

## **Gestione ospedaliera**

Giacomo Bergami - Paolo de Luca - Fabian  
Priftaj - Matej Torok



## Indice

<b>Parte 1. Documento Primo</b>	<b>7</b>
Capitolo 1. Piano di processo	9
1.1. Capitolato	10
1.2. Individuazione dei compiti (Risorse umane)	10
1.3. Scadenze	10
1.4. Valutazione dello sforzo	11
1.5. Analisi dei rischi	13
1.6. Diario	14
<b>Parte 2. Documento Secondo</b>	<b>17</b>
Capitolo 2. Analisi dei requisiti	19
2.1. Premessa	19
2.2. Incontro con il cliente	20
2.2.1. Chiarimenti sulle specifiche	20
2.2.2. Requisiti non funzionali	23
2.3. Casi d'uso	25
2.3.1. Diagramma dei Casi d'Uso	25
2.3.2. Testo dei Casi d'Uso	25
2.4. Glossario	33
2.5. Diario	34
Capitolo 3. Revisioni delle stime precedenti	37
3.1. Aggiunta di Tool di sviluppo	37
3.2. Revisione della valutazione dello sforzo	37
3.2.1. Valutazione dell'analisi dello sforzo con gli Use Case	37
3.2.2. Revisione dei computi effettuati precedentemente	38
<b>Parte 3. Documento Terzo</b>	<b>39</b>
Capitolo 4. Analisi I	41
4.1. Premessa	41
4.2. CRC cards	42
4.3. Modello di dominio	42
4.4. Documento dell'architettura software	44

4.4.1. Amministratore	44
4.4.1.1. Vista dei Dati	44
4.4.1.2. Vista dei Casi d'Uso	45
4.4.2. Utente	52
4.4.2.1. Vista dei Dati	52
4.4.2.2. Vista dei Casi d'Uso	52
4.4.3. Ampliamenti al modello di dominio suggeriti	60
4.4.4. Definizione delle priorità dei casi d'uso	60
4.5. Diario	62
Capitolo 5. Strutturazione dell'interfaccia grafica	63
5.1. Analisi del Client	63
5.1.1. Interfaccia di avvio	63
5.1.2. Gestione delle prenotazioni	66
5.1.3. Prenotazione della Visita	66
5.2. Analisi del Server	69
Capitolo 6. Revisioni delle stime precedenti	71
6.1. Ulteriore estensione dei Casi d'Uso	71
<b>Parte 4. Documento Quarto</b>	<b>73</b>
Capitolo 7. Sviluppo I	75
7.1. Premessa	75
7.2. Raffinamento del Modello di Dominio	76
7.2.1. Premesse	76
7.2.1.1. Strutturazione dei campi Messaggio	76
7.2.1.2. Implicazioni dell'utilizzo del framework "EJP"	77
7.2.1.3. Applicazione di design pattern	77
7.2.2. Progettazione agli oggetti	78
7.2.2.1. Utente: Vista dei Casi d'Uso	78
7.2.2.2. Amministratore: Vista dei Casi d'Uso	81
7.2.2.3. Vista dei Dati	87
Capitolo 8. Progettazione della base di dati	91
8.1. Analisi dei requisiti	91
8.2. Progettazione concettuale	91
8.3. Progettazione Logica	92
8.3.1. Eliminazione delle generalizzazioni	92
8.4. Traduzione Logica	92
8.5. Installazione del database	98
8.6. Utilizzo del Framework EJP	98
<b>Parte 5. Documento Quinto</b>	<b>101</b>

Capitolo 9. Valutazione finale	103
9.1. Differenze di implementazione	103
9.1.1. Database	103
9.1.2. Amministratore	105
9.1.3. Modifica del modello di business	105
9.2. Valutazione del prodotto	108
9.2.1. Diario	108
9.2.2. Testing	108
9.2.3. Revisione dello sforzo del progetto	113
9.2.4. Analisi di qualità	114
9.2.5. Valutazione del ruolo produttivo	114



## **Parte 1**

# **Documento Primo**





## CAPITOLO 1

### Piano di processo

#### Indice

1.1. Capitolato	10
1.2. Individuazione dei compiti (Risorse umane)	10
1.3. Scadenze	10
1.4. Valutazione dello sforzo	11
1.5. Analisi dei rischi	13
1.6. Diario	14

L'**obiettivo** di questo progetto è quello di sviluppare l'applicazione OTH-OPEDIATRICS per la gestione delle prenotazioni di un sistema infermieristico, che potrà essere utilizzata in un qualunque sistema che supporti una *Java Virtual Machine* (JVM). Lo **scopo** di questa documentazione è quella di dettagliare, nel caso specifico, il *Piano di processo*; i documenti successivi ancora da redigere forniranno ulteriori dettagli sulle idee di progettazione sviluppate dal team.

Per quanto concerne le **risorse** da impegnare, il gruppo dispone unicamente del tempo ed i componenti del team: questi ultimi in particolare contribuiscono concretamente allo sviluppo grazie alla prima risorsa. Riscontriamo che la risorsa tempo è quella maggiormente critica, in quanto dobbiamo portare a termine contemporaneamente altri progetti in collaborazione con altre aziende. Poiché questa è la nostra prima esperienza con il linguaggio Java, non disponiamo di codice riutilizzabile che potrebbe velocizzare i tempi di sviluppo. Elenchiamo qui sotto quali strumenti software disponiamo per lo sviluppo del progetto:

- *jbTeX* per la compilazione semiautomatica dei file *LaTeX*.
- *TeXLive* e tutte le librerie annesse.
- ECLIPSE per lo sviluppo di codice Java, ed in particolare la libreria SWING per la grafica.
- ARGOUML per la generazione dei diagrammi *UML*.
- LIBREOFFICE CALC per effettuare calcoli in genere.
- Utilizzo di un wiki WIKISPOT privato assieme ad uno spazio web per la gestione del versioning.

Utilizziamo SKYPE come software di teleconferenza. Si consideri che altri tool, non elencati qui, potranno essere considerati in fase di sviluppo. ALLEGATI:

`analisi_rischi.ods`. Contiene il computo per la valutazione dello sforzo.

### 1.1. Capitolato

I reparti ortopedia e pediatria di un ente ospedaliero hanno bisogno di una applicazione che gestisse le prenotazioni dei pazienti. Il programma sarà utilizzato da pazienti e amministratori che avranno maggiori permessi di manipolazione sulle prenotazioni.

Le possibili operazioni per gli amministratori dovranno essere:

- Autenticazione
- Scelta del reparto
- Definire prenotazioni ad alta priorità
- Visualizzare ed aggiornare referti medici
- Visualizzare lo storico prenotazioni del paziente, e le date disponibili.
- Stampa

Le possibili operazioni per i pazienti dovranno essere:

- Registrazione/ Autenticazione
- Scelta del reparto
- Effettuare una prenotazione
- Visualizzare lo storico prenotazioni e date disponibili
- Stampa

### 1.2. Individuazione dei compiti (Risorse umane)

All'interno di questa fase, si assegnano i ruoli ai seguenti componenti del team XYZT

**Project Manager:** Matej Torok

**Quality Manager:** Giacomo Bergami

**Tool Specialist:** Paolo de Luca

**Developer:** Fabian Priftaj

Il committente di questo progetto è stato il gruppo "CrySoft".

### 1.3. Scadenze

In quanto questa è la nostra prima esperienza di sviluppo di programmazione in ambito ospedaliero, non possediamo uno storico atto a prefissare una scadenza entro la quale stabilire delle milestone(s): poiché nella programmazione extreme programming, si è abituati a seguire passo passo l'evoluzione del progetto, punteremo man mano ad aggiornare il cliente ed effettuare piccoli rilasci successivi di codice funzionante.

Elenchiamo qui sotto le scadenze:

**Piano di Processo:** 21-3-2012

**Analisi dei Requisiti:** 30-03-2012

Unadjusted Function Point (UFP)	Semplice	Medio	Arduo
<i>Input (EI)</i>	3	4	6
<i>Output (EO)</i>	4	5	7
<i>Interrogazioni (EQ)</i>	3	4	6
<i>File logici interni (ILF)</i>	5	7	10
<i>File logici esterni (EIF)</i>	7	10	15

TABELLA 1. Definizione degli Unadjusted Function Point con definizione dei pesi in base al grado di difficoltà.

Unadjusted Function Point (UFP)	Tipologia	Quantità
<i>Input (EI)</i>	Semplice	2
<i>Input (EI)</i>	Media	6
<i>Output (EO)</i>	Semplice	3
<i>Output (EO)</i>	Media	2
<i>Interrogazioni (EQ)</i>	Semplice	8
<i>File logici interni (ILF)</i>	-	0
<i>File logici esterni (EIF)</i>	Semplice	1
Totale		75

TABELLA 2. Definizione degli Unadjusted Function Point con definizione dei pesi in base al grado di difficoltà.

**Revisione al cliente dell'Analisi dei requisiti:** 13-04-2012

**Rilascio del prodotto al cliente e relazione finale:** 25-05-2012

Per il momento è noto che il cliente potrà valutare il prodotto entro il **31 Maggio 2012**.

#### 1.4. Valutazione dello sforzo

Abbiamo la necessità di effettuare una prima valutazione approssimativa dello sforzo che ci verrà richiesto ai fini del svolgimento di questo progetto: effettuiamo quindi una stima dei **function point**, tramite i *Unadjusted Function Point* (UFP) definiti come seguono:

- ◇ Input (EI)
- ◇ Interrogazioni (EQ)
- ◇ Output (EO)
- ◇ File logici interni (ILF)
- ◇ File logici esterni (EIF)

VAF	Valutazione scala 1-5
F1	2
F2	2
F3	4
F4	1
F5	0
F6	5
F7	3
F8	5
F9	1
F10	0
F11	3
F12	0
F13	0
F14	0
Totale	26

TABELLA 3. *Definizione degli Unadjusted Function Point con definizione dei pesi in base al grado di difficoltà.*

In Tabella 1 nella pagina precedente riportiamo la valutazione dei pesi assegnati ai precedenti capisaldi in base alla loro difficoltà<sup>1</sup>. In Tabella 2 nella pagina precedente riportiamo conseguentemente i valori da noi utilizzati per effettuare la nostra stima: in quanto non possediamo ancora informazioni precise rispetto alle esigenze del cliente in merito alla navigabilità e l'usabilità del prodotto, ci basiamo unicamente sulle informazioni deducibili dal capitolato pervenutoci. Per ogni *Value Adjustment Factors* (VAF), la cui lista viene proposta qui sotto, viene attribuito un *Degree of Influence* (DI), ottenendo la situazione riportata nella Tabella 3.

- (1) Il sistema ha bisogno di operazioni di backup e di ripristino affidabili?
- (2) È necessario utilizzare comunicazioni specializzate per trasferire informazioni da o verso l'applicazione?
- (3) Esistono funzioni di elaborazione distribuite?
- (4) Le prestazioni rappresentano un fattore critico?
- (5) Il sistema dovrà funzionare in un ambiente preesistente, pesantemente utilizzato?
- (6) Il sistema richiede di inserire on-line dei dati?
- (7) Il sistema on-line richiede una transazione di input costituita da più schermate od operazioni?

<sup>1</sup><http://www.devshed.com/c/a/Practices/An-Overview-of-Function-Point-Analysis/3/>

- (8) I files IFL vengono aggiornati on-line?
- (9) Esistono input/output, file od interrogazioni complesse?
- (10) L'elaborazione interna è complessa?
- (11) Il codice è progettato per essere riusabile?
- (12) Nel progetto sono presenti la conversione e l'installazione?
- (13) Il progetto è stato concepito per essere installato in più organizzazioni?
- (14) L'applicazione è progettata per facilitare la modifica e l'usabilità da parte dei clienti?

Possiamo in seguito effettuare il seguente computo:

$$FP = conteggio - UFP \cdot (0.65 + 0.01 \cdot \sum_{i=1}^{14} F_i)$$

Ottenendo quindi

$$FP = 75 \cdot (0.65 + 0.01 \cdot 26) = 68.25$$

Considerando inoltre che il fattore di conversione FP/LOC in base a Java, che è il linguaggio di implementazione del prodotto, è  $fploc_{Java} = 63$  come valore medio<sup>2</sup>, otterremo il seguente valore in LOC

$$LOC = 63 \cdot 66 = 4300$$

Utilizzando inoltre la formula di *Bailey-Basili*<sup>3</sup> per il calcolo dello sforzo, che è

$$E_{bb} = 5,5 + 0,7 \cdot KLOC^{1,16} \text{ mesi/persona}$$

Otteniamo

$$E_{bb} = 5.5 + 0.7 \cdot 4,3^{1,16} \simeq 9.30 \text{ mesi/persona}$$

Bisogna notare che siamo giunti a questo risultato non conoscendo nulla sulle esigenze del cliente, e quindi non conoscendo le user stories, come imposto dall'XP. Ci rimandiamo pertanto alla discussione diretta con il cliente per effettuare un miglioramento della stima iniziale. Confidiamo nel fatto che, quella fornita, sia una sovrastima allo sforzo complessivo che ci verrà richiesto, e che riusciremo quindi meglio a valutare solo in seguito all'analisi delle *user stories* che ci dovranno essere fornite dai clienti.

### 1.5. Analisi dei rischi

A questo punto del progetto, non possiamo che fornire un'analisi molto generica dei rischi dello stesso.

Il team è coeso e motivato a portare a termine il progetto, conseguentemente non ci dovrebbero essere né problemi di coesione con il team, né di relazioni

<sup>2</sup>R.S. Pressman: "Principi di Ingegneria del Software". McGraw-Hill editore.

<sup>3</sup>R.S. Pressman: op. cit.

personali fra le componenti del gruppo, in quanto tutti i membri del team hanno affrontato precedentemente altre collaborazioni.

Inoltre, in quanto XP si basa sulle tecniche di **planning game**, non ci dovrebbero essere particolari problemi di scheduling, in quanto giorno per giorno verrà stabilito il da farsi.

I rischi che si possono riscontrare ora sono i due seguenti:

**Ambiente di sviluppo:** Non tutti i componenti del gruppo conoscono IDE di sviluppo quali *Eclipse*: sarà quindi necessario un iniziale costo sulla formazione del personale nella conoscenza dei tool da utilizzare. Tuttavia, le esperienze precedenti di programmazione e la conoscenza di altri linguaggi di programmazione ObjectOriented, dovrebbero semplificare la fase di apprendimento.

**Cliente:** La maggiore incognita a questo punto del progetto consta proprio nella figura del Cliente, che non abbiamo ancora avuto l'occasione di incontrare. In questo caso il rischio è dovuto all'impossibilità di valutare i seguenti capisaldi:

- ◊ Qual è il grado di sofisticazione del cliente, in quale modo si è in grado di comunicare con lo stesso, e con quale tempestività?
- ◊ I requisiti sono stati ben compresi dal team di sviluppo e dai clienti?
- ◊ Gli utenti finali hanno attese realistiche?

## 1.6. Diario

**Giovedì 8 Marzo:** L'intero gruppo si è riunito nel pomeriggio per discutere la pianificazione dei giorni successivi. Durante l'incontro si è discusso per:

- (1) l'assegnamento dei ruoli come precedentemente definito;
- (2) valutazione delle componenti in gioco nell'analisi dei rischi e sul calcolo dello sforzo;
- (3) identificazione dei primi dubbi sulle specifiche da chiarire con il cliente. (verranno dettagliati con dovizia di particolari nella fase successiva del progetto);
- (4) identificazione cenni all'utilizzo dei tool;
- (5) organizzazione sulle modalità di contatto per reperire i clienti (a noi non familiari)
- (6) pianificazione del successivo meeting;
- (7) prima stesura del documento.

Totale ore di lavoro: 6

**Venerdì 9 Marzo:** Durante la sera il gruppo si è riunito "virtualmente" su Skype. Durante l'incontro si è discusso per:

- (1) la presentazione della prima stesura del *Piano di processo* da parte del Quality Manager: da questa è nata un'accesa discussione riguardo i Value Adjustment Factors ed in seguito si è valutata la miglior formula per il calcolo dei FP;
- (2) le ipotetiche richieste del cliente come qualità, diversificazione e semplicità di input; si prevede un primo incontro con il cliente per il dì **23 Marzo 2012**
- (3) la definizione dei tool, e raffinamento delle scadenze, già precedentemente discusse.
- (4) seconda ed ultima stesura del documento;

Totale ore di lavoro: 6





**Parte 2**

**Documento Secondo**



## CAPITOLO 2

### Analisi dei requisiti

#### Indice

2.1. Premessa	19
2.2. Incontro con il cliente	20
2.2.1. Chiarimenti sulle specifiche	20
2.2.2. Requisiti non funzionali	23
2.3. Casi d'uso	25
2.3.1. Diagramma dei Casi d'Uso	25
2.3.2. Testo dei Casi d'Uso	25
2.4. Glossario	33
2.5. Diario	34

#### 2.1. Premessa

La fase di sviluppo si strutturerà come segue:

◇ *Definizione dei diagrammi dei Casi d'Uso*

Per questa fase si mira ad effettuare una prima analisi dei casi d'uso deducibili dal capitolato fornitoci dal cliente. In seguito all'incontro con il cliente, possiamo dedurre i requisiti funzionali o/e comportamentali del sistema, che mostrano quali sono i compiti base.

◇ *Definizione del Testo dei Casi d'Uso*

Una volta effettuati i chiarimenti sulle specifiche con il cliente (vedi Sezione 2.2 nella pagina successiva), si vogliono ottenere gli scenari di interazione con il sistema. Inoltre, per meglio venire in contro alle esigenze dei clienti, svilupperemo queste ultime in uno **stile concreto**<sup>1</sup>. Associamo inoltre i seguenti documenti:

– *Glossario*: (v. Sezione 2.4 a pagina 33).

– *Revisione delle stime precedenti* (v. Capitolo 3 a pagina 37).

Successivamente verranno generate, all'interno della terza fase del progetto, i seguenti documenti:

◇ *Definizione degli Scenari di Interazione con il sistema*

◇ *Definizione del Modello di Progetto*

---

<sup>1</sup>C. Larman: Applicare UML e i Pattern. Terza edizione, edizioni Pearson Education Italia

## 2.2. Incontro con il cliente

**2.2.1. Chiarimenti sulle specifiche.** Riportiamo sotto forma di dialogo i chiarimenti sulle specifiche.

**Q:** Gli amministratori sono predefiniti, oppure bisogna crearli tramite registrazione?

**R:** Le credenziali degli amministratori devono già essere cablate all'interno dell'applicativo, in modo da procedere unicamente con l'autenticazione. Per credenziali si intende la coppia di informazioni NOME UTENTE e PASSWORD: non deve essere quindi possibile la creazione di nuovi utenti amministratori.

**Q:** Bisogna dividere interfaccia utente da interfaccia amministratore?

**R:** Quando si entra nell'applicazione, è possibile o accedere in modalità cliente, nella quale è possibile o effettuare un log-in, o effettuare una registrazione, oppure accedere tramite log-in o come amministratore di pediatria, o come amministratore di ortopedia.

**Q:** Cosa può fare un amministratore una volta autenticato?

**R:** L'amministratore deve convalidare le richieste dei pazienti, creando la nuova prenotazione. Inoltre l'amministratore può spostare o annullare prenotazioni già convalidate, su richiesta del paziente stesso. Ogni assegnamento di un appuntamento verrà notificato al paziente.

**Q:** Deve esistere un super-utente con massimo privilegio, in grado di convalidare le richieste da parte di clienti o amministratori?

**R:** Gli unici due utenti ad avere massimo privilegio devono essere i due amministratori del reparto di pediatria e di ortopedia: non esistono altri amministratori se non questi già predefiniti.

**Q:** Cosa si intende con "maggiori permessi di manipolazione sulle prenotazioni"?

**R:** Mentre agli utenti è permesso effettuare le prenotazioni, agli amministratori è permesso modificarle.

**Q:** Cosa si intende (in dettaglio) con "definire prenotazioni ad alta priorità"?

**R:** Si definiscono tre livelli di priorità: una bassa priorità (codice verde), una media (codice giallo) ed una alta (codice rosso). La semantica di questi termini è conosciuta agli amministratori, e non si richiede che l'applicazione conosca l'effettiva caratterizzazione della stessa.

**Q:** L'utente può avere accesso al proprio referto medico?

**R:** Sì, tramite la specifica richiesta di "visualizzare lo storico delle prenotazioni", è possibile accedere anche ai referti medici associati alle visite.

**Q:** Cosa si intende con “scelta del reparto” da parte del cliente o dell’amministratore?

**R:** Il cliente sceglie il reparto nel quale verrà visitato. L’amministratore invece non sceglie il reparto, in quanto ad ogni amministratore è associato uno ed un solo reparto di sua competenza

**Q:** È l’amministratore che decide la data della visita dietro richiesta del paziente? Si ha di fatti un conflitto con la richiesta: “il paziente visualizza lo storico prenotazioni e date disponibili”.

**R:** L’amministratore ha il potere di fissare le visite all’interno di un giorno e di un’ora prefissata in una data sala del reparto, in base alle prenotazioni pervenute dagli utenti ed in base al loro grado di priorità desunto, che viene fissato nella prenotazione.

**Q:** Quali informazioni definiscono una prenotazione?

**R:** Una prenotazione deve contenere le informazioni riguardanti la DATA, l’ORARIO, la PRIORITÀ, il TIPO DI VISITA (definita dal reparto), la STANZA ed il MEDICO.

**Q:** Le visite hanno durata variabile o fissa?

**R:** Le visite sono tutte della stessa durata: un’ ora.

**Q:** Il sistema deve organizzare la gestione dei medici? O sono già presenti come gli amministratori?

**R:** No, il sistema non deve gestire i medici perché non sono di competenza del vostro sistema informativo (già disponiamo di un sistema per i turni dei medici): l’amministratore dovrà comunque assegnare un medico alla data prenotazione.

**Q:** Cosa può fare l’utente come paziente quando accede al sistema?

**R:** L’utente può visualizzare per un dato paziente (o sè stesso o il suo tutelato), le richieste di prenotazione e le prenotazioni già convalidate. Inoltre il paziente può annullare una prenotazione, o anche richiedere un posticipo.

**Q:** Come fa il paziente a ricevere la notifica della data fissata per la visita?

**R:** Sarà lo stesso amministratore che si occuperà di effettuare la notifica personalmente all’utente, tramite il sistema di notifica che dovrà essere implementato all’interno del sistema. Nota che anche i pazienti potranno usufruire dello stesso servizio per contattare eventualmente l’amministrazione.

**Q:** Con questa soluzione da voi proposta, non si potrebbe riscontrare starvation? Inoltre, quante prenotazioni possono essere gestite in un giorno?

**R:** Questa eventualità non si potrà mai presentare in quanto la nostra clinica riceve pochi pazienti, e gli slot orari proposti dovrebbero soddisfare le esigenze. Questa organizzazione si basa di fatti su un'organizzazione già adottata all'interno della clinica, e si dimostra efficace. Di fatti, nella clinica si ricevono al massimo 48 pazienti al giorno.

**Q:** Il singolo referto deve essere associato alla visita oppure al singolo paziente?

**R:** Ogni referto è associato ad ogni visita, propria di ciascun paziente. Quest'associazione avverrà a visita ultimata.

**Q:** Quali sono le informazioni associate ad un singolo cliente?

**R:** Queste sono NOME, COGNOME, INDIRIZZO, NUMERO DI TELEFONO, E-MAIL e DATA DI NASCITA.

**Q:** Può effettivamente effettuare una prenotazione un qualsiasi paziente minorenne? Con questo sistema infatti, non sembra che si specifichi il tutore del paziente in questione.

**R:** Di fatti, deve essere previsto un elemento tutore, il quale certifichi l'identità del paziente minorenne, che potrà accedere ad entrambi i reparti.

**Q:** Un tutore in particolare deve essere un paziente all'interno del sistema?

**R:** Un tutore può anche non essere un paziente del sistema: inoltre anche questo ha associato NOME, COGNOME, INDIRIZZO, NUMERO DI TELEFONO, E-MAIL e DATA DI NASCITA.

**Q:** Quindi se un tutore ha più tutelati, e vuole effettuare una prenotazione in favore di ognuno di essi, dovrà autenticarsi con le credenziali del tutelato e poi con le proprie, e questo per ogni tutelato?

**R:** Esattamente!

Possiamo quindi ottenere i seguenti requisiti funzionali scritte come **user stories**:

- ◇ Come Amministratore, voglio ricevere le richieste di visita in modo da poter notificare la prenotazione riservata per un paziente ed associarvi un grado di priorità.
- ◇ Come Amministratore, voglio gestire le prenotazioni in modo da associarvi o/e modificarvi i referti
- ◇ Come Amministratore, voglio visualizzare le prenotazioni in modo da decidere in quale data effettuare le prenotazioni
- ◇ Come Amministratore, voglio visualizzare le prenotazioni in modo da ottenere l'insieme delle prenotazioni relativo ad un paziente, ed eventualmente i referti.
- ◇ Come Amministratore, voglio poter modificare i reperti medici del paziente.

- ◇ Come Paziente maggiorenne e tutore, voglio ottenere l'autenticazione allo scopo di ottenere per me o per il mio tutelato i servizi del sistema.
- ◇ Come Utente, voglio essere in grado di registrare me stesso come Paziente Maggiorenne o come Tutore.
- ◇ Come Utente, voglio scegliere un reparto in modo da prenotare una visita associata a quel tipo di servizio medico.
- ◇ Come Utente, voglio essere in grado di visualizzare le mie prenotazioni ed eventualmente i referti associati.
- ◇ Come Utente, voglio ricevere le notifiche del sistema sulle operazioni di cancellazione, spostamento, prenotazione richieste all'amministratore.
- ◇ Come Utente, voglio essere in grado di effettuare l'eliminazione della prenotazione e di riceverne conferma.
- ◇ Come Utente, voglio essere in grado di effettuare uno spostamento della prenotazione già fissata.
- ◇ Come Tutore, voglio essere in grado di registrare un paziente minorenne di cui sono responsabile.

**2.2.2. Requisiti non funzionali.** Riportiamo ancora una volta sotto forma di dialogo l'incontro avvenuto con il cliente:

**Q:** Quali sono le cose da stampare (e dove stampare) da parte del cliente o dell'amministratore?

**R:** Ogni amministratore ha la possibilità di vedere e gestire unicamente le prenotazioni ed i referti medici del reparto di sua competenza. Inoltre, ogni cliente ha la possibilità di visualizzare tutte e solo le sue prenotazioni, alle quali seguono le visite mediche, alle quali sono associati dei referti medici. Questi ultimi, sono costituiti solamente da testo, che non segue una particolare formattazione. Il cliente è in grado di accedere allo storico delle prenotazioni e quindi delle visite. Con "stampa" si intendeva una generica visualizzazione dei dati di competenza di ciascun utente del sistema: mentre per l'amministratore si richiede unicamente la visualizzazione sullo "schermo" dei dati, per l'utente si richiede unicamente il salvataggio delle informazioni su file.

**Q:** Come distribuire i giorni per le prenotazioni di diversi livelli di priorità? In caso inoltre di giorni "vicini" non disponibili in quanto occupati da prenotazioni di bassa priorità, queste si possono spostare per dare spazio a quelli ad alta priorità? Qual è la durata in ore di ogni singola visita? quali sono i giorni in cui si possono effettuare le visite?

**R:** Ogni reparto ha due sale, e quindi all'interno di uno slot orario di una stessa giornata è possibile un'unica visita. Per ogni giorno lavorativo si rilevano i seguenti slot orari:

- Dalle 8 alle 12 e dalle 14 alle 18 si accettano le nuove prenotazioni, di qualunque priorità

- Dalle 12 alle 14 e dalle 18 alle 20 si possono slittare quelle prenotazioni non urgenti degli slot orari precedenti, che cedono la loro visita riservata in quello slot orario ad altre prenotazioni urgenti che sopraggiungono.
- Gli slot dalle 12 alle 14 e dalle 18 alle 20 non sono previsti di Sabato, di Domenica e nei giorni festivi: se alle prenotazioni a bassa o media priorità di quei giorni vengono privilegiate delle prenotazioni urgenti per quella data, le prime verranno posticipate nei giorni seguenti.

**Q:** Quali sono le credenziali che un utente deve fornire all'atto dell'autenticazione?

**R:** Ogni utente si deve loggare con il suo CODICE FISCALE e una PASSWORD. Ci sono due tipi di autorità che si possono loggare: amministratore e paziente; se quest'ultimo è minorenne, allora è necessaria la richiesta di autenticazione del suo tutore.

**Q:** Come avviene l'iterazione col sistema di un paziente minorenne?

**R:** Le prenotazioni per i pazienti minorenni possono essere fatte dal suo tutore. Innanzitutto il tutore accede con le credenziali del tutelato. Dopodiché le interazioni con il sistema sono tutte subordinate all'autenticazione del tutore attraverso le proprie credenziali.

**Q:** In particolare, come preferite che avvenga l'associazione tra tutore ed utente, in base alle vostre esigenze di sicurezza?

**R:** Vorremmo che, all'atto della registrazione di un paziente minorenne, siano richieste anche le credenziali del suo tutore. All'atto della gestione delle visite, come conferma, oltre all'iniziale fase di login, deve essere necessaria per ogni operazione una conferma, fornita tramite l'inserimento delle credenziali del Tutore.

Riportiamo nel punto elenco qui sotto sia i requisiti non funzionali forniti riportati dal dialogo evidenziabile sopra, sia dei nuovi requisiti che ci sono stati forniti esplicitamente, assieme ad eventuali **vincoli**.

- ◇ Il sistema informativo deve essere sviluppato in Java.
- ◇ Il sistema informativo deve poter sostenere il peso di 48 pazienti al giorno.
- ◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore, ed in più il paziente minorenne può accedere solamente mediante la supervisione del tutore; è richiesta autenticazione sia per il paziente minorenne, sia per il tutore.
- ◇ L'Utente deve essere in grado di "stampare" su file.
- ◇ L'Utente deve essere in grado di mandare un messaggio all'amministratore tramite lo stesso sistema di messaggi del punto precedente.



- ◊ Le credenziali dell'Amministratore devono essere cablate all'interno del sistema informativo.
- ◊ L'Amministratore deve essere in grado di "stampare" su interfaccia grafica.
- ◊ L'Amministratore deve essere in grado di notificare la prenotazione tramite un sistema di messaggi interno al sistema.
- ◊ L'Amministratore deve essere in grado di inserire i referti nel sistema tramite inserimento di testo.
- ◊ L'Amministratore deve poter visualizzare un calendario settimanale ove poter convalidare la prenotazione delle visite, e visualizzare gli slot orari già occupati.

### 2.3. Casi d'uso

**2.3.1. Diagramma dei Casi d'Uso.** Per il diagramma dei casi d'uso, si fa riferimento alla figura 2.2.1 nella pagina seguente. Questo è il frutto risultante dalla discussione con il cliente effettuata precedentemente nella Sottosezione 2.2.1 a pagina 20.

**2.3.2. Testo dei Casi d'Uso.** Dopo aver fornito al cliente il diagramma dei Casi d'Uso, e dopo esserci soffermati sulle richieste dei clienti «*concentrandoci sullo scopo*», abbiamo ideato alcuni possibili scenari d'interazione con il sistema, allo scopo di concretizzare maggiormente il problema e raccogliere le impressioni del cliente in merito alla soluzione<sup>2</sup> in via di proposizione. In particolare è interessante sapere:

- In quale modo si caratterizzerebbe un buon output generato da una soluzione di successo, o come si arriva, aggiungiamo, ad una soluzione di insuccesso?
- È possibile mostrare o descrivere l'ambiente operativo in cui verrà utilizzata la soluzione?

Per raccogliere questi requisiti funzionali, ci siamo interessati ad uno **stile concreto** per la scrittura dei casi d'uso<sup>3</sup>: di fatti ora «*le decisioni sull'interfaccia utente sono incorporate nel testo del caso d'uso*», per meglio comprendere come l'interazione dell'utente con il sistema possa soddisfare i desideri del committente.

---

<sup>2</sup>R.S. Pressman: "Principi di Ingegneria del Software". McGraw-Hill editore.

<sup>3</sup>C. Larman: op. cit.



- (2) L'utente inserisce Codice Fiscale e Password del paziente al quale sono associate le richieste di servizio.
- (3) Il sistema verifica la correttezza delle credenziali e le riconosce valide
- (4) Il sistema consente l'accesso e visualizza l'interfaccia di default.

**Estensioni:**    **Estensione A:**    3: Il sistema non riconosce come valide le credenziali del paziente.

4: Il sistema non consente l'accesso visualizzando un messaggio d'errore.

5: Il sistema ritorna al meccanismo di autenticazione.

**Estensione B:**    3: Il sistema riconosce il paziente come minorenne.

4: Il sistema informa l'Utente che deve fornire le credenziali di Tutore.

5: Il Tutore inserisce le sue credenziali di Codice Fiscale e Password.

6: Il sistema verifica la correttezza delle credenziali e le riconosce valide.

7: Il sistema consente l'accesso e visualizza l'interfaccia di default.

**Estensione C:**    **B:**    6: Il sistema verifica la correttezza delle credenziali e le riconosce valide.

7: Il sistema non riconosce come valide le credenziali del paziente.

8: Il sistema non consente l'accesso visualizzando un messaggio d'errore.

9: Il sistema ritorna al meccanismo di autenticazione.

**Requisiti non funzionali:**    ♦ Le credenziali dell'Amministratore devono essere cablate all'interno del sistema informativo.

---

**Nome del caso d'uso:** Registrare il paziente

**Attore primario:** L'utente (del sistema): il paziente maggiorenne od il tutore del paziente minorenne

**Parti interessate ed interessi:** L'utente: vuole registrare o sé stesso, o il paziente minorenne di cui ha tutela

**Pre-condizioni:** Nessuna

**Garanzia di successo:** Il sistema visualizzerà all'utente un messaggio di avvenuta registrazione

**Scenario principale di successo:** (1) L'utente sceglie l'opzione di registrazione

(2) L'utente inserisce i dati del paziente che dovrà ricevere le cure

- (3) L'utente richiede la registrazione
- (4) L'utente riceve la notifica di avvenuta registrazione con una notifica tramite il sistema

**Estensioni:**    **Estensione A:**    4: L'utente richiede la registrazione con successo in quanto è un paziente maggiorenne

**Estensione B:**    4: (1) Il sistema richiede all'utente l'associazione delle credenziali del tutore, in quanto il paziente in fase di registrazione è minorenne  
(2) Il tutore inserisce le proprie credenziali  
(3) Il sistema conferma la presenza del tutore all'interno del sistema  
(4) Il sistema effettua la registrazione del paziente minorenne e dell'associazione con il tutore utente del sistema

**Estensione C:**    4: Il sistema informa l'utente della mancata registrazione del paziente, specificando quali siano i campi che sono stati ignorati in fase di registrazione

**Estensione D:**    4B (3): (1) Il sistema informa il tutore che lui stesso non esiste all'interno del sistema  
(2) Il tutore annulla la registrazione del minore

**Requisiti non funzionali:**    ◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore, ed in più il paziente minorenne può accedere solamente mediante la supervisione del tutore; è richiesta autenticazione sia per il paziente minorenne, sia per il tutore.

---

---

**Nome del caso d'uso:** Contattare l'amministratore

**Attore primario:** L'utente (del sistema)

**Parti interessate e interessi:**   ◇ L'utente: vuole contattare l'amministratore.

      ◇ L'amministratore: riceve la richiesta dell'utente

**Pre-condizioni:** L'utente è già autenticato come un paziente (o come se stesso, o come il paziente di cui è tutore).

**Garanzia di successo:** Il sistema visualizzerà un messaggio di avvenuta ricezione.

**Scenario principale di successo:** (1) L'utente visualizza la lista delle prenotazioni.

(2) L'utente sceglie l'opzione d'invia un messaggio all'amministratore del reparto.

(3) L'utente richiede il posticipo dell'ultima prenotazione.

(4) L'utente invia la richiesta.

(5) L'utente viene informato dal sistema dell'avvenuto invio.

(6) L'amministratore riceve la richiesta, che viene valutata.

(7) L'amministratore effettua il posticipo della visita interagendo con il sistema.

(8) L'utente viene informato dal sistema dell'avvenuta gestione della richiesta.

(9) Dopo la visita, L'amministratore assocerà il referto alla prenotazione.

**Estensioni:**   **Estensione A:** Inoltre il tutore deve inserire le sue credenziali all'interno del sistema per ogni operazione con il sistema: punti 1 e 5.

**Estensione B:**    **4:** L'utente richiede l'annullamento della prenotazione.

**Estensione C:**    **8:** L'utente viene informato dal sistema dell'avvenuta gestione della richiesta.

**9:** L'amministratore riceve la gestione di un caso urgente.

**10:** L'amministratore sposta la prenotazione dell'utente in questione

**11:** L'utente riceve la notifica che la sua prenotazione è stata spostata

**Requisiti non funzionali:**   ◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore, ed in più il paziente minorenne può accedere solamente mediante la supervisione del tutore; è richiesta autenticazione sia per il paziente minorenne, sia per il tutore.

---

**Nome del caso d'uso:** Effettuare una prenotazione

**Attore primario:** L'utente (del sistema)

**Parti interessate e interessi:**   ◇ L'utente: vuole effettuare una prenotazione.  
  ◇ L'amministratore: riceve la richiesta dell'utente.

**Pre-condizioni:** L'utente è già autenticato come un paziente (o come se stesso, o come il paziente di cui è tutore).

**Garanzia di successo:** Il sistema visualizzerà un messaggio di avvenuta assegnazione.

**Scenario principale di successo:** (1) L'utente seleziona verso quale reparto inoltrare la sua richiesta.

(2) L'utente fornisce una breve descrizione dei sintomi.

(3) L'utente inoltra la richiesta verso il sistema.

(4) Il sistema visualizza l'elenco delle richieste pendenti all'amministratore.

(5) L'amministratore vaglia la richiesta pervenutagli, assegnando una prenotazione dipendentemente al grado di priorità.

(6) Il sistema memorizza la prenotazione, ed invia una notifica all'utente, ed un'altra all'amministratore.

(7) L'utente riceve una notifica dal sistema di avvenuta assegnazione della prenotazione.

(8) L'amministratore riceve la notifica e ritorna all'interfaccia di default.

**Estensioni:**   **Estensione A:**   **1:** Se il paziente in questione è un minorenne, il reparto scelto è per forza quello di pediatria.

**3:** Il tutore deve inserire le sue credenziali all'interno del sistema per ogni operazione con il sistema.

**Requisiti non funzionali:**   ◇ L'amministratore deve essere in grado di "stampare" su interfaccia grafica.

◇ L'amministratore deve essere in grado di notificare la prenotazione tramite un sistema di messaggi interno al sistema.

◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore, ed in più il paziente minorenne può accedere solamente mediante la supervisione del tutore; è richiesta autenticazione sia per il paziente minorenne, sia per il tutore.

◇ L'amministratore deve poter visualizzare un calendario settimanale ove poter convalidare la prenotazione delle visite, e visualizzare gli slot orari già occupati.

---

---

**Nome del caso d'uso:** Visualizzare lo storico delle prenotazioni

**Attore primario:** L'utente (del sistema)

**Parti interessate e interessi:** L'utente: vuole accedere alle informazioni delle prenotazioni

**Pre-condizioni:** L'utente è già autenticato come un paziente (o come se stesso, o come il paziente di cui è tutore), e deve avere prenotazioni a lui associate all'interno del sistema.

**Garanzia di successo:** Il sistema visualizzerà le prenotazioni.

**Scenario principale di successo:** (1) L'utente visualizza la lista delle prenotazioni.

(2) L'utente seleziona una particolare prenotazione.

(3) L'utente sceglie di visualizzare il referto associato alla prenotazione.

(4) Il sistema visualizza il referto a video

**Estensioni:**    **Estensione A:**    **3:** Il paziente non trova alcun referto da visualizzare: si interrompe lo scenario

**Estensione B:** Inoltre il tutore deve inserire le sue credenziali all'interno del sistema per ogni operazione con il sistema: punti 1 e 3.

**Estensione C:**    **4:** L'utente del sistema, può richiedere di salvare il referto come file all'interno del suo sistema.

**Requisiti non funzionali:**    ♦ L'amministratore deve essere in grado di "stampare" su interfaccia grafica.

♦ L'utente deve essere in grado di "stampare" su file.

♦ L'amministratore deve essere in grado di inserire i referti nel sistema tramite inserimento di testo.

---

**Nome del caso d'uso:** Accedere al sistema

**Attore primario:** L'amministratore

**Parti interessate e interessi:** L'amministratore: vuole accedere nel sistema informativo.

**Pre-condizioni:** Nessuna.

**Garanzia di successo:** Il sistema visualizzerà l'interfaccia di default, ovvero il calendario delle prenotazioni.

**Scenario principale di successo:** (1) Il sistema presenta all'amministratore il meccanismo di autenticazione

(2) L'amministratore inserisce Nome Utente e Password

(3) Il sistema verifica la correttezza delle credenziali e le riconosce valide

(4) Il sistema consente l'accesso e visualizza l'interfaccia di default, ovvero . il calendario delle prenotazioni.

**Estensioni:** **Estensione A:** 3: Il sistema non riconosce come valide le credenziali dell'amministratore.

4: Il sistema non consente l'accesso visualizzando un messaggio d'errore.

5: Il sistema ritorna al meccanismo di autenticazione.

**Requisiti non funzionali:** ◇ Le credenziali dell'Amministratore devono essere cablate all'interno del sistema informativo.

**Nome del caso d'uso:** Inserire i referti dei pazienti

**Attore primario:** L'amministratore

**Parti interessate e interessi:** L'amministratore: inserisce il referto all'interno delle prenotazioni alla fine di associarlo alla prenotazione, e quindi alla visita precedentemente assegnata.

**Pre-condizioni:** L'amministratore si autentica ed accede nel sistema. Deve essere presente all'interno del sistema una prenotazione del paziente, e la visita associata alla prenotazione deve essere avvenuta.

**Garanzia di successo:** Il sistema visualizzerà l'associazione tra prenotazione ed il referto.

**Scenario principale di successo:** (1) L'amministratore richiede di poter visualizzare le prenotazioni.

(2) L'amministratore seleziona una particolare prenotazione.

(3) L'amministratore seleziona l'opzione "Inserisci referto"

(4) L'amministratore inserisce i dati del referto in modalità testuale.

(5) Il sistema associa il referto alla visita medica

(6) Il sistema visualizza l'associazione tra la prenotazione ed il referto medico.

**Estensioni:** Nessuna

**Requisiti non funzionali:** ◇ L'amministratore deve essere in grado di inserire i referti nel sistema tramite inserimento di testo.

**Nome del caso d'uso:** Modificare i referti dei pazienti

**Attore primario:** L'amministratore

**Parti interessate e interessi:** L'amministratore: vuole modificare un referto medico.

**Pre-condizioni:** L'amministratore è già autenticato con successo.

**Garanzia di successo:** Il sistema visualizzerà l'associazione tra prenotazione ed il nuovo referto.

**Scenario principale di successo:** (1) L'amministratore sceglie l'opzione "Modifica referto".

(2) Il sistema visualizza l'interfaccia per la modifica dei reperti medici.

(3) L'amministratore inserisce i dati modificati del referto.



(4) L'amministratore salva il nuovo referto.

**Estensioni:** Nessuna

**Requisiti non funzionali:** ◇ L'amministratore "stampa" su interfaccia grafica.

---

**Nome del Caso d'Uso:** Gestire le prenotazioni

**Attore primario:** L'amministratore

**Parti interessate e interessi:** ◇ L'amministratore: vuole visualizzare le richieste di prenotazione e convalidarle.  
◇ L'utente: riceve la notifica dell'amministratore.

**Pre-condizioni:** L'amministratore si autentica ed accede nel sistema.

**Garanzia di successo:** Il sistema visualizzerà un messaggio di effettuata gestione.

**Scenario principale di successo:** (1) L'amministratore visualizza la lista delle prenotazioni pendenti.

(2) L'amministratore attribuisce alla prenotazione data priorità e medico.

(3) L'amministratore conferma la prenotazione.

(4) L'utente riceve dal sistema la notifica di conferma della prenotazione.

**Estensioni:** **Estensione A:** (scenario di modifica delle prenotazioni)

3: (1) L'amministratore sceglie la prenotazione da posticipare

(2) Il sistema visualizza l'interfaccia di modifica della prenotazione, mostrando le altre date libere all'interno del calendario.

(3) L'amministratore posticipa all'interno di uno slot orario libero una prenotazione già precedentemente convalidata.

(4) L'amministratore notifica al paziente la cui prenotazione è stata posticipata.

(5) L'amministratore conferma la nuova prenotazione, in sostituzione di quella modificata.

**Requisiti non funzionali:** ◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore.

---

## 2.4. Glossario

Si produce il glossario visionabile in Tabella 1 a pagina 36.

### 2.5. Diario

**20 Marzo:** Il gruppo si è riunito alle ore 21.00 tramite SKYPE per stendere le *users stories* da consegnare ai nostri committenti, in quanto ritenute fondamentali per il Piano di Sviluppo. Unico assente è stato Fabian per motivi personali, che sarebbero state risolvibili con maggior preavviso.

Ore di lavoro: 5

**23 Marzo:** Il gruppo si è riunito in lab. Ercolani alle ore 13:45 con il cliente per effettuare L'Analisi dei Requisiti. Unico assente è Matej, causa malattia. Fabian ha lasciato il meeting alle 14.30 in quanto doveva prendere il treno per tornare a casa. Il meeting con il cliente finisce alle 14.50 circa. La sera il gruppo si è riunito su SKYPE, Matej è stato informato e si è iniziata la discussione sul proseguimento dei lavori.

Ore di lavoro: 8

**24 Marzo:** Il gruppo si è riunito su SKYPE ed è continuata la discussione riguardo il proseguimento del lavoro, seguita da una spartizione del lavoro futuro.

Ore di lavoro: 5

**26 Marzo:** Durante il fine settimana sono stati prodotti le bozze per il Glossario di progetto, gli Use Case, una prima versione del documento da consegnare per questa fase, il database e questo stesso diario. In giornata sono state riviste la stima dei costi e diverse delle bozze sopra elencate.

Ore di lavoro: 6

**27 Marzo:** Il gruppo si è riunito in laboratorio Ercolani per continuare il lavoro. Unico assente Matej causa malattia, che comunque era presente su SKYPE. Sono stati rivisti gli Use Case.

Ore di lavoro: 5

**28 Marzo:** Il gruppo si è riunito in laboratorio Ercolani con il cliente per chiarire ulteriori dettagli, in quanto erano rimaste delle incomprensioni in seguito alla riunione avvenuta il 30 Marzo. Unico assente Matej causa malattia, che comunque era presente su SKYPE. Successivamente sono stati rivisti nuovamente gli Use Case. Segno qui per annotare una dimenticanza di ruolo: Non ho mai annotato che da circa una settimana Fabian ha iniziato a creare l'interfaccia grafica aggiornandola e modificandola via via che il progetto assume una forma più concreta.

Ore di lavoro: 7

Nome	Definizione
<b>Paziente</b>	È un utilizzatore del sistema, dal quale otterrà valore tramite l'effettuazione di una visita.
<b>Paziente minorenn</b>	Il paziente minorenn è un PAZIENTE che, per usufruire dei servizi del sistema, ha bisogno di: fornire le proprie credenziali (unicamente all'atto dell'autenticazione), essere associato ad uno ed un solo tutore, e fornirne le credenziali del tutore.
<b>Paziente maggiorenne</b>	Il paziente maggiorenne è un utilizzatore del sistema che, per usufruire dei servizi del sistema, ha bisogno unicamente di fornire le proprie credenziali (unicamente all'atto dell'autenticazione). Assieme al Tutore è un <b>Utente</b> del sistema.
<b>Tutore</b>	Il tutore è un utilizzatore del sistema che può accedere a questo usando le sue credenziali e quelle del suo tutelato, allo scopo di garantire a quest'ultimo i servizi offerti al sistema. Assieme al Paziente maggiorenne è un <b>Utente</b> del sistema.
<b>Amministratore</b>	L'amministratore è unico, per reparto, e preesistente nel sistema. Egli si occupa di classificare le richieste dei pazienti ed assegnare loro una prenotazione, di cambiare una prenotazione preesistente qualora vi sia l'esigenza, da lui considerata, di privilegiarne una a priorità maggiore (nei confronti di altre già concordate).
<b>Prenotazione</b>	Una prenotazione è un oggetto creato dall'amministratore e riferito ad un unico paziente che indica la data, la sala dalla quale segue l'indicazione di un reparto, per cui il dato paziente ha richiesto una visita. Un insieme di prenotazioni è detto <b>storico</b>
<b>Reparto</b>	Il reparto è il luogo dove avvengono le diverse tipologie di visite in base alla distinzione tra Ortopedia e Pediatria. Ogni reparto è dotato di 2 sale (distinte).
<b>Sala</b>	La sala è il luogo dove può avvenire, in ogni ora prefissata, una sola visita
<b>Referto</b>	Il referto è un oggetto rilasciato al paziente al termine della visita. Ogni visita produce uno ed un solo referto, il quale è associato ad un unico paziente e può essere controllato da quest'ultimo attraverso il sistema informativo.
<b>Messaggio</b>	Un messaggio è formato da 3 parametri: un campo destinatario, un campo mittente e contenuto del messaggio. Se il campo destinatario è un paziente allora il mittente sarà un amministratore, viceversa, mittente paziente destinatario amministratore. Può essere una richiesta di visita od una conferma di prenotazione, o una richiesta di cancellazione o di spostamento della stessa.

TABELLA 1. *Glossario.*

## CAPITOLO 3

### Revisioni delle stime precedenti

#### Indice

3.1. Aggiunta di Tool di sviluppo	37
3.2. Revisione della valutazione dello sforzo	37
3.2.1. Valutazione dell'analisi dello sforzo con gli Use Case	37
3.2.2. Revisione dei computi effettuati precedentemente	38

Dopo le considerazioni effettuate nel Capitolo 7 a pagina 75, possiamo effettuare una prima revisione delle stime che sono state precedentemente effettuate (v. Sezione 1.4 a pagina 11).

#### 3.1. Aggiunta di Tool di sviluppo

Si è introdotto INKSCAPE per la compilazione dei files di estensione SVG in PDF, e per modifiche grafiche ai diagrammi ottenuti con ARGOUML

#### 3.2. Revisione della valutazione dello sforzo

**3.2.1. Valutazione dell'analisi dello sforzo con gli Use Case.** Si potrebbe pensare di effettuare una nuova valutazione della stima dello sforzo tramite l'impiego degli Use Case; di fatti, da questi use case, dipendono i seguenti elementi di progettazione:

- ◇ Dimensione dell'applicazione in LOC
- ◇ Numero di test case prodotti

Tuttavia, in quanto questi Use Case possono essere creati a differenti livelli di astrazione, non ci sentiamo di utilizzare questa tipologia di stima in sostituzione all'analisi precedente<sup>1</sup>. Inoltre, non è nemmeno possibile utilizzare la formula di Smith<sup>2</sup>, in quanto non possediamo dati medi o/e storici in LOC per use case di questo tipo di sistema, necessari al computo richiesto dalla formula.

<sup>1</sup>R.S. Pressman: op. cit.

<sup>2</sup>R.S. Pressman: op. cit.

Unadjusted Function Point (UFP)	Tipologia	Quantità
<i>Input (EI)</i>	Semplice	2
<i>Input (EI)</i>	Media	3
<i>Output (EO)</i>	Semplice	1
<i>Output (EO)</i>	Media	2
<i>Interrogazioni (EQ)</i>	Semplice	8
<i>File logici interni (ILF)</i>	-	0
<i>File logici esterni (EIF)</i>	Semplice	1
Totale		55

TABELLA 1. Definizione degli Unadjusted Function Point : seconda stima.

**3.2.2. Revisione dei computi effettuati precedentemente.** In considerazione di quanto rilevato precedentemente, ed in seguito alla modifica dei requisiti apportata come conseguenza dell'incontro nel cliente, descritto nella Sezione 2.2 a pagina 20, possiamo effettuare una revisione della stima dello sforzo rispetto alla valutazione fornita nel **Documento Primo**.

Rieffettuando la computazione delle stime precedentemente effettuate, abbiamo che l'unica variazione degna di nota si è riscontrata negli UFP, dei quali rieffettuiamo la stima all'interno della Tabella 1, dovuta al fatto che si è preferito unire interfacce grafiche con compiti simili.

$$\begin{aligned}
 FP &= conteggio - UFP \cdot (0.65 + 0.01 \cdot \sum_{i=1}^{14} F_i) \\
 &= 55 \cdot (0.65 + 0.01 \cdot 26) = 50.05
 \end{aligned}$$

Considerando ancora il fattore di conversione FP/LOC con il linguaggio Java, otteniamo:

$$LOC = 63 \cdot 66 = 3153$$

Da cui dalla stima di *Bailey-Basili*<sup>3</sup> otteniamo:

$$\begin{aligned}
 E_{bb} &= 5,5 + 0,7 \cdot KLOC^{1,16} \text{mesi/persona} \\
 &= 5.5 + 0.7 \cdot 4,3^{1,16} \simeq 8.15 \text{mesi/persona}
 \end{aligned}$$

Si osserva quindi una riduzione dei costi, basata unicamente dalla riduzione della stima effettuata dalla valutazione dei dati transitanti dell'interfaccia: tuttavia non reputiamo questa stima soddisfacente per la valutazione della complessità intrinseca del progetto.

<sup>3</sup>R.S. Pressman: op. cit.

## **Parte 3**

# **Documento Terzo**





## CAPITOLO 4

### Analisi I

#### Indice

---

4.1. Premessa	41
4.2. CRC cards	42
4.3. Modello di dominio	42
4.4. Documento dell'architettura software	44
4.4.1. Amministratore	44
4.4.2. Utente	52
4.4.3. Ampliamenti al modello di dominio suggeriti	60
4.4.4. Definizione delle priorità dei casi d'uso	60
4.5. Diario	62

---

#### 4.1. Premessa

La fase di sviluppo si strutturerà come segue:

◇ *Definizione del Modello di Progetto*

Partendo dal *Glossario* (Sezione 2.4 a pagina 33) definito nella fase precedente, si definiscono le prime schede CRC (v. Sezione 4.2 nella pagina seguente), dalle quali si arriva alla definizione del *Modello di Dominio* del nostro applicativo (v. Sezione 4.3 nella pagina successiva); da quest'ultimo discende pure l'*Analisi del Database*.

◇ *Documento dell'architettura software*

Si effettua una prima revisione del modello di dominio, iniziando con l'analisi delle operazioni fondamentali che saranno richieste dall'UTENTE del sistema per conto del PAZIENTE (sè stesso o il suo tutelarlo).

Partendo dalla trattazione dei Casi d'Uso, si effettua la descrizione dell'architettura software mediante la **Vista dei Dati** mediante DIAGRAMMI DI ATTIVITÀ, e la **Vista dei Casi d'Uso** tramite DIAGRAMMI DI SEQUENZA. Questi verranno successivamente raffinati, una volta completato il *Modello di Business* dell'applicazione.

Associamo inoltre i seguenti documenti:

- *Analisi del Database*: (v. Capitolo 8 a pagina 91) in seguito alla definizione del modello di progetto, viene effettuata la strutturazione del database, in modo da ottenere già le componenti base sulle quali effettuare.
- *Analisi del Client*: (v. Capitolo 5.1 a pagina 63) ci limitiamo a descrivere le operazioni che possono essere effettuate tramite l'interfaccia grafica.
- *Analisi del Server*: (v. Capitolo 5.2 a pagina 69) anche in questo caso, ci limitiamo a descrivere le operazioni che possono essere effettuate tramite l'interfaccia grafica.

## 4.2. CRC cards

In base al colloquio col cliente, possiamo già fornire una prima specifica delle schede CRC presentate in Tabella 1 nella pagina successiva.

## 4.3. Modello di dominio

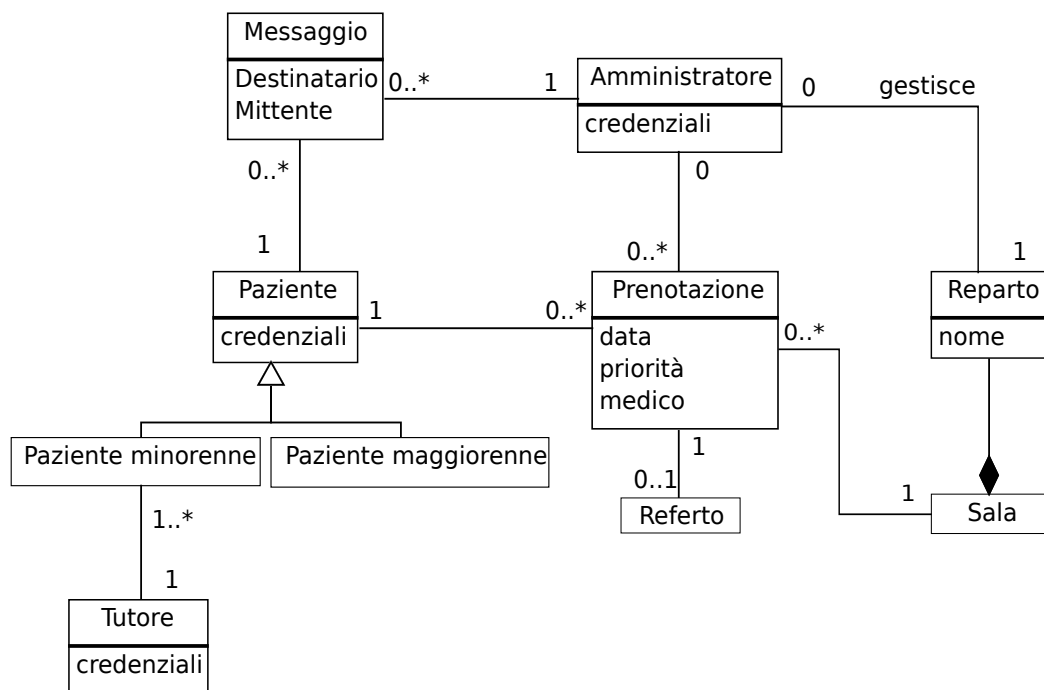
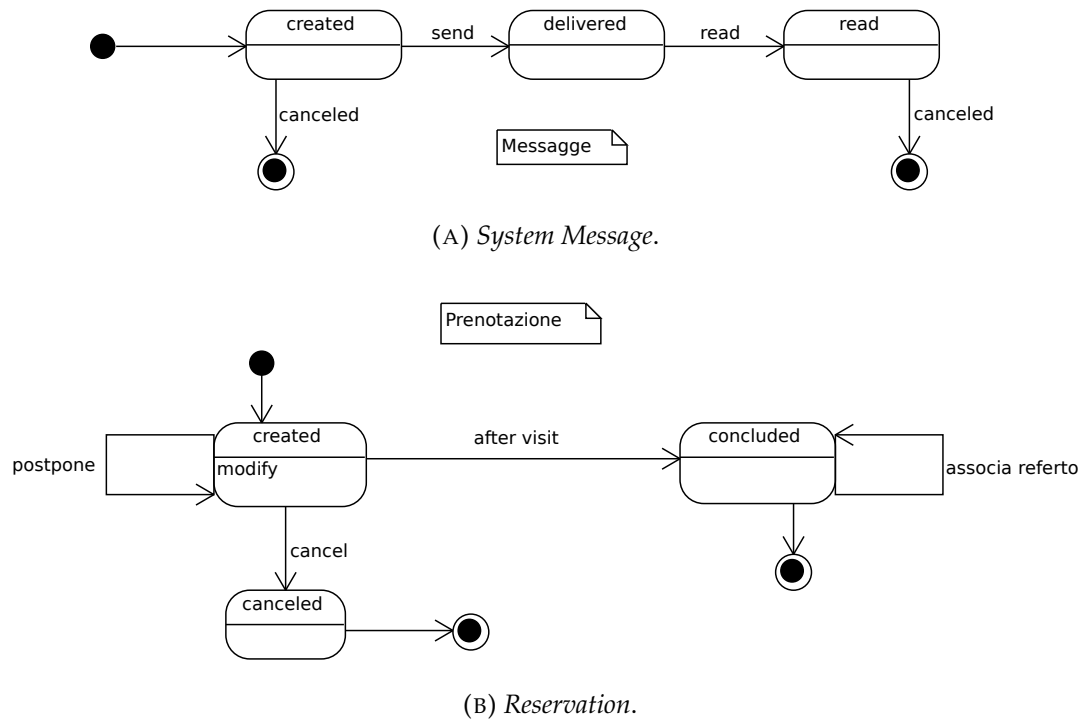


FIGURA 4.3.1. *Domain Model*.

In seguito ad un primo sviluppo delle schede CRC, possiamo ottenere il modello di dominio presentato in Figura 4.3.1.

<b>Reparto</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i>	<b>Referto</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i>
<b>Prenotazione</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i>	<b>Sala</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i>
<b>Paziente</b> <i>Superclasses:</i> <i>Subclasses:</i> Paziente Minorenne, Paziente Maggioreenne <i>Responsibilities</i> <i>Collaborators</i> <i>Autentica:</i> <i>Registra:</i>	
<b>Amministratore</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i> <i>Crea Prenotazione:</i> PAZIENTE, PRENOTAZIONE, REPARTO, SALA, MESSAGGIO	
<b>Messaggio</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i>	<b>Paziente Minorenne</b> <i>Superclasses:</i> Paziente <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i> <i>Registra Tutore:</i> TUTORE <i>Autentica:</i> TUTORE
<b>Paziente Maggioreenne</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i> <i>Prenota:</i> AMMINISTRATORE <i>Visualizza Storico:</i> PRENOTAZIONE <i>Visualizza Referto:</i> REFERTO <i>Stampa Storico:</i> PRENOTAZIONE <i>Stampa Referto:</i> REFERTO <i>Informa amministratore:</i> MESSAGGIO, AMMINISTRATORE	
<b>Tutore</b> <i>Superclasses:</i> <i>Subclasses:</i> <i>Responsibilities</i> <i>Collaborators</i> <i>Prenota:</i> PAZIENTE MINORENNE, AMMINISTRATORE <i>Visualizza Storico:</i> PRENOTAZIONE, PAZIENTE MINORENNE <i>Visualizza Referto:</i> REFERTO, PAZIENTE MINORENNE <i>Stampa Storico:</i> PRENOTAZIONE, PAZIENTE MINORENNE <i>Stampa Referto:</i> REFERTO, PAZIENTE MINORENNE <i>Informa amministratore:</i> MESSAGGIO, AMMINISTRATORE	

TABELLA 1. CRC Cards.

FIGURA 4.4.1. *Statecharts for some classes of the Domain Model.*

#### 4.4. Documento dell'architettura software

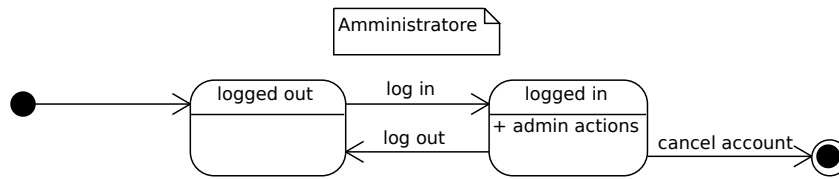
In questa sezione analizzeremo diffusamente il processo di dettaglio dell'architettura software, effettuando la distinzione tra lato Amministratore e lato Utente, come avverrà nelle sottosezioni seguenti. Abbiamo voluto evidenziare il diagramma degli stati di alcune classi del nostro modello di dominio, tramite la Figura 4.4.1.

Ci sentiamo di effettuare un'unica precisazione per quanto riguarda la Figura 4.4.1(b): con "modifica del referto" intendiamo sia la modifica della prenotazione dovuta al primo inserimento del documento, sia la modifica effettiva del referto associato.

Inoltre ci sentiamo di rimarcare l'impossibilità di utilizzare, tramite il tool ARGOUML i "combined fragments": siamo stati conseguentemente costretti ad apportare le opportune modifiche tramite l'utilizzo di INKSCAPE.

##### 4.4.1. Amministratore.

4.4.1.1. *Vista dei Dati.* Possiamo fornire una descrizione dei possibili stati dell'amministratore in Figura 4.4.2 nella pagina successiva: nota che, anche se non è prevista l'eliminazione da interfaccia grafica di un singolo amministratore, questa è sempre possibile andando ad effettuare modifiche nel database.

FIGURA 4.4.2. *Administrator's Statechart.*

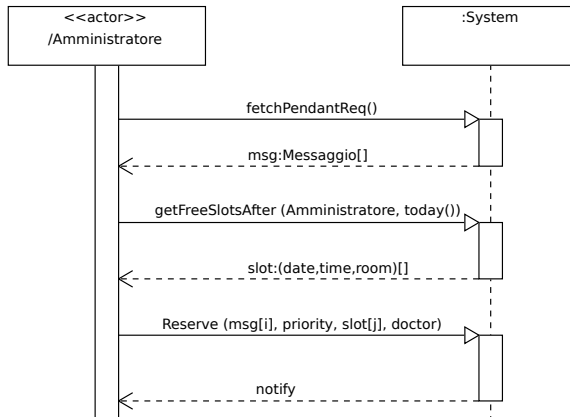
4.4.1.2. *Vista dei Casi d'Uso.* Nella Figura 4.4.3 nella pagina seguente, mostriamo i primi DIAGRAMMI DI SEQUENZA attinenti all'interazione tra Amministratore ed una generica istanza del Sistema (*System*). Questi diagrammi fanno riferimento ai Casi d'Uso descritti nella Sottosezione 2.3.2 a pagina 25, ed in particolare ai seguenti:

- ◊ Contattare l'amministratore (lato Amministratore)
- ◊ Effettuare una prenotazione (lato Amministratore)
- ◊ Accedere al sistema (Attore primario: amministratore)
- ◊ Inserire i referti dei pazienti
- ◊ Modificare i referti dei pazienti
- ◊ Gestire le prenotazioni

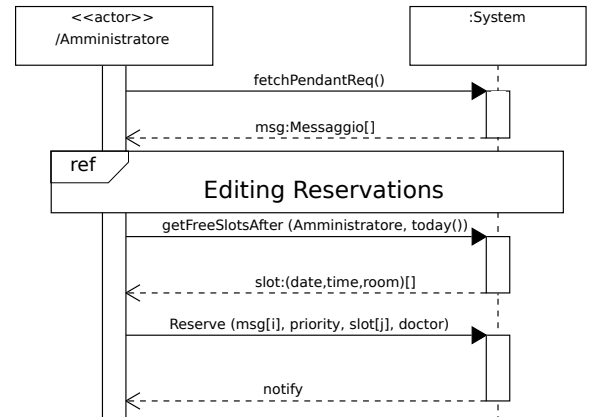
Per ognuna delle operazioni ottenuto dal DIAGRAMMA DI CLASSE "System Methods to implement", definiamo i contratti corrispondenti dettagliati qui sotto, allo scopo di definire l'interfaccia di sistema pubblica dei nostri metodi:

- ◊ `getFreeSlotsAfter`
- ◊ `fetchReservations`
- ◊ `addReport`
- ◊ `postponeReservs`
- ◊ `Reserve`
- ◊ `fetchPendantReq`
- ◊ `Login`
- ◊ `fetchRemoveReq`
- ◊ `fetchPostponeReq`
- ◊ `destroyRequestPostpone`
- ◊ `freeSlotReserv`

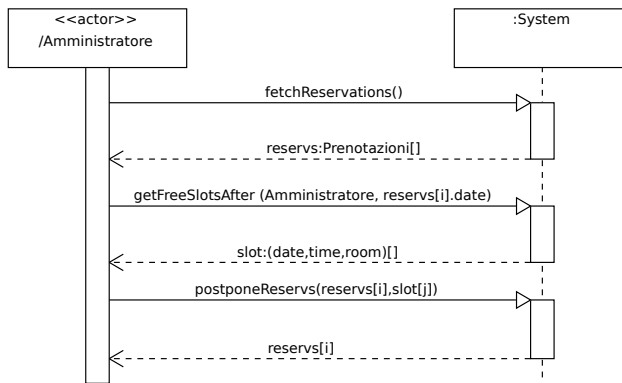
Nelle post-condizioni si considerano i suggerimenti forniti all'interno della Sottosottosezione 4.4.3 a pagina 60.



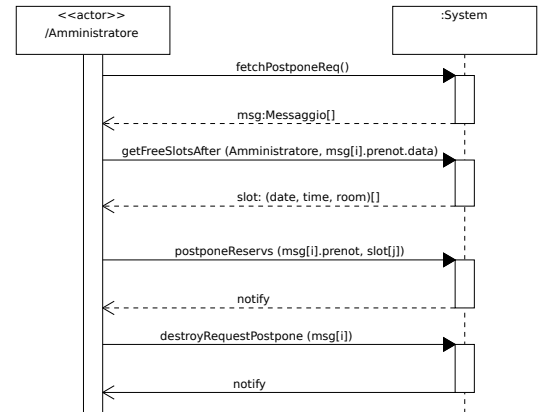
(A) Adding Patient's Reservation.



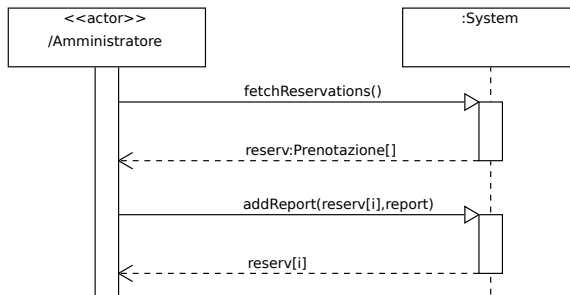
(B) Adding Patient's Urgent Reservation.



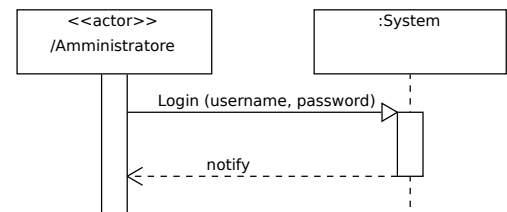
(C) Editing Reservations.



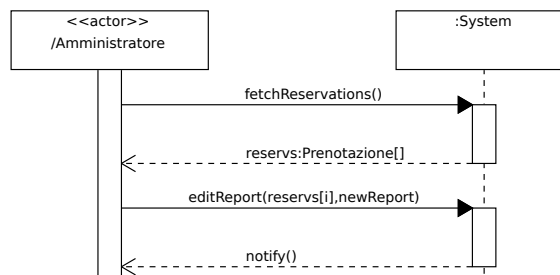
(D) Postponing Reservations.



(E) Adding Report to Reservations.



(F) Authentication.



(G) Editing Reservations' Report.

FIGURA 4.4.3. Administrator's Use Case View (1).

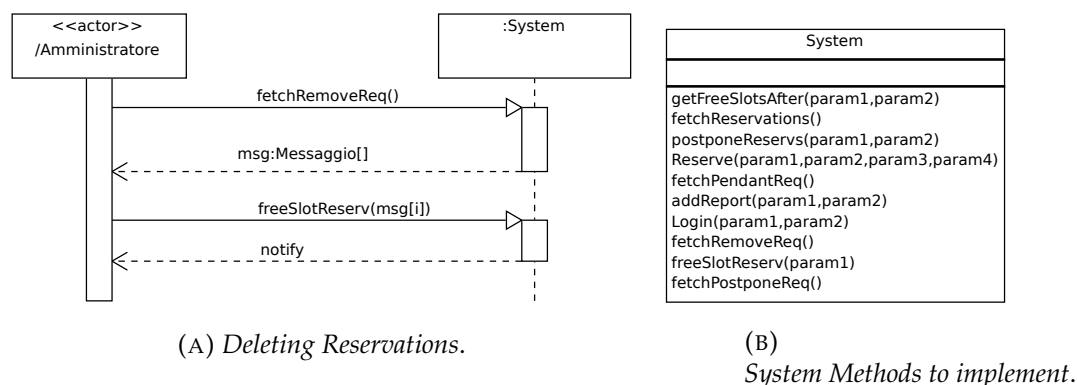


FIGURA 4.4.4. Administrator's Use Case View (1)

CONTRATTO 01: <b>getFreeSlotsAfter</b>	
Operazione:	<code>getFreeSlotsAfter(amministratore, time)</code>
Use Case:	Gestire le prenotazioni, Effettuare una prenotazione (lato amministratore)
Pre-condizioni:	Nessuna
Post-condizioni:	<ul style="list-style-type: none"> <li>Sono creati gli Slot <code>s[]</code> non ancora occupati nel sistema a partire dal tempo <code>time</code> per il reparto gestito dall'amministratore (creazione di istanza).</li> </ul>
CONTRATTO 02: <b>fetchReservations</b>	
Operazione:	<code>fetchReservations()</code>
Use Case:	Inserire i referti dei pazienti, Modificare i referti dei pazienti, Gestire le prenotazioni
Pre-condizioni:	Sono presenti delle prenotazioni all'interno del sistema che devono essere gestite
Post-condizioni:	<ul style="list-style-type: none"> <li>Sono ottenute le Prenotazioni <code>p[]</code> memorizzate all'interno del sistema, ed associate all'Amministratore <code>admin.reservations = p[]</code> (creazione di istanza).</li> </ul>

---

**CONTRATTO 03: addReport**

---

*Operazione:* addReport (prenotazione, report)

*Use Case:* Inserire i referti dei pazienti

*Pre-condizioni:* L'amministratore associa un Referto alla Prenotazione a cui ha fatto seguito la vista che l'ha generato.

*Post-condizioni:*

- È associata la prenotazione *prenotazione* ad un Referto esistente: *prenotazione.referto = report* (associazione formata).
- 

---

**CONTRATTO 04: postponeReservs**

---

*Operazione:* postponeReservs (prenotazione, newSlot)

*Use Case:* Contattare l'amministratore (lato Amministratore), Gestire le prenotazioni

*Pre-condizioni:* Mentre per quanto concerne *Contattare l'amministratore* è necessario che un utente abbia effettuato la richiesta, nel secondo caso è necessario che sia pervenuta una richiesta da un utente.

*Post-condizioni:*

- È liberato lo Slot temporale *prenotazione.slot* associato alla Prenotazione (*cancellazione di istanza*)
  - All'interno della Prenotazione, l'associazione dello Slot liberato precedentemente è sostituita con un nuovo Slot temporale ottenuto tra quelli disponibili *newSlot* (associazione formata)
  - È creata una nuova istanza *msg* di *MAmmministratore* (*creazione di istanza*)
  - È settata la tipologia *msg.kind* del messaggio a *defer* (*modifica di attributo*)
  - È associato il mittente nel messaggio dalla Prenotazione: *msg.mittente = newSlot.sala.amministratore* (associazione formata)
  - È associato il destinatario nel messaggio dalla Prenotazione: *msg.destinatario = prenotazione.paziente* (associazione formata)
  - È associata la prenotazione al messaggio: *msg.prenotazione = prenotazione* (associazione formata)
-



---

**CONTRATTO 05: Reserve**

---

<i>Operazione:</i>	<code>Reserve(messaggio, valPriorità, newSlot, dottore)</code>
<i>Use Case:</i>	Effettuare una prenotazione (lato Amministratore)
<i>Pre-condizioni:</i>	Un utente richiede di effettuare una visita tramite l'invio di un <code>Messaggio</code> all'amministratore.
<i>Post-condizioni:</i>	<ul style="list-style-type: none"><li>• È stata creata una nuova Prenotazione <i>p</i> (creazione di istanza)</li><li>• È stata associata alla Prenotazione il Paziente, contenuto all'interno del messaggio: <i>p.paziente</i> = <i>messaggio.mittente</i> (associazione formata).</li><li>• È stata associata alla Prenotazione uno Slot temporale tra quelli disponibili: <i>p.slot</i> = <i>newSlot</i> (associazione formata).</li><li>• È stata associata alla Prenotazione una Priorità di visita: <i>p.priorità</i> = <i>Priorità</i> (modifica di attributo)</li><li>• È stata associata alla Prenotazione un Medico curante: <i>p.dottore</i> = <i>dottore</i> (modifica di attributo)</li><li>• È creata una nuova istanza <i>msg</i> di <code>MAmministratore</code> (creazione di istanza)</li><li>• È settata la tipologia <i>msg.kind</i> del messaggio a <i>reserve</i> (modifica di attributo)</li><li>• È associato il mittente del messaggio: <i>msg.mittente</i> = <i>messaggio.destinatario</i> (associazione formata)</li><li>• È associato il destinatario del messaggio: <i>msg.destinatario</i> = <i>msg.mittente</i> (associazione formata)</li><li>• È associata la prenotazione al messaggio: <i>msg.prenotazione</i> = <i>p</i> (associazione formata)</li><li>• È rimossa l'istanza di <i>messaggio</i> (cancellazione di istanza).</li></ul>

---

---

**CONTRATTO 06: fetchPendantReq**


---

*Operazione:* `fetchPendantReq()`  
*Use Case:* Effettuare una prenotazione (lato Amministratore)  
*Pre-condizioni:* Nessuna  
*Post-condizioni:*

- Sono associate le nuove istanze dei messaggi `msgs[]` di richiesta di visita all'Amministratore competente in `admin.messages` (modifica di attributo)

---



---

**CONTRATTO 07: Login**


---

*Operazione:* `Login(Username, Password)`  
*Riferimento (Use Case):* Accedere al sistema (Attore primario: amministratore)  
*Pre-condizioni:* Nessuna  
*Post-condizioni:*

- È stata creata un'istanza `g` della classe `Guest`.
- Si setta l'attributo `g.kind` o `isOrthopedic` o `isPediatrics` affermativamente nel caso in cui l'amministratore afferisca al reparto di ortopedia o a quello di pediatria.
- È settato l'attributo `g.logged` a `true` se l'autenticazione è avvenuta con successo.

---



---

**CONTRATTO 08: fetchRemoveReq**


---

*Operazione:* `fetchRemoveReq()`  
*Riferimento (Use Case):* Contattare l'amministratore (lato Amministratore)  
*Pre-condizioni:* Almeno un utente invia all'amministratore la richiesta di cancellazione della visita.  
*Post-condizioni:*

- I nuovi messaggi di cancellazione `msgsdel[]` della visita sono associati all'amministratore competente: `admin.messagesdel = msgsdel[]` (modifica di attributo)

---

---

<b>CONTRATTO 09:</b>	<b>fetchPostponeReq</b>
<i>Operazione:</i>	<code>fetchPostponeReq()</code>
<i>Riferimento (Use Case):</i>	Contattare l'amministratore (lato Amministratore)
<i>Pre-condizioni:</i>	Almeno un utente invia all'amministratore la richiesta di posticipazione della visita.
<i>Post-condizioni:</i>	<ul style="list-style-type: none"> <li>• I nuovi messaggi di richiesta di posticipazione <code>msgspost[]</code> della visita sono associati all'amministratore competente: <code>admin.messagespost = msgspost[]</code> (modifica di attributo)</li> </ul>

---



---

<b>CONTRATTO 10:</b>	<b>destroyRequestPostpone</b>
<i>Operazione:</i>	<code>destroyRequestPostpone(msg)</code>
<i>Riferimento (Use Case):</i>	Contattare l'amministratore (lato Amministratore), Gestire le prenotazioni.
<i>Pre-condizioni:</i>	L'amministratore deve aver gestito la richiesta di spostamento contenuta nel messaggio.
<i>Post-condizioni:</i>	<ul style="list-style-type: none"> <li>• È cancellata l'istanza <code>msg</code> (cancellazione)</li> </ul>

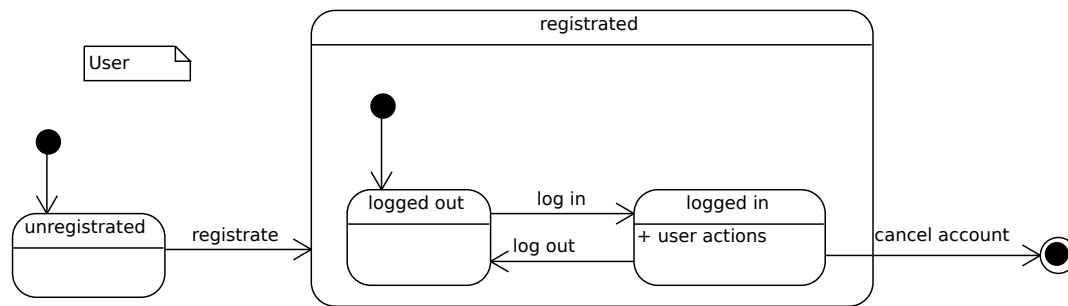
---

CONTRATTO 11: <b>freeSlotReserv</b>	
Operazione:	<code>freeSlotReserv(messaggio)</code>
Riferimento (Use Case):	Contattare l'amministratore (lato Amministratore)
Pre-condizioni:	Almeno un utente invia all'amministratore la richiesta di cancellazione della visita.
Post-condizioni:	<ul style="list-style-type: none"> <li>• È creata una nuova istanza <i>msg</i> di <i>MAmmministratore</i> (creazione di istanza)</li> <li>• È settata la tipologia <i>msg.kind</i> del messaggio a <i>revoke</i> (modifica di attributo)</li> <li>• È associato il mittente del messaggio: <i>msg.mittente</i> = <i>messaggio.destinatario</i> (associazione formata)</li> <li>• È associato il destinatario del messaggio: <i>msg.destinatario</i> = <i>msg.mittente</i> (associazione formata)</li> <li>• È associato il contenuto testuale al messaggio: <i>msg.content</i> = <i>p.toText()</i> (associazione formata)</li> <li>• È eliminata una istanza di Prenotazione <math>\pi</math> precedentemente inviata dal Paziente <i>p</i>, che è indicata all'interno del messaggio nell'attributo <i>message.prenotazione</i> (cancellazione di istanza).</li> <li>• È rimossa l'istanza di <i>messaggio</i> (cancellazione di istanza).</li> </ul>

#### 4.4.2. Utente.

4.4.2.1. *Vista dei Dati.* Possiamo fornire una descrizione dei possibili stati dell'utente in Figura 4.4.5 nella pagina successiva: nota che, anche se non è prevista l'eliminazione da interfaccia grafica di un singolo utente, questa è sempre possibile andando ad effettuare modifiche nel database. Questa considerazione può essere valutata anche per le credenziali del Tutore.

4.4.2.2. *Vista dei Casi d'Uso.* Nella Figura 4.4.6 a pagina 54 e nella Figura 4.4.7 a pagina 55 mostriamo i primi DIAGRAMMI DI SEQUENZA attinenti all'interazione tra Utente ed una generica istanza del Sistema (*System*). Questi diagrammi fanno riferimento ai Casi d'Uso descritti nella Sottosezione 2.3.2 a pagina 25

FIGURA 4.4.5. *User's Statechart.*

Elenchiamo i contratti che verranno dettagliati più sotto, allo scopo di definire l'interfaccia di sistema pubblica dei nostri metodi:

- ◇ registerNewPatient
- ◇ login
- ◇ fetchReservation
- ◇ fetchReport
- ◇ tutorCredentials
- ◇ retireReservation
- ◇ deferReservation
- ◇ bindTutorForPatient
- ◇ printOnSupport
- ◇ reserveVisit
- ◇ registerNewPatientFromTutor
- ◇ registerTutor

---

**CONTRATTO 12: registerNewPatient**


---

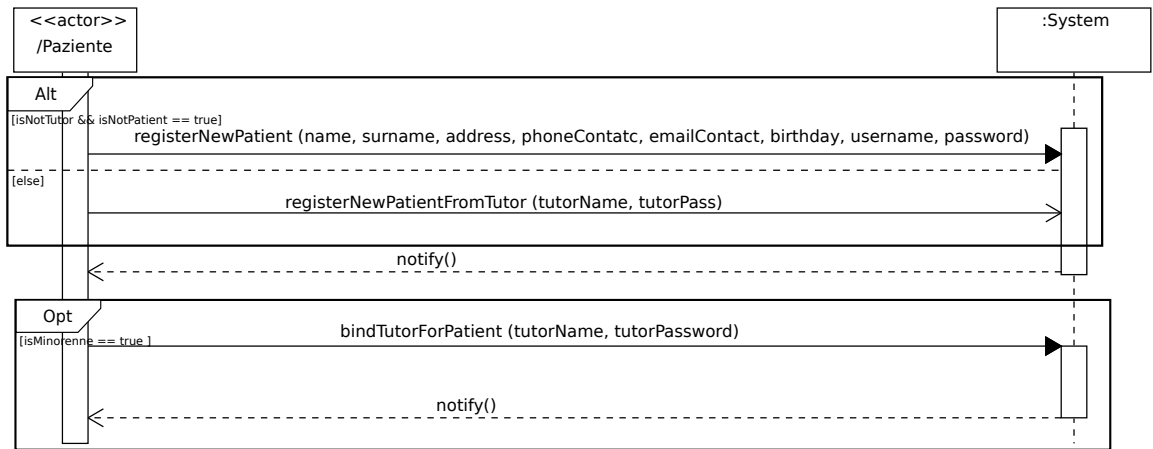
*Operazione:* registerNewPatient(name, surname, address, phoneContact, emailContact, birthday, username, password)

*Use Case:* Registrare il paziente

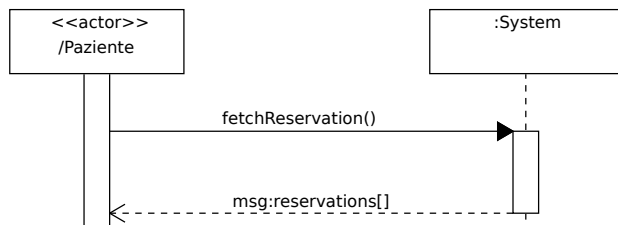
*Pre-condizioni:* Nessuna.

*Post-condizioni:*

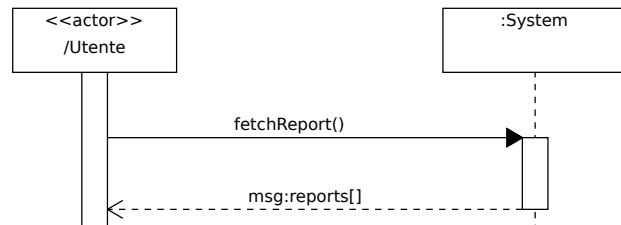
- Viene inserito un nuovo paziente nel sistema, creando l'elemento *patient* con i rispettivi parametri se il paziente è maggiorenne. (*creazione di istanza*)
-



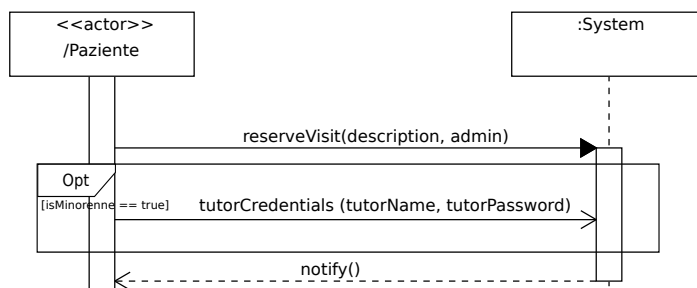
(A) Patient's Registration.



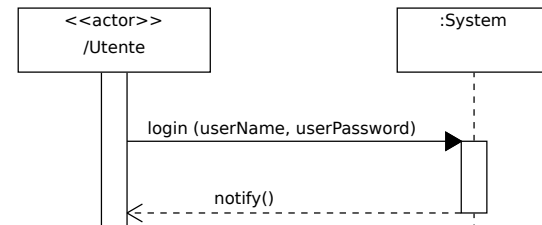
(B) View Reservations.



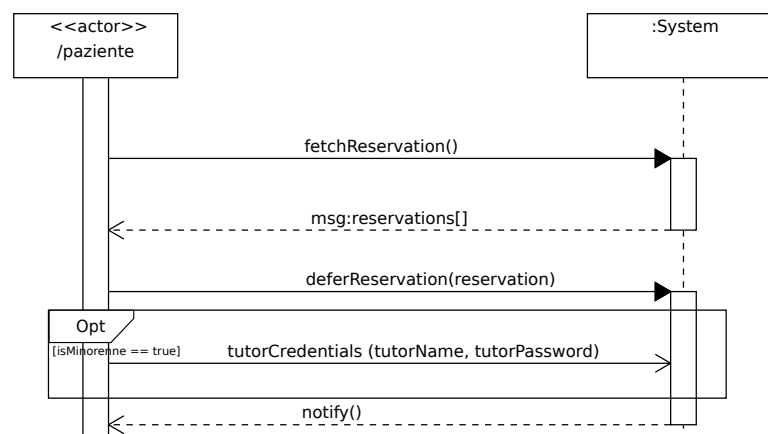
(C) View Reports.



(D) Request Reservations.

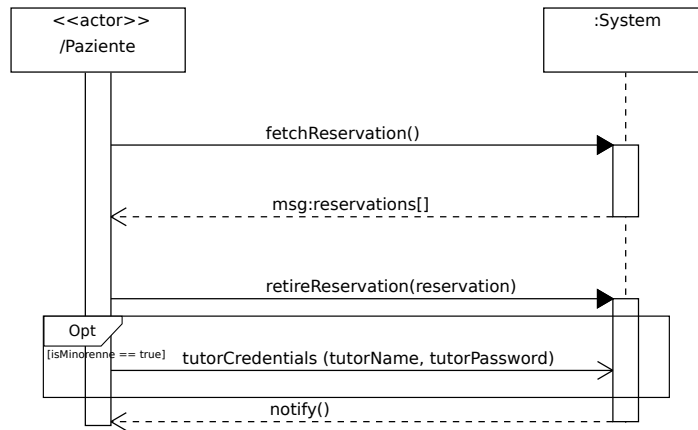


(E) User's Login as Patient.

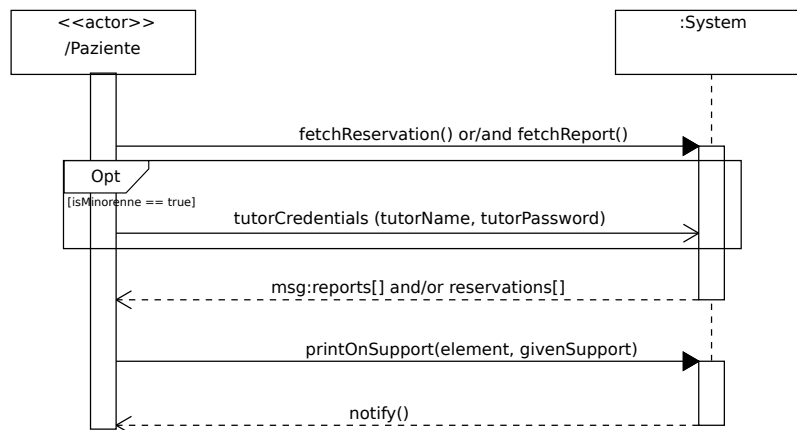


(F) Postpone Patient's Reservation.

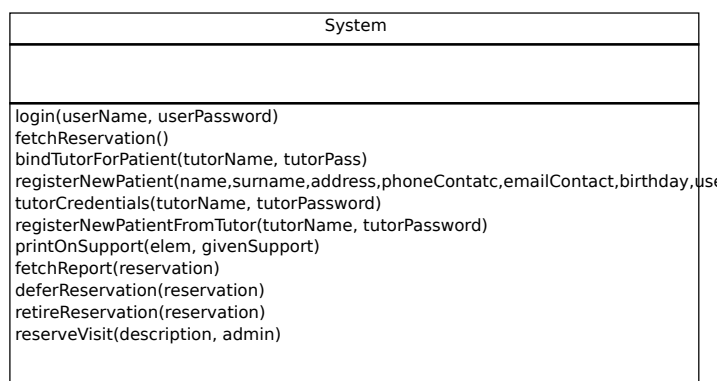
FIGURA 4.4.6. Users' Use Case View (1).



(A) User's Revoke of Reservation.



(B) User's Print.



(C) System Methods to implement.

FIGURA 4.4.7. Users' Use Case View (2).

---

**CONTRATTO 13: login**

---

*Operazione:* login(userName, userPassword)

*Use Case:* Accedere al sistema.

*Pre-condizioni:* Nessuna

*Post-condizioni:*

- È stata creata un'istanza *g* della classe *Guest* (*creazione di istanza*).
  - Viene settato lo stato di questa classe come *g.isPatientMinor* o in *g.isPatientMajor* a seconda che il paziente sia minorenne o maggiorenne.
  - È settato l'attributo *g.logged* a *true* se il paziente è maggiorenne (*g.isPatientMajor*) e se l'autenticazione è avvenuta con successo
- 

---

**CONTRATTO 14: fetchReservation**

---

*Operazione:* fetchReservation()

*Use Case:* Visualizzare lo storico delle prenotazioni

*Pre-condizioni:* Il paziente è stato autenticato dal sistema

*Post-condizioni:*

- Sono ottenute le istanze delle prenotazioni *reserv[]* di visita in *patient.reservations* (*modifica di attributo ed associazione formata*)
- 

---

**CONTRATTO 15: fetchReport**

---

*Operazione:* fetchReport(reservation)

*Use Case:* Visualizzare lo storico delle prenotazioni

*Pre-condizioni:* Il paziente è stato autenticato dal sistema ed ha richiesto di ottenere le prenotazioni.

*Post-condizioni:*

- Ottiene da *reservation* il referto nell'attributo *reservation.report*
-



---

**CONTRATTO 16: tutorCredentials**

---

<i>Operazione:</i>	<code>tutorCredentials( tutorName, tutorPassword)</code>
<i>Use Case:</i>	Contattare l'amministratore, Effettuare una prenotazione
<i>Pre-condizioni:</i>	L'utente accede in favore di un paziente minorenni.
<i>Post-condizioni:</i>	<ul style="list-style-type: none"><li>• Si associa al Guest <i>g</i> corrente nell'attributo <i>tutor</i> l'istanza di tutore con le credenziali indicate (<i>associazione formata</i>).</li></ul>

---

---

**CONTRATTO 17: retireReservation**

---

<i>Operazione:</i>	<code>retireReservation(res)</code>
<i>Use Case:</i>	Contattare l'amministratore
<i>Pre-condizioni:</i>	L'Utente è loggato nel sistema ed ha in precedenza effettuato una prenotazione
<i>Post-condizioni:</i>	<ul style="list-style-type: none"><li>• È creata una nuova istanza di Messaggio <i>msg</i> all'interno del sistema (<i>creazione di istanza</i>)</li><li>• È settata la tipologia di richiesta <i>msg.kind</i> a <i>revoke</i> (<i>modifica di attributo</i>)</li><li>• È associato al messaggio <i>msg</i> la Prenotazione <i>res</i> da revocare (<i>associazione formata</i>)</li><li>• È associato al messaggio <i>msg</i> l'amministratore tramite <i>msg.destinatario = res.sala.amministratore</i> (<i>associazione formata</i>)</li><li>• È associato al messaggio <i>msg</i> l'utente <i>patient</i> che ne ha fatto richiesta nell'attributo <i>msg.patient</i> (<i>associazione formata</i>)</li></ul>

---

---

**CONTRATTO 18: deferReservation**


---

<i>Operazione:</i>	<code>deferReservation(res)</code>
<i>Use Case:</i>	Contattare l'amministratore
<i>Pre-condizioni:</i>	L'Utente è loggato nel sistema ed ha in precedenza effettuato una prenotazione
<i>Post-condizioni:</i>	<ul style="list-style-type: none"> <li>• È creata una nuova istanza di Messaggio <i>msg</i> all'interno del sistema (<i>creazione di istanza</i>)</li> <li>• È settata la tipologia di richiesta <i>msg.kind</i> a <i>defer</i> (<i>modifica di attributo</i>)</li> <li>• È associato al messaggio <i>msg</i> la Prenotazione <i>res</i> da posticipare (<i>associazione formata</i>)</li> <li>• È associato al messaggio <i>msg</i> l'amministratore tramite <i>msg.destinatario = res.sala.amministratore</i> (<i>associazione formata</i>)</li> <li>• È associato al messaggio <i>msg</i> l'utente <i>patient</i> che ne ha fatto richiesta nell'attributo <i>msg.patient</i> (<i>associazione formata</i>)</li> </ul>

---



---

**CONTRATTO 19: bindTutorForPatient**


---

<i>Operazione:</i>	<code>bindTutorForPatient(tutorName, tutorPassword)</code>
<i>Use Case:</i>	Registrare il paziente
<i>Pre-condizioni:</i>	Si richiede la registrazione di un paziente è minorenne
<i>Post-condizioni:</i>	<ul style="list-style-type: none"> <li>• Viene inserito un nuovo paziente nel sistema se la richiesta ha successo, creando l'istanza <i>patient</i> (<i>creazione di istanza</i>).</li> <li>• A <i>patient</i> vengono associati i parametri forniti precedentemente da <code>registerNewPatient</code> (<i>associazione formata</i>).</li> <li>• Viene associato il <i>tutor</i> indicato da <i>tutorName</i> e <i>tutorPassword</i> al <i>patient</i> nell'attributo <i>patient.tutor</i> (<i>associazione formata</i>)</li> </ul>

---

---

**CONTRATTO 20: printOnSupport**

---

<i>Operazione:</i>	<code>printOnSupport(element, givenSupport)</code>
<i>Use Case:</i>	Visualizzare lo storico delle prenotazioni
<i>Pre-condizioni:</i>	L'utente è autenticato all'interno del sistema e sono presenti delle prenotazioni o dei referti ad esse associate
<i>Post-condizioni:</i>	<ul style="list-style-type: none"><li>• Viene richiesto al sistema di stampare il dato referto o la data prenotazione, sul supporto richiesto</li></ul>

---

---

**CONTRATTO 21: reserveVisit**

---

<i>Operazione:</i>	<code>reserveVisit(description, admin)</code>
<i>Use Case:</i>	Effettuare una prenotazione
<i>Pre-condizioni:</i>	Il paziente è autenticato dal sistema
<i>Post-condizioni:</i>	<ul style="list-style-type: none"><li>• È creata una nuova istanza di Messaggio <i>msg</i> all'interno del sistema (<i>creazione di istanza</i>)</li><li>• È settata la tipologia di richiesta <i>msg.kind</i> a <i>reserve</i> (<i>modifica di attributo</i>)</li><li>• È associato al messaggio <i>msg</i> l'utente <i>patient</i> che ne ha fatto richiesta nell'attributo <i>msg.patient</i> (<i>associazione formata</i>)</li><li>• È associata al messaggio <i>msg</i> una descrizione <i>description</i> sulla richiesta della visita nell'attributo <i>msg.content</i> (<i>modifica di attributo</i>).</li><li>• È associato l'amministratore all'interno del messaggio nel campo destinatario: <i>msg.destinatario = admin</i></li></ul>

---

---

**CONTRATTO 22: registerNewPatientFromTutor**


---

<i>Operazione:</i>	<code>registerNewPatientFromTutor( tutorName, tutorPass)</code>
<i>Use Case:</i>	Registrare paziente
<i>Pre-condizioni:</i>	il paziente è presente nel sistema come tutore, ma non come Paziente.
<i>Post-condizioni:</i>	<ul style="list-style-type: none"> <li>• È inserito un nuovo paziente nel sistema (una volta che la richiesta è stata accettata), creando l'elemento <i>patient</i> con i parametri forniti dalla <code>registerNewPatientFromTutor</code>.</li> </ul>

---



---

**CONTRATTO 23: registerTutor**


---

<i>Operazione:</i>	<code>RegisterTutor( tutorName, tutorPass)</code>
<i>Use Case:</i>	Registrare paziente
<i>Pre-condizioni:</i>	Nessuna
<i>Post-condizioni:</i>	Viene inserito un nuovo tutor nel sistema, identificato dai parametri forniti. Un tutore potrebbe non essere necessariamente un utente del sistema.

---

**4.4.3. Ampliamenti al modello di dominio suggeriti.**

- È necessario creare una nuova classe `Slot` che contenga gli attributi di `DATA`, `ORA` e `STANZA`.
- È necessario inserire un attributo `PRENOTAZIONE` all'interno della classe `Messaggio`
- È necessario distinguere i messaggi come `MPaziente` e `MAmmministratore`, con `Messaggio` come loro generalizzazione.
- È necessario inserire una nuova astrazione tra *Paziente* ed *Amministratore* detta *Guest*

Ci riserviamo di effettuare queste modifiche nella fase successiva, in quanto in quel momento, ci appresteremo ad effettuare ulteriori modifiche al modello di dominio.

**4.4.4. Definizione delle priorità dei casi d'uso.** Come richiesto dal processo XP, dopo aver attentamente valutato la difficoltà dei vari casi d'uso, possiamo evidenziare le priorità di implementazione.

- Come Amministratore, voglio ricevere le richieste di visita in modo da poter notificare la prenotazione riservata per un paziente ed associarvi un grado di priorità.

priorità 10

- Come Utente, voglio scegliere un reparto in modo da prenotare una visita associata a quel tipo di servizio medico.

priorità 9

- Come Amministratore, voglio visualizzare le prenotazioni in modo da decidere in quale data effettuare le prenotazioni

priorità 8

- Come Utente, voglio essere in grado di registrare me stesso come Paziente Maggiorennne o come Tutore.

priorità 6

- Come Utente, voglio essere in grado di visualizzare le mie prenotazioni ed eventualmente i referti associati.

priorità 6

- Come Tutore, voglio essere in grado di registrare un paziente minorennne di cui sono responsabile.

priorità 5

- Come Amministratore, voglio gestire le prenotazioni in modo da associarvi o/e modificarvi i referti

priorità 5

- Come Amministratore, voglio visualizzare le prenotazioni in modo da ottenere l'insieme delle prenotazioni relativo ad un paziente, ed eventualmente i referti.

priorità 5

- Come Amministratore, voglio poter modificare i reperti medici del paziente.

priorità 5

- Come Utente, voglio essere in grado di effettuare l'eliminazione della prenotazione e di riceverne conferma.

priorità 5

- Come Utente, voglio essere in grado di effettuare uno spostamento della prenotazione già fissata.

priorità 5

- Come Paziente maggiorenne e tutore, voglio ottenere l'autenticazione allo scopo di ottenere per me o per il mio tutelato i servizi del sistema.  
priorità 4
- Come Utente, voglio ricevere le notifiche del sistema sulle operazioni di cancellazione, spostamento, prenotazione richieste all'amministratore.  
priorità 4

#### 4.5. Diario

- 31 Marzo:** Il gruppo si è riunito virtualmente su skype per suddividere il lavoro per la fase corrente. Il risultato è stato il seguente:  
**Fabian:** strutturazione dell'interfaccia grafica, GUI statechart diagram.  
**Giacomo:** progettazione della base di dati, redazione del documento  $\text{\LaTeX}$ , creazione delle schede CRC, diagramma di sequenza(amministratore) e modello di dominio.  
**Matej:** definizione delle priorità e strutturazione degli Statechart diagrams  
**Paolo:** creazione delle schede CRC, diagramma di sequenza (utente) e modello di dominio.  
 Ore di lavoro: 4.
- 2 Aprile:** Il gruppo si è riunito alle 12.00, e si è imposto una deadline per ottenere una prima bozza dei compiti relativi alla corrente fase prima dell'arrivo delle vacanze. pasquali. Di conseguenza si effettua una stima grossolana delle ore di lavoro. Ore di lavoro: 7.
- Festività Pasquali:** Quando possibile, i membri del gruppo si sono ritrovati su skype per fare report di eventuali integrazioni e/o modifiche apportate ai compiti per la fase corrente. Ore di lavoro: 8.
- 12 Aprile:** Il gruppo si è riunito "virtualmente" su SKYPE, per valutare il documento fornito dai committee per la precedente fase. Assente Paolo. Ore di lavoro: 5.
- 13 Aprile:** Il gruppo si è riunito "virtualmente" su SKYPE, per fare degli ultimi ritocchi al documento corrente. Ore di lavoro: 5.

## CAPITOLO 5

### Strutturazione dell'interfaccia grafica

#### Indice

<b>5.1. Analisi del Client</b>	<b>63</b>
5.1.1. Interfaccia di avvio	63
5.1.2. Gestione delle prenotazioni	66
5.1.3. Prenotazione della Visita	66
<b>5.2. Analisi del Server</b>	<b>69</b>

#### 5.1. Analisi del Client

**5.1.1. Interfaccia di avvio.** L'applicazione presenta un'interfaccia grafica semplice e intuitiva da usare. All'avvio ciò che vede l'utente è rappresentato in Figura 5.1.1 nella pagina successiva(a).

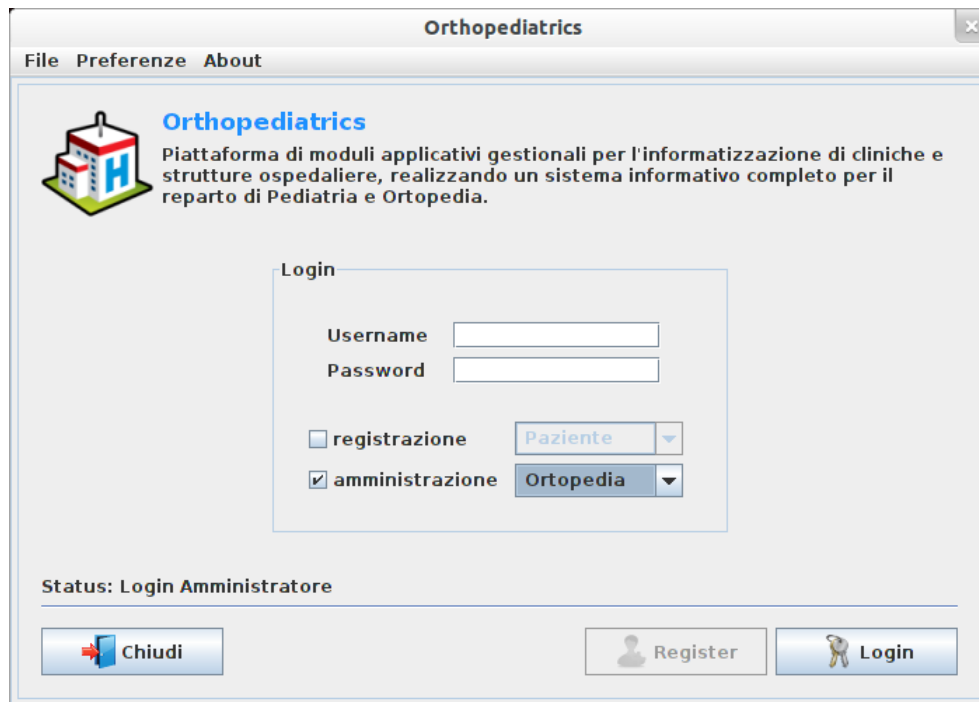
L'utilizzo di Orthopediatrics è riservato a due tipi di utenti: Pazienti e Amministratori. Per i pazienti, questi possono trovarsi in due stati all'interno del sistema: registrati oppure non registrati. Per effettuare la registrazione come paziente, è sufficiente selezionare la casella "*registrazione*" ed automaticamente viene abilitato il bottone "*Register*", premendo sul quale il paziente viene rimandato ad una nuova visualizzazione in cui dovrà effettuare l'inserimento dei dati personali, come mostrato in Figura 5.1.1 nella pagina seguente(b).

Al termine di questa procedura, in caso di errore, il paziente verrà informato di eventuali problemi; in caso di successo invece verrà rimandato alla schermata precedente in modo da effettuare l'autenticazione presso il sistema.

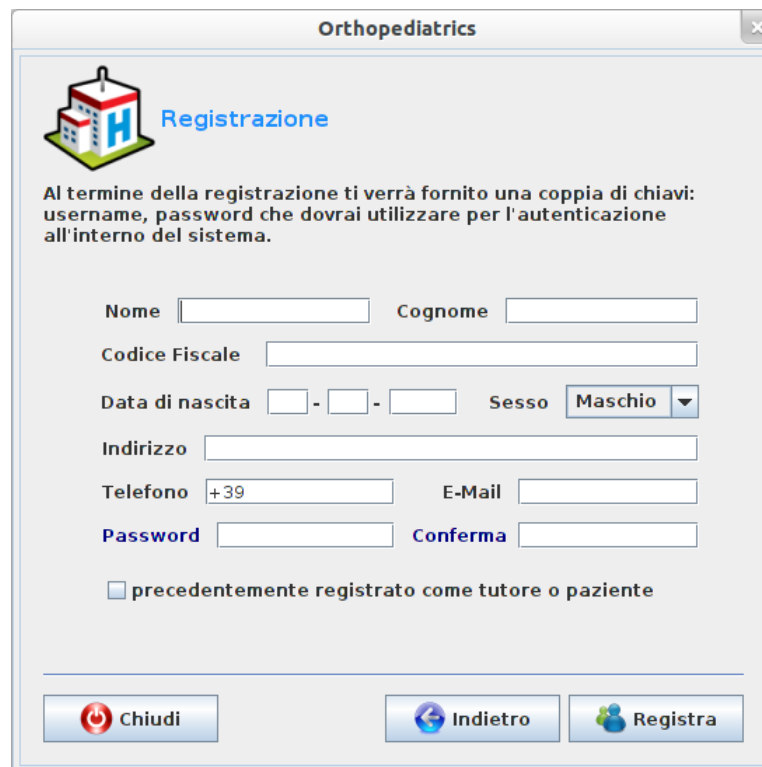
La fase di autenticazione avviene tramite *Username* e *Password*. Se trattassi di pazienti questi devono inserire semplicemente le loro credenziali e cliccare sul bottone "*Login*", dopo aver inserito nel campo *Username* il loro proprio codice fiscale.

Se l'utente che desidera accedere al sistema è un amministratore, questo deve necessariamente selezionare la casella "*amministratore*" abilitando in questo modo il menù a tendina della scelta del reparto, selezionando successivamente il reparto di appartenenza.

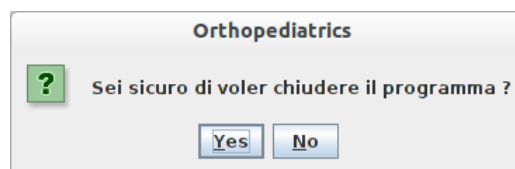
Un supporto agli utenti del sistema è fornito tramite *lo stato* ("**Status**") che informa l'utente, a seconda delle scelte e opzioni effettuate, del tipo di operazione che si sta eseguendo. In caso di problemi legati alla fase di login, lo



(A) UIStart.



(B) UIRegistration.



(C) UIExitMessage.

FIGURA 5.1.1. First UI Windows.



File

Orthopediatrics

Amministrare pazienti del reparto di Ortopedia

Oggi: Martedì 8 Maggio 2012

POSTPONE: 8 Maggio 2012 alle ore 11:53

Informazioni

ID Paziente: Fabian Priftaj

Descrizione: RICHIESTA DI SPOSTAMENTO

Il paziente Fabian Priftaj richiede lo spostamento della prenotazione n. 201252080 con le seguenti info:  
>> Prenotazione n. 201252080, confermata per il giorno 20 Maggio 2012, alle ore 8:00, in Sala A1, reparto di Ortopedia

Priorità

Selezionare la sala: Sala A1

Calendario

Gestione referti

Aggiorna

Cancella

(A) UIAdmin.

Orthopediatrics

Calendario delle prenotazioni settimanali  
Reparto di Ortopedia, Sala A1

Maggio 2012

<< Aprile

Giugno >>

Orario	Gio 10	Mer 16	Dom 20	Sab 26
08:00h - 09:00h			201252080	
09:00h - 10:00h				
10:00h - 11:00h	2012510100			
11:00h - 12:00h		2012516110		
12:00h - 13:00h				
13:00h - 14:00h				
14:00h - 15:00h				
15:00h - 16:00h				
16:00h - 17:00h				2012526160
17:00h - 18:00h				
18:00h - 19:00h				
19:00h - 20:00h				

Info

Cancella

Sposta

Conferma Prenotazione

(B) UICalendario (A Calendar View).

FIGURA 5.1.2. Administrator's UI Windows.

status informa l'utente di eventuali errori.

Dalla schermata iniziale l'utente può in qualunque momento decidere di chiudere l'applicazione semplicemente cliccando sul bottone "Chiudi", visualizzando la schermata di Figura 5.1.1 nella pagina precedente(c).

**5.1.2. Gestione delle prenotazioni.** Se l'utente che effettua l'autenticazione presso il sistema è un amministratore, allora l'interfaccia che il sistema mette a disposizione per la gestione dei pazienti è rappresentata nella Figura 5.1.2 nella pagina precedente(a).

Nella parte sinistra vengono visualizzate le richieste di prenotazione dei pazienti. Selezionando una singola richiesta viene aggiornato il campo "*Informazioni*" (nella parte destra) con alcune semplici informazioni di base riguardanti il paziente. Cliccando sul bottone **Altro** si viene rimandati ad una nuova schermata dove vengono visualizzate le informazioni complete del paziente.

L'amministratore in base alla descrizione della richiesta, stabilisce una priorità espressa in tre diversi colori: rosso, giallo, verde.

In base alla priorità attribuita alla richiesta l'amministratore deve stabilire il giorno in cui la visita deve avvenire. Cliccando sul bottone **Calendario** si viene trasferiti ad una nuova schermata rappresentata in Figura 5.1.2 nella pagina precedente(b).

Il calendario fornisce informazioni riguardanti un'unica sala, che deve essere scelta nella schermata precedente tra le sale disponibili nel menu a tendina.

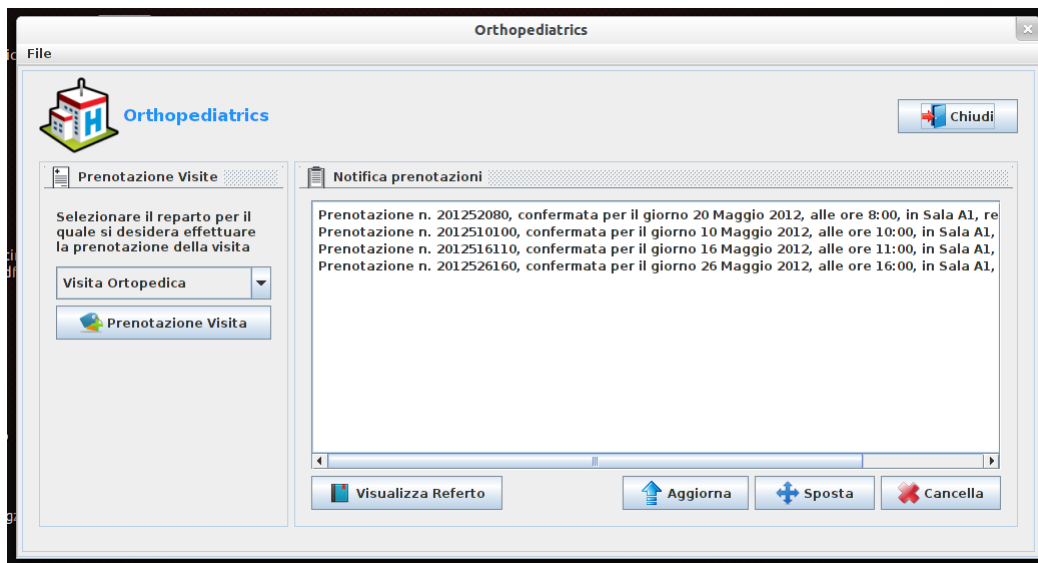
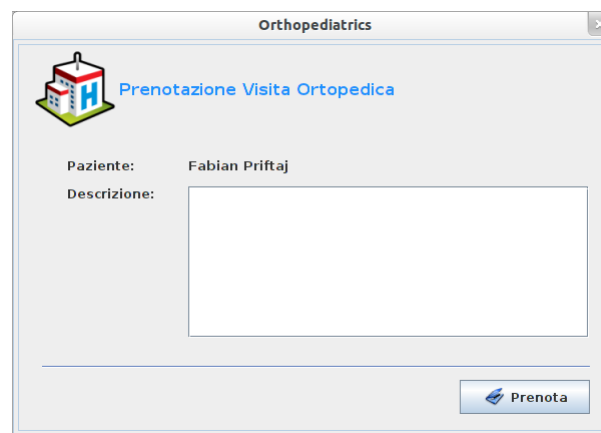
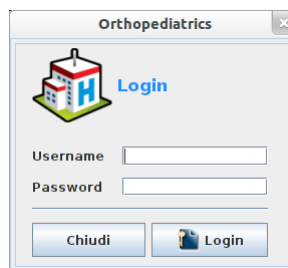
L'amministratore dopo aver selezionato la data opportuna, per confermare la richiesta di prenotazione deve cliccare su **Conferma**, altrimenti per tornare alla schermata precedente è sufficiente cliccare su **Indietro**. In caso di informazioni mancanti, prima di procedere l'amministratore verrà informato tramite messaggi di pop-up.

**5.1.3. Prenotazione della Visita.** La fase di prenotazione della visita è un'attività svolta dai pazienti registrati presso il sistema. Il paziente può prenotare, cancellare o spostare una visita utilizzando la seguente interfaccia messa a disposizione dal sistema (Figura 5.1.3 a fronte(a)).

Inizialmente il paziente deve indicare il tipo di visita da effettuare: *ortopedica* o *pediatrica*. In base a ciò viene selezionato il reparto desiderato nell'area "*Prenotazione Visite*". Per prenotare la visita, dopo aver scelto il reparto, è sufficiente cliccare su **Prenota**. Successivamente viene mostrata una nuova finestra in cui è visualizzato un campo di testo che il paziente dovrà obbligatoriamente riempire con una descrizione dettagliata. Tale descrizione deve essere sufficientemente chiara da permettere agli amministratori di valutare le condizioni del paziente.

Il sistema è utilizzato da paziente *maggiorescenti* e *minorescenti*. In caso di pazienti con età inferiore a 18 è previsto un tutore. Questo deve obbligatoriamente essere registrato all'interno del sistema. Questo tutore è associato in fase di registrazione del paziente: Ogniqualvolta un paziente minorenne tenta di prenotare una visita è richiesto dal sistema un'ulteriore autenticazione, che riguarda il tutore, tramite la Figura 5.1.3 nella pagina successiva(c).

Tutte queste possibili interazioni sono riassunte all'interno dei diagrammi degli stati rappresentato in Figura 5.1.4 a pagina 68.

(A) *UIPatient.*(B) *UIPrenotazione (A Reservation View).*(C)  
*UITutor-Login.*FIGURA 5.1.3. *User's UI Window.*

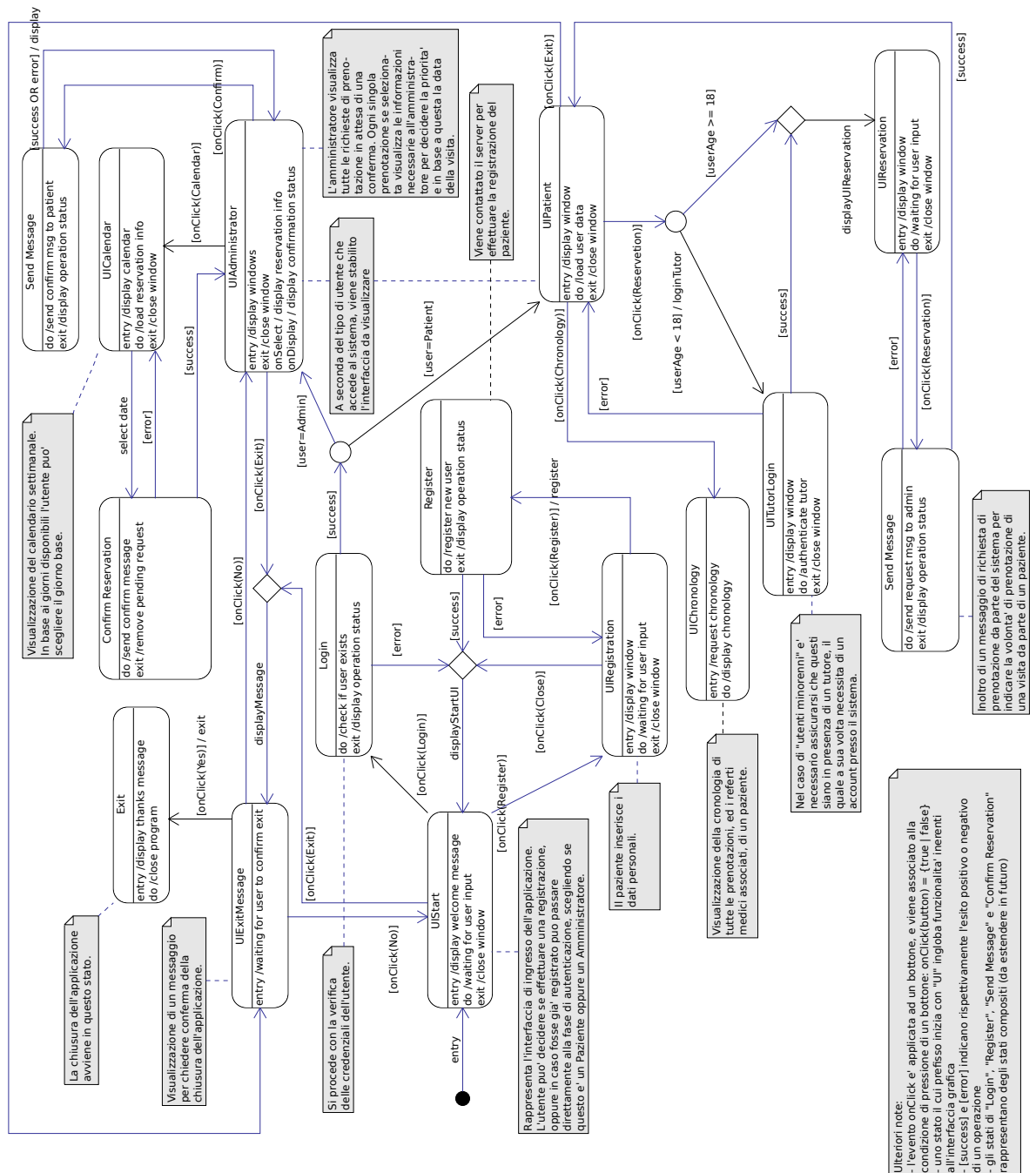


FIGURA 5.1.4. *Client UI Statechart. This statechat has the aim to link all the GUI screenshot that will follow.*

## 5.2. Analisi del Server

Il server rappresenta l'interfaccia messa a disposizione dal sistema per lo smistamento e la gestione delle richieste proveniente da parte dei pazienti e dagli amministratori. La gestione del server è facilitata da un interfaccia grafica come da Figura 5.2.1(a):

Il server può trovarsi in tre stati diversi, che sono visualizzati dalla label "Status":

**ready:** questo stato si ha nel momento in cui viene visualizzato l'interfaccia grafica (avvio del server)

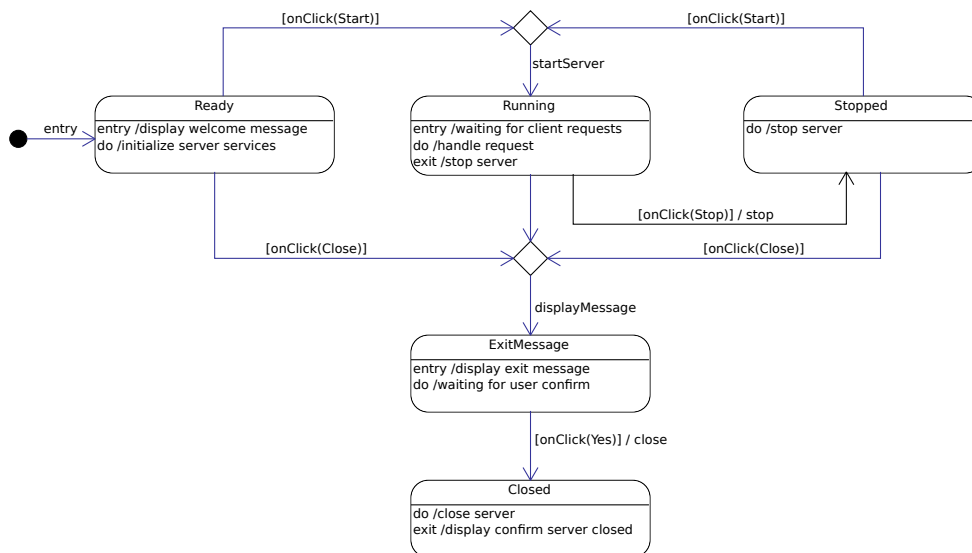
**running:** rappresenta lo stato che si raggiunge cliccando sul bottone "Start"

**stopped:** il server viene messo in pausa e non viene gestita nessun tipo di richiesta

Cliccando sul pulsante **Exit** il server termina. Questa descrizione può essere



(A) *UIServer*.



(B) *UIServer Statechart Diagram*.

FIGURA 5.2.1. *A Server Overview*.

sintetizzata mediante il diagramma degli stati illustrato in Figura 5.2.1 nella pagina precedente(b).

## CAPITOLO 6

### Revisioni delle stime precedenti

#### Indice

---

##### 6.1. Ulteriore estensione dei Casi d'Uso

71

##### 6.1. Ulteriore estensione dei Casi d'Uso

Facendo riferimento al Testo dei Casi d'Uso della Sottosezione 2.3.2 a pagina 25, dobbiamo sottolineare la necessità di effettuare delle estensioni al caso d'uso *Registrare il paziente*. Per completezza, riportiamo qui l'intero testo del caso in questione con le modifiche apportate, utilizzando sempre lo stile espositivo proposto dal Larman.

**Nome del caso d'uso:** Registrare il paziente

**Attore primario:** L'utente (del sistema): il paziente maggiorenne od il tutore del paziente minorenn

**Parti interessate ed interessi:** L'utente: vuole registrare o sé stesso, o il paziente minorenn di cui ha tutela

**Pre-condizioni:** Nessuna

**Garanzia di successo:** Il sistema visualizzerà all'utente un messaggio di avvenuta registrazione

**Scenario principale di successo:** (1) L'utente sceglie l'opzione di registrazione

(2) L'utente inserisce i dati del paziente che dovrà ricevere le cure

(3) L'utente richiede la registrazione

(4) L'utente riceve la notifica di avvenuta registrazione con una notifica tramite il sistema

**Estensioni:**    **Estensione A:**    4: L'utente richiede la registrazione con successo in quanto è un paziente maggiorenne

**Estensione B:**    4: (1) Il sistema richiede all'utente l'associazione delle credenziali del tutore, in quanto il paziente in fase di registrazione è minorenn

(2) Il tutore inserisce le proprie credenziali

(3) Il sistema conferma la presenza del tutore all'interno del sistema

- (4) Il sistema effettua la registrazione del paziente minorenni e dell'associazione con il tutore utente del sistema

**Estensione C:**     **4:** Il sistema informa l'utente della mancata registrazione del paziente, specificando quali siano i campi che sono stati ignorati in fase di registrazione

**Estensione D:**     **4B (3):** (1) Il sistema informa il tutore che lui stesso non esiste all'interno del sistema  
(2) Il sistema annulla la registrazione del minore

**Estensione E:**     **2:** (1) In quanto il paziente in questione che sta effettuando la registrazione è già presente all'interno del sistema come tutore (*precondizione*), ed è quindi maggiorenne, richiede di registrarsi come tutore importando quei dati.  
(2) Si autentica come Tutore nel sistema  
(3) L'utente riceve una notifica di conferma dal sistema

**Estensione F:**     **B2:** (1) Il nuovo tutore inserisce le proprie credenziali, specificando che esiste come Paziente Maggiore, autenticandosi come tale.  
(2) Il sistema conferma l'esistenza del Paziente Maggiore dandone notifica  
(3) Il sistema crea un nuovo Tutore dalle credenziali del Paziente Maggiore del quale si è comprovata l'esistenza  
(4) Il sistema conferma la registrazione del paziente minorenni e vi associa il nuovo tutore

**Estensione G:**     **B2:** (1) Il nuovo tutore specifica che vuole essere registrato nel sistema  
(2) Il Tutore immette le sue informazioni all'interno del sistema  
(3) Il sistema conferma la presenza del nuovo tutore all'interno del sistema  
(4) Il sistema effettua la registrazione del paziente minorenni e dell'associazione con il tutore utente del sistema

**Requisiti non funzionali:**     ◇ Il sistema informativo deve poter prevedere autenticazione per i vari utenti e per l'amministratore, ed in più il paziente minorenni può accedere solamente mediante la supervisione del tutore; è richiesta autenticazione sia per il paziente minorenni, sia per il tutore.

---



## **Parte 4**

# **Documento Quarto**



## CAPITOLO 7

### Sviluppo I

#### Indice

7.1. Premessa	75
7.2. Raffinamento del Modello di Dominio	76
7.2.1. Premesse	76
7.2.2. Progettazione agli oggetti	78

#### 7.1. Premessa

In questo documento, cerchiamo di apportare un'ultima e finale modifica al modello di dominio, allo scopo di rendere possibile la creazione dei seguenti documenti, che portano globalmente alla prima stesura di codice

◇ *Raffinamento del modello di dominio*

Questo documento è alla base delle documentazioni che verranno fornite: si vogliono applicare a quest'ultimo alcuni *Design Pattern(s)*, allo scopo di prevenire successive problematiche di implementazione (v. SottoSottosezione 7.2.1.3 a pagina 77). In seguito, si procederà con l'assegnamento dei metodi alle rispettive classi tramite l'applicazione dei pattern GRASP (v. Sezione 7.2 nella pagina successiva). In questa fase sono già contemplate le considerazioni che emergono dai due punti successivi.

◇ *Analisi del database*

In base al raffinamento del modello di dominio sopra proposto, si effettua una definizione di un modello ER per la nostra base di dati, e conseguentemente si produce il codice SQL relativo (v. Capitolo 8 a pagina 91). Tramite l'analisi del database introduciamo per la prima volta il *framework* EJP all'interno dei nostri elementi di sviluppo, in base alla quale effettuare il mapping tra modello ER (*Entity-Relationship*) ed OO (*Object Oriented*). In base a queste considerazioni, riteniamo conseguentemente inutile lo sviluppo di un server come proposto nella fase precedente (v. Sezione 5.2 a pagina 69), in quanto già questo *framework* risolve a nostro parere, moltissimi problemi di implementazione. (v. Sezione 8.6 a pagina 98).

## 7.2. Raffinamento del Modello di Dominio

**7.2.1. Premesse.** In seguito ai primi suggerimenti forniti nella Sezione 4.4.3 a pagina 60, si è iniziato a ristrutturare il modello di dominio.

- Per quanto concerne lo “Slot” orario indicato nella nostra precedente visualizzazione, questo sarà un’astrazione delle informazioni di data, ora e prenotazione già presente all’interno come campi di `Prenotazione`

**7.2.1.1. Strutturazione dei campi `Messaggio`.** Dopo aver ribadito che “una richiesta di prenotazione (inviata appunto tramite messaggio) non è ancora una prenotazione” (la quale sarà di un tipo di dato `Prenotazione`), procediamo ad analizzare i campi necessari per le due tipologie di messaggi, poiché dobbiamo poter modellare le nostre classi `MAmmministratore` e `MPaziente`, i quali rappresentano rispettivamente i messaggi inviati dagli amministratori e quelli inviati dai pazienti, possiamo fare le seguenti considerazioni:

- Anche se sarebbe possibile distinguere per ogni tipo di messaggio, `MPaziente` ed `MAmmministratore` tra tipologie di *prenotazione*, *posticipo* e *cancelazione* ed analoghe conferme lato amministratore, preferiamo mantenere distinti i valori come analizzato nel primo Documento dell’Architettura software pubblicato (v. Sezione 4.4 a pagina 44). Il tipo della richiesta o della notifica viene quindi specificato nel campo *kind* del messaggio.

**MPaziente:** Per i messaggi paziente, possiamo sottolineare che:

- ◊ I messaggi per l’amministratore costituiscono le richieste (es. di prenotazione) che devono essere gestite dagli amministratori: una volta che queste sono state gestite, il sistema provvederà in automatico a rimuoverle dal sistema.
- ◊ Per revocare, annullare o posticipare la visita, non è richiesto l’inserimento di testo.
- ◊ Unicamente per effettuare una richiesta di prenotazione, non è necessario associare al messaggio una prenotazione, in quanto questa deve essere ancora creata dall’Amministratore. Negli altri due casi di richiesta è invece necessaria, in quanto bisogna specificare su quale richiesta si stia facendo l’interrogazione.
- ◊ L’inserimento di una descrizione è necessaria unicamente nella descrizione della richiesta lato amministratore.
- ◊ Il mittente è il paziente che effettua la richiesta.
- ◊ Il destinatario è l’amministratore responsabile della gestione dei messaggi per quel reparto nel quale l’utente chiede di essere visitato.

**MAmmministratore:** Per i messaggi amministratore, possiamo sottolineare che questi verranno generati automaticamente dall’interazione dello stesso con l’interfaccia grafica:

- ◇ Per notificare la revoca di una prenotazione, è necessario utilizzare il campo `testo`, in quanto è comunque necessario memorizzare le informazioni di una prenotazione che è stata rimossa.
- ◇ Il mittente è l'amministratore che sta effettuando la risposta.
- ◇ Il destinatario è il paziente che riceverà la notifica.
- ◇ Unicamente nell'invio di una notifica di cancellazione, è richiesto specificare la prenotazione che è stata cancellata: negli altri casi questo campo è di per se inutilizzato, e lasciato per eventuali estensioni.

Ulteriori dettagli sulla definizione degli attributi necessari alle classi, sono forniti all'interno dell'Analisi Logica del Database (v. 8.3 a pagina 92).

7.2.1.2. *Implicazioni dell'utilizzo del framework "EJP"*. Possiamo sottolineare che l'utilizzo di un *framework per la persistenza* come EJP<sup>1</sup>, di cui parleremo dettagliatamente nella Sezione 8.6 a pagina 98, possa effettivamente demandare la gestione della rappresentazione delle singole classi al framework: conseguentemente all'applicazione di un design pattern *Singleton* per l'accesso al database in questo contesto, non si ha conseguentemente alcun incremento di responsabilità nella gestione del cambiamento delle classi. Questo framework infatti si occupa automaticamente di effettuare il mapping tra modelli OO e ER.

7.2.1.3. *Applicazione di design pattern*. In seconda analisi, possiamo passare all'applicazione dei design pattern all'interno di questo contesto.

**Factory Method:** Abbiamo applicato questo design pattern per effettuare il "login" dei pazienti e per effettuare la loro creazione: tuttavia se dovessimo applicare il **Single Responsibility Principle** in questo contesto, dovremmo dividere il "login" dalla loro registrazione, in quanto sono due aspetti differenti dello stesso problema. Tuttavia, in questo modo, si rischierebbe di riprodurre tre volte questa gerarchia: una per la definizione dei dati da creare (5 classi), ed altre due per definire le Factory di autenticazione e di creazione: preferiamo per tanto tenere unite queste responsabilità. L'elemento radice della *Factory* è `GuestC`.

Si vuole inoltre sottolineare come i metodi ivi definiti *login* e *create* siano stati inseriti puramente per sottolineare la necessità di riflettere anche quivi la gerarchia imposta per gli utenti.

**Facade:** Per effettuare la visualizzazione delle prenotazioni e dei referiti, abbiamo identificato la classe `ViewSupport` allo scopo di riunire le diverse logiche di visualizzazione implementate.

**Singleton:** Per garantire **High Cohesion**, allo scopo di fornire un accesso al database in un unico punto dell'applicazione, .

---

<sup>1</sup>Easy Java Persistence: <http://www.easierjava.com/>

Dopo queste ultime valutazioni, possiamo effettuare una prima revisione del Modello di Dominio, come presentato in Figura 7.2.8 a pagina 88.

**7.2.2. Progettazione agli oggetti.** Procediamo al raffinamento del modello di dominio continuando con la documentazione dell'Architettura Software. Ora andremo a dettagliare le interazioni del sistema, puntualizzando quali classi del modello del dominio risponderanno alle procedure identificate nella fase precedente (v. Sezione 4.4 a pagina 44).

Per effettuare quest'analisi, applicheremo il pattern *Controller*, ovvero ci chiederemo *quale sia il primo oggetto, oltre allo strato UI* (indicato genericamente nei nostri diagrammi tramite una generica classe non appartenente al nostro Modello di Dominio *UIState*), *a ricevere ed a coordinare un'operazione di sistema*. In questa sede introdurremo anche le funzioni che saranno utilizzate allo scopo di effettuare le chiamate alla classe *DB*, allo scopo di interagire con il Database.

**7.2.2.1. Utente: Vista dei Casi d'Uso.**

**bindTutorForPatient:** Si utilizza **Creator**: la responsabilità di associare il tutore al paziente è delegata a *CPaziente*. Questa operazione viene effettuata solamente all'atto della registrazione, nel caso specifico in cui il paziente sia riconosciuto come minorenne. La classe *CPaziente* ha la responsabilità di creare istanze valide di pazienti, ragion per cui è sempre essa che prima verifica la fattibilità interpellando il database. Naturalmente con la *bindTutorForPatient* non vengono creati il pazienti minorenni.

**deferReservation:** Si utilizza **Information Expert**: la responsabilità di creare il messaggio per l'amministratore è delegata alla classe *ModelPatient*. Questa operazione e la *retireReservation* sono quasi identiche: ciò che cambia sarà solamente la tipologia del messaggio inviato all'amministratore. Come già detto precedentemente, si presuppone che l'implementazione della classe *Messaggio* contenga un campo per la discriminazione delle categorie di messaggi. Non è stata modellata l'eliminazione della prenotazione in questo stadio, poiché la gestione delle prenotazioni è esclusivamente dell'amministratore. Quanto appena detto non esclude la possibilità che l'implementazione finale elimini già in questa fase la prenotazione, pur fornendo nel messaggio i dati necessari a ricreare la nuova prenotazione.

**fetchReport:** Vedi *fetchReport*. Nel caso specifico *ModelPatient* non ha bisogno di parametri: infatti è sufficiente fornire al DB l'identificativo interno del paziente corrente per ottenere tutte le prenotazioni. Si utilizza **High Cohesion**: di fatti a DB viene affidata la responsabilità di prelevare anche le informazioni per il paziente.

**login:** Si utilizza **Information Expert**: La responsabilità di controllare le credenziali del paziente è delegata a *CPaziente*. Quando l'utente

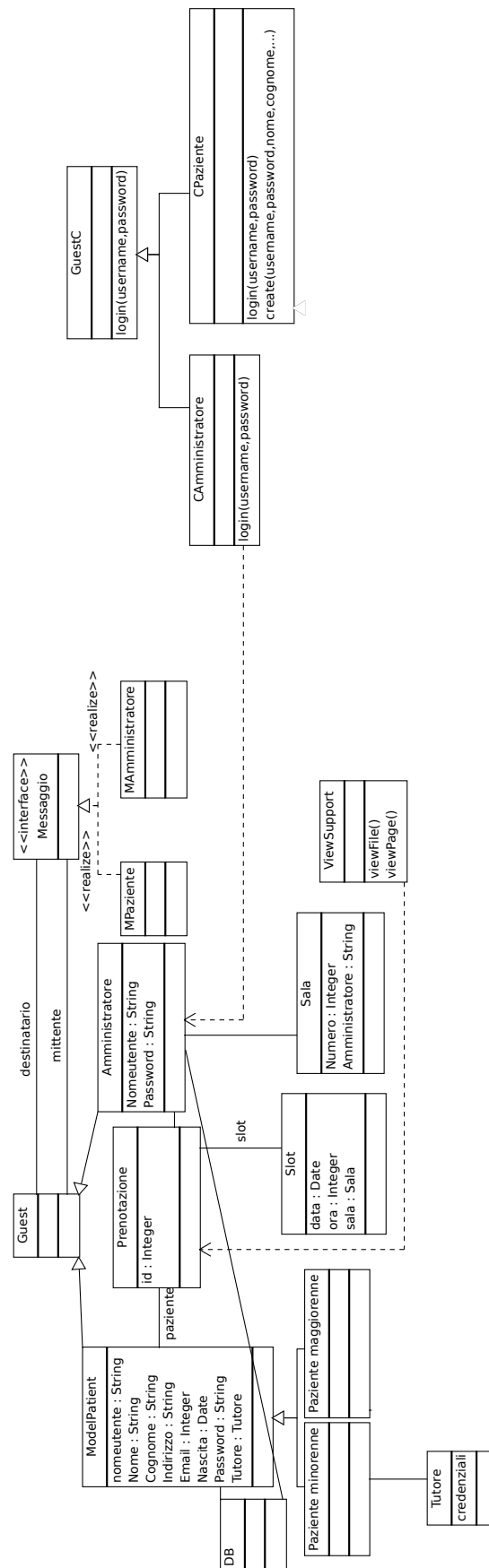


FIGURA 7.2.1. First Revision of the Domain Model presented in Section 4.3 a pagina 42.

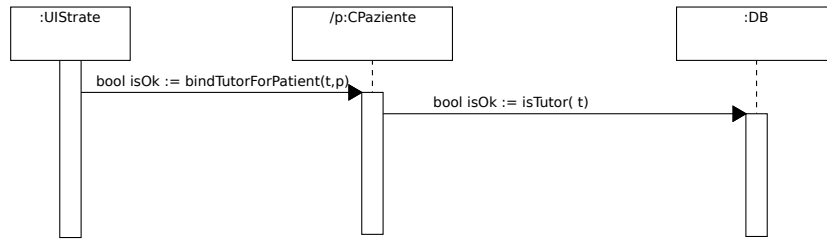
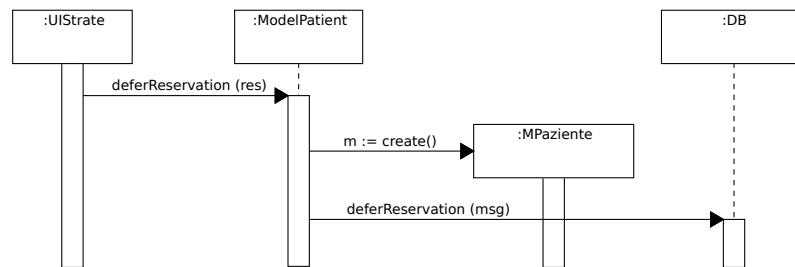
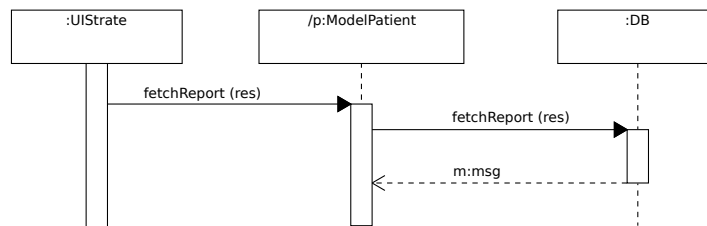
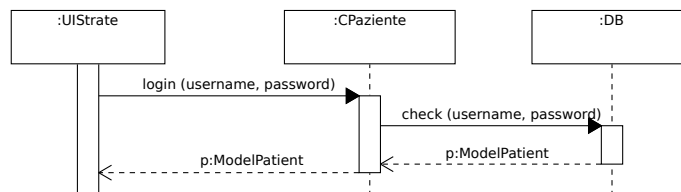
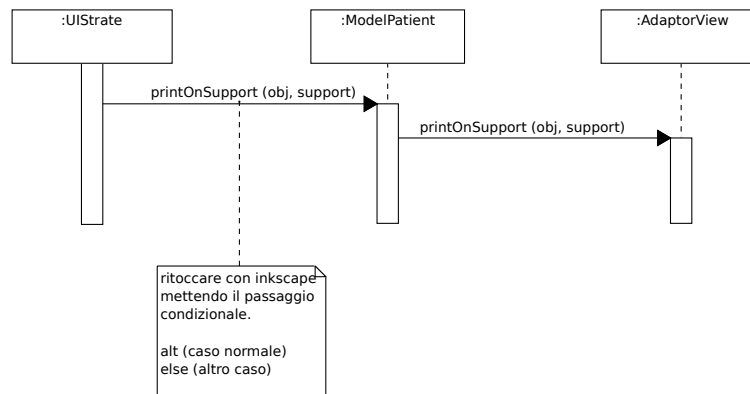
(A) *bindTutorForPatient*.(B) *deferReservation*.(C) *fetchReport*.(D) *login*.(E) *printOnSupport*.

FIGURA 7.2.2. Use Case Realization for Guest's view (1)



deve ancora autenticarsi, si troverà in uno stato non ancora definito, come ospite del sistema. In caso di successo ciò che viene restituito sono tutte le informazioni che accertano il paziente nel sistema, per questo ad ogni login viene creato il paziente verificato.

**printOnSupport:** In questo caso il metodo di stampa vero e proprio dipenderà dal supporto scelto. Data la varietà di supporti una scelta vantaggiosa è l'utilizzo di un Facade. In questo modo la scalabilità *in the many* richiederà solo l'utilizzo della classe appropriata per la visualizzazione del contenuto finale. Si fa ricorso di **Polymorphism**: di fatti, in base al supporto utilizzato per effettuare la "stampa", ViewSupport avrà poi la responsabilità di decidere su quale supporto stampare.

**registerNewPatient:** Si utilizza **Creator**: di fatti questa la classe che ha la responsabilità di creare un nuovo paziente è CPaziente. In questo caso può si supporre che i dati passati al creatore della classe paziente siano validi ( perché vi sono stati controllati precedentemente). Si ritiene necessario attribuire alla classe CPaziente anche questa responsabilità, in quanto è responsabile della corretta istanziatura delle classi Paziente. L'azione di registrazione potrebbe quindi fallire sia in seguito ad un valore non accettabile dal creatore, sia in seguito ad una doppia registrazione dell'utente. Non è infatti possibile che lo stesso paziente possa essere stato registrato 2 volte.

**registerNewPatientFromTutor:** Vedi registerNewPatient. Nel caso specifico anziché costruire il nuovo paziente attraverso i dati regolari, viene fornita la possibilità di specificare le credenziali di un tutore per registrare il paziente.

**reserveVisit:** Si può fare un discorso analogo a quanto fatto per deferReservation. Nel caso specifico viene fatta la richiesta di visita, quindi il paziente aspetterà una qualche notifica da parte dell'amministratore.

**tutorCredentials:** Si utilizza **Information Expert**: la classe che si occupa di controllare le credenziali del paziente è Paziente minorenn. Questa caso coinvolge almeno le azioni di registrazione, il login, richiesta di prenotazione/annullamento/posticipo di una visita, da parte dell'utente minorenn.

7.2.2.2. *Amministratore: Vista dei Casi d'Uso.* Riportiamo qui di seguito le funzioni che andremo a contestualizzare all'interno del Dominio del nostro problema, e descritti nella SottoSottosezione 4.4.1.2 a pagina 45:

- ◇ getFreeSlotsAfter
- ◇ fetchReservations
- ◇ addReport
- ◇ postponeReservs
- ◇ Reserve
- ◇ fetchPendantReq

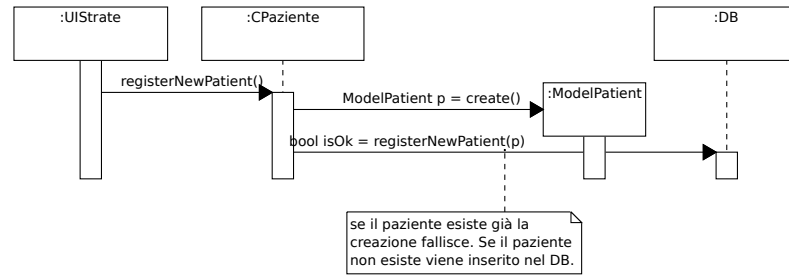
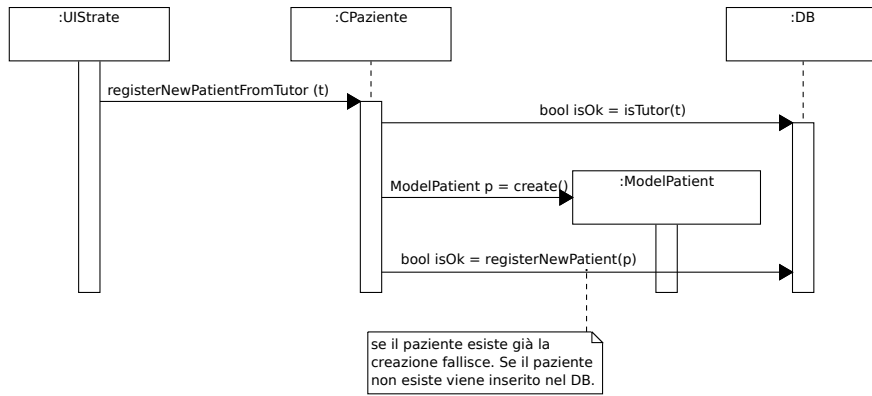
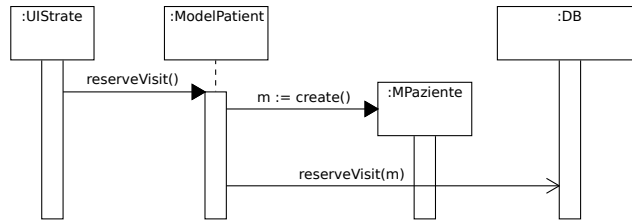
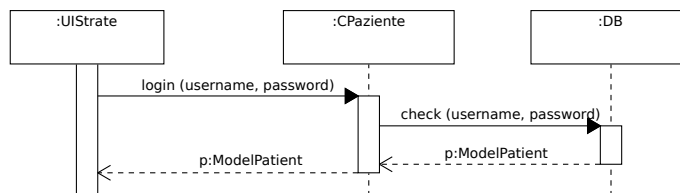
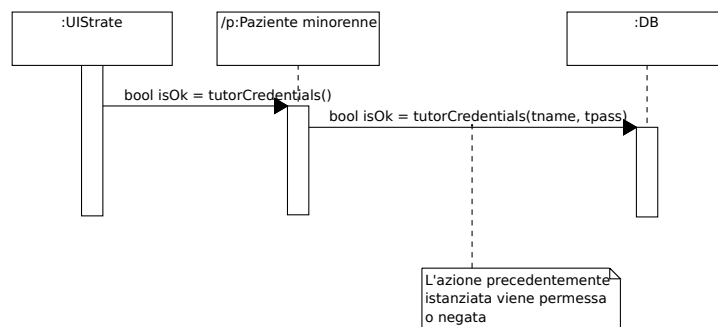
(A) *registerNewPatient*.(B) *registerNewPatientFromTutor*.(C) *reserveVisit*.(D) *login*.(E) *tutorCredentials*.

FIGURA 7.2.3. Use Case Realization for Guest's view (2)

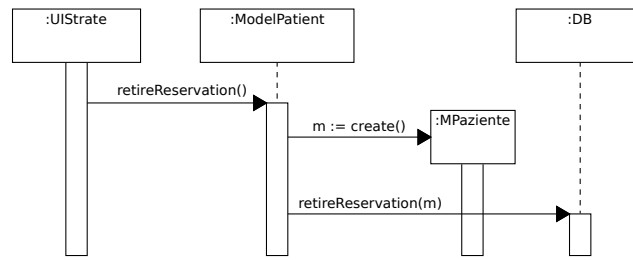
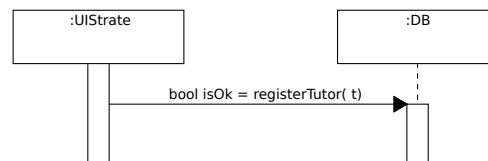
(A) *retireReservation.*(B) *registerTutor.*

FIGURA 7.2.4. Use Case Realization for Guest's view (3)

- ◇ Login
- ◇ fetchRemoveReq
- ◇ fetchPostponeReq
- ◇ destroyRequestPostpone
- ◇ freeSlotReserv

**getFreeSlotsAfter:** Possiamo evidenziare come questa operazione richieda un accesso al database, per ottenere quali siano gli slots occupati. Tuttavia dobbiamo sottolineare come, all'interno della base di dati che memorizzerà permanentemente le operazioni, siano memorizzati per ovvie ragioni solamente i dati relativi agli slot occupati, e non quelli liberi. Conseguentemente, è necessario effettuare un computo ulteriore dopo aver prelevato gli slots occupati lato database: la responsabilità di questo computo tuttavia non può essere delegato ad `DBAdmin`, per non attribuire anche il compito di effettuare la gestione del tempo. In questo caso si rende quindi necessario aggiungere una classe accessoria `DBTime`, che medi con la precedente per effettuare un primo computo degli `Slot` temporali liberi.

**fetchReservation:** In questo caso facciamo utilizzo del **GRASP Expert**, in quanto ci chiediamo quale sia l'elemento che contenga le informazioni relative a quale sia l'amministratore corrente nel sistema. Questa risposta ci viene fornita dalla classe `Amministratore`. Inoltre, allo scopo di

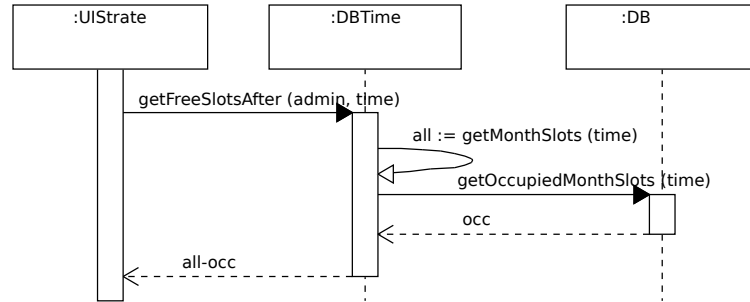
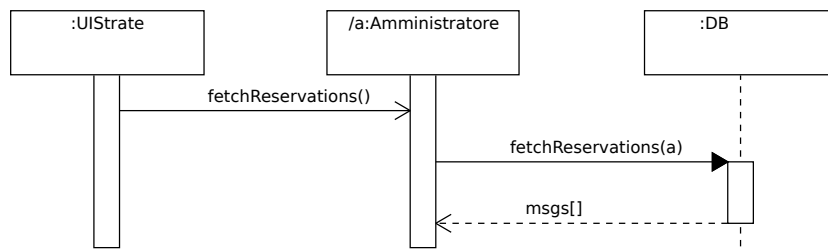
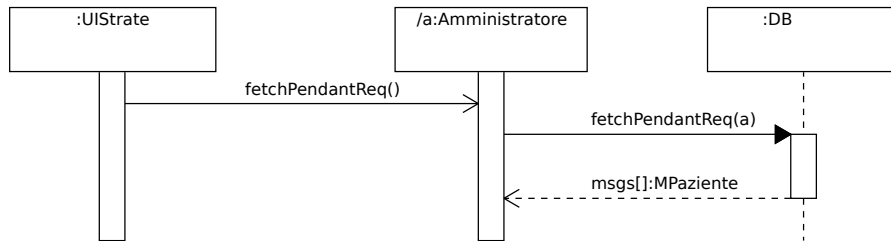
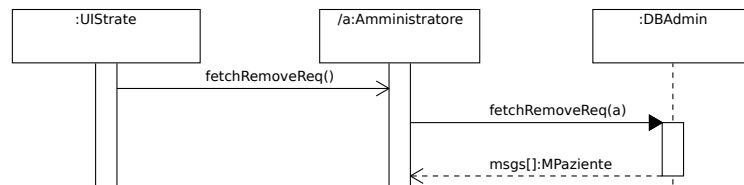
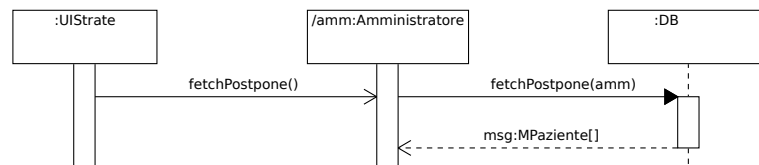
(A) 1. *getFreeSlotsAfter*.(B) 2. *fetchReservation*.(C) 6. *fetchPendantReq*.(D) 8. *fetchRemoveReq*.(E) 9. *fetchPostponeReq*.

FIGURA 7.2.5. Use Case Realization for Admin's view (1)

effettuare il **Principio di Separazione Controllo-Interrogazione**, identifichiamo una funzione *fetchReservations(: admin)* verso la DBAdmin, la quale si occupa di ottenere il risultato senza modificare lo stato di Amministratore (interrogazione), ed attribuiamo alla funzione in discussione un comando, che invece ne effettua la modifica di stato.

In questo caso, la classe **Expert**, identifica in qualche modo anche l'oggetto radice del nostro sistema. Considerazioni analoghe possono essere effettuate per i contratti *fetchPendantReq* (**Contratto 06**), *fetchPostponeReq* (**Contratto 08**) e *fetchPostponeRew* (**Contratto 09**), che conseguentemente non tratteremo, visto che il problema che si presenta in questi contesti è risolvibile in un modo del tutto identico con quello già enunciato.

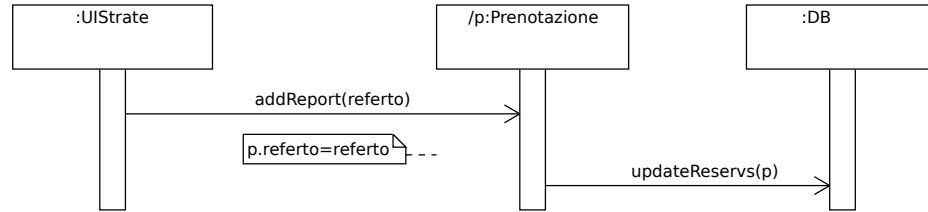
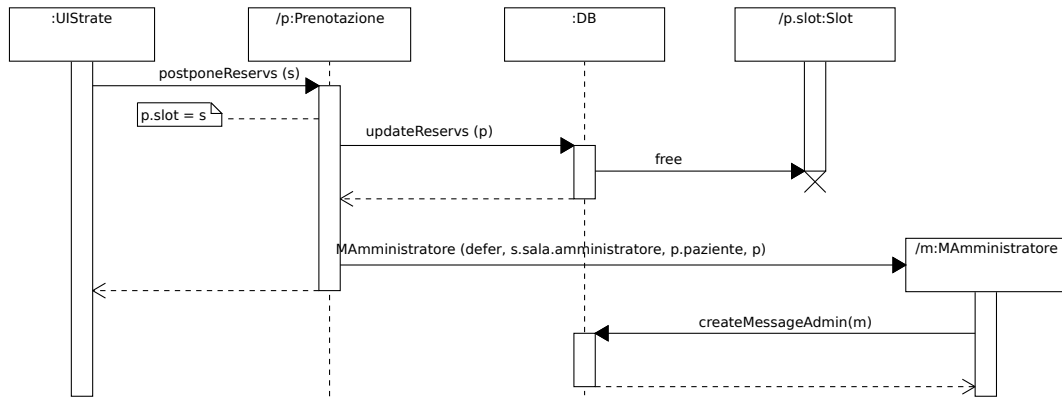
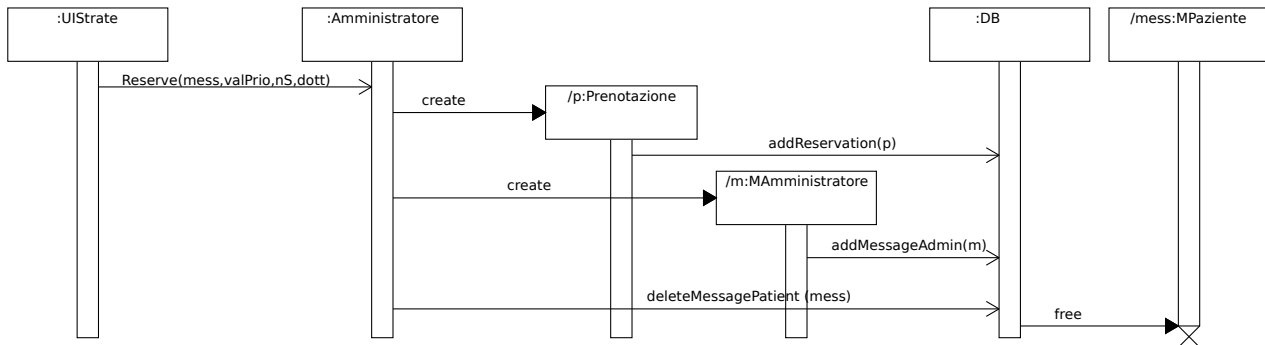
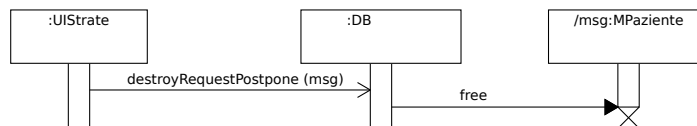
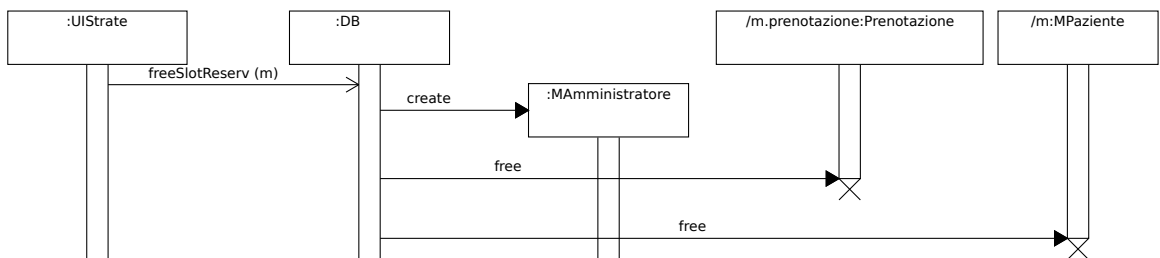
**addReport:** In questo caso, obbedendo al GRASP **Controller**, l'oggetto che è ricevitore del comando di questo contratto è la stessa Prenotazione: possiamo quindi trasformare la funzione *addReport( $\alpha, \beta$ )* nel metodo  *$\alpha.addReport(\beta)$* : questa operazione modificherà lo stato interno dell'oggetto, che verrà aggiornato a sua volta all'interno del database.

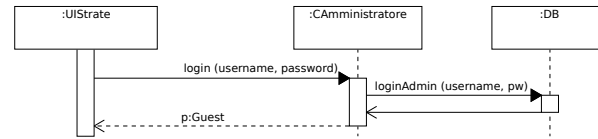
**postponeReservs:** In ottemperanza all'osservazione effettuata precedente, trasformiamo la funzione *postponeReservs( $\alpha, \beta$ )* in modo analogo nel metodo  *$\alpha.postponeReservs(\beta)$* . In particolare possiamo osservare tramite il diagramma associato come si possano creare oggetti all'interno del database (ovvero creando prima l'oggetto, e trasferendolo successivamente ad una funzione gestita dal framework), e come sia possibile eliminarli (ovvero tramite il passaggio di un'istanza dell'oggetto stesso che si conosce essere contenuto come record di una tabella del database).

**Reserve:** In questo caso abbiamo che l'elemento radice del nostro sistema è effettivamente la classe Amministratore: in questo modo è inoltre possibile semplificare la procedura di estrazione dell'amministratore (informazione contenuta all'interno della classe stessa), senza doverla per forza ottenere dal messaggio in lettura.

**destroyRequestPostpone:** In questo caso abbiamo che l'oggetto che rappresenta il sottosistema principale, sempre in obbedienza al principio **Controller**, è di fatti la stessa classe DBAdmin: l'unico compito che vi si richiede è di fatti quello dell'eliminazione del messaggio.

**freeSlotReserv:** Analogamente a quanto discusso precedentemente, la classe bfController è ancora DBAdmin: in questo caso tuttavia si aggiunge una responsabilità alla classe, che è quella della creazione del messaggio, che comunque richiede l'interazione con il framework EJP.

(A) 3. *addReport.*(B) 4. *postponeReservs.*(C) 5. *Reserve.*(D) 10. *destroyRequestPostpone.*(E) 11. *freeSlotReserv.*FIGURA 7.2.6. *Use Case Realization for Admin's view (2)*

FIGURA 7.2.7. *Login - Use Case Realization for Admin's view (3)*

**Login:** Analogamente a quanto discusso precedentemente, la responsabilità nell'effettuazione della procedura di Login è affidata alla classe costruttrice (**Creator**) *CAmmministratore*, in quanto ha la responsabilità di ottenere le istanze di *Amministratori*.

Possiamo a questo punto mostrare in Figura 7.2.8 nella pagina seguente una seconda versione del Modello di Business: per praticità di lettura abbiamo preferito tenere in questa versione unicamente i metodi propri di ogni singola classe, diversamente da quanto già effettuato in Figura 7.2.8 nella pagina successiva.

7.2.2.3. *Vista dei Dati.* il DFD in Figura 7.2.9 a pagina 89 rappresenta flow di dati più importanti dell'applicazione, ovvero i dati corrispondenti alle richieste dei vari servizi, prenotazioni e scambio di messaggi tra pazienti ed amministratori, che servono "come mezzo di trasporto"<sup>2</sup> di informazioni tra clienti ed amministratori. Si può osservare che la comunicazione è indiretta con uno "storage mediano". Una generica richiesta viene opportunamente trasformata in un messaggio che viene inserito in DB. *Amministratore* preleva questi messaggi ed in base al loro contenuto decide operazione da svolgere. Al termine di ogni operazione viene generato un messaggio per informare client dell'avvenuta gestione della richiesta ed anche questo viene inserito in DB da dove viene prelevato al login del paziente.

il DFD in Figura 7.2.10 a pagina 90(b) rappresenta flow e storage di dati personali dei pazienti mentre DFD in Figura 7.2.10 a pagina 90(b) rappresenta la stessa cosa in contesto dei dati dei tutor.

<sup>2</sup>Ricordiamo che è prerogativa di XP parlare per metafore.

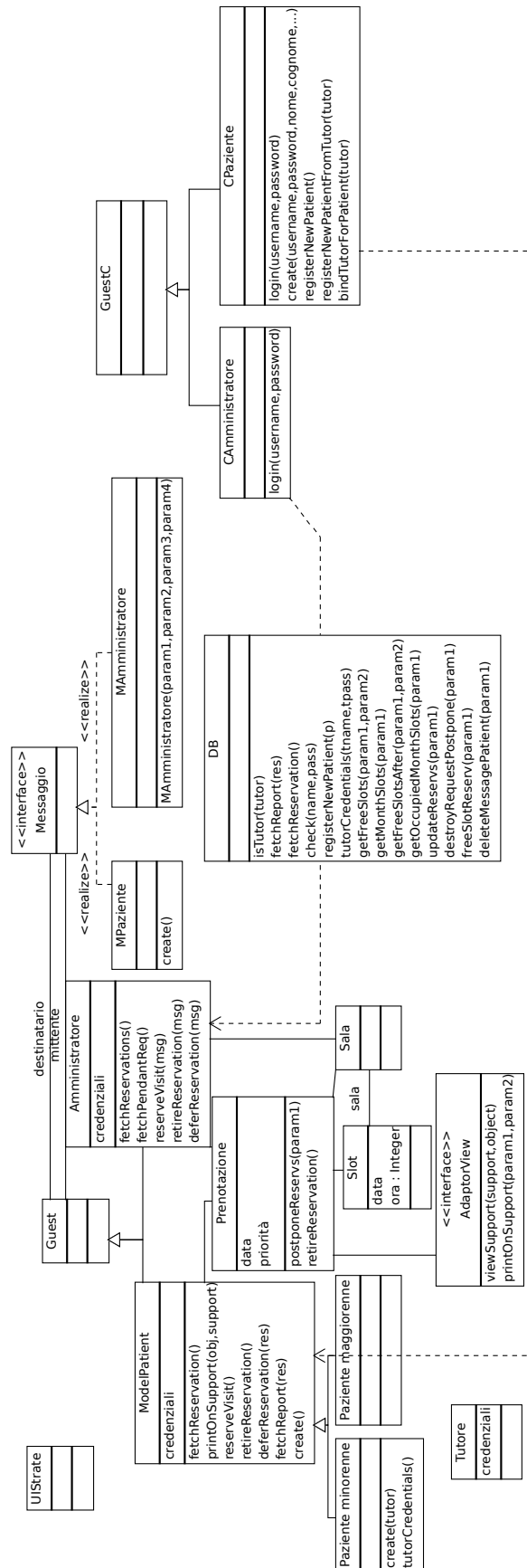


FIGURA 7.2.8. *Second Revision of the Domain Model presented in Section 4.3 a pagina 42, with only methods.*



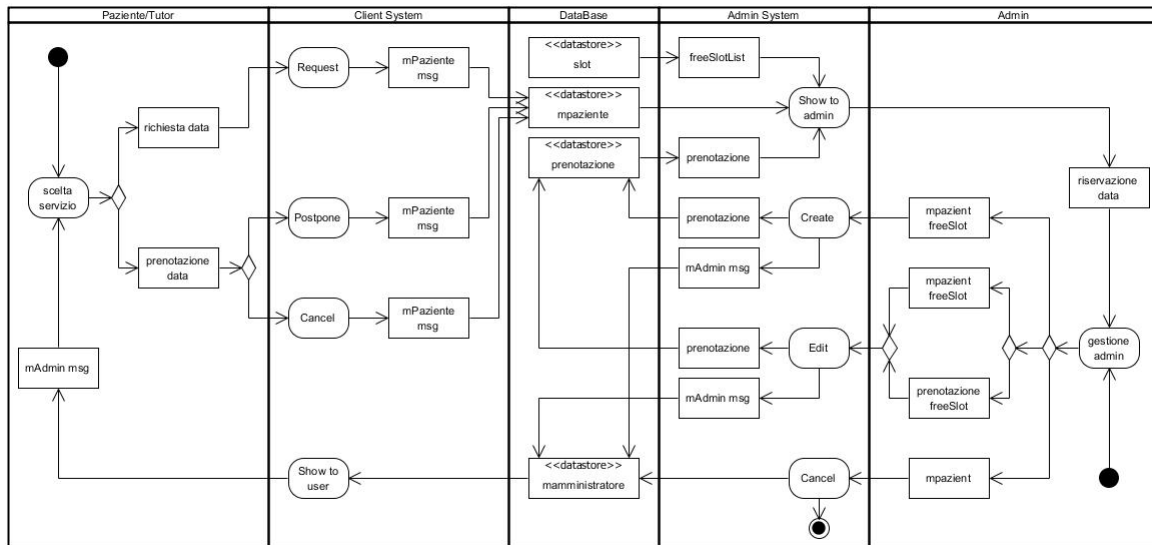
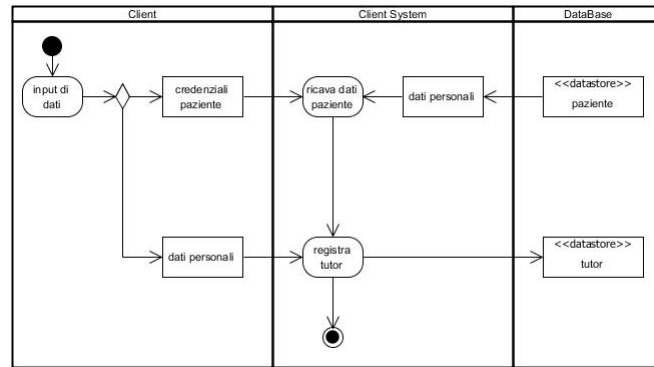
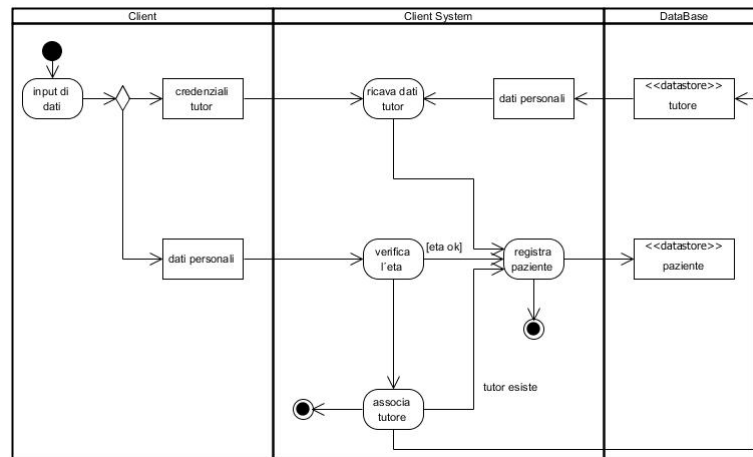


FIGURA 7.2.9. Messages' Data Flow.



(A) Tutor's Registration.



(B) Patient's Registration.

FIGURA 7.2.10. User's Data Flow.

## CAPITOLO 8

### Progettazione della base di dati

#### Indice

8.1. Analisi dei requisiti	91
8.2. Progettazione concettuale	91
8.3. Progettazione Logica	92
8.3.1. Eliminazione delle generalizzazioni	92
8.4. Traduzione Logica	92
8.5. Installazione del database	98
8.6. Utilizzo del Framework EJP	98

ALLEGATO: viene fornito il modello del database `database/DATABASE.sql`  
In questo documento procederemo ad effettuare le seguenti analisi:

- ◇ *Progettazione del Database tramite modello ER*: (v. Sezione 8.1) identificheremo la struttura logica del nostro database, per poi effettuare la traduzione in linguaggio SQL3
- ◇ *Analisi del Framework*: (v. Sezione 8.6 a pagina 98) ci preoccupiamo di come di effettuare il mapping dei dati memorizzati all'interno di un database relazionale, in un linguaggio Object Oriented.

#### 8.1. Analisi dei requisiti

Questa coincide con l'analisi del Piano di Sviluppo presentata nel Capitolo 7 a pagina 75. Possiamo sottolineare che la realizzazione di una base di dati all'interno del nostro progetto sia necessaria, in quanto è necessario mantenere dei dati persistenti all'interno della nostra applicazione.

#### 8.2. Progettazione concettuale

Dal modello di dominio, possiamo ottenere un modello equivalente per la nostra base di dati, che può essere tradotto in un diagramma ENTITY-RELATIONSHIP<sup>1</sup> (d'ora in poi ER). Contestualmente ampliamo il Glossario della Sezione 2.4 a

<sup>1</sup>NOTA: all'interno dei diagrammi ER, per ogni entità si descrive «il numero minimo e massimo di occorrenze di relazione, a cui una occorrenza dell'entità può partecipare» (Atzeni, Ceri, Paraboschi, Torlone: "Basi di dati: Modelli e linguaggi di interrogazione". McGraw-Hill Edizioni.)

pagina 33 con il Glossario dei dati in Tabella 1 nella pagina successiva, dove indichiamo per ciascuna entità gli attributi corrispondenti; non palesiamo inoltre il significato delle relazioni, in quanto di per se deducibili nella Figura 8.2.1 a pagina 94.

### 8.3. Progettazione Logica

Dall'analisi dei requisiti non funzionali del nostro sistema, possiamo ottenere la Tavola dei Volumi e delle Operazioni osservabile in Tabella 2 a pagina 95. I valori di cui sotto ci sono stati forniti dal cliente come stima per eccesso dei dati della clinica di un anno.

**8.3.1. Eliminazione delle generalizzazioni.** In quanto la soluzione di "accorpamento del genitore nelle generalizzazioni figlie" comporti il dover periodicamente verificare se gli utenti minorenni abbiano o meno raggiunto la maggiore età, andando quindi ad aumentare il numero delle operazioni non necessarie all'interno del database, si ritiene utile effettuare un "accorpamento delle entità figlie nella generalizzazione del genitore". Ciò risulta altresì vantaggioso in quanto le entità figlie possono essere distinte dalla presenza o meno della relazione con il Tutore, che è inoltre l'unica per la quale l'entità figlia Paziente Minorenne distingue l'entità Paziente Maggioreenne. Conseguentemente, all'interno della Base di Dati, potremmo ottenere un'unica entità Paziente.

Inoltre, in quanto è presente una relazione (1-1) tra amministratore e reparto, possiamo ipotizzare un'unione tra le due tabelle, andando ad identificare ciascun amministratore con un reparto di pertinenza. Si rivela però necessario aggiungere un attributo di REPARTO.

Il risultato ottenuto è il diagramma rappresentato in Figura 8.3.1 a pagina 96.

### 8.4. Traduzione Logica

Traduciamo il diagramma in relazioni.

**Amministratore:** Amministratore(NomeUtente, Password)

**Referto:** Referto(Id, Contenuto)

**Sala:** Sala(Numero, Amministratore)

**Tutore:** Tutore(CF, Nome, Cognome, Indirizzo, Numero di telefono, E-Mail, Data di nascita, Password)

**Paziente:** Identifichiamo il Paziente Maggioreenne da quello Minorenne dalla presenza o meno del tutore.

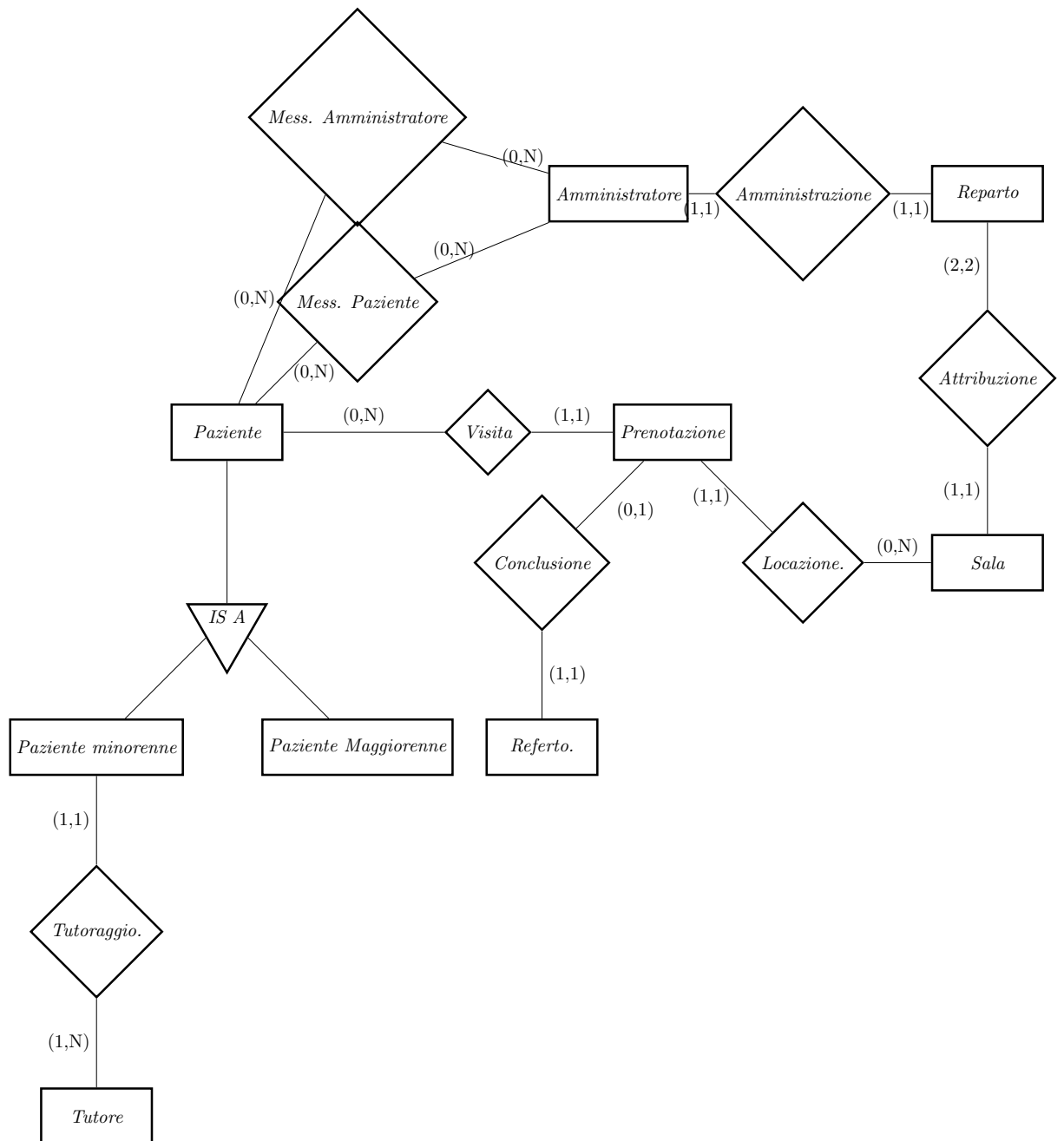
Paziente(CF, Nome, Cognome, Indirizzo, Numero di telefono, E-Mail, Data di nascita, Password, Tutore)

**Prenotazione:** Prenotazione(Id, Priorità, Medico, Ora, Data, Referto, Sala)

**Messaggi Paziente:** MPaziente(Paziente, Amministratore)

Entità	Descrizione	Attributi	Id
<b>Tutore</b>	<i>Ha la supervisione del minorenne e vi accede in sua vece.</i>	Nome, Cognome, Indirizzo, Numero di telefono, E-Mail, Data di nascita, Password	CODICE FISCALE
<b>Paziente minorenne</b>	<i>Necessita della mediazione di un tutore per accedere al sistema.</i>		
<b>Paziente maggiorenne</b>	<i>Può accedere ed interagire col sistema senza la mediazione di un tutore.</i>		
<b>Paziente</b>	<i>Astrazione di un generico cliente che può ricevere delle cure</i>	Nome, Cognome, Indirizzo, Numero di telefono, E-Mail, Data di nascita, Password	CODICE FISCALE
<b>Referto</b>	<i>Fornisce il responso dei medici in seguito alla visita del paziente</i>	ID	
<b>Prenotazione</b>	<i>Indica quando un paziente potrà accedere alle cure</i>	Priorità, Medico, Ora, Data	ID
<b>Amministratore</b>	<i>Può gestire le prenotazioni del sistema attinenti al suo reparto</i>		NOME UTENTE, PASSWORD
<b>Reparto</b>	<i>Identifica le tipologie di visite che possono essere effettuate all'interno delle sue sale</i>		NOME
<b>Sala</b>	<i>Identifica il luogo dove possono avvenire le visite di un tale reparto.</i>		NUMERO

TABELLA 1. Dizionario dei Dati.

FIGURA 8.2.1. *First ER diagram.*

**Messaggi Amministratore:** MAmministratore (Amministratore,  
Paziente)

Forniamo quindi di seguito il listato del database:

Nome	Tipo	Volume
<b>Tutore</b>	E	15
<b>Paziente minorenne</b>	E	30
<b>Paziente maggiorenne</b>	E	42
<b>Paziente</b>	E	72
<b>Referto</b>	E	135
<b>Prenotazione</b>	E	225
<b>Amministratore</b>	E	2
<b>Reparto</b>	E	2
<b>Sala</b>	E	4
<b>Tutoraggio</b>	R	45
<b>Mess. Paziente</b>	R	230
<b>Mess. Amministratore</b>	R	230
<b>Visita</b>	R	230
<b>Amministrazione</b>	R	2
<b>Attribuzione</b>	R	4
<b>Locazione</b>	R	225
<b>Conclusione</b>	R	135

TABELLA 2. *Tavola dei Volumi e delle Operazioni.*

```

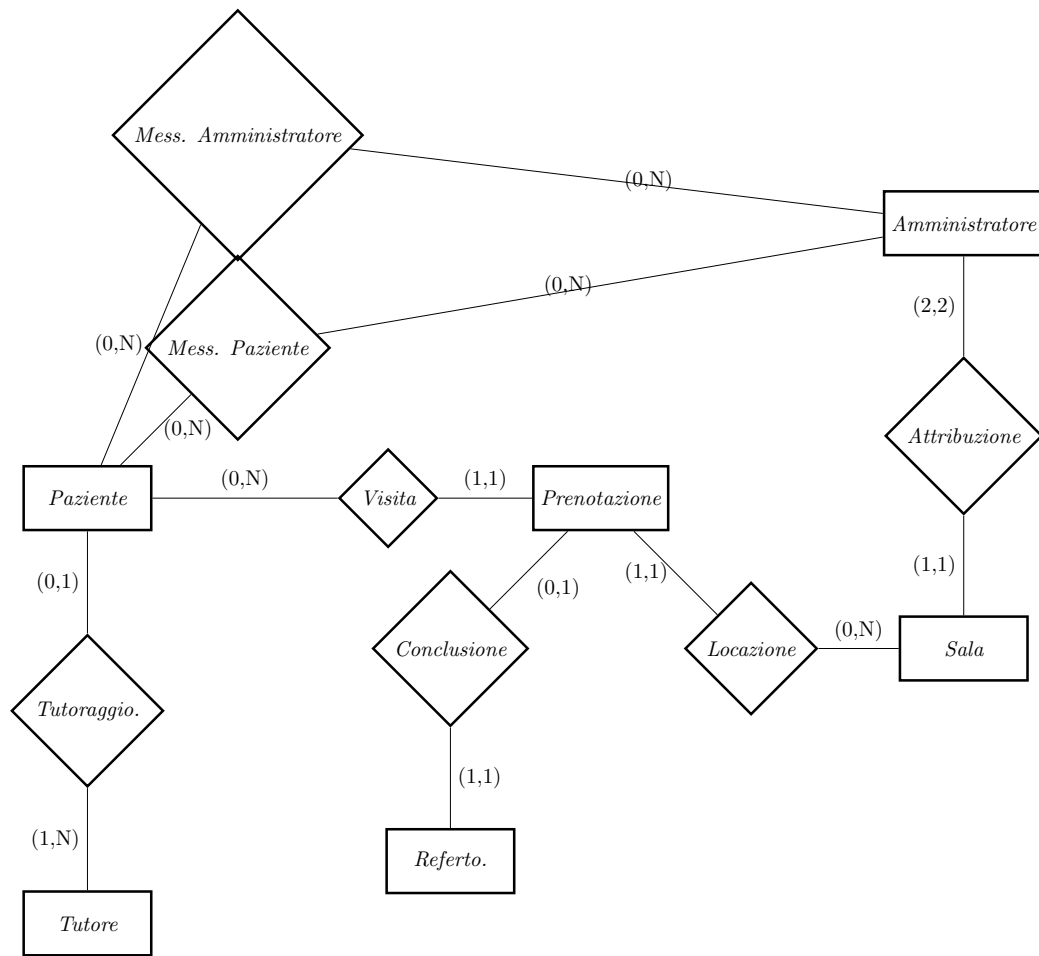
CREATE TABLE amministratore (
    nomeutente text PRIMARY KEY,
    password text NOT NULL
);

CREATE TABLE referto (
    id integer PRIMARY KEY,
    contenuto text NOT NULL
);

CREATE TABLE sala (
    numero integer PRIMARY KEY,
    amministratore text NOT NULL,
    unique (numero, amministratore),
    foreign key (amministratore) references amministratore(
        nomeutente)
);

CREATE TABLE tutore (
    nomeutente character(16) PRIMARY KEY,
    nome text NOT NULL,
    cognome text NOT NULL,

```

FIGURA 8.3.1. *Final ER diagram.*

```

indirizzo text NOT NULL,
telefono text NOT NULL,
email text NOT NULL,
nascita date NOT NULL,
password text NOT NULL,
unique (nome,cognome,indirizzo,telefono,email,nascita,password
)
);

```

```

CREATE TABLE paziente (
  nomeutente character(16) PRIMARY KEY,
  nome text NOT NULL,
  cognome text NOT NULL,
  indirizzo text NOT NULL,

```



```

telefono text NOT NULL,
email text NOT NULL,
nascita date NOT NULL,
password text NOT NULL,
tutore character(16) NOT NULL,
unique (nome,cognome,indirizzo,telefono,email,nascita,password
        ,tutore),
foreign key (tutore) references tutore(nomeutente)
);

```

```

CREATE TABLE mamministratore (
    destinatario character(16) NOT NULL,
    mittente text NOT NULL,
    type number(1) NOT NULL,
    prenot integer,
    content text,
    foreign key (prenot) references prenotazione(id),
    foreign key (mittente) references amministratore(
        nomeutente),
    foreign key (destinatario) references paziente(nomeutente)
);

```

```

CREATE TABLE mpaziente (
    mittente character(16) NOT NULL,
    destinatario text NOT NULL,
    type number(1) NOT NULL,
    prenot integer,
    content text,
    foreign key (prenot) references prenotazione(id),
    foreign key (destinatario) references amministratore(
        nomeutente),
    foreign key (mittente) references paziente(nomeutente)
);

```

```

CREATE TABLE prenotazione (
    id integer PRIMARY KEY,
    priorit  integer NOT NULL,
    medico text NOT NULL,
    ora integer NOT NULL,
    data date NOT NULL,
    referto integer NOT NULL,
    sala integer NOT NULL,
    paziente text NOT NULL,

```

```
foreign key (paziente) references paziente(nomeutente),  
foreign key (referto) references referto(id),  
foreign key (sala) references sala(numero),  
unique (priorita, medico, ora, data, referto, sala, paziente)  
);
```

---

### 8.5. Installazione del database

Si consiglia di installare il software POSTGRESQL per la gestione del database: per creare un nuovo database per un utente che vi ha i privilegi di accesso, digitare:

```
createdb orthopediatrics
```

dove `orthopediatrics` è il nome del database. Per importare il database dal file `database/DATABASE.sql` digitare:

```
cat database/DATABASE.sql | psql -d orthopediatrics
```

Per la definizione dell'utente principale di accesso al database, si veda il sito

<http://www.sakana.fr/blog/2007/06/06/postgresql-create-a-user-a-database-and-grant-accesses/>

### 8.6. Utilizzo del Framework EJP

Abbiamo scelto di utilizzare il *framework* "Easy Java Persistence"<sup>2</sup> in quanto semplificava notevolmente il processo di interfacciamento con il database, in quanto non sono necessarie né configurazioni tramite file XML, né si ha la necessità di estendere classi o di implementare interfacce fornite dal nostro framework.

Altro vantaggio di questo framework è dovuto al fatto che modifiche al database, non implicano dirette modifiche agli oggetti del nostro applicativo e viceversa: predispone inoltre di metodi di *rollback* e la definizione di *transactions* per non rendere inconsistente lo stato del nostro database.

Parallelamente, è necessario effettuare alcuni accorgimenti all'interno della definizione delle nostre classi:

- La definizione di una classe `Classname` è elemento sufficiente per accedere alla tabella `classname` del nostro database.
- È opportuno definire degli attributi che corrispondano agli attributi delle tabelle, contemporaneamente a dei metodi *getter* e *setter*.
- È opportuno definire per ogni classe un metodo `toString()`.
- Ogni relazione con una tabella `Foreign` è mappata all'interno della classe come una `List<Foreign>`.

---

<sup>2</sup><http://www.easierjava.com>

In questo modo si rivela non necessaria la definizione di un servizio di comunicazione *client-server*, in quanto già questo framework supporta meccanismi di sincronizzazione e di accesso remoto ai database.



## **Parte 5**

# **Documento Quinto**



## CAPITOLO 9

### Valutazione finale

#### Indice

---

<b>9.1. Differenze di implementazione</b>	<b>103</b>
9.1.1. Database	103
9.1.2. Amministratore	105
9.1.3. Modifica del modello di business	105
<b>9.2. Valutazione del prodotto</b>	<b>108</b>
9.2.1. Diario	108
9.2.2. Testing	108
9.2.3. Revisione dello sforzo del progetto	113
9.2.4. Analisi di qualità	114
9.2.5. Valutazione del ruolo produttivo	114

---

In quest'ultima parte, ci proponiamo di effettuare un'ultima revisione dei diagrammi precedentemente forniti, in quanto in fase di implementazione si sono effettuate scelte stilistiche differenti da quelle precedentemente fornite (Sezione 9.1).

Ci proponiamo in fine di generare un'ultima revisione delle stime precedentemente effettuate, allo scopo di dimostrare la qualità del software da noi prodotto (Sezione 9.2 a pagina 108).

In questa sezione inseriamo la parte di Diario (v. 9.2.1 a pagina 108) riguardante anche il Documento precedente (v. Parte 4 a pagina 74.)

#### 9.1. Differenze di implementazione

**9.1.1. Database.** Rispetto al modello iniziale del database, sono state effettuate alcune piccole modifiche riguardanti la sua struttura interna. Queste modifiche sono state necessarie in quanto il framework utilizzato **EJP** (*Easy Java Persistence*) "obbliga" ad esempio, la strutturazione della progettazione delle classi del database. Tuttavia le piccole modifiche applicate sono compensate da una potenza di espressione e semplicità di utilizzo dello stesso per raggiungere gli obbiettivi prefissati.

In seguito ad alcuni problemi riscontrati nello stesso framework, sono state rimosse tutte le referenze dal database e, per semplificare ulteriormente il mapping delle classi all'interno del database, sono stati modificati alcuni campi nelle singole relazioni.

```
CREATE TABLE amministratore (  
  nomeutente text PRIMARY KEY,  
  password text NOT NULL,  
  reparto text NOT NULL,  
  unique (nomeutente, password))
```

```
CREATE TABLE paziente (  
  nomeutente character(16) PRIMARY KEY,  
  password text NOT NULL,  
  nome text NOT NULL,  
  cognome text NOT NULL,  
  indirizzo text NOT NULL,  
  telefono text NOT NULL,  
  email text NOT NULL,  
  nascita date NOT NULL,  
  tutore character(16),  
  unique (nomeutente, password, nome, cognome, indirizzo, telefono,  
  email, nascita, tutore))
```

```
CREATE TABLE tutore (  
  nomeutente character(16) PRIMARY KEY,  
  password text NOT NULL,  
  nome text NOT NULL,  
  cognome text NOT NULL,  
  indirizzo text NOT NULL,  
  telefono text NOT NULL,  
  email text NOT NULL,  
  nascita date NOT NULL,  
  unique (nomeutente, password, nome, cognome, indirizzo, telefono,  
  email, nascita))
```

```
CREATE TABLE prenotazione (  
  id integer PRIMARY KEY,  
  paziente character(16) NOT NULL,  
  priorita integer NOT NULL,  
  ora integer NOT NULL,  
  data date NOT NULL,  
  sala integer NOT NULL,  
  reparto integer NOT NULL,  
  unique (id, paziente, priorita, ora, data, sala, reparto))
```

```
CREATE TABLE messaggio (  
  --
```



```

data date NOT NULL,
orario text NOT NULL,
mittente text NOT NULL,
destinatario text NOT NULL,
content text NOT NULL,
tipo integer NOT NULL,
prenotazione integer,
unique (data, orario, mittente, destinatario, content, tipo,
prenotazione))

CREATE TABLE referto (
id integer PRIMARY KEY,
prenotazione integer NOT NULL,
contenuto text NOT NULL,
medico text NOT NULL,
paziente text NOT NULL,
unique (id, prenotazione, contenuto, medico, paziente))

```

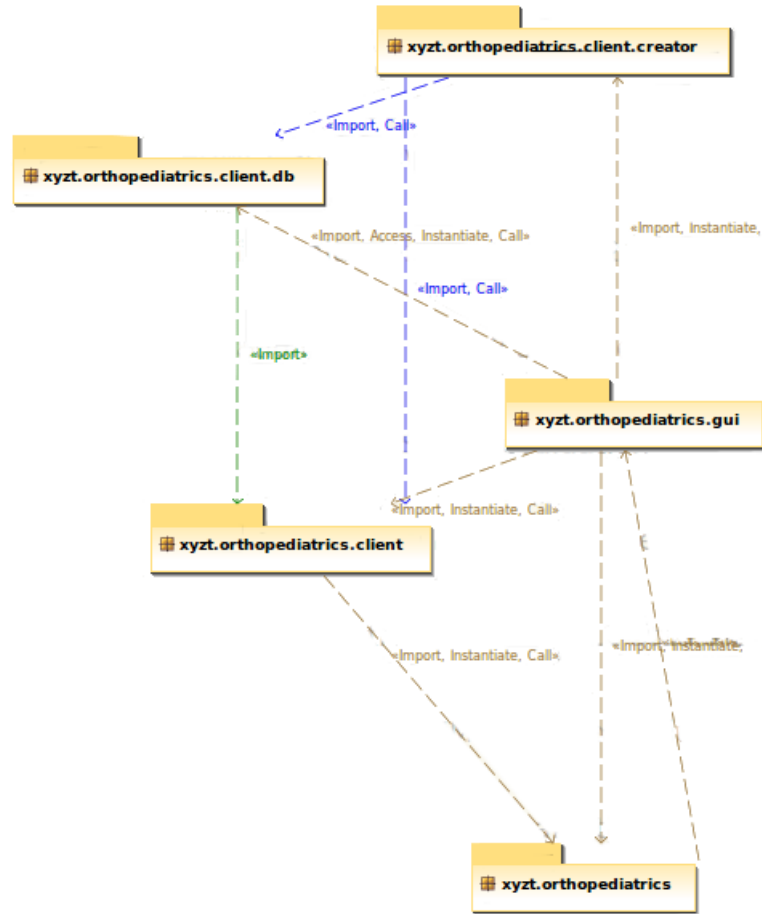
---

**9.1.2. Amministratore.** Come stabilito con il cliente, esistono due amministratori predefiniti all'interno del sistema; siccome questi devono esistere sempre abbiamo deciso di creare due amministratori che vengono automaticamente inseriti all'interno del database durante la creazione dello stesso. Gli amministratori si distinguono come:

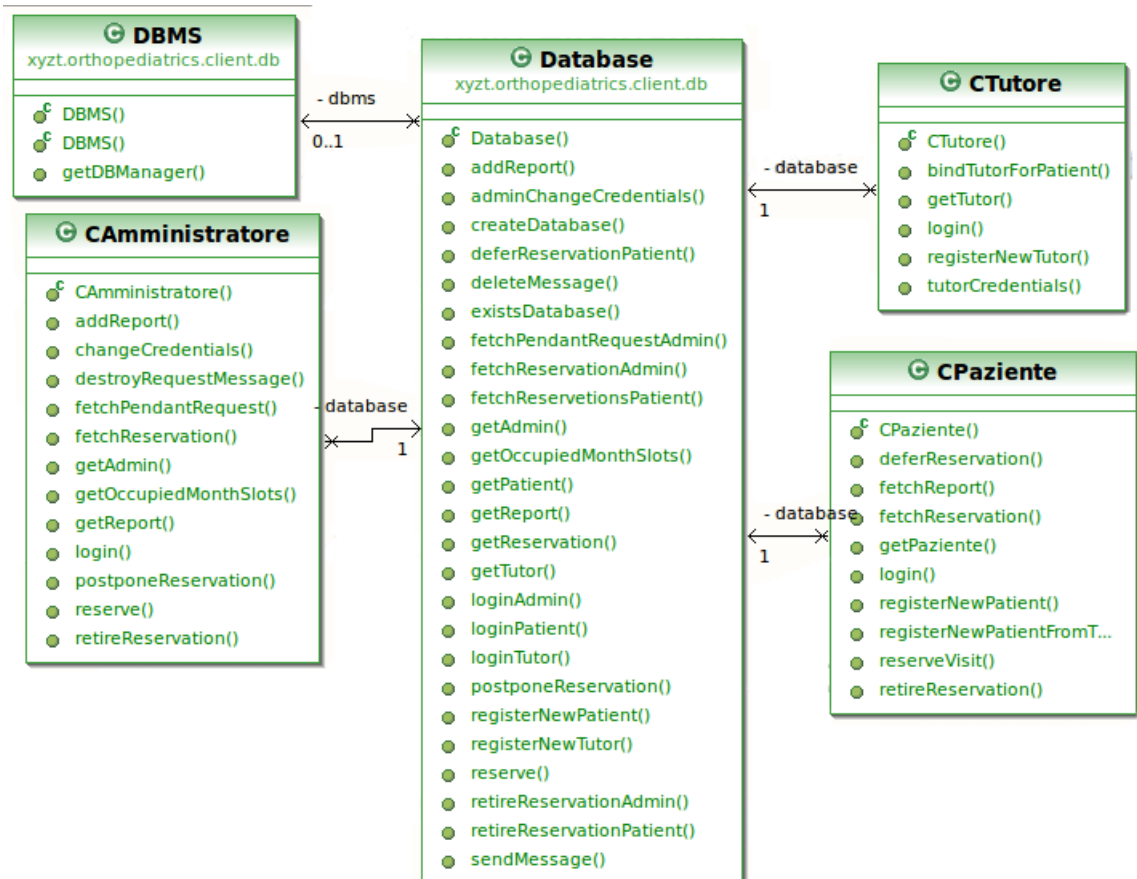
- Reparto di Ortopedia
  - username: *admino*
  - password: *admino*
- Reparto di Pediatria
  - username: *adminp*
  - password: *adminp*

Questi possono essere cambiati semplicemente effettuando un query al database (dall'amministratore del database), oppure l'amministratore può cambiare i propri dati di accesso da interfaccia grafica.

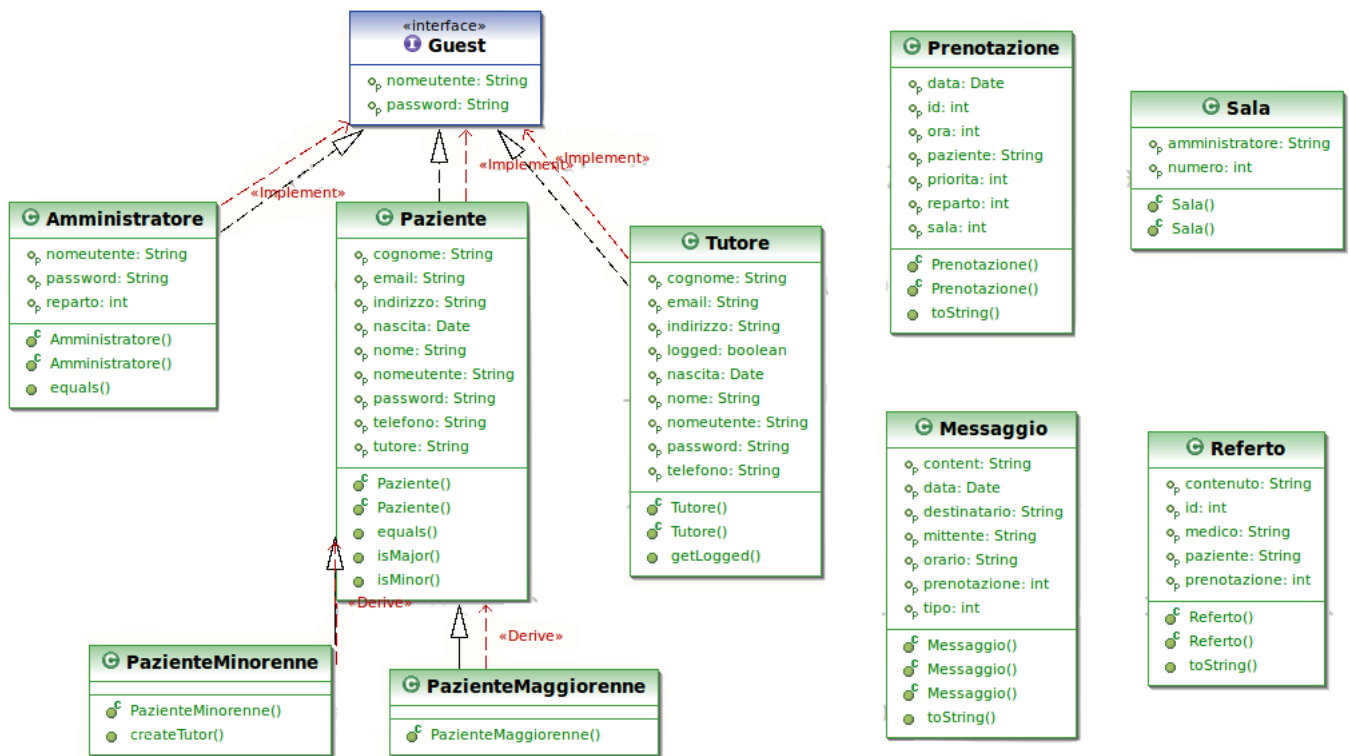
**9.1.3. Modifica del modello di business.** Rispetto al modello iniziale di progettazione delle classi, sono state effettuate alcune modifiche semplificando ulteriormente la struttura interna delle classi; queste modifiche (come detto in precedenza) sono una conseguenza dovuta in parte all'impiego del framework *EJP*, e in parte ad una scelta di progettazione, in quanto vengono divise le singole responsabilità delle classi. In dettaglio è voluto spostare le funzionalità contenute all'interno delle classi del modello di dominio (in particolare Paziente, Amministratore e Tutore) nelle classi "Creator" che caratterizzano il design pattern "Abstract Factory" (v. Figura 9.1.1 nella pagina successiva(b)).



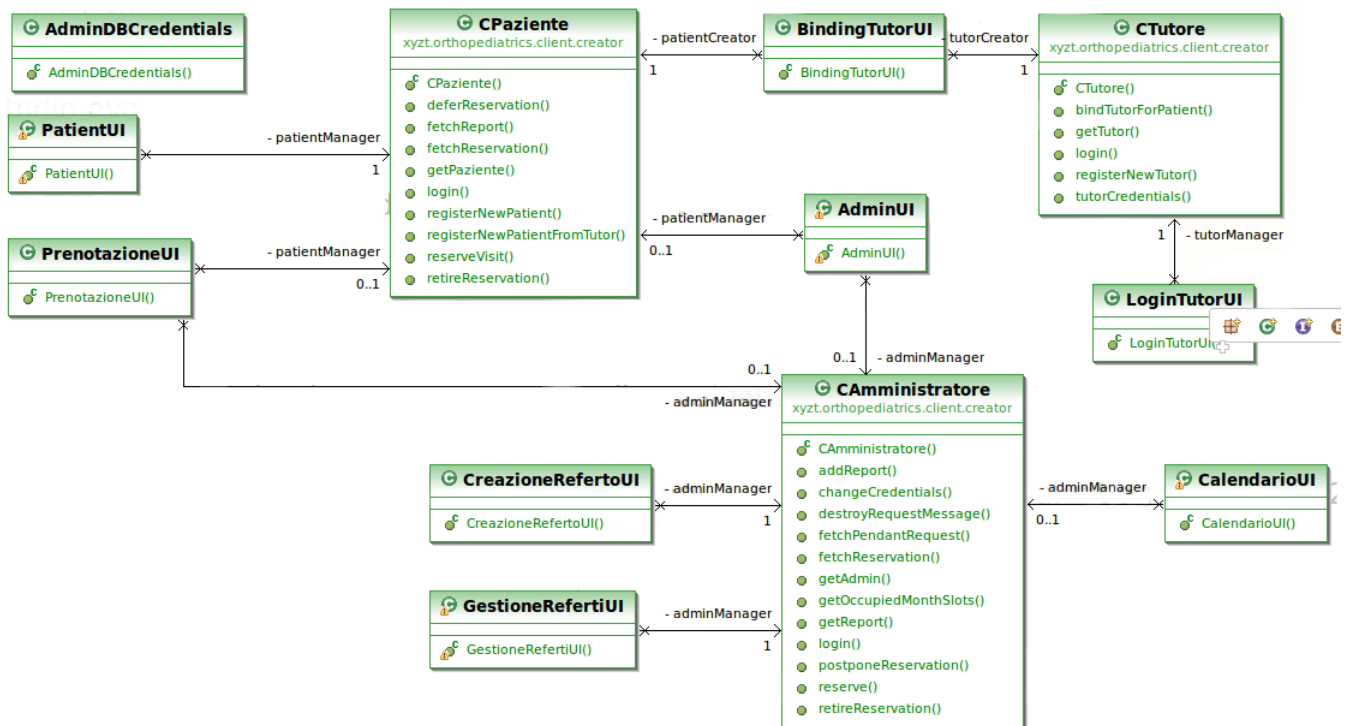
(A) Package diagram for the whole project.



(B) New organization for Database Data Management.



(A) Classes used for direct Database mapping.



(B) Database Mapping and GUI state.

FIGURA 9.1.2. Implementation's View (1).

Inizialmente la gestione dei messaggi era di responsabilità di due classi distinte: *MAmmministratore* e *MPaziente*, contenenti rispettivamente i messaggi diretti all'amministratore e al paziente. Si è voluto unificare tale modello in un'unica classe **Messaggio** contenente i messaggi diretti ad entrambi gli utenti del sistema (Amministratore e Paziente), distinguendoli come *mittente* e *destinatario* (v. Figura 9.1.2 nella pagina precedente(a)). Possiamo inoltre evidenziare tramite la Figura 9.1.2 nella pagina precedente(b) come sia stata realizzata l'interazione tra database ed interfaccia grafica, senza la necessità di ricorrere all'implementazione di uno strato intermedio.

[!thp] Possiamo inoltre evidenziare tramite i diagrammi rappresentati in Figura 9.1.3 a fronte e 9.1.4 a pagina 110, come questi possano differire rispetto a quelli presentati nella Sottosezione 7.2.2 a pagina 78.

## 9.2. Valutazione del prodotto

### 9.2.1. Diario.

**Giovedì 26 Aprile:** Il gruppo si è ritrovato alle ore 13.00 con il cliente per discutere aggiornamenti, fornire chiarimenti sulla fase precedente ed fornire un'informazione e parziale documentazione della fase attuale.  
Ore di lavoro: 5

**Vacanze:** Durante il periodo di vacanze, i vari membri del gruppo hanno svolto le attività di loro competenza. Ore di lavoro: 29.

Possiamo ora rappresentare tutte le fasi dello sviluppo tramite il Diagramma di Gantt proposto in Figura 9.2.1 a pagina 110.

**9.2.2. Testing.** Noi abbiamo effettuato due tipologie di testing: quello **white box** in allegato assieme al codice sorgente, e quello **black box**, che viene fornito qui sotto.

**Utente:** Identifichiamo i test lato utente

◇ *Registrazione*

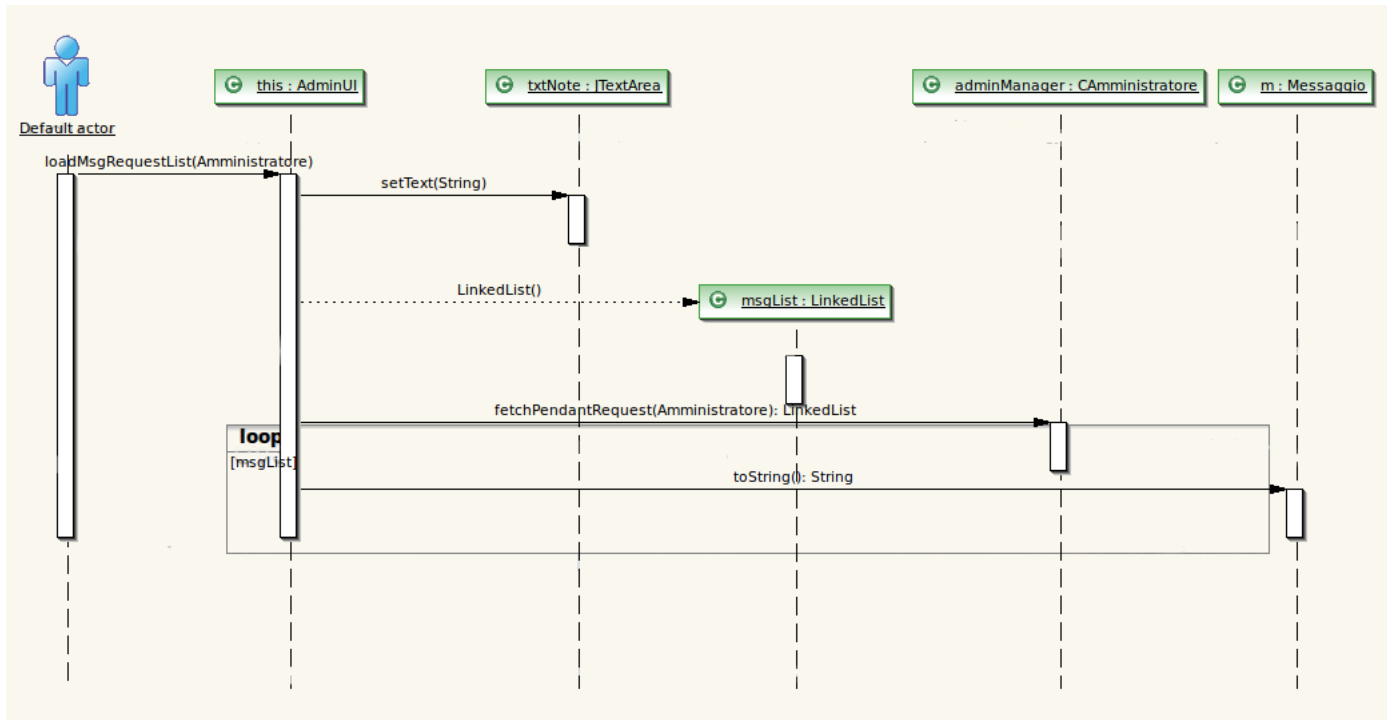
Forniamo di seguito una descrizione dei passi da effettuare.

**inserimento dei dati personali:** Si ha input non valido qualora

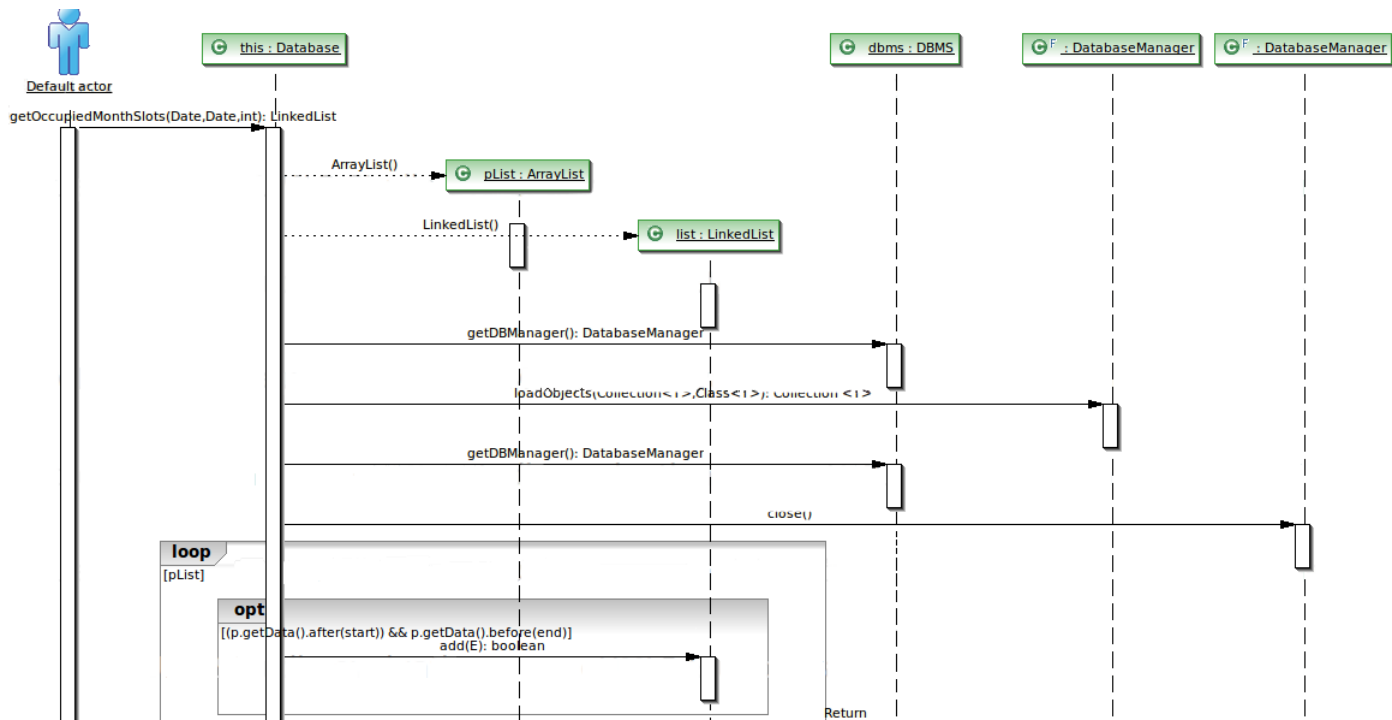
- manca uno o più di un dato richiesto
- il codice fiscale contiene meno di 16 caratteri
- l'email è *well-formed*
- la data non può contenere caratteri

**Invio di informazioni:**

Si ha **successo** quando si ottiene un messaggio di conferma di effettuato inserimento dell'utente, e quindi si considera passato il test. Si ha invece **fallimento** qualora l'utente non sia stato inserito.

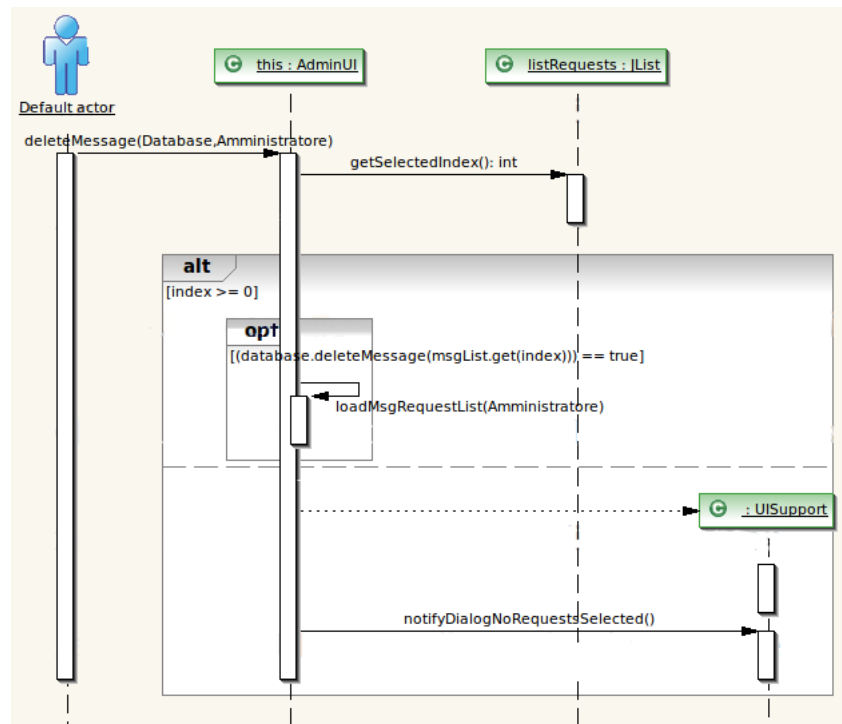
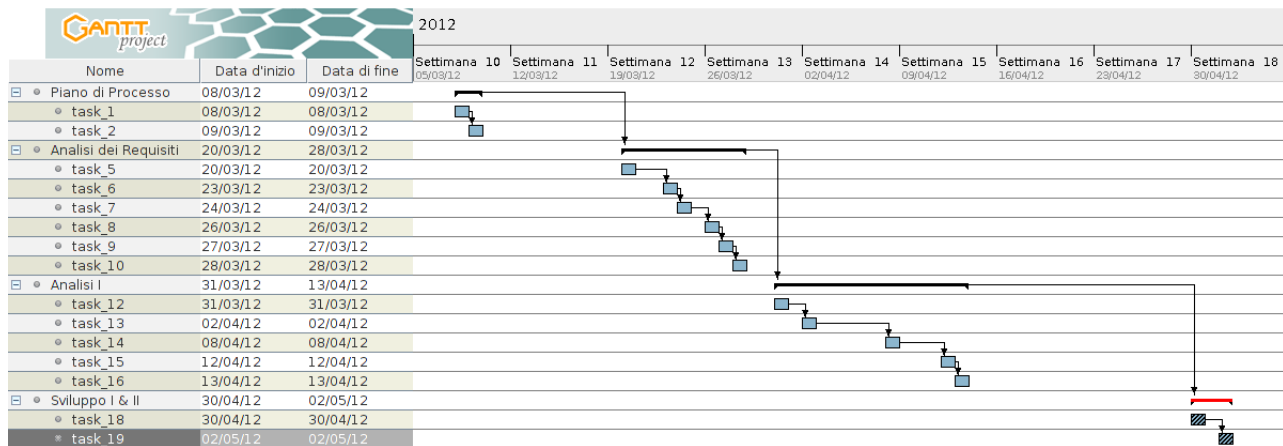


(A) Loading admin's Message Request list.



(B) Getting current Ward's prenotations.

FIGURA 9.1.3. Some interaction diagrams for the final implementation.

FIGURA 9.1.4. *Deleting received Request Messages.*FIGURA 9.2.1. *Developing's Gantt Diagram.*

◇ *Accesso*

È necessario effettuare l'inserimento delle credenziali per l'autenticazione, e conseguentemente effettuare il login.

Si ha **successo** qualora l'utente riesca ad entrare nel sistema,

altrimenti si ha **fallimento** se l'utente non riesce ad autenticarsi, a causa di un'errata o mancata registrazione.

◇ *Richiedere prenotazione*

È opportuno inserire i dati, quali la scelta reparto e l'inserimento della descrizione dei "sintomi": in seguito bisogna inoltrare la richiesta.

L'esecuzione è avvenuta con **successo** se:

- si visualizza un messaggio di conferma
- il test di "visualizzare delle prenotazioni" è esso stesso riuscito.

altrimenti porta a **fallimento** se:

- l'utente non riesce ad inviare richiesta
- la visualizzazione delle prenotazioni fallisce

◇ *Visualizzare prenotazioni*

Le prenotazioni vengono aggiornate tramite il comando di refresh. Se non venissero visualizzate, allora si avrebbe un caso di fallimento. Altri possibili motivi di errore potrebbero essere i seguenti:

- si visualizzano le prenotazioni che appartengono ad un altro paziente
- si visualizzano delle prenotazioni che non sono state richieste

◇ *Posticipo prenotazione*

Si effettua una scelta della prenotazione ed una richiesta di posticipazione

In caso di **successo** si visualizza un messaggio di conferma, e si deve controllare successivamente alla gestione che la prenotazione sia stata effettivamente posticipata. Si ha **fallimento** invece se:

- l'utente non riesce a fare la richiesta
- non si riesce a scegliere la prenotazione
- la prenotazione non viene posticipata

◇ *Cancellazione di prenotazione*

Si effettua una scelta della prenotazione ed una richiesta di cancellazione

In caso di **successo** si visualizza un messaggio di conferma, e si deve controllare successivamente alla gestione che la prenotazione sia stata effettivamente cancellata. Si ha **fallimento** invece se:

- non si riesce a fare la richiesta
- non si riesce a scegliere la prenotazione
- la prenotazione non viene cancellata

◇ *Visualizzare referto*

Si sceglie la prenotazione di cui si vuole visualizzare referto, che deve essere visualizzata in caso di **successo**. In caso contrario, si ha il fallimento: altro caso di errata gestione della visualizzazione

è la visualizzazione del referto errato.

◇ *Logout*

Avviene semplicemente uscendo dal programma: non dovrebbero essere previsti particolari casi di errore.

**Admin:** Identifichiamo i test lato amministratore.

◇ *Accesso*

Come sopra

◇ *Visualizzare le richieste(prenotazione, posticipazione, cancellazione)*

Si ha **successo** se le richieste vengono visualizzate, altrimenti si ha **fallimento**

◇ *Creare nuova prenotazione*

Forniamo di seguito una descrizione dei passi da effettuare.

- : scegliere richiesta da gestire;
- : creare prenotazione: in particolare è necessario scegliere la data, associare la stanza, associare il medico associare la priorità e specificare il paziente;
- : salva prenotazione

Si ha **successo** solo se la prenotazione è stata creata o salvata all'interno del database; in caso di **fallimento** invece, si potrebbero verificare i seguenti casi:

- la prenotazione non viene salvata
- non si riesce a scegliere la richiesta da gestire
- non si riescono ad editare data/stanza/medico/priorità.

◇ *Modificare prenotazione visualizzata*

Forniamo qui la descrizione dei passi da effettuare.

- visualizzare le prenotazioni
- scegliere la prenotazione
- modificare della prenotazione
- salvare prenotazione

Si rientra nel caso di **successo** se la prenotazione è stata modificata e salvata, altrimenti si potrebbero verificare i seguenti casi di fallimento:

- le prenotazioni non vengono visualizzate
- non si riesce a scegliere la prenotazione
- l'edit fallisce
- le modifiche non vengono salvate

◇ *Posticipare prenotazione visualizzate*

È necessario scegliere la prenotazione da posticipare, per poi inviare il comando di posticipa.

L'operazione è stata effettuata con **successo** se la prenotazione viene effettivamente posticipata, altrimenti potrebbero avvenire i seguenti casi di errore:



- non si riesce a scegliere la prenotazione da posticipare
- la prenotazione non viene posticipata
- si riesce ad anticipare la prenotazione
- ◇ *Cancellare prenotazione visualizzata*  
 È opportuno scegliere la prenotazione da cancellare, per poi inoltrare il comando di cancellazione.  
 Si ha **successo** se la prenotazione è stata opportunamente cancellata, altrimenti (in caso di **fallimento**) non si riesce a scegliere prenotazione da cancellare o/e la prenotazione non viene cancellata.
- ◇ *Creare/modificare ed associare referti*  
 È opportuno eseguire i seguenti step:
  - scegliere la prenotazione da modificare
  - creare o scegliere il referto da modificare
  - associare referto
 L'esecuzione è avvenuta con **successo** se il referto viene creato/-modificato, altrimenti porta a **fallimento** se si verifica uno o più dei casi seguenti:
  - failure - non si riesce a creare referto
  - non si riesce ad associare referto
  - si riesce ad associare più di un referto alla stessa prenotazione
- ◇ *Visualizzare referti*  
 È necessario scegliere la prenotazione di cui si vuole visualizzare referto.  
 Si ha **successo** se il referto viene visualizzato, altrimenti in caso di **fallimento** non viene visualizzato, o comunque non viene visualizzato il referto previsto.

**9.2.3. Revisione dello sforzo del progetto.** Per effettuare una stima finale dello sforzo compiuto, ci basiamo sulle effettive righe di codice, calcolate tramite il tool SLOCCOUNT di *David A. Wheeler*. Il nostro codice è costituito da 4642 linee di codice, con una stima molto prossima alle 4300 precedentemente stimate (v. Sezione 1.4 a pagina 11). Si ha inoltre un costo di produzione stimato di \$135.420, considerando un salario annuale medio di \$56.286 per sviluppatore. Conseguentemente, utilizzando sempre la formula di *Bailey-Basili* per la valutazione dello sforzo, otterremo il seguente sforzo:

$$E_{bb} = 5.5 + 0.7 \cdot KLOC^{1.16} \simeq 9.65 \text{ mesi/persona}$$

Effettuando la somma delle ore ottenute nel diario, otteniamo un totale di 111 ore complessive di lavoro per ogni componente del gruppo. Stimando ora che ogni componente del gruppo abbia lavorato per 5 ore al giorno, dati che questo team sta lavorando correntemente ad altri progetti, otteniamo un lavoro di 22 giorni/persona continuativi. Possiamo comunque evidenziare come la

stima ottenuta tramite analisi dei FP, sia stata comunque una sovrastima del tempo che è stato effettivamente necessario allo sviluppo.

**9.2.4. Analisi di qualità.** Tramite il tool METRICS di ECLIPSE, possiamo desumere le seguenti considerazioni della qualità del software:

**Complessità Ciclomatica:** In Figura 9.2.2 nella pagina successiva(a), possiamo notare che sono pochi i metodi a possedere complessità ciclomatica alta: questi sono prevalentemente per la gestione della GUI, che comunque rimane al di sotto del limite 50 di testabilità (Il massimo è raggiunto con `ClientUI` che ha Complessità Ciclomatica 26).

**Numero di parametri:** In figura 9.2.2 a fronte(a), Possiamo notare anche in questo caso che sono pochi i metodi nei quali si ha un elevato numero di parametri (il massimo raggiunto è 9): questo è dovuto al fatto che spesso la creazione di istanze di oggetti per record del database, necessita esso stesso un numero di parametri pari agli attributi delle tabelle nel suddetto database.

**Mancanza di Coesione nei Metodi:** Abbiamo considerato queste stime, anche se sono considerate poco attendibili dallo stesso tool di sviluppo<sup>1</sup>. Utilizzando la stima di *Chidamber-Kemerer*, possiamo evidenziare che l'unica classe che risulti problematica sia quella PAZIENTE: tuttavia ci teniamo a sottolineare che gran parte del codice con Mancanza di Coesione è in gran parte generato automaticamente dal plug-in di ECLIPSE per la creazione di GUI, e che quindi è difficilmente ri-fattorizzabile, a costo di introdurre ulteriori errori (v. Figura 9.2.3 a pagina 116).

**9.2.5. Valutazione del ruolo produttivo.** In questo punto forniamo una descrizione della valutazione personale del ruolo produttivo da noi ricoperto.

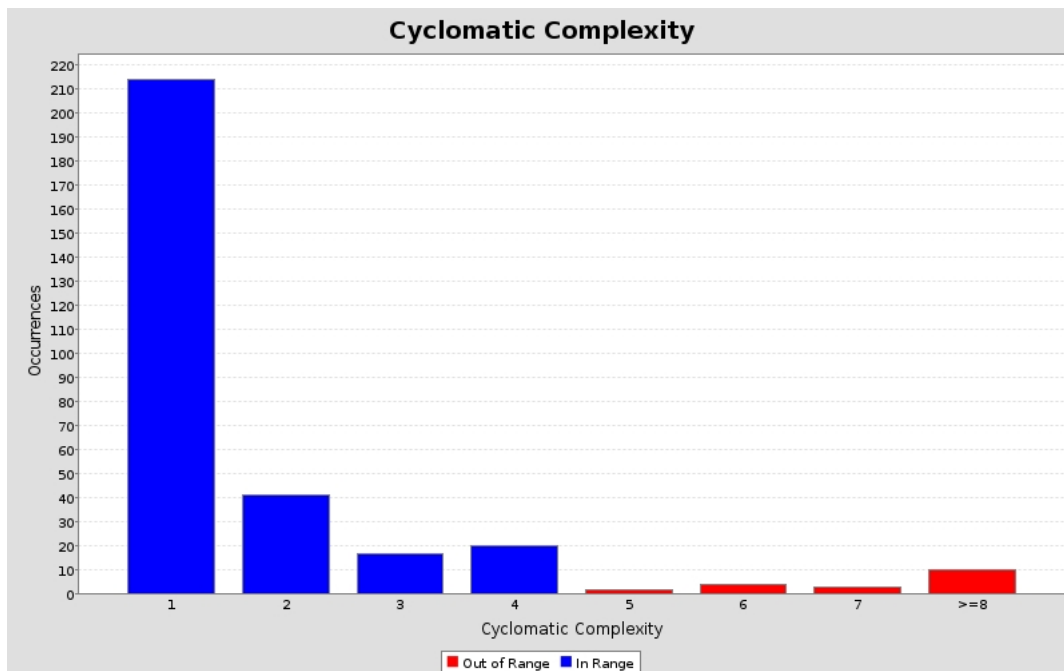
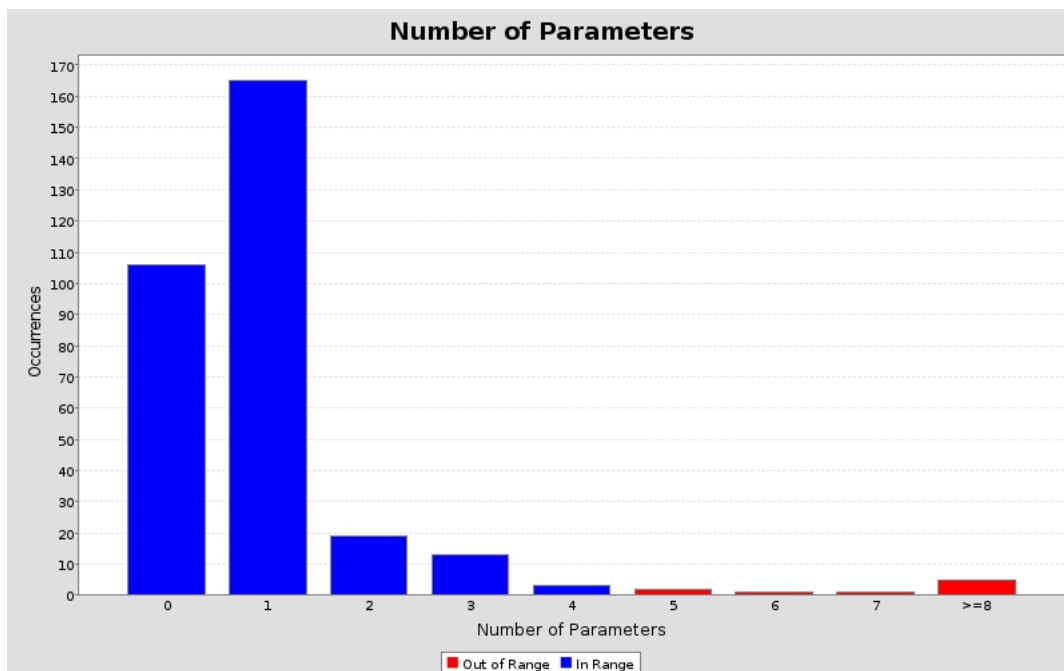
**Bergami Giacomo:** In quanto *Quality Manager* del team, posso sottolineare come sia stato periglioso la gestione della documentaizone, in quanto dovevo garantire l'omogeneità del documento finale, e indicare i punti della documentazione che non erano stati svolti in modo corretto.

Per quanto concerne invece la generazione dei test, il mio compito è stato delegato a Fabian Priftaj per il *white box*, in quanto conosceva maggiormente il codice, ed a Matej Torok per il *black box*, per l'esperienza avuta sul campo con progetti precedenti.

<sup>1</sup>«Despite its importance, it is difficult to establish a clear mechanism for measuring it. This is probably due to the fact that good abstractions have deep semantics and a class that is clearly cohesive when viewed from a semantic point of view may not be so when viewed from a purely symbolic point of view.

As an aside, the somewhat inelegant name is due to the wish to have lower metric values representing a 'better' situation.»

<http://eclipse-metrics.sourceforge.net/descriptions/pages/cohesion/LackOfCohesionInMethods.html>

(A) *Cyclomatic Complexity in Methods.*(B) *Number of Parameters in Methods.*FIGURA 9.2.2. *Quality's Metrics (1).*

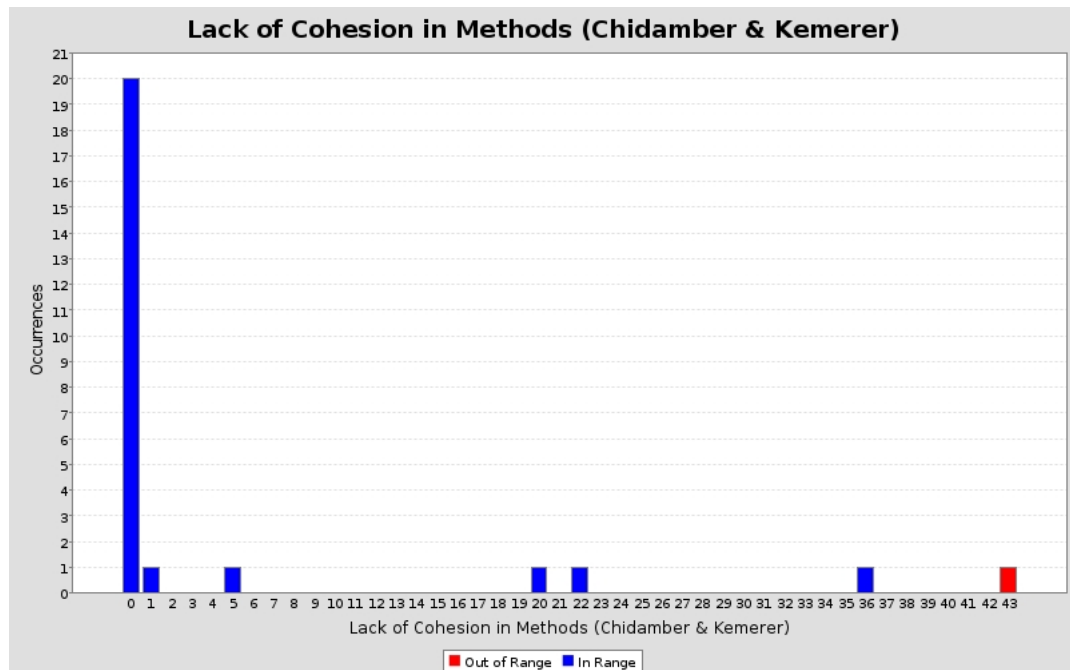


FIGURA 9.2.3. *Quality's Metrics (2) - Mancanza di Coesione nei Metodi.*

**De Luca Paolo:** Le responsabilità dovute alla carica di *Tool specialist*, si sono riversate anche sul resto del team, in quanto ci è stato necessario usare molti di strumenti nuovi, senza alcuna esperienza precedente. In questo quadro, la scelta degli strumenti si è spesso riversata verso strumenti quanto più semplici da utilizzare e facilmente accessibili.

Tuttavia, l'utilizzo di ARGOUML è stato spesso stressante a causa dell'instabilità e la ridotta usabilità dello stesso. Altra nota di demerito va ad ECLIPSE, poiché la generazione di una semplice interfaccia grafica richiede molto tempo ma soprattutto un hardware molto prestante, anche per semplici oggetti: infatti è necessario attendere un delay molto elevato per ogni aggiunta/modifica all'interfaccia.

**Fabian Priftaj:** Dato il mio ruolo di *Sviluppatore*, le difficoltà maggiori da me incontrate sono state riscontrate nel cercare di unificare le mie idee di quella che doveva essere la mia visione dell'applicazione, con le idee degli altri componenti del gruppo, dato il loro compito di effettuare la parte della progettazione della documentazione e della struttura dell'applicazione e della sua progettazione in termini di UML.

Ho dovuto quindi seguire, per la maggior parte, i loro schemi di implementazione e questo è stata la novità (e difficoltà) di questo progetto, in quanto era necessario non superare il confine, ovvero cercare di rimanere strettamente aderenti alle specifiche evitando personalizzazioni

o quant'altro.

In alcuni momenti invece è stato necessario effettuare alcuni cambiamenti rispetto a quello che era il modello di dominio di partenza, in quanto la realizzazione di tali moduli comportava un maggiore sforzo di implementazione e di conseguenza anche di tempo. Per ridurre questi fattori ho effettuato quindi una realizzazione diversa di alcuni componenti del modello di dominio; di conseguenza siamo dovuti ritornare indietro per rivalutare il modello iniziale ogni qualvolta lo abbiamo ritenuto necessario.

Per la persistenza dei dati è stato utilizzato il framework java EJP e questo ha in qualche modo limitato il supporto totale al modello di dominio di partenza, comportando un'ulteriore modifica al modello di dominio iniziale.

Parallelamente allo sviluppo del codice, ho provveduto al testing dei moduli ritenuti più importanti, ovvero quei moduli che formano il nucleo dell'applicazione stessa; i moduli trascurati dal testing sono quelli inerenti all'interfaccia grafica dell'applicazione in quanto ritenuto assolutamente non necessario.

Secondo il mio parere personale i miglioramenti che si potrebbero fare a questo tipo di applicazione sono pochi; questo è dovuto alle specifiche in se, in quanto limitate, dato lo scarso numero di funzionalità che l'applicazione deve possedere.

**Torok Matej:** A mio parere, il ruolo del Project manager in questo progetto sembrava il meno rilevante, visto che molte delle sue responsabilità sono state svolte dallo staff che ha preparato il progetto (in particolare stabilendo scadenze, cosa doveva essere prodotto dalla documentazione, quale processo di sviluppo adottare, ecc.).

Tra le responsabilità rimaste, c'è da notare la valutazione dello sforzo e l'analisi dei rischi. Durante lo svolgimento dei miei compiti, non si sono incontrate difficoltà particolari, se non alcuni dubbi su come svolgere la valutazione senza avere abbastanza informazioni sulle effettive esigenze del cliente. Purtroppo, la revisione delle stime dopo aver parlato con cliente, non si è rilevata, causa il metodo di stima adottato, particolarmente significativa nell'apportare una evidente variazione delle stesse.

Come ultima osservazione, possiamo affermare che con questo progetto abbiamo potuto apprezzare l'utilizzo di tool automatici per l'ottenimento di risultati efficaci in tempi immediati: faccio riferimento soprattutto agli ultimi Diagrammi di Interazione ed alla valutazione della qualità, che prevede dei calcoli lunghi e laboriosi nei metodi delle classi.

Da questo progetto abbiamo imparato che, allo scopo di scrivere una documentazione completa e dettagliata, è necessario impiegare molto tempo: questo

può essere tuttavia svantaggioso qualora i possibili *competitor* impieghino meno energie nella sua produzione, allo scopo di produrre il prodotto finale il prima possibile al cliente.

Siamo consapevoli che il progetto consegnato sarebbe stato ulteriormente migliorabile, ad esempio tenendo conto dell'efficienza e della responsività; avremmo potuto rendere inoltre il codice maggiormente riusabile, inserendo un ulteriore *layer* software tra l'interfaccia grafica e database, applicando eventualmente altri design pattern per gestire quest'ulteriore livello di indirizione, utilizzando ad esempio l'*Hollywood Principle*.