

## **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

<b>1</b>	<b>Introduzione.....</b>	<b>5</b>
1.1	Informazioni sul progetto.....	5
1.2	Abstract.....	5
1.3	Scopo.....	5
<b>2</b>	<b>Analisi.....</b>	<b>6</b>
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
<b>3</b>	<b>Progettazione.....</b>	<b>13</b>
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
<b>4</b>	<b>Implementazione.....</b>	<b>21</b>
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
<b>5</b>	<b>Test .....</b>	<b>53</b>
5.1	Protocollo di test.....	53
5.2	Risultati test.....	60
5.3	Mancanze/limitazioni conosciute.....	67
<b>6</b>	<b>Consuntivo.....</b>	<b>67</b>
<b>7</b>	<b>Conclusioni.....</b>	<b>69</b>
7.1	Sviluppi futuri.....	69
7.2	Considerazioni personali.....	69
<b>8</b>	<b>Glossario .....</b>	<b>69</b>
<b>9</b>	<b>Sitografia .....</b>	<b>70</b>
<b>10</b>	<b>Allegati.....</b>	<b>70</b>

## 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](mailto:filippo.finke@samtrevano.ch)

**Classe:** I4AC Scuola Arti e Mestieri Trevano, Informatica

**Formatore:** Fabrizio Valsangiacomo, [fabrizio.valsangiacomo@edu.ti.ch](mailto:fabrizio.valsangiacomo@edu.ti.ch)

**Perito:** Gianluca Costante, [gianluca.costante@gmail.com](mailto: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	
<b>Nome</b>	Piattaforma dell'applicativo
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Si necessita di un applicativo web.
Sotto requisiti	
<b>001</b>	Il sito deve funzionare su browser moderni (Esempio: Chrome).

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

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

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

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

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

ID: REQ-006	
<b>Nome</b>	Pagina di gestione recuperi
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Si necessita di una pagina che permetta di aggiungere i ritardi recuperati da parte degli studenti.
Sotto requisiti	
<b>001</b>	Dovrà essere possibile inserire il ritardo recuperato da parte dello studente.
<b>002</b>	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

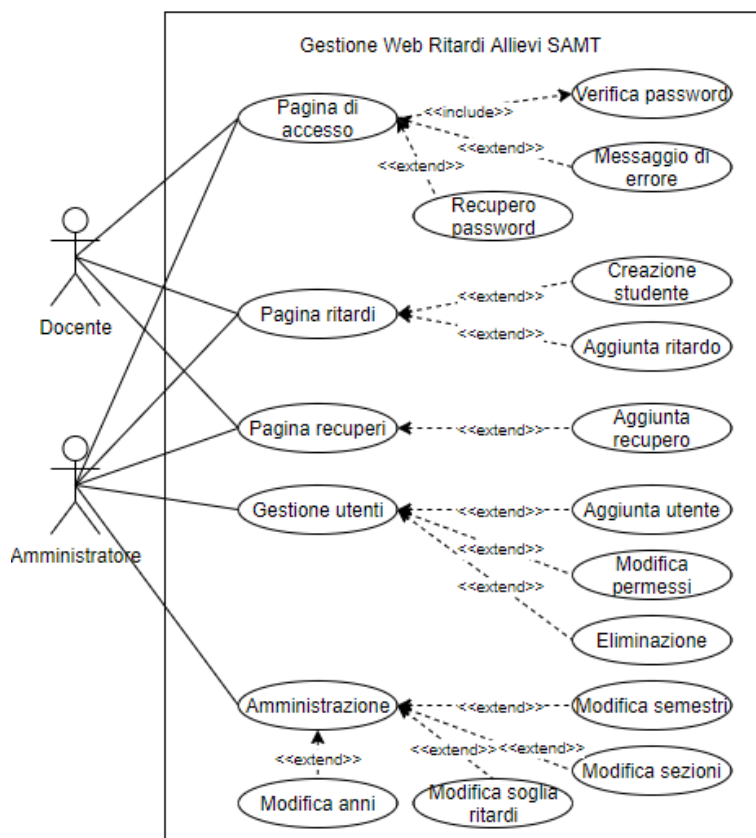


Figura 1 Schema caso d'uso.

## 2.4 Pianificazione

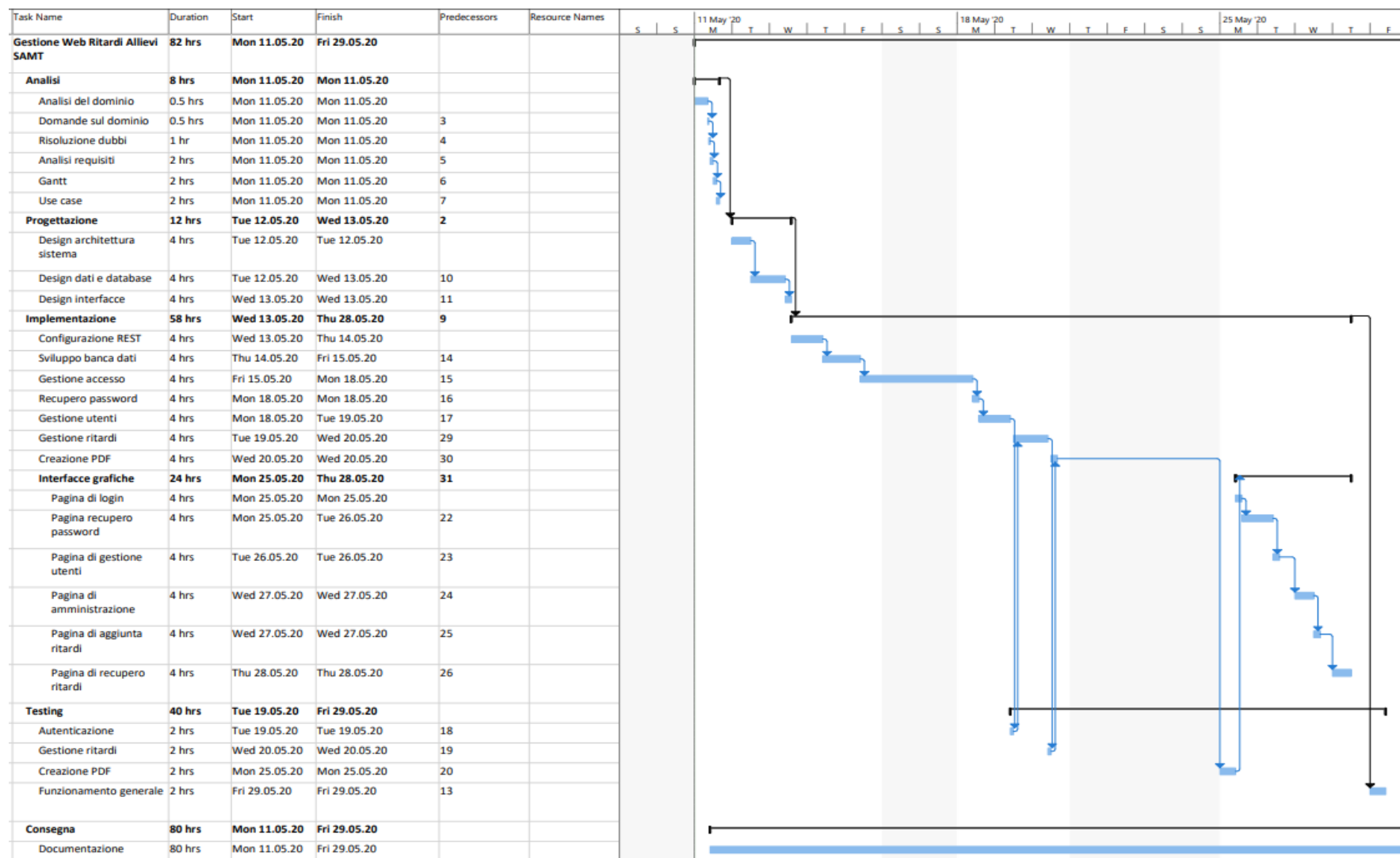


Figura 2 Diagramma di Gantt preventivo.

### 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.

<b>Analisi</b>	<b>8 hrs</b>	<b>Mon 11.05.20</b>	<b>Mon 11.05.20</b>		
Analisi del dominio	0.5 hrs	Mon 11.05.20	Mon 11.05.20		
Domande sul dominio	0.5 hrs	Mon 11.05.20	Mon 11.05.20	3	
Risoluzione dubbi	1 hr	Mon 11.05.20	Mon 11.05.20	4	
Analisi requisiti	2 hrs	Mon 11.05.20	Mon 11.05.20	5	
Gantt	2 hrs	Mon 11.05.20	Mon 11.05.20	6	
Use case	2 hrs	Mon 11.05.20	Mon 11.05.20	7	

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.

<b>Progettazione</b>	<b>12 hrs</b>	<b>Tue 12.05.20</b>	<b>Wed 13.05.20</b>	<b>2</b>	
Design architettura sistema	4 hrs	Tue 12.05.20	Tue 12.05.20		
Design dati e database	4 hrs	Tue 12.05.20	Wed 13.05.20	10	
Design interfacce	4 hrs	Wed 13.05.20	Wed 13.05.20	11	

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.

<b>Implementazione</b>	<b>58 hrs</b>	<b>Wed 13.05.20</b>	<b>Thu 28.05.20</b>	<b>9</b>	
Configurazione REST	4 hrs	Wed 13.05.20	Thu 14.05.20		
Sviluppo banca dati	4 hrs	Thu 14.05.20	Fri 15.05.20	14	
Gestione accesso	4 hrs	Fri 15.05.20	Mon 18.05.20	15	
Recupero password	4 hrs	Mon 18.05.20	Mon 18.05.20	16	
Gestione utenti	4 hrs	Mon 18.05.20	Tue 19.05.20	17	
Gestione ritardi	4 hrs	Tue 19.05.20	Wed 20.05.20	29	
Creazione PDF	4 hrs	Wed 20.05.20	Wed 20.05.20	30	
<b>Interfacce grafiche</b>	<b>24 hrs</b>	<b>Mon 25.05.20</b>	<b>Thu 28.05.20</b>	<b>31</b>	
Pagina di login	4 hrs	Mon 25.05.20	Mon 25.05.20		
Pagina recupero password	4 hrs	Mon 25.05.20	Tue 26.05.20	22	
Pagina di gestione utenti	4 hrs	Tue 26.05.20	Tue 26.05.20	23	
Pagina di amministrazione	4 hrs	Wed 27.05.20	Wed 27.05.20	24	
Pagina di aggiunta ritardi	4 hrs	Wed 27.05.20	Wed 27.05.20	25	
Pagina di recupero ritardi	4 hrs	Thu 28.05.20	Thu 28.05.20	26	

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'intero 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.

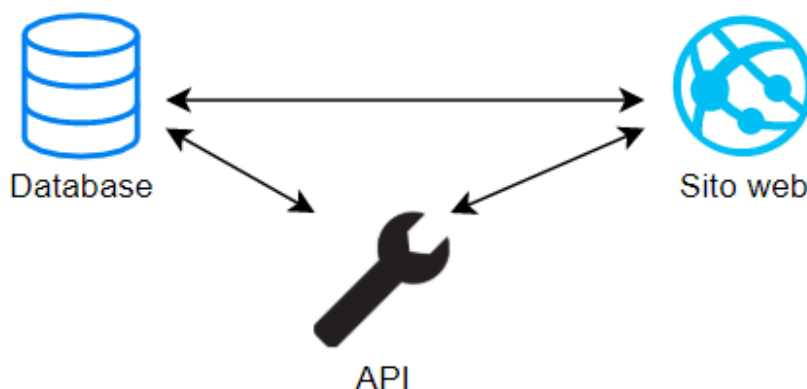


Figura 8 Architettura del sistema semplificata.

### 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 tabelle. 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 tabelle, 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.

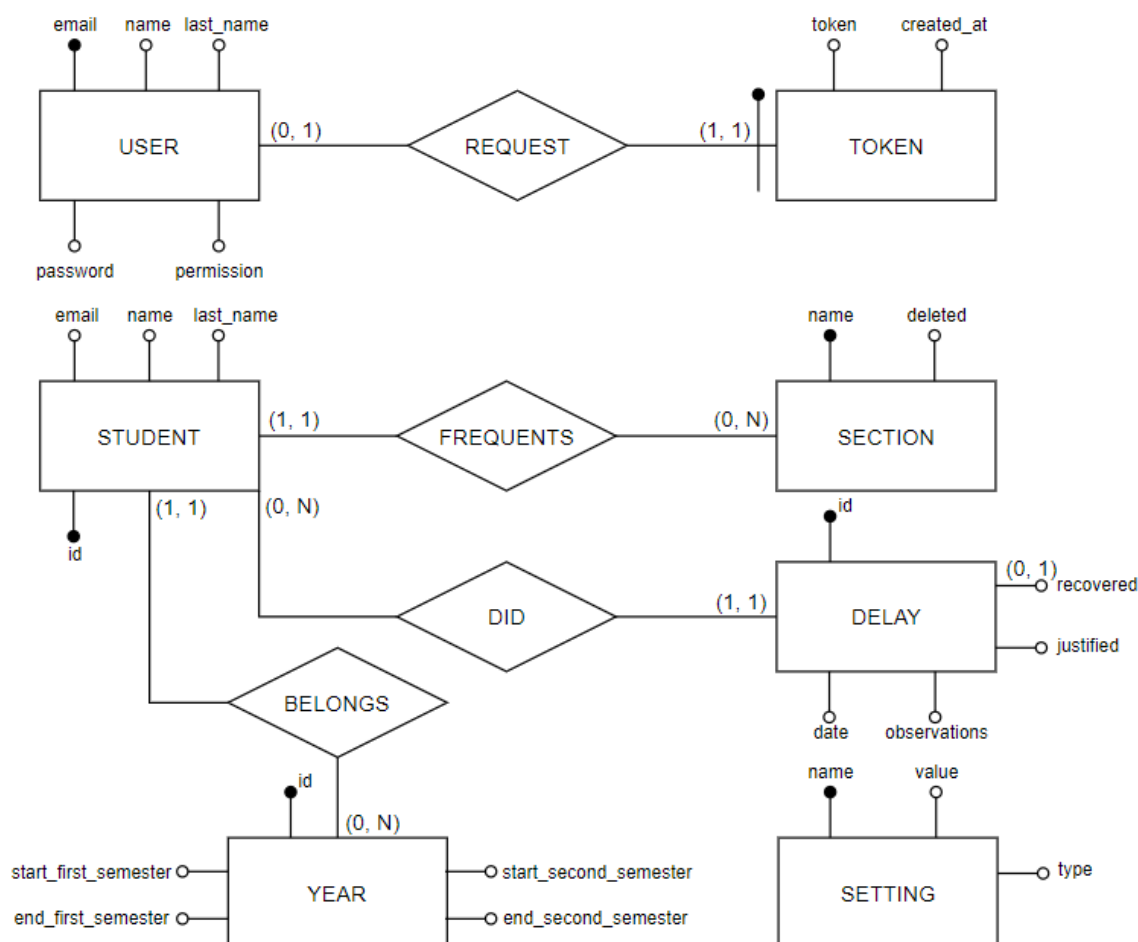


Figura 9 Schema ER banca dati.

### 3.2.2 Descrizioni delle tabelle

USER	
Attributo	Descrizione
email	Rappresenta un indirizzo e-mail dell'utente. È un attributo di tipo <u>stringa</u> con limite di 255 caratteri. Non può essere nullo e deve essere univoco.  Esempio: filippo.finke@sam-trevano.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.I2KvE7SK6le1UH67QVikBlyBjjTHgVG

permission	<p>Rappresenta il permesso di un utente. È un attributo di tipo intero e non può essere nullo. Può contenere i seguenti valori:</p> <ul style="list-style-type: none"> <li>1 - Inserimento ritardi.</li> <li>2 - Visione ritardi.</li> <li>4 - Creazione PDF.</li> <li>8 - Amministratore.</li> </ul> <p>Esempio: 8</p>
------------	---

<b>TOKEN</b>	
Attributo	Descrizione
token	<p>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.</p> <p>Esempio: 4b9eb466ad2615314c8194b1c46e6ef8e910d0b41a682e32de73503883e09b58</p>
created_at	<p>Rappresenta la data di creazione del codice di recupero password. È un attributo di tipo DATETIME e non può essere nullo.</p> <p>Esempio: 2020-05-12 14:16:59</p>

<b>STUDENT</b>	
Attributo	Descrizione
id	<p>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.</p> <p>Esempio: 1</p>
email	<p>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.</p> <p>Esempio: <a href="mailto:filippo.finke@samtrevano.ch">filippo.finke@samtrevano.ch</a></p>
name	<p>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.</p> <p>Esempio: Filippo</p>
last_name	<p>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.</p> <p>Esempio: Finke</p>

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

<b>DELAY</b>	
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.

<b>SETTING</b>	
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

<b>YEAR</b>	
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 primo semestre. È un attributo di tipo DATE.  Esempio: 2020-05-11
end_second_semester	Rappresenta la data di fine del primo 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.

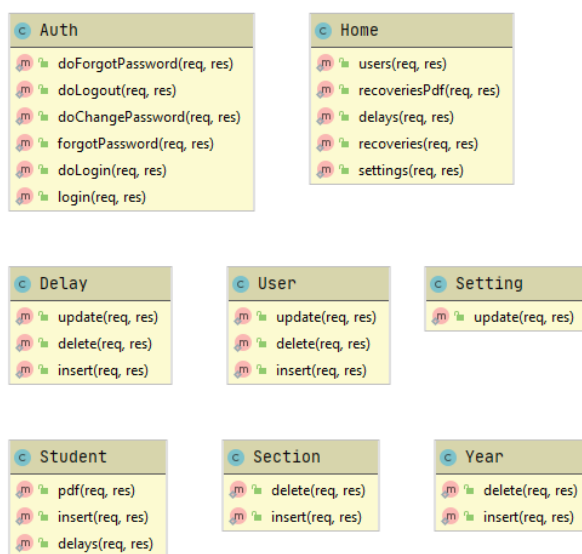


Figura 10 UML Classi controller.

#### 3.3.2 UML Libs

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

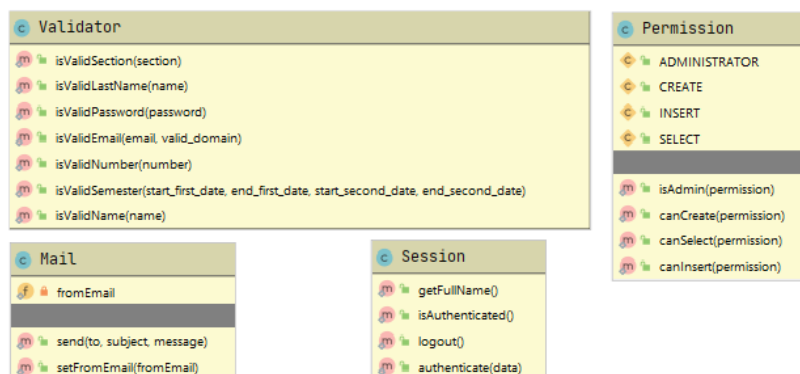


Figura 11 UML Classi libs.

### 3.3.3 UML Middlewares

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

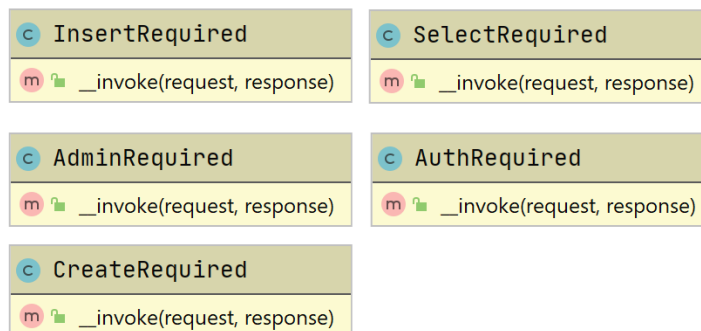


Figura 12 UML Classi middlewares.

### 3.3.4 UML Models

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

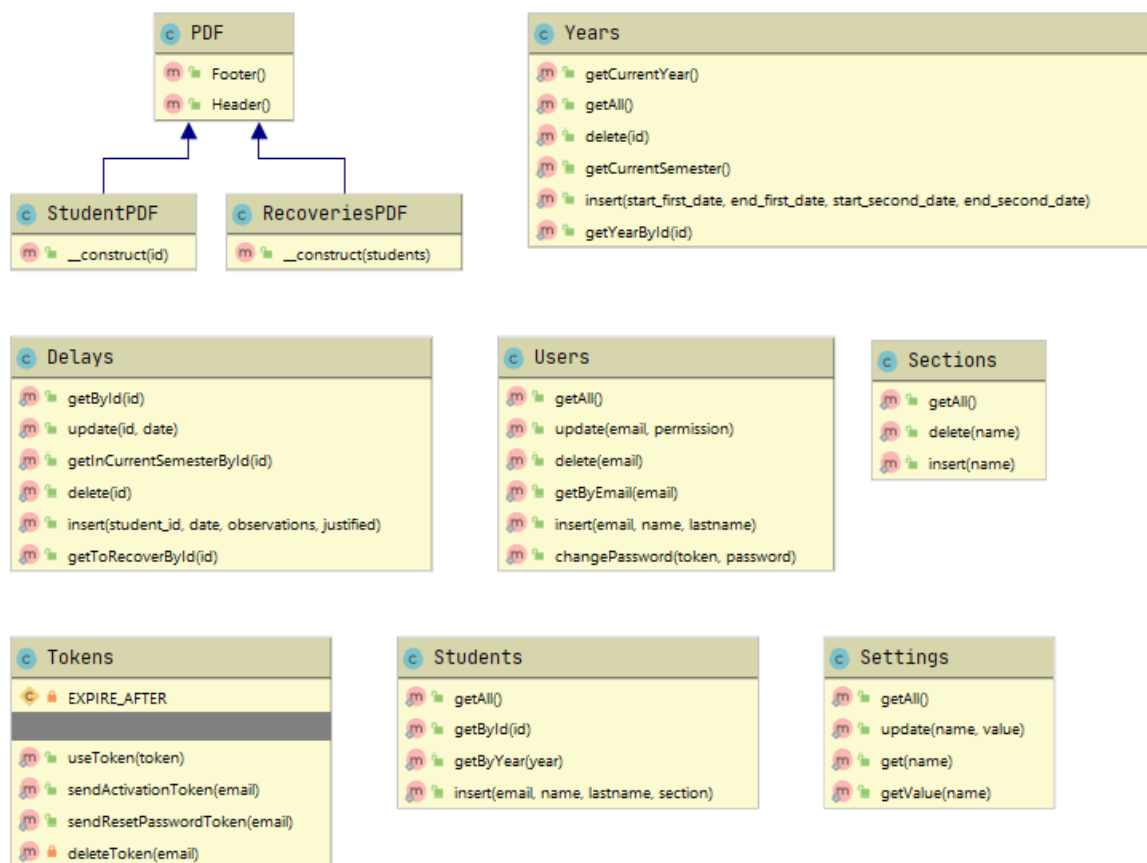


Figura 13 UML Classi models.

### 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.

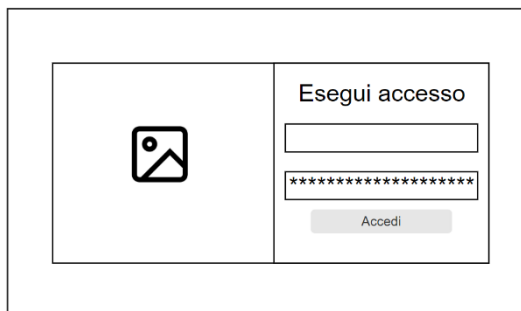


Figura 14 Mockup pagina di accesso.

#### 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.

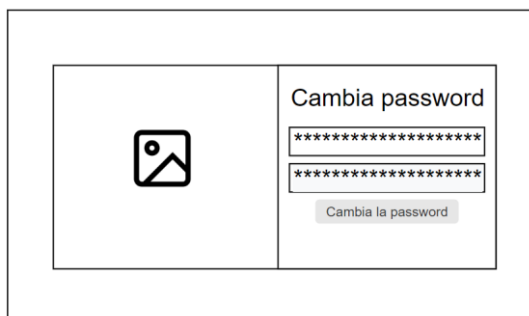


Figura 15 Mockup pagina di cambio password.

#### 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.



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 e studenti.

<b>Ritardi</b>	<b>Utente</b>	
Ritardi	Studenti	
Recuperi	Nome Cognome	Aggiungi ritardo
Utenti	Nome Cognome	Aggiungi ritardo
Impostazioni	Nome Cognome	Aggiungi ritardo
	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.

<b>Ritardi</b>	<b>Utente</b>	
Ritardi	Recuperi	
Recuperi	Nome Cognome	Aggiungi recupero
Utenti	Nome Cognome	Aggiungi recupero
Impostazioni	Nome Cognome	Aggiungi recupero
	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.

<b>Ritardi</b>	<b>Utente</b>	
Ritardi	Utenti	
Recuperi	Nome Cognome	Modifica Elimina
Utenti	Nome Cognome	Modifica Elimina
Impostazioni	Nome Cognome	Modifica Elimina
	Nome Cognome	Modifica Elimina
	<div>Aggiungi utente</div>	

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.

Ritardi	Utente	
Ritardi	Impostazioni	
Recuperi	Impostazione	VALORE
Utenti	Impostazione	VALORE
Impostazioni	Impostazione	VALORE
	<div>Applica</div>	

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@samtreveno.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.

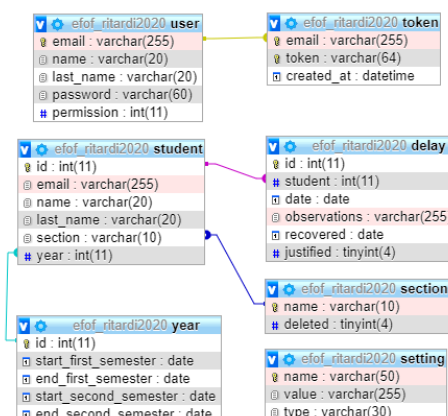


Figura 21 Schema database generato.

## 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:



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.

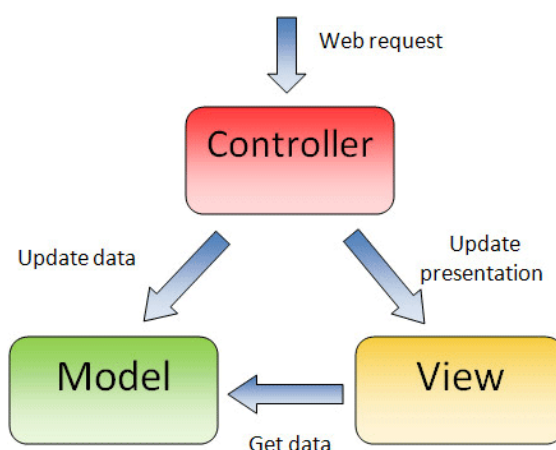


Figura 22 Schema pattern MVC.

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.



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 architetturale REST. Ho deciso di utilizzare REST in modo da semplificare l'implementazione delle chiamate per quando riguarda aggiornamento, inserimento e cancellazione di dati.

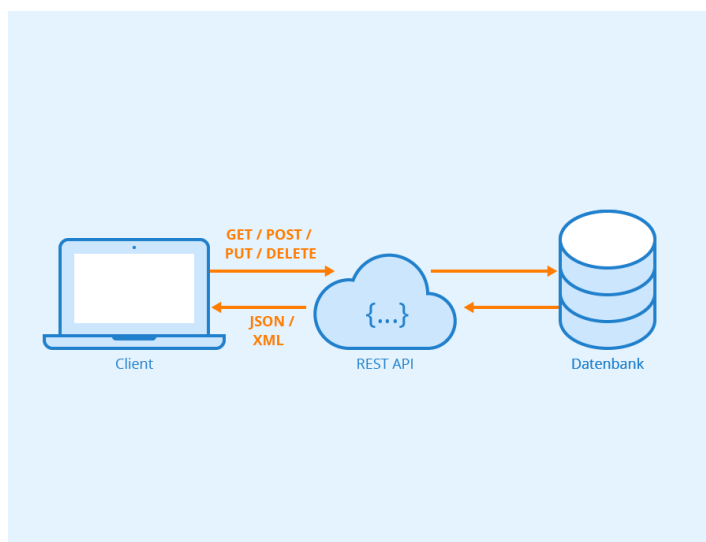


Figura 23 Schema pattern REST.

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::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.
$routeur = 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:

```

$routeur->get('/percorso', 'Classe::metodo');
$routeur->post('/percorso', 'Classe::metodo');
$routeur->put('/percorso', 'Classe::metodo');
$routeur->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(
    $routeur->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($name)
{
    return preg_match('/^[A-Za-zÀ-ÖØ-öø-ÿ]{1,20}$/', $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-öø-ÿ]{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";
    $content = "Salve,<br>può cambiare la sua password premendo il seguente lin
k: <a href='$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() - cr
eated_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 getByld 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 getByld($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 chiama 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 getCurrentSemesterById($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);
$stm->bindValue(':date', $date);
$stm->bindValue(':observations', $observations);
$stm->bindValue(':justified', $justified);
try {
    $stm->execute();
    $id = $pdo->lastInsertId();
    if (count(self::getToRecoverById($student_id)) > 0 && !$justified) {
        $date = date("d.m.Y", strtotime($date));
        $email = Students::getId($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.');
```

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";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':id', $id);
    $stm->bindValue(':date', $date);
    try {
        return $stm->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";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':id', $id);
    try {
        return $stm->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 interv
            allo di tempo!");
        }
    }

    $pdo = Database::getConnection();
    $query = "INSERT INTO year VALUES(null, :sfs, :efs, :sss, :ess)";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':sfs', $start_first_date);
    $stm->bindValue(':efs', $end_first_date);
    $stm->bindValue(':sss', $start_second_date);
    $stm->bindValue(':ess', $end_second_date);
    try {
        if ($stm->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";
    $stm = $pdo->prepare($query);
    $stm->bindValue(':id', $id);
    try {
        return $stm->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@sam-trevano.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)";
$stmt = $pdo->prepare($query);
$stmt->bindValue(':name', $name);
try {
    return $stmt->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 parametrio 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 `bcrypt`. 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 una 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](https://www.freepik.com/free-vector/login-concept-illustration_6183517.htm)

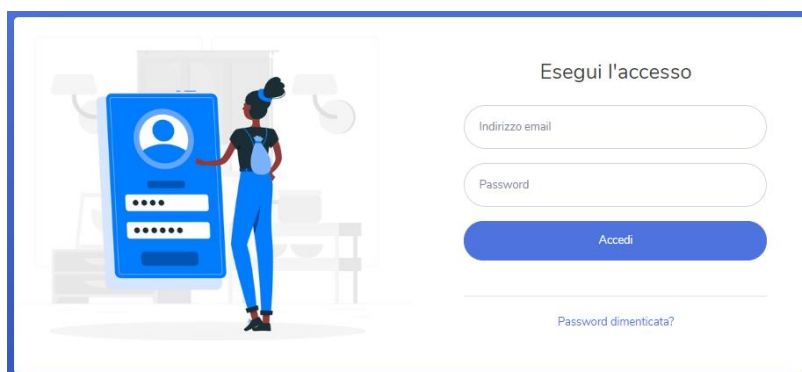


Figura 26 Pagina di accesso.

##### 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](https://www.freepik.com/free-vector/login-concept-illustration_6183517.htm)

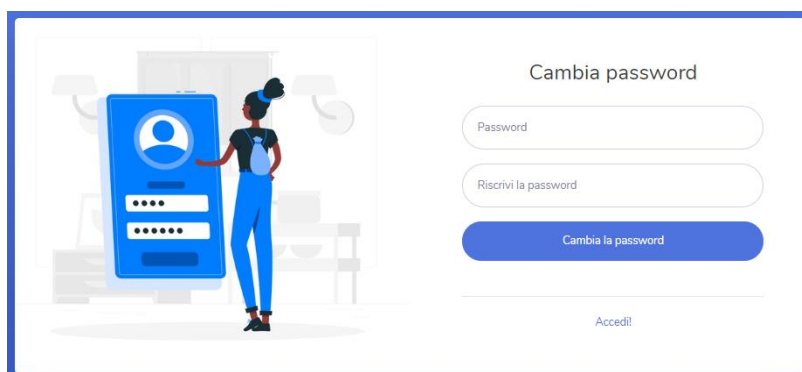


Figura 27 Pagina di cambio password.

#### 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](https://www.freepik.com/free-vector/forgot-password-concept-illustration_7070629.htm)

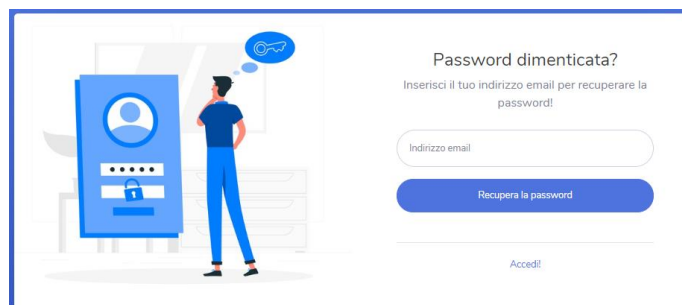


Figura 28 Pagina di recupero password.

#### 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.

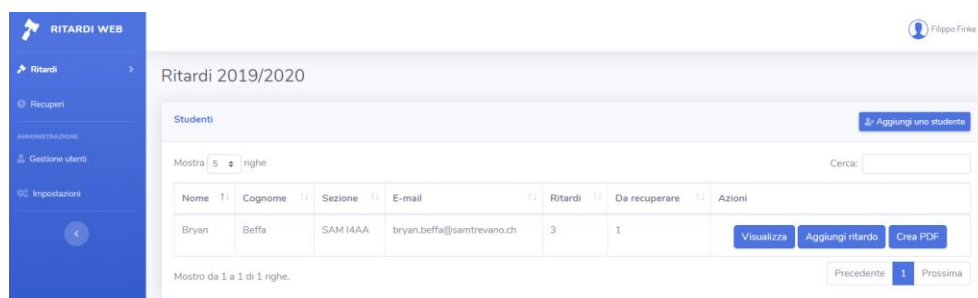


Figura 29 Pagina di gestione ritardi.

#### 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.

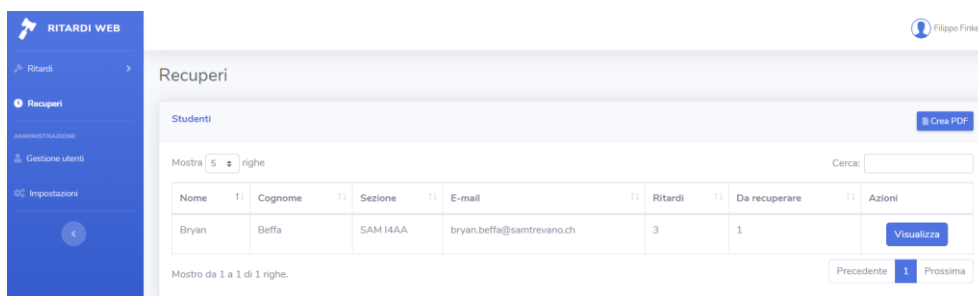


Figura 30 Pagina di gestione recuperi.

#### 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.

**Gestione utenti**

Utenti + Aggiungi un utente

Mostra 5 righe Cerca:

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 <span>Elimina</span>
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 <span>Elimina</span>

Mostro da 1 a 2 di 2 righe. Precedente 1 Prossima

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.

**Impostazioni**

**Sezioni** + Aggiungi una sezione

Mostra 5 righe Cerca:

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

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

**Anni scolastici** + Aggiungi un anno scolastico

Mostra 5 righe Cerca:

Primo semestre	Secondo semestre	Azioni
01.09.2019 - 16.01.2020	17.01.2020 - 25.06.2020	<span>Elimina</span>

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

**Configurazione**

Cerca:

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

Mostro da 1 a 2 di 2 righe.

Figura 32 Pagina impostazioni.

## 5 Test

### 5.1 Protocollo di test

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

<b>Test Case:</b>	TC-001	<b>Nome:</b>	Accesso all'applicativo con credenziali valide.
<b>Riferimento:</b>	REQ-001		
<b>Descrizione:</b>	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.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Recarsi nella pagina di accesso dell'applicativo.</li> <li>2. Inserire come indirizzo e-mail: "<a href="mailto:admin@samtreveno.ch">admin@samtreveno.ch</a>"</li> <li>3. Inserire come password: "123456"</li> <li>4. Premere il bottone "Accedi"</li> </ol>		
<b>Risultati attesi:</b>	L'accesso all'applicativo avverrà con successo e l'utente verrà reindirizzato alla pagina principale del sito web.		

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

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

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

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

<b>Test Case:</b>	TC-005	<b>Nome:</b>	Pagina di gestione utenti
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	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.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Selezionare "Gestione Utenti" dalla barra di navigazione laterale</li> </ol>		
<b>Risultati attesi:</b>	Verranno mostrati tutti gli utenti in una tabella e non sarà possibile eliminare l'utente corrente.		

<b>Test Case:</b>	TC-006	<b>Nome:</b>	Aggiunta di un utente
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	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.		

<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Selezionare "Gestione Utenti" dalla barra di navigazione laterale</li> <li>2. Premere il pulsante "Aggiungi un utente"</li> <li>3. Inserire come nome: "Test"</li> <li>4. Inserire come cognome: "Test"</li> <li>5. Inserire come e-mail: "<a href="mailto:test@test.com">test@test.com</a>"</li> <li>6. Premere il bottone "Crea".</li> </ol>
<b>Risultati attesi:</b>	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.

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

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

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

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

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

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



<b>Risultati attesi:</b>	Verrà creato un nuovo anno scolastico ed aggiunto alla tabella.
--------------------------	---

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

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

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

<b>Test Case:</b>	TC-016	<b>Nome:</b>	Aggiunta di uno studente
<b>Riferimento:</b>	REQ-005		
<b>Descrizione:</b>	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"		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Selezionare "Ritardi" dalla barra di navigazione laterale.</li> </ol>		

	<ol style="list-style-type: none"> <li>Premere il bottone: “Aggiungi uno studente”</li> <li>Inserire come nome: “Test”</li> <li>Inserire come cognome: “Test”</li> <li>Inserire come e-mail: “<a href="mailto:test@samtrevano.ch">test@samtrevano.ch</a>”</li> <li>Selezionare una sezione.</li> <li>Premere il bottone: “Crea”</li> </ol>
<b>Risultati attesi:</b>	Lo studente verrà creato ed aggiunto alla tabella.

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

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

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

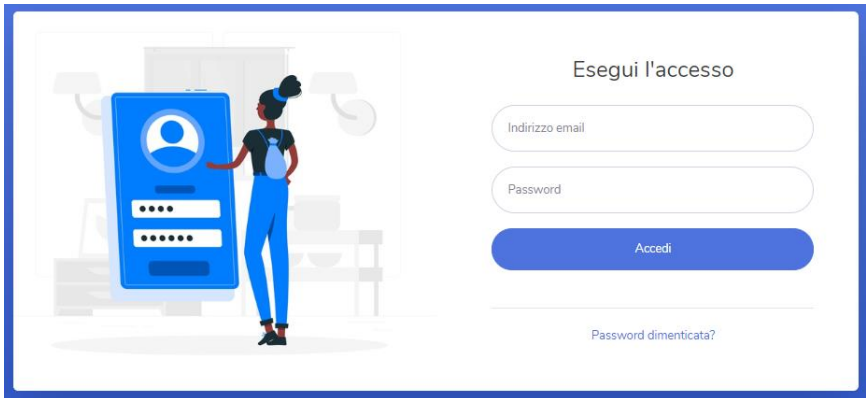
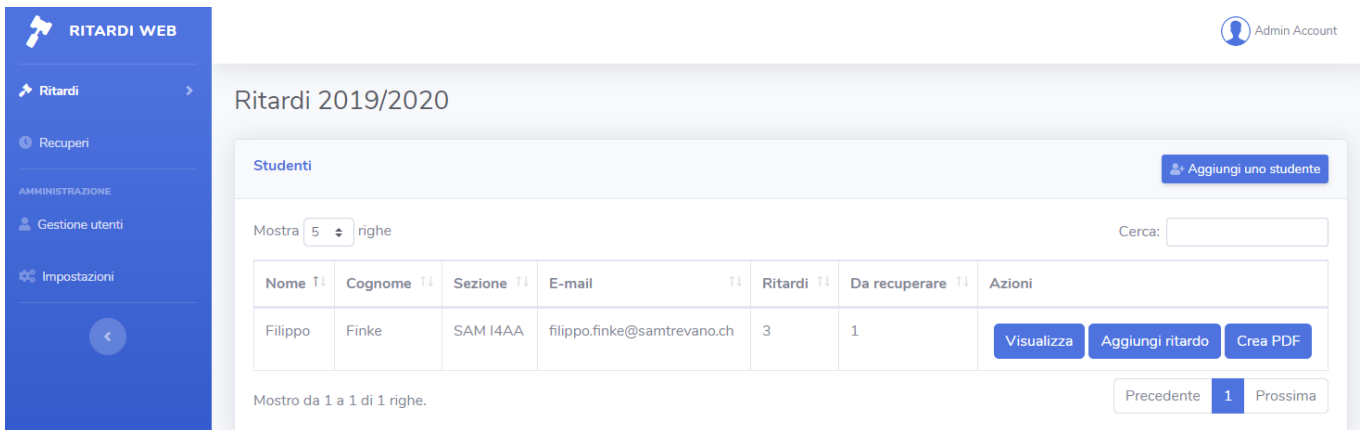
<b>Test Case:</b>	TC-020	<b>Nome:</b>	Pagina di gestione recuperi
<b>Riferimento:</b>	REQ-006		
<b>Descrizione:</b>	L'applicativo deve possedere una pagina attraverso la quale deve essere possibile gestire i recuperi dei ritardi degli studenti.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo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

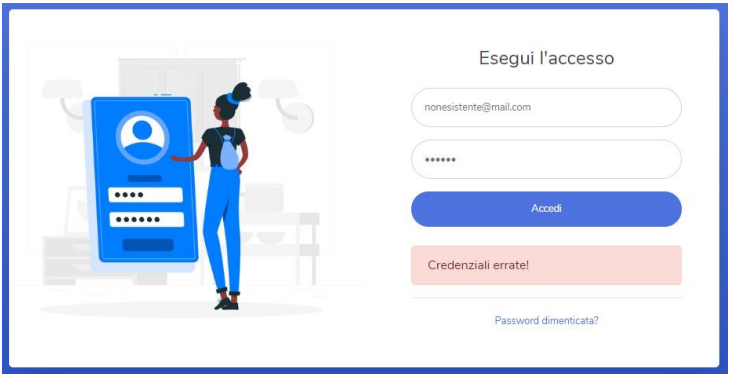
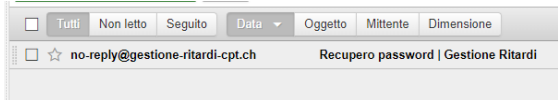
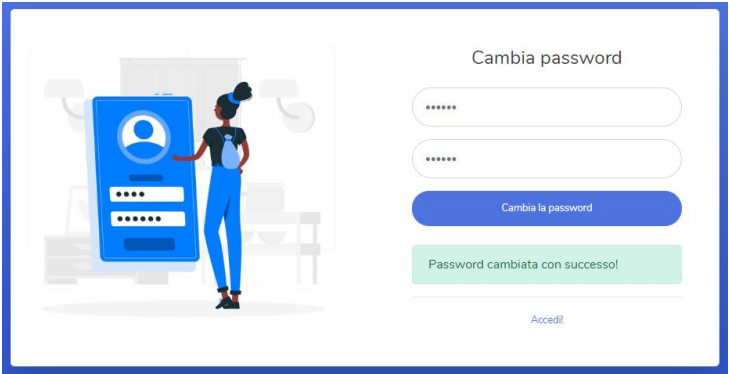
<b>Test Case:</b>	TC-021	<b>Nome:</b>	Aggiunta di un recupero
<b>Riferimento:</b>	REQ-006		
<b>Descrizione:</b>	Attraverso la pagina di gestione dei recuperi deve essere possibile aggiungere un recupero di un ritardo ad uno studente.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Selezionare "Recuperi" dalla barra di navigazione laterale.</li> <li>2. Premere il bottone: "Visualizza" su un utente qualsiasi.</li> <li>3. Premere il bottone: "Recuperato" su un ritardo.</li> <li>4. Inserire la data: "25.05.2020"</li> <li>5. Premere il bottone: "Salva"</li> </ol>		
<b>Risultati attesi:</b>	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.		

<b>Test Case:</b>	TC-022	<b>Nome:</b>	Creazione file PDF lista studenti che devono recuperare.
<b>Riferimento:</b>	REQ-006		
<b>Descrizione:</b>	Attraverso la pagina di gestione dei recuperi deve essere possibile creare un file PDF contenente gli studenti che devono recuperare dei ritardi.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo ed accessibile dalla rete. L'utente deve aver eseguito l'accesso al sito web.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Selezionare "Recuperi" dalla barra di navigazione laterale.</li> <li>2. Premere il bottone: "Crea PDF"</li> </ol>		
<b>Risultati attesi:</b>	Verrà aperto un modale contenente un file PDF.		


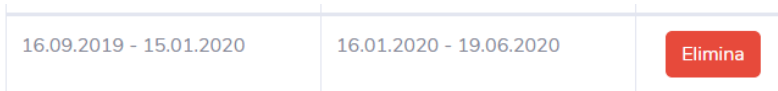
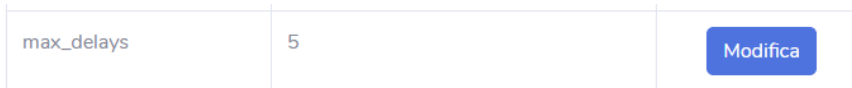
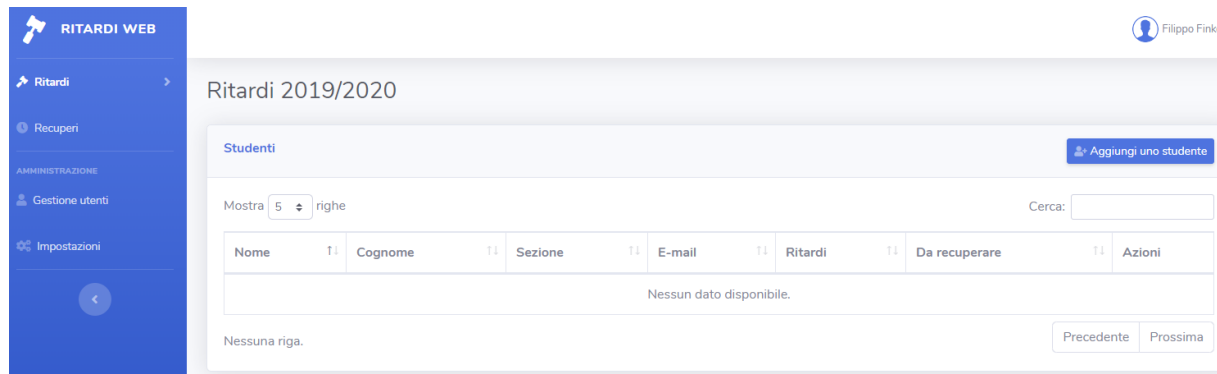
<b>Test Case:</b>	TC-023	<b>Nome:</b>	Applicativo su hosting esterno
<b>Riferimento:</b>	REQ-007		
<b>Descrizione:</b>	L'applicativo deve essere messo in produzione su un hosting.		
<b>Prerequisiti:</b>	L'applicativo deve essere attivo su un hosting ed accessibile dalla rete.		
<b>Procedura:</b>	1. Navigare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


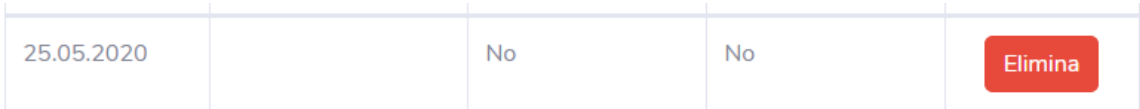

## 5.2 Risultati test

Codice test	Stato	Risultati
TC-000	PASSATO	<p>Utilizzando Edge ed accedendo al sito web viene mostrata la pagina di accesso correttamente.</p>  <p>Figura 33 Pagina di accesso.</p>
TC-001	PASSATO	<p>L'accesso avviene correttamente e l'utente viene reindirizzato alla pagina principale di gestione dei ritardi.</p>  <p>Figura 34 Pagina principale.</p>

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

TC-005	PASSATO	<p>Vengono mostrati tutti gli utenti presenti nel sito web e non vi è la possibilità di eliminare l'utente corrente.</p>  <p>Figura 38 Pagina di gestione utenti.</p>
TC-006	PASSATO	<p>Viene inviata una e-mail di attivazione del profilo e viene mostrato nella lista utenti.</p>  <p>Figura 39 Creazione utente.</p>
TC-007	PASSATO	L'utente viene eliminato e rimosso dalla lista utenti.
TC-008	PASSATO	I permessi dell'utente vengono aggiornati.
TC-009	PASSATO	<p>Vengono mostrate tutte le impostazioni del sito web in tre tabelle diverse.</p>  <p>Figura 40 Pagina impostazioni.</p>

TC-010	PASSATO	<p>La sezione viene aggiunta correttamente.</p>  <p>Figura 41 Sezione creata correttamente.</p>
TC-011	PASSATO	La sezione viene rimossa con successo.
TC-012	PASSATO	<p>L'anno scolastico viene aggiunto correttamente.</p>  <p>Figura 42 Anno scolastico creato correttamente.</p>
TC-013	PASSATO	L'anno scolastico viene rimosso con successo.
TC-014	PASSATO	<p>L'impostazione viene aggiornata con successo.</p>  <p>Figura 43 Impostazione aggiornata.</p>
TC-015	PASSATO	<p>La pagina di gestione ritardi viene visualizzata correttamente.</p>  <p>Figura 44 Pagina di gestione ritardi.</p>

TC-016	PASSATO	<p>Lo studente viene creato correttamente.</p>  <p>Figura 45 Creazione studente.</p>
TC-017	PASSATO	<p>Il ritardo viene aggiunto con successo e il contatore presente nella tabella viene incrementato.</p>  <p>Figura 46 Aggiunta ritardo.</p>
TC-018	PASSATO	<p>Viene mostrato un modale con tutti i ritardi dello studente.</p>  <p>Figura 47 Modale di visualizzazione ritardi.</p>



TC-019

PASSATO

Viene mostrato un modale contenente il PDF.

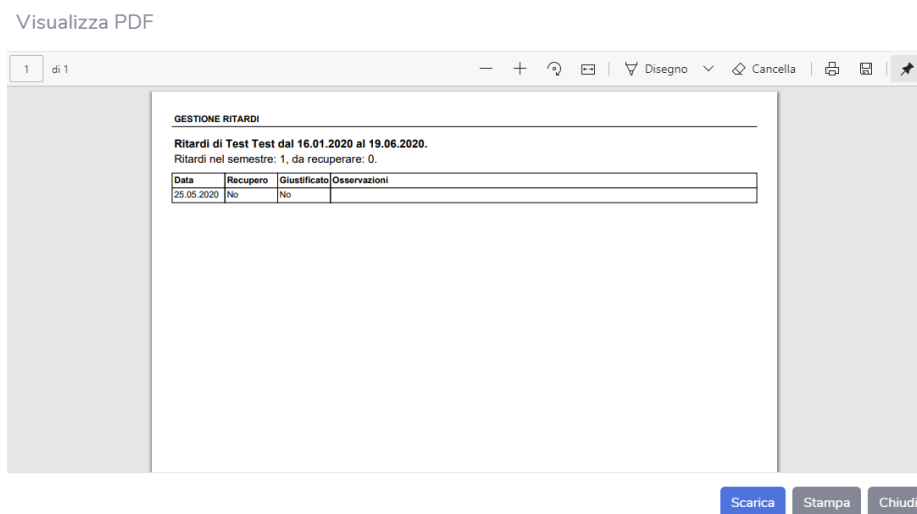


Figura 48 Generazione PDF studente.

TC-020

PASSATO

Viene mostrata la pagina di gestione recuperi con la lista di studenti che devono recuperare dei ritardi.

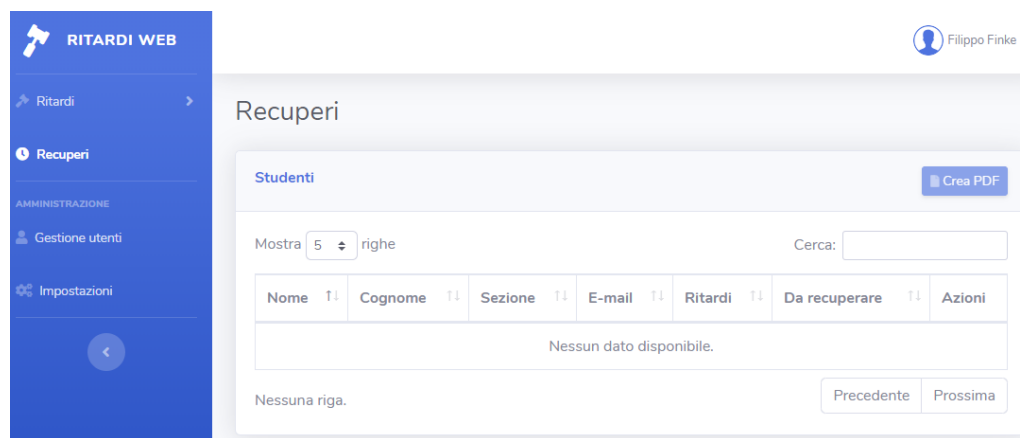
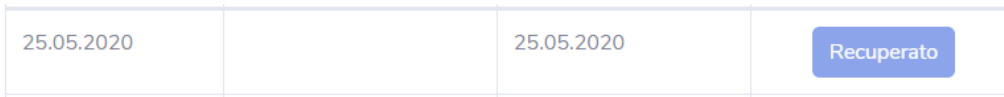
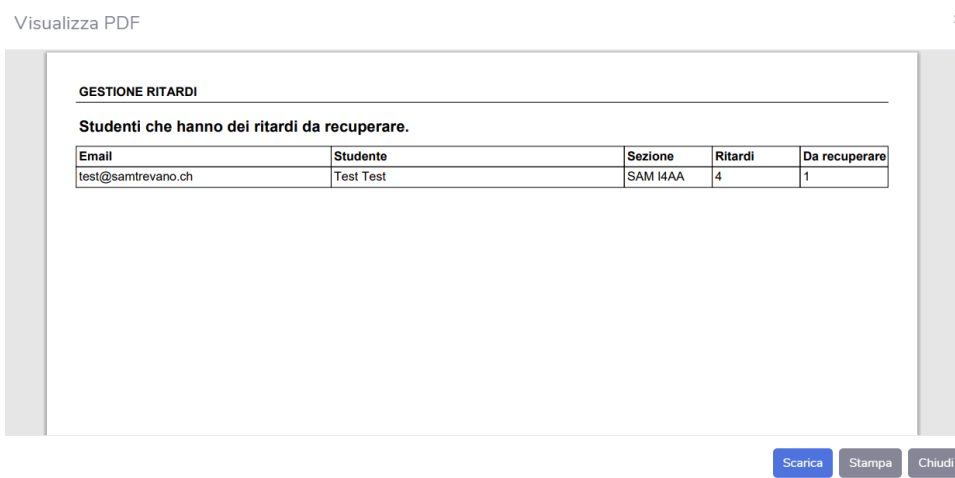


Figura 49 Pagina di gestione recuperi.

TC-021	PASSATO	<p>Il ritardo viene segnato come recuperato e il conteggio di ritardi da recuperare viene decrementato.</p>  <p>Figura 50 Ritardo recuperato.</p>
TC-022	PASSATO	<p>Il PDF contenente la lista degli studenti che devono recuperare dei ritardi viene creato correttamente.</p>  <p>Figura 51 PDF Recupero ritardi.</p>
TC-023	PASSATO	<p>L'applicativo è raggiungibile anche su un hosting esterno.</p>

### 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.

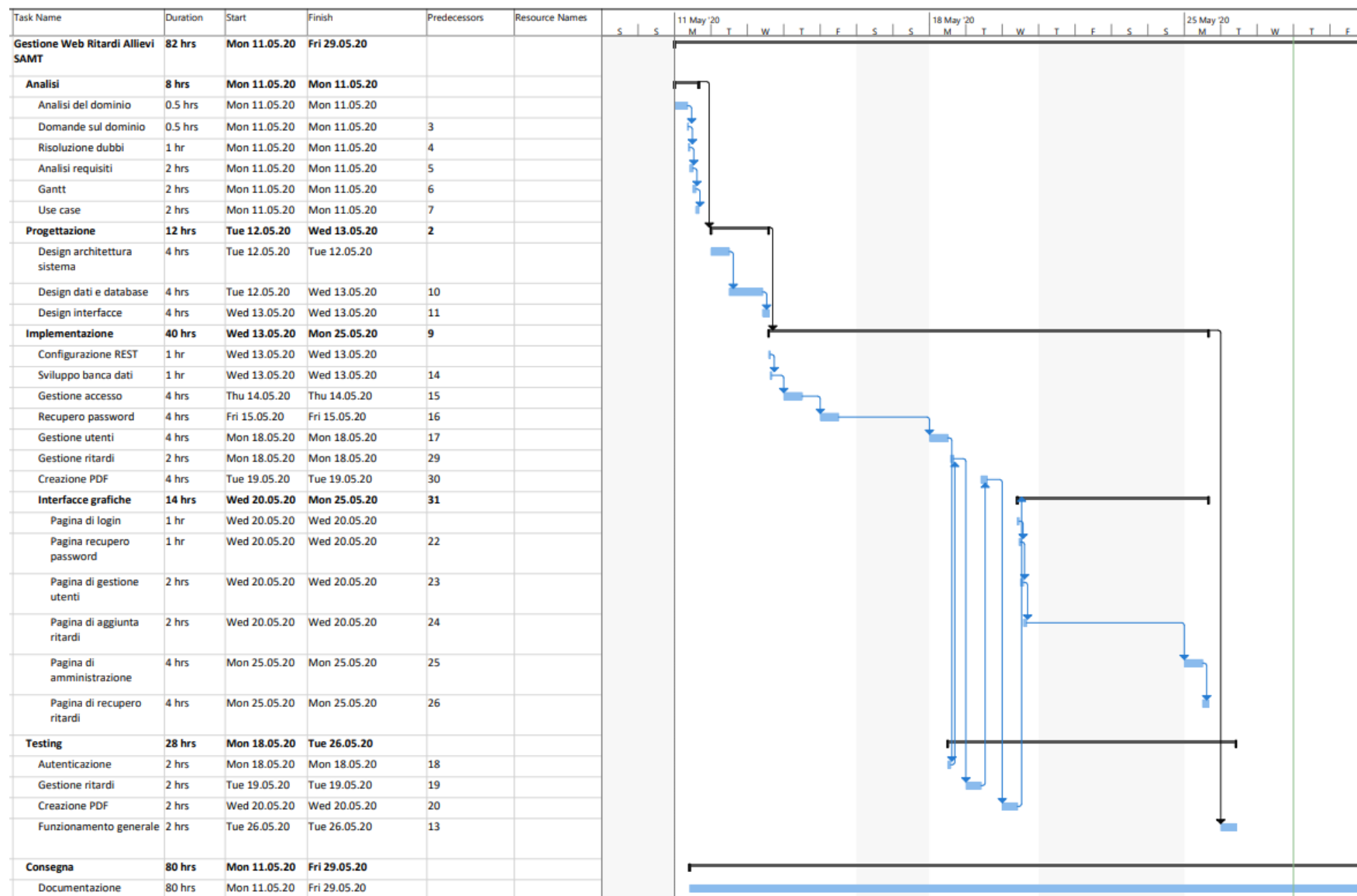


Figura 52 Diagramma di Gantt consuntivo.

## 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 programmatori 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](http://gitsam.cpt.local/lavoro_finale_lpi_2020/gestione-web-ritardi-allievi-samt)