# UNIVERSITÀ DEGLI STUDI DI PARMA

**Facoltà di Scienze Matematiche, Fisiche e Naturali**

**Corso di Laurea in Informatica**


---

## Analisi, progettazione e realizzazione di un sito web per la Cooperativa *Il Piccolo Principe* di Reggio Emilia


**Relatore:**

**Prof. Eduardo Calabrese**

**Laureanda:**

**Anna Celada**



**ANNO ACCADEMICO 2005-2006**

*A mia sorella Chiara e ai miei genitori, per la fiducia che hanno sempre riposto in me e per avermi sempre spronata a portare a termine questo cammino.*

# Indice

# 1 Introduzione

Il sito che ho realizzato è stato commissionato dalla Cooperativa *Il Piccolo Principe* di Reggio Emilia la quale si occupa di organizzare attività di vario tipo per le persone diversamente abili. Attualmente sono stati attivati otto progetti e per ognuno di essi è stato adeguatamente formato un gruppo di volontari che si occupa della messa in pratica delle attività proposte dalla Cooperativa stessa. L'obiettivo principale che si vuole raggiungere con il sito è dedicare uno spazio web a quest'importante progetto che da anni si occupa di un'intenzione così nobile. Si cerca dunque di dare una possibilità in più per rafforzare lo spirito di gruppo tra i volontari e i responsabili che vi operano attivamente attraverso spazi appositamente dedicati come il forum e di creare uno spazio per i disabili e i loro familiari nel quale raccogliere foto e racconti delle esperienze più significative realizzate con il Piccolo Principe. La presenza del sito nel World Wide Web è oltresì un modo per far conoscere l'esistenza della cooperativa a chi ancora non ha avuto modo di farlo: nel portale si potranno trovare informazioni sulle attività in programma e i contatti dei referenti delle stesse in modo da permettere a chiunque di conoscere ed eventualmente prendere parte a tali progetti.

## 1.1 Struttura del sito

Per renderci meglio conto della struttura effettiva del progetto vediamo in concreto quali sono i file principali che lo costituiscono. Partiamo da *index.php* ovvero il codice sorgente della home page. In esso troviamo la struttura che compare in tutti gli altri file con estensione php. Fra i tag `<head>...</head>` troviamo il primo *include* che richiama il file *inclusioni.php* ci rimanda al foglio di stile (*stile.php*) e al codice dello script per il menu a bottoni visibile nella parte sinistra dello schermo. Chiuso l'header

si apre il corpo del documento nel quale troviamo l'inclusione di (*inizioPagina.php*) che contiene il contatore e l'impaginazione della parte superiore della pagina. A questo punto seguono il titolo della sezione e il contenuto specifico del file in questione. Infine troviamo un ultimo *include* che richiama il file *finePagina.php* in cui vengono chiusi tutti i tag precedentemente aperti, che invoca i menu laterali da visualizzare e impagina la parte inferiore del documento.

La seconda voce del menu ci porta al file *progetti.php* che spiega brevemente al navigatore come poter far parte della cooperativa ed elenca le attività attualmente disponibili. Ognuna di esse costituisce un link ad una nuova pagina che spiega in modo più dettagliato il senso del progetto e com'è strutturato. In seguito a tali spiegazioni si trova l'elenco dei contatti a cui fare riferimento per chiedere ulteriori informazioni. I dati presenti nell'elenco vengono estrapolati dal database e quindi sono aggiornabili direttamente da esso senza dover apportare modifiche al codice php.

Procediamo con le prossime due opzioni del menu: la prima spiega cosa occorre fare per diventare un volontario del Piccolo Principe e chi contattare *comeDiventareVolontario.php*, la seconda continua a parlare del percorso di formazione del volontario.

Passiamo ora alla pagina successiva interamente aggiornata dall'amministratore del sito. Nella sezione dedicata agli eventi (*eventi.php*) troviamo un elenco di appuntamenti ordinati per data di inserimento che informano i visitatori delle novità di vario genere legate alla cooperativa. Segue la pagina dedicata alla ricerca di parole chiave all'interno del portale per trovare più rapidemente determinate informazioni (*search.php*). Scorrendo il menu si arriva all'album, destinato a contenere le foto degli eventi ritenuti più significativi dall'amministratore.

In un sito che si prefigge come obiettivo quello di sensibilizzare le persone e promuovere le iniziative organizzate dal Piccolo Principe non può mancare una newsletter (*newsletter.php*) alla quale si può iscrivere chiunque ne abbia la necessità. La procedura di iscrizione è estremamente semplice: si tratta di compilare il campo appropriato con il proprio indirizzo email e cliccare sul bottone di conferma. Il sistema controlla se l'indirizzo immesso è già presente nel database o meno, e in quest'ultimo caso, procede con l'iscrizione dell'utente. Viene inviato un messaggio alla casella di posta inserita affinchè l'utente possa completare il tutto. Questa fase funge anche da controllo e da tutela della base di dati perché si evita che vengano introdotti indirizzi inesistenti nel sistema.

L'ultima voce del menu porta alla pagina del login (*login.php*) mediante la quale si può accedere alla parte privata del sito. Al centro del foglio si trovano il campo per il nome dell'utente e la password, affiancati dal bottone di invio. L'utente che accede per la prima volta a questo spazio si può iscrivere compilando il form contenuto in *registrati.php*. I campi presenti sono 15 di cui 6 obbligatori. Una volta immessi i dati viene prima richiesta una conferma tramite mail all'utente, così come avviene per la newsletter, in un secondo tempo viene presentata la richiesta di iscrizione all'amministratore del sito cui spetta la decisione di accettare o meno la creazione del nuovo account.

Abbiamo esaminato a questo punto tutte le voci del menu inerente alla parte pubblica del sito. Le stesse sono presenti nella versione inglese accessibile mediante il link a forma di bandiera inglese posto nella cornice superiore della schermata. Vediamo ora il menu che compare dopo essersi loggati come utenti.

La prima pagina è quella di accoglienza (*benvenuto.php*) nella quale viene visualizzato un messaggio di benvenuto seguito dal nome dell'utente. Pas-

siamo alle parti più significative di questa sezione a partire dal forum. Si tratta di un servizio che permette agli utenti di comunicare tra loro compilando un semplice form. I files che costituiscono il forum sono i seguenti: *indexForum.php* nel quale compare l'elenco degli argomenti di discussione, *forum.php* visualizza i topics aperti dagli utenti all'interno di un determinato argomento di discussione e visualizza la data dell'ultimo messaggio inserito. I files *nuovo.php* e *risposta.php* sono dedicati all'inserimento di nuovi messaggi. Per visualizzare tutti i messaggi presenti in corrispondenza di un particolare topics si accede alla pagina *thread.php*.

Passando alla voce successiva dal menu laterale si giunge al profilo dell'utente contenente i dati personali inseriti nel momento della creazione dell'account. Alcuni di essi non sono modificabili mentre altri possono essere modificati, rimossi o inseriti se precedentemente non erano stati compilati. I files php cui mi riferisco sono *modificaProfilo.php* e *verificaProfilo.php*.

Come ultima voce troviamo il logout che permette all'utente di chiudere la sessione aperta in precedenza.

Passiamo all'ultimo menu che compare dopo essersi loggati come amministratore. La prima pagina che si apre contiene un messaggio di benvenuto così come avveniva con il generico utente. Successivamente troviamo la voce profilo che permette all'amministratore di modificare la password di accesso al portale (*modificaProfiloAdmin.php* e *verificaProfiloAdmin.php*). In seguito troviamo il link crea newsletter che ci rimanda al form per la creazione e l'invio di email che verranno inoltrate a tutti gli indirizzi presenti nel database. Procedendo con la nostra analisi del menu incontriamo le voci crea eventi e cancella eventi che permettono l'inserimento e/o la cancellazione di eventi nella pagina pubblica eventi.php. I files coinvolti sono: *creaEventi.php*, *creaEventi2parte.php*, *eventiCancella.php* e *eventiCancella2.php*.

L'amministratore può inoltre visualizzare l'elenco di tutti i nominativi registrati nel database relativi ai volontari e ai responsabili. Le azioni che può fare sono di vario tipo: ha la facoltà di attivare/disattivare gli account dei volontari, cancellarli dalla baase di dati, inserire nuovi nominativi in riferimento ai responsabili e visualizzare tutti i dati presenti nell'ordine preferito. Per fare tutto ciò ho creato le seguenti pagine: *elencoVolontari.php*, *cancellaIscritto.php*, *cancellaIscritto2.php*, *DisattivaIscritto.php*, *AttivaIscritto.php*, *elencoResponsabili.php*, *cancellaResponsabile.php*, *cancellaResponsabile2.php*, *inserisciResponsabile.php*, *verificaRegistrazioneResponsabile.php*.

# 2 Analisi

## 2.1 Analisi dei requisiti

Il portale è dedicato a tre categorie di utenti: il generico navigatore del web, che può attingere informazioni sulla cooperativa e aggiornarsi sugli eventi da essa organizzati, può informarsi su come vengono formati i volontari nel caso fosse sua intenzione prendere parte al Piccolo Principe.
La seconda categoria è appunto quella dei volontari i quali sono già parte attiva del sistema; la parte del portale che essi potranno utilizzare è, oltre quella pubblica, la sezione privata. In essa troveranno un forum che potranno utilizzare per lo scambio di idee con tutti i volontari iscritti al portale. Inoltre, sempre all'interno della suddetta sezione, essi potranno chiedere consigli al responsabile della formazione dei volontari, il quale veste il ruolo di moderatore di una parte del forum stesso.
L'ultima categoria che resta da esaminare è quella dell'amministratore del portale: esso ha la possibilità di modificare i contenuti del sito per poterlo mantenere vivo e in costante aggiornamento. Egli ha anche il compito di gestire le iscrizioni da parte dei volontari. Senza la sua approvazione non sarà possibile accedere ai contenuti privati; questo controllo viene eseguito per garantire la sicurezza e la privacy delle informazioni contenute in tale sezione.

Gli obiettivi principali per i quali è stato realizzato il progetto sono i seguenti:

- realizzare uno spazio sul web che funga da mezzo comunicativo per le persone che partecipano ai progetti della Cooperativa e, al contempo, da mezzo pubblicitario per far conoscere ad un maggion numero di persone l'esistenza di una struttura di tale importanza quale è il Piccolo

Principe.

- fornire ai volontari uno strumento per la comunicazione più rapida e semplice di informazioni di vario tipo e incentivare la crescita in ognuno di loro di uno spirito di gruppo che aiuti ogni singola persona a crescere come volontario e come individuo grazie al confronto e allo scambio di opinioni che può avvenire fra le pagine del portale ad essi dedicato.

## 2.2 Elenco degli attori

Gli attori sono le persone che interagiscono con il sistema, essi hanno precisi diritti e doveri. Qui di seguito analizziamo perciò chi sono gli attori e quali sono le loro caratteristiche.

**Amministratore** - E' colui che si carica dell'onere della gestione del sito. Egli funge da moderatore relativamente alle richieste di iscrizioni da parte dei volontari i quali, per ottenere un proprio account, devono attendere che l'amministratore li attivi. Questa figura deve inoltre preoccuparsi di tenere aggiornata la parte riguardante i contenuti dell'intero portale (le pagine pubbliche, ad esempio), ed è il solo che può inserire nuovi eventi nell'apposita sezione.

**Disabile** - Letteralmente significa persona diversamente abile. E' colui che usufruisce dei servizi offerti dalla Cooperativa e può usufruire dei servizi inerenti alla parte pubblica del sito.

**Moderatore** - Il ruolo del moderatore si riferisce a una persona che coordina e ha la facoltà di prendere delle decisioni. Oltre all'amministratore del sito abbiamo un altro moderatore che corrisponde alla figura del

responsabile del progetto della formazione che coordina la parte del forum dedicata a tale attività.

**Responsabile** - Questo ruolo è rivestito da più persone nell'ambito dei diversi progetti. Ognuno di essi coordina le attività e le persone che fanno parte di un determinato progetto. I loro nominativi sono stati inseriti in apposite pagine del sito affinchè il navigatore possa, in caso di necessità, mettersi in contatto con costoro.

**Utente/Navigatore** - E' colui che, pur non essendo un volontario o un responsabile, accede al sito per consultare le informazioni contenute nelle pagine pubbliche.

**Volontario** - Colui che accompagna i disabili nel corso delle attività organizzate dalla cooperativa. I volontari costituiscono il cuore dei progetti perché senza di essi non sarebbe possibile realizzare alcun tipo di attività. Essi possono registrarsi al portale ed ottenere uno speciale account che permetterà loro di accedere ai contenuti privati del sito per scambiarsi opinioni vicendevolmente.

Tutte queste persone possono essere utenti più o meno esperti del web; per questo motivo uno dei requisiti che è necessario tenere sempre ben presente è la semplicità della struttura del sito stesso per agevolare la navigazione fra le sue pagine.

## 2.3 Diagramma dei casi d'uso

Nei paragrafi precedenti abbiamo dato uno sguardo agli attori che intervengono nel nostro sistema e alle attività che svolgono al suo interno. Quello che dobbiamo fare ora è mettere in relazione le due cose.
A tal scopo ci viene in aiuto il diagramma dei casi d'uso, più comunemente

chiamato *Use Case Diagram*. Un caso d'uso corrisponde ad una particolare modalità di interazione col sistema. Il loro utilizzo aiuta il progettista e il committente a capire quali sono i requisiti che dovrà soddisfare il progetto. I casi d'uso fungono al contempo da guida per l'intero progetto di sviluppo in quanto costituiscono un solido punto di partenza per la progettazione del sistema.

Nella notazione UML (*Unified Modeling Language*) un caso d'uso è rappresentato da un ovale mentre un attore è indicato mediante un omino stilizzato. Per rappresentare la comunicazione tra queste due parti si utilizzano delle rette non orientate che collegano appunto le due figure.

Torniamo al nostro servizio web e analizziamo a fondo quali sono i possibili modi di interazione tra gli attori e le risorse che esso offre.

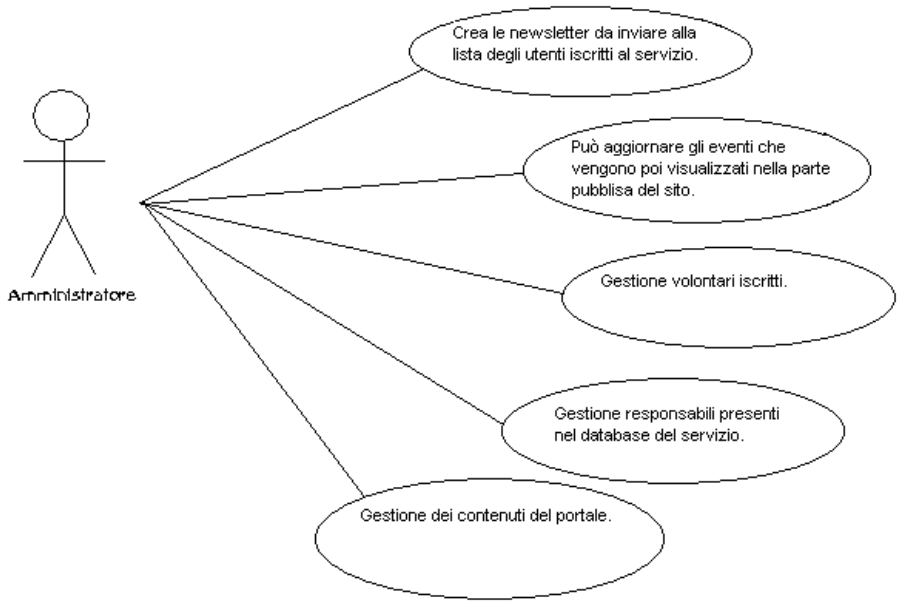


Figura 1: Use case diagram relativo all'amministratore.

In figura 1 vediamo illustrato il diagramma dei casi d'uso relativi all'am-

ministratore (fig.1). Egli ha la facoltà di eseguire differenti operazioni per manipolare i dati che popolano il database cui fa riferimento il sistema e può inoltre tenere sempre aggiornati i contenuti delle pagine pubbliche.

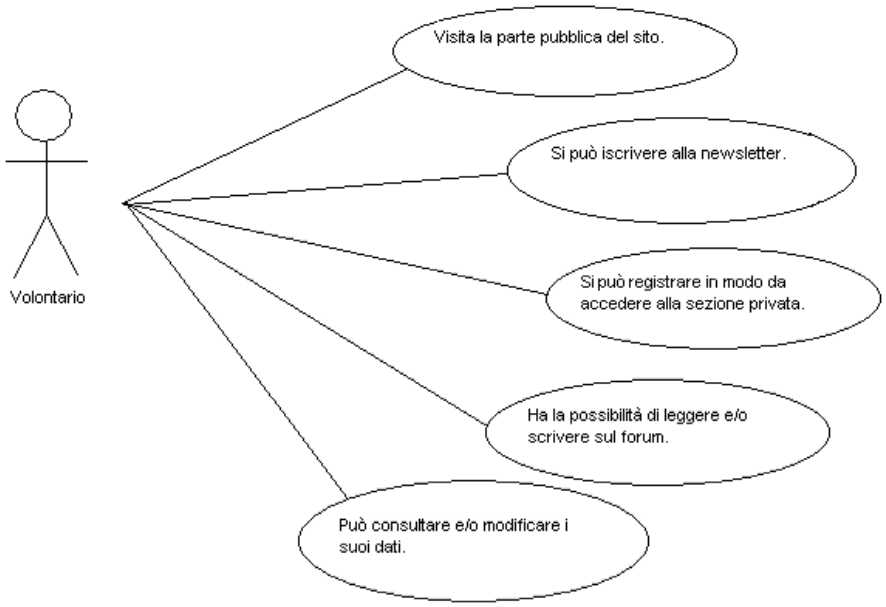


Figura 2: Use case diagram relativo al volontario.

Vediamo ora il diagramma relativo al volontario (fig.2). Egli può compilare il modulo di registrazione che gli permetterà di inviare la richiesta di un nuovo account personale all'amministratore sopra citato. Come prima verifica verrà inoltrata un'email all'indirizzo inserito nel campo appropriato onde evitare l'inserimento di dati fasulli nel database con spiacevoli inconvenienti. L'email che riceverà l'utente contiene uno speciale link che permette di completare la domanda di iscrizione. Se l'amministratore deciderà di accettare tale richiesta, il volontario avrà il suo nuovo account che gli permetterà di accedere alla sezione privata.

Una volta ottenuto tale permesso egli si potrà loggare per mezzo di un sem-

plice form; in seguito a tale operazione comparirà sotto al menu principale un secondo menu con i link ai contenuti supplementari.
Tra le varie pagine troverà in particolare un forum diviso in più argomenti, uno dei quali è dedicato alla *formazione* ed è coordinato dal responsabile del suddetto progetto.

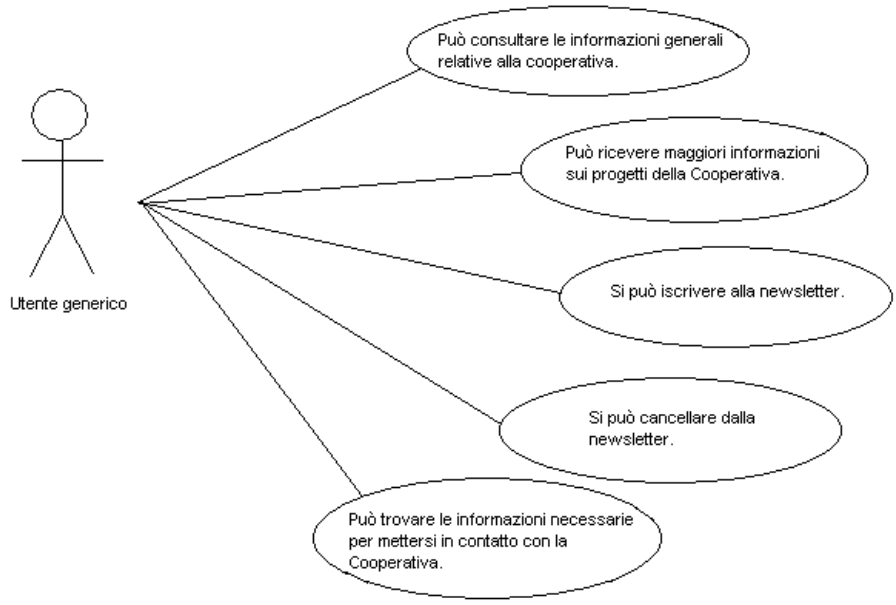


Figura 3: Use case diagram relativo al generico utente.

L'utente generico (fig.3) è colui che ha accesso solamente alle pagine pubbliche. Egli può trovare ricche notizie sulla cooperativa e sui progetti da essa gestiti. Sarà in grado di reperire le informazioni necessarie per comunicare direttamente coi responsabili dei progetti menzionati. Un'altra possibilità per essere continuamente informato sulle iniziative organizzate (come uscite, gite, ecc.), è quella di registrarsi all'efficiente servizio di newsletter. Anche in questo caso il sottoscrivente riceverà nella sua casella di posta un'email per

completare l'iscrizione, in modo da impedire che un cattivo utente inserisca email di altre persone oppure indirizzi non validi. Per confermare la richiesta basterà quindi cliccare sul link ricevuto nella mail, un'operazione semplice e quindi alla portata di tutti!

## 2.4 Prototipo di analisi

Il prototipo di analisi costituisce un potente strumento per agevolare la comunicazione tra il cliente e l'analista. Quest'ultimo mostra le interfacce utente del futuro sistema all'acquirente, in modo tale da permettergli di avere un'idea precisa di quale sarà il risultato finale del lavoro. In questo modo viene statisticamente ridotto il numero delle modifiche dei requisiti che dovranno essere apportate al sistema.
Vediamo ora quali sono i prototipi di analisi del nostro progetto.

Qui di seguito il prototipo della pagina destinata al login (fig.4), composta di due campi (rispettivamente *utente* e *password*), seguiti dal bottone *ok* per inviare al database i dati immessi. Al di sotto del form si trova il collegamento alla pagina che permette di registrare un nuovo utente al servizio.

Vediamo perciò il prototipo della pagina d'iscrizione al portale (fig.5): essa è composta da vari form. Alcuni campi sono contrassegnati da un asterisco per ricordare all'utente che sono dati necessari senza i quali non è possibile proseguire alle fasi successive. Tutti gli altri spazi sono invece facoltativi e possono essere lasciati in bianco senza alcuna segnalazione di errore (come succede invece nel caso precedente). In fondo alla pagina si trovano il bottone *cancella* che permette di risettare allo stato iniziale il form, in altre parole cancella tutti i dati inseriti perdendone la loro memoria. Il secondo

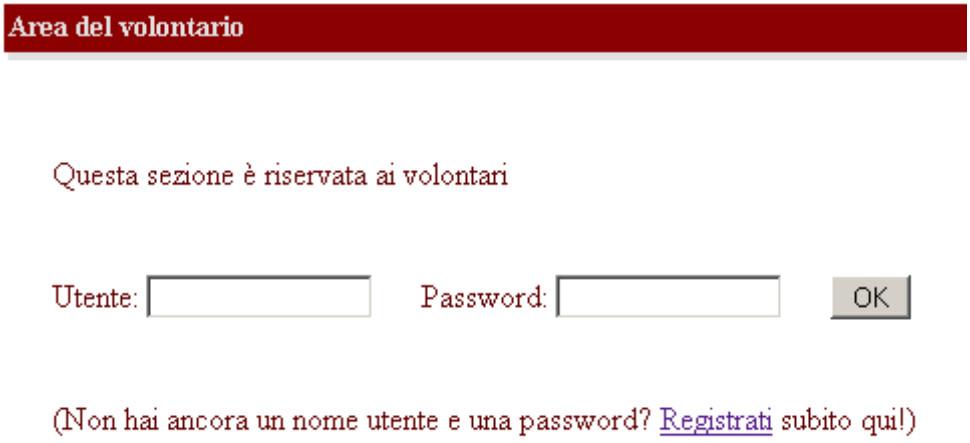


Figura 4: Prototipo di login.

tasto, *invia*, è invece quello che permette di inviare i dati al database e di iniziare così la procedura di registrazione.

I tre prototipi che seguono sono utilizzabili solo dall'amministratore. Il primo di essi (fig.6) è composto di un primo campo denominato *oggetto* nel quale dev'essere inserito il titolo dell'email da inviare alla lista di utenti iscritti. Il secondo invece è destinato a contenere il corpo del messaggio da inoltrare.

Il secondo (fig.7) riguarda il modulo che permette di inserire nuovi eventi all'interno del database, che verranno visualizzati nelle pagine pubbliche e quindi visibili a tutte le categorie di utenti. Oltre ai campi standard *titolo* e *messaggio* analoghi ai precedenti, ho aggiunto il campo *data* il quale, se compilato, restituirà la stringa immessa dell'amministratore, se lasciato in bianco restituirà invece la data in cui il messaggio in questione è stato inserito.

Relativamente al campo dedicato al corpo del messaggio sono stati aggiunti


Figura 5: Prototipo di registrazione.

quattro bottoni che permettono di formattare a proprio piacimento il testo inserito. Essi permettono rispettivamente di scrivere una porzione di testo in grassetto, corsivo e sottolineato. L'ultimo di essi permette invece di ordinare i dati come un elenco.

L'ultimo componente della terna è l'interfaccia per visualizzare l'elenco com-


Figura 6: Prototipo per la creazione di newsletter.

pleto dei volontari registrati (fig.8). L'elenco è comprensivo di due categorie di utenti: coloro che sono già attivi e quelli che invece hanno risposto all'e-mail ricevuta dopo aver compilato tutti i campi (in fase di registrazione), ma ancora non possono loggarsi. Questi ultimi hanno il campo *active* settato a *1* e questo impedisce che essi possano accedere al menu supplementare. Per attivare un utente è necessario l'intervento dell'amministratore, il quale deve premere il bottone *attiva* posto sotto al nominativo interessato. E' possibile anche l'operazione inversa, ovvero la disattivazione di un utente attivo (la procedura è la medesima). Il bottone *cancella* si commenta da solo, se non per il fatto che prima di eliminare definitivamente il nominativo viene chiesta una conferma per tutelare l'integrità dei dati presenti nel database da errori commessi accidentalmente da un amministratore distratto!

Segue l'interfaccia per l'iscrizione alla newsletter che permette di inserire il


Figura 7: Prototipo per l'inserimento di nuovi eventi.

proprio indirizzo email nel database della cooperativa, in modo da essere sempre informati sulle iniziative da essa organizzate. L'utente non deve far altro che inserire il proprio indirizzo di posta elettronica nell'apposito spazio e premere il tasto *invia*. Per evitare che il database si popoli di indirizzi fasulli o di terze persone, l'iscrizione verrà considerata valida solo nel momento in cui l'utente accederà all'indirizzo contenuto nell'email inviatagli (fig.9).

Infine diamo uno sguardo al forum, visibile solo dopo aver effettuato la procedura di login. E' possibile creare nuovi *topics*, cioè aprire nuovi dibattiti (in tal caso è obbligatorio inserire il titolo), oppure si può semplicemente rispondere ai messaggi presenti. Anche qui troviamo due bottoni che si com-


Figura 8: Prototipo per visualizzare l'elenco dei volontari iscritti.

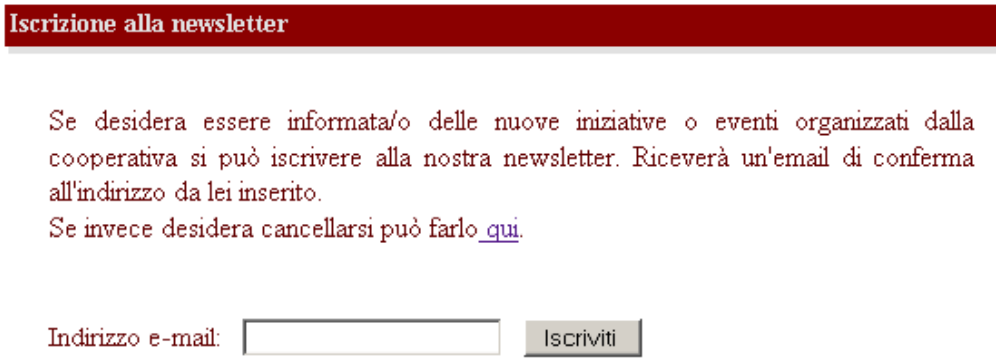


Figura 9: Prototipo per l'iscrizione alla newsletter.

mentano da soli: *cancella* (per svuotare i campi nel caso di errori), e *invia* (fig.10).

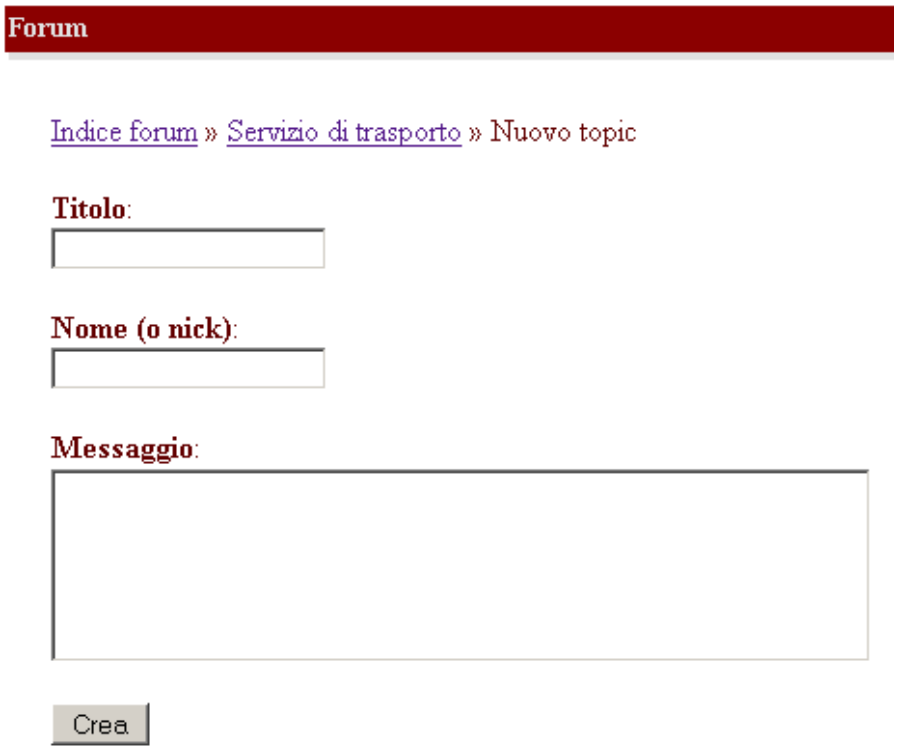


Figura 10: Prototipo per l'inserimento di messaggi nel forum.

## 2.5 Sequence Diagram

Il diagramma delle sequenze (anche detto *Sequence Diagram*) permette di rappresentare lo svolgimento temporale delle interazioni tra gli oggetti. Analizziamo innanzitutto gli elementi che compaiono nei diagrammi che seguono. Secondo la notazione UML gli oggetti vengono rappresentati sottoforma di rettangoli al di sotto dei quali è posta una linea tratteggiata verticale che scende verso il basso rappresentante la durata di vita dell'oggetto. Nel momento in cui quest'ultimo assume il controllo del flusso esecutivo del programma, la linea tratteggiata viene sostituita con un riquadro verticale che ricalca la linea originaria. Per indicare la comunicazione tra due oggetti si utilizzano delle frecce orizzontali. La freccia con la punta piena mostra

una chiamata a un metodo mentre la freccia con la punta aperta mostra un messaggio che restituisce il controllo del flusso esecutivo al programma. Vediamo ora due diagrammi relativi all'interazione dell'amministratore con il portale.

Il vantaggio di utilizzare dei sequence diagram è la possibilità di modellare

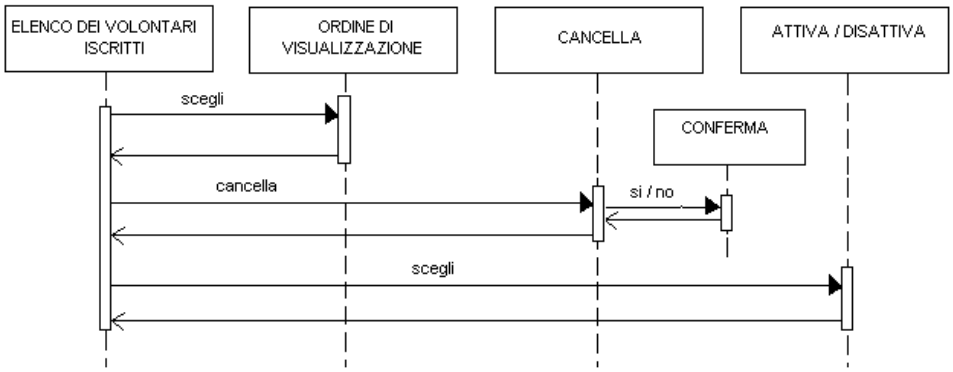


Figura 11: State diagram per la gestione dei volontari iscritti al sito.

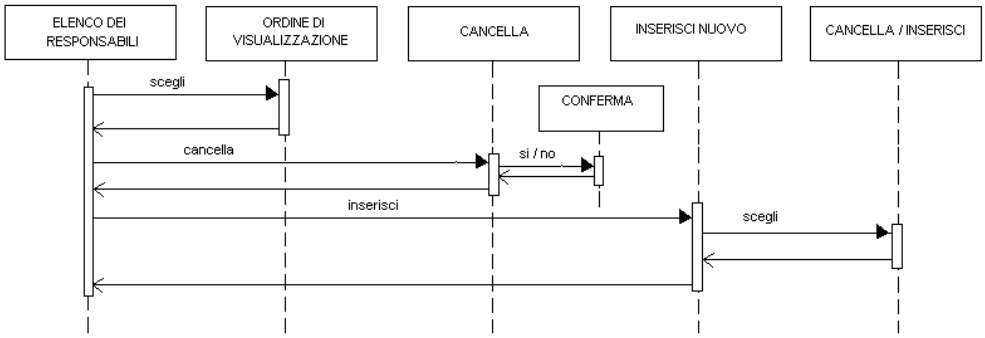


Figura 12: State diagram per la gestione e l'inserimento dei responsabili.

le interfacce rappresentate verso altri sistemi. A questo livello di analisi i metodi vengono considerati solo superficialmente e servono a rappresentare le funzionalità più significative. La strutturazione interna esatta è invece lasciata in sospeso perché verrà trattata in fase di progettazione.

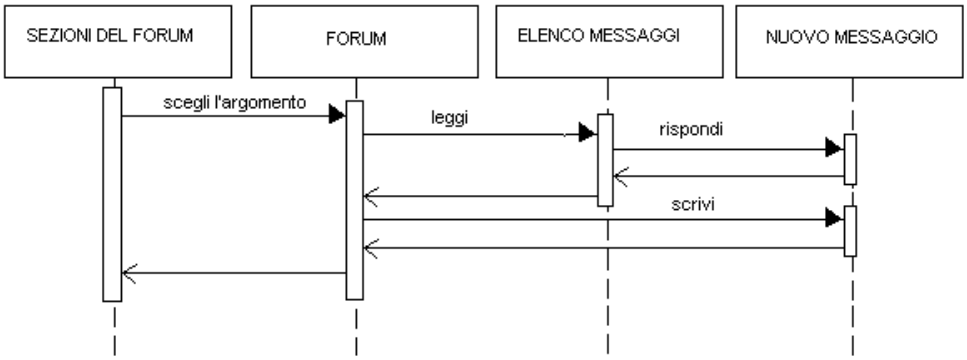


Figura 13: State diagram per l'utilizzo del forum.

# 3 Progettazione

Lo stadio di progettazione, diversamente da quello di analisi, deve tener conto di tutti i fattori relativi alla tecnologia che verrà utilizzata per la realizzazione del progetto. Alla fine di questo stadio si otterrà una descrizione tecnica esatta dell'architettura, del database, dell'interfaccia utente e delle strutture di dati che sono necessari per il raggiungimento dell'obiettivo prefissato.

Il sistema è stato progettato in modo da garantire la sicurezza, quindi con particolare attenzione al controllo dei diritti d'accesso degli utenti.

## 3.1 Progettazione delle interfacce utente

L'interfaccia utente è ogni parte del progetto a cui gli utenti possono accedere. Questo non riguarda solamente la parte grafica; anzi, bisogna stare attenti a non appesantire troppo le pagine da una grafica complessa e quindi lenta a caricarsi perché gli utenti che accedono al portale hanno connessioni intenet diverse e alcune potrebbero essere molto lente (a 56 Kbps o addirittura a 33.6 Kbps) e di fronte a tempi di attesa troppo lunghi per il caricamento delle pagine potrebbero stancarsi e abbandonare l'idea di visitare

il sito. La differenza tra prototipo d'analisi e interfaccia è che quest'ultima può essere utilizzata per l'implementazione.
Un aspetto invece fondamentale è la chiarezza: cosa s'intende con questa affermazione? Come dicevo poco fa, il pubblico al quale è rivolto il progetto è molto vario ed è quindi necessario creare pagine intuitive per agevolare il più possibile la navigazione nelle stesse. A tal scopo è bene utilizzare elementi standard nelle interfacce: in questo modo l'utente si trova di fronte ad elementi che già conosce e sa come affrontarli. Quando possibile, sarebbe bene mostrare agli utenti tutte le azioni e le alternative che può eseguire in modo da non condizionarlo nelle scelte da prendere. Dopo alcune scelte, come l'invio di messaggi o la cancellazione dal database di un nominativo, è bene inviare un feedback all'utente per informarlo dello stato reale del sistema. Per esempio, quando si inserisce un indirizzo email nella pagina di iscrizione alla newsletter viene stampato a video un messaggio che notifica se l'iscrizione è andata a buon fine o meno e, in quest'ultimo caso, ne spiega il motivo.

## 3.2 Gestione degli errori

Il sitema dev'essere in grado di gestire gli errori che si possono verificare. Per prima cosa è necessario individuare le tipiche cause di errore: interruzione da parte dell'utente di una determinata azione, campi obbligatori lasciati in bianco in fase di compilazione dei form, problemi di connessione al database, ecc. Nel progetto realizzato, quando un form non è compilato adeguatamente, viene segnalato un errore e si invita il navigatore a ricompilare i campi nel modo corretto, seguendo le istruzioni indicate nella pagina. Viene quindi mostrato un link, sia grafico che testuale, alla pagina precedente per agevolare l'utente e per indirizzarlo dove ritenuto necessario. Nel

caso di problemi di connessioni al database, e quindi estranei al progetto, viene semplicemente indicato un messaggio che rimanda l'utente all'ultima pagina visitata.

## 3.3 Database

La maggior parte delle pagine del sito realizzato si appoggiano ad una base di dati che viene utilizzata per diversi scopi all'interno di esso. Per interrogare il database relazionale ho utilizzato il linguaggio SQL (*Structured Query Language*), uno standard industriale molto diffuso. Le tabelle presenti sono nove; vediamole ora da vicino.

**anagrafica:** è composta di diciassette campi. Il primo, *id*, costituisce la *primary key*, cioè la chiave primaria che si autoincrementa ogni qualvolta viene effettuata una nuova iscrizione. Il campo *nomeUtente* è contrassegnato dalla parola *unique*, poichè è il nome da inserire nel form di login e per ovvi motivi non può essere identico per persone diverse. Il campo *progetto* può memorizzare alcune delle sigle corrispondenti ai progetti attivi. C'è poi *active* che contiene il valore "0" se l'utente non ha mai risposto all'email di conferma inviata al momento dell'iscrizione al sito. In caso di risposta invece assume il valore "1" ma ancora non è possibile accedere ai contenuti privati mediante login perché l'amministratore non ha ancora accettato la richiesta ricevuta. Quand'egli deciderà di acconsentire l'iscrizione, allora la cifra verrà incrementata ulteriormente assumendo il valore "2". Di default il campo active assume il valore "0". I restanti tredici campi sono dichiarati *varchar* di lunghezza variabile a seconda dell'utilizzo previsto. I campi obbliga-

tori sono sei, gli altri (ad eccezione di quelli con argomenti di default) possono non essere specificati.

**eventi:** la tabella eventi consiste solamente di quattro campi: l'identificatore che si autoincrementa e costituisce la chiave primaria, il titolo dell'evento da inserire, il corpo del messaggio e la data che, se non è esplicitamente specificata, prende in considerazione quella del giorno stesso in cui viene inserito il record nel database. Titolo e messaggio sono dichiarati *not null*, il messaggio è inoltre di tipo *text*.

**forum_principale:** è la prima delle tabelle dedicate al forum. In esso sono memorizzati proprio gli argomenti di discussione principali che non possono essere modificati da nessuno, nemmeno dall'amministratore. Uno di essi, come già detto in precedenza, è interamente dedicato alla formazione, gli altri sono di carattere più vario e generale.

**forum_thread:** otto sono i campi che compongono questa tabella. Come sempre c'è il campo *id* e in aggiunta a questo c'è ora il *topic_id* che si riferisce alla chiave primaria della tabella che vediamo qui di seguito. Tale riferimento è obbligatoriamente *not null*, ma questo non crea disagi ad alcuno poichè il campo viene compilato a livello di codice senza che l'utente se ne accorga. In questa tabella sono memorizzati i messaggi veri e propri del forum.

**forum_topics:** grazie a questa tabella gli utenti possono aprire, all'interno del forum, nuovi argomenti di discussione. Accanto al titolo del topic comparirà il nome di colui che ha inserito il primo messaggio e la data ad esso relativa. Questa tabella si collega direttamente alla chiave primaria della tabella *forum_id*.

**forum_topics_date:** si tratta di un piccolo magazzino ausiliario che per-

mette la ricerca, tramite il campo *data*, degli ultimi messaggi inseriti nel gruppo di discussione. Esso si riferisce sia alla tabella *forum_topics* sia a *forum_principale*.

**progetti:** la chiave primaria qui è costituita dalla coppia (*id*, *nomeProgetto*), per impedire che vengano inseriti due progetti con lo stesso nome. Si tratta chiaramente di una precauzione di efficienza piuttosto che di necessità in quanto sarebbe bastato dichiarare il solo campo *id* come chiave primaria. Al nome di ogni progetto è associata una sigla per agevolare le future operazioni effettuate sul database.

**responsabili:** si tratta di una tabella del tutto simile ad *anagrafica* in quanto contiene nominativi di persone che operano attivamente all'interno della cooperativa. La differenza è che, in questo caso, i dati non vengono inseriti dalla persona direttamente interessata ma dall'amministratore del sistema e alcuni di questi record verranno visualizzati nelle pagine pubbliche mediante una ricerca basata sul tipo di progetto assegnato a ciascuno di essi. Ci sono otto campi di cui quattro obbligatori, uno che si autoincrementa (*id*) e *email* che di default è inizializzato con una stringa vuota.

**rubrica:** si tratta di un'altra tabella dinamica che viene popolata dagli utenti stessi, volontari o utenti generici, che si iscrivono alla newsletter. I campi sono perciò solamente tre: la chiave primaria, l'indirizzo email e il valore active che viene impostato a 1 quando l'utente ha risposto all'email di conferma dell'iscrizione. Diversamente dalla tabella anagrafica, qui il moderatore non ha accesso alle email degli iscritti e non può cancellare alcun nominativo.

# 4 Implementazione

## 4.1 Linguaggi e strumenti utilizzati

Completate le fasi di analisi e di progettazione dell'applicazione web è giunto il momento di dare uno sguardo agli strumenti che hanno permesso la realizzazione dello stesso. Preso atto che l'ambiente lato server in cui PHP dà il meglio di sè è quello Unix-like, resta il fatto che Windows è il sistema operativo più utilizzato dall'utente comune; per questo motivo la mia scelta è ricaduta proprio su quest'ultimo.

Il linguaggio che fa da cornice al tutto è l'*HTML (Hypertext Markup Language)*: esso costituisce lo scheletro, la parte statica del sito che non permette interazioni dirette tra il navigatore e i contenuti presentati. L'HTML però è nato con ben poche ambizioni: il suo utilizzo era destinato solamente alla gestione di singole pagine; ben presto perciò ci si scontra con la difficoltà di posizionamento all'interno di esse di immagini e testo. Per far fronte al problema si può ricorrere ai *fogli di stile (CSS)*. Il foglio di stile che ho creato contiene la formattazione completa delle pagine, a partire dalla scelta del font da utilizzare nei diversi contesti (menu, intestazione, corpo del testo, ...), alla scelta del colore o della dimensione dello stesso, dal posizionamento delle immagini alla localizzazione dei diversi menu, ecc. Tutto questo in un solo foglio che viene successivamente incluso in tutte le pagine create. L'utilizzo di un css esterno risulta molto comodo soprattutto in fase di aggiornamento del portale in quanto per modificarne la struttura grafica è necessario apportare cambiamenti solamente in un file senza correre il rischio di dimenticarsi di aggiornare alcune pagine.

Per donare invece dinamicità al portale, in modo da renderlo vivo e coinvolgente e allo stesso tempo più utile al consumatore, mi sono servita del *PHP (Hypertext Preprocessor)*: buona parte del suo successo deriva dal fatto che

non solo si tratta di un linguaggio quasi perfettamente multipiattaforma, ma che è oltretutto possibile ricreare in Windows un ambiente di esecuzione molto simile a quello dove gli script, una volta testati in locale, andranno a girare dopo essere stati caricati sul server in hosting. Il webserver che ho utilizzato per eseguire gli script Php è *Apache (release 2.0.54)*. Alla base del tutto c'è ovviamente una base di dati senza la quale non si avrebbe la dinamicità richiesta al progetto; per interagire con essa ho installato la release 4.1 di *MySQL*, la versione più diffusa del celebre server di database. Il linguaggio utilizzato per interrogare la base di dati è *SQL (Structured Query Language)* che è di fatto lo standard tra i linguaggi per la gestione di data base relazionali. Oltre a quanto detto esso possiede i comandi per creare, modificare ed eliminare le tabelle dal database relazionale, cioè le funzioni di linguaggio *DDL (Data Definition Language)*, e per inserire, modificare ed eliminare le righe di una tabella, cioè le funzioni di linguaggio *DML (Data Manipulation Language)*. Ecco così costruita la tipica configurazione Apache+PHP+MySQL che a questo punto si può considerare multipiattaforma.

## 4.2 Implementazione delle interfacce utente

Uno dei compiti principali di questa fase è la realizzazione delle funzionalità che erano state descritte durante la progettazione. Vediamo ora alcune regole che ho cercato di rispettare per la buona riuscita del progetto. Prima di tutte vi è la *coerenza*: ad esempio, se per visualizzare un elenco in ordine alfabetico è necessario premere un determinato tasto, è bene che ogni qualvolta mi ritrovo di fronte ad un ordinamento alfabetico la procedura sia sempre la stessa in modo da non confondere l'utente. In secondo luogo vi è la denominazione: è importante che i bottoni che permettono l'interazione

dell'utente col sistema siano etichettati in modo chiaro ed intuitivo. Lo stesso vale per i messaggi d'errore stampati a video: pensiamo ad un utente che si trova nella pagina relativa al login. Egli inserisce le stringhe richieste e preme il testo *invio*: se i dati immessi corrispondono ad un preciso record nel database, allora viene effettuato l'accesso senza problemi; se uno dei due dati inseriti risulta essere errato, egli si ritrova di fronte al seguente messaggio: *Non sei registrato!*. Questo fa capire all'utente che per accedere alle pagine desiderate è necessario effettuare una registrazione. Se l'utente ha già eseguito la prima fase di iscrizione quello che leggerà a video sarà: *Devi completare la registrazione. Se non hai ancora ricevuto l'email per la conferma dell'iscrizione clicca qui.* Se invece i dati sono corretti, ma la fase di registrazione non è stata completata perché l'amministratore non ha ancora accettato la richiesta, il messaggio inviato è il seguente: *Devi attendere che l'amministratore accetti la tua richiesta di iscrizione.* Questo è un esempio di inequivocabilità dei messaggi inviati agli utenti, affinchè essi possano capire il perché degli errori verificatisi e, se possibile, rimediare agli stessi.

Parlando ancora di coerenza, questa volta a livello grafico, è importante utilizzare sempre gli stessi colori o gli stessi font per elementi con lo stesso significato. A tal fine le mie scelte sono state le seguenti: il classico colore blu corrisponde ai link da visitare, il grigio invece l'ho dedicato ai link presenti nel menu laterale, tutti gli altri contenuti son di colore rosso scuro.

Un'altra buona abitudine da seguire è quella di raggruppare sensatamente e ordinatamente gli elementi dell'interfaccia. Si pensi alla schermata di registrazione dell'utente: tutti i campi sono volutamente allineati, uno sotto l'altro, proprio per facilitarne la compilazione (vedi fig.5). Per ottenere questo effetto è possibile aiutarsi con le tabelle trasparenti che permettono l'ottenimento di un buon risultato. Anche per la visualizzazione dell'elenco dei

volontari e dei responsabili ho utilizzato le tabelle che risulteranno più o meno lunghe a seconda dei record trovati nel database.

## 4.3 Implementazione della base di dati

Per accedere ai record memorizzati nel database ho utilizzato il PHP combinato alla sintassi di SQL, mentre per la creazione delle tabelle ho utilizzato solamente SQL. Vediamo ad esempio la sintassi che genera la tabella *responsabili*:

```
CREATE TABLE responsabili (
  id int(11) not null auto_increment,
  sigla varchar(15) not null,
  zona varchar(200) not null,
  nome varchar(40) not null,
  cognome varchar(40) not null,
  tel1 varchar(20) default '',
  tel2 varchar(20) default '',
  email varchar(50) not null default '',
  primary key (id)
);
```

Commentiamo ora le righe di codice appena viste: il campo *id* è obbligatoriamente non nullo e si incrementa automaticamente a partire dal valore più alto trovato nella tabella. Esso è contrassegnato come *primary key*, cioè come chiave primaria. Tutti i campi dichiarati *varchar* indicano stringhe di tipo char di lunghezza variabile da 1 ad un massimo di N caratteri (dove N è il numero indicato fra parentesi tonde). Per far sì che un campo facoltativo sia inizializzato con una stringa vuota si inserisce nella sua dichiarazione il parametro *default ' '*. La definizione di una tabella è una procedura molto

semplice e la sintassi è rapida e intuitiva. La parte più complessa è invece la parte di progettazione, quindi le decisione dei campi da utilizzare, quali nomi adoperare per essi e soprattutto non creare campi ridondanti e non dimenticarsi invece di creare campi di importanza rilevante. Le tabelle non sempre sono collegate fra loro, così come quella appena esaminata è indipendente dalle altre presenti nel database. A volte però è necessario metterle in relazione fra loro in modo da poter memorizzare le informazioni più complesse in modo efficiente. Questo è quello che accade, ad esempio, nel forum. Vediamo perciò come viene implementato questo servizio.

```
CREATE TABLE forum_principale (
   id int(11) not null auto_increment,
   titolo varchar(255) not null unique,
   primary key (id)
);

CREATE TABLE forum_topics (
   id INT(11) not null auto_increment,
   forum_id integer not null references forum_principale(id),
   data date not null,
   ora time not null,
   autore varchar(255) not null ,
   titolo varchar(255) not null ,
   primary key (id)
);

CREATE TABLE forum_thread (
   id INT(11) not null auto_increment,
```

```
   topic_id int(11) not null references forum_topics(id),
   data date not null,
   ora time not null,
   autore varchar(255) not null,
   titolo varchar(255) not null,
   testo text not null,
   primary key (id)
);

CREATE TABLE forum_topics_date (
   topic_id integer not null references forum_topics(id),
   forum_id integer not null references forum_principale(id),
   data date not null,
   primary key (topic_id)
);
```

La novità introdotta in queste tabelle rispetto alla precedente è la seguente richiesta: *references nomeTabella(nomeCampo)*; questo attributo fa sì che la tabella che stiamo definendo e quella citata dopo la parola *references* siano messe in relazione fra loro. Il legame che le unisce è il campo indicato fra parentesi tonde; nel mio caso questo corrisponde sempre al campo id, ovvero alla chiave primaria. Non è però necessario ciò, perché potrebbe essere utilizzato un qualsiasi altro campo, a seconda della necessità da soddisfare.
Altri attributi che non abbiamo esaminato precedentemente sono *unique*, che impedisce di avere due o più record contenenti lo stesso valore in un determinato campo, nonostante questo non sia la primary key. Abbiamo poi il tipo *date* e il tipo *time*, utilizzati rispettivamente per memorizzare la data e l'orario. Infine c'è il tipo *text*, che memorizza sequenze di caratteri lunghe

(infatti non richiede il numero di caratteri massimo consentito).

## 4.4 Procedura di registrazione e invio di email da web

Vediamo ora come integrare i comandi sql e php. Consideriamo innanzitutto il codice utilizzato per connettersi alla base di dati.

```
<?PHP
$Host = 'localhost';
$User = 'nomeUser';
$Password = 'password';
$DBName = 'nomeDatabase';
$link = mysql_connect($Host,$User,$Password);
mysql_select_db($DBName); ?>
```

All'interno dei tag <?PHP e ?> inizializziamo le variabili *$Host, $User, $Password, $DBName* che ci servono per effettuare l'accesso. Il comando che stabilisce la connessione è *mysql_ connect*, il quale prende come argomenti i primi tre dei parametri appena citati. Il nome del database viene utilizzato nella riga immediatamente successiva per indicare quale database considerare fra l'elenco di quelli disponibili. Queste sei righe di codice vengono in un secondo momento incluse nei file che richiedono il recupero dei dati del database. Questo avviene tramite il comando:

```
include("connessione.php");
```

Analizziamo il caso della pagina destinata alla registrazione (vedi fig.5). Per prima cosa è necessario indicare con quali tabelle lavoreremo:

```
$TableName = 'anagrafica';
$TableProgetti = 'progetti';
```

Il simbolo *$* indica che stiamo lavorando con una variabile. Ora iniziamo con il form nel seguente modo:

```
<form
    action = "verificaRegistrazione.php"
    method = "post"
    name   = "Inserisci"
>
```

di seguito inseriamo i campi bianchi (nome, cognome, indirizzo, ...) che verranno compilati dall'utente. Al momento della scelta del progetto ho pensato di utilizzare un menu a tendina in modo da facilitare la compilazione del modulo. In questo caso si evita che vengano immessi nomi di progetti inesistenti e inoltre ad ognuno di essi è associata una sigla che collega la tabella dei progetti all'anagrafica. Il menu è realizzato in questo modo:

```
$query_prog = "SELECT * FROM $TableProgetti";
        $result_prog = mysql_db_query($DBName,$query_prog,$link)
                or die("Errore di connessione...");
        while($Row=mysql_fetch_array($result_prog)){
                $progetto = $Row["nomeProgetto"];
                echo "<option value='$progetto'>",
                      $Row["nomeProgetto"],"</option>";
            }
```

dentro la variabile *$query_prog* memorizzo la query che viene eseguita nella riga successiva. Il comando per fare ciò è *mysql_db_query* che prende come parametri il database sul quale eseguire l'interrogazione, il comando SQL da lanciare e il risultato della funzione mysql_connect(). Se tutto è andato bene si procede con il ciclo while, che memorizza nella *$Row* un array che

corrisponde alla riga caricata o false se non ci sono più righe. Ora resta solamente da stampare i risultati all'interno del menu; per farlo adopero il comando *echo* che mi permette di utilizzare i tag html anche se mi trovo nella parte di codice riservata al linguaggio PHP. Il tag che mi permette di aggiungere nuove voci al menu è il seguente: *<option value='$progetto'>...</option>*. Tutti gli altri campi presenti nel form sono di tipo text per poter contenere i dati che l'utente vorrà inserire.

```
<input type="submit" name="Inserisci" value="Inserisci">
<input type="reset"  name="cancella"  value="Cancella">
```

In fondo alla pagina ho inserito il bottone *reset* per cancellare il contenuto di tutti i campi nel caso l'utente voglia inserire altri dati e il bottone *invia* per inoltrare la richiesta d'iscrizione.
Vediamo ora cosa succede in *verificaRegistrazione.php* richiamata dalla pagina appena esaminata.

```
if(isset($_POST['Inserisci'])) {
$nomeUtente=$_POST['nomeUtenteNew'];
$password=$_POST['passwordNew'];
$nome=$_POST['nomeNew'];
$cognome=$_POST['cognomeNew'];
$indirizzo=$_POST['indirizzoNew'];
$cap=$_POST['capNew'];
$comune=$_POST['comuneNew'];
[...]
}
```

Per prima cosa devo verificare se il bottone *Invia* è stato premuto. In tal caso recupero i dati immessi con la sintassi *$_POST['passwordNew']*. Devo ripetere l'operazione per tutti i campi, anche se non sono stati riempiti

tutti. In secondo luogo devo eseguire alcuni controlli prima di poter inserire all'interno del database le informazioni: ad esempio devo testare che nessuno degli utenti già iscritti abbia scelto lo stesso *nomeUtente*. Qui di seguito ecco il codice per effettuare il controllo:

```
if($Row1[nomeUtente] == $nomeUtente) {
            echo "Nome utente già esistente.";
            [...]
            }
```

Alcuni dei campi a disposizione nel form erano contrassegnati da un asterisco poichè ritenuti obbligatori ai fini della registrazione; da qui la necessità di verificare che tutti siano stati effettivamente compilati. Tale controllo viene eseguito da un *if* che confronta le variabili in questione con la stringa vuota.

```
if(($nomeUtente=="")
   || ($password=="")
   || ($nome=="")
   || ($cognome=="")
   || ($email=="")){
       echo "Errore di compilazione. Ricorda che i campi
             contrassegnati da (*) sono obbligatori!";
   [...]
   }
```

Gli altri controlli riguardano la sintassi dell'indirizzo email, la lunghezza delle stringhe inserite, ecc.
Supponendo di aver superato positivamente tutti i precedenti test, ci apprestiamo ad inserire i dati nel nostro magazzino. Vediamo la sintassi della query per l'aggiunta di un nuovo record.

```
$inserisci=
  "INSERT INTO
      $TableName(id,nomeUtente,password,nome,cognome,[...])
   VALUES
      ('','$nomeUtente','$password','$nome','$cognome',[...]);";
$result=mysql_db_query($DBName,$inserisci,$link)
      or die("Errore di inserimento...");
echo "La registrazione è andata a buon fine!";
[...]
```

La query presentata è composta di due parti: nella prima indico la tabella da modificare (*$TableName*) seguita dai campi che di essa devo aggiornare; la seconda parte, che inizia con *VALUES*, prende i veri valori da inserire. Nella riga successiva eseguo l'interrogazione col comando già visto in precedenza e infine, se tutto è terminato con successo, stampo a video un messaggio per l'utente per comunicargli che la sua richiesta è stata eseguita correttamente. Terminata questa fase si passa alla validazione dell'indirizzo email inserito; viene perciò inviato un messaggio di posta elettronica all'indirizzo indicato. Al momento dell'iscrizione viene creato un campo denominato *$active* nel quale è memorizzato il valore 0; fino a che permane in questo stato l'utente non si può loggare. Il primo passo da fare per diventare un membro della sezione privata è cliccare sul link che viene inviato tramite posta elettronica: a questo punto il valore di *$active* verrà incrementato a 1. Questo però non è ancora sufficiente per essere un membro effettivo della Rete dei Volontari, perché l'ultimo anello della catena è l'approvazione da parte dell'amministratore che deve settare il campo *$active* al valore 2. Qui di seguito vediamo la porzione di codice utilizzata per l'invio dell'email:

```
$active = 1;
```

```
$subject = "Attiva la registrazione";
$message = "Clicca sotto per iscriverti".
"http://spaw.ce.unipr.it/progetti/SAP/attivaRegistrazione.php?
utente=$nomeUtente&active=$active";
$headers="From:LaReteDeiVolontari";
mail($email,$subject,$message,$headers);
session_start();
```

Nella variabile *$active* ho memorizzato il valore 1, nelle due righe immediatamente successive si trovano il titolo (*$subject*) e il corpo del messaggio con il link cliccabile; dentro a *$headers* c'è l'intestazione e infine con il comando *mail*, che prende come parametri l'indirizzo del destinatario, il titolo, il testo e il messaggio, si invia il messaggio.
Una volta cliccato sul link si aprirà sullo schermo dell'utente il contenuto della pagina *attivaRegistrazione.php* La parte di codice che analizziamo è quella riguardante la query per l'aggiornamento del database.

```
$nomeUtente=$_GET['utente'];
$active=$_GET['active'];
```

Recupero le variabili che mi servono per poter attivare l'utente iscritto, vale a dire la chiave primaria per trovare il record corrispondente nella tabella senza che vi siano ambiguità e il campo active, oggetto della modifica da apportare.

```
$query_aggiorna="UPDATE $TableName
                 SET active='$active'
                 WHERE nomeUtente='$nomeUtente'";
$result = mysql_db_query($DBName,$query_aggiorna,$link)
                       or die("Errore di inserimento...");
```

```
echo "La registrazione è stata completata con successo!";
```

La query contiene rispettivamente il nome della tabella da aggiornare (*UPDATE $TableName*), il nome del campo interessato (*SET active*) e il corrispondente valore (*'$active'*), l'indicazione di quale riga dev'essere manipolata (*WHERE nomeUtente = '$nomeUtente'*). La variabile *$result* contiene il comando di esecuzione dell'interrogazione e, se tutto è andato a buon fine, di seguito verrà stampato un messaggio di successo dell'operazione.

## 4.5 Gestione delle sessioni: procedura di login/logout

Vediamo ora come avviene la procedura di login e quindi come funziona la gestione delle sessioni con PHP.

```
<form action=controllaLogin.php method=post>
 Utente:   <input type="text"     size=15 name="nomeUtente">;
 Password: <input type="password" size=15 name="password">;
           <input type=submit value=" OK ">
 (Non hai ancora un nome utente e una password?
 <a href="registrati.php">Registrati</a> subito qui!)
</form>
```

Una volta ottenuto e attivato l'account, l'utente accede alla pagina *login.php* per poter entrare nella sezione privata. Egli inserisce il nome utente e la password negli appositi campi e preme il bottone *ok*. Il form ci rimanda alla pagina *controllaLogin.php* nella quale troviamo i seguenti controlli:

```
session_start();
```

Per dare inizio ad una sessione si usa *session_ start()* e questo dev'essere fatto prima di aver rilasciato codice php (è quindi sempre meglio inserire

questa riga di codice all'inizio della pagina). Quando diamo inizio ad una sessione il PHP crea automaticamente delle variabili e un cookie contenente delle informazioni. Se si vuole chiudere la sessione si può utilizzare il comando *session_ destroy()*; se invece non lo facciamo noi la sessione scade autonomamente alla chiusura del browser. Procediamo ora con l'analisi del nostro file. Andiamo avanti fino ad incontrare queste righe di codice:

```
require("connessione.php");
$query="select * from anagrafica";
$result=mysql_db_query($DBName,$query,$link) or die("Errore...");
```

Per accedere alla base di dati abbiamo bisogno di conoscere i parametri di connessione; per questo motivo dobbiamo sempre ricordarci di richiamare il file *connessione.php* che contiene queste informazioni. Questo viene realizzato con la sintassi appena illustrata. Eseguo ora una query che restituisce tutti i record contenuti in anagrafica, perché tra breve dovrò confrontare le stringhe inserite precedentemente nel form (*nomeUtente* e *password*) con i valori memorizzati nella tabella.

```
$Controllo = 0;
```

Creo una variabile denominata *$Controllo* che mi servirà per le gestione degli utenti e dei relativi permessi.

```
if ($_POST["nomeUtente"]=="admin" and
    $_POST["password"]=="admin"){
       $_SESSION['login'] = "1";
       $_SESSION['nomeUtente'] = $_POST["nomeUtente"];
       $Controllo = "2";
    }
```

Se i dati immessi nel form corrispondono a quelli dell'amministratore allora apro la sessione (con il comando *$_SESSION['login'] = 1;*) e imposto *$Controllo* al valore *2*.

```
while($Row = mysql_fetch_array($result))
```

Se il controllo precedente fallisce devo eseguire il confronto che dicevo prima, cioè devo verificare che i dati inseriti corrispondano ad un preciso record attivo del database.

```
{if ($Row[nomeUtente]==$_POST["nomeUtente"] and
    $Row[password]==$_POST["password"]     and
    $Row[active]==2){
        $_SESSION['login'] = "1";
        $_SESSION['nomeUtente'] = $_POST["nomeUtente"];
        $Controllo = "1";
    }
```

Se *$Controllo* viene impostata a *1* significa che l'utente è regolarmente iscritto al portale e quindi può accedere liberamente al menu privato. Nel caso in cui invece il campo *active* sia ancora impostato a *1* la variabile di controllo viene messa a *3*.

```
else if ($Row[nomeUtente]==$_POST["nomeUtente"] and
         $Row[password]==$_POST["password"]     and
         $Row[active]==1){
         $Controllo = "3";
    }
```

Infine, se il valore di active è quello di default, vale a dire *0*, il controllo assume il valore *4* e non è permesso loggarsi.

```
    else if ($Row[nomeUtente]==$_POST["nomeUtente"] and
              $Row[password]==$_POST["password"]     and
              $Row[active]==0){
        $Controllo = "4";
    }
} //chiude il ciclo WHILE
```

A questo punto del file il contenuto di *$Controllo* è stato definito, perciò lo salviamo in una variabile di sessione nel seguente modo:

```
$_SESSION['Controllo'] = $Controllo;
```

dove *$_SESSION* è un array superglobale che viene creato dal PHP quando si apre una sessione. A questo punto, a seconda del valore che assume la nostra variabile, il programma decide come comportarsi.

```
if ($_SESSION['login'] == "1")
```

La prima verifica riguarda la variabile *login*: se è impostata a *1*, allora posso procedere con gli ulteriori controlli, altrimenti viene stampato a video il messaggio *Non sei registrato!* che sollecita l'utente a registrarsi o a visitare la parte pubblica del sito. Supponiamo quindi di aver superato il primo *if* e procediamo con la lettura del codice.

```
{if ($Controllo == 2) {
    $provaNome=$Row[nomeUtente];
    echo "Benvenuto ",$_POST["nomeUtente"],"!";
    [...]
 }  //chiude IF controllo == 2

if ($Controllo == 1) {
```

```
    $provaNome=$Row[nomeUtente];
    echo "Ciao ",$_POST["nomeUtente"],"!";
    [...]
 }  //chiude IF controllo == 2

if ($Controllo == 0){
     echo "Nome utente o password errati!";
     $_SESSION['login'] = "0";
     [...]
 }
}
```

I tre blocchi di istruzioni qui sopra rappresentano tre differenti casi: nel primo viene dato un messaggio di benvenuto all'amministratore, nel secondo il messaggio viene mandato al volontario e infine viene segnalato il verificarsi di un errore nell'inserimento del nome utente o della password. Se invece la variabile di sessione *$login* non è uguale a *1*, allora l'accesso non può essere effettuato e questo può accadere in due diverse situazioni: nel primo caso si tratta di un account non attivo perché in attesa dell'approvazione dell'administrator, nel secondo siamo invece nel caso di una registrazione incompleta. Qui di seguito ne vediamo il codice.

```
else if ($Controllo == 3){
    $nomeUtente=$nomeUtente;
    echo "Devi attendere che l'amministratore accetti
          la tua richiesta di iscrizione.";
    $_SESSION['login'] = "0";
    [...]
} //chiude IF controllo == 3
```

```
else if ($Controllo == 4){
    $nomeUtente=$nomeUtente;
    echo "Devi completare la registrazione.";
    [...]
    $_SESSION['login'] = "0";
} //chiude IF controllo == 4
```

A questo punto ci siamo loggati e alla sinistra dello schermo possiamo consultare il menu aggiuntivo del volontario o dell'amministratore. La sessione rimane attiva finchè l'utente non decide di eseguire il logout, oppure finchè non chiude il browser. Durante questo periodo il navigatore può consultare liberamente tutte le pagine, sia quelle private che quelle pubbliche.
Vediamo ora come avviene la fase di logout.

```
 echo "Sei sicuro di voler uscire?";
 echo "<a href=\"benvenuto.php\">NO</a>";
 echo "<a href=\"indexLogout.php\" target=\"_top\">SI</a>";
```

Per sicurezza chiediamo una conferma prima di distruggere la sessione. Se l'utente ci ripensa lo ridirezioniamo alla pagina di benvenuto della sezione privata; in caso contrario carichiamo la pagina iniziale del portale. In realtà non carichiamo la *index.php* ma una copia, chiamata appositamente *indexLogout.php*, nella quale ho aggiunto le seguenti righe di codice:

```
$_SESSION['login'] = "0";
$Controllo=0;
session_destroy();
```

La prima istruzione serve per impostare la variabile indicata fra parentesi quadre a 0. Anche la variabile di controllo viene azzerata e infine distruggiamo la sessione. Da questo momento in poi la sessione precedentemente aperta non è più in vita e quindi tutte le pagine private non saranno più visitabili, lo stesso vale per il menu secondario che non sarà più visibile fino al prossimo login.

Abbiamo dato uno sguardo più approfondito a come avviene l'interazione tra PHP e MySQL e quale sintassi è necessario usare, non andiamo ad analizzare il resto del sito perché la trattazione diventerebbe più simile ad un corso di PHP che altro, ma abbiamo reso l'idea di come funzioni questa collaborazione di linguaggi.

# 5 Test

La fase di test serve da verifica finale o parziale per controllare che i requisiti fissati in fase di analisi siano stati raggiunti. Generalmente la stategia di test è decisa dal tester rispetto ad un prefissato obiettivo di qualità, ma ciò non toglie che anche i clienti vi partecipino chiedendo che vengano effettuati controlli riguardo alle priorità di qualità del software.

## 5.1 Metodi di test

La fase di test è composta da diversi passaggi, in quanto un sistema che viene testato secondo vari metodi è considerato più sicuro poichè si riducono notevolmente le possibilità di errore. Vediamo ora alcuni di questi metodi. Primo di tutti è il metodo BLACK-BOX, che esamina dall'esterno l'elemento ignorando completamente i dettagli tecnici interni. Nel caso di errori si può solamente stabilire dove esso si è verificato. Un secondo test riguarda il funzionamento del database, in altre parole il corretto trasferimento dei dati da e verso di esso e la verifica di coerenza e correttezza degli stessi. Infine è necessario testare il corretto funzionamento del sito su diversi tipi di browser.

## 5.2 Test black-box

Procediamo subito con un primo test completo delle aree di input relative alla registrazione di un utente mediante il form dell'omonima pagina. Suddividiamo i dati in classi di equivalenza in base alle quali verrà eseguito il test.

| N. | Descrizione del caso di test | Risultato previsto | Dati d'input dimostrativi |
|---|---|---|---|
| 1 | Input del nome utente | Risultato della ricerca | nomeUtente='Anna' |
| 2 | Nome utente vuoto | Messaggio d'errore | nomeUtente='null' |
| 3 | Nome utente non valido | Messaggio d'errore | nomeUtente='Anna' |
| 4 | Input della password | Risultato della ricerca | password='1234' |
| 5 | Password vuota | Messaggio d'errore | password='null' |
| 6 | Input del nome | Risultato della ricerca | nome='Anna' |
| 7 | Nome vuoto | Messaggio d'errore | nome='null' |
| 8 | Input del cognome | Risultato della ricerca | nome='Celada' |
| 9 | Cognome vuoto | Messaggio d'errore | nome='null' |

## 5.3 Test sul database

Può capitare di incontrare dei problemi dovuti alla presenza di dati errati nel database. Come conseguenza di questi errori potrebbe venire danneggiata anche la parte di database che in precedenza era integra. E' consigliabile perciò verificare la semantica e la sintassi per ogni campo di ciascuna tabella, esaminare i collegamenti e le dipendenze tra le tabelle quando si ha a che fare con riferimenti alle chiavi. Tutti questi controlli sono da eseguire periodica-

mente durante la fase d'implementazione. Per quanto concerne la sezione dedicata alla registrazione degli utenti ho posto dei controlli nel momento in cui vengono compilati alcuni campi del form.
Il primo di questi riguarda la presenza di informazioni all'interno dei campi obbligatori. In caso di mancata compilazione, l'utente viene avvisato tramite un messaggio d'errore che lo sollecita a non lasciare in bianco gli spazi non facoltativi. Superata questa prima fase vi è un controllo sulla correttezza dei dati immessi. Ad esempio l'indirizzo email deve contenere il carattere speciale `@` per essere considerato valido. Un'altra verifica concerne il nome utente: il navigatore deve scegliere un nickname non ancora presente nel database. In caso di nominativo già in uso viene mostrato un messaggio che rimanda l'utente alla pagina di iscrizione e lo invita a cambiare nome.
Questi sono alcuni dei test che vengono effettuati sul database per garentirne l'integrità e permettere un corretto funzionamento del sistema.

## 5.4 Test sui browser

La compatibilità del sito è stata testata con seguenti i browser: Mozilla Firefox 1.5.0.3, Netscape 8.1, Opera 8.54, Internet Explorer 6.0.

# 6 Conclusioni e lavori futuri

Il progetto realizzato verrà utilizzato dalla cooperativa Il Piccolo Principe e sarà amministrato da un responsabile della cooperativa stessa. Egli verrà fornito della password per accedere alla sezione privata tramite la quale potrà mantenere il portale costantemente aggiornato. La password potrà essere sostituita immediatamente del responsabile stesso e cambiata ogniqualvolta lo ritenga opportuno. Il sito è stato progettato in modo da poter essere ampliato e/o modificato a seconda delle esigenze future della cooperativa.

# 7 Ringraziamenti

Ringrazio i responsabili della Cooperativa Il Piccolo Principe per avermi permesso di lavorare con loro, in particolare Alessandro Patroncini che ha seguito con pazienza e attenzione lo svilupparsi del progetto e ha fornito tutto il materiale da inserire nelle pagine web. Ringrazio inoltre Patrizia Melioli e il responsabile della formazione dei volontari, Fernando Angelucci. Un doveroso ringraziamento va al mio relatore, il prof. Eduardo Calabrese che ha seguito e coordinato il lavoro durante questi mesi. E' doveroso ora ringraziare la mia famiglia, in particolare i miei genitori e mia sorella Chiara, che mi hanno spronata a tener duro nei momenti di debolezza; senza i loro consigli e le loro parole di incoraggiamento non sarei arrivata a questo traguardo della mia vita. Insieme a loro ringrazio Delia e Nadia, le mie compagne di studi preferite con le quali ho condiviso le fatiche e le soddisfazioni del percorso universitario e con loro ringrazio Samuele e gli altri compagni di corso.
A questo punto ringrazio le mie sorelline acquisite Lucia e Cristina che mi hanno e mi continuano a sopportare giorno dopo giorno senza le quali mancherebbe anche una parte di me! Ringrazio la neo-dottoressa Stefy e la fisioterapista Ile che di ansia da esame ne sanno qualcosa... Ringrazio Alle, Lollo e Ricky un trio pericoloso...senza il quale però non saprei stare!! Fabio che è una persona davvero speciale, grazie delle chiacchierate-fiume e di tutti i consigli che mi dai sempre! Giovanni che spero mi porterà presto a sciare...dato che da ottobre non avrò più da studiare tutti i week-end! Gloria, Muz, Barbara, Nick, Piccio, Max, Paolo, Renzo, Alan, Simone, Flavio e tutta la comitiva! Ringrazio i Flipiti Flops, i Buddi al Bar e gli amici dei Buddi, in particolare Enrico, Ross e Anna...ovvero i miei colleghi cantanti e musicisti con i quali trascorro tante serate di prove e concerti. Insieme a loro ringrazio Cristina, Navid e Stefania, le mie super insegnanti di canto!

Potrei andare avanti con tanti altri ringraziamenti ma mi dilungherei troppo. Ringrazio quindi tutte le persone (e sono davvero tante!), che mi hanno dimostrato sostegno e affetto nell'affrontare questi anni di università!

# A Appendice

## UML: Unified Modelling Language

L´Unified Modelling Language è un linguaggio standard per specificare, visualizzare, costruire e documentare sistemi software e che usa prevalentemente notazioni grafiche per esprimere il disegno dei progetti software. Grazie agli aiuti UML i gruppi di progetto comunicano, esplorano disegni potenziali e convalidano l´architettura del software. Ogni diagramma è progettato per consentire di far osservare agli sviluppatori e ai clienti il sistema software da una diversa prospettiva e da differenti gradi di astrazione.
Per quanto concerne l'efficienza è necessario capire bene le richieste del cliente e fare in modo che esse vengano considerate come requisiti indispensabili del lavoro da svolgere. Successivamente, si useranno tali requisiti per generare il codice necessario alla costruzione del sistema assicurandosi che non si perda di vista l'obiettivo finale. Tale procedimento prende il nome di modellazione (*Modeling*). Il Visual Modeling è il processo che prevede la visualizzazione grafica di un modello, utilizzando un insieme ben definito di elementi grafici, che nel linguaggio UML sono rappresentati dai 9 diagrammi di base. Questo ci fa capire che l'UML si avvale principalmente di diagrammi; quelli che ho utilizzato nello sviluppare il progetto sono lo usecase e il sequence diagram. Nel primo tipo vengono utilizzati i casi d'uso: un *caso d'uso* è un'interazione fra il sistema e l'utente, quindi in esso dovranno essere rappresentate tutte le funzionalità esterne del sistema software. All'interno di un singolo caso d'uso vi possono essere più percorsi possibili. Tali cammini prendono il nome di *scenari*; quindi per meglio descrivere il nostro usecase possiamo realizzare diversi scenari, i quali devono interessare i casi più significativi per meglio descrivere le diverse interazioni che vi possono essere fra più casi d'uso.
Questi scenari possono essere descritti in maniera grafica dal *sequence dia-*

*gram.*

Il vantaggio offerto da UML è che la comunicazione e l'interazione tra tutte le risorse umane che prendono parte allo sviluppo del sistema sono molto più efficienti e dirette. Parlare lo stesso linguaggio aiuta ad evitare rischi di incomprensioni e quindi sprechi di tempo. Inoltre il progetto risulterà documentato nei minimi dettagli ancor prima che ne venga scritto il relativo codice da parte degli sviluppatori e questo comporta un'implementazione più agevole ed efficiente oltre al fatto che in tal modo è più facile scrivere del codice riutilizzabile in futuro. I costi di sviluppo, dunque, si abbassano notevolmente con l'utilizzo di UML. Infine, ma non ultimo in ordine di importanza, grazie a una precisa documentazione diviene ancora più semplice effettuare eventuali modifiche future a beneficio dei tempi e costi di mantenimento del sistema.

## Riferimenti bibliografici

[1] Wolfgang Zuser, Stefan Biffl, Thomas Grechenig, Monika Kohle
*Ingegneria del software con UML e Unified Process*
McGraw Hill, 2004.

[2] Luca Cattaneo
*HTML e CSS*
McGraw Hill, 2003.

[3] Larry Ullman
*PHP per il World Wide Web*
Pearson Education Italia, 2001.

[4] Stig Bakken, Andi Gutmans, Derick Rethans
*PHP 5*
Apogeo, 2005.

[5] Tobias Oetiker, Hubert Partl, Irene Hyna e Elisabeth Schlegl
*Una (mica tanto) breve introduzione a LATEX2e*
GNU General Public License, 2000.

[6] www.php.net