

**ALMA MATER STUDIORUM - UNIVERSITÁ DI BOLOGNA**

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE

INFORMATICHE

## **Workout Manager Android App**

Relazione finale in  
Mobile Web Design

Relatore  
Dott. Mirko Ravaioli

Presentata da  
Lucchi Riccardo

II Sessione  
Anno Accademico 2013-2014



## Ringraziamenti

Prima di cominciare la discussione di questa tesi, desidero ringraziare di cuore tutti coloro che, in un modo o nell'altro, mi hanno aiutato in questo percorso universitario.

Ringrazio innanzitutto il mio professore di Mobile Web Design, nonché relatore di questa tesi, Mirko Ravaioli, per la disponibilità e cortesia da lui mostratami. Probabilmente senza la sua idea questo progetto non esisterebbe nemmeno.

Proseguo a ringraziare tutti i miei amici e colleghi, in particolar modo Carletto, Rez, Mel, Pit, Lucaz e J.I Joe, con cui ho condiviso questi tre anni fantastici e che mi hanno saputo dimostrare (a modo loro, anche se non sempre indolore) quanto bene mi vogliono! Mi viene difficile pensare che siamo già giunti al termine di questo viaggio. Anche se io continuerò a percorrere questa strada per altri due anni, non sarà più lo stesso non vedervi più tutti insieme, ragazzi. Ma non buttiamola troppo sul tragico, tanto lo sapete benissimo che io vi continuerò a tartassare a vita! Dopotutto a che servono gli amici? :)

Un ringraziamento particolare va al collega che mi ha affiancato nello sviluppo di questa tesi, nonché mio grandissimo amico, Carlo Console (lo stesso Carletto menzionato poco fa). Questo povero ragazzo è riuscito a sopportarmi per otto ore al giorno, tutti i giorni, per gli ultimi 10 mesi (eh si, anche durante il periodo di tirocinio!). Non è sicuramente da tutti tener botta a un rompiscatole come me per così tanto tempo!

Scherzi a parte, affrontare in compagnia questo progetto è stata un'esperienza unica. Grazie a questa persona ho potuto capire il vero significato di "lavoro di squadra". Ho condiviso con lui molti momenti di tensione e smarrimento, dai quali non avrei mai pensato di uscirne. Ma alla fine ho capito che se questi momenti vanno affrontati con tanta pazienza e determinazione (e magari con una

buona dose di caffè), la soddisfazione che ne risulta è incomparabile! Quindi voglio rendere un GRAZIE speciale ad un amico SPECIALE.

Un piccolo grande grazie anche tutti quei "signori anonimi" che, rispondendo ad altri utenti su StackOverflow, senza saperlo hanno reso felice anche me :-)

Vorrei infine ringraziare le persone a me più care, i miei genitori.

Grazie mamma, grazie papà, per essermi stati sempre vicini, per avermi incoraggiato e sostenuto nelle mie scelte, per avermi permesso di studiare e di conseguire questa laurea. E' grazie a voi che sono riuscito ad arrivare fino a questo punto, ed è grazie a voi che potrò continuare su questa strada. Non avete mai dubitato di me, e in tutti questi anni non siete mai stati altro che fonte di grandissima soddisfazione. Vi devo davvero tantissimo, e spero che nel prossimo futuro portò in qualche modo sdebitarmi ... anche perché ho già promesso un Porsche Cayenne e una casa in montagna! :-)

Un grazie di cuore anche alla mia fidanzata Terry, che anche lei con la sua infinita pazienza è riuscita a sopportarmi per tutto questo tempo. Riuscirò in qualche modo ad accontentare pure lei prossimamente, magari con una bella casona a Roma... o semplicemente con un carico industriale di vestiti nuovi! (ok forse mi son fatto prendere un po' troppo la mano, già sento il peso di un mutuo addosso!!).

Dato che ancora non posso permettermi tutto questo, l'unica cosa che al momento posso dedicarvi e di cui vi dovrete accontentare, è questa tesi, sommata a tutto il bene che vi voglio e vi ho sempre voluto.

GRAZIE INFINITE.

Dicembre 2014



# Indice

Introduzione	1
Capitolo 1 - Le basi: FitMaster lato Server	4
1.1 - Progettazione ed implementazione del database	4
1.2 – Configurazione del server per l’accesso da remoto	6
1.3 – Progettazione dell’applicazione lato server	8
Capitolo 2 - Progettazione dell’applicazione lato client	10
2.1 – Analisi delle funzionalità	10
2.2 – Implementazione del software	11
2.2.1 – Schermata di login	13
2.2.2 – Navigation Drawer	19
2.2.3 – Sezione Allenamenti	21
2.2.4 – Sezione Statistiche	35
2.2.5 – Sezione Contatta Trainer	37
2.2.6 – Sezione Profilo	38
2.2.7 – Sezione Archivio Esercizi	38
2.2.8 – Sezione Info	40
Capitolo 3 - Conclusioni	40

## Introduzione

L'obiettivo di questa tesi è quello di progettare e sviluppare un sistema che risponda alle esigenze organizzative di una palestra. Prima di entrare nel dettaglio, bisogna effettuare alcune premesse ed esprimere un paio di concetti base per capire al meglio come sono caratterizzati questi centri sportivi.




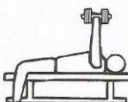

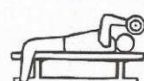
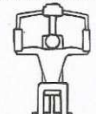
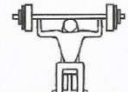







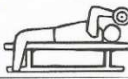




Normalmente le palestre mettono a disposizione ai propri clienti una serie di macchinari e strumenti utilizzati per svolgere attività fisica, e delle persone specializzate (personal trainer) che indichino loro quali esercizi svolgere e in che modo eseguirli. Affinché una persona possa usufruire di tali servizi, essa deve essere necessariamente registrata (pagando una quota mensile o annuale) presso una sede da lei scelta. A registrazione avvenuta, il nuovo cliente viene a contatto con il suo istruttore, il cui scopo è quello di capire le motivazioni che lo hanno spinto a frequentare il centro sportivo. Una volta capiti questi *obiettivi*, è compito del trainer fornire un *allenamento* adeguato a raggiungerli.

Solitamente un *allenamento* è caratterizzato da:

- Uno *scopo* (o l'obiettivo spiegato all'istruttore), che può variare dalla perdita di peso, all'aumento di massa muscolare, o semplicemente al mantenimento dei risultati già ottenuti, a seconda della scelta spiegata dal cliente;
- Una *data di inizio*;
- Una *durata* concordata tra il cliente e il personal trainer, che può variare da un paio di settimane a diversi mesi;
- Una *frequenza*, ossia il numero di giorni a settimana che il cliente vuole dedicare all'attività fisica;
- Una o più *schede*, ossia degli insiemi di esercizi mirati ad allenare una specifica parte del corpo, da eseguire in determinati giorni settimanali.

A prescindere dal numero di schede che compongono l'allenamento, ciascuna di esse è formata da diversi esercizi che vanno eseguiti in un determinato *ordine*. Il personal trainer, oltre a specificare quali esercizi effettuare e come eseguirli, deve anche indicarne il numero di *serie* (quante volte ciascun esercizio deve essere effettuato) e le rispettive *ripetizioni* (numero di movimenti da fare in ciascuna serie). In aggiunta l'istruttore può anche consigliare al cliente il *tempo di recupero* da effettuare tra ciascuna serie e/o il *peso* da utilizzare in determinati esercizi.

Un esempio di quanto detto finora è mostrato nell'immagine seguente:

<b>CENTRO SPORTIVO</b>			
Nome <u>Mario Rossi</u>		Data <u>24/10/14</u>	
Note <u>4 x Sett (2 mesi)</u>		Scheda: <u>A - B - A - B</u>	
<b>A1</b> PANCA ORIZZONTALE (PETTORALI)  SERIE - RIPETIZ. - Kg. <b>3 10 20</b>	<b>A2</b> PANCA INCLINATA (PETTORALI)  SERIE - RIPETIZ. - Kg. <b>4 6 MAX</b>	PANCA DECLINATA (PETTORALI)  SERIE - RIPETIZ. - Kg.	CROCI ORIZZONTALI (PETTORALI)  SERIE - RIPETIZ. - Kg.
CROCI INCLINATE (PETTORALI)  SERIE - RIPETIZ. - Kg.	PULL OVER (PETTORALI)  SERIE - RIPETIZ. - Kg.	PECTORAL MACHINE (PETTORALI)  SERIE - RIPETIZ. - Kg.	LENTO DIETRO (DELTOIDE)  SERIE - RIPETIZ. - Kg.
<b>A3</b> ALZATE LATERALI (DELTOIDE)  SERIE - RIPETIZ. - Kg. <b>3 8 /</b>	<b>A4</b> ALZATE FRONTALI (DELTOIDE)  SERIE - RIPETIZ. - Kg. <b>3 8 5</b>	TRAZIONI AL MENTO (TRAPEZIO)  SERIE - RIPETIZ. - Kg.	<b>B1</b> CURL (BICIPITE)  SERIE - RIPETIZ. - Kg. <b>4 8 10</b>
PANCA SCOTT (BICIPITE)  SERIE - RIPETIZ. - Kg.	<b>B2</b> ALTERNATI (BICIPITI)  SERIE - RIPETIZ. - Kg. <b>3 12 /</b>	CONCENTRATI (BICIPITI)  SERIE - RIPETIZ. - Kg.	<b>B3</b> ESTENSIONI (TRICIPITI)  SERIE - RIPETIZ. - Kg. <b>4 8 MAX</b>
CONCENTRATI (TRICIPITI)  SERIE - RIPETIZ. - Kg.	<b>B4</b> LAT-MACHINE (TRICIPITE)  SERIE - RIPETIZ. - Kg. <b>2 15 15</b>	PARALLELE (TRICIPITE)  SERIE - RIPETIZ. - Kg.	REMATORE (DORSALI)  SERIE - RIPETIZ. - Kg.

[Fig. 1 – Esempio reale di scheda]



Con questo esempio viene mostrato un allenamento per il cliente Mario Rossi, iniziato il 24 ottobre 2014 con *durata* 2 mesi. L'allenamento è formato da due schede da alternare per 4 giorni alla settimana (ad esempio lunedì A – martedì B – giovedì A – venerdì B). Con il colore rosso vengono evidenziati tutti gli esercizi relativi alla scheda A, mirati ad allenare il petto e le spalle, mentre con il colore blu quelli appartenenti alla scheda B, con lo scopo di potenziare i bicipiti ed i tricipiti; i numeri accanto alle lettere indicano l'*ordine* con cui effettuare gli esercizi. Infine in verde sono mostrati i *dettagli* di ciascun esercizio.

Con queste premesse è possibile capire al meglio l'obiettivo di questa tesi. L'idea è quella di costruire un sistema che permetta la gestione delle attività sopra citate in maniera informatizzata. Nello specifico verranno implementati due software, uno su piattaforma Windows indirizzato ai personal trainer che lavorano nelle palestre, e uno per Smartphone Android dedicato ai clienti di questi centri sportivi. Le funzionalità principali del primo prodotto saranno la registrazione di nuovi clienti nel database della palestra e la creazione in formato digitale dei loro allenamenti; l'app Android si occuperà invece di collegarsi al server per scaricare i nuovi allenamenti del cliente ogni qualvolta ve ne sia bisogno. In aggiunta questa applicazione sarà in grado di guidare l'utente durante il suo allenamento attraverso un'interfaccia semplice e intuitiva che gli mostrerà cosa fare e in che modo, monitorando continuamente i progressi da lui compiuti nel tempo.

In questa tesi verrà brevemente riassunto quanto fatto nel progetto lato server affrontato dal mio collega Carlo Console e successivamente si procederà con l'analisi e la progettazione dell'app Android.

# Capitolo 1

## Le basi: FitMaster lato Server

Prima di procedere con lo sviluppo dell'applicazione Android, è bene riassumere quanto affrontato nella tesi "Fitness App per Win/Android - FitMaster", per avere una visione globale sul funzionamento dell'intero progetto. Senza l'ausilio di un server e del suo applicativo sarebbe stato impossibile sviluppare questa tesi.

I principali punti affrontati per costruire una base solida per questo progetto sono stati i seguenti:

- Progettazione ed implementazione del database ospitato dal Server;
- Configurazione del server per l'accesso da remoto;
- Progettazione ed implementazione dell'applicazione lato Server;

Andiamo ora ad analizzare come sono stati realizzati questi punti.

### 1.1 - Progettazione ed implementazione del database

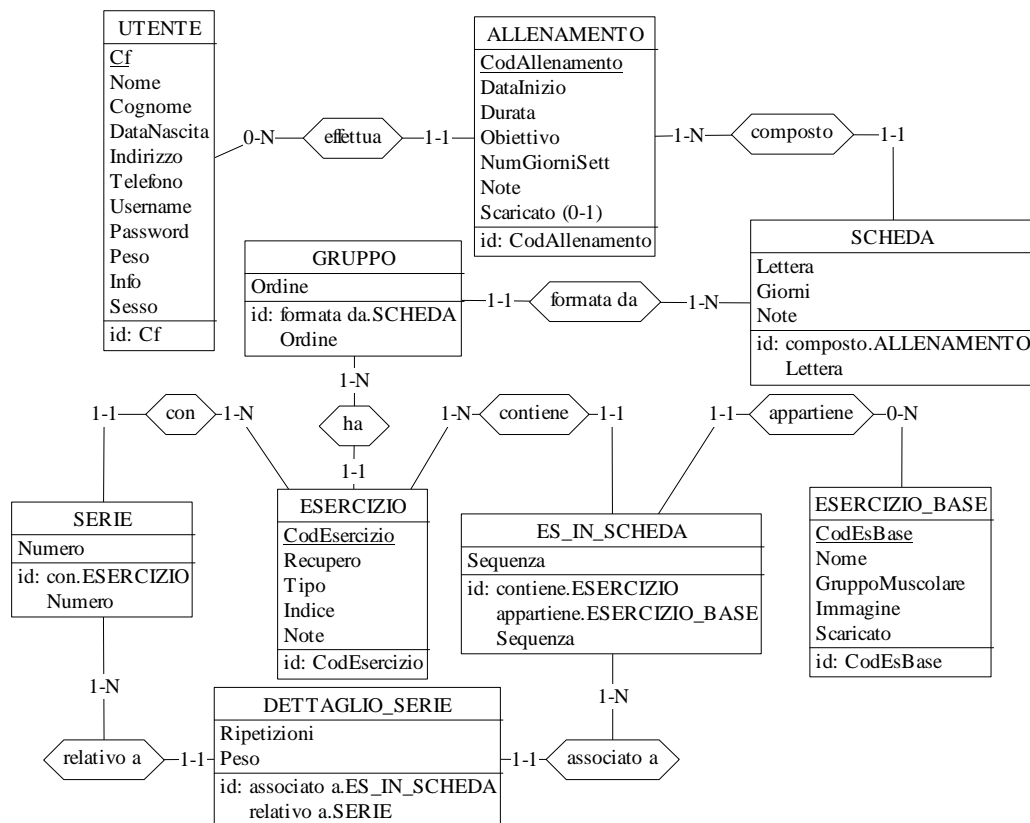
Per realizzare il database è stato innanzitutto necessario analizzare la realtà delle palestre, per cercare di capire quali sono i concetti fondamentali da tenere in considerazione durante la fase di progettazione.

Dall'analisi delle premesse precedentemente effettuate, è stato quindi possibile ricreare una situazione generica, che illustra gli aspetti principali (evidenziati in grassetto) da memorizzare nel database sotto forma di informazioni.

“Un **cliente**, dopo essersi registrato presso una palestra, effettua un colloquio con un personal trainer, a cui spiegherà gli **obiettivi** che intende raggiungere e la **frequenza** con cui può allenarsi. Il compito dell'istruttore è quello di creare un **allenamento** adatto alle esigenze esposte dal nuovo arrivato, consigliando quindi una **durata** per il raggiungimento dei risultati richiesti. Un allenamento può essere formato da una o più **schede** (in questo caso ciascuna scheda è

indirizzata a delle specifiche **parti del corpo** e va eseguita in determinati **giorni della settimana**) di cui ciascuna composta da una serie di **esercizi** che vanno eseguiti in un **ordine** preciso. Il trainer deve fornire inoltre **dettagli** relativi su come eseguire ciascun esercizio, sul **numero di serie e ripetizioni** per ognuno di esso; in aggiunta può consigliare anche il **peso** da utilizzare e specificare un **tempo di recupero** da effettuare tra le varie serie. In alcuni casi capita anche che il trainer suggerisca una o più **alternative** relative ad un esercizio, sarà poi il cliente a decidere quale tra esse svolgere. Un particolare tipo di esercizio che viene spesso utilizzato nelle palestre è la **superserie**, ossia l'esecuzione di più esercizi senza pause tra l'uno e l'altro; il recupero viene effettuato solamente a fine sequenza.”

Analizzando attentamente questo testo, è stato possibile creare uno schema concettuale che rispecchi fedelmente la realtà illustrata (fig.2);



[Fig.2 – Schema E/R finale]

Successivamente, tramite una serie di particolari raffinamenti, le entità e le associazioni sono state trasformate in relazioni, e su esse sono stati definiti dei vincoli di integrità, al fine di impostare la struttura del database. Lo schema concettuale è stato quindi trasformato in schema logico.

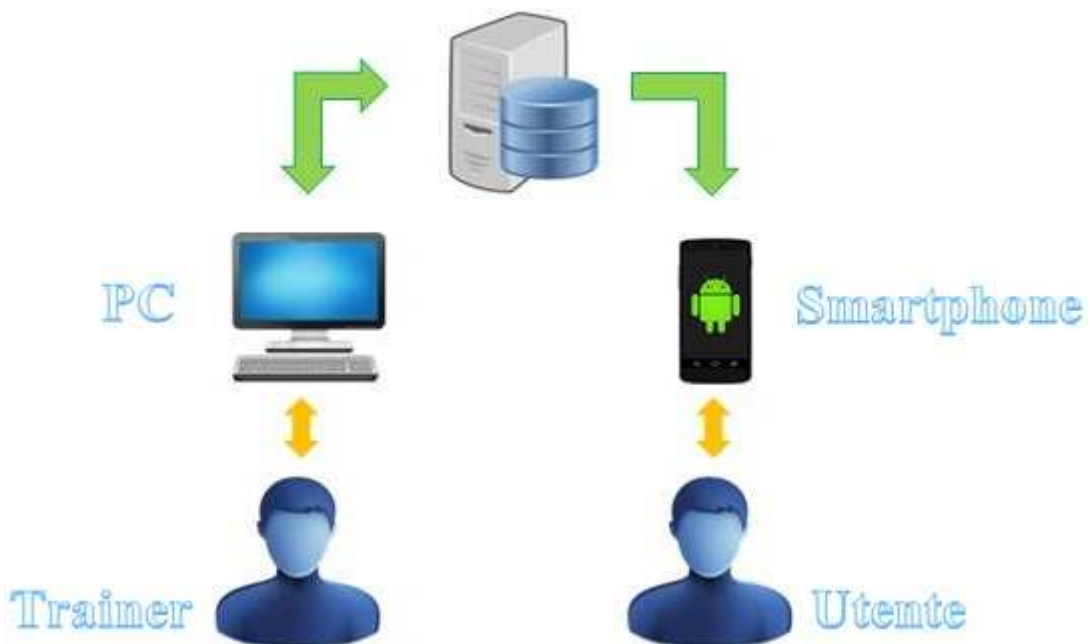
A questo punto si è passati all'implementazione vera e propria del database. È stato scelto di utilizzare Microsoft SQL Server 2012 come DBMS (Database Management System) e di conseguenza il T-SQL (Transact SQL) come linguaggio per l'esecuzione delle query.

Creata l'istanza del database (chiamata "FitMasterDB"), sono state eseguite le opportune query DDL (Data Definition Language) per la creazione delle tabelle e la definizione delle relazioni tra esse.

Dopo aver implementato il database locale, si è passati alla configurazione del server per permetterne l'accesso da remoto.

## 1.2 – Configurazione del server per l'accesso da remoto

Per creare un sistema affidabile ed efficiente è stato deciso di configurare un unico database centralizzato (fig. 3) contenente tutti i dati della palestra, accessibili da remoto da qualsiasi macchina o Smartphone.



[Fig.3 – Rappresentazione del sistema centralizzato]

Con questo sistema i dati presenti sul server saranno accessibili da remoto sia dal personal trainer (tramite l'applicazione Windows, con permessi di

lettura/scrittura), sia dai clienti della palestra (attraverso l'app Android, con permessi di sola lettura).

Per configurare il server centralizzato in modo appropriato è stato necessario:

- Creare le credenziali di accesso per poter accedere al server e attribuire i permessi di lettura/scrittura sui dati (utilizzando Microsoft SQL Server Management Studio);
- Permettere le connessioni remote al server. Per far ciò è stato abilitato 'SQL Server Browser', servizio che si interpone tra il server e il client, il quale resta in attesa delle richieste in arrivo di risorse Microsoft SQL Server, e fornisce la connessione all'istanza del server corretta. È stato inoltre abilitato l'ascolto del server sulla porta TCP 1433 (porta di default del servizio SQL Server) tramite 'SQL Server Configuration Manager';
- Creare le regole su Windows Firewall, in modo da consentire il passaggio di tutto il traffico TCP con porta di destinazione 1433. Per fare ciò sono state create due regole, una per consentire il passaggio dei dati attraverso la porta sopra menzionata, e l'altra per far funzionare correttamente il servizio SQL Server Browser;
- Creare la regola sul firewall del router, per permettere l'indirizzamento del traffico proveniente dall'esterno verso l'IP del server (operazione chiamata 'Port Forwarding'). Questa regola è caratterizzata dall'IP interno (IP del server) sul quale si vogliono indirizzare i dati, dalla porta interna (ossia la porta su cui il server è in ascolto, che in questo caso è la 1433) e dalla porta esterna (ossia la porta sulla quale arrivano le richieste dall'esterno, che è sempre la 1433). In altre parole, attraverso l'utilizzo di questa regola, tutto il traffico in arrivo sulla porta esterna 1433 viene reindirizzato verso la porta 1433 del server. Questa regola è stata aggiunta tramite web browser dalla sezione 'Port Mapping' del router.

Con queste quattro operazioni è stato possibile rendere il server centralizzato perfettamente raggiungibile dall'esterno.

Come ultimo passo è stato progettato e implementato l'applicativo lato server, riservato al personale della palestra.

### **1.3 – Progettazione dell'applicazione lato server**

Prima di implementare l'applicazione vera e propria, sono state analizzate le funzioni che essa deve mettere a disposizione ai personal trainer; ne sono emerse quindi le seguenti caratteristiche:

- Gestione dei clienti, che comprende l'aggiunta (nel momento della registrazione), eventuali modifiche di alcuni campi e l'eliminazione (nel caso in cui un utente smetta di frequentare la palestra);
- Creazione e gestione degli allenamenti per i clienti nella maniera più accurata possibile (con la possibilità quindi di utilizzare schede, esercizi, esercizi alternativi, superserie, ...);
- Organizzazione dell'archivio esercizi (con possibilità quindi di aggiunta e rimozione), utilizzato nella fase di creazione degli allenamenti.

Per lo sviluppo di questo software si è deciso di utilizzare Microsoft Visual Studio 2013 Premium come ambiente di sviluppo, Visual C# come linguaggio di programmazione e Windows Form come User Interface.

Una volta definite le principali caratteristiche dell'applicazione, si è passati alla stesura del codice: si è deciso di strutturare il programma su tre sezioni principali, per organizzare al meglio le funzionalità sopra citate e rendere quindi l'utilizzo del software il più semplice possibile.

Le sezioni implementate sono:

- “Clienti”, suddivisa in due Tab, in cui è possibile registrare o modificare/eliminare un cliente (rispettivamente tramite le Tab “Registrazione” e “Gestione”);
- “Allenamento”, anch'essa suddivisa in due Tab, in cui è possibile creare un nuovo allenamento da attribuire ad un cliente (Tab “Nuovo”), oppure visualizzarne i dettagli di uno già esistente (Tab “Gestione”). La creazione

di un allenamento è stata implementata in modo tale da renderla il più simile possibile a come avviene nella realtà, quindi con la possibilità di utilizzare schede, esercizi alternativi e superserie, come mostrato in (fig. 4). È inoltre possibile specificare i dettagli per ciascun esercizio, quali serie, ripetizioni, peso, tempo di recupero ed eventuali note; si ricordi che per l'inserimento di serie e ripetizioni sono supportate sia la formattazione "Nr. serie x Nr. ripetizioni", utilizzata per specificare serie tutte uguali in un esercizio, sia "Nr. rip. – Nr. rip. -...", per poter definire delle serie che differiscono per numero di ripetizioni, come accade per le "serie piramidali".

- "Esercizi", in cui è possibile organizzare (inserendo o eliminando dalla tabella 'Esercizio Base' del database remoto) gli esercizi messi a disposizione dalla palestra.



[Fig.4 – FitMaster Windows, creazione di una nuova scheda]

Arrivati a questo punto ci ritroviamo in possesso di un server centralizzato e di un'applicazione lato server in grado di interagire con esso; è possibile procedere ora con la progettazione ed implementazione dell'app client side (rivolta al sistema operativo Android).

## Capitolo 2

### Progettazione dell'applicazione lato client

In questo capitolo verrà analizzata la parte progettuale e implementativa dell'applicazione lato client per Android, destinata ai clienti delle palestre.

L'obiettivo fondamentale di questa tesi è quello di creare un software in grado di recuperare e mostrare all'utente le informazioni a lui dedicate (principalmente i suoi allenamenti), inserite nel database centralizzato tramite l'utilizzo dell'applicazione Windows.

Prima di procedere con la stesura del codice, è necessario effettuare un'accurata analisi sulle funzionalità che l'app deve fornire.

#### 2.1 – Analisi delle funzionalità

La funzionalità principale di cui l'app dovrà esser dotata è sicuramente la possibilità di poter scaricare gli allenamenti di un cliente sul proprio Smartphone (ogni qualvolta ve ne siano di nuovi) e di visualizzarli con un layout grafico che ne ricordi la struttura di un allenamento cartaceo. Inoltre, effettuando uno studio su ciò che potrebbe essere più utile ai clienti di una palestra, sono emerse le seguenti peculiarità:

- Possibilità di poter utilizzare l'applicazione sia connettendosi al server, sia rimanendo offline;
- Possibilità di tener traccia dei progressi effettuati durante il periodo di allenamento;
- Disponibilità di alcuni strumenti utili allo svolgimento degli esercizi (come ad esempio un timer per impostare il tempo di recupero tra una serie e l'altra);
- Possibilità di consultare l'archivio di tutti gli esercizi che la palestra mette a disposizione, per poter richiedere eventuali varianti a quelli effettuati di frequente;



- Possibilità di contattare il trainer in qualunque momento, anche quando non si è presenti in palestra.

Così come l'applicazione lato server è stata munita di una guida all'utilizzo del software, anche l'app mobile verrà dotata di un tutorial che spieghi passo a passo tutte le funzionalità messe a disposizione; verrà infine creata una sezione nella quale sarà possibile visualizzare le informazioni sul prodotto.

## **2.2 – Implementazione del software**

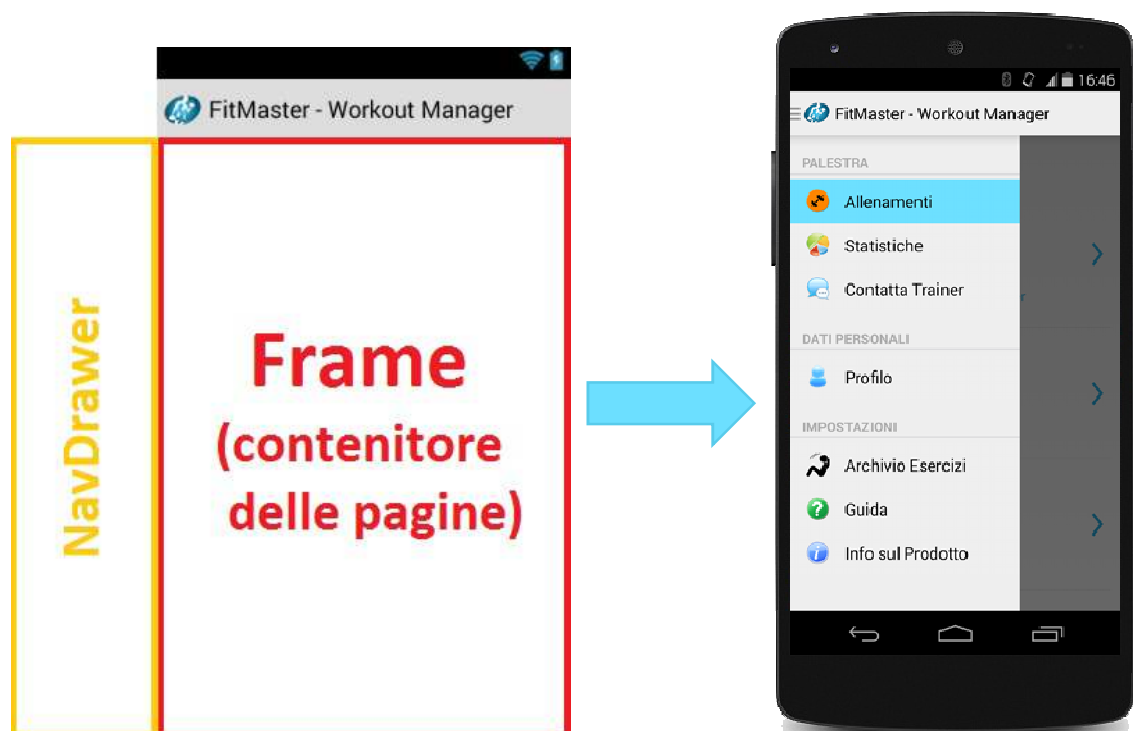
Per lo sviluppo di quest'app si è deciso di utilizzare Eclipse Luna come ambiente di sviluppo e Java come linguaggio di programmazione.

Con le premesse fatte nel paragrafo precedente è possibile incominciare ad abbozzare un'interfaccia grafica per l'applicazione. Si è pensato quindi di visualizzare, all'avvio del software, una pagina dalla quale è possibile scegliere di utilizzare l'app in modo "connesso" (con la richiesta di un login) oppure "offline": la prima modalità consente di connettersi al server remoto per poter scaricare eventuali nuovi allenamenti per l'utente, mentre la seconda permette di utilizzare l'applicazione senza la presenza di una connessione dati attiva; in ciascun caso comunque, l'applicazione si interfacerà sempre al database locale per massimizzare le prestazioni. Effettuato l'accesso, viene caricato il corpo principale dell'applicazione (MainActivity), costituito da un DrawerLayout contenente un menu laterale (ListView strutturata su tre sezioni) e da un "contenitore di pagine" (Frame): ogni volta che viene selezionata una voce dal menu, viene caricata la rispettiva schermata (Fragment) nel contenitore. Le voci che caratterizzano il menu laterale sono:

- Sezione Palestra
  - 'Allenamenti', che permette la visualizzazione di tutti gli allenamenti del cliente che ha effettuato l'accesso; selezionando un allenamento ne verranno mostrati i relativi dettagli;

- ‘Statistiche’, che permette di visualizzare i progressi effettuati nel periodo di allenamento tramite dei grafici;
  - ‘Contatta Trainer’, tramite la quale è possibile comunicare con il trainer via mail;
- Sezione Dati Personali
- ‘Profilo’, dalla quale si accede ad una sezione contenente le informazioni personali;
- Sezione Impostazioni
- ‘Archivio esercizi’, dalla quale è possibile accedere a tutti gli esercizi messi a disposizione dalla palestra;
  - ‘Guida’, che comprende un tutorial alle funzioni offerte dall’app;
  - ‘Info sul prodotto’, dalla quale è possibile ottenere informazioni sul software utilizzato.

Di seguito viene mostrata la struttura della MainActivity e come viene visualizzata una volta installata l’app (fig. 5).



[Fig. 5 – Struttura e visualizzazione della MainActivity]

Detto questo, andremo ora ad analizzare ciascuna delle sezioni principali che compongono l’applicazione, partendo dalla pagina di login.

## 2.2.1 – Schermata di login

Come prima pagina caricata all'avvio dell'applicazione viene mostrata la schermata di login (il cui codice è contenuto in 'LoginFragment.java'), caratterizzata dai classici spazi per l'inserimento di username e password e dal pulsante 'Accedi', che permette l'utilizzo dell'app in modalità "connessa". Inoltre l'interfaccia (fig. 6) è stata dotata di una "CheckBox" utilizzata per ricordare le credenziali di accesso e di una "TextView", che alla pressione permette l'esecuzione del software in modalità "offline".



Prima di procedere con la connessione al server, è indispensabile implementare un database locale nel quale memorizzare i dati recuperati da remoto. Siccome Microsoft SQL Server non è supportato sul sistema operativo Android, è necessario ricostruire l'intera struttura del database utilizzando un DBMS compatibile con l'ambiente. Si è deciso quindi di utilizzare SQLite, una libreria software scritta in C che implementa un DBMS SQL in grado di creare una base di dati (comprese tabelle, query, ...) incorporata in un unico file.

[Fig.6 – Schermata di login]

Per poter creare il database locale e potervi accedere da ogni sezione del programma, è stato creato un file Java chiamato 'FitMasterDB' definito da:

- Nome del file contenente la base di dati: 'FitMasterDB.db'. Questo file è salvato nella cartella dell'app (/data/data/it.unibo.fitmaster/databases/);
- Versione del database;
- Evento "onCreate", richiamato quando viene istanziato il database all'avvio dell'app, in cui viene controllato se il database fisico esiste: in

caso positivo viene restituito il riferimento alla base di dati, altrimenti vengono eseguite le query DDL (Data Definition Language) necessarie alla definizione della sua struttura e viene creato il file 'FitMasterDB.db';

- Metodi per l'inserimento di dati nelle varie tabelle (un metodo per ciascuna tabella della base di dati);
- Metodi utilizzati per il recupero di informazioni (ciascuno dei quali è caratterizzato da almeno una query di selezione).

Come detto precedentemente, il database (variabile da noi chiamata fitDB) viene istanziato nella MainActivity all'avvio dell'app; per potervi accedere da ogni parte dell'applicazione è necessario richiamare la seguente istruzione:

`“MainActivity.fitDB.getReadableDatabase()”`

in cui MainActivity.fitDB indica il riferimento alla variabile di tipo SQLiteDatabase istanziata nell'Activity principale e getReadableDatabase() è il metodo con cui si specifica il tipo di accesso al database (in questo caso di sola lettura).

Tornando alla sezione di login, vediamo ora come viene effettuata la connessione al database remoto. È necessario quindi:

- Includere nel progetto il file Jar “jtds-1.2.7” che contiene i driver JDBC, utilizzati per permettere la comunicazione tra Microsoft SQL Server e la nostra app. JDBC (Java Data Base Connectivity) è di fatto un connettore per database, che consente l'accesso e la gestione dei dati memorizzati sulle basi di dati da qualsiasi programma scritto in Java, indipendentemente dal tipo di DBMS utilizzato;
- Creare un file Java che permetta l'utilizzo della stessa connessione in ciascuna sezione del programma, così come è stato fatto per il database locale.
- Dichiarare i permessi di accesso ad Internet nel file 'AndroidManifest.xml' attraverso la seguente riga di codice:  
`<uses-permission android:name="android.permission.INTERNET" />`

Questa classe è stata chiamata ‘SQLConnection.java’ ed è definita da una variabile di tipo Connection (utilizzata per memorizzare i parametri della connessione al server), dal metodo “setConnection” (che inizializza e prova a stabilire tale connessione, sfruttando i driver precedentemente inclusi) ed infine dal metodo “getConnection”, che viene utilizzato per recuperare l’istanza della connessione, quando necessario,.

Di seguito viene mostrato il codice che permette la connessione al server remoto richiamato all’interno del metodo “setConnection”:

```
1. // Metodo per effettuare la connessione al database SQL Server
2. // I parametri passati sono quelli inizializzati dal metodo 'setConnection'
3. private Connection connetti(String server, String db, String user, String pass)
4. {
5.     StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
6.     StrictMode.setThreadPolicy(policy);
7.     Connection conn = null;
8.     String ConnString = null;
9.     try
10.    {
11.        // Viene registrato il driver da utilizzare per stabilire la connessione con un DB 'SQL Server'
12.        Class.forName("net.sourceforge.jtds.jdbc.Driver");
13.        ConnString = "jdbc:jtds:sqlserver://" + server + ";" + "databaseName=" + db + ";user="
14.            + user + ";password=" + pass + ";socketKeepAlive=true";
15.        // Viene effettuata la connessione
16.        conn = DriverManager.getConnection(ConnString);
17.    }
18.    catch (Exception e)
19.    { e.printStackTrace(); }
20.    // Viene restituita la connessione con il database remoto
21.    return conn;
22. }
```

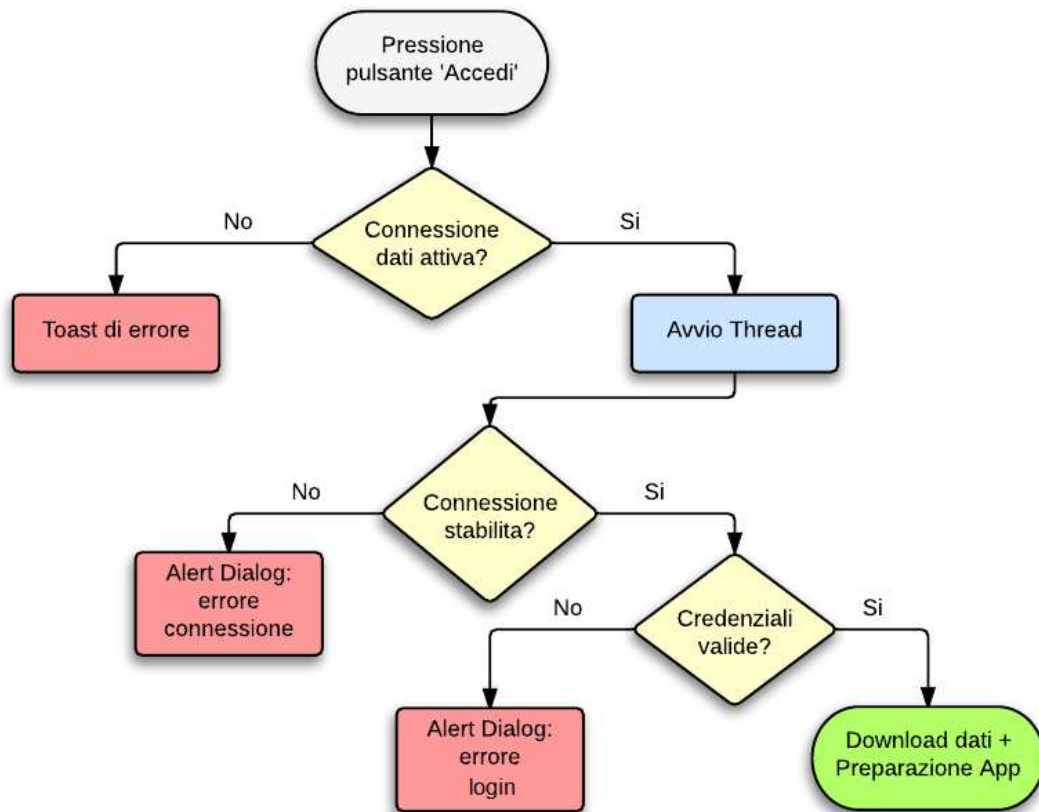
Prima di proseguire con la stesura del codice, è opportuno introdurre il tipo di dato “SharedPreferences”, poiché verrà spesso utilizzato nel progetto. La classe SharedPreferences consente di salvare delle coppie chiave-valore (di tipi di dato primitivi) in modo persistente e permette di recuperarle nei successivi avvisi dell’app; in questo modo è possibile accedere a delle informazioni anche dopo la chiusura del software.

Una volta predisposte le classi per l’accesso al database locale e alla connessione in remoto, si procede con l’implementazione della parte funzionale della schermata di login. Partiamo con l’analisi del pulsante ‘Accedi’. Alla

pressione di questo tasto, viene innanzitutto controllato se vi è una connessione dati attiva (Wifi o 3g): in caso negativo viene mostrato un “Toast” che informa l’utente di doverne attivare una, altrimenti, in caso positivo, viene avviato un Thread, il cui scopo è quello di effettuare la connessione al server SQL. Se questa non viene stabilita, verrà visualizzato un messaggio di errore tramite un AlertDialog, altrimenti si procederà con la fase di autenticazione. Nel caso in cui le credenziali siano errate, verrà visualizzato a video un messaggio di errore di login. Se invece le credenziali risultano corrette, verrà eseguito il seguente set di istruzioni:

1. Viene richiamato il metodo “scaricaDati”, che permette di scaricare tutti i dati relativi al cliente che ha effettuato l’accesso; durante questa fase viene mostrato uno “Spinner” che scomparirà da solo a download terminato;
2. Viene creata, qualora non esista già, una SharedPreferences per memorizzare l’username dell’ultimo utente che ha effettuato l’accesso. Questa preference viene utilizzata per compilare automaticamente i campi username e password qualora si scelga l’opzione di memorizzazione della password;
3. Viene creata, qualora non esista già, una SharedPreferences associata all’utente appena connesso. Essa viene utilizzata all’interno del metodo “scaricaDati”, ma verrà spiegata successivamente;
4. Viene impostato il valore (booleano) della SharedPreferences relativa al primo utilizzo dell’applicazione. Anche questa scelta verrà chiarita successivamente;
5. Viene caricata la sezione principale dell’applicazione, inizializzando il menu laterale e caricando la sezione degli allenamenti.

Quanto detto viene riassunto utilizzando il seguente flow chart (fig.7).



[Fig.7 – Diagramma di flusso delle operazioni presenti nel pulsante ‘Accedi’]

Soffermiamoci un momento sul metodo ‘scaricaDati’ sopra citato, in quanto svolge un ruolo molto importante per il funzionamento dell’intera applicazione. Come menzionato precedentemente, ogni volta che viene effettuato un login, viene creata una “SharedPreference” associata all’utente connesso: in questo modo è possibile sapere quali utenti hanno effettuato l’accesso almeno una volta e quali no. Questa SharedPreference, combinata con quella relativa al primo utilizzo dell’app, consente di gestire al meglio il download dei dati sullo Smartphone. Con esse è infatti possibile gestire tutti i casi in cui si può trovare l’applicazione:

- Primo utilizzo (nessun utente ha mai effettuato l’accesso): vengono scaricati tutti i dati relativi al cliente connesso (dati personali e allenamenti inclusi) e tutti gli esercizi della tabella “Esercizio Base” presente nel server remoto;

- Primo login per l'utente (ma qualcun altro ha già effettuato l'accesso precedentemente): vengono scaricati tutti i dati ad esso associati (compreso l'utente stesso), ma vengono recuperati solamente gli esercizi base il cui campo "Scaricato" è impostato a 'false'. Questa azione viene svolta per evitare situazioni di ridondanza dati nella tabella "Esercizio Base" del database locale;
- L'utente ha effettuato almeno una volta l'accesso: vengono scaricati solo gli allenamenti/esercizi base con campo "Scaricato" impostato a 'false'. In altre parole vengono scaricati solamente i dati che non sono ancora presenti nel database dello Smartphone.

In ciascuno dei precedenti casi, dopo aver scaricato un allenamento/esercizio base dal server, viene impostato il corrispondente campo "Scaricato" a "true", per indicare che non bisognerà più recuperare quel dato nei successivi login, poiché già presente sul device.

Per quanto riguarda la funzionalità principale del metodo "scaricaDati", una volta definiti gli elementi da scaricare, si procede con un download "a cascata" dei dati, a partire dalla tabella 'Allenamento' fino ad arrivare alla tabella 'Dettaglio Serie' del server remoto.

Quanto detto finora riguardava l'utilizzo dell'app in modalità "connessa", ma come accennato precedentemente, il software può essere utilizzato anche senza una connessione dati attiva. Per poter utilizzare questa modalità è necessario che un utente abbia effettuato il login almeno una volta: alla pressione della TextView 'Salta il login e rimani offline', il sistema caricherà automaticamente i dati dell'ultimo utente connesso. Qualora si provi ad utilizzare l'applicazione in modalità "offline" senza aver mai effettuato un accesso, il sistema mostrerà a video un messaggio di errore tramite AlertDialog.



Per concludere, la schermata di login offre la possibilità di memorizzare le credenziali di accesso: se viene spuntato il relativo CheckBox, agli avvii successivi dell'app verranno caricati automaticamente username e password dell'ultimo utente connesso, recuperati tramite SharedPreferences.

Terminata l'implementazione di questa pagina, possiamo procedere con la progettazione del corpo principale dell'applicazione. Per prima cosa verrà analizzato il menu laterale e successivamente verranno affrontate le schermate relative alle singole voci che lo compongono.

### **2.2.2 – Navigation Drawer**

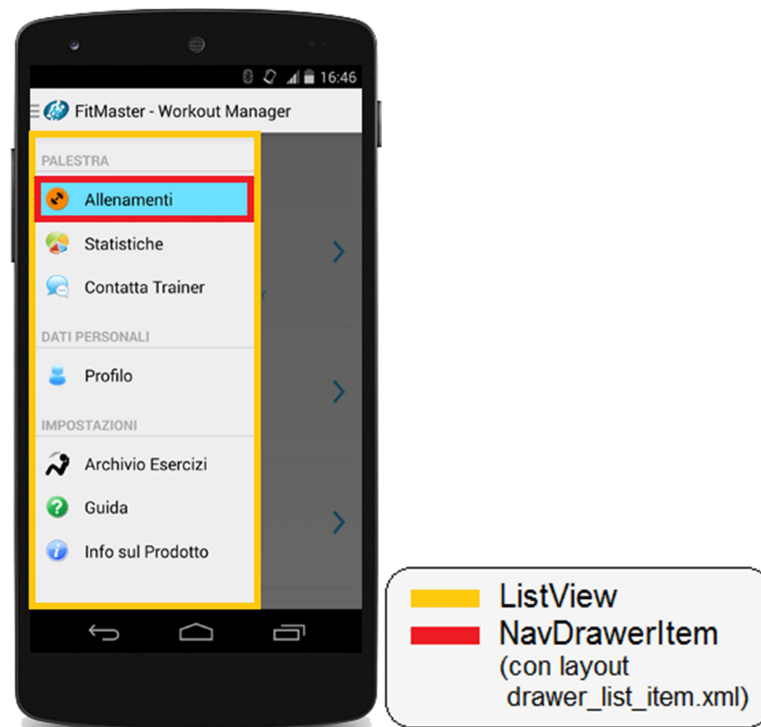
La prima azione svolta subito dopo la fase di login è l'inizializzazione del menu laterale, comunemente chiamato NavigationDrawer. Come anticipato nel capitolo precedente, la MainActivity è formata da un DrawerLayout contenente un Frame e una ListView. Quest'ultima contiene una serie di voci, tramite le quali l'utente è in grado di muoversi all'interno dell'app. È stato deciso di utilizzare un DrawerLayout in quanto consente di mostrare o nascondere il menu laterale tramite "swype" laterale (oppure attraverso la pressione del pulsante dell'ActionBar).

Ciascun elemento del menu laterale è stato pensato come oggetto composto da una piccola immagine e da un testo (fig. 8). Per poter associare un tipo di dato personalizzato ad una ListView, è necessario implementare un Adapter che ne gestisca la rappresentazione grafica. Per implementare il NavDrawer sono quindi necessari i seguenti componenti:

- Un file xml (drawer\_list\_item.xml) che descrive il layout grafico di ciascuna delle voci che compongono il menu laterale;
- Una classe Java (NavDrawerItem.java) che rappresenta il tipo di dato personalizzato (tramite proprietà e metodi) di cui è composta la lista;
- Un file Java (NavDrawerListAdapter) per adattare gli elementi di tipo NavDrawerItem alla ListView.

Per prima cosa è necessario inizializzare un vettore di “NavDrawerItem” con tutte le voci che andranno a comporre il menu laterale (i testi e le icone che li caratterizzano vengono recuperati dal file strings.xml). Successivamente bisogna dichiarare un Adapter che faccia riferimento all’array appena creato e impostarlo sulla ListView della MainActivity: in questo modo, la lista verrà riempita con tanti elementi (ciascuno con layout drawer\_list\_item.xml) quante sono le voci del vettore.

A questo punto rimane solamente da gestire l’evento che permette la navigazione tra le varie sezioni dell’app. Per far ciò viene impostato alla ListView un “listener” in grado di rilevare la pressione dei suoi elementi. Alla selezione di una voce, il sistema caricherà nel frame della MainActivity il Fragment corrispondente, aggiornando inoltre il titolo presente nella ActionBar con il nome della sezione scelta.

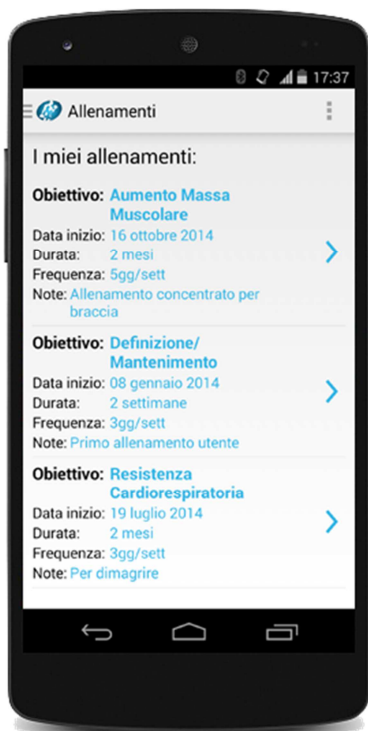


[Fig.8 – Struttura del NavigationDrawer]

Una volta impostato il NavigationDrawer, viene caricata la pagina relativa alla prima voce del menu laterale, ossia la sezione ‘Allenamenti’. Questa schermata sarà oggetto di analisi nel paragrafo successivo.

### 2.2.3 – Sezione Allenamenti

Si dedicherà particolare attenzione alla sezione ‘Allenamenti’, in quanto permette all’utente di visualizzare tutti i dettagli degli allenamenti che gli sono stati assegnati dal trainer. Dal momento che vi sono molte informazioni da visualizzare, per mantenere un layout grafico semplice e pulito è stato deciso di strutturare la sezione su più pagine. La prima pagina che viene visualizzata alla pressione della voce ‘Allenamenti’ della NavigationDrawer, è quella contenente l’elenco di tutti gli allenamenti dell’utente che ha effettuato l’accesso (fig. 9).



[Fig.9 – Sezione allenamenti]

Il Fragment in questione (il cui codice è contenuto in “WorkoutFragment.java”) è basato su una ScrollView (oggetto che permette di scorrere verticalmente la finestra) contenente una TextView e una ListView; come si può intuire, quest’ultima ha il compito di visualizzare la lista degli allenamenti disponibili. Per poter implementare questa lista, è stato seguito lo stesso procedimento adottato per la creazione del menu laterale: è stata definita la classe e il layout grafico di ciascun elemento che la compone, e l’adattatore che ne permette l’inserimento.

È stato infine gestito l’evento “onClick” sugli elementi della lista: alla pressione di ciascuno di essi, il sistema caricherà nel Frame della MainActivity un nuovo Fragment (chiamato “WorkoutSchedeFragment”) che si occupa di visualizzare tutte le schede associate all’allenamento selezionato, ciascuna con i relativi esercizi.

Per poter accedere a tutte le schede e ai relativi contenuti dalla stessa pagina, è stato deciso di strutturare l'interfaccia su più "Tab" (una per ciascuna scheda, tramite l'utilizzo dell'oggetto "FragmentTabHost"), ognuna delle quali formata da un contenitore (Frame). Quest'ultimo viene utilizzato per caricare il Fragment ("SchedaFragment.java") dedicato alla visualizzazione di tutti gli esercizi che compongono la scheda.



Ricapitolando: selezionando un allenamento dalla lista presente nella pagina "WorkoutFragment", viene caricato il Fragment "WorkoutSchedeFragment", formato da tante "Tab" quante sono le schede dell'allenamento selezionato (fig.10). Ciascuna Tab è formata da un Frame che ospita il Fragment "SchedaFragment", che viene utilizzato per visualizzare tutte le informazioni della scheda selezionata.

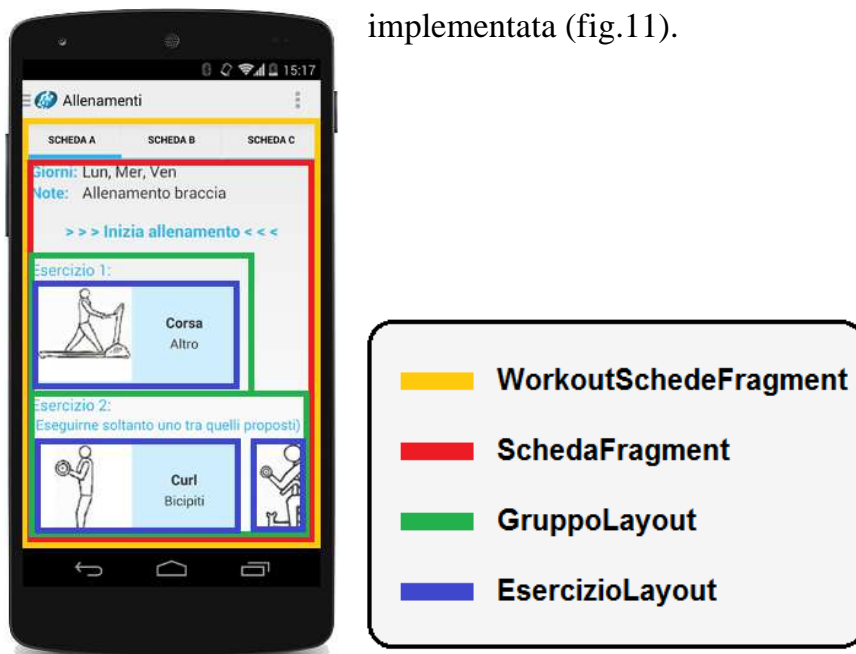
[Fig.10 – Schermata di visualizzazione di un allenamento]

Andiamo ora ad analizzare come viene impostata la struttura di ciascuna scheda. Per ognuna di esse si vogliono visualizzare gli attributi che la caratterizzano (giorni e note), un pulsante che permetta di iniziare l'allenamento selezionato e una lista verticale che mostri tutti gli esercizi che la compongono (nel caso in cui vi siano degli esercizi alternativi, essi devono essere affiancati orizzontalmente all'esercizio a cui fanno riferimento).

Per l'implementazione di questo Fragment è stato utilizzato come oggetto contenitore una "CustomScrollView": questa classe, che deriva dalla "ScrollView", è stata da noi personalizzata per ottimizzare al meglio la fluidità della pagina. Senza di essa infatti, potrebbero verificarsi dei conflitti tra lo scroll

verticale (utilizzato per visualizzare gli esercizi) e lo scroll orizzontale (utilizzato per scorrere gli eventuali esercizi alternativi).

Al contrario di quanto svolto finora, non è più sufficiente l'utilizzo di una ListView per implementare la lista degli esercizi, poiché ciascun elemento di essa dovrebbe essere a sua volta una lista orizzontale (azione sconsigliata persino da Google). Per aggirare questo problema, si è deciso di creare sia la lista verticale che quella orizzontale in modo dinamico. Al caricamento del Fragment, oltre ad esser inizializzati i campi 'giorni' e 'note' con i valori relativi alla scheda selezionata, vengono creati tanti oggetti di tipo "GruppoLayout" quanti sono i gruppi associati a quella scheda. Questa classe è caratterizzata principalmente da una TextView (che indica il numero dell'esercizio) e da una HorizontalScrollView. Ogni volta che viene istanziato un oggetto di questo tipo, vengono inseriti all'interno della sua HorizontalScrollView, tanti elementi di tipo "EsercizioLayout" quanti sono gli esercizi ad esso associati. Anche EsercizioLayout, come GruppoLayout, è un tipo di dato la cui struttura grafica viene generata automaticamente da codice (non dispone quindi di un file xml che ne descrive il suo layout). Essa è definita principalmente da un'immagine e due TextView, che contengono rispettivamente i valori degli attributi dell'esercizio base da svolgere. La seguente immagine riassume la struttura appena implementata (fig.11).

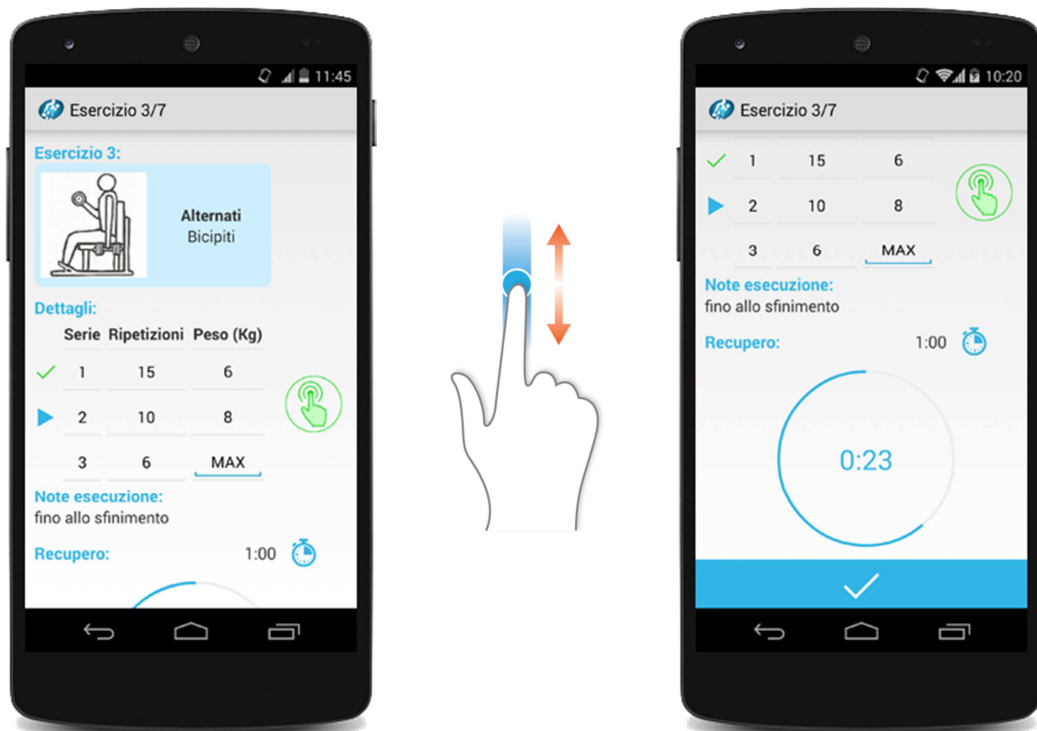


[Fig.11 – Struttura adottata nella sezione Scheda]

A questo punto ci ritroviamo con una schermata che mostra l'elenco degli esercizi di ciascuna scheda relativa all'allenamento selezionato, ma è necessario poterne visualizzare anche i dettagli. Per questo motivo si è pensato di far aprire, alla pressione di ciascun esercizio, una finestra che mostri tutte le informazioni relative ad esso. Inoltre, se si è deciso di iniziare l'allenamento (tramite la pressione del pulsante 'Inizia allenamento'), questa finestra dovrà permettere all'utente di indicare al sistema quali serie ha svolto durante l'allenamento, per poter costruire un grafico che rappresenti i progressi effettuati nel tempo; se invece l'allenamento non è stato iniziato, la nuova finestra deve solamente mostrare i dettagli relativi all'esercizio selezionato.

Per poter aprire questa nuova pagina, è necessario quindi impostare a ciascun `EsercizioLayout` l'evento "onClick". Si ricordi che l'elenco degli esercizi può essere formato sia da 'esercizi normali' che da 'superserie', quindi è necessaria l'implementazione di un'interfaccia grafica per entrambe le categorie. Sarà poi il sistema a gestire quale Activity caricare, in base all'esercizio selezionato.

Partiamo analizzando l'Activity relativa agli esercizi normali, da noi chiamata "EsNormaleActivity", mostrata nella pagina seguente (fig.12). Questa finestra è particolarmente importante in quanto deve mostrare all'utente i dettagli su come eseguire l'esercizio selezionato, indicandone serie, ripetizioni, peso, eventuali note e tempo di recupero. È stato pensato inoltre di dotare l'interfaccia di un pratico 'Timer' che permette all'utente di effettuare la pausa specificata.



[Fig.12 – Interfaccia dell’Activity EsNormale]

Per via dei molti elementi che devono essere visualizzati, si è deciso di basare la struttura dell’intera pagina su una ScrollView. All’inizializzazione di questa Activity, viene prima di tutto impostato il titolo presente nell’ActionBar con una stringa che ricorda all’utente l’esercizio che sta eseguendo; per esempio, se la scheda è composta da nove esercizi e l’utente sta eseguendo il terzo, viene impostato come titolo ‘Esercizio 3/9’.

Il primo oggetto istanziato all’interno della ScrollView è un ‘EsercizioLayout’ contenente le informazioni relative all’esercizio base da eseguire; in questo caso i dati non sono stati recuperati dal database locale, ma sono stati passati via ‘Bundle’ dall’Activity principale.

Subito dopo la rappresentazione dell’esercizio viene mostrata una tabella che, per ciascuna serie, ne indica ripetizioni e peso. Per il recupero di tali informazioni è necessario interrogare il database locale tramite una query di selezione sulla tabella ‘Dettaglio Serie’, filtrata sul campo ‘CodEsercizio’ e ordinata per il numero di serie; in questo modo vengono recuperati tutti i dettagli associati all’esercizio selezionato. Per poter visualizzare questi dati a video tramite una struttura che ricordi una tabella, occorre creare una classe, da noi

chiamata 'TableRowLayout', che si occupi di gestirne il layout grafico. Questa classe è formata principalmente da tre campi allineati orizzontalmente, due TextView, per mostrare rispettivamente i campi 'Serie' e 'Ripetizioni', e un EditText per il campo 'Peso'.

Riassumendo, per creare la tabella dei dettagli occorre:

1. Dichiarare la query da eseguire sulla tabella 'Dettaglio Serie';
2. Accedere al database locale in modalità di sola lettura;
3. Inizializzare una variabile di tipo Cursor, per poter esaminare riga per riga i risultati ottenuti dall'esecuzione della query precedentemente dichiarata;
4. Creare e aggiungere all'interfaccia grafica un oggetto di tipo TableRowLayout, per ciascuna serie recuperata.

Quanto detto finora è stato implementato con l'utilizzo del seguente codice:

```
1. // Inizializzazione della query
2. String sqlGetDettagli = "SELECT * FROM DETTAGLIO_SERIE WHERE CodEsercizio = "
3.     + esSelezionato.getInt("CodEsercizio") + " "
4.     + "ORDER BY Numero ASC";
5. // Apertura del database in modalità di sola lettura
6. SQLiteDatabase db = MainActivity.fitDB.getReadableDatabase();
7. // Layout grafico dove inserire le varie righe
8. tabella = (LinearLayout) findViewById(R.id.tableDettagli);
9. // Inizializzazione cursore
10. Cursor cursor = db.rawQuery(sqlGetDettagli, null);
11. if (cursor.moveToFirst())
12. {
13.     do
14.     {
15.         // Inizializzazione del TableRowLayout con gli elementi della serie selezionata
16.         TableRowLayout riga = new TableRowLayout(getApplicationContext(),
17.             (cursor.getInt(4) + 1), cursor.getString(5), cursor.getString(6));
18.         // Aggiunta del TableRowLayout al layout grafico
19.         tabella.addView(riga);
20.     } while (cursor.moveToNext());
21. }
22. cursor.close();
23. db.close();
```

Questa tabella viene visualizzata sia nel caso in cui l'utente abbia iniziato l'allenamento, sia nel caso in cui abbia aperto la finestra soltanto per visualizzarne i dettagli. Come detto precedentemente, se l'utente ha deciso di iniziare l'allenamento, deve essere possibile indicare al sistema quali serie sono state svolte. Per far questo, è stata modificata la struttura di TableRowLayout



aggiungendo una colonna con un'immagine rappresentante una “v” oppure una “freccia”: la prima viene utilizzata per indicare le serie già eseguite, l'altra per specificare quella che si sta eseguendo. Per poter indicare al sistema quali serie sono state effettuate e quali no, è stato deciso di affiancare alla tabella un tasto con il simbolo “tap”. Alla pressione di questo pulsante viene impostata la serie attuale come ‘eseguita’ e la serie seguente come ‘da eseguire’. Una volta completate tutte le serie dell'esercizio selezionato, il pulsante “tap” viene disabilitato.

Subito dopo la tabella dei dettagli, vengono visualizzate le note di esecuzione dell'esercizio e il tempo di recupero; anche questi attributi, come quelli relativi all'esercizio base presenti a inizio pagina, sono stati recuperati via Bundle. Siccome il tempo di recupero potrebbe non esser stato assegnato oppure potrebbe non esser adeguato per l'utente, il sistema deve permetterne la modifica. Si è deciso quindi di affiancare al campo ‘Recupero’ un tasto (con l'immagine di un timer), tramite il quale è possibile impostare il tempo di riposo manualmente. Alla pressione di questo pulsante, viene aperto un AlertDialog composto principalmente da due NumberPicker, uno relativo ai minuti e l'altro ai secondi. Per evitare di impostare un tempo di recupero troppo elevato, si è deciso di limitare la scelta dei minuti ad un massimo di 5. Una volta scelto il recupero desiderato, è possibile confermarlo tramite la pressione del tasto ‘OK’, oppure annullarlo tramite il pulsante ‘Annulla’.

Come è stato anticipato precedentemente, per facilitare la sessione di allenamento all'utente, si è deciso di aggiungere un Timer per contare il tempo di recupero tra una serie e l'altra. Al fine di creare un'interfaccia semplice e intuitiva per questo strumento, si è pensato di riprodurre lo stile grafico adottato da Google per il suo Widget, che consiste in una TextView cliccabile circonscritta da una ProgressBar. Alla pressione della TextView è possibile avviare o interrompere il conto alla rovescia, mentre la ProgressBar mostra in tempo reale l'avanzamento del tempo. Normalmente la ProgressBar è visualizzata come una barra orizzontale: per poterla rendere circolare, è necessario assegnarle un background personalizzato. È stato quindi creato, nella cartella drawable, il file

'progressbar.xml', in cui sono definiti i valori degli attributi che permettono di ottenere una ProgressBar con il layout desiderato. I valori sono i seguenti:

- android:shape = "ring", per avere la ProgressBar circolare anziché lineare;
- android:thickness = "3dp", per avere una ProgressBar sottile;
- android:color = "#EEEEED" sull'item "Background", per specificare che la parte della ProgressBar relativa al tempo rimanente deve essere di colore grigio;
- android:color = "#33B5E5" sull'item "Progress", per specificare che la parte relativa al tempo passato dev'essere di colore celeste.

A questo punto si ha a disposizione una ProgressBar circolare: il problema è che questo oggetto mostra l'avanzamento della barra a partire dall'inizio verso la fine (quindi in senso orario), implementando quindi il layout di un cronometro piuttosto che di un conto alla rovescia. Per sistemare correttamente la grafica in modo tale che la barra di avanzamento scorra dalla fine verso l'inizio (quindi in senso antiorario), è necessario implementare l'Override della classe ProgressBar (da noi chiamata InverseProgressBar). In questa nuova classe va impostata la scala dell'asse X a '-1' nell'evento onDraw (tramite l'attributo "scale" dell'oggetto Canvas) per ottenere il caricamento inverso della barra.

Ora che è stata sistemata la parte grafica, andiamo ad analizzare la parte funzionale del Timer. Una volta inizializzata la TextView con il messaggio 'Avvia', viene impostato il 'click listener' che permette l'inizio del conto alla rovescia. Alla pressione della Label viene inizializzato un oggetto di tipo CountdownTimer specificando durata (con i valori recuperati dal campo 'Recupero', opportunamente convertiti in millisecondi, in quanto unica unità di misura accettata da questo tipo di dato) e l'intervallo di tempo (sempre in millisecondi, nel nostro caso 100ms) in cui viene richiamato l'evento 'onTick'. Con questo evento è possibile specificare le azioni che devono essere eseguite ad ogni 'intervallo di tempo' trascorso: nel nostro caso ogni 100 millisecondi viene aggiornata la ProgressBar e il tempo rimanente che compare nella TextView centrale. L'intervallo di tempo da noi adottato è stato scelto per mantenere

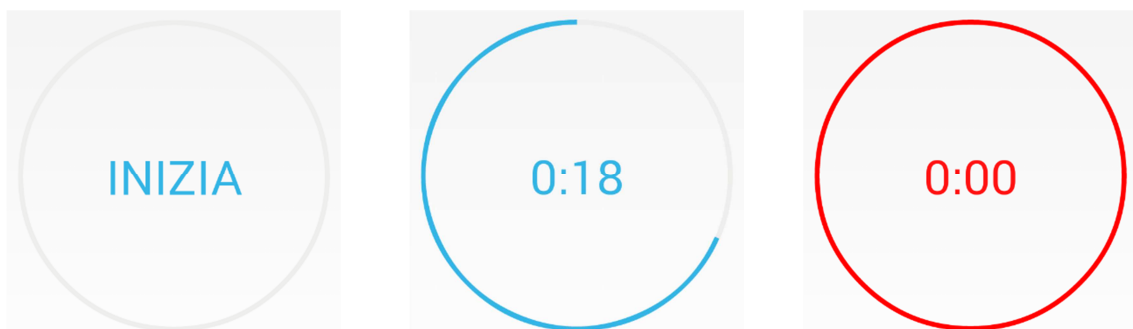
l'avanzamento della ProgressBar fluido e allo stesso tempo per cercare di non utilizzare troppe risorse del sistema.

Una volta scaduto il tempo, viene richiamato l'evento "onFinish", in cui viene impostato di colore rosso l'intero layout grafico del Timer, viene associata alla TextView centrale un'animazione di lampeggiamento e viene fatto partire un secondo CountdownTimer (della durata di 5 secondi) in cui vengono emessi ripetutamente tre 'beep' consecutivi. Questo suono è stato creato tramite l'utilizzo della classe ToneGenerator, scegliendo come tono base un tono di allarme (nel nostro caso 'TONE\_CDMA\_ALERT\_CALL\_GUARD') e specificandone il volume e la durata.

Al termine del periodo dell'allarme (ossia allo scadere del secondo countdown) vengono reimpostati i colori iniziali (la scritta della TextView assume il valore 'Riavvia'), viene interrotta l'animazione di lampeggiamento e viene fermata la riproduzione dei toni.

Come detto precedentemente, è possibile mettere in pausa il Timer premendo la TextView prima che il tempo sia scaduto. Alla pressione della Label quindi, se il testo è diverso da 'Avvia' e 'Riavvia', viene salvato in una variabile il numero di millisecondi rimanenti e viene interrotto il CountdownTimer. Premendo nuovamente la TextView, viene avviato un nuovo CountdownTimer con durata totale equivalente al tempo residuo del vecchio Timer.

Di seguito viene mostrata l'interfaccia del Timer appena implementato durante le varie fasi del funzionamento (fig.13).



[Fig.13 – Timer nella fase di partenza, di conto alla rovescia e a tempo scaduto]

Prima di procedere con la spiegazione dell'ultimo oggetto che compone la pagina relativa ai dettagli di un esercizio, è bene chiarire cosa succede quando l'utente decide di iniziare un allenamento. Tramite la pressione del pulsante 'Inizia allenamento' presente nella MainActivity, vengono inizializzate una serie di liste, ciascuna dedicata ad un gruppo muscolare. Ogni volta che viene completato correttamente un esercizio della scheda, viene effettuata una media sui pesi utilizzati e viene ritornata insieme al gruppo muscolare allenato: questo valore viene poi inserito nella lista appropriata. In questo modo nella MainActivity saranno momentaneamente memorizzate tante medie (divise tra le liste dei gruppi muscolari) quanti sono gli esercizi associati alla scheda. A fine allenamento, per poter memorizzare in modo permanente i dati (da mostrare nella sezione Statistiche), vengono create, o aperte qualora esistano già, tante SharedPreference quanti sono i gruppi muscolari allenati. Ognuna di queste ha il compito di memorizzare una lista di coppie chiave-valore, in cui la chiave è il giorno in cui è stato effettuato l'allenamento e il valore è la media delle medie dei pesi di ciascun esercizio, relativo a quel gruppo muscolare. Con questo sistema, ad ogni allenamento vengono creati tanti punti del tipo "giorno-mediaPesi" quanti sono i gruppi muscolari coinvolti. Per chiarire meglio questa parte viene esposto il seguente esempio: si vuole eseguire una scheda composta da tre esercizi, due per bicipiti e uno per tricipiti. Questi vengono effettuati utilizzando i seguenti pesi:

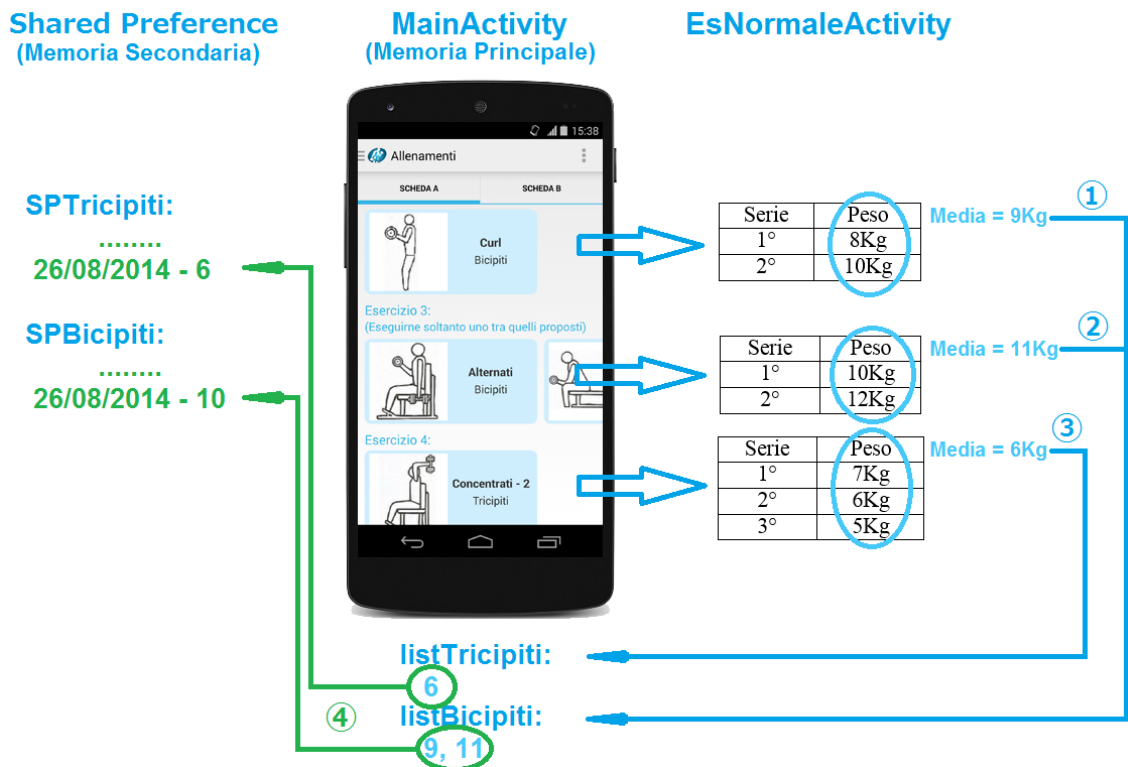
1° Es. - Curl (Bicipiti)	
1°	8Kg
2°	10Kg

2° Es. - Alternati (Bicipiti)	
1°	10Kg
2°	12Kg

3° Es. - Concentrati (Tricipiti)	
1°	7Kg
2°	6Kg
3°	5Kg

Nella MainActivity vengono quindi inizializzate due liste, una per i bicipiti e l'altra per i tricipiti. Finiti i tre esercizi, la prima lista conterrà due valori (9Kg e 11Kg, rispettivamente calcolati effettuando la media dei pesi delle serie del primo e del secondo esercizio), mentre la seconda conterrà un solo valore (6Kg), riferito all'unico esercizio che allena i tricipiti. Al termine dell'allenamento si accederà alle SharedPreference relative ai bicipiti e ai tricipiti: nella prima verrà inserito il valore "data\_odierna - 10Kg", dove 10Kg è

stato calcolato dalla media degli elementi della lista inerente ai bicipiti, mentre nella seconda verrà inserito “data\_odierna – 6Kg”, calcolato allo stesso modo facendo riferimento alla lista dei tricipiti. In questo modo, tramite la sezione Statistiche è possibile sapere che in data “data\_odierna” è stato utilizzato un peso medio di 10Kg per i bicipiti e di 6Kg per i tricipiti. Di seguito viene mostrato uno schema funzionale basato sull’esempio precedente (fig.14):



[Fig.14 – Esempio pratico della registrazione dei dati di un allenamento]

Per concludere la descrizione della pagina relativa ai dettagli di un esercizio, bisogna parlare del tasto ancorato al lato inferiore della schermata, utilizzato per chiudere l’Activity. Come accennato sopra, il comportamento di questo pulsante varia in funzione della pressione del tasto ‘Inizia allenamento’ presente nell’Activity principale. Se si è aperta la pagina corrente senza aver iniziato l’allenamento, l’unica funzione associata alla pressione del pulsante è la chiusura dell’Activity, altrimenti, se è stato avviato l’allenamento, prima di terminare l’Activity devono essere passati alla MainActivity i dati relativi all’esercizio svolto, per poterli poi inserire nella sezione statistiche.

In quest'ultimo caso, per poter completare correttamente l'esercizio occorre eseguire tutte le serie che lo compongono, indicando di volta in volta i progressi effettuati, tramite la pressione del tasto 'Tap'. Una volta svolte correttamente tutte le serie, il pulsante "Fine" viene colorato di verde, per indicare all'utente il completamento dell'esercizio: alla pressione di esso, viene restituito all'Activity principale il gruppo muscolare interessato dall'esercizio e la media aritmetica dei pesi utilizzati nelle varie serie (si ricordi che è possibile specificare il peso per ciascuna serie dalla tabella dei dettagli). Questo valore, come spiegato precedentemente, viene aggiunto alla lista delle medie relative al gruppo muscolare associato; si ricordi che tutti i controlli che vengono effettuati alla chiusura dell'Activity inerente all'esercizio, vengono gestiti dal metodo "onActivityResult" relativo all'Activity principale.

Nel caso in cui si preme il tasto 'Fine esercizio' senza aver completato tutte le serie, il sistema chiederà all'utente se desidera veramente annullare l'esercizio: se viene premuta l'opzione 'Esci ora', l'Activity corrente verrà terminata saltando la fase di salvataggio dei dati, altrimenti l'AlertDialog verrà chiuso permettendo all'utente di continuare l'allenamento.

Oltre a ritornare la media dei pesi, ogni volta che un esercizio viene completato correttamente, viene restituito anche il 'Gruppo' di cui ne fa parte: questo valore verrà inserito in una lista apposita (istanziata nella MainActivity) che viene utilizzata per memorizzare l'elenco degli esercizi completati correttamente (vedremo a breve come questa lista viene utilizzata).

Quando viene completato l'ultimo esercizio, l'Activity corrente, oltre a restituire i valori descritti precedentemente, ritorna anche un valore booleano per far sapere all'Activity principale che l'allenamento è stato concluso. A questo punto la MainActivity, come accennato in questo paragrafo, procede con l'effettuare la media delle medie dei pesi utilizzati per ciascun gruppo muscolare (recuperate dalle liste in memoria principale); ciascuna di queste medie calcolate viene infine aggiunta alla rispettiva mappa presente nelle SharedPreferences, utilizzando come valore di chiave la data in cui si è svolto l'allenamento. Prima di proseguire, è bene effettuare un piccolo chiarimento sulla memorizzazione

delle mappe relative ai vari gruppi muscolari: tramite le SharedPreference è possibile salvare solamente i tipi di dato primitivi (Integer, String, Boolean, ...). Per poter salvare un oggetto di tipo “Map<Long, Integer>” è necessario convertirlo in stringa tramite l'utilizzo della libreria ‘GSON’, messa a disposizione da Google. Questa classe viene utilizzata per convertire gli oggetti Java nelle loro rispettive rappresentazioni JSON, e viceversa, per recuperare dalle stringhe JSON l'equivalente oggetto iniziale.

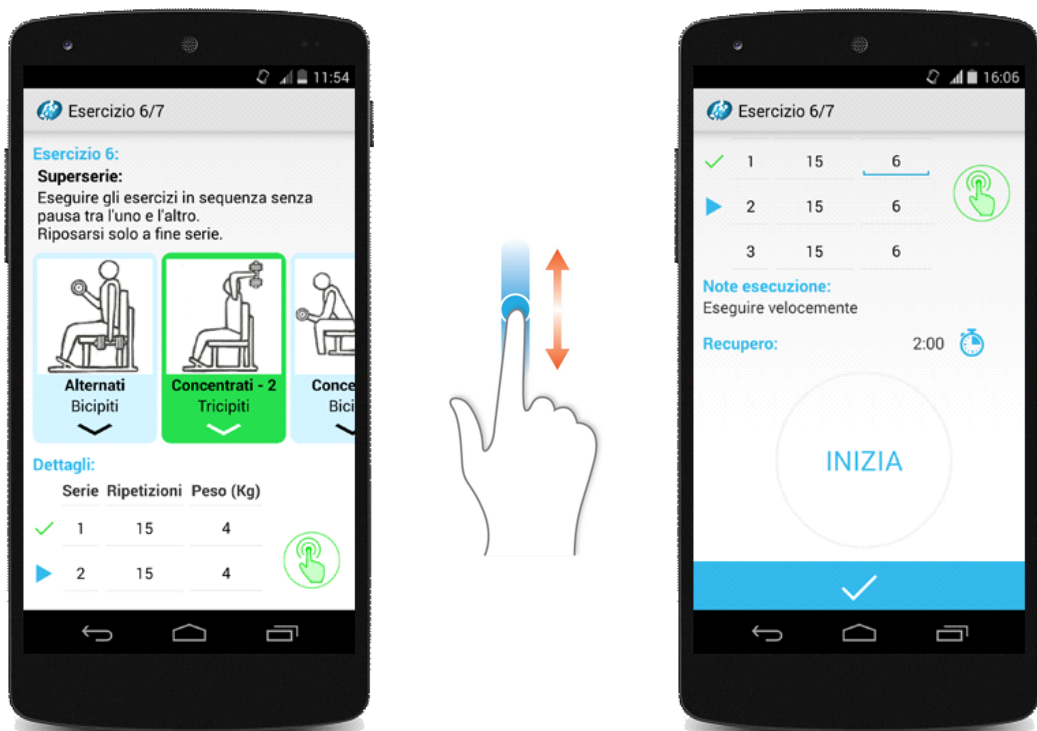
In questa sezione del programma vengono solamente inseriti i punti all'interno delle varie mappe: questi verranno recuperati ed utilizzati successivamente nella sezione statistiche.

Oltre al salvataggio delle medie, completato l'ultimo esercizio, il sistema compila un resoconto sull'allenamento appena svolto (fig.15): tramite un AlertDialog personalizzato viene mostrato all'utente il numero di esercizi completati correttamente, il tempo impiegato (recuperato da un oggetto di tipo ‘Chronometer’ avviato alla pressione del tasto ‘Inizia Allenamento’) e un giudizio basato sul numero degli esercizi svolti, sia sotto forma di testo, sia di “RatingBar”.



[Fig.15 – Valutazione di fine allenamento]

Quanto detto finora riguardava la pagina relativa agli esercizi ‘normali’ (EsNormaleActivity), ma come detto, in una scheda possono comparire anche delle ‘superserie’: alla pressione di un esercizio di questo tipo, dev’esser caricata una schermata con un layout grafico adeguato alla sua rappresentazione, ma che offra le stesse funzionalità già implementate per gli esercizi singoli. Questa interfaccia viene implementata in un’Activity a parte, da noi chiamata “SuperSerieActivity” (fig. 16).



[Fig.16 – Interfaccia dell’Activity Superserie]

Come si può notare dall’immagine, la struttura grafica è del tutto simile a quella di un esercizio normale, con la differenza che al posto di un oggetto di tipo ‘EsercizioLayout’ per la rappresentazione dell’esercizio base selezionato, viene utilizzata una HorizontalScrollView, composta da tutti gli esercizi (in modo ordinato) che compongono la superserie. L’aspetto grafico di ciascuno di essi è definito in una classe a parte chiamata “SuperSerieLayout”, formata principalmente da un’immagine e due testi disposti verticalmente.

Al caricamento della pagina viene eseguita una query di selezione sulla tabella ‘Es In Scheda’, per recuperare tutti i codici degli esercizi base associati al



codice dell'esercizio selezionato. A questo punto viene effettuata una nuova query sulla tabella 'Esercizio Base' per ottenere i dettagli relativi agli esercizi che compongono la superserie; ognuno di questi verrà infine inserito nella HorizontalScrollView sotto forma di 'SuperSerieLayout'.

Per rendere più chiara possibile l'interfaccia di questa schermata, è stato deciso di visualizzare i dettagli di un solo esercizio base per volta. È stato quindi aggiunto un "clickListener" su ciascun esercizio della superserie, in modo tale da evidenziare l'esercizio selezionato e visualizzarne, nella tabella sottostante, tutti i dettagli che lo caratterizzano.

Come già affermato, il restante layout grafico e le funzionalità sono del tutto identiche a quelle già esposte.

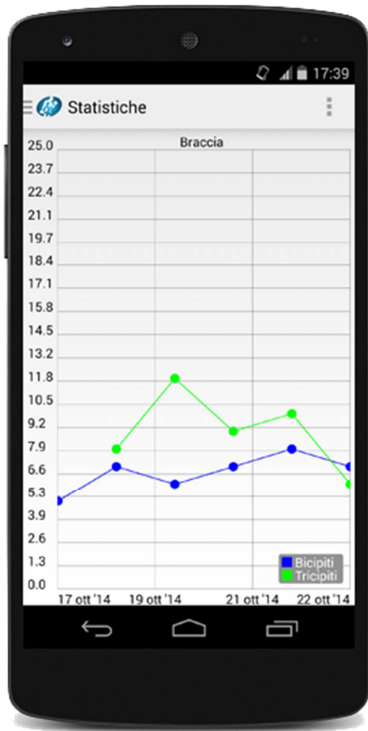
Detto questo, è stata terminata la parte di progettazione della sezione allenamenti. Nel paragrafo successivo verrà analizzata la sezione 'Statistiche'.

#### **2.2.4 – Sezione Statistiche**

Questa sezione è stata pensata per tenere traccia dei progressi effettuati dall'utente nel tempo. La struttura di questa pagina (contenuta in ChartFragment) è basata su tre Tab, una per ciascuna parte del corpo, rispettivamente 'Braccia', 'Torso' e 'Gambe'. Ognuna di queste Tab contiene un grafico che mostra i dati dei gruppi muscolari che ne fanno parte: ad esempio, nel grafico relativo alla Tab 'Braccia' vengono visualizzati separatamente i dati relativi ai bicipiti, tricipiti e deltoidi.

Per l'implementazione dei grafici è stata utilizzata una libreria chiamata 'GraphView', scaricabile gratuitamente da <http://android-graphview.org/>, che permette di creare facilmente dei diagrammi personalizzabili. Prima di inserire i dati nel grafico, ne è stata definita la sua struttura: per l'asse delle ascisse è stato deciso di visualizzare le date degli allenamenti (generalmente vengono visualizzate quattro date ad intervalli di sette giorni, in modo tale da avere una panoramica immediata dei progressi effettuati nell'ultimo mese), mentre per l'asse delle ordinate si è scelto di mostrare il peso medio.

Come approfondito nel paragrafo precedente, a fine allenamento vengono salvati nelle SharedPreference i pesi medi utilizzati sui vari gruppi muscolari. In questa sezione bisogna recuperare quei dati, e inserirli sotto forma di punti nel grafico. Le SharedPreference relative alle mappe sono memorizzate sotto forma di stringhe: è quindi necessario riconvertirle nel tipo Map<Long, Integer> tramite



l'utilizzo della libreria 'Gson', per poterne leggere i punti. Una volta recuperati tutti i dati, si procede con l'inserimento nel grafico: vengono create quindi tante 'GraphViewSeries' quanti sono i gruppi muscolari interessati. Tramite l'utilizzo di questa classe è possibile creare una linea colorata che colleghi tutti i punti recuperati dalla stessa SharedPreference. Il risultato di queste operazioni è la visualizzazione di una linea blu che collega tutte le medie relative ai bicipiti e una linea verde che collega tutte le medie riguardanti i tricipiti, sullo stesso grafico (fig.17).

[Fig.17 – Schermata Statistiche relativa al grafico per le braccia]

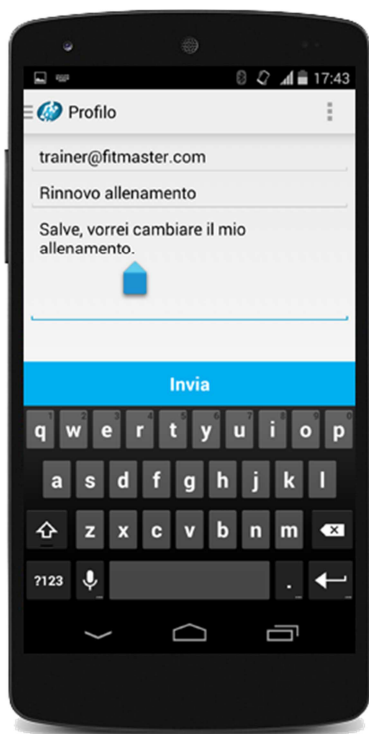
Il grafico è stato anche dotato di una legenda che identifica i gruppi muscolari in base ai colori utilizzati.

È stata infine gestita l'attivazione automatica dello scorrimento orizzontale qualora i dati memorizzati coprano un arco temporale superiore al mese.

## 2.2.5 – Sezione Contatta Trainer

Capita spesso che un utente abbia bisogno di parlare con il proprio personal trainer, ma non sempre gli è possibile poterlo incontrare di persona. Per questo motivo è stata aggiunta questa sezione all'app, che permette di comunicare con l'istruttore via mail, in qualunque momento.

L'interfaccia di questa pagina (fig.18), implementata in TrainerFragment.Java, è strutturata su tre semplici campi di testo, uno per il destinatario della mail, uno per l'oggetto del messaggio e l'ultimo per il testo.



Alla pressione del tasto 'Invia Mail', viene aperto un AlertDialog in cui viene richiesto di scegliere il client con cui inviare la mail: i valori dei tre campi appena compilati verranno caricati automaticamente nelle relative caselle dell'applicazione scelta, lasciando quindi all'utente il singolo compito di spedire la mail. A mail inviata, l'applicazione client verrà chiusa e i campi 'oggetto' e 'corpo messaggio' verranno puliti. Nel caso in cui non sia presente un client email installato sul device, il sistema informerà l'utente dell'impossibilità di spedire il messaggio.

[Fig.18 – Schermata Contatta Trainer]

In aggiunta, alla pressione del tasto 'Invia Mail' viene memorizzato in una SharedPreferences l'indirizzo mail del personal trainer, in modo tale che ai successivi avvii dell'app, questo possa essere caricato in modo automatico.

Per rendere più comodo l'invio delle mail, è stato deciso di mantenere sempre visibile il tasto apposito, anche nel momento in cui la tastiera viene aperta.

## 2.2.6 – Sezione Profilo



[Fig.19 – Schermata Profilo]

Questa sezione (implementata nel file ProfileFragment.Java) è riservata alla visualizzazione dei dati personali relativi all'utente che ha effettuato l'accesso. Questa pagina è basata principalmente su una ScrollView verticale (fig.19), contenente tanti campi di testo quanti sono gli attributi della tabella 'Utente', suddivisi per categorie. Per rendere l'applicazione più fluida, i valori di questi campi vengono caricati ogni volta dal database locale.

Per ovvi motivi, la visualizzazione del campo password viene offuscata.

## 2.2.7 – Sezione Archivio Esercizi

Questa sezione è stata progettata per poter consultare tutti gli esercizi che la palestra solitamente utilizza per la compilazione delle schede. Al momento questa pagina non ha ulteriori funzionalità, ma sarà una base utile per i progetti futuri (che verranno spiegati in un capitolo a parte). È possibile visualizzare sia l'intero archivio esercizi, sia solamente quelli che riguardano una particolare zona del corpo.

Questa pagina è definita nel Fragment "ExerciseFragment" ed è composta da una ListView e da una HorizontalScrollView (fig.20): la prima viene utilizzata per visualizzare la lista di esercizi base recuperati dal database locale, mentre la seconda mette a disposizione una serie di layout (affiancati orizzontalmente), con i quali è possibile filtrare tale lista.



Questi ultimi sono stati dichiarati direttamente via xml e sono composti da un'immagine e da un testo, che indica il nome del filtro; l'immagine è una rappresentazione della tavola anatomica in cui viene evidenziata la zona muscolare interessata. Alla pressione di ciascuno di questi layout viene eseguita una query di selezione sulla tabella "Esercizio Base", per recuperare tutti gli esercizi dedicati alla fascia muscolare selezionata. Questi elementi verranno infine caricati nella ListView.

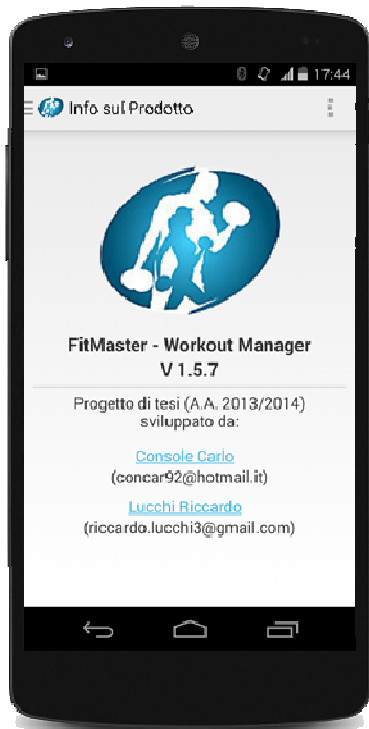
[Fig.20 – Schermata relativa all'Archivio Esercizi]

Di seguito viene mostrata una tabella che indica i filtri messi a disposizione nella HorizontalScrollView e i gruppi muscolari a cui fanno riferimento:

<b>Filtro</b>	<b>Gruppi Muscolari</b>
Tutti	Tutti
Spalle	Deltoidi, Trapezi
Petto	Pettorali
Braccia	Bicipiti, Tricipiti, Avambracci
Schiena	Dorsali, Lombari
Addome	Addominali alti, bassi, laterali
Gambe	Adduttori, Abduttori, Quadricipiti, Femorali, Polpacci
Altro	Altro

Anche per questa ListView, come di consueto, è stato necessario definire il tipo di oggetto che viene utilizzato nella lista (EsBase, definito da un'immagine e due testi relativi agli attributi della tabella 'Esercizio Base'), oltre all'adattatore che ne permette l'inserimento.

## 2.2.8 – Sezione Info



[Fig.21 – Schermata di Info sul Prodotto]

Siamo dunque arrivati a spiegare l'ultima schermata di questa tesi, quella relativa alle informazioni sul prodotto. In questa pagina viene visualizzato in primo piano il logo del software e a seguire la versione, l'anno accademico in cui è stato progettato e i nominativi degli sviluppatori (con relativi gli indirizzi email). Tramite la pressione di questi ultimi è possibile accedere alle rispettive pagine Facebook. A lato viene mostrata l'interfaccia di questa pagina (fig.21).

## Capitolo 3 Conclusioni

Siamo giunti ora al termine di quest'esperienza, che ci ha permesso di realizzare il sistema ideato. Il progetto FitMaster – Workout Manager continuerà ad essere migliorato anche fuori dall'ambito universitario. Abbiamo già in mente una serie di aggiornamenti per rendere l'applicazione mobile ancora migliore:

- Possibilità di creare un allenamento direttamente da Smartphone;
- Aggiunta di un campo 'Video Dimostrativo' agli esercizi presenti nell'archivio, che ne mostri all'utente la corretta esecuzione;
- Condivisione su Facebook del riepilogo compilato a fine allenamento;

- Espansione della sezione ‘Statistiche’ su tutti i gruppi muscolari del corpo, con aggiunta delle Tab dedicate agli esercizi a corpo libero (addominali, push-up,...) e al fitness (corsa, step...);
- Creazione di una sezione dedicata alle sfide con gli amici (come ad esempio l’esecuzione di un numero massimo di flessioni nell’arco di un tempo prestabilito).

Inoltre, nel caso in cui questo progetto riscontri un’adeguata notorietà, saremmo pronti ad estenderlo anche per piattaforma iOS / Windows Phone.