Università degli Studi di Salerno
Corso di Ingegneria del Software

mobilFarm
Object Design Document
Versione 0.7

Data: 05/01/2017

Partecipanti:

Nome	Matricola
D'Avanzo Giuseppina	0512103032
Fiorentino Giuseppa	0512103158
Menichino Alfonso	0512102852
Volpe Alberto	0512103311

Revision History

Data	Versione	Descrizione	Autore

Sommario

1.1. Compromessi dell'Object Design 5
- Sviluppo delegato a terzi vs. Sviluppo interno 5
- Rispetto dei tempi di consegna vs. Sviluppo completo delle funzionalità 5
- Usabilità vs Sicurezza 5

1.2. Linee guida e convenzioni 5
1.2.1 Classi Java 6
1.2.2 Layout XML 7
1.2.3 Manager php 8
1.2.4 Database mySQL 9
1.3 Definizioni, acronimi, e abbreviazioni 9
1.4 Riferimenti 10
2 Packages 10
- GestoreAccountClient 10
- GestoreAccountServer 11
- GestoreFarmacoClient 11
- GestoreFarmacoServer 12
- GestorePianoClient 12
- GestorePianoServer 13

3 Interfacce delle Classi 14
3.1 GestoreAccountClient 14
3.1.1 Class LoginHome 14
3.1.2 Class Credenziali 14
3.1.3 Class RegistrazionePaziente 14
3.1.4 Class RegistrazioneDottore 14
3.1.5 Class DottoreHome 14
3.1.6 Class PazienteHome 14
3.1.7 Class Rubrica 14
3.1.8 Class RicercaDottore 14
3.1.9 Class ProfiloDottore 14
3.1.10 Class SchedaPaziente 15
3.1.11 Class ProfiloPaziente 15
3.1.12 Class SchedaDottore 15
3.1.13 Class ModalitaSOS 15
3.1.14 Class GestioneAccount 15
3.1.15 Class GestioneRubrica 15
3.2 GestoreFarmacoClient 15
3.2.1 Class VMF 15
3.2.2 Class FarmacoInScadenza 15
3.2.3 Class RicercaFarmaco 16
3.2.4 Class NuovoFarmaco 16
3.2.5 Class GestioneFarmaco 16
3.2.6 Class GestioneVMF 16
3.2.7 Class ListaFarmaciInScadenzaAdapter 16
3.2.8 Class ListaFarmaciVMFAdapter 16
3.3 GestorePianoClient 16

3.3.1 Class PianiTerapeutici 16
3.3.2 Class PianoTerapeutico 16
3.3.3 Class FarmacoPianoTerapeutico 16
3.3.4 Class FarmacoDaAssumere 16
3.3.5 Class GestionePianoTerapeutico 17
3.3.6 Class ListaFarmaciPianoAdapter 17
3.3.7 Class ScegliFarmacoPianoAdapter 17
3.4 UtilityClient 17
3.4.1 Class mobilFarmServer 17
3.4.2 Class FontsOvverride 17
3.4.3 Class UIManager 17
3.4.4 Class ListaDottoriAdapter 17
3.4.5 Class ListaFarmaciAdapter 17
3.4.6 Class ListaPazientiAdapter 17
3.4.7 Class ListaPianiTerapeuticiAdapter 17
3.4.8 Class NotificationManager 18
3.4.9 Class NotificationMessage 18
3.4.10 Class TimePickerFragment 18
3.4.11 Class ActionNotAuthorizedException 18
3.4.12 Class ConnectionErrorException 18
3.4.13 Class ErrorValueException 18
3.4.14 Class ResponseErrorException 18
3.5 GestoreAccountServer 18
3.5.1 Class AccountManager 18
3.6 GestorePianoServer 19
3.6.1 Class PianoTerapeuticoManager 19
3.7 GestoreFarmacoServer 19
3.7.1 Class FarmacoManager 19

1. Introduzione

Questo documento, denominato Object Design Document (ODD), conterrà le indicazioni relative al design, le indicazioni relative allo sviluppo delle interfacce dei sottosistemi e le suddivisioni dei sottosistemi nei package e nelle classi.

1.1. Compromessi dell'Object Design

Dobbiamo scegliere tra diversi modi di implementare il modello di analisi, con l'obiettivo di minimizzare il tempo di esecuzione, la memoria e altre misure di costo.

- **Sviluppo delegato a terzi vs. Sviluppo interno**

  Dato il know-how interno, mobilFarm verrà sviluppato da un team interno sulla base delle tecnologie e framework open-source e con il supporto hardware e di gestione di Amazon Web Services. Delegare ad una società terza lo sviluppo di mobilFarm, comprometterebbe maggiormente le risorse economiche e non ci permetterebbe di acquisire ulteriore know-how per sviluppi futuri.

- **Rispetto dei tempi di consegna vs. Sviluppo completo delle funzionalità**

  Al fine di produrre un prodotto completo e di qualità abbiamo deciso di focalizzarci sullo sviluppo completo delle funzionalità, mettendo in secondo piano le tempistiche di consegna.

- **Usabilità vs Sicurezza**

  mobilFarm avendo come target di utenza anche quello delle persone di età over 60, garantisce una maggiore usabilità, per questo target di persone, mettendo in evidenza la password al momento del login, questo a discapito della sicurezza.

1.2. Linee guida e convenzioni

Questa sezione fornisce una panoramica circa la convenzione di denominazione in modo da rendere il programma più comprensibile, quindi più facili da leggere. Prima di tutto, dovremmo conoscere le regole di convenzione di denominazione per ogni tipo di identificatore come di seguito:

- **Identificare le classi(Activity)**: I nomi delle classi dovrebbero essere nomi, in caso misto con la prima lettera di ogni interno parola in maiuscolo (CamelCase). Oltre a questo, il nome della classe deve anche essere semplice e descrittivo.

- **Identificare i Layout**: I layout si dividono in activity, fragment e layout che descrivono la forma dei ListView. Il loro nome scritto totalmente in minuscolo con un "_ "che separa le varie parole.

- **Identificare le interfacce**: I nomi delle interfacce hanno la prima lettera maiuscola, come le classi.

- **Identificare i metodi**: I nomi dei metodi dovrebbero essere verbi, in caso misto con la prima lettera minuscola e la prima lettera di ogni interno parola in maiuscolo (CamelCase).

- **Identificare le variabili**: I nomi delle variabili non possono iniziare con il carattere "_" o "$", anche se è permesso all'interno del nome. I nomi delle variabili rappresentativi.

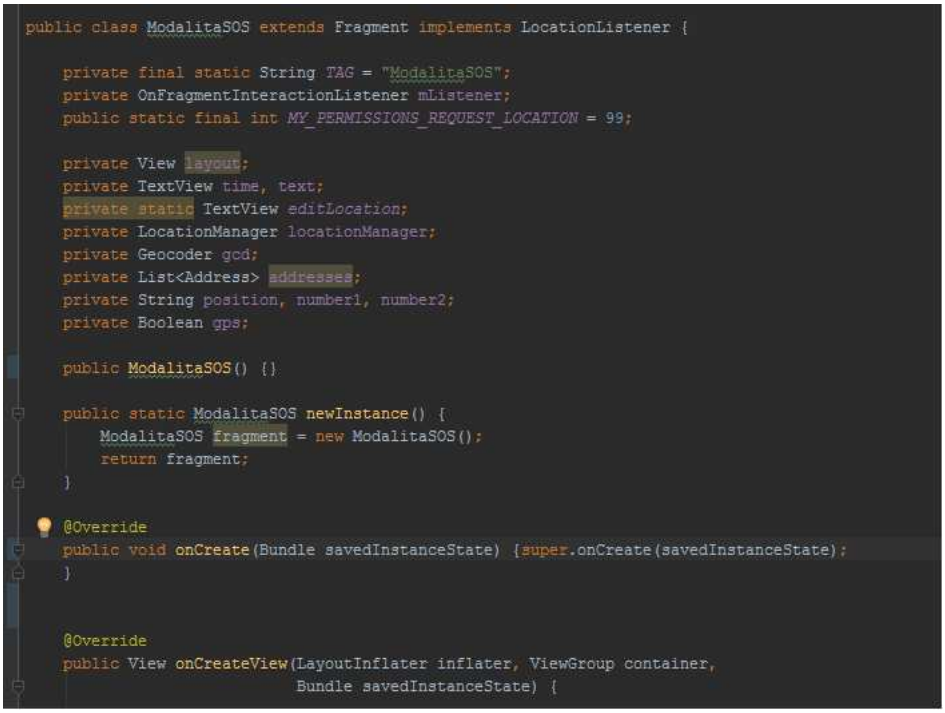
- **Identificare i manager**: I manager sono pagine php che si occupano dell'interazione con il Database.

1.2.1 Classi Java

I file sono strutturati:

- Istruzione package.
- Istruzione import.
- Dichiarazione della classe.
- Dichiarazione delle variabili di istanza.
- Metodo costruttore
- Dichiarazione dei metodi.

Esempio di un’Activity:



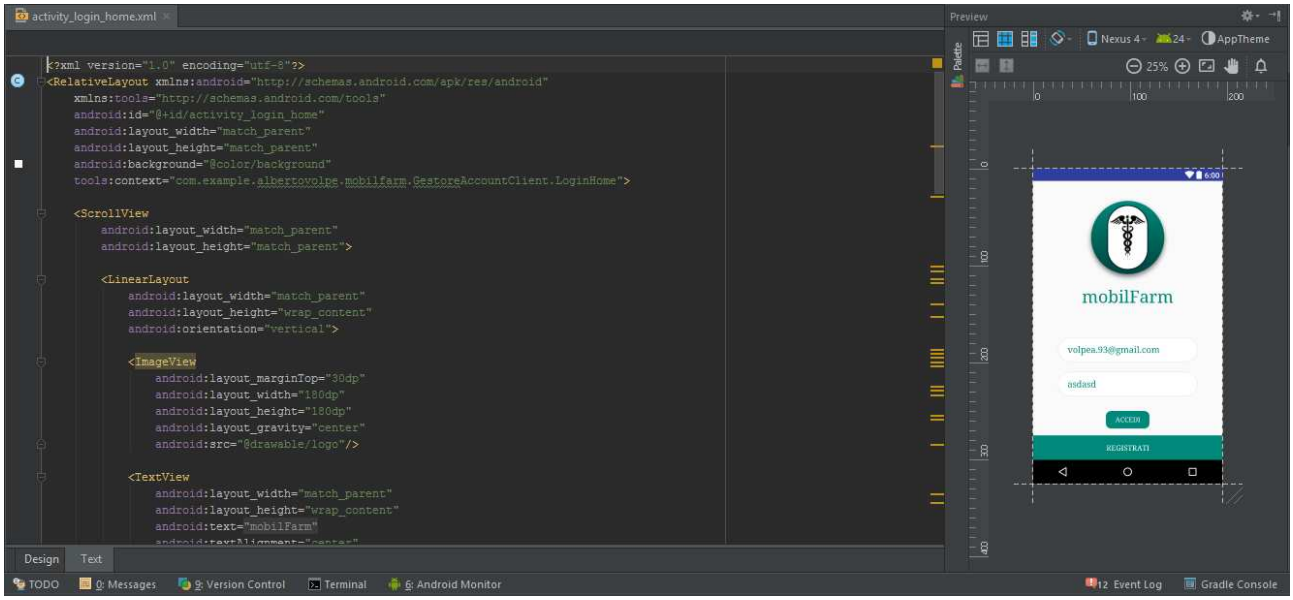
```
public class ModalitaSOS extends Fragment implements LocationListener {

    private final static String TAG = "ModalitaSOS";
    private OnFragmentInteractionListener mListener;
    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;

    private View layout;
    private TextView time, text;
    private static TextView editLocation;
    private LocationManager locationManager;
    private Geocoder gcd;
    private List<Address> addresses;
    private String position, number1, number2;
    private Boolean gps;

    public ModalitaSOS() {}

    public static ModalitaSOS newInstance() {
        ModalitaSOS fragment = new ModalitaSOS();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState);
    }


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

1.2.2 Layout XML

Le pagine di Layout XML devono utilizzare l’indentazione per facilitare la lettura, secondo le seguenti regole:

- Un’indentazione consiste in una tabulazione;
- Ogni tag deve avere un’indentazione maggiore del tag che lo contiene;
- Sebbene l’uso di questa pratica sia arbitrario, ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertuta;

Esempio di un Layout:



1.2.3 Manager php

Ogni pagina manager deve iniziare con il tag php “<?php” e terminare con “?>”. È buona norma indicare, appena aperto il tag, le informazioni riguardanti le specifiche di connessione, come il nome del Database e la directory del server.

Esempio di un manager php:

```php
<?php

header('Content-Type: application/json');
date_default_timezone_set("UTC");

//decode the request with json (for the requests form the mobile app)
$_REQUEST = json_decode(file_get_contents('php://input'), true);

$directoryServer = "mobilfarm.cqplmnbzeyqt.us-east-1.rds.amazonaws.com:3306";
$username = "mobilFarm";
$password = "mobilFarm";
$dbName = "mobilFarm";

switch($_REQUEST['action'])
{
  case "visualizzaPianiTerapeutici":
  {
    /* Verifico se l'utente è autorizzato a utilizzare questa operazione */
    if(isset($_COOKIE['UserID']) && ($_COOKIE['Type'] == "Paziente"))
    {
       try
       {
          /* Connetto al Database Server */
          $access = mysql_connect($directoryServer, $username, $password) or die(mysql_error());
          if($access)
          {
            /* Seleziono il Database */
            mysql_select_db($dbName);

            /* Parsing dei dati ricevuti per evitare la MYSQL Injection */
            $userID = mysql_real_escape_string(strip_tags($_COOKIE['UserID']));

            /* Preparo ed eseguo la query */
            $query = 'SELECT PianiTerapeutici.IDTerapia, PianiTerapeutici.NomePiano, PianiTerapeutici.CompilatoDa, Utenti.Nome, Utenti.
              Cognome
                      FROM Utenti, PianiTerapeutici
                      WHERE PianiTerapeutici.CompilatoDa=Utenti.UserID AND PianiTerapeutici.InviatoA = \''.$userID.'\'';
            $result = mysql_query($query) or die(mysql_error());
```

1.2.4 Database mySQL

Le tabelle del database dovrebbero essere in terza forma normale di Codd (3NF). I nomi delle tabelle devo seguire le seguenti regole:

- Devono essere composte solo da lettere;
- Il nome della tabella deve iniziare con una lettera maiuscola, se ci sono più parole si applica la sintassi del camelCase;
- il nome deve essere un sostantivo plurale tratto dal dominio del problema ed esplicativo del contenuto.

I nomi degli attributi:

- Il nome del campo deve iniziare con una lettera maiuscola, se ci sono più parole si applica la sintassi camelCase;
- I nomi dei campi non devono contenere accenti

1.3 Definizioni, acronimi, e abbreviazioni

Acronimi e abbreviazioni:

MVC: Model View Controller;
RAD: Requirements Analysis Document;
SDD: System Design Document;
ODD: Object Design Document
DB: Database
DBMS: Database Management System
GPS: Global Position System
S.O.S.: Save Our Souls / Soccorso Occorre Subito / Richiesta di soccorso in codice Morse;
VMF: VirtualmobilFarm;
DP: dot per inch.

Definizioni:

VirtualmobilFarm: Armadietto farmaceutico virtuale;
Storia Clinica: insieme di tutti i piani terapeutici di un paziente;
Deuteranopia: l'insensibilità al verde;
Protanopia: insensibilità al rosso;
Tritanopia: insensibilità al blu.

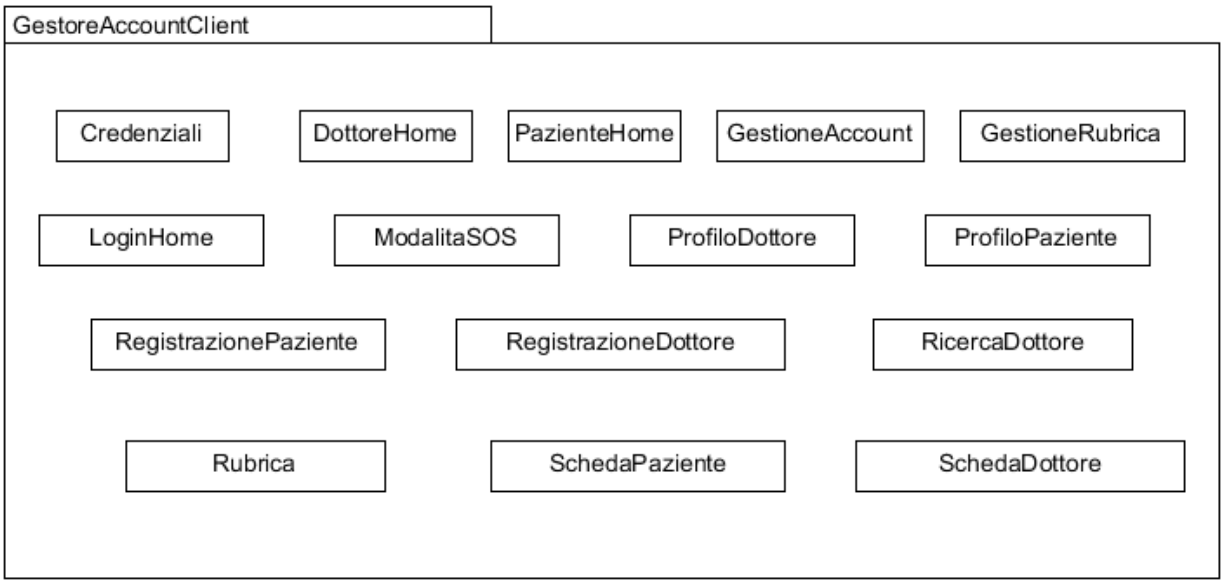
1.4 Riferimenti

- B. Bruegge, A.H. Dutoit, Object Oriented Software Engineering – Using UML, Patterns and Java, Prentice Hall;
- Sommerville, Software Engineering, Addison Wesley.
- Lehrstuhl für Angewandte Softwaretechnik.

2 Packages

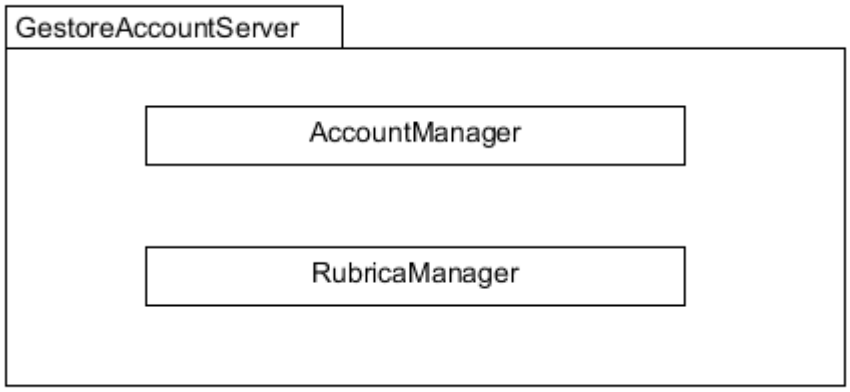
Durante l'applicazione del System Design, abbiamo composto tre sottosistemi: GestoreAccount, GestoreFarmaco, GestorePiano.
Questi sottosistemi sono trasversali rispetto all'intero sistema, proprio per questo suddivideremo ogni sottosistema rispettivamente in due package: il primo offrirà le funzionalità lato client, mentre il secondo si occuperà di offrire le relative API lato server.

- ***GestoreAccountClient***



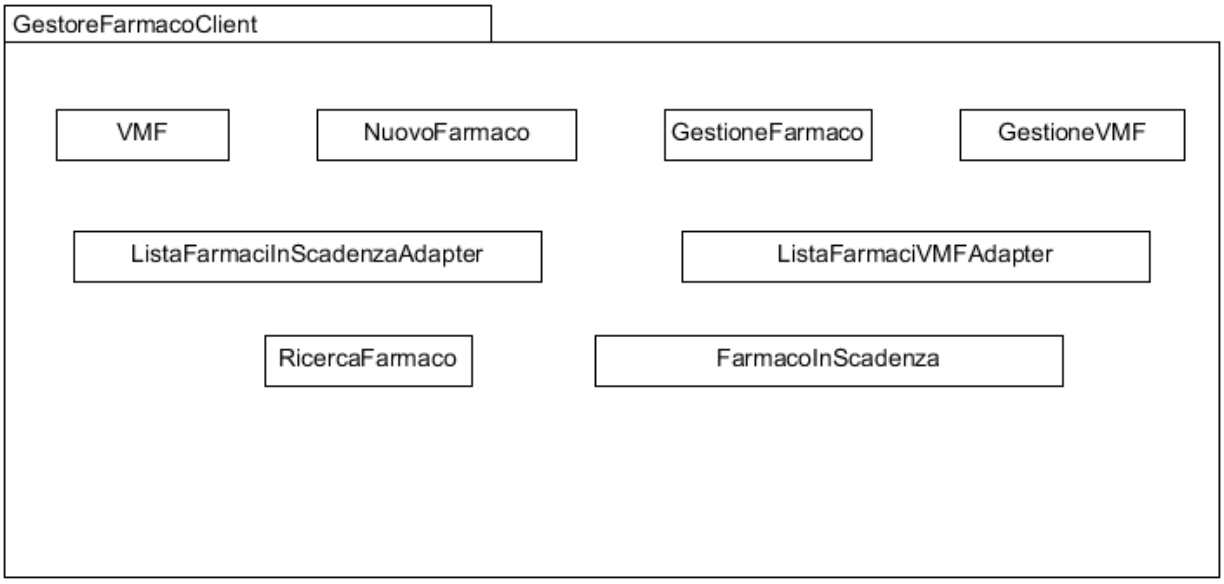
package che conterrà tutte le funzionalità del client relative alla gestione degli account;

- ***GestoreAccountServer***



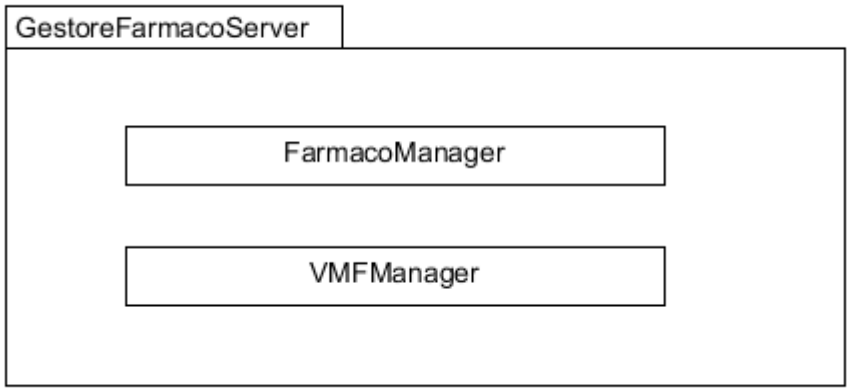
package che conterrà tutte le API sul server per leggere, inserire, eliminare e aggiornare le informazioni relative agli account;

- ***GestoreFarmacoClient***



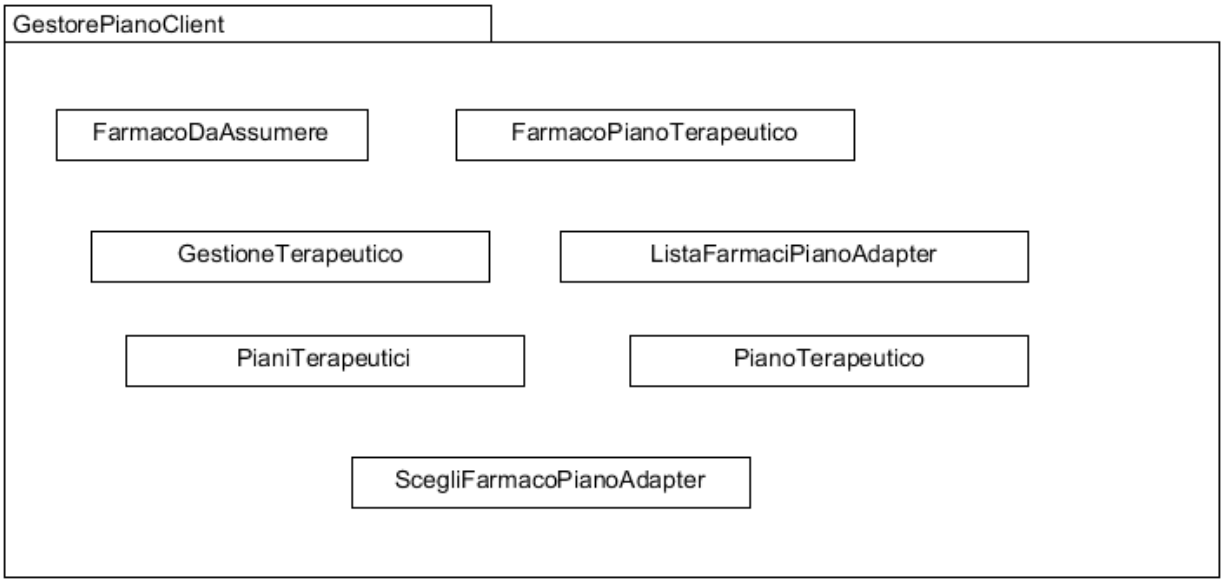
package che conterrà tutte le funzionalità del client relative alla gestione dei farmaci e del virtualmobilFarm;

- ***GestoreFarmacoServer***



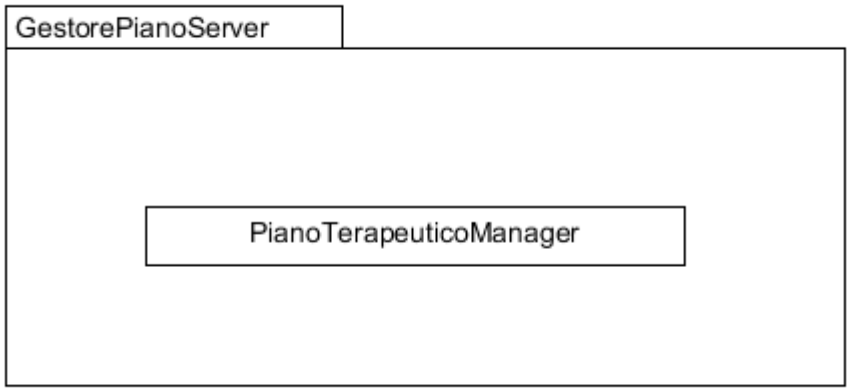
package che conterrà tutte le API sul server per leggere, inserire, eliminare e aggiornare le informazioni relative ai farmaci pubblici e ai propri farmaci (VMF);

- ***GestorePianoClient***



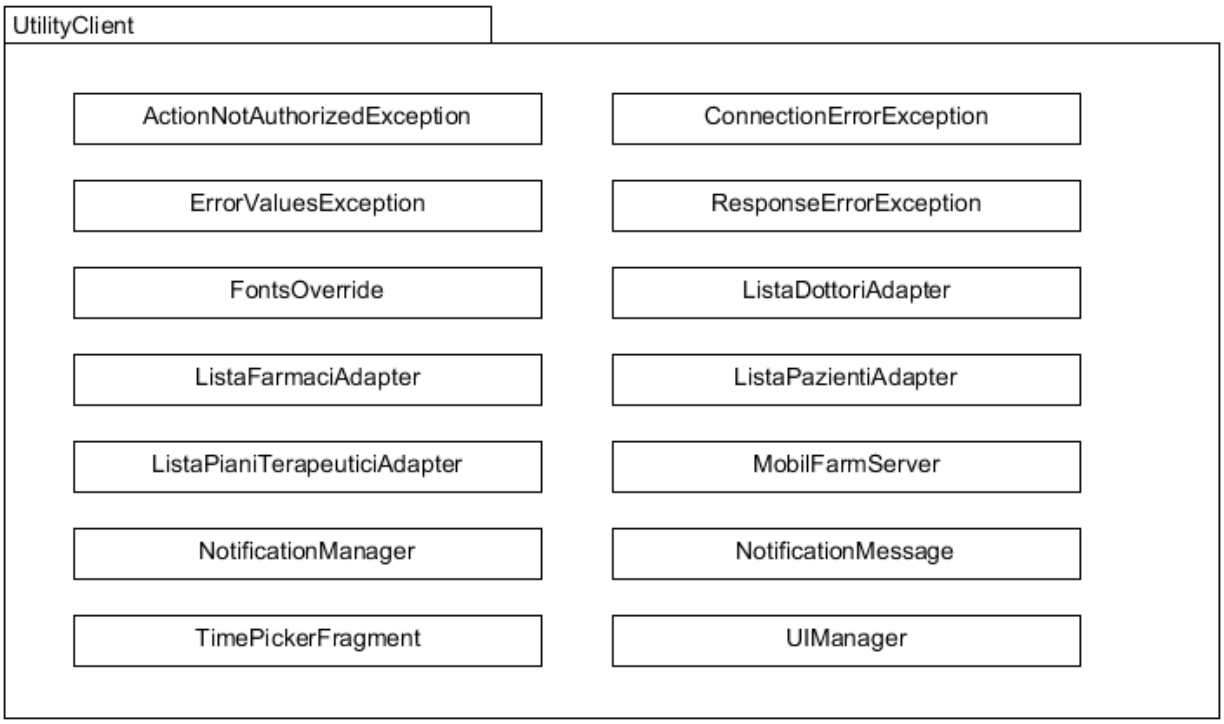
package che conterrà tutte le funzionalità del client relative alla gestione dei piani terapeutici;

- ***GestorePianoServer***



package che conterrà tutte le API sul server per leggere, inserire, eliminare e aggiornare le informazioni relative ai piani terapeutici;

Infine avremo un altro package denominato **UtilityClient** in cui saranno presenti le classi di utility che, al fine di supportare determinate funzionalità, vengono imposte del sistema.



3 Interfacce delle Classi

La descrizione delle interfacce verrà sviluppata all'interno dell'ambiente di sviluppo e successivamente generata grazie al tool javadoc.

3.1 GestoreAccountClient

3.1.1 Class LoginHome

Questa classe permette all'utente già registrato a mobilFarm di poter accedere al sistema vero e proprio effettuando il Login e permette ai nuovi utenti, non ancora registrati, di potersi registrare a mobilFarm.

3.1.2 Class Credenziali

Primo passo fondamentale verso la registrazione del proprio account, l'inserimento della propria mail con la quale effettuare la registrazione e una password. Inoltre è qui che l'utente decide la tipologia del suo Account (Paziente o Dottore).

3.1.3 Class RegistrazionePaziente

Secondo passo per la creazione del proprio account. Qui l'utente, avendo scelto la tipologia paziente, compila i campi richiesti per la registrazione in particolare i dettagli clinici che gli riguardano, come ad esempio le allergie.

3.1.4 Class RegistrazioneDottore

Secondo passo per la creazione del proprio account. Qui l'utente, avendo scelto la tipologia dottore, compila i campi richiesti per la registrazione in particolare il suo campo di specializzazione

3.1.5 Class DottoreHome

Schermata principale del sistema. Qui viene mostrata la lista dei farmaci presenti nel VMF scaduti.

3.1.6 Class PazienteHome

Schermata principale del sistema. Qui viene mostrata la lista dei farmaci presenti nel VMF scaduti.

3.1.7 Class Rubrica

Visualizzazione della rubrica contenente tutti i relativi dottori/pazienti aggiunti.

3.1.8 Class RicercaDottore

Permette la ricerca di un dottore tra tutti quelli registrati a mobilFarm.

3.1.9 Class ProfiloDottore

Schermata di riepilogo delle informazioni che il dottore ha inserito all'atto della registrazione. Alcune di esse possono essere modificate come il numero di telefono oppure le informazioni relative alla posizione dello studio medico.

3.1.10 Class SchedaPaziente

Scheda riassuntiva del paziente selezionato dal dottore dalla propria rubrica: in questo caso il dottore può decidere di rimuovere il paziente dalla propria rubrica. E' possibile vedere quali piani terapeutici ha seguito quel paziente e quale medico li ha prescritti.

3.1.11 Class ProfiloPaziente

Schermata di riepilogo delle informazioni che il paziente ha inserito all'atto della registrazione. Alcune di esse possono essere modificate come il numero di telefono. Inoltre qui è possibile inserire i numeri di emergenza per il servizio S.O.S.

3.1.12 Class SchedaDottore

Scheda riassuntiva del dottore selezionato dal paziente:

- durante una ricerca: in questo caso il paziente può decidere di aggiungere il dottore alla propria rubrica;
- selezionato dalla propria rubrica: in questo caso il paziente può decidere di rimuovere il dottore dalla propria rubrica.

3.1.13 Class ModalitaSOS

Permette l'invio di una richiesta di soccorso composta da:

- Una chiamata al servizio di emergenza;
- L'invio di un messaggio ai numeri di emergenza inseriti nella sezione del profilo contenente la posizione di dove è stato inviato l'S.O.S.

3.1.14 Class GestioneAccount

Questa classe, di tipo final, è il core di GestoreAccountClient e offre servizi relativi alla gestione dell'account a tutte le altre classi del package.
In questo modo separiamo la specifica gestione dell'interfaccia (obbligatoria per lo specifico sistema) rispetto ai servizi che vengono offerti dalla nostra piattaforma, al fine di massimizzare il riuso in vista di un futuro sviluppo su altri sistemi operativi.

3.1.15 Class GestioneRubrica

Questa classe, di tipo final, offre tutti i servizi relativi alla gestione della rubrica.

3.2 GestoreFarmacoClient

3.2.1 Class VMF

Permette la visualizzazione del proprio VirtualMobilFarm dove è possibile aggiungere ed eliminare farmaci.

3.2.2 Class FarmacoInScadenza

Gestisce la scadenza dei vari farmaci presenti nel VMF indicando nella Home di ogni utente i farmaci più prossimi alla scadenza.

3.2.3 Class RicercaFarmaco

Permetta la ricerca di un farmaco tra tutti quelli presenti nel database.

3.2.4 Class NuovoFarmaco

Permette ad un dottore di poter inserire un nuovo farmaco nel database così da tenerlo sempre aggiornato in caso del lancio di un nuovo farmaco sul mercato.

3.2.5 Class GestioneFarmaco

Questa classe, di tipo final, è il core di GestoreFarmacoClient e offre servizi relativi alla gestione dei farmaci a tutte le altre classi del package.
In questo modo separiamo la specifica gestione dell'interfaccia (obbligatoria per lo specifico sistema) rispetto ai servizi che vengono offerti dalla nostra piattaforma, al fine di massimizzare il riuso in vista di un futuro sviluppo su altri sistemi operativi.

3.2.6 Class GestioneVMF

Questa classe, di tipo final, offre tutti i servizi relativi al VirtualMobilFarm.

3.2.7 Class ListaFarmaciInScadenzaAdapter

Adapter ottimizzato per la visualizzazione dei farmaci in scadenza.

3.2.8 Class ListaFarmaciVMFAdapter

Adapter ottimizzato per la visualizzazione dei farmaci nel VMF dell'utente

3.3 GestorePianoClient

3.3.1 Class PianiTerapeutici

Visualizza tutti i piani terapeutici del paziente loggato o del paziente selezionato dal dottore dalla sua rubrica.

3.3.2 Class PianoTerapeutico

Permette la visualizzazione di tutti i farmaci relativi al piano terapeutico.
L'utente dottore che ha scritto il piano è abilitato alla modifica dei farmaci e della relativa posologia.

3.3.3 Class FarmacoPianoTerapeutico

Questa classe gestisce il fragment FarmacoPianoTerapeutico, permettendo la visualizzazione di tutte le informazioni relative all'assunzione di un determinato farmaco al fine di conseguire in maniera corretta il piano terapeutico.
Se l'utente che visualizza il fragment è un Dottore, i campi saranno modificabili.

3.3.4 Class FarmacoDaAssumere

Permette la visualizzazione dei farmaci da somministrare nella giornata.

3.3.5 Class GestionePianoTerapeutico

Questa classe, di tipo final, è il core di GestorePianoClient e offre servizi relativi alla gestione dei piani terapeutici a tutte le altre classi del package.
In questo modo separiamo la specifica gestione dell'interfaccia (obbligatoria per lo specifico sistema) rispetto ai servizi che vengono offerti dalla nostra piattaforma, al fine di massimizzare il riuso in vista di un futuro sviluppo su altri sistemi operativi.

3.3.6 Class ListaFarmaciPianoAdapter

Adapter ottimizzato per la visualizzazione dei farmaci inerenti ad un piano terapeutico.

3.3.7 Class ScegliFarmacoPianoAdapter

Adapter ottimizzato relativo alla ricerca di un farmaco per l'inserimento in piano terapeutico.

3.4 UtilityClient

3.4.1 Class mobilFarmServer

Tutte le classi core dei diversi Package utilizzeranno questa classe al fine di connettersi, inviare e ricevere dati dal server.
Ciò permetterà di separare l'impostazione dei vari servizi offerti rispetto all'infrastruttura di network del sistema mobilFarm.

3.4.2 Class FontsOvverride

Questa classe permetterà di sovrascrivere il font di sistema con un font selezionato da noi e assente dalla lista dei font di Android.

3.4.3 Class UIManager

Al fine di avere una UI coerente in tutta l'applicazione sarà presente questa classe che ci permetterà di impostare delle proprietà grafiche di default.

3.4.4 Class ListaDottoriAdapter

Adapter ottimizzato per la visualizzazione dei dottori nella rubrica del paziente.

3.4.5 Class ListaFarmaciAdapter

Adapter ottimizzato per la visualizzazione dei farmaci quando vengono ricercati per l'aggiunta al VMF.

3.4.6 Class ListaPazientiAdapter

Adapter ottimizzato per la visualizzazione dei dottori nella rubrica del paziente.

3.4.7 Class ListaPianiTerapeuticiAdapter

Adapter ottimizzato per la visualizzazione della lista dei piani terapeutici.

3.4.8 Class NotificationManager

Classe final che prepara le notifiche all'avvio di mobilFarm.

3.4.9 Class NotificationMessage

Classe che estende BroadcastReciver e gestisce la singola notifica.

3.4.10 Class TimePickerFragment

Classe che richiama il widget proprietario di Android per la scelta dell'ora.

3.4.11 Class ActionNotAuthorizedException

Eccezione lanciata quando non si possiede l'autorizzazione per una determinata operazione.

3.4.12 Class ConnectionErrorException

Eccezione lanciata quando non si è connessi ad Internet.

3.4.13 Class ErrorValueException

Eccezione lanciata dalle classi "Gestione" quando il formato o la lunghezza dei dati inseriti non sono corretti.

3.4.14 Class ResponseErrorException

Eccezione lanciata quando ci sono problemi al server.

3.5 GestoreAccountServer

3.5.1 Class AccountManager

Questa classe php si occupa di ricevere le richieste di gestione dell'account inerenti al sottosistema GestoreAccount provenienti dal Package GestoreAccountClient tramite mobilFarmServer. Gestirà la lettura, scrittura, modifica e eliminazione delle informazioni inerenti alle tabelle Utente, Dottore, Paziente presenti nel DBMS MySql.

3.5.2 Class RubricaManager

Questa classe php si occupa di ricevere le richieste di gestione della rubrica inerenti al sottosistema GestoreAccount provenienti dal Package GestoreAccountClient tramite mobilFarmServer. Gestirà la lettura, scrittura, modifica e eliminazione delle informazioni inerenti alla tabella Seguire del DBMS MySql.

3.6 GestorePianoServer

3.6.1 Class PianoTerapeuticoManager

Questa classe php si occupa di ricevere tutte le richieste inerenti al sottosistema GestorePiano provenienti dal Package GestorePianoClient tramite mobilFarmServer.
Gestirà la lettura, scrittura, modifica e eliminazione dei dati inerenti alle tabelle PianiTerapeutici, Prescrizioni presenti nel DBMS MySql.

3.7 GestoreFarmacoServer

3.7.1 Class FarmacoManager

Questa classe php si occupa di ricevere le richieste relative alla creazione di un nuovo farmaco e alla sua ricerca provenienti dal Package GestoreFarmacoClient tramite mobilFarmServer. Gestirà la lettura, scrittura, modifica e eliminazione dei dati inerenti alla tabella Farmaci presenti nel DBMS MySql.

3.7.2 Class VMFMAnager

Questa classe php si occupa di ricevere le richieste relative e alla gestione del VMF provenienti dal Package GestoreFarmacoClient tramite mobilFarmServer. Gestirà la lettura, scrittura, modifica e eliminazione dei dati inerenti alla tabella VMF presenti nel DBMS MySql.