

DESIGN DOCUMENT

Farisano Gino

Esposito Vincenzo

Scovotto Vanessa

Valitutto Giuseppe

Indice

1.Scopo del Sistema.....	4
1.1 Obiettivi di design.....	4
1.1.1 Performance Criteria	4
1.1.2 Dependability Criteria.....	4
1.1.3 Maintenance Criteria.....	5
1.1.4 End-user Criteria.....	5
2.Architettura proposta.....	6
2.1 Decomposizione del sistema	7
2.1.1 FullDentInterfaceLayer	8
2.1.2 FullDentApplicationLayer	8
2.1.3 FullDentStorageLayer	9
2.1.4 Mapping HW-SW	9
2.1.5 Deployment diagram (Smartphone Android e client pc)	9
2.1.6 FullDentInterfaceLayer	10
2.1.7 FullDentApplicationLayer	11
2.1.8 interfaceLayer.GestioneUtente	12
2.1.8 interfaceLayer.GestioneAppuntamenti.....	12
2.1.9 applicationLayer.GestioneUtente.....	13
2.1.10 applicationLayer.GestioneAppuntamenti	13
3.Gestione dei dati persistenti	14
3.1 Controllo degli accessi e sicurezza.....	14
3.1.1 Controllo degli accessi	14
3.1.2 Sicurezza	15
Controllo del flusso.....	15
Condizioni limite	15
3.1.3 Servizi dei sottosistemi	17
4.Compromessi del sistema.....	19
4.1 Sicurezza vs Efficienza	19
4.2 Interfaccia vs Usabilità.....	19
4.3 Comprensibilità vs Tempo	19
4.4 Linee guida per la documentazione	19
4.4.1 Tool da utilizzare.....	19
4.4.2 Naming convention	20

4.4.3 Packages	20
Introduzione	20
Descrizione	20
FullDentInterfaceLayer	20
FullDentApplicationLayer	20
FullDentStorageLayer	20
4.4.4 Comunicazione tra layer.....	21
5. Interfaccia delle classi.....	22
5.1 Bean (Lato client).....	22
Interfaccia delle classi.....	23
5.2 (Lato client Smartphone Android)	23
6.Sequence Diagram di basso livello (Client PC e dispositivo Android)	24
6.1 Inserimento Appuntamento	24
6.2 Elimina Appuntamento.....	25
6.3 Inserimento Appuntamento.....	26

1.Scopo del Sistema

L'obiettivo del progetto è quello di risolvere i problemi legati alla gestione delle prenotazioni di un appuntamento in un centro odontoiatrico. Il sistema software presenta dei vincoli importanti che devono essere rispettati nella fase di design del sistema, fra questi troviamo vincoli sulla sicurezza del sistema, vincoli di architettura ed altri che sono specificati nel seguito del documento. Nel documento di design si ripropone di fronteggiare i vari problemi e vincoli del sistema riscontrati fornendo un'architettura vantaggiosa.

1.1 Obiettivi di design

Gli obiettivi di design identificati rispecchiano quattro tipologie di categorie ben distinte: Performance criteria, Dependability criteria, Maintenance criteria, End user criteria.

1.1.1 Performance Criteria

Tempo di risposta	Il sistema garantisce l'esecuzione di una richiesta in un periodo di tempo inferiore ai 30 secondi.
-------------------	---

1.1.2 Dependability Criteria

Robustezza	L'utente che userà FULLDENT deve accedere al sistema attraverso un nome utente e una password. Questi dati vengono inviati all'utente dalla segreteria, nel caso in cui l'utente inserisce un nome utente diverso, il sistema visualizzerà un messaggio di errore.
Affidabilità	In caso di errori relativi al funzionamento del sistema, all'utente deve essere mostrato una pagina in cui si specificano i dettagli dell'errore.
Disponibilità	Il sistema è disponibile all'utente 24 ore su 24, per ogni sua esigenza.
Sicurezza & Safety	L'accesso all'applicazione avviene tramite la sottomissione di nome utente e password.

1.1.3 Maintenance Criteria

Estensibilità	Il sistema software è stato diviso in moduli semi-indipendenti che lo rendono facilmente comprensibile ed estendibile per l'aggiunta di nuove funzionalità.
Modificabilità	Grazie al manuale che definisce nei dettagli ogni funzionalità del sistema, è possibile apportare delle modifiche senza particolari difficoltà.
Leggibilità	La progettazione modulare del software, unita all'aggiunta di commenti, aiuteranno gli sviluppatori nella comprensione del codice.
Tracciabilità dei requisiti	La tracciabilità dei requisiti a partire dal codice è relativamente semplice per via della modularità delle componenti che costituiscono il prodotto software.

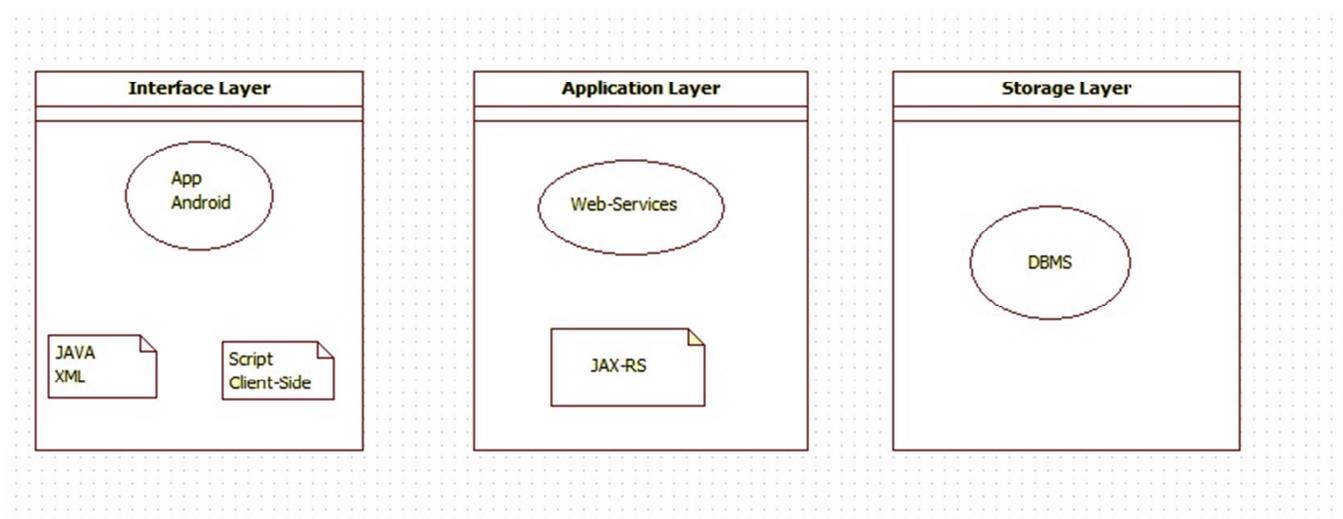
1.1.4 End-user Criteria

Usabilità	Il sistema deve garantire un uso intuitivo di tutte le funzionalità chiave offerte per la gestione della struttura.
-----------	---

2. Architettura proposta

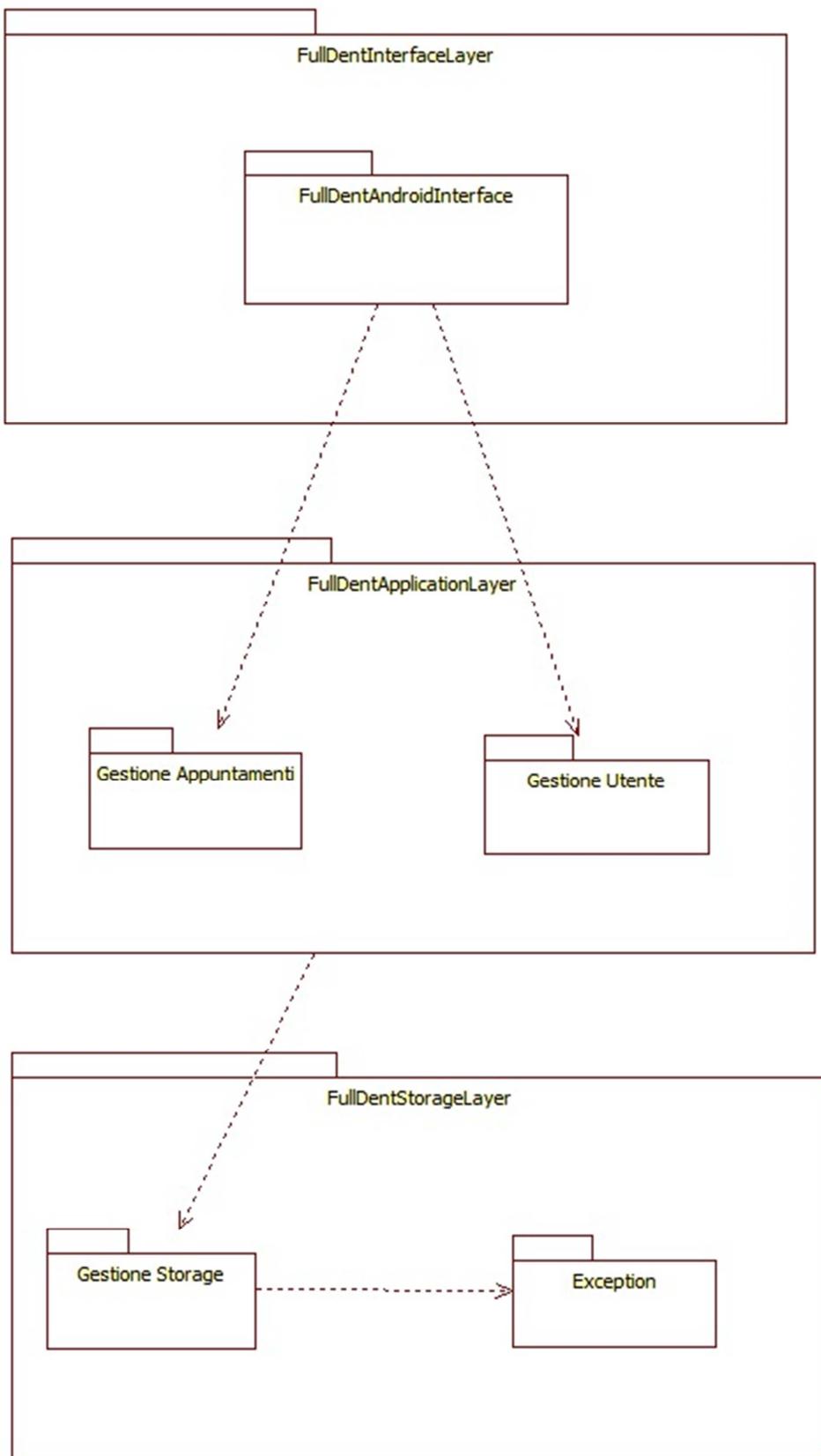
L'architettura scelta per lo sviluppo, a differenza del sistema precedente, è di tipo client-server. Il server tiene traccia di tutti i dati del sistema. Il client fa uso dell'interfaccia del dispositivo mobile per poter interagire con il server ed avere accesso ai dati (Gestione Prenotazioni, Visualizza storico dell'utente).

L'applicazione mobile FULLDENT è sviluppata su tre livelli logici-funzionali (applicazione Three-Tier).



- **Interface layer:** rappresenta l'interfaccia utente di FULLDENT, costituita da form e bottoni che permettono di inserire, inviare e visualizzare i dati. Essa è costituita da varie tecnologie combinate tra loro: Java, XML e controlli tramite l'utilizzo di script. A differenza del sistema precedente, che era un'applicazione stand-alone, qui l'interfaccia cambia radicalmente essendo un'applicazione mobile sviluppata per Android.
- **Application layer:** elabora i dati e le richieste dell'utente. Le elaborazioni a livello application generano i risultati richiesti dall'utente. JAX-RS svolge interamente questo ruolo.
- **Storage layer:** il DBMS gestisce la base di dati in cui vengono memorizzate tutte le informazioni del sistema (eventi, utenti, strutture). Rispetto al sistema precedente, lo storage layer è memorizzato su un server-web, inoltre è stato modificato, per permettere la gestione di alcune nuove funzionalità (Registrazione Utenti).

2.1 Decomposizione del sistema



Il sistema è suddiviso in tre layer:

- FullDentInterfaceLayer
- FullDentApplicationLayer
- FullDentStorageLayer

L'utente interagisce con il sistema tramite la componente FullDentAndroidInterface, collocata nel FullDentInterfaceLayer, rappresentante l'interfaccia utente. FullDentInterfaceLayer interagisce con i moduli presenti in FullDentApplicationLayer, quali GestioneUtente, GestioneAppuntamenti, fornendo le funzionalità richieste. Infine, il FullDentStorageLayer si interfaccia con i servizi offerti dal DBMS, utilizzando le componenti JDBC.

Diamo ora una breve descrizione per ogni layer:

2.1.1 FullDentInterfaceLayer

FullDentInterface	Modulo rappresentante l'interfaccia grafica con cui l'utente interagisce. Tramite essa può inviare richieste al sistema e visualizzare i risultati per ogni richiesta
-------------------	---

2.1.2 FullDentApplicationLayer

GestioneUtente	Modulo che si occupa di gestire le operazioni di Registrazione utente.
GestioneAppuntamenti	Modulo che si occupa di gestire le operazioni di inserimento, cancellazione, visualizzazione di un prenotazione.

2.1.3 FullDentStorageLayer

GestioneStorage	Modulo che si interpone tra il database e il FullDentApplicationLayer, che permette di minimizzare l'accoppiamento, fungendo da interfaccia che consente di effettuare operazioni sul database
Exception	Modulo per la gestione delle eccezioni generate dal sistema

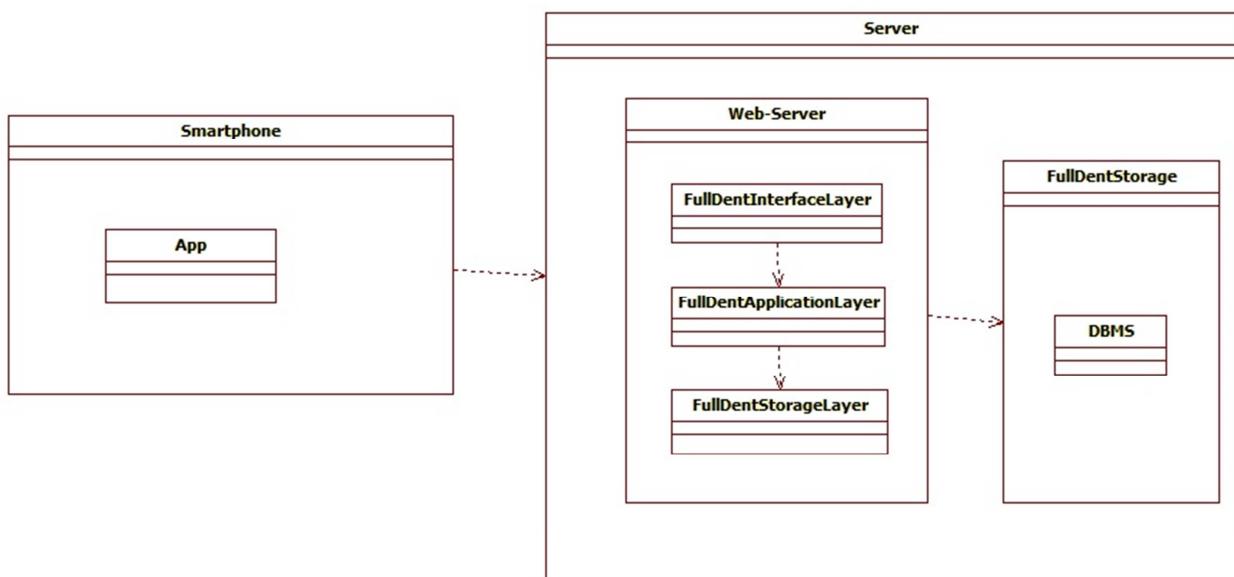
2.1.4 Mapping HW-SW

Il web-Server utilizzato è Apache Tomcat. Il FullDentInterfaceLayer è sul server, e tutte le sue componenti sono visualizzabili attraverso l'utilizzo di un'interfaccia su un dispositivo mobile (lato client).

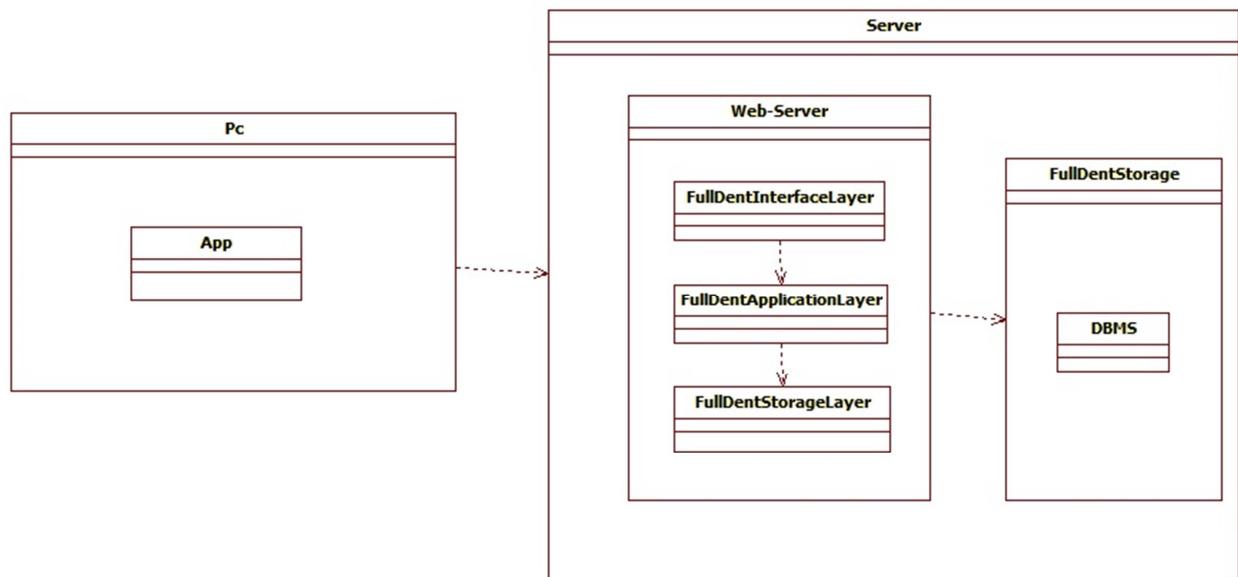
Il FullDentApplicationLayer implementa la logica di controllo del sistema, e funge da tramite tra il FullDentInterfaceLayer e FullDentStorageLayer. Quest'ultimo risulta composto da classi scritte in Java. Tramite l'utilizzo del driver JDBC, è possibile accedere al database memorizzato su di un altro nodo del sistema. Il database è gestito con MySQL, un DBMS relazionale open-source, leggero e che supporta l'accesso concorrente ai dati.

Il seguente deployment diagram ci mostra il mapping hardware-software di FullDent, modificato rispetto al sistema precedente.

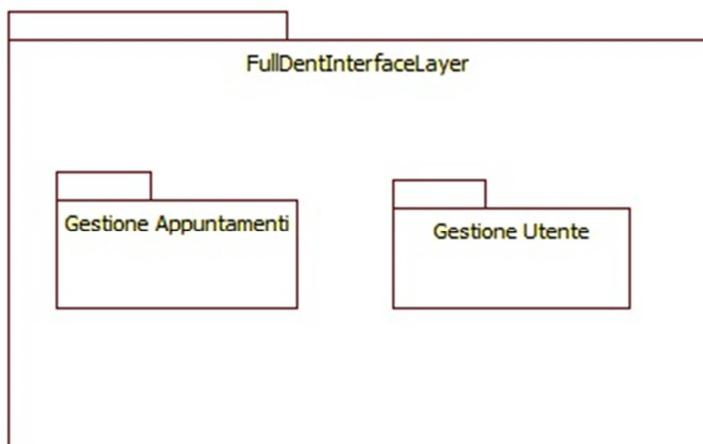
2.1.5 Deployment diagram (Smartphone Android e client pc)



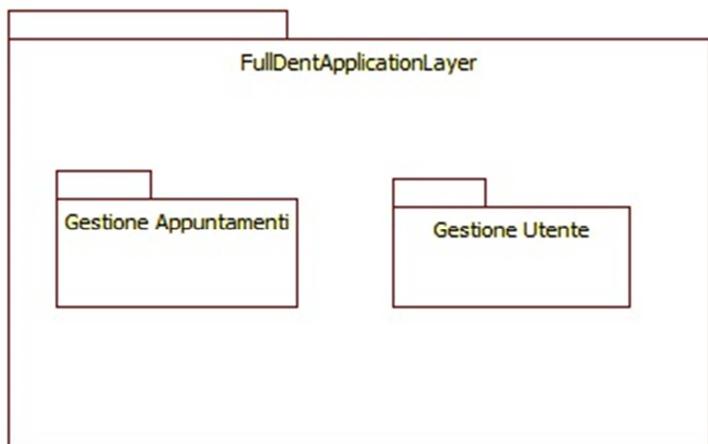
(Personal Computer)



2.1.6 FullDentInterfaceLayer

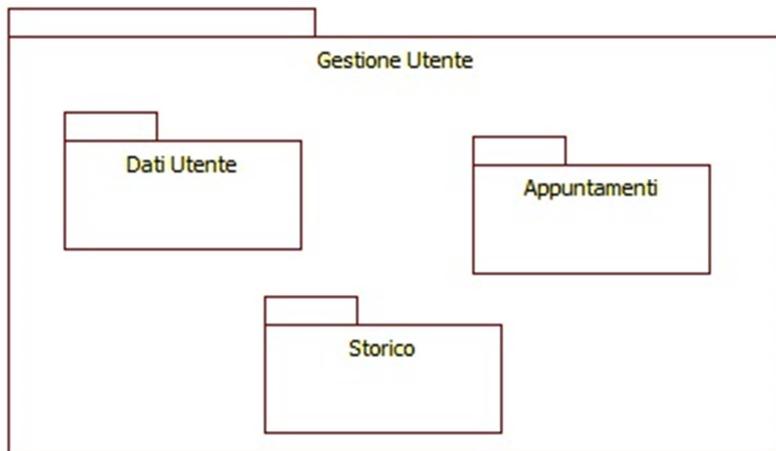


2.1.7 FullDentApplicationLayer

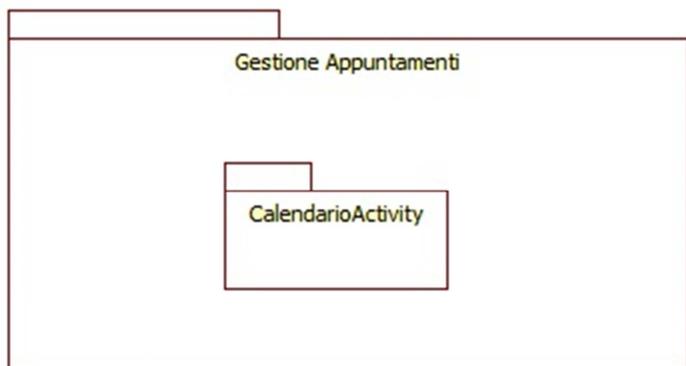


Andiamo ora a visualizzare in dettaglio i diversi moduli individuati

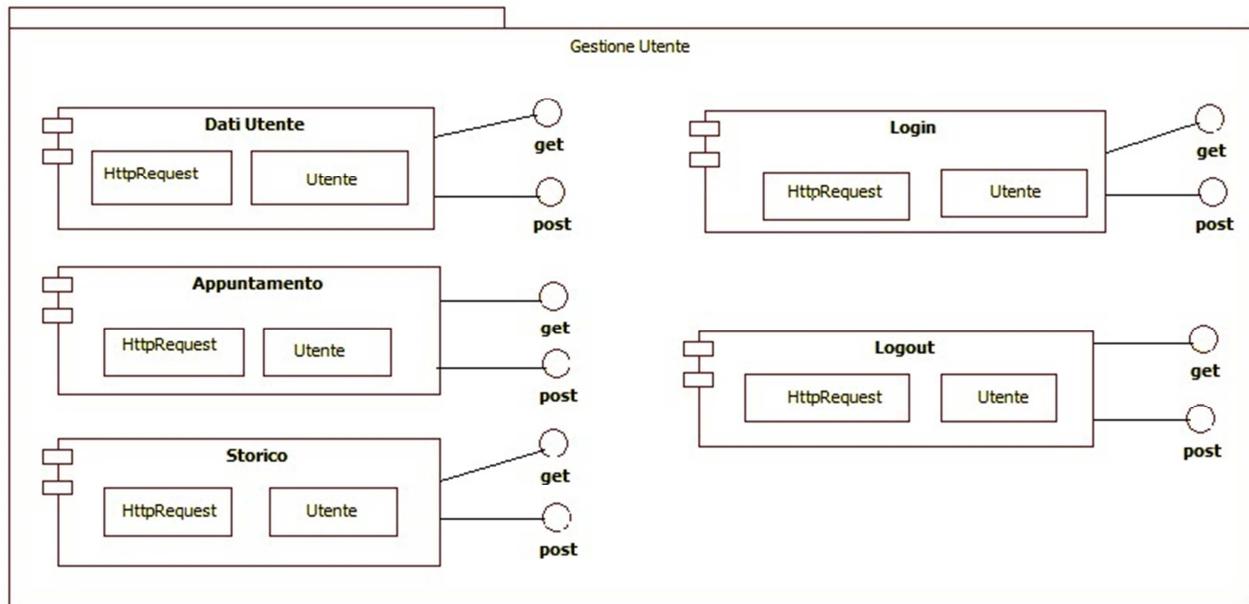
2.1.8 interfaceLayer.GestioneUtente



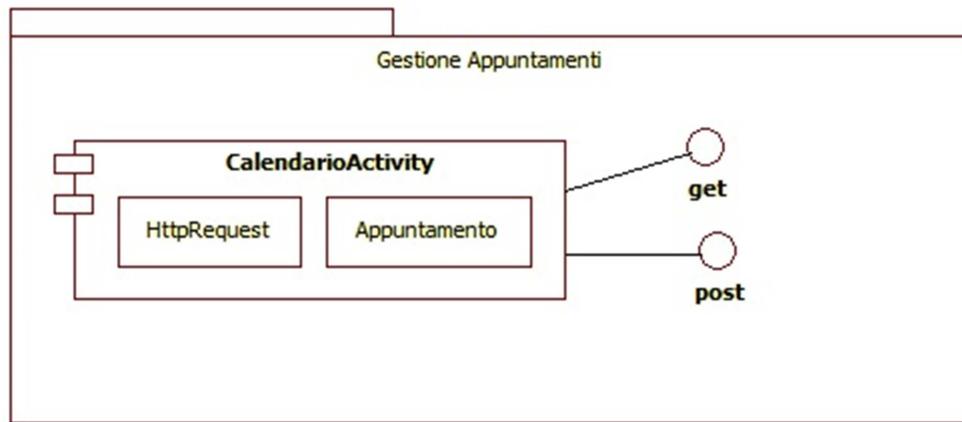
2.1.8 interfaceLayer.GestioneAppuntamenti



2.1.9 applicationLayer.GestioneUtente



2.1.10 applicationLayer.GestioneAppuntamenti



3. Gestione dei dati persistenti

Gestire dati persistenti equivale a gestire dati la cui esistenza deve essere garantita in ogni esecuzione del sistema. La documentazione prodotta in fase di analisi ha permesso l'individuazione dei seguenti dati da memorizzare in modo persistente:

- Dati utenti
- Dati Prenotazioni

Per memorizzare tali dati e rispettare gli obiettivi di design, si è scelto di utilizzare un database relazionale (MySQL). Infatti una **base di dati relazionale serve numerosi utenti**: tale condivisione dei dati riduce le possibili ridondanze e migliora nettamente la coerenza dei dati, contribuendo così a mantenerne l'integrità. Le operazioni sui dati e sui relativi attributi vengono realizzate attraverso **interrogazioni (query)** utilizzando un linguaggio formalizzato standard: il linguaggio SQL. La struttura fondamentale del modello relazionale è appunto la "relazione", cioè una tabella costituita da righe (tuple) e colonne (attributi). Per permettere l'interazione tra il database relazionale MySQL e il linguaggio di programmazione Java, viene fatto utilizzo dell'API di JDBC.

3.1 Controllo degli accessi e sicurezza

3.1.1 Controllo degli accessi

FullDent può essere utilizzato da due tipi di figure: dalla segretaria e dall'utente.

1) **Segretaria**: è la persona che si occupa dell'inserimento di un nuovo utente nel centro, della gestione delle prenotazioni. Per poter avere tali privilegi è necessario autenticarsi come segretaria del sistema.

2) **Utente**: è colui che ha la funzione di loggarsi, modificare i propri dati, prenotare un appuntamento visualizzare le proprie prenotazioni e visualizzare il proprio storico.

3.1.2 Sicurezza

In un sistema informatico bisogna fare delle differenze su cosa può e cosa non può fare un attore, dato che non può avere libero accesso ad ogni oggetto del sistema. L'accesso di un sistema deve essere gestito per evitare malfunzionamenti critici.

Oggetto → Attore ↓	Gestione Utente	Gestione Prenotazioni
Segretaria	<<insert>> <<modify>> <<search>> <<viewStorico>>	<<insert>> <<search>> <<delete>>
Utente Registrato	<<login>> <<modify>> <<viewStorico>> <<viewAppuntamento>> >	<<insert>> <<search>>

L'utente non registrato non può accedere al sistema.

Legenda

<<insert>>	Inserisce una prenotazione o un nuovo utente
<<viewAppuntamento>>	Visualizza appuntamento
<<delete>>	Cancella una prenotazione
<<modify>>	Modifica un oggetto esistente
<<login>>	Logga l'utente al sistema
<<search>>	Cerca un'appuntamento o una utente registrato
<<viewStorico>>	Visualizza lo storico di un utente

Controllo del flusso

Il controllo del flusso viene gestito interamente da classi Java che interagiscono con il client. Quando il client fa una richiesta, una funzione si prende carico di inizializzarla e di inviarla alle funzioni preposte. Il server elabora la risposta e la invia al client, tramite apposite funzioni. Il tutto viene gestito con un event-driven control.

Condizioni limite

Le condizioni limite riguardano sia il server che il client. Lato client le condizioni limite evidenziano gli errori che possono essere riscontrati durante la connessione al server. Lato

server, i casi limite fanno riferimento all'avvio, allo spegnimento del sistema e al riavvio. Nel caso dell'avvio del sistema, il sistema mette in attesa, inizializzando tutti quei servizi che devono essere resi disponibili in remoto. Nel caso dello spegnimento del sistema, il server rende indisponibili i servizi in remoto. Entrambe i casi, invece, sono presenti quando si riavvia il sistema.

Casi d'uso:

Nome: Startup del sistema	
Attori: Amministratore	
Descrizione: l'utente effettua il riavvio del server	
Precondizioni: l'utente accede al sistema	
Sequenza degli eventi	
Utente	Sistema
1. Richiede l'avvio del server	2. Mette a disposizione tutti i servizi in remoto e avvia il server
Postcondizioni: i servizi servizi sono resi disponibili al client in remoto. Il sistema invia una notifica all'amministratore	

Nome Shutdown del server	
Attori: Amministratore	
Descrizione: l'utente effettua l'arresto del server	
Precondizioni: l'utente accede al sistema	
Sequenza degli eventi	
Utente	Sistema
1. Richiede l'arresto del server	2. Rimuove i servizi resi disponibili in remoto e arresta il server
Postcondizioni: I servizi sono resi disponibili al client in remoto. Il sistema invia una notifica all'amministratore	

3.1.3 Servizi dei sottosistemi

Sottosistema	GestioneUtente
Descrizione	Sottosistema che gestisce le operazioni che riguardano l'utente
Servizi offerti	
Login	Permette all'utente di effettuare l'autenticazione all'applicazione
Logout	Permette all'utente loggato di effettuare il logout dal sistema qualora decida di disconnettersi
Dati Utente	Permette all'utente registrato di modificare i propri dati personali, forniti in fase di registrazione
Storico	Permette all'utente registrato di visualizzare lo storico dei suoi servizi.
Appuntamento	Permette all'utente registrato di visualizzare un appuntamento

Sottosistema	GestioneAppuntamenti
Descrizione	Sottosistema che gestisce le operazioni che riguardano gli appuntamenti
Servizi offerti	
Inserisci appuntamento	Permette all'utente e la segretaria di inserire un nuovo appuntamento
Cancella appuntamento	Permette alla segretaria di cancellare un appuntamento
Cerca appuntamento	Permette all'utente o alla segretaria di cercare un'appuntamento

4.Compromessi del sistema

Le scelte progettuali imposte ci hanno spinto a decidere di utilizzare componenti già esistenti per gestire alcune delle funzionalità più complesse del sistema. Verrà quindi utilizzata la componente open-source JDBC, per la gestione dell'accesso al database

4.1 Sicurezza vs Efficienza

La sicurezza rappresenta un aspetto importante del sistema. Noi abbiamo sviluppato un sistema di sicurezza basato su username e password forniti dalla segretaria, garantendo efficienza al sistema, a discapito di una migliore efficienza, essendo che gli algoritmi crittografici sono onerosi in termini di efficienza. Il sistema precedente, in quanto un'applicazione stand-alone, in termini di efficienza, era fortemente dipendente dalla macchina su cui girava. Ora, data la natura mobile dell'applicazione, l'efficienza dipende fortemente dalla ripartizione del carico di elaborazione tra client e server.

4.2 Interfaccia vs Usabilità

Mantenere alto il grado di facilità e soddisfazione con cui l'utente è in grado di interagire con il software ha sempre rappresentato una parte essenziale nello sviluppo del sistema. Infatti, FullDent propone un'interfaccia intuitiva, adatta anche a coloro che ritengono essere alle prime armi nell'utilizzo di strumentazioni di questo tipo.

4.3 Comprensibilità vs Tempo

Una buona norma, durante la stesura, sarà di rendere il codice comprensibile e di facile lettura. Tale norma sarà assicurata corredando il codice di commenti e evitando ambiguità nella scelta di nomi per classi ed oggetti; quindi classi e metodi devono essere di facile lettura anche ad utenti che non hanno partecipato alla realizzazione del sistema.

I commenti invece svolgono un ruolo fondamentale nella fase di "rivisitazione" del codice, in quanto possono facilitare di molto la comprensione di passaggi apparentemente oscuri. Tuttavia l'uso frequente di commenti incide negativamente sulla rapidità di sviluppo del sistema.

4.4 Linee guida per la documentazione

4.4.1 Tool da utilizzare

Come ambiente di sviluppo, come nel sistema precedente, verrà utilizzato Eclipse (www.eclipse.org).

4.4.2 Naming convention

Gli sviluppatori dovranno seguire le seguenti linee guida per la stesura delle interfacce:

1. I nomi assegnati alle classi, metodi, variabili costanti e vari parametri sono stati dichiarati in lingua italiana a favore di una migliore leggibilità del codice;
2. E' preferibile avere nomi lunghi ma esplicativi;
3. I nomi delle classi devono iniziare con una lettera maiuscola;
4. I nomi dei metodi devono iniziare con una lettera minuscola;
5. I nomi delle variabili devono essere scritti in minuscolo;
6. I nomi delle costanti devono essere scritte in maiuscolo e si utilizza l'underscore per separare parole diverse.

4.4.3 Packages

Introduzione

Il codice è stato suddiviso in tre layer. I layer application e storage sono suddivisi a loro volta in package.

Descrizione

FullDentInterfaceLayer

- **FullDentInterface:** layer contenente tutto il codice riguardante l'interfaccia grafica del sistema, con la quale interagirerà l'utente.

FullDentApplicationLayer

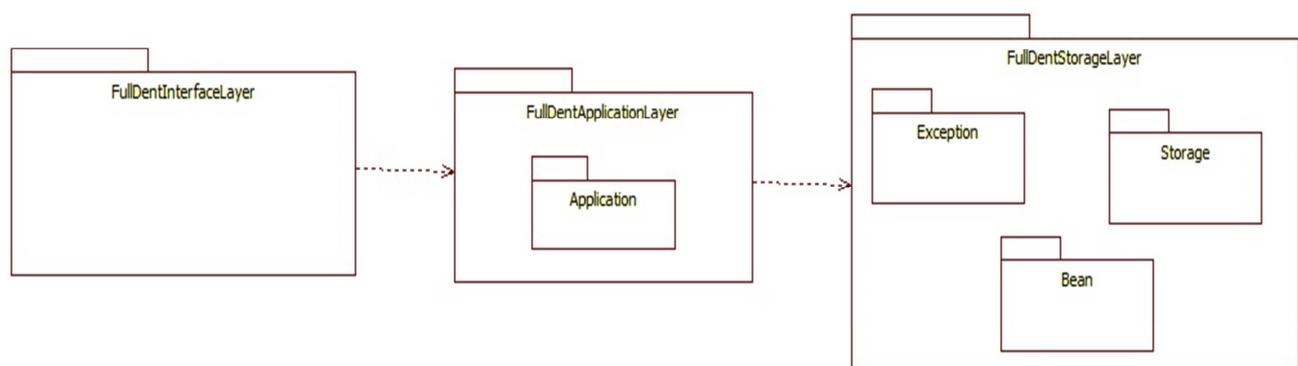
- **Application:** package contenente tutte le classi utilizzate per le operazioni riguardanti la gestione dell'utente, gestione degli appuntamenti.

FullDentStorageLayer

- **Storage:** package contenente tutte le classi che si occupano di interagire con il database, fornendo metodi appropriati per la manipolazione dei record memorizzati;
- **Bean:** package contenente tutte le classi che rappresentano le entità del sistema;
- **Exception:** package contenente tutte le classi che rappresentano le eccezioni che il sistema può generare.

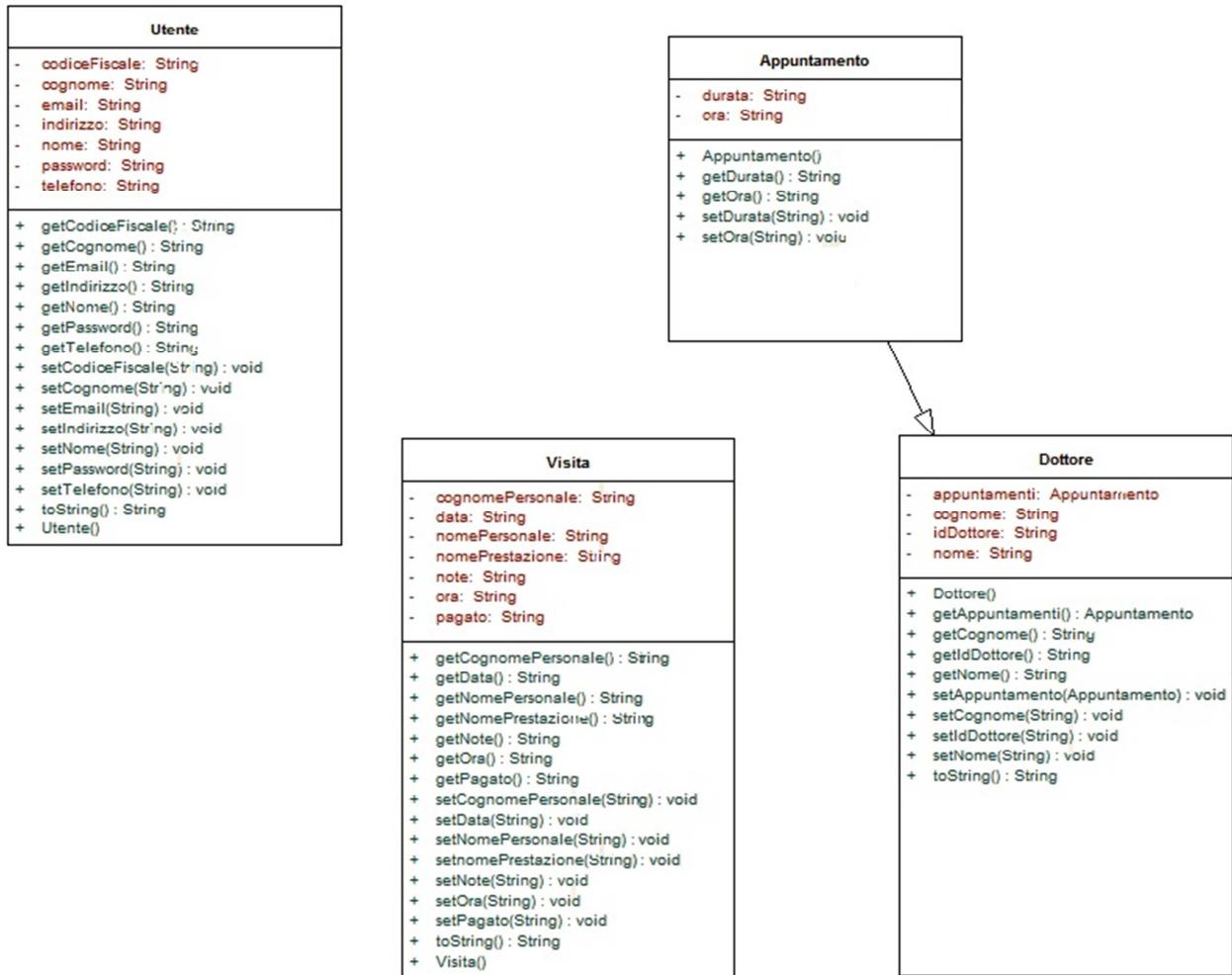
4.4.4 Comunicazione tra layer

La figura mostra la comunicazione tra i vari oggetti che compongono il sistema al lato client utilizzando un **PC**. Allo stesso modo avviene anche tramite lo smartphone.



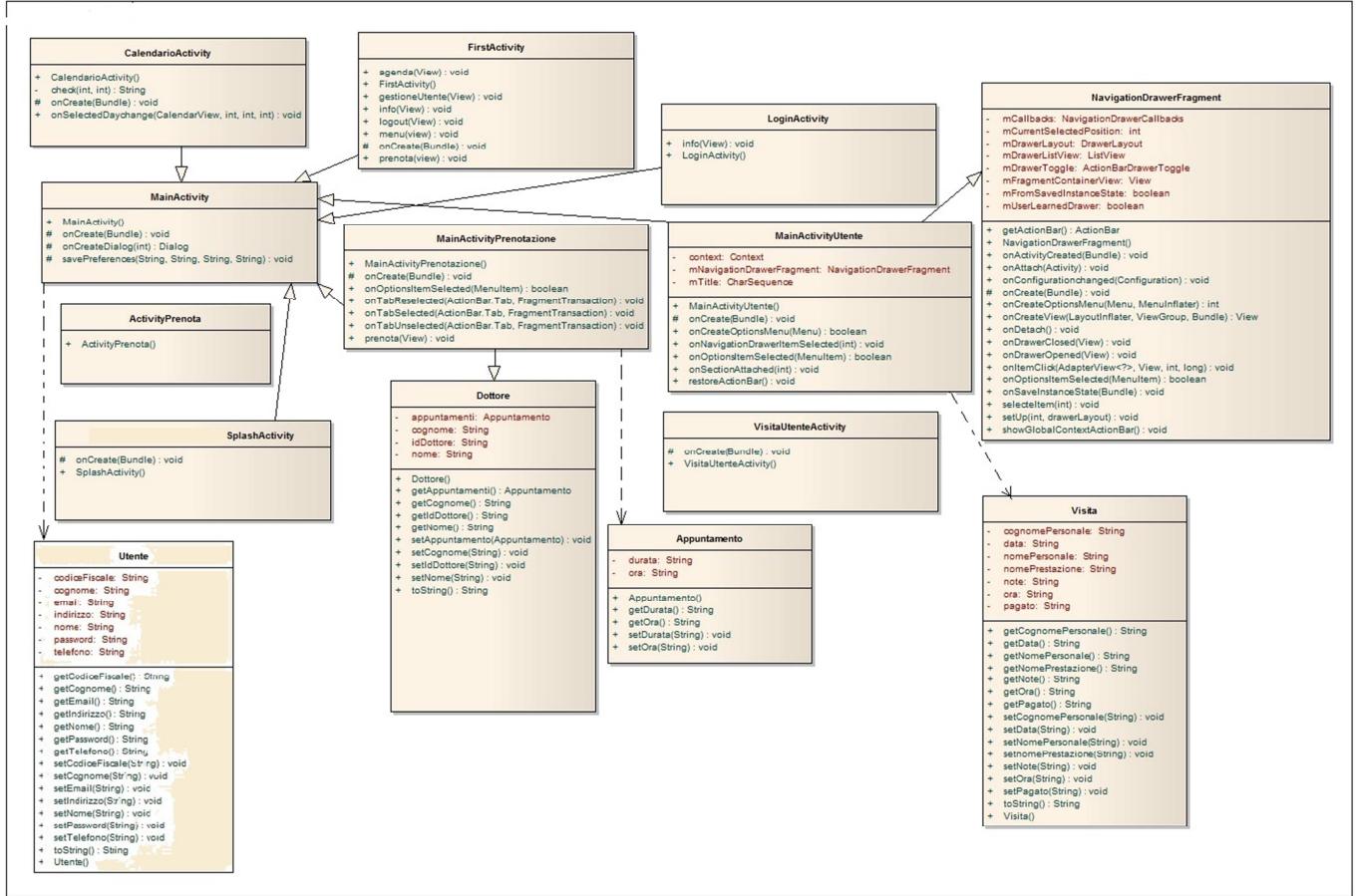
5. Interfaccia delle classi

5.1 Bean (Lato client)



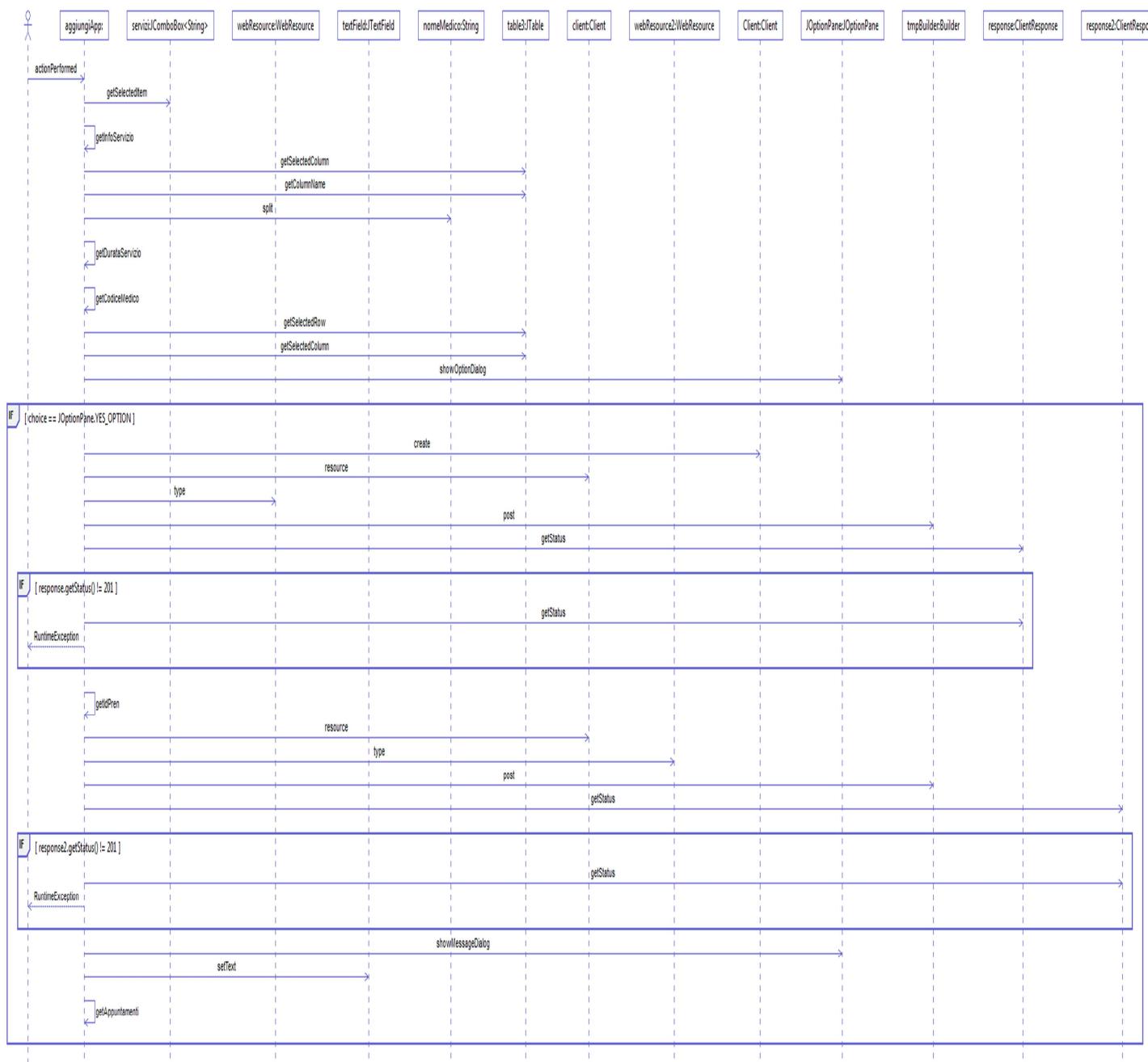
Interfaccia delle classi

5.2 (Lato client Smartphone Android)

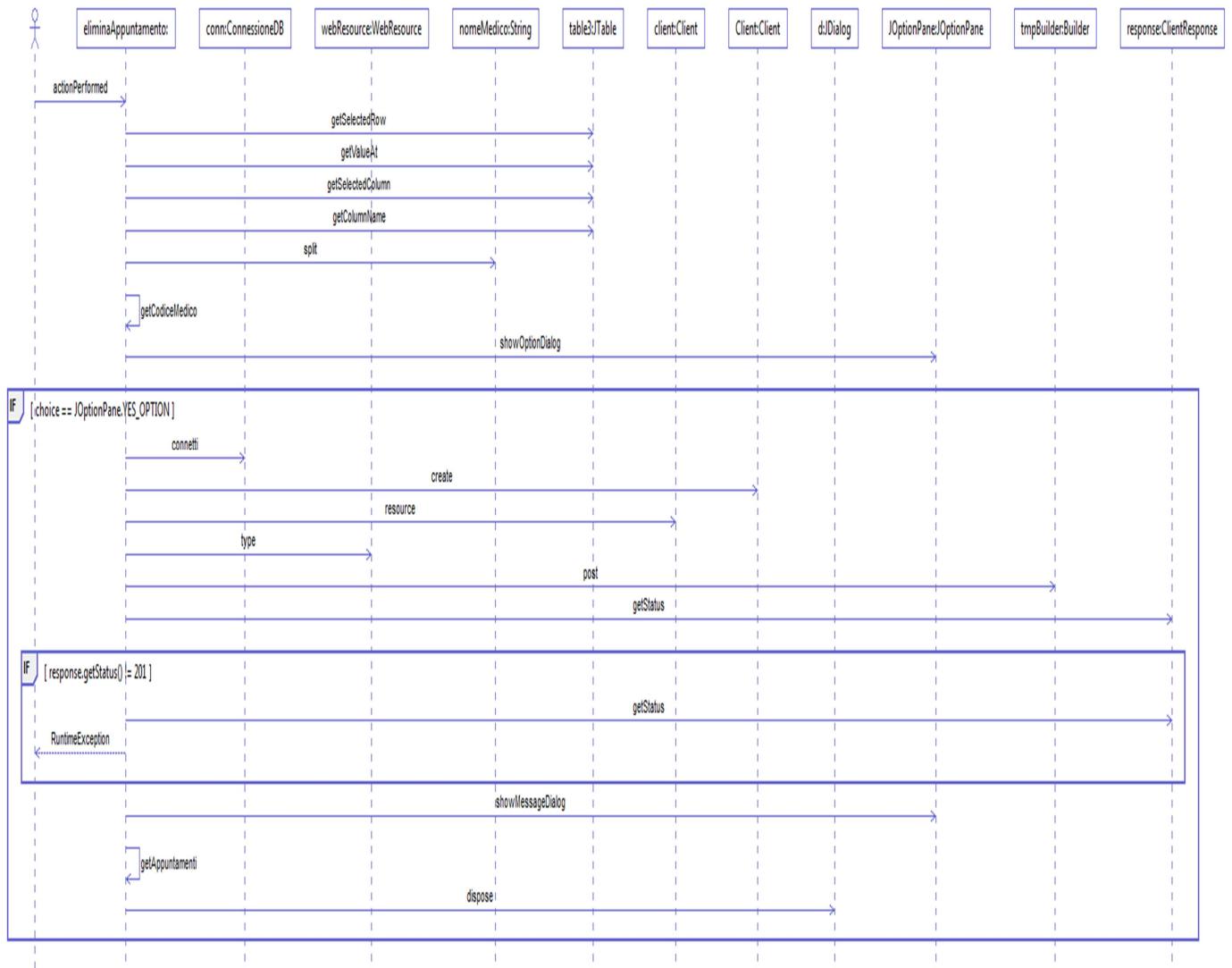


6.Sequence Diagram di basso livello (Client PC e dispositivo Android)

6.1 Inserimento Appuntamento



6.2 Elimina Appuntamento



6.3 Inserimento Appuntamento

