



Gestione richieste congedi docenti

Titolo del progetto: Gestione richieste congedi docenti
Alunno/a: Filippo Finke
Classe: I4AC
Anno scolastico: 2019/2020
Docente responsabile: Fabrizio Valsangiacomo

1	<u>INTRODUZIONE</u>	5
1.1	INFORMAZIONI SUL PROGETTO	5
1.2	ABSTRACT.....	5
1.3	SCOPO	5
2	<u>ANALISI</u>	6
2.1	ANALISI DEL DOMINIO	6
2.2	ANALISI E SPECIFICA DEI REQUISITI	6
2.3	USE CASE	10
2.3.1	UTENTI DI DOMINIO	10
2.3.2	UTENTI LOCALI	11
2.4	PIANIFICAZIONE	12
2.4.1	ANALISI	13
2.4.2	PROGETTAZIONE	13
2.4.3	IMPLEMENTAZIONE.....	13
2.4.4	TESTING.....	14
2.4.5	CONSEGNA.....	14
2.5	ANALISI DEI MEZZI	14
2.5.1	SOFTWARE	14
2.5.2	HARDWARE	14
3	<u>PROGETTAZIONE</u>	15
3.1	SCHEMA DI RETE CONCETTUALE.....	15
3.2	DESIGN DEI DATI E DATABASE	16
3.2.1	SCHEMA ER.....	16
3.2.2	DESCRIZIONI DELLE TABELLE.....	16
3.2.3	SCHEMA LOGICO	20
3.3	DESIGN DELLE INTERFAZIE	20
3.3.1	PAGINA DI ACCESSO	20
3.3.2	PAGINA PRINCIPALE	21
3.3.3	PAGINA DI UN RECIPIENTE	21
3.3.4	PAGINA STORICO	22
3.3.5	PAGINA AMMINISTRAZIONE UTENTI	22
3.3.6	PAGINA AMMINISTRAZIONE CONGEDI.....	23
3.4	DESIGN DELL'ARCHITETTURA DEL SISTEMA.....	23
3.4.1	UML CONTROLLERS.....	23
3.4.2	UML LIBS	24
3.4.3	UML MIDDLEWARES	24
3.4.4	UML MODELS	25
4	<u>IMPLEMENTAZIONE</u>	25
4.1	GESTIONE VERSIONI	25
4.2	GESTORE DI PACCHETTI	26

4.3 DATABASE	26
4.4 APPLICATIVO WEB	27
4.4.1 STRUTTURA	27
4.4.1.1 Model View Controller (MVC).....	28
4.4.1.2 Representational State Transfer (REST)	28
4.4.2 INOLTRO RICHIESTE	29
4.4.3 CONFIGURAZIONE.....	29
4.4.4 AUTENTICAZIONE	30
4.4.4.1 Autenticazione banca dati locale	30
4.4.4.2 Interfaccia LDAP	31
4.4.4.3 Connessione server LDAP da remoto.....	32
4.4.4.4 Gestione delle sessioni e percorsi.....	33
4.4.5 INTERFACCIA INVIO EMAIL	35
4.4.6 VALIDAZIONE CAMPI	36
4.4.7 RISORSE LOCALI.....	37
4.4.8 CALENDARIO.....	38
4.4.9 GESTIONE DATI E INTERROGAZIONE BANCA DATI.....	43
4.4.9.1 Contenitori	43
4.4.9.2 Tabella administrators	44
4.4.9.3 Tabella users	47
4.4.9.4 Tabella permissions.....	49
4.4.9.5 Tabella reasons	49
4.4.9.6 Tabella requests.....	52
4.4.9.7 Stato delle richieste di congedo.....	57
4.4.9.8 Creazione PDF richieste di congedo.....	58
4.4.9.9 Tabella substitutes	59
4.4.9.10 Tabella tokens	60
4.4.10 SICUREZZA.....	60
4.4.10.1 Interrogazione database	61
4.4.10.2 Salvataggio dati	61
4.4.10.3 Password locali.....	61
4.4.10.4 Recupero password	61
4.4.11 INTERFACCE GRAFICHE	63
4.4.11.1 Pagina di accesso	63
4.4.11.2 Pagina principale.....	64
4.4.11.3 Pagina personale, congedi in uscita	65
4.4.11.4 Pagina personale, storico	65
4.4.11.5 Contenitore segreteria.....	66
4.4.11.6 Contenitore direzione e vice direzione	67
4.4.11.7 Storico generale	68
4.4.11.8 Visualizzazione PDF	69
4.4.11.9 Pagina di amministrazione utenti	69
4.4.11.10 Pagina di amministrazione motivazioni	71
4.4.11.11 Pagina di recupero/impostazione password.....	72
5 TEST	72
5.1 PROTOCOLLO DI TEST.....	72
5.2 RISULTATI TEST	79

5.3 MANCANZE/LIMITAZIONI CONOSCIUTE.....	88
<u>6 CONSUNTIVO</u>	88
<u>7 CONCLUSIONI.....</u>	90
7.1 SVILUPPI FUTURI.....	90
7.1.1 SVILUPPO DI CONTROLLI SUL CONTEGGIO GIORNI.....	90
7.2 CONSIDERAZIONI PERSONALI	90
<u>8 GLOSSARIO</u>	90
<u>9 SITOGRAFIA</u>	91
<u>10 ALLEGATI.....</u>	91

1 Introduzione

1.1 Informazioni sul progetto

Allievi coinvolti nel progetto: Filippo Finke

Classe: Informatica 4AC presso la sede Scuola Arti e Mestieri Trevano

Docenti responsabili: Fabrizio Valsangiacomo

Data inizio: 23.01.2020

Data consegna: 06.04.2020

1.2 Abstract

The aim of the project "Gestione richieste congedi docenti" is to transfer the management system for teachers' leave requests to a computer system. The aim is therefore to make it easier for teachers to write their leave requests, by integrating a system of access to the web application through the school domain accounts using the LDAP protocol and to simplify the work regarding the management of these requests by the school administration (secretariat, deputy directors and management). This application will then be used by teachers if they have to plan a leave of absence. The application also makes the drafting and management of leave much faster than paper, thanks to the digital system will be possible to perform detailed research, in-depth checks and additional features that would take a long time on paper. The leave will pass through different containers inside the web application where they will be reviewed, approved, modified or refused by the school administration. Thanks to this system you will also be able to keep track of your requested leave and take their status into account. There will also be an automatic system for the creation of the paper representation of leave through the use of PDF.

1.3 Scopo

Lo scopo del progetto “Gestione richieste congedi docenti” è quello di trasferire il sistema di gestione delle richieste di congedo dei docenti su un sistema informatico. Lo scopo è quindi di rendere più semplice la stesura dei congedi da parte dei docenti, integrando un sistema di accesso all'applicativo web attraverso gli account del dominio scolastico sfruttando il protocollo LDAP e di semplificare il lavoro per quanto riguarda la gestione di essi da parte della amministrazione scolastica (segreteria, vice direttori e direzione). Questo applicativo verrà quindi utilizzato dai docenti qualora dovessero pianificare un congedo. L'applicativo inoltre rende la stesura e la gestione dei congedi molto più veloce rispetto alla carta, grazie al sistema digitale sarà possibile eseguire ricerche dettagliate, controlli approfonditi e ulteriori funzionalità che su carta prederebbero molto tempo. I congedi attraverseranno diversi contenitori presenti all'interno dell'applicativo web nel quale verranno revisionati, approvati, modificati o rifiutati da parte dell'amministrazione scolastica. Grazie a questo sistema si potrà tenere traccia anche dei propri congedi richiesti e tenere conto del loro stato. Sarà inoltre presente un sistema automatico per la creazione della rappresentazione cartacea dei congedi attraverso l'utilizzo di PDF.

2 Analisi

2.1 Analisi del dominio

È stato richiesto lo sviluppo di un gestionale web per la richiesta di congedo dei docenti del Centro Professionale di Trevano. Il prodotto dovrà essere un applicativo web, accessibile attraverso la rete utilizzando qualsiasi browser. Gli utenti che accederanno a questo applicativo saranno principalmente i docenti che andranno ad inserire le proprie richieste di congedo. Inoltre saranno disponibili ulteriori permessi per la gestione di congedi. Le varie richieste da parte dei docenti verranno gestite attraverso dei contenitori virtuali i quali saranno accessibili solamente a determinati gruppi di utenti. Vi sarà un contenitore dedicato alla segreteria la quale riceverà i congedi inviati dai docenti ed avrà il compito di verificare la validità dei dati immessi e di approvare il congedo per il passaggio al recipiente successivo. Il recipiente successivo sarà accessibile da vice direzione e direzione i quali avranno la parola definitiva sullo stato del congedo, ovvero se sarà stato accettato oppure no. I permessi saranno a cascata quindi chi farà parte del gruppo di direzione potrà vedere tutti i recipienti sottostanti. È inoltre richiesta una pagina accessibile solamente a chi è amministratore locale dell'applicativo e che quindi si occuperà della gestione di esso (aggiunta di titoli di congedi, descrizioni, gestione amministratori locali, etc.). Il sistema inoltre utilizzerà un sistema di posta elettronica per tenere informati sia i docenti che l'amministrazione scolastica sullo stato dei vari congedi. Quando un congedo sarà approvato definitivamente dalla direzione scolastica verrà generato un PDF di esso che verrà poi inviato per email ai diretti interessati. All'interno dell'applicativo sarà inoltre presente uno storico dei congedi registrati all'interno del sistema in modo di permettere all'amministrazione di eseguire ricerche e controlli. L'accesso all'applicativo verrà gestito attraverso l'utilizzo di due diversi sistemi, il primo sistema di autenticazione che verrà controllato sarà l'accesso locale all'applicativo per permettere ai gestori del prodotto di ricevere ulteriori permessi. Il secondo sistema che verrà utilizzato più spesso sarà l'accesso attraverso le stesse credenziali dei profili della rete scolastica attraverso l'interrogazione dei server LDAP.

2.2 Analisi e specifica dei requisiti

È richiesto da parte del committente lo sviluppo di un applicativo che sia accessibile tramite rete e che sia web. Il prodotto deve potersi interfacciare con i server LDAP della scuola in modo da permettere l'accesso da parte dei docenti utilizzando le credenziali del dominio. È inoltre richiesto che vengano gestiti a parte gli utenti amministratore che si occuperanno della gestione del software. Attraverso l'autenticazione LDAP l'applicativo dovrà riconoscere il livello di permesso dell'utente e se farà parte di uno dei seguenti gruppi:

- Docente
- Segreteria
- Vice direttori
- Direttori o Direttrici

Attraverso questi gruppi ricavati dai server scolastici sarà possibile dunque distinguere la tipologia di utente e di come verranno mostrate le pagine stesse. Per eseguire l'autenticazione è dunque richiesta una pagina dedicata, la quale avrà anche la funzione di recupero password per gli utenti locali. Per distinguere tra utenti di gestione dell'applicativo e utenti del dominio del CPT verrà utilizzato il nominativo, che per gli utenti locali sarà il proprio indirizzo di posta elettronica mentre per gli utenti del CPT sarà "nome.cognome". Il sistema richiede spesso l'utilizzo di email quindi è richiesto di implementare una parte del programma che si occupi solamente della gestione dell'invio di posta elettronica. Sarà presente una struttura a recipienti nel quale verranno salvati i vari congedi, questa struttura varierà in modo dinamico in base ai permessi citati in precedenza. Una volta eseguito l'accesso gli utenti avranno la possibilità di compilare il formulario dei congedi ed inviarlo al primo recipiente, ovvero la segreteria. Per tenere traccia dell'andamento dei congedi scolastici è inoltre richiesta una sezione separata nel quale mostrare tutti i congedi registrati nel sistema, attraverso questa pagina sarà anche possibile eseguire delle ricerche mirate. Per la gestione dell'applicativo invece è richiesta una sezione riservata solamente agli utenti amministratori locali che avranno la possibilità di aggiungerne altri oppure eliminarli. All'aggiunta di un amministratore locale verrà inviata un email all'indirizzo specificato contenente una password generata in modo casuale che verrà poi sostituita al primo accesso. Una sezione aggiuntiva nel pannello di amministrazione è la possibilità di gestire le cause e/o motivazioni dei vari congedi.

ID: REQ-000	
Nome	Piattaforma dell'applicativo
Priorità	1
Versione	1.0
Note	Si necessita di un applicativo web.
Sotto requisiti	
001	Il sito deve funzionare sui browser più utilizzati.

ID: REQ-001	
Nome	Interfaccia con server LDAP scolastico
Priorità	1
Versione	1.0
Note	Si necessita di interfacciarsi con i server LDAP scolastici per eseguire il login degli utenti normali.
Sotto requisiti	
001	Possibilità di accedere con username formato “nome.cognome” e password LDAP.

ID: REQ-002	
Nome	Interfaccia con server MySQL locale
Priorità	1
Versione	1.0
Note	Si necessita di interfacciarsi con un server MySQL per autenticare gli utenti locali di gestione del software.
Sotto requisiti	
001	Possibilità di accedere con email come username.

ID: REQ-003	
Nome	Interfaccia di invio posta elettronica
Priorità	1
Versione	1.0
Note	Si necessita di una interfaccia che permetta di inviare semplicemente messaggi di posta elettronica.

ID: REQ-004	
Nome	Pagina di accesso
Priorità	1
Versione	1.0
Note	Si necessita di una pagina specifica per l'accesso all'applicativo web. / Dipende dal requisito REQ-001, REQ-002, REQ-003
Sotto requisiti	
001	Si necessita una maschera di login
002	Si necessita di poter distinguere la tipologia di accesso (LDAP o locale).
003	Finestra a comparsa al primo login per gli utenti locali.
004	Possibilità di recuperare la password, solamente per gli utenti locali, tramite email.

ID: REQ-005	
Nome	Pagina di un recipiente
Priorità	1
Versione	1.0
Note	Si richiede l'utilizzo di "recipienti" virtuali per organizzare i congedi, verrà utilizzata una sola pagina generata in modo dinamico per ogni contenitore. / Dipende dal requisito REQ-001, REQ-002, REQ-003
Sotto requisiti	
001	Sarà accessibile solamente a chi ne ha i permessi (segreteria, vice direttori o direzione).
002	Possibilità di vedere i congedi al suo interno.
003	Possibilità di modificare dati dei congedi.
004	Possibilità di mandare i congedi ad un recipiente successivo.

ID: REQ-006	
Nome	Pagina principale
Priorità	1
Versione	1.0
Note	Si richiede una pagina principale nel quale i docenti del CPT possano creare ed inviare i congedi. Dipende dal requisito REQ-001, REQ-002, REQ-003
Sotto requisiti	
001	Possibilità di selezionare le cause del congedo.
002	Possibilità di inviare un congedo al recipiente successivo.

003	In caso si abbiano i permessi mostrare tutti i recipienti.
004	In caso si abbiano i permessi mostrare la pagina di storico.

ID: REQ-007	
Nome	Pagina storico
Priorità	1
Versione	1.0
Note	Si richiede una pagina nel quale venga mostrato lo storico dei congedi salvati nel sistema. Dipende dal requisito REQ-001, REQ-002, REQ-003
Sotto requisiti	
001	Possibilità di accedere solamente se si ha determinati permessi.
002	Possibilità di eseguire ricerche.
003	I congedi dovranno essere classificati per anno.

ID: REQ-008	
Nome	Pagina di amministrazione utenti
Priorità	1
Versione	1.0
Note	Si necessita di una pagina per la gestione degli utenti locali dell'applicativo. Dipende dal requisito REQ-001, REQ-002, REQ-003
Sotto requisiti	
001	Accessibile unicamente dagli utenti amministratori locali.
002	Possibilità di vedere tutti gli utenti locali disponibili.
003	Possibilità di aggiungere utenti locali con nome, cognome ed email come identificativo.
004	Possibilità di eliminare utenti locali (non si può eliminare se stessi).

ID: REQ-009	
Nome	Pagina di amministrazione motivazioni
Priorità	2
Versione	1.0
Note	Si necessita di una pagina per la gestione dei titoli delle motivazioni dei congedi.
Sotto requisiti	
001	Possibilità di aggiungere motivazioni.

Gestione richieste congedi docenti

002	Possibilità di modificare le motivazioni.
003	Possibilità di eliminare le motivazioni.

ID: REQ-010	
Nome	Messa in produzione
Priorità	1
Versione	1.0
Note	L'applicativo dovrà essere messo in produzione su un host esterno il prima possibile, questo in modo tale da permettere test approfonditi.

2.3 Use case

2.3.1 Utenti di dominio

In questo schema gli agenti hanno già eseguito l'accesso attraverso il server LDAP della scuola. Sono presenti quattro diversi livelli di permessi all'interno dell'applicativo web che sono rappresentati dagli agenti. Gli utenti comuni sono considerati "Docente", hanno la possibilità di accedere ad una pagina principale del sito web e di visionare il proprio contenitore personale. Inoltre possono anche compilare ed inviare congedi al livello successivo. Un'altra tipologia di utente è la "Segreteria" essa ha la possibilità di fare tutto ciò che un "Docente" può fare, ma in aggiunta può vedere tutti i congedi in attesa di revisione, che una volta revisionati possono essere mandati al livello successivo. Il livello successivo è la vice direzione e direzione, a questo livello di permessi è possibile eseguire tutto ciò che la "Segreteria" può fare, in più è possibile approvare o rifiutare i congedi in attesa. In più del "Docente" tutti gli altri agenti hanno la possibilità di accedere allo storico dei congedi nel quale è possibile eseguire delle ricerche.

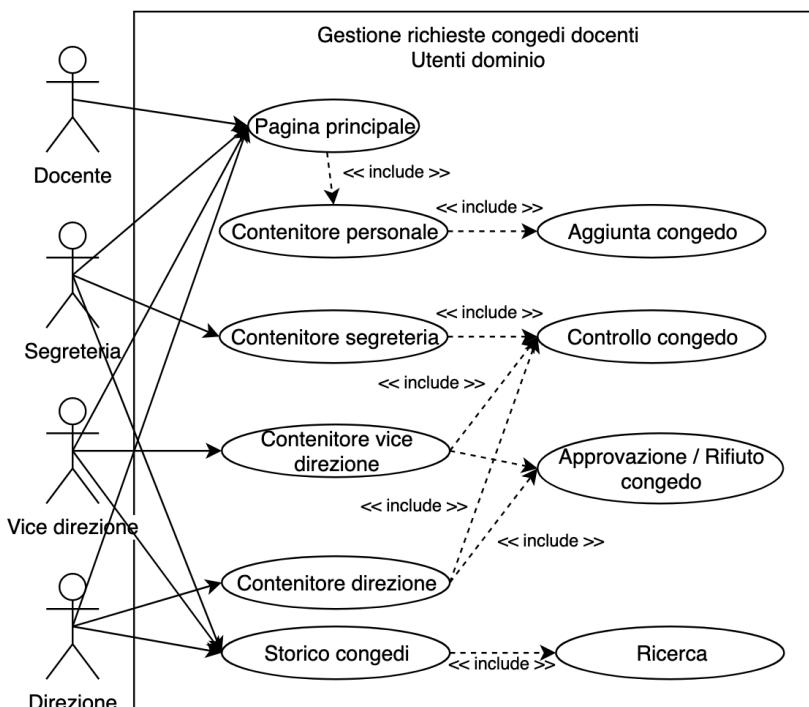


Figura 1 Schema caso d'uso utenti di dominio.

2.3.2 Utenti locali

In questo schema l'agente è uno solo, lo stato iniziale è che ha già eseguito l'accesso all'applicativo web. L'amministratore può gestire l'intero applicativo. Una volta entrato ha la possibilità di recarsi alla gestione degli utenti, nella quale può aggiungere, modificare ed eliminare utenti locali. Inoltre ha la possibilità di recarsi in un'altra pagina dedicata alla gestione delle cause/motivazioni dei congedi, attraverso questa pagina può aggiungerne, modificarne oppure eliminarne.

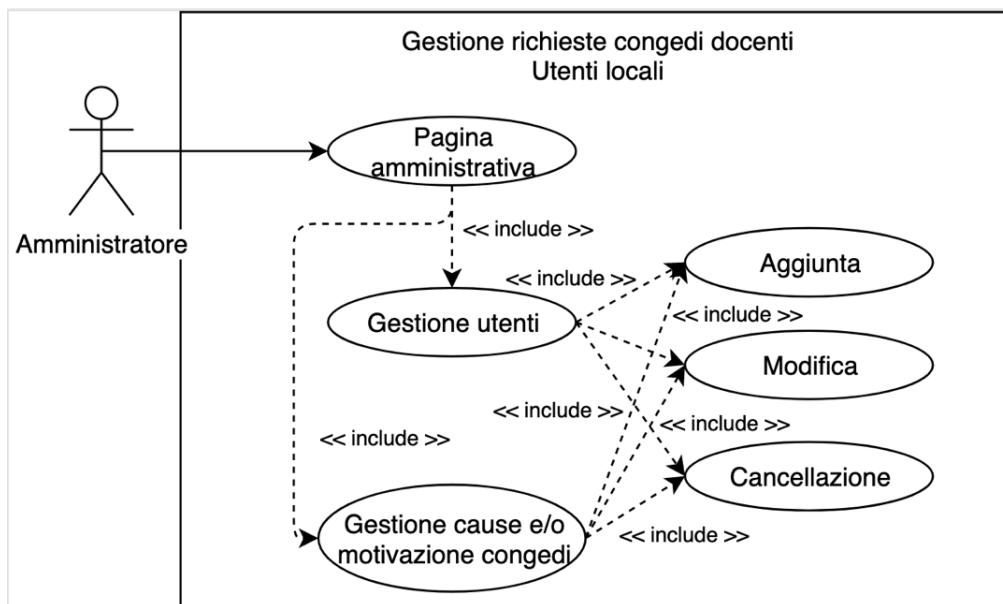


Figura 2 Schema caso d'uso utenti locali.

2.4 Pianificazione

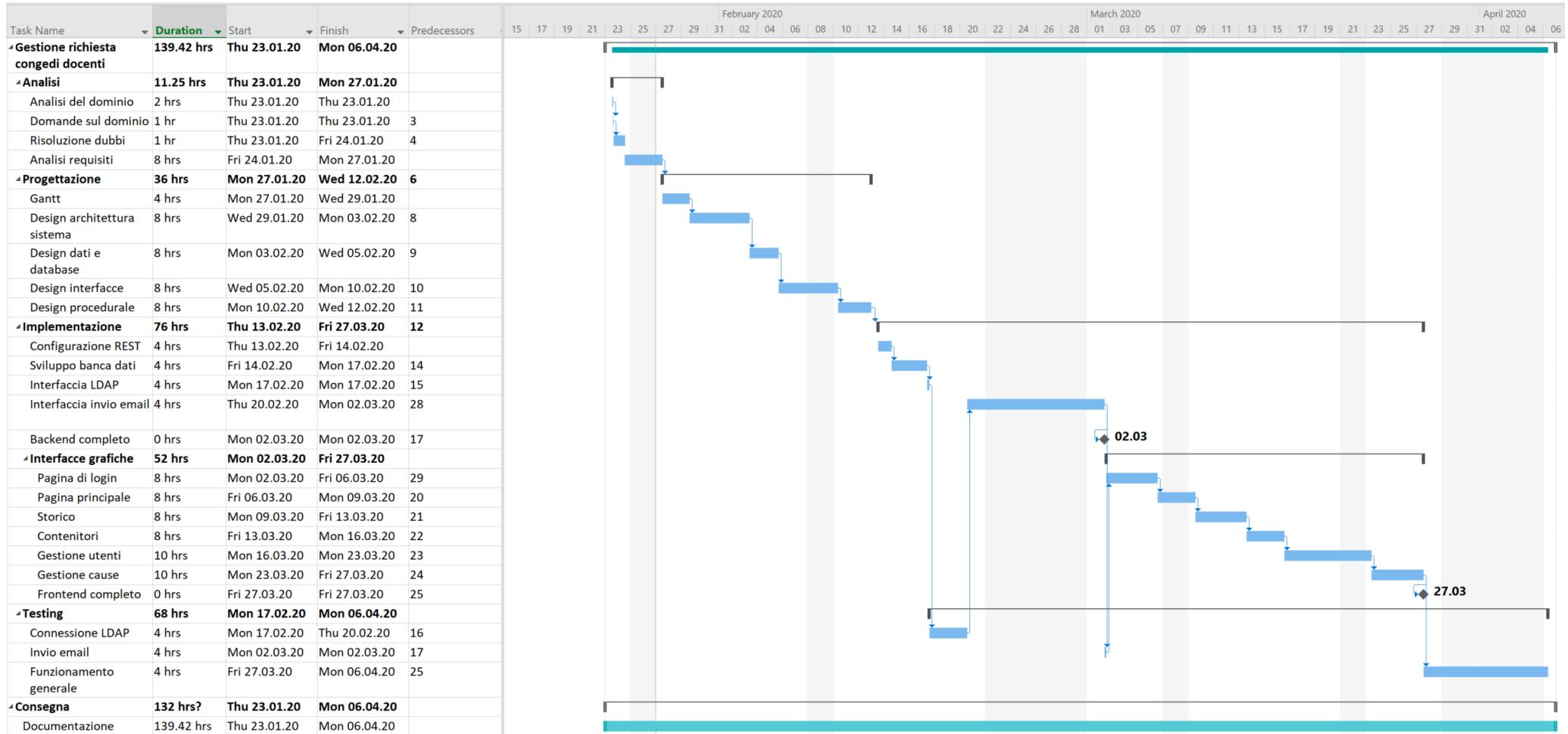


Figura 3 Diagramma di Gantt preventivo.

2.4.1 Analisi

Ho suddiviso la fase di analisi in cinque attività principali. Questa è la fase più corta rispetto alle altre fasi, come anche descritto nel quaderno dei compiti. Durante questa fase mi sono occupato di capire di cosa tratta il progetto e quali sono le richieste da parte del committente.

Analisi	11.25 hrs	Thu 23.01.20	Mon 27.01.20	
Analisi del dominio	2 hrs	Thu 23.01.20	Thu 23.01.20	
Domande sul dominio	1 hr	Thu 23.01.20	Thu 23.01.20	3
Risoluzione dubbi	1 hr	Thu 23.01.20	Fri 24.01.20	4
Analisi requisiti	8 hrs	Fri 24.01.20	Mon 27.01.20	

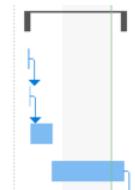


Figura 4 Diagramma di Gantt, Analisi.

2.4.2 Progettazione

La progettazione è la seconda fase del progetto “Gestione richiesta congedi docenti” composta da cinque attività. All’interno di questa fase di progettazione è incluso la creazione del diagramma di Gantt da seguire durante la durata del progetto, il design dell’architettura del sistema, il design dei dati e dei database, design delle interfacce ed in fine il design procedurale. Considero questa fase molto importante in quanto le fasi successive saranno basate su di essa.

Progettazione	36 hrs	Mon 27.01.20	Wed 12.02.20	6
Gantt	4 hrs	Mon 27.01.20	Wed 29.01.20	
Design architettura sistema	8 hrs	Wed 29.01.20	Mon 03.02.20	8
Design dati e database	8 hrs	Mon 03.02.20	Wed 05.02.20	9
Design interfacce	8 hrs	Wed 05.02.20	Mon 10.02.20	10
Design procedurale	8 hrs	Mon 10.02.20	Wed 12.02.20	11

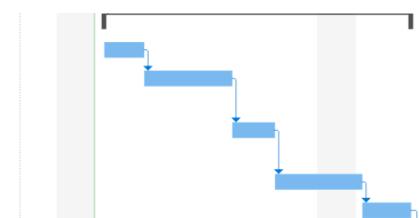


Figura 5 Diagramma di Gantt, Progettazione.

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. All’interno di questa sezione vi è la configurazione per l’utilizzo di REST, lo sviluppo della banca dati, lo sviluppo di classi per agevolare l’autenticazione attraverso LDAP e classi per la gestione dell’invio di posta elettronica. Inoltre è presente una pietra miliare per indicare lo sviluppo del backend. Vi è una sotto fase che consiste nello sviluppo delle varie interfacce grafiche, alla fine dello sviluppo di esse vi è un’altra pietra miliare che indica il termine del frontend.

Implementazione	76 hrs	Thu 13.02.20	Fri 27.03.20	12
Configurazione REST	4 hrs	Thu 13.02.20	Fri 14.02.20	
Sviluppo banca dati	4 hrs	Fri 14.02.20	Mon 17.02.20	14
Interfaccia LDAP	4 hrs	Mon 17.02.20	Mon 17.02.20	15
Interfaccia invio email	4 hrs	Thu 20.02.20	Mon 02.03.20	28
Backend completo	0 hrs	Mon 02.03.20	Mon 02.03.20	17
Interfaccie grafiche	52 hrs	Mon 02.03.20	Fri 27.03.20	
Pagina di login	8 hrs	Mon 02.03.20	Fri 06.03.20	29
Pagina principale	8 hrs	Fri 06.03.20	Mon 09.03.20	20
Storico	8 hrs	Mon 09.03.20	Fri 13.03.20	21
Contenitori	8 hrs	Fri 13.03.20	Mon 16.03.20	22
Gestione utenti	10 hrs	Mon 16.03.20	Mon 23.03.20	23
Gestione cause	10 hrs	Mon 23.03.20	Fri 27.03.20	24
Frontend completo	0 hrs	Fri 27.03.20	Fri 27.03.20	25

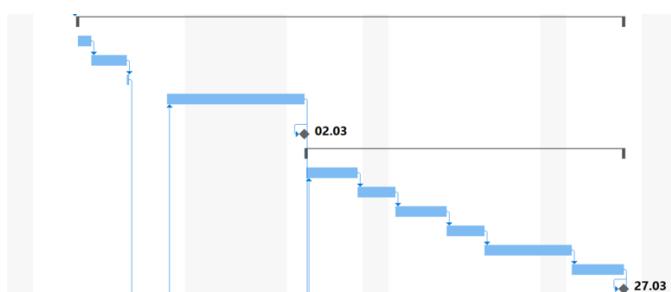


Figura 6 Diagramma di Gantt, Implementazione.

2.4.4 Testing

Un'altra fase molto importante è il testing, in questo caso viene data priorità di test delle classi di autenticazione LDAP e di invio di posta elettronica. È presente inoltre un'altra attività dedicata al test generale del prodotto una volta terminato frontend e backend.

Testing	68 hrs	Mon 17.02.20	Mon 06.04.20	
Connessione LDAP	4 hrs	Mon 17.02.20	Thu 20.02.20	16
Invio email	4 hrs	Mon 02.03.20	Mon 02.03.20	17
Funzionamento generale	4 hrs	Fri 27.03.20	Mon 06.04.20	25

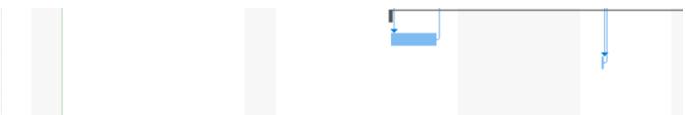


Figura 7 Diagramma di Gantt, Testing.

2.4.5 Consegna

L'ultima fase consiste nella consegna del prodotto, è presente una sola attività che riguarda la documentazione. Questo perché verrà aggiornata durante il corso di tutto il progetto.

Consegna	132 hrs?	Thu 23.01.20	Mon 06.04.20	
Documentazione	139.42 hrs	Thu 23.01.20	Mon 06.04.20	



Figura 8 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 2016
- Microsoft Project 2019
- Microsoft VS Code 1.37.1
- VMware Fusion 11.0
- MySQL 8.0.13
- PHP 7.3.5
- Draw.io (<https://draw.io>)
- HighlightCode (<https://highlight.hohli.com>)

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/>)
- EnDecryptText 2.0 – Ivan Raimondi

Template utilizzati:

- desk-app (<https://github.com/dropways/deskapp>)

Gestore librerie utilizzato:

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

2.5.2 Hardware

Il progetto è stato sviluppato su un MacBook Pro 2018.

Le specifiche hardware sono:

- 8 GB di RAM
- Intel Core i5 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 Schema di rete concettuale

Un computer che vuole collegarsi all'applicativo web deve essere collegato ad una rete nella quale sia presente anche il web server che si occupa di servire il sito. Questa rete può essere locale oppure pubblica. Quando i due dispositivi saranno connessi nella stessa rete basterà conoscere l'indirizzo IP del web server per potersi collegare. Il computer per collegarsi deve possedere un software in grado di interpretare il protocollo di comunicazione, quindi un semplice browser. Inoltre quando l'accesso viene eseguito attraverso le credenziali del dominio LDAP il web server l'applicativo comunicherà con il server scolastico per la convalida dei dati stessi.

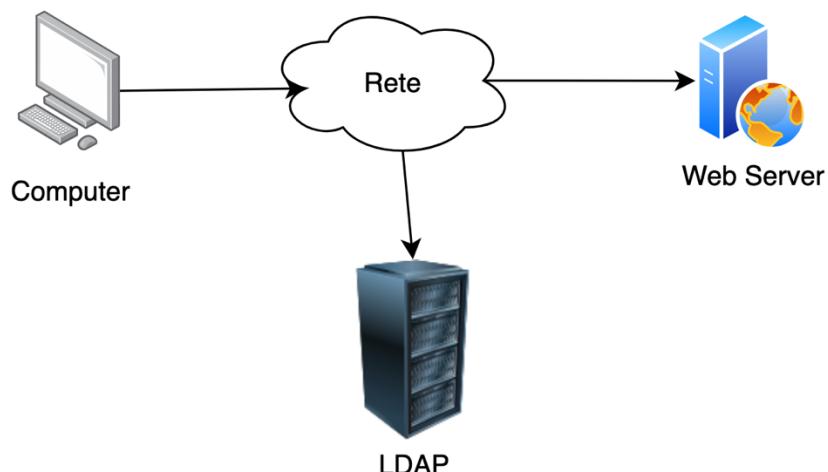


Figura 9 Schema di rete basilare.

3.2 Design dei dati e database

I dati dell'applicativo verranno salvati all'interno di un database MySQL. Verrà anche utilizzato un server LDAP, già presente nella rete scolastica, utilizzato per l'autenticazione con gli account di dominio.

3.2.1 Schema ER

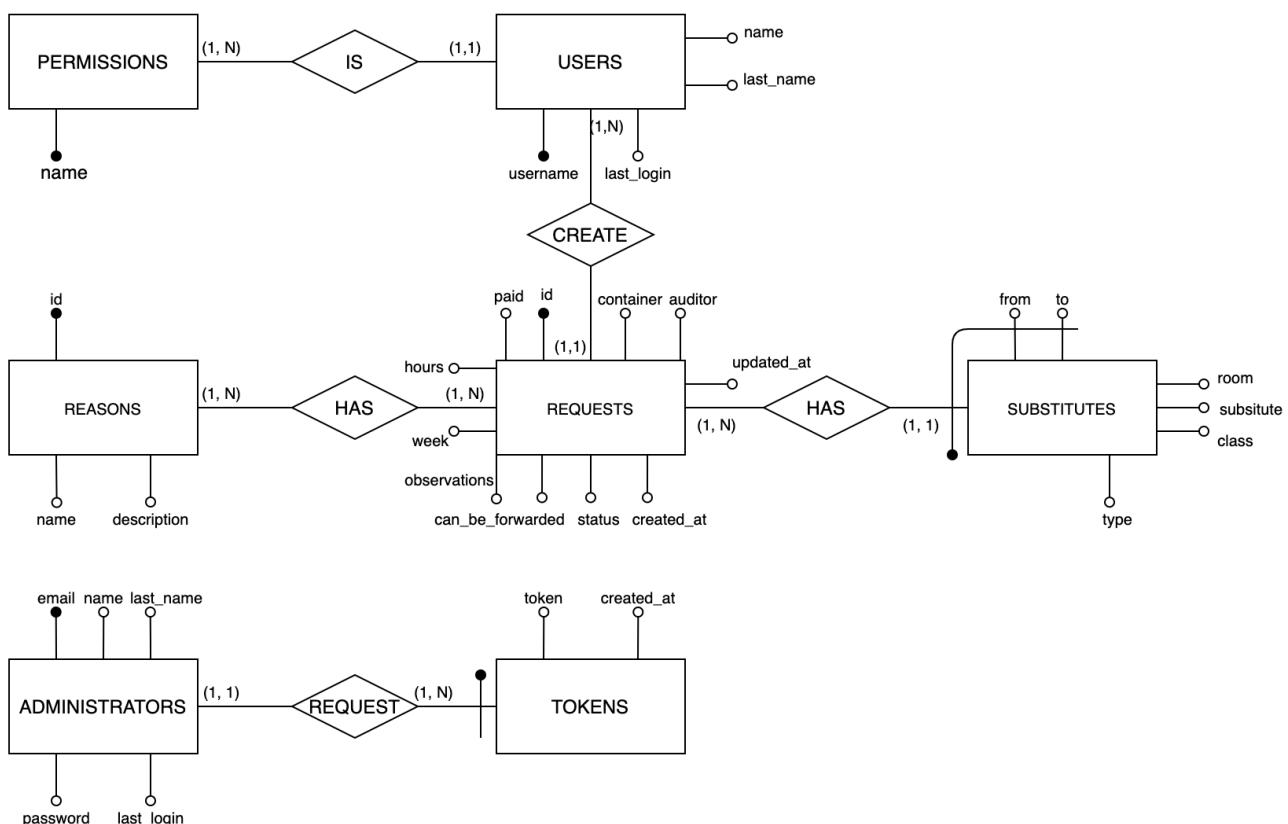


Figura 10 Schema ER banca dati.

Questo è lo schema ER utilizzato dall'applicativo web. È composto da sette tabelle principali ed una tabella ponte. La gestione degli amministratori del sito web avviene interrogando la tabella "ADMINISTRATORS" la quale contiene i dati relativi agli amministratori locali, è inoltre presente una tabella che si occupa della gestione dei token utilizzati per il recupero password degli utenti locali. La tabella "REASONS" contiene i motivi dei congedi che verranno correlati attraverso l'utilizzo di una tabella ponte con i congedi stessi salvati nella tabella "REQUESTS" la quale inoltre è collegata agli orari e possibili supplenti che vengono salvati nella tabella "SUBSTITUTES". Gli utenti LDAP vengono salvati nella tabella "USERS" al primo login, questo permette di impostare il permesso di default, ovvero il docente.

3.2.2 Descrizioni delle tabelle

PERMISSIONS	
Attributo	Descrizione
name	Rappresenta il nome di un permesso all'interno del sistema. È un attributo di tipo stringa con un limite di 30 caratteri. Non può essere nullo e deve essere univoco. Esempio: docente

USERS	
Attributo	Descrizione
username	Rappresenta l'identificativo di un utente che accede attraverso l'utilizzo di LDAP. È un attributo di tipo stringa con limite di 20 caratteri, non può essere nullo e deve essere univoco. Esempio: filippo.finke
name	Rappresenta il nome di un amministratore. 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 amministratore. 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
last_login	Rappresenta l'ultimo accesso dell'utente all'applicativo. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59

ADMINISTRATORS	
Attributo	Descrizione
email	Rappresenta un indirizzo email dell'utente. È un attributo di tipo stringa con limite di 255 caratteri. Non può essere nullo e deve essere univoco. Esempio: filippo.finke@samtrevano.ch
name	Rappresenta il nome di un amministratore. 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 amministratore. 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 amministratore. È un attributo di tipo stringa con limite di 255 caratteri. Non può essere nullo. All'interno di questo attributo verrà salvata un hash della password dell'utente. Esempio: \$2y\$10\$NmiaiLmr3dhUg3ePIExyt.l2KvE7SK6le1UH67QVikBlyBjjTHgVG
last_login	Rappresenta l'ultimo accesso dell'utente all'applicativo. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59

Gestione richieste congedi docenti

TOKENS	
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: 2020-01-29 14:16:59</p>

REASONS	
Attributo	Descrizione
id	<p>Rappresenta l'identificatore di una motivazione di congedo. 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>
name	<p>Rappresenta il nome della motivazione. È un attributo di tipo stringa, ha un limite di 255 caratteri e non può essere nullo.</p> <p>Esempio: Assenza per attività fuori sede</p>
description	<p>Rappresenta la descrizione di una motivazione. È un attributo di tipo stringa, ha un limite di 255 caratteri.</p> <p>Esempio: Assenza per un attività fuori dalla sede scolastica.</p>

REQUESTS	
Attributo	Descrizione
id	<p>Rappresenta l'identificatore di una richiesta di congedo. 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>
status	<p>Rappresenta lo stato della richiesta di congedo. È un attributo di tipo TINYINT.</p> <p>Esempio: 0</p>
container	<p>Rappresenta il contenitore nel quale si trova la richiesta. È un attributo di tipo TINYINT.</p> <p>Esempio: 0</p>
week	<p>Rappresenta la settimana nella quale è previsto il congedo. È un ENUM e può contenere "A" oppure "B"</p> <p>Esempio: A</p>
observations	<p>Rappresenta le osservazioni correlate con la richiesta di congedo. È un attributo di tipo stringa con un limite di 255 valori, può contenere qualsiasi carattere e può essere nullo.</p> <p>Esempio: Presumo che il docente X possa eseguire la supplenza.</p>
auditor	<p>Rappresenta il nome e il cognome dell'ultimo utente che ha eseguito una revisione al congedo.</p> <p>Esempio: Filippo Finke</p>

Gestione richieste congedi docenti

paid	Rappresenta se le ore di supplenza del congedo sono pagate oppure no. Esempio: 1
hours	Rappresenta le ore di congedo pagate. Esempio: 15
can_be_forwarded	Indica se una richiesta può essere mandata direttamente anche dalla segreteria. Esempio: 0
updated_at	Rappresenta l'ultimo aggiornamento della richiesta di congedo. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59
created_at	Rappresenta la data di creazione della richiesta di congedo. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59

SUBSTITUTES

Attributo	Descrizione
from_date	Rappresenta la data di inizio dell'assenza. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59
to_date	Rappresenta la data di fine dell'assenza. È un attributo di tipo DATETIME e non può essere nullo. Esempio: 2020-01-29 14:16:59
room	Rappresenta l'aula nella quale eseguire la supplenza. È un attributo di tipo stringa con limite di 5 caratteri e può essere nullo. Esempio: A417
substitute	Rappresenta il nome completo del docente di supplenza. È un attributo di tipo stringa con limite 30 caratteri e può essere nullo. Esempio: Filippo Finke
class	Rappresenta la classe da sostituire. È un attributo di tipo stringa con limite 15 caratteri e può essere nullo. Esempio: SAMT I4AC
type	Rappresenta il tipo di supplenza previsto. È un ENUM e può avere i seguenti valori: <ul style="list-style-type: none"> - SI: supplenza interna - SO: scambio d'orario - SP: sorveglianza parallela - SE: supplente esterno Esempio: SAMT I4AC

3.2.3 Schema logico

```
permissions(name)
users(username, name, last_name, permission(FK), last_login*)
reasons(id, name, description)
requests(id, username(FK), week, container, status, observations*, auditor*, paid*, hours*, can_be_forwarded,
updated_at, created_at)
substitutes(request(FK), from_date, to_date, type*, room*, substitute*, class*)
request_reason(request_id(FK), reason_id(FK))
administrators(email, name, last_name, password, last_login)
tokens(email(FK), token(UNIQUE), created_at)
```

Questo è lo schema logico del database, è presente una tabella ponte chiamata “request_reason” che permette di collegare i motivi di una richiesta di congedo ad un congedo.

3.3 Design delle interfacce

3.3.1 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 LDAP oppure il database locale. La funzionalità di recupero password verrà mostrata solamente quando l'utente sbaglierà la password e sarà un utente locale in quanto non è richiesto il reset di password anche per utenti LDAP.

Accesso

username

password

Accedi

Recupera password

Figura 11 Pagina di accesso.

3.3.2 Pagina principale

Questo è uno schizzo della pagina principale mostrata agli utenti che eseguono l'accesso attraverso LDAP. Attraverso questa pagina è quindi possibile riempire un modulo di congedo oppure navigare nei vari recipienti. Verranno mostrati solamente i recipienti relativi al permesso dell'utente corrente. Inoltre nella parte superiore destra verrà mostrato nome, cognome e il permesso dell'utente.

Gestione congedi		Nome Cognome permesso
Home In attesa (0) Miei congedi > Segreteria Da revisionare (1) Vice direzione Da revisionare (1) Storico > 	Motivazioni Calendario <input type="checkbox"/> Testo <input type="checkbox"/> Testo <input type="checkbox"/> Testo <input type="checkbox"/> Testo <input type="checkbox"/> Testo <input type="checkbox"/> Testo	

Figura 12 Pagina principale.

3.3.3 Pagina di un recipiente

Questo è lo schizzo della pagina di un recipiente, la quale sarà modificata dinamicamente in base ai recipienti. In questo caso viene mostrato un potenziale recipiente della segreteria. All'interno del recipiente sarà quindi possibile vedere tutte le richieste di congedo, con il loro stato e da chi sono state inviate.

Gestione congedi > Segreteria		Nome Cognome permesso
Home In attesa (0) Miei congedi > Segreteria Da revisionare (1) Vice direzione Da revisionare (1) Storico > 	<input type="text"/> Richiesta da Nome Cognome Data >	

Figura 13 Pagina di un recipiente.

3.3.4 Pagina storico

Questa è la pagina dello storico, all'interno di questa pagina vengono mostrati tutti i congedi registrati nel sistema. Inoltre nella parte superiore della pagina è possibile eseguire delle ricerche mirate.

Gestione congedi > Storico		Nome Cognome permesso	
Home		Ricerca	
In attesa (0)		Congedo di Nome Cognome	Approvato in data DATA Apri
Miei congedi >		Congedo di Nome Cognome	Approvato in data DATA Apri
Segreteria		Congedo di Nome Cognome	Approvato in data DATA Apri
Da revisionare (1)		Congedo di Nome Cognome	Approvato in data DATA Apri
Vice direzione		Congedo di Nome Cognome	Approvato in data DATA Apri
Da revisionare (1)			
Storico >			

Figura 14 Pagina storico.

3.3.5 Pagina amministrazione utenti

Attraverso questa pagina è possibile aggiungere utenti di tipo amministratore che potranno gestire l'applicativo web. È inoltre possibile modificare i permessi da assegnare agli utenti che eseguiranno l'accesso attraverso LDAP.

Amministrazione > Utenti		Nome Cognome permesso	
Utenti		Nuovo utente	
Congedi		Amministratori	
		Nome Cognome Permesso Azioni	
		Nome Cognome Permesso Azioni	
		Segreteria	
		Nome Cognome Permesso Azioni	
		Nome Cognome Permesso Azioni	

Figura 15 Pagina amministrazione utenti.

3.3.6 Pagina amministrazione congedi

Questa è la pagina di amministrazione della tipologia dei congedi. Attraverso questa pagina è dunque possibile modificare le motivazioni e/o cause dei congedi presenti nella pagina principale dell'applicativo.

Amministrazione > Congedi		Nome Cognome permesso
Utenti Congedi	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">Nuova tipologia</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Tipologia Azioni</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Tipologia Azioni</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Tipologia Azioni</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Tipologia Azioni</div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 2px;">Tipologia Azioni</div>	

Figura 16 Pagina amministrazione congedi.

3.4 Design dell'architettura del sistema

3.4.1 UML Controllers

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

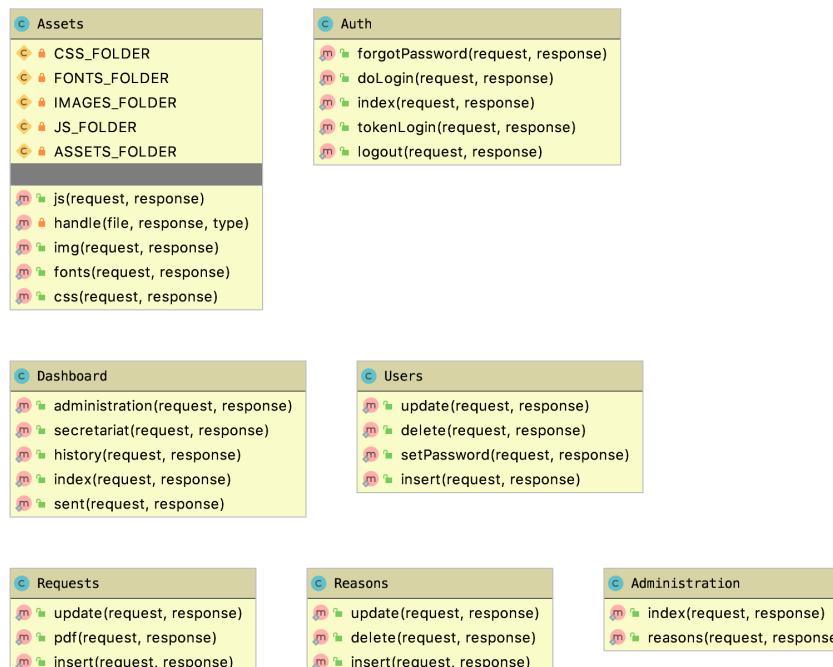


Figura 17 UML Controllers.

3.4.2 UML Libs

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

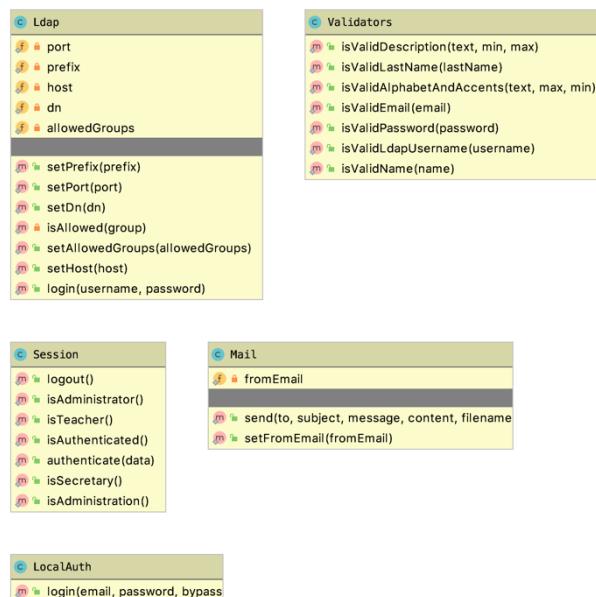


Figura 18 UML Libs.

3.4.3 UML Middlewares

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



Figura 19 UML Middlewares.

3.4.4 UML Models

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

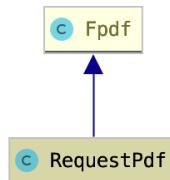


Figura 20 UML RequestsPDF.



Figura 21 UML Models.

4 Implementazione

4.1 Gestione versioni

Come gestione di tutti i file che riguardano lo sviluppo del progetto, quindi documentazione, diari, codice e altro ho utilizzato una repository GitLab che è stata messa a disposizione sul server scolastico da parte dei formatori. Ho dunque utilizzato questa repository per tenere traccia di tutti i cambiamenti all'interno del progetto, i quali sono stati caricati su GitLab con descrizioni apposite in modo da permettere di tornare avanti oppure indietro nel corso del progetto.

4.2 Gestore di pacchetti

Per lo sviluppo del progetto ho utilizzato delle librerie esterne. Per la gestione di pacchetti aggiuntivi di PHP ho utilizzato dunque Composer, il quale permette di gestire librerie esterne in modo semplificato. Ho quindi impostato Composer attraverso il file di configurazione “composer.json” nel seguente modo:

```
{  
    "name": "filippofinke/gestione-congedi",  
    "description": "Applicativo web per la gestione delle richieste di congedo  
dei docenti del CPT",  
    "type": "project",  
    "require": {  
        "filippofinke/php-rest": "dev-master"  
    },  
    "authors": [  
        {  
            "name": "Filippo Finke",  
            "email": "filippo.finke@samtrevano.ch"  
        }  
    ],  
    "minimum-stability": "dev",  
    "autoload": {  
        "psr-4": {  
            "FilippoFinke\\": "src/"  
        }  
    }  
}
```

Questo file mi permette dunque di importare la libreria “php-rest” la quale verrà utilizzata all’interno del progetto.

4.3 Database

L’account di accesso al database è stato fornito dal formatore. Il database è stato sviluppato in base allo schema ER. Le parti importanti nell’implementazione del database sono è stata la tabella ponte aggiuntiva per il collegamento della tabella “REASONS” e quella “REQUESTS”, la tabella è stata implementata nel seguente modo:

```
CREATE TABLE request_reason(  
    request INT NOT NULL,  
    reason INT NOT NULL,  
    FOREIGN KEY(request) REFERENCES requests(id),  
    FOREIGN KEY(reason) REFERENCES reasons(id)  
)
```

Questa tabella ha dunque i riferimenti alle tabelle “REASONS” e “REQUESTS”. Un altro fattore importante nell’implementazione della banca dati è stata la separazione della tabella “TOKENS” da “ADMINISTRATORS” questo per alleggerire il peso del database in quanto non tutti gli amministratori richiederanno un recupero password frequentemente. Ho dunque implementato la tabella separatamente nel seguente modo:

```
CREATE TABLE tokens (
    email VARCHAR(255) NOT NULL,
    token VARCHAR(64) UNIQUE NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
    FOREIGN KEY(email) REFERENCES administrators(email) ON UPDATE CASCADE ON
DELETE CASCADE
);
```

Nella tabella “TOKENS” è quindi presente un riferimento alla tabella “ADMINISTRATORS” attraverso la colonna email. Per eseguire la connessione al database utilizzando PHP mi sono basato sulla classe Database già presente nel framework php-rest, il codice per collegarsi è il seguente:

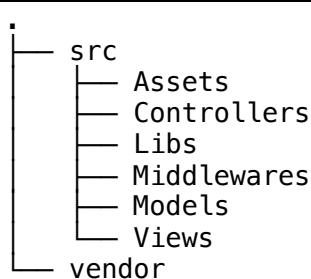
```
Database::setHost(DB_HOST);
Database::setDatabase(DB_NAME);
Database::setUsername(DB_USERNAME);
Database::setPassword(DB_PASSWORD);
$connection = Database::getConnection();
```

Vengono quindi impostati i parametri di connessione al database attraverso metodi appositi, successivamente si può ricavare la connessione da qualsiasi parte del codice con il metodo getConnection.

4.4 Applicativo Web

4.4.1 Struttura

L'applicativo è stato sviluppato con l'ausilio di php-rest e il template desk-app. La struttura del codice è la seguente:



La cartella src contiene tutto il codice dell'applicativo web. All'interno di questa cartella sono presenti ulteriori sottocartelle, la cartella Assets contiene tutte le risorse (come ad esempio i font, i fogli di stile, javascript, etc.), la cartella Controllers contiene tutte le classi che gestiscono gruppi di percorsi. La cartella Libs contiene classi utilizzate in generale all'interno del codice, come per esempio classi di gestione di sessione o di autenticazione. La cartella Middlewares contiene tutte le classi che si occupano di gestire gli accessi a determinati percorsi. All'interno di Models invece sono presenti tutte le classi utilizzate per interfacciarsi con i dati presenti nel sito web i quali verranno poi mostrati attraverso i controller nelle interfacce grafiche, le quali sono salvate nella cartella Views. È inoltre presente una cartella chiamata vendor, all'interno di questa cartella è salvato in modo automatico il codice generato da composer per la gestione e l'integrazione di librerie esterne (php-rest). L'applicativo è stato sviluppato utilizzando i pattern MVC e REST.

4.4.1.1 Model View Controller (MVC)

L'applicativo è in parte strutturato utilizzando il pattern MVC, quindi viene diviso ciò che sono i dati, dalle viste e dalla logica dei vari percorsi.

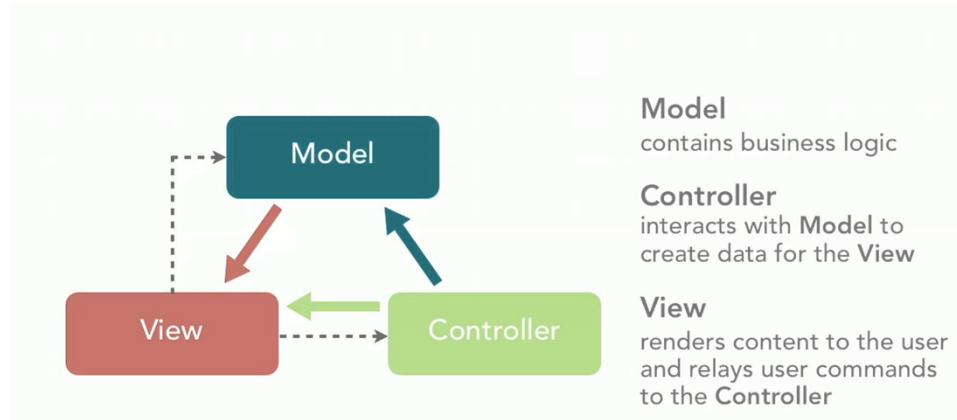


Figura 22 Schema pattern MVC.

Il pattern è diviso in tre categorie:

- Model, sono classi che si occupano di gestire tutto ciò che riguarda i dati.
- View, sono le interfacce grafiche da mostrare all'utente.
- Controller, sono classi che si occupano di collegare le interfacce grafiche con i dati.

All'interno dell'applicativo viene utilizzato MVC per quanto riguarda la gestione dei percorsi riguardanti la dashboard, i percorsi per l'gestione dell'applicativo e per la gestione delle risorse locali. Inoltre tutto quello che riguarda la gestione dei dati e quindi l'interrogazione del database è gestita da classi Model apposite. Anche tutte le interfacce grafiche dell'applicativo sono delle viste caricate dai vari Controllers.

4.4.1.2 Representational State Transfer (REST)

Una seconda parte dell'applicativo è stata sviluppata basandosi sul pattern REST. Questo in modo da permettere chiamate semplificate per l'aggiornamento, inserimento e cancellazione di dati.

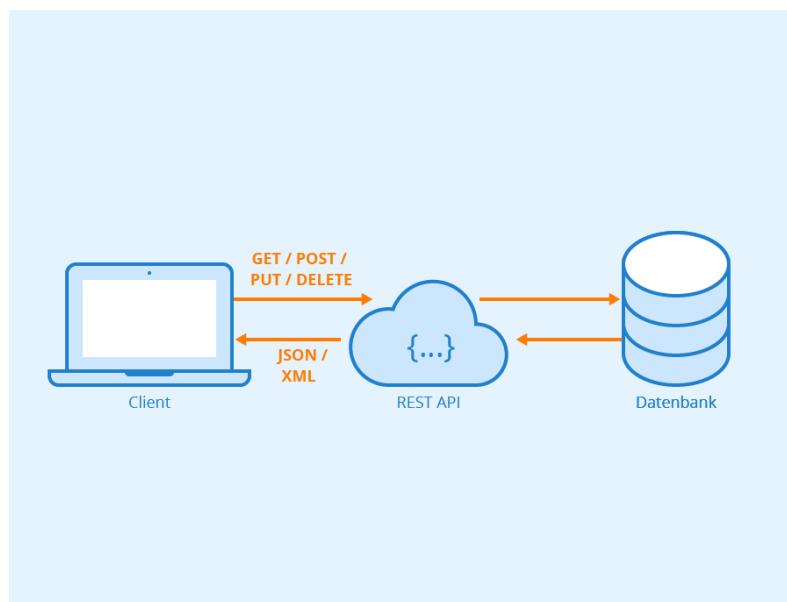


Figura 23 Schema pattern REST.

REST viene utilizzato per permettere una maggiore scalabilità dell'applicativo, permette di non essere dipendenti dalla struttura del backend in quanto le chiamate alle API REST saranno sempre le stesse. Ciò permette di avere diversi frontend in diversi linguaggi interagire con il backend. REST viene utilizzato nell'applicativo di gestione dei congedi per quanto riguarda l'aggiornamento, inserimento e cancellazione di tutti i dati presenti. Inoltre è molto utile per quanto riguarda manipolazione di dati in tempo reale eseguendo chiamate alle API REST attraverso JavaScript.

4.4.2 Inoltro richieste

L'applicativo possiede un file .htaccess il quale si occupa di inoltrare tutte le richieste che verranno eseguite da parte degli utenti ad un solo ed unico file che si occuperà di eseguirne il routing. Il file in questione è chiamato index.php ed è colui che si occupa di caricare tutte le dipendenze dell'applicativo ed inoltrare le varie richieste a classi specifiche. Il file .htaccess contiene le seguenti regole:

```
RewriteEngine On  
RewriteBase /  
RewriteRule ^(.+)$ index.php [QSA,L]
```

La prima linea del file specifica al server web di abilitare il modulo RewriteEngine il quale permette di riscrivere o eseguire inoltri delle richieste in entrata. La seconda linea specifica quando applicare le regole successive. La terza riga è la regola vera e propria che applica un regex alla richieste le quali vengono poi inoltrate al file index.php il quale si occuperà di elaborarle. L'opzione QSA viene utilizzata per permettere di eseguire l'inoltro anche di parametri GET, mentre l'opzione L dice al webserver di non eseguire altre regole se questa è stata soddisfatta.

4.4.3 Configurazione

L'applicativo web possiede un file di configurazione il quale permette di adattare dinamicamente alcune impostazioni. Attraverso questo file è possibile definire i parametri di connessione al database, l'indirizzo di posta elettronica dal quale verranno inviate le email, configurazione riguardanti l'interrogazione LDAP (server al quale collegarsi, prefisso del dominio, percorso di ricerca utenti e gruppi permessi all'accesso) e altre impostazioni per dati dinamici come per esempio orari. All'interno del sorgente è presente un file di configurazione di esempio chiamato config_sample.php, questo file dovrà essere rinominato in config.php una volta che il sistema verrà messo in produzione. Per caricare il file di configurazione all'interno dell'applicativo viene utilizzato il seguente codice:

```
// Controllo la presenza del file di configurazione.  
if (!file_exists("config.php")) {  
    exit("File di configurazione 'config.php' mancante! Puoi ricavarlo  
attraverso il file di esempio 'config_sample.php'");  
}  
// Includo del file di configurazione dell'applicativo.  
require __DIR__ . '/config.php';
```

Viene quindi controllata la presenza del file di configurazione da importare e se presente viene caricato, altrimenti la pagina stampa solamente un messaggio di informazione e poi termina l'esecuzione del codice.

4.4.4 Autenticazione

Sono presenti due metodi di autenticazione all'interno dell'applicativo di gestione dei congedi, il primo metodo il quale sarà anche il più utilizzato è attraverso le credenziali del dominio scolastico, quindi autenticazione ed accesso attraverso LDAP. Il secondo metodo di autenticazione invece verrà utilizzato per la gestione del sito web e per eseguire l'accesso verrà interrogata una banca dati locale. Per determinare quale metodo di autenticazione per verificare le credenziali immesse da parte dell'utente verrà utilizzato il nome utente, il quale per l'autenticazione attraverso il database locale dovrà essere per forza un indirizzo di posta elettronica, mentre per l'autenticazione con LDAP tutt'altro. Il codice che si occupa di richiamare il metodo di autenticazione adeguato alle credenziali dell'utente è il seguente:

```
if (Validators::isValidEmail($username)) {  
    $user = LocalAuth::login($username, $password);  
} else {  
    $user = Ldap::login($username, $password);  
}
```

Il codice in questione è presente nel controller di autenticazione chiamato Auth. Questo codice si occupa dunque di controllare se il nome utente è un indirizzo email valido, in caso di riscontro positivo verrà tentato l'accesso attraverso l'autenticazione locale in caso contrario verrà provato l'accesso attraverso LDAP.

4.4.4.1 Autenticazione banca dati locale

Per eseguire l'accesso con credenziali locali è stata sviluppata una classe apposita. Questa classe si occupa di interrogare il database MySQL e di verificare l'esistenza dell'utente che sta provando ad accedere e della validità delle credenziali immesse. All'interno di questa classe chiamata LocalAuth è presente un solo metodo il quale si occupa di eseguire l'accesso, il metodo è chiamato login ed il codice contenuto è il seguente:

```
class LocalAuth  
{  
    public static function login($email, $password, $bypass = false)  
    {  
        $user = Administrators::getByEmail($email);  
        if ($user && (password_verify($password, $user["password"]) || $bypass))  
        {  
            if ($user["last_login"] == null) {  
                $_SESSION["force_reset_password"] = true;  
            }  
            return new Administrator($email, $user["name"], $user["last_name"]);  
        }  
        return false;  
    }  
}
```

Il metodo in questione si occupa in primo luogo di verificare la presenza dell'utente nella banca dati locale eseguendo una ricerca in base all'indirizzo di posta elettronica. Se l'utente in questione esiste viene eseguito un controllo con la password inserita nella pagina di login e quella presente nel database sotto forma di hash. Se i controlli vanno a buon fine viene controllato se l'utente deve eseguire un reset della password oppure no e successivamente viene ritornato un oggetto di tipo Administrator contenente i dati dell'utente. È inoltre presente un parametro bypass il quale viene utilizzato internamente nell'applicativo web per eseguire l'accesso senza l'utilizzo di una password.

4.4.4.2 Interfaccia LDAP

Per interfacciarsi con LDAP è stata sviluppata una classe apposita. Questa classe permette di verificare che le credenziali di un utente siano corrette attraverso un server LDAP. Inoltre questa classe permette di autorizzare solamente determinati utenti. La parte più importante di questa classe riguarda la gestione dell'accesso, è quindi presente un metodo login che accetta due parametri, username e password. Il metodo di controllo dell'accesso è stato implementato in questo modo:

```
public static function login($username, $password) {
    if (EXTERNAL_LDAP) {
        $data = ExternalLdap::login($username, $password);
        if ($data && self::isAllowed($data["appartenenza"])) {
            list($name, $lastName) = explode(".", $data["username"], 2);
            return new LdapUser($username, ucfirst($name), ucfirst($lastName));
        } else {
            return false;
        }
    }
    $connectionString = ldap_connect("ldap://" . self::$host . ":" . self::$port . "/");
    ldap_set_option($connectionString, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_set_option($connectionString, LDAP_OPT_REFERRALS, 0);
    if ($connectionString) {
        $bind = @ldap_bind($connectionString, self::$prefix . $username,
$password);
        if ($bind) {
            $filter = "(sAMAccountName=" . $username . ")";
            $result = ldap_search($connectionString, self::$dn, $filter,
array());
            $entries = ldap_get_entries($connectionString, $result);
            if ($entries["count"] == 1) {
                if (($entries[0]["useraccountcontrol"][0] & 2) == 2) {
                    // Account disabilitato.
                    return false;
                }
                $name = utf8_encode($entries[0]["givenname"][0]);
                $lastName = utf8_encode($entries[0]["sn"][0]);
                $groups = $entries[0]["memberof"];
                foreach ($groups as $key => $value) {
                    if ($key != "count") {
                        if (self::isAllowed($value)) {
                            return new LdapUser($username, $name, $lastName);
                        }
                    }
                }
            }
            // L'utente non ha il permesso.
            return false;
        }
    }
}
```

```
        } else {
            // L'utente non è stato trovato.
            return false;
        }
    } else {
        // Errore username o password.
        return false;
    }
} else {
    // Errore nella stringa di connessione al server LDAP.
    return false;
}
}
```

Questo metodo si occupa dunque di validare in primo luogo che la stringa di connessione al server LDAP sia corretta, successivamente prova ad eseguire l'accesso al dominio con le credenziali che sono state passate come parametri. Se l'accesso tramite LDAP ha un esito positivo, viene ricavato l'utente corrente e ne vengono ricavati i permessi, i quali vengono successivamente controllati per stabilire se un utente può accedere oppure no all'applicativo.

4.4.4.3 Connessione server LDAP da remoto

L'applicativo oltre ad avere la possibilità di collegarsi direttamente ad un server LDAP all'interno della rete nel quale è stato installato ha la possibilità di eseguire delle query al server LDAP scolastico da remoto attraverso delle chiamate HTTP. Le chiamate a questo servizio esterno sono molto più limitate rispetto all'interrogazione di un server LDAP locale ma permettono comunque il funzionamento dell'applicativo. La chiamata HTTP al servizio esterno messo a disposizione dalla scuola richiede come parametri username e password dell'account LDAP da verificare e ne ritorna le seguenti informazioni: username, email, gruppo di appartenenza e scuola. In questo caso verrà utilizzato l'username dell'utente per ricavare nome e cognome ed il gruppo di appartenenza per determinare se un utente può accedere all'applicativo oppure no. È stata implementata dunque una ulteriore classe chiamata ExternalLDAP che si occupa di eseguire queste chiamate al servizio esterno attraverso un metodo login. Il metodo è il seguente:

```
public static function login($username, $password)
{
    if ($username && $password) {
        $cod = new EnDecryptText();
        $u=$cod->Encrypt_Text($username);
        $p=$cod->Encrypt_Text($password);
        $url = "http://server-esterno:porta/api.php?u=$u&p=$p&chi=cpt";
        parse_str(file_get_contents($url), $out);
        return $out;
    } else {
        return false;
    }
}
```

Il metodo in questione fa utilizzo della classe EnDecryptText la quale è stata fornita dal formatore per eseguire la codifica dei parametri da inviare al server il quale si occuperà di elaborare la richiesta.

4.4.4.4 Gestione delle sessioni e percorsi

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. Per esempio per impostare la sessione di un utente come autenticato è presente il metodo authenticate il quale permette di salvare dati aggiuntivi nella sessione. Il codice del metodo è il seguente:

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

Questo codice si occupa dunque di impostare il valore della chiave authenticated a true all'interno della sessione, questo indice verrà poi utilizzato per controllare se un utente ha eseguito oppure no l'accesso all'applicativo con il metodo isAuthenticated:

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

Il metodo isAuthenticated si occupa dunque di controllare la presenza dell'indice e del suo valore. Inoltre in questa classe dedicata alla gestione delle sessioni è presente un metodo per eseguire la disconnessione:

```
public static function logout()
{
    foreach ($_SESSION as $key => $value) {
        unset($_SESSION[$key]);
    }
    session_destroy();
}
```

Il metodo in questione si occupa di eliminare qualsiasi indice presente nella sessione e di distruggere definitivamente la sessione dell'utente. Sono presenti ulteriori metodi per ricavare lo stato del permesso di un utente, come per esempio il metodo isAdministrator il quale controlla se l'utente è un amministratore:

```
public static function isAdministrator()
{
    return $_SESSION["permission"] == "Administrator";
}
```

Questo metodo non fa altro che comparare l'indice del permesso dell'utente con la stringa Administrator per determinare se un utente è amministratore oppure no, sono presenti ulteriori metodi per ogni permesso all'interno della classe i quali vengono gestiti nella stessa maniera. Grazie a questi metodi è possibile dunque gestire i vari accessi alle varie pagine e percorsi presenti nell'applicativo web grazie all'utilizzo di classi o funzioni Middlewares, queste classi o funzioni permettono di eseguire del codice prima che un determinato percorso sia visitato, permettendo dunque di eseguire controlli sui permessi e decidere se un utente ha il permesso di accedere al percorso oppure no. Un esempio di Middleware è il seguente:

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

Questa classe si occupa di controllare se l'utente che sta visitando il percorso ha eseguito l'accesso all'applicativo attraverso la classe Session e il metodo isAuthenticated, se l'utente non ha eseguito l'accesso verrà rimandato alla pagina di login eseguendo un redirect, altrimenti verrà caricato il percorso normalmente. Sono presenti ulteriori classi simili le quali permettono di distinguere diverse condizioni di accesso, come per esempio verificare che un utente appartenga alla segreteria, che abbia eseguito il login con LDAP e così via. Per assegnare queste classi prima di un determinato percorso è possibile utilizzare i metodi predisposti dal framework php-rest before, per indicare di chiamare il metodo prima, oppure after, per indicare di chiamare il metodo dopo, quando viene istanziato un percorso. Per esempio:

```
// Pagina di visione e modifica congedi.
$router->get(
    '/dashboard/{id:[0-9]+}',
    'FilippoFinke\Controllers\Dashboard::index'
)
// Controllo che l'utente appartenga alla segreteria.
->before(new SecretaryRequired()),
```

In questo caso viene assegnato il controllo che l'utente appartenga alla segreteria per il percorso di visione e modifica dei congedi. I percorsi dell'applicativo vengono aggiunti attraverso l'oggetto Router il quale si occupa di eseguire la presenza di essi e quali funzioni o classi chiamare.

```
// Oggetto nel quale verranno salvati tutti i percorsi dell'applicativo.
$router = new Router();
```

Attraverso questo oggetto è quindi possibile aggiungere tutti i percorsi al quale l'applicativo dovrà rispondere:

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

L'oggetto Router supporta percorsi di qualsiasi tipologia di richiesta, come per esempio: GET, POST, PUT, DELETE, etc. È possibile inoltre creare dei gruppi di percorsi:

```
// Gruppo di percorsi di gestione.  
$adminRoutes = new RouteGroup();  
// Aggiunta dei percorsi al gruppo.  
$adminRoutes->add(  
    // Percorso per inserire una motivazione.  
    $router->post('/reasons', 'FilippoFinke\Controllers\Reasons::insert'),  
    ...  
    // Percorso per l'eliminazione di un utente.  
    $router->delete('/users', 'FilippoFinke\Controllers\Users::delete')  
)  
// Aggiunta controllo autenticazione.  
->before(new AuthRequired())  
// Aggiunta controllo permesso amministratore.  
->before(new AdministratorRequired());
```

Come ad esempio in questo caso nell'applicativo viene creato un gruppo dedicato ai percorsi di gestione di esso in modo da applicare i controlli di permessi al gruppo stesso.

4.4.5 Interfaccia invio email

Per rendere più semplice e veloce l'invio di email all'interno dell'applicativo è stata sviluppata una classe dedicata. La classe in questione si chiama "Mail", è presente un solo metodo chiamato send. Il metodo in questione si occupa di inviare un messaggio di posta elettronica ad un indirizzo email con un soggetto e un contenuto. Opzionalmente la classe ha anche la possibilità di inviare degli allegati. Il metodo è stato implementato nel seguente modo:

```
public static function send($to, $subject, $message, $file = null, $filename = null)  
{  
    $separator = md5(time());  
    $eol = "\r\n";  
    $headers = "From: name <".self::$fromEmail.">" . $eol;  
    $headers .= "MIME-Version: 1.0" . $eol;  
    $headers .= "Content-Type: multipart/mixed; boundary=\"$separator\" . \"$\""  
    . $eol;  
    $headers .= "Content-Transfer-Encoding: 7bit" . $eol;  
    $headers .= "This is a MIME encoded message." . $eol;  
  
    $body = "--" . $separator . $eol;  
    $body .= "Content-Type: text/html; charset=\"utf-8\" . $eol . $eol;  
    $body .= $message . $eol . $eol;  
  
    if ($file && $filename) {  
        $content = chunk_split(base64_encode($file));  
    }  
}
```

```
$body .= "--" . $separator . $eol;
$body .= "Content-Type: application/octet-stream; name=\"" . $filename .
"\\" . $eol;
$body .= "Content-Transfer-Encoding: base64" . $eol;
$body .= "Content-Disposition: attachment" . $eol;
$body .= $content . $eol . $eol;
$body .= "--" . $separator . "--";
}

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

Il metodo si occupa di generare dinamicamente gli header dello standard relativo alla posta elettronica in base ai parametri che sono stati selezionati nella chiamata del metodo. In questo modo viene costruito il contenuto dell'email il quale può essere inviato. Grazie a questo classe basterà chiamare questo metodo statico da qualsiasi altra parte del codice per inviare una email.

4.4.6 Validazione campi

La validazione dei campi è indispensabile per non avere delle incongruenze con i dati che verranno salvati all'interno della banca dati. I controlli di validazione e risanamento dei dati vengono eseguiti sia lato client, quindi da parte del browser, sia lato server. Nella parte lato server sono presenti dei metodi di verifica scritti in PHP all'interno della classe Validators la quale racchiude tutti i metodi di questo tipo. Ad esempio per verificare che la descrizione o le osservazioni di un congedo siano valide è presente il metodo isValidDescription il quale è stato implementato nel seguente modo:

```
public static function isValidDescription($text, $min = 1, $max = 255)
{
    $safe = htmlspecialchars($text);
    return strlen($text) >= $min && strlen($text) <= $max && $safe == $text;
}
```

Il metodo si occupa dunque di prevenire attacchi di tipo XSS e controllare che il testo abbia una lunghezza tra il minimo e il massimo specificati come parametro. Lo stesso metodo è presente sotto forma di funzione per il client, il codice è stato portato su JavaScript:

```
function isValidDescription(text, min = 1, max = 255) {
    return text.length >= min && text.length <= max;
}
```

Ovviamente sono presenti ulteriori metodi di validazione, come per esempio la validazione del nome, del cognome, dell'email e molti altri. Per ogni metodo di verifica lato server scritto in PHP è presente un metodo trascritto in JavaScript per la parte lato client, vengono eseguiti controlli anche lato client per diminuire il carico del server.

4.4.7 Risorse locali

L'applicativo ha bisogno dell'utilizzo di risorse locali scritte in CSS, JavaScript, font ed immagini per funzionare correttamente, le risorse utilizzate si trovano all'interno della cartella Assets nel codice sorgente. Per servire queste risorse agli utenti che visitano il sito web è presente un controller che si occupa della gestione di esse. Sono registrati quattro percorsi relativi alle risorse nel router delle richieste, i percorsi sono i seguenti:

```
// Percorso per file javascript.  
$router->get('/assets/js/{asset}', 'FilippoFinke\Controllers\Assets::js');  
// Percorso per file css.  
$router->get('/assets/css/{asset}', 'FilippoFinke\Controllers\Assets::css');  
// Percorso per i font.  
$router->get('/assets/fonts/{asset}', 'FilippoFinke\Controllers\Assets::fonts');  
// Percorso per le immagini.  
$router->get('/assets/img/{asset}', 'FilippoFinke\Controllers\Assets::img');
```

Il controller che si occupa delle risorse è anch'esso chiamato Assets e si trova nella cartella Controllers, questa classe si occupa di recuperare le risorse locali, verificarne l'esistenza e servirle all'utente. All'interno del Controller sono definite le costanti che rappresentano i percorsi dal quale ricavare i dati da servire, ad esempio la cartella base e le risorse relative a JavaScript sono scritte nel seguente modo:

```
// Cartella base delle risorse.  
private const ASSETS_FOLDER = __DIR__ . '/../Assets';  
// Cartella per i file JavaScript.  
private const JS_FOLDER = self::ASSETS_FOLDER . '/js/';
```

Questo permette di rendere dinamica la ricerca delle risorse qualora si cambi cartella principale. Per ogni tipologia di risorsa è presente un metodo che si occupa di ricavare quale file è stato richiesto e la sua tipologia, ad esempio per le risorse di tipo CSS il metodo chiamato è il seguente:

```
public static function css($request, $response)  
{  
    $asset = $request->getAttribute('asset');  
    return self::handle(self::CSS_FOLDER.$asset, $response, 'text/css');  
}
```

Questo metodo si occupa principalmente di definire il tipo di dato che sarà dato in risposta alla richiesta dell'utente. Il tutto viene delegato ad un metodo ulteriore chiamato handle il quale si occupa di gestire tutti i controlli e di servire il dato. Il metodo in questione è implementato nel seguente modo:

```
private static function handle($file, $response, $type = null)  
{  
    $must = realpath(self::ASSETS_FOLDER);  
    $realPath = realpath($file);  
  
    if (strpos($realPath, $must) !== false && file_exists($file)) {  
        if (!$type) {  
            $type = mime_content_type($file);  
        }  
    }  
}
```

```
        return $response
        ->withHeader("Cache-Control", "public, max-age=3600")
        ->withHeader("Expires", gmdate('D, d M Y H:i:s', time() + 3600) . 'GMT')
        ->withHeader("Content-Type", $type)
        ->withBody(file_get_contents($file))
        ->withStatus(200);
    } else {
        return $response->withStatus(404)->withText("Not found");
    }
}
```

Il metodo handle si occupa di ricavare il percorso reale della cartella base di tutte le risorse e lo confronta con il file che è stato richiesto, questo per prevenire attacchi di tipo Directory Traversal. Inoltre viene controllata l'esistenza del file richiesto, se presente viene controllato il tipo di file da servire nel caso non sia stato specificato nella chiamata del metodo e successivamente ne viene letto il contenuto il quale viene inviato all'utente in risposta alla richiesta. In caso la risorsa richiesta non sia disponibile, viene ritornata una risposta segnalando che il file non è stato trovato (404). Inoltre vengono inviati anche degli header per permettere la cache dei file da parte degli utenti che accedono all'applicativo web il quale aumenterà la velocità di navigazione notevolmente.

4.4.8 Calendario

L'applicativo dove essere più simile possibile al sistema corrente cartaceo. Al momento viene utilizzato un calendario su un foglio di carta per selezionare le date del congedo, viene suddiviso in giorni sull'asse delle ordinate mentre gli orario disponibili sull'asse delle ascisse. All'interno di questo calendario devono venire salvati i giorni di congedo con ulteriori dati come: aula, classe, possibile supplente e tipologia di supplenza. Eseguendo diverse ricerche non è stata trovata nessuna libreria che soddisfacesse questi requisiti. Per questo motivo ho deciso di implementare una libreria per calendari totalmente da zero, la libreria si chiama FinkelLendar ed è stata scritta utilizzando JavaScript. La libreria possiede un metodo costruttore il quale accetta parametri per modificare in modo dinamico il calendario, il metodo accetta un elemento nel quale eseguire il render del calendario, i nomi da assegnare alle varie righe e gli orari da stampare nelle colonne:

```
constructor(element, labels, hours) {
    this.element = element;
    this.labels = labels;
    this.days = [];
    this.dates = [];
    for (var i = 0; i < labels.length; i++) {
        this.days.push([]);
    }
    this.hours = hours;
    this.selecting = false;
    this.currentSelection = [];

    this.week = "";
}
```

Questo metodo inoltre si occupa di resettare tutti gli stati del calendario, come per esempio lo stato di selezione. Per istanziare il calendario si necessiterà dunque di un contenitore HTML:

```
<div class="calendar" id="calendar"></div>
```

Successivamente basterà creare un oggetto di tipo FinkeLendar per eseguire il rendering, in questo caso i labels e gli orari vengono caricati direttamente da PHP.

```
var calendar = new FinkeLendar(
    document.getElementById('calendar'),
    labels,
    hours
);
```

Una volta istanziato il calendario sarà possibile assegnare tutti gli eventi utili alla gestione di esso, come ad esempio quando viene eseguita una selezione di date:

```
calendar.setOnSelected(function(event) {
    toEdit = event.target;
    $('#calendar-modal').modal('toggle');
});
```

In questo caso quando una o più date verranno selezionate verrà aperto un modale per modificare i dati. Una volta assegnati tutti gli eventi basterà creare il calendario:

```
calendar.draw();
```

L'assegnazione degli eventi all'interno della classe FinkeLendar funziona nel seguente modo, viene passata una funzione come parametro ad un metodo interno che si occupa di salvare in un'attributo locale la funzione da richiamare quando lanciato un determinato evento, in questo caso viene assegnato il callback per quando viene cliccato un blocco di un ora:

```
setOnHourClick(callback) {
    this.onHourClick = callback;
}
```

Questo secondo metodo esegue la stessa cosa, però per quando vengono selezionati uno o più blocchi di orari:

```
setOnSelected(callback) {
    this.onSelected = callback;
}
```

La gestione della selezione viene eseguita dal metodo seguente che si occupa di aggiornare lo stato del calendario quando viene premuto il mouse all'interno di esso e notificando su quale blocco è stato il puntatore:

```
onCalendarPress(event) {
    this.selecting = true;
    this.onCalendarHover(event);
}
```

Il metodo onCalendarHover si occupa dunque di aggiornare lo stato dei blocchi sui quali si è passati con il puntatore tenendo premuto il tasto sinistro del mouse, si occupa di aggiungere ad una coda interna i vari blocchi selezionati se essi non lo sono già stati in precedenza, aggiornare alcuni attributi come ad esempio il colore di sfondo:

```
onCalendarHover(event, bypass = false) {  
    var e = event.target;  
    var day = e.dataset.day;  
    if (  
        (this.selecting  
        && !this.isSelected(e)) || bypass  
    ) {  
        e.setAttribute("data-selected", "true");  
        this.currentSelection.push(e);  
        e.style.background = "#dffe";  
        this.days[day].push(e);  
    }  
}
```

L'ultimo metodo legato agli eventi è colui che si occupa di aggiornare gli stati quando viene rilasciato il tasto del mouse e quindi è terminata la selezione, il metodo si occupa dunque di aggiornare lo stato della selezione corrente, di riordinare tutti gli elementi selezionati, di renderizzare il calendario e di notificare il click di un blocco:

```
onCalendarRelease(event) {  
    var e = event.target;  
    this.selecting = false;  
    if (e.dataset.selected == "false") {  
        this.onCalendarHover(event);  
        if (this.onSelected) {  
            this.onSelected(event);  
        }  
        this.reorder();  
        this.render();  
    } else if (this.onHourClick) {  
        this.onHourClick(event);  
    }  
}
```

Un altro metodo per la gestione dello stato del calendario è colui che si occupa di definire la tipologia di settimana nel quale ci si trova, questo metodo non fa altro che impostare degli attributi interni con il valore della settimana:

```
setWeek(week) {  
    this.week = week;  
    this.select.value = week;  
}
```

Per evitare che vengano selezionate le stesse date molteplici volte è presente un metodo chiamato isSelected che si occupa di eseguire questo controllo su un elemento il quale viene confrontato con gli elementi già salvati in precedenza e determina se è già stato selezionato oppure no:

```
isSelected(element) {
    for (var day = 0; day < this.days.length; day++) {
        if (this.days[day].indexOf(element) != -1) {
            return true;
        }
    }
    return false;
}
```

Quando vi è bisogno di aggiornare lo stato visivo del calendario ci sono tre metodi, il primo metodo draw si occupa di generare la struttura grezza di tutto l'html del calendario all'interno del contenitore definito nel metodo costruttore, il metodo in questione è molto lungo quindi ne è stata documentata una piccola parte che si occupa della creazione basilare della struttura HTML senza tutti i dettagli:

```
draw() {
    this.element.innerHTML = "";
    // Creazione struttura header
    var header = document.createElement("div");
    ...
    var spacer = document.createElement("div");
    ...
    var btn = document.createElement("button");
    ...
    this.select = document.createElement("select");
    ...
    spacer.append(btn);
    spacer.append(this.select);
    header.append(spacer);
    // Creazione colonne header
    for (var i = 0; i < this.hours.length; i++) {
        ...
    }
    this.element.append(header);
    ...
    // Creazione struttura a griglia
    for (var i = 0; i < this.labels.length; i++) {
        ...
    }
}
```

Il metodo reorder viene chiamato dopo ogni selezione di orari per permettere di avere una sequenza sempre in ordine cronologico delle selezioni, il metodo si occupa dunque di comparare le date di inizio e di fine dei vari blocchi ed in base a quelle vengono ordinati cronologicamente.

```
reorder() {
    for (var day = 0; day < this.days.length; day++) {
        this.days[day].sort(function (a, b) {
            return Date.parse("1970-01-01 " + a.dataset.start) - Date.parse("1970-
01-01 " + b.dataset.start);
        });
    }
}
```

Il metodo render è colui che si occupa di aggiornare la grafica e di unire i blocchi che contengono gli stessi dati e si trovano vicini tra di loro, questo metodo è molto importante per l'esperienza utente:

```
render() {
    // Selezioni per ogni riga.
    for (var day = 0; day < this.days.length; day++) {
        // Reset variabili di controllo.
        var lastStart = null;
        var lastEnd = null;
        var lastElement = null;
        var elements = 0;
        var dayHours = [];
        // Copia dell'array locale.
        for (var i = 0; i < this.days[day].length; i++) {
            dayHours.push(this.days[day][i]);
        }
        // Ogni blocco di orario selezionato.
        for (var i = 0; i < dayHours.length; i++) {
            var element = dayHours[i];
            var start = element.dataset.start;
            var end = element.dataset.end;
            // Comparazione inizio e fine degli orari per determinare se unire o no.
            if (lastEnd == start && lastElement.innerText == element.innerText) {
                lastEnd = end;
                lastElement.style.background = "#defffe";
                lastElement.setAttribute("data-end", lastEnd);
                elements += 1;
                // Unisce i due elementi
                lastElement.className = "col-" + elements + " calendar-box";
                element.remove();
                var index = this.days[day].indexOf(element);
                if (index != -1) {
                    this.days[day].splice(index, 1);
                }
            }
        }
    }
}
```

```
        }
    } else {
        // Non unire nulla.
        lastStart = start;
        lastEnd = end;
        lastElement = element;
        elements = 1;
    }
}
}
```

4.4.9 Gestione dati e interrogazione banca dati

Per ricavare i dati presenti nella banca dati sono presenti delle classi apposite chiamate Model. Queste classi si occupano di eseguire una mappatura con la banca dati dell'applicativo in modo da disporre metodi che potranno essere utilizzati da altre classi. Per ogni tabella presente nel database è presente una sua classe dedicata con la logica che si occupa di gestirla. Tutte le classi contengono metodi che si interfacciano attraverso l'utilizzo di una connessione al database attraverso PDO e con l'utilizzo di Prepared Statements per prevenire attacchi di tipo SQL Injection.

4.4.9.1 Contenitori

L'applicativo ha come requisito di possedere dei contenitori nel quali verranno inserite le differenti richieste di congedo, per la gestione dei contenitori viene utilizzata la classe Container la quale contiene delle costanti che rappresentano i vari contenitori, questo perché all'interno del database ne viene salvato solamente un identificativo. Sono presenti due contenitori, il contenitore relativo alla segreteria e quello della direzione. I due contenitori hanno i seguenti valori nel codice:

```
// Contenitore della segreteria.
public const SECRETARY = 0;

// Contenitore di amministrazione.
public const ADMINISTRATION = 1;
```

All'interno della classe Container è presente un metodo che permette di ricavare la stringa del contenitore attraverso il suo identificativo, il metodo accetta come parametro l'identificativo di un contenitore e ne ritorna il nome:

```
public static function get($container) {
    switch ($container) {
        case self::SECRETARY:
            return 'segreteria';
        case self::ADMINISTRATION:
            return 'direzione';
        default:
            return null;
    }
}
```

4.4.9.2 Tabella administrators

Per quanto riguarda la tabella administrators sono presenti due classi di gestione, la prima classe si chiama Administrator mentre la seconda Administrators. La classe Administrator è utilizzata per rappresentare un singolo amministratore con i suoi relativi dati: email, nome, cognome e permesso. Essa contiene solamente metodi getter e setter per le proprietà descritte in precedenza ed in più un metodo che si occupa di eseguire l'aggiornamento dell'ultimo accesso dell'utente in questione. Il metodo è il seguente:

```
public function updateLastLogin()
{
    $pdo = Database::getConnection();
    $query = "UPDATE administrators SET last_login = CURRENT_TIMESTAMP WHERE
email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':email', $this->username);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

Il metodo updateLastLogin si occupa dunque di aggiornare l'ultimo accesso dell'utente corrente che viene rappresentato da un oggetto di tipo Administrator. La classe Administrators invece viene utilizzata per gestire tutti gli amministratori, sono quindi presenti metodi di ricerca, inserimento ed eliminazione. Il primo metodo presente nella classe è chiamato getAll e si occupa di ritornare tutti gli amministratori presenti nel sistema, questo viene utilizzato principalmente nelle viste di gestione del software. Il metodo contiene il seguente codice:

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

Un altro metodo utilizzato per ricavare le informazioni di un singolo amministratore è il metodo `getByEmail` il quale si occupa di ricavare tutti i dati riguardanti un amministratore in base alla sua email:

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

Successivamente è presente anche un metodo per la cancellazione di un profilo di amministratore. Il metodo `delete` accetta come parametro una email dell'account amministratore da cancellare:

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

Un altro metodo molto importante è quello di inserimento, il metodo in questione si occupa di inserire un nuovo amministratore ed inviare una email con la password per l'attivazione dell'account.

```
public static function insert($name, $lastName, $email)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO administrators(name, last_name, email, password)
VALUES (:name, :last_name, :email, :password)";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':name', $name);
    $stm->bindParam(':last_name', $lastName);
    $stm->bindParam(':email', $email);
    $password = self::randomPassword();
```

```
$hash = password_hash($password, PASSWORD_DEFAULT);
$stmt->bindParam(':password', $hash);
try {
    $content = "contenuto email";
    return $stmt->execute() && Mail::send($email, 'Titolo email', $content);
} catch (PDOException $e) {
    return false;
}
}
```

Per la generazione della password temporanea viene utilizzato lo stesso metodo della generazione dei token di recupero password, il quale permette di generare una stringa randomica di 20 caratteri che andrà utilizzata come password di attivazione:

```
private static function randomPassword()
{
    $keyspace =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $pieces = [];
    $max = mb_strlen($keyspace, '8bit') - 1;
    for ($i = 0; $i < 20; ++$i) {
        $pieces []= $keyspace[random_int(0, $max)];
    }
    return implode('', $pieces);
}
```

Per finire è presente un metodo utilizzato dalla funzione di recupero password il quale si occupa di impostare la password ad un account amministratore attraverso la sua email, il metodo in questione accetta come parametri l'email dell'amministratore e una password in chiaro dalla quale verrà creata l'hash da salvare nel database:

```
public static function setPassword($email, $password)
{
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $pdo = Database::getConnection();
    $query = "UPDATE administrators SET password = :password WHERE email = :email";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':email', $email);
    $stmt->bindParam(':password', $hash);
    try {
        return $stmt->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

4.4.9.3 Tabella users

La tabella users contiene le informazioni riguardanti gli utenti LDAP. La struttura dei Models per questa tabella è molto simile a quella degli amministratori descritta in precedenza. Anche per questa tabella sono presenti due classi: LdapUser e LdapUsers, le quali rispettivamente rappresentano un singolo utente oppure mettono a disposizione metodi per ricerca, modifica ed inserimento di utenti. Nella classe LdapUser è presente anche in questo caso un metodo che si occupa di aggiornare l'ultima data di accesso:

```
public function updateLastLogin()
{
    $pdo = Database::getConnection();
    $query = "UPDATE users SET last_login = CURRENT_TIMESTAMP WHERE username = :username";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':username', $this->username);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

Un altro metodo che viene richiamato in automatico alla creazione di un oggetto di LdapUser è setPermission, esso si occupa di ricavare il permesso dell'utente LDAP corrente dal database e salvarlo all'interno di un attributo dedicato al proprio permesso, se non viene trovato nessun permesso l'utente è inesistente e viene quindi creato con il permesso predefinito di docente:

```
private function setPermission()
{
    $data = LdapUsers::getByUsername($this->username);
    if ($data) {
        $this->permission = $data["permission"];
    } else {
        LdapUsers::insert($this->username, $this->name, $this->lastName);
        $this->permission = "Docente";
    }
}
```

La seconda classe chiamata LdapUsers contiene i metodi per ricavare tutti gli utenti, ricavarli per nome utente LDAP, inserimento ed aggiornamento. Per ricavare tutti gli utenti è presente il metodo getAll:

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

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

Per ricavare un utente dal suo nome utente è presente il metodo getByUsername che accetta come parametro un nome utente da cercare:

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

Per l'inserimento di un utente è presente il metodo insert che accetta un username, un nome ed un cognome:

```
public static function insert($username, $name, $lastName)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO users(username, name, last_name) VALUES(:username,
:name, :last_name)";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':username', $username);
    $stm->bindParam(':name', $name);
    $stm->bindParam(':last_name', $lastName);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

In fine è presente un metodo per eseguire l'aggiornamento del permesso di un utente chiamato setPermission il quale accetta un username ed il permesso da assegnare.

```
public static function setPermission($username, $permission)
{
    $pdo = Database::getConnection();
    $query = "UPDATE users SET permission = :permission WHERE username =
$username";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':username', $username);
    $stm->bindParam(':permission', $permission);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

4.4.9.4 Tabella permissions

Per la tabella della banca dati permissions è presente solamente un metodo che permette di ricavare tutti i permessi presenti nell'applicativo web:

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

4.4.9.5 Tabella reasons

Come per le altre classi Model importanti anche per la tabella reasons è presente una classe chiamata Reasons la quale contiene metodi per ricerca, aggiornamento, inserimento ed eliminazione. Il primo metodo viene utilizzato per ricavare tutti i motivi presenti nel database ed è chiamato getAll:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM reasons";
    try {
        $stm = $pdo->query($query);
```

```
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        return false;
}
}
```

Il secondo metodo di ricerca presente nella classe permette di eseguire una ricerca tramite identificativo, il metodo si chiama getRequestById:

```
public static function getByRequestId($id)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM reasons WHERE id IN (SELECT reason FROM
request_reason WHERE request = :id)";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":id", $id);
    try {
        $stmt->execute();
        return $stmt->fetchAll(\PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        return false;
    }
}
```

Per inserire una motivazione invece è presente il metodo insert che accetta un titolo ed una descrizione:

```
public static function insert($name, $description)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO reasons(name, description) VALUES (:name,
:description)";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':description', $description);
    try {
        return $stmt->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

Per eliminare una motivazione è presente il metodo delete che accetta come parametro l'identificativo della motivazione da eliminare dal database:

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

Per eseguire l'aggiornamento dei dati di una motivazione è presente il metodo update che permette di modificare il nome e la descrizione in base all'identificativo di essa:

```
public static function update($id, $name, $description)
{
    $pdo = Database::getConnection();
    $query = "UPDATE reasons SET name = :name, description = :description WHERE
id = :id";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':id', $id);
    $stm->bindParam(':name', $name);
    $stm->bindParam(':description', $description);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

Un metodo molto importante all'interno della classe è colui che si occupa di creare un collegamento tra le motivazioni di assenza e le richieste di congedo. Il metodo si chiama connect e permette, attraverso la tabella ponte presente nel database, di collegare una motivazione ad una richiesta di congedo. Chiamando più volte questo metodo è possibile collegare più motivazioni ad una singola richiesta di congedo. Il metodo accetta come primo parametro l'identificativo della motivazione da collegare ad una richiesta di congedo e come secondo parametro l'id della richiesta di congedo al quale collegare la motivazione. Il codice è il seguente:

```
public static function connect($reason_id, $request_id)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO request_reason VALUES(:request_id, :reason_id)";
    $stm = $pdo->prepare($query);
```

```
$stm->bindParam(":request_id", $request_id, \PDO::PARAM_INT);
$stmt->bindParam(":reason_id", $reason_id, \PDO::PARAM_INT);
try {
    return $stmt->execute();
} catch (PDOException $e) {
    return false;
}
}
```

4.4.9.6 Tabella requests

La tabella requests è la più importante dell'applicativo, anche per essa è presente una classe Model chiamata Requests. La classe in questione è quella più grande e con più funzioni tra tutte le classi del trattamento dei dati. All'interno di questa classe sono presenti tutti i metodi adeguati per la ricerca tra i contenitori, l'inserimento, modifica ed eliminazione di richieste di congedo. Il primo metodo riguarda la ricerca di tutte le richieste di congedo in attesa di un utente, le quali verranno poi visualizzate nella scheda in uscita all'interno dell'applicativo. Per eseguire una chiamata a questo metodo è richiesto il nome utente al quale devono appartenere le richieste di congedo, il codice è il seguente:

```
public static function getWaitingByUsername($username)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM requests WHERE username = :username AND status =
:status";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":username", $username);
    $stmt->bindValue(":status", RequestStatus::WAITING);
    try {
        $stmt->execute();
        return $stmt->fetchAll(\PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
    }
    return false;
}
```

Il prossimo metodo presente nella classe permette di ricavare lo storico dei congedi creati da un utente e che sono stati revisionati, quindi i congedi che non sono più in attesa. Il codice accetta dunque un parametro che corrisponde al nome utente LDAP dal quale ricavare i congedi:

```
public static function getPersonalHistory($username)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM requests WHERE status <> :status AND username =
$username ORDER BY updated_at DESC";
    $stmt = $pdo->prepare($query);
    $stmt->bindValue(":status", RequestStatus::WAITING);
    $stmt->bindValue(":username", $username);
    try {
```

```
$stm->execute();
return $stm->fetchAll(\PDO::FETCH_ASSOC);
} catch (\PDOException $e) {
}
return false;
}
```

Per ricavare tutti i congedi che sono stati revisionati, quindi non in attesa, è presente il metodo getAll simile alle altre classi Model il quale si occupa di recuperare tutti i congedi dal database:

```
public static function getAll()
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM requests WHERE status <> :status ORDER BY updated_at DESC";
    $stm = $pdo->prepare($query);
    $stm->bindValue(":status", RequestStatus::WAITING);
    try {
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
    }
    return false;
}
```

È presente un altro metodo dedicato alla ricerca di congedi all'interno del database, esso è chiamato getByStatusAndContainer e si occupa di ricavare i congedi dal database in base al loro stato e al loro contenitore. Per definire lo stato e il contenitore della ricerca sono utilizzate le classi Model dedicate (RequestStatus e Container). Il codice per eseguire questo tipo di ricerca è il seguente:

```
public static function getByStatusAndContainer($status, $container)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM requests WHERE status = :status AND container = :container ORDER BY created_at DESC";
    $stm = $pdo->prepare($query);
    $stm->bindParam(":status", $status);
    $stm->bindValue(":container", $container);
    try {
        $stm->execute();
        return $stm->fetchAll(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
    }
    return false;
}
```

Un metodo simile ai precedenti è getById il quale permette di ricavare una singola richiesta di congedo attraverso il proprio identificativo:

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

Per eseguire aggiornamenti alle richieste di congedo sono disponibili due metodi, il primo setContainer ed il secondo update. Il metodo setContainer è utilizzato per aggiornare solamente il contenitore nel quale risiede una richiesta di congedo, l'implementazione del metodo è la seguente:

```
public static function setContainer($id, $container)
{
    $pdo = Database::getConnection();
    $query = "UPDATE requests SET container = :container WHERE id = :id";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":container", $container);
    $stmt->bindParam(":id", $id);
    try {
        return $stmt->execute();
    } catch (PDOException $e) {
    }
    return false;
}
```

Il metodo update invece viene utilizzato per aggiornare tutti i parametri relativi ad una richiesta di congedo i quali sono: la settimana, lo stato del congedo, le osservazioni, se è stato pagato oppure no e le ore riconosciute. Il metodo in questione è più complesso degli altri in quanto è dinamico e permette di aggiornare la query da eseguire in base ai parametri specificati. Inoltre aggiorna in modo automatico la colonna che indica il nome di chi ha revisionato per ultimo un congedo. Il codice del metodo è stato implementato nel seguente modo:

```
public static function update($id, $week = null, $status = null, $observations = null, $paid = null, $hours = null)
{
    $pdo = Database::getConnection();
    $toSet = "";
    // Per ogni parametro viene eseguito il controllo.
```

```
if ($week) {
    $toAdd = "week = :week";
    if ($toSet == "") {
        $toSet = $toAdd;
    } else {
        $toSet .= ", ".$toAdd;
    }
}

...
$query = "UPDATE requests SET auditor = :auditor, $toSet WHERE id = :id";
$stmt = $pdo->prepare($query);
$stmt->bindParam(":id", $id);
$stmt->bindValue(":auditor", $_SESSION["name"]." ".$_SESSION["lastName"]);
// Per ogni parametro viene assegnato il valore.
if ($week) {
    $stmt->bindParam(":week", $week);
}
...
try {
    return $stmt->execute();
} catch (PDOException $e) {
}
return false;
}
```

Dopo la modifica dei congedi è presente un metodo dedicato all'inserimento, per eseguire un inserimento è richiesto solamente il nome utente al quale collegare il congedo e la settimana (A oppure B). Il codice che si occupa dell'inserimento è il seguente:

```
public static function insert($username, $week)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO requests(username, week) VALUES (:username, :week)";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(":username", $username);
    $stmt->bindParam(":week", $week);
    try {
        if ($stmt->execute()) {
            return $pdo->lastInsertId();
        }
    } catch (PDOException $e) {
    }
    return false;
}
```

Successivamente all'inserimento sono presenti metodi che permettono di eliminare tutte le relazioni con un congedo. Il primo metodo si chiama deleteReasons e viene utilizzato per eliminare il collegamento tra un congedo e le proprie motivazioni:

```
public static function deleteReasons($id)
{
    $pdo = Database::getConnection();
    $query = "DELETE FROM request_reason WHERE request = :id";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':id', $id);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
    }
    return false;
}
```

Il secondo metodo invece viene utilizzato per eliminare tutti i collegamenti del congedo con i blocchi di orari presenti nella tabella substitutes:

```
public static function deleteSubstitutes($id)
{
    $pdo = Database::getConnection();
    $query = "DELETE FROM substitutes WHERE request = :id";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':id', $id);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
    }
    return false;
}
```

In fine è presente un metodo che si occupa di richiamare la generazione del PDF per un congedo in base al suo identificativo. Il metodo generatePdfForId accetta come primo parametro l'identificativo del congedo da utilizzare per la generazione del PDF mentre come secondo parametro la destinazione la quale può essere: nel browser direttamente oppure in formato stringa (utilizzato per l'invio nelle email). Il metodo è implementato nel seguente modo:

```
public static function generatePdfForId($id, $type = "I")
{
    $request = self::getById($id);
    if ($request && ($request["username"] == $_SESSION["username"] || Session::isSecretary())) {
        $reasons = Reasons::getByRequestId($id);
        $hours = Substitutes::getByRequestId($id);
        $user = LdapUsers::getByUsername($request["username"]);
    }
}
```

```
$pdf = new RequestPdf($reasons, $request, $hours, $user);
return $pdf->getContent($type);
} else {
    return false;
}
}
```

4.4.9.7 Stato delle richieste di congedo

Le richieste di congedo possiedono uno stato il quale viene salvato all'interno del database sotto forma di un numero. Una richiesta di congedo può avere i seguenti stati:

- WAITING, la richiesta di congedo si trova in attesa in un contenitore.
- ACCEPTED, la richiesta di congedo è stata accettata.
- REJECTED, la richiesta di congedo è stata rifiutata.
- NOTICED, la richiesta di congedo è stata constatata.

Come per i contenitori sono presenti delle costanti che determinano lo stato di una richiesta di congedo:

```
// Richiesta in attesa.
public const WAITING = 0;

// Richiesta accettata.
public const ACCEPTED = 1;

// Richiesta respinta.
public const REJECTED = 2;

// Richiesta constatata.
public const NOTICED = 3;
```

Come nella classe Container, è presente un metodo che permette di ricavare la stringa relativa allo stato della richiesta di congedo attraverso il suo identificativo:

```
public static function get($status) {
    switch ($status) {
        case self::WAITING:
            return 'In attesa';
        case self::ACCEPTED:
            return 'Accettata';
        case self::REJECTED:
            return 'Respinta';
        case self::NOTICED:
            return 'Constatata';
        default:
            return null;
    }
}
```

4.4.9.8 Creazione PDF richieste di congedo

Per la creazione dei PDF è stata usata la libreria FPDF per php. Per la creazione dell'intestazione del file PDF viene chiamato in modo automatico il metodo Header il quale contiene il seguente codice che permette di stampare un'immagine ed un testo:

```
public function Header()
{
    $this->Image(__DIR__ . '/cpt-logo.jpeg', 10, 10, -300);
    $this->SetFont('Arial', 'B', 12);
    $this->Cell(260, 16, 'ASSENZE E CONGEDI', 'B', 0, 'R');
    $this->Ln(16);
}
```

La stessa cosa viene utilizzata per il piede di pagina il quale è contenuto all'interno del metodo Footer:

```
public function Footer()
{
    $this->SetY(-17);
    $this->Cell($this->GetPageWidth() - 20, 0, '', 'T');
    $this->Image(__DIR__ . '/ti-logo.png', ($this->GetPageWidth() - 20) / 2,
200, 15, 15);
    $this->SetFont('Arial', '', 10);
    $this->Cell(-18, 10, 'Pagina ' . $this->PageNo() . ' di {nb}', 0, 0, 'C');
}
```

Il codice per la stampa del resto del documento PDF è molto lungo e di poca importanza, la creazione del documento avviene nel metodo costruttore nel quale sono contenuti diversi cicli che permettono di generare dinamicamente delle liste e griglie orarie. La griglia oraria viene riempita in modo automatico con gli orari salvati all'interno del database. Stessa cosa vale per la lista di motivazioni presente nella prima pagina del documento. Per determinare la destinazione del file PDF è presente un metodo getContent il quale si occupa di stampare il file PDF a schermo se la destinazione è il browser oppure di ritornare una stringa contenente il file PDF. Il codice del metodo getContent è il seguente:

```
public function getContent($type)
{
    $output = $this->Output($type, $this->fileName);
    if ($type == "I") {
        return true;
    } else {
        return $output;
    }
}
```

4.4.9.9 Tabella substitutes

Per la tabella substitutes è presente una classe Model chiamata Substitutes la quale permette di ricavare i blocchi di orari salvati da parte del calendario all'interno del database attraverso l'identificativo della richiesta di congedo. Il metodo si chiama getByIdRequest ed il codice è il seguente:

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

Inoltre è presente un metodo utilizzato per salvare i vari blocchi del calendario all'interno del database. Il metodo richiede come primo parametro l'id della richiesta di congedo al quale collegare il blocco del calendario, successivamente sono richieste le date di inizio e di fine, la tipologia di supplenza, l'aula, il supplente ed in fine la classe. Il codice per l'inserimento è il seguente:

```
public static function insert($request, $from, $to, $type, $room, $substitute,
$class)
{
    $pdo = Database::getConnection();
    $query = "INSERT INTO substitutes VALUES(:id, :from, :to, :type, :room,
:substitute, :class)";
    $stm = $pdo->prepare($query);
    $stm->bindParam(":id", $request);
    $stm->bindParam(":from", $from);
    $stm->bindParam(":to", $to);
    $type = ($type == "")?null:$type;
    $stm->bindParam(":type", $type);
    $stm->bindParam(":room", $room);
    $stm->bindParam(":substitute", $substitute);
    $stm->bindParam(":class", $class);
    try {
        return $stm->execute();
    } catch (PDOException $e) {
        return false;
    }
}
```

4.4.9.10 Tabella tokens

La tabella tokens possiede la propria classe Model chiamata Tokens. All'interno di questa classe sono presenti metodi per la creazione, invio ed eliminazione di codici di recupero password. I metodi di generazione del token di recupero password e di accesso sono descritti nel capitolo [Sicurezza](#) nella sezione [Recupero Password](#). Sono inoltre presenti due ulteriori metodi, uno per la creazione del codice di recupero password ed uno per eseguire il reset di eventuali token degli utenti. Il metodo per la creazione del token di recupero password si occupa anche della notifica tramite email, il codice è il seguente:

```
public static function sendToken($email)
{
    $pdo = Database::getConnection();
    $token = self::generateToken();
    $hash = hash('sha256', $token);
    $query = "INSERT INTO tokens(email, token) VALUES(:email, :token)";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':email', $email);
    $stm->bindParam(':token', $hash);
    try {
        $content = "contenuto";
        return $stm->execute() && Mail::send($email, 'titolo', $content);
    } catch (PDOException $e) {
        return false;
    }
}
```

Il codice per eseguire il reset dei token generati in precedenza è contenuto all'interno del metodo resetTokens il quale accetta come parametro l'email di un utente amministratore, questo metodo non fa altro che rimuovere tutti i record dalla tabella tokens che hanno un collegamento con l'email dell'utente specificato:

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

4.4.10 Sicurezza

La sicurezza è molto importante in questo progetto, questo dovuto al fatto che l'applicativo conserverà dati sensibili e sarà inoltre accessibile da reti esterne a quelle scolastiche.

4.4.10.1 Interrogazione database

Tutte le interrogazioni al database vengono eseguite utilizzando la classe PDO predisposta dal linguaggio di programmazione PHP, la quale offre metodi che permettono di evitare attacchi di tipo SQL Injection. Tutte le classi Modal utilizzano i Prepared Statements i quali permettono di validare in automatico le query da eseguire alla banca dati ed evitare attacchi.

4.4.10.2 Salvataggio dati

Prima che qualsiasi dato venga salvato all'interno del sistema viene validato e sanitizzato dalle classi Validators le quali si occupano di verificare la validità dei dati e se contengono possibili codici dannosi. In questo modo vengono prevenuti attacchi di tipo XSS o anche chiamati Cross-Site Scripting. Esempio per verificare che il testo delle osservazioni sia privo di codice malevolo viene utilizzato il seguente validatore:

```
public static function isValidDescription($text, $min = 0, $max = 255)
{
    $safe = htmlspecialchars($text);
    return strlen($text) >= $min && strlen($text) <= $max && $safe == $text;
}
```

Viene eseguito il metodo htmlspecialchars il quale controlla che non siano presenti tag html, se differente dalla stringa iniziale vuol dire che è presente qualche carattere malevolo.

4.4.10.3 Password locali

La password degli utenti locali (amministratori) viene salvata all'interno del database sotto forma di un hash generata utilizzando l'algoritmo bcrypt questo per prevenire che le password possano essere rubate dal database in caso un utente riesca ad accedervi (Password Leaks). Viene utilizzata la funzione password_hash di PHP con il parametro PASSWORD_DEFAULT, questo in modo tale che se bcrypt diventasse obsoleto e PHP scegliesse un altro algoritmo da utilizzare non siano necessari cambiamenti di codice. Il codice utilizzato per generare un hash della password è il seguente:

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

4.4.10.4 Recupero password

Il recupero della password viene eseguito attraverso l'uso di una stringa, definita token, con una lunghezza di 20 caratteri. Questo token viene generato in modo randomico dall'applicativo e viene salvato in correlazione con l'utente che ha richiesto il recupero della password assieme alla data di creazione della stringa. Questo token viene salvato all'interno del database utilizzando una sha256, questo in modo tale che anche se un utente abbia accesso al database in lettura non possa resettare le password degli utenti. Inoltre quando viene salvato questo token all'interno della banca dati viene salvata anche la sua data di creazione. I token hanno una vita massima di 10 minuti, una volta scaduti il token verrà disattivato. La generazione del token viene eseguita nel seguente modo:

```
private static function generateToken()
{
    $keyspace =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $pieces = [];
    $max = mb_strlen($keyspace, '8bit') - 1;
    for ($i = 0; $i < 20; ++$i) {
        $pieces []= $keyspace[random_int(0, $max)];
    }
}
```

```
        }
        return implode('', $pieces);
    }
```

Viene assegnato un set di caratteri da utilizzare per la creazione del token. Da questo set vengono ricavati dei caratteri da posizioni casuali in modo da formare una stringa di 20 caratteri massimo. Per controllare la validità e la presenza del token viene utilizzata questa funzione:

```
public static function login($token)
{
    $hash = hash('sha256', $token);
    $pdo = Database::getConnection();
    $query = "SELECT * FROM tokens WHERE token = :token";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':token', $hash);
    try {
        $stm->execute();
        $token = $stm->fetch(\PDO::FETCH_ASSOC);
        if ($token && time() - strtotime($token["created_at"]) <= 60 * 10) {
            $_SESSION["force_reset_password"] = true;
            $user = LocalAuth::login($token["email"], false, true);
            $user->updateLastLogin();
            Session::authenticate(array(
                "username" => $user->getUsername(),
                "name" => $user->getName(),
                "lastName" => $user->getLastName(),
                "permission" => $user->getPermission()
            ));
            self::resetTokens($token["email"]);
            return true;
        }
        return false;
    } catch (PDOException $e) {
        return false;
    }
}
```

Questa funzione login accetta un token come parametro, come prima cosa viene eseguita una sha256 del token inserito dall'utente in modo da poterlo comparare con i dati presenti nel database. Una volta eseguita l'hash viene interrogato il database per la ricerca di record con questo token, se è presente all'interno del database viene controllata la data di creazione e che sia minore di 10 minuti dal momento nel quale è stato utilizzato, se entrambi i requisiti sono soddisfatti l'utente può accedere all'applicativo. Una volta eseguito l'accesso verrà chiesto all'utente di cambiare la password. Se invece il token è inesistente o scaduto l'utente verrà rimandato alla pagina di accesso.

4.4.11 Interfacce grafiche

4.4.11.1 Pagina di accesso

La pagina di accesso all'applicativo permette di accedere alle zone riservate del sito web attraverso una combinazione di credenziali. Le credenziali possono essere LDAP oppure locali attraverso l'utilizzo di una email. La pagina si presenta con un semplice form il quale chiede username e password per accedere:

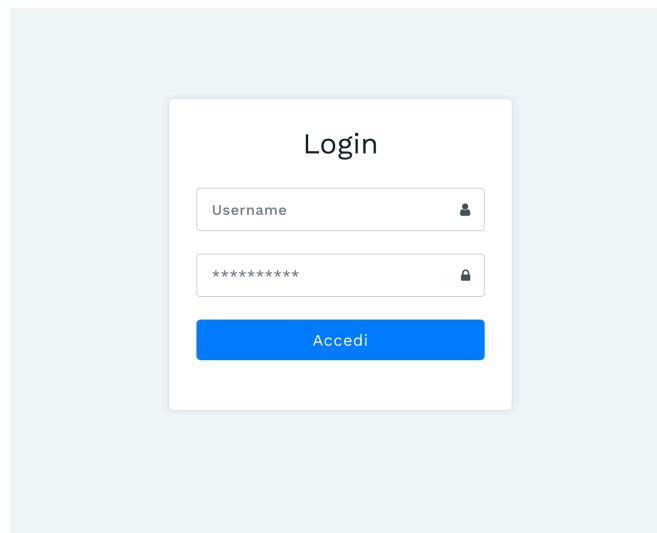


Figura 24 Pagina di accesso.

Per convalidare le credenziali inserite dall'utente viene eseguita una richiesta al backend utilizzando JavaScript. Se un utente locale inserisce delle credenziali errate viene mostrato un riferimento che permette di richiedere una email di recupero password:

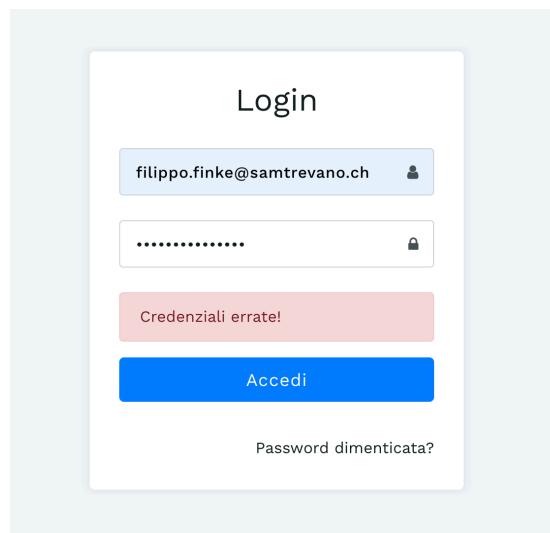


Figura 25 Riferimento recupero password.

Anche per questa chiamata viene utilizzato JavaScript che si occupa di informare il backend di inviare una email di recupero password.

4.4.11.2 Pagina principale

La pagina principale dell'applicativo è la Home, attraverso questa pagina è possibile compilare le richieste di congedo, sono presenti due tab:

- Motivazione
- Orario

Nella prima sezione è possibile selezionare le varie motivazioni relative alla richiesta di congedo, mentre nella seconda sezione è presente un calendario il quale permette di selezionare le date del congedo.

The screenshot shows the main application interface titled "Gestione congedi". On the left, there's a sidebar with "Congedi CPT" at the top, followed by "Home" and "Personale 0". The main content area has a header "Gestione congedi" and a breadcrumb "Dashboard > Home". Below this, there are two tabs: "Motivazione" (selected) and "Orario". Under "Motivazione", there are two input fields: one for "Adozione (16 settimane, previo giustificazione dei motivi)" and another for "Prova Prova". Both fields have a blue checkbox icon. A date "Data: 23.03.2020" is shown below the first field. To the right, a signature "Firma: Pinco Pallino" is displayed above a blue "Invia la richiesta" button. At the bottom, a footer note reads "Filippo Finke I4AC 2019 - 2020".

Figura 26 Pagina principale.

Il calendario per la selezione dei congedi appare in questo modo:

The screenshot shows the "Orario" tab of the leave request interface. It features a grid for selecting work hours. The columns represent time slots from 08:20 to 13:15. The rows are labeled with days of the week: LUNEDÌ, MARTEDÌ, MERCOLEDÌ, GIOVEDÌ, VENERDÌ, and SABATO. Each row contains a date input field (e.g., "gg/mm/aaaa") and a day name. A red "X" button is located in the top-left corner of the grid. A date "Data: 23.03.2020" is at the bottom left, and a signature "Firma: Pinco Pallino" is at the bottom right, along with a blue "Invia la richiesta" button.

Figura 27 Calendario richieste di congedo.

Attraverso questo calendario è dunque possibile selezionare la tipologia di settimana del congedo (A oppure B), selezionare le date dei singoli giorni della settimana ed in fine selezionare le ore di assenza. Alla selezione delle ore di congedo verrà mostrato all'utente un formulario il quale permetterà di aggiungere informazioni aggiuntive ai vari orari selezionati:

Figura 28 Modifica dati blocchi orari.

4.4.11.3 Pagina personale, congedi in uscita

Nel contenitore personale è presente una pagina la quale contiene i congedi che sono stati inviati dall'utente corrente, all'interno di questa pagina è solamente possibile visionare lo stato e altre informazioni riguardanti i congedi che sono in attesa di essere revisionati:

Data di creazione	Motivi/o	Assenze	Stato
23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke	In attesa in segreteria

Figura 29 Contenitore personale, pagina in uscita.

4.4.11.4 Pagina personale, storico

All'interno del contenitore personale è presente una pagina dedicata allo storico di tutte le richieste di congedo le quali sono state revisionate. La pagina permette di accedere a file PDF riguardanti tutte le richieste di congedo personali, la pagina si presenta in questo modo:

Data	Motivi/o	Assenze	Stato	Azione
23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke	Constatata	Visualizza PDF

Figura 30 Storico personale.

4.4.11.5 Contenitore segreteria

Il contenitore della segreteria è il primo posto nel quale verranno inoltrate le richieste di congedo, in questa pagina è presente una tabella la quale mostra tutte le richieste di congedo in attesa di revisione. Inoltre attraverso questa pagina è possibile inoltrare i congedi nel contenitore della direzione. La pagina è la seguente:

The screenshot shows a web-based application interface for managing leave requests. On the left, there is a sidebar with navigation links: Home, Personale (with 0 notifications), and Segreteria (with 1 notification, highlighted in blue). The main content area is titled 'Gestione congedi' and shows a table of leave requests. The table has columns: Docente (Teacher), Data di creazione (Creation Date), Motivo/o (Reason), Assenze (Absences), and Azioni (Actions). One row is visible for 'Finke Filippo' with creation date '23.03.2020', reason 'Adozione', absence 'Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI', and actions 'Revisiona' (in blue) and 'Conferma' (in green). Below the table, it says '1-1 di 1 congedi'. At the bottom right, there are buttons for 'Prima' (Previous), '1' (Current page), and 'Dopo' (Next). The footer of the page reads 'Filippo Finke I4AC 2019 - 2020'.

Figura 31 Contenitore segreteria.

Cliccando sul bottone Revisiona si viene inoltrati alla pagina principale con la possibilità di revisionare e modificare la richiesta di congedo. La pagina di modifica per la segreteria viene mostrata nel seguente modo:

The screenshot shows the 'Modifica come Segreteria' (Edit as Secretary) page for a leave request. The sidebar is identical to Figure 31. The main content area is titled 'Gestione congedi' and shows a form for modifying the leave request. It has tabs 'Motivazione' (selected) and 'Orario'. Under 'Motivazione', there are two checkboxes: 'Adozione (16 settimane, previo giustificazione dei motivi)' (checked) and 'Prova Prova'. Under 'Orario', there is another checkbox 'Prova Prova'. Below the form, it says 'Data: 23.03.2020'. At the bottom left is a blue button 'Salva la richiesta' (Save the request), and at the bottom right is a black button 'Torna indietro' (Go back). The footer of the page reads 'Filippo Finke I4AC 2019 - 2020'.

Figura 32 Revisione come segreteria.

4.4.11.6 Contenitore direzione e vice direzione

I congedi che sono stati revisionati ed approvati dalla segreteria procedono al contenitore della direzione o vice direzione. All'interno di questo contenitore sono mostrati tutti i congedi revisionati dalla segreteria. Attraverso questa pagina è possibile confermare un congedo ed inviarlo ai diretti interessati tramite posta elettronica in modo automatico. Il contenitore è il seguente:

Docente	Data di creazione	Motivo/o	Assenze	Osservazioni	Stato	Azioni
Filipe Finke	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI		In attesa	Revisione

Figura 33 Contenitore direzione.

Cliccando sul bottone revisiona si verrà inoltrati al congedo in modalità di modifica, la quale permetterà di definire lo stato finale della richiesta di congedo. La pagina di modifica è la seguente:

Figura 34 Modalità modifica come direzione.

La pagina è molto simile a quella di modifica della segreteria solamente con più funzionalità. È possibile definire lo stato della richiesta di congedo, se la supplenza è pagata oppure no e le ore riconosciute. Inoltre in questa pagina è possibile mandare la richiesta di congedo nuovamente nel contenitore della segreteria permettendo di avere la decisione definitiva oppure no.

4.4.11.7 Storico generale

Per gli utenti che appartengono alla direzione o vice direzione è disponibile un ulteriore pagina la quale consiste nello storico generale di tutte le richieste di congedo immesse nel sistema. La pagina contiene dunque una tabella con le diverse richieste di congedo:

The screenshot shows a web-based application interface for managing leave requests. On the left, there is a sidebar with navigation links: Home, Personale (with 0 notifications), Segreteria (with 0 notifications), Direzione (with 0 notifications), and Storico (which is currently selected). The main content area is titled 'Gestione congedi' and shows a table of leave requests. The table has columns: Docente, Data, Motivi/o, Assenze, Stato, Revisionato da, and Azione. There are four rows of data:

Docente	Data	Motivi/o	Assenze	Stato	Revisionato da	Azione
Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI	Respinta	Pinco Pallino	Visualizza PDF
Pallino Pinco	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF
Finke Filippo	23.03.2020	Adozione Prova	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke - 08:20-09:50 I4AC A-417 Finke - 09:05-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF
Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 - 08:20-09:50 I4AC A-417 - 09:05-09:50 I4AC A-417	Constatata	Filippo Finke	Visualizza PDF

Figura 35 Storico generale.

Anche attraverso questa pagina è possibile visionare il PDF delle singole richieste di congedo, la pagina del PDF è descritta nel capitolo [Visualizzazione PDF](#).

4.4.11.8 Visualizzazione PDF

In diverse pagine è presente un bottone che permette di visualizzare il file PDF di una richiesta di congedo, premendo quel pulsante verrà mostrato il file PDF della richiesta di congedo a schermo intero permettendo all'utente di scaricare, stampare oppure chiudere il file PDF. Un esempio di visualizzazione è il seguente:



Figura 36 Visualizzazione PDF.

4.4.11.9 Pagina di amministrazione utenti

Accedendo all'applicativo come utente amministratore si viene redirezionati ad un pannello di controllo apposito. La prima pagina disponibile è la gestione degli utenti, attraverso questa pagina è possibile vedere tutti gli utenti presenti all'interno del sistema, aggiungere amministratori, eliminare amministratori ed aggiornare i permessi dei vari utenti LDAP. Per eliminare utenti amministratori è disponibile un tasto dedicato alla rimozione a fianco di ogni riga la quale è possibile eliminare, mentre per aggiornare i permessi degli utenti LDAP è disponibile una tendina la quale permette di scegliere il permesso che verrà salvato in automatico.

Username	Nome	Cognome	Permesso	Tipo	Ultimo accesso	Azioni
fabrizio.valsangiacomo@edu.ti.ch	Fabrizio	Valsangiacomo	Amministratore	Locale	15:06 13.03.2020	<button>Elimina</button>
filippo.finke	Filippo	Finke	Direzione	LDAP	11:09 23.03.2020	
filippo.finke@samtrevano.ch	Filippo	Finke	Amministratore	Locale	11:49 23.03.2020	
pinco.pallino	Pinco	Pallino	Direzione	LDAP	11:26 23.03.2020	
utente.test	Utente	Test	Docente	LDAP	16:26 18.03.2020	

Figura 37 Gestione utenti.

Cliccando sul bottone Aggiungi amministratore è possibile creare un account amministratore. Per la creazione di un account amministratore è richiesto un indirizzo di posta elettronica, un nome ed un cognome. Alla creazione dell'utente le credenziali verranno inviate per email.

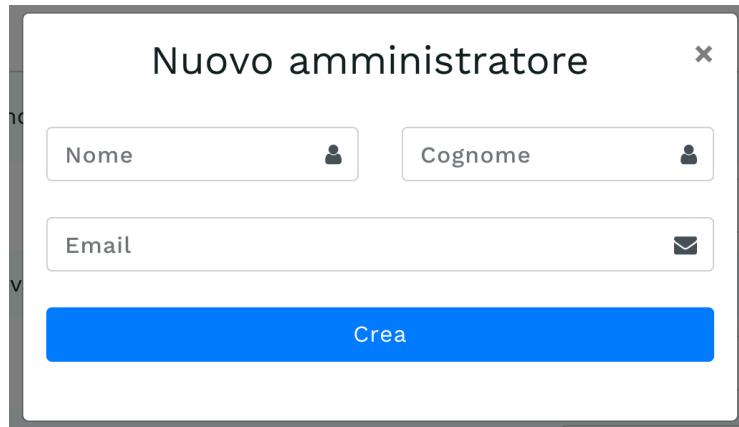


Figura 38 Schermata aggiunta utente amministratore.

Cliccando invece sul bottone Aggiungi utente LDAP è possibile aggiungere al sistema un utente LDAP in modo che quando l'utente esegua l'accesso abbia già dei permessi assegnati, se un utente non è presente nel sistema verrà creato con i permessi predefiniti.



Figura 39 Schermata aggiunta utente LDAP.

4.4.11.10 Pagina di amministrazione motivazioni

Oltre alla pagina di gestione degli utenti è presente una pagina che permette di gestire le motivazioni dei vari congedi. Attraverso questa pagina è dunque possibile aggiungere delle motivazioni, modificarne quelle già esistenti ed eliminarle. La pagina è la seguente:

Congedi CPT

Gestione Motivazioni

Amministrazione > Motivazioni

Aggiungi motivazione

5 righe per pagina

Cerca

Nome ↑↓ Descrizione ↑↓ Azioni

Nome	Descrizione	Azioni
Adozione	(16 settimane, previo giustificazione dei motivi)	Modifica Elimina
Malattia	Assenza causa malattia	Modifica Elimina

1-2 di 2 righe

Prima 1 Prossima

Filippo Finke I4AC 2019 - 2020

Figura 40 Gestione motivazioni.

Premendo sul bottone Aggiungi motivazione è possibile aggiungere una nuova motivazione al sistema, questo attraverso un formulario il quale chiede un nome ed una descrizione della motivazione da aggiungere:

Nuova motivazione

Motivazione

Descrizione

Crea

Figura 41 Schermata di aggiunta motivazioni.

4.4.11.11 Pagina di recupero/impostazione password

Una volta creato un utente amministratore ne vengono inviate le credenziali per posta elettronica. Al primo accesso ne è richiesto il cambio della password, la pagina utilizzata per eseguire il cambio della password è la stessa utilizzata per eseguire il recupero password. La pagina è la seguente:

The screenshot shows a web application interface for managing leave requests. On the left, there's a sidebar with 'Utenti' (Users) selected and 'Motivazioni' (Motivations). The main content area has a header 'Imposta la tua nuova password' and a breadcrumb 'Amministrazione > Imposta password'. It contains two input fields: 'Password' and 'Ripeti la password', both with a lock icon. Below them is a blue button labeled 'Imposta la password'. At the bottom, a footer bar displays 'Filippo Finke I4AC 2019 - 2020'.

Figura 42 Pagina di recupero/impostazione password.

5 Test

5.1 Protocollo di test

Test Case: Riferimento:	TC-000 REQ-000	Nome:	L'applicativo deve essere web.
Descrizione:	L'applicativo deve essere accessibile ed interpretato da un browser.		
Prerequisiti:	L'applicativo deve essere attivo ed accessibile dalla rete. Nella macchina che esegue il test deve essere presente Google Chrome.		
Procedura:	1. Aprire Google Chrome 2. Accedere all'applicativo con l'ip o il dominio della macchina.		
Risultati attesi:	Accedendo all'applicativo web verrà mostrata una pagina di accesso.		

Test Case: Riferimento:	TC-001 REQ-001	Nome:	Accesso attraverso interfaccia LDAP.
Descrizione:	Per eseguire l'accesso all'applicativo deve essere possibile utilizzare le credenziali LDAP del dominio scolastico.		
Prerequisiti:	Avere l'applicativo aperto su un browser sulla pagina di accesso.		
Procedura:	1. Inserire le proprie credenziali LDAP all'interno della schermata di accesso. 2. Premere il pulsante accedi.		
Risultati attesi:	La pagina web mostra un messaggio di colore verde "Accesso eseguito!" e redireziona l'utente al pannello principale dell'applicativo.		

Gestione richieste congedi docenti

Test Case:	TC-002	Nome:	Accesso attraverso interfaccia MySQL.
Riferimento:	REQ-002		
Descrizione:	L'applicativo deve poter comunicare con una banca dati locale per il salvataggio di dati e per poter permettere ad utenti di gestione di esso di poter accedere attraverso il proprio indirizzo email.		
Prerequisiti:	Avere l'applicativo aperto su un browser sulla pagina di accesso.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire le credenziali di un utente amministratore. <ol style="list-style-type: none"> a. L'username deve essere un indirizzo di posta elettronica. 2. Premere il pulsante accedi. 		
Risultati attesi:	La pagina web mostra un messaggio di colore verde "Accesso eseguito!" e redireziona l'utente al pannello di amministrazione.		

Test Case:	TC-003	Nome:	Interfaccia di posta elettronica.
Riferimento:	REQ-003		
Descrizione:	L'applicativo deve avere la possibilità di inviare posta elettronica.		
Prerequisiti:	Avere l'applicativo aperto su un browser sulla pagina di accesso.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire le credenziali di un utente amministratore. <ol style="list-style-type: none"> a. L'username deve essere un indirizzo di posta elettronica. b. La password inserita deve essere errata. 2. Premere il pulsante accedi. 3. Premere il pulsante "Password dimenticata?" 		
Risultati attesi:	La pagina web mostra un messaggio di colore verde "Email di recupero inviata!" e nella propria cartella di posta elettronica deve essere presente un messaggio con soggetto "Recupero password Gestione congedi".		

Test Case:	TC-004	Nome:	Pagina di accesso LDAP.
Riferimento:	REQ-004		
Descrizione:	L'applicativo necessita di una pagina di accesso attraverso la quale poter accedere. L'accesso può essere eseguito utilizzando le credenziali LDAP del dominio scolastico.		
Prerequisiti:	Avere l'applicativo aperto su un browser sulla pagina di accesso.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire le proprie credenziali LDAP all'interno della pagina di accesso. 2. Premere il pulsante accedi. 		
Risultati attesi:	La pagina web mostra un messaggio di colore verde "Accesso eseguito!" e redireziona l'utente al pannello principale dell'applicativo.		

Test Case:	TC-005	Nome:	Pagina di accesso Locale.
Riferimento:	REQ-004		
Descrizione:	L'applicativo necessita di una pagina di accesso attraverso la quale poter accedere. L'accesso può essere eseguito utilizzando le credenziali locali per permettere la gestione dell'applicativo web.		
Prerequisiti:	Avere l'applicativo aperto su un browser sulla pagina di accesso.		
Procedura:	<ol style="list-style-type: none"> 1. Inserire le credenziali di un utente amministratore. <ol style="list-style-type: none"> a. L'username deve essere un indirizzo di posta elettronica. 		

Gestione richieste congedi docenti

	2. Premere il pulsante accedi.
Risultati attesi:	La pagina web mostra un messaggio di colore verde “Accesso eseguito!” e redireziona l’utente al pannello di gestione dell’applicativo.

Test Case: Riferimento:	TC-006 REQ-005	Nome:	Pagina di un recipiente.
Descrizione:	L’applicativo deve essere strutturato su dei recipienti, all’interno di questi recipienti è possibile vederne i congedi assegnati.		
Prerequisiti:	L’utente deve avere eseguito l’accesso all’applicativo utilizzando delle credenziali LDAP e deve trovarsi nella pagina principale di esso.		
Procedura:	1. Selezionare nella barra di navigazione a sinistra il recipiente “Personale”. a. Dal menu a tendina selezionare “In uscita”.		
Risultati attesi:	Deve essere mostrato il contenitore del recipiente personale che mostra i congedi in uscita. Nella pagina deve essere presente una tabella con i vari congedi (se presenti). Inoltre nella barra di navigazione a sinistra deve essere evidenziato il recipiente “Personale”.		

Test Case: Riferimento:	TC-007 REQ-006	Nome:	Pagina principale.
Descrizione:	L’applicativo deve possedere una pagina principale attraverso la quale è possibile creare ed inviare congedi.		
Prerequisiti:	L’utente deve avere eseguito l’accesso all’applicativo utilizzando delle credenziali LDAP e deve trovarsi nella pagina principale di esso.		
Procedura:	1. Selezionare almeno una motivazione. 2. Selezionare la scheda “Orario”. 3. Selezionare la prima data disponibile per il lunedì. 4. Premere sul blocco d’orario “8:20 – 09:05”. 5. Inserire dati fasulli nel modale di inserimento del blocco orario. 6. Premere il pulsante “Invia la richiesta”.		
Risultati attesi:	Viene mostrata all’utente una notifica con scritto “Congedo creato” in alto a destra della pagina. Inoltre nella barra di navigazione laterale viene incrementato il numero di congedi in uscita.		

Test Case: Riferimento:	TC-008 REQ-007	Nome:	Pagina storico.
Descrizione:	L’applicativo deve possedere una pagina contenente lo storico di tutti i congedi salvati nel sistema.		
Prerequisiti:	L’utente deve avere eseguito l’accesso all’applicativo utilizzando delle credenziali LDAP, deve possedere il permesso “Direzione” oppure “Vice direzione” e deve trovarsi nella pagina principale di esso. Per ogni riga dovrà essere presente un pulsante per la visualizzazione del PDF.		
Procedura:	1. Selezionare il contenitore chiamato “Storico” nella parte sinistra della pagina. 2. Ricercare “2020”		
Risultati attesi:	Dovrà essere presente una lista di tutti i congedi contenenti il numero 2020 evidenziato in ogni riga. Inoltre per ogni congedo dovrà essere possibile visualizzarne il PDF.		

Gestione richieste congedi docenti

Test Case:	TC-009	Nome:	Pagina di amministrazione utenti.
Riferimento:	REQ-008		
Descrizione:	L'applicativo deve possedere una pagina che permetta di vedere gli utenti presenti nell'applicativo. La pagina deve essere presente nel pannello di amministrazione accessibile con credenziali amministrative. Dovrà inoltre essere possibile eseguire delle ricerche per qualsiasi dato.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> 1. Nel campo cerca inserire "Filippo" 2. Premere invio 		
Risultati attesi:	Dovranno essere mostrati tutti gli account che nel nome o cognome possiedono "Filippo" inoltre se l'account è amministratore e corrente non dovrà avere la possibilità di essere eliminato.		

Test Case:	TC-010	Nome:	Pagina di amministrazione motivazioni.
Riferimento:	REQ-009		
Descrizione:	L'applicativo deve possedere una pagina che permetta di aggiungere e modificare le motivazioni dei congedi. Questa pagina è accessibile solamente ad utenti di amministrazione.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> 1. Selezionare nella barra di navigazione a sinistra "Motivazioni" 		
Risultati attesi:	L'utente dovrà essere mandato alla pagina di gestione delle motivazioni. In questa pagina sarà presente una tabella nella quale verranno mostrate tutte le motivazioni inserite. Inoltre sarà possibile modificarle, aggiungerle ed eliminarle.		

Test Case:	TC-011	Nome:	Applicativo su host esterno.
Riferimento:	REQ-010		
Descrizione:	L'applicativo deve funzionare correttamente anche su un host esterno.		
Prerequisiti:	L'applicativo deve essere caricato su un host esterno. In questo caso è stato utilizzato infomaniak.		
Procedura:	<ol style="list-style-type: none"> 1. Recarsi all'indirizzo dell'host esterno alla propria macchina contenente l'applicativo. <ol style="list-style-type: none"> a. http://samtinfo.ch/i16finfil 		
Risultati attesi:	Dovrà essere mostrata la pagina di accesso dell'applicativo.		

Test Case:	TC-012	Nome:	Creazione utente amministratore.
Riferimento:	REQ-008		
Descrizione:	Deve essere possibile creare ulteriori utenti amministratori per la gestione dell'applicativo.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> 1. Premere il pulsante "Aggiungi amministratore" in alto a destra della pagina. 2. Inserire nel campo nome "TestNome". 3. Inserire nel campo cognome "TestCognome". 		

	<ol style="list-style-type: none"> 4. Inserire nel campo email un indirizzo di posta elettronica al quale si può accedere. 5. Premere il pulsante “Crea”
Risultati attesi:	Verrà inviata una email all'indirizzo di posta elettronica del nuovo amministratore ed esso verrà aggiunto alla lista di utenti presenti nel sistema.

Test Case:	TC-013	Nome:	Creazione utente LDAP.		
Riferimento:	REQ-008				
Descrizione:	Deve essere possibile creare utenti LDAP in modo da permettere di assegnare dei permessi ad essi prima che eseguino l'accesso.				
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.				
Procedura:	<ol style="list-style-type: none"> 1. Premere il pulsante “Aggiungi utente LDAP” in alto a destra della pagina. 2. Inserire nel campo nome “TestNome”. 3. Inserire nel campo cognome “TestCognome”. 4. Inserire nel campo username “nome.cognome”. 5. Selezionare il permesso “Docente”. 6. Premere il pulsante “Crea” 				
Risultati attesi:	L'utente verrà aggiunto alla lista di utenti presenti nel sistema con la tipologia LDAP.				

Test Case:	TC-014	Nome:	Creazione motivazione.		
Riferimento:	REQ-009				
Descrizione:	Deve essere possibile aggiungere delle motivazioni all'interno del sistema.				
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.				
Procedura:	<ol style="list-style-type: none"> 1. Nella barra di navigazione a sinistra selezionare “Motivazioni” 2. Premere il pulsante “Aggiungi motivazione” 3. Inserire nel titolo “Prova” 4. Inserire nella descrizione “Prova” 				
Risultati attesi:	La motivazione viene aggiunta e mostrata nella lista di motivazioni presenti nel sistema.				

Test Case:	TC-015	Nome:	Modifica motivazione.		
Riferimento:	REQ-009				
Descrizione:	Deve essere possibile modificare le motivazioni presenti all'interno del sistema.				
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.				
Procedura:	<ol style="list-style-type: none"> 1. Nella barra di navigazione a sinistra selezionare “Motivazioni” 2. Selezionare la motivazione con titolo “Prova” e premere il pulsante “Modifica” 3. Inserire nel titolo “Titolo” 4. Inserire nella descrizione “Descrizione” 5. Premere il pulsante “Aggiorna” 				

Risultati attesi:	La motivazione viene aggiornata e mostrata nella lista di motivazioni presenti nel sistema.		
--------------------------	---	--	--

Test Case: Riferimento:	TC-016 REQ-009	Nome:	Eliminazione motivazione.
Descrizione:	Deve essere possibile modificare le motivazioni presenti all'interno del sistema.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> Nella barra di navigazione a sinistra selezionare "Motivazioni" Selezionare la motivazione con titolo "Prova" e premere il pulsante "Elimina" 		
Risultati attesi:	La motivazione viene eliminata e non viene più mostrata nella lista di motivazioni presenti nel sistema.		

Test Case: Riferimento:	TC-017 REQ-008	Nome:	Eliminazione utente amministratore.
Descrizione:	Deve essere possibile eliminare gli amministratori presenti nell'applicativo tranne il profilo dell'utente corrente.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> Selezionare l'utente amministratore con il nome "TestNome" e premere il pulsante "Elimina". 		
Risultati attesi:	L'utente amministratore viene eliminato dal sistema.		

Test Case: Riferimento:	TC-018 REQ-008	Nome:	Aggiornamento permessi LDAP.
Descrizione:	Deve essere possibile aggiornare i permessi degli utenti LDAP presenti nell'applicativo.		
Prerequisiti:	L'utente deve avere eseguito l'accesso all'applicativo utilizzando delle credenziali locali, dunque deve essere un utente amministratore. Inoltre l'utente deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> Selezionare un utente che abbia la tipologia LDAP. Selezionare dal selettore del permesso la sigla "Direzione" 		
Risultati attesi:	Verrà mostrata a schermo una notifica che confermerà l'aggiornamento del permesso.		

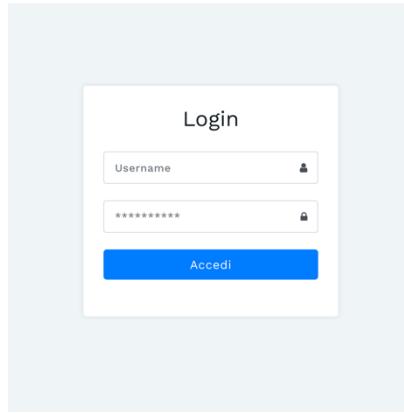
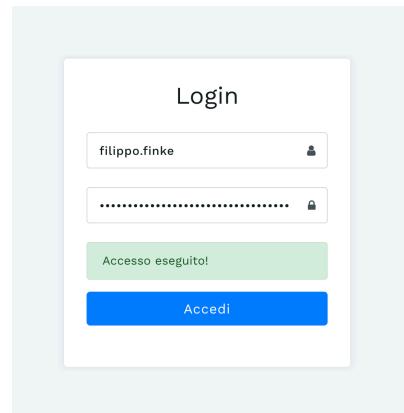
Test Case: Riferimento:	TC-019 REQ-006	Nome:	Creazione richiesta di congedo.
Descrizione:	Deve essere possibile creare una richiesta di congedo attraverso la pagina principale.		
Prerequisiti:	L'utente deve aver eseguito l'accesso all'applicativo utilizzando un account di dominio scolastico LDAP e deve trovarsi nella pagina principale.		
Procedura:	<ol style="list-style-type: none"> Selezionare almeno una motivazione. Selezionare la sezione "Orario". Impostare la settimana ad "A". Selezionare lunedì e mettere la prima data disponibile. Premere sul blocco orario "09:05 – 09:50". 		

	<ol style="list-style-type: none"> 6. Inserire in classe “I4AC”. 7. Inserire in aula “A-417”. 8. Selezionare tipologia “Supplenza interna”. 9. Inserire in supplente “Finke”. 10. Premere il pulsante “Salva”. 11. Premere il pulsante “Invia la richiesta”.
Risultati attesi:	Verrà mostrata a schermo una notifica che confermerà la creazione della richiesta di congedo e il numero sopra il contenitore personale sarà incrementato.

Test Case:	TC-020	Nome:	Inoltro al recipiente di direzione.		
Riferimento:	REQ-005				
Descrizione:	Deve essere possibile inoltrare i congedi da un recipiente ad uno successivo.				
Prerequisiti:	L'utente deve aver eseguito l'accesso all'applicativo utilizzando un account di dominio scolastico LDAP, deve fare parte della segreteria e deve trovarsi nella pagina principale. Inoltre deve essere già stato inserito un congedo nel sistema.				
Procedura:	<ol style="list-style-type: none"> 1. Nella barra di navigazione sinistra selezionare il recipiente “Segreteria” 2. Selezionare “In entrata” 3. Premere il pulsante “Conferma” sul primo congedo presente. 4. Premere il tasto “OK” nella notifica. 				
Risultati attesi:	Verrà mostrata a schermo una notifica che confermerà lo spostamento di recipiente del congedo. Inoltre non verrà più mostrato nel recipiente dedicato alla segreteria.				

Test Case:	TC-021	Nome:	Revisione richiesta di congedo.		
Riferimento:	REQ-005				
Descrizione:	Deve essere possibile revisionare le richieste di congedo in entrata in modo tale da poterne cambiare lo stato, cambiare supplenti, cambiare orari e così via.				
Prerequisiti:	L'utente deve aver eseguito l'accesso all'applicativo utilizzando un account di dominio scolastico LDAP, deve fare parte della direzione e deve trovarsi nella pagina principale. Inoltre deve essere già stato inserito un congedo nel sistema il quale deve essere stato approvato dalla segreteria.				
Procedura:	<ol style="list-style-type: none"> 1. Nella barra di navigazione sinistra selezionare il recipiente “Direzione” 2. Selezionare “In entrata” 3. Premere il pulsante “Revisiona” sul primo congedo presente. 4. Selezionare lo stato “Constatata”. 5. Impostare il pagamento della supplenza a “No”. 6. Premere il pulsante “Salva la richiesta”. 				
Risultati attesi:	Verrà mostrata una notifica che confermerà l'aggiornamento del congedo.				

5.2 Risultati test

Codice test	Stato	Risultati
TC-000	PASSATO	<p>L'applicativo è accessibile tramite un browser e viene mostrata correttamente la pagina di accesso.</p>  <p>Figura 43 Pagina di accesso.</p>
TC-001	PASSATO	<p>L'accesso con credenziali LDAP funziona correttamente e l'utente viene mandato alla pagina principale.</p>  <p>Figura 44 Messaggio di successo.</p>

Gestione richieste congedi docenti

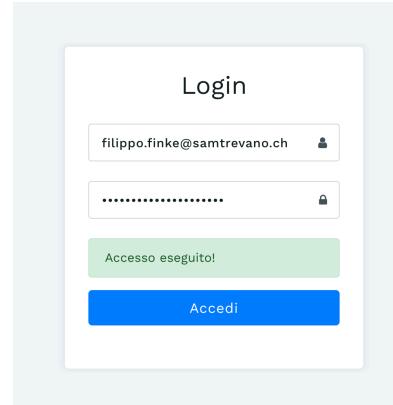
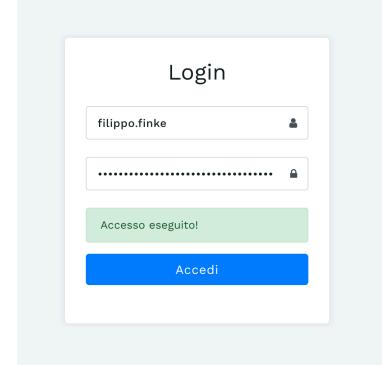
TC-002	PASSATO	Accendendo con un account locale utilizzando un indirizzo di posta elettronica come username e la relativa password l'accesso funziona correttamente e l'utente viene mandato al pannello di amministrazione.
 <p>The screenshot shows a 'Login' interface. It has two input fields: one for email ('filippo.finke@samtrevano.ch') and one for password ('.....'). Below the fields is a green button bar containing the text 'Accesso eseguito!' (Access granted!). At the bottom is a blue button labeled 'Accedi' (Log in).</p>		
TC-003	PASSATO	Inserendo l'email di account amministratore e richiedendo il recupero password viene ricevuta una email correttamente.
TC-004	PASSATO	L'accesso viene eseguito correttamente utilizzando credenziali del dominio scolastico LDAP.



Figura 46 Email di recupero password.

		 <p>The image shows a login interface with a light gray background. At the top center is the word "Login". Below it are two input fields: the first contains the email "filippo.finke" and the second contains a password represented by dots. Underneath the fields is a green horizontal bar with the text "Accesso eseguito!". At the bottom is a blue rectangular button labeled "Accedi".</p>
TC-005	PASSATO	Eseguendo l'accesso con email e password di un account amministratore locale l'accesso viene eseguito correttamente.
TC-006	PASSATO	Il recipiente personale "in uscita" viene mostrato correttamente.

--	--	--

Figura 49 Recipiente "In uscita".

TC-007	PASSATO	Il congedo viene creato con successo e il contatore di congedi in uscita viene incrementato.
TC-008	PASSATO	La pagina dello storico viene caricata correttamente, inoltre vengono mostrati solamente i congedi contenenti l'anno 2020.

Figura 50 Contatore congedi in uscita incrementato.

		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Docente</th> <th>Data</th> <th>Motivo/o</th> <th>Assenze</th> <th>Stato</th> <th>Revisionato da</th> <th>Azione</th> </tr> </thead> <tbody> <tr> <td>Finke Filippo</td> <td>23.03.2020</td> <td>Adozione</td> <td>Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI</td> <td>Respinta</td> <td>Pinco Pallino</td> <td>Visualizza PDF</td> </tr> <tr> <td>Pallino Pinco</td> <td>23.03.2020</td> <td>Adozione</td> <td>Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke</td> <td>Constatata</td> <td>Filippo Finke</td> <td>Visualizza PDF</td> </tr> <tr> <td>Finke Filippo</td> <td>23.03.2020</td> <td>Adozione</td> <td>Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke - 08:20-09:50 I4AC A-417 Finke - 09:05-09:50 I4AC A-417 Finke</td> <td>Constatata</td> <td>Filippo Finke</td> <td>Visualizza PDF</td> </tr> </tbody> </table>	Docente	Data	Motivo/o	Assenze	Stato	Revisionato da	Azione	Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI	Respinta	Pinco Pallino	Visualizza PDF	Pallino Pinco	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF	Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke - 08:20-09:50 I4AC A-417 Finke - 09:05-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF
Docente	Data	Motivo/o	Assenze	Stato	Revisionato da	Azione																								
Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke SI	Respinta	Pinco Pallino	Visualizza PDF																								
Pallino Pinco	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF																								
Finke Filippo	23.03.2020	Adozione	Settimana A 23.03.2020 - 08:20-09:50 I4AC A-417 Finke - 08:20-09:50 I4AC A-417 Finke - 09:05-09:50 I4AC A-417 Finke	Constatata	Filippo Finke	Visualizza PDF																								

Figura 51 Pagina storico.

TC-009	PASSATO	<p>La pagina di gestione degli utenti viene mostrata correttamente, inoltre la ricerca ritorna solamente utenti che contengono "Filippo" nel nome o nel cognome.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Username</th> <th>Nome</th> <th>Cognome</th> <th>Permesso</th> <th>Tipo</th> <th>Ultimo accesso</th> <th>Azioni</th> </tr> </thead> <tbody> <tr> <td>filippo.finke</td> <td>Filippo</td> <td>Finke</td> <td>Docente</td> <td>LDAP</td> <td>14:37 25.03.2020</td> <td>Filippo</td> </tr> <tr> <td>filippo.finke@samtrevalo.ch</td> <td>Filippo</td> <td>Finke</td> <td>Amministratore</td> <td>Locale</td> <td>13:32 26.03.2020</td> <td></td> </tr> </tbody> </table>	Username	Nome	Cognome	Permesso	Tipo	Ultimo accesso	Azioni	filippo.finke	Filippo	Finke	Docente	LDAP	14:37 25.03.2020	Filippo	filippo.finke@samtrevalo.ch	Filippo	Finke	Amministratore	Locale	13:32 26.03.2020	
Username	Nome	Cognome	Permesso	Tipo	Ultimo accesso	Azioni																	
filippo.finke	Filippo	Finke	Docente	LDAP	14:37 25.03.2020	Filippo																	
filippo.finke@samtrevalo.ch	Filippo	Finke	Amministratore	Locale	13:32 26.03.2020																		
TC-010	PASSATO	<p>La pagina delle motivazioni viene caricata correttamente mostrando tutte le motivazioni presenti nel sistema con i relativi pulsanti per eseguire modifiche o eliminazione.</p>																					

Gestione richieste congedi docenti

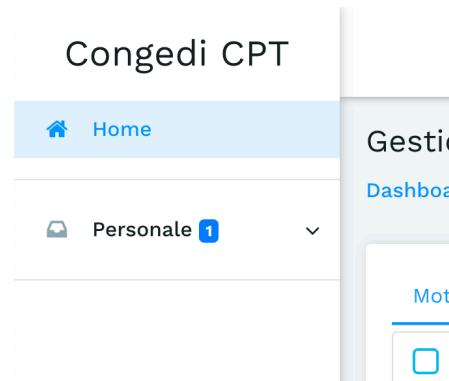
--	--	--

Figura 53 Pagina di gestione motivazioni.

TC-011	PASSATO	L'applicativo è accessibile attraverso http://samtinfo.ch/i16finfil e la pagina di accesso viene caricata correttamente.
TC-012	PASSATO	La creazione di un utente amministratore funziona correttamente, vengono inviate per posta elettronica le credenziali di accesso e l'utente viene aggiunto.

Gestione richieste congedi docenti

		Gestione richieste congedi docenti										
		Utente aggiunto correttamente.										
TC-013	PASSATO	Credenziali di accesso Gestione congedi										
		<no-reply@gestione-congedi-cpt.ch>	to: <xaxue@psles.com>	received a minute ago								
<p>Salve, è stato creato un account amministratore con questo indirizzo email. Credenziali di accesso: Email: axue@psles.com Password: yYEnY0Eg1hZ7VjaZF3pq</p>												
Figura 56 Credenziali per posta elettronica.												
TC-014	PASSATO	L'utente viene creato correttamente e mostrato all'interno degli utenti presenti nel sistema.										
		nome.cognome	TestNome	TestCognome	Docente	LDAP	14:39	26.03.2020				
Figura 57 Utente creato correttamente.												
TC-015	PASSATO	La motivazione viene creata correttamente ed aggiunta alla lista di motivazioni.										
		Prova	Prova			Modifica	Elimina					
Figura 58 Motivazione aggiunta.												
TC-015		La motivazione viene aggiornata correttamente e mostrata nella lista di motivazioni.										

		Titolo Descrizione	Modifica	Elimina
Figura 59 Motivazione aggiornata.				
TC-016	PASSATO	La motivazione viene eliminata correttamente.		
TC-017	PASSATO	L'account amministratore viene eliminato correttamente.		
TC-018	PASSATO	Viene mostrata una notifica di conferma ed il permesso dell'utente viene aggiornato correttamente.		
TC-019	PASSATO	La richiesta di congedo viene creata correttamente e il numero di notifica viene incrementato.		
Figura 60 Numero incrementato.				
TC-020	PASSATO	La richiesta di congedo viene inoltrata correttamente al recipiente successivo e ne viene mostrata una notifica.		

Gestione richieste congedi docenti

Congedi CPT

Home

Personale 1

Segreteria 0

Gestione congedi

Dashboard > Segreteria

Mostra 5 congedi per pagina

Cerca

Docente Data di creazione Motivo/o Assenze Azioni

Nessun dato da mostrare.

Prima Dopo

Filippo Finke I4AC 2019 - 2020

Figura 61 Recipiente segreteria.

TC-021	PASSATO	La richiesta di congedo viene aggiornata correttamente e viene mostrata una notifica.
--------	---------	---

5.3 Mancanze/limitazioni conosciute

Il progetto è stato sviluppato rispettando tutti i requisiti descritti all'interno del Quaderno dei Compiti (QdC) ed è stato completato. Non sono presenti delle mancanze o limitazioni all'interno dell'applicativo web.

6 Consuntivo

Il GANTT consuntivo è molto simile al GANTT preventivo anche se sono state aggiunte alcune attività. La fase dell'analisi del progetto è risultata molto più corta rispetto a quello previsto nel diagramma preventivo, il tempo risparmiato in questa fase è stato investito nella fase dello sviluppo del frontend nell'attività di sviluppo del calendario. L'attività di sviluppo di una classe dedicata al calendario non era stata prevista nella pianifica preventiva in quanto era presente una libreria chiamata Full Calendar la quale risolveva il problema del calendario però non rispettava i requisiti del progetto e dunque è stato deciso di sviluppare da zero una libreria per la gestione del calendario delle richieste di congedo.

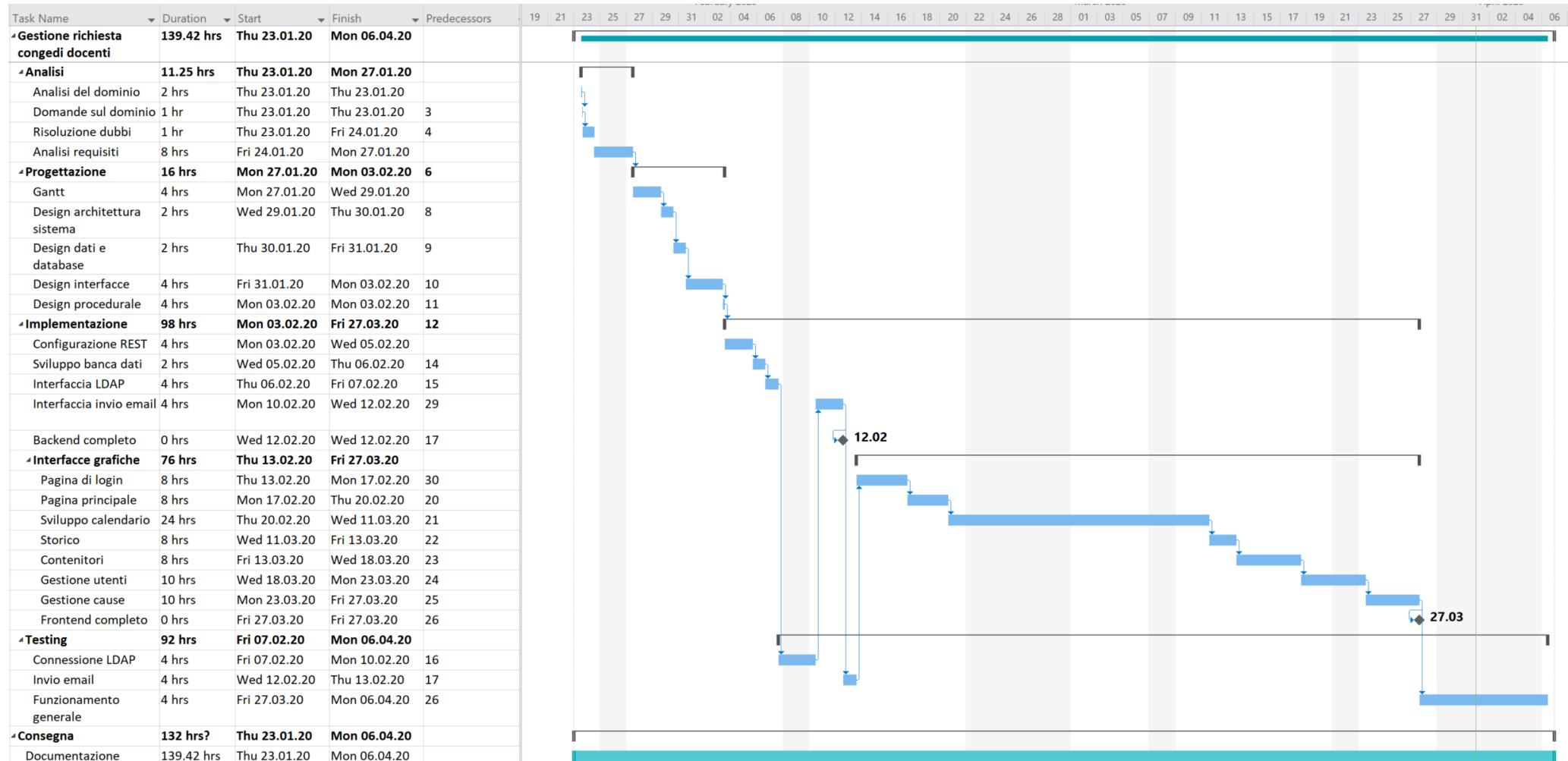


Figura 62 Diagramma di GANTT consuntivo.

7 Conclusioni

L'applicativo è stato sviluppato con lo scopo di velocizzare il processo di richiesta, creazione e gestione delle richieste di congedo dei docenti del centro professionale di Trevano. Nel mercato corrente non sono presenti software o applicativi utili alla risoluzione di questo problema in quanto le specifiche erano molto precise e specifiche alla scuola. Ritengo dunque che il progetto potrà portare dei benefici al CPT.

7.1 Sviluppi futuri

Si potrebbero aggiungere dati riguardanti tutte le aule presenti nella scuola, tutte le classi e tutti i docenti in modo da permettere all'applicativo di consigliare quali dati inserire nella selezione degli orari del congedo. Inoltre sarebbe interessante digitalizzare anche i formulari a parte richiesti per determinate motivazioni al posto di mantenerli cartacei in modo da digitalizzare totalmente questo ambito. Un ulteriore sviluppo futuro sarebbe la possibilità di selezionare dove ricevere le notifiche riguardanti il cambio stato delle proprie richieste di congedo, ad esempio attraverso posta elettronica oppure sms. Inoltre sarebbe interessante aggiungere delle regole per applicare controlli automatici su limite di giorni possibili da richiedere ogni anno, sequenze di requisiti e molto altro.

7.1.1 Sviluppo di controlli sul conteggio giorni

Per sviluppare la soluzione di controlli servirebbe aggiungere delle colonne adeguate alla tabella delle motivazioni contenenti le varie regole da verificare prima di approvare l'inserimento di un congedo. Aggiungendo questi campi alla tabella motivazioni prima del salvataggio del congedo si dovrebbero eseguire tutte queste verifiche ricavando le regole di validazioni del database per ogni singola motivazione con le date scelte dall'utente nella richiesta di congedo stessa. Una volta che tutti i controlli sono passati la richiesta di congedo potrà essere salvata all'interno del sistema e mandata a contenitori successivi. Inoltre per implementare questa funzionalità si dovrebbe aggiungere una tabella contenente il conteggio dei giorni e delle ore legato alle varie motivazioni la quale andrebbe correlata con la tabella degli utenti già presenti nel sistema. In questo modo il sistema diventerebbe più completo e automatizzerebbe ancora di più il lavoro da parte della amministrazione della scuola.

7.2 Considerazioni personali

Il progetto è stato molto utile per consolidare le mie conoscenze di sviluppo nell'ambito web. Ho potuto lavorare al progetto con librerie principalmente sviluppate da me stesso il quale mi hanno fatto capire il mio livello di mantenimento e usabilità del codice a lungo termine. Inoltre questo progetto copriva a 360° lo sviluppo web il quale mi ha permesso di approfondire alcune delle mie conoscenze e mettere in pratica tutti ciò che conosco al riguardo.

8 Glossario

Parola	Descrizione
LDAP	Lightweight Directory Access Protocol, è un protocollo per l'interrogazione e la modifica di servizi di directory.
PDF	Portable Format Document, è un formato di file utilizzato per la rappresentazione di testo ed immagini sviluppato da Adobe.
REST	Representational State Transfer, è uno stile di sviluppo di software.
frontend	Interfaccia accessibile da parte degli utenti, la parte grafica.
backend	Parte di un software che elabora i dati generati dal frontend.
mockup	Rappresentazione di interfacce grafiche in modo generale.
MySQL	È un software per la gestione di database.
PHP	Hypertext Preprocessor, linguaggio di programmazione lato server utilizzato spesso nello sviluppo web.
Composer	Gestore di pacchetti aggiuntivi per il linguaggio PHP.
hash	È il risultato di una funzione di hash, un algoritmo che permette di generare una stringa di lunghezza fissa composta da caratteri casuali.
bcrypt	È una funzione di hash.

middlewares	Funzioni o classi che fungono da intermediari.
FPDF	FPDF è una classe PHP che permette di generare file PDF.
CSS	CSS (Cascading Style Sheet) è un linguaggio che descrive lo stile di file HTML.
JavaScript	JavaScript è un linguaggio di programmazione.

9 Sitografia

- <https://docs.microsoft.com/>, *SAM-Account-Name attribute - Win32 apps*, 29.01.2020
- <https://stackoverflow.com/>, *Best practice on generating reset password tokens*, 29.01.2020
- <https://github.com/filippofinke/php-rest>, *Simple php rest api framework*, 30.01.2020
- <https://www.php.net/manual/>, *PHP: get_declared_classes*, 30.01.2020
- <https://devdojo.com/articles/php-to-string-equivalent>, *PHP toString Equivalent*, 30.01.2020
- <https://support.microsoft.com/>, *Come utilizzare i flag di...*, 30.01.2020
- <https://devblogs.microsoft.com/>, *How to detect a valid Active Directory...*, 03.02.2020
- https://datatables.net/examples/basic_init/, *DataTables example - Language options*, 03.02.2020
- <https://www.php.net/manual/en/function.mb-strlen.php>, *PHP: mb_strlen - Manual*, 03.02.2020
- <https://stackoverflow.com/>, *How to disable text selection highlighting*, 06.02.2020
- <https://datatables.net/reference/option/language/language>, 20.02.2020
- <https://stackoverflow.com/>, *Preventing Directory Traversal in PHP but allowing paths*, 20.02.2020
- <http://www.fpdf.org/>, *FPDF*, 02.03.2020
- <https://developer.mozilla.org/>, *Pragma - HTTP | MDN*, 09.03.2020

10 Allegati

- Manuale di installazione
- Diari di lavoro
- Quaderno dei compiti
- Codice sorgente presente su GitLab scolastico
 - http://gitsam.cpt.local/2019_2020_2_semestre/gestione-congedi-docenti