

Candidato: Matteo Forni

Azienda: Scuola di Arti e Mestieri Trevano

Periodo: 03.09.2019 - 20.12.2019

Presentazione: Gennaio 2020

Situazione iniziale

La richiesta di partenza di questo progetto era quella di strutturare un sito che consentisse di visualizzare i grotti nel canton Ticino. Le località dovevano essere navigabili tramite una mappa interattiva o una lista a cui è possibile applicare dei filtri.

Doveva essere inclusa una parte di votazione e di aggiunta di un nuovo ristorante. Inoltre doveva essere presente una sezione di gestione dei grotti e degli account utente.

Attuazione

Il prodotto è stato implementato utilizzando PHP con la struttura di gestione di file MVC. Per avere una precisione elevata nella mappa interattiva sono state utilizzate le API di Google Maps che offrono una buona documentazione e una grande accuratezza.

Tutta la parte visualizzabile dall'utente finale è stata creata utilizzando il framework Material Design Bootstrap che comprende già delle funzionalità molto comode che sono state inserite nel prodotto finale come la tabelle ordinabili automaticamente tramite il nome della colonna. Uno degli obiettivi del sito era quello di essere intuitivo e responsive e grazie a tale framework è stato più facile.

Risultati

I risultati ottenuti sono soddisfacenti in quanto tutti i requisiti sono stati soddisfatti e il sito funziona senza interruzioni ed errori. Non sempre è stato evidente o veloce superare i problemi e risolvere gli errori ma sono convinto di avere creato un buon prodotto che possiede tutte le funzionalità richieste.



Piattaforma per la scoperta dei grotti in Ticino

Piattaforma per la scoperta dei grotti del Ticino

1	Introduzione.....	4
1.1	Informazioni sul progetto.....	4
1.2	Abstract.....	4
1.3	Scopo.....	4
2	Analisi.....	5
2.1	Analisi del dominio.....	5
2.2	Analisi e specifica dei requisiti.....	5
2.3	Use case.....	7
2.4	Pianificazione.....	8
2.4.1	Analisi.....	9
2.4.2	Progettazione.....	9
2.4.3	Implementazione.....	10
2.4.4	Test.....	11
2.5	Analisi dei mezzi.....	11
2.5.1	Software.....	11
2.5.2	Hardware.....	11
3	Progettazione.....	12
3.1	Design dell'architettura del sistema.....	12
3.2	Design dei dati e database.....	13
3.3	Design delle interfacce.....	15
3.3.1	Pagina iniziale.....	15
3.3.2	Pagina di login.....	15
3.3.3	Pagina di registrazione.....	16
3.3.4	Pagina di creazione dei grotti.....	17
3.3.5	Pagina di amministrazione.....	18
4	Implementazione.....	19
4.1	Struttura del progetto.....	19
4.2	Database.....	19
4.2.1	Tabella utente.....	19
4.2.2	Tabella grotto.....	20
4.2.3	Tabella foto.....	20
4.2.4	Tabella voto.....	20
4.3	Pagina Home.....	21
4.4	Pagina Grotto.....	22
4.5	Pagina di login.....	22
4.6	Pagina di registrazione.....	23
4.7	Pagina di ripristino della password.....	23
4.8	Pagina di aggiunta grotto.....	24
4.9	Pagina di amministrazione.....	24
4.9.1	Sezione utenti.....	25
4.9.2	Sezione grotti.....	25
4.9.3	Sezione inserimenti.....	26
4.9.4	Sezione immagini.....	26
4.10	Pagina di warning.....	26
4.11	Models.....	27
4.11.1	Classe DBConnection.....	28
4.11.2	Classe input_manager.....	28
4.11.3	Classe mail_manager.....	28
4.11.4	Classe utente_model.....	28
4.11.5	Classe grotto_model.....	29
4.11.6	Classe foto_model.....	30
4.11.7	Classe voto_model.....	31
4.11.8	Classe fascia_prezzo_model.....	31
4.11.9	Classe ruolo_model.....	32
4.12	Controllers.....	32
4.12.1	Controller home.....	33
4.12.2	Controller login.....	33

4.12.3 Controller register.....	33
4.12.4 Controller first login.....	34
4.12.5 Controller grotto.....	34
4.12.6 Controller warning.....	34
4.12.7 Controller new user.....	35
4.12.8 Controller add.....	35
4.12.9 Controller admin.....	35
4.12.10 Controller gestione.....	36
4.12.11 Controller reset.....	36
5 Test.....	37
5.1 Protocollo di test.....	37
5.2 Risultati test.....	44
5.3 Mancanze/limitazioni conosciute.....	44
6 Consuntivo.....	44
7 Conclusioni.....	46
7.1 Sviluppi futuri.....	46
7.2 Considerazioni personali.....	46
8 Bibliografia.....	47
8.1 Sitografia.....	47
8.2 Indice delle immagini.....	47
9 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	
Nome	Creazione pagina riservata agli amministratori
Priorità	1
Versione	1.0
Note	Si necessitano i permessi di amministratore per accedervi
Sotto requisiti	
001	Si dovrà avere la possibilità di creare/modificare/eliminare gli utenti
002	Si dovrà avere la possibilità di creare/modificare/eliminare dei grotti
003	Si dovrà avere la possibilità di rendere un utente normale amministratore
004	Si dovrà avere la possibilità di approvare o rifiutare la creazione di un grotto da parte di un utente

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

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

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

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

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

2.3 Use case

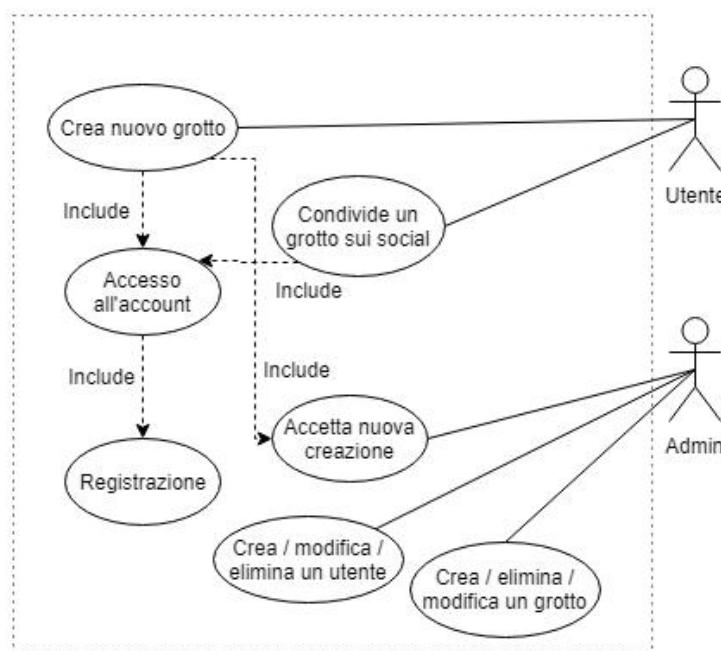


Figure 1 Use Case

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

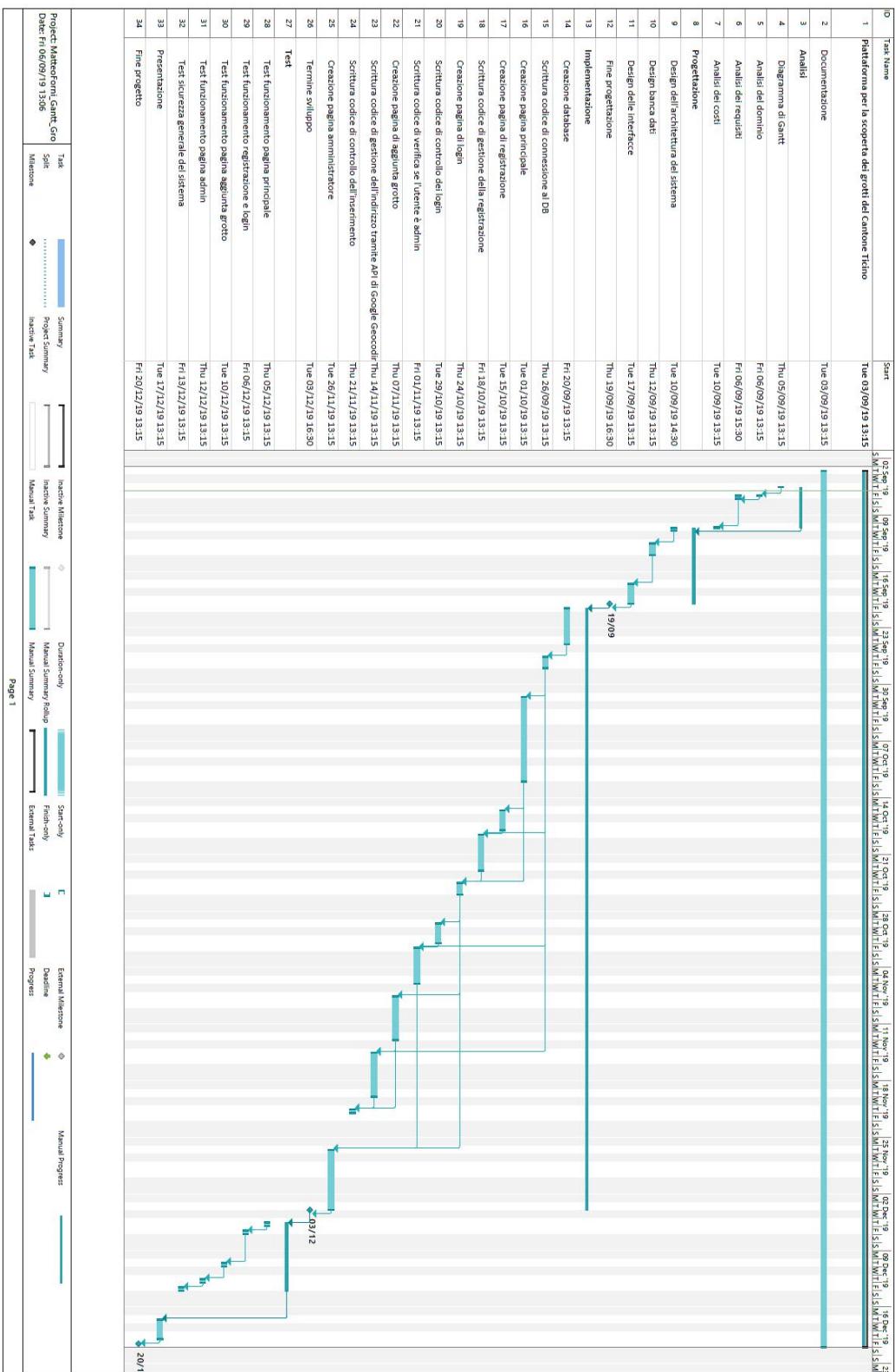


Figure 2 Diagramma di Gantt

Piattaforma per la scoperta dei grotti del Ticino

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

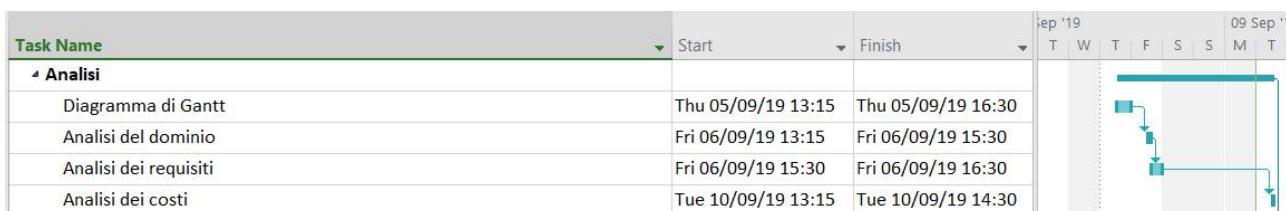


Figure 3 Gantt capitolo Analisi

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 iniziare 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

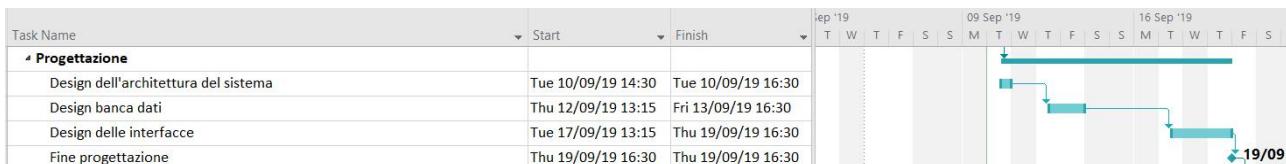


Figure 4 Gantt capitolo Progettazione

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.

Piattaforma per la scoperta dei grotti del Ticino

2.4.3 Implementazione

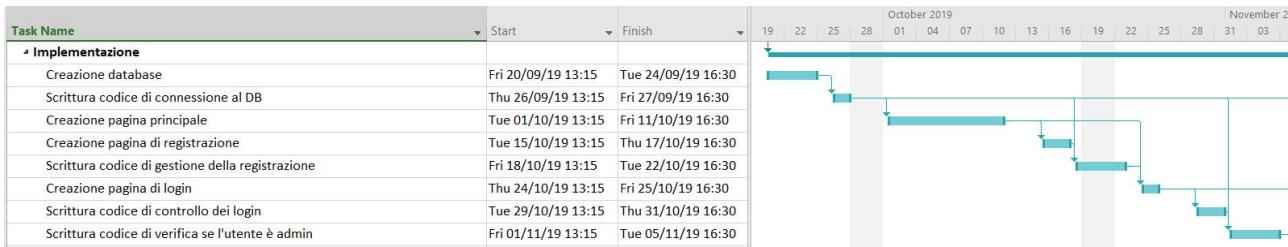


Figure 5 Gantt capitolo implementazione prima parte

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.

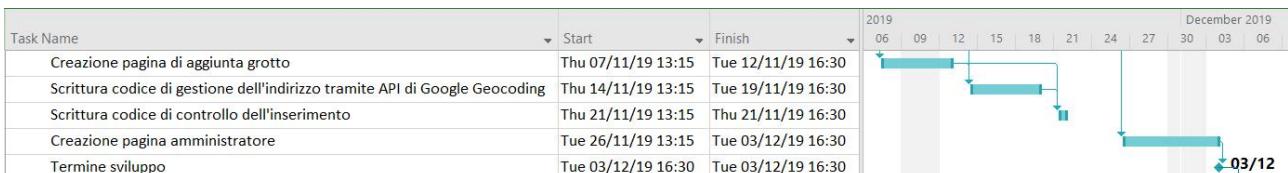


Figure 6 Gantt capitolo implementazione seconda parte

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.

Piattaforma per la scoperta dei grotti del Ticino

2.4.4 Test

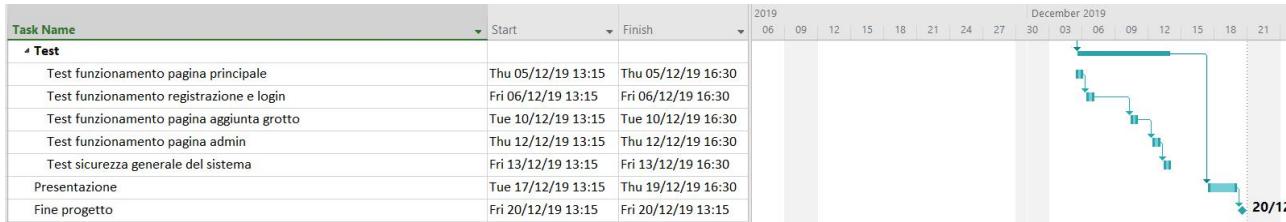


Figure 7 Gantt capitolo Test

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

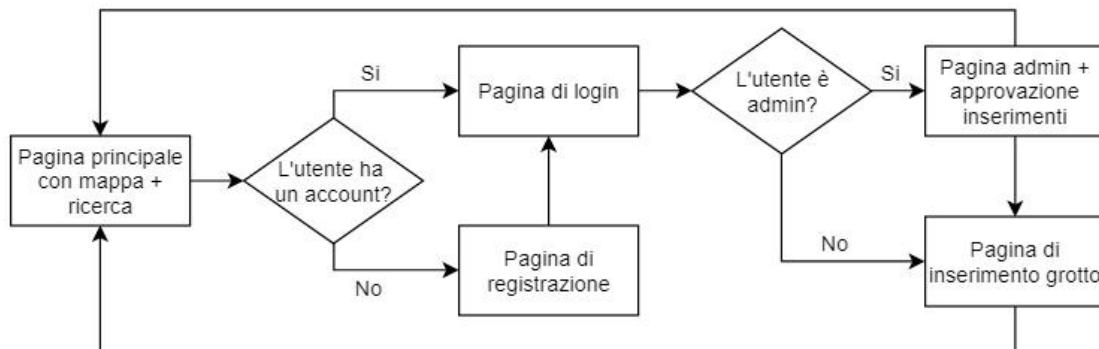


Figure 8 Design del sistema

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.

Piattaforma per la scoperta dei grotti del Ticino

3.2 Design dei dati e database

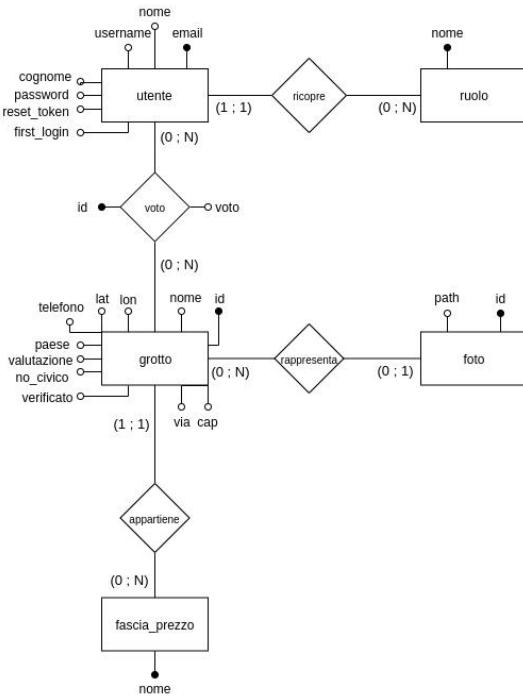


Figure 9 Diagramma ER del database

Il diagramma ER del database è piuttosto semplice come si può vedere dall'immagine sopra. 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)
GROTTA(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

Piattaforma per la scoperta dei grotti del Ticino

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

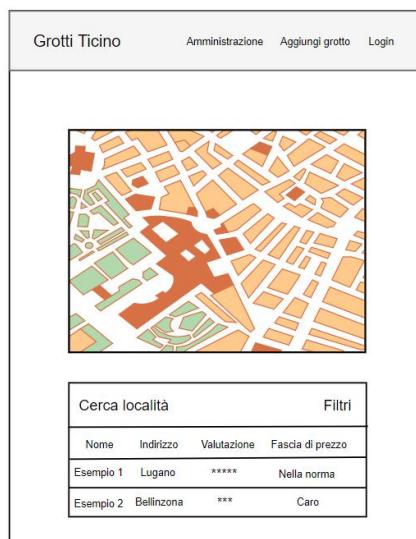


Figure 11 Mockup pagina Index

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 pulsanti. 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 mostra un layout semplice. In alto a sinistra c'è il logo "Grotti Ticino". A destra, ci sono i link "Login" e "Home". Sotto il logo, il titolo "Registrazione" è scritto in un font leggermente più grande. La parte centrale della pagina contiene sei campi input, ciascuno con una etichetta a sinistra: "Nome", "Cognome", "Username", "Email", "Password" e "Ripeti Password". I campi sono disposti verticalmente. In basso a destra, c'è un pulsante contenente 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

Grotti Ticino	
	Amministrazione
	Home
Nome	<input type="text"/>
Fascia_Prezzo	<input type="text"/>
Valutazione	<input type="text"/>
CAP	<input type="text"/>
Paese	<input type="text"/>
Via	<input type="text"/>
No. Civico	<input type="text"/>
<input type="button" value="Verifica"/> <input type="button" value="Salva"/>	

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.

Piattaforma per la scoperta dei grotti del Ticino
3.3.5 Pagina di amministrazione

Grotti Ticino	Aggiunta Grotto	Home																								
Crea, modifica, elimina utenti		<table border="1"> <thead> <tr> <th>email</th><th>username</th><th>nome</th><th>cognome</th><th>modifica</th><th>elimina</th></tr> </thead> <tbody> <tr> <td>a@a.a</td><td>a_b</td><td>a</td><td>b</td><td>modifica</td><td>elimina</td></tr> <tr> <td>b@b.b</td><td>c_d</td><td>c</td><td>d</td><td>modifica</td><td>elimina</td></tr> <tr> <td>c@c.c</td><td>e_f</td><td>e</td><td>f</td><td>modifica</td><td>elimina</td></tr> </tbody> </table>	email	username	nome	cognome	modifica	elimina	a@a.a	a_b	a	b	modifica	elimina	b@b.b	c_d	c	d	modifica	elimina	c@c.c	e_f	e	f	modifica	elimina
email	username	nome	cognome	modifica	elimina																					
a@a.a	a_b	a	b	modifica	elimina																					
b@b.b	c_d	c	d	modifica	elimina																					
c@c.c	e_f	e	f	modifica	elimina																					
Crea, modifica, elimina grotti		<table border="1"> <thead> <tr> <th>nome</th><th>indirizzo</th><th>lat</th><th>lon</th><th>modifica</th><th>elimina</th></tr> </thead> <tbody> <tr> <td>a</td><td>6900 a, a 10</td><td>1</td><td>2</td><td>modifica</td><td>elimina</td></tr> <tr> <td>b</td><td>6900 b, b 10</td><td>3</td><td>4</td><td>modifica</td><td>elimina</td></tr> <tr> <td>c</td><td>6900 c, c 10</td><td>5</td><td>6</td><td>modifica</td><td>elimina</td></tr> </tbody> </table>	nome	indirizzo	lat	lon	modifica	elimina	a	6900 a, a 10	1	2	modifica	elimina	b	6900 b, b 10	3	4	modifica	elimina	c	6900 c, c 10	5	6	modifica	elimina
nome	indirizzo	lat	lon	modifica	elimina																					
a	6900 a, a 10	1	2	modifica	elimina																					
b	6900 b, b 10	3	4	modifica	elimina																					
c	6900 c, c 10	5	6	modifica	elimina																					
Verifica nuovi inserimenti		<table border="1"> <thead> <tr> <th>user</th><th>grotto</th><th>voto</th><th>prezzo</th><th>accetta</th><th>rifiuta</th></tr> </thead> <tbody> <tr> <td>a</td><td>6900 a, a 10</td><td>1</td><td>2</td><td>accetta</td><td>rifiuta</td></tr> <tr> <td>b</td><td>6900 b, b 10</td><td>3</td><td>4</td><td>accetta</td><td>rifiuta</td></tr> <tr> <td>c</td><td>6900 c, c 10</td><td>5</td><td>6</td><td>accetta</td><td>rifiuta</td></tr> </tbody> </table>	user	grotto	voto	prezzo	accetta	rifiuta	a	6900 a, a 10	1	2	accetta	rifiuta	b	6900 b, b 10	3	4	accetta	rifiuta	c	6900 c, c 10	5	6	accetta	rifiuta
user	grotto	voto	prezzo	accetta	rifiuta																					
a	6900 a, a 10	1	2	accetta	rifiuta																					
b	6900 b, b 10	3	4	accetta	rifiuta																					
c	6900 c, c 10	5	6	accetta	rifiuta																					

Figure 15 Mockup pagina admin

La pagina degli admin sarà di base molto vuota e semplice, conterrà tre buttoni che consentiranno di modificare gli utenti, i grotti o di verificare gli inserimenti. Una volta premuto su uno di questi buttoni 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 doppiioni 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

Piattaforma per la scoperta dei grotti del Ticino

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.

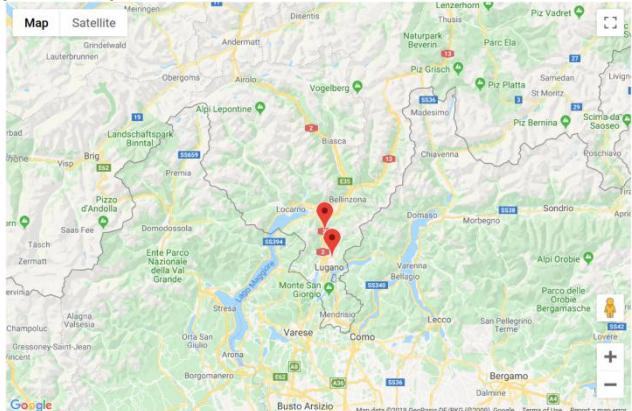


Figure 17 Homepage, mappa con puntatori



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	☆☆☆☆☆

Previous 1 2 Next

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

The screenshot shows two side-by-side sections. The left section, titled 'Vota', contains the text 'Puoi votare un grotto solo una volta' (You can vote for a grotto only once) above a five-star rating scale, which is currently empty. Below the scale is a blue 'VOTA' button. The right section, titled 'Aggiungi immagini', contains the text 'Aggiungi immagini al grotto' (Add images to the grotto) above a file upload interface with 'Upload', 'Scegli il file' (Select file), and 'Browse' buttons. Below this is a blue 'AGGIUNGI' (ADD) button.

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.

The screenshot shows a login form titled 'Accedi'. It has two input fields: 'E-mail' and 'Password'. Below the password field is a link 'Password dimenticata?' (Forgot password?). A large blue 'ACCEDI' (LOG IN) button is at the bottom. At the very bottom, there is a link 'Non hai un account? Registrati' (Don't have an account? Register).

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 esisti 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.

The screenshot shows a registration form titled "Registrazione". It contains the following fields:

- Nome (Name) and Cognome (Surname) in separate input fields.
- Username: A single input field with a note below it: "Minimo 3 caratteri" (Minimum 3 characters).
- E-mail: A single input field with a note below it: "Minimo 8 caratteri" (Minimum 8 characters).
- Password and Ripeti la password (Repeat password): Two input fields for password entry.
- A large blue "REGISTRATI" (Register) button at the bottom.

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 quale 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.

The screenshot shows a password reset form titled "Reimposta la tua password". It contains a single input field for "Email" containing "esempio@esempio.es" and a large blue "INVIA" (Send) button.

Figure 24 Pagina di reset, invio email

The screenshot shows a password reset form titled "Reimposta la tua password". It contains three input fields: "Email" with "matteo20012001@gmail.com", "Password", and "Ripeti la Password". Below the fields is a large blue "INVIA" (Send) button.

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.

Aggiungi un grotto

Nome

CAP

Paese

Via

Numero Civico

Telefono

Scegli una fascia di prezzo

AGGIUNGI

VERIFICA

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

Piattaforma per la scoperta dei grotti del Ticino

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.

Gestione utenti						
Email	Nome	Cognome	Username	Ruolo	Modifica	Elimina
matteo@gmail.com	Matteo	Forni	fornimatteo	utente	<button>MODIFICA</button>	<button>ELIMINA</button>
matteo@samtrevano.ch	Matteo	Forni	matteofofni	admin	<button>MODIFICA</button>	<button>ELIMINA</button>

Figure 28 Pagina admin, sezione utenti

Aggiungi un utente

<input placeholder="Nome" type="text"/>	<input placeholder="Cognome" type="text"/>
<input placeholder="Username" type="text"/>	Minimo 3 caratteri
<input placeholder="E-mail" type="text"/>	
CREA UTENTE	

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 grotta.

Gestione Grotti						
Nome	Indirizzo	Telefono	Fascia di Prezzo	Valutazione	Modifica	Elimina
Primo esempio	6952 Canobbio, Via Trevano 25	+41 00 000 000	Caro	☆☆☆☆☆	<button>MODIFICA</button>	<button>ELIMINA</button>
Secondo esempio	6952 Canobbio, Via Trevano 25	+41 00 000 000	Nella norma	☆☆☆☆☆	<button>MODIFICA</button>	<button>ELIMINA</button>

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	ACCETTA	ELIMINA	
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/1c1b3f6afdf5a9e2a317ff84d8c22cd93.png	Primo esempio	ELIMINA	
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

Piattaforma per la scoperta dei grotti del Ticino

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.

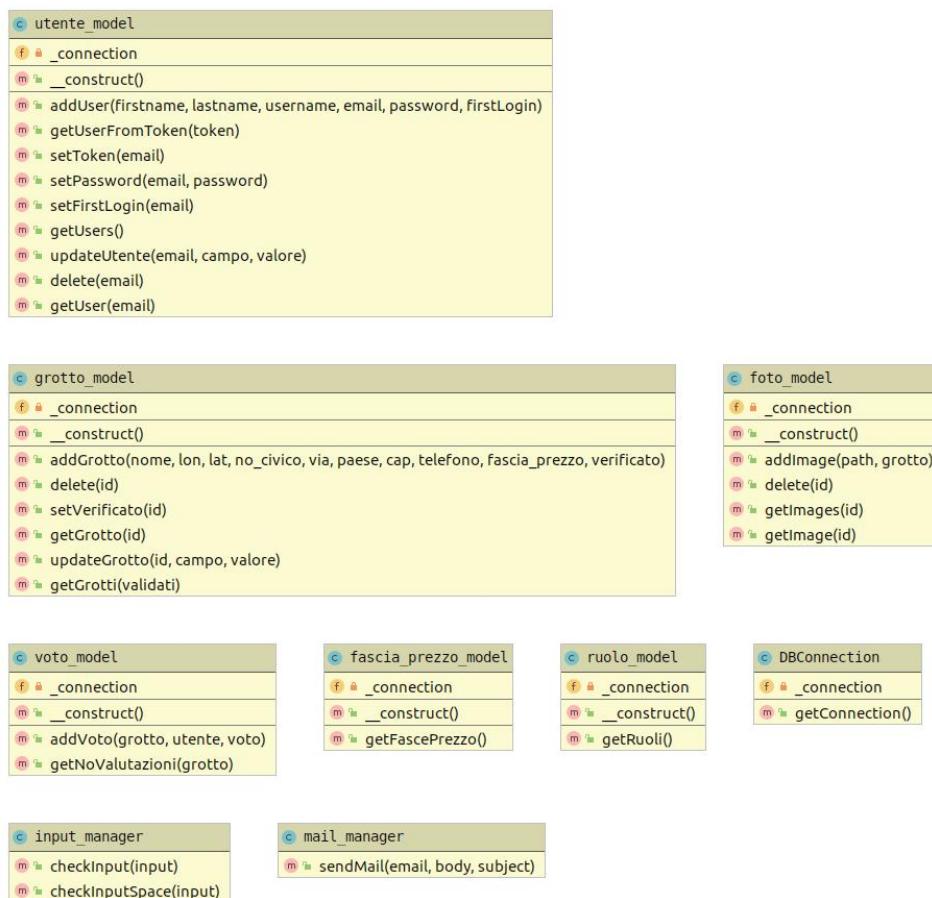


Figure 34 Diagramma UML delle classi nella cartella models

Piattaforma per la scoperta dei grotti del Ticino

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

Piattaforma per la scoperta dei grotti del Ticino

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.

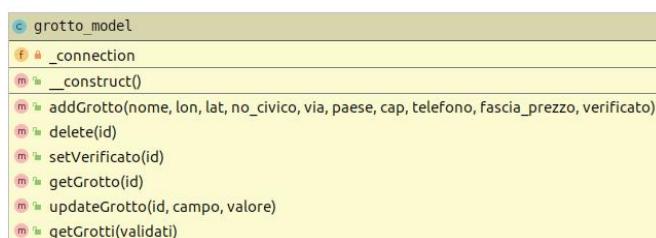


Figure 37 Classe grotto model

4.11.5.1 Metodi di lettura

I metodi che consentono di leggere 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 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.

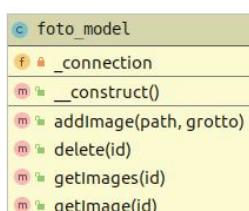


Figure 38 Classe foto model

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 primo 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.

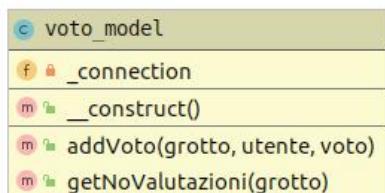


Figure 39 Classe voto model

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.

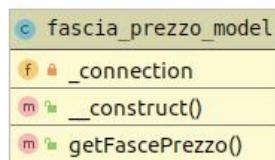


Figure 40 Classe fascia di prezzo model

Piattaforma per la scoperta dei grotti del Ticino

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.

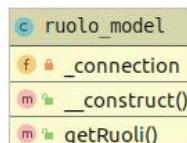


Figure 41 Classe ruolo model

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.

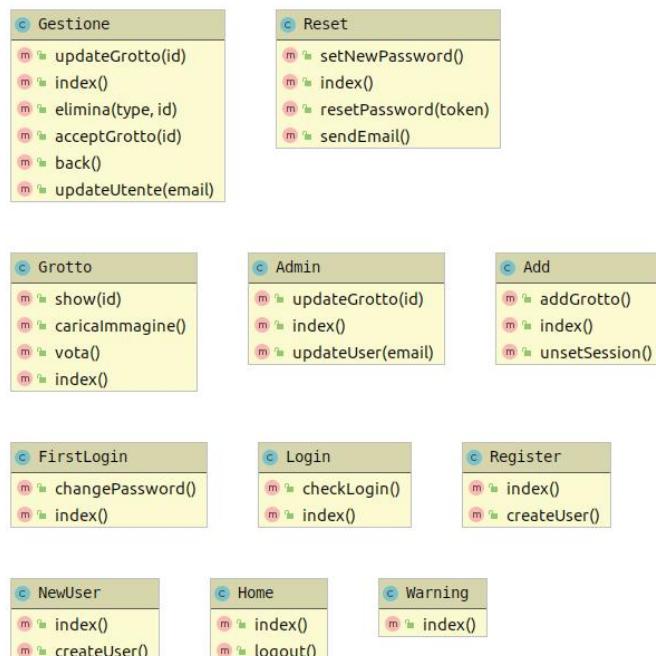


Figure 42 Diagramma UML dei controllers

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.

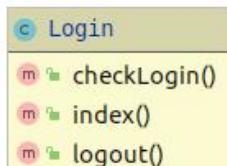


Figure 44 Controller login

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

Piattaforma per la scoperta dei grotti del Ticino

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.

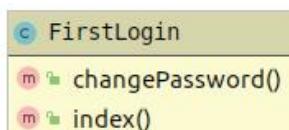


Figure 46 Controller first login

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 *caricalimmagine* 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*.

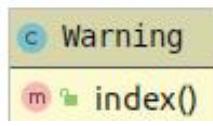


Figure 48 Controller warning

Piattaforma per la scoperta dei grotti del Ticino

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.

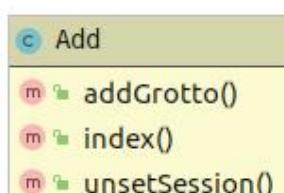


Figure 50 Controller add

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 carico tutti i dati che verranno poi utilizzati nella pagina di gestione a cui verrà rimandato l'admin.

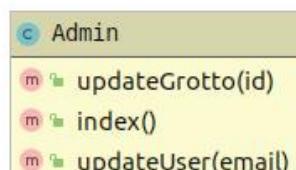


Figure 51 Controller admin

4.12.10 Controller gestione

La classe *Gestione* si occupa di coordinare il corretto funzionamento della pagina che da 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.

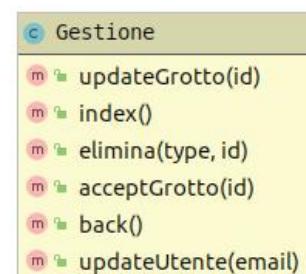


Figure 52 Controller gestione

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.

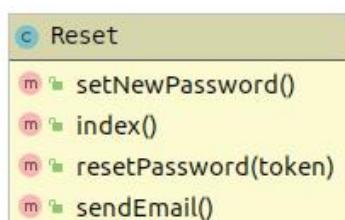


Figure 53 Controller reset

Piattaforma per la scoperta dei grotti del Ticino

5 Test

5.1 Protocollo di test

Test Case:	TC-01	Nome:	Test di funzionamento della mappa nella homepage		
Riferimento:	REQ-02	Descrizione: Verificare che la mappa che mostra i grotti situata nella prima pagina funzioni correttamente			
Prerequisiti:	Dovranno esserci dei grotti nel database (uno minimo) per vedere dei risultati				
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Cercare nella mappa il/i puntatore/i corrispondente/i al/i grotto/i 3. Premere su un puntatore con il mouse <ol style="list-style-type: none"> a. Premere all'esterno della finestra in un punto qualsiasi della mappa b. Premere su uno dei campi della finestra aperta 				
Risultati attesi:	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.				

Test Case:	TC-02	Nome:	Test di funzionamento della sezione di ricerca nella homepage		
Riferimento:	REQ-02	Descrizione: Verificare che la tabella che mostra i grotti situata nella prima pagina funzioni correttamente eseguendo anche il filtraggio delle informazioni			
Prerequisiti:	Dovranno esserci dei grotti nel database (uno minimo) per vedere dei risultati				
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Scendere fino alla sezione contenente la tabella 3. Verificare che i dati siano mostrati correttamente 4. Premere sui titoli delle varie colonne 5. Premere su una delle righe della tabella che non sia l'intestazione 				
Risultati attesi:	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.				

Piattaforma per la scoperta dei grotti del Ticino

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

Piattaforma per la scoperta dei grotti del Ticino

Test Case:	TC-04	Nome:	Test di funzionamento della pagina di ripristino della password
Riferimento:	REQ-03		
Descrizione:	Verificare che la pagina di ripristino della password funzioni correttamente.		
Prerequisiti:	Dovrà esistere l'utente nel database		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login 3. Selezionare la voce "<i>Password dimenticata?</i>" 4. Inserire un email di un account non esistente 5. Premere "<i>invia</i>" 6. Non inserire nulla nel campo 7. Premere "<i>invia</i>" 8. Inserire l'email di un account esistente 9. Verificare la casella di posta dell'indirizzo inserito 10. Se ricevuto il messaggio premere sul link 11. Inserire due password diverse 12. Premere "<i>invia</i>" 13. Non inserire nulla 14. Premere "<i>invia</i>" 15. Inserire una password incorrecta (meno di 8 caratteri) 16. Premere "<i>invia</i>" 17. Inserire due password uguali 18. Premere "<i>invia</i>" 19. Aprire la pagina di login e inserire le nuove credenziali 20. Premere "<i>accedi</i>" 		
Risultati attesi:	<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>		

Piattaforma per la scoperta dei grotti del Ticino

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

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

Piattaforma per la scoperta dei grotti del Ticino

Risultati attesi:	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.
	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.

Test Case:	TC-07	Nome:	Test di funzionamento della pagina di aggiunta di un grotto		
Riferimento:	REQ-05	Descrizione: Verificare che la pagina di aggiunta di un grotto funzioni correttamente			
Prerequisiti:	Dovrà esistere l'utente nel database				
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Selezionare nel menu in alto a destra (visualizzazione desktop) la voce login 3. Accedere con un utente registrato 4. Premere nel menu il link “Aggiungi Grotto” 5. Inserire nei campi dei valori sbagliati <ol style="list-style-type: none"> a. Inserire dei numeri o dei caratteri speciali nei campi “Nome”, “Paese” e “Via” b. Inserire delle lettere nel campo “CAP” c. Inserire dei caratteri speciali nel campo “Numero Civico” d. Inserire dei numeri di telefono non validi e. Lasciare i campi vuoti 6. Premere su “Aggiungi” 7. Inserire dei valori validi 8. Premere “Verifica” 9. Premere su “Aggiungi” 				
Risultati attesi:	<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>				

Piattaforma per la scoperta dei grotti del Ticino

Test Case:	TC-08	Nome:	Test di funzionamento della pagina di visualizzazione di un grotto		
Riferimento:	REQ-05				
Descrizione:	Verificare che la pagina di visualizzazione di un grotto funzioni correttamente quando vi si accede senza aver effettuato il login				
Prerequisiti:	Dovrà esistere il grotto nel database				
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto 3. Verificare che i dati siano corretti 4. Se vi sono immagini verificare il funzionamento dello slideshow 				
Risultati attesi:	<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>				

Test Case:	TC-09	Nome:	Test di funzionamento della pagina di visualizzazione di un grotto		
Riferimento:	REQ-05				
Descrizione:	Verificare che la pagina di visualizzazione di un grotto funzioni correttamente quando vi si accede avendo effettuato il login				
Prerequisiti:	<p>Dovrà esistere il grotto nel database</p> <p>Dovrà esistere l'account nel database</p>				
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Eseguire il login con un account 3. Andare nella homepage 4. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto 5. Verificare che i dati siano corretti 6. Se vi sono immagini verificare il funzionamento dello slideshow 7. Andare in fondo alla pagina 8. Aggiungere una votazione 9. Aggiungere una seconda votazione 10. Aggiungere un file <ol style="list-style-type: none"> a. Aggiungere un file che non sia un'immagine b. Aggiungere un'immagine già collegata ad un altro grotto c. Aggiungere un'immagine valida 				
Risultati attesi:	<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>				

Piattaforma per la scoperta dei grotti del Ticino

			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.
			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.
			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.

Test Case:	TC-10	Nome:	Test di funzionamento della condivisione sui social
Riferimento:	REQ-06		
Descrizione:	Verificare che la condivisione sui social (Facebook) di un grotto funzioni correttamente		
Prerequisiti:	Dovrà esistere il grotto nel database		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Selezionare nella lista o tramite la finestra ad apparizione collegata ai puntatori nella mappa un grotto 3. Premere il tasto “Condividi su Facebook” 		
Risultati attesi:	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.		

Test Case:	TC-11	Nome:	Test di funzionamento della pagina riservata agli admin
Riferimento:	REQ-01		
Descrizione:	Verificare che la pagina riservata agli admin funzioni correttamente		
Prerequisiti:	Dovrà esistere l'utente nel database ed avere i privilegi di amministratore		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il sito alla pagina principale 2. Accedere tramite la pagina di login con un account admin 3. Premere nel menu il link “Amministrazione” 4. Verificare che nelle tabelle “utenti”, “grotti”, “inserimenti” e “immagini” si vedano i dati giusti ordinati correttamente 5. Premere il tasto “modifica” di una riga delle tabelle “utenti” e “grotti” <ol style="list-style-type: none"> a. Verificare il funzionamento dei form ripetendo le procedure spiegate nei TC-03 e TC-07 b. Verificare il funzionamento del tasto “verifica” e del tasto “indietro” 6. Premere il tasto “accetta” di una riga nella tabella “inserimenti” 7. Verificare che i tasti di eliminazione in tutte le tabelle funzionino <ol style="list-style-type: none"> a. Premere sul tasto elimina nella riga del proprio utente 8. Premere su una delle righe nella tabella “immagini” 		
Risultati attesi:	<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>		

Piattaforma per la scoperta dei grotti del Ticino

	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 i 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.

Piattaforma per la scoperta dei grotti del Ticino

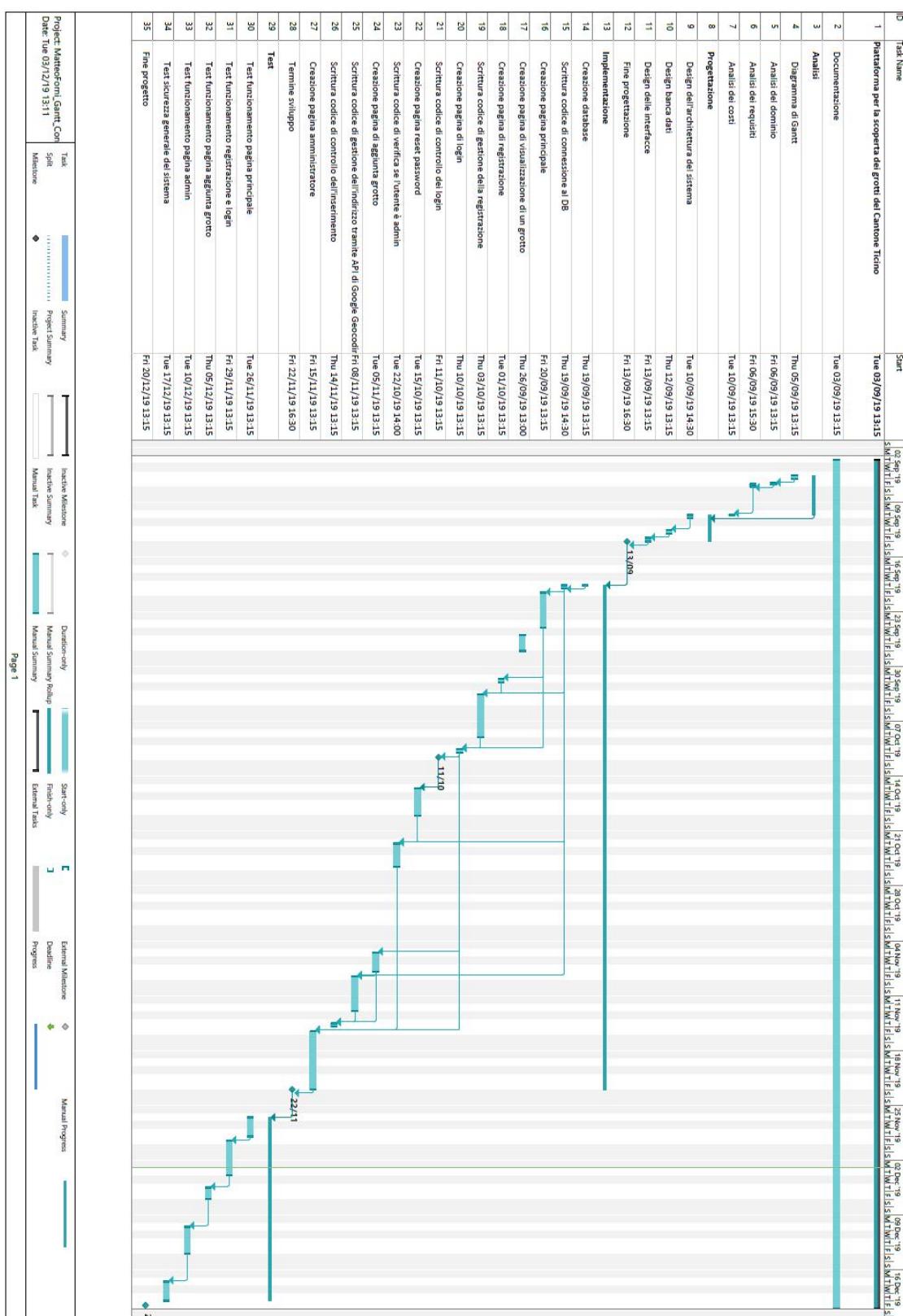


Figure 54 Diagramma di Gantt consuntivo

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/Dokumententliste%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, 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/>, MDBootstrap 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

Piattaforma per la scoperta dei grotti del Ticino

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

Diario di lavoro

Luogo	Trevano
Data	03.09.2019

Lavori svolti

Oggi durante la prima ora abbiamo avuto la possibilità di leggere tutti i quaderni dei compiti di tutti e venti i progetti. Al termine dell'ora abbiamo potuto scegliere quale progetto avremmo voluto seguire ed abbiamo ricevuto il QdC che abbiamo iniziato a leggere. Al termine della pausa ho avuto la possibilità di chiedere delle spiegazioni riguardo alcuni punti che non erano molto chiari con il docente responsabile Luca Peduzzi. Fatto ciò ho iniziato a pensare alla base da utilizzare per il codice ed ho deciso di utilizzare il pattern MVC. Dopodiché mi sono informato sulle possibilità esistenti per creare le mappe e per trovare le coordinate di un posto tramite il suo indirizzo, ho trovato le API di Google Geocoding la scelta migliore perché molto precise anche se la versione di sviluppo (gratis) ha delle possibilità in meno. Dopodiché mi sono occupato di creare una repository per il progetto su GitHub, software che utilizzerò come gestione delle versioni, ed iniziato a creare le cartelle per la documentazione, diari, presentazione e codice. Nella cartella del codice chiamata application ho inserito la base MVC fornitaci dal docente Sartori l'anno scorso.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio fare il diagramma di gantt ed iniziare l'analisi del dominio e se ho tempo l'analisi delle specifiche e requisiti.

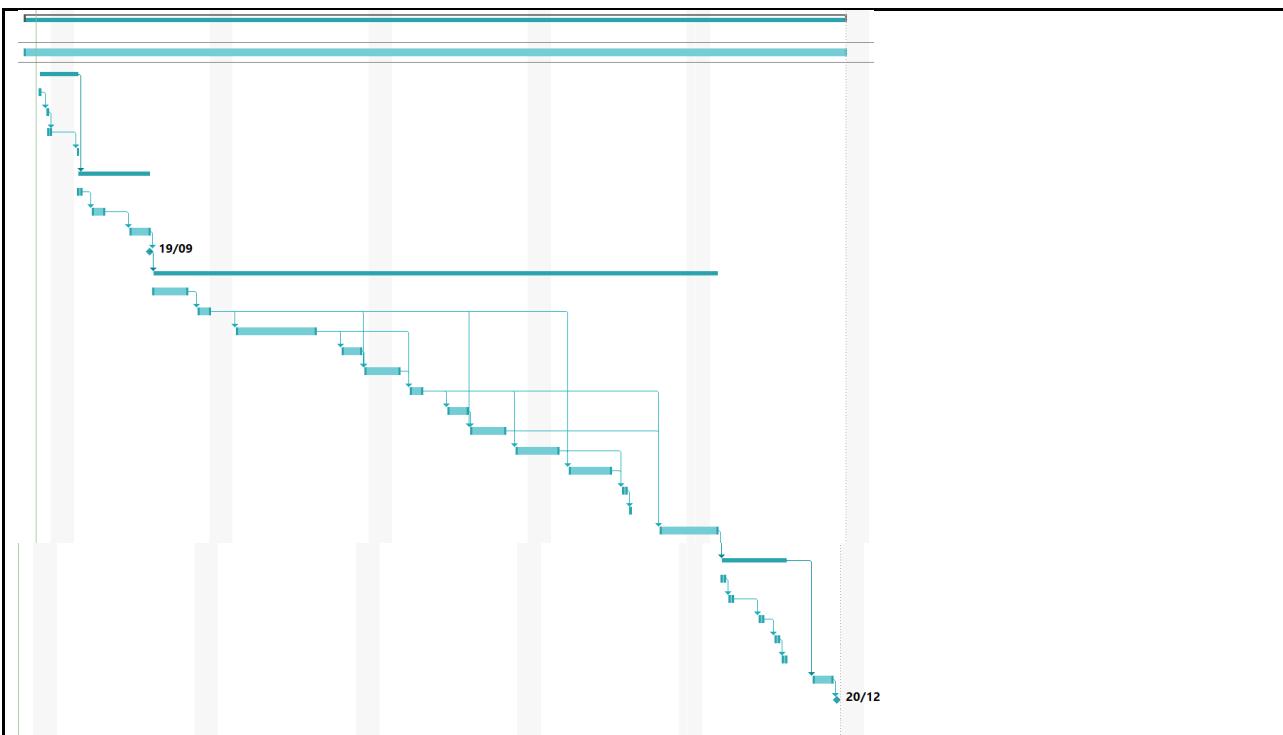
Diario di lavoro

Luogo	Trevano
Data	05.09.2019

Lavori svolti

Oggi per la prima parte della lezione ho installato il software per creare i diagrammi di Gantt Microsoft Project grazie ad un installer datoci da Valsangiacomo durante il secondo anno. Alla fine dell'installazione ho iniziato a creare il diagramma inserendo le attività che secondo me saranno le più importanti del progetto e assegnandogli una tempistica.

# Piattaforma per la scoperta dei grotti del Cantone Ticino	Tue 03/09/19 13:15	Fri 20/12/19 16:30	
Documentazione	Tue 03/09/19 13:15	Fri 20/12/19 16:30	
Analisi			
Diagramma di Gantt	Thu 05/09/19 13:15	Thu 05/09/19 13:30	
Analisi del dominio	Fri 06/09/19 13:15	Fri 06/09/19 15:30	4
Analisi dei requisiti	Fri 06/09/19 15:30	Fri 06/09/19 16:30	5
Analisi dei costi	Tue 10/09/19 13:15	Tue 10/09/19 14:30	6
Progettazione			3
Design dell'architettura del sistema	Tue 10/09/19 14:30	Tue 10/09/19 16:30	
Design banca dati	Thu 12/09/19 13:15	Fri 13/09/19 16:30	9
Design delle interfacce	Tue 17/09/19 13:15	Thu 19/09/19 16:30	10
Fine progettazione	Thu 19/09/19 16:30	Thu 19/09/19 16:30	11
Implementazione			12
Creazione database	Fri 20/09/19 13:15	Tue 24/09/19 16:30	
Scrittura codice di connessione al DB	Thu 26/09/19 13:15	Fri 27/09/19 16:30	14
Creazione pagina principale	Tue 01/10/19 13:15	Fri 11/10/19 16:30	15
Creazione pagina di registrazione	Tue 15/10/19 13:15	Thu 17/10/19 16:30	16
Scrittura codice di gestione della registrazione	Fri 18/10/19 13:15	Tue 22/10/19 16:30	17,18
Creazione pagina di login	Thu 24/10/19 13:15	Fri 25/10/19 16:30	16,18
Scrittura codice di controllo dei login	Tue 29/10/19 13:15	Thu 31/10/19 16:30	19
Scrittura codice di verifica se l'utente è admin	Fri 01/11/19 13:15	Tue 05/11/19 16:30	20,15
Creazione pagina di aggiunta grotto	Thu 07/11/19 13:15	Tue 12/11/19 16:30	19
Scrittura codice di gestione dell'indirizzo tramite API di Google Geocoding	Thu 14/11/19 13:15	Tue 19/11/19 16:30	22,15
Scrittura codice di controllo dell'inserimento	Thu 21/11/19 13:15	Thu 21/11/19 16:30	23,22
Creazione pagina di ricerca	Fri 22/11/19 13:15	Fri 22/11/19 14:15	24
Creazione pagina amministratore	Tue 26/11/19 13:15	Tue 03/12/19 16:30	21,19
Termino sviluppo	Tue 03/12/19 16:30	Tue 03/12/19 16:30	26
Test			27
Test funzionamento pagina principale	Thu 05/12/19 13:15	Thu 05/12/19 16:30	
Test funzionamento registrazione e login	Fri 06/12/19 13:15	Fri 06/12/19 16:30	29
Test funzionamento pagina aggiunta grotto	Tue 10/12/19 13:15	Tue 10/12/19 16:30	30
Test funzionamento pagina admin	Thu 12/12/19 13:15	Thu 12/12/19 16:30	31
Test sicurezza generale del sistema	Fri 13/12/19 13:15	Fri 13/12/19 16:30	32
Presentazione	Tue 17/12/19 13:15	Thu 19/12/19 16:30	28
Fine progetto	Fri 20/12/19 13:15	Fri 20/12/19 13:15	34



Come si può vedere dalle immagini gran parte del tempo è stata stimata di venire utilizzata durante l'implementazione. Questo è dovuto alle innumerevoli pagine che dovrò implementare sia frontend che backend. La documentazione dura tutto il tempo dato che essa verrà portata avanti con l'avanzare del progetto. La pianificazione è stata fatta dando molto tempo allo sviluppo perché prevedo che ci saranno lezioni in cui riscontrerò problemi e necessiterò di tempo per risolverli.

Dopo aver terminato il diagramma di Gantt ho scritto i capitoli "Informazioni sul progetto" e "Scopo" della documentazione.

Non sono riuscito ad iniziare l'analisi del dominio come programmato martedì dato che il Gantt mi ha preso parecchio tempo.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi tranne alcune difficoltà nel ricordarmi il funzionamento di determinate funzionalità di MS Project che ho risolto cercando sui forum ufficiali.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

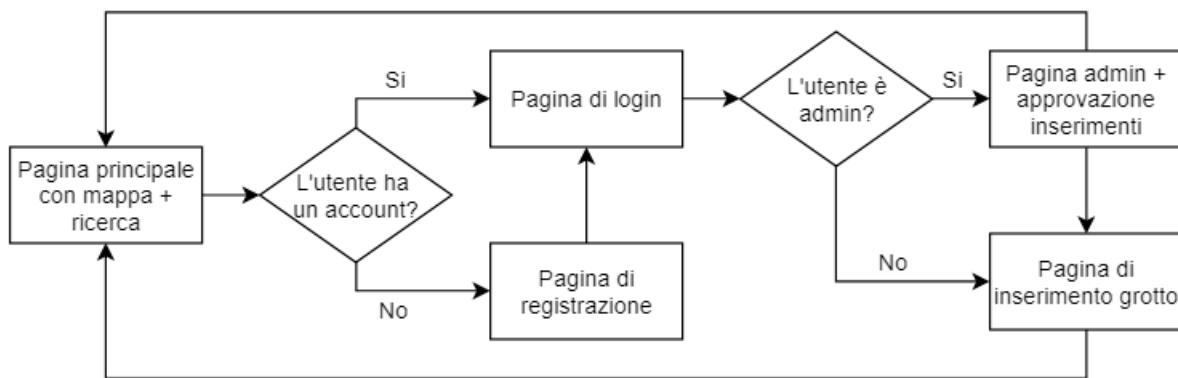
La prossima giornata voglio scrivere i capitoli della documentazione "analisi del dominio" e "analisi dei requisiti" e se avrò tempo iniziare il design dell'architettura del sistema.

Diario di lavoro

Luogo	Trevano
Data	06.09.2019

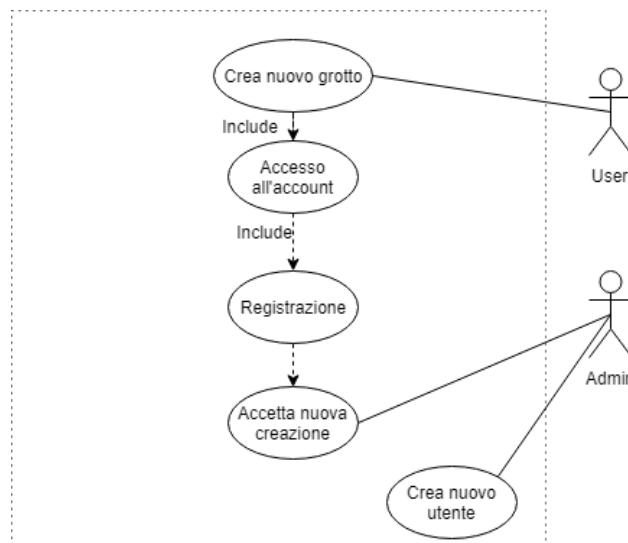
Lavori svolti

Oggi ho fatto come da programma il capitolo della documentazione “analisi del dominio” ed una prima bozza che sarà da discutere con il docente del capitolo “analisi e specifica dei requisiti”. Ho anche avuto tempo per iniziare e completare il design dell’architettura del sistema che si presenta così:



Esso rappresenta il flusso che si seguirà in teoria per muoversi nel sito e parte con la pagina principale che consentirà di passare ad una pagina di login/registrazione dopodiché si passerà, se non si è admin, alla pagina di creazione di un grotto mentre, se si hanno i privilegi di amministratore, alla pagina riservata ad essi.

Completato il diagramma dell’architettura del sistema ho iniziato a fare quello degli use case ma non sono riuscito a terminarlo. Questo è quello che ho fatto per oggi:



Problemi riscontrati e soluzioni adottate

Oggi gli unici problemi che ho riscontrato erano nel esportare il diagramma di gantt in una forma leggibile dato che esso è piuttosto grande. Ho inoltre avuto alcune difficoltà nell'iniziare li use case dato che non gli avevo mai fatti.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare gli ultimi capitoli di analisi quindi Use Case, spiegazione del gantt, hardware e software ed iniziare il diagramma ER dell'architettura del database.

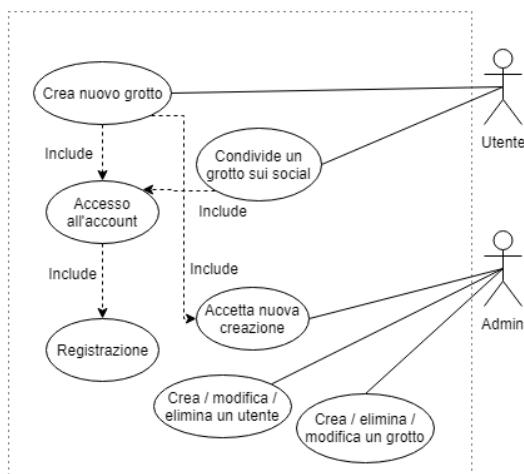
Diario di lavoro

Luogo	Trevano
Data	10.09.2019

Lavori svolti

La giornata di oggi è stata piuttosto incentrata sulla documentazione, ho infatti completato tutti i capitoli di analisi esclusa l'analisi dei mezzi che non posso ancora fare perché non conosco con certezza tutte le librerie che andrò ad utilizzare.

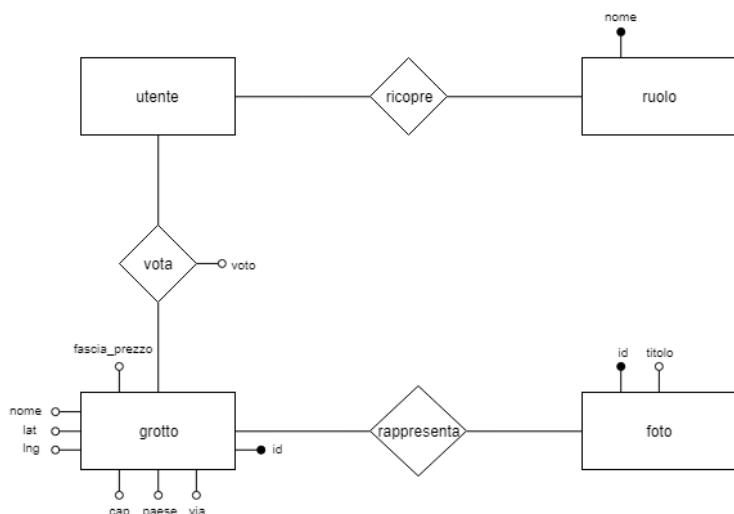
Oltre alla documentazione ho completato il diagramma degli use case



Esso ora include pure la condivisione sui social e tutte le attività dell'admin che non erano state inserite la volta scorsa.

Fatto ciò ho iniziato a redigere il diagramma ER del database prima su carta e poi su PC ma non ho avuto il tempo di terminare la copia sul computer.

Per adesso lo schema sul PC è il seguente:



Esso comprende quattro tabelle, le due più importanti sono utente e grotto che possiedono parecchi attributi (nello schema attuale non vi sono tutti). La tabella ruolo serve a specificare se un utente è admin o utente normale e l'ho fatta così che se in futuro si volesse aggiungere un nuovo ruolo (es. gerente grotto) si può fare senza problemi. Nella tabella grotto ci sono sia gli attributi latitudine e longitudine che l'indirizzo, questo è stato fatto perché contattare tutte le volte le API di Google Maps renderebbe il sito lento.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

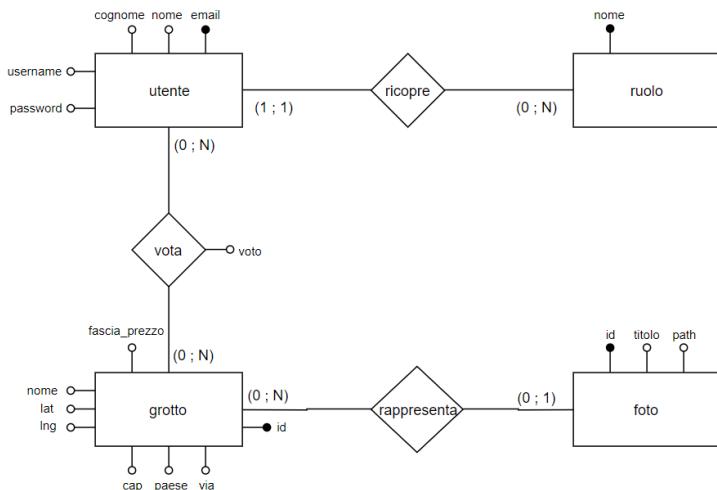
La prossima giornata voglio terminare l'ER e iniziare il design delle interfacce.

Diario di lavoro

Luogo	Trevano
Data	12.09.2019

Lavori svolti

Oggi ho completato il diagramma ER del database aggiungendo le part mancanti dall'ultima volta (gli attributi di alcune tabelle e le cardinalità) e l'ho spiegato sulla documentazione.



Mi sono reso conto durante la descrizione del capitolo nella documentazione che nella tabella immagini mancava un attributo "path" che ho poi aggiunto. Ho anche creato lo schema logico del DB.

```

RUOLO (nome)
UTENTE (email, nome, cognome, username, password, nome_ruolo(FK))
GROTO (id, via, paese, cap, lat, lng, nome, fascia_prezzo)
FOTO (id, titolo, path, id_grotto*(FK)) path unique
VOTO (email_utente(FK), id_grotto(fk), voto)
    
```

Fatto ciò sono tornato indietro nella documentazione al capitolo precedente e l'ho documentato, il diagramma del funzionamento del sistema era già stato fatto ma non era mai stato inserito nella documentazione.

Infine ho iniziato a creare il design delle interfacce, per adesso ho sviluppato quello dell'index, della pagina di login e di quella di registrazione.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono leggermente in anticipo (metà lezione circa).

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare il design delle interfacce e forse terminare la progettazione.

Diario di lavoro

Luogo	Trevano
Data	13.09.2019

Lavori svolti

Oggi ho completato i design delle interfacce creando quello della pagina di creazione di un grotto e quello della pagina admin. Inoltre l'ho documentato nell'apposito capitolo.

Grotti Ticino	Amministrazione	Home
Nome <input type="text"/> Fascia_Prezzo <input type="text"/> Valutazione <input type="text"/> CAP <input type="text"/> Paese <input type="text"/> Via <input type="text"/> No. Civico <input type="text"/>	La pagina di creazione di un grotto comprenderà i campi d'inserimento del nome, della fascia di prezzo, della valutazione e dell'indirizzo. Inoltre se si vorrà, tramite il tasto verifica, si potrà controllare se la posizione del puntatore viene messa nel punto giusto sulla mappa.	
<input type="button" value="Verifica"/> <input type="button" value="Salva"/> 		

Grotti Ticino	Aggiunta Grotto	Home																																																																							
Crea, modifica, elimina utenti <input type="button"/> Crea, modifica, elimina grotti <input type="button"/> Verifica nuovi inserimenti <input type="button"/>	La pagina admin di base contiene solamente tre buttoni. Premendo su il primo si aprirà un modale che consentirà di gestire gli utenti, con il secondo uno che farà gestire i grotti e con il terzo si potranno accettare/rifiutare gli inserimenti.																																																																								
	<table border="1"> <thead> <tr> <th>email</th> <th>username</th> <th>nome</th> <th>cognome</th> <th>modifica</th> <th>elimina</th> </tr> </thead> <tbody> <tr> <td>a@a.a</td> <td>a_b</td> <td>a</td> <td>b</td> <td>modifica</td> <td>elimina</td> </tr> <tr> <td>b@b.b</td> <td>c_d</td> <td>c</td> <td>d</td> <td>modifica</td> <td>elimina</td> </tr> <tr> <td>c@c.c</td> <td>e_f</td> <td>e</td> <td>f</td> <td>modifica</td> <td>elimina</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>nome</th> <th>indirizzo</th> <th>lat</th> <th>lon</th> <th>modifica</th> <th>elimina</th> </tr> </thead> <tbody> <tr> <td>a</td> <td>6900 a, a 10</td> <td>1</td> <td>2</td> <td>modifica</td> <td>elimina</td> </tr> <tr> <td>b</td> <td>6900 b, b 10</td> <td>3</td> <td>4</td> <td>modifica</td> <td>elimina</td> </tr> <tr> <td>c</td> <td>6900 c, c 10</td> <td>5</td> <td>6</td> <td>modifica</td> <td>elimina</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>user</th> <th>grotto</th> <th>voto</th> <th>prezzo</th> <th>accetta</th> <th>rifiuta</th> </tr> </thead> <tbody> <tr> <td>a</td> <td>6900 a, a 10</td> <td>1</td> <td>2</td> <td>accetta</td> <td>rifiuta</td> </tr> <tr> <td>b</td> <td>6900 b, b 10</td> <td>3</td> <td>4</td> <td>accetta</td> <td>rifiuta</td> </tr> <tr> <td>c</td> <td>6900 c, c 10</td> <td>5</td> <td>6</td> <td>accetta</td> <td>rifiuta</td> </tr> </tbody> </table>	email	username	nome	cognome	modifica	elimina	a@a.a	a_b	a	b	modifica	elimina	b@b.b	c_d	c	d	modifica	elimina	c@c.c	e_f	e	f	modifica	elimina	nome	indirizzo	lat	lon	modifica	elimina	a	6900 a, a 10	1	2	modifica	elimina	b	6900 b, b 10	3	4	modifica	elimina	c	6900 c, c 10	5	6	modifica	elimina	user	grotto	voto	prezzo	accetta	rifiuta	a	6900 a, a 10	1	2	accetta	rifiuta	b	6900 b, b 10	3	4	accetta	rifiuta	c	6900 c, c 10	5	6	accetta	rifiuta
email	username	nome	cognome	modifica	elimina																																																																				
a@a.a	a_b	a	b	modifica	elimina																																																																				
b@b.b	c_d	c	d	modifica	elimina																																																																				
c@c.c	e_f	e	f	modifica	elimina																																																																				
nome	indirizzo	lat	lon	modifica	elimina																																																																				
a	6900 a, a 10	1	2	modifica	elimina																																																																				
b	6900 b, b 10	3	4	modifica	elimina																																																																				
c	6900 c, c 10	5	6	modifica	elimina																																																																				
user	grotto	voto	prezzo	accetta	rifiuta																																																																				
a	6900 a, a 10	1	2	accetta	rifiuta																																																																				
b	6900 b, b 10	3	4	accetta	rifiuta																																																																				
c	6900 c, c 10	5	6	accetta	rifiuta																																																																				

Durante la lezione mi sono reso conto di aver dimenticato due attributi nella tabella grotto e quindi gli ho aggiunto i campi **no_civico** e **valutazione**.
 Fatto ciò ho creato il database su sql con il seguente codice:

Piattaforma per la scoperta dei Grotti del Canton Ticino

```
create database grotti;
use grotti;

create table ruolo(
    nome varchar(50) primary key
);

create table utente(
    email varchar(50) primary key,
    username varchar(50) not null,
    nome varchar(50) not null,
    cognome varchar(50) not null,
    password varchar(255) not null,
    nome_ruolo varchar(50) not null,
    foreign key(nome_ruolo) references ruolo(nome)
);

create table grotto(
    id int primary key auto_increment,
    nome varchar(50) not null,
    lon double not null,
    lat double not null,
    no_civico varchar(10) not null,
    via varchar(50) not null,
    paese varchar(50) not null,
    cap int not null,
    fascia_prezzo enum("Buon mercato", "Nella norma", "Caro") not null,
    valutazione int not null
);

create table foto(
    id int primary key auto_increment,
    titolo varchar(50) not null,
    path varchar(50) not null
);

create table voto(
    email_utente varchar(50),
    id_grotto int,
    voto int not null,
    primary key(email_utente, id_grotto),
    foreign key(email_utente) references utente(email),
    foreign key(id_grotto) references grotto(id)
);
```

Problemi riscontrati e soluzioni adottate
Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione
Sono in anticipo (circa una lezione).

Programma di massima per la prossima giornata di lavoro
La prossima giornata voglio creare la classe di collegamento al db e iniziare a sviluppare l'header e il footer. Inoltre dopo aver discusso con Paolo Weissaupt mi sono reso conto che sarebbe meglio creare una tabella a parte per la fascia di prezzo e anche questo lo farò la prossima volta.

Diario di lavoro

Luogo	Trevano
Data	19.09.2019

Lavori svolti

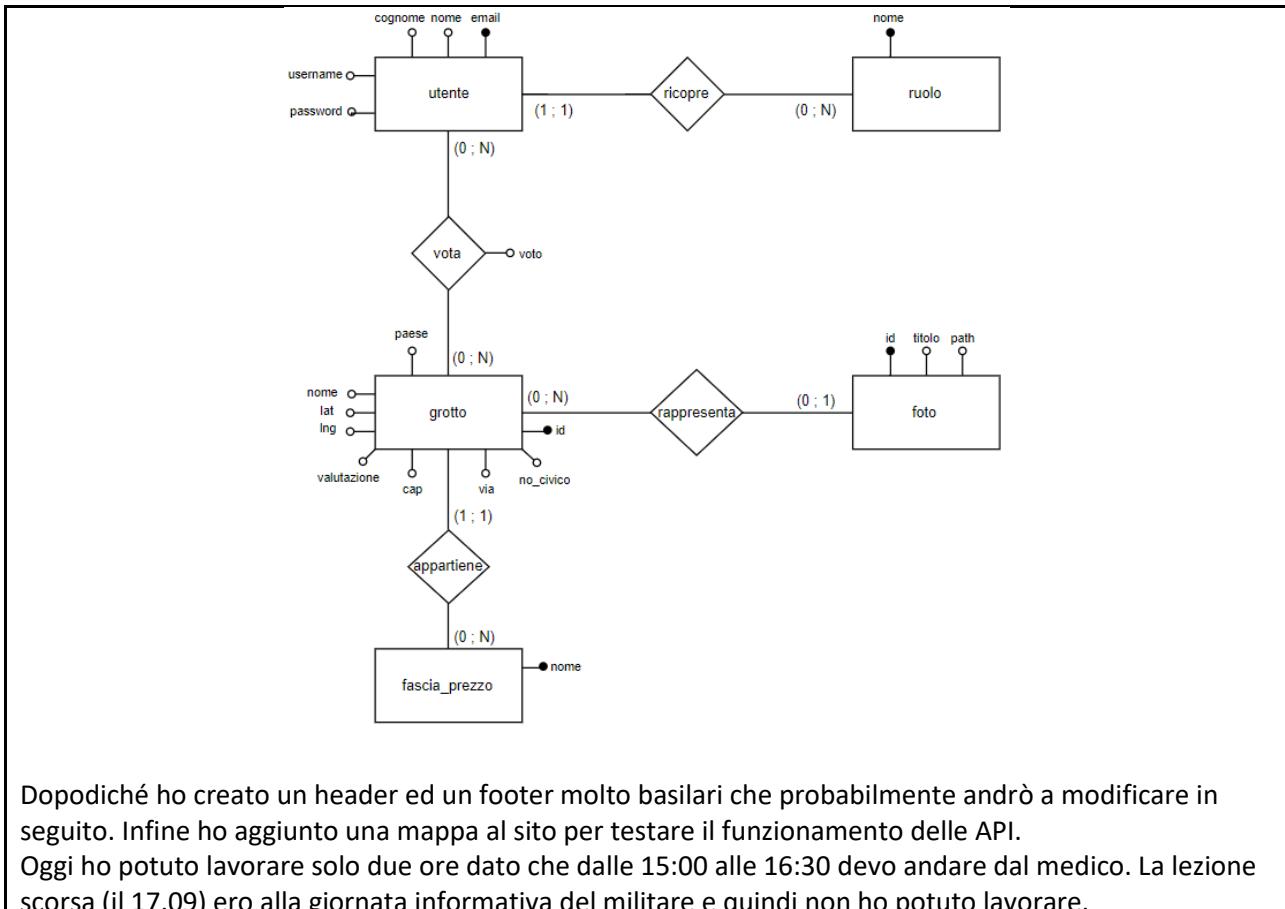
Oggi ho creato la classe di collegamento al database in PHP utilizzando PDO.

```
class db_connection
{
    private $_connection;

    private function __construct()
    {
    }

    /**
     * @return PDO La connessione con il database
     */
    public function getConnection()
    {
        if($this->_connection == null){
            try{
                $this->_connection = new PDO(DSN, USER, PASSWORD);
                return $this->_connection;
            }catch(PDOException $e){
                print "Error!: " . $e->getMessage() . "<br/>";
                die();
            }
        }
    }
}
```

Fatto ciò ho messo a posto l'ER del database e di conseguenza lo schema logico e il DB su mysql. Adesso, dopo aver aggiunto la tabella fascia_prezzo, l'ER appare così.



Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo (circa mezza lezione).

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio iniziare a implementare la pagina principale.

Diario di lavoro

Luogo	Trevano
Data	20.09.2019

Lavori svolti

Oggi ho iniziato l'implementazione della pagina iniziale. Come prima cosa ho messo la mappa in maniera responsive al centro della pagina. Dopodiché ho caricato, tramite la classe di connessione al database, tutti i grotti così da iniziare a stampare la lista di essi. Fatto ciò ho implementato, grazie al componente di material design bootstrap chiamato DataTable, i filtri su tutte le colonne. La documentazione con gli esempi si può trovare a questo link: <https://mdbootstrap.com/docs/jquery/tables/datatables/>. Infine ho aggiunto la colonna "telefono" alla tabella grotto di cui mi ero dimenticato ed ho cambiato il tipo di dato della colonna "valutazione" da int a double.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo (circa mezza lezione).

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare l'implementazione della pagina principale e iniziare la pagina di registrazione.

Diario di lavoro

Luogo	Trevano
Data	24.09.2019

Lavori svolti

Oggi ho continuato con l'implementazione della pagina principale. Inizialmente ho fatto sì che la mappa caricasse i marker dal database, questo è fatto grazie ad una funzione da me scritta che per ogni grotto nel DB genera un marker.

Fatto ciò ho aggiunto alla stessa funzione il codice che crea la finestra contenente i dettagli di un grotto che viene visualizzata quando si clicca su di un marker. Essa mi ha impiegato parecchio tempo dato che ho avuto dei problemi con JavaScript e PHP.

Inizialmente l'idea era quella di caricare i marker con PHP e passarli a javascript con cui li avrei creati ma dopo un po' di tentativi ho capito che questa non era la scelta migliore, ho quindi deciso di utilizzare l'array di PHP per ciclare i grotti e inserire i valori. Tutto ciò viene fatto nella funzione setMarkers().

Essa inizia con un loop di PHP:

```
function setMarkers(map, locations) {
    <?php foreach ($_SESSION['grotti'] as $row): ?>
    ...
}
```

Dopodiché contiene una variabile che rappresenta il codice html da creare che viene passata all'infowindow, questo è l'oggetto delle API di Google fatto apposta per creare delle finestre sopra i marker.

```
var contentString = `

<div class='content modal-body'>
    <h1 id="nome" style='color:black;'><?php echo $row['nome']; ?> </h1>
    <strong id="indirizzo">Indirizzo</strong>
    <?php echo(" " . $row['cap'] . ". " . $row['paese'] . ", "
    $row['via'] . ". " . $row['no_civico']); ?>
    <br>
    <strong id="telefono">Telefono</strong><?php echo " " .
    $row['telefono']; ?>
    <br>
    <strong id="valutazione">Valutazione</strong><?php echo " " .
    $row['valutazione']; ?>
    <br><br>
</div>`;

var infowindow<?php echo $row['id']; ?> = new google.maps.InfoWindow({
    content: contentString
});
```

Infine crea il marker e ad ogni marker associa un listener che, al click sul punto della mappa, genera l'infowindow e ci zooma sopra.

```
var marker<?php echo $row['id']; ?> = new google.maps.Marker({
    position: luogo,
    map: map,
    animation: google.maps.Animation.DROP,
    title: <?php echo "" . $row['nome'] . ""; ?>
});
```

```
marker<?php echo $row['id']; ?>.addListener('click', function() {  
    if (infowindowOpen != null) {  
        infowindowOpen.close();  
    }  
    infowindow<?php echo $row['id']; ?>.open(map, marker<?php echo  
$row['id']; ?>);  
    infowindowOpen = infowindow<?php echo $row['id']; ?>;  
  
    map.setZoom(13);  
    map.setCenter(marker<?php echo $row['id']; ?>.getPosition());  
});
```

Problemi riscontrati e soluzioni adottate

Oggi ho inizialmente riscontrato problemi con il passaggio dell'array da PHP a Javascript, utilizzando il metodo descritto da tutte le guide l'array era sempre nullo e non capivo il perché. Dopo aver chiesto aiuto a Filippo Finke e aver cercato l'errore per circa 15 minuti siamo giunti alla conclusione che si trattava di un problema di encoding, ho quindi provato a cambiare il charset di mysql da UTF-8 a UTF-8mb4 ma anche questo non ha portato a grandi risultati. Dopo ulteriori ricerche ho capito che era PHP che quando si connetteva al DB prendeva i dati giusti ma avendo lui l'encoding su UTF-8 li cambiava. Ho quindi risolto il problema aggiungendo il codice seguente al DSN.

```
define ('DSN', 'mysql:host=127.0.0.1;dbname=grotti; charset=utf8mb4');
```

Il secondo problema riscontrato è stato riguardo l'inserimento dei marker sulla mappa, infatti avevo provato in un modo presentato in una guida ma non ha funzionato. Esso creava infatti i marker ma avevano tutti lo stesso titolo e infowindow. Dopo vari tentativi ho ripreso un sito che abbiamo fatto io e Luca Di Bello durante il nostro stage in Irlanda e l'ho modificato. La soluzione finale è quella rappresentata sopra nel punto «Lavori svolti».

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare l'implementazione della pagina principale e iniziare la pagina di registrazione.

Diario di lavoro

Luogo	Trevano
Data	26.09.2019

Lavori svolti

Oggi ho continuato con l'implementazione della pagina principale. Inizialmente ho messo a posto un piccolo problema che ho notato nella pagina principale: lo zoom sulla mappa non sempre eseguiva l'animazione e l'ho risolto cambiando il valore che esso doveva raggiungere (io richiedevo uno zoom di 7 quando il minimo era 9). Dopodiché ho iniziato a implementare il frontend della pagina di login, questo mi serviva dato che è da essa che si può passare a quella di registrazione. Grazie al framework, Material Design Bootstrap, che sto utilizzando l'operazione è stata piuttosto veloce. Fatto ciò ho creato anche il frontend della pagina di registrazione con lo stesso metodo e stile di quella di login. Ho poi continuato facendo i controlli backend dei dati inseriti, essi verificano inizialmente che il metodo di connessione sia corretto (post), poi che tutti i campi siano stati compilati ed infine richiama una funzione che elimina da essi possibili caratteri speciali e malevoli.

La classe è la seguente:

```
class InputManager {
    public function checkInput($input) {
        $input = trim($input);
        $input = stripslashes($input);
        $input = htmlspecialchars($input);
        return $input;
    }
}
```

Una volta controllati tutti i valori essi vengono controllati con degli if che ne verificano la lunghezza.

```
$im = new InputManager();
$firstname = filter_var($im->checkInput($_POST['firstname']), FILTER_SANITIZE_STRING);
$lastname = filter_var($im->checkInput($_POST['lastname']), FILTER_SANITIZE_STRING);
$username = filter_var($im->checkInput($_POST['username']), FILTER_SANITIZE_STRING);
$email = filter_var($im->checkInput($_POST['email']), FILTER_SANITIZE_EMAIL);
$password = filter_var($im->checkInput($_POST['password']), FILTER_SANITIZE_STRING);
$repassword = filter_var($im->checkInput($_POST['repassword']), FILTER_SANITIZE_STRING);

if(!(strlen($firstname) > 0 && strlen($firstname) < 50)){
    array_push($errors, "Il nome deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($lastname) > 0 && strlen($lastname) < 50)){
    array_push($errors, "Il cognome deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($username) > 0 && strlen($username) < 50)){
    array_push($errors, "Lo username deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($email) > 0 && strlen($email) < 50)){
    array_push($errors, "L'email deve essere formattata nel seguente modo:
    indirizzo@dominio.xx");
}
if(!(strlen($password) >= 8)){
    array_push($errors, "La password deve essere almeno lunga 8 caratteri");
}
```

Se anche qui tutto risulta corretto si passa a controllare che le due password siano uguali e che l'email non sia già in uso.

```
if($password == $repassword) {  
    $db = (new db_connection)->getUsers();  
  
    foreach ($db as $row) {  
        if ($row['email'] == $email) {  
            array_push($errors, "L'email è già in uso");  
            $exists = true;  
        }  
    }  
}
```

Se tutto risulta corretto si procede con l'inserire l'utente nel database, questo viene fatto richiamando la seguente funzione.

```
public function addUser($firstname, $lastname, $username, $email, $password) {  
    $db = $this->getConnection();  
    $query = $db->prepare('insert into utente(email, nome, cognome, username,  
password, `nome_ruolo`) values (:email, :firstname, :lastname, :username,  
:password, :nome_ruolo')');  
  
    $password = hash('sha256', $password);  
    $type = 'utente';  
  
    $query->bindParam(':email', $email, PDO::PARAM_STR);  
    $query->bindParam(':firstname', $firstname, PDO::PARAM_STR);  
    $query->bindParam(':lastname', $lastname, PDO::PARAM_STR);  
    $query->bindParam(':username', $username, PDO::PARAM_STR);  
    $query->bindParam(':password', $password, PDO::PARAM_STR);  
    $query->bindParam(':nome_ruolo', $type, PDO::PARAM_STR);  
  
    $query->execute();  
}
```

Essa esegue l'insert nella tabella utilizzando i prepared statements che impediscono le SQL Injections.

Se nelle verifiche si riscontrano errori si dovrà rimandare l'utente alla pagina di login e mostrare i messaggi d'errore.

Problemi riscontrati e soluzioni adottate

Oggi ho avuto un solo grande problema che è stato il fatto che, dopo aver inserito l'utente nel database, richiamando la pagina di login lo stile non venisse caricato. Questo era dovuto al fatto che il require (che utilizzavo io) non ricarica la pagina ma aggiunge alla corrente il contenuto di un file php e quindi non si caricavano gli stili. Ho risolto il problema chiedendo aiuto al docente Sartori che mi ha consigliato di utilizzare il metodo header al posto di require.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare l'implementazione della pagina di registrazione e iniziare la pagina di login (backend).

Diario di lavoro

Luogo	Trevano
Data	27.09.2019

Lavori svolti

Oggi ho continuato con l'implementazione della pagina di registrazione, non sono riuscito a fare molto perché subito dall'inizio mi sono bloccato con un problema con la scrittura nel database. Risolto il problema, dopo le prime due ore, sono dovuto andare ad una riunione di resoconto finale dello stage all'estero che ho fatto quest'estate e quindi le due ore seguenti non ho avuto l'occasione di fare molto. La prossima volta testerò tutta la pagina ma in teoria è finita.

Problemi riscontrati e soluzioni adottate

Dopo un'ora di lezione a provare a capire dove avesse problemi il metodo di inserimento di un utente nel database sono riuscito a trovarne il problema. Questo era nel metodo che scrive nel db che, utilizzando i prepared statements, non scriveva ma non ritornava nemmeno errori, cercando su internet ho trovato un secondo metodo di utilizzare i prepared statements e utilizzandolo il problema non si presenta più. Il metodo di scrittura nel db addUser() ora si presenta così:

```
public function addUser($firstname, $lastname, $username, $email, $password) {
    try {
        $db = $this->getConnection();
        $query = $db->prepare('INSERT INTO utente(email, nome, cognome, username,
password, nome_ruolo) VALUES (?, ?, ?, ?, ?, ?)');
        $password = hash('sha256', $password);
        $type = "utente";

        $query->bindParam(1, $email, PDO::PARAM_STR);
        $query->bindParam(2, $firstname, PDO::PARAM_STR);
        $query->bindParam(3, $lastname, PDO::PARAM_STR);
        $query->bindParam(4, $username, PDO::PARAM_STR);
        $query->bindParam(5, $password, PDO::PARAM_STR);
        $query->bindParam(6, $type, PDO::PARAM_STR);

        $query->execute();
    } catch (Exception $e) {
        header('Location: ' . URL . 'db_error');
    }
}
```

La parte finale, nel catch, serve a richiamare una pagina d'errore così da non mostrare all'utente finale un errore di PHP poco chiaro.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio iniziare la pagina di login (backend).

Diario di lavoro

Luogo	Trevano
Data	01.10.2019

Lavori svolti

Oggi ho iniziato con l'implementazione del backend della pagina di registrazione e l'ho portato a termine. Ho iniziato creando una funzione che verifica il login di un utente verificando che l'email sia in uso da un account e, se così fosse, che la password corrisponda con quella dell'account.

```
public function checkLogin(){
    //richiamo le classi di cui avrò bisogno
    unset($_SESSION['warning']);
    require_once "./application/models/db_connection.php";
    require_once "./application/models/input_manager.php";

    $errors = array();

    //verifico il metodo di richiesta
    if($_SERVER["REQUEST_METHOD"] == "POST") {
        //verifico che i campi siano impostati e che non siano stringhe vuote
        if (isset($_POST['email']) && !empty($_POST['email']) &&
            isset($_POST['password']) && !empty($_POST['password'])) {

            //genero un nuovo InputManager e testo gli inserimenti
            $im = new InputManager();

            $email = filter_var($im->checkInput($_POST['email']),
FILTER_SANITIZE_EMAIL);
            $password = filter_var($im->checkInput($_POST['password']),
FILTER_SANITIZE_STRING);

            //prendo tutti gli utenti nel database
            $db = (new db_connection)->getUsers();

            //hash della password così da poterla comparare con quella nel db
            $password = hash('sha256', $password);
            foreach ($db->fetchAll() as $row) {
                //controllo che l'email sia in uso da un utente
                if ($row['email'] == $email) {
                    //controllo che la password corrisponda
                    if($row['password'] == $password){
                        $_SESSION['user'] = (new db_connection)->getUser($email);
                        //verifico se è admin o utente normale
                        if($row['nome_ruolo'] == 'admin'){
                            header('Location: ' . URL . 'admin');
                            exit();
                        }elseif($row['nome_ruolo'] == 'utente'){
                            header('Location: ' . URL . 'home');
                            exit();
                        }
                    }else{
                        //se la password è sbagliata ritorno l'errore
                        array_push($errors, "Password o email sbagliate");
                        $_SESSION['warning'] = $errors;
                        header('Location: ' . URL . 'login');
                        exit();
                    }
                }
            }
            //se l'email non è in utilizzo da nessun user ritorno l'errore
            array_push($errors, "Password o email sbagliate");
            $_SESSION['warning'] = $errors;
            header('Location: ' . URL . 'login');
            exit();
        }
    }
}
```

Dopo aver scritto e testato la funzione ho implementato alcune piccole cose nei controller ad esempio nell'home se si è loggati e a dipendenza di che livello di privilegi si hanno si vedono degli header diversi. Headers che sono andato a modificare oggi aggiungendo i percorsi alle pagine e inserendo nei due per gli utenti loggati la possibilità di uscire dall'account dato che ho creato la funzione.

```
public function logout(){
    unset($_SESSION['user']);
    header('Location: ' . URL . 'home');
    exit();
}
```

Ho inoltre commentato tutti i controllers e i models che non avevano ancora tutte le spiegazioni.

Per far sì che il login funzionasse ho implementato la funzione getUser nel model di gestione del database. Essa ritorna un utente in base alla sua email.

```
public function getUser($email){
    try{
        $db = $this->getConnection();

        $query = $db->prepare('SELECT * FROM utente where email=?');
        $query->bindParam(1, $email, PDO::PARAM_STR);

        $query->execute();

        return $query->fetchAll();
    }catch (Exception $e){
        header('Location: ' . URL . 'warning');
    }
}
```

Problemi riscontrati e soluzioni adottate

Inizialmente ho riscontrato un problema con il login dato che nel database avevo degli hash non corretti e quindi facendo il paragone la password risultava sempre sbagliata. Ho risolto eliminando i record e reinserendoli nel database dalla pagina di registrazione.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio creare i validatori frontend per le pagine di registrazione e di login.

Diario di lavoro

Luogo	Trevano
Data	03.10.2019

Lavori svolti

Oggi come da programma ho impostato i controlli frontend con javascript, essi vengono fatti con delle classi mie e di Luca Di Bello che utilizzavamo nello stage in Irlanda, di cui avevo già testato il funzionamento e sapevo come utilizzarle. Esse, al caricamento della pagina, assegnano a tutti gli input un listener sia on key up (bottone si alza) che on key down (pressione del bottone). Quando il listener viene richiamato nella classe register la funzione manage verifica quale è il campo utilizzato e richiama il validatore corretto dalla classe validate. Se la validazione va a buon fine la parte bassa dell'input si colora di verde altrimenti di rosso e appare un messaggio d'errore.

Fatto ciò ho creato una pagina interattiva che consente di, dopo aver premuto su una linea della tabella nella home, di visualizzare i dettagli e le immagini di un grotto. Per fare ciò ho dovuto creare una funzione che prende i dati dal database di un grotto in base all'id passato.

```
public function getGrotto($id) {
    try{
        $db = $this->getConnection();
        $query = $db->prepare('SELECT * FROM grotto WHERE id=?');
        $query->bindParam(1, $id);
        $query->execute();
        return $query->fetchAll();
    }catch (Exception $e){
        header('Location: ' . URL . 'warning');
    }
}
```

Ho anche scritto la funzione che ritorna tutte le immagini di un grotto così da mostrarle nella pagina sotto forma di un carousel (slide show).

```
public function getImages($id) {
    try{
        $db = $this->getConnection();
        $query = $db->prepare('SELECT * FROM foto WHERE grotto=?');
        $query->bindParam(1, $id);
        $query->execute();
        return $query->fetchAll();
    }catch (Exception $e){
        header('Location: ' . URL . 'warning');
    }
}
```

Inoltre ho aggiunto al database, nella tabella utente, i campi reset_token (il token per resettare la password) e first_login (un booleano che indica se l'utente è stato creato dall'admin o no).

Problemi riscontrati e soluzioni adottate

L'unico problema che ho riscontrato oggi è che avevo dimenticato la foreign key del grotto nella tabella foto e in un secondo momento che la colonna path consentiva di inserire solamente dei campi di 50 caratteri ma un percorso è facilmente più lungo.

Ho risolto inserendo la colonna grotto e impostando path come varchar(255).

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio completare la pagina di gestione degli errori e iniziare la pagina di aggiunta di un grotto.

Diario di lavoro

Luogo	Trevano
Data	04.10.2019

Lavori svolti

Oggi come da programma ho messo a posto la pagina d'errore, essa viene richiamata ad ogni errore con il DB e le viene passato il codice ed il messaggio d'errore. Nella pagina viene stampato un messaggio generico per gli utenti di base (C'è stato un errore! Aspetta o riprova ...) e anche il messaggio d'errore con il codice di MySQL per gli utenti che necessitano di risolvere il problema.

Fatto ciò ho iniziato a creare la pagina di aggiunta dei grotti, inizialmente ho creato la pagina con il form e impostato tutti i campi di esso dopodiché ho perso parecchio tempo cercando una libreria che consentisse di inserire nella pagina una valutazione fatta con le stelle. Dopo un po' di ricerche ho trovato «auxiliary-review» fatta appunto da «auxiliary». Ho impiegato circa mezz'ora a capire come farla funzionare dopodiché l'ho implementata.

Dato che ho perso parecchio tempo ad informarmi sulla libreria di valutazione non sono riuscito a procedere ulteriormente con la pagina.

Inoltre ho aggiunto al DB il campo verificato alla tabella grotto, esso serve per capire se un grotto è già stato verificato da un admin o meno. Se il grotto non è verificato apparirà nella pagina di amministrazione per essere verificato/scartato altrimenti mente se è già stato approvato verrà mostrato nella pagina principale.

Problemi riscontrati e soluzioni adottate

Mi ha preso abbastanza tempo riuscire a far sì che il valore scelto dall'utente venga passato con il metodo post insieme agli altri elementi del form. Ho risolto il problema inserendo nel form un input nascosto a cui, tramite jQuery, assegno come valore il numero scelto dall'utente sotto forma di stelle.

```
//Ad ogni click setto come value dell'input nascosto val il valore della valutazione
$(".rating").click(function () {
    var value = String($(".rating").rate("getValue"));
    $('#val').val(value);
});
```

Un altro piccolo problema l'ho riscontrato cercando di centrare le stelle, facendolo tramite CSS infatti esse smettevano di funzionare. Ho dovuto quindi farlo tramite JQuery.

```
//Mantengo sempre centrate le stelle
var margin = $('.rating-container').width()/2 - $('.rating').width()/2;
$('.rating').css('margin-left', margin);
$(window).resize(function () {
    var margin = $('.rating-container').width()/2 - $('.rating').width()/2;
    $('.rating').css('margin-left', margin);
});
```

Questo viene fatto una volta per quando viene caricata la pagina e poi ogni volta che la dimensione della finestra varia.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio completare la pagina di aggiunta di un grotto.

Diario di lavoro

Luogo	Trevano
Data	08.10.2019

Lavori svolti

Oggi come da programma ho continuato con la pagina di aggiunta di un grotto. Teoricamente essa è finita ma non sono in grado di testarlo dal momento che non funzionano le mappe di Google.

Per completarla ho dovuto mettere a posto la validazione front end e creare tutto il backend. Esso è piuttosto simile a quello della pagina di registrazione, inizialmente si verifica il metodo di invio del form dopodiché se tutti i campi sono stati compilati, se non vi sono problemi si passa a controllare che i campi non contengano stringhe malevoli e che la lunghezza di esse sia corretta.

Fatto ciò tramite la funzione addGrotto si aggiungerà un grotto al database, se sarà stato creato da un admin sarà già «verificato» altrimenti dovrà essere controllato da un admin.

Inoltre ho aggiunto alla pagina una mappa che consentirà, tramite il Geocoding, di verificare che l'indirizzo sia stato inserito correttamente dato che nel database verranno salvate le coordinate ritornate proprio dalle API di Geocoding.

Problemi riscontrati e soluzioni adottate

Il problema che ho riscontrato oggi è che Google ha cambiato le politiche di fatturazione e ora richiedono una carta di credito per poter usare le loro API. Sfortunatamente la mia carta di credito è di Postfinance e loro non consentono a Google di effettuare prelievi automatizzati dalle carte di credito. Per questo motivo sto creando una carta virtuale di Revolut che teoricamente dovrebbe funzionare. Adesso sono in attesa di esser verificato.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio completare la pagina di aggiunta di un grotto e iniziare ad implementare i voti.

Diario di lavoro

Luogo	Trevano
Data	10.10.2019

Lavori svolti

Oggi come da programma ho provato a risolvere i problemi relativi alla pagina di creazione del grotto, di base la pagina funziona (inserisce nel DB) ma devo passare delle coordinate a caso hardcoded dato che la API di Google non funzionano e quindi non possono generare i dati dall'indirizzo.

Fatto ciò ho sviluppato il voto dei grotti, inizialmente ho fatto alcune modifiche al database dato che ho notato che aveva dei problemi, ho infatti modificato il tipo di dato del voto da int a float ed ho rimosso, nella tabella grotto, al campo votazione il «not null» così che adesso di base un grotto può non avere voti. Questo ha implicato la rimozione della votazione iniziale quando si crea un grotto dato che non mi sembrava corretto poterlo fare.

Per far funzionare il tutto ho creato un trigger in mysql, esso parte dopo ogni inserimento nella tabella voto ed esegue la media dei voti del grotto che ne ha appena ricevuto uno ed inserisce nella tabella grotto la nuova media.

```
/*CREAZIONE TRIGGER AFTER INSERT SULLA TABELLA VOTO*/
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 ;
```

Fatto ciò ho aggiunto la valutazione con le stelle alla tabella iniziale e nei marker della mappa, quella nella tabella so che funziona ma purtroppo quella nella mappa non posso visualizzarla quindi non posso garantire il corretto funzionamento.

Problemi riscontrati e soluzioni adottate

Oggi i problemi che ho riscontrato sono stati inizialmente relativi al trigger in mysql, infatti io lo avevo fatto seguendo la documentazione ufficiale ma non riuscivo a farlo funzionare. Dopo svariato tempo perso ho provato a chiedere al docente Sartori ma nemmeno lui è stato di aiuto. Alla fine sono riuscito a risolvere il problema chiedendo aiuto al docente Muggiasca che insegna un modulo su mysql che comprende i triggere e lui mi ha consigliato di aggiungere il delimiter.

Un altro problema che ho avuto è stato che non riuscivo a passare correttamente dalla pagina di votazione al database le votazioni con numeri non interi. Dopo un po' di ricerche ho risolto inserendo un'opzione al metodo di verifica dei numeri di PHP.

```
$valutazione=filter_var($im->checkInput($_POST['val']),FILTER_SANITIZE_NUMBER_FLOAT,
FILTER_FLAG_ALLOW_FRACTION);
```

L'opzione FILTER_FLAG_ALLOW_FRACTION consente appunto di inserire numeri con la virgola e quindi il mio problema si è risolto.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Piattaforma per la scoperta dei Grotti del Canton Ticino

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio iniziare ad implementare il cambio della password e magari risolvere i problemi con le API di Google.

Diario di lavoro

Luogo	Trevano
Data	11.10.2019

Lavori svolti

Oggi durante le prime due ore non ho potuto lavorare dato che il docente Valsangiacomo ci ha spiegato come funzionano le valutazioni dei progetti di esame.

Nelle due ore dopo la pausa ho iniziato a scrivere il codice di reset della password, esso è suddiviso su un controller ed un model. Nel controller, dopo aver caricato un form con un input solo e aver atteso l'inserimento di un'email da parte dell'utente, verifico che i dati passati siano corretti e che l'email sia effettivamente in uso. Se tutti i requisiti sono soddisfatti viene richiamato un metodo nel model di gestione del database che genera un token e lo inserisce nel campo dell'utente «reset_token».

```
$db = $this->getConnection();
$query = $db->prepare('UPDATE utente SET reset_token=? WHERE email=?');

$token = bin2hex(random_bytes(15));

$query->bindParam(1, $token);
$query->bindParam(2, $email);

$query->execute();

return $token;
```

Fatto ciò viene richiamato il model di gestione dell'email che, dopo aver impostato tutte le opzioni correttamente, invia l'email con il link alla pagina di reset (non ancora creata).

```
$body = "Per reimpostare la tua password premi sul seguente link: \n" . URL . "reset/
reset/" . $token;

try{
    $mail = new PHPMailer();
    ... opzioni ...
    $mail->SetFrom(EMAIL);
    $mail->Subject = "Grotti Ticino - Reimposta la tua password";
    $mail->Body = $body;
    $mail->AddAddress($email);

    if(!$mail->Send()) {
        $error = 'Mail error: ' . $mail->ErrorInfo;
        $_SESSION['warning'] = $error;
        header('Location: ' . URL . 'warning');
        exit();
    } else {
        return true;
    }
} catch (Exception $e) {
    $_SESSION['warning'] = $e->getCode() . " - " . $e->getMessage();
    header('Location: ' . URL . 'warning');
    exit();
}
```

Per fare tutto ciò ho dovuto generare un nuovo account gmail chiamato grottiticino@gmail.com.

Problemi riscontrati e soluzioni adottate

Il problema che ho riscontrato oggi è che quando provo ad inviarmi un'email di reset della password il sito inizia a caricare all'infinito e non invia alcuna email però non genera nemmeno degli errori. Non sono riuscito a capire cosa e dove sia il problema.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare il reset della password.

Diario di lavoro

Luogo	Trevano
Data	15.10.2019

Lavori svolti

Oggi durante le prime due ho risolto i problemi riscontrati la lezione precedente, esso era dovuto al fatto che la rete della scuola bloccava il traffico in uscita e entrata con il protocollo TTLS quindi il sito cercava di collegarsi all'infinito senza poterlo mai farlo. Sono venuto a conoscenza di questa cosa perché anche Filippo Finke stava avendo lo stesso problema ed ha scoperto che il traffico era bloccato. Abbiamo quindi chiesto al docente Raimondi di sbloccare il traffico del protocollo sulla porta 587 e il problema si è risolto. A questo punto ho iniziato a testare che tutto funzionasse correttamente.

Per poter cambiare le password ho dovuto creare una nuova funzione nella classe db_connection.

```
public function setPassword($email, $password) {
    try{
        $db = $this->getConnection();
        $query = $db->prepare('UPDATE utente SET password=? WHERE email=?');

        $password = hash('sha256', $password);
        $query->bindParam(1, $password);
        $query->bindParam(2, $email);

        $query->execute();

    }catch (Exception $e){
        $_SESSION['warning'] = $e->getCode() . " - " . $e->getMessage();
        header('Location: ' . URL . 'warning');
        exit();
    }
}
```

Ed essa viene richiamata dal controller reset.

```
...
$user = (new DBConnection())->getUser($_SESSION['mail_sent']);

if($user['reset_token'] == $token) {

    (new DBConnection())->setPassword($_SESSION['mail_sent'], $_SESSION['password']);
    $_SESSION['password_change'] = true;
...
}
```

Fatto ciò, dato che ho ricevuto una chiave delle API di Google funzionante, ho potuto testare la pagina di aggiunta dei grotti che comprende il geocoding. Anche essa praticamente funzionava già ed ho quindi dovuto modificare solamente dei piccoli dettagli.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato grossi problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio iniziare la pagina di amministrazione.

Diario di lavoro

Luogo	Trevano
Data	17.10.2019

Lavori svolti

Oggi ho inizialmente messo a posto un problema che ho notato testando il sito ed era che le stelle delle valutazioni nella pagina dei grotti non erano selezionabili e quindi non si poteva votare. Questo era dovuto a delle classi CSS sbagliate ed ho risolto rimuovendo le classi inutili.
Fatto ciò sono passato a creare la pagina di admin, come prima cosa ho creato l'interfaccia di base della pagina che si presenta così.

The screenshot shows the homepage of the administration interface. At the top, there is a header bar with the text "Grotti Ticinesi" on the left and navigation links "Home", "Amministrazione", "Aggiungi Grotto", and "Logout" on the right. Below the header, the title "Amministrazione grotti Ticino" is centered, followed by the text "Connesso come: Matteo Forni (matteoforni)". There are three main buttons: "Gestione utenti" (with a blue "GESTISCI UTENTI" button), "Gestione Grotti" (with a blue "GESTISCI GROTTI" button), and "Gestione Inserimenti" (with a blue "GESTISCI INSERIMENTI" button). At the bottom of the page, a footer bar contains the text "© 2018 Copyright: Piattaforma per la scoperta dei grotti del Canton Ticino".

Quando si preme uno dei tre button appare la tabella relativa.

The screenshot shows the "Gestione utenti" (User Management) page. The title "Amministrazione grotti Ticino" and user information "Connesso come: Matteo Forni (matteoforni)" are at the top. Below is a table with columns: Email, Nome, Cognome, Username, Ruolo, Modifica (yellow button), and Elimina (red button). Two rows of data are shown:

Email	Nome	Cognome	Username	Ruolo	Modifica	Elimina
matteo.forni@samtrevano.ch	Matteo	Forni	matteoforni	admin	MODIFICA	ELIMINA
matteoforni@hotmail.com	Matteo	Forni	fornimatteo	utente	MODIFICA	ELIMINA

Below the table, there are links for "Gestione Grotti" and "Gestione Inserimenti".

Le colonne della tabelle sono cliccabili e poteranno ad una pagina di gestione del singolo utente / grotto.

Per fare questa pagina ho dovuto modificare il metodo getGrotti dato che prima

ritornava solo i grotti verificati mentre ora io necessitavo anche quelli non ancora controllati.

```
public function getGrotti($validati){  
    try {  
        if($validati){  
            $verificato = 1;  
        }else{  
            $verificato = 0;  
        }  
    ...  
}
```

Problemi riscontrati e soluzioni adottate

Oggi l'unico problema che ho riscontrato era che i metodi getUtenti e getGrotti ritornavano direttamente la query senza farvi un fetch. Nelle pagine fatte in precedenza non vi erano problemi mentre oggi generava il seguente errore:

Uncaught PDOException: You cannot serialize or unserialize PDOStatement instances

Dopo un po di tentativi sono arrivato alla conclusione, anche grazie ad un aiuto di Filippo Finke che mi ha aiutato a isolare la causa del problema, di ritornare nelle due funzioni il fetch della query.

```
return $query->fetchAll();
```

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare la pagina di amministrazione, più precisamente le pagine di modifica / eliminazione di un'utente / grotto.

Diario di lavoro

Luogo	Trevano
Data	18.10.2019

Lavori svolti

Oggi ho iniziato ricontrollando la pagina di amministrazione e mi sono reso conto che era piuttosto scomoda da utilizzare con i bottoni che facevano apparire le tabelle, ho quindi deciso di rimuoverli e lasciare le tabelle visibili. Queste sono limitate a mostrare al massimo 7 righe per pagina così che chi visualizza la pagina non debba scrollare troppo per trovare il record che cerca.

Dopo aver ristrutturato il frontend ho continuato con quello che mi mancava, ho iniziato creando la funzione che richiama la pagina "gestione" che mostra in un form l'utente / grotto da modificare. La pagina è un insieme delle pagine di aggiunta di un grotto e di registrazione di un utente e andrà a richiamare i metodi che verificheranno le modifiche e le renderanno effettive (questi metodi sono ancora da fare). Fatto ciò sono passato a scrivere le funzioni di approvazione di un grotto e di eliminazione di un elemento sia nel controller che nel model.

A seguire prima la parte della funzione di eliminazione situata nel controller che verifica che tipo di campo si vuole eliminare, se questo esiste e, nel caso degli utenti, che ci sia sempre almeno un admin dopo l'eliminazione.

...

```
if($type == 'grotto') {
    $id = filter_var($im->checkInput($id), FILTER_SANITIZE_NUMBER_INT);
    $grotto = (new DBConnection)->getGrotto($id);
    if($grotto != null) {
        (new DBConnection)->delete($type, $id);
    }
} elseif ($type == 'utente') {
    $id = filter_var($im->checkInput($id), FILTER_SANITIZE_EMAIL);
    $utenti = (new DBConnection)->getUsers();
    $utente = (new DBConnection)->getUser($id);
    if($utente != null) {
        foreach($utenti as $item) {
            if($item['nome_ruolo'] == 'admin' && $item['id'] != $utente['id']) {
                (new DBConnection)->delete($type, $id);
            } else{
                array_push($errors, "Deve sempre esserci almeno un admin");
                $_SESSION['errors'] = $errors;
            }
        }
    }
}
...

```

Fatto ciò ho impiegato il tempo restante nella lezione per testare il corretto funzionamento del codice scritto.

Problemi riscontrati e soluzioni adottate

Oggi ho riscontrato un problema durante l'eliminazione di un campo dalla tabella, infatti cercavo di eliminare un utente o un grotto ma in entrambi i casi non mi era

consentito dato che essi erano inseriti nella tabella **voto** come foreign key.
Per risolvere il problema ho dovuto aggiungere al database dove dichiaro le chiavi esterne l'operazione da seguire in caso di eliminazione.

```
create table voto(
    email_utente varchar(50),
    id_grotto int,
    voto float not null,
    primary key(email_utente, id_grotto),
    foreign key(email_utente) references utente(email) on delete cascade,
    foreign key(id_grotto) references grotto(id) on delete cascade
);
```

Come si può vedere sopra la parte in grassetto fa sì che all'eliminazione del campo a cui si fa riferimento vengano eliminati tutti i figli. Quindi in questo caso succederà che:

- Se si elimina un grotto tutti i voti ad esso verranno persi
- Se si elimina un utente tutti i voti effettuati da esse verranno persi

Questo ho dovuto farlo anche nella tabella foto per i grotti.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare la pagina di modifica di un utente / grotto.

Diario di lavoro

Luogo	Trevano
Data	22.10.2019

Lavori svolti

Oggi ho continuato come da programma con la pagina di modifica di un utente / grotto. Più precisamente ho sviluppato le due funzioni che verificano gli inserimenti dell'utente (admin) e gli inserisce nel db se corretti.

Le due funzioni sono piuttosto simili ma hanno alcuni particolari diversi dovuti al fatto che un utente deve essere sottoposto a più controlli di un grotto. Infatti la funzione per gli utenti verifica inizialmente che non sia stata modificata la password, se questo è successo la scrive nel database e poi invia un'email all'utente che ha subito la modifica per informarlo della sua nuova password.

Fatto ciò si verifica se gli altri campi sono diversi da come erano in precedenza, se ci sono differenze esse vengono inserite nel database. Un'ulteriore caratteristica della funzione per utenti è che se si cambia ruolo più precisamente da admin a utente si deve verificare che vi sia almeno un altro admin per rendere effettiva la modifica.

La funzione per grotti invece controlla solamente le differenze tra dati vecchi e nuovi e le scrive nel db. Ha però come caratteristica che se si modifica un campo dell'indirizzo bisogna ricalcolare le coordinate, questo è fatto ogni volta da frontend quando si esegue il submit del form. Per la modifica di un grotto si può anche visualizzare una mappa che mostra dove verrà messo il puntatore.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato grandi problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare la pagina di modifica di un utente / grotto e iniziare a implementare la parte di aggiunta di un utente da parte di un admin.

Diario di lavoro

Luogo	Trevano
Data	24.10.2019

Lavori svolti

Oggi ho continuato come da programma con la pagina di modifica di un utente / grotto, dopo aver aggiunto i validatori frontend e aver corretto dei piccoli bug grafici l'ho terminata ed ho potuto testarla.

Fatto ciò sono passato a sviluppare la pagina di aggiunta di un utente da parte di un admin, la pagina è praticamente uguale a quella di registrazione le uniche differenze sono che non si può inserire una password perché essa verrà creata in maniera random ed inviata via email all'indirizzo inserito nella creazione dell'account.

Fatto ciò ho continuato la documentazione nel capitolo implementazione e facendo ciò mi sono reso conto che mancavano delle parti più o meno importanti nel sito di cui mi ero dimenticato e le ho inserite nel file TODO.md.

Dopodiché ho iniziato a testare le validazioni frontend per essere certo che tutte verifichino tutto correttamente.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato grandi problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio implementare la possibilità di vedere il numero di valutazioni effettuate su di un determinato grotto e la finestra di modifica della password al primo accesso di un utente creato da un admin.

Diario di lavoro

Luogo	Trevano
Data	25.10.2019

Lavori svolti

Oggi come da programma ho implementato il codice che consente di visualizzare il numero di valutazioni eseguite su un grotto. Esso è piuttosto semplice ed è composto da una stampa di un valore ricavato dal database con la seguente query:

```
SELECT count(*) FROM voto where id_grotto=?
```

Fatto ciò ho creato la pagina di cambio password al primo login. Essa viene richiamata dopo il login e verifica che il campo first_login sia impostato a 1, se lo è richiede all'utente di inserire una nuova password e poi lo rimanda ad eseguire nuovamente il login mentre se è a 0 fa loggare l'utente normalmente.

Essa è abbastanza simile alla pagina di reset della password dato che i campi sono gli stessi tranne che non bisogna verificare che l'email sia in uso ed essa non deve poter nemmeno essere modificata.

Se tutto è corretto (le password ripetute sono uguali) viene impostata la nuova password e viene cambiato il campo first_login da 1 a 0.

```
(new DBConnection)->setPassword($email, $password);  
(new DBConnection)->setFirstLogin($email);
```

Dopodiché ho sviluppato in parte l'aggiunta delle immagini. Per capire come fare mi sono basato sull'esempio del seguente sito:

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

La funzionalità è stata aggiunta nella pagina che mostra il grotto selezionato affianco il form che consente di votare.

Una volta inserita l'immagine si effettuano i vari controlli (se è un'immagine,...) e se li supera tutti viene salvata (per evitare problemi con spazi/caratteri speciali,...) con un nome composto dall'hash del contenuto dell'immagine più l'estensione, grazie a questo nome si può anche verificare che l'immagine non esista già.

Problemi riscontrati e soluzioni adottate

Oggi il problema era che non avevo mai utilizzato i files e l'upload di files con PHP quindi ho dovuto ricercare ed informarmi sulle tecniche da utilizzare. Questo lo sono riuscito a fare grazie al sito menzionato in precedenza ed alcuni consigli da parte di compagni sul nome dei files da utilizzare.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare l'aggiunta delle immagini ed occuparmi della documentazione.

Diario di lavoro

Luogo	Trevano
Data	05.11.2019

Lavori svolti

Oggi come da programma ho migliorato e completato il codice che consente di associare delle immagini ad un grotto. Ho dovuto, per farlo funzionare, modificare il percorso in cui salvavo le immagini dato che lo avevo inserito sbagliato e sviluppare la parte di aggiunta al database.

Dopo aver eseguito dei test del funzionamento ho creato la sezione nella pagina admin in cui un utente con i privilegi necessari può eliminare delle immagini. Questa parte è piuttosto simile alle altre tabelle dato che i dati vengono mostrati nello stesso modo. Nella parte frontend l'unica cosa più complicata è la stampa del nome di un grotto in base all'id di esso, questo viene fatto con il seguente codice:

```
echo $_SESSION['grotti_validati'][array_search($row['grotto'], array_column($_SESSION['grotti_validati'], 'id'))]['nome'];
```

La parte seguente serve a cercare quale elemento ha nella colonna id il valore salvato nella variabile \$row['grotto'] nella sessione contenente i grotti validati così da stamparne il nome.

```
array_search($row['grotto'], array_column($_SESSION['grotti_validati'], 'id'))
```

Fatto ciò ho iniziato a scrivere la parte backend dell'eliminazione, per farlo ho modificato le funzioni già usate per eliminare un grotto/utente.

Problemi riscontrati e soluzioni adottate

Oggi il problema che ho riscontrato era che non capivo perché dopo aver eliminato un'immagine non potessi associarla nuovamente al grotto e mi sono reso conto che eliminavo l'immagine solamente dal database e non dalla cartella dove era salvata. Per risolvere il problema mi è bastato aggiungere un costrutto try catch contenente sia l'eliminazione dal db che quella dal filesystem. Per toglierla dal filesystem uso il metodo unlink.

```
try {
    unlink($immagine['path']);
    (new DBConnection)->delete($type, $id);
} catch (Exception $e) {
    ...
}
```

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e mettere a posto dei piccoli errori noti nel sito.

Diario di lavoro

Luogo	Trevano
Data	07.11.2019

Lavori svolti

Oggi come da programma ho continuato la documentazione, più precisamente il capitolo dell'implementazione terminando la spiegazione del database e iniziando quella della pagina principale.

Fatto ciò ho cercato di risolvere alcuni problemi nella pagina in particolare il fatto che le immagini non venivano eliminate dal filesystem quando veniva rimosso un grotto da un admin. Per risolvere il problema ho aggiunto il codice che verifica, prima dell'eliminazione che una località, se essa ha immagini collegate e se vi sono le cancella tramite il metodo unlink(path) di PHP. Per fare funzionare tutto ciò, dato che uso linux, ho però dovuto modificare a livello di filesystem il proprietario della cartella *immagini* cambiandolo con l'utente apposito di apache chiamato www-data e con il gruppo chiamato allo stesso modo.

Quando sono riuscito a risolvere il problema precedente ho verificato parte dei form dato che avevo notato che in alcuni mancavano dei controlli.

Problemi riscontrati e soluzioni adottate

Oggi non ho riscontrato problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e migliorare delle piccole parti del sito.

Diario di lavoro

Luogo	Trevano
Data	08.11.2019

Lavori svolti

Oggi come da programma, nelle prime due ore ho continuato la documentazione, più precisamente il capitolo dell'implementazione completando il sottocapitolo dedicato alla pagina home e scrivendo quelli riguardanti la pagina che mostra un grotto in dettaglio, quella di login, quella di registrazione e la pagina di reset della email.

Documentando la pagina di reset della email mi sono reso conto che era stata creata senza pensare accuratamente alla situazione, essa inizialmente richiedeva l'email dell'account e direttamente la nuova password, fatto ciò inviava un token per email (che salvava anche nel db) e se il token del link, inviato per email, premuto dall'utente corrispondeva con quello nel database cambiava la password.

Dopo un'analisi più attenta ho ristrutturato la pagina facendo sì che inizialmente chieda solo l'email e dopo aver inviato il messaggio con il token, se corrisponde con quello nel database, mostra il form per cambiare password. Inoltre adesso tutta la pagina funziona con i cookies e non con le sessioni così che se si chiude il browser durante le operazioni non si perde tutto.

Problemi riscontrati e soluzioni adottate

Oggi ho avuto parecchie problemi con i cookies dato che non li avevo mai utilizzati, inizialmente li creavo solo per una pagina e sulle altre non funzionavano e ho scoperto che per utilizzarli in modo globale quando li si crea bisogna inserire anche la path di dove devono essere utilizzati, in questo caso '/' per usarli ovunque:

```
setcookie(nome, valore, tempo di durata, '/');
```

Il secondo problema che ho riscontrato era che quando eliminavo un cookie da una pagina esso si eliminava ma se cambiavo pagina si ricreava automaticamente. Dopo diverso tempo perso ho scoperto che era un problema di chrome che lo teneva salvato e lo ricaricava in continuazione. Ho risolto eliminando i cookies dalle impostazioni del browser.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e migliorare delle piccole parti del sito.

Diario di lavoro

Luogo	Trevano
Data	12.11.2019

Lavori svolti

Oggi come da programma, nelle prime due ore ho continuato la documentazione, più precisamente il capitolo dell'implementazione completando il sottocapitolo dedicato alla pagina di amministrazione e di aggiunta di un grotto.

Fatto ciò ho messo a posto i models dato che molte funzioni erano nella classe DBConnection quando avrebbero dovuto essere separate nei model relativi alla tabella. Dopo aver spostato i metodi nel posto corretto ho verificato che tutto funzionasse ancora correttamente.

Infine ho aggiunto la libreria che esegue lo smooth scroll nella pagina di amministrazione, essa si chiama smoothie-js e la documentazione e il codice possono essere trovati al seguente link:

<https://github.com/marlospomin/smoothie>

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e iniziare a creare lo schema UML delle classi.

Diario di lavoro

Luogo	Trevano
Data	14.11.2019

Lavori svolti

Oggi come da programma, nelle prime due ore ho creato i diagrammi UML delle classi per ogni cartella contenente file PHP nel progetto. Ho fatto così perché generandone solamente uno sarebbe risultato uno schema molto complesso. Dopodiché ho continuato la documentazione, più precisamente il capitolo dell'implementazione completando il sottocapitolo che parla dei models inserendo appunto anche lo schema relativo creato in precedenza.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in anticipo sulla pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e cercare di implementare la condivisione di un grotto sui social.

Diario di lavoro

Luogo	Trevano
Data	15.11.2019

Lavori svolti

Oggi ho iniziato la lezione documentandomi su come poter condividere un link o un'immagine di un grotto su Instagram ma ho scoperto leggendo la documentazione ufficiale che questo non è possibile ed è consentito solo ad applicazioni mobile.

Questo è il link della documentazione ufficiale:

<https://developers.facebook.com/docs/instagram-basic-display-api/>

Dopodiché mi sono documentato su come fare per condividere i link su facebook. Su questa piattaforma è possibile ma il sito deve essere su un hosting. Ho provato quindi a caricarlo sul hosting della scuola (infomaniak) ma non consente di caricare cartelle e quindi dovrei caricare un file per volta e ci metterei troppo. Ho quindi provato a connettermi con filezilla ma il proxy di scuola blocca le connessioni e quindi dovrò provare da casa. Inoltre ho scoperto che l'hosting di infomaniak non consente di usare i trigger di MySQL e quindi i voti sui grotti non funzionerebbero.

Fatto ciò mi sono occupato di abbellire la pagina di amministrazione eliminando i bottoni dalle tabelle e sostituendoli con delle icone prese da font awesome.

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

In seguito mi sono occupato di verificare il funzionamento corretto di tutti i forms (click senza niente, click con dei campi mancanti,...) e del corretto caricamento del header in base all'utente utilizzato (se sono admin devo vedere anche la pagina di amministrazione altrimenti no e se non sono loggato devo vedere solo la pagina di login).

Infine ho continuato con la documentazione (capitolo implementazione).

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

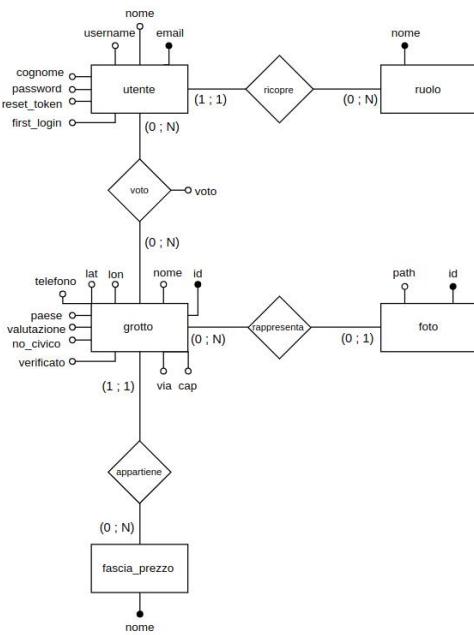
La prossima giornata voglio occuparmi della documentazione e del ER.

Diario di lavoro

Luogo	Trevano
Data	19.11.2019

Lavori svolti

Oggi ho iniziato la lezione creando il diagramma ER conclusivo ed il relativo schema logico. Essi hanno come differenze dai precedenti che contengono più campi che non erano stati previsti all'inizio. Di seguito i due schemi aggiornati.



RUOLO(nome)

UTENTE(email, nome, cognome, username, password, reset_token, first_login*, nome_ruolo(FK))

FASCIA_PREZZO(nome)

GROTTA(id, via, paese, cap, no_civico, lat, lon, nome, valutazione*, telefono, verificato, fascia_prezzo(FK))

FOTO(id, path, id_grotto*(FK)) path unique

VOTO(email_utente(FK), id_grotto(fk), voto)

Fatto ciò ho continuato con la documentazione terminando il capitolo relativo alle classi nella cartella models ed ho iniziato quello sui controllers.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Piattaforma per la scoperta dei Grotti del Canton Ticino

1/2

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e fare un po' di testing.

Diario di lavoro

Luogo	Trevano
Data	21.11.2019

Lavori svolti

Oggi ho iniziato la lezione continuando la documentazione portando quasi a termine il capitolo dei controller. Dopodiché ho iniziato a fare un po di testing e mi sono accorto che era scomodo per un utente non vedere mai con che utente è connesso e che tipi di privilegi ha, per risolvere ho aggiunto nella navbar il nome e cognome dello user ed andandoci sopra con il mouse si vede il tipo di utente (utente o admin). Questa cosa non viene mostrata su dispositivi mobili di dimensioni dei telefoni.

Fatto ciò mi sono accorto che nella pagina di amministrazione non chiedevo conferma prima di eliminare un campo ed ho quindi implementato il codice che lo fa con dei modali.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio occuparmi della documentazione e continuare con il testing.

Diario di lavoro

Luogo	Trevano
Data	22.11.2019

Lavori svolti

Oggi ho iniziato la lezione continuando la documentazione terminando il capitolo dei controller e con esso la sezione dell'implementazione.

Fatto ciò mi sono dedicato a del testing generale del progetto, ho riscontrato tre errori principali che erano i seguenti:

- L'admin non poteva eliminare gli utenti perché otteneva sempre il messaggio d'errore “non puoi eliminare il tuo stesso account”
- Nella mappa le valutazioni non venivano caricate
- Dopo l'aggiunta di un grotto rimaneva, sotto il messaggio di conferma, la mappa per verificare la posizione del ristorante.
- Se si fa una richiesta da remoto senza passare dal controller (es. curl) vengono saltati tutti i controlli perché vengono eseguiti solo nei controllers e non nei models.
- La prima lettera del tipo di utente nella toolbar era minuscola e non bella da vedere.

Per risolvere il primo errore ho impiegato parecchio tempo perché non trovavo dove fosse il problema ma alla fine ho scoperto che c'era, in un if, un controllo scritto al contrario. Era scritto `=!` e non `!=`.

Il secondo bug l'ho risolto con un workaround dal momento che non si riesce ad eseguire le funzioni della libreria auxiliary-rater nelle finestre all'interno della mappa. Ho quindi optato per mostrare il numero e non le stelle quindi se vi sono voti stampa “Valutazione: 3 su 5” mentre se non ve ne sono mostra “Nessun voto”.

Per non mostrare la mappa l'ho solamente spostata all'interno del form dove si inseriscono i dati.

Il quarto bug è ancora da risolvere ma dovrò fare, nei models, delle verifiche con delle regex.

Infine l'ultimo bug trovato l'ho risolto utilizzando il metodo `ucfirst(string)` durante la stampa.

Problemi riscontrati e soluzioni adottate

Oggi l'unico problema che ho avuto era trovare cosa causasse il primo bug della lista sopra citata. Ho impiegato quasi 40 minuti ma l'ho trovato.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio iniziare il capitolo dei test nella documentazione e risolvere il problema 4 della lista sopra.

Diario di lavoro

Luogo	Trevano
Data	26.11.2019

Lavori svolti

Oggi ho iniziato la lezione eseguendo dei test e mi sono reso conto che all'eliminazione di un utente un admin non riceveva alcun messaggio di successo e quindi li ho implementati utilizzando gli Alerts di Material Design Bootstrap.

Fatto ciò ho notato che le medie delle votazioni erano in alcuni casi sbagliate perché non arrotondate allo 0.5 e quindi l'ho implementato modificando il codice come segue.

```
...
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;
...

...
set @avg_review = (select avg(voto) from voto where id_grotto=NEW.id_grotto);
set @avg_round = round(@avg_review*2)/2;
update grotto set valutazione = @avg_round where id=NEW.id_grotto;
...
```

In questo modo ottengo sempre il valore arrotondato allo 0.5 più vicino.

Dopodiché ho risolto alcuni problemi di sessioni che non chiudevo e che causavano alcuni problemi nel caricare i dati semplicemente andando alla fine del loro utilizzo e inserendo il codice: `unset($_SESSION['nome della sessione']);`

Fatto ciò mi sono occupato di trasformare le scritte delle fasce di prezzo in simboli traducendole nel seguente modo:

- Caro --> \$\$\$
- Nella norma --> \$\$
- Buon prezzo --> \$

Per farlo ho utilizzato le API di font awesome come per tutte le altre icone.

Infine, dopo il colloquio con il docente responsabile per fare il punto della situazione, ho continuato la documentazione iniziando a fare i primi test case.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto grossi problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio continuare il capitolo dei test nella documentazione e continuare con il testing.

Diario di lavoro

Luogo	Trevano
Data	28.11.2019

Lavori svolti

Oggi ho iniziato la lezione sviluppando due nuovi test case che verificano il corretto funzionamento della pagina di login, di modifica della password e del primo login. Eseguendo i test scritti ho notato alcuni problemi nella validazione dei campi, avevo dimenticato alcune validazioni e altre non erano fatte con cura. Ho dunque passato il resto della lezione a risolvere i problemi riscontrati e la prossima volta dovrò testare che le modifiche apportate funzionino e non creino problemi ad altre parti del sito.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto grossi problemi tranne nel testare le regex che verificano i campi dal momento che non sono un esperto nel loro utilizzo. Ho dunque dovuto informarmi su internet.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio continuare il capitolo dei test nella documentazione e continuare con il testing.

Diario di lavoro

Luogo	Trevano
Data	29.11.2019

Lavori svolti

Oggi ho iniziato la lezione sviluppando tutti i test case mancanti così da completare il capitolo dei protocolli di test.

Fatto ciò ho fatto alcuni test sui form di aggiunta degli utenti da parte dell'admin ed ho apportato alcune modifiche al contenuto della mail contenente la password che viene inviata all'account appena creato aggiungendo il link alla pagina di login.

Dopodiché ho verificato che i validatori frontend venissero caricati su tutti i form del sito ed ho iniziato a testarne il funzionamento (sia frontend che quelli backend) seguendo i miei protocolli di test appena generati.

Fatto ciò ho fatto un test e mi sono reso conto che eliminando un utente elimino anche i suoi voti ma non aggiorno la tabella se non vi sono più voti. Quindi ho creato un nuovo trigger che, se non vi sono più voti su un grotto, setta a null il campo valutazione della località.

```
delimiter $$$
create trigger avg_after_delete after delete on voto for each row
begin
    set @counter = (select count(*) from voto where id_grotto=OLD.id_grotto);
    IF (@counter = 0) THEN
        update grotto set valutazione=null where id=OLD.id_grotto;
    END IF;
end;
$$$
delimiter ;
```

Infine ho aggiunto a tutti i validatori backend la regola che accetta i caratteri accentati perché mi sono accorto testando la pagina di registrazione che non li accettavo.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto grossi problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

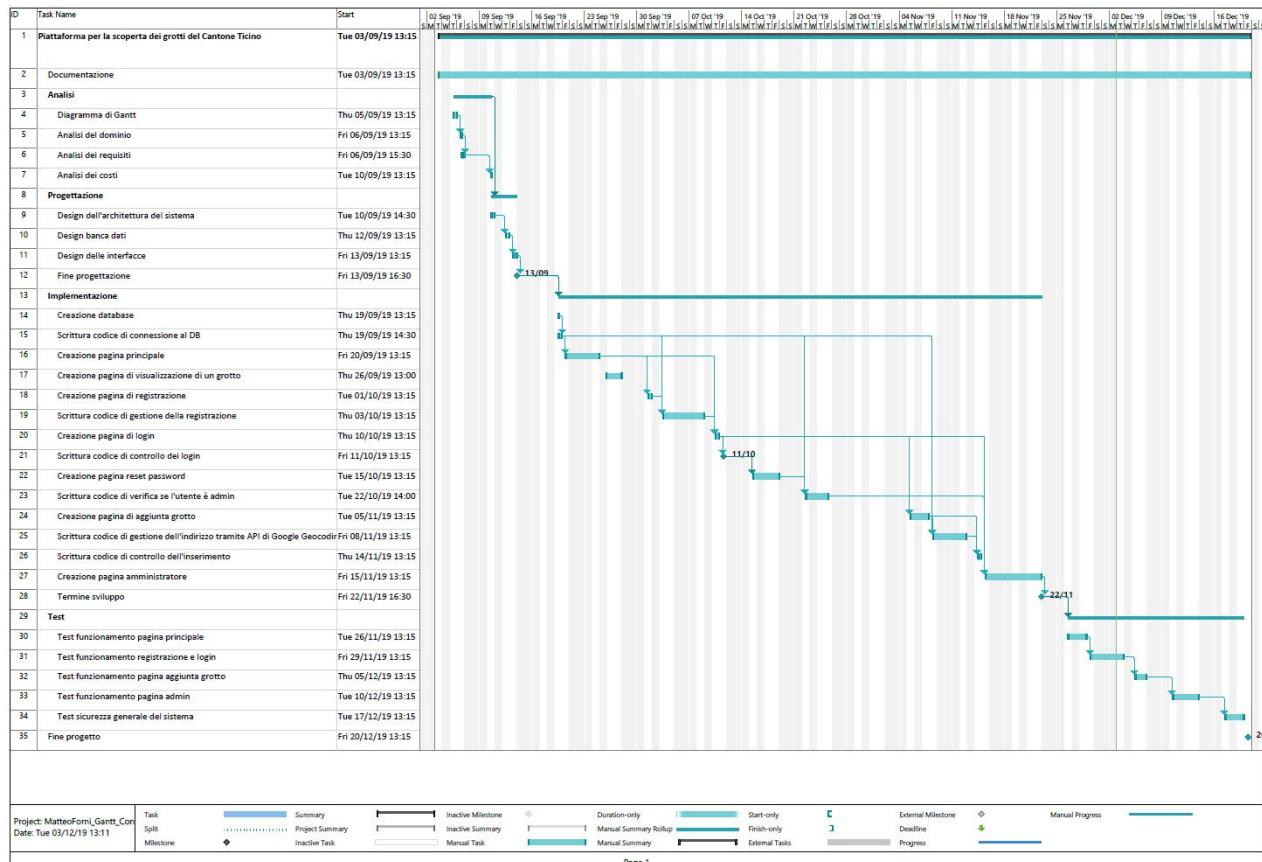
La prossima giornata voglio iniziare il capitolo dei risultati dei test nella documentazione e continuare con il testing.

Diario di lavoro

Luogo	Trevano
Data	03.12.2019

Lavori svolti

Oggi ho iniziato la lezione eseguendo tutti i test spiegati nel capitolo "Protocolli di test" e ho inserito i risultati nella tabella creata nel capitolo "Risultati dei test". Fatto ciò ho creato il gantt consuntivo.



Infine ho dovuto modificare leggermente il database per far sì che non venissero eliminati i voti all'eliminazione di un utente. Questo è stato fatto modificando la tabella voto.

Adesso la tabella è così (evidenziate sono le modifiche):

```
create table voto(
    id int primary key auto_increment,
    email_utente varchar(50),
    id_grotto int,
    voto float not null,
    foreign key(email_utente) references utente(email) on delete set null,
    foreign key(id_grotto) references grotto(id) on delete cascade
);
```

Piattaforma per la scoperta dei Grotti del Canton Ticino

1/2

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto grossi problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio continuare la documentazione e continuare con il testing.

Diario di lavoro

Luogo	Trevano
Data	06.12.2019

Lavori svolti

La lezione del 05.12 non ho lavorato perché siamo stati ospiti della HEP-FR per vedere la scuola e i corsi di bachelor da loro.

Oggi ho completato i capitoli del consuntivo e delle conclusioni della documentazione.

Inoltre ho commentato tutte le classi e i metodi che non erano già stati commentati in precedenza.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio continuare la documentazione scrivendo l'abstract e eseguendo dei test generali di funzionamento e sicurezza.

Diario di lavoro

Luogo	Trevano
Data	10.12.2019

Lavori svolti

Oggi ho scritto l'abstract in italiano (un foglio a parte da inserire prima della documentazione) dopodiché ho riletto la documentazione in cerca di errori e ho fatto un po di testing al sito.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio continuare la documentazione scrivendo l'abstract e eseguendo dei test generali di funzionamento.

Diario di lavoro

Luogo	Trevano
Data	12.12.2019

Lavori svolti

Oggi ho scritto l'abstract in inglese dopodiché ho eseguito un pò di test generali per verificare di non aver dimenticato niente e non avere in giro piccoli errori.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare la documentazione e rileggerla per verificare che non ci siano errori.

Diario di lavoro

Luogo	Trevano
Data	13.12.2019

Lavori svolti

Oggi ho riletto la documentazione correggendo eventuali errori di scrittura trovati.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio stampare tutti i documenti ed iniziare a rilegarli.

Diario di lavoro

Luogo	Trevano
Data	17.12.2019

Lavori svolti

Oggi ho stampato i file da consegnare al termine del progetto ed ho iniziato la rilegatura ma non sono riuscito a finirla dato che c'è una sola rilegatrice e tutti devono usarla.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare la rilegatura.

Diario di lavoro

Luogo	Trevano
Data	19.12.2019

Lavori svolti

Oggi ho terminato la rilegatura ed ho preparato il CD per la consegna.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio consegnare il progetto.

Diario di lavoro

Luogo	Trevano
Data	20.12.2019

Lavori svolti

Oggi ho consegnato il progetto.

Problemi riscontrati e soluzioni adottate

Oggi non ho avuto problemi.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

-

1 INFORMAZIONI GENERALI

Candidato	Nome: Matteo	Cognome: Forni
	matteo.forni@samtrevano.ch	
Luogo di lavoro	Scuola Arti e Mestieri / CPT Trevano-Canobbio	
Orientamento	<input type="checkbox"/> 88601 Sviluppo di applicazioni <input checked="" type="checkbox"/> 88602 Informatica aziendale <input type="checkbox"/> 88603 Tecnica dei sistemi	
Superiore professionale	Nome: Luca	Cognome: Peduzzi
	luca.peduzzi@edu.ti.ch	
Perito 1	Nome:	Cognome:
Perito 2	Nome:	Cognome:
Periodo	3 settembre 2019 – 20 dicembre 2019 (presentazioni: 7-17 gennaio 2020)	
Orario di lavoro	Secondo orario scolastico 1° semestre	
Numero di ore	174	
Pianificazione (in H o %)	Analisi: 10% Implementazione: 50% Test: 10% Documentazione: 30%	

2 PROCEDURA

- Il candidato realizza il lavoro autonomamente sulla base del quaderno dei compiti ricevuto il 1 ° giorno.
- Il quaderno dei compiti è approvato dai periti. È anche presentato, commentato e discusso con il candidato. Con la sua firma, il candidato accetta il lavoro proposto.
- Il candidato ha conoscenza della scheda di valutazione prima di iniziare il lavoro.
- Il candidato è responsabile dei suoi dati.
- In caso di problemi gravi, il candidato o il superiore professionale avvertono immediatamente il perito.
- Il candidato ha la possibilità di chiedere aiuto, ma deve menzionarlo nella documentazione.
- Alla fine del tempo a disposizione per la realizzazione del LPI, il candidato deve inviare via e-mail il progetto al superiore professionale e al perito 1. In parallelo, una copia cartacea della documentazione dovrà essere fornita in duplice copia (superiore professionale e perito). Quest'ultima deve essere in tutto identica alla versione elettronica.

3 TITOLO

Piattaforma per la scoperta dei grotti del Cantone Ticino

4 HARDWARE E SOFTWARE DISPONIBILE

- 1 PC fornito dalla scuola con i tool necessari per lo svolgimento del progetto (Apache, MySql, php, ecc...).
- 1 Accesso presso l'hosting interno messo a disposizione dalla scuola per caricare il progetto.

5 PREREQUISITI

6 DESCRIZIONE DEL PROGETTO

Si tratta di creare tramite applicativo via web e e mail una mappatura geografica dei grotti del Cantone Ticino.

L'applicazione deve poter mostrare dei markers su una mappa corrispondente ai grotti che rispettivamente vengono inseriti dagli utenti. Una volta cliccato sul marker l'utente può andare a visionare nel dettaglio le informazioni riguardanti il grotto selezionato.

Requisiti:

- Bisogna prevedere un amministratore che possa accedere al sito in modo completo, quindi può creare, cancellare, modificare dei campi o altre modifiche direttamente dalla pagina web senza dovere andare in MySql;
- L'amministratore deve potere creare dei nuovi utenti inserendo l'account email come username e una password automatica random, la quale dovrà essere spedita automaticamente all'utente. Al login dovrà esserci un sistema che fa cambiare la password provvisoria. Bisogna anche prevedere la possibilità di richiedere una nuova password in caso di perdita;
- L'amministratore deve potere fare diventare in qualsiasi momento un utente amministratore o utente con solo diritti limitati. Inoltre, dovrà gestire gli accessi e cancellare gli utenti. Deve esserci sempre almeno un amministratore;
- Deve essere creata una pagina di amministrazione nella quale un amministratore o anche un utente possa inserire un nuovo grotto.
- La pagina di creazione del Grotto oltre ai classici campi di inserimento fotografie e campi di testo deve prevedere anche un sistema di votazione (a stelle) e tre fasce di prezzo: Buon mercato – Nella norma – Caro, rappresentabili magari anche con delle icone
- Una pagina di ricerca che permetta di trovare i grotti tramite una mappa interattiva

- Una pagina di ricerca che permetta di trovare i grotti per nome / località / fascia di prezzo
- Una pagina per approvare le i contenuti generati dagli utenti
- Possibilità di condividere il contenuto sui social
- In base al tempo a disposizione, nuovi requisiti possono essere inseriti nel progetto dopo discussione fra formatore e allievo.

7 RISULTATI FINALI

Il candidato è responsabile della consegna al superiore professionale e al perito:

- Una pianificazione iniziale (entro il primo giorno) / progetto di semestre entro la prima settimana.
- Una documentazione del progetto
- Un diario di lavoro
- Implementazione del progetto

8 PUNTI TECNICI SPECIFICI VALUTATI

La griglia di valutazione definisce i criteri generali secondo cui il lavoro del candidato sarà valutato (documentazione, diario, rispetto dei standard, qualità, ...).

Inoltre, il lavoro sarà valutato sui seguenti 7 punti specifici (punti da A14 a A20):

135 – *Documentazione DB, tabelle, ecc...*

237 – *Analisi della sicurezza (Applicazione Web)*

240 – *Sicurezza di base di dati*

148 – *Solidità verifica dei dati, intercettazione degli errori di inserimento*

193 – *Design del GUI*

254 – *Responsive Web Design*

232 – *Programmazione web professionale*

9 FIRMA

Candidato

Canobbio, 03.09.2018

Superiore professionale

Canobbio, 03.09.2018

Perito 1

Perito 2

(luogo e data)

(luogo e data)
