

Dipartimento di Ingegneria e Scienza dell’Informazione

CORSO DI INGEGNERIA DEL SOFTWARE

****

Documento di progetto: Documento di Architettura

Gruppo: T33

**Indice**

[1. Scopo del documento 3](#_Toc118923077)

[2. Requisiti Funzionali 3](#_Toc118923078)

[2.1 Utente anonimo 3](#_Toc118923079)

[2.1.1 UML generale utente anonimo 6](#_Toc118923080)

[2.2 Utente studente 7](#_Toc118923081)

[2.2.1 UML generale utente studente 18](#_Toc118923082)

[2.3 Utente offerente 19](#_Toc118923083)

[2.3.1 UML generale utente offerente 30](#_Toc118923084)

[3. Requisiti Non Funzionali 31](#_Toc118923085)

[4. Analisi del Contesto 36](#_Toc118923088)

[3.1 Utenti e sistemi esterni 36](#_Toc118923089)

[3.2 Diagramma di contesto 37](#_Toc118923090)

[5. Analisi dei Componenti 39](#_Toc118923091)

[4.1 Definizione dei componenti 39](#_Toc118923092)

[4.2 Diagramma dei componenti 43](#_Toc118923093)

[6. Note ed eventuali 51](#_Toc118923094)

# Scopo del documento

Il presente documento riporta la definizione dell’architettura del progetto StayBusy attraverso l’utilizzo del diagramma delle classi in Unified Modeling Language (UML) e del codice in Object Constraint Language (OCL), che permettere di esprimere in modo formale e privo di ambiguità le regole che vengono applicate al diagramma UML e per descrivere anche la logica del software.

# Diagramma delle classi

Nel presente capitolo vengono elencate e descritte le varie classi previste nel progetto StayBusy. In particolare, ogni attore e sistema esterno rappresentato nel digramma di contesto, oltre che ad ogni componente presente nel diagramma dei componenti verranno ora rappresentati attraverso l’utilizzo di una o più classi, eventualmente associate tra loro. In questo caso, se necessario, sono state inserite anche delle informazioni aggiuntive, al fine di rappresentare al meglio le relazioni tra di esse.

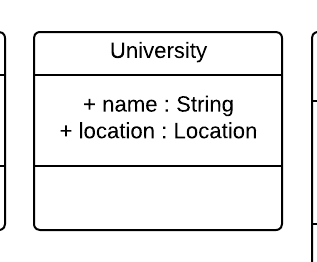
Nello specifico, di seguito vengono riportate le classi individuate dai diagrammi di contesto e delle componenti rappresentati nello scorso documento, ognuna delle quali è accompagnata da una breve descrizione di alcuni attributi e funzioni e, in alcuni casi, dal codice OCL in modo da poter rappresentare e descrivere i vincoli che sono presenti tra classi descritte.

**2.1 Tipi di dato**

In primo luogo, di seguito è presente una breve descrizione dei tipi di dato che sono stati introdotti per definire alcuni dati che vengono utilizzati dal sistema.

* UNIVERSITY

Durante la creazione del proprio account personale, l’utente deve inserire il nome della sede universitaria frequentata. Per questo motivo è stato creato un tipo di dato “University” caratterizzato da due attributi, ovvero dal *nome* dell’università di appartenenza e dalla sua *ubicazione*. Questo tipo di dato viene quindi utilizzato dal sistema per verificare l’effettiva esistenza dell’Ateneo indicato e per indirizzare lo studente alla pagina di accesso della propria università.



* TIME

Il tipo di dato “Time” è stato creato per il salvataggio del tempo, espresso dagli attributi *minutes* e *hours*. "Time" viene utilizzato dal sistema per esprimere l’orario di inizio e fine di un servizio. Nello specifico, è applicato sia nella classe " Disponibility" – in modo da salvare l’orario di disponibilità che uno studente imposta al momento della creazione dell’account e che può modificare in ogni istante – che nella classe "Announcement" – al fine di rappresentare l’orario di richiesta per un servizio specificato dall’utente offerente al momento di creazione dell’annuncio.



* DATE

Il tipo di dato “Date” è stato creato per il salvataggio della data, rappresentata attraverso gli attributi *day*, *month* e *year*. Così come per il tipo di dato *Time*, *Date* viene utilizzato nella classe " Disponibility" – in modo da salvare i giorni di disponibilità dell’utente studente per lo svolgimento di un servizio – e nella classe "Announcement "– al fine di rappresentare la data di richiesta per un servizio specificata dall’utente offerente al momento di creazione dell’annuncio.

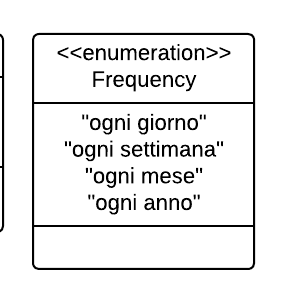
È stato deciso di non inserire direttamente un attributo *ora* di tipo “Time*”* all’interno del tipo di dato “Date” poiché l’orario di disponibilità e di richiesta è composto da *StartTime* – ovvero orario di inizio di un servizio – e *EndTime* – ovvero l’orario di conclusione di un servizio - che verranno espressi direttamente nelle classi "Disponibility*"* e "Annoucement*"*.

Immagine che contiene tavolo

Descrizione generata automaticamente

* FREQUENCY

Il tipo di dato “Frequency” è stato creato al fine di semplificare il processo di salvataggio delle disponibilità dell’utente studente. Questo tipo è un’enumerazione con quattro valori. In questo caso è stato preferito l’utilizzo di un’enumerazione data la sua chiarezza espositiva e la maggior agevolazione della modifica della classe nel caso si avesse intenzione di modificare/aggiungere attributi. In particolare, “Frequency” verrà utilizzato per permettere all’utente studente di non dover inserire le proprie disponibilità giorno per giorno, bensì di poter selezionare questa funzionalità che gli consente direttamente di selezionare il tipo di frequenza che il sistema deve utilizzare per salvare le sue fasce orarie. Per questo motivo, questo tipo di dato è utilizzato nella classe "Availability".



* OUTCOME

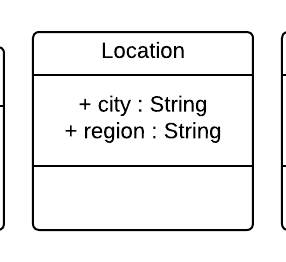
Il tipo di dato “Outcome” è stato creato per gestire lo stato delle richieste degli annunci. Questo tipo è un’enumerazione di tre valori - “Accettato”, “Respinto”, “In attesa” – attraverso cui l’utente può sapere qual è lo stato di una richiesta da lui effettuata per uno specifico servizio. Per questo motivo, questo tipo di dato è utilizzato nella classe "Candidate".

Immagine che contiene tavolo

Descrizione generata automaticamente

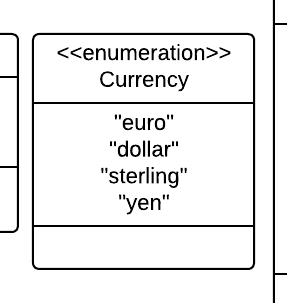
* LOCATION

Il tipo di dato “Location” è stato creato per permettere il salvataggio di un luogo ed è costituito da due attributi, ovvero dal nome della città e dalla sua regione di appartenenza. "Location" viene utilizzato sia dalla classe "University" per poter rappresentare l'ubicazione dell'Ateneo che dalla classe "Announcement" in modo da permettere il salvataggio del luogo in cui dovrà essere svolto il servizio richiesto.



* CURRENCY

Il tipo di dato “Currency” è stato creato per salvare la tipologia di pagamento che verrà effettuato al termine di un servizio. Questo tipo è un’enumerazione con quattro valori che rappresentano i vari tipi di pagamento che vengono forniti dall'applicazione. "Currency" viene dunque utilizzato dalla classe "Payment" che si occupa della gestione dei pagamenti.



CLASSI FUNZIONALI

Di seguito vengono presentate le classi che hanno la funzione di raccogliere e raggruppare i metodi riguardanti funzionalità cardine utilizzate da più classi.

* WEBPAGE

F

Immagine che contiene tavolo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

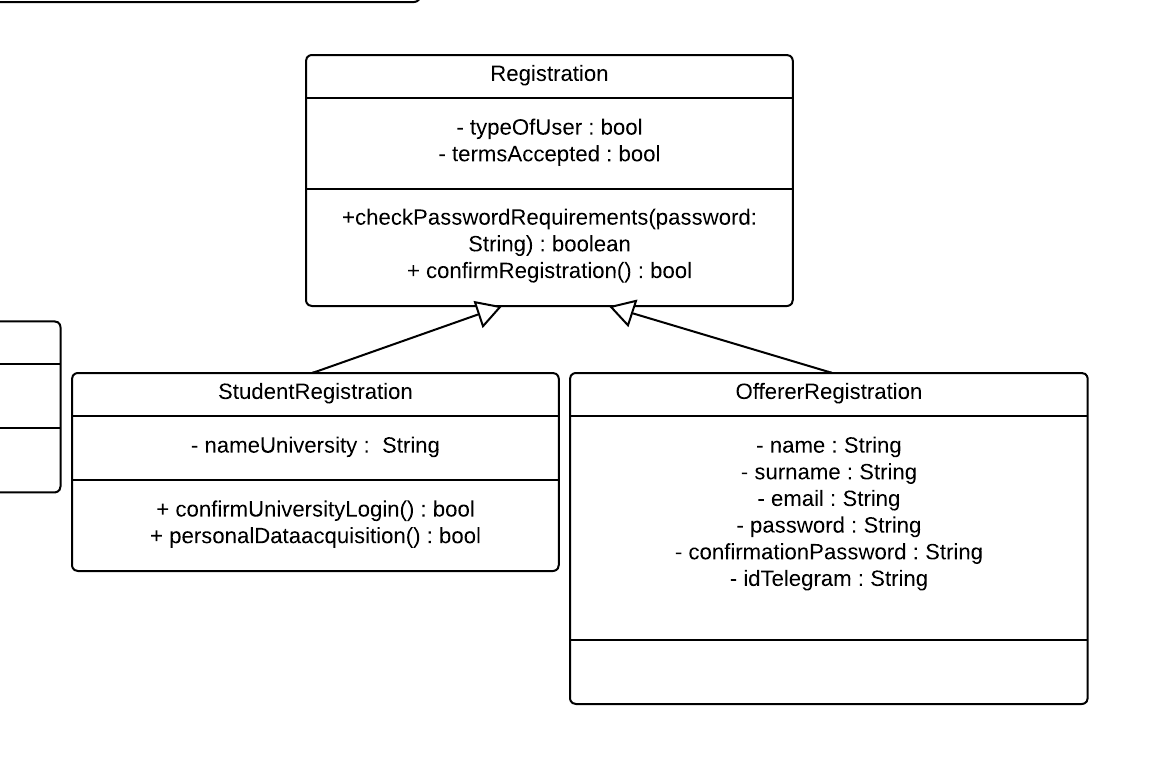
CLASSI INDIVIDUATE DAL DIAGRAMMA DI CONTESTO E DELLE COMPONENTI

Di seguito vengono presentate le classi derivare dall’analisi del diagramma di contesto e di quello delle componenti descritti nel documento precedente.

* GESTIONE REGISTRAZIONE

Dall’analisi delle componenti **5.2.4 Interfaccia login studente** e **5.2.14 Interfaccia login offerente**, si evince la necessità della creazione di una classe *Registration*. Questa gestisce la creazione di un nuovo account controllando la password inserita soddisfi i requisiti *Strong Password* che deve rispettare per essere accertata – che sono stati definiti nella sezione di sicurezza dei requisiti non funzionali del documento D1 - e che, conclusa la registrazione, l’utente ne effettui correttamente la conferma speditagli mediante il servizio di posta elettronica.

Nel documento D1 sono state distinte due tipologie di account – utente studente e utente offerente. Per questo motivo si è resa necessaria la creazione di due sottoclassi figlie della classe *Gestione Registrazione* che si distinguono per la tipologia di utente che si registra per utilizzare l’applicazione. In particolare, una delle sottoclassi individuate è *StudentRegistration,* sottoclasse che gestisce la richiesta della verifica delle credenziali effettuata dal componente **5.2.2 Credenziali universitarie studenti** per confermare la registrazione dell’account e che si occupa dell’acquisizione da questa componente dei dati personali utente e del loro salvataggio. Inoltre, la seconda sottoclasse di *Registration* è *OffererRegistration*, ovvero una classe che fornisce la corretta creazione dell’account e l’autenticazione nel sistema al componente **Home page offerente** e si occupa del salvataggio dei dati personali inseriti dall’utente offerente.



* GESTIONE AUTENTICAZIONE

Dall’analisi delle componenti **5.2.4 Interfaccia login studente** e **5.2.14 Interfaccia login offerente**, si evince la necessità della creazione di una classe *Login*. Questa classe si occupa di gestire tutte le funzionalità che l’utente può utilizzare una volta effettuati correttamente registrazione e accesso all’applicazione.

DA FARE

