Egoless Group
egolessgroup@googlegroups.com

Definizione di Prodotto per C04/PQS

Versione 2.0

Stato del documento:
Formale ed
Esterno

Sommario :

Questo documento si prefigge di illustrare la progettazione di dettaglio per il prodotto relativamente al capitolato **C04**: "**Sistema *software* per l'informatizzazione della gestione di qualità a norma ISO 9000:2000 nelle scuole**".

Redazione

Nominativo	Ruolo	Data
Stefano Gazzola	Progettista	16 Febbraio 2007
Eric Miotto	Progettista	17 Febbraio 2007

Lista di Distribuzione

Nominativo	Ruolo
Tullio Vardanega	Committente
Renato Conte	Committente
Lucia Meneghello	Verificatore
Margherita Collicelli	Responsabile
Eric Miotto	Amministratore/Programmatore
Stefano Gazzola	Programmatore
Roberto Pordon	Verificatore
Lorenzo Daniele	Programmatore
Alberto Meneghello	Programmatore

Approvato da:

Versione	Nominativo	Data
2.0	Margherita Collicelli	16 marzo 2007

Registro delle Modifiche:

Versione	Autore	Data
2.0	Margherita Collicelli	16 marzo 2007
Approvato documento per la Revisione di qualifica.		
1.6	Lucia Meneghello	16 marzo 2007
Verificato documento.		
1.5	Stefano Gazzola	15 marzo 2007
Correzioni varie.		
1.4	Lorenzo Daniele	14 marzo 2007
Steso capitolo sulla GUI di APPDidattica con l'aggiunta di diagrammi (5.1.5).		
1.3	StefanoGazzola	14 marzo 2007
Inseriti consigli per l'utilizzo dei Bean		
1.2	Eric Miotto	10 marzo 2007
Aggiorrnata sezione 2.2 sugli strumenti.		
1.1	Margherita Collicelli	27 febbraio 2007
Verifica sintattica.		
1.0	Lorenzo Daniele	23 febbraio 2007
Approvato documento per la Revisione di Progetto Definitiva.		
0.8	Stefano Gazzola	22 febbraio 2007
Modifica capitoli 5,cambiamento della scaletta, tabella classi bean,immagine WS.		
0.7	Stefano Gazzola	22 febbraio 2007
Modificato capitolo 4.1 la parte Model.		
0.6	Lucia Meneghello	21 febbraio 2007
Aggiornata lista di distribuzione.		
0.59	Stefano Gazzola	20 febbraio 2007
Integrata parte relativa ai Model e alla loro implementazione con JavaBeans		
0.58	Eric Miotto	19 febbraio 2007
Aggiunto riferimento alle norme di progetto. Integrata sezione 4. Piccole correzioni all'indice.		
0.57	Eric Miotto	18 febbraio 2007
Rivista struttura del documento ed indice.		
0.56	Stefano Gazzola	18 febbraio 2007
Modificato capitolo 5.1,5.2 e 6		

0.55	Eric Miotto	18 febbraio 2007

Rivista sezione 3. Aggiunte sezioni 3.2.2 e 3.2.3.

0.51	Eric Miotto	18 febbraio 2007

Corretto qualche piccolo errore.

0.5	Stefano Gazzola	18 febbraio 2007

Correzione ordinamento registro modifiche,correzioni varie di sintassi

0.4	Stefano Gazzola	17 febbraio 2007

Sviluppo capitolo 3

0.3	Stefano Gazzola	17 febbraio 2007

Inserimento del capitolo 3

0.2	Eric Miotto	17 febbraio 2007

Corretto frontespizio e piè di pagina, integrato indice, primo contenuto per la sezione2.2.

0.1	Stefano Gazzola	16 febbraio 2007

Prima bozza di documento e definizione dei capitoli

Indice

1 Introduzione

1.1 Scopo del documento

Il seguente documento si propone di dettagliare nel modo più chiaro e corretto possibile come le componenti del progetto architetturale (vedi [ST]) debbano essere implementate.

Più in particolare si cercherà di decomporre ciascuna delle componenti architetturali in moduli a grana più fine dotati di una propria identità e valutati attraverso metriche precise per definirne la loro natura. In questo modo conoscendo le caratteristiche di ciascun modulo sarà più semplice poter pianificarne temporalmente lo sviluppo massimizzando il grado di parallelismo.

Le scelte dell'analisi di dettaglio saranno quindi fortemente guidate da valutazioni oggettive che permettano di conferire ai moduli caratteristiche tali per cui anche le attività di sviluppo e verifica siano il più efficienti e semplici possibile.

Verranno poi fornite delle linee guida che diano al programmatore tutte le informazioni necessarie per concretizzare il prodotto al fine di circoscrivere il suo campo d'azione e di indirizzarlo correttamente nello sviluppo.

1.2 Scopo del prodotto

Per lo scopo del prodotto fare riferimento a [AR].

1.3 Riferimenti

- [AR] Analisi dei Requisiti, Egoless Group
- [G] Glossario, Egoless Group
- [PP] Piano di progetto, Egoless Group
- [ST] Specifica tecnica, Egoless Group
- [NP] Norme di Progetto, Egoless Group
- [UML] Specifica di UML, versione 2.0, OMG
  http://www.uml.org/
- [T-JEE5] Tutorial Java Enterprise Edition 5
- [JD_AppD]: Documentazione JavaDoc di APPDidattica

2 Standard di progetto

2.1 Norme di progettazione e di codifica

Per le norme di progettazione di codifica si fa riferimento a [NP].

2.2 Strumenti di lavoro

Per realizzare il prodotto verranno impiegate le seguenti tecnologie:

- Java 6.0;
- Java Enterprise Edition 5.0;
- Glassfish 1.0 UR1 P01 Build 02 (equivalente a Sun Java System Application Server Platform Edition 9);
- JAX-WS 2.0;
- NetBeans 5.5 con Enterprise Pack.

3 Struttura e funzionamento dei Web Service

Questo capitolo presenta la concezione e il funzionamento di un Web Service implementato utilizzando gli strumenti forniti da Java Enterprise Edition. Tale presentazione illustrerà le funzionalità e i modi di operare che saranno comuni sia per l'interazione tra un modulo applicativo (APP-...) e il corrispondente Web Service (WS-...) sia tra il Web Service stesso e lo strato sottostante WEB SERVICE-API.

3.1 Modello di una applicazione Java EE

Questo capitolo presenta la concezione e il funzionamento di un Web Service implementato utilizzando gli strumenti forniti da Java Enterprise Edition. Tale presentazione illustrerà le funzionalità e i modi di operare che saranno comuni sia per l'interazione tra un modulo applicativo (APP-...) e il corrispondente Web Service (WS-...) sia tra il Web Service stesso e lo strato sottostante WEB SERVICE-API

3.1.1 Modello generale

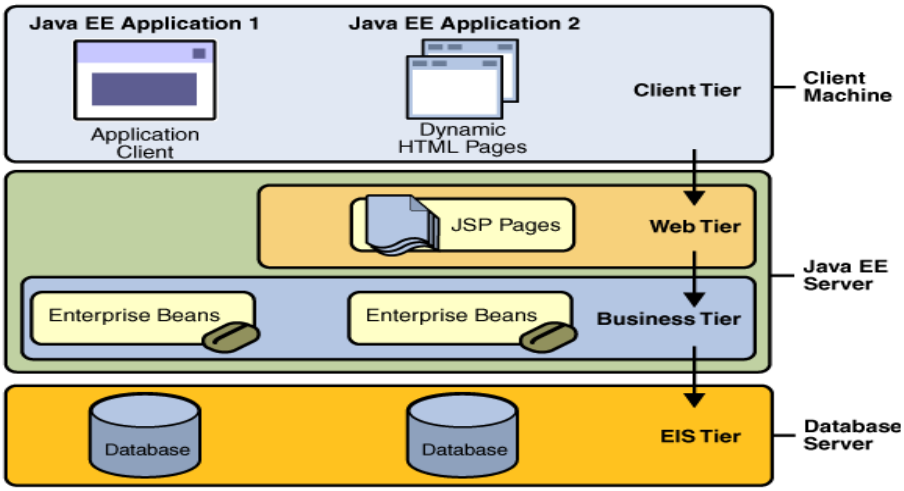


Illustrazione 1: Modello adottato dalla piattaforma JEE (immagine presa dal JEE 5 Tutorial)

La piattaforma Java EE utilizza un modello multi-layered per sviluppare le proprie applicazioni enterprise. I livelli che ci toccano più direttamente per la nostra applicazione sono:

- Il Client Tier che si trova in una macchina client e conterrà l'applicativo (Application Client) che vuole fruire del servizio.
- Il Business Tier che si trova sulla macchina server e conterrà gli applicativi relativi ai servizi che vorremo esporre (Enterprise Beans) e che saranno consumati.

L'insieme delle funzionalità offerte dalla piattaforma Java EE prevede che Client Tier e Business Tier comunichino direttamente fra loro, mettendo a disposizione strumenti che catturino i bisogni del Client e sappiano interfacciarli con il Server.

3.1.2 I Container

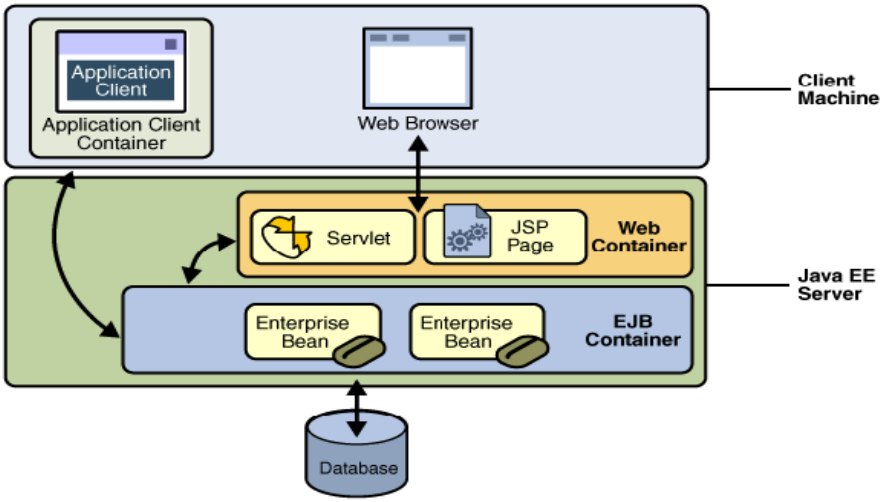


Illustrazione 2: Utilizzo dei container(immagine presa dal JEE 5 Tutorial)

Sulla base di quanto precedentemente esposto è facile vedere che un approccio di questo tipo porta ad un fiorire di problematiche a basso livello non indifferenti (controllo di stato, multithreading, resource polling...) che richiederebbero un notevole sforzo di progettazione e sviluppo.

Proprio per questo la piattaforma Java EE mette a disposizione diverse tipologie di contenitori (Container) che forniscono servizi diversi a seconda delle nostre esigenze. Questa proprietà ci permetterà di focalizzare i nostri sforzi più sulla risoluzione dei problemi relativi al servizio offerto e non tanto sui problemi di contorno. Avendo a disposizione già funzionalità fatte e testate ci basterà parametrizzarle e personalizzarle a seconda delle nostre esigenze.

L'utilizzo dei container sarà particolarmente utile per il supporto che ci forniranno per la gestione della sicurezza e delle autorizzazioni per l'accesso alle risorse.

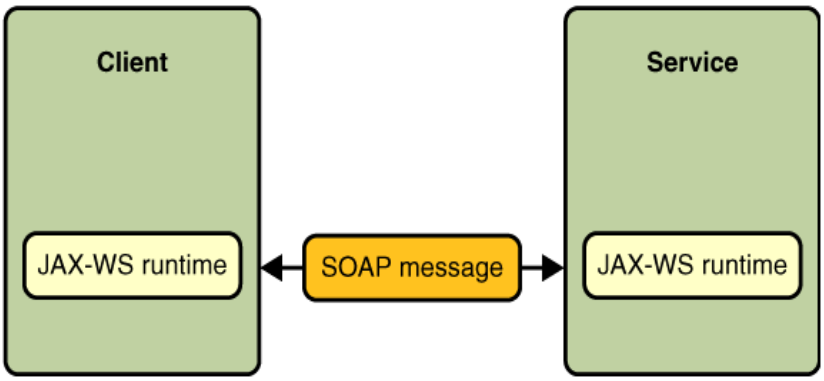
3.2 Comunicazione con il Web Service

3.2.1 Utilizzo e funzionamento di JAX-WS

Ciascuna applicazione, oltre ad essere opportunamente inserita nel proprio Container, dovrà poter comunicare con la propria controparte per richiedere/fornire un servizio.

Il protocollo a livello applicativo su cui si baserà tale comunicazione è SOAP (Simple Object Application Protocol) ,protocollo message-driven dove l'informazione sarà plasmata in messaggi costruiti con il linguaggio XML. Questo garantirà di poter sviluppare eventualmente le varie componenti con linguaggi diversi. L'uso di un file WSDL (Web Service Description Language) permette inoltre agli utenti di sapere quali

sono i messaggi e i tipi supportati dal Web Service.



Considerate le premesse nasce spontaneo il bisogno di interfacciare il linguaggio Java a tale protocollo di comunicazione e di gestire la conversione dell'informazione nelle due direzioni. Questo compito sarà delegato ad una API messa a disposizione da Java EE,JAX-WS 2.0, che consentirà di mappare in automatico l'informazione contenuta nei messaggi con le classi Java corrispondenti.

Il protocollo a livello rete su cui invece si baserà la comunicazione è quello HTTP dove la sicurezza verrà garantita dal SSL. Questo permetterà di dislocare le varie componenti in vari punti di Internet.

3.2.2 Messa in opera di un Web Service

Per creare un Web Service nel nostro prodotto si creerà un Enterprise JavaBean. In particolare, ci sarà una classe che sarà annotata con l'annotazione `@WebService`: questo serve ad indicare che la classe implementerà la logica del Web Service. La classe avrà vari metodi: quelli pubblici che dovranno essere esposti tramite Web Service dovranno essere annotati con `@WebMethod`.

La compilazione da NetBeans crea un normale file JAR. Quando però viene fatto il *deployment* di quest'ultimo nel server, viene avviato il tool `wsgen` che analizza il bytecode prodotto, legge le annotazioni e crea tutta l'infrastruttura necessaria per mettere in opera il Web Service.

3.2.3 Utilizzo di un Web Service

Per consumare un Web Service, è necessario creare un cosiddetto Web Service Client da un progetto in NetBeans. Una volta specificato il percorso del WSDL che descrive il Web Service, verrà invocato il tool `wsimport` che analizzerà il WSDL e genererà una serie di classi tramite le quali sarà possibile comunicare con il Web Service senza doversi preoccupare dei dettagli.

4 Specifica delle componenti dello strato applicativo

4.1 Pattern architetturale usato per ciascuna componente

Se focalizziamo la nostra attenzione sui primi due strati (applicativo e logico) della nostra architettura possiamo classificare il nostro sistema come “interattivo”. Questo perché il sistema cattura particolari eventi che vengono scatenati dall'esterno e si propone di servirli in maniera opportuna. Più in particolare il sistema si interfaccerà con l'utente fornendogli il modo di richiedere dati, che verranno presentati sotto forma di particolari viste, e di modificarli attraverso opportuna funzionalità. Tutto questo si svolgerà secondo un rapporto di “modifica-propagazione” secondo il quale ciascun cambiamento di stato dei dati produrrà una propagazione delle modifiche sulle viste di cui disponiamo.

In base a queste considerazioni il nostro sistema,nei suoi primi due strati, viene modellato utilizzando un pattern architetturale opportuno, l'MVC, che descrive perfettamente le parti in gioco nel sistema: il “Model”, le “View” e i “Controller”.

Ciò che ci proponiamo di fare ora è quello di calare tale pattern nel nostro contesto dando un identità alle parti astratte appena presentate.

Controllers and Views
Per quanto riguarda l'interfaccia grafica che forniremo al nostro applicativo sarà sviluppata utilizzando le librerie grafiche Swing fornite da Java. Più in particolare avremmo un unico frame che conterrà più tab, una per ogni dominio di interesse. Ciascuna View sarà quindi identificata proprio con ciascuna tab che presenterà i dati richiesti attraverso opportuni componenti (Jtable, eccetera, ...).

La parte di Controller è intrinseca all'interfaccia grafica. Infatti a ciascun componente sono associati eventi diversi che saranno opportunamente catturati e gestiti da particolari gestori di eventi (i listener) messi a disposizione dalle classi di Swing. Una volta catturato l'evento si cercherà di interagire con il Model attraverso le funzionalità esposte da quest'ultimo.

Model
Il model contiene i dati che verranno poi visualizzati dall'interfaccia grafica. Il model in prima approssimazione si preoccuperà di:

- caricare oggetti da Web Service;
- permettere la modifica di tali oggetti;
- segnalare ai soggetti interessati le modifiche fatte sugli oggetti e di fornire dettagli su tali modifiche. Si implementa quindi il pattern Observer;

- restituire gli oggetti memorizzati al Web Service.

Più in particolare saranno previsti più Model, uno per ciascuna area di interesse, che incapsuleranno al proprio interno uno o più oggetti del web service. I Model saranno implementati come classi JavaBeans.

I JavaBeans avranno le seguenti responsabilità:

- recuperare gli oggetti necessari dal web service sottostanti appoggiandosi alle classi generate da JAX-WS;
- permettere l'accesso alle proprietà degli oggetti tramite l'interfaccia del JavaBeans;
- il pattern Observer verrà implementato tramite un oggetto di tipo `propertyChangeSupport` e l'interfaccia `propertyChangeListener` previsti per i JavaBean; verrà notificata ogni modifica apportata a qualche proprietà tramite lancio di eventi;
- restituire al web service gli oggetti generati modificati tramite le classi generate da JAX-WS. Il salvataggio degli oggetti avverrà solo se i dati sono realmente cambiati: questo comportamento è garantito dal JavaBean;
- permettere di essere duplicati per poter generare più modelli a partire dagli stessi dati o per poter ripristinare i dati prima di una modifica estensiva. Ciò è ottenuto implementando l'interfaccia Cloneable e definendo il metodo Clone di Object.

4.2 Considerazioni di valenza generale

Modularità delle GUI e Riuso

Ciascun componente disporrà quindi di una parte di Views e di una Controllers che verrà identificata da qui in avanti con il termine GUI (Graphic User Interface). Tale GUI sarà costruita cercando di individuare più moduli, più classi che possano essere costruite e sviluppate con il massimo grado di parallelismo possibile.
Inoltre,considerato che all'interno della GUI ci sono molte parti in comune (per esempio funzionalità di ricerca) si farà particolare attenzione al riuso delle classi che descriveranno tali parti.

5 Specifica delle componenti dello strato applicativo

5.1 APP Didattica [CSA-01]

5.1.1 Tipo, obiettivo e funzione

Riferimento alla [ST]

5.1.2 Relazioni d'uso di altre componenti

Riferimento alla [ST]

5.1.3 Elenco classi che formeranno il Model

Nella [JD_AppD] vengono descritte le caratteristiche delle classi Bean che andranno a formare la parte Model dell'APP-DIDATTICA.

5.1.4 Note per l'uso dei Bean

Trattamento delle relazioni tra gli oggetti

Quando il programmatore di GUI deve far uso di classi che sono legate logicamente ad altre deve tener presente le seguenti indicazioni.

Se ad esempio si è caricato uno StudenteBean ad esso saranno associati una certa quantità di voti. Si è scelto di non creare una lista di oggetti di tipo Voto interna alla classe StudenteBean. Se da un lato sarebbe comodo tirar su la classe dal Web Service già con tutti i riferimenti dall'altra diventa difficile gestirli. In questo modo se vogliamo conoscere tutti i voti legati ad uno studente basta fare una ricerca di voti parametrizzata con l'identificativo desul particolare studente. Da tali oggetti di tipovoto costruire attorno il Beans e fare le modifiche del caso.

In definitiva ogni oggetto che ha una relazione con un altro oggetto avrà il suo id che funzionerà da chiave esterna.

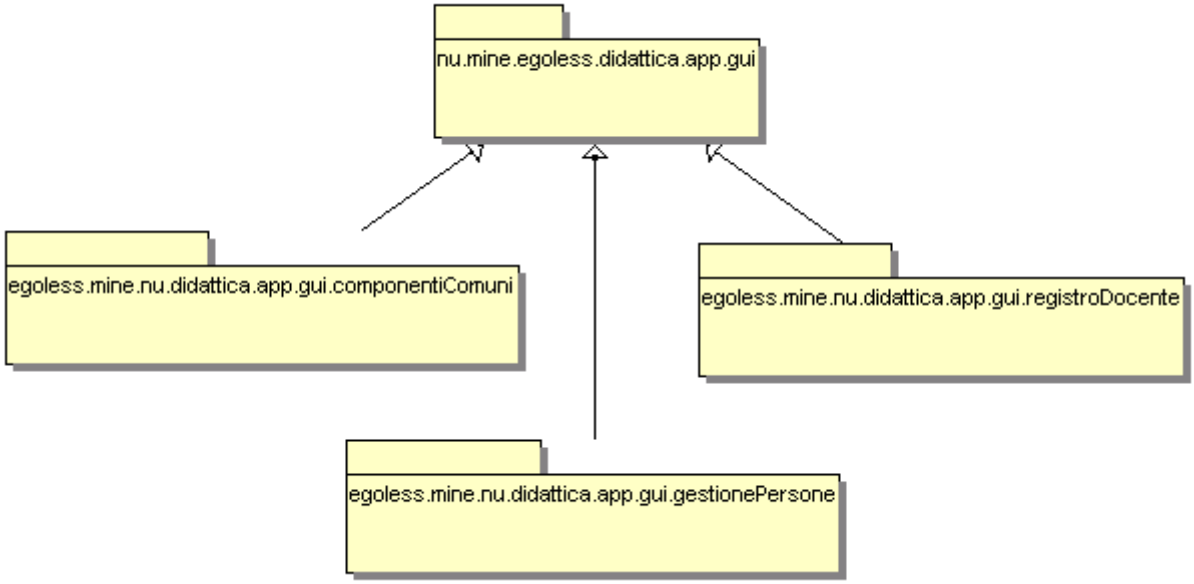
Salvataggio e caricamento degli oggetti

Quando il programmatore vorrà creare una nuova classe lo farà costruendosi il Bean corrispondente usando il costruttore vuoto. Dopo aver settato le proprietà della classe con i metodi di set a disposizione salverà l'oggetto mediante il metodo “salvaSuWS()”, disponibile in ogni Java Bean. In automatico il comando di salvataggio ritornerà un id consistente e valido per quell'oggetto. Il metodo “salvaSuWS()” si occuperà in automatico di aggiornare il campo id dell'oggetto contenuto nel JavaBean. In questo modo se il programmatore ne avrà bisogno può fare il get di tale identificativo.

5.1.5 GUI

Organizzazione classi della GUI per gestione persona

Struttura dei package per la View



Diagrammi di stato per il Model

I seguenti due diagrammi mostrano (1) il ciclo di vita di un PersonaBean che si riferisce ad un entità presente nel database, (2) un PersonaBean che rappresenta una nuova istanza non presente nel database e che eventualmente ivi verrà salvata.
Da precisare che le istanze della classe PersonaBean vengono utilizzate come model per rappresentare i dati mostrati nella view quando l'utente vuole gestire i dati di una persona. Tali diagrammi di stato sono ovviamente validi anche per StudenteBean e InsegnanteBean, che rappresentano due specializzazioni attraverso l'aggiunta di attributi, della classe PersonaBean.

Diagramma (1) : Notiamo che non è possibile modificare un PersonaBean che si riferisca ad un istanza del database prima di avervi caricato i dati. Detto altrimenti, sarebbe possibile non caricarlo, modificarlo e poi salvare i dati nel database (cosa che provocherebbe la sovrascrittura dei vecchi dati presenti) ma l'applicativo prevede di evitare tale situazione per impedire all'utente di creare situazioni non volute. Ciò viene esplicato dal diagramma ove l'unica transizione possibile dallo stato “creato” è verso lo stato “caricato”.
Notiamo altresì che allo stato attuale dell'applicazione non è possibile ricaricare i dati dopo che tale operazione sia stata effettuata all'inizio (cioè non viene implementata per ora l'operazione di restoring)

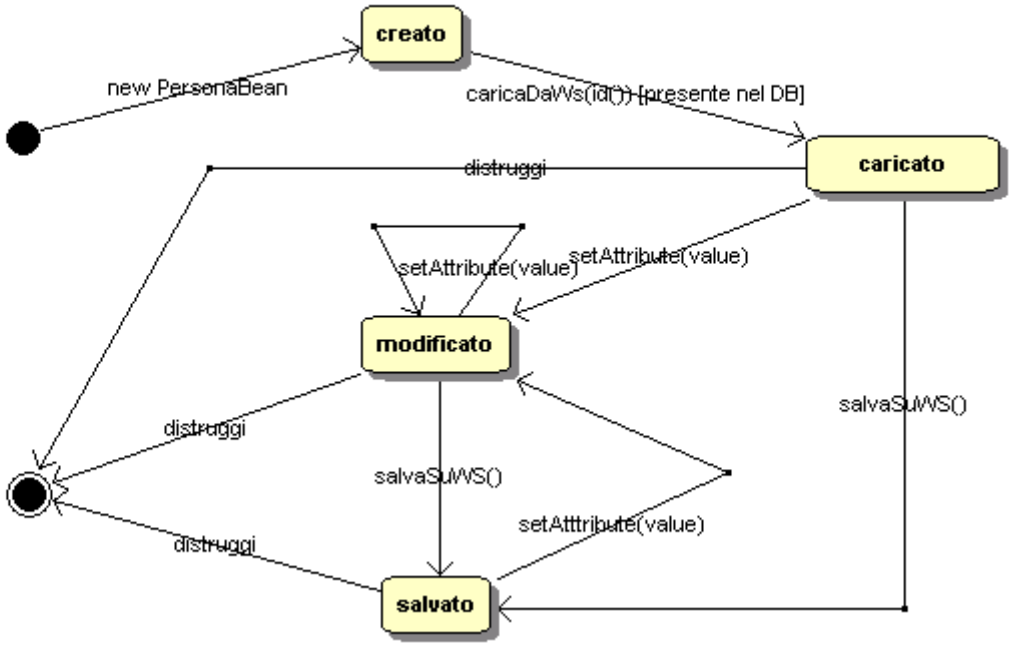
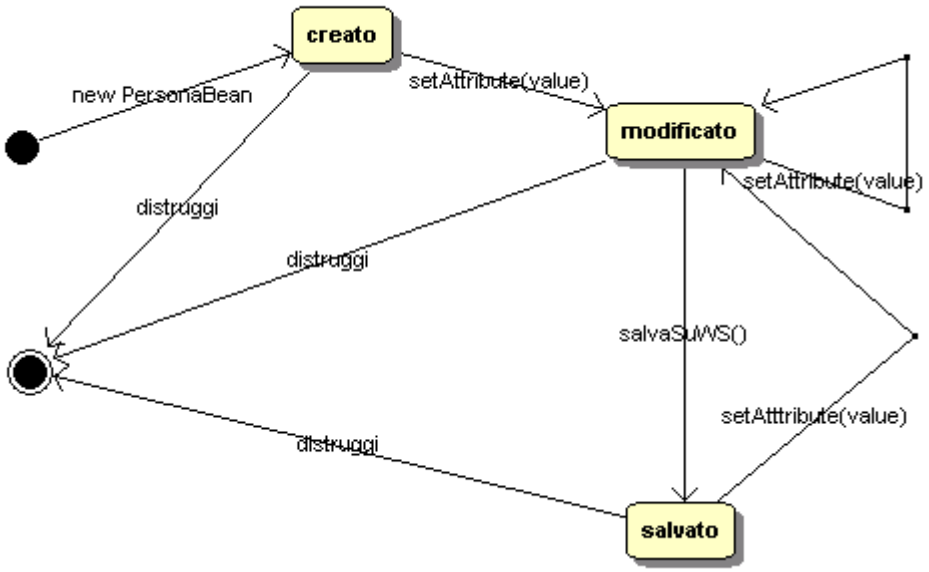
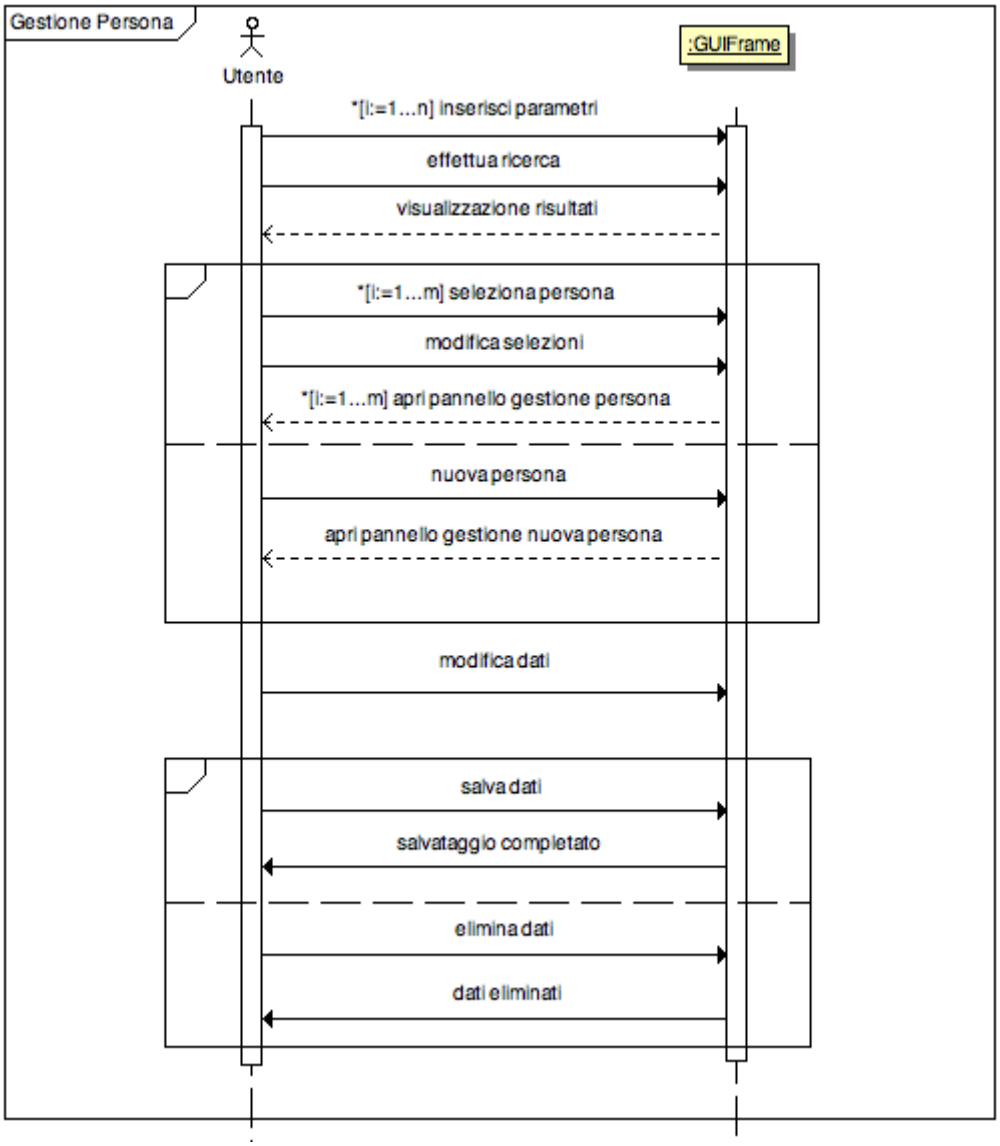


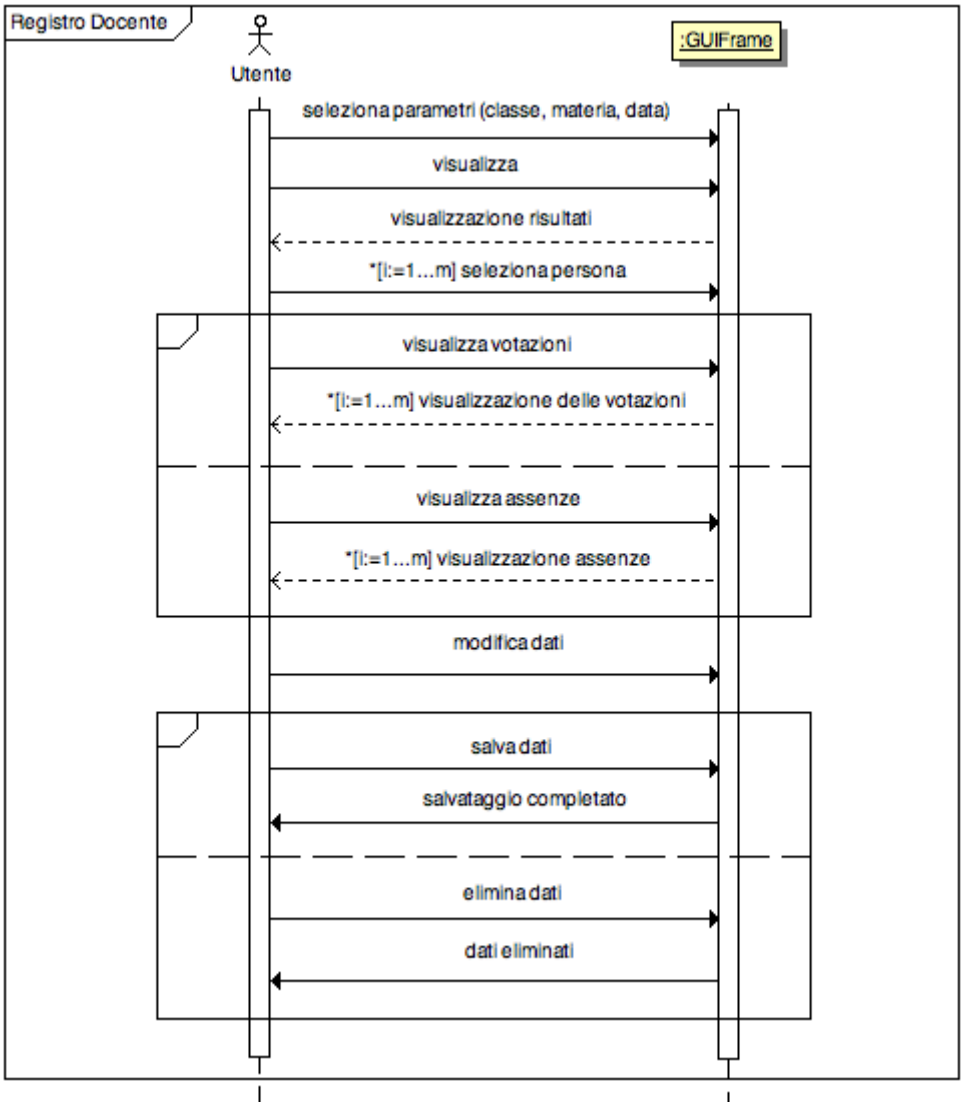
Diagramma (2) : come nel diagramma precedente non viene permesso di ricaricare i dati dopo averli salvati (non avrebbe senso volerli ricaricare se non sono stati salvati)



Diagrammi di sequenza per la GUI

I seguenti diagrammi mostrano l'interazione tra l'utente e la GUI e così esplicano le funzionalità che quest'ultima dovrebbe fornire per quanto riguarda la gestione dei dati delle persone e del registro docente.




Registro Docente
Utente
:GUIFrame
seleziona parametri (classe, materia, data)
visualizza
visualizzazione risultati
*[i:=1...m] seleziona persona
visualizza votazioni
*[i:=1...m] visualizzazione delle votazioni
visualizza assenze
*[i:=1...m] visualizzazione assenze
modifica dati
salva dati
salvataggio completato
elimina dati
dati eliminati

6 Specifica delle componenti dello strato Web Service

6.1 WS-Didattica [CSW-01]

6.1.1 Tipo, obiettivo e funzione

Riferimento alla [ST]

6.1.2 Relazioni d'uso di altre componenti

Riferimento alla [ST]

6.1.3 Considerazioni in merito a i metodi usati //(da collocare)

Funzionamento dei metodi di aggiunta e modifica

Ogni metodo di aggiunta e modifica per un dato oggetto può essere impiegato per manipolare le relazioni dell'oggetto rispetto ad altri. Per esempio, è possibile utilizzare modificaStudente per cambiare la classe a cui uno studente è iscritto. Per fare ciò, per ogni oggetto in relazione è sufficiente specificare il suo id, senza popolarlo interamente. In questo modo è possibile riutilizzare la classe, dato che i campi non popolati non vengono trasferiti tramite Web Service.

Trattamento degli errori

Tutti i metodi di manipolazione restituiscono void. Nel caso avvengano degli errori, ogni metodo lancia un'eccezione opportunamente definita che conterrà un codice e/o un messaggio d'errore che aiuterà l'utente del Web Service a capire la causa dell'errore.

6.1.4 Diagrammi delle classi

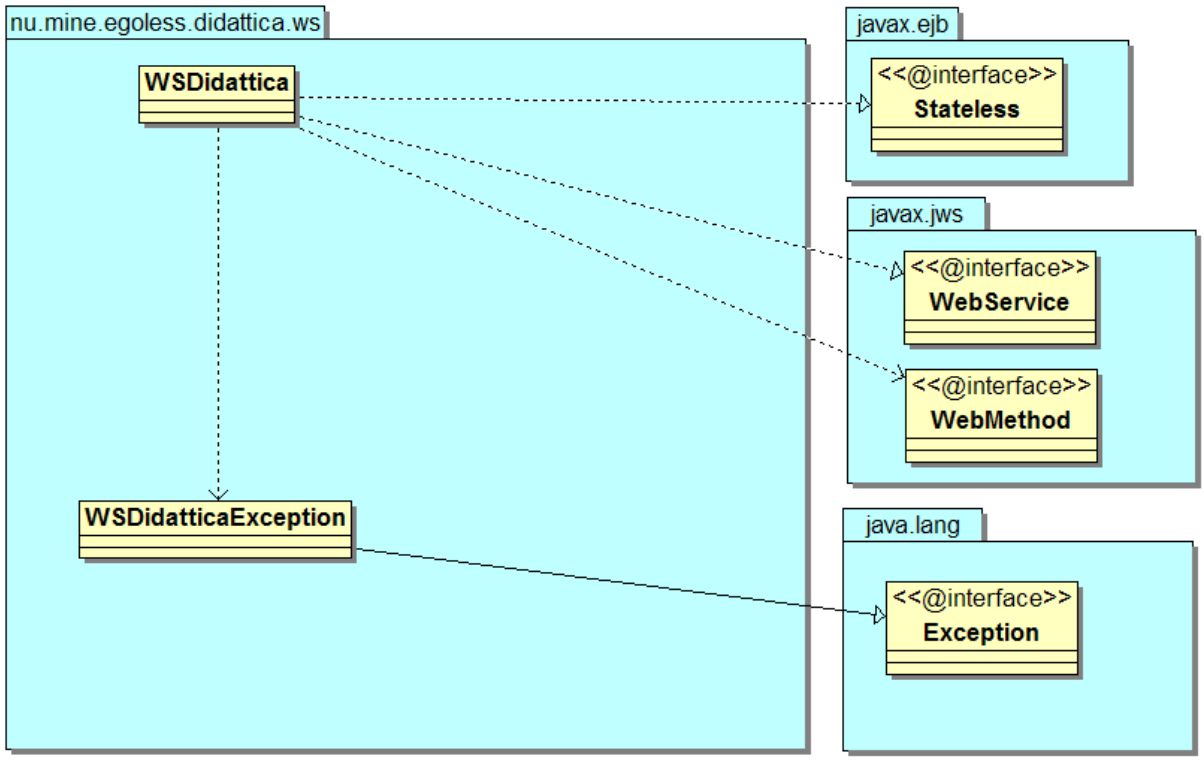


Illustrazione 3: Implementazione del Web Service

7 Tracciamento componenti-moduli

In questo momento è in fase di sperimentazione un tool di tracciamento. Quindi lo sviluppo di questa sezione sarà fatto in seguito.