

Candidato: Filippo Finke
Azienda: Scuola d'Arti e Mestieri Trevano
Periodo: 11.05.2020 – 29.05.2020
Presentazione: 13:30 - 14:30 10.06.2020

Situazione iniziale

Lo scopo del progetto “Gestione Web Ritardi Allievi SAMT” è quello di semplificare, velocizzare ed automatizzare il processo di assegnazione dei ritardi all’interno della Scuola d’Arti e Mestieri di Trevano. Al momento i ritardi vengono segnati principalmente su carta o con altre metodologie che però non garantiscono l’integrità dei dati stessi e non permettono di eseguire ricerche velocemente, controlli e altro. Il progetto in questione si occupa dunque di digitalizzare ciò che viene fatto manualmente e di creare un sito web che permetta di gestire il tutto in modo automatizzato. Questo permetterà di velocizzare il lavoro che dovrà essere svolto dai docenti. Grazie a questo sistema si potrà inoltre tenere uno storico dei ritardi accumulati e recuperati di tutti gli studenti che sono stati registrati. Sarà inoltre possibile creare un file PDF con lo storico di uno studente. Oltre a questo, sarà possibile fare accedere più docenti di classe allo stesso sistema per avere una visione generale dell’andamento riguardante i ritardi dei vari studenti. Nella pagina di gestione dei recuperi di questi ritardi sarà inoltre presente una opzione per ricavare una lista in formato PDF di tutti gli alunni che dovranno recuperare i ritardi in sede, questa lista sarà molto utile a chi dovrà fare recuperare i ritardi in presenza.

Attuazione

L’applicativo web è stato sviluppato utilizzando principalmente il linguaggio di programmazione PHP per quanto riguarda l’elaborazione delle richieste web e la gestione della banca dati. In appoggio a PHP è stato utilizzato HTML per la definizione della struttura delle diverse pagine presenti nel sito web. Per assegnare uno stile, ovvero personalizzare la grafica, di queste pagine è stato utilizzato CSS. L’applicativo fa uso di un framework chiamato php-rest il quale permette di gestire percorsi, gestione permessi, connessione al database e altre cose molto utili in modo semplificato. L’utilizzo di questo framework assieme al template SB Admin 2 hanno aiutato a velocizzare lo sviluppo dell’applicativo. Il progetto inoltre si basa sui pattern MVC e REST in modo da avere una struttura del codice molto più semplice da mantenere. All’interno del software è inoltre presente la generazione dinamica di file PDF. Per la creazione di questi file è stata utilizzata la libreria FPDF per il linguaggio PHP. Per gestire queste librerie è stato utilizzato il gestore di pacchetti Composer in modo da permettere una installazione semplificata.

Risultati

Il progetto è stato sviluppato completamente rispettando i requisiti del committente. Il tutto è stato caricato su un hosting esterno. Il sito web è stato sviluppato in modo che possa essere configurato in modo semplice attraverso un file di configurazione. Inoltre, è possibile modificare delle ulteriori impostazioni riguardanti l’applicativo web attraverso un pannello di impostazioni le quali verranno applicate una volta salvate. L’applicativo sarà quindi gestibile completamente attraverso delle interfacce web (gestione utenti, impostazioni, etc.) una volta installato.



Gestione Web Ritardi Allievi SAMT

Titolo del progetto: Gestione Web Ritardi Allievi SAMT
Alunno/a: Filippo Finke
Classe: I4AC
Anno scolastico: 2019/2020
Docente responsabile: Fabrizio Valsangiacomo

Indice

1 Introduzione	5
1.1 Informazioni sul progetto	5
1.2 Abstract	5
1.3 Scopo	5
2 Analisi	6
2.1 Analisi del dominio	6
2.2 Analisi e specifica dei requisiti	6
2.3 Use case	9
2.4 Pianificazione	10
2.4.1 Analisi.....	11
2.4.2 Progettazione	11
2.4.3 Implementazione.....	11
2.4.4 Testing	12
2.4.5 Consegna.....	12
2.5 Analisi dei mezzi.....	12
2.5.1 Software	12
2.5.2 Hardware.....	13
3 Progettazione	13
3.1 Design dell'architettura del sistema	13
3.2 Design dei dati e database.....	13
3.2.1 Schema ER	13
3.2.2 Descrizioni delle tabelle	14
3.2.3 Schema logico	17
3.3 Diagrammi delle classi	17
3.3.1 UML Controllers	17
3.3.2 UML Libs	17
3.3.3 UML Middlewares	18
3.3.4 UML Models	18
3.4 Design delle interfacce	19
3.4.1 Pagina di accesso	19
3.4.2 Pagina cambio password.....	19
3.4.3 Pagina di recupero password	19
3.4.4 Pagina di aggiunta ritardi	20
3.4.5 Pagina di aggiunta recuperi	20
3.4.6 Pagina di gestione utenti.....	20
3.4.7 Pagina impostazioni	21
4 Implementazione	21
4.1 Gestione versioni.....	21
4.2 Gestore di pacchetti	21
4.3 Database	22
4.3.1 Account di accesso	22
4.3.2 Implementazione banca dati	22
4.3.3 Interrogazione database	22
4.3.4 Schema generato.....	23
4.4 Applicativo web	24
4.4.1 Struttura	24
4.4.2 Model View Controller (MVC)	24
4.4.3 Representational State Transfer (REST)	25
4.4.4 Routing delle richieste.....	25
4.4.5 Configurazione	26
4.4.6 Autenticazione	26
4.4.6.1 Gestione delle sessioni	27
4.4.6.2 Gestione dei permessi	28
4.4.6.3 Gestione percorsi e accessi	28
4.4.7 Invio di e-mail	29
4.4.8 Validazione dei dati	30

4.4.9	Gestione dati ed interrogazione database	30
4.4.9.1	Tabella user	30
4.4.9.2	Tabella token	33
4.4.9.3	Tabella student	35
4.4.9.4	Tabella section.....	37
4.4.9.5	Tabella delay	38
4.4.9.6	Tabella year	41
4.4.9.7	Tabella setting	43
4.4.10	Creazione PDF	45
4.4.10.1	PDF Studente	46
4.4.10.2	PDF Recuperi	47
4.4.11	Sicurezza	48
4.4.11.1	Interrogazione database	48
4.4.11.2	Salvataggio dei dati	49
4.4.11.3	Salvataggio credenziali.....	49
4.4.11.4	Recupero password.....	49
4.4.12	Interfacce grafiche	50
4.4.12.1	Pagina di accesso.....	50
4.4.12.2	Pagina di cambio password.....	50
4.4.12.3	Pagina di recupero password	51
4.4.12.4	Pagina di gestione ritardi	51
4.4.12.5	Pagina di gestione recuperi	51
4.4.12.6	Pagina di gestione utenti	52
4.4.12.7	Pagina impostazioni.....	52
5	Test	53
5.1	Protocollo di test.....	53
5.2	Risultati test.....	60
5.3	Mancanze/limitazioni conosciute.....	67
6	Consuntivo.....	67
7	Conclusioni.....	69
7.1	Sviluppi futuri.....	69
7.2	Considerazioni personali.....	69
8	Glossario	69
9	Sitografia	70
10	Allegati.....	70

Indice delle figure

Figura 1 Schema caso d'uso.	9
Figura 2 Diagramma di Gantt preventivo.	10
Figura 3 Diagramma di Gantt, Analisi.	11
Figura 4 Diagramma di Gantt, Progettazione.	11
Figura 5 Diagramma di Gantt, Implementazione.	11
Figura 6 Diagramma di Gantt, Testing.	12
Figura 7 Diagramma di Gantt, Consegna.	12
Figura 8 Architettura del sistema semplificata.	13
Figura 9 Schema ER banca dati.	14
Figura 10 UML Classi controller.	17
Figura 11 UML Classi libs.	17
Figura 12 UML Classi middlewares.	18
Figura 13 UML Classi models.	18
Figura 14 Mockup pagina di accesso.	19
Figura 15 Mockup pagina di cambio password.	19
Figura 16 Mockup pagina di recupero password.	19
Figura 17 Mockup pagina di aggiunta ritardi.	20
Figura 18 Mockup pagina di aggiunta recuperi.	20
Figura 19 Mockup pagina di gestione utenti.	20
Figura 20 Mockup pagina impostazioni.	21
Figura 21 Schema database generato.	23
Figura 22 Schema pattern MVC.	24
Figura 23 Schema pattern REST.	25
Figura 24 Esempio PDF studente.	47
Figura 25 Esempio PDF recuperi.	48
Figura 26 Pagina di accesso.	50
Figura 27 Pagina di cambio password.	50
Figura 28 Pagina di recupero password.	51
Figura 29 Pagina di gestione ritardi.	51
Figura 30 Pagina di gestione recuperi.	51
Figura 31 Pagina di gestione utenti.	52
Figura 32 Pagina impostazioni.	52
Figura 33 Pagina di accesso.	60
Figura 34 Pagina principale.	60
Figura 35 Pagina di accesso, errore credenziali.	61
Figura 36 E-mail recupero password.	61
Figura 37 Cambio password.	61
Figura 38 Pagina di gestione utenti.	62
Figura 39 Creazione utente.	62
Figura 40 Pagina impostazioni.	62
Figura 41 Sezione creata correttamente.	63
Figura 42 Anno scolastico creato correttamente.	63
Figura 43 Impostazione aggiornata.	63
Figura 44 Pagina di gestione ritardi.	63
Figura 45 Creazione studente.	64
Figura 46 Aggiunta ritardo.	64
Figura 47 Modale di visualizzazione ritardi.	64
Figura 48 Generazione PDF studente.	65
Figura 49 Pagina di gestione recuperi.	65
Figura 50 Ritardo recuperato.	66
Figura 51 PDF Recupero ritardi.	66
Figura 52 Diagramma di Gantt consuntivo.	68

1 Introduzione

1.1 Informazioni sul progetto

Titolo: Gestione Web Ritardi Allievi SAMT

Allievo coinvolto nel progetto: Filippo Finke, filippo.finke@samtrevano.ch

Classe: I4AC Scuola Arti e Mestieri Trevano, Informatica

Formatore: Fabrizio Valsangiacomo, fabrizio.valsangiacomo@edu.ti.ch

Perito: Gianluca Costante, gianluca.costante@gmail.com

Data inizio: 11.05.2020

Data fine: 29.05.2020

Durata: 80 ore

1.2 Abstract

The aim of the project "Gestione Web Ritardi Allievi SAMT" is to simplify, speed up and automate the process of delay assignment within the SAMT. At the moment the delays are marked mainly on paper or with other methods that do not guarantee the integrity of the data themselves and do not allow to perform quick searches, controls on the data and other stuff. This project deals with digitizing what is done manually and creating a website that allows you to manage everything in an automated manner. This will speed up the work to be done by the teachers. Thanks to this system it will also be possible to keep a record of the accumulated and recovered delays of all the students who have been registered. You will also be able to create a PDF file with the history of a student. In addition to this, it will be possible to access more than one class teacher to the same system in order to have an overview of the progress regarding the delays of the various students. In the management page of these delays recoveries there will also be an option to get a PDF list of all the students who will have to recover the delays in attendance, this list will be very useful for those who will have to recover the delays in attendance.

1.3 Scopo

Lo scopo del progetto “Gestione Web Ritardi Allievi SAMT” è quello di semplificare, velocizzare ed automatizzare il processo di assegnazione dei ritardi all’interno della Scuola d’Arti e Mestieri di Trevano. Al momento i ritardi vengono segnati principalmente su carta o con altre metodologie che però non garantiscono l’integrità dei dati stessi e non permettono di eseguire ricerche velocemente, controlli e altro. Il progetto in questione si occupa dunque di digitalizzare ciò che viene fatto manualmente e di creare un sito web che permetta di gestire il tutto in modo automatizzato. Questo permetterà di velocizzare il lavoro che dovrà essere svolto dai docenti. Grazie a questo sistema si potrà inoltre tenere uno storico dei ritardi accumulati e recuperati di tutti gli studenti che sono stati registrati. Sarà inoltre possibile creare un file PDF con lo storico di uno studente. Oltre a questo, sarà possibile fare accedere più docenti di classe allo stesso sistema per avere una visione generale dell’andamento riguardante i ritardi dei vari studenti. Nella pagina di gestione dei recuperi di questi ritardi sarà inoltre presente una opzione per ricavare una lista in formato PDF di tutti gli alunni che dovranno recuperare i ritardi in sede, questa lista sarà molto utile a chi dovrà fare recuperare i ritardi in presenza.

2 Analisi

2.1 Analisi del dominio

È stato richiesto lo sviluppo di un gestionale web per la gestione dei ritardi della Scuola d'Arti e Mestieri di Trevano. Il prodotto dovrà essere un applicativo web accessibile attraverso la rete utilizzando browser moderni (Esempio: Chrome). Gli utenti che accederanno a questo applicativo saranno principalmente docenti di classe che andranno ad inserire e gestire i ritardi degli studenti. Saranno disponibili ulteriori permessi che permetteranno di distinguere gli utenti in utenti amministratori e utenti normali. Gli utenti amministratori potranno gestire gli utenti presenti all'interno dell'applicativo. Gli utenti normali avranno dei permessi limitati (inserimento dei ritardi, visione dei dati o creazione PDF) che permetteranno a questo utente di eseguire solamente determinate azioni in modo limitato. Sarà presente una pagina di amministrazione che permetterà agli amministratori di modificare alcune impostazioni dell'applicativo come ad esempio la soglia limite dei ritardi prima del recupero e altre opzioni. I docenti di classe potranno inserire studenti ed i relativi ritardi all'interno del sistema, sarà inoltre possibile inserire quando uno studente ha recuperato un determinato ritardo in modo da poterne tenere uno storico. Vi sarà inoltre la possibilità di inserire dei ritardi giustificati, ovvero dei ritardi che vengono mostrati nel conteggio totale ma non dovranno essere recuperati dallo studente. Una volta raggiunta la soglia massima di ritardi un sistema automatico invierà una e-mail di notifica allo studente informandolo che verrà contattato per eseguire il recupero del ritardo. Ogni semestre verrà eseguito un reset automatico del numero di ritardi degli studenti ma non dei ritardi da recuperare. L'applicativo dovrà inoltre essere caricato su un hosting.

2.2 Analisi e specifica dei requisiti

È richiesto da parte del committente lo sviluppo di un applicativo web che sia accessibile tramite la rete e che sia web. Il prodotto dovrà possedere un sistema di autenticazione con un sistema di permessi integrato. Il sistema dovrà dunque distinguere i vari utenti in base ai loro permessi. È dunque richiesta una pagina di login attraverso la quale gli utenti andranno ad accedere all'applicativo. Gli amministratori del software potranno aggiungere ed eliminare gli utenti, alla creazione di un utente le credenziali per l'accesso verranno inviate in modo automatico attraverso un messaggio di posta elettronica e al primo accesso verrà richiesto all'utente di cambiare la propria password. In caso di perdita della propria password essa potrà essere recuperata attraverso il proprio indirizzo e-mail. Gli amministratori potranno aggiungere utenti amministratori oppure utenti limitati i quali potranno avere i seguenti permessi:

- Inserimento dei ritardi
- Visione dei dati
- Creazione dei PDF

Dovrà essere presente una pagina di amministrazione attraverso la quale sarà possibile configurare le date di inizio e di fine di ogni semestre, i nomi delle sezioni scolastiche, gli anni scolastici e la soglia di ritardi massimi dopo i quali essi andranno recuperati. I docenti avranno a disposizione una pagina attraverso la quale potranno registrare gli allievi che hanno cominciato ad accumulare dei ritardi e di conseguenza sarà possibile anche aggiungere ritardi ad allievi già registrati. Sarà disponibile un'altra pagina che permetterà ai docenti di inserire i ritardi recuperati da parte degli studenti. Di ogni studente verrà inoltre tenuto lo storico dei ritardi. Il prodotto dovrà essere caricato su un hosting.

ID: REQ-000	
Nome	Piattaforma dell'applicativo
Priorità	1
Versione	1.0
Note	Si necessita di un applicativo web.
Sotto requisiti	
001	Il sito deve funzionare su browser moderni (Esempio: Chrome).

ID: REQ-001	
Nome	Pagina di accesso
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di inserire le proprie credenziali per accedere all'applicativo web.
Sotto requisiti	
001	Si necessita di una maschera di login.
002	Si necessita di un modale di cambio password per nuovi utenti.

ID: REQ-002	
Nome	Pagina di recupero password
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di recuperare la propria password.
Sotto requisiti	
001	Dovrà essere inviata una e-mail di recupero password se l'utente che la richiede è esistente.

ID: REQ-003	
Nome	Pagina di gestione utenti
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di gestire gli utenti che avranno accesso all'applicativo web.
Sotto requisiti	
001	Dovrà essere possibile creare utenti.
002	Le credenziali verranno inviate per posta elettronica.
003	Dovrà essere possibile eliminare utenti.
004	Dovrà essere possibile modificare i permessi di ogni singolo utente.
005	Dovrà essere sempre presente almeno un utente amministratore.
006	Non dovrà essere possibile eliminare il proprio utente.

ID: REQ-004	
Nome	Pagina di amministrazione
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di gestire le impostazioni dell'applicativo web.
Sotto requisiti	
001	Dovrà essere possibile inserire le date di inizio e fine di ogni semestre.
002	Dovrà essere possibile inserire i nomi delle sezioni.
003	Dovrà essere possibile inserire gli anni scolastici.
004	Dovrà essere possibile modificare la soglia di ritardi massimi. (Di base tre compreso)

ID: REQ-005	
Nome	Pagina di aggiunta ritardi
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di aggiungere ritardi agli studenti.
Sotto requisiti	
001	Dovrà essere possibile inserire nuovi studenti.
002	Dovrà essere possibile aggiungere un ritardo ad uno studente.
003	Al raggiungimento della soglia massima dovrà essere inviata una e-mail di notifica.
004	Dovrà essere tenuto un istoriato dei ritardi per anno.
005	Dovrà essere possibile creare un PDF con le informazioni dello studente.

ID: REQ-006	
Nome	Pagina di gestione recuperi
Priorità	1
Versione	1.0
Note	Si necessita di una pagina che permetta di aggiungere i ritardi recuperati da parte degli studenti.
Sotto requisiti	
001	Dovrà essere possibile inserire il ritardo recuperato da parte dello studente.
002	Dovrà essere possibile creare un PDF contenente tutti gli studenti che devono recuperare dei ritardi.

ID: REQ-007	
Nome	Messa in produzione
Priorità	1
Versione	1.0
Note	L'applicativo dovrà essere messo in produzione su un hosting.

2.3 Use case

Gli agenti di questo schema si dividono in docente ed amministratore. Entrambi gli agenti hanno la possibilità di accedere alla pagina di accesso attraverso la quale sarà possibile accedere all'applicativo. Attraverso la pagina di login sarà possibile recarsi in un'altra pagina dedicata al recupero password. Entrambi gli agenti potranno eseguire azioni di gestione di recuperi e ritardi ovvero potranno creare degli studenti ed assegnarci dei ritardi. Ulteriormente potranno anche gestire i recuperi dei ritardi eseguiti dagli studenti. Successivamente solo gli amministratori potranno accedere alle pagine di gestione degli utenti dove sarà possibile aggiungere, modificare ed eliminare gli utenti presenti nell'applicativo. L'amministratore potrà accedere ad una ulteriore pagina di amministrazione che permetterà di modificare le impostazioni dell'applicativo come: i semestri, gli anni scolastici, le sezioni e la soglia massima dei ritardi. In questo caso l'utente docente ha tutti i permessi per accedere alle pagine di gestione dei ritardi, prima di tutte queste pagine viene eseguito un controllo dei permessi dell'utente per decidere se esso può accedere oppure no ad una determinata pagina. I permessi controllati saranno:

- Inserimento dei ritardi
- Visione dei dati
- Creazione dei PDF

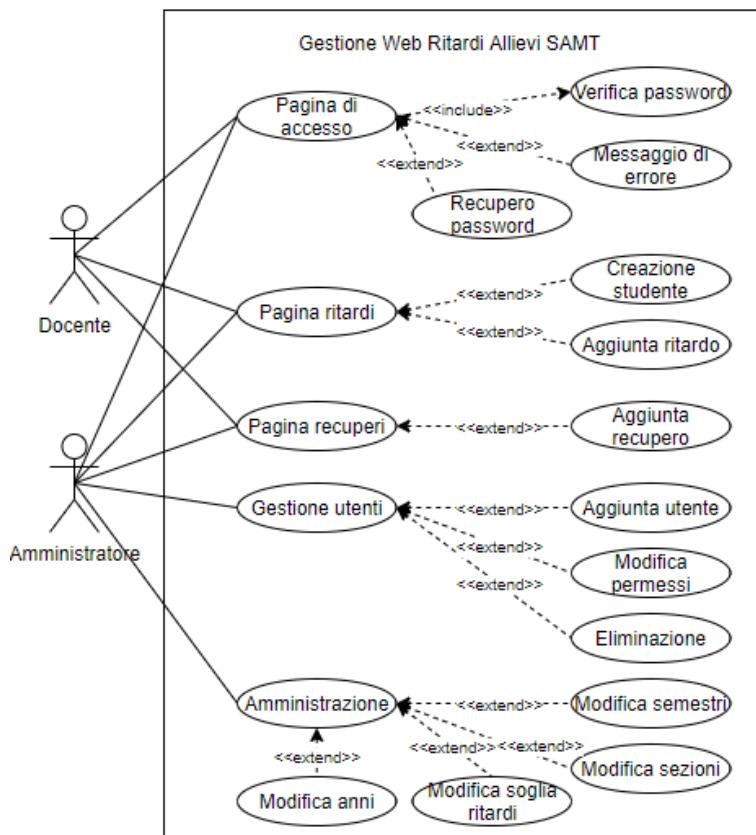


Figura 1 Schema caso d'uso.

2.4 Pianificazione

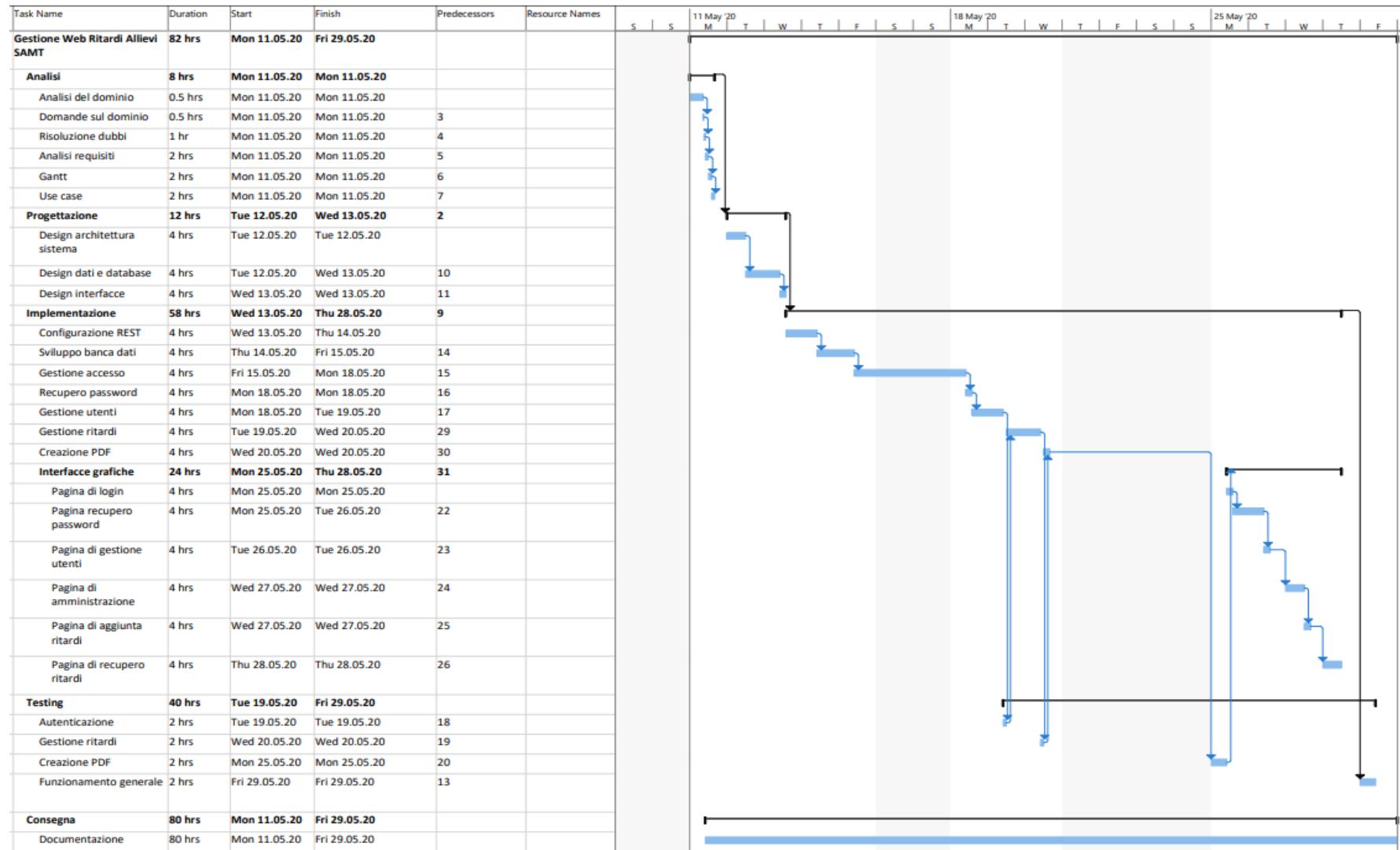


Figura 2 Diagramma di Gantt preventivo.

2.4.1 Analisi

Ho suddiviso la fase di analisi in sei attività. Durante questa fase mi sono occupato di capire di cosa tratta il progetto e quali sono le richieste da parte del committente.



Figura 3 Diagramma di Gantt, Analisi.

2.4.2 Progettazione

La progettazione è la seconda fase del progetto “Gestione Web Ritardi Allievi SAMT” ed è composta da tre attività. Questa fase contiene le attività di design della struttura del progetto. Considero questa fase molto importante in quanto le fasi successive saranno basate su di essa.



Figura 4 Diagramma di Gantt, Progettazione.

2.4.3 Implementazione

La fase di implementazione è la più lunga all'interno del progetto. Questa fase consiste nella vera scrittura del codice che andrà a comporre il prodotto finale. Questa fase è strettamente collegata all'analisi e alla progettazione in quanto si basa su di esse per lo sviluppo.

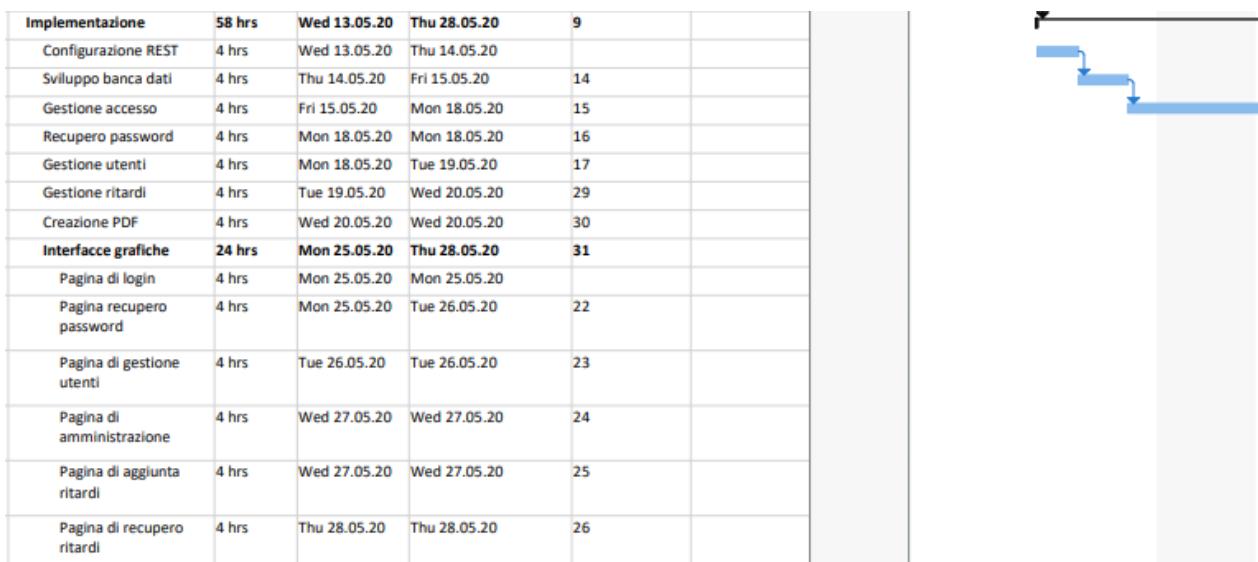


Figura 5 Diagramma di Gantt, Implementazione.

2.4.4 Testing

Un'altra fase molto importante è il testing. All'interno di questa fase sono presenti i test che verranno eseguiti alle varie sezioni del progetto. Le attività di testing sono state generalizzate in modo tale da rendere il Gantt leggibile, durante il corso del progetto verranno eseguiti test più approfonditi.

Testing	40 hrs	Tue 19.05.20	Fri 29.05.20	
Autenticazione	2 hrs	Tue 19.05.20	Tue 19.05.20	18
Gestione ritardi	2 hrs	Wed 20.05.20	Wed 20.05.20	19
Creazione PDF	2 hrs	Mon 25.05.20	Mon 25.05.20	20
Funzionamento generale	2 hrs	Fri 29.05.20	Fri 29.05.20	13

Figura 6 Diagramma di Gantt, Testing.

2.4.5 Consegna

L'ultima fase del diagramma è la consegna del prodotto, all'interno di essa vi è una sola attività che riguarda la documentazione. Questa attività ha la durata dell'interno progetto perché verrà aggiornata nel corso dello sviluppo di esso.

Consegna	80 hrs	Mon 11.05.20	Fri 29.05.20
Documentazione	80 hrs	Mon 11.05.20	Fri 29.05.20

Figura 7 Diagramma di Gantt, Consegna.

2.5 Analisi dei mezzi

2.5.1 Software

I software utilizzati per la realizzazione del progetto sono:

- Google Chrome 76.0
- Microsoft Word 265 18.2004
- Microsoft Project 2019
- Microsoft VS Code 1.37.1
- PhpStorm 2020.1.1
- MySQL 8.0.13
- PHP 7.3.5
- Draw.io (<https://draw.io>)
- PaperCut 5.7.0
- Postman 7.24.0

Librerie utilizzate:

- jQuery 3.4.1 (<https://jquery.com/>)
- Bootstrap 4.3.1 (<https://getbootstrap.com/>)
- php-rest (<https://github.com/filippofinke/php-rest>)
- FPDF (<http://www.fpdf.org/>)

Template utilizzati:

- SB Admin 2 (<https://github.com/BlackrockDigital/startbootstrap-sb-admin-2>)

Gestore librerie utilizzato:

- Composer 1.7.3 (<https://getcomposer.org/download/>)

Nota: Il progetto richiede un server mail per funzionare, in questo caso viene utilizzato PaperCut durante lo sviluppo per simularne uno.

2.5.2 Hardware

Il progetto è stato sviluppato su un computer fisso.

Le specifiche hardware sono:

- 32 GB di RAM
- Intel Core I7-7700K 4 core

Il progetto potrà essere messo in produzione su una qualsiasi macchina con più di:

- 512MB di RAM
- 2GB di disco

3 Progettazione

3.1 Design dell'architettura del sistema

Questo è lo schema dell'architettura del sito web molto semplificato, l'applicativo web interagisce con il database in due modi. Il primo modo è diretto tramite l'utilizzo dei metodi messi a disposizione da PHP per l'interrogazione della banca dati MySQL. Mentre il secondo modo per interrogare la banca dati è attraverso delle richieste HTTP eseguite in modo asincrono da JavaScript. Per implementare questo sistema vengono dunque utilizzate due stili architetturali: REST e MVC. Vengono ricavati i dati in modo diretto in azioni che non vanno ad influenzare una buona esperienza utente, mentre dove vengono eseguite delle azioni da parte dell'utente (ad esempio l'aggiunta di un utente) vengono eseguite delle chiamate alle API REST in modo da offrire una migliore esperienza all'utente.

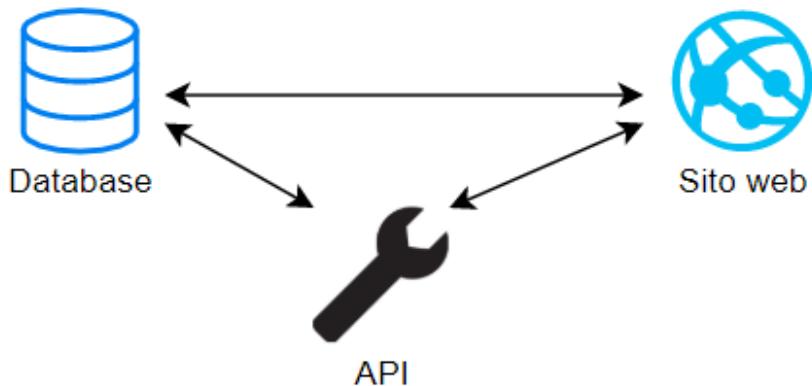


Figura 8 Architettura del sistema semplificata.

3.2 Design dei dati e database

Tutti i dati riguardanti l'applicativo verranno salvati all'interno di un database MySQL.

3.2.1 Schema ER

Questo è lo schema ER utilizzato dall'applicativo web. È composto da sette tavole. La tabella "USER" viene utilizzata per salvare tutte le informazioni relative agli utenti che potranno utilizzare l'applicativo web. La tabella "TOKEN" viene utilizzata per immagazzinare i codici di recupero password che vengono generati quando un utente dimentica la propria password. La tabella "STUDENT" contiene tutti gli studenti che hanno fatto almeno un ritardo e che dunque sono stati inseriti dai propri docenti di classe. Successivamente in correlazione con gli studenti sono presenti tre tavole, la prima "SECTION" rappresenta la sezione e classe di uno studente, mentre la seconda tabella "DELAY" contiene tutti i ritardi che sono stati fatti dai vari alunni con varie informazioni, l'ultima tabella in correlazione è "YEAR" la quale contiene i vari semestri per i vari anni in modo da poter identificare ogni utente per anno. In fine rimane la tabella "SETTING" che contiene le impostazioni del sito web in formato: nome impostazione e valore. All'interno del database potranno esserci degli studenti duplicati ma con anni diversi in modo da potere tenere traccia di uno storico.

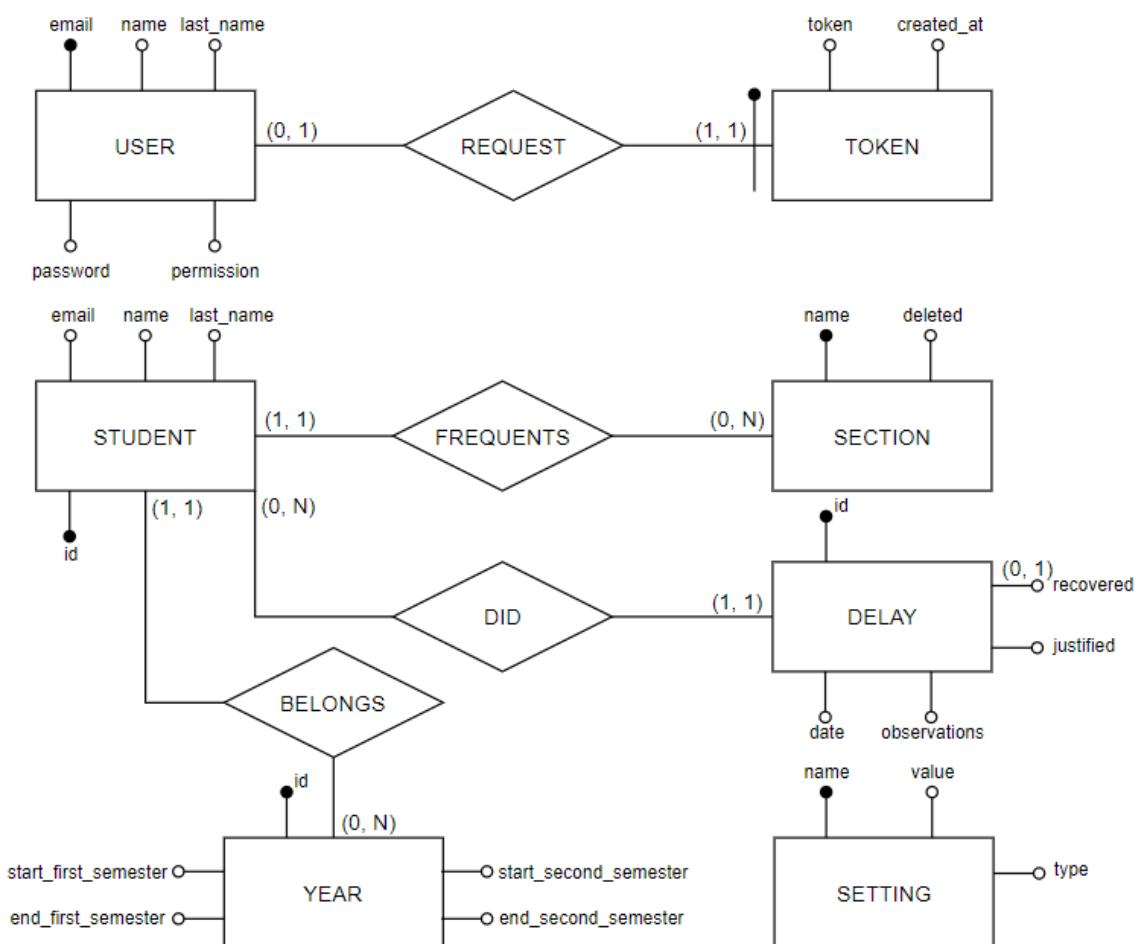


Figura 9 Schema ER banca dati.

3.2.2 Descrizioni delle tabelle

USER	
Attributo	Descrizione
email	Rappresenta un indirizzo e-mail dell'utente. È un attributo di tipo stringa con limite di 255 caratteri. Non può essere nullo e deve essere univoco. Esempio: filippo.finke@samtrevano.ch
name	Rappresenta il nome di un utente. È un attributo di tipo stringa con un limite di 20 caratteri. Può contenere solamente lettere dell'alfabeto e non può essere nullo. Esempio: Filippo
last_name	Rappresenta il cognome di un utente. È un attributo di tipo stringa con un limite di 20 caratteri. Può contenere solamente lettere dell'alfabeto e non può essere nullo. Esempio: Finke
password	Rappresenta la password di un utente. È un attributo di tipo stringa con limite di 60 caratteri. Non può essere nullo. All'interno di questo attributo verrà salvata un hash della password dell'utente. Esempio: \$2y\$10\$NmiaiLmr3dhUg3ePIExyt.I2KvE7SK6le1UH67QVikBlyBjTHgVG

permission	Rappresenta il permesso di un utente. È un attributo di tipo intero e non può essere nullo. Può contenere i seguenti valori: 1 - Inserimento ritardi. 2 - Visione ritardi. 4 - Creazione PDF. 8 - Amministratore. Esempio: 8
------------	---

TOKEN	
Attributo	Descrizione
token	Rappresenta un codice che verrà utilizzato per eseguire il recupero della password. Questo codice verrà generato in modo casuale. All'interno dell'attributo verrà salvata un hash in SHA256 del token di recupero password. Il campo è di tipo stringa con un limite di 64 caratteri e non può essere nullo. Esempio: 4b9eb466ad2615314c8194b1c46e6ef8e910d0b41a682e32de73503883e09b58
created_at	Rappresenta la data di creazione del codice di recupero password. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-05-12 14:16:59

STUDENT	
Attributo	Descrizione
id	Rappresenta l'identificatore di uno studente. È un attributo di tipo intero e viene impostato in modo automatico dall'applicativo. Non può essere nullo e deve essere univoco. Esempio: 1
email	Rappresenta un indirizzo e-mail dell'utente. È un attributo di tipo <u>stringa</u> con limite di 255 caratteri. Non può essere nullo e deve essere univoco. Esempio: filippo.finke@samtrevano.ch
name	Rappresenta il nome di un utente. È un attributo di tipo stringa con un limite di 20 caratteri. Può contenere solamente lettere dell'alfabeto e non può essere nullo. Esempio: Filippo
last_name	Rappresenta il cognome di un utente. È un attributo di tipo stringa con un limite di 20 caratteri. Può contenere solamente lettere dell'alfabeto e non può essere nullo. Esempio: Finke

SECTION	
Attributo	Descrizione
name	Rappresenta il nome della sezione. È un attributo di tipo stringa, ha un limite di 10 caratteri e non può essere nullo. Esempio: SAM I4AC
deleted	Determina se una sezione è stata eliminata oppure no. Esempio: 0

DELAY	
Attributo	Descrizione
id	Rappresenta l'identificatore di un ritardo. È un attributo di tipo intero e viene impostato in modo automatico dall'applicativo. Non può essere nullo e deve essere univoco. Esempio: 1
recovered	Rappresenta la data di recupero del ritardo. È un attributo di tipo DATE e può essere nullo. Esempio: 2020-05-12
justified	Flag che indica se il ritardo è giustificato oppure no. Esempio: 0
date	Rappresenta la data del ritardo. È un attributo di tipo DATE e non può essere nullo. Esempio: 2020-05-12
observations	Rappresenta le osservazioni correlate con il ritardo. È un attributo di tipo stringa con un limite di 255 valori, può contenere qualsiasi carattere e può essere nullo. Esempio: Ritardo causa treni.

SETTING	
Attributo	Descrizione
name	Rappresenta il nome dell'impostazione. È un attributo di tipo stringa, ha un limite di 50 caratteri e non può essere nullo. Esempio: website_title
value	Rappresenta il valore dell'impostazione. È un attributo di tipo stringa, ha un limite di 255 caratteri e non può essere nullo. Esempio: Gestione ritardi
type	Rappresenta il tipo di dato dell'impostazione. Esempio: email

YEAR	
Attributo	Descrizione
id	Rappresenta l'identificatore di un anno. È un attributo di tipo intero e viene impostato in modo automatico dall'applicativo. Non può essere nullo e deve essere univoco. Esempio: 1
start_first_semester	Rappresenta la data di inizio del primo semestre. È un attributo di tipo DATE. Esempio: 2019-05-10
end_first_semester	Rappresenta la data di fine del primo semestre. È un attributo di tipo DATE. Esempio: 2020-05-11
start_second_semester	Rappresenta la data di inizio del secondo semestre. È un attributo di tipo DATE. Esempio: 2020-05-11
end_second_semester	Rappresenta la data di fine del secondo semestre. È un attributo di tipo DATE. Esempio: 2020-09-12

3.2.3 Schema logico

Questo è lo schema logico del database:

user(email, name, last_name, password, permission)

token(email(FK), token, created_at)

setting(name, value, type)

section(name, deleted)

student(id, email, name, last_name, section(FK), year(FK))

delay(id, email(FK), date, observations*, recovered*, justified)

year(id, start_first_semester, end_first_semester, start_second_semester, end_second_semester)

3.3 Diagrammi delle classi

3.3.1 UML Controllers

Questo è il diagramma UML delle classi controller presenti nell'applicativo.

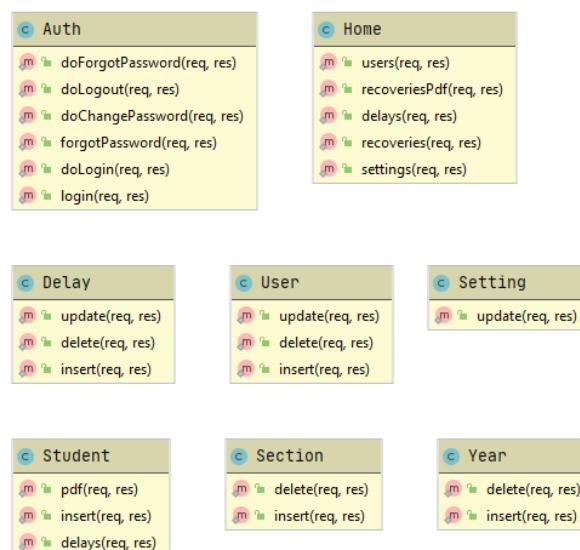


Figura 10 UML Classi controller.

3.3.2 UML Libs

Questo è il diagramma UML delle classi di aiuto / librerie presenti all'interno dell'applicativo.

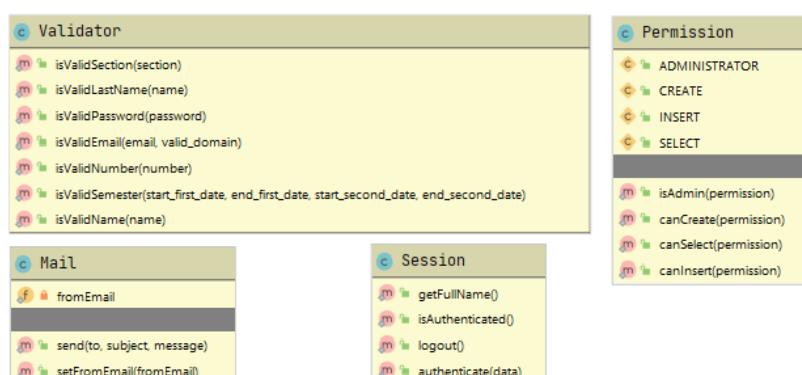


Figura 11 UML Classi libs.

3.3.3 UML Middlewares

Questo è il diagramma UML dei middlewares presenti all'interno dell'applicativo.

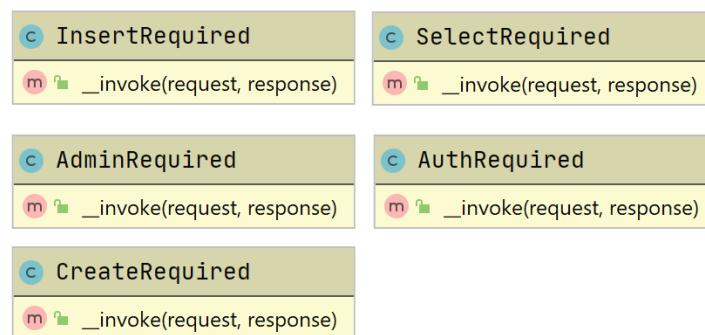


Figura 12 UML Classi middlewares.

3.3.4 UML Models

Questo è il diagramma UML delle classi models presenti all'interno dell'applicativo.

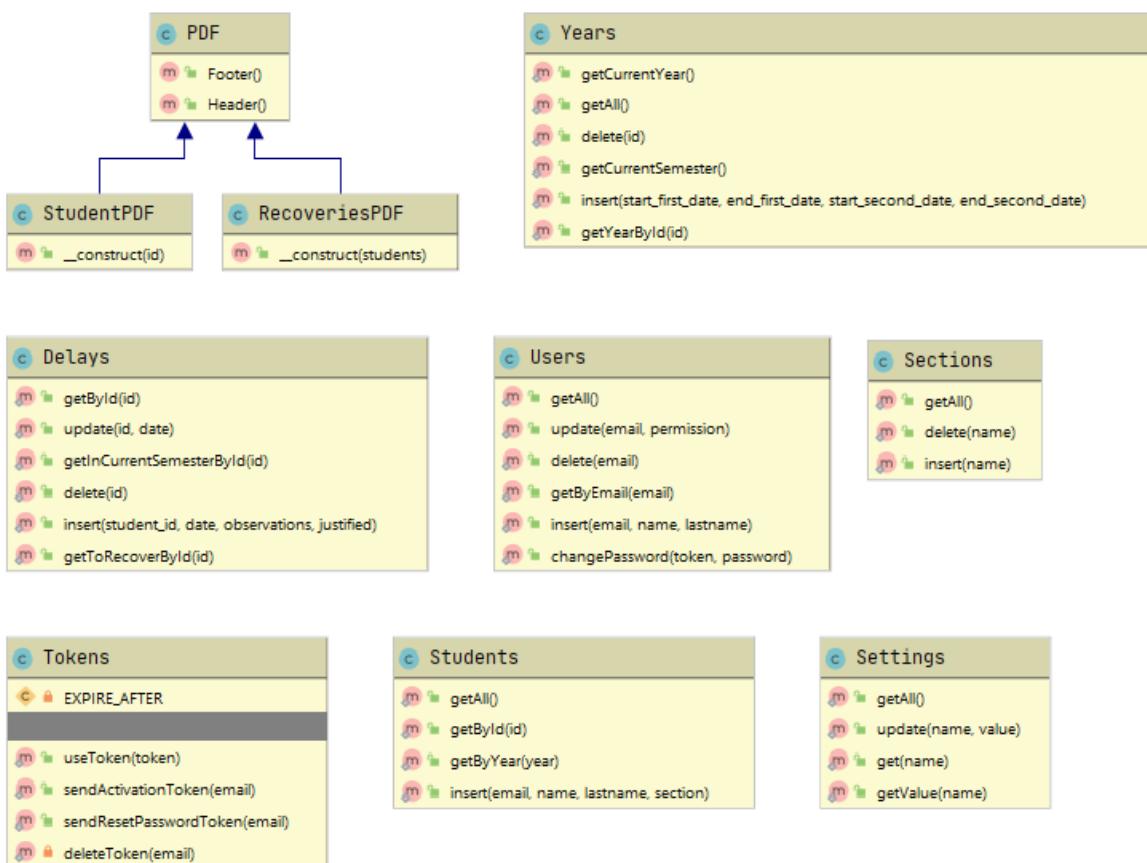


Figura 13 UML Classi models.

3.4 Design delle interfacce

3.4.1 Pagina di accesso

Questo è un mockup della pagina utilizzata per accedere all'applicativo web. Attraverso questa pagina è quindi possibile inserire username e password per poter accedere tramite il database locale.

Esegui accesso

Accedi

Figura 14 Mockup pagina di accesso.

3.4.2 Pagina cambio password

Questo è un mockup della pagina utilizzata per cambiare la password del proprio account utente all'interno dell'applicativo web. Questa pagina sarà accessibile solamente attraverso un token di recupero ricevuto tramite posta elettronica.

Cambia password

Cambia la password

Figura 15 Mockup pagina di cambio password.

3.4.3 Pagina di recupero password

Questo è un mockup della pagina utilizzata per richiedere una e-mail di recupero password. Inserendo l'email di un account registrato all'interno del sito web verrà inviato un messaggio di posta elettronica contenente un link che permetterà il cambio password.

Recupero password

Recupera password

Figura 16 Mockup pagina di recupero password.

3.4.4 Pagina di aggiunta ritardi

Questo è un mockup della pagina utilizzata per aggiungere ritardi agli studenti. Nella parte sinistra è presente una barra di navigazione la quale rimarrà anche nel resto delle pagine della dashboard, inoltre nella parte superiore destra verrà mostrato il nome utente di chi ha eseguito l'accesso. Attraverso questa pagina sarà dunque possibile aggiungere ritardi a studenti.

Ritardi	Utente
Studenti	
Nome Cognome	Aggiungi ritardo

Figura 17 Mockup pagina di aggiunta ritardi.

3.4.5 Pagina di aggiunta recuperi

Questo è un mockup della pagina utilizzata per aggiungere recuperi di ritardi agli studenti. Questa pagina permetterà dunque di aggiungere i recuperi dei ritardi eseguiti dagli studenti.

Ritardi	Utente
Recuperi	
Nome Cognome	Aggiungi recupero

Figura 18 Mockup pagina di aggiunta recuperi.

3.4.6 Pagina di gestione utenti

Questo è un mockup della pagina utilizzata per gestire gli utenti all'interno dell'applicativo web. Questa pagina sarà accessibile solamente agli amministratori e permetterà di creare nuovi utenti. Alla creazione di un utente esso riceverà un messaggio di posta elettronica contenente un link che permetterà il cambio password.

Ritardi	Utente
Utenti	
Nome Cognome	Modifica Elimina
	Aggiungi utente

Figura 19 Mockup pagina di gestione utenti.

3.4.7 Pagina impostazioni

Questo è un mockup della pagina utilizzata per cambiare le impostazioni dell'applicativo web. Anche questa pagina è accessibile solamente da parte degli amministratori dell'applicativo web e permetterà di modificare alcune impostazioni di esso.

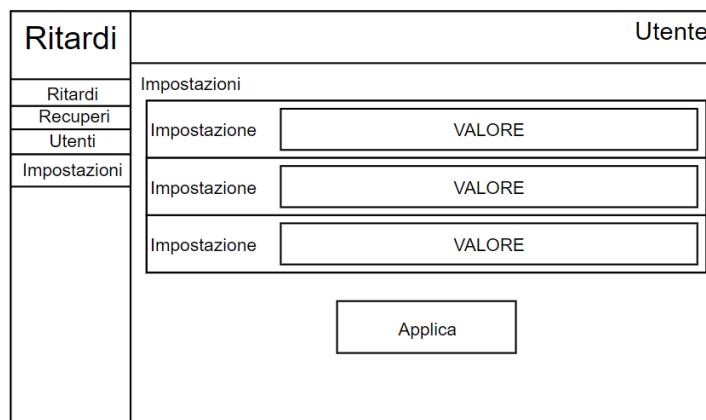


Figura 20 Mockup pagina impostazioni.

4 Implementazione

4.1 Gestione versioni

Per gestire i file che compongono l'intero progetto soprattutto il codice di esso ho utilizzato una repository GitLab messa a disposizione dalla scuola su un server interno da parte dei formatori. Questa repository mi permette di tenere traccia di tutti i cambiamenti che ho eseguito ai diversi file del progetto. Inoltre, caricando i file modificati sulla repository del progetto con messaggi adeguati alle modifiche eseguite mi permette di poter tornare in dietro nel codice in ogni momento e di tenere traccia di quanto fatto.

4.2 Gestore di pacchetti

Il progetto utilizza dei pacchetti esterni o librerie per funzionare correttamente. Per importare queste librerie esterne è stato utilizzato Composer. Composer è un gestore di pacchetti per il linguaggio di programmazione PHP. Esso permette dunque di scaricare le librerie e di generare i file di caricamento automatico dei vari pacchetti scaricati. Ho dunque creato il file di configurazione chiamato "composer.json" il quale permette di specificare diverse informazioni che andranno utilizzate dal gestore di pacchetti. All'interno del file di configurazione ho dunque impostato come dipendenza richiesta php-rest, la libreria che andrò ad utilizzare per lo sviluppo del sito web.

```
{  
    "name": "filippo/ritardi-web",  
    "description": "Gestione Web Ritardi Allievi SAMT",  
    "authors": [  
        {  
            "name": "Filippo Finke",  
            "email": "filippo.finke@samtrevano.ch"  
        }  
    ],  
    "minimum-stability": "dev",  
    "autoload": {
```

```
"psr-4": {
    "FilippoFinke\\": "src/"
},
"require": {
    "filippofinke/php-rest": "dev-master"
}
}
```

4.3 Database

4.3.1 Account di accesso

L'account di accesso al database è stato fornito dal formatore. Esso può creare un database e su di esso ha permessi completi (modifica struttura e dati).

4.3.2 Implementazione banca dati

Il database è stato implementato seguendo lo schema ER creato nel capitolo di progettazione. Non è stato necessario implementare nessuna tabella ponte aggiuntiva quindi anche il database finale è composto da sette tabelle. Il database inoltre può essere suddiviso in quattro parti differenti. La prima parte comprende le tabelle "USER" e "TOKEN" le quali sono collegate tra di loro attraverso delle chiavi esterne, questa parte viene utilizzata per la gestione degli account che potranno accedere all'applicativo. Successivamente vi sono tre tabelle "STUDENT", "SECTION" e "DELAY" le quali anch'esse collegate tra di loro attraverso chiavi esterne e vengono utilizzate per salvare i dati riguardanti gli allievi e i ritardi. Oltre a queste sono presenti ulteriori due tabelle completamente distaccate dal resto ovvero "SETTING" e "YEAR" le quali vengono utilizzate per salvare rispettivamente le impostazioni del sito web e i semestri scolastici. La parte che ritengo importante in quanto differente da quanto sviluppato in precedenza è la tabella "SETTING" essa è stata implementata nel seguente modo:

```
CREATE TABLE setting (
    name VARCHAR(50) PRIMARY KEY,
    value VARCHAR(255) NOT NULL
);
```

Anche se l'implementazione è molto semplice questa tabella è molto utile perché all'interno di essa è possibile salvare le impostazioni del sito web ed in futuro aggiungerne di nuove nel formato nome impostazione e valore senza dover eseguire grandi modifiche al codice.

4.3.3 Interrogazione database

Per eseguire la connessione al database e di conseguenza interrogarlo il framework php-rest mi permette di ricavare una connessione ad esso attraverso una classe statica chiamata "Database". A questa classe è possibile impostare i parametri di connessione quali: server, nome database, username e password. Per impostare questi parametri sono presenti dei metodi setter. Il codice dunque per impostare i parametri e ricavare una connessione è il seguente:

```
// Imposto indirizzo del server MySQL.
Database::setHost(DB_HOST);
// Imposto del database da utilizzare.
Database::setDatabase(DB_NAME);
```

```
// Imposto del nome utente per accedere al database.  
Database::setUsername(DB_USERNAME);  
// Imposto della password per accedere al database.  
Database::setPassword(DB_PASSWORD);  
// Controllo connessione alla banca dati.  
try {  
    // Successo  
    Database::getConnection();  
} catch (PDOException $e) {  
    // Errore  
}
```

Il metodo `getConnection()` verrà utilizzato da tutte le classi che dovranno interrogare il database. Questo metodo ritorna una connessione PDO, quindi una volta ricavata si potranno utilizzare tutti i metodi supportati dalla classe PDO di PHP.

La documentazione di PDO si può trovare al seguente indirizzo: <https://www.php.net/manual/en/book pdo.php>.

4.3.4 Schema generato

Questo è lo schema generato automaticamente attraverso l'utilizzo di phpMyAdmin sull'hosting esterno.

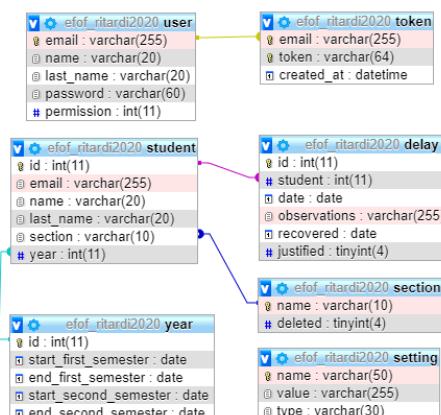


Figura 21 Schema database generato.

4.4 Applicativo web

4.4.1 Struttura

L'applicativo è stato sviluppato con l'ausilio di php-rest ed il template SB Admin 2. La struttura delle cartelle che compongono il codice è la seguente:

```
└── assets
└── src
    ├── Controllers
    ├── Libs
    ├── Middlewares
    ├── Models
    └── Views
└── vendor
```

La cartella "assets" contiene tutte le risorse statiche necessarie al funzionamento del sito web, queste risorse sono legate principalmente al frontend ed includono file di stile, JavaScript, immagini e fonts. I file che sono presenti in questa cartella vengono dunque serviti direttamente dal server web. Successivamente la cartella "src" contiene gran parte del codice dell'applicativo, all'interno di questa cartella vi sono altre cartelle le quali si dividono per funzione. La cartella "Controllers" contiene tutte le classi che gestiscono le risposte da dare ai vari percorsi quando vengono visitati da un utente. La cartella "Libs" contiene tutte le classi che vengono utilizzate all'interno del codice come ad esempio la classe di invio di posta elettronica. Successivamente la cartella "Middlewares" contiene tutte le classi che si occupano di stabilire gli accessi ai vari percorsi. All'interno della cartella "Models" invece vengono salvate tutte le classi utilizzate per interfacciarsi con la banca dati dell'applicativo web. Tutta la parte grafica del sito web viene salvata all'interno della cartella "Views". L'ultima cartella presente chiamata "vendor" contiene tutto il codice che è stato generato in modo automatico dal gestore di pacchetti Composer. L'applicativo web è stato sviluppato utilizzando due pattern architetturali, MVC e REST.

4.4.2 Model View Controller (MVC)

L'applicativo è in parte strutturato in modo tale da poter utilizzare il pattern MVC, ovvero viene diviso ciò che sono i dati dalle viste e dalla logica.

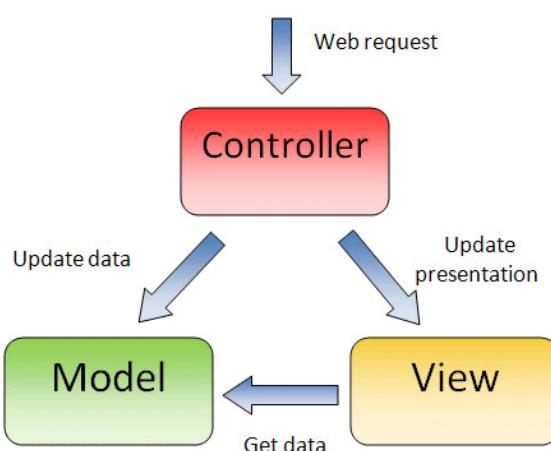


Figura 22 Schema pattern MVC.

Il pattern MVC è dunque diviso in tre categorie diverse:

- Controller, sono le classi che si occupano di collegare le interfacce grafiche con i dati.
- Model, sono le classi che si occupano di interagire con il database per la gestione dei dati.
- View, sono le interfacce grafiche che vengono mostrate all'utente.

Gestione Web Ritardi Allievi SAMT

All'interno del progetto "Gestione dei ritardi degli allievi SAMT" le classi "Controller" si occupano di collegare ciò che sono le viste con i dati ricavati dalla banca dati attraverso i "Model". Le classi "Model" dunque implementano dei metodi per facilitare l'interrogazione del database (come ad esempio: inserimento, aggiornamento ed eliminazione). Mentre le "View" sono semplicemente delle pagine HTML e PHP che verranno mostrate all'utente attraverso i "Controller".

4.4.3 Representational State Transfer (REST)

Un'altra parte dell'applicativo è stata sviluppata utilizzando il pattern architettonico REST. Ho deciso di utilizzare REST in modo da semplificare l'implementazione delle chiamate per quando riguarda aggiornamento, inserimento e cancellazione di dati.

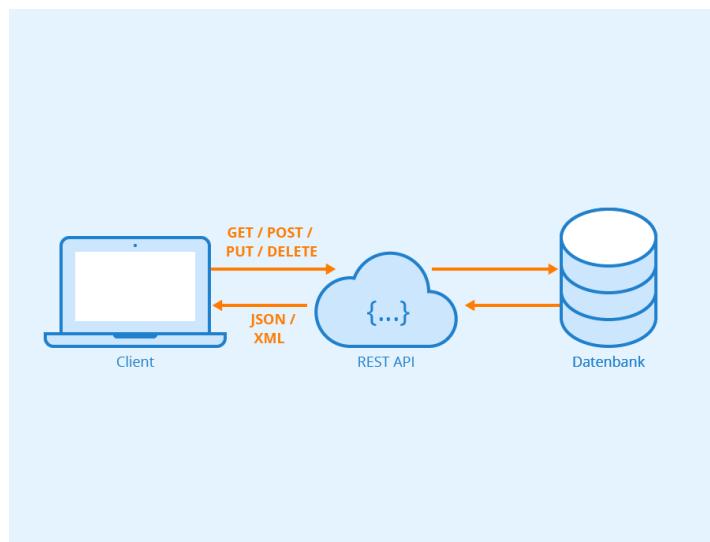


Figura 23 Schema pattern REST.

REST può essere utilizzato per separare completamente il frontend dal backend e quindi di poter supportare diverse interfacce grafiche in quanto solamente le chiamate HTTP alle API REST sono state definite. In questo caso REST viene utilizzato principalmente per rendere le azioni sui dati come ad esempio: inserimento, aggiornamento e cancellazione semplificate attraverso chiamate AJAX da parte di JavaScript, questo inoltre migliora l'esperienza utente in quanto il tutto avviene in background.

4.4.4 Routing delle richieste

Il framework php-rest mette a disposizione una classe "Router" la quale si occupa di inoltrare le richieste eseguite dagli utenti ai corrispettivi "Controllers". Per fare questo tutte le richieste che arrivano al webserver devono essere inoltrate ad un file nel quale viene istanziato questo oggetto. All'interno di questo progetto il file di entrata è chiamato "index.php" ed esso contiene tutto quello che riguarda il caricamento dei pacchetti esterni, l'aggiunta di percorsi, caricamento delle impostazioni e altro. Per direzionare tutte le richieste in arrivo al server web verso questo file viene utilizzato il file ".htaccess" il quale permette di stabilire delle regole che permettono di inoltrare le varie richieste in arrivo. Il file .htaccess contiene le seguenti regole:

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.+)\$ index.php [QSA,L]
```

La prima istruzione che viene data è quella di attivare il modulo "RewriteEngine" il quale permette di sovrascrivere le richieste. La seconda linea permette di specificare la base che verrà utilizzata per riscrivere le richieste in arrivo. La seconda e terza riga controllano se il file oppure la cartella specificata nella richiesta esista in questo modo è possibile servire file statici senza avere bisogno di passare attraverso il "Router" in PHP. Di conseguenza se le condizioni precedenti non sono state soddisfatte la richiesta viene inoltrata al file

“index.php” il quale si occuperà di instradare la richiesta al “Controller” adeguato. L’opzione “QSA” viene utilizzata per specificare di inoltrare anche parametri GET mentre l’opzione “L” dice al webserver di fermarsi se viene eseguita la regola e quindi di non eseguirne altre.

4.4.5 Configurazione

Il sito web contiene un file di configurazione che permette di specificare alcune impostazioni fondamentali. Attraverso questo file è possibile definire i parametri di connessione al database. All’interno del progetto è presente un file di configurazione di esempio chiamato “config.sample.php” il quale contiene una configurazione di esempio. Questo file è stato creato in modo da guidare l’amministratore alla configurazione del sito web basterà quindi rinominare questo file in “config.php” una volta che il sito web verrà caricato su un hosting. L’applicativo controlla la presenza del file di configurazione nel seguente modo:

```
// Controllo del file di configurazione.  
if (!file_exists("config.php")) {  
    $response = new Response();  
    $info = array(  
        "title" => "File di configurazione mancante!",  
        "message" => "Messaggio di errore."  
    );  
    $response->render(__DIR__ . "/src/Views/Error/error.php", $info);  
    exit;  
}  
// Includo del file di configurazione dell'applicativo.  
require __DIR__ . '/config.php';
```

Come prima cosa viene dunque controllata la presenza del file di configurazione “config.php”. Se il file non è presente viene creata una nuova risposta caricando la vista “error.php” e passandoci un array contenente le informazioni dell’errore. In questo modo è possibile utilizzare la pagina “error.php” per mostrare tutti gli errori in quanto essa si modifica dinamicamente in base a titolo e messaggio di errore. Mentre se il file di configurazione esiste viene semplicemente caricato.

4.4.6 Autenticazione

Il codice che si occupa di verificare le credenziali dell’utente all’accesso è presente nel controller di autenticazione all’interno di un metodo chiamato doLogin. Questo metodo si occupa di verificare che la combinazione e-mail e password che è stata passata dall’utente interrogando la banca dati. Il primo controllo che viene eseguito è la validità dell’e-mail, successivamente viene controllato se un utente con l’e-mail specificata è presente all’interno del database. Se l’utente è esistente ne viene verificata la password ed in caso di credenziali corrette viene impostata la sessione come utente autenticato.

```
public static function doLogin(Request $req, Response $res)  
{  
    $email = $req->getParam("email");  
    $password = $req->getParam("password");  
    if (isset($email) && Validator::isValidEmail($email) && isset($password) && Validator  
    or::isValidPassword($password)) {  
        $user = Users::getByEmail($email);  
        if ($user && password_verify($password, $user["password"])) {  
            unset($user["password"]);  
            Session::authenticate($user);  
        }  
    }  
}
```

```
        return $res->withStatus(200);
    }
}
return $res->withStatus(403);
}
```

Il metodo inoltre risponde alla richiesta con due codici di stato. Se lo stato della risposta è 403 vuol dire che le credenziali sono errate e che quindi l'accesso è stato negato, mentre se la risposta è 200 vuol dire che le credenziali inserite sono corrette e che di conseguenza l'accesso è stato eseguito con successo.

4.4.6.1 Gestione delle sessioni

Per gestire l'autenticazione dell'applicativo vengono utilizzate delle sessioni le quali mantengono lo stato dell'utente nelle varie pagine del programma. Per velocizzare e semplificare l'utilizzo delle sessioni è stata sviluppata una classe chiamata Session la quale permette di interfacciarsi con i metodi per la gestione delle sessioni del linguaggio in maniera semplificata. Ad esempio, per impostare la sessione di un utente come autenticato è presente il metodo authenticate il quale permette di segnare l'utente come autenticato ed inoltre permette il salvataggio di informazioni aggiuntive alla sessione (come ad esempio: Nome, Cognome, e-mail, etc.). Il codice del metodo authenticate è il seguente:

```
public static function authenticate($data = null) {
    $_SESSION["authenticated"] = true;
    if (is_array($data)) {
        foreach ($data as $key => $value) {
            $_SESSION[$key] = $value;
        }
    }
}
```

Per determinare se un utente ha eseguito l'accesso oppure no viene utilizzata la chiave **authenticated** all'interno della sessione. Inoltre, il metodo in questione accetta un array il quale può essere salvato all'interno della sessione stessa. All'interno della classe Session è inoltre presente un altro metodo che permette di verificare se l'utente corrente ha eseguito l'accesso:

```
public static function isAuthenticated() {
    return (isset($_SESSION["authenticated"]) && $_SESSION["authenticated"] == true);
}
```

Il codice si occupa dunque di controllare la presenza dell'indice **authenticated** all'interno della sessione e che esso sia true. Un altro metodo molto importante è logout, questo metodo permette di distruggere la sessione corrente e quindi permette di disconnettere un utente dall'applicativo web, il codice che esegue ciò è il seguente:

```
public static function logout() {
    session_unset();
    session_destroy();
}
```

In aggiunta al metodo di logout è presente anche un altro metodo utilizzato principalmente dalle interfacce grafiche che permette di ritornare il nome e cognome completo dell'utente corrente:

```
public static function getFullName()
{
    if (isset($_SESSION["name"]) && isset($_SESSION["last_name"])) {
        return $_SESSION["name"] . " ." . $_SESSION["last_name"];
    }
    return "Errore";
}
```

4.4.6.2 Gestione dei permessi

Per verificare i permessi assegnati ad un utente è presente una classe Permission la quale si occupa di tradurre il valore del permesso e di suddividerlo nelle varie attività che un utente può eseguire all'interno del sito. In questa classe sono dunque salvati i valori di ogni singolo permesso:

```
// Permesso di inserimento dei ritardi.
const INSERT = 1;
// Permesso di visione dei ritardi.
const SELECT = 2;
// Permesso di creazione dei PDF.
const CREATE = 4;
// Permesso di amministratore.
const ADMINISTRATOR = 8;
```

All'interno della classe inoltre, per ogni permesso è presente un metodo che si occupa di controllare che un determinato bit sia impostato eseguendo una semplice OR tra il permesso da verificare e il permesso dell'utente, ad esempio per determinare se un utente è amministratore è presente il metodo isAdmin:

```
public static function isAdmin($permission = null)
{
    if ($permission === null) {
        $permission = $_SESSION["permission"];
    }
    return ($permission & self::ADMINISTRATOR) === self::ADMINISTRATOR;
}
```

Il metodo si occupa dunque di ricavare il permesso come parametro oppure dalla sessione corrente ed esegue una OR tra il permesso passato e il valore del permesso di amministratore. Per ogni permesso sono presenti dei metodi molto simili a questo dove l'unica differenza è il permesso da controllare. Utilizzando questa classe è dunque possibile distinguere in modo semplice le azioni che potranno essere eseguite da un utente.

4.4.6.3 Gestione percorsi e accessi

Per determinare se un utente può accedere ad un determinato percorso vengono utilizzati dei Middlewares all'interno dell'applicativo. Queste classi si occupano dunque di stabilire se l'utente può eseguire l'accesso ad una determinata pagina oppure negarlo. Ad esempio, il Middleware utilizzato per determinare se un utente ha eseguito l'accesso all'applicativo è il seguente:

```
class AuthRequired
{
    public function __invoke($request, $response)
    {
```

```
if (!Session::isAuthenticated()) {  
    $response->redirect("/login");  
    exit;  
}  
}  
}
```

Quando la classe viene invocata viene dunque controllato lo stato della sessione dell'utente corrente. Se l'utente non ha eseguito l'accesso il codice inoltra l'utente alla pagina di login e termina. In questo modo bloccherà l'utente dal visitare pagine riservate solamente ad utenti che hanno eseguito il login. Sono presenti diversi Middlewares all'interno dell'applicativo e la struttura di essi è molto simile. Per creare un percorso attraverso la libreria php-rest è necessario istanziare un oggetto di tipo Router il quale permetterà di registrare tutti i percorsi di esso. Per creare un oggetto Router è sufficiente chiamare il suo costruttore:

```
// Creo un nuovo oggetto router che si occuperà di smistare le richieste.  
$router = new Router();
```

Successivamente sarà possibile assegnare qualsiasi percorso a questo oggetto ed esso verrà registrato in automatico. Questo oggetto supporta tutte le tipologie di richieste come ad esempio: GET, POST, PUT, DELETE etc. Per registrare un percorso è dunque possibile utilizzare la seguente sintassi:

```
$router->get('/percorso', 'Classe::metodo');  
$router->post('/percorso', 'Classe::metodo');  
$router->put('/percorso', 'Classe::metodo');  
$router->delete('/percorso', 'Classe::metodo');
```

È inoltre possibile raggruppare dei gruppi di percorsi in modo da poterli gestire meglio utilizzando un oggetto di tipo RouteGroup. Ad ogni percorso o gruppo di percorsi è possibile aggiungere dei middlewares prima che il percorso sia visitato oppure dopo attraverso i metodi before e after.

```
$homeRoutes = new RouteGroup();  
$homeRoutes->add(  
    $router->get("/", "FilippoFinke\Controllers\Home::index")  
)  
->before(new AuthRequired()); // Controllo autenticazione.
```

Ad esempio, in questo caso viene creato il gruppo che conterrà tutti i percorsi della dashboard principale. A questo gruppo è stato aggiunto un percorso. Tutte le richieste che verranno servite da questo gruppo eseguiranno come prima cosa il middleware AuthRequired il quale si occuperà di verificare se un utente ha eseguito l'accesso oppure no.

4.4.7 Invio di e-mail

Per rendere l'invio di messaggio di posta elettronica più semplice ed accessibile possibile ho creato una classe dedicata solamente a questa funzionalità. Questa classe contiene solamente un metodo importante chiamato send il quale si occupa di inviare messaggi di posta elettronica ad un indirizzo e-mail con un soggetto e del contenuto. Il contenuto del metodo è molto semplice in quanto si basa sulla funzione mail di PHP.

```
public static function send($to, $subject, $message){  
    $eol = "\r\n";  
    $headers = "From: <" . self::$fromEmail . ">" . $eol;  
    $headers .= "Content-Type: text/html; charset=UTF-8" . $eol;
```

```
    return mail($to, $subject, $message, $headers);
}
```

Il metodo si occupa di generare un header conforme allo standard richiesto per inviare messaggi di posta elettronica. In questo header viene impostato il mittente e la tipologia di contenuto del messaggio. Successivamente il metodo si basa sulla funzione mail di php la quale si occupa di collegarsi al server mail specificato nel file di configurazione del linguaggio di programmazione e di inviare il messaggio.

Nota: L'invio di e-mail dipende strettamente dal fornitore del servizio di posta elettronica, è possibile che le e-mail non vengano inviate a causa di filtri anti-spam, phishing e così via.

4.4.8 Validazione dei dati

La validazione dei dati è fondamentale, essa permette di aumentare la sicurezza dell'applicativo e di permettere di non avere incongruenze nei dati all'interno della banca dati. I controlli su di essi vengono effettuati sia lato client (quindi dal browser) e sia lato server (ovvero da PHP). In questo modo anche se il client venisse manipolato vi è un ulteriore controllo che non è possibile oltrepassare. Per la validazione lato server è stata creata una classe Validator la quale contiene tutti i metodi utilizzati per la validazione. Mentre per eseguire la verifica dei dati lato client l'applicativo si appoggia alle funzionalità di validazione di HTML5 e in alcuni casi di JavaScript. Come esempio per eseguire un controllo sul nome inserito da parte dell'utente lato server viene utilizzato il seguente codice:

```
public static function isValidName($name)
{
    return preg_match('/^([A-Za-zÀ-ÖØ-ংঃ]+){1,20}$/', $name);
}
```

Il metodo in questione accetta un parametro, il quale sarà la stringa da validare. Su questa stringa esegue una espressione regolare la quale controlla che siano presenti solamente dei caratteri dell'alfabeto con i vari accenti ed inoltre ne controlla la lunghezza. Per eseguire lo stesso controllo lato client viene utilizzata la proprietà pattern di un input la quale è messa a disposizione da HTML5:

```
<input pattern="[A-Za-zÀ-ÖØ-ংঃ]{1,20}" name="name" type="text" required>
```

In questo modo i controlli lato client corrispondono a quelli lato server. All'interno della classe Validator sono presenti svariati metodi di validazione come ad esempio: verifica email, verifica date, verifica numeri e così via.

4.4.9 Gestione dati ed interrogazione database

L'applicativo web utilizza un database MySQL per lo stoccaggio dei dati. Il sito dunque possiede delle classi specifiche le quali permettono di interrogare questa banca dati per ricavarne i dati da mostrare all'utente. Queste classi sono chiamate Model ed esse si occupano di mettere a disposizione del programmatore dei metodi utili per interrogare specifiche tabelle del database. Per ogni tabella presente nel database è dunque presente anche una classe Model la quale possiede la logica per gestirla. Tutte le classi Model interrogano il database attraverso una connessione ricavata da PDO ed utilizzano solamente Prepared Statements. Vengono utilizzati i Prepared Statements per prevenire attacchi di tipo SQL Injection in quanto questi metodi messi a disposizione da PHP si occupano di validare e verificare le query SQL.

4.4.9.1 Tabella user

La tabella user contiene tutti gli utenti che possono accedere all'applicativo web. Per la gestione di questa tabella è presente una classe model chiamata Users la quale contiene metodi per: ricavare tutti gli utenti registrati, ricavare singoli utenti, inserimento di nuovi utenti, aggiornamento di permessi, eliminazione e cambio password. Il primo metodo, getAll, permette di ricavare tutti gli utenti presenti nel database:

```
public static function getAll() {
    $pdo = Database::getConnection();
    $query = "SELECT * FROM user";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Il secondo metodo invece permette di ricavare tutti i dati di un solo utente attraverso il suo indirizzo email, esso si chiama getByEmail ed accetta come parametro l'e-mail attraverso la quale cercare l'utente:

```
public static function getByEmail($email)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM user WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':email', $email);
    try {
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Per inserire un nuovo utente all'interno della banca dati è presente il metodo insert, esso accetta tre parametri: e-mail, nome e cognome. Il metodo in questione si occupa di creare un nuovo utente inserendolo all'interno della banca dati e di utilizzare la classe model Tokens per inviare un codice di attivazione dell'account attraverso un messaggio di posta elettronica. Il codice è il seguente:

```
public static function insert($email, $name, $lastname)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO user VALUES(:email, :name, :lastname, '', 7)";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':email', strtolower($email));
    $stm->bindValue(':name', ucfirst($name));
    $stm->bindValue(':lastname', ucfirst($lastname));
    try {
        return $stm->execute() && Tokens::sendActivationToken($email);
    } catch (\PDOException $e) {
        throw new Exception("Un utente con questa email esiste già!");
    }
}
```

Il metodo seguente, chiamato update, permette di aggiornare il permesso di un utente. Il metodo accetta dunque due parametri, l'e-mail dell'utente da modificare e il valore del nuovo permesso da inserire.

```
public static function update($email, $permission)
{
    $pdo = Database::getConnection();
    $query = "UPDATE user SET permission = :permission WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':email', $email);
    $stm->bindValue(':permission', $permission);
    try {
        return $stm->execute();
    } catch (\PDOException $e) {
        return false;
    }
}
```

Per eliminare un utente è presente il metodo delete che attraverso l'e-mail può eliminare degli utenti dal database. Il codice per eliminare un utente è il seguente:

```
public static function delete($email)
{
    $pdo = Database::getConnection();
    $query = "DELETE FROM user WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':email', $email);
    try {
        return $stm->execute();
    } catch (\PDOException $e) {
        return false;
    }
}
```

L'ultimo metodo presente all'interno della classe Users si chiama changePassword e permette di cambiare la password di un utente in base ad un token di recupero ed una nuova password. Il metodo si appoggia dunque alla classe model Tokens per verificare che un token sia valido e che possa essere utilizzato prima di eseguire un aggiornamento alla banca dati, il codice è il seguente:

```
public static function changePassword($token, $password)
{
    if (Validator::isValidPassword($password)) {
        $email = Tokens::useToken($token);
        if ($email) {
            $pdo = Database::getConnection();
            $query = "UPDATE user SET password = :password WHERE email = :email";
            $stm = $pdo->prepare($query);
            $stm->bindValue(':email', $email);
            $stm->bindValue(':password', password_hash($password, PASSWORD_DEFAULT));
            try {

```

```
        $_SESSION["login_email"] = $email;
        return $stm->execute();
    } catch (\PDOException $e) {
    }
}
}
return false;
}
```

Viene inoltre impostata una variabile di sessione con l'e-mail dell'utente al quale è stata cambiata la password in modo da consigliarla all'accesso.

4.4.9.2 Tabella token

La tabella token viene utilizzata per salvare tutte le informazioni relative ai token di recupero generati dall'applicativo. Questa tabella permette di collegare una stringa generata in modo casuale ad un utente con la sua relativa data di creazione. Questo permette dunque di implementare delle funzionalità come ad esempio il recupero password di un utente. La classe model chiamata Tokens permette di generare ed inviare in modo automatico questi codici agli utenti. Il primo metodo della classe è sendActivationToken il quale accetta un indirizzo e-mail come parametro, viene utilizzato per generare un token nel futuro in modo che possa valere diversi giorni e viene utilizzato per l'attivazione di un profilo. Il codice è il seguente:

```
public static function sendActivationToken($email)
{
    $token = bin2hex(random_bytes(20));
    $hash = hash("sha256", $token);
    try {
        $pdo = Database::getConnection();
        $query = "INSERT INTO token VALUES(:email, :token, :created_at)";
        $stm = $pdo->prepare($query);
        $stm->bindValue('email', $email);
        $stm->bindValue('token', $hash);
        $time = time() + 86400 * 7;
        $stm->bindValue('created_at', date("Y-m-d", $time));
        $link = "http://" . $_SERVER['SERVER_NAME'] . BASE . "login/$token";
        $content = "Salve,<br>può accedere al suo account attraverso questo link: <a href='$link'$link</a><br><br>Esso ha una validità di 7 giorni.<br><br>Gestione Ritardi Web SAMT";
        return $stm->execute() && Mail::send($email, "Nuovo account | Gestione Ritardi", $content);
    } catch (\PDOException $e) {
    }
}
```

Successivamente è presente un metodo molto simile che permette di inviare un token di recupero password, il codice è il seguente:

```
public static function sendResetPasswordToken($email)
{
    if (Users::getByEmail($email)) {
```

```
    self::deleteToken($email);
    $token = bin2hex(random_bytes(20));
    $hash = hash("sha256", $token);
    try {
        $pdo = Database::getConnection();
        $query = "INSERT INTO token(email, token) VALUE(:email, :token)";
        $stm = $pdo->prepare($query);
        $stm->bindValue('email', $email);
        $stm->bindValue('token', $hash);
        $link = "http://" . $_SERVER['SERVER_NAME'] . BASE . "login/$token";
        $content = "Salve,<br>pù cambiare la sua password premendo il seguente link: <a href='$link'$link</a><br><br>Esso ha una validità di " . (self::EXPIRE_AFTER) .
        " minuti.<br><br>Gestione Ritardi Web SAMT";
        return $stm-
    }>execute() && Mail::send($email, "Recupero password | Gestione Ritardi", $content);
    } catch (\PDOException $e) {
    }
}
return true;
}
```

La classe mette a disposizione anche un metodo chiamato useToken che permette di controllare la validità di un token e di ritornare l'utente al quale esso era assegnato. Il metodo accetta dunque un parametro, ovvero il token da verificare. Il metodo è il seguente:

```
public static function useToken($token)
{
    $token = hash("sha256", $token);
    $pdo = Database::getConnection();
    $query = "SELECT email FROM token WHERE token = :token AND CURRENT_TIMESTAMP() - created_at <= " . (self::EXPIRE_AFTER * 60);
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue('token', $token);
        $stm->execute();
        $email = $stm->fetchColumn();
        if ($email) {
            $stm = $pdo->prepare("DELETE FROM token WHERE token = :token");
            $stm->bindValue('token', $token);
            $stm->execute();
            return $email;
        }
    } catch (\PDOException $e) {
    }
    return false;
}
```

L'ultimo metodo presente nella classe è: deleteToken. Esso viene utilizzato per eliminare tutti i codici assegnati ad un utente all'interno della tabella token attraverso il suo indirizzo e-mail. Il codice del metodo è il seguente:

```
private static function deleteToken($email)
{
    try {
        $pdo = Database::getConnection();
        $query = "DELETE FROM token WHERE email = :email";
        $stm = $pdo->prepare($query);
        $stm->bindValue('email', $email);
        return $stm->execute();
    } catch (\PDOException $e) {
    }
    return false;
}
```

4.4.9.3 Tabella student

La tabella student viene utilizzata per salvare tutte le informazioni relative agli studenti. Questa tabella possiede una classe model chiamata Students la quale contiene i seguenti metodi: getAll, getByYear, getById ed insert. Il metodo getAll viene utilizzato per ricavare tutti gli studenti presenti nel database ed il codice è il seguente:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM student ORDER BY name ASC";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Il metodo getByYear accetta come parametro l'identificativo di un anno, esso permette di ricavare tutti gli studenti in un determinato anno, il codice è il seguente:

```
public static function getByYear($year)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM student WHERE year = :year ORDER BY name ASC";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':year', $year);
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

```
}
```

Successivamente il metodo getById viene utilizzato per ricavare i dati di uno studente attraverso il suo identificativo univoco. Il metodo accetta dunque l'id come parametro e se uno studente è presente ne ritorna i dati, l'implementazione è la seguente:

```
public static function getById($id)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM student WHERE id = :id";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':id', $id);
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

L'ultimo metodo della classe è il metodo insert, esso accetta quattro parametri: e-mail, nome, cognome e sezione scolastica. Attraverso questo metodo è dunque possibile inserire un nuovo studente all'interno del sistema inserendo come anno scolastico quello corrente.

```
public static function insert($email, $name, $lastname, $section)
{
    $year = Years::getCurrentYear()["id"];
    $pdo = Database::getConnection();
    $query = "INSERT INTO student VALUES(null, :email, :name, :lastname, :section, :year)";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':email', strtolower($email));
    $stm->bindValue(':name', ucfirst($name));
    $stm->bindValue(':lastname', ucfirst($lastname));
    $stm->bindValue(':section', $section);
    $stm->bindValue(':year', $year);
    try {
        return $stm->execute();
    } catch (\PDOException $e) {
        throw new \Exception("Uno studente con questa email esiste già!");
    }
}
```

4.4.9.4 Tabella section

La tabella section contiene tutte le sezioni scolastiche presenti all'interno dell'applicativo. La classe model di gestione di questa tabella si chiama Sections e permette di ricavare tutte le sezioni, inserirne di nuove oppure di eliminarne già esistenti. Il primo metodo della classe si chiamata getAll e permette di ricavare tutte le sezioni presenti nella banca dati che non sono state segnate come eliminate:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM section WHERE deleted = 0";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Il secondo metodo, ovvero insert, permette di creare una nuova sezione all'interno dell'applicativo. Questo metodo accetta solamente un parametro il quale sarà il nome della sezione da aggiungere:

```
public static function insert($name)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO section VALUES(:name, 0)";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':name', $name);
    try {
        return $stm->execute();
    } catch (\PDOException $e) {
        throw new Exception("Un sezione con questo nome esiste già!");
    }
}
```

L'ultimo metodo presente nella classe è il metodo delete che anch'esso accetta solamente un parametro, ovvero il nome della sezione da eliminare. L'eliminazione avviene in modo soft, ovvero viene modificato un flag all'interno del database che segna la sezione come se fosse eliminato, questo permette dunque di eliminare delle sezioni non più utilizzate ma che rimarranno nello storico. L'implementazione è la seguente:

```
public static function delete($name)
{
    $pdo = Database::getConnection();
    $query = "UPDATE section SET deleted = :flag WHERE name = :name";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':flag', 1);
    $stm->bindValue(':name', $name);
    try {
        return $stm->execute();
    }
```

```
    } catch (\PDOException $e) {
        return false;
    }
}
```

4.4.9.5 Tabella delay

La tabella delay si occupa di immagazzinare tutte le informazioni riguardanti i ritardi degli studenti. Anche per questa tabella è presente una classe model chiamata Delays. All'interno di questa classe sono presenti metodi che permettono di: ricavare tutti i ritardi di uno studente, ricavare tutti i ritardi da recuperare, ricavare i ritardi nel semestre corrente, inserire ritardi, aggiornare lo stato di un ritardo oppure eliminarli. Il primo metodo permette di ricavare tutti i ritardi di uno studente, esso si chiama getById ed accetta come parametro l'id dello studente:

```
public static function getById($id)
{
    $pdo = Database::getConnection();
    $query = "SELECT id, DATE_FORMAT(date, '%d.%m.%Y') as 'date', observations, DATE_FORMAT(recovered, '%d.%m.%Y') as 'recovered', justified FROM delay WHERE student = :student ORDER BY delay.date DESC";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':student', $id);
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Il secondo metodo permette di ricavare i ritardi che devono essere recuperati da uno studente. Il codice è il seguente:

```
public static function getToRecoverById($id)
{
    $year = Years::getCurrentYear();
    if (!$year) return false;
    $pdo = Database::getConnection();
    $query = "SELECT id, DATE_FORMAT(date, '%d.%m.%Y') as 'date', observations, DATE_FORMAT(recovered, '%d.%m.%Y') as 'recovered', justified FROM delay WHERE student = :student AND recovered IS NULL AND justified = 0 AND date BETWEEN :start AND :end ORDER BY delay.date DESC";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':student', $id);
        $stm->bindValue(':start', $year["start_first_semester"]);
        $stm->bindValue(':end', $year["end_second_semester"]);
        $stm->execute();
        $delays = $stm->fetchAll(\PDO::FETCH_ASSOC);
    }
```

```
$max = Settings::getValue('max_delays');
if (count($delays) >= $max) {
    return array_slice($delays, ($max - 1));
} else {
    return [];
}
} catch (\PDOException $e) {
    return false;
}
}
```

Successivamente un metodo simile che permette di ricavare solamente i ritardi presenti nel semestre corrente.

```
public static function getInCurrentSemesterById($id)
{
    $semester = Years::getCurrentSemester();
    $start = $semester[0];
    $end = $semester[1];

    $pdo = Database::getConnection();
    $query = "SELECT id, DATE_FORMAT(date, '%d.%m.%Y') as 'date', observations, DATE_FORMAT(recovered, '%d.%m.%Y') as 'recovered', justified FROM delay WHERE student = :student AND date BETWEEN :start AND :end ORDER BY delay.date DESC";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':student', $id);
        $stm->bindValue(':start', $start);
        $stm->bindValue(':end', $end);
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Successivamente è presente il metodo insert che permette di inserire un nuovo ritardo ad uno studente, esso accetta quattro parametri: l'id dello studente, la data del ritardo, eventuali osservazioni e se il ritardo è giustificato oppure no. Il metodo insert si occupa inoltre di verificare il numero di ritardi da recuperare all'aggiunta del nuovo ritardo e se questo numero oltrepassa la soglia limite viene inviato un messaggio di posta elettronica che informa lo studente che dovrà eseguire un recupero del ritardo a scuola. Il codice è il seguente:

```
public static function insert($student_id, $date, $observations, $justified)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO delay VALUES(null, :student, :date, :observations, null, :justified)";
    $stm = $pdo->prepare($query);
```

```
$stm->bindValue(':student', $student_id);
$stmt->bindValue(':date', $date);
$stmt->bindValue(':observations', $observations);
$stmt->bindValue(':justified', $justified);

try {
    $stmt->execute();
    $id = $pdo->lastInsertId();
    if (count(self::getToRecoverById($student_id)) > 0 && !$justified) {
        $date = date("d.m.Y", strtotime($date));
        $email = Students::getById($student_id)["email"];
        Mail::send($email, 'Recupero ritardo | Gestione Ritardi', 'Salve,<br>lei ha
raggiunto il numero massimo di ritardi consentiti con il ritardo in data: ' . $date .
', verrà contattato per un recupero.');
    }
    return $id;
} catch (\PDOException $e) {
    throw new \Exception("Impossibile inserire il ritardo!" . $e->getMessage());
}
}
```

Per impostare un ritardo come recuperato è presente il metodo update che accetta come parametri l'identificativo del ritardo da aggiornare e la data di recupero. Il codice è il seguente:

```
public static function update($id, $date)
{
    $pdo = Database::getConnection();
    $query = "UPDATE delay SET recovered = :date WHERE id = :id";
    $stmt = $pdo->prepare($query);
    $stmt->bindValue(':id', $id);
    $stmt->bindValue(':date', $date);
    try {
        return $stmt->execute();
    } catch (\PDOException $e) {
        return false;
    }
}
```

Come ultimo metodo presente nella classe Delays vi è delete. Questo metodo permette di eliminare un singolo ritardo dal suo identificativo, l'implementazione del metodo è la seguente:

```
public static function delete($id)
{
    $pdo = Database::getConnection();
    $query = "DELETE FROM delay WHERE id = :id";
    $stmt = $pdo->prepare($query);
    $stmt->bindValue(':id', $id);
    try {
        return $stmt->execute();
    }
```

```
    } catch (\PDOException $e) {
        return false;
    }
}
```

4.4.9.6 Tabella year

La tabella year contiene le informazioni riguardo gli anni scolastici salvati all'interno dell'applicativo web. Questa tabella possiede una classe model chiamata Years che permette di interfacciarsi con la tabella stessa. Attraverso questa classe è possibile: ricavare tutti gli anni scolastici, ricavare l'anno scolastico corrente, ricavare il semestre corrente, inserire un nuovo anno scolastico oppure eliminarne uno. Il primo metodo, molto simile alle altre classi model, permette di ricavare tutti gli anni scolastici salvati all'interno della banca dati. Il metodo si chiama getAll e l'implementazione è la seguente:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM year";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Successivamente è presente il metodo getYearById che permette di ricavare un anno utilizzando il suo identificativo, il codice è il seguente:

```
public static function getYearById($id)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM year WHERE id = :id";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(":id", $id);
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Successivamente vi è il metodo getCurrentYear che viene utilizzato per ricavare l'anno corrente nel quale ci si trova rispetto agli anni scolastici presenti nel database, il metodo è il seguente:

```
public static function getCurrentYear()
{
    $pdo = Database::getConnection();
```

```
$query = "SELECT * FROM year WHERE CURRENT_DATE() >= start_first_semester AND CURRENT_DATE() <= end_second_semester;";
try {
    $stm = $pdo->query($query);
    return $stm->fetch(\PDO::FETCH_ASSOC);
} catch (\PDOException $e) {
    return false;
}
```

Un metodo di appoggio a getCurrentYear è il metodo getCurrentSemester che si occupa di ricavare il semestre corrente nel quale ci si trova in base alla data corrente. Questo metodo ritorna il semestre sotto forma di un array. L'implementazione è la seguente:

```
public static function getCurrentSemester()
{
    $year = self::getCurrentYear();
    if (!$year) return false;

    $start = $end = null;
    $start_first = \strtotime($year["start_first_semester"]);
    $end_first = \strtotime($year["end_first_semester"]);
    $current = time();
    if ($current >= $start_first && $current <= $end_first) {
        $start = $year["start_first_semester"];
        $end = $year["end_first_semester"];
    } else {
        $start = $year["start_second_semester"];
        $end = $year["end_second_semester"];
    }
    return array($start, $end);
}
```

Per inserire un anno scolastico all'interno del database dell'applicativo web è presente il metodo insert, esso si occupa di eseguire tutti i controlli per determinare che più anni o semestri non si sovrappongano. Il metodo insert accetta dunque quattro parametri: data di inizio del primo semestre, data di fine del primo semestre, data di inizio del secondo semestre e data di fine del secondo semestre. Il codice di inserimento è il seguente:

```
public static function insert($start_first_date, $end_first_date, $start_second_date, $end_second_date)
{
    $years = self::getAll();
    $start_date = \strtotime($start_first_date);
    $end_date = \strtotime($end_second_date);

    foreach ($years as $year) {
        $start_year = \strtotime($year['start_first_semester']);
        $end_year = \strtotime($year['end_second_semester']);
```

```
if (
    ($start_date <= $end_year && $start_date >= $start_year) ||
    ($start_date <= $start_year && $end_date >= $end_year)
) {
    throw new \Exception("È già presente un anno scolastico nello stesso intervallo di tempo!");
}

$pdo = Database::getConnection();
$query = "INSERT INTO year VALUES(null, :sfs, :efs, :sss, :ess)";
$stmt = $pdo->prepare($query);
$stmt->bindValue(':sfs', $start_first_date);
$stmt->bindValue(':efs', $end_first_date);
$stmt->bindValue(':sss', $start_second_date);
$stmt->bindValue(':ess', $end_second_date);
try {
    if ($stmt->execute()) {
        return $pdo->lastInsertId();
    }
} catch (\PDOException $e) {
}
return false;
}
```

In fine all'interno della classe Years è presente il metodo delete, che attraverso l'identificativo dell'anno scolastico permette di eliminarlo dalla banca dati. Il codice è il seguente:

```
public static function delete($id)
{
    $pdo = Database::getConnection();
    $query = "DELETE FROM year WHERE id = :id";
    $stmt = $pdo->prepare($query);
    $stmt->bindValue(':id', $id);
    try {
        return $stmt->execute();
    } catch (\PDOException $e) {
        return false;
    }
}
```

4.4.9.7 Tabella setting

All'interno del database è presente la tabella setting, questa tabella si occupa di gestire tutte le impostazioni del sito web. La classe model di gestione per questa tabella è chiamata Settings e permette di ricavare tutte le impostazioni, ricavare una impostazione dal suo nome, ricavare il valore di una impostazione oppure aggiornarne il valore. Non è possibile aggiungere nuovi valori di impostazioni in quanto esse verranno utilizzate solamente dall'applicativo. Il metodo per ricavare tutte le impostazioni è molto simile alle classi precedenti, il codice è il seguente:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM setting";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Successivamente è presente il metodo get il quale accetta come parametro il nome di una impostazione. Questo metodo ritorna tutte le informazioni su di essa, ovvero nome, valore e tipo. Il codice è il seguente:

```
public static function get($name)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM setting WHERE name = :name";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(':name', $name);
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Il metodo utilizzato più spesso di questa classe è il metodo getValue che attraverso il nome di una impostazione permette di ricavarne il valore, l'implementazione del metodo è la seguente:

```
public static function getValue($name)
{
    $setting = self::get($name);
    if ($setting) {
        return $setting["value"];
    }
    return null;
}
```

In fine è presente l'ultimo metodo update, esso accetta due parametri: l'impostazione e il nuovo valore da applicare. Questo metodo si occupa di controllare che l'impostazione da aggiornare esista e che la tipologia di dato passata dall'utente da assegnare come impostazione rispecchi la tipologia correlata con l'impostazione all'interno della banca dati. Il codice è il seguente:

```
public static function update($name, $value)
{
```

```
$setting = self::get($name);

if ($setting) {
    if (call_user_func_array(array("FilippoFinke\Libs\Validator", "isValid" . $setting["type"]), array($value))) {
        $pdo = Database::getConnection();
        $query = "UPDATE setting SET value = :value WHERE name = :name";
        try {
            $stm = $pdo->prepare($query);
            $stm->bindValue(':value', $value);
            $stm->bindValue(':name', $name);
            return $stm->execute();
        } catch (\PDOException $e) {
            return false;
        }
    } else {
        throw new Exception("Il valore deve essere di tipo: " . $setting["type"]);
    }
} else {
    throw new Exception("Impostazione inesistente");
}
}
```

Il metodo in questione si occupa dunque di eseguire un controllo sulla tipologia del dato chiamando in modo dinamico un metodo di validazione presente nella classe Validator per determinare se il nuovo valore dell'impostazione sarà valido oppure no.

4.4.10 Creazione PDF

Per generare i file PDF presenti nell'applicativo viene utilizzata una libreria chiamata FPDF. La documentazione della libreria è presente al seguente link: <http://www.fpdf.org/>. La creazione avviene dunque lato server attraverso PHP. È stata creata una classe PDF che estende la classe FPDF la quale esporta tutti i metodi per la creazione di file PDF, in questo modo sarà possibile sovrascrivere dei metodi che verranno chiamati in modo automatico, come ad esempio i metodi di generazione dell'intestazione e del piè di pagina. In questo file sono dunque presenti due metodi: header, footer. All'interno del metodo header è presente il codice che verrà utilizzato per la creazione dell'intestazione in tutte le pagine ed il codice è il seguente:

```
public function Header()
{
    $this->SetFont('Arial', 'B', 12);
    $this->Cell($this->GetPageWidth() - 20, 7, 'GESTIONE RITARDI', 'B', 0, 'T');
    $this->Ln(12);
}
```

Successivamente è presente il metodo footer che si occupa di creare un piè di pagina su tutte le pagine che verranno generate, l'implementazione è la seguente:

```
public function Footer()
{
    $this->SetY(-17);
```

```
$this->Cell(($this->GetPageWidth() - 20), 0, '', 'T');
$this->SetFont('Arial', '', 10);
$this->Cell(-18, 10, 'Pagina ' . $this->PageNo() . ' di {nb}', 0, 0, 'C');
$this->SetX(14);
$this->Cell(10, 10, date("d.m.Y"), 0, 0, 'C');
}
```

4.4.10.1 PDF Studente

Per la creazione del PDF personale dello studente è presente la classe StudentPDF che estende la classe PDF. Questa classe accetta come unico parametro l'identificativo di un utente e permette la creazione di un PDF personale attraverso l'utilizzo della libreria FPDF. La parte principale del PDF è la creazione della tabella, il codice è il seguente:

```
$this->Cell(25, 7, "Data", 1);
$this->Cell(25, 7, "Recupero", 1);
$this->Cell(25, 7, "Giustificato", 1);
$this->Cell(202, 7, "Osservazioni", 1);
$this->Ln();
$this->SetFont('Arial', '', 12);

foreach ($delays as $delay) {

    foreach ($delay as $key => $value) {
        $delay[$key] = iconv('UTF-8', 'windows-1252', $value);
    }

    $width = $this->GetStringWidth($delay["observations"]);
    $cellHeight = ceil($width / 202) * 7;

    $this->Cell(25, $cellHeight, $delay["date"], 1);
    $this->Cell(25, $cellHeight, $delay["recovered"] ?? "No", 1);
    $this->Cell(25, $cellHeight, ($delay["justified"]) ? "Si" : "No", 1);
    $this->MultiCell(202, 7, $delay["observations"], 1);
}

$this->Output('I', $student["name"] . $student["last_name"] . ".pdf", true);
```

Il codice in questione si occupa di creare una tabella contenente tutti i ritardi di uno studente nel semestre corrente e con le relative informazioni. Il risultato è il seguente:

GESTIONE RITARDI

Ritardi di Bryan Beffa dal 25.01.2020 al 29.06.2020.

Ritardi nel semestre: 3, da recuperare: 1.

Data	Recupero	Giustificato	Osservazioni
21.05.2020	No	No	Incidente BUS
21.05.2020	No	No	Ritardo treni
20.05.2020	No	No	Causa sveglia

26.05.2020

Pagina 1 di 1

Figura 24 Esempio PDF studente.

4.4.10.2 PDF Recuperi

Come per la creazione del file PDF degli studenti è presente una classe chiamata RecoveriesPDF che permette di generare il resoconto di chi deve recuperare i ritardi in un file PDF. La classe accetta come parametro un array di studenti che dovrà essere stampato all'interno del file. Seguendo lo stesso approccio della classe degli studenti viene creata una tabella nel seguente modo:

```
$this->SetFont('Arial', 'B', 12);
$this->Cell(87, 7, "Email", 1);
$this->Cell(100, 7, "Studente", 1);
$this->Cell(30, 7, "Sezione", 1);
$this->Cell(30, 7, "Ritardi", 1);
$this->Cell(30, 7, "Da recuperare", 1);
$this->Ln();
$this->SetFont('Arial', '', 12);

foreach ($students as $student) {

    foreach ($student as $key => $value) {
        $student[$key] = iconv('UTF-8', 'windows-1252', $value);
    }

    $y = $this->GetY();
    $this->MultiCell(87, 7, $student["email"], 1);
    // Calcolo l'altezza della cella.
    $h = $this->GetY() - $y;
    $this->SetY($y);
    $this->SetX(97);
    $this->Cell(100, $h, $student["name"] . " " . $student["last_name"], 1);
    $this->Cell(30, $h, $student["section"], 1);
    $this->Cell(30, $h, count($student["delays"]), 1);
}
```

```
$this->Cell(30, $h, count($student["to_recover"]), 1);  
$this->Ln($h);  
}
```

Il risultato del PDF dei recuperi è il seguente:

GESTIONE RITARDI				
Studenti che hanno dei ritardi da recuperare.				
Email	Studente	Sezione	Ritardi	Da recuperare
bryan.beffa@samtrevano.ch	Bryan Beffa	SAM I4AA	3	1

26.05.2020

Pagina 1 di 1

Figura 25 Esempio PDF recuperi.

4.4.11 Sicurezza

La sicurezza è molto importante in qualsiasi applicativo, soprattutto gli applicativi che vengono resi disponibili attraverso la rete. Per questo motivo anche per questo sito web sono state applicate delle misure di sicurezza in modo da coprire falle ed attacchi comuni.

4.4.11.1 Interrogazione database

Tutte le richieste che vengono eseguite da parte dell'applicativo al database utilizzano la classe PDO la quale è inclusa in PHP. Questa classe mette a disposizione dei metodi di interrogazione del database sicuri se usati nel modo corretto in quanto si occupano di validare e verificare le query. Utilizzando dunque PDO ed i Prepared Statements permette di proteggere l'applicativo da attacchi di tipo SQL Injection. Attacchi di questo tipo possono essere molto dannosi in quanto permettono ad un utente malintenzionato di eseguire query di sua volontà sulla banca dati dell'applicativo. Di seguito un esempio di come viene utilizzato PDO ed i Prepared Statements:

```
$pdo = Database::getConnection();  
$query = "INSERT INTO section VALUES(:name)";  
$stm = $pdo->prepare($query);  
$stm->bindValue(':name', $name);  
try {  
    return $stm->execute();  
} catch (\PDOException $e) {  
    // Errore  
}
```

Viene utilizzato il metodo `bindValue` il quale permette di sostituire una variabile all'interno della query da eseguire validando automaticamente il parametrazione che andrà sostituito in modo che la query non venga manipolata in nessun caso.

4.4.11.2 Salvataggio dei dati

Come citato in precedenza tutti i dati vengono validati prima di essere salvati all'interno della banca dati. Vengono eseguiti questi controlli sui dati in modo da prevenire attacchi XSS anche chiamati Cross-Site Scripting. Attacchi di tipo XSS permettono ad un utente malintenzionato di far eseguire del codice JavaScript malevolo da parte del sito web in modo involontario. Questo attacco può essere dunque molto pericoloso se il codice malevolo viene salvato all'interno della banca dati e poi distribuito a tutti gli utenti che visitano il sito web. Per ovviare a questo problema vengono dunque eseguiti controlli di validazione su tutti i dati che sono stati passati dagli utenti. In caso un campo sia completamente libero come ad esempio le osservazioni di un ritardo viene utilizzata la funzione `htmlspecialchars` di PHP la quale permette di rimpiazzare caratteri speciali in entità di HTML facendo in modo che la stringa non venga eseguita. L'utilizzo di questa funzione è molto semplice:

```
htmlspecialchars($observations)
```

La funzione in questione accetta dunque del testo e ne ritorna il testo con i caratteri speciali sostituiti da entità di HTML.

4.4.11.3 Salvataggio credenziali

Le credenziali degli utenti vengono salvate all'interno della banca dati dell'applicativo web sotto forma di una hash generata utilizzando l'algoritmo di cifratura `bcrypt`. Vengono salvate solamente le hash delle password all'interno del database in modo tale che anche se un malintenzionato riesca ad accedere al database in sola lettura non possa risalire alla password di ogni utente e di conseguenza non possa accedere all'applicativo. Il linguaggio PHP mette a disposizione due funzioni utilizzate principalmente per questo genere di cose, la prima funzione è chiamata `password_hash` e permette di creare una hash di una stringa di testo mentre la seconda è `password_verify` la quale permette di verificare che una stringa passata abbia ed una hash siano uguali. Queste due funzioni vengono dunque utilizzate per salvare e verificare le credenziali degli utenti. Per creare una hash la sintassi è la seguente:

```
$hash = password_hash($password, PASSWORD_DEFAULT);
```

Viene passato il parametro `PASSWORD_DEFAULT` alla funzione il quale indica di utilizzare l'algoritmo più sicuro presente in PHP. Al momento l'algoritmo considerato sicuro è `brypt`. Mentre per verificare se una password corrisponda ad una determinata hash viene utilizzata la funzione `password_verify`:

```
$equals = password_verify('password', 'hash');
```

Essa ritorna un valore booleano che determina se le due stringhe corrispondono alla stessa cosa. In questo caso l'hash da verificare verrebbe ricavata dalla banca dati.

4.4.11.4 Recupero password

Il recupero password viene eseguito attraverso un token di recupero che verrà inviato all'utente tramite un messaggio di posta elettronica. Questo token è composto da una serie di 20 byte casuali. Questo token viene salvato nella tabella token insieme all'indirizzo e-mail dell'utente che ha richiesto il recupero password. All'interno della tabella viene anche salvata la data di creazione del token in modo da poter stabilire una validità di tempo. All'interno del database però viene salvata solamente un hash generata con l'algoritmo sha256 del token in questo. In questo modo seguendo lo stesso principio del salvataggio delle password se un utente ha accesso al database in sola lettura non potrà eseguire il cambio password attraverso il token. Il token viene generato utilizzando la funzione `random_bytes` e `bin2hex` di PHP nel seguente modo:

```
$token = bin2hex(random_bytes(20));
```

Questo token verrà dunque inviato per e-mail all'utente mentre nel database ne verrà salvata l'hash in sha256:

```
$hash = hash("sha256", $token);
```

Successivamente per verificare se un token è valido viene controllata semplicemente la presenza di esso nella banca dati e che non sia scaduto attraverso la data di creazione.

4.4.12 Interfacce grafiche

4.4.12.1 Pagina di accesso

Questa è la pagina di accesso dell'applicativo web. La pagina chiede all'utente un indirizzo e-mail ed una password che verranno utilizzate come credenziali. Questa pagina utilizza una richiesta JavaScript al server per verificare che le credenziali inserite siano valide.

Nota: L'immagine nella pagina di accesso è stata presa dal sito freepik dall'autore stories.

- https://www.freepik.com/free-vector/login-concept-illustration_6183517.htm

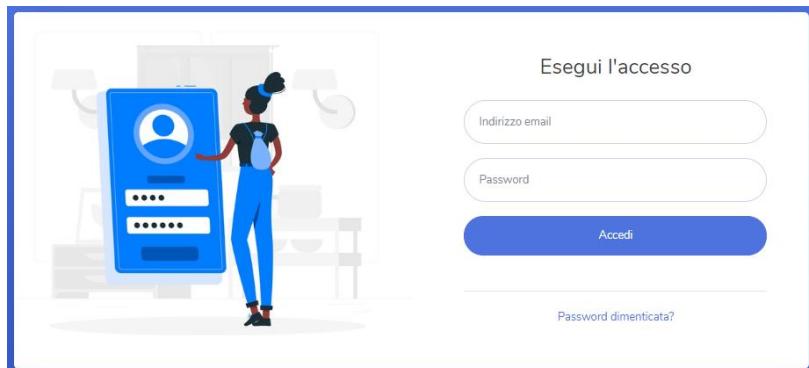


Figura 26 Pagina di accesso.

4.4.12.2 Pagina di cambio password

Questa è la pagina utilizzata per cambiare la password. Viene richiesta semplicemente la nuova password due volte.

Nota: L'immagine nella pagina di cambio password è stata presa dal sito freepik dall'autore stories.

- https://www.freepik.com/free-vector/login-concept-illustration_6183517.htm

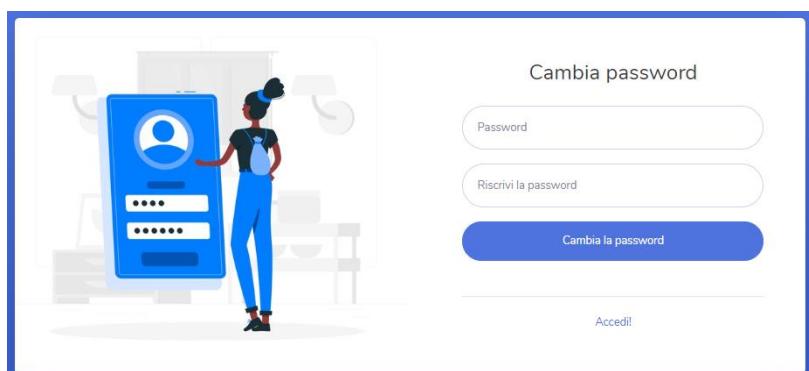


Figura 27 Pagina di cambio password.

4.4.12.3 Pagina di recupero password

Questa è la pagina di recupero password, richiede all'utente solamente l'indirizzo e-mail al quale verrà inviato un messaggio di posta elettronica contenente il token di recupero.

Nota: L'immagine nella pagina di cambio password è stata presa dal sito freepik dall'autore stories.

- https://www.freepik.com/free-vector/forgot-password-concept-illustration_7070629.htm

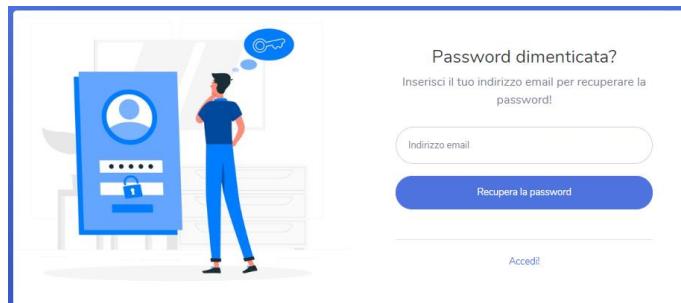


Figura 28 Pagina di recupero password.

4.4.12.4 Pagina di gestione ritardi

Questa è la pagina di gestione dei ritardi. A sinistra di tutte le pagine delle pagine di gestione sarà presente una sezione dedicata alla navigazione all'interno del sito web. All'interno della pagina di gestione dei ritardi è possibile creare dei nuovi studenti. Gli studenti verranno visualizzati all'interno di una tabella attraverso la quale sarà possibile visualizzare tutti i ritardi di uno studente, aggiungere i ritardi oppure creare un PDF.

Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni
Bryan	Beffa	SAM IAA	bryan.beffa@samtrevano.ch	3	1	<button>Visualizza</button> <button>Aggiungi ritardo</button> <button>Crea PDF</button>

Figura 29 Pagina di gestione ritardi.

4.4.12.5 Pagina di gestione recuperi

Questa è la pagina dedicata alla gestione dei recuperi dei ritardi da parte degli studenti. In questa pagina vengono mostrati solamente gli studenti che devono recuperare dei ritardi. Inoltre, è possibile visualizzare la lista di ritardi da recuperare attraverso il bottone visualizza. Attraverso questa pagina è possibile anche confermare il recupero di uno studente in modo che possa essere rimosso dalla lista.

Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni
Bryan	Beffa	SAM IAA	bryan.beffa@samtrevano.ch	3	1	<button>Visualizza</button>

Figura 30 Pagina di gestione recuperi.

4.4.12.6 Pagina di gestione utenti

Questa è la pagina di gestione degli utenti dell'applicativo. Attraverso questa pagina è possibile creare nuovi utenti che potranno accedere al sito web, alla creazione di un utente verrà inviata una e-mail di attivazione all'indirizzo di posta elettronica assegnato. Nella pagina è presente una tabella nella quale sono mostrati tutti gli utenti presenti nel sito web. Per ogni utente è possibile assegnare oppure rimuovere i singoli permessi ed inoltre eliminare l'utente. Non è possibile eliminare l'utente corrente, questo in modo tale che vi sia sempre un utente amministratore all'interno dell'applicativo.

Nome	Cognome	E-mail	Permessi	Azioni
Admin	Account	admin@samtrevano.ch	<input checked="" type="checkbox"/> Inserimento ritardi <input checked="" type="checkbox"/> Creazione PDF	<input checked="" type="checkbox"/> Visione ritardi <input checked="" type="checkbox"/> Amministratore Elimina
Filippo	Finke	filippo.finke@samtrevano.ch	<input type="checkbox"/> Inserimento ritardi <input type="checkbox"/> Creazione PDF	<input type="checkbox"/> Visione ritardi <input type="checkbox"/> Amministratore Elimina

Figura 31 Pagina di gestione utenti.

4.4.12.7 Pagina impostazioni

Questa è la pagina delle impostazioni dell'applicativo web. Attraverso questa pagina è dunque possibile gestire le sezioni scolastiche, gli anni scolastici presenti nell'applicativo e delle impostazioni di configurazione come ad esempio il numero massimo di ritardi consentiti prima di iniziare ad eseguire dei recuperi.

Nome	Azioni
SAM I4AA	Elimina

Primo semestre	Secondo semestre	Azioni
01.09.2019 - 16.01.2020	17.01.2020 - 25.06.2020	Elimina

Impostazione	Valore	Azione
from_email	gestione-ritardi@no-reply.ch	Modifica
max_delays	3	Modifica

Figura 32 Pagina impostazioni.

5 Test

5.1 Protocollo di test

Test Case:	TC-000	Nome:	L'applicativo deve essere web.
Riferimento:	REQ-000		
Descrizione:	L'applicativo deve essere accessibile ed interpretato da un browser.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete.		
Procedura:	<ol style="list-style-type: none"> 1. Aprire un browser (ES: Google Chrome). 2. Accedere all'applicativo con l'IP oppure il dominio della macchina. 		
Risultati attesi:	Accedendo all'applicativo web verrà mostrata una pagina di accesso.		

Test Case:	TC-001	Nome:	Accesso all'applicativo con credenziali valide.
Riferimento:	REQ-001		
Descrizione:	L'applicativo deve possedere una pagina di accesso che permette solamente agli utenti presenti nella banca dati di accedere e di rifiutare l'accesso in caso le credenziali non siano valide.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete.		
Procedura:	<ol style="list-style-type: none"> 1. Recarsi nella pagina di accesso dell'applicativo. 2. Inserire come indirizzo e-mail: "admin@samtrevano.ch" 3. Inserire come password: "123456" 4. Premere il bottone "Accedi" 		
Risultati attesi:	L'accesso all'applicativo avverrà con successo e l'utente verrà reindirizzato alla pagina principale del sito web.		

Test Case:	TC-002	Nome:	Accesso all'applicativo con credenziali invalide.
Riferimento:	REQ-001		
Descrizione:	L'applicativo deve possedere una pagina di accesso che permette solamente agli utenti presenti nella banca dati di accedere e di rifiutare l'accesso in caso le credenziali non siano valide.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete.		
Procedura:	<ol style="list-style-type: none"> 1. Recarsi nella pagina di accesso dell'applicativo. 2. Inserire come indirizzo e-mail: "nonesistente@mail.com" 3. Inserire come password: "123456" 4. Premere il bottone "Accedi" 		
Risultati attesi:	L'applicativo mostrerà un messaggio di errore informando l'utente che le credenziali inserite sono errate.		

Test Case:	TC-003	Nome:	Recupero password attraverso indirizzo e-mail.
Riferimento:	REQ-002		

Descrizione:	L'applicativo deve possedere una pagina che permette di recuperare la password in caso essa sia stata dimenticata attraverso l'indirizzo e-mail di un utente.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete.		
Procedura:	<ol style="list-style-type: none"> 1. Recarsi nella pagina di accesso dell'applicativo. 2. Premere sul testo: "Password dimenticata?" 3. Inserire l'e-mail del proprio profilo. 4. Premere il bottone "Recupera la password" 		
Risultati attesi:	L'applicativo mostrerà un messaggio di successo e verrà inviata una e-mail contenente un codice di recupero password.		

Test Case:	TC-004	Nome:	Cambio / Recupero password.
Riferimento:	REQ-001		
Descrizione:	L'applicativo deve possedere una pagina che permette di recuperare e cambiare la password del proprio account dopo averla richiesta tramite la pagina di recupero.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver richiesto un link di recupero password.		
Procedura:	<ol style="list-style-type: none"> 1. Navigare sul link presente all'interno della e-mail di recupero password. 2. Inserire come password: "123456" 3. Riscrivere la password: "123456" 4. Premere il bottone: "Cambia la password" 		
Risultati attesi:	L'applicativo mostrerà un messaggio di successo e reindirizzerà l'utente alla pagina di accesso.		

Test Case:	TC-005	Nome:	Pagina di gestione utenti
Riferimento:	REQ-003		
Descrizione:	L'applicativo deve possedere una pagina che permette gestire gli utenti che hanno accesso all'applicativo web. In questa pagina dovrà essere possibile vedere tutti gli utenti, aggiungerne di nuovi, modificare i permessi ed eliminarne. Dovrà sempre essere presente un amministratore.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Gestione Utenti" dalla barra di navigazione laterale 		
Risultati attesi:	Verranno mostrati tutti gli utenti in una tabella e non sarà possibile eliminare l'utente corrente.		

Test Case:	TC-006	Nome:	Aggiunta di un utente
Riferimento:	REQ-003		
Descrizione:	Attraverso la pagina di gestione utenti dovrà essere possibile aggiungere un nuovo utente. Alla creazione dell'utente verrà inviato per e-mail un link per l'attivazione del profilo.		

Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Gestione Utenti” dalla barra di navigazione laterale 2. Premere il pulsante “Aggiungi un utente” 3. Inserire come nome: “Test” 4. Inserire come cognome: “Test” 5. Inserire come e-mail: “test@test.com” 6. Premere il bottone “Crea”. 		
Risultati attesi:	Verrà inviato un messaggio di posta elettronica per attivare l'account alla e-mail dell'account che si vuole creare ed esso verrà mostrato nella lista degli utenti.		

Test Case:	TC-007	Nome:	Eliminazione di un utente
Riferimento:	REQ-003		
Descrizione:	Attraverso la pagina di gestione utenti dovrà essere possibile eliminare degli utenti tranne quello corrente.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Gestione Utenti” dalla barra di navigazione laterale 2. Premere il bottone: “Elimina” su un utente qualsiasi. 3. Premere il bottone “Ok” sul messaggio di conferma. 		
Risultati attesi:	L'utente verrà eliminato dal sito web e rimosso dalla tabella degli utenti.		

Test Case:	TC-008	Nome:	Aggiornamento permessi di un utente
Riferimento:	REQ-003		
Descrizione:	Attraverso la pagina di gestione utenti dovrà essere possibile aggiornare i permessi degli utenti tranne di quello corrente.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Gestione Utenti” dalla barra di navigazione laterale 2. Selezionare la spunta su uno o più permessi di un utente. 		
Risultati attesi:	I permessi dell'utente verranno aggiornati in modo automatico.		

Test Case:	TC-009	Nome:	pagina impostazioni
Riferimento:	REQ-004		
Descrizione:	L'applicativo deve possedere una pagina attraverso la quale deve essere possibile modificare le impostazioni del sito web.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Impostazioni” dalla barra di navigazione laterale. 		
Risultati attesi:	Verranno mostrate tre diverse tabelle contenenti le impostazioni correnti.		

Test Case:	TC-010	Nome:	Aggiunta di una sezione
Riferimento:	REQ-004		
Descrizione:	Attraverso la pagina delle impostazioni deve essere possibile creare una nuova sezione scolastica.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Impostazioni" dalla barra di navigazione laterale. 2. Premere il bottone: "Aggiungi una sezione" 3. Inserire come nome: "Test" 4. Premere il bottone: "Crea" 		
Risultati attesi:	Verrà creata una nuova sezione chiamata "Test" e verrà mostrata all'interno della tabella.		

Test Case:	TC-011	Nome:	Rimozione di una sezione
Riferimento:	REQ-004		
Descrizione:	Attraverso la pagina delle impostazioni deve essere possibile eliminare le sezioni scolastiche.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Impostazioni" dalla barra di navigazione laterale. 2. Premere il bottone: "Elimina" sulla sezione chiamata "Test" 3. Premere il bottone: "Ok" sul messaggio di conferma. 		
Risultati attesi:	La sezione verrà eliminata e rimossa dalla tabella.		

Test Case:	TC-012	Nome:	Aggiunta di un anno scolastico
Riferimento:	REQ-004		
Descrizione:	Attraverso la pagina delle impostazioni deve essere possibile aggiungere degli anni scolastici.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Impostazioni" dalla barra di navigazione laterale. 2. Premere il bottone: "Aggiungi un anno scolastico" 3. Inserire le seguenti date nella sezione "Primo semestre": <ol style="list-style-type: none"> a. 16.09.2019 b. 15.01.2020 4. Inserire le seguenti date nella sezione "Secondo semestre": <ol style="list-style-type: none"> a. 16.01.2020 b. 19.06.2020 5. Premere il bottone: "Crea" 		

Risultati attesi:	Verrà creato un nuovo anno scolastico ed aggiunto alla tabella.		
--------------------------	---	--	--

Test Case:	TC-013	Nome:	Rimozione di un anno scolastico
Riferimento:	REQ-004		
Descrizione:	Attraverso la pagina delle impostazioni deve essere possibile rimuovere degli anni scolastici.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Impostazioni” dalla barra di navigazione laterale. 2. Selezionare un anno scolastico e premere “Elimina”. 		
Risultati attesi:	L'anno scolastico verrà eliminato e rimosso dalla tabella.		

Test Case:	TC-014	Nome:	Modifica di una impostazione
Riferimento:	REQ-004		
Descrizione:	Attraverso la pagina delle impostazioni deve essere possibile aggiornare il valore di una impostazione.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Impostazioni” dalla barra di navigazione laterale. 2. Premere il bottone “Modifica” sull'impostazione “max_delays”. 3. Inserire il numero: “5” 4. Premere il bottone “Salva”. 		
Risultati attesi:	L'impostazione verrà aggiornata con successo.		

Test Case:	TC-015	Nome:	Pagina di gestione ritardi
Riferimento:	REQ-005		
Descrizione:	L'applicativo deve possedere una pagina attraverso la quale deve essere possibile gestire gli studenti e i ritardi.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Ritardi” dalla barra di navigazione laterale. 		
Risultati attesi:	Verrà mostrata una tabella con la lista degli studenti presenti nell'applicativo.		

Test Case:	TC-016	Nome:	Aggiunta di uno studente
Riferimento:	REQ-005		
Descrizione:	Attraverso la pagina di gestione dei ritardi deve essere possibile aggiungere un nuovo studente. Gli studenti dovranno avere l'email che termina con “@samtrevano.ch”		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare “Ritardi” dalla barra di navigazione laterale. 		

	<ol style="list-style-type: none"> 2. Premere il bottone: "Aggiungi uno studente" 3. Inserire come nome: "Test" 4. Inserire come cognome: "Test" 5. Inserire come e-mail: "test@samtrevano.ch" 6. Selezionare una sezione. 7. Premere il bottone: "Crea"
Risultati attesi:	Lo studente verrà creato ed aggiunto alla tabella.

Test Case:	TC-017	Nome:	Aggiunta di un ritardo		
Riferimento:	REQ-005				
Descrizione:	Attraverso la pagina di gestione dei ritardi deve essere possibile aggiungere un nuovo ritardo.				
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.				
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Ritardi" dalla barra di navigazione laterale. 2. Premere il bottone: "Aggiungi ritardo" sullo studente "Test" 3. Inserire come data: "25.05.2020" 4. Premere il bottone: "Inserisci" 				
Risultati attesi:	Verrà aggiunto un ritardo allo studente e il contatore nella tabella verrà incrementato.				

Test Case:	TC-018	Nome:	Visualizzazione ritardi		
Riferimento:	REQ-005				
Descrizione:	Attraverso la pagina di gestione dei ritardi deve essere possibile visualizzare tutti i ritardi di uno studente.				
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.				
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Ritardi" dalla barra di navigazione laterale. 2. Premere il bottone: "Visualizza" sullo studente "Test". 				
Risultati attesi:	Verrà aperto un modale contenente una tabella con tutti i ritardi dello studente.				

Test Case:	TC-019	Nome:	Creazione file PDF studente.		
Riferimento:	REQ-005				
Descrizione:	Attraverso la pagina di gestione dei ritardi deve essere possibile creare un file PDF contenente tutte le informazioni dello studente.				
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.				
Procedura:	<ol style="list-style-type: none"> 1. Selezionare "Ritardi" dalla barra di navigazione laterale. 2. Premere il bottone: "Crea PDF" sullo studente "Test". 				
Risultati attesi:	Verrà aperto un modale contenente un file PDF.				

Gestione Web Ritardi Allievi SAMT

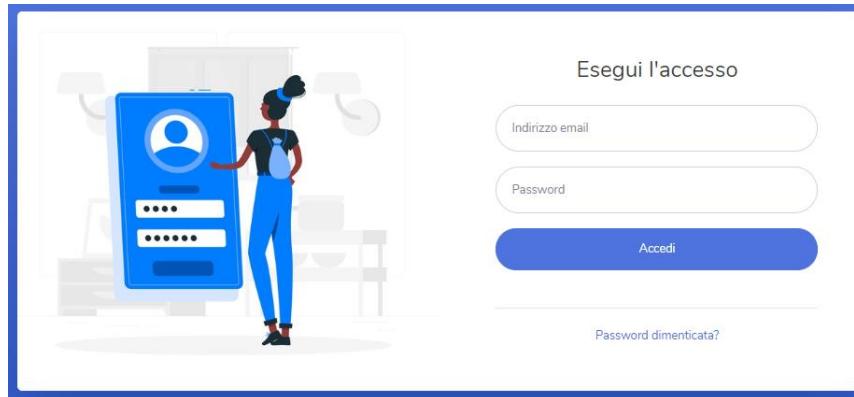
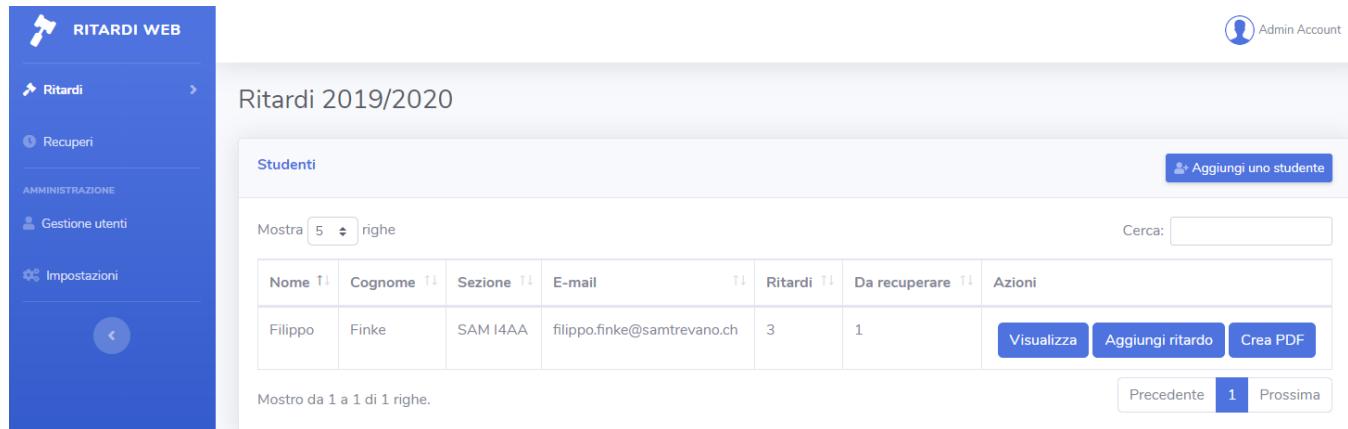
Test Case:	TC-020	Nome:	Pagina di gestione recuperi
Riferimento:	REQ-006		
Descrizione:	L'applicativo deve possedere una pagina attraverso la quale deve essere possibile gestire i recuperi dei ritardi degli studenti.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	1. Selezionare “Recuperi” dalla barra di navigazione laterale.		
Risultati attesi:	Verrà mostrata una tabella con la lista degli studenti che hanno almeno un ritardo da recuperare.		

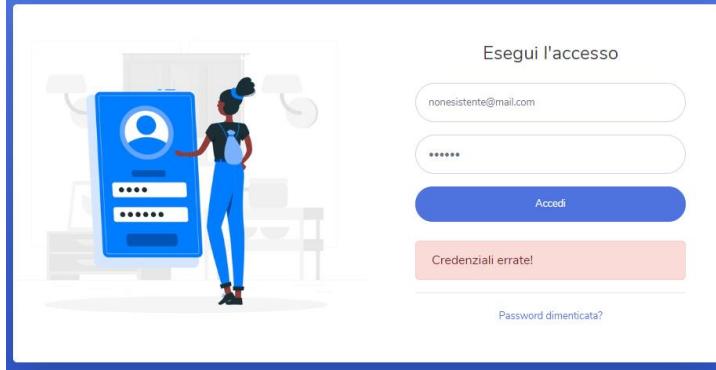
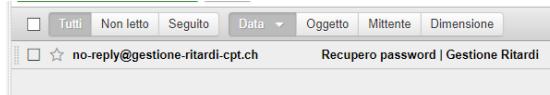
Test Case:	TC-021	Nome:	Aggiunta di un recupero
Riferimento:	REQ-006		
Descrizione:	Attraverso la pagina di gestione dei recuperi deve essere possibile aggiungere un recupero di un ritardo ad uno studente.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	1. Selezionare “Recuperi” dalla barra di navigazione laterale. 2. Premere il bottone: “Visualizza” su un utente qualsiasi. 3. Premere il bottone: “Recuperato” su un ritardo. 4. Inserire la data: “25.05.2020” 5. Premere il bottone: “Salva”		
Risultati attesi:	Verrà segnato il recupero del ritardo, decrementato il numero di ritardi da recuperare nella tabella degli studenti e disabilitato il bottone di recupero nel modale corrente.		

Test Case:	TC-022	Nome:	Creazione file PDF lista studenti che devono recuperare.
Riferimento:	REQ-006		
Descrizione:	Attraverso la pagina di gestione dei recuperi deve essere possibile creare un file PDF contenente gli studenti che devono recuperare dei ritardi.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
Procedura:	1. Selezionare “Recuperi” dalla barra di navigazione laterale. 2. Premere il bottone: “Crea PDF”		
Risultati attesi:	Verrà aperto un modale contenente un file PDF.		

Test Case:	TC-023	Nome:	Applicativo su hosting esterno
Riferimento:	REQ-007		
Descrizione:	L'applicativo deve essere messo in produzione su un hosting.		
Prerequisiti:	L'applicativo deve essere attivo su un hosting ed accessibile dalla rete.		
Procedura:	1. Navigare su “ http://samtinfo.ch/ritardi2020/ ”		
Risultati attesi:	Verrà mostrata la pagina di accesso dell'applicativo.		

5.2 Risultati test

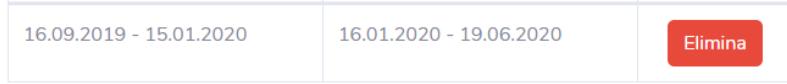
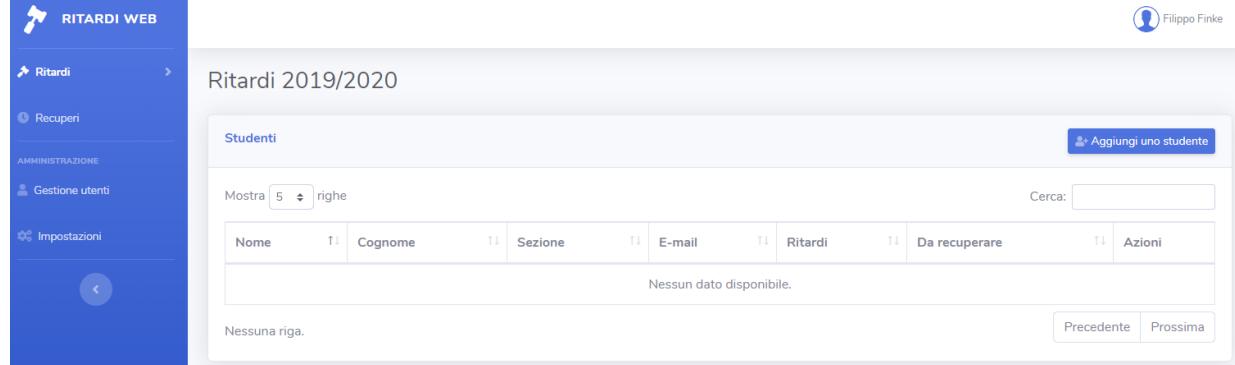
Codice test	Stato	Risultati														
TC-000	PASSATO	<p>Utilizzando Edge ed accedendo al sito web viene mostrata la pagina di accesso correttamente.</p>  <p>Esegui l'accesso</p> <p>Indirizzo email</p> <p>Password</p> <p>Accedi</p> <p>Forgot password?</p>														
TC-001	PASSATO	<p>L'accesso avviene correttamente e l'utente viene reindirizzato alla pagina principale di gestione dei ritardi.</p>  <p>Ritardi 2019/2020</p> <p>Studenti</p> <p>Mostra 5 righe</p> <table border="1"><thead><tr><th>Nome</th><th>Cognome</th><th>Sezione</th><th>E-mail</th><th>Ritardi</th><th>Da recuperare</th><th>Azioni</th></tr></thead><tbody><tr><td>Filippo</td><td>Finke</td><td>SAM I4AA</td><td>filippo.finke@samtrevano.ch</td><td>3</td><td>1</td><td>Visualizza Aggiungi ritardo Crea PDF</td></tr></tbody></table> <p>Mostro da 1 a 1 di 1 righe.</p> <p>Precedente 1 Prossima</p>	Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni	Filippo	Finke	SAM I4AA	filippo.finke@samtrevano.ch	3	1	Visualizza Aggiungi ritardo Crea PDF
Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni										
Filippo	Finke	SAM I4AA	filippo.finke@samtrevano.ch	3	1	Visualizza Aggiungi ritardo Crea PDF										

TC-002	PASSATO	L'applicativo mostra un errore tentando di accedere con credenziali non valide.  Figura 35 Pagina di accesso, errore credenziali.
TC-003	PASSATO	Viene mostrato il messaggio di successo e viene inviato il link di recupero password per e-mail.  Figura 36 E-mail recupero password.
TC-004	PASSATO	Viene mostrato un messaggio di successo e l'utente viene reindirizzato alla pagina di accesso.  Figura 37 Cambio password.

Gestione Web Ritardi Allievi SAMT

TC-005	PASSATO	Vengono mostrati tutti gli utenti presenti nel sito web e non vi è la possibilità di eliminare l'utente corrente.
		<p>The screenshot shows a user management interface. At the top, there's a navigation bar with 'RITARDI WEB' and links for 'Ritardi', 'Recuperi', 'AMMINISTRAZIONE', 'Gestione utenti', and 'Impostazioni'. Below the navigation is a search bar with placeholder 'Cerca:' and a button '+ Aggiungi un utente'. A table lists users with columns for Nome, Tipologia, E-mail, and Permessi. The 'Permessi' column includes checkboxes for 'Inserimento ritardi', 'Creazione PDF', 'Visione ritardi', and 'Amministratore'. Red 'Elimina' buttons are visible next to each user entry. At the bottom, it says 'Mostro da 1 a 2 di 2 righe.' and has 'Precedente' and 'Prossima' buttons.</p>
		Figura 38 Pagina di gestione utenti.
TC-006	PASSATO	Viene inviata una e-mail di attivazione del profilo e viene mostrato nella lista utenti.
		<p>The screenshot shows a user creation form. It has fields for 'Nome' (Test), 'Cognome' (Test), 'E-mail' (test@test.com), and checkboxes for 'Inserimento ritardi' (checked), 'Creazione PDF' (checked), 'Visione ritardi' (checked), and 'Amministratore' (unchecked). A red 'Elimina' button is also present.</p>
		Figura 39 Creazione utente.
TC-007	PASSATO	L'utente viene eliminato e rimosso dalla lista utenti.
TC-008	PASSATO	I permessi dell'utente vengono aggiornati.
TC-009	PASSATO	Vengono mostrate tutte le impostazioni del sito web in tre tabelle diverse.
		<p>The screenshot shows a settings page with three main sections: 'Sezioni', 'Anni scolastici', and 'Configurazione'. Each section has its own table with columns for Nome/Primo semestre, Azioni/Secondo semestre, and other details like 'Cerca:' and 'Modifica' buttons. The 'Sezioni' table shows one row for 'SAM 14AA'. The 'Anni scolastici' table shows two rows for '01.09.2019 - 25.01.2020' and '26.01.2020 - 29.06.2020'. The 'Configurazione' table shows one row for 'from_email' set to 'no-reply@gestione-ritardi-cpt.ch'.</p>
		Figura 40 Pagina impostazioni.

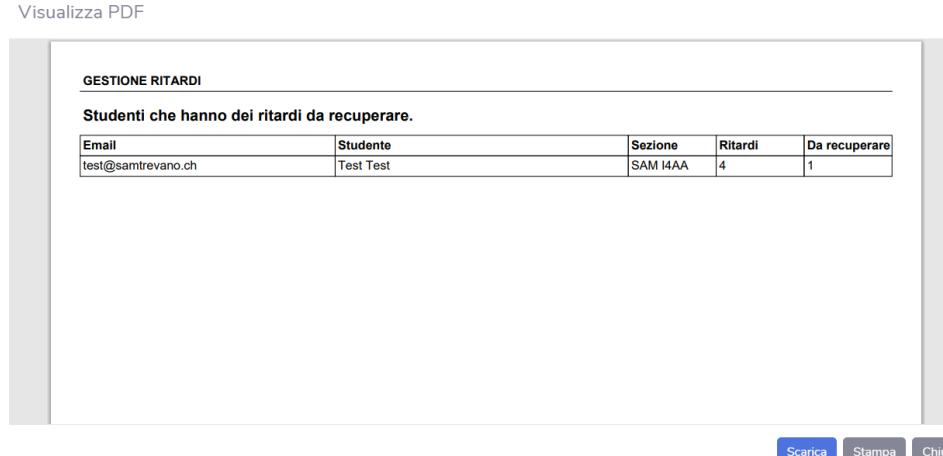
Gestione Web Ritardi Allievi SAMT

TC-010	PASSATO	La sezione viene aggiunta correttamente. 
TC-011	PASSATO	La sezione viene rimossa con successo.
TC-012	PASSATO	L'anno scolastico viene aggiunto correttamente. 
TC-013	PASSATO	L'anno scolastico viene rimosso con successo.
TC-014	PASSATO	L'impostazione viene aggiornata con successo. 
TC-015	PASSATO	La pagina di gestione ritardi viene visualizzata correttamente. 

TC-016	PASSATO	<p>Lo studente viene creato correttamente.</p> <table border="1"><tr><td>Test</td><td>Test</td><td>SAM I4AA</td><td>test@samtrevano.ch</td><td>0</td><td>0</td></tr><tr><td colspan="6"><button>Visualizza</button> <button>Aggiungi ritardo</button> <button>Crea PDF</button></td></tr></table> <p>Figura 45 Creazione studente.</p>	Test	Test	SAM I4AA	test@samtrevano.ch	0	0	<button>Visualizza</button> <button>Aggiungi ritardo</button> <button>Crea PDF</button>															
Test	Test	SAM I4AA	test@samtrevano.ch	0	0																			
<button>Visualizza</button> <button>Aggiungi ritardo</button> <button>Crea PDF</button>																								
TC-017	PASSATO	<p>Il ritardo viene aggiunto con successo e il contatore presente nella tabella viene incrementato.</p> <table border="1"><tr><td>25.05.2020</td><td>No</td><td>No</td><td><button>Elimina</button></td></tr></table> <p>Figura 46 Aggiunta ritardo.</p>	25.05.2020	No	No	<button>Elimina</button>																		
25.05.2020	No	No	<button>Elimina</button>																					
TC-018	PASSATO	<p>Viene mostrato un modale con tutti i ritardi dello studente.</p> <div data-bbox="729 722 1830 1214"><p>Visualizzazione di test@samtrevano.ch</p><table><tr><td>Mostra 5 righe</td><td>Cerca: <input type="text"/></td></tr><tr><th>Data</th><th>Osservazioni</th><th>Recuperato</th><th>Giustificato</th><th>Azione</th></tr><tr><td>25.05.2020</td><td></td><td>No</td><td>No</td><td><button>Elimina</button></td></tr><tr><td colspan="5">Mostro da 1 a 1 di 1 righe.</td></tr><tr><td colspan="2"><button>Precedente</button></td><td>1</td><td colspan="2"><button>Prossima</button></td></tr></table></div> <p>Figura 47 Modale di visualizzazione ritardi.</p>	Mostra 5 righe	Cerca: <input type="text"/>	Data	Osservazioni	Recuperato	Giustificato	Azione	25.05.2020		No	No	<button>Elimina</button>	Mostro da 1 a 1 di 1 righe.					<button>Precedente</button>		1	<button>Prossima</button>	
Mostra 5 righe	Cerca: <input type="text"/>																							
Data	Osservazioni	Recuperato	Giustificato	Azione																				
25.05.2020		No	No	<button>Elimina</button>																				
Mostro da 1 a 1 di 1 righe.																								
<button>Precedente</button>		1	<button>Prossima</button>																					

Gestione Web Ritardi Allievi SAMT

TC-019	PASSATO	<p>Viene mostrato un modale contenente il PDF.</p> <p>The screenshot shows a PDF document titled "GESTIONE RITARDI" with the subtitle "Ritardi di Test Test dal 16.01.2020 al 19.06.2020". It displays a table with three columns: Data, Recupero, Giustificato, and Osservazioni. A single row is present with the values: 25.05.2020, No, No, and an empty Osservazioni field. At the bottom of the PDF viewer are three buttons: "Scarica", "Stampa", and "Chiudi".</p>
TC-020	PASSATO	<p>Viene mostrata la pagina di gestione recuperi con la lista di studenti che devono recuperare dei ritardi.</p> <p>The screenshot shows the "Recuperi" page of the SAMT web application. On the left is a sidebar with navigation links: "Ritardi", "Recuperi" (which is highlighted), "AMMINISTRAZIONE", "Gestione utenti", and "Impostazioni". The main content area is titled "Recuperi" and contains a table with columns: Nome, Cognome, Sezione, E-mail, Ritardi, Da recuperare, and Azioni. A message at the bottom states "Nessun dato disponibile." Below the table are "Precedente" and "Prossima" buttons. In the top right corner, there is a user profile icon for "Filippo Finke".</p>

TC-021	PASSATO	<p>Il ritardo viene segnato come recuperato e il conteggio di ritardi da recuperare viene decrementato.</p> 										
TC-022	PASSATO	<p>Il PDF contenente la lista degli studenti che devono recuperare dei ritardi viene creato correttamente.</p>  <p>Visualizza PDF</p> <p>GESTIONE RITARDI</p> <p>Studenti che hanno dei ritardi da recuperare.</p> <table border="1"><thead><tr><th>Email</th><th>Studente</th><th>Sezione</th><th>Ritardi</th><th>Da recuperare</th></tr></thead><tbody><tr><td>test@samtrevano.ch</td><td>Test Test</td><td>SAM I4AA</td><td>4</td><td>1</td></tr></tbody></table> <p>Scarica Stampa Chiudi</p>	Email	Studente	Sezione	Ritardi	Da recuperare	test@samtrevano.ch	Test Test	SAM I4AA	4	1
Email	Studente	Sezione	Ritardi	Da recuperare								
test@samtrevano.ch	Test Test	SAM I4AA	4	1								
TC-023	PASSATO	L'applicativo è raggiungibile anche su un hosting esterno.										

5.3 Mancanze/limitazioni conosciute

Il progetto è stato sviluppato seguendo i requisiti descritti all'interno del Quaderno dei Compiti (QdC) senza tralasciare nulla. L'applicativo è dunque stato completato e non presenta delle mancanze o limitazioni.

6 Consuntivo

Rispetto alla pianificazione preventiva, nel Gantt consuntivo sono cambiati diversi tempi di diverse attività. Nella fase di implementazione, durante lo sviluppo del backend dell'applicativo web le attività di sviluppo e configurazione della base sul quale sviluppare l'intero prodotto sono state molto più veloci del previsto. Questo perché, utilizzando un framework php-rest, da me sviluppato mi ha permesso di avere una base solida sulla quale sviluppare il progetto e inoltre, avendo già una certa familiarità con essa mi ha permesso di velocizzare lo sviluppo dell'applicativo in generale. Inoltre, anche lo sviluppo della gestione dei ritardi è risultata molto più veloce rispetto alla pianifica iniziale. Anche per quanto riguarda lo sviluppo delle interfacce grafiche mi sono ritrovato in anticipo rispetto alla pianifica. Ho utilizzato un template chiamato SB Admin 2 per velocizzare lo sviluppo della parte grafica dell'applicativo, esso conteneva delle pagine di base già predefinite che quindi mi hanno permesso di risparmiare del tempo. Ho dunque terminato lo sviluppo del codice con circa due giorni di anticipo rispetto alla pianifica iniziale, tempo che ho dedicato alla stesura della documentazione e ad ulteriori test.

Gestione Web Ritardi Allievi SAMT

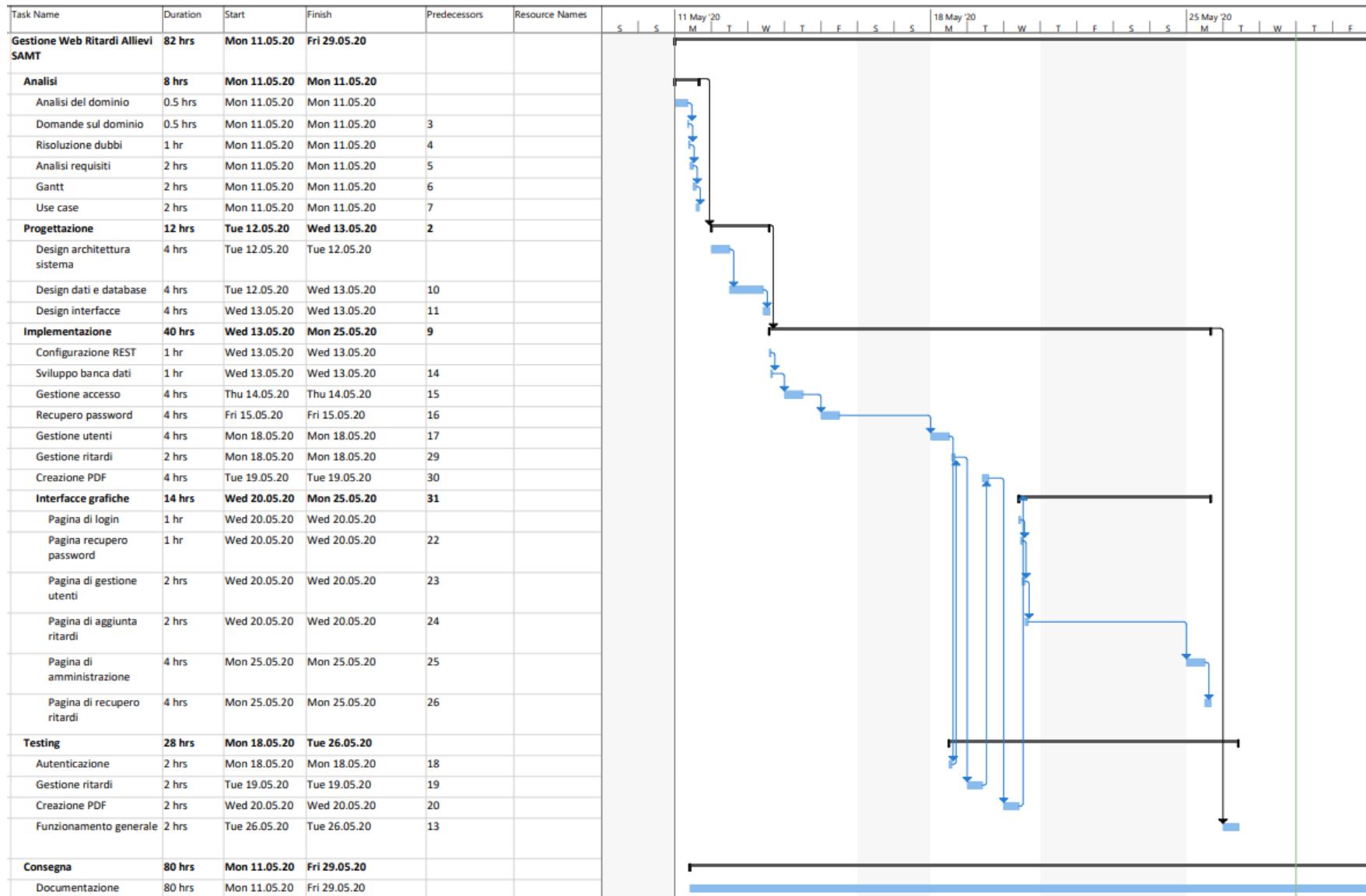


Figura 52 Diagramma di Gantt consuntivo.

7 Conclusioni

L'applicativo è stato sviluppato con lo scopo di velocizzare il processo di gestione dei ritardi della SAMT. Nel mercato esistono delle soluzioni che però sono indirizzate solamente a dipendenti di aziende per la gestione degli orari di lavoro. Il progetto che è stato sviluppato seguendo i requisiti del committente in modo da avere un applicativo web personalizzato e che dunque contenga tutte le funzionalità necessarie. L'applicativo sviluppato porterà dunque dei benefici alla SAMT per quanto riguarda la gestione dei ritardi e soprattutto la gestione dei recuperi di essi.

7.1 Sviluppi futuri

Sarebbe interessante espandere il progetto aggiungendo anche una sezione all'interno del sito che permetta agli studenti di accedervi e di visionare i dati riguardanti i propri ritardi. In questo modo si creerebbe un portale unico per la gestione dei ritardi all'interno di tutta la scuola. In questo modo ogni singolo studente avrebbe la possibilità di rimanere aggiornato sul suo stato dei ritardi semplicemente attraverso l'utilizzo del sito web. Essendo che a scuola è presente un dominio Active Directory sarebbe interessante integrare il server LDAP scolastico per gestire l'autenticazione degli utenti, permettendo dunque di accedere all'applicativo tramite gli account di dominio scolastici senza dovere creare manualmente degli utenti per i docenti e alunni. Integrando il sistema LDAP sarebbe possibile dunque ricavare in modo automatico le sezioni presenti nella scuola, tutti gli studenti e altre informazioni utili che al momento devono essere inserite manualmente all'interno dell'applicativo attraverso la pagina di impostazioni in modo da poterle consigliare all'utente durante l'inserimento di nuovi dati all'interno dell'applicativo. Sarebbe anche utile implementare un calendario nel quale mostrare tutti i ritardi di un allievo in modo tale di avere una visione generale dell'andamento di uno studente.

7.2 Considerazioni personali

Lo sviluppo di questo progetto è stato molto utile perché mi ha permesso di consolidare le mie conoscenze e tecniche di sviluppo nell'ambito della programmazione web. Una buona parte del progetto si basa su codice interamente scritto da me, compreso il framework php-rest, questo mi ha fatto dunque capire lo stile che posseggo di scrittura del codice, di mantenimento e usabilità del codice. Il progetto ha coperto tutti gli aspetti della programmazione web (login, gestione utenti, gestione dati, PDF, etc.) e questo mi ha permesso di applicare tutte le mie conoscenze nell'ambito e di poterle approfondire. Inoltre, sono contento che questo progetto verrà utilizzato all'interno della scuola e che permetterà di risparmiare del tempo ai docenti.

8 Glossario

Parola	Descrizione
AJAX	Tecnica di sviluppo che permette di sviluppare delle applicazioni interattive.
API	Application Programming Interface, ovvero un insieme di funzionalità raggruppate per funzionamento.
backend	Parte di un software che elabora i dati generati dal frontend.
bcrypt	È una funzione di hash.
Composer	Gestore di pacchetti aggiuntivi per il linguaggio PHP.
CSS	CSS (Cascading Style Sheet) è un linguaggio che descrive lo stile di file HTML.
flag	Un attributo che permette di determinare lo stato di un oggetto.
FPDF	Fpdf è una classe PHP che permette di generare file PDF.
framework	I framework in programmazione sono delle strutture già sviluppate da altri programmati che possono essere utilizzate.
frontend	Interfaccia accessibile da parte degli utenti, la parte grafica.
hash	È il risultato di una funzione di hash, un algoritmo che permette di generare una stringa di lunghezza fissa composta da caratteri casuali.
HTML	HyperText Markup Language, è un linguaggio di markup utilizzato per la formattazione e impaginazione di documenti ipertestuali.
JavaScript	JavaScript è un linguaggio di programmazione.
middlewares	Funzioni o classi che fungono da intermediari.
mockup	Rappresentazione di interfacce grafiche in modo generale.

modale	Schermata a comparsa.
MySQL	È un software per la gestione di database.
PDF	Portable Format Document, è un formato di file utilizzato per la rappresentazione di testo ed immagini sviluppato da Adobe.
PHP	Hypertext Preprocessor, linguaggio di programmazione lato server utilizzato spesso nello sviluppo web.
REST	Representational State Transfer, è uno stile di sviluppo di software.
Server web	Il server web è un'applicazione che è in grado di gestire delle richieste web.

9 Sitografia

- <http://draw.io/>, diagrams.net, 11.05.2020
- <https://github.com/filippofinke/php-rest>, Simple php rest api framework, 13.05.2020
- <https://www.php.net/manual/en/book pdo.php>, PHP: PDO - Manual, 13.05.2020
- <https://developer.mozilla.org/it/docs/Web/JavaScript/Reference>, Riferimento JavaScript, 20.05.2020
- <https://datatables.net/reference/option/language>, language, 20.05.2020
- <https://startbootstrap.com/themes/sb-admin-2/>, SB Admin 2 - Free Bootstrap..., 20.05.2020
- <https://api.jquery.com/>, jQuery API Documentation, 20.05.2020
- <https://datatables.net/manual/>, Manual, 20.05.2020
- <https://stackoverflow.com/>, mysql – Best way to store settings, 20.05.2020
- <https://getbootstrap.com/docs/4.1/getting-started/introduction/>, Introduction, 20.05.2020
- <http://www.fpdf.org/>, FPDF, 25.05.2020
- <https://www.bignames.co.uk/>, Common email problems, 26.05.2020
- <https://www.freepik.com/>, Mobile concepts, 27.05.2020

10 Allegati

- Manuale di installazione
- Diari di lavoro
- Quaderno dei compiti
- Codice sorgente presente su GitLab scolastico
 - http://gitsam.cpt.local/lavoro_finale_lpi_2020/gestione-web-ritardi-allievi-samt

Manuale di installazione



Manuale di installazione Gestione Web Ritardi Allievi SAMT

Titolo del progetto: Gestione Web Ritardi Allievi SAMT
Alunno/a: Filippo Finke
Classe: I4AC
Anno scolastico: 2019/2020
Docente responsabile: Fabrizio Valsangiacomo

Indice

1	Informazioni	3
2	Importare il database	3
3	Configurazione dell'applicativo web	3
4	Generazione file usando composer.....	4
5	Caricare l'applicativo su un host.....	4
6	Accesso all'applicativo	4
7	Configurazioni nel dettaglio	4
7.1	Configurazione errori.....	4
7.2	Configurazione percorso di base	5
7.3	Configurazione connessione al database	5

Indice delle figure

Figura 1 Importare il database su phpMyAdmin.....	3
---	---

1 Informazioni

L'applicativo richiede un server mail per funzionare in quanto esso utilizza la funzione mail di PHP: Se si vuole testare il progetto localmente consiglio l'utilizzo di Papercut, un software che simula il funzionamento di un server mail. È inoltre richiesta l'installazione di Composer in modo da potere scaricare le librerie aggiuntive utilizzate.

Papercut: <https://github.com/ChangemakerStudios/Papercut-SMTP>

Composer: <https://getcomposer.org/download/>

2 Importare il database

La prima cosa da fare per installare l'applicativo è importare il database. Il file contenente il database si trova nella cartella **Database** del progetto. Per importare il database utilizzando phpMyAdmin bisogna per prima cosa selezionare la banca dati nel quale verranno importate le varie tabelle, dopodiché selezionare la sezione **Importa** e successivamente selezionare il file del database da importare.

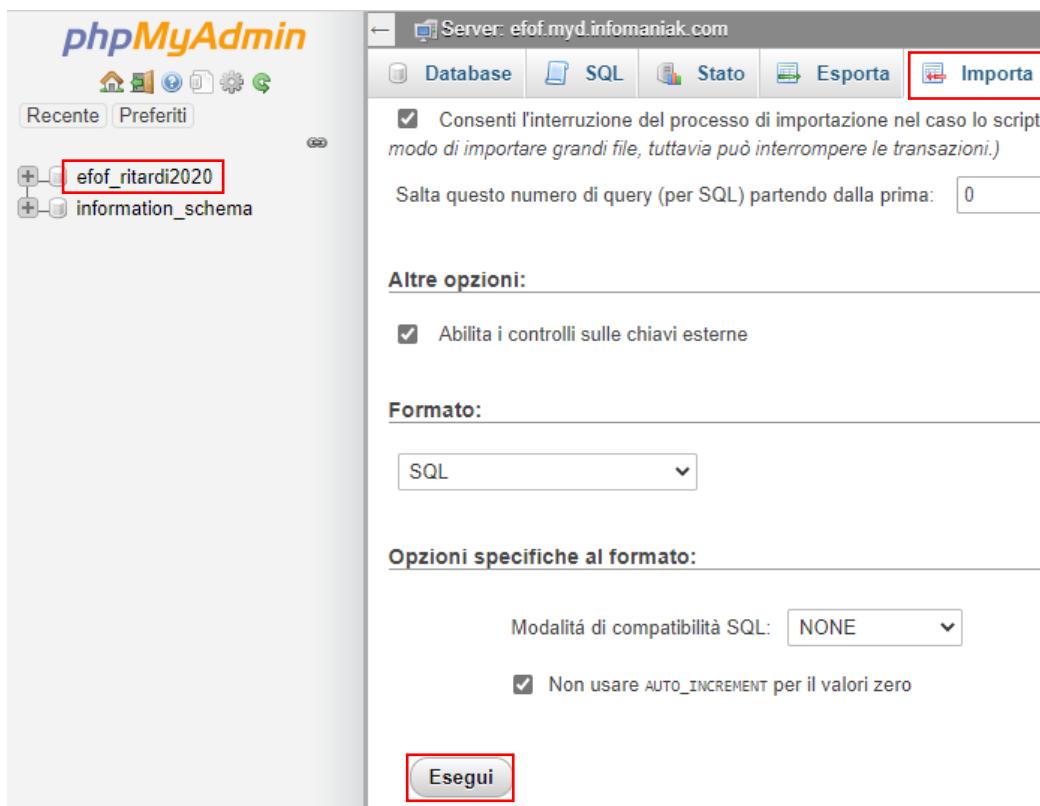


Figura 1 Importare il database su phpMyAdmin.

3 Configurazione dell'applicativo web

Una volta importato il database è necessario eseguire una adeguata configurazione per il corretto funzionamento del sito web. Per eseguire la configurazione copiare il file chiamato **config_sample.php** e rinominarlo in **config.php**. In questo file è possibile modificare diverse impostazioni che andranno a modificare il programma dinamicamente. La parte più importante è la connessione alla banca dati la quale è da configurare specificando server, nome database e credenziali di accesso:

```
// Indirizzo di connessione del database.  
define("DB_HOST", "IP_SERVER_DATABASE");  
// Il nome del database.  
define("DB_NAME", "NOME_DATABASE");  
// L'username dell'utente da utilizzare per collegarsi al database.  
define("DB_USERNAME", "USERNAME");  
// La password da utilizzare per il collegamento al database.  
define("DB_PASSWORD", "PASSWORD");
```

Inoltre, se l'applicativo si trova in una cartella che non sia quella principale del web server è necessario modificare il **BASE** nel file di configurazione:

```
define("BASE ", "/CARTELLA");
```

La stessa modifica va fatta all'interno del file **.htaccess** nel quale si deve specificare la stessa cartella:

```
RewriteBase /CARTELLA
```

4 Generazione file usando composer

Una volta eseguita la configurazione è necessario utilizzare composer per generare i file necessari al funzionamento del progetto. Utilizzare dunque i seguenti comandi:

```
composer install  
composer -o dumpautoload
```

5 Caricare l'applicativo su un host

Dopo aver eseguito i punti precedenti è ora possibile caricare tutto il contenuto dell'applicativo web presente nella cartella **Sito o applicativo** sul proprio hosting.

6 Accesso all'applicativo

Per accedere al pannello di gestione dell'applicativo l'account di default dopo l'installazione è:

```
Username: admin@samtrevano.ch  
Password: 123456
```

È consigliato creare subito un nuovo account amministratore attraverso il pannello di gestione utenti ed eliminare l'account predefinito.

7 Configurazioni nel dettaglio

7.1 Configurazione errori

Nel file di configurazione sono presenti tre regole utilizzate per impostare il livello degli errori che andranno mostrati all'interno dell'applicativo. Le righe di configurazione sono le seguenti:

```
ini_set('display_errors', 1);  
ini_set('display_startup_errors', 1);  
error_reporting(E_ALL);
```

Ulteriori informazioni sui livelli di errore sono presenti al seguente riferimento:
<https://www.php.net/manual/en/function.error-reporting.php>

7.2 Configurazione percorso di base

La configurazione del percorso di base dell'applicativo avviene attraverso la modifica di una riga presente nel file di configurazione. Questa modifica va effettuata solamente se l'applicativo è stato installato in una cartella che non sia quella principale del web server. Il formato deve essere "/CARTELLA_INSTALLAZIONE".

```
define("BASE ", "/");
```

7.3 Configurazione connessione al database

Per eseguire la connessione al database è necessario modificare le credenziali di accesso, il nome ed il server del database per permettere all'applicativo di potersi collegare ed interrogare la banca dati. Le righe del file di configurazione sono le seguenti:

```
// Indirizzo di connessione del database.  
define("DB_HOST", "SERVER_IP");  
// Il nome del database.  
define("DB_NAME", "DATABASE");  
// L'username dell'utente da utilizzare per collegarsi al database.  
define("DB_USERNAME", "USERNAME");  
// La password da utilizzarre per il collegamento al database.  
define("DB_PASSWORD", "PASSWORD");
```

Diari di lavoro

Diario di lavoro

Luogo	Balerna
Data	11.05.2020

Lavori svolti

La giornata di lavoro di oggi è stata dedicata interamente ad eseguire l'analisi sul Quaderno Dei Compiti consegnatoci dal formatore. Ho dunque ricevuto il QdC del progetto che dovrò andare a svolgere e successivamente ho avuto il tempo di porre delle domande al formatore in modo tale da risolvere i miei dubbi.

Come viene eseguito il login? *Uguale sia per utenti che amministratori.*

La registrazione avviene inserendo il nome, il cognome, email come username sia per utenti che amministratori? *Sì, per entrambi.*

Con il sistema del calendario cosa intende? *Semplice menu a comparsa come calendario.*

I ritardi accumulati quindi vanno recuperati anche nel secondo semestre? *Sì.*

Per fare in modo che un ritardo sia visibile ma non conteggiato, che cosa intende? *Il ritardo è stato giustificato, viene mostrato ma non calcolato come recupero.*

L'email di notifica avvisa solamente lo studente che deve recuperare un ritardo, giusto? *Esatto, solamente di notifica.*

Su che tipo di hosting "interno" andrà il progetto? *Sarà su Infomaniak.*

Successivamente mi sono occupato di scrivere i seguenti capitoli all'interno della documentazione:

- Informazioni sul progetto
- Abstract
- Scopo
- Analisi
 - o Analisi del dominio
 - o Analisi e specifica dei requisiti
- Use case
- Pianificazione
 - o Analisi
 - o Progettazione
 - o Implementazione
 - o Testing
 - o Consegna
- Analisi dei mezzi
 - o Software
 - o Hardware

Ho dunque creato il diagramma d'uso, ovvero il seguente:

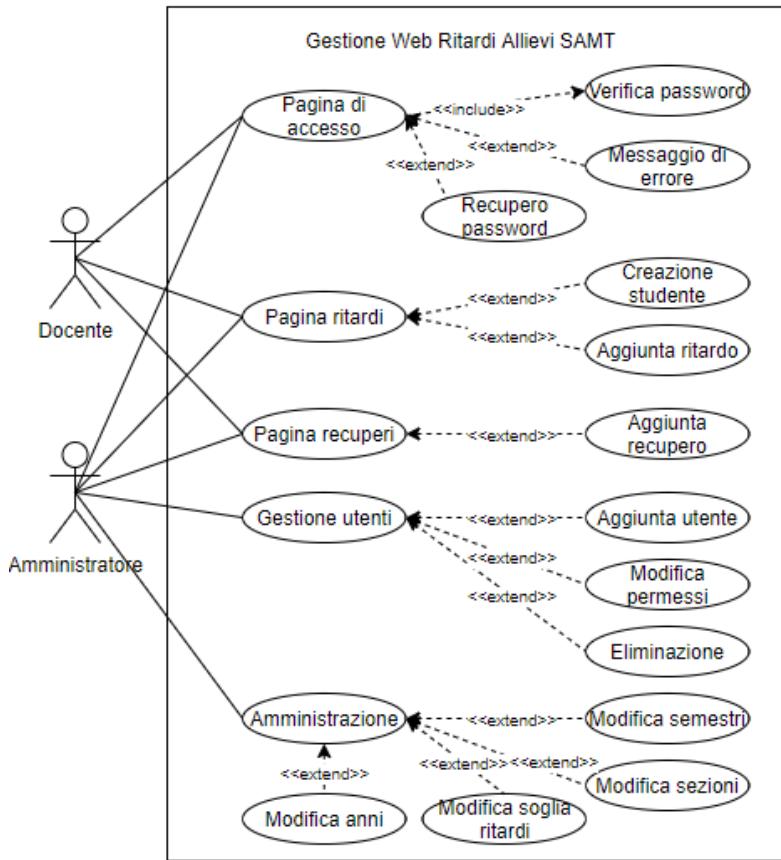


Figura 1 Diagramma d'uso.

Ho inoltre creato anche la pianifica preventiva che andrò a seguire durante il corso del progetto. Il tutto è stato caricato anche su GitLab.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Mi trovo in linea con la pianificazione preventiva.

Programma di massima per la prossima giornata di lavoro

Iniziare e completare il capitolo di progettazione.

Diario di lavoro

Luogo	Balerna
Data	12.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato di correggere il GANTT seguendo alcune indicazioni che mi sono state date dal Perito e di conseguenza ne ho aggiornato anche la documentazione. Successivamente mi sono occupato della creazione dello schema ER del database che verrà poi utilizzato come riferimento per l'implementazione. Lo schema ER è il seguente:

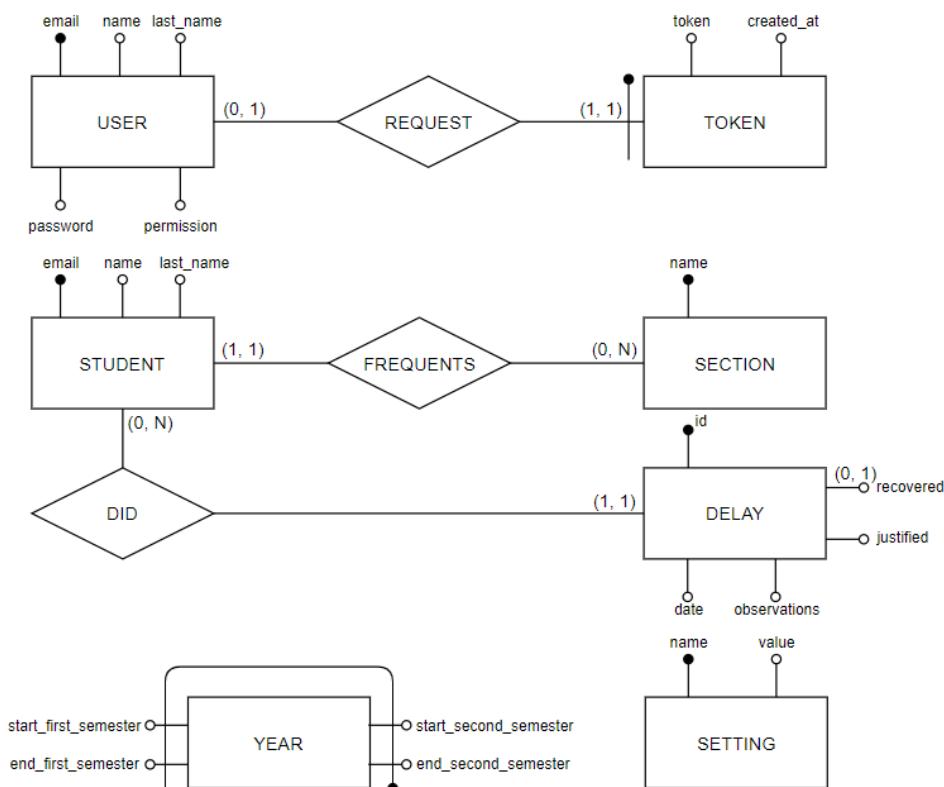


Figura 1 Schema ER database.

Terminato il diagramma ER ho documentato ogni singola tabella con le relative descrizioni degli attributi. Ho anche creato un piccolo schema semplificato che descrive la struttura architetturale che verrà utilizzata nello sviluppo del progetto.

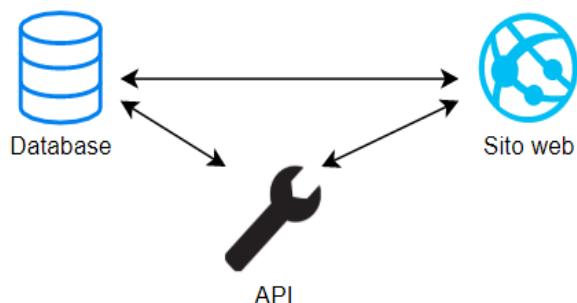
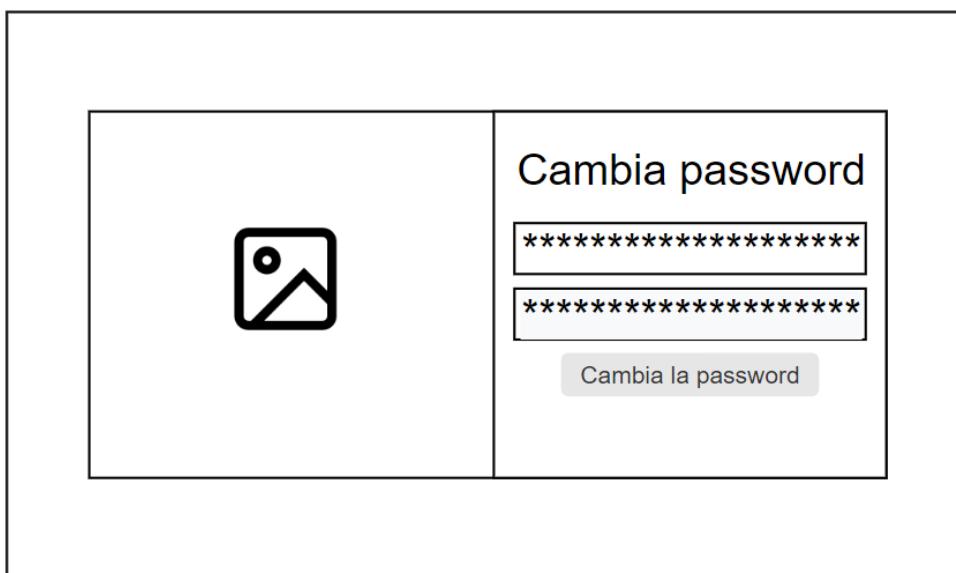


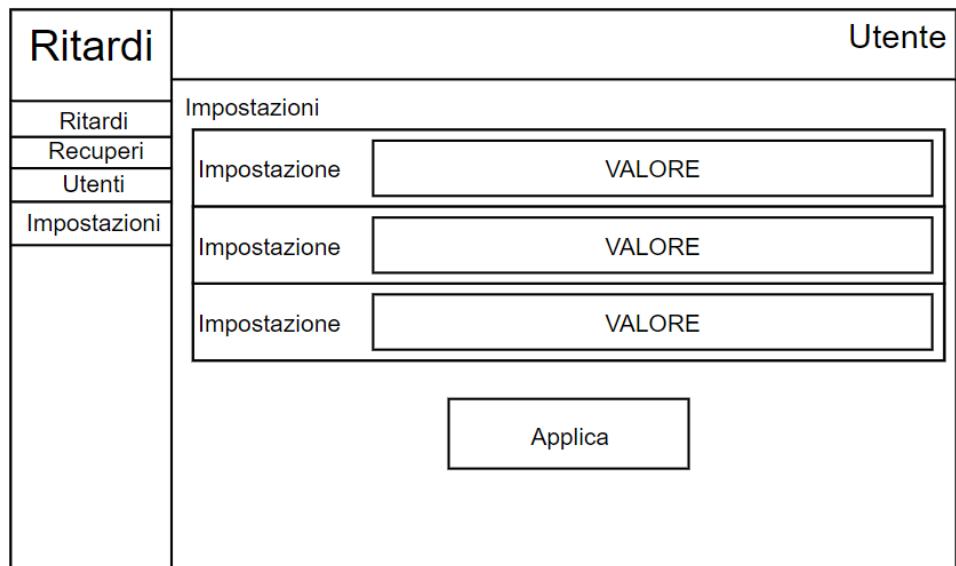
Figura 2 Schema semplificato.

Successivamente mi sono occupato della creazione dei vari mockup per le interfacce grafiche. Di seguito i mockup creati:



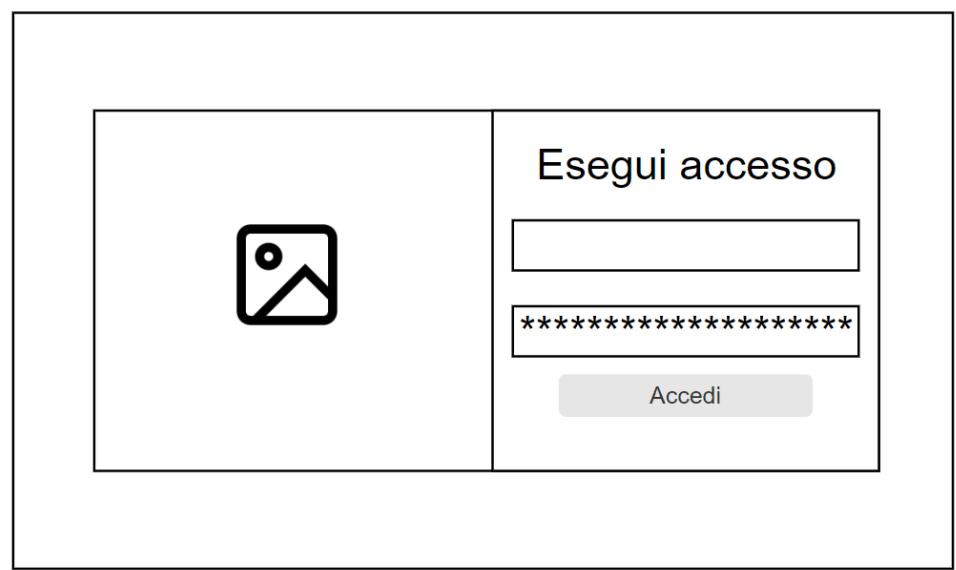
A wireframe mockup of a password change interface. On the left is a large empty rectangular area. On the right, there is a vertical column. At the top of the column is the text "Cambia password". Below this are two input fields, each containing a series of asterisks (***) representing a password. At the bottom of the column is a grey rectangular button labeled "Cambia la password".

Figura 3 Mockup cambio password.



A wireframe mockup of a settings page. On the left is a sidebar with the following menu items: Ritardi (selected), Recuperi, Utenti, and Impostazioni. The main content area is titled "Utente" at the top. Below this is a section titled "Impostazioni" which contains three rows. Each row has a label "Impostazione" followed by a blank rectangular input field and the word "VALORE" to its right. At the bottom of the content area is a grey rectangular button labeled "Applica".

Figura 4 Mockup pagina impostazioni.



A wireframe mockup of a login interface. On the left is a large empty rectangular area. On the right, there is a vertical column. At the top of the column is the text "Esegui accesso". Below this are two input fields: the top one is empty and the bottom one contains a series of asterisks (***) representing a password. At the bottom of the column is a grey rectangular button labeled "Accedi".

Figura 5 Mockup pagina accesso.

Ritardi	Utente
Ritardi	
Recuperi	
Utenti	
Impostazioni	

Figura 6 Pagina di gestione recuperi.

Figura 7 Pagina di recupero password.

Ritardi	Utente
Ritardi	
Recuperi	
Utenti	
Impostazioni	

Figura 8 Pagina di aggiunta ritardi.

Ritardi	Utente	
	Utenti	
	Nome Cognome	Modifica Elimina
	Nome Cognome	Modifica Elimina
	Nome Cognome	Modifica Elimina
Nome Cognome	Modifica Elimina	
Aggiungi utente		

Figura 9 Pagina di gestione utenti.

Problemi riscontrati e soluzioni adottate
Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione
Mi trovo in linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro
Iniziare l'implementazione configurando la base REST.

Diario di lavoro

Luogo	Balerna
Data	13.05.2020

Lavori svolti

Nella giornata di oggi mi sono occupato di completare i seguenti capitoli della documentazione:

- Implementazione
 - o Gestione versioni
 - o Gestore di pacchetti
 - o Database
 - Account di accesso
 - Implementazione banca dati
 - Interrogazione database
 - o Applicativo web
 - Struttura
 - Model View Controller (MVC)
 - Representational State Transfer (REST)
 - Routing delle richieste
 - Configurazione

Come da pianifica mi sono occupato di configurare il framework php-rest. Ho dunque creato un progetto attraverso la creazione di un file “composer.json” nel quale ho specificato le dipendenze del progetto.

```
{  
    "name": "filippo/ritardi-web",  
    "description": "Gestione Web Ritardi Allievi SAMT",  
    "authors": [  
        {  
            "name": "Filippo Finke",  
            "email": "filippo.finke@samtrevano.ch"  
        }  
    ],  
    "minimum-stability": "dev",  
    "autoload": {  
        "psr-4": {  
            "FilippoFinke\\": "src/"  
        }  
    },  
    "require": {  
        "filippofinke/php-rest": "dev-master"  
    }  
}
```

Una volta creato il file composer ho creato la struttura di cartelle del progetto:

```
└── assets
└── src
    ├── Controllers
    ├── Libs
    ├── Middlewares
    ├── Models
    └── Views
└── vendor
```

Successivamente ho creato il file .htaccess che verrà utilizzato per il routing delle richieste:

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.+)\$ index.php [QSA,L]
```

Ho creato inoltre un file di configurazione che verrà utilizzato per ricavare le credenziali di connessione al database di esempio chiamato "config.sample.php":

```
<?php
/**
 * Config.php
 * File nel quale risiede la configurazione del progetto.
 *
 * @author Filippo Finke
 */
/**
 * Configurazioni riguardanti la banca dati.
 */
// Indirizzo di connessione del database.
define("DB_HOST", "127.0.0.1");
// Il nome del database.
define("DB_NAME", "ritardi");
// L'username dell'utente da utilizzare per collegarsi al database.
define("DB_USERNAME", "USERNAME");
// La password da utilizzare per il collegamento al database.
define("DB_PASSWORD", "PASSWORD");
```

Una volta creato il file di configurazione ho creato due viste di prova che ho inserito nella cartella "Views". La prima vista chiamata "test.php" la quale contiene solo un messaggio di test e la seconda chiamata "error.php" la quale verrà caricata in caso di errori. Per caricare la vista di test ho dunque creato un controller anch'esso di prova, il codice è il seguente:

```
<?php
namespace FilippoFinke\Controllers;
use FilippoFinke\Request;
use FilippoFinke\Response;
class Test {
    public static function index(Request $req, Response $res) {
        return $res->render(__DIR__ . '/../Views/test.php');
    }
}
```

Il codice in questione carica semplicemente la vista "test.php". In fine ho creato il file principale del progetto nel quale verrà eseguito il routing delle richieste, ovvero "index.php". Essendo che utilizzerò il server web di PHP per testing ho aggiunto come prima cosa un controllo per i file statici presenti nel progetto:

```
if (php_sapi_name() == 'cli-server') {
    $path = $_SERVER["REQUEST_URI"];
    if (strpos($path, '/assets/') !== false) {
        return false;
    }
}
```

In questo modo i file statici verranno serviti dal web server senza passare dal router di php-rest. Successivamente ho aggiunto un controllo per la presenza del file di configurazione:

```
// Controllo del file di configurazione.
if (!file_exists("config.php")) {
    $response = new Response();
    $info = array(
        "title" => "File di configurazione mancante!",
        "message" => "Impossibile caricare il file di configurazione (config.php)!<br><br>Se stai installando l'applicativo puoi trovare un file di configurazione di esempio chiamato config.sample.php che puoi rinominare config.php"
    );
    $response->render(__DIR__ . "/src/Views/Error/error.php", $info);
    exit;
}
require __DIR__ . '/config.php';
```

In questo modo se il file di configurazione non è presente verrà mostrata la vista "error.php" alla quale verranno passate le informazioni presenti nell'array \$info.

Dopo il controllo del file di configurazione ho creato la connessione al database con i dati ricavati da esso:

```
// Imposto indirizzo del server MySQL.
```

```
Database::setHost(DB_HOST);
// Imposto del database da utilizzare.
Database::setDatabase(DB_NAME);
// Imposto del nome utente per accedere al database.
Database::setUsername(DB_USERNAME);
// Imposto della password per accedere al database.
Database::setPassword(DB_PASSWORD);
// Controllo connessione alla banca dati.
try {
    Database::getConnection();
} catch (PDOException $e) {
    $response = new Response();
    $info = array(
        "title" => "Impossibile stabilire la connessione con il database!",
        "message" => "Errore: " . $e->getMessage()
    );
    $response->render(__DIR__ . "/src/Views/Error/error.php", $info);
    exit;
}
```

Ed in fine ho registrato dei percorsi di prova:

```
$router = new Router();
$router->setNotFound(function (Request $req, Response $res) {
    return $res->redirect("/");
});
$router->get("/", "FilippoFinke\Controllers\Test::index");
$router->start();
```

Ho inoltre aggiunto un campo “type” alla tabella “SETTING” all’interno dello schema ER che verrà utilizzato per verificare il tipo di dato dell’impostazione.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato nessun problema nella giornata di oggi.

Punto della situazione rispetto alla pianificazione

Essendo che ho terminato la configurazione della base REST e lo sviluppo della banca dati mi trovo due giorni in anticipo rispetto alla pianifica iniziale.

Programma di massima per la prossima giornata di lavoro

Iniziare lo sviluppo della gestione degli accessi all’interno del sito web.

Diario di lavoro

Luogo	Balerna
Data	14.05.2020

Lavori svolti

Ho riscontrato diversi problemi con la macchina che ho sto utilizzando per l'esame ma sono comunque riuscito a lavorare. Nella giornata di oggi mi sono occupato di implementare la gestione degli accessi interrogando il database MySQL. Il controllo dello stato di un utente verrà eseguito attraverso le sessioni di PHP. Per questo ho implementato una classe chiamata Session la quale mi permetterà di gestire le sessioni. Il metodo utilizzato per autenticare un utente è il seguente:

```
public static function isAuthenticated() {
    return (isset($_SESSION["authenticated"])) && $_SESSION["authenticated"] == true);
}
```

Successivamente ho creato una classe per semplificare la gestione dei permessi chiamata Permission nella quale ho salvato in delle costanti i valori dei permessi:

```
const INSERT = 1;
const SELECT = 2;
const CREATE = 4;
const ADMINISTRATOR = 8;
```

E successivamente ho implementato un metodo per ogni permesso, ad esempio canInsert:

```
public static function canInsert($permission = null) {
    if ($permission === null) {
        $permission = $_SESSION["permission"];
    }
    return ($permission & self::INSERT) === self::INSERT || self::isAdmin($permission);
}
```

In questo modo potrò verificare i permessi di ogni utente sia attraverso la sessione che tramite parametro del metodo. Ho implementato anche dei validatori di base nella classe Validator, come ad esempio per verificare che un nome sia valido:

```
public static function isValidName($name) {
    return preg_match('/^([A-Za-zÀ-ӽ]+){1,20}$/', $name);
}
```

Ho implementato validazione per nome, cognome, password, email e numeri. Le classi implementate in precedenza le ho utilizzate all'interno dei middleware AdminRequired e AuthRequired i quali verranno utilizzati per dare delle condizioni di accesso ai vari percorsi dell'applicativo. Ad esempio il codice di AuthRequired utilizza la classe Session per controllare se l'utente ha eseguito l'accesso:

```

class AuthRequired {
    public function __invoke($request, $response) {
        if (!Session::isAuthenticated()) {
            $response->redirect("/login");
            exit;
        }
    }
}

```

In questo modo mi basterà assegnare queste classi ai percorsi per determinare i permessi di accesso. Successivamente ho creato una classe Model la quale verrà utilizzata per verificare la presenza degli utenti all'interno della banca dati, ho quindi implementato diversi metodi tra cui:

```

public static function getByEmail($email) {
    $pdo = Database::getConnection();
    $query = "SELECT * FROM user WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':email', $email);
    try {
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}

```

Questo metodo verrà utilizzato dal login per verificare la presenza di un utente. Ho dunque creato anche un controller chiamato Auth il qual andrà a gestire tutti i percorsi e le richieste relative agli accessi nel quale ho implementato metodi per caricare delle viste di prova, come ad esempio quella di login:

```

public static function login(Request $req, Response $res) {
    return $res->render(__DIR__ . '/../Views/Auth/login.php');
}

```

Successivamente ho implementato anche il metodo per eseguire il login:

```

public static function doLogin(Request $req, Response $res) {
    $email = $req->getParam("email");
    $password = $req->getParam("password");
    if (isset($email) && Validator::isValidEmail($email) && isset($password)) {
        $user = Users::getByEmail($email);
        if ($user && password_verify($password, $user["password"])) {
            unset($user["password"]);
            Session::authenticate($user);
            return $res->withStatus(200);
        }
    }
    return $res->withStatus(403);
}

```

Il login verrà verificato attraverso gli status code della risposta. Come ultima cosa ho aggiunto dei percorsi al router del progetto:

```
// Percorso pagina di accesso.  
$router->get("/login", "FilippoFinke\Controllers\Auth::login");  
// Percorso pagina di recupero password.  
$router->get("/forgot-password", "FilippoFinke\Controllers\Auth::forgotPassword");  
// Percorso per eseguire il login.  
$router->post("/login", "FilippoFinke\Controllers\Auth::doLogin");  
// Percorso per eseguire la disconnessione.  
$router->get("/logout", "FilippoFinke\Controllers\Auth::doLogout");  
// Gruppo di percorsi dove è richiesto solamente l'accesso.  
$homeRoutes = new RouteGroup();  
$homeRoutes->add(  
    // Percorso pagina principale.  
    $router->get("/", function($req, $res) {  
        return $res->withHtml("<a href='/logout'>Esci</a>");  
    })  
)  
// Controllo autenticazione.  
->before(new AuthRequired());
```

In questo modo ho potuto eseguire dei test sul funzionamento del login.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato nella giornata di oggi.

Punto della situazione rispetto alla pianificazione

Al momento mi trovo in linea con la pianificazione essendo che ho terminato la gestione dell'accesso all'applicativo web.

Programma di massima per la prossima giornata di lavoro

Sviluppare la funzionalità di recupero password ed iniziare a sviluppare la gestione utenti.

Diario di lavoro

Luogo	Balerna
Data	15.05.2020

Lavori svolti

Nella giornata di oggi mi sono occupato di aggiornare la documentazione ed aggiungere i seguenti capitoli:

- Autenticazione
 - o Gestione delle sessioni
 - o Gestione dei permessi
 - o Gestione percorsi e accessi
- Invio di e-mail

Come pianificato mi sono occupato di implementare la funzionalità di recupero password ed ho iniziato la gestione utenti. Per l'implementazione della funzionalità di recupero password come prima cosa ho creato una classe Mail prendendo spunto da un vecchio progetto la quale mi permetterà di inviare messaggi di posta elettronica in un modo semplificato. Il metodo principale della classe Mail è send il quale permette di inviare e-mail:

```
public static function send($to, $subject, $message)
{
    $eol = "\r\n";

    // header principale
    $headers = "From: <" . self::$fromEmail . ">" . $eol;
    $headers .= "Content-Type: text/html; charset=UTF-8" . $eol;

    return mail($to, $subject, $message, $headers);
}
```

Successivamente ho creato una classe Model chiamata Users che andrà a gestire l'interrogazione con il database per quanto riguarda la tabella user. In questa classe sono presenti svariati metodi i quali permettono di ricavare informazioni da parte della tabella user, ad esempio controllare se un utente esiste attraverso la sua e-mail:

```
public static function getByEmail($email)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM user WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':email', $email);
    try {
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

```
}
```

Sono inoltre presenti metodi per inserimento, aggiornamento ed eliminazione. Per implementare il recupero e cambio della password andrò ad utilizzare dei token salvati all'interno del database. Per questi token ho creato dunque una classe Model dedicata chiamata Tokens. Questa classe contiene tutti i metodi utilizzati per la generazione, invio ed utilizzo di un token. Ad esempio il metodo di invio per il recupero password è il seguente:

```
public static function sendResetPasswordToken($email) {
    if (Users::getByEmail($email)) {
        self::deleteToken($email);
        $token = bin2hex(random_bytes(20));
        $hash = hash("sha256", $token);
        try {
            $pdo = Database::getConnection();
            $query = "INSERT INTO token(email, token) VALUE(:email, :token)";
            $stm = $pdo->prepare($query);
            $stm->bindParam('email', $email);
            $stm->bindParam('token', $hash);
            $link = "http://" . $_SERVER['SERVER_NAME'] . "/login/$token";
            $content = "Contenuto";
            return $stm->execute() && Mail::send($email, "Titolo", $content);
        } catch (\PDOException $e) {
        }
    }
    return true;
}
```

Il codice in questione si occupa di recuperare l'utente corrente, eliminare tutti i token se presenti, creare un token in modo randomico, lo salva nel database ed invia per email. La parte importante di questo codice è il fatto che all'interno del database viene salvato solamente una hash del token in modo tale che se un utente avesse accesso in lettura al database non abbia i mezzi per resettare le password di altri utenti richiedendo un semplice recupero password. All'utente verrà invece inviato il token originale, il concetto è simile a ciò che si usa per salvare e validare le password di accesso. Un altro metodo importante di questa classe è il metodo useToken il quale esegue i controlli sul token passato dall'utente e lo utilizza.

```
public static function useToken($token)
{
    $token = hash("sha256", $token);
    $pdo = Database::getConnection();
    $query = "SELECT email FROM token WHERE token = :token AND CURRENT_TIMESTAMP() - created_at <= " . (self::EXPIRE_AFTER * 60);
    try {
        $stm = $pdo->prepare($query);
        $stm->bindParam('token', $token);
        $stm->execute();
        $email = $stm->fetchColumn();
        if ($email) {
            $stm = $pdo->prepare("DELETE FROM token WHERE token = :token");
            $stm->bindParam('token', $token);
        }
    }
}
```

```
        $stm->execute();
        return $email;
    }
} catch (\PDOException $e) {
}
return false;
}
```

Il metodo in questione si occupa di controllare la presenza del token nel database, di eliminarlo e di ritornare l'utente assegnato ad esso in modo da poter eseguire future azioni su quell'utente. Questa funzionalità viene utilizzata per il cambio password nella classe Model Users:

```
public static function changePassword($token, $password)
{
    if (Validator::isValidPassword($password)) {
        $email = Tokens::useToken($token);
        if ($email) {
            $pdo = Database::getConnection();
            $query = "UPDATE user SET password = :password WHERE email = :email";
            $stmt = $pdo->prepare($query);
            $stmt->bindParam(':email', $email);
            $stmt->bindParam(':password', password_hash($password, PASSWORD_DEFAULT));
            try {
                $_SESSION["login_email"] = $email;
                return $stmt->execute();
            } catch (\PDOException $e) {
            }
        }
    }
    return false;
}
```

Il metodo in questione viene utilizzato per eseguire il cambio password sfruttando un token inviato tramite posta elettronica. Essendo che alla creazione di un utente dovranno essere inviate delle credenziali ad un utente per accedere, verrà inviato un link con un token di attivazione che al primo accesso permetterà il cambio password.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Al momento ho terminato la funzionalità di recupero password e cambio password. Ho dunque un leggero vantaggio rispetto alla pianificazione preventiva.

Programma di massima per la prossima giornata di lavoro

Continuare e terminare la gestione degli utenti del sito web.

Diario di lavoro

Luogo	Balerna
Data	18.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato di aggiornare ed aggiungere i seguenti capitoli della documentazione:

- Validazione dei dati
- Gestione dati ed interrogazione database
- Sicurezza
 - o Interrogazione database
 - o Salvataggio dei dati
 - o Salvataggio credenziali
 - o Recupero password

Inoltre, come stabilito dalla pianifica mi sono occupato di terminare la gestione degli utenti ed inoltre ho terminato anche la gestione delle impostazioni dell'applicativo. Per implementare la gestione degli utenti avendo già la classe Model di gestione mi sono occupato di sviluppare il controller contenente i vari metodi di aggiornamento, inserimento e modifica. Ho dunque creato il controller User il quale contiene i metodi: insert, update e delete. Questi metodi utilizzano la classe Users per interagire con il database, ad esempio l'inserimento di un utente è il seguente:

```
public static function insert(Request $req, Response $res) {
    $name = $req->getParam('name');
    $lastname = $req->getParam('lastname');
    $email = $req->getParam('email');
    try {
        if (
            Validator::isValidName($name)
            && Validator::isValidLastName($lastname)
            && Validator::isValidEmail($email)
            && Users::insert($email, $name, $lastname)
        ) {
            return $res->withStatus(200);
        }
    } catch (Exception $e) {
        return $res->withStatus(500)->withText($e->getMessage());
    }
    return $res->withStatus(400);
}
```

Il metodo ricava i parametri che vengono passati con la richiesta, controlla la validità dei dati e prova ad inserire l'utente. In caso di successo la pagina ritornerà il codice 200, in caso di errore il codice 500 mentre se mancano alcuni dati il codice 400.

Successivamente ho creato la classe Model per la gestione delle sezioni, questa classe è molto simile alle altre classi Model. Contiene metodi che permettono di gestire i dati delle sezioni, come ad esempio il metodo getAll che permette di ricavare tutte le sezioni presenti:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM section";
    try {
        $stm = $pdo->query($query);
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Come per la gestione utenti ho creato una classe controller per le sezioni. In questo controller sono presenti due metodi: insert e delete. Il metodo insert è molto simile a quello degli utenti:

```
public static function insert(Request $req, Response $res)
{
    $name = $req->getParam('name');
    try {
        if (Validator::isValidSection($name) && Sections::insert($name)) {
            return $res->withStatus(200);
        }
    } catch (Exception $e) {
        return $res->withStatus(500)->withText($e->getMessage());
    }
    return $res->withStatus(400);
}
```

Il metodo ricava i parametri che vengono passati con la richiesta, controlla la validità dei dati e prova ad inserire la nuova sezione. In caso di successo la pagina ritornerà il codice 200, in caso di errore il codice 500 mentre se mancano alcuni dati il codice 400. Come per la gestione delle sezioni e degli utenti, ho creato una classe Model per la gestione delle impostazioni. La classe Settings permette di: ricavare tutte le impostazioni, ricavare il valore di una impostazione e modificarne il valore. Per eseguire la modifica di un' impostazione il codice è il seguente:

```
public static function update($name, $value)
{
    $setting = self::get($name);

    if ($setting) {
        if (
            call_user_func_array(
                array("FilippoFinke\Libs\Validator", "isValid" . $setting["type"]),
                array($value)
            )
        ) {
    }
```

```
$pdo = Database::getConnection();
$query = "UPDATE setting SET value = :value WHERE name = :name";
try {
    $stm = $pdo->prepare($query);
    $stm->bindParam(':value', $value);
    $stm->bindParam(':name', $name);
    return $stm->execute();
} catch (\PDOException $e) {
    return false;
}
} else {
    throw new Exception("Il valore deve essere di tipo: " . $setting["type"]);
}
} else {
    throw new Exception("Impostazione inesistente");
}
}
```

Quando viene eseguito l'aggiornamento di un'impostazione viene prima controllata la validità del nuovo valore chiamando in modo dinamico la classe validator passandone il tipo. Anche per questa tabella ho dunque creato una classe controller chiamata Setting la quale contiene solamente un metodo che permette di aggiornare il valore di un'impostazione:

```
public static function update(Request $req, Response $res)
{
    $setting = $req->getAttribute('setting');
    $value = $req->getParam('value');
    try {
        if ($value !== null && Settings::update($setting, $value)) {
            return $res->withStatus(200);
        }
    } catch (Exception $e) {
        return $res->withStatus(500)->withText($e->getMessage());
    }
    return $res->withStatus(400);
}
```

Anche in questo caso i codici di stato sono gli stessi a quelli delle altre classi controller. In fine ho implementato la classe Model ed il Controller per la tabella year la quale permetterà di ricavare i semestri. Anche in questo caso i metodi sono simili e sullo stesso concetto delle altre classi.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato nessun problema nello sviluppo della parte di gestione di utenti ed impostazioni.

Punto della situazione rispetto alla pianificazione

Avendo terminato la gestione degli utenti e la gestione delle impostazioni al momento ho circa 2 ore di vantaggio rispetto alla pianifica.

Programma di massima per la prossima giornata di lavoro

Iniziare lo sviluppo del sistema di gestione dei ritardi.

Diario di lavoro

Luogo	Balerna
Data	19.05.2020

Lavori svolti

Nella giornata di oggi, come consigliato dal perito, mi sono occupato principalmente di sviluppo quindi non ho eseguito nessun aggiornamento alla documentazione. Ho sviluppato i middleware rimanenti per ogni permesso presente nel sistema. Ad esempio il middleware InsertRequired:

```
class InsertRequired
{
    public function __invoke($request, $response)
    {
        if (!Permission::canInsert()) {
            $response->withStatus(401);
            exit;
        }
    }
}
```

Ho creato anche i middlewares chiamati SelectRequired e CreateRequired. Queste classi verranno utilizzate per determinare gli accessi ai vari percorsi. Inoltre, per la gestione dei ritardi ho dovuto creare due ulteriori classe Model una per la tabella delay ed una per la tabella student. All'interno della classe Model Students sono presenti metodi per l'inserimento e ricerca di studenti. Ad esempio il metodo che permette di ricavare uno studente dalla sua email è il seguente:

```
public static function getByEmail($email)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM student WHERE email = :email";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindParam(':email', $email);
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Mentre nella classe di gestione per la tabella delay sono presenti tutti i metodi per gestire i ritardi, come ad esempio un metodo per ricavare tutti i ritardi in un semestre di un determinato studente in base alla sua e-mail:

```
public static function getInCurrentSemesterByEmail($email)
{
    $semester = Years::getCurrentSemester();
    $start = $semester[0];
    $end = $semester[1];

    $pdo = Database::getConnection();
    $query = "SELECT id, DATE_FORMAT(date, '%d.%m.%Y') as 'date', observations,
              DATE_FORMAT(recovered, '%d.%m.%Y') as 'recovered', justified FROM
              delay WHERE email = :email AND date BETWEEN :start AND :end";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindParam(':email', $email);
        $stm->bindParam(':start', $start);
        $stm->bindParam(':end', $end);
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Per entrambe le classi Modal ho creato i rispettivi Controller chiamati Delay e Student. Successivamente mi sono occupato di aggiungere come dipendenza la libreria fpdf all'interno del file composer.json:

```
"fpdf/fpdf": "dev-master"
```

In questo modo sarà possibile utilizzare questa libreria esterna per la creazione dei PDF. Per generare i PDF ho dunque creato una classe che estende FPDF la quale permetterà di generare in modo automatico i file. All'interno della classe sono presenti due metodi per creare l'header ed il footer del pdf:

```
public function Header()
{
    $this->SetFont('Arial', 'B', 12);
    $this->Cell(($this->GetPageWidth() - 20), 7, 'GESTIONE RITARDI', 'B', 0, 'T');
    $this->Ln(12);
}
```

L'header ed il footer contengono codice simile che permettono di impostare le varie pagine. All'interno del costruttore viene generato il file PDF.

Il risultato del PDF è il seguente:

GESTIONE RITARDI			
Ritardi di Bryan Beffa dal 17.01.2020 al 25.06.2020.			
Ritardi nel semestre: 3, da recuperare: 0.			
Data	Recupero	Giustificato	Osservazioni
02.02.2020	No	No	Moto
02.02.2020	No	No	Sveglia
02.02.2020	02.02.2020	No	asd

19.05.2020

Pagina 1 di 1

Figura 1 PDF generato.

Ho creato anche un ulteriore controller chiamato Home il quale si occuperà di caricare le viste del sito web che al momento non sono ancora state sviluppate. Ad esempio per caricare la pagina di gestione degli utenti il metodo è il seguente:

```
public static function users(Request $req, Response $res)
{
    $users = Users::getAll();
    return $res->render(
        __DIR__ . '/../Views/Home/users.php',
        array("users" => $users)
    );
}
```

Terminato di implementare le classi Model ed i vari Controller ho registrato i nuovi percorsi all'interno del file index.php:

```
// Percorso creazione studente.
$router->post("/student", "FilippoFinke\Controllers\Student::insert")
->before(new InsertRequired());
// Percorso per creare un pdf dei ritardi.
$router->get("/student/{email}/pdf", "FilippoFinke\Controllers\Student::pdf")
->before(new CreateRequired());
// Percorso per ricavare i ritardi da recuperare di uno studente.
$router->get("/student/{email}/{type}", "FilippoFinke\Controllers\Student::delays")
```

```
->before(new SelectRequired()),
// Percorso per ricavare i ritardi di uno studente.
$router->get("/student/{email}", "FilippoFinke\Controllers\Student::delays")
->before(new SelectRequired()),
// Percorso di rimozione ritardo.
$router->delete("/delay/{id}", "FilippoFinke\Controllers\Delay::delete")
->before(new InsertRequired()),
// Percorso di aggiornamento ritardo.
$router->post("/delay/{id}", "FilippoFinke\Controllers\Delay::update")
->before(new InsertRequired()),
// Percorso di aggiunta ritardo.
$router->post("/delay", "FilippoFinke\Controllers\Delay::insert")
->before(new InsertRequired())
```

Specificando già i permessi per ogni azione utilizzando le classi create in precedenza.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato durante lo sviluppo della gestione dei ritardi , degli studenti e della creazione del PDF.

Punto della situazione rispetto alla pianificazione

Essendo che ho concluso la gestione dei ritardi e la creazione dei PDF sono in anticipo di un giorno rispetto alla pianifica.

Programma di massima per la prossima giornata di lavoro

Iniziare la creazione dell'interfaccia grafica di login.

Diario di lavoro

Luogo	Balerna
Data	20.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato principalmente dello sviluppo di interfacce grafiche e di aggiornare alcuni capitoli della documentazione. Ho dunque aggiunto i seguenti capitoli:

- Gestione dati ed interrogazione database
 - o Tabella user
 - o Tabella student
 - o Tabella section
 - o Tabella setting

Ho inoltre preparato alcuni capitoli che andranno riempiti.

Mi sono occupato dunque di scaricare ad aggiungere il template SB Admin 2 all'interno della cartella assets. Il template è stato scaricato dal seguente link:

- <https://github.com/BlackrockDigital/startbootstrap-sb-admin-2>

Successivamente mi sono occupato di aggiornare tutti i percorsi dell'applicativo che prima erano assoluti sostituendoli con la variabile BASE presente all'interno del file di configurazione in modo da rendere il sito web dinamico in caso esso sia installato in sottocartelle e non nella cartella principale del web server. Esempio di cambiamento nei middlewares:

```
$response->redirect(BASE . "login");
```

Successivamente ho creato la grafica della pagina di accesso che è la seguente:

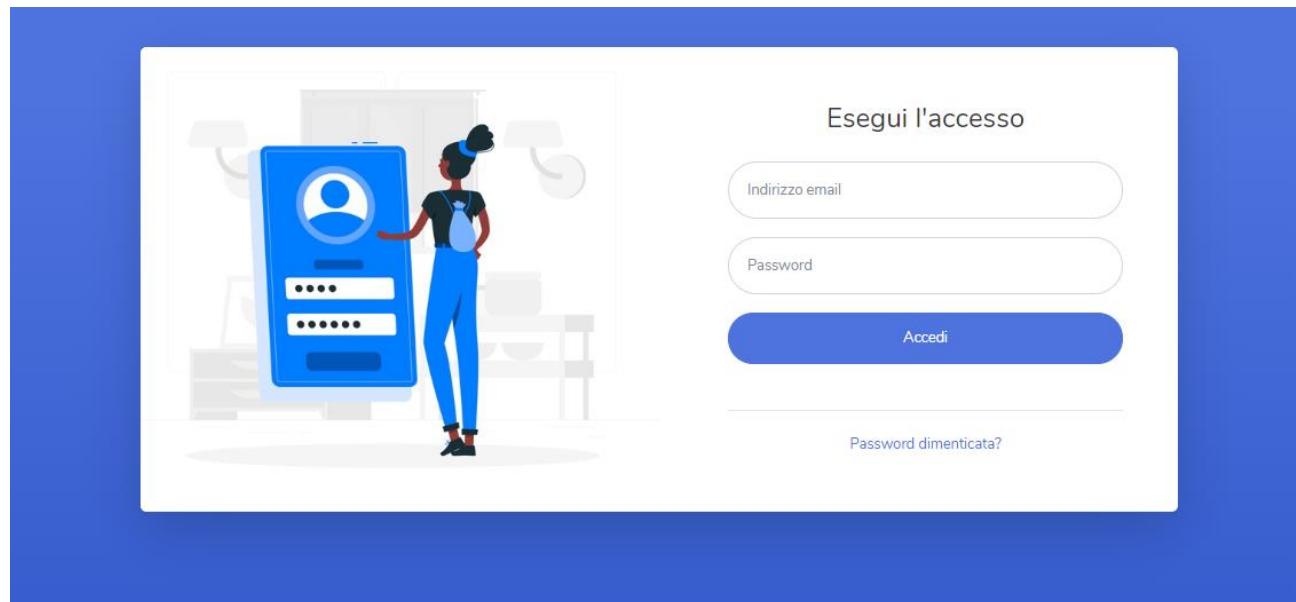


Figura 1 Pagina di accesso.

Ho integrato il backend della pagina con il frontend attraverso delle chiamate AJAX ai vari percorsi (login, etc...).

All'interno della pagina di accesso è anche contenuta la pagina di cambio password, essa cambia dinamicamente in caso sia presente un token nell'url, il risultato è il seguente:

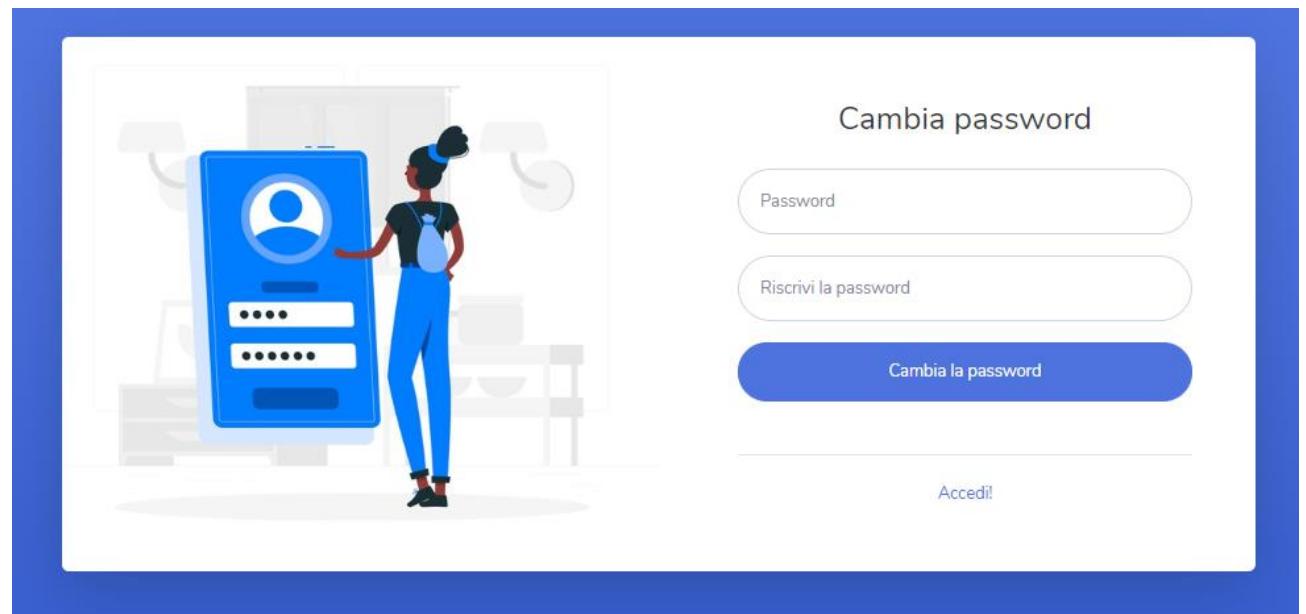


Figura 2 Pagina di accesso, cambio password.

Ho implementato anche la pagina di recupero password:

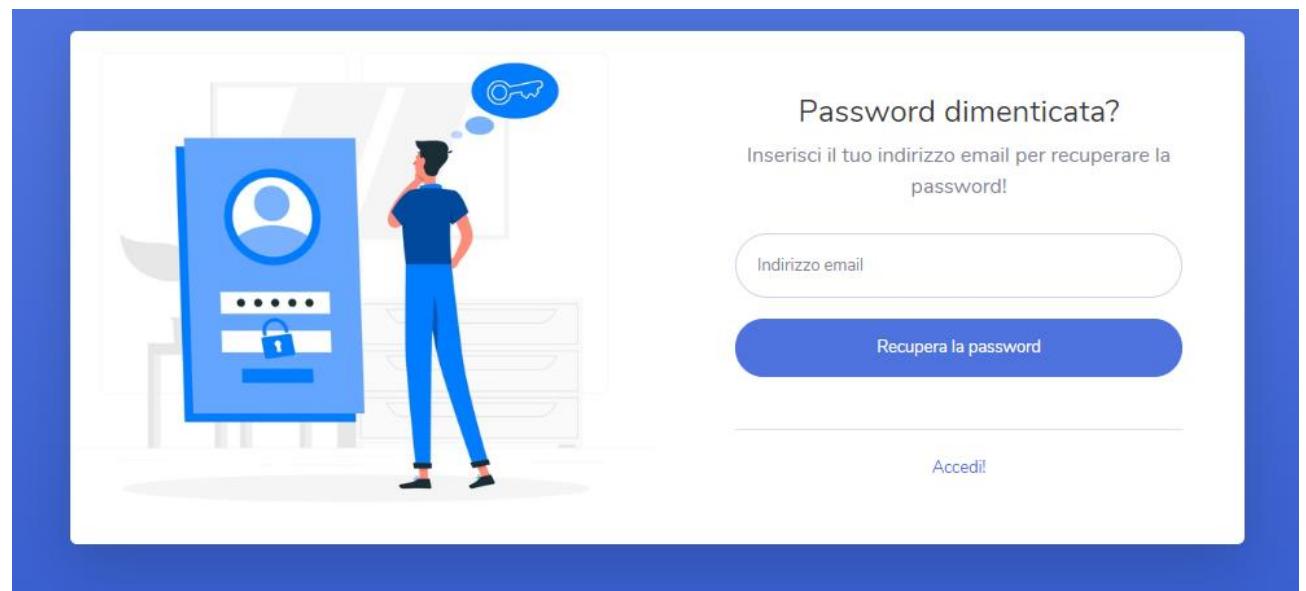


Figura 3 Pagina di recupero password.

Successivamente ho implementato anche la pagina di gestione degli utenti dell'applicativo:

Nome	Cognome	E-mail	Permessi	Azioni	
Admin	Profile	admin@ritardi.ch	<input type="checkbox"/> Inserimento ritardi <input type="checkbox"/> Creazione PDF	<input type="checkbox"/> Visione ritardi <input type="checkbox"/> Amministratore	<button>Elimina</button>
Fabrizio	Valsangiacomo	fabrizio.valsangiacomo@edu.ti.ch	<input checked="" type="checkbox"/> Inserimento ritardi <input checked="" type="checkbox"/> Creazione PDF	<input checked="" type="checkbox"/> Visione ritardi <input type="checkbox"/> Amministratore	<button>Elimina</button>
Fabrizio	Valsangiacomo	fabrizio.valsangiacomo@gmail.com	<input checked="" type="checkbox"/> Inserimento ritardi <input checked="" type="checkbox"/> Creazione PDF	<input checked="" type="checkbox"/> Visione ritardi <input type="checkbox"/> Amministratore	<button>Elimina</button>

Figura 4 Pagina di gestione utenti.

E come ultima pagina ho implementato quella di gestione dei ritardi:

Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni
Bryan	Beffa	SAM I4AA	bryan.beffa@samtrevano.ch	1	0	<button>Visualizza</button> <button>Aggiungi ritardo</button> <button>Crea PDF</button>

Figura 5 Pagina di gestione ritardi.

L'implementazione di tutte queste pagine è stata molto più semplice del previsto in quanto una base della struttura di essere era già messo a disposizione da parte dal template utilizzato (SB Admin 2). Inoltre, grazie all'utilizzo di Bootstrap e jQuery è stato molto semplice eseguire delle modifiche per personalizzare le pagine. Inoltre, essendo il metodo di comunicazione tra front-end e back-end lo stesso in tutte le pagine è stato molto semplice implementare le chiamate AJAX attraverso con JavaScript e adattarle per ogni pagina. Ad esempio, l'unica funzione utilizzata per gestire la richiesta di accesso da parte dell'interfaccia grafica è la seguente:

```
$("#login-form").on("submit", (e) => {
  e.preventDefault();

  let params = $("#login-form").serialize();

  $("#login-btn")
    .prop("disabled", true)
    .html('<div class="spinner-border"></div>');
  $.post("login", params)
    .then(() => {
      location.href = "";
    })
    .catch((err) => {
      $("#error-login").text("Credenziali errate!").show();
    })
})
```

```
.then(() => {
  $("#login-btn").prop("disabled", false).html("Accedi");
});
});
```

Questa funzione viene chiamata all'invio del form di accesso (quando si preme Enter oppure il tasto Accedi). Come prima cosa vengono ricavati i dati da inviare dal form di accesso, successivamente viene disabilitato il bottone di accesso e viene inviata la richiesta al server. In caso di successo l'utente viene mandato alla pagina principale altrimenti viene mostrato un errore. Tutte le chiamate sono state eseguite utilizzando questo approccio, di conseguenza è stato molto veloce sviluppare queste funzioni per ogni pagina senza problemi.

Nella giornata di oggi ho inoltre avuto una chiamata con il formatore dove ho avuto modo di informarlo del problema dell'invio di e-mail da parte dell'hosting che mi è stato fornito ed ha confermato la mia teoria che il problema è esterno al mio codice in quanto questo problema accadeva anche in altri progetti.

Problemi riscontrati e soluzioni adottate

Ho riscontrato un problema nell'invio delle e-mail da parte dell'applicativo web sull'hosting esterno Infomaniak. Quando vengono inviati dei messaggi di posta elettronica essi possono partire anche dopo decine di minuti. Dopo aver eseguito svariati test ho constatato che il problema riguarda la piattaforma e non il codice, di conseguenza non vi è nulla che posso fare per risolvere questo ritardo.

Punto della situazione rispetto alla pianificazione

Grazie all'utilizzo del template SB Admin 2 ho risparmiato molto tempo nella creazione delle interfacce grafiche e di conseguenza sono in anticipo di circa due giorni rispetto alla pianifica preventiva.

Programma di massima per la prossima giornata di lavoro

Nella prossima giornata di lavoro punto a riuscire a terminare tutte le interfacce grafiche in modo da poter mostrare il prodotto e risolvere eventuali piccole modifiche e problemi.

Diario di lavoro

Luogo	Balerna
Data	25.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato di aggiornare i seguenti capitoli della documentazione:

- Interfacce grafiche
 - o Pagina di accesso
 - o Pagina di cambio password
 - o Pagina di recupero password
 - o Pagina di gestione ritardi
 - o Pagina di gestione recuperi
 - o Pagina di gestione utenti
 - o Pagina impostazioni
- Test
 - o Protocollo di test
- Glossario
- Sitografia

Ho inoltre creato un manuale di installazione dell'applicativo web e terminato lo sviluppo delle interfacce grafiche creando la pagina di recupero dei ritardi e la pagina di gestione delle impostazioni. La pagina di gestione dei recuperi è molto simile a quella di gestione dei ritardi in quanto la struttura è la stessa. La pagina di gestione dei recuperi è dunque la seguente:

Nome	Cognome	Sezione	E-mail	Ritardi	Da recuperare	Azioni
Bryan	Beffa	SAM I4AA	bryan.beffa@samtrevano.ch	5	1	<button>Visualizza</button>

Figura 1 Pagina di gestione recuperi.

Attraverso questa pagina è dunque possibile vedere solamente gli studenti che hanno dei ritardi da recuperare. Successivamente ho sviluppato anche la pagina di gestione delle impostazioni dell'applicativo web. Questa pagina contiene tre tabelle le quali permettono di eseguire le modifiche alle impostazioni del sito web. Il risultato della pagina è il seguente:

RITARDI WEB

Impostazioni

Sezioni

Mostra 5 righe Cerca:

Nome	Azioni
SAM I4AA	<button>Elimina</button>

Mostro da 1 a 1 di 1 righe. Precedente 1 Prossima

Anni scolastici

Mostra 5 righe Cerca:

Primo semestre	Secondo semestre	Azioni
01.09.2019 - 25.01.2020	25.01.2020 - 29.06.2020	<button>Elimina</button>

Mostro da 1 a 1 di 1 righe. Precedente 1 Prossima

Configurazione

Cerca:

Impostazione	Valore	Azione
from_email	gestione-ritardi@no-reply.ch	<button>Modifica</button>
max_delays	3	<button>Modifica</button>

Mostro da 1 a 2 di 2 righe.

Figura 2 Pagina di gestione impostazioni.

Ho avuto anche un colloquio con il perito e il committente. In questo colloquio sono saltate fuori delle migliorie / funzionalità da implementare nel progetto che non avevo ritenuto logiche. Ho dunque creato una lista di cose da fare:

- Togliere animazione di attesa nell'invio delle e-mail.
- Risolvere problema ritardi giustificati.
- Aggiustare grandezza celle file PDF.
- Aggiungere un PDF all'interno della pagina dei recuperi.
- Suddividere gli studenti per anni.
- Permettere di selezionare quale anno si vuole vedere.
- Permettere l'eliminazione soft di sezioni (con un flag).

Queste modifiche comporteranno dunque un cambiamento alla struttura del database e alla logica delle pagine implementate fino ad ora.

Problemi riscontrati e soluzioni adottate

Nella giornata di oggi non ho riscontrato nessun problema.

Punto della situazione rispetto alla pianificazione

Essendo che oggi ho terminato lo sviluppo delle interfacce grafiche sono in anticipo di 3 giorni rispetto alla pianifica iniziali.

Programma di massima per la prossima giornata di lavoro

Risolvere i problemi e aggiungere le funzionalità richieste dal formatore.

Diario di lavoro

Luogo	Balerna
Data	26.05.2020

Lavori svolti

Nella giornata di oggi mi sono occupato di aggiungere delle funzionalità richieste dal committente:

- Aggiungere un PDF nella sezione dei recuperi.
- Permettere di eliminare le sezioni mantenendole nel database.
- Suddividere gli studenti per anno scolastico.
- Permettere di selezionare quale anno scolastico da visualizzare.
- Permettere di eliminare gli anni scolastici con i relativi dati (Utenti e ritardi).

Per eseguire le seguenti modifiche ho dunque aggiornato lo schema del database aggiungendo:

- Flag “deleted” alla tabella “section” in modo da poter segnare i dati come eliminati.
- Aggiunti attributi “id” e “year” allo studente in modo da poterlo identificare per anno ed avere più utenti con la stessa e-mail.
- Aggiornata la tabella delay utilizzando il nuovo id dello studente rimpiazzando l’e-mail.

Dopo le modifiche, lo schema ER è dunque il seguente:

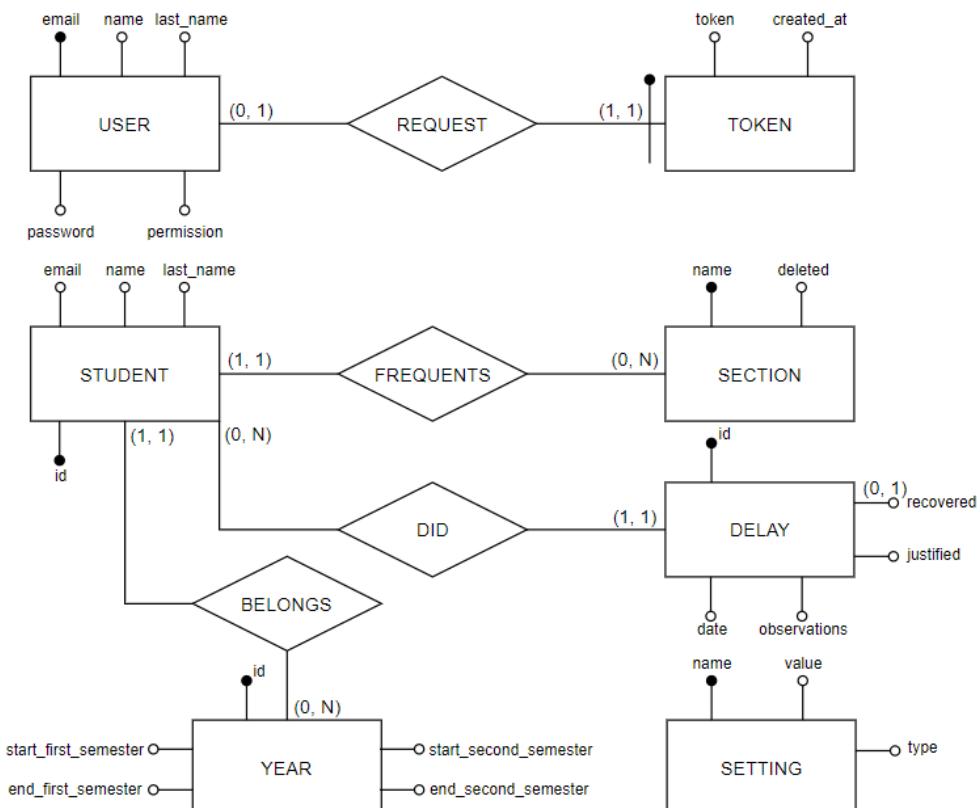


Figura 1 Schema ER aggiornato.

Successivamente ho convertito le classi model dall’utilizzo dei vecchi campi del database all’utilizzo dei nuovi campi. Ho dunque dovuto aggiornare la classe model Students rimpiazzando tutti i metodi che prima utilizzavano l’e-mail come identificativo con il nuovo id auto generato. La stessa cosa è stata fatta anche per

la classe model Sections. In questa classe è stato aggiunto il supporto al nuovo flag in modo da poter eliminare le sezioni mantenendole nel database, ad esempio il metodo delete è stato aggiornato nel seguente modo:

```
public static function delete($name)
{
    $pdo = Database::getConnection();
    $query = "UPDATE section SET deleted = :flag WHERE name = :name";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':flag', 1);
    $stm->bindValue(':name', $name);
    try {
        return $stm->execute();
    } catch (\PDOException $e) {
        return false;
    }
}
```

Al posto che eliminare il dato dal database viene aggiornato il flag “deleted” della sezione. In questo modo il dato verrà considerato eliminato dall’applicativo web.

L’aggiornamento della struttura del database relativa allo studente ha avuto un impatto negativo anche sulla classe model Delays la quale ho dovuto adattare per l’utilizzo dell’identificatore al posto dell’indirizzo di posta elettronica.

Ho eseguito una modifica anche alla classe Year aggiungendo un metodo chiamato getYearById che permette di ricavare un anno dal proprio identificativo, questo perché verrà utilizzato per stabilire gli anni ai quali appartengono gli studenti, il codice è il seguente:

```
public static function getYearById($id)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM year WHERE id = :id";
    try {
        $stm = $pdo->prepare($query);
        $stm->bindValue(":id", $id);
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
}
```

Successivamente mi sono occupato di generalizzare la classe PDF in modo da mantenere solamente i metodi header e footer in modo che altre classi possano estendere la classe PDF senza dover implementare i metodi nuovamente. Ho dunque diviso i PDF personali degli studenti da quelli della sezione di recupero.

GESTIONE RITARDI				
Ritardi di Bryan Beffa dal 25.01.2020 al 29.06.2020.				
Ritardi nel semestre: 3, da recuperare: 1.				
Data	Recupero	Giustificato	Osservazioni	
21.05.2020	No	No	Incidente BUS	
21.05.2020	No	No	Ritardo treni	
20.05.2020	No	No	Causa sveglia	

26.05.2020 Pagina 1 di 1

Figura 2 Esempio PDF studente.

GESTIONE RITARDI				
Studenti che hanno dei ritardi da recuperare.				
Email	Studente	Sezione	Ritardi	Da recuperare
bryan.beffa@samtrevano.ch	Bryan Beffa	SAM I4AA	3	1

26.05.2020	Pagina 1 di 1
Figura 3 Esempio PDF recuperi.	

Terminata l'implementazione dei PDF ho aggiornato tutti i Controllers e le relative viste in modo che possano utilizzare i metodi modificati delle classi Model.

Durante questo refactor del codice ho inoltre aggiunto un aggiornamento dei permessi dell'utente ad ogni richiesta in modo che se un utente viene eliminato / aggiornato le modifiche vengano applicate senza che esso debba disconnettersi.

```
if (Session::isAuthenticated()) {
    $user = Users::getByEmail($_SESSION["email"]);
    if ($user) {
        unset($user["password"]);
        Session::authenticate($user);
    } else {
        Session::logout();
    }
}
```

Ho inoltre trovato la causa del lentissimo invio di posta elettronica attraverso l'hosting esterno, la soluzione si trova nella sezione "Problemi riscontrati e soluzioni adottate".

Successivamente mi sono occupato di aggiornare i seguenti capitoli della documentazione:

- Progettazione
 - o Design dei dati e database
 - Schema ER
 - Descrizioni delle tabelle
 - Schema logico
- Implementazione
 - o Applicativo web
 - Gestione dati ed interrogazione database
 - Tabella user
 - Tabella token
 - Tabella student
 - Tabella section
 - Tabella delay
 - Tabella year
 - Tabella setting
 - Creazione PDF
 - PDF Studente
 - PDF Recuperi
- Test
 - o Protocollo di test
- Glossario

Problemi riscontrati e soluzioni adottate

Eseguendo dei test utilizzando svariati headers per l'invio di posta elettronica sono giunto alla conclusione che il server mail di Infomaniak esegue dei controlli sull'header "From". Utilizzando una e-mail come gestione-ritardi@no-reply.ch il mail server esegue dei controlli e dopo una decina di minuti l'e-mail viene inviata correttamente. Mentre utilizzando una e-mail come ad esempio no-reply@gestione-ritardi-cpt.ch l'e-mail viene inviata senza nessun problema velocemente.

Il controllo da parte dell'hosting viene eseguito, giustamente, per prevenire attacchi di phishing o altre attività illegali attraverso il loro mail server.

Punto della situazione rispetto alla pianificazione

Nonostante l'aggiunta delle nuove funzionalità mi trovo in anticipo di circa un giorno rispetto alla pianifica iniziale.

Programma di massima per la prossima giornata di lavoro

Eseguire i test dell'applicativo e continuare la documentazione.

Diario di lavoro

Luogo	Balerna
Data	27.05.2020

Lavori svolti

Nella giornata di oggi ho aggiornato i seguenti capitoli della documentazione:

- Indice delle figure
- Progettazione
 - Diagrammi delle classi
 - UML Controllers
 - UML Libs
 - UML Middlewares
 - UML Models
- Test
 - Protocollo di test
 - Risultati test
 - Mancanze/limitazioni conosciute
- Glossario
- Sitografia
- Conclusioni
 - Sviluppi futuri
 - Considerazioni personali
- Allegati

Ho inoltre aggiornato il manuale di installazione:

- Indice delle figure

Ho creato inoltre l'abstract che andrà messo come prima pagina della documentazione mettendo i seguenti capitoli:

- Situazione iniziale
- Attuazione
- Risultati

Successivamente mi sono occupato di aggiornare tutti i commenti presenti all'interno del codice sorgente aggiungendo i commenti alle seguenti parti:

- File index.php
- Database
- Classi Controller
- Classi Libs
- Classi Middlewares
- Classi Models
- Viste
- Files JavaScript

Ho aggiornato inoltre il manuale di installazione dell'applicativo.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato nessun problema durante la giornata di oggi.

Punto della situazione rispetto alla pianificazione

Ho due giorni di anticipo rispetto alla pianifica in quanto ho eseguito i test dell'applicativo in modo parziale.

Programma di massima per la prossima giornata di lavoro

Continuare la fase di testing dell'applicativo, creare un database di base contenente dati di prova e creare il diagramma di gantt consuntivo.

Diario di lavoro

Luogo	Balerna
Data	28.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato di creare l'abstract che andrà messo come prima pagina della documentazione, successivamente ho continuato la fase di testing concludendo il capitolo del protocollo di test e i vari risultati. Per concludere ho aggiornato il manuale di installazione aggiungendo delle informazioni aggiuntive sull'installazione dell'applicativo. Successivamente come consigliato dal perito, mi sono occupato di creare un file di test del database contenente diversi dati generati in modo casuale in modo da potere riempire il database per la demo dell'applicativo.

Durante la giornata di oggi ho inoltre provato ad installare l'applicativo su una macchina virtuale seguendo il manuale di installazione per verificare che essa copra tutti i casi.

Problemi riscontrati e soluzioni adottate

Non ho riscontrato problemi essendo che la giornata di oggi è stata principalmente di documentazione.

Punto della situazione rispetto alla pianificazione

Ho un vantaggio di circa quattro ore rispetto alla pianifica iniziale essendo che ho terminato la fase di testing.

Programma di massima per la prossima giornata di lavoro

Rileggere quanto prodotto durante lo sviluppo del progetto e preparare il tutto per la consegna.

Diario di lavoro

Luogo	Balerna
Data	29.05.2020

Lavori svolti

Durante la giornata di oggi mi sono occupato di revisionare quanto fatto fino ad ora, di creare i vari file PDF, controllare la struttura del progetto ed aggiornare l'ultima versione dell'applicativo sull'hosting esterno.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione essendo il progetto pronto per la consegna.

Programma di massima per la prossima giornata di lavoro

-

Quaderno dei compiti

1 INFORMAZIONI GENERALI

Candidato	Nome: Filippo	Cognome: Finke
	filippo.finke@samtrevano.ch	+41 76 355 95 97
Luogo di lavoro	Scuola Arti e Mestieri / CPT Trevano-Canobbio	
Orientamento	<input type="checkbox"/> 88601 Sviluppo di applicazioni <input checked="" type="checkbox"/> 88602 Informatica aziendale <input type="checkbox"/> 88603 Tecnica dei sistemi	
Superiore professionale	Nome: Fabrizio	Cognome: Valsangiacomo
	fabrizio.valsangiacomo@edu.ti.ch	
Perito 1	Nome: Gianluca	Cognome: Costante
	gianluca.costante@gmail.com	
Perito 2	Nome: Roberto	Cognome: Guidi
	roberto.guidi.86@gmail.com	
Periodo	Dal 11.05.2020 al 29.05.2020	
Orario di lavoro	Secondo orari della convocazione della DFP	
Numero di ore	80	
Pianificazione 80h (in ore o %)	Analisi: 10%	
	Implementazione: 50%	
	Test: 10%	
	Documentazione: 30%	

2 PROCEDURA

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

3 TITOLO

Gestione dei ritardi degli allievi SAMT tramite applicativo via web

4 HARDWARE E SOFTWARE DISPONIBILE

1 PC preparato dall'allievo prima dell'inizio del LPI.

1 Accesso presso l'hosting messo a disposizione dalla scuola per caricare il progetto.

5 PREREQUISITI

.....

6 DESCRIZIONE DEL PROGETTO

Requisiti:

- bisogna prevedere un amministratore che possa accedere al sito in modo completo ed inoltre possa creare, cancellare, modificare dei campi, direttamente dalla pagina web senza dovere andare in MySql;
- predisporre una pagina di login per la gestione delle persone autorizzate ad utilizzare l'applicativo;
- bisogna che ci sia sempre un amministratore attivo e non si può eliminare se stesso;
- l'amministratore deve potere creare e attivare ulteriori amministratori e utenti o eliminare quelli non più necessari. La registrazione avviene inserendo il nome, il cognome, l'account email come username. La password viene creata in automatico in modo random, la quale dovrà essere spedita automaticamente all'utente via email. Al login dovrà esserci un sistema che fa cambiare la password provvisoria. Bisogna anche prevedere la possibilità di richiedere una nuova password in caso di perdita;
- l'amministratore deve potere fare diventare in qualsiasi momento un utente amministratore o utente con solo diritti limitati (inserimento dei ritardi, visione dei dati e creazione dei pdf). Inoltre dovrà gestire gli accessi e cancellare gli utenti. Deve esserci sempre almeno un amministratore;
- non deve essere creato un account email esterno tipo gmail o altro, ma deve essere gestito dal hosting;
- deve essere creata una pagina di amministrazione nella quale vengano inserite dall'amministratore la data di inizio e fine di ogni semestre con il sistema del calendario, i nomi delle sezioni (es. SAM I1AA, SAM I1AB, ecc....), gli anni scolastici (es. 2019-2020, 2020-2021, ecc...) e la scelta del numero di ritardi per iniziare i recuperi (default è 3 compreso). In seguito ogni ritardo richiede un recupero e all'inizio del secondo semestre, il conteggio riparte da zero, però i ritardi del 1 semestre vanno recuperati fino al loro azzeramento;
- un'altra pagina deve essere creata per fare in modo che i docenti di classe registrati possano inserire i nominativi degli allievi che hanno cominciato ad accumulare i ritardi (campi: Cognome, Nome, Email SAMT, Sezione, Data ritardo, Osservazioni). Se un allievo è già registrato con dei ritardi deve essere possibile aggiungere un ulteriore ritardo con la data da scegliere tramite un calendario. Inoltre il loro numero deve figurare visivamente, ed una pagina di gestione deve permettere di inserire la data per la quale un allievo ha fatto il recupero e rendere visibile ancora il numero di altri recuperi registrati. Deve anche essere implementata un'opzione (tipo check) per fare in modo che un ritardo sia visibile ma non conteggiato. Bisogna anche tenere un istoriato degli anni precedenti;
- per ogni ritardo per il quale ci deve essere un recupero, bisogna implementare l'invio di una email informativa all'allievo con la data del ritardo e una descrizione tipo: Riceverai una ulteriore email per la pianificazione del recupero del ritardo;

- bisogna implementare dei sistemi di ricerca (Cerca) per facilitare la ricerca di informazioni;
 - in ogni momento deve potere essere creato un report in pdf con i nominativi degli allievi, il numero di ritardi da recuperare e quelli già eseguiti con la data;
 - durante lo svolgimento del progetto verrà messo a disposizione un hosting “interno”, sul quale deve essere caricato il gestionale.
-

7 RISULTATI FINALI

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

- Una pianificazione iniziale (entro il primo giorno)
 - Una documentazione del progetto
 - Un diario di lavoro
 - Implementazione LPI
-

8 PUNTI TECNICI SPECIFICI VALUTATI

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

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

1. 135 – Documentazione DB, tabelle, ecc...
2. 237 – Analisi della sicurezza (Applicazione Web)
3. 240 – Sicurezza di base di dati
4. 148 – Solidità verifica dei dati, intercettazione degli errori di inserimento
5. 193 – Design del GUI
6. 254 – Responsive Web Design
7. 232 – Programmazione web professionale

9 FIRMA

Candidato

(luogo e data)

Superiore professionale

(luogo e data)

Perito 1

(luogo e data)

Perito 2

(luogo e data)