



## Progetto LiberA

L'implementazione di  
un'applicazione Web (PHP,  
CSS3, HTML5, MYSQL,  
JAVASCRIPT, XAMPP) per  
facilitare la gestione di  
un'associazione culturale,  
contestualmente  
un'associazione di danza.

Nasce dalla necessità di  
velocizzare i processi di  
registrazione/deregistrazione e  
informativi più utilizzati dagli  
addetti.

---

*Studente/Relatore: Antonio Zambito*

*Matricola: 124/1032*

*CdL: Informatica (0124)*

*Prof.titolare: Raffaele Montella*

*Sede: Centro Direzionale – Isola C4 (NA)*

---

# INDICE

➤ **Requisiti e Scopo**

Requisiti e Scopo

➤ **Creazione DB e configurazione Server Virtuale**

➤ **Forma e stile: CSS, JavaScript**

➤ **PHP: pagine e functions, interazione con il DB. Funzioni sulle classi**

➤ **Procedure e funzioni**

## 1. Requisiti e Scopo

L'applicazione web LiberA è un'idea del suddetto relatore, in seguito ad una (apparentemente) frivola richiesta da parte del presidente di un'associazione di danza, la quale esprime, in sua presenza, volutamente il desiderio di poter possedere uno strumento informatico in grado di permetterle una gestione meno cartacea dell'associazione e che fosse in grado di velocizzare i processi di inserimento persone o per semplici informazioni riguardanti le stesse o le attività svolte all'interno dell'associazione.

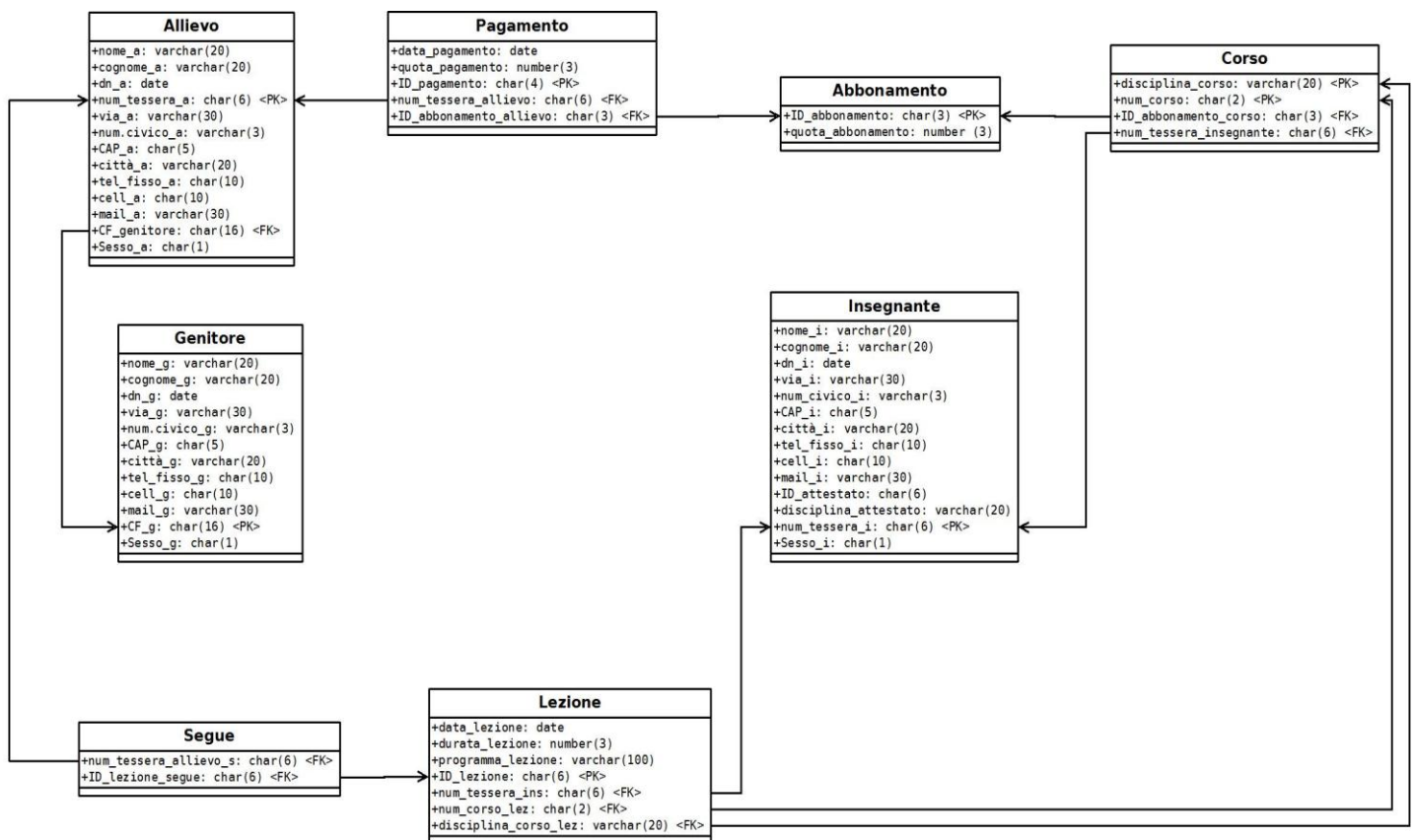
A tal scopo il relatore ha pensato bene di sviluppare questa idea (richiesta), in primis, facendone oggetto del proprio progetto per l'esame di *Basi di Dati*, inerente al piano di studi del CdL Informatica presso la Università Parthenope. Portato a termine lo sviluppo di tale progetto (in ambiente Oracle 11g Xpress Edition, senza alcuna interfaccia grafica, operando con riga di comando e non pronto all'uso di qualsiasi utente non esperto) in SQL, PL/SQL, si pensa bene di dare forma e colore a questa idea e portare ad uno stato più avanzato e *for-all* il progetto.

Da qui segue l'implementazione del progetto in oggetto alla seguente relazione: LiberA (acronimo di *Liberdanza Administration*).

LiberA segue la linea del precedente progetto di *Basi di Dati*, ovvero utilizza lo stesso schema relazionale per l'implementazione del Database e gli stessi requisiti raccolti in fase di intervista al richiedente.

Si tratta, innanzitutto di una Web-Application; le tecnologie/linguaggi utilizzati per lo sviluppo di tale applicazione sono:

PHP, HTML, CSS, Javascript, MySQL, XAMPP. Le prime 4 sono state utilizzate, unitamente, per dare un form all'applicazione e per permettere l'interazione di quest'ultima col Database creato in XAMPP (Server Virtuale, utile per la prova dell'applicazione e la gestione dell'applicazione tutta) ed implementato/manipolato con linguaggio MySQL. Di seguito, per scopo illustrativo, è mostrato lo schema relazionale di cui sopra:



Per poter comprendere ancor meglio lo schema, si rimanda allo studio dei Database Relazionali (*Basi di Dati*).

## 2. Creazione DB e configurazione Server Virtuale

Dopo aver introdotto, anche tecnicamente, il progetto, si comincia con la fase di sviluppo in ambiente XAMPP, ovvero il Server virtuale di cui si fa uso per l'implementazione e lo sviluppo del progetto. E' stato scelto questo virtual server perchè risulta essere molto pratico ed efficiente in quanto racchiude, nel suo *core*, PHPMyAdmin, MySQL, FileZilla FTP e tanti altri utili strumenti per la creazione di un applicazione web. Il primo step è quello della creazione del Database dell'applicazione, in cui andranno memorizzati tutti i dati richiesti e necessari (ad esempio, contestualmente, gli users che hanno accesso all'applicazione, con rispettive password). Lo step successivo è la configurazione, secondo le proprie necessità, del Server virtuale inserendo anche i dati di accesso ad esso.

## 3. Forma e stile: CSS, JavaScript

Una volta terminata la configurazione, si mette in standby la parte configurativa riguardante XAMPP e si comincia con la vera e propria implementazione in PHP delle pagine che comporranno l'applicazione. Per editare in un determinato linguaggio, si è fatto uso di Notepad++, una sorta di Writer adatto ai programmatori, il quale riesce ad evidenziare o colorare le parole chiave, a seconda del linguaggio di programmazione con cui si sta scrivendo. Un buon uso di un'applicazione è dato anche dallo stile grafico che essa assume, riesce ad attrarre l'utente e far sì che il suo uso sia piacevole e divertente, oltre che utile. Per dare uno stile adatto all'esigenza richiesta, si è fatto affidamento ad un CSS trovato in rete, su di un affidabilissimo sito [<http://templated.co/>], sul quale è possibile reperire numerosi Template e CSS già pronti all'uso in modo da risparmiare tempo nello sviluppo vero e proprio, ma comunque editabili. All'interno dello stesso pacchetto contenente i CSS, si possono trovare i file JS che permettono di creare movimenti e transizioni all'interno della pagina.


## 4. PHP: pagine e functions, interazione con il DB. Funzioni sulle classi

La prima pagina da creare è l'indice che permette l'accesso all'applicazione, ovvero la pagina di Login che, in questo caso, sarà la pagina di apertura dell'applicazione (*index.php*).

# Liberdanza Management

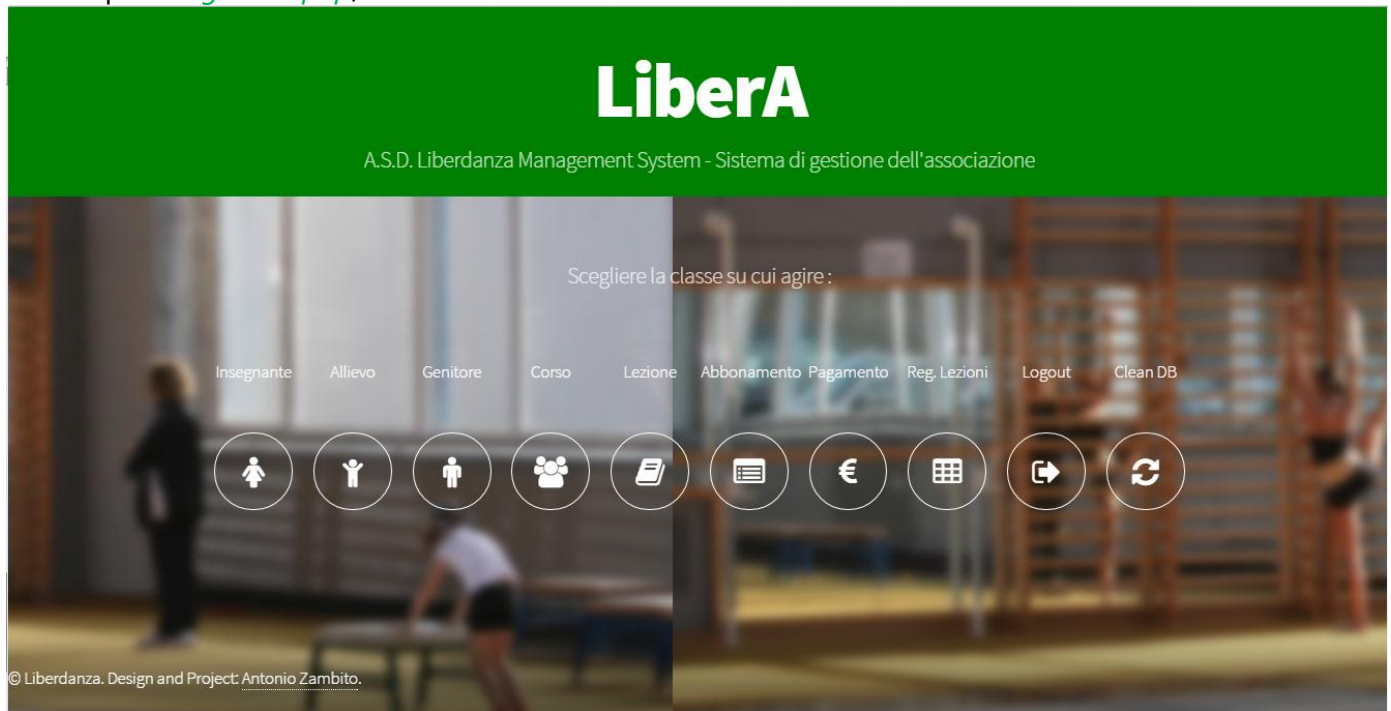
## Login

Login

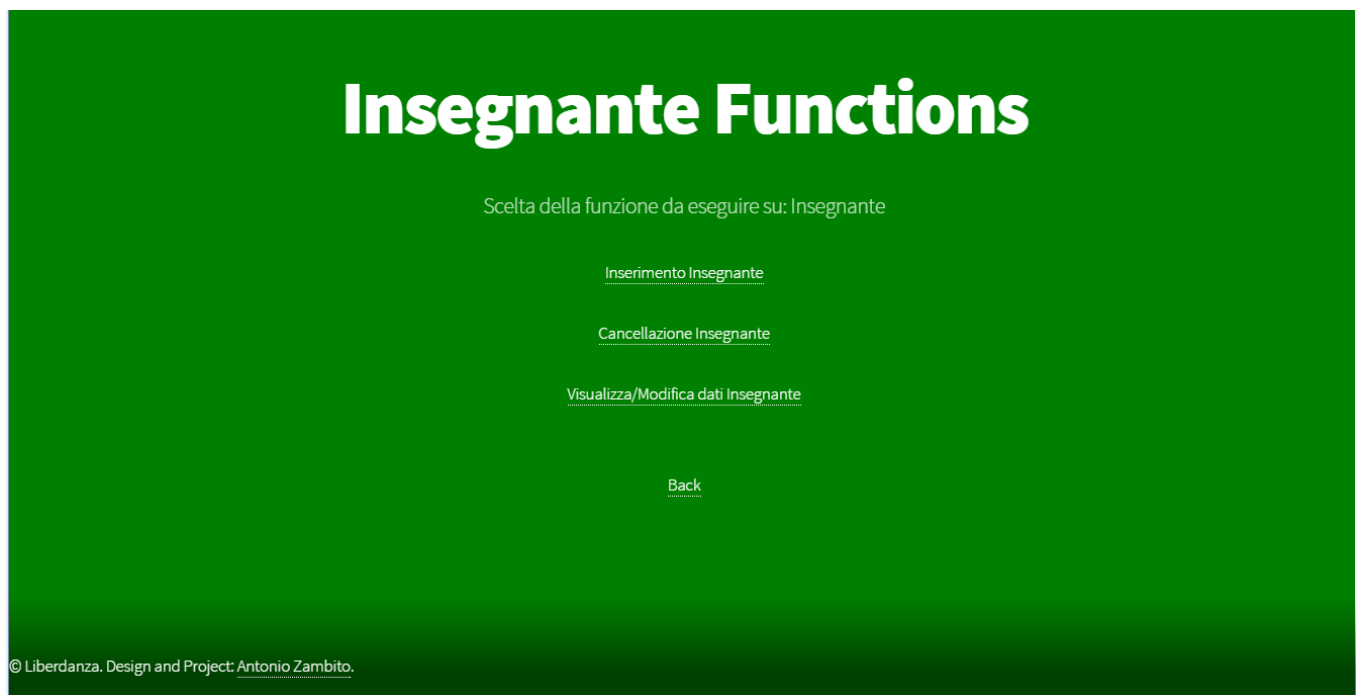


(Login.Page)  
4

Ad accesso eseguito, si aprirà la Start Page dell'applicazione, nella quale le funzioni disponibili sono state divise per classi su cui agire (*first.php*). Ad esempio viene definita la classe Insegnante, al cui click si aprirà la corrispondente pagina in cui saranno mostrate le funzioni disponibili per la classe scelta. (*'nome\_classe.php'*, nell'esempio *insegnante.php*).



(Start Page)



(Pagina della classe Insegnante contenente solo le funzioni riguardanti la classe)

Si procede, quindi, con la definizione delle pagine che permettono l'esecuzione delle funzioni relative alle singole classi. Una buona parte delle classi di questo schema ha delle funzioni condivise (Inserimento,

Cancellazione, Selezione e Modifica) con l'unica differenza degli attributi relativi alla singola classe che possono differire da un'altra.

Un esempio di tali funzioni può essere l'inserimento di un Insegnante nel Database, come mostrato nello screenshot seguente:

The screenshot shows a web form titled "Inserimento Insegnante" on a dark blue background. The form consists of several input fields with labels in Italian, followed by a submit button and a back link.

Labels and fields (from top to bottom):

- Nome Insegnante
- Cognome Insegnante
- Data di nascita Insegnante(GG/MM/AAAA)
- Via Insegnante
- Num.civico Insegnante
- CAP Insegnante
- Città Insegnante
- Tel.fisso Insegnante
- Cell Insegnante
- Mail Insegnante
- Sesso Insegnante
- M (radio button)
- Inserisci
- Back .....

(form di inserimento dati classe Insegnante)

Ogni pagina *.php* è stata implementata in modo che, se la query (o più di una) che interagisce col Database va a buon fine si viene re-indirizzati alla Start Page (*first.php*) o alla pagina della classe di riferimento; seguendo l'esempio presente, una volta effettuato l'inserimento dell'*insegnante*, se tale operazione è andata a buon fine, comparirà un messaggio di conferma contenente anche il numero di tessera assegnato all'*insegnante* appena inserita. Questo messaggio è presente anche per la fase di inserimento di un allievo, nella quale è permesso, dopo aver inserito i dati, associare un genitore, nel caso in cui l'allievo sia minorenne.

# Inserimento Effettuato.

## Numero tessera assegnato: 11

[Back](#)

Allievo minorenne? Inserisci il CF del genitore/tutore

Num.tessera allievo

CF Genitore allievo

[Aggiungi](#)

[Allievo Maggiorene - Back](#)

(Pagina successiva all'inserimento di un Allievo con form per associazione del genitore)

## 5. Procedure e funzioni

Sono state implementate, successivamente alle operazioni di base definite sopra, altre tre funzioni che permettono:

- la popolazione di un registro delle presenze degli allievi alle lezioni ([registro\\_lezioni.php](#));
- la pulizia del registro delle presenze, cioè l'eliminazione delle tuple ripetute, per evitare visualizzazioni multiple di un dato ([doppioni.php](#));
- Logout dall'applicazione, che termina la sessione attiva e reindirizza alla pagina di Login ([logout.php](#));
- Cancellazione totale di tutti i dati, operazione solitamente svolta all'inizio di un nuovo anno accademico, nell'esempio presente all'inizio del mese di settembre ([svuota.php](#), [truncate\\_all.php](#)).

Si noti che l'ultima funzione non è stata inserita nell'interfaccia utente per motivi di integrità del Database, volendo evitare accidentali perdite di tutti i dati. Tale funzione verrà integrata più in avanti.

A scopo illustrativo, segue il codice *php* della function *truncate\_all.php* :

```

1  <!-- PAGINA PHP che svuota totalmente il DB -->
2  <?php
3      //connessione al DB//
4      $conn=mysqli_connect("localhost","root","tonito91","liberdanza");
5      if(!$conn) {
6          die("Connessione fallita!" . mysqli_connect_error());
7      }
8      //Disabilitazione del controllo dei vincoli referenziali di chiave esterna//
9      //Permette lo svuotamento delle tabelle senza alcun ordine e rispetto di vincoli di integrità referenziale//
10     $sql="SET foreign_key_checks = 0";
11     //elenco delle query che svuotano le tabelle//
12     $sql1="TRUNCATE TABLE abbonamento";
13     $sql2="TRUNCATE TABLE allievo";
14     $sql3="TRUNCATE TABLE corso";
15     $sql4="TRUNCATE TABLE genitore";
16     $sql5="TRUNCATE TABLE insegnante";
17     $sql6="TRUNCATE TABLE lezione";
18     $sql7="TRUNCATE TABLE pagamento";
19     $sql8="TRUNCATE TABLE segue";
20     //Query che riabilita il controllo dei vincoli referenziali di chiave esterna//
21     //Viene ripristinato l'integrità per gli inserimenti immediatamente futuri//
22     $sql9="SET foreign_key_checks = 1";
23     //Esecuzione delle query definite sopra//
24     $res=mysqli_query($conn,$sql);
25     $res1=mysqli_query($conn,$sql1);
26     $res2=mysqli_query($conn,$sql2);
27     $res3=mysqli_query($conn,$sql3);
28     $res4=mysqli_query($conn,$sql4);
29     $res5=mysqli_query($conn,$sql5);
30     $res6=mysqli_query($conn,$sql6);
31     $res7=mysqli_query($conn,$sql7);

```

(truncate\_all.php – Part 1)

```

32     $res8=mysqli_query($conn,$sql8);
33     $res9=mysqli_query($conn,$sql9);
34     //Se tutte le query vanno a buon fine, cioè se tutte le tabelle vengono svuotate, allora l'operazione di svuotamento è riuscita//
35     if($res && $res1 && $res2 && $res3 && $res4 && $res5 && $res6 && $res7 && $res8 && $res9) {
36         echo "Tutti i dati sono stati cancellati.";
37         echo '<meta http-equiv="refresh" content="3; url=first.php">';
38     }
39     //altrimenti, rimanda alla pagina che conferma lo svuotamento//
40     else {
41         echo "Cancellazione fallita. Riprovare.";
42         echo '<meta http-equiv="refresh" content="3; url=svuota.php">';
43     }
44     //Chiusura della connessione col DB//
45     mysqli_close($conn);
46     ?>

```

(truncate\_all.php – Part 2)

Link web-application: <https://students.uniparthenope.it/~0124001032/libera/>