

Gestione Vacanze

Titolo del progetto: Gestione Vacanze
Alunno/a: Mattia Toscanelli
Classe: SAM I4AC
Anno scolastico: 2019/2020
Docente responsabile: Massimo Sartori

1	Introduzione	3
1.1	Informazioni sul progetto	3
1.2	Abstract	3
1.3	Scopo	3
2	Analisi	4
2.1	Analisi del dominio	4
2.2	Analisi e specifica dei requisiti	4
2.3	Use case	7
2.4	Pianificazione	8
2.4.1	Analisi	9
2.4.2	Progettazione	10
2.4.3	Implementazione	11
2.4.4	Testing	12
2.4.5	Documentazione	13
2.5	Analisi dei mezzi	14
2.5.1	Software	14
2.5.2	Hardware	14
2.5.3	Librerie	14
3	Progettazione	15
3.1	Design dell'architettura del sistema	15
3.2	Design dei dati e database	16
3.2.1	Schema E-R	16
3.2.2	Schema logico:	16
3.2.3	Vincoli	16
3.3	Design delle interfacce	17
3.3.1	Pagina di login	17
3.3.2	Pagina di registrazione	17
3.3.3	Pagina richiesta nuova password	18
3.3.4	Pagina reimposta password	18
3.3.5	Pagina principale con il calendario	19
3.3.6	Pagina amministrazione	21
3.3.7	Pagina di stampa	21
3.4	Diagramma delle classi	22
3.4.1	Classe Database	22
3.4.2	Classe SendMail	23
3.4.3	Classe Util	24
3.4.4	Classe Validator	25
4	Implementazione	26
4.1	Struttura MVC	26
4.2	Database	27
4.3	Pagina di login	29
4.4	Pagina di registrazione	31
4.5	Pagina per cambiare password	37
4.6	Pagina principale con il calendario	40
4.6.1	Calendario per Gestori e Visualizzatori	40
4.6.2	Calendario per Docenti	43
4.6.3	Ulteriori funzioni per il calendario	52
4.7	Pagina di amministrazione	54
4.8	Ulteriori funzioni dell'applicazione	61
5	Test	62
5.1	Protocollo di test	62
5.2	Risultati test	64
5.3	Mancanze/limitazioni conosciute	64
6	Consuntivo	65
7	Conclusioni	67
7.1	Sviluppi futuri	67
7.2	Considerazioni personali	67
8	Bibliografia	68
8.1	Sitografia	68
8.2	Indice delle figure	68
9	Allegati	70

1 Introduzione

1.1 Informazioni sul progetto

Titolo:

Allievi: Mattia Toscanelli, impiegato nello svolgimento del progetto.

Classe: I4AC

Docenti: Massimo Sartori

Sezione scuola: Scuola Arti e Mestieri Trevano, Informatica

Data inizio: 03.09.2019

Fine: 20.12.2019

1.2 Abstract

This document contains the documentation for the realization of the software to manage the working hours during the holidays. In this software there are 3 types of users: Viewers, Teachers and Managers. Viewers can only see the calendar and at most print it out. Teachers can interact with the calendar: adding, editing, moving, resizing and deleting lessons. Finally, Managers can access a private page where they can edit or delete users. They can assign working hours to teachers at the same time. Finally I can add or remove lesson days. In this document you can find all the different stages of the project. The first part described is the analysis phase. This phase describes the problem in general, the purpose of this project, the requirements and the means used. The aim of this project is to have software that manages a shared calendar autonomously.

The second phase of the project is design. In this phase the flow diagram, the database used, the various interface designs and finally the UML schema of the most important classes are shown.

The third phase is implementation. In this phase the whole process of how the project is carried out is described. The code used and the various functionalities are described.

The fourth phase is the test phase. In this phase, the pre-established tests of the project are carried out. The tests have been created according to the requirements of the customer.

The fifth and final phase is the conclusion. This phase describes the concepts that were learned during the course of the project. It also describes possible improvements that could be made to make the project even easier.

1.3 Scopo

Lo scopo di questo progetto è quello di realizzare un sito web che permetta ai docenti registrati di inserire delle lezioni al di fuori dell'orario scolastico, cioè durante le vacanze. In pratica esistono delle scuole particolari che procedono per tutto l'anno e che dunque non si fermano durante le vacanze scolastiche. Le vacanze possono essere i sabati, i ponti o i giorni di festa. Ogni docente avrà la capacità di riservare le proprie ore su un calendario ben prestabilito, assegnato ogni inizio di anno scolastico, fino al raggiungimento del proprio monte-ore. C'è anche una pagina di visualizzazione il quale permetterà di vedere il calendario con le varie ore riservate dai docenti senza la capacità di poterla modificare, ma se si vuole si può stampare.

2 Analisi

2.1 Analisi del dominio

Bisogna trovare un metodo che permetta ai docenti di prenotare le ore di lezione durante le vacanze scolastiche. L'obiettivo di questo progetto è dunque quello di creare un applicativo web che permetta di fare ciò senza l'impiego di una persona esterna che gestisca il tutto oppure di un colloquio tra colleghi. Infatti bisognerà che ogni docente deve avere la capacità di entrare nel sito web e senza nessuna difficoltà potersi prenotare le proprie ore. Queste ore verranno prenotate in base ad un calendario condiviso sul quale verranno mostrate le ore già occupate da altri docenti e le altre disponibili. Ci sarà anche un gestore che avrà il potere di gestire quali giorni di vacanza si potrà lavorare e per quante ore. Ogni docente che vuole lavorare durante le vacanze avrà un numero di ore che dovrà eseguire durante l'arco di tutto l'anno e il gestore dovrà avere la capacità di visualizzare un resoconto (cioè quante ore di lavoro mancano per ognuno) in qualsiasi momento. Ogni docente che vorrà usufruire di questo servizio dovrà registrarsi al sito e il gestore dovrà avere la funzione che gli permetterà di accettare o rifiutare la richiesta di registrazione. L'interfaccia della applicazione deve essere molto semplice e intuitiva, in poche parole deve essere comprensibile anche al meno esperto di informatica. Al momento il committente non dispone di un mezzo su cui gestire questo problema e sul mercato non si trova niente che possa risolvere il problema.

2.2 Analisi e specifica dei requisiti

ID: REQ-001	
Nome	Sito web per la gestione delle vacanze dei docenti
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Deve essere facilmente accessibile da tutti.
002	Ogni funzione del sito non deve presentare imprevisti.
003	Il sito deve essere responsive, cioè deve essere adattato a qualsiasi tipo di schermo.

ID: REQ-002	
Nome	Pagina di registrazione
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Tutti possono accedere alla pagina di registrazione.
002	Ogni registrazione dovrà poi essere confermata da un Gestore.
003	I dati da inserire per la registrazione sono i seguenti: Nome, Cognome, Indirizzo e-mail, numero di telefono, password.
004	Ogni utente registrato verrà assegnato a un tipologia di utente: Docente, il quale potrà interagire con il calendario oppure Visualizzatore, il quale potrà appunto solamente visualizzare il calendario.

ID: REQ-003	
Nome	Pagina di login
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Tutti gli utenti registrati posso accedere.
002	Ci deve essere la possibilità di cambiare la password in caso di smarrimento.

ID: REQ-004	
Nome	Gestione degli utenti
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Ci sono tre tipologie di utenti: Gestore, Docente e Visualizzatore
002	Per ogni utente di registrano i seguenti dati: Nome, Cognome, Indirizzo e-mail, numero di telefono, password.
003	La password deve avere almeno 8 caratteri con almeno 1 numero/carattere speciale.
004	Il Gestore è l'utente con più poteri. Esso decide il calendario scolastico, decide chi può registrarsi al sito, decide quante ore lavorative a ciascun docente e vedere una situazione globale di ciascun docente.
005	Il Docente deve poter interagire con il calendario creato dal Gestore, può inserire le proprie ore di lezione nel calendario e stampare il calendario con anche gli altri docenti.

ID: REQ-005	
Nome	Pagina principale con calendario scolastico
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Solo il Gestore e i Docenti possono modificare il calendario scolastico, invece i Visualizzatori potranno solamente stamparlo.
002	Il calendario scolastico deve essere disponibile sia in formato testuale si in formato calendario.
003	Possono esserci solamente due lezioni al giorno.
004	Ogni lezione ha la durata minima di 1 ora con un massimo di 4 ore. Questa deve essere definita dalle 08:00 fino alle 17:00.
005	Non ci possono essere due lezioni nello stesso momento.
006	Ogni lezione aggiunta si può spostare o eliminare.

ID: REQ-006	
Nome	Pagina d'amministratore
Priorità	1
Versione	1.0
Note	-
Sotto requisiti	
001	Questa pagina è accessibile solamente al Gestore.
002	C'è una tabella dove vengono mostrati tutti gli utenti registrati al sito.
003	Ci deve essere la possibilità di aggiungere o eliminare gli utenti.
004	Ci deve essere la possibilità di modificare il calendario.
005	Ci deve visualizzare le ore di lavoro rimanenti dei docenti.

2.3 Use case

Questo è lo Use case (casi di utilizzo) per l'applicazione Gestione Vacanze:

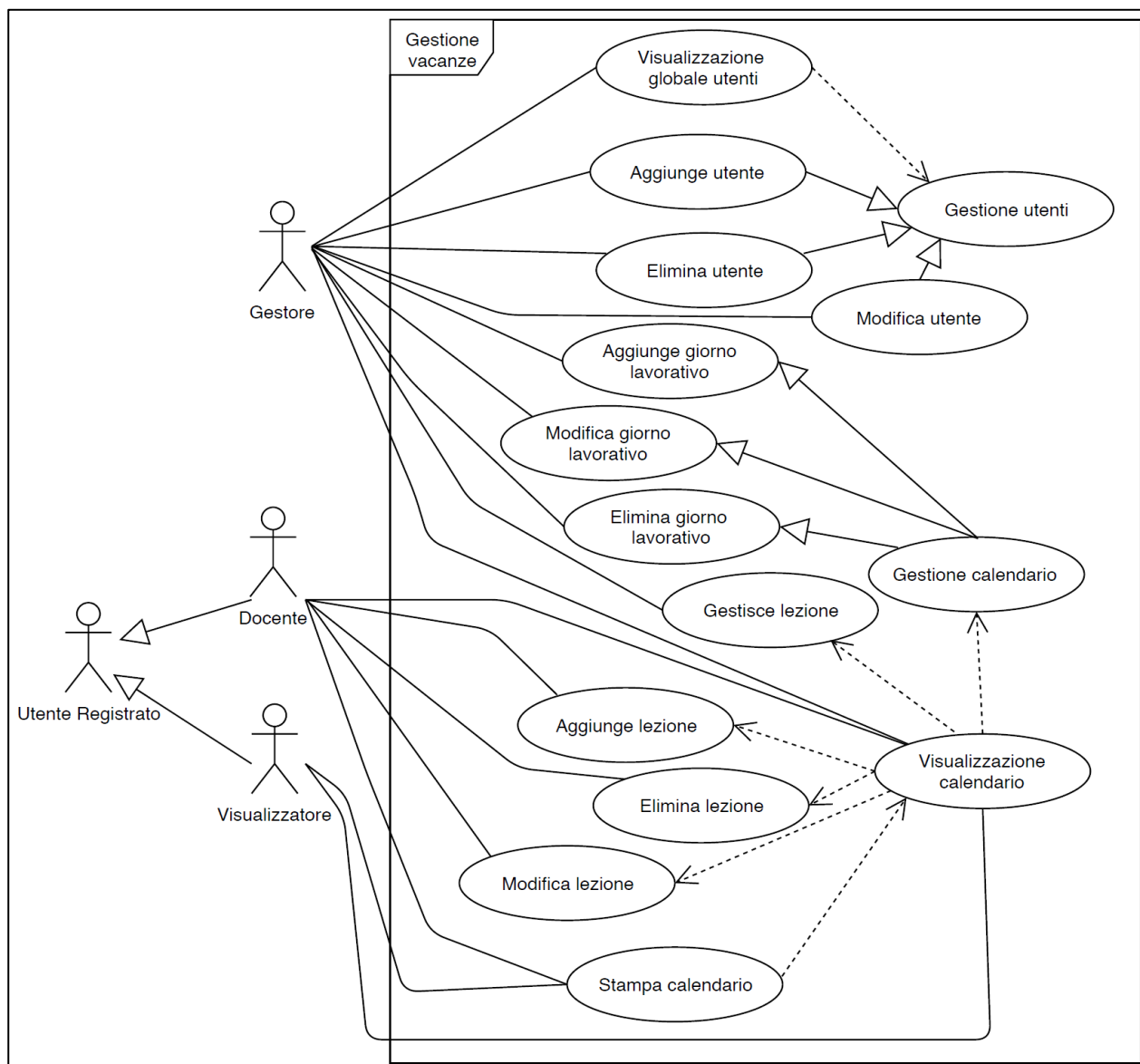


Figura 1 Use Case

Lo schema inizia con tre tipologie di scenari: Gestore (cioè il capo dell'applicazione), Docente e Visualizzatore. Per quanto riguarda i due scenari Docente e Visualizzatore è necessaria una registrazione, la quale verrà valutata dal Gestore se essere accettata oppure no.

Il Gestore avrà la possibilità di svolgere parecchie funzioni di amministrazione, innanzitutto potrà visualizzare una situazione globale di tutti gli utenti registrati, mostrando tutti i dati non sensibili, e in caso fosse un docente, sapere quante ore rimanenti di lavoro deve svolgere. Oltre a ciò avrà la possibilità di aggiungere un utente, modificare per esempio le ore di lavoro di un utente e infine eliminare un utente. Tutte queste attività riguardano l'amministrazione degli utenti. Il Gestore avrà anche la possibilità di gestire il calendario scolastico per le vacanze, potrà aggiungere/modificare/rimuovere un giorno lavorativo.

Il Docente avrà la capacità di visualizzare il calendario prestabilito dal gestore e assegnarsi le ore di lezione che più gli aggradano, quindi potrà aggiungere/modificare/eliminare una lezione. Infine potrà stampare il calendario.

Il Visualizzatore è il tipo di utente con meno poteri, infatti avrà la capacità di visualizzare il calendario completo ed eventualmente stampa.

2.4 Pianificazione

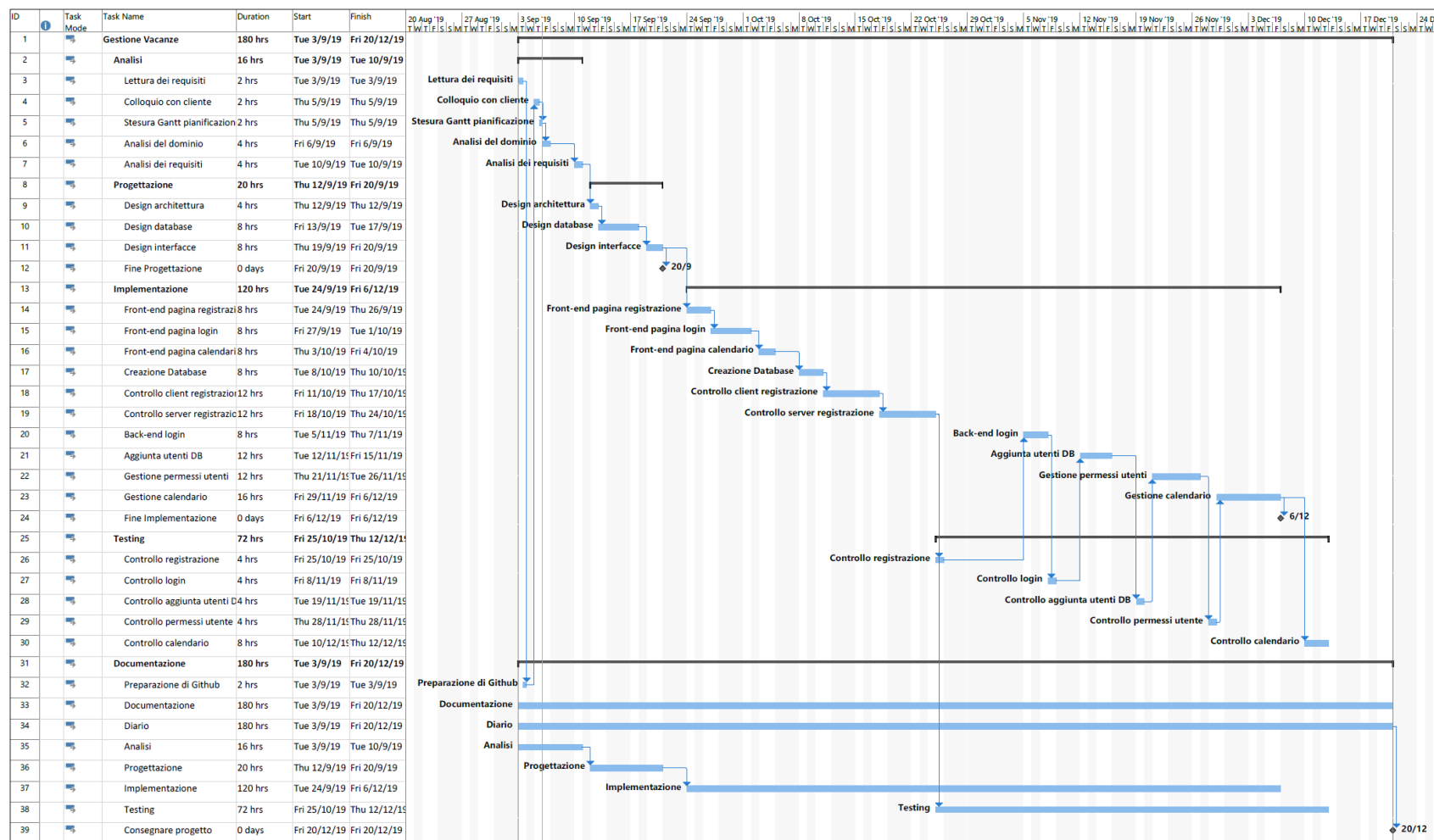


Figura 2 Gantt Preventivo Intero

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

2.4.1 Analisi

Analisi	16 hrs	Tue 3/9/19	Tue 10/9/19
Lettura dei requisiti	2 hrs	Tue 3/9/19	Tue 3/9/19
Colloquio con cliente	2 hrs	Thu 5/9/19	Thu 5/9/19
Stesura Gantt pianificazione	2 hrs	Thu 5/9/19	Thu 5/9/19
Analisi del dominio	4 hrs	Fri 6/9/19	Fri 6/9/19
Analisi dei requisiti	4 hrs	Tue 10/9/19	Tue 10/9/19

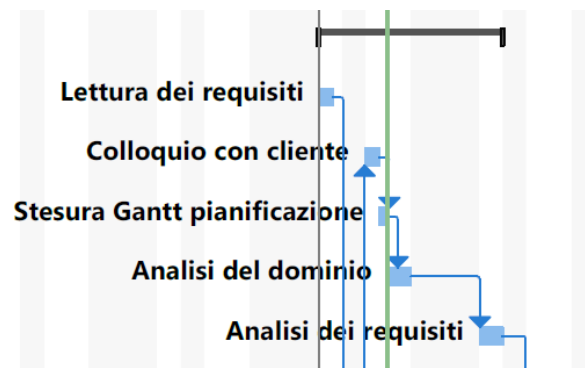


Figura 3 Gantt preventivo - Analisi

L'analisi si suddivide in 5 lavori:

- Lettura dei requisiti → la lettura del diario dei compiti consegnato dal responsabile
- Colloquio con il cliente → la risoluzione dei dubbi con il responsabile per quanto riguarda il diario dei compiti
- Stesura Gantt pianificazione → la progettazione dello schema Gantt
- Analisi del dominio → riflessione se ci sono dei prodotti simili sul mercato e se c'è seriamente bisogno di intraprendere questo progetto
- Analisi dei requisiti → i requisiti che deve avere necessariamente applicazione

Per le attività di analisi ho previsto che sono necessario solamente 4 giorni di lavoro.

2.4.2 Progettazione

Progettazione	20 hrs	Thu 12/9/19	Fri 20/9/19
Design architettura	4 hrs	Thu 12/9/19	Thu 12/9/19
Design database	8 hrs	Fri 13/9/19	Tue 17/9/19
Design interfacce	8 hrs	Thu 19/9/19	Fri 20/9/19
Fine Progettazione	0 days	Fri 20/9/19	Fri 20/9/19

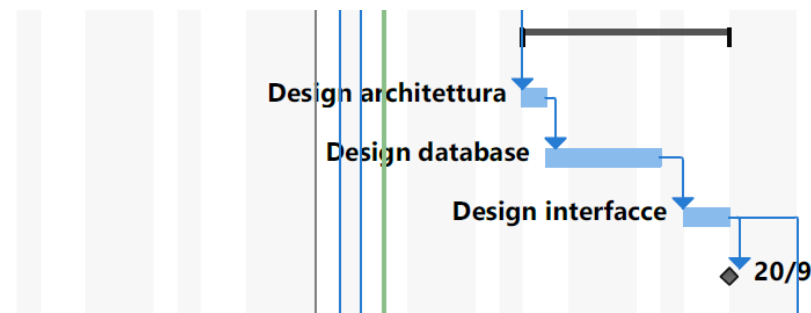


Figura 4 Gantt preventivo - Progettazione

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

- Design architettura → come funziona l'applicazione, come sono collegate le varie operazioni l'una fra l'altra
- Design database → come sono composti i database
- Design interfacce → come si presentano all'utente finale le varie interfacce grafiche
- Fine progettazione → punto conclusivo della progettazione per dare inizio all'implementazione

Ho previsto che per svolgere queste 4 attività mi sono necessari 5 giorni di lavoro.

2.4.3 Implementazione

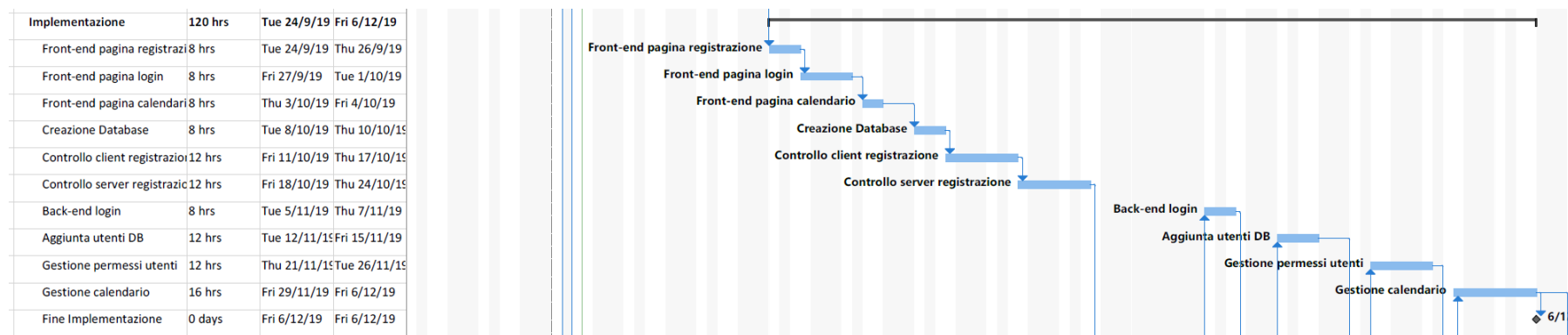


Figura 5 Gantt preventivo - Implementazione

L'implementazione è la fase del progetto che richiede più tempo per essere svolta e si suddivide in 10 attività e un punto cardine:

- Front-end pagina registrazione → interfaccia grafica per la registrazione di un utente
- Front-end pagina login → interfaccia grafica per il login che consente l'accesso al calendario
- Front-end pagina calendario → interfaccia per la visualizzazione del calendario
- Creazione Database → scrittura e utilizzo dei vari database
- Controllo client registrazione → controllo lato client dei dati inseriti nella pagina di registrazione
- Controllo server registrazione → controllo lato server dei dati inseriti nella pagina di registrazione
- Back-end login → controllo e accesso dei dati inseriti in modo protetto e accesso al calendario
- Aggiunta utenti DB → gestione di aggiunta utenti tramite l'applicazione
- Gestione permessi utenti → gestione dei permessi per l'accesso a funzionalità riservate
- Gestione calendario → gestione back-end del calendario
- Fine implementazione → punto conclusivo della implementazione

Per completare la fase di implementazione ho calcolato che ci vogliono 30 giorni lavorativi.

2.4.4 Testing

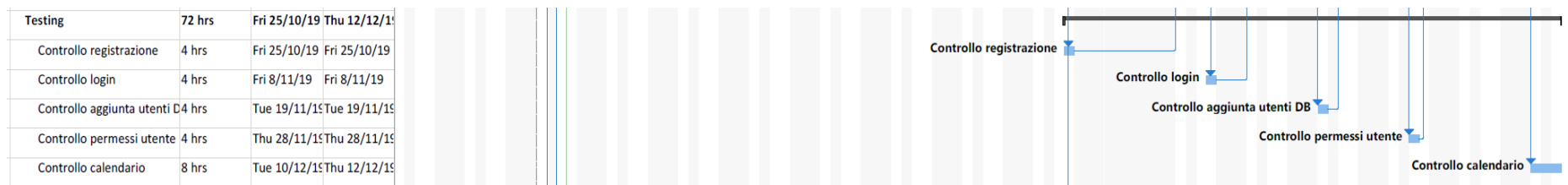


Figura 6 Gantt preventivo - Testing

La fase di testing si suddivide in 5 lavori:

- Controllo registrazione → controllo che tutti i campi della registrazione vengano controllati
- Controllo login → controllo che il login funzioni
- Controllo aggiunta utenti → controllo che ad ogni registrazione gli utenti vengano aggiunti al database
- Controllo permessi utente → controllo che i permessi degli utenti registrati abbiamo i permessi per accedere a determinate funzioni
- Controllo calendario → controllo che le funzionalità del calendario funzionino

Ho previsto che per svolgere queste 5 attività mi sono necessari 12 giorni di lavoro.

2.4.5 Documentazione

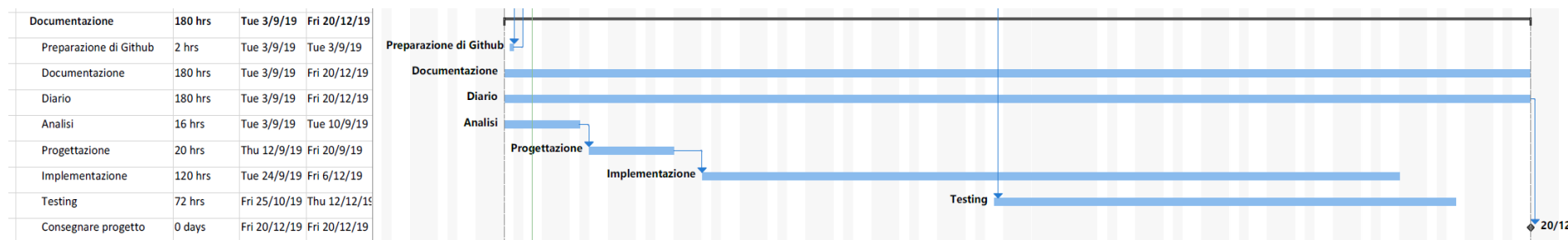


Figura 7 Gantt preventivo - Documentazione

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

- Preparazione di GitHub → preparazione dell'ambiente su GitHub per mostrare l'andamento del progetto
- Documentazione → scrittura della documentazione del progetto
- Diario → scrittura dei lavori effettuati giornalmente
- Analisi → analisi sulle funzioni del progetto
- Progettazione → progettazione delle funzioni del progetto
- Implementazione → implementazione dei lavori che deve svolgere il progetto
- Testing → verifica delle funzioni implementate
- Consegnare progetto → momento di consegna del progetto

2.5 Analisi dei mezzi

Per la realizzazione di questo progetto si ha bisogno di:

- Un PC con performance in grado di girare Windows 10 e un WebServer Apache.

2.5.1 Software

I programmi utilizzati per svolgere questo progetto sono:

- Word → per la documentazione
- PowerPoint → per la presentazione
- Draw.io → per i disegni dei vari schemi e progettazioni
- PhpStorm → per lo sviluppo web
- Apache → per il Web Server
- PHP → per l'utilizzo del linguaggio PHP

2.5.2 Hardware

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

- OS: Windows 10 Pro
- CPU: i7-8550U
- RAM: 8,00GB
- Scheda video: Nvidia MX150

2.5.3 Librerie

- Notify.js
- Fullcalendar.js

3 Progettazione

3.1 Design dell'architettura del sistema

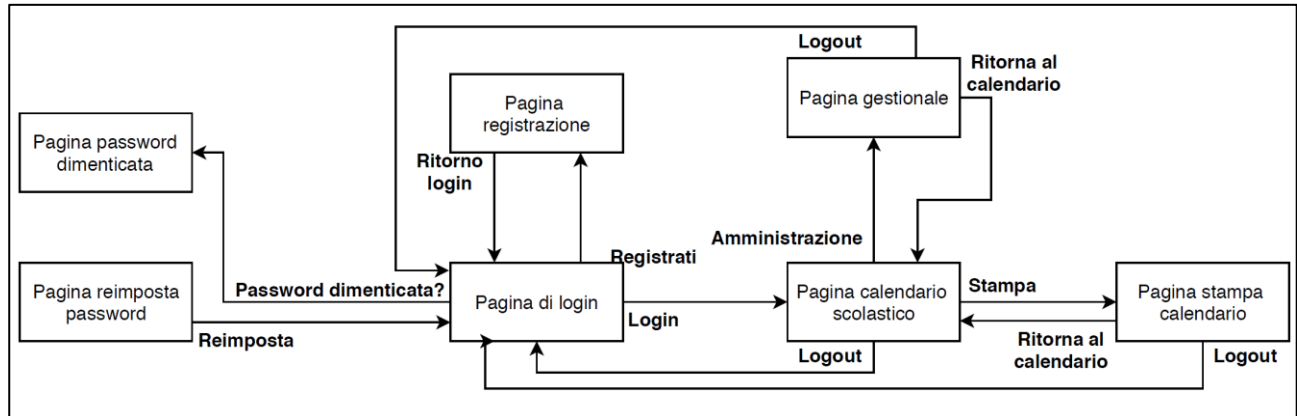


Figura 8 Design dell'architettura del sistema

La pagina di partenza della mia applicazione, cioè la pagina iniziale a cui si collega l'utente finale per utilizzare il mio sito, è la pagina di login. Se l'utente non è ancora registrato alla pagina può premere il pulsante "Registrati" e verrà re-indirizzato alla pagina di registrazione. A questo punto dovrà riempire vari campi con i suoi dati personali e registrarsi. Una volta inviato il modulo di registrazione, il gestore dovrà accettare la richiesta di registrazione e l'account dell'utente verrà attivato. Quando l'account è attivo l'utente potrà effettuare il login e accedere alla pagina con il calendario scolastico. Se l'utente ha dimenticato la password reimpostare la sua password cliccando il pulsante "Password dimenticata?" nella pagina di login e verrà re-indirizzato ad una pagina dove dovrà inserire la sua email per recuperare la sua password. A questo punto gli arriverà una mail la quale conterrà un link per resettare la password che lo porterà sulla pagina per reimpostare la password.

Una volta effettuato l'accesso l'utente verrà mandato alla pagina del calendario. In questa pagina si avranno delle funzionalità in base a chi ha accesso alla pagina (vedi capitolo [2.3 Use case](#)). Sarà però possibile per chiunque visualizzare il calendario e avere la possibilità di stamparlo cliccando il pulsante "Stampa". Cliccando questo pulsante si accederà ad un'altra pagina per avere una visualizzazione di stampa. Dalla pagina di stampa l'utente potrà effettuare un logout premendo appunto il pulsante "Logout" oppure ritornare alla pagina principale premendo il "Ritorna al calendario". Ritornando alla pagina di visualizzazione del calendario, se l'utente che ha effettuato l'accesso è il Gestore esso potrà accedere alla una pagina gestionale attraverso il pulsante "Amministrare". In questa pagina potrà gestire gli utenti (per esempio eliminare un utente) oppure gestire il calendario (per esempio aggiungere un giorno lavorativo). Anche in questo caso l'utente potrà effettuare un logout premendo appunto il pulsante "Logout" oppure ritornare alla pagina dove viene mostrato il calendario premendo il "Ritorna al calendario".

3.2 Design dei dati e database

3.2.1 Schema E-R

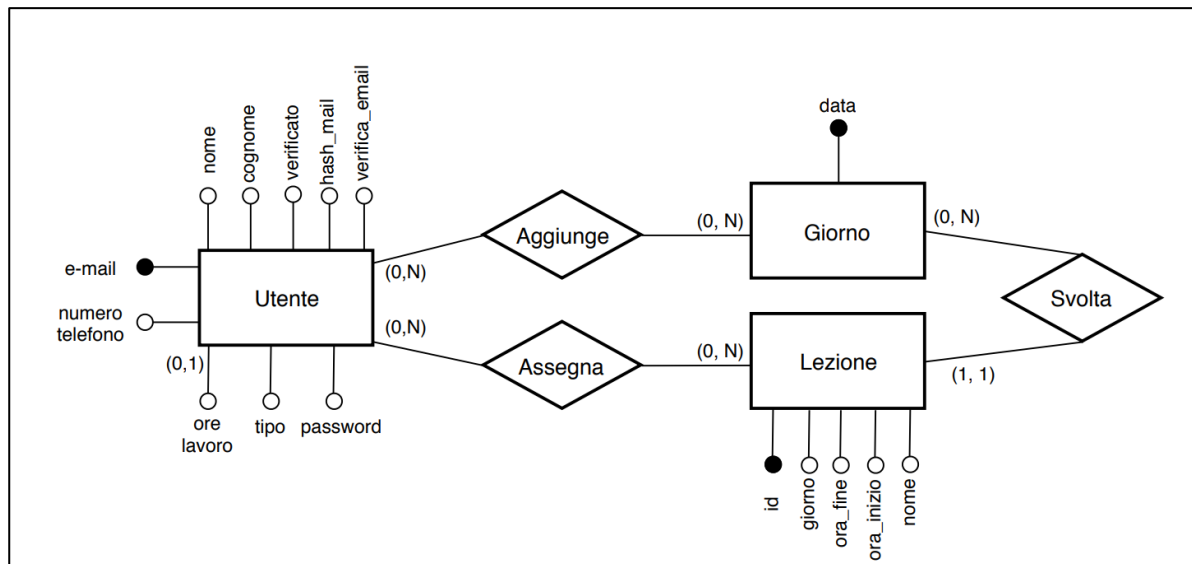


Figura 9 Schema E-R

3.2.2 Schema logico:

Utente(e-mail, nome, cognome, numero_telefono, tipo, password, ore_lavoro*, verificato, hash_mail, verifica_email)

Giorno(data)

Aggiunge(Utente e-mail(FK), Giorno data(FK))

Lezione(id, nome, ora_inizio, ora_fine, Giorno(FK))

Assegna(Utente e-mail(FK), id_lezione(FK))

3.2.3 Vincoli

Per quanto riguarda la tabella Utente ci sono dei vincoli da rispettare:

- Il campo Tipo può contenere solo i valori Visualizzatore, Docente o Gestore.
 - Se Tipo = Visualizzatore, non può partecipare alla relazione Aggiunge né alla relazione Assegna. Inoltre il campo ore_lavoro non deve essere definito.
 - Se Tipo = Docente, va definito il campo ore_lavoro e può partecipare alla relazione Assegna ma non ha quella Aggiunge.
 - Se Tipo = Gestore, può partecipare alla relazione Aggiunge ma non alla relazione Assegna. Inoltre il campo ore_lavoro non deve essere definito.

C'è anche un piccolo vincolo per quanto riguarda la tabella Lezione ed è il seguente:

- Ci possono essere solo 2 lezioni al giorno.
- L'intervallo di tempo fra ora_inizio e ora_fine deve essere minore di 1 ora e maggiore di 4 ore.

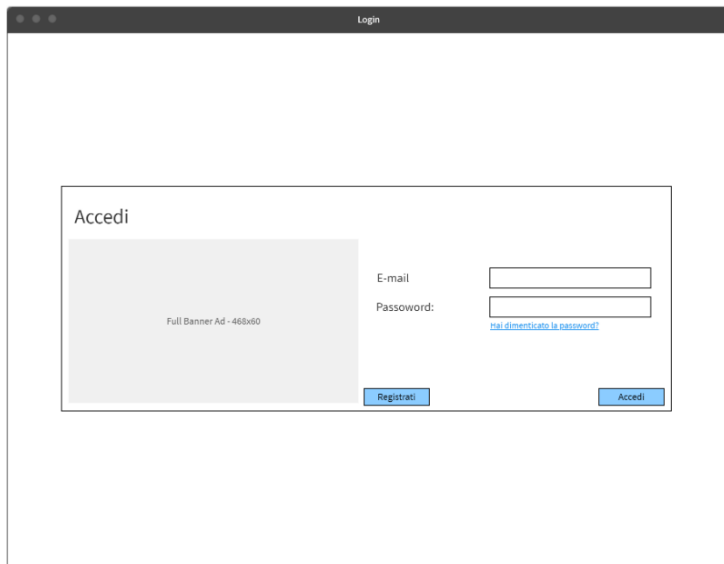
Per quanto riguarda i dati ci sono delle piccole regole da gestire:

- Tutti i campi di testo devono avere almeno 3 lettere.
- Numero di telefono deve avere almeno 10 numeri.
- Password deve avere almeno 8 caratteri tra cui deve essere presente un numero o carattere speciale. (Naturalmente nel database verranno inserite le password cifrate quindi sarà da gestire lato server).

3.3 Design delle interfacce

In questo capitoletto vengono mostrate delle anteprime delle varie interfacce delle pagine.

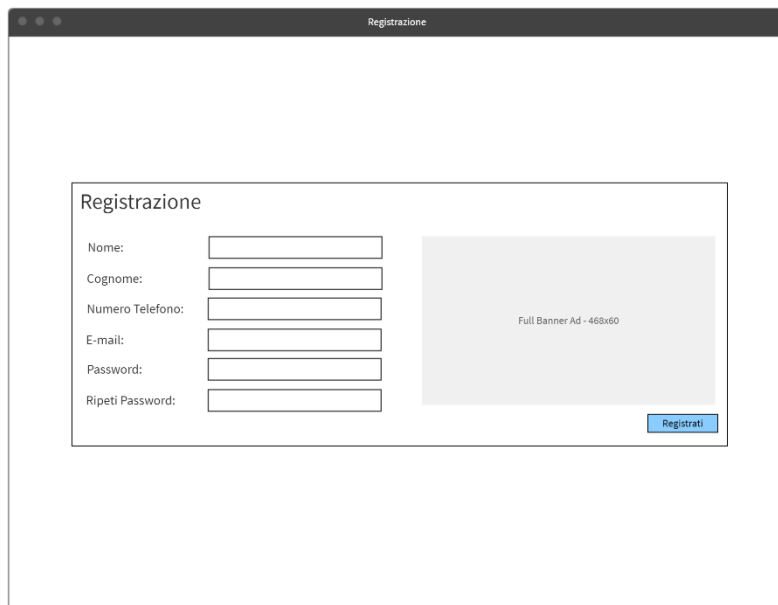
3.3.1 Pagina di login



Questa è la pagina iniziale a cui l'utente finale si collega. In questa pagina si ha la possibilità di registrarsi all'applicativo web premendo il bottone "Registrati", oppure se si avesse già fatto la registrazione, effettuare il login. Inoltre offre una funzione aggiuntiva, cioè in caso l'utente avesse dimenticato la password cliccando il link "Hai dimenticato la password" si potrà avviare una procedura di ripristino password.

Figura 10 Design - Pagina di login

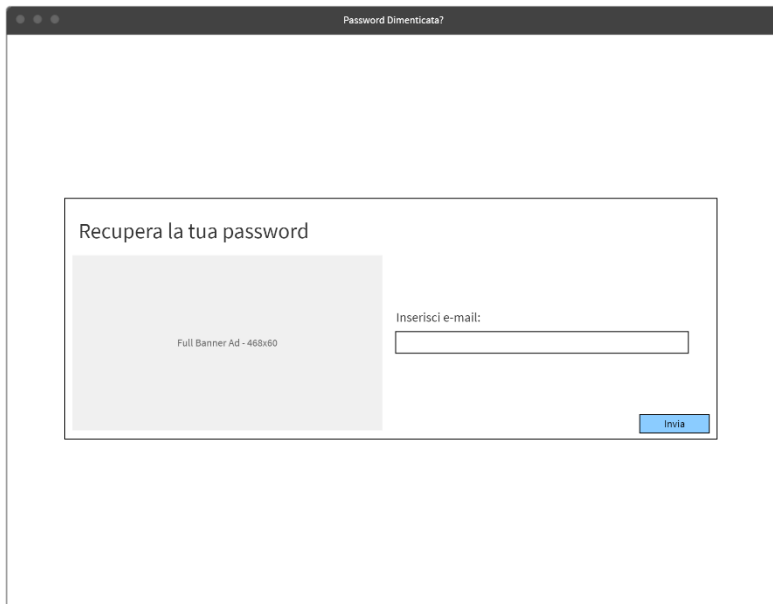
3.3.2 Pagina di registrazione



Questa è la pagina di registrazione. La pagina è composta da un form da 6 campi. L'utente finale dovrà compilare tutti questi campi con i suoi dati personali e in modo corretto. Una volta completata la compilazione gli basterà cliccare il bottone "Registrati" per inviare la richiesta di registrazione al sito.

Figura 11 Design - Pagina di registrazione

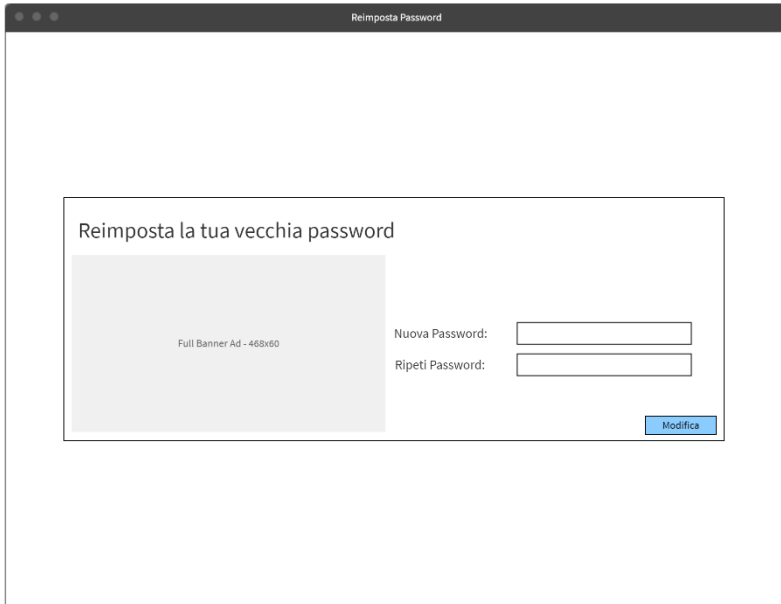
3.3.3 Pagina richiesta nuova password



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

Figura 12 Design - Pagina recupero password

3.3.4 Pagina reimposta password

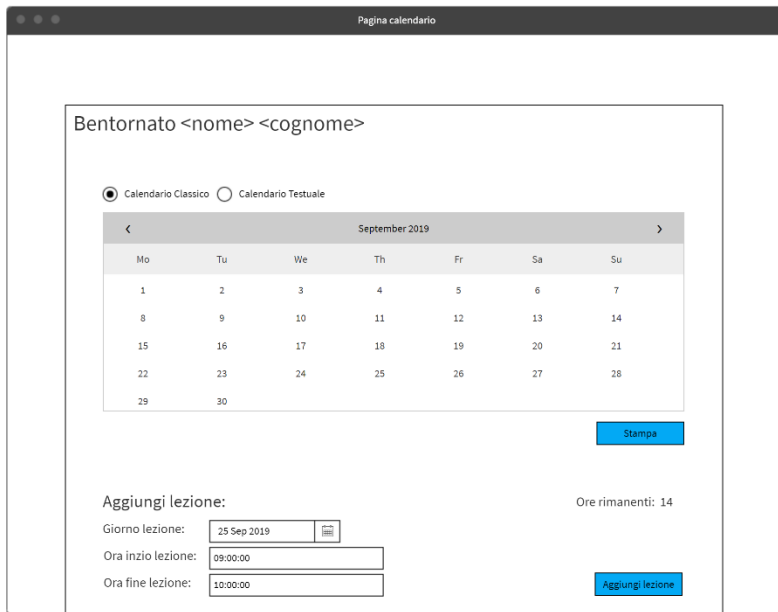


Una volta inviata la richiesta di recupero password, arriverà una mail con il link che riporterà a questa pagina. In questa pagina si dovrà inserire due volte la nuova password che si vuole mettere. Una volta riempiti i due campi basterà cliccare il bottone "Modifica".

Figura 13 Design - Pagina reimposta password

3.3.5 Pagina principale con il calendario

3.3.5.1 Pagina per Docenti



Bentornato <nome> <cognome>

☒ Calendario Classico ☐ Calendario Testuale

September 2019						
Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

[Stampa](#)

Aggiungi lezione:

Giorno lezione:

Ora inizio lezione:

Ora fine lezione:

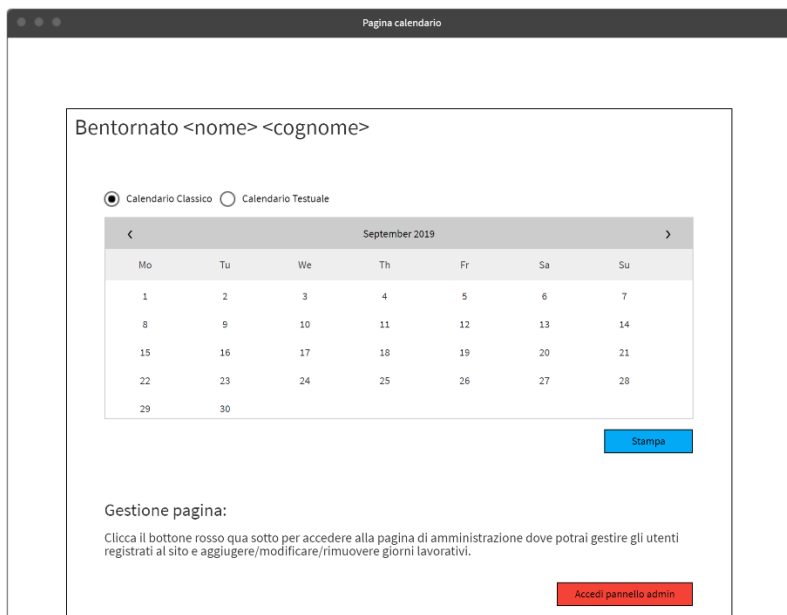
[Aggiungi lezione](#)

Ore rimanenti: 14

Questa è la pagina principale del progetto vista da un account di tipo Docente. In questa pagina si potrà visualizzare il calendario scolastico. Il calendario si può vedere sia in formato classico (come mostrato accanto) si in formato testuale (mostrato in seguito). Essendo che l'account che ha effettuato l'accesso è un docente si avrà la possibilità di aggiungere, modificare o eliminare le lezioni. Infine si ha la possibilità di stampare il calendario tramite il bottone "Stampa".

Figura 14 Design - Pagina calendario Docente

3.3.5.2 Pagina per Gestori



Bentornato <nome> <cognome>

☒ Calendario Classico ☐ Calendario Testuale

September 2019						
Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

[Stampa](#)

Gestione pagina:

Clicca il bottone rosso qua sotto per accedere alla pagina di amministrazione dove potrai gestire gli utenti registrati al sito e aggiungere/modificare/rimuovere giorni lavorativi.

[Accedi pannello admin](#)

Questa è la stessa pagina mostrata sopra ma vista da uno che ha effettuato l'accesso come gestore. Infatti come si può ben notare non c'è più la possibilità di modificare le lezioni ma si aggiunge il bottone "Accedi pannello admin" che porterà ad una pagina di gestione del sito.

Figura 15 Design - Pagina calendario Gestore

3.3.5.3 Pagina per Visualizzatori

Questa è la stessa pagina mostrata sopra ma vista da uno che ha effettuato l'accesso come Visualizzatore. Infatti non ha più nessuna funzione se non quello di visualizzare il calendario o stamparlo.

Figura 16 Design - Pagina calendario Visualizzatore

3.3.5.4 Pagina con calendario in formato testuale

Questa è sempre la stessa pagina con l'accesso eseguito da docente ma con il calendario visto in formato testuale.

Figura 17 Design - Pagina calendario in formato testuale

3.3.6 Pagina amministrazione

Pannello Admin

Gestione utenti: Resetta ore docenti:

#	Nome	Cognome	e-mail	Numero Telefono	Tipo	Ore Rimanenti
#0	Fabio	Bernasconi	fabio@gmail.com	0987654321	Admin	-
#1	Nicola	Verdi	nicola@gmail.com	2043274687	Docente	12
#2	Roberto	Rossi	roberto@gmail.com	0723642974	Docente	23
#3	Elia	Bianchi	elia@gmail.com	0766789213	Visualizzatore	-

Nome: Numero Telefono:
 Cognome: Tipo:
 Email: Ore rimanenti:

Gestione lezioni:

Gestione lezioni:

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Giorno lezione:

Questa è la pagina di amministrazione la quale possono accedere solo i Gestori. In questa pagina si hanno due funzioni principali. La prima funzione è quella della gestione degli utenti. Viene mostrata una tabella di tutti gli utenti registrati al sito e si ha la possibilità di aggiungere, modificare o eliminare un utente. Inoltre si ha la possibilità anche di resettare le ore per tutti i docenti registrati.

Scorrendo giù per la pagina si può trovare la seconda funzione, cioè la gestione dei giorni lavorativi. Qui si potrà aggiungere, modificare o eliminare i giorni in cui è possibile fare lezione.

Figura 18 Design - Pagina pannello admin

3.3.7 Pagina di stampa

Pagina Stampa

Stampa

Tipologia: ☒ Personale ☐ Completo Formato: ☒ Normale ☐ Testuale

Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Questa è l'ultima pagina che offre il mio sito, cioè la pagina dove si può effettuare la stampa del calendario. Sopra ci sono due impostazioni "Tipologia" e "Formato". La tipologia viene mostrata solamente hai docenti e permette di stampare il calendario completo oppure solamente con le proprie ore di lezione. Invece il formato può essere modificato da chiunque e permette di stampare il calendario come mostrato in figura o in formato testuale.

Figura 19 Design - Pagina di stampa

3.4 Diagramma delle classi

3.4.1 Classe Database

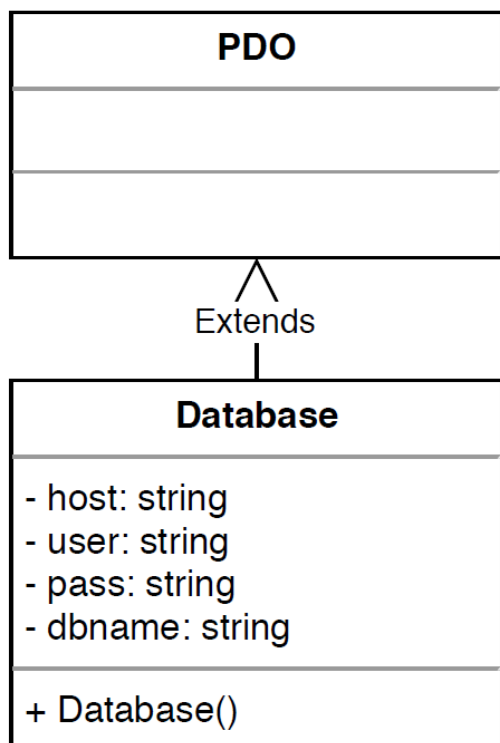


Figura 20 UML - Classe Database

Questa classe è utilizzata per eseguire l'accesso con il database tramite PHP. Essa presenta quattro attributi privati e un metodo costruttore senza parametri.

Gli attributi sono:

- host → definisce l'host dove si trova il database, nel mio caso "localhost".
- user → definisce il nome utente per eseguire l'accesso al servizio MySQL.
- pass → definisce la password corrispondente all'utente che esegue l'accesso.
- dbname → definisce il nome del database dove in seguito si vorranno eseguire le operazioni.

Il metodo costruttore si preoccupa solamente di stabilire la connessione con il Database attraverso il metodo padre della classe PDO. Infatti la mia classe Database estende la classe PDO di PHP. In caso ci dovesse essere un problema di connessione con il database MySQL viene scatenato una Exception che porterà ad una pagina di errore. Grazie a questa classe Database si potranno eseguire delle interrogazioni in qualsiasi parte dell'applicazione.

3.4.2 Classe SendMail

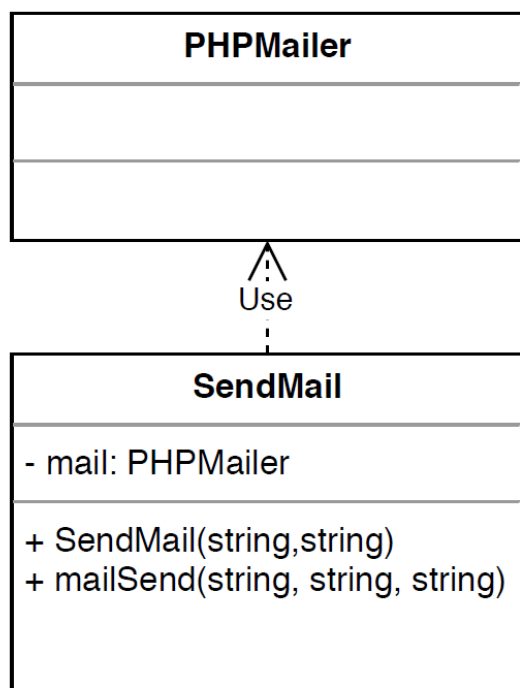


Figura 21 UML - Classe SendMail

Questa classe è utilizzata per inviare le email tramite PHP. Essa presenta un attributo privato, un costruttore con 2 parametri e un metodo di utilizzo della classe.

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

Ritornando alla classe SendMail nel costruttore definisco tutti i vari parametri per il collegamento al client di posta elettronica, come per esempio l'host, la porta, le credenziali d'accesso,... . Il costruttore presenta due parametri obbligatori: il primo parametro è la stringa per email d'accesso e il secondo parametro è la stringa per la password.

Infine c'è l'attributo mailSend che mi permette di inviare una mail. Ha tre parametri: il primo mi definisce la email a chi devo inviare un messaggio, il secondo mi definisce l'oggetto della email e il terzo mi definisce il contenuto della mail. Questa classe verrà utilizzata per inviare link di conferma o di recupero password.

3.4.3 Classe Util

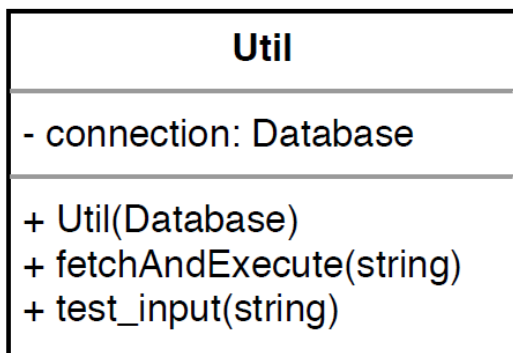


Figura 22 UML - Classe Util

Questa classe è utilizzata per facilitarmi le operazioni. Più precisamente per eseguirmi le interrogazioni al database in modo sicuro (con i prepare statement) e per eseguire dei controlli rapidi degli input passati dagli utenti attraverso le pagine.

Partendo con ordine la classe ha un attributo privato, un costruttore e due metodi di utilizzo della classe.

L'attributo connection mi definisce la connessione al mio database ed esso viene passato attraverso il metodo costruttore della classe.

In seguito la classe presenta due metodi:

- `fetchAndExecute(string)` → mi permette attraverso il di passarli una query da fare sul database. Esso si occupa dunque di preparare la query in modo da evitare attacchi interni ed eseguirla al database. Inoltre in caso di una SELECT ritorna un array multidimensionale con il risultato della query.
- `test_input(string)` → questo metodo verrà utilizzato praticamente da tutti i controller per evitare che l'utente immetta dati, attraverso form o altro, malevoli con lo scopo di danneggiare il sistema. In pratica questo metodo rimuove caratteri speciali, spazi e slash dalle stringhe passate come parametro.

3.4.4 Classe Validator

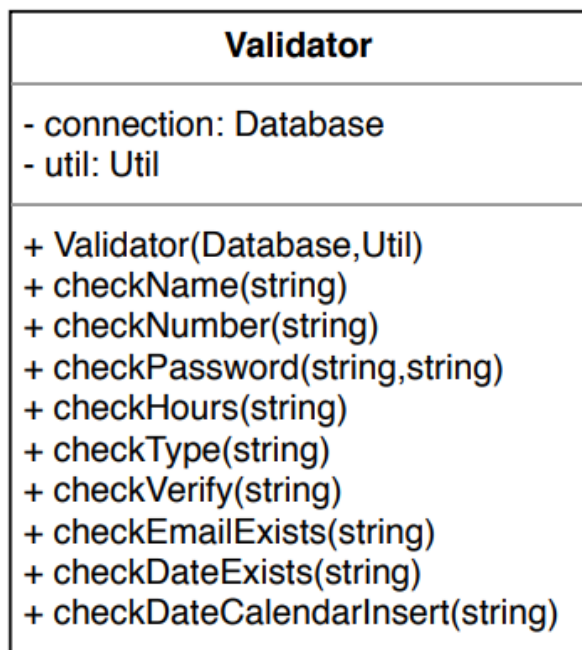


Figura 23 UML - Classe Validator

Questa classe è utilizzata per controllare gli input di registrazione o di modifica di dati. La classe ha due attributi privati, un costruttore e nove metodi di utilizzo della classe.

Gli attributi della classe sono:

- connection → definisce la connessione al mio database ed esso viene passato attraverso il metodo costruttore della classe.
- util → mi permette di effettuare delle interrogazioni al database in modo sicuro e anch'esso viene passato attraverso il metodo costruttore della classe.

In seguito la classe presenta i nove metodi di controllo:

- checkName(string) → Metodo per verificare i nomi, minimo 3 lettere, massimo 50 lettere e niente caratteri speciali.
- checkNumber(string) → Metodo per verificare il numero di telefono, solo numeri e "+", massimo 14 caratteri.
- checkPassword(string,string) → Metodo per verificare se le password rispettano i criteri di sicurezza, minimo 8 caratteri, una maiuscola e un carattere speciale. Inoltre viene confrontato se le due password inserite come parametro sono uguali.
- checkHours(string) → Metodo per verificare le ore di lavoro dei Docenti, solo numeri maggiori di 0 e inferiori di 999.
- checkType(string) → Metodo per verificare che il tipo assegnato all'utente sia valido (quindi che sia "Gestore", "Docente" o "Visualizzatore").
- checkVerify(string) → Metodo per verificare che la verifica dell'utente sia valida (quindi che sia 0 o 1).
- checkEmailExists(string) → Metodo per verificare che la email inserita nella registrazione non sia già stata utilizzata da un altro utente.
- checkDateExists(string) → Metodo per verificare che la data inserita nel pannello admin per i giorni lavorativi non sia già stata aggiunta.
- checkDateCalendarInsert(string) → Metodo per verificare che la data inserita nel pannello admin per i giorni lavorativi sia maggiore della data attuale.

Questa classe viene creata una medesima copia anche in JavaScript per effettuare un controllo lato Client e appesantire in modo più ridotto il server.

4 Implementazione

La progettazione di questo progetto è stata quasi del tutto rispettata. Infatti le uniche modifiche svolte sono l'eliminazione della pagina di stampa (sostituito con un bottone) e la possibilità per i docenti di interagire direttamente dal calendario per aggiungere/modificare/eliminare le proprie lezioni.

4.1 Struttura MVC

Il progetto è iniziato con la creazione della struttura MVC del progetto. MVC (Main-View-Model) è un modello molto utilizzato al giorno d'oggi ed è molto utile per avere una struttura ordinata del codice. Questo modello si divide in 3 categorie:

- **Model:** fornisce tutti i metodi per acceder ai dati utili dell'applicazione.
- **View:** visualizza i dati contenuti nel model e si occupa dell'integrazione con utenti e agenti.
- **Controller:** riceve i comandi dell'utente e in base a ciò modifica lo stato degli altri due componenti.

La struttura MVC del mio progetto è la seguente:

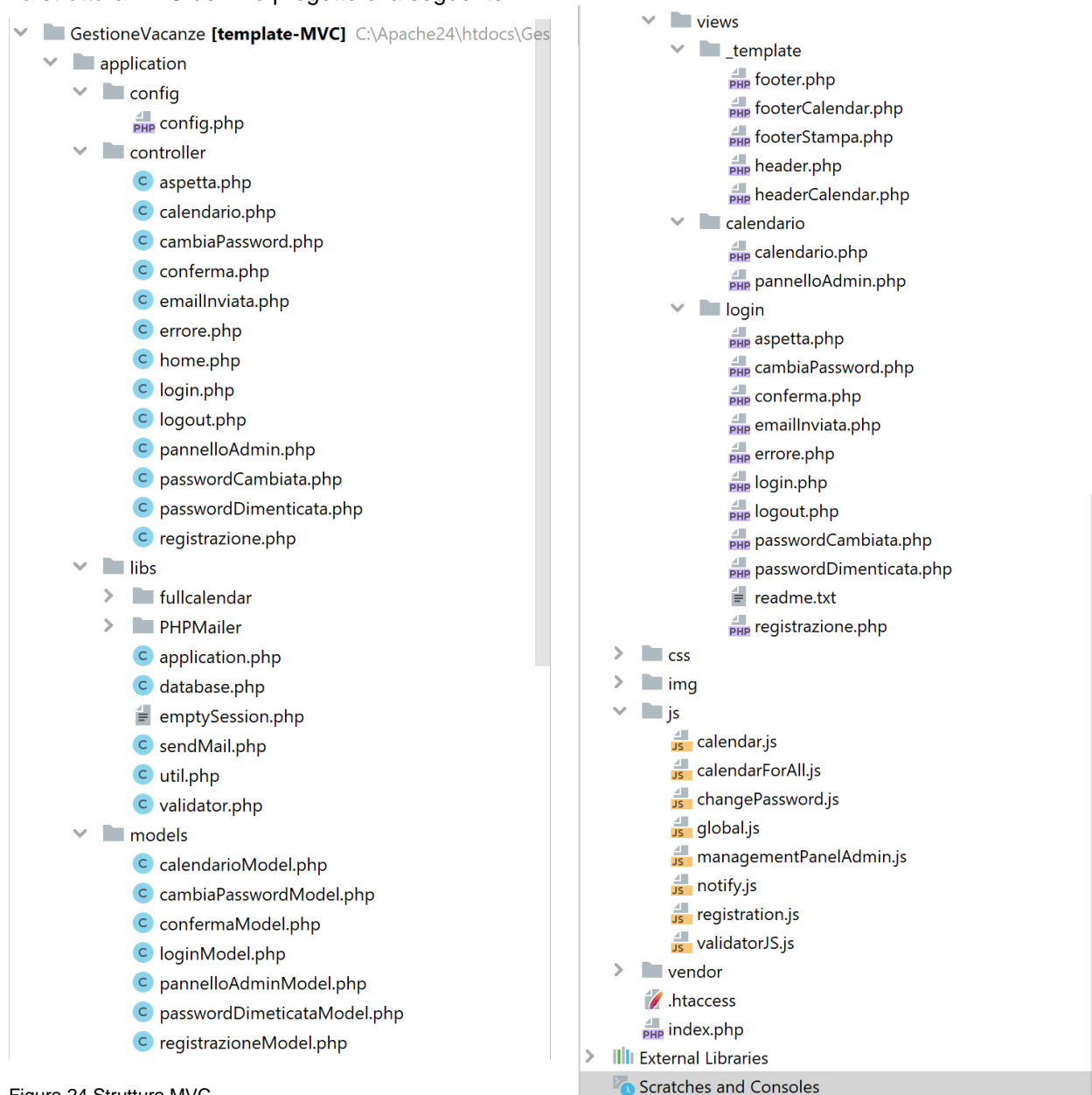


Figura 24 Struttura MVC

In totale ho 13 controller che corrispondono a loro volta a 13 View. Inoltre io ho un'ulteriore cartella chiamata "libs" dove contengo le classi descritte nel capitolo [3.4 Diagramma delle classi](#) che mi permettono di facilitarmi le operazioni di controllo e di utilizzo del database.

4.2 Database

Prima di incominciare a descrivere il codice delle pagine dell'applicazione voglio spiegare il mio database. Il mio database è formato da 5 tabelle: utente, lezione, aggiunge, assegna e giorno e lo script per la sua creazione è il seguente:

```
create database gestione_vacanze;
use gestione_vacanze;

drop table if exists utente;
create table utente(
    email varchar(150) primary key,
    nome varchar(40) not null,
    cognome varchar(40) not null,
    numero_telefono varchar(15) not null,
    ore_lavoro float(4,1),
    tipo varchar(14) not null,
    verificato tinyint(1) not null,
    password varchar(100) not null,
    verifica_email tinyint(1) not null,
    hash_mail varchar(32) not null
);

drop table if exists giorno;
create table giorno(
    giorno date primary key
);

drop table if exists lezione;
create table lezione(
    id integer AUTO_INCREMENT primary key,
    nome varchar(40) not null,
    ora_inizio time not null,
    ora_fine time not null,
    giorno date,
    foreign key (giorno) references giorno(giorno)
);

drop table if exists aggiunge;
create table aggiunge(
    email varchar(150),
    giorno date,
    foreign key (email) references utente(email),
    foreign key (giorno) references giorno(giorno),
    primary key(email,giorno)
);

drop table if exists assegna;
create table assegna(
    email varchar(150),
    id_lezione integer,
    foreign key (email) references utente(email),
    foreign key (id_lezione) references lezione(id),
    primary key (email,id_lezione)
);
```

Il nome del database è “gestione_vacanze” e la prima tabella “utente” contiene gli attributi:

- email → L'email dell'utente.
- nome → Il nome dell'utente.
- cognome → Il cognome dell'utente.
- numero_telefono → Il numero di telefono dell'utente.
- tipo → Il tipo dell'utente. Si differenzia in “Visualizzatore”, “Docente” e “Gestore”.
- ore_lavoro → Le ore di lavoro dell'utente. Questo campo è diverso da 0 solamente se il tipo è “Docente”
- verificato → Esprime se il Gestore ha accettato la registrazione dell'utente, quindi può essere solo “0” se non accettata dal Gestore o “1” se la registrazione è stata accettata.
- password → La password dell'utente, però per motivi di sicurezza viene salvata la hash della password.
- verifica_email → Esprime se l'utente ha verificato la email tramite il link di conferma, anche in questo caso il valore può essere “0” se non verificata o “1” se verificata.
- hash_mail → In questo campo viene salvato un numero random che ha sua volta viene codificato per identificare l'utente quando deve verificare la email o cambiare la password. (Questo argomento verrà spiegato meglio in seguito).

La tabella “giorno” contiene un unico attributo di nome “giorno” che contiene appunto il giorno lavorativo che il Gestore dell'applicazione ha messo a disposizione per poter lavorare. Quindi i Docenti potranno a loro volta inserire le proprie lezioni in questi determinati spazi di tempo.

La tabella “lezioni” contiene i seguenti attributi:

- id → L'identificatore della lezione. Viene utilizzato principalmente per la gestione delle lezioni nel calendario.
- nome → Il nome/titolo della lezione.
- ora_inizio → L'ora di inizio della lezione.
- ora_fine → L'ora di fine della lezione.
- giorno → Il giorno in cui si svolge la lezione. Questo giorno viene preso dalla tabella “giorno”, quindi un Gestore deve prima aggiungere una data lavorativa prima di poterci inserire lezioni.

Infine ci sono le tabelle “aggiunge” e “assegna”. Entrambe sono tabelle ponti, la tabella “aggiunge” collega “utente” con “giorno”, invece la tabella “assegna” collega “utente” con “lezione”.

4.3 Pagina di login

La prima pagina a cui si può collegare un qualsiasi utente è la pagina di login. L'interfaccia grafica di questa pagina è la seguente:

Figura 25 Pagina di login

Si tratta di una normalissima pagina di login dove l'utente registrato inserisce la propria email e la propria password per accedere. Cliccando il bottone "Login" verrà appunto effettuato il controllo di accesso, perciò viene richiamato il controller "Login.php" che riceverà tramite il metodo POST le credenziali di accesso.

```
$email = Util::test_input($_POST[POST_EMAIL]);
$password = $_POST[POST_PASSWORD];
require_once 'application/models/loginModel.php';
$loginModel = new LoginModel();
$result = $loginModel->access($email,$password);
if ($result == 0) {
    $_SESSION[SESSION_ERR] = "Email o password non corretti!";
    header("Location:" . URL . "login");
} else if($result == 1){
    $result = $loginModel->getUser($email);
    $_SESSION[SESSION_TYPE] = $result[0][DB_USER_TYPE];
    $_SESSION[SESSION_NAME] = $result[0][DB_USER_NAME];
    $_SESSION[SESSION_SURNAME] = $result[0][DB_USER_SURNAME];
    $_SESSION[SESSION_EMAIL] = $email;
    header("Location:" . URL . "calendario");
}
```

[...]

Come si può ben notare il controller legge la variabile `$_POST` per ricavare i dati del form di login e crea un oggetto di tipo `LoginModel`. A questo punto richiama il metodo `access($email,$password)` di `LoginModel` e gli passa come parametri email e password. Questo metodo `access()` ritorna "0" se le credenziali inserite sono sbagliate, "1" se le credenziali inserite sono corrette e "2" se le credenziali son corrette ma l'account non è ancora stato verificato da un Gestore.

Se le credenziali di accesso sono corrette e l'account è già stato verificato da un Gestore, verrà eseguita una query per ricavare tutti i dati utili dell'utente che sta effettuando l'accesso grazie al metodo `getUser($email)` di `LoginModel`. In pratica questo metodo è sufficiente inserire la email dell'utente come parametro ed esegue una query nella tabella "Utente" ed esso ritorna il risultato di quest'ultima. Dunque dopo aver richiamato questo metodo "riempio" le variabili di sessione che mi saranno utili per dopo login e richiamo la pagina principale dell'applicazione, cioè la pagina di calendario (questa verrà spiegata nei capitoli successivi).

In caso di credenziali sbagliate viene riempita la variabile di sessione per mostrare l'errore in modo grafico all'utente che sta effettuando l'accesso. Per mostrare qualsiasi errore verificato lato server, utilizzo questa tecnica in HTML per tutte le pagine:

```
<?php if(isset($_SESSION[SESSION_ERR])): ?>
    <script>$.notify('<?php echo $_SESSION[SESSION_ERR] ?>', 'error');</script>
<?php endif; ?>
```

In pratica verifico se la variabile di sessione che contiene l'errore è stata impostata. Se così fosse, viene stampato l'errore con una notifica grazie alla libreria non scritta da me di nome "*notify.js*".

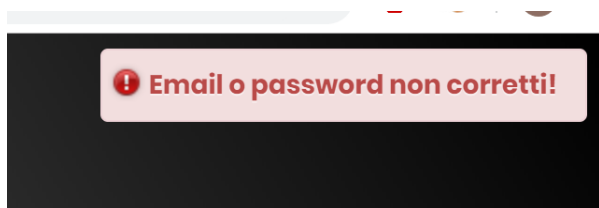


Figura 26 Notifica di errore

Per far visualizzare la notifica utilizzo questa funzione di "*notify.js*" scritta in JavaScript:

```
$.notify(<messaggio>, <tipo_notifica>);
```

Questa funzione come primo parametro richiede il messaggio da visualizzare, nel caso dell'immagine soprastante ho inserito "Email o password non corretti!"; come secondo parametro bisogna esprimere il tipo della notifica. Questo tipo si suddivide in 4 categorie: "success" per mostrare una notifica che ha avuto esito positivo, quindi viene visualizzata in verde, "info" per mostrare un'informazione ed è di colore blu, "warn" per mostrare un messaggio di warning e viene visualizzata in giallo ed infine "error" per mostrare una notifica di errore e viene visualizzata in rosso.

Infine se le credenziali di accesso fossero giuste, ma il Gestore non ha ancora confermato la registrazione, l'utente viene re-indirizzato in una pagina dove viene spiegato il motivo di perché non è possibile effettuare l'accesso.

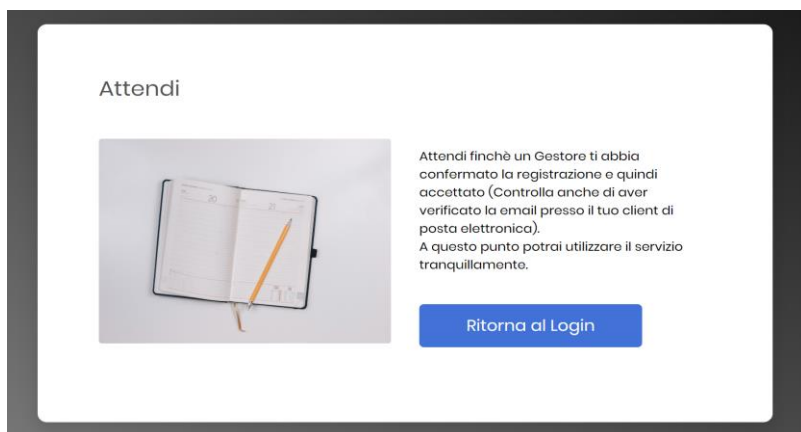


Figura 27 Pagina attendi

Il metodo per verificare il numero di telefono è il seguente:

```
function checkNumber(val) {
    val = val.replace(/ /g, "");
    val = val.replace(/-/g, "");
    var regexNumber = /^[+]?[0-9-#]{10,14}$/;
    return regexNumber.test(val);
}
```

Questo metodo controlla che il numero di telefono passato come parametro abbia dai 10 ai 14 numeri. In più è permesso iniziare il numero con il "+" ed è permesso l'utilizzo del carattere "#".

Il metodo per verificare la email è il seguente:

```
function checkEmail(val) {
    var regexEmail = /^([<>()\\[\]\\. ,;: \s@"]+)(\.[^<>()\\[\]\\. ,;: \s@"]+)*|(".*")@((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\]|([a-zA-Z-0-9]+\.)+[a-zA-Z]{2,}))$/;
    return regexEmail.test(val);
}
```

Questo metodo controlla che la e-mail passata come parametro sia valida. La e-mail deve essere del formato [testo@testo.testo](#). Inoltre per la e-mail viene fatto anche un ulteriore metodo, infatti bisogna controllare che la e-mail inserita dall'utente non ci sia già nel database perciò eseguo questo metodo:

```
function checkEmailDuplicate(val) {
    var email = val;
    var response;
    return ($.ajax({
        url: (URL+"Registrazione/emailExist"),
        type: "POST",
        dataType: "json",
        data: {email: email},
        success: function (text) {
            response = text;
        }
    }));
}
```

Questo metodo fa una richiesta ajax al controller PHP "emailExist" che ha sua volta richiama il metodo getUserA(\$email) nel model "LoginModel":

```
public function getUserA($email){
    $selectAccess = "select * from utente where email='$email'";
    $result = $this->util->fetchAndExecute($selectAccess);
    if(count($result)>0){
        echo 1;
    }else{
        echo 0;
    }
}
```

A questo punto il metodo del getUserA(\$email) fa una query al database e cerca tutti gli utenti che hanno la e-mail uguale a quella inserita dell'utente nella registrazione. Se la query trova almeno un risultato il metodo ritorna un "1" se invece non trova neanche un risultato ritorna "0". Quindi questo valore viene letto dal metodo in JavaScript per poi agire di conseguenza con l'utente.

Infine come ultimo metodo di verifica è quello delle password:

```
function checkPassword(val1, val2) {
    var regexLetter = /[a-zA-Z]/;
    var regexDigit = /\d/;
    var regexSpecial = /^[^a-zA-Z\d]/;
    return (regexDigit.test(val1) || regexSpecial.test(val1)) &&
        regexLetter.test(val1) && val1.length >= 8 &&
        val1 == val2;
}
```

Questo metodo controlla che le due password passate come parametro siano identiche e che abbiano almeno 8 caratteri. Inoltre in queste password deve essere presente un carattere speciale e/o un carattere speciale.

In caso ci fosse un errore in un input viene colorato di rosso il bordo dell'input e viene mostrata anche la notifica di errore:

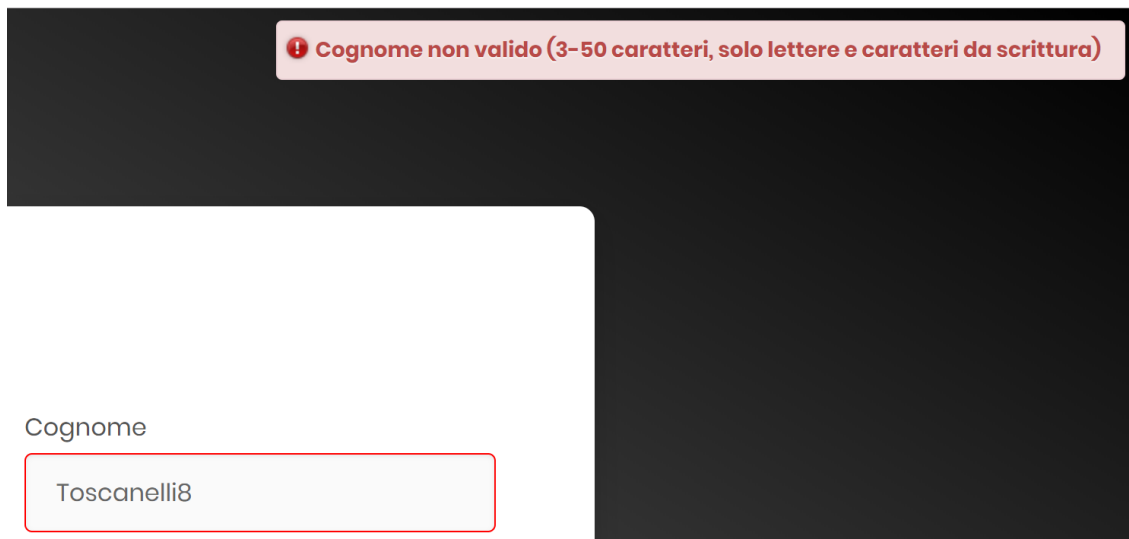


Figura 29 Messaggio di errore pagina di registrazione

Invece se tutti i dati inseriti riesco a “sorpassare” il controllo lato client viene richiamato il metodo insert() del controller “Registrazione”:

```
public function insert(){
    require_once 'application/models/registrazioneModel.php';
    $name = $surname = $number = $email = $password1 = $password2 = "";
    if($_SERVER["REQUEST_METHOD"] == "POST"){
        $name = Util::test_input($_POST[POST_FIRST_NAME]);
        $surname = Util::test_input($_POST[POST_LAST_NAME]);
        $number = Util::test_input($_POST[POST_PHONE_NUMBER]);
        $email = Util::test_input($_POST[POST_EMAIL]);
        $password1 = Util::test_input($_POST[POST_PASSWORD]);
        $password2 = Util::test_input($_POST[POST_RE_PASSWORD]);
        $rm = new RegistrazioneModel($name, $surname, $number, $email, $password1, $password2);
        if($rm->insertUser()){
            header("Location: ".URL."aspetta");
        }else{
            header("Location: ".URL."errore");
        }
    }else{
        header("Location: ".URL."errore");
    }
}
```

In pratica questo metodo si occupa di controllare inizialmente che la richiesta del metodo è di tipo “POST”, se non fosse così l'utente viene mandato nella pagina di errore:

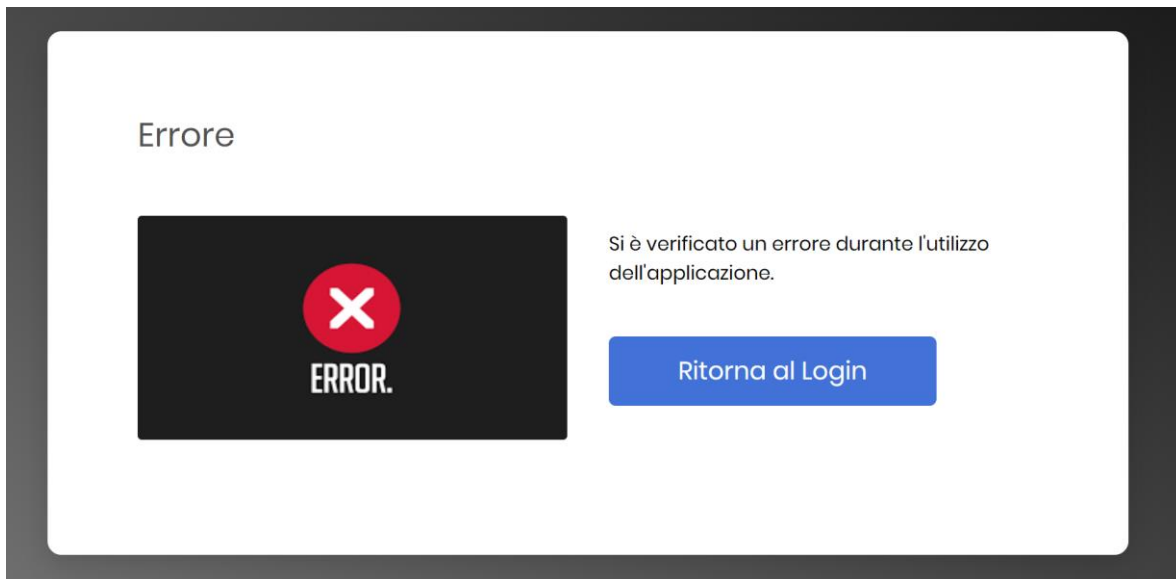


Figura 30 Pagina di errore

Se invece il metodo era stato richiamato con il metodo “POST”, i dati passati vengono inizialmente “puliti” dal metodo `test_input($var)` della classe “Util” (trasforma i caratteri speciali per non creare problemi) per poi essere passati come input al costruttore di “RegistrazioneModel”. In seguito viene richiamato il metodo `insertUser()` sempre del model “RegistrazioneModel”:

```
function insertUser(){
    if($this->checkAll()) {
        $hash = md5(rand(1000,5000));
        require 'application/libs/sendMail.php';
        $body = "Benvenuto/a $this->name $this->
surname,<br> la rigranzio per essersi iscritto al sito web per la gestione delle lezioni durante
le vacanze scolastiche!<br><br><a href='".URL."conferma/confirm/$hash/$this-
>email'> Per verificare la sua mail clicchi questo link!</a>";
        try{
            $s = new SendMail("<email_gmail>", "<password_gmail>");
            $s->mailSend($this->email, "Verifica la tua email", $body);
            $password = password_hash($this->password1, PASSWORD_DEFAULT);
            $insertUser = "INSERT INTO utente (email,nome,cognome,numero_telefono,ore_lavoro,tipo
,verificato,password,verifica_email,hash_mail)
VALUES ('$this->email','$this->name','$this->surname','$this-
>number',0,'Visualizzatore',0,'$password',0,'$hash')";
            $this->util->fetchAndExecute($insertUser);
        } catch (Exception $e){
            header("Location: ".URL."errore");
            exit;
        }
        return true;
    }else{
        return false;
    }
}
```

In poche parole inizialmente viene richiamato metodo `checkAll()` che effettuerà nuovamente i controlli degli input eseguiti in JavaScript ma questa volta lato Server, più precisamente utilizzerà la classe “Validator” descritta nel capitolo [3.4.4 Classe Validator](#). Anche in questo caso se i controlli non fossero andati a buon fine l'utente viene mandato nella pagina di errore, altrimenti viene preparata una mail di conferma della

registrazione all'utente. In questa mail sarà presente un codice criptato che verrà aggiunto al link di conferma per renderlo univoco. Questo codice criptato verrà salvato anch'esso nel database assieme ai dati dell'utente. A questo punto viene richiamata la classe "SendMail" alla quale bisogna passare le credenziali di accesso del client di posta elettronica. Una volta fatto ciò è sufficiente richiamare il metodo mailSend e bisognerà passare come parametro l'e-mail dell'utente, l'oggetto e il contenuto (con il link di verifica) della mail. Se la e-mail viene inviata senza problemi l'utente verrà aggiunto al database.

La email che arriva all'utente avrà un'aspetto simile:



Figura 31 Verifica e-mail

Per poter accedere all'applicazione bisognerà verificare la e-mail ed essere accettati dal Gestore. Quando l'utente cliccherà sul link contenuto nella e-mail verrà richiamato il metodo confirm(\$hash,\$email) del controller "Conferma":

```
public function confirm($hash = null,$email= null){
    if(($email != null) && ($hash != null)) {
        require_once 'application/models/confermaModel.php';
        $conferma_model = new ConfermaModel();
        if($conferma_model->confirm(Util::test_input($hash), Util::test_input($email))){
            header("Location: " . URL . "conferma");
        }else{
            header("Location: " . URL . "errore");
        }
    }else{
        header("Location: " . URL . "errore");
    }
}
```

Questo metodo riceverà come input la hash, cioè il numero generato precedentemente nella registrazione che in seguito è stato aggiunto al link della mail, e la email (in questo caso dell'utente che si è registrato). In caso che almeno uno di questi due parametri non siano stati passati, l'utente verrà mandato alla pagina di errore, altrimenti viene richiamato il metodo confirm(\$hash,\$email) del model "ConfermaModel". Questo di occuperà di verificare che la e-mail corrisponda ad un utente del database ed in seguito di verificare che la hash corrisponda a quel determinato utente. Se così non fosse la funzione ritorna "false" e l'utente verrà mandato nella pagina di errore. Se invece i due parametri fossero corretti viene resettato il codice hash dell'utente così da non rendere più valido il link di conferma della mail.

Una volta verificata la e-mail l'utente verrà trasportato su questa pagina dove gli verrà detto che per poter incominciare ad utilizzare l'applicazione dovrà aspettare la conferma del Gestore:

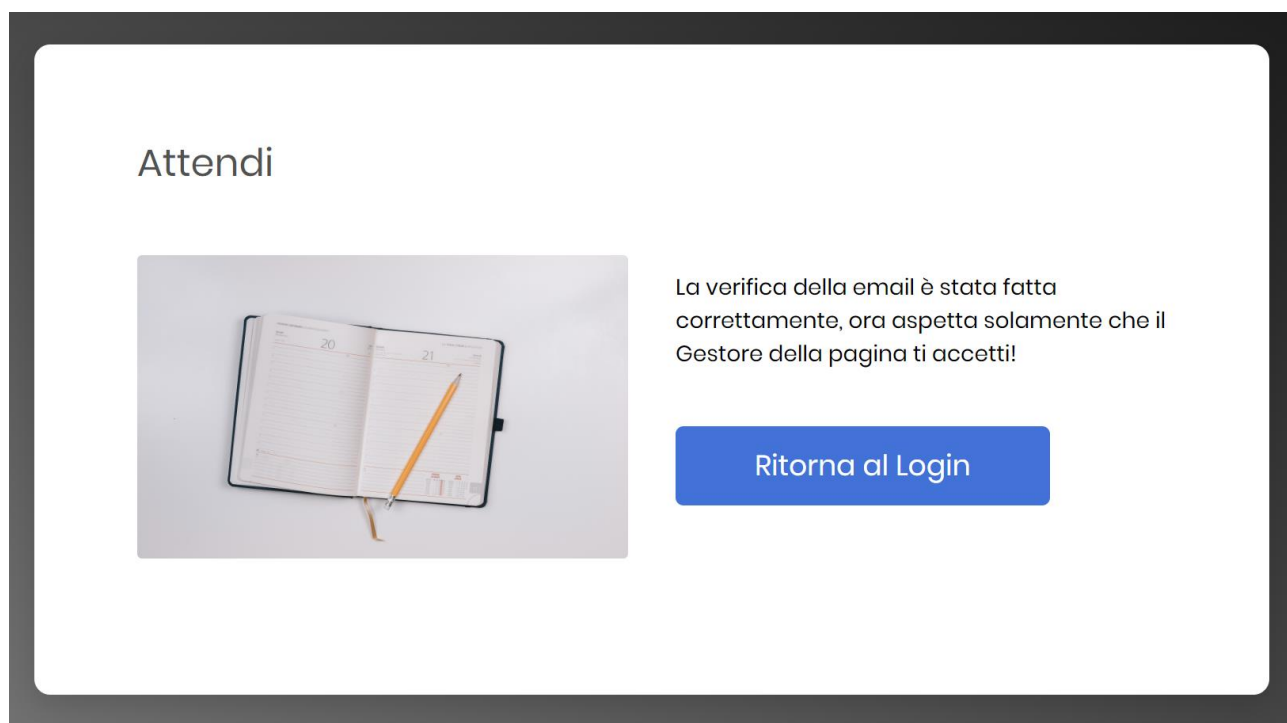


Figura 32 Pagina attendi

A questo punto l'utente può solo che attendere.

4.5 Pagina per cambiare password

Ritornando per l'ultima volta alla pagina di login, sotto al campo password è presente la voce "Hai dimenticato la password?". Se essa viene cliccata iniziata la procedura di ripristino della password e l'utente viene indirizzato in questa pagina:

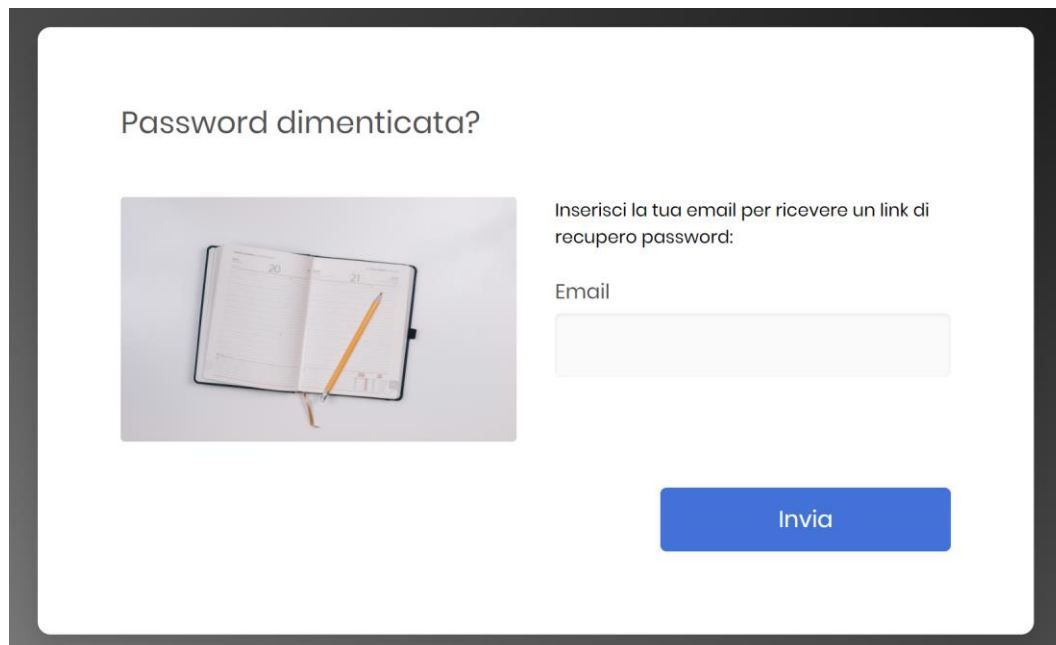


Figura 33 Pagina password dimenticata

A questo punto l'utente dovrà inserire la propria e-mail per ricevere il link di recupero password. Prima di inviare la mail però viene verificato che la e-mail inserita nel campo input corrisponda ad un utente già registrato. Il processo per inviare la e-mail è il medesimo utilizzato per la conferma della registrazione. Infatti viene generato nuovamente un codice criptato che viene sia assegnato nel database all'utente, sia aggiunto nel link di recupero. Dunque se la e-mail inserita nell'input soprastante è valida viene inviata la mail e l'utente viene trasportato in questa pagina dove gli viene detto che è stata inviata una mail con il link di recupero:

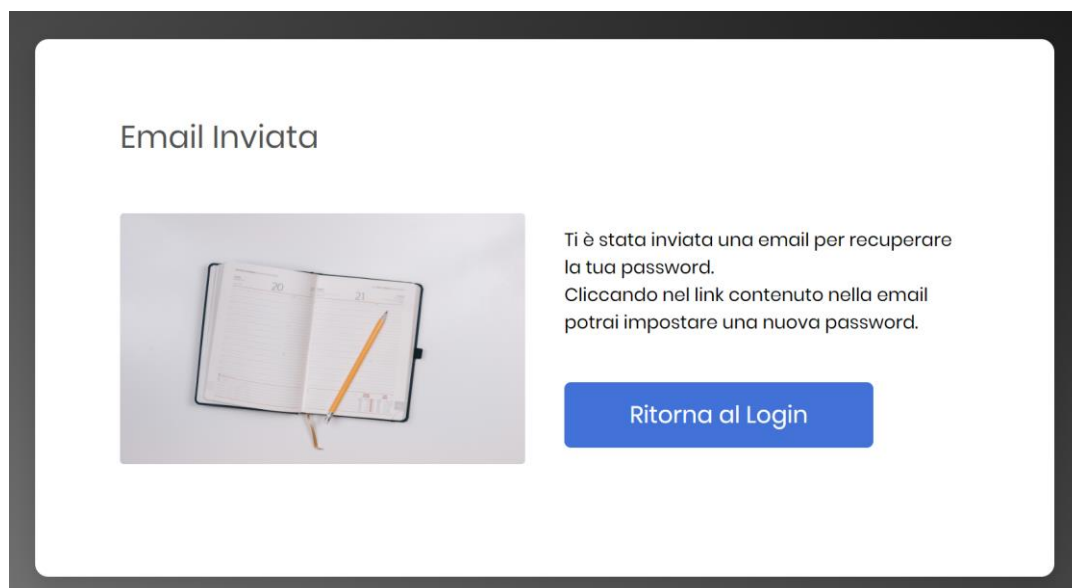


Figura 34 Pagina e-mail inviata

La mail ricevuta dall'utente è la seguente:

Modifica la password

Da: GestioneVacanze <gestione.vacanze2019@gmail.com>

data: 14/12/2019 18:04

Ciao Mattia Toscanelli,
Recentemente è stata richiesta la procedura di modifica password!

[Per modificare la tua password clicca questo link!](#)

Figura 35 E-mail modifica password

Come per la verifica della registrazione, il link contenuto nella mail è monouso. Questo significa che se viene premuto una seconda volta non sarà più possibile modificare la password, ma bisognerà richiedere un nuovo link di recupero.

Per resettare la hash utilizzo questo metodo nel model "CambiaPasswordModel":

```
public function confirm($hash,$email){
    if($hash != 0) {
        $selectCheck = "select * from utente where email='$email' AND hash_mail='$hash'";
        $result = $this->util->fetchAndExecute($selectCheck);
        if ($result != null) {
            $updateUsers = "UPDATE utente SET hash_mail='0', verifica_email=1 WHERE email='$email'";
            $this->util->fetchAndExecute($updateUsers);
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

Per evitare che il link possa essere utilizzato più volte resetto nuovamente la hash dell'utente a "0" così che quando confronto la hash dell'utente con quella della email risultino diverse. Naturalmente per cambiare la password non si può avere la hash uguale a "0" se no tutti potrebbero cambiare la password.

Tornando alla modifica della password, quando l'utente clicca questo link verrà trasportato in questa pagina:

Figura 36 Pagina cambia password

A questo punto dovrà inserire due volte la nuova password. Una volta fatto gli basterà cliccare il bottone “Cambia” e si avvierà il controllo lato client delle due password (lo stesso svolto nella pagina di registrazione). Se il controllo lato client viene sorpassato viene chiamato il metodo `modifyPassword()` del controller “CambiaPassword” che ha sua volta richiama il metodo `modify` del model “CambiaPasswordModel”:

```
public function modify($email, $password1, $password2){
    if($this->validator->checkPassword($password1,$password2)){
        $password = password_hash($password1,PASSWORD_DEFAULT);
        $updateUsers = "UPDATE utente SET password='$password' WHERE email='$email'";
        $this->util->fetchAndExecute($updateUsers);
        return true;
    }
    return false;
}
```

A questo punto avviene anche il controllo lato server e se anche esso ha esito positivo viene codificata la password con il metodo di PHP `password_hash($pass,PASSWORD_DEFAULT)` e in seguito viene impostata all'utente. Una volta fatto ciò l'utente viene trasportato in una pagina dove gli viene detto che la password è stata modificata:

Figura 37 Pagina Password modificata

4.6 Pagina principale con il calendario

4.6.1 Calendario per Gestori e Visualizzatori

Questa è la pagina principale del progetto e il contesto della pagina varia in base al tipo di utente che ha effettuato l'accesso. La prima visualizzazione che andrò a mostrare è per gli utenti di tipo Visualizzatore e per i Gestori:

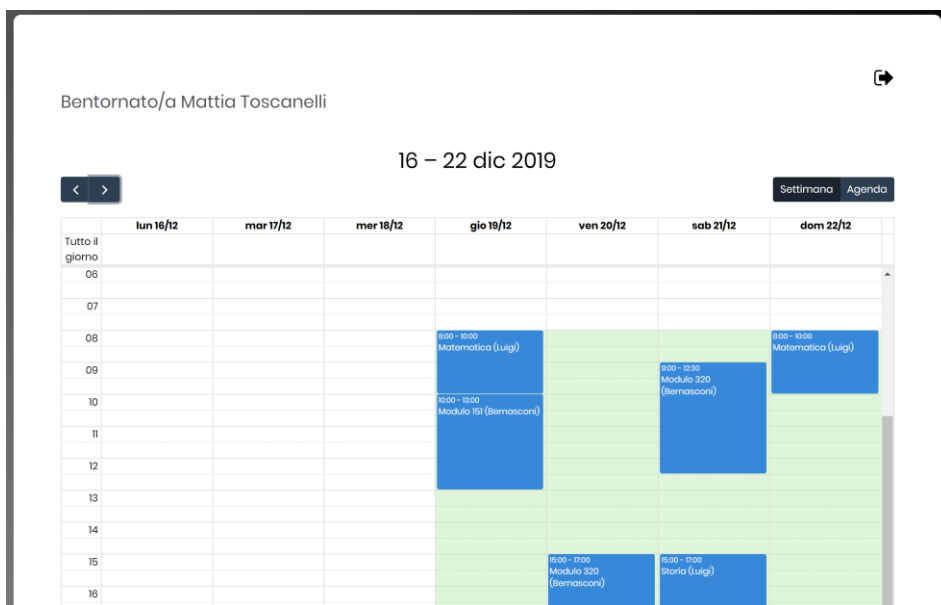


Figura 38 Visualizzazione calendario per Gestore e Visualizzatore

Il Visualizzatore ha la possibilità di vedere il calendario ma senza avere la possibilità di modifica. Il calendario che ho utilizzato si chiama “FullCalendar.js”. FullCalendar è open source ed è ottimo per la visualizzazione di eventi, ma oltre alla possibilità di trascinare gli eventi non è in grado di modificare i dati associati. Perciò offre delle API per far sbizzarrire il più possibile il programmatore.

Tornando al calendario per il Visualizzatore ha la possibilità di cambiare settimana attraverso le frecce in alto a sinistra e gli è permesso cambiare la visualizzazione in formato testuale cliccando sul bottone “Agenda*” in alto a destra:

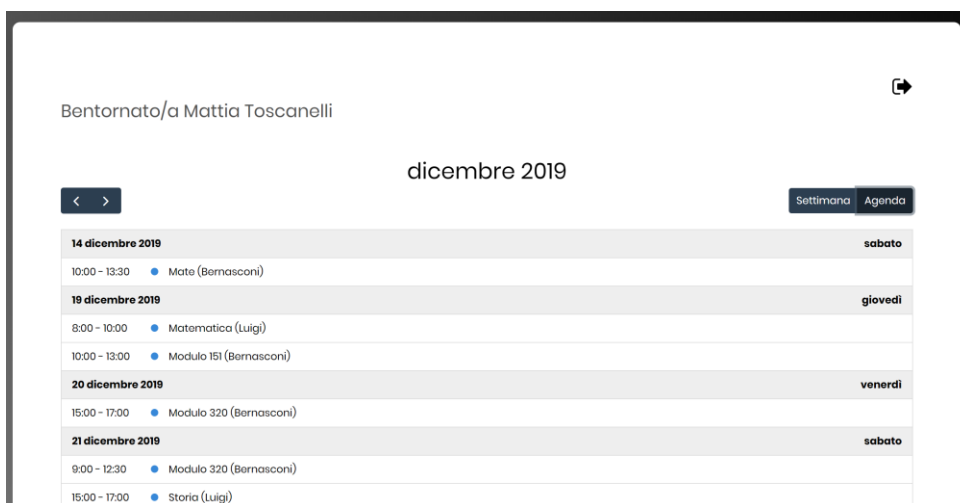


Figura 39 Visualizzazione calendario in formato testuale

Tutte le proprietà del calendario come già detto prima sono uguali sia per i Visualizzatori sia per i Gestori e sono scritte in JavaScript:

```
document.addEventListener('DOMContentLoaded', function() {
    var calendarEl = document.getElementById('calendar');
    var calendar = new FullCalendar.Calendar(calendarEl, {
        plugins: [ 'interaction', 'dayGrid', 'timeGrid', 'list' ],
        header: {
            left: 'prev,next',
            right: 'timeGridWeek,listMonth'
        },
        events: (URL+'Calendario/load'),
        defaultView: 'timeGridWeek',
        defaultDate: moment(new Date()).format("Y-MM-DD"),
        navlinks: true, // can click day/week names to navigate views
        locale: 'it',
        timeZone: 'local',
        firstDay: 1,
        selectable:true,
        selectHelper:true,
        eventClick:function(eventClickInfo) {
            var date = moment(eventClickInfo.event.start).format("Y-MM-DD");
            var start = moment(eventClickInfo.event.start).format("HH:mm:ss");
            var end = moment(eventClickInfo.event.end).format("HH:mm:ss");
            var title = eventClickInfo.event.title;
            alert("Lezione di "+title+"\nData: "+date+"\nOra inizio: "+start+"\nOra fine: "+end);
        }
    });
    calendar.render();
});
```

Tra tutte le righe possiamo trovare interessante l'header che esprime dove devono venir posizionati i bottoni per muoversi all'interno del calendario. Poi possiamo trovare la proprietà event che permette di caricare le lezioni nel calendario. Essa fa una richiesta al metodo load() del controller "Calendario" che a sua volta richiama il metodo loadEvents(\$flag) del model "CalendarioModel".

```
public function loadEvents($flag){
    $data = array();
    $query = "SELECT * FROM giorno WHERE DATE(giorno) >= DATE(NOW())";
    $result = $this->util->fetchAndExecute($query);
    foreach($result as $row) {
        $data[] = array(
            'groupId' => 'availableForLesson',
            'start' => $row[DB_LESSON_DAY]. "T". "08:00:00",
            'end' => $row[DB_LESSON_DAY]. "T". "17:00:00",
            'rendering' => 'background',
        );
    }
    $query = "SELECT id, CONCAT(l.nome, ' ('.u.cognome, ')') as 'nome', ora_inizio, ora_fine, giorno, a.email FROM lezione l, assegna a, utente u WHERE l.id = a.id_lezione AND u.email = a.email AND DATE(l.giorno) >= DATE(NOW()) ORDER BY l.id;";
    $result = $this->util->fetchAndExecute($query);
    foreach($result as $row) {
        if($row[DB_USER_EMAIL]!=$SESSION[SESSION_EMAIL]) {
            if($flag == null) {
                $data[] = array(
                    'id' => $row[DB_LESSON_ID],
                    'title' => $row[DB_LESSON_NAME],
                    'start' => $row[DB_LESSON_DAY] . "T" . $row[DB_LESSON_START],
                    'end' => $row[DB_LESSON_DAY] . "T" . $row[DB_LESSON_END],
                    'constraint' => 'availableForLesson',
                );
            }
        }
    }
}
```

```

    }else{
        $data[] = array(
            'id' => $row[DB_LESSON_ID],
            'title' => $row[DB_LESSON_NAME],
            'start' => $row[DB_LESSON_DAY]. "T". $row[DB_LESSON_START],
            'end' => $row[DB_LESSON_DAY]. "T". $row[DB_LESSON_END],
            'constraint' => 'availableForLesson',
            'color' => 'purple',
        );
    }
}
echo json_encode($data);
}

```

Se non viene passato niente come parametro la variabile \$flag viene istanziata con “null”. Questo flag viene utilizzato dagli utenti di tipo Docente che richiedono di visualizzare sul calendario solamente le proprie ore di lezione. Passando oltre questo metodo si preoccupa di interrogare la tabella giorno per avere tutti i giorni disponibili messi a disposizione dall’admin (questo verrà spiegato in seguito, quando verrà parlato della pagina pannello admin). In seguito interroga le tabelle lezione, assegna e utente per ricavare tutti i dati necessari al calendario per mostrare le lezioni. Una volta preparato l’array contenente gli array dei giorni di lavoro e gli array delle lezioni dei docenti esso viene stampato in json con il metodo di PHP json_encode(). Questo avviene perché FullCalendar richiede tutti gli eventi in formato json.

Ritornando allo script con le proprietà del calendario per i Visualizzatori e Gestori scendendo in basso si può trovare il metodo eventClick(). Esso viene richiamato quando l’utente clicca un evento nel calendario. In pratica all’interno di questo metodo vengono presi tutti i dati dell’evento cliccato e vengono mostrati in un semplice alert:

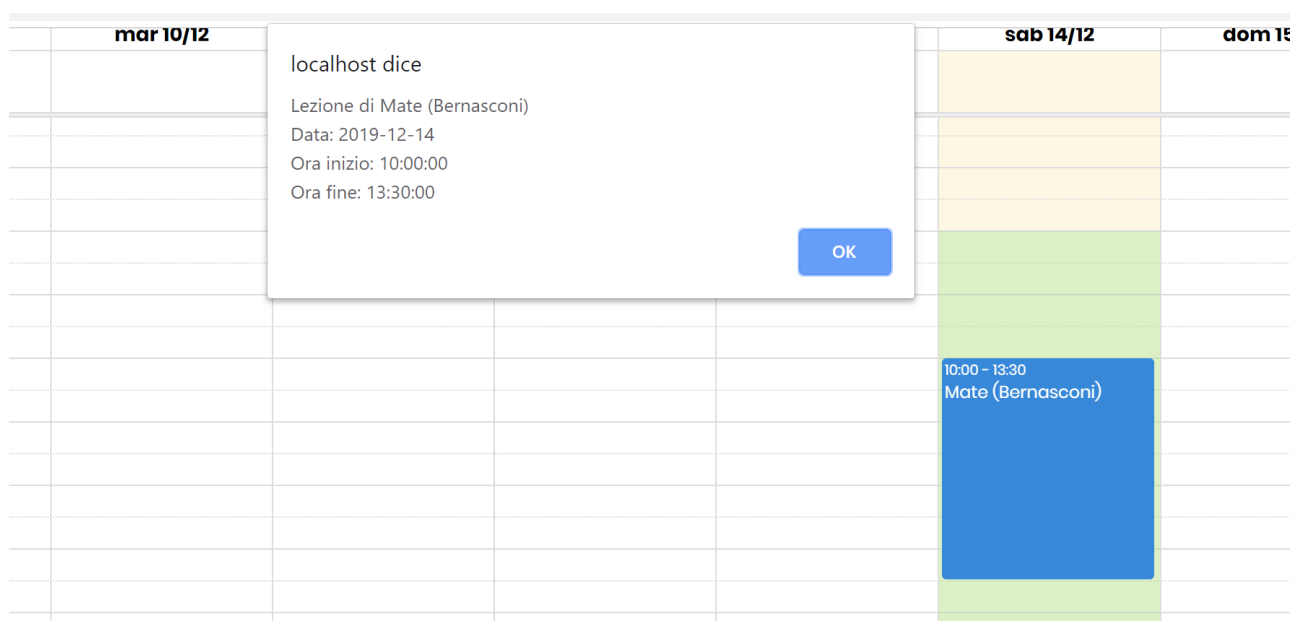


Figura 40 Evento cliccato per Gestore e Visualizzatore

4.6.2 Calendario per Docenti

Come già detto prima gli utenti di tipo Docente hanno funzionalità in più rispetto agli utenti di tipo Visualizzatore e di tipo Gestore. Infatti devono avere la possibilità di editare il calendario. Queste modifiche devono poter far aggiungere/spostare/ridimensionare/eliminare lezioni ai Docenti. Prima di spiegare tutto il funzionamento vi mostro un esempio di visualizzazione del calendario:

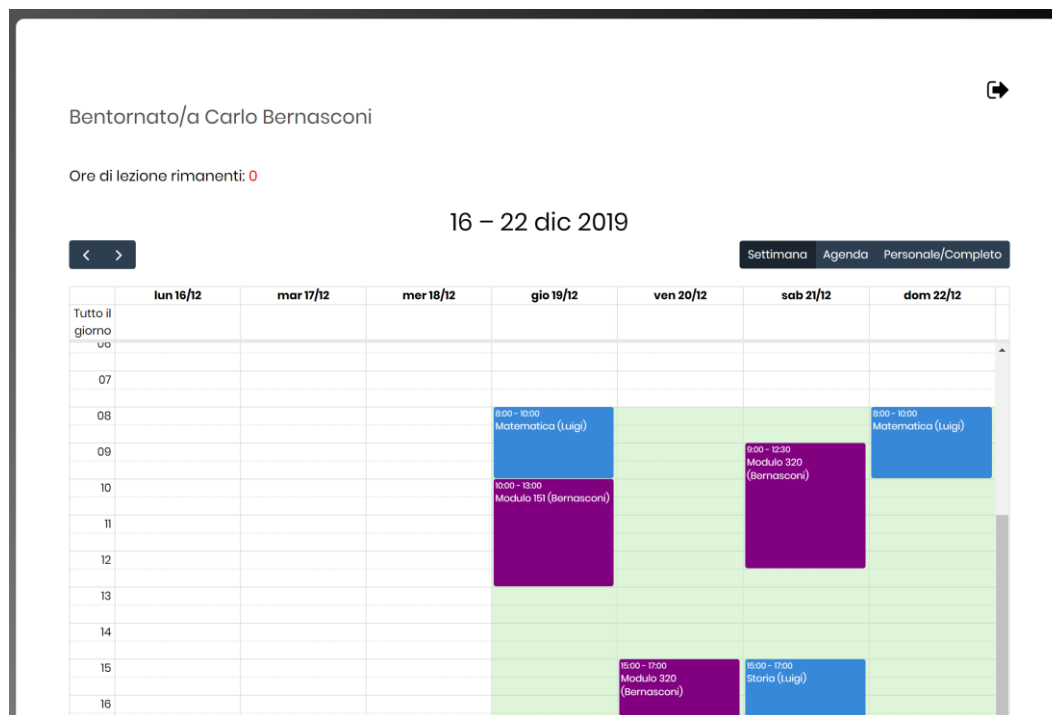


Figura 41 Visualizzazione calendario per Docenti

A differenza del calendario per gli altri due tipi di utente, i Docenti presentano già sotto il messaggio "Bentornato.." il conteggio delle ore di lavoro rimanenti. In caso che questo numero sia a 0 (come nell'immagine soprastante) il numero viene colorato di rosso, altrimenti è di colore nero. Ritornando al calendario si può ben notare che ci sono due colori differenti per la rappresentazioni delle lezioni, ci sono lezioni di colore blu e lezioni di colore viola. Le lezioni di colore blu sono di altri docenti, quindi non è possibile effettuare modifiche. Per verificare se una lezione appartiene ad un altro docente utilizzo questa funzione che si trova nel model "CalendarioModel":

```
private function isMyLessons($user,$id){
    $query = "SELECT * from assegna WHERE id_lezione=$id AND email='$user'";
    $email = $this->util->fetchAndExecute($query);
    return ($email != null);
}
```

In pratica questa funzione bisogna passare come argomento l'e-mail di chi sta effettuando la modifica e l'id della lezione che si sta andando a modificare. A questo punto il metodo esegue una query per ricavare tutte le lezioni che hanno quell'id e che sono assegnate alla e-mail dell'utente. Se la query ritorna un risultato significa che la lezione appartiene all'utente che ha effettuato la modifica.

Dunque le lezioni che si possono modificare sono di colore viola.

Le proprietà del calendario descritte in Javascript sono le stesse utilizzate per i Visualizzatori e Gestori ma con l'aggiunta di qualche funzione in più. La prima tra queste è la seguente:

editable: true,

In pratica questa proprietà, come dice già il suo nome in inglese, permette la manipolazione del calendario, quindi si potranno eseguire le seguenti quattro azioni direttamente sul calendario:

- Aggiunta di lezioni
- Spostamento di lezioni
- Ridimensionamento di lezioni
- Eliminazione di lezioni

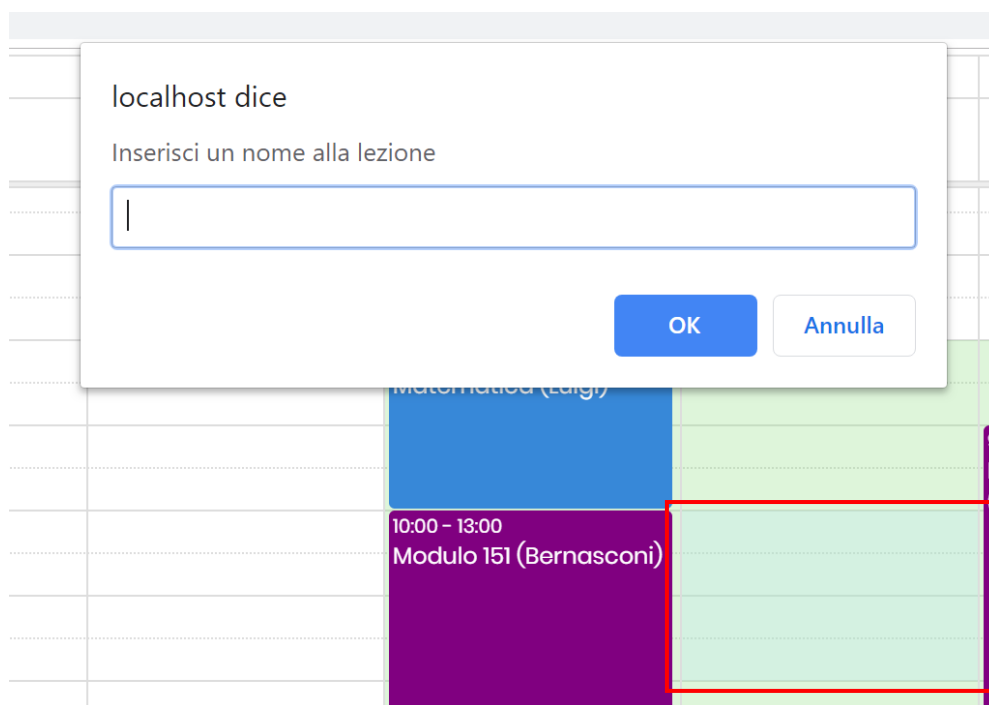


Figura 42 Aggiunta di lezioni

Per aggiungere una lezione bisogna prima selezionare una parte all'interno dei contenitori verdi. Ogni tacca corrisponde a mezz'ora. Una volta selezionata la durata della lezione bisognerà inserire un nome alla lezione. Il nome della lezione dovrà avere almeno 3 caratteri e massimo 50 caratteri, inoltre sono ammesse lettere e numeri. Assegnato anche il nome della lezione sarà sufficiente cliccare il bottone "OK". Per tutta la fase di aggiunta ho aggiunto il seguente codice nel JavaScript con le proprietà del calendario:

```
select: function(selectionInfo) {
    var title = prompt("Inserisci un nome alla lezione");
    if(title) {
        if(v.checkTitle(title)){
            var start = moment(selectionInfo.start).format("Y-MM-DD HH:mm:ss");
            var end = moment(selectionInfo.end).format("Y-MM-DD HH:mm:ss");
            if(v.checkEvent(start, end) != null) {
                $.ajax({
                    url: (URL+"Calendario/insert"),
                    type: "POST",
                    data: {title: title, start: start, end: end},
                    success: function(response) {
                        if(response.includes(",")){
                            $.notify("Lezione Aggiunta", "success");
                        }
                    }
                });
            }
        }
    }
}
```

```

        document.getElementById("ore_lavoro").innerHTML = "Ore di lezione rim
ananti: " + (response.split(',')[1]==0?"<b style='color:red'>"+response.split(',')[1]+"</b>
":response.split(',')[1]);
    }else if(response == "2"){
        $.notify("Giorno non disponibile per lezioni (Solo in spazi verdi)",
'error');
    }else if(response == "3"){
        $.notify("Massimo due lezioni al giorno", 'error');
    }else if(response == "4"){
        $.notify("Non ci possono essere due lezioni nello stesso momento", 'e
rror');
    }
    calendar.refetchEvents();
}
    })
}
    }else{
        $.notify("Lezione non valida (Min 1, Max. 4 ore, nell'intervallo 8-
17 e nei giorni verdi)", 'error');
    }
}
    }else{
        $.notify("Nome lezione non valido (Min 3 caratteri)", 'error');
    }
}
}
}

```

In pratica questa funzione riceve come parametro l'evento che si vuole aggiungere. E come prima cosa mostra l>alert per ricevere il nome della lezione. Se il nome della lezione rispetta i criteri prestabiliti vengono lette le ore di inizio e fine lezioni attraverso la libreria *"moment.js"*. A questo punto le due ore vengono verificate nel metodo *checkEvent(oraInizio,oraFine)* che controllerà se l'intervallo di tempo è maggiore di 1 ora e inferiore di 4 ore. Se anche questo controllo viene superato viene fatta una richiesta in ajax al metodo *insert()* del controller "Calendario" e gli vengono passati tutti i dati della lezione da aggiungere. Questo metodo *insert()* a sua volta passerà i dati al metodo *insert(\$title, \$start, \$end, \$email)* del model "CalendarioModel":

```

public function insertEvents($title, $start, $end, $email){
    $date = explode(" ", $start)[0];
    $selectDay = "SELECT * FROM giorno WHERE giorno='$date'";
    $day = $this->util->fetchAndExecute($selectDay);
    if($day != null) {
        $selectEvent = "SELECT * FROM lezione WHERE giorno='$date'";
        $event = $this->util->fetchAndExecute($selectEvent);
        if (count($event) < 2) {
            if ($this->checkOverlap(null,explode(" ", $start)[1],explode(" ", $end)[1], $date)==0){
                $query = "INSERT INTO lezione VALUES (null, '$title', '$start', '$end', '$date)";
                $this->util->fetchAndExecute($query);
                $selectEvent =
"SELECT * FROM lezione WHERE giorno='$date' and ora_inizio='$start'";
                $event = $this->util->fetchAndExecute($selectEvent);
                $id = $event[0][DB_LESSON_ID];
                $query = "INSERT INTO assegna VALUES ('" . $email . "', '$id)";
                $this->util->fetchAndExecute($query);
                $diff = $this->getDiffHour(explode(" ", $start)[1], explode(" ", $end)[1]);
                if($this->getHours($email)-$diff > 0){
                    $query = "UPDATE utente set ore_lavoro=ore_lavoro-
$diff where email='$email'";
                }else{
                    $query = "UPDATE utente set ore_lavoro=0 where email='$email'";
                }
                $this->util->fetchAndExecute($query);
                $hour = $this->getHours($email);
                echo "1,$hour";
            }else {
                echo 4;
            }
        }
    }
}

```

```

    }
  }else{
    echo 3;
  }
}
}else{
  echo 2;
}
}
}

```

I parametri che riceve sono 4: il titolo della lezione da aggiungere, l'ora di inizio lezione, l'ora di fine lezione e la email dell'utente che vuole aggiungere la lezione. Come prima cosa viene fatta una query per verificare che la data in cui si vuole aggiungere una lezione sia disponibile per mettere lezioni. Se la risposta è sì viene controllato se sono già presenti 2 o più lezioni nello stesso giorno. Questo controllo viene eseguito perché appunto non ci possono essere più di 2 lezioni nello stesso giorno. In caso ci fossero meno di 2 lezioni viene effettuato l'ultimo controllo. Questo controllo verifica che la lezione che si vuole aggiungere non sia in contemporaneo ad un'altra lezione. Per verificare la concorrenza tra lezioni utilizzo questo metodo `checkOverlap($id_lezione,$ora_inizio,$ora_fine,$data_lezione)` sempre nel model "CalendarioModel":

```

public function checkOverlap($id, $timeNewStart, $timeNewEnd, $dateLesson){
    $timeNewStart = strtotime($dateLesson."T".$timeNewStart);
    $timeNewEnd = strtotime($dateLesson."T".$timeNewEnd);
    $selectDay = "";
    if($id != null) {
        $selectDay = "SELECT * FROM lezione WHERE giorno='$dateLesson' and id<>$id";
    }else{
        $selectDay = "SELECT * FROM lezione WHERE giorno='$dateLesson'";
    }
    $days = $this->util->fetchAndExecute($selectDay);
    $check = 0;
    foreach ($days as $row) {
        $timeStart = strtotime($dateLesson . "T" . $row[DB_LESSON_START]);
        $timeEnd = strtotime($dateLesson . "T" . $row[DB_LESSON_END]);
        if (!(($timeNewStart >= $timeEnd) || ($timeNewEnd <= $timeStart))) {
            $check++;
        }
    }
    return $check != 0;
}

```

Viene eseguita una query per cercare tutte le lezioni appratenti al giorno in cui si vuole aggiungere la lezione. A questo punto vengono verificate in sequenza tutte le lezioni controllando le ore di inizio e di fine rispetto alla lezione da aggiungere. Se non ci sono congruenze ritorna "false" altrimenti "false". Se pure questo controllo ha esito positivo la lezione viene aggiunta e le ore di lavoro del Docente vengono scalate.

In caso che uno dei controlli non avesse avuto successo, come tutti i controlli che verranno a seguire per la modifica delle lezioni, vengono mostrati a schermo con delle notifiche i problemi che sono stati riscontrati, sia lato client sia lato server.

Per spostare una lezione bisogna semplicemente cliccare la lezione e trascinarla dove si vuole (naturalmente sempre negli spazi verdi).

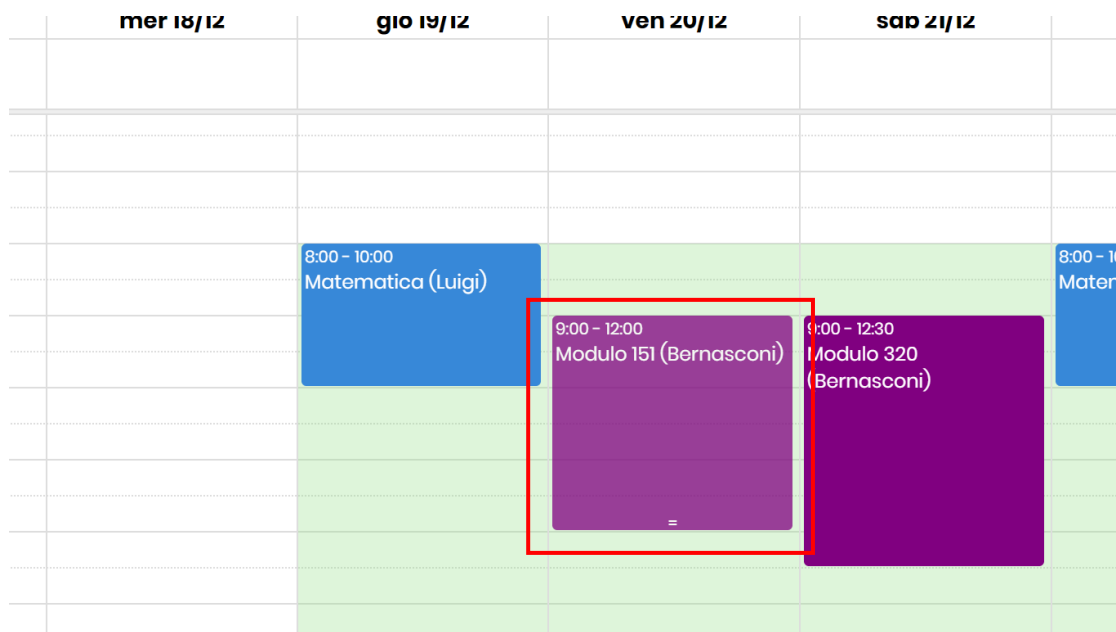


Figura 43 Spostamento di una lezione

Quando si sposta una lezione viene richiamato questo metodo nelle proprietà del calendario in JavaScript:

```
eventDrop: function(eventDropInfo) {  
    [...]  
}
```

Questo metodo riceve come parametro la lezione spostata, così da ricavare id, ora di inizio, ora di fine della lezione, un po' come l'aggiunta delle lezioni. Anch'esso richiama attraverso ajax un metodo del controller "Calendario". In questo caso richiama il metodo update() che a sua volta il richiama il metodo updateEvents(\$id,\$start,\$end,\$email) del model "CalendarioModel":

```
public function updateEvents($id, $start, $end, $email){  
    if($this->isMyLessons($email,$id)){  
        $date = explode(" ", $start)[0];  
        $selectEvent = "SELECT * FROM lezione WHERE giorno='$date'";  
        $event = $this->util->fetchAndExecute($selectEvent);  
        if (count($event) < 2) {  
            if ($this->checkOverlap($id,explode(" ", $start)[1],explode(" ", $end)[1],$date)==0){  
                $query = "UPDATE lezione set ora_inizio='$start', ora_fine='$end', giorno='$date'  
where id=$id";  
                $this->util->fetchAndExecute($query);  
                echo 1;  
            }else {  
                echo 3;  
            }  
        }else{  
            $selectEvent = "SELECT * FROM lezione WHERE giorno='$date' && id=$id";  
            $event = $this->util->fetchAndExecute($selectEvent);  
            if($event != null) {  
                if ($this->  
>checkOverlap($id,explode(" ", $start)[1], explode(" ", $end)[1], $date) == 0) {  
                    $query = "UPDATE lezione set ora_inizio='$start', ora_fine='$end', giorno='$d  
ate' where id=$id";  
                    $this->util->fetchAndExecute($query);
```


Questo metodo riceve come parametro la lezione ridimensionata, così da ricavare id, ora di inizio, ora di fine della lezione, un po' come l'aggiunta e lo spostamento delle lezioni. Anch'esso richiama attraverso ajax un metodo del controller "Calendario". In questo caso richiama il metodo `resize()` che a sua volta il richiama il metodo `resizeEvents($id,$start,$end,$email)` del model "CalendarioModel":

```
public function resizeEvents($id, $start, $end, $email){
    if($this->isMyLessons($email,$id)) {
        $date = explode(" ", $start)[0];
        if ($this->checkOverlap($id, explode(" ", $start)[1], explode(" ", $end)[1], $date) == 0) {
            $query = "SELECT ora_fine from lezione WHERE id=$id";
            $oldEnd = $this->util->fetchAndExecute($query);
            $query = "UPDATE lezione SET ora_fine='$end' WHERE id=$id";
            $this->util->fetchAndExecute($query);
            $email = $this->getEmailByLesson($id);
            $timeOldEnd = strtotime($oldEnd[0][DB_LESSON_END] . " " . $date);
            $timeEnd = strtotime($end);
            $diff = $this->getDiffHour(explode(" ", $end)[1], $oldEnd[0][DB_LESSON_END]);
            if ($timeOldEnd > $timeEnd) {
                $query = "UPDATE utente set ore_lavoro=ore_lavoro+$diff where email='$email'";
            } else {
                if($this->getHours($email)+$diff > 0){
                    $query = "UPDATE utente set ore_lavoro=ore_lavoro+$diff where email='$email'";
                } else {
                    $query = "UPDATE utente set ore_lavoro=0 where email='$email'";
                }
            }
            $this->util->fetchAndExecute($query);
            $hour = $this->getHours($email);
            echo "1,$hour";
        } else {
            echo 2;
        }
    } else {
        echo 3;
    }
}
```

I parametri che riceve sono 4: l'id della lezione da spostare, l'ora di inizio lezione, l'ora di fine lezione e la e-mail dell'utente che vuole ridimensionare la lezione. A questo punto esegue il metodo `isMyLessons($email,$id_lezione)`, spiegato precedentemente, che mi permette di capire se la lezione spostata appartiene all'utente che esegue l'azione. In seguito viene verificato che non ci siano lezioni nello stesso momento. Se anche questo controllo ha esito positivo la lezione viene ridimensionata e se la lezione è stata ingrandita vengono tolte le ore di lezione al docente, se invece è stata diminuita vengono aggiunte le ore di lezione corrispondenti.

Infine per eliminare una lezione bisogna semplicemente cliccare su quest'ultima e comparirà un alert che richiederà definitivamente se si vuole veramente eliminare la lezione:

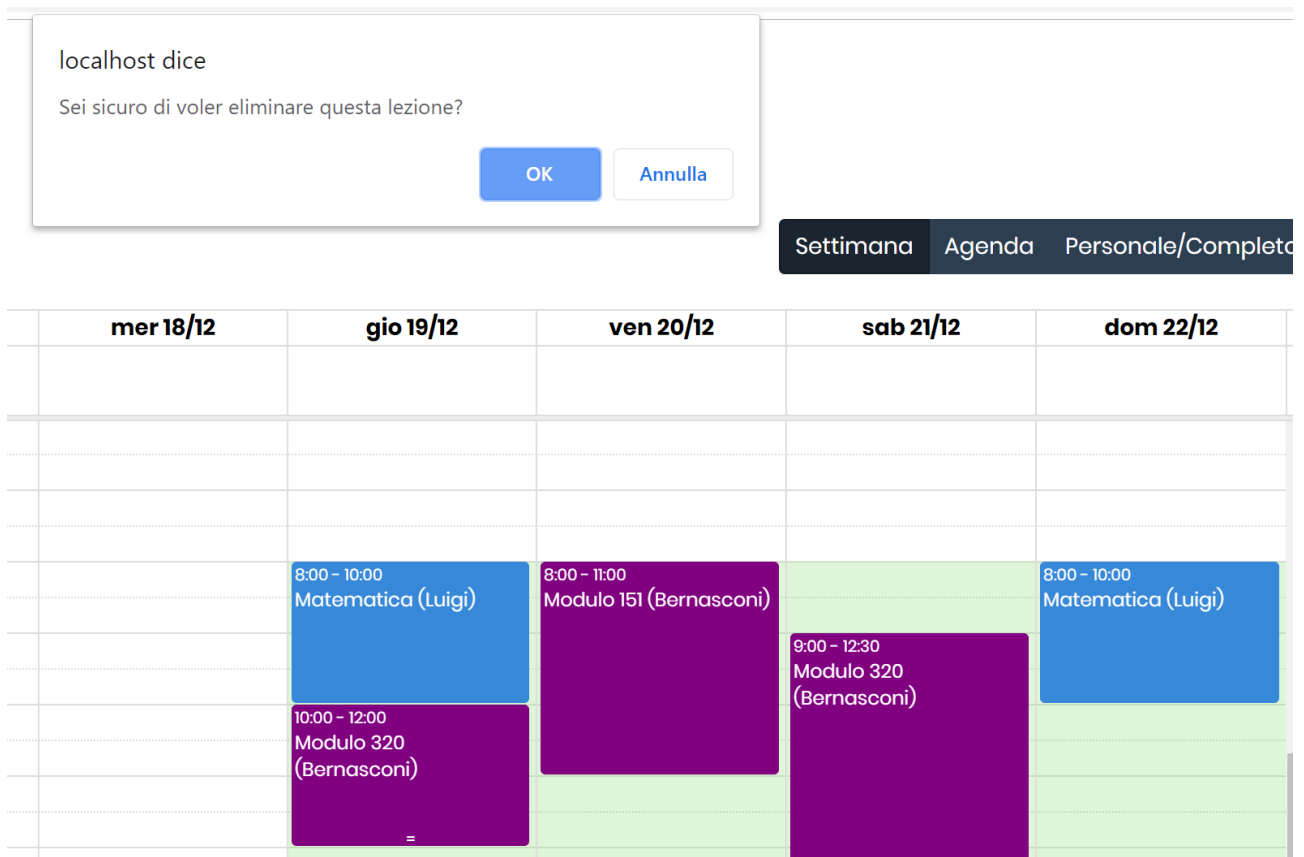


Figura 45 Eliminazione lezione

Quando si clicca una lezione viene richiamato questo metodo nelle proprietà del calendario in JavaScript:

```
eventClick: function(eventClickInfo) {
    if(confirm("Sei sicuro di voler eliminare questa lezione?")) {
        [...]
    }
}
```

Questo metodo lo utilizzo per eliminare le lezioni. Quando si clicca su una propria lezione viene richiesto con un alert se si è sicuri di eliminare la lezione. Se la risposta è "OK" la lezione verrà eliminata. Per eliminare la lezione vengono presi i dati dall'evento passato come parametro (id, ora di inizio, ora di fine della lezione), un po' come l'aggiunta, lo spostamento e il ridimensionamento delle lezioni. Anch'esso richiama attraverso ajax un metodo del controller "Calendario". In questo caso richiama il metodo delete() che a sua volta il richiama il metodo deleteEvents(\$id,\$start,\$end,\$email) del model "CalendarioModel":

```
public function deleteEvents($id, $start, $end, $email){
    if($this->isMyLessons($email,$id)){
        $email = $this->getEmailByLesson($id);
        $query = "DELETE from assegna WHERE id_lezione=$id";
        $this->util->fetchAndExecute($query);
        $query = "DELETE from lezione WHERE id=$id";
        $this->util->fetchAndExecute($query);
        $diff = $this->getDiffHour(explode(" ", $start)[1], explode(" ", $end)[1]);
        $query = "UPDATE utente set ore_lavoro=ore_lavoro+$diff where email='$email'";
        $this->util->fetchAndExecute($query);
        $hour = $this->getHours($email);
    }
}
```

```

        echo "1,$hour";
    }else{
        echo 2;
    }
}

```

I parametri che riceve sono 4: l'id della lezione da spostare, l'ora di inizio lezione, l'ora di fine lezione e la e-mail dell'utente che vuole eliminare la lezione. A questo punto esegue, come per le tre azioni già descritte, il metodo `isMyLessons($email,$id_lezione)`. Fatto ciò eliminare tutte le correlazioni con la lezione da eliminare, elimina la lezione ed infine aggiunge le ore di lavoro al docente che ha deciso di eliminare l'evento.

I docenti possono eseguire un'altra funzione. Infatti si può notare che in alto a destra del calendario esiste un altro bottone chiamato "Personale/Completo" che permette al docente di carica solo le proprie lezioni:

Ore di lezione rimanenti: 0



Figura 46 Calendario personale

Se esso viene cliccato nuovamente sarà possibile rivedere il calendario completo.

4.6.3 Ulteriori funzioni per il calendario

Sotto al calendario si può trovare il bottone “Stampa”. Se cliccato verrà aperta la visualizzazione di stampa dove verrà mostrato solamente il calendario.



Figura 47 Bottone per stampare il calendario

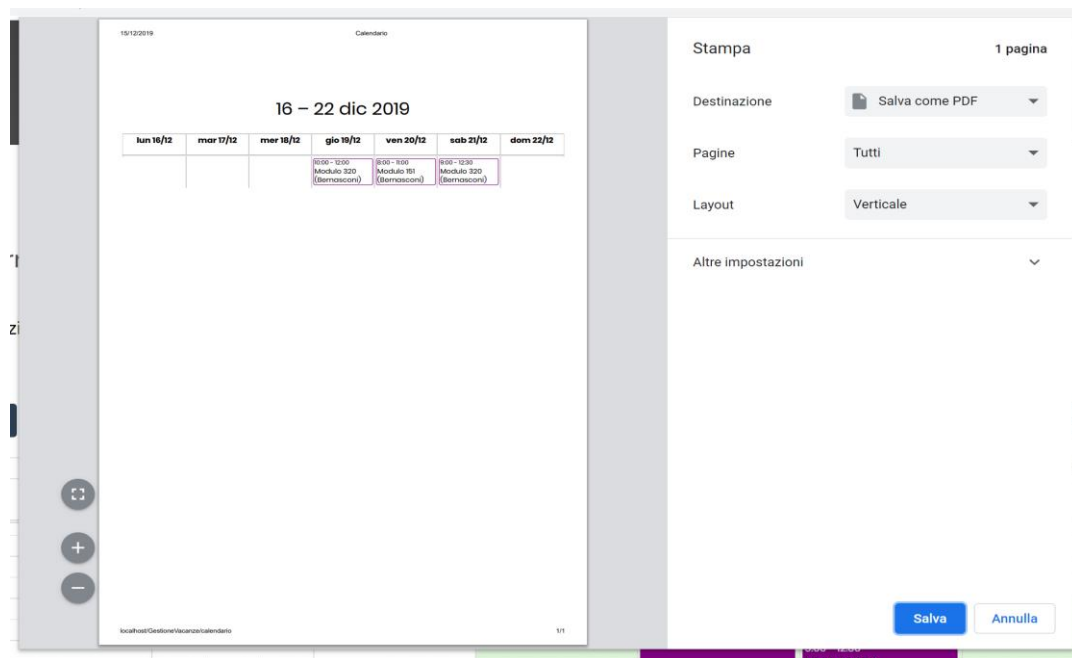


Figura 48 Visualizzazione stampa calendario

Per mostrare solo il calendario nella visualizzazione di stampa ho aggiunto questo codice di stile:

```
@media print {
  body * {
    visibility: hidden;
  }
  #cal, #cal * {
    visibility: visible;
  }
  #cal {
    position: absolute;
    margin-left: auto;
    margin-right: auto;
    left: 0;
    right: 0;
    top: 100px;
  }
}
```

Infine i Gestori della pagina hanno un'ulteriore opzione al termine della pagina del calendario. Infatti se si scorre fino a fine pagina si può trovare il bottone "Pannello Admin"

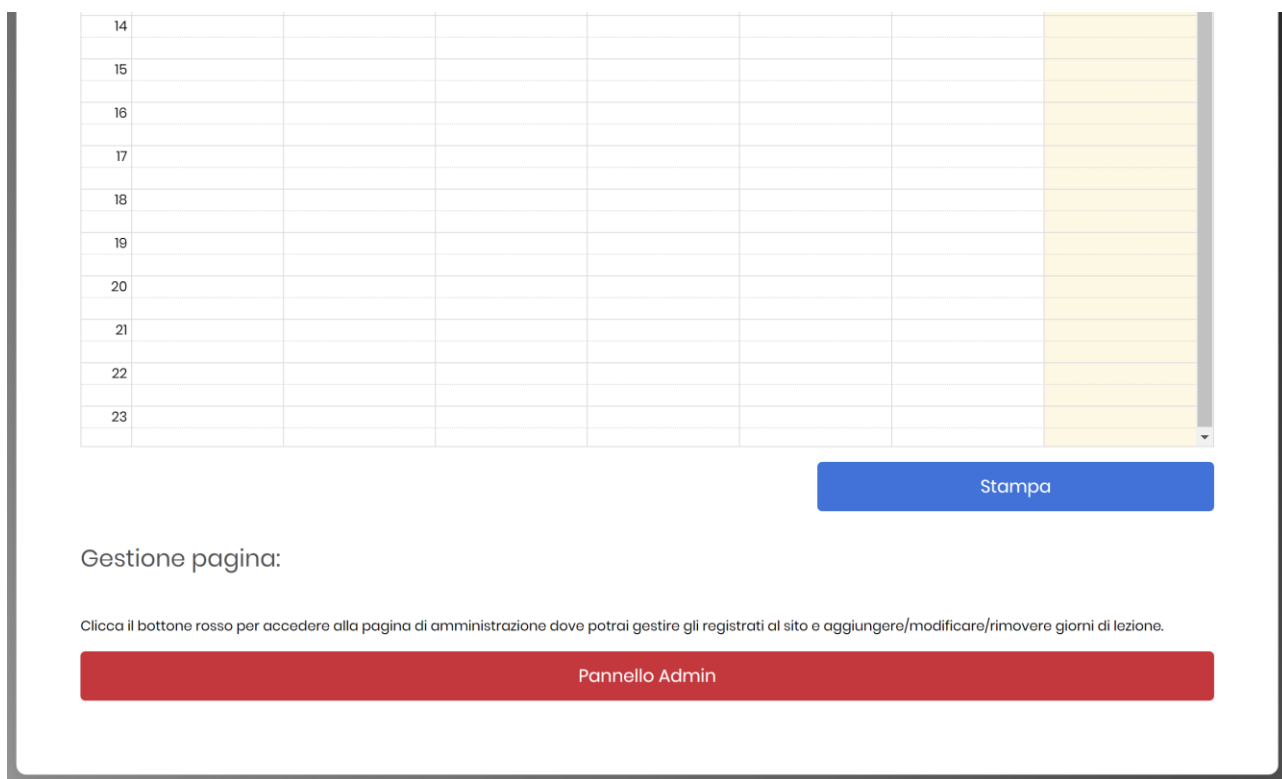


Figura 49 Bottone pannello admin

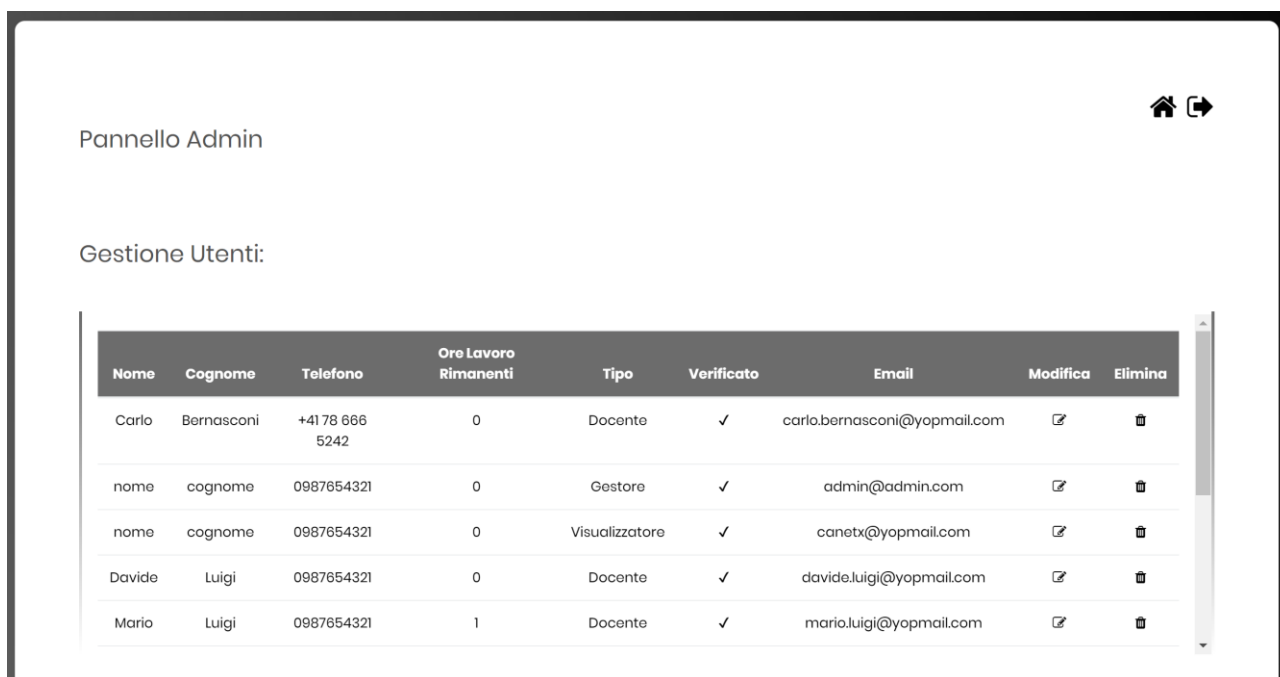
Se esso viene cliccato si accede alla pagina di amministrazione della pagina. Questa verrà descritta nel capitolo successivo.

4.7 Pagina di amministrazione

La pagina di amministrazione, come già detto, è accessibile solamente ai Gestori della pagina. In questa pagina si potranno eseguire le seguenti azioni:

- Modifica di utenti
- Eliminazioni di utenti
- Modifica in contemporaneo delle ore di lavoro dei docenti
- Aggiunta di giorni di lavoro
- Eliminazione di giorni di lavoro
- Eliminazione di giorni di lavoro fino ad oggi

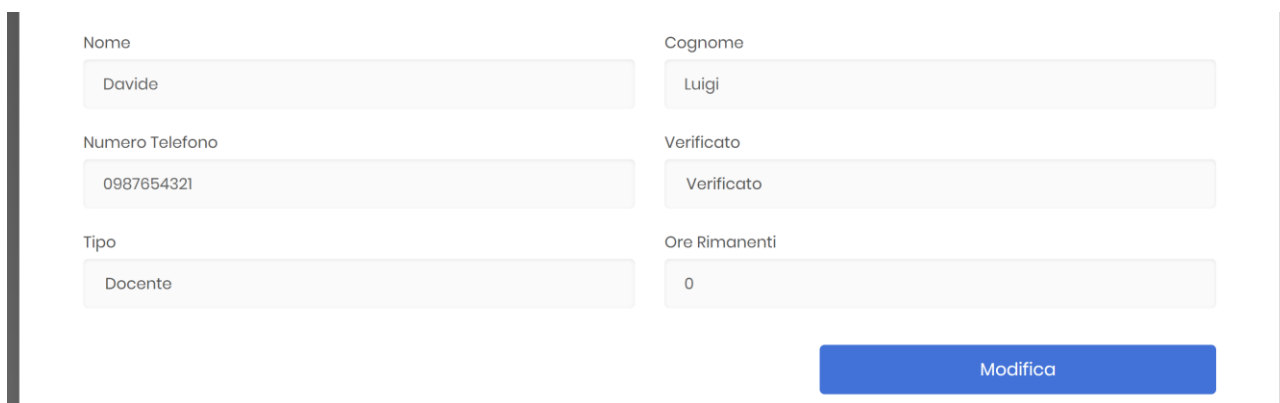
La prima parte della pagina che viene mostrata è la tabella degli utenti registrati:



Nome	Cognome	Telefono	Ore Lavoro Rimanenti	Tipo	Verificato	Email	Modifica	Elimina
Carlo	Bernasconi	+41 78 666 5242	0	Docente	✓	carlo.bernasconi@yopmail.com		
nome	cognome	0987654321	0	Gestore	✓	admin@admin.com		
nome	cognome	0987654321	0	Visualizzatore	✓	canetx@yopmail.com		
Davide	Luigi	0987654321	0	Docente	✓	davide.luigi@yopmail.com		
Mario	Luigi	0987654321	1	Docente	✓	mario.luigi@yopmail.com		

Figura 50 Pannello admin - tabella utenti

In questa tabella si possono vedere tutti gli utenti che hanno verificato la e-mail dopo aver eseguito la registrazione. Questi utenti vengono letti dalla tabella utente e mostrati in una tabella HTML. Premendo la matita di fianco ai dati dell'utente si può modificare l'utente corrispondente. Infatti una volta cliccata comparirà un form sotto la tabella con tutti i dati dell'utente da poter modificare:



Nome

Cognome

Numero Telefono

Verificato

Tipo

Ore Rimanenti

Figura 51 Pannello admin - modifica utente

Una volta modificati i dati dell'utente sarà sufficiente cliccare il bottone “Modifica” e come per la registrazione avverrà innanzitutto il controllo lato client dei dati per poi richiamare il metodo `modifyUser($email_utente)` che leggerà tutti i dati dal form tramite il metodo “POST”, pulirà i dati con il metodo `test_input($var)` della classe “Util”, per poi richiamare il metodo `modifyUser($user, array $data, $userEmail)` del model “PannelloAdminModel”:

```
function modifyUser($user, array $data, $userEmail){
    if($this->validator->checkName($data[0]) && $this->validator->checkName($data[1]) &&
        $this->validator->checkNumber($data[2]) && $this->validator->checkVerify($data[4]) &&
        $this->validator->checkHours($data[3]) && $this->validator->checkType($data[5])) {
        $hours = 0;
        if($data[3] == '' || $data[5]!="Docente"){
            $hours = 0;
        }else{
            $hours = $data[3];
        }
        $hours = intval($hours);
        $verify = "";
        if($data[4] == "Verificato"){
            $verify = 1;
        }else if($data[4] == "Non verificato"){
            $verify = 0;
        }
        if($user != $userEmail){
            $selectUsers = "UPDATE utente SET nome='$data[0]', cognome='$data[1]',
                numero_telefono='$data[2]', ore_lavoro=$hours,
                tipo='$data[5]', verificato=$verify WHERE email='$user'";
            $this->util->fetchAndExecute($selectUsers);
            return true;
        } else {
            return false;
        }
    }else{
        return false;
    }
}
```

Questo metodo riceve come parametri, l'e-mail dell'utente da modificare, i dati modificati nel form (in un array) e la l'e-mail del Gestore che sta effettuando la modifica. Come prima cosa questo metodo verifica i dati inseriti nel form con la classe “Validator”. Se tutti i controlli hanno esito positivo vengono settate a “0” le ore lavorative se il tipo di utente modificato non è un Docente. In seguito viene trasformata la voce “Verificato” con il numero “1” e se la voce fosse “Non verificato” con “0”. Questo campo stabilisce la conferma della registrazione da parte del Gestore. Quando un utente si registra questo campo è uguale a “0”, ma se esso diventa a “1” l'utente potrà effettuare l'accesso. Una volta preparati tutti i dati da modificare dell'utente viene fatto un ultimo controllo, cioè viene verificato che la e-mail di chi sta effettuando la modifica sia differente della e-mail dell'utente da modificare. Se le e-mail sono diverse i dati dell'utente verranno modificati.

Se viene cliccato il cestino di fianco hai dati dell'utente verrà inizialmente chiesto attraverso un alert se si è sicuri di eliminare quel determinato utente:



Figura 52 Pannello admin - eliminazione utente

Se la risposta è "OK" viene richiamato il metodo deleteUser(\$user) del controller "PannelloAdmin" che a sua volta richiama il metodo deleteUser(\$user, \$userEmail) del model "PannelloAdminModel":

```
function deleteUser($user, $userEmail){
  if($user != $userEmail){
    $selectIdEvents = "SELECT id_lezione from assegna WHERE email='$user'";
    $ids = $this->util->fetchAndExecute($selectIdEvents);
    foreach ($ids as $id){
      $query = "DELETE from assegna WHERE id_lezione=".$id[DB_ASSIGN_LESSON_ID];
      $this->util->fetchAndExecute($query);
      $query = "DELETE from lezione WHERE id=".$id[DB_ASSIGN_LESSON_ID];
      $this->util->fetchAndExecute($query);
    }
    $selectUsers = "DELETE FROM utente WHERE email='$user'";
    $this->util->fetchAndExecute($selectUsers);
    return true;
  }else{
    return false;
  }
}
```

Questo metodo riceve come parametri, l'e-mail dell'utente da eliminare e la l'e-mail del Gestore che sta effettuando l'eliminazione. Prima di eliminare l'utente viene fatto controllo, cioè viene verificato che la e-mail di chi sta effettuando l'eliminazione sia differente della e-mail dell'utente da eliminare, per evitare che il Gestore si auto-elimini da solo. In caso che la e-mail fosse differente vengono inizialmente selezionate tutte le lezioni appratenti all'utente che si vuole eliminare. In seguito vengono eliminate tutte le correlazioni delle lezioni di quel determinato utente e poi le lezioni dell'utente. Una volta eliminate tutte le lezioni viene eliminato l'utente stesso dal database.

Scorrendo giù per la pagina si può trovare la voce “Reimposta ore di lavoro per docenti”. In questa sezione si può inserire un ora da resettare a tutti i docenti dell’applicazione. Per esempio se si inserisce “40” nell’input ore di lavoro e poi si preme il bottone “Modifica”, tutti i docenti del sito avranno a disposizione 40 ore di lavoro da usufruire.

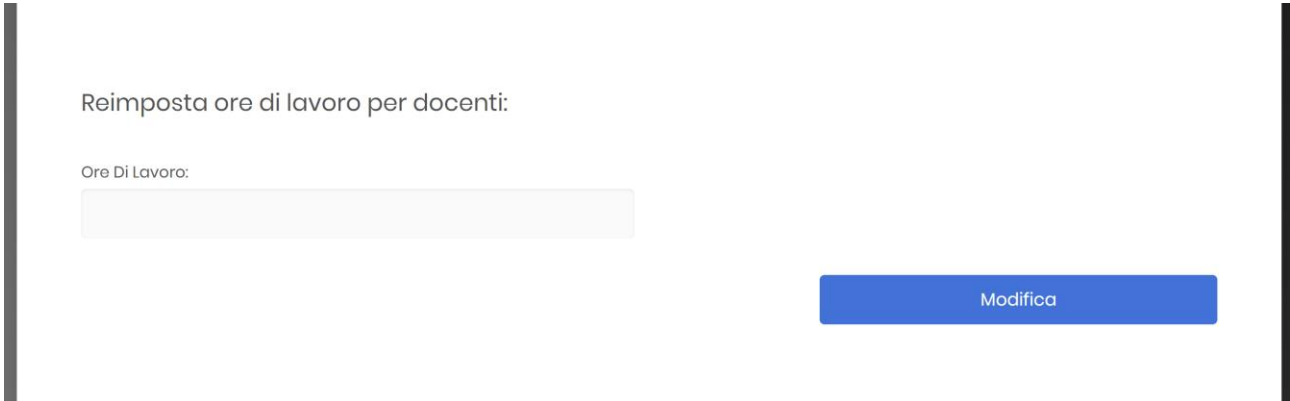


Figura 53 Pannello admin - reimposta ore di lavoro per docenti







Quando si clicca il bottone modifica viene verificato tramite client che le ore inserite siano maggiori di 0 e inferiori di 999. Se questo controllo va a buon fine viene richiamato il metodo `setAllHours()` nel controller “PannelloAdmin” che a sua volta richiama il metodo `modifyHours($hours)` del model “PannelloAdminModel”:


```
function modifyHours($hours){
    if($this->validator->checkHours($hours)){
        $hours = intval($hours);
        $updateUtente = "UPDATE utente SET ore_lavoro=$hours WHERE tipo='Docente'";
        $this->util->fetchAndExecute($updateUtente);
        return true;
    }
    return false;
}
```

Questo metodo riceve come input le ore inserite dall’input, verifica lato server se le ore inserite siano maggiori di 0 e inferiori di 999 e se così fosse assegna queste ore a tutti i docenti del sito.

Scorrendo fino alla fine della pagina di amministrazione si può trovare un'ultima tabella. Questa tabella mostra tutti i giorni disponibili per poter piazzare le lezioni:

Giorni disponibili per lezioni:

Giorno	Elimina
2019-12-12	
2019-12-13	
2019-12-14	
2019-12-19	
2019-12-20	
2019-12-21	

Giorno Lezione


Elimina Tutti

Aggiungi

Figura 54 Pannello admin - tabella dei giorni lavorativi

Accanto ai giorni disponibili per le lezioni si può trovare un cestino, come per la tabella utente. Se esso viene cliccato come per gli utenti viene eliminato quest'ultimo dal database. Quando si vuole eliminare un giorno viene richiesto prima di tutto se si è sicuri di voler eliminare quel determinato giorno:

nelloAdmin

Giorni disponibili per lezioni:

localhost dice

Sei sicuro di voler eliminare questo giorno?

OK

Annulla




Giorno	Elimina
2019-12-12	
2019-12-13	
2019-12-14	

Figura 55 Pannello admin - eliminazione giorno

Se la risposta è “OK” viene richiamato il metodo deleteDay(\$day) del controller “PannelloAdmin” che a sua volta richiama il metodo deleteDay(\$day) del model “PannelloAdminModel”:

```
function removeDay($day){
    $selectIdEvent = "SELECT id from lezione where giorno='$day'";
    $ids = $this->util->fetchAndExecute($selectIdEvent);
    foreach ($ids as $id){
        $query = "DELETE from assegna WHERE id_lezione='".$id[DB_LESSON_ID]."'";
        $this->util->fetchAndExecute($query);
        $query = "DELETE from lezione WHERE id='".$id[DB_LESSON_ID]."'";
        $this->util->fetchAndExecute($query);
    }
    $deleteAggiunge = "DELETE FROM aggiunge WHERE giorno='$day'";
    $this->util->fetchAndExecute($deleteAggiunge);
    $deleteDay = "DELETE FROM giorno WHERE giorno='$day'";
    $this->util->fetchAndExecute($deleteDay);
}
```

Questo metodo riceve come parametro la data del giorno da eliminare. Inizialmente selezionate tutte le lezioni appratenti a quel determinato giorno. In seguito vengono eliminate tutte le correlazioni delle lezioni di quel determinato giorno per poi eliminare le lezioni stesse. Una volta eliminate tutte le lezioni viene eliminato la correlazione nella tabella aggiunge corrispondente a quel giorno per poi infine eliminare il giorno stesso.

Se si vuole aggiungere un giorno di lavoro bisogna inserire la data nell'input proposto sotto la tabella dei giorni. Una volta scelta la data basta semplicemente cliccare il bottone “Aggiungi” e verrà verificata la data con un controllo lato client. Questo controllo verifica che la data inserita sia maggiore della data odierna. Se la risposta è si viene richiamato il metodo addDay() nel controller “PannelloAdmin” che a sua volta richiama il metodo addDay(\$day, \$userEmail) nel model “PannelloAdminModel”:

```
function addDay($day, $userEmail){
    if($this->validator->checkDateCalendarInsert($day)) {
        $dataN = explode("/", $day);
        $day = $dataN[2] . "-" . $dataN[1] . "-" . $dataN[0];
        $insertDay = "INSERT INTO giorno VALUES ('$day')";
        $this->util->fetchAndExecute($insertDay);
        $insertDay = "INSERT INTO aggiunge VALUES ('$userEmail', '$day')";
        $this->util->fetchAndExecute($insertDay);
        return true;
    }else{
        return false;
    }
}
```

Questo metodo riceve come parametro il la data del giorno da aggiungere e la e-mail del Gestore che vuole aggiungere questo giorno. In seguito viene verificata la data da aggiungere lato server. Il controllo è uguale a quello lato client con l'aggiunta che verifica anche che il giorno non sia già stato aggiunto. Se il controllo ha esito positivo il giorno viene aggiunto nel database.

Esiste un ulteriore bottone sotto alla tabella dei giorni, il bottone si chiama “Elimina tutti”. Questo metodo di occupa di eliminare tutti i giorni di lavoro fino ad oggi. Prima di procedere richiedere la conferma tramite un alert:

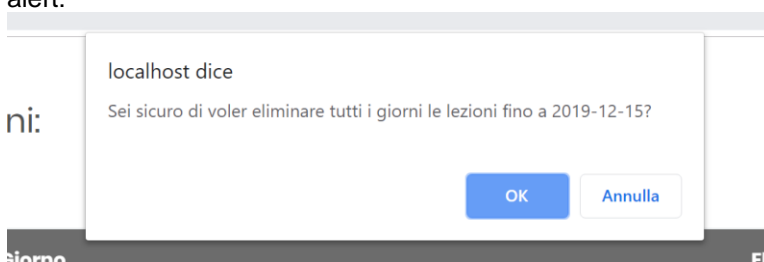


Figura 56 Pannello admin - elimina tutti i giorni

Se la risposta è “OK” viene richiamato il metodo deleteDays() del controller “PannelloAdmin” che a sua volta richiama il metodo removeAllDay() del model “PannelloAdminModel”:

```
function removeAllDay(){
    $selectIdEvents = "SELECT id from lezione WHERE DATE(giorno) < DATE(NOW())";
    $ids = $this->util->fetchAndExecute($selectIdEvents);
    foreach ($ids as $id){
        $query = "DELETE from assegna WHERE id_lezione='".$id[DB_LESSON_ID]."'";
        $this->util->fetchAndExecute($query);
        $query = "DELETE from lezione WHERE id='".$id[DB_LESSON_ID]."'";
        $this->util->fetchAndExecute($query);
    }
    $deleteAggiunge = "DELETE FROM aggiunge WHERE DATE(giorno) < DATE(NOW())";
    $this->util->fetchAndExecute($deleteAggiunge);
    $deleteDay = "DELETE FROM giorno WHERE DATE(giorno) < DATE(NOW())";
    $this->util->fetchAndExecute($deleteDay);
}
```

Questo metodo seleziona inizialmente tutte lezioni avvenute prima della giornata odierna. Una volta selezionate una ad una viene eliminata per prima la correlazione nella tabella assegna per poi essere eliminata la lezione stessa. Una volta eliminate tutte le lezioni prima della giornata odierna vengono eliminate tutte le correlazioni nella tabella aggiunge ai giorni prima di oggi per poi infine eliminare i giorni stessi.

4.8 Ulteriori funzioni dell'applicazione

Esistono due ulteriori bottoni all'inizio delle pagine dopo il login. I due bottoni sono i seguenti:



Figura 57 Bottoni di navigazione

La “casetta”, se premuta, riporta l'utente alla pagina principale, dunque calendario.

La “freccia verso destra” effettuerà il logout. Quando viene premuta richiama viene distrutta la sessione per poi essere ricreata. L'utente verrà trasportato su questa pagina finale dove gli verrà detto che se vuole ritornare ad utilizzare l'applicazione dovrà effettuare nuovamente il login:

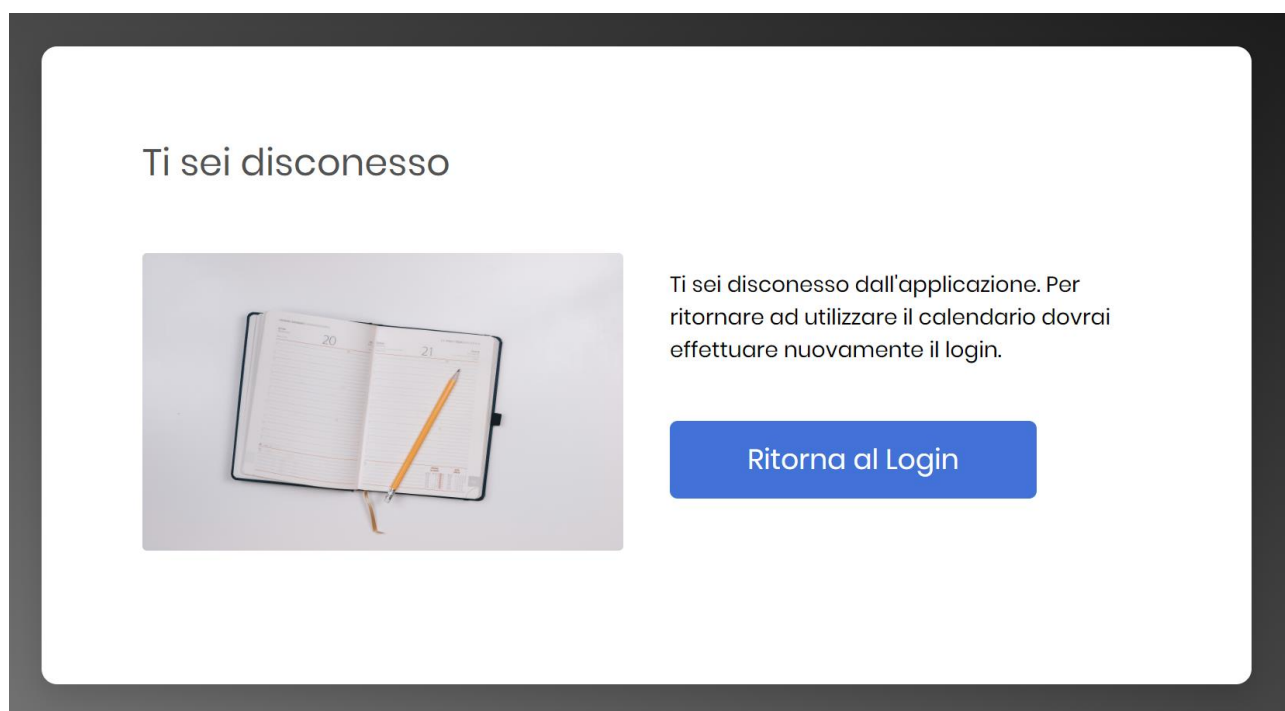


Figura 58 Pagina di logout

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	Verifica se l'applicazione è responsive
Riferimento:	REQ-001		
Descrizione:	Test per verificare se l'applicazione è responsive. Si verifica se in base allo schermo disponibile per l'applicazione, essa si adatta.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze 2. Effettuare il login con i propri dati di accesso 3. Tasto destro sulla pagina 4. Ispeziona elemento 5. In alto a sinistra selezionare un dispositivo mobile (es: iPhone X) 6. Visualizzare il risultato della pagina 		
Risultati attesi:	La pagina si adatta in base allo spazio dello schermo.		

Test Case:	TC-002	Nome:	Verifica pagina di registrazione
Riferimento:	REQ-002		
Descrizione:	Test per verificare la registrazione di un utente.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze 2. Cliccare il bottone "Registrati" 3. Compilare i campi nel seguente modo: <ul style="list-style-type: none"> • Nome: Mattia • Cognome: Toscanelli • Numero di telefono: 098 765 43 21 • Email: mattia.toscanelli@yopmail.com • Password: Test1234 • Re-Password: Test1234 4. Cliccare il bottone registrati. 5. Aprire il Prompt dei comandi. 6. Accedere a MySQL con il comando "mysql -u root -p" 7. Inserire la password: root 8. Utilizzare il database con il comando "use gestione_vacanze" 9. Verificare che l'utente sia stato inserito nel database con il comando: <pre>"SELECT * from utente WHERE e-mail='mattia.toscanelli@yopmail.com'" </pre> 		
Risultati attesi:	Un risultato da parte della query della tabella utente.		

Test Case:	TC-003	Nome:	Verificare che la pagina di login funzioni.
Riferimento:	REQ-003		
Descrizione:	Test per verificare se la pagina di login funziona.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze 2. Effettuare il login con i seguenti dati di accesso: <ul style="list-style-type: none"> • E-mail: mattia.toscanelli@yopmail.com • Password: Test1234 3. Verificare che dopo aver effettuato il login si sia aperta la pagina principale con il calendario 		
Risultati attesi:	La pagina principale si è aperta.		

Test Case:	TC-004	Nome:	Verifica delle funzionalità di un docente.
Riferimento:	REQ-004		
Descrizione:	Test per verificare se un docente può aggiungere una lezioni al calendario.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze 2. Effettuare il login con un account di tipo Docente 3. Trovare un settore verde nel calendario 4. Selezionare un determinato intervallo di ore nel calendario 5. Inserire un nome alla lezione (es: Matematica) 6. Premere "OK" 7. Verificare che la lezione è stata aggiunta al calendario 		
Risultati attesi:	La lezione è stata inserita correttamente nel calendario.		

Test Case:	TC-005	Nome:	Verifica delle funzionalità di un visualizzatore.
Riferimento:	REQ-005		
Descrizione:	Test per verificare se un docente può aggiungere una lezioni al calendario.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze 2. Effettuare il login con un account di tipo Visualizzatore 3. Trovare un settore verde nel calendario 4. Selezionare un determinato intervallo di ore nel calendario 5. Verificare che il visualizzatore non può aggiungere una lezione 		
Risultati attesi:	La lezione non è stata inserita nel calendario.		

Test Case:	TC-006	Nome:	Verifica della modifica degli utenti nella pagina di amministrazione
Riferimento:	REQ-001		
Descrizione:	Test per verificare la modifica di un utente nella pagina di amministrazione.		
Prerequisiti:	Google Chrome installato sulla propria macchina.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il Google Chrome e inserire l'URL http://localhost/GestioneVacanze. 2. Effettuare il login con un account di tipo Gestore 3. Aprire la pagina di amministrazione cliccando il bottone rosso "Pannello Admin" 4. Cliccare la matita di fianco ad un utente 5. Modificare il nome con "Luigi" 6. Cliccare il bottone "Modifica" 7. Verificare che il nome dell'utente sia diventato "Luigi" 		
Risultati attesi:	Il nome dell'utente è stato modificato con "Luigi".		

5.2 Risultati test

Questa è la tabella dei risultati dei test eseguiti:

Codice Test	Risultato Test
TC-001	Riuscito
TC-002	Riuscito
TC-003	Riuscito
TC-004	Riuscito
TC-005	Riuscito
TC-005	Riuscito
TC-006	Riuscito

Notando la tabella si può dire che il progetto rispetta tutti i requisiti.

5.3 Mancanze/limitazioni conosciute

Il progetto è stato sviluppato e testato sul browser Google Chrome. L'applicazione non funziona se si va ad utilizzare Internet Explorer. Infatti esso, essendo obsoleto, non supporta le classi di JavaScript e quindi non permette la validazione dei dati tramite client. Inoltre è presente un'ulteriore limitazione quando si utilizza l'applicazione da telefono. Infatti i docenti non possono ridimensionare o spostare le proprie lezioni dal calendario tramite telefono. Quindi se si vuole utilizzare il calendario nei migliori dei modi bisogna fare girare l'applicazione su un dispositivo che dispone di un mouse.

6 Consuntivo

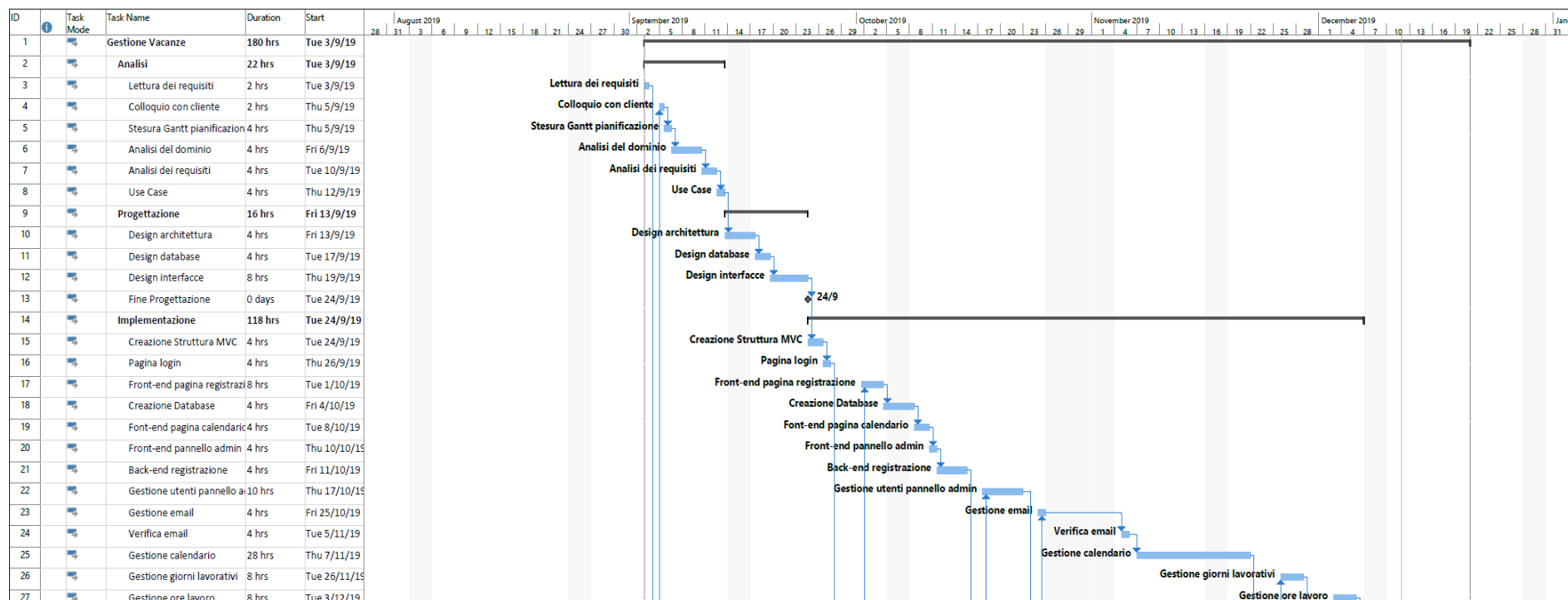


Figura 59 Gantt consuntivo - prima parte



Infine la fase di Documentazione è stata la parte che non ha risentito di alcun cambiamento. L'attività documentazione non è proprio corretto che è stata eseguita tutti i giorni ma comunque è stata quasi sempre aggiornata.

7 Conclusioni

Grazie questo progetto il mio cliente potrà gestire le lezioni al di fuori delle ore scolastiche. Sicuramente questo progetto non è stata una perdita di tempo in quanto ho imparato molte cose che mi saranno utili nella mia vita da programmatore. Il prodotto è facile e intuitivo, adatto a qualsiasi tipo di utente, che sia un esperto di informatica sia all'utente meno tecnologico di questo pianeta. Questo progetto garantisce sicuramente più ordine ai Gestori, in quanto gli è permesso vedere un quadro generale delle attività dei Docenti, e più ordine tra le relazioni fra docenti, in quanto tutte le ore di lezioni sono stabilite su un singolo calendario condiviso. Infine fa risparmiare molto probabilmente tanto tempo questo calendario perché prima queste attività dovevano essere gestite da una persona, invece ora il meccanismo è tutto automatizzato.

7.1 Sviluppi futuri

In futuro si possono aggiungere diverse funzioni. Una tra queste potrebbe essere che quando un Docente dovesse finire le proprie ore di lavoro non si possono aggiungere più altre lezioni, oppure si potrebbe inviare una mail al Docente che ha finito le ore per renderlo attento a ciò che sta facendo.

Un'altra interessante aggiunta potrebbe essere l'aggiunta dei giorni di lavoro con degli intervalli anziché il bisogno di aggiungere i giorni uno ad uno.

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

7.2 Considerazioni personali

Come già detto precedentemente ho imparato molte cose che mi saranno utili nella mia vita da programmatore. La prima tra queste è sicuramente la gestione di un calendario JavaScript che carica i dati da un database e li elabora lato server con PHP. Grazie a questo progetto ho finalmente imparato il concetto di creare un progetto con la struttura MVC, e questo mi è utile anche per altre materie. Inoltre ho approfondito la mia conoscenza del linguaggio di PHP, in quanto prima di iniziare questo progetto non sapevo interagire con Database, utilizzare ajax per il passaggio di dati client-server, e molto altro, In conclusione posso dire che sono molto soddisfatto del lavoro che ho svolto.

8 Bibliografia

8.1 Sitografia

- <https://www.draw.io/>, *Schemi*, 09-10-2019.
- <https://mockflow.com/>, Mockup interfacce, 19-10-2019
- <https://colorlib.com/>, Template sito web, 24-09-2019
- <https://fontawesome.com/v4.7.0/>, Font e icon, 24-09-2019
- <https://www.w3schools.com/>, Guide e Tutorial web, Ultima visita 12.12.2019
- <https://notifyjs.jpillora.com/>, Libreria notify.js (notifiche), 01-10-2019
- <https://www.mysql.com/it/>, Struttura Database, 03-10-2019
- <https://www.php.net/>, Guide e Tutorial PHP, Utilizzato frequentemente fino al 12.12.2019
- <https://fullcalendar.io/>, Calendar JS, 10-10-2019
- <https://getbootstrap.com/docs/4.4/content/tables/>, Tabella Bootstrap, 10-10-2019
- <https://github.com/PHPMailer/PHPMailer>, PHPMailer (invio e-mail), 22-20-2019
- <https://www.webslesson.info/2017/12/jquery-fullcalendar-integration-with-php-and-mysql.html>, Guida FullCalendar con Database, 12-11-2019
- <https://stackoverflow.com/>, Soluzioni a problemi riscontrati, Ultima visita 12.12.2019
- <http://www.yopmail.com/it/>, WebMail per testare il progetto, Ultima visita 12.12.2019

8.2 Indice delle figure

Figura 1 Use Case.....	7
Figura 2 Gantt Preventivo Intero	8
Figura 3 Gantt preventivo - Analisi	9
Figura 4 Gantt preventivo - Progettazione.....	10
Figura 5 Gantt preventivo - Implementazione	11
Figura 6 Gantt preventivo - Testing	12
Figura 7 Gantt preventivo - Documentazione.....	13
Figura 8 Design dell'architettura del sistema.....	15
Figura 9 Schema E-R	16
Figura 10 Design - Pagina di login	17
Figura 11 Design - Pagina di registrazione	17
Figura 12 Design - Pagina recupero password	18
Figura 13 Design - Pagina reimposta password	18
Figura 14 Design - Pagina calendario Docente.....	19
Figura 15 Design - Pagina calendario Gestore	19
Figura 16 Design - Pagina calendario Visualizzatore.....	20
Figura 17 Design - Pagina calendario in formato testuale	20
Figura 18 Design - Pagina pannello admin	21
Figura 19 Design - Pagina di stampa	21
Figura 20 UML - Classe Database	22
Figura 21 UML - Classe SendMail.....	23
Figura 22 UML - Classe Util	24
Figura 23 UML - Classe Validator	25
Figura 24 Struttura MVC.....	26
Figura 25 Pagina di login	29
Figura 26 Notifica di errore	30
Figura 27 Pagina attendi.....	30
Figura 28 Pagina di registrazione	31
Figura 29 Messaggio di errore pagina di registrazione	33
Figura 30 Pagina di errore	34
Figura 31 Verifica e-mail.....	35
Figura 32 Pagina attendi.....	36
Figura 33 Pagina password dimenticata	37
Figura 34 Pagina e-mail inviata	37
Figura 35 E-mail modifica password	38

Figura 36 Pagina cambia password	39
Figura 37 Pagina Password modificata	39
Figura 38 Visualizzazione calendario per Gestore e Visualizzatore	40
Figura 39 Visualizzazione calendario in formato testuale	40
Figura 40 Evento cliccato per Gestore e Visualizzatore	42
Figura 41 Visualizzazione calendario per Docenti	43
Figura 42 Aggiunta di lezioni	44
Figura 43 Spostamento di una lezione	47
Figura 44 Ridimensionamento lezione	48
Figura 45 Eliminazione lezione	50
Figura 46 Calendario personale	51
Figura 47 Bottone per stampare il calendario	52
Figura 48 Visualizzazione stampa calendario	52
Figura 49 Bottone pannello admin	53
Figura 50 Pannello admin - tabella utenti	54
Figura 51 Pannello admin - modifica utente	54
Figura 52 Pannello admin - eliminazione utente	56
Figura 53 Pannello admin - reimposta ore di lavoro per docenti	57
Figura 54 Pannello admin - tabella dei giorni lavorativi	58
Figura 55 Pannello admin - eliminazione giorno	58
Figura 56 Pannello admin - elimina tutti i giorni	59
Figura 57 Bottoni di navigazione	61
Figura 58 Pagina di logout	61
Figura 59 Gantt consuntivo - prima parte	65
Figura 60 Gantt Consuntivo - seconda parte	66

9 Allegati

- Diari di lavoro
- Mandato e/o Qdc
- Manuale di installazione del prodotto
- Codice sorgente trovabile su GitHub:
<https://github.com/MattiaToscanelli/GestioneVacanze/tree/master/Codice/GestioneVacanze>
- Database