

        }
);
```

OnItenLongClickListener - Κώδικας για "πατημένο άγγιγμα" (**hold**) στην οθόνη

```
myListView.setOnItemLongClickListener(
        new AdapterView.OnItemLongClickListener() {
          @Override
          public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {

                  //Κώδικας διαδικασίας ορισμού Σετ και Επαναλήψεων

          }

        }
);
```



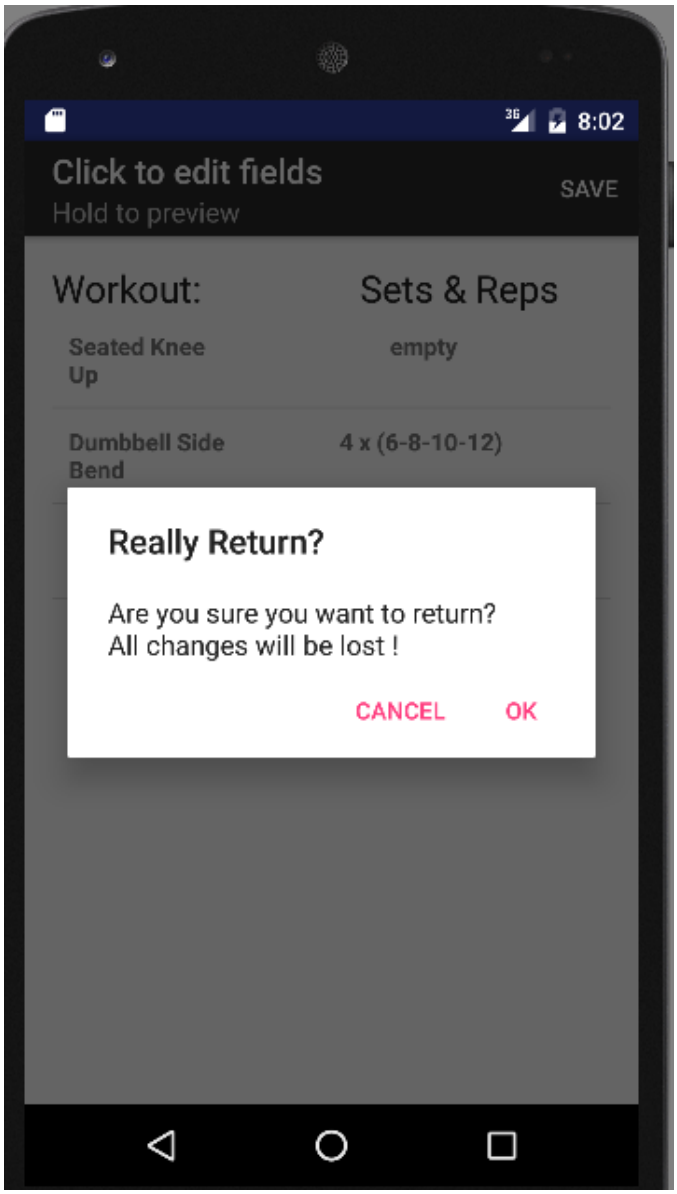
Εικόνα 7.7 Λίστα με ασκήσεις που επέλεξε ο χρήστης



Εικόνα 7.8 Ορισμός σετ και επαναλήψεων σε αναδυόμενο παράθυρο

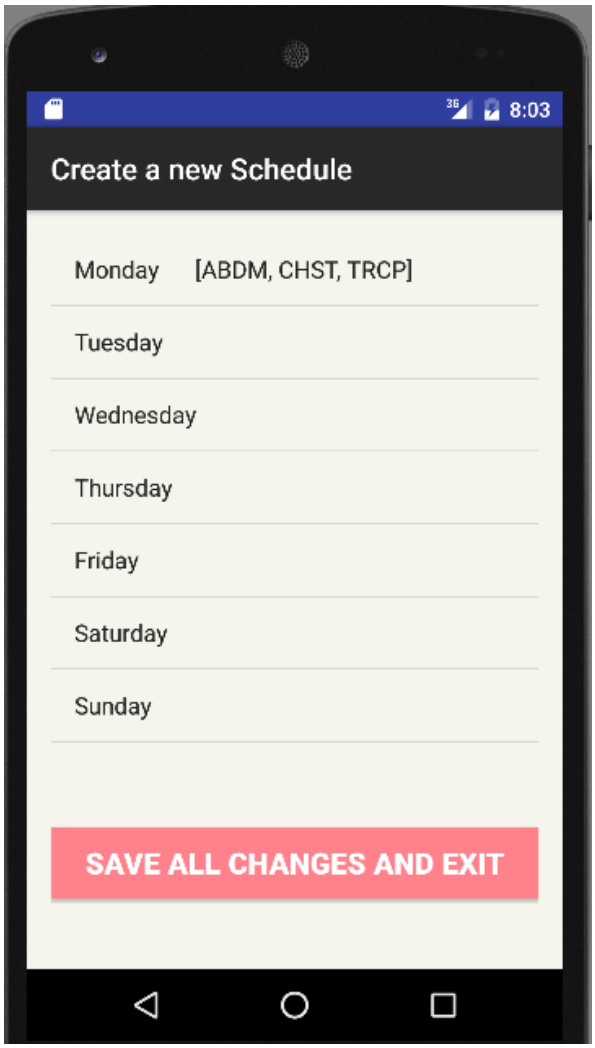
Στο παράθυρο επιλογής σετ και επαναλήψεων ο χρήστης καλείται να επιλέξει στο πάνω πεδίο έναν αριθμό που θα αντιπροσωπεύει τα σετ και στο κάτω πεδίο πόσες επαναλήψεις για κάθε σετ επιθυμεί. Στο παράδειγμα ορίσαμε για την άσκηση Decline Press **3 σετ** με **8-6-8 επαναλήψεις** αντίστοιχα.

Στην περίπτωση που ο χρήστης θελήσει να προσθαφαιρέσει ασκήσεις από στην λίστα μπορεί να το κάνει πατώντας το πλήκτρο "επιστροφή" στην συσκευή του αλλά θα χάσει όσες τιμές έχει ορίσει στα σετ και στις επαναλήψεις. Ένα παράθυρο προειδοποίησης θα εμφανιστεί στη οθόνη του για επιβεβαίωση(βλ. εικόνα 7.9).



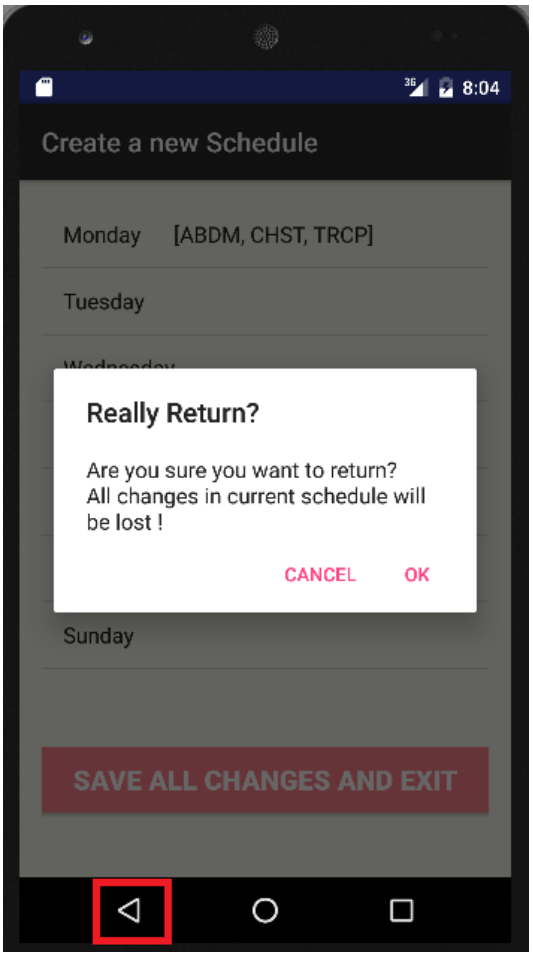
Εικόνα 7.9 Παράθυρο προειδοποίησης

Όταν ο χρήστης είναι ικανοποιημένος με τις επιλογές του είναι σε θέση τώρα να καταχωρίσει τις αλλαγές, όπως αναφέραμε, προσωρινά στο σύστημα πατώντας το κουμπί "**Save**" (**Αποθήκευση**) που υλοποιούμε στην **γραμμή εργαλείων** (**toolbar**). Αυτό που γίνεται είναι να κρατηθούν τα δεδομένα σε έναν προσωρινό χώρο στην Βάση δεδομένων μας, μέσω κατάλληλων συναρτήσεων που υλοποιούμε, όταν ο χρήστης πατήσει το κουμπί. Αφού όλα εκτελεστούν σωστά επιστρέφουμε στην αρχική Activity της ενότητας 7.1 μέσω Intent από την οποία ο χρήστης μπορεί να ορίσει ασκήσεις για τις επόμενες μέρες της εβδομάδας ή ακόμη και να ξανά ορίσει την ίδια ημέρα από την αρχή. Στο παράδειγμά μας ορίσαμε για την ημέρα Δευτέρα:



Πατώντας το πλήκτρο SAVE ALL CHANGES AND EXIT όλα τα προσωρινά δεδομένα κατοχυρώνονται μόνιμα στο σύστημα και τελικώς δημιουργείται το ολοκληρωμένο πρόγραμμα γυμναστικής με όνομα που είχε ορίσει ο χρήστης στην αρχή.

Πατώντας το πλήκτρο ◁ ένα παράθυρο προειδοποίησης εμφανίζεται ενημερώνοντας ότι όλα τα προσωρινά δεδομένα και αλλαγές θα χαθούν και η διαδικασία δημιουργίας προγράμματος γυμναστικής θα ακυρωθεί(βλ. εικόνα 7.10).

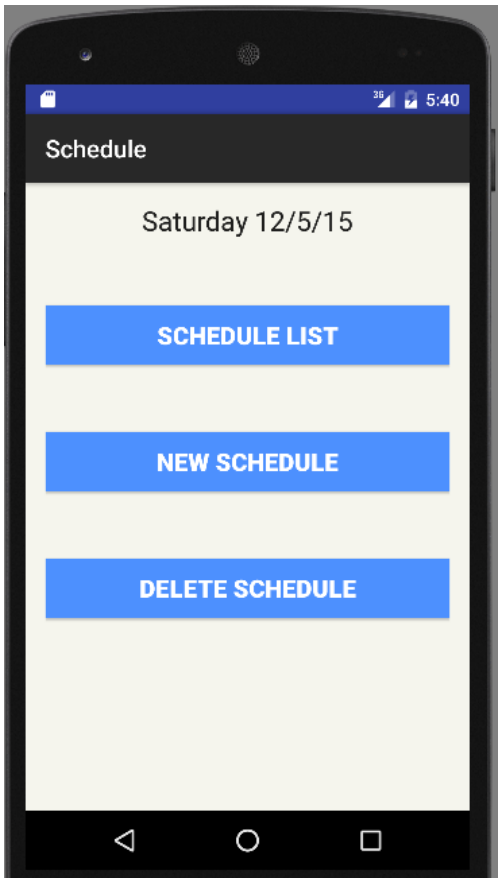


Εικόνα 7.10 Παράθυρο ακύρωσης διαδικασίας

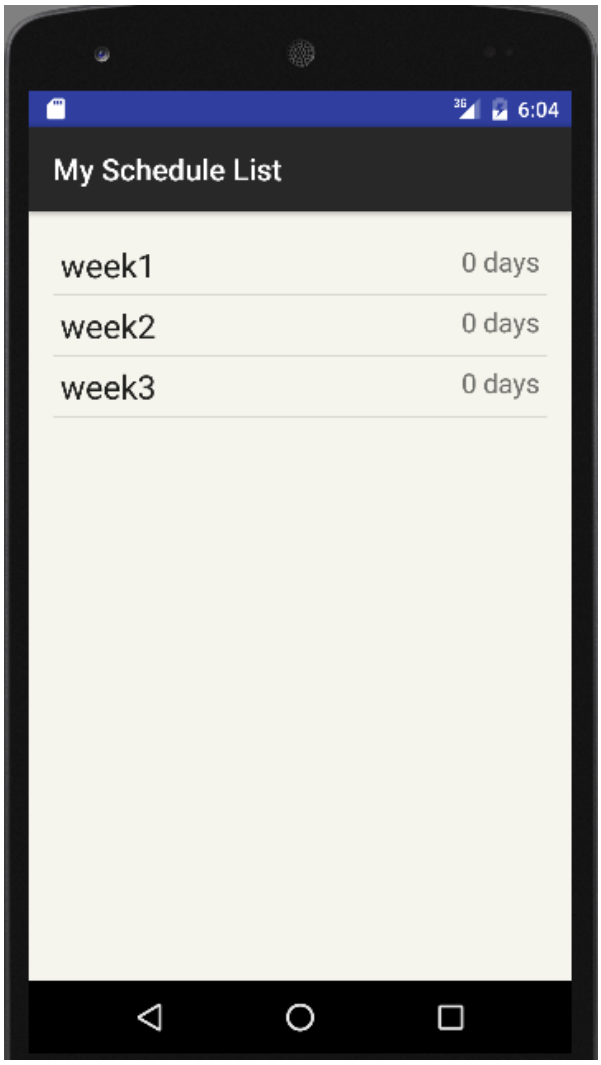
7.5 Δραστηριότητες(Activities) επιλογής και προβολής προγραμμάτων από λίστα

Αφού περιγράψαμε την διαδικασία δημιουργίας ενός προγράμματος γυμναστικής σε αυτή την ενότητα θα παρουσιάσουμε τις Δραστηριότητες τις οποίες θα ανοίξει ο χρήστης έτσι ώστε να επιλέξει και να προβάλει τα έως τώρα προγράμματα που έχει ο ίδιος δημιουργήσει.

Από το βασικό μενού προγραμμάτων γυμναστικής ο χρήστης πιέζει το κουμπί **Schedule List** όπου και θα μεταβεί σε μια νέα οθόνη.

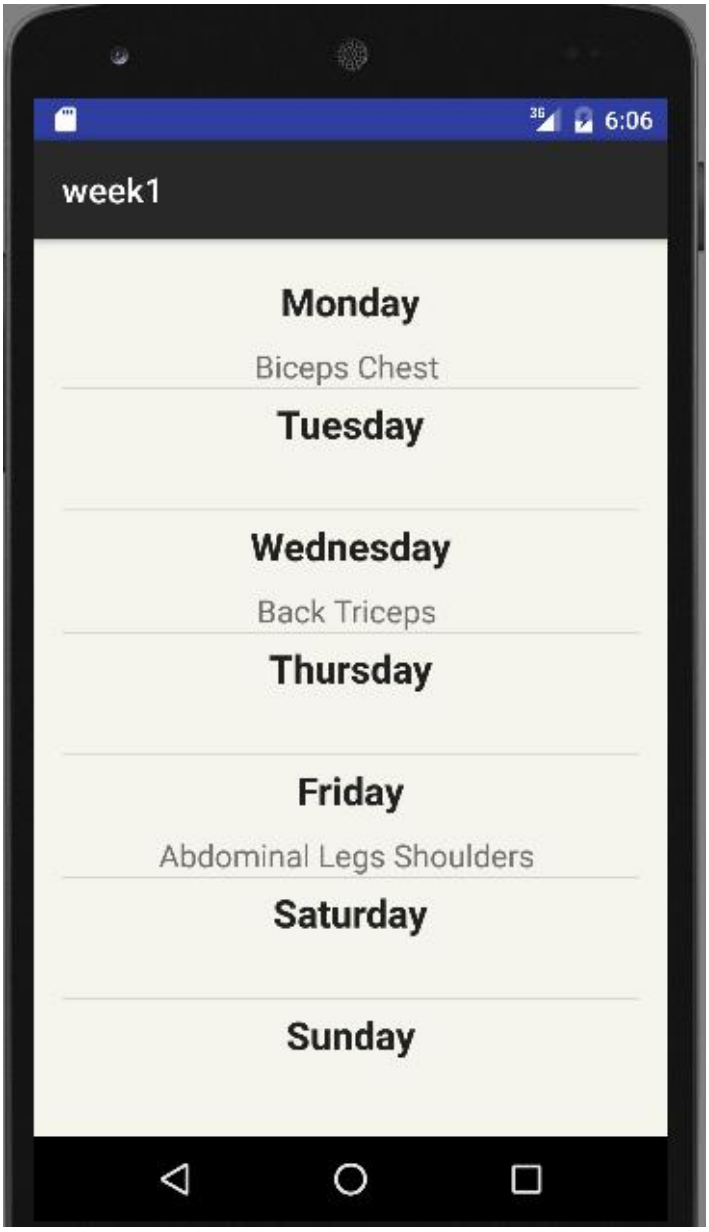


Η νέα αυτή οθόνη (βλ. εικόνα 7.11) αποτελείται από μια Activity που περιέχει λίστα (**ListView**) με **Custom Adapter** από δύο αντικείμενα κειμένου (**TextView**) ένα για το όνομα και ένα για την διάρκεια του προγράμματος. Τα ονόματα των προγραμμάτων τα αντλεί από την **Βάση Δεδομένων** που έχουν καταχωρηθεί μόνιμα καλώντας κατάλληλη συνάρτηση. Την διάρκεια του κάθε προγράμματος γυμναστικής την βρίσκει στα **Shared Preferences** στο αρχείο **DurationLog.xml** που αναφέρθηκε σε προηγούμενη ενότητα.



Εικόνα 7.11 Λίστα με ονόματα των προγραμμάτων γυμναστικής του χρήστη

Επιλέγοντας ένα όνομα από την λίστα ο χρήστης μεταβαίνει στην επόμενη Δραστηριότητα (βλ. εικόνα 7.12) με τις μέρες της εβδομάδας. Πάλι εδώ υλοποιείται **Custom Adapter** σε λίστα με δύο αντικείμενα κείμενου(**TextView**) στοιχισμένα το ένα κάτω απ' το άλλο, με το πάνω TextView να περιέχει την ημέρα και το κάτω TextView τις μυϊκές ομάδες που είχε ορίσει ο χρήστης κατά την δημιουργία.

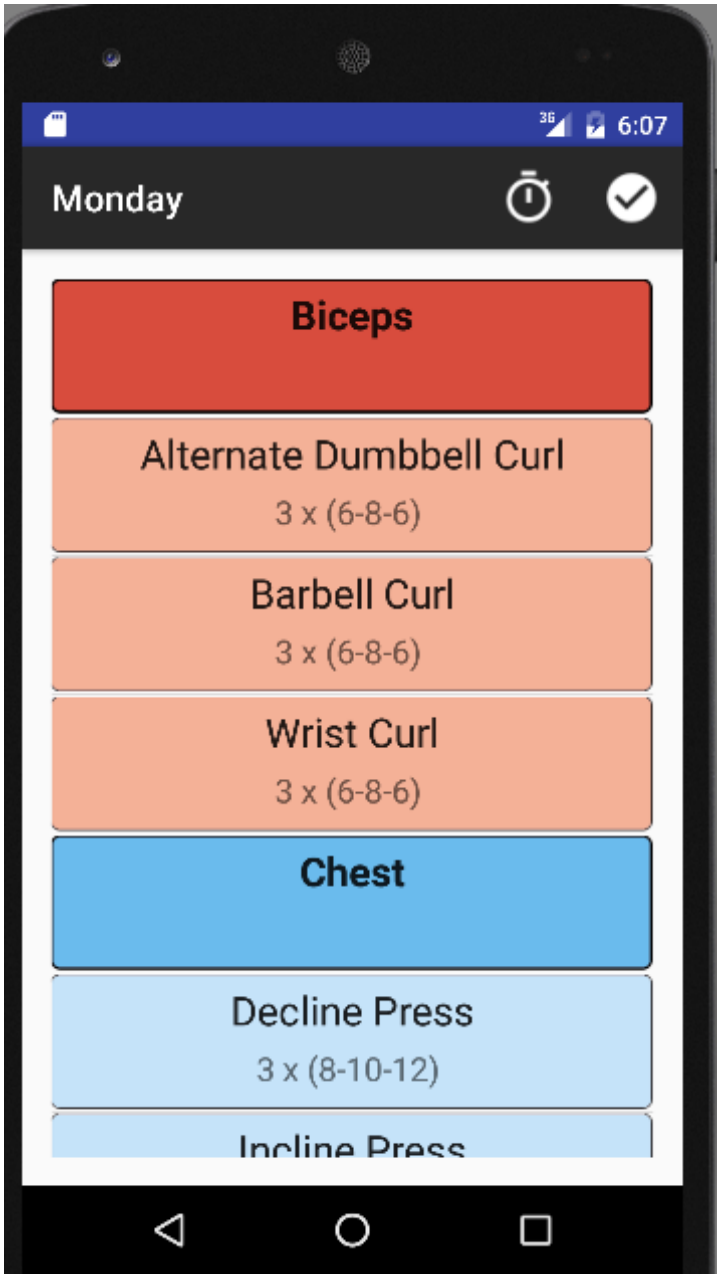


Εικόνα 7.12 Εβδομαδιαίο πρόγραμμα γυμναστικής με όνομα week1

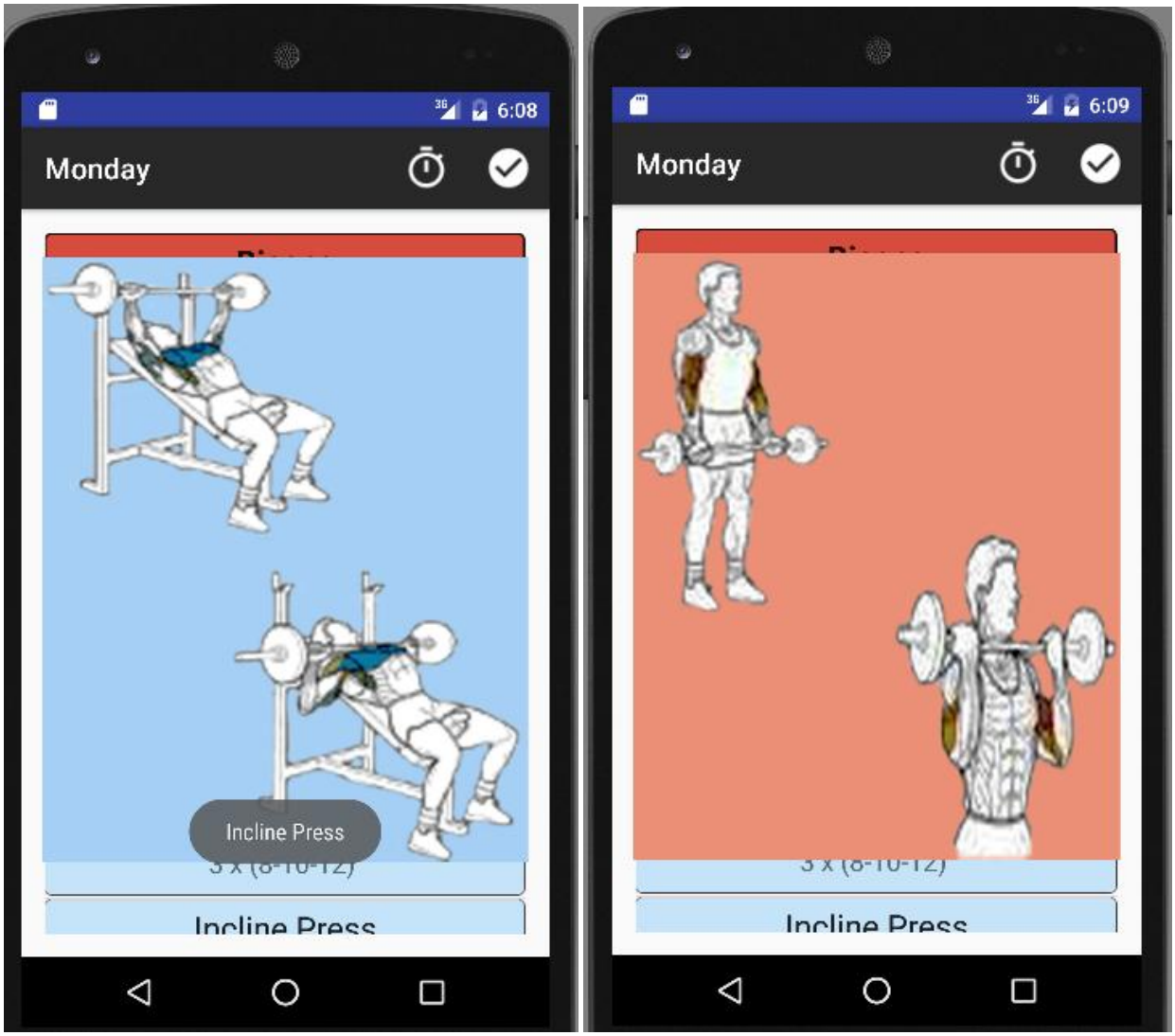
Από αυτό το μενού ο χρήστης επιλέγει μια από τις μέρες είτε είναι η σημερινή είτε κάποια άλλη και μια νέα τελική οθόνη θα εμφανιστεί στην συσκευή του. Πρόκειται για την δραστηριότητα η οποία είναι υπεύθυνη να παρουσιάσει όλες τις πληροφορίες οι οποίες βρίσκονται αποθηκευμένες στην Βάση Δεδομένων σχετικά με το όνομα του προγράμματος που έχει διαλέξει ο χρήστης να προβάλει.

Η οθόνη αποτελείται από μία έγχρωμη λίστα , με κάθε μυϊκή ομάδα να ανήκει σε διαφορετικό χρώμα, με τις ασκήσεις και τα αντίστοιχα σετ και επαναλήψεις τους (βλ. εικόνα 7.13). Αυτό υλοποιείται με έναν **Custom Adapter** με δυο TextView αντικείμενα τα οποία λαμβάνει τιμές για τα ονόματα και τα σετ/επαναλήψεις από την Βάση Δεδομένων και τα κατατάσσει σε κατάλληλες μυϊκές ομάδες με το κατάλληλο χρώμα στο παρασκήνιο (background).

Ο χρήστης πιέζοντας κάποιο στοιχείο στην οθόνη του θα ανοίξει ένα αναδυόμενο παράθυρο (pop-up window) με εικόνα για την συγκεκριμένη άσκηση (βλ. εικόνα 7.14) την οποία και μπορεί να κλείσει πιέζοντας οπουδήποτε εξωτερικά στην εικόνα. Είναι ακριβώς ο ίδιος μηχανισμός που υλοποιήθηκε και σε άλλες Δραστηριότητες σε προηγούμενες ενότητες.

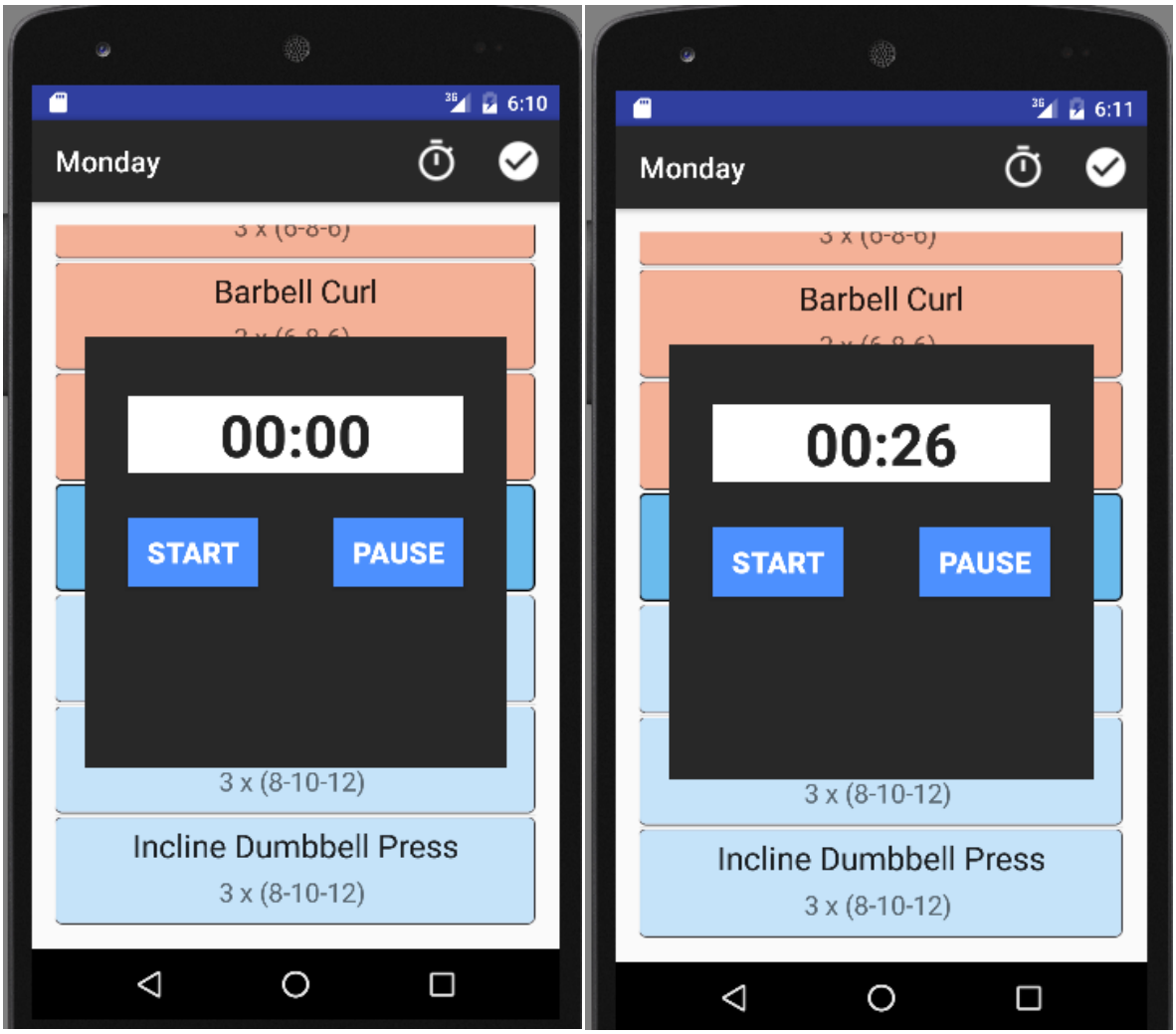


Εικόνα 7.13 Έγχρωμο μενού/λίστα με τις ασκήσεις που έχει ορίσει ο χρήστης



Εικόνα 7.14 Αναδυόμενο(pop up) παράθυρο προβολής των ασκήσεων

Στην συνέχεια υλοποιήσουμε , μιας και είναι πρακτικό κατά την διάρκεια εκγύμνασης, ένα ψηφιακό χρονόμετρο το οποίο και θα ανοίγει από την γραμμή εργαλείων (**toolbar**) πιέζοντας το εικονίδιο  Το ψηφιακό αυτό χρονόμετρο(εικόνα 7.15) αποτελείται από ένα πεδίο στο οποίο αναγράφονται τα λεπτά και δευτερόλεπτα και δυο κουμπιά, το ένα για **έναρξη** (**start**) και το άλλο για **παύση** (**pause**).



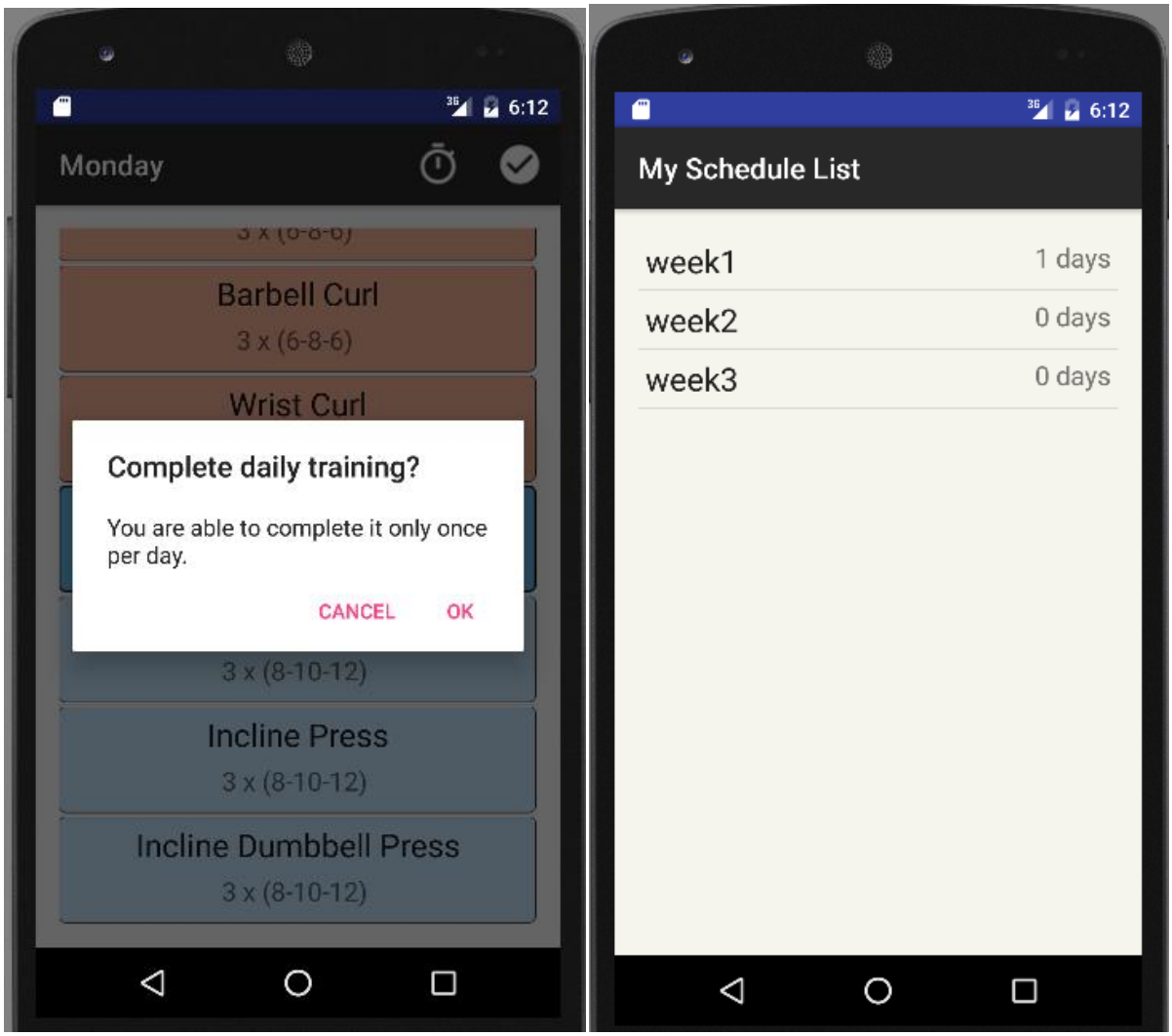
Εικόνα 7.15 Ψηφιακό χρονόμετρο

Τέλος υλοποιούμε και έναν μηχανισμό καταγραφής της διάρκειας που ο χρήστης χρησιμοποιεί το κάθε πρόγραμμα γυμναστικής ξεχωριστά.

Η βασική ιδέα είναι, κάθε ημέρα που ο χρήστης τελειώνει με την εκγύμνασή του να κρατάει σε ένα αρχείο για να θυμάται πόσες φορές έχει κάνει χρήση του συγκεκριμένου προγράμματος για μελλοντική αναφορά. Αυτή η πληροφορία αποθηκεύεται στο αρχείο **DurationLog.xml** (Shared Preferences)στο σύστημα με κατάλληλο μηχανισμό όταν πιέσει το κουμπί  από την γραμμή εργαλείων (toolbar).

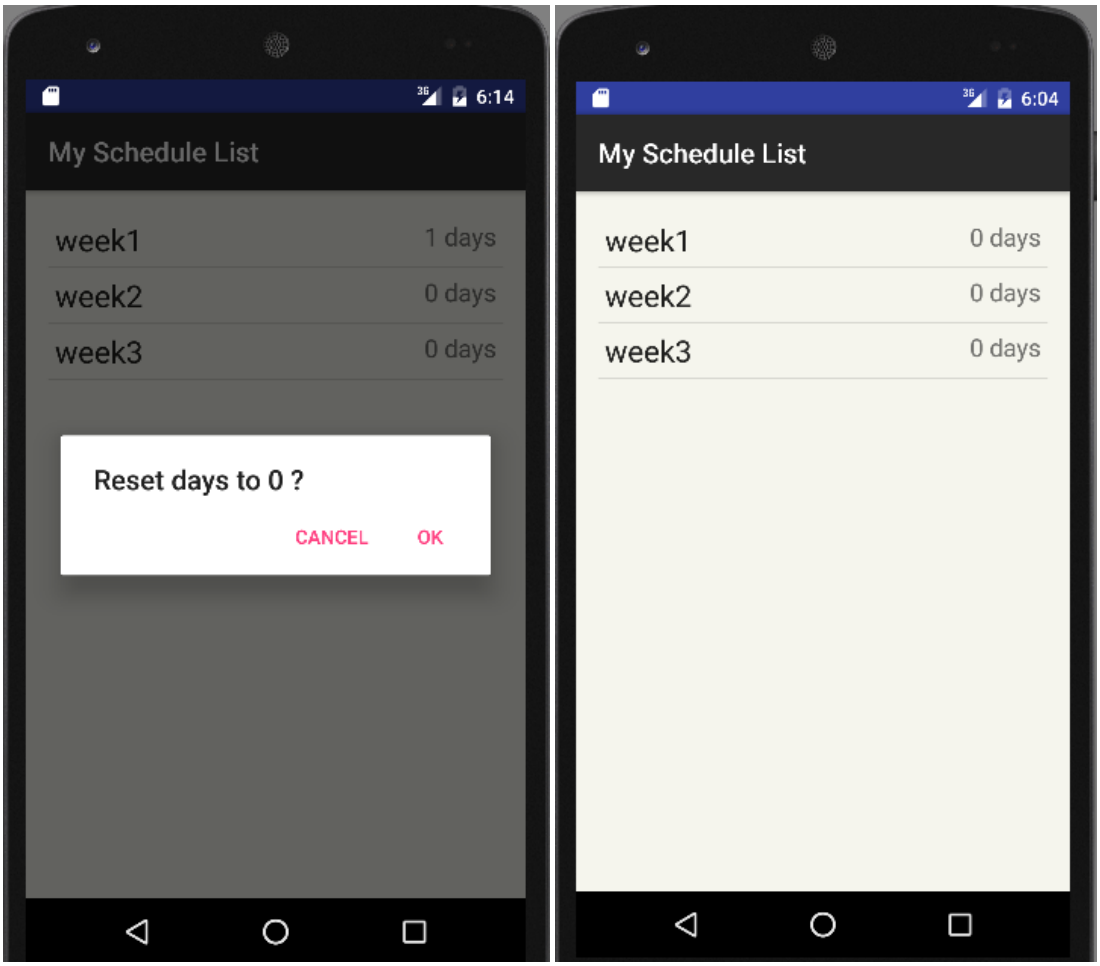
Τότε ένα αναδυόμενο παράθυρο για επιβεβαίωση θα εμφανιστεί περιμένοντας τον χρήστη να αποδεχτεί και έτσι θα αυξήσει τον δείκτη χρήσης του προγράμματος κατά μια ημέρα(βλ. εικόνα 7.16).

Να σημειώσουμε ότι την παραπάνω διαδικασία ολοκλήρωσης και αύξησης της διάρκειας του προγράμματος μπορεί να την πραγματοποιήσει κανείς μόνο μια φόρα την ημέρα(ανά 24ωρο) . Δηλαδή πατώντας το πλήκτρο ολοκλήρωσης πολλάπλες φορές δεν θα αυξήσει την μέτρηση.



Εικόνα 7.16 Μηχανισμός ολοκλήρωσης ημερήσιας εκγύμνασης

Για να μηδενιστεί ο μετρητής πιέζουμε εκτεταμένα πάνω σε ένα όνομα από την λίστα και γίνεται αρχικοποίηση των ημερών σε 0.

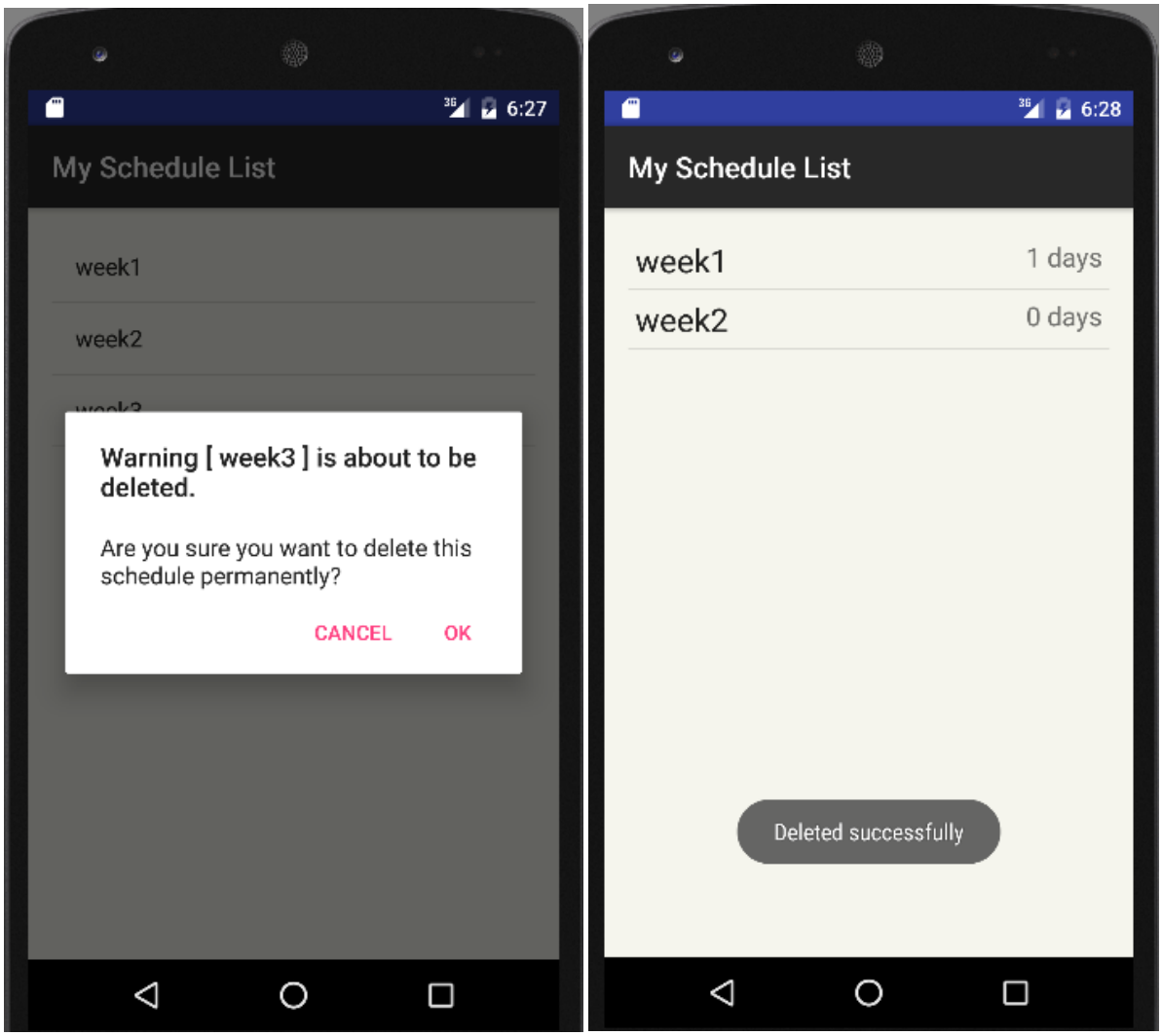


Εικόνα 7.17 Μηδενισμός μετρητή διάρκειας του προγράμματος week1

7.6 Δραστηριότητα(Activity) διαγραφής προγράμματος από την Βάση Δεδομένων

Έχουμε περιγράψει την διαδικασία δημιουργίας και προβολής ενός προγράμματος γυμναστικής. Σε αυτό το σημείο θα περιγράψουμε την Δραστηριότητα από την οποία ο χρήσης θα διαγράφει μόνιμα όλα τα δεδομένα που βρίσκονται είτε στην Βάση Δεδομένων είτε σε αρχεία των Shared Preferences και ανήκουν σε ένα πρόγραμμα γυμναστικής το οποίο δεν χρειάζεται πλέον.

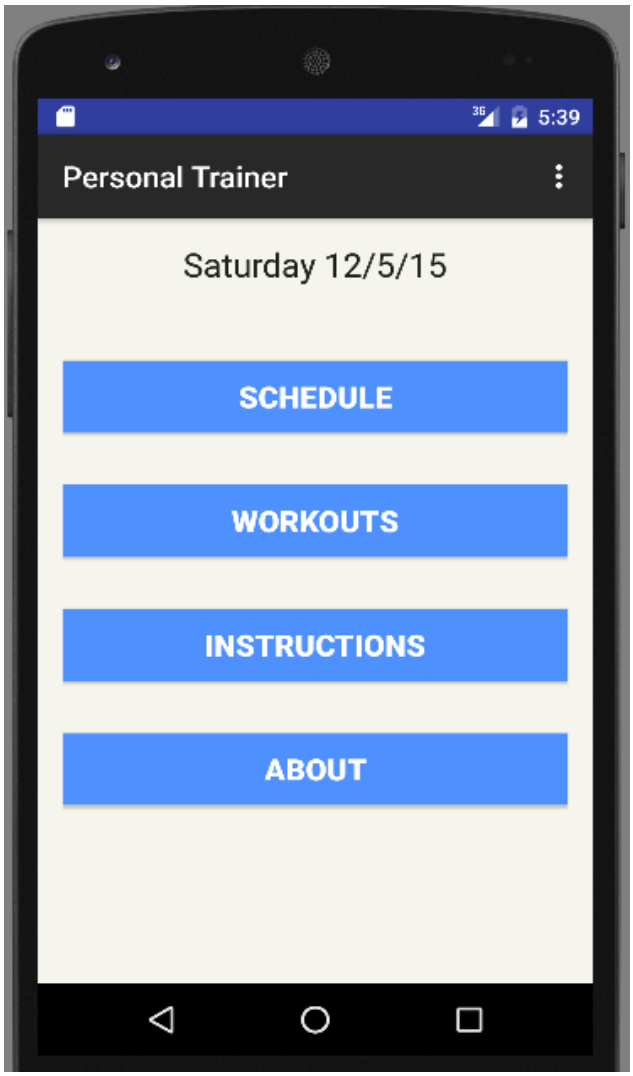
Από το βασικό μενού επιλέγει "Delete Schedule" (διαγραφή προγράμματος) και μία λίστα με τα ονόματα όλων των προγραμμάτων που υπάρχουν στην Βάση Δεδομένων θα εμφανιστούν στην οθόνη. Πατώντας πάνω σε ένα όνομα ο χρήστης θα ερωτηθεί μέσω ενός αναδυόμενου παραθύρου και καλείται να επιβεβαιώσει την ενέργεια μόνιμης διαγραφής του τρέχοντος προγράμματος. Αποδέχοντας την ενέργεια καλείται η κατάλληλη συνάρτηση στο παρασκήνιο και όλα τα σχετικά με το πρόγραμμα δεδομένα διαγράφονται από την μόνιμη μνήμη της Βάσης Δεδομένων και όλων τον αρχείων και επιστρέφει στο βασικό μενού. Πλέον το όνομα αυτό δεν θα εμφανίζεται στην λίστα των ενεργών προγραμμάτων(βλ. εικόνα 7.18).



Εικόνα 7.18 Μόνιμη διαγραφή του προγράμματος με όνομα week3

8 Δραστηριότητα με Spinner

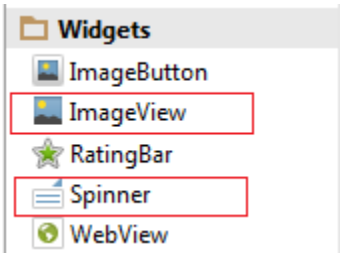
Έως τώρα από το αρχικό μενού(εικόνα 8.1) καλύψαμε όλες τις λειτουργίες και δραστηριότητες που μπορεί ο χρήστης να χρησιμοποιήσει ξεκινώντας απ' το κουμπί Schedule.

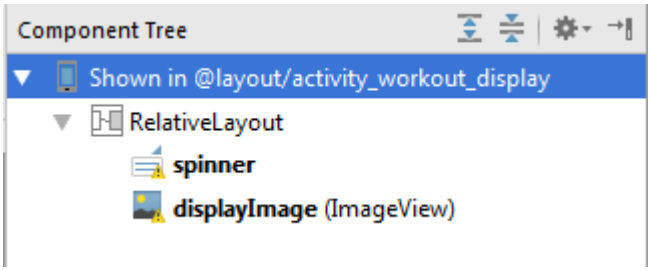


Εικόνα 8.1 Αρχικό μενού

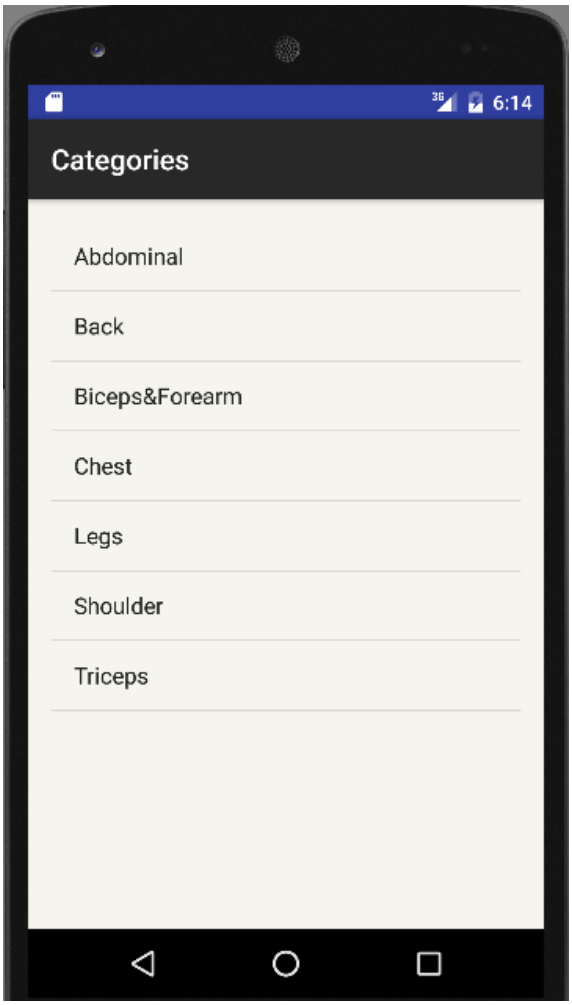
Συνεχίζοντας αναλύουμε τις τελευταίες δραστηριότητες στις οποίες μπορεί να πλοηγηθεί ο χρήστης αρχίζοντας από το κουμπί **Workouts**. Σκοπός αυτής της δραστηριότητας είναι να παρέχει στον χρήστη μια γρήγορη πρόσβαση στον τρόπο με τον οποίο μπορεί να εκτελέσει σωστά μια άσκηση, δηλαδή με εικόνες όπως περιγράψαμε σε άλλες ενότητες αντί όμως να εμφανίζονται μέσω αναδυόμενων παραθύρων θα υλοποιηθεί μέσω μηχανισμού **Spinner**.

Επιλέγοντας το κουμπί **Workouts** θα μεταβεί σε μια νέα οθόνη με μια λίστα με όλες τις κατηγορίες ασκήσεων (βλ. εικόνα 8.2). Σε αυτό το σημείο επιλέγοντας κάποιο στοιχείο από την λίστα θα καταλήξει σε μια νέα Δραστηριότητα η οποία αποτελείται από ένα αντικείμενο **Spinner** από την παλέτα των **widgets** και από ένα αντικείμενο **ImageView**.






```
ImageView displayImage = (ImageView) findViewById(R.id.displayImage);

Spinner spinner = (Spinner) findViewById(R.id.spinner);
```

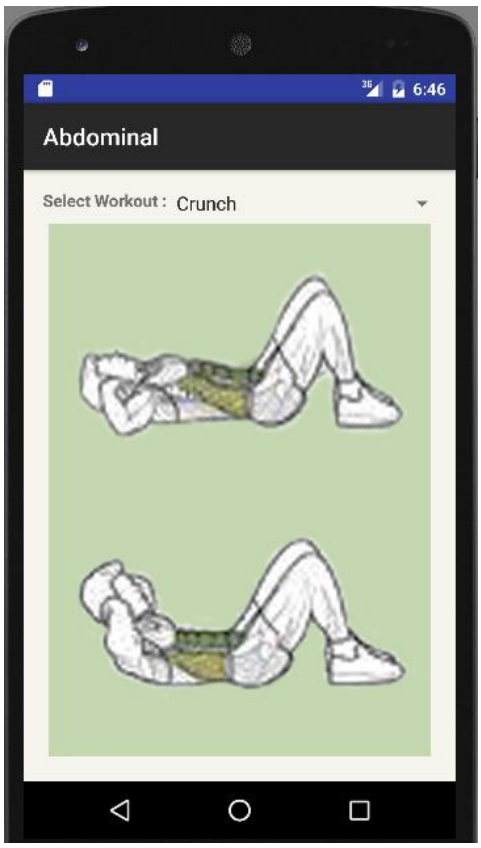


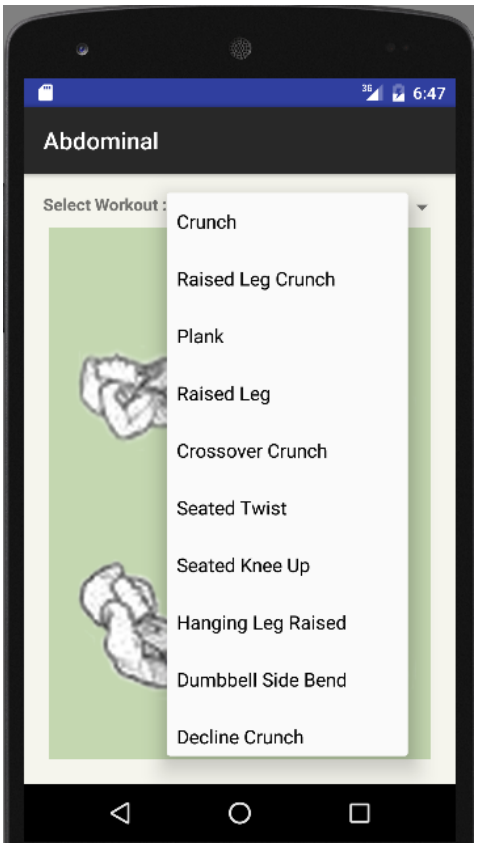
Εικόνα 8.2 Λίστα κατηγοριών

Το **Spinner** πρόκειται για ένα μηχανισμό **drop down** μενού που ως επιλογές έχουμε ορίσει όλες τις ασκήσεις που ανήκουν στην κατηγορία που μόλις επέλεξε ο χρήστης. Διαλέγοντας κάποια επιλογή απ' το **drop down** μενού η εικόνα στην περιοχή του **ImageView** αλλάζει δυναμικά έτσι ώστε να αντιπροσωπεύσει την επιλογή αυτή.

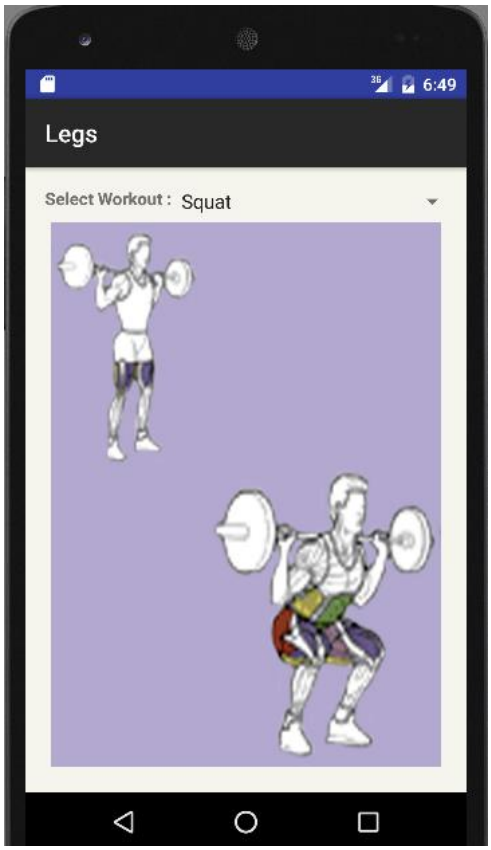
Κώδικας υλοποίησης για Spinner:

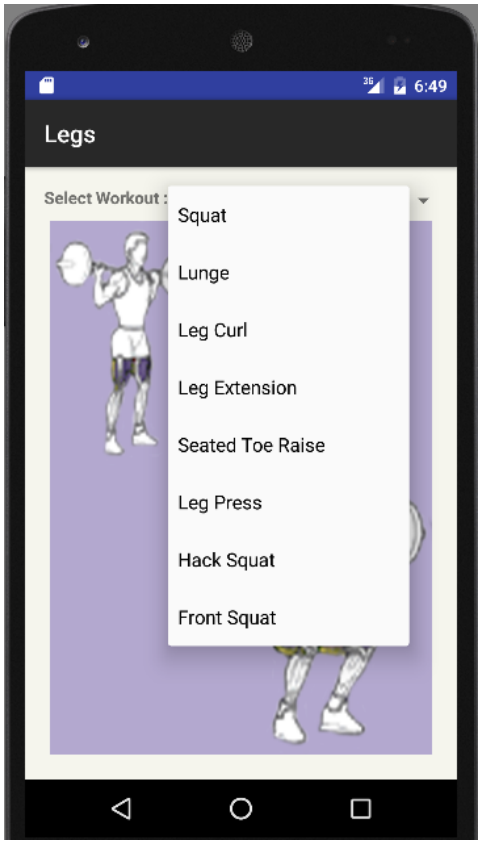
```
    ArrayAdapter<CharSequence> adapter;

    //init adapter
    adapter = ArrayAdapter.createFromResource(this, R.array.init_list, android.R.layout.simple_spinner_item);

  //Switch case για επιλογή κατηγορίας και ορισμός δικής μας λίστας



 adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
       spinner.setAdapter(adapter);

      spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
          @Override
           public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {


                    //Συνάρτηση για επιλογή της κατάλληλης εικόνας

                    displayImage.setImageResource(R.drawable.selected_image);
           }

          @Override
          public void onNothingSelected(AdapterView<?> parent) {

          }
     });
}
```

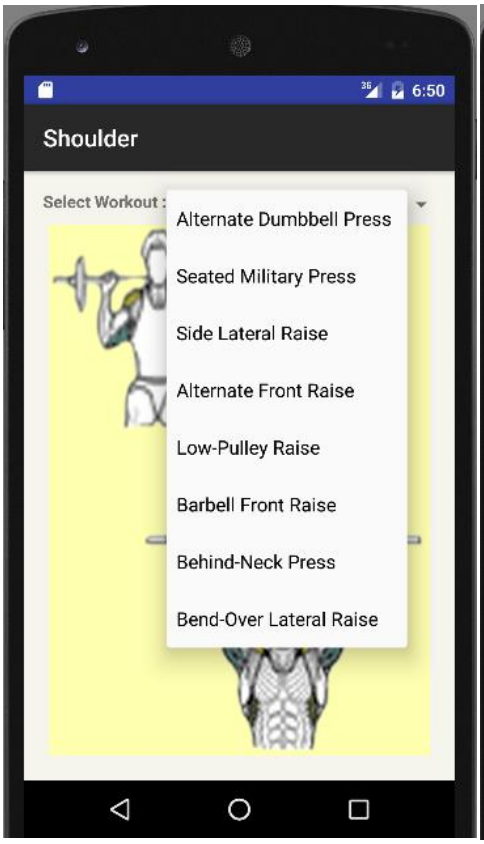

6:46
Abdominal
Select Workout : Crunch

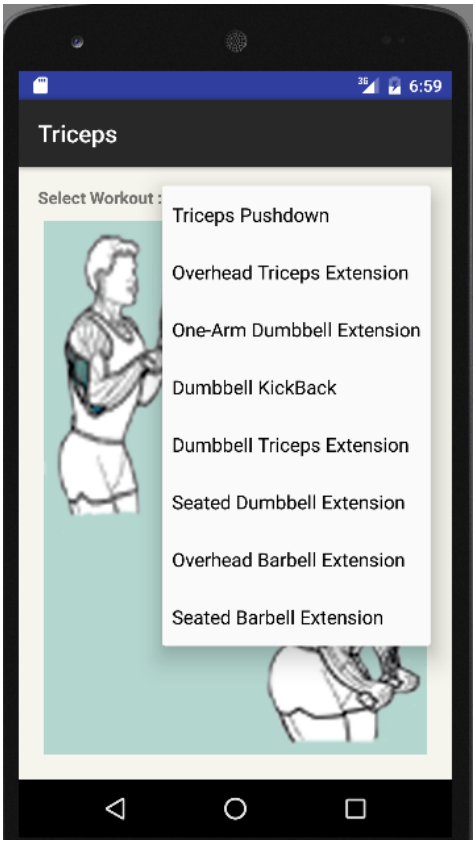

6:47
Abdominal
Select Workout :
Crunch
Raised Leg Crunch
Plank
Raised Leg
Crossover Crunch
Seated Twist
Seated Knee Up
Hanging Leg Raised
Dumbbell Side Bend
Decline Crunch


6:48
Abdominal
Select Workout : Crossover Crunch


6:49
Legs
Select Workout : Squat


6:49
Legs
Select Workout :
Squat
Lunge
Leg Curl
Leg Extension
Seated Toe Raise
Leg Press
Hack Squat
Front Squat


6:50
Shoulder
Select Workout :
Alternate Dumbbell Press
Seated Military Press
Side Lateral Raise
Alternate Front Raise
Low-Pulley Raise
Barbell Front Raise
Behind-Neck Press
Bend-Over Lateral Raise

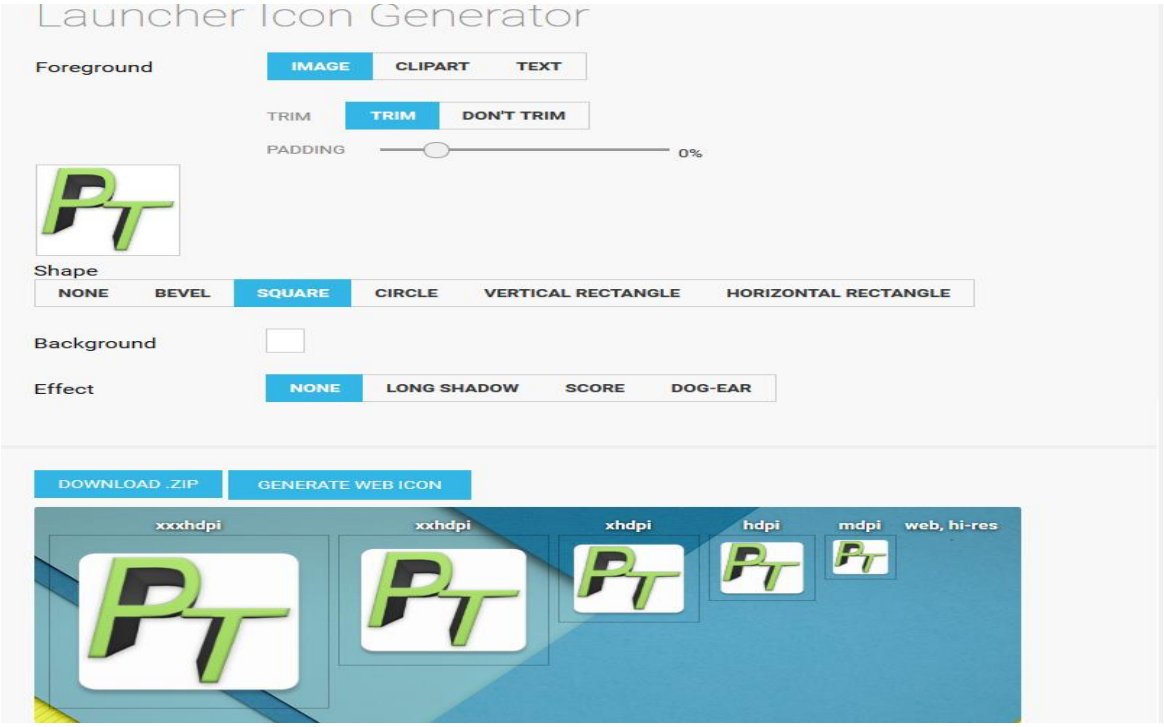

6:59
Triceps
Select Workout :
Triceps Pushdown
Overhead Triceps Extension
One-Arm Dumbbell Extension
Dumbbell KickBack
Dumbbell Triceps Extension
Seated Dumbbell Extension
Overhead Barbell Extension
Seated Barbell Extension

9 Εικόνες και λογότυπο(logo)

Όπως θα έχει γίνει κατανοητό η εφαρμογή έχει αναπτυχθεί έτσι ώστε η κύρια αλληλεπίδραση με τον χρήστη να γίνεται μέσω εικόνων για ευκολότερη απομνημόνευση των ασκήσεων από αυτόν.

Όλες οι εικόνες που χρησιμοποιήθηκαν καθ' όλη την διαδικασία ανάπτυξης του android application δημιουργήθηκαν και επεξεργάστηκαν μέσω της ευρέως γνωστής εφαρμογής **Adobe Photoshop CC 2014**. Οι εικόνες βρίσκονται αποθηκευμένες εσωτερικά της εφαρμογής στον κατάλογο **res/drawable**.

Το **λογότυπο**(**logo**)  με όνομα αρχείου **ic_launcher.png** δημιουργήθηκε εξ ολοκλήρου στο Photoshop και του δόθηκαν διαστάσεις κατάλληλες για κάθε συσκευή android (ανάλυση οθόνης) στον παρόν σύνδεσμο logo



10 Κατέβασμα(Download) της εφαρμογής

Την εφαρμογή μπορεί κανείς να την βρει και να την κατεβάσει από τον παρακάτω σύνδεσμο:

Dropbox link - PersonalTrainer.apk

ή μέσω e-mail στην διεύθυνση lalosapost@gmail.com

Προαπαιτούμενα για την εγκατάσταση είναι ο χρήστης να διαθέτει μια συσκευή με κατώτερη έκδοση λογισμικού Android 3.0 (Honeycomb API 11) η καλύτερη. Μόνο το .apk αρχείο παρέχεται στον αναγνώστη χωρίς τον πηγαίο κώδικα, τον οποίο και επέλεξα να μη τον μοιράσω δημόσια.

11 Σκέψεις για το μέλλον

Μελλοντικά σχέδια αποτελούν:

- Ο εμπλουτισμός ακόμη περισσότερο το γραφικού περιβάλλοντος
- Καλύτερη υποστήριξη για Tablet συσκευές
- Προσθήκη περισσότερων κατηγοριών και ασκήσεων γυμναστικής
- Προσθήκη βίντεο επιπλέον με τις εικόνες

- Δυνατότητα του χρήστη να δημιουργεί και να προσθέτει τις δικές του ασκήσεις στο πρόγραμμα
- Προώθηση και ανέβασμα της εφαρμογής στο Online Google app Store

11 Παρατηρήσεις - Σχόλια

Ολόκληρη η εργασία δημιουργήθηκε εξολοκλήρου σε λειτουργικό περιβάλλον Windows 7 χρησιμοποιώντας το πρόγραμμα Android Studio. Όλες οι δοκιμές και τρεξίματα τις εφαρμογής γίνανε μέσω του Emulator (Προσομοιωτή) που παρέχει το Android Studio καθώς και σε φυσική συσκευή smartphone Motorola Moto G. Η μέγιστη έκδοση λειτουργικού Android έως σήμερα είναι το 6.0 Marshmallow. Ίσως σε μερικές συσκευές Tablets ανάλογα με το μέγεθος της οθόνης να εμφανίζεται μαύρο κενό γύρο από τις εικόνες λόγο του μεγέθους τους 400x500 pixels. Για οποιαδήποτε απορία ή παρατήρηση σχετικά με την εφαρμογή το e-mail επικοινωνίας είναι το lalosapost@gmail.com

12 Βιβλιογραφία

1) Επίσημη ιστοσελίδα κατεβάσματος του Android Studio
   http://developer.android.com/sdk/index.html

2) Επίσημη ιστοσελίδα εισαγωγής στα εργαλεία
   http://developer.android.com/tools/studio/index.html

3) Stackoverflow forums
   http://stackoverflow.com

4) thenewboston ιστότοπος και forums
   https://www.thenewboston.com/forum/category.php?id=10

5) mysamplecode blog page
   http://www.mysamplecode.com

6) androidinterview blog page
   http://www.androidinterview.com

7) color-hex
   http://www.color-hex.com/

8) Δημιουργία λογότυπου(logo)
   https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html