
Candidato: Bryan Beffa
Azienda: SAMT
Periodo: 03.09.2019 - 20.12.2019
Presentazione: 07.01.2020 – 14.01.2020

Situazione iniziale

Il committente ha richiesto un'applicazione web utile per la gestione di casi di documentazione tecnica relativa all'informatica. Ogni caso di documentazione deve avere una categoria di assegnazione, un titolo, una descrizione, una data di creazione e modifica. Il sito web deve possedere un sistema di autenticazione (login), un sistema di gestione degli utenti (aggiunta, modifica ed eliminazione) alla quale solo gli amministratori possono accedere, un sistema di visualizzazione dei casi registrati, un sistema di registrazione dei casi alla quale tutti possono accedere, un sistema di modifica/eliminazione di un caso (solo gli admin possono utilizzare questa funzione), un sistema di ricerca di casi tramite dei campi di filtraggio (testo, data, categoria, modalità di ordinamento). Gli utenti con privilegi di tipo admin possono inoltre aggiungere e rimuovere le categorie.

L'applicativo deve essere semplice, efficace e user-friendly (intuitivo). Inoltre deve presentare un'interfaccia grafica gradevole.

Attuazione

Per lo sviluppo di questo applicativo mi sono affidato al framework mvc di php. Questo framework permette una divisione efficace delle varie parti di codice. In questo modo ho potuto separare le pagine di visualizzazione (views) dal codice php. Il codice è stato diviso in controller e models permettendo la separazione modulare delle varie pagine di codice. In questo modo ogni controller utilizza i metodi messi a disposizione dalle classi models (ad esempio per la connessione al database o per modificare i dati di una determinata tabella).

Per la parte grafica è stato utilizzato il framework material design bootstrap che mette a disposizione molte classi css e librerie javascript così da ottenere delle pagine web ordinarie e apprezzabili esteticamente.

Risultati

Gli obiettivi richiesti dal committente sono stati raggiunti. Tutte le funzionalità che sono state richieste ad inizio progetto sono state implementate nell'applicazione web. L'applicativo è stato progettato in modo da rendere l'esperienza di utilizzo il più semplici ed intuitiva possibile. Ci sono molte funzionalità che possono essere implementate in un eventuale sviluppo futuro, come ad esempio il cambio di password utilizzando un sistema mail server.

Knowledge Base

Titolo del progetto: Knowledge Base
Alunno/a: Bryan Beffa
Classe: Info I4AA
Anno scolastico: 2019/2020
Docente responsabile: Ivan Raimondi

1	Introduzione	4
1.1	Informazioni sul progetto	4
1.2	Abstract	4
1.3	Scopo	4
1.4	Analisi	5
1.5	Analisi del dominio	5
1.6	Analisi e specifica dei requisiti	5
1.7	Caso d'uso	9
1.8	Pianificazione	9
1.9	Analisi dei mezzi	11
1.9.1	Software	11
1.9.2	Hardware	11
2	Progettazione	12
2.1	Design dell'architettura del sistema	12
2.2	Design dei dati e database	13
2.2.1	Tabella users	13
2.2.2	Tabella cases	14
2.2.3	Tabella categories	14
2.2.4	Tabella representation	14
2.3	Design delle interfacce	15
2.3.1	Pagina di login	15
2.3.2	Pagina di gestione utenti	15
2.3.3	Pagina di visualizzazione casi	16
2.4	Design procedurale	17
3	Implementazione	18
3.1	Database	18
3.2	Framework mvc	18
3.2.1	Come funziona	19
3.2.2	Configurazione	20
3.2.3	Riflessione personale	20
Controllers	21
3.2.4	Home	21
3.2.5	DbError	22
3.2.6	Users	23
3.2.7	ResearchCases	26
3.3	Models	29
3.3.1	DbManager	29
3.3.2	Validator	30
3.3.3	PasswordManager	31
3.3.4	DocCase	32
3.3.5	User	33
3.3.6	CategoryManager	34
3.3.7	MessageManager	37
3.3.8	CaseManager	38
3.3.9	UserManager	43
3.4	Pagine	48
3.4.1	Pagina di login	48
3.4.2	Pagina di errore	49
3.4.3	Pagina di ricerca casi	50
3.4.4	Pagina gestione utenti	55
4	Test	59
4.1	Protocollo di test	59
4.2	Risultati test	66
4.3	Mancanze/limitazioni conosciute	67
5	Consuntivo	68
6	Conclusioni	71
6.1	Sviluppi futuri	71

6.2	Considerazioni personali.....	71
7	Bibliografia.....	71
7.1	Sitografia	71
8	Allegati.....	72

1 Introduzione

1.1 Informazioni sul progetto

Allievo coinvolto nel progetto: Beffa Bryan

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

Docenti responsabili: Ivan Raimondi

Data inizio: 03:09:2019

Data fine: 20:12:2019

1.2 Abstract

The teachers need a website application with a login system. In this web application, the users can view cases of technical documentation and sort them using filters. The cases can be filtered by category, last entered, most recurrent cases. There must be two types of users: the first is the administrator who can add or remove users using a “user management” page. The admin can also add, remove or modified cases and categories as well. The second type of users are the operators who can just view and filter cases and create a new case.

1.3 Scopo

Mi è stato richiesto di realizzare un applicativo web per la gestione di casi di documentazione tecnica per un'azienda di supporto informatico. Nell'applicativo devono essere presenti due tipi di utenti: amministratore e operatori (utenti base). L'amministratore possiede più privilegi che comprendono l'aggiunta, la modifica e l'eliminazione di utenti, la modifica, eliminazione e registrazione dei casi di documentazione tecnica, aggiunta ed eliminazione di una categoria. L'amministratore è l'unico tipo di utente che può raggiungere la pagina di gestione degli utenti. Gli operatori non possono registrarsi, hanno i privilegi per registrare nuovi casi di documentazione tecnica e possono visualizzare tutti i casi effettuando delle ricerche applicando alcuni filtri di ricerca. Solo gli amministratori possono effettuare le modifiche dei dati personali di un utente base.

1.4 Analisi

1.5 Analisi del dominio

Mi è stato richiesto di creare un applicativo web per gestire casi di documentazione di una ditta di informatica. Nel sistema sono definiti 2 tipi di utenti, ovvero amministratore ed operatori. Deve esserci la possibilità di effettuare delle ricerche relativi ai casi registrati. In base al tipo di utente si hanno diversi tipi di permessi. Gli operatori (utenti con privilegi base) possono inserire dei nuovi casi, ma non hanno la possibilità di eliminarli o modificarli. L'amministratore può registrare nuovi casi, eliminarli e modificarli. Inoltre possono aggiungere, eliminare gli utenti e modificare le loro informazioni personali. Possono anche eliminare ed aggiungere categorie. L'applicativo web deve essere user friendly, di facile utilizzo e con una grafica apprezzabile.

1.6 Analisi e specifica dei requisiti

Il committente richiede un applicativo web per la gestione di casi di documentazione tecnica per la propria azienda. Mi è stato richiesto di creare due tipi di utenza, l'amministratore, l'unico utente che può eliminare e modificare i casi di documentazione tecnica, e gli operatori (utenti con privilegi base), che possono visualizzare e registrare nuovi casi nel sistema. L'amministratore può svolgere tutte le funzioni, comprese quelle degli operatori. Deve essere presente un sistema di ricerca dei vari casi, che permette la ricerca e successivamente la visualizzazione dei risultati. Si devono poter visualizzare gli ultimi casi, i casi più ricorrenti ed in base alla ricerca effettuata dall'utente.

Vi deve essere anche un sistema di registrazione per i nuovi utenti alla quale solo l'admin può avere accesso, un sistema di login ed un sistema di cambio delle loro informazioni personali (compresa la password). L'applicativo deve essere un sistema user friendly, apprezzabile a livello grafico e di semplice utilizzo. Ogni caso appartiene ad una categoria, possiede la data di prima apparizione ed il numero di rappresentazioni. I casi possono avere delle varianti. Gli utenti admin possono inoltre aggiungere o eliminare le categorie.

ID: REQ-01

Nome	Pagina di login
Priorità	1
Versione	1.0
Note	La pagina deve presentare una mascherina per il login
Sotto requisiti	
001	Campo testuale per inserire l'indirizzo email
002	Campo testuale nascosto per inserire la password
003	Bottone "accedi" per effettuare il login
004	Messaggio di errore credenziali sbagliate
005	Messaggio di errore per problemi di connessione al database

ID: REQ-02

Nome	Pagina di gestione utenti (admin)
Priorità	1
Versione	1.0
Note	Si necessita una pagina di gestione degli utenti alla quale solo l'admin può accedere
Sotto requisiti	
001	Possibilità di inserire nuovo utente
002	Possibilità di eliminare un utente
003	Possibilità di richiedere un cambio password per un utente
004	Per ogni azione richiesta di conferma

ID: REQ-03

Nome	Pagina visualizzazione casi
Priorità	1
Versione	1.0
Note	Pagina in cui si possono visualizzare i casi
Sotto requisiti	
001	Maschera in cui si possono effettuare le varie ricerche relative ai casi esistenti
002	Bottone "aggiungi un caso" per registrare un nuovo caso
003	Bottone "elimina caso" (solo amministratore)
004	Bottone "modifica caso" (solo amministratore)
005	Bottone "elimina categoria" (solo amministratore)
006	Bottone "aggiunta categoria" (solo amministratore)

ID: REQ-04	
Nome	Pagina di registrazione casi
Priorità	1
Versione	1.0
Note	Pagina che permetta la registrazione di un nuovo caso
Sotto requisiti	
001	Campo testuale titolo
002	Campo testuale per la descrizione del caso
003	Possibilità di aggiungere una nuova categoria
004	Input per assegnare una categoria al caso

ID: REQ-05	
Nome	Ricerca casi
Priorità	1
Versione	1.0
Note	Sistema di ricerca casi di documentazione tecnica
Sotto requisiti	
001	Possibilità di visualizzare ultimi casi
002	Possibilità di visualizzare casi più ricorrenti
003	Bottone “registra nuovo caso”
004	Bottone registra nuova categoria (admin)
005	Bottone “logout” per effettuare la disconnessione del account corrente

ID: REQ-06

Nome	Sistema di registrazione
Priorità	1
Versione	1.0
Note	Scrittura del codice lato server per il sistema di registrazione di un utente
Sotto requisiti	
001	Pagina di registrazione e database

ID: REQ-07

Nome	Sistema di login
Priorità	1
Versione	1.0
Note	Scrittura del codice lato server per il sistema di login
Sotto requisiti	
001	Pagina di login e database

ID: REQ-08

Nome	Sistema di aggiunta/eliminazione categoria (admin)
Priorità	1
Versione	1.0
Note	L'admin può aggiungere o eliminare una categoria
Sotto requisiti	
001	Campo testuale in cui inserire il nome della categoria
002	Bottone per eliminare una categoria

1.7 Caso d'uso

Questo schema rappresenta il caso d'uso dell'applicativo web evidenziando i vari ruoli ed il loro comportamento. Come si può notare ogni viene mostrato il soggetto (admin, operatori, database, sistema) e le rispettive funzionalità oltre al come sono collegate tra loro.

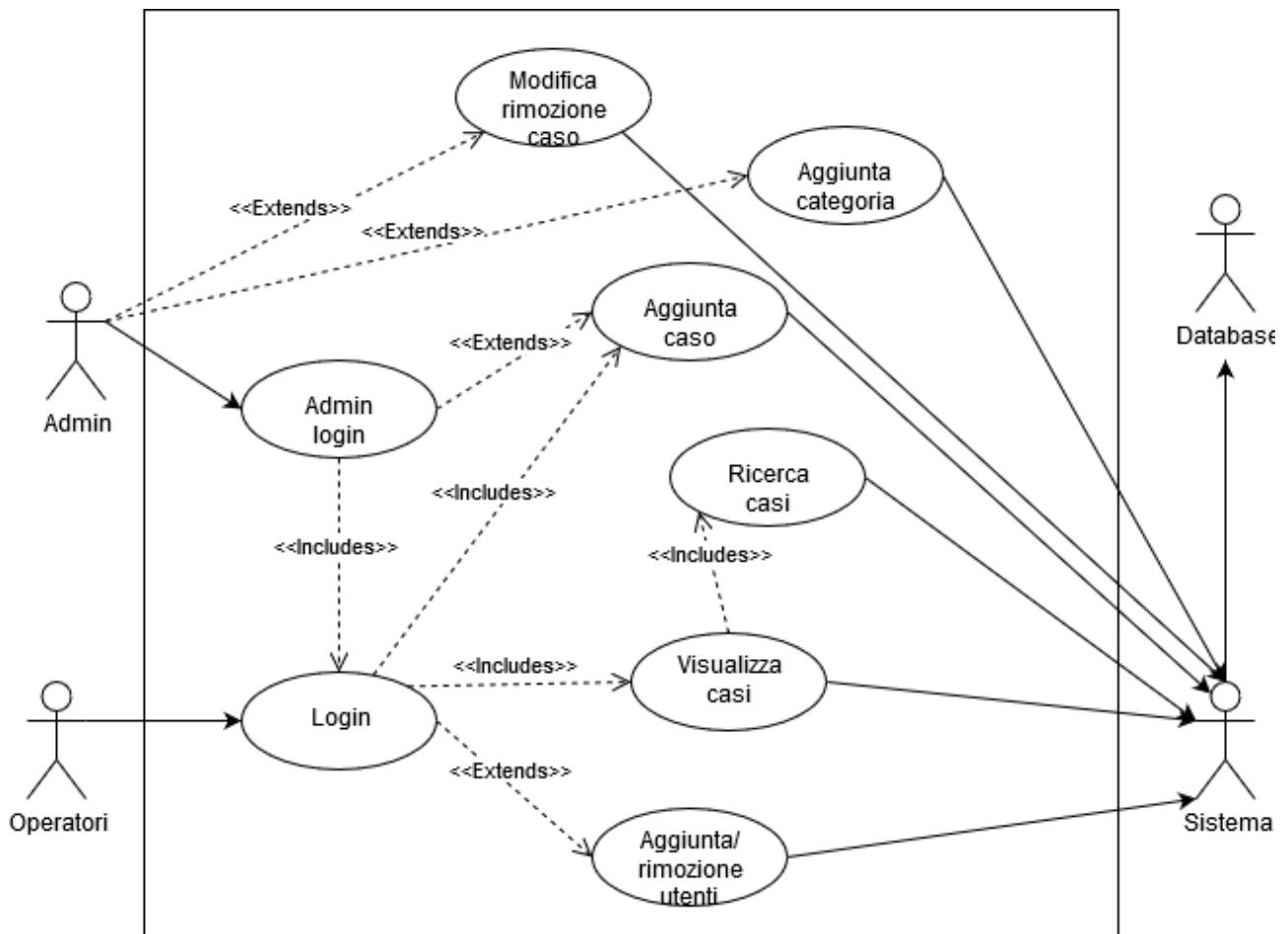


Figura 1 Caso d'uso

1.8 Pianificazione

Ho diviso il progetto in 4 fasi principali. La prima fase è quella di progettazione dove vengono svolte attività incentrate sull'analisi del progetto. La seconda fase è quella di sviluppo, dove avviene la stesura effettiva del codice delle pagine, script. La terza attività è il momento in cui vengono testati le varie parti di progetto. L'ultima fase, che copre tutta la durata del progetto, è la scrittura della documentazione. Ho utilizzato un diagramma di Gantt per gestire il mio progetto.

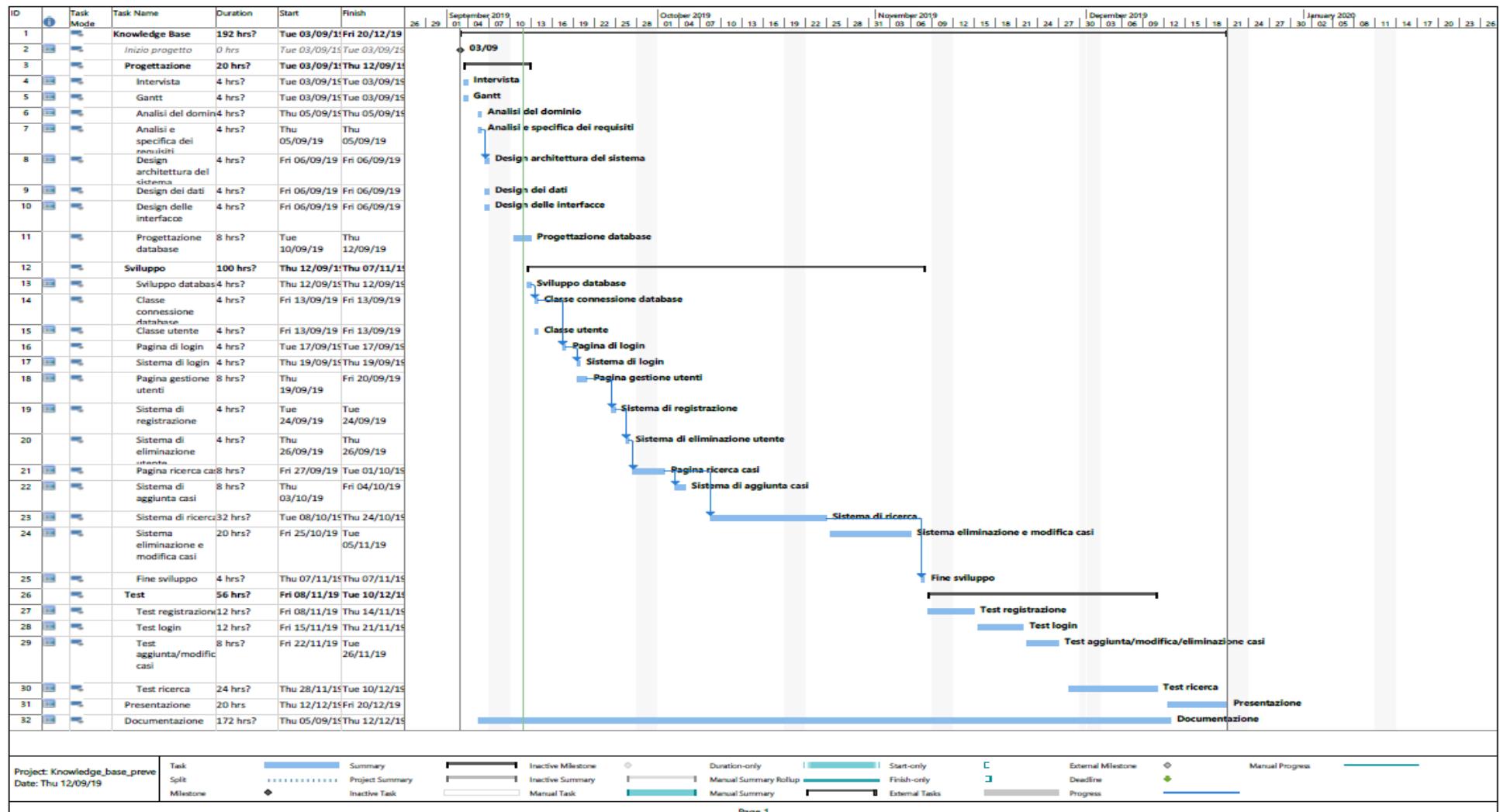


Figura 2 Diagramma di Gantt

1.9 Analisi dei mezzi

1.9.1 Software

I software che sono stati utilizzati per la realizzazione di questo progetto sono:

- JetBrains PhpStorm 2018.3.4
- Microsoft Word 2015
- Microsoft Powerpoint 2015
- Microsoft Project 2016 Professional
- XAMPP 7.3.9
- Php 7.1.33
- XDebug 2.6.1
- MySQL 15.1
- MySQL Workbench 8.0.17 CE

Il progetto è stato testato sui seguenti browser:

- Google Chrome 78.0.0
- Microsoft Edge 14.11.2019 → alcuni problemi di grafica ma nessuna conseguenza sulla funzionalità
- Mozilla Firefox 70.0.1 → animazione di caricamento ferma, il resto funziona correttamente

1.9.2 Hardware

L'intero progetto è un sito web, quindi non necessitavo di hardware specifico ed ho eseguito il tutto sul mio computer portatile Asus con sistema operativo windows 10.

2 Progettazione

In questo capitolo è rappresentato e spiegato come ho deciso di progettare l'intero sistema. Ho progettato tutto il sistema nel modo che ritenevo più efficace e coerente per garantire un prodotto che superasse le aspettative.

Ho cercato di rendere la navigazione e la struttura delle pagine il più semplice possibile e pulita possibile.

2.1 Design dell'architettura del sistema

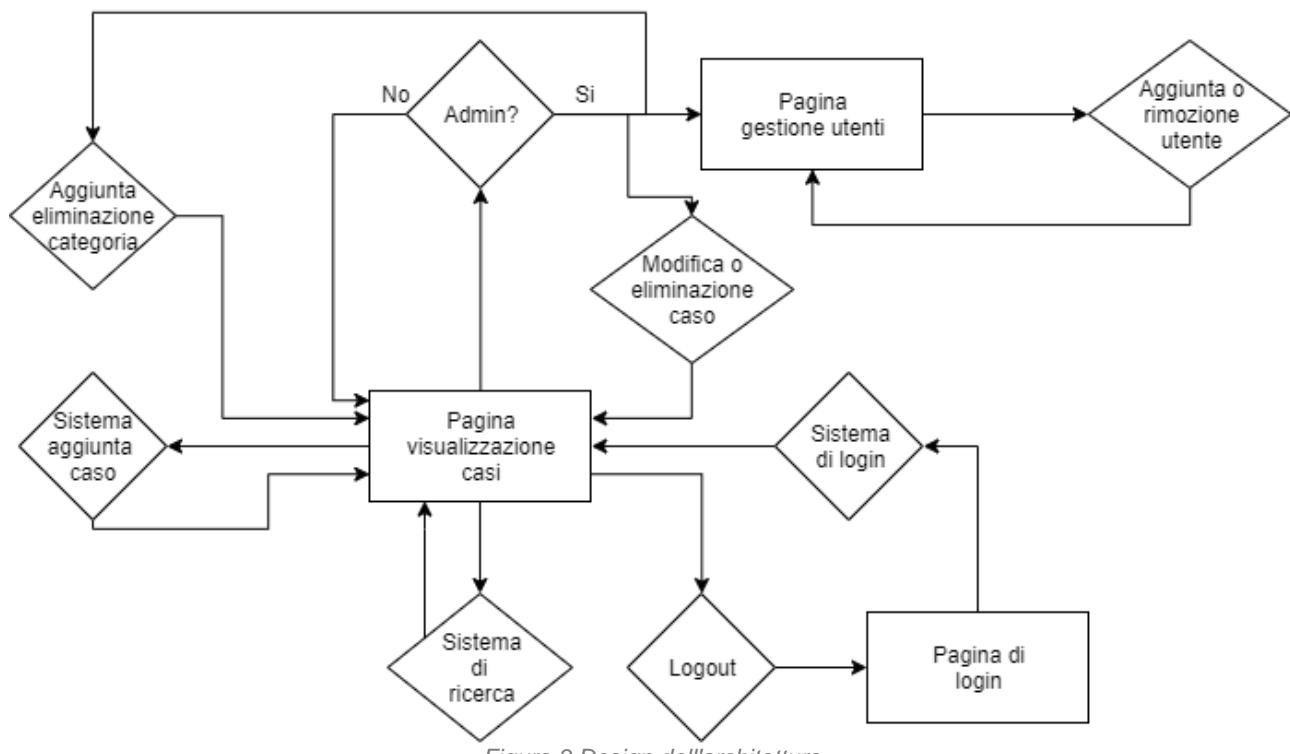


Figura 3 Design dell'architettura

L'utente parte dalla pagina di login, dove può effettuare l'accesso se è in possesso di credenziali corrette. Una volta effettuato l'accesso con delle credenziali valide l'utente viene reindirizzato alla pagina di visualizzazione casi.

Qui l'utente può effettuare delle ricerche tramite il sistema di ricerca casi. In questa pagina possono essere aggiunti nuovi casi.

Se l'utente possiede i privilegi amministrativi, può aggiungere ed eliminare le categorie. Inoltre può accedere alla pagina di gestione utente, in cui ha la possibilità di aggiungere, eliminare utenti.

2.2 Design dei dati e database

Dopo aver analizzato gli elementi che devono essere gestiti all'interno dell'applicativo ho progettato questo schema in cui sono presenti quattro tabelle **users**, **case**, **category** e **rappresentations**.

Questo database può essere modificato ed essere dettagliato da chiunque prenda in mano questo progetto in un secondo momento, così da poter rendere l'applicativo ancora più completo.

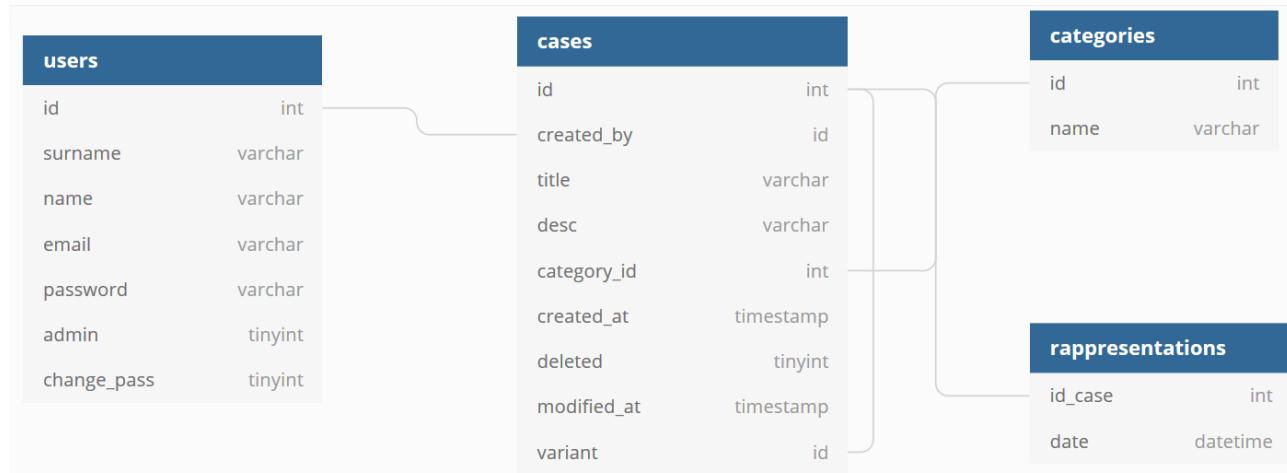


Figura 4 Schema del database

2.2.1 Tabella users

Nella tabella **users** ho inserito gli attributi che definiscono un modello di utente. Per gli utenti ho inserito solo le informazioni principali che mi servono all'interno del applicativo.

Attributo	Descrizione
Id	Numero identificativo dell'utente, il tipo di dato è un int
Name	Nome dell'utente, il tipo di dato è varchar
Surname	Cognome dell'utente, il tipo di dato è varchar
Email	Email dell'utente, varchar
Password	Password criptata dell'utente, varchar
Change_pass	Tinyint, definisce è stato richiesto un cambio di password dell'utente
Admin	Tinyint, definisce se un utente è admin o meno

2.2.2 Tabella cases

La tabella **cases** definisce il modello di “caso di documentazione tecnica”. Ogni caso deve avere i seguenti attributi.

Nella tabella seguente vi sono gli attributi e la loro descrizione:

Attributo	Descrizione
Id	Numero identificativo del caso, il tipo di dato è un int
Created_by	Foreign key, definisce l'utente che ha registrato il caso
Title	Tipo di dato varchar, indica il titolo del caso
Descr	Descrizione del caso
Category_id	Foreign key che definisce la categoria del caso
Created_at	Timestamp, indica la data e l'ora della creazione del caso
Deleted	Indica se il caso deve essere visualizzato dagli utenti
Modified_at	Timestamp, quando è stato modificato il caso
Variant	Int, id del caso a cui fa riferimento, campo non obbligatorio

2.2.3 Tabella categories

La tabella **categories** definisce un modello di “categoria”. Per ogni categoria vi è solo il numero identificativo della categoria ed il suo nome.

Attributo	Descrizione
Id	Numero identificativo category, il tipo di dato è un int
Name	Nome della categoria, tipo di dato varchar

2.2.4 Tabella representation

La tabella **representation** è un modello di rappresentazione, ogni caso viene salvato all'interno di questa tabella in modo da sapere ogni caso quante volte è apparso e quando.

Se un caso ha una variante, questa viene inserita all'interno della tabella con la data di apparizione. Questa tabella è pensata per un'implementazione futura in cui gli utenti di tipo admin possano vedere quando sono stati aggiunti e creati casi che possiedono una variante.

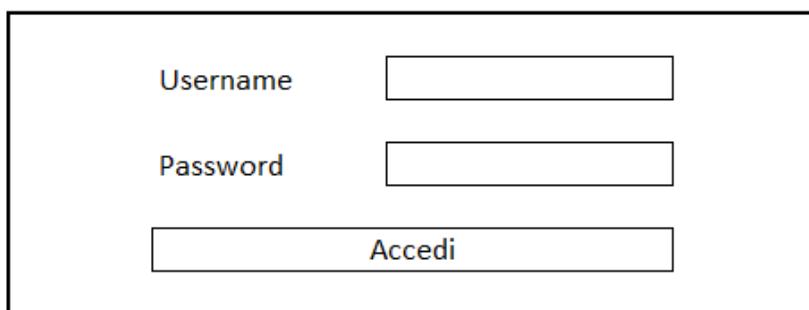
Attributo	Descrizione
Id_case	Id del caso
Date	Timestamp, data e ora di quando è stato registrato il caso
Id_variant	Id della variante

Nota: Durante lo sviluppo del progetto mi sono accorto che questa tabella non era necessaria per le funzionalità richieste, ma in futuro può essere utilizzata per salvare la data delle modifiche. Per il momento è utilizzata solo per salvare i casi che vengono registrati con la relativa variante.

2.3 Design delle interfacce

2.3.1 Pagina di login

La pagina di login è molto semplice, presenta un campo testuale per inserire il nome dell'utente, un campo testuale nascosto per inserire la password ed infine il bottone “accedi”.



The diagram shows a simple login interface. It consists of three input fields and one button. The first field is labeled "Username" and contains a placeholder for text. The second field is labeled "Password" and contains a placeholder for text. Below these two fields is a large rectangular button with the word "Accedi" centered in blue text.

Figura 5 Design interfaccia di login

2.3.2 Pagina di gestione utenti

La pagina di gestione utenti è raggiungibile solo dagli utenti amministratori. La pagina presenta una struttura molto semplice, una griglia in cui si possono visualizzare tutte le informazioni dei utenti. L'unico valore non visualizzabile è la richiesta di password, al suo posto vi è un bottone per (campo “richiesta cambio password”) che serve ad effettuare una richiesta di cambio password.



The diagram shows a user management interface. It features a table with columns for ID, Nome, Cognome, Email, Password, Richiesta cambio password, Admin, and Elimina. A single row is shown with values: ID 34, Nome Andrea, Cognome Bianchi, Email aa@aa.aa, Password sd2%2ed2, Richiesta cambio password X, Admin No, and Elimina X. Below the table is a button labeled "Aggiungi nuovo utente".

ID	Nome	Cognome	Email	Password	Richiesta cambio password	Admin	Elimina
34	Andrea	Bianchi	aa@aa.aa	sd2%2ed2	X	No	X

Figura 6 Design interfaccia gestione utenti

2.3.3 Pagina di visualizzazione casi

La pagina di visualizzazione casi permette a tutti gli utenti (admin ed operatori) di visualizzare i vari casi di documentazione tecnica. A questa pagina possono accedere sia gli operatori che l'admin. L'admin ha la possibilità di modificare o cancellare un caso.

Layout admin:

Possibilità di eliminare e modificare un caso.

Utenti	Casi																												
Filtro casi																													
<table border="1"><tr><td>Caso:</td><td>titolo</td><td>Created</td><td>xx.xx.xxxx</td></tr><tr><td>Categoria:</td><td>categoria</td><td></td><td></td></tr><tr><td>Variante di</td><td>Case_id</td><td></td><td></td></tr><tr><td colspan="4">Descrizione</td></tr><tr><td colspan="2"><table border="1"><tr><td>Delete</td><td>Times: 4</td></tr><tr><td>Modifica</td><td></td></tr></table></td><td></td><td></td></tr><tr><td colspan="4">Aggiungi caso</td></tr></table>		Caso:	titolo	Created	xx.xx.xxxx	Categoria:	categoria			Variante di	Case_id			Descrizione				<table border="1"><tr><td>Delete</td><td>Times: 4</td></tr><tr><td>Modifica</td><td></td></tr></table>		Delete	Times: 4	Modifica				Aggiungi caso			
Caso:	titolo	Created	xx.xx.xxxx																										
Categoria:	categoria																												
Variante di	Case_id																												
Descrizione																													
<table border="1"><tr><td>Delete</td><td>Times: 4</td></tr><tr><td>Modifica</td><td></td></tr></table>		Delete	Times: 4	Modifica																									
Delete	Times: 4																												
Modifica																													
Aggiungi caso																													

Figura 7 Layout admin

Layout operatori:

Il layout è uguale a quello dell'admin, ma alcune funzioni non sono presenti.

Casi																					
Filtro casi																					
<table border="1"><tr><td>Caso:</td><td>titolo</td><td>Created</td><td>xx.xx.xxxx</td></tr><tr><td>Categoria:</td><td>categoria</td><td></td><td></td></tr><tr><td>Variante di</td><td>Case_id</td><td></td><td></td></tr><tr><td colspan="4">Descrizione</td></tr><tr><td colspan="4">Times: 4</td></tr><tr><td>Aggiungi caso</td></tr></table>	Caso:	titolo	Created	xx.xx.xxxx	Categoria:	categoria			Variante di	Case_id			Descrizione				Times: 4				Aggiungi caso
Caso:	titolo	Created	xx.xx.xxxx																		
Categoria:	categoria																				
Variante di	Case_id																				
Descrizione																					
Times: 4																					
Aggiungi caso																					

Figura 8 Layout operatori

2.4 Design procedurale

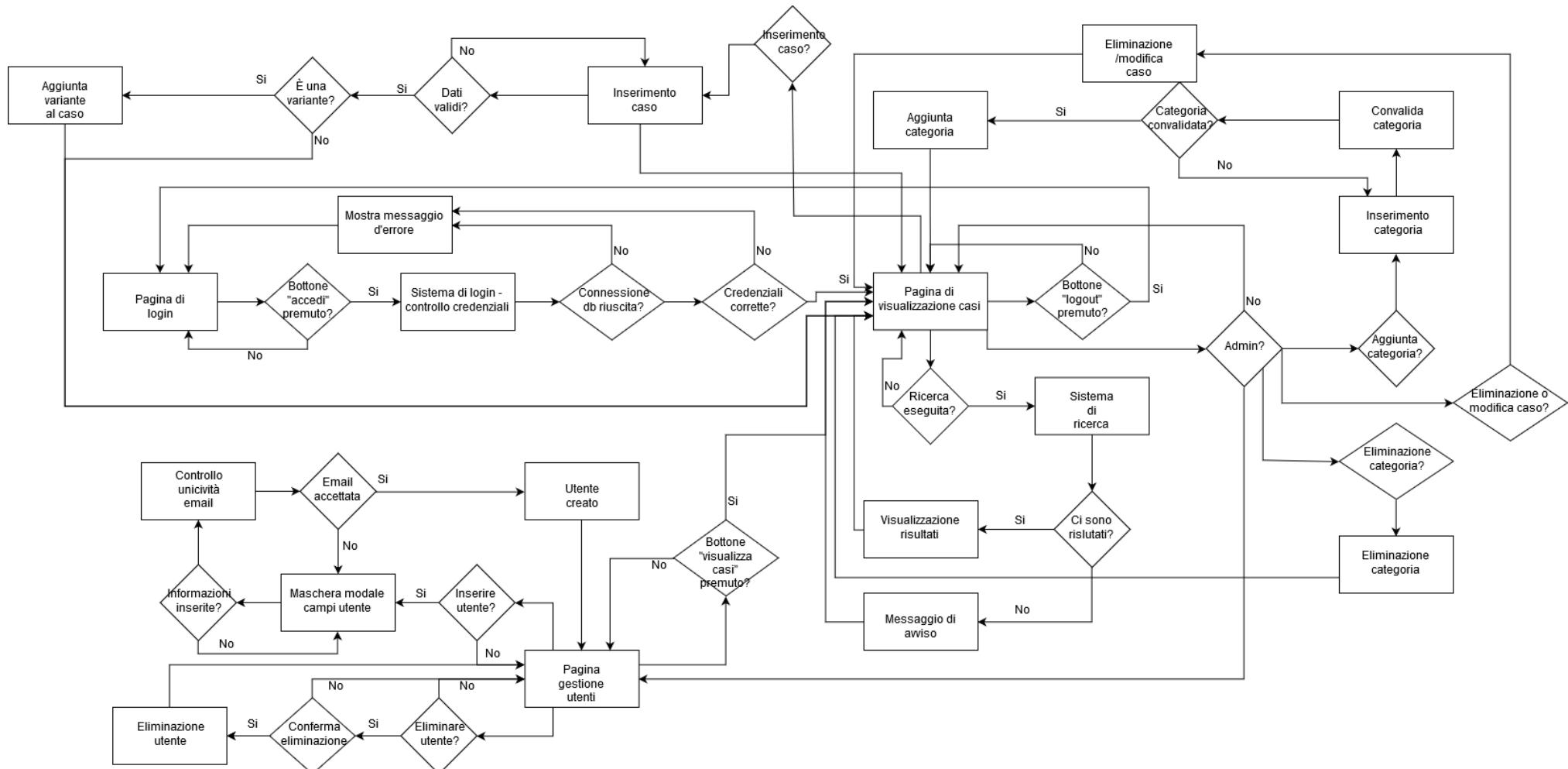


Figura 9 Diagramma procedurale

3 Implementazione

3.1 Database

Ho progettato il database ragionando a tutti gli scenari possibili e presentabili durante la navigazione all'interno dell'applicativo.

Durante l'implementazione mi sono accorto che alcune cose pianificate durante la parte di progettazione non erano ottimizzate, quindi ho dovuto effettuare delle modifiche alla struttura della banca dati.

Le modifiche che ho effettuato erano sulle colonne di alcune tabelle in cui mancavano delle condizioni, ad esempio l'attributo password non può essere nullo.

```
ALTER TABLE users ADD COLUMN password varchar(255) NOT NULL;
```

Inoltre ho aggiunto in un secondo momento che quando avviene un update di un record, in automatico viene impostato il timestamp della modifica.

```
ALTER TABLE cases CHANGE `modified_at` `modified_at` TIMESTAMP  
NULL ON UPDATE CURRENT_TIMESTAMP;
```

3.2 Framework mvc

Per questo progetto ho utilizzato il framework mvc (il template creato dal docente Sartori). La struttura delle cartelle corrisponde alla seguente:

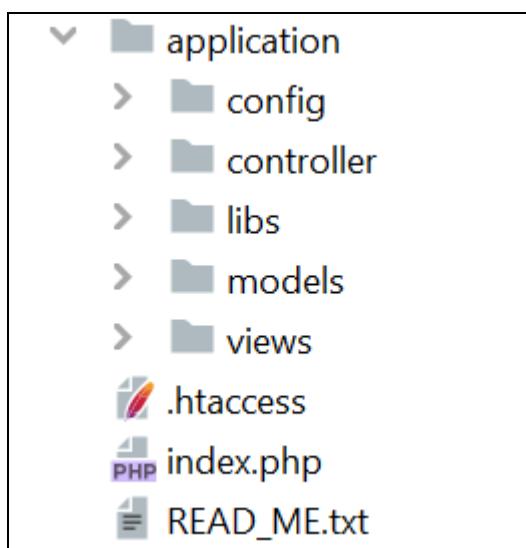


Figure 10 Struttura del progetto

Come vediamo la cartella principale application include i controller, le librerie, models, views ed alcuni file di configurazione (.htaccess e cartella config).

I controller sono delle classi che si occupano di effettuare delle operazioni (ad esempio prendere i dati da un determinato model) e richiamare delle views.

Le classi models sono delle classi che lavorano con i dati delle varie tabelle oppure che possiedono metodi per la validazione di input.

Nella cartella views invece sono presenti i layout (praticamente tutto html se non qualche parte di codice in php per rendere le pagine dinamiche).

3.2.1 Come funziona

Il funzionamento è molto semplice, nei controller bisogna inserire dei metodi pubblici che servono ad effettuare delle operazioni (ad esempio mostrare la pagina di login). Andando quindi nel browser bisogna inserire tutto il percorso del progetto fino ad arrivare al livello precedente della cartella **application**.

Esempio in questo progetto: http://localhost:8080/knowledge_base

Successivamente inserire il carattere "/" e digitare il nome del controller che si desidera utilizzare ed infine, dopo aver inserito sempre il carattere "/", inserire il nome del metodo che si vuole richiamare.

Esempio: http://localhost:8080/knowledge_base/home/index

In questo modo verrà mostrata il layout della pagina che viene chiamata all'interno del metodo **index()** del controller Home.php. Se non viene inserito il nome del metodo oppure né del metodo né del controller vengono utilizzati dei parametri di default configurati nel file application.php presente nella cartella libs.

Viene chiamato il controller di default anche nel caso che venga digitato un controller non esistente (stessa cosa vale per i metodi).

```
if (file_exists('./application/controller/' . $this->url_controller . '.php')) {  
    //controllo se il metodo esiste  
    if (method_exists($this->url_controller, $this->url_action)) {  
        } else {  
            //chiamo il metodo di default  
            $this->url_controller->index();  
        }  
    } else {  
        //controller di default e metodo di default  
        require './application/controller/home.php';  
        $home = new Home();  
        $home->index();  
    }  
}
```

3.2.2 Configurazione

Per configurare in modo corretto il framework bisogna modificare il percorso nel file .htaccess a partire dalla root del webserver al livello prima della cartella application.

```
# If your app is in the root of your web folder, then leave it commented out
RewriteBase /knowledge base
```

Infine configurare il percorso nel file config.php presente nella cartella config. Bisogna impostare il percorso inserendo l'indirizzo del server con la porta del webserver ed il percorso del progetto a partire dalla cartella root del webserver:

```
/**
 * Configurazione di : URL del progetto
 */
define('URL', 'http://localhost:8080/knowledge base/');
```

3.2.3 Riflessione personale

Personalmente questo framework mi ha facilitato il lavoro durante lo sviluppo permettendo la separazione modulare tra i controller e le classi dei models, oltre alla divisione dalle views. Mi ha reso il lavoro più semplice e la parte di astrazione modulare più intuitiva. Sono quindi soddisfatto di aver utilizzato questo framework per sviluppare l'applicativo web.

Controllers

3.2.4 Home

Nel controller home.php ci sono pochi metodi, la maggior parte di essi sono statici e vengono utilizzati dagli altri controller.

Questi sono i metodi che la classe Home contiene:

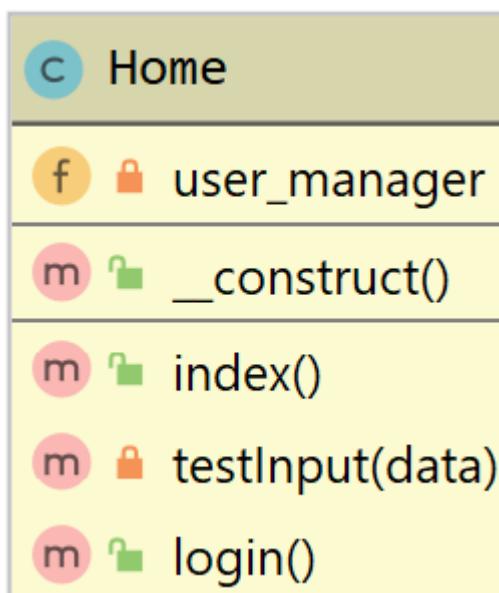


Figure 11 Classe Home

L'attributo privato **\$user_manager** è un riferimento alla classe UserManager che contiene i metodi utili per gli utenti, in questo caso serve per il login.

Questo attributo viene inizializzato nel costruttore del controller in un blocco **try catch**. Questo perché nel costruttore della classe UserManager viene chiamato il metodo statico di connessione al database. Se questo fallisce allora l'attributo **\$user_manager** sarà **null**.

In ogni metodo viene controllato se l'attributo è **null** o meno da chiamare la pagina di errore di connessione al database se la connessione fallisse.

3.2.4.1 Metodo Index

Il metodo index è un semplice metodo che mostra la pagina di login al client e contiene il seguente codice.

```
if (isset($this->user_manager)) {  
    require_once 'application/views/templates/head.php';  
    require_once 'application/views/login/index.php';  
} else {  
    //redirect to no connection page  
    DbError::noDatabaseConnection();  
}
```

3.2.5 DbError

Il controller dbError contiene un unico metodo statico **noDatabaseConnectio()** che viene richiamato dagli altri controller quando la connessione al database fallisce.

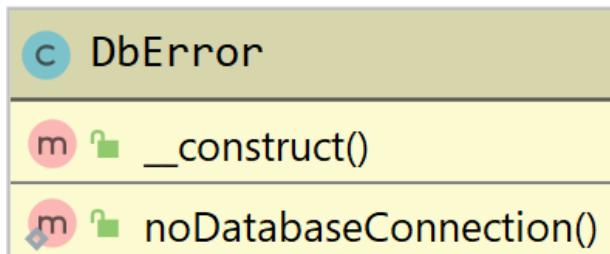


Figure 12 Classe BdError

Questo metodo non fa altro che richiamare il metodo statico **logout()** della classe UserManager.php in modo che l'utente connesso al momento del fallimento di connessione venga scollegato.

Mostra inoltre all'utente la pagina di errore di connessione al database.

```
public static function noDatabaseConnection()
{
    UserManager::logout();
    require_once 'application/views/templates/head.php';
    require_once 'application/views/errors/database_error.php';
}
```

3.2.6 Users

Il controller users.php contiene metodi utili per la gestione degli utenti. Il controller contiene i seguenti metodi.

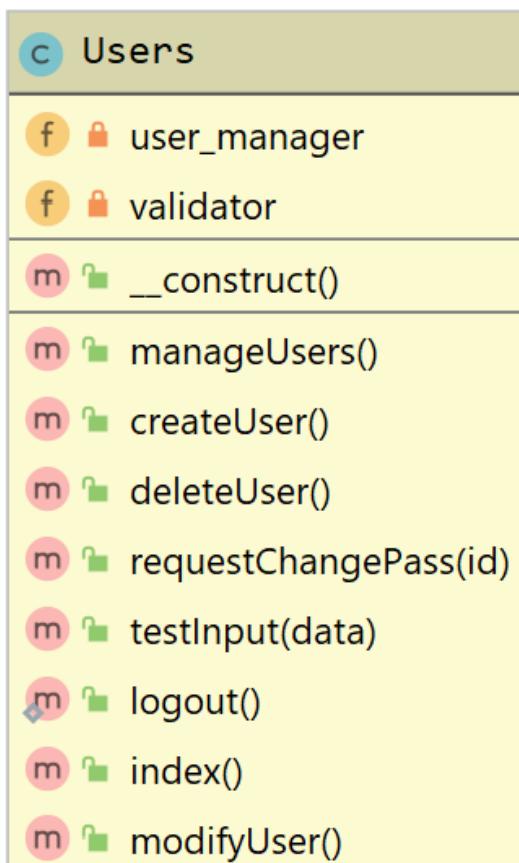


Figure 13 classe Users

Questa classe contiene un'attributo **\$user_manager** che è un riferimento alla classe in cui ci sono i metodi che effettuano operazione sui dati degli utenti. **\$validator** invece è un riferimento alla classe Validator.php.

3.2.6.1 Metodo manageUsers

Il metodo **manageUsers** serve a mostrare la pagina di gestione degli utenti. In questo metodo salvati i dati degli utenti (**\$user_manager->getUsersList()**).

```
//get users list  
$users = $this->user_manager->getUsersList();
```

3.2.6.2 Metodo deleteUser

Il metodo **deleteUser** prova ad eliminare l'utente che possiede come id, il numero identificativo passato nella variabile **\$_POST['userToDelete']**. Viene utilizzato il metodo **deleteUser** della classe UserManager.php.

```
$id = intval($this->testInput($_POST['userToDeleteId']));  
  
//try to delete user  
$this->user_manager->deleteUser($id)
```

3.2.6.3 Metodo logout

Il metodo logout non fa altro che richiamare il metodo statico **logout()** della classe UserManager.php e reindirizzare l'utente alla pagina di login.

3.2.6.4 Metodo createUser

In questo metodo vengono controllati i campi inseriti dall'utente al momento che vuole creare un utente. I campi vengono controllati tramite i metodi della classe validator.

Se i campi sono validi, utilizzando il metodo **createUser(\$user)** della classe UserManager.php, viene effettuato un tentativo di creazione dell'utente (non sempre va a buon fine).

```
//try to create user  
$this->user_manager->createUser($user)
```

La password viene criptata con l'algoritmo **bcrypt**. Questo viene fatto con il metodo **password_hash(\$password, PASSWORD_DEFAULT)** in cui viene passato come primo parametro la password da criptare e secondariamente l'algoritmo da utilizzare.

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

3.2.6.5 Metodi requestChangePass e testInput

Il metodo **requestChangePass()** è implementato nel codice ma la funzione non è abilitata nell'applicativo. Questo metodo richiama un metodo sempre della classe di gestione degli utenti che cambia il valore dell'attributo `change_pass` del database (vedi implementazioni future).

Il metodo `testInput` corrisponde esattamente al metodo del controller `home.php`.

3.2.6.6 Metodo index

Questo metodo non fa altro che chiamare il metodo **manageUsers()**. Questo metodo viene richiamato se l'utente digita nell'url un metodo che non è presente nel controller.

3.2.6.7 Metodo modifyUser

Questo metodo viene invocato quando si tenta di modificare un utente esistente (dalla pagina di gestione utenti alla quale solo gli admin hanno accesso).

Questo metodo corrisponde al metodo **createUser()** con la particolarità che controlla che la password sia vuota o meno:

- Password vuota → utente non vuole modificare la password corrente
- Password inserita → vengono effettuati i controlli di complessità della password

Se la password è settata viene effettuato l'hash sempre con l'algoritmo bcript.

```
//set empty password
$password = "";

//check if there is a password
if (isset($_POST['modified_password'])) && !empty($_POST['modified_password'])) {

    $password = $this->testInput($_POST['modified_password']);

    //check if the password is complex enough
    if (!PasswordManager::checkStrength($password)) {
        MessageManager::setErrorMsg("La password non rispetta le condizioni di sicurezza");
        MessageManager::printErrorMsg();
        $this->manageUsers();
        exit();
    }

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

Come avviene nel metodo di creazione viene inizializzato un oggetto User a cui si passano i parametri dell'utente modificato che viene passato al metodo **modifyUser()** della classe UserManager.php.

```
//try to update user
if ($this->user_manager->modifyUser($user, $id)) {
```

3.2.7 ResearchCases

In questo controller vi sono tutti i metodi utili per svolgere le funzioni implementate nella pagina ricerca_casi.php. Questo controller contiene i seguenti metodi.

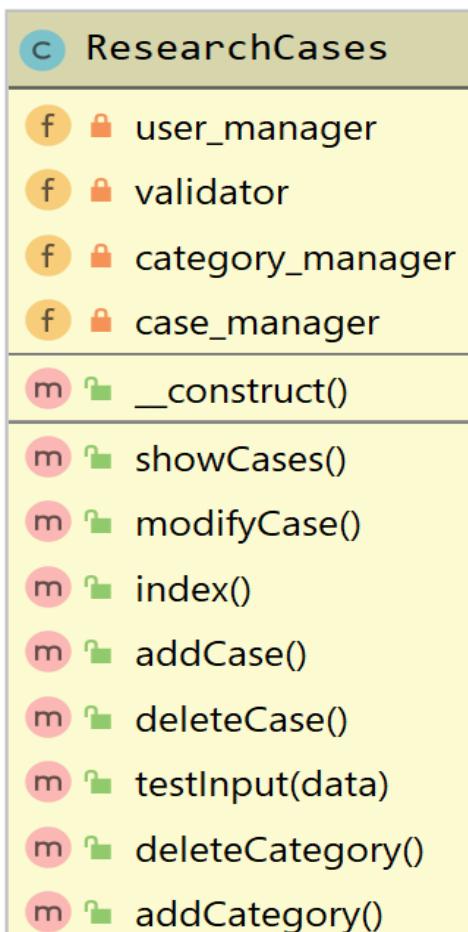


Figure 14 Classe ResearchCases

Questo controller possiede degli attributi che sono dei riferimenti alle classi che servono per la gestione dei casi, utenti, categorie e validator.

3.2.7.1 Metodo showCases

Questo metodo viene richiamato quando l'utente vuole visualizzare la pagina ricerca_casi.php. Questo metodo controlla se le variabili per i filtri sono applicati e successivamente, se i filtri sono attivi, crea e salva all'interno delle variabili di sessione i vari filtri. Una volta fatto ciò richiama il metodo che ritorna i risultati della query effettuata nel metodo **getCases()** della classe CasesManager.php.

```
//get cases
$cases = $this->case_manager->getCases();
```

In base ai valori inseriti nei campi di ricerca vengono settate delle variabili di sessione che verranno utilizzate successivamente nella classe CaseManager.php. La variabile `$_POST["order_results"]` contiene un valore intero che corrisponde al tipo di ordinamento:

- Se il valore è 0 → ordinamento dal caso più recente
- Se il valore è 1 → ordinamento dal caso con più rappresentazioni
- Se il valore è 2 → ordinamento dal caso meno recente

```
if (isset($_POST['order_results']) && intval($_POST['order_results']) == 1) {

    //order by most represent
    $_SESSION['order_results'] = 1;
} else if (isset($_POST['order_results']) && intval($_POST['order_results']) == 2) {

    //order by less recent
    $_SESSION['order_results'] = 2;
} else {
    //order by most recent
    $_SESSION['order_results'] = 0;
}
```

Successivamente, utilizzando il metodo **getAllCases**, salvo in una variabile tutti i casi (quindi senza filtri) per stampare le opzioni all'interno del menu dropdown durante la modifica di un caso.

```
$all_cases = $this->case_manager->getAllCases();
```

Infine, l'utente viene reindirizzato l'utente alla pagina ricerca_casi.php.

3.2.7.2 Metodi deleteCategory e deleteCase

I due metodi vengono chiamati quando un utente con privilegi da admin, per eliminare una categoria o rispettivamente un caso.

Nel metodo **deleteCategory()** viene chiamato il metodo **deleteCategory(\$id)** della classe CategoryManager.php.

```
$category_id = $this->testInput($_POST['delete_category_id']);

//try to delete category
$this->category_manager->deleteCategory($category_id);
```

Nel metodo **deleteCase** viene svolto lo stesso procedimento per l'eliminazione di un caso.

```
$id = intval($this->testInput($_POST['caseToDeleteId']));  
  
//try to delete the case  
$this->case_manager->setDeletedCase($id);
```

3.2.7.3 Metodo index

Questo metodo non fa altro che chiamare il metodo **showCases()**. Questo metodo viene richiamato se l'utente digita nell'url un metodo che non è presente nel controller.

3.2.7.4 Metodi addCase e addCategory

Il procedimento è molto simile a quello per l'aggiunta di un nuovo utente. In entrambi i metodi vengono controllati che le rispettive variabili **\$_POST** siano settate e tramite la classe Validator.php che i loro valori rispettino la sintassi corretta.

Aggiunta categoria:

```
$new_category = $this->testInput($_POST['new_category']);  
  
$this->validator->validateTextInput($new_category, 1, 50);  
  
//add new category  
if ($this->category_manager->addCategory($new_category))
```

Stesso procedimento per l'aggiunta di un caso, ogni campo viene validato.

```
//create DocCase object  
$case = new DocCase($title, $category, $variant, $description);  
  
$this->case_manager->addCase($case);
```

3.2.7.5 Metodo modifyCase

Questo metodo viene chiamato quando l'utente decide di salvare delle modifiche apportate ad un caso. Una volta effettuati i controlli sulla sintassi dei valori inseriti, utilizza un metodo della classe CaseManager.php per modificare il caso richiesto.

```
//create DocCase object  
$case = new DocCase($title, $category, $variant, $description);  
  
$this->case_manager->modifyCase($case, $id);
```

3.3 Models

All'interno della cartella models ci sono tutte le classi che interagiscono con i dati del database (effettuano query), classi che modellizzano gli oggetti (caso, utente), classi per la validazione dati, per la connessione alla banca dati.

I metodi di eliminazione, modifica e aggiunta dati **ritornano un valore booleano** per sapere se l'operazione è andata a buon fine o meno.

3.3.1 DbManager

Questa classe possiede un unico metodo statico che serve ad effettuare la connessione al database.

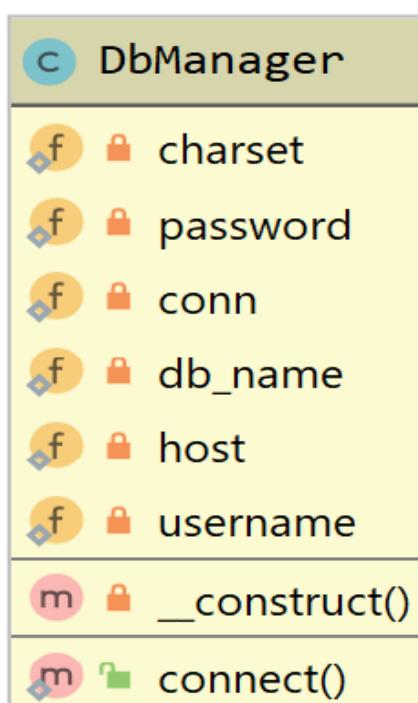


Figure 15 Classe DbManager

Questo metodo viene utilizzato da tutte le classi, controller che devono connettersi o controllare la connessione al database.

```
public static function connect()
{
    if (!self::$conn) {

        self::$conn = new PDO("mysql:host=" . self::$host . ";dbname=" .
        self::$db_name . ";charset=" . self::$charset, self::$username,
        self::$password);

        // set the PDO error mode to exception
        self::$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return self::$conn;
    }
    return self::$conn;
}
```

Nelle altre classi per controllare che la connessione sia riuscita creo una variabile e le assegno il valore ritornato dal metodo **connect()**. Successivamente nei vari metodi controllo che il suo valore non sia **null**. Se la variabile risulta **null** significa che la connessione è fallita.

```
private $conn;

public function construct()
{
    try {
        $this->conn = DbManager::connect();
    } catch (PDOException $ex) {
        throw $ex;
    }
}
```

3.3.2 Validator

La classe Validator.php contiene un metodo **validateTextInput()** che controlla che il parametro di testo passato al metodo contenga un numero di caratteri che sia compreso tra il numero minimo e massimo desiderato.

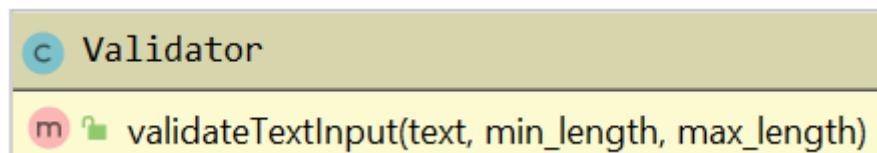


Figure 16 Classe Validator

Questo metodo controlla quindi che il testo non contenga più o meno caratteri del consentito.

```
public function validateTextInput($text, $min_length, $max_length) {
    return (strlen($text) >= $min_length && strlen($text) <= $max_length);
}
```

La variabile **\$text** corrisponde al testo da controllare, **\$min** il numero minimo di caratteri e **\$max** il numero Massimo di caratteri consentiti.

3.3.3 PasswordManager

La classe password manager contiene due metodi statici utili per la gestione delle password.

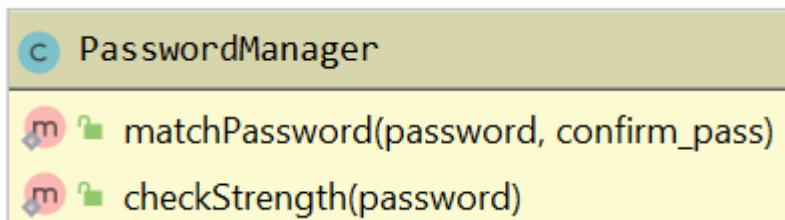


Figure 17 Classe PasswordManager

Entrambi i metodi vengono utilizzati all'interno del controller users.php nel metodo **createUser()**.

3.3.3.1 Metodo matchPassword

Questo metodo confronta che la password inserita dall'utente corrisponda alla password di conferma, in modo da evitare che l'utente sbagli ad inserire la prima password.

Bisogna passare come parametri la password e la password di conferma.

```
public static function matchPassword($password, $confirm_pass)
{
    return $password === $confirm_pass;
}
```

3.3.3.2 Metodo checkStrength

Nel metodo **checkStrength(\$password)**, tramite le regular expression, viene controllato che la password inserita dall'utente rispetti una sintassi di sicurezza minima:

- Almeno 8 caratteri
- Un carattere maiuscolo
- Massimo 50 caratteri
- Almeno un numero

```
public static function checkStrength($password)
{
    //length 8, 1 uppercase, at least 1 digit, max length 50 characters
    $pattern = '/^(?=.*[0-9])(?=.*[A-Z]).{8,50}$/';

    return preg_match($pattern, $password);
}
```

Si deve passare come parametro la password da controllare.

3.3.4 DocCase

Questa classe modellizza l'oggetto “caso”. Questa classe contiene i metodi **getter** dei vari attributi che vengono settati nel costruttore.

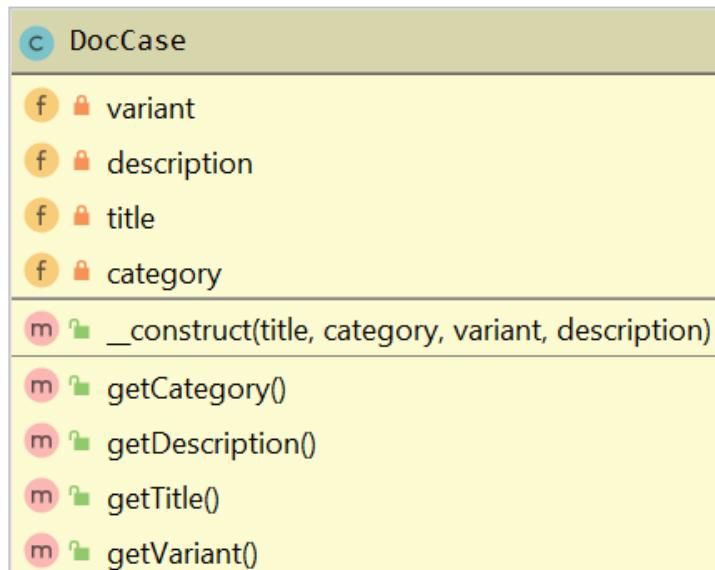


Figure 18 Classe DocCase

Questa classe la utilizzo all'interno del metodo **addCase()** del controller `researchCases.php`, in cui passo alla classe `CaseManager.php` l'oggetto **CaseDoc** per evitare di passare tutti i parametri a mano e non farli gestire a quest'ultima.

3.3.4.1 Metodo __construct

Questo metodo richiede come parametri il titolo del caso, l'id della categoria alla quale appartiene, l'id della variante (**null** se non è una variante di nessun caso) e la descrizione del caso.

```
public function __construct($title, $category, $variant, $description)
{
    $this->title = $title;
    $this->category = $category;
    $this->variant = $variant;
    $this->description = $description;
}
```

3.3.4.2 Metodi getter

Questi metodi non fanno altro che ritornare il valore dei rispettivi attributi. Qui sotto è illustrato un esempio di metodo getter: **getTitle()**

```
public function getTitle()
{
    return $this->title;
}
```

3.3.5 User

Questa classe ha la stessa funzione della classe DocCase.php ma l'oggetto modellizzato in questo caso è “l'utente”. Anche in questo caso sono presenti solo i metodi **getter** ed il costruttore.

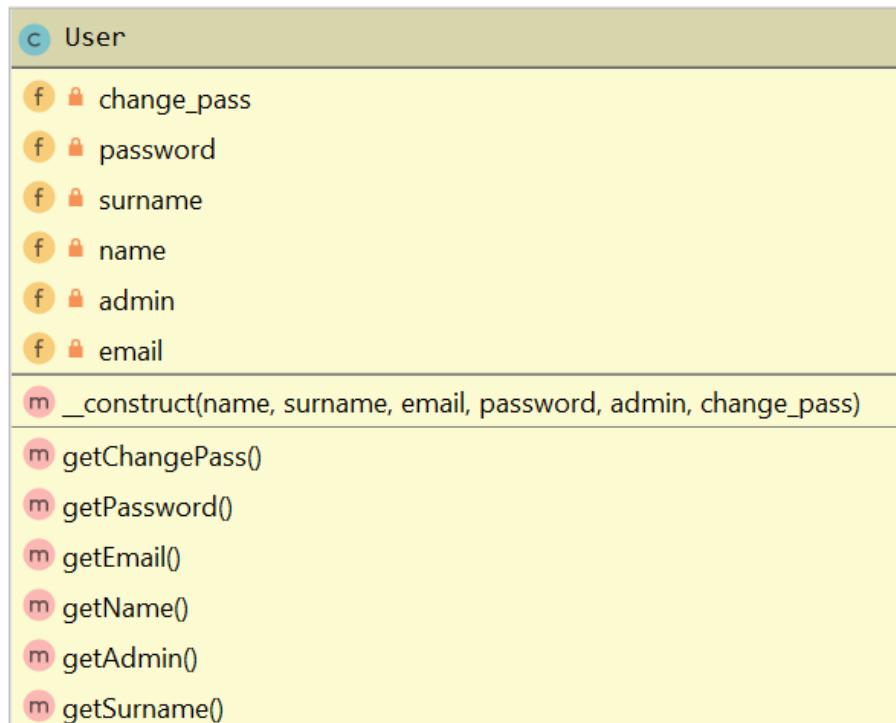


Figure 19 Classe User

Questa classe viene utilizzata dal controller users.php nel metodo di creazione dell'utente (`createUser($user)`).

3.3.5.1 Metodo __construct

Il metodo `__construct()` non fa altro che assegnare i valori ricevuti come parametro ai relativi attributi. Richiede il nome dell'utente, il cognome, la sua email, la password, se è admin o meno (1 corrisponde ad utente admin) e se è stato richiesto un cambio di password (implementazione futura, valore di default 0).

```

public function __construct(string $name, string $surname, string $email, string
$password, $admin, $change_pass )
{
    $this->name = $name;
    $this->surname = $surname;
    $this->email = $email;
    $this->password = $password;
    $this->admin = $admin;
    $this->change_pass = $change_pass;
}

```

3.3.6 CategoryManager

La classe CategoryManager.php è una classe che serve a lavorare con i dati presenti nel database relativi alla categoria. La classe possiede i seguenti metodi:

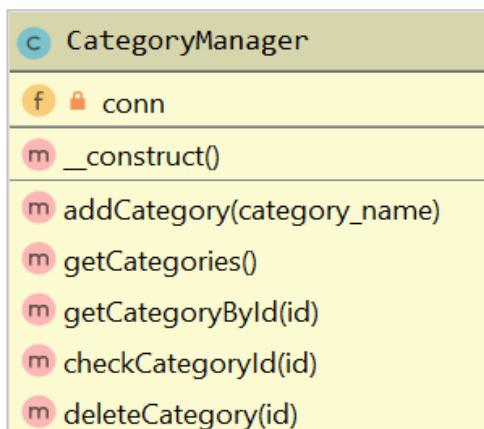


Figure 20 Classe CategoryManager

3.3.6.1 Metodi getCategories e getCategoryById

Questi due metodi sono utilizzati nel controller researchCase.php, svolgono una funzione molte simile con una differenza.

Il metodo **getCategories()** ritorna tutte le categorie presenti all'interno del sistema.

```

public function getCategories() {
    try {
        //prepare query
        $prepared_query = $this->conn->prepare("SELECT * FROM categories");
        $prepared_query->execute();
        return $prepared_query->fetchAll();

    } catch (PDOException $ex) {
    }
}

```

Il metodo **getCategoryById(\$id)** ritorna la categoria desiderato passando come parametro l'id della categoria.

```

public function getCategoryById($id) {
    try {
        //prepare query
        $prepared_query = $this->conn->prepare("SELECT name FROM categories
WHERE id = :id");
        $prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
        $prepared_query->execute();

        $res = $prepared_query->fetch();
        return $res[0];

    } catch (PDOException $ex) {
    }
}

```

3.3.6.2 Metodi addCategory e deleteCategory

Il metodo **addCategory(\$category_name)** è un metodo che viene utilizzato quando l'utente prova a creare una nuova categoria e richiede il nome della categoria come parametro.

In questo caso non ho creato la classe Category.php perché la categoria richiede solo il nome.

```
public function addCategory($category_name) {
    try {
        //prepare query
        $prepared_query = $this->conn->prepare("INSERT INTO categories(name)
VALUES (:category_name)");
        $prepared_query->bindParam(':category_name', $category_name,
PDO::PARAM_STR);
        $prepared_query->execute();

        return true;
    } catch (PDOException $ex) {
        Home::setErrorMsg("La categoria esiste già");
        return false;
    }
}
```

Il metodo **deleteCategory(\$id)** serve ad eliminare una categoria dal database. Come parametro viene richiesto l'id della categoria.

```
public function deleteCategory($id) {
    try {
        //prepare query
        $prepared_query = $this->conn->prepare("DELETE FROM categories WHERE id
= :id");
        $prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
        $prepared_query->execute();

        return true;
    } catch (PDOException $ex) {
        echo $ex;
        Home::setErrorMsg("Impossibile eliminare questa categoria");
        return false;
    }
}
```

3.3.6.3 Metodo checkCategory

Questo metodo controlla e ritorna se esiste una categoria che possiede come id l'identificatore passato come parametro.

```
public function checkCategoryId($id)
{
    try {
        $prepared_query = $this->conn->prepare("SELECT * FROM CATEGORIES WHERE
id = :id");
        $prepared_query->bindParam(":id", $id, PDO::PARAM_INT);
        $prepared_query->execute();
        $res = $prepared_query->fetchAll(PDO::FETCH_ASSOC);

        return sizeof($res) > 0;
    } catch (PDOException $ex) {
    }
}
```

Questo metodo serve per evitare che l'utente inserisca una categoria (id della categoria), durante la creazione o modifica di un caso, che non è presente nella banca dati.

3.3.7 MessageManager

Questa classe contiene dei metodi che vengono utilizzati per mostrare all'utente dei messaggi di errore o di successo.

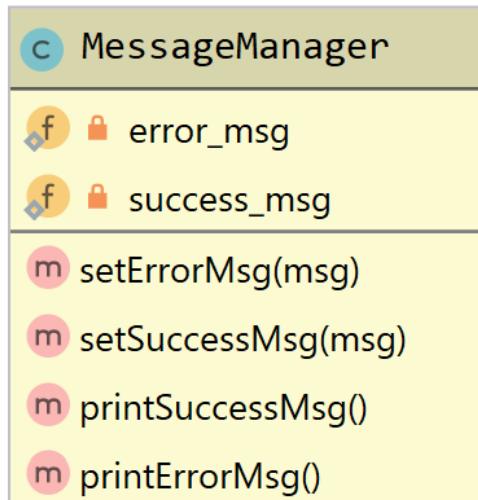


Figure 21 Classe MessageManager

3.3.7.1 Metodi setErrorMsg e setSuccessMsg

Questi due metodi contengono lo stesso codice con l'unica differenza dell'azione che svolgono. Il metodo `setErrorMsg($msg)` imposta il testo passato come parametro alla variabile statica `$error_msg` mentre `setSuccessMsg($msg)` alla variabile statica `$success_msg`.

3.3.7.2 Metodi printErrorMsg e printSuccessMsg

Questi due metodi non fanno altro che stampare, tramite degli alert, creati utilizzando mdbootstrap, le stringhe salvate all'interno delle variabili `$error_msg` e `$success_msg`.



Errore! Email già utilizzata

Figure 22 Messaggio di errore



Ottimo! L'utente è stato creato con successo

Figure 23 Messaggio di successo

3.3.8 CaseManager

Questa classe viene utilizzata per effettuare tutte le operazioni sui dati relativi ai casi di documentazione tecnica. La classe presenta i seguenti metodi.

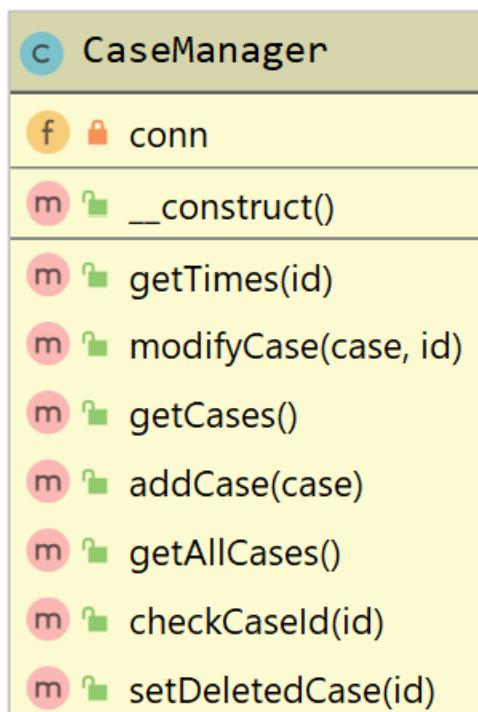


Figure 24 Classe CaseManager

3.3.8.1 Metodo getCases()

Questo metodo ritorna i casi applicando i filtri. Nel controller researchCases.php nel metodo **showCases()** vengono settate le variabili di sessione utilizzati per i filtri.

```
//set session variables value
(isset($_SESSION['text_filter'])) ? $text_filter = $_SESSION['text_filter'] :
$text_filter = "";
(isset($_SESSION['date_filter'])) ? $date_filter = $_SESSION['date_filter'] :
$date_filter = "";

//if value == 0 -> all categories
(isset($_SESSION['category_filter']) && intval($_SESSION['category_filter']) != 0) ? $category_filter = $_SESSION['category_filter'] : $category_filter = "";
```

Se le variabili di sessione non sono state inizializzate, la query che viene eseguita successivamente non applica nessun filtro di ricerca perché le variabili filtri contengono il valore “%%” che ritorna tutti i risultati. Altrimenti se contiene ad esempio del testo viene effettuata la query in cui si cercano tutti i casi che contengano quel testo all'interno del **titolo** o della **descrizione**.

```
$id = $text_filter;
$text_filter = "%" . $text_filter . "%";
$date_filter = "%" . $date_filter . "%";
$category_filter = "%" . $category_filter . "%";

if (intval($_SESSION['order_results']) == 0 ||
    intval($_SESSION['order_results']) == 2) {
```

Sul campo **id** e **variante** viene fatta la query precisa, quindi il numero deve corrispondere esattamente a quello richiesto al contrario del **titolo** e la **descrizione** che devono contenere il numero. Questa è la query per quanto riguarda l'ordinamento per data (dal più recente o viceversa).

```
//order by date
$sql = "SELECT * FROM cases WHERE DELETED = 0 AND
      (description LIKE :text_filter
       OR id LIKE :id
       OR title LIKE :text_filter
       OR variant LIKE :id)
      AND (created_at LIKE :date_filter)
      AND (category_id LIKE :category_id" .
      $include_null_category_id .
      "order by created at " . $order;
```

Se il tipo di ordinamento corrisponde a quello dal caso più ricorrente a quello meno ricorrente viene eseguita una query leggermente più complessa che effettua una join sulla stessa tabella in modo da ritornare i casi che sono stati ripresentati più volte in ordine ascendente.

```
$sql = "SELECT c.* FROM cases c
        INNER JOIN cases a ON
        c.id = a.variant
        where a.deleted = 0 AND a.variant IS NOT NULL
        AND (c.description LIKE :text_filter
             OR c.id LIKE :id
             OR c.title LIKE :text_filter
             OR c.variant LIKE :id)
        AND (c.created_at LIKE :date_filter)
        AND (c.category_id LIKE :category_id" . $include_null_category_id .
" group by a.variant
        order by count(a.variant) desc;" ;
```

Successivamente, indipendentemente dalla query, viene chiamato il metodo della classe PDO **prepare()** per evitare sql injection. Infine viene ritornato il risultato della query al relativo controller.

```
$prepared_query = $this->conn->prepare($sql);

//bind params
$prepared_query->bindParam(':text_filter', $text_filter, PDO::PARAM_STR);
$prepared_query->bindParam(':id', $id, PDO::PARAM_STR);
$prepared_query->bindParam(':date_filter', $date_filter, PDO::PARAM_STR);
$prepared_query->bindParam(':category_id', $category_filter, PDO::PARAM_STR);

$prepared_query->execute();

return $prepared_query->fetchAll(PDO::FETCH_ASSOC);
```

3.3.8.2 Metodo getAllCases

Questo metodo ritorna tutti i casi senza applicare filtri.

```
$prepared_query = $this->conn->prepare("SELECT * FROM CASES WHERE deleted = 0");
$prepared_query->execute();
$res = $prepared_query->fetchAll(PDO::FETCH_ASSOC);

return $res;
```

3.3.8.3 Metodo checkCaseId

Questo metodo ritorna un valore booleano che indica se esiste o meno un caso che possiede come id il valore passato come parametro del metodo.

```
$prepared_query = $this->conn->prepare("SELECT * FROM CASES WHERE deleted = 0
AND ID = :id");
$prepared_query->bindParam(":id", $id, PDO::PARAM_INT);
$prepared_query->execute();
$res = $prepared_query->fetchAll(PDO::FETCH_ASSOC);

return sizeof($res) > 0;
```

3.3.8.4 Metodo addCase e setDeleteCase

Il metodo **addCase(\$case)** richiede come parametro un oggetto di tipo DocCase relativo al nuovo caso. Questo metodo prova ad inserire nella tabella cases il nuovo record. Per ottenere le informazioni vengono utilizzati i metodi della classe DocCase.php.

```
//get values
$title = $case->getTitle();
getDescription = $case->getDescription();
$category_id = $case->getCategory();
$variant = $case->getVariant();

//prepare query
$prepared_query = $this->conn->prepare($sql);
```

Il metodo **setDeletedCase(\$id)** serve a settare l'attributo **deleted** della tabella **cases** del caso con id passato come parametro ad 1. In questo modo il caso non viene realmente eliminato, ma non viene più mostrato agli utenti.

Prima di tutto bisogna settare a **null** i campi **variant** della tabella dei casi che possiedono come variante il caso che si desidera eliminare.

```
//get cases that have the variant id of the deleted case
$variant_query = $this->conn->prepare("SELECT * FROM CASES WHERE variant =
:id");
$variant_query->bindParam(':id', $id, PDO::PARAM_INT);
$variant_query->execute();

$results = $variant_query->fetchAll(PDO::FETCH_ASSOC);

//set null variant field
foreach ($results as $result) {
    $prepared_query = $this->conn->prepare("UPDATE CASES set variant = null
where id = :id");
    $prepared_query->bindParam(':id', $result['id'], PDO::PARAM_INT);
    $prepared_query->execute();
}
```

Successivamente si setta a 1 il campo **deleted** del caso passato come parametro.

```
//prepare query
$prepared_query = $this->conn->prepare("UPDATE CASES SET deleted = 1 WHERE ID =
:id");
$prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
$prepared_query->execute();
//prepare query
$prepared_query = $this->conn->prepare("UPDATE CASES SET deleted = 1 WHERE ID =
:id");
$prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
$prepared_query->execute();
```

3.3.8.5 Metodo getTimes

Questo metodo ritorna il numero di volte che il caso che possiede l'id passato come parametro del metodo è stato ripresentato come variante. Questo metodo è stato ideato per evitare di fare ogni volta una join della tabella per ottenere il risultato, così da migliorare le prestazioni.

```
//get times
$prepared_query = $this->conn->prepare("SELECT count(*) FROM cases WHERE variant =
:id and deleted = 0");
$prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
$prepared_query->execute();

$res = $prepared_query->fetch();

return intval($res[0]);
```

3.3.8.6 Metodo modifyCase

Questo metodo richiede come parametro l'id del caso che si desidera modificare. Mentre il parametro **\$case** che è l'oggetto DocCase che contiene il caso modificato. Come nel metodo **addCase()**, utilizzando i metodi della classe DocCase-php si ottiene il titolo del caso, la descrizione, variante e categoria.

Una volta fatto ciò bisogna aggiornare il caso esistente con i nuovi dati.

```
$sql = "UPDATE cases SET title = :title, description = :description, category_id = :category_id, variant = :variant WHERE ID = :id;";

//get values
$title = $case->getTitle();
$description = $case->getDescription();
$category_id = $case->getCategory();
$variant = $case->getVariant();

...

//prepare query
$prepared_query = $this->conn->prepare($sql);

...

$prepared_query->execute();
```

3.3.9 UserManager

La classe UserManager.php ha la stessa funzione della classe CaseManager.php ed include alcuni metodi aggiuntivi. La classe possiede la variabile di connessione al database, presente in tutte le classi che lavorano utilizzando la banca dati **\$conn** ed i seguenti metodi.

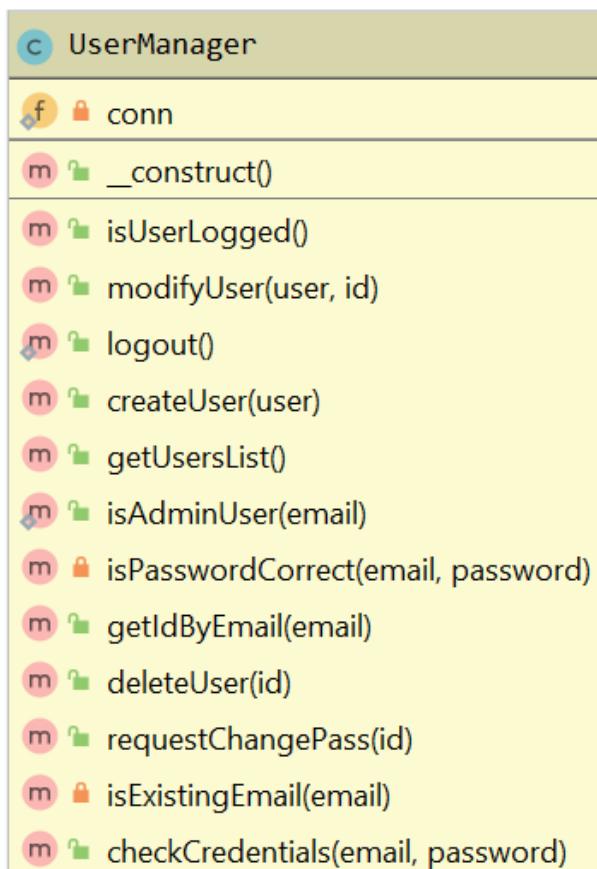


Figure 24 Classe UserManager

3.3.9.1 Metodo getUsersList

Questo metodo non fa altro che ritornare la lista di tutti gli utenti. Viene utilizzato nel controller users.php per stampare successivamente gli utenti in una lista nella pagina di gestione utenti.

```
//get the password from database
$prepared_query = self::$conn->prepare("SELECT * FROM users");
$prepared_query->execute();

$users = $prepared_query->fetchAll();

return $users;
```

3.3.9.2 Metodo checkCredentials

Questo metodo controlla se, utilizzando il metodo privato **isExistingEmail(\$email)** che ritorna se l'email passata come parametro esiste all'interno del database e successivamente, se l'email esiste, il metodo **isPasswordCorrect(\$email, \$password)**, le credenziali utilizzate durante il login sono corrette o meno.

```
if ($this->isExistingEmail($email)) {  
    //check if the password is correct  
    return $this->isPasswordCorrect($email, $password);  
}  
  
return false;
```

3.3.9.3 Metodo isUserLogged

Questo metodo controlla se l'utente è già loggato, per evitare che ad ogni azione vengano richieste le credenziali di accesso.

Per fare ciò controlla se nelle variabili di sessioni sono presenti delle credenziali valide.

```
//check if the variables are initialized  
if (isset($_SESSION['email']) && isset($_SESSION['password'])) {  
    return $this->checkCredentials($_SESSION['email'], $_SESSION['password']);  
}  
return false;
```

3.3.9.4 Metodo logout

Questo metodo statico viene utilizzato dai controller per eliminare le variabili di sessione, così da cancellare le credenziali. In questo modo per accedere alle pagine bisogna effettuare il login.

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

3.3.9.5 Metodi isExistingEmail e isPasswordCorrect

Il metodo **isExistingEmail(\$email)** controlla e ritorna, tramite valore booleano, se l'email esiste all'interno della banca dati.

```
//get the number of rows that contain the user email (MAX 1)  
$prepared_query = self::$conn->prepare("SELECT count(*) FROM users WHERE email = :email");  
$prepared_query->bindParam(':email', $email, PDO::PARAM_STR);  
$prepared_query->execute();  
$res = $prepared_query->fetch();  
  
//check if the row count is 1  
return intval($res[0]) == 1;
```

Il metodo **isPasswordCorrect(\$email, \$password)** ritorna se la password relativa all'email passata come parametro al metodo è corretta o meno.

```
//get the password from database
$prepared_query = self::$conn->prepare("SELECT password FROM users WHERE email =
:email");
$prepared_query->bindParam(':email', $email, PDO::PARAM_STR);
$prepared_query->execute();
$res = $prepared_query->fetch();

$hashed_password = $res[0];

//check if the inserted password is correct
return password_verify($password, $hashed_password);
```

3.3.9.6 Metodo isAdminUser

Questo metodo statico ritorna se l'utente, identificato dall'email passata come parametro **\$email**, è admin o meno:

- Se ritorna 0 → l'utente non è admin
- Se ritorna 1 → l'utente è admin

```
//get is_admin value
$prepared_query = self::$conn->prepare("SELECT is_admin FROM users WHERE email =
:email");
$prepared_query->bindParam(':email', $email, PDO::PARAM_STR);
$prepared_query->execute();

//get result
$res = $prepared_query->fetch();
$is admin = $res[0];

return intval($is_admin);
```

3.3.9.7 Metodo createUser

Questo metodo prova a creare l'utente utilizzando l'oggetto User passato come parametro.

Come prima cosa controlla con il metodo **isExistingEmail(\$email)** che l'email non sia già utilizzata da qualche utente. Successivamente tramite i metodi della classe User.php vengono estratti i dati dell'utente e inseriti nella query.

```
if (!(!$this->isExistingEmail($user->getEmail()) )) {  
    //prepare query  
    $prepared_query = self::$conn->prepare("INSERT INTO USERS (name, surname, email,  
    password, is_admin, change_pass) VALUES (:name, :surname, :email, :password,  
    :is_admin, :change_pass)";  
  
    //get params  
    $name = $user->getName();  
    $surname = $user->getSurname();  
    $email = $user->getEmail();  
    $password = $user->getPassword();  
    $is_admin = $user->getAdmin();  
    $change_pass = $user->getChangePass();  
    ...  
  
    $prepared_query->execute();  
  
    return true;  
}
```

3.3.9.8 Metodo getEmailById

Questo metodo richiede come parametro l'id dell'utente, e tramite quest'ultimo ritorna la relativa email.

```
//prepare query  
$prepared_query = self::$conn->prepare("SELECT id FROM USERS WHERE EMAIL =  
:email");  
$prepared_query->bindParam(':email', $email, PDO::PARAM_STR);  
$prepared_query->execute();  
  
$res = $prepared_query->fetch();  
return $res[0];
```

3.3.9.9 Metodo deleteUser

Questo metodo prova ad eliminare l'utente utilizzando l'id ricevuto come parametro. Se l'id non esiste l'operazione non fallisce ma nessun record viene eliminato.

```
//prepare query  
$prepared_query = self::$conn->prepare("DELETE FROM USERS WHERE ID = :id");  
$prepared_query->bindParam(':id', $id, PDO::PARAM_INT);  
$prepared_query->execute();  
  
return true;
```

3.3.9.10 Metodo requestChangePass

Questo metodo è stato creato ma al momento non è implementato ed utilizzato da nessuna classe (vedi capitolo implementazioni future). Questo metodo non fa altro che settare il valore dell'attributo **change_pass** della tabella utenti ad 1.

```
//prepare query
$prepared_query = self::$conn->prepare("UPDATE USERS SET change_pass = 1 WHERE
id = :id");
$prepared_query->bindParam(':id', $id, PDO::PARAM_INT);
$prepared_query->execute();
```

Questo metodo richiede come parametro l'id dell'utente che ha richiesto il cambio della password.

3.3.9.11 Metodo modifyUser

Questo metodo si occupa della modifica di un utente già esistente all'interno della banca dati. Come parametri riceve l'oggetto utente, che contiene i valori da aggiornare, e l'id dell'utente da modificare.

Come prima cosa viene controllato che l'email passata corrisponda a quella già registrata, in tal caso si evita di fare una modifica dell'email e viene quindi omessa nella query. In questo metodo la query viene costruita a tappe. Se l'email non corrisponde viene controllato che non la possieda già un altro utente.

```
//check if the email is the current user email
if ($this->getIdByEmail($user->getEmail()) === intval($id)) {
    $sql .= "UPDATE USERS set name = :name, surname = :surname, is_admin =
:is_admin, change_pass = :change_pass";

} else {

    //check if the email is already used
    if (!$this->isExistingEmail($user->getEmail())) {
        $change_email = true;
        $sql .= "UPDATE USERS set name = :name, surname = :surname, email =
:email, is_admin = :is_admin, change_pass = :change_pass";
    }
}
```

Successivamente viene controllato che la password non sia vuota, in modo da non inserirla nella query se l'utente non ha richiesto un cambio password.

```
//check if update password
if (!empty($password)) {
    $sql .= ", password = :password";
}
```

Infine viene inserito aggiunta la condizione nella query per identificare l'utente e successivamente, dopo aver effettuato il controllo sui vari valori, viene eseguita l'operazione.

```
$sql .= " WHERE ID = :id";

//prepare query
$prepared_query = self::$conn->prepare($sql);
...
$prepared_query->execute();
```

3.4 Pagine

Per quanto riguarda le pagine ho utilizzato il framework Material Design Bootstrap. Questo framework mette a disposizione molte classi css che facilitano, oltre a rendere il risultato più presentabile, e velocizzano il lavoro di creazioni delle interfacce.

3.4.1 Pagina di login

Per la pagina di login ho scelta una struttura molto semplice che rispecchia il design delle interfacce che ho progettato.

Vi è una maschera centrata nello schermo con all'interno i campi in cui inserire la propria email e la password.

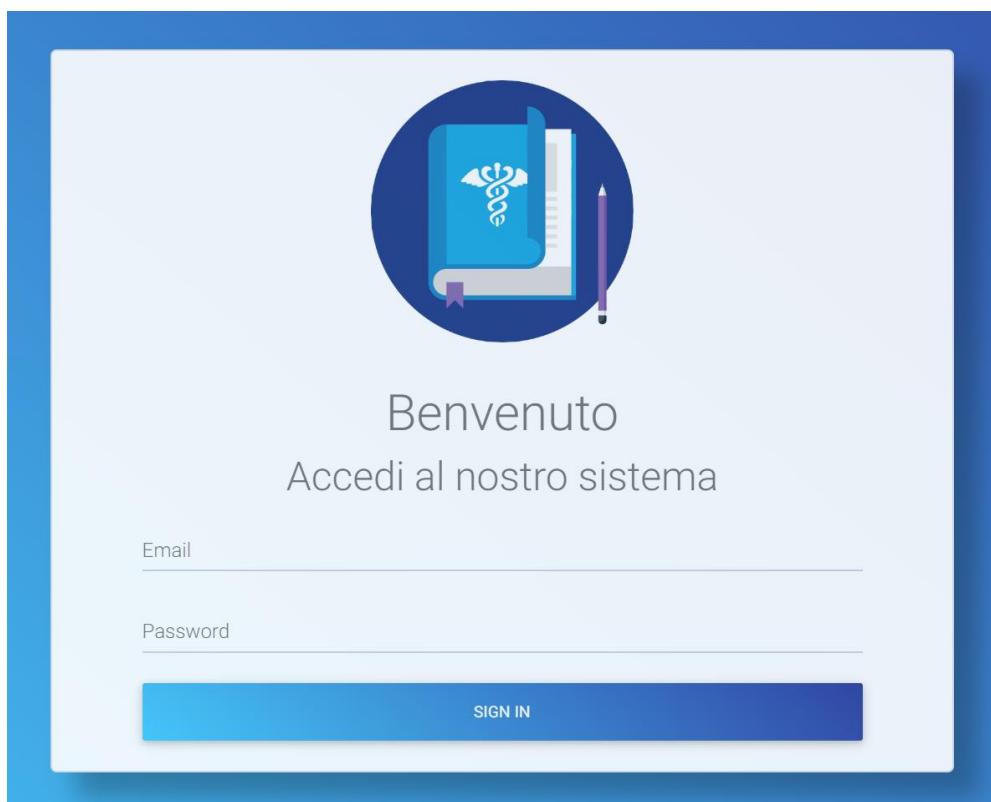


Figure 25 Pagina di login

Se le credenziali sono corrette si accede alla pagina di ricerca casi, altrimenti appare un messaggio di errore.

Errore! Le credenziali fornite non sono corrette

Figure 26 Messaggio credenziali sbagliate

3.4.2 Pagina di errore

Questa pagina viene mostrata quando l'utente non riesce a connettersi al database. È una semplice pagina che avverte l'utente che il tentativo di connessione al database è fallito.

In ogni metodo dell'applicazione (di interazione con l'utente) viene controllata la connessione al database e, se questa fallisce, viene chiamato il metodo statico **noDatabaseConnection()** del controller dbError.php.



Figure 27 Pagina errore di connessione alla banca dati

Nel metodo **noDatabaseConnection()** oltre a mostrare la pagina di errore viene richiamato il metodo statico **logout()** del controller users.php per evitare che un utente lasci la postazione tenendo la pagina aperta e qualcun altro, una volta che la connessione è stata stabilita, possa ricaricare la pagina ed effettuare operazioni con le credenziali del primo utente.

3.4.3 Pagina di ricerca casi

3.4.3.1 Filtro di ricerca

A questa pagina hanno accesso tutti gli utenti che effettuano il login. Si possono visualizzare i vari casi e filtrarli in base ad alcune opzioni disponibili.

Filtri disponibili:

- Ordine:
 - Più recenti
 - Meno recenti
 - Più ricorrenti
- Testo
- Categoria
- Data specifica



The screenshot shows a search interface with the following elements:

- Top right button:** NASCONDI FILTRI (Hide Filters)
- Section title:** Filtri Base:
- Order dropdown:** Scegli ordinamento (Casi più recenti)
- Advanced section title:** Filtri avanzati:
- Search input:** Inserisci parola chiave (Ricerca)
- Category dropdown:** Selezione categoria (Tutte le categorie)
- Date input:** Scegli una data (es. 20.10.2019)
- Bottom left button:** CERCA (Search)

Figure 28 Campi di ricerca

Tutte i filtri possono essere combinati tra di loro, si può quindi ad esempio effettuare una ricerca selezionando un tipo di ordinamento e una parola chiave.



The screenshot shows a search interface with the following elements:

- A blue button at the top right labeled "NASCONDI FILTRI" (Hide Filters).
- A section titled "Filtri Base:" with a dropdown menu set to "Casi meno recenti".
- A section titled "Filtri avanzati:" containing three input fields:
 - "Inserisci parola chiave" with the value "Schermo blu".
 - "Selezione categoria" with the value "Diagnostica".
 - "Scegli una data" with the value "06/11/2019".
- A blue "CERCA" (Search) button at the bottom left.

Figure 29 Filtri combinati tra loro

Se una ricerca non produce nessun risultato viene stampato un messaggio che avvisa l'utente che nessun caso corrisponde ai criteri di ricerca che ha impostato.

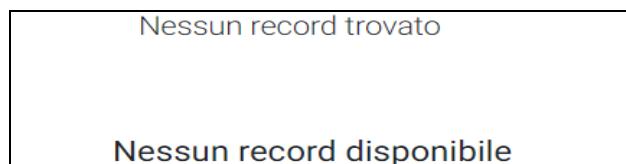


Figure 30 Nessun record trovato

I casi vengono mostrati con una semplice formattazione indicando le caratteristiche principali (nome, id, categoria, variante, data creazione, descrizione, numero di ripresentazioni). Ho preferito fare in modo che tutti i risultati vengano caricati una volta sola, per poi poter navigare velocemente e liberamente tra i casi. In questo modo impiega più tempo quando viene eseguita la query ma successivamente tramite il campo cerca tra i risultati si possono effettuare ricerche istantanee.

Nome: Schermo blu

ID: 12149
Categoria: trouble shooting
Variante di: -
Data creazione: 12.11.2019 14:53:04
Descrizione:
Lo schermo del computer, con sistema operativo windows 10 home, risulta blu ed è bloccato. Staccare il cavo di alimentazione e inserirlo nuovamente o tenere premuto il tasto di accensione fino allo spegnimento.
Numero di ripresentazioni: 0

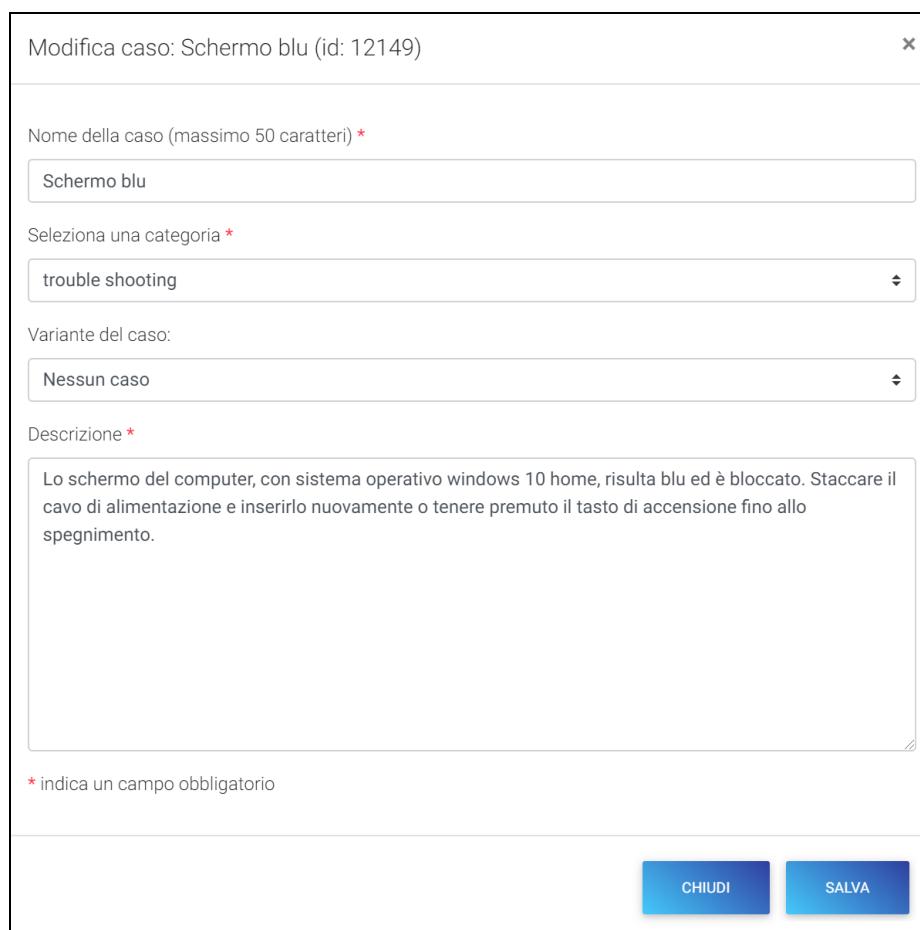
[MODIFICA](#) [ELIMINA](#)

Figure 31 Esempio di caso di documentazione

3.4.3.2 Aggiunta e modifica caso

Qualunque utente può aggiungere un caso. Se si clicca sul bottone “**Aggiungi un caso**” si apre una finestra modale (che corrisponde a quella di modifica di un caso) in cui si deve inserire il titolo del caso, la categoria alla quale appartiene, se è una variante di un altro caso (selezionare il caso di cui è una variante) e la descrizione. Se la creazione dovesse fallire viene mostrato all’utente il relativo errore.

Se si clicca il bottone “modifica” su un caso (solo gli admin possono effettuare questa operazione) si apre una finestra modale in cui si possono modificare i dati del caso. Una volta inseriti i dati si può decidere se salvare le modifiche o meno.



Modifica caso: Schermo blu (id: 12149) ×

Nome della caso (massimo 50 caratteri) *

Schermo blu

Selezione una categoria *

trouble shooting

Variante del caso:

Nessun caso

Descrizione *

Lo schermo del computer, con sistema operativo windows 10 home, risulta blu ed è bloccato. Staccare il cavo di alimentazione e inserirlo nuovamente o tenere premuto il tasto di accensione fino allo spegnimento.

* indica un campo obbligatorio

CHIUDI SALVA

Figure 32 Finestra aggiunta/modifica caso

In caso di fallimento della modifica viene mostrato il messaggio con il relativo errore e i dati inseriti dall’utente rimangono in memoria. In questo modo l’utente non deve riscrivere i dati inseriti in precedenza.

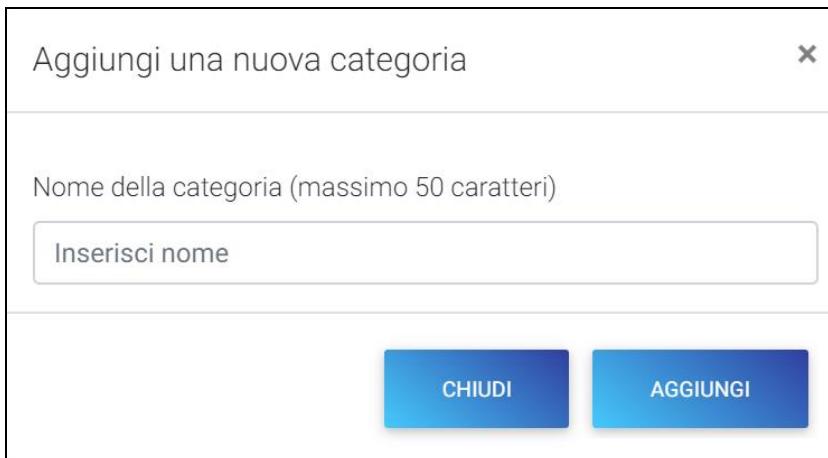
3.4.3.3 Aggiunta, eliminazione categoria

Solo gli admin possono aggiungere o eliminare una categoria. Per aggiungere o eliminare una categoria basta cliccare sui relativi buttoni posti sotto la maschera dei filtri.

[Aggiungi caso](#)
[Aggiungi una categoria](#)
[Elimina una categoria](#)

Figure 33 Buttoni funzionalità

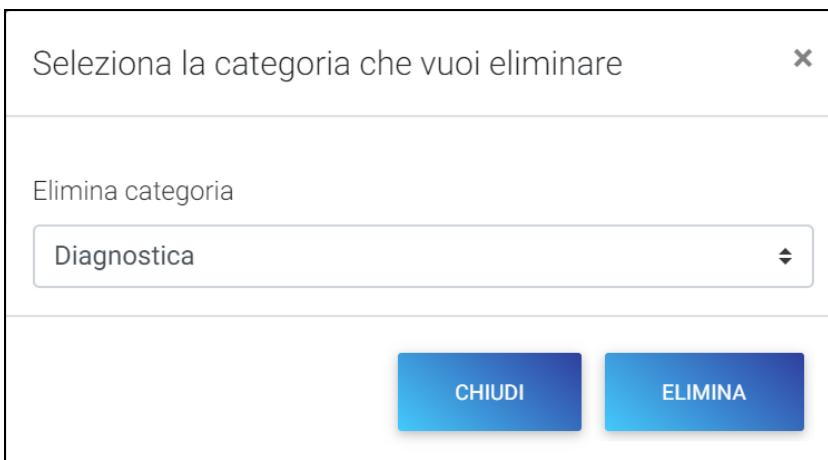
Per aggiungere una categoria bisogna inserire un nome che non esista già (messaggio di errore se la categoria esiste già al momento della creazione) e che non superi i 50 caratteri.



The dialog box has a title 'Aggiungi una nuova categoria' and a close button 'X'. It contains a text input field labeled 'Nome della categoria (massimo 50 caratteri)' with placeholder text 'Inserisci nome'. At the bottom are two blue buttons: 'CHIUDI' (Close) on the left and 'AGGIUNGI' (Add) on the right.

Figure 34 Finestra aggiunta categoria

Per eliminare una categoria basta selezionarla, premere su elimina e confermare l'eliminazione.

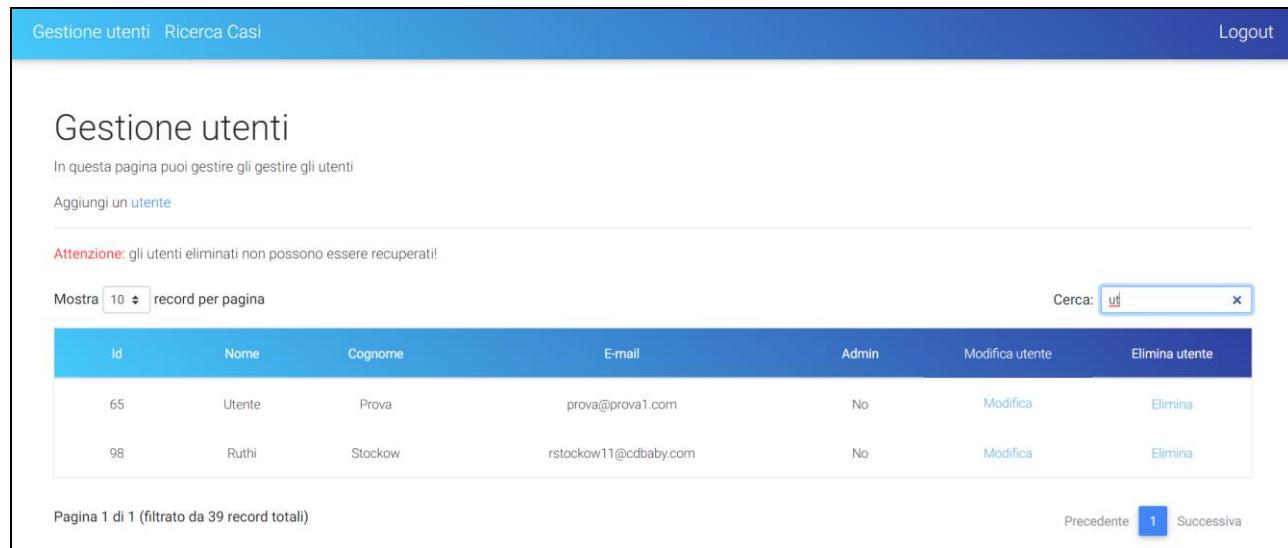


The dialog box has a title 'Selezione la categoria che vuoi eliminare' and a close button 'X'. It contains a dropdown menu labeled 'Elimina categoria' with the option 'Diagnostica' selected. At the bottom are two blue buttons: 'CHIUDI' (Close) on the left and 'ELIMINA' (Delete) on the right.

Figure 35 Finestra eliminazione categoria

3.4.4 Pagina gestione utenti

Solo l'admin può accedere a questa pagina in cui sono vengono mostrati gli utenti registrati. Si possono aggiungere nuovi utenti oppure rimuoverli/modificarli.



The screenshot shows a web-based application for managing users. At the top, there's a blue header bar with the text "Gestione utenti Ricerca Casi" on the left and "Logout" on the right. Below the header, the main title "Gestione utenti" is displayed. A sub-instruction "In questa pagina puoi gestire gli utenti registrati" follows. There's a link "Aggiungi un utente". A warning message "Attenzione: gli utenti eliminati non possono essere recuperati!" is present. On the left, a dropdown menu "Mostra 10 record per pagina" is shown. On the right, there's a search bar "Cerca: ut" with a clear button. The central part of the page is a table listing users:

ID	Nome	Cognome	E-mail	Admin	Modifica utente	Elimina utente
65	Utente	Prova	prova@prova1.com	No	Modifica	Elimina
98	Ruthi	Stockow	rstockow11@cdbaby.com	No	Modifica	Elimina

At the bottom, it says "Pagina 1 di 1 (filtrato da 39 record totali)". Navigation links "Precedente" and "Successiva" are shown, with the page number "1" highlighted in a blue box.

Figure 36 Pagina di gestione utenti

Quando si vuole aggiungere un nuovo utente si apre una finestra modale in cui inserire i dati dell'utente.

Inserisci nuovo utente ×

Nome (massimo 50 caratteri)

Cognome (massimo 50 caratteri)

Email

Utente base Utente admin

Password (min 8 caratteri, max 50 caratteri, 1 maiuscola ed un numero)

Conferma password

CHIUDI INVIA

Figure 37 Finestra di aggiunta utente

Se durante la creazione avviene qualche errore viene stampato tramite un popup. I dati inseriti in precedenza non vanno persi, se si clicca di nuovo su crea un utente sono ancora presenti.

Errore! Email già utilizzata

Figure 38 Messaggio di errore, email già utilizzata

Inserisci nuovo utente ×

Nome (massimo 50 caratteri)
Marco

Cognome (massimo 50 caratteri)
Alonso

Email
marco.alonso@hotmail.com

Utente base Utente admin

Password (min 8 caratteri, max 50 caratteri, 1 maiuscola ed un numero)
Inserire password

Conferma password
Confermare password

CHIUDI INVIA

Figure 39 Finestra di aggiunta, informazioni conservative

Quando invece si desidera eliminare un utente basta premere sul **Elimina**. Prima di eliminare in modo definitivo un utente viene chiesta una conferma dell'azione tramite un modal.

Se un utente viene eliminato non vi è più nessun modo di recuperare lui ed i suoi dati.

Elimina utente ×

Sei sicuro di voler eliminare l'utente Rodi?

CHIUDI ELIMINA

Figure 40 Conferma eliminazione utente

Premendo su “modifica” si apre una finestra modale di modifica dell’utente, in cui si può cambiare il nome, cognome, indirizzo email, privilegi e password (opzionale).

Modifica utente

Nome (massimo 50 caratteri) *

Cognome (massimo 50 caratteri) *

Email *

Utente base Utente admin

Password (min 8 caratteri, max 50 caratteri, 1 maiuscola ed un numero)

* indica un campo obbligatorio

CHIUDI **SALVA**

Figure 41 Finestra modifica utente

4 Test

4.1 Protocollo di test

Test Case:	TC-001	Nome:	Pagina di login
Riferimento:	REQ-01 REQ-07		
Descrizione:	Inserire credenziali di accesso non corrette		
Prerequisiti:	Pagina di login, sistema di login, database con presenti i dati		
Procedura:	<ol style="list-style-type: none">1. Aprire il browser e cercare localhost:8080/knowledge_base2. Inserire nei campi di testo nome utente e password non corretti3. Premere il bottone “accedi”		
Risultati attesi:	Viene mostrato un messaggio di errore all’utente in cui viene comunicato che le credenziali inserite non sono corrette.		

Test Case:	TC-002	Nome:	Pagina di login
Riferimento:	REQ-01 REQ-07		
Descrizione:	Inserire credenziali di accesso corrette		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none">1. Aprire il browser e cercare localhost:8080/knowledge_base2. Inserire nei campi di testo nome utente e password corretti3. Premere il bottone “accedi”		
Risultati attesi:	L’utente riesce ad accedere in modo corretto e reindirizzato nella pagina di ricerca casi.		

Test Case:	TC-003	Nome:	Visualizzazione utenti, pagina di gestione utenti
Riferimento:	REQ-02	(admin)	
Descrizione:	Visualizzazione utenti registrati		
Prerequisiti:	Pagina di login, sistema di login, database con presenti i dati, pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul menu in alto la voce “Gestione utenti” 		
Risultati attesi:	L’utente riesce a visualizzare gli utenti in modo corretto e può cercarli tramite il campo testuale cerca.		

Test Case:	TC-004	Nome:	Creazione nuovo utente, operazione fallita
Riferimento:	REQ-02 REQ-06		
Descrizione:	Tentativo fallito di creazione utente		
Prerequisiti:	Pagina di login, sistema di login, database con presenti i dati, pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul menu in alto la voce “Gestione utenti” 5. Premere su “Aggiungi un utente” 6. Inserire i dati rispettando la sintassi 7. Inserire due password diverse 8. Premere “crea” 9. Ripetere l’operazione non rispettando la sintassi per ogni campo richiesto (vedi specifiche per ogni campo richiesto nel testo sopra all’input) 		
Risultati attesi:	All’utente appare un messaggio di errore esplicativo.		

Test Case:	TC-005	Nome:	Creazione nuovo utente, operazione riuscita
Riferimento:	REQ-02 REQ-06		
Descrizione:	Tentativo fallito di creazione utente		
Prerequisiti:	Pagina di login, sistema di login, database con presenti i dati, pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul menu in alto la voce “Gestione utenti” 5. Premere su “Aggiungi un utente” 6. Inserire i dati rispettando la sintassi dei vari campi 7. Premere “crea” 		
Risultati attesi:	All’utente appare un messaggio in cui viene informato che l’operazione ha avuto successo. L’utente appare nella lista degli utenti.		

Test Case:	TC-006	Nome:	Eliminazione utente, operazione riuscita
Riferimento:	REQ-02		
Descrizione:	Eliminazione di un utente		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati, pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul menu in alto la voce “Gestione utenti” 5. Premere su “Elimina” sull’utente che si desidera eliminare 6. Confermare l’eliminazione 		
Risultati attesi:	All’utente appare un messaggio in cui viene informato che l’operazione ha avuto successo. L’utente non appare nella lista degli utenti.		

Test Case:	TC-007	Nome:	Eliminazione utente, operazione fallita
Riferimento:	REQ-02		
Descrizione:	Eliminazione del l'utente con cui si è loggati		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati, pagina gestione utenti		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul menu in alto la voce “Gestione utenti” 5. Premere “Elimina” sull'utente con cui si è loggati 6. Confermare l'eliminazione 		
Risultati attesi:	All'utente appare un messaggio di errore in cui viene avvisato che non può eliminare il proprio account. L'operazione non è andata a buon fine.		

Test Case:	TC-008	Nome:	Visualizzazione casi
Riferimento:	REQ-03		
Descrizione:	Visualizzazione casi presenti nel sistema		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti 3. Premere il bottone “accedi” 4. Aspettare che la pagina sia caricata 		
Risultati attesi:	In questa pagina vengono mostrati all'utente i casi presenti nel sistema. Tramite il campo di testo “cerca in live” l'utente può cercare in modo dinamico i casi inserendo del testo.		

Test Case:	TC-009	Nome:	Filtraggio casi
Riferimento:	REQ-03 REQ-05		
Descrizione:	Filtraggio casi		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti 3. Premere il bottone “accedi” 4. Aspettare che la pagina sia caricata 5. Premere sul bottone “Mostra filtri” 6. Inserire il tipo di ordinamento desiderato 7. Inserire il testo da cercare 8. Inserire la categoria del caso che si vuol filtrare 9. Inserire la data di creazione del caso 10. Premere il bottone “cerca” 		
Risultati attesi:	All’utente vengono mostrati i casi che corrispondono ai filtri applicati in precedenza. Tramite il campo di testo “cerca in live” l’utente può cercare in modo dinamico tra i casi filtrati inserendo del testo.		

Test Case:	TC-010	Nome:	Aggiunta caso, operazione fallita
Riferimento:	REQ-03		
Descrizione:	Aggiunta di un caso, non inserendo i dati rispettando la sintassi		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti 3. Premere il bottone “accedi” 4. Aspettare che la pagina sia caricata 5. Premere su “Aggiungi caso” 6. Inserire nei campi “nome” oppure “descrizione” il carattere “spazio” 7. Inserire gli altri dati rispettando la sintassi 8. Premere “Crea” 		
Risultati attesi:	All’utente viene mostrato un messaggio di errore che indica che i campi testuali devono contenere del testo.		

Test Case:	TC-011	Nome:	Aggiunta caso, operazione riuscita
Riferimento:	REQ-03		
Descrizione:	Aggiunta di un caso, inserendo i dati rispettando la sintassi		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti 3. Premere il bottone “accedi” 4. Aspettare che la pagina sia caricata 5. Premere su “Aggiungi caso” 6. Riempire i dati con dei valori che rispettano la sintassi 7. Premere “Crea” 		
Risultati attesi:	All’utente viene mostrato un messaggio di successo che indica che il caso è stato creato con successo. Il caso appare nella lista dei casi.		

Test Case:	TC-012	Nome:	Eliminazione caso
Riferimento:	REQ-03		
Descrizione:	Eliminazione di un caso		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “accedi” 4. Premere sul bottone “elimina” sul caso desiderato 5. Premere “Chiudi” per annullare l’eliminazione o “Elimna” per confermare l’eliminazione 		
Risultati attesi:	Se l’utente non conferma l’eliminazione e preme “Chiudi” il caso non viene eliminato. Altrimenti, se l’utente preme “Elimna” il caso viene eliminato, viene mostrato un messaggio di successo ed il caso viene eliminato. Il caso non appare più tra la lista di casi.		

Test Case:	TC-013	Nome:	Aggiunta di una categoria, operazione fallita
Riferimento:	REQ-03 REQ-08		
Descrizione:	Aggiunta di una categoria, operazione fallita		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “Aggiungi categoria” 4. Inserire il nome di una categoria già esistente 5. Premere il bottone “crea” 		
Risultati attesi:	All’utente viene mostrato un messaggio di errore che avvisa l’utente che la categoria esiste già. L’operazione fallisce.		

Test Case:	TC-014	Nome:	Aggiunta di una categoria, operazione riuscita
Riferimento:	REQ-03 REQ-08		
Descrizione:	Aggiunta di una categoria, operazione riuscita		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “Aggiungi categoria” 4. Inserire il nome di una categoria che non esistente già 5. Premere il bottone “crea” 		
Risultati attesi:	All’utente viene mostrato un messaggio di successo che avvisa l’utente che la categoria è stata creata con successo. L’operazione è riuscita.		

Test Case:	TC-015	Nome:	Eliminazione categoria
Riferimento:	REQ-03		
Descrizione:	Eliminazione categoria		
Prerequisiti:	Pagina di login, sistema di login, pagina di ricerca casi, database con presenti i dati		
Procedura:	<ol style="list-style-type: none"> 1. Aprire il browser e cercare localhost:8080/knowledge_base 2. Inserire nei campi di testo nome utente e password corretti di un amministratore 3. Premere il bottone “Eliminazione categoria” 4. Selezionare la categoria che si desidera eliminare 5. Premere il bottone “Elimina” 6. Premere il bottone “Chiudi” per annullare l’operazione o “Elimina” per eliminare la categoria 		
Risultati attesi:	Se l’utente preme “Chiudi” l’azione viene annullata. Se l’utente preme “Elimina” la categoria viene eliminata e viene mostrato un messaggio di operazione riuscita all’utente.		

4.2 Risultati test

Codice-Test	Risultato	Note
TC-01	Passato	
TC-02	Passato	
TC-03	Passato	
TC-04	Passato	
TC-05	Passato	
TC-06	Passato	
TC-07	Passato	
TC-08	Passato	
TC-09	Passato	
TC-10	Passato	
TC-11	Passato	
TC-12	Passato	
TC-13	Passato	
TC-14	Passato	
TC-15	Passato	

4.3 Mancanze/limitazioni conosciute

L'applicativo presenta alcune limitazioni. Il layout dei "casi" potrebbe essere migliorato, in modo da facilitare la lettura e la navigazione. Potrebbe essere aggiunto un bottone che rimanda al caso padre quando viene premuto.

Manca una visualizzazione riassuntiva dei casi, un piccolo elenco in cui vengono mostrati i titoli dei casi e premendoci sopra si viene riportati al caso desiderato.

Nell'applicazione manca inoltre un sistema di recupero autonomo della password. Al momento solo gli amministratori possono modificare le informazioni personali degli utenti, ovvero nome, cognome, email, tipo di utente e password.

Modifica utente

Nome (massimo 50 caratteri) *

Utente

Cognome (massimo 50 caratteri) *

Prova

Email *

prova@prova1.com

Utente base Utente admin

Password (min 8 caratteri, max 50 caratteri, 1 maiuscola ed un numero)

Inserire password

* indica un campo obbligatorio

CHIUDI SALVA

Figure 42 Finestra modifica utente

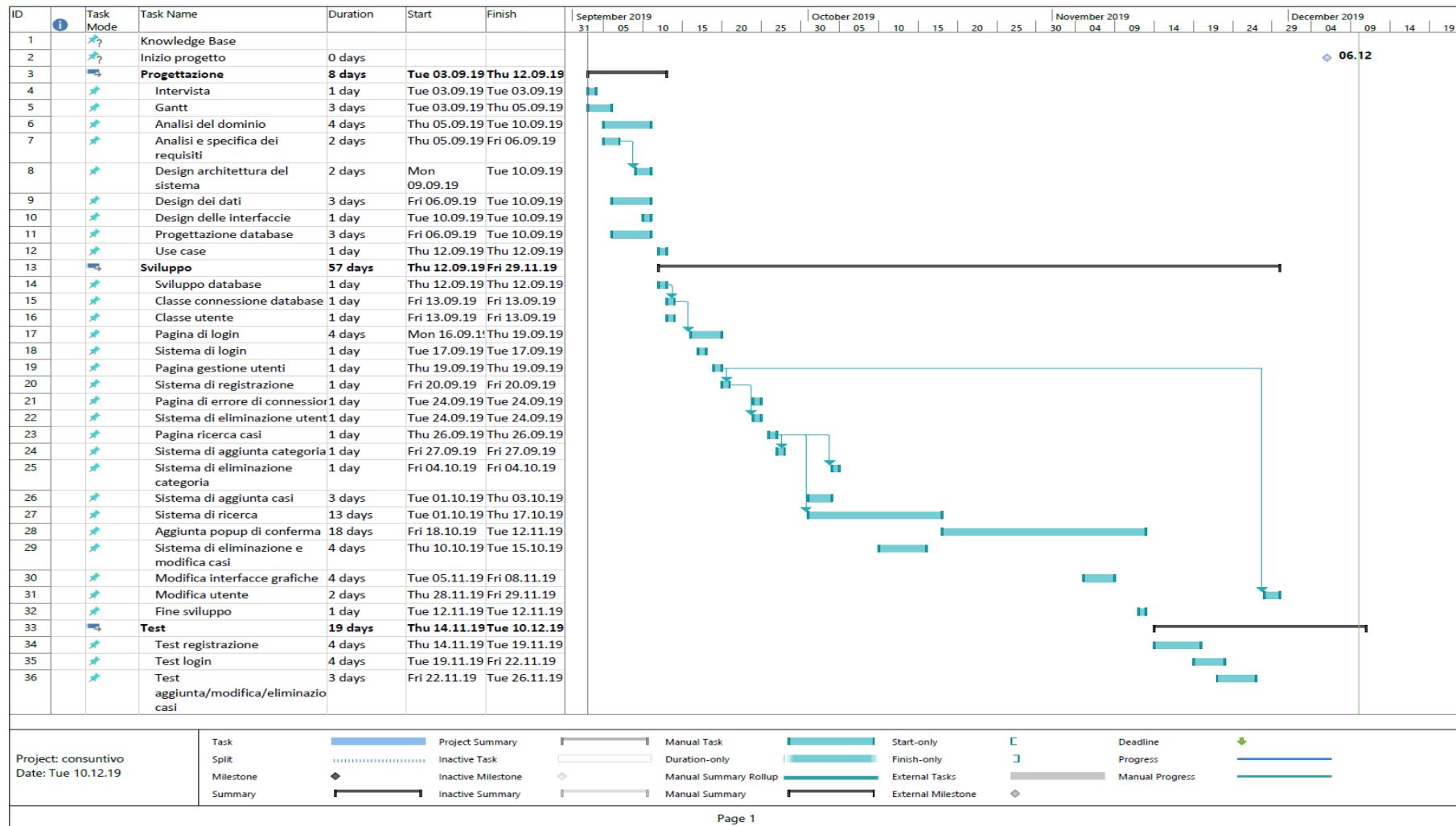
5 Consuntivo

Il consuntivo finale non è molto diverso da quello che ho pianificato nella fase di analisi, ci sono state attività che sono durate leggermente più del previsto ed altre di meno.

Alcune attività risultano più lunghe perché sono state effettuate delle modifiche in un secondo momento oppure sono state lasciate delle parti in sospeso. Sono state inoltre aggiunte delle task che nel preventivo le avevo considerate come una parte di un'altra attività, quindi nel consuntivo risultano un maggior numero di attività rispetto al preventivo per questo motivo.

Durante le lezioni ho seguito la mia pianificazione passo per passo e mi è stato molto utile in quanto i tempi corrispondevano quasi durante tutto il progetto. Sono riuscito ad implementare tutte le funzionalità che mi sono state richieste dal committente.

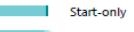
In conclusione il consuntivo corrisponde quasi al preventivo se non con alcune variazioni di durate delle attività.



Page 1

Titolo del progetto: Knowledge Base
 Alunno/a: Bryan Beffa
 Classe: Info I4AA
 Anno scolastico: 2019/2020
 Docente responsabile: Ivan Raimondi

ID	Task Mode	Task Name	Duration	Start	Finish	September 2019					October 2019					November 2019					December 2019						
						31	05	10	15	20	25	30	05	10	15	20	25	30	04	09	14	19	24	29	04	09	14
37	✓	Test aggiunta/eliminazione categoria	4 days	Tue 03.12.19	Fri 06.12.19																						
38	✓	Test ricerca	3 days	Fri 06.12.19	Tue 10.12.19																						
39	✓	Documentazione	76 days	Tue 03.09.19	Tue 17.12.19																						

Project: consultivo Date: Tue 10.12.19	Task		Project Summary		Manual Task		Start-only		Deadline	
	Split		Inactive Task		Duration-only		Finish-only		External Tasks	
	Milestone		Inactive Milestone		Manual Summary Rollup		External Milestone		Manual Progress	
	Summary		Inactive Summary		Manual Summary		External Milestone		Manual Progress	

Page 2

6 Conclusioni

6.1 Sviluppi futuri

Ho pensato a molti sviluppi futuri per questa applicazione. Come prima cosa inserirei un sistema di recupero password autonomo con l'utilizzo di un server mail. In questo modo quando un utente ha bisogno di effettuare un cambio password non deve recarsi dagli amministratori ma lo può effettuare individualmente ricevendo un codice per cambiare la password per email.

Si potrebbe aggiungere il cambio di informazioni personali autonomo, anche in questo caso non si dovrebbe più andare dall'amministratore per cambiare ad esempio indirizzo email.

Come consigliato dal docente Raimondi aggiungerei un layout riassuntivo dei casi di documentazione, così da potersi muovere cliccando sul caso desiderato direttamente dalla finestra riassuntiva.

6.2 Considerazioni personali

Sono soddisfatto del prodotto finale. Questa applicazione web svolge le funzioni richieste dal committente e può essere facilmente portata avanti da altre persone. Ho cercato di rendere l'applicazione user-friendly elaborando layout semplici e intuitivi.

Personalmente ho trovato il progetto interessante e mi ha permesso di testare ciò che ho appreso durante la mia formazione professionale. Se avessi avuto più tempo a disposizione avrei potuto aggiungere più funzionalità all'applicazione web.

7 Bibliografia

Per la realizzazione di questo progetto non ho consultato riviste, libri, manuali ma solo i siti web descritti nel capitolo sitografia.

7.1 Sitografia

<https://www.draw.io/>, Draw.io, dal 05.09.2018 al

<https://mdbbootstrap.com/docs/jquery/getting-started/download/>, material design bootstrap download, 17.09.2019

<https://mdbbootstrap.com/docs/jquery/forms/basic/>, form base di material design, 17.09.2019

<https://mdbbootstrap.com/docs/jquery/navigation/navbar/>, navbar material design, 19.09.2019

<https://mdbbootstrap.com/docs/jquery/components/alerts/>, material design alert, 20.09.2019

<https://stackoverflow.com/questions/>, stackoverflow, 15.10.2019 – 20.12.2019

<https://mockaroo.com/>, mackaroo creazione dati, 25.10.2019

Titolo del progetto:	Knowledge Base
Alunno/a:	Bryan Beffa
Classe:	Info I4AA
Anno scolastico:	2019/2020
Docente responsabile:	Ivan Raimondi

8 Allegati

Elenco degli allegati:

- Quaderno dei compiti
- Guida installazione XDebug
- Diari di lavoro
- Codici sorgente, git <https://github.com/bryanbeffa/Knowledge-Base>

Guide



Installazione XDEBUG su XAMPP

Documento: Installazione XDEBUG su XAMPP
Alunno/a: Bryan Beffa
Classe: Info I4AA
Anno scolastico: 2019/2020
Docente responsabile: Ivan Raimondi



Sommario

XDEBUG.....	3
Installazione	3
Test installazione XDebug	5
Phpstorm.....	5
Configurazione XDebug.....	5
Test su phpstorm	8
Fonti:	8

Documento: Installazione XDEBUG su XAMPP
Alunno/a: Bryan Beffa
Classe: Info I4AA
Anno scolastico: 2019/2020
Docente responsabile: Ivan Raimondi



XDEBUG

Installazione

Per installare XDEBUG sul webserver XAMPP bisogna eseguire i seguenti passaggi:

Scaricare il file .dll di xdebug con la versione corretta dal sito ufficiale (<https://xdebug.org/download>), oppure utilizzare il file presente nella cartella **ext** di php. Per verificare la propria versione di php aprire il **cmd** ed eseguire il seguente comando.

```
php -version
```

Il risultato sarà il seguente

```
PHP 7.1.33 (cli) (built: Oct 23 2019 09:24:14) ( ZTS MSVC14 (Visual C++ 2015) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies
```

Figure 1 Risultato comando

Una volta scaricato il file bisogna spostarlo nella cartella **ext** presente nella cartella in cui è installato php.

Successivamente modificare il file di configurazione di php **php.ini** ed inserire infondo al file il seguente codice.

```
[XDebug]
zend_extension = "c:\xampp\php\ext\php_xdebug.dll"
;zend_extension = "c:\xampp\php\ext\php_xdebug-2.7.2-7.3-vc15-
x86_64.dll"
xdebug.remote_autostart = 1
xdebug.profiler_append = 0
xdebug.profiler_enable = 0
xdebug.profiler_enable_trigger = 0
xdebug.profiler_output_dir = "c:\xampp\tmp"
;xdebug.profiler_output_name = "cachegrind.out.%t-%s"
xdebug.remote_enable = 1
xdebug.remote_handler = "dbgp"
xdebug.remote_host = "127.0.0.1"
xdebug.remote_log = "c:\xampp\tmp\xdebug.txt"
xdebug.remote_port = 9000
xdebug.trace_output_dir = "c:\xampp\tmp"
;36000 = 10h
xdebug.remote_cookie_expire_time = 36000
```



Attenzione: a dipendenza del file che si desidera utilizzare commentare l'alternativa (prime opzioni del codice da aggiungere). In questo caso io uso il file già presente in **xampp** e commento quindi la seconda opzione.

Documento: Installazione XDEBUG su XAMPP
Alunno/a: Bryan Beffa
Classe: Info I4AA
Anno scolastico: 2019/2020
Docente responsabile: Ivan Raimondi

Test installazione XDebug

A questo punto salvare le modifiche e riavviare il webserver. Per verificare che l'installazione sia andata a buon fine ripetere il comando **php --version** nel cmd.

Se l'operazione è andata a buon fine, nel risultato sarà presente il messaggio **Xdebug** e la relativa versione.

```
PHP 7.1.33 (cli) (built: Oct 23 2019 09:24:14) ( ZTS MSVC14 (Visual C++ 2015) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies
    with Xdebug v2.6.1, Copyright (c) 2002-2018, by Derick Rethans
```

Figure 2 xdebug installato con successo

Phpstorm

Configurazione XDebug

Aprire phpstorm con il progetto desiderato, andare su File → Settings → Languages & Framework → php.ed aggiungere un nuovo client interpreter.

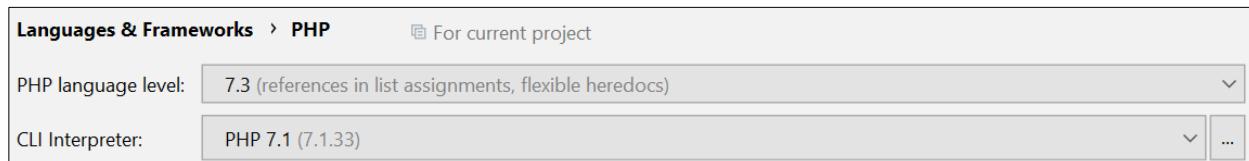


Figure 3 Aggiunta nuovo interprete

Selezionare il percorso dell'eseguibile di php e della libreria di xdebug.

Documento:	Installazione XDEBUG su XAMPP
Alunno/a:	Bryan Beffa
Classe:	Info I4AA
Anno scolastico:	2019/2020
Docente responsabile:	Ivan Raimondi

Name: Visible only for this project

General

PHP executable:   

i PHP version: 7.1.33 Debugger: Xdebug 2.6.1

i Configuration file: C:\xampp\php\php.ini [Open in Editor](#)

Additional

Debugger extension: 

Configuration options: 

These options will be passed using the "-d" command line option

Figure 4 Configurazione nuovo interprete

Andare in **File → Settings → Languages & Frameworks → PHP → Debug** e controllare che la configurazione corrisponda alla seguente.

Xdebug

Debug port: Can accept external connections

Force break at first line when no path mapping specified

Force break at first line when a script is outside the project

Figure 5 Configurazione debugger

A questo punto bisogna configurare la pagina web che si vuole debuggare. Andare quindi su **Run → Edit Configuration** ed aggiungere una nuova Php web page.

Aggiungere il server localhost e l'URL della pagina da debuggare.

Documento:	Installazione XDEBUG su XAMPP
Alunno/a:	Bryan Beffa
Classe:	Info I4AA
Anno scolastico:	2019/2020
Docente responsabile:	Ivan Raimondi

Configuration

Server: localhost

Start URL: http://localhost:8080/knowledge_base
http://localhost:8080/knowledge_base

Browser:  Default

Figure 6 Aggiunta PHP Web Page

Documento: Installazione XDEBUG su XAMPP
Alunno/a: Bryan Beffa
Classe: Info I4AA
Anno scolastico: 2019/2020
Docente responsabile: Ivan Raimondi

Test su phpstorm

Per verificare che il debugger sia stato impostato correttamente andare su **Run → Web Server Debug Validation** e premere “validate”. Se la configurazione è stata effettuata correttamente appariranno delle spunte.

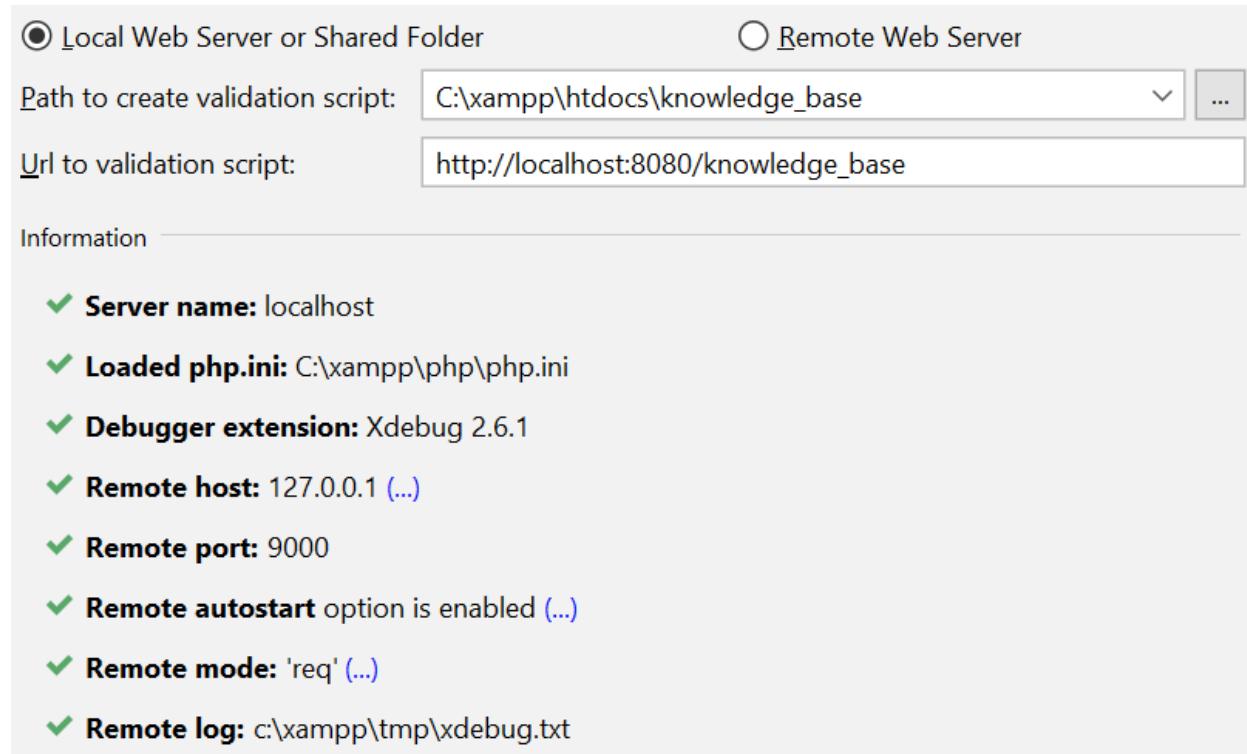


Figure 7 Controllo configurazione

Fonti:

Github: <https://gist.github.com/odan/1abe76d373a9cbb15bed> → 11.22.2019

Libreria di xdebug: <https://xdebug.org/download> → 11.22.2019

Configurazion xdebug

su phpstorm: <https://dev.to/orivolfo/installing-xdebug-on-phpstorm-for-dummies-4k8a> → 11.22.2019

Documento:	Installazione XDEBUG su XAMPP
Alunno/a:	Bryan Beffa
Classe:	Info I4AA
Anno scolastico:	2019/2020
Docente responsabile:	Ivan Raimondi

Diari di lavoro

Diario di lavoro

Luogo	Trevano
Data	2019-09-03

Lavori svolti

Oggi ci è stato affidato il primo progetto individuale che si intitola "knowledge base." Ho letto la descrizione del progetto e annotato le domande per il committente. Successivamente ho posto le domande al mio perito I. Raimondi per chiarire meglio i requisiti del progetto. Oggi ho inoltre iniziato a preparare la struttura delle cartelle del progetto, i vari template che utilizzo per la documentazione ed i diari. Abbiamo aggiunto una piccola modifica relativa agli autori: vi è amministratore e gli utenti/operatori. Durante il progetto dovrò ragionare su quali sono le funzioni di ogni tipo di utente. Inoltre ho iniziato a pianificare il diagramma Gantt del progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

-

Programma di massima per la prossima giornata di lavoro

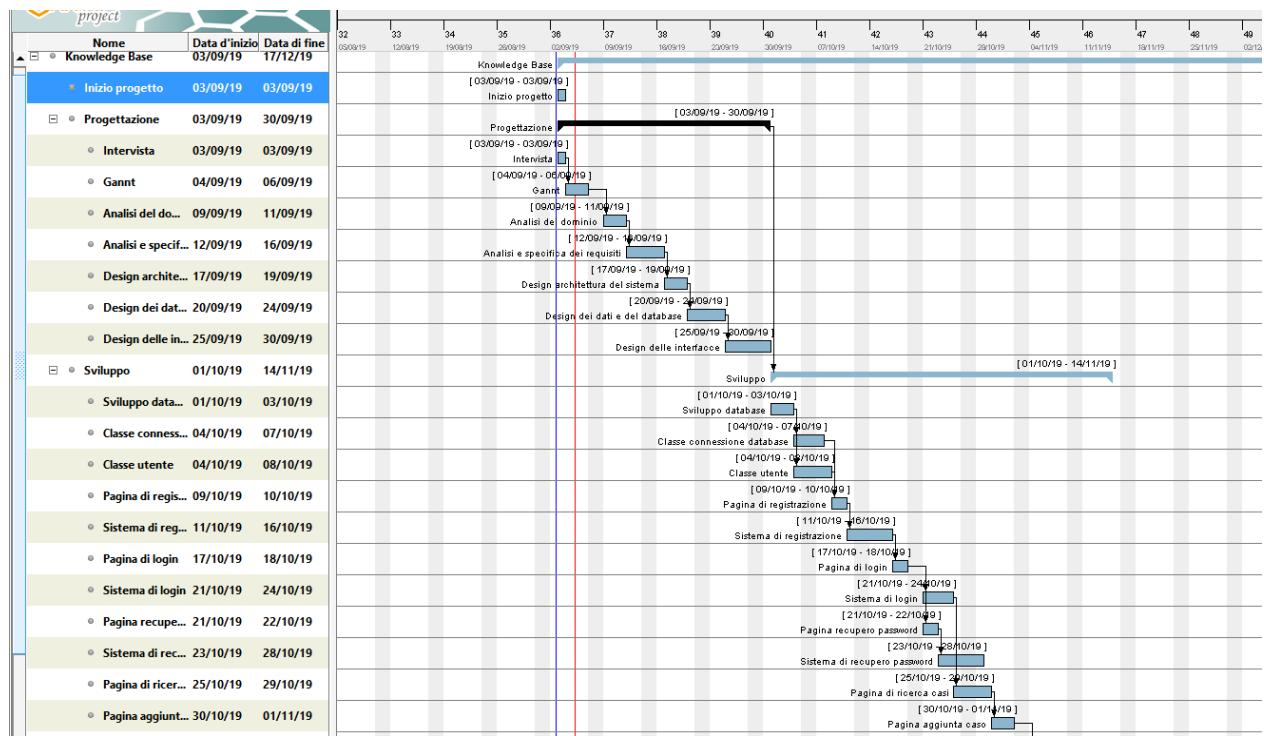
Analisi e pianificazione del progetto

Diario di lavoro

Luogo	Trevano
Data	2019-09-05

Lavori svolti

Oggi ho continuato il diagramma di Gantt che ho iniziato la lezione precedente.



Successivamente mi sono occupato del capitolo **1.4 analisi del dominio** in cui ho inserito la descrizione generale del progetto e delle sue funzioni ed ho iniziato il capitolo **1.3 scopo**.

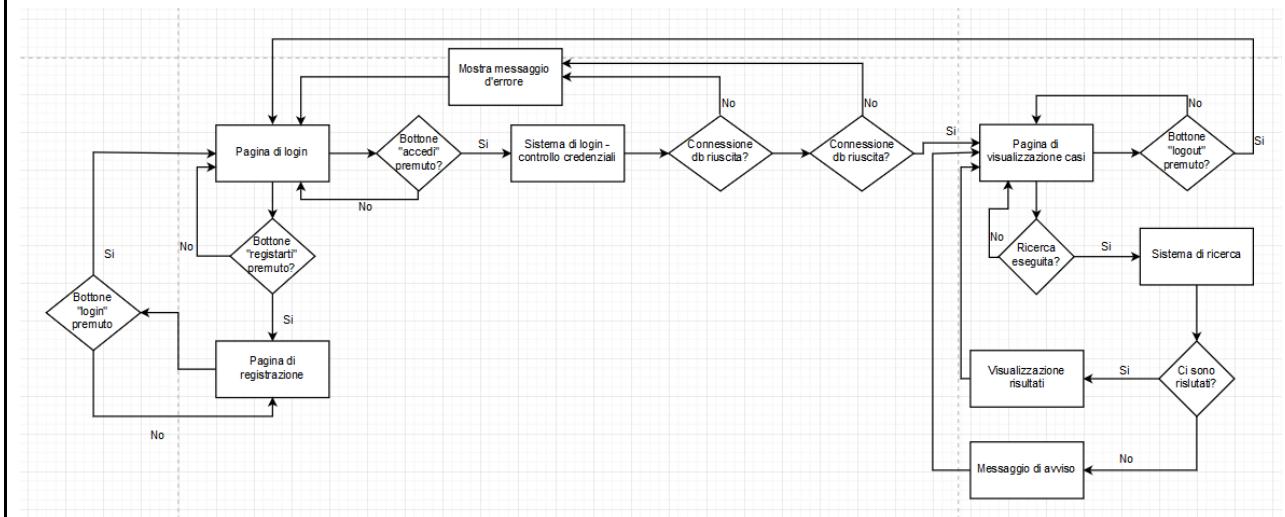
Ho iniziato il capitolo **1.5 analisi e specifica dei requisiti** inserendo le informazioni specifiche del progetto, descrivendo le funzioni degli utenti, inserendo quali operazioni si possono svolgere e come deve essere l'applicativo.

Successivamente ho iniziato la stesura delle tabelle dei requisiti:

ID: REQ-01	
Nome	Pagina di login
Priorità	1
Versione	1.0
Note	La pagina deve presentare una mascherina per il login
Sotto-requisiti	
001	Bottone "registrati" per essere indirizzati alla pagina di registrazione
002	Bottone "password dimentica" per essere indirizzati alla pagina di recupero password
003	Campo testuale per inserire l'indirizzo email
004	Campo testuale nascosto per inserire la password
005	Bottone "accedi" per effettuare il login
006	Messaggio di errore credenziali sbagliate
007	Messaggio di errore per problemi di connessione al database

Ho utilizzato il sito draw.io per iniziare a creare il diagramma procedurale del sistema.

Link: <https://www.draw.io/>



Problemi riscontrati e soluzioni adottate

Non avendo il software di Microsoft Project ho dovuto creare il Gantt tramite l'applicativo GanttProject.

Punto della situazione rispetto alla pianificazione

In anticipo rispetto la pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare il diagramma procedurale, iniziare a progettare il design dei dati e del database ed il design delle interfacce.

Applicativo web per la gestione di casi di supporto e documenti tecnici "Knowledge Base"

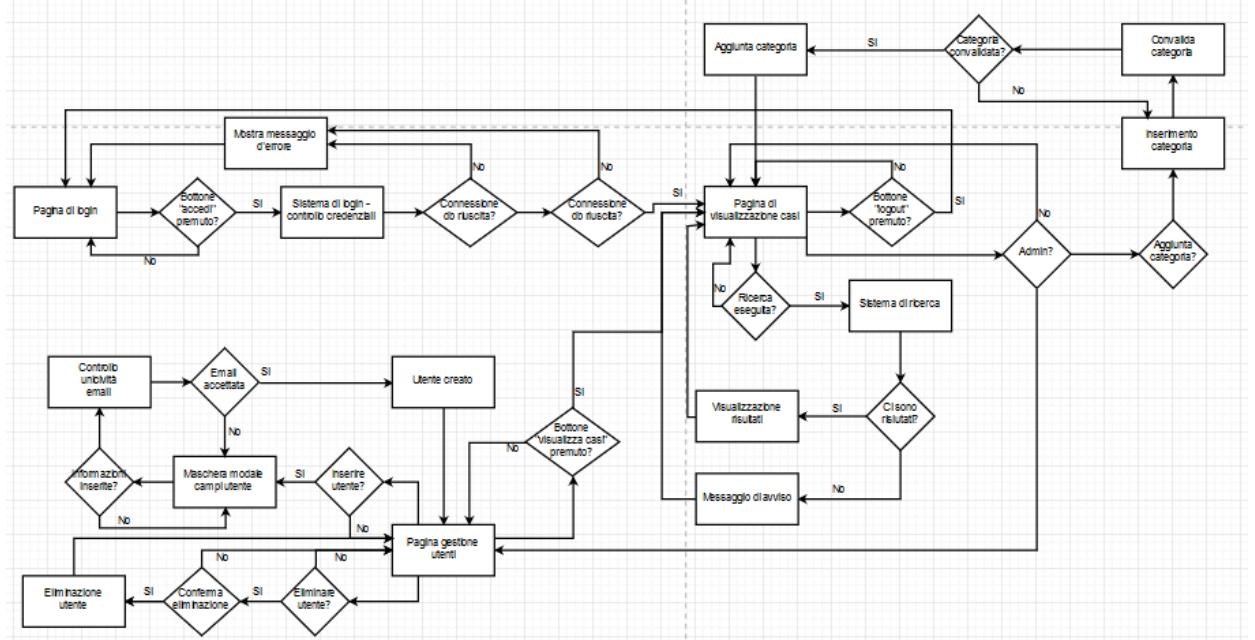
Diario di lavoro

Luogo	Trevano
Data	2019-09-06

Lavori svolti

Come prima cosa ho continuato il diagramma procedurale, ho effettuato dei cambiamenti dopo aver richiesto al docente alcune informazioni:

- Deve esserci una pagina di registrazione? No solo l'amministratore può registrare nuovi utenti.



In seguito ho rimosso un requisito dopo aver ricevuto la risposta alla stessa domanda:

ID: REQ-02	
Nome	Pagina di registrazione
Priorità	1
Versione	1.0
Note	Si necessita una pagina che contenga i campi di registrazione
Sotto requisiti	
001	Campo testuale per inserire il nome
002	Campo testuale per inserire il cognome
003	Campo testuale per inserire il proprio indirizzo email

004	Campo testuale nascosto per creare una password
005	Campo testuale nascosto per confermare la password inserita
006	Bottone “registrati” per effettuare la registrazione
007	Messaggio di errore email già utilizzata
008	Messaggio di errore per problemi di connessione al database
009	Messaggio di errore password troppo semplice/corta

Successivamente ho iniziato a progettare il database, ho utilizzato il sito dbdiagram.io per creare lo schema del database.

Link: <https://dbdiagram.io>



Ho inserito dei sottocapitoli in cui descrivo le 3 tabelle ed i loro attributi. Questo schema potrebbe essere soggetto a cambiamenti futuri.

In seguito ho iniziato il capitolo della documentazione che tratta del design delle interfacce, dove ho creato la struttura di base delle 3 pagine.

Non ho terminato in tempo la pagina di visualizzazione casi.

Problemi riscontrati e soluzioni adottate

Ho avuto problemi con l'installazione di project, non ho potuto quindi rifare il diagramma di gantt che ho terminato l'ultima lezione tramite il software GanttProject.

La prossima lezione devo rifare il gantt su Project.

Punto della situazione rispetto alla pianificazione

Sto rispettando la pianificazione del Gantt

Programma di massima per la prossima giornata di lavoro

Terminare il design delle interfacce, creare il diagramma di architettura del sistema, rifare il Gantt sul software Project.

Diario di lavoro

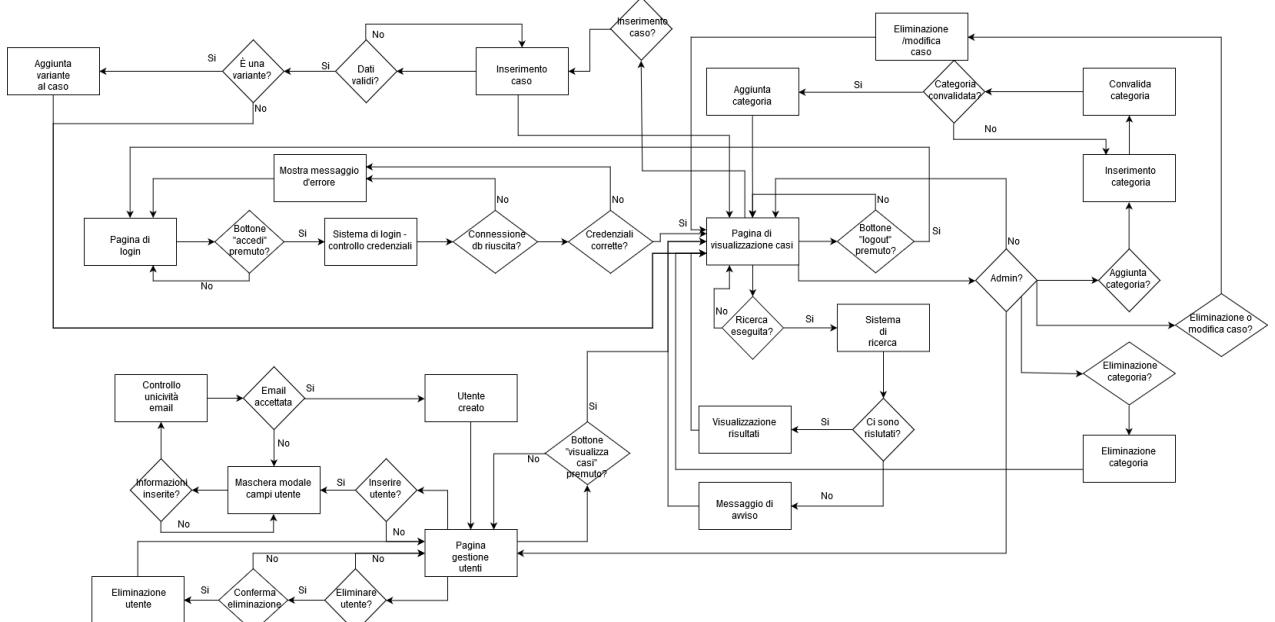
Luogo	Trevano
Data	2019-09-09
Progetto	Knowledge Base

Lavori svolti

Dopo aver chiesto al docente responsabile I. Raimondi, mi è stato riferito che non devo rifare il diagramma di Gantt usando il software Project per questo progetto. Quindi non ho ricreato il Gantt preventivo.

Come prima cosa ho terminato il capitolo **1.3 Scopo** della documentazione. Ho completato il capitolo **1.4 Analisi del dominio**.

Ho dovuto fare degli accorgimenti al diagramma procedurale, il docente mi ha fatto notare la mancanza delle varianti dei casi, l'assenza dell'eliminazione delle categorie ed alcune label sbagliate.



Successivamente ho terminato il design delle interfacce, inserendo l'interfaccia dell'admin e dell'utente.

Casi

Filtro casi

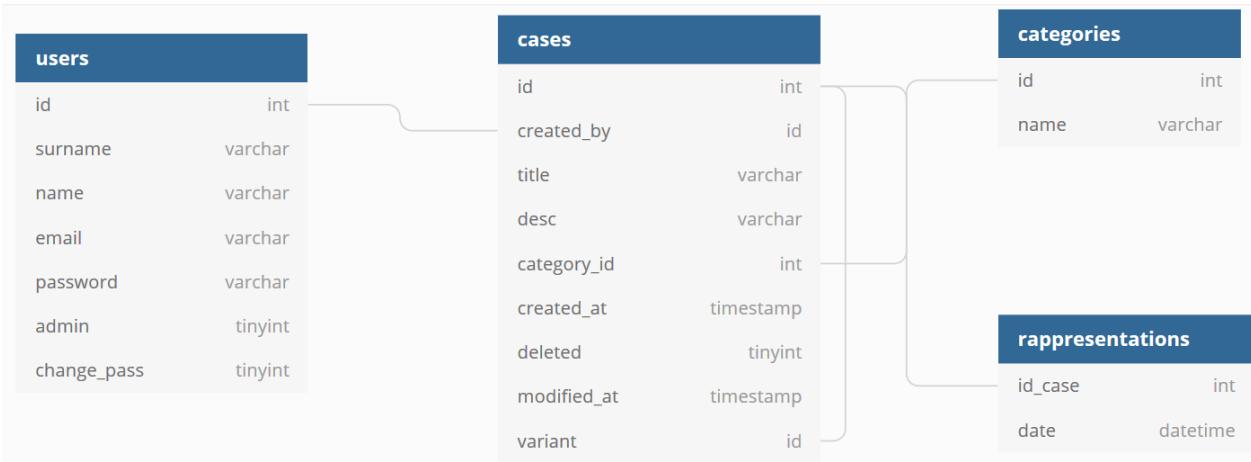
Casò: titolo	Created xx.xx.xxxx
Categoria: categoria	
Descrizione	
Times: 4	

Aggiungi caso

Esempio di interfaccia operatori.

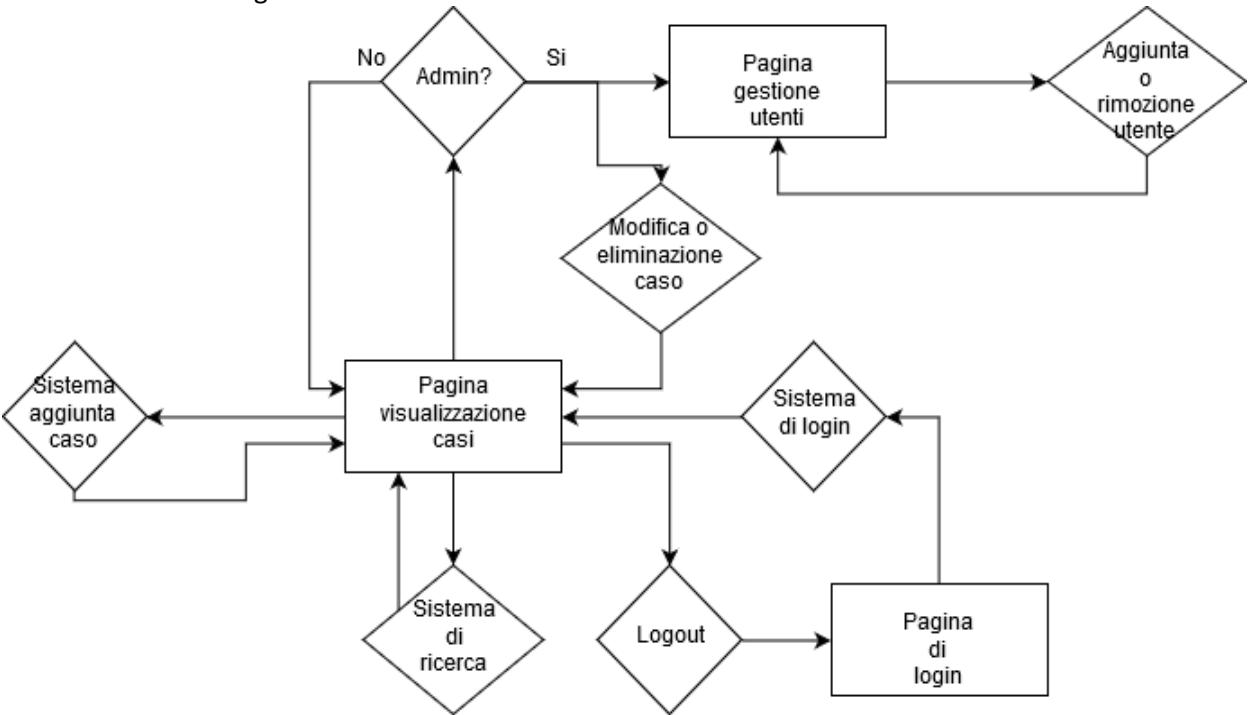
Durante la lezione ho chiesto al docente se la richiesta del cambio password tramite email fosse una buona idea. Mi è stato consigliato di lasciare come ultima cosa in modo da implementarla solo se mi rimane tempo.

Sempre dopo aver parlato con il responsabile, ho accurato che dovevo progettare in modo più approfondito il database aggiungendo la tabella per le rappresentazioni e la tabella per le varianti.



La prossima lezione consegnerò il diagramma di Gantt preventivo.

Infine ho creato il diagramma della struttura e l'ho inserito nella documentazione:



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Sto rispettando la pianificazione che ho effettuato.

Programma di massima per la prossima giornata di lavoro

Applicativo web per la gestione di casi di supporto e documenti tecnici "Knowledge Base"

La prossima lezione devo principalmente rifare il diagramma di Gantt su project anche se mi è stato detto che per questo progetto non era necessario. Lo rifaccio perché non riesco a pianificare in modo preciso le mie attività.

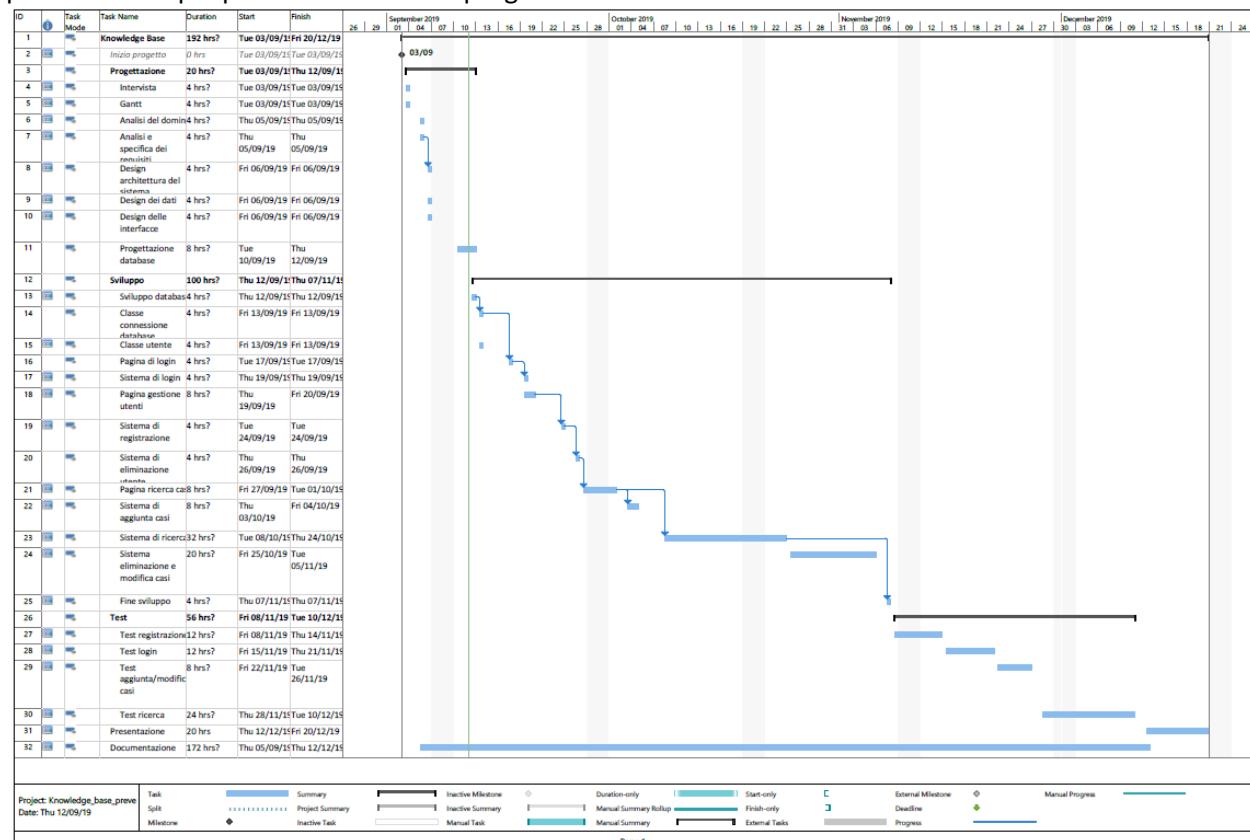
Devo creare gli usecase del progetto.

Diario di lavoro

Luogo	Trevano
Data	2019-09-12
Progetto	Knowledge Base

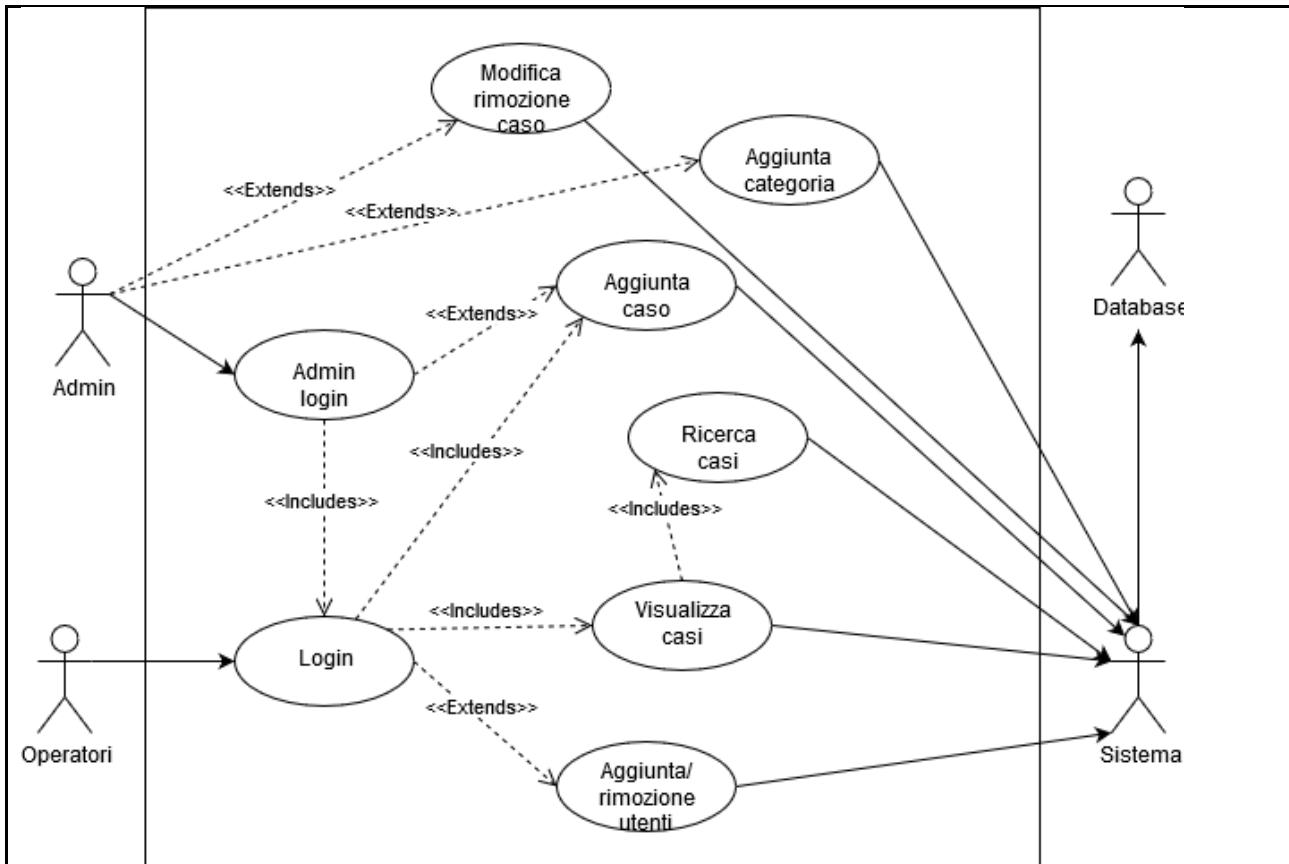
Lavori svolti

La prima parte di lezione mi sono occupato di rifare il Gantt su Microsoft Project, per avere una pianificazione più pulita e corretta del progetto:



In successione ho creato un utente che userò per la gestione del database con il quale ho sviluppato il la struttura del database: **knowledge_base_db**.

Successivamente ho creato lo usecase del progetto tramite il sito www.draw.io e lo ho inserito all'interno della documentazione (**1.6 use case**).



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Mi trovo in linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Creare la classe per la connessione effettuare la connessione al database e la classe che definisce un modello di utente.

Diario di lavoro

Luogo	Trevano
Data	2019-09-13
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho preso un template del modello mvc di php che abbiamo ricevuto l'anno scorso nel modulo 133. Ho deciso di creare l'applicativo web utilizzando questo modello per rendere la struttura più pulita e semplice, oltre al fatto che mvc è molto utilizzato.

La classe DbManager tramite il metodo connect controlla se la connessione è già avvenuta, altrimenti la crea.

```
public function connect();
```

Nella classe utente ho inserito come attributi i campi presenti all'interno della tabella users del database. Ho creato una classe di supporto **UserManager** che permette di svolgere funzioni utili per gli utenti (registrazione utenti, eliminazione, richiesta cambio password) che implementerò durante il progetto.

Problemi riscontrati e soluzioni adottate

Ho avuto dei problemi con il computer, la scheda di rete non funzionava più. Dopo aver riavviato il computer per un'ora è tornata a funzionare in modo corretto.

Punto della situazione rispetto alla pianificazione

Mi trovo in linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima volta devo creare la pagina di login e iniziare il sistema di login.

Diario di lavoro

Luogo	Trevano
Data	2019-09-17
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione mi sono occupato di creare la pagina di login utilizzando il framework material design bootstrap che ho scaricato dalla pagina web: <https://mdbootstrap.com/docs/jquery/getting-started/download/>.

Ho creato la pagina di login prendendo il codice da un esempio (link: <https://mdbootstrap.com/docs/jquery/forms/basic/>). Ovviamente ho apportato delle modifiche per renderlo più originale possibile.



Benvenuto
Accedi al nostro sistema

Name

E-mail

Ho recuperato l'icona dal sito: <https://www.iconfinder.com/search>.

Ho testato la connessione al database, utilizzando la classe che ho creato DbManager.php, stampando una semplice stringa con funziona o non funziona. L'esito è stato positivo.

Dopodiché ho iniziato a creare metodi all'interno della classe UserManager.php per il controllo delle credenziali. Viene controllato che l'indirizzo email sia o meno presente all'interno del database. Il controllo dell'email funziona in modo corretto.

Il Sistema di controllo credenziali funziona in modo corretto.

Problemi riscontrati e soluzioni adottate

Mi sono accorto che nella creazione del database ho dimenticato di mettere il campo password. Ho aggiunto quindi in seguito tramite il comando:

ALTER TABLE users ADD COLUMN password varchar(255) NOT NULL;

Punto della situazione rispetto alla pianificazione

Sono in linea con la mia pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima volta devo collegare il sistema di login con l'interfaccia grafica della pagina di login, aggiungendo i messaggi di errore.

Diario di lavoro

Luogo	Trevano
Data	2019-09-19
Progetto	Knowledge Base

Lavori svolti

Oggi come prima cosa ho corretto i campi del form (tipo inout password and email). Successivamente ho iniziato a creare la pagina di gestione utenti (accessibile solo all'admin).

Ho preso il navbar da questo sito e l'ho modificato secondo le mie necessità.
[\(https://mdbootstrap.com/docs/jquery/navigation/navbar/\)](https://mdbootstrap.com/docs/jquery/navigation/navbar/).

Successivamente ho creato il codice html e php per stampare la tabella degli utenti registrati nel database.

Knowledge Base Gestione utenti Ricerca Casi

Gestione utenti

Id	Nome	Cognome	E-mail	Richiesta cambio password	Admin	Elimina
1	bryan	surname	ciao@ciao.ciao			x

Aggiungi un [utente](#)

Successivamente ho creato la maschera modale per la registrazione di nuovi utenti:

Inserisci nuovo utente	x	Ho aggiunto la gestione dei menu in base ai privilege dell'utente.
Nome	Utente di tipo admin può accedere alla pagina di gestione utenti, mentre l'utente normale no.	
Cognome		
Email	Infine ho effettuato i controlli sulle variabili di sessione in modo da non poter accedere alle pagine saltando il login.	
Password		
Conferma password	Le variabili di sessione in cui sono salvate le credenziali vengono cancellate se viene effettuato il logout o se la sessione viene chiusa.	
<input type="button" value="CHIUDI"/> <input type="button" value="SUBMIT"/>		

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Sono in linea rispetto la pianificazione.

Programma di massima per la prossima giornata di lavoro

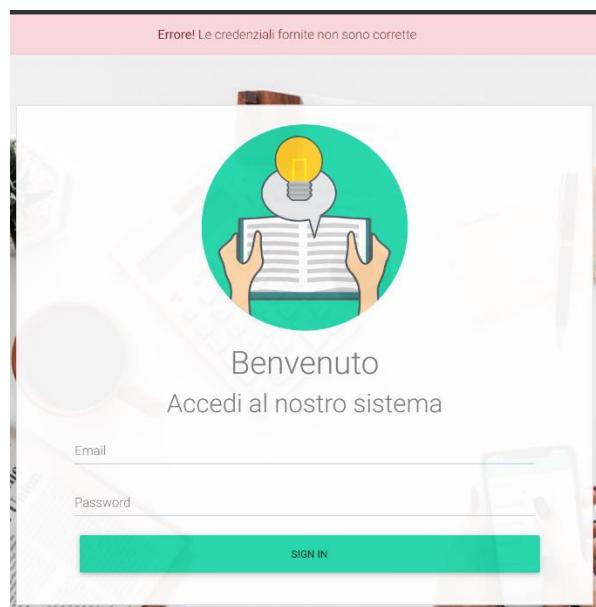
La prossima volta devo continuare la pagina di gestione utenti , inserire gli errori di login che non ho fatto questa lezione.

Diario di lavoro

Luogo	Trevano
Data	2019-09-20
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho modificato la pagina di login e completata a livello estetico. Successivamente ho aggiunto il popup di errore se vengono inserite delle credenziali non corrette. (Preso un modello da:



<https://mdbootstrap.com/docs/jquery/components/alerts/>).

Ho creato la classe PasswordManager in cui controllo che la password e la conferma password siano uguali. In questa classe potranno essere implementati i vari controlli sulla strength della password.

Inserisci nuovo utente ×

Nome

Cognome

Email

Utente base Utente admin

Password

Conferma password

CHIUDI SUBMIT

Successivamente ho iniziato il sistema di registrazione, gli utenti vengono registrati in modo corretto, ma non ho ancora creato la validazione dei campi e gestito i messaggi di errore che potrebbero presentarsi.

Al momento viene stampato il messaggio di errore se la password è diversa.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Sono in linea rispetto la pianificazione.

Programma di massima per la prossima giornata di lavoro

La prossima volta devo continuare il sistema di creazione utenti (gestioni degli errori).

Diario di lavoro

Luogo	Trevano
Data	2019-09-24
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho creato un metodo statico per settare dalla classe user manager un messaggio di errore. Questo messaggio viene settato se l'email è già stata utilizzata, o se vi è un problema a creare l'utente.

Essendo in leggero anticipo ho creato i metodi e il sistema di richiesta password. Solo un utente admin può richiedere il cambio password per un utente.

Ho deciso che gli utenti possono essere eliminati in modo definitivo senza rimanere salvati nel database, al contrario di come ho pensato (per il momento) al sistema di eliminazione dei casi, in cui vengono solo nascosti agli utenti.

Successivamente mi sono portato in avanti e ho creato il sistema di eliminazione utente, finito e funzionante.

Ho aggiunto una pagina di errore di connessione al database. Se per qualsiasi motivo il database non dovesse essere raggiungibile, appena viene effettuata un'azione che richiede l'accesso al database, appare una pagina di errore e l'utente viene disconnesso. L'utente viene disconnesso per prevenire "furti di identità" (se viene ricaricata la pagina quando il database è funzionante un qualsiasi utente potrebbe loggarsi con le credenziali dell'utente che era connesso in precedenza).



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

Per la prossima giornata di lavoro ho intenzione di iniziare la pagina di ricerca casi.

Diario di lavoro

Luogo	Trevano
Data	2019-09-26
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione mi sono occupato di aggiungere tutti i bind prima dell'esecuzione delle query.

Il resto del tempo ho iniziato a creare la pagina di ricerca casi disponendo i vari elementi per la ricerca (campi per filtrare i dati) e creando a livello grafico e strutturale un esempio di visualizzazione di un caso.

Knowledge Base Gestione utenti Ricerca Casi

Logout

Ricerca casi

Filtri Base

Scegli ordinamento

Casi più recenti

Filtri avanzati

Inserisci parola chiave

Ricerca

Selezione categoria

Tutte le categorie

Scogli una data

gg/mm/aaaa

CERCA

Risultati:

Nome: Prova

Categoria: Reti aziendali

Variante di: Caso numero 2

Data creazione: 26.09.2019 14:36:02

Ho creato una classe di supporto CategoryManager che al momento contiene il metodo che utilizzo per aggiungere le opzioni all'input select scegli categoria

Selezione categoria

Tutte le categorie

Tutte le categorie
sistematica
diagnostica
reti
permessi

CERCA

Problemi riscontrati e soluzioni adottate

Ho notato che il campo deleted nella tabella cases era di tipo tinyint(4). Il campo deve essere un boolean, quindi ho modificato con il comando `alter table cases change deleted tinyint(1) not null default 0;` l'attributo.

Punto della situazione rispetto alla pianificazione

Applicativo web per la gestione di casi di supporto e documenti tecnici "Knowledge Base"

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

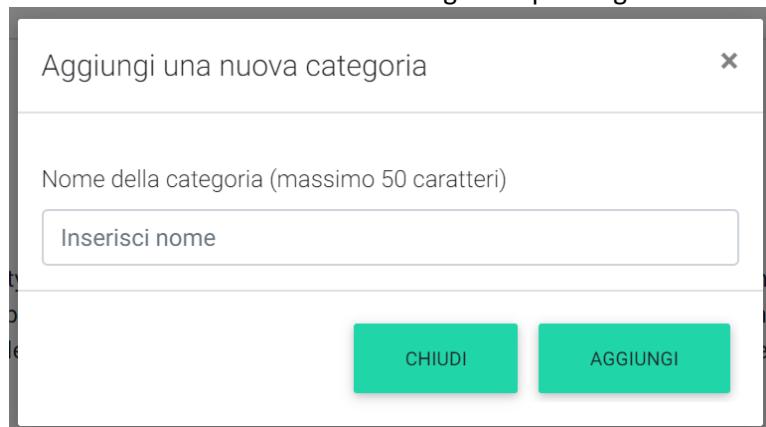
Per la prossima giornata di lavoro ho intenzione di creare l'aggiunta casi e l'aggiunta categoria (admin). Fare tutti i controlli sul input dei dati e migliorare l'interfaccia.

Diario di lavoro

Luogo	Trevano
Data	2019-09-27
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho aggiunto la creazione delle categorie da parte degli utenti admin. Se una categoria esiste già all'interno del database appare un messaggio di errore, altrimenti la categoria viene aggiunta correttamente. Se il nome della categoria è più lungo di 50 caratteri appare un messaggio di errore.



Successivamente ho avuto un'ora di colloquio con A. Togni per gli stage che abbiamo svolto durante l'estate.

Quando sono tornato in aula ho creato una classe per la gestione dei casi CaseManager.php. In questa classe per ora ho implementato il metodo che ritorna i casi presenti nel database. Questo metodo lo utilizzo per riempire le opzioni dell'input select durante la creazione di un caso nel campo variante.

Ho aggiunto per ora la maschera modale per l'aggiunta di un caso:

The image shows a modal dialog titled "Aggiungi un nuovo caso". It contains fields for "Nome della caso (massimo 50 caratteri)*" (Name of the case (maximum 50 characters)*), "Seleziona una categoria*" (Select a category*), "Variante del caso:" (Case variant:), and "Descrizione*" (Description*). At the bottom, there is a note "* indica un campo obbligatorio" and two buttons: "CHIUDI" (Close) and "AGGIUNGI" (Add).

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

Per la prossima giornata di lavoro ho intenzione di creare il sistema di aggiunta di un caso. Questo implica che devo fare i controlli sui dati e controllare i vari errori.

Diario di lavoro

Luogo	Trevano
Data	2019-10-01
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho aggiunto nel codice dove mancava il controllo che le variabili post non fossero nulle per evitare eventuali errori che potrebbero verificarsi.

Successivamente ho creato una classe validator in cui ci sono metodi di validazione utili per gli input degli utenti.

Ho creato successivamente la classe DocCase (non ho potuto creare la classe Case, l'ambiente di sviluppo non accettava il nome). Questa classe modellizza un caso di documentazione tecnica.

Ho creato il Sistema di aggiunta casi. Durante l'aggiunta casi devono essere settati dall'utente 3 input (titolo, categoria, descrizione). Inoltre si può selezionare se il caso è una variante o meno.

Aggiungi un nuovo caso

Nome della caso (massimo 50 caratteri) *

Inserisci nome

Selezione una categoria *

diagnostica

Variante del caso:

Nessun caso

Nessun caso
(id: 1) Crash della rete
(id: 4) Schermo blu
(id: 5) Schermo blu
(id: 7) Schermo blu
(id: 9) Schermata blu
(id: 11) schermata blu
(id: 16) Schermo blu
(id: 17) Schermo blu
(id: 18) Schermo blu
(id: 19) Schermo blu

* indica un campo obbligatorio

CHIUDI AGGIUNGI

Siccome l'eliminazione di un post è relativamente semplice l'ho già implementata. Quando un post viene eliminato rimane comunque salvato nel database. Semplicemente non viene più mostrato agli utenti nella pagina di ricerca.

id	created_by	title	description	category_id	created_at	deleted	modified_at	variant
1	NULL	Crash della rete	Oggi pomeriggio è avvenuto un crash della rete ...	NULL	2019-10-01 15:32:13	1	0000-00-00 00:00:00	NULL
4	1	Schermo blu	schermo bluuuu	1	2019-10-01 15:34:23	1	0000-00-00 00:00:00	1

Come ultima cosa ho aggiunto un metodo che ritorna che in base all'id del caso passato il numero di volte che è stato ripresentato il caso.

Stessa metodologia per mostare il nome della categoria.

Nome: Periferiche non connesse

ID: 22

Categoria: trouble shooting

Variante di: -

Data creazione: 2019-10-01 15:58:02

Descrizione:

dewevwvr

Numero di ripresentazioni: 2

Nome: Periferiche non connesse

ID: 23

Categoria: trouble shooting

Variante di: 22

Data creazione: 2019-10-01 15:58:51

Descrizione:

Stesso problema

Numero di ripresentazioni: 0

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

Per la prossima giornata di lavoro ho intenzione di migliorare la creazione dei post e tutta la parte sulla visualizzazione di essi e probabilmente iniziare il sistema di ricerca dei casi.

Diario di lavoro

Luogo	Trevano
Data	2019-10-03
Progetto	Knowledge Base

Lavori svolti

Oggi per prima cosa ho ristrutturato e creato dei controller per la gestione delle varie pagine. Prima avevo tutti i metodi all'interno del controller home, ora invece ho creato un controller Users dove vi sono i metodi relativi agli utenti.

Ho creto il controller researchCases in cui sono presenti i metodi relativi alla pagina ricerca_casi.php (metodi di aggiunta, filtro, eliminazione/modifica casi, aggiunta categoria).

Questa parte mi ha preso molto tempo perché alcuni metodi non erano più accessibili da classi esterne, ho dovuto quindi adeguare l'accessibilità dei metodi, renderne alcuni statici.

Successivamente ho aggiunto il primo filtro base (ordina per data i vari casi).

Ho spostato la logica della query che mostra I casi: prima mi faccio ritornare tutti i casi e controllavo quando vengono stampati se il caso è stato cancellato; mi sono accorto che basta semplicemente farmi ritornare solo i casi che non sono stati cancellati.

Ho aggiunto che quando un caso viene eliminato da un utente admin, a tutti i casi che lo avevano come riferimento alla variante, viene settato il valore null (campo variant).

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Leggermente in anticipo

Programma di massima per la prossima giornata di lavoro

Per la prossima giornata di lavoro devo mettere apposto il numero di rappresentazioni di un caso (conta anche quella dei casi eliminati). Inoltre devo iniziare a sviluppare il sistema di sviluppo con i filtri avanzati.

Diario di lavoro

Luogo	Trevano
Data	2019-10-04
Progetto	Knowledge Base

Lavori svolti

Per prima cosa ho sistemato la query che ritorna il numero di ripresentazioni: se un caso che ha come variante un altro caso viene eliminato, nel contatore del caso padre non verrà contato il caso che è stato eliminato:

```
SELECT count(*) FROM cases WHERE variant = :id and deleted = 0;
```

Successivamente ho implementato la ricerca per testo (effettua la ricerca controllando **id, titoli, descrizioni e id della variante**).

The screenshot shows a search interface for cases. At the top, there are navigation links: 'Knowledge Base' and 'Ricerca Casi' on the left, and 'Logout' on the right. Below the header, the title 'Ricerca casi' is displayed. The interface includes two main sections: 'Filtri Base:' and 'Filtri avanzati:'. Under 'Filtri Base:', there is a dropdown menu labeled 'Casi più recenti'. Under 'Filtri avanzati:', there are three input fields: 'Inserisci parola chiave' containing '30', 'Selezione categoria' containing 'Tutte le categorie', and 'Scegli una data' containing 'gg/mm/aaaa'. A green 'CERCA' button is located below these fields. Below the filters, the section 'Risultati:' is shown, with the message 'Risultati contengono nel testo il numero 30:'.

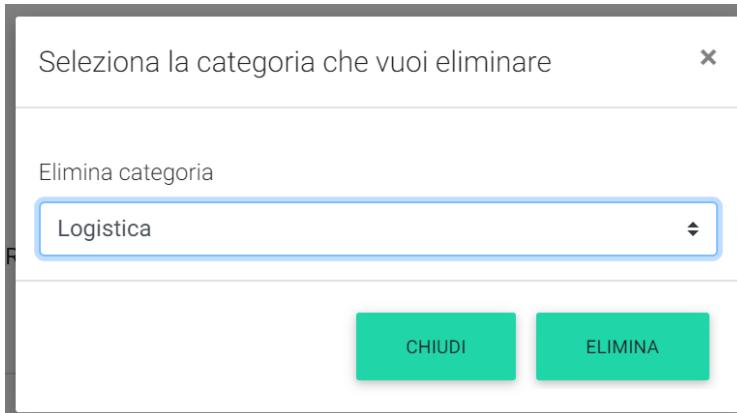
Nome: Prova
ID: 35
Categoria: diagnostica
Variante di: 30
Data creazione: 2019-10-04 14:00:44
Descrizione:
Prova numero 100
Numero di ripresentazioni: 0

Nome: 23r232r
ID: 30
Categoria: diagnostica
Variante di: -
Data creazione: 2019-10-03 16:05:36
Descrizione:
dwedwed
Numero di ripresentazioni: 1

In seguito ho aggiunto il filtro di ricerca per categoria e per data.

Ho deciso che quando avviene una ricerca utilizzando il campo testuale (nel campo id) cerca solo il valore esatto: se cerco il “3” tra gli id mi trova solo il caso 3. Cerca comunque tutti i casi che contengono il 3 tra la variante, descrizione.

Successivamente ho creato la parte di eliminazione categoria. A tutti i casi che appartengono alla categoria viene settato il valore null. In ogni caso esce un popup di avvertimento che avvisa l’utente di tipo admin che procedendo con l’eliminazione vengono settati i valori null nel campo categoria dei vari casi.



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In anticipo di due lezioni

Programma di massima per la prossima giornata di lavoro

La prossima giornata sarà incentrata sui messaggi di errori. Non vengono più stampati i messaggi di errore durante la creazione/eliminazione di utenti, categorie, casi. Questo perché ho aggiunto più controller l'ultima volta e non ho gestito i messaggi di errore.

Il filtro di ricerca della categoria deve mantenere in una sessione l'ultima categoria cercata.

Diario di lavoro

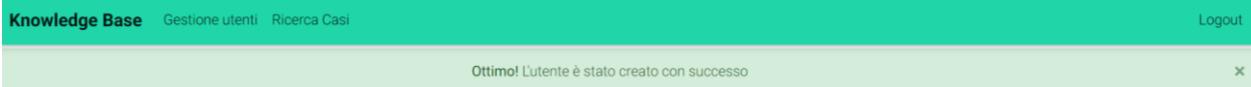
Luogo	Trevano
Data	2019-10-08
Progetto	Knowledge Base

Lavori svolti

Per prima cosa ho aggiunto alla documentazione il capitolo relativo alla pagina di errore di connessione al database.

Successivamente ho aggiunto commenti di alcuni metodi ed implementato il metodo che controlla la complessità della password (min 8 caratteri, almeno un numero, almeno una maiuscola).

Tutti gli errori che possono essere scatenati durante la fase di creazione di un utente vengono stampanti tramite un alert. Viene stampato anche se la creazione va a buon fine.



Ho messo apposito il formato della data nella visualizzazione dei vari casi (formato: dd.mm.YYYY).

I messaggi di successo ed errore durante la creazione/eliminazione di un caso o di una categoria sono stati aggiunti.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In anticipo di una lezione

Programma di massima per la prossima giornata di lavoro

La prossima giornata mi devo dedicare alla modifica dei casi (parte admin). Ogni admin può modificare casi già esistenti.

Iniziare a lavorare sulla ricerca per i casi più ricorrenti.

Il filtro di ricerca della categoria deve mantenere in una sessione l'ultima categoria cercata (non l'ho implementata oggi).

Diario di lavoro

Luogo	Trevano
Data	2019-10-10
Progetto	Knowledge Base

Lavori svolti

Mi sono dedicato alla modifica dei vari casi. Per fare ciò ho utilizzato un modal (copiato e modificato quello che ho utilizzato per la creazione di un caso).

Volendo mostrare i dati del caso durante la modifica devo richiamare il modal tramite javascript e passare i campi tramite php.

Modifica il caso: Prova modifica caso (id: 94)

Nome della caso (massimo 50 caratteri) *

Prova modifica caso

Selezione una categoria *

Robotica

Variante del caso:

(id: 93) prova

Descrizione *

Prima prova della modifica di un caso

* indica un campo obbligatorio

CHIUDI SALVA

Il resto del tempo l'ho impiegato nel provare a risolvere il problema del modal. Ci sono stati piccoli miglioramenti ma il sistema non è ancora funzionante.

Problemi riscontrati e soluzioni adottate

Ho avuto problemi nel richiamare il modal da js: le funzioni trovate online non fanno comparire il modal. Ho provato più soluzioni ma nessuna funziona come dovrebbe.

Non ho risolto il problema. Per ora si apre il modal ma da dei problemi: sfondo nero e cambia la dimensione degli elementi dietro (toolbar, sfondo nero). Inoltre non stampa i dati corretti all'interno della modifica.

The screenshot shows the 'Ricerca casi' (Search cases) page of the Knowledge Base application. At the top, there is a navigation bar with links for 'Knowledge Base', 'Gestione utenti', 'Ricerca Casi', and 'Logout'. A red circle highlights the 'Logout' link. Below the navigation, the page title 'Ricerca casi' is displayed. There are two sections for filtering: 'Filtri Base:' and 'Filtri avanzati:'. Under 'Filtri Base:', there is a dropdown menu set to 'Casi più recenti' and a button labeled 'CERCA'. Under 'Filtri avanzati:', there are three input fields: 'Inserisci parola chiave' (with a placeholder 'Ricerca'), 'Selezione categoria' (set to 'Tutte le categorie'), and 'Scegli una data' (placeholder 'gg/mm/aaaa'). A green button labeled 'CERCA' is located below these filters. The results section shows a single result card with the following details:

Nome: random	
ID:	91
Categoria:	Sicurezza
Variante di:	-

Punto della situazione rispetto alla pianificazione

In anticipo di una lezione.

Programma di massima per la prossima giornata di lavoro

La prossima giornata devo continuare il sistema di modifica dei casi, cercare di mettere apposto il problema con la pagina modale.

Iniziare a lavorare sulla ricerca per i casi più ricorrenti.

Il filtro di ricerca della categoria deve mantenere in una sessione l'ultima categoria cercata (non l'ho implementata oggi).

Diario di lavoro

Luogo	Trevano
Data	2019-10-11
Progetto	Knowledge Base

Lavori svolti

Le prime due ore di lezione il docente Valsangiacomo ci ha spiegato i criteri di valutazione di del LP.

La prima cosa che ho fatto sul progetto è passare tramite php ad un metodo js i dati da inserire all'interno del form di modifica di un caso i dati del caso scelto.

```
onclick="showModifyModal('<?php echo $case['title'] ?>', '<?php echo $case['id'] ?>', '<?php echo $case['description'] ?>', '<?php echo $case['category_id'] ?>', '<?php echo $case['variant'] ?>')"
```

Per il momento il come appare a livello grafico il modal lo lascio in sospeso e mi occupo che funzioni a livello di codice (che modifichi in modo corretto e salvi le modifiche).

Problemi riscontrati e soluzioni adottate

Il sistema di modifica non è ancora funzionante, devo trovare un modo per salvare l'id del caso da modificare e passarlo alla funzione php.

Punto della situazione rispetto alla pianificazione

In anticipo di una lezione.

Programma di massima per la prossima giornata di lavoro

Devo continuare il sistema di modifica dei casi. Devo cercare di risolvere i problemi che sto riscontrando.

Queste due cose le farò come prima cosa in modo da non rimanere troppo indietro per colpa della modifica dei casi:

- Iniziare a lavorare sulla ricerca per i casi più ricorrenti.
- Il filtro di ricerca della categoria deve mantenere in una sessione l'ultima categoria cercata (non l'ho implementata oggi).

Diario di lavoro

Luogo	Trevano
Data	2019-10-15
Progetto	Knowledge Base

Lavori svolti

Dopo aver parlato con il mio compagno Filippo Finke ho scoperto che esiste l'input di tipo hidden, ovvero che non viene mostrato all'utente e viene utilizzato apposta in questi casi.

```
<input type="hidden" id="modifyCaseId" name="modify_case_id">
```

Tramite javascript vado a riempire tutti i campi passando come parametro:

```
function showModifyModal(caseName, caseId, caseDescription, categoryId, variantId) {
```

In questa funzione js vado ad inserire negli input del form i valori precedenti alla modifica.

The screenshot shows a modal dialog box with the following details:

- Title:** Modifica caso: Schermo blu (id: 96)
- Nome della caso:** Schermo blu
- Selezione una categoria:** Sicurezza
- Variante del caso:** Nessun caso
- Descrizione:** Casoooo numero 1
- Note:** * indica un campo obbligatorio
- Buttons:** CHIUDI (Close) and SALVA (Save)

Dopo ciò ho dovuto fare in modo che quando vado a scegliere una variante del caso non appaia come opzione del select il caso che si sta modificando.

Questa parte l'ho fatta tramite javascript all'interno dello stesso metodo per inserire i valori nel form. Mostro tutte le options e successivamente nascondo quella che corrisponde al caso che si sta modificando.

In questo modo l'utente non può selezionare come variante di un caso il caso stesso.

```
var select = document.getElementById("modifyCaseVariantSelect");

//show all options
var options = select.options;
for (var i = 0; i < options.length; i++) {
    options[i].hidden = false;
}

//set hidden variant
var hiddenVariant = "variant" + caseId;

//hide current case as variant
document.getElementById(hiddenVariant).hidden = true;
```

La parte di modifica ora funziona in modo corretto.

Nel filtro di ricerca della categoria viene mantiene l'ultima categoria cercata.

```
<option value="=php echo $category['id']?> <?php echo
((isset($_SESSION['category_filter'])) && $_SESSION['category_filter'] ==
$category['id']) ? "selected>" : ">") . $category['name'] . "</option>" ?>
```

In mysql ho dovuto modificare il campo modified_at in modo che quando si modifica un caso in automatico setta la data di modifica:

Settare timestamp automaticamente quando avviene una modifica

```
ALTER TABLE cases CHANGE `modified_at` `modified_at` TIMESTAMP
NULL ON UPDATE CURRENT_TIMESTAMP;
```

Settare solo all'inserimento del record il campo created_at:

```
ALTER TABLE cases CHANGE `created_at` `created_at` TIMESTAMP NOT NULL DEFAULT now();
```

Fonti:

<https://stackoverflow.com/questions/32703553/timestamp-in-mysql-column>

Mi sono accorto di aver sbagliato ad inserire il nome della tabella representation (dovrebbe essere representation), così ho messo apposto la doc.

Inoltre nella tabella representation manca il campo relativo alla variante quindi ho dovuto aggiungerlo. In questa tabella viene inserito quando un caso viene creato/modificato, il timestamp e la variante che è stata settata.

Se un caso viene creato/modificato ma non è una variante di un altro caso nella tabella non viene inserito niente.

Questa tabella è pensata per un'implementazione futura in cui gli utenti di tipo admin possano vedere quando sono stati aggiunti e creati casi che possiedono una variante.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In anticipo di 2 lezioni

Programma di massima per la prossima giornata di lavoro

Implementare la ricerca per i casi più ricorrenti.

Pensavo di implementare la ricerca per i casi meno recenti (dato che ho implementato la ricerca dei casi più recenti).

La prossima lezione mi focalizzerò maggiormente sulla documentazione.

Diario di lavoro

Luogo	Trevano
Data	2019-10-17
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho risolto il problema spiegato nella tabella dei problemi riscontrati.

Dopo ciò ho dovuto aggiungere la query che va ad inserire i dati all'interno della tabella representation utilizzando l'ultimo id inserito.

Successivamente ho aggiunto al filtro dell'ordine dei post la possibilità di ordinare i risultati dal meno recente al più recente.

Problemi riscontrati e soluzioni adottate

Provando ad aprire e chiudere le varie finestre modali ho notato che il contenuto del body veniva spostato verso sinistra:

La soluzione mi ha preso abbastanza tempo ma sono riuscito a trovarla:
cercando su internet ho trovato il modo di risolvere, ovvero ho dovuto aggiungere due classi css in cui specifico i seguenti attributi:

```
body.modal-open-noscroll {  
    margin-right: 0 !important;  
    padding-right: 0 !important;  
    overflow: hidden;  
}
```

Il problema era dato dal fatto che veniva aggiunta ogni volta al tag body lo spazio per la scrollbar del modal.

Fonte: <https://stackoverflow.com/questions/19960162/bootstrap-3-modal-fires-and-causes-page-to-shift-to-the-left-momentarily-brows>

Provando a modificare alcuni casi sono imbattuto in un errore: se provo a modificare un caso che contiene un apostrofo (nel nome, o nella descrizione) non viene aperta la finestra modale per permettere la modifica.

Questo perché quando passo gli argomenti tramite php contengono il carattere apostrofo il metodo interpreta come finito il parametro inserito e quindi vi è un errore di formattazione degli argomenti passati

Errore mostrato dalla console di firefox:

Uncaught SyntaxError: Invalid or unexpected token

Sintassi interpretata con l'apostrofo di troppo:

```
onclick="showModifyModal('Schermo bludmowefnm' , '96', 'Problema con l'accesso alla rete  
scolastica' , '24', '')"
```

Al momento non so come evitare questo problema

Punto della situazione rispetto alla pianificazione

In anticipo di 1 lezioni

Programma di massima per la prossima giornata di lavoro

Oggi non ho avuto tempo di implementare la ricerca per i casi più ricorrenti, quindi probabilmente la prossima lezione mi concentrerò su questa parte.

La prossima lezione mi focalizzerò maggiormente sulla documentazione e cercare di mettere apposto il problema che ho riscontrato (veda capitolo problemi)

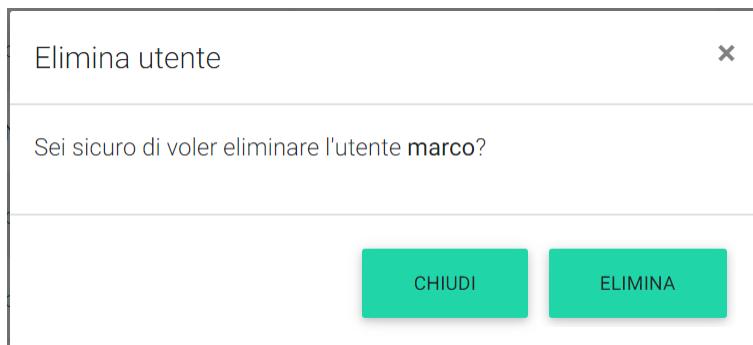
Diario di lavoro

Luogo	Trevano
Data	2019-10-18
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho aggiunto un bottone che rende visibile o nasconde la maschera di ricerca. Questo parte l'ho fatta utilizzando le classi che mette a disposizione mdbootstrap.

Successivamente ho aggiunto, dove è richiesta, la conferma delle azioni tramite una finestra modale. L'ho implementato nella pagina di eliminazione degli utenti per il momento.



Successivamente ho aggiunto un metodo che ritorna i nomi e gli id di tutti i casi anche quando ci è un filtro attivo. In questo modo se si modifica un caso mentre il filtro è attivo si può comunque selezionare come variante un caso che non appare nei risultati.

Il resto del tempo mi sono dedicato alla documentazione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Oggi non ho avuto tempo di implementare la ricerca per i casi più ricorrenti, quindi probabilmente la prossima lezione mi concentrerò su questa parte.

Continuare la documentazione e trovare una soluzione per la parte di modifica casi (problema carattere apostrofo).

Diario di lavoro

Luogo	Trevano
Data	2019-10-22
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione ho visto con il docente Raimondi la query per ottenere tutti i dati dei casi più ricorrenti, dopo un po' di prove siamo giunti alla conclusione che l'unico modo per farsi che la query ritornasse i risultati desiderati fosse effettuare una inner join sulla stessa tabella.

```
SELECT
    c.*
FROM
    cases c
INNER JOIN cases a ON
    c.id = a.variant group by a.variant order by count(a.variant)
desc;
```

Successivamente mi sono dedicato all'implementazione della query all'interno dell'applicativo. Le prime due ore ho finito questa parte, ora il filtro di ricerca è finito e funzionante.

La seconda parte di lezione mi sono dedicato alla documentazione. Ho continuato la parte di implementazione, ho finito la descrizione delle pagine e le loro funzionalità.

Infine ho iniziato l'aggiunta della finestra modale di conferma di eliminazione di un caso.

Problemi riscontrati e soluzioni adottate

Applicativo web per la gestione di casi di supporto e documenti tecnici "Knowledge Base"

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare l'aggiunta della conferma di eliminazione caso.

Continuare la documentazione e trovare una soluzione per la parte di modifica casi (problema carattere apostrofo).

Diario di lavoro

Luogo	Trevano
Data	2019-10-24
Progetto	Knowledge Base

Lavori svolti

Come prima cosa oggi ho terminato l'implementazione della finestra modale di conferma di eliminazione di un caso.

Successivamente ho aggiunto la finestra modale anche per la modifica dati, chiede all'utente se è sicuro di salvare i dati inseriti.



Ad un certo punto non andava più la finestra modale di eliminazione utenti. L'errore era che all'interno del tag html <a> ho rimosso per sbaglio l'attributo **data-toggle="modal"**.

Mi sono informato e ho scoperto che le email possono essere lunghe al massimo 320 caratteri, così ho dovuto cambiare il campo email varchar(256) in varchar(320):

```
alter table users change email email varchar(320);
```

Ho aggiunto un controllo nel metodo per l'eliminazione degli utenti:
l'utente che è loggato non può eliminare il suo account.

Successivamente ho aggiunto controlli nella creazione e la modifica dei casi. Se una persona va a modificare da ispeziona elemento il valore dei delle varianti il sistema controlla che il valore sia accettabile (sia presente nel db e il campo deleted non sia a 1).

Infine mi sono dedicato un po' alla documentazione dove ho iniziato i capitolo sui controllers (cap 3).

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Come prima cosa devo fare i controlli sulla categoria selezionata durante la modifica/creazione di un caso.

Controllare la sicurezza di tutti i form e mettere fare aggiustamenti dove serve.

Continuare la documentazione e trovare una soluzione per la parte di modifica casi (problema carattere apostrofo).

Diario di lavoro

Luogo	Trevano
Data	2019-10-25
Progetto	Knowledge Base

Lavori svolti

Come prima cosa oggi ho inserito il controllo sulla categoria inserita dall'utente durante la creazione/modifica di un caso. Se da ispeziona elemento inserisce un valore che non è valido la pagina ritorna un messaggio di errore.

Errore! La categoria inserita non è valida

Per il momento penso che tutti i controlli sui campi inseriti dall'utente siano fatti, magari nelle prossime lezioni, durante i test mi accorgerò che ne mancano alcuni.

Tramite il sito mockaroo (<https://mockaroo.com/>) ho inserito all'interno della tabella dei casi 2000 records, i tempi per effettuare le query non sono cambiati, i risultati vengono stampati molto velocemente.

Ho risolto il problema degli apostrofi che ho riscontrato le scorse lezioni. Per risolvere il problema ho utilizzato la seguente soluzione:

```
onclick="showModifyModal('<?php echo str_replace("'", "\\'", $case['title']) ?>' ,  
'<?php echo $case['id'] ?>', '<?php echo str_replace("'", "\\'",  
$case['description']) ?>', '<?php echo $case['category_id'] ?>', '<?php echo  
$case['variant'] ?>')
```

Ho aggiunto il metodo **str_replace** in cui passo il parametro ' all'interno della stringa del titolo o della descrizione e la sostituisco con \'. In questo modo il metodo non interpreta gli apostrofi come chiusura dei parametri.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione ho intenzione di dividere i casi su più pagine in modo da rendere la pagina meno carica e più veloce.

Continuare la documentazione.

Diario di lavoro

Luogo	Trevano
Data	2019-11-05
Progetto	Knowledge Base

Lavori svolti

Come prima cosa oggi ho fatto alcune ricerche su come dividere i risultati in più pagine, così da non rendere la navigazione lenta.

Ho interrotto la ricerca della impaginazione per parlare con il docente Raimondi che mi ha consigliato di trovare una soluzione più pulita per il problema relativo ai caratteri speciali dalla lettura dei dati dal db.

Non avendo trovato una soluzione per il momento ho lasciato il codice funzionante, dopo aver cercato online ho trovato che il metodo prepare di pdo svolge le stesse funzioni di mysqli_real_escape_string(). Quindi la scrittura nel database avviene in modo corretto.

Ho mantenuto il metodo str_replace per modificare il carattere ‘ in \' in modo da poter passare i parametri testuali a javascript senza problemi.

Il resto della lezione ho dedicato il mio tempo a modificare le interfacce grafiche.

Gestione utenti Ricerca Casi

Logout

Ricerca casi

MOSTRA FILTRI

Aggiungi caso
Aggiungi una categoria
Elimina una categoria

Risultati:

Nome: ciao'

ID: 2135
Categoria: trouble shooting
Variante di: -
Data creazione: 05.11.2019 13:21:21
Descrizione:
ciao come va
Numero di ripresentazioni: 0

MODIFICA

ELIMINA

Ho aggiunto una DataTable (mdbbootstrap mette a disposizione questa possibilità) in cui viene effettuata un'impaginazione automatica. In questo modo l'utente (admin) può scegliere quanti utenti visualizzare, effettuare una ricerca tra gli utenti.

Mostra 10 record per pagina						Cerca: mar
ID	Nome	Cognome	E-mail	Admin	Elimina utente	
40	marco	bianchi	wedewd@cic.cee	No	Elimina	
45	Mario	Rossi	mario@rossi.com	No	Elimina	
49	Mario	Rossi	mario.rossi@hotmail.com	No	Elimina	

Pagina 1 di 1 (filtrato da 12 record totali) Precedente 1 Successiva

Link: <https://datatables.net/manual/options>

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

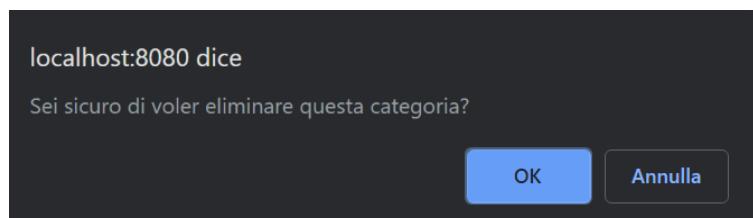
In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione ho intenzione di dividere i casi su più pagine in modo da rendere la pagina meno carica e più veloce (vedere se e come è possibile implementare le DataTables su un layout non tabulare, nel mio caso delle card views).

Continuare la documentazione.

Aggiungere l'ultima finestra modale di conferma, devo implementarla per la conferma di eliminazione di una categoria. Al momento la conferma è fittizia con il popup di default:



Dovrà essere simile alla conferma di eliminazione di un utente:



Diario di lavoro

Luogo	Trevano
Data	2019-11-07
Progetto	Knowledge Base

Lavori svolti

Oggi non ero presente a lezione causa malattia.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

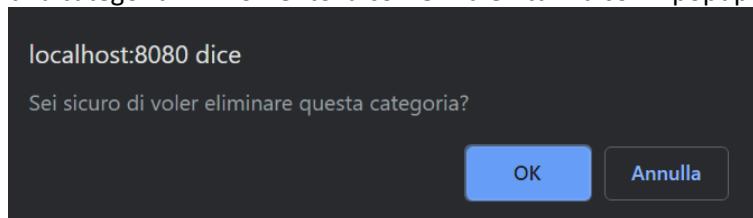
In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione ho intenzione di dividere i casi su più pagine in modo da rendere la pagina meno carica e più veloce (vedere se e come è possibile implementare le DataTables su un layout non tabulare, nel mio caso delle card views).

Continuare la documentazione.

Aggiungere l'ultima finestra modale di conferma, devo implementarla per la conferma di eliminazione di una categoria. Al momento la conferma è fittizia con il popup di default:



Dovrà essere simile alla conferma di eliminazione di un utente:



Diario di lavoro

Luogo	Trevano
Data	2019-11-08
Progetto	Knowledge Base

Lavori svolti

Oggi ho inserito una tabella nella pagina di ricerca casi per utilizzare le dataTables anche sui casi. In questo modo oltre ad avere l'impaginazione, l'utente può scegliere quanti casi visualizzare, effettuare una ricerca testuale in live.

Mostra record per pagina

Cerca in live:

Nome: Egret, cattle

ID: 7043

Categoria: trouble shooting

Variante di: -

Data creazione: 08.11.2019 12:48:26

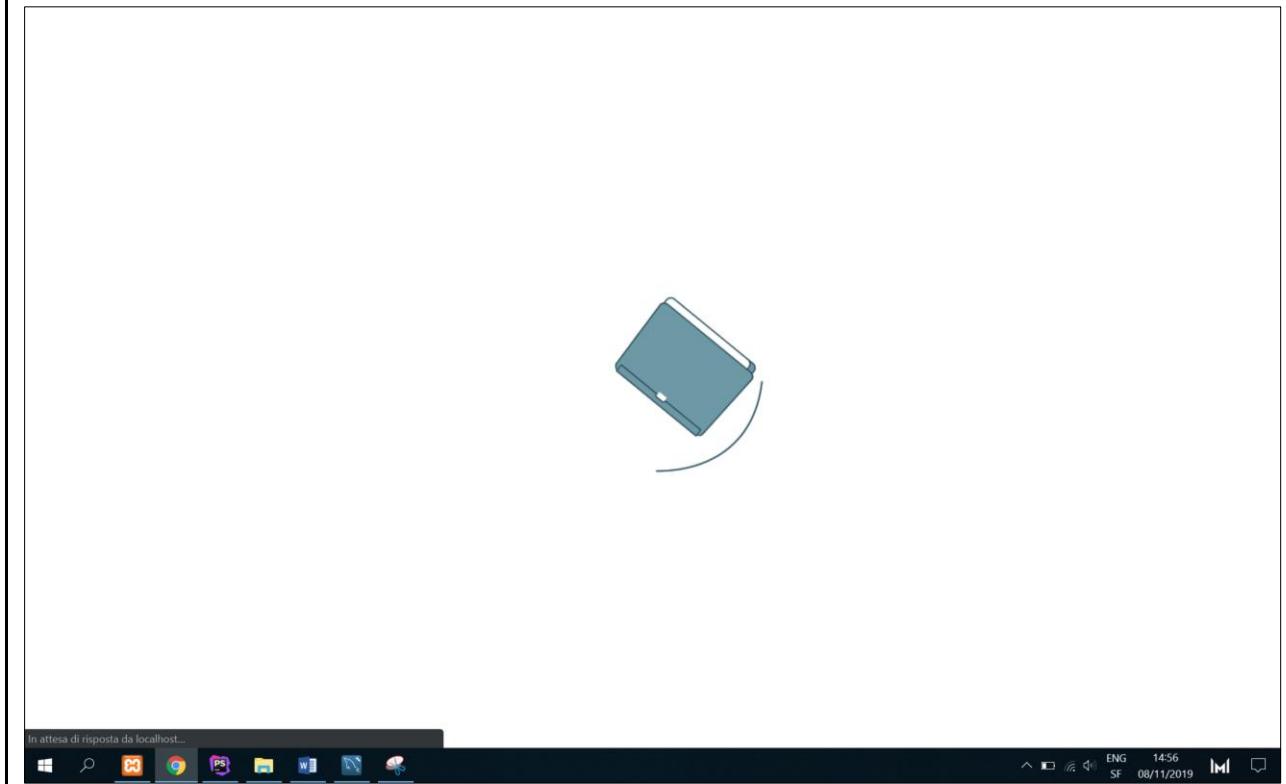
Descrizione:

Impaginazione:

Pagina 1 di 100

Precedente 2 3 4 5 ... 100 Successiva

Al primo caricamento, se ci sono molti casi, la pagina ci mette un po' a caricare. Per questo motivo ho aggiunto un'animazione nell'attesa che la pagina sia completamente carica.



Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Aggiungere l'ultima finestra modale di conferma, devo implementarla per la conferma di eliminazione di una categoria. Al momento la conferma è fittizia con il popup di default:

localhost:8080 dice

Sei sicuro di voler eliminare questa categoria?

OK

Annulla

Dovrà essere simile alla conferma di eliminazione di un utente:



Diario di lavoro

Luogo	Trevano
Data	2019-11-12
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho implementato il popup di conferma di eliminazione di una categoria che mi mancava. Tutte le finestre modali sono state implementate.



Successivamente mi è stato richiesto dal docente Raimondi di creare una scaletta per un eventuale demo, in cui descrivo i vari passaggi.

Ho cambiato la pagina di login:



Per il resto del tempo mi sono dedicato alla documentazione, in particolare il capitolo **implementazione** ed iniziato a documentare i test-case.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione devo continuare a documentare i test che ho effettuato ed il loro esito.

Diario di lavoro

Luogo	Trevano
Data	2019-11-14
Progetto	Knowledge Base

Lavori svolti

Oggi mi sono dedicato interamente alla documentazione, in particolare ho finito le tabelle dei test case.
Una volta terminati i test case ho continuato il capitolo implementazione.

Mi sono accorto che ho sbagliato a scrivere i requisiti nella documentazione. Le chiedo se posso aggiustarli o devo scrivere da qualche parte le modifiche.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione devo continuare la parte di implementazione della documentazione. Finire la parte dei controller ed iniziare le classi del models.

Diario di lavoro

Luogo	Trevano
Data	2019-11-15
Progetto	Knowledge Base

Lavori svolti

Oggi mi sono dedicato interamente alla documentazione. Ho continuato la parte di implementazione inserendo le classi della cartella models (ne mancano 2).

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione devo continuare la parte di implementazione della documentazione. Sistemare i requisiti che sono stati scritti male.

Diario di lavoro

Luogo	Trevano
Data	2019-11-19
Progetto	Knowledge Base

Lavori svolti

Come prima cosa oggi ho creato una classe MessageManager.php all'interno della cartella models in cui ho spostato in metodi del controller home relativi ai messaggi (printErrorMsg(),). Il mio compagno Filippo Finke mi aveva fatto notare che l'utente poteva chiamare questi metodi dall'url. In questo modo gli utenti non possono più chiamare i metodi essendo stati spostati in un'altra classe.

Ho dovuto riadattare la doc per questa modifica.

Problemi riscontrati e soluzioni adottate

La tabella degli utenti del database mysql si è corrotta. Ho dovuto installare di nuovo mysql. Ho passato 3 ore di lezione per ripristinare il database con i suoi dati e mysql.

Ricreando l'ambiente mi dà problemi con il cambio di pagina che non viene più effettuato. Devo trovare come sistemare questo problema.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Sistemare il problema che ho riscontrato.

Diario di lavoro

Luogo	Trevano
Data	2019-11-21
Progetto	Knowledge Base

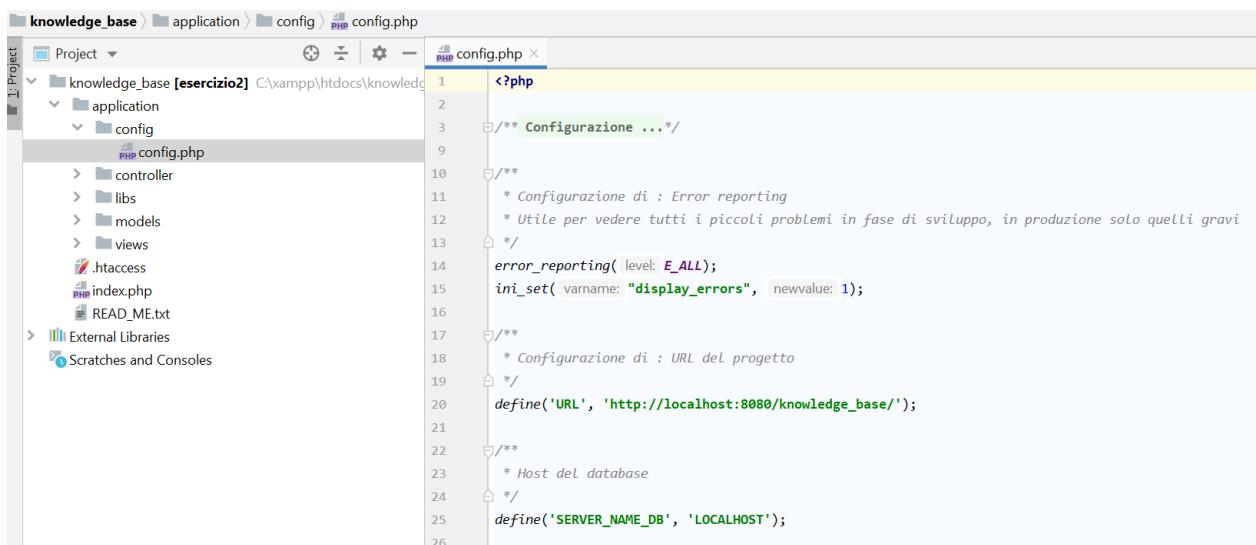
Lavori svolti

Come prima cosa ho risolto il problema che avevo con il cambio di pagina. Successivamente ho effettuato gli ultimi test che mancavano e ho documentato il loro risultato.

Una volta fatto ciò ho continuato la parte della documentazione di implementazione, inserendo la classe CaseManager.php. Verso la fine della lezione ho iniziato a documentare la classe UserManager.php.

Problemi riscontrati e soluzioni adottate

Ho risolto il problema del cambio pagina. Avevo sbagliato il percorso del file di configurazione di MVC.



```

knowledge_base > application > config > config.php
Project knowledge_base [esercizio2] C:\xampp\htdocs\knowledge_base
  <--> application
    <--> config
      config.php
        <--> controller
        <--> libs
        <--> models
        <--> views
        .htaccess
        index.php
        READ_ME.txt
  <--> External Libraries
  Scratches and Consoles
  
```

```

<?php
/*
 * Configurazione ...
 */
/**
 * Configurazione di : Error reporting
 * Utile per vedere tutti i piccoli problemi in fase di sviluppo, in produzione solo quelli gravi
 */
error_reporting( level: E_ALL );
ini_set( varname: "display_errors", newvalue: 1 );

/*
 * Configurazione di : URL del progetto
 */
define('URL', 'http://localhost:8080/knowledge_base/');

/*
 * Host del database
 */
define('SERVER_NAME_DB', 'LOCALHOST');

```

Problema prima:

```
define('URL', 'localhost:8080/knowledge base/');
```

Soluzione:

```
define('URL', 'http://localhost:8080/knowledge base/');
```

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare la documentazione e terminare la parte di implementazione.

Diario di lavoro

Luogo	Trevano
Data	2019-11-22
Progetto	Knowledge Base

Lavori svolti

Come prima cosa ho terminato il capitolo dell'implementazione commentando la classe Usermanager.php che mi mancava.

Ho creato la guida di installazione di xdebug e configurazione su phpstorm (ide).

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare la documentazione ed iniziare i capitoli finali. Inoltre devo completare la guida d'installazione del debugger e creare la guida d'installazione del sito.

Diario di lavoro

Luogo	Trevano
Data	2019-11-26
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione ho mostrato la demo al docente Raimondi durante la quale mi ha posto delle domande e dato dei consigli su cosa potrei implementare o modificare.

Ho provato a cambiare il formato dell'input per il filtro della data. Dopo aver cercato online per un po' di tempo ho letto che ogni browser, in base alla lingua impostata, presenta il suo formato di data. Così ho dovuto scaricare delle librerie per il time picker in modo da scaricare anche il formato corretto.

Ho dovuto quindi settare le seguenti proprietà:

```
$('.datepicker').datepicker({  
    format: "dd.mm.yyyy",  
    weekStart: 1,  
    todayBtn: "linked",  
    clearBtn: true,  
    language: "it",  
    todayHighlight: true,  
    autoclose: true,  
})
```

Ho dovuto convertire la data che mi dava l'input nel formato corretto in modo che la query funzionasse senza

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare la documentazione ed iniziare i capitoli finali. Inoltre devo completare la guida d'installazione del debugger e creare la guida d'installazione del sito. Inoltre analizzerò e valuterò cosa cambiare nel in base ai consigli ricevuti.

Diario di lavoro

Luogo	Trevano
Data	2019-11-28
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione ho sistemato l'input di ricerca della data che ho modificato l'ultima volta per la mostrare all'utente il formato corretto.

Successivamente ho aggiunto la modifica degli utenti nella pagina di gestione alla quale può accedere solo l'admin.

Per il momento il cambio password non può essere effettuato, questo perché in uno sviluppo futuro dovrebbe venire inserito un sistema di recupero password (tramite email codice per modifica password).

Modifica utente

Nome (massimo 50 caratteri)
Robbie

Cognome (massimo 50 caratteri)
Stenson

Email
rstenson4@technorati.com

Utente base Utente admin

Password (min 8 caratteri, max 50 caratteri, 1 maiuscola ed un numero)
Inserire password

CHIUDI SALVA

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima volta devo continuare il sistema di modifica degli utenti e gestire tutti gli eventuali errori nel modo corretto.

Diario di lavoro

Luogo	Trevano
Data	2019-11-29
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione ho terminato la parte di modifica degli utenti. Gli utenti admin possono modificare gli utenti. Per ora solo tramite un utente amministratore si può modificare la password. Se non viene inserita alcuna password al momento della modifica la password non viene alterata.

In futuro potrebbe essere aggiunto un sistema di recupero password accessibile direttamente agli utenti.

Id	Nome	Cognome	E-mail	Admin	Modifica utente	Elimina utente
65	Utente	Prova	prova@prova1.com	No	Modifica	Elimina
66	Niles	Ettritch	nettritch5@album.net	No	Modifica	Elimina
67	Dene	Fennelow	dfennelow6@bloomberg.com	No	Modifica	Elimina
68	Torrence	Radnedge	tradnedge7@businessinsider.com	No	Modifica	Elimina
69	Tiff	Room	troom8@abc.net.au	No	Modifica	Elimina
71	Olvan	Dunan	odunana@dyndns.org	No	Modifica	Elimina
72	Sydney	Havers	shaversb@seattletimes.com	No	Modifica	Elimina

Ottimo! L'utente è stato modificato con successo.

Successivamente ho aggiornata la documentazione inserendo la parte di modifica dell'utente.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione devo continuare la documentazione ed iniziare il capitolo sugli sviluppi futuri. Inoltre devo valutare cosa posso aggiungere con il tempo che ho a disposizione.

Diario di lavoro

Luogo	Trevano
Data	2019-12-03
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione ho reinserito l'input date che utilizzavo in precedenza, questo perché in automatico prende il formato della data dal sistema operativo. In questo modo ogni utente visualizza il formato della data come preferisce.

Successivamente mi sono occupato della documentazione, in particolare ho iniziato il capitolo delle limitazioni conosciute, il capitolo sulle conclusioni.

Sono andato in tutti i diari per vedere quali siti ho utilizzato durante il corso del progetto per poi inserirli nel capitolo bibliografia della documentazione. Ho inoltre creato il merge in formato pdf dei diari che ho fatto fino a questo momento in modo da arrivare alla consegna pronto.

La terza ora ho dovuto reinstallare project per iniziare il gantt consuntivo. Per fare ciò ho aperto tutti i diari dall'inizio del progetto.

Ho aggiunto un capitolo dove spiego come funziona il framework mvc, che però devo terminare la prossima lezione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione devo continuare la documentazione ed iniziare il capitolo sugli sviluppi futuri. Inoltre devo valutare cosa posso aggiungere con il tempo che ho a disposizione.

Diario di lavoro

Luogo	Trevano
Data	2019-12-06
Progetto	Knowledge Base

Lavori svolti

La prima parte di lezione mi sono dedicato alla creazione del Gantt consuntivo. Per fare ciò ho aperto tutti i diari da inizio anno fino ad adesso.

Oggi ho effettuato dei test sulla ricerca aggiuntivi.

Inoltre la scorsa volta ho effettuato test aggiuntivi su l'aggiunta di nuove categoria ed eliminazioni. Oggi ho ripetuto alcuni test.

Inoltre ho fatto alcune ricerche online su un eventuale diagramma gerarchico dei casi (non avendo molto tempo a disposizione probabilmente non lo implementerò).

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione continuo ad effettuare i test (devo aggiornare i risultati dei test sull'aggiunta/eliminazione categoria e sulla ricerca dei casi) ed iniziare a concludere la documentazione.

Diario di lavoro

Luogo	Trevano
Data	2019-12-10
Progetto	Knowledge Base

Lavori svolti

Oggi ho terminato i test. I risultati sono rimasti invariati. Ho riportato i loro esito nella relativa tabella dei risultati.

Ho creato l'abstract (documento a parte) utilizzando la base che ci è stata consegnata dal docente Valsangiacomo.

Inoltre ho effettuato una revisione della documentazione completando alcune parti.

Ho sistema un bug che ho trovato nel progetto:

quando viene creato un utente e l'operazione fallisce (solo nel caso che le due password non corrispondono) i dati non vengono salvati e riproposti, costringendo l'utente a inserire nuovamente tutti i dati.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

La prossima lezione mi dedicherò alla conclusione della documentazione. Devo terminare il consuntivo ed inserirlo nella documentazione.

Quaderno dei compiti

1 INFORMAZIONI GENERALI

Candidato	Nome: Bryan	Cognome: Beffa
	bryan.beffa@samtrevano.ch	079 860 82 88
Luogo di lavoro	Scuola Arti e Mestieri / CPT Trevano-Canobbio	
Orientamento	<input type="checkbox"/> 88601 Sviluppo di applicazioni <input checked="" type="checkbox"/> 88602 Informatica aziendale <input type="checkbox"/> 88603 Tecnica dei sistemi	
Superiore professionale	Nome: Ivan	Cognome: Raimondi
	ivan.raimondi@edu.ti.ch	
Perito 1	Nome:	Cognome:
Perito 2	Nome:	Cognome:
Periodo	3 settembre 2019 – 20 dicembre 2019 (presentazioni: 7-17 gennaio 2020)	
Orario di lavoro	Secondo orario scolastico 1° semestre	
Numero di ore	174	
Pianificazione 80h (in ore o %)	Analisi: 10% Implementazione: 50% Test: 10% Documentazione: 30%	

2 PROCEDURA

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

3 TITOLO

Applicativo web per la gestione di casi di supporto e documenti tecnici "knowledge base"

4 HARDWARE E SOFTWARE DISPONIBILE

1 PC

Ambiente di sviluppo per siti in PHP

Software ...AMP (web server, database e linguaggio di programmazione PHP)

Eventuali framework o template per lo sviluppo del sito

5 PREREQUISITI

nessuno

6 DESCRIZIONE DEL PROGETTO

Realizzare un applicativo web per la gestione della documentazione tecnica di una ditta di supporto informatico. Nel programma si devono poter registrare i casi che si presentano giornalmente in modo ordinato. Poder cercare facilmente tra i casi registrati è lo scopo finale del progetto. Il lavoro viene svolto a fasi e comprende

- preparazione di un ambiente di lavoro specifico con web server, PHP, editor, debugger
- realizzazione del programma in PHP
 - definizione dei ruoli e delle funzioni del programma
 - definizione della banca dati
 - funzioni
 - gestione categorie e casi
 - inserimento caso
 - ricerca
 - visualizzazione ultimi casi, casi più ricorrenti, ...
 - il programma deve espletare le funzioni previste e funzionare correttamente
 - struttura del programma basata su programmazione ad oggetti: classi e astrazione
 - l'interfaccia deve essere user friendly, facile da utilizzare e graficamente apprezzabile
- dettagli applicativo
 - attori: amministratori, operatori/utenti
 - elementi: caso, categorie, caso con prima data di apparizione, varianti, numero di ripresentazioni

7 RISULTATI FINALI

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

- Una pianificazione iniziale (entro il primo giorno)
- Una documentazione del progetto comprendente:
 - prerequisiti tecnici
 - installazione e configurazione del sito
 - descrizione della banca dati
 - descrizione dei programmi/pagine
 - schema funzioni per ruolo (use case)

- test e risultati
 - Un diario di lavoro
 - La pianificazione finale
-

8 PUNTI TECNICI SPECIFICI VALUTATI

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

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

1. 190 - *Design della GUI (Elaborazione di una maschera/schermo/sito/Internet)*
2. 192 - *Criteri di sicurezza IT*
3. 193 - *Design del GUI*
4. 194 - *Attendibilità dei dati inseriti dall'utilizzatore*
5. 196 - *Illustrazione dello stato effettivo/obiettivo del Workflow*
6. 232 - *Programmazione web professionale*
7. 237 - *Analisi di sicurezza (Applicazione Web)*

9 FIRMA

Candidato

Canobbio, 03.09.2019

Superiore professionale

Canobbio, 03.09.2019

Perito 1

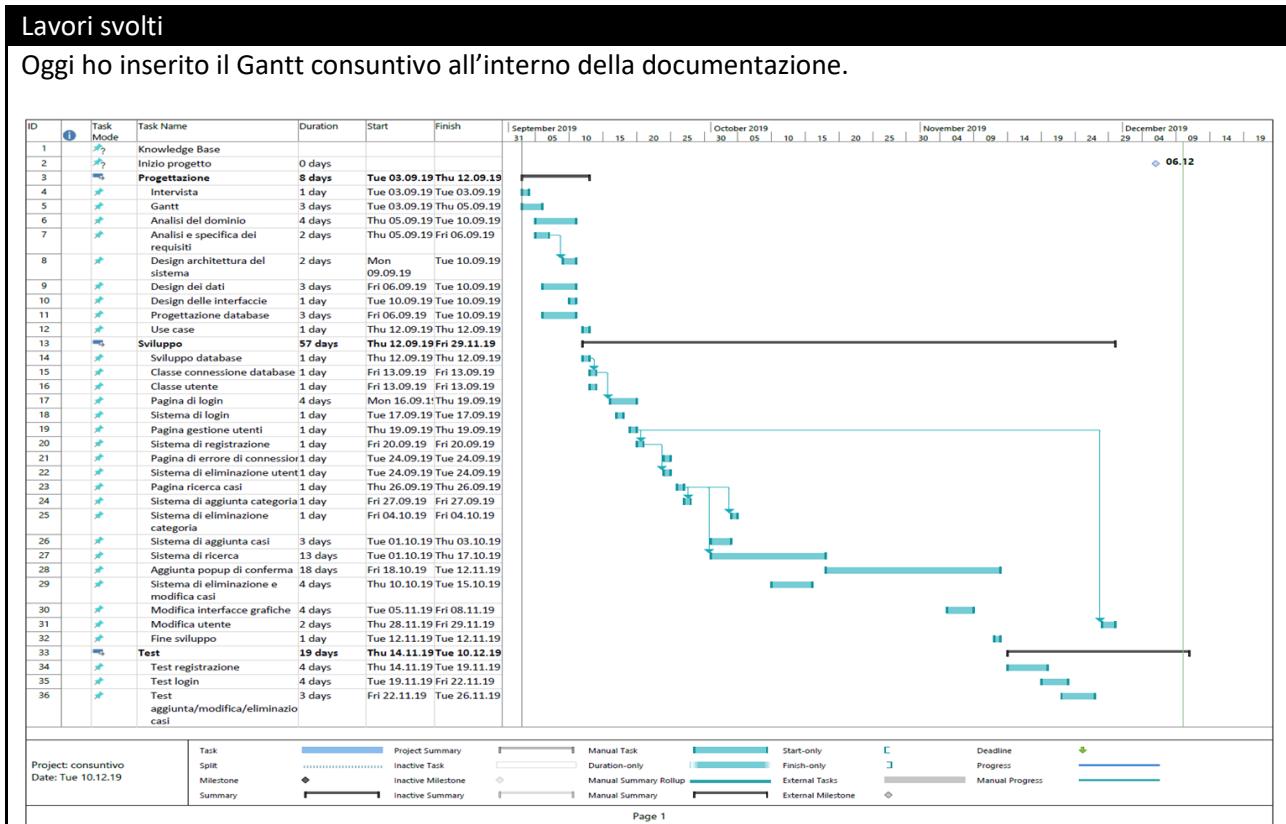
(luogo e data)

Perito 2

(luogo e data)

Diario di lavoro

Luogo	Trevano
Data	2019-12-12
Progetto	Knowledge Base



Ho iniziato a stampare i documenti per la consegna: diari fino ad oggi, guida xdebug ed abstract.
Il resto del tempo ho modificato la documentazione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione
In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro
Le prossime lezione saranno dedicate alla rilettura della documentazione e alla conclusione del progetto.

Diario di lavoro

Luogo	Trevano
Data	2019-12-13
Progetto	Knowledge Base

Lavori svolti

Oggi ho stampato la documentazione per evitare di arrivare all'ultimo con le stampe. Con il compagno Filippo Finke abbiamo posto alcune domande al docente Valsangiacomo relative alla consegna del progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

-

Diario di lavoro

Luogo	Trevano
Data	2019-12-17
Progetto	Knowledge Base

Lavori svolti

Oggi mi sono dedicato alla rilegatura del progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Masterizzazione su cd del progetto.

Diario di lavoro

Luogo	Trevano
Data	2019-12-19
Progetto	Knowledge Base

Lavori svolti

Oggi mi sono alla masterizzazione del progetto sul cd docente valsangiacomo.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

Consegna del progetto.

Diario di lavoro

Luogo	Trevano
Data	2019-12-20
Progetto	Knowledge Base

Lavori svolti

Consegna del progetto.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione

Programma di massima per la prossima giornata di lavoro

-

1 INFORMAZIONI GENERALI

Candidato	Nome: Bryan	Cognome: Beffa
	bryan.beffa@samtrevano.ch	079 860 82 88
Luogo di lavoro	Scuola Arti e Mestieri / CPT Trevano-Canobbio	
Orientamento	<input type="checkbox"/> 88601 Sviluppo di applicazioni <input checked="" type="checkbox"/> 88602 Informatica aziendale <input type="checkbox"/> 88603 Tecnica dei sistemi	
Superiore professionale	Nome: Ivan	Cognome: Raimondi
	ivan.raimondi@edu.ti.ch	
Perito 1	Nome:	Cognome:
Perito 2	Nome:	Cognome:
Periodo	3 settembre 2019 – 20 dicembre 2019 (presentazioni: 7-17 gennaio 2020)	
Orario di lavoro	Secondo orario scolastico 1° semestre	
Numero di ore	174	
Pianificazione 80h (in ore o %)	Analisi: 10% Implementazione: 50% Test: 10% Documentazione: 30%	

2 PROCEDURA

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

3 TITOLO

Applicativo web per la gestione di casi di supporto e documenti tecnici "knowledge base"

4 HARDWARE E SOFTWARE DISPONIBILE

1 PC

Ambiente di sviluppo per siti in PHP

Software ...AMP (web server, database e linguaggio di programmazione PHP)

Eventuali framework o template per lo sviluppo del sito

5 PREREQUISITI

nessuno

6 DESCRIZIONE DEL PROGETTO

Realizzare un applicativo web per la gestione della documentazione tecnica di una ditta di supporto informatico. Nel programma si devono poter registrare i casi che si presentano giornalmente in modo ordinato. Poder cercare facilmente tra i casi registrati è lo scopo finale del progetto. Il lavoro viene svolto a fasi e comprende

- preparazione di un ambiente di lavoro specifico con web server, PHP, editor, debugger
- realizzazione del programma in PHP
 - definizione dei ruoli e delle funzioni del programma
 - definizione della banca dati
 - funzioni
 - gestione categorie e casi
 - inserimento caso
 - ricerca
 - visualizzazione ultimi casi, casi più ricorrenti, ...
 - il programma deve espletare le funzioni previste e funzionare correttamente
 - struttura del programma basata su programmazione ad oggetti: classi e astrazione
 - l'interfaccia deve essere user friendly, facile da utilizzare e graficamente apprezzabile
- dettagli applicativo
 - attori: amministratori, operatori/utenti
 - elementi: caso, categorie, caso con prima data di apparizione, varianti, numero di ripresentazioni

7 RISULTATI FINALI

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

- Una pianificazione iniziale (entro il primo giorno)
- Una documentazione del progetto comprendente:
 - prerequisiti tecnici
 - installazione e configurazione del sito
 - descrizione della banca dati
 - descrizione dei programmi/pagine
 - schema funzioni per ruolo (use case)

- test e risultati
 - Un diario di lavoro
 - La pianificazione finale
-

8 PUNTI TECNICI SPECIFICI VALUTATI

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

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

1. 190 - *Design della GUI (Elaborazione di una maschera/schermo/sito/Internet)*
2. 192 - *Criteri di sicurezza IT*
3. 193 - *Design del GUI*
4. 194 - *Attendibilità dei dati inseriti dall'utilizzatore*
5. 196 - *Illustrazione dello stato effettivo/obiettivo del Workflow*
6. 232 - *Programmazione web professionale*
7. 237 - *Analisi di sicurezza (Applicazione Web)*

9 FIRMA

Candidato

Canobbio, 03.09.2019

Superiore professionale

Canobbio, 03.09.2019

Perito 1

(luogo e data)

Perito 2

(luogo e data)
