Sistema di gestione dei Piani di Studi
progettazione con UML

Stefano Guidotti
Sergio Magnani
Leo Orlandini
Stefano Pagani

Indice

Elenco delle figure

Elenco delle tabelle

Parte I

Analisi di sistema

Capitolo 1

Attività preliminari

1.1 Scopo del progetto

Questo documento intende elencare in dettaglio tutti i passi che sono stati compiuti per progettare una applicazione Web che ha come obiettivo la gestione di tutte le funzionalità relative ai Piani di Studio all'interno di un Corso di Studi.

1.2 Analisi grammaticale

Un `Piano di Studi` è un elemento associato ad uno `studente` ed è costituito da un elenco di `esami` che lo studente ha sostenuto durante la sua carriera universitaria e che dovrà sostenere per conseguire la Laurea. L'applicazione che intendiamo progettare deve consentire ad uno studente di presentare un Piano di Studio mediante un `form HTML` che permette di scegliere gli esami da inserire. Un Piano di Studi può essere anche compilato modificando un Piano di Studi esistente. Per poter compilare il Piano di Studi lo studente deve essere in possesso di determinati requisiti. Ogni Piano di Studi deve rispettare una serie di `vincoli`, che vengono stabiliti dal `Consiglio del Corso di Studi`. I professori appartenenti al Consiglio del Corso di Studi si riuniscono di tanto in tanto e, attraverso il sistema, visualizzano ogni Piano di Studio proposto, decidendo se approvarlo o meno. In ogni momento uno studente può visualizzare tramite il sistema il suo Piano di Studio (l'ultimo

approvato tra quelli che ha proposto) e può consultare i vincoli per la compilazione del Piano di Studi. Infine se la `segreteria` lo ritiene opportuno può eliminare tutti i Piani di Studi associati ad un certo studente.

La figura sottostante mostra a grandi linee come intendiamo strutturare l'architettura del sistema che stiamo progettando.

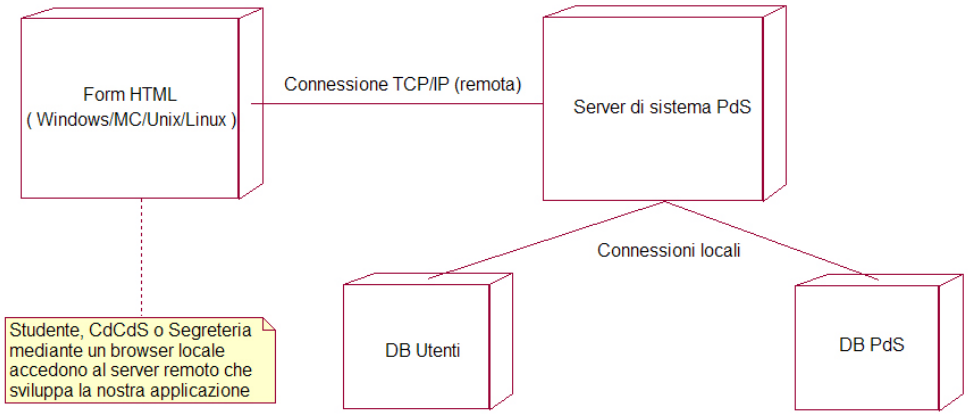


Figura 1.1: diagramma di deployment

In relazione alla figura 1.1 va specificato che le connessioni di tipo locale ai database potrebbero essere connessioni di tipo remoto (JavaRMI) nel caso in cui i database risiedessero su server differenti.

1.3 Elenco oggetti potenziali e individuazione dei ruoli

OGGETTO POTENZIALE	CLASSIFICAZIONE
Studente	entità esterna
CdCdS	ruolo o entità esterna
Segreteria	entità esterna
Esame	cosa
Vincolo	cosa
Form HTML	cosa
Piano di Studi	cosa

Tabella 1.1: elenco oggetti potenziali

Definizioni delle categorie (in accordo con quanto letto in [6]):

entità esterna che produce o consuma informazioni

ruolo svolto dalle persone che interagiscono col sistema

cosa che fa parte del dominio informativo del sistema

Gli oggetti individuati in tabella 1.1 sono i primi candidati a diventare elementi dei futuri diagrammi delle classi.

Per poter essere approvato un PdS deve rispettare i vincoli imposti dal CdCdS. I vincoli per la compilazione previsti dalla nostra applicazione sono tre:

- Un Piano di Studi deve contenere esami per un totale di crediti compresi tra 180 e 210
- Un Piano di Studi deve contenere un certo numero di esami obbligatori
- Un Piano di Studi deve contenere un certo numero di crediti di tipo matematico ed un certo numero di crediti di tipo informatico

Oltre a questo un PdS può essere standard o non standard; un Piano di Studi viene classificato come non standard se contiene esami non appartenenti al Corso di Studi o se il computo dei crediti totali supera le 200 unità. Quando il CdCdS si riunisce i Piani di Studi standard vengono approvati in automatico mentre quelli non standard vengono analizzati singolarmente e i membri del CdCdS decidono se approvarli o meno.

Il sistema permetterà solamente di modificare i parametri di questi vincoli ma non di cambiare la tipologia di questi vincoli

Capitolo 2

Requisiti funzionali e casi d'uso

2.1 Analisi dei requisiti

I requisiti di un sistema possono essere suddivisi in

funzionali : descrivono ad alto livello delle funzionalità del sistema

non funzionali : definiscono le proprietà e i vincoli del sistema

del dominio : descrivono le caratteristiche e le funzionalità imposte dal dominio

I requisiti *funzionali* sono poi divisibili tra quelli specifici degli utenti e quelli del sistema:

- Requisiti degli utenti
  - Uno `studente` deve poter compilare un nuovo Piano di Studi
  - Uno `studente` deve poter modificare il proprio Piano di Studi
  - Uno `studente` deve poter visualizzare il proprio Piano di Studi
  - Uno `studente` deve poter consultare i vincoli imposti dal CdCdS per l'approvazione di un Piano di Studio
  - Il `CdCdS` deve poter decidere se approvare o meno i Piani di Studio proposti dagli studenti
  - Il `CdCdS` deve poter visualizzare i Piani di Studio da approvare

  - Il `CdCdS` deve poter visualizzare i vincoli imposti per l'approvazione dei Piani di Studio
  - Il `CdCdS` deve poter cambiare i vincoli
  - La `segreteria` deve poter eliminare un Piano di Studio

- Requisiti del sistema
  - Deve essere in grado di mantenere l'elenco degli utenti
  - Deve essere in grado di mantenere l'elenco dei Piani di Studi degli studenti
  - Deve essere in grado di verificare automaticamente l'appartenenza allo standard e la coerenza con i vincoli di un Piano di Studi

Per quanto riguarda i requisiti *non funzionali*

- Deve essere garantita la privacy dei dati
- In caso di crash i dati non devono essere corrotti
- Un piano di studi deve essere acceduto in mutua esclusione
- Il sistema deve essere accedibile da più utenti contemporaneamente
- Il sistema deve essere accedibile da un qualunque PC indipendentemente dal sistema operativo
- Il sistema deve essere facilmente utilizzato dagli utenti

Relativamente ai requisiti del *dominio*

- L'accesso al sistema deve essere effettuata via Web
- Ad ogni studente possono essere associati al più due Piani di Studio; uno attivo e uno da approvare
- Ad ogni utente devono essere associate una *user_id* e una *password*

2.2 Individuazione dei casi d'uso

Dopo aver definito le funzionalità offerte dal sistema (i requisiti, appunto) è necessario specificare più in dettaglio il flusso degli eventi che "realizzano" le funzionalità espresse dai requisiti: ogni caso d'uso descrive una modalità di utilizzo del sistema da parte dei vari utenti:

`Consegna di un Piano di Studi`

`Modifica di un Piano di Studi`

`Approvazione di un Piano di Studi`

`Consultazione del Piano di Studi`

`Eliminazione di un Piano di Studi`

`Consultazione dei vincoli di compilazione`

`Modifica dei vincoli di compilazione`

2.3 Descrizione in linguaggio naturale dei casi d'uso

2.3.1 Consegna Piano di Studi

Obiettivo Lo studente compila il modulo del piano di studi via Web.

Priorità Alta

Descrizione **a** lo studente si collega al sito

b lo studente prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d il sistema verifica che lo studente possa compilare il piano di studi

e lo studente compila il modulo del piano di studi

f il sistema controlla se il piano di studi rispetta i vincoli per la compilazione (il numero dei crediti deve essere compreso tra 180 e 210)

g il sistema stabilisce se il piano di studi è standard o meno

h il sistema aggiunge il piano di studi all'elenco dei piani di studi contrassegnandolo come standard o non standard

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

d' lo studente non può compilare un piano di studi; si restituisce un messaggio di errore

f' il piano di studi non rispetta i vincoli; si deve permettere di ricompilare il piano di studi

Ipotesi nessuna

Precondizioni lo studente è in possesso di un ID utente e di una password per autenticarsi; lo studente deve poter compilare il piano di studi (per farlo deve avere almeno 120 crediti)

Postcondizioni il piano di studi rispetta i vincoli per la compilazione

Requisiti soddisfatti 1

2.3.2 Approvazione di un piano di studi non standard

Obiettivo Il Consiglio del Corso di Studi analizza un piano di studi non standard (cioè che contiene esami esterni o che supera i 200 crediti) e decide se approvarlo o meno

Priorità Alta

Descrizione a Il CdCdS si collega al sito

b Il CdCdS prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d Il CdCdS richiede di poter analizzare tutti i piani di studi non standard da approvare

e ogni piano di studi non standard viene mostrato a video

f il CdCdS decide di approvare il piano di studi

g il sistema contrassegna il piano di studi come approvato

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

f' il CdCdS decide di scartare il piano di studi; viene avvisato lo studente che occorre ricompilare il piano di studi

Ipotesi nessuna

Precondizioni il piano di studi analizzato è non standard

Postcondizioni lo studente viene informato se il piano di studi che ha proposto è stato approvato o meno

Requisiti soddisfatti 1

2.3.3 Approvazione di un piano di studi standard

Obiettivo Il sistema approva automaticamente un piano di studi standard (cioè che contiene solo esami del corso di studi)

Priorità Alta

Descrizione a Il CdCdS si collega al sito

b Il CdCdS prova ad autenticarsi

c Il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d Il sistema approva in automatico ogni piano di studi standard

e Il sistema informa lo studente che il suo piano di studi è stato approvato

Alternative

Ipotesi nessuna

Precondizioni nessuna

Postcondizioni lo studente viene informato se il piano di studi che ha proposto è stato approvato

Requisiti soddisfatti

2.3.4 Modifica di un piano di studi

Obiettivo Uno studente modifica via Web un piano di studi proposto da lui stesso in precedenza

Priorità Alta

Descrizione a lo studente si collega al sito

b lo studente prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d il sistema verifica che lo studente possa compilare il piano di studi

e il sistema visualizza a schermo il piano di studi che lo studente intende modificare

f lo studente apporta le modifiche e definisce un nuovo piano di studi

f il sistema controlla se il piano di studi rispetta i vincoli per la compilazione (il numero dei crediti deve essere compreso tra 180 e 210)

g il sistema stabilisce se il piano di studi è standard o meno

h il sistema aggiunge il piano di studi all'elenco dei piani di studi contrassegnandolo come standard o non standard

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

f' il piano di studi non rispetta i vincoli; si deve permettere di ricompilare il piano di studi

Ipotesi nessuna

Precondizioni lo studente è in possesso di un ID utente e di una password per autenticarsi

Postcondizioni il piano di studi rispetta i vincoli per la compilazione

Requisiti soddisfatti 1

2.3.5 Consultazione dei vincoli di compilazione

Obiettivo Vengono stampati a video i vincoli imposti dal CdCdS per la compilazione del piano di studi

Priorità Media

Descrizione a lo studente si collega al sito

b lo studente prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d i vincoli imposti dal CdCdS vengono mostrati a video

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

Ipotesi nessuna

Precondizioni nessuna

Postcondizioni nessuna

2.3.6 Consultazione di un piano di studi

Obiettivo Lo studente consulta via Web il proprio piano di studi

Priorità Bassa

Descrizione a lo studente si collega al sito

b lo studente prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d lo studente sceglie se intende visualizzare il piano di studi approvato o quello pendente

e il sistema visualizza il piano di studi appropriato

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

Ipotesi Esiste almeno un piano di studi (approvato o da approvare) associato allo studente

Precondizioni nessuna

Postcondizioni nessuna

2.3.7 Eliminazione di un piano di studi

Obiettivo La segreteria elimina un piano di studi

Priorità Bassa

Descrizione a lo studente si collega al sito

b lo studente prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d la segreteria richiede la cancellazione del piano di studi

e il sistema elimina il piano di studi

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

Ipotesi nessuna

Precondizioni nessuna

Postcondizioni Il piano di studi non esiste più nel database di sistema

2.3.8 Modifica dei vincoli di compilazione

Obiettivo La segreteria modifica i vincoli per la compilazione

Priorità Alta

Descrizione a la segreteria si collega al sito

b la segreteria prova ad autenticarsi

c il sistema controlla se la coppia <username,password> inserita viene riconosciuta

d la segreteria richiede la visualizzazione dei vincoli di compilazione attuali

e la segreteria modifica i parametri che regolano la definizione dei vincoli

Alternative c' il sistema non trova nessuna corrispondenza della coppia; si deve permettere di riprovare l'autenticazione

Ipotesi nessuna

Precondizioni nessuna

Postcondizioni I nuovi vincoli definiti vengono subito utilizzati per le compilazioni/modifiche di piani di studi

2.4 Diagramma dei casi d'uso

Nel diagramma in figura 2.1 alcune relazioni sono state etichettate con gli stereotipi ≪`extend`≫ ed ≪`include`≫:

Stereotipi ≪`extend`≫

La relazione tra `Modifica` e `Compilazione` indica che il caso d'uso `Modifica` è definito aggiungendo nuove funzionalità al caso d'uso `Compilazione`: in particolare quando si intende modificare un Piano di Studi si utilizza lo stesso form usato per la compilazione con i campi inizializzati con le informazioni presenti nel Piano di Studi che si intende modificare.

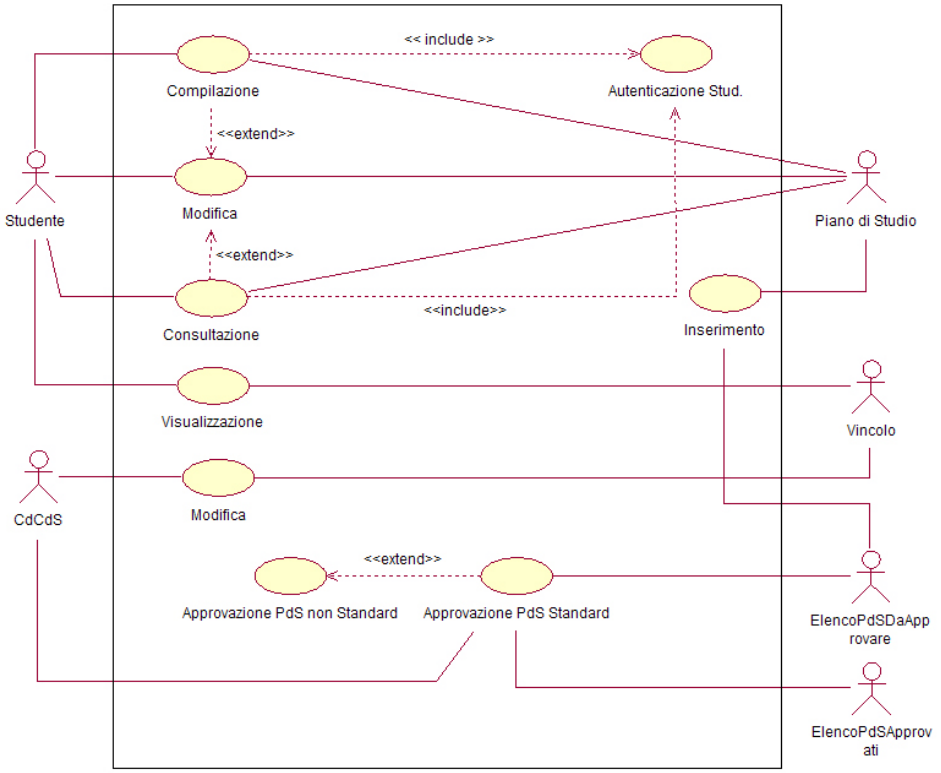


Figura 2.1: diagramma dei casi d'uso

La relazione tra `Modifica` e `Consultazione` indica che il caso d'uso `Modifica` è definito aggiungendo nuove funzionalità al caso d'uso `Consultazione`: in particolare, quando uno studente sceglie di modificare un Piano di Studi, questo viene visualizzato in una pagina (la stessa che viene utilizzata per la colmpilazione) contenente un pulsante che permette di realizzare la modifiche desiderate.

La relazione tra `Approvazione PdS non Standard` e `Approvazione PdS Standard` indica che il caso d'uso `Approvazione PdS non Standard` è definito aggiungendo nuove funzionalità al caso d'uso `Approvazione PdS Standard`: infatti, mentre un Piano di Studi standard viene approvato automaticamente, l'approvazione di un Piano di Studi non standard richiede il parere positivo da parte del Consiglio del Corso di Studi.

Stereotipi ≪include≫

La relazione tra `Compilazione` e `Autenticazione Stud.` indica che il caso d'uso `Compilazione` è definito tramite il riutilizzo al suo interno del caso d'uso `Autenticazione Stud.`: questo perchè quando uno studente chiede di compilare un Piano di Studio deve prima autenticarsi.

Ugualmente la relazione tra `Consultazione` e `Autenticazione Stud.` indica che il caso d'uso `Consultazione` è definito tramite il riutilizzo al suo interno del caso d'uso `Autenticazione Stud.`: questo perchè quando uno studente chiede di consultare il suo Piano di Studio deve prima autenticarsi.

2.5 Diagrammi delle attività

Questo genere di diagrammi nascono come supporto a varie attività, che comprendono l'analisi dei casi d'uso, dei workflow, diagrammi di sequenza e altre ancora. Proprio per questi motivi si è deciso di utilizzarli principalmente al fine di approfondire ed aiutare la comprensione dell'analisi dei requisiti. In più, è stato prodotto un singolo diagramma delle attività da affiancare ad un diagramma di sequenza per aiutarne la comprensione. Il motivo di quest'ultima scelta nasce dalla capacità dei diagrammi di attività di rappresentare le condizioni al contrario delle sequenze. Prima di iniziare l'analisi dei vari casi d'uso utilizzando questa metodologia, si fa presente che non saranno presenti tutti i diagrammi per ogni caso d'uso perchè sarebbe in certi frangenti ripetitivo; quindi saranno rappresentati solo quelli di maggior interesse.

2.5.1 Diagramma delle attività: Compilazione PdS

Lo Studente si connette al servizio tramite il form di autenticazione visualizzato in una finestra del Browser WEB. La fase di autenticazione è identica a quella descritta nel diagramma precedente. Se lo Studente viene autorizzato, può procedere con la richiesta di compilazione del PdS. Nel caso possieda i requisiti necessari, procede con la compilazione del sua piano di studi. Successivamente, il modulo viene controllato affinché rispetti i vincoli in precedenza definiti dal CdCdS. Se il modulo non rispetta i vincoli viene

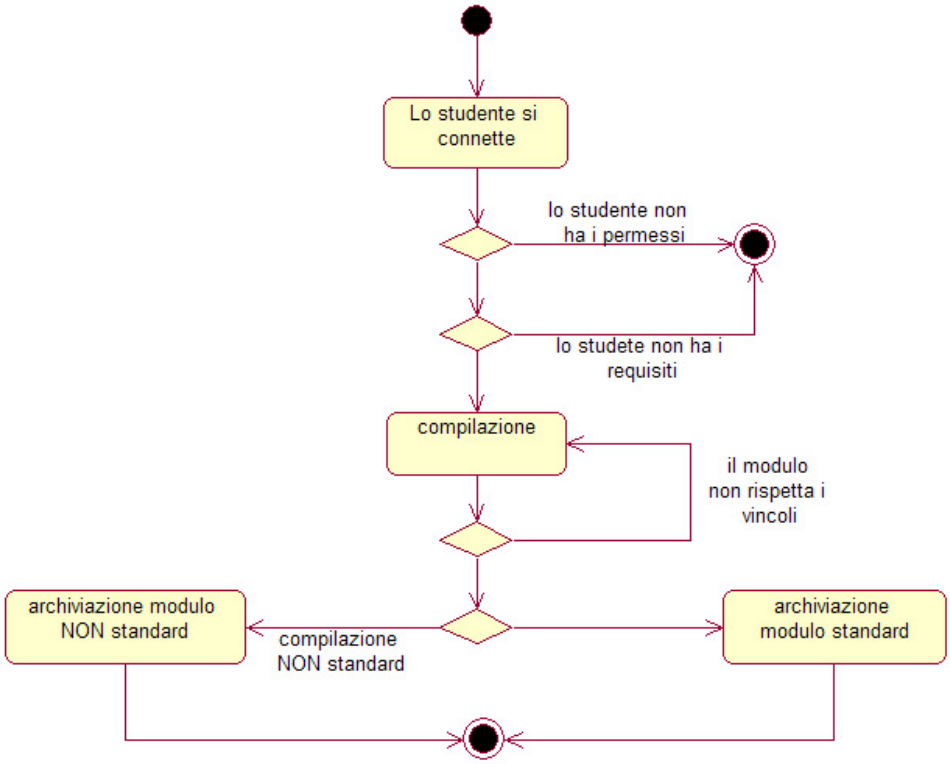


Figura 2.2: diagramma di attività: Compilazione PdS

restituito allo Studente un messaggio che lo avvisa di dover apportare delle modifiche al modulo compilato al fine di renderlo conforme alle richieste. Nel caso in cui il modulo venga accetto dal sistema come conforme ai vincoli, viene successivamente archiviato come standard o non standard a seconda del tipo di compilazione.

2.5.2 `Diagramma delle attività`: Modifica PdS

Nel caso lo Studente voglia modificare un suo precedente PdS, deve prima connettersi al servizio, autentificarsi nel modo già descritto e successivamente scegliere l'opzione di modifica offerta dal sistema che va a ricercare il precedente PdS compilato e archiviato e lo visualizzata sul terminale dello studente come form di compilazione pre-compilato. Lo Studente vi apporta le modifiche volute e se il nuovo modulo rispetta i vincoli, viene archiviato

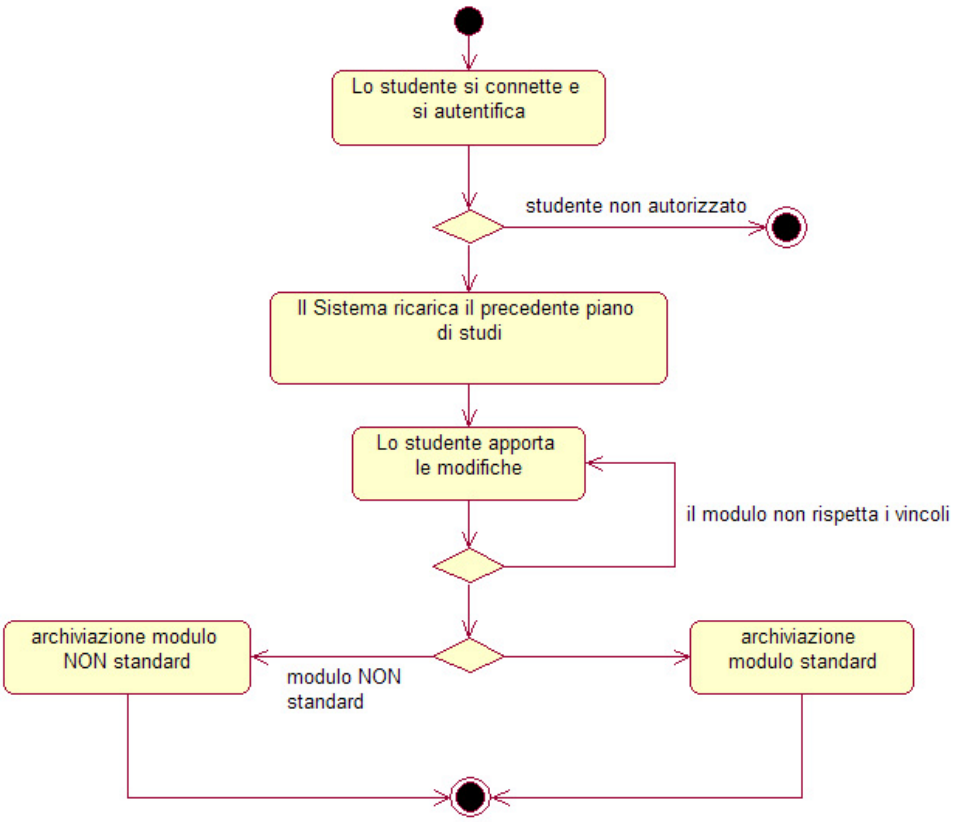


Figura 2.3: diagramma di attività: Modifica PdS

come PdS di tipo standard o non standard a seconda del tipo di esami inseriti e delle opzioni scelte.

2.5.3 Diagramma delle attività: Consultazione PdS

Se lo Studente vuole visualizzare il PdS deve prima procedere con l'operazione di autenticazione già descritta e successivamente sceglie l'opzione di visualizzazione dei PdS. Il sistema prende in input l'ID e il Tipo del PdS dallo Studente e lo va a ricercare nella lista dei PdS; una volta trovato lo restituisce visualizzandolo a schermo.

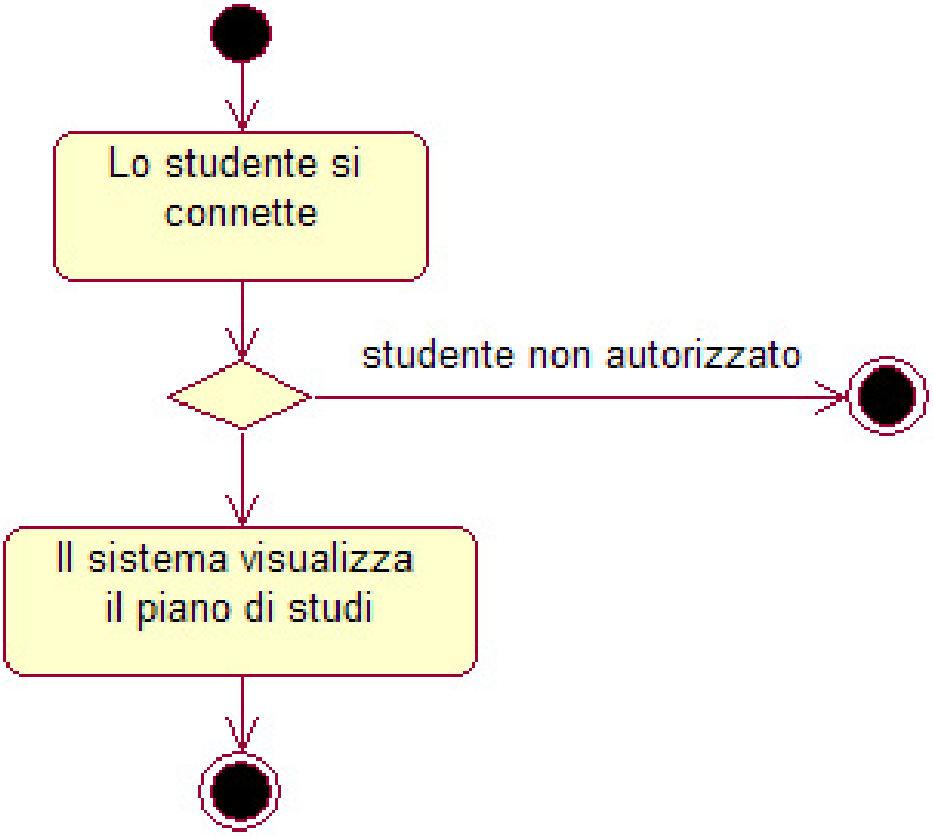


Figura 2.4: diagramma di attività: Consultazione PdS

2.5.4 `Diagramma delle attività`: Approvazione PdS

Il CdCdS si connette al servizio tramite il form di autenticazione visualizzato in una finestra del Browser WEB. A questo punto se l'utente viene riconosciuto come CdCdS, ossia Id e Password vengono riconosciuti come validi, può procedere con la fase di approvazione dei Piani di Studi altrimenti viene inviato un messaggio di errore. La fase di approvazione comincia con la visualizzazione dell'elenco dei Piani di Studi. Per tutti quelli standard, l'approvazione è automatica: il PdS viene archiviato come approvato e viene inviato un messaggio allo studente che lo informa a riguardo. Nel caso invece che il PdS non sia standard, il CdCdS lo visualizza, lo analizza e dopodichè decide se approvarlo o meno. Anche in questo caso viene inviato un

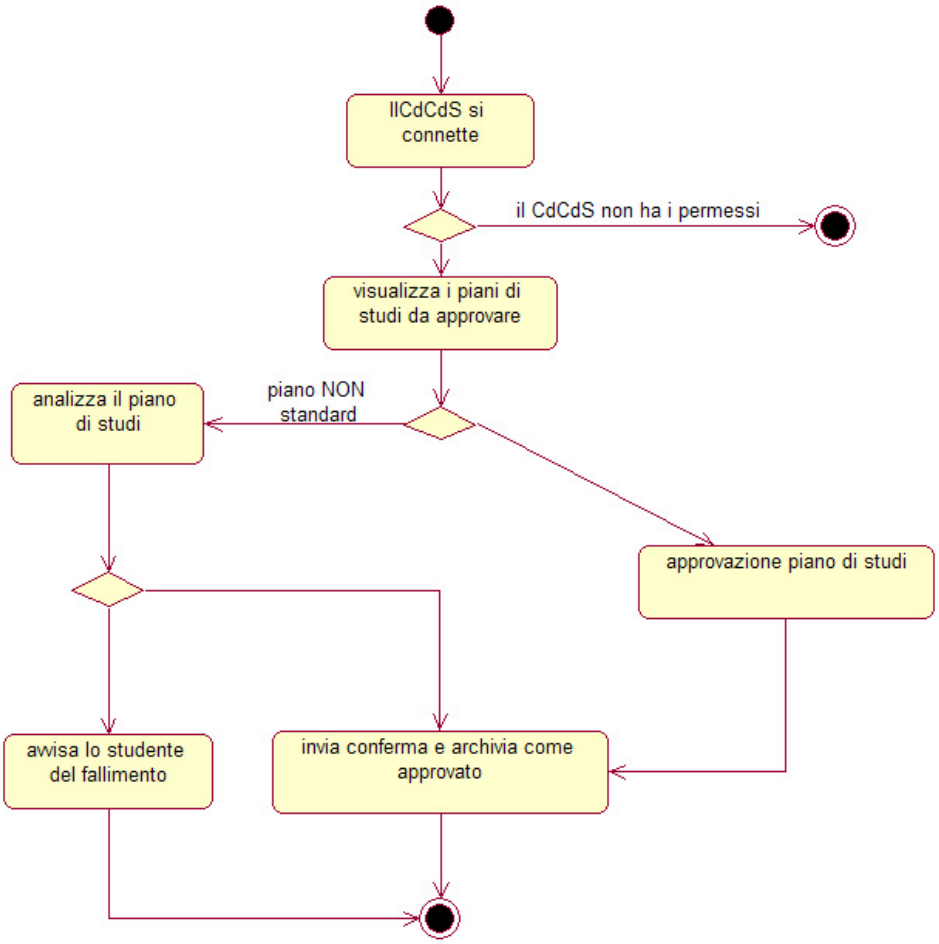


Figura 2.5: diagramma di attività: Approvazione PdS

messaggio allo studente.

Parte II

Progettazione di sistema

Capitolo 3

Gestione di processo

3.1 Scelta del processo di sviluppo

Viene ora descritta la strategia di sviluppo del software che abbiamo scelto di adottare nella progettazione della nostra applicazione.

Dopo aver analizzato i vari processi di sviluppo proposti in letteratura abbiamo optato per il modello a fontana [3]. Questo modello è fortemente orientato alla programmazione ad oggetti estendendo i principi del modello a cascata grazie ad un buon livello di parallelismo tra fasi; in questo modo è possibile evitare il vincolo imposto dal modello a cascata per cui ogni fase inizia solo quando la precedente è terminata, inoltre rappresenta più efficacemente le reali necessità dello sviluppo.

3.2 Analisi delle attività

3.2.1 Analisi del team di sviluppo

Il team di sviluppo è costituito da quattro elementi aventi compiti precisi:

- Stefano Guidotti, Project Manager del progetto. Il compito principale è il coordinamento del gruppo. Tra le operazioni a lui attribuite troviamo la valutazione dei tempi, degli sforzi, dei rischi e dei requisiti. Vista la conoscenza in merito alla programmazione ad oggetti ha il compito di

rappresentare il reparto di implementazione sviluppando in particolare il grafico delle classi a livello implementativo

- Sergio Magnani, web designer. Viste le pregresse conoscenze in materia di sviluppo web il suo compito principale è lo sviluppo e il mantenimento del sito del gruppo. Tra le operazioni attribuitegli troviamo lo sviluppo del diagramma di deployment, dei componenti e di sequenza

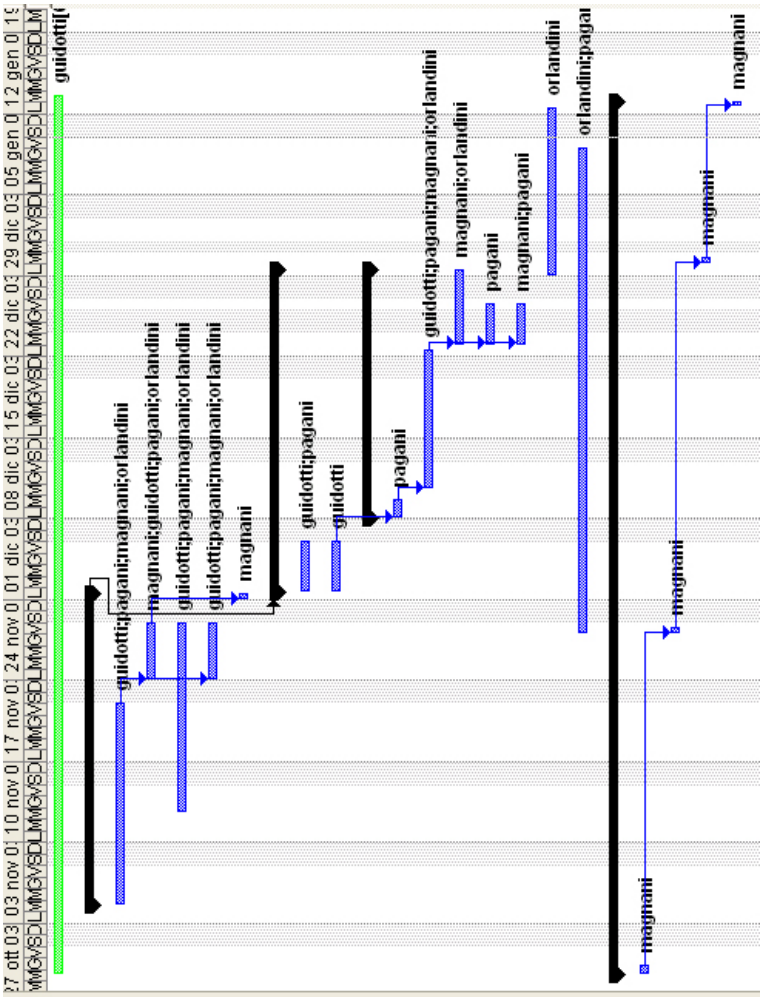
- Leo Orlandini, librarian. In vitrù della sua abillità nel redigere documenti LaTeX il suo compito fondamentale è sviluppare il codice dei documenti prodotti e l'assemblamento della relazione finale. Tra le attività principali troviamo lo sviluppo dei diagrammi di sequenza e delle classi

- Stefano Pagani backup man. Grazie alle conoscenze acquisite riguardo lo sviluppo di software orientato agi oggetti la sua attività principale è la studio del processo *object-oriented*, la produzione dei diagrammi CRC, di sequenza, di stato e dei componenti

Tutto il gruppo ha collaborato all'analisi dei requisiti e nello studio delle classi, nonché ha partecipato all'intera stesura della relazione qui presente

3.2.2 Diagramma di Gantt

Dal diagramma di Gantt in figura 3.1 è possibile vedere come i compiti assegnati ad ognuno dei componenti del gruppo si è realmente evoluto nel tempo. Sono state distinte quattro principali attività che sono l'analisi dei requisiti, la gestione e la progettazione, la produzione dalla documentazione e dalla seguente relazione, ed infine, la gestione del sito web. Queste sono visibili a sinistra in formato testuale in grassetto e rappresentate dalle linee nere che durata la durata di una intera sezione di produzione. Sul grafico è possibili anche vedere a chi è stata assegnata quella parte di attività.



	Nome attività	Durata
1	**project manager**	71 g
2	**analisi dei requisiti**	**25,03 g?**
3	discussione e analisi del dominio	17,53 g
4	analisi grammaticale	6,5 g?
5	produzione scenari	16,88 g
6	produzione casi d'uso	6,5 g?
7	deployment	1 g?
8	**gestione e progettazione**	**25,25 g**
9	studio del processo	5 g
10	stime rischi, tempi, impegno e risors	5 g
11	**produzione dei diagrammi**	**20,25 g**
12	CRC	2 g
13	Classi	12 g
14	sequenza	5 g
15	stato	3 g
16	componenti	3 g
17	**stesura relazione**	14,5 g?
18	**stesura documentazione**	39,25 g
19	**gestione pagina web**	**70,88 g?**
20	creazione	1 g?
21	agg. scenari e casi d'uso	1 g
22	agg. analisi delle classi, seq., stato	1 g
23	agg. relazione finale	0,5 g

Figura 3.1: diagramma di Gantt

3.3 Analisi dei tempi

La valutazione dei tempi di sviluppo è un arte che necessita di esperienza e di analisi pregresse. Stime più accurate possono essere espresse da Project Manager esperti sulla base di progetti passati e della conoscenza riguardo i componenti del team con cui collabora. Il nostro team è costituito da membri che mai in passato hanno partecipato insieme allo sviluppo di un progetto: è quindi molto improbabile riuscire a fare una valutazione delle capacità di collaborazione raggiungibili. Inoltre vista la nostra inesperienza nel confrontarci con la progettazione UML e nello sviluppo di applicazioni Java2 Enterprise Edition è sostanzialmente impossibile fare una stima ragionevolmente precisa delle righe di codice prodotte e dei punti funzione. Nonostante tutto grazie agli strumenti di analisi basati sul metodo di valutazione *COCOMO II*[7] e ipotizzando l'implementazione di 500 righe di codice, è stato possibile stimare uno sforzo di 5.5 mesi. Ovviamente questi dati sono impensabili viste le nostre scadenze, quindi la valutazione corretta dei tempi di sviluppo è data appunto dalle date di consegna del progetto.

3.3.1 Analisi dei punti funzione

Per analizzare la complessità del software da sviluppare abbiamo inoltre utilizzato la tecnica dei *punti funzione*[4].

	valori			coefficienti			parziale
input utente	5	3	0	3	4	6	27
output utente	8	6	0	4	5	7	88
interrogazioni utente	2	7	3	3	4	6	52
file	20	10	0	7	10	15	240
interfacce esterne	0	1	0	5	7	10	7
totale							414

Tabella 3.1: punti funzione

Il risultato delle risposte alle domande di *Arthur* per l'analisi dei valori di aggiustamento della complessitá hanno dato come risultato 32. Quindi

applicando la formula $fp = totale \cdot [0.65 + 0.01 \cdot \sum F_i]$ a partire dai valori riportati nella tabella 3.1 risulta che i punti funzione sono 402. Purtroppo non avendo dati riguardo la produttività del team anche conoscendo il numero totale dei *pf* non é possibile fare una stima del tempo e dello sforzo necessario per lo sviluppo del progetto.

3.3.2 Organizzazione temporale: Milestone

Non avendo dati pregressi su cui basarsi per organizzare in modo dettagliato i tempi e le attività, abbiamo definito una serie di *milestones* da rispettare per poter giungere ad una soluzione positiva del progetto; le operazioni sono conseguenza del processo di sviluppo da noi scelto e rappresentano gli *step* fondamentali. Per poter rispettare le scadenze è stato necessario organizzare il carico di lavoro in modo dinamico, aumentando il volume a ridosso delle date. É stato ipotizzato un solo giorno di anticipo rispetto alla data di consegna: quindi qualora una scadenza non fosse stata rispettata sarebbe stato possibile recuperare un giorno senza pregiudicare il progetto. Nel caso che una data fosse stata violata per più di un paio di giorni allora sarebbe stato necessario forzare i lavori in modo da rispettare almeno la scadenza seguente.

Sono state definite le seguenti *milestone*

1 dicembre : analisi dei requisiti (definizione e diagrammi dei casi d'uso, analisi grammaticale, analisi dei requisiti)

5 dicembre : analisi del processo e stesura delle stime di qualità, dei rischi e dell'impegno

23 dicembre : fine stesura degli schemi delle classi

30 dicembre : fine stesura dei diagrammi di sequenza

9 gennaio : fine stesura relazione

11 gennaio : consegna del prodotto

All'inizio della progettazione abbiamo costruito un diagramma di Gantt basandoci unicamente su queste milestones; in realtà non avrebbe avuto senso

riportare questo diagramma perchè basato solo su previsioni fatte all'inizio del progetto. Abbiamo preferito piuttosto inserire il diagramma di Gantt costruito durante tutta la fase di progettazione: in esso sono riportate le effettive date che hanno scandito tutte le fasi della progettazione.

3.4 Definizione della qualità

Comparando il processo di sviluppo che abbiamo adottato nella fase di progettazione con gli standard definiti nel *Capability Maturity Model*[5], possiamo affermare che il livello da noi conseguito è il secondo. Ovviamente questa stima nasce dall'analisi che abbiamo compiuto rispetto al nostro processo quindi, non potendoci avvalere di enti di certificazione esterna, è puramente ipotetica.

Per supportare questa valutazione possiamo analizzare il nostro comportamento rispetto ai vari requisiti del livello due:

Gestione dei requisiti : abbiamo definito i requisiti necessari per il sistema sulla base dell'analisi del problema assegnatoci. La decisione è stata presa attraverso riunioni dei membri del team di sviluppo

Pianificazione del progetto : abbiamo pianificato le attività utilizzando strumenti software standard quali MS Project. Abbiamo utilizzato metriche di analisi delle attività e dell'impegno, impiegando metodi standard come i grafici di Gantt

Tracking del progetto : abbiamo definito varie *milestone* per la gestione delle scadenze in modo da garantire il successo del progetto, gestendo il carico di lavoro in funzione dell'avvicinarsi delle date e della pianificazione originale

Assicurazione della qualità : abbiamo definito i vincoli di qualità del processo come l'aderenza agli standard descritti in letteratura e il rispetto della pianificazione ipotizzata. Per quel che riguarda il software da implementare, abbiamo imposto come vincoli qualitativi il rispetto dei requisiti

Gestione del software : abbiamo attuato lo scambio dei dati e la gestione delle versioni dei diagrammi (e in futuro del software) avvalendoci di un server ftp per il mantenimento dei file

3.5 Analisi dei rischi

Per garantire il successo del progetto sono stati presi in considerazione i rischi relativi ai tempi di sviluppo, al team, alle risorse a disposizione e ai requisiti di qualità del software.
Per ottenere un'analisi accurata abbiamo compilato una tabella per definire il grado di pericolosità dei vari rischi in funzione del loro grado di probabilità e del possibile danno causato. La probabilità varia da un minimo di 0.1 per i rischi molto improbabili fino a 1 per quelli praticamente certi. Per poter confrontare i rischi tenendo condo di entrambi i valori abbiamo definito il "grado di pericolosità" (gp) come il prodotto della probabilità per il danno

$$gp = p \cdot d$$

Per quel che concerne i rischi relativi al team abbiamo considerato tutti i fattori che avrebbero potuto introdurre ritardi nello sviluppo. Ne segue che i fattori relativi ai tempi di consegna sono correlati a quelli del team e questi sono in parte funzione della complessità stessa del problema e dei requisiti di qualità desiderati. Possiamo quindi affermare che l'aumento della complessità strutturale del problema e l'introduzione di vincoli sulla qualità determinano un maggiore investimento di risorse umane e qualora queste siano strutturalmente limitate inducono un allungamento dei tempi di sviluppo.

Analizzando il grado di pericolosità riportato per ogni rischio rappresentato nella tabella 3.2 abbiamo evidenziato i problemi che avrebbero potuto determinare il fallimento del progetto e abbiamo quindi sviluppato strategie di contenimento dei danni. Dalla tabella 3.2 appare che il problema maggiore da risolvere è la "Complessità dell'ambiente di sviluppo"; in effetti l'impiego di *JBoss* come piattaforma su qui implementare il nostro progetto sarà uno degli ostacoli maggiori poiché indurrà un maggiore sforzo da parte di tutto il team, nonché una probabile dilatazione dei tempi a causa del necessario

Rischio	**Danno**	**Probabilità**	**gp**
Specifiche incomplete	9	0.3	2.7
Incapacità del team	9	0.2	1.8
Errata stima dei tempi	9	0.5	3.6
Perdita di attenzione causa feste	5	0.9	4.5
Indisposizione di elementi del team	2	0.8	1.6
Complessità dell'ambiente di sviluppo	7	0.8	5.6
Basse prestazioni	6	0.3	1.8
Complessità del prodotto	6	0.7	4.2

Tabella 3.2: analisi dei rischi

apprendimento delle procedure base di *deployment*. Quindi è persino possibile che anche la stima di probabilità sull'errata analisi dei tempi tenda ad aumentare.

Per contenere i rischi nel tentativo di limitare al massimo i danni, abbiamo delineato alcune semplici strategie da seguire: per far fronte alle difficoltà presentate da *JBoss* sarà necessario preventivare un periodo di apprendimento della piattaforma di sviluppo di almeno 4 giorni considerando un'attività media di 8-10 ore al giorno. Questo sottrarrà tempo alla fase di implementazione; sarà quindi necessario recuperare questo tempo limitando al massimo la perdita di attenzione dei membri del team ed evitando il più possibile di incorrere in malattie che farebbero saltare questo equilibrio già portato al limite. Un'altra soluzione sarebbe stata anticipare la data d'inizio dei lavori; cosa impossibile poiché l'attività è già in corso.

Qualora le specifiche risultassero incomplete questo causerebbe un notevole danno poiché annullerebbero buona parte dell'operato costringendo l'intero team a rivedere tutto il lavoro e quindi portando ad una sicura violazione della data di consegna; pertanto è assolutamente necessario dedicare

all'analisi del problema tutto il tempo utile per una sicura stesura dei requisiti e delle specifiche onde evitare di dover rimaneggiare il lavoro già fatto.

Capitolo 4

Analisi orientata agli oggetti

4.1 Modelli UML a livello strutturale

4.1.1 `Diagramma delle classi` - livello concettuale

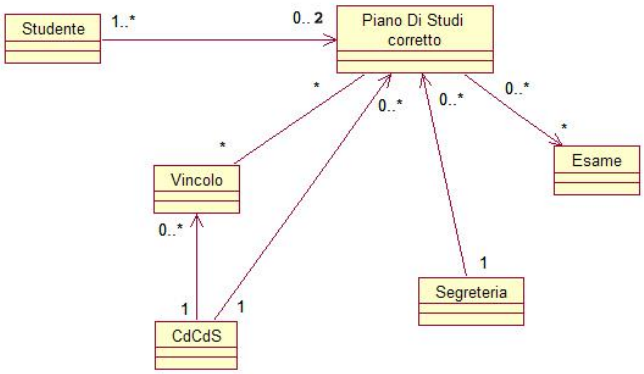


Figura 4.1: diagramma delle classi - livello concettuale

Il diagramma in figura 4.1 descrive le classi del sistema ad un livello molto alto, a livello concettuale[2]; le associazioni sono state individuate a partire dai requisiti identificati in precedenza. Le classi fondamentali a livello concettuale sono: `studente`, `piano di studi`, `vincolo`, `esame`, `segreteria` e `CdCdS`. Le cardinalità scelte evidenziano i seguenti fatti:

- Ad ogni studente vengono associati al più due piani di studi: uno approvato ed uno pendente (se non ne ha definito ancora nessuno zero)
- Ogni piano di studi può essere associato ad uno o più studenti (perchè più studenti possono aver scelto lo stesso piano di studi)
- Ad ogni piano di studi sono associati un certo numero di esami non fisso, quindi non determinabile a priori
- Ogni esame può essere contenuto in zero o più piani di studi
- Ogni piano di studi rispetta un certo numero di vincoli, non calcolabili a priori
- Ogni vincolo viene rispettato da tutti i piani di studi corretti
- Il CdCdS può scegliere di definire zero o più vincoli per l'approvazione dei piani di studi
- Ogni vincolo viene definito dal CdCdS
- Il CdCdS può decidere di approvare o meno zero o più piani di studi
- Ogni piano di studi può essere approvato dal CdCdS
- La segreteria può eliminare zero o più piani di studi
- Ogni piano di studi può essere eliminato dalla segreteria

4.2 Modelli UML a livello comportamentale

4.2.1 Schede CRC

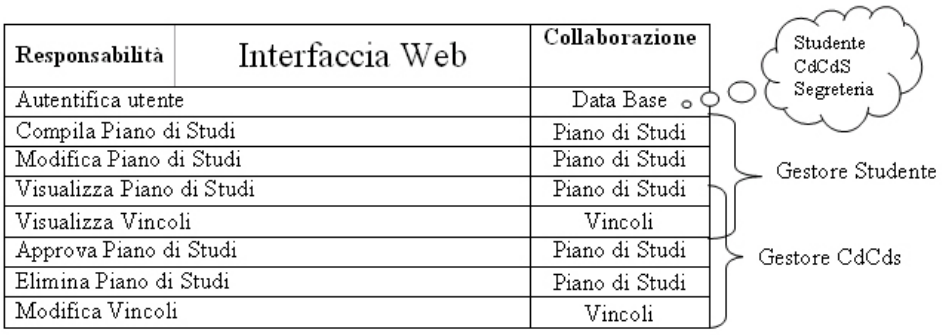


Figura 4.2: scheda CRC: Interfaccia Web

L'`Interfaccia Web` contiene le funzionalità di sistema che gli utenti posso richiedere via Web. Poichè questa scheda contiene troppe responsabilità verrà suddivisa in più interfacce (GestoreCdCdS, GestoreStudente e GestoreSegreteria) a seconda dell'utente. Quindi, come mostrato in figura sarà suddivisa in gestore Studente e gestore CdCdS. Nella figura 4.2 si può inoltre notare che l'operazione di autentificazione collabora con un database: la nuvoletta a fianco mostra a sua volta che il database potrebbe essere suddiviso in tre sottoparti (come per esempio tre java beans).

Responsabilità	Piano di Studi	Collaborazione
Inserisci Piano di Studi in elenco		Elenco PDS
Modifica Piano di Studi da Approvare		Elenco PDS
Cancella Piano di Studi dall'elenco		Elenco PDS
Controlla vincoli del Piano di Studi		Vincoli

Figura 4.3: scheda CRC: PdS

Le operazioni di inserimento, modifica ed eliminazione di un piano di

studi interagiscono con la classe che racchiude tutti i Piani di Studi presenti sul sistema. Sostanzialmente la classe `ElencoPdS` altro non è che una lista di istanze della classe `PdS`. L'ultima operazione relativa ai piani di studi è sicuramente una delle più importanti del sistama e richiede la collaborazione con la classe vincoli: esegue un confronto fra il Pds prodotto dallo studente e i vincoli presenti nel sistema. Questa operazione permette di inserire in elenco solo Piani di Studi formalmente corretti.

Responsabilità	Interfaccia Autentifica	**Collaborazione**
Autentifica utente		Studente CdCdS Segreteria
Passa modalità utente		Gestore Stud Gestore CdCds Gestore Segr

Figura 4.4: scheda CRC: Interfaccia Web di autenticazione

La classe `Interfaccia Autentifica` deriva dalla divisione di `Interfaccia Web`; la responsabilità AutentificaUtente permette l'accesso al sistema tramite password. La seconda responsabilità gestisce l'accesso all'interfaccia opportuna in base alla tipologia di utente.

Responsabilità	Gestore Studente	**Collaborazione**
Visualizza Vincoli		Vincoli
Visualizza Piano di Studi		Elenco PdS
Compila Piano di Studi		PdS Elenco PdS
Modifica Piano di Studi		Elenco PdS

Figura 4.5: scheda CRC: Interfaccia Web Studente

L'operazione VisualizzaVincoli permette agli studenti di consultare i vincoli mediante la collaborazione con la la classe Vicoli in cui sono espressi. La

responsabilità CompilaPianoDiStudi collabora sia con la classe PdS (perchè deve essere creato un nuovo oggetto dei quel tipo) sia con ElencoPdS(perchè deve essere poi inserito in questo elenco). Invece le operazioni VisualizzaPianoDiStudi e ModificaPianoDiStudi collaborano con la sola classe ElencoPdS perchè cercano il Piano di Studi in esame per visualizzarlo o per passarlo alla compilazione in caso di modifica.

Responsabilità	Gestore CdCdS	**Collaborazione**
Visualizza vincoli		Vincoli
Modifica vincoli		Vincoli
Visualizza Piano di Studi		Elenco PdS
Approva Piano di Studi		Elenco PdS

Figura 4.6: scheda CRC: Interfaccia Web CdCdS

VisualizzaVincoli e VisualizzaPianoDiStudi sono uguali alle responsabilità omonime di `GestoreStudente`; a questo punto si potrebbe pensare di riutizzare il codice e di creare una classe astratta implementata da `GestoreStudente` e `GestoreCdCdS` che racchiude queste operazioni. L'operazione ModificaVincoli collabora solamente con la classe Vincoli. Merita invece più attenzione la responsabiltità ApprovaPianoDiStudi che collabora con ElencoPdS per reperire il Piano di Studi che sarà valutato dal CdCdS: successivamente nel caso sia corretto verrà marcato approvato altrimenti verrà cancellato della lista in caso negativo.

Responsabilità	Elenco PdS	**Collaborazione**
Inserisci Piano di Studi in elenco		Piano di Studi
Cancella Piano di Studi dall'elenco		Piano di Studi
Visualizza Piano di Studi		Piano di Studi

Figura 4.7: scheda CRC: Elenco PdS

`ElencoPdS` implementa tre semplici metodi di inserimento, eliminazione e visualizzazione di elementi PdS in ElencoPdS.

Responsabilità	Vincoli	**Collaborazione**
Inserisci vincoli		Interfaccia web
Modifica vincoli		Interfaccia web
Controlla Piano di studi		Piano di Studi

Figura 4.8: scheda CRC: Vincoli

Le operazioni di `Vincoli` permettono di inserire e modificare i vincoli che i Piani di Studi devono rispettare; entrambe devono collaborare con GestoreCdCdS, in quanto il CdCdS è l'unico utente che ha i diritti per operare sugli attributi della classe Vincoli.

4.2.2 `Diagramma delle classi` - livello di specifica

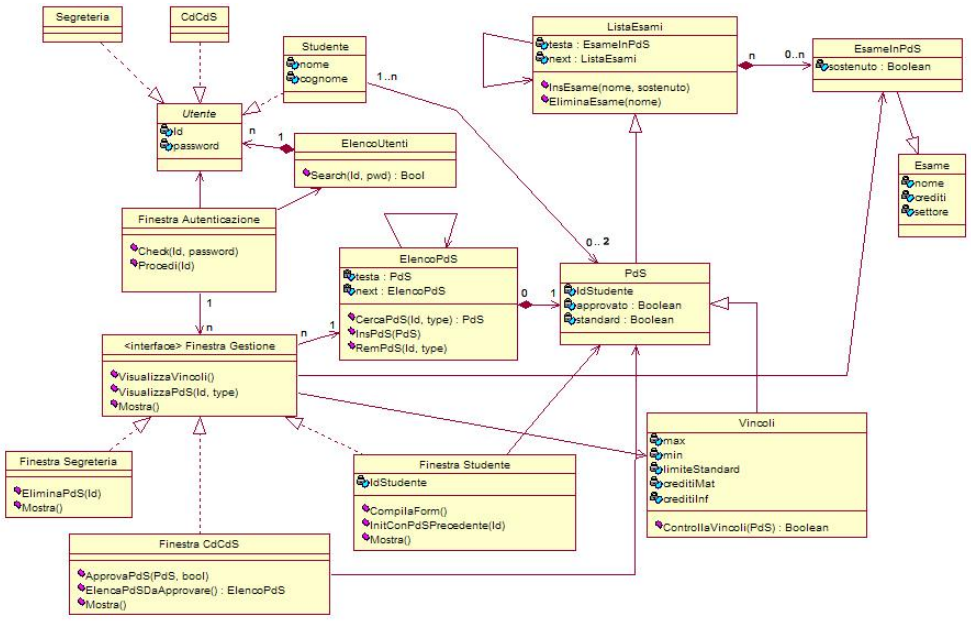


Figura 4.9: diagramma delle classi - livello di specifica

In figura 4.9 è mostrato il diagramma delle classi a livello di interfaccia/specifica. La scelta dei nomi, degli attributi e dei metodi di tutte le classi rispetta la scelta stilistica che prevede nomi che iniziano con una lettera minuscola per gli attributi e nomi che iniziano con una maiuscola per i metodi.

Descrizione delle classi, degli attributi e dei metodi

- ***`Utente`*** *Classe astratta che rappresenta un utente generico*

  `Id` stringa che identifica l'uente in ambito universitario

  `password` password usata (insieme all'Id) dall' utente per l'autenticazione

- **`Studente`** *Classe che implementa la classe* `Utente`*: rappresenta uno studente*

`nome` nome dello studente

`cognome` cognome dello studente

- `Segreteria` *Classe che implementa la classe* `Utente`*: rappresenta la segreteria*
- `CdCdS` *Classe che implementa la classe* `Utente`*: rappresenta il CdCdS*
- `ElencoUtenti` *Lista degli utenti del sistema*

  `Search(id,password):Bool` Confronta la coppia <Id,password> con quelle contenute nella lista. Ritorna `true` se c'è corrispondenza

- `FinestraAutenticazione` *Rappresenta la finestra che permette ad un qualsiasi utente di autenticarsi*

  `Check(Id,password)` Controlla se la copppia <Id,password> trova una corrispondenza in `ElencoUtenti`

  `Procedi(Id)` Apre la finestra opportuna in base alla categoria (studente,segreteria,CdCdS) dell'utente (la tipologia di utente viene individuata da `Id`)

- ***`Finestra Gestione`*** *Interfaccia per le tre finestre con cui gli utenti possono interagire con il sistema: FinestraSegreteria, FinestraCdCdS e FinestraStudente*

  `VisualizzaVincoli(id,type)` Mostra a schermo i vincoli imposti dal CdCdS per l'approvazione di un PdS

  `VisualizzaPdS()` Visualizza a schermo un Piano di Studi associato allo studente identificato da Id; type specifica se va mostrato il PdS approvato o quello da approvare

  `Mostra()` Mostra la pagina specifica in base all'utente che si è autenticato

- `Finestra Segreteria` *Finestra con cui la Segreteria interagisce con il sistema*

`EliminPdS(Id)` Elimina tutti Piani di Studi associati allo studente identificato da `Id`

- **`Finestra CdCdS`** *Finestra con cui il CdCdS interagisce con il sistema*

  `ApprovaPdS(pds,bool)` Approva o scarta il `PdS` passato come parametro (*pds*); `bool` indica se il PdS va approvato (1) o scartato (0). Se viene approvato elimina il vecchio PdS approvato per lo studente considerato. Se invece viene scartato *pds* viene eliminato da ElencoPdS.

  `ElencaPdSDaApprovare():ElencoPdS` Ritorna una lista di PdS da approvare

- **`Finestra Studente`** *Finestra con cui uno studente interagisce con il sistema*

  `IdStudente` Id dello studente che sta utilizzando l'istanza di `FinestraStudente`

  `CompilaForm()` Permette ad uno studente di scegliere gli esami da inserire in un PdS: la scelta viene realizzata tramite una sequenza di inserimenti-eliminazioni di esami

  `InitConPdSPrecedente(Id,type)` Inizializza il form con le informazioni contenute in un altro PdS: il PdS con cui inizializzare il form viene restituito da una chiamata a CercaPdS() con parametri `Id` e `type`

- **`ListaEsami`** *Classe per la gestione di una lista di esami*

  `InsEsame(nome,sostenuto` Inserisce un esame nella lista `sostenuto` indica se l'esame è già stato sostenuto (`1`) o meno (`2`)

  `EliminaEsame(nome)` Elimina un esame dall'elenco

- ***`PdS`*** *Classe che rappresenta l'entità Piano di Studi; è la lista di tutti gli esami che lo studente associato al PdS ha già sostenuto o sosterrà*

  `IdStudente` Id dello studente cui il PdS è associato

`approvato` valore booleano che indica se il piano di studio è già stato approvato o se è in attesa di essere valutato

`standard` valore booleano che indica se il piano di studio è standard o meno

- `ElencoPds` *Classe che rappresenta un elenco di PdS*

  `testa:PdS` Punta alla testa della lista

  `next:ElencoPdS` Puntatore all'elenco degli elementi successivi; serve per scorrere l'elenco

  `CercaPds(id,type):PdS` Cerca e ritorna un PdS in base all'id e al tipo(indica se il PdS da cercare è approvato o meno)

  `InsPds(PdS)` Inserisce un PdS nella lista

  `RemPds(id,type)` Rimuove un PdS (identificato da Id e type) dall'elenco

- `Esame` *Rappresenta ogni singolo esame appartenente al CdS*

  `nome` nome dell'esame

  `crediti` numero di crediti dell'esame

  `settore` settore didattico di appartenenza dell'esame

- `EsameInPdS` *Rappresenta un esame inserito in una lista di esami: estende la classe Esame*

  `sostenuto` indica se l'esame è già stato superato o meno

4.3 Diagrammi di stato

L'utilizzo di questo genere di diagrammi è argomento di discussione, in quanto non tutti gli sviluppatori li ritengono utili. Al contrario, noi siamo convinti che l'utilizzo di questi contribuisca a rendere più chiaro l'evoluzione delle classi cruciali del sistema. Quando parliamo di cambiamento di stato intediamo la variazione degli attributi interni delle varie classi in esame e l'uso che ne viene fatto. I diagrammi che presenteremo sono due: il primo per chiarire come un piano di studi si evolve a seconda degli eventi; il secondo mostra più in dettaglio il funzionamento delle varie interfacce utente.

4.3.1 Diagramma di stato: PdS

I diagrammi di stato (vedi figura 4.10) sono utili per capire a fondo lo scopo e il comportamento delle classi principali: ha senso quindi costruirne solo per le classi che sviluppano le funzionalità chiave del sistema. Di norma si tende ad accostare questi diagramma a quelli di sequenza per una migliore comprensione dell'utente.

Descrizione degli stati e delle transizioni

- `Inizio`
  Quando lo studente richiede di poter compilare il Piano Di Studi grazie all'interfaccia Web (generata e gestita dalla classe `FinestraStudente`) avviene una chiamata al metodo `compilaForm()`

- `Attendi Compilazione PdS`
  Intervallo di tempo durante il quale il sistema rimane in attesa che lo studente compili il modulo senza compiere nessuna operazione; quando la compilazione termina si uscirà da questo stato per passare ad un'altro, ma prima saranno compiute le due attività contrassegnate con `exit`. La prima (`PdS()`, metodo costruttore) si occupa di creare una nuova istanza (pds) della classe PdS mentre la seconda (`ControllaVincoli(pds)`) controlla se il PdS appena creato rispetta i vincoli imposti dal CdCdS.

  Questo passaggio può essere compreso meglio oservando diagramma di sequenza di `Compilazione PdS` (figura 4.14) dove la classe Finestra-

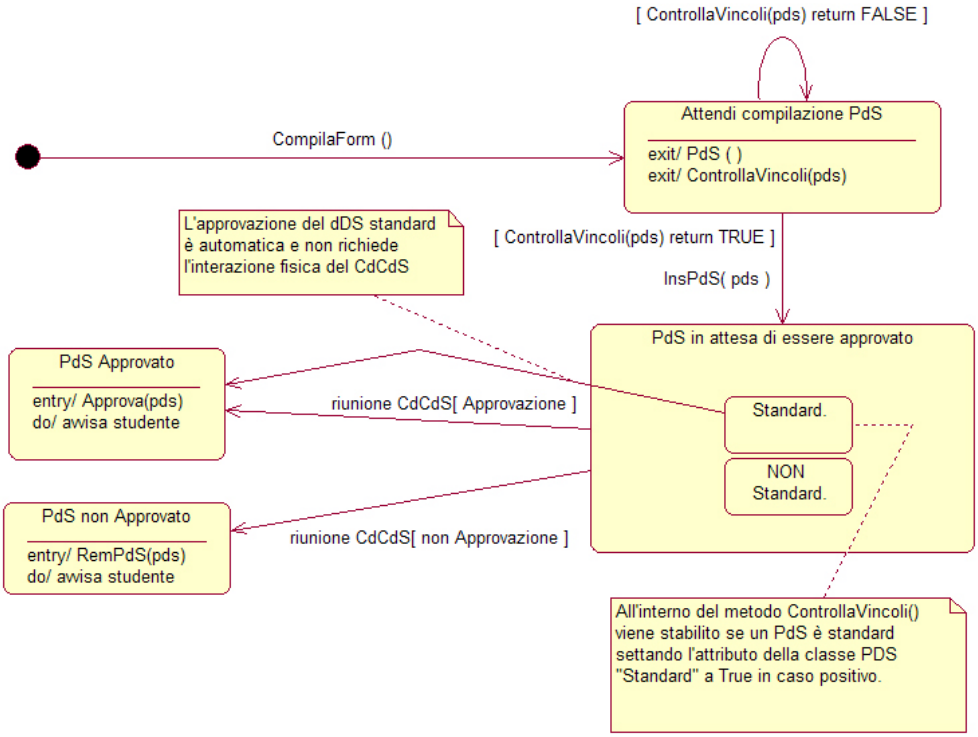


Figura 4.10: diagramma di stato: PdS

Studente crea una istanza di PDS richiamando il costruttore passando i parametri presi in input dal Form HTML di compilazione.

- [*ControllaVincoli(PdS) return TRUE*] *InsPdS(pds)*
  Se il controllo dei vincoli ritorna un valore positivo pds viene inserito in ElencoPdS e si passa allo stato `PdS in attesa di essere approvato`.

- [*ControllaVincoli(PdS)*] *return FALSE*]
  Al contrario se i vincoli non sono soddisfatti si ritorna nello stesso stato `Attesa Compilazione PdS`.

- `PdS in attesa di essere approvato`
  La caratteristica fondamentale di questo stato sta nei due sotto-stati mostrati. Infatti il sistema si comporta in maniera diversa con i PdS standard rispetto a quelli non standard. Quando il CdCdS si riuisce

tutti i PdS standard vengono approvati in automatico, mentre quelli non standard vengono visualizzati grazie a `FinestraCdCdS` per permettere al CdCds di analizzare nel dettaglio il PdS prima di dare l'ok alla approvazione.

- *riunone CdCdS* [*Approvazione*]

- *riunone CdCdS* [*Non approvazione*]
  Quando il CdCdS si riunisce vengono esaminati i PdS non standard: i PdS approvati transiscono nello stato `PdS approvato` mentre quelli scartati vanno nello stato `PdS non approvato`.

- `PdS approvato`
  Quando giunge in questo stato stato un PdS viene approvato con la chiamata al metodo `Approva`. Lo studente viene poi avvisato dell'approvazione del suo PdS.

- `PdS non approvato`
  Quando giunge in questo stato un PdS viene rimosso da `ElencoPdS`. Il sistema avvisa lo studente che il suo Piano di Studi è stato respinto.

4.3.2 `Diagramma di stato`: Interfacce di sistema

Quando un utente si collega all'indirizzo relativo al nostro servizio, il sistema sarà fermo nello stato `Interfaccia Autentifica`. Quando verrà inserito il nome utente e password, nel caso queste risultino valide, si passerà allo stato `Finestra Gestione`; questo rappresenta una classe astratta che è suddivisa in tre diverse estensioni di Finestra Gestione, una per ogni tipo di utente. L'evento che porta a questi tre sottostati è il medesimo; quello che cambia è la guardia, che specifica la tipologia dell'utente.

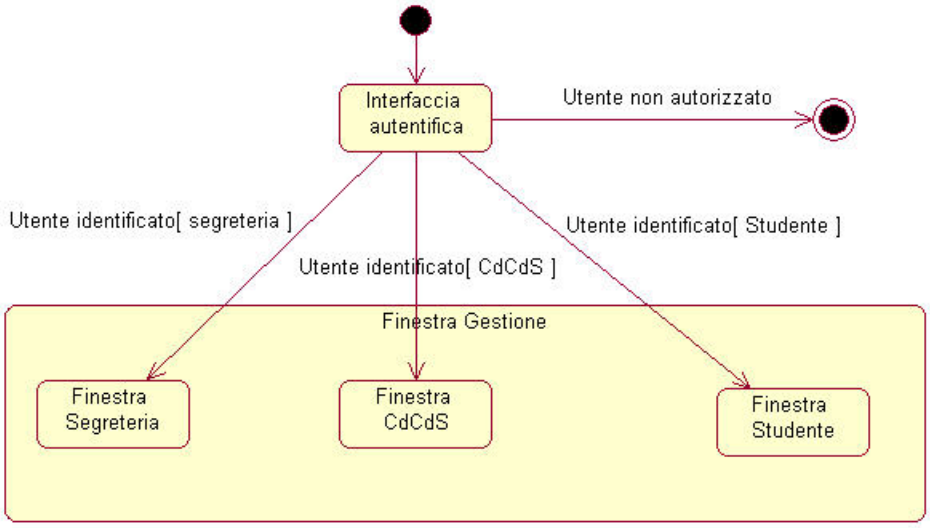


Figura 4.11: diagramma di stato: Interfacce di sistema

4.4 Diagrammi di interazione

Dovendo decidere se inserire nella documentazione il diagramma di collaborazione o quello di sequenza abbiamo preferito quelli di sequenza perchè pongono più enfasi sulla successione esatta degli eventi; in questo modo è più facile idividuare l'esatto l'ordine delle chiamate di procedure. Nonostante questo vantaggio, i diagrammi di sequenza sono limitati quando bisogna rappresentare delle alternative e per questo abbiamo deciso di allegare anche alcune schede CRC che permettono di comprendere al meglio il comportamento di ogni classe.

4.4.1 Diagramma di sequenza: Autenticazione Utente

Come mostrato nella figura 4.12, un qualunque utente (studente, segreteria o CdCdS) che vuole autenticarsi utilizza la `FinestraAutenticazione`. Il metodo `Check()` viene invocato quando l'utente richiede l'autenticazione dopo aver immesso il suo Id e la sua password. La FinestraAutenticazione invoca quindi il metodo `Search()` che prende come parametri l'Id e la password immessi dall'utente e restituisce un valore booleano. Se Search() restituisce 1 (utente in elenco) viene invocato il metodo `Procedi()` che al suo interno contiene una chiamata al metodo Mostra() della opportuna finestra di gestione (GestioneSegreteria, GestioneCdCdS. GestioneStudente)

4.4.2 Diagramma di sequenza: Visualizzazione PdS

Nella figura 4.13 viene rappresentato il diagramma di sequenza della visualizzazione di un PdS; quando uno studente vuole visionare un Piano di Studi questo viene mostrato in FinestraStudente attraverso `Mostra()`. L'utente ha la possibilità di scegliere il Piano di Studi tra quelli a lui associati (può averne due: uno approvato ed uno pendente); questo verrà mostrato a video attraverso il metodo `VisualizzaPdS()`: il secondo parametro indica se deve essere mostrato il PdS approvato o quello pendente.

NB: All'interno di `VisualizzaPdS()` la ricerca del Piano di Studi da visualizzare viene realizzata con il metodo `ElencoPdS.CercaPdS()`.

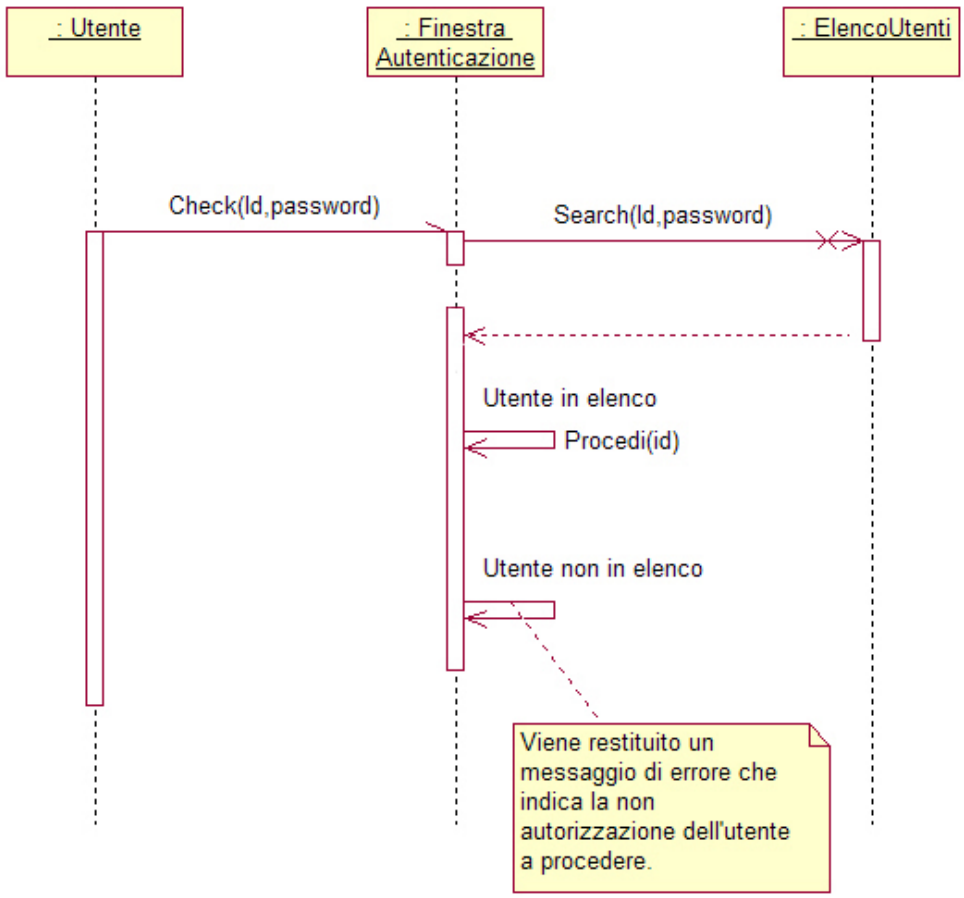


Figura 4.12: diagramma di sequenza: Autenticazione utente

4.4.3 `Diagrammi di sequenza`: Compilazione PdS

Quando uno studente vuole proporre un Piano di Studi può definirne uno da zero o modificare uno dei suoi Piani di Studi (sia approvato che pendente). Entrambe queste operazioni vengono realizzate tramite una stessa finestra (`FinestraStudente`).

Compilazione di un nuovo PdS

Come mostrato nella figura 4.14, quando uno studente sceglie di definire un nuovo Piano di Studi viene mostrata le finestra di compilazione (`Mostra()`) e lo studente può scegliere gli esami da inserire nel Piano di Studi (`CompilaForm()`).

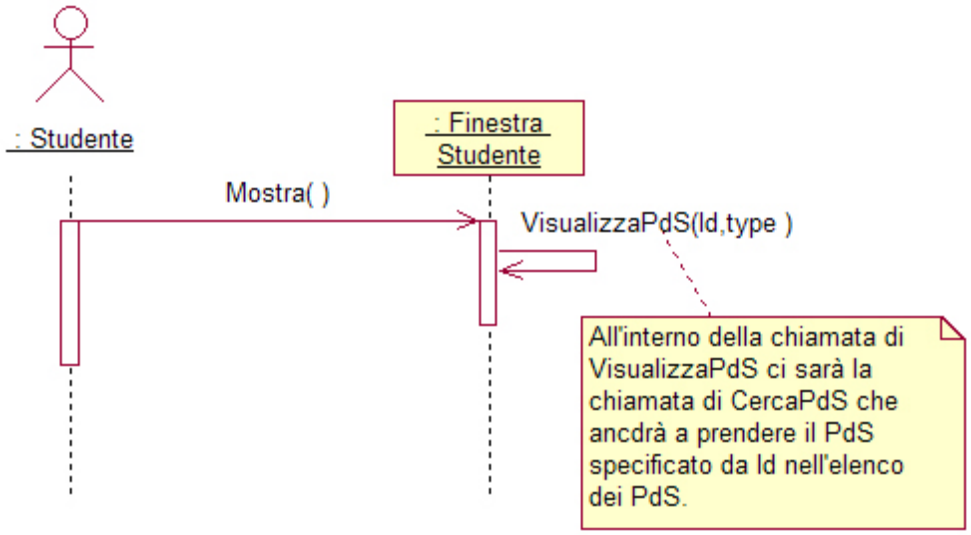


Figura 4.13: diagramma di sequenza: Visualizzazione PdS

Quando lo studente ha finito di scegliere gli esami viene richiamato il costruttore `PdS()`: questo crea una nuova istanza di PdS. `ControllaVincoli()` verifica se gli esami inseriti nel nuovo Piano di Studi rispettano i vincoli: in caso positivo il nuovo elemento creato viene inserito in ElencoPdS (`InsPdS()`). Se invece i vincoli non sono soddisfatti il nuovo PdS creato viene distrutto.

Modifica di un PdS esistente

Il diagramma di sequenza della modifica in figura 4.15 differisce da quello della compilazione per il fatto che prima della chiamata a `CompilaForm()` il form stesso viene inizializzato con gli esami presenti nel Piano di Studi che si vuole modificare. Questo viene realizzato con una chiamata a `InitConPdSPrecedente`; il secondo parametro passato a questo metodo permette allo studente di scegliere se modificare il proprio Piano di Studi approvato o quello pendente.

4.4.4 `Diagrammi di sequenza:` Approvazione PdS

Quando i componenti del CdCdS si riuniscono tutti i Piani di Studi standard vengono approvati in automatico mentre ognuno dei Piani di Studi non

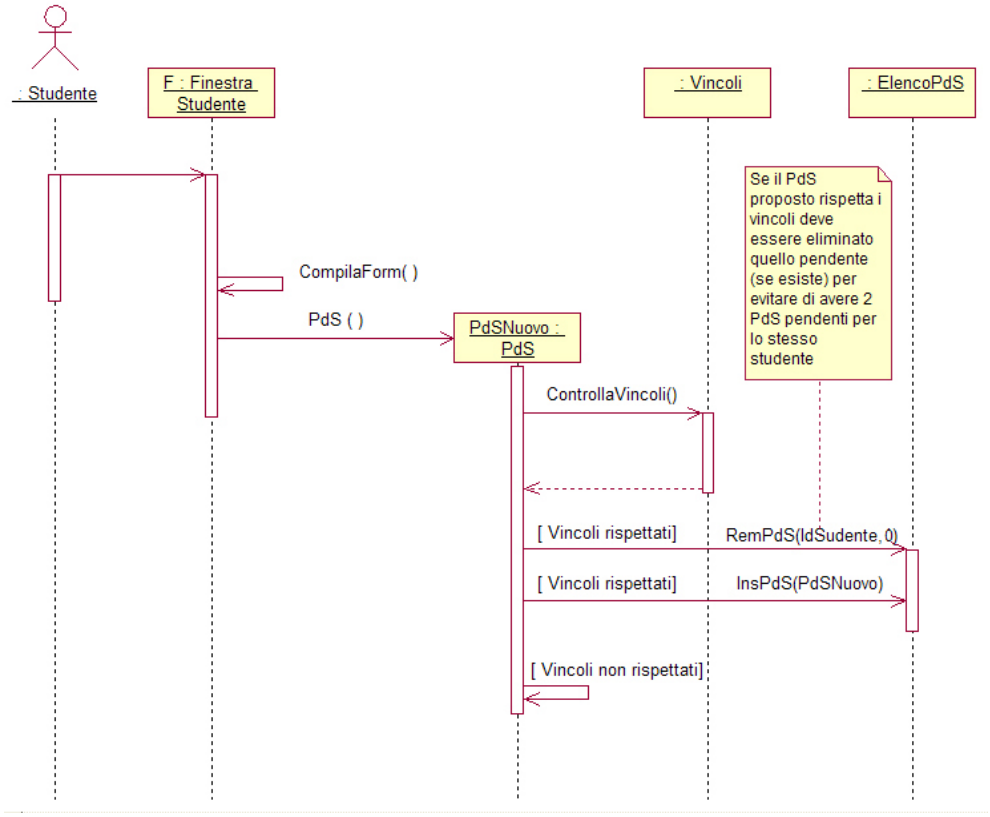


Figura 4.14: diagramma di sequenza: Compilazione PdS

standard viene analizzato e valutato singolarmente.

Approvazione PdS non standard

Come mostrato in figura 4.16, quando viene mostrata la finestra per l'approvazione dei Piani di Studi (`Mostra()`) viene invocata la funzione `ElencaPdSDaApprovare()` che ritorna una lista di PdS, tutti con l'attributo `approvato` settato a 0. Ogni Piano di Studi non standard (attributo `standard` settato a 0) viene visualizzato in una finestra (`VisualizzaPdS()`) che permette al CdCdS di analizzarlo in dettaglio per decidere se approvarlo o meno. Su ogni Piano di Studi che il CdCdS decide di approvare viene invocato il metodo `ApprovaPdS()` (questo setta l'attributo `approvato` a 1 ed eventualmente elimina il Piano di Studi approvato per lo studente cui è stato approvato in nuovo PdS). Viceversa, su ogni Piano di Studi che il CdCdS decide di non approvare viene invocato in

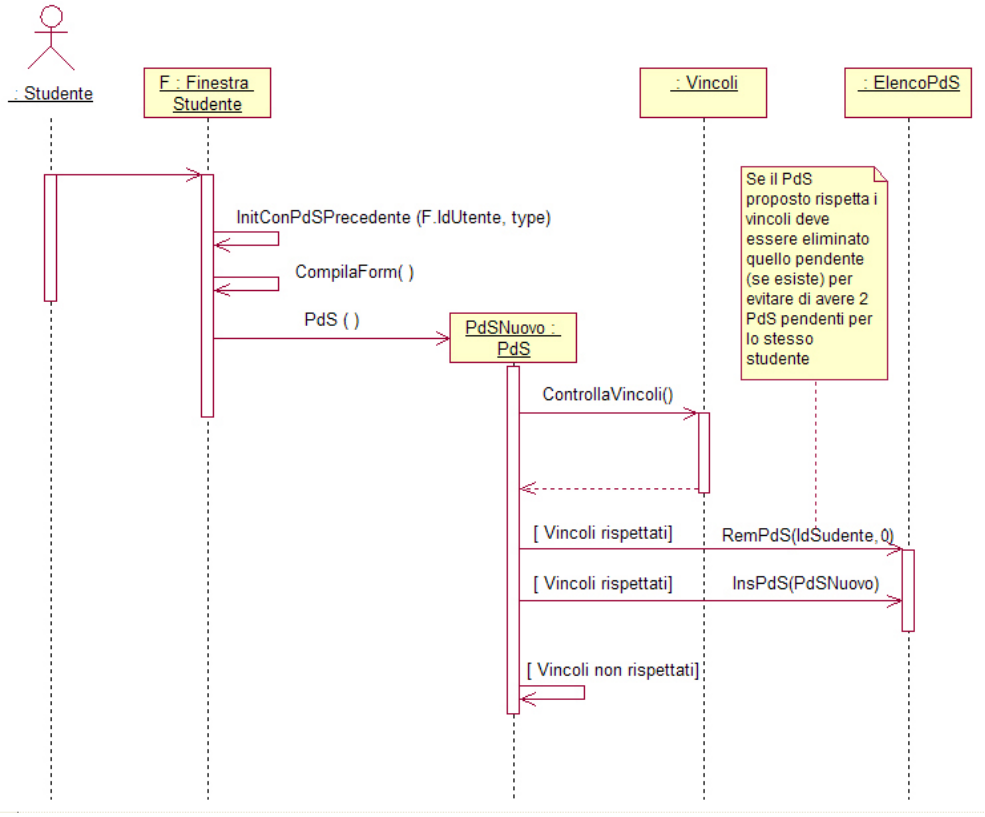


Figura 4.15: diagramma di sequenza: Modifica PdS

metodo `RemPdS()`, che elimina il PdS respinto da ElencoPdS.

Approvazione PdS standard

Come mostrato in figura 4.17, quando viene mostrata la finestra per l'approvazione dei Piani di Studi (`Mostra()`) viene invocata la funzione `ElencaPdSDaApprovare()` che ritorna una lista di PdS, tutti con l'attributo `approvato` settato a 0. Ogni PdS standard (attributo `standard` settato a 1) deve venire approvato in automatico, perciò su ogni PdS standard viene invocato il metodo `ApprovaPdS()` che setta l'attributo `approvato` a 1.

4.4.5 `Diagramma di sequenza`: Visualizzazione vincoli

Come mostrato in figura 4.19, quando uno studente richiede di consultare i vincoli per la compilazione dei Piani di Studi viene mostrata `FinestraStudente`,

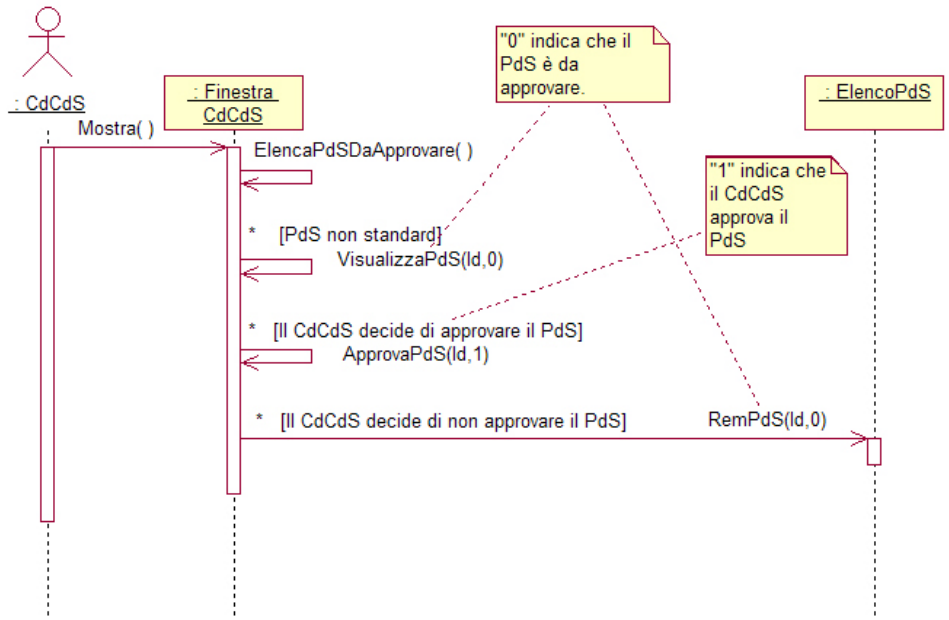


Figura 4.16: diagramma di sequenza: Approvazione PdS non standard

attraverso il metodo `Mostra()`. I vincoli vengono poi visualizzati a schermo attraverso il metodo `VisualizzaVincoli()`.

4.4.6 `Diagramma di sequenza:` Modifica vincoli

Come mostrato in figura 4.19, quando il CdCdS desidera modificare i parametri che stabiliscono i vincoli di approvazione viene mostrata `FinestraCdCdS`, attraverso il metodo `Mostra()`. Attraverso una chiamata al metodo `AggiornaVincoli()` il CdCdS può modificare i parametri dei vincoli.

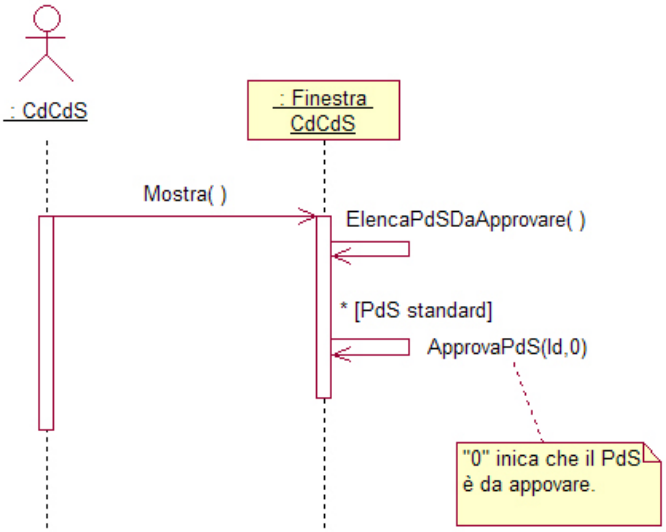


Figura 4.17: diagramma di sequenza: Approvazione PdS standard

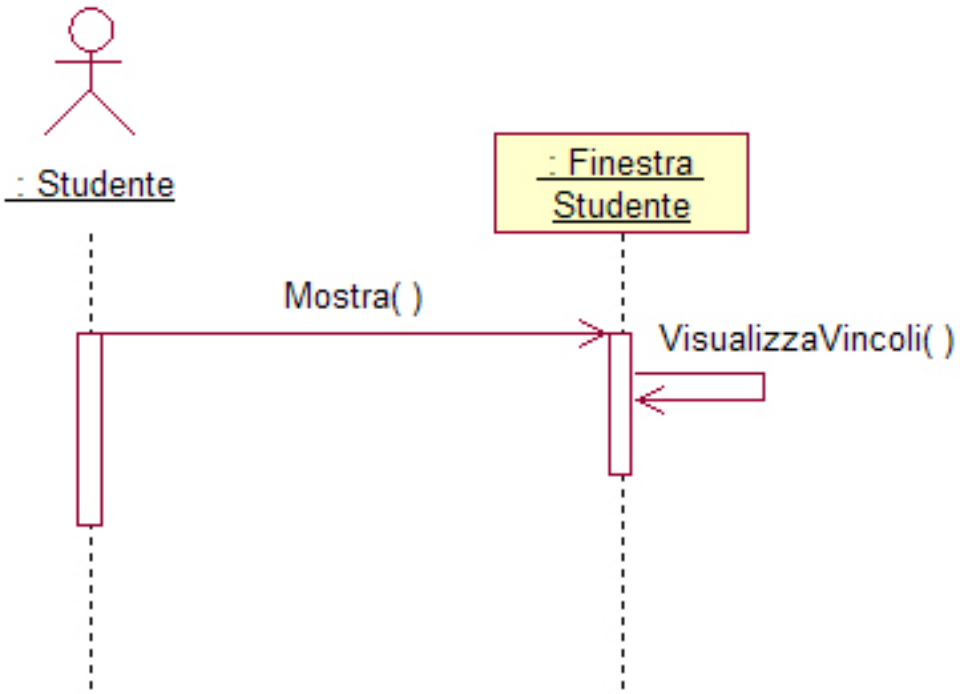


Figura 4.18: diagramma di sequenza: Visualizzazione vincoli

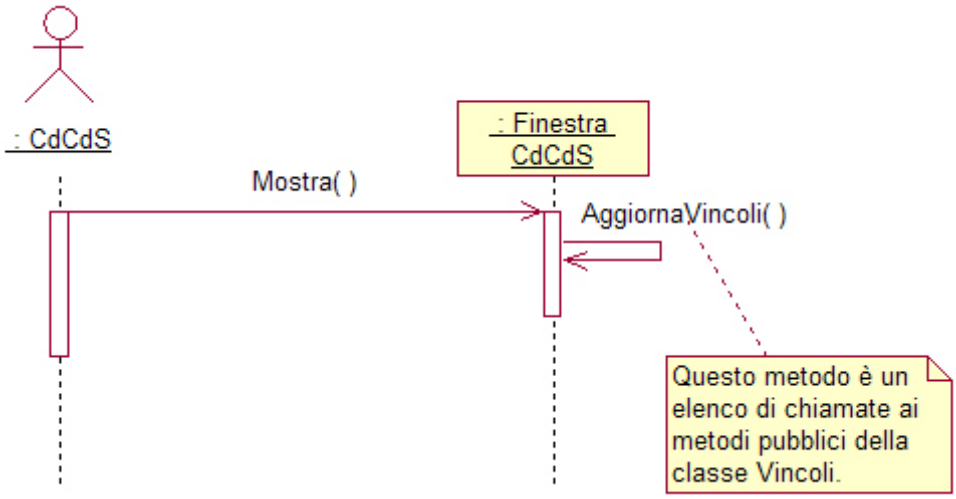


Figura 4.19: diagramma di sequenza: Modifica vincoli

4.5 Modelli UML a livello implementativo e architetturale

4.5.1 Diagramma dei componenti

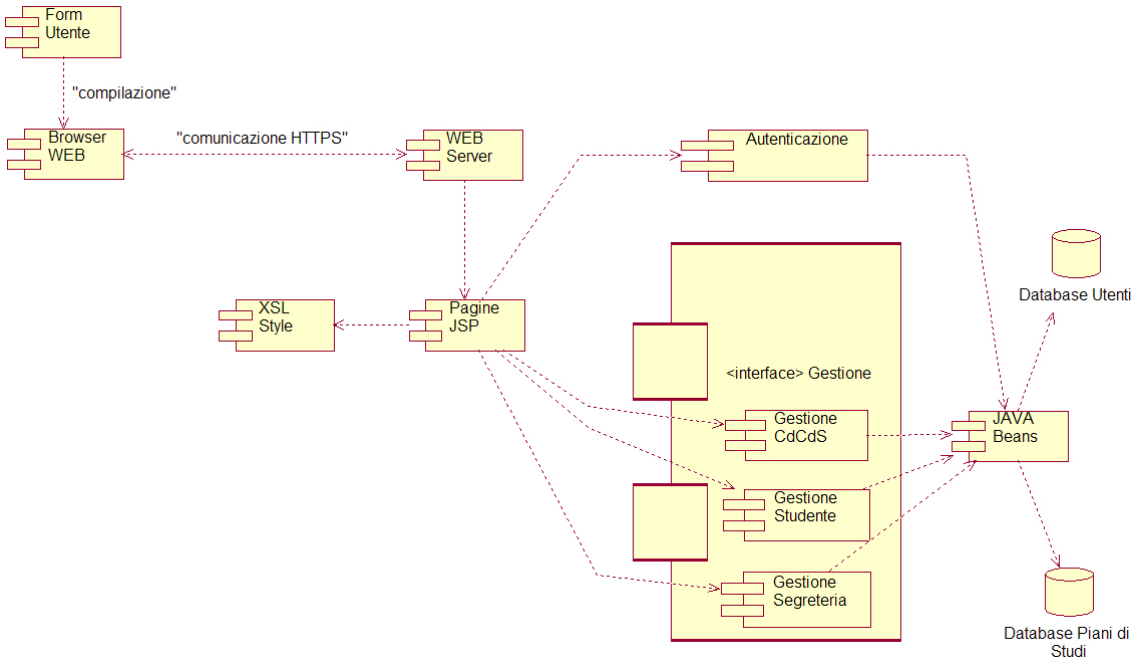


Figura 4.20: diagramma dei componenti

Ogni utente (Studente, CdCdS o Segreteria), dopo essersi autenticato presso il al server, utilizza il componente `Form Utente` per realizzare tutte le interazioni con il sistema. `Form Utente` rappresenta le varie finestre che vengono visualizzate sul Browser Web per le varie operazioni previste: `Autenticazione`, `GestioneSegreteria`, `GestioneCdCdS` e `GestioneStudente`. Le applicazioni `Browser Web` e `Web Server` non verranno implementate in quanto applicazioni già esistenti e che si presume essere installate sulle macchine che compongono il sistema; in realtà un prerequisito indispensabile affinché l'utente possa utilizzare il servizio è avere a disposizione queste due applicazioni. La struttura del sistema si basa sull'architettura client-server, dove il client sincronizza il proprio comportamento in base alle risposte del

server. Una volta compilato le varie finestre (`Form Utente`) il `Browser Web` comunica le informazioni al `Web Server` sul quale risiedono le applicazioni sviluppate su un'architettura J2EE (JBOSS). Il servizio utilizza le pagine dinamiche JSP per la comunicazione e la visualizzazione dei dati richiesti dall'utente. Le pagine JSP compilate dall'utente vengono parsate secondo uno schema XML (XSL Style) grazie al quale si ricavano le informazioni semantiche sulle quali opereranno `Autenticazione` oppure `Gestione`. Questi ultimi due componenti sono applicazioni sviluppate in JAVA lato server e che si appoggiano all'architettura J2EE (JBOSS). Queste utilizzano le componenti JAVA Beans per interfacciarsi con i database *Database Utenti* e *Database Piani di Studi* al fine di recuperare o registrare le informazioni gestite dalle applicazioni. La comunicazione tra i componenti non si appoggia su particolari tipi di protocolli di crittografia, eccezion fatta per Browser WEB - WEB Server i quali comunicano utilizzando il protocollo HTTPS con cifratura a 128 bit.

4.5.2 `Diagramma delle classi` - livello di implementazione

Il diagramma in figura 4.21 rappresenta lo schema delle classi a livello implementativo, ovvero la struttura del codice che verrà prodotto in seguito. Il livello di dettaglio è piuttosto alto in modo da poter giungere al codice in modo quasi automatico. Per ogni classe vengono rappresentati sia i metodi di interfaccia che quelli locali; i costruttori e i metodi per l'accesso ai campi privati sono sottintesi.

Descrizione delle classi, degli attributi e dei metodi

- `Autenticazione` *Crea la finestra di autenticazione*

  `Instance()` Metodo statico per la crazione di un'istanza della finestra di autenticazione ad ogni connessione di un client

  `Check(id,password)` Autentica la coppia user_id e password controllando all'interno della lista di utenti attraverso il metodo `search(id,password)`

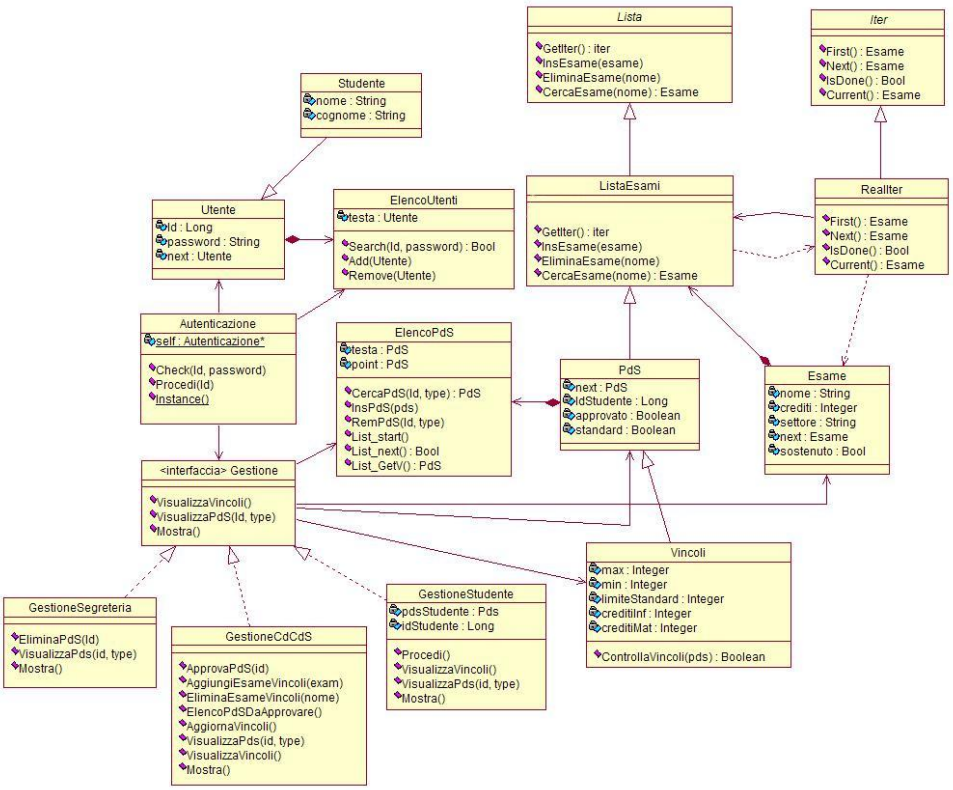


Figura 4.21: diagramma delle classi: livello di implementazione

`Procedi(id)` Chiama l'apertura dell'interfaccia di gestione specifica per l'utente autenticato

`self:Autenticazione *` Puntatore all'istanza creata della classe

- ***`Utente`*** *Rappresenta ogni utente che può interagire con il sistema*

  `id` User id dell'utente

  `password` Password dell'utente

  `next` Puntatore al prossimo utente in `ElencoUtenti`

- **`Studente`** *Classe che estende la classe* ***`utente`***

  `nome` Nome dello studente

  `cognome` Cognome dello studente

- `ElencoUtenti` *Lista degli utenti del sistema*

  `Search(id,password):Bool` Metodo per il controllo della coppia user_id e password con quelli contenuti nella lista. Ritorna `true` se c'è corrispondenza

  `Add(Utente)` Inserisce un utente nella lista

  `Remove(Utente)` Rimuove un utente

  `testa` Puntatore al primo utente della lista

- `Gestione` *Classe base per la gestione delle funzioni dei vari utenti*

  `VisualizzaVincoli()` Visualizza tutti i vincoli

  `VisualizzaPdS(id,type)` Visualizza il PdS identificato da `it` e `type`

  `Mostra()` Mostra la pagina specifica in base all'utente

- `GestioneSegreteria` *Classe per la gestione dei metodi specifici della segreteria*

  `EliminaPds(id)` Elimina tutti i PdS relativi all'utente identificato da id

  `Mostra()` Mostra la pagina dando la possibiltà di eliminare un PdS o di visualizzare un PdS

  `VisualizzaPdS(id,type)` Visualizza il PdS identificato da `it` e `type`

- `GestioneCdCdS` *Classe per la gestione dei metodi per la modifica dei vincoli e per l'approvazione dei PdS da parte del CdCdS*

  `ApprovaPds(id)` Approva il PdS identificato da `id`

  `AggiungeEsameVincoli(exam)` Aggiunge un esame ai vincoli

  `EliminaEsameVincoli` Rimuove un esame dai vincoli

  `ElencoPdsDaApprovare()` Elenca uno alla volta i PdS da approvare: il CdCdS deciderà se approvarli o meno

  `AggiornaVincoli()` Aggiorna i vincoli numerici modificati

`VisualizzaVincoli()` Visualizza i vincoli (numerici ed esami) dando la possibilità di modificarli

`VisualizzaPds(id,type)` Visualizza il PdS da approvare identificato da `id` dando la possibilità di approvarlo o eliminarlo

`Mostra()` Mostra la pagina dando la possibiltà di visualizzare i PdS da approvare e di modificare i vincoli

- `GestioneStudente` *Classe per la gestione dei metodi per la gestione dei PdS e la visualizzazione dei vincoli da parte dello studente*

  `VisualizzaPds(id,type)` Visualizza il PdS identificato da `id` e dal tipo dando la possibilità di inserire nuovi esami dalla lista; se non esiste visualizza la lista di esami da scegliere

  `Procedi()` Crea un nuovo PdS in base agli esami scelti; viene controllata la conformità con i vincoli grazie a `ControllaVincoli(PdS)`. Se conforme viene inserito tra i PdS da approvare

  `VisualizzaVincoli()` Visualizza i vincoli

  `Mostra()` Mostra la pagina dando la possibiltà di visualizzare i vincoli e di visualizzare/modificare il PdS dello studente

  `IdStudente` Identidicativo dello studente

  `PdsStudente` Pds dello studente

- `Esame` *Classe che rappresenta l'entità esame*

  `nome` Stringa che rappresenta il nome dell'esame

  `crediti` Intero che rappresenta i crediti dell'esame

  `settore` Stringa che rappresenta il settore

  `sostenuto` Boolean che rappresenta il fatto che un esame sia stato sostenuto o meno

  `next` Puntatore al prossimo esame

- `Lista` *Classe astratta che rappresenta i metodi utili ad una lista di esami*

`GettIter():iter` Metodo che ritorna un iteratore per una lista

`InsEsame(esame)` Metodo per inserire un esame in una lista

`EliminaEsame(nome)` Metodo per eliminare un esame da una lista

`CercaEsame(nome):Esame` Metodo per la ricerca di un esame in una lista

- `ListaEsami` *Implementazione dei metodi della classe astratta* `Lista`
- `Iter` *Classe astratta che rappresenta un iteratore per una lista*

  `First():Esame` Metodo che ritorna il primo elemento di una lista

  `Next():Esame` Metodo che fa avanzare l'iteratore

  `IsDone():Bool` Metodo per il controllo della fine della lista. Ritorna `true` se la lista è terminata

  `Current():Esame` Metodo che ritorna l'elemento corrente di una lista

- `RealIter` *Classe che implementa i metodi definiti in* `Iter` *per la gestione di un iteratore per una lista*
- `Pds` *Classe che rappresenta l'entità Piano di Studi; altro non è che una lista di esami. Estende la classe ListaEsami*

  `idStudente` Unsigned Long che rappresenta la mastricola dello studente

  `approvato` Bool che identifica se un PdS è stato approvato o meno del CdCdS

  `standard` Bool che identifica se un PdS è standard o meno

  `next` Puntatore al prossimo elemento di ElencoPdS

- `ElencoPds` *Classe che rappresenta un elenco di PdS*

  `CercaPds(Id,type):PdS` Cerca e ritorna un PdS in base all'Id e al tipo(approvato o non approvato)

  `InsPds(PdS)` Inserisce un PdS nella lista

`RemPds(id,type)` Rimuove un PdS dall'elenco in base al suo id e al tipo

`List_star()` Inizializza l'iteratore per la lista

`List_next():Bool` Fa avanzare l'iteratore, ritorna `false` se la lista è terminata

`List_GetV():PdS` Ritorna il PdS al punto corrente dell'iteratore

`testa:PdS` Puntatore al primo PdS della lista

`point:PdS` Attuale posizione dell'iteratore

- `Vincoli` *Classe per la gestione dei vincoli*

`ControllaVincoli(PdS)` Controlla se il PdS passato è coerente con i vincoli espressi dal CdCdS

`max` Intero che rappresenta il numero massimo di crediti

`min` Intero che rappresenta il numero minimo di crediti

`limiteStandard` Intero che rappresenta il numero di crediti oltre cui un PdS viene considerato comunque non standard

`creditiInf` Intero che rappresenta il numero minimo di crediti informatici

`creditiMat` Intero che rappresenta il numero minimo di crediti matematici

4.5.3 Descrizione dei *design pattern*

Nella definizione delle classi a livello implementativo abbiamo utilizzato due `design pattern` [1] : uno per l'implementazione dell'iteratore delle liste e l'altro per la gestione degli accessi. Il primo design pattern utilizzato è *Iterator*: definisce un metodo per l'accesso e lo scorrimento ai dati di una lista. È implementato dalle classi `Lista, ListaEsami, Iter, Realiter` ed `Esame`; fornisce quattro funzioni per l'inizializzazione dell'iteratore, per l'avanzamento, la restituzione dell'elemento corrente e il controllo di fine. Lo stesso principio è stato definito in maniera differente all'interno della

classe `ElencoPdS`, dando quindi un'implementazione alternativa dell'iteratore. Il secondo `design pattern` utilizzato è `Singleton`, per gestire attraverso un'unica classe accessi multipli al sistema. È implementato dalla classe`Autenticazione` e dal metodo di accesso del client che è sottinteso. Ogni volta che un client richiede l'accesso al sistema viene istanziata una nuova classe `Autenticazione` associata al client attraverso il metodo `Instance()`.

4.5.4 Diagramma dei package

Data la non eccessiva complessità del sistema progettato ed il numero non troppo elevato di classi abbiamo ritenuto opportuno non definire alcun diagramma dei package; ad ogni modo, facendo riferimento al diagramma delle classi a livello di implementazione, sarebbe stato possibile individuare tre package principali come mostrato nella figura 4.21:

`Piani di Studi` : PdS, ElencoPdS, Vincoli, ListaEsami, Esame, Vincoli

`Utenti` Utente, Studente, ElencoUtenti

`Interfacce di Sistema` Autenticazione, Gestione, GestioneSegreteria, GestioneCdCdS, GestioneStudente

Le dipendenze che sarebbero individuate in questo eventuale diagramma dei package sarebbero due:

`da Interfaccia di Sistema a PdS` in quanto le funzionalità messe a disposizione degli utenti attraverso le interfacce di sistema richiamano vari metodi delle classi nel package `Piani di Studi` e utilizzano istanze di classi di questo package come parametri per le chiamate a questi metodi

`da Utenti a Interfaccia di Sistema` in quanto tutti gli utenti richiamano metodi della classe Autenticazione

`da Interfaccia di Sistema a Utenti` in quanto la classe Autenticazione invoca il metodo SearchId di ElencoUtenti.

Capitolo 5

Conclusioni

L'applicazione degli strumenti messi a disposizione dall'ingegneria del software ha permesso lo sviluppo di una documentazione completa per lo sviluppo e il mantenimento dell'aplicazione che andremo ad implementare. Grazie ai diagrammi delle classi a livello implementativo e di quelli di sequenza, si è ridotto notevolmente lo svorzo di implementazione lasciando le difficoltà solo per la fase di deplyment sull'architettura su cui verrà eseguito il software. Purtroppo le stime dei tempi e dello sforzo necessario allo sviluppo non sono attendibili, per i motivi descritti nei capitoli precedenti; non è quindi possibile valutare con correttezza l'attività che sarà necessaria all'implementazione.

La documentazione fornita appare come un buon strumento di presentazione dell'aplicazione ad un ipotetico cliente, nonché un valido aiuto all'eventuale equipe di mantenimento del software.

Il passo successivo sarà l'implementazione del software con la conseguente verifica delle stime e della progettazione fatta in fase di analisi.

Appendice A

Applicazioni utilizzate

Durante l'attività di progettazione sono stati utilizzati i seguenti tool e programmi:

Rational Rose per la generazione dei diagrammi UML

Microsoft Project 2003 per la generazione del diagramma di Gantt

COSTAR 7.0 per le stime dei tempi con COCOMO2

Per quanto riguarda invece le attività di stesura della documentazione sono stati utilizzati:

Adobe Photoshop per il ritocco di immagini

DreamWeaver MX 2004 per la gestione del sito

WinEdt 5.3 come editor di documenti LaTeX

MikTek come compilatore di documenti LaTeX

Adobe Acrobat Reader

Per la gestione remota e lo scambio del materiale prodotto tra i vari membri del gruppo sono stati utilizzati diversi tool di trasferimento file: Winscp, Scp, FtpPro, mail

A.1 Problemi riscontrati nel software utilizzato

L'unico vero problema è stato riscontrato nell'utilizzo di Rational Rose. L'applicazione ha presentato alcuni problemi nella creazione di alcuni schemi UML; in particolare non abbiamo trovato affatto agevole l'utilizzo dell'interfaccia nella modifica e nella gestione degli schemi già creati. In alcuni casi l'applicazione diventava instabile e necessitava un riavvio per poter continuare nel lavoro. Proprio per questi motivi è nata la necessità di utilizzare un software di fotoritocco come Photoshop per poter modificare alcuni diagrammi, che altrimenti Rational Rose non ci permetteva di modificare nella maniera corretta.

Bibliografia

[1] The object oriented pattern digest. http://patterndigest.com.

[2] Martin Fowler. *UML Distilled*. Prima edition, 2000.

[3] J. Lewi, E. Steegmans, and J. De Man. Object-oriented approach to software development, a walk through a number of topics. In *Proceeding of 5th Annual European Computer Conference. CompEuro '91, Advanced Computer Tecnology, Reliable Systems and Applications*, pages 626–633, Maggio 1991.

[4] David Longstreet. *Function Points Analysis Training Course*.

[5] Mark C. Paulk, Bill Curtis, Mary B. Chrissis, and Charles V. Weber. Capability maturity model, version 1.1. *IEEE Software*, 10(4):18–27, Luglio 1993.

[6] Roger S. Pressman. *Principi di ingegneria del software*. Terza edition, 2000.

[7] University of Southern California. *USC COCOMOII Reference Manual*.