Progetto di Ingegneria del Software 2

SWIMv2

Design Document

A.A. 2012/2013

Autori:
Emanuele Uliana (799256), Gabriele Rufolo (743695), Walter Rubino (742519)

Docente:
Prof.ssa Raffaela Mirandola

Versione 1.0 del 22/12/2012

Indice

1 Generali

1.1 Panoramica della piattaforma

La piattaforma sviluppata ha il compito di promuovere, tramite l'utilizzo di una web application, la collaborazione e la condivisione di informazioni utili alla risoluzione di problemi tra gli utenti.

Nel dettaglio, il presente documento di Design definisce la struttura concettuale e funzionale fornendo una precisa descrizione delle guidelines che saranno seguite nello sviluppo e nel deployment dell'applicazione.

Il documento è inoltre conforme con le specifiche presenti nel RASD.

Il sistema è utilizzato da tre categorie di utenti: gli ospiti, gli utenti registrati e gli amministratori. Tramite un sistema di autenticazione il sistema controlla chi può svolgere determinate azioni in base al proprio ruolo.

Per evitare l'accesso a servizi non consentiti si è deciso di utilizzare il Role-Based Access Control. Con tale sistema di autenticazione si consentirà di usufruire solamente dei servizi specifici per il proprio ruolo.

1.2 Scopo di questo documento

Lo scopo di questo documento è quello di descrivere nel dettaglio, anche tramite l'ausilio di diagrammi E-R, UML, UX, ecc. , lo scheletro del sistema, i suoi vari componenti (database, interfaccia utente e logica applicativa), le tecnologie utilizzate e la gestione della sicurezza per comprendere al meglio come è stata sviluppata SWIMv2.

1.3 Glossario

In questo documento vengono usate le seguenti sigle/abbreviazioni:

- HW: Hardware
- SW: Software
- DD: Design Document
- PP: Project Planning
- RASD: Requirements Analysis and Specification Document
- DB: Database
- J2EE: Java Enterprise Edition
- UX: User Experience (diagram)
- DBMS: DataBase Management System
- AS: Application Server

2 Descrizione del sistema

Il sistema è composto da varie pagine HTML ognuna delle quali fornisce uno specifico servizio al cliente. Ogni webpage include la tecnologia Javascript che consente non solo di migliorare l'interfaccia stessa della pagine ma permette anche di ottimizzare la facilità di navigazione.

Per sviluppare il sistema sono stati usati i seguenti tool:

- WaveMaker, per creare l'interfaccia grafica;
- Eclipse Juno, per creare la logica di business;
- Mysql, per gestire la base di dati.

2.1 L'architettura del sistema

SWIMv2 è sviluppato secondo l'architettura client-server, e quest'ultimo secondo il pattern MVC, in modo da tenere nettamente separati lo stato del sistema, la logica di business e l'interfaccia grafica. In particolare, per quanto riguarda il client, possiamo immaginare che esso sia un comunissimo browser che manda delle richieste ad un server ed elabora le sue risposte, mostrando all'utente le informazioni desiderate tramite la view. Il server invece è locale, ospita il database e la logica di business e fornisce un'interfaccia grafica personalizzata a seconda dello stato del sistema; tale stato varia a seconda dei comandi ricevuti dal client, ma non in modo arbitrario: il controller si occupa di mantenere il DB in uno stato consistente e di comporre volta per volta l'HTML da passare al browser; a questo proposito, particolare importanza hanno le servlet, componenti che appunto generano le pagine web dinamiche che l'utente vede. Il server, come già specificato è JBoss AS 5, un'implementazione open source della piattaforma J2EE.

Possiamo riassumere la struttura dell' applicazione come una multi-tier composta dai seguenti livelli:

- Client tier: è composta dal client side dell'applicazione: si occupa di inviare le richieste al webserver tramite il browser. Include il codice Javascript e il codice HTML per la presentazione dei servizi.
- Web tier: comprende le servlet per le risposte alle richieste Ajax e quindi con il client tier. Hanno anche il compito di gestire la sessione con gli utenti del sistema e di ricevere da quest'ultimi gli input inviati e di passarli quindi al business tier.
- Business tier: gestisce la logica di sistema
- Data tier: comprende la base di dati della quale si servirà il business tier. Tale base dati comprenderà tutti i dati e tutte le informazioni del sistema.

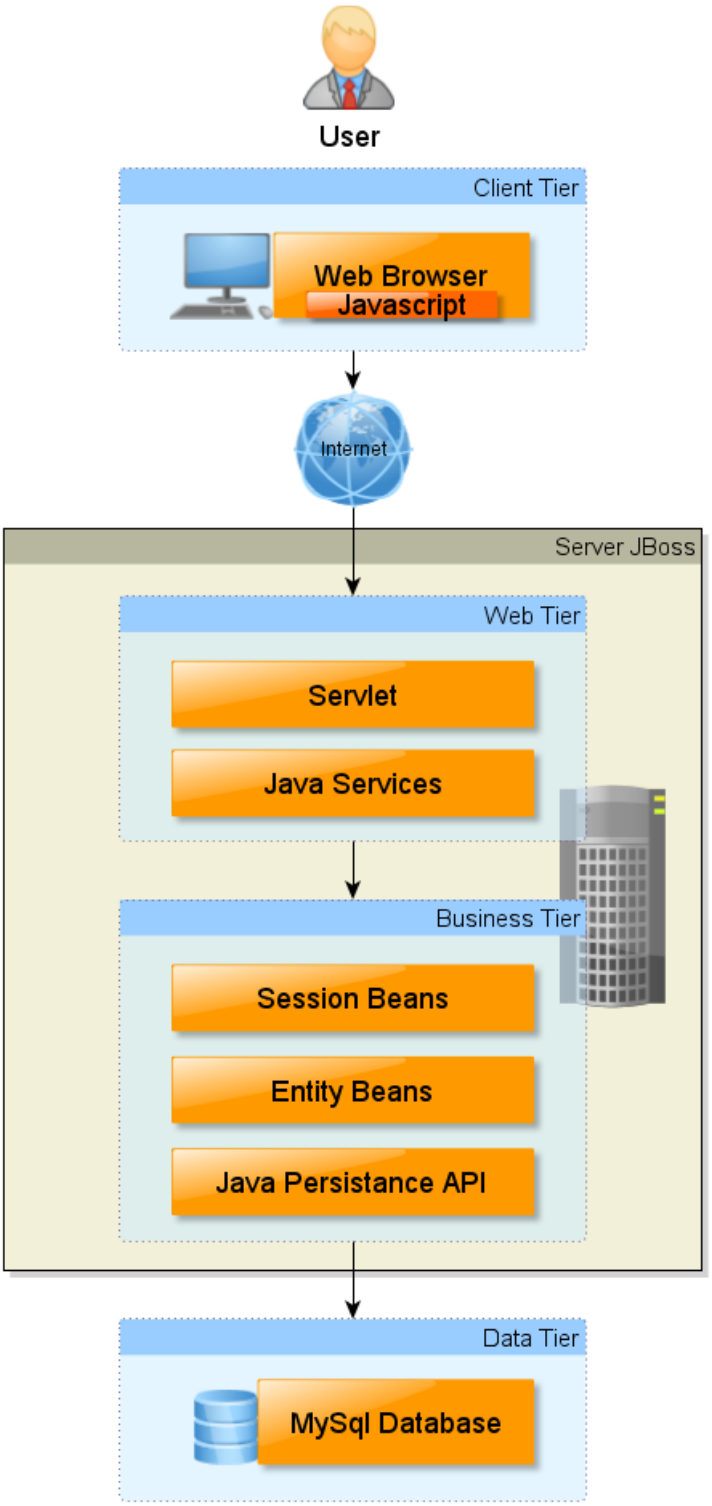

Figura 1: Architettura e schema di deployment del sistema

2.2 Scomposizione in sottosistemi

L'applicazione si può suddividere in sistemi componenti, un' analisi legata alla gestione delle funzioni principali di SWIM ci ha portato ad individuare: il login, i sottosistemi di ospite, utente registrato, amministratore e l'archiviazione dei messaggi, discussioni, registrazioni ecc.

- Login: si occupa della gestione dell'accesso al sistema, in base alla tipologia dell'utente seleziona il sottosistema di utilizzo;
- Archivio: si occupa della memorizzazione dei dati importanti per le attività all'interno della piattaforma quali la memorizzazione dell'utenza registrata e dei contenuti tramite l'utilizzo del DBMS ;
- Ospite: interagisce col sistema tenendo conto delle restrizioni imposte dallo stato anonimo dell'utente (impossibilità di replicare ai messaggi, inviare feedback. . .)
- Utente registrato: gestisce tutte le azioni dell'utente autenticato e la condivisione delle informazioni del suo profilo
- Amministratore: gestisce sia tutte azioni esclusive dell'amministratore (come l'accettazione di nuove abilità) che quelle comuni all'utente registrato (come l'invio di messaggi)

L'esistenza dei sottosistemi dell'utenza di SWIM è giustificata dalla loro diversa interazione con l'archivio.

Gli amministratori potranno modificare le tabelle all'interno del DBMS, gli utenti registrati interagiranno con l'archiviazione dei messaggi e infine gli ospiti potranno visualizzare il contenuto di una parte dei dati senza poter apportare nessuna modifica; è bene precisare che le funzionalità di tali sottosistemi non sono in mutua esclusione infatti la ricerca all'interno dell'applicazione può essere svolta da tutti gli internauti.

Per gestire tale meccanismo abbiamo scelto una soluzione che prevede di sfruttare la logica dell'applicazione, essa sarà in un certo senso intelligente e secondo l'appartenenza di classe renderà possibili le corrispettive interazioni con il database; ciò eviterà inutili accessi all'archivio.

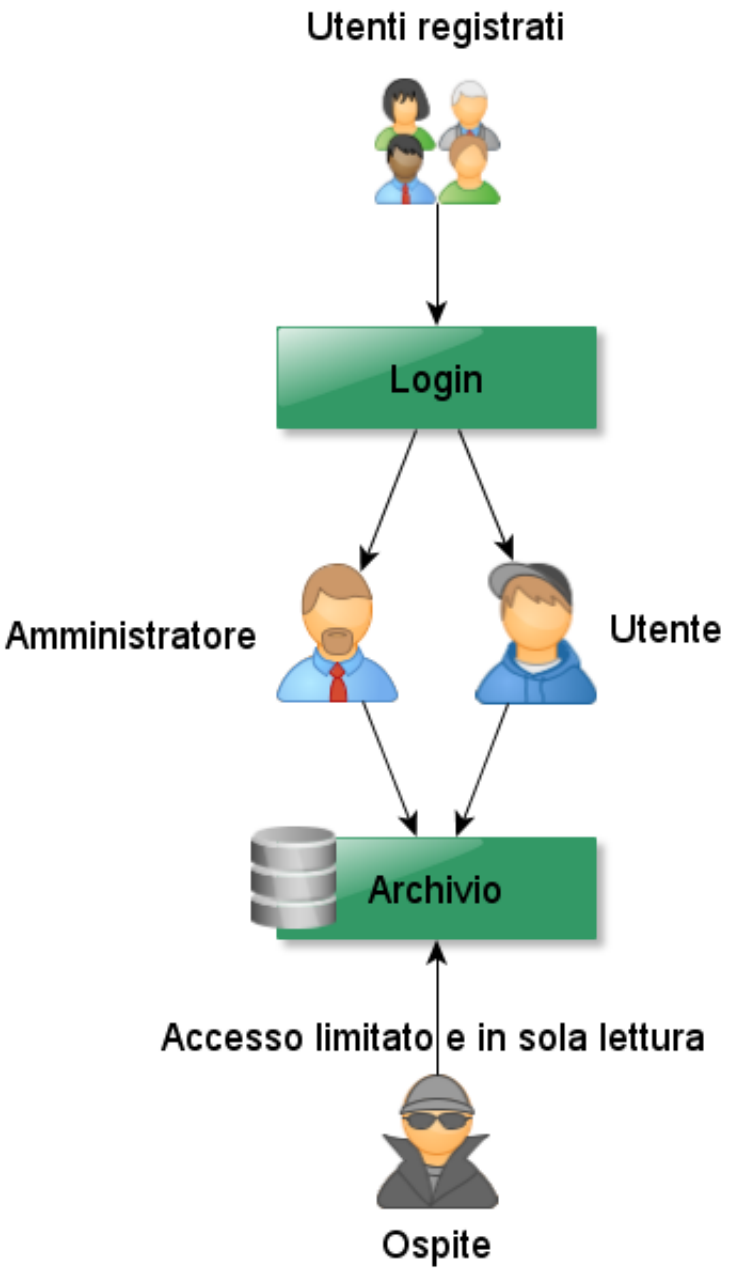

Figura 2: Rappresentazione dei sottosistemi e delle loro interazioni

3 Design del Database

Il database ha il ruolo della memorizzazione dei dati persistenti riguardanti l'applicazione.

3.1 Modello concettuale

La progettazione concettuale serve a rappresentare in maniera semplificata la realtà che poi andrà a identificare il sistema, per rendere facilmente leggibile tale astrazione viene fatto uso del diagramma entità-relazione dal significato rapido e intuitivo. Questo modello ha il compito di organizzare il mantenimento dei dati in maniera propedeutica al loro utilizzo successivo e pertanto sarà essenziale alla realizzazione effettiva di tale raccolta informativa.

Il diagramma entità-relazione è formato dalle entità che entrano in gioco in SWIMv2 e dalle relazioni che intercorrono tra di esse; è un passo intermedio verso la definizione delle tabelle del database del sistema e mette in evidenza in che modo le entità interagiscono tra di loro.

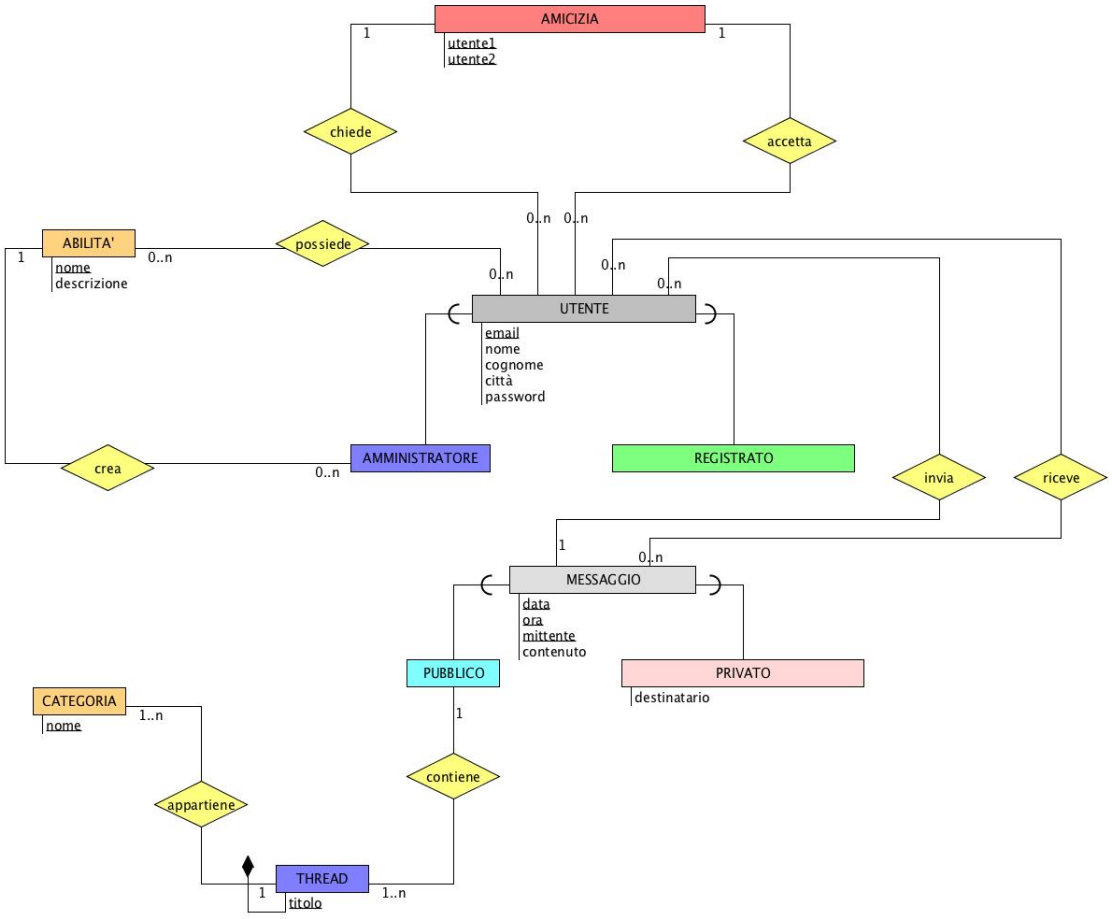

Figura 3: Diagramma ER che sintetizza la struttura del database

3.1.1 Entitá

- **Utente**: è l'entità padre rappresentante un individuo autenticato in SWIM, contenente quindi i dati di registrazione: nome, cognome,città, indirizzo email,password e un campo admin (ottenuto tramite un collasso verso l'alto della generalizzazione, nella tabella sarà di tipo boolean) che indica se l'utente è un amministratore.
- **Amministratore**: è una delle due entità figlie di Utente che ha delle funzionalità in più riguardanti la gestione delle abilità.
- **Registrato**: è l'altra entità figlia di Utente che ha le funzionalità dell'utente registrato a SWIM quali la richiesta di amicizia ad altri utenti e la replica ai messaggi.
- **Messaggio**: è un'altra entità padre modella l'esistenza degli scambi testuali che avvengono tra gli utenti, ha come attributi la data, l'ora, il mittente e il contenuto.
- **Privato**: è un' entità figlia di Messaggio oltre a tutti gli attributi del padre è dotata anche del campo destinatario.
- **Pubblico**: questa entità si distingue dal Messaggio Privato poichè non ha il campo destinatario in quanto contenuta all'interno dei Thread.
- **Thread**: quando viene creata una discussione all'interno dell'applicazione nasce il Thread, esso è dotato di un titolo.
- **Categorie**: rappresenta l'entità modellizzante gli argomenti principali di discussione all'interno del sistema
- **Abilità**: è un' entità che rappresenta un modello delle capacità degli utenti, ha come attributi il nome e una breve descrizione.
- **Amicizia**: l'entità rappresentante l'amicizia presente tra due utenti, ha come attributi le chiavi dei due amici.

3.1.2 Relazioni

- **chiede - accetta**: una delle funzionalità di SWIM permette all'utenza di richiedere e accettare richieste di amicizia tra di loro, questa relazione modellizza tale concetto.
- **invia - riceve**: questa relazione indica la possibilità di inviare e ricevere messaggi di tipo pubblico e/o privato.
- **possiede**: l'Utente possiede delle determinate abilità che caratterizzano il suo ruolo all'interno del Social Network.
- **crea**: un Amministratore può creare delle nuove abilità che potrebbero anche essergli state suggerite da un Utente Registrato tramite messaggio privato.
- **contiene**: un Thread contiene tutti i Messaggi Pubblici che lo riguardano.
- **appartiene**: un Thread ha una Categoria di appartenenza che permette quindi di catalogare le varie argomentazioni trattate.

3.2 Progetto Logico

La progettazione logica ha il ruolo di trasformare il concetto espresso tramite il diagramma entità-relazioni in una descrizione logica della struttura dati. In questa fase l'ER prodotto precedentemente può subire delle variazioni necessarie a garantirne la funzionalità ed un' immediata traduzione in tabelle effettive all'interno del DBMS.

3.2.1 Ristrutturazione dello schema Entità-Relazione

- La generalizzazione di Utente subisce un collasso verso l'alto, viene introdotto un attributo booleano admin che permetterà di differenziare l'attività dei due tipi diversi di utenti all'interno di SWIM.
- La generalizzazione Messaggio subisce un collasso verso il basso avremo quindi la tabella Messaggi Privati e la tabella Messaggi Pubblici.
- La relazione riceve-pubblici cessa di avere significato in quanto i Messaggi Pubblici sono associati ai Thread non ai singoli utenti, viene pertanto eliminata.

3.2.2 Traduzione verso il modello relazionale

- Le relazioni chiede-accetta 1 a molti si riassumono nel realizzare l'unica tabella Amicizia con le due chiavi degli utentiche essendo già presenti in tale relazione non devon esser aggiunte.
- La relazione possiede essendo una molti a molti va risolta creando una tabella apposita che ha come attributi chiave il nome dell'Abilità e l'email dell'Utente.
- La relazione crea 1 a molti va trattata aggiungendo all'interno della tabella delle Abilità come attributo l' email-dell'amministratore che l'ha realizzata.
- La relazione appartiene 1 a molti con l'entità debole Thread si risolve aggiungendo all'interno della tabella Thread l'attributo nome-categoria come chiave.
- La relazione contiene 1 a molti viene trattata come crea ossia si aggiunge alla tabella Messaggio Pubblico l'attributo titolo-thread.
- La relazione riceve-privati essendo una molti a molti porta alla realizzazione di una tabella apposita contenente le chiavi del Messaggio Privato in quanto l'attributo mittente indica già la mail dell' Utente che ha mandato il messaggio.
- Le relazioni invia-privati e invia-pubblici essendo 1 a molti si sarebbero risolte aggiungendo alle tabelle dei due tipi di messaggi l'attributo email-mittente che siccome è già presente non necessitano di ulteriori modifiche.

Le tabelle risultanti dalla traduzione del diagramma ER sono:

- Utente(**email**, nome, cognome, città, password, admin);
- Amicizia(**email-utente1**, **email-utente2**);
- Abilità(**nome**, email-amministratore, descrizione);
- Messaggio Pubblico(**data**, **ora**, **mittente**, titolo-thread, contenuto);
- Messaggio Privato(**data**, **ora**, **mittente**, destinatario, contenuto);
- Categorie(**nome**);
- Thread(**titolo**, **nome-categoria**);
- Possiede-Abilità(**email-utente**, **nome**);
- Riceve-Privati(**email-mittente**, **data**, **ora**);

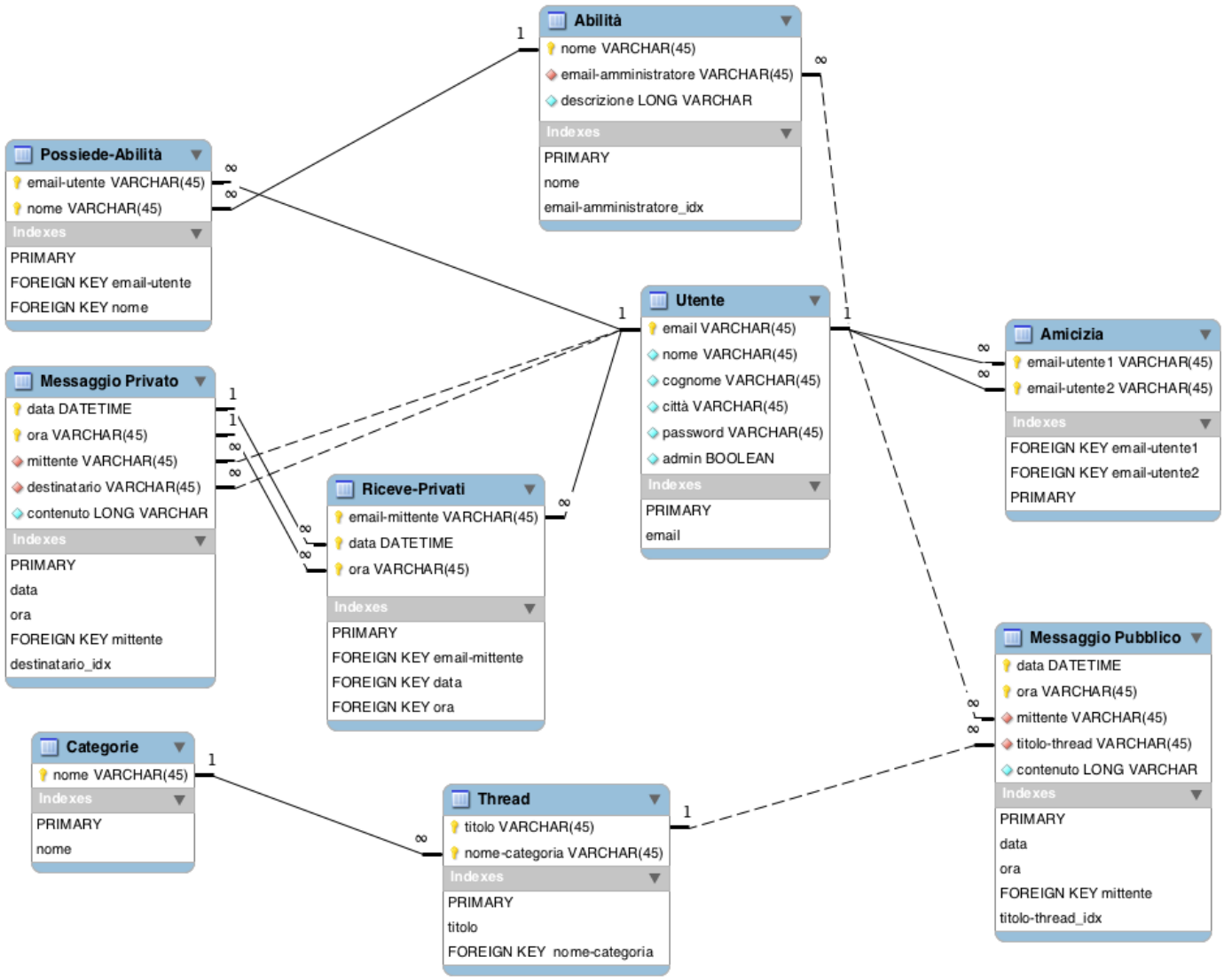

Figura 4: Relazioni tra le tabelle nel database

4 Design

In questa sezione si trovano dei diagrammi volti a modellizare la navigazione all'interno dell'applicazione.

4.1 UX Diagrams

Gli UX Diagrams mostrano una visione delle schermate, degli eventi e delle funzionalità che caratterizzano l'esperienza dell'utente durante l'uso del sistema. Avremo quindi screen per modellizzare le schermate e gli input form che mostrano dei moduli soggetti a compilazione, essi vanno a rappresentare gli stereotipi del modello UX, tramite le frecce possono essere invece indicati degli eventi oppure l'attivazione di un metodo, l'essere orientate o meno va a sottointendere la direzione di navigazione. Ogni entità può essere dotata di attributi e metodi qualora vadano a modificare lo stato del sistema.
Tratteremo questo tipo di modellizzazione dai 3 differenti punti di vista possibili nell'uso di SWIM ossia quelli dell' ospite, dell'utente registrato e dell'amministratore.

4.1.1 UX per un ospite

Analizzeremo innanzitutto la navigazione dell'ospite, che ricordiamo può compiere ricerche all'interno dell'applicazione, visionare il contenuto dei thread, registrarsi ed eventualmente procedere con l'accesso al sistema da utente registrato.

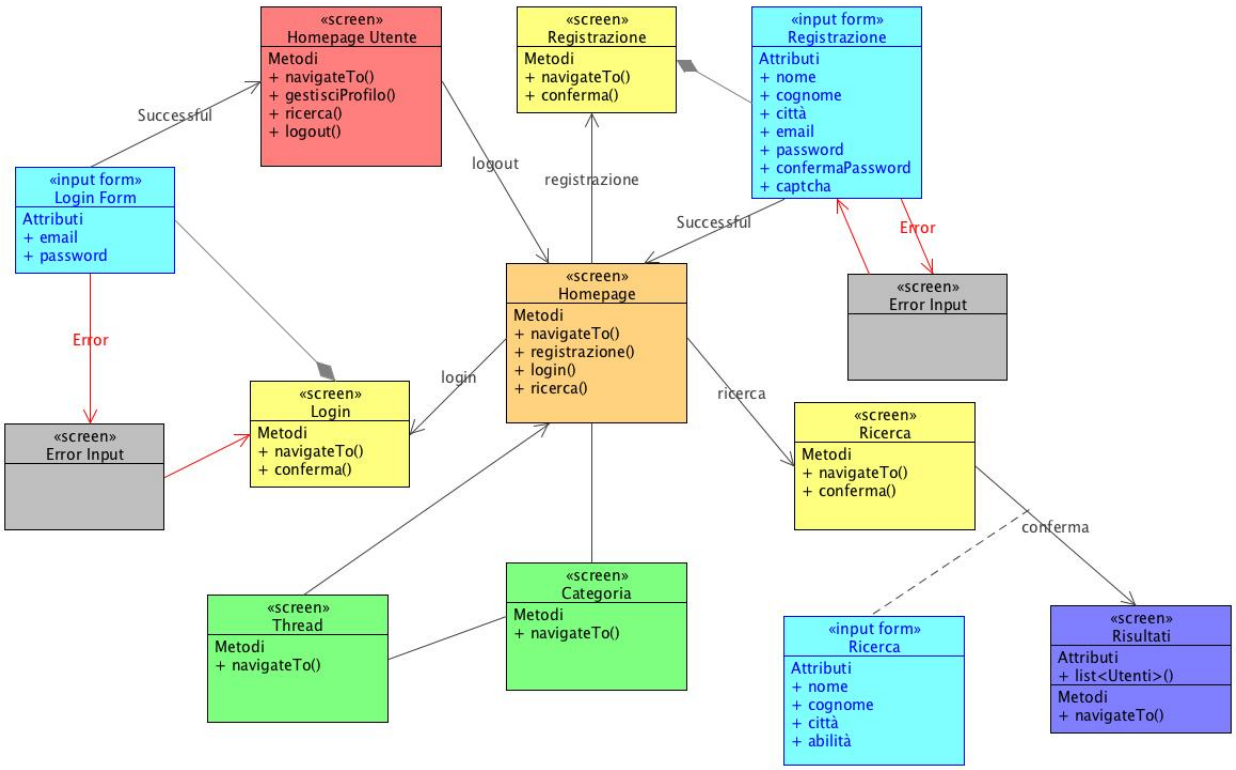

Figura 5: Modello di navigazione all'interno di SWIM da parte di un ospite

- I moduli della registrazione e dell'accesso sono stati modellati come classi contenute per descrivere l'eventuale meccanismo di reindirizzamento nel caso sia presente una casistica di errore.
- Il modulo di ricerca invece è stato risolto come classe associata ad associazione in quanto la ricerca nell' eventualità sfortunata porta ad una lista risultante nulla.
- L' Homepage è stata identificata come riferimento principale in quanto è il fulcro della maggior parte delle diverse sequenze che si possono realizzare, come si può notare il colore è distinto rispetto a quello dell' Homepage Utente che in questa modellizzazione iniziale rappresenta solo una destinazione dalla quale non avverrà altro se non l'operazione di logout.
- In giallo sono state colorate le schermate che rappresentano le effettive operazioni che può svolgere l'ospite, mediante le quali interagirà con il database.
- In verde invece le schermate attraverso le quali l'ospite potrà navigare con il solo scopo di leggere.

4.1.2 UX per un utente registrato

Passiamo adesso ad un' analisi riguardante la navigazione con i privilegi degli utenti registrati, dopo aver effettuato l'autentificazione infatti, potranno svolgere attività quali la gestione del loro profilo, l'invio di messaggi privati, la partecipazione attiva ai thread tramite messaggi pubblici e l'inviare e richiedere amicizie agli altri utenti di SWIM.

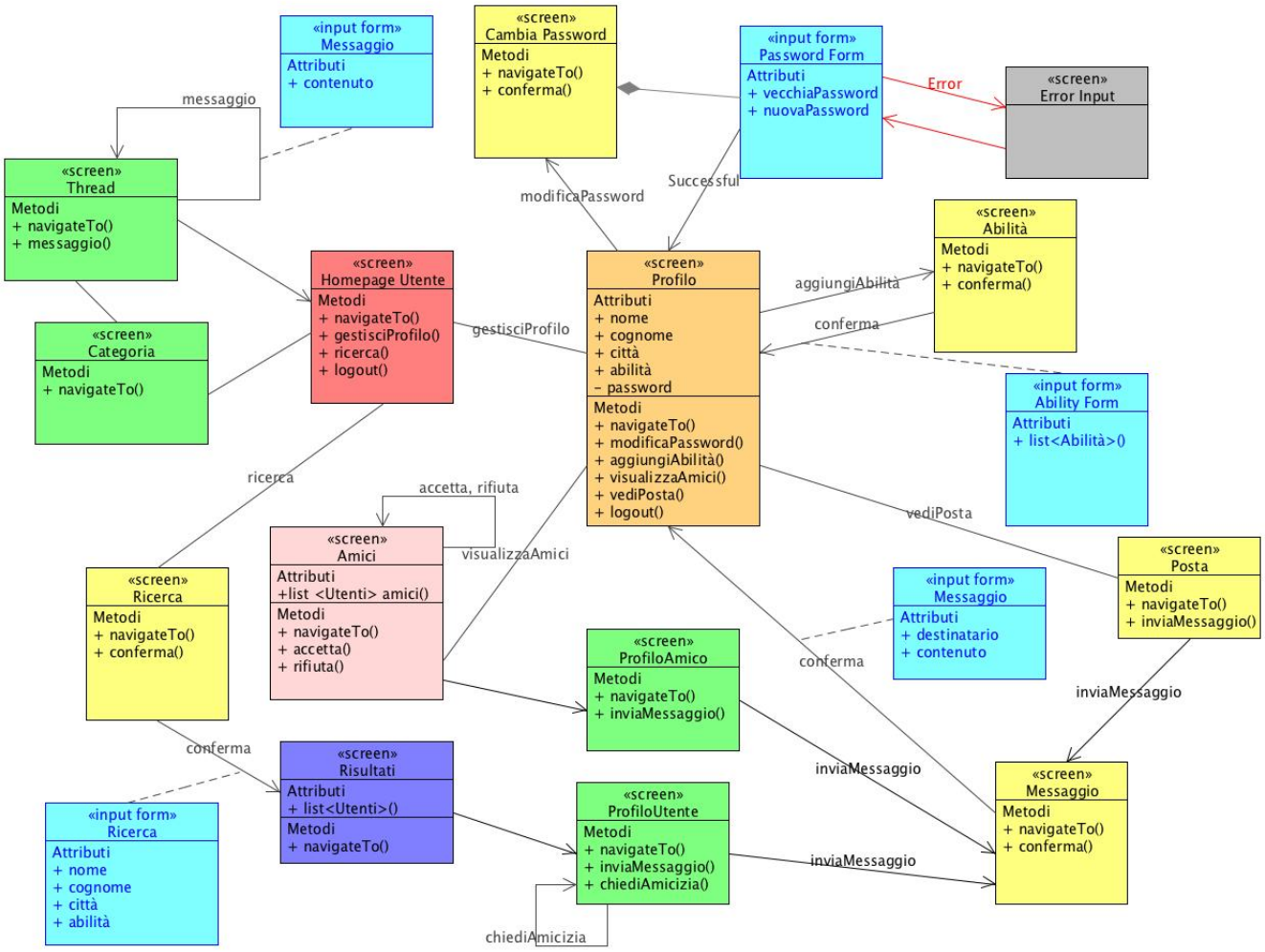

Figura 6: Modello di navigazione all'interno di SWIM da parte di un utente registrato

- ♢ Per questo modello il fulcro è il Profilo in quanto da esso è possibile compiere la maggior parte delle attività privilegiate dell'utente autenticato.
- ♢ Il modulo per la modifica della password è stato rappresentato con una classe contenuta per sottolineare la possibilità di immettere una password vecchia errata.
- ♢ Il modulo per la modifica delle abilità è stato modellato con una classe associata ad associazione in quanto l'aggiunta di abilità alle propie facoltà avrà sempre esito positivo.

- Anche i moduli per la scrittura dei messaggi privati e pubblici sono stati modellati con classi associate ad associazione, i messaggi non incorrono in casistiche di errore grazie al fatto che, ove necessario, il campo destinatario sarà completato tramite selezione dalla lista amici o autoimmissione da parte del sistema.

- Gli autoanelli indicano azioni che possono essere compiute e che vanno quindi a modificare lo stato del sistema, la particolarità è data dal fatto che sarà visualizzata la stessa schermata, nella quale saranno presenti le eventuali modifiche.

- Le convenzioni usate per i colori sono le medesime del diagramma precedente.

4.1.3 UX per un amministratore

Infine trattiamo la navigazione da parte di un amministratore, per semplicità di lettura, sarà modellata solo l'attività più importante che lo contraddistingue da un utente registrato ossia la facoltà di creare nuove abilità che saranno quindi disponibili all'utenza di SWIM.

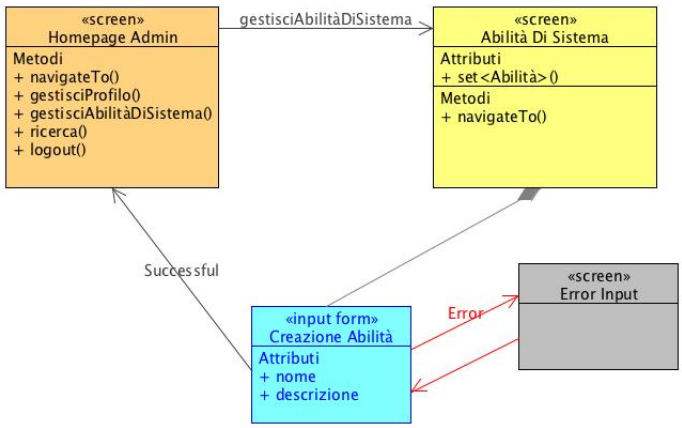

Figura 7: Modello di navigazione da parte di un amministratore di SWIM

- Il modulo per la creazione di una nuova abilità è stato realizzato mediante una classe contenuta per tener conto dell'eventualità che possa essere immesso il nome di un' abilità già esistente.

5 Sicurezza

Per accedere a tutte le funzioni e per motivi di sicurezza sono richiesti la registrazione ed il login: entrambe le procedure sono molto semplici, in quanto l'interfaccia della web app è progettata per essere il più user-friendly possibile. Tutte le informazioni che gli utenti forniscono nella fase di registrazione sono memorizzate in un database di tipo relazionale MySql.
L'ultima sezione del Design Document riguarda le metodologie impiegate per garantire la sicurezza del sistema: innanzittutto vengono impiegati hash salato e crittografia per evitare che nel database siano presenti in chiaro dati sensibili; poi, l'altro grande punto è il filtraggio delle stringhe immesse in qualunque modulo per evitare attacchi del tipo SQL Injection e XSS.