Inventario HW

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

Situazione iniziale 3

Attuazione 3

Risultati 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.3 Use case 8

2.4 Pianificazione 9

2.5 Analisi dei mezzi 10

2.5.1 Software 10

2.5.2 Hardware 10

3 Progettazione 10

3.1 Design dell’architettura del sistema 10

3.2 Design dei dati e database 11

3.3 Design delle interfacce 12

3.4 Design procedurale 17

4 Implementazione 18

4.1 Database 18

4.2 Libs 19

4.2.1 Collegamento 19

4.2.2 Hash 19

4.2.3 View 21

4.3 Views 22

4.3.1 HeaderAdmin 22

4.4 Controllers 22

4.4.1 Home 22

4.4.2 Login 23

4.4.3 Logout 24

4.4.4 Register 25

4.4.5 Modifyuser 26

4.4.6 Deleteuser 27

4.4.7 Iframe 28

4.4.8 Inventory 29

4.5 Models 32

4.5.1 Login\_Model 32

4.5.2 Register\_Model 33

4.5.3 Modifyuser\_Model 36

4.5.4 Inventory\_Model 37

5 Test 40

5.1 Protocollo di test 40

5.2 Risultati test 41

5.3 Mancanze/limitazioni conosciute 41

6 Consuntivo 41

7 Conclusioni 41

7.1 Sviluppi futuri 41

7.2 Considerazioni personali 41

8 Bibliografia 41

8.1 Bibliografia per articoli di riviste: 41

8.2 Bibliografia per libri 41

8.3 Sitografia 42

9 Allegati 42

# Introduzione

## Informazioni sul progetto

* Allievo: Dennis Donofrio
* Docente: Guido Montalbetti
* Azienda: SAM Trevano
* Classe: I3BC
* Materia: M306 / Laboratorio progetti
* Periodo: 16.09.2021 – 23.12.2021

## Abstract

## Situazione iniziale

All’inizio di questo progetto mi è stato chiesto di realizzare un sistema che possa gestire i vari computer e periferiche. Al momento tutti questi oggetti sono catalogati in un file Excel. Ogni volta per prendere o portare un oggetto bisogna chiedere all’amministratore di aggiungere o togliere questo componente. Questo progetto permette di sapere quanti oggetti ci sono in totale, quanti oggetti sono liberi, quanti oggetti sono occupati e in quale aula sono usati attualmente. Un utente con i permessi di livello amministratore può creare degli oggetti, rimuoverli, riservarli, creare nuovi utenti e rimuovere degli utenti.

L’utente base può semplicemente riservare degli oggetti e liberarli.

## Attuazione

In questo progetto vengono utilizzati HTML, CSS, PHP e MySQL. Per l’attuazione di questo progetto si tiene presente la semplicità di utilizzo, in modo da dare la possibilità a tutti di utilizzarlo.

## Risultati

Il sito è riuscito nel suo intento con abbastanza successo. Quasi tutte le funzionalità richieste nel QdC.

## Scopo

Lo scopo di questo progetto è quello di semplificare la riservazione di materiale informatico. In questo modo si possono riservare degli oggetti e vedere quali oggetti sono ancora disponibili, quanti e altre informazioni.

# Analisi

## Analisi del dominio

All’inizio di questo progetto mi è stato chiesto di realizzare un sistema che possa gestire i vari computer e periferiche. Al momento tutti questi oggetti sono catalogati in un file Excel. Ogni volta per prendere o portare un oggetto bisogna chiedere all’amministratore di aggiungere o togliere questo componente. Questo progetto permette di sapere quanti oggetti ci sono in totale, quanti oggetti sono liberi, quanti oggetti sono occupati e in quale aula sono usati attualmente. Un utente con i permessi di livello amministratore può creare degli oggetti, rimuoverli, riservarli, creare nuovi utenti e rimuovere degli utenti.

L’utente base può semplicemente riservare degli oggetti e liberarli.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Ambiente funzionante |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Apache, PHP, MySQL funzionanti |
| **Sotto requisiti** | |
| **001** | Apache con la porta 80 |
| **002** | MySql con la porta 3306 |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Struttura database |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessita di tabelle riservate agli utenti e tabelle riservate ai componenti HW |
| **Sotto requisiti** | |
| **001** | Tabella utente |
| **002** | Tabella oggetto |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Creazione login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Una pagina unica che faccia accedere gli utenti. |
| **Sotto requisiti** | |
| **001** | Si necessita una maschera di login |
| **002** | Si necessita di un tasto per cambiare password in caso di smarrimento |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Distinzione utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Amministratori e utenti base avranno permessi diversi. |
| **Sotto requisiti** | |
| **001** | Si necessita di utenti base ed amministratori |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Creazione varie pagine web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano 4 tipi di pagine. |
| **Sotto requisiti** | |
| **001** | Si necessita una pagina per la home degli utenti |
| **002** | Si necessita una pagina per la home degli amministratori |
| **003** | Si necessita una pagina per vedere tutto l’inventario |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Creazione di filtri |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano di filtri per migliorare l’efficacia della ricerca |
| **Sotto requisiti** | |
| **001** | Filtri sulla categoria |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Percentuale di soglia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Percentuale sotto la quale verrà inviata una email per comunicare che la disponibilità quella tipologia è sceso sotto la percentuale |
| **Sotto requisiti** | |
| **001** | Si necessita la possibilità di parametrizzare la percentuale di attenzione |
| **002** | Se la percentuale è stata superata, viene inviata una mail notificando che una tipologia sta finendo |

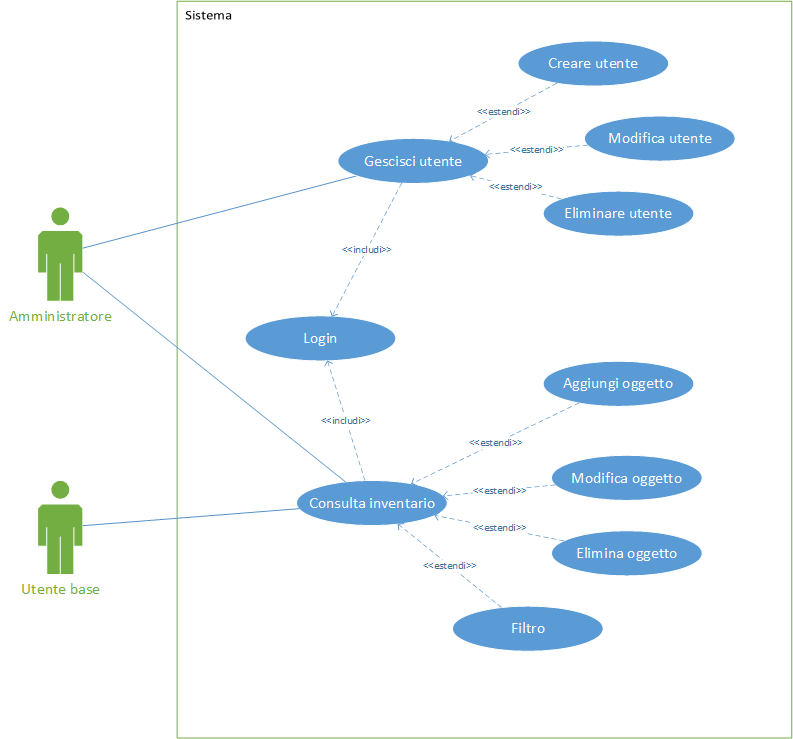
|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Funzioni amministratori |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli amministratori hanno il controllo completo dell’applicativo. |
| **Sotto requisiti** | |
| **001** | Si necessita la modifica da parte degli amministratori |
| **002** | Gli amministratori potranno aggiungere o rimuovere i componenti HW |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Gestione utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli amministratori hanno la possibilità di aggiungere, modificare o rimuovere un utente. |
| **Sotto requisiti** | |
| **001** | Si necessita un collegamento alla pagina di gestione degli utenti |
| **002** | Si necessitano 3 pagine per le varie operazioni sugli utenti |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Modifica inventario |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti hanno la possibilità di aggiungere, modificare o rimuovere un oggetto dall’inventario. |
| **Sotto requisiti** | |
| **001** | Si necessita un collegamento alla pagina di gestione dell’inventario |
| **002** | Si necessitano due pagine per le varie operazioni sull’inventario |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Impostazione soglia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Gli utenti hanno la possibilità impostare una soglia per la quantità minima di componenti. |
| **Sotto requisiti** | |
| **001** | Si necessita l’invio di mail. |
| **002** | Si necessita una pagina per l’impostazione della soglia. |

## Use case



## Pianificazione

|  |
| --- |
|  |

## Analisi dei mezzi

### Software

* Sublime text
* Apache2.4
* MySQL
* PHP7
* VirtualBox
* HTML5
* CSS3

### Hardware

PC – specifiche:

* Intel Xeon E3-1240
* 16 GB RAM
* 256 GB SSD
* Nvidia Quadro

# Progettazione

## Design dell’architettura del sistema

|  |
| --- |
|  |

Struttura delle cartelle

Ho usato la struttura delle cartelle mvc, che mi permette di gestire in modo adeguato questo applicativo web.

Nella cartella ‘views’ ci sono tutti i file con il contenuto html.

Nella cartella ‘models’ ci sono tutti i file che fanno delle query al database.

Nella cartella ‘controllers’ ci sono tutti i file che ricevono le richieste dal browser.

Nella cartella ‘config’ ci sono i file che contengono parti di codice che si possono riutilizzare in altri file, come la connessione al database o delle costanti.

Nella cartella ‘libs’ ci sono delle classi che vengono usate come super-classi.

Nella cartella ‘public’ ci sono i file javascript, le immagini, tutte le cartelle di css e bootstrap.

## Design dei dati e database

|  |
| --- |
|  |

La struttura del database è composta da 4 tabelle. Queste tabelle sono necessarie per il corretto funzionamento dell’applicativo. Di seguito ci sono le descrizioni complete di tutte le tabelle.

* **user:** Questa tabella contiene molti attributi. L’id permette di distinguere gli utenti e controllare se un utente ha effettuato il login nel sito. Is\_admin permette di sapere se un utente ha privilegi di livello amministratore o utente base. Name e surname sono essenziali per l’autenticazione perché quando si registra un utente non si chiede di inserire il nome utente, ma viene creato automaticamente. Il nome utente viene creato concatenando il nome ed il cognome, separati dal carattere ‘.’. L’attributo email potrebbe essere utile per un futuro sviluppo. Può essere utilizzata per mandare delle mail di conferma nella registrazione, per la modifica, per la password persa o per altre comunicazioni importanti. L’ultimo attributo contiene l’hash della password. Questo è utile per la sicurezza degli utenti e per la sicurezza dell’accesso.
* **classroom:** Questa tabella contiene solo due attributi. L’attributo number definisce il numero dell’aula. L’attributo description contiene una descrizione dell’aula, per esempio “Aula disegnatori”.
* **type:** Questa tabella contiene solo 3 attributi. L’id permette di distinguere i tipi a livello di database, in modo da non dover lavorare con le stringhe. L’attributo id è univoco. L’attributo name contiene il nome del tipo, per esempio “proiettore”. L’attributo description contiene una breve descrizione del tipo, per esempio “Proiettore e retro proiettore”.
* **object:** Questa tabella è la più delicata perché contiene 3 chiavi esterne. In totale ha 7 attributi. L’id permette di distinguere i vari oggetti a livello di database. L’id è univoco. L’attributo description contiene una descrizione dell’oggetto nell’inventario. L’attributo serial\_number contiene il numero seriale dell’oggetto. L’attributo riservation\_date contiene la data di riservazione di un oggetto, vuol dire la data in cui un oggetto è stato spostato dal magazzino ed inserito in un’aula.

## Design delle interfacce

|  |
| --- |
|  |

Pagina di login

Questa pagina può essere visualizzata da tutti. È presente un textbox per l’inserimento dello username e un campo password per l’inserimento della password.

|  |
| --- |
|  |

Pagina home utente base

Questa pagina può essere visualizzata da tutti gli utenti che hanno effettuato il login e sono degli utenti base. C’è la possibilità di passare alla pagina per modificare il proprio account e il bottone per passare alla pagina dell’inventario.

|  |
| --- |
|  |

Pagina home amministratore

Questa pagina può essere visualizzata da tutti gli utenti che hanno effettuato il login e sono degli utenti amministratori. C’è la possibilità di passare alla pagina per modificare il proprio account. Inoltre si può passare alla pagina per modificare i vari utenti e con il bottone si può passare alla pagina dell’inventario.

|  |
| --- |
|  |

Pagina modifica utenti

Questa è la pagina che si presenta quando un amministratore vuole apportare delle modifiche ad un utente. Ci sono 3 bottoni denominati ‘Add’, ‘Modify’ e ‘Remove’. Se viene cliccato il bottone ‘Add’ si viene reindirizzati alla pagina per aggiungere un utente. Se viene cliccato il bottone ‘Modify’ si viene reindirizzati alla pagina per modificare un utente. Se viene cliccato il bottone ‘Remove’ si viene reindirizzati alla pagina per rimuovere un utente.  
Nell’header della pagina si trova il bottone per tornare alla propria home ed il bottone per eseguire il logout.

|  |
| --- |
|  |

Pagina registrazione utente

Questa è la pagina che si presenta quando viene cliccato il bottone per aggiungere un utente. È possibile inserire se l’utente che stiamo creando vogliamo che diventi un amministratore. Bisogna inserire il nome, cognome, la mail e la password. Dopo aver riempito con i campi corretti tutti i campi bisogna cliccare il bottone ‘Register’ per registrare effettivamente un utente.

|  |
| --- |
|  |

Pagina eliminazione utente

Questa è la pagina che si presenta quando viene cliccato il bottone per rimuovere un utente. C’è un menu a tendina dove è possibile visualizzare tutti gli utenti. Per eliminare un utente è necessario confermare inserendo il nome utente. Dopo aver completato tutti i campi si può cliccare il bottone ‘Delete’ per rendere effettiva l’eliminazione dell’utente.

|  |
| --- |
|  |

Pagina modifica utente

Questa è la pagina che si presenta quando viene cliccato il bottone per modificare un utente. Ci sono 4 campi da completare per modificare un utente. Come prima cosa bisogna scegliere l’utente da modificare dal menu a tendina. Poi bisogna procedere all’inserimento obbligatorio dell’indirizzo email e della password. Questi campi devono essere riempiti anche se sono uguali a quelli precedenti.

|  |
| --- |
|  |

Pagina principale inventario

Questa è la pagina che si presenta quando viene cliccato il bottone ‘Inventario HW’ nella pagina home. In questa pagina è possibile vedere tutti gli oggetti presenti nel magazzino, aggiungerli, modificarli e toglierli. Inoltre è possibile selezionare una categoria per decidere quali oggetti visualizzare. Queste categorie vengono utilizzate come filtri.

|  |
| --- |
|  |

Pagina aggiunta oggetto

Questa è la pagina che si presenta quando si clicca il bottone ‘+’ nella pagina principale dell’inventario. In questa pagina tutti i campi sono obbligatori. Come prima cosa bisogna inserire il tipo selezionandolo nel menu a tendina. Bisogna inserire anche una descrizione abbastanza esplicativa ed il numero seriale, solitamente scritto su un’etichetta incollata sul dispositivo. Dopo aver compilato tutti i campi bisogna cliccare il bottone ‘Applica’ per rendere effettiva l’aggiunta di un oggetto.

# Implementazione

## Database

|  |
| --- |
|  |

Questa tabella serve per salvare un utente con tutte le sue informazioni. Il nome utente è auto generato con il nome ed il cognome dell’utente. Quando si va ad inserire la password viene calcolato l’hash e salvato nel database, in modo da non compromettere la sicurezza dell’utente.

|  |
| --- |
|  |

Questa tabella serve per salvare un oggetto con tutte le sue particolarità. Ha una colonna ‘active’ per sapere se un oggetto è attivo oppure è stato disattivato, quindi archiviato. Le colonne ‘user\_id’ e ‘riservation\_date’ non vengono completate quando si inserisce un oggetto. Quelle colonne si riempiono quando si riserva un oggetto.

## Libs

### Collegamento

Per collegarsi al database viene usato un file nella cartella libs. Il file si chiama ‘Connection.php’.

|  |
| --- |
|  |

Viene utilizzato questo codice per aprire una connessione verso il database. Nelle variabili ‘$servername’, ‘$username’, ‘$password’ e ‘$dbname’ vengono salvati i rispettivi dati. Nella variabile ‘$conn’ viene creato un oggetto di tipo mysqli. Se non è possibile creare una connessione con il database viene chiusa.

### Hash

|  |
| --- |
|  |

Questa classe serve per cifrare le password. Quando si istanzia un oggetto di tipo ‘Hash’ bisogna passare come argomento la password in chiaro. C’è il metodo ‘cipherPassword’ al quale bisogna passare l’algoritmo che si vuole usare ed il sale da aggiungere alla password cifrata. In questo modo si salva la password cifrata nella variabile ‘cipherPass’. Per poter leggere la password cifrata bisogna usare il metodo ‘getCipherPass’, il quale ritorna la password cifrata. Questa classe è utile per cifrare una password, la quale può essere inserita nel database o usata per controllare se il nome utente e la password sono corretti.

### View

|  |
| --- |
|  |

Nella classe ‘View’ c’è uno dei metodi più utilizzati in questo sito web. Il metodo è il metodo ‘render’. Come parametri obbligatori c’è il nome del file che vogliamo fare visualizzare all’utente. Invece come parametri opzionali c’è il parametro ‘noInclude’ che permette di fare visualizzare la pagina senza l’header ed il footer. È utile quando bisogna inserire una pagina dentro un iframe. In questo caso non vogliamo che l’utente possa navigare liberamente nel sito dall’iframe, potrebbe portare ad avere loop indesiderati. In questo modo è possibile ottenere solo il body della pagina richiesta.

## Views

### HeaderAdmin

|  |
| --- |
|  |

Questo è il div dell’header admin. Come aggiunta all’header da non loggato ha il link ‘Logout’. Questo link va a richiamare il controller ‘logout’ per eseguire il logout.

## Controllers

### Home

|  |
| --- |
|  |

Questa classe viene richiamata per entrare nella pagina home. C’è la funzione index, che viene chiamata se non è specificata la funzione da eseguire o se la funzione specificata non esiste. Questa funzione apre la pagina home sul browser.

La funzione ‘action’ permette ai vari bottoni presenti sul browser di eseguire delle azioni diverse l’uno dall’altro. Un’azione può essere quella di voler passare alla pagina principale dell’inventario. Un’altra è quella di passare alla pagina per la modifica degli utenti. Da questa pagina si può passare alle pagine per aggiungere, modificare o rimuovere un utente. In questo caso i messaggi dal browser al server vengono mandati con il metodo ‘POST’.

### Login

|  |
| --- |
|  |

Questa classe serve per fare il login. La funzione ‘index’ apre il form di login sul browser.

La funzione ‘login’ serve per richiamare la classe nella cartella ‘models’ e controllare se nel database c’è un record con quel nome utente e password. Nel costrutto try catch si istanzia un nuovo ‘Login\_Model’ e si prova ad effettuare il login. Se il metodo login ritorna true, vuol dire che l’accesso è stato eseguito correttamente e quindi mostra la pagina home. Invece se solleva un’eccezione, fa rivedere il form di login con l’errore scritto al di sotto.

### Logout

|  |
| --- |
|  |

Questa classe serve per effettuare il logout. La funzione ‘index’ permette di eseguire il logout. Per effettuare il logout bisogna prima cancellare il contenuto della variabile di sessione e dopo cancellare completamente questa variabile. Dopo aver cancellato questa variabile fa vedere la pagina di disconnessione.

### Register

|  |
| --- |
|  |

Questa classe permette di registrare un nuovo utente. La funzione index controlla se chi sta tentando di accedere è autenticato. Se è autenticato fa vedere la pagina per registrare un utente, altrimenti fa vedere la pagina di login.

La funzione ‘register’ utilizza il costrutto try catch per gestire le eccezioni. Come prima cosa si istanzia un’istanza di ‘Login\_Model’. Dopo si esegue il metodo ‘register’ della classe appena istanziata. Se questa operazione va a buon fine viene mostrata la pagina di successo della registrazione, altrimenti fa rivedere la pagina di registrazione con scritto sotto l’errore.

### Modifyuser

|  |
| --- |
|  |

Questa classe permette di modificare un utente.

Nella funzione ‘index’ viene controllato se la richiesta è stata fatta da un utente autenticato o meno. Se è autenticato viene mostrata la pagina con la modifica dell’utente, altrimenti viene mostrata la pagina con il form di login.

La funzione ‘modifyuser’ utilizza il costrutto try catch per gestire le eccezioni. Come prima cosa viene creata un’istanza di ‘ModifyUser\_Model’. Dopo viene chiamato il metodo ‘modifyuser’ della classe appena istanziata. Se la richiesta va a buon fine viene mostrata la pagina di successo della modifica, altrimenti viene rimostrato il form di modifica con l’errore scritto al di sotto.

### Deleteuser

|  |
| --- |
|  |

Questa classe permette di eliminare un utente.

La funzione ‘index’ mostra la pagina per eliminare un utente.

La funzione ‘getUsers’ istanzia ‘User\_Model’ e mette l’output del metodo ‘getUsers’ della classe ‘User\_Model’ nella variabile ‘users’ contenuta nella view.

La funzione ‘deleteuser’ utilizza il costrutto try catch per gestire le eccezioni. Come prima cosa istanzia ‘User\_Model’. Se il metodo ‘deleteuser’ della classe ‘User\_Model’ ritorna true, viene mostrata la pagina di successo dell’eliminazione, altrimenti viene rimostrata la pagina di elimina con l’errore al di sotto.

### Iframe

|  |
| --- |
|  |

Questa classe permette di visualizzare le varie pagine dell’inventario.

La funzione ‘index’ mostra la pagina nell’iframe dell’inventario.

La funzione ‘getObject’ riceve un parametro che viene usato per trovare la categoria da far vedere. Se il ‘typeid’ è maggiore di 0 si crea una nuova istanza di ‘Inventory\_Model’. Alla variabile ‘objects’ viene assegnato il risultato della funzione ‘getObject’ della classe ‘Inventory\_Model’. Infine si richiama il metodo index della classe ‘Iframe’.

### Inventory

|  |
| --- |
|  |

La classe ‘Inventory’ permette di visualizzare l’inventario nella sua interezza.

La funzione ‘index’ mostra la pagina dell’inventario.

La funzione ‘getPage’ controlla che il ‘typeid’ che è stato passato come parametro non sia vuoto e che sia maggiore di -2. Se ‘typeid’ soddisfa i requisiti si crea un’istanza di ‘Iframe’ per avere la visualizzazione completa della pagina. Dopo si assegna alla variabile ‘link’ il valore di ‘typeid’ e si richiama la funzione ‘index’.

La funzione ‘action’ controlla quale bottone è stato premuto e su che linea. Per prima cosa si controlla se la variabile POST[‘button’] è settata e dopo si controlla quale valore abbia. Se il valore è ‘Delete’, si assegna ‘id’ alla view e si mostra la pagina di conferma di eliminazione.

Se invece il valore della variabile POST è ‘Modify’ viene creata un’istanza di ‘Inventory\_Model’, assegnato il risultato della funzione ‘getClassroom’ alla variabile ‘classroom’ e assegnato il risultato della funzione ‘getSingleObject’ alla variabile ‘object’. Dopo viene mostrata la pagina per la modifica.

Se invece il valore della variabile POST è ‘Add’ viene creata un’istanza di ‘Inventory\_Model’, assegnato il risultato della funzione ‘getClassroom’ alla variabile ‘classroom’ e assegnato il risultato della funzione ‘getTypes’ alla variabile ‘types’. Dopo viene mostrata la pagina per aggiungere un oggetto.

|  |
| --- |
|  |

La funzione ‘confirmDelete’ controlla se l’oggetto che si sta eliminando è veramente da eliminare. Controlla se il bottone cliccato ha come valore ‘Confirm’. Se si crea un’istanza di ‘Inventory\_Model’ e richiama il metodo delete. Successivamente mostra la pagina di successo dell’eliminazione.

Se il valore è ‘Back’ torna semplicemente alla pagina principale dell’inventario.

La funzione ‘modify’ permette di modificare un oggetto. Utilizza il costrutto try catch per gestire le eccezioni. Crea una nuova istanza di ‘Inventory\_Model’ e prova ad eseguire il metodo ‘modify’. Se ritorna true mostra la pagina di successo di modifica, altrimenti fa rivedere la pagina di modifica con l’errore al di sotto.

La funzione ‘add’ permette di aggiungere un oggetto all’inventario. Utilizza il costrutto try catch per gestire le eccezioni. Crea una nuova istanza di ‘Inventory\_Model’ e prova ad eseguire il metodo ‘add. Se ritorna true mostra la pagina di successo di aggiunta, altrimenti fa rivedere la pagina di aggiunta con l’errore al di sotto.

## Models

### Login\_Model

|  |
| --- |
|  |

Prima di fare qualsiasi cosa viene inizializzata la variabile session. Come viene mostrato nell’immagine, la variabile ‘session’ viene inizializzata prima della dichiarazione della classe.

|  |
| --- |
|  |

Nel costruttore viene controllato se la variabile ‘POST[‘username’]’ e la variabile ‘POST[‘password’]’ sono settati. Se non sono settati solleva un’eccezione con l’errore “Username o password non inseriti”.

Altrimenti controlla se il nome utente è maggiore di 3 e la password è maggiore di 7 caratteri. Se non fosse così, viene sollevata un’eccezione con l’errore “Username o password non corretti”. Viene dato un errore non specifico in modo da non dare indizi ad un eventuale malintenzionato che vuole accedere al sito.

Se vengono passati i controlli, username e password vengono assegnate alle rispettive variabili.

|  |
| --- |
|  |

Questa funzione viene utilizzata per controllare se lo username passato è valido. Per essere valido deve avere il carattere ‘.’ (punto) al suo interno. Lo username è sempre composto dal nome concatenato al cognome, separati da un punto. In caso lo username non fosse valido viene sollevato un’eccezione con l’errore “Username non valido”. Altrimenti viene ritornata la posizione del carattere ‘.’(punto).

|  |
| --- |
|  |

Questa funzione permette di estrarre il nome dell’utente dallo username. Come prima cosa viene richiamato il metodo ‘isUsernameValid’ in modo da controllare se il nome utente è valido e poter ottenere la posizione del carattere ‘.’(punto). Dopo viene ricavato il nome utilizzando la funzione ‘substr’ passando il nome utente come parametro, insieme la posizione di partenza e la posizione di arrivo.

|  |
| --- |
|  |

Questo metodo è il metodo principale per effettuare il login.

Come prima cosa vengono importate delle classi che servono per il corretto funzionamento del codice. Bisogna importare ‘libs/Hash.php’ e ‘libs/Connection.php’. Per cercare l’utente nel database bisogna avere la password cifrata. Per cifrarla viene usata la classe ‘Hash’.

Nella query vengono selezionati solo gli utenti che hanno gli stessi username e password passati dalla variabile ‘POST’. Dopo viene controllato se le righe risultate dalla query sono uguali a 1. Se è vero viene impostata la variabile ‘$\_SESSION’ con lo ‘userid’ e il flag per verificare che sia amministratore. Se il login va a buon fine ritorna true, altrimenti false.

### Register\_Model

|  |
| --- |
|  |

La funzione costruttore controlla che tutte le variabili siano settate. Se è vero, assegna ad ogni variabile il valore corretto, altrimenti solleva un’eccezione con l’errore “Compilare tutti i campi”.

|  |
| --- |
|  |

Queste funzioni sono dei getter. Vengono utilizzati per controllare meglio l’accesso ai dati. L’unico compito è quello di ritornare un attributo.

|  |
| --- |
|  |

Queste funzioni sono dei setter. Il loro scopo è quello di filtrare i dati che vengono inseriti, se c’è la necessità. Nell’immagine sopra si possono notare due funzioni che hanno un filtro all’interno. Se i parametri di queste funzioni soddisfano i requisiti impostati nell’if, il parametro viene assegnato all’attributo, altrimenti viene sollevata un’eccezione con l’errore “Inserire un nome/cognome valido”.

|  |
| --- |
|  |

Questa funzione permette di verificare se l’attributo ‘pass1’ e l’attributo ‘pass2’ sono uguali. In tal caso ritorna true.

|  |
| --- |
|  |

Questa funzione permette di ritornare una parola con la prima lettera maiuscola e tutte le altre minuscole.

|  |
| --- |
|  |

Questa funzione permette di registrare un utente.

Come prima cosa controlla che le due password siano uguali. Se non sono uguali solleva un’eccezione con l’errore “Le due password non corrispondono”, altrimenti controlla se le password soddisfano i criteri di sicurezza. Se non li soddisfano solleva un’eccezione con l’errore “Complessità password non valida”, altrimenti procede con la registrazione.

Come prima cosa viene calcolato l’hash della password. Dopo viene creato il nome utente concatenando il nome ed il cognome, separati dal carattere ‘.’(punto).

Nella query viene fatta una select per tutti i record che hanno come nome utente quello calcolato in precedenza.

Se le righe ritornate dalla query sono maggiori di 0 viene sollevata un’eccezione con l’errore “Utente esistente”, altrimenti viene eseguita la query per l’aggiunta dell’utente. Infine ritorna true.

### Modifyuser\_Model

|  |
| --- |
|  |

Questa funzione permette di ottenere una lista a tendina html pronta da essere inserita in una pagina web. Come prima cosa fa una query di tutti i nomi utente e dopo crea il pattern per la select.

|  |
| --- |
|  |

Questa funzione permette di modificare l’email e la password di un utente. Viene controllato se entrambe le password coincidono. Se non coincidono viene sollevata un’eccezione con l’errore “Le due password non corrispondono”, altrimenti controlla la complessità delle password. Se le password non sono abbastanza complesse viene sollevata un’eccezione con l’errore “Complessità password non valida”, altrimenti si cerca il nome ed il cognome dell’utente utilizzando una query.

Dopo viene calcolato l’hash della password.

Infine viene aggiunto il nuovo utente con un’altra query.

### Inventory\_Model

|  |
| --- |
|  |

Questa funzione ritorna un’array contenente i tipi dalla tabella ‘type’.

Viene effettuata una query dove vengono presi gli attributi ‘id’ e ‘name’.

|  |
| --- |
|  |

Questa funzione ritorna un array contenente gli oggetti i una specifica categoria e attivi. Si prendono solo gli oggetti attivi perché quelli non attivi sono archiviati. Nella pagina dell’inventario si devono vedere solo gli oggetti non archiviati.

|  |
| --- |
|  |

Questa funzione ritorna un array di lunghezza 1 che contiene l’oggetto con l’id ‘$id’. Questo viene applicato con la clausola ‘WHERE’ nella query.

|  |
| --- |
|  |

Questa funzione ritorna un array contenente tutte le aule.

Come prima cosa viene importato il file ‘Connection.php’. Dopo viene eseguita la query sql e inserita in un array. Alla fine viene ritornato questo array.

|  |
| --- |
|  |

Questa funzione ritorna un array contenente tutti gli elementi non attivati, quindi archiviati. È utile per mostrare la pagina con tutti gli oggetti eliminati.

|  |
| --- |
|  |

Questa funzione permette di ‘eliminare’ un oggetto. Non viene eliminato un oggetto in modo da avere uno storico di tutti gli oggetti passati dall’inventario. Quindi questa funzione non elimina effettivamente l’oggetto dall’inventario, ma lo archivia.

|  |
| --- |
|  |

Questa funzione permette di modificare un oggetto dell’inventario. Prima di tutto viene controllato se la variabile ‘POST[‘class’]’ è settata ed è maggiore di 0. Dopo viene controllato se la variabile ‘POST[‘reserved’]’ è settata. Se è settata imposta lo userid con l’id dello user corrente preso dalla variabile session e come data imposta quella corrente, altrimenti gli setta a null.

Dopo viene creata la query sql dove vengono impostati i valori nuovi e viene eseguita.

|  |
| --- |
|  |

Questa funzione permette di aggiungere un oggetto all’inventario. Come prima cosa controlla che la variabile POST sia settata correttamente. Dopo viene scritta una query sql per aggiungere un nuovo elemento. Infine viene eseguita. In caso che la variabile POST non sia settata correttamente viene sollevata un’eccezione con l’errore “Complete all fields”.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Apache e php funzionante |
| **Descrizione:** | Si controlla se apache funziona sulla porta 80. | | |
| **Prerequisiti:** | * Avere apache installato * Avere php installato | | |
| **Procedura:** | 1. Creare un file .php e scrivere dentro “<?php phpinfo(); ?>” 2. Mettere il file nella cartella dei file web di apache 3. Aprire il browser 4. Cercare “localhost” | | |
| **Risultati attesi:** | Si apre una pagina con molte informazioni su php. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Database funzionante |
| **Descrizione:** | Si controlla se è possibile il collegamento al database | | |
| **Prerequisiti:** | * Avere apache funzionante sulla porta 80 * Avere php funzionante * Avere mysql installato | | |
| **Procedura:** | 1. Creare un file .php e scrivere dentro il seguente codice:   <?php $servername = "localhost"; $username = <username>; $password = <password>;  // Create connection $conn = new mysqli($servername, $username, $password);  // Check connection if ($conn->connect\_error) {   die("Connection failed: " . $conn->connect\_error); } echo "Connected successfully"; ?>   1. Mettere il file nella cartella dei file web di apache 2. Aprire il browser 3. Cercare “localhost” | | |
| **Risultati attesi:** | Si apre una pagina con scritto “Connected successfully” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Login |
| **Descrizione:** | Si controlla se è possibile effettuare il login. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso | | |
| **Procedura:** | 1. Accedere alla pagina di login 2. Inserire i dati di accesso 3. Premere il bottone ‘Login’ | | |
| **Risultati attesi:** | Si apre la pagina home. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Accesso al servizio con un utente amministratore |
| **Descrizione:** | Si controlla se è possibile effettuare l’accesso con i privilegi di amministratore. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso * Avere un utente amministratore con il quale fare l’accesso | | |
| **Procedura:** | 1. Accedere alla pagina di login 2. Inserire i dati di accesso di un amministratore 3. Premere il bottone ‘Login’ | | |
| **Risultati attesi:** | Si apre la pagina home con il bottone per gestire gli utenti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-005 | **Nome:** | Pagina home |
| **Descrizione:** | Si controlla se è possibile visualizzare una pagina home. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso | | |
| **Procedura:** | 1. Effettuare il login. | | |
| **Risultati attesi:** | Si apre la pagina home dell’utente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Filtrare I componenti |
| **Descrizione:** | Si controlla se è possibile filtrare i componenti. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso * Avere un utente amministratore con il quale fare l’accesso * Avere degli oggetti nell’inventario | | |
| **Procedura:** | 1. Effettuare l’accesso. 2. Spostarsi nella pagina principale dell’inventario. 3. Scegliere l’attributo per il quale si vuole filtrare. 4. Premere il tasto ‘Filtra’. | | |
| **Risultati attesi:** | Vengono mostrati i dati filtrati., | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-009 | **Nome:** | Creazione di nuovi utenti |
| **Descrizione:** | Si controlla se è possibile aggiungere un nuovo utente. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso * Avere un utente amministratore con il quale fare l’accesso | | |
| **Procedura:** | 1. Effettuare l’accesso. 2. Spostarsi nella pagina della gestione utenti. 3. Premere il bottone ‘Add’. 4. Completare tutti i campi. 5. Premere il bottone ‘Register’. | | |
| **Risultati attesi:** | Si apre la pagina di successo della registrazione. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-009 | **Nome:** | Modifica utenti |
| **Descrizione:** | Si controlla se è possibile modificare un utente. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso * Avere un utente amministratore con il quale fare l’accesso | | |
| **Procedura:** | 1. Effettuare l’accesso. 2. Spostarsi nella pagina della gestione utenti. 3. Premere il bottone ‘Modify. 4. Completare tutti i campi. 5. Premere il bottone ‘Modify. | | |
| **Risultati attesi:** | Si apre la pagina di successo della modifica. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-009 | **Nome:** | Eliminazione utenti |
| **Descrizione:** | Si controlla se è possibile eliminare un utente. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso * Avere un utente amministratore con il quale fare l’accesso | | |
| **Procedura:** | 1. Effettuare l’accesso. 2. Spostarsi nella pagina della gestione utenti. 3. Premere il bottone ‘Delete. 4. Completare tutti i campi. 5. Premere il bottone ‘Confirm. | | |
| **Risultati attesi:** | Si apre la pagina di successo dell’eliminazione. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-005 | **Nome:** | Logout |
| **Descrizione:** | Si controlla se è possibile effettuare il logout. | | |
| **Prerequisiti:** | * Avere Apache funzionante * Avere il database con un utente con cui fare l’accesso | | |
| **Procedura:** | 1. Effettuare il login 2. Dalla pagina home premere il bottone ‘Logout’. | | |
| **Risultati attesi:** | Si apre la pagina di successo del logout. | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Esito** | **Risultati** | **Data** |
| TC-001 | **PASSATO** |  | 23.12.2021 |
| TC-002 | **PASSATO** |  | 23.12.2021 |
| TC-003 | **PASSATO** |  | 23.12.2021 |
| TC-004 | **PASSATO** |  | 23.12.2021 |
| TC-005 | **PASSATO** |  | 23.12.2021 |
| TC-006 | NON PASSATO | Questa funzione non è stata implementata per mancanza di tempo. | 23.12.2021 |
| TC-007 | **PASSATO** |  | 23.12.2021 |
| TC-008 | **PASSATO** |  | 23.12.2021 |
| TC-009 | **PASSATO** |  | 23.12.2021 |
| TC-010 | **PASSATO** |  | 23.12.2021 |
| TC-011 | NON PASSATO | Questa funzione non è stata implementata. | 23.12.2021 |

## Mancanze/limitazioni conosciute

Una delle mancanze conosciute è la soglia. Non è stata implementata perché richiedeva troppo tempo.

I filtri più specifici non sono stati applicati perché richiedevano troppo tempo. È presente una versione base dei filtri.

Un’altra limitazione conosciuta è che il sito non è responsive. Non è stato implementato perché non era richiesto nel progetto, ma al fine di avere un sito più utilizzabile potrebbe impiegare un progetto in futuro.

# Consuntivo

|  |
| --- |
|  |

# Conclusioni

## Sviluppi futuri

Una cosa interessante che si potrà implementare è una soglia, che permette di gestire il limite minimo di oggetti. Un’altra cosa potrebbe essere interessante è l’implementazione di filtri più precisi.

## Considerazioni personali

Da questo progetto ho imparato a lavorare con più facilità con php. Inoltre ho cominciato la mia esperienza con i progetti a lungo termine. Un’altra cosa che ho imparato è essere più preciso ed ordinato, in modo che quando sarei tornato la settimana dopo a continuare il progetto, sarei riuscito a riprendere senza ristudiare tutto quello che avevo scritto.

Infine ho imparato a gestire meglio il tempo. Ho cominciato a fare questo progetto con poche conoscenze delle tempistiche delle varie fasi. Alla fine di questo progetto ho una migliore conoscenza dei tempi necessari per svolgere le varie attività.

# Bibliografia

## Sitografia

* <https://www.w3schools.com/php/default.asp>, 16.12.2021
* <https://stackoverflow.com/>, 16.12.2021
* <https://www.php.net/>, 16.12.2021

# Allegati

Elenco allegati:

* Diari di lavoro
* QdC
* Codici in GitHub
* Gantt
* Design delle interfacce
* Design del database