Gestione web dei corsi di aggiornamento

Indice

[1 Introduzione 4](#_Toc41608868)

[1.1 Informazioni sul progetto 4](#_Toc41608869)

[1.2 Abstract 4](#_Toc41608870)

[1.3 Scopo 4](#_Toc41608871)

[2 Analisi 5](#_Toc41608872)

[2.1 Analisi del dominio 5](#_Toc41608873)

[2.2 Analisi e specifica dei requisiti 5](#_Toc41608874)

[2.3 Use case 8](#_Toc41608875)

[2.4 Pianificazione 9](#_Toc41608876)

[2.4.1 Analisi 10](#_Toc41608877)

[2.4.2 Progettazione 11](#_Toc41608878)

[2.4.3 Implementazione 12](#_Toc41608879)

[2.4.4 Testing 13](#_Toc41608880)

[2.4.5 Documentazione 14](#_Toc41608881)

[2.5 Analisi dei mezzi 15](#_Toc41608882)

[2.5.1 Software 15](#_Toc41608883)

[2.5.2 Hardware 15](#_Toc41608884)

[2.5.3 Librerie 15](#_Toc41608885)

[2.5.4 Template HTML 15](#_Toc41608886)

[3 Progettazione 16](#_Toc41608887)

[3.1 Design dell’architettura del sistema 16](#_Toc41608888)

[3.2 Design dei dati e database 17](#_Toc41608889)

[3.2.1 Schema E-R 17](#_Toc41608890)

[3.2.2 Schema logico 17](#_Toc41608891)

[3.2.3 Vincoli 18](#_Toc41608892)

[3.2.4 Foreign key e utenti 18](#_Toc41608893)

[3.2.5 Descrizione tabelle dati 19](#_Toc41608894)

[3.3 Design delle interfacce 21](#_Toc41608895)

[3.3.1 Pagina principale 21](#_Toc41608896)

[3.3.2 Pagina contatti 21](#_Toc41608897)

[3.3.3 Pagina di login 22](#_Toc41608898)

[3.3.4 Pagina richiesta nuova password 22](#_Toc41608899)

[3.3.5 Pagina reimposta password 23](#_Toc41608900)

[3.3.6 Pagina amministrativa 23](#_Toc41608901)

[3.3.7 Pagina ricerca corsi 24](#_Toc41608902)

[3.3.8 Pagina di iscrizione ad un corso 24](#_Toc41608903)

[3.3.9 Pagina gestione corso 25](#_Toc41608904)

[3.3.10 Pagina personale 25](#_Toc41608905)

[3.4 Design procedurale 26](#_Toc41608906)

[3.4.1 Classe Database 26](#_Toc41608907)

[3.4.2 Classe SendMail 27](#_Toc41608908)

[3.4.3 Classe Util 28](#_Toc41608909)

[3.4.4 Classe MSession 28](#_Toc41608910)

[3.4.5 Classe Validator 29](#_Toc41608911)

[4 Implementazione 31](#_Toc41608912)

[4.1 Pattern MVC 31](#_Toc41608913)

[4.2 Creazione database 31](#_Toc41608914)

[4.2.1 Collegamento al database tramite PHP 32](#_Toc41608915)

[4.3 Pagina di login 32](#_Toc41608916)

[4.3.1 Metodo di accesso al sito 33](#_Toc41608917)

[4.4 Pagina password dimenticata 33](#_Toc41608918)

[4.4.1 Invio e-mail 33](#_Toc41608919)

[4.5 Pagina reimposta password 34](#_Toc41608920)

[4.5.1 Riconoscimento utente 34](#_Toc41608921)

[4.5.2 Modifica Password 35](#_Toc41608922)

[4.6 Pagina Home 35](#_Toc41608923)

[4.7 Pagina contatti 36](#_Toc41608924)

[4.8 Pagina di amministrazione 36](#_Toc41608925)

[4.8.1 Gestione pagina principale 36](#_Toc41608926)

[4.8.2 Gestione pagina contatti 38](#_Toc41608927)

[4.8.3 Impostazioni dell’applicazione 39](#_Toc41608928)

[4.8.4 Gestione utenti 41](#_Toc41608929)

[4.9 Pagina gestione corsi 42](#_Toc41608930)

[4.9.1 Gestione corsi 42](#_Toc41608931)

[4.9.2 Gestione svolgimenti 45](#_Toc41608932)

[4.10 Ricerca corsi 52](#_Toc41608933)

[4.11 Pagina iscrizione 53](#_Toc41608934)

[4.11.1 reCAPTCHA v2 Google 54](#_Toc41608935)

[4.12 Pagina personale 56](#_Toc41608936)

[4.13 Pagina errore 56](#_Toc41608937)

[5 Test 57](#_Toc41608938)

[5.1 Protocollo di test 57](#_Toc41608939)

[5.2 Risultati test 65](#_Toc41608940)

[5.3 Mancanze/limitazioni conosciute 70](#_Toc41608941)

[6 Consuntivo 71](#_Toc41608942)

[7 Conclusioni 73](#_Toc41608943)

[7.1 Sviluppi futuri 73](#_Toc41608944)

[7.2 Considerazioni personali 73](#_Toc41608945)

[8 Bibliografia 74](#_Toc41608946)

[8.1 Sitografia 74](#_Toc41608947)

[8.2 Indice delle figure 74](#_Toc41608948)

[8.3 Glossario 75](#_Toc41608949)

[9 Allegati 76](#_Toc41608950)

# Introduzione

## Informazioni sul progetto

**Titolo:** Gestione web corsi aggiornamento

**Allievo**: Mattia Toscanelli, impiegato nello svolgimento del progetto. (contatto: mattia.toscanelli@samtrevano.ch)

**Classe**: I4AC

**Superiore professionale**: Massimo Sartori (contatto: massimo.sartori@edu.ti.ch)

**Perito 1:** Gianluca Costante (contatto: gianluca.costante@gmail.com)

**Perito 2:** Roberto Guidi (contatto: roberto.guidi.86@gmail.com)

**Sezione scuola**: Scuola Arti e Mestieri Trevano, Informatica

**Data inizio**: 11.05.2020

**Data fine**: 29.05.2020

**Durata progetto:** 80 ore

## Abstract

*This document contains the documentation for the realization of the web software for the management of updating courses provided by a training company. In this application there are 4 types of users. Users not registered to the site can view the homepage, subscribe to courses and view a contact page. Registering users can perform the same operations as an unregistered user and in addition can view the courses done and those still to be done. Teachers can only be set to teach a course. Finally, there are page administrators who can edit photos and description of the home page, edit the contact page, create and manage courses and send news email.*

*This document is divided into 5 parts:*

* *The first part described is the analysis phase. This phase describes the problem in general, the purpose of this project, the requirements and the means used. The objective of this project is to have a software that manages the updating courses of a company.*
* *The second phase of the project is design. In this phase the flow diagram, the database used, the various interface designs and finally the UML schema of the most important classes are shown.*
* *The third phase is implementation. In this phase the whole process of how the project is carried out is described. The code used and the various functionalities are described.*
* *The fourth phase is the test phase. In this phase, the pre-established tests of the project are carried out. The tests have been created according to the requirements of the customer.*
* *The fifth and final phase is the conclusion. This phase describes the concepts that were learned during the course of the project. It also describes possible improvements that could be made to make the project even easier.*

## Scopo

Lo scopo di questo progetto è quello di realizzare un sito web che permetta di gestire dei corsi di aggiornamento erogati da un’azienda formatrice. In poche parole bisognerà gestire tutto il processo di creazione dei corsi e dell’iscrizioni da parte dei corsisti. Il sito web è accessibile da chiunque, con la possibilità di visualizzare l’homepage e i corsi in programma. Per ogni corso è possibile effettuare un’iscrizione per un numero massimo di iscrizioni, fornendo i propri dati personali. Inoltre è possibile specificare se il partecipante al corso vuole iscriversi anche per il pranzo in comunque. Infine è presente anche una scheda pdf che riassume tutte le specifiche di un determinato corso. L’utente che si iscrive a un corso potrà registrarsi al sito, così da visualizzare tutti i corsi eseguiti in precedenza o ancora da fare, oppure può decidere di non registrarsi. Infine sono presenti uno o più gestori della pagina i quali possono modificare la homepage, modificare la pagina di contatti inviare e-mail di pubblicità, creare e gestire i corsi di aggiornamento, gestire le iscrizioni e modificare le impostazioni del sito.

# Analisi

## Analisi del dominio

Bisogna trovare un metodo che permetta di gestire dei corsi di aggiornamento erogati da un’azienda formatrice. L’obiettivo di questo progetto è dunque quello di creare un applicativo web che permetta agli utenti che accedono al sito di poter trovare il proprio corso ed iscriversi. Sul mercato esistono già dei prodotti simili a questo progetto che svolgono sì e no le stesse operazioni. L’idea di questo progetto è di realizzare un prodotto più facile da utilizzare rispetto a quelli già presenti, in poche parole deve essere di facile comprensione anche al meno esperto di informatica. In questa applicazione sono presenti 4 tipologie di utenti: gli utenti non registrati, gli utenti registrati, i docenti e gli amministratori.

Gli utenti non registrati hanno la possibilità di visualizzare la homepage, la pagina di contatti e di iscriversi ai corsi proposti. Per iscriversi ad un corso l’utente dovrà compilare una serie di dati personali. Una volta iscritto riceverà una e-mail di informazione in cui gli verrà detto che si è iscritto al corso.

In seguito ci sono gli utenti registrati che hanno le stesse funzioni degli utenti non registrati, ma con la differenza che quando si vogliono iscrivere ad un corso non devono ogni volta inserire i propri dati. Inoltre avranno accesso ad una pagina personale in cui possono visualizzare/gestire i propri corsi.

I docenti non hanno una vera e propria funzione nel sito, essi possono essere selezionati come insegnanti di uno svolgimento di un corso.

Infine ci sono gli amministratori della pagina che hanno la possibilità di gestire la homepage, la pagina di contatti, i corsi con le corrispettive iscrizioni e gestire le impostazioni.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Pagina di login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Tutti gli utenti che vogliono accedere devono aver effettuato almeno un’iscrizione ad un corso e aver salvato l’account. |
| **002** | Ci deve essere la possibilità di cambiare la password in caso di smarrimento. |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Pagina principale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | La pagina deve mostrare delle foto, un testo di descrizione e i corsi disponibili. |
| **002** | Solo gli amministratori della pagina possono modificare le foto e il testo della pagina principale. |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Pagina di contatti |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | La pagina deve mostrare i contatti dei vari dipendenti dell’azienda. |
| **002** | Solo gli amministratori della pagina possono modificare i dati di questa pagina. |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Pagina singola di un determinato corso. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Ogni corso deve presentare le seguenti informazioni:  Titolo, descrizione, durata, orario, numero massimo di partecipanti, prezzo, via, città, CAP, termine di iscrizione e tipo di attestato rilasciato. |
| **002** | Ogni corso può presentare le seguenti informazioni:  Materiale necessario e pranzo. |
| **003** | Qualsiasi utente può iscriversi ad un corso. |
| **004** | Ogni iscrizione ad un corso viene inviata una email informativa. |
| **005** | Ogni corso deve avere una scheda riassuntiva scaricabile. |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Pagina personale utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Devono essere presenti i corsi fatti nel passato. |
| **002** | Devono essere presenti i corsi che verranno svolti nel futuro, con la possibilità di disdire il corso. |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Pagina gestione corsi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Si possono aggiungere/modificare/eliminare corsi. |
| **002** | Ci deve essere la possibilità di replicare il corso per una data futura. |
| **003** | Si possono visualizzare tutti gli svolgimenti. |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Pagina gestione svolgimenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Devono essere mostrate tutte le iscrizioni al corso, con la possibilità di confermare o rimuovere le iscrizioni. |
| **002** | Si può aggiungere un’iscrizione per un utente che ha chiamato per telefono |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Pagina di amministrazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Questa pagina è accessibile solamente dall’amministratore. |
| **002** | Ci deve essere una sezione per la modifica della pagina principale e della pagina di contatti. |
| **003** | Ci deve essere la possibilità di inviare delle email per la newsletter. |
| **004** | Ci deve essere la possibilità di modificare le impostazioni del sito. |

## Use case

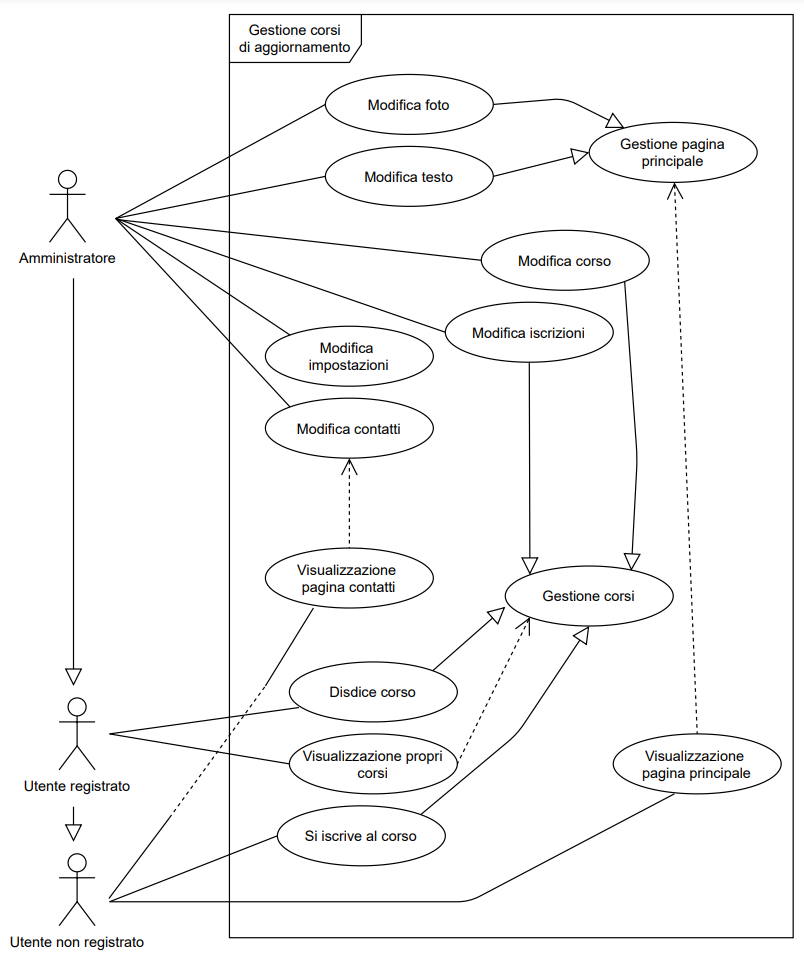


Figura 1 Use case

Lo schema inizia con tre tipologie di scenari: Amministratore (cioè colui che gestisce tutta l’applicazione), Utente registrato e Utente non registrato.

L’utente non registrato ha la possibilità di accedere alla pagina principale, dove gli vengono mostrate delle foto e una breve descrizione del sito. In seguito può accedere alla pagina di visualizzazione dei contatti dove può vedere i vari dipendenti dell’azienda. Infine può visualizzare e iscriversi ad dei corsi.

Per diventare un utente registrato bisogna aver effettuato almeno un’iscrizione ad un corso. Questo utente implementa tutte le funzioni di un utente non registrato e in più ha l’accesso ad una pagina dove può visualizzare tutti i corsi eseguiti e/o da fare. Inoltre ha la possibilità di disdire un corso da fare.

L’ultimo scenario è l’amministratore, esso ha la possibilità di accedere ad una pagina riservata dove può gestire la pagina principale, quindi modificare le foto e modificare il testo di descrizione del sito. Inoltre ha la possibilità di modificare la pagina di contatti attraverso un text editor. Può anche aggiungere, modificare o eliminare un corso e di gestire le corrispettive iscrizioni. Infine può modificare le impostazioni dell’applicazione.

## Pianificazione



Figura 2 Gantt Preventivo Intero

Il Gantt si suddivide in 4 fasi di lavoro (Analisi, Progettazione, Implementazione e Testing) e 1 fase di ricapitolazione.

### Analisi



Figura 3 Gantt Preventivo – Analisi

L’analisi si suddivide in 6 attività:

* Lettura dei requisiti 🡪 la lettura del diario dei compiti consegnato dal responsabile.
* Colloquio con il cliente 🡪 la risoluzione dei dubbi con il responsabile per quanto riguarda il diario dei compiti.
* Analisi del dominio 🡪 riflessione se ci sono dei prodotti simili sul mercato e se c’è seriamente bisogno di intraprendere questo progetto.
* Analisi dei requisiti 🡪 i requisiti che deve avere necessariamente applicazione.
* Use Case 🡪 la progettazione dello schema Use Case.
* Gantt 🡪 La stesura del Gantt preventivo e la sua correlativa descrizione.

Per le attività di analisi ho previsto che sono necessari 8 ore di lavoro.

### Progettazione



Figura 4 Gantt Preventivo – Progettazione

La progettazione si suddivide in 3 attività e un punto cardine:

* Design architettura 🡪 come funziona l’applicazione e come sono collegate le varie operazioni l’una fra l’altra.
* Design database 🡪 come è composto il database, schema E-R con correlativa descrizione.
* Design interfacce 🡪 come si presentano all’utente finale le varie interfacce grafiche.
* Fine progettazione 🡪 punto conclusivo della progettazione per dare inizio all’implementazione.

Ho previsto che per svolgere queste 3 attività mi sono necessari 8 ore di lavoro.

### Implementazione

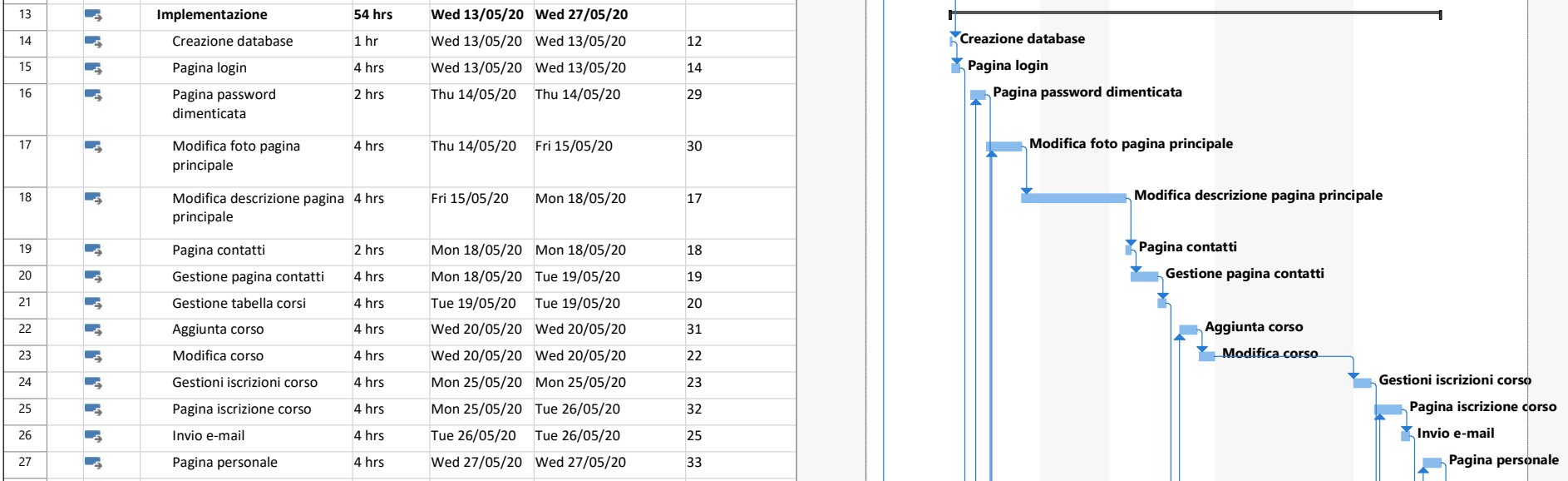


Figura 5 Gantt Preventivo – Implentazione

L’implementazione è la fase del progetto che richiede più tempo per essere svolta e si suddivide in 14 attività e un punto cardine:

* Creazione database 🡪 scrittura del codice SQL basato sullo schema E-R eseguito nella fase di progettazione
* Pagina login 🡪 creazione interfaccia grafica e back-end della pagina di login per utenti e amministratori.
* Pagina password dimenticata 🡪 creazione interfaccia grafica e back-end della pagina per recuperare la password.
* Modifica foto pagina amministrazione 🡪 aggiunta/modifica/eliminazione foto della pagina principale.
* Modifica descrizione pagina amministrazione 🡪 modifica del testo di descrizione del sito della pagina principale.
* Pagina contatti 🡪 Interfaccia grafica per vedere i contatti delle persone. (Es. direttore, presidente, …)
* Gestione pagina contatti 🡪 modifica della pagina di contatti.
* Gestione tabella corsi 🡪 tabella presente nella pagina di amministrazione per accedere alla gestione di un determinato corso.
* Aggiunta corso 🡪 Form di aggiunta di un corso con validazione lato server.
* Modifica corso🡪 Form di modifica di un corso con validazione lato server.
* Gestione iscrizione corso 🡪 Tabella per gestire le iscrizioni degli utenti di un corso
* Pagina iscrizione corso 🡪 Interfaccia grafica e back-end per l’iscrizione ad un corso da parte dell’utente.
* Invio E-mail 🡪 Invio delle email informative di iscrizione ad un corso.
* Pagina personale 🡪 Interfaccia grafica per visualizzare i corsi dove si è iscritti o i corsi già eseguiti.

Per completare la fase di implementazione ho calcolato che ci vogliono circa 54 ore lavorative.

### Testing

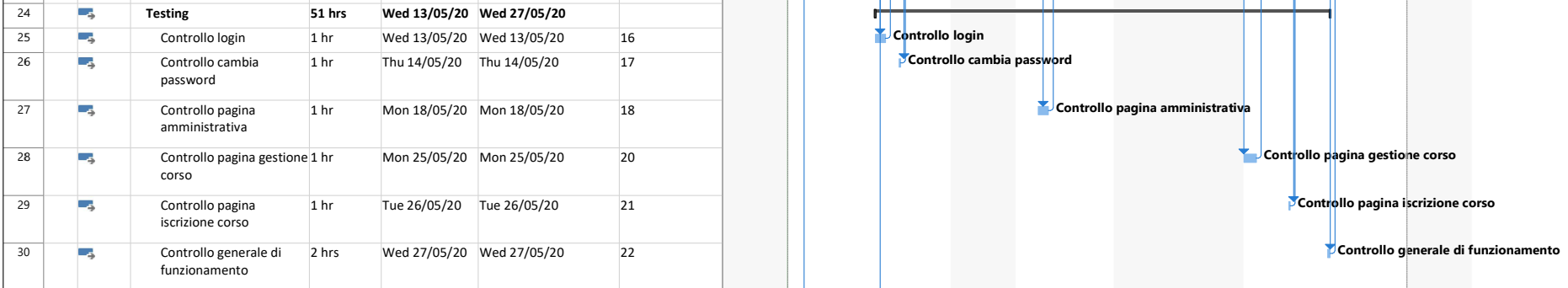


Figura 6 Gantt Preventivo - Testing

La fase di testing si suddivide in 6 attività:

* Controllo login 🡪 controllo per verificare che il login funzioni.
* Controllo cambia password 🡪 controllo per far sì che possa modificare la password.
* Controllo pagina amministrativa 🡪 controllo per verificare le funzionalità della pagina amministrativa, cioè della modifica della pagina home, della pagina dei contatti.
* Controllo pagina gestione corso 🡪 controllo per verificare l’aggiunta, la modifica e l’eliminazione di un corso.
* Controllo pagina iscrizione corso 🡪 controllo per verificare l’iscrizione ad un corso.
* Controllo generale di funzionamento 🡪 controllo globale di tutte le funzionalità dell’applicazione.

Ho previsto che per svolgere queste 6 attività sono necessari circa 7 ore di lavoro suddivise in tutto l’arco del progetto.

### Documentazione

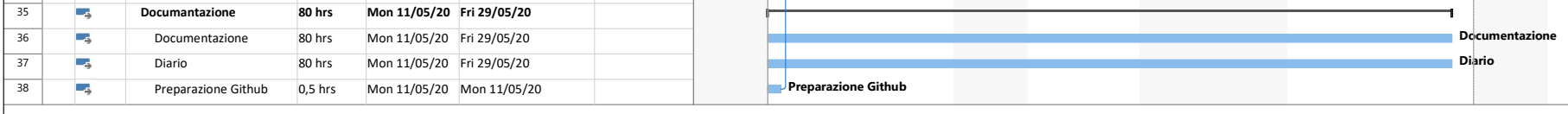
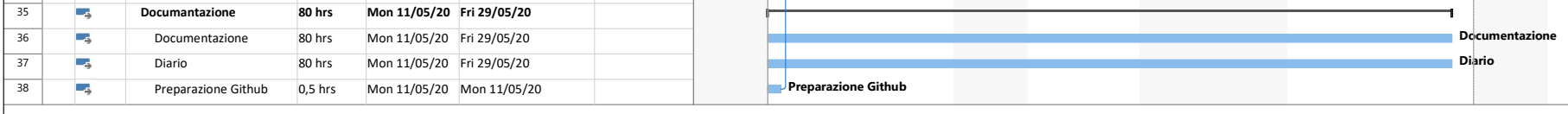


Figura 7 Gantt Preventivo - Documentazione

La fase riassuntiva si suddivide in 7 lavori e un punto cardine:

* Documentazione 🡪 scrittura della documentazione del progetto.
* Diario 🡪 scrittura dei lavori effettuati giornalmente.
* Preparazione di Gitub 🡪 preparazione dell’ambiente su Github interno per mostrare l’andamento del progetto.

Questa fase di documentazione dura per tutto lo svolgimento del progetto.

## Analisi dei mezzi

Per la realizzazione di questo progetto si ha bisogno di:

* Un PC con performance in grado di girare Windows 10 e un Web Server Apache.

### Software

I programmi utilizzati per svolgere questo progetto sono:

* Word 2016 🡪 per la scrittura della documentazione.
* PowerPoint 2016 🡪 per la presentazione.
* Draw.io 13.0.3 🡪 per i disegni dei vari schemi e progettazioni.
* PhpStorm 2019.3.1 🡪 per lo sviluppo web.
* Apache 2.4 🡪 per il Web Server.
* PHP 7.1.9 🡪 per l’utilizzo del linguaggio PHP.
* MySQL 8.0.20 🡪 Salvataggio di dati.
* Project 2016 🡪 Gantt Preventivo e Gantt Consuntivo.
* GitHub Desktop 2.4.3 🡪 per commit e push in formato grafico sulla repo scolastica.

### Hardware

Il pc su cui è stato sviluppato il progetto ha le seguenti caratteristiche:

* OS: Windows 10 Home
* CPU: i7-6700K
* RAM: 16,00GB
* Scheda video: Nvidia 980 Ti

### Librerie

Le librerie ho utilizzerò sono le seguenti:

* MeekroDB 2.3 🡪 Tutte le operazioni su database
* PHPMailer 5.5 🡪 Invio delle e-mail
* DataTables 1.10.21 🡪 Tabelle responsive
* Summernote 0.8.16 🡪 Text area editor
* ReCAPTCHA v2 1.2.4 (API) 🡪 Evitare attacchi bot

### Template HTML

Per lo svolgimento del sito utilizzo un template che si chiama “srtdash” che mi aiuta a sviluppare le immagini in modo responsive. Esso utilizza Bootstrap 4 come framework grafico.

# Progettazione

## Design dell’architettura del sistema

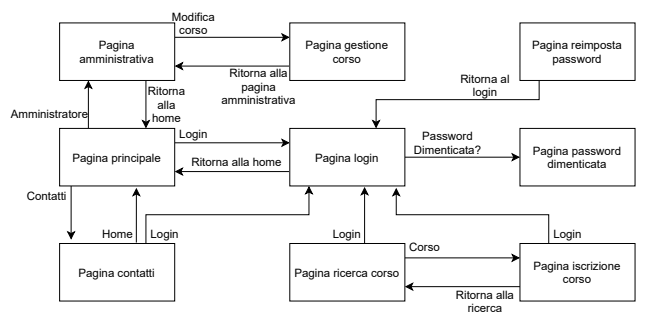


Figura 8 Design dell'architettura del sistema

Prima di spiegare la struttura dell’applicazione avviso che nello schema non sono presenti tutti i possibili accessi alle pagine, in quanto nel sito è presente una navbar per accedere a quasi tutte le pagine e lo schema sarebbe diventato troppo confusionario.

La pagina di partenza della mia applicazione, cioè la pagina iniziale a cui si collega l’utente finale per utilizzare il mio sito, è la pagina principale. In questa pagina l’utente può farsi un’idea di quello che dispone tutto il sito, infatti può visualizzare delle immagini e leggere una breve descrizione di quello che offre l’azienda. In questa pagina, come nella maggior parte delle altre, sarà possibile effettuare un accesso cliccando il bottone “Login” in una navbar in alto. Nella pagina di login si può naturalmente fare il login inserendo le credenziali di accesso (e-mail e password) oppure si può modificare la propria password cliccando il bottone “Hai dimenticato la password?”. Quando viene richiesta la procedura di modifica password tramite la pagina di password dimenticata viene inviata una e-mail, all’utente diretto interessato, la quale contiene un link di recupero. Cliccando questo link l’utente verrà trasportato sulla pagina reimposta password dove esso potrà inserire una nuova password.

Ritornando alla pagina principale si può accedere alla pagina di ricerca corsi. In questa pagina sarà presente un filtro che permette di catalogare i corsi in base alla loro tipologia.

Cliccando su un singolo corso, l’utente potrà visualizzare tutte le informazioni di un determinato corso con la possibilità di stampare un pdf riassuntivo. Inoltre avrà la possibilità di iscriversi al corso inserendo i propri dati oppure, se l’utente ha effettuato l’accesso, utilizzare i propri dati salvati da una precedente iscrizione. Una volta iscritto l’utente riceverà una breve email informativa di iscrizione al corso.

Se l’utente che ha effettuato il login è un admin, esso potrà accedere alla pagina amministrativa dove potrà gestire la pagina principale e la pagina di contatti. Inoltre avrà la possibilità di accedere alle pagine per la gestione dei singoli corsi, svolgimenti e iscrizioni.

## Design dei dati e database

### Schema E-R

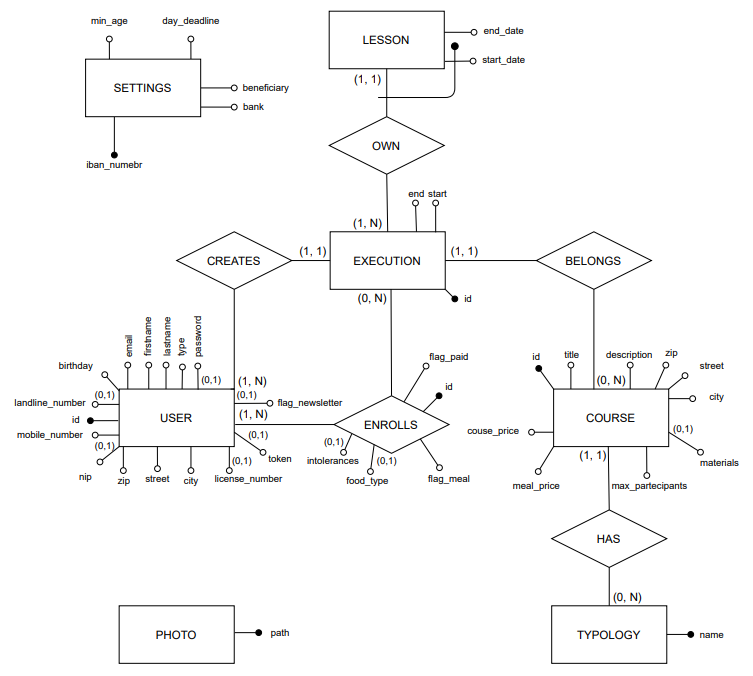


Figura 9 Diagramma E-R

### Schema logico

user(id, email, firstname, lastname, birthday, zip, city, street, mobile\_number, flag\_newsletter, landline\_number\*, password\*, token\*, type\*, nip\*, license\_number\*)

typology(name)

photo(path)

course(id, title, description, zip, street, city, max\_partecipants, materials, name\_typology (FK), meal\_price, course\_price)

execution(id, id\_user (FK), id\_course (FK), start, end)

lesson(start\_date, id\_execution (FK), end\_date)

enrolls(id, id\_user (FK), id\_execution(FK), intolerances\*, food\_type\*, flag\_meal)

settings(iban\_number, registration\_deadline, beneficiary, day\_deadline, min\_age)

### Vincoli

Nella tabella Utente l’attributo “type” specifica il livello di gerarchia dell’utente. I livelli sono 3 e sono i seguenti:

* 0: utente che ha effettuato un’iscrizione ad un corso ma che non ha voluto registrarsi al sito.
* 1: utente che ha effettuato un’iscrizione ad un corso e che ha deciso di creare un account per salvare i propri dati.
* 2: docente, colui che svolge i corsi (tipo di utente disponibile ma non le ha funzioni nel sito).
* 3: amministratore del sito, cioè colui che può effettuare tutte le operazioni di amministrazione dell’applicazione (creare corsi, gestire la pagina, ecc.)

Ci sono dei vicoli da rispettare per la tabella Utente:

* Il campo “type” può contenere i valori 0, 1 e 3.
  + Se “type” = 0 o “type” = 1, non può partecipare alla relazione “creates” ma può partecipare alla relazione “enrolls”.
  + Se “type” = 3 o “type = 2, può partecipare alle relazioni “creates” e “enrolls”.
* Se “type” = 1, “type” = 2 o “type” = 3 non ci possono essere utenti con lo stesso valore “email”.

C’è anche un piccolo vincolo per quanto riguarda la tabella “lesson” ed è il seguente:

* L’intervallo di tempo fra “start\_date” e “end\_date” deve essere un numero positivo.

Per quanto riguarda i dati ci sono delle piccole regole da gestire:

* Tutti i campi di testo devono avere almeno 3 lettere.
* I campi flag devono avere i valori “0” o “1”.
* Numeri di telefono devono avere almeno 10 numeri.
* Numero nip è composto da 6 numeri.
* Numero di licenza è composto da 12 numeri.
* Il cap è composto da 4 numeri.
* Password deve avere almeno 8 caratteri tra cui deve essere presente un numero o carattere speciale. (Naturalmente nel database verranno inserite le password cifrate quindi sarà da gestire lato server).

### Foreign key e utenti

L’utente utilizzato per le query sul database ha le seguenti credenziali:

* Nome: courses\_management\_admin
* Password: CoursesManagement&1

Questo utente ha i permessi di scrittura e lettura su tutte le tabelle del database “courses\_management”.

Nella tabella “execution” ci sono due foreign key. Se si elimina uno “user” la modifica non si propaga nella tabella “execution”, invece se si elimina un “course” vengono eliminati tutti i riferimenti nella tabella “execution”.

Nella tabella “enrolls” ci sono due foreign key. Se si elimina uno “user” oppure una “execution” vengono eliminati tutti i rifermenti nella tabella “enrolls”.

Nella tabella “course” c’è una foreign key. Se si elimina una “tipology” vengono eliminati tutti i riferimenti nella tabella “course”.

Infine nella tabella “lesson” c’è una foreign key. Se si elimina una “execution” vengono eliminati tutti i riferimenti nella tabella “execution”

### Descrizione tabelle dati

#### user

Tabella per gestire gli utenti del sito:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| id | Identificativo utente. | INTEGER |
| email | L’email dell’utente. | VARCHAR(100) |
| firstname | Il nome dell’utente. | VARCHAR(50) |
| lastaneme | Il cognome dell’utente | VARCHAR(50) |
| birthday | La data di nascita dell’utente. | DATE |
| zip | Il codice di avviamento postale dell’utente. | INTEGER |
| city | La città dell’utente. | VARCHAR(50) |
| street | La via dell’utente. | VARCHAR(50) |
| mobile\_number | Il numero di telefono monile dell’utente. | VARCHAR(15) |
| flag\_newsletter | Flag per sapere se l’utente si è iscritto alla newsletter. | TINYINT(1) |
| landline\_number | Il numero di telefono fisso dell’utente. | VARCHAR(15) |
| password | La password dell’utente. | VARCHAR(255) |
| token | Il codice identificativo per cambiare la password. | VARCHAR(255) |
| type | Il livello di gerarchia dell’utente. | INTEGER |
| nip | Il numero nip dell’utente. | VARCHAR(9) |
| license\_number | Il numero di licenza dell’utente. | VARCHAR(12) |

#### typology

Tabella per gestire le tipologie di un corso:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| name | Tipologia di un corso. | VARCHAR(50) |

#### photo

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| path | Il percorso della foto | VARCHAR(100) |

#### course

Tabella per gestire il contenuto di un corso:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| id | Identificativo di un corso. | INTEGER |
| title | Il titolo di un corso. | VARCHAR(100) |
| description | La descrizione di un corso. | TEXT |
| zip | Il codice di avviamento postale dove si svolge il corso. | INTEGER |
| street | La via dove si svolge il corso. | VARCHAR(50) |
| city | La città dove si svolge il corso. | VARCHAR(50) |
| max\_partecipants | Il numero massimo di partecipanti di un corso | INTEGER |
| materials | Il materiale necessario per eseguire il corso. | TEXT |
| name\_typology | La tipologia del corso. | VARCHAR(50) |
| meal\_price | Il prezzo del pasto. | FLOAT(5,2) |
| course\_price | Il prezzo del corso. | FLOAT(6,2) |

#### execution

Tabella per gestire lo svolgimento di un corso:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| id | Identificativo dello svolgimento di un determinato corso. | INTEGER |
| id\_user | L’id dell’utente che ha creato il corso. | INTEGER |
| id\_course | L’id del contenuto del corso. | INTEGER |
| start | La data di inizio svolgimento. | DATE |
| end | La data di fine svolgimento. | DATE |

#### lesson

Tabella per gestire le lezioni di un corso:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| start\_date | La data e ora dell’inizio della lezione. | DATETIME |
| end\_date | La data e ora della fine della lezione. | DATETIME |
| id\_execution | L’id dell’esecuzione di un corso. | INTEGER |

#### enrolls

Tabella per gestire le iscrizioni ad un corso:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| id | Identificativo di un’iscrizione | INTEGER |
| id\_user | L’id dell’utente che si vuole iscrivere ad un corso. | INTEGER |
| id\_execution | L’id del corso dove ci si vuole iscrivere. | INTEGER |
| Intolerances | Le intolleranze di un utente. | TEXT |
| food\_type | La preferenza di cibo dell’utente (Vegetariano, Vegano...) | VARCHAR(50) |
| flag\_meal | Flag per sapere se l’utente vuole partecipare al pranzo. | TINYINT(1) |
| flag\_paid | Flag per sapere se l’iscrizione è stata pagata. | TINYINT(1) |

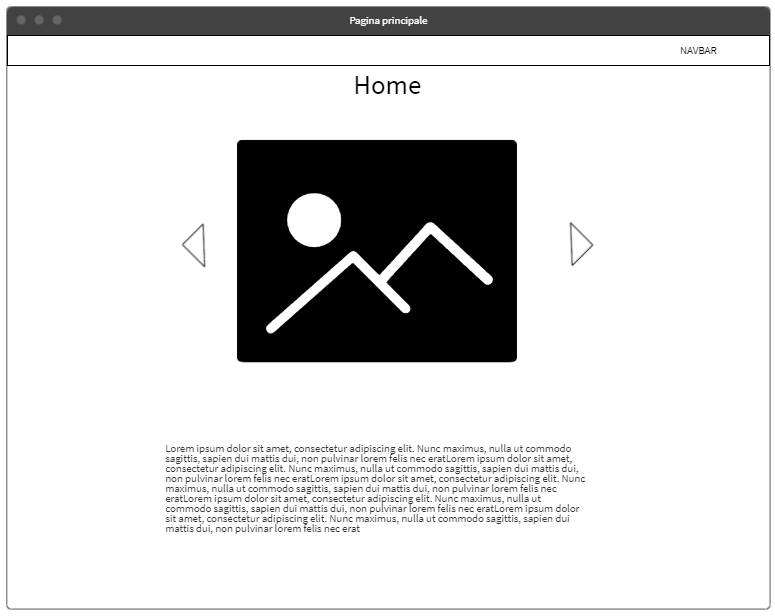
#### Settings

Tabella per gestire le informazioni generali dell’azienda:

|  |  |  |
| --- | --- | --- |
| Nome campo | Descrizione | Tipo |
| iban\_number | Il numero iban dove inviare i soldi del pagamento. | VARCHAR(50) |
| bank | La banca dove inviare i soldi del pagamento. | VARCHAR(50) |
| beneficiary | Il beneficiario dove inviare i soldi del pagamento. | VARCHAR(50) |
| day\_deadline | Il numero di giorni prima dell’inizio del corso. Specifica quando l’iscrizione al corso non è più disponibile. | INTEGER |
| min\_age | L’età minima per iscriversi ad un corso | INTEGER |

## Design delle interfacce

### Pagina principale



Questa è la pagina iniziale a cui l’utente finale si collega. In questa pagina si ha la possibilità di visualizzare delle immagini e una breve descrizione di cosa contiene il sito. Al di sopra della pagina si può trova una navbar. Attraverso questa navbar si potrà accedere alla pagina di ricerca corsi e alla pagina di contatti.

Figura 10 Design - Pagina principale

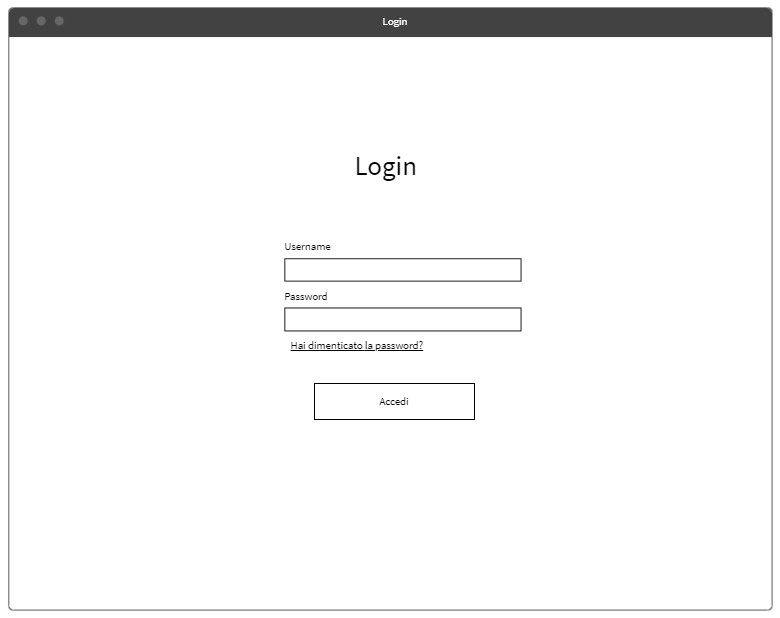
### Pagina contatti



Figura 11 Design - Pagina contatti

Questa è la pagina per vedere i contatti dei dipendenti dell’azienda.

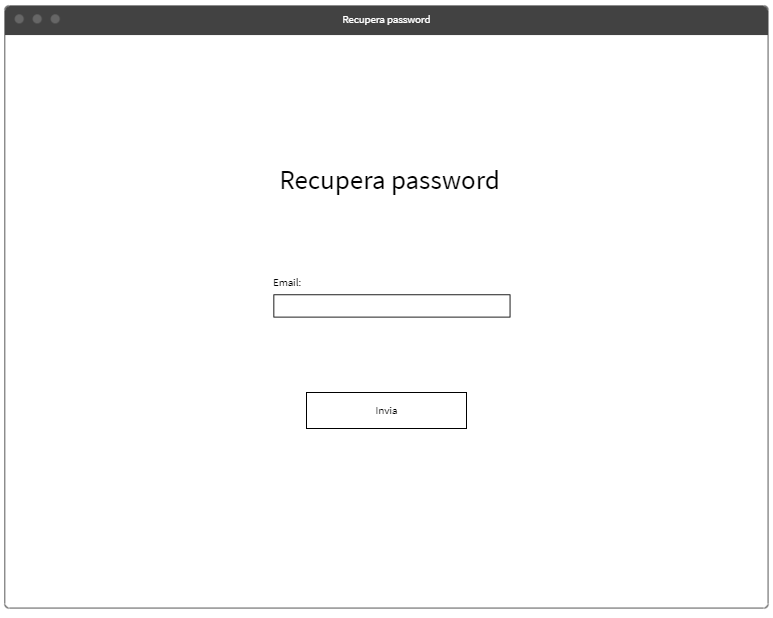
### Pagina di login



Questa è la pagina di login in cui l’utente ha la possibilità di effettuare l’accesso inserendo le proprie credenziali. Inoltre in caso che l’utente avesse dimenticato la password si può cliccare il link “Hai dimenticato la password?” e si avvia una procedura di ripristino password.

Figura 12 Design - Pagina di login

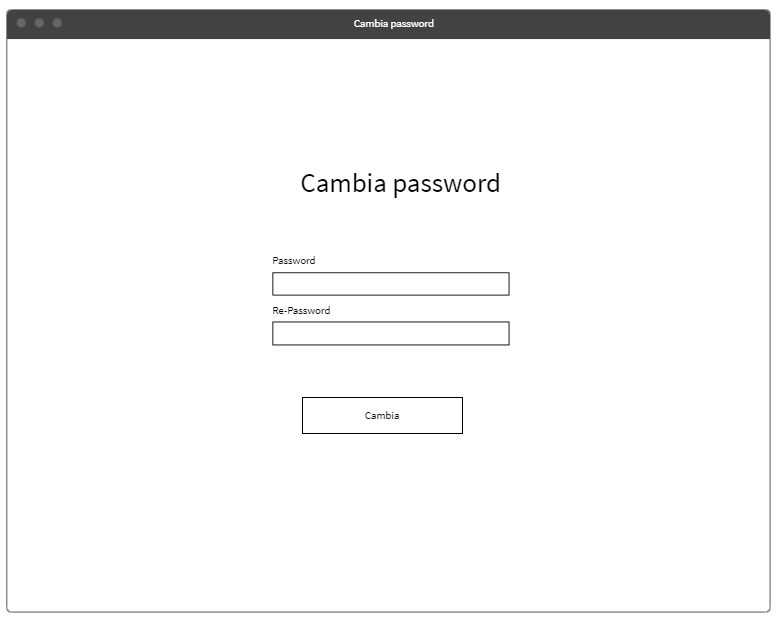
### Pagina richiesta nuova password



Questa pagina permette di richiedere una procedura per il ripristino della password. L’utente finale raggiunge questa pagina cliccando il link “Hai dimenticato la password?” dalla pagina di Login. Inserendo la propria mail e cliccando il bottone “Invia” si potrà avviare la procedura.

Figura 13 Design - Pagina richiesta recupero password

### Pagina reimposta password



Una volta inviata la richiesta di recupero password, arriverà una e-mail con un link che riporterà a questa pagina. In questa pagina si dovrà inserire due volte la nuova password di accesso. Una volta riempiti i due campi basterà cliccare il bottone “Cambia” e il cambiamento verrà applicato.

Figura 14 Design - Pagina reimposta password

### Pagina amministrativa

Figura 15 Design - Pagina amministrativa

Questa è la pagina di amministrazione del sito. Questa non può essere visualizzata se non dagli amministratori dell’applicazione.

Nella prima parte della pagina è possibile effettuare la modifica della pagina principale, scorrendo è possibile modificare la pagina di contatti. Inoltre è possibile modificare le impostazioni del sito e inviare delle email di newsletter.

### Pagina ricerca corsi



Questa è la pagina per ricercare un appartamento. Grazie ad una select all’inizio della pagina è possibile filtrare i corsi in base alla sua tipologia. Cliccando su un corso si accede alla sua corrispettiva pagina di iscrizione.

Figura 16 Design - Pagina ricerca appartamenti

### Pagina di iscrizione ad un corso

Questa è la pagina per iscriversi ad un corso, nella prima parte della pagina sono presenti tutte le varie informazioni. Scorrendo la pagina è possibile inserire i propri dati per iscriversi a quel determinato corso. In caso l’utente ha effettuato l’accesso tutti i campi saranno pre-compilati.

Infine sarà possibile scegliere le date del corso in caso che quest’ultimo venga ripetuto più volte.

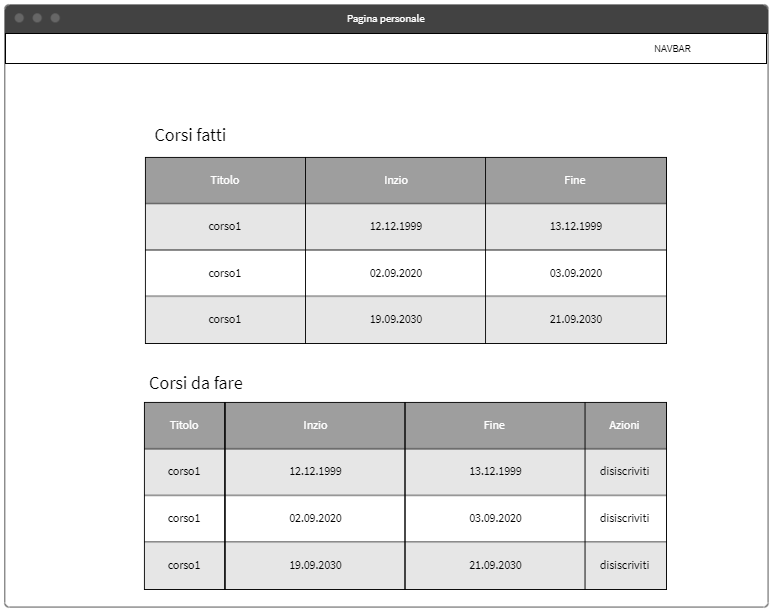
Figura 17 Design - Pagina iscrizione corso

### Pagina gestione corso

Questa pagina è accessibile solo agli amministratori del sito. In questa pagina è possibile modificare i dati di un corso e gestire i vari svolgimenti di quest’ultimo attraverso la tabella in fondo alla pagina. Per ogni svolgimento si potranno aggiungere/  
visualizzare/rimuovere iscrizioni oppure eliminare un corso.

Figura 18 Design - Pagina gestione corso

### Pagina personale



Questa pagina è accessibile solamente dagli utenti registrati al sito. In questa pagina è possibile vedere i corsi già fatti e i corsi ancora da fare.

Figura 19 Design - Pagina personale

## Design procedurale

### Classe Database

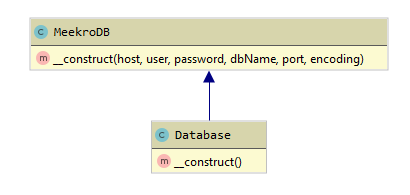


Figura 20 UML - Classe Database

Questa classe è utilizzata per eseguire l’accesso con il database tramite PHP. Essa presenta solamente un costruttore senza parametri.

Il metodo costruttore richiama il metodo costruttore padre (infatti estende la classe MeekroDB) e gli passa 4 parametri:

* host 🡪 l’host dove gira il database (nel mio caso è “localhost”).
* user: 🡪 il nome utente per eseguire l’accesso a MySQL.
* password 🡪 la password corrispondente all’utente che esegue l’accesso.
* dbName 🡪 il nome del database dove verranno eseguite le operazioni.

Come detto poco fa la classe Database estende la classe MeekroDB e dunque tutti i suoi metodi. MeekroDB è una libreria di PHP che si occupa di gestire tutte le operazioni in MySQL, permette di risparmiare righe di codice e garantisce la sicurezza al SQL Injection al 100%. Questa classe è utilizzabile sia in modo procedurale sia ad oggetti. Io per semplicità uso quella a oggetti. Un esempio di query (SELECT) fatto con questa classe è questo:

DB::query("SELECT \* FROM tbl WHERE name=%s AND age > %i AND height <= %d", $name, $age, $height)

Il metodo query() è, come già dice il suo nome, il metodo che mi permette di fare le query al database. Il primo parametro è obbligatorio, infatti bisogna inserire la stringa di query. I parametri successivi variano in base hai “segnaposti”. In questa query se ne possono vedere 3: %s, %i e %d:

* %s 🡪 indica che bisognerà inserire una stringa.
* %i 🡪 indica che bisognerà inserire un numero intero.
* %d 🡪 indica che bisognerà inserire un numero decimale.

Grazie a questi “segnaposti” non bisogna più scrivere tutte le righe di codice i prepare statement ma sarà sufficiente scriverne solamente una. I segnaposti ce ne sono molti: %t (timestamp), %ls (elenco di stringhe), %? (rilevazione automatica), %ss (stringhe per ricerca), …

La classe offre molto di più di questo singolo metodo query() ma io non utilizzerò altro.

### Classe SendMail

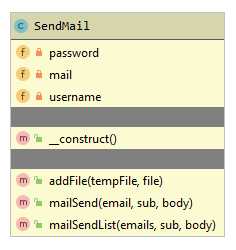


Figura 21 UML - Classe SendMail

Questa classe è utilizzata per inviare le email tramite PHP. Essa presenta tre attributi privati, un costruttore senza parametri e tre metodi di utilizzo della classe.

L’attributo mail è un oggetto di tipo PHPMailer ed è utilizzato per la configurazione e l’invio delle mail. Questo oggetto PHPMailer è una libreria che ho scaricato da GitHub che mi permette appunto di collegarmi in modo semplice ad un client di posta elettronica attraverso un account creato da me dedicato. In seguito utilizza questo account per inviare le email in modo autonomo.

Ritornando alla classe SendMail nel costruttore definisco tutti i vari parametri per il collegamento al client di posta elettronica, come per esempio l’host, la porta, le credenziali d’accesso, ….

Il metodo mailSend() mi permette di inviare le e-mail. Ha tre parametri: il primo definisce la e-mail a chi devo inviare il messaggio, il secondo mi definisce l’oggetto della email e il terzo mi definisce il contenuto della mail. Questa classe verrà utilizzata per inviare link di recupero password, le email di iscrizione e le email di news. Un esempio di codice è il seguente:

mailSend($email, "Oggetto", $body)

* $email 🡪 la e-mail dell’utente a cui devo inviare il messaggio.
* “Oggetto” 🡪 Oggetto della e-mail.
* $body 🡪 il contenuto della e-mail.

Il metodo mailSendList() è uguale al metodo mailSend() ma con la differenza che al posto di passare una singola email, viene passato un array di email, così da inviare un messaggio comune a più utenti.

Infine il metodo addFile() mi permette di allegare dei file alla e-mail. Ha due parametri: il primo specifica la posizione del file nella cartella temporanea e il secondo specifica il nome del file.

### Classe Util

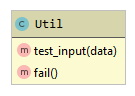


Figura 22 UML - Classe Util

Questa classe è utilizzata per facilitarmi le operazioni. Più precisamente viene utilizzata eseguire dei controlli rapidi degli input passati dagli utenti attraverso le pagine e per ritornare velocemente un controllo fallito. La classe non presenta né attributi né un metodo costruttore, bensì 2 metodi statici.

I metodi statici che presenta la classe sono i seguenti:

* test\_input(string) 🡪 questo metodo viene utilizzato praticamente da tutti i controller per evitare che l’utente immetta dati, attraverso form o altro, malevoli con lo scopo di danneggiare il sistema. In pratica questo metodo rimuove caratteri speciali, spazi e slash dalle stringhe passate come parametro.
* fail() 🡪 questo metodo è utilizzato soprattutto per i metodi ajax e serve per ritornare una risposta di errore al client.

### Classe MSession

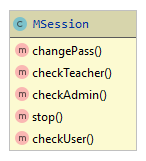


Figura 23 UML - Classe DSession

Questa classe è utilizzata per gestire le sessioni utilizzate nell’applicazione. Anche questa classe non presenta né attributi né un metodo costruttore, bensì 5 metodi statici:

I metodi statici che presenta la classe sono i seguenti:

* changePass() 🡪 Metodo per distruggere la sessione nel momento in cui l’utente sta modificando la password.
* checkTeacher() 🡪 Metodo verificare che l’utente che vuole accedere ad una pagina o ad un metodo sia un’insegnante.
* checkAdmin() 🡪 Metodo verificare che l’utente che vuole accedere ad una pagina o ad un metodo sia un admin.
* stop() 🡪 Metodo per distruggere la sessione (utilizzato per eseguire il logout).
* checkUser() 🡪 Metodo verificare che l’utente che vuole accedere ad una pagina o ad un metodo sia un utente registrato.

### Classe Validator

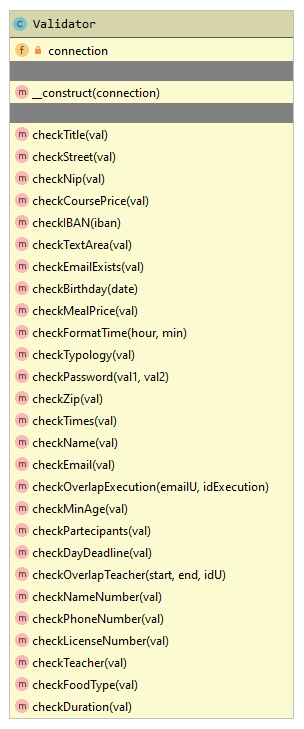


Figura 24 UML - Classe Validator

Questa classe è utilizzata per verificare tutti gli input lato server passati da form dagli utenti che utilizzano l’applicazione.

La classe ha un attributo privato, un costruttore e 27 metodi di utilizzo della classe.

L’unico attributo della classe è “connection” e definisce la connessione al database ed esso viene istanziato grazie al singolo parametro del metodo costruttore della classe.

In seguito la classe presenta i 27 metodi di controllo:

* checkTitle(string) 🡪 Metodo per verificare il titolo di un appartamento.
* checkStreet(string) 🡪 Metodo per verificare una via.
* checkNip(string) 🡪 Metodo per verificare il numero nip.
* checkCourcePrice(string) 🡪 Metodo per verificare il prezzo di un corso.
* checkIBAN(string) 🡪 Metodo per verificare un numero IBAN.
* checkTextArea(string) 🡪 Metodo per verificare il contenuto di una text area.
* checkEmailExists(string) 🡪 Metodo per verificare se una email non appartenga a nessuno nell’applicazione.
* checkBirthday(string) 🡪 Metodo per verificare la data di nascita.
* checkMealPrice(string) 🡪 Metodo per verificare il prezzo di un pasto.
* checkFromatTime(string) 🡪 Metodo per verificare il formato di un orario.
* checkTypology(string) 🡪 Metodo per verificare la tipologia di un corso.
* checkPassword(string,string) 🡪 Metodo per verificare la lunghezza della password e i criteri si sicurezza. Inoltre verifica che le due password siano identiche.
* checkZip(string) 🡪 Metodo per verificare il codice di avviamento postale.
* checkTimes(string) 🡪 Metodo per verificare un orario.
* checkName(string) 🡪 Metodo per verificare i nomi, cognomi, città, …
* checkEmail(string) 🡪 Metodo per verificare se una email è valida (nel formato testo@testo.testo) e che non appartenga a nessuno nell’applicazione.
* checkOverlapExecution(string,int) 🡪 Metodo per verificare se un utente è già iscritto in un determinato svolgimento.
* checkMinAge(string) 🡪 Metodo per verificare l’età minima per iscriversi a un corso.
* checkPartecipants(string) 🡪 Metodo per verificare il numero massimo di partecipanti a un corso.
* checkDayDeadline(string) 🡪 Metodo per verificare il numero di giorni prima dell’inizio del corso per calcolare la sua scedenza.
* checkOverlapTeacher(date, date, int) 🡪 Metodo per verificare se un docente in un certo intervallo di date è già occupato in un altro corso.
* checkNameNumber(string) 🡪 Metodo per verificare un nome che contiene anche numeri.
* checkPhoneNumber(string) 🡪 Metodo per verificare un numero di telefono.
* checkLicenseNumber(string) 🡪 Metodo per verificare il numero di licenza.
* checkTeacher(string) 🡪 Metodo per verificare un docente.
* checkFoodType(string) 🡪 Metodo per verificare la tipologia di cibo.
* checkDuration(string) 🡪 Metodo per verificare il numero di giorni di uno svolgimento.

# Implementazione

In questo capitolo verrà spiegato come ho sviluppato il progetto, dunque saranno presenti interfacce grafiche, codice, librerie, metodi di implementazione, … utilizzati con altrettante descrizioni.

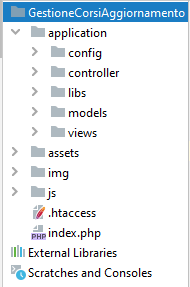
## Pattern MVC

Per lo sviluppo di questo progetto ho deciso di utilizzare il pattern MVC. MVC (Model-View-Controller) è un modello molto utilizzato al giorno d’oggi per lo sviluppo di applicazioni ed è molto utile per avere una struttura ordinata del codice. Il modello MVC che ho utilizzato ci è stata fornito dal docente Massimo Sartori. Questo modello a tre sottoinsiemi:

* Model: fornisce tutti i metodi per accedere ai dati dell’applicazione.
* View: si occupa di far visualizzare i dati all’utente e inoltre coordina le iterazioni fra quest’ultimo e l’applicazione sottostante.
* Controller: riceve i comandi dall’utente e in base a ciò modifica lo stato degli altri due componenti.

In questa documentazione verranno descritti solo i metodi dei Model, in quanto i controller nella mia applicazione si occupano solamente di inviare i dati dalle view al model.

La mia struttura è la seguente:



Nella cartella application si possono notare le 3 categorie appena spiegata e in più ci sono 2 cartelle: libs e config. Nella cartella libs vengono inserite tutte le librerie scaricate e le classi che servono per il funzionamento dell’applicazione (come quella per l’invio delle e-mail). Nella cartella config vengono inserite tutte le variabili costanti utili per gli indici delle sessioni e di array oppure per salvare dati di accesso (per esempio per accedere al database). Al di fuori della cartella application si può trovare la cartella asset, che contiene i font, i file di stile e codice Javascript del template grafico utilizzato, la cartella js, dove sono contenuti tutti i file Javascript da me creati e la cartella che contiene tutte le immagini del sito. Infine sono presenti due file: .htaccess e index.php. Il file .htaccess contiene le impostazioni della directory del progetto e il file index.php e il file di avvio di tutta l’applicazione.

Figura 25 Struttura MVC

## Creazione database

Prima di incominciare a scrivere il codice in PHP, ho scritto il database in MySQL. Il database è formato da 8 tabelle. L’unica tabella su cui voglio fare una piccola precisazione è la tabella settings:

***CREATE TABLE*** settings(  
 ***iban\_number*** VARCHAR(50) ***PRIMARY KEY***,  
 ***bank*** VARCHAR(50) ***NOT* NULL**,  
 ***beneficiary*** VARCHAR(50) ***NOT* NULL**,  
 ***day\_deadline INTEGER NOT* NULL**,  
 ***min\_age INTEGER NOT* NULL**);

Questa tabella contiene tutte le impostazioni dell’applicazione ed è composta da una singola riga. All’interno della tabella si possono trovare le informazioni di pagamento di un corso, i giorni per calcolare la data di scadenza dei vari corsi e l’età minima per iscriversi a un corso. Ho scelto di strutturare la tabella con 5 colonne anziché 2 (es. tipo, valore), così che in futuro si potrebbero aggiungere più configurazioni.

Per l’utilizzo del database ho creato un utente specifico. Per creare questo utente ho scritto le seguenti due query:

***CREATE USER* 'courses\_management\_admin'**@**'localhost' *IDENTIFIED WITH mysql\_native\_password BY* 'CoursesManagement&1'**;  
***GRANT ALL ON courses\_management***.\* ***TO* 'courses\_management\_admin'**@**'localhost'**;

In poche parole creo un utente con la password mysql\_native\_password. Questo tipo di password serve per poter accedere al database tramite PHP. In seguito imposto i permessi totali all’utente sul database dell’applicazione.

### Collegamento al database tramite PHP

Per collegarmi al database imposto innanzitutto 4 variabili costanti nel file di config:

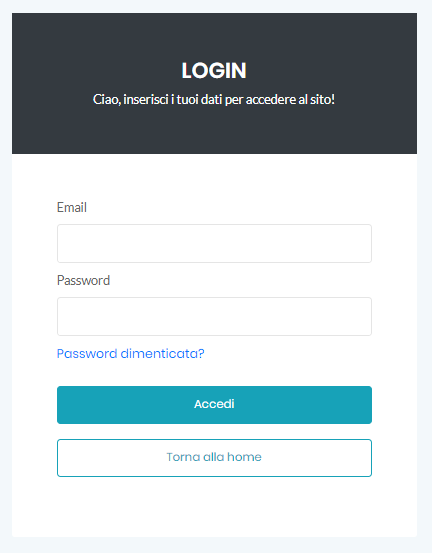
*define*(**'DB\_HOST'**, **'localhost'**);  
*define*(**'DB\_USER'**, **'courses\_management\_admin'**);  
*define*(**'DB\_PASS'**, **'CoursesManagement&1'**);  
*define*(**'DB\_NAME'**, **'courses\_management'**);

Dopodiché richiamo il costruttore della mia classe Database che ha sua richiama il costruttore padre della libreria Meekrodb (spiegata nel capitolo [3.4.1 Classe Database](#_Classe_Database)) passandogli tutte le variabili costanti configurate precedentemente.

**public function** \_\_construct()  
{  
 **parent**::*\_\_construct*(***DB\_HOST***, ***DB\_USER***, ***DB\_PASS***, ***DB\_NAME***);  
}

## Pagina di login

La prima pagina che ho sviluppato è la pagina di login. L’interfaccia grafica è la seguente:



Si tratta di una normalissima pagina di login dove l’utente che ha scelto di registrarsi dopo un’iscrizione di un corso può inserire email e password può accedere. Da questa pagina è possibile ritornare anche alla pagina home cliccando il bottone “Torna alla home” oppure cambiare la password cliccando “Password dimenticata?”.

Figura 26 Pagina di login

### Metodo di accesso al sito

Per eseguire l’accesso prendo i valori dai due input (email e password) e li invio al server tramite una richiesta ajax. Il metodo che si occupa di eseguire l’accesso è il sequente:

**public function** access($email, $password)  
{  
 $result = $this->getUser($email);  
 **if** (*count*($result) > 0) {  
 **if** (*password\_verify*($password, $result[0][***DB\_USER\_PASSWORD***])) {  
 **echo *SUCCESSFUL***;  
 **return** $result;  
 } **else** {  
 **echo *LOGIN\_DENY***;  
 }  
 } **else** {  
 **echo *LOGIN\_DENY***;  
 }  
}

In pratica controlla innanzitutto che la email dell’utente che vuole eseguire l’accesso abbia un riferimento nel database e che sia registrato al sito attraverso il metodo getUser(). In seguito controlla, con il metodo password\_verfify() di PHP, che la password sia la medesima della password salvata in hash nel database.

## Pagina password dimenticata

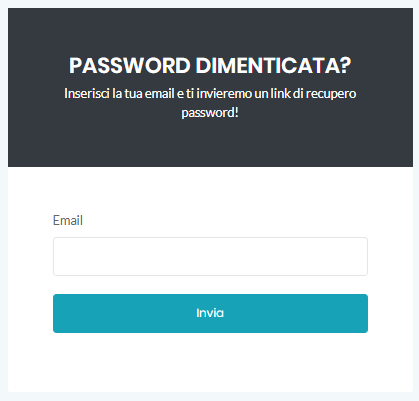


Figura 27 Pagina password dimenticata

In questa pagina è possibile richiedere la procedura per cambiare la password in caso che l’utente l’abbia dimenticata. Inserendo la propria email e cliccando il bottone “Invia” verrà ricevuto un link nel proprio client di posta elettronica dove potrà avviare la procedura di cambiamento password.

### Invio e-mail

Per inviare l’email di recupero password utilizzo il metodo sendEmail() del model ForgotPasswordModel:

$selectCheck = **"select** *\** **from user where email=%s && type <> 0"**;  
$result = $this->**connection**->query($selectCheck, $this->**email**);  
**if** ($result != **null**) {  
 **try** {  
 $token = *hash*(**'sha256'**, random\_bytes(16) . $this->**email**);  
 $updateUsers = **"UPDATE user SET token=%s WHERE email=%s"**;  
 **require 'application/libs/sendMail.php'**;  
 $body = **"Ciao "** . $result[0][***DB\_USER\_FIRSTNAME***] . **" "** . $result[0][***DB\_USER\_LASTNAME***] . **",<br>  
 Recentemente è stata richiesta la procedura di modifica password!<br><br>  
 <a href='"** . ***URL*** . **"resetPassword/resPassword/"**.$token.**"/**$this->**email'> Per modificare la tua password clicca questo link!</a>"**;  
 **try** {  
 $s = **new** SendMail();  
 $s->mailSend($this->**email**, **"Modifica la password"**, $body);  
 $this->**connection**->query($updateUsers, $token, $this->**email**);  
 **return true**;

In questo metodo controllo che l’utente che sta richiedendo la password sia presente nel database e che sia registrato al sito (type<>0). Se è presente una correlazione creo un token in sha256 di una stringa random, la quale verrà aggiunta nel link di recupero, e l’assegno all’utente nel database. A questo punto preparo oggetto, contenuto e link della email e invio il tutto grazie al metodo mailSend() della classe SendMail (descritta nel capitolo [3.4.2 classe SendMail](#_Classe_SendMail)).

## Pagina reimposta password

In questa pagina è possibile modificare la propria password. Per fare ciò è sufficiente inserire due volte la password che più si desidera e premere il bottone “Cambia”.

Figura 28 Pagina reimposta password

### Riconoscimento utente

Per riconoscere che l’utente che vuole modificare la password sia lo stesso che lo ha richiesto utilizzo questo metodo:

**function** resPassword($hash, $email)  
{  
 **if**($hash != **null**) {  
 $selectUsers = **"SELECT** *\** **FROM user WHERE email=%s AND token=%s"**;  
 $result = $this->**connection**->query($selectUsers, $email, $hash);  
 **if** ($result != **null**) {  
 $updateUsers = **"UPDATE user SET token=NULL WHERE email=%s"**;  
 $this->**connection**->query($updateUsers, $email);  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }**else**{  
 **return false**;  
 }  
}

Esso di occupa di verificare che email e hash contenuti nel link della email di recupero siano gli stessi salvati nel database. Se è così viene resettato il token nel database per disattivare il link presente nella email e l’utente potrà modificare la password.

### Modifica Password

Per modificare la password utilizzo questo metodo:

**public function** modifyPassword($email, $password1, $password2)  
{  
 **if** ($this->**validator**->checkPassword($password1, $password2)) {  
 $password = *password\_hash*($password1, ***PASSWORD\_DEFAULT***);  
 $updateUsers = **"UPDATE user SET password=%s WHERE email=%s"**;  
 $this->**connection**->query($updateUsers, $password, $email);  
 **echo *SUCCESSFUL***;  
 **return true**;  
 }  
 **echo *ERROR***;  
 **return false**;  
}

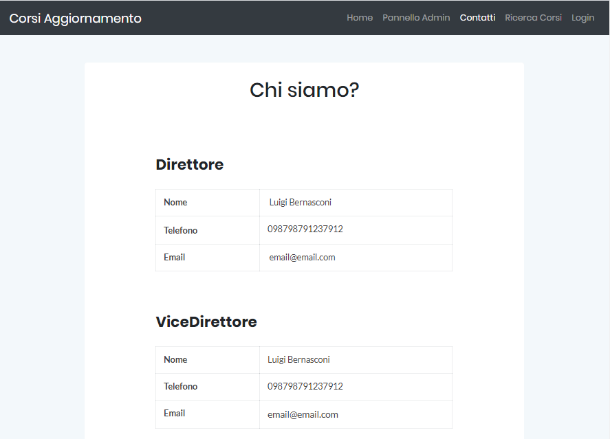
In pratica si occupa di verificare che le password inserite siano le stesse e che abbiano almeno 8 caratteri e un numero/carattere speciale grazie al metodo checkPassword() della classe Validator. Se le password passano questo controllo con successo viene salvata la nuova hash della password.

## Pagina Home

Questa è la pagina principale dell’applicazione. Nella prima parte si può trovare un carosello contente delle foto. Nella seconda parte si può leggere una breve descrizione. Questa pagina è completamente modificabile tramite la pagina di amministrazione.

Figura 29 Pagina home

## Pagina contatti



Questa è dove si possono trovare tutti i dati dei dipendenti dell’azienda. Anche in questo caso la pagina è completamente modificabile tramite la pagina di amministrazione.

Figura 30 Pagina contatti

## Pagina di amministrazione

La pagina di amministrazione permette di gestire un po’ tutte le funzionalità del sito ed è suddivisa in quattro capitoli: pagina principale, pagina contatti, impostazioni e utenti.

### Gestione pagina principale

In questa parte della pagina è possibile aggiungere/eliminare foto della pagina principale oppure modificare la descrizione sempre di quest’ultima attraverso l’editor html presente nella parte inferiore dell’immagine.

Figura 31 Gestione pagina principale

#### Aggiunta Foto

Quando aggiungo una foto, la prima cosa che faccio è controllarla. I controlli che eseguo sono questi:

$uploadOk = ***SUCCESSFUL***;  
$imageFileType = *strtolower*(*pathinfo*($target\_file, ***PATHINFO\_EXTENSION***));  
**if**($photo[***PICTURE\_TMP\_NAME***]!=**null**){  
 **if** (!(*getimagesize*($photo[***PICTURE\_TMP\_NAME***]) !== **false**)) {  
 $uploadOk = ***ERROR***;  
 }  
}**else**{  
 $uploadOk = ***ERROR***;  
}  
**if** ($photo[***PICTURE\_SIZE***] > 5000000) {  
 $uploadOk = ***ERROR***;  
}  
**if** ($imageFileType != **"jpg"** && $imageFileType != **"png"** && $imageFileType != **"jpeg"** && $imageFileType != **"gif"**) {  
 $uploadOk = ***ERROR***;  
}

Il primo controllo che faccio è verificare che il file messo nella cartella temporanea non sia nullo. In seguito verifico tramite il metodo getimagesize() di PHP che il file che si vuole caricare sia un immagine, in caso contrario questo metodo ritorna false. Dopo mi accerto che l’immagine abbia un peso inferiore di 5 MB. Infine controllo l’estensione del file.

Se tutti i controlli vanno a buon fine, combino il nome del file e il timestamp e creo un hash in sha256. Questo viene fatto per evitare che ci siano due file con lo stesso nome nella stessa cartella.

$hash = *hash*(**'sha256'**, *basename*($photo[***PICTURE\_NAME***]).*time*());

Infine sposto l’immagine dalla cartella temporanea alla cartella immagini e aggiungo il percorso al database.

#### Eliminazione foto

Per eliminare un’immagine utilizzo questo metodo:

**public function** delPhoto($path)  
{  
 $selectPhoto = **"SELECT** *\** **FROM photo WHERE path=%s"**;  
 $photo = $this->**connection**->query($selectPhoto, $path);  
 **if**(*count*($photo)>0){  
 $deletePhoto = **"DELETE FROM photo WHERE path=%s"**;  
 $this->**connection**->query($deletePhoto, $path);  
 *unlink*($photo[0][***DB\_PATH\_PHOTO***]);  
 **echo *SUCCESSFUL***;  
 } **else** {  
 **echo *ERROR***;  
 }  
}

In pratica verifico che l’immagine che si vuole eliminare sia presente nel database. In seguito elimino il percorso dal database e infine elimino l’immagine dalla cartella tramite il metodo unlink() di PHP.

#### Modifica descrizione

Per modificare la descrizione della pagina principale utilizzo una libreria che si chiama “summernote”. Questa libreria mi ha permesso di creare questo tipo di textarea:

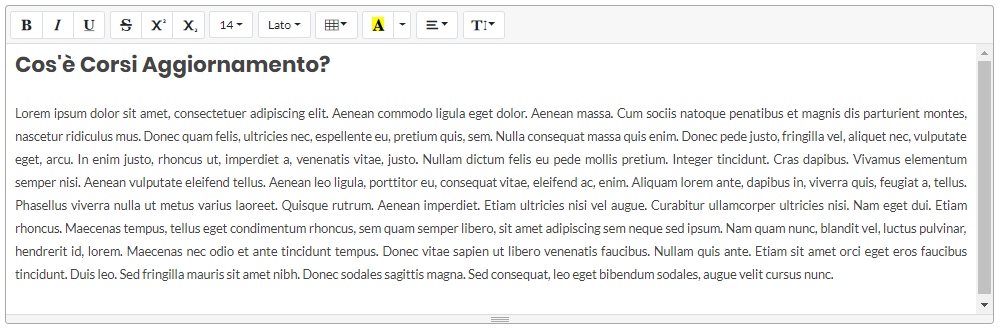


Figura 32 Textarea summernote

Per utilizzare questa libreria ho inserito questi 3 percorsi al file HTML (il primo serve per lo stile, il secondo per il funzionamento Javascript e il terzo per impostare la lingua italiana).

<**link rel="stylesheet" href="application/libs/summernote/summernote-bs4.css"**>

<**script src="application/libs/summernote/summernote-bs4.js"**></**script**>  
<**script src="application/libs/summernote/lang/summernote-it-IT.js"**></**script**>

Una volta importati questi 3 file imposto alla textarea con id “description\_homepage” la libreria summernote. Questa è la configurazione della mia textarea:

$(**'#description\_homepage'**).**summernote**({  
 **toolbar**: [  
 [**'font'**, [**'bold'**, **'italic'**, **'underline'**]],  
 [**'font'**, [**'strikethrough'**, **'superscript'**, **'subscript'**]],  
 [**'fontsize'**, [**'fontsize'**]],  
 [**'fontname'**, [**'fontname'**]],  
 [**'table'**, [**'table'**]],  
 [**'color'**, [**'color'**]],  
 [**'para'**, [**'paragraph'**]],  
 [**'height'**, [**'height'**]]  
 ],  
 **lang**: **"it-IT"**,  
 **fontNames**: [**"Helvetica"**, **"sans-serif"**, **"Arial"**, **"Arial Black"**, **"Comic Sans MS"**, **"Courier New"**, **"Poppins"**],  
 **height**: 300  
});

* toolbar 🡪 imposto tutti i pulsanti per le varie funzioni di modifica (es font, tabelle, colore, …).
* lang 🡪 imposto la lingua dell’editor.
* fontNames 🡪 specifico i font che si possono utilizzare.
* height 🡪 specifico l’altezza in px della textarea.

Per modificare la descrizione utilizzo questo metodo:

**public function** saveDescription($text)  
{  
 *file\_put\_contents*($this->**pathDescription**, $text);  
}

Questo metodo non fa altro che sovrascrivere il contenuto del file dove salvo la descrizione, in base al contenuto che gli si passa come parametro.

### Gestione pagina contatti

Questa è la tab della pagina di amministrazione che mi permette di modificare il contenuto della pagina dove vengono mostrati i contatti dei dipendenti. La procedura di modifica è la medesima utilizzata per la descrizione della pagina principale appena descritta.

Figura 33 Gestione pagina contatti

### Impostazioni dell’applicazione

Questa è la tab della pagina di amministratore per modificare le impostazioni del sito. Più precisamente è possibile modificare i dati di pagamento (IBAN, banca e beneficiario), modificare i giorni prima dell'inizio dei corsi per stabilire la data di chiusure delle iscrizioni, modificare l’età minima per iscriversi ad un corso l’aggiunta/eliminazione delle tipologie dei corsi.

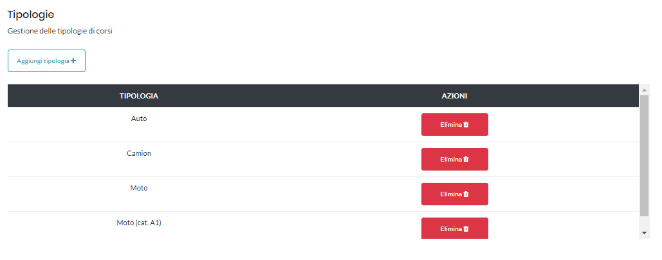


Figura 34 Impostazioni dell'applicazione

#### Gestione Pagamento, giorni scadenza e età minima di iscrizione.

La gestione dei dati di pagamento, dei giorni di scadenza e dell’età minima di iscrizione è molto simile. Il metodo per modificare i dati di pagamento è il seguente:

**public function** modifyPayment($iban, $bank, $beneficiary)  
{  
 $data = **array**();  
 $checkIBAN = $checkBank = $checkBeneficiary = $checkAll = ***SUCCESSFUL***;  
 **if**(!$this->**validator**->checkIBAN($iban)){  
 $checkIBAN = ***ERROR***;  
 $checkAll = ***ERROR***;  
 }  
 **if**(!$this->**validator**->checkNameNumber($bank)){  
 $checkBank = ***ERROR***;  
 $checkAll = ***ERROR***;  
 }  
 **if**(!$this->**validator**->checkName($beneficiary)){  
 $checkBeneficiary = ***ERROR***;  
 $checkAll = ***ERROR***;  
 }  
 **if**($checkAll == ***SUCCESSFUL***){  
 $updateSettings = **"UPDATE settings SET iban\_number=%s, bank=%s, beneficiary=%s"**;  
 $this->**connection**->query($updateSettings,$iban,$bank,$beneficiary,$iban);  
 }  
 $data[] = **array**(  
 ***SATUTS*** => $checkAll,  
 ***CHECK\_IBAN*** => $checkIBAN,  
 ***CHECK\_BANK*** => $checkBank,  
 ***CHECK\_BENEFICIARY*** => $checkBeneficiary  
 );  
 **echo** *json\_encode*($data);  
}

Questo metodo si occupa di verificare tutti i dati passati tramite il form e ritornare un json che contiene lo stato dell’operazione (fallito/bocciato) e i vari risultati dei controlli degli input. Grazie a questo json posso mostrare all’utente delle notifiche di successo/errore. Lo stesso procedimento è stato fatto per la gestione dei giorni di scadenza e per la gestione dell’età minima di iscrizione, ma naturalmente con un solo dato da controllare.

#### Aggiunta tipologia

Per aggiungere una tipologia utilizzo questo metodo:

**function** addTypology($typology){  
 **if**($this->**validator**->checkNameNumber($typology)){  
 $selectTypology = **"SELECT** *\** **FROM typology WHERE name=%s"**;  
 $typologies = $this->**connection**->query($selectTypology,$typology);  
 **if**(*count*($typologies) == 0) {  
 $insertTypology = **"INSERT INTO typology (name) VALUES (%s)"**;  
 $this->**connection**->query($insertTypology,$typology);  
 **echo *SUCCESSFUL***;  
 }**else**{  
 **echo *ERR\_DUP***;  
 }  
 }**else**{  
 **echo *ERROR***;  
 }  
}

Esso verifica che la tipologia non sia già stata aggiunta in precedenza e che abbia un nome valido. In caso che entrami i controlli vadano a buon fine la tipologia viene aggiunta al database e la tabella html viene aggiornata tramite ajax.

#### Rimozione tipologia

Per rimuovere una tipologia utilizzo questo metodo:

**public function** delTypology($name)  
{  
 $selectTypology = **"SELECT** *\** **FROM typology WHERE name=%s"**;  
 $typologies = $this->**connection**->query($selectTypology, $name);  
 **if**(*count*($typologies)>0){  
 $deleteTypology= **"DELETE FROM typology WHERE name=%s"**;  
 $this->**connection**->query($deleteTypology, $name);  
 **echo *SUCCESSFUL***;  
 } **else** {  
 **echo *ERROR***;  
 }  
}

In pratica controlla che la tipologia sia presente nel database e nel caso la elimina.

### Gestione utenti

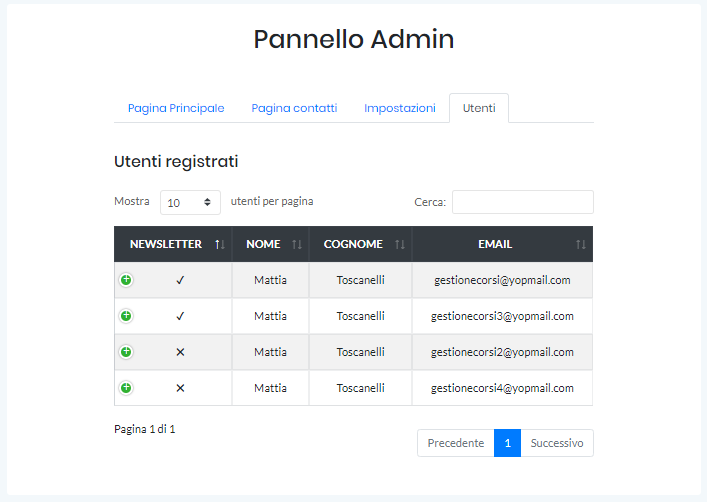


Figura 35 Tabella gestione utenti

Nell’ultima tab della pagina di amministratore è possibile visualizzare tutti gli utenti registrati in una tabella responsive.

Per la creazione di questa tabella ho utilizzato la libreria DataTables. Per utilizzare questa libreria ho inserito questi 5 percorsi al file HTML (i primi due per lo stile e gli altri per il Javascript).

<**link rel="stylesheet" type="text/css" href="application/libs/datatables/DataTables-1.10.21/css/dataTables.bootstrap4.css"**/>  
<**link rel="stylesheet" type="text/css" href="application/libs/datatables/Responsive-2.2.4/css/responsive.dataTables.css"**/>

<**script type="text/javascript" src="application/libs/datatables/datatables.min.js"**></**script**>  
<**script type="text/javascript" src="application/libs/datatables/Responsive-2.2.4/js/dataTables.responsive.js"**></**script**>  
<**script type="text/javascript" src="application/libs/datatables/DataTables-1.10.21/js/dataTables.bootstrap4.js"**></**script**>

Una volta importati questi 5 file imposto alla tabella con id “user\_table” la libreria DataTables. Questa è la configurazione della mia tabella:

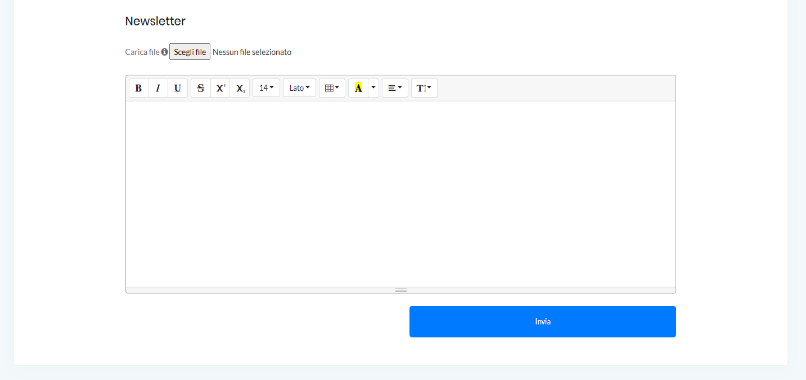
$(**'#user\_table'**).DataTable({  
 **"language"**: {  
 **"lengthMenu"**: **"Mostra \_MENU\_ utenti per pagina"**,  
 **"zeroRecords"**: **"Nessun utente trovato!"**,  
 **"info"**: **"Pagina \_PAGE\_ di \_PAGES\_"**,  
 **"infoEmpty"**: **"Nessun utente trovato!"**,  
 **"infoFiltered"**: **""** },  
 **responsive**: **true**});

In pratica imposto la lingua italiana e rendo la tabella responsive.

Per polare la tabella utenti utilizzo questo metodo che si occupa di ricavare tutti gli utenti registrati:

**public function** getAllUsers()  
{  
 $selectUsers= **"SELECT** *\** **FROM user WHERE type <> 0"**;  
 $users = $this->**connection**->query($selectUsers);  
 **return** $users;  
}

#### Invio email newsletter



Sotto alla tabella che mostra tutti gli utenti registrati al sito si può trovare questo editor dove è possibile inviare email a tutti gli utenti iscritti alla newsletter. Sopra questo editor è presente anche un input file multiplo, così da poter allegare anche dei file con peso sommativo massimo di 25MB.

Figura 36 Form invio email newsletter

Quando si vuole inviare una email di newsletter viene richiamato il metodo sendNewsMail(). Come prima cosa questo metodo ricerca tutti gli utenti iscritti alla newsletter:

$selectEmail = **"SELECT email FROM user WHERE flag\_newsletter=1"**;  
$emails = $this->**connection**->query($selectEmail);

In seguito viene creato un oggetto SendMail (descritto nel capitolo [3.4.2 classe SendMail](#_Classe_SendMail)) e viene eseguito un for per allegare (in caso che ci fossero) tutti i file passati tramite input:

**for** ($i = 0; $i < *count*($files); $i++) {  
 **try** {  
 $s->addFile($files[**"files"** . $i][***FILE\_TMP\_NAME***], $files[**"files"** . $i][***FILE\_NAME***]);  
 } **catch** (Exception $e) {  
 Util::*fail*();  
 }  
}

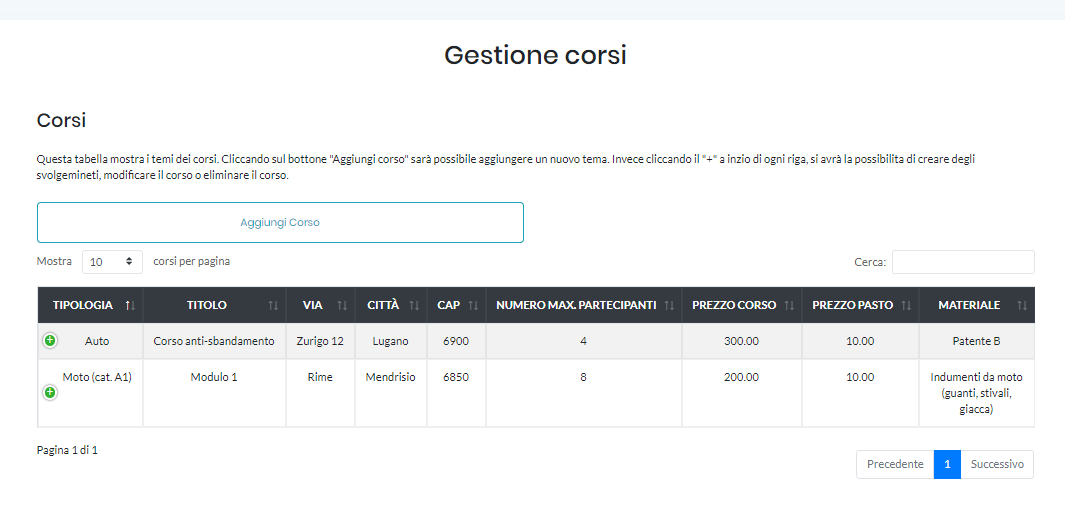
Infine invio la e-mail a tutti gli utenti con il metodo mailSend() della classe SendMail:

$s->mailSend($em[***DB\_USER\_EMAIL***], **"News"**, $body);

## Pagina gestione corsi

La pagina di gestione dei corsi mi permette di gestire tutti i corsi e i loro corrispettivi svolgimenti.

### Gestione corsi

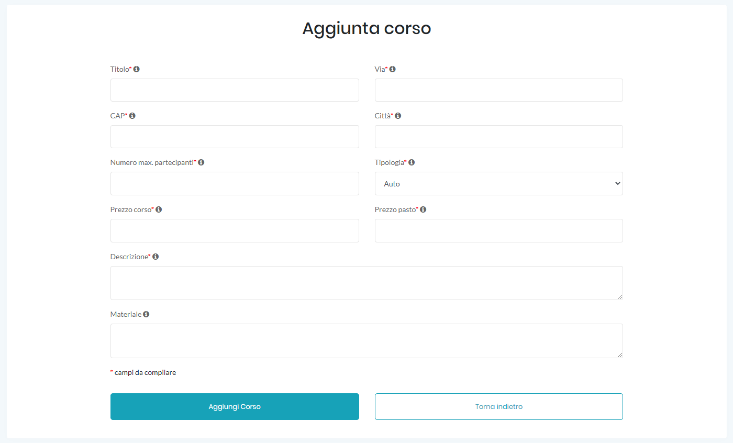


Nella prima parte della pagina è possibile visualizzare una tabella dove sono presenti tutti i corsi con le corrispettive caratteristiche. Si possono aggiungere/

modificare/eliminare i corsi. Infine si possono creare degli svolgimenti sempre di questi corsi.

Figura 37 Pagina di gestione corsi

#### Aggiunta/Modifica corso



Questa è la pagina per aggiungere un corso. La pagina per la modifica è la medesima ma con i gli input già complitati. La pagina è composta da 10 input, dei quali 9 sono obbligatori. Grazie a questo form si specificano tutte le caratteristiche di un corso.

Figura 38 Aggiunta corso

Per aggiungere un corso utilizzo questo metodo:

**public function** addCourse($title, $street, $zip, $city, $maxPartecipants, $typology, $coursePrice, $mealPrice, $courseDescription, $materials)  
{  
 $data = **array**();  
  
 *//variabili per salvare i controlli dei vari input* $checkTitle = $checkStreet = $checkZip = $checkCity = $checkMaxPartecipants = $checkTypology = $checkCoursePrice = $checkMealPrice = $checkCourseDescription = $checkMaterials = $checkAll = ***SUCCESSFUL***;  
  
 *//eseguo tutti i controlli* **if** (!$this->**validator**->checkTitle($title)) {  
 $checkTitle = ***ERROR***;  
 $checkAll = ***ERROR***;  
 }

[...]

**if** ($checkAll == ***SUCCESSFUL***) {  
 $addCourse = **"INSERT INTO course (title,description,zip,city,street,max\_partecipants,materials,meal\_price,course\_price,name\_typology) VALUES (%s,%s,%i,%s,%s,%i,%s,%d,%d,%s)"**;  
 $this->**connection**->query($addCourse, $title, $courseDescription, $zip, $city, $street, $maxPartecipants, $materials, $mealPrice, $coursePrice, $typology);  
 }

Nella prima parte del metodo preparo tutte le variabili che conterranno tutti i risultati dei vari check effettuati tramite la classe Validator (descritta nel capitolo [3.4.5 classe Validator](#_Classe_Validator)). In seguito controllo tutti i 10 input e se tutti sono andati a buon fine aggiunto il corso al database.

Infine preparo un array contente tutti gli stati dei controlli che codificherò in json e invierò al client, così da poter adattare la pagina in base a questi risultati.

$data[] = **array**(  
 ***SATUTS*** => $checkAll,  
 ***CHECK\_TITLE*** => $checkTitle,  
 ***CHECK\_COURSE\_DESCRIPTION*** => $checkCourseDescription,  
 ***CHECK\_ZIP*** => $checkZip,  
 ***CHECK\_CITY*** => $checkCity,  
 ***CHECK\_STREET*** => $checkStreet,  
 ***CHECK\_MAX\_PARTECIPANTS*** => $checkMaxPartecipants,  
 ***CHECK\_MATERIALS*** => $checkMaterials,  
 ***CHECK\_MEAL\_PRICE*** => $checkMealPrice,  
 ***CHECK\_COURSE\_PRICE*** => $checkCoursePrice,  
 ***CHECK\_TYPOLOGY*** => $checkTypology  
);

**echo** *json\_encode*($data);

Il metodo per modificare un corso è il medesimo ma con la differenza che al posto di eseguire un INSERT nel database eseguo un UPDATE.

#### Eliminazione corso

Per rimuovere un corso utilizzo questo metodo:

**public function** deleteCourse($id)  
{  
 $selectCourse = **"SELECT** *\** **FROM course WHERE id=%i"**;  
 $courses = $this->**connection**->query($selectCourse, $id);  
 **if** (*count*($courses) > 0) {  
 $selectIdExecutions = **"SELECT e.id FROM course c, execution e WHERE c.id=e.id\_course AND c.id=%i"**;  
 $idsExe = $this->**connection**->query($selectIdExecutions,$id);  
 **foreach** ($idsExe **as** $idExe){  
 $this->deleteExecution($idExe[***DB\_EXECUTION\_ID***],1);  
 }  
 $deleteCourse = **"DELETE FROM course WHERE id=%i"**;  
 $this->**connection**->query($deleteCourse, $id);  
 **echo *SUCCESSFUL***;  
 } **else** {  
 **echo *ERROR***;  
 }  
}

In pratica controlla che il corso sia presente nel database e nel caso lo elimina. Ricordo che se viene eliminato un corso vengono eliminate a sua volta tutti gli svolgimenti e corrispettive iscrizioni. Quando viene eliminato un corso che ha degli svolgimenti che sono ancora da fare e che contengono delle iscrizioni, viene inviata un email ad ogni utente iscritto per informarlo di questo avvenimento (l’invio di questa email viene spiegata in seguito nel capitolo [4.9.2.2 Eliminazione Svolgimento](#_Eliminazione_svolgimento)):

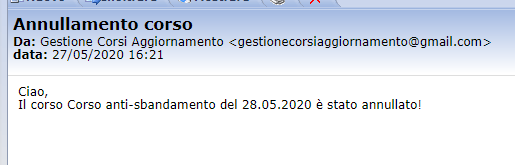
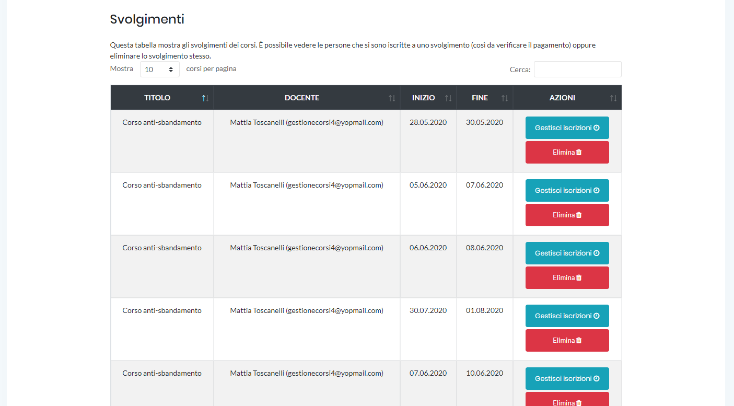


Figura 39 Corso annullato

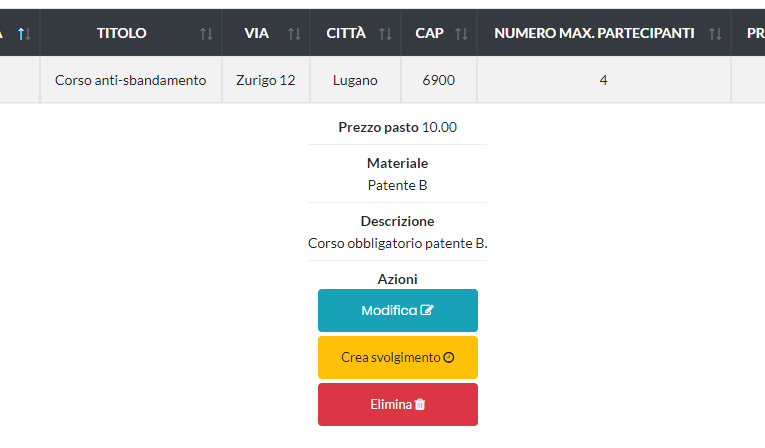
### Gestione svolgimenti



Scorrendo per la pagina si possono trovare tutti gli svolgimenti dei corsi. In grazie a questa tabella e possibile gestire le iscrizioni o eliminare lo svolgimento.

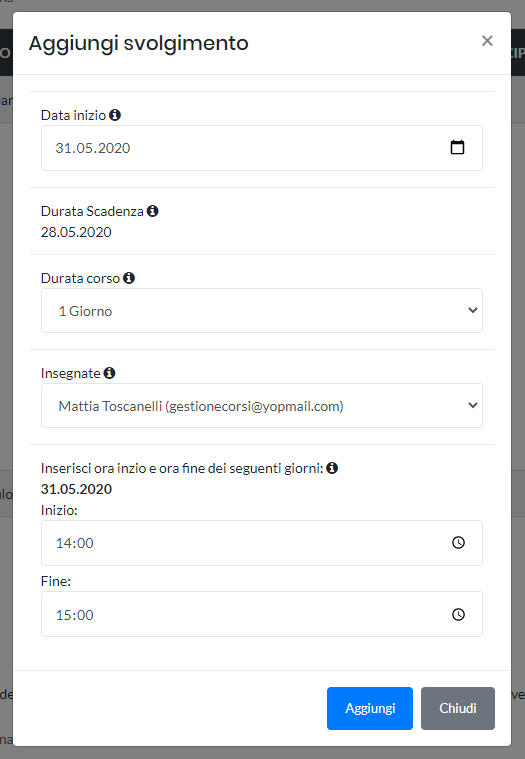
Figura 40 Tabella svolgimenti

#### Aggiunta svolgimento



Per creare uno svolgimento bisogna tornare nella tabella dei corsi. Selezionando un corso avremo la possibilità di cliccare sul bottone “Crea Svolgimento” il quali si aprirà il modale di aggiunta svolgimento.

Figura 41 Aggiunta svolgimento



Questo è il modale per creare uno svolgimento. Nel primo input bisogna specificare la data di inizio del corso. Una volta selezionata viene calcolata direttamente la data di scadenza, cioè la data di chiusura per l’iscrizione. In seguito bisogna specificare la durata dello svolgimento attraverso una select che va da 1 a 5 giorni. Poi bisogna selezionare l’insegnante del corso (che sono gli utenti admin e docenti). Infine ci sono gli input per specificare gli orari dei giorni. Il numero di input varia in base alla durata selezionata.

Figura 42 Modale aggiunta svolgimento

Per aggiornare la data di scadenza ogni volta che l’utente cambia la data di inizio svolgimento utilizzo questo metodo:

**function** *updateDedalineExecution*() {  
 $.when(  
 *getDeadline*()  
 ).**done**(**function** (result) {  
 **var** date = ***document***.getElementById(**"in\_date"**).**value**;  
 ***document***.getElementById(**"day1"**).**innerText** = *getDateFormat*(date);  
 **var** deadline = **new *Date***(date);  
 deadline.setDate(deadline.getDate() - ***Number***(result));  
 ***document***.getElementById(**"deadline\_date"**).**innerText** = *getDateFormat*(deadline);  
 **var** dateLessons = ***document***.getElementsByClassName(**"dL"**);  
 **for** (**var** i = 0; i < dateLessons.**length**; i++) {  
 **var** d = **new *Date***(date);  
 d.setDate(d.getDate() + i + 1)  
 dateLessons[i].**innerHTML** = *getDateFormat*(d);  
 }  
 });  
}

Prima di tutto faccio una richiesta ajax al server per ricavare i giorni da sottrarre per calcolare la data di scadenza attraverso il metodo getDeadiline(). In seguito calcolo la data di scadenza e la imposto all’elemento HTML in modo formattato DD.MM.YYYY grazie al metodo getDateFormat().

Invece per generare gli input time per impostare gli intervalli di orari di uno svolgimento in modo dinamico utilizzo questo metodo:

**function** *changeDay*(x) {  
 **var** count = ***Number***(x.**value**);  
 **if** (***last\_count*** < count) {  
 **var** date = ***document***.getElementById(**"in\_date"**).**value**;  
 **var** newData = **new *Date***(date);  
 **for** (**var** i = ***last\_count*** + 1; i <= count; i++) {  
 $(**'#exe\_body'**).append(**'[codice HTML input]'**);  
 }  
 ***last\_count*** = count;  
 } **else** {  
 **for** (**var** i = ***last\_count***; i > count; i--) {  
 $(**'#raw\_exe'** + i + **''**).remove();  
 }  
 ***last\_count*** = count;  
 }  
}

In base al numero salvato precedentemente di input aggiungo o rimuovo gli input.

Il metodo invece che si occupa di aggiungere lo svolgimento al database si chiama addExecution(). Esso si occupa di verificare tutti i valori degli input. Inoltre verifico che l’insegnate dello svolgimento non sia già occupato nello stesso periodo con un altro svolgimento attraverso il metodo checkOverlapTeacher():

**public function** checkOverlapTeacher($start, $end, $idU)  
{  
 $dStart = *strtotime*($start);  
 $dEnd = *strtotime*($end);  
 $selectDays = **"SELECT start, end FROM execution WHERE id\_user=%i"**;  
 $days = $this->**connection**->query($selectDays, $idU);  
 $check = 0;  
 **foreach** ($days **as** $row) {  
 $xstart = *strtotime*($row[***DB\_EXECUTION\_START***]);  
 $xend = *strtotime*($row[***DB\_EXECUTION\_END***]);  
 **if** (!(($dStart > $xend) || ($dEnd < $xstart))) {  
 $check++;  
 }  
 }  
 **return** $check != 0;  
}

In patica seleziono tutti gli intervalli di date e verifico attraverso un foreach se vanno in collisione con le date dello svolgimento da aggiungere.

#### Eliminazione svolgimento

Per rimuovere uno svolgimento utilizzo il metodo deleteExecution(). Come prima cosa verifica che lo svolgimento da eliminare esista, in seguito ricava i dati degli utenti iscritti allo svolgimento in caso che quest’ultimo dovesse ancora iniziare:

$selectExecution = **"SELECT** *\** **FROM execution WHERE id=%i"**;  
$execution = $this->**connection**->query($selectExecution, $id);  
**if** (*count*($execution) > 0) {  
 $selectEmail = **"SELECT u.firstname, u.lastname, u.email, ex.start FROM user u, enrolls en, execution ex   
 WHERE en.id\_user=u.id AND en.id\_execution = ex.id AND ex.start > CURDATE() AND ex.id=%i"**;  
 $user = $this->**connection**->query($selectEmail, $id);

Se ci dovessero essere utenti ricavo il titolo del corso e invio una email di annullamento corso a tutti i partecipanti (come quella mostrata nel capitolo [4.9.1.2 Eliminazione corso](#_Eliminazione_corso)):

**if** (*count*($user) > 0) {  
 $selectTitleCourse = **"SELECT title FROM course c, execution e WHERE c.id = e.id\_course AND e.id=%i"**;  
 $title = $this->**connection**->query($selectTitleCourse,$id);  
 **require\_once 'application/libs/sendMail.php'**;  
 $body = **"Ciao,<br>  
 Il corso "**.$title[0][***DB\_COURSE\_TITLE***].**" del "**.*date*(**"d.m.Y"**, *strtotime*($user[0][***DB\_EXECUTION\_START***])).**" è stato annullato!</a>"**;  
 **try** {  
 $s = **new** SendMail();  
 $emails = **array**();  
 **for** ($i = 0; *count*($user) > $i; $i++){  
 $emails[$i] = $user[$i][***DB\_USER\_EMAIL***];  
 }  
 $s->mailSendList($emails, **"Annullamento corso"**, $body);  
 } **catch** (Exception $e) {

Infine elimino lo svolgimento dal database:

$deleteExecution = **"DELETE FROM execution WHERE id=%i"**;  
$this->**connection**->query($deleteExecution, $id);

#### Gestione iscrizioni svolgimento

Figura 43 Gestione iscrizioni

Quando si clicca il bottone “Gestione iscrizioni” nella tabella dove vengono mostrati tutti gli svolgimenti si viene indirizzati in questa pagina. Nella prima parte della pagina possiamo vedere ori orari dello svolgimento, la chiusura delle iscrizioni/disiscrizioni e dei posti disponibili.

Per ricavare tutti gli orari e le date di uno svolgimento richiamo il metodo getAllDateExecution():

**public function** getAllDateExecution($idExecution)  
{  
 $dateExecution = **array**();  
 $selectLessons = **"SELECT start,end FROM lesson WHERE id\_execution=%i"**;  
 $lessons = $this->**connection**->query($selectLessons, $idExecution);  
 $dateLessons = **array**();  
 **for** ($i = 0; *count*($lessons) > $i; $i++) {  
 $dateLessons[$i][***START\_LESSON***] = $lessons[$i][***START\_LESSON***];  
 $dateLessons[$i][***END\_LESSON***] = $lessons[$i][***END\_LESSON***];  
 }  
 $dateExecution[***DATE\_EXECUTION***] = $dateLessons;  
 $dateExecution[***DEADLINE***] = *date*(**"d.m.Y"**, *strtotime*(**'-'** . $this->getDeadline() . **' day'**, *strtotime*($dateLessons[0][***START\_LESSON***])));  
 **return** $dateExecution;  
}

Esso si preoccupa di ricavare la data di inizio dello svolgimento e calcolare la data di scadenza con i giorni impostati nelle impostazioni del sito. Inoltre fornisce tutti gli intervalli di orari delle varie lezioni. Per ricavare invece il numero di posti disponibili richiamo il metodo getNumberCourseErollments():

**public function** getNumberCourseEnrollments($idExecution)  
{  
 $selectCourseId = **"SELECT id\_course FROM execution WHERE id=%i;"**;  
 $idCourse = $this->**connection**->query($selectCourseId, $idExecution);  
 **if** (*count*($idCourse) > 0) {  
 $selectEnroll = **"SELECT** *COUNT***(***\****) as 'member' FROM enrolls WHERE id\_execution=%i;"**;  
 $member = $this->**connection**->query($selectEnroll, $idExecution);  
 $selectCourse = **"SELECT max\_partecipants FROM course WHERE id=%i;"**;  
 $max\_member = $this->**connection**->query($selectCourse, $idCourse[0][***DB\_ID\_COURSE***]);  
 **if** (*is\_numeric*($member[0][***MEMBER***]) && *is\_numeric*($max\_member[0][***DB\_COURSE\_MAX\_PARTECIPANTS***])) {  
 **return array**(*intval*($max\_member[0][***DB\_COURSE\_MAX\_PARTECIPANTS***]) - *intval*($member[0][***MEMBER***]), $max\_member[0][***DB\_COURSE\_MAX\_PARTECIPANTS***]);

In questo caso ricavo il numero massimo di partecipanti dalla tabella del corso e conto le righe di iscrizione del determinato svolgimento. Fatto ciò calcolo i posti rimanenti e li ritorno.

##### Pagamento iscrizione

Una volta che l’utente ha pagato, l’amministratore della pagina può spuntare questa checkbox per confermare il pagamento. Una volta confermato l’utente riceve una e-mail con tutti gli orari del corso.

Figura 44 Pagamento iscrizione

Il metodo che si occupa di fare questo si chiama enrollmentPaid(). Esso verifica innanzitutto lo stato della checkbox (se è stata selezionata o se è stata rimossa la selezione). Grazie a questo stato posso inviare una email per informare che il pagamento è andato a buon fine:

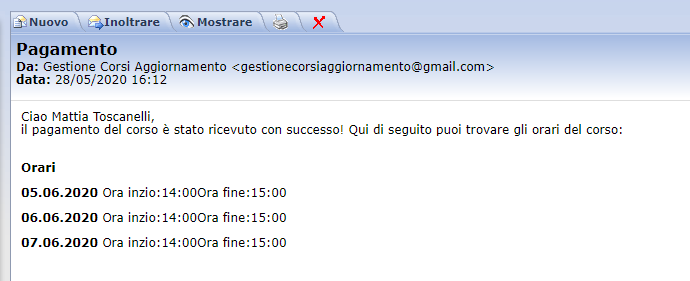


Figura 45 Pagamento eseguito

oppure che è stato rimborsato:

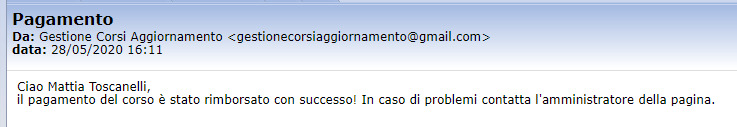


Figura 46 Pagamento rimborsato

Inoltre questo metodo si occupa di salvare lo stato del pagamento nel database:

$updateEnrollment = **"UPDATE enrolls SET flag\_paid=%i WHERE id=%i"**;  
$this->**connection**->query($updateEnrollment, ($status == **"true"**) ? 1 : 0, $idErollment);

##### Cancellazione iscrizione

Per cancellare un’iscrizione bisogna cliccare il bottone “Elimina” alla fine di ogni riga nella tabella delle iscrizioni. In linea di principio le iscrizioni possono venir rimosse senza problemi, ma quando la data di scadenza di un determinato svolgimento è già passata e l’utente ha già pagato, non viene permessa questa operazione. Il metodo che mi permette di fare questo è il seguente:

**public function** deleteEnrollment($id)  
{  
 $selectEnrolls = **"SELECT id\_user, id\_execution, flag\_paid FROM enrolls WHERE id=%i"**;  
 $enrollment = $this->**connection**->query($selectEnrolls, $id);  
 **if** (*count*($enrollment) > 0) {  
 **if** ($enrollment[0][***DB\_FLAG\_PAID***] == 1) {  
 $selectLessons = **"SELECT** *\** **FROM execution WHERE id=%i"**;  
 $lessons = $this->**connection**->query($selectLessons, ($enrollment[0][***EXECUTION\_ID***]));  
 $deadline = $this->getDeadline();  
 $deadline = *strtotime*(**"-**$deadline **day"**, *strtotime*($lessons[0][***DB\_EXECUTION\_START***]));  
 **if** ($deadline < *strtotime*(**'today midnight'**)) {  
 **echo *ERR\_DATE***;  
 **exit**;  
 }  
 }

[...]

$deleteEnrollment = **"DELETE FROM enrolls WHERE id=%i"**;  
 $this->**connection**->query($deleteEnrollment, $id);  
 **echo *SUCCESSFUL***;  
 } **else** {  
 **echo *ERROR***;  
 }  
}

Se tutti i controlli sono andati a buon fine l’iscrizione viene rimossa dal database. Una volta che l’utente è stato rimosso dallo svolgimento riceve questa e-mail:

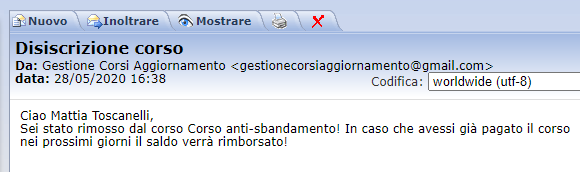


Figura 47 Disiscrizione corso

##### Iscrizione corso da parte dell’amministratore

Se la data di scadenza del corso non è già passata e se ci sono ancora posti disponibili, l’amministratore può aggiungere delle iscrizioni per gli utenti che hanno chiamato per telefono o per email. Gli utenti che vengono inseriti dall’amministratore non potranno accedere al sito.

Figura 48 Aggiunta iscrizione da parte dell'admin

Una volta compilati tutti i campi eseguo questo metodo:

**function** addEnrollment($firstname, $lastname, $birthday, $street, $zip, $city, $mobileNumber, $landlineNumber, $licenseNumber, $nip, $incluFood, $food, $executionId, $foodType, $intolerances, $email)  
{  
 $checkAll = $checkFirstname = $checkLastname = $checkBirthday = $checkStreet = $checkZip = $checkCity = $checkMobileNumber = $checkLandlineNumber = $checkNip = $checkEmail = $checkLicenseNumber = $checkFoodType = $checkIntolerances = $checkOverlap = ***SUCCESSFUL***;  
  
 *//eseguo tutti i controlli*

**[...]**

**if** ($checkAll == ***SUCCESSFUL***) {

*//inserisco iscrizione nel database*

[...]

*//invio e-mail all’utente*

[...]

} **else** {  
 Util::*fail*();  
 }  
 }

*//ritorno stato dei controlli*

In pratica controllo innanzitutto tutti i vari campi, compreso il controllo per verificare se un utente con la stessa email è già iscritto al corso. In seguito aggiungo i dati dell’utente e dell’iscrizione nel database e infine invio questa e-mail all’utente per avvisarlo che è stato iscritto al corso.

La e-mail che viene inviata contiene i dati di pagamento IBAN e alcuni dati del corso:

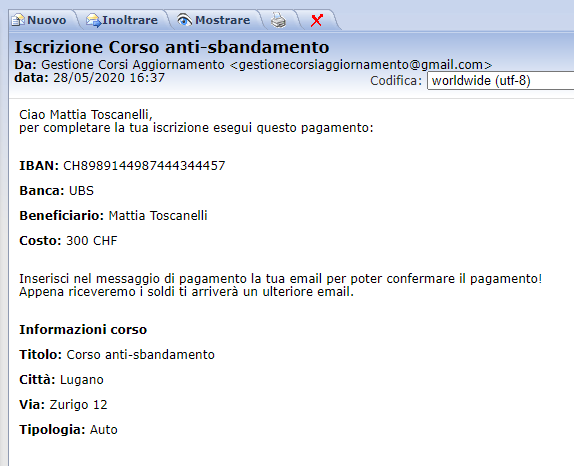
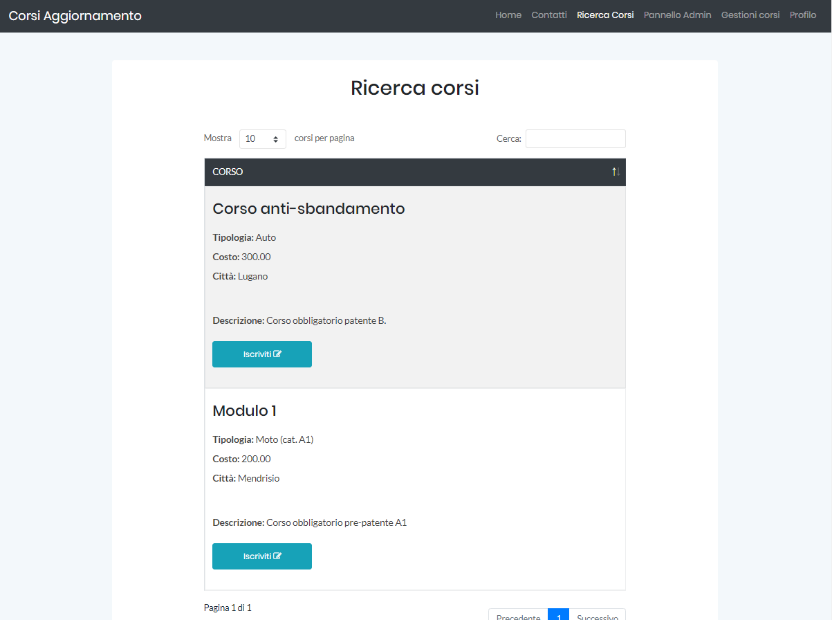


Figura 49 E-mail di iscrizione

## Ricerca corsi



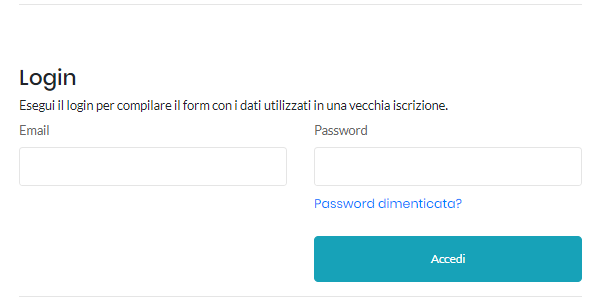
Questa pagina è accessibile da parte di qualsiasi utente. In essa possiamo ricercare i corsi. Ogni corso presenta un titolo, la tipologia, il costo, la città e la descrizione. Cliccando sul bottone iscriviti l’utente verrà mandato nella pagina di iscrizione.

Figura 50 Pagina ricerca corsi

## Pagina iscrizione

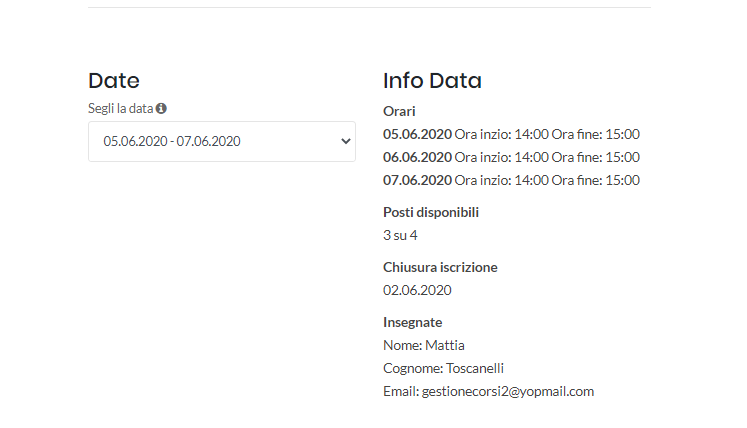
Nella prima parte della pagina per iscriversi ad un corso sono presenti tutti i vari dati del corso, compreso il luogo e i costi. Sotto a queste informazioni c’è la possibilità di scaricare il corso in formato pdf.

Figura 51 Informazioni pagina iscrizione



Scorrendo per la pagina è possibile fare il login direttamente da qua, senza dover accedere alla pagina di login. Grazie al login, l’utente che ha già fatto un’iscrizione e ha deciso di salvare il suo account, si troverà con il form di iscrizione pre-compilato. Il metodo per effettuare il login è il medesimo spiegato nel capitolo [4.3.1 Metodo di accesso al sito](#_Metodo_di_accesso).

Figura 52 Login pagina iscrizione



Scorrendo ancora un po’ la pagina si può scegliere la data di quando si vuole eseguire il corso. Selezionata la data a desta possiamo vedere i vari orari, i posti disponibili, la data di chiusura iscrizione e l’insegnante dello svolgimento. Il metodo utilizzato per ricavare tutte le info è più e o meno lo stesso utilizzato nel capitolo [4.9.2.3 Gestione iscrizioni svolgimento](#_Gestione_iscrizioni_svolgimento) ma con la differenza che qua ricavo le informazioni per tutti gli svolgimenti e non per uno solo.

Figura 53 Selezionamento data

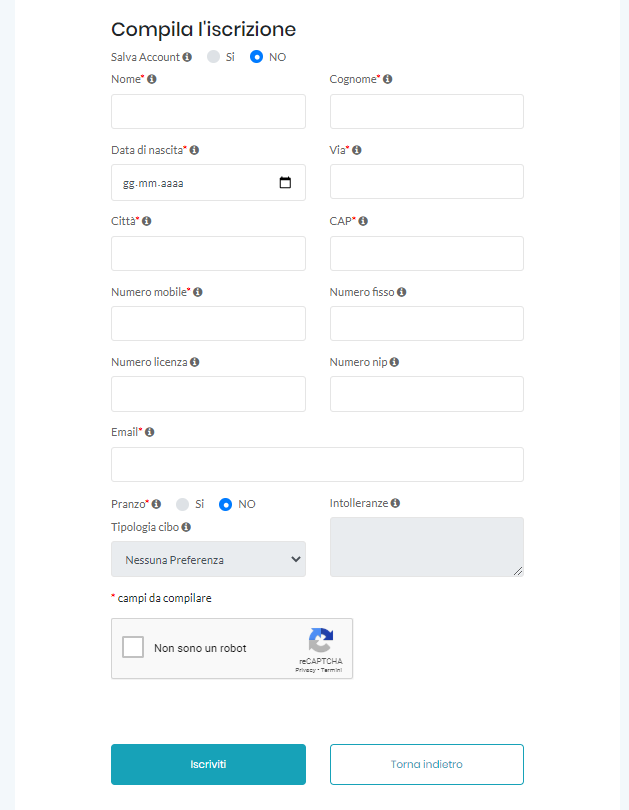


Figura 54 Iscrizione corso

Infine nell’ultima parte della pagina si può trovare il form per iscriversi ad un corso. Il form è molto simile a quello descritto precedentemente per l’aggiunta da parte degli amministratori. Le differenze sono che in questo form si può scegliere se salvare il proprio account oppure no, e se partecipare alle email di newsletter. Inoltre è presente il reCAPTCHA v2 di Google. Il metodo per controllare tutti i campi è praticamente lo stesso utilizzato nel capitolo [4.9.2.3.3 Iscrizione corso da parte del docente](#_Iscrizione_corso_da). Anche in questo caso l’utente riceve una email contente le informazioni del corso e del pagamento dopo aver effettuato l’iscrizione.

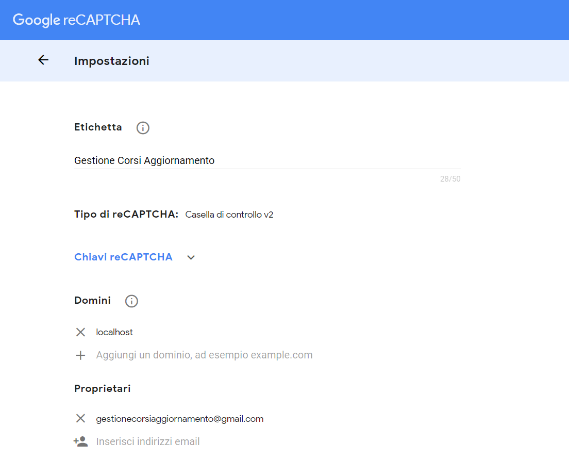
Per riempire automaticamente il form con i dati dell’utente che ha effettuato il login utilizzo questo metodo:

**public function** getAllDataUser($email)  
{  
 $selectUser = **"SELECT** *\** **FROM user WHERE email=%s"**;  
 $user = $this->**connection**->query($selectUser, $email);  
 **return** $user;  
}

Quando un utente già registrato si iscrive ad un corso, invece che aggiungerlo al database, modifico i suoi dati in base al riempimento del form dell’ultima iscrizione.

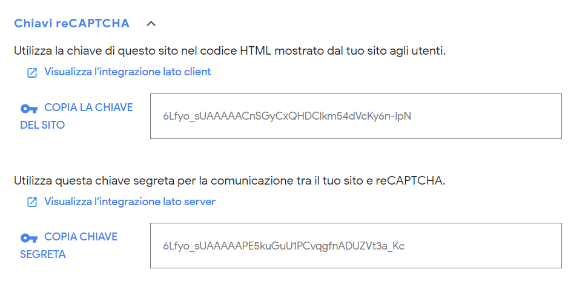
### reCAPTCHA v2 Google

Per implementare il reCAPTCHA v2 ho utilizzato la guida presente sul sito di Google. Ho preferito utilizzare il reCAPTCHA v2 anziché quello di ultima generazione perché è quello ancora più utilizzato nei giorni nostri. Inoltre volevo un feedback da parte del captcha, cosa che quello di nuova generazione non fa. Più precisamente ho scelto la versione per testare di non essere un robot.



Innanzitutto ho creato il mio captcha, assegnando un’etichetta, il dominio dove gira l’applicazione e l’email del proprietario.

Figura 55 Impostazioni reCAPTCHA v2



In seguito ho generato le due chiavi, quella del sito e quella segreta.

Figura 56 Generazione chiavi

Una volta creato il captcha l’ho implementato all’interno dell’applicazione. Prima di tutto ho aggiunto un riferimento alle API del recaptcha di Google:

<**script src="https://www.google.com/recaptcha/api.js"**></**script**>

In seguito ho aggiunto il div che contiene l’input del recaptcha:

<**div class="g-recaptcha" data-sitekey="<?php echo *WEBSITE\_KEY***; **?>"**></**div**>

Nella variabile WEBSITE\_KEY è contenuto il valore della chiave del sito. Essa è modificabile attraverso il file di configurazione.

Poi ho aggiunto al FormData che contiene il valore di tutti gli input del form di iscrizione anche il recaptcha:

formData.append(**'g-recaptcha'**, grecaptcha.getResponse());

Infine sul server eseguo il controllo di validazione:

$secretKey = ***SECRET\_KEY***;  
$responseKey = $\_POST[***CAPTCHA\_V2***];  
$userIP = $\_SERVER[**'REMOTE\_ADDR'**];  
$url = **"https://www.google.com/recaptcha/api/siteverify?secret=**$secretKey**&response=**$responseKey**&remoteip=**$userIP**"**;  
$response = *file\_get\_contents*($url);  
$response = *json\_decode*($response);  
**if** (!$response->**success**) {  
 $checkCaptcha = ***ERROR***;  
 $checkAll = ***ERROR***;  
}

Questo pezzo di codice prepara il link di Google per inviare la richiesta di verifica ed essi mi ritorna un json che contiene anche la risposta della validazione del captcha.

## Pagina personale

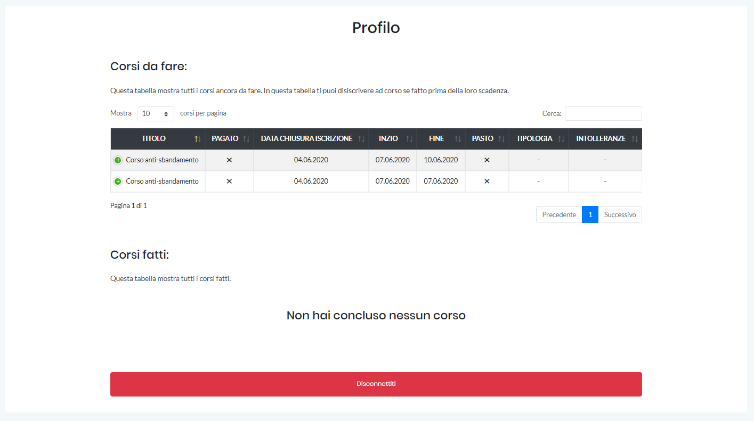
Questa è la pagina personale dell’applicazione. Essa è accessibile solamente agli utenti registrati al sito. All’interno della pagina possiamo trovare due tabelle (nell’immagine una sola) che mostrano i corsi da fare (nel futuro) e i corsi già fatti. Inoltre è possibile rimuovere l’iscrizione per i corsi ancora da fare se fatto prima della sua data di scadenza. Infine è possibile eseguire il logout. Il metodo per rimuovere un’iscrizione è lo stesso utilizzato nel capitolo [4.9.2.3.2 Cancellazione iscrizione](#_Cancellazione_iscrizione). Per eseguire il log out invece distruggo solamente la sessione.

Figura 57 Pagina profilo

## Pagina errore

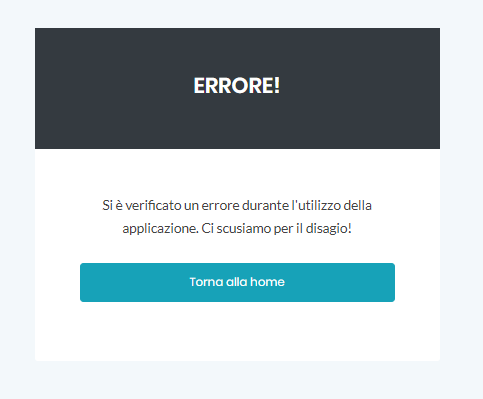


Figura 58 Pagina errore

In caso che l’utente cerca di entrare in pagine non ha lui permesse viene re-indirizzato in questa pagina di errore.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Verifica che la pagina di login funzioni |
| **Descrizione:** | Test per verificare se la pagina di login funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il browser e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Effettuare il login con i seguenti dati di accesso:    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Visualizzare la notifica “Acceso Eseguito” | | |
| **Risultati Attesi:** | La pagina home mostra la scritta “Profilo” nella navbar. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-001 | **Nome:** | Verifica cambiamento password |
| **Descrizione:** | Test per verificare se la procedura di cambiamento funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il browser e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Cliccare la voce “Password Dimenticata?” 4. Inserire la email gestionecorsi@yopmail.com e premere invia 5. Aprire il client di posta elettronica “yopmail.com” e accedere all’account gestionecorsi@yopmail.com | | |
| **Risultati Attesi:** | Il link per modificare la password presente nell’email ricevuta. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-008 | **Nome:** | Verifica modifica pagina principale |
| **Descrizione:** | Test per verificare se la modifica della pagina principale funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Pannello Admin” nella navbar 5. Modificare la descrizione del sito con “Testo di prova” e premere salva 6. Tornare nella pagina principale cliccando “Home” nella navbar | | |
| **Risultati attesi:** | La pagina principale presenta la descrizione “Testo di prova” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-002 | **Nome:** | Verifica visualizzazione pagina home |
| **Descrizione:** | Test per verificare se la pagina home mostra foto e descrizione. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) | | |
| **Risultati attesi:** | La pagina home con le immagini in un carosello e una breve descrizione. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-002 | **Nome:** | Verifica modifica pagina contatti |
| **Descrizione:** | Test per verificare se la modifica della pagina contatti funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Pannello Admin” nella navbar 5. Cliccare la tab “Pagina Contatti” 6. Modificare i contatti del sito con “Testo di prova 2” e premere salva 7. Andare nella pagina contatti cliccando “Contatti” nella navbar | | |
| **Risultati attesi:** | La pagina contati presenta il testo “Testo di prova 2” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-003 | **Nome:** | Verifica visualizzazione pagina contatti |
| **Descrizione:** | Test per verificare se la pagina contatti mostra i dati dei dipendenti. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Contatti” nella navbar | | |
| **Risultati attesi:** | La pagina contatti con i dati dei vari dipendenti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-002 | **Nome:** | Modifica impostazioni dei dati di pagamento |
| **Descrizione:** | Test per verificare se la modifica delle impostazioni di pagamento funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Pannello Admin” nella navbar 5. Cliccare la tab “Impostazioni” 6. Modificare i dati di pagamento 7. Cambiare Banca con “UBS” 8. Accedere in MySQL con l’account seguente:    * Nome: courses\_management\_admin    * Password: CoursesManagement&1 9. Eseguire questo comando in MySQL:    * SELECT \* FROM SETTINGS | | |
| **Risultati attesi:** | I dati aggiornati nel database. | | |
| **Test Case:**  **Riferimento**: | TC-008  REQ-006 | **Nome:** | Verifica Aggiunta corso |
| **Descrizione:** | Test per verificare se l’aggiunta di un corso funziona. | | |
| **Prerequisiti:** | Script database eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare il bottone “Aggiungi corso” 6. Compilare il corso con i seguenti dati:    * Titolo: Nuovo Corso    * Via: Zurigo    * CAP: 6900    * Città: Lugano    * Num max. Partecipanti: 10    * Tipologia: Auto    * Prezzo corso: 100    * Prezzo pasto: 5    * Descrizione: Corso di prova 7. Cliccare “Aggiungi Corso” | | |
| **Risultati attesi:** | Il corso aggiunto visualizzabile nella tabella dei corsi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-006 | **Nome:** | Verifica Modifica corso |
| **Descrizione:** | Test per verificare se la modifica di un corso funziona. | | |
| **Prerequisiti:** | Script database eseguito  TC-008 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare il bottone “Modifica” in fondo alla riga del corso con il titolo “Nuovo corso” 6. Modificare il titolo con “Corso Vecchio” 7. Cliccare “Modifica Corso” | | |
| **Risultati attesi:** | Il corso modificato visualizzabile nella tabella dei corsi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-006 | **Nome:** | Verifica aggiunta svolgimento di un corso |
| **Descrizione:** | Test per verificare l’aggiunta di uno svolgimento di un determinato corso. | | |
| **Prerequisiti:** | Script database eseguito  TC-009 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare il bottone “Crea Svolgimento” in fondo alla riga del corso con il titolo “Nuovo Vecchio” 6. Lasciare tutto invariato e cliccare “Aggiungi” | | |
| **Risultati attesi:** | Lo svolgimento del corso è stato aggiunto e visualizzabile nella tabella degli svolgimenti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-004 | **Nome:** | Verifica aggiunta iscrizione corso |
| **Descrizione:** | Test per verificare l’aggiunta di un’iscrizione ad un corso. | | |
| **Prerequisiti:** | Script database eseguito  TC-010 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Ricerca Corsi” nella navbar 3. Cercare il corso con titolo “Corso Vecchio” 4. Cliccare il bottone “Iscriviti” 5. Compilare l’iscrizione con i seguenti dati:    * Salva Account: NO    * Nome: Mattia    * Cognome: Toscanelli    * Data di nascita: 12.12.1992    * Via: Rime 10    * Città: Mendrisio    * CAP: 6850    * Numero mobile: 0987654321    * Email: provagestione@yopmail.com    * Pranzo: NO 6. Eseguire il captcha 7. Cliccare il bottone “Iscriviti” 8. Aprire il client di posta elettronica “yopmail.com” e accedere all’account provagestione@yopmail.com | | |
| **Risultati attesi:** | Nel client di posta elettronica deve essere presente una email con le informazioni del corso e i dati di pagamento. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-007 | **Nome:** | Verifica conferma pagamento iscrizione |
| **Descrizione:** | Test per verificare la conferma del pagamento di un’iscrizione. | | |
| **Prerequisiti:** | Script database eseguito  TC-011 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare nella tabella di gestione degli svolgimenti il bottone “Gestisci iscrizioni” alla fine della riga con titolo svolgimento “Corso Vecchio” 6. Nella tabella iscritti troviamo l’utente con email provagestione@yopmail.com e vistare la spunta 7. Aprire il client di posta elettronica “yopmail.com” e accedere all’account provagestione@yopmail.com | | |
| **Risultati attesi:** | Nel client di posta elettronica deve essere presente una email con la conferma del pagamento del corso. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-007 | **Nome:** | Verifica cancellazione iscrizione da parte dell’admin. |
| **Descrizione:** | Test per verificare la cancellazione di un’iscrizione da parte dell’admin. | | |
| **Prerequisiti:** | Script database eseguito  TC-012 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare nella tabella di gestione degli svolgimenti il bottone “Gestisci iscrizioni” alla fine della riga con titolo svolgimento “Corso Vecchio” 6. Nella tabella iscritti troviamo l’utente con email provagestione@yopmail.com e cliccare il bottone “Elimina” 7. Aprire il client di posta elettronica “yopmail.com” e accedere all’account provagestione@yopmail.com | | |
| **Risultati attesi:** | Nel client di posta elettronica deve essere presentare una email con la il messaggio che l’utente è stato rimosso dal corso. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-005 | **Nome:** | Verifica funzionamento pagina profilo |
| **Descrizione:** | Test per verificare il funzionamento della pagina profilo. | | |
| **Prerequisiti:** | Script database eseguito  TC-010 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Ricerca Corsi” nella navbar 5. Cercare il corso con titolo “Corso Vecchio” 6. Cliccare il bottone “Iscriviti” 7. Eseguire il captcha 8. Mantenere i dati pre-compilati e premere il bottone “Iscriviti” 9. Cliccare “Profilo” nella navbar | | |
| **Risultati attesi:** | Nella tabella corsi da fare è presente l’iscrizione al corso “Corso Vecchio”. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-015  REQ-014 | **Nome:** | Verifica cancellazione Corso |
| **Descrizione:** | Test per verificare l’eliminazione di un corso. | | |
| **Prerequisiti:** | Script database eseguito  TC-010 eseguito | | |
| **Procedura:** | 1. Aprire il Google Chrome e inserire l’URL [http://localhost/GestioneCorsiAggiornamento/Codice/ GestioneCorsiAggiornamento](http://localhost/GestioneCorsiAggiornamento/Codice/GestioneCorsiAggiornamento%20) 2. Cliccare “Login” nella navbar 3. Eseguire l’accesso con i nuovi dati    * E-mail: gestionecorsi@yopmail.com    * Password: Password&1 4. Cliccare “Gestione Corsi” nella navbar 5. Cliccare nella tabella di gestione corsi il bottone “Elimina” alla fine della riga con titolo svolgimento “Corso Vecchio” | | |
| **Risultati attesi:** | 1. Nel client di posta elettronica dell’utente gestionecorsi@yopmail.com deve essere presente una email con la il messaggio che il corso “Corso Vecchio” è stato annullato. 2. Nella tabella svolgimenti è scomparso lo svolgimento del corso “Corso Vecchio” 3. Nella tabella corsi è scomparso il corso “Corso Vecchio” | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Codice Test** | **Funzionamento** | **Commento** |
| TC-001 | OK | Viene visualizzata la notifica “Accesso eseguito”:    Nella navbar al posto della scritta “Login” é presente la voce “Profilo”: |
| TC-002 | OK | Viene visualizzato il messaggio che dice che è stata inviata l’email per recuperare la password:    L’e-mail ricevuta nel client di posta elettronica per modificare la password:    Se si clicca il link si può cambiare la password: |
| TC-003 | OK | La pagina principale ha cambiato la sua descrizione con “Testo di prova”: |
| TC-004 | OK | La pagina home mostra sia le immagini sia una descrizione: |
| TC-005 | OK | La pagina dei contatti è stata modificata con il testo “Testo prova 2”: |
| TC-006 | OK | La pagina dei contatti mostra tutti i dati dei vari dipendenti: |
| TC-007 | OK | La tabella “settings” è stata modificata: |
| TC-008 | OK | Il corso è stato aggiunto nella tabella di gestione dei corsi: |
| TC-009 | OK | Il corso è stato modificato nella tabella di gestione dei corsi: |
| TC-0010 | OK | Lo svolgimento è stato aggiunto nella tabella di gestione degli svolgimenti: |
| TC-011 | OK | La e-mail di conferma iscrizione è stata ricevuta: |
| TC-012 | OK | La e-mail di conferma pagamento è stata ricevuta: |
| TC-013 | OK | La e-mail di disiscrizione dal corso è stata ricevuta: |
| TC-014 | OK | Nella tabella della pagina di è presente la nuova iscrizione al corso “Corso Vecchio”: |
| TC-015 | OK | Viene inviata la e-mail di annullamento corso:    Lo svolgimento del corso “Corso Vecchio” non è più presente nella tabella di gestione svolgimenti:    Infine nella tabella corsi è scomparso il corso “Corso Vecchio”: |

## Mancanze/limitazioni conosciute

Il progetto è stato sviluppato e testato sul browser Google Chrome. L’applicazione non funziona se si va ad utilizzare Internet Explorer. Infatti esso, essendo obsoleto, non supporta le classi di JavaScript e quindi non permette la validazione dei dati lato client.

# 

# Consuntivo

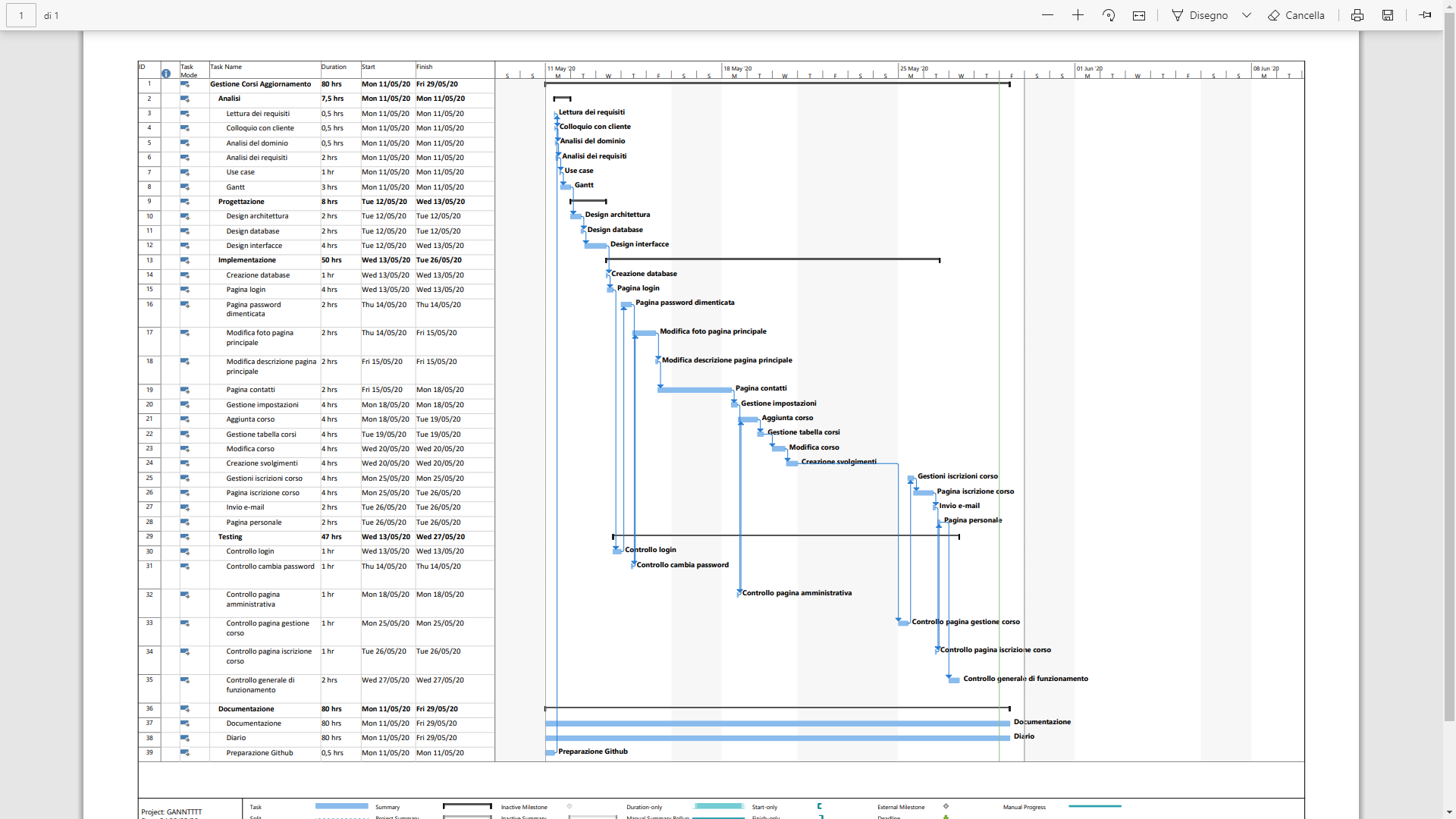


Figura 59 Prima parte - Gantt Consuntivo

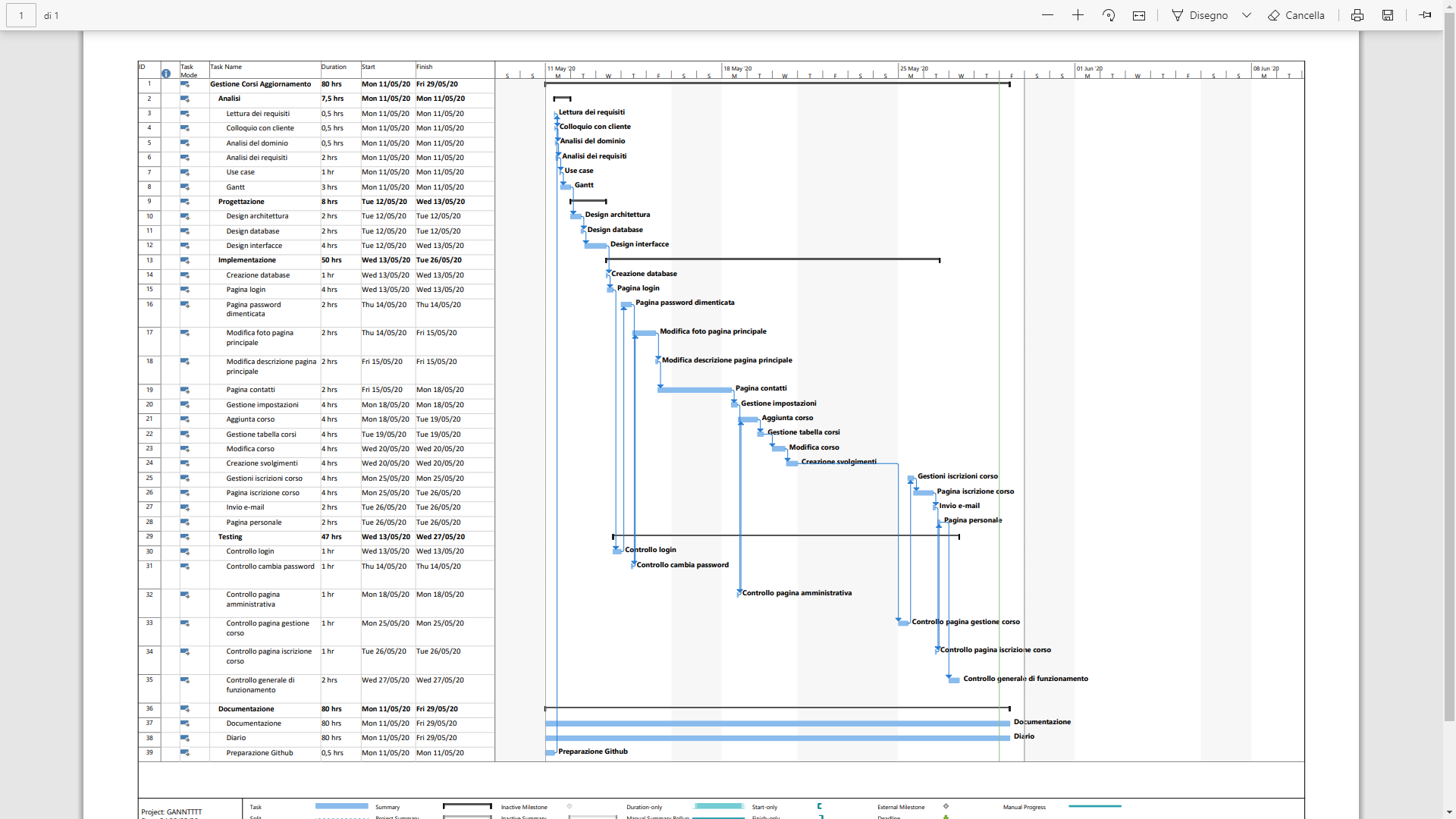


Figura 60 Seconda parte - Gantt Consuntivo

Il gantt preventivo è stato quasi del tutto rispettato. Infatti nel gantt consuntivo le fasi di Analisi, Progettazione e Documentazione sono rimaste invariate. Anche la fase di implementazione è rimasta pressoché uguale, infatti fino all’attività pagina contatti non è cambiato nulla ma è stata modificata l’attività “Gestione pagina contati” con l’attività “gestione impostazioni”. Questa modifica è data dal fatto che in un primo momento non avevo compreso tutte le funzionalità del sito. Per quanto riguarda il resto dell’implementazioni è stato solamente cambiato un po’ l’ordine delle attività ed è stata aggiunta l’attività “Creazione svolgimenti” sempre per lo stesso motivo, cioè che non avevo capito la gestione dei corsi. Infine sono riuscito a concludere la fase di implementazione 4 ore prima del previsto.

Nella fase di Testing, a causa delle leggere modifiche effettuate nella fase di implementazione, alcune attività hanno avuto dei cambiamenti di data.

Per quanto riguarda l’attività “Documentazione” posso dire che nell’ultima settimana di lavoro ho preferito non farla nei primi due giorni, così da concludere il codice del progetto, per poi dedicarmi solo a lei nei seguenti tre giorni.

# Conclusioni

Grazie a questo progetto il mio cliente potrà gestire i corsi di aggiornamento con un’applicazione web rapida, sicura e di facile utilizzo. Sul mercato esistono già dei prodotti simili al mio, ma il mio prodotto offre una grafica più aggiornata rispetto alla concorrenza. Inoltre la mia applicazione ha un’ergonomia migliore quando si cerca di compilare un’iscrizione, infatti per ogni campo è spiegato dettagliatamente come deve essere compilato. Infine, a differenza degli altri siti di corsi, il mio sito permette di registrarsi e accedere ad una pagina personale dove si possono vedere tutti i corsi da fare e quelli già fatti

Sicuramente questo progetto non è stata una perdita di tempo in quanto ho imparato ad utilizzare nuove librerie a API, come la libreria DataTables (per gestire le tabelle in modo dinamico), la libreria Summernote (Editor di testo in HRTL) e le API di Google (per rendere la pagina di iscrizione al sicuro da attacchi bot con il reCAPTCHA). Il prodotto è facile e intuitivo, adatto a qualsiasi tipo di utente, che sia un esperto di informativa sia all’utente meno tecnologico di questo pianeta. Questo progetto garantisce sicuramente più ordine agli utenti finali, infatti come detto già prima, possono accedere ad una pagina personale dove vedono tutti i propri corsi. Infine risulta comodo anche per l’amministratore che può modificare la descrizione e le foto della pagina principale, modificare la pagina di contatti, modificare le impostazioni del sito e inviare email di newsletter oppure gestire tutti i vari corsi.

## Sviluppi futuri

In futuro si potrebbero aggiungere diverse funzioni. Una tra queste è quello di assegnare una funzione specifica agli utenti di tipo docente. Infatti in questa applicazione non possono effettivamente fare niente e magari si potrebbe creare una pagina dove possono vedere tutti i corsi da loro insegnati con tutti gli utenti iscritti. Un’altra aggiunta che si potrebbe aggiungere è la gestione degli utenti, infatti l’amministratore non può aggiungere/modificare/eliminare gli utenti, ma deve farlo direttamente dal database. I corsi che vengono creati hanno un luogo fisso, questo significa che se si vuole svolgere un determinato corso in un altro paese rispetto a quello specificato bisogna creare un altro corso. Una modifica che implementerei è che al posto di specificare l’indirizzo durante la creazione di un corso lo farei direttamente quando si crea uno svolgimento. Un’ultima aggiunta che si potrebbe fare è quello di aggiungere una rappresentazione grafica (cioè con una cartina) il luogo dove verrà svolto il corso utilizzando per esempio le API di Google.

Ci sono molte piccole funzioni che si potrebbe aggiungere senza neanche troppi sforzi.

## Considerazioni personali

Come già detto precedentemente ho imparato molte cose che mi saranno utili nella mia vita da programmatore. La prima tra queste è quella di aver sviluppato un intero progetto in modo individuale in solamente tre settimane. Questa cosa non mi era mai successi, infatti gli scorsi progetti duravano circa un semestre. Inoltre, come già detto prima ho imparato ad utilizzare nuove librerie, come quella per creare editor di testi in HTML, quella per gestire le tabelle in modo dinamico, … tutte conoscenze che potrò utilizzare per futuri progetti.

# Bibliografia

## Sitografia

<https://www.mockflow.com/>, Mockup delle interfacce, 13.05.2020

<https://github.com/puikinsh/srtdash-admin-dashboard>, Template sito web, 13.05.2020

<https://www.w3schools.com/>, Guide e Tutorial web, Ultima visita 27.05.2020

<https://meekro.com/>, MySQL Library PHP, 13.05.2020

<https://www.php.net/>, Guide e Tutorial PHP, Utilizzato frequentemente fino al 27.05.2020

<https://www.pexels.com/it-it/>, Immagini Free Copyright, 14.05.2020

<https://summernote.org/>, Text Editor HTML, 18.05.2020

<https://datatables.net/>, Tabelle interattive responsive, 18.05.2020

<https://github.com/PHPMailer/PHPMailer>, PHPMailer (invio e-mail), 13.05.2020

<https://stackoverflow.com/>, Soluzioni a problemi riscontrati, Ultima visita 27.05.2020

<https://www.iconfinder.com/>, Favicon Free Copyright, 28.05.2020

<http://www.yopmail.com/>, Sito Email Temporanee, Ultima visita 29.05.2020

## Indice delle figure

[Figura 1 Use case 8](#_Toc41645216)

[Figura 2 Gantt Preventivo Intero 9](#_Toc41645217)

[Figura 3 Gantt Preventivo – Analisi 10](#_Toc41645218)

[Figura 4 Gantt Preventivo – Progettazione 11](#_Toc41645219)

[Figura 5 Gantt Preventivo – Implentazione 12](#_Toc41645220)

[Figura 6 Gantt Preventivo - Testing 13](#_Toc41645221)

[Figura 7 Gantt Preventivo - Documentazione 14](#_Toc41645222)

[Figura 8 Design dell'architettura del sistema 16](#_Toc41645223)

[Figura 9 Diagramma E-R 17](#_Toc41645224)

[Figura 10 Design - Pagina principale 21](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645225)

[Figura 11 Design - Pagina contatti 21](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645226)

[Figura 12 Design - Pagina di login 22](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645227)

[Figura 13 Design - Pagina richiesta recupero password 22](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645228)

[Figura 14 Design - Pagina reimposta password 23](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645229)

[Figura 15 Design - Pagina amministrativa 23](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645230)

[Figura 16 Design - Pagina ricerca appartamenti 24](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645231)

[Figura 17 Design - Pagina iscrizione corso 24](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645232)

[Figura 18 Design - Pagina gestione corso 25](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645233)

[Figura 19 Design - Pagina personale 25](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645234)

[Figura 20 UML - Classe Database 26](#_Toc41645235)

[Figura 21 UML - Classe SendMail 27](#_Toc41645236)

[Figura 22 UML - Classe Util 28](#_Toc41645237)

[Figura 23 UML - Classe DSession 28](#_Toc41645238)

[Figura 24 UML - Classe Validator 29](#_Toc41645239)

[Figura 25 Struttura MVC 31](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645240)

[Figura 26 Pagina di login 32](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645241)

[Figura 27 Pagina password dimenticata 33](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645242)

[Figura 28 Pagina reimposta password 34](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645243)

[Figura 29 Pagina home 35](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645244)

[Figura 30 Pagina contatti 36](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645245)

[Figura 31 Gestione pagina principale 36](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645246)

[Figura 32 Textarea summernote 37](#_Toc41645247)

[Figura 33 Gestione pagina contatti 38](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645248)

[Figura 34 Impostazioni dell'applicazione 39](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645249)

[Figura 35 Tabella gestione utenti 41](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645250)

[Figura 36 Form invio email newsletter 42](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645251)

[Figura 37 Pagina di gestione corsi 42](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645252)

[Figura 38 Aggiunta corso 43](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645253)

[Figura 39 Corso annullato 44](#_Toc41645254)

[Figura 40 Tabella svolgimenti 45](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645255)

[Figura 41 Aggiunta svolgimento 45](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645256)

[Figura 42 Modale aggiunta svolgimento 45](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645257)

[Figura 43 Gestione iscrizioni 48](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645258)

[Figura 44 Pagamento iscrizione 49](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645259)

[Figura 45 Pagamento eseguito 49](#_Toc41645260)

[Figura 46 Pagamento rimborsato 49](#_Toc41645261)

[Figura 47 Disiscrizione corso 50](#_Toc41645262)

[Figura 48 Aggiunta iscrizione da parte dell'admin 51](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645263)

[Figura 49 E-mail di iscrizione 52](#_Toc41645264)

[Figura 50 Pagina ricerca corsi 52](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645265)

[Figura 51 Informazioni pagina iscrizione 53](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645266)

[Figura 52 Login pagina iscrizione 53](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645267)

[Figura 53 Selezionamento data 53](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645268)

[Figura 54 Iscrizione corso 54](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645269)

[Figura 55 Impostazioni reCAPTCHA v2 55](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645270)

[Figura 56 Generazione chiavi 55](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645271)

[Figura 57 Pagina profilo 56](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645272)

[Figura 58 Pagina errore 56](file:///C:\Apache24\htdocs\gestione-web-corsi-aggiornamento\3_Documentazione%20(word%20e%20pdf)\DocumentazioneGestioneCorsiAggiornamento.docx#_Toc41645273)

[Figura 59 Prima parte - Gantt Consuntivo 71](#_Toc41645274)

[Figura 60 Seconda parte - Gantt Consuntivo 72](#_Toc41645275)

## Glossario

|  |  |
| --- | --- |
| Parola | Definizione |
| Array | Contenitore di più valori |
| API | Application Programmig Interface è appunto un’interfaccia di programmazione delle applicazioni che permette di che permette di comunicare con altri servizi o prodotti senza sapere effettivamente come sono stati sviluppati. |
| Ajax | È una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Fonte: <https://it.wikipedia.org/wiki/AJAX>) |
| Back-end | Parte del software che gestisce tutto ciò che non comprende l’interfaccia grafica, quindi gestione delle immagini, dei controlli dei dati, gestione del database, … |
| Client | È un componente software presente in una macchina host. Esso accede a servizi o alle risorse di un server. |
| CSS | Linguaggio che permette di personalizzare lo stile di una pagina HTML. |
| Database | Contenitore che ospita tabelle di dati. |
| Framework | Insieme di classi astratte e delle relazioni tra di esse. |
| Font-end | Parte del software visibile all’utente finale, quindi testo, foto, video, … |
| Hash | Funzione irreversibile che permette di generare una una stringa di lunghezza fissa con caratteri casuali. Una stringa A viene trasformata in B, ma B non può tornare ad essere A. Di solito viene utilizzato per la sicurezza delle password. |
| HTML | Linguaggio di murkup utilizzato per la formattazione e l’impaginazione di pagine web. |
| Javascript | Linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client. (Fonte: <https://it.wikipedia.org/wiki/JavaScript>) |
| Libreria | Insieme di metodi. |
| Metodo  (Funzione) | Raggruppamento di più istruzioni di codice in un unico blocco. |
| Modale | Interfaccia grafica che attirano l’attenzione dell’utente. Quando attivi disabilitano la pagina madre. |
| MySQL | Sistema per gestire i database, archiviare i dati. |
| Navbar | Raggruppamento di link sopra alla pagina web. |
| PHP | Linguaggio di scripting interpretato, originariamente concepito per la programmazione di pagine web dinamiche. (Fonte: <https://it.wikipedia.org/wiki/PHP>) |
| Server | Componente hardware che offre risorse e/o servizi. |
| Variabile | Contenitore di un valore. |

# Allegati

* Diari di lavoro
* Quaderno dei compiti
* Manuale di installazione del prodotto (README Gitsam)
* Codice sorgente trovabile su Gitsam:
* <http://212.117.109.242:3230/lavoro_finale_lpi_2020/gestione-web-corsi-aggiornamento>
* Database