Progetto di Ingegneria del Software 2

SWIMv2: Small World Hypothesis Machine v2

Autori: Professore:   
Bulla Jacopo Mottola Luca  
Caio Davide  
Cappa Stefano

Questo documento rappresenta la terza deliverable. Insieme ad esso sono stati forniti anche il RASD e il Project Planning aggiornati.

Lo scopo del Design Document (DD) è di mostrare in modo dettagliato la struttura del prodotto. Ovviamente, tutte le scelte effettuale nella stesura del precedente documento saranno rispettate e descritte in modo più preciso e a “basso livello”, in particolare il capitolo 2.3 del RASD, cioè quello sulle assunzioni e le scelte di progettazione. Perciò, saranno introdotti i concetti in modo incrementale dalla descrizione dell’architettura fino a mostrare il comportamento e funzionamento di SWIMv2, tramite apposti schemi.

Saranno utilizzati: UX Diagram, BCE, ER Model, EER Model, Logical Model e Diagrammi di Sequenza.

Indice

1. Introduzione 4

1.1. Errata Corrige 4

1.2. Definizioni, acronimi e abbreviazioni 5

2. Architettura 6

2.1. Presentation Tier 7

2.2. Business Tier 7

2.3. Data Tier 7

2.4. Design Pattern 7

3. Design 8

3.1. Progettazione concettuale: ER Model 9

3.2. Progettazione concettuale: EER Model 12

3.3. Progettazione logica: Logic Model 15

3.4. Modello di navigazione: UX Model 16

3.5. Altri diagrammi 19

4. Progetto per JEE 21

Indice delle figure 24

# Introduzione

## Errata Corrige

E’ stato aggiornato il RASD alla versione 1.2 per aggiungere una voce nel capitolo 2.3, per modificare il Class Diagram e per inserire l’indice delle numerose figure del documento.

Inizialmente non è stato specificato il modo con cui l’utente può proporre l’aggiunta di abilità nell’insieme generale di Abilità e nemmeno come può modificare quelle dell’insieme personale delle abilità.

Elemento aggiunto:

* **L’Utente può modificare il suo profilo (aggiungere/rimuovere Abilità dall’insieme personale delle Abilità e/o cambiare la foto del profilo) autonomamente senza nessuna autorizzazione da parte dell’Amministratore. Attenzione: l’Utente può aggiungere nuove abilità solo se presenti nell’insieme generale delle abilità. In caso contrario può proporne l’aggiunta all’Amministratore. La funzione di modifica profilo non è definita in modo chiaro nelle specifiche, ma è sottointesa dalla modifica delle Abilità. Il team di sviluppo ha deciso di estenderla anche alla modifica della foto (un’altra caratteristica aggiunta in fase di analisi dei requisiti per motivi estetici) per permettere maggiore personalizzazione del profilo dell’Utente.**
* **Amministratore**: **può ricevere proposte di aggiunta all’insieme generale delle abilità. Egli ha la possibilità di accettare o rifiutare la proposta. Inoltre, può aggiungere Abilità autonomamente all’Insieme generale delle Abilità. L’Amministratore è una singola persona, con un’Email e una Password, tramite le quali esegue il login da una pagina realizzata esclusivamente per lui, cioè la pagina di Login per l’Amministratore.**
* **Il sistema non prevede più Amministratori, poiché una volta completato l’insieme generale delle Abilità, le richieste di proposta Abilità diminuiranno e il carico di lavoro dell’Amministratore sarà proporzionato.** Di conseguenza, in futuro non saranno necessari altri Amministratori, a meno di non modificare notevolmente il database introducendo un sistema di sicurezza e suddividendo il carico di lavoro per la gestione di SWIMv2.

Modifica al Class Diagram:

* Aggiunti gli attributi “nome”, “descrizione”, “dataStipula” e “dataTermine” in Collaborazione e rimossi gli attributi “stipulato” e “isTerminata”.

## Definizioni, acronimi e abbreviazioni

* Design pattern: è "una soluzione progettuale generale a un problema ricorrente". é una descrizione o un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la progettazione e lo sviluppo del software.
* Browser: programma installato in un sistema operativo che permette l’accesso alla rete internet attraverso protocolli standard. Questo software mostra pagine web interpretando del codice ricevuto da un server. Esempi di codice con cui possono essere scritte sono: HTML, CSS, Javascript ecc…
* JEE: la Java Platform, Enterprise Edition o Java EE è una [piattaforma software](http://it.wikipedia.org/wiki/Piattaforma_(informatica)) di calcolo [Java](http://it.wikipedia.org/wiki/Linguaggio_Java). La piattaforma fornisce [API](http://it.wikipedia.org/wiki/API) e un ambiente di [runtime](http://it.wikipedia.org/wiki/Runtime" \o "Runtime) per lo sviluppo e l'[esecuzione](http://it.wikipedia.org/wiki/Esecuzione_(informatica)) di [software](http://it.wikipedia.org/wiki/Software) per le imprese, compresi i servizi di [rete](http://it.wikipedia.org/wiki/Rete_(informatica)) e [web](http://it.wikipedia.org/wiki/Web), e di altre grandi [applicazioni](http://it.wikipedia.org/wiki/Applicazione) di rete a più livelli, scalabile, affidabile e sicuro. La Java EE estende la [Java 2 Platform, Standard Edition](http://it.wikipedia.org/wiki/Java_2_Platform,_Standard_Edition) (Java SE). La piattaforma incorpora un design basato in gran parte su componenti modulari in esecuzione su un [server di applicazioni](http://it.wikipedia.org/wiki/Application_server). Il software per Java EE è principalmente sviluppato in [linguaggio di programmazione](http://it.wikipedia.org/wiki/Linguaggio_di_programmazione) [Java](http://it.wikipedia.org/wiki/Linguaggio_Java).



* EJB: gli Enterprise JavaBean (EJB) sono i componenti che implementano, lato [server](http://it.wikipedia.org/wiki/Server), la [logica di business](http://it.wikipedia.org/wiki/Business_logic) all'interno dell'architettura [Java EE](http://it.wikipedia.org/wiki/Java_EE). Le specifiche per gli EJB definiscono diverse proprietà che questi devono rispettare, tra cui la persistenza, il supporto alle transazioni, la gestione della [concorrenza](http://it.wikipedia.org/wiki/Concorrenza_(informatica)) e della [sicurezza](http://it.wikipedia.org/wiki/Sicurezza_informatica) e l'integrazione con altre tecnologie, come [JMS](http://it.wikipedia.org/wiki/Java_Message_Service), [JNDI](http://it.wikipedia.org/wiki/JNDI), e [CORBA](http://it.wikipedia.org/wiki/CORBA). Lo standard attuale, EJB 3, differisce notevolmente dalle versioni precedenti; è stato completato nella primavera del [2006](http://it.wikipedia.org/wiki/2006)
* Session Bean: nella Java Platform è un tipo di Enterprise Bean. Una Session Bean implementa una business task ed è ospitata in un EJB container.

* Entity Bean: è un tipo di Enterprise JavaBean, un componente J2EE server-side che rappresenta dati persistenti gestiti in un database.

# Architettura

Nel RASD non è stato specificato come sarà realizzato il lato client, ma **il team di sviluppo ha proposto queste due possibili soluzioni:**

* **Un’applicazione web, alla quale l’Utente potrà accedervi con un qualunque browser.**
* **Un’applicazione Java da scaricare ed installare sul proprio computer.**

Dopo un’accurata analisi, sono stati evidenziati i seguenti punti fondamentali per la scelta:

1. Un social network, per definizione, deve operare solo attraverso il web senza avere accesso al computer locale, a meno di possibili funzioni di upload/download.
2. Gli utenti sono distribuiti su molti sistemi operativi, comprese diverse distribuzioni Linux (anche se in modo meno rilevante)
3. Gli utenti di “smartphone” e “tablet” sono cresciuti così tanto da diventare un punto di riferimento nella realizzazione di un social network. Purtroppo, questi dispositivi usano sistemi operativi molto differenti, non supportano tutti gli stessi software/linguaggi di programmazione e spesso richiedono strumenti di sviluppo e requisiti differenti.

Possibili soluzioni ai punti elencati sopra:

1. **Realizzare un’applicazione web** accessibile tramite browser.
2. **Rendere indipendente il lato client dal sistema operativo (es. tramite un’applicazione Java)**
3. Sfruttare il browser (seguendo certi standard) per **rendere il lato client compatibile sui principali dispositivi mobili.**

Di conseguenza, **la migliore soluzione è la prima, cioè realizzare l’interfaccia come un’applicazione web, alla quale l’utente potrà accedervi con un qualunque browser, indipendentemente dal sistema operativo e dal tipo di dispositivo (mobile o fisso).**

**L’architettura del sistema è quella chiama Three-Tier** (“a tre strati/livelli”). Grazie ad essa vi è una separazione tra l’interfaccia utente (Presentation Tier), la logica funzionale (Business Tier) e la persistenza dei dati (Data Tier) in più strati. Essi possono essere sullo stesso server oppure su più macchine.

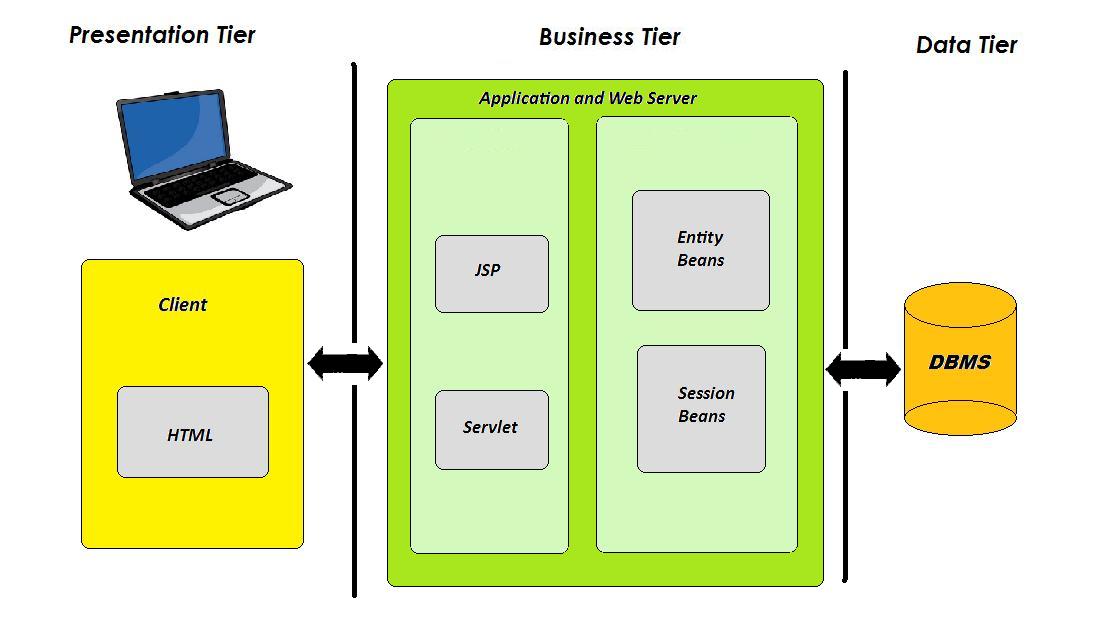


Fig. . – Architettura Three-Tier

## Presentation Tier

Strato che si occupa della presentazione, cioè della visualizzazione dei dati. Essi sono mostrati tramite un’interfaccia grafica per far sì che l’utente possa interagire col sistema, tramite funzioni predefinite e semplici da usare. Alcune delle tecnologie utilizzate per raggiungere tale obiettivo sono JSP, HTML, JavaScript, e JSF.

## Business Tier

Gestisce la logica, cioè fornisce l’accesso al database ed è lo strato più importante del sistema, perché costituisce il cuore della logica di business. La tecnologia utilizzata è EJB, poiché costituisce un requisito di progetto fornito dal committente.

## Data Tier

Strato che si occupa di gestire la persistenza dei dati, cioè il database. Il team di sviluppo ne ha scelto uno di tipo relazionale, cioè MySQL.

## Design Pattern

La suddivisione tra visualizzazione (interfaccia utente), controllo (fornisce informazioni alla GUI, gestisce la logica e utilizza i dati) e modello (rappresenta i dati) permette di ricondursi al Design Patter MVC (Model – View – Controller).

Lo schema seguente (Fig. 2.2) mostra le interazioni dirette tra i vari moduli software e quelle indirette (frecce sottili). Le prime sono effettuate tramite associazioni dirette (richieste), le seconde tramite quelle indirette, spesso con Design pattern.

**CONTROLLER**

**MODEL**

**VIEW**

Fig. . – Architettura Three-Tear

# Design

Questo capitolo descrive la fase di design, cioè la progettazione dell’intero prodotto.

Inizialmente, sarà mostrata quella concettuale, cioè la definizione ad alto livello della struttura del database. In seguito, verrà mostrata la progettazione logica, sempre riferita al database.

Terminata la realizzazione della persistenza dei dati ad alto livello, il team di sviluppo si è concentrato sul modello di navigazione. L’obiettivo è di scendere ad un livello più basso e dettagliato della descrizione del sistema in termini di visualizzazione/navigazione delle pagine web.

Per fornire una descrizione più dettagliata e per legarsi alla divisione in tre livelli descritta nel capitolo precedente, il team di sviluppo ha deciso di adottare anche i diagrammi di analisi (BCE).

Solo dopo a tutte queste fasi e dopo aver definito nel dettaglio ogni comportamento del sistema, sarà mostrata la progettazione di SWIMv2 basata su JEE (rispettando il vincolo imposto dal committente), cioè definendo gli Entity Bean e i Session Bean.

Infine, per una maggiore chiarezza ed evitare interpretazioni errate di alcune scelte di progettazione del team di sviluppo, sono stati allegati alcuni Diagrammi di Sequenza.

**Attenzione: è ovvio che ogni passo della fase di design debba rispettare tutte le scelte progettuali effettuate nei precedenti documenti, in particolare il RASD e il capitolo sulle assunzioni (vedi 2.3).**

## Progettazione concettuale: ER Model

La progettazione concettuale di un database consiste nell’individuazione delle entità e delle relazioni.

Essendo il livello più alto con cui è rappresentata una base di dati, è facilmente leggibile ed interpretabile.

Per comprenderne il significato sono necessarie alcune informazioni sulla notazione usata:

* Entità: identifica uno o più oggetti o classe con caratteristiche comuni. Nello schema ER è rappresentata come un rettangolo con all’intero un nome.
* Relazione: identifica un legame tra due o più entità. E’ rappresentata come un rombo con all’interno un verbo o un sostantivo, collegato alle entità che deve “legare”.
* Attributo: elemento contenuto in un’entità. Nello schema ER è rappresentato come un’ellisse/cerchio collegato ad una entità ed un nome.
* Attributo chiave: attributo che identifica univocamente gli elementi appartenenti alla stessa entità. Spesso è indicato come un cerchietto nero o col nome sottolineato.

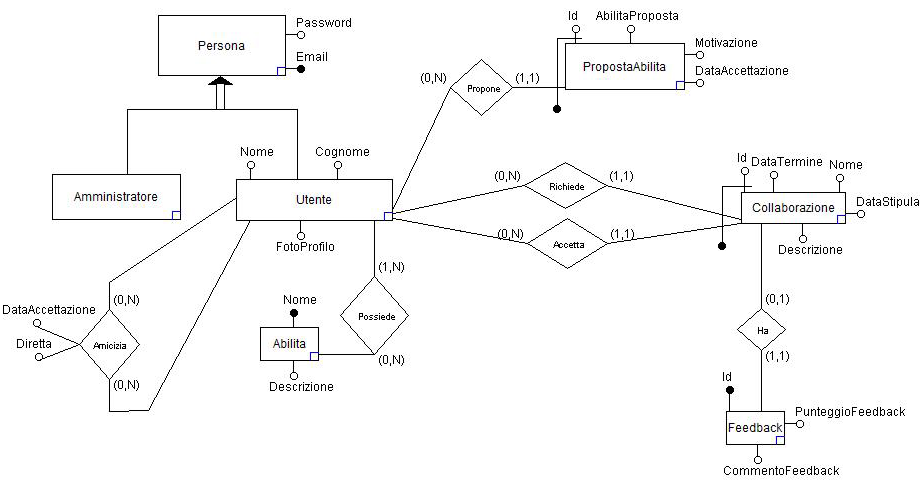


Fig. . - Schema ER

* + 1. Entità

**Persona**

Entità generica che non sarà rappresentata negli schemi successivi. L’unico scopo è di esprimere che “Utente” e “Amministratore” condividono alcuni attributi.

Essa è identificabile grazie all’Email, che per definizione è univoca. Per tale motivo è stata scelta come unica chiave primaria (indicata sullo schema ER con il cerchio nero).

Gli attributi sono:

* *Email*: chiave primaria
* *Password*: parola chiave che permette il Login, insieme all’Email.

**Utente**

Entità che rappresenta l’utilizzatore registrato del sistema ed estende l’entità Persona. **E’ importante notare che la classe Visitatore non è stata rappresentata nel database, in quanto non è necessario memorizzare i suoi dati.**

Gli attributi sono:

* *Cognome*: cognome reale dell’Utente. Esso sarà utilizzato per eseguire le ricerche di utenti.
* *Nome*: nome reale dell’Utente. Esso sarà utilizzato per eseguire le ricerche di utenti.
* *FotoProfilo*: Immagine che l’Utente può caricare in fase di registrazione e/o modificare in seguito tramite la funzione di modifica del profilo.

**Amministratore**

Entità senza attributi che estende Persona. **Si è scelto di non rappresentare la relazione di “dichiarazione” delle Abilità, perché “Amministratore” è un’entità che ha accesso a tutto il database ed inoltre non vi è alcune necessità di associare l’Email all’Abilità dichiarata inizialmente o approvata in seguito.**

**Collaborazione**

Entità che rappresenta la collaborazione tra due Utenti, cioè il rapporto di cooperazione che costituisce la richiesta di aiuto.

Poiché vi possono essere moltissime Collaborazioni, soprattutto in vista di un’estensione del sistema, si è scelto di assegnare un identificativo univoco progressivo.

Gli attributi sono:

* *Id*: identificativo progressivo univoco che costituisce la chiave primaria insieme alle relazioni con l’entità Collaborazione.
* *Nome*: attributo che rappresenta il nome della Collaborazione. Esso è utilizzato solo per mostrare la lista delle collaborazioni e renderle facilmente identificabili.
* *Descrizione*: descrizione della Collaborazione, cioè una sorta d’invito ad accettare che l’Utente può fare verso quello che possiede le abilità ricercate.
* *DataStipula*: data che rappresenta l’istante temporale di apertura o stipula della Collaborazione, cioè nel momento in cui l’Utente ricevente risponde in modo positivo alla richiesta.
* *DataTermine*: data che rappresenta l’istante temporale di fine Collaborazione, cioè quando l’Utente ritiene sia il momento di terminare il rapporto con l’altro Utente per lasciare, eventualmente, un Feedback

**Feedback**

Entità che rappresenta il commento di Feedback associato alla Collaborazione.

Gli attributi sono:

* *Id:* **Identificativo univoco del Feedback che corrisponde all’Id della Collaborazione.**
* *CommentoFeedback*: eventuale commento dell’Utente richiedente per spiegare meglio l’esperienza avuta durante la Collaborazione con l’Utente ricevente.
* *PunteggioFeedback*: valore intero da 1 a 5 che rappresenta il voto. Esso sarà utilizzato per calcolare il punteggio di Feedback dell’Utente ricevente.

**Abilita**

Entità che rappresenta l’Abilità, cioè la capacità di svolgere una determinata mansione o di conoscere un certo argomento. E’ una parte fondamentale del sistema, perché tramite essa avviene la ricerca di aiuto, e rende identificabili le conoscenze che ha un particolare Utente.

**Attenzione: l’attributo Descrizione di Abilità potrà essere utilizzato solo dall’Amministratore al momento dell’accettazione e non dall’Utente.**

Gli attributi sono:

* *Nome*: nome dell’Abilità che costituisce la chiave primaria.
* *Descrizione*: eventuale descrizione dell’Abilità. Attributo opzionale che può essere utile per spiegare la differenza di alcune Abilità, che potrebbero avere nomi ambigui o facili da confondere.

**PropostaAbilita**

Entità che rappresenta la proposta di un Utente di aggiungere un’Abilita nell’insieme generale delle Abilità. Quando il sistema crea la proposta di Abilita su richiesta dell’Utente, non viene generata una nuova Abilità, ma il sistema ne richiede la creazione all’Amministratore. Egli può scegliere se confermarla, inserendola nell’insieme generale delle abilità e di conseguenza il sistema creerà l’Abilità associata.

Questo significa che **non ci sarà una relazione diretta** tra PropostaAbilita e Abilita.

Gli attributi sono:

* *Id*: identificativo progressivo univoco che costituisce la chiave primaria insieme alla relazione con l’entità Utente.
* *AbiltaProposta*: nome dell’abilità proposta. E’ una semplice Stringa e non costituisce nessuna chiave esterna.
* *Motivazione*: eventuale commento dell’Utente per motivare la proposta di aggiunta.
* *DataAccettazione*: indica la data di accettazione della proposta d’abilità da parte dell’Amministratore. Se assume valore NULL significa che la proposta è ancora in attesa di una risposta. Se l’Amministratore la rifiuta, il sistema rimuove l’informazione da PropostaAbilità

* + 1. Relazioni binarie
* **Amicizia** l’Utente “è amico di” da 0 a N Utenti (amici), mentre l’Utente (amico) è a sua volta “amico di” da 0 a N Utenti. Cioè ogni Utente ha da 0 a N amici e questi amici hanno a loro volta da 0 a N amici.

Questa relazione ha i seguenti attributi:

* + *DataAccettazione*: indica la data di accettazione dell’amicizia. Se assume valore NULL significa che la richiesta è ancora in attesa di una risposta. Quando l’utente rifiuta una richiesta di amicizia, il sistema rimuove l’informazione sull’Amicizia.
  + *Diretta*: Indica se è un’amicizia stretta in modo diretto, su richiesta di un utente, o indiretta, su proposta del sistema.
* **Possiede**: l’Utente “possiede” da 1 a N Abilità, mentre l’Abilità “è posseduta” da 0 o N Utenti.
* **Richiede**: l’Utente “inizia” da 0 a N Collaborazioni, mentre la Collaborazione “è iniziata” da un solo Utente. **Questa relazione identifica l’Utente *richiedente*.**
* **Accetta:** l’Utente “partecipa” da 0 a N Collaborazioni, mentre la Collaborazione “è composta” da un solo Utente. **Questa relazione identifica l’Utente *ricevente*.**
* **Ha:** la Collaborazione può “avere” un Feedback, mentre il Feedback, se esiste, “è riferito” ad una ed una sola Collaborazione.
* **Propone:** l’Utente “propone” da 0 a N ProposteAbilità, mentre la PorpostaAbilità “è proposta” da uno ed un solo Utente.

## Progettazione concettuale: EER Model

Durante la progettazione concettuale, il team si sviluppo ha scelto di realizzare anche l’EER (Enhanced ER o Extended ER) che “estende” l’ER Model.

In questo schema sono messe in evidenza le cardinalità delle relazioni tramite una simbologia differente rispetto l’ER. Inoltre, **questo modello mostra più nel dettaglio le entità, perché indica il tipo degli attributi e le chiavi esterne.**

Simboli delle entità:

* Chiave gialla: indica la chiave primaria
* Rombo rosso: indica una chiave esterna (non utilizzato)
* Rombo bianco: attributo che può essere NULL
* Rombo azzurro: attributo che non può essere NULL

Le relazioni si distinguono in:

* “Nonidentifying Relationship” quando la chiave esterna non fa parte della chiave primaria della tabella figlia (non utilizzata in questo schema).
* “Identifying Relationship” quando la chiave esterna fa parte della chiave primaria nella tabella figlia.

Cardinalità delle relazioni:

* Machintosh HD:Users:Ks89:Dropbox:Ing Sof 2:Bulla - Caio - Cappa:3 - DD:3 - EER:1.png Indica una cardinalità “1”
* Machintosh HD:Users:Ks89:Dropbox:Ing Sof 2:Bulla - Caio - Cappa:3 - DD:3 - EER:2.png Indica una cardinalità “N”

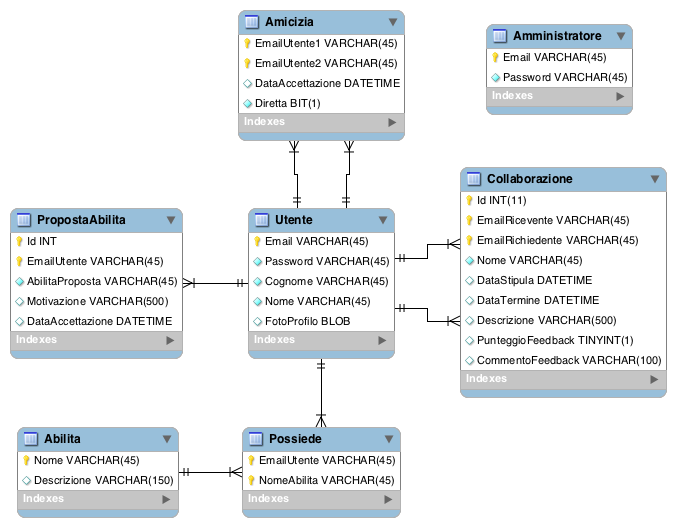


Fig. . - EER Model

Una differenza fondamentale dallo schema ER precedente è la “fusione” di Feedback in Collaborazione, poiché ad una Collaborazione è associato al massimo un solo Feedback. Quindi, si può aggiungere nella Collaborazione i due attributi “PunteggioFeedback” e “CommentoFeedback” dell’entità Feedback. Nel caso esso non sia stato rilasciato, assumeranno valori NULL, altrimenti vi sarà associato il punteggio e l’eventuale commento. In seguito saranno spiegati meglio tutti gli attributi di questa entità.

Ora segue la spiegazione dettagliata di ogni entità dello schema EER e dei loro attributi.

**Amministratore**

Entità che rappresenta l’Amministratore, cioè una persona unica che si occupa di gestire il social network.

Poiché “eredita” tutti gli attributi di Persona (nello schema ER), avrà i seguenti attributi:

* *Email*: chiave primaria che costituisce, insieme alla Password, le informazioni di Login dell’Amministratore.
* *Password*: parola chiave che permette il Login, insieme all’Email. Ovviamente non può valere NULL.

Come già specificato nello schema ER, **si è scelto di non rappresentare nessuna relazione con l’Amministratore perché è un’entità che “vede” e “opera” sull’intero sistema, senza che nessuna sua informazione sia tracciabile. In futuro sarà comunque possibile modificare il database per introdurre alcuni sistemi di sicurezza.**

**Utente**

Come già anticipato, rappresenta l’Utente, cioè chi si è registrato a SWIMv2.

Non vi è alcuna differenza con lo schema ER a parte il fatto che “eredita” gli attributi da Persona, proprio come l’Amministratore. Per maggiore completezza sono riportati tutti gli attributi dell’entità.

Attributi:

* *Email*: chiave primaria che costituisce, insieme alla Password, le informazioni di Login dell’Utente.
* *Password*: parola chiave che permette il Login, insieme all’Email. Ovviamente non può valere NULL.
* *Cognome*: cognome reale dell’Utente. Esso sarà utilizzato per eseguire le ricerche di utenti. Ovviamente non può valere NULL.
* *Nome*: nome reale dell’Utente. Esso sarà utilizzato per eseguire le ricerche di utenti. Ovviamente non può valere NULL.
* *FotoProfilo*: Immagine che l’Utente può caricare in fase di registrazione e/o modificare in seguito tramite la funzione di modifica del profilo. Tale attributo è stato rappresentato tramite il tipo BLOB, perché indica un file.

**Possiede**

Entità che indica quanto un’Abilità è presente nell’insieme personale delle Abilità di un Utente.

La chiave primaria è costituita da entrambi gli attributi.

Attributi:

* *EmailUtente*: chiave esterna, parte della chiave primaria, che indica quale Utente possiede una certa Abilità. Ovviamente, tale attributo si riferisce a “Email” della tabella “Utente”.
* *NomeAbilita*: chiave esterna, parte della chiave primaria, che indica il nome dell’Abilità posseduta da un Utente. Ovviamente, tale attributo si riferisce a “Nome” della tabella “Abilità”.

**Abilità**

Entità che rappresenta la capacità di svolgere una determinata mansione o di conoscere un certo argomento. Tramite essa avviene la ricerca di aiuto e rende identificabili le conoscenze che ha un particolare Utente. **Attenzione: l’attributo Descrizione di Abilità potrà essere utilizzato solo dall’Amministratore al momento dell’accettazione e non dall’Utente.**

Attributi:

* *Nome*: chiave primaria che rappresenta il nome dell’Abilità, cioè quella stringa da selezionare durante l’aggiunta di un’abilità nell’insieme personale/generale delle Abilità o durante la ricerca di aiuto.
* *Descrizione*: eventuale descrizione dell’Abilità. Attributo opzionale che può essere utile per spiegare la differenza di alcune Abilità, che potrebbero avere nomi ambigui o facili da confondere.

**PropostaAbilita**

E’ la stessa entità dello schema ER con l’aggiunta dell’Email dell’Utente che propone un’Abilità all’Amministratore.

Attributi:

* *Id*: identificativo progressivo univoco che costituisce la chiave primaria insieme all’attributo EmailUtente.
* *EmailUtente*: identificativo dell’Utente che ha proposto l’Abilità. Tale attributo è la chiave esterna verso l’Email Utente ed insieme all’Id costituisce la chiave primaria.
* *AbiltaProposta*: nome dell’abilità proposta. E’ una semplice Stringa e non costituisce nessuna chiave esterna. Ovviamente, non potrà mai essere NULL.
* *Motivazione*: eventuale commento dell’Utente per motivare la proposta di aggiunta. Questo attributo può assumere valore NULL.
* *DataAccettazione*: rappresenta la data di accettazione della proposta. Quest’attributo è NULL quando non è ancora stata accettata dall’Amministratore. **Se la proposta venisse rifiutata, l’intera tupla verrebbe cancellata e di conseguenza non è necessario utilizzare un attributo “rifiutata”.**

**Amicizia**

E’ la stessa entità dello schema ER con l’aggiunta delle Email dei due Utenti che stringono amicizia, perché sono chiavi esterne.

Attributi:

* *EmailUtente1*: identificativo dell’Utente che richiede amicizia. Tale attributo è la chiave esterna verso l’Email Utente ed insieme all’altro EmailUtente costituisce la chiave primaria.
* *EmailUtente2*: identificativo dell’Utente a cui è proposta l’amicizia. Tale attributo è la chiave esterna verso l’Email Utente ed insieme all’altro EmailUtente costituisce la chiave primaria.
* *DataAccettazione*: rappresenta la data di accettazione della richiesta di amicizia. Questo attributo è NULL quando non è ancora stata accettata. **Se l’amicizia venisse rifiutata, l’intera tupla sarà cancellata e di conseguenza non è necessario utilizzare un attributo “rifiutata”.**
* *Diretta*: attributo che assume valore ‘1’ se l’amicizia è stata richiesta in modo diretto,‘0’ in modo indiretto.

**Collaborazione**

E’ la stessa entità dello schema ER precedente. La chiave primaria è costituita dall’insieme dell’Id della Collaborazione e le due Email degli utenti che vi partecipano. Questi ultimi due attribuiti sono anche chiavi esterne, collegate ad Email dell’entità Utente.

Gli attributi sono:

* *Id*: identificativo progressivo univoco che costituisce la chiave primaria insieme alle Email dei due utenti partecipanti
* *EmailRichiedente*: rappresenta l’email dell’utente che ha avviato la collaborazione, essa è una chiave esterna collegata alla tabella Utente, che insieme all’id e alla email dell’altro utente costituisce la chiave primaria.
* *EmailRicevente*: : rappresenta l’email dell’utente che ha accettato la collaborazione, cioè colui che fornisce l’aiuto. Essa è una chiave esterna collegata alla tabella Utente, che insieme all’id e alla email dell’altro utente costituisce la chiave primaria.
* *Nome*: attributo che rappresenta il nome della Collaborazione. Esso è utilizzato solo per mostrare la lista delle collaborazioni, all’interno della sezione Utente, e renderle facilmente identificabili.
* *Descrizione*: descrizione della Collaborazione, cioè una sorta d’invito ad accettare che l’Utente può fare verso quello che possiede le abilità ricercate. Quest’attributo può essere NULL.
* *DataStipula*: vale NULL se la collaborazione non è ancora stata stipulata (non confermata), invece assume come valore una data, nel caso in cui sia confermata dall’Utente (data del momento dell’accettazione).
* *DataTermine*: vale NULL se la collaborazione non è ancora stata terminata (in corso/ancora da confermare o rifiutata), invece assume come valore una data, nel caso in cui l’Utente che l’ha aperta decida di chiuderla (data del momento della chiusura/termine della Collaborazione).
* *PunteggioFeedback*: valore intero da 1 a 5 che rappresenta il voto. Esso sarà utilizzato per calcolare il punteggio di Feedback dell’Utente ricevente. Nel caso in cui il valore assuma valore NULL è perché il Feedback non è ancora stato rilasciato. Questo attributo potrà essere solo NULL finche la DataTermine sarà uguale a NULL.
* *CommentoFeedback*: eventuale commento dell’Utente richiedente per spiegare meglio l’esperienza avuta durante la Collaborazione con l’Utente ricevente. Tale attributo assume valore NULL nel caso in cui il Feedback non sia stato rilasciato, oppure nel caso in cui il commento non sia stato fornito dall’Utente nel momento del rilascio. Questo attributo potrà essere solo NULL finche la DataTermine e PunteggioFeedback saranno uguali a NULL.

## Progettazione logica: Logic Model

La progettazione logica è una possibile traduzione dello schema ER. Essa mostra sempre le entità con gli attributi e le chiavi, ma si concentra soprattutto sul descrivere i collegamenti concettuali con altre tabelle.

Nello schema seguente sono indicate in giallo le “primary key” (chiavi primarie) e in rosso le “foreign key” (chiavi esterne). In realtà tutte le “foreign key” fanno parte di chiavi primarie, quindi non sono state evidenziate in rosso, ma si è scelto di indicarle in giallo per mettere in maggior risalto le primary key.

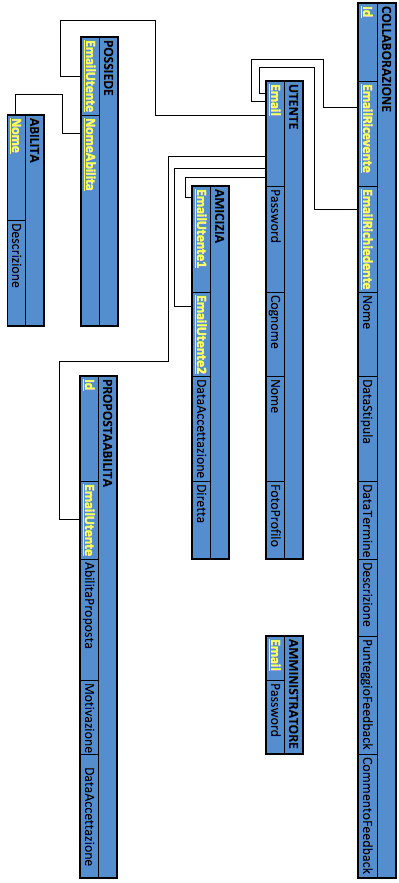


Fig. . – Schema Logico

## Modello di navigazione: UX Model

Un modello di navigazione, detto User eXperience Model (“UX Model”), rappresenta la navigazione tra le pagine di SWIMv2. Ci sono <<screen>>, <<input form>>, aggregazioni, composizioni e metodi**.**

Gli elementi come <<screen>>, <<input form>> hanno nome, attributi e metodi.

Il simbolo “$” indica che la <<screen>> è raggiungibile in qualunque momento durante la navigazione, mentre il simbolo “+” rappresenta una lista di elementi con gli attributi specificati nella classe oppure all’esterno tramite la relazione di aggregazione (rombo vuoto).

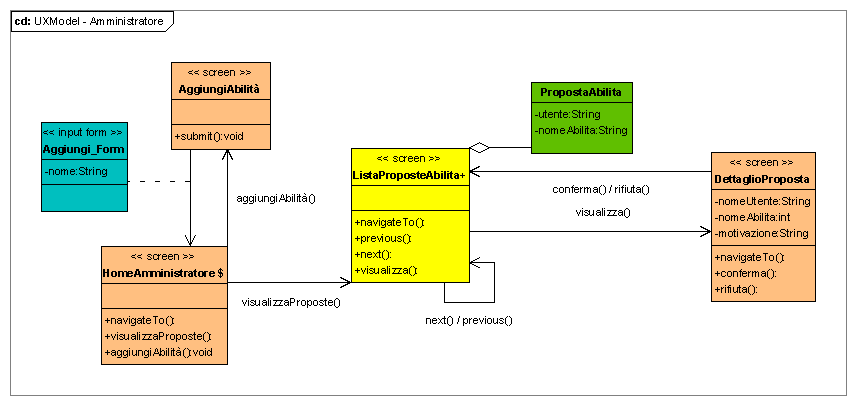


Fig. . - UXModel Amministratore

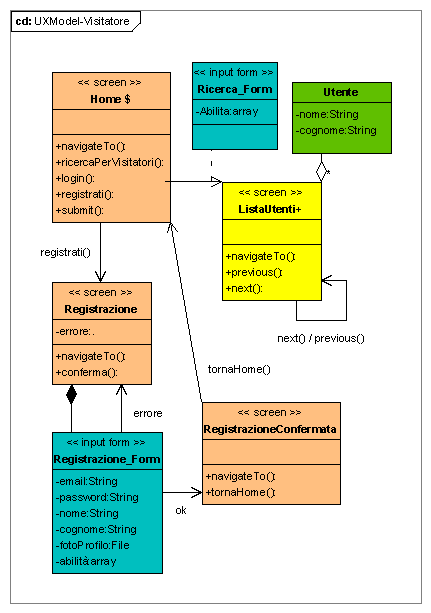


Fig. . - UXModel Visitatore

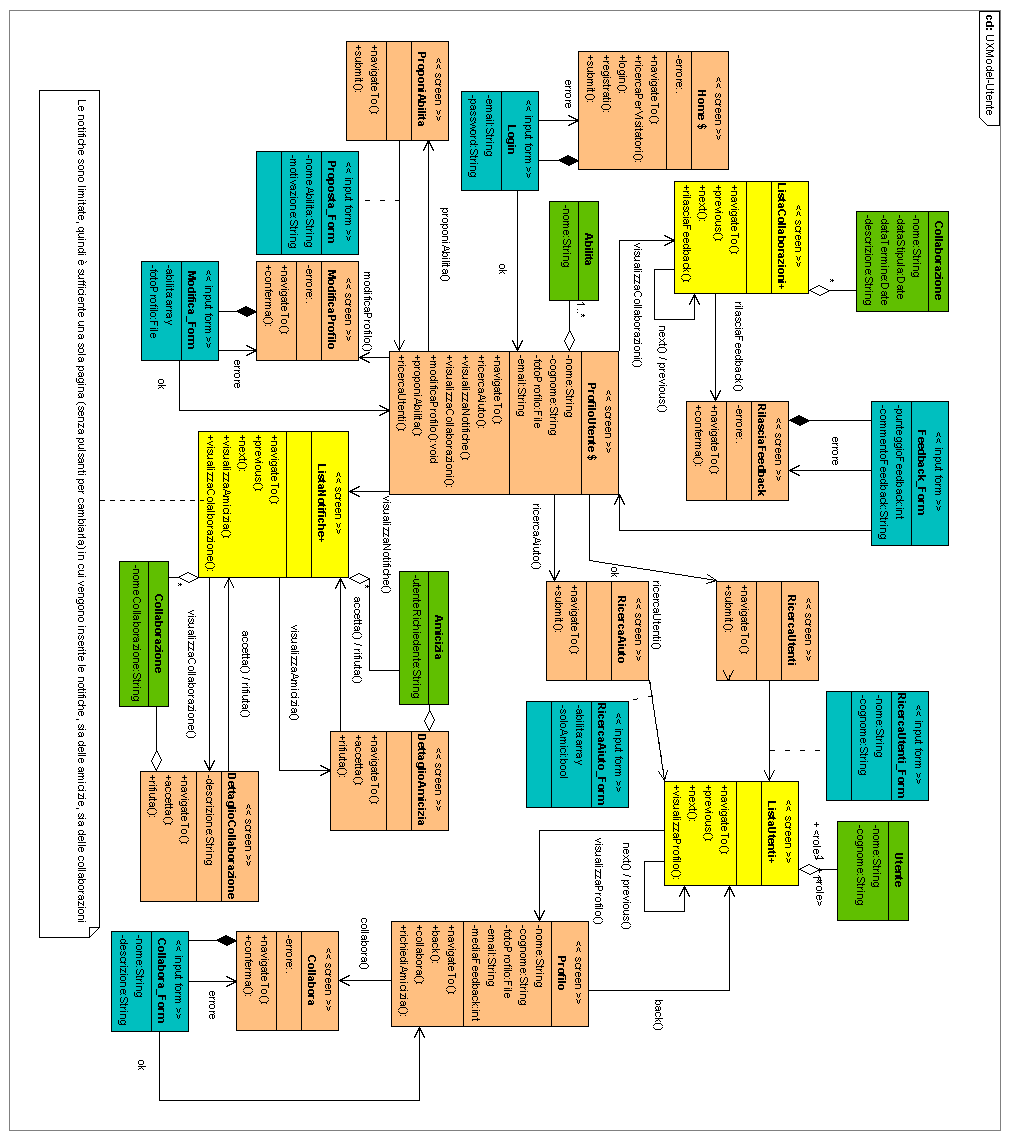


Fig. . - UXModel Utente

## Altri diagrammi

* + 1. Diagramma di analisi (BCE)

Il diagramma Boundary-Control-Entity divide il sistema in presentazione (boundary), controllo (control) e dati (entity).

Questi diagrammi usano il seguente standard:

* Classi: rappresentate come rettangoli con il tipo (<<control>>, <<entity>> o <<boundary>>), il nome, gli attributi e i metodi.
  + <<boundary>>: classe che fa parte dell’interfaccia utente
  + <<entity>>: classe che rappresenta la persistenza dei dati
  + <<control>>: classe che rappresenta la logica del sistema
* Relazioni: associazioni tra classi specificando le cardinalità
* Frecce: indicano

A seguire lo schema completo.

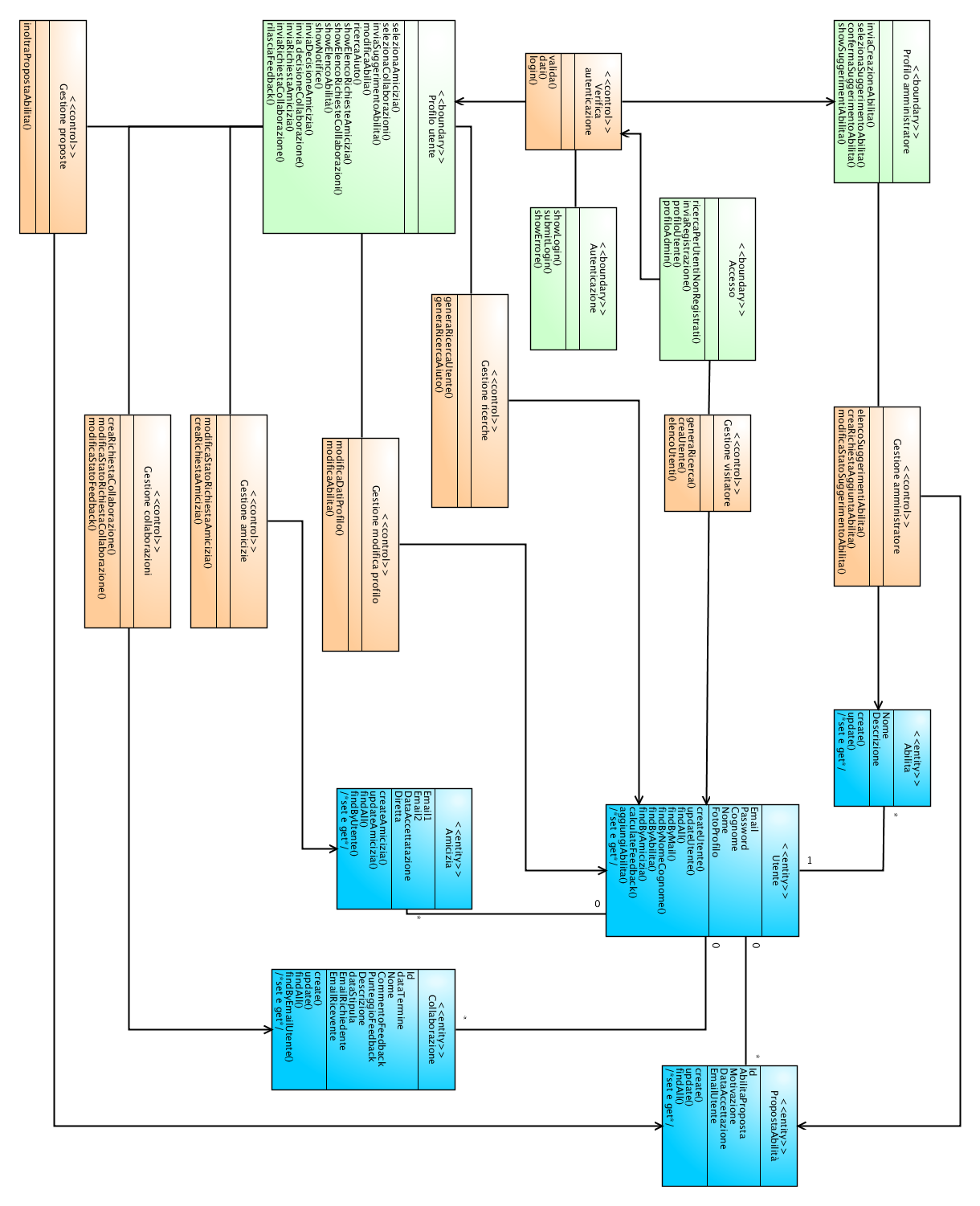


Fig. . - BCE Model

# Progetto per JEE

Dopo aver progettato il sistema ad alto livello, è necessario scendere in quello più basso in cui sono specificati gli elementi che saranno implementati direttamente tramite JEE.

Prima sono definiti gli “Entity Beans” per rappresentare i dati e in seguito i “Session Bean” che rappresentano i controlli.

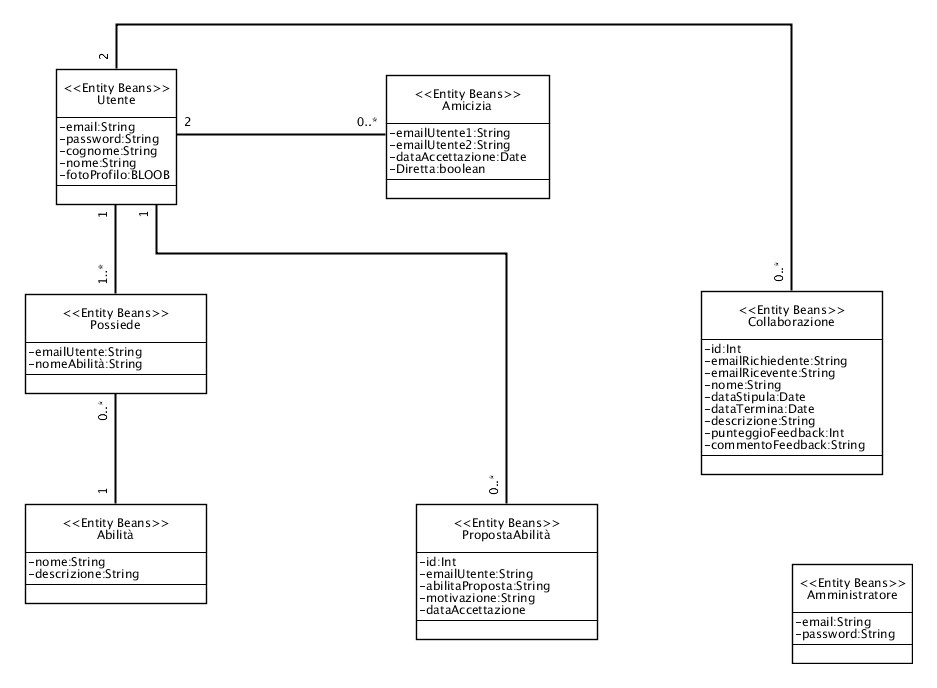


Fig. . - EntityBeans

Di solito, i “Session Bean” coincidono con i <<control>> del diagramma BCE, compresi i metodi.

Un Session Bean può essere di tipo Stateless o Stateful. Sotto sono specificati i nomi e i tipi.

* GestioneRegistrazione (stateless)
* GestioneLogin (stateful)
* GestioneAmministratore (stateless)
* GestioneModificaProfilo (stateless)
* GestioneRicerche (stateless)
* GestioneProposte (stateless)
* GestioneAmicizie (stateless)
* GestioneCollaborazioni (stateless)
  + 1. Diagrammi di sequenza

Per descrivere meglio la sequenza di operazioni che avvengono nel sistema sono stati allegati alcuni diagrammi di sequenza.

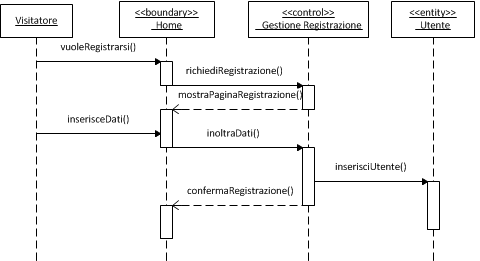


Fig. . - Sequence Diagram registrazione

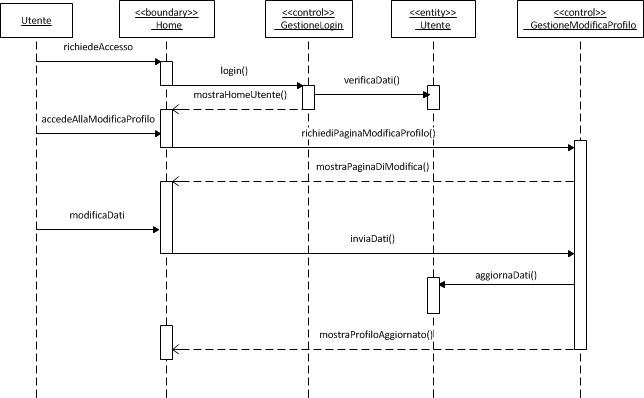


Fig. . - Sequence Diagram modifica profilo

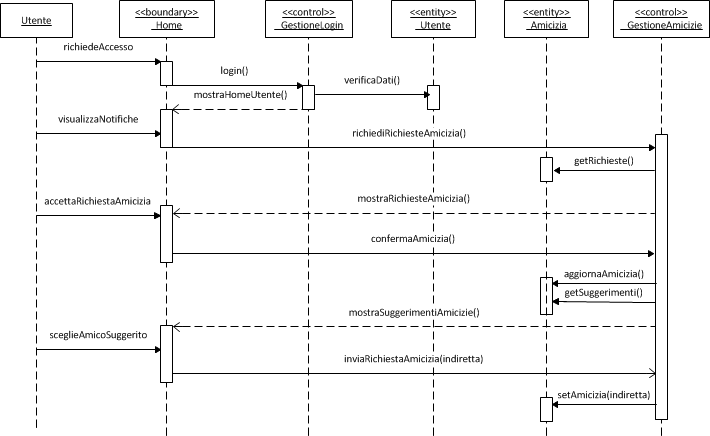


Fig. . - Sequence Diagram accetta richiesta amicizia

# Indice delle figure

Fig. 2.1 – Architettura Three-Tier 7

Fig. 2.2 – Architettura Three-Tear 8

Fig. 3.1 - Schema ER 10

Fig. 3.2 - EER Model 13

Fig. 3.3 – Schema Logico 16

Fig. 3.4 - UXModel Amministratore 17

Fig. 3.5 - UXModel Visitatore 18

Fig. 3.6 - UXModel Utente 19

Fig. 3.7 - BCE Model 21

Fig. 4.1 - EntityBeans 22

Fig. 4.2 - Sequence Diagram registrazione 23

Fig. 4.3 - Sequence Diagram modifica profilo 23

Fig. 4.4 - Sequence Diagram accetta richiesta amicizia 24