GestioneProgetti

1 Implementazione 2

2 Test 2

2.1 Protocollo di test 2

2.2 Risultati test 3

2.3 Mancanze/limitazioni conosciute 3

3 Consuntivo 3

4 Conclusioni 3

4.1 Sviluppi futuri 3

4.2 Considerazioni personali 3

5 Bibliografia 3

5.1 Bibliografia per articoli di riviste: 3

5.2 Bibliografia per libri 4

5.3 Sitografia 4

6 Allegati 4

# Implementazione

## Preparazione dell’ambiente di lavoro

La prima attività che ho svolto nell’implementazione è stata quella di impostare il mio ambiente di lavoro. Ho quindi installato sul mio computer un web server (XAMPP), un editor di testo comodo per la programmazione (Visual Studio Code), e un client DB per poter creare e gestire il mio database (HeidiSQL). Ho inoltre creato una repository su GitHub ([www.github.com](http://www.github.com)) per poter eseguire il backup dei miei dati, e una bacheca Trello ([www.trello.com](http://www.trello.com)) per tenere traccia delle task in corso e delle task ancora da iniziare. È un modo molto comodo per sapere sempre quello che bisogna fare.

## Creazione database e tabelle

Dopo aver preparato l’ambiente di lavoro, ho creato il database e le sue tabelle. È d’obbligo creare il database prima di iniziare a implementare il sito, perché il sito lavora su database.

Il codice per creare il database è il seguente:

CREATE DATABASE IF NOT EXISTS `gestioneprogetti`;

USE `gestioneprogetti`;

.

La tabella **utenti** l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `utenti` (

`id\_utente` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(50) NOT NULL,

`password` varchar(100) NOT NULL,

`email` varchar(100) NOT NULL,

`amministratore` tinyint(1) NOT NULL DEFAULT '0',

`amministratore\_team` tinyint(1) NOT NULL DEFAULT '0',

PRIMARY KEY (`id\_utente`)

)

Questa tabella presenta 6 campi:

* L’id dell’utente, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* Lo username dell’utente, che è di tipo varchar e non può essere NULL.
* La password dell’utente che è di tipo varchar e non può essere NULL.
* L’email dell’utente che è di tipo varchar e non può essere NULL.
* Un valore booleano (tinyint) che definisce se l’utente è amministratore. Il valore di default è 0 (ovvero, l’utente non è amministratore). Non può essere NULL.
* Un valore booleano (tinyint) che definisce se l’utente è amministratore di team. Il valore di default è 0 (ovvero, l’utente non è amministratore di team). Non può essere NULL.

La tabella **teams** l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `teams` (

`id\_team` int(11) NOT NULL AUTO\_INCREMENT,

`nome` varchar(50) NOT NULL,

PRIMARY KEY (`id\_team`)

)

Questa tabella presenta 2 campi:

* L’id del team, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* Il nome del team, che è di tipo varchar e non può essere NULL.

La tabella **appartenenza**, che è la tabella che lega gli utenti ai teams, l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `appartenenza` (

`id\_appartenenza` int(11) NOT NULL AUTO\_INCREMENT,

`id\_utente` int(11) NOT NULL,

`id\_team` int(11) NOT NULL,

PRIMARY KEY (`id\_appartenenza`),

KEY `FK\_appartenenza\_utenti` (`id\_utente`),

KEY `FK\_appartenenza\_teams` (`id\_team`),

CONSTRAINT `FK\_appartenenza\_teams` FOREIGN KEY (`id\_team`) REFERENCES `teams`

(`id\_team`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_appartenenza\_utenti` FOREIGN KEY (`id\_utente`) REFERENCES

`utenti` (`id\_utente`) ON DELETE CASCADE ON UPDATE CASCADE

)

Questa tabella presenta 3 campi:

* Id\_appartenenza, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* L’id dell’utente, che è un intero, non può essere NULL, e fa riferimento all’id\_utente della tabella **utenti**.
* L’id del team, che è un intero, non può essere NULL, e fa riferimento all’id\_team della tabella **teams**.

La tabella **progetti** l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `progetti` (

`id\_progetto` int(11) NOT NULL AUTO\_INCREMENT,

`nome` varchar(50) NOT NULL,

`datainizio` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`id\_team` int(11) NOT NULL,

PRIMARY KEY (`id\_progetto`),

KEY `FK\_progetti\_teams` (`id\_team`),

CONSTRAINT `FK\_progetti\_teams` FOREIGN KEY (`id\_team`) REFERENCES `teams`

(`id\_team`) ON DELETE CASCADE ON UPDATE CASCADE

)

Presenta 4 campi:

* L’id del progetto, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* Il nome del progetto, che è di tipo varchar e non può essere NULL.
* La data di creazione del progetto, che è di tipo timestamp, non può essere NULL, e come valore di default ha la data e ora attuale (di quando si crea un progetto)
* L’id del team a cui appartiene il progetto, che è di tipo intero, non può essere NULL, e fa riferimento al campo id\_team della tabella **teams**.

La tabella **assegnamenti**, che è la tabella che collega gli utenti ai progetti (quale utente svolge quale progetto) l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `assegnamenti` (

`id\_assegnamento` int(11) NOT NULL AUTO\_INCREMENT,

`id\_utente` int(11) NOT NULL,

`id\_progetto` int(11) NOT NULL,

PRIMARY KEY (`id\_assegnamento`),

KEY `FK\_assegnamenti\_progetti` (`id\_progetto`),

KEY `FK\_assegnamenti\_utenti` (`id\_utente`),

CONSTRAINT `FK\_assegnamenti\_progetti` FOREIGN KEY (`id\_progetto`) REFERENCES

`progetti` (`id\_progetto`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_assegnamenti\_utenti` FOREIGN KEY (`id\_utente`) REFERENCES

`utenti` (`id\_utente`) ON DELETE CASCADE ON UPDATE CASCADE

)

Questa tabella presenta 3 campi:

* L’id dell’assegnamento, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* L’id dell’utente, che è un intero, non può essere NULL, e fa riferimento all’id\_utente della tabella **utenti**.
* L’id del progetto, che è un intero, non può essere NULL, e fa riferimento all’id\_progetto della tabella **progetti**.

La tabella **tasks** l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `tasks` (

`id\_task` int(11) NOT NULL AUTO\_INCREMENT,

`nome` varchar(20) NOT NULL,

`tipo` varchar(20) NOT NULL,

`priorita` int(11) NOT NULL,

`descrizione` varchar(10000) NOT NULL,

`completamento` tinyint(4) NOT NULL DEFAULT '0',

`inizio` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`fine` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',

`id\_progetto` int(11) NOT NULL,

PRIMARY KEY (`id\_task`),

KEY `FK\_\_progetti` (`id\_progetto`),

CONSTRAINT `FK\_\_progetti` FOREIGN KEY (`id\_progetto`) REFERENCES `progetti`

(`id\_progetto`) ON DELETE CASCADE ON UPDATE CASCADE

)

Questa tabella presenta 9 campi:

* L’id della task, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* Il nome della task, che è di tipo varchar e non può essere NULL.
* Il tipo (task o milestone), che è di tipo varchar e non può essere NULL.
* La priorità, che è di tipo intero e non può essere NULL.
* La descrizione della task, che è di tipo varchar e non può essere NULL.
* La percentuale di completamento, che è di tipo tinyint e non può essere NULL. Di default ha il valore 0.
* L’inizio della task, che è di tipo timestamp, non può essere NULL, e come valore di default ha la data e ora attuale (di quando si crea la task)
* La fine della task, che è di tipo timestamp e non può essere NULL.
* L’id del progetto a cui la task appartiene, che è un intero e non può essere NULL, e fa riferimento al campo id\_progetto della tabella **progetti**.

La tabella **svolgimento**, che è la tabella che collega le task agli utenti (quale utente è assegnato a quale task) l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `svolgimento` (

`id\_svolgimento` int(11) NOT NULL AUTO\_INCREMENT,

`id\_utente` int(11) NOT NULL,

`id\_task` int(11) NOT NULL,

PRIMARY KEY (`id\_svolgimento`),

KEY `FK\_svolgimento\_utenti` (`id\_utente`),

KEY `FK\_svolgimento\_tasks` (`id\_task`),

CONSTRAINT `FK\_svolgimento\_tasks` FOREIGN KEY (`id\_task`) REFERENCES `tasks`

(`id\_task`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_svolgimento\_utenti` FOREIGN KEY (`id\_utente`) REFERENCES

`utenti` (`id\_utente`) ON DELETE CASCADE ON UPDATE CASCADE

)

Questa tabella presenta 3 campi:

* L’id\_svolgimento, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* L’id dell’utente, che è un intero, non può essere NULL, e fa riferimento all’id\_utente della tabella **utenti**.
* L’id della task, che è un intero, non può essere NULL, e fa riferimento all’id\_task della tabella **tasks**.

La tabella **richieste**, che è la tabella in cui vengono salvate le richieste per diventare amministratori di team, l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `richieste` (

`id\_richiesta` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(50) NOT NULL,

PRIMARY KEY (`id\_richiesta`)

)

Questa tabella presenta 2 campi:

* L’id\_richiesta, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* Lo username dell’utente, che è di tipo varchar e non può essere NULL.

La tabella **richieste\_team**, che è la tabella in cui vengono salvate le richieste di entrare in un determinato team, l’ho creata nel seguente modo:

CREATE TABLE IF NOT EXISTS `richieste\_team` (

`id\_richiesta\_team` int(11) NOT NULL AUTO\_INCREMENT,

`id\_richiedente` int(11) NOT NULL,

`team` varchar(50) NOT NULL,

PRIMARY KEY (`id\_richiesta\_team`)

)

Questa tabella presenta 3 campi:

* L’id della richiesta, che è la chiave primaria, di tipo intero e si auto incrementa. Non può avere NULL come valore.
* L’id del richiedente, che è di tipo intero e non può avere NULL come valore.
* Il nome del team, che è di tipo varchar e non può avere NULL come valore.

Queste ultime due tabelle, ovvero quelle delle richieste, non hanno relazioni con altre tabelle, perché le ho trattate come due tabelle “a parte”. L’inserimento, la modifica e l’eliminazione dei dati su queste due tabelle ho deciso di farla tutta da codice (es. quando elimino un utente, vado manualmente con php a fare la query che elimina le richieste di quell’utente anziché farlo fare in automatico con le relazioni e l’ON DELETE CASCADE).

## Implementazione web

### Il file ‘config.php’

<?php

//indirizzo sever con database

define('DB\_SERVER', '127.0.0.1');

//utente

define('DB\_USERNAME', 'root');

//password

define('DB\_PASSWORD', '');

//nome database

define('DB\_DATABASE', 'gestioneprogetti');

//eseguo la connessione

$db = mysqli\_connect(DB\_SERVER, DB\_USERNAME, DB\_PASSWORD, DB\_DATABASE);

?>

Il file “config.php” è il file di configurazione che permette di connettersi al database. Gli vengono passati ip, nome utente, password e nome del database server. Dopodiché viene creata una connessione, che viene salvata nella variabile $db.

### Il file ‘session.php’

<?php

include('config.php');

//inizio la sessione

session\_start();

//se non è settato in sessione l'utente che ha fatto il login

if(!isset($\_SESSION['login\_user']) && !isset($\_SESSION['login\_id'])){

//reindirizzo al login

header("location: login.php");

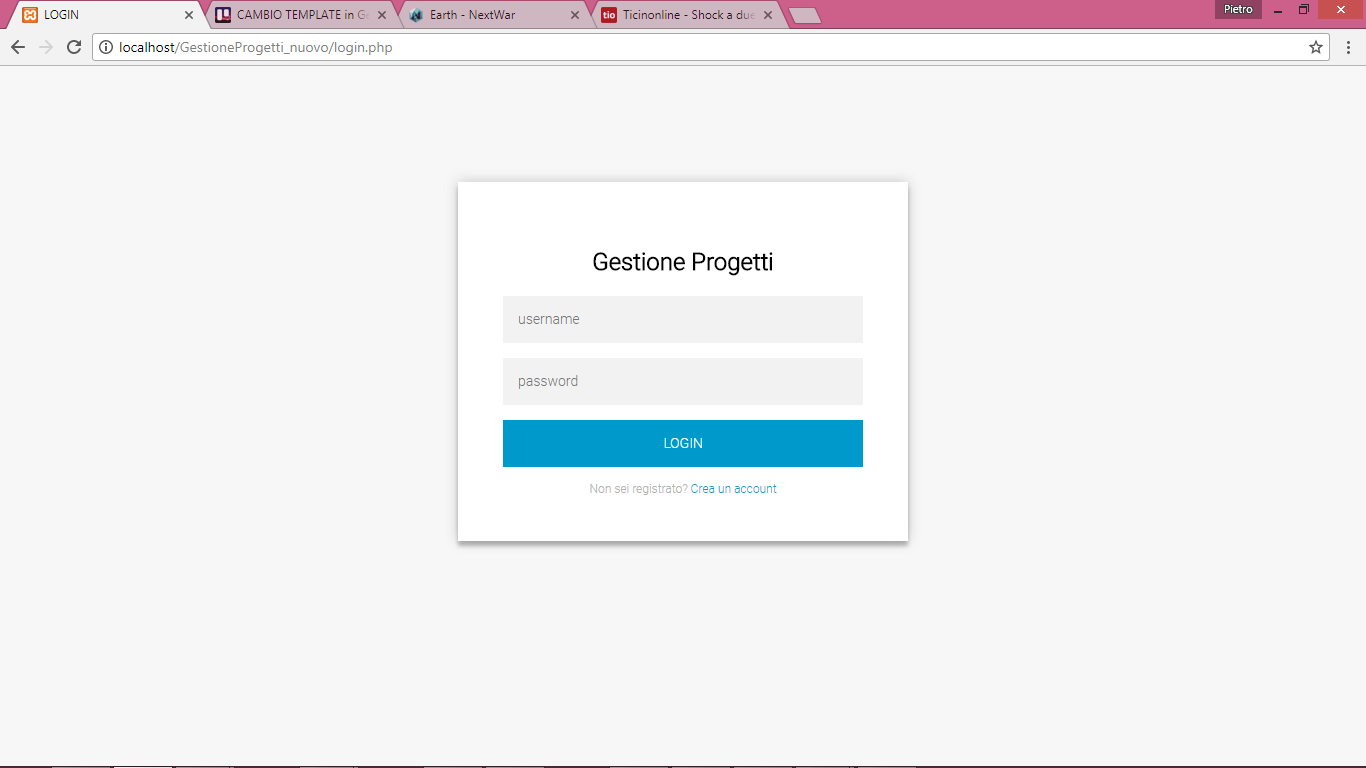
}

?>

Il file “session.php” serve per fare in modo che, se un utente non ha ancora fatto il login, esso non possa accedere a nessuna pagina nel sito. Se un utente prova ad accedere a una pagina del sito senza aver prima aver fatto il login, verrà reindirizzato alla pagina “login.php”.

### Login

La pagina di login si presenta nel seguente modo:



È molto simile a come è stata progettata. Ci sono due input per nome utente e password e un bottone per effettuare il login. Sotto al bottone “LOGIN” è presente un link per accedere alla pagina di registrazione, se non si ha ancora un account.

Il codice per il login è molto semplice. Per prima cosa includo il file ‘session.php’ (che a sua volta include il file ‘config.php). Dopodiché, quando viene premuto il bottone “LOGIN”, vado a leggere tramite POST i valori inseriti negli input nel seguente modo:

//leggo i valori inseriti negli input

$myusername = mysqli\_real\_escape\_string($db,$\_POST['username']);

$mypassword = md5(mysqli\_real\_escape\_string($db,$\_POST['password']));

La funzione ***mysqli\_real\_escape\_string()*** serve per evitare le SQL Injections (aggiunge dei \ davanti agli apici inseriti negli input). La funzone ***md5()*** serve per crittografare la password e aumentare la sicurezza.

Dopo aver letto i valori inseriti, vado a controllare se i dati sono corretti:

//preparo la query che controlla se i dati inseriti sono giusti

$stmt = $db->prepare("SELECT id\_utente, username,

password, amministratore, amministratore\_team

FROM utenti

WHERE username=? AND password=? LIMIT 1");

Per prima cosa perparo una query che seleziona i dati dell’utente con quello username e con quella password (valori inseriti negli input).

Se la query è corretta, passo i valori da controllare alla query e la eseguo.

if($stmt){

$stmt->bind\_param('ss', $myusername, $mypassword);

//eseguo la query

$stmt->execute();

Dopodiché salvo i risultati della query in delle variabili, e controllo se la query ha prodotto un risultato. Se produce un risultato significa che i dati inseriti sono corretti, e dunque salvo nella sessione i dati dell’utente. Infine lo reindirizzo alla pagina “index.php”.

$stmt->bind\_result($id\_utente, $myusername, $mypassword, $admin, $adminteam);

$stmt->store\_result();

//se i dati inseriti sono giusti

if($stmt->num\_rows == 1)

{

if($stmt->fetch())

{

//salvo i dati dell'utente nella sessione

$\_SESSION['is\_admin'] = $admin;

$\_SESSION['teamadmin'] = $adminteam;

$\_SESSION['login\_user'] = $myusername;

$\_SESSION['login\_id'] = $id\_utente;

//reindirizzo l'utente alla pagina index

header('Location: index.php');

}

}

//se i dati non sono corretti

else {

//stampo il messaggio di errore

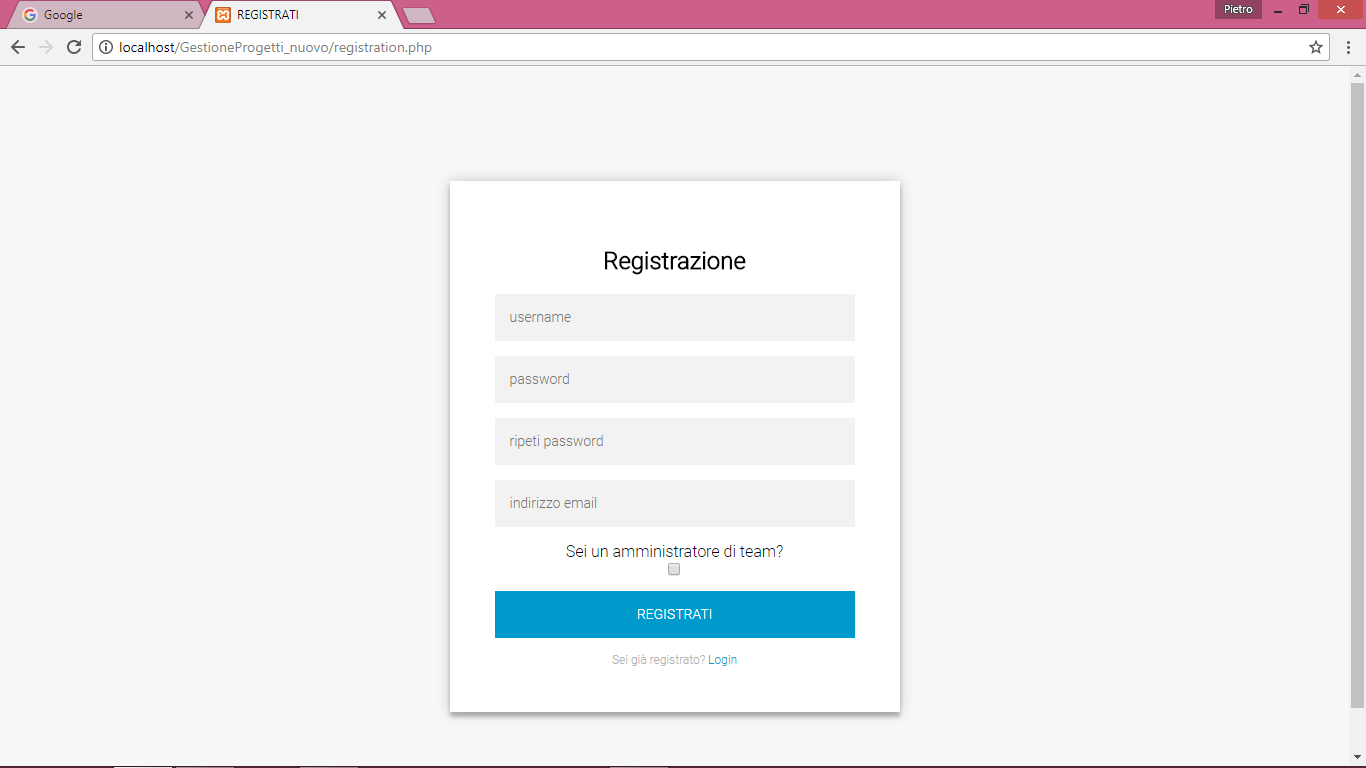
alert("Username o password errati");

}

$stmt->close();

### Registrazione

La pagina di registrazione si presenta nel seguente modo:



È molto simile a come è stata progettata. Ci sono un totale di 5 input e un bottone per effettuare la registrazione. Negli input bisogna inserire lo username, la password, la ripetizione della password, l’indirizzo email e un checkbox per fare la richiesta di diventare amministratore di team.

Il codice della pagina di registrazione è anch’esso abbastanza semplice. Per prima cosa bisogna includere il file “session.php”. Dopodiché, quando il bottone “REGISTRATI” viene cliccato, vengono salvati gli input inseriti in delle variabili:

//salvo i dati inseriti in delle variabili

$myusername = mysqli\_real\_escape\_string($db,$\_POST['username']);

$mypassword = md5(mysqli\_real\_escape\_string($db,$\_POST['password']));

$repeat = md5(mysqli\_real\_escape\_string($db,$\_POST['repeatpassword']));

$myemail = mysqli\_real\_escape\_string($db,$\_POST['email']);

Le due password le salvo con la crittografia md5. Dopo aver salvato le variabili vado ad effettuare i seguenti controlli sui dati inseriti:

* Lo username deve essere lungo almeno 3 caratteri
* Lo username non deve essere già in uso
* La password deve essere di almeno 8 caratteri
* La password deve contenere almeno un numero e una lettera maiuscola
* L’email deve essere inserita in un formato valido ([xxx@yyy.zzz](mailto:xxx@yyy.zzz))
* La password e la ripetizione della password devono essere uguali

//query per salvare i dati nel database

$stmt = $db->prepare("INSERT INTO utenti (username, password, email,

amministratore\_team) VALUES(?, ?, ?, ?)");

Se tutti i valori inseriti sono corretti preparo la query che andrà a inserire i dati dell’utenti nel database.

Dopodiché controllo se il campo “amministratore di team” è stato cliccato. Se l’utente vuole quindi registrarsi come amministratore di team, preparo la query che aggiunge la richiesta al database e che verrà poi accettata o rifiutata dall’amministratore.

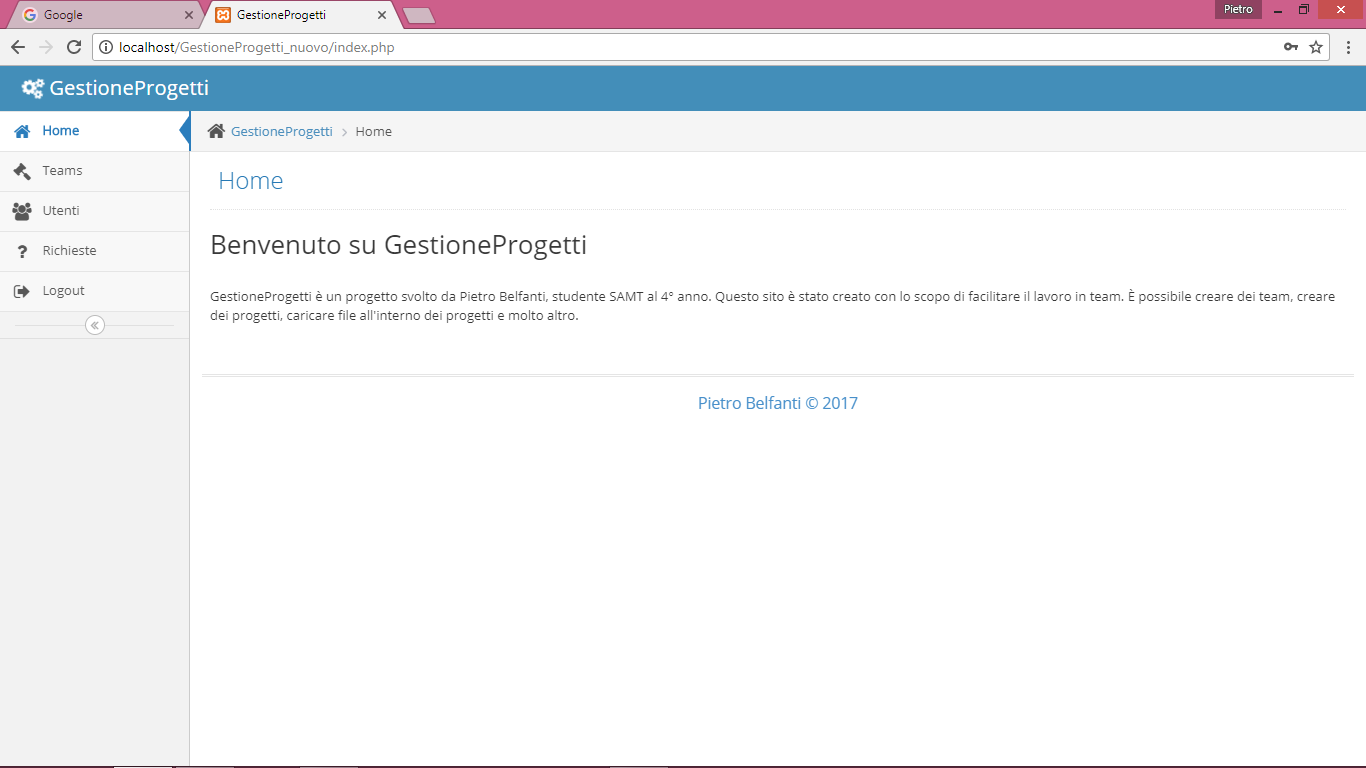
//preparo la richiesta da inserire nel database

$stmt2 = $db->prepare("INSERT INTO richieste (username) VALUES (?)");

Dopo aver eseguito anche questa query, se non si sono verificati errori reindirizzo l’utente al login, altrimenti stampo un messaggio di errore.

### Template

La home del sito si presenta nel seguente modo:



È un template responsive gratis che ho scaricato dal seguente link:

<https://colorlib.com/wp/free-html5-admin-dashboard-templates/> (template ace-master)

La pagina è composta da una barra di navigazione laterale, da una barra superiore contenente un icona e il nome del progetto, e dal contenuto principale.

### Gestione amministrativa

L’amministratore generale ha una gestione generale sui teams e sugli utenti e può accettare/rifiutare le richieste per diventare amministratori di team.

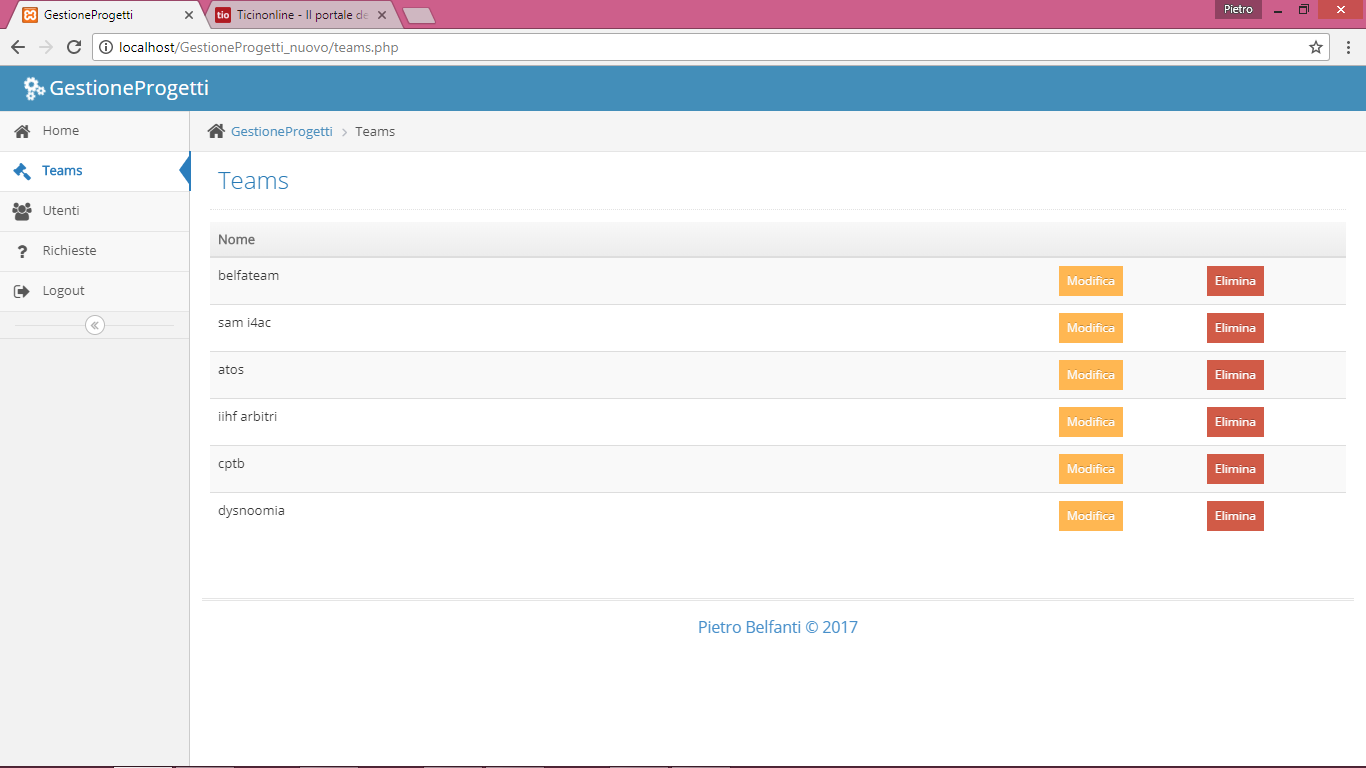
Per ogni pagina di amministrazione è presente un controllo per vedere se l’utente in sessione è effettivamente un amministratore o no. Se l’utente in sessione non è un amministratore viene reindirizazo alla pagina index.

if($\_SESSION['is\_admin'] == 0){

    header('Location: index.php');

}

#### Teams



La prima pagina di amministrazione è quella dei teams. È composta da una tabella contenente i nomi i di tutti i team, e per ogni team è possibile modificare o eliminare il team.

Per stampare una tabella contenente i dati del database, il codice è il seguente:

//query per selezionare tutti i dati della tabella teams

$sql = "SELECT \* FROM gestioneprogetti.teams";

$result = mysqli\_query($db,$sql) or die(mysql\_error());

Per prima cosa selezionodal database tutti i teams esistenti.

//leggo il numero di righe prodotte dalla query

$nr = mysqli\_num\_rows($result);

Poi memorizzo il numero di righe prodotte dalla query.

Se la query non produce risultati significa che non sono ancora stati creati team, dunque invece della tabella stampo un messaggio con scritto che nessun team è ancora stato creato.

//se la query produce 0 righe

//stampo la scritta di defuault

if($nr == 0){

    echo "<br>Non è ancora stato creato nessun team.<br><br>";

}

Se invece sono stati trovati dei teams:

//se la query produce più di una riga

else{

    //stampo il div contenente la tabella

    echo "<div class='table-responsive'>";

    echo "<table class='table table-striped'>";

    echo "<thead><tr><th>Nome</th><th></th> <th></th></tr></thead><tbody>";

    while($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)) {

        echo "<tr>";

        echo '<td width="74%">' . $row['nome'] . '</td>';

        //stampo il bottone di modifica

        echo '<td width="13%"><a class="btn btn-warning btn-xs"

href="modifica\_team.php?nome=' . $row['nome'] . '&id\_team=' .

$row['id\_team']. '">Modifica</a></td>';

        //stampo il bottone di eliminazione

        echo '<td width="13%"><a class="btn btn-danger btn-xs"

onclick="confirmationDelete(this);return false;"

href="elimina\_team.php?nome=' . $row['nome'] .

'">Elimina</a></td>';

        echo "</tr>";

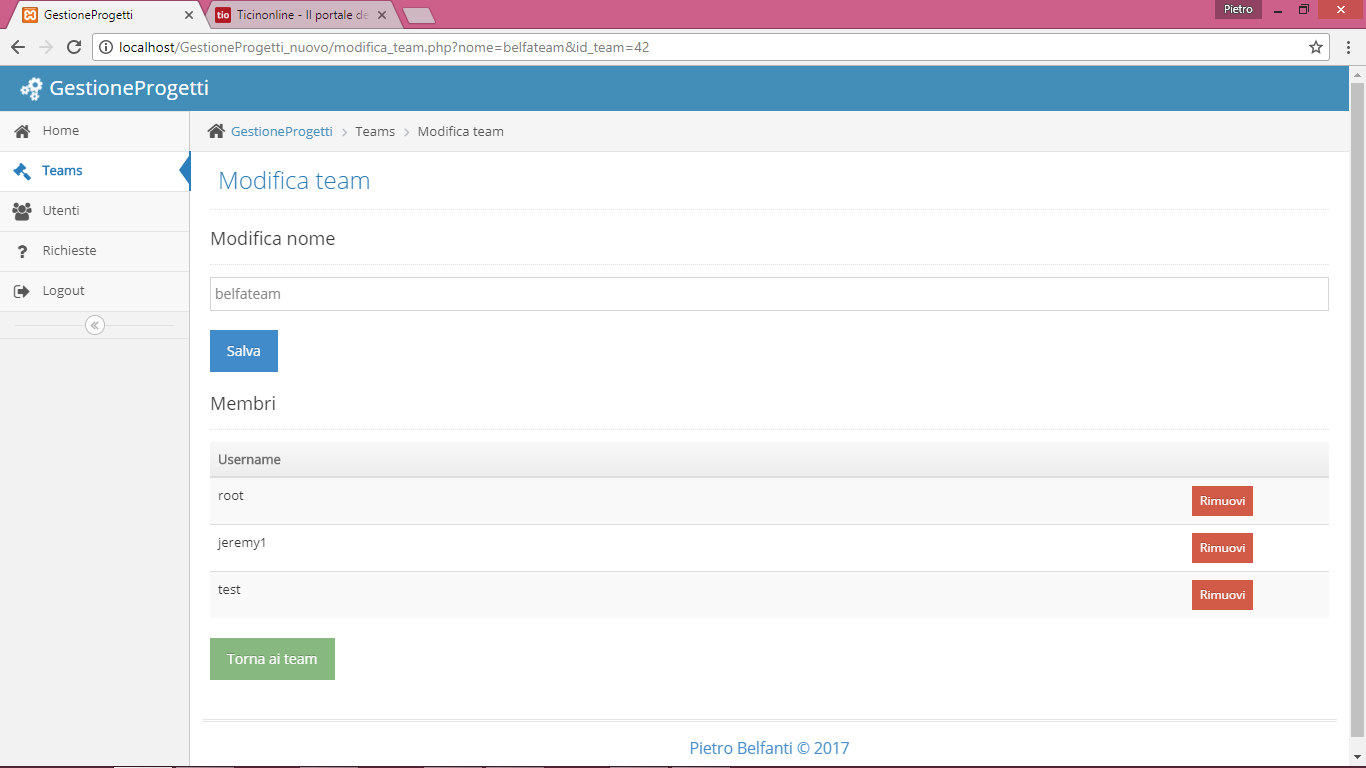
    }

    echo "</tbody></table>";

    echo "</div>";

}

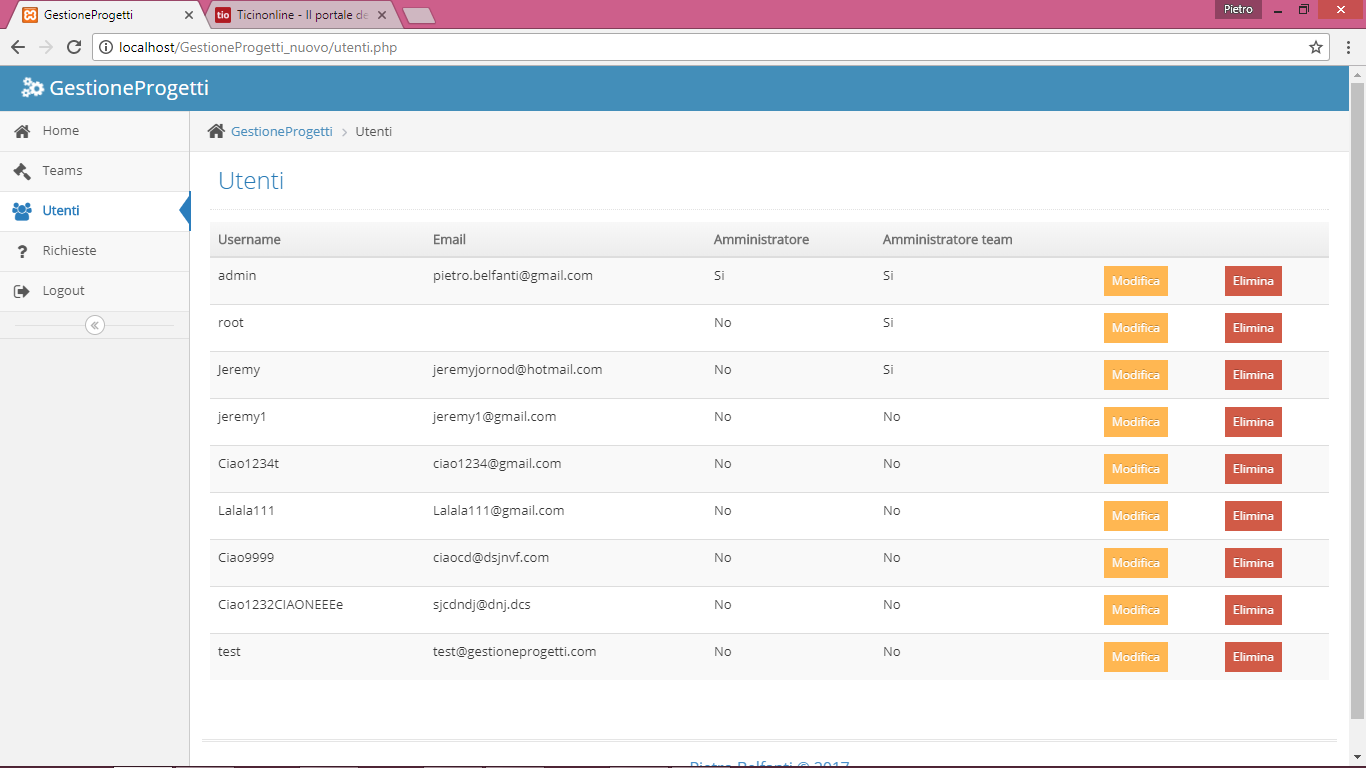
Apro il tag del div e della tabella (applicando delle classi bootstrap per il responsive e per la grafica), e poi faccio un ciclo while che cicla tutti gli elementi presenti nel risultato della query. Per ogni team trovato verrà quindi aggiunto un <tr> contenente il nome del team, il bottone di modifica e il bottone di eliminazione.



Cliccando su “modifica” si viene reindirizzati a questa pagina. È presente un form contenente un input per modificare il nome, e poi c’è una tabella che contiene i membri del team con un bottone per rimuovere questi utenti dal team. Quando si clicca sul bottone “salva” nella modifica del nome, viene semplicemente eseguita una query che modifica nel database il nome del team che si sta modificando. La funzione rimuovi semplicemente legge l’id\_utente dell’utente che si vuole rimuovere, e si esegue una query che elimina dalla tabella **appartenenza** il record dove l’id\_utente è quello da rimuovere e l’id\_team è quello del team che si sta modificando. Rimuovendo un utente da un team, esso verrà automaticamente rimosso anche dai progetti del team e dalle task dei progetti del team.

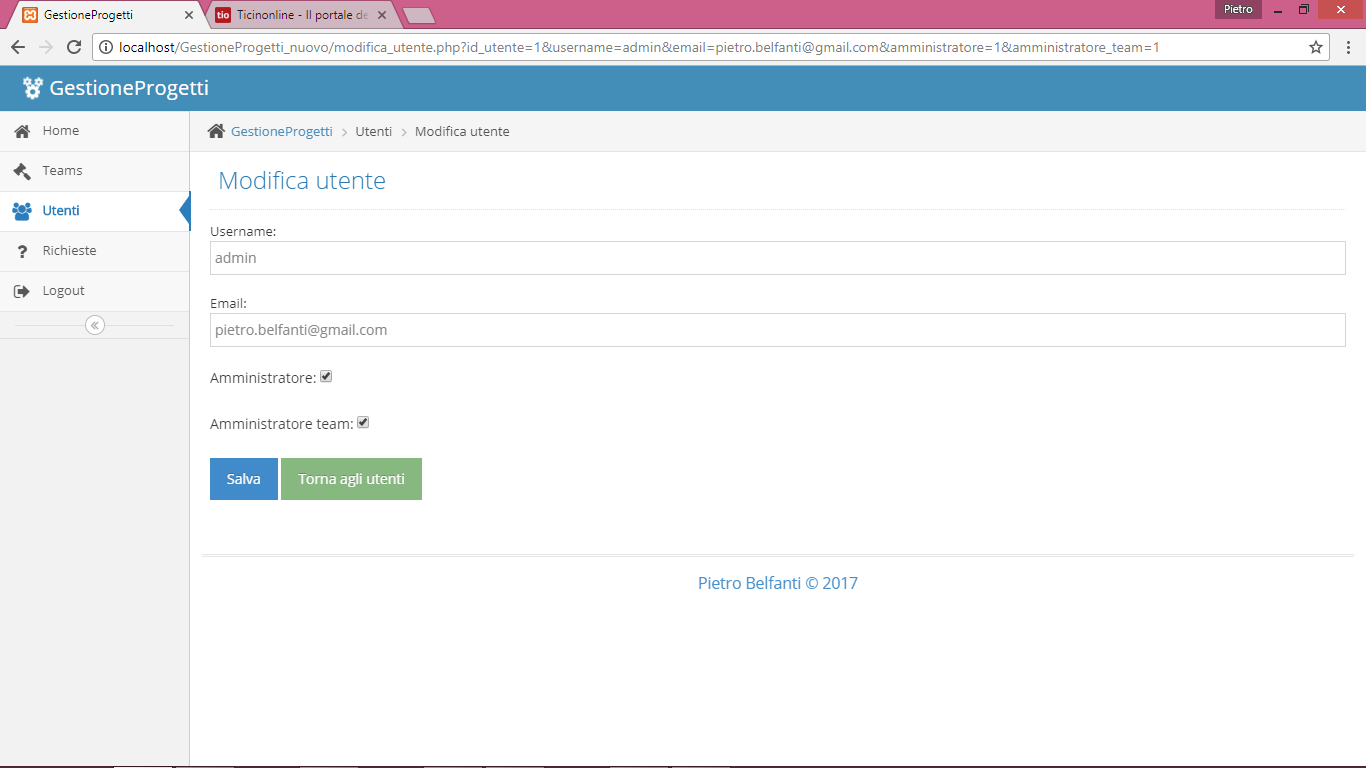
Cliccando su “elimina\* semplicemente viene eliminato dalla tabella **teams** in base all’id del team che si sta eliminando.

#### Utenti



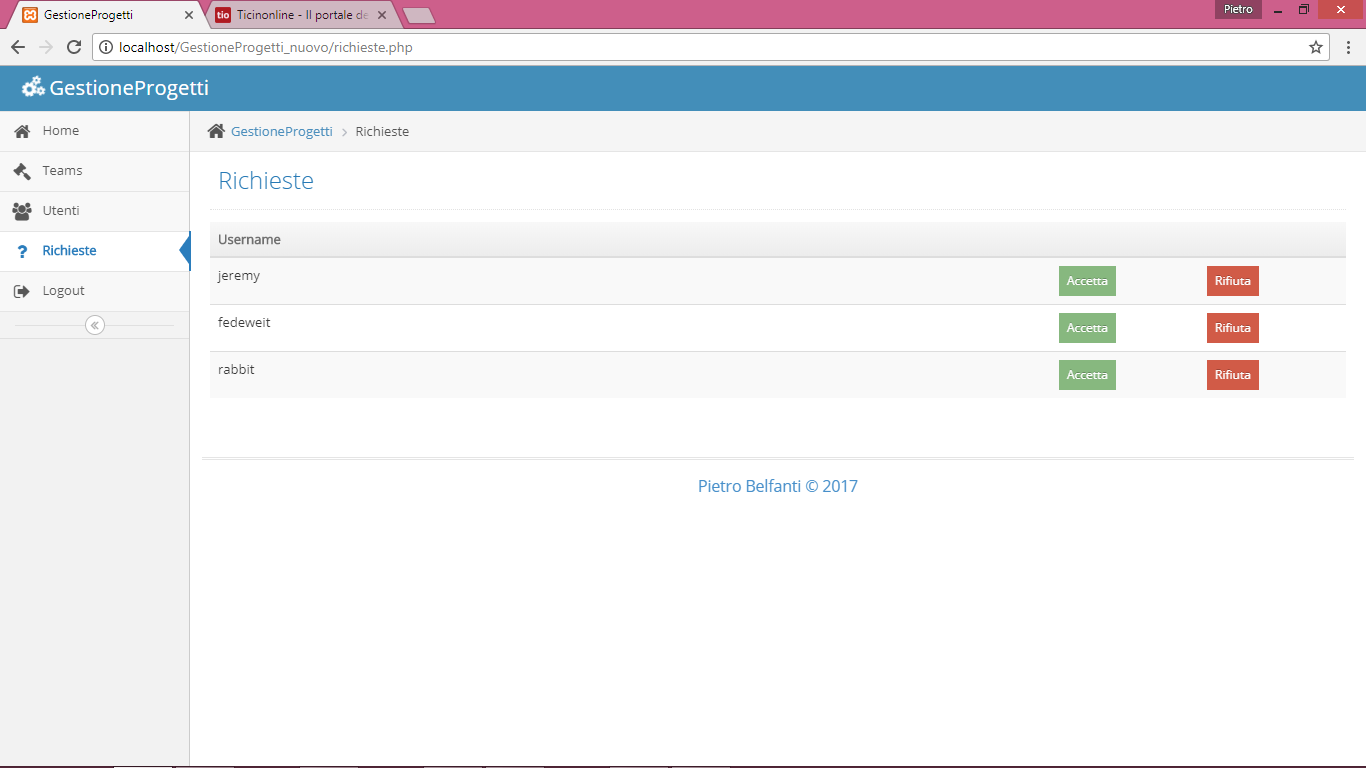
La pagina di gestione degli utenti è uguale a quella di gestione dei teams. C’è la tabella contenente i dati di ogni utente e i bottoni di modifica/eliminazione.

L’eliminazione funziona allo stesso modo. La pagina di modifica è un po’ differente:



Sono presenti 4 input (username, email, amministratore, amministratore di team) che hanno come valore di default i dati dell’utente (passati tramite POST). Il bottone “salva” esegue le modifiche (sempre stando ai controlli visti nella pagina di registrazione) e il bottone “torna agli utenti” serve per tornare alla pagina contenente la lista degli utenti. La modifica avviene tramite una query che va a fare un update dei dati nella tabella **utenti** in base ai nuovi dati inseriti.

#### Richieste



La pagina delle **richieste** presenta anche essa una tabella conenente gli username degli utenti che vogliono diventare amministratori di team, e i bottoni accetta/rifiuta richiesta.

Il modo per stampare la tabella è sempre quello visto in precedenza.

La funzione “accetta” semplicemente seleziona l’id dell’utente di cui si vuole accettare la richiesta, e in base a quell’id si va a modificare il campo “amministratore\_team” nella tabella **utenti**.

//controllo se è settato lo username

if(isset($\_GET['username'])){

$user=$\_GET['username'];

//preparo la query che rende l'utente amministratore di team

$query = mysqli\_query($db, "UPDATE utenti set amministratore\_team=1 WHERE

username='$user'");

//se la query ha successo

if($query){

//preparo la query che elimino la richiesta accettata

$query2= mysqli\_query($db,"DELETE FROM gestioneprogetti.richieste WHERE

username='$user'");

//se la query ha successo

if($query2){

//reindirizzo l'utente alla pagina delle richieste

header("Location: richieste.php");

}

}

}

La funzione “rifiuta” semplicemente esegue una query che elimina la richiesta dalla tabella **richieste.**

//query che elimina la richiesta dell'utente

$user2= mysqli\_query($db,"DELETE FROM gestioneprogetti.richieste WHERE

username='$user'");

### Logout

La funzione di logout è molto semplice:

session\_start();

//distruggo la sessione corrente

if(session\_destroy()) {

//reindirizzo l'utente al login

header("Location: login.php");

}

### Gestione funzionalità del sito per amministratori di team e membri

Gli amministratori di team e i membri hanno accesso allo spazio del sito dedicato ai teams (con progetti, tasks,..) e tutte le seguenti pagine avranno un controllo per fare in modo che solo amministratori di team e membri possono accedere a queste pagine.

//se l'utente in sessione è amministratore

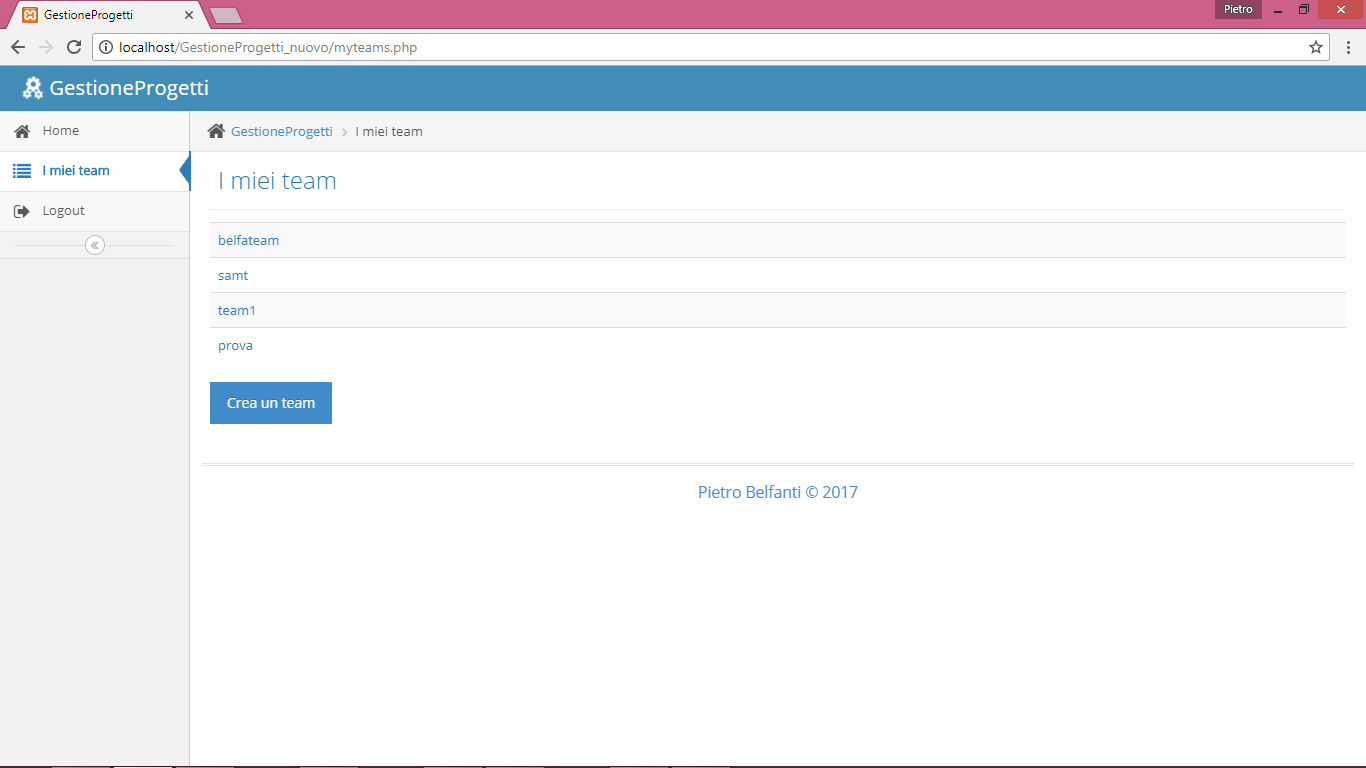
if($\_SESSION['is\_admin'] == 1){

//lo reindirizzo all'index

header('Location: index.php');

}

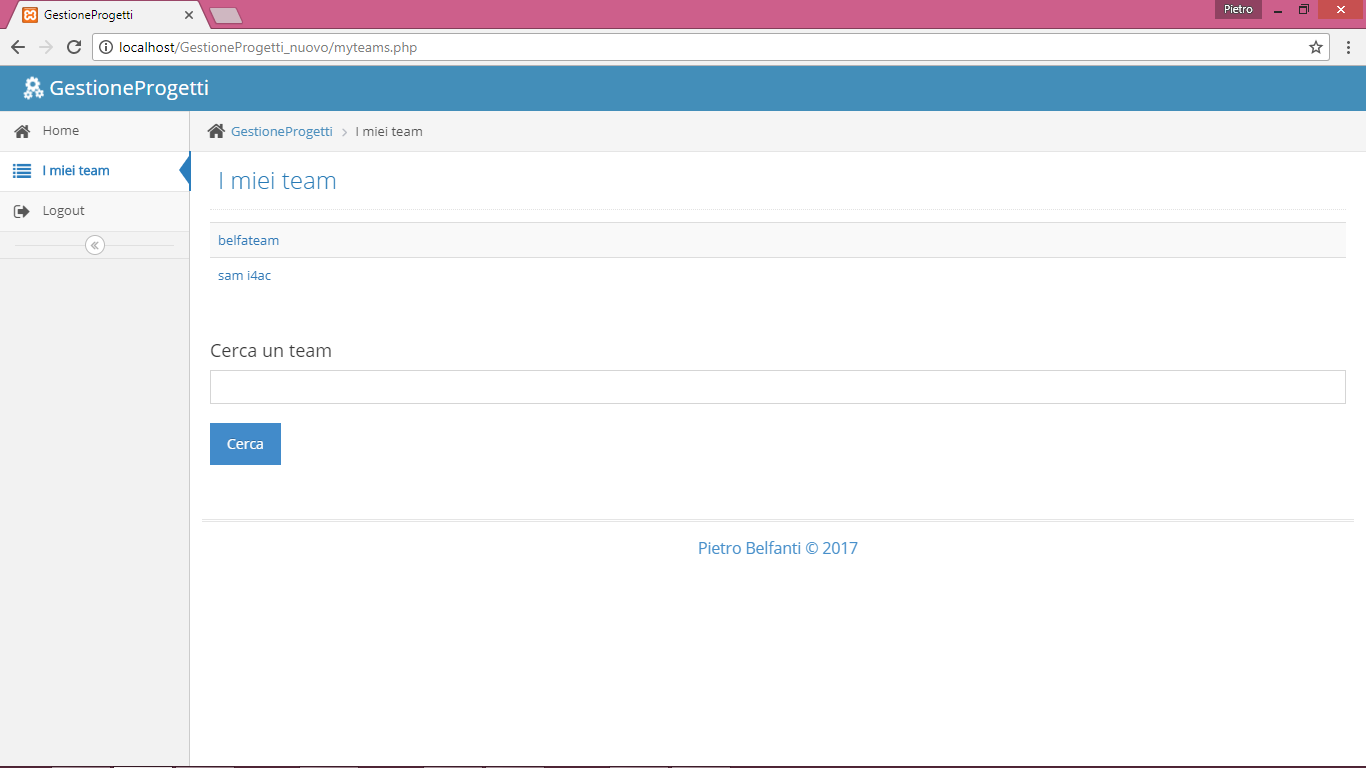
#### I miei team



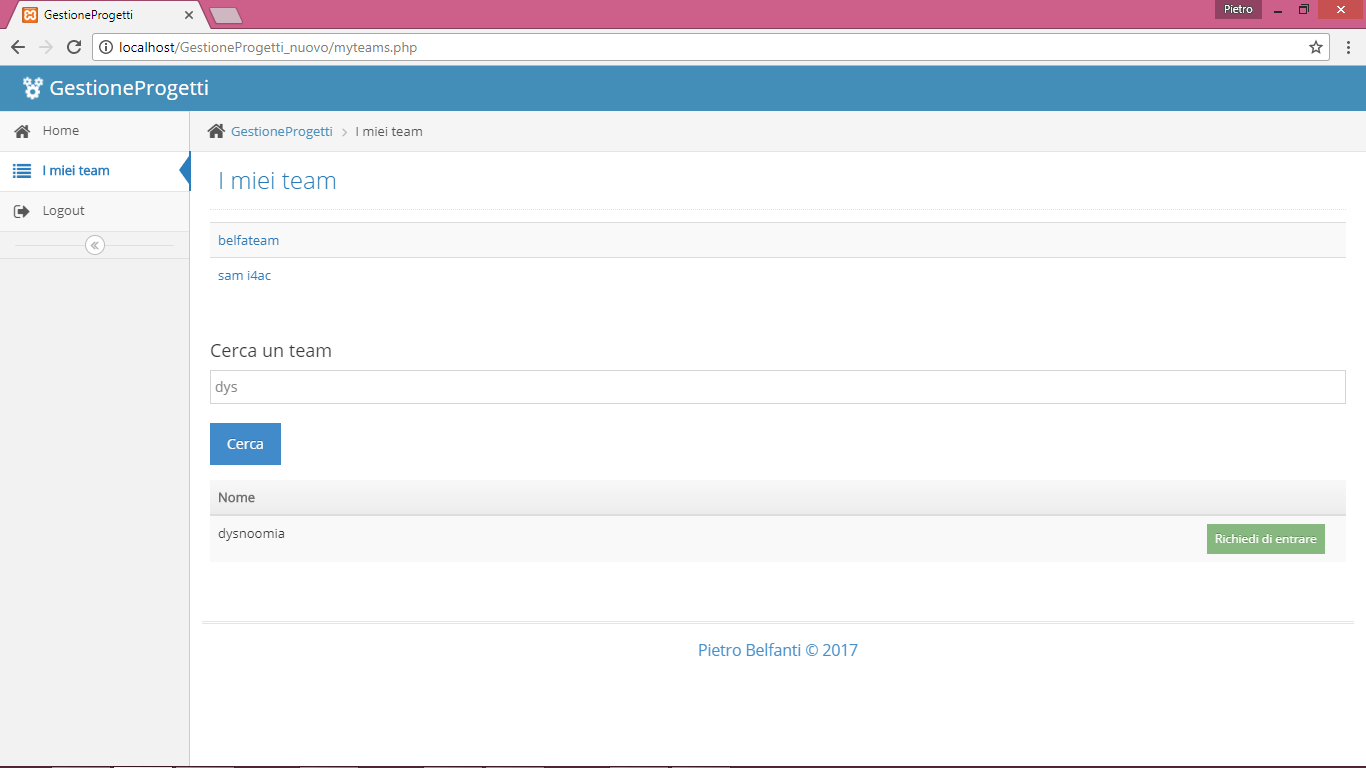
Per gli amministratori di team, la pagina dei team, si presenta in questo modo. Viene stampataa una tabella in base ai team a cui appartiene l’utente in sessione. In fondo c’è il bottone per creare un team.

La pagina di creazione di un team consiste semplicemente in un input di tipo testo in cui si deve inserire il nome del team, e cliccare sul bottone “Salva”. Quando si crea un team, viene controllato che il nome non sia già in uso da un altro team.

Per i membri, oltre alla tabella con i team di cui si fa parte, non c’è il bottone per creare il team ma bensì un form per cercare i team esistenti e eseguire una richiesta di entrata ai team.



Quando si cerca un team, viene preso l’input inserito (Es. “dys”) e vengono stampati in una tabella tutti i team con il nome che inizia per il valore inserito.



Nella tabella viene stampato il risultato della seguente query:

$result = mysqli\_query($db, "SELECT t.nome from teams t where t.nome LIKE

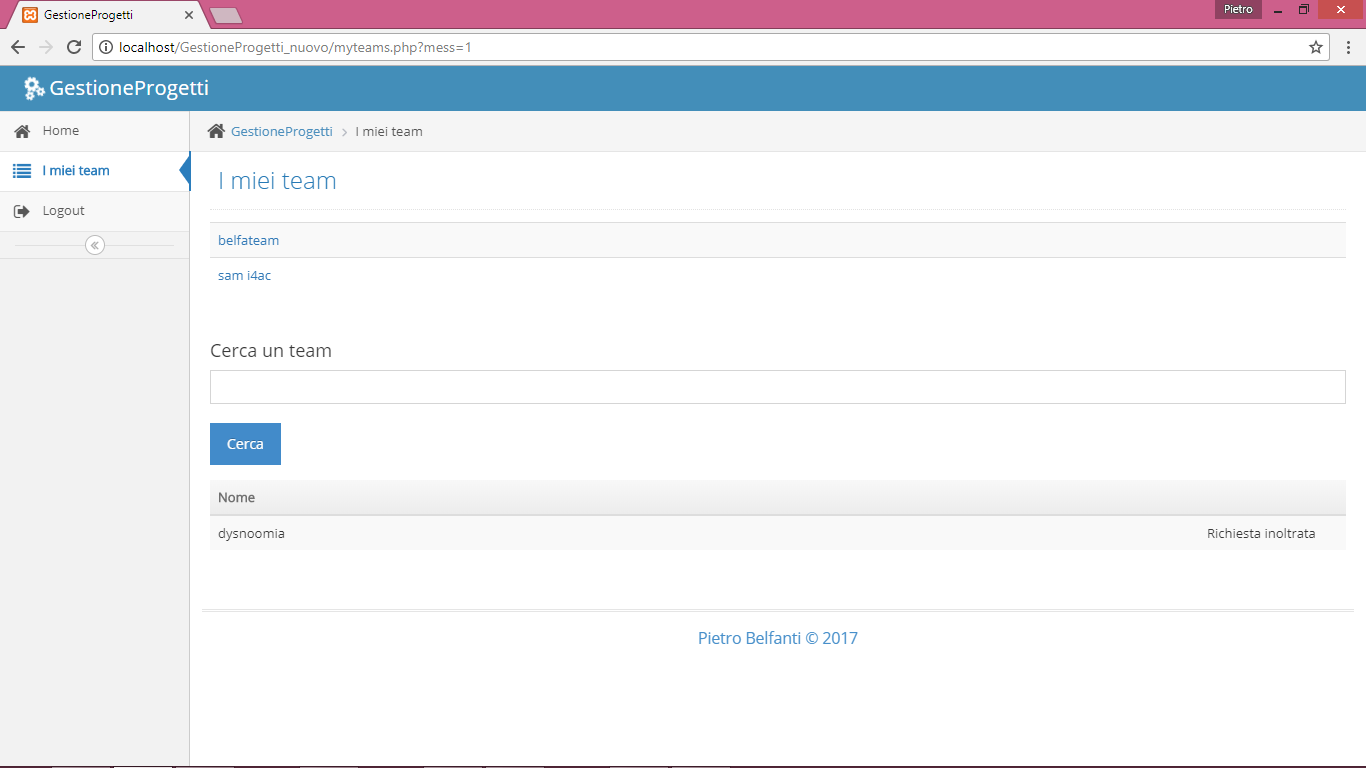
'{$name}%' AND t.id\_team NOT IN (Select a.id\_team

from appartenenza a where a.id\_utente = $id)");

Questa query seleziona il nome del team di cui l’utente in sessione non fa parte di cui il nome appunto inizia con il valore inserito nell’input.

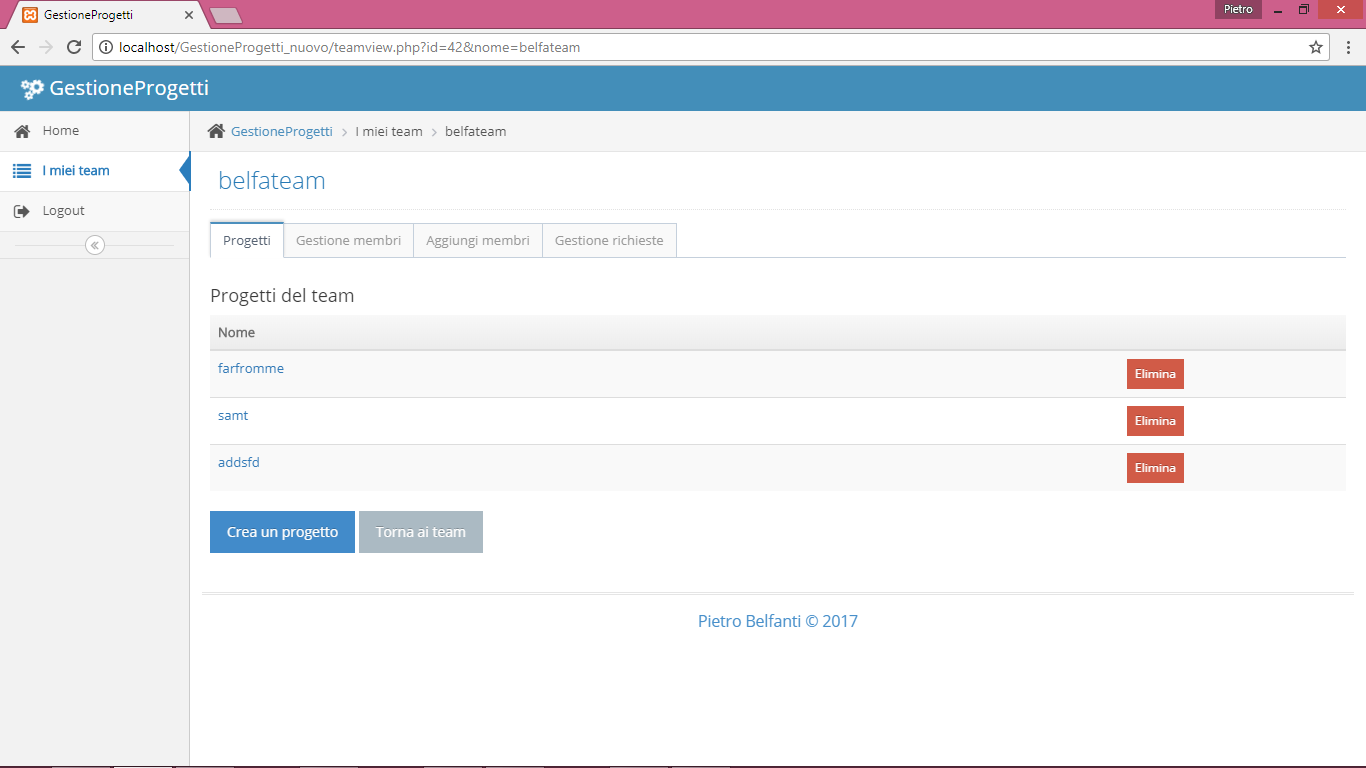
Quando si clicca su “richiedi di entrare”, viene eseguita una query che va a inserire la richiesta nella tabella **richieste\_team**.

Se in seguito, si ricerca lo stesso team, dopo aver fatto la richiesta, ho fatto in modo che invece del bottone “richiedi di entrare” esca la scritta “Richiesta inoltrata”.



Semplicemente prima di stampare la tabella, controllo se esiste una richiesta verso quel team, e se è presente non stampo il bottone ma la scritta.

#### Gestione del team

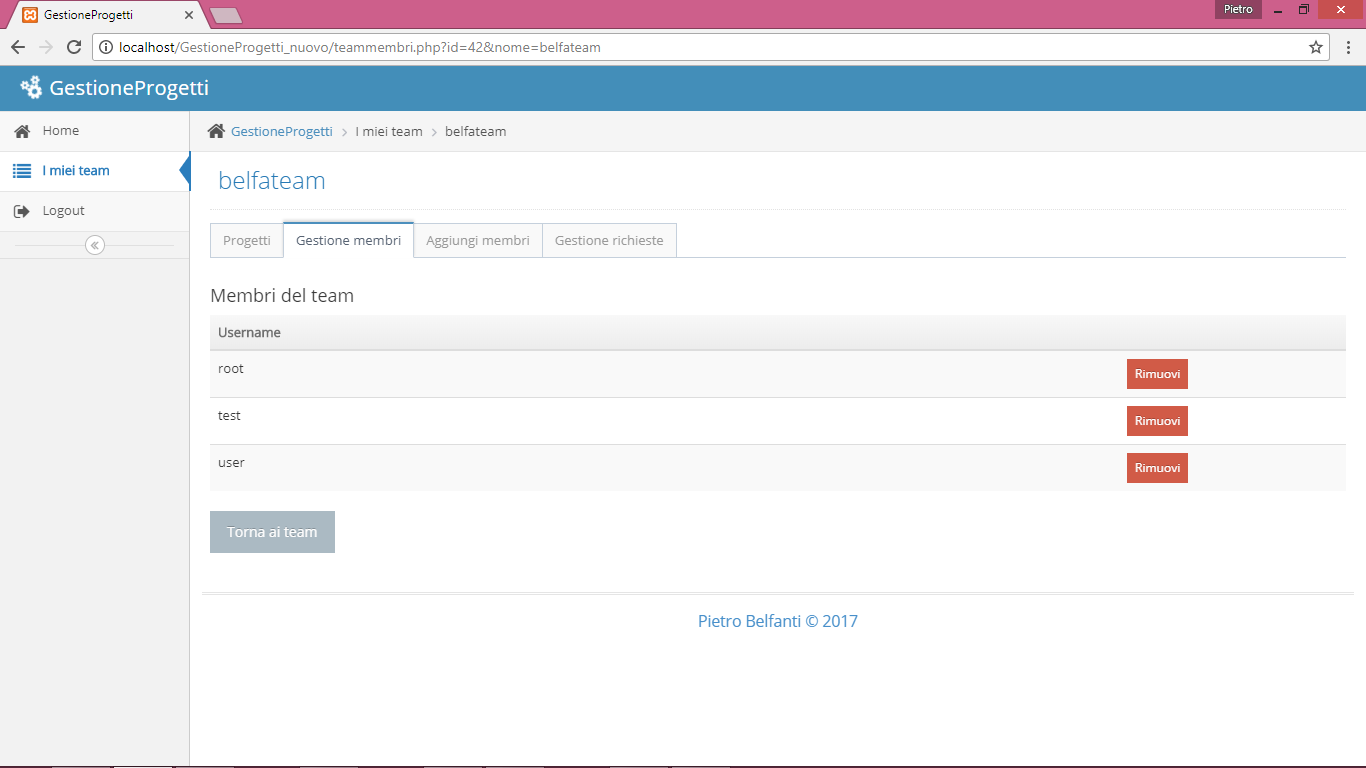


Dalla pagina dei team di cui si fa parte, cliccando su un team, si viene reindirizzati alla pagina di gestione del team. Questa pagina contiene il titolo (che è il nome del team in cui si è entrati). Contiene un menu di navigazione contenente 4 sezioni:

* Progetti: sezione in cui vengono visualizzati tutti i progetti del team. Per gli amministratori del team è possibile creare e eliminare i teams.
* Gestione membri: sezione in cui si possono rimuovere i membri dal team (accessibile solo agli amministratori del team).
* Aggiungi membri: sezione in cui si possono ricercare degli utenti e aggiungerli al team (accessibile solo agli amministratori del team).
* Gestione richieste: sezione in cui si possono accettare/rifiutare le richieste di entrata nel team da parte dei membri (accessibile solo agli amministratori del team).

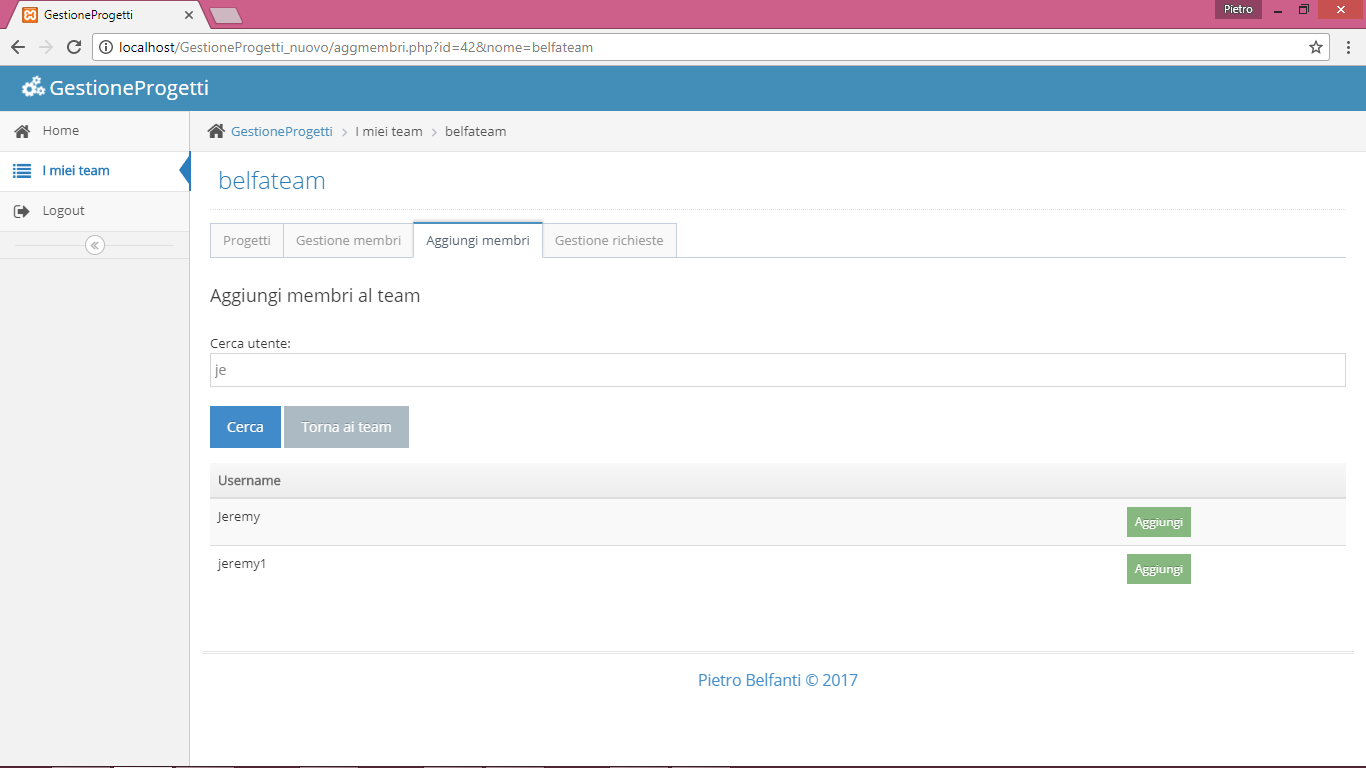
Nella sezione dei progetti, la funzione “Crea un progetto” è uguale alla funzione “Crea un team” nella pagina dei teams. La funzione “Elimina” è la solita funzione che va a eliminare dal database il progetto tramite una query.

La sezione di gestione dei membri si presenta nel seguente modo:



Vengono visualizzati sotto forma di tabella i membri che appartengono al team. C’è il bottone “Rimuovi” che fa la stessa identica cosa del bottone “Rimuovi” visto al capitolo **1.3.6.1**.

La sezione di aggiunta dei membri si presenta nel seguente modo:



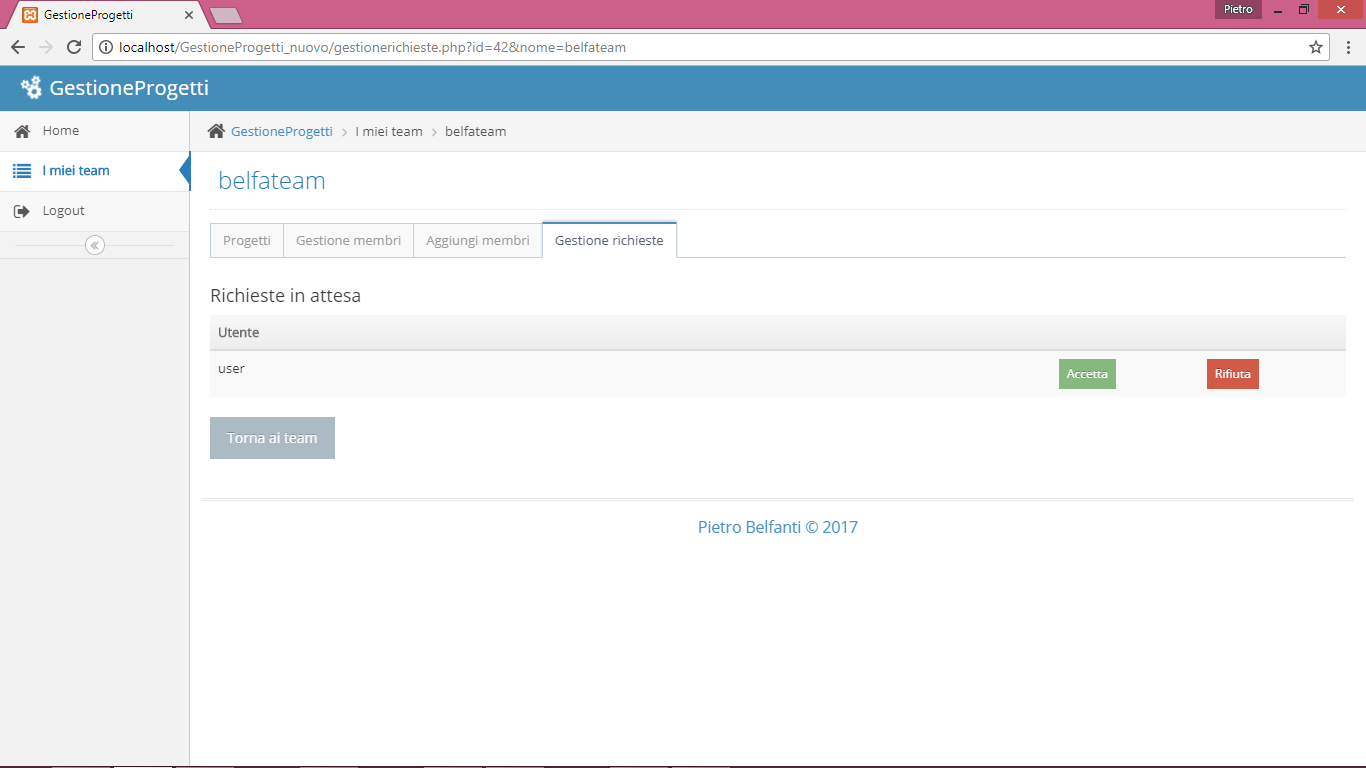
C’è lo stesso form di ricerca che era presente nella ricerca dei team da parte dei membri standard. La funzionalità è la stessa ma invece di cercare i team di cui non si fa parte, questa ricerca funziona in modo da cercare gli utenti che non fanno ancora parte del team e che abbiano lo username che inizia con il valore inserito nell’input.

La funzione “Agguingi” va a prendere l’id del team e dell’utente che si vuole aggiungere, e li inserisce nella tabella **appartenenza**.

$query = mysqli\_query($db, "INSERT INTO appartenenza (id\_utente, id\_team)

VALUES($user, $idteam);");

La pagina di gestione delle richieste si presenta nel seguente modo:



Viene visualizzata una tabella contenente le richieste (salvate nella tabella **richieste\_team**) del team in cui si è, e per ogni riga sono presenti i bottoni accetta/rifiuta richiesta.

Quando si accetta la richiesta, essa viene eliminata dalla tabella **richieste\_team** e inoltre viene inserito un record nella tabella **appartenenza** contenente id del team e dell’utente che si ha accettato.

$query = mysqli\_query($db, "INSERT INTO appartenenza (id\_utente, id\_team)

VALUES($user, $idteam);");

$query2 = mysqli\_query($db, "DELETE FROM richieste\_team

where id\_richiedente = $user AND team = $idteam";);

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

* Diari di lavoro
* Codici sorgente
* Guida per l’utilizzo
* QdC