

teachify

Design Deliverable

Traini Luca

241090

lucatra89@gmail.com

Project Guidelines

- List of Challenging/Primary/Important Tasks [mandatory]
- Justification of the design documents you are going to submit [mandatory]
- Conceptual Design:
 - Product Overview [mandatory]
 - Competitors [Strategy phase in User-centric design]
 - Personas [Strategy phase in User-centric design]
 - Requirement [Scope phase in User-centric design] [mandatory]
 - Use Cases
 - Interaction Design [User-centric design]
 - Lo-Fi Wireframe [User-centric design]
 - Hi-Fi Wireframe [User-centric design]
 - Conceptual Design Decisions [mandatory]
- Technical Design (not meant to be in sequential order):
 - Conallen Analysis Model
 - IFML Domain Model
 - RESTful API Design
 - Business Logic e ORM
 - Technical Design Decisions [mandatory]
- Effort Recording [mandatory]
- Final Considerations [mandatory]

Important:

- document risky/difficult/complex/highly discussed concepts
- document design decisions for risky/difficult/complex/highly discussed concepts
- iterate: do not spend more than 10 full days for each iteration
- prioritize requirements, scenarios, users, etc. etc.
- note: in each iteration, document only what you require very important. If, for example, you do not need an Analysis Model, simply do not use it!!!

List of Challenging/Primary/Important Tasks

CHALLENGING TASK	DATE THE TASK IS IDENTIFIED	DATE THE CHALLENGE IS RESOLVED	EXPLANATION ON HOW THE CHALLENGE HAS BEEN MANAGED
Realizzazione della ricerca geografica dei tutors	05/02/2015	27/02/2015	Si è capito di dover realizzare una funzionalità che permetta di individuare i tutor più vicini rispetto ad una particolare posizione. La soluzione è stata quella di appoggiarsi ad un servizio di geocoding esterno che permetta di individuare la latitudine e la longitudine della posizione in modo da poterle utilizzare nel DB per effettuare query spaziali.
Prenotazione diretta delle ore in cui effettuare lezioni	06/02/15	25/02/15	Si è capito che questa funzionalità era troppo dispendiosa in termini di implementazione e di quantità di dati da mantenere. Per questo si è deciso che fare in modo l'utente possa richiedere la disponibilità al tutor indicando una descrizione del testuale delle propria richiesta. Una volta che il tutor ha accettato la richiesta verranno resi noti i contatti del tutor(telefono, email, etc).

Justification of the design documents you are going to submit

Il documento tenta di dare una visione globale dell'applicazione, modellandone ogni singola prospettiva. Nella prima fase è stato utilizzato lo "User Centered Design", attraverso l'approccio proposto da Jesse James Garret; grazie alla notevole rilevanza che dà all'utente quest'approccio è molto utile per realizzare applicazioni moderne con un alto grado di usabilità, requisito fondamentale nello scenario tecnologico attuale. La qualità della User Experience è infatti uno dei fattori fondamentali che può determinare il successo di una applicazione, per cui oltre a individuare i classici requisiti e casi d'uso, è necessario decidere in maniera dettagliata quali saranno i target e le personas dell'applicazione e fare scelte in base alle loro ipotetiche esigenze. Realizzare una User Experience di qualità vuol dire confutare e decidere quali saranno i flussi di navigazione all'interno del portale (Interaction Design), e parallelamente strutturare in maniera visuale le informazioni che dovranno essere fruite dall'utente (Lo-Fi Wireframe), attraverso questi step l'applicazione inizia ad avere una natura più concreta e iniziano ad evidenziarsi problematiche legate all'utilizzo reale dell'applicazione, problematiche che dagli Use Case o dai requisiti non sarebbero mai potuti emergere. Oltre questo per avere una visione maggiormente dettagliata di quello che sarà il risultato finale è necessario creare dei prototipi che mostrino, attraverso pagine web statiche oppure attraverso immagini realizzate con tool grafici, quale sarà l'aspetto reale dell'applicazione. La progettazione della Sitemap e dei Lo-Fi e Hi-Fi Wireframe permetterà di guardare il prodotto attraverso gli occhi dell'utente, senza farsi influenzare (almeno in questa fase) dalle scelte tecnologiche e dai tecnicismi, ovviamente questo in alcuni casi può portare a gestire delle complessità eccessive, per cui è opportuno sempre tener presente le caratteristiche ed i limiti delle tecnologie attuali. Ovviamente una volta definita dettagliatamente la User Experience è opportuno iniziare a progettare quello che sarà "il motore" dell'applicazione, analizzando in maniera più tecnica in che modo l'applicazione verrà realizzata e definendo i componenti software che la comporranno. Attraverso l'Analysys Model di Conallen, e soprattutto al suo approccio, che si adatta perfettamente alle applicazioni che sfruttano il pattern architettonale MVC è stato possibile delineare in maniera precisa quali fossero le componenti software da implementare sia client e server, e in che modo esse comunicano. Parallelamente alla definizione delle componenti è opportuno iniziare a definire più in dettaglio quali saranno i dati di business (domain model) dell'applicazione, per modellarli è stato utilizzato il Domain Model di IFML. Visto che nell'applicazione verrà utilizzata un'architettura REST è opportuno documentare il più possibile le URL che verranno utilizzate e la loro struttura, per questo nel documento verrà riportato uno schema che descrive le API in maniera sintetica ma efficace. Ovviamente la fase di modellazione e quella di sviluppo spesso si sovrappongono temporalmente per questo è opportuno, durante (se non prima) della definizione delle componenti, iniziare a scegliere gli strumenti tecnologici che dovranno essere utilizzati e giustificarne l'utilizzo. Un'altra parte fondamentale della progettazione/implemetazione è stato quello relativo all Business Logic ed al relativo mapping Object-Relational, infatti è stata dedicato un paragrafo che descrivesse brevemente le componenti che implementano la persistenza dei dati e le scelte più critiche effettuate nel mapping. Sia per la fase di design concettuale che per quella tecnica, sono stati riportate in maniera sintetica, sotto forma di grafici, le decisioni più importanti e determinanti effettuate, esse sono state dettagliate nei file Conceptual.xls e Technical.xls.

Strategy

Product overview

Il portale è rivolto a chi offre e cerca lezioni private per qualsiasi disciplina o materia in qualsiasi regione d'Italia.

Il tutor potrà registrarsi al sistema indicando le fasce d'orario in cui è disponibile, potrà inserire le proprie generalità (nome , cognome , descrizione , domicilio ...) ed il costo orario delle lezioni. Inoltre i tutors potranno inserire le diverse tipologie di insegnamenti che essi forniscono indicando la materia di competenza e il livello scolastico (scuole elementari, scuole superiori,).

L'utente che ha necessità di ripetizioni potrà effettuare ricerche in base alla materia e alla posizione. Potrà verificare quali sono i tutor più adatti alle proprie esigenze consultando i feedback degli altri utenti.

Una volta scelto l'annuncio più adatto alle proprie esigenze l'utente potrà effettuare una richiesta di disponibilità. Il tutor potrà verificare le richieste ed eventualmente segnalare la propria disponibilità. Segnalando la propria disponibilità l'utente avrà finalmente accesso ai contatti del tutor (numero telefonico, email,...).

Gli amministratori avranno accesso ad un servizio di back-end dal quale sarà possibile aggiungere o eliminare gli amministratori di sistema e dal quale si potranno gestire le materie, i livelli scolastici del sistema e le tariffe adottabili dai tutor.

Competitors

Sono stati individuati 3 principali competitors:

LEZIONI PRIVATE

<http://www.lezioniprivate.net/>

Modello di business:

Totalmente gratuito. Sostentamento attraverso banner pubblicitari.

Caratteristiche:

Sostanzialmente è esclusivamente una bacheca di annunci. Ha un aspetto notevolmente antiquato e presenta anti-pattern di design. È possibile effettuare ricerche per categorie provincie. È possibile anche pubblicare richieste di annunci.

STUDY HOME

<http://www.studyhome.it/>

Modello di business:

Pagamento della prestazione tramite coupon prodotto dal sistema sul quale viene trattenuta una fee. Nessun pagamento all'iscrizione né per lo studente né per il tutor. Il pagamento

avviene a prestazione conclusa. Possibili sconti in base ad un meccanismo di accumulazione crediti.

Caratteristiche:

Il sistema ha un'aspetto professionale e rispetta i trend delle moderne applicazioni web . Ricerca in base a posizione geografica , materia e target. Il meccanismo dei coupon potrebbe essere un limite in paese come l'Italia. L'applicazione è disponibile esclusivamente nel territorio italiano.

FIRST TUTORS

<https://www.firsttutors.com/>

Modello di business:

Lo studente paga una volta(€7 per lezione dal costo orario inferiore a 29,99€ altrimenti il costo è di 14€) per ricevere i contatti del tutor.

Il tutor paga €7.00 più le spese 3.50% della tariffa oraria per le prime cinque ore di lezione richieste e solo se l'accordo è confermato sia dall'insegnante che dallo studente).

Caratteristiche:

Il portale è piuttosto consolidato ed è presente in numerosi paesi Canada , Regno Unito, Sud Africa, Italia , etc.

First tutor ha un aspetto piuttosto professionale e presenta anche dei meccanismi di feedback da parte degli utenti e la ricerca di tutor per codice di avviamento postale e per materia, tuttavia il sito si presenta piuttosto antiquato e non segue affatto i trend delle moderne applicazioni web.

Il competitor di riferimento per l'applicazione è senza dubbio Study home, un'applicazione ben fatta e che rispetta tutti canoni delle moderne applicazioni web. Punto di forza dell'applicazione è la possibilità di effettuare prenotazioni direttamente da un calendario i cui sono visualizzate gli orari in cui un tutor è disponibile. Tuttavia l'app allarga il proprio boundary anche al pagamento della prestazione attraverso coupon forniti dall'applicazione, quest'ultimo fattore può essere considerato uno dei punti di deboli dell'app in un contesto come quello italiano.

Teachify non punta a fornire un servizio di prenotazione diretta del tutor, bensì ,come avviene già in [firsttutors.com](https://www.firsttutors.com/)), a fornire i contatti che poi l'utente potrà utilizzare per comunicare con il tutor. I punti di forza su cui cercherà di puntare l'applicazione è il meccanismo dei feedback e dei punteggio assegnato al tutor (I feedback potranno essere lasciati da un utente per cui è stata accettata la richiesta di disponibilità da parte del tutor). L'idea è quella di lanciare il servizio gratuito, e poi una volta consolidata la posizione nel mercato farlo diventare a pagamento per i tutor che vogliono essere registrati nel sistema.

User research

L'applicazione ha l'obiettivo di creare un canale utile a chi voglia sostenere o fornire lezioni private. I target dell'app possono essere suddivisi essenzialmente in quattro categorie:

- **Genitori:**

Age : 30-50 anni.

Background tecnologico : Molto variegato. Include sia persone che utilizzano la tecnologia in maniera abbastanza frequente e sia persone che la utilizzano saltuariamente.

Lavoro : Dall'operaio al dirigente d'azienda.

Ruolo : In cerca di ripetizioni.

- **Studenti in cerca di ripetizioni**

Age : 13-27 anni.

Background tecnologico : Buona conoscenza della tecnologia e del web. Sono abituati ad utilizzare internet per soddisfare le proprie esigenze (acquisti online, Social network , chat).

Lavoro : Studente delle medie/superiori/università.

Ruolo : In cerca di ripetizioni.

- **Studenti o ex**

Age : 19-27 anni.

Background tecnologico : Buona conoscenza della tecnologia e del web. Sono abituati ad utilizzare internet per soddisfare le proprie esigenze (acquisti online, Social network , chat).

Lavoro : Studente dell'università o ex studente.

Ruolo : Impartisce lezioni.

- **Professori o ex**

Age : 35-65 anni.

Background tecnologico : Molto variegato. Include sia persone che utilizzano la tecnologia in maniera abbastanza frequente e sia persone che la utilizzano saltuariamente.

Lavoro : Professore(in servizio o in pensione) delle medie o superiori.

Ruolo : Impartisce lezioni.

Personas



Roberta ha un figlio di 11 anni, Luca, che ha grosse difficoltà con la matematica, purtroppo non riesce ancora a trovare qualcuno che possa dargli delle ripetizioni. Grazie a Teachify ora Roberta può comodamente trovare da casa sua un tutor che può supportare Luca in vista del prossimo compito in classe. Inoltre potrà verificare i feedback degli altri utenti per rassicurarsi sulle effettive capacità del tutor.



“Giorgia ha 24 anni e sta per laurearsi in Ingegneria ambientale. È stanca di pesare economicamente sulle spalle dei suoi genitori operai. Per questo ha deciso di iniziare ad impartire lezioni private. Ha lasciato dei bigliettini sulla bacheca dell'università ma a quanto pare questo non basta. Da quando ha scoperto Teachify riceve continui richieste di lezioni. Ciò è dato anche dalla sua bravura e dai feedback positivi lasciati dai ragazzi a cui lei ha impartito lezioni.



Nicola è uno studente fuori sede del primo anno di Informatica. Sin dalle superiori ha sempre avuto molta difficoltà con la matematica. Purtroppo il corso di Analisi Matematica 1 è propedeutico a tutte le altre materie. Quindi Nicola sta pensando di trovare qualcuno di fidato che possa aiutarlo. Purtroppo è ancora disorientato nella nuova città e non sa a chi affidarsi. Grazie a teachify potrà trovare facilmente il tutor più adatto alle proprie esigenze.



“Gino è un professore di matematica delle scuole superiori. Purtroppo la moglie ha da poco perso il lavoro, per questo il suo stipendio inizia ad andare stretto alle esigenze della famiglia. Grazie a teachify ha cominciato a farsi conoscere e ad avere dei buoni feedback con conseguente aumento richieste di ripetizioni.

Requisiti

Le priorità dei requisiti vengono indicate con punteggio da 1 a 5 con il seguente formalismo. ($^x\{x\}$) dove x indica la priorità.

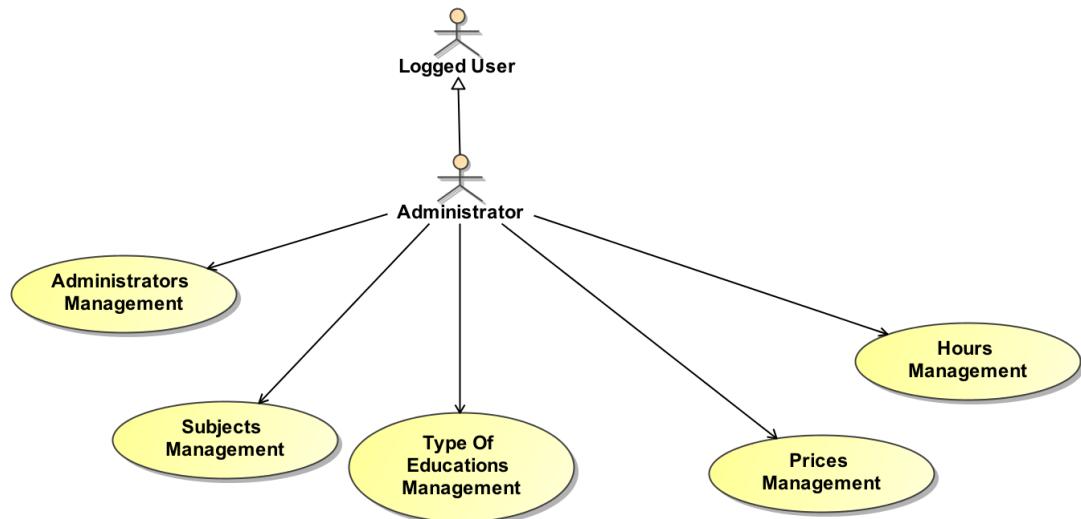
1. L'utente deve essere in grado di scegliere il tutor indicando un luogo, la materia ed il livello scolastico(Università , Scuole Medie, Superiori, Discipline Extra). **($^5\{5\}$)**
2. Il sistema deve fornire un elenco di tutors in base alle richieste dell'utente, essi devono essere ordinati in base alla distanza dal punto indicato nella ricerca. Se l'utente non indica alcun punto geografico la ricerca non verrà eseguita .**($^5\{5\}$)**
3. L'utente può consultare le informazioni del tutor su una pagina in cui verrano riportate le ore ed i giorni in cui è disponibile per le lezioni, una descrizione generica,il costo orario delle lezioni, e le diverse discipline da lui insegnate.**($^5\{5\}$)**
4. Nel profilo del tutor l'utente potrà visualizzare i feedback degli altri utenti ed un punteggio assegnato in base alla media dei voti dei feedback.**($^4\{4\}$)**
5. L'utente può lasciare un feedback (solo nel caso in cui il tutor ne abbia accolto la richiesta di disponibilità) nel quale deve lasciare un punteggio compreso fra 1 e 5 ed al quale può associare un commento.**($^4\{4\}$)**
6. L'utente può richiedere i contatti del tutor scrivendo un messaggio testuale nel quale potrà indicare le proprie esigenze.**($^5\{5\}$)**
7. Non appena il tutor avrà accettato la richiesta di disponibilità, l'utente avrà accesso ai contatti del tutor (email , telefono , skype).**($^5\{5\}$)**
8. L'utente potrà accedere al servizio di ricerca senza loggarsi, la login sarà necessaria solo nel momento in cui l'utente vorrà effettuare una richiesta di contatto.**($^5\{5\}$)**
9. L'utente deve essere in grado di registrarsi al sistema indicando nome, cognome, email, ed una password.**($^5\{5\}$)**
10. L'utente loggato potrà modificare le proprie informazioni (nome e cognome) dal proprio profilo.**($^5\{5\}$)**
11. L'utente loggato potrà modificare la propria password dal proprio profilo.**($^2\{2\}$)**
12. L'utente loggato potrà gestire un'immagine nel proprio profilo .**($^3\{3\}$)**
13. L'utente loggato ha accesso alla " dashboard dello studente" dalla quale potrà verificare lo stato delle richieste effettuate e l'elenco dei tutor che hanno accettato la richiesta.**($^5\{5\}$)**
14. L'utente loggato potrà facilmente diventare tutor effettuando la registrazione come tutor, aggiungendo obbligatoriamente una località che lo identifichi.**($^2\{2\}$)**
15. Il tutor avrà accesso a tutte le funzionalità disponibili per l'utente loggato.**($^5\{5\}$)**
16. Il tutor avrà accesso alla " dashboard del tutor" dal quale potrà gestire informazioni quali : l'indirizzo, una descrizione, costo orario delle lezioni.**($^5\{5\}$)**
17. Il tutor può inserire ed eliminare gli intervalli orari in cui è disponibile in ogni giorno della settimana.
18. Il tutor può inserire ed eliminare le materie insegnate indicando anche i livelli scolastici a cui le lezioni sono rivolte. **($^5\{5\}$)**
19. Un utente non registrato nel sistema potrà diventare tutor inserendo nome, cognome, email , password ed una località.**($^5\{5\}$)**
20. Il tutor può visualizzare le richieste effettuate dagli utenti e decidere se accettare o rifiutare di condividere i propri contatti con loro. **($^5\{5\}$)**
21. Il tutor potrà gestire i contatti (email , skype , telefono) dalla "dashboard del tutor".
22. Il tutor potrà selezionare la propria tariffa oraria fra quelle rese disponibili dal tutor dalla "dashboard del tutor".

23. L'amministratore può accedere alla dashboard amministrativa loggandosi con una email ed una password.**(*5)**
24. L'amministratore può aggiungere altri amministratori al sistema (Può aggiungere esclusivamente utenti già registrati nel sistema).**(*5)**
25. L'amministratore può eliminare i permessi da amministratore ad altri utenti.**(*3)**
26. L'amministratore può eliminare sia utenti che tutor dal sistema.**(*2)**
27. L'amministratore può rimuovere feedback dal sistema.**(*2)**
28. L'amministratore può aggiungere, rimuovere e modificare le materie selezionabili dai tutor .**(*5)**
29. L'amministratore può aggiungere, rimuovere e modificare i livelli scolastici selezionabili dai tutor .**(*5)**
30. L'amministratore può aggiungere e rimuovere le tariffe orarie adottabili dai tutor .**(*5)**
31. L'amministratore può aggiungere e rimuovere le ore selezionabili dai tutor per indicare gli intervalli di disponibilità .**(*5)**

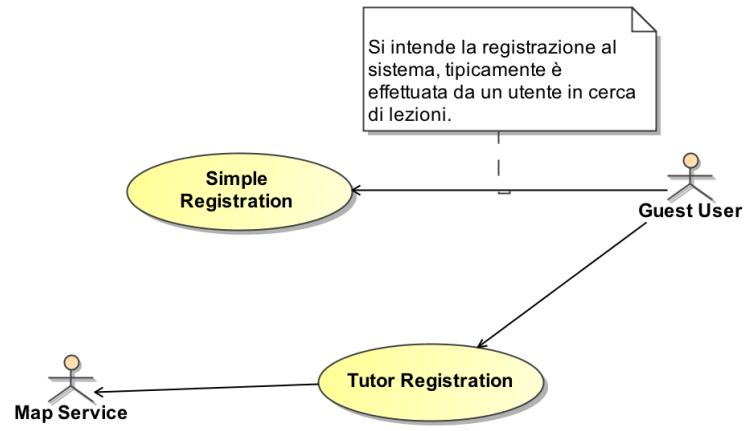
Per rendere la realizzazione del progetto fattibile nei giusti tempi si è deciso, in base alle priorità assegnate di rimandare a sviluppi futuri le funzionalità legate ai seguenti requisiti: 11, 14, 26, 27. Tali requisiti non verranno considerati nella documentazione che segue.

Use cases

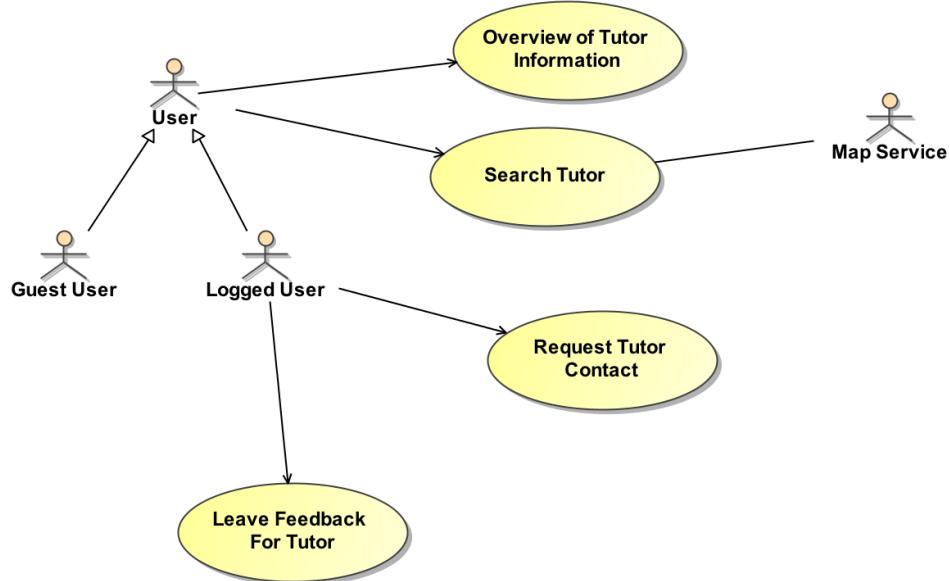
Di seguito vengono mostrati i diagrammi che raccolgono i vari Use Case presenti nell'applicazione, per la tracciabilità sono indicati anche i requisiti a cui fanno riferimento.

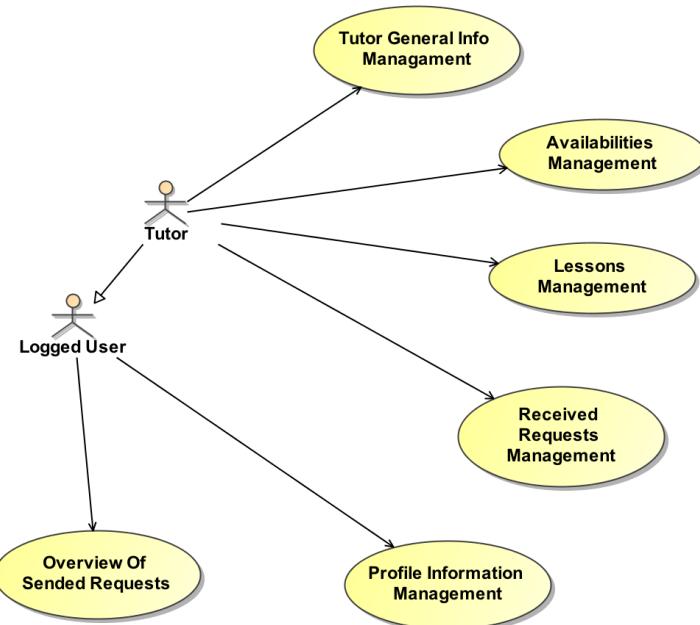


Use Case	Descrizione	Requisiti
Administrators Management	Eliminazione e aggiunta di amministratori di sistema	24, 25
Subjects Management	Eliminazione, aggiunta e modifica di materie che possono essere utilizzate dai tutor per indicare le proprie lezioni	28
Type Of Educations Management	Eliminazione, aggiunta e modifica dei livelli scolastici che possono essere utilizzate dai tutor per indicare le proprie lezioni	29
Prices Management	Eliminazione e aggiunta delle tariffe orarie applicabili dai tutor	30
Hours Management	Eliminazione e aggiunta delle ore che possono essere utilizzate dai tutor per indicare gli intervalli di disponibilità giornaliera dei tutor	31



Use Case	Descrizione	Requisiti
Simple Registration	Registrazione come utenti semplici del sistema (in cerca di lezioni)	9
Tutor Registration	Registrazione come tutor (offrono lezioni)	19



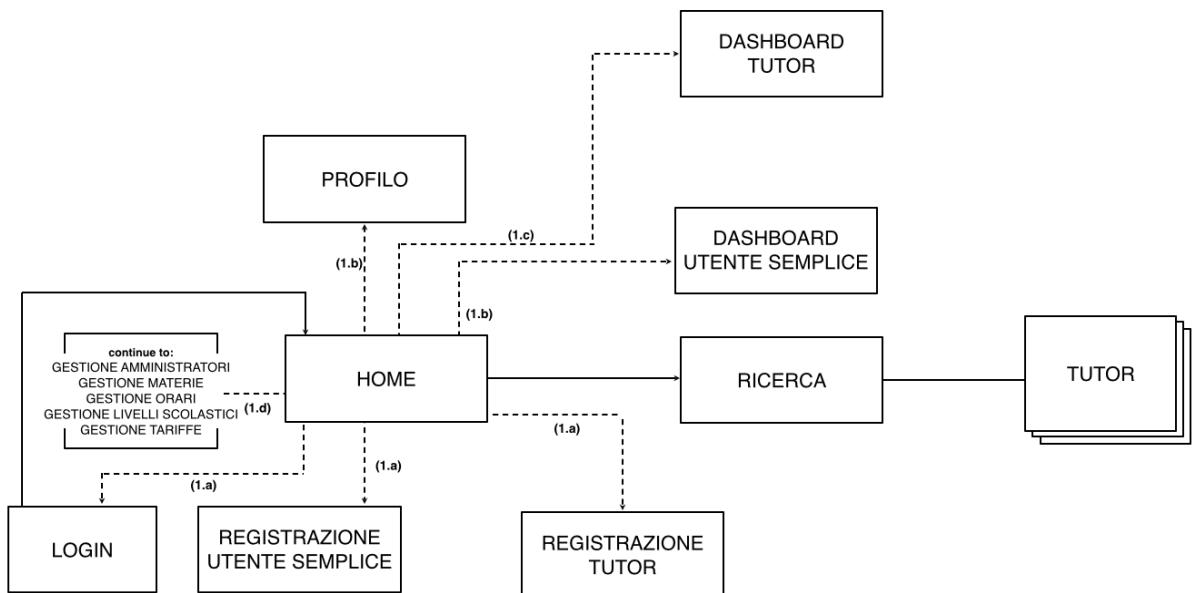


Use Case	Descrizione	Requisiti
Overview Of Sended Requests	Verificare lo stato delle richieste effettuate. Se le richieste sono state accettate dal tutor vengono mostrati i suoi contatti(skype, email, telefono)	13
Profile Information Management	Gestione della foto profilo, del nome e cognome	10, 12
Received Requests Management	Verificare le richieste ricevute, con la possibilità di accettarle o rifiutarle	20,7
Lessons Management	Gestione delle materie per cui di effettuano lezioni con i livelli scolastici di riferimento.	18
Availabilities Management	Gestione degli intervalli orari giornalieri di disponibilità.	17
Tutor General Info Managment	Gestione delle tariffe, della descrizione e del punto geografico che lo identifica.	16
Use Case	Descrizione	Requisiti
Leave Feedback For Tutor	Lasciare una valutazione del tutor (solo se il tutor ha accettato la sua richiesta)	5
Request Tutor Contact	Richiedere al tutor la possibilità di accedere ai suoi contatti.	6, 7, 8

Use Case	Descrizione	Requisiti
Search Tutor	Effettuare ricerche indicando una località, e la materia ed il livello scolastico per cui si desidera avere lezioni private.	1, 2, 8
Overview of Tutor Information	Visualizzare il profilo del tutor in cui sono indicati gli orari giornalieri in cui è disponibile, le materie ed i livelli scolastici a cui sono indirizzate le lezioni, la tariffa oraria, una descrizione e il domicilio,	3,4,7

Interaction Design

Per descrivere il flusso della user experience all'interno dell'app sono state utilizzate delle sitemap basate sul “Visual vocabulary” di Jesse James Garret.



(1.a) Se l'utente non è loggato nel sistema

(1.b) Se l'utente è loggato nel sistema

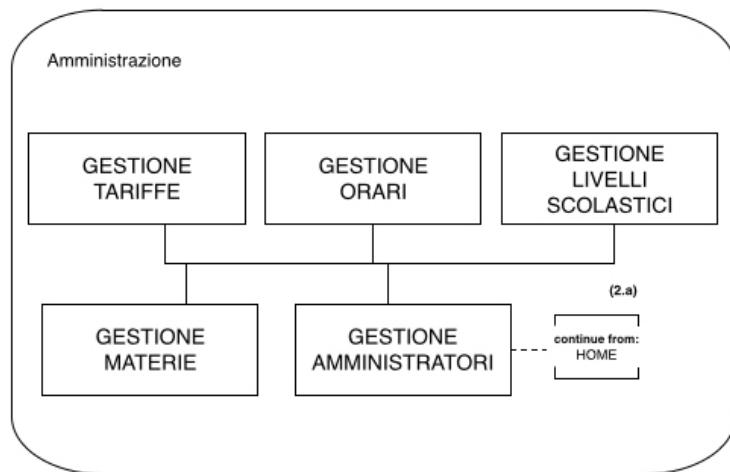
(1.c) Se l'utente loggato è registrato anche come tutor

(1.d) Se l'utente è loggato ma non è registrato come tutor

(1.d) Se l'utente loggato è registrato come amministratore

Il portale ha come punto di accesso la pagina HOME, la scelta è ricaduta su di essa poichè focalizza quello che è il main task dell'applicazione, ossia la ricerca dei tutor in base al tipo di materia, al livello scolastico e la località geografica desiderata. Una volta effettuata la ricerca, l'utente verrà rediretto alla pagina RICERCA nella quale verrà mostrata la lista dei tutor che soddisfano i criteri precedentemente indicati. Attraverso tale lista sarà possibile accedere al dettaglio di un particolare tutor, ossia alla pagina TUTOR. Dalla pagina TUTOR sarà anche possibile tornare alla pagina RICERCA con l'elenco dei risultati precedenti, ovviamente questa scelta è stata fatta per rendere più agevole la consultazione dei risultati della ricerca. Attraverso la pagina RICERCA sarà inoltre possibile effettuare ulteriori ricerche che comporteranno l'aggiornamento dei risultati nella pagina. Per loggarsi al sistema è necessario accedere alla pagina LOGIN, non appena l'utente verrà autenticato verrà rediretto alla HOME ed avrà accesso alle pagine PROFILO e DASHBOARD STUDENTE attraverso l'header. Le route che sono raggiungibili dall'header del portale lo sono sempre in ogni punto del sito. Tali connettori sono stati omessi nel diagramma per non rendere eccessivamente confusionario il diagramma. Le route raggiungibili dall'header sono:

- HOME , REGISTRAZIONE SEMPLICE, REGISTRAZIONE TUTOR. (Se l'utente non è loggato);
- HOME, PROFILO, DASHBOARD STUDENTE (Se l'utente è loggato);
- Se l'utente loggato è registrato come tutor, oltre a poter accedere alle 3 pagine disponibili per l'utente loggato potrà accedere anche alla DASHBOARD TUTOR;
- Se l'utente loggato è registrato come amministratore, oltre a poter accedere alle 3 pagine disponibili per l'utente loggato potrà accedere anche alla AMMINISTRAZIONE che ha il suo punto di ingresso nella pagina GESTIONE AMMINISTRATORI;



(2.a) Se l'utente loggato è registrato come amministratore

Una volta acceduti all' AMMINISTRAZIONE, l'utente potrà navigare indifferentemente attraverso le pagine GESTIONE TARFFE, GESTIONE ORARI, GESTIONE LIVELLI SCOLASTICI,

GESTIONE MATERIE e GESTIONE AMMINISTRATORI. In ciascuna di esse l'utente potrà modificare , eliminare e aggiungere informazioni che saranno utilizzate nel sistema.

VOCABULARY

UTENTE SEMPLICE: inteso come utente che si registra al sistema e ne usufruisce per cercare dei tutor.

TARIFFA: intesi come costo orario per lezione applicabile da parte del tutor.

ORARI: inteso come orari che sono utilizzabili dai tutor per indicare gli intervalli di disponibilità durante la giornata.

LIVELLI SCOLASTICI: livelli di istruzione a cui saranno rivolte le lezioni del tutor (scuola elementare, medie, superiori, ecc.).

Lo-Fi Wireframe

Tutte le viste che verranno descritte di seguito faranno riferimento alla omonima pagina presente nella sitemap mostrata nell'Interaction Design.

This wireframe shows a search interface. At the top right are buttons for 'REGISTRATI', 'ACCEDI', and 'INSEGNA'. Below them is a search bar with placeholder text 'CERCA TUTOR DA...'. Underneath the search bar is a dropdown menu labeled 'POSIZIONE' with options like 'MATERIA', 'TIPO', and 'CERCARE'. Two red numbered circles are present: circle 1 is on the 'CERCARE' button, and circle 2 is on the 'TIPO' dropdown item.

This wireframe shows a dashboard-like interface. At the top right are buttons for 'REGISTRATI', 'ACCEDI', and 'INSEGNA'. Below them is a search bar with placeholder text 'CERCA TUTOR DA...'. The main area contains three circular icons representing users. To the right, there is a map with several location markers. Two red numbered circles are present: circle 1 is on the 'TIPO CAU' button, and circle 2 is on the 'PROFILO' button.

This wireframe shows a registration form. It includes fields for 'EMAIL', 'NUME', 'COGNOME', 'PASSWORD', and 'CONFIRM PASSWORD'. At the bottom is a large 'REGISTRATI!' button. Red numbered circle 1 is on the 'EMAIL' field, and circle 2 is on the 'REGISTRATI!' button.

This wireframe shows a registration form. It includes fields for 'EMAIL', 'NOME', 'COGNOME', 'PASSWORD', 'CONFIRM PASSWORD', and 'INDIRIZZO'. At the bottom is a large 'REGISTRATI!' button. Red numbered circle 1 is on the 'EMAIL' field, and circle 2 is on the 'REGISTRATI!' button.

C

D

HOME (Figura A)

La vista è minimale e mette in risalto il main task dell'applicazione, ossia la ricerca dei tutor per posizione(A.1). Una volta inserita la posizione geografica e selezionati i campi relativi alla materia ed al livello scolastico desiderati, sarà possibile effettuare la ricerca (A.2).

RICERCA (Figura B)

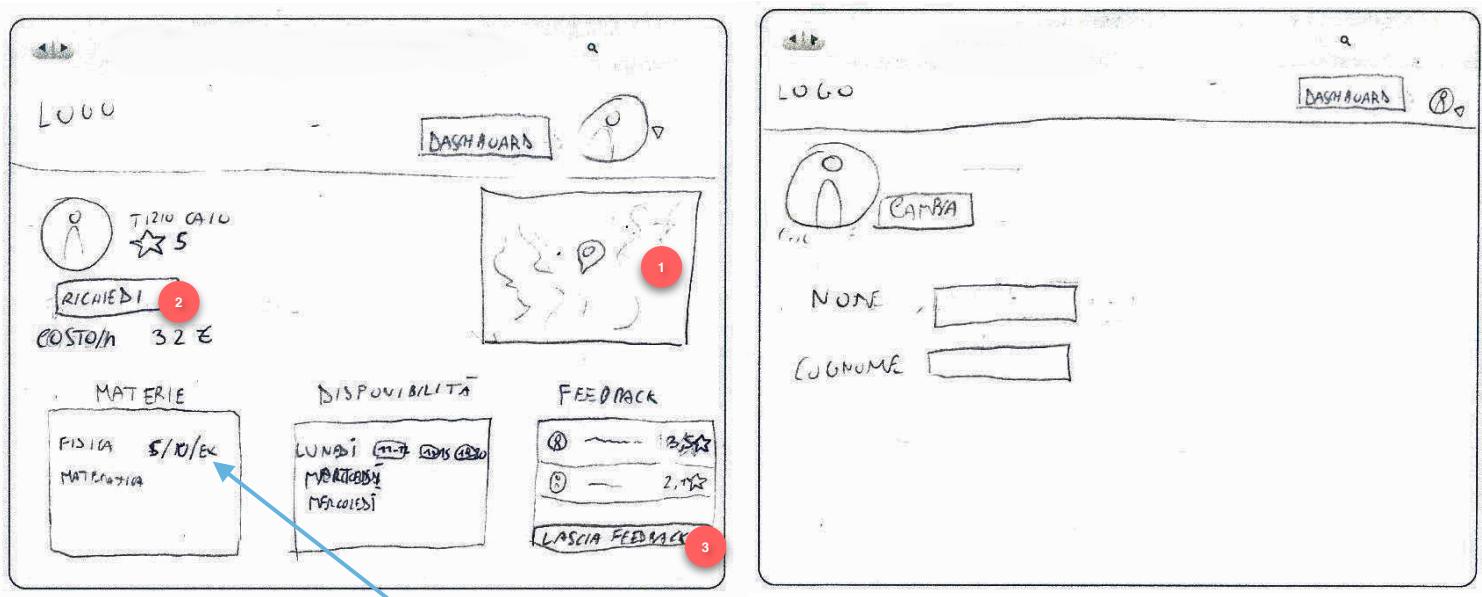
In questa vista vengono mostrati i risultati dalla ricerca effettuata. Sulla sinistra c'è la lista dei tutor con una descrizione sommaria e la distanza in base al punto indicato nella ricerca, per ciascun elemento della lista è possibile accedere al dettaglio (B.2). Sulla destra c'è invece una mappa centrata sul punto della ricerca con dei marker che indicano le posizioni dei vari tutor. Attraverso la searchbox(B.1) è possibile effettuare ulteriori ricerche, aggiornando così la lista dei tutor e la mappa.

REGISTRAZIONE UTENTE SEMPLICE (Figura C)

In questa vista l'utente può registrarsi al sistema indicando un email, una password, un nome ed un cognome.

REGISTRAZIONE TUTOR (Figura D)

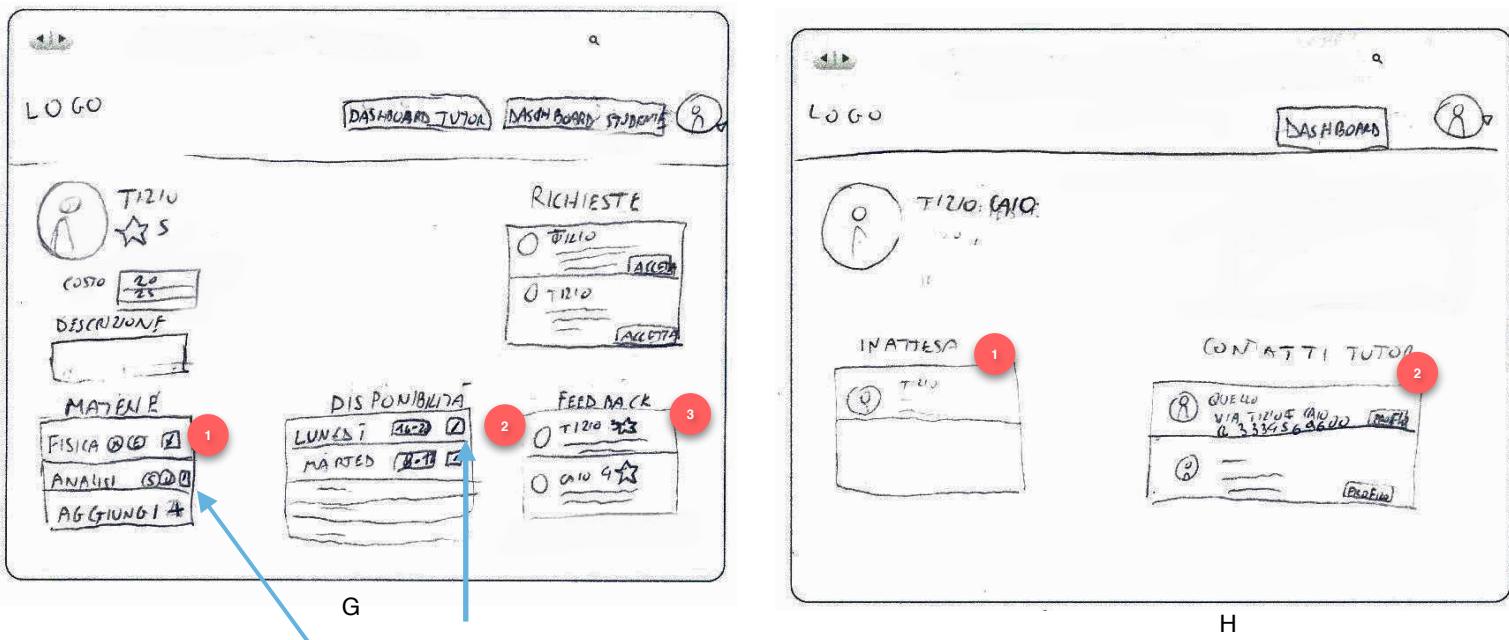
In questa vista l'utente può registrarsi al sistema come tutor. Oltre alle informazioni necessarie alla registrazione semplice l'utente dovrà inserire obbligatoriamente una posizione geografica.



E

I livelli scolastici relativi alla materia vengono mostrati attraverso un badge su sfondo colorato. Su ciascun badge c'è una lettera che identifica quel particolare livello scolastico ("s" per superiori, "u" per università, etc)

F



Attraverso questi pulsanti si aprono delle finestre modali che permettono la gestione, rispettivamente, delle materie e delle disponibilità

TUTOR (Figura E)

Qui l'utente può visualizzare tutte le caratteristiche di un tutor:

- la posizione (E.1)
- il punteggio medio assegnato dagli utenti
- il costo orario delle lezioni
- le materie ed i livelli scolastici a cui fanno riferimento
- gli intervalli orari giornalieri di disponibilità
- i feedback lasciati dagli altri utenti

Attraverso un pulsante(E.2) è possibile, se l'utente è loggato, effettuare una richiesta di disponibilità al tutor. Se l'utente ha già effettuato una richiesta verso il tutor il pulsante non viene mostrato. Se l'utente invece ha già effettuato la richiesta ed essa è stata accettata dal tutor l'utente potrà lasciare un feedback(E.3), inoltre in tal caso verranno mostrati i contatti personali del tutor(skype, telefono e email).

PROFILO(Figura F)

Attraverso il profilo l'utente potrà modificare il nome, il cognome e la propria foto del profilo.

DASHBOARD TUTOR(Figura G)

In questa vista il tutor può gestire le proprie informazioni relative all'insegnamento, aggiungendo ed eliminando materie, oppure modificandone i livelli scolastici di riferimento(G.1). Potrà gestire semplicemente gli orari (G.2) in cui si rende disponibile per le lezioni. Inoltre potrà modificare la propria locazione, il costo orario per le lezioni e la descrizione.

Sul lato destro verranno visualizzate le richieste ricevute dagli utenti, e sarà possibile accettarle oppure rifiutarle (G.3).

DASHBOARD UTENTE SEMPLICE(Figura H)

Qui l'utente può comodamente visualizzare lo stato delle proprie richieste, visualizzando la lista dei tutor a cui sono state effettuate richieste di disponibilità ma che non hanno ancora risposto(H.1) ed i tutor che invece l'hanno accettata, con i relativi contatti (telefono, skype e email).

Hi-Fi Wireframe

Come si può vedere dalle immagini riportate sotto , si è scelto di utilizzare uno stile minimale, per esaltare la semplicità e l'immediatezza dell'app. Gli Hi-Fi Wireframe sono stati realizzati in HTML5 e CSS3, e sono stati utilizzati sia nella fase di prototipizzazione e sia nell'implementazione effettiva.

DASHBOARD TUTOR (Figura G - LoFi)

The wireframe displays the Tutor Dashboard interface. At the top, there's a header with the Teachify logo, navigation links for 'Dashboard Studente' and 'Dashboard Tutor', and a user profile picture. The main area is divided into several sections:

- User Profile:** Shows a profile picture of Roberto Palma, his name, a 'Stars' rating of 4.4, and a 'Contatti' section with social media handles (robby.palma, robby@gmail.com, 3485765900).
- Address:** A text input field for 'Indirizzo' with a 'Modifica' button.
- Description:** A text input field containing a bio about being a nuclear engineer since 2005 and a private tutor for math and physics.
- Contatti:** A section for contact information.
- Tariffa oraria €/h:** A dropdown menu set to 20.
- Materie:** A table showing subjects taught: Fisica (S, U, P) and Matematica (E, M, S, U, P).
- Disponibilità:** A table showing availability: Lunedì (16-20, 10-13, edit), Martedì (16-20, 10-13, edit).
- Feedback:** A section showing a message from Gino Bruni: "Spero che mio figlio possa continuare ad avere le sue lezioni. La ringrazio della attenzione." with a 4-star rating.
- Richieste:** A list of requests:
 - Roberta Primiero:** Needs math lessons for her 10-year-old child on Friday afternoons. Buttons: Rifiuta (red), Accetta (green).
 - Nicola Pozzi:** Needs math lessons for an imminent assignment. Buttons: Rifiuta (red), Accetta (green).

L'apporto degli Hi-Fi wireframe è stato essenziale nella fase di prototipizzazione mostrando anticipatamente le problematiche legate alla user-experience non ancora individuate dai Lo-Fi. Ad esempio la rappresentazione/gestione delle materie e dei relativi livelli scolastici e la rappresentazione/gestione delle disponibilità orarie giornaliere dei tutor. Per quanto riguarda le materie si è scelto di indicarle attraverso una lista con dei badge colorati che indicassero le livelli scolastici a cui esse sono riferite, per modificare i livelli scolastici si è deciso di utilizzare

una finestra modale dal quale si possono agevolmente eliminare ed aggiungere livelli scolastici, non appena una materia non ha più alcun livello scolastico essa viene eliminata automaticamente (la finestra modale è mostrata nello screen-shot che segue il testo). Anche per la gestione delle disponibilità si è scelto di utilizzare lo stesso pattern. Tali soluzioni hanno incrementato notevolmente l'usabilità e la versatilità dell'applicazione, rendendola un'applicazione moderna e reattiva

GESTIONE LIVELLI SCOLASTICI DI UNA MATERIA

RICERCA (Figura B - LoFi)

Conceptual Design Decisions

I QOC sono descritti in dettaglio nel file Conceptual.xls. Di seguito viene presentata esclusivamente un riassunto grafico.

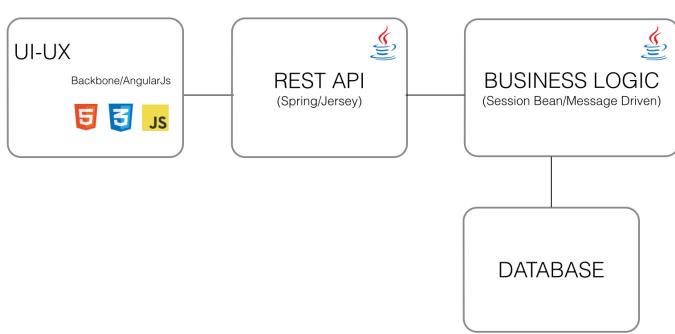
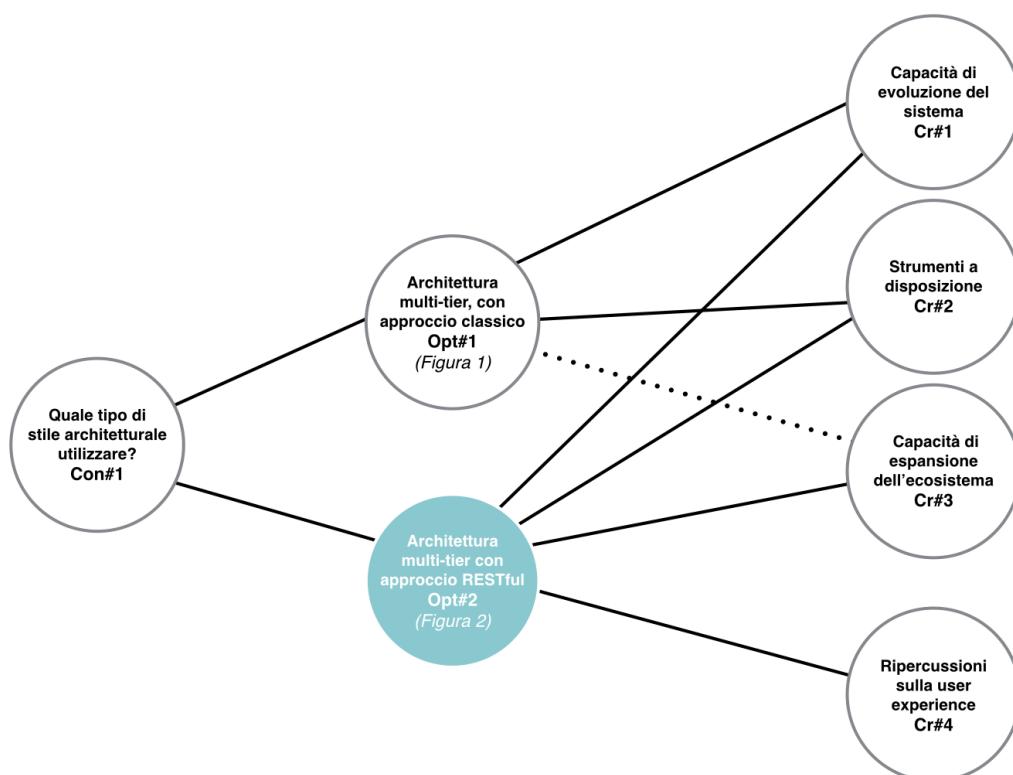


Figura 1

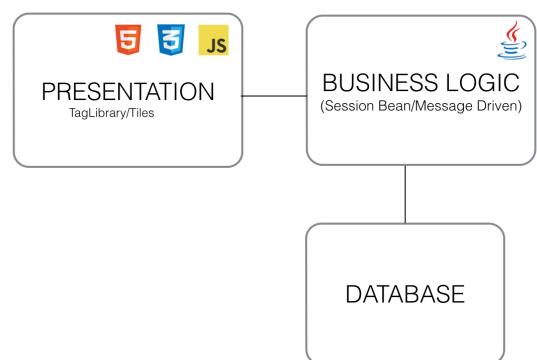
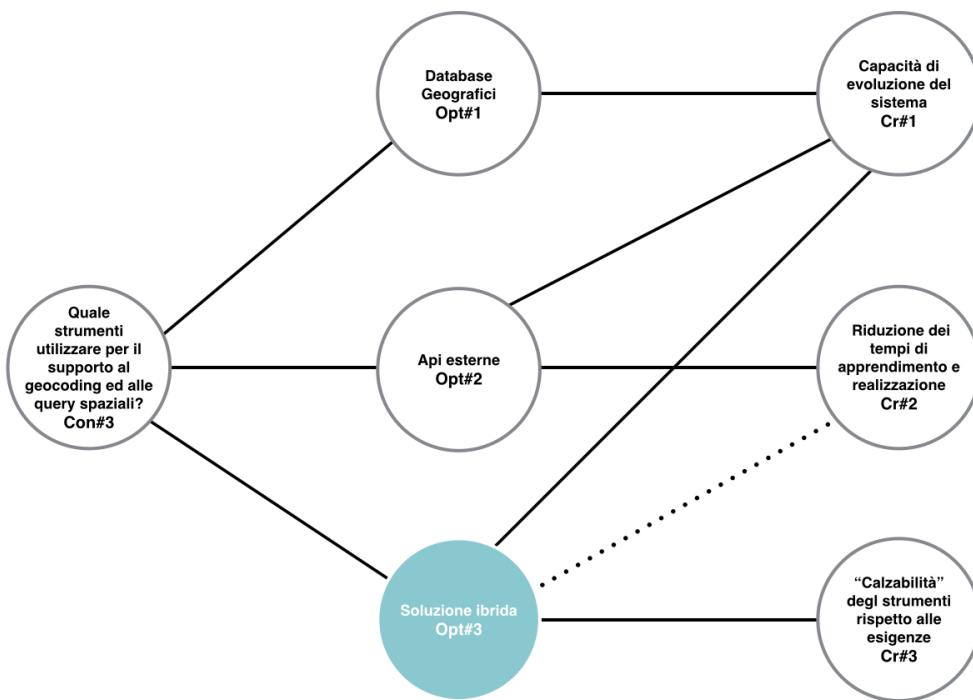
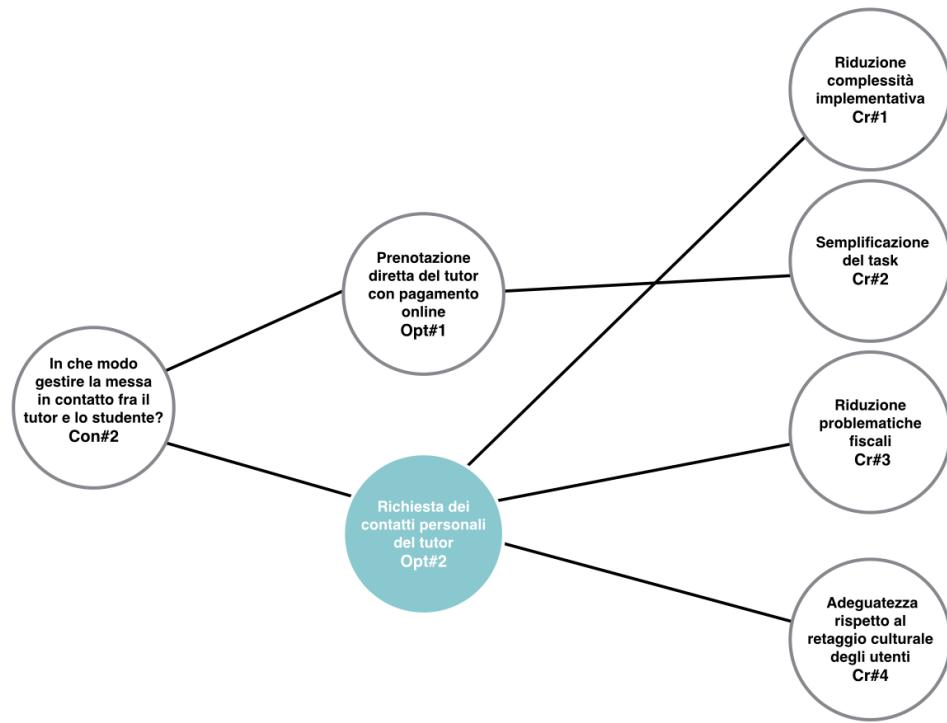


Figura 2



La scelta è ricaduta su questi tre poichè sono quelli che hanno delineato la natura dell'applicazione, queste scelte infatti influenzano irrimediabilmente lo sviluppo e le scelte implementative da effettuare.

La scelta dell'architettura da utilizzare oltre che influenzare la natura dell'applicazione stessa, influenzera anche le competenze necessarie dei componenti del team, nel caso del progetto ovviamente questo non avviene, ma in un progetto reale sarebbe essenziale capire l'architettura per poi andare a scegliere i giusti candidati, ad esempio nel caso dell'approccio RESTful di certo sarebbe necessario avere un esperto nell'ambito Front-End (JavaScript e relativi framework), inoltre sarebbe necessario avere un buon esperto nella progettazione di API REST e così via.

Il secondo QOC invece riguarda il main task del portale, ossia quello che è effettivamente il motivo per cui una persona dovrebbe utilizzare la Web App. Ovviamenete questa scelta è certamente quello che potrebbe decidere se l'app è realmente vincente oppure no, per questo va attentamente confutata e documentata , senza precludere la possibilità di eventuali cambi di strategia futuri. Anche questa scelta avrà ripercussioni sulle risorse umane necessarie, ad esempio se si fosse scelto di tirar dentro anche il pagamento online della lezione, in quel caso sarebbe stato necessario assumere dei consulenti in ambito legale, che si occupassero di tutte le problematiche relative al fisco e alle normative di legge.

Il terzo QOC è più tecnico, riguarda quasi l'implementazione, tuttavia ho scelto di inserirlo fra i concettuali poichè è un problema che si è manifestato ben prima dell'inizio della fase di implementazione. La ricerca geografica è infatti uno dei task più importanti dell'applicazione per cui, si è deciso di scegliere da subito quale fosse la strada da prendere per affrontare questa problematica. Inoltre questa scelta avrebbe potuto influenzare in maniere importante i tempi di realizzazione, ad esempio se si fosse optato per l'utilizzo di una basi di dati geografica, questo avrebbe comportato una fase di apprendimento della tecnologia, che di certo non è del tutto banale. Alla fine si è scelto di utilizzare un approccio ibrido, ossia utilizzare api esterne esclusivamente per il geocoding e, invece effettuare le query spaziali attraverso query specifiche oppure attraverso il supporto dei dati spaziali offerto dai DB relazionali. Questa scelta che è più sensata e meno costosa in termini di tempi di sviluppo, è quella che meglio si adatta alle esigenze dell'applicazione.

Design Solution

Analisis Model

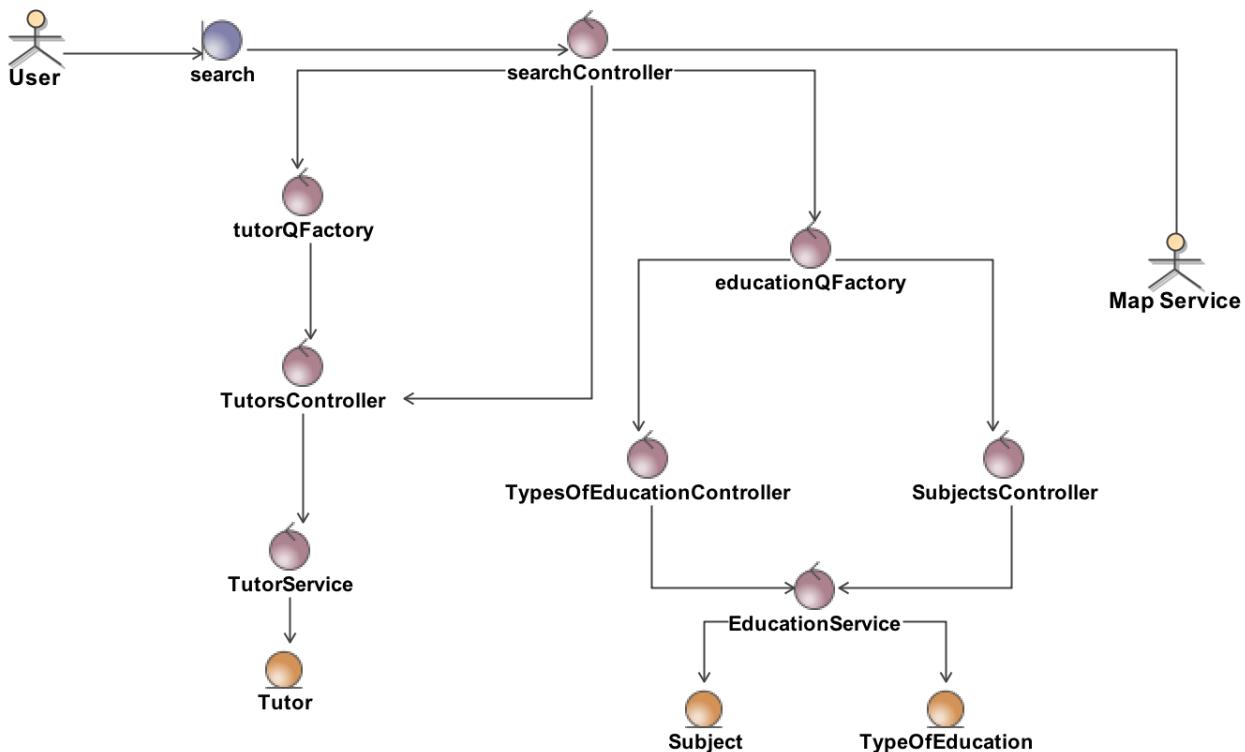
Oltre alla classica categorizzazione basata sull'approccio WAE di Conallen è stato applicata una convenzione per specificare ulteriormente le varie componenti, seguendo le seguenti regole:

- Le componenti stereotipate come control che hanno come suffisso Controller e che iniziano con lettera minuscola, indentificano i controller Angular che controlleranno la view con omonimo prefisso. Esse sono controller client-side.
- Le componenti stereotipate come control che hanno come suffisso Controller e che iniziano con una lettera maiuscola si occupano della gestione delle risorse, ossia di rispondere alle richieste REST.
- Le componenti stereotipate come control che hanno come suffisso QFactory, sono dei moduli che si occupano della gestione della business logic client-side (vedi QOC **Con#6Opt#2**). In particolare esse fungeranno da intermediari tra i controller Angular

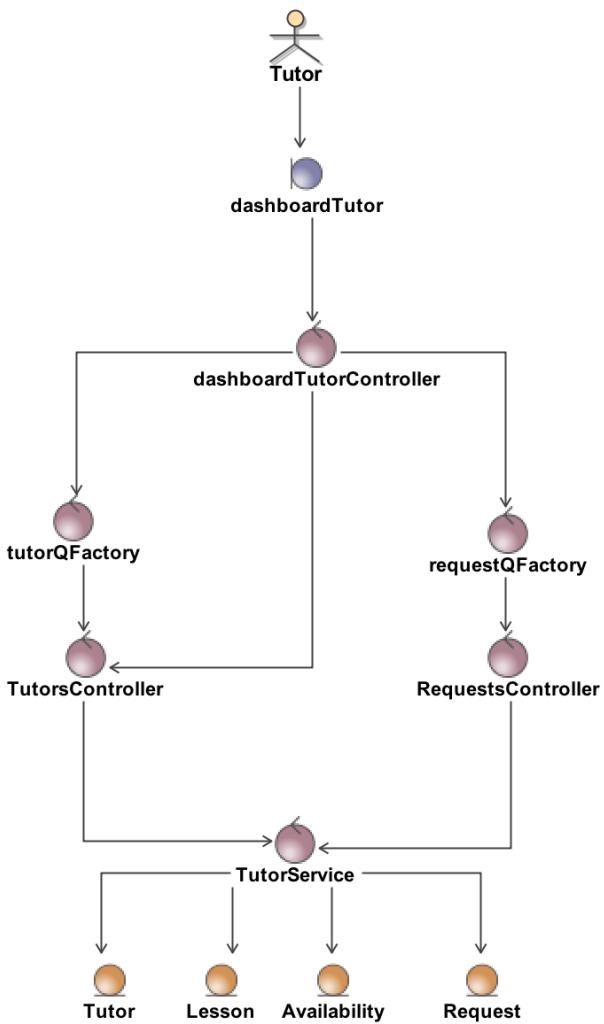
(client-side) ed i gestori di risorse (server-side). È stato scelto il nome QFactory poichè ciascuno di essi è una Factory Angular e poichè i metodi da essi forniti restituiscono delle Promise Q, che sono le Promise Harmony-like fornite da Angular.

- Le componenti stereotipate come control che hanno come suffisso Service sono le componenti server-side che forniscono l'accesso alle operazioni di business-logic, ossia che permettono il reperimento, l'aggiunta, la modifica e l'eliminazione degli Entity nel Database.

Nei modelli Magic Draw sono dettagliati tutti i diagrammi dell'Analysis Model relativi alle funzionalità dell'applicazione, con le relative informazioni sulla tracciabilità rispetto agli Use Case. Nella documentazione invece saranno riportati esclusivamente quelli più importanti e complessi. Di seguito viene riportato un diagramma che mostra una visione più dettagliata dello Use Case "Search Tutor" (vedere **Con#4Opt#3**). L'utente accede alla view "search", tale view è gestita dal controller "searchTutor", che attraverso i parametri passati nella URL ricava la località cercata e la traduce in coordinata geografica attraverso i Map Service; tra i parametri ci saranno anche le materie ed i livelli scolastici desiderati. Una volta ottenuta la coordinata essa potrà essere utilizzata insieme agli altri due parametri per ricavare, attraverso l'intermediario tutorQFactory, l'elenco delle url dei tutor ordinati per lontananza con la relativa distanza dal punto indicato. Una volta reperite le url il searchController accederà direttamente al TutorsController per reperire le informazioni di ogni singolo tutor. Attraverso educationQFactory invece verranno reperite le informazioni relative alle materie ed ai livelli scolastici. Tali informazioni saranno utilizzate per popolare le select delle searchbox. Infatti attraverso la view "search" l'utente potrà effettuare ulteriori ricerche che comporteranno esattamente la stessa procedura precedentemente elencata.

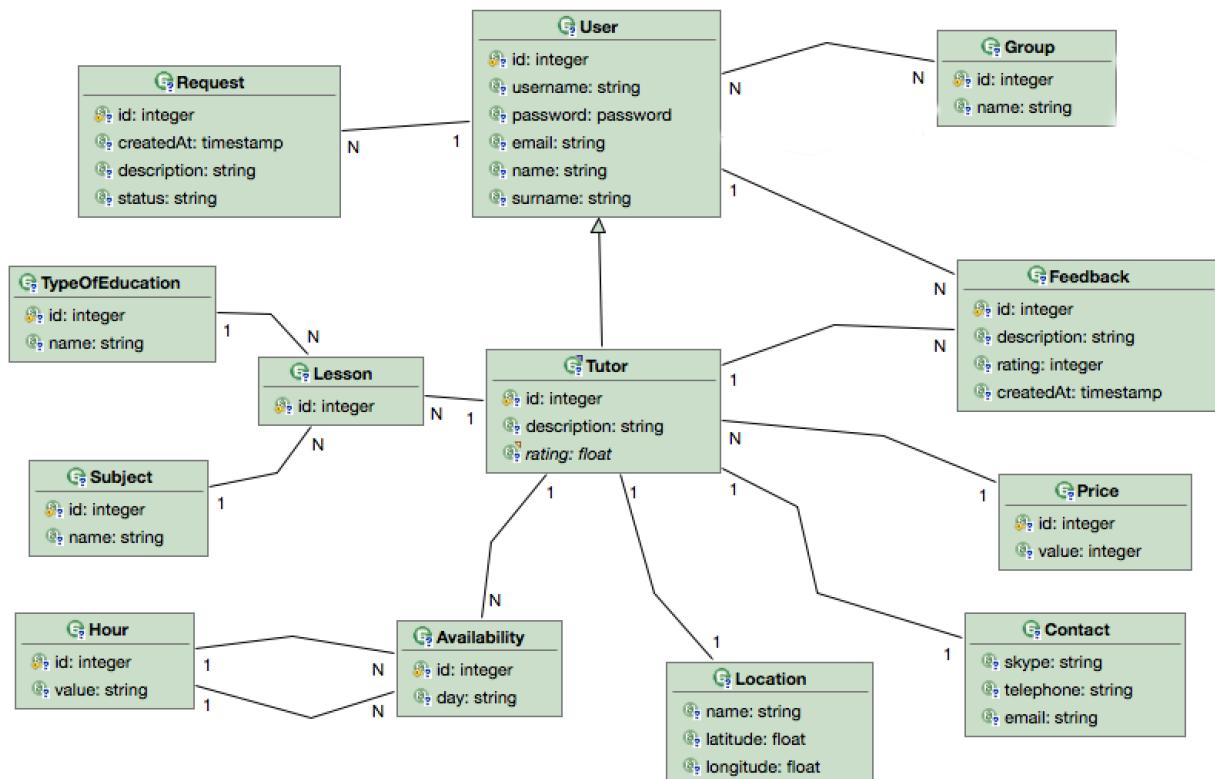


Il diagramma che segue tratta gli Use Case “Tutor General Info Management”, “Availabilities Management”, “Lessons Management” e “Received Request Management”, sostanzialmente esse raccolgono tutte le funzionalità della dashboard del tutor; la view che viene mostrata infatti è proprio “dashboardTutor” e secondo la convenzione di nomenclatura utilizzata il relativo controller è proprio “dashboardTutorController”. Attraverso “tutorQFactory” il controller accederà a dei metodi che permettono: l’eliminazione e l’aggiunta di lezioni o disponibilità (Lesson e Availability), la modifica della tariffa oraria, la modifica della descrizione e la modifica del domicilio. Il reperimento dei feedback verrà fatta direttamente dal controller tramite le url restituite da un dei metodi forniti da “tutorQFactory” (notare la freccia da “dashboardTutorController” a “TutorsController”). “requestQFactory” verrà invece utilizzato per recuperare tutte le richieste ricevute dal tutor che non hanno ancora avuto una risposta, inoltre esso fornirà dei metodi che permetteranno di accettare o rifiutare la richiesta dell’utente, come tutte le QFactory esso svolgerà una funzione di intermediario, in questo caso con RequestsController il quale attraverso TutorService renderà persistente le modifiche effettuata oppure recupererà le informazioni richieste.



Domain Model

Per rappresentare i dati di business dell'applicazione è stato utilizzato il Domain Model di IFML. Verrà riportata una minima spiegazione per le entità più ambigue. Nel modello realizzato nel tool WebRatio, c'è una tabella in più chiamata Module ed una relazione 1 a N tra User e Group, esse sono si default nel tool e devono essere ignorate. Nelle figure in basso tali elementi sono stati eliminati dall'immagine.



- **Request** : rappresenta una richiesta che viene inviata da un utente semplice verso un tutor. Alla creazione lo status di default è “Waiting”, a seguito della risposta del tutor esso può diventare “Accepted” o “Rejected”.
- **Availability**: rappresenta un intervallo di disponibilità giornaliero, è utilizzato per indicare gli orari giornalieri di disponibilità. Sarà indenfificato da un ora di inizio (*from Hour*), da un ora di fine (*to Hour*) e da un giorno (*day*) ossia uno dei giorni della settimana.
- **Lesson**: indica un tipo di lezione che il tutor è disposto ad impartire. Ad esempio:”il tutor imparte lezioni di Fisica per Superiori”. In questo caso “Fisica” indicherà la materia di riferimento (**Subject**) mentre “Superiori” sarà il livello scolastico (**Type Of Education**).

Tecnologie Client Side

Sviluppare una Single Page Application vuol dire delegare alla logica Client-Side il controllo della navigazione tra le pagine e la renderizzazione dei dati della business logic richiamati tramite chiamate REST. Per fare questo ovviamente è opportuno scegliere strumenti adatti che supportino lo sviluppatore, a questo scopo è stato utilizzato il framework Angular che fornisce un'architettura MVC all'applicazione. L'obiettivo è quello di realizzare un'applicazione modulare, ossia composta da una serie di pezzi di codice separati che implementano ciascuno funzionalità indipendenti. Scrivere applicazioni modulari incrementa manutenibilità e facilità di debugging del codice. Attualmente non esiste alcun modo per importare il codice attraverso JavaScript puro, Angular fornisce alcuni costrutti che permettono la creazione di moduli e di iniettarli all'interno di altri moduli, però sempre internamente alla propria tecnologia ed tra l'altro non permette il caricamento dinamico di codice. Per questo è opportuno utilizzare degli script loaders che supportino AMD o CommonJS, che sono allo stato attuale i formati più utilizzati per l'importazione e creazione di moduli in JavaScript. A questo scopo è stato utilizzato RequireJS che sfrutta il formato AMD, che fornisce allo sviluppatore un'ambiente di lavoro pulito, evitando l'inquinamento del namespace, permettendo un'importante separazione dei contenuti e gestendo in automatico la problematica delle dipendenze fra script. Oltre ad Angular e Require sono state utilizzate ulteriori librerie: Underscore che fornisce un gran numero di utilities come la manipolazione di array e oggetti, jQuery che rende meno ostica la manipolazione del DOM appianando le tediote differenze fra browser e Leaflet che fornisce delle api per una gestione semplice ed efficace delle mappe nell'applicazione fornendo un meccanismo che permette di astrarsi dai servizi di mappe dei quali si utilizzano esclusivamente i tiles, rendendo così molto più semplice il passaggio da un servizio ad un altro.

Tecnologie Server Side

A livello server side è stato utilizzato Spring Framework che implementa il pattern architettonico MVC. Spring utilizza la Dependency Injection ed attraverso essa ha permesso un totale disaccoppiamento tra interfacce e implementazioni, soprattutto per quanto riguarda la Business Logic; durante la fase di sviluppo si è partiti con una "implementazione banale" che restituiva oggetti a caso e poi, una volta che i Controller (API REST) sono state completati si è passati all'implementazione "reale" della Business Logic. Spring inoltre ha permesso lo sviluppo agevole delle API REST fornendo una maniera semplice ed intuitiva sia per catturare le richieste attraverso le Annotazioni Java sia per costruire le risposte canoniche alle chiamate REST. A Spring è stato integrato Jackson che fornisce un meccanismo totalmente trasparente per la trasformazione di Java Bean in JSON e viceversa. Questa infrastruttura ha permesso uno sviluppo più organizzato e flessibile, riducendo in maniera decisiva i tempi di sviluppo. Gran parte dei controller dell'applicazione sono controller REST, esistono tuttavia due che invece restituiscono JSP: AppController e RegisterController. La prima restituisce le index del portale o del back-end a seconda della url, invece la seconda si occupa delle registrazioni di utenti o di tutor mediante il classico submit di una form. L'autenticazione e la protezione delle URL avviano attraverso SpringSecurity

RESTful API Design

Una fase fondamentale e delicata della progettazione di una applicazione REST è la progettazione delle URL a cui attingere per effettuare le chiamate che restituiranno i dati dell'applicazione. Una buona progettazione in questo ambito può portare benefici sia al fruitore delle API e sia al realizzatore. Ovviamente è opportuno attingere allo stato dell'arte delle applicazioni REST ed essere il più standard possibile(seppur uno standard ufficiale non c'è) soprattutto quando stiamo parlando di canoniche richieste CRUD. Di seguito viene mostrata una tabella che riassume in che modo sono state strutturate le URL:

API Root : /teachify/rest

(Il metodo "list" restituisce l'elenco delle risorse stesse. Al contrario del metodo "retrieve" che restituisce invece un elenco di url utilizzabili per il reperimento delle risorse)

SUBJECTS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /subjects	All	Restituisce l'elenco delle materie scolastiche disponibili nel sistema
insert	POST /subjects	Admin	Inserisci una nuova materia
update	PUT /subjects/ <i>id</i>	Admin	Aggiorna la materia
delete	DELETE /subjects/ <i>id</i>	Admin	Elimina la materia

TYPES OF EDUCATION

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /typesofeducation	All	Restituisce l'elenco dei livelli scolastici disponibili nel sistema
insert	POST /typesofeducation	Admin	Inserisce un nuovo livello scolastico
update	PUT /typesofeducation/ id	Admin	Aggiorna il livello scolastico
delete	DELETE /typesofeducation/ id	Admin	Elimina un livello scolastico

PRICES

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /prices	All	Restituisce l'elenco delle tariffe che possono essere adottate dai tutor
insert	POST /prices	Admin	Inserisce una nuova tariffa
delete	DELETE /prices/ id	Admin	Elimina una tariffa

HOURS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /hours	All	Restituisce l'elenco delle ore che possono essere utilizzate dal tutor per indicare gli intervalli di disponibilità giornaliera
insert	POST /hours	Admin	Inserisce una nuova ora
delete	DELETE /hours/ id	Admin	Elimina una ora

TUTORS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
search	POST /tutors/search	All	Restituisce un elenco di oggetti, che contengono le uri dei tutor, ordinati rispetto alla vicinanza ad una coordinata geografica e che insegnano una particolare materia per un particolare livello scolastico, con la relativa distanza.
get	GET /tutors/ id	All	Restituisce un tutor
update	PUT /tutors/ id /description	Tutor	Aggiorna la descrizione di un tutor
update	PUT /tutors/ id /location	Tutor	Aggiorna la posizione geografica del tutor
update	PUT /tutors/ id /price	Tutor	Aggiorna la tariffa oraria di un tutor

AVAILABILITIES

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
insert	POST /tutors/ <i>id</i> /availabilities	Tutor	Aggiunge una disponibilità al tutor
delete	DELETE /tutors/ <i>tutorid</i> /availabilities/ <i>id</i>	Tutor	Elimina una disponibilità al tutor

LESSONS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
insert	POST /tutors/ <i>id</i> /lessons	Tutor	Aggiunge una lezione del tutor
delete	DELETE /tutors/ <i>tutorid</i> /lessons/ <i>id</i>	Tutor	Elimina una lezione del tutor

FEEDBACK

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
retrieve	GET /tutors/ <i>tutorid</i> /feedback	All	Restituisce l'elenco delle uri di tutti i feedback ricevuti da un tutor
insert	POST /tutors/ <i>tutorid</i> /feedback	Student	Inserisce un feedback verso un utente
get	GET /tutors/ <i>tutorid</i> /feedback/ <i>id</i>	All	Restituisce il feedback identificato dall'id

REQUESTS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /requests?status=waiting&tutor= tutorid	All	Restituisce le richieste ricevute non ancora gestite dal tutor
list	GET /requests?user= userid	All	Restituisce le richieste effettuate da uno studente
insert	POST /requests	Logged	Inserisce una richiesta verso un tutor
update	PUT /requests/ id /status	Tutor	Aggiorna lo stato della request, viene utilizzata solitamente per accettare o rifiutare una richiesta dello studente
isrequested	POST /requests/isrequested	Logged	Nel payload è richiesto un oggetto di tipo Tutor. Verifica che l'utente in sessione abbia già effettuato una richiesta verso il tutor. Ritorna lo stato della richiesta come una stringa (Accepted, Rejected, Waiting) oppure se nn è ancora stata effettuata una richiesta restituisce "204 no content".

PROFILE

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
get	GET /profile	Logged	Restituisce il profilo dell'utente loggato
update	PUT /profile	Logged	Aggiorna il profilo dell'utente loggato

ADMINISTRATORS

METODI	RICHIESTA HTTP	PERMESSI	DESCRIZIONE
list	GET /admins	Admin	Restituisce tutti gli amministratori del sistema
insert	POST /admins	Admin	Inserisce un nuovo amministratore
delete	DELETE /admins/ id	Admin	Cancella un amministratore

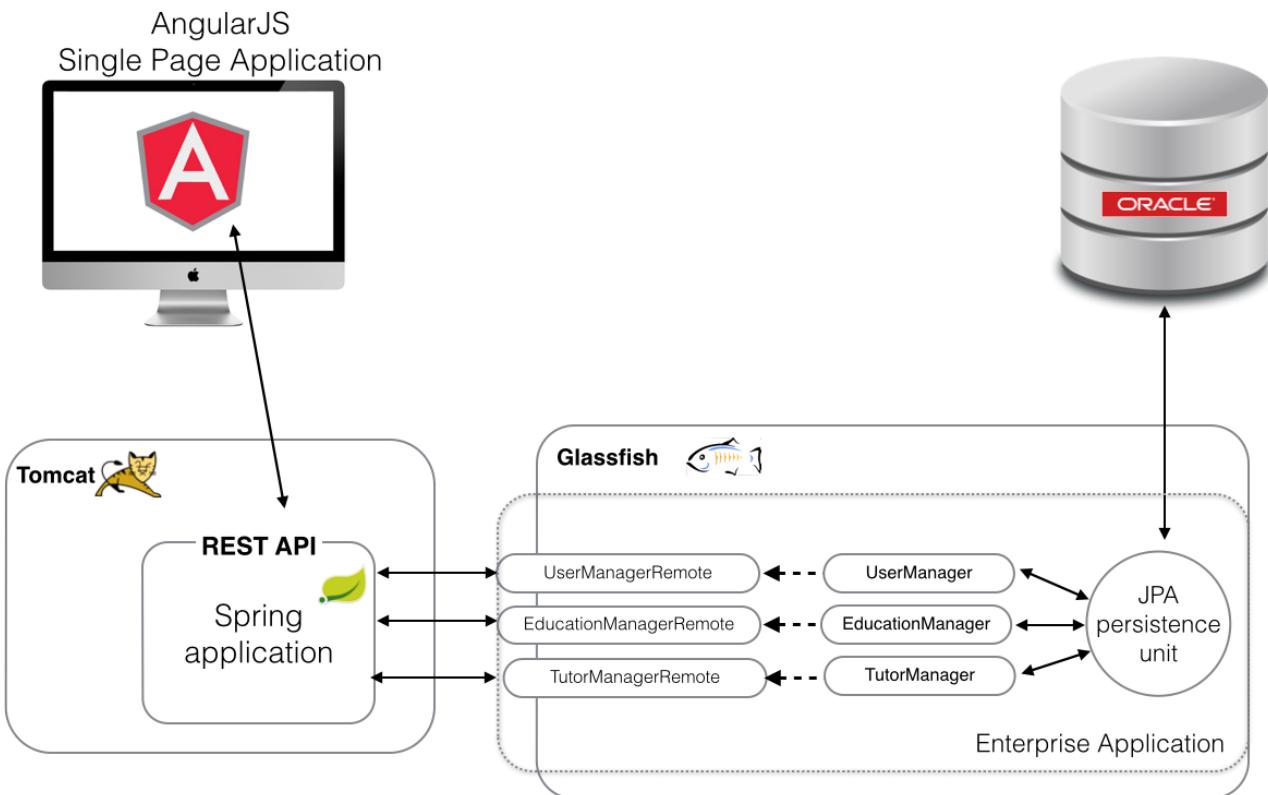
Queste documentazione fornisce un contratto tra i team che lavorano nel Front-End e quelli che lavorano nel Back-End, in tal modo essi potranno lavorare separatamente tenendo esclusivamente fede alle direttive fornite in queste tabelle.

Business Logic e ORM

L'applicazione sfrutta degli EJB remoti per la business logic. Tali EJB(Session Bean Stateless) sono incapsulati all'interno di una Enterprise Application deployata all'interno dell'application server Glassfish. I Session Bean utilizzano JPA per l'accesso, la persistenza e la gestione dei dati tra oggetti Java e database relazionali.

I Session Bean sono tre: UserManager, EducationManager e TutorManager; ciascuno di essi espone un'interfaccia utilizzabile da remoto: UserManagerRemote, EducationManagerRemote, TutorManagerRemote. Di seguito vengono elencate le rispettive competenze :

- **UserManager:** Fornisce tutti i metodi per la gestione degli User. Dal reperimento degli oggetti all'assegnazione e rimozione dei permessi da amministratore. Oltre che ovviamente i canonici metodi per la creazione e aggiornamento dell'istanza.
- **EducationManager:** Fornisce i metodi per il reperimento, l'aggiunta, la modifica e l'eliminazione di istanze delle seguenti classi: Subject, TypeOfEducation, Price e Hour.
- **TutorManager:** Fornisce i metodi per la creazione e la modifica di Tutor. Attraverso questo SessionBean è anche possibile aggiungere e rimuovere istanze di Lesson e di Availability, e, creare e reperire Request.



Per il mapping degli oggetti nel mondo relazionale come già detto è stato utilizzato JPA. Ovviamente come è noto i principali problemi legati al mapping sono sostanzialmente tre: l'aggregazione, l'ereditarietà e le associazioni. I mapping banali non verranno trattati.

User

La classe *User* è mappata nella tabella *USERS*. *User* è inoltre classe padre di *Tutor*. La scelta relativa al tipo mapping da utilizzare per l'ereditarietà è stata senza dubbio una delle parti più complesse e delicate della fase di implementazione (**Con#5**). La scelta di un mapping One Inheritance Path One Table (Table per Class) è stata scartata poiché comportava una serie di complicazione nella gestione e manutenibilità delle relazioni dell'entità padre sulle figlie. In una prima fase si era scelto il pattern di mapping Flat che, pur sprecasse memoria, diminuiva in maniera sostanziale i join necessari per il reperimento dei tutor nel main task dell'applicazione, ossia la ricerca di tutor in base alla posizione. Tuttavia durante lo sviluppo sono state effettuate delle modifiche sulla query, in particolare si è scelto che gli oggetti restituiti non dovessero essere *Tutor*, bensì *TutorInfo* (coppia id-distanza), a questo punto l'utilizzo del pattern One Class One Table (Joined) non avrebbe comportato join eccessivi visto che non è più necessario ricostruire l'intera istanza di *Tutor*, inoltre il pattern avrebbe inciso in maniera decisiva nell'ottimizzazione della memoria. Quest'ultimo è stato considerato il pattern più adatto alle esigenze dell'applicazione.

Tutor

Sui tutors si effettuano query spaziali per cui nella tabella ci sarà una colonna in cui verranno riportati i dati che permettono il supporto di tali query. Ovviamente JPA non supporta tali dati

per cui, a questo punto sarebbe stato necessario utilizzare degli strumenti (come JGeometry) che permettessero di gestire i dati spaziali all'interno del framework di persistenza (**Con#3**). Tuttavia le query che li utilizzano sono sostanzialmente due: aggiornamento della latitudine e longitudine del Tutor e il reperimento dell'elenco degli id dei tutor con la relativa distanza da un determinato punto geografico. A seguito di questa considerazione si è deciso utilizzare un'approccio più rudimentale utilizzando delle NamedNativeQuery. Nel caso dell'aggiornamento dei dati spaziali ovviamente non si fa che modificare la latitudine e longitudine attraverso una classica Update.

Per la seconda query la situazione è più complessa: sarebbe eccessivo reperire tutti i dati di un singolo tutor se poi viene utilizzato esclusivamente l'id. Per cui si è deciso di creare un'altra entità chiamata TutorInfo. Tale entità non è mappata da nessuna tabella concreta, bensì essa è mappata da una query, ciò è possibile attraverso l'annotation di JPA SqlResultSetMapping, che permette di indicare la classe dell'oggetto che verrà restituito dalla query indicando in che modo le colonne si mappano con i membri della classe, il mapping avrà un nome a cui si farà riferimento nella dichiarazione della NamedNativeQuery che effettua la query spaziale.

Per la modifica del Tutor l'EJB TutorManager non offre il canonico metodo "updateTutor", bensì si è deciso di utilizzare dei metodi più puntuali come "updateTutorDescription", "updateTutorContact", "updateTutorLocation" e "updateTutorPrice". Questa scelta è nata dalle esigenze dell'applicazione, in particolare questo è dovuto dal fatto che le modifiche avvengono attraverso chiamate REST. Facendo un esempio se si deve modificare la descrizione del tutor questo avverrà attraverso una chiamata ajax all'URL [teachify/rest/tutors/{id}/description](#) con nel payload la stringa con la nuova descrizione. A questo punto appare ovvio che non ha senso effettuare la modifica dell'intero tutor visto che la modifica è così puntuale. Ovviamente è opportuno tener sempre presente la problematica del round-trip delle reti, infatti tale approccio è stato utilizzato esclusivamente per i quattro metodi precedentemente elencati.

Lesson

Lesson è mappato nella tabella *Lessons*. Ad ogni *Lesson* è associata un *Subject* ed un *TypeOfEducation*. Tali associazioni vengono realizzate attraverso l'annotation ManyToOne. Per evitare violazioni di constraint è necessario che prima che venga eliminata un materia o un livello scolastico debbano essere eliminati "manualmente" tutti gli oggetti *Lesson* associati, ciò è infatti quello che avviene nei metodi "deleteSubject" e "deleteTypeOfEducation" del Session Bean *EducationManager*.

Availability

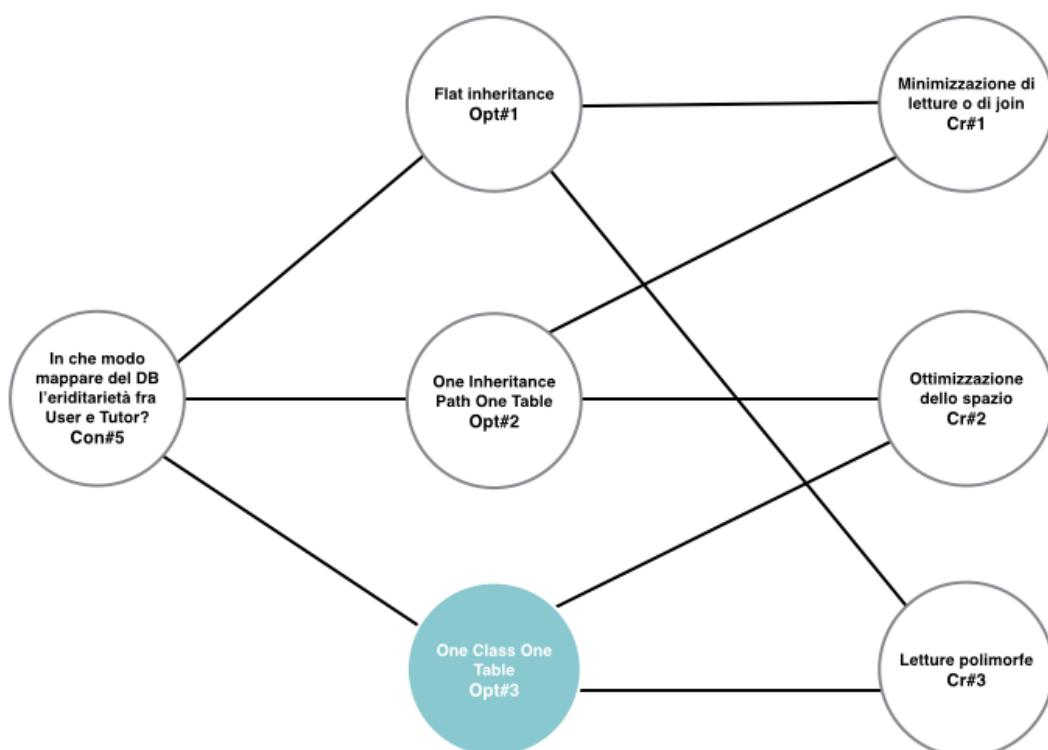
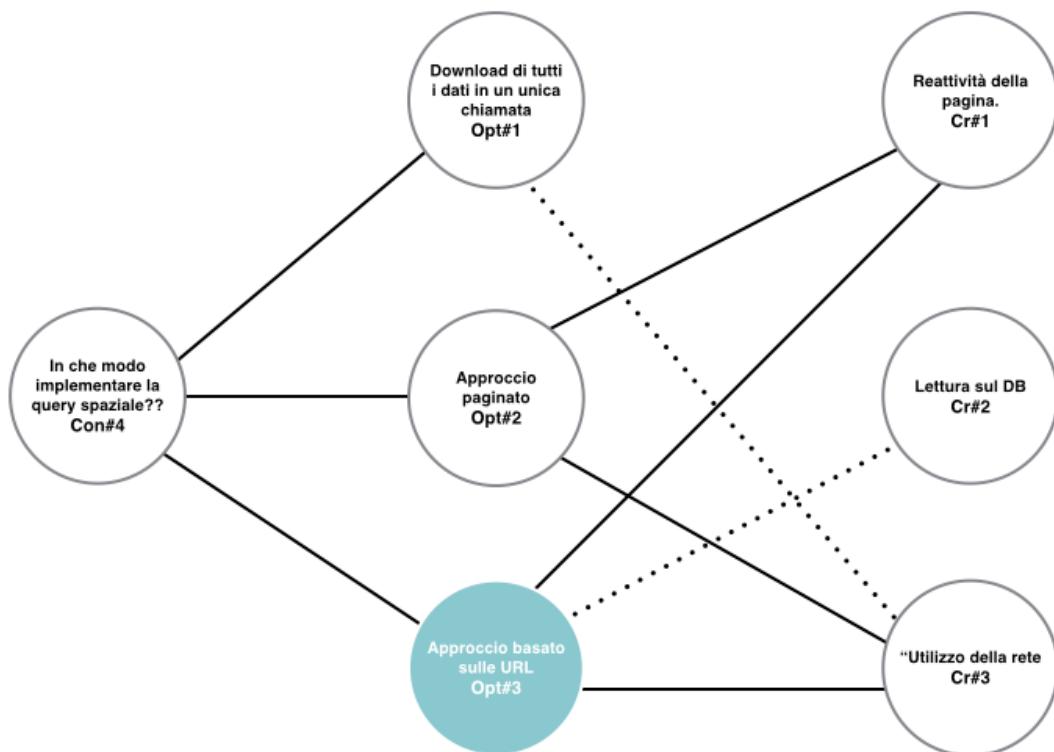
Availability è mappato nella tabella *Availabilities*.

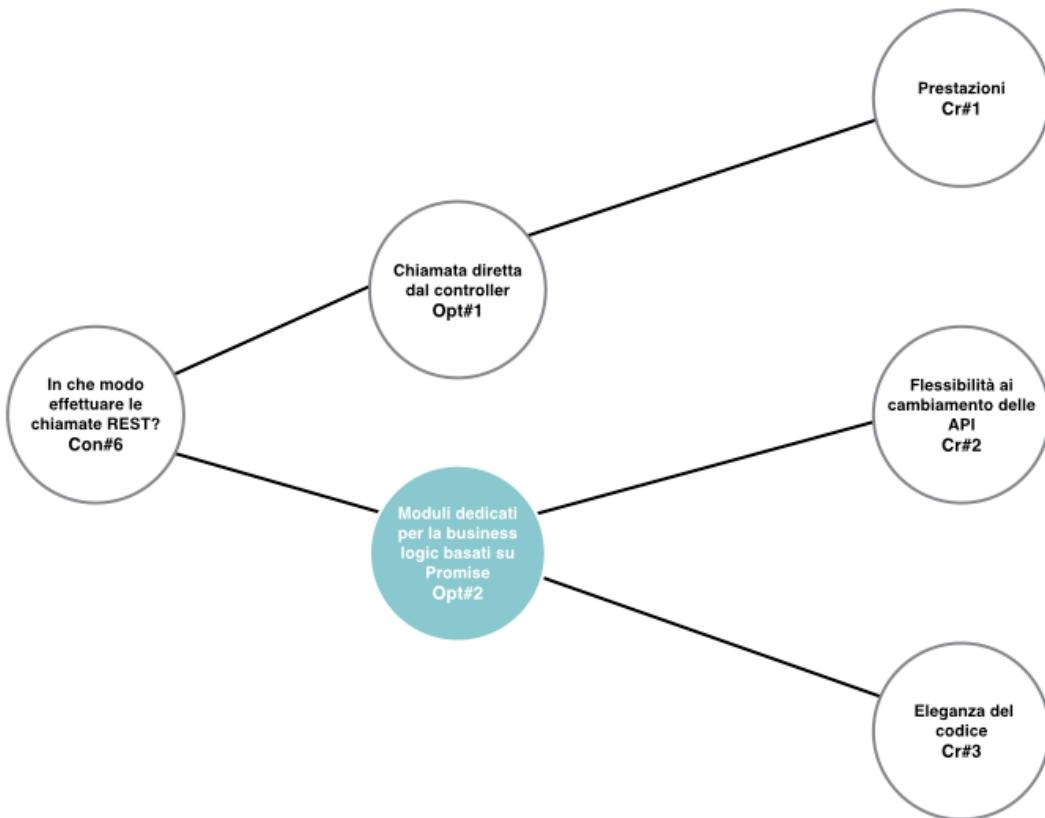
Ogni oggetto di questa classe ha una doppia relazione con l'entità Hour, entrambe le relazioni sono di tipo ManyToOne.

Anche in questo caso se si volesse rimuovere un oggetto Hour sarebbe necessario rimuovere tutte *Availability* che sono collegate all'oggetto (metodo "deleteHour" di *EducationManager*).

Technical Design Decision

I QOC riguardanti le scelte tecniche effettuate sono riportati nel file QOC Technical.xls, di seguito viene mostrata una sintetica rappresentazione grafica:





Sono state scelte queste come Technical Decision più importanti poichè sono state quelle più lungamente confutate. Le prime due sono state oggetto di refactoring durante la fase di sviluppo. Infatti durante una prima fase si era deciso per il **Con#4#Opt#1**, ossia la restituzione di tutti i tutor che soddisfano la richiesta, questo approccio che poi durante lo sviluppo si è dimostrato molto poco efficiente aveva influenzato la strategia ORM con conseguente utilizzo della Flat Inheritance(**Con#5#Opt#1**) per il mapping dell'ereditarietà fra User e Tutor. Quando si è deciso di cambiare la strategia per implementare la query spaziale, ci è resi conto che l'approccio ORM utilizzato che era stato pesantemente influenzato dal **Con#4#Opt#1** era totalmente inadeguato, a seguito di questi cambiamenti è stato opportuno analizzare la nuova dinamica di chiamata e le ripercussioni che le varie strategie di mapping avrebbero avuto su questo meccanismo. Al termine dell'analisi si è deciso di utilizzare la **Con#5Opt#3**, l'approccio One Class One Table, che è sembrata la più adatta in termini di prestazioni e di ottimizzazione della memoria. Il terzo QOC invece è stato fra i primi ad essere preso in considerazione visto che lo sviluppo è partito proprio dal Front-End dell'applicazione. Ci si è resi conto da subito che era necessario prendere una decisione omogenea per tutta l'applicazione che avrebbe influenzato lo sviluppo di tutte le componenti della business logic client-side. Tale decisione ha avuto ripercussioni molto positive durante lo sviluppo, infatti anche i cambiamenti che venivano effettuate nelle API e nelle strutture dei dati restituite non andavano a coinvolgere un gran numero di controller, bensì comportavano modifiche lievi e ben centralizzate nei moduli della business logic.

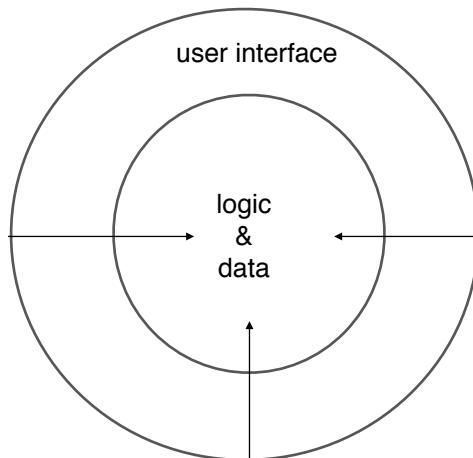
Effort Recording

Learning	Doing
<ul style="list-style-type: none">• ripasso use cases (2 giorni)• ripasso del visual vocabulary (2 giorni)• studio approccio di conallen (1 giorni)• studio documentazione google api (1 giorni)• studio modellazione tool(3 giorni)	<ul style="list-style-type: none">• stesura requisiti e use cases(5 giorni)• realizzazione structure, skeleton e surface (15 giorni)• realizzazione analysis model (15 giorni)• design e documentazione API (7 giorni)• design e modellazione sul tool del domain model (5 giorni)• design business logic e ORM con relativa documentazione (9 giorni)

Vedi il file Log.xls

Final Consideration

La documentazione e la produzioni di artefatti che supportino e guidino durante la realizzazione di un progetto, ha a mio avviso un impatto molto positivo sulla qualità del prodotto finale; facilita infatti la manutenibilità del codice ed se utilizzata con cognizione di causa può anche velocizzare i tempi di sviluppo.



Per lo sviluppo è stato utilizzato l'approccio chiamato “Designing from the outside in”, ossia partire dalla User Interface e poi lavorare all’indietro realizzando le funzionalità a livello implementativo, ovviamente il processo non è stato lineare, infatti in alcuni casi è stato necessario effettuare modifiche a livello di User Interface a causa di una eccessiva complessità riscontrata nell’implementazione.

Prima che l’applicazione iniziasse a prendere forma, sono stati identificati quali fossero gli utenti tipo, delineandone le caratteristiche. Per avere una visione più precisa degli utenti sono

state ideate delle personas relative alle classificazioni trovate, in modo da tener presenti le loro caratteristiche per le scelte progettuali.

Dopodiché sono stati definiti i requisiti prioritizzati e gli Use Case con le relative informazioni di tracciabilità, in modo di definire le caratteristiche dell'applicazione. Le priorità sono state fondamentali nella scelta dei requisiti da scartare durante l'implementazione. Quindi si è iniziati a realizzare la progettazione della User Experience dell'utente, che a mio avviso è stata una fase fondamentale; questa fase comprende la realizzazione dei Lo-Fi Wireframe, dell'Interaction Design e degli Hi-Fi Wireframe, questi artefatti hanno permesso di avere una visione più concreta dell'applicazione che si stava realizzando, definendo lo "spartito" da rispettare durante la fase implementativa. Inoltre gli Hi-Fi Wireframe oltre ad avere una valenza documentale sono stati anche riutilizzati durante la fase d'implementazione, visto che sono stati realizzati in HTML5 e CSS3. Coerentemente all'approccio top-down utilizzato nella progettazione anche la fase implementativa è iniziata dal Front-End, prima attraverso la realizzazione della navigazione tra le view della Single Page Application, e successivamente con la realizzazione dell'accesso ai dati della Business Logic attraverso chiamate REST. Per sostenere quest'ultima fase sono state create delle API fittizie che restituivano json a caso ma del formato corretto, per far questo è stato necessario iniziare a definire le classi del Domani Model. Grazie ai meccanismi automatici di serializzazione in JSON dei Java Bean, il formato degli oggetti JSON è stato determinato dalla struttura delle classi del Domani Model. Una volta ultimato il Front-End si è potuti passare all'implementazione lato Server, ossia le API REST e la Business Logic. Parallelmente all'implementazione per avere una prospettiva più diretta e essenziale delle componenti software Client e Server e delle loro relazioni, è stato utilizzato l'Analysis Model di Conallen con il supporto di MagicDraw, che grazie alla tracciabilità tra i modelli permette refactoring non onerosi in caso di modifica o aggiunta di componenti. Quest'ultimo è un aspetto fondamentale poiché permette di risalire da un componente al requisito di riferimento percorrendo la catena di tracciabilità o viceversa, sia in caso di problemi che in caso di modifiche ai requisiti dell'applicazione.

Inoltre per distinguere le diverse componenti sono state introdotte delle convenzioni di nomenclatura che permettessero di individuare immediatamente il ruolo delle diverse componenti. Oltre questo, sono stati descritte più in dettaglio le parti implementative più delicate come la descrizione delle API REST, e le scelte più delicate effettuate nel mapping ORM considerando i rischi e le problematiche che sarebbero potute insorgere. In conclusione posso affermare che la produzione di questi artefatti ha di fatto un ottimo impatto sul processo di produzione del software sia in termini di qualità del prodotto finale che in termini di manutenibilità in prospettive future.