Progettazione di un'applicazione web CGI

Marco Liverani *

Sommario

In queste pagine proponiamo la descrizione di un semplicissimo progetto per la realizzazione di un'applicazione web CGI. Lo scopo del progetto è puramente didattico e ci aiuterà a mettere a fuoco i diversi elementi e le differenti tecnologie utilizzate per la realizzazione dell'applicazione.

1 Introduzione

Ci proponiamo di descrivere il progetto per la realizzazione di una web application CGI per la gestione di una semplice rubrica di indirizzi e-mail e di numeri telefonici, on-line.

La difficoltà nella realizzazione di un progetto di questo tipo, non è nella progettazione o nello sviluppo del software per la realizzazione delle singole funzioni dell'applicazione, quanto piuttosto nella molteplicità di tecnologie e linguaggi che occorre mettere in campo per realizzare il progetto.

La tecnologia web è per sua natura eterogenea e questo si riflette anche nell'architettura delle web application: affrontando la progettazione e lo sviluppo di un programma (pensiamo ad un programma per l'esecuzione di un calcolo complesso mediante un algoritmo che usa concetti matematici complessi) spesso non è l'*architettura* del software il primo aspetto su cui focalizzare l'attenzione; piuttosto è la complessità computazionale dell'algoritmo, la progettazione delle strutture dati più adatte alla rappresentazione efficienti di grandi moli di dati o gli aspetti più tecnici di programmazione vera e propria ad essere al centro dell'attenzione del progettista e del programmatore; al contrario, nella realizzazione di una *web application* è l'aspetto architetturale del software il primo punto su cui focalizzare l'attenzione. Sebbene l'esempio sviluppato nelle pagine seguenti sia necessariamente molto semplice, cercheremo di mettere a fuoco anche l'aspetto architetturale.

2 Requisiti funzionali e sui dati

Lo scopo del nostro progetto è quello di realizzare un sistema per la gestione e la consultazione di una rubrica di indirizzi (nomi, telefoni, indirizzi e-mail, ecc.). Il sistema deve garantire la possibilità di:

- caricare in archivio nuovi record, consentendo all'utente di inserire i vari attributi attraverso una form;
- eseguire delle ricerche sull'archivio, specificando anche solo alcuni dei valori degli attributi (o valori parziali), per selezionare i record di interesse;
- eliminare dall'archivio singoli record.

Con la funzione di ricerca deve anche essere possibile visualizzare l'intero archivio degli indirizzi, senza alcuna limitazione nel numero di elementi da visualizzare.

* E-mail: liverani@mat.uniroma3.it. Ultima modifica del 11 maggio 2020.

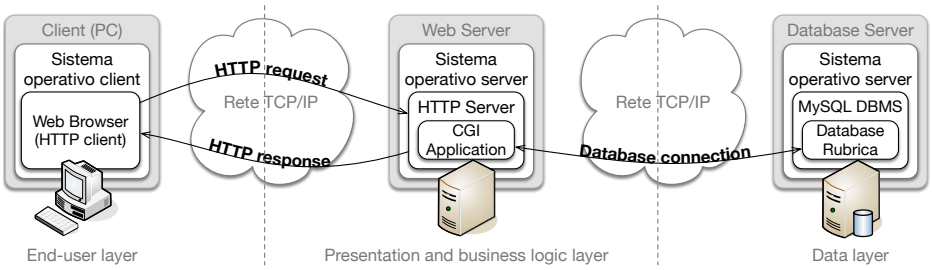

Fig. 1: Schematizzazione dell'architettura del sistema

Ogni elemento presente in archivio dovrà essere caratterizzato dai seguenti attributi; il sistema deve consentire l'inserimento di record anche privi di tutti gli attributi di seguito elencati:

- nome: il nome proprio della persona registrata sulla rubrica;
- cognome: il cognome della persona registrata sulla rubrica;
- e-mail: l'indirizzo di posta elettronica della persona;
- telefono: numero di telefono;
- data di nascita: la data di nascita della persona.

3 Architettura del sistema

L'architettura informatica del sistema è quella tipica di un'applicazione *web based* a due livelli: presentation layer e data layer; nel caso in esame non sembra utile prevedere un layer che implementa la logica di business, distinto dalla logica di presentazione, dal momento che le funzionalità sono estremamente semplici e non sono evidenziate specifiche esigenze di calcolo o di elaborazione.

Come evidenziato in Figura 1, l'architettura del sistema è fondata su tre ambienti distinti (includendo anche l'ambiente client), collegati ed integrati fra di loro attraverso una rete TCP/IP. La componente client comunicherà esclusivamente con la componente Web Server: in questo caso la comunicazione avviene mediante il protocollo standard HTTP (*HyperText Transfer Protocol*). La componente Web Server a sua volta comunica, oltre che con il client, anche con la componente Database Server, attraverso un protocollo di accesso al DBMS (*DataBase Management System*) implementato dal prodotto software MySQL; in questo caso viene usato un protocollo proprietario del prodotto MySQL.

3.1 HTTP client

Lo strato dedicato all'utente finale è composto da un computer client (un personal computer, un notebook, un tablet, uno smartphone, ecc.) dotato di un suo sistema operativo mediante il quale viene gestita la connessione di rete basata sul protocollo TCP/IP. Il sistema operativo utilizzato sulla postazione client è del tutto trascurabile, ai fini del corretto funzionamento del sistema applicativo in oggetto.

L'unico requisito effettivo della componente client è costituito dalla presenza di un web browser, ossia di una applicazione, compatibile con il sistema operativo della postazione client, in grado di svolgere il ruolo di client nell'ambito di una sessione di comunicazione basata sul protocollo HTTP. In altre parole, sulla postazione client deve essere

Fig. 2: La diversa visualizzazione di una form di inserimento dati con il browser Google Chrome su un personal computer (in alto) e con il browser Safari su un dispositivo Apple iOS (in basso)

installato un web browser, come Firefox, Google Chrome, Microsoft Internet Explorer o Edge, Apple Safari, ecc.

Il web browser ha il compito di gestire la sessione di dialogo con il server mediante il protocollo HTTP e di effettuare il *rendering*, la corretta presentazione sul dispositivo di output della postazione di lavoro dell'utente (tipicamente lo schermo del computer), delle informazioni prodotte in output dal server HTTP sulla base delle richieste del client. Il browser gestisce anche l'interazione dell'utente con le "schermate" della web application, presentando i *widget* delle form di inserimento dati (bottoni, menù a tendina, campi di testo o di selezione, ecc.) secondo le modalità grafico-funzionali previste dall'interfaccia utente del sistema operativo della postazione di lavoro.

Inevitabilmente, quindi, su differenti tipi di *device*, con differenti sistemi operativi e web browser, le pagine HTML appariranno con alcune differenze tipiche del sistema su cui vengono visualizzate. Questi aspetti, come vedremo più avanti, possono essere trascurati da chi progetta e sviluppa la web application, demandando la problematica della corretta resa dell'output prodotto dal programma, direttamente al web browser.

3.2 HTTP server

Lo strato di *presentation* e di *business logic* è la componente applicativa vera e propria che, come si intuisce dalla denominazione, si occupa della presentazione dei dati e delle funzioni dell'applicazione all'utente e, al tempo stesso, dell'implementazione delle funzioni "di business" che realizzano lo scopo del sistema stesso (acquisire i dati e registrarli in archivio, selezionare dei dati dall'archivio e presentarli all'utente, ecc.).

Trattandosi di una web application (e non, ad esempio, di un'applicazione client/-server) questo strato del sistema è basato su una componente software fondamentale: il server HTTP. È l'applicazione che fa da controparte al web browser (HTTP client) installato sulla postazione di lavoro dell'utente ed implementa il protocollo HTTP (vedi Figura 3). È un'applicazione attiva in attesa di connessioni sulla porta TCP definita nella configurazione dell'applicazione stessa (tipicamente la porta TCP 80). Questa componente software non è oggetto dello sviluppo nell'ambito di questo progetto, dal momento che esistono numerosi prodotti software open source o di mercato, che implementano correttamente questo servizio. Ad esempio il prodotto open source Apache HTTP Server o il prodotto di mercato Microsoft Internet Information Server.

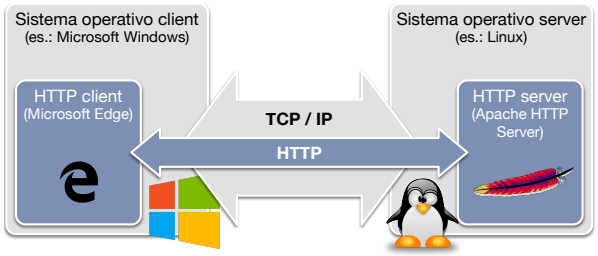

Fig. 3: Comunicazione via HTTP del web browser con il web server, attraverso la connessione TCP/IP tra il client e il server

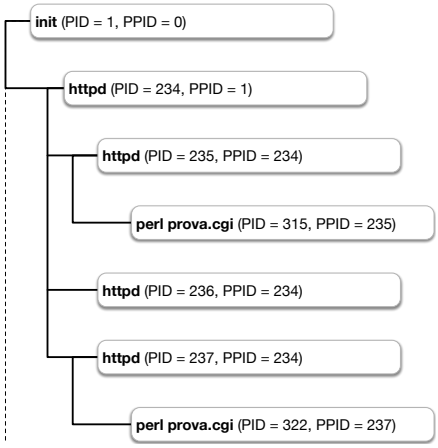

Fig. 4: Gerarchia dei processi che vengono eseguiti sul server web

Il server HTTP riceve le richieste HTTP dal client e costruisce la risposta sulla base della URL con cui viene identificata la risorsa richiesta; questa può essere una risorsa *statica*, ossia una pagina HTML (*HyperText Markup Language*), un'immagine grafica (file in formato GIF, JPEG, PNG, ecc.) o comunque un file presente sul server, ovvero può essere una risorsa *dinamica*, ossia un contenuto prodotto eseguendo un programma sul server. Nel nostro caso i contenuti dinamici sono l'output prodotto in formato HTML da un insieme di programmi CGI (*Common Gateway Interface*), eseguiti dal server HTTP. I programmi CGI sono realizzati come script in linguaggio Perl, ma la specifica CGI è una specifica aperta che può essere implementata indipendentemente dal server HTTP e dal linguaggio utilizzato per sviluppare i programmi CGI.

Come sappiamo i programmi CGI sono seguiti sul server come processi figli del processo HTTP che sta gestendo la sessione di comunicazione con il client HTTP. Se avvengono più connessioni contemporanee al server da parte dei client, saranno eseguiti tanti processi quanti sono le risorse dinamiche corrispondenti a programmi CGI.

In Figura 4 è schematizzata la gerarchia dei processi attivi sul sistema server a fronte di connessioni contemporanee da parte di più client. Nell'esempio in figura il server HTTP è realizzato tramite il programma **httpd**, che viene eseguito come figlio del processo **init**. Il processo **httpd** produce tre processi figli tramite la funzione "`fork`" per gestire più connessioni contemporanee con i client (ogni processo può gestire una sola connessione per volta). I processi **httpd** "figli", identificati dal PID 235 e 237, stanno eseguendo due processi distinti (PID 315 e 322) corrispondenti allo stesso programma: l'interprete del linguaggio Perl esegue lo script **prova.cgi**.

Il server HTTP riceve dal client le richieste per le risorse "dinamiche" (gli script CGI)

insieme ai dati forniti in input. I dati in input vengono inviati tramite i comandi GET o POST del protocollo HTTP dal client al server in formato *URL encoded*. Questo formato prevede che ciascun dato in input sia fornito come coppia di elementi: un identificativo del nome dell'attributo fornito in input e il valore dell'attributo stesso. Le coppie nome/-valore sono formate da una stringa di caratteri in cui il nome dell'attributo è separato dal suo valore dal carattere "="; ad esempio: "`nome=Marco`". L'insieme delle coppie chiave/-valore che costituiscono l'intero insieme dei dati in input, sono concatenate l'una di seguito all'altra in un'unica stringa di caratteri; ciascuna coppia è separata dalla successiva dal carattere "`&`"; ad esempio: "`nome=Marco&cognome=Liverani&citta=Roma`". Se la stringa dei valori forniti in input al server HTTP dal client, contiene caratteri particolari come i caratteri "=" o "`&`" o, in generale, caratteri il cui codice ASCII è superiore a 127, come ad esempio le lettere accentate, questi caratteri saranno codificati con una stringa composta dal simbolo "`%`" seguito dalle due cifre del codice ASCII del carattere in formato esadecimale; ad esempio la lettera "`à`" viene codificata con la stringa "`%E0`", visto che il codice ASCII esteso del carattere è 224, corrispondente a E0 in base 16.

Il client può eseguire la richiesta HTTP con il comando GET o con il comando POST. Nel primo caso la stringa dei parametri che il client fornisce in input al programma CGI richiesto al server, viene accodata alla URL della richiesta, separandola dal nome del programma CGI con il carattere "`?`". Ad esempio se si richiama il programma CGI "`search.cgi`" passandogli in input la stringa di parametri "`nome=Marco&citta=Roma`", la richiesta HTTP del client è la seguente:

```
GET search.cgi?nome=Marco&citta=Roma
```

Viceversa, se la richiesta HTTP avviene con il comando POST, la stringa con i parametri da passare in input al programma CGI non è presente nella URL della richiesta e viene invece inserita tra gli altri parametri che sono parte integrante della richiesta HTTP, pur non essendo riportati esplicitamente nella URL.

Il server HTTP, secondo le specifiche dell'interfaccia applicativa CGI, rende disponibile la stringa dei parametri in input al programma CGI in due diverse modalità a seconda del tipo di comando utilizzato per richiamare il programma CGI stesso (GET o POST). Il server HTTP prima di eseguire il programma CGI imposta una variabile d'ambiente del sistema operativo, nella stessa "shell" in cui viene eseguito il programma CGI; la variabile è denominata `REQUEST_METHOD` e contiene come valore la stringa "`GET`" o la stringa "`POST`", a seconda del comando HTTP utilizzato per richiamare il programma CGI.

Se il comando utilizzato è "`GET`", allora il server HTTP imposta una seconda variabile d'ambiente nella shell in cui verrà eseguito il programma CGI; la variabile `QUERY_STRING` contiene la stringa dei parametri passati in input al programma CGI, in formato *URL encoded*. Se invece il comando utilizzato per invocare l'applicazione CGI è "`POST`", allora il server HTTP fornirà la stringa dei parametri in input, nel formato *URL encoded* sul canale di input standard, in modo che il programma CGI possa acquisire la stringa dei parametri effettuando una lettura in input di una stringa su tale canale.

Dopo aver eseguito il programma, il server HTTP, secondo la specifica CGI, si occupa di catturare l'output prodotto dal programma stesso e di inviarlo come risposta HTTP al client; al termine dell'invio della risposta, il server chiude la connessione con il client HTTP. Il protocollo HTTP è infatti un protocollo *stateless*, in cui la sessione di comunicazione tra il client e il server, ha una durata brevissima, sufficiente ad acquisire una richiesta del client da parte del server e inviare la risposta corrispondente del server al client; quindi la connessione viene chiusa.

Il server HTTP prima di inviare al client l'output acquisito dal programma CGI, verifica che sia chiaramente indicato il tipo MIME (*Multipurpose Internet Mail Exchange*) dello stream di dati prodotti dal programma CGI. Tipicamente il programma CGI produrrà un output in formato HTML e quindi il tipo MIME indicato nell'intestazione

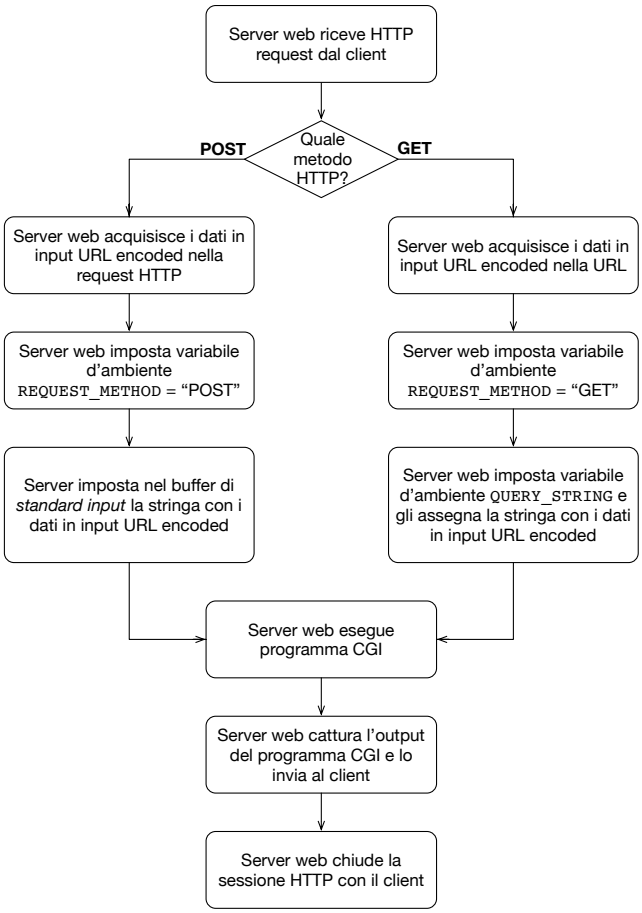

Fig. 5: Schematizzazione con un diagramma di flusso del processo di elaborazione di una richiesta per una risorsa "dinamica" da parte del server HTTP

dell'output prodotto dal programma CGI sarà "`text/html`". In questo caso, in testa allo stream di dati prodotti dal programma CGI, sarà inserita la seguente dichiarazione, prodotta dallo stesso programma CGI:

```
Content-type: text/html
```

In assenza dell'indicazione del tipo MIME da parte del programma CGI, come prima informazione prodotta in output, il server HTTP produce un messaggio di errore interno («*Internal Server Error*»)e chiude comunque la sessione HTTP con il client.

3.3 Database server

Per garantire la "persistenza" dei dati, la web application del nostro progetto utilizza il servizio di DBMS offerto dal prodotto MySQL. Questo servizio viene acceduta dall'applicazione CGI attraverso un protocollo applicativo proprietario che a sua volta sfrutta la connessione di rete basata sul protocollo TCP/IP tra il web server e il database server. In questo modo è possibile realizzare il servizio HTTP server e Database server su due macchine fisicamente distinte e collegate in rete fra di loro, ovvero su una stessa macchina che implementa entrambi i servizi: in questo caso la connessione tra l'applicazione CGI eseguita dal web server e il database MySQL eseguito dal database server, può avvenire utilizzando l'indirizzo di *loopback* 127.0.0.1 (corrispondente al nome host

"localhost"). Nella nostra web application di esempio è proprio questa la configurazione adottata: il server HTTP Apache e il server database MySQL vengono eseguiti sulla stessa macchina server e quindi comunicano fra loro utilizzando l'indirizzo di loopback.

Sul database server MySQL deve essere definito un database e un'utenza autorizzata ad effettuare la connessione al database e ad eseguire le operazioni necessarie sui dati per realizzare correttamente le funzioni offerte dalla web application.

In particolare quindi viene definito sul sistema MySQL il database "rubricaDB" utilizzando il seguente comando SQL nella shell dei comandi di MySQL, con l'utenza di database administrator (admin):

```
mysql> create database rubricaDB;
```

Viene inoltre definito un utente applicativo denominato "rubricaUser" utilizzato dal programma CGI per collegarsi al DBMS ed eseguire delle operazioni sulle tabelle del database rubricaDB; la password assegnata all'utente è la stringa "xy-99", che in un'applicazione reale dovrà essere definita con criteri di maggiore complessità (una password più lunga e con caratteri maiuscoli, minuscoli, simboli di interpunzione e cifre numeriche):

```
mysql> create user 'rubricaUser'@'localhost' identified by 'xy-99';
```

L'utente così definito potrà effettuare la connessione al database server soltanto da localhost: in altri termini non sarà possibile effettuare una connessione al database con questa utenza da un altro host; questo particolare consente di rafforzare la sicurezza del sistema e in particolare della componente database.

Per garantire all'utente rubricaUser i necessari permessi di lettura e scrittura sulle tabelle che saranno definite nel database rubricaDB è necessario eseguire il seguente comando:

```
mysql> grant all on rubricaDB.* to 'rubricaUser'@'localhost';
```

4 Specifiche funzionali e sui dati

Riprendendo i requisiti funzionali e sui dati, presentati nella Sezione 2, possiamo precisare meglio le specifiche funzionali da cui segue il progetto di realizzazione del sistema per la gestione della rubrica degli indirizzi via web. Il sistema accoglie l'utente con un menù principale che presenta due sole scelte:

1. inserimento di un nuovo record in archivio;
2. selezione dei record presenti in archivio, sulla base di specifici criteri di selezione.

La prima opzione viene realizzata proponendo una form di inserimento dati, con cui l'utente potrà inserire tutti i dati che desidera, relativi alla nuova voce da inserire nell'archivio della rubrica degli indirizzi. Nessun attributo è obbligatorio. Gli attributi sono quelli già descritti nella Sezione 2: nome, cognome, indirizzo e-mail, numero di telefono e data di nascita. La form di inserimento dati può essere realizzata come una semplice pagina HTML "statica", dal momento che non c'è nessuna elaborazione da eseguire per presentare la maschera di inserimento.

Una volta completato l'inserimento dei dati nella form, selezionando un bottone, l'utente può indicare al web browser che l'inserimento è terminato e che quindi il browser può inviare i dati al server web, invocando il programma CGI che si occupa dell'inserimento dei dati nel database rubricaDB. Il programma CGI viene realizzato come uno script Perl che, mediante le librerie DBI (*DataBase Interface*) per il DBMS MySQL,

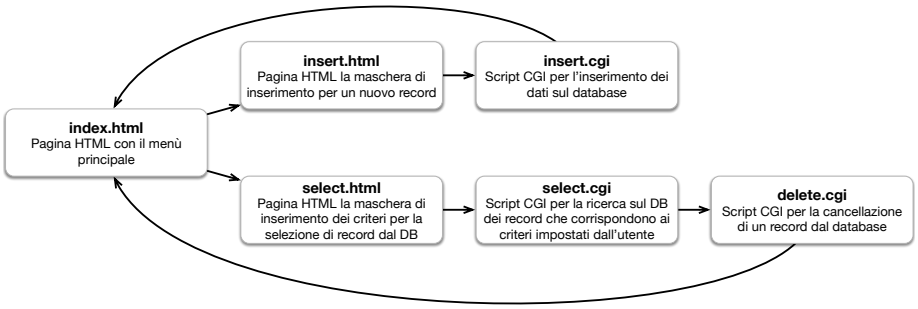

Fig. 6: Schema della concatenazione dei file HTML e degli script CGI in linguaggio Perl, che implementano le diverse funzioni della web application

eseguirà l'inserimento vero e proprio del nuovo record sul database, mediante l'istruzione SQL `insert`. Al termine dell'inserimento, il programma CGI presenta un messaggio all'utente in una pagina HTML di output, su cui è presente anche il link per tornare al menù principale.

Selezionando la seconda opzione del menù principale, viene visualizzata una form di inserimento dati con cui è possibile inserire i criteri di selezione dei record presenti sul database. I criteri di selezione sono costituiti da tre stringhe di caratteri, anche nulle, contenenti parte dell'attributo nome, dell'attributo cognome e dell'attributo e-mail; le tre stringhe possono essere anche tutte nulle, in tal caso significa che l'utente non ha impostato alcun criterio di selezione e quindi vengono restituiti tutti i record presenti nel database. La pagina con la maschera di inserimento dei criteri di selezione viene realizzata come una pagina HTML statica, dal momento che non è necessaria alcuna elaborazione per costruire la form.

Una volta completato l'inserimento dei criteri di selezione, l'utente, selezionando un bottone presente nella form, richiama il programma CGI per l'esecuzione della ricerca sul database dei record che corrispondono ai criteri impostati. Questa funzione è realizzata con uno script CGI in linguaggio Perl. L'output prodotto dallo script è una pagina HTML con tutte le informazioni relative ai record selezionati, riportate in forma tabellare. Accanto ad ogni riga della tabella di output è presente un link per richiamare la funzione di cancellazione del record corrispondente alla voce selezionata dall'utente.

Selezionando una delle righe della tabella, viene invocato un altro programma CGI che provvede ad eliminare dal database il record selezionato.

Possiamo quindi schematizzare come in Figura 6 la struttura delle pagine e degli script CGI in linguaggio Perl, che compongono la web application per la gestione della rubrica degli indirizzi.

4.1 Specifiche sui dati

Per semplificare al massimo il nostro progetto, il database `rubricaDB` è costituito da una sola entità, denominata `rubrica`. L'entità è composta da sei attributi, così come indicato in Tabella 1.

Tutti gli attributi a meno di `cognome` e `id` possono anche essere nulli. L'attributo `id` rappresenta un codice identificativo univoco dei record presenti nella tabella. È costituito da un numero intero che deve essere incrementato automaticamente di un'unità dal DBMS ogni volta che viene aggiunto un record nella tabella. Pertanto è un attributo non nullo e rappresenta la chiave primaria della tabella. Il numero telefonico, pur essendo costituito prevalentemente da cifre numeriche, viene gestito come stringa di caratteri, in modo da accettare anche simboli come "+", "-" e "/", che spesso sono presenti nei numeri telefonici; inoltre, trattando il numero telefonico come una stringa di caratteri

Attributo	Tipo	Vincoli
id	numero intero	non nullo, chiave primaria, auto-incrementante
nome	stringa di 20 caratteri	
cognome	stringa di 30 caratteri	non nullo
email	stringa di 50 caratteri	
tel	stringa di 30 caratteri	
data_nascita	data	

Tab. 1: Attributi della tabellarubrica

(e non come un numero naturale), può anche iniziare con uno o più cifre "0". Infine la data di nascita viene tratta come un attributo di tipo "data" e non come una semplice stringa di caratteri. In questo modo si potranno eseguire operazioni di ordinamento o di estrazioni dei record basate anche sulle date di nascita, demandando al DBMS l'onere di operare sulle date.

Per la realizzazione dell'entità rubrica come tabella del database rubricaDB, può essere utilizzato il seguente comando SQL sulla shell del prodotto MySQL, utilizzando l'utenza di amministratore del database:

```
mysql>  create table ‘rubrica‘ (
  ‘id‘ int(11) not null auto_increment,
  ‘nome‘ char(20),
  ‘cognome‘ char(30) not null,
  ‘email‘ char(50),
  ‘tel‘ char(30),
  ‘data_nascita‘ date,
  primary key (‘id‘)
) engine=InnoDB;
```

La sintassi utilizzata in questa istruzione SQL è quella tipica del prodotto RDBMS MySQL; utilizzando prodotti software differenti (PostgreSQL, Microsoft SQL Server, Oracle Database, IBM DB2, ecc.) la sintassi di questa istruzione può variare in alcuni dettagli, senza alterare però gli aspetti sostanziali nella definizione della tabella.

4.2 Il menù principale

Il menù principale del sistema è la pagina con cui viene accolto l'utente sulla web application ed è anche la pagina attraverso cui l'utente può selezionare una delle due funzioni principali rese disponibili dall'applicazione: l'inserimento di un nuovo record e la selezione dei record presenti nel database.

La pagina con il menù principale viene quindi realizzata mediante un file HTML "statico", che presenta due link alle due distinte funzionalità attivabili dal menù. La pagina HTML ha la struttura riportata di seguito:

```
1  <!DOCTYPE HTML>
2  <html>
3   <head>
4    <title>Rubrica degli indirizzi - Men&ugrave; principale</title>
5    <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8    <h1>Rubrica degli indirizzi</h1>
9    <h2>Men&ugrave; principale</h2>
10   <dl>
11    <dt><a href="select.html" title="Selezione dei dati presenti nella
          rubrica">Selezione dati</a></dt>
```

```
12    <dd>Ricerca e visualizza i dati presenti nella rubrica degli indirizzi</
          dd>
13    <dt><a href="insert.html" title="Inserimento dati in rubrica">
          Inserimento dati</a></dt>
14    <dd>Inserimento di un nuovo record nell'archivio degli indirizzi</dd>
15   </dl>
16  </body>
17 </html>
```

Il file HTML è codificato secondo lo standard HTML 5, come si evince dalla prima riga. Per la presentazione dei contenuti utilizziamo con coerenza in tutte le pagine della web application le stesse "convenzioni grafiche", in modo tale da aiutare l'utente ad orientarsi nelle diverse pagine dell'applicazione. Ad esempio, nell'intestazione della pagina, il titolo è definito riportando il nome della web application e, di seguito, il nome della funzione visualizzata nella pagina ("Menù principale", in questo caso).

Sempre nell'intestazione della pagina viene incluso il file CSS (*Cascading Style Sheet*) che il browser applicherà a questa e alle altre pagine della web application, nel *rendering* dei contenuti sullo schermo dell'utente; il collegamento tra la pagina HTML e il file CSS avviene mediante il seguente tag presente nell'intestazione della pagina:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Il file CSS "`style.css`" con lo stile grafico da applicare ai diversi elementi delle pagine HTML della web application, è riportato di seguito:

```
1  body {
2    padding: 0pt;
3    margin: 0pt;
4  }
5
6  h1 {
7    font: bold 24px sans-serif;
8    text-align: center;
9    color: white;
10   background: #E0E0E0;
11   display: block;
12   padding: 8pt;
13   margin: 0pt;
14   box-shadow: 0px -5px 5px #B0B0B0 inset;
15 }
16
17 h2 {
18   font: bold 18px sans-serif;
19   display: block;
20   padding: 8pt;
21   color: green;
22 }
23
24 a {
25   text-decoration: none;
26 }
27
28 a:hover {
29   color: red;
30 }
31
32 dt a {
33   padding: 3pt;
34   color: green;
35   border: solid 2pt white;
36 }
37
38 dt a:hover {
39   background: #EEFFEE;
40   color: green;
```

```
41    border: solid 2pt green;
42  }
43
44  dt {
45    font: bold 14px sans-serif;
46    padding: 5pt;
47  }
48
49  dd {
50    padding-bottom: 20pt;
51  }
52
53  p {
54    margin-left: 8pt;
55  }
56
57  table {
58    margin-left: 8pt;
59  }
60
61  form th {
62    text-align: right;
63    background: white;
64    border-bottom: none;
65  }
66
67  form td {
68    border-bottom: none;
69  }
70
71  th {
72    background: #EEFFEE;
73    border-bottom: 3pt solid green;
74    padding: 3pt;
75  }
76
77  td {
78    border-bottom: 1pt solid green;
79    padding: 3pt;
80  }
```

La pagina del menù principale presenta due link per l'attivazione delle due funzioni principali della web application; il primo link ipertestuale rimanda alla pagina `select.html` per la funzione di selezione dei record presenti in archivio, mentre il secondo link rimanda alla pagina `insert.html` per la funzione di inserimento di un nuovo record in archivio.

4.3 Inserimento di un nuovo record

La funzione per l'inserimento di un nuovo record in archivio è implementata attraverso due componenti: la pagina HTML presente nel file `insert.html` e il programma Perl CGI codificato nel file `insert.cgi`. La prima si limita a costruire la pagina HTML contenente la form per l'inserimento dei dati del nuovo record. Il sorgente in linguaggio HTML per la costruzione della form è riportato di seguito:

```
1  <!DOCTYPE HTML>
2  <html>
3   <head>
4    <title>Rubrica degli indirizzi - Inserimento dati</title>
5    <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8    <h1>Rubrica degli indirizzi</h1>
9    <h2>Inserimento di un nuovo contatto</h2>
10   <form action="insert.cgi" method="GET">
```

```
11    <table>
12     <tr><th>Nome</th>
13      <td><input name="nome" type="text" size="20"></td></tr>
14     <tr><th>Cognome</th>
15      <td><input name = "cognome" type="text" size="30"></td></tr>
16     <tr><th>E-mail</th>
17      <td><input name = "mail" type="text" size="30"></td></tr>
18     <tr><th>Telefono</th>
19      <td><input name = "telefono" type="text" size="15"></td></tr>
20     <tr><th>Data di nascita</th>
21      <td><input name = "giorno" type="text" size="2">/<input name="mese"
            type="text" size="2">/<input name="anno" type="text" size="4"></td
            ></tr>
22     <tr><td colspan="2" align="center">
23      <input type="submit" value="Salva"> <input type="reset" value="
            Ripristina"></td></tr>
24    </table>
25   </form>
26  </body>
27 </html>
```

La pagina HTML include lo stesso *stylesheet* del menù principale (riga 5 del sorgente HTML), pertanto il layout grafico-funzionale è il medesimo, come mostrato in Figura 8.

La pagina HTML presenta i campi di input allineati all'interno delle celle di una tabella, in modo da garantire una presentazione ordinata della maschera di inserimento dati. La form presenta inoltre i due bottoni di tipo "submit" e "reset": selezionando il primo il browser eseguirà una connessione HTTP con il server web, richiedendo la risorsa `insert.cgi` con il metodo GET e passando i parametri in input, concatenati alla stessa URL della pagina richiesta. Ad esempio:

```
insert.cgi?nome=Mario&cognome=Rossi&mail=rossi@libero.it&
   telefono=069876543&giorno=17&mese=7&anno=1977
```

Il programma CGI `insert.cgi` utilizza la libreria "`cgi-lib.pl`", una libreria del linguaggio Perl che implementa alcune utili funzioni per rendere più semplice e veloce lo sviluppo di programmi CGI. In particolare la prima operazione che deve essere implementata dallo script Perl è la lettura dei parametri forniti in input al programma dal server HTTP. Come abbiamo visto nella Sezione 3.2, il server HTTP imposta il valore della variabile d'ambiente `REQUEST_METHOD`, assegnandogli il valore "`GET`" o "`POST`" a seconda del metodo HTTP utilizzato dal client per inviare al server web i dati inseriti in input dall'utente. Se viene utilizzato il metodo GET, i dati in input, sotto forma di

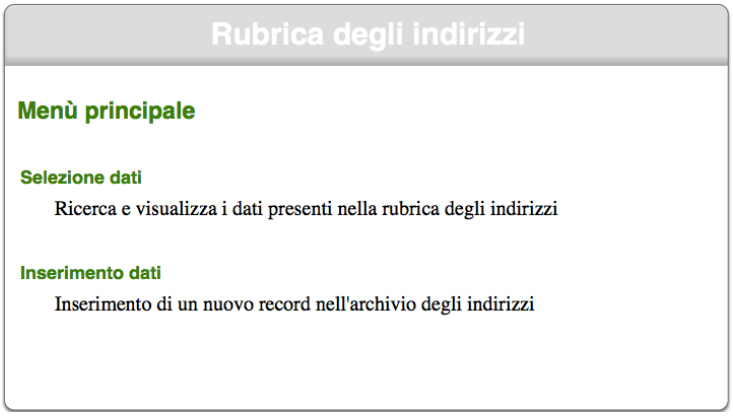

Fig. 7: La pagina con il menù principale

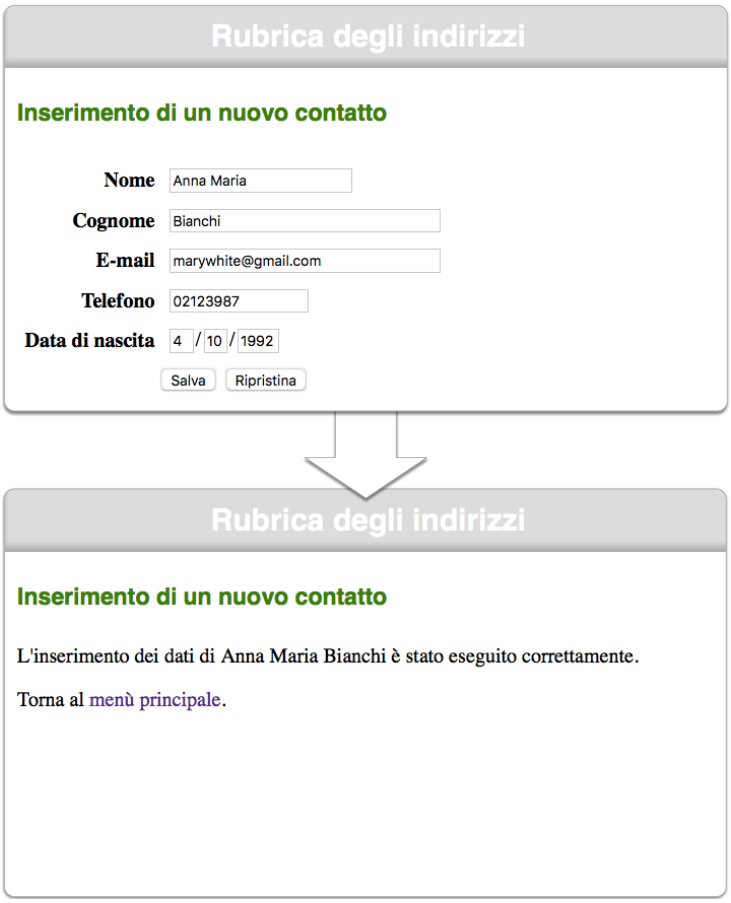

Fig. 8: La maschera di inserimento dati per l'aggiunta di un nuovo record in archivio (in alto) e l'output visualizzato dal programma CGI al termine dell'inserimento del nuovo record sul database (in basso)

un'unica stringa in formato URL encoded, vengono assegnati alla variabile d'ambiente `QUERY_STRING`. Se invece viene utilizzato il metodo POST, allora la stringa con i dati in formato URL encoded, verrà passata dal server web al programma CGI sul canale di standard input.

La funzione `ReadParse` presente nella libreria `cgi-lib.pl`, si occupa di verificare quale metodo è stato usato per passare i parametri in input allo script, controllando il valore della variabile d'ambiente `REQUEST_METHOD`; quindi, dopo aver acquisito la stringa in formato URL encoded, si occupa di eseguire la suddivisione delle stringa nei suoi elementi, costituiti da un nome di un attributo e dal suo valore. I dati vengono quindi assegnati ad una struttura dati tipica del linguaggio Perl (e di altri linguaggi), denominata *hash table* (o anche array associativo). In questa struttura dati, viene costruito un array, i cui indici sono costituiti da stringhe di caratteri (anziché da numeri interi, come avviene di solito per gli array). In particolare, la funzione `ReadParse` costruisce la hash table "`%in`", utilizzando come chiave di ciascun elemento il nome dell'attributo e come valore il valore dell'attributo stesso.

Ad esempio, facendo riferimento all'esempio riportato a pagina 12, se la stringa in formato URL encoded è la seguente:

```
nome=Mario&cognome=Rossi&mail=rossi@libero.it&telefono=069876543&
    giorno=17&mese=7&anno=1977
```

allora la funzione ReadParse costruisce una hash table con i seguenti elementi:

```
$in{"nome"} = "Mario"
$in{"cognome"} = "Rossi"
$in{"mail"} = "rossi@libero.it"
...
```

Di seguito riportiamo il sorgente in linguaggio Perl dello script "insert.cgi", che, dopo aver acquisito in input i dati inseriti dall'utente nella form, esegue l'operazione di inserimento del nuovo record sul database:

```
1  #!/usr/local/bin/perl
2  require "cgi-lib.pl";
3  &ReadParse;
4  print "Content-Type: text/html\n\n";
5  print "<!DOCTYPE HTML>\n";
6  print "<html>\n<head><title>Rubrica degli indirizzi - Inserimento dati - OK
       </title>\n";
7  print "<link rel=\"stylesheet\" type=\"text/css\" href=\"style.css\">\n</
       head>\n";
8  print "<h1>Rubrica degli indirizzi</h1>\n";
9  print "<h2>Inserimento di un nuovo contatto</h2>\n";
10 use DBI;
11 $db = DBI->connect("dbi:mysql:dbname=rubricaDB", "rubricaUser", "xy-99") or
        die DBI::errstr;
12 $query = $db->prepare("insert into rubrica (nome, cognome, email, tel,
       data_nascita) values ('$in{nome}', '$in{cognome}', '$in{mail}', '$in{
       telefono}', '$in{anno}-$in{mese}-$in{giorno}')");
13 $query->execute();
14 $query->finish();
15 $db->disconnect();
16 print "<p>L'inserimento dei dati di $in{nome} $in{cognome} &egrave; stato
       eseguito correttamente.</p>\n";
17 print "<p>Torna al <a href=\"index.html\" title=\"Men&ugrave; principale\">
       men&ugrave; principale</a>.</p>\n";
18 print "</body>\n</html>\n";
```

Lo script, dopo aver caricato la libreria cgi-lib.pl (riga 2), esegue la funzione ReadParse (riga 3). Quindi (riga 4) produce come prima riga di output, con l'istruzione print il tipo MIME del contenuto che produrrà in output nelle istruzioni successive. In questo caso il tipo MIME è "text/html", visto che l'output è costituito proprio da una pagina HTML. Da notare che la riga contenente l'indicazione del tipo MIME, deve essere seguita necessariamente da una riga completamente vuota, quindi l'istruzione print di riga 4 termina con due caratteri di "a capo" (\n). Poi, con le istruzioni print da riga 5 a riga 9, viene prodotta in output la prima parte della pagina HTML con cui sarà presentato all'utente l'esito dell'operazione di inserimento del nuovo record nel database.

Con l'istruzione a riga 10 viene caricata la libreria DBI, necessaria per la connessione del programma Perl con il DBMS MySQL. Questa libreria mette a disposizione alcune funzioni (metodi) per la connessione con il database e l'esecuzione di query in linguaggio SQL. A riga 11 viene eseguita la connessione con il database "rubricaDB", utilizzando l'account "rubricaUser" definito con le istruzioni riportate a pagina 7.

A riga 12 viene costruita la query in linguaggio SQL per l'inserimento nella tabella del database di un nuovo record composto dai dati forniti in input dall'utente. La query prevede l'inserimento di tutti i campi (a meno dell'attributo id che viene calcolato e valorizzato automaticamente dal DBMS): alcuni valori potranno essere anche nulli, nel qual caso in corrispondenza di tale valore il campo del record sul database risulterà nullo. Come valori dei campi, naturalmente, vengono usati gli elementi della hash table %in. La data viene composta concatenando l'anno, il mese e il giorno inseriti dall'utente, separandoli con dei trattini, così come previsto dal formato data di MySQL.

Con le istruzioni alle righe 13, 14 e 15, la query SQL viene eseguita, quindi viene conclusa la transazione con il database e chiusa la connessione. Nella pagina HTML viene visualizzato un messaggio in cui si informa l'utente che il record è stato inserito nel database; viene anche presentato un link per tornare al menù principale (vedi Figura 8).

4.4 Selezione dei record presenti sul database

La funzione per la selezione e la visualizzazione dei record presenti nel database viene implementata in modo analogo alla precedente funzione per l'inserimento di dati. Un file HTML (`select.html`) visualizza una form di inserimento dati con cui l'utente potrà impostare una parte del nome, del cognome e dell'indirizzo e-mail della persona che sta cercando sul database. Se non inserisce alcun valore nei campi della form, significa che non intende impostare alcun criterio per selezionare i dati sul database e che quindi il sistema visualizzerà tutti i dati presenti, senza alcun filtro.

Di seguito riportiamo il sorgente in linguaggio HTML del file `select.html`. Anche questo file include il foglio di stile CSS, in modo da garantire una certa omogeneità grafica rispetto alle altre pagine del sistema (vedi Figura 9). Quindi presenta una form con tre campi, denominati `nome`, `cognome` e `mail`.

```
1  <!DOCTYPE HTML>
2  <html>
3   <head>
4    <title>Rubrica degli indirizzi - Consultazione dati</title>
5    <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8    <h1>Rubrica degli indirizzi</h1>
9    <h2>Consultazione dati</h2>
10   <form action="select.cgi" method="GET">
11   <table>
12    <tr><th>Nome</th>
13     <td><input name="nome" type="text" size="20"></td></tr>
14    <tr><th>Cognome</th>
15     <td><input name="cognome" type="text" size="30"></td></tr>
16    <tr><th>E-mail</th>
17     <td><input name="mail" type="text" size="30"></td></tr>
18    <tr><td colspan="2" align="center"><input type="submit" value="Cerca">
19     <input type="reset" value="Ripristina"></td></tr>
20   </table>
21   </form>
22  </body>
23 </html>
```

Quando l'utente ha completato la compilazione dei campi della form e ha selezionato il bottone di *submit*, il browser richiede la pagina `select.cgi`, passandogli come dati in input i valori inseriti nella form. Il programma CGI è implementato con lo script in linguaggio Perl riportato di seguito. Lo script utilizza la stessa libreria `cgi-lib.pl` e `DBI` già viste nel caso dello script `insert.cgi`.

Con la chiamata alla funzione `ReadParse` a riga 3, vengono acquisiti i dati in input e memorizzati negli elementi della tabella hash "`%in`": `$in{nome}` contiene il valore del campo `nome` della form, `$in{cognome}` e `$in{mail}` contengono rispettivamente il valore del campo `cognome` e `mail`.

La query sul database, questa volta una query con l'istruzione SQL `select`, viene preparata ed eseguita analogamente a quanto già visto per lo script `insert.cgi`, nelle righe 11, 12 e 13. L'istruzione `select`, come si può vedere a riga 12, utilizza l'operatore `like` e il carattere *jolly* "`%`" per definire il criterio con cui vengono selezionati i record sulla tabella `rubrica`. In questo modo i valori inseriti dall'utente nella form saranno usati

come sotto-stringhe che devono essere contenute nei valori dei campi corrispondenti nella tabella del database.

```
1  #!/usr/local/bin/perl
2  require "cgi-lib.pl";
3  &ReadParse;
4  print "Content-Type: text/html\n\n";
5  print "<!DOCTYPE HTML>\n";
6  print "<html>\n<head><title>Rubrica degli indirizzi - Consultazione dati</
       title>\n";
7  print "<link rel=\"stylesheet\" type=\"text/css\" href=\"style.css\">\n</
       head>\n";
8  print "<h1>Rubrica degli indirizzi</h1>\n";
9  print "<h2>Consultazione dati</h2>\n";
10 use DBI;
11 $db = DBI->connect("dbi:mysql:dbname=rubricaDB", "rubricaUser", "xy-99") or
        die DBI::errstr;
12 $query = $db->prepare("select id, nome, cognome, email, tel, data_nascita
       from rubrica where nome like '%$in{nome}%' and cognome like '%$in{
       cognome}%' and email like '%$in{mail}%' order by cognome, nome");
13 $query->execute();
14 if ($query->rows() == 0) {
15   print "<p>Nessun record selezionato.</p>";
16 } else {
17   print "<table>\n<tr><th>Nome</th><th>Cognome</th><th>E-mail</th><th>
         Telefono</th><th>Data di nascita</th><th>Canc.</th></tr>\n";
18   for ($i=1; $i <= $query->rows(); $i++) {
19     ($id, $nome, $cognome, $email, $tel, $data) = $query->fetchrow();
20     print "<tr><td>$nome</td><td>$cognome</td><td><a href=\"mailto:$email
         \">$email</a></td><td>$tel</td><td>$data</td><td><a href=\"delete.
         cgi?id=$id\" title=\"Elimina il record n. $id\">X</a></td></tr>\n";
21   }
22   print "</table>\n";
23 }
24 $query->finish();
25 $db->disconnect();
26 print "<p>Torna al <a href=\"index.html\" title=\"Men&ugrave; principale\">
       men&ugrave; principale</a>.</p>\n";
27 print "</body>\n</html>\n";
```

La query eseguita a riga 13, restituisce un *cursore* che ci permette di scorrere gli elementi della tabella selezionati con la query SQL. L'oggetto `$query` è proprio la modalità in cui la libreria DBI del linguaggio Perl realizza il cursore. il metodo `rows()` applicato al cursore ci restituisce il numero di righe selezionate; alle righe 14 e 15, utilizzando l'informazione restituita da questo metodo, il sistema visualizza un messaggio di avviso, nel caso in cui i criteri impostati come filtro di ricerca non abbiano consentito di selezionare alcun record dalla tabella.

Se invece il numero di righe selezionate è maggiore di zero (righe 16–23), viene visualizzata una tabella HTML e, con un ciclo `for`, vengono presi in esame, uno dopo l'altro, secondo l'ordine con cui questi sono stati selezionati dalla query SQL, i record corrispondenti ai criteri di ricerca. Il metodo `$query->fetchrow()` permette di acquisire la riga su cui è puntato il cursore e di distribuire i vari valori nelle variabili elencate a sinistra dell'operatore "=" (si tratta di una lista di variabili); in questo modo alla variabile `$id` viene assegnato il primo campo del record selezionato, nell'ordine in cui sono elencati nella query SQL e corrispondente all'attributo `id`, alla variabile `$nome` viene assegnato il secondo campo del record selezionato (il valore dell'attributo `nome`), e così via.

Questi valori vengono poi usati visualizzandoli in una riga della tabella di output in formato HTML, prodotta a riga 20 con l'istruzione `print`. Da notare che l'ultima colonna della tabella è un link ipertestuale che, in modalità GET, richiama lo script CGI `delete.cgi` passandogli come unico argomento il valore del campo `id` del record vi-

sualizzato. Il valore della chiave `id` è infatti l'unico dato in input che deve essere fornito allo script per la cancellazione di un record.

Con le istruzioni alle righe 24–27, viene conclusa la query e viene chiusa la connessione con il database, quindi viene completata anche la pagina HTML con l'output dell'operazione di selezione.

4.5 Eliminazione di un record

La funzione di eliminazione viene invocata selezionando il link corrispondente alla colonna "Canc." presente nella tabella di output dello script CGI `select.cgi`. Lo script è molto semplice e ricalca in buona sostanza gli script CGI visti nelle pagine precedenti.

Anche in questo caso, infatti, vengono usate le librerie `cgi-lib.pl` e `DBI`. Viene acquisito in input l'identificativo del record da cancellare utilizzando la funzione `ReadParse`, quindi viene costruita la prima parte della pagina HTML di output. La connessione al database viene stabilita come di consueto con il metodo `DBI->connect(...)`, quindi viene preparata ed eseguita la query SQL con l'istruzione `delete`: la condizione utilizzata nella clausola `where` non ammette ambiguità: deve essere eliminato il solo record per cui il valore dell'attributo `id` corrisponde con il valore fornito in input allo script CGI (riga 12).

Dopo aver eseguito la query (riga 13), viene chiusa la connessione e viene chiusa anche la pagina HTML di output, con alcuni messaggi informativi ed un link ipertestuale che rimanda al menù principale.

```
1  #!/usr/local/bin/perl
2  require "cgi-lib.pl";
3  &ReadParse;
4  print "Content-Type: text/html\n\n";
5  print "<!DOCTYPE HTML>\n";
6  print "<html>\n<head><title>Rubrica degli indirizzi - Cancellazione dati -
       OK</title>\n";
7  print "<link rel=\"stylesheet\" type=\"text/css\" href=\"style.css\">\n</
       head>\n";
8  print "<h1>Rubrica degli indirizzi</h1>\n";
9  print "<h2>Cancellazione di un contatto</h2>\n";
10 use DBI;
11 $db = DBI->connect("dbi:mysql:dbname=rubricaDB", "rubricaUser", "xy-99") or
        die DBI::errstr;
12 $query = $db->prepare("delete from rubrica where id='$in{id}'");
13 $query->execute();
14 $query->finish();
15 $db->disconnect();
16 print "<p>La cancellazione del record selezionato (id n. $in{id}) &egrave;
       stata eseguita correttamente.</p>\n";
17 print "<p>Torna al <a href=\"index.html\" title=\"Men&ugrave; principale\">
       men&ugrave; principale</a>.</p>\n";
18 print "</body>\n</html>\n";
```

L'output del programma è presentato in Figura 9. In questo esempio l'utente specifica nel campo nome la stringa "`Mar`": il sistema seleziona così tutti i record in cui il valore dell'attributo `nome` contiene la sottostringa "`Mar`": Maria Bianchi e i due Mario Rossi corrispondono a questo criterio e quindi vengono visualizzati nella tabella HTML di output.

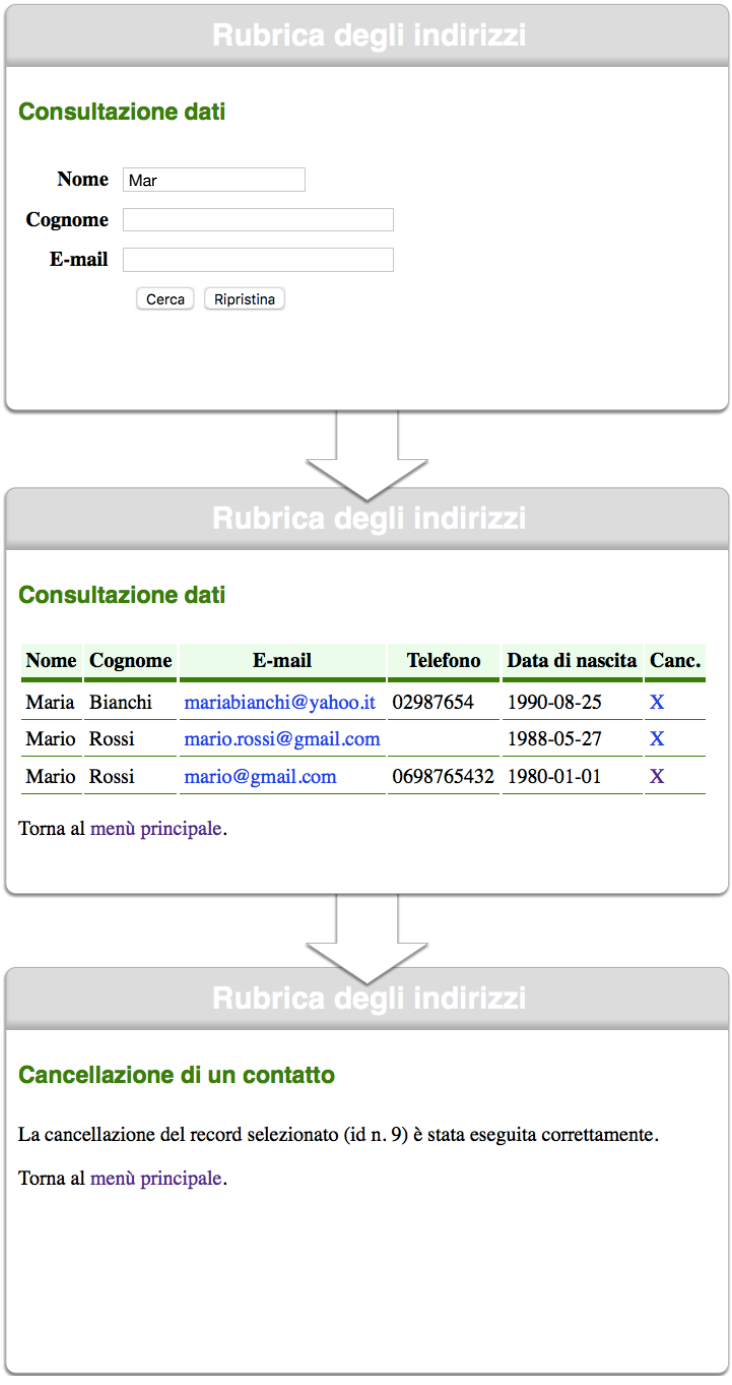

Fig. 9: La maschera di inserimento dati per l'aggiunta di un nuovo record in archivio