

# **Piattaforma per la scoperta dei grotti in Ticino**

1	Introduzione.....	4
1.1	Informazioni sul progetto.....	4
1.2	Abstract.....	4
1.3	Scopo.....	4
	Analisi.....	5
1.4	Analisi del dominio.....	5
1.5	Analisi e specifica dei requisiti.....	5
1.6	Use case.....	7
1.7	Pianificazione.....	8
1.7.1	Analisi.....	9
1.7.2	Progettazione.....	9
1.7.3	Implementazione.....	10
1.7.4	Test.....	11
1.8	Analisi dei mezzi.....	11
1.8.1	Software.....	11
1.8.2	Hardware.....	11
2	Progettazione.....	12
2.1	Design dell'architettura del sistema.....	12
2.2	Design dei dati e database.....	13
2.3	Design delle interfacce.....	15
2.3.1	Pagina iniziale.....	15
2.3.2	Pagina di login.....	15
2.3.3	Pagina di registrazione.....	16
2.3.4	Pagina di creazione dei grotti.....	17
2.3.5	Pagina di amministrazione.....	18
3	Implementazione.....	19
3.1	Struttura del progetto.....	19
3.2	Database.....	19
3.2.1	Tabella utente.....	19
3.2.2	Tabella grotto.....	20
3.2.3	Tabella foto.....	20
3.2.4	Tabella voto.....	20
3.3	Pagina Home.....	21
3.4	Pagina Grotto.....	22
3.5	Pagina di login.....	22
3.6	Pagina di registrazione.....	23
3.7	Pagina di ripristino della password.....	23
3.8	Pagina di aggiunta grotto.....	24
3.9	Pagina di amministrazione.....	24
3.9.1	Sezione utenti.....	25
3.9.2	Sezione grotti.....	25
3.9.3	Sezione inserimenti.....	26
3.9.4	Sezione immagini.....	26
3.10	Pagina di warning.....	26
3.11	Models.....	27
3.11.1	Classe DBConnection.....	28
3.11.2	Classe input_manager.....	28
3.11.3	Classe mail_manager.....	28
3.11.4	Classe utente_model.....	28
3.11.5	Classe grotto_model.....	29
3.11.6	Classe foto_model.....	30
3.11.7	Classe voto_model.....	31
3.11.8	Classe fascia_prezzo_model.....	31
3.11.9	Classe ruolo_model.....	32
3.12	Controllers.....	32
3.12.1	Controller home.....	33
3.12.2	Controller login.....	33

3.12.3 Controller register.....	33
3.12.4 Controller first login.....	34
3.12.5 Controller grotto.....	34
3.12.6 Controller warning.....	34
3.12.7 Controller new user.....	35
3.12.8 Controller add.....	35
3.12.9 Controller admin.....	35
3.12.10 Controller gestione.....	36
3.12.11 Controller reset.....	36
4 Test.....	37
4.1 Protocollo di test.....	37
4.2 Risultati test.....	44
4.3 Mancanze/limitazioni conosciute.....	44
5 Consuntivo.....	44
6 Conclusioni.....	46
6.1 Sviluppi futuri.....	46
6.2 Considerazioni personali.....	46
7 Bibliografia.....	47
7.1 Sitografia.....	47
7.2 Indice delle immagini.....	47
8 Allegati.....	48

## **1 Introduzione**

---

### **1.1 Informazioni sul progetto**

Allievo: Matteo Forni  
Docente responsabile: Luca Peduzzi  
Scuola Arti e Mestieri di Trevano, sezione informatica, classe I4AA  
Data di inizio: 03.09.2019  
Termine di consegna: 20.12.2019

### **1.2 Abstract**

The project requirement is to create a website that allows to see and manage Ticino's restaurants. It must have a interactive map where you can check the location of the restaurant and find all its information. When a user logs in his account he has the possibility to vote a restaurant and add pictures. A page for administration is required, an admin can add, delete and modify users and restaurants.

The result is an easy and straightforward website where users can discover new places and let people know where they ate better.

### **1.3 Scopo**

Lo scopo di questo progetto è quello di creare un sito web che consenta di esplorare in maniera semplice e intuitiva i vari grotti del Canton Ticino. Grazie a questo prodotto si potranno visualizzare le posizioni dei punti di ristorazione con le informazioni su di essi. Dovrà essere disponibile inoltre una ricerca dei grotti a cui si possono applicare dei filtri come il nome, la località e la fascia di prezzo.

Eseguendo l'accesso al sito si avrà la possibilità di inserire dei nuovi ristoranti con una valutazione, la fascia di prezzo e le informazioni su di esso.

## 2 Analisi

### 2.1 Analisi del dominio

Il progetto dovrà essere una piattaforma per la scoperta dei grotti del Ticino, momentaneamente esiste un sito che fa già questo e si tratta di ticino.ch. Esso si comporta in maniera molto simile a come dovrà fare questo progetto, consente infatti di visualizzare tramite una mappa i vari grotti e di filtrarli tramite dei filtri simili a quelli richiesti in questa piattaforma. Il sito già esistente utilizza però delle mappe che non consentono di inserire i marker tramite un indirizzo dato che in esso un utente non può aggiungere una località.

Il contesto del prodotto è quindi in parte organizzato ma in maniera molto semplice e che necessita di un aggiornamento continuo da parte degli amministratori dato che solo loro possono aggiungere i grotti alla mappa.

Gli utenti che utilizzeranno il sito saranno principalmente senza alcuna conoscenza tecnica ed è per questo motivo che le interfacce dovranno essere molto semplici e facili da utilizzare. Il sito mira ad essere utilizzato dal maggior numero di persone nel cantone e fuori.

### 2.2 Analisi e specifica dei requisiti

ID: REQ-01	
<b>Nome</b>	Creazione pagina riservata agli amministratori
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Si necessitano i permessi di amministratore per accedervi
Sotto requisiti	
<b>001</b>	Si dovrà avere la possibilità di creare/modificare/eliminare gli utenti
<b>002</b>	Si dovrà avere la possibilità di creare/modificare/eliminare dei grotti
<b>003</b>	Si dovrà avere la possibilità di rendere un utente normale amministratore
<b>004</b>	Si dovrà avere la possibilità di approvare o rifiutare la creazione di un grotto da parte di un utente

ID: REQ-02	
<b>Nome</b>	Creazione pagina principale
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	-
Sotto requisiti	
<b>001</b>	Ci dovrà essere una mappa interattiva con la posizione dei grotti
<b>002</b>	Ci dovrà essere una sezione riservata alla ricerca dei grotti
<b>003</b>	Si dovrà potere applicare alla ricerca dei filtri in base al zona/fascia di prezzo/nome

ID: REQ-03	
<b>Nome</b>	Creazione pagina di login
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	-
Sotto requisiti	
<b>001</b>	Si dovrà avere la possibilità di aggiornare la password
<b>002</b>	Se l'utente è stato creato da un admin si dovrà cambiare la password

ID: REQ-04	
<b>Nome</b>	Creazione pagina di registrazione
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	-
Sotto requisiti	
<b>001</b>	Ci dovrà essere un controllo istantaneo dei dati inseriti

ID: REQ-05	
<b>Nome</b>	Creazione pagina di creazione di un grotto
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Bisognerà avere effettuato il login per accedervi
Sotto requisiti	
<b>001</b>	Si dovrà avere la possibilità di inserire testo e immagini
<b>002</b>	Si dovrà avere la possibilità di assegnare una valutazione (sotto forma di stelle)
<b>003</b>	Si dovrà avere la possibilità di assegnare una fascia di prezzo

ID: REQ-06	
<b>Nome</b>	Condivisione sui social di una località
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	-

## 2.3 Use case

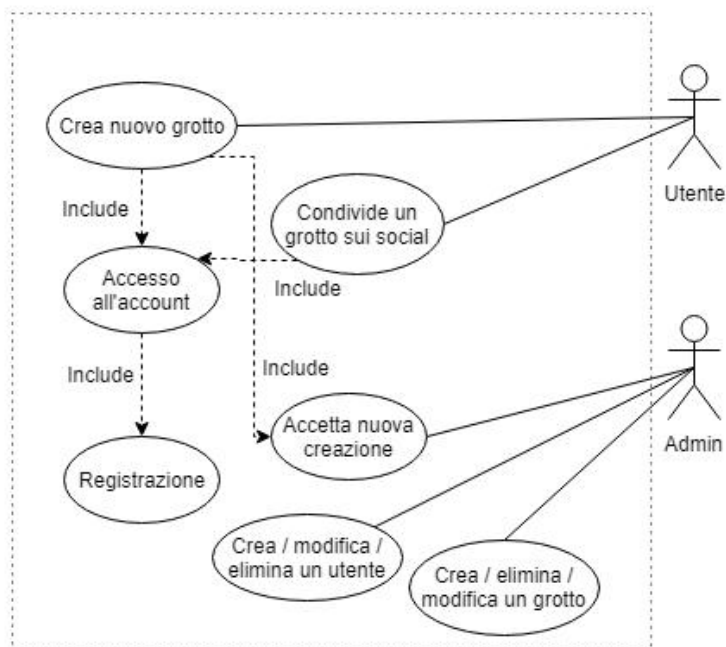


Figure 1 Use Case

La foto precedente rappresenta lo use case del prodotto, come si può vedere dallo schema l'utente di base ha la possibilità, dopo aver eseguito il login, di creare un nuovo grotto o di condividerne uno sui social. Senza effettuare il login esso può solamente visualizzare i grotti sulla mappa interattiva o tramite la sezione di ricerca, essendo questo scontato non è stato inserito nello schema.

L'amministratore può invece creare, modificare o eliminare un utente oppure un grotto ed inoltre è incaricato di verificare e accettare o rifiutare le creazioni degli utenti. Per essere un admin bisogna per forza eseguire il login e quindi questa parte non è stata inserita nell'immagine per evitare di ripetere inutilmente una cosa scontata.

## 2.4 Pianificazione

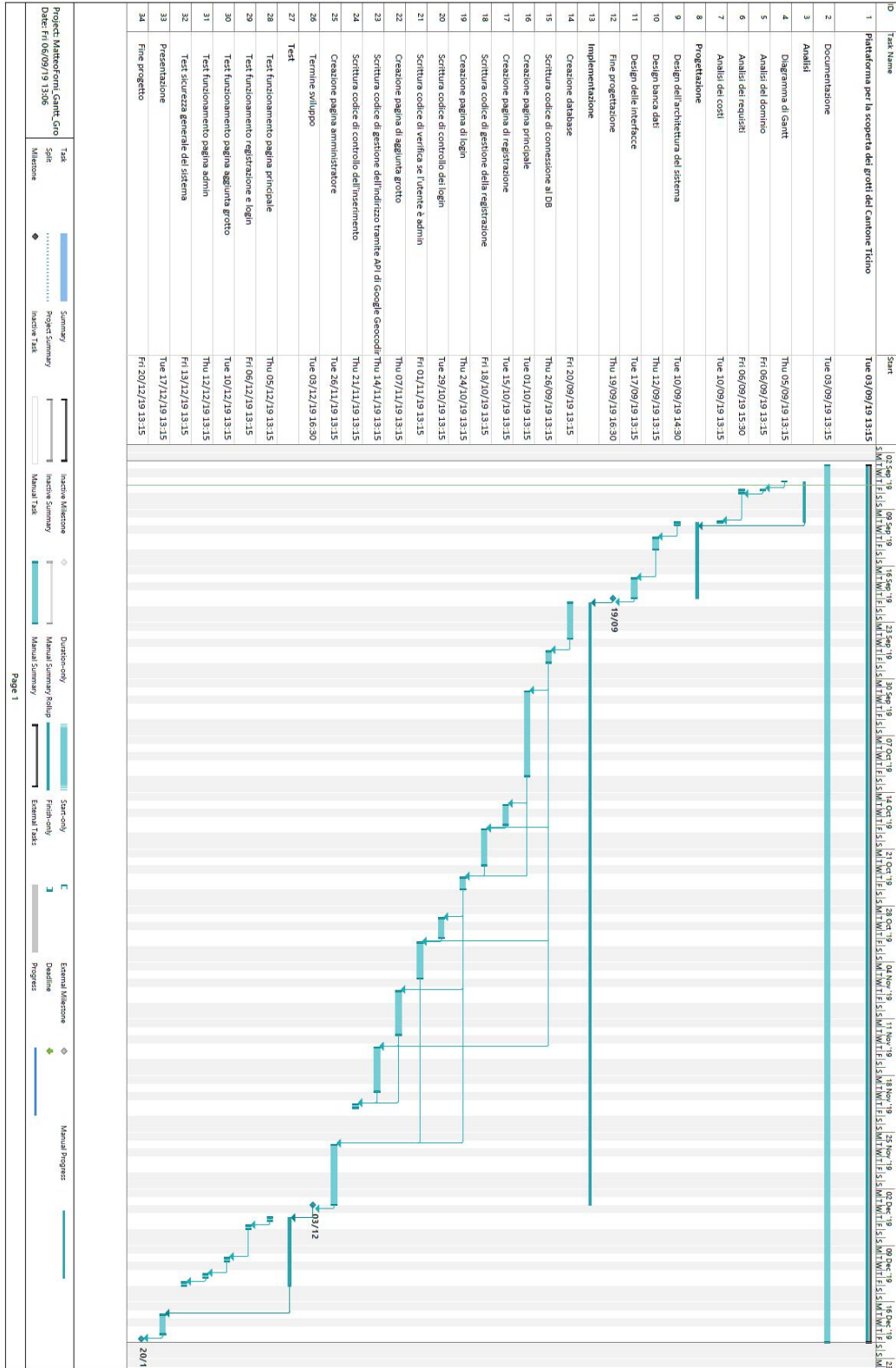


Figure 2 Diagramma di Gantt



Nella figura superiore si può vedere il diagramma di Gantt completo, esso rappresenta in maniera indicativa come dovrebbe andare il progetto. Esso è stato diviso in quattro grandi categorie che sono Analisi, Progettazione, Implementazione e Test, oltre ad esse vi è la documentazione che procede lungo tutto il progetto dato che verrà completata con l'avanzare delle altre attività.

### 2.4.1 Analisi

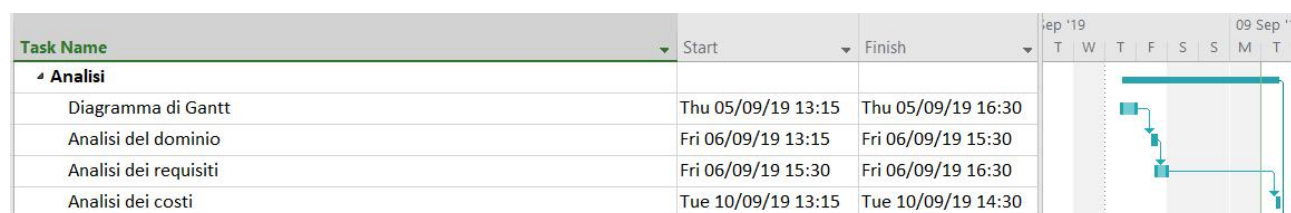


Figure 3 Gantt capitolo Analisi

L'analisi si suddivide in quattro attività, esse sono la scrittura del diagramma di Gantt, l'analisi del dominio, l'analisi dei requisiti e l'analisi dei costi. La prima operazione è quella che a parer mio richiede più tempo e quindi le ho assegnato mezza giornata di lavoro composta da quattro ore scolastiche.

La seconda attività è piuttosto facile e veloce da fare ed ho quindi previsto di impiegarci circa due ore e un quarto considerando la revisione finale dello scritto.

L'analisi dei requisiti è anche piuttosto veloce da completare e quindi le ho assegnato il resto della giornata rimanente dopo aver fatto la seconda operazione.

La lezione successiva, il 10.09.2019, ho previsto di iniziaria con l'ultima attività compresa nell'analisi che è l'analisi dei costi che comprende l'analisi dei mezzi necessari per lo sviluppo del progetto, i software utilizzati e l'hardware necessario.

### 2.4.2 Progettazione

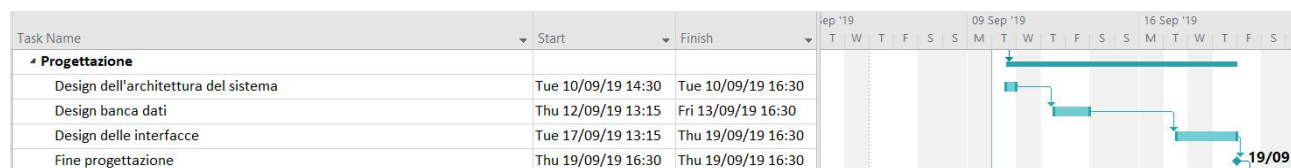


Figure 4 Gantt capitolo Progettazione

La progettazione è stata suddivisa in quattro attività di cui l'ultima rappresenta soltanto la milestone che indica la fine della progettazione e l'inizio dell'implementazione.

Il 10.09.2019, dopo aver finito l'ultimo incarico di analisi, ho previsto di creare il design dell'architettura del sistema. Lo schema non dovrebbe prendere troppo tempo dato che sono abbastanza in chiaro sul flusso di lavoro del sito. Fatto ciò ho occupato la giornata seguente con il design della banca dati, questo perché può prendere abbastanza tanto tempo per arrivare ad una forma ottimizzata.

Infine, come ultima attività della progettazione, vi è il design delle interfacce che potrebbe prendere abbastanza tanto tempo a dipendenza di quanto in fretta riesco a sviluppare la base di tutte le pagine in modo da renderle semplici e veloci da comprendere.

### 2.4.3 Implementazione

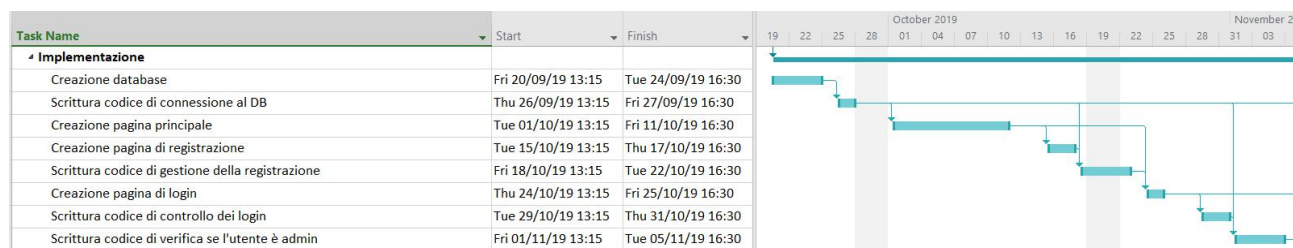


Figure 5 Gantt capitolo implementazione prima parte

L'implementazione, essendo la parte più lunga e complessa del progetto, è stata suddivisa in due parti così da semplificarne la spiegazione. La prima parte comprende otto attività ed è previsto che si prolunghi fino al 05.11.19.

Inizialmente si crea il database che era stato progettato in precedenza e si sviluppa il codice che andrà a collegare esso con le pagine web.

Quando il database è terminato si può iniziare a sviluppare le pagine vere e proprie iniziando da quella principale che, secondo le previsioni, è quella che occupa più tempo. Essa comprende infatti tutta la mappa interattiva e la sezione di ricerca.

Finita la prima pagina si passa a quella di registrazione e una volta terminata si scriverà il codice che verificherà l'inserimento dell'utente sia front end che back end. La stessa procedura avviene in seguito con la pagina di login che utilizza un codice molto simile a quello utilizzato per la registrazione.

Come ultima attività della prima parte di implementazione vi è la scrittura del codice che verifica se un utente è admin o meno così da consentirgli, se possiede tutti i permessi, l'accesso alla sezione dedicata agli amministratori.

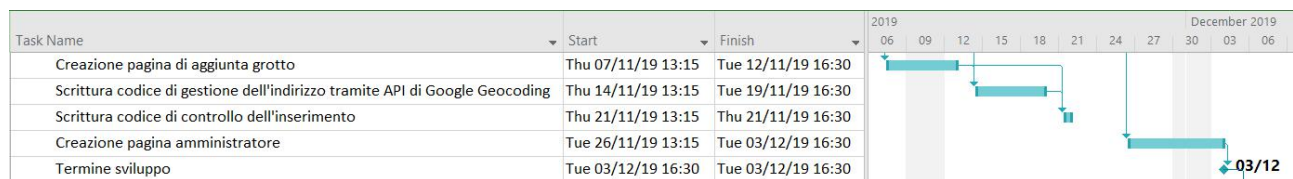


Figure 6 Gantt capitolo implementazione seconda parte

La seconda parte d'implementazione comprende quattro attività ed una milestone che rappresenta il termine dello sviluppo.

La prima delle attività consiste nel creare la pagina che gestisce l'inserimento di un grotto da parte degli utenti, con essa bisogna sviluppare il codice che verifica ciò che l'utente scrive e il codice che gestisce la trasformazione da indirizzo a coordinate fatto grazie alle API di Google Maps.

L'ultima pagina da creare è quella dedicata agli admin che prevedo sia piuttosto complicata e per questo l'attività dura parecchio tempo. Essa deve gestire le creazioni, eliminazioni e modifiche di utenti e grotti ed inoltre deve contenere una sezione dedicata alla verifica degli inserimenti da parte degli utenti.

#### 2.4.4 Test

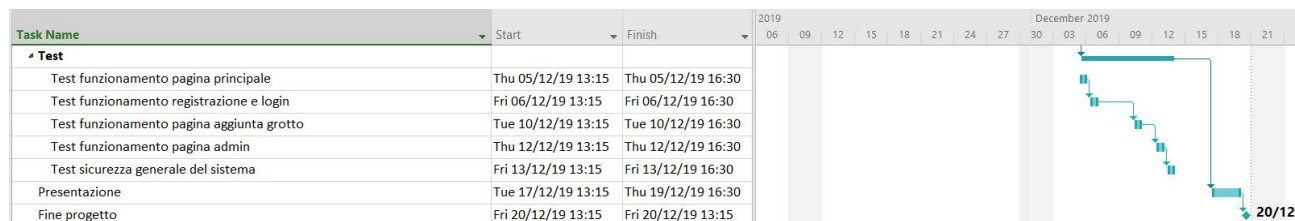


Figure 7 Gantt capitolo Test

L'ultima parte del progetto sono i test, essi saranno test di funzionamento generale e mirano a verificare che l'integrazione delle pagine funzioni anche se si fanno volutamente cose sbagliate. I test procedono in maniera piuttosto lineare iniziando a controllare la prima pagina per poi procedere in ordine di creazione. Infine vi sarà un test generale per controllare la sicurezza del progetto. Alla fine dei test vi sono due lezioni che vengono utilizzate per creare la presentazione ed infine vi è una milestone che rappresenta il termine del progetto.

### 2.5 Analisi dei mezzi

#### 2.5.1 Software

I software che sono stati utilizzati per questo progetto sono i seguenti:

- JetBrains PhpStorm 2019 2.3 utilizzato per scrivere il codice
- MySQL Workbench 6.3.8 utilizzato per gestire il database
- Google Chrome 77 utilizzato per testare il progetto
- Mozilla Firefox 69 utilizzato per testare il progetto
- WPS Writer 11.1 utilizzato per redigere la documentazione
- WPS Presentation 11.1 utilizzato per fare la presentazione
- Microsoft Project 2010 utilizzato per generare il diagramma di Gantt

Mentre le librerie utilizzate sono le seguenti:

- Material Design Bootstrap 4.8.7 utilizzata per la grafica del sito e dei controlli javascript
- API Google Maps Javascript 3.38 utilizzata per generare la mappa
- API Google Geocoding 3.1 utilizzata per calcolare le coordinate dall'indirizzo
- PHPMailer 6.1 utilizzata per inviare le email
- Auxiliary-rater 1.0 utilizzata per generare i campi di voto con le stelle
- Smoothie-js 1.0 utilizzata per rendere animato lo scroll nella pagina per gli admin

#### 2.5.2 Hardware

L'hardware su cui è stato svolto il progetto è un laptop Dell XPS 15 9570 con installato Ubuntu 18.04 come sistema operativo. Il prodotto può essere fatto funzionare su una qualsiasi macchina che abbia un webserver funzionante che funzioni con il pattern MVC di PHP, quindi con il modulo di rewrite attivato e l'override delle cartelle consentito.

### 3 Progettazione

#### 3.1 Design dell'architettura del sistema

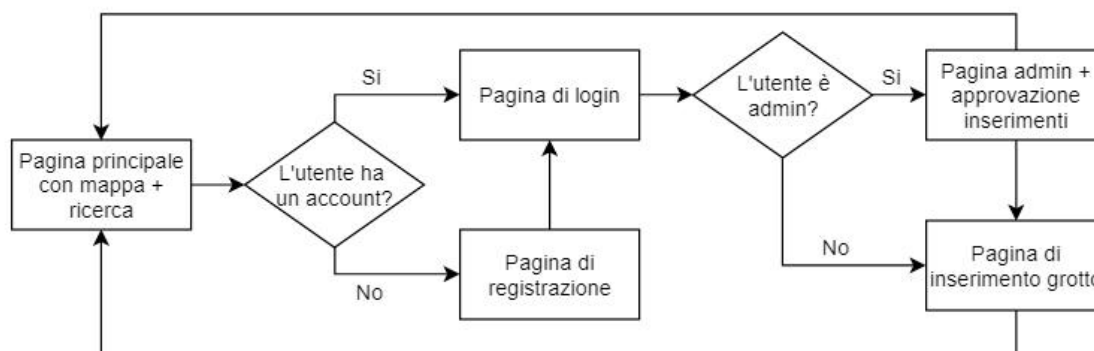


Figure 8 Design del sistema

Il design del sistema di questo progetto è piuttosto semplice, l'utente inizia collegandosi alla pagina principale che sarà quella contenente la mappa interattiva e la sezione di ricerca dei grotti con possibilità di filtrare i risultati. Da questa pagina l'utilizzatore del sito potrà decidere se restare dove si trova, in caso necessiti solo di eseguire una ricerca, oppure di spostarsi verso la pagina di login. Se esso non possiede un account avrà la possibilità di crearne uno, grazie ad un pulsante che lo porterà alla sezione di registrazione, altrimenti potrà accedere al suo account. Se verrà per caso dimenticata la password ci sarà un bottone che consentirà di ripristinarla.

Dopo aver eseguito il login, se l'utente possederà i privilegi di amministratore, si troverà nella pagina riservata agli admin che consente di creare, modificare e eliminare grotti ed utenti e anche di approvare o rifiutare i grotti inseriti dagli utenti. Se vorrà potrà poi spostarsi alla pagina di creazione dei grotti.

Se l'utente non è un amministratore potrà solo visualizzare la pagina di aggiunta di una località. In tutte le parti e sezioni del sito vi sarà la possibilità di navigare verso tutte le pagine grazie ad un menu.

### 3.2 Design dei dati e database

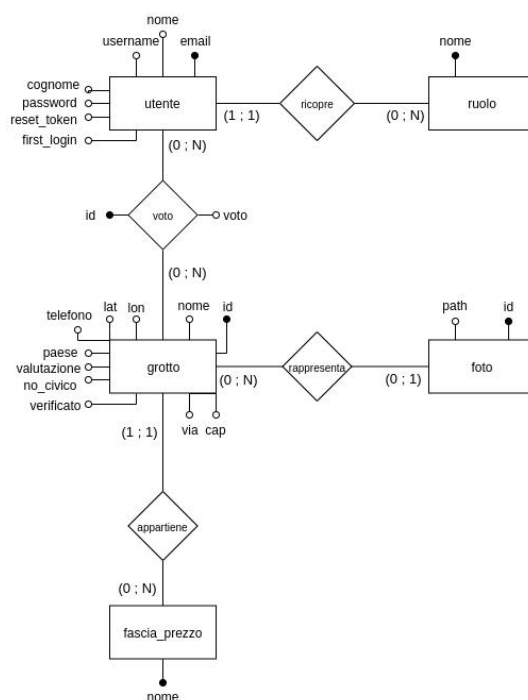


Figure 9 Diagramma ER del database

Il diagramma ER del database è piuttosto semplice come si può vedere dall'immagine soprastante. Esso comprende cinque tabelle che rappresentano rispettivamente il **ruolo** che uno user possiede quindi admin o normale, l'**utente** con tutti i suoi attributi, il **grotto**, le **foto** che possono essere collegate ad una località e la **fascia di prezzo** di un ristorante.

La tabella ruolo poteva essere sostituita da due attributi all'interno di utente ma così facendo si avrebbero problemi se un giorno si volesse inserire un nuovo incarico come ad esempio il gerente di un grotto. Un utente può avere solo un ruolo mentre un ruolo può essere ricoperto da zero a molti utenti.

La tabella utente comprende le informazioni di base di una persona e utilizza l'email come chiave dato che essa è sicuramente univoca.

La tabella grotto contiene un id che identifica la località e poi le informazioni sulla sua posizione sia in coordinate che come indirizzo, questo perché contattare tutte le volte le API di Google renderebbe il sito lento. Inoltre comprende un attributo rappresentante la fascia di prezzo in cui si trova il grotto ed uno con la valutazione di esso.

L'ultima tabella contiene le foto che vengono identificate da un id e possiedono un titolo e un percorso locale. Un grotto può possedere molte immagini e un'immagine può essere associata ad un solo grotto.

RUOLO(nome)

UTENTE(email, nome, cognome, username, password, reset\_token, first\_login\*, nome\_ruolo(FK))

FASCIA\_PREZZO(nome)

GROTTO(id, via, paese, cap, no\_civico, lat, lon, nome, valutazione\*, telefono, verificato, fascia\_prezzo(FK))

FOTO(id, path, id\_grotto\*(FK)) *path unique*

VOTO(id, email\_utente(FK), id\_grotto(FK), voto)

Figure 10 Schema logico del database

Lo schema logico rende chiara un'ultima particolarità del database, l'associazione vota dello schema ER viene tradotta in una tabella che contiene come chiavi esterne le chiavi di utente e grotto ed ha inoltre un attributo voto.

**Tabella ruolo**

Attributo	Tipo	Descrizione
nome	varchar(25)	Chiave primaria, rappresenta il nome del ruolo.

**Tabella utente**

Attributo	Tipo	Descrizione
email	varchar(50)	Chiave primaria, rappresenta il l'email dell'utente.
username	varchar(50)	Rappresenta il nome utente.
nome	varchar(50)	Rappresenta il nome di battesimo dell'utente.
cognome	varchar(50)	Rappresenta il cognome dell'utente.
password	varchar(64)	Rappresenta la password dell'utente, verrà salvato l'hash del valore.

**Tabella grotto**

Attributo	Tipo	Descrizione
id	int	Chiave primaria, l'identificatore del grotto
nome	varchar(50)	Il nome del grotto
lon	double	La coordinata longitudine del grotto
lat	double	La coordinata latitudine del grotto
via	varchar(50)	La via in cui si situa il grotto
paese	varchar(50)	Il paese (comune) in cui si situa il grotto
cap	int	Il cap del paese
no_civico	varchar(10)	Il numero civico
fascia_prezzo	enum	La fascia di prezzo (buon mercato/nella norma/caro)
valutazione	int	La valutazione ricevuta

**Tabella foto**

Attributo	Tipo	Descrizione
id	int	Chiave primaria, l'identificatore della foto
titolo	varchar(50)	Il titolo dell'immagine
path	varchar(50)	Il percorso assoluto dell'immagine sul server

**Tabella voto**

Attributo	Tipo	Descrizione
email_utente	varchar(50)	Chiave primaria, chiave esterna, l'email dell'utente
id_grotto	int	Chiave primaria, chiave esterna, l'identificatore del grotto
voto	int	Il voto assegnato

### 3.3 Design delle interfacce

#### 3.3.1 Pagina iniziale

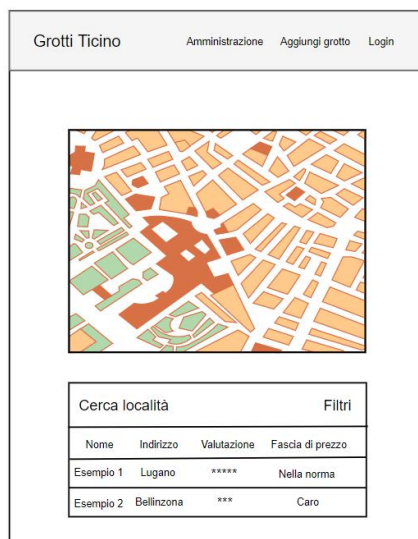


Figure 11 Mockup pagina Index

La pagina principale dovrà essere pulita e semplice da utilizzare, la mia idea era quella di suddividerla in due parti: la prima contenente la mappa interattiva che mostra le posizioni dei grotti mentre la seconda con la sezione di ricerca a cui potranno venire applicati dei filtri.

#### 3.3.2 Pagina di login

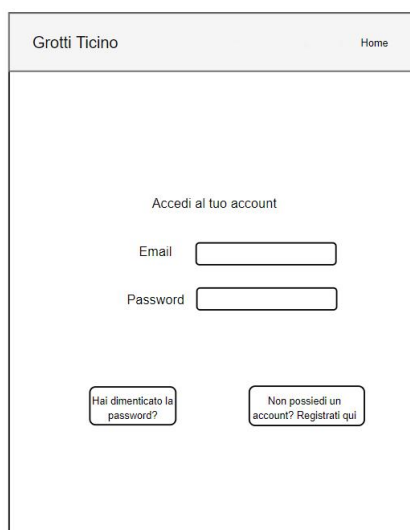
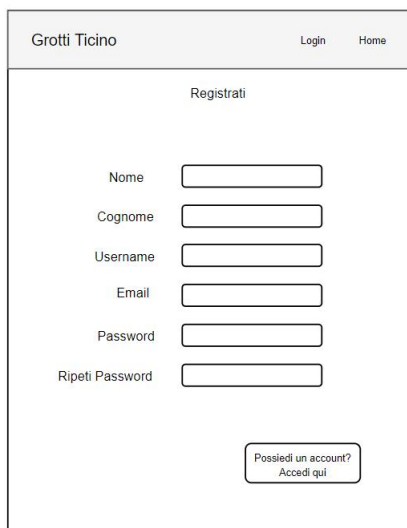


Figure 12 Mockup pagina di login

La pagina di login sarà molto semplice dato che conterrà solamente quattro elementi. Il primo sarà il campo per inserire l'email, il secondo sarà l'input per la password e poi vi saranno due bottoni. Il primo dei pulsanti servirà a cambiare la password del proprio account mentre il secondo consentirà di raggiungere la pagina di registrazione.

### 3.3.3 Pagina di registrazione



Il mockup della pagina di registrazione "Grotti Ticino" presenta una struttura semplice. In alto a sinistra c'è il titolo "Grotti Ticino", e in alto a destra ci sono i link "Login" e "Home". Al centro della pagina, sotto il titolo "Registrati", sono disposti sei campi di input etichettati: "Nome", "Cognome", "Username", "Email", "Password" e "Ripeti Password". In basso a destra, c'è un pulsante con il testo "Possiedi un account? Accedi qui".

Figure 13 Mockup pagina registrazione

La pagina di registrazione conterrà tutti gli input necessari a inserire i campi necessari per creare un account che sono:

- Nome
- Cognome
- Username
- Email
- Password
- Ripetizione della password



### 3.3.4 Pagina di creazione dei grotti

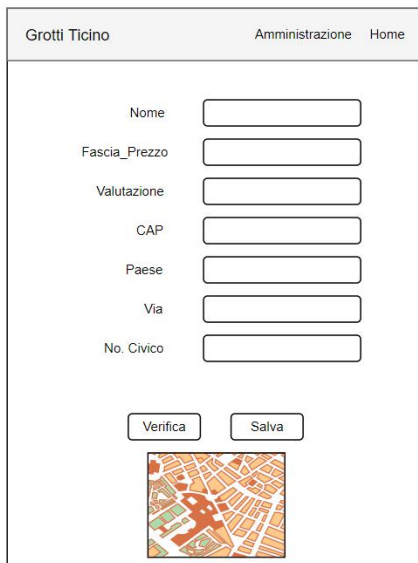


Figure 14 Mockup pagina creazione grotti

La pagina di creazione dei grotti conterrà tutti i campi necessari per la creazione di un grotto quindi:

- Nome
- Fascia di prezzo
- Valutazione
- CAP
- Paese
- Via
- Numero Civico

Alla fine dell'inserimento si potrà controllare tramite un bottone apposito se la posizione sulla mappa viene marcata nel punto corretto. Se tutto sarà stato inserito nella maniera giusta si potrà salvare l'inserimento che verrà passato agli admin da verificare.

### 3.3.5 Pagina di amministrazione

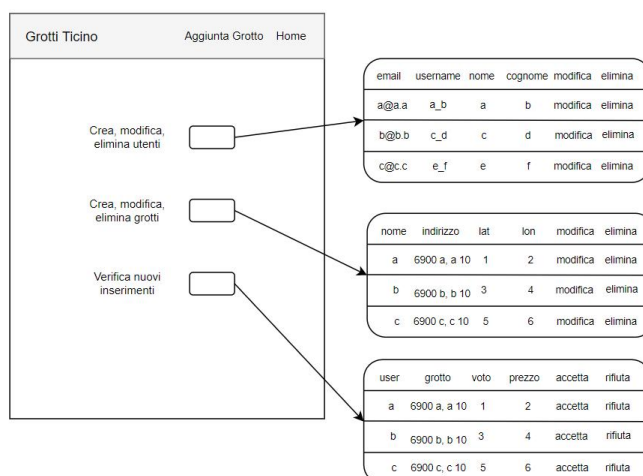


Figure 15 Mockup pagina admin

La pagina degli admin sarà di base molto vuota e semplice, conterrà tre bottoni che consentiranno di modificare gli utenti, i grotti o di verificare gli inserimenti. Una volta premuto su uno di questi bottoni si aprirà una finestra modale che mostrerà la lista dei campi che si vorranno gestire.

## 4 Implementazione

### 4.1 Struttura del progetto

Il progetto è basato su un pattern MVC e quindi tutto il codice è suddiviso nelle seguenti cartelle principali:

- views
- models
- controllers
- assets

Nella cartella **views** si trovano tutti i file contenenti la parte frontend del sito, essi sono suddivisi in ulteriori cartelle relative alla pagina che vanno a caricare. Questi file contengono principalmente tag di HTML e codice Javascript oltre ad alcuni controlli e inserimenti di dati in PHP.

La cartella **models** contiene codice PHP che si occupa di un compito preciso come gestire il database, verificare la validità dei dati o inviare email.

Nella sezione **controllers** vi sono dei file PHP che si occupano di mettere in comunicazione le views (front end) con i models (back end). Essi servono anche a caricare le views quando la pagina relativa viene richiamata.

L'ultima cartella contiene i fogli di stile della pagina così come le librerie utilizzate, i codici in Javascript e le immagini dei grotti che vengono mostrate quando si seleziona una località.

### 4.2 Database

Il database è stato sviluppato sulla base della progettazione ma nel corso del progetto ho subito alcune modifiche. Esso è suddiviso in sei tabelle che rappresentano rispettivamente i ruoli degli utenti, gli utenti, le fasce di prezzo dei grotti, i grotti, le immagini dei locali e le valutazioni di essi.

Le tabelle rappresentanti i ruoli e le fasce di prezzo sono state create così da semplificare una futura espansione del progetto dato che consentono di aggiungere un campo senza dover modificare nient'altro.

#### 4.2.1 Tabella utente

La tabella utente è una delle più importanti di tutto il database, essa contiene otto campi di cui i primi cinque (nome, cognome, username, email e password) saranno inseriti dall'utente alla registrazione mentre i seguenti tre (reset\_token, first\_login, nome\_ruolo) verranno usati per verificare gli stati dell'utente e valutare cosa mostrargli.

L'email rappresenta anche l'identificatore di un account dato che non ve ne possono essere due uguali. La password è salvata codificata utilizzando l'algoritmo sha256 per evitare problemi di sicurezza. Il campo reset\_token verrà compilato e inviato per email al momento che un utente vorrà cambiare la propria password per accertarsi che non sia un estraneo con intenzioni dannose. Il campo first\_login contiene un booleano che indicherà se l'utente è stato creato da un admin o meno, se questo è vero l'utente dovrà modificare la password al primo login.

#### 4.2.2 Tabella grotto

Nello schema rappresentante i grotti vi sono dodici campi, i primi sei (nome, via, cap, paese, fascia di prezzo e telefono) saranno valori inseriti da un utente alla creazione dello stesso. I campi lon e lat indicano le coordinate geografiche della località e serviranno a mostrare il luogo sulla mappa, verranno inserite automaticamente calcolandole dall'indirizzo passato dall'utente. Il campo verificato indica se un grotto è stato creato da un admin o da un utente, se lo ha creato quest'ultimo allora la località dovrà essere valutata da qualcuno con privilegi più elevati.

#### 4.2.3 Tabella foto

La tabella foto contiene i dati relativi alle immagini associate ai grotti, la path di essa indica il percorso relativo al pattern MVC dell'immagine. Il nome della foto è un hash del suo contenuto così che non vi siano problemi di caratteri speciali in esso e per evitare doppioni della stessa immagine.

#### 4.2.4 Tabella voto

La tabella voto contiene la valutazioni da uno a cinque fatte dagli utenti ai grotti. Essa contiene la chiave dell'utente, quella del grotto e il voto, ad ogni inserimento su di essa un trigger di MySQL farà la media di tutte le valutazioni del grotto che ne ha appena ricevuta una nuova e aggiornerà il campo valutazione nella tabella del grotto. A seguito il codice che esegue l'operazione appena spiegata.

delimiter \$\$\$

create trigger avg\_after\_review after insert on voto for each row

begin

set @avg\_review = (select avg(voto) from voto where id\_grotto=NEW.id\_grotto);

update grotto set valutazione = @avg\_review where id=NEW.id\_grotto;

end;

\$\$\$

delimiter ;

Figure 16 Codice di creazione del trigger MySQL

### 4.3 Pagina Home

La pagina home è suddivisa in due sezioni e comprende due dei requisiti principali che sono una mappa interattiva per visualizzare i grotti e una lista di essi in cui poterli ordinare per i vari campi.

Nella sezione superiore vi è la mappa generata grazie alle credenziali di Google Maps, essa al caricamento della pagina andrà a prendere tutti i grotti caricati dal database e genererà per ogni uno di essi un puntatore alle coordinate corrispondenti. Se si premerà sul puntatore verrà mostrata, dopo una breve animazione di zoom, una finestra modale contenente le informazioni di base relative alla località scelta. Il codice per creare la mappa e tutti i puntatori è in Javascript.

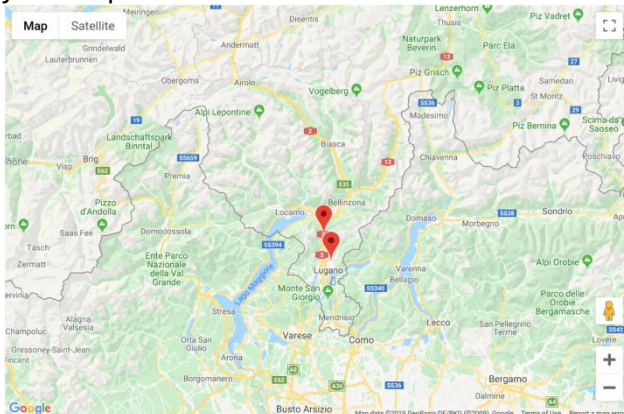


Figure 17 Homepage, mappa con puntatori

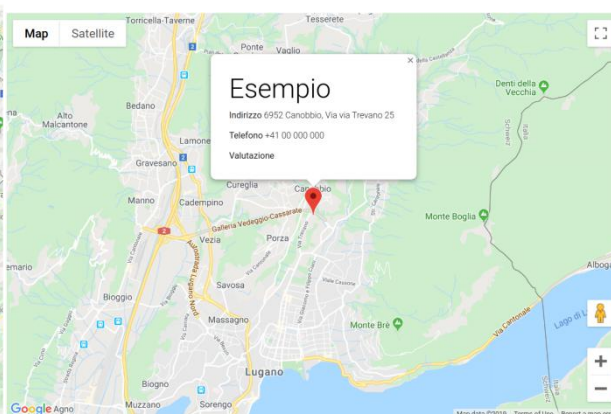


Figure 18 Homepage, mappa con grotto selezionato

Nella sezione di pagina inferiore si trova la tabella che consente di visualizzare tutti i grotti e filtrarli per uno qualsiasi dei loro campi. La tabella è stata creata utilizzando le DataTables di Material Design Bootstrap ed è impostata per mostrare 15 località per pagina. È possibile filtrare i campi semplicemente premendo sui nomi delle colonne evidenziati in grassetto.

Nome	Indirizzo	Telefono	Fascia di Prezzo	Valutazione
Primo esempio	6952 Canobbio, Via Trevano 25	+41 00 000 000	Caro	☆☆☆☆☆
Esempio	6952 Canobbio, Via via Trevano 25	+41 00 000 000	Nella norma	☆☆☆☆☆

Figure 19 Homepage, tabella dei grotti

#### 4.4 Pagina Grotto

La pagina grotto serve a mostrare tutti i dettagli di una località specifica e di dare la possibilità agli utenti che hanno eseguito il login di votare un ristorante e di assegnargli delle immagini. Essa è strutturata in maniera dinamica, se vi sono immagini all'inizio genera un carosello che le mostrerà cambiando foto a intervalli di cinque secondi mentre se non ve ne saranno assegnate a quel grotto non lo mostrerà. Sotto il carosello vi sarà una lista contenente tutte le informazioni della località ed in seguito, se sarà stato eseguito il login, vi saranno le sezioni per votare e aggiungere foto.

##### Contatti

Telefono: +41 00 000 000

##### Dove siamo

Indirizzo: 6952 Canobbio Via Trevano 25

##### Valutazioni

Fascia di prezzo: Caro

Valutazione media (0 utenti hanno votato questa località):

☆☆☆☆

Figure 20 Pagina grotto, informazioni

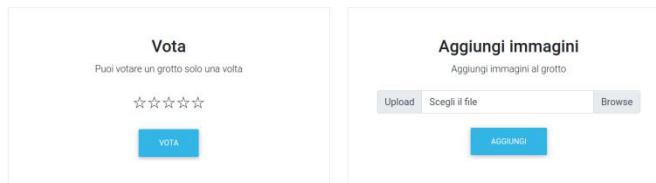


Figure 21 Pagina grotto, valutazione e aggiunta immagini

#### 4.5 Pagina di login

La pagina di login è strutturata in maniera semplice e contiene un form che consentirà di collegarsi al proprio account inserendo l'email e la password, di raggiungere la pagina per la registrazione oppure di cambiare la password del proprio account.

Per verificare se un utente può collegarsi o meno vengono caricati tutti gli utenti dal database e si verifica se ne esiste uno con l'email inserita nel campo apposito, questo viene fatto richiamando il model DBConnection che gestisce la connessione con il database, se l'hash della password inserita equivale a quello salvato nel database e che tipo di permessi ha l'utente. Se vi è stato un errore la pagina verrà ricaricata mostrando l'errore in basso al form e ricompilando automaticamente i campi riempiti in precedenza.

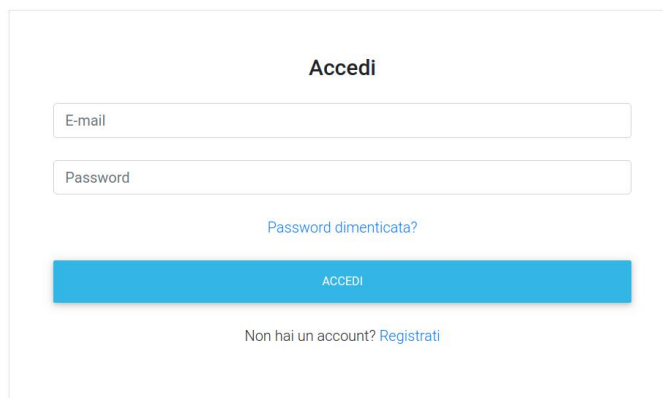


Figure 22 Pagina di login, form di connessione

#### 4.6 Pagina di registrazione

La pagina di registrazione consentirà ad un utente di creare un nuovo account, essa contiene un form con i campi per inserire il nome, il cognome, lo username, l'email e la password ripetuta due volte per sicurezza.

Per generare un nuovo account si verifica che le password inserite siano uguali e che non ne esista già uno con quell'email. Se vi è stato un errore la pagina verrà ricaricata mostrando l'errore in basso al form e ricompilando automaticamente i campi riempiti in precedenza. Una volta eseguita la registrazione con successo si verrà mandati alla pagina di login per accedere con l'utente appena generato.

Figure 23 Pagina di registrazione, form d'inserimento

#### 4.7 Pagina di ripristino della password

La pagina di ripristino della password è accessibile dalla pagina di login e consente di inserire l'email dell'account alla quali si vuole reimpostare la password. Una volta premuto il tasto di invio verrà impostato un token nel campo reset\_token dell'utente composto da 15 bytes di caratteri casuali che verrà anche inviato per email all'indirizzo inserito in precedenza. Quando si riceverà il messaggio di posta elettronica basterà premere sul link contenente il token e se il token sarà corrispondente a quello nel database l'utente potrà inserire una nuova password.

Figure 24 Pagina di reset, invio email

Figure 25 Pagina di reset, cambio credenziali

#### 4.8 Pagina di aggiunta grotto

La pagina di aggiunta grotto è l'unica che possono vedere in più gli utenti che hanno effettuato il login, con permessi di base. Essa contiene un form in cui inserire tutte le caratteristiche della località e sotto di esso vi è una mappa che alla pressione del testo verifica genera un puntatore nell'indirizzo inserito così da essere sicuri che, una volta generato il grotto, nella pagina principale il puntatore sarà nel punto giusto. Il grotto creato dovrà, se non si è admin, venire verificato da un amministratore del sito prima di essere mostrato.

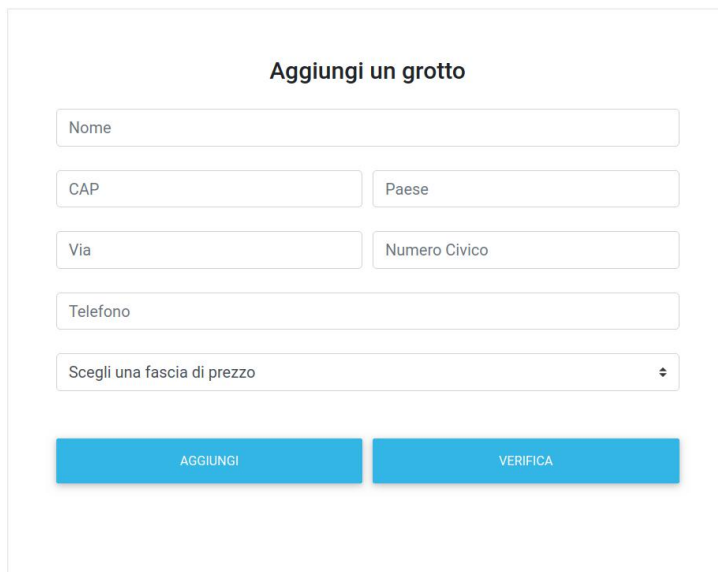


Figure 26 Pagina di aggiunta grotti, form d'inserimento

#### 4.9 Pagina di amministrazione

La pagina di amministrazione è riservata a utenti con privilegi elevati ed è suddivisa in quattro sezioni: una per gestire gli utenti, una per gestire i grotti, una per verificare i grotti inseriti dagli utenti normali e una per gestire le immagini. Le varie sezioni sono raggiungibili scendendo nella pagina o utilizzando il menu sotto il titolo. Tutte le tabelle contenute nella pagina sono state fatte con delle DataTables di Material Design Bootstrap così da poter inserire i controlli per la paginazione e per l'ordinamento dei campi.

### Amministrazione grotti Ticino

Connesso come: Matteo Forni (matteoforni)

---

Utenti   Grotti   Inserimenti   Immagini

---

Figure 27 Pagina admin, lista delle sezioni



#### 4.9.1 Sezione utenti

La sezione per la gestione degli utenti contiene una tabella che ne visualizza cinque per pagina e consente di ordinarli in base a uno dei campi della tabella che sono il nome, il cognome, l'email, lo username e il ruolo. Come ultime due colonne della tabella vi sono due bottoni che consentono rispettivamente di modificare o eliminare un utente, non si può eliminare un admin se non ve ne è almeno un altro. Se si selezionerà nella tabella di modificare un utente verrà mostrato un form compilato con i valori attuali in cui sarà possibile cambiare tutti i campi, tranne l'email, e se verrà modificata la password verrà inviata un email con la nuova password all'utente in questione.

Inoltre sotto la tabella vi è un riquadro contenente un link che porta ad una pagina in cui si può creare un utente tramite un form d'inserimento dei campi principali di un account. Quando esso è creato da un admin verrà inviata un email all'indirizzo inserito contenente una password provvisoria che andrà cambiata al primo accesso.



Figure 28 Pagina admin, sezione utenti

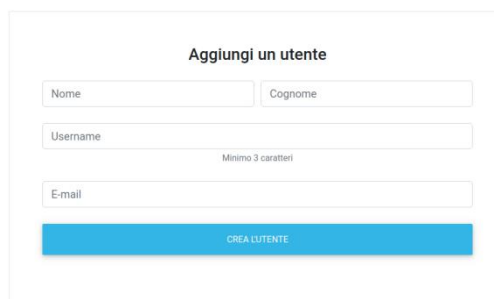


Figure 29 Pagina admin, aggiunta utente

#### 4.9.2 Sezione grotti

La sezione per la gestione dei grotti contiene una tabella, contenente tutti i grotti già verificati, che ne mostra cinque per volta e consente di ordinarli a seconda di una delle colonne di essa. Nella tabella vengono mostrati il nome, l'indirizzo, la fascia di prezzo, il numero di telefono e la valutazione media della località. Gli ultimi due campi servono per modificare o eliminare il grotto. Se si selezionerà il tasto per modificare un ristorante verrà mostrato un form compilato con i campi attuali dove sarà possibile modificarli. Sotto il form sarà presente una mappa per verificare l'indirizzo inserito.

Sotto la tabella è presente un riquadro contenente un link che porta alla pagina di creazione di una grotto.



Figure 30 Pagina admin, sezione grotti

#### 4.9.3 Sezione inserimenti

La sezione per la gestione degli inserimenti contiene una tabella con tutti i grotti che devono ancora essere accettati perché inseriti da utenti senza privilegi d'amministratore. La tabella contiene li stessi campi della tabella nella sezione per le località già verificate con l'unica differenza che al posto del tasto modifica vi è quello per accettare il campo, se questo verrà accettato non sarà più mostrato nella tabella inserimenti ma in quella dei grotti e verrà inoltre mostrato nella mappa e nella lista presenti nella prima pagina.

Gestione Inserimenti

Nome	Indirizzo	Telefono	Fascia di Prezzo	Accetta	Elimina
Esempio d'inserimento	6952 Canobbio, via Trevano 25	+41 00 000 000	Caro	<a href="#">ACCETTA</a>	<a href="#">ELIMINA</a>

Previous [1](#) Next

Figure 31 Pagina admin, sezione inserimenti

#### 4.9.4 Sezione immagini

L'ultima sezione è quella di gestione delle immagini e contiene anch'essa una tabella che rappresenterà cinque foto per volta. Nella tabella sarà possibile ordinare i campi in base ad una delle colonne che sono l'id, il percorso relativo al pattern MVC e il nome del grotto alla quale sono assegnate. Premendo sul campo si potrà vedere un'anteprima dell'immagine in una finestra modale. Nell'ultima colonna vi sarà un bottone che consentirà di eliminare un bottone.

Gestione Immagini

ID	Path	Nome del Grotto	Elimina
19	/application/assets/img/1c1b3f6afd5a9e2a317ff84d8c22cd93.png	Primo esempio	<a href="#">ELIMINA</a>

Previous [1](#) Next

Figure 32 Pagina admin, sezione immagini

#### 4.10 Pagina di warning

Nel caso avvenisse un errore di qualunque genere con il server o con la connessione al database l'utente verrà riportato ad una pagina che mostra l'errore ritornato più una frase di facile comprensione per i meno esperti.

## C'è stato un errore!

Prova a ricaricare la pagina, se l'errore si ripete aspetta un'attimo.

Messaggio e codice d'errore: 2002 - SQLSTATE[HY000] [2002] Connection refused

Figure 33 Pagina d'errore, errore connessione al database

## 4.11 Models

Nella cartella models sono contenute le classi di comunicazione con il database, vi sono infatti una classe per ogni tabella ed esse contengono tutti i metodi che vanno a leggere, scrivere o modificare dei campi del relativo schema. Oltre alle classi che si occupano dei dati ce ne sono altre tre: una incaricata per la connessione con il database, una che gestisce l'invio di email e l'ultima che si occupa di verificare e "pulire" gli inserimenti degli utenti così da evitare che delle stringhe malevoli vengano inserite.

Tutte le classi relative alle tabelle del database hanno tutte la stessa nomenclatura secondo la regola: <nome della tabella>\_model. Esse contengono codice molto simile fra loro infatti tutte nel costruttore richiamano il metodo *"getConnection"* della classe DBConnection e poi contengono metodi che eseguono le query necessarie sulla tabella. I metodi che vanno a lavorare sui dati se eseguono un inserimento o una verifica utilizzano i costrutti di PDO *"bindParam"* che consentono di evitare le SQL injections, inoltre tutte le funzioni se generano un errore riporteranno alla pagina d'errore che mostrerà un testo di facile comprensione, per un utente inesperto, più il messaggio d'errore ritornato nel caso servisse ad uno degli admin.

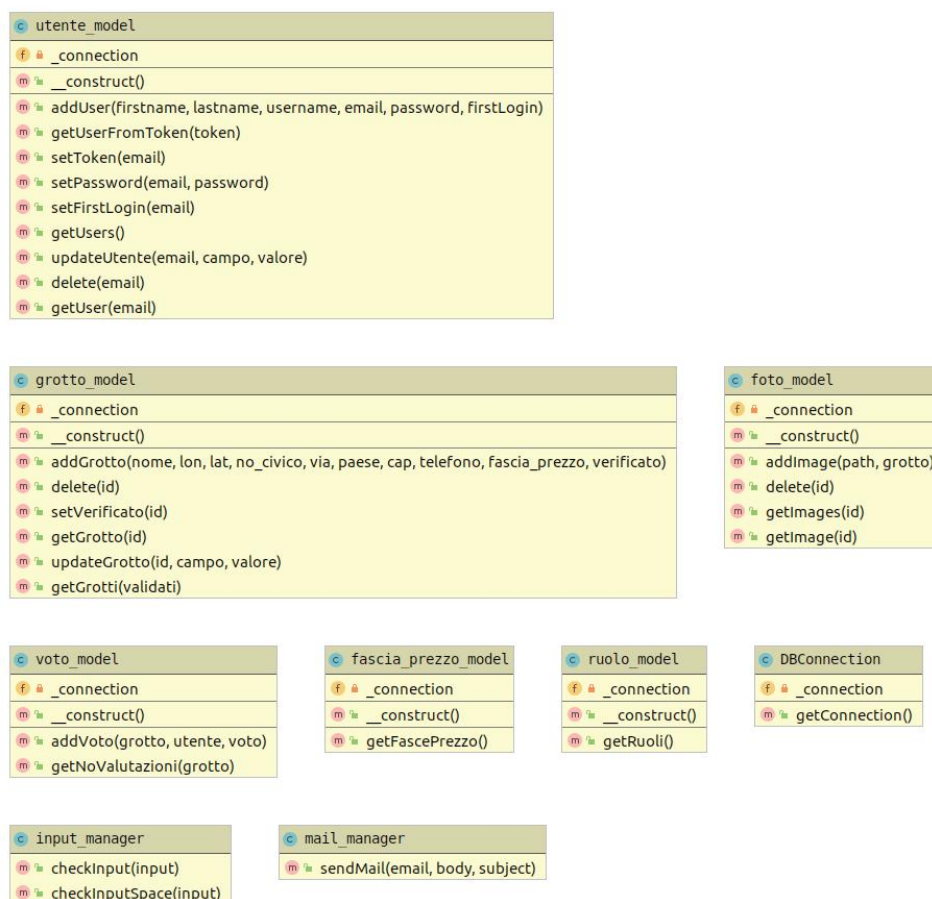


Figure 34 Diagramma UML delle classi nella cartella models

#### 4.11.1 Classe DBConnection

La classe DBConnection è quella che esegue la connessione con il database e contiene solamente il metodo “*getConnection*” che genera, se non esiste già, una connessione e la ritorna così da poter far query sul database.

```
public function getConnection() {
    if($this->_connection == null){
        try{
            $this->_connection = new PDO(DSN, USER, PASSWORD);
            return $this->_connection;
        }catch(PDOException $e){
            ...
        }
    }
}
```

Figure 35 Codice di connessione al database

#### 4.11.2 Classe input\_manager

La classe input\_manager contiene due funzioni e servono entrambe ad eliminare caratteri speciali e a formattare l’inserimento degli utenti. La differenza tra i due metodi è che “*checkInput*” elimina gli spazi dalla stringa passata mentre “*checkInputSpace*” esegue tutti i controlli come il precedente senza eliminare gli spazi. Questo viene fatto perché vi sono alcune stringhe tipo i nomi o gli indirizzi in cui molto spesso gli spazi sono necessari.

#### 4.11.3 Classe mail\_manager

Il file mail\_manager contiene la classe corrispondente e consente tramite il metodo “*sendMail*” di inviare delle email specificando il soggetto, il contenuto e il destinatario. L’invio delle email è fatto basandosi sulla libreria “*phpmailer*” ed il server SMTP di Google.

#### 4.11.4 Classe utente\_model

La classe *utente\_model* serve a comunicare con il database ed esegue le operazioni di base sulla tabella relativa.

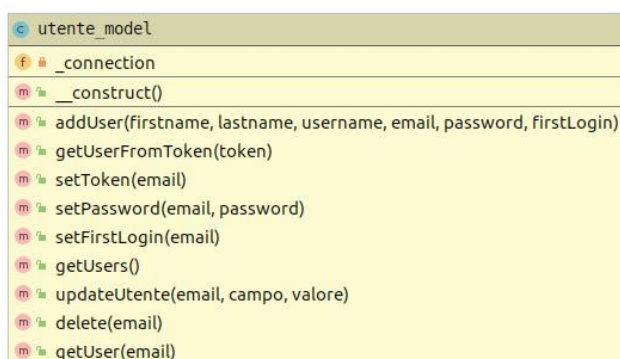


Figure 36 Classe utente model

#### 4.11.4.1 Metodi di lettura

I metodi che consentono di leggere dalla tabella utente sono i seguenti:

- `getUsers()`
- `getUser(email)`
- `getUserFromToken(token)`

Il primo consente di ritornare tutti gli utenti contenuti nello schema mentre il secondo ritorna l'utente, se esiste, con l'email uguale a quella passata come parametro. Il terzo metodo serve quando viene eseguito un reset della password e ritorna l'utente, se esiste, con impostato il token passato come parametro.

#### 4.11.4.2 Metodi di scrittura

La funzione `addUser` consente di scrivere un nuovo dato nel database, il metodo richiede di passare il nome, il cognome, lo username, l'email, la password e il booleano che rappresenta se al primo login dovrà cambiare password o meno. Il campo `firstLogin` verrà messo a zero se l'utente si sarà creato da solo tramite la pagina di registrazione mentre se sarà stato creato da un admin verrà impostato a uno.

#### 4.11.4.3 Metodi di modifica

I metodi seguenti consentono di modificare una riga della tabella:

- `setToken(email)`
- `setPassword(email, password)`
- `setFirstLogin(email)`
- `updateUtente(email, campo, valore)`

Il primo metodo imposta un nuovo token di reset della password all'utente generandolo in maniera casuale, il secondo serve ad impostare una nuova password all'account con l'email specificata, il terzo metodo imposta il booleano che fa sì che l'utente cambi la password a zero, questo avviene dopo che la password sia stata modificata con successo. L'ultima funzione consente di aggiornare l'utente in maniera più generale e la utilizzano gli admin, essa va a modificare il campo specificato dall'attributo corrispondente inserendovi il valore voluto. Può aggiornare tutti i campi dell'utente meno che l'email che non può mai essere cambiata.

#### 4.11.4.4 Metodi di eliminazione

La funzione `delete` consente di eliminare un utente secondo l'email passata, quando esso viene eliminato verranno cancellati di conseguenza tutti i suoi voti assegnati a dei grotti.

#### 4.11.5 Classe grotto\_model

La classe `grotto_model` serve ad eseguire le operazioni di scrittura, lettura, modifica ed eliminazione sulla tabella relativa.

```
class grotto_model
{
    _connection
    __construct()
    addGrotto(nome, lon, lat, no_civico, via, paese, cap, telefono, fascia_prezzo, verificato)
    delete(id)
    setVerificato(id)
    getGrotto(id)
    updateGrotto(id, campo, valore)
    getGrotti(validati)
}
```

Figure 37 Classe grotto model

#### 4.11.5.1 Metodi di lettura

I metodi che consentono di leggere dei dati dallo schema sono i seguenti:

- `getGrotto(id)`
- `getGrotti()`

Il primo ritorna un array contenente i dati del grotto cercato mentre il secondo genera una matrice di tutti i grotti con i relativi dati.

#### 4.11.5.2 Metodi di scrittura

La funzione che si occupa di inserire dei dati all'interno della tabella è *addGrotto*, essa consente, passando tutti i dati, di aggiungere una località. I campi *lon* e *lat* vengono generati automaticamente dalle API di Geocoding di Google quando un utente inserisce l'indirizzo completo e invia i dati.

#### 4.11.5.3 Metodi di modifica

I metodi che consentono di modificare un ristorante sono i due elencati di seguito:

- `updateGrotto(id, campo, valore)`
- `setVerificato(id)`

La prima funzione serve a modificare un qualsiasi campo all'interno della tabella e viene utilizzato dagli admin quando modificano una località nella pagina a loro riservata. Il secondo consente di impostare il campo booleano *verificato* in un grotto e viene richiamato quando un amministratore accetta l'inserimento di un utente con privilegi di base.

#### 4.11.5.4 Metodi di eliminazione

Il metodo che consente di rimuovere una riga dal database è chiamato *delete* e riceve come parametro l'identificatore del grotto da eliminare. Quando un grotto viene eliminato vengono rimossi dal database anche i voti relativi ad esso e le sue immagini.

#### 4.11.6 Classe foto\_model

La classe *foto\_model* consente di eseguire tutte le operazioni necessarie sulla tabella relativa alle immagini dei grotti.

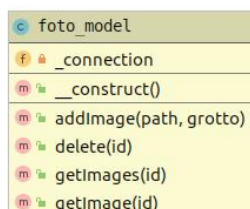


Figure 38 Classe foto model

##### 4.11.6.1 Metodi di lettura

La classe comprende, come molte delle classi già viste in precedenza, due metodi per leggere la tabella *foto*. Il primo consente di ottenere tutte le immagini contenute organizzate in una matrice mentre il secondo ritorna solamente un array con all'interno i dati della foto con l'id passato come parametro.

#### 4.11.6.2 Metodi di scrittura

La funzione che si occupa di inserire una nuova immagine nella tabella si chiama *addImage* e richiede come parametri il percorso di dove è salvata l'immagine e il grotto alla quale deve essere associata.

#### 4.11.6.3 Metodi di eliminazione

Il metodo *delete* consente di eliminare una foto dal database e richiede l'identificatore dell'immagine che si vuole rimuovere.

#### 4.11.7 Classe voto\_model

La classe che gestisce i voti nel database contiene solamente due metodi che sono rispettivamente *addVoto* e *getNoValutazioni*, il primo consente di inserire una nuova votazione nella tabella passando come parametro gli identificatori del grotto che l'ha ricevuta, dell'utente che l'ha fatta e il valore inserito. I voti possono andare da zero a cinque con la precisione al mezzo punto.

La seconda funzione ritorna il numero di votazioni che sono state fatta su un determinato grotto per sapere se la media dei voti è veritiera o se è basata su pochi dati.

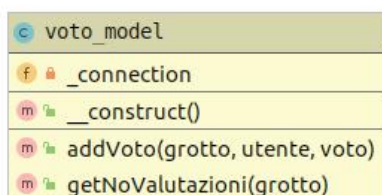


Figure 39 Classe voto model

#### 4.11.8 Classe fascia\_prezzo\_model

La classe *fascia\_prezzo\_model* si occupa di gestire la tabella relativa del database, essa contiene un solo metodo che consente di ottenere tutte le fasce di prezzo, non vi sono funzioni per l'aggiunta perché esse sono inserite a mano dallo sviluppatore sulla base dei requisiti del progetto e non verranno più modificate se non in caso di una nuova richiesta del cliente.

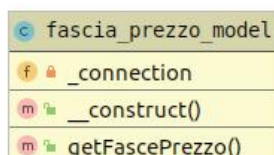


Figure 40 Classe fascia di prezzo model



#### 4.11.9 Classe ruolo\_model

La classe di gestione dei ruoli consente di eseguire la lettura dal database dei ruoli che un utente può avere così da assegnargli i giusti permessi. Essa contiene solamente una funzione che carica tutti ruoli perché, secondo i requisiti del progetto, vi sono solamente due ruoli e non ci è la necessità di aggiungerne. Se in futuro si dovesse inserire un nuovo tipo di utente basterà fare un inserimento nella tabella.

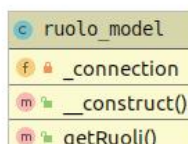


Figure 41 Classe ruolo model

#### 4.12 Controllers

Le classi situate nella cartella *controllers* del progetto servono a elaborare e passare i dati tra *views* e *models* e viceversa, quindi essi ricevono gli input dell'utente e si occupano di verificarli e passarli alle classi che eseguiranno le operazioni richieste sul database con quei valori e fatto ciò ricavano i dati ricavati dal database e li passano alle pagine così che esse possano mostrarli all'utente. Ogni pagina ha un relativo controller che la gestisce e la carica tramite il metodo *index*, oltre alla pagina la funzione carica anche l'header in base all'utente collegato e il footer.

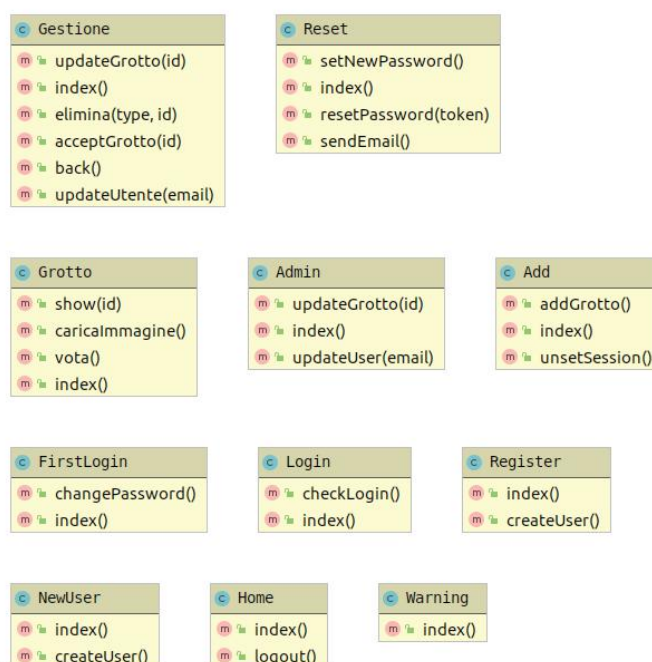


Figure 42 Diagramma UML dei controllers



#### 4.12.1 Controller home

Il controller *Home* è la classe che si occupa di gestire la pagina principale, esso carica, prima di mostrare la pagina all'utente, nel metodo `index` tutti i grotti così da mostrarli sia nella mappa che nella lista sottostante.

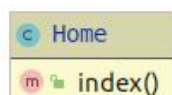


Figure 43 Controller home

#### 4.12.2 Controller login

La classe che gestisce la pagina di login oltre a caricarla contiene un metodo chiamato *checkLogin* che consente di verificare i dati inseriti nel form da un utente. Se saranno stati inseriti tutti i valori richiesti la password verrà trasformata in hash con l'algoritmo sha256 dopodiché verranno caricati dal database tutti gli utenti con l'ausilio della classe *utente\_model* e se ve ne sarà uno con l'email passata che possiede la password inserita allora l'utente verrà connesso. Se l'account possiede privilegi di admin verrà, dopo la procedura di login, reindirizzato alla pagina di amministrazione altrimenti tornerà alla homepage ma avrà la possibilità di inserire un grotto o di votarne uno e di aggiungervi delle immagini.

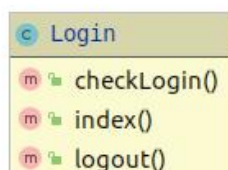


Figure 44 Controller login

#### 4.12.3 Controller register

La classe *Register* consente di gestire e caricare la pagina di registrazione, oltre a ciò essa contiene un metodo che andrà a creare un nuovo utente nel database se la registrazione è andata a buon fine. Esso verifica inizialmente che tutti i dati siano stati inseriti e che non contengano caratteri vietati tramite la classe *input\_manager* dopodiché verifica che le due password inserite siano uguali e che non vi sia un altro utente con la stessa email, se non esiste richiama la classe *utente\_model* che tramite il metodo *addUser* genera il nuovo utente.



Figure 45 Controller register

#### 4.12.4 Controller first login

Il controller della pagina che obbliga l'utente a modificare la password al primo login è piuttosto semplice dato che la pagina non consente di fare molte operazioni. Esso contiene infatti solo la funzione che gestisce le modifiche della password, inizialmente verifica l'inserimento con la classe *input\_manager* e poi, se le due password sono uguali richiama il model della tabella utente e rende effettiva la modifica sul database. Una volta fatto ciò verrà modificato anche il campo *first\_login*, così da non far cambiare all'utente la password tutte le volte, ed infine reindirizzerà l'utilizzatore alla pagina di login.

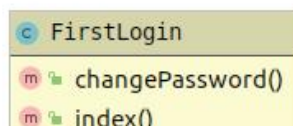


Figure 46 Controller first login

#### 4.12.5 Controller grotto

La classe di gestione della pagina grotti contiene innanzitutto la funzione che mostra il grotto corretto, essa infatti ottiene al click di una riga dalla tabella nella pagina home l'id della località selezionata e ne carica i dati relativi che comprendono anche le immagini e il numero di valutazioni, fatto ciò li passa alla pagina che mostra il grotto scelto. Se sarà stato eseguito con successo il login l'utente potrà caricare un'immagine o votare il ristorante, questo verrà fatto grazie ai metodi *caricaImmagine* e *vota*. Il primo verifica che l'immagine non esista già dopodiché esegue l'hash del titolo della foto e la salva al percorso scelto dopodiché richiama il model delle immagini che aggiungerà quest'ultima al database. Il secondo metodo gestisce le votazioni sulla località verificando che si abbia fatto una scelta valida e poi la inserisce, utilizzando il *grotto\_model*, nella relativa tabella.

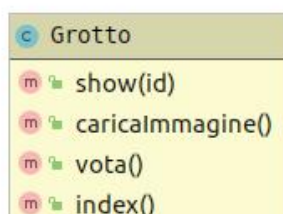


Figure 47 Controller grotto

#### 4.12.6 Controller warning

Il controller warning si occupa solamente di mostrare la pagina di gestione degli errori tramite il metodo *index*.

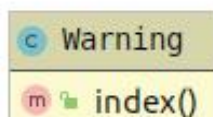


Figure 48 Controller warning

#### 4.12.7 Controller new user

La classe *NewUser* è incaricata di gestire la pagina delle aggiunte di nuovi utenti da parte degli amministratori. Questo viene eseguito tramite la funzione *createUser* che controlla inizialmente se tutti i campi inseriti sono corretti, dopodiché se l'email inserita non è già utilizzata e infine se non vi sono problemi aggiunge l'utente al database utilizzando il model relativo ed invia un email all'indirizzo passato in precedenza con la password da utilizzare al primo login.



Figure 49 Controller new user

#### 4.12.8 Controller add

Il controller *Add* serve a coordinare la pagina di aggiunta di un grotto, esso contiene il metodo *addGrotto* che viene richiamato al click sul bottone di aggiunta nel form. Una volta chiamato il metodo verifica gli inserimenti ed in seguito se la fascia di prezzo esiste, se non vi sono problemi lo aggiunge, richiamando il model, alla tabella *grotto* nel database.

Il metodo *unsetSession* consente invece di cancellare la sessione che dice alla pagina di mostrare il messaggio di successo alla fine di un inserimento così da consentire all'utente di aggiungere un'ulteriore località.

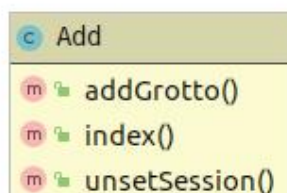


Figure 50 Controller add

#### 4.12.9 Controller admin

Il controller *Admin* consente di gestire la pagina riservata agli amministratori, esso contiene solamente due metodi ed entrambi servono a caricare le pagine che consentono di modificare rispettivamente i grotti o gli utenti all'interno del database. Tutti e due funzionano allo stesso modo verificando prima se il campo selezionato effettivamente esista e se ciò è vero ne caricano tutti i dati che verranno poi utilizzati nella pagina di gestione a cui verrà rimandato l'admin.

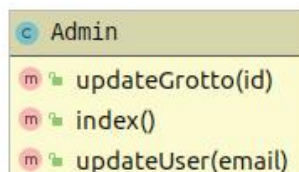


Figure 51 Controller admin

#### 4.12.10 **Controller gestione**

La classe *Gestione* si occupa di coordinare il corretto funzionamento della pagina che dà la possibilità agli amministratori di cambiare i dati di un utente o grotto del database. Essa utilizza i metodi *updateGrotto* e *updateUtente* quando un admin ha terminato la modifica e preme il tasto per salvare. Le due funzioni verificano i nuovi valori e che tutti i requisiti siano rispettati prima di passarli ai rispettivi model per effettuare la modifica sulla banca dati. I requisiti da rispettare sono che ci deve sempre essere almeno un admin e se in un grotto si modifica l'indirizzo devono essere cambiate anche le coordinate geografiche.

Il metodo *elimina* serve a richiamare il metodo con lo stesso nome nel model corrispondente al campo che si vuole eliminare e anche qui vengono verificati gli stessi requisiti della modifica più uno che dice che non si può eliminare il proprio account.

La funzione *back* consente semplicemente di ritornare alla pagina di amministrazione da quella di modifica mentre l'ultimo metodo viene utilizzato per accettare un grotto inserito da un utente senza privilegi.

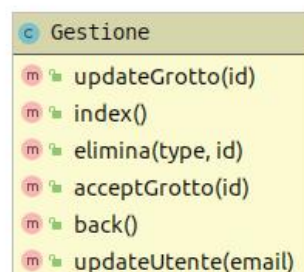


Figure 52 Controller gestione

#### 4.12.11 **Controller reset**

La classe che gestisce la pagina di reset della password ha tre metodi che consentono di gestire tutte le operazioni necessarie per eseguire la modifica. La funzione *sendEmail* fa sì che venga inviata un email con il link per reimpostare la password all'indirizzo passato nel primo form d'inserimento. Essa verifica che esista un account che utilizza l'indirizzo inserito e, se ce n'è uno, inserisce nel database al campo *reset\_token* una stringa di caratteri generata in maniera casuale. Fatto ciò si occupa di inviare un email con il link alla pagina di modifica della password, quest'ultimo è valido per un giorno dopodiché bisognerà ripetere l'operazione per ottenere un nuovo token.

La funzione *resetPassword* viene richiamata al click sul link ricevuto via email, essa si occupa di verificare che esista effettivamente l'utente che si vuole andare a modificare tramite il token e poi richiama la pagina in cui inserire la nuova password.

L'ultimo metodo va a richiamare il model dell'utente per rendere effettivi i cambiamenti se i dati inseriti nel form sono corretti.

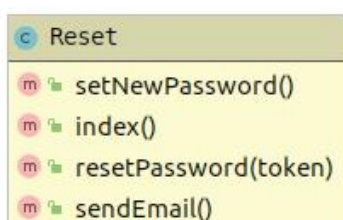


Figure 53 Controller reset

## 5 Test

### 5.1 Protocollo di test

<b>Test Case:</b>	TC-01	<b>Nome:</b>	Test di funzionamento della mappa nella homepage
<b>Riferimento:</b>	REQ-02		
<b>Descrizione:</b>	Verificare che la mappa che mostra i grotti situata nella prima pagina funzioni correttamente		
<b>Prerequisiti:</b>	Dovranno esserci dei grotti nel database (uno minimo) per vedere dei risultati		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Cercare nella mappa il/i puntatore/i corrispondente/i al/i grotto/i</li> <li>3. Premere su un puntatore con il mouse <ol style="list-style-type: none"> <li>a. Premere all'esterno della finestra in un punto qualsiasi della mappa</li> <li>b. Premere su uno dei campi della finestra aperta</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	Al primo click con il mouse dovrebbe aprirsi una finestra che mostra i dati della località (nome, indirizzo, numero di telefono e valutazione) e viene eseguito uno zoom su di essa. Quando si preme all'esterno del riquadro la finestra si chiude e la visuale viene allontanata dal puntatore con un'animazione. Se si premerà su uno dei campi della finestra si verrà reindirizzati alla pagina relativa al grotto.		

<b>Test Case:</b>	TC-02	<b>Nome:</b>	Test di funzionamento della sezione di ricerca nella homepage
<b>Riferimento:</b>	REQ-02		
<b>Descrizione:</b>	Verificare che la tabella che mostra i grotti situata nella prima pagina funzioni correttamente eseguendo anche il filtraggio delle informazioni		
<b>Prerequisiti:</b>	Dovranno esserci dei grotti nel database (uno minimo) per vedere dei risultati		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Scendere fino alla sezione contenente la tabella</li> <li>3. Verificare che i dati siano mostrati correttamente</li> <li>4. Premere sui titoli delle varie colonne</li> <li>5. Premere su una delle righe della tabella che non sia l'intestazione</li> </ol>		
<b>Risultati attesi:</b>	Inizialmente la tabella dovrebbe mostrare tutti i campi del database organizzati secondo l'indentazione quindi: nome, indirizzo, telefono, fascia di prezzo e valutazione. Premendo su uno dei campi dell'intestazione le righe dovrebbero venire ordinate alfabeticamente in base a quella colonna, se si tratta di una delle ultime due colonne le righe verranno ordinate in base al valore in numero della valutazione (1-5) o della fascia di prezzo (caro=3, nella norma=2, buon prezzo=1). Selezionando una riga si dovrebbe visualizzare la pagina relativa al grotto.		

<b>Test Case:</b>	TC-03	<b>Nome:</b>	Test di funzionamento della pagina di login con utente non creato da un admin
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Verificare che la pagina di login funzioni correttamente quando si utilizza un utente creato tramite la pagina apposita e non da un admin		
<b>Prerequisiti:</b>	Dovrà esistere l'utente nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login</li> <li>3. Inserire delle combinazioni di credenziali sbagliate <ol style="list-style-type: none"> <li>a. Utilizzare un email sbagliata con la password corretta</li> <li>b. Utilizzare l'email corretta con una password sbagliata</li> <li>c. Utilizzare sia email che password sbagliate</li> <li>d. Inserire email con formati scorretti <ol style="list-style-type: none"> <li>i. Provare con @test.test</li> <li>ii. Provare con test@test</li> <li>iii. Provare con .test</li> <li>iv. Provare con test</li> <li>v. Provare con test.test</li> <li>vi. Provare con test[*ç@test&amp;%.test`^</li> </ol> </li> <li>e. Inserire dei formati di password scorretti <ol style="list-style-type: none"> <li>i. Password più corte di 8 caratteri</li> <li>ii. Password più lunghe di 50 caratteri</li> <li>iii. Password contenenti caratteri speciali che non siano i seguenti: *%&amp;_?!?+ #/</li> </ol> </li> </ol> </li> <li>4. Premere sul tasto accedi</li> <li>5. Inserire i valori corretti</li> </ol>		
<b>Risultati attesi:</b>	Dovrebbero venire ritornati messaggi di errore per ogni caso di test tranne che per l'ultimo in cui si inseriscono i valori corretti e l'utente si connette.		

<b>Test Case:</b>	TC-04	<b>Nome:</b>	Test di funzionamento della pagina di ripristino della password
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Verificare che la pagina di ripristino della password funzioni correttamente.		
<b>Prerequisiti:</b>	Dovrà esistere l'utente nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login</li> <li>3. Selezionare la voce "Password dimenticata?"</li> <li>4. Inserire un email di un account non esistente</li> <li>5. Premere "invia"</li> <li>6. Non inserire nulla nel campo</li> <li>7. Premere "invia"</li> <li>8. Inserire l'email di un account esistente</li> <li>9. Verificare la casella di posta dell'indirizzo inserito</li> <li>10. Se ricevuto il messaggio premere sul link</li> <li>11. Inserire due password diverse</li> <li>12. Premere "invia"</li> <li>13. Non inserire nulla</li> <li>14. Premere "invia"</li> <li>15. Inserire una password incorretta (meno di 8 caratteri)</li> <li>16. Premere "invia"</li> <li>17. Inserire due password uguali</li> <li>18. Premere "invia"</li> <li>19. Aprire la pagina di login e inserire le nuove credenziali</li> <li>20. Premere "accedi"</li> </ol>		
<b>Risultati attesi:</b>	<p>Quando si premerà il pulsante per procedere ai punti 5, 7, 12 e 14 dovrebbe venire ritornato un errore che avvisa l'utente di cosa ha fatto di sbagliato e si resterà fermi alla pagina in cui ci si trova. Quando si inserirà un'email corretta dovrebbe venire inviato un messaggio all'indirizzo digitato in precedenza contenente il link alla pagina per modificare la password.</p> <p>Una volta inserite due password corrette l'utente dovrebbe venire portato alla pagina di conferma della modifica e potrà eseguire il login con la nuova password.</p>		

<b>Test Case:</b>	TC-05	<b>Nome:</b>	Test di funzionamento della pagina di login con utente creato da un admin
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Verificare che la pagina di login funzioni correttamente quando si utilizza un utente creato tramite la pagina di creazione riservata agli admin		
<b>Prerequisiti:</b>	Dovrà esistere l'utente nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login</li> <li>3. Inserire le credenziali dell'utente generato dall'admin</li> <li>4. Premere "Accedi"</li> <li>5. Inserire due password diverse fra loro</li> <li>6. Premere "invia"</li> <li>7. Non inserire nulla</li> <li>8. Premere "invia"</li> <li>9. Inserire una password incorretta (meno di 8 caratteri)</li> <li>10. Premere "invia"</li> <li>11. Inserire due password uguali e corrette</li> </ol>		
<b>Risultati attesi:</b>	<p>Quando si inseriscono le credenziali dell'utente generato dall'admin per la prima volta si dovrebbe venire reindirizzati ad una pagina che chiede di inserire una nuova password.</p> <p>Se si premerà senza inserire una password, inserendone due diverse fra loro o incorrette verranno mostrati i messaggi di errori relativi. Altrimenti la password verrà cambiata con successo.</p>		

<b>Test Case:</b>	TC-06	<b>Nome:</b>	Test di funzionamento della pagina di registrazione
<b>Riferimento:</b>	REQ-04		
<b>Descrizione:</b>	Verificare che la pagina di registrazione funzioni correttamente.		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login</li> <li>3. Premere il link "Registrati"</li> <li>4. Inserire nei campi dei valori con formati sbagliati <ol style="list-style-type: none"> <li>a. Inserire dei caratteri speciali (escluso il seguente: -) o dei numeri nei campi <i>nome</i> e <i>cognome</i></li> <li>b. Inserire dei caratteri speciali (esclusi i seguenti ._-) nel campo <i>username</i></li> <li>c. Inserire delle email non valide</li> <li>d. Inserire dei numeri di telefono non validi</li> <li>e. Inserire password più corte di 8 caratteri</li> <li>f. Lasciare i campi vuoti</li> </ol> </li> <li>5. Premere "registrati"</li> <li>6. Inserire dei dati corretti</li> <li>7. Premere "registrati"</li> </ol>		



<b>Risultati attesi:</b>	<p>Quando si inseriscono valori sbagliati e poi si preme il tasto per eseguire la registrazione la pagina dovrebbe ritornare degli errori sotto il form e impedire all'utente di procedere. Inserendo valori sbagliati il bordo del campo selezionato dovrebbe diventare rosso.</p> <p>Se verranno inseriti dei valori validi il bordo dei campi dovrebbe diventare verde e premendo il tasto l'utente verrà registrato e riportato alla pagina di login dove potrà accedere.</p>
--------------------------	---

<b>Test Case:</b>	TC-07	<b>Nome:</b>	Test di funzionamento della pagina di aggiunta di un grotto
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Verificare che la pagina di aggiunta di un grotto funzioni correttamente		
<b>Prerequisiti:</b>	Dovrà esistere l'utente nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login</li> <li>3. Accedere con un utente registrato</li> <li>4. Premere nel menu il link "Aggiungi Grotto"</li> <li>5. Inserire nei campi dei valori sbagliati <ol style="list-style-type: none"> <li>a. Inserire dei numeri o dei caratteri speciali nei campi "Nome", "Paese" e "Via"</li> <li>b. Inserire delle lettere nel campo "CAP"</li> <li>c. Inserire dei caratteri speciali nel campo "Numero Civico"</li> <li>d. Inserire dei numeri di telefono non validi</li> <li>e. Lasciare i campi vuoti</li> </ol> </li> <li>6. Premere su "Aggiungi"</li> <li>7. Inserire dei valori validi</li> <li>8. Premere "Verifica"</li> <li>9. Premere su "Aggiungi"</li> </ol>		
<b>Risultati attesi:</b>	<p>Quando si inseriscono valori sbagliati e poi si preme il tasto per eseguire la registrazione la pagina dovrebbe ritornare degli errori sotto il form e impedire all'utente di procedere. Inserendo valori sbagliati il bordo del campo selezionato dovrebbe diventare rosso.</p> <p>Se verranno inseriti dei valori validi il bordo dei campi dovrebbe diventare verde e premendo il tasto all'utente dovrebbe venire mostrato il messaggio di successo dove verrà spiegato che, se non si è admin, bisognerà aspettare che un amministratore accetti l'inserimento prima di vederlo nella lista/mappa.</p> <p>Premendo sul tasto "Verifica", avendo inserito dei valori corretti, si dovrebbe vedere nella mappa sottostante il puntatore di dove si trova il grotto per verificare che venga inserito al posto giusto.</p>		

<b>Test Case:</b>	TC-08	<b>Nome:</b>	Test di funzionamento della pagina di visualizzazione di un grotto
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Verificare che la pagina di visualizzazione di un grotto funzioni correttamente quando vi si accede senza aver effettuato il login		
<b>Prerequisiti:</b>	Dovrà esistere il grotto nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto</li> <li>3. Verificare che i dati siano corretti</li> <li>4. Se vi sono immagini verificare il funzionamento dello slideshow</li> </ol>		
<b>Risultati attesi:</b>	<p>I dati mostrati dovrebbero essere corretti e coerenti con quelli nella tabella o nella mappa.</p> <p>Nella sezione del voto si dovrebbe vedere, se ve ne sono stati, il numero di voti effettuati per quella località.</p> <p>Se vi sono immagini collegate al grotto (più di una) deve esserci uno slideshow che cambia foto automaticamente mentre se ve ne è solo una relativa a quella località si vedrà continuamente solo quella. Se non ve ne sono non si vedrà nulla.</p>		

<b>Test Case:</b>	TC-09	<b>Nome:</b>	Test di funzionamento della pagina di visualizzazione di un grotto
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Verificare che la pagina di visualizzazione di un grotto funzioni correttamente quando vi si accede avendo effettuato il login		
<b>Prerequisiti:</b>	<p>Dovrà esistere il grotto nel database</p> <p>Dovrà esistere l'account nel database</p>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Eseguire il login con un account</li> <li>3. Andare nella homepage</li> <li>4. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto</li> <li>5. Verificare che i dati siano corretti</li> <li>6. Se vi sono immagini verificare il funzionamento dello slideshow</li> <li>7. Andare in fondo alla pagina</li> <li>8. Aggiungere una votazione</li> <li>9. Aggiungere una seconda votazione</li> <li>10. Aggiungere un file <ol style="list-style-type: none"> <li>a. Aggiungere un file che non sia un immagine</li> <li>b. Aggiungere un immagine già collegata ad un altro grotto</li> <li>c. Aggiungere un immagine valida</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	<p>I dati mostrati dovrebbero essere corretti e coerenti con quelli nella tabella o nella mappa.</p> <p>Nella sezione del voto si dovrebbe vedere, se ve ne sono stati, il numero di voti effettuati per quella località.</p> <p>Se vi sono immagini collegate al grotto (più di una) deve esserci uno slideshow</p>		

	<p>che cambia foto automaticamente mentre se ve ne è solo una relativa a quella località si vedrà continuamente solo quella. Se non ve ne sono non si vedrà nulla.</p> <p>Quando si inserirà una votazione per la prima volta la pagina dovrebbe venire aggiornata e dovrebbe venire mostrato la nuova media delle valutazioni e verrà incrementato il conteggio dei voti. Se si prova ad inserire un secondo voto non dovrebbe venire consentito.</p> <p>Quando si inserisce un file dovrebbe funzionare solo se il file è un'immagine e non è in uso da nessun'altro grotto altrimenti verrà mostrato un messaggio d'errore.</p>
--	--

<b>Test Case:</b>	TC-10	<b>Nome:</b>	Test di funzionamento della condivisione sui social
<b>Riferimento:</b>	REQ-06		
<b>Descrizione:</b>	Verificare che la condivisione sui social (Facebook) di un grotto funzioni correttamente		
<b>Prerequisiti:</b>	Dovrà esistere il grotto nel database		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto</li> <li>3. Premere il tasto "Condividi su Facebook"</li> </ol>		
<b>Risultati attesi:</b>	Quando si premerà sul tasto di condivisione dovrebbe venire aperta una pagina di facebook in cui si potrà condividere con il proprio account il link alla pagina.		

<b>Test Case:</b>	TC-11	<b>Nome:</b>	Test di funzionamento della pagina riservata agli admin
<b>Riferimento:</b>	REQ-01		
<b>Descrizione:</b>	Verificare che la pagina riservata agli admin funzioni correttamente		
<b>Prerequisiti:</b>	Dovrà esistere l'utente nel database ed avere i privilegi di amministratore		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Aprire il sito alla pagina principale</li> <li>2. Accedere tramite la pagina di login con un account admin</li> <li>3. Premere nel menu il link "Amministrazione"</li> <li>4. Verificare che nelle tabelle "utenti", "grotti", "inserimenti" e "immagini" si vedano i dati giusti ordinati correttamente</li> <li>5. Premere il tasto "modifica" di una riga delle tabelle "utenti" e "grotti" <ol style="list-style-type: none"> <li>a. Verificare il funzionamento dei form ripetendo le procedure spiegate nei TC-03 e TC-07</li> <li>b. Verificare il funzionamento del tasto "verifica" e del tasto "indietro"</li> </ol> </li> <li>6. Premere il tasto "accetta" di una riga nella tabella "inserimenti"</li> <li>7. Verificare che i tasti di eliminazione in tutte le tabelle funzionino <ol style="list-style-type: none"> <li>a. Premere sul tasto elimina nella riga del proprio utente</li> </ol> </li> <li>8. Premere su una delle righe nella tabella "immagini"</li> </ol>		
<b>Risultati attesi:</b>	<p>Accedendo alla pagina riservata agli amministratori si dovrebbero visualizzare tutte le tabelle contenenti gli utenti, i grotti, gli inserimenti e le immagini ordinate correttamente.</p> <p>Nella tabella degli utenti e dei grotti la penultima colonna serve per modificare il campo e premendo sull'icona in essa si dovrebbe aprire un form contenente i dati del campo selezionato dove si potranno eseguire le modifiche.</p> <p>Nella tabella relativa agli inserimenti il penultimo campo consente ad un admin di</p>		

accettare il grotto inserito da un utente di base così da farlo apparire nella tabella della homepage, nella mappa e nella tabella nella pagina admin.

In tutte le tabelle l'ultimo campo serve a eliminare il campo, premendolo dovrebbe venire chiesta conferma e poi dovrebbe venire eliminato il campo. Se si tratta di un campo utente non si dovrebbe poter eliminare il proprio account e nemmeno, se ve ne è uno solo, l'admin.

## 5.2 Risultati test

PROTOCOLLO	ESITO	NOTE
TC-01	PASSATO	-
TC-02	PASSATO	-
TC-03	PASSATO	-
TC-04	PASSATO	-
TC-05	PASSATO	-
TC-06	PASSATO	-
TC-07	PASSATO	-
TC-08	PASSATO	-
TC-09	PASSATO	-
TC-10	FALLITO	Non è stato implementato
TC-11	PASSATO	-

## 5.3 Mancanze/limitazioni conosciute

Il progetto presenta alcune limitazioni e sono relative alla precisione nell'inserimento dei puntatori nella mappa. Esse sono dovute alle API di Google MAPS che non sono perfette e con certi indirizzi, soprattutto quelli situati in strade o edifici recenti, sbagliano di alcuni metri nel posizionamento del marker sulla mappa. Google è comunque la compagnia che fornisce la precisione maggiore in questo campo.

Una mancanza nota all'interno del progetto è la possibilità di condividere un grotto su facebook o su un altro social network. La funzionalità non è presente perché facebook, che è l'unico che consente di condividere delle pagine web e che fornisce delle API ben documentate, richiede che il sito sia su un hosting. È stato provato a caricare il prodotto sull'hosting della scuola *samtinfo.ch* ma esso non supporta i trigger in mysql e inoltre non avevo i permessi per accedere a phpmyadmin. È stato quindi deciso di non inserire la funzionalità per mancanza di un server sulla quale caricare il sito e perché non ritenuta fondamentale, potrà essere una possibile evoluzione del prodotto.

## 6 Consuntivo

Le principali differenze che si possono notare nel diagramma di Gantt consuntivo è il fatto che sono state aggiunte due attività nella sezione dell'implementazione. La prima rappresenta la creazione di una pagina che consenta di visualizzare un grotto che non era stata calcolata in precedenza così come la seconda che aggiunge la parte di implementazione della pagina di reset della password utilizzando la classe *phpmailer*.

Come si può notare dall'immagine che segue i tempi erano stati calcolati leggermente male dando troppo tempo all'analisi che poi in realtà me ne ha impiegato meno. Il tempo risparmiato nella prima parte è stato poi impiegato nella scrittura del codice e nel testing.

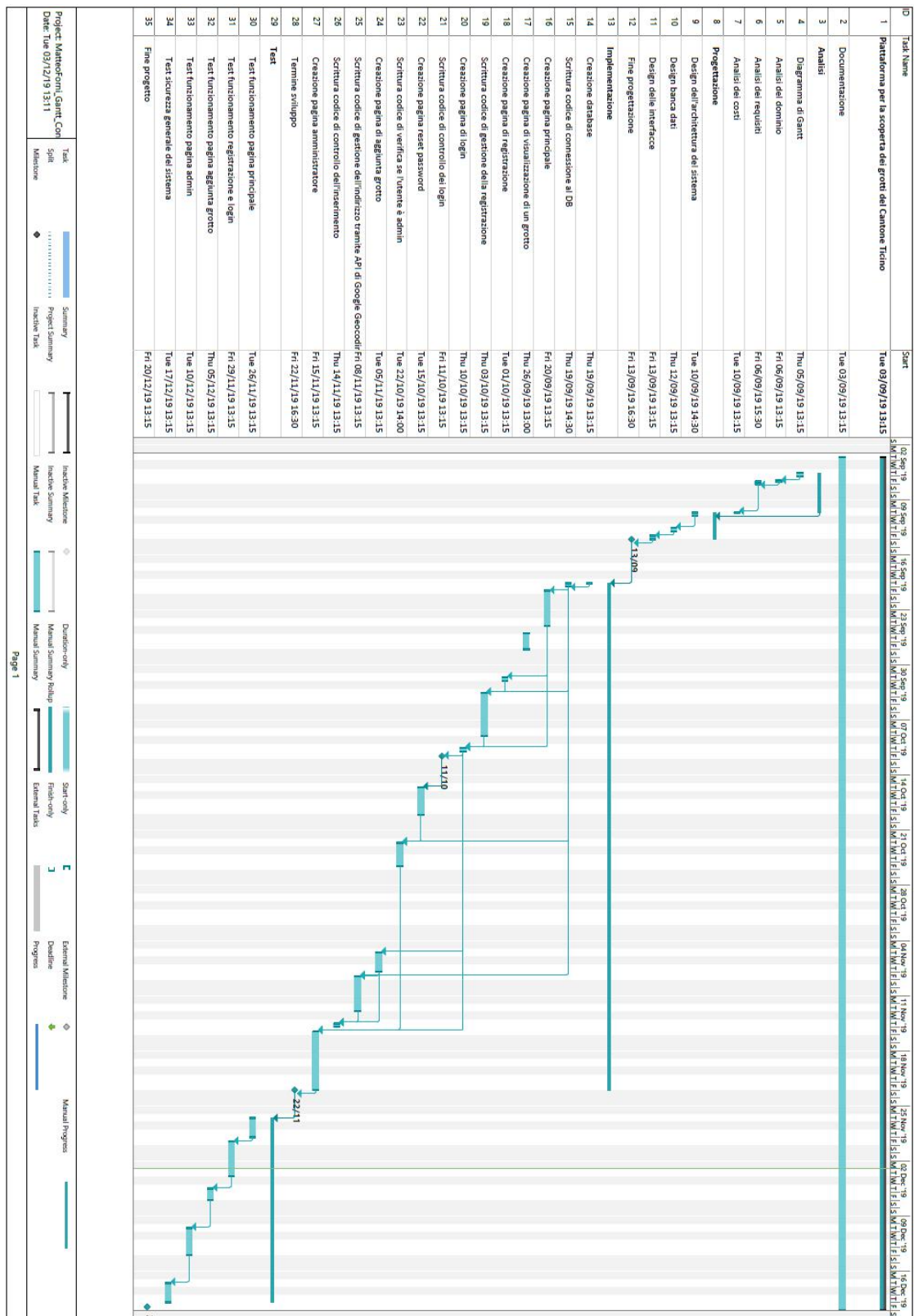


Figure 54 Diagramma di Gantt consuntivo

## **7 Conclusioni**

---

La soluzione implementata potrà tornare di utilità a chiunque se volesse visualizzare i grotti del Ticino ma, senza un gruppo di utenti abbastanza presente e attivo che aggiunge e vota le località il prodotto perde un po' il suo senso. I risultati ottenuti sono però soddisfacenti e il progetto potrà essere portato avanti rendendolo più interessante e utilizzato.

### **7.1 Sviluppi futuri**

Al prodotto possono venire apportate svariate modifiche per aggiornarlo o per aggiungere funzionalità, penso che le più consistenti e utili sarebbero le seguenti due. Come primo sviluppo si potrebbe aggiungere la condivisione su Facebook o, se esce un social network nuovo e utilizzato, su un'altra piattaforma.

Come secondo sviluppo sarebbe interessante implementare la possibilità di aggiungere dei commenti ai propri voti così da motivare la scelta e, se è il caso, avere una discussione con il gestore del grotto. Questo implicherebbe anche aggiungere dei ruoli come ad esempio *proprietario*.

### **7.2 Considerazioni personali**

Il progetto mi ha consentito di solidificare parecchie conoscenze già apprese in passato ma utilizzate di rado così come di migliorarmi in un discorso più globale dato che ho dovuto integrare parecchie nozioni di ambiti e materie diverse.

Questo è stato il primo progetto di preparazione all'esame finale e sono soddisfatto dei risultati ottenuti che mi hanno, anche, consentito di migliorare le mie abilità di gestione del tempo e delle attività da svolgere. Grazie a ciò ho preso conoscenza dei miei punti deboli su cui dovrò lavorare di più in futuro.

## 8 Bibliografia

### 8.1 Sitografia

[https://www.gr.ch/DE/institutionen/verwaltung/dvs/alg/dokumentation/Vermessung/Dokumentliste%20AV/Brosch\\_LV95\\_it.pdf](https://www.gr.ch/DE/institutionen/verwaltung/dvs/alg/dokumentation/Vermessung/Dokumentliste%20AV/Brosch_LV95_it.pdf), Nuove coordinate per la Svizzera, 20-09-2019

<https://developers-dot-devsite-v2-prod.appspot.com/maps/documentation/javascript/tutorial>, Google Maps API Documentation, 20-09-2019

<https://developers-dot-devsite-v2-prod.appspot.com/maps/documentation/javascript/examples/geocoding-reverse>, Reverse Geocoding, 20.09.2019

<https://mdbootstrap.com/docs/jquery/>, Material Design Bootstrap Documentation, 20.09.2019

<https://mdbootstrap.com/docs/jquery/tables/datatables/>, Material Design Bootstrap Datatables, 20.09.2019

<http://www.mysqltutorial.org/mysql-on-delete-cascade/>, MySQL on delete cascade, 18.10.2019

<https://code.tutsplus.com/tutorials/how-to-upload-a-file-in-php-with-example--cms-31763>, PHP upload a file, 25.10.2019

[https://www.w3schools.com/php/php\\_cookies.asp](https://www.w3schools.com/php/php_cookies.asp), PHP cookies, 08.11.2019

<https://www.php.net/manual/en/features.cookies.php>, PHP cookies, 08.11.2019

<https://github.com/marlospomin/smoothie>, Smooth Scroll Javascript Plugin, 12.11.2019

<https://www.jetbrains.com/help/phpstorm/quick-start-guide-phpstorm.html>, PHPStorm documentation, 14.11.2019

<https://developers.facebook.com/docs/instagram-basic-display-api/>, Instagram share API, 15.11.2019

<https://fontawesome.com/icons?d=gallery>, Fontawesome icons gallery, 15.11.2019

<https://mdbootstrap.com/docs/jquery/components/alerts/>, MDBBootstrap alerts, 22.11.2019

<https://bugs.mysql.com/bug.php?id=35177>, Trigger OLD and NEW values, 28.11.2019

### 8.2 Indice delle immagini

Figure 1 Use Case.....	7
Figure 2 Diagramma di Gantt.....	8
Figure 3 Gantt capitolo Analisi.....	9
Figure 4 Gantt capitolo Progettazione.....	9
Figure 5 Gantt capitolo implementazione prima parte.....	10
Figure 6 Gantt capitolo implementazione seconda parte.....	10
Figure 7 Gantt capitolo Test.....	11

Figure 8	Design del sistema.....	12
Figure 9	Diagramma ER del database.....	13
Figure 10	Schema logico del database.....	13
Figure 11	Mockup pagina Index.....	15
Figure 12	Mockup pagina di login.....	15
Figure 13	Mockup pagina registrazione.....	16
Figure 14	Mockup pagina creazione grotti.....	17
Figure 15	Mockup pagina admin.....	18
Figure 16	Codice di creazione del trigger MySQL.....	20
Figure 17	Homepage, mappa con puntatori	
Figure 18	Homepage, mappa con grotto selezionato.....	21
Figure 19	Homepage, tabella dei grotti.....	21
Figure 20	Pagina grotto, informazioni	
Figure 21	Pagina grotto, valutazione e aggiunta immagini.....	22
Figure 22	Pagina di login, form di connessione.....	22
Figure 23	Pagina di registrazione, form d'inserimento.....	23
Figure 24	Pagina di reset, invio email.....	23
Figure 25	Pagina di reset, cambio credenziali.....	23
Figure 26	Pagina di aggiunta grotti, form d'inserimento.....	24
Figure 27	Pagina admin, lista delle sezioni.....	24
Figure 28	Pagina admin, sezione utenti	
Figure 29	Pagina admin, aggiunta utente.....	25
Figure 30	Pagina admin, sezione grotti.....	25
Figure 31	Pagina admin, sezione inserimenti.....	26
Figure 32	Pagina admin, sezione immagini.....	26
Figure 33	Pagina d'errore, errore connessione al database.....	26
Figure 34	Diagramma UML delle classi nella cartella models.....	27
Figure 35	Codice di connessione al database.....	28
Figure 36	Classe utente model.....	28
Figure 37	Classe grotto model.....	29
Figure 38	Classe foto model.....	30
Figure 39	Classe voto model.....	31
Figure 40	Classe fascia di prezzo model.....	31
Figure 41	Classe ruolo model.....	32
Figure 42	Diagramma UML dei controllers.....	32
Figure 43	Controller home.....	33
Figure 44	Controller login.....	33
Figure 45	Controller register.....	33
Figure 46	Controller first login.....	34
Figure 47	Controller grotto.....	34
Figure 48	Controller warning.....	34
Figure 49	Controller new user.....	35
Figure 50	Controller add.....	35
Figure 51	Controller admin.....	35
Figure 52	Controller gestione.....	36
Figure 53	Controller reset.....	36
Figure 54	Diagramma di Gantt consuntivo.....	45

## 9 Allegati

Elenco degli allegati:

- Abstract
- Diari di lavoro
- Quaderno dei compiti
- Prodotto