Dipartimento di Ingegneria e Scienza dell'Informazione
2022/2023

YINCO

Documento di architettura

Doc. Name	D3 YINCO Documento di architettura	Doc. Number	Rev 0.3
Description	Il documento include diagrammi delle classi e codice in OCL		

INDICE

1. Scopo del documento

Il presente documento riporta la definizione dell'architettura del progetto Yinco usando diagrammi delle classi in Unified Modeling Language (UML) e codice in Object Constraint Language (OCL). Nei seguenti paragrafi definiremo l'architettura del sistema dettagliando da un lato le classi che dovranno essere implementate a livello di codice e dall'altro la logica che regola il comportamento del software. Le classi vengono rappresentate tramite un digramma delle classi in linguaggio UML. La logica viene descritta in OCL perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

2. Diagramma delle classi

Nel seguente capitolo vengono presentate le classi previste nell'ambito del progetto Yinco. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi che identificano i dati gestiti dalla classe e una lista di metodi che definiscono le operazioni previste all'interno della classe. Ogni classe può essere anche associata ad altre classi e, tramite questa associazione, è possibile fornire informazioni su come queste si relazionano tra loro. Riportiamo di seguito le classi individuate attraverso vari gruppi principali in cui si divide il progetto, per facilitarne la comprensione.

2.1 Utenti

Analizzando lo use case diagram realizzato per il progetto Yinco, si nota la presenza di due attori "**utente anonimo**" e "**utente autenticato**". L'attore "utente anonimo" è colui che ha un ristretto numero di domande che può richiedere al sistema (cfr obiettivi D1), mentre "utente autenticato" è colui che ha un maggiore numero di domande che può richiedere al sistema. Entrambi questi attori hanno specifiche funzioni e attributi, ma hanno anche molto in comune. Sono state quindi individuate due classi "**Utente_Anonimo**" e "**Utente_Autenticato**" con attributi e funzioni specifici e una classe "**utente**" con funzioni e attributi in comune. Le classi "utenteAnonimo" e "utenteAutenticato" sono collegate alla classe "utente" tramite una generalizzazione. La classe utente ha come attributi l'email e la password dell'utente e un booleano, "**is_online**" il quale verrà utilizzato per capire se l'utente sia autenticato o meno in quanto cambiano il modo in cui l'utente potrà interagire con il sistema. I metodi della classe "Utente" sono semplicemente un metodo "login" e un metodo "domanda_utente". La classe "Utente_Anonimo" non ha ulteriori attributi o metodi in quanto sono tutti già specificati nella classe "Utente". La classe "Utente_Autenticato" invece ha un ulteriore attributo "tempo_trascorso_ultimo_accesso" di tipo minuti il quale viene utilizzato per verificare una condizione del metodo "login_automatico". Il secondo metodo della classe "Utente_Autenticato" è il metodo "logout".

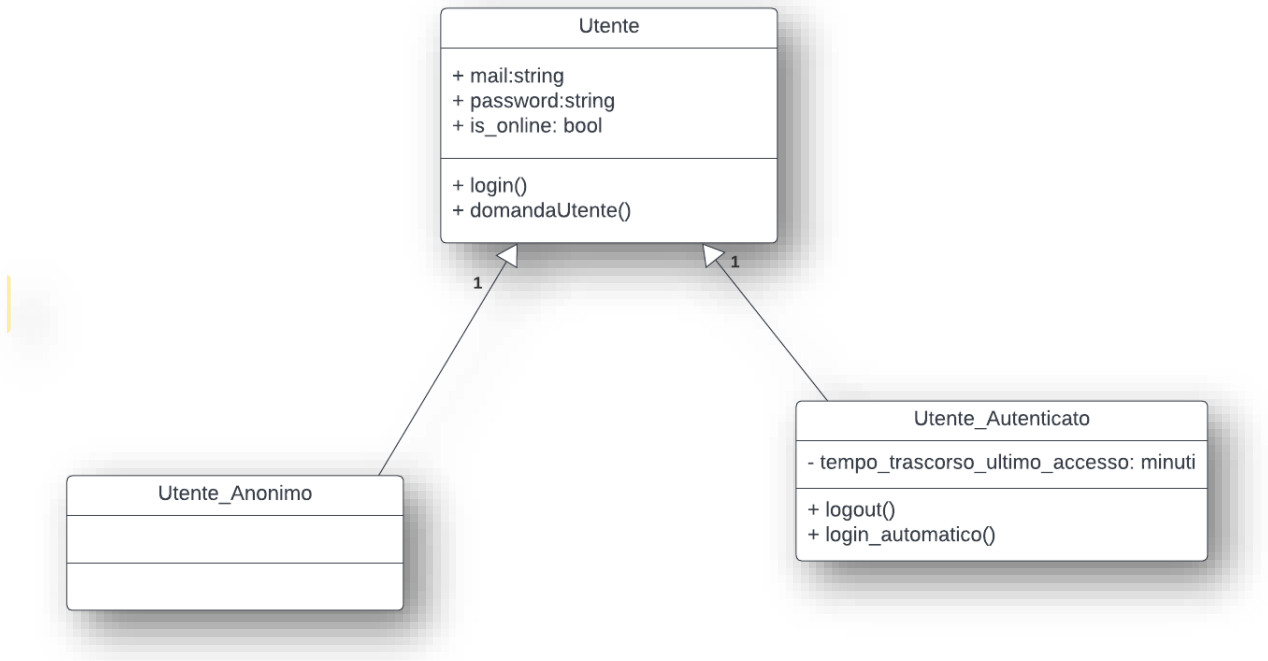

Figura 2.1: Specifica classi utente

2.2 Gestione autenticazione

Lo use case diagram analizzato presenta un sistema subordinato denominato "sistema credenziali universitarie". Questo elemento rappresenta il meccanismo di autenticazione degli utenti attraverso un sistema esterno che gestisce le credenziali universitarie. È stata quindi identificata una classe "**autenticazione**" la quale si interfaccia con il sistema di gestione delle credenziali dell'ateneo. Il software sviluppato per il progetto Yinco non memorizzerà le e-mail e le password di ogni singolo utente, ma passerà i dati di autenticazione ad un sistema paritario esterno, il sistema di credenziali universitarie, che valuterà questi dati e risponderà specificando se le credenziali inserite sono valide o meno. Gli attributi di questa classe sono semplicemente la mail e la password dell'utente. Il metodo "**verifica_credenziali**" verificherà se la mail e la password inserite siano corrette.

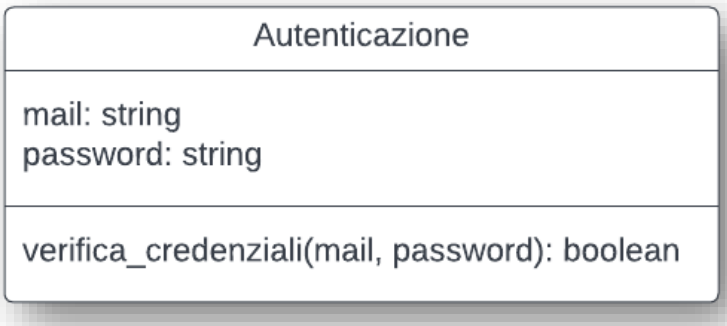

Figura 2.2: Specifica classe autenticazione

2.3 Interfaccia utente

L'**interfaccia utente** è il cuore di questo progetto, in quanto da essa si può accedere alle impostazioni, si può interagire con il sistema Yinco e si può trovare informazioni riguardo ai relatori del sistema. Quindi, vista la complessità della componente, sono state create 4 classi opportunamente relazionate. La classe **relatori sistema** viene utilizzata come ausiliaria rispetto alla classe '**Contatti**' ed è stata ideata per poter identificare il tipo dell'attributo "**amministratori**". Nella classe contatti possiamo trovare due attributi: l'attributo "amministratori" di tipo relatori_sistema e l'attributo "**sede_progetto**" di tipo immagine. La sede del progetto viene ottenuta utilizzando il metodo "**get_mappa**" utilizzando l'API di Google Maps. La classe "**impostazioni**" ha un attributo "**lingua**" del tipo enum in quanto l'utente avrà la possibilità di scegliere se preferisce avere il sistema in lingua italiana o inglese. Questa classe presenta due metodi, un metodo "**preferenza_lingua**", il quale prende come parametro l'attributo lingua e un metodo "**preferenza_notifiche**". Il metodo preferenza_notifiche verrà poi utilizzato per capire se l'utente vuole ricevere nella propria mail istituzionale delle mail riguardanti gli esami e le tasse. Questa funzione verrà vista più approfonditamente nella parte di OCL delle notifiche. Per finire la classe "**Interfaccia_Utente**" ha due attributi: "**domanda_utente**" e "**risposta**" entrambi di tipo string. Questi due attributi vanno a indicare ciò che il sistema riceverà (domanda_utente) e invierà (risposta) quando l'utente anonimo o autenticato che sia interagirà con il sistema. Il metodo di questa classe (invio_risposta) serve a poter inviare all'utente una risposta alla domanda ricevuta.

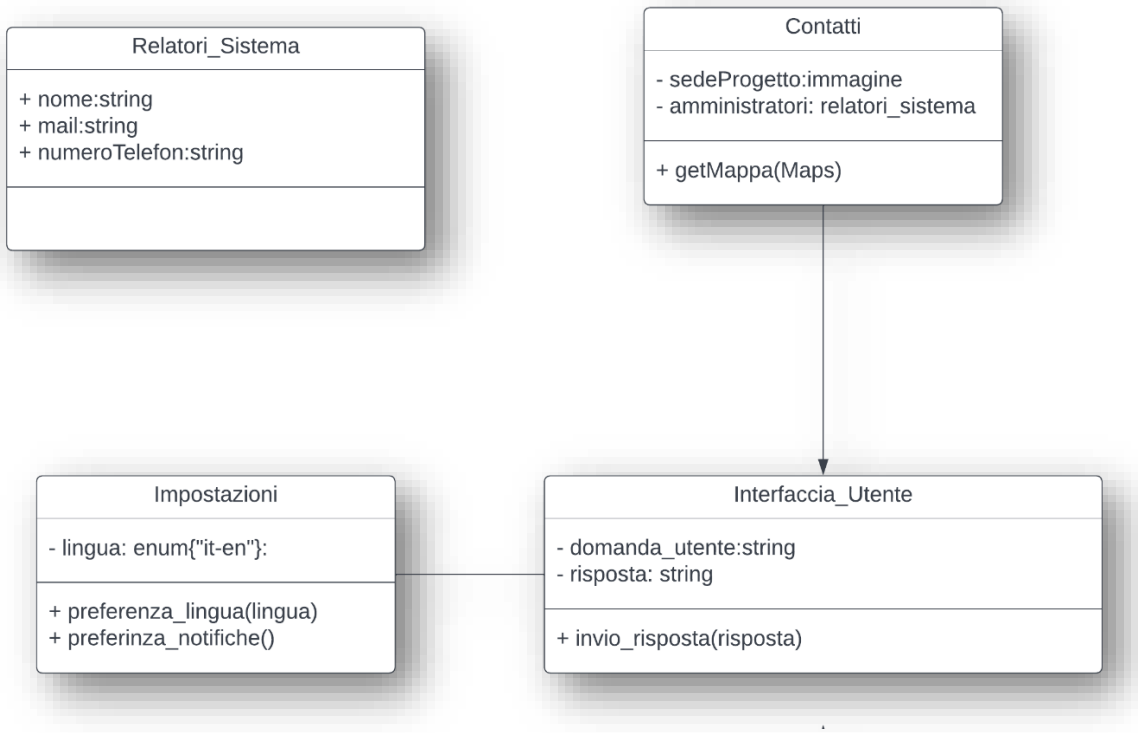

Figura 2.3: Specifica classi interfaccia utente

2.4 Ricerca

Nel diagramma dei componenti era possibile vedere una sezione dedicata alla ricerca della risposta alla domanda posta dall'utente, nominalmente rappresentata dai componenti: Gestione Chat, Ricerca Docente e Ricerca Database. In base a questa partizione, sono state individuate tre classi: la classe **Ricerca**, la classe **Docente** e la classe **Informazione**.

RICERCA

La classe **ricerca** è la classe principale di questa sezione e presenta al suo interno diversi attributi e metodi. Per quanto riguarda gli attributi, ne presenta uno che indica il livello di accesso dell'utente, ovvero se è un utente autenticato oppure anonimo (**authentication**), uno che indica il database che bisogna interpellare per poter ottenere i dati richiesti (**chosen_database**), un campo usato per ritornare il docente trovato durante la ricerca (**docente_da_ritornare**), così come uno per le informazioni (**info_da_ritornare**) e un attributo usato per contenere le parole chiave attraverso cui effettuare la ricerca e la decisione del database (**keyword**). Per quanto riguarda i metodi, **create_and_return_link** è usata per generare il link da presentare all'utente per l'informazione che il database esterno, se la ricerca ha avuto successo, è stato in grado di tornare al sistema; **return_docente** è usato per ritornare all'utente il link contenuto all'interno della classe docente; **decide_database** è il metodo usato per decidere a quale database inviare la richiesta in base alle keyword individuate dal metodo **checkKeyword**, il quale prende la domanda posta dall'utente e contenuta all'interno della classe Interfaccia Utente, la analizza e poi ritorna le varie keywords trovate; infine, **verify_user** è usato per verificare che il livello di privilegi dell'utente corrisponda a quello della informazione che ha cercato: se corrisponde, allora viene chiamato il metodo per propagare all'utente l'informazione ottenuta, altrimenti dovrà mandare all'utente un errore.

DOCENTE

La classe docente serve per identificare gli attributi chiave che un dato di tipo docente deve avere e che devono essere, perciò, richiesti e ritornati dal database esterno con cui si configura. Gli attributi della classe sono il cognome del docente (**cognome**), il link alla sua pagina Unitrento Digital University (del tipo https://webapps.unitn.it/du/it/Persona/), contenuto nell'attributo link, e una copia delle keyword ottenute nella classe Ricerca. Questa classe presenta solo due metodi: il metodo di ricerca del docente sul database esterno (**effettua_ricerca**), effettuato in base alle keyword precedentemente ottenute, e il metodo di propagazione del dato ottenuto alla classe Ricerca (**ritorna_docente**), il quale poi lo ritornerà all'utente con il metodo return_docente già visto nel paragrafo precedente.

INFORMAZIONE

La classe informazione, come la classe Docente, serve per identificare gli attributi che compongono l'oggetto che poi dovrà essere richiesta al database esterno. Essa ha vari attributi di base, come il titolo (**title**), l'anno in cui è stata scritta la notizia (**year**), il

testo in sé e per sé (**body**), dei tags che sono proprio dell'informazione (**tags**) e su cui verrà effettuata la ricerca, confrontandoli con le keyword trovate nella classe Ricerca e presenti anche in questa classe (keywords). Infine, è presente anche un attributo (**privilege_level**), che indica il livello di privilegi che bisogna avere per accedere all'informazione. Questo attributo non è presente nella classe Docente in quanto è stato deciso, in fase di progettazione, di permettere la ricerca dei docenti solo agli utenti autenticati, dunque sarebbe stato futile aggiungere un attributo costante. Per quanto riguarda i metodi, questa classe ha un metodo di ricerca sul database (**effettua_ricerca_info**), che confronterà le keyword e i tags delle varie informazioni per trovare quelle che soddisfano la richiesta, e un metodo di ritorno dell'informazione alla classe Ricerca, la quale poi lo ritornerà all'utente (**ritorna_ricerca_info**).

Di seguito è possibile trovare una figura che illustra queste tre classi e le relazioni che intercorrono tra di loro. È importante notare che tra la classe ricerca e le due classi docente e informazione sussiste una cardinalità del tipo 1:1, in quanto un utente può effettuare una sola ricerca alla volta e solo un docente o un'informazione può essere ritornato in una ricerca.

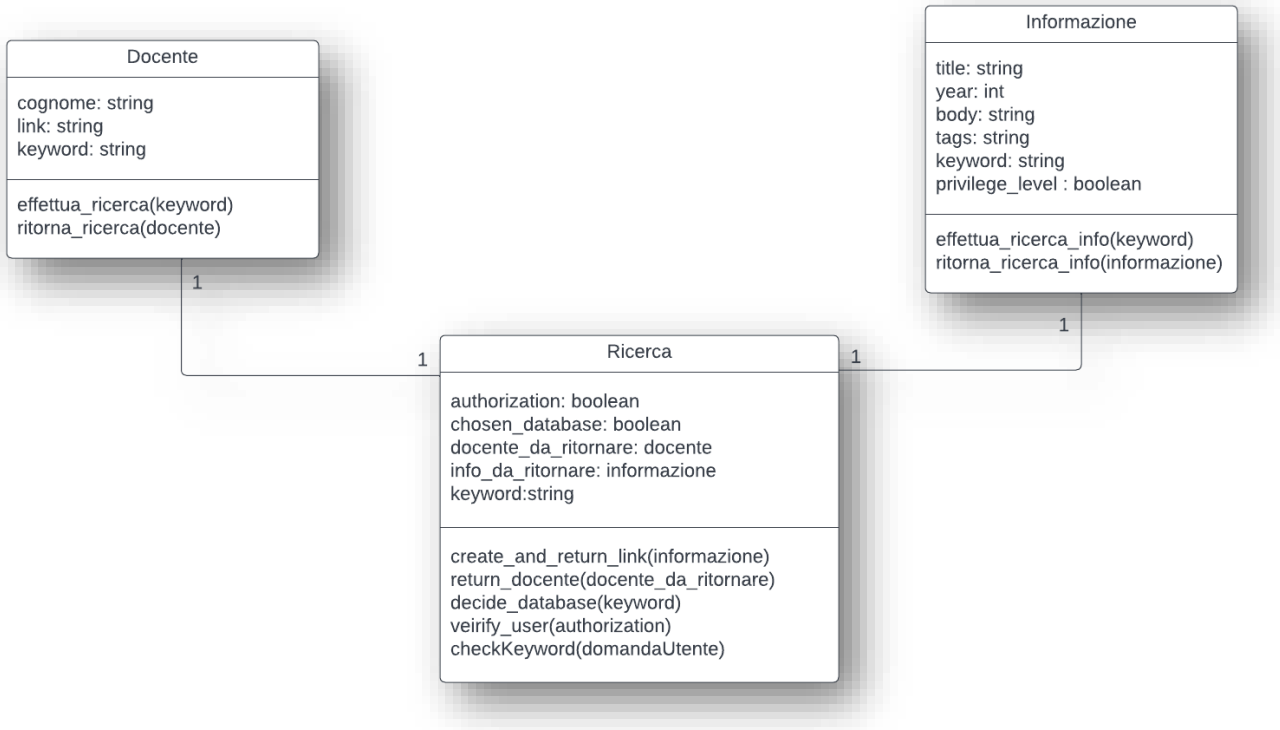

Figura 2.4: Specifica delle classe Ricerca, Docente ed Informazione

2.5 Gestione notifiche

Per quanto riguarda la gestione delle notifiche di Yinco, si è deciso di dividere questo processo in tre classi: **Mail**, **Esse3** e **Preferenze**. Quest'ultime, inoltre, sfruttano a loro volta altre quattro classi ausiliari, per la gestione di alcuni loro attributi. Queste sono: **Esami**, **Data**, **Preferenza** e **Tasse.** La classe **data** ha tre interi che definiscono il giorno, il mese e l'anno; la classe **esami**, presenta come attributi la materia a cui fa

riferimento e il giorno in cui si terrà l'esame (di tipo data); **Preferenza** possiede solo una stringa, ovvero l'email, e un booleano per la preferenza; infine, la classe **Tasse** che presenta l'importo e la scadenza entro cui pagare, anch'esso di tipo data.

ESSE3

All'interno di questa classe gestiamo tutto ciò che riguarda gli esami e le tasse di uno studente, proprio per questo abbiamo due primi attributi **tassa** e **esame**, rispettivamente dei tipi descritti poco fa; e i secondi due, **send_tassa** e **send_esame**, sono due booleani che verranno sfruttati per l'invio delle email di notifica. Questo processo verrà spiegato nel prossimo capitolo attraverso del codice OCL. Inoltre, troviamo anche due metodi simili i quali ci modificano, in vero o in falso, gli attributi specifici presenti nella classe nel caso in cui ci si trovi in prossimità della data di scadenza di un evento e non lo si sia già svolto, questi metodi sono **check_data_tassa** e **check_data_esame.**

PREFERENZE

La classe **Preferenze** presenta un unico attributo, **preferenza_utente**, di tipo Preferenza, il quale ci permette di salvare in modo agevole tutte le preferenze degli utenti del nostro sistema. Troviamo anche un metodo, **set_preferenza**, il quale dato in input una mail, ne va a modificare la preferenza.

MAIL

La seguente classe, nonostante non sia di eccessive dimensioni, è la più importante delle tre, poiché è proprio da essa che parte l'invio vero e proprio delle email. Qui troviamo un unico attributo che è la **mail** e due metodi **gmail_tasse** e **gmail_esami**, i quali invieranno le notifiche riguardo i loro specifici argomenti sfruttando i valori salvati in Esse3. Anche in questo caso, la decisione del momento in cui inviare tali mail verrà spiegato meglio con del codice OCL.

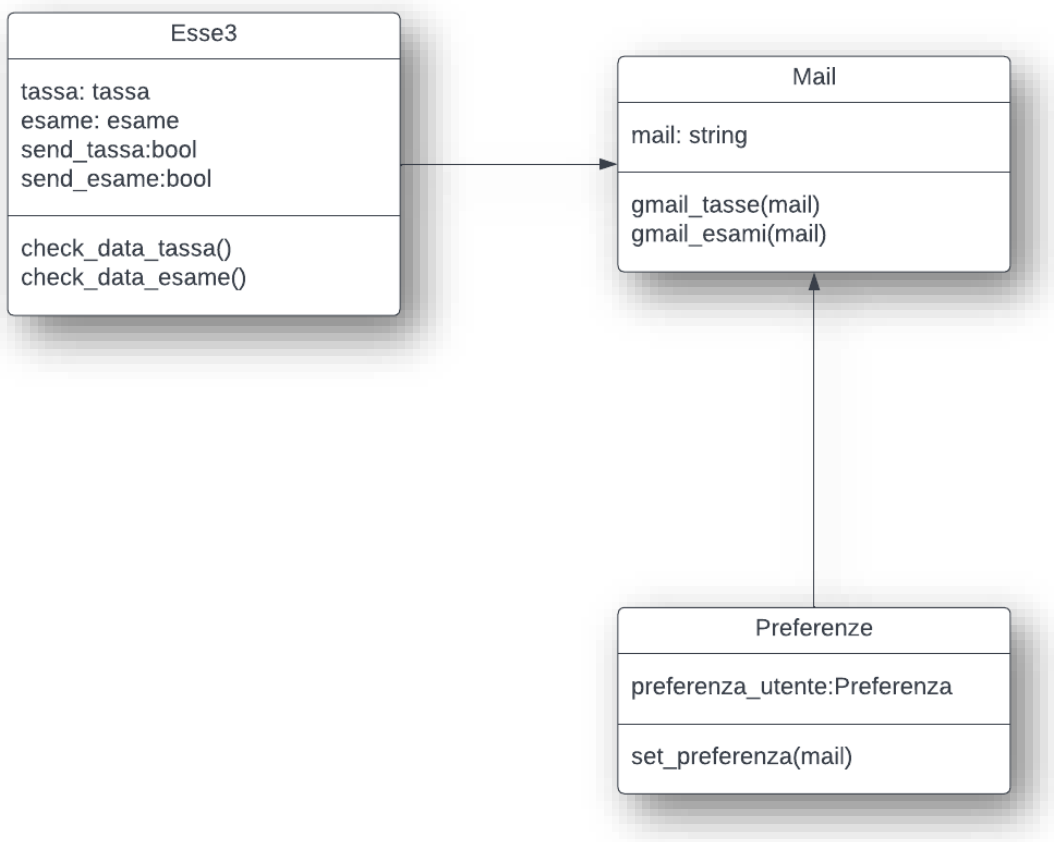

Figura 2.4: Specifica classi di notifica

3. Codice in Object Constraint Language

In questo capitolo è descritto in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL) perché tali concetti non sono esprimibili in nessun altro modo formale nel contesto di UML.

3.1 Login Automatico

Il **login automatico** da parte di un utente è possibile se e solo il tempo trascorso dall'ultimo accesso dell'utente in questione è inferiore ai 15 minuti. Questa condizione è espressa in OCL attraverso una precondizione

```
context Utente_Autenticato :: login_automatico()
pre: tempoTrascorsoUltimoAccesso<= 15
```

3.2 Logout

Il **logout** da parte di un utente è possibile se e solo l'utente ha già effettuato il login. Questa condizione è espressa in OCL attraverso una precondizione

```
context  Utente_Autenticato :: logout()
pre: verifyLogin=true
```

3.3 Ricerca nel database

Affinché la ricerca possa avere luogo sul giusto database, il metodo **decide_database** e **chosen_database** devono avere lo stesso valore: vale a dire, se entrambi hanno valore **true**, allora vuol dire che l'informazione richiesta è da cercare nel database con cui si interfaccia la classe Informazioni; se entrambi sono **false**, allora va cercata nel database con cui si interfaccia la classe Docenti; mentre, se i due attributi sono diversi, non deve essere fatto nulla e deve essere riportato all'utente solo un generico errore. Le precedenti condizioni sono espresse con due precondizioni dal seguente codice OCL:

```
context Docente::effettua_ricerca(keywords)
pre: (decide_database(keywords)=false) AND
(chosen_database=false)

context Informazione::effettua_ricerca_info(keywords)
pre:(decide_database(keywords)=true) AND
(chosen_database=true)
```

3.4 Scelta preferenza notifiche

La scelta di attivare le notifiche o meno da parte deve essere permessa solamente nel caso in cui quest'ultimo abbia fatto l'accesso con le credenziali di ateneo, in modo da ottenere la sua mail a cui inviare i vari avvisi. Perciò il metodo presente in impostazioni, ossia **preferenza_notifiche,** deve essere usabile solo nel momento in cui l'attributo **is_online** è uguale a true. Il codice OCL è il seguente:

```
context Prefenze :: set_preferenza
pre : is_online=true
```

3.5 Invio delle mail

Il momento in cui inviare le mail di notifica da parte del sistema dipende da due attributi fondamentalmente **send_esame,** o **send_tassa,** e **scelta_preferenza.** Nel momento in cui entrambi sono true il sistema può attivare il metodo **gmail_esami,** o **gmail_tasse,** e far spedire la mail. Il codice OCL che rappresenta ciò è il seguente:

```
context Mail :: gmail_esami
pre : (Esse3.send_esame=true) AND
(Preferenze.scelta_preferenza=true)
```

```
context Mail :: gmail_tasse
pre : (Esse3.send_tassa=true) AND
(Preferenze.scelta_preferenza=true)
```

4. Diagramma delle classi completo

Riportiamo di seguito il diagramma delle classi unito e con tutti i codici OCL descritti nel capitolo precedente.