ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

ANALISI, PROGETTAZIONE E REALIZZAZIONE DI UN PROTOTIPO DI SCHEDULER PER ATTIVITÀ COMMERCIALI

Relazione finale in:

BASI DI DATI

Relatore

Chiar.mo Prof. Dario Maio

Presentata da

Giulia Tardozzi

Correlatore

Dott.ssa Annalisa Franco

Sessione II

Anno Accademico 2012 – 2013

SOMMARIO

1
INTRODUZIONE

La realizzazione di un prototipo di scheduler per attività commerciali nasce da un'idea, durante la mia collaborazione con l'azienda Librasoft snc che si occupa di gestione software per piccole e medie imprese. Essa mira ad automatizzare il sistema di appuntamenti con il pubblico di alcuni tipi di attività. Tutto ciò è stato ispirato dall'utilizzo sempre più incalzante della tecnologia nella società attuale a tal punto da avere la necessità di ridurre l'utilizzo delle comunicazioni telefoniche, sostituendole con un sistema online.

L'obiettivo di questo lavoro di tesi è la realizzazione di tale sistema: verrà prestata particolare attenzione anche all'aspetto grafico, dirigendosi verso un layout moderno e che sfrutta le tecnologie 2.0 oltre alla corretta ed efficace gestione dei dati.

La realizzazione di tale sistema informativo prevedrà quindi una prima fase di analisi dei requisiti e di progettazione del dominio applicativo, in modo da ricostruire il sistema informativo globale.

Nell'ultima fase verrà realizzata l'applicazione web per fornire all'utente un'interfaccia che gli consenta di interagire con la base di dati progettata.

Tramite varie interviste su un campione di 56 attività nell'area di Forlì-Cesena è stato possibile analizzare tutti gli aspetti e le esigenze che caratterizzano l'attuale sistema informativo, in modo da poter assecondare la maggior parte

delle richieste, al fine di rendere il prototipo più malleabile possibile, in modo che si adatti a più categorie commerciali possibili.

La relazione si articola in diversi capitoli.

Il primo presenta un'introduzione generale al progetto.

Nel secondo capitolo si procede con l'intervista al committente e con l'analisi dei requisiti del sistema da modellare, che descrive nel complesso il problema analizzato e le funzionalità richieste.

Nel terzo capitolo si passa alla progettazione graduale dello schema concettuale, rappresentato tramite il formalismo Entity/Relationship e parallelamente tramite un formalismo più esplicativo del framework Cakephp del quale mi servirò per realizzare il mio applicativo web.

Nel quarto capitolo sono definite le scelte delle tecnologie utilizzate.

Nel quinto capitolo è presentato il sistema interamente realizzato con la presentazione del codice delle query più interessanti dell'applicazione.

Nel sesto e ultimo capitolo vengono infine presentate le conclusioni e alcuni possibili sviluppi futuri.

2
ANALISI DEI REQUISITI

2.1 Premessa

La realizzazione di uno scheduler destinato alle attività commerciali, ma non solo, nasce da un'idea. L'idea è la seguente: *"Sono a casa, decido che voglio andare a tagliarmi i capelli, ma non conosco il numero del mio parrucchiere. Allora guardo in internet, sul sito, cerco il numero di telefono, controllo gli orari di apertura e chiamo. Se tutto va bene e l'orario da me prescelto è libero, allora prenoto."*

Non sarebbe più semplice e comodo se si potesse saltare tutti i passaggi per arrivare direttamente alla prenotazione?

È proprio qui che nasce l'idea di realizzare un sistema in grado di fare ciò. Ma facciamo un passo indietro.

All'inizio del progetto la mia attenzione si è soffermata sullo studio di fattibilità e sull'impatto che questa applicazione poteva avere a livello di mercato e se, conseguentemente, valesse la pena sviluppare un'applicazione di questo tipo. Così, insieme all'agente commerciale dell'azienda con cui ho collaborato, ossia Librasoft snc, abbiamo stilato una lista di domande da rivolgere ai vari commercianti del settore specifico durante la nostra prima fase di analisi dei requisiti. Questo ci ha permesso di capire che il mio prototipo poteva essere fruibile dal mercato.

2.2 Preparazione del questionario

Dopo un'accurata discussione su ciò che poteva essere chiesto ai commercianti in modo da non entrare troppo nelle specifico finendo per confonderli, ma neanche lasciando troppi concetti vaghi, rischiando di non far comprendere l'obiettivo e le caratteristiche della nostra applicazione, è stata stilata una lista di domande **[1]**.

Tabella 1 - Questionario

#	DOMANDA
1	In media, quanti appuntamenti prende in un giorno?
2	Qual è l'età media dei suoi clienti?
3	Accetta appuntamenti anche molto lontani nel tempo?
4	Quali sono i metodi di prenotazione? Al telefono e di persona? In che rapporto sono? Sono di più quelli che prende telefonicamente o di persona?
5	Quanto tempo impiega, in linea di massima, per prenderne uno? a. Al telefono b. Di persona
6	Qual è la durata media di un appuntamento?
7	È mai capitato che un cliente disdicesse un appuntamento all'ultimo momento e quindi non riuscisse a coprire il "buco", perdendo così tempo lavorativo e l'intero importo del trattamento che lei aveva riservato per quel cliente?
8	Lei utilizza il computer e internet?
9	Ha mai pensato di utilizzare uno strumento online per la gestione degli appuntamenti?
10	Sa cos'è Facebook? Potrebbe, a quel punto servirsi della grande diffusione di Facebook, ad esempio mettendo un annuncio del tipo: "sono le 17.40, alle 18 ho un posto libero per una manicure, il primo che prenota commentando questo post, potrà usufruire del servizio al 50% del prezzo da listino." Sarebbe disposto a questo?

	Cioè, invece che perdere l'intero guadagno, sarebbe disposto a guadagnare meno?
11	La stessa cosa può essere sfruttata in qualunque momento, così da non rimanere fermi ed avere magari nuovi clienti che, sfruttando l'occasione, poi potrebbero sempre appoggiarsi a lei. Sarebbe disposto a rinunciare a ridurre del 50% un guadagno nell'immediato, acquisendo però un nuovo cliente che potrà fruttarle in futuro?
12	Ha mai pensato di fare una sorta di lista "clienti fedeltà"? Potrebbe ad esempio, vendere un trattamento scontato oppure regalare un prodotto ogni 10 trattamenti svolti? Così il cliente si sente più propenso a venire più spesso.
13	Oppure effettuare periodiche offerte, solamente per chi stampa un foglio online?
14	Invece per quanto riguarda le prenotazioni online: se durassero 2 o 3 minuti? Immaginando di fare una decina di telefonate al giorno, essendo necessari 2 o 3 minuti per ciascuna, diventa 2-3 x 10 = 20- 30, cioè da 20 a 30 minuti persi per la prenotazione, tempo che potrebbe essere impiegato per un altro trattamento.
15	Inoltre, ha mai pensato che un trattamento possa non fruttarle al massimo per quanto riguarda il rapporto tempo/costo? Ad esempio, se un trattamento che costa 30 euro le impiega 45 minuti, ma nel suo listino ha trattamenti sempre da 30 euro ma che durano 15 minuti, lei potrebbe preferire quei 3 trattamenti che le permetterebbero di guadagnare 3 volte tanto.
16	In più, ha mai pensato che alcuni trattamenti, siccome molto dispendiosi dal punto di vista di prodotti utilizzati, in realtà non le fruttino come dovrebbero? Ossia che le possa costare di più mantenere i prodotti "freschi" in magazzino, piuttosto che il guadagno che essi portano? Tramite un sistema, si possono fare delle statistiche analizzando le prenotazioni , in modo da poterle proporre la soluzione ideale per la sua attività.

17	Invece, per quanto riguarda l'emissione della ricevuta fiscale? Quanto tempo impiega per compilarla? Se le bastasse spingere un bottone che le permette di fare tutto in automatico, così da farle risparmiare tempo ed evitare eventuali errori?

Fra queste, dopo varie considerazioni, ne sono state selezionate tre. Questo, sia allo scopo di centrare l'obiettivo, sia a quello di ottenere risposte più esaurienti da parte degli intervistati, i quali diversamente, trovandosi sommersi da troppe domande, sarebbero potuti diventare non disponibili nei nostri confronti. Inoltre, attraverso poche domande, ma ben indirizzate, è possibile analizzare anche il linguaggio "non verbale", il quale può diventare un ottimo indicatore di interesse o no da parte del cliente. Le domande selezionate sono le numero 9, 10 e 12.

2.3 Raccolta dei dati e statistiche

La raccolta dei dati è stata condotta su un campione di 56 attività commerciali nella zona di Forlì-Cesena, concentrandosi su parrucchieri e centri estetici.

Sulla base dei dati raccolti, sono state effettuate delle statistiche, rappresentate dai seguenti grafici.

Per quanto riguarda lo strumento di prenotazione online:

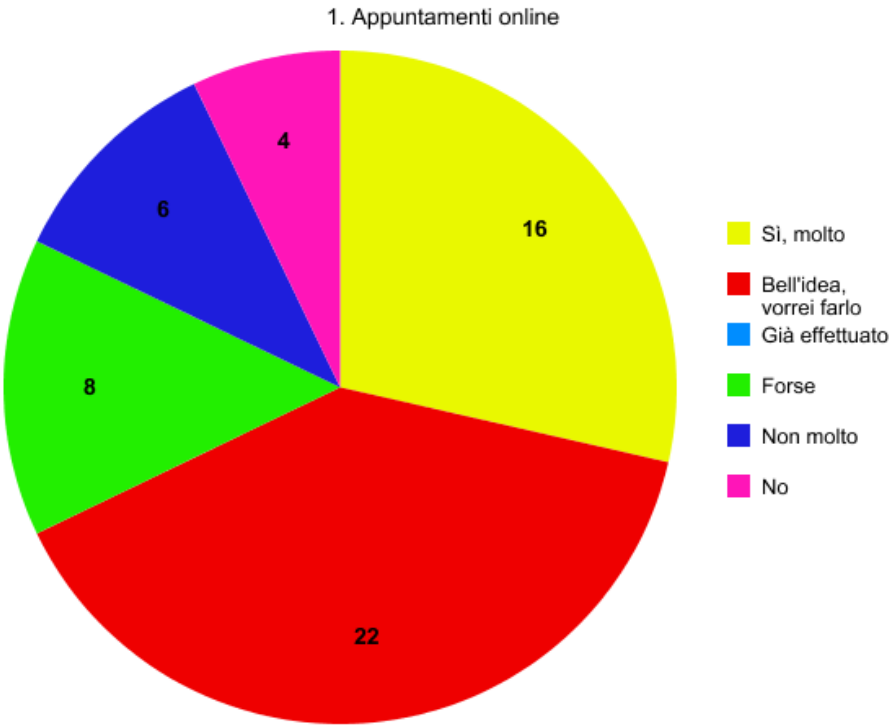

Figura 1 - Appuntamenti online

Quindi la percentuale dei giudizi risulta la seguente:

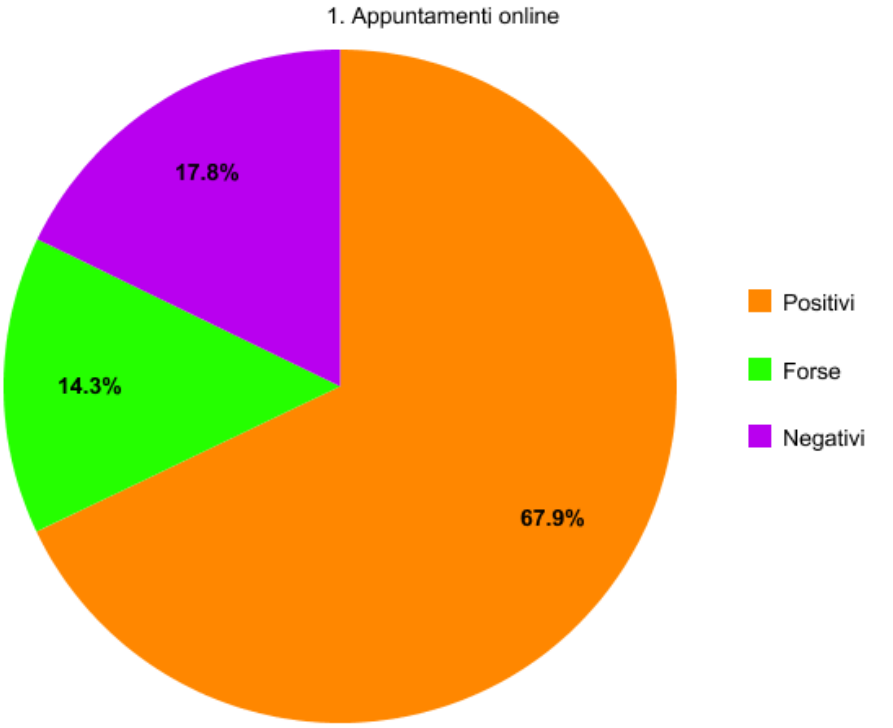

Figura 2 - Appuntamenti online in percentuale

Per quanto riguarda la prenotazione tramite "Facebook":

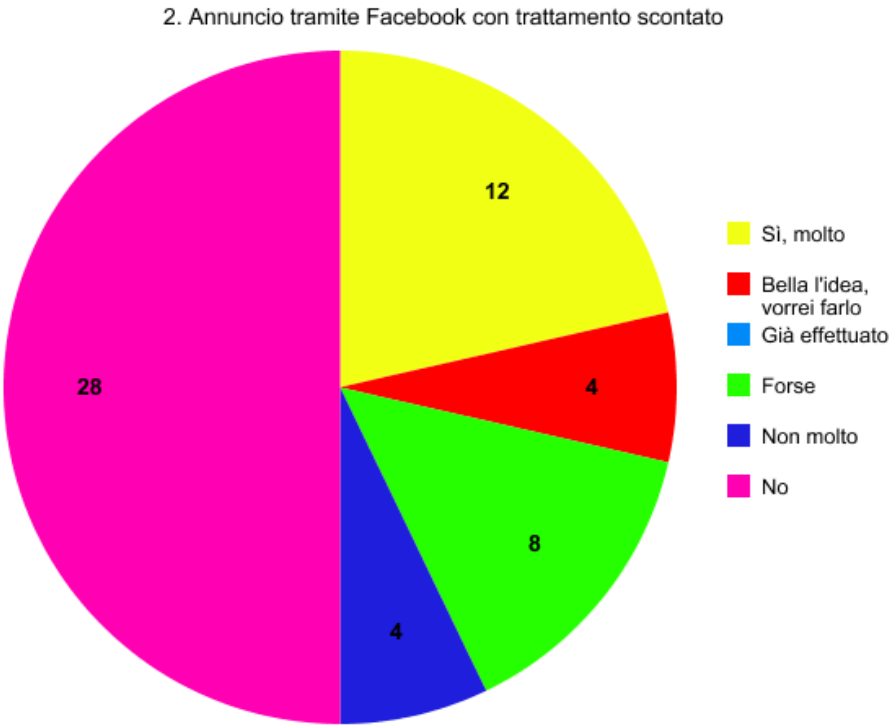

Figura 3 - Utilizzo Facebook

Quindi la percentuale dei giudizi risulta la seguente:

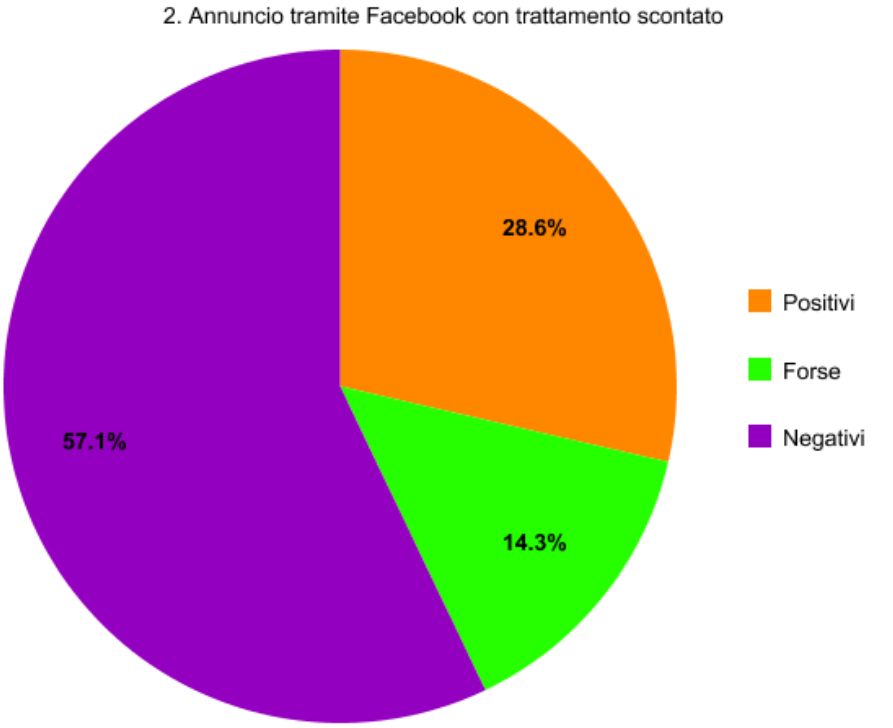

Figura 4 - Utilizzo Facebook in percentuale

Per quanto riguarda la “fidelizzazione” dei clienti:

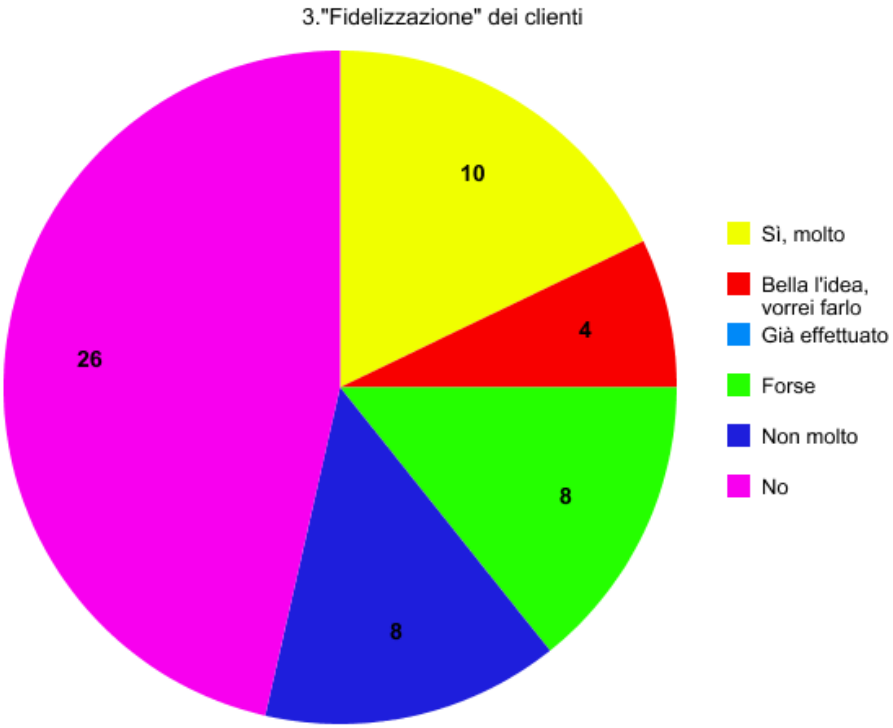

Figura 5 - Fidelizzazione

Quindi la percentuale dei giudizi risulta la seguente:

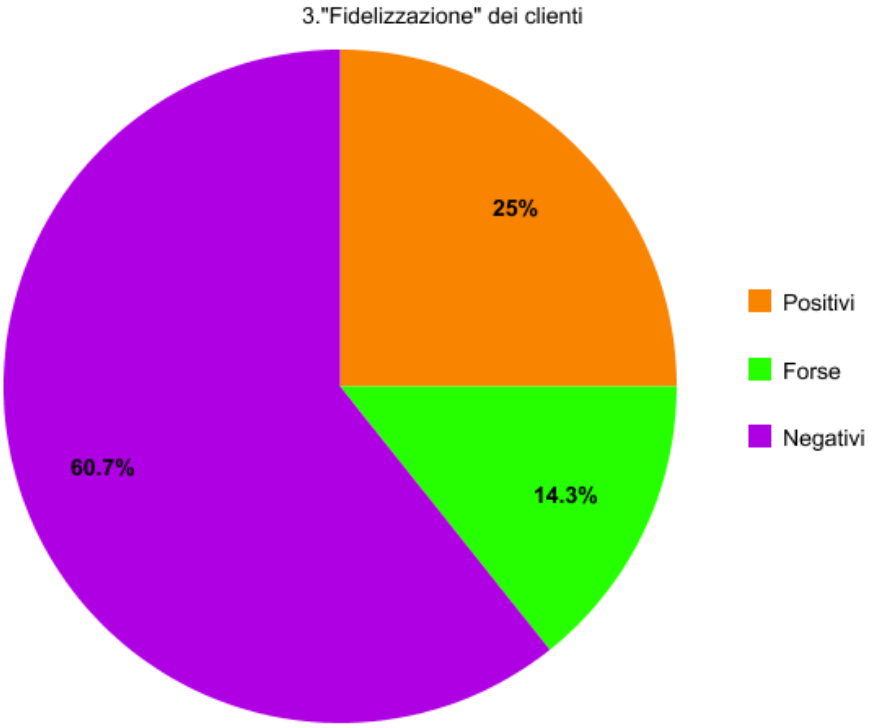

Figura 6 - Fidelizzazione in percentuale

Concludendo, riassumiamo ciò che è emerso dalla nostra statistica:

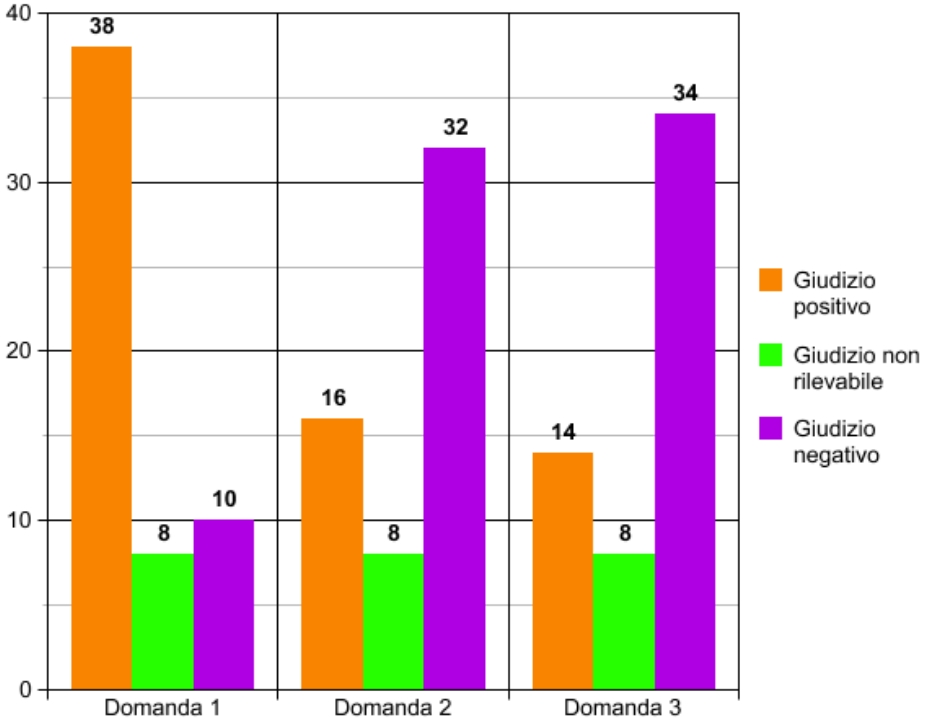

Figura 7 - Giudizi a confronto

Quindi la percentuale dei giudizi risulta la seguente:

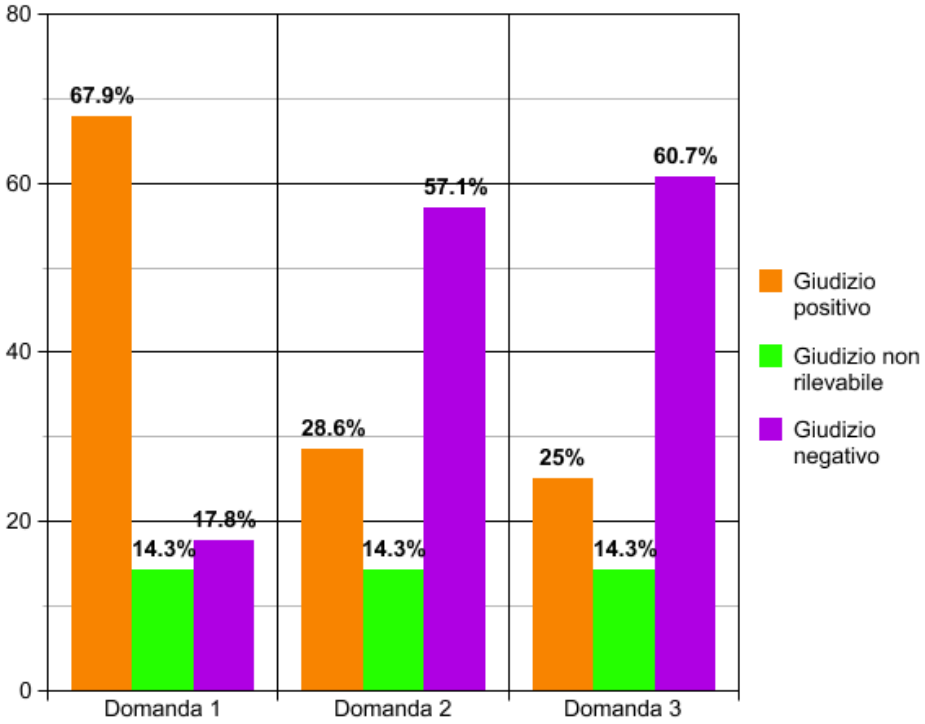

Figura 8 - Giudizi a confronto in percentuale

2.4 Conclusioni successive alle statistiche

In base al risultato ottenuto di fronte alla prima domanda, si evince che il giudizio è sicuramente positivo e che quindi vale la pena investire in un progetto di questo tipo. Nonostante il risultato negativo alla seconda domanda, non si può di certo negare che "Facebook" sia diventato negli ultimi anni uno strumento di divulgazione di massa enormemente utilizzato. Tuttavia questo risultato c'era, in un certo senso, da aspettarselo. Infatti esso può dipendere sia dall'età dei titolari oppure proprio anche dalla domanda stessa. Essa può essere stata posta male e non essere stata totalmente chiara e quindi aver portato a una risposta negativa. Per quanto riguarda la terza domanda, il risultato non è ampiamente negativo come nella precedente, tuttavia esso non è ancora sufficiente ad investire nello sviluppo di questa funzionalità.

Quindi, a fronte dei risultati ottenuti, abbiamo deciso di non realizzare la parte relativa a "Facebook" e alla "Fidelizzazione", bensì di concentrarci principalmente sul primo aspetto.

2.5 Specifiche in linguaggio naturale

Viene di seguito trascritta la descrizione in linguaggio naturale dei requisiti di cui il sistema deve disporre successivamente alla nostra indagine. Il testo qui riportato descrive il punto di partenza del sistema che dovrà essere implementato. Sarà in seguito necessario analizzare i contenuti al fine di apportare eventuali modifiche, risolvere le ambiguità presenti ed estrarre i contenuti fondamentali che saranno utili in fase di progettazione.

"L'attività è di proprietà di una persona che ci lavora e che ha, alle sue dipendenze, altri impiegati. Esistono varie tipologie di trattamenti che il cliente può prenotare online. Ogni dipendente può seguire un solo appuntamento contemporaneamente. Per ogni servizio, presente nel listino prezzi ed effettuato al cliente, viene emessa una fattura. Inoltre il titolare dell'azienda si preoccupa

dei prodotti in magazzino e si occupa di organizzare incontri con i fornitori per rifornire appunto il magazzino."

Tuttavia, in seguito a successivi incontri con i titolari dei negozi sopra citati, abbiamo constatato che molte attività possiedono già un sistema interno per la gestione dei prodotti in magazzino e per l'emissione delle fatture. Così, il nostro interesse si è definitivamente spostato nella direzione della realizzazione di un applicativo che permettesse di prendere appuntamenti online.

2.6 Specifiche in linguaggio naturale aggiornate

A seguito di ciò, le specifiche in linguaggio naturale vengono ridotte e aggiornate all'ambito di nostro interesse e da noi trattato.

"L'attività è di proprietà di una persona che ci lavora e che ha, alle sue dipendenze, altri impiegati[1]. Esistono varie tipologie di trattamenti[4] che il cliente[2] può prenotare online. Il cliente[2] visualizza il calendario degli appuntamenti[3] e decide in quale orario inserire il proprio. Ogni impiegato[1] può seguire un solo appuntamento contemporaneamente. Il titolare, può controllare l'andamento mensile del negozio e il rendimento dei propri dipendenti. Ogni impiegato può controllare l' ultimo trattamento svolto e il futuro che svolgerà. "

2.7 Risoluzione dei conflitti semantici

Si procederà ora con l'analisi del testo sopra riportato, in modo da risolvere le ambiguità trovate, identificare omonimie o sinonimie che pregiudicano l'esatta comprensione del testo e descrivere più dettagliatamente alcuni concetti. Seguiranno la riscrittura completa delle specifiche con le nuove indicazioni e l'estrapolazione dei concetti cardine (che costituiranno la base per la stesura dello schema Entity/Relationship).

Si procederà ora con l'identificazione dei termini utilizzati nel testo che risultano troppo generici e non descrivono adeguatamente il concetto al quale fanno riferimento.

Tabella 2 - Risoluzione conflitti semantici

Numero di nota	Termine	Sostituzione	Motivazione	Significato
1	Impiegato	Dipendente	Termine ambiguo	Colui che lavora nel negozio e non è il titolare.
2	Cliente	Persona: mentre semplicemente visualizza. Utente: dopo la registrazione e login.	Termine ambiguo	Colui che visualizza inizialmente il sito, non viene considerato un cliente fino al momento in cui può prendere appuntamento. Si rende necessario il sistema di login, quindi distinzione tra persona e utente.
3	Appuntamento	Evento	Termine più specifico	Mette in risalto le caratteristiche temporali di inizio e fine.

4	Tipologia di appuntamento	Tipologia di evento	Termine più specifico	Mette in risalto la caratteristica temporale di durata.

Riportiamo di seguito il testo delle specifiche apportando le opportune modifiche. Si noti come il nuovo testo risulti più strutturato e chiarisca tutti gli elementi di ambiguità presenti, risultando così più leggibile dal punto di vista del progettista.

"Ogni persona può accedere al sito e visualizzarne i contenuti come la pagina iniziale, il listino prezzi e le informazioni per contattare l'attività. Ogni persona, la quale intende riservarsi un evento, deve essere registrato prima di poter svolgere questa azione. Durante la registrazione vengono chiesti dati personali come nome, cognome, email, username e password e avrà il ruolo "regular". Successivamente è possibile eseguire il login con le proprie credenziali. Una volta eseguito il login, l'utente può riservarsi un evento, scegliendo tra le varie tipologie di evento messe a disposizione. È possibile successivamente, modificare data e ora. Inoltre può visualizzare lo storico dei propri precedenti eventi. Il titolare è anch'esso un utente con il ruolo di "admin" e può controllare lo stato economico dell'attività e il rendimento dei propri dipendenti. I dipendenti, anch'essi utenti, avranno il ruolo di "worker" e potranno consultare l' ultimo evento svolto e il prossimo futuro. Una volta terminate tutte le attività, l'utente eseguirà il logout."

2.8 Individuazione dei concetti fondamentali

Utilizzando il testo riveduto e corretto, verranno individuati i concetti chiave in base ai quali verrà realizzato lo schema scheletro, il quale verrà analizzato e raffinato nelle fasi successive fino a ottenere lo schema definitivo.

Grazie al lavoro di analisi svolto fino a questo punto è possibile individuare gli elementi (e relativi attributi) basilari su cui formare lo schema scheletro.

Appare chiaro che gli elementi chiave sono rappresentati dagli eventi (appuntamenti), dagli utenti e dai lavoratori.

2.9 Schema scheletro

L'analisi dei requisiti ha portato all'individuazione di tre entità fondamentali: "Events", "Users", "Workers" (per esigenze di implementazione, come verrà spiegato nel capitolo "Tecnologie utilizzate", utilizziamo i nomi in inglese).

Le entità e le associazioni che intercorrono tra esse sono rappresentate nel seguente schema scheletro.

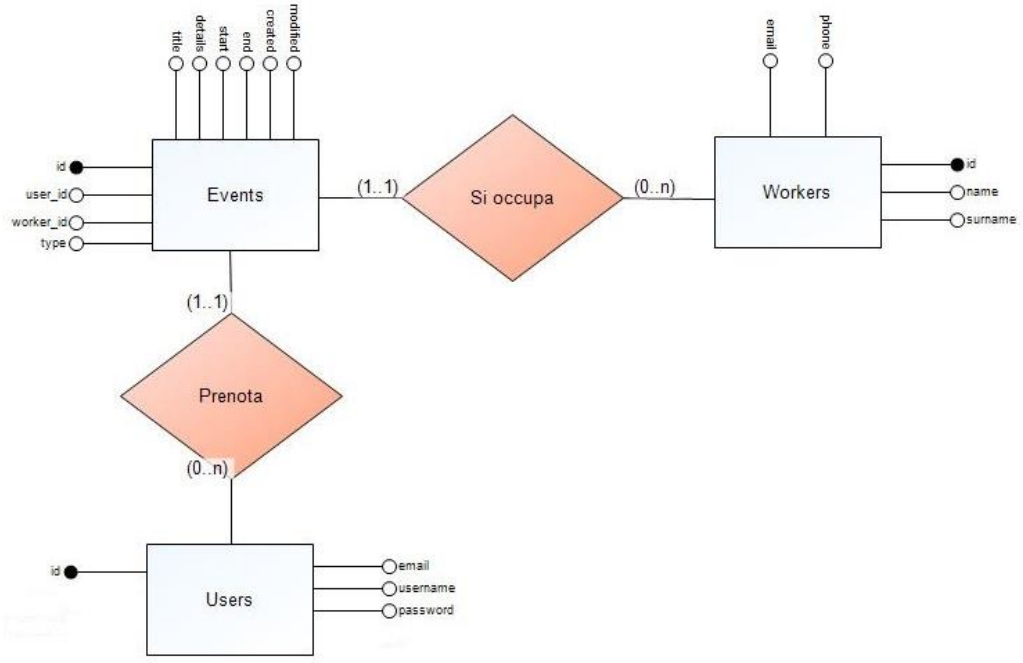

Figura 9 - Schema scheletro

Nella sezione seguente verrà approfondita l'analisi delle entità principali , precisando con maggior dettaglio il ruolo di ciascuna. Tale raffinamento permetterà di giungere alla definizione dello schema concettuale finale.

3
PROGETTAZIONE

La progettazione **[2]** è stata divisa in due parti. La prima di progettazione concettuale e la seconda di logica.

3.1 Progettazione concettuale

Verrà ora sviluppato lo schema Entity/Relationship, assumendo come base lo schema scheletro proposto alla fine della sezione precedente.

3.1.1 Sviluppo dell' entità "users"

Dal testo dell'intervista e dai successivi incontri con i vari titolari delle attività, si è colta la necessità, come già accennato in precedenza, di avere, al posto dell'entità generica "customers", una nuova entità "users", per mettere in risalto la distinzione tra un utilizzatore comune dell'applicazione e un altro che usufruisce della possibilità di prendere appuntamenti. Tale utente dovrà possedere le informazioni base necessarie all'autenticazione (username, password ed email), inoltre è necessario introdurre un sistema di autorizzazioni che permetta di stabilire le azioni che sono permesse a un utente che ha effettuato il login.

Viene così aggiunto l'attributo "role" con tre diverse scelte che corrispondono ai tre possibili ruoli all'interno dell'attività: "admin", "worker" e "regular". "Admin" sarà il ruolo che identificherà il proprietario, il quale possiede quindi i privilegi di amministrazione;

"worker" sarà il ruolo dei dipendenti e "regular" il ruolo associato a qualunque altro utente dell'applicazione.

Inoltre sono state aggiunti anche gli attributi "name" e "surname" da utilizzare nel sistema di registrazione per poter tener traccia dei dettagli della persona che ha preso appuntamento.

L'entità "users" risulta quindi la seguente:

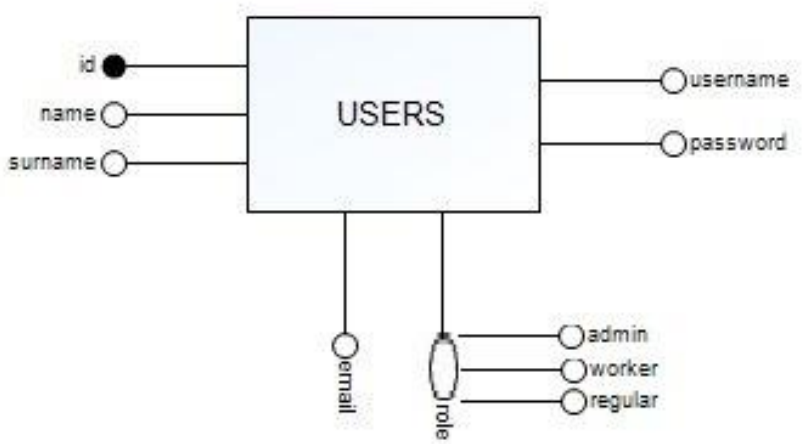

Figura 10 – Entità users

3.1.2 Sviluppo dell'entità "workers"

L'entità "workers" è stata sviluppata per tenere traccia dei dati personali dei dipendenti.

L'entità "workers" risulta quindi la seguente:

Figura 11 - Entità workers

3.1.3 Sviluppo dell'entità "events"

L'entità "events" è stata sviluppata per identificare ogni singolo appuntamento nel calendario.

L'entità "events" risulta quindi la seguente:

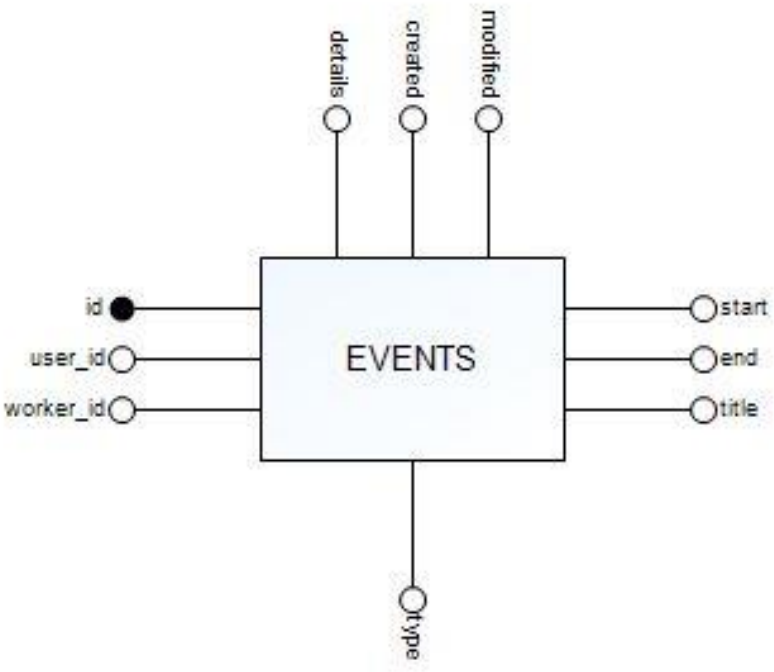

Figura 12 - Entità events

Successivamente, ci si è resi conto che utilizzare solamente un attributo per individuare la tipologia dell'evento, risultava troppo

riduttivo. Perciò si è deciso di sviluppare un'ulteriore entità, "event_types"

3.1.4 Sviluppo dell'entità "event_types"

L'entità "event_types" serve per tenere traccia di tutti i dettagli necessari a distinguere le varie tipologie o combinazioni di tipologie degli eventi.

L'entità "event_types" risulta quindi la seguente:

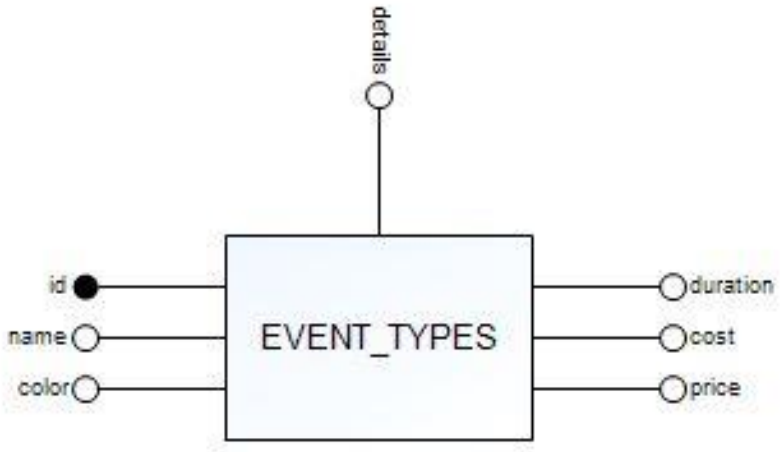

Figura 13 - Entità event_types

1. L'attributo "id" serve ad identificare unicamente ogni singola tipologia.
2. L'attributo "name" serve ad identificare le tipologie degli eventi.
3. L'attributo "color" servirà per gestire il colore degli eventi durante il drag & drop nel calendario dello scheduler.
4. L'attributo "duration" per decidere quale porzione di calendario sarà occupata da un evento di una certa tipologia.
5. Gli attributi "cost", "price" e "details" serviranno per rendere più chiaro ogni aspetto della tipologia.

3.1.5 Schema concettuale finale

Alla luce di ciò che è emerso nei paragrafi precedenti, lo schema concettuale finale risulta il seguente:

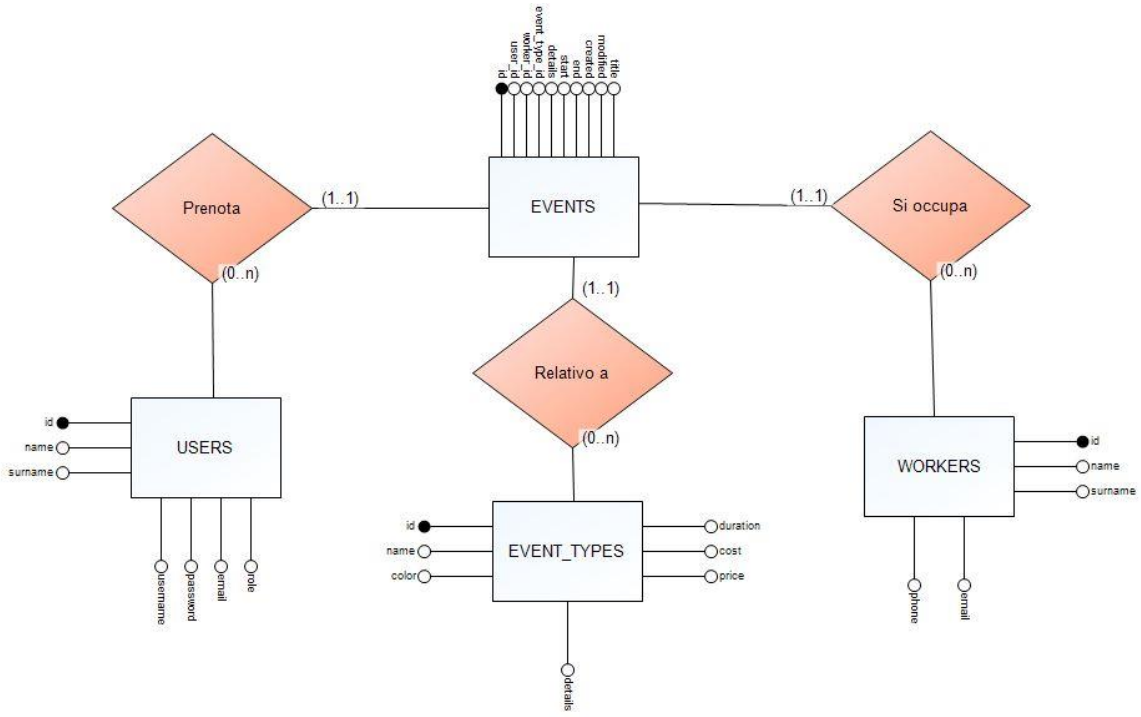

Figura 14 - Schema finale

3.2 Progettazione logica

3.2.1 Requisiti funzionali

Per completare la progettazione del sistema è necessario innanzitutto analizzare le tipologie di utenze che si avvarranno del sistema.

3.2.2 Classi di utenti

Dalla precedente progettazione concettuale si evidenziano tre diversi tipi di autorizzazione agli utenti del sistema, ognuno dei quali è indipendente dagli altri:

a. Admin
b. Worker
c. Regular

3.2.3 Creazione database

Il database utilizzato è MySQL **[3]**. Ora procederò con la creazione delle tabelle che serviranno per l'applicazione.

1. Tabella "events"

```
CREATE TABLE 'events' (
'id' INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
'event_type_id' INT(11) NOT NULL,
'worker_id' INT(11) NOT NULL,
'user_id' INT(11) NOT NULL,
'title' VARCHAR(255) NOT NULL,
'start' DATETIME NOT NULL,
'end' DATETIME NOT NULL,
'created' DATETIME NOT NULL,
'modified' DATETIME NOT NULL);
```

2. Tabella “event_types”

```
CREATE TABLE 'event_types' (
'id' INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
'name' VARCHAR(20) NOT NULL,
'color' VARCHAR(255) NOT NULL,
'duration' INT(11) NOT NULL,
'cost' DECIMAL(10,2) NOT NULL,
'price' DECIMAL(10,2) NOT NULL,
'details' TEXT);
```

3. Tabella “users”

```
CREATE TABLE 'event_types' (
'id' INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
'name' VARCHAR(255) NOT NULL,
'surname' VARCHAR(255) NOT NULL,
'email' VARCHAR(255) NOT NULL,
'username' VARCHAR(50) NOT NULL,
'password' VARCHAR(50) NOT NULL,
'role' ENUM('admin','worker','regular') DEFAULT 'regular');
```

4. Tabella “workers”

```
CREATE TABLE 'event_types' (
'id' INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
'name' VARCHAR(255) NOT NULL,
'surname' VARCHAR(255) NOT NULL,
'email' VARCHAR(255) NOT NULL,
'phone' VARCHAR(50),
'admin' TINYINT(1) NOT NULL);
```

3.2.4 Progettazione tramite console CakePHP

Come verrà approfondito nel capitolo successivo, CakePHP mette a disposizione una console che, tramite il procedimento detto “Baking” **[3]**, va a creare, grazie alle convenzioni, tutte le associazioni necessarie nel database, nonché la configurazione del database stesso.

Per velocizzare il “Baking”, ho realizzato uno schema Entity/Relationship rivisitato, basandomi su quello precedente, rispettando però la sintassi di CakePHP. In questo modo il “Baking” risulta più intuitivo e immediato durante la progettazione stessa. In questo diagramma ho messo in risalto solamente quegli attributi che verranno realmente coinvolti durante il “Baking”.

Il risultato è quindi il seguente:

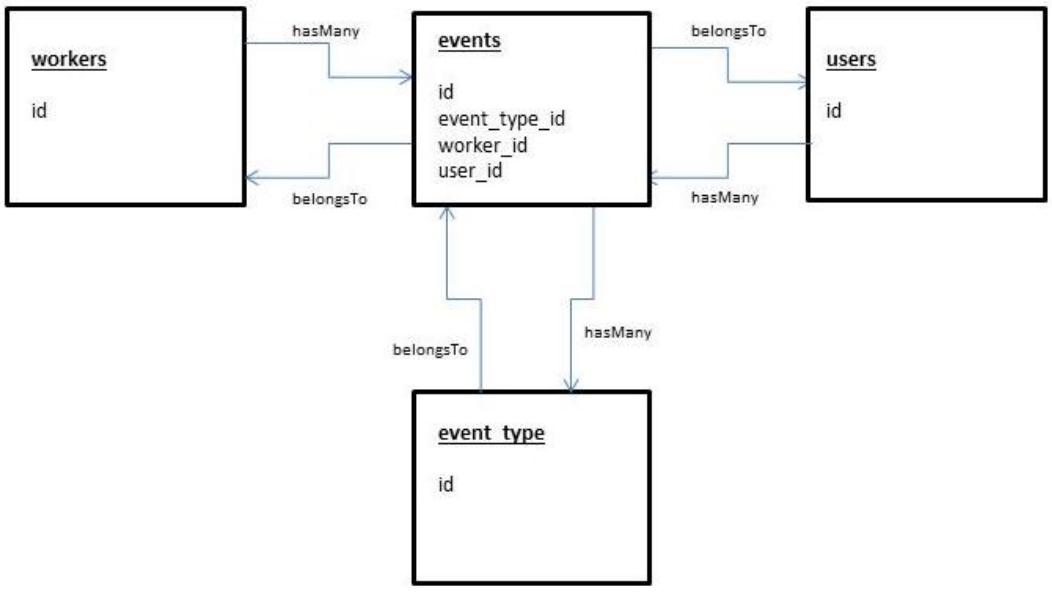

Figura 15 - Schema E/R in sintassi CakePHP

Procederò ora con il presentare le varie fasi del "Baking".

Il "Baking" è in grado di gestire la creazione di alcune parti dell'applicazione:

- Configurazione del database
- Model
- View
- Controller

Innanzitutto bisogna provvedere alla configurazione del database generando così un nuovo file *database.php* all'interno della cartella *app/config* che contiene tutte le informazioni necessarie alla gestione e al corretto funzionamento del database.

Dopo aver terminato questa fase, per prima cosa bisogna creare i vari Models. Automaticamente, grazie alle convenzioni, viene identificato il tipo di ogni associazione con gli altri Models.

Successivamente si devono configurare i Controllers relativi ai vari Models precedenti. Il "Baking" dà la possibilità di gestire la creazione dei Controllers in maniera interattiva.

In questa modalità, ad ogni passo, è possibile modificare le opzioni di cui si necessita per modellare i vari aspetti del Controller.

Infine, creando le Views, si hanno a disposizione le operazioni CRUD indispensabili per la nostra applicazione.

3.2.5 Query SQL

Le query **[4]** di interrogazione del database verranno presentate nel capitolo relativo all'implementazione, in quanto imprescindibili dalla sintassi di CakePHP.

4
TECNOLOGIE UTILIZZATE

4.1 Premessa

In questa sezione verranno descritte in dettaglio le tecnologie utilizzate per la realizzazione dell'applicazione. Ho scelto di riservare un capitolo a parte a queste tecnologie, precedente alla presentazione del sistema realizzato, in quanto lo studio di esse, essendo tecnologie di nuova generazione, abbia occupato una parte rilevante del mio lavoro.

4.1.1 Web usability e User Experience

Questo paragrafo sposta l'accento su quegli argomenti che hanno importanti implicazioni strategiche quando si vuole fare business online. In tal caso si parla di usabilità (o funzionalità) dei siti e verranno presentati alcuni accorgimenti necessari per mettere al centro della nostra web strategy il cliente.

Gli utenti hanno la possibilità di scegliere tra un numero enorme di siti web da visitare ed è quindi logico pensare che difficilmente perderanno tempo con siti confusi, lenti, non rispondenti alle loro esigenze.

L'enorme possibilità di scelta ha fatto sì che il pubblico abbia sviluppato una certa impazienza e pretende spesso la gratificazione

immediata: se non riesce a capire come usare il sito entro un minuto, è facile che se ne vada.

Il navigatore inoltre, diventando sempre più esperto, è in grado di valutare la qualità dei siti molto velocemente ed egli pretende sempre nuove soluzioni innovative per quanto riguarda la grafica e i metodi di interazioni col sito stesso. Un sito di scarsa qualità, oltre ad attrarre poca attenzione, può provocare dei danni all'immagine del titolare dell'attività. È quindi molto importante, quando si decide di intraprendere un'iniziativa sul web, evitare di farlo con approssimazione: si rischia altrimenti non solo il fallimento dell'iniziativa in sé, ma anche di arrecare danno all'immagine aziendale.

Negli ultimi anni il design è diventato molto importante. Infatti, sul web la componente artistica e creativa ha grande importanza. L'obiettivo principale dei progetti in rete dovrebbe essere rendere più semplice ed usabile possibile il nostro sito. Un elemento di grande aiuto nell'elaborazione progettuale è l'osservazione dei comportamenti degli utenti, scoprire quello che a loro piace e apprezzano, capire quali difficoltà incontrano e quali invece sono le soluzioni che trovano più facili da usare. È proprio a seguito di queste problematiche che a metà degli anni '90, Donald Norman introdusse il termine User Experience.

Con User Experience (o esperienza d'uso) si intende ciò che una persona prova quando utilizza un prodotto, un sistema o un servizio. La User Experience concerne gli aspetti esperienziali, affettivi, l'attribuzione di senso e di valore collegati al possesso di un prodotto e all'interazione con esso, ma include anche le percezioni personali quali l'utilità, la semplicità d'utilizzo e l'efficienza del sistema. Essa ha una natura soggettiva perché riguarda i pensieri e le sensazioni di un individuo nei confronti di un sistema; inoltre è dinamica, dal momento che si modifica nel tempo al variare delle circostanze.

Dunque, una difficoltà nell'implementazione di pagine web funzionali è costituita dalla necessità di adeguare in continuazione le proprie conoscenze in funzione della rapidissima evoluzione tecnologica e alle continue e mutevoli richieste da parte degli utenti.

4.1.2 Framework: perché usarli?

Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework c'è sempre una serie di librerie di codici utilizzabili con uno o più linguaggi di programmazione.

Lo scopo di un framework è quello di risparmiare allo sviluppatore la riscrittura di codice già steso in precedenza per compiti simili. Questa circostanza si è presentata sempre più spesso man mano che le interfacce utente sono diventate sempre più complesse, o più in generale man mano che è aumentata la qualità di software con funzionalità secondarie simili.

Il termine inglese *framework* quindi può essere tradotto come *intelaiatura* o *struttura*, che è appunto la sua funzione, a sottolineare che al programmatore rimane solo da creare il contenuto vero e proprio dell'applicazione.

Passiamo ora ad analizzare brevemente i vantaggi derivanti dall'utilizzo di framework:

- Rapidità di sviluppo: utilizzando framework o librerie già pronte, si accorciano notevolmente i tempi di sviluppo (cosa che rende intrinsecamente più competitivi, potendo offrire al cliente dei tempi di sviluppo minori e conseguentemente minori costi di sviluppo);
- Riusabilità: un proprio frammento di codice sviluppato utilizzando un framework standard può essere facilmente riutilizzato in un altro progetto, senza alcun o con pochissime righe di codice;
- Evitare di perdere tempo "reinventando la ruota": non si perde tempo a risolvere per ogni progetto una problematica già affrontata e risolta da tempo, magari da decine di sviluppatori, arrivando ad una soluzione largamente accettata e condivisa, che ormai può essere data quasi per "scontata";
- Aderenza agli standard: l'uso di framework rende facilmente condivisibile e manutenibile anche da terze persone il proprio codice sorgente;

- Accrescimento personale: imparare a sviluppare con framework diffusi comporta implicitamente un accrescimento personale ed una "maggiore rivendibilità" dello sviluppatore sul mercato del lavoro;
- Concentrarsi sul reale obiettivo del progetto: non dovendo investire parte del proprio tempo a risolvere problemi ancillari, si possono massimizzare i propri sforzi sull'analisi delle problematiche per la quale l'applicazione è in fase di sviluppo.

4.2 Progettare un'applicazione web con WAMP

Per lo sviluppo dell'applicazione web ho utilizzato la piattaforma WAMP **[5]** che è considerata la combinazione più diffusa e collaudata negli ambienti di produzione e sviluppo.

WAMP è un acronimo con cui si indica una piattaforma software di sviluppo web/database che prende il nome dalle iniziali dei componenti software con cui è realizzata:

- Windows: il sistema operativo che deve essere già installato sul PC;
- Apache: il Web server;
- MySQL: il database server con SQLite e relativi tool grafici **[6]**;
- PHP: il linguaggio di scripting **[7][8]**.

4.3 Il framework CakePHP

4.3.1 Introduzione

CakePHP **[9]** è un framework per lo sviluppo veloce di applicazioni PHP, gratuito e open-source **[10]**. È una struttura su cui i programmatori possono creare applicazioni web. CakePHP elimina la monotonia dallo sviluppo web e mette a disposizione tutti gli strumenti di cui si ha bisogno per iniziare a programmare. Invece di

"reinventare la ruota" ogni volta che si inizia un nuovo progetto, si scarica una copia di CakePHP e si parte direttamente dal core dell'applicazione.

4.3.2 Il pattern MVC

CakePHP utilizza la tecnologia MVC (Model, View, Controller) o anche design pattern software MVC.

Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il Model che rappresenta il gestore dei dati dell'applicazione;
- il Controller che gestisce ed indirizza le richieste fatte dal client;
- le View che forniscono la presentazione dei dati del modello al client.

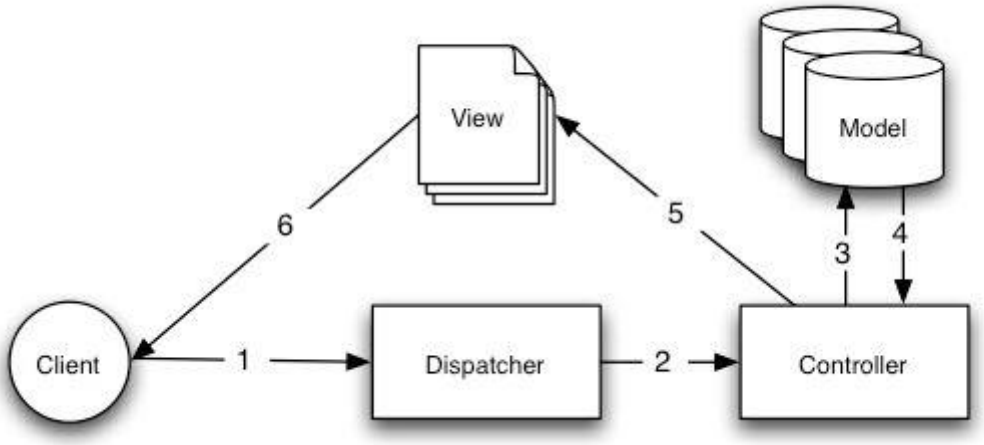

Figura 16 - Tipica richiesta MVC di CakePHP

Il tipico ciclo di richiesta di CakePHP comincia con la richiesta da parte dell'utente di una pagina o di una risorsa dell'applicazione. Questa richiesta è prima di tutto analizzata da un dispatcher, il quale selezionerà il corretto controller che dovrà gestirla.

Una volta che la richiesta arriva al controller, questo comunicherà con il model per elaborare i dati che potrebbero essere necessari.

Dopo questa comunicazione, il controller delega alla corretta view il compito di generare un output con i dati forniti dal model.

Infine, appena l'output viene generato, questo viene immediatamente restituito all'utente.

Il design pattern MVC è un modello testato e trasforma un'applicazione in un pacchetto modulare e rapidamente sviluppabile. Dividere i compiti dell'applicazione in model, view e controller separati rende l'applicazione leggera e versatile, poiché nuove caratteristiche potranno essere facilmente aggiunte, in tal modo, sarà possibile costruire il nuovo sulla base del vecchio.

4.3.3 La struttura di CakePHP

CakePHP possiede oltre le classi per Controller, Model e View, anche classi aggiuntive ed oggetti che rendono lo sviluppo MVC veloce ed agevole. Components, Behavior ed Helper sono classi che forniscono estendibilità e riusabilità per aggiungere velocemente funzionalità alle altre classi MVC di base dell'applicazione.

4.3.3.1 Struttura dei file

Essendo ora a conoscenza di come CakePHP svolga il proprio lavoro, siamo in grado di analizzare come i suoi file sono organizzati.

Appena installato, la directory avrà il seguente contenuto:

- app
- cake
- docs
- vendors

Nella directory "app" saranno svolte la maggior parte delle "attività": in essa verranno salvati i file dell'applicazione. La cartella "cake" contiene il core del framework. "Docs" è per i file di tipo readme, changelog ed informazioni di licenza d'uso. In ultimo,

nella directory “vendors” è possibile collocare librerie di terze parti utilizzabili all’interno dell’applicazione CakePHP.

4.3.3.2 La directory App

La directory app è il posto dove si svolge la maggior parte del lavoro di sviluppo e contiene le cartelle elencate qui di seguito nella tabella 1.

Tabella 3 - Directory “app” in CakePHP

Nome	Descrizione
config	Contiene i pochi file di configurazione che CakePHP utilizza. I dettagli di connessione col database, bootstrapping, file di configurazione del core e altro dovrebbero essere salvati qui.
Controllers	Contiene i controllers e components.
Locale	Qui vanno posti i file per le stringhe di internazionalizzazione.
Models	Contiene model, behavior e datasource per la web application.
Plugins	Contiene i pacchetti plugin utilizzati.
tmp	Questo è il posto dove CakePHP salva i file temporanei. I dati che vengono salvati dipendono da come è stato configurato CakePHP, ma questa directory è di solito usata per salvare le descrizioni dei models, i file di log e, talvolta, anche le informazioni di sessione.
Vendor	Ogni classe o libreria di terze parti dovrebbe essere posizionata qua. Facendo questo sarà facile accedervi utilizzando la funzione vendor().
Views	I file di presentazione vanno salvati qui: file per element, pagine di errore, helper, layout e view.
webroot	In un setup di produzione, questa cartella dovrebbe servire come root della web application. Sottocartelle poste qui possono servire anche a contenere CSS stylesheets, immagini e file Javascript.

4.3.3.3 Le convenzioni in CakePHP

Nel mentre si perde un po' di tempo ad imparare le convenzioni in uso su CakePHP, nel lungo periodo si risparmierà molto: seguendo le convenzioni, infatti, si ottiene gratuitamente l'accesso a molte funzionalità e ci si sarà liberati dalla necessità di mantenere un pacchetto di file di configurazione. L'utilizzo delle convenzione, inoltre, rende lo sviluppo molto uniforme, permettendo ad altri sviluppatori di unirsi e di collaborare molto più facilmente.

Ora elencherò ed introdurrò le principali convenzioni utili al corretto funzionamento di un'applicazione in CakePHP.

a. In generale, i nomi dei file sono in minuscolo, mentre i nomi delle classe sono del tipo CamelCased.
b. I nomi per le classi dei modelli sono singolari.
c. I nomi delle tabelle del database corrispondenti ai modelli CakePHP sono plurali e minuscoli.
d. Foreign keys nelle relazioni hasMany (0-n oppure 1-n), belongsTo o hasOne (1-1) sono risconosciute di default come il nome singolare del modello associato seguite da _id.
e. Il join tra tabelle, utilizzate nella relazione hasAndBelongsToMany (0-n oppure 1-n associata ad 0-n oppure 1-n), dovrebbe essere denominato in accordo ai nomi delle tabelle che si vanno ad unire e quindi rispettando l'ordine alfabetico.
f. Tutte le tabelle con le quali CakePHP interagisce richiedono una singola chiave primaria per identificare ogni riga. Se si desidera utilizzare una tabella che non ha un singolo campo come chiave primaria, la convenzione di CakePHP è che un singolo campo di chiave primaria è aggiunto alla tabella.
g. CakePHP non supporta chiavi primarie complesse.
h. I nomi per le classi dei controller sono plurali, CamelCased e finiscono in Controller.
i. La prima funzione che dovrebbe essere scritta per un controller è la funzione index(). Quando una richiesta specifica un controller, ma non un'azione, il comportamento di default di CakePHP è quello di renderizzare la funzione index() del controller stesso.

j. I file per le view vengono denominate in accordo alle funzioni dei controller.

4.3.4 Sviluppare con CakePHP

I requisiti di sistema necessari all'installazione di CakePHP sono:

- Un Server http: è suggerito (ma non indispensabile) l'utilizzo di Apache con mod_rewrite abilitato;
- PHP 4.3.2 o versioni superiori;
- Teoricamente non è richiesto un database, ma se fosse necessario CakePHP ne supporta parecchi tra cui MySQL (4 o superiore), Oracle e SQLite.

Tutti questi requisiti ci sono stati forniti direttamente dal pacchetto Wamp.

Successivamente la preparazione all'installazione consiste nei seguenti passi:

1. Si scarica una copia di CakePHP.
2. Si configura il webserver con php.
3. Si controllano i permessi dei file.

4.3.5 Controllers

Il controller è utilizzato per gestire la logica dell'applicazione. Normalmente un controller gestisce la logica relativa a un model. In CakePHP i controllers prendono il nome al plurale dei relativi models.

I controllers sono classi che estendono la classe CakePHP AppController, che a sua volta estende la classe Controller. La classe AppController può contenere metodi condivisi da tutti i controllers dell' applicazione. La classe AppController estende, come detto, la classe Controller, che è una classe standard di CakePHP.

I controllers includono tutti i metodi che normalmente chiameremo azioni. Le azioni sono dunque metodi del controller finalizzati a

renderizzare le views. Un'azione rappresenta un solo metodo del controller. CakePHP richiama le azioni di un determinato controller in base alle richieste provenienti dall'URL.

Esploriamo ora alcuni attributi e metodi forniti da CakePHP. La classe AppController è la classe genitrice di tutti i controller dell'applicazione. AppController estende la classe Controller inclusa nelle librerie base di CakePHP. AppController viene definita come segue:

```
<?php
class AppController extends Controller
{
}
?>
```

Attributi e metodi del Controller creati nell'AppController saranno disponibili a tutti i controller dell'applicazione. Questo è il posto ideale per creare codice che è comune a tutti i controller.

Molto importanti sono gli attributi che definiscono helpers, components e models che devono essere usati dal controller. Usando questi attributi, queste classi MVC saranno accessibili dal controller come variabili ($this->ModelName, per esempio). In particolare i controllers hanno accesso alla classe del model principale a cui sono associati. Gli Helper Html e Session sono disponibili di default.
I metodi principali del controller sono i seguenti:

set(string $var, mixed $value)

Il metodo set() è il mezzo principale per inviare dati dal controller alla vista. Una volta usato set(), la variabile potrà essere utilizzata nella vista.
Il metodo set() utilizza un array associato come primo parametro. Questo può essere spesso utilizzato come modo veloce per assegnare delle informazioni alla vista.

render(string $action, string $layout, string $file)

Il metodo render() è chiamato automaticamente alla fine di ogni azione richiesta al controller. Questo metodo esegue tutte le logiche della vista (usando i dati utilizzati nel metodo set()), inserisce le viste nel proprio layout e le restituisce all'utente finale. La vista di default usata dal render è determinata dalla convenzione.

Il controllo di flusso può essere gestito con la seguente proprietà:

`redirect(string $url, integer $status, boolean $exit)`

Il metodo di controllo del flusso usato maggiormente è appunto redirect(). Questo metodo prende in ingresso un array il quale specifica il controller e la relativa azione cui si deve riferire e reindirizzare.

4.3.6 Models

I models rappresentano i dati e sono usati in applicativi CakePHP per accedere ai dati medesimi. Un model attinge solitamente dalle tabelle di un database. Un model può essere collegato ad altri e possono essere associati tra loro, allo scopo di rendere l'accesso ai dati più facile.

I models CakePHP estendono la classe speciale AppModel, che risulta essere il genitore per tutti i model dell'applicazione. Come AppController, questa classe è il punto perfetto per inserire la logica condivisa da tutti i models dell'applicazione.

Una volta che si è definito un model è possibile utilizzarlo nel relativo controller.

Utilizzare i models per ritrovare e salvare i dati può far risparmiare molto tempo e codice. I models di CakePHP forniscono un percorso standard e centralizzato per memorizzare i dati, ma allo stesso tempo forniscono un livello di sicurezza per l'applicazione.

4.3.7 Views

Il layer view di CakePHP rappresenta il modo in cui si parla agli utenti.

I file delle views di CakePHP sono scritti in chiaro PHP ed hanno un'estensione di tipo .ctp (CakePHP Template). Questi file contengono tutta la logica di presentazione necessaria per prendere i dati da presentare, che sono stati ricevuti dal controller, e renderli all'utente in un formato leggibile.

I file delle views sono memorizzati in una sotto directory di /app/View/, nominata col nome del controller che la usa, seguita dalle azioni alle quali corrisponde.

Il livello delle views di CakePHP può essere composto da differenti parti:

- Layouts: i file delle views, contengono il codice di presentazione che racchiude la maggior parte delle interfacce dell'applicazione. Molte views vengono reindirizzate dentro un layout.
- Elements: piccoli, riusabili frammenti di codice di una view. Gli elements sono normalmente reindirizzati dentro le views.
- Helpers: queste classi incapsulano la logica della view che è necessaria in molte parti del livello view. Tra le altre cose, gli helpers in CakePHP possono aiutarci, ad esempio, a costruire form o funzionalità AJAX.

4.3.8 La console di CakePHP

Questo paragrafo fornisce una spiegazione della console di CakePHP e va a completare ciò che è stato anticipato precedentemente.

Una strada veloce per mettere in piedi e far partire un'applicazione solo con un database e alcune classi base, è l'utilizzo della console Bake di CakePHP. Questa console può creare ogni "ingrediente" base: models, views e controllers compresa la configurazione con il

database. Non si parla di semplici classi "scheletro", bensì di un'applicazione totalmente funzionante grazie alle operazioni CRUD (Create, Read, Update, Delete), le quali sono alla base di ogni tipo di applicazione. Per utilizzare la console di CakePHP, è necessario aprire una shell di sistema e recarsi all'interno della cartella app della nostra applicazione e digitare *"cake bake"*. A questo punto è possibile scegliere l'alternativa che si desidera fra quelle sottoelencate:

```
Interactive Bake Shell
---------------------------------------------------------------
[D]atabase Configuration
[M]odel
[V]iew
[C]ontroller
[Q]uit
What would you like to Bake? (D/M/V/C/Q)
```

4.4 JQuery

Il web 2.0 con il quale tutti noi abbiamo a che fare giornalmente è basato massicciamente sull'uso di JavaScript. Dal punto di vista degli sviluppatori, tuttavia, la natura lato client del linguaggio è da sempre la fonte principale di problemi e mal di testa. Infatti, poiché ogni browser implementa uno specifico motore JavaScript, ognuna (ma soprattutto IE) con specifiche ed eccezioni proprie, è spesso impossibile essere certi dei funzionamenti cross-browser di uno script.

In risposta a questi problemi sono nati progetti di librerie (o meglio framework) in grado di garantire il funzionamento cross-browser degli script e di estendere o comunque facilitare le funzioni native di JavaScript.

A partire dal 2006 nasce appunto jQuery **[11]**, un framework sviluppato da John Resig con il preciso intento di rendere il codice più sintetico e di limitare al minimo l'estensione degli oggetti globali per ottenere la massima compatibilità con altre librerie.

Da questo principio è nata una libreria in grado di fornire un'ampia gamma di funzionalità, che vanno dalla manipolazione degli stili CSS e degli elementi HTML, agli effetti grafici per passare a comodi

metodi per chiamate AJAX cross-browser. Il tutto, appunto, senza toccare nessuno degli oggetti nativi JavaScript.

La prima cosa fondamentale da sapere è che tutto ruota attorno all'oggetto/funzione $, a sua volta un'abbreviazione (o alias) di jQuery.

Le due caratteristiche principali di jQuery sono anzitutto la brevità del codice utilizzato, ma soprattutto il fatto che l'elemento da ricercare sia passato alla funzione $() sotto forma di selettore CSS.

I buoni motivi per cui usare jQuery sono:

- È possibile usare la libreria in tutti i progetti, senza paura di incappare in incompatibilità nel codice.
- Ha un semplice sistema di estensione che permette di aggiungere nuove funzionalità (plugin) oltre a quelle predefinite.
- Una numerosa community che mette a disposizione plugin e supporto di ottimo livello.
- Sintassi sintetica ed efficiente.

4.5 JQuery UI

JQuery UI **[12]** è una libreria Open Source di plugin basata, come si può intuire, sulla libreria JavaScript jQuery.

Il progetto jQuery UI, nonostante alcuni gravi problemi di compatibilità e stabilità delle prime versioni, ha raggiunto ormai una buona maturità ed un'architettura estendibile che lo rendono un ottimo punto di partenza per tutti gli sviluppatori che vogliono dedicarsi alla realizzazione di una moderna applicazione per il Web.

Partito come un progetto parallelo a jQuery, UI è stato ben presto inglobato nello sviluppo della libreria, pur mantenendo un gruppo di sviluppo autonomo.

Al momento jQuery UI è suddivisa in quattro macroaree di interesse:

- Interactions: drag & drop, ridimensionamento, ordinamento e selezione.
- Widgets: numerosi widgets.
- Effects: transazioni animate.
- Utilities: utilità di basso livello da usare per le interazioni, i widgets e gli effetti.

4.6 Bootstrap: il framework di Twitter

Da qualche anno si stanno sempre più affermando tra gli addetti ai lavori, i framework CSS. Si tratta di un pacchetto di file (HTML, CSS e JavaScript) che consentono di iniziare a sviluppare un front-end partendo già da una base solida, collaudata e standard.

Ebbene, il team di sviluppatori che ha realizzato Twitter ha deciso di crearsi un proprio framework per allineare e standardizzare i vari progetti interni. La cosa interessante è che nel 2011 hanno deciso di condividerlo con il resto del mondo, rilasciando come progetto open source chiamato appunto Bootstrap Twitter **[13]**.

Per iniziare, basta scaricare il pacchetto che contiene tre cartelle IMG, JS e CSS, cioè tutto l'occorrente per partire.

Bootstrap definisce un gran numero di classi CSS, per dare un aspetto gradevole a molti elementi usati molto spesso se non sempre. Non è da sottovalutare inoltre che, usando Bootstrap, saremo abbastanza sicuri che non ci saranno problemi di compatibilità multi-browser. Oltre che dal punto di vista grafico, Bootstrap ci aiuta molto anche nell'implementazione di controllo ed elementi dinamici delle pagine.

Bootstrap è molto più leggero degli altri framework, al punto che possiamo considerarlo più una libreria. Oltre alla leggerezza, Bootstrap presenta molti vantaggi che vale la pena approfondire.

- Look originale: una delle prime regole del Web è l'originalità. Anche se vogliamo apparire solo con una semplice pagina di presentazione, il nostro sito, assieme con la grafica e il logo, deve apparire unico. Con Bootstrap diventa molto più semplice,

e quindi più veloce, ottenere un aspetto originale, che risulti quasi irriconoscibile se confrontato con il tema iniziale.

- Semplicità: è sicuramente uno dei punti di forza di Bootstrap. Non richiede l'appoggio né di un linguaggio di programmazione server side, né di un database. Per usare Bootstrap all'interno delle nostre pagine è sufficiente includere la libreria JavaScript e lo stile CSS.
- Responsive Design: (o design adattivo) è una moderna metodologia che permette di realizzare pagine Web capaci di adattarsi dinamicamente a qualsiasi dispositivo. Il design adattivo è molto importante in questa fase storica, che vede gli utenti navigare sul Web da dispositivi molto diversi, che vanno dai computer di casa allo smartphone di ultima generazione, passando per vari tablets. Tutte le interfacce, i template e le funzionalità di Bootstrap sono caratterizzate da un design adattivo. Il vantaggio è enorme, visto che il design adattivo è oggigiorno praticamente obbligatorio. Sviluppare da zero una pagina adattiva non è difficile, ma richiede comunque un certo tempo e attenzione. Utilizzando Bootstrap riusciamo a creare pagine accattivanti, originali e adattive in pochi minuti.

4.7 FullCalendar

Il plugin FullCalendar **[14]** di jQuery permette di creare un calendario sullo stile di quello di Google e di poterlo includere nelle pagine web: si possono impostare eventi, durate e ogni altro tipo di informazione che è possibile associare ad un calendario.

Il plugin fornisce la possibilità di trascinare eventi tramite il drag & drop e di utilizzare AJAX per caricare al volo gli eventi senza il refresh della pagine.

Come tutti i calendari che si rispettino, FullCalendar offre tre diversi tipi di visualizzazione:

- Mese
- Settimana
- Giorno

Ognuna di queste visualizzazioni è personalizzabile a seconda delle esigenze.

Come funziona il plugin?

Il calendario è generato – dinamicamente – all'interno di un con "id=#calendar". Per quanto riguarda il codice JavaScript: dopo aver recuperato la data corrente viene istanziato il metodo *fullCalendar()* al quale vengono passati diversi parametri che ne consentono la personalizzazione. Come per molti plugin la sintassi da utilizzare è quella JSON, ovvero si deve passare una coppia chiave/valore per definire, appunto, i diversi parametri.

Come prima cosa è stato creato l'**header** del calendario, più precisamente:

- sono stati inseriti a sinistra i pulsanti per muoversi all'interno del calendario (ovvero i pulsanti *precedente*, *successivo* e *oggi*);
- al centro è inserito il titolo, che non è altro che l'identificativo della schermata che viene mostrata all'utente (se si sta visualizzando il calendario in modalità mese, ad esempio, il titolo sarà il mese visualizzato);
- a destra vengono mostrati i pulsanti che permettono di cambiare le varie modalità di visualizzazione.

Il successivo parametro passato è il parametro **editable**: questo permette di rendere gli eventi modificabili dall'utente, ovvero di poterli spostare e ridimensionare (cambiare la durata) a piacimento.
Finalmente passiamo agli eventi (**events**).
In questo semplice esempio abbiamo raggruppato diverse tipologie di eventi che l'utente può utilizzare all'interno del calendario.
Gli eventi sono identificati per mezzo del paramentro "events" che è un array di oggetti JavaScript i quali possono contenere a loro volta diverse proprietà, qui di seguito elencate:

- **id**: Stringa o intero che identifica univocamente l'evento (per ripetere un evento bisogna utilizzare lo stesso valore);

- **title**: Testo che compare all'interno dell'evento sul calendario;
- **allDay**: true o false. Permette di determinare se un evento dura tutto il giorno oppure no (di default è true);
- **start**: Oggetto Date di JavaScript utilizzato per indicare l'inizio di un evento. I formati supportati sono IETF ("Wed, 18 Oct 2009 14:00:00 EST"), ISO8601 ("2009-11-05T13:15:30Z") o il formato UNIX timestamp;
- **end**: stesso che per start, ma identifica la fine dell'evento;
- **url**: stringa che identifica l'URL dell'evento e che sarà attivato al click su di esso;
- **className**: stringa o array di classi CSS da applicare all'evento;
- **editable**: true o false. Rende l'evento modificabile o meno. Sovrascrive il parametro "editable" visto sopra;
- **source**: sorgente dell'evento. Può essere un array, una stringa o una funzione. Questo parametro è riempito automaticamente dal plugin, quindi non deve essere settato a mano.

Le uniche proprietà obbligatorie sono *title* e *start*.

5
IMPLEMENTAZIONE

L'implementazione si è articolata in diverse fasi **[15]**, per giungere poi alla realizzazione di un'applicazione ben strutturata, facilmente "navigabile" e moderna sotto il profilo stilistico.

5.1 Models

I models rappresentano le tabelle del database nell'applicazione e sono stati trattati per primi, per i quali sono stati esplicitati le relazioni con gli altri models, in modo da utilizzarli in maniera corretta durante l'implementazione.

5.2 Default

Prima di tutto ho stabilito quali metodi e quali impostazioni dovessero essere condivise da tutti i Controller della mia applicazione. Così ho modificato il file AppController.
Per prima cosa ho specificato i metodi che mi hanno permesso di gestire le autorizzazioni del sistema.
Il metodo "*isAuthorized*" controlla che l'utente sia autenticato e viene richiamato ogni qualvolta ne abbiamo bisogno. Mentre il metodo "*beforeFilter*" identifica quali azioni è possibile svolgere prima di aver completato l'autenticazione.

Tutti i file del progetto condividono una porzione di codice comune, nel quale sono caricati tutti i frammenti, gli script e fogli di stile **[16][17]** necessari in tutte la pagine.

5.2 Sistema di registrazione e login

Il sistema di registrazione permette a qualsiasi persona di registrarsi e poter così usufruire, previa autenticazione, dei privilegi riservati agli utenti loggati. Ogni qualvolta si tenta di eseguire un'azione non permessa senza autenticazione, l'utente viene reindirizzato alla seguente pagina:

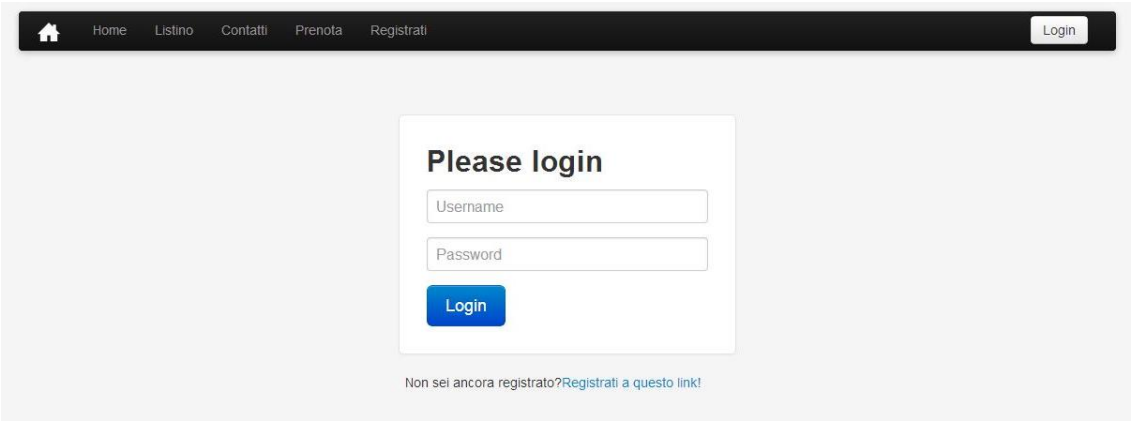

Figure 17 - Login

Tuttavia, se la persona non è ancora registrata, può farlo cliccando il link sottostante e inserendo i dati richiesti nei seguenti modi.

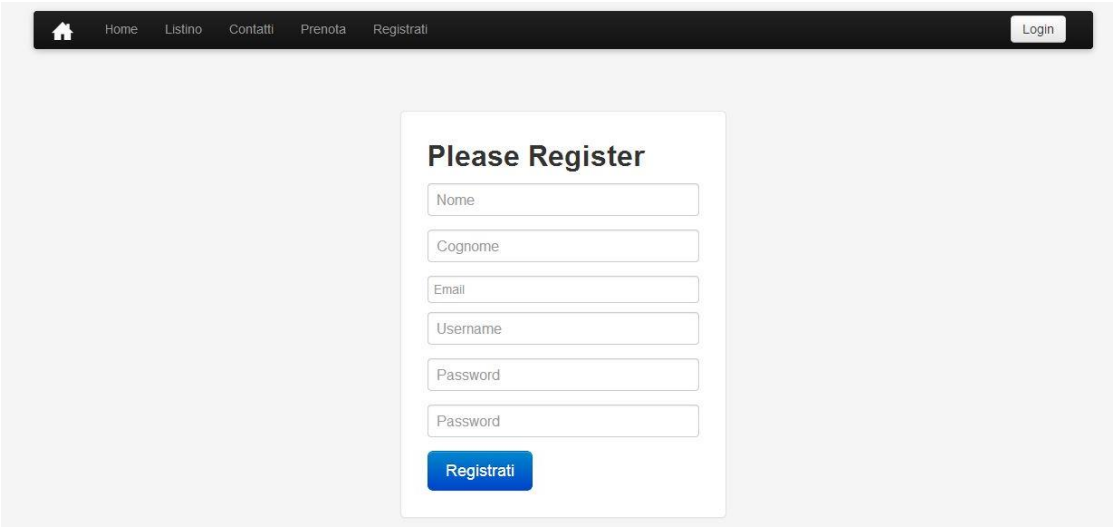

Figure 18 - Registrazione

5.3 Navbar

Come si può immaginare, la navbar è condivisa da tutte le pagine del sito. Infatti essa è stata implementata come un "*element*", cioè un frammento di codice riutilizzabile da tutte le views. I link contenuti nella navbar sono divisi in una parte, i quali sono statici,

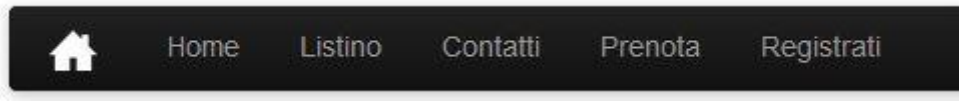

Figure 19 - Navbar - parte statica

mentre nell'altra sono caricati dinamicamente a seconda del tipo di utente loggato.

Figure 20 - Navbar - parte dinamica

La navbar risulta quindi la seguente:

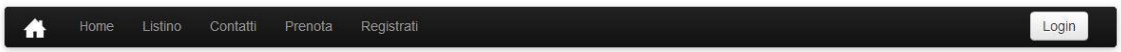

Figure 21 - Navbar

La sintassi utilizzata per implementare la navbar rispetta anch'essa la sintassi di Bootstrap.

Analizziamo ora gli aspetti dinamici della navbar, poi successivamente ci occuperemo di quelli statici.

Come già specificato durante la progettazione, esistono tre classi di utenti.

1. Caso 1: si autentica l'admin:

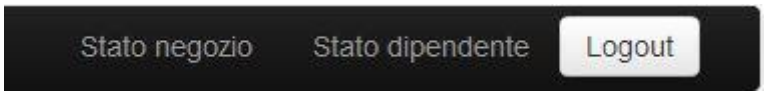

Figure 22 - Navabr "admin"

I link caricati in maniera dinamica in questo caso risultano i seguenti:

a. Il link "stato negozio" è esclusivo dell'admin e offre una panoramica sullo stato economico dell'attività e il rendimento dei dipendenti.
b. Il link "stato dipendente" è condiviso anche dai dipendenti e visualizza il trattamento di cui si è appena finito di occupare e il prossimo da effettuare.
c. Il link "logout" serve per interrompere la sessione.

a. Stato negozio

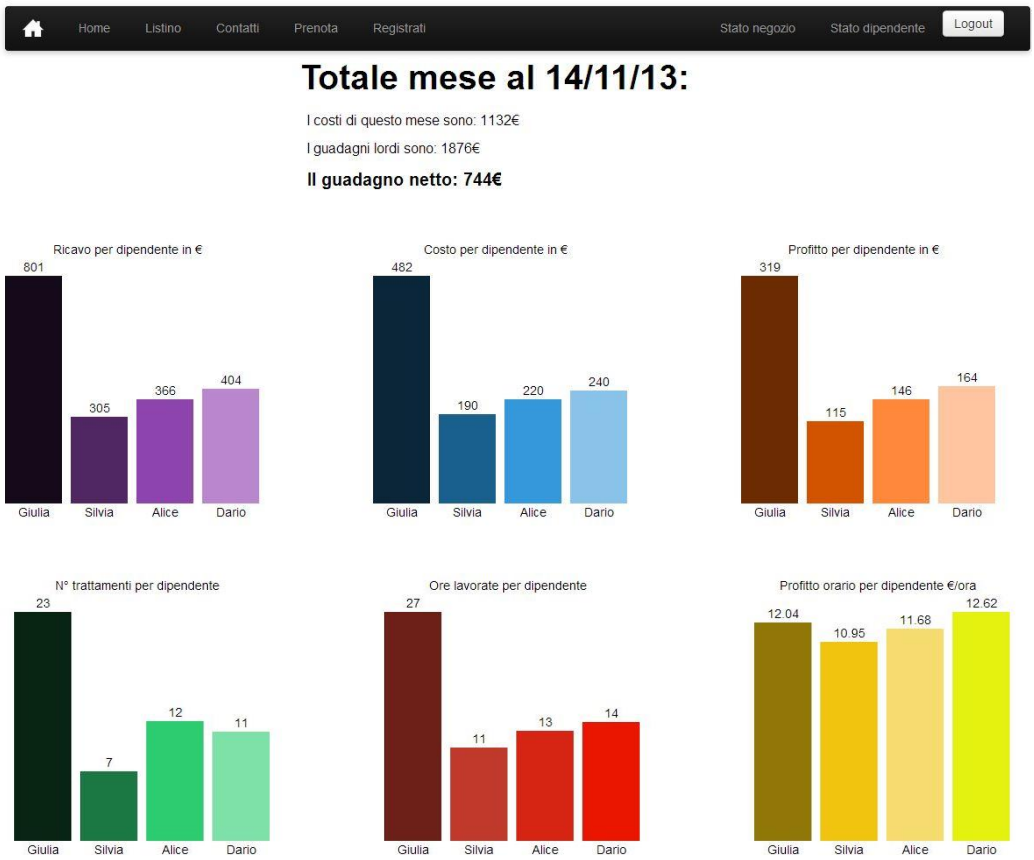

Figure 23 - Stato negozio

Per quanto riguarda i grafici, sono stati realizzati grazie al framework jQuery jqBar **[18]**. Essi sono caricati dinamicamente tramite chiamate AJAX e array JSON **[19]**. Il database viene interrogato dal WorkersController tramite apposite query e i dati restituiti vengono passati alle funzioni necessarie alla costruzione dei grafici.
Ora invece presento le funzioni necessarie al reperimento dei dati dal database (query) e necessarie a popolare i grafici e non solo:

1. Stato negozio

Viene visualizzato un riepilogo dello stato economico dell'attività.
L'idea per recuperare i dati è quella di recuperare gli eventi del mese corrente fino al giorno corrente; e per ogni evento riscontrato ricavare prezzo, costo e guadagno.

```
public function stato_negozio(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('EventType');
 $this->loadModel('Event');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
 $mese_corrente = date('n');
 $giorno_corrente = date('d');
 $costo =0;
 $prezzo=0;
```

Eseguo la query:

```
$events_mese= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente)));
foreach($events_mese as $event) {
 $costi = $this->EventType->
find('all',array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>
array('EventType.prezzo',
'EventType.costo')));
```

SQL STANDARD

```
SELECT EVENTTYPE.ID, EVENTTYPE.PREZZO,
EVENTTYPE.COSTO
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE MONTH(EVENT.START) =
        MONTH(CURDATE())
        AND DAY(EVENT.START) <
        DAY(CURDATE()) )
```

Calcolo i valori che servono:

```
    foreach($costi as $price){
        $costo=$costo+ $price['EventType']['costo'] ;
        $prezzo=$prezzo+ $price['EventType']['prezzo'] ;
        $guadagno=($prezzo-$costo);
    }
 }
```

Passo i valori calcolati in modo da poterli riutilizzare nella relativa view:

```
$this->set('costo',$costo);
$this->set('prezzo',$prezzo);
$this->set('guadagno',$guadagno);
}
```

2. Grafico 1

Nel grafico 1 viene specificato il ricavo in euro di ogni dipendente nel mese corrente. Il procedimento è lo stesso della query precedente, solo che questa volta ci si limita solo al ricavo, distinto per ogni dipendente.

```
public function ricavo_dipendenti(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('Event');
 $this->loadModel('EventType');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
 $mese_corrente = date('n');
 $giorno_corrente = date('d');
 $array = array();
```

Eseguo la query:

```
 $dipendenti = $this->Worker->
find('all');
 foreach($dipendenti as $dipendente){
   $ricavo = 0;
   $events= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id']),
'fields'=>
array('Event.event_type_id',
'Event.worker_id')));
   foreach($events as $event) {
     $prezzo = $this->EventType->
find('all'
,array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>
array('EventType.prezzo')));
     foreach($prezzo as $price){
         $ricavo =
 $ricavo+$price['EventType']['prezzo'];
     }
   }
 }
}
```

SQL STANDARD

```
SELECT EVENTTYPE.ID,
EVENTTYPE.PREZZO
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE EVENT.WORKER_ID IN
                (SELECT WORKER.ID
                FROM WORKER))
```

3. Grafico 2

Nel grafico 2 vengono evidenziati i costi dei trattamenti svolti da ogni singolo dipendente nel mese corrente. Si procede col reperire tutti gli eventi del mese corrente per ciascun dipendente, in modo da recuperarne successivamente il costo di ciascuno.

```
public function costo_dipendenti(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('Event');
 $this->loadModel('EventType');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
 $mese_corrente = date('n');
 $giorno_corrente = date('d');
 $array = array();
```

Eseguo la query:

```
$dipendenti = $this->Worker->
find('all');
foreach($dipendenti as $dipendente){
   $costo = 0;
   $events= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id']),
'fields'=>
array('Event.event_type_id',
'Event.worker_id')));
   foreach($events as $event) {
     $prezzo = $this->EventType->
find('all',
array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>
array('EventType.costo')));
     foreach($prezzo as $price){
       $costo=
$costo+$price['EventType']['costo'];
```

SQL STANDARD

```
SELECT EVENTTYPE.ID,
EVENTTYPE.COSTO
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE EVENT.WORKER_ID IN
                (SELECT WORKER.ID
                FROM WORKER))
```

4. Grafico 3

Nel grafico 3 vengono evidenziati i profitti dei trattamenti svolti da ogni singolo dipendente nel mese corrente. Si procede come nella query precedente, ma vengono recuperati sia il ricavo che il costo, in modo da farne poi la differenza, per poter calcolare il profitto di ogni dipendente.

```
public function profitto_dipendenti(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('Event');
 $this->loadModel('EventType');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
 $mese_corrente = date('n');
 $giorno_corrente = date('d');
 $array = array();
```

Eseguo la query:

```
 $dipendenti = $this->Worker->
find('all');
 foreach($dipendenti as $dipendente){
   $guadagno = 0;
   $events= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id']),
'fields'=>
array('Event.event_type_id',
'Event.worker_id')));
   foreach($events as $event) {
     $prezzo = $this->EventType->
find('all',
array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>array('EventType.prezzo',
'EventType.costo')));
     foreach($prezzo as $price){
       $guadagno =
 $guadagno+$price['EventType']['prezzo']-
$price['EventType']['costo'];
```

SQL STANDARD

```
SELECT EVENTTYPE.ID,
EVENTTYPE.COSTO,
EVENTTYPE.PREZZO
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE EVENT.WORKER_ID IN
                (SELECT WORKER.ID
                FROM WORKER))
```

5. Grafico 4

Nel grafico 4 viene evidenziato il numero di trattamenti svolti da ogni dipendente nel mese corrente.

```
public function numero_trattamenti_per_lavoratore(){
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
 $mese_corrente = date('n');
 $giorno_corrente = date('d');
 $array = array();
```

Eseguo la query:

```
 $dipendenti = $this->Worker->
find('all');
 foreach($dipendenti as $dipendente){
   $numero = $this->
Worker->Event->find('count',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id'])));
```

SQL STANDARD

```
SELECT COUNT(*)
FROM EVENT, WORKER
WHERE MONTH(EVENT.START) =
MONTH(CURDATE())
AND DAY(EVENT.START)
<DAY(CURDATE())
AND EVENT.WORKER_ID =
WORKER.ID
GROUP BY WORKER.ID
```

6. Grafico 5

Nel grafico 5 viene evidenziato il numero di ore effettive lavorate da ogni dipendente nel mese corrente. Questo numero di ore viene calcolato come somma dei tempi di tutti i trattamenti svolti da ogni dipendente.

```
public function ore_lavorate_per_dipendente(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('Event');
 $this->loadModel('EventType');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
$mese_corrente = date('n');
$giorno_corrente = date('d');
$array = array();
$ore_lavorate =0;
```

Eseguo la query:

```
 $dipendenti = $this->Worker->
find('all');
 foreach($dipendenti as $dipendente){
   $events= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id']),
'fields'=>array('Event.event_type_id',
'Event.worker_id')));
   foreach($events as $event) {
     $prezzo = $this->EventType->
find('all',
array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>array('EventType.durata')));
     foreach($prezzo as $price){
       $ore_lavorate= $ore_lavorate+
$price['EventType']['durata'];
     }
   }
 $ore_lavorate = ceil($ore_lavorate/60);
```

SQL STANDARD

```
SELECT EVENTTYPE.ID,
EVENTTYPE.DURATA,
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE EVENT.WORKER_ID IN
                (SELECT WORKER.ID
                FROM WORKER))
```

7. Grafico 6

Nel grafico 6 viene evidenziata la produttività oraria di ogni dipendente nel mese corrente, cioè il rapporto tra profitto e ore effettive lavorate.

```
public function profitto_orario_per_dipendente(){
```

Carico dinamicamente i model di cui necessito:

```
$this->loadModel('Event');
$this->loadModel('EventType');
```

Inizializzo le variabili che serviranno durante l'interrogazione al database:

```
$mese_corrente = date('n');
$giorno_corrente = date('d');
$array = array();
```

Eseguo la query:

```
 $dipendenti = $this->Worker->
find('all');
 foreach($dipendenti as $dipendente){
   $ore_lavorate = 0;
   $profitto = 0;
   $events= $this->Event->
find('all',
array('conditions'=>
array('MONTH(Event.start)'=>
$mese_corrente,
'DAY(Event.start) <'=>
$giorno_corrente,
'Event.worker_id'=>
$dipendente['Worker']['id']),
'fields'=>array('Event.event_type_id',
'Event.worker_id')));
   foreach($events as $event) {
     $prezzo = $this->EventType->
find('all',
array('conditions'=>
array('EventType.id'=>
$event['Event']['event_type_id']),
'fields'=>
array('EventType.prezzo',
'EventType.costo',
'EventType.durata')));
     foreach($prezzo as $price){
       $profitto =
$profitto+$price['EventType']['prezzo']-
$price['EventType']['costo'];
       $ore_lavorate = $ore_lavorate +
$price['EventType']['durata'];
     }
   }
 $profitto_orario =
number_format($profitto/
$ore_lavorate*60,2);
```

SQL STANDARD

```
SELECT EVENTTYPE.ID,
EVENTTYPE.DURATA,
EVENTTYPE.PREZZO,
EVENTTYPE.COSTO
FROM EVENTTYPE
WHERE EVENTTYPE.ID IN
        (SELECT EVENT.EVENT_TYPE_ID
        FROM EVENT
        WHERE EVENT.WORKER_ID IN
                (SELECT WORKER.ID
                FROM WORKER))
```

b. Stato dipendente

Figure 24 - Stato dipendente

2. Caso 2: si autentica un "worker":

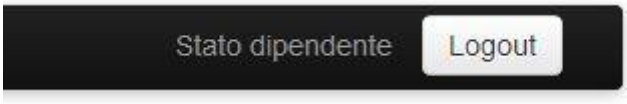

Figure 25 - Navbar "worker"

a. Il link "stato dipendente" è condiviso anche dall'"admin" e visualizza il trattamento di cui si è appena finito di occupare e il prossimo da effettuare.
b. Il link "logout" serve per interrompere la sessione.

3. Caso 3: si autentica uno "user":

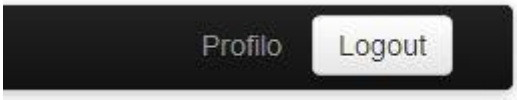

Figure 26 - Navbar "user"

a. Il link "profilo" visualizza il profilo con cui l'utente è registrato e inoltre visualizza lo storico degli ultimi tre trattamenti effettuati presso questa attività.
b. Il link "logout" serve per interrompere la sessione.

a. Profilo

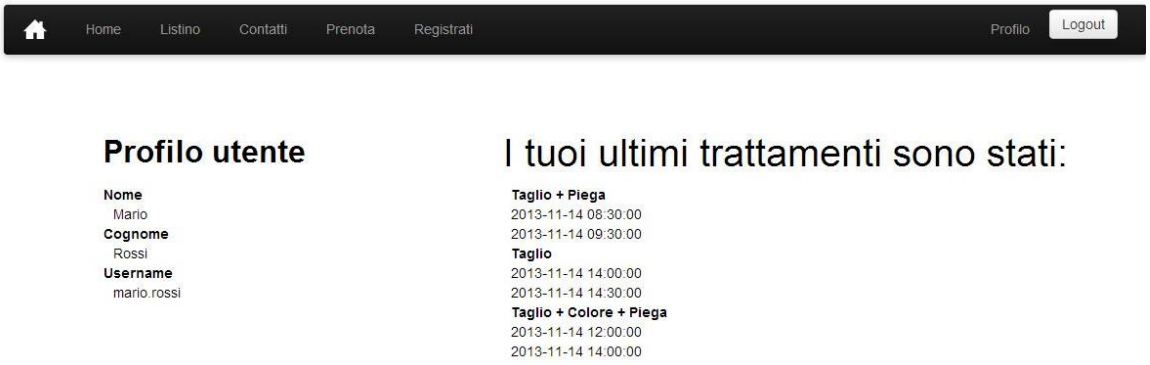

Figure 27 - Profilo utente

5.4 Home (pagina iniziale)

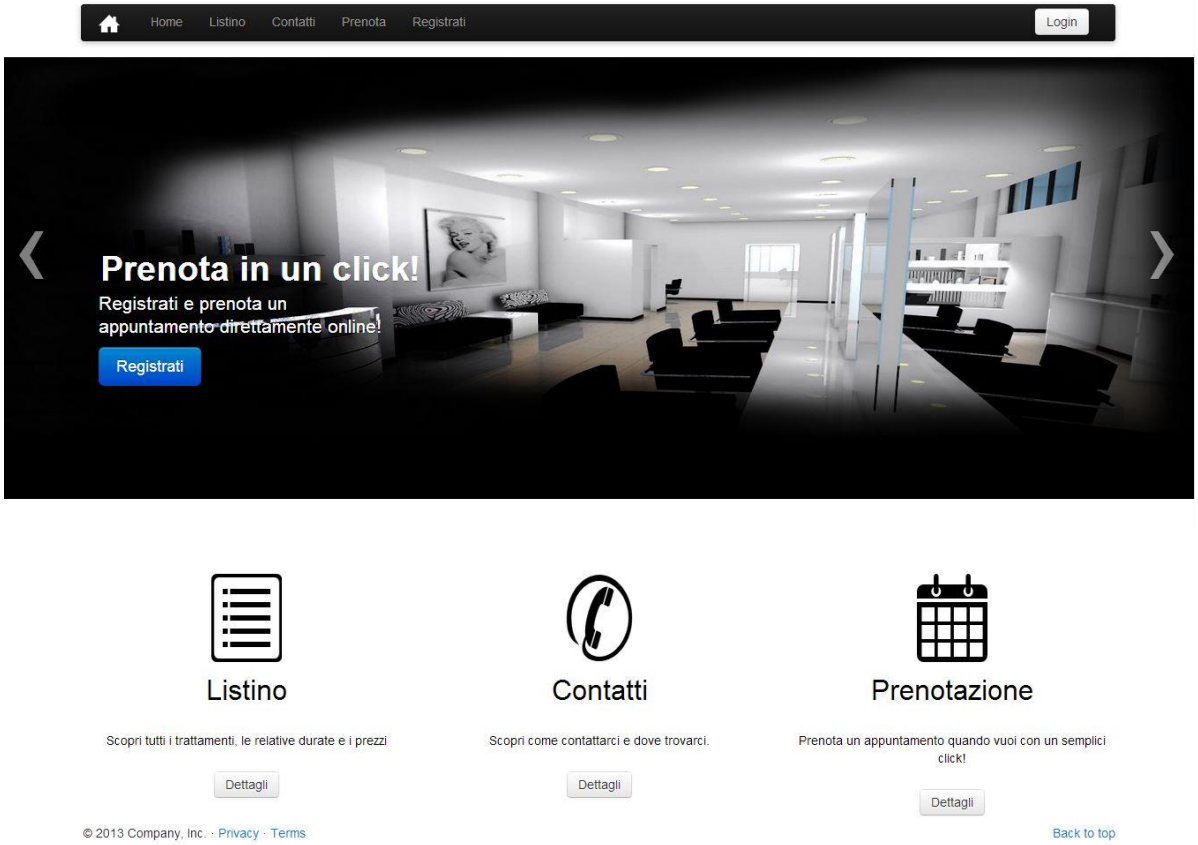

Figure 28 - Pagina iniziale

Nella pagina iniziale è stato sviluppato un "carousel Bootstrap", ossia un "div" all'interno del quale, grazie a un semplice script JavaScript, scorrono in automatico delle immagini. Mentre le icone sono state realizzate appositamente per l'applicazione **[20]**.

5.5 Listino

Il listino elenca tutte le tipologie di trattamenti che possono essere eseguiti. È caricato in maniera dinamica, affinché, qualora si volesse aggiungere, modificare o eliminare una tipologia di trattamento, basterà intervenire sul database stesso. Inoltre è stata differenziata la visualizzazione di alcuni comandi a seconda della persona autenticata.

1. Caso 1: "Admin"

L'"admin" è l'unico che può intervenire direttamente, perciò gli sono stati messi a disposizione questi comandi:

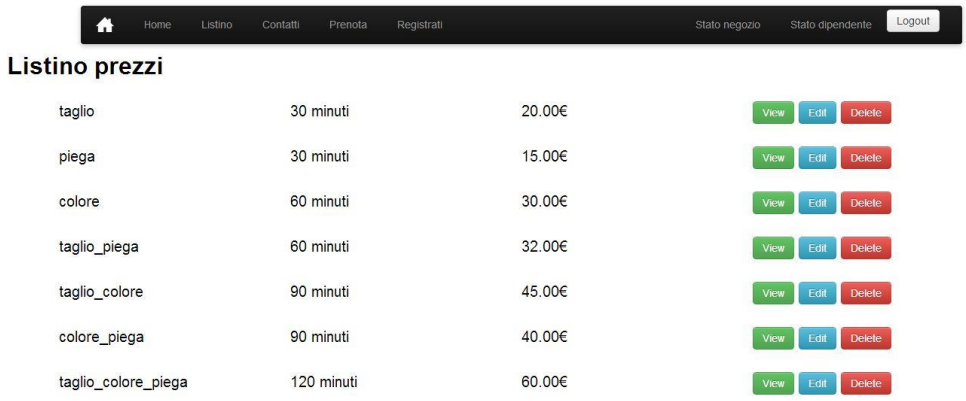

Figure 29 - Listino "admin"

2. Caso 2 "Worker" o "User"

Al contrario, un "worker" o uno "user" non possono modificare le tipologie dei trattamenti, perciò avranno solamente questi comandi:

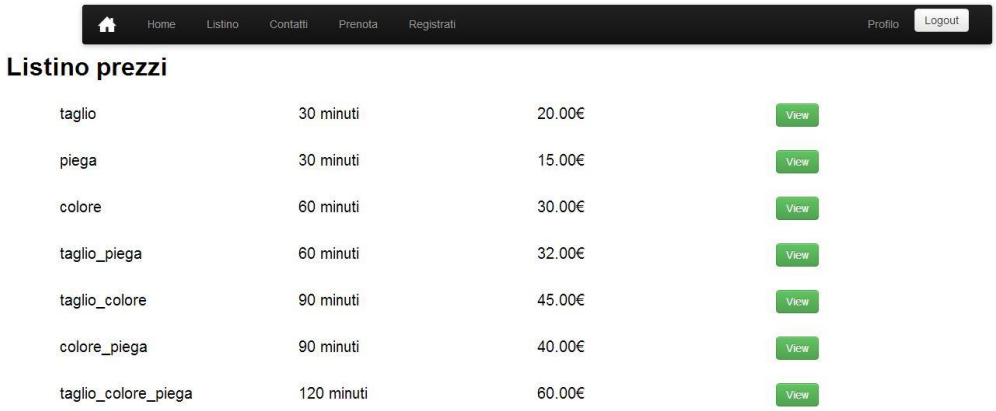

Figure 30 - Listino "worker" o "user"

Cliccando su view è possibile vedere il dettaglio della tipologia di trattamento.

Figure 31 - Dettagli tipo trattamento

5.6 Contatti

La pagina contatti non è stata sviluppata perché troppo specifica relativamente all'attività stessa. Tuttavia essa conterrà semplicemente le informazioni (numero di telefono, email, via ecc.) per contattare l'attività.

5.7 Registra

Abbiamo già presentato il sistema di registrazione e login. Questo link è stato inserito esclusivamente per migliorare la navigabilità del sito.

5.8 Prenota

Analizziamo ora il vero proprio scheduler dell'applicazione, ossia quello strumento che permette agli utenti di prendere appuntamenti online.
Come già anticipato nel capitolo precedente, come supporto per il calendario, è stato utilizzato il framework jQuery FullCalendar.
È stato sviluppato tramite la tecnica AJAX in modo che non si debba effettuare il "refresh" della pagina per visualizzare gli eventi appena inseriti, modificati o cancellati.

Passiamo ad analizzare passo per passo il nostro scheduler.

5.8.1 Situazione iniziale

Inizialmente, abbiamo implementato il "contenitore" di tutta la logica della nostra applicazione. Quindi la nostra situazione è la seguente:

Figure 32 - Scheduler - situazione iniziale

La pagina soprastante è stata realizzata dividendo la parte di semplice visualizzazione dalla parte di gestione del calendario, la quale avverrà tramite uno script JavaScript che comunicherà via AJAX con i vari controllers e quindi con il database.

Tutto il calendario, creato grazie allo script JavaScript "ready.js" verrà creato all'interno del "div" con in "id=#calendar".

Gli eventi esterni rappresentano gli eventi che è possibile scegliere e per essi è stata scelto un colore in modo che fosse distinguibile dagli altri, ma che fosse anche conforme ai colori utilizzati in questa epoca della tecnologia, ossia colori cosiddetti "flat".

5.8.2.Inserimento nuovo evento

L'inserimento di un nuovo evento avviene tramite drag & drop di uno degli eventi esterni, i quali vengono inizializzati dinamicamente nello stesso momento in cui viene creato il calendario.

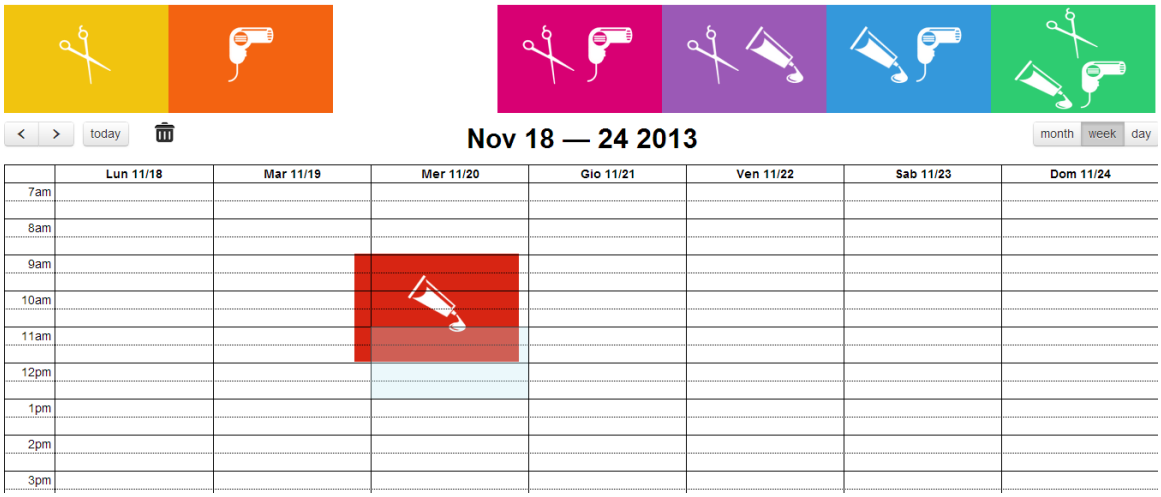

Figure 33 - Scheduler - inserimento drag & drop

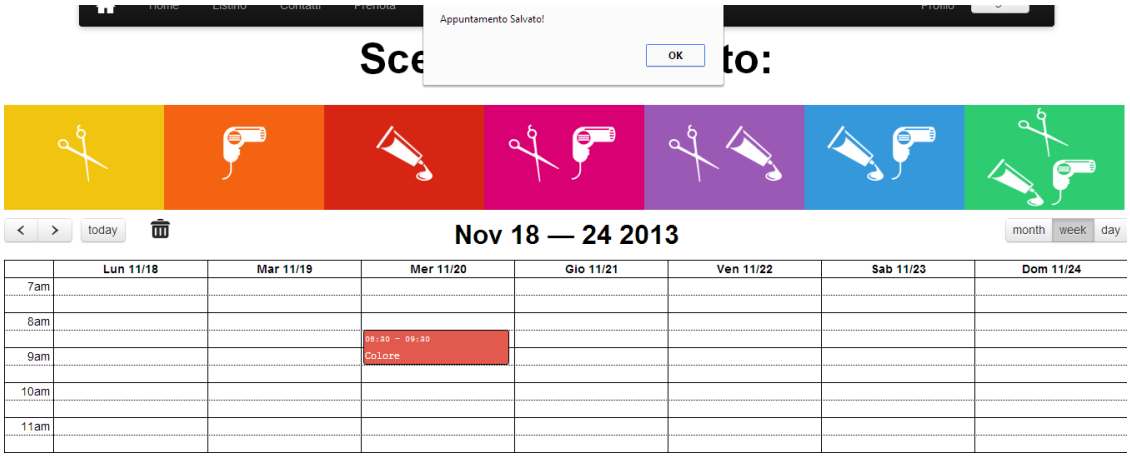

Figure 34 - Scheduler - evento inserito

Gli eventi esterni vengono inizializzati tramite AJAX richiamando il controller "events", in particolare la funzione "carica_tipi". Successivamente l'inserimento avviene dopo il "drop" di uno degli eventi esterni.

L'inserimento avviene tramite una chiamata AJAX al controller events e, in particolare, alla funzione "insert_event" che effettua l'inserimento dell'evento esterno, precedentemente inizializzato, nel database:

```
function insert_event(){
```

Carico dinamicamente i model di cui necessito:

```
 $this->loadModel('Worker');
```

Controllo se quella ricevuta è una richiesta AJAX e assegno i valori ricevuti alle variabili che utilizzerò nell'interrogazione del database:

```
 if($this->RequestHandler->isAjax()){
   $user_id = $this->request->data['user_id'];
   $event_type_id = $this->request->data['event_type_id'];
   $title = $this->request->data['title'];
   $start = $this->request->data['start'];
   $end= $this->request->data['end'];
```

Ora controllo la disponibilità dei dipendenti. L'inserimento viene effettuato non appena si trova il primo dipendente libero:

```
   $workers = $this->Worker->find('all');
   $si_puo_salvare = 0;
   $salvataggio =0;
   foreach ($workers as $worker){
```

```
     $events = $this->Worker->Event-
>find('all',array('conditions'=>array('Event.worker_id'=>$worker
['Worker']['id'],'DAY(Event.start)'=>'DAY($start)'),'order'=>arr
ay('Event.start ASC')));
     foreach($events as $event){
      if($event['Event']['start']<=$start &&
$start<=$event['Event']['end']){
       if($event['Event']['start'] <= $start && $start <=
$event['Event']['end']){
         $si_puo_salvare = 1;
        }
       else{
         $si_puo_salvare = 0;
          }
      }
      else{
           $si_puo_salvare = 0;
       }
     }
     if($si_puo_salvare == 1){
     if($salvataggio == 0){
```

Questa condizione serve per controllare se è già stato fatto l'inserimento nel database. Se è verificata, allora procedo con l'inserimento:

```
          $worker_id=$worker['Worker']['id'];
           $this->Event->create();
          $this->Event->saveField('user_id', $user_id);
          $this->Event->saveField('worker_id', $worker_id);
          $this->Event->saveField('event_type_id',
$event_type_id);
          $this->Event->saveField('title',  $title);
          $this->Event->saveField('start', $start);
          $this->Event->saveField('end', $end);
          $id = $this->Event->id;
          $salvataggio =1;
      }
       else{
  }
}else{
```

Inoltre il cliente che prenota l'appuntamento, ovviamente è lo stesso "loggato", il quale è reperito tramite la funzione "user_corrente" del controller "users":

```
public function user_corrente(){
            $userId = $this->Auth->user('id');
            $this->set("json_user", json_encode($userId));
        }
```

5.8.3 Spostamento evento

Lo spostamento e quindi aggiornamento di un evento avviene sempre tramite drag & drop in uno slot di tempo diverso da quello di partenza.

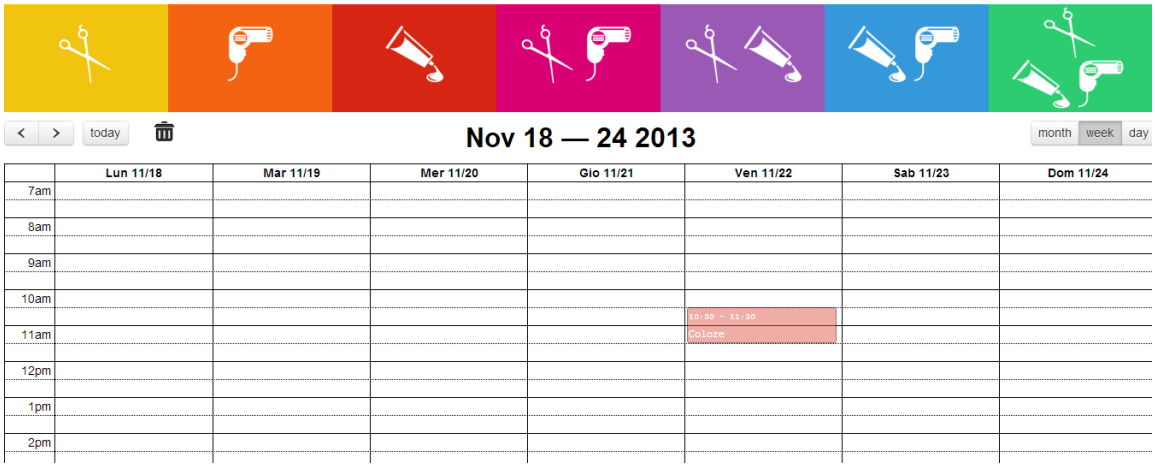

Figure 35 - Scheduler - spostamento drag & drop

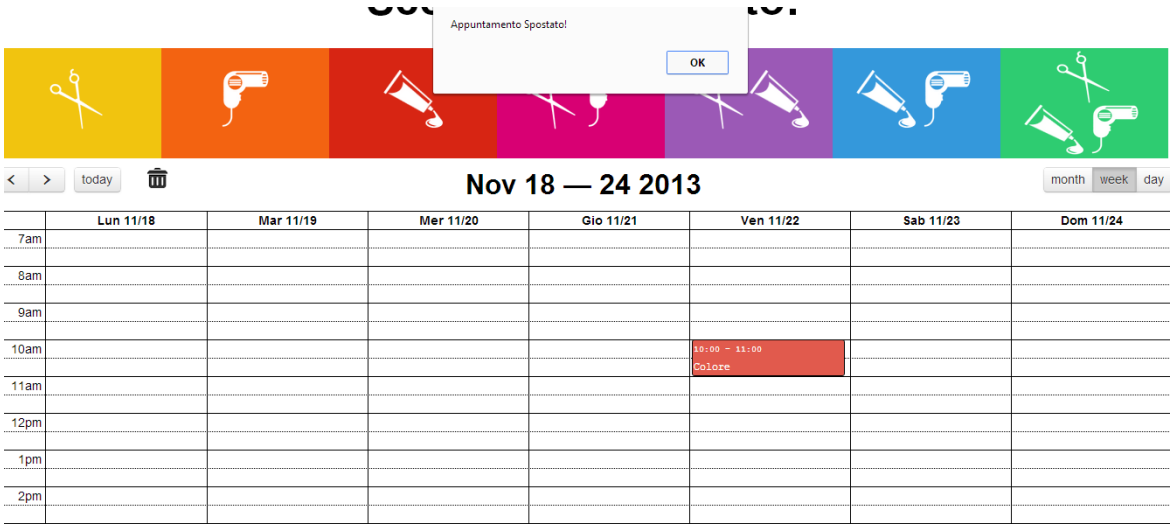

Figure 36 - Scheduler - evento spostato

Più precisamente, lo spostamento avviene dopo il "drop" di uno degli eventi già presenti nel calendario in un altro slot di tempo.
Ovviamente anche lo spostamento avviene tramite una chiamata AJAX al controller events e, in particolare, alla funzione "update_event" che andrà ad aggiornare i campi di inizio e fine dell'evento spostato.

```
function update_event() {
```

Controllo se quella ricevuta è una richiesta AJAX e assegno i valori ricevuti alle variabili che utilizzerò nell'interrogazione del database:

```
 if($this->RequestHandler->isAjax()){
   $this->Event->id = $this->request->data['id'];
   $start = $this->request->data['start'];
   $end= $this->request->data['end'];
```

A questo punto aggiorno i campi di inizio e fine per complementare lo spostamento:

```
   $this->Event->saveField('start', $start);
    $this->Event->saveField('end', $end);
 }
}
```

5.8.4 Eliminazione evento

Come si può immaginare, anche l'eliminazione viene gestita tramite drag & drop, in particolare rilasciando l'evento che si vuole eliminare sull'icona rappresentante un cestino. Per fare ciò è stato intercettato il drag & drop su una precisa posizione del cursore.

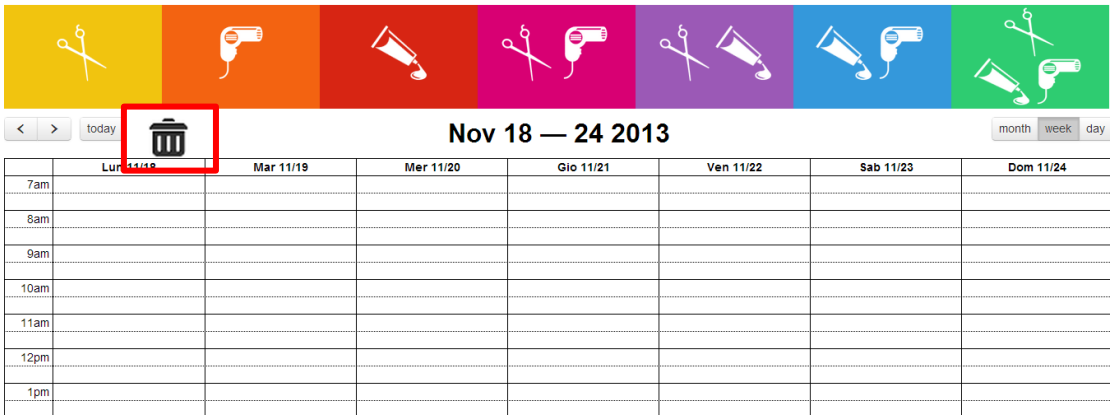

Figure 37 - Scheduler - eliminazione drag & drop

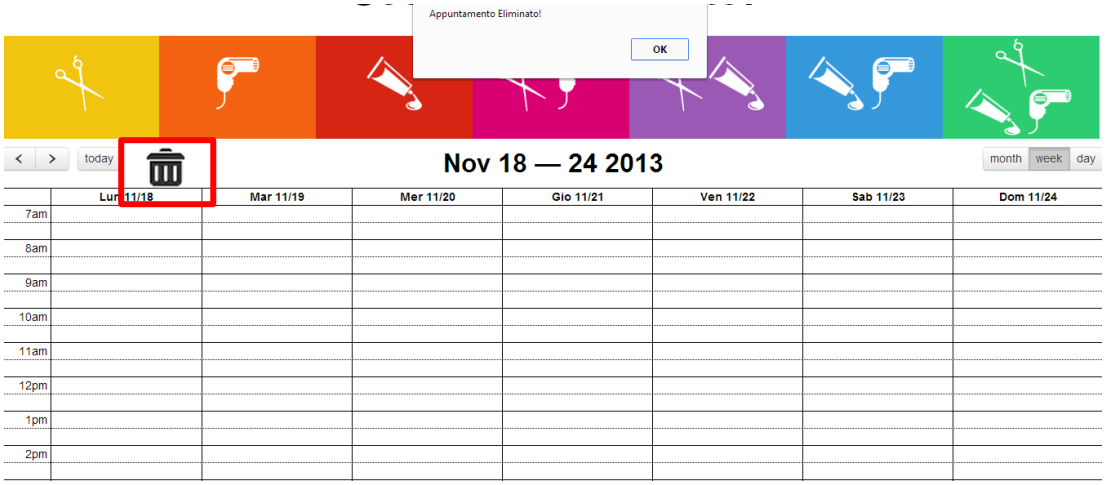

Figure 38 - Scheduler - evento eliminato

Naturalmente anche l'eliminazione avviene tramite una chiamata AJAX al controller events e, in particolare, alla funzione "delete_event" che semplicemente elimina l'evento che abbiamo selezionato durante il drag & drop:

```
function delete_event(){
```

Controllo se quella ricevuta è una richiesta AJAX:

```
  if($this->RequestHandler->isAjax()){
```

Se la condizione è verificata, allora posso procedure con l'eliminazione dal database dell'evento prescelto, il cui id è stato ricavato intercettando l'evento di "drag".

```
    $this->Event->delete($this->request->data['id']);
  }
```

5.8.5 Visualizzazione di tutti gli eventi

Infine, per visualizzare tutti gli eventi salvati nel calendario, compresi quelli appena inseriti, viene richiamata, sempre tramite AJAX, la funzione "feed" del controller "events". La funzione si occupa di recuperare tutti gli eventi presenti nel database che vengono successivamente disposti nello slot di tempo adatto.

```
function feed($id=null) {
```

Controllo se quella ricevuta è una richiesta AJAX (rispettando la metodologia del plugin FullCalendar:

```
 $this->layout = "ajax";
 $vars = $this->params['url'];
```

Eseguo la query che mi recupera gli eventi presenti nella settimana in oggetto, ricavata tramite lo script di FullCalendar:

```
 $conditions = array('conditions' =>
array('UNIX_TIMESTAMP(start) >=' => $vars['start'],
'UNIX_TIMESTAMP(start) <=' => $vars['end']));
 $events = $this->Event->find('all', $conditions);
```

Assegno tutti i valori alle variabili di cui ho bisogno per poter visualizzare ogni caratteristica di un certo evento:

```
 foreach($events as $event){
  $end = $event['Event']['end'];
  $data[] = array(
   'id' => $event['Event']['id'],
   'title'=>$event['Event']['title'],
   'start'=>$event['Event']['start'],
   'end' => $end,
   'url' => Router::url('/') .
'/events/view/'.$event['Event']['id'],
   'details' => $event['Event']['details'],
   'className' => $event['EventType']['name']
   );
 }
}
```

Il risultato finale è il seguente, il quale simula uno scheduler funzionante contenente tutti gli appuntamenti nella settimana dal 4 al 10 novembre 2013.

Figure 39 - Esempio scheduler funzionante

6

CONCLUSIONI E SVILUPPI FUTURI

Il problema oggetto di questo lavoro di tesi è stato la realizzazione di un prototipo di scheduler per attività commerciali.

Questo sistema è nato da un'idea durante la collaborazione con l'azienda Librasoft snc.

L'esigenza di realizzare questo sistema è scaturita dalla necessità di inserire le nuove tecnologie in un ambito in cui sono ancora poco sviluppate e diffuse.

Al fine di individuare le caratteristiche principali del dominio applicativo da modellare e i requisiti per il sistema in oggetto, sono state effettuate numerose interviste con i titolari delle varie attività.

Si è passati poi alla progettazione graduale e, successivamente, è stata svolta un'ampia panoramica sulle nuove tecnologie utilizzate.

Infine è stata presentata, passo dopo passo, la sua realizzazione.

Sebbene molte funzionalità richieste al sistema siano già state implementate in questo lavoro di tesi, alcuni aspetti dovranno essere ulteriormente sviluppati in futuro.

In particolare si dovrà dare la possibilità ad un titolare di gestire più attività contemporaneamente, utilizzando un solo account. Inoltre, si dovrà procedere con l'integrazione con Facebook, nonostante fosse stata lasciata da parte per questo lavoro. In più, per venire incontro ulteriormente alle esigenze degli utenti, si potranno sviluppare applicazioni per iPhone e smartphone.

BIBLIOGRAFIA E SITOGRAFIA

[1] Arlow J, Neustadt I., *Uml 2 e unified process*, 2° ed. McGraw-Hill, 2006

[2] Atzeni P., Ceri S., Paraboschi S, Torlone R., *Basi di dati, modelli e linguaggi di interrogazione*, 3° ed. McGraw-Hill, 2009

[3] Stobart S., Vassileiou M., *PHP e MySQL. Guida completa*, Apogeo, 2004

MySQL, http//:www.mysql.it

[4] Van Der Lans R., *Introduzione a SQL*, 2° ed. Addison-Wesley, 2001

[5] *Wampserver*, http//:www.wampserver.com, 09/09/2013

[6] *PhpMyAdmin*, http//:www.phpmyadmin.net/home_page/index.php, 09/09/2013

[7] Tansley D., *Pagine web dinamiche con PHP e MySQL*, Pearson, 2002

[8] Wandschneider M., *Sviluppare applicazioni web con PHP e MySQL*, Apogeo, 2006

[9] Cake Software Foundation, *CakePHP Cookbook Documentation – Release 2.x*, 2013

Golding D., *Beginning CakePHP – From novice to professional*, Apress, 2008

CakePHP, http//:www.cakephp.org, 10/09/2013

[10] Chan K, Omokore J., Millar R.K., *Pratical CakePHP Projects*, Apress, 2009

[11] *jQuery*, http//:www.jquery.com, 02/10/2013

[12] *jQuery UI*, http//:www.jqueryui.com, 13/10/2013

[13] Cochran D., *Twitter Bootstrap web development how-to*, [Packt]* Publishing, 2012

Bootstrap, http//:www.getbootstrap.com, 20/10/2013

[14] *FullCalendar*, http//:www.arshaw.com/fullcalendar/, 25/10/2013

[15] *Stack Overflow*, http//:www.stackoverflow.com, 23/10/2013

[16] Castro E., Hyslop B., *Html5 e Css3 per il World Wide Web*, Hops, 2012

[17] Gigliotti G., *Html5 e Css3*, Apogeo, 2011

[18] *jqBarGraph: jQuery Bar Graph Plugin*, http//:www.workshop.rs/jqbargraph/, 06/11/2013

[19] *W3Schools Online Web Tutorials*, http//:www.w3schools.com, 20/09/2013

[20] *The Noun Project*, http//:www.thenounproject.com, 06/11/2013