Immagine che contiene testo

Descrizione generata automaticamente

Dipartimento di Ingegneria e Scienza dell’Informazione

2022/2023

**YINCO**

**Documento di architettura**

|  |  |  |  |
| --- | --- | --- | --- |
| Doc. Name | D3 YINCO Documento di architettura | Doc. Number | Rev 0.1 |
| 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 il diagramma di contesto 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 l’attore “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 “**utenteAnonimo**” e “**utenteAutenticato**” 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.

Immagine che contiene testo, bigliettodavisita, screenshot

Descrizione generata automaticamente

**Figura 2.1: Specifica classi utente**

**2.2 Gestione autenticazione**

Lo use case diagram analizzato, presenta un ulteriore 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 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. Di seguito i dettagli di questa classe con i propri attributi e metodi.

Immagine che contiene testo

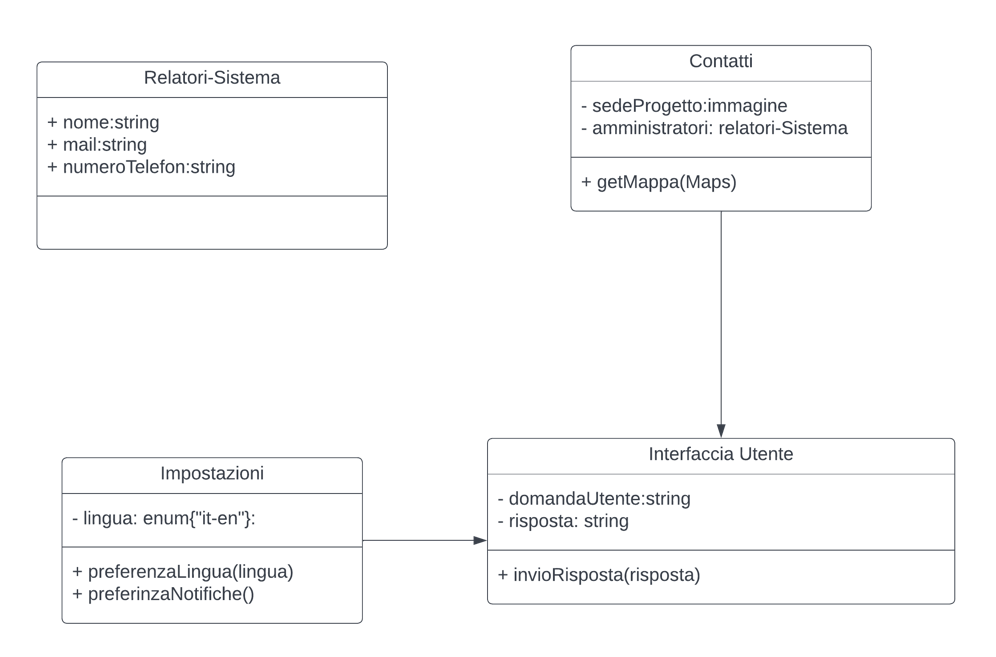
Descrizione generata automaticamente

**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. È usata per poter identificare il tipo dell’attributo “**amministratori**” della classe contatti.

(Manca la specifica dei metodi e attributi delle altre classi)



**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, c**reate\_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.

Immagine che contiene testo

Descrizione generata automaticamente

**Figura 2.4: Specifica delle classe Ricerca, Docente ed Informazione**

**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

Immagine che contiene testo

Descrizione generata automaticamente

**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

Immagine che contiene testo

Descrizione generata automaticamente

**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:

