Università Degli Studi di Salerno
Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Informatica

Project: COD
System Design Document (SDD)

PARTECIPANTI

Cognome Nome (matricola)	Ruolo	Gruppo
Lodato Luciano (0521000952)	Team Manager	
Bernardo De Stefano (0512100207)	Team Leader Trainer Interno	Team B
Andrea Cantarella (0512100033)	Team Leader	Team A
Diego Attianese (0512100150)		Team B
Nicola Gallo (0512100015)		Team A
Giuseppe Marmora (0512100108)	Trainer Interno	Team A
Luca Guadagno (0512100102)		Team B

Anno Accademico 2010/2011

Sommario

1. Introduzione....**1**
1.1 Scopo del sistema....1
1.2 Obiettivi di Design....1
1.2.1Criteri di performance....1
1.2.2Criteri di affidabilità....2
1.2.3Criteri di costi....2
1.2.4Criteri di mantenimento....2
1.2.5Criteri End User....3
2. Architettura software corrente....**5**
2.1 Sistema ex-novo....5
3. Architettura software proposta....**7**
3.1 Overview....7
3.2 Decomposizione in sottosistemi....8
3.3 Mappaggio su hardware e software....10
3.4 Gestione dei dati persistenti....13
3.5 Controllo di accesso e aspetti relativi alla sicurezza....14
3.6 Flusso di controllo globale....15
3.7 Condizioni limite....16
3.7.1Casi d'uso per StartUp....16
3.7.2StartSistema....16
3.7.3StartAccount....17
3.7.4StartCartellaOstetrica....18
3.7.5StartVisualizzaPaziente....18
3.7.6StartVisita....19
3.7.7Casi d'uso per Shutdown....20
3.7.8ShutDownSistema....20
3.8 Comportamento del sistema in condizioni eccezionali....21
4. Servizi dei sottosistemi....**23**
4.1 Prefazione....23
4.2 Proprietà....23
4.2.1Accoppiamento (coupling)....23
4.2.2Coesione....23
4.3 Sottosistemi individuati....24

1. Introduzione

1.1 Scopo del sistema

Questo sistema ha come scopo primario quello di fornire un'applicazione con la quale gestire digitalmente delle cartelle ostetriche, facilitando l'iterazione del paziente e del dottore con essa. L'applicazione che verrà sviluppata dovrà fornire una serie di funzionalità che facilitino la gestione di tutte le attività connesse alla cartella ostetrica. In particolare, l'amministratore di sistema una volta identificato con l'applicazione, potrà effettuare queste operazioni:

- Gestire l'elenco dei dottori che potranno utilizzare l'applicazione.
- Gestire le informazioni relative ai dottori.

I dottori una volta effettuato l'accesso, dovranno poter:

- Gestire l'elenco dei pazienti.
- Gestire l'elenco delle visite.
- Gestire le cartelle dei pazienti.

I pazienti una volta effettuato l'accesso, dovranno poter:

- Consultare e\o stampare la propria cartella.

1.2 Obiettivi di Design

L'individuazione degli obiettivi di design rappresenta il primo step all'interno della fase di progettazione del sistema. Attraverso questa attività sarà possibile identificare le qualità sulle quali dovrà vertere la realizzazione dell'applicazione futura. I criteri di design che verranno presi in considerazione sono stati organizzati come segue:

1.2.1 Criteri di performance

1. Tempo di risposta: siccome il sistema che verrà sviluppato dovrà essere basato su database relazionale, il tempo di risposta dipenderà dalle tuple che dovranno essere gestite.
2. Throughput: il throughput potrà variare notevolmente, in quanto sarà influenzato anche dai tempi utente e dal sistema operativo in uso.
3. Memoria: il sistema potrà impiegare una quantità di memoria variabile, a seconda delle operazioni che l'utente esige che vengano effettuate.

1.2.2 Criteri di affidabilità

1. **Affidabilità**: l'applicazione assicurerà la piena coincidenza tra comportamento specificato e comportamento osservato: in tal modo si garantirà anche una maggiore usabilità del sistema da parte degli utenti.
2. **Disponibilità**: il sistema sarà sempre disponibile per la realizzazione delle attività di cui dispone; la sua disponibilità sarà quindi massima .
3. **Tolleranza ai fault**: il sistema avrà una buona capacità di operare sotto condizioni di errore, in quanto, in seguito ad ogni possibile condizione di errore, varrà visualizzato un messaggio di notifica, dopo il quale si permetterà la ripetizione dell'operazione precedente.
4. **Sicurezza**: il sistema ha una buona capacità di resistere ad attacchi di malintenzionati, poiché vi si potrà accedere solo attraverso l'immissione di login e password.

1.2.3 Criteri di costi

Per la realizzazione di questo sistema non è stato sostenuto alcun costo di sviluppo e di installazione; inoltre, poiché l'applicazione è stata sviluppata ex novo, non si è dovuto tenere in considerazione alcun costo relativo alla conversione dei dati appartenenti a qualche sistema preesistente. E' alquanto difficile, invece stimare i costi riguardanti la manutenzione e amministrazione dell'intera applicazione.

1.2.4 Criteri di mantenimento

1. **Estendibilità**: la struttura del sistema permetterà di aggiungervi agevolmente nuove funzionalità e nuove classi.
2. **Modificabilità**: il sistema potrà essere facilmente modificato, poiché le classi risultano essere piuttosto indipendenti tra loro: apportando una modifica su una di esse, non sarà quindi necessario modificare anche le altre.
3. **Adattabilità**: il sistema potrà essere trasferito su differenti domini di applicazione con una facilità che dipenderà essenzialmente dalla nuova applicazione che dovrà essere sviluppata.
4. **Portabilità**: il sistema potrà essere facilmente portato su differenti piattaforme, essendo realizzato interamente in Java.
5. **Leggibilità**: il codice per la realizzazione del sistema risulterà essere quanto più facile da comprendere, anche attraverso l'utilizzo di commenti chiari ed efficaci. Inoltre si cercherà di utilizzare nomi per le funzioni e per le variabili che siano coerenti con il loro significato, in modo da evitare ambiguità e confusione nelle interpretazioni.
6. **Tracciabilità dei requisiti**: il codice verrà strutturato in modo da rendere il più agevole possibile l'individuazione dei vari requisiti che il sistema dovrà soddisfare.

1.2.5 Criteri End User

1. Usabilità: su questo criterio è opportuno soffermarsi. L'usabilità di un sistema può essere analizzata considerando tre diversi principi: apprendibilità, flessibilità e robustezza. Questa applicazione sarà molto facile da apprendere, in quanto avrà una buona prevedibilità, sicchè un utente, anche al primo accesso, riuscirà in poco tempo a comprendere le funzionalità messe a disposizione e a raggiungere gli obiettivi prefissati. Inoltre sarà un sistema molto coerente: operazioni simili tra loro (inserimenti, cancellazioni, modifiche) potranno essere eseguite tutte allo stesso modo. Dal punto di vista della robustezza si può dire che il sistema disporrà di una buona navigabilità, poiché informerà, di volta in volta, l'utente riguardo la sua localizzazione all'interno del sito; inoltre sarà in grado di fornire un buon grado di ripristinabilità, consentendo di correggere facilmente errori commessi durante il suo utilizzo attraverso operazioni di recovery rollback .

2. Architettura software corrente

2.1 Sistema ex-novo

Attualmente non è presente alcun SI capillare sul territorio, in quanto non è stata mai creata o resa pubblica un cartella ostetrica completa nelle informazioni e consistente. Alcuni enti hanno deciso di adottare pen-drive/card o altri strumenti elettronici sul quale venissero memorizzati i dati tuttavia i formati non erano universali e tanto meno condivisibili con altri enti.

3. Architettura software proposta

3.1 Overview

La fase di progettazione del sistema prevede che quest'ultimo sia decomposto in sottosistemi, allo scopo di ridurne la complessità e permettendo di analizzare meglio ogni singolo aspetto dell'applicazione. Dopo un'attenta riflessione, si è deciso di adottare, per la realizzazione del sistema COD, uno stile architetturale client/server: questo tipo di architettura si adatta ottimamente a quelle che sono le caratteristiche principali del sistema in quanto, un sottosistema, detto server, fornisce servizi, e un insieme di client richiedono e sfruttano i servizi offerti dal server. Il collegamento tra server e client avviene tramite la connessione a Internet. Il server contiene un database centralizzato. Gli utenti possono interagire solamente con il Client, quindi il client è responsabile dell'interazione con gli utenti.
Le funzioni principali fornite dal client sono:

- Fornire interfaccia utente personalizzata.
- Elaborazione front-end dei dati per verificare i vincoli.
- Una volta collezionati i dati, avvia la transazione con il server.

Le funzioni principali fornite dal server sono:

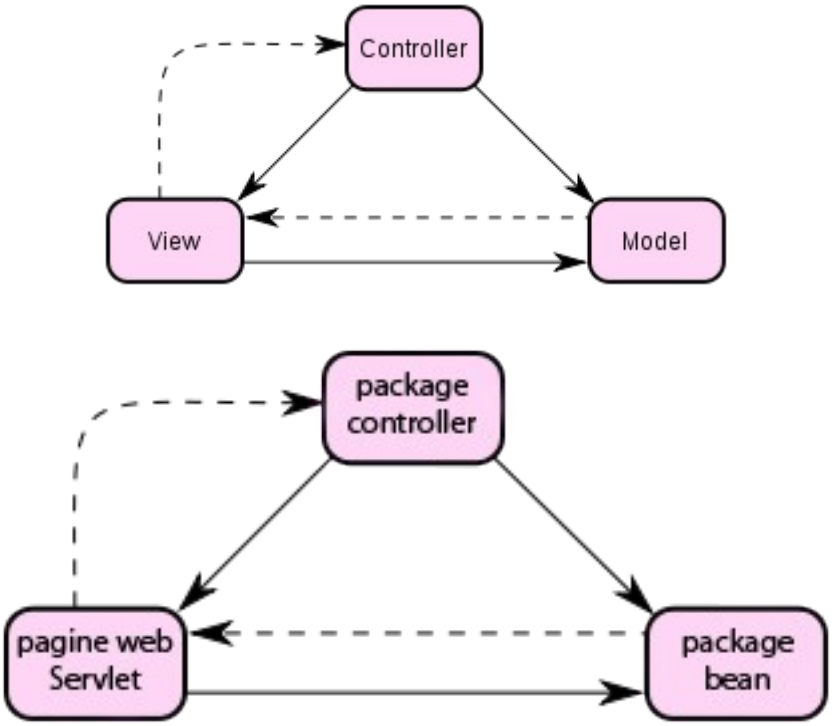
- Gestione centralizzata dei dati.
- Garantire l'integrità dei dati e consistenza del database.
- Garantire la sicurezza del database.
- Gestire la concorrenza delle operazioni(accessi multipli);
- Elaborazione centralizzata.

L'interazione tra client e server avviene tramite un browser web.
Per lo sviluppo del sistema viene utilizzato un pattern architetturale denominato MVC (model - view – controller).
Il pattern è basato sulla separazione dei vari compiti divisi in tre ruoli principali:

- **Model**, composto nel nostro sistema dal package bean, fornisce i metodi per accedere ai dati dell'applicazione;
- **View**, composto nel nostro sistema da tutte le pagine web e dal package businesslogic (Servlet), si occupa dell'interazione tra sistema e utenti;
- **Controller**, composto dal package controller, riceve i comandi dall'utente, attraverso il View, e li attua modificando gli stati degli altri due componenti.

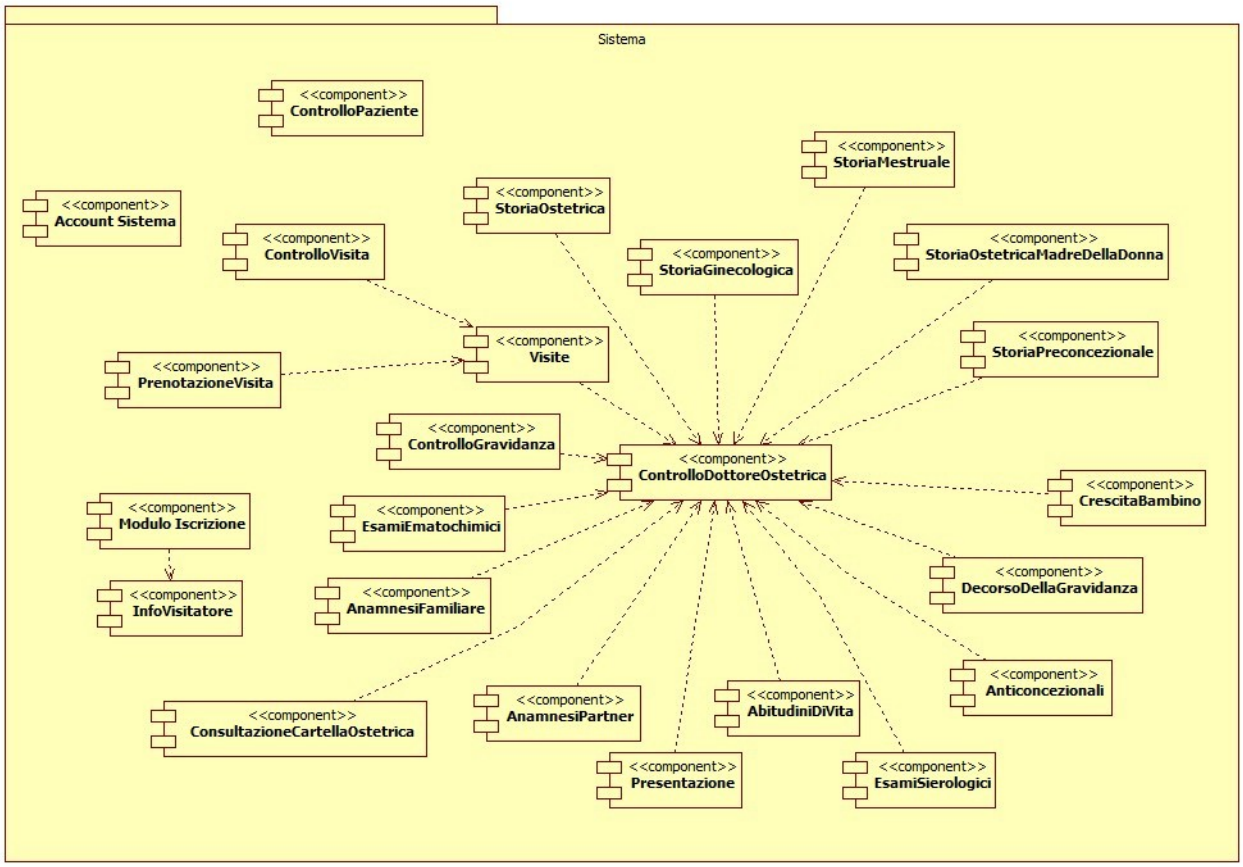
Overview

Di seguito, la struttura del pattern MVC:



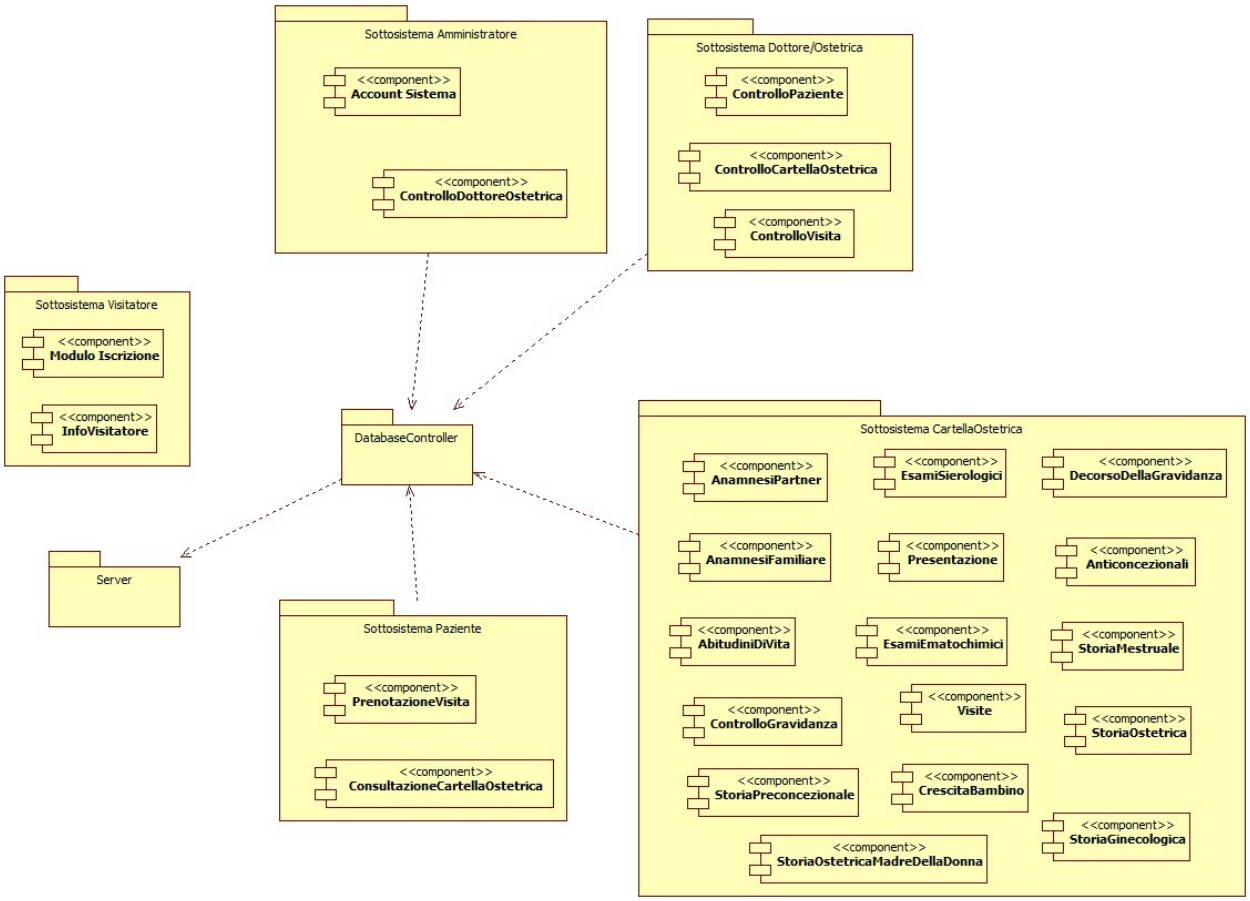
3.2 Decomposizione in sottosistemi.

La decomposizione del sistema deve fornire una semplificazione di quest'ultimo, ma allo stesso tempo deve fare in modo da non creare situazioni in cui vi sia un'eccessiva dipendenza tra i vari moduli. Ciò potrebbe far sì che, in caso di eventuali modifiche ad una singola componente, possa essere necessario anche modificare in maniera radicale quelle che dipendono da essa.
Per cercare di monitorare tali rischi, si è deciso di partizionare il sistema in sottosistemi pari (peer), ognuno dei quali è responsabile di un certo insieme di servizi. Di seguito viene mostrato il Diagramma delle Componenti relativo all'applicazione da progettare, all'interno del quale vengono messi in evidenza i singoli moduli individuati all'interno del sistema e le relazioni di dipendenza tra ognuno di essi:



Nel seguente grafico, invece, viene proposta la suddivisione in sottosistemi, che è stata effettuata tenendo conto delle considerazioni fatte precedentemente:

Decomposizione in sottosistemi.



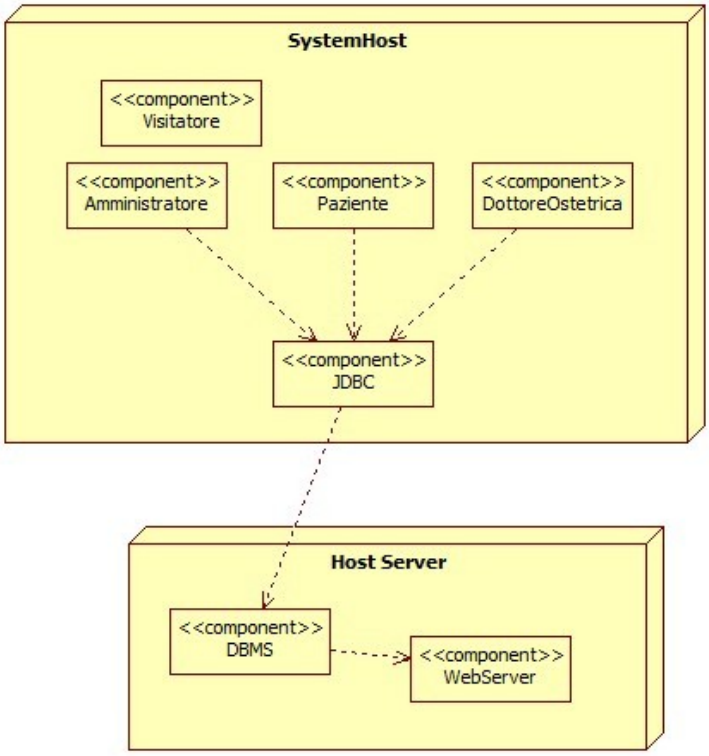
Come si può ben vedere dal diagramma, i sottosistemi individuati sono i seguenti:

- Sottosistema Visitatore: rappresenta la sezione che si occuperà di gestire le richieste di utenti che non sono registrati al sistema;
- Sottosistema Amministratore: fornirà i servizi necessari alla gestione dell'intera applicazione;
- Sottosistema Dottore/Ostetrica: fornirà i servizi necessari alla gestione dei Pazienti, delle cartelle ostetriche e delle visite.
- Sottosistema Paziente: fornirà i servizi necessari alla visualizzazione della propria cartella ostetrica e alla prenotazione di una nuova visita.
- Sottosistema CartellaOstetrica: è composto da tutte le informazioni riguardanti la cartella ostetrica.

3.3 Mappaggio su hardware e software

Dopo aver effettuato la decomposizione in sottosistemi, è necessario determinare

l'allocazione delle risorse hardware e software relative al sistema. Siccome l'applicazione sarà implementata interamente in JSP (Java Server Pages),linguaggio java sviluppato per il web, con interfacce grafiche in HTML e CSS, risulterà essere un'applicazione multiutente sviluppata per il web. Tutte le risorse software risiederanno su un server, attraverso la quale ogni utente registrato al sistema potrà accedervi, tramite la propria macchina, attraverso login e password. Per mostrare l'effettiva distribuzione delle varie componenti, è stato realizzato il seguente Deployment Diagram:



Il server comunicherà col database centrale attraverso la componente software JDBC, un'API composta da un'insieme di classi e interfacce scritte in Java che forniscono agli sviluppatori la possibilità di scrivere applicazioni robuste per interfacciarsi con i database usando esclusivamente tecnologia Java.
Inoltre verrà utilizzata Java Persistence API che memorizza i dati della connessione JDBC di un database in file XML ed ha i seguenti vantaggi:

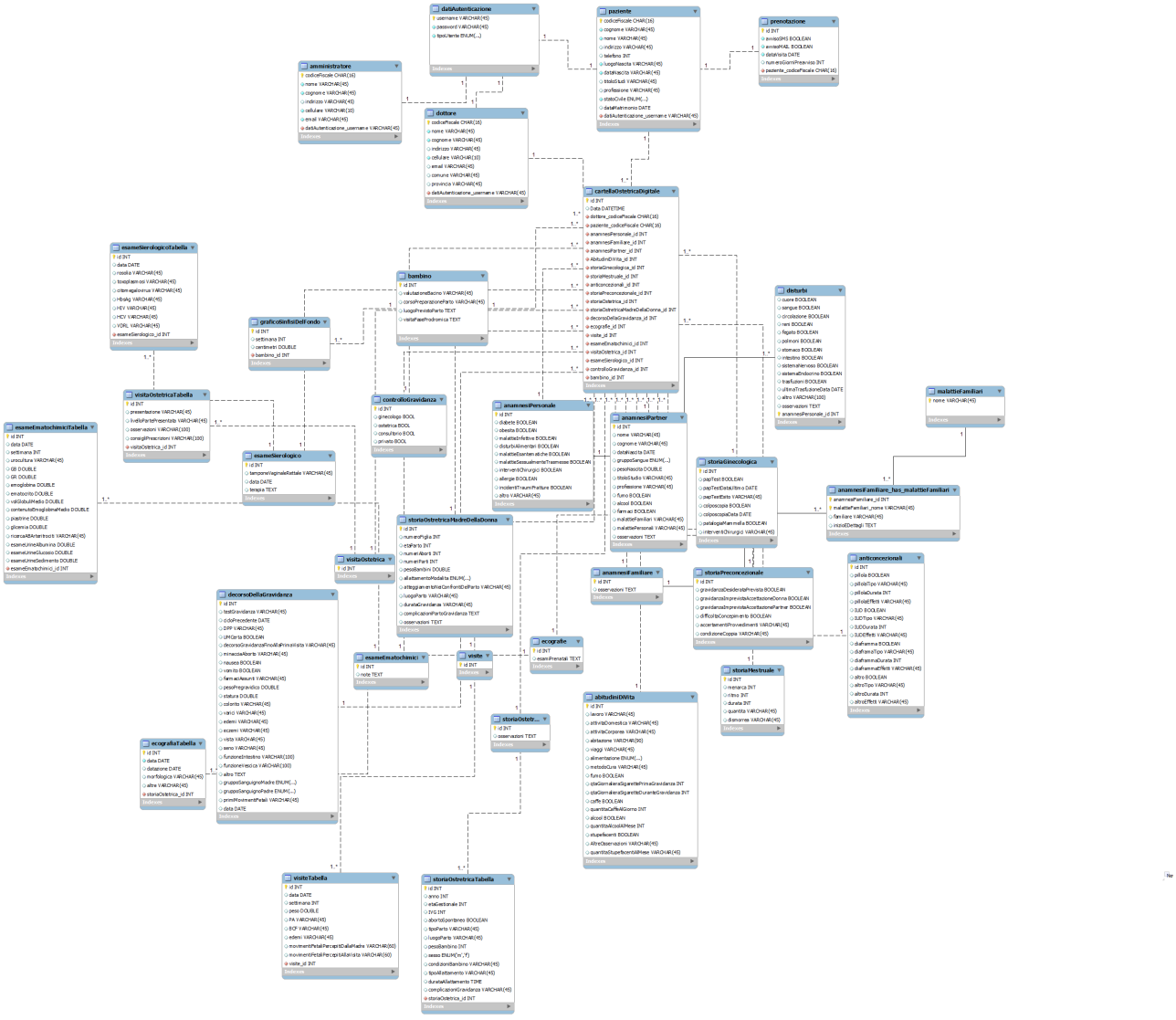
- svincola l'applicazione dall'indirizzo del server di database, quindi una modifica della locazione del server database non necessita della ricompilazione dell'applicazione;
- svincola l'applicazione dal database in quando il driver JDBC viene caricato a caldo durante l'inizializzazione dell'applicazione, quindi nel caso si sceglie di cambiare il server di database basta aggiungere la libreria JDBC del nuovo database alle librerie dell'applicazione, modificare il file di configurazione di

Java Persistence API e riavviare l'applicazione.

Le effettive operazioni sulla base di dati verranno effettuate da un DBMS, ossia una componente software in grado di gestire pienamente la creazione, la modifica e la cancellazione di database, nonché l'esecuzione di query sugli stessi. Il DBMS scelto che utilizzeremo è MySql. Per usare le API di JDBC con un particolare DBMS occorrerà un driver basato su tecnologia JDBC che avrà il compito di mediare tra JDBC e il database.

3.4 Gestione dei dati persistenti.

Il sistema di gestione per il COD dovrà essere in grado di gestire una gran quantità di informazioni, relative sia ai pazienti che ai dottori e ostetriche. Per questo motivo si è deciso di affidare il controllo di questi dati ad un database relazionale, che potrà essere gestito attraverso un DBMS, al quale l'applicazione impartirà determinati comandi attraverso la mediazione delle API costituenti JDBC. Di seguito viene mostrato lo schema E-R rappresentante la base di dati relativa al COD.



3.5 Controllo di accesso e aspetti relativi alla sicurezza.

Il sistema richiede un meccanismo di autenticazione in modo da differenziare i vari accessi. Al primo accesso al sistema dell'utente viene creato un oggetto Control che contiene le informazioni relative ai permessi di accesso al sistema. Quindi i vari utenti che accederanno al sistema verranno attribuiti all'oggetto Control per tutto il ciclo di vita dell'utente che accede al sistema. Di default il livello dell'utente è visitatore e per cambiare questo livello basterà inserire userId e password all'interno di una TextBox. Il sistema successivamente andrà a verificare nel database la tupla (ID,password) digitata dall'utente. Nel caso la tupla non sia presente il livello dell'utente rimarrà a visitatore e il sistema mostrerà un messaggio di errore "Login o password errata" invitando all'utente di reinserire i campi correttamente. In caso contrario, l'utente accede alle varie operazioni in base ai propri privilegi.
In questo caso si avranno vari livelli:

- Visitatore: l'utente visitatore non accede al sistema e può soltanto alla homepage di COD.
- Amministratore : l'utente amministratore ha diritto alla gestione di tutti i dottori/ostetriche, quindi un nuovo inserimento, una modifica dati oppure una eliminazione dal database.
- Dottore/ostetrica: l'utente Dottore/Ostetrica accede all'area relativa ai dottori/ostetriche ed ha i privilegi di poter accedere a tutte le cartelle ostetriche relative ai propri pazienti, effettuarne modifiche, e gestirne la prenotazione delle visite.
- Paziente: l'utente paziente accede all'area relativa ai pazienti ed ha il privilegio di controllare la propria cartella ostetrica e di richiedere la prenotazione di una nuova visita.

3.6 Flusso di controllo globale.

In questa fase è necessario cercare di descrivere la sequenza azioni che il sistema deve compiere per eseguire una richiesta. Si è scelto di usare un controllo del flusso globale di tipo event-driven control (controllo guidato dagli eventi) molto adatto per le applicazioni dotate di interfaccia utente. Il sistema attende che si verifichi un determinato evento (premi bottone,apri menu..) e quando accade viene spedito il controllo al componente interessato.
Ad esempio il sistema è strutturato in una homepage a cui possono accedere tutti gli utenti. Si può dire che questo è lo stato iniziale del sistema. Da questo momento in poi l'applicazione attende che su verifichi un evento che può essere una autenticazione oppure una richiesta di registrazione.

3.7 Condizioni limite.

Nel sistema si avrà una fase di inizializzazione in cui l'addetto accede alla macchina, avvia un browser, e si collega alla homepage di www.cod.org. Per l'avvio del sistema non è necessario alcuna informazione particolare di configurazione. Per accedere invece alle varie aree private bisogna essere registrati dall'amministratore e quindi disporre di un proprio account.
All'avvio del sistema l'interfaccia utente mostra una pagina iniziale che mostra una finestra di login e password validi per l'autenticazione.
L'interfaccia si presenta all'utente attraverso una pagina molto visuale e intuitiva comprendente un menù, delle caselle di testo e dei bottoni che gestiscono le varie operazioni accessibili all'area pubblica.
La fase di terminazione si ha quando il sistema cessa di fornire servizi agli utente. Questa fase può avvenire quando l'utente autenticato decide di effettuare il logout dal sistema.
Aggiornamenti locali del sistema quali inserimento, modifiche, cancellazione dei vari oggetti presenti nel sistema sono comunicati al database attraverso un sottosistema di tipo reposity. La fase di fallimento sarà gestita dal server del sistema. Dovendo essere un sistema sicuro e che garantisca l'affidabilità, il server provvederà, tramite sistema RAID e Backup, a garantire che il sistema funzioni sempre.
Per ogni oggetto persistente ora verranno definite le fasi in cui vengono creati, distrutti e archiviati nel database.

- Account(Dottore/Ostetrica) : l'oggetto Dottore/Ostetrica viene creato e distrutto dall'Amministratore.
- Account(Paziente): l'oggetto Paziente viene creato e distrutto dal Dottore/Ostetrica.
- CartellaOstetrica: l'oggetto CartellaOstetrica viene creato dal Dottore/Ostetrica e viene distrutto nel momento in cui il Dottore/Ostetrica intende distruggere un oggetto Paziente.
- Visita: l'oggetto visita viene creato e distrutto dal Dottore/Ostetrica.

3.7.1 Casi d'uso per StartUp.

3.7.2 StartSistema

Nome Use Case	StartSistema
Partecipanti	Inizializzato dall'utente
Flusso Eventi	• L'utente accede alla macchina da cui eseguire il sistema.

	• Il sistema si mette in attesa di un input dall'utente. • L'utente avvia il browser e si collega al sito www.cod.org. • Il browser si collega al sito e visualizza una form di login. • L'utente inserisce i dati di autenticazione e preme sul tasto per il Login. • Il sistema controlla la correttezza dei dati di autenticazione .
Condizioni d'ingresso	L'utente accede alla macchina.
Condizioni d'uscita	L'utente è loggato al sistema con successo.
Eccezione	Il visitatore non inserisce i dati di autenticazione in modo corretto.

3.7.3 StartAccount

Nome Use Case	StartAccount
Partecipanti	Inizializzato dall'Amministratore
Flusso Eventi	• L'amministratore riceve la mail di richiesta abilitazione da parte di un dottore/ostetrica e accede all'area gestione account. • Il sistema risponde mostrando a video tutte le informazioni riguardanti il nuovo account. • L'Amministratore seleziona l'operazione effettua inserimento account. • Il sistema risponde mostrando a video un form per la conferma dell'esattezza dei dati. • L'amministratore conferma i dati ricevuti. • Il sistema inserisce i nuovi dati in input, genera la coppia

Condizioni limite.

	<”username”-”password”>, la invia via mail al Dottore/Ostetrica e notifica l’avvenuta operazione.
Condizioni d'ingresso	L’Amministratore riceve una mail di abilitazione account.
Condizioni d'uscita	L'amministratore abilita un (ex)visitatore al COD e quest'ultimo da adesso verrà considerato dal Sistema come un Dottore/Ostrica
Eccezione	I dati ricevuti dall'amministratore sono errati oppure non soddisfano i requisiti di abilitazione al COD(il visitatore non è inscritto all'albo dei medici).

3.7.4 StartCartellaOstetrica

Nome Use Case	StartCartellaOstetrica
Partecipanti	Inizialiazzato dal Paziente.
Flusso Eventi	• Il Paziente accede al sistema. • Il sistema risponde mostrando a video tutte le informazioni della cartella ostetrica del relativo paziente.
Condizioni d'ingresso	Il Paziente ha effettuato il login
Condizioni d'uscita	
Eccezione	

3.7.5 StartVisualizzaPaziente

Nome Use Case	StartVisualizzaPaziente
Partecipanti	Inizializzato dal Dottore/Ostetrica.

Flusso Eventi	• Il paziente accede all'area gestione Pazienti • Il sistema risponde mostrando a video tutte le possibili operazioni di gestione pazienti • Il Dottore/Ostetrica seleziona l'operazione di ricerca paziente. • Il sistema risponde mostrando a video un form per effettuare la ricerca del Paziente • Il Dottore/Ostetrica inserisce i dati e avvia la ricerca • Il sistema mostra un form contenente l'elenco dei Pazienti trovati. • Il Dottore/Ostetrica seleziona il paziente cercato interessato. • Il sistema mostra un form contenente tutte le informazioni riguardanti il Paziente.
Condizioni d'ingresso	
Condizioni d'uscita	

3.7.6 StartVisita

Nome Use Case	StartVisita
Partecipanti	Inizializzato dal Dottore/Ostetrica.
Flusso Eventi	• Il Dottore/Ostetrica avvia la funzione Inserisci prenotazione visita. • Il sistema mostra un form per l'inserimento della data della visita. • Il Dottore/Ostetrica inserisce la data della visita da effettuare e preme sul pulsante memorizza • Il sistema memorizza la nuova visita con la relativa data.

Condizioni limite.

Condizioni d'ingresso	Il Dottore/Ostetrica effettua la ricerca del paziente.
Condizioni d'uscita	
Eccezione	Dati Errati.

3.7.7 Casi d’uso per Shutdown

3.7.8 ShutDownSistema

Nome Use Case	ShutDownSistema
Partecipanti	Inizializzato dall’utente.
Flusso Eventi	• L'utente clicca sul pulsante “Logout” • Il sistema mostra una finestra di conferma di Logout. • L'utente conferma il Logout. • Il sistema visualizza la homepage del sistema.
Condizioni d'ingresso	L'utente è loggato al sistema.
Condizioni d'uscita	
Eccezione	

3.8 Comportamento del sistema in condizioni eccezionali.

Il sistema visualizzerà un messaggio di errore quando un utente non si riuscirà a connettere al Database o quando l'utente inserirà dei dati di autenticazione errati. In caso di operatività parziale del DB saranno possibili solo operazioni di visualizzazione. Questo comportamento si avrà quando si cercherà di memorizzare informazioni oltre il limite di memoria del DB. Altri comportamenti da gestire saranno quelli associati ad errori di incoerenza dei dati che si potranno verificare nelle operazione di inserimento o modifica associati ai vari casi d'uso. Delle situazioni da gestire saranno quelle in cui un utente tenterà di accedere a operazioni non consentite.

4. Servizi dei sottosistemi

4.1 Prefazione

Utilizzando come base i casi d'uso e seguendo un'architettura Client-Server il sistema viene decomposto in sottosistemi con l'obiettivo di snellire e semplificare l'applicazione.
Questa operazione di semplificazione è stata fatta tenendo in considerazione le dipendenze tra i vari moduli, con l'obiettivo di evitare situazioni in cui la modifica di un modulo comporti una sostanziale revisione di tutti gli altri moduli ad esso collegati.

4.2 Proprietà

Il sistema è stato diviso in sottosistemi tenendo presente due proprietà:

4.2.1 Accoppiamento (coupling)

Misura quanto un sistema è dipendente da un altro. I nostri sottosistemi sono loosely coupled (leggermente accoppiati) in quanto una modifica in un sottosistema avrà poco impatto nell'altro sistema, mentre nei sistemi strongly coupled (fortemente accoppiati) se si verificasse una modifica su uno dei sottosistemi si otterrebbe un forte impatto sull'altro. La condizione ideale di accoppiamento da noi scelta richiede meno sforzo quando devono essere modificate delle componenti,centrando l'obiettivo da noi prefissato.
Ad esempio, tre componenti usano lo stesso servizio esposto da una componente che potrebbe essere modificata spesso, conviene frapporre tra di essere una nuova componente che ci permette di evitare una modifica alle tre componenti che usufruiscono del servizio.

Ovviamente ove non sono presenti componenti che si pensa debbano essere modificate spesso, non conviene utilizzare questa strategia in quanto aggiungerebbe complessità di sviluppo e di calcolo al sistema.

4.2.2 Coesione

Misura la dipendenza tra le classi contenute in un sottosistema. La coesione è alta se due componenti di un sottosistema realizzano compiti simili o sono collegate l'una con l'altra attraverso associazioni (ereditarietà), è invece bassa nel caso contrario. L'obiettivo è quello di avere sottosistemi con coesione interna alta.
La decomposizione del sistema avviene utilizzando layer e/o partizioni.

4.3 Sottosistemi individuati

I sottosistemi individuati sono i seguenti:

- **Sottosistema Visitatore**: rappresenta la sezione che si occuperà di gestire le richieste di utenti che non sono registrati al sistema e si occuperà della visualizzazione delle informazioni generali riguardanti il sistema;
- **Sottosistema Amministratore:** fornirà i servizi necessari alla gestione dell'intera applicazione,in particolare la gestione degli account e la gestione dei Dottori/Ostetriche registrati al sistema;
- **Sottosistema Dottore/Ostetrica**: fornirà i servizi necessari alla gestione dei Pazienti, delle cartelle ostetriche e delle visite;
- **Sottosistema Paziente**: fornirà i servizi necessari alla visualizzazione della propria cartella ostetrica e alla prenotazione di una nuova visita;
- **Sottosistema NotifyProvider**: fornirà i servizi necessari all'invio di email ed sms.