

# **PROGETTO CPT Meeting Room Scheduler**

## **| Diario di lavoro - 03.09.2019**

**Luca Di Bello**

**Canobbio, 03.09.2019**

### **Lavori svolti**

Nella prima giornata del progetto ho letto ed analizzato il quaderno dei compiti, in seguito abbiamo stilato un Gantt molto basico ed una serie di domande da porre al professor Fabrizio Valsangiacomo, il mio supervisore professionale.

### **Problemi riscontrati e soluzioni adottate**

Dato che il professor Valsangiacomo non è disponibile le domande relative al quaderno dei compiti le invierò per email insieme ai diari.

### **Punto della situazione rispetto alla pianificazione**

Sto rispettando i tempi.

### **Programma di massima per la prossima giornata di lavoro**

Finire il diagramma di Gantt ed iniziare il capitolo relativo all'analisi del progetto.

# PROGETTO CPT Meeting Room Scheduler

## | Diario di lavoro - 05.09.2019

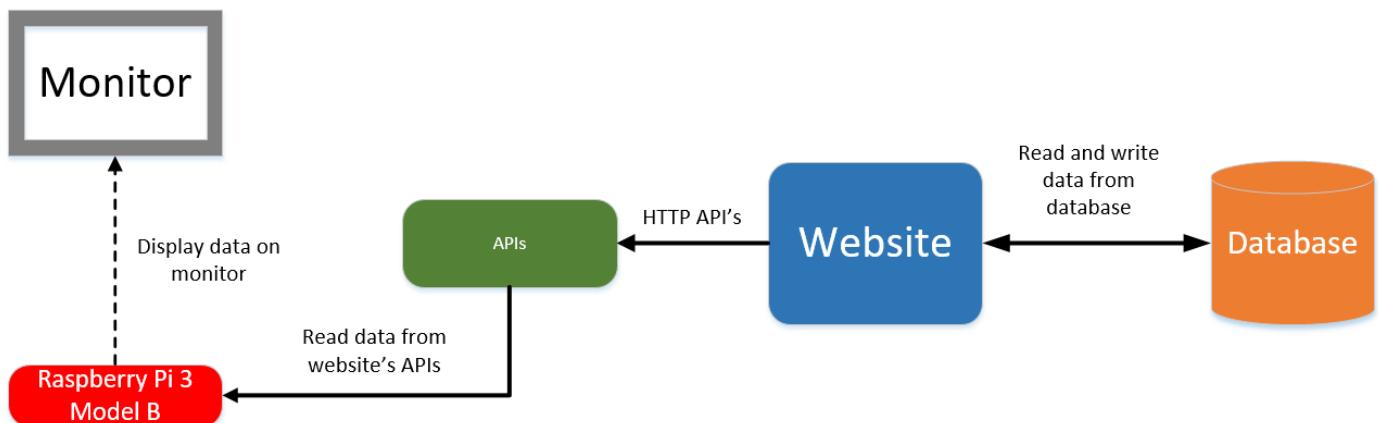
Luca Di Bello

Canobbio, 05.09.2019

### Lavori svolti

Durante la giornata di oggi ho continuato la documentazione. Ho finito l'analisi del dominio, l'analisi dei requisiti ed ho provato a fare il Gantt ma ho riscontrato diversi problemi durante l'installazione di *Microsoft Project 2016* e *Visio 2016*.

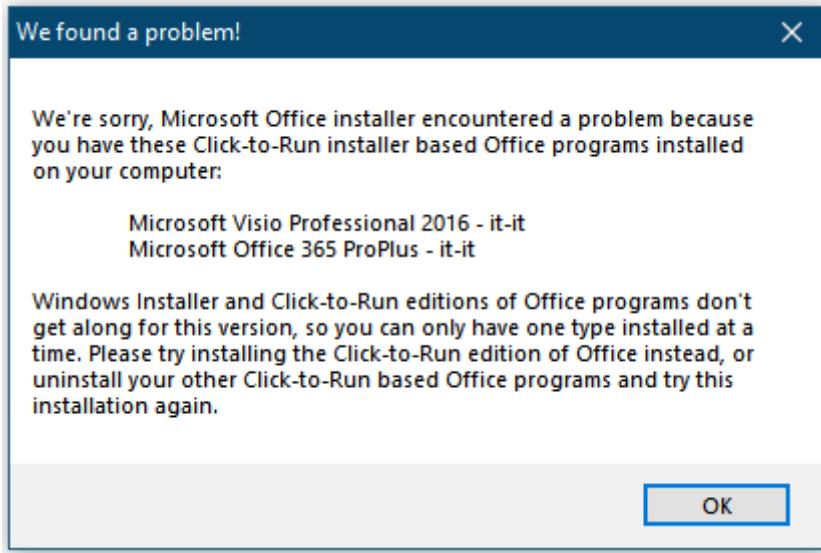
In seguito, dopo aver trovato la soluzione, ho creato un diagramma che illustra la struttura del programma utilizzando Visio:



Nell'ultima mezz'ora di lezione ho iniziato a rifare il diagramma di Gantt utilizzando *Project 2016*.

### Problemi riscontrati e soluzioni adottate

*Project 2016* e *Visio 2016* non sono supportati nel mio sistema, questo perch? ho installato nel mio sistema una versione troppo recente di Office (*Office 365*):



Per risolvere questo problema ho dovuto creare una nuova macchina virtuale Windows per poi installare i due prodotti Microsoft.

## Punto della situazione rispetto alla pianificazione

Sono al passo con i tempi prestabiliti.

## Programma di massima per la prossima giornata di lavoro

Finire definitivamente il diagramma di Gantt e continuare la progettazione.

# PROGETTO CPT Meeting Room Scheduler

## | Diario di lavoro - 06.09.2019

Luca Di Bello

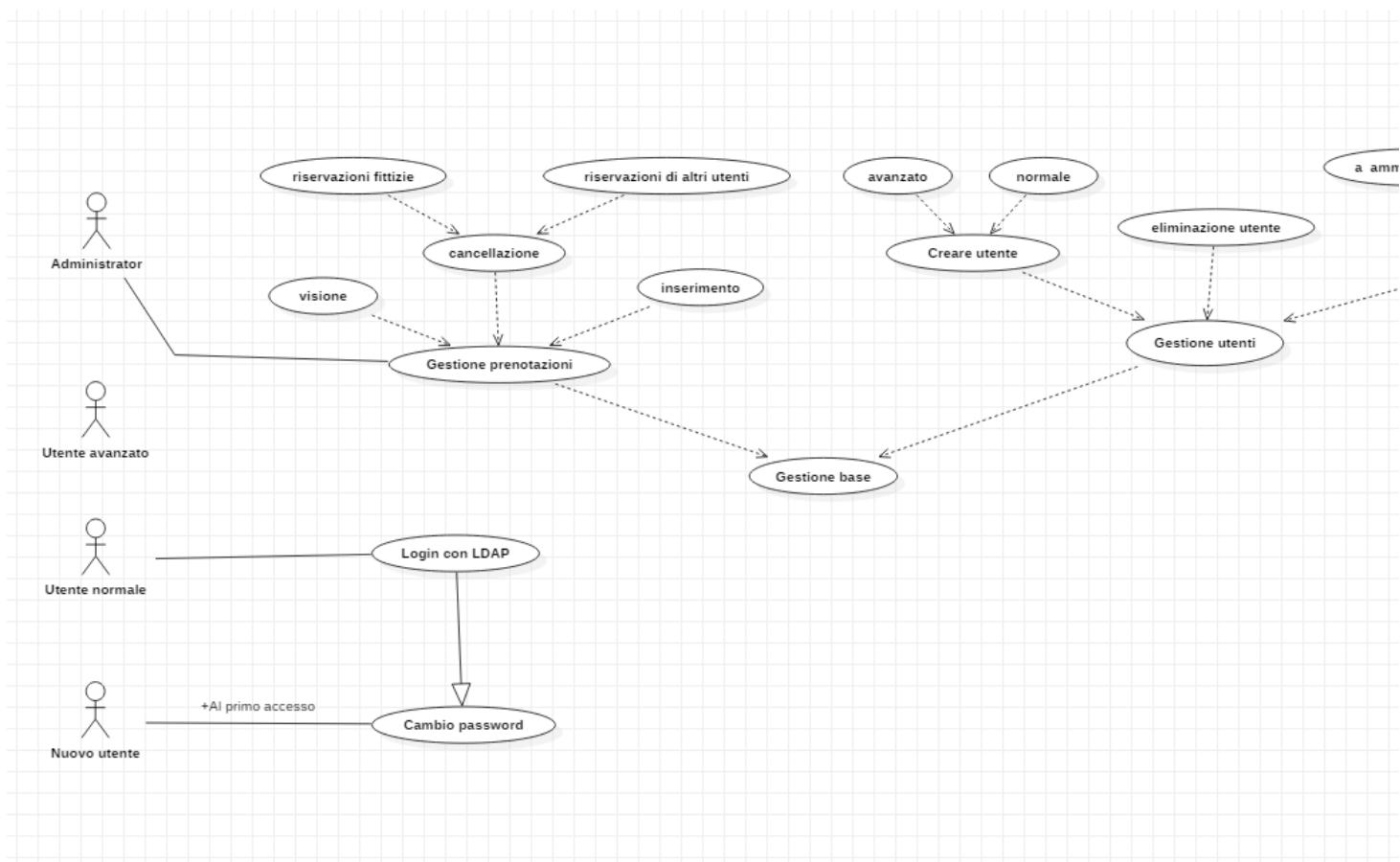
Canobbio, 06.09.2019

### Lavori svolti

Durante le prime 2 ore di lezione ho terminato il diagramma di Gantt utilizzando il prodotto installato ieri.

Durante le due ore di lezione seguenti ho iniziato a stilare il diagramma Use Case con l'aiuto di Paolo Gubeli, il quale mi ha consigliato un programma molto leggero e veloce: StarUML.

Per la creazione del mio diagramma ho utilizzato gli esempi forniti dal professor Valsangiacomo durante la lezione.



Come si può vedere dallo screenshot sovrastante il diagramma non è stato terminato per mancanza

di tempo.

Durante la lezione il mio supervisore mi ha esaminato con me i diari di lavoro delle lezioni scorse e mi ha spiegato come poter migliorare la qualità dei miei diari.

I prossimi diari non verranno più in Markdown ma verranno scritti utilizzando il template scolastico con *Microsoft Word*.

## **Problemi riscontrati e soluzioni adottate**

Non ho riscontrato nessun problema.

## **Punto della situazione rispetto alla pianificazione**

Sono al passo con i tempi prestabiliti.

## **Programma di massima per la prossima giornata di lavoro**

Finire il capitolo relativo all'analisi.

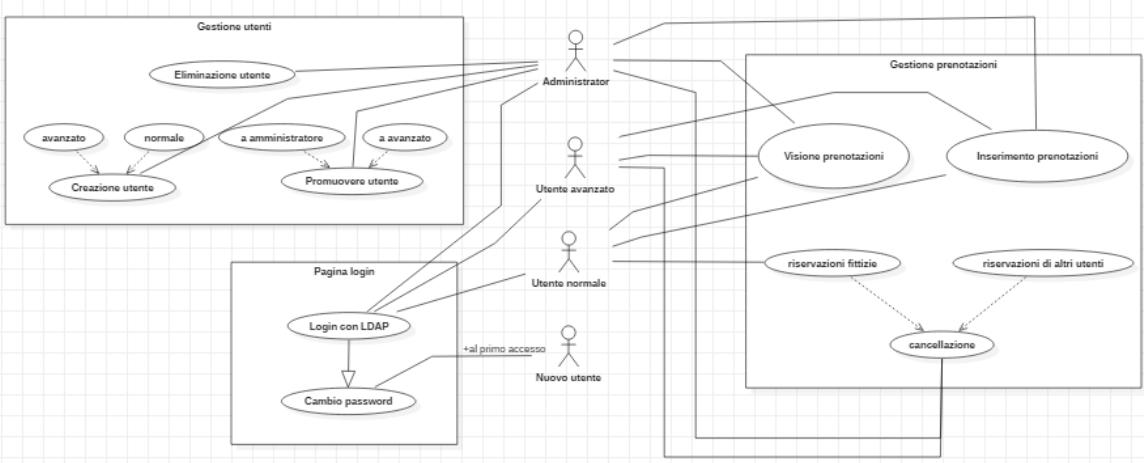
# Diario di lavoro

Luogo	Canobbio
Data	10.09.2019

## Lavori svolti

Come pianificato la scorsa lezione oggi dovevo finire il diagramma Use Case. Come prima cosa ho riletto nuovamente tutto il QDC (Quaderno Dei Compiti) ed ho notato un possibile mio errore di comprensione (descritto con cura nella sezione sottostante).

Subito dopo ho subito continuato il mio diagramma *Use Case*, finendolo verso le 14:20:

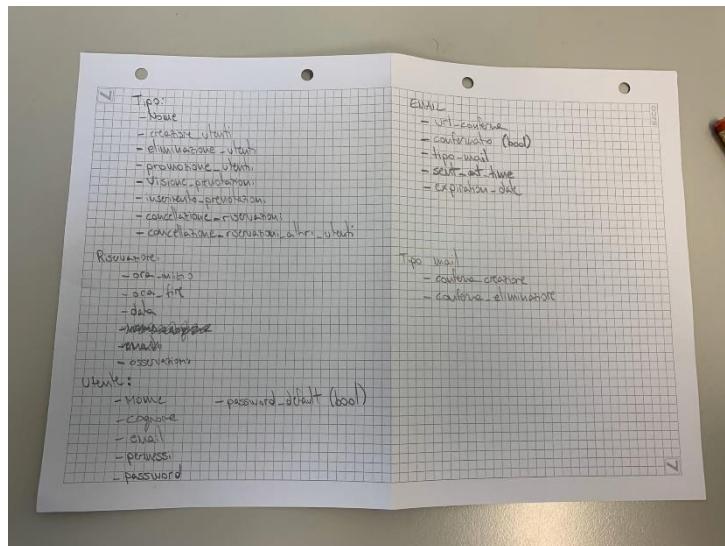


Come si può vedere molto bene dall'immagine sovrastante ho deciso di dividere il diagramma in 3 macro-gruppi (gestione utenti, gestione prenotazioni e pagina login) per semplificare sia la lettura che la stesura del diagramma.

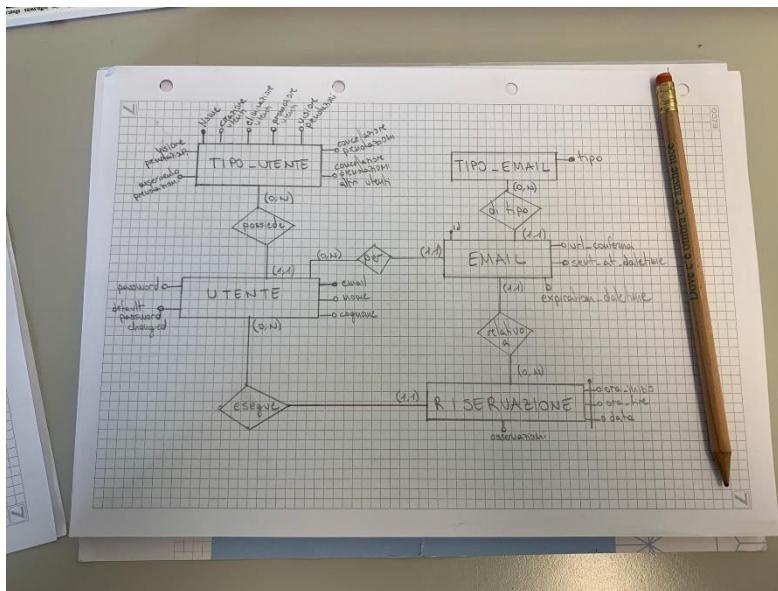
L'utente con i permessi di amministratore ha tutti i permessi sia nella gestione utenti sia nella gestione delle prenotazioni. L'utente avanzato invece ha tutti i permessi nella gestione delle prenotazioni però non può eliminare, creare o promuovere utenti. L'utente normale invece può soltanto creare prenotazioni, visionare le prenotazioni fatte dagli altri professori ed eliminare le prenotazioni create da lui (non quelle di altri utenti).

Il “nuovo utente” invece come si può vedere, ha soltanto i permessi per cambiare la password. Questo perché al primo login di un utente il sistema richiede di cambiare la password di default con una password personale. Una volta cambiata la password l'utente viene promosso automaticamente ad “Utente normale”.

Durante la seconda parte della lezione ho iniziato a fare la progettazione del database. Come prima cosa ho riletto il quaderno dei compiti segnandomi a matita su un foglio di carta quali dati si necessita salvare all'interno del database.



Sono riuscito a stilare un diagramma ER con le informazioni trovate in precedenza ma sfortunatamente non sono riuscito a ricontrolare il risultato per mancanza di tempo.



Come si può ben vedere nel diagramma è già stata fatta una prima generalizzazione esportando in altre tabelle i valori predefiniti (*tipo utente* e *tipo email*). L'entità *e-mail* è collegata a tutto perché dalla tabella mail devo sapere a chi è stata inviata ed a quale riservazione riguardava.

**Problemi riscontrati e soluzioni adottate**

Leggendo il QDC ho trovato un mio possibile errore di comprensione: nella consegna si parla più volte di “utente amministratore” e di “utente avanzato”, io ho sempre creduto che fossero due utenti separati (con due tipi di permessi separati) ma non mi risulta chiaro. Appena il professor Valsangiacomo sarà libero gli chiederò di persona. Il diagramma Use Case momentaneamente è stato steso immaginando l’utente avanzato e l’utente amministratore come utenti separati.

Durante la pausa delle 14:45 sono andato a consultare il professore: L’utente amministratore e l’utente avanzato sono due utenti differenti.

**Punto della situazione rispetto alla pianificazione**

Sono leggermente in ritardo rispetto alla pianificazione iniziale (ritardo di circa 2 ore di lavoro).

**Programma di massima per la prossima giornata di lavoro**

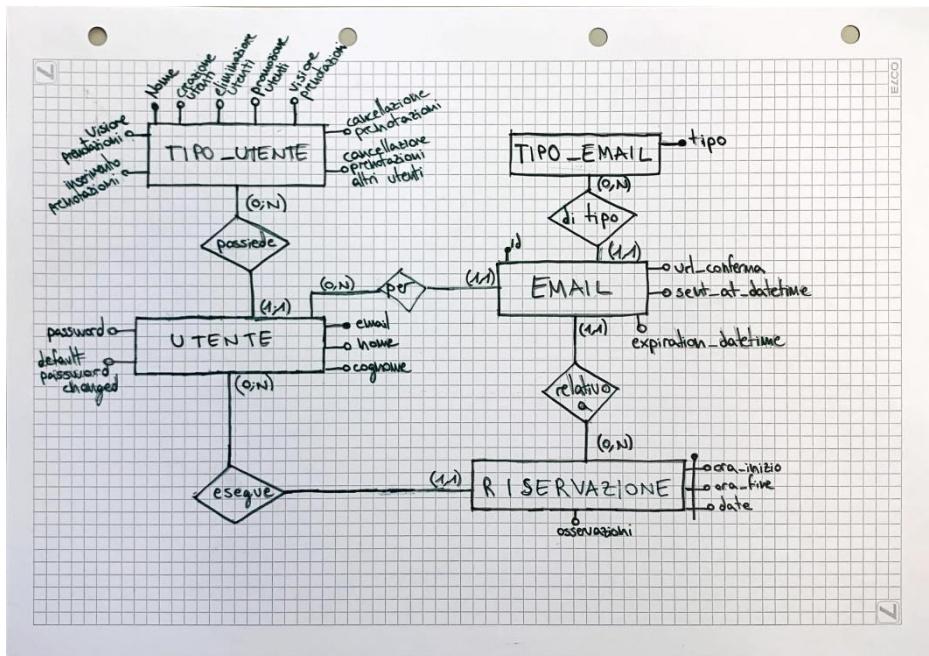
Finire il diagramma ER del database e fare i mockup delle pagine del sito

# Diario di lavoro

Luogo	Canobbio
Data	12.09.2019

## Lavori svolti

Come pianificato la scorsa lezione oggi come prima cosa dovevo controllare ed inseguito finire il diagramma ER del database. Una volta controllato che tutto fosse perfetto ho ripassato tutto con un pennarello nero ed ho scansionato il foglio per poterlo aggiungere alla documentazione.



Lo schema non ha avuto cambiamenti radicali, ho soltanto cambiato il nome di qualche attributo. Lo schema è composto di 5 entità principali, di quali due non sono altro che generalizzazioni (*tipo\_utente* e *tipo\_email*).

L'entità *tipo\_utente* possiede molti attributi; ognuno di questi attributi è un permesso dell'utente. Ho deciso di farlo in questo modo per rendere l'applicazione "aggiornabile": se in futuro si avrà la necessità di creare un nuovo tipo di utente in questa maniera non ci sarà la necessità di cambiare codice ma bisognerà soltanto aggiungere un record all'interno della tabella. Mi sono ispirato al funzionamento dei permessi di MySQL.

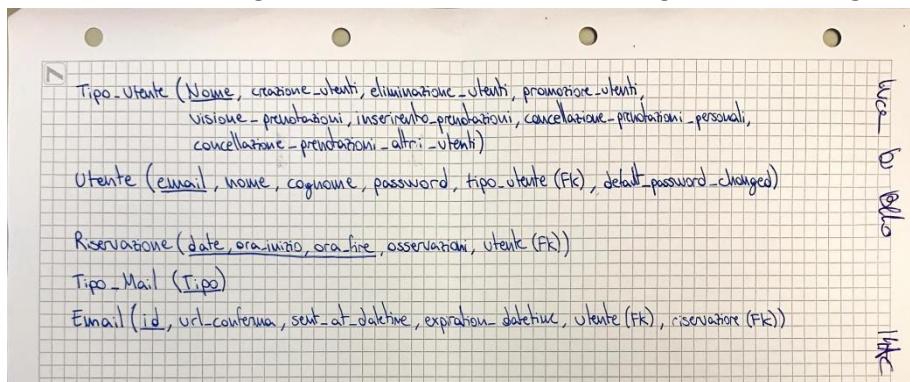
L'entità *utente* come si può intuire conterrà i dati dell'utente che utilizzerà l'applicazione tra questi anche il tipo di permessi che ha (indicato tramite l'associazione intitolata *possiede*). L'attributo invece *default\_password\_changed* servirà per tenere in memoria se l'utente ha cambiato la password al primo accesso o meno.

L'entità *riservazione* conterrà tutti i dati utili per l'identificazione della riservazione, quindi chi ha riservato e quando ha riservato.

L'entità *e-mail* è collegata a ben 3 entità: *utente*, *riservazione* e *tipo\_email*. Questo perché un'e-mail è inviata ad un utente nello specifico ed è riguardante una specifica riservazione. L'entità *tipo\_email* è utilizzata per specificare se è un e-mail1 di conferma di eliminazione o di creazione.

Gli attributi dell'entità mail servono per tenere in memoria l'url di conferma e il tempo di vita dell'url (24 ore solitamente).

Subito dopo ho scritto lo schema logico del database utilizzando il diagramma ER disegnato:



Subito dopo la pausa delle 14:45 ho iniziato a fare i mockup delle pagine web utilizzando un sito web chiamato [mockflow.com](http://mockflow.com).

Ho realizzato 4 mockup: pagina di login, pagina gestione utenti, pagina gestione prenotazioni e pagina visualizzazione prenotazioni (che verrà creata dal raspberry Pi).

Questo è il mockup che raffigura la pagina di login:

Questo è il mockup che raffigura la pagina di gestione utenti:

Nome	Cognome	Email	Password changed	Edit	Delete	Promuovi
Luca	Di Bello	luca.dibello@edu.ti.ch	Yes	@edit_button	@delete_button	@promuovi_button
Mattia	Lazzaroni	mattia.lazzaroni@edu...	No	@edit_button	@delete_button	@promuovi_button
Mattia	Toscanelli	mattia.toscanelli@edu...	Yes	@edit_button	@delete_button	@promuovi_button

Questo è il mockup che raffigura la pagina di gestione delle prenotazioni:

The modal has four buttons at the top: Button 1, Button 2, Button 3, and Button 4. Below them is a title bar with the text "Modale per inserimento prenotazione". The main content is a table with the following data:

data	dalle	alle	nome prof.	cognome prof.	elimina
25.10.19	8:20	10:45	Luca	Di Bello	@delete_button
25.1.20	8:20	10:45	Luca	Di Bello	@delete_button
25.10.22	8:20	10:45	Luca	Di Bello	@delete_button

Ed infine questo è il mockup che raffigura la pagina di visualizzazione delle prenotazioni che verrà mostrata sul monitor:

# Prenotazioni

**Data & Ora**

12 May 2016

10:20

**Professore**

Nome: Luca  
Cognome: Di Bello

**Osservazioni**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non pulvinar lorem felis nec erat.

data	dalle	alle	nome prof.	cognome prof.
25.10.19	8:20	10:45	Luca	Di Bello
25.1.20	8:20	10:45	Luca	Di Bello
25.10.22	8:20	10:45	Luca	Di Bello

Ho inserito nella documentazione un nuovo capitolo intitolato “2.2.2 Tipi di dati” questo capitolo illustrerà tutti i tipi di dati di ogni attributo di ogni entità. Questo è un piccolo screenshot:

Riservazione	
Nome attributo	Tipo attributo
Date	Primary Key, Datetime
Ora_inizio	Time
Ora_fine	Time
Password	Varchar(128)
Osservazioni	Varchar(512)
Utente	Foreign Key, varchar(255)

Problemi riscontrati e soluzioni adottate

Non ho riscontrato alcun tipo di problema.

Punto della situazione rispetto alla pianificazione

Sono leggermente in ritardo rispetto alla pianificazione iniziale.

Programma di massima per la prossima giornata di lavoro

Inserire il Gantt nella documentazione e spiegarlo, poi scrivere un abstract per completare finalmente la parte di analisi.

# Diario di lavoro

Luogo	Canobbio
Data	13.09.2019

## Lavori svolti

All'inizio della lezione sono stato chiamato dal professor Valsangiacomo per esaminare sia il Gantt preventivo stilato nei primi giorni di progetto sia i vari diari che ho inviato durante la settimana.

Il Gantt preventivo non aveva molti problemi: ho soltanto dovuto rimuovere l'attività relativa alla costruzione del documento PowerPoint, questo mi ha fatto guadagnare 9 ore di lavoro in più che ho aggiunto all'analisi e alla progettazione.

Una volta aver corretto ed esportato il diagramma di Gantt in formato A3 (come da consiglio) ho potuto finalmente aggiungerlo alla documentazione e spiegarlo nel dettaglio, per far questo ho impiegato fino alle 14:45 (pausa del pomeriggio).

Ho deciso di spiegare il Gantt pezzo per pezzo, dividendolo in 4 sotto capitoli:

- Analisi
- Progettazione
- Implementazione
- Testing

Per ogni sotto capitolo ho inserito uno screenshot ed una descrizione abbastanza dettagliata di quello che verrà fatto all'interno di ogni attività. Questo è un piccolo screen:

### ▪ 1.7.1.1 → Analisi¶

↳ 1.1 Analisi	0,63 days	Tue 03/09/19	Thu 05/09/19
1.1.1 Intervista con il committente	15 mins	Tue 03/09/19	Tue 03/09/19
1.1.2 Raccolta requisiti	15 mins	Tue 03/09/19	Tue 03/09/19
1.1.3 Analisi dei requisiti	90 mins	Tue 03/09/19	Tue 03/09/19
1.1.4 Analisi dei mezzi	30 mins	Thu 05/09/19	Thu 05/09/19
1.1.5 Usecase	90 mins	Thu 05/09/19	Thu 05/09/19

Figure 3 - Gantt attività analisi¶

Ho deciso di suddividere il capitolo di analisi in 5 attività:¶

- → Intervista con il committente¶
  - → Intervista con il committente, utilizzata per chiarire i dubbi relativi alla consegna¶
- → Raccolta dei requisiti¶
  - → Raccolta dei requisiti tramite la lettura approfondita del QdC (Quaderno Dei Compiti)¶
- → Analisi dei requisiti¶
  - → Riflessione riguardo alle richieste raccolte, analizzandole e suddividendole in modo sensato¶
- → Analisi dei mezzi¶
  - → Riflessione riguardo ai mezzi da utilizzare per il progetto (sia Software che Hardware)¶
- → Use-case¶
  - → Stesura del diagramma Use-Case¶

Durante la seconda parte della lezione ho scritto e corretto l'abstract, terminando così il capitolo dell'implementazione. L'abstract è stato scritto pensando al problema della carta: ha un costo elevato (a lungo andare) e c'è un rischio altissimo di perdita di informazioni.

**Problemi riscontrati e soluzioni adottate**

Ho dovuto rimuovere dal diagramma di Gantt l'attività relativa alla costruzione del diagramma PowerPoint. Questo perché durante l'esame avremo un week-end per la costruzione e la prova di esso.

**Punto della situazione rispetto alla pianificazione**

Avendo rimosso l'attività relativo alla costruzione del documento PowerPoint sono riuscito a guadagnare 9 ore di lavoro aggiuntive (3 giorni di lavoro circa). Questo mi ha permesso di tornare a rispettare le tempistiche pianificate con il diagramma di Gantt.

**Programma di massima per la prossima giornata di lavoro**

Iniziare a fare il diagramma di flusso.

# Diario di lavoro

Luogo	Canobbio
Data	17.09.2019

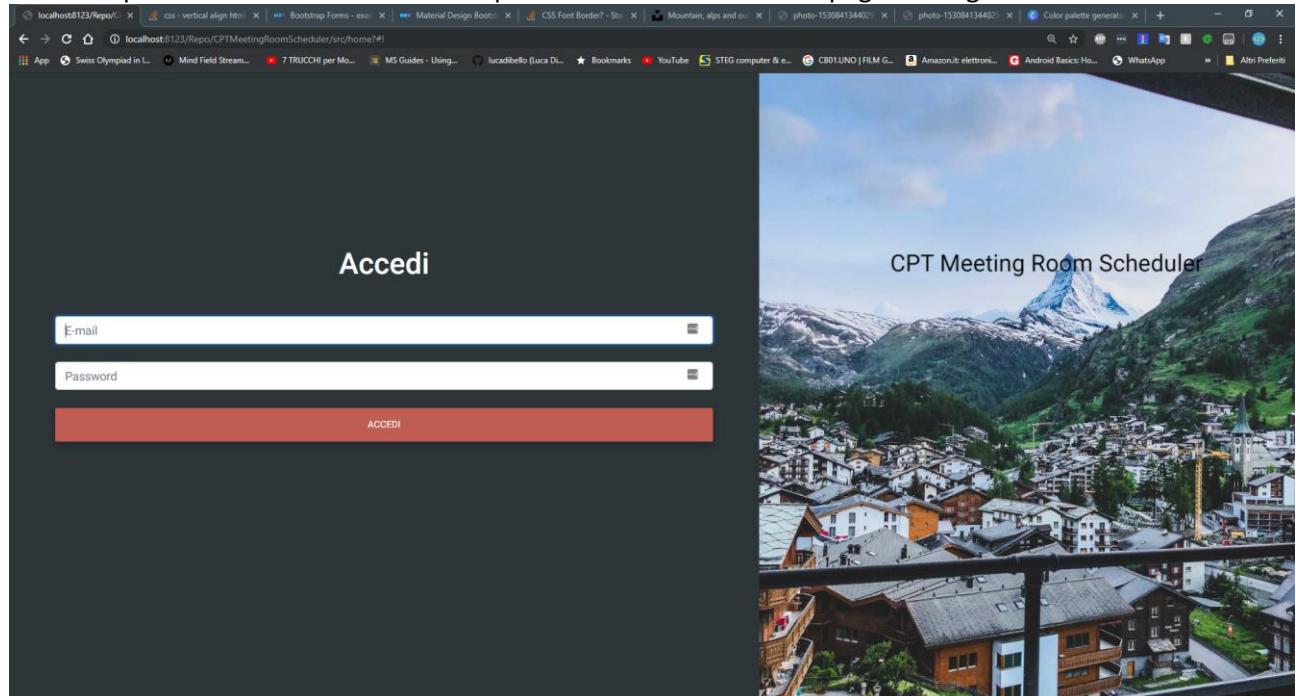
## Lavori svolti

Subito prima dell'inizio della lezione sono stato dal professor Valsangiacomo per chiedere informazioni relative ai diagrammi di flusso. Mi è stato detto che per il momento non sono richiesti e che quindi posso già iniziare a fare qualcosa relativo all'implementazione.

Dato che i diagrammi di flusso servono per progettare la logica del sistema oggi mi sono occupato soltanto della creazione della parte front-end delle varie pagine.

Come prima cosa ho creato la base del sito web da un mio vecchio progetto fatto in Irlanda durante lo stage. Esso infatti utilizzava una base PHP che segue il pattern MVC che ci è stata data dal professor Sartori l'anno scorso durante il corso di PHP.

Subito dopo ho eliminato tutte i file non più utilizzati ed ho creato la pagina di login:



Come si può vedere la scritta nell'immagine a destra non è molto visibile. Quando avrò tempo (probabilmente lo farò a casa) creerò un logo per il sito web utilizzando adobe illustrator.

Tutta la pagina è stata creata senza nessun template, utilizzando soltanto il framework *Material Design Bootstrap* e *Parallax.js*.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato alcun problema.

Punto della situazione rispetto alla pianificazione

Avendo posticipato la creazione

Programma di massima per la prossima giornata di lavoro

Creazione pagina di gestione utenti

# Diario di lavoro

Luogo	Canobbio
Data	19.09.2019

## Lavori svolti

Durante la lezione odierna ho deciso di dedicarmi all'implementazione del database e nella creazione delle librerie che permetteranno la lettura e scrittura di dati al loro interno.

Come prima cosa ho iniziato a creare il database e le varie tabelle seguendo il diagramma ER stilato durante le lezioni scorse. Per far questo ho utilizzato *MySQL Workbench 6.3* il quale permette la creazione di tabelle tramite GUI.

Questo è lo schema di come risulta il database dopo la creazione di tutte le tabelle:

Name	Engine	Version	Row Format
email	InnoDB	10	Dynamic
riservazione	InnoDB	10	Dynamic
tipo_mail	InnoDB	10	Dynamic
tipo_utente	InnoDB	10	Dynamic
utente	InnoDB	10	Dynamic

Prima di eseguire il codice SQL generato dal programma ho avuto la prontezza di salvarlo in un file esterno. Questo mi sarà utile per la creazione del database sul server finale (quello dove stazionerà il database dopo la consegna).

```

12      `cancellazione_prenotazioni_personalni` BIT NOT NULL DEFAULT 0,
13      `cancellazione_prenotazioni_altri_utenti` BIT NOT NULL DEFAULT 0,
14  PRIMARY KEY (`nome`);

15
16  # utente table
17  CREATE TABLE `cptmrs`.`utente` (
18      `email` VARCHAR(255) NOT NULL,
19      `nome` VARCHAR(100) NOT NULL,
20      `cognome` VARCHAR(100) NOT NULL,
21      `password` VARCHAR(128) NOT NULL,
22      `tipo_utente` VARCHAR(50) NOT NULL,
23      `default_password_changed` BIT NOT NULL DEFAULT 0,
24  PRIMARY KEY (`email`),
25  INDEX `fk_utente_tipo:utente_idx`(`tipo_utente` ASC),
26  CONSTRAINT `fk_utente_tipoutente`
27    FOREIGN KEY (`tipo_utente`)
28      REFERENCES `cptmrs`.`tipo_utente`(`nome`)
29      ON DELETE CASCADE
30      ON UPDATE CASCADE);

31
32  # riservazione table
33  CREATE TABLE `cptmrs`.`riservazione` (
34      `id` INT NOT NULL AUTO_INCREMENT,
35      `data` DATE NOT NULL,
36      `ora_inizio` TIME NOT NULL,
37      `ora_fine` TIME NOT NULL,
38      `osservazioni` VARCHAR(512) NULL,
39      `utente` VARCHAR(255) NULL

```

Nel codice SQL ho aggiunto dei commenti a mano per poter riconoscere più facilmente le tabelle.

Subito dopo ho iniziato la creazione dell'interfaccia PHP con il database, per questo ho deciso di utilizzare un framework per database molto famoso: *MeekroDB (Versione 2.3)*.

Una volta letto per bene tutte le varie funzionalità di questo framework mi sono reso conto che la creazione di questa "libreria per l'interfacciamento con il database" era totalmente inutile: il framework permetteva di fare il tutto in maniera totalmente sicura e quindi sono subito passato all'attività relativa al login con LDAP.

Con l'aiuto di Giulio sono subito riuscito a trovare la soluzione al problema del login. Il professorBarchi l'anno scorso gli aveva dato tutti i dati necessari per la connessione con l'active directory di scuola ed il codice per poter eseguire l'accesso. Questa è l'e-mail del professor Barchi per Giulio:

From: Adriano Barchi <[adriano.barchi@edu.ti.ch](mailto:adriano.barchi@edu.ti.ch)>  
 Subject: Re: Procedura connessione LDAP  
 Date: 15 February 2019 at 15:26:45 CET  
 To: Giulio Bosco <[giulio.bosco@samtrevano.ch](mailto:giulio.bosco@samtrevano.ch)>

Ciao Giulio,  
 guarda questo link:

<https://samjlevy.com/php-login-script-using-ldap-verify-group-membership/>  
 ovviamente devi avere la libreria ldap php

poi devi sostituire i diversi nomi in authenticate.php:

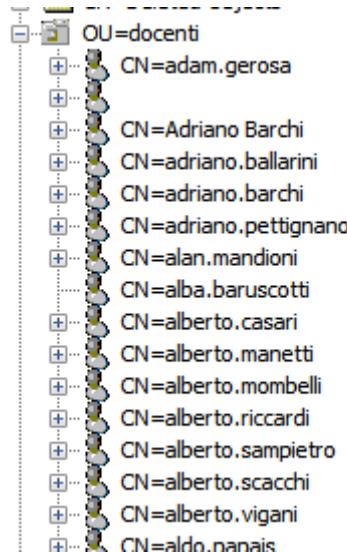
- \$ldap\_host sarà 'cpt.local'
- \$ldap\_dn sarà l'indirizzo fqdn, esepio, per degli allievi di 3a info: 'OU=3,OU=I,OU=IN,OU=SAM,OU=allievi,DC=CPT,DC=local'
- \$ldap\_user\_group sarà: 'SAM-IN-I-3'
- \$ldap\_usr\_dom sarà : '@cpt.local'

La struttura che devi utilizzare la puoi analizzare con AdExplorer.exe che trovi a questo percorso:  
 \sv-104-info1.cpt.local\dati\SW\Tools\AD-Ldap-Explorer.

devi solo copiarlo da qualche parte, eseguirlo ed inserire il nome del sever del dominio: cpt.local  
 ed accedere con il tuo account.

Ciao, buon lavoro  
 A. Barchi

Grazie al tool consigliato nella mail sono riuscito a trovare la path per i profili dei professori:



Questa è la path che utilizzerò per l'implementazione del login con LDAP: *OU=docenti,DC=CPT,DC=local*

Una volta aver implementato il codice ed averlo incorporato nel programma ho provato a fare un login e mi è uscito questo errore:

**( ! ) Fatal error: Uncaught Error: Call to undefined function ldap\_connect()**

Con una semplice ricerca web ho scoperto che per utilizzare la funzione *ldap\_connect()* bisognava abilitare un modulo chiamato "*ldap*": una volta abilitato la funzione invia correttamente i dati.

Per testare il funzionamento del sistema ho provato a loggare utilizzando il mio account scolastico ed utilizzando la path: "*OU=4,OU=I,OU=IN,OU=SAM,OU=allievi,DC=CPT,DC=local*".

**Problemi riscontrati e soluzioni adottate**

Nessun problema degno di nota. Tutti i piccoli problemi sono descritti sopra.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo di un paio di ore rispetto alla pianificazione.

**Programma di massima per la prossima giornata di lavoro**

Finire sistema di login.

# Diario di lavoro

Luogo	Canobbio
Data	20.09.2019

## Lavori svolti

Come pianificato ieri oggi ho terminato la pagina di login inserendo un nuovo di controllo ed una nuova feature.

Quando un utente esegue un login ora il sistema controlla se l'utente è effettivamente un professore utilizzando LDAP. Se il sistema rileva che è un professore controlla se quel determinato professore è stato abilitato all'utilizzo dell'applicazione.

Se il professore non è abilitato all'utilizzo del tool viene mostrato un messaggio a schermo nella pagina di login:

Sembra che il tuo account non sia abilitato per accedere a questo servizio, contatta l'amministratore

Subito dopo ho iniziato a lavorare su un sistema che permette il controllo dei permessi dell'utente.

Come prima cosa ho creato una classe chiamata *PermissionManager*, la quale permette di leggere i

permessi dell'utente dal database. I permessi vengono formattati e messi in una classe chiamata

*Permissions* la quale ha delle funzioni che semplificano l'identificazione di un determinato permesso:

- canCreareUtenti():bool
- canEliminareUtenti():bool
- canPromozioneUtenti():bool
- canVisionePrenotazioni():bool
- canInserirePrenotazioni():bool
- canCancellazionePrenotazioniPrivate():bool
- canCancellazionePrenotazioniGlobali():bool

Quando l'utente effettua il login correttamente il sistema crea una variabile di sessione chiamata “permissions” la quale contiene al suo interno un oggetto di tipo “Permissions”. Per esempio, se mi loggo con il mio account (“luca.dibello”, il quale è stato impostato con permessi di amministratore) questo è quello che contiene l'oggetto:

```
C:\Users\luca6\Google Drive\Repo\CPTMeetingRoomScheduler\src\application\views\home\index.php:28:
object(Permissions)[2]
    private 'PERMISSIONS_NAME' => string 'admin' (Length=5)
    private 'creazione_utenti' => boolean true
    private 'eliminazione_utenti' => boolean true
    private 'promozione_utenti' => boolean true
    private 'visione_prenotazioni' => boolean true
    private 'inserimento_prenotazioni' => boolean true
    private 'cancellazione_prenotazioni_personali' => boolean true
    private 'cancellazione_prenotazioni_altri_utenti' => boolean true
```

Andando a vedere il codice per la connessione al database ho notato un errore molto grosso: le credenziali di accesso (username, password, database ed host) non erano delle costanti globali ma delle stringhe:

```
DB::$user = "root";
```

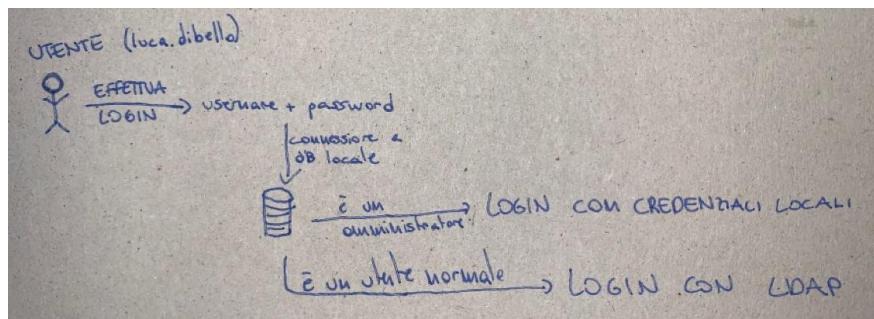
Il problema di questo approccio è che quando bisognerà cambiare server e/o database non sarà molto pulito. Per mettere apposto ho deciso di creare nel file *config.php* delle costanti:

```
define('DATABASE_USERNAME', 'root');
define('DATABASE_PASSWORD', 'root');
define('DATABASE_NAME', 'cptmrs');
```

Che vengono poi utilizzate nel file *index.php* per il setup di MySQL:

```
// setup database variables
DB::$user = DATABASE_USERNAME;
DB::$password = DATABASE_PASSWORD;
DB::$dbName = DATABASE_NAME;
```

Questo è uno schemino che ho fatto per illustrare come il login di un utente viene effettuato:



**Problemi riscontrati e soluzioni adottate**

Oggi non ho riscontrato alcun tipo di problema, tutti i lavori sono stati molto semplici ed oltre a qualche errore di sintassi non ho avuto alcun problema.

**Punto della situazione rispetto alla pianificazione**

Sono leggermente in anticipo rispetto alla pianificazione.

**Programma di massima per la prossima giornata di lavoro**

Finire sistema di controllo permessi ed implementare il sistema relativo alle password per i

# Diario di lavoro

Luogo	Canobbio
Data	24.09.2019

## Lavori svolti

Come prima cosa ho finito il sistema di login implementando sia il login tramite database locale che con LDAP utilizzando lo stesso username (come richiesto dal professore).

Il sistema prova prima a loggarsi cercando le credenziali sul database locale, se non trova nulla o le credenziali trovate sono errate prova a loggarsi tramite LDAP. Se anche il login tramite LDAP non va a buon fine viene mostrato un messaggio a schermo.

Quando l'utente possiede ancora la password di default (quella generata alla creazione dell'utente) essa non è cryptata all'interno del database, ma è scritta in chiaro. Ho deciso così perché renderà più semplice il primo login dell'utente: non dovrà scrivere un hash di 32 o più caratteri ma soltanto una stringa random di 10/15.

Quando l'utente esegue il primo accesso, e quindi che cambia password, essa sarà salvata come hash all'interno del database.

Infatti, subito dopo mi sono messo a lavorare sul sistema di login, creando un'interfaccia molto semplice ed intuitiva:

## Cambia la tua password

Il sistema ha rilevato che hai eseguito il tuo primo accesso. Utilizza questo modulo per sostituire la tua password con una personale.

- × 8 Characters Long
- × One Uppercase Letter

- × One Lowercase Letter
- × One Number

× Passwords Match

Successivamente ho iniziato a creare un sistema che genera dei link univoci, utilizzati per accedere alla pagina di cambio password. Questo è un esempio di link:

[changepassword?id=\\$2y\\$10\\$rPE0THG57He3STAIrSt8JO9ao5L1h5zLBzBFL8AIA0rNVipuqV2Z2](changepassword?id=$2y$10$rPE0THG57He3STAIrSt8JO9ao5L1h5zLBzBFL8AIA0rNVipuqV2Z2)

Come si può vedere questo link non è completo. Questo perché è la parte di URL che definisce il controller che andrà a contattare e la chiave univoca che identifica la richiesta (parametro GET id).

**Problemi riscontrati e soluzioni adottate**

Ho risolto i diversi problemi che c'erano con il sistema di login. Ora funziona correttamente e rispetta i requisiti del QdC.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo di diversi giorni di lavoro rispetto alla pianificazione. Ho “guadagnato” molto tempo decidendo di fare i diagrammi di flusso in un secondo momento.

**Programma di massima per la prossima giornata di lavoro**

Finire sistema di controllo permessi ed implementare il sistema relativo alle password per i

# Diario di lavoro

Luogo	Canobbio
Data	27.09.2019

## Lavori svolti

Durante la scorsa giornata di lavoro ho deciso di salvare nel database buona parte del link di conferma:

`changepassword?id=$2y$10$rPE0THG57He3STAIrSt8JO9ao5L1h5zLBzBFL8AIA0rNVipuqV2Z2`

Oggi, riguardando i vari lavori svolti la scorsa lezione, mi sono reso conto che è totalmente inutile salvare la prima parte dell'url dato che rimane sempre statica ('`changepassword?id=`'). Ora nel database viene salvato soltanto il token.

La gestione della scadenza ho deciso di implementarla quando l'applicazione sarà un po' più completa.

Ho aggiunto una pagina di errore provvisoria che viene mostrata quando un utente inserisce un token non valido.

**Il token che hai inserito non è valido. Per favore controlla le tue email.**

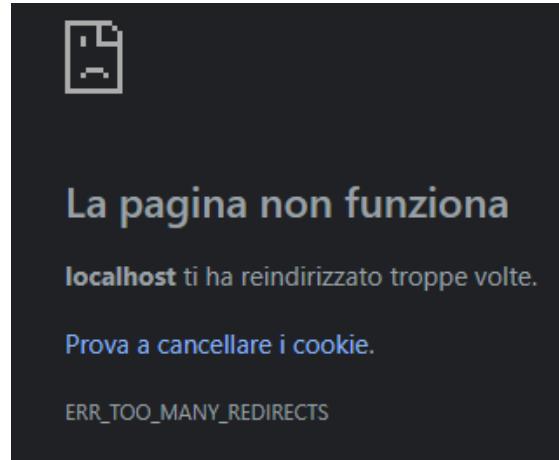
**Puoi usare questo link per inviare un'altra mail con il codice di conferma.**

Subito dopo la pausa del pomeriggio ho dovuto prender parte ad una riunione relativa agli stage all'estero. Essa mi ha preso solamente 45 minuti.

Appena tornato dalla riunione ho sfruttato l'ultima ora di lezione per eseguire del buon debugging sul problema relativo ai redirect (spiegato in modo dettagliato nella sezione *problemri riscontrati*).

Problemi riscontrati e soluzioni adottate

Quando si esegue il login la pagina esegue troppi redirect e quindi Chrome termina la connessione con un errore *ERR\_TOO\_MANY\_REDIRECTS*.



Ho eseguito un po' di debugging ma non ho ancora trovato il motivo. Sicuramente il problema sono i metodi *Header*, utilizzati per eseguire redirect tra pagine.

Punto della situazione rispetto alla pianificazione

Sono in anticipo di una lezione rispetto alla pianificazione iniziale.

Programma di massima per la prossima giornata di lavoro

Correggere errore con relativo al login e finire finalmente il sistema di cambio password.

# Diario di lavoro

Luogo	Canobbio
Data	01.10.2019

## Lavori svolti

Come prima cosa ho iniziato a fare debugging sul problema dei redirect della scorsa lezione. Per fare debugging ho utilizzato il modulo XDebug, XDebug Helper (estensione che inoltra tutte le richieste POST e GET al debugger) e PHPStorm. Grazie al debugger sono riuscito ad individuare e risolvere il problema (Vedi “*problemi riscontrati e soluzioni adottate*”).

Dopo aver messo apposto il problema ho rifatto la pagina di cambio password dato che non era di mio gradimento.

The screenshot shows a web form titled "Cambia la password". It contains two input fields: "Nuova password" (New password) and "Conferma password" (Confirm password), each preceded by a lock icon. Below the fields is a teal "REGISTER" button. A note above the fields states: "La tua password deve essere lunga almeno 8 caratteri, contenere lettere e numeri, non deve contenere spazi o emoji."

Grazie all'aiuto di Filippo Finke sono riuscito a capire perché non potevo caricare le pagine web da metodi di un determinato controller. I miei link iniziavano con “./application/...” che però non vengono caricati se la pagina è caricata da un metodo, quindi ho dovuto cambiare tutti i link facendoli partire dalla root del progetto: /application/...

Ho anche creato una semplice libreria chiamata *RedirectManager* per semplificare i redirect. Ora per eseguire un redirect devo scrivere:  
*RedirectManager::redirect(<controller>)*.

Sono riuscito a finire sia l'interfaccia sia la logica che sta dietro alla funzione relativa al cambio password al primo login.

Quando si cambia password il sistema esegue un logout (utilizzato per pulire tutte le variabili della sessione), fa un redirect alla pagina di login la quale mostra un modal con delle informazioni:

### Problemi riscontrati e soluzioni adottate

Ho trovato la soluzione all'errore della scorsa volta (*ERR\_TOO\_MANY\_REDIRECTS*). Quando io utilizzavo il metodo *Header* per fare il redirect ad un'altra pagina utilizzavo un link errato, che faceva pensare all'applicazione di utilizzare un metodo di un controller:

Ho messo *login/changepassword?id=...* al posto di *../changepassword?id=...*

### Punto della situazione rispetto alla pianificazione

Sono al passo rispetto alla pianificazione iniziale.

### Programma di massima per la prossima giornata di lavoro

Finire il definitivamente il sistema per il cambio password mostrando le informazioni nella home.

# Diario di lavoro

Luogo	Canobbio
Data	03.10.2019

## Lavori svolti

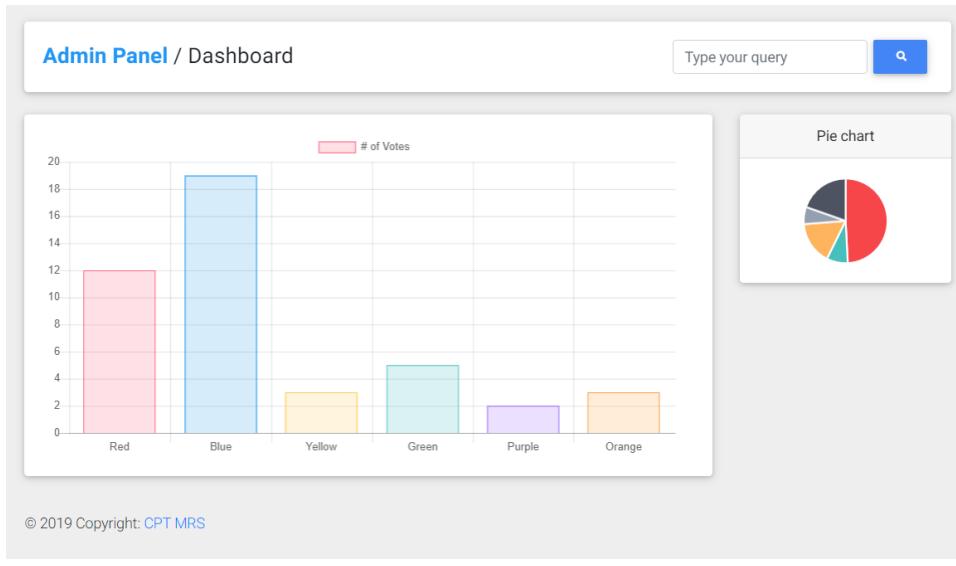
Testando il sistema di cambio password ho notato che, una volta cambiata la password (la quale viene criptata tramite *hash*), l'utente non riesce più a loggarsi dato che il sistema non riconosce più la password.

La soluzione all'errore è illustrata nella sezione “*Problemi riscontrati e soluzioni adottate*”.

Subito dopo aver messo apposto il tutto ho iniziato a fare un po' di testing sul sistema di login.

Una volta aver finito tutti i vari test ho deciso di iniziare la creazione della pagina per la gestione degli utenti. Ho deciso anche che lo sviluppo del sistema che andrà a controllare i permessi degli utenti verrà creato in parallelo allo sviluppo della pagina di gestione utenti e prenotazioni. Questo perché mi sono accorto che non avrei nulla per poter controllare il corretto funzionamento di questa funzionalità.

Ho utilizzato il pannello admin di *Material Design Bootstrap*. Per avere già la base funzionante non ho fatto altro che riutilizzare il pannello admin che ho creato per un mio progetto personale (sito web per una piccola azienda di Chiasso).



In seguito, ho creato un all'interno del controller “admin” (utilizzato per accedere alla pagina web) un sistema che controlla se l'utente ha i permessi necessari per accedere al pannello admin. Ho deciso che di default soltanto gli utenti che fanno parte del gruppo “*admin*” possano accedere al pannello. Questo è il codice che controlla i permessi per accedere al pannello:

```
// This function checks if a user can access any of the admin panel functions
private function is_user_valid(): int{
    if(Auth::isAuthenticated()){
        $user_perms = PermissionManager::getPermissions();
        if($user_perms instanceof Permissions && $user_perms->getPermissionName() == ADMIN_PANEL_USER_PERMISSION_GROUP){
            return self::STATUS_OK;
        }
        else{
            return self::STATUS_BAD_PERMISSIONS;
        }
    }
    else{
        return self::STATUS_NOT_LOGGED;
    }
}
```

Questa è la pagina di gestione utenti. Come si può ben vedere dai nomi dei campi non ho fatto in tempo a cambiare i nomi dei campi ed aggiungere quelli mancanti. Tutto questo andrà fatto la prossima lezione.

The screenshot shows a web application interface for managing users and categories. At the top, there is a navigation bar with links for 'Dashboard', 'Gestione utenti', and 'Gestione prenotazioni'. On the right side of the header, there is a GitHub repository link. The main content area has two sections: 'Aggiungi utente' (Add User) and 'Categorie' (Categories). The 'Aggiungi utente' section contains fields for 'Nome\*' (Name), 'Descrizione categoria' (Category Description), and an 'Immagine di categoria' (Category Image) input field containing the value 'admin'. A green button labeled 'AGGIUNGI CATEGORIA' (Add Category) is visible. Below this, the 'Categorie' section displays a table with one row, indicating 'Non ci sono categorie.' (There are no categories.). The table columns are: #, Nome categoria (Category Name), Numero episodi (Number of episodes), Descrizione (Description), Path immagine (Image path), Data creazione (Creation date), Ultima modifica (Last modification), and Azioni (Actions).

**Problemi riscontrati e soluzioni adottate**

Andando ad esaminare il codice relativo al login tramite database locale ho notato che ho inserito i parametri del metodo *password\_verify* nel modo sbagliato: ho invertito il valore dell'hash e della password.  
L'utente ora riesce a cambiare la password ed accedere.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo di 7 giorni di lezione rispetto alla pianificazione. Naturalmente devo ancora creare i diagrammi di flusso, i quali preferisco fare più avanti dato che ho notato che la logica del programma cambia molto frequentemente per via della risoluzione dei bug.

**Programma di massima per la prossima giornata di lavoro**

Finire la pagina di gestione utenti la quale mostrerà tutti gli utenti locali che hanno accesso al sistema

# Diario di lavoro

Luogo	Canobbio
Data	04.10.2019

## Lavori svolti

Oggi ho creato una nuova libreria chiamata *User*. Essa permette di salvare al suo interno tutti i dati di un utente. Viene utilizzata dal programma per una gestione semplificata dei dati dell'utente.

Tramite il model *UserModel* posso adesso caricare tutti i dati degli utenti salvati all'interno della banca dati locale sotto forma di array di oggetti *User*. Questo viene utilizzato per scrivere i dati all'interno della tabella per la visualizzazione degli utenti nella pagina “*Gestione Utenti*” del pannello admin.

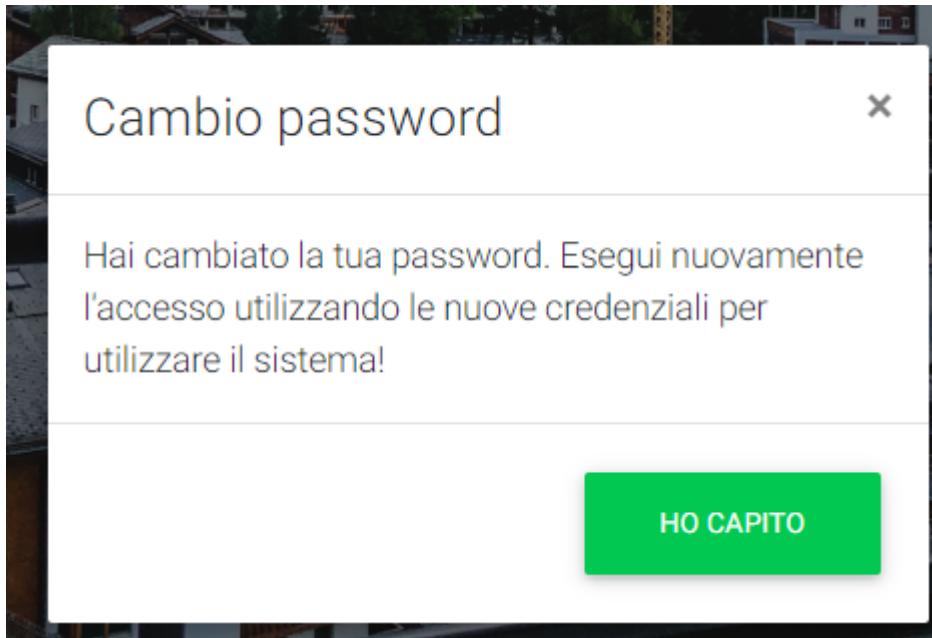
Tramite la libreria JavaScript *DataTables* ho potuto inserire delle funzioni molto utili per cercare/filtrare dati all'interno della tabella degli utenti:

Utenti				
Username	Nome	cognome	Permessi	Password cambiata
luca.dibello	Luca	Di Bello	admin	No
mattia.lazzaroni	Mattia	Lazzaroni	user	No
Showing 1 to 2 of 2 entries				
Search: <input type="text"/> Opzioni: <a href="#">MODIFICA</a> <a href="#">ELIMINA</a> <a href="#">MODIFICA</a> <a href="#">ELIMINA</a>				
Previous <a href="#">1</a> Next				

La libreria permette di fare una ricerca all'interno dei dati, di ordinare i dati per colonna (es: ordinare alfabeticamente per username) e di dividere i dati su più pagine (di default 10 record per pagina).

Dopo aver fatto questo ho messo apposto un altro problema relativo alla generazione dei link per il cambio password (spiegato in modo dettagliato nella sezione *Problemi riscontrati e soluzioni adottate*).

Dopo aver risolto il problema ho aggiunto un cookie che verrà utilizzato come flag nella pagina di login. Se un account ha cambiato password questo flag viene impostato a true e se entro 24 ore dal cambio password accede alla pagina di login viene mostrato un messaggio che conferma il cambio della password.



### Problemi riscontrati e soluzioni adottate

Ogni tanto quando cercavo di effettuare l'accesso con un nuovo utente l'applicazione mi dava l'errore di quando non trova il token per l'identificazione:  
Hai fornito un id non valido.

Controllando per bene il token ho notato che criptandolo con il metodo *password\_hash* c'era la possibilità che qualche carattere all'interno del token diventavano degli '/' (slash).

localhost:8123/changepassword/id/**\$2y\$10\$rTOsP.XwQLRllvtNDmZp0OcHe/jrtQZU/2U8jeEYLhQBhT.VRIQYq**

Proprio per questo ho deciso di virare su un MD5. Dato che questo metodo non genera automaticamente un *salt* ho al posto di criptare un numero random tra 1 e 10, cripto il timestamp del sistema per essere sicuri di avere sempre un token univoco. Ecco un esempio di link utilizzando il nuovo sistema:

localhost:8123/changepassword/id/**bde80344dc4c32450c6ce5fdfc6bb63f**

### Punto della situazione rispetto alla pianificazione

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

### Programma di massima per la prossima giornata di lavoro

Inserire la funzione di eliminazione ed aggiunta di utenti (tutto con validazione).

# Diario di lavoro

Luogo	Canobbio
Data	08.10.2019

## Lavori svolti

Ho iniziato a creare il sistema di API per la creazione, aggiunta e modifica di un utente. Per far questo utilizzo un nuovo controller chiamato *Api*.

I vari URL per le richieste saranno questi:

Azione	Url	Dati extra
Delete	Api/delete/<username>	-
Add	Api/add	Array POST
Update	Api/update/<username>	Array POST

Tutti i dati vengono validati grazie alla classe UserValidator (creata all'interno della libreria *User*).

Una volta che viene creato l'utente, egli deve ricevere per e-mail le credenziali per l'accesso al sistema. Avevo eliminato il campo "email" dal database senza pensare a questo particolare quindi ho dovuto ricreare il campo ed andare a modificare la libreria *User*.

Ho creato il form per l'aggiunta degli utenti. Esso funziona perfettamente e mostra anche gli errori trovati da back-end. Questo è un piccolo screenshot:

The screenshot shows a web form titled "Aggiungi utente". It contains several input fields: "Nome" (Name) and "Cognome" (Surname) with placeholder text "Nome" and "Cognome" respectively; "Email" with placeholder "@edu.ti.ch"; "Username"; and a dropdown menu for "Permessi utente" (User permissions) with "admin" selected. At the bottom is a large green button labeled "CREA UTENTE" (Create User).

Ho creato e testato anche il pulsante relativo all'eliminazione dell'utente:



Ho creato anche il metodo per la modifica dell'utente (api/update) ma non è stato ancora testato perché non ho ancora finito l'interfaccia grafica necessaria per questa funzione.

Verso la fine della lezione ho iniziato a fare i controlli dei permessi utilizzando l'oggetto di tipo *Permission* generato al login dell'utente.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato alcun tipo di problema.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

**Programma di massima per la prossima giornata di lavoro**

Finire la modifica dell'utente, aggiungere la possibilità di fare una promozione dell'utente ad un nuovo gruppo (es: user, admin, ...) e finire la gestione dei permessi.

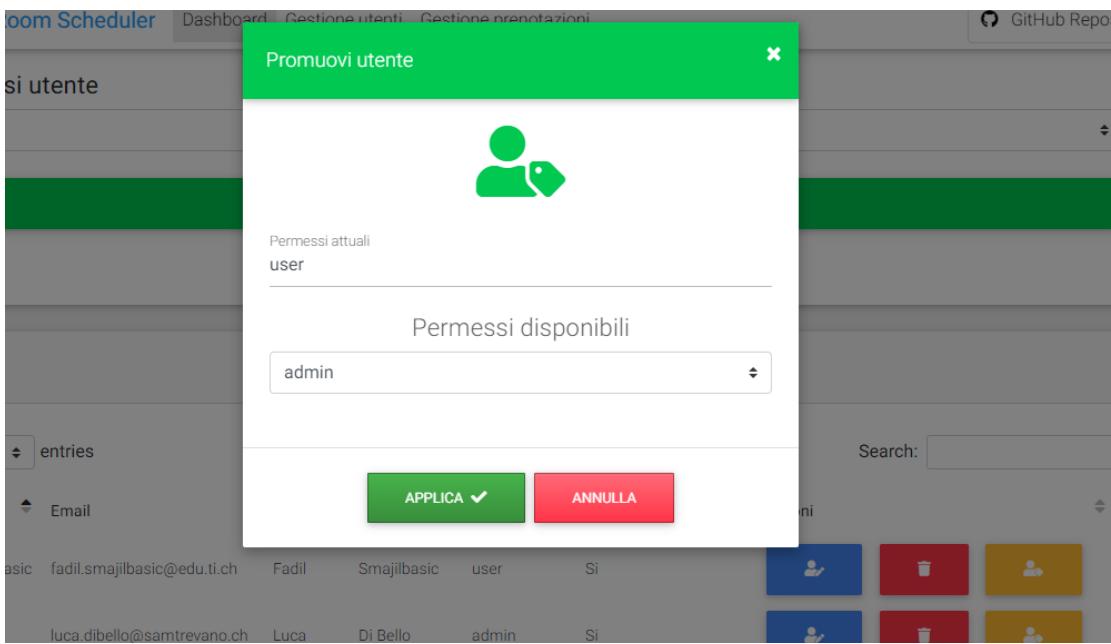
# Diario di lavoro

Luogo	Canobbio
Data	10.10.2019

Lavori svolti

Come prima cosa ho aggiunto un nuovo metodo all'api: il metodo *promote*. Esso, come dice già il nome, è utilizzato per promuovere un utente e quindi cambiarli i permessi.

Subito dopo ho creato l'interfaccia grafica tramite modale:



Dopo aver creato e testato il sistema di promozione ho aggiunto dei controlli relativi ai permessi. Se un utente non può eseguire nessuna azione sugli utenti il collegamento dalla navbar sparisce e se si prova ad accedere alla pagina viene mostrato un errore a schermo.

Se invece, per esempio, l'utente non ha il permesso di promuovere utenti il tasto relativo alla promozione viene nascosto.



Figure 2 - Con tutti i permessi



Figure 2 - Senza il permesso "promote"

Ho aggiunto anche una sezione all'interno del pannello admin chiamata *Notifiche*, essa infatti mostra tutti gli errori generati durante le operazioni in back-end.

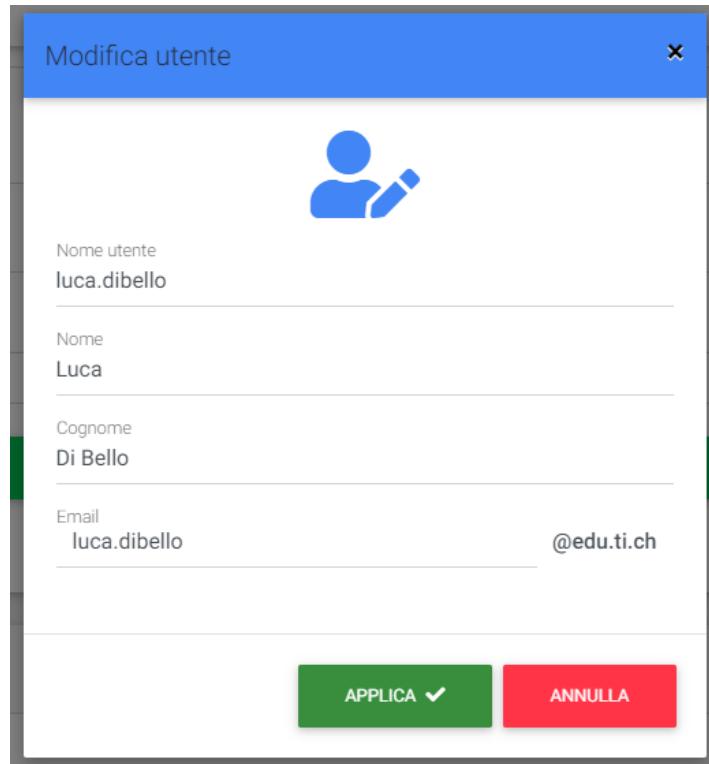
Per esempio, se un utente che non ha permessi di eliminare altri utenti cerca di eliminarne uno comunque chiamando le API viene mostrato questo errore:

## Notifiche

Non hai i permessi necessari per eliminare gli utenti

Ho terminato anche la modifica degli utenti. Se si prova a modificare lo username di un utente tutte le foreign key all'interno del database vengono ri-aggiornate per evitare dati orfani.

Questo è il modale per la modifica di un utente:



### Problemi riscontrati e soluzioni adottate

Ho scoperto (a mie spese) che JQuery non può selezionare un id con un '.' In mezzo.

Ogni riga della tabella ha un proprio identificativo (nome\_utente.cognome\_utente). Questo id contenendo un '.' Non era selezionabile tramite jQuery.

Per risolvere non ho fatto altro che utilizzare javascript puro per selezionare il record della tabella e poi jQuery per ricavare i dati:

```
// get row
let row = document.getElementById(username);
// read user permission name
let permission_name = $(row).find(".permessi").text();
```

### Punto della situazione rispetto alla pianificazione

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

### Programma di massima per la prossima giornata di lavoro

Aggiungere il permesso *update* all'interno del database e creare i relativi attributi e controlli nelle varie classi. Successivamente iniziare con l'implementazione della gestione riservazioni.

# Diario di lavoro

Luogo	Canobbio
Data	11.10.2019

## Lavori svolti

Durante le prime due ore di lezione il docente Valsangiacomo ha tenuto una presentazione PowerPoint dove spiegava come i vari periti andranno a valutare il nostro LPI (lavoro pratico di maturità).

Nella seconda ora di lezione ho inserito (come pianificato la scorsa lezione) il nuovo permesso al database. Questo nuovo permesso è stato anche integrato nella libreria *Permissions*, nella classe *PermissionManager* e *PermissionModel*.

Dopo ho controllato (sotto consiglio del docente Valsangiacomo) se il sito web funzionava su tutti i browser. Oggi ho controllato su Internet Explorer ed il sito risulta totalmente funzionante.

Subito dopo ho iniziato a creare la pagina per l'aggiunta delle prenotazioni (pagina accessibile agli utenti). Ho deciso di fare prima la pagina "pubblica" al posto di quella gestionale (per il pannello admin) dato che necessito di avere dei dati all'interno del database.

Preferisco avere già un sistema funzionante per l'inserimento dati al posto di dover inserire dei dati per testing a mano all'interno della tabella.

## Problemi riscontrati e soluzioni adottate

Non ho riscontrato alcun problema.

## Punto della situazione rispetto alla pianificazione

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

## Programma di massima per la prossima giornata di lavoro

Continuare la pagina per l'aggiunta delle prenotazioni.

# Diario di lavoro

Luogo	Canobbio
Data	15.10.2019

## Lavori svolti

Durante la prima parte della lezione ho fatto su un pezzo di carta un piccolo schizzo di come ho pensato la pagina di gestione e visualizzazione delle prenotazioni dell'utente.

Subito dopo ho iniziato ad implementare lo stesso sistema che ho utilizzato per la gestione degli utenti: caricamento dei dati del database in array di oggetti *Booking*.

Per far questo ho dovuto ovviamente creare un oggetto *Booking*, il quale salva al suo interno tutti i dati di ogni record, quindi:

- **Id**
- **DateTime inizio**
- **DateTime fine**
- **Possibili osservazioni (di default Null dato che è facoltativo)**
- **Username creatore**

Successivamente ho iniziato la creazione del model (*BookingModel*) che verrà utilizzato per la lettura ed il *parsing* dei dati salvati all'interno del database.

Dopo aver terminato e testato il corretto funzionamento del model ho aggiunto una piccola tabella che mostra tutte le prenotazioni che ha effettuato:

Prenotazioni personali				
Data	Ora inizio	Ora fine	Osservazioni	Opzioni
2019-10-15	08:50:00	09:50:00	<b>MOSTRA OSSERVAZIONI</b>	<b>AZIONI DA FINIRE</b>

© 2019 Copyright CPT Meeting Room Scheduler

Figure 1 - Prenotazioni dell'utente 'luca.dibello'

## Problemi riscontrati e soluzioni adottate

Non ho riscontrato alcun problema dato che il codice è praticamente identico a quello del pannello admin (gestione utenti).

Durante l'ultima ora di lezione sono stato assente. Sono dovuto andare al palazzo dei congressi di Lugano a vedere una conferenza relativa al 5G.

Punto della situazione rispetto alla pianificazione

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

Programma di massima per la prossima giornata di lavoro

Aggiungere funzione che permette all'utente di aggiungere una prenotazione.

# Diario di lavoro

Luogo	Canobbio
Data	17.10.2019

## Lavori svolti

Durante le prime due ore di lezione ho aggiunto un nuovo *handler* nella classe API che verrà utilizzato per le operazioni sui dati relativi alle prenotazioni.

```
/*
 * Action url: api/booking/add
 * Permission needed: visione_prenotazioni
 * Extra data: none
 */
elseif($action == "add" && $_SERVER["REQUEST_METHOD"] == "POST"){
    if(PermissionManager::getPermissions()->canInserirePrenotazioni()){
        // Sanitize POST data and promote user
        $result = BookingModel::add(filter_input_array( type: INPUT_POST, definition: FILTER_SANITIZE_STRING));

        // If it detects errors
        if(is_array($result)){
            $GLOBALS["NOTIFIER"]->add_all($result);
        }
    }
} else{
    $GLOBALS["NOTIFIER"]->add("Non hai i permessi necessari per inserire le prenotazioni");
}

RedirectManager::redirect( path: "home/prenotazioni");
```

Come si può vedere nello screenshot il metodo controlla anche se l'utente ha effettivamente i permessi necessari per l'utilizzo di questa funzione (l'utente necessita i permessi per l'inserimento di prenotazioni).

Dopo il controllo dei permessi utilizzo il nuovo metodo *add* creato all'interno della classe *BookingModel*. Questo nuovo metodo non fa altro che prendere i dati ricevuti dalle API, validarli e, se tutto va a buon fine, inserire i dati all'interno del database. Questo è il codice:

```
/**
 * This method is used to add a new booking to the database.
 *
 * @param array $data Booking data array
 * @return array|bool Return array if the booking was added to the database, otherwise it returns an error as an
 * array
 */
public static function add(array $data){
    // Check booking data
    if(self::validateBookingData($data)){
        // Insert data into database
        $result = DB::insert('riservazione', array(
            'data' => $data['dataInizio']->format(BOOKING_DATE_FORMAT),
            'ora_inizio' => $data['dataInizio']->format(BOOKING_TIME_FORMAT),
            'ora_fine' => $data['dataFine']->format(BOOKING_TIME_FORMAT),
            'osservazioni' => ($data['osservazioni'] ? null : $data['osservazioni']),
            'utente' => $_SESSION['username']
        ));

        // Return true (if the insert query was successful) otherwise it returns an error as array
        return (!result ? array("C'è stato un errore durante l'inserimento dei dati
            all'interno del database. Contattare un amministratore.") : true);
    }
    // Booking data not valid
    else{
        $GLOBALS["NOTIFIER"]->add_all(self::$errors);
        RedirectManager::redirect( path: "admin/utenti");
    }
}
```

Successivamente ho aiutato per una quindicina di minuti Mattia Toscanelli e Mattia Lazzaroni per delle parti di codice.

Prima della fine della lezione mi sono ricordato di un requisito: *ci deve sempre essere almeno un amministratore*. Dato che nella gestione utenti non avevo implementato questo controllo ho creato un metodo nella classe *UserModel* che conta tutti gli amministratori presenti all'interno del database.

### Problemi riscontrati e soluzioni adottate

Non ho riscontrato nessun problema

### Punto della situazione rispetto alla pianificazione

Sono in anticipo in anticipo rispetto alla pianificazione iniziale.

### Programma di massima per la prossima giornata di lavoro

Creare il form per aggiungere gli utenti e testare il sistema.

# Diario di lavoro

Luogo	Canobbio
Data	18.10.2019

## Lavori svolti

Durante la prima ora di lezione ho aggiunto il controllo per l'eliminazione degli amministratori. Questo è il codice che ho aggiunto alla classe *UserModel* all'interno del metodo *delete*:

```
/* API METHODS */
    public static function delete(string $username) {

        // Internal function
        function delete_user($username) {
            $result = DB::delete("utente", "username=%s", $username);
            return (!$result ? array("C'Ã" stato un errore durante
l'eliminazione dell'utente.
                    Contattare un amministratore.") : true);
        }

        // Read user permission group
        $perm_name = PermissionModel::getUserPermissionGroup($username);

        // Check if user is admin
        if($perm_name == ADMIN_PERMISSION_GROUP){
            if(self::countAdmins() > MINIMUM_ADMIN_ALLOWED){
                // Calls delete internal function
                return delete_user($username);
            }
            else{
                return array("Ci devono essere almeno ".
MINIMUM_ADMIN_ALLOWED . " admin(s) nel sistema. Impossibile
                    eliminare l'account");
            }
        }
        else{
            // Calls delete internal function
            return delete_user($username);
        }
    }
}
```

Come si può vedere questo metodo quando rileva l'eliminazione di un admin utilizza il metodo *countAdmins()* per contare quanti admins sono presenti all'interno del database. Se gli admin contati sono maggiori di *MINIMUM\_ADMIN\_ALLOWED* (che di default ha valore 1) l'utente viene eliminato, altrimenti viene mostrato un errore all'interno della pagina e

I'operazione di eliminazione viene annullata.

Durante la seconda ora di lezione invece ho avuto un colloquio con il docente Valsangiacomo. Durante il colloquio mi è stato consegnato il nuovo QdC e mi è stato illustrato con precisione cosa devo migliorare sia a livello di documentazioni che a livello di diari e pianificazione (*Gantt*).

Mi è stato fatto notare che manca qualche requisito all'interno della documentazione e che manca l'attività relativa all'implementazione del *Raspberry* all'interno del diagramma di Gantt.

Mi è stato anche riferito anche che dalla settimana prossima in poi il venerdì oltre ad inviare insieme al diario di lavoro la documentazione, tutto il codice del sito dovrà essere messo su un server Infomaniak.

Durante le due ore seguenti ho continuato la documentazione, iniziando a scrivere il primo capitolo che spiega come ho implementato la gestione dei permessi all'interno del sistema. Questo capitolo spiega anche la struttura delle classi *PermissionManager* e *Permissions* dato che sono le classi principali che utilizzo per la lettura dei dati.

#### Problemi riscontrati e soluzioni adottate

Durante il colloquio mi è stato fatto notare che sto facendo diversi errori durante questo progetto. Per esempio il fatto di aggiungere contenuti alla doc troppo raramente.

#### Punto della situazione rispetto alla pianificazione

Sono in anticipo rispetto alla pianificazione iniziale. Da ora in poi parte della lezione verranno sfruttate per la stesura della documentazione.

#### Programma di massima per la prossima giornata di lavoro

Creare il form per aggiungere gli utenti e testare il sistema e continuazione documentazione finendo di scrivere il capitolo relativo alla gestione permessi.

# Diario di lavoro

Luogo	Canobbio
Data	22.10.2019

## Lavori svolti

Durante la giornata di oggi non ho potuto lavorare al progetto. Sono passato a Linux e durante la lezione ho impostato tutti i vari programmi e *tools* utili per la continuazione del progetto.

Queste sono le informazioni relative al mio sistema operativo corrente:

```
luca@luca-rog:~$ screenfetch
      ./+o+-
      yyyyyy- -yyyyyy+
      ://+=====--yyyyyyo
      .++ .:/++++++/-+.+sss/`+
      .:+o: /+++++++/:-:/-+
      o:+o:+o:+. `..`.-/oo+++++/
      .:+o:+o:/ . `+sssooo+/`+
      .++/+:+oo+o: `/sssooo.
      /+++/++: `oo+o /::--:.
      \+/+o+++ `o++o `++///.
      .++o++++oo+: `/ddhhh.
      .+o+oo:. `oddhhhh+
      \+..++o+o``-`.:ohdhhhhh+
      `:o+++ `ohhhhhhhhyo++os:
      .o: `syhhhhhhh/.oo++o`+
      /osyyyyyyo++ooo+++/`+
      ````+oo++o\:
      `oo++.

luca@luca-rog:~$
```

Ho installato l'IDE (PhpStorm 2019.2), Apache, MySQL Server, PHP 7.4. Mi manca ancora qualche programma da installare (VMWare).

Per scrivere questo documento sto utilizzando WPS Writer (il *Word* del pacchetto *WPS Office*).

**Problemi riscontrati e soluzioni adottate**

Ho installato PHP, Apache e MySQL server. Il problema è che il servizio di mysql non mi ha chiesto nè password nè lo username quindi non riesco ad accedere al servizio.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo di un po' di ore rispetto alla pianificazione iniziale.

**Programma di massima per la prossima giornata di lavoro**

Continuare il progetto e completare gli obiettivi pianificati la volta scorsa.

# Diario di lavoro

Luogo	Canobbio
Data	24.10.2019

## Lavori svolti

Durante la giornata di oggi mi sono prevalentemente concentrato sulla documentazione. Ho completato il sottocapitolo dell'implementazione relativo alla sicurezza spiegando i metodi principali per il caricamento ed il controllo dei permessi all'interno del sistema. Devo ancora aggiungere delle immagini e lo schema della tabella "tipo\_utente" per mostrare come vengono salvati i dati all'interno del database.

Successivamente ho impostato il webserver, installando il ed abilitando il modulo di apache "*mod\_rewrite*", utile per abilitare la *RewriteRule* del file *.htaccess* (utilizzato per il template MVC).

Dopo ho installato le estensioni di PHPUtili per la connessione con il servizio LDAP (php7.2\_ldap) e la estensione per mysqli per la connessione con il database, la quale è necessaria per il corretto funzionamento di MeekroDB (la libreria che uso per la connessione al database).

Linux non installa nulla automaticamente, infatti ho dovuto installare tutti i moduli ed estensioni manualmente tramite linea di comando.

Successivamente ho installato il dizionario italiano (*language pack*) all'interno di WPS Office ed ho installato FileZilla per testare le credenziali del server Infomaniak inviatomi dal docente Valsangiacomo due settimane fa.

**Problemi riscontrati e soluzioni adottate**

Sono riuscito a mettere apposto il problema relativo al server MySQL. L'errore non era causato dal fatto che utilizzavo il comando senza i permessi di amministratore (non mettevo *sudo* davanti al comando).

Ho scoperto che MySQL (su linux) quando installato crea un utente base chiamato "*root*" senza password. Per accedere con questo utente, essendo senza password, si necessita di avere i permessi di amministratore (e quindi di sapere la password del sistema). Per risolvere questo problema ho digitato il seguente comando per impostare una password all'utente root:

```
ALTER USER 'user-name'@'localhost' IDENTIFIED BY 'NEW_USER_PASSWORD';
```

E poi ho digitato il comando per far ricaricare tutti i permessi degli utenti dal DBMS:

```
FLUSH PRIVILEGES;
```

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo alla pianificazione iniziale.

**Programma di massima per la prossima giornata di lavoro**

La prossima lezione entrerò con Windows per fare il *dump* del database e poi continuerò la documentazione.

# Diario di lavoro

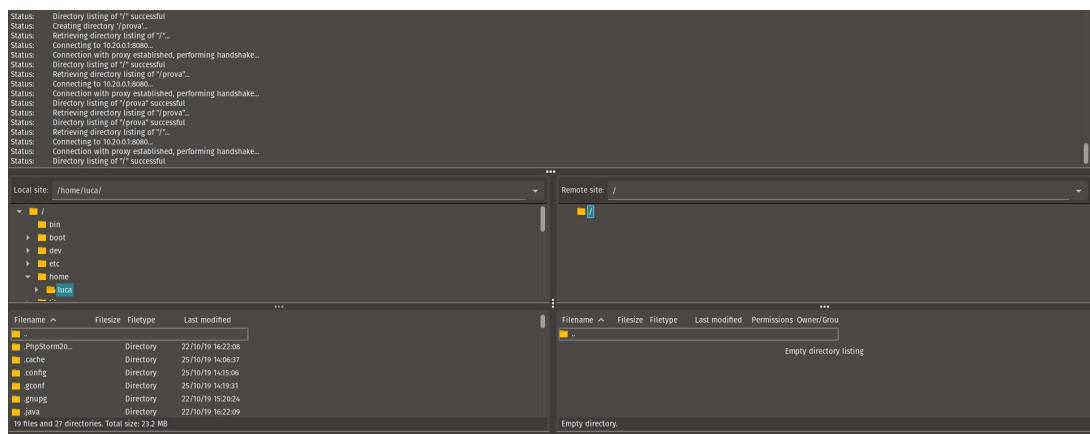
Luogo	Canobbio
Data	25.10.2019

## Lavori svolti

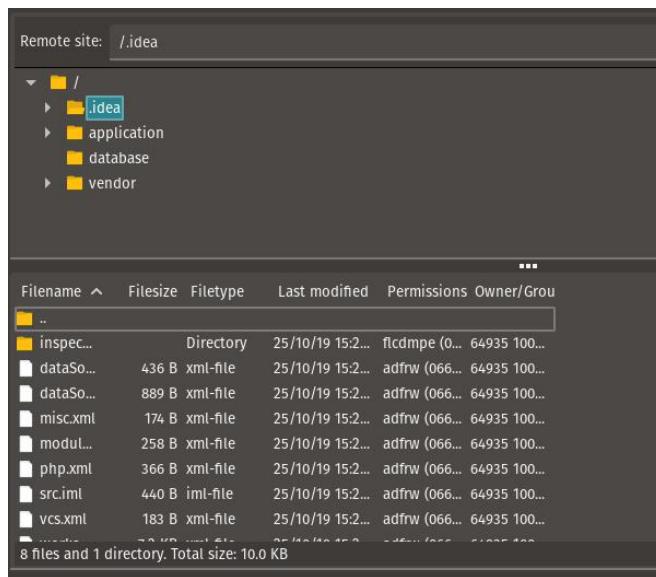
Durante le prime di ore di lezione ho installato sul mio sistema operativo una nuova shell: ZSH.

Questa shell mi permette di essere molto più produttivo dato che permette di installare dei *plugins* che offrono molteplici funzionalità specifiche per gli sviluppatori.

Durante la seconda ora di lezione ho impostato *FileZilla* per passare attraverso il proxy utilizzando il tutorial fornитoci dal docente Valsangiacomo su Moodle.



Una volta impostato ho caricato tutti i miei file sul Server:



Dopo questo ho continuato la documentazione aggiungendo immagini e testo.

**Problemi riscontrati e soluzioni adottate**

Avevo un problema con la *repository* di GitHub: non riuscivo ad eseguire nessuna operazione (pull, fetch, push, ...). Il problema era che, sovrascrivendo i file di git presenti nella cartella, non riusciva più ad eseguire nessuna operazione sulla repo.

Per risolvere questo problema non ho fatto altro che clonare nuovamente la repository ed poi inserire i file che ho modificato la scorsa lezione.

**Punto della situazione rispetto alla pianificazione**

Sono in anticipo di una decina di giorni rispetto alla pianificazione. In questo momento (secondo il Gantt) dovrei essere circa a metà dell'implementazione del sistema di gestione degli utenti.

Sfrutterò questo tempo per continuare la documentazione dato che sono piuttosto indietro.

**Programma di massima per la prossima giornata di lavoro**

Continuare la documentazione

# Diario di lavoro

Luogo	Canobbio
Data	05.11.2019

## Lavori svolti

Durante la lezione di oggi ho continuato la documentazione ed ho recuperato il database dell'applicativo. Per recuperare il database dei dati non ho fatto altro che fare un *dump* del tutto..

Ho potuto lavorare soltanto le prime due ore di lezione dato che ho avuto un impegno ed ho dovuto assentarmi.

**Problemi riscontrati e soluzioni adottate**

Durante la lezione odierna non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono leggermente in anticipo rispetto alla pianificazione iniziale. Oggi infatti, secondo la pianificazione, avrei dovuto iniziare la pagina di gestione delle prenotazioni ma io ho già svolto qualche lavoro.

**Programma di massima per la prossima giornata di lavoro**

Continuare la documentazione

# Diario di lavoro

Luogo	Canobbio
Data	07.11.2019

## Lavori svolti

Durante la lezione di oggi ho fatto l'*import* del database. Una volta importato ho dovuto aggiungere i dati relativi ai tipi di utenti (admin, user e avanzato) dato che, stranamente, non sono stati esportati correttamente.

Per la visualizzazione delle prenotazioni ho deciso di utilizzare una libreria JavaScript chiamata *FullCalendar*. Essa permette molteplici funzioni e molta customizzazione. Esso legge i dati sotto forma di JSON quindi ho dovuto creare una classe che permetteva la conversione degli oggetti *Booking* in JSON.

La classe model che esegue questa operazione è chiamata *CalendarModal* e permette di convertire uno o più oggetti *Booking* in JSON. L'output JSON è già formattato in modo da poter essere letto e riconosciuto dalla libreria.

Esempio di output JSON:

```
[  
  {  
    "id":1,  
    "title":"Luca Di Bello",  
    "start":"11-07-2019 15:30",  
    "end":"11-07-2019 16:00",  
    "note":"Questo evento è stato creato per prova"  
  }  
]
```

Come si può vedere chiaramente dallo spezzone di JSON generato il titolo dell'evento è il nome ed il cognome del professore che lo ha creato. Questo perchè in questo modo si può vedere a colpo d'occhio di chi è un determinato evento: Il titolo viene sempre mostrato, anche senza cliccare sull'evento.

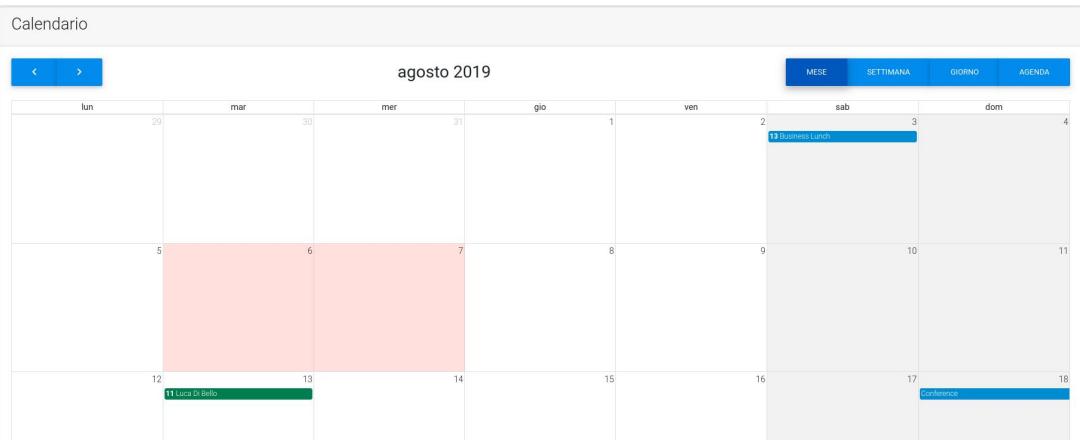
Esempio di evento nel calendario:



Dopo aver fatto questo ho iniziato a giocare con le impostazioni della libreria per capirne la logica. Dopo averci giochicchiato un po' infatti sono riuscito a cambiare tema al calendario facendo utilizzare il tema material design che offre *Material Design Bootstrap*.

Sono riuscito anche a impostare la lingua italiana. È importante che il calendario sia in italiano dato che non tutti gli utilizzatori del sito web palleggeranno l'inglese.

Questo è come il calendario è visualizzabile dall'utente:



(Nota: come si può vedere il calendario risulta molto grande, devo trovare un modo per rimpicciolirlo e renderlo responsivo anche su dispositivi mobili/con schermi più piccoli)

**Problemi riscontrati e soluzioni adottate**

Durante la lezione odierna non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono al passo con la pianificazione. Secondo la pianificazione iniziale in questo momento sono all'attività relativa allo sviluppo della gestione delle prenotazioni.

**Programma di massima per la prossima giornata di lavoro**

Continuare il calendario caricando direttamente gli eventi del database tramite la classe *CalendarModel* e *BookingModel*. Dovrò anche aggiungere delle impostazioni per semplificare la visualizzazione del calendario (es: colori diversi per ogni professore).

# Diario di lavoro

Luogo	Canobbio
Data	11.11.2019

## Lavori svolti

Come prima cosa ho creato un nuovo controller, il quale gestirà la pagina del calendario.

Dopo aver spostato il calendario in una nuova pagina ho scritto le API che verranno utilizzate da Raspberry Pi per leggere tutte le riservazioni presenti nel sistema. Non è concesso a tutti l'utilizzo di queste API dato che è necessario un *token* di accesso. Questo token è salvato all'interno della configurazione (file *config/config.php*) all'interno dell'applicativo web.

Per testare il tutto ho scritto un piccolo script in *Python* che non fa altro che eseguire una richiesta POST alle API e leggerne la risposta (la quale è sotto forma di JSON). Questo è lo script:

```
import requests  
  
_TOKEN="058c24b04169e44528ff2be1ac83f5dd787aa2109ad64fdcf142538f4d8617b583  
2e532a7c4a004398c3a3b4f12d1eac47423680fd71c02105d33c77cae12d5d"  
  
r = requests.post("http://localhost:8123/api/calendar", data={"token": _TOKEN})  
  
print("Server response:")  
  
print(r.text)
```

Ho aggiunto anche le tooltip sui pulsanti per l'eliminazione, modifica e promozione degli utenti all'interno del pannello admin

Cose da fare per settimana prossima:

- Aggiungere nel Gantt l'attività relativa all'implementazione del Raspberry Pi

**Problemi riscontrati e soluzioni adottate**

Durante la lezione odierna non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono al passo con la pianificazione. Secondo la pianificazione iniziale in questo momento sono all'attività relativa allo sviluppo della gestione delle prenotazioni.

**Programma di massima per la prossima giornata di lavoro**

Aggiungere le funzioni di modifica e creazione di riservazioni (lato utente).

# Diario di lavoro

Luogo	Canobbio
Data	12.11.2019

## Lavori svolti

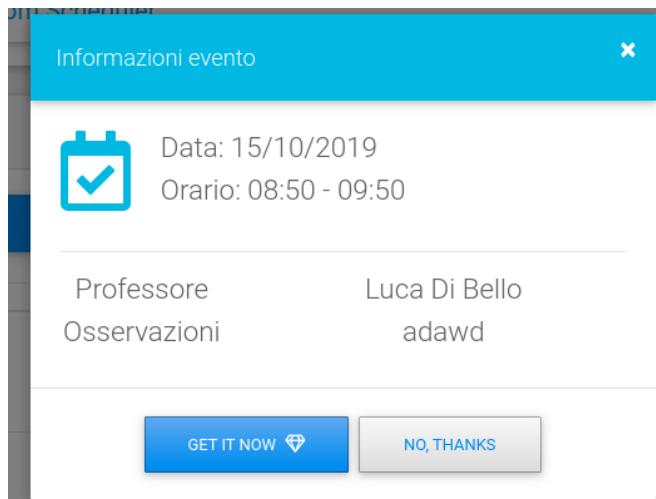
Ho deciso di seguire il consiglio del professor Valsangiacomo: gli utenti del sito vedranno e gestiranno tutte le prenotazioni con il calendario interattivo.

Oggi sono riuscito a far caricare i dati JSON tramite una richiesta AJAX. La richiesta non fa altro che andare ad utilizzare l'API del calendario, questo è il codice che esegue questa operazione:

```
$ajax({
  // Calls the calendar api
  url: '/api/calendar',
  method: 'POST',
  dataType: 'json',
  data: {
    token: "<?php echo API_TOKEN; ?>" // Setup API authentication code
  },
  success: function(doc) {
    ...
  });
});
```

Si può vedere che insieme alla richiesta invio un parametro chiamato "TOKEN": esso servirà per autenticarsi e quindi utilizzare le API. Se non venisse passato oppure se il token fosse errato le API ritornerebbero un JSON che descrive l'errore.

Quando si clicca un evento presente sul calendario viene mostrato un *modal* che mostra delle informazioni dettagliate sull'evento. Dovrò aggiungere i pulsanti per eliminare l'evento:





**Problemi riscontrati e soluzioni adottate**

Ho eseguito un aggiornamento di sistema alla versione 19.10. Dopo questo aggiornamento apache2 non mi rilevava php. Per risolvere ho dovuto fare il seguente comando nel terminale:  
e2enmod php7.3

Dopo aver abilitato php ho dovuto riavviare il servizio.

**Punto della situazione rispetto alla pianificazione**

Sono al passo con la pianificazione. Secondo la pianificazione iniziale in questo momento sono all'attività relativa allo sviluppo della gestione delle prenotazioni.

**Programma di massima per la prossima giornata di lavoro**

Aggiungere la funzione di eliminazione all'interno del modale e poi aggiungere delle funzioni al calendario (es: inserimento)

# Diario di lavoro

Luogo	Canobbio
Data	14.11.2019

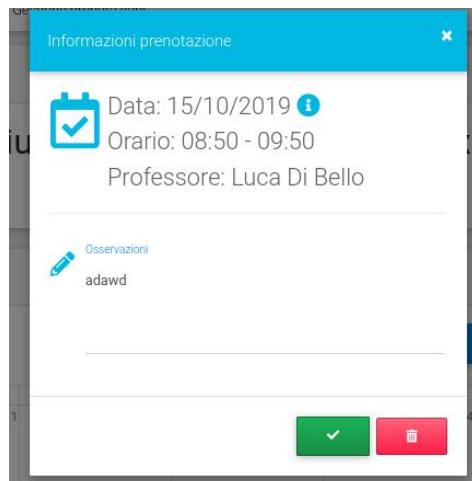
## Lavori svolti

Come prima cosa oggi ho corretto delle funzioni presenti nelle API che non avevano il controllo dei permessi. Per esempio il metodo *delete* dell'API dei *booking* non controllava se l'utente che eseguiva la richiesta era effettivamente loggato o meno.

Ho modificato il modale di visualizzazione della prenotazione cambiando i pulsanti ed aggiungendo i *tooltips*.

Sto ancora decidendo se aggiungere all'interno dello stesso modale la funzione di modifica oppure un pulsante che apre un altro modale utilizzato solo ed esclusivamente per la modifica dei dati.

Per adesso questo è il design che ho scelto:



Dopo aver corretto questo ho iniziato ad informarmi su come fare ad inserire e modificare gli eventi tramite il calendario. Ho scoperto che quando si effettua un click su una casella del calendario esso genera un evento *dateClick*. Tramite questo evento sarò in grado di aprire un modale per la creazione di un nuovo evento sapendo già il giorno scelto dall'utente

**Problemi riscontrati e soluzioni adottate**

Rileggendo il QDC mi sono accorto che mancano dei permessi all'interno della tabella "tipo\_utente". Io per adesso ho soltanto il permesso chiamato "eliminazione\_prenotazioni\_altri\_utenti" però non ho quello per l'inserimento e la modifica.

**Punto della situazione rispetto alla pianificazione**

Sono al passo con la pianificazione. Secondo la pianificazione iniziale in questo momento sono all'attività relativa allo sviluppo della gestione delle prenotazioni.

**Programma di massima per la prossima giornata di lavoro**

Creazione Virtual Machine con Gantt project e finire modale di inserimento.

# Diario di lavoro

Luogo	Canobbio
Data	15.11.2019

## Lavori svolti

Come prima cosa ho creato una virtual machine con Windows 10. Su di essa ci ho installato *Microsoft Project* e *Microsoft Visio*.

Dopo aver installato il tutto ho modificato il mio Gantt vecchio inserendo l'attività relativa al raspberry (configurazione + implementazione).

Successivamente ho continuato la documentazione fino alle 16:00, continuato il capitolo dell'implementazione.

L'ultima mezz'ora l'ho utilizzata per mettere i nuovi *files* sul server ed aggiornare il database.

**Problemi riscontrati e soluzioni adottate**

Non ho ancora messo a posto il problema relativo ai permessi.

**Punto della situazione rispetto alla pianificazione**

Sono al passo con la pianificazione. Secondo la pianificazione iniziale in questo momento sono all'attività relativa allo sviluppo della gestione delle prenotazioni.

**Programma di massima per la prossima giornata di lavoro**

Continuare il calendario.

# Diario di lavoro

Luogo	Canobbio
Data	19.11.2019

## Lavori svolti

Durante la giornata di oggi ho aggiunto la funzione di creazione ed eliminazione di eventi al calendario.

La funzione di eliminazione viene chiamata tramite una richiesta AJAX:

```
$.ajax({
    type: "POST",
    url: "/api/booking/delete/" + event.id,
    success: function (result) {
        if(result["success"]){
            console.log("[] event deleted");

            calendar.refetchEvents();
            $('#eventInfo').modal('hide');

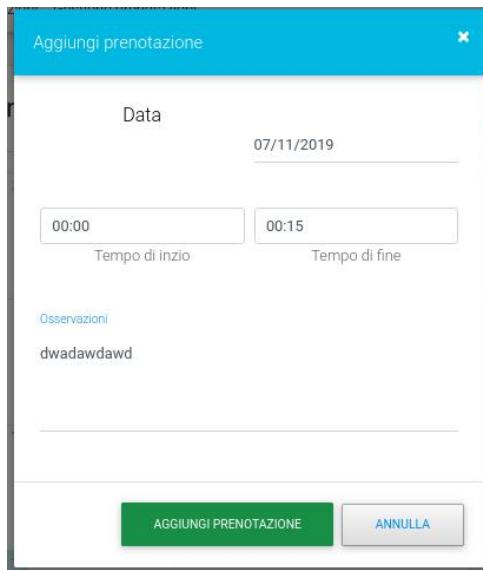
            // Notify success
            $.notify("Evento eliminato con successo", "success");
        }
        else{
            console.log(result["errors"]);

            result["errors"].forEach(function(item, index, arr){
                $.notify(item, "warn");
            });
            $('#eventInfo').modal('hide');
        }
    },
    dataType: "json"
});
```

Se le API rispondono in modo positivo (quindi sono riuscite ad eseguire l'operazione) mostra una notifica con scritto “Evento eliminato con successo”. Quando le API invece producono degli errori, essi vengono mostrati sempre tramite notifiche.

In questo modo il calendario risulta molto fluido e molto bello da utilizzare. Sarà complicato renderlo utilizzabile da telefono ma riuscirò a trovare una soluzione.

Questo è il form utilizzato per l'aggiunta degli eventi:



The form is titled "Aggiungi prenotazione". It contains the following fields:

- Data: 07/11/2019
- Tempo di inizio: 00:00
- Tempo di fine: 00:15
- Osservazioni: dwadawdawd

At the bottom are two buttons: "AGGIUNGI PRENOTAZIONE" (green background) and "ANNULLA" (grey background).

(Nota: La grafica è ancora da cambiare, non mi piace lo stile)

Per adesso l'inserimento di un evento necessita il caricamento della pagina, durante la prossima lezione proverò ad implementarlo come con l'eliminazione, quindi tramite una richiesta POST con AJAX.

**Problemi riscontrati e soluzioni adottate**

Non ho ancora messo a posto il problema relativo ai permessi. Durante le prossime lezioni (quando avrò la necessità di utilizzare questo permesso) metterò le notifiche.

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione. Il 15 novembre avrei dovuto già finire l'implementazione del calendario.

Questo ritardo è dovuto al fatto che il docente Valsangiacomo mi ha spinto ad utilizzare il calendario interattivo per la gestione delle prenotazioni

**Programma di massima per la prossima giornata di lavoro**

Riprogrammare le API in modo che siano compatibili con le richieste AJAX ed inserire la funzionalità che permette di modificare le prenotazioni già presenti.

# Diario di lavoro

Luogo	Canobbio
Data	21.11.2019

## Lavori svolti

Durante la lezione di oggi sono riuscito ad eseguire anche l'inserimento dei dati con AJAX. Questa operazione mi è costata molto tempo dato che ho dovuto riscrivere tutta la parte JavaScript che costruisce la richiesta e ne analizza la risposta.

Dopo questo ho inserito i nuovi permessi all'interno della tabella MySQL, questi sono i nuovi permessi aggiunti:

- Inserimento\_prenotazioni\_altri\_utenti
- Modifica\_prenotazioni
- Modifica\_prenotazioni\_altri\_utenti

Ovviamente dopo aver aggiunto questi permessi ho dovuto anche aggiungerli nella classe *Booking* e nel model *BookingModel* per il corretto funzionamento di essi.

Successivamente ho iniziato a lavorare sulla modifica dei dati presenti, creando una nuova api accessibile tramite questo URL:

[http://localhost:8123/api/booking/update/<id\\_del\\_booking\\_da\\_modificare>](http://localhost:8123/api/booking/update/<id_del_booking_da_modificare>)

Questa API va contattata tramite POST con tutti i dati da modificare al booking.

Ho scritto le API ma non sono funzionanti. Per testarle utilizzerò un programma chiamato *Postman* il quale è utilizzato per questo genere di situazioni.

**Problemi riscontrati e soluzioni adottate**

C'è un problema nell'api relativa alla modifica delle prenotazioni. Ho scaricato un programma chiamato *Postman* che mi sarà utile per trovare il problema..

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione. Il 15 novembre avrei dovuto già finire l'implementazione del calendario.

Questo ritardo è dovuto al fatto che il docente Valsangiacomo mi ha spinto ad utilizzare il calendario interattivo per la gestione delle prenotazioni

**Programma di massima per la prossima giornata di lavoro**

Finire la funzione di modifica dei dati ed inserire la funzione che permette di fare il drag-and-drop degli eventi sul calendario per modificare la data/ora di esso.

# Diario di lavoro

Luogo	Canobbio
Data	22.11.2019

## Lavori svolti

Durante la giornata di lavoro di oggi sono riuscito a completare il calendario.

Ora le funzioni di drag-and-drop, modifica orario, eliminazione ed inserimento sono completamente funzionanti.

Dopo aver fatto questo ho testato il sistema per mezz'oretta circa per assicurarmi che il tutto funzioni correttamente. Per gestire l'overlapping degli eventi ho abilitato un'opzione già presente nel calendario chiamata 'eventOverlap'.

Durante l'ultima parte della lezione ho caricato il nuovo database sul server ed ho caricato le modifiche al sito. Caricando il sito ho potuto notare che i link non sono giusti e che quindi non carica né css né script php correttamente.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione. Il 15 novembre avrei dovuto già finire l'implementazione del calendario.

**Programma di massima per la prossima giornata di lavoro**

Fare i controlli da backend dell'overlap degli eventi.

# Diario di lavoro

Luogo	Canobbio
Data	26.11.2019

## Lavori svolti

Ho utilizzato questa giornata di lavoro per fix di bug ed aggiunta di controlli.

Come prima cosa ho aggiunto dei controlli al calendario: ora non è più possibile inserire date nel passato. Successivamente, debuggando il sistema di login, mi sono accorto che LDAP non riusciva a connettersi al servizio LDAP di CPT. Questo perchè non riusciva a risolvere il nome *cpt.local* in 10.20.4.1.

Dopo questo ho iniziato a scrivere una query SQL che mi ritorna i record che hanno un overlap tra di loro (quindi che sono sovrapposti). Questa query mi tornerà utile per controllare se uno slot orario è già occupato o meno da un altro evento.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione iniziale. Ho quasi finito l'attività relativa alla gestione delle prenotazioni

**Programma di massima per la prossima giornata di lavoro**

Finire e testare il sistema di controllo dell'overlap degli eventi.

# Diario di lavoro

Luogo	Canobbio
Data	28.11.2019

## Lavori svolti

Durante la giornata di oggi sono riuscito a completare il calendario, e quindi l'attività riguardante la gestione delle prenotazioni.

Come prima cosa sono riuscito a risolvere il problema relativo alla query. Il problema non era la query SQL ma io che facevo un controllo sbagliato.

Quando un oggetto viene ridimensionato nella GUI non viene subito modificato all'interno del database ma passava attraverso la classe *BookingValidator*, il quale compito è quello di controllare che i dati siano corretti e sensati.

Io controllavo anche l'*overlap* (sovraposizione) delle prenotazioni, quindi controllavo che non c'era nessun evento in quel range di tempo. Il problema sta nel fatto che la prenotazione era ancora salvata nel database con la data di inizio identica a quella del nuovo *booking* inviato alle API, quindi risultava che c'era già un evento in quel lasso di tempo.

Dopo aver messo apposto questo problema ho modificato la classe *BookingValidator*, modificando il metodo *validateDateTime* (il quale svolgeva molti controlli) in dei metodi più specifici, questo per permettere di isolare il problema e mostrare un messaggio di errore più significativo all'utente.

Questi sono tutti i messaggi di errore disponibili nella pagina del calendario:

1. L'ora di inizio non può essere maggiore di quella di fine
2. Non è possibile fare una prenotazione su giorni/orari già passati
3. Non puoi fare una prenotazione che si estende su più giorni
4. Orario non valido. L'orario scelto è in sovrapposizione con un'altra prenotazione
5. La descrizione è troppo lunga

Ho anche inserito un opzione nel file *config.php* che permette di scegliere i giorni bloccati nel calendario:

```
define('BOOKING_HIDDEN_DAYS', array(
    DaysOfWeek::Sunday,
    DaysOfWeek::Saturday
));
```

Con queste impostazioni all'interno del calendario vengono nascosti il *week-end* (sabato e domenica). Le impostazioni vengono caricate con questo codice PHP all'interno del costruttore JavaScript:

```
hiddenDays: <?php echo "[" . implode(", ", BOOKING_HIDDEN_DAYS) . "]"?>,
```

Con questo codice i giorni inseriti nella costante *BOOKING\_HIDDEN\_DAYS* vengono formattati

come un array JavaScript, quindi, dopo che il codice PHP viene eseguito il valore è questo:

hiddenDays: [6,1]

Subito dopo ho acceso il Raspberry Pi ed ho configurato il layout della tastiera (ho messo layout italiano dato che il layout svizzero francese non è disponibile stranamente) ed ho attivato il servizio ssh per connettermi dal mio portatile. Ho anche modificato l'hostname del raspberry, chiamandolo *CPTMRS* (CPT Meeting Room Scheduler) proprio come il nome del progetto.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato nessun problema.

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione iniziale. Ho quasi finito l'attività relativa alla gestione delle prenotazioni

**Programma di massima per la prossima giornata di lavoro**

Finire e testare il sistema di controllo dell'overlap degli eventi.

# Diario di lavoro

Luogo	Canobbio
Data	28.11.2019

## Lavori svolti

Durante la giornata di oggi mi sono concentrato sul testing del calendario. Ho provato a renderlo responsive, e quindi utilizzabile su telefonino, ma non sono ancora riuscito ad avere dei risultati buoni.

Ho anche impostato un nuovo utente root per il raspberry: “*luca*” ed ho messo un ip fisso per l’interfaccia ethernet: 10.20.4.50.

**Problemi riscontrati e soluzioni adottate**

Non sono riuscito a rendere responsivo il calendario.

**Punto della situazione rispetto alla pianificazione**

Sono in ritardo rispetto alla pianificazione iniziale. Ho quasi finito l'attività relativa alla gestione delle prenotazioni

**Programma di massima per la prossima giornata di lavoro**

Creare la pagina da utilizzare con il raspberry Pi, la quale leggerà e mostrerà i dati presenti nel database.

# Diario di lavoro

Luogo	Canobbio
Data	03.12.2019

## Lavori svolti

Durante il pranzo mi è venuto in mente (fortunatamente) che ho installato la versione di Raspbian (OS del Raspberry). La versione che ho installato e configurato non ha la possibilità di avere una GUI.

Ho dovuto installare BalenaEtcher (programma utile per il *burn* di ISO su SD/USB/CD/...) ma ho avuto problemi durante il download della versione *Rasbian Desktop*. La connessione della scuola dopo un po' si interrompeva, probabilmente perchè stavo generando troppo traffico.

Per ovviare a questo problema questa sera porterò il Raspberry e la scheda SD a casa mia, almeno posso installare il sistema operativo e fare gli aggiornamenti senza il problema del proxy.

Dopo aver fatto questo ho messo apposto un piccolo problema che ho notato durante una piccola fase di test che ho svolto ad inizio lezione il quale è spiegato nello specifico nella sezione *Problemi riscontrati e soluzioni adottate*.

Dopo aver risolto questo problema ho iniziato a lavorare per la generazione di report. Per far questo ho creato inizialmente due classi:

1. ReportGenerator: classe che riceve due impostazioni:
  - a) Tipo di report (il quale viene selezionato tramite la classe ReportType)
  - b) Flag che permette di mostrare anche prenotazioni vecchie. (VEDI NOTA IN FONDO AL PARAGRAFO)
2. ReportType: Classe che funge da enumeratore (Enum) il quale contiene tutti i possibili tipi di report implementati nel sistema:
  - a) Report giornaliero
  - b) Report settimanale
  - c) Report mensile
  - d) Report annuale

Nota: Di default vengono scritti nel report soltanto le prenotazioni future, se il flag viene impostato a *true* nel documento vengono mostrate anche le prenotazioni passate (mostra tutti i dati salvati nel database relativi a quel giorno/settimana/mese o anno)

Dopo aver creato quelle due classi ho scaricato tramite composer la libreria FPDF (package: "fpdf/fpdf") all'ultima versione ed ho seguito un esempio disponibile sulla documentazione ufficiale per la generazione di tabelle utilizzando dati dal database (vedi <http://www.fpdf.org/en/script/script14.php>).

Dopo aver modificato questo esempio, cambiando font, stile ed aggiungendo informazioni utili (data di generazione del report ed utente che ha generato il report)

sono arrivato a questo risultato:

<b>Report aula riunioni</b>			
<b>Utente</b>	<b>Data</b>	<b>Ora inizio</b>	<b>Ora fine</b>
luca.dibello	05/12/2019	00:00	00:15
luca.dibello	11/12/2019	00:00	00:15
luca.dibello	12/12/2019	00:00	00:15
luca.dibello	17/12/2019	00:00	00:15
luca.dibello	18/12/2019	00:00	00:15
luca.dibello	19/12/2019	00:00	00:15
luca.dibello	13/12/2019	00:00	00:15

Nota: i dati inseriti sono i dati che ho utilizzato per test ed il tipo di report è stato impostato con il flag YEAR, quindi legge tutte le prenotazioni fissate per l'interno anno (quelle del 2019)

Se sfortunatamente c'è un problema con la connessione al database viene mostrato un errore a schermo il quale mostra il problema che ha riscontrato il DBMS e la query che ha prodotto il problema:

**C'è stato un errore durante la generazione del PDF. Contattare un amministratore**

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FROM riservazione WHERE TIMES

Query: SELECT utente, DATE\_FORMAT(data, "%M %d %Y") as 'Data' , TIME\_FORMAT(ora\_inizio, "%H %i") as 'Ora inizio', TIME\_FORMAT(ora\_fine, "%H %i") as 'Ora fine', FF

Questo sarà molto utile in quanto, quando sarà in produzione, se la segreteria riscontrerà un problema potrà fornire un errore con delle informazioni significative.

**Problemi riscontrati e soluzioni adottate**

Non potevo creare un evento con delle osservazioni ma, quando cliccato l'evento creato, non veniva visualizzata alcuna osservazione.

Il problema era in una riga nell'handler dell'evento *eventClick*: In JavaScript (con JQuery) per impostare il valore di un campo di un form (*input*) si usa la funzione `".val(<valore>)"` mentre io utilizzavo `".text(<valore>)"`. In poche parole trattavo il campo come se fosse un normale elemento HTML come i paragrafi o gli *headers* (*h1,h2,...*).

**Punto della situazione rispetto alla pianificazione**

Sono in perfetto orario secondo la tabella di marcia. Non avendo seguito a pieno la pianificazione iniziale ho potuto svolgere più attività in parallelo.

Secondo il Gantt preventivo adesso dovrei sviluppare un sistema per la lettura dei dati dalle API, il quale ho già svolto per il caricamento degli eventi nel calendario.

**Programma di massima per la prossima giornata di lavoro**

Finire l'implementazione dei report mensili (Generazione query nel metodo `_generate_query`)

# Diario di lavoro

Luogo	Canobbio
Data	06.12.2019

## Lavori svolti

Durante la giornata di oggi ho fatto troubleshooting sul Raspberry. Dopo aver installato a casa il nuovo sistema operativo ho provato ad accenderlo ma non aveva output HDMI.

Dato che a casa non avevo testato il nuovo sistema operativo ho pensato subito che fosse un problema di mal-installazione di esso, quindi ho formattato la scheda SD ed ho installato una versione di Raspbian chiamata "*NOOBS*". Essa è fatta per gli utenti novelli, quindi non c'è bisogno di scrivere la ISO sulla scheda SD (fa tutto il raspberry).

Anche qui nessun risultato. Quindi ho deciso di cambiare monitor, cavo ed ho reinstallato la versione classica (Raspbian) e finalmente è partito.

Ho configurato il raspberry accendendo il server SSH interno, cambiando la risoluzione in 16:9 ed ho impostato una nuova password: Cp7M65\_2019

**Problemi riscontrati e soluzioni adottate**

Descritti nel capitolo precedente.

**Punto della situazione rispetto alla pianificazione**

Sono in perfetto orario secondo la tabella di marcia. Non avendo seguito a pieno la pianificazione iniziale ho potuto svolgere più attività in parallelo.

Secondo il Gantt preventivo adesso dovrei sviluppare un sistema per la lettura dei dati dalle API, il quale ho già svolto per il caricamento degli eventi nel calendario.

**Programma di massima per la prossima giornata di lavoro**

Finire l'implementazione dei report mensili (Generazione query nel metodo generate\_query) e continuare la documentazione a casa.

# Diario di lavoro

Luogo	Canobbio
Data	10.12.2019

## Lavori svolti

Durante questo week-end ho lavorato un po' al progetto iniziando a sviluppare il sistema di invio email per l'invio della password all'indirizzo mail di un utente creato. Oltre a questo lavoretto ho fatto un po' di *bug-fix* ed ottimizzazione del sistema.

Ieri a scuola, testando il progetto durante la pausa, mi sono accorto che c'erano dei problemi a livello del calendario: non riusciva a caricare i dati degli utenti di LDAP. Questo perchè, anche se l'utente era loggato con LDAP, cercavo i suoi dati all'interno del database locale. Il problema sembra banale ma mi ha portato a modificare tutta la struttura del sistema di login e del sistema dedicato al caricamento dei permessi dal database.

Ho avuto la necessità di creare un nuovo oggetto chiamato LdapUser, il quale salva al suo interno il nome, il cognome, lo username e l'email di un utente di LDAP al suo login. Quando invece un utente normale esegue un login viene creato un oggetto di tipo User (che non è altro un'estensione di LdapUser che aggiunge campi supplementari). Entrambi gli oggetti vengono creati all'index "user" dell'array di sessione: `$_SESSION["user"]`.

Le informazioni contenute nell'oggetto LdapUser vengono lette direttamente tramite LDAP in questo modo:

```
// Leggo la risposta della query LDAP e la converto in un array
$entries = ldap_get_entries($ldap, $result);
// Leggo le informazioni
$data = $entries[0];
$firstname = $data["givenname"][0];
$surname = $data["sn"][0];
$email = $data["mail"][0];

// Costruisco l'oggetto LdapUser
$user = new LdapUser(
    $this->username,
    $firstname,
    $surname,
    $email
);
```

Questo oggetto mi permette di accedere alle informazioni dell'utente senza dover eseguire query nel database locale (nel caso fosse un utente locale) o nel database di Active Directory scolastico (nel caso fosse un utente LDAP).

Dopo aver risolto questo ho testato il sistema sia con il mio utente personale LDAP (luca.dibello) sia con quello admin (luca.dibello): i permessi vengono caricati correttamente ed il sistema risulta funzionante.

Dopo aver fatto questo ho iniziato a sviluppare un applicativo in *Python* per il raspberry. Quando viene fatto partire esso carica il file config.json (il quale contiene le impostazioni del applicativo) ed apre un webserver sulla porta specificata nella configurazione.

Ho deciso di sviluppare questo sistema con Flask (modulo di Python) dato che l'ho già utilizzato e lo sviluppo di applicativi web risulta molto semplice e veloce. Esso è ben documentato e fornisce di default un *templating language* chiamanto *Jinja*.

Per ora lo script è impostato in verbose mode:

```
* Serving Flask app "rasp-cptmrs" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5005/ (Press CTRL+C to quit)
* Restarting with stat
[!] Config loaded correctly
* Debugger is active!
* Debugger PIN: 110-632-192
```

**Problemi riscontrati e soluzioni adottate**

Descritti nel capitolo precedente.

**Punto della situazione rispetto alla pianificazione**

Sono in perfetto orario secondo la tabella di marcia. Non avendo seguito a pieno la pianificazione iniziale ho potuto svolgere più attività in parallelo.

Secondo il Gantt preventivo adesso dovrei sviluppare un sistema per la lettura dei dati dalle API, il quale ho già svolto per il caricamento degli eventi nel calendario.

**Programma di massima per la prossima giornata di lavoro**

Finire l'implementazione dei report mensili (Generazione query nel metodo generate\_query) e continuare la documentazione a casa.

# Diario di lavoro

Luogo	Canobbio
Data	12.12.2019

## Lavori svolti

Durante la giornata di oggi ho continuato lo sviluppo del sistema per la visualizzazione delle prenotazioni sul monitor.

Alla fine ho deciso di non usare Python ma di usare solamente JavaScript per leggere i dati dalle API. Ho anche implementato un sistema di impostazioni per impostare il funzionamento dello schermo.

Stasera continuerò a casa l'implementazione del sistema e continuerò anche la documentazione.

Problemi riscontrati e soluzioni adottate

Descritti nel capitolo precedente.

Punto della situazione rispetto alla pianificazione

Ho quasi finito il sistema che mostra le prenotazioni sul monitor

Programma di massima per la prossima giornata di lavoro

Continuare l'implementazione del sistema Raspberry

# Diario di lavoro

Luogo	Canobbio
Data	13.12.2019

## Lavori svolti

All'inizio lezione ho provato insieme al docente il nuovo alimentatore, provando il vecchio monitor (il quale pensavamo non funzionante per via del poco voltaggio dato al Raspberry) ma nessun risultato positivo: il monitor non funziona.

Dopo questo ho fatto la demo con il docente responsabile, andando a guardare l'interno sistema alla ricerca di errori.

Grazie ai consigli del docente Valsangiacomo ho messo apposto la pagina che verrà mostrata sul sistema raspberry. Ora la pagina risulta pulita e riesce a mostrare (con il monitor del mio pc portatile) fino a 12 prenotazioni alla volta.

Questa è la nuova pagina, la quale verrà mostrata sullo schermo attaccato al muro:

### Prenotazioni 13/12/2019

Lista prenotazioni per l'aula A-4\*\*

Data	Orario	Docente
13/12/2019	16:30 - 16:45	luca.admin
13/12/2019	16:45 - 17:00	luca.admin
13/12/2019	17:00 - 17:15	luca.admin
13/12/2019	19:00 - 19:15	luca.admin
13/12/2019	19:15 - 19:30	luca.admin
13/12/2019	19:30 - 19:45	luca.admin
13/12/2019	19:45 - 20:00	luca.admin
13/12/2019	20:00 - 20:15	luca.admin
13/12/2019	20:15 - 20:30	luca.admin
13/12/2019	20:30 - 20:45	luca.admin
13/12/2019	20:45 - 21:00	luca.admin
13/12/2019	21:00 - 21:15	luca.admin

© 2019 Copyright: CPT Meeting Room Scheduler  
Sviluppatore: Luca Di Bello I4AC

Ultimo aggiornamento: 13/12/2019 16:19

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato alcun problema durante la lezione odierna.

**Punto della situazione rispetto alla pianificazione**

Ho finito l'attività relativa allo sviluppo del sistema con raspberry. Mi manca il sistema per l'invio delle mail.

**Programma di massima per la prossima giornata di lavoro**

Continuare l'implementazione del sistema di invio delle mail e rendere responsivo il calendario (anche rimpicciolendolo un attimo dato che adesso è troppo grande).

Devo anche trovare un modo per dividere le prenotazioni in base al mese all'interno dei report.

# Diario di lavoro

Luogo	Canobbio
Data	17.12.2019

## Lavori svolti

Durante la lezione di oggi mi sono dedicato alla documentazione.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato alcun problema durante la lezione odierna.

Punto della situazione rispetto alla pianificazione

Ho finito il progetto. Mi manca solamente mettere l'indirizzo ip statico al raspberry.

Programma di massima per la prossima giornata di lavoro

Stampare la documentazione e mettere l'ip statico al raspberry

# Diario di lavoro

Luogo	Canobbio
Data	20.12.2019

## Lavori svolti

Durante la lezione di oggi mi sono dedicato alla stampa ed alla rilegatura della documentazione.

Ho messo l'ip statico al raspberry ed ho caricato al suo interno il codice che andrà a gestire lo schermo.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato alcun problema durante la lezione odierna.

**Punto della situazione rispetto alla pianificazione**

Progetto finito e consegnato.

**Programma di massima per la prossima giornata di lavoro**

-

# Diario di lavoro

Luogo	Canobbio
Data	19.12.2019

## Lavori svolti

Durante la lezione di oggi mi sono dedicato alla stampa di tutti i diari. Successivamente ho creato e commentato il diagramma di Gantt consuntivo.

Oggi dovevo impostare l'ip statico sul raspberry ma non essendoci in sede il docente Valsangiacomo non ho potuto farlo.

**Problemi riscontrati e soluzioni adottate**

Non ho riscontrato alcun problema durante la lezione odierna.

**Punto della situazione rispetto alla pianificazione**

Ho finito il progetto. Mi manca solamente mette l'indirizzo ip statico al raspberry e stampare la documentazione.

**Programma di massima per la prossima giornata di lavoro**

Stampare la documentazione e mettere l'ip statico al raspberry.