Gestione Vacanze

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 9

2.4.3 Implementazione 10

2.4.4 Testing 11

2.4.5 Documentazione 12

2.5 Analisi dei mezzi 13

2.5.1 Software 13

2.5.2 Hardware 13

3 Progettazione 14

3.1 Design dell’architettura del sistema 14

3.2 Design dei dati e database 15

3.2.1 Schema E-R 15

3.2.2 Schema logico: 15

3.2.3 Vincoli 15

3.3 Design delle interfacce 16

3.3.1 Pagina di login 16

3.3.2 Pagina di registrazione 16

3.3.3 Pagina richiesta nuova password 17

3.3.4 Pagina reimposta password 17

3.3.5 Pagina principale con il calendario 18

3.3.6 Pagina amministrazione 20

3.3.7 Pagina di stampa 20

3.4 Diagramma delle classi 21

3.4.1 PHP 21

4 Implementazione 24

5 Test 26

5.1 Protocollo di test 26

5.2 Risultati test 26

5.3 Mancanze/limitazioni conosciute 26

6 Consuntivo 26

7 Conclusioni 26

7.1 Sviluppi futuri 27

7.2 Considerazioni personali 27

8 Bibliografia 27

8.1 Bibliografia per articoli di riviste: 27

8.2 Bibliografia per libri 27

8.3 Sitografia 27

9 Allegati 27

# Introduzione

## Informazioni sul progetto

**Titolo**: Sistema didattico per Arduino con libreria per attuatori e relativa documentazione

**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

## Abstract

E’ una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## 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.

# Analisi

## 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.

## 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. |

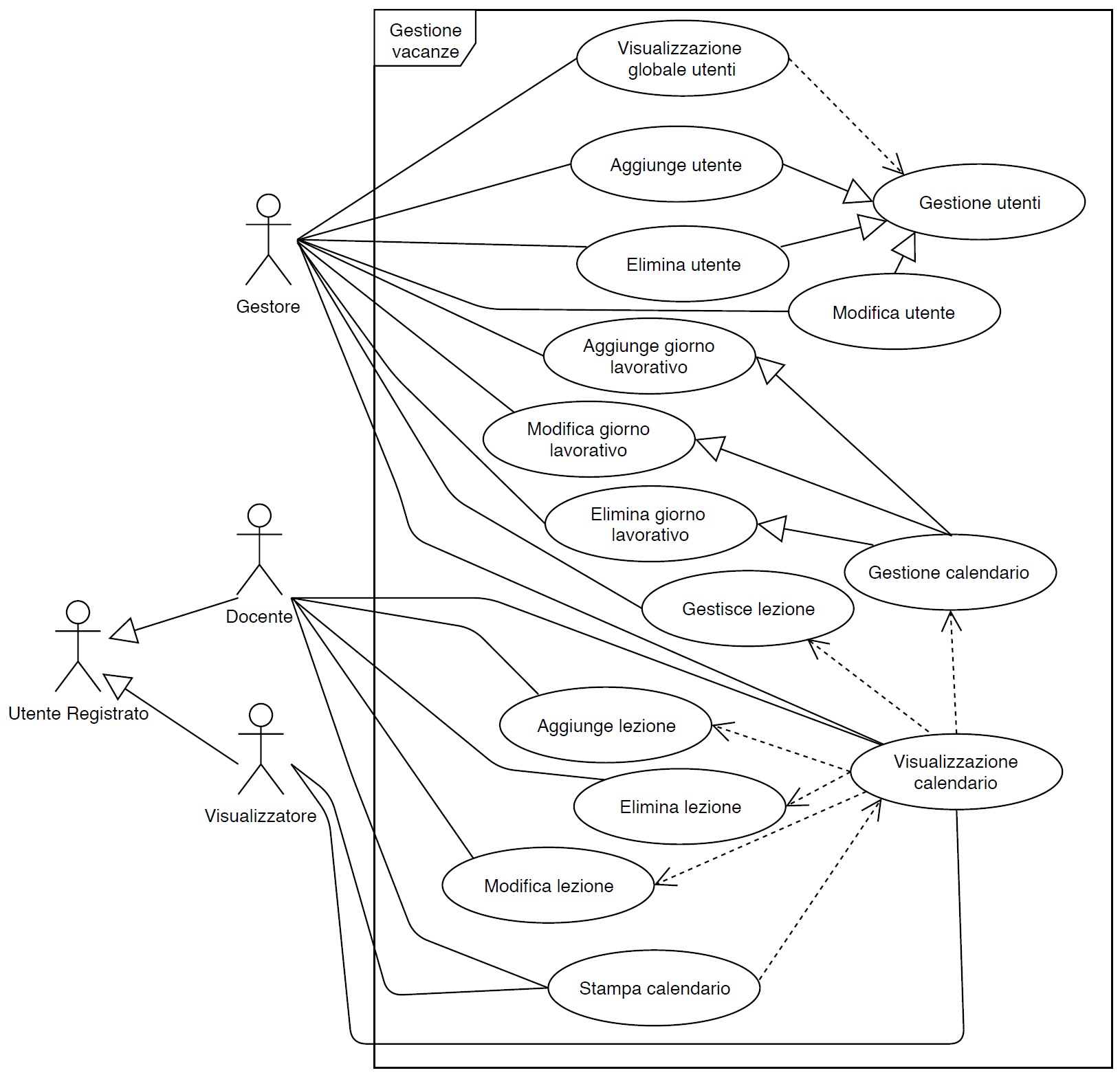
|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Gestione degli utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Ci sono tre tipologie di utenti: Gestore, Docente e Visualizzatore |
| **002** | Per ogni utente di registrano i seguenti 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 potrenno 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 3 ore. Questa deve essere definita dalle 08:00 fino alle 16:00. |
| **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. |

## Use case

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



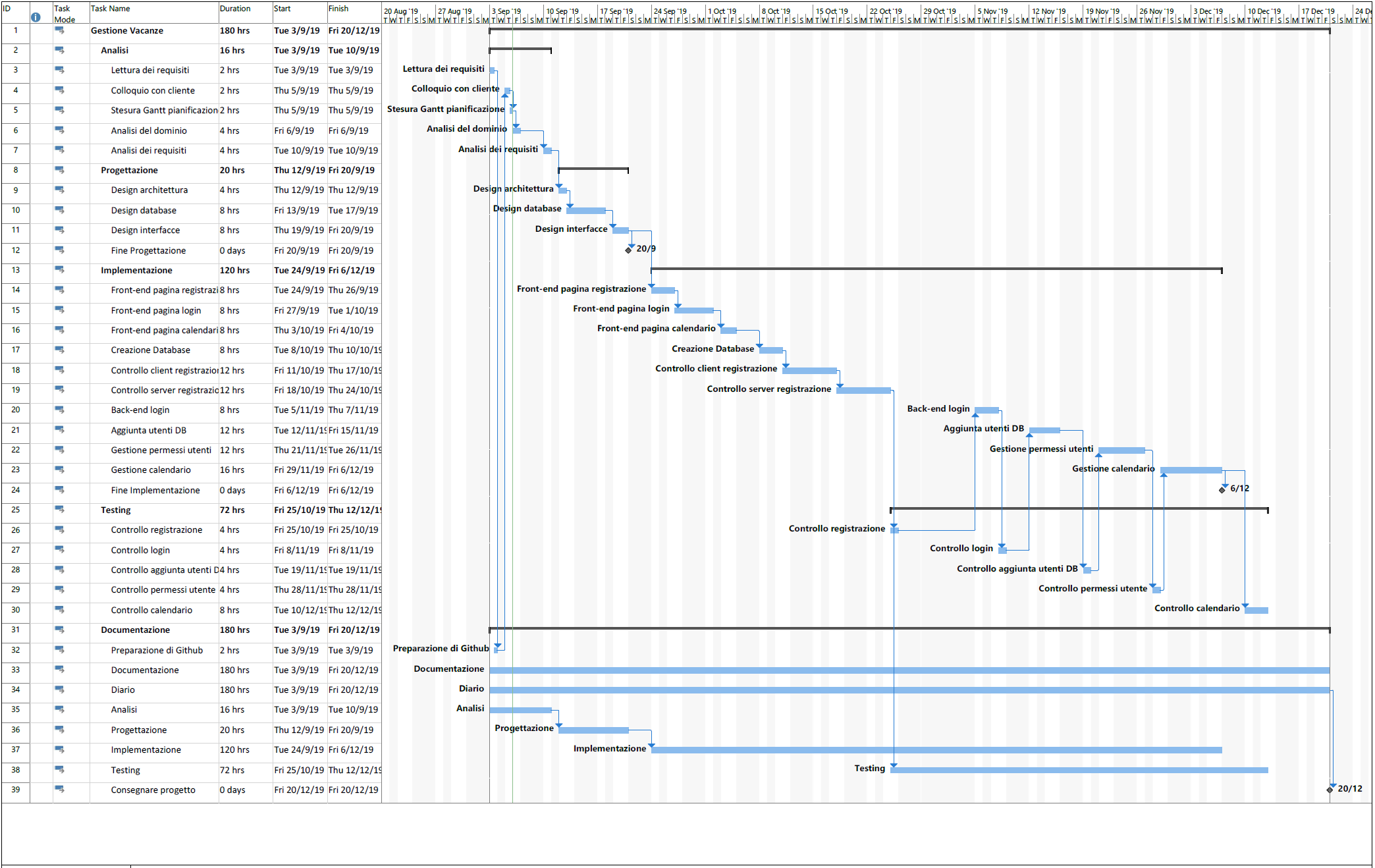
Lo schema inizia con tre tipologie di scenari: Gestore (cioè il capo dell’applicazione), Docente e Visualizzatore. Per quanto riguarda i due scenari Docente e Visualizzatore è necessaria una registrazione, la quale verrà valutata dal Gestore se essere accettata oppure no.

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 lavoravo.

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.

## Pianificazione

.

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

### Analisi

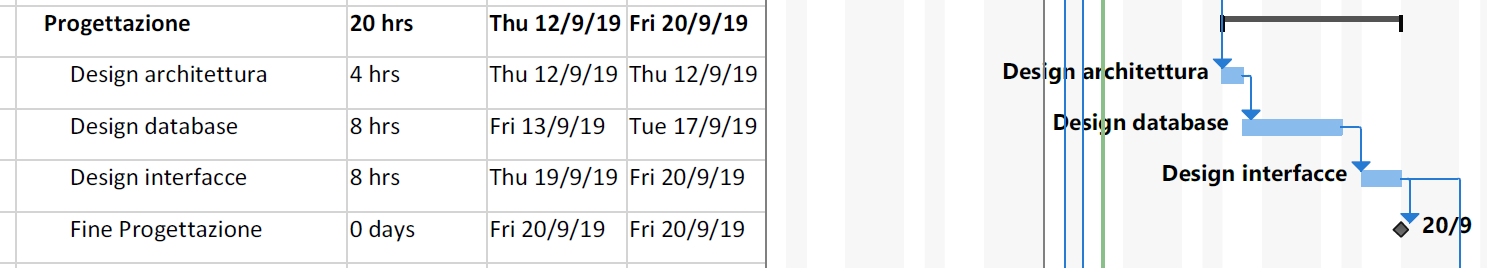


L’analisi si suddivide in 5 lavori:

* 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 Gannt 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.

### Progettazione

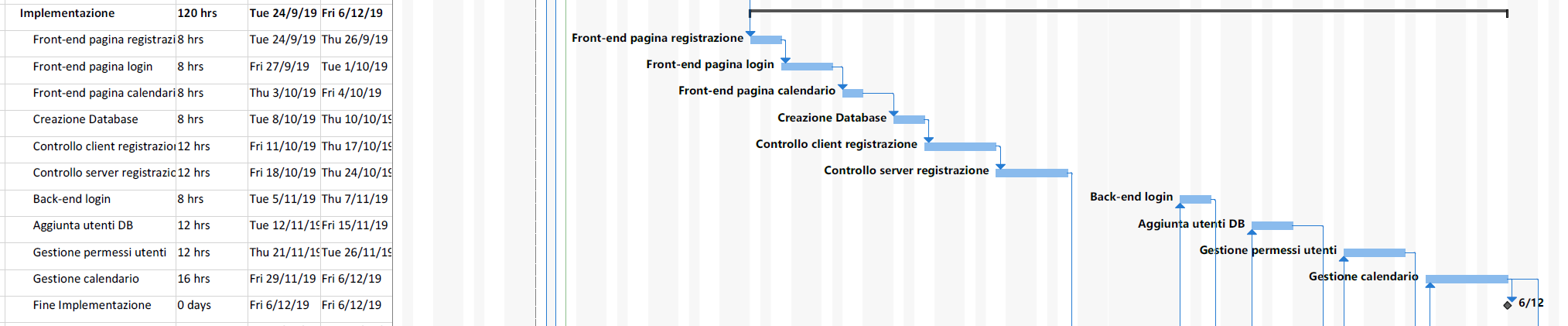


La progettazione si suddivide in 3 lavori e un punto cardine:

* 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.

### 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
* Aggiutna 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.

### Testing

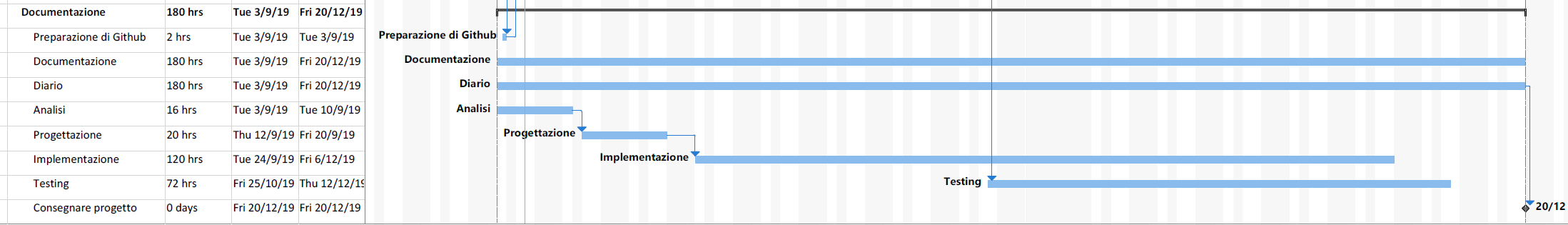


La fase di testing si suddivide in 5 lavori:

* Controllo registrazione 🡪 controllo che tutti i campi della registrazione vangano 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.

### 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

## Analisi dei mezzi

Per la realizzione di questo progetto si ha bisogno di:

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

### Software

I programmi utilizzati per svolgere questo progetto sono:

* Word 🡪 per la documentazioni
* PowerPoint 🡪 per la presentazione
* Draw.io 🡪 per i disegni dei vari schemi e progettazioni
* PhpStorm 🡪 per lo sviluppo web
* Apache 🡪 per il webserver
* PHP 🡪 per l’utilizzo del linguaggio php

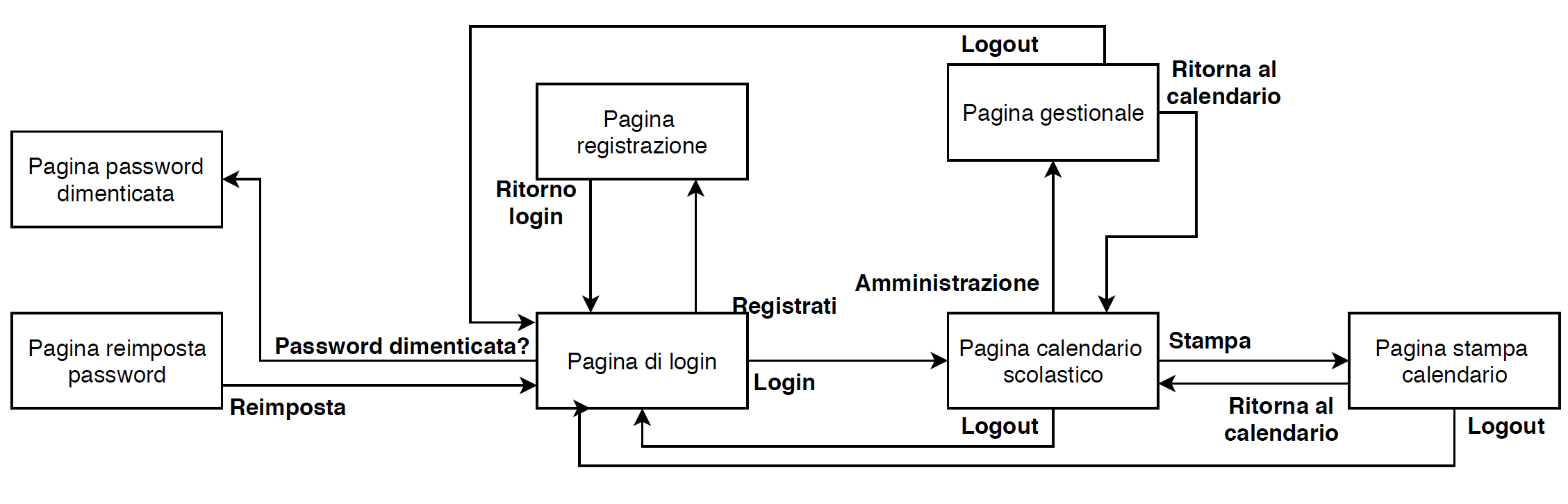
### Hardware

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

# Progettazione

## Design dell’architettura del sistema

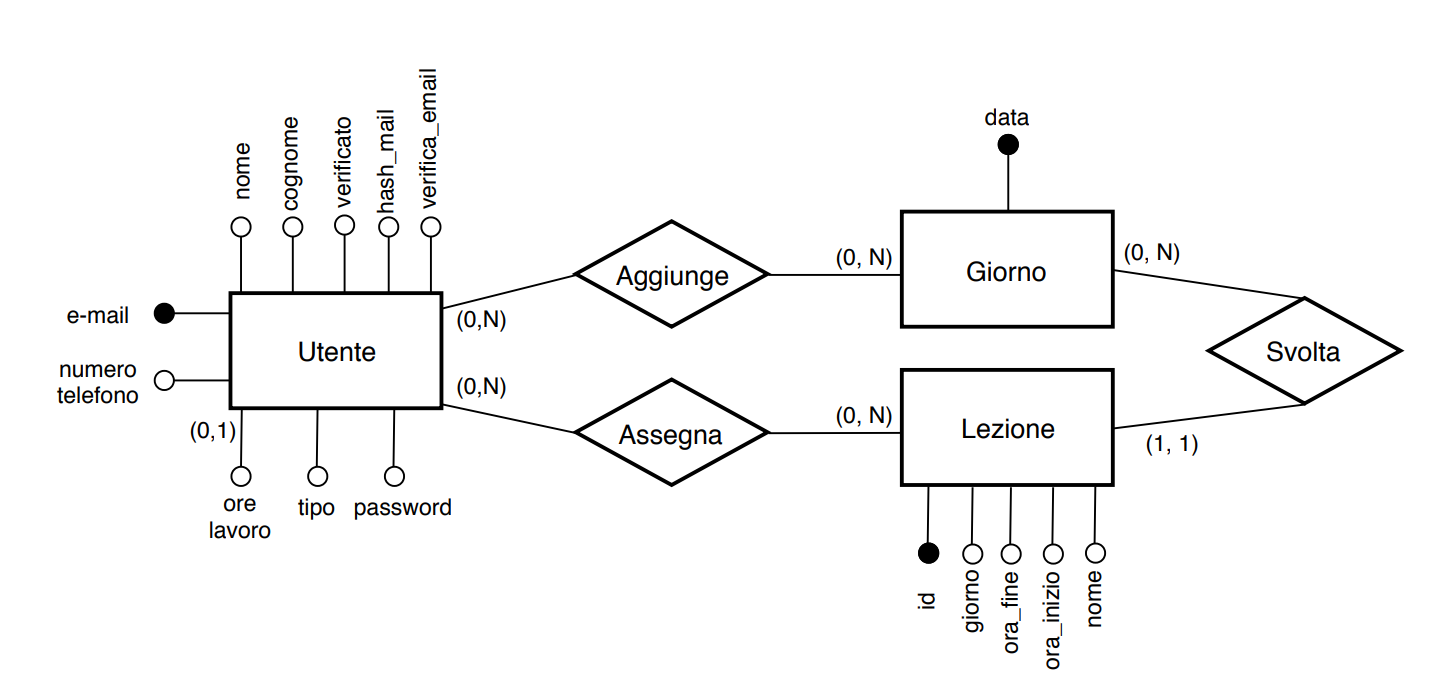


La pagina di partenza della mia applicazione, cioè la pagina iniziale a cui si collega l’utente finale per utilizzare il mio sito, è la pagina di login. Se l’utente non è ancora registrato alla pagina può premere il pulsante “Registrati” e verrà re-indirizzato alla pagina di registrazione. A questo punto 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](#_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 effetturare un logout premendo appunto il pulsante “Logout” oppure ritornare alla pagina principale premendo il “Ritorna al calendario”. Ritornando alla pagina di visualizzazione del calendatrio, se l’utente che ha effettuato l’accesso è il Gestore esso potrà accedere alla una pagina gestionale attraverso il pulsaente “Amministrazione”. 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 effetturare un logout premendo appunto il pulsante “Logout” oppure ritornare alla pagina dove viene mostrato il calendario premendo il “Ritorna al calendario”.

## Design dei dati e database

### Schema E-R



### 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))

### Vincoli

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

* Il campo Tipo può contente solo i valori Visualizzatore, Docente o Gestore.
  + Se Tipo = Visualizzatore, non può partecipare alla relazione Aggiunge ne alla relazione Aggiunge. Inoltre il campo ore\_lavoro non deve essere definito.
  + Se Tipo = Docente, va definito il campo ore\_lavoro e può partecipare la relazione Assegna ma non ha quella Aggiunge.
  + Se Tipo = Gestore, può partecipare alla relazione Aggiunge ma non alla relazione Assegna. Inoltre il camp 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.

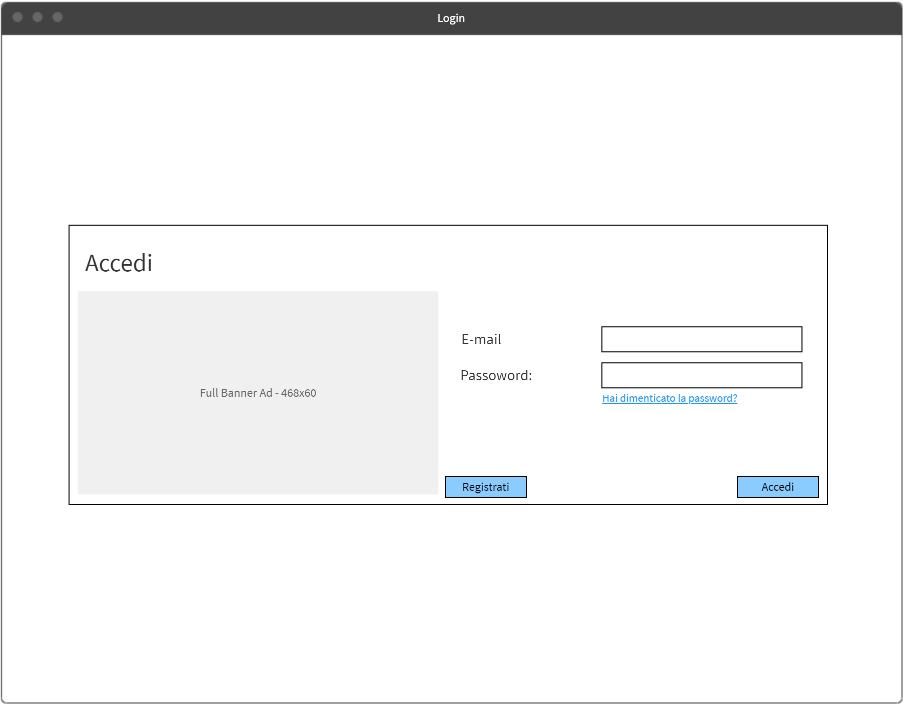
Per quanto riguarda i dati ci sono delle piccole regole da gestire:

* 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).

## Design delle interfacce

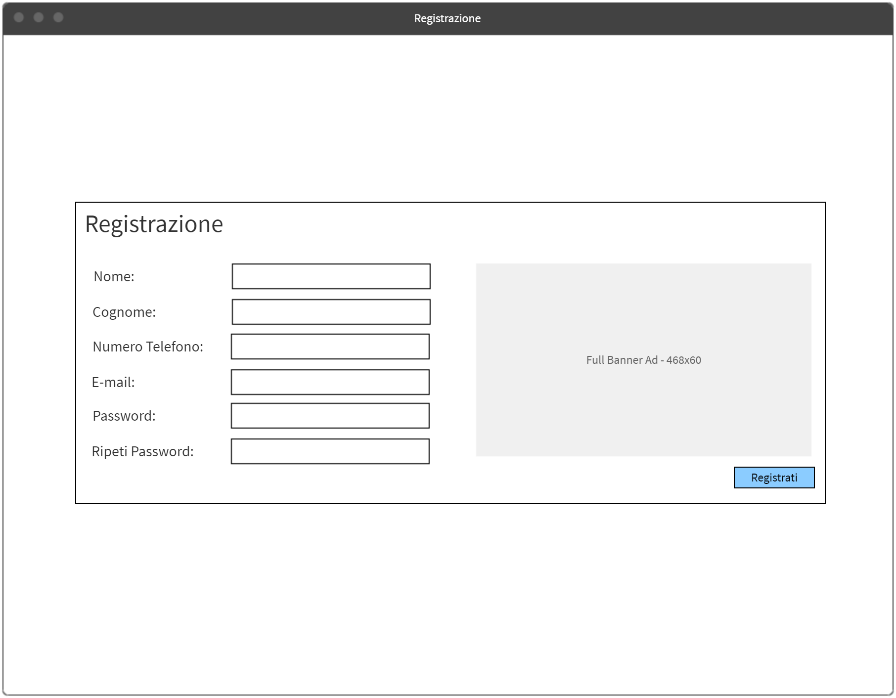
In questo capitoletto vengono mostrate delle anteprime delle varie interfacce delle pagine.

### 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.

### Pagina di registrazione

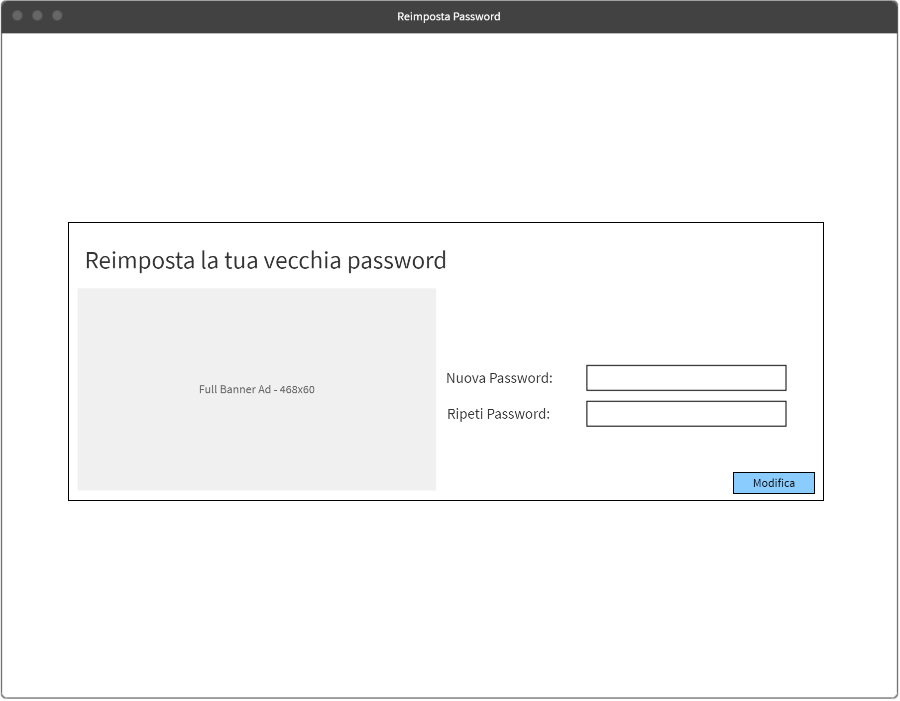


Questa è la pagina di registrazione. La pagina è composta da un form da 6 campi. L’utente finale dovrà compilare tutti questi campi con i suoi dati personali e in modo corretto. Una volta completata la compilazione gli basterà cliccare il bottone “Registrati” per inviare la richiesta di registrazione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

### 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.

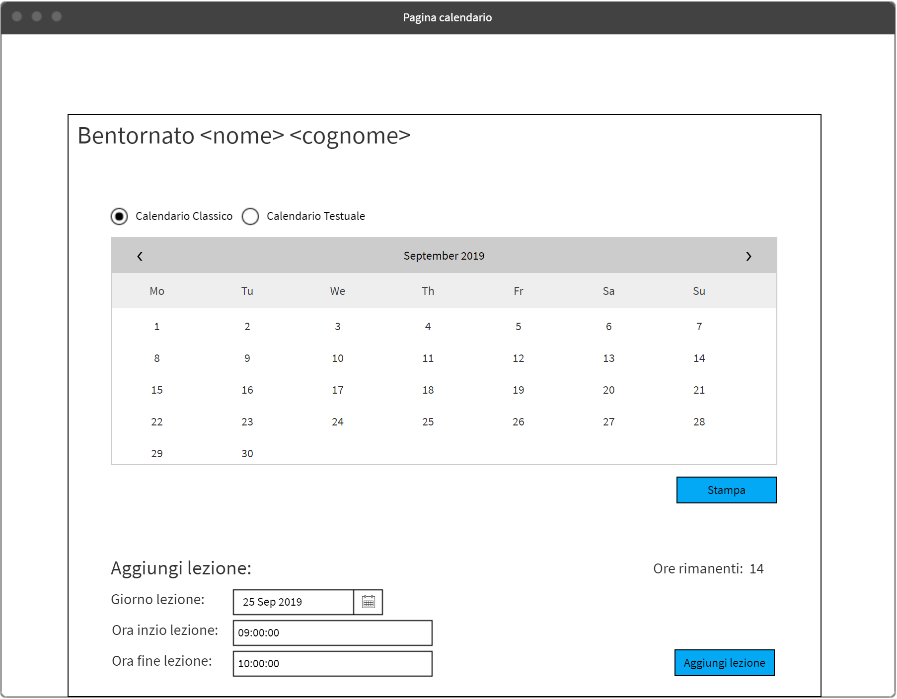
### Pagina reimposta password



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”.

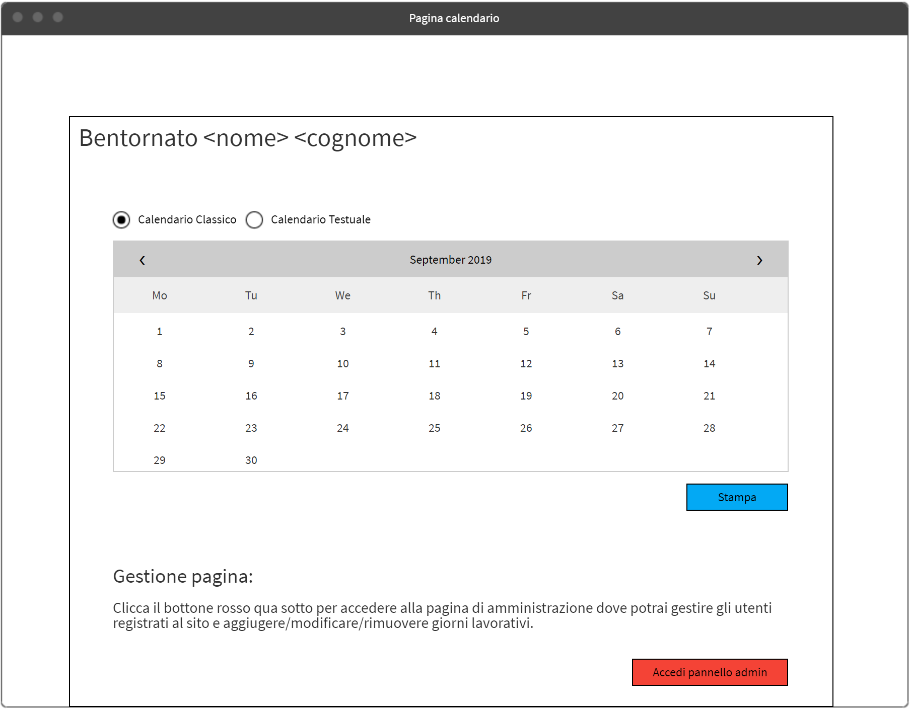
### Pagina principale con il calendario

#### Pagina per Docenti



Questa è la pagina principale del progetto vista da un account di tipo Docente. In questa pagina si potrà visualizzare il calendario scolatico. Il calendario si puo 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 aggiugere, modificare o elliminare le lezioni. Infine si ha la possibilità di stampare il calendario tramite il bottone “Stampa”.

#### Pagina per Gestori



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 un pagina di gestione del sito.

#### Pagina per Visualizzatori

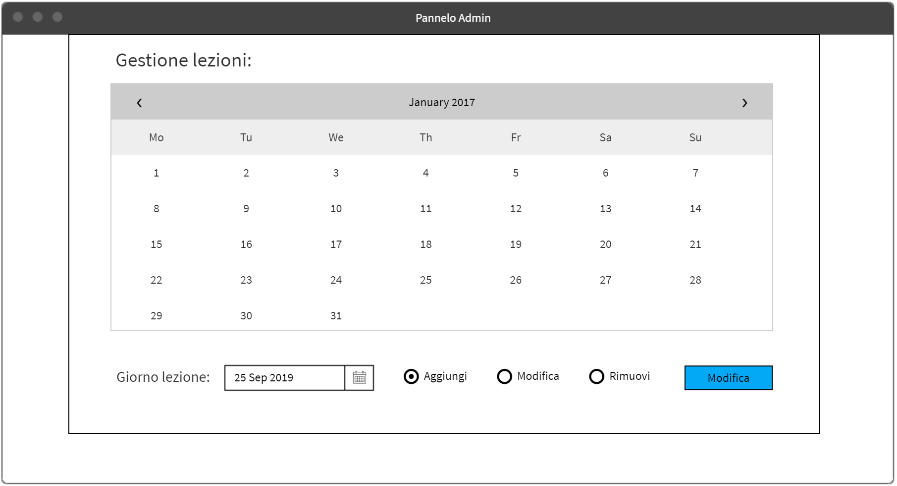
Questa è la stessa pagina mostrata sopra ma vista da uno che ha effettuato l’accesso come Visualizzatore. Infatti non ha più nessuna funzione se non quello di visualizzare il calendario o stamparlo.

#### 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.

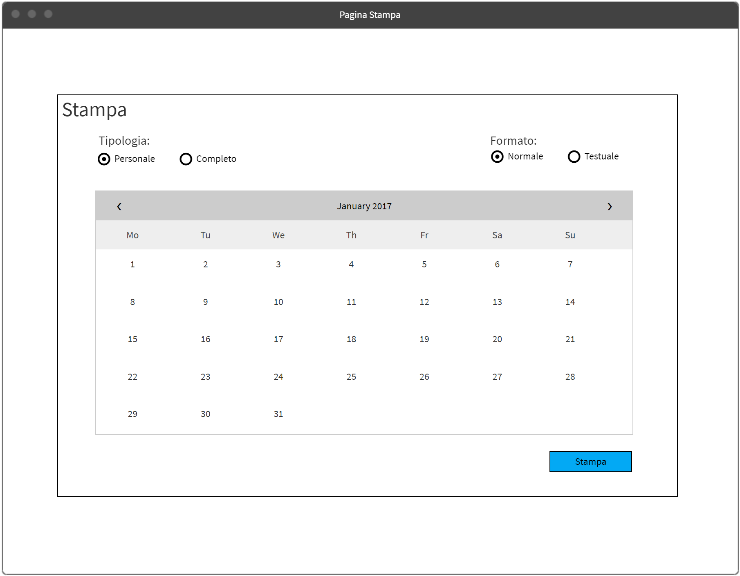
### Pagina amministrazione

Questa è la pagina di amministrazione la quale possono accederci 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 giu 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 qui è possibile fare lezione.

### Pagina di stampa

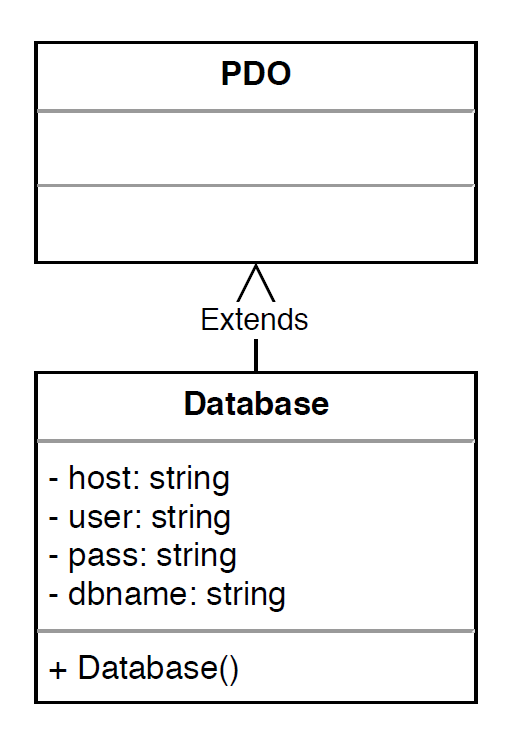


Questa è l’ultima pagina che offre il mio sito, cioè la pagina dove si può effettuare la stampa del calendario. Sopra ci sono due impostazione “Tipologia” e “Formato”. La tipologia viene mostrata solamente hai docenti e permette di stampare il calendario completo oppure solamente con le proprie ore di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

## Diagramma delle classi

### PHP

#### Classe Database



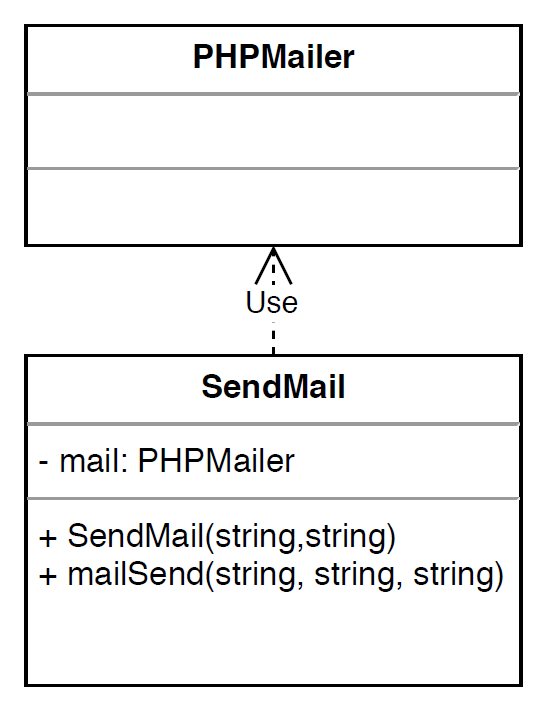
Questa classe è utilizzata per eseguire l’accesso con il database tramite PHP. Essa presenta quattro attributi privati e un metodo costruttore senza parametri.

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 sevizio 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.

#### Classe SendMail



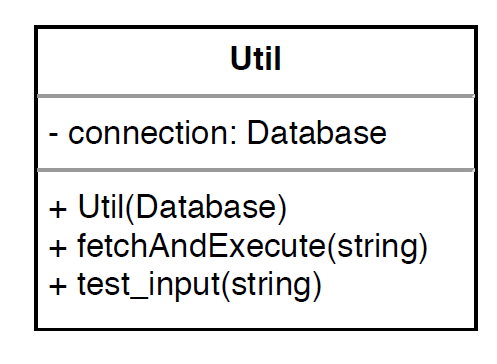
Questa classe è utilizzata per inviare le email tramite PHP. Essa presenta un attributo privato, un costruttore con 2 parametri e un metodo di utilizzo della classe.

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.

#### 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.

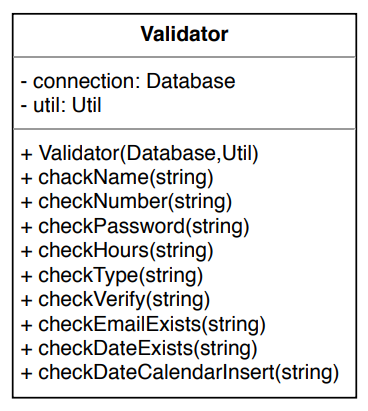
Partendo con ordine la classe ha un attributo privato, un costruttore e due metodi di utilizzo della classe.

L’attributo connection mi definisce la connessione al mio database ed esso viene passato attraverso al 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 prametro.

#### 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 due metodi di utilizzo della classe.

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

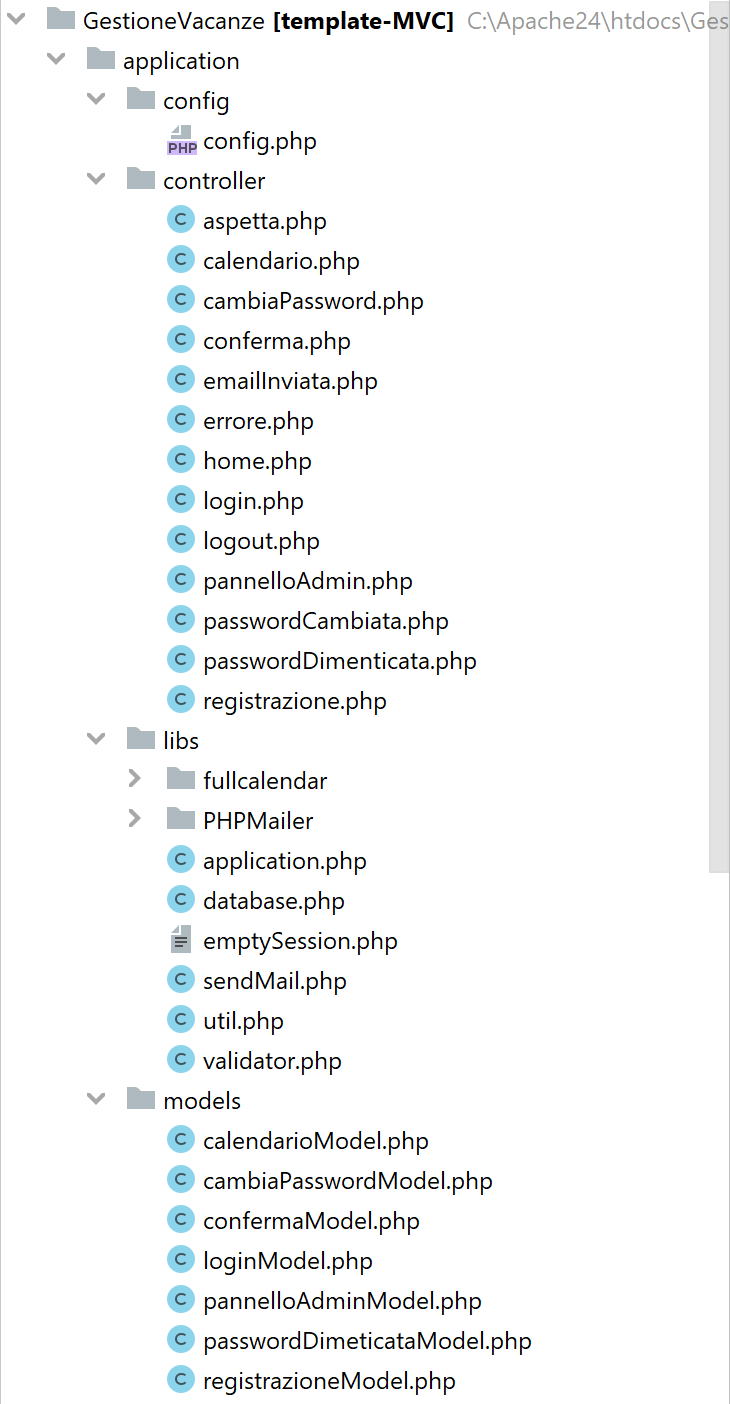
In seguito la classe presenta due metodi:

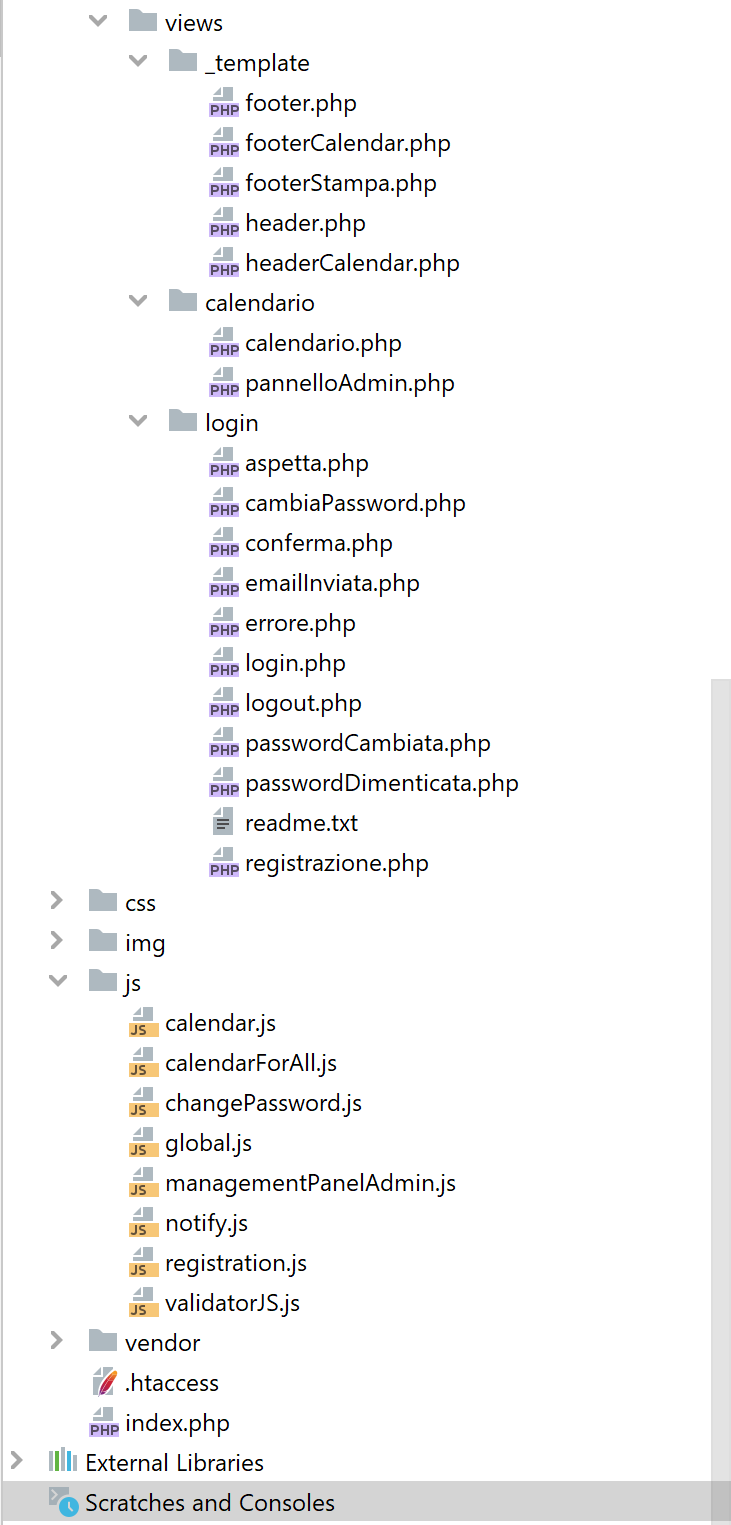
* fetchAndExecute(string) 🡪 mi permette attraverso il di passarli una query da fare sul database. Esso si occupa dunque di preparare la query in modo da evitare attacchi interni ed eseguirla al database. Inoltre in caso di una SELECT ritorna un array multidimensionale con il risultato della query.
* 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 prametro.

# Implementazione

La prima cosa che ho fatto è creare la 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:



In totale ho 13 controller che corrispondo a loro volta a 13 View.

Però, 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 int(4),

    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)

);

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …