

## **Società di Basket**

**Titolo del progetto:** Società di Basket  
**Alunno/a:** Simone Esposito  
**Classe:** Info 4  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Adriano Barchi

1	Introduzione.....	5
1.1	Informazioni sul progetto .....	5
1.2	Abstract.....	5
1.3	Scopo.....	5
2	Analisi.....	6
2.1	Analisi del dominio.....	6
2.2	Analisi e specifica dei requisiti .....	6
2.3	Use case.....	10
2.4	Pianificazione.....	1
2.5	Analisi dei mezzi .....	1
2.5.1	Software .....	1
2.5.2	Hardware.....	1
3	Progettazione, Implementazione e Test.....	2
3.1	Database .....	2
3.2	Connessione al Database.....	3
3.2.1	Introduzione .....	3
3.2.2	Progettazione – Diagramma UML.....	3
3.2.3	Spiegazione UML.....	3
3.2.4	Implementazione.....	3
3.2.5	Test Case .....	4
3.3	Sessioni .....	5
3.3.1	Introduzione .....	5
3.3.2	Progettazione – Diagramma UML.....	5
3.3.3	Spiegazione UML.....	5
3.3.4	Implementazione.....	5
3.3.5	Test Case .....	7
3.4	User .....	8
3.4.1	Introduzione .....	8
3.4.2	Progettazione – Diagramma UML.....	8
3.4.3	Spiegazione UML.....	8
3.4.4	Implementazione.....	9
3.4.5	Test Case .....	10
3.5	Account.....	12
3.5.1	Introduzione .....	12
3.5.2	Progettazione – Diagramma UML.....	12
3.5.3	Spiegazione UML.....	12
3.5.4	Implementazione.....	13
3.5.5	Test Case .....	14
3.6	Giocatore .....	15
3.6.1	Introduzione .....	15
3.6.2	Progettazione – Diagramma UML.....	15
3.6.3	Spiegazione UML.....	15
3.6.4	Implementazione.....	16
3.6.5	Test Case .....	17
3.7	Eventi.....	19
3.7.1	Introduzione .....	19
3.7.2	Progettazione .....	19
3.7.3	Implementazione.....	21
3.7.4	Test Case .....	24
3.8	Allenamento.....	27
3.8.1	Introduzione .....	27
3.8.2	Progettazione .....	27
3.8.3	Implementazione.....	29
3.8.4	Test Case .....	30
3.9	Esercizi .....	33
3.9.1	Introduzione .....	33
3.9.2	Progettazione .....	33
3.9.3	Implementazione.....	36

3.9.4	Test Case.....	37
3.10	Palestra.....	40
3.10.1	Introduzione .....	40
3.10.2	Progettazione .....	40
3.10.3	Implementazione.....	41
3.10.4	Test Case.....	41
3.11	Partita.....	44
3.11.1	Introduzione .....	44
3.11.2	Progettazione .....	44
3.11.3	Implementazione.....	46
3.11.4	Test Case.....	49
3.12	Risultato.....	52
3.12.1	Introduzione .....	52
3.12.2	Progettazione .....	52
3.12.3	Implementazione.....	54
3.12.4	Test Case.....	56
3.13	Categorie e Lavori .....	58
3.13.1	Introduzione .....	58
3.13.2	Progettazione .....	58
3.13.3	Implementazione.....	59
3.13.4	Test Case.....	60
3.14	Annuncio.....	61
3.14.1	Introduzione .....	61
3.14.2	Progettazione .....	61
3.14.3	Grafica.....	61
3.14.4	Implementazione.....	62
3.14.5	Test Case.....	63
3.15	Email.....	65
3.15.1	Introduzione .....	65
3.15.2	Progettazione .....	65
3.15.3	Implementazione.....	65
3.15.4	Test Case.....	66
3.16	Menu.....	67
3.16.1	Introduzione .....	67
3.16.2	Progettazione .....	67
3.16.3	Implementazione.....	68
3.16.4	Test Case.....	69
3.17	Main.....	70
3.17.1	Introduzione .....	70
3.17.2	Progettazione .....	70
3.17.3	Implementazione.....	71
3.17.4	Test Case.....	71
3.18	Login e Logout.....	72
3.18.1	Progettazione .....	72
3.18.2	Implementazione.....	72
3.18.3	Test Case.....	73
3.19	Registrazione.....	74
3.19.1	Introduzione .....	74
3.19.2	Progettazione .....	74
3.19.3	Test Case.....	75
3.20	Scheda.....	76
3.20.1	Introduzione .....	76
3.20.2	Progettazione .....	76
3.20.3	Implementazione.....	77
3.20.4	Test Case.....	78
3.21	Report.....	79
3.21.1	Introduzione .....	79
3.21.2	Progettazione.....	79

3.21.3	Implementazione.....	79
3.21.4	Test Case.....	79
3.22	Statistiche .....	80
3.22.1	Introduzione .....	80
3.22.2	Progettazione .....	80
3.22.3	Implementazione.....	81
3.22.4	Test Case.....	81
4	Test dei Requisiti.....	82
4.1	Risultati dei Test Case.....	93
5	Consuntivo.....	96
6	Conclusioni .....	1
6.1	Sviluppi futuri .....	1
6.2	Considerazioni personali .....	1
7	Bibliografia.....	1
7.1	Sitografia.....	1
8	Allegati.....	1

## **1 Introduzione**

---

### **1.1 Informazioni sul progetto**

Allievo coinvolto: Simone Esposito

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

Docenti responsabili: Adriano Barchi

Data inizio: 30 / 08 / 2016

Data fine: 22 / 12 / 2016

### **1.2 Abstract**

*A society of Basket needs a web site to represent themselves and for help to manage all works they need to do. This project has the purpose to realize what they need, all done with we learned in three year here in the professional school. The entire site is realized in the following language code: sql, php, css and html.*

### **1.3 Scopo**

Questo progetto ha lo scopo principale di prepararci al progetto finale di fine apprendistato. Per farlo dobbiamo utilizzare quanto appreso nell'intera formazione, in quanto tocca molti punti visti nei vari moduli, come ad esempio la realizzazione di un database, oppure progettazione di diagrammi. Come scopo secondario è quello di poter partire da un bisogno di un cliente o una società, capirne quanto necessita e poi realizzare concretamente tutto ciò che necessitano ed è fattibile.

## 2 Analisi

### 2.1 Analisi del dominio

Fino a ora la società gestiva tutto quanto a mano, dunque si cerca un modo per risolvere il tutto rendendolo più automatico. Lo scopo di questo progetto è proprio quello di permettere ai gestori, di entrare, ovunque essi siano, all'interno del sito e gestire quanto si necessita. Attualmente la società di basket utilizza un sito solamente per presentarsi e nulla di più.

### 2.2 Analisi e specifica dei requisiti

ID: REQ-001	
<b>Nome</b>	Realizzare un sito che permetta la gestione di una società di Basket
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Il sito deve presentare le classiche pagine con le informazioni e presentazioni della società
<b>002</b>	Il sito dovrà prevedere zone pubbliche e altre private
<b>003</b>	Il sito dovrà essere fruibile su un hosting esterno alla scuola
<b>004</b>	Il sito dovrà prevedere una pagina di login che ti permette di accedere alle aree private del sito

ID: REQ-002	
<b>Nome</b>	Realizzare delle pagine pubbliche visibili anche dagli ospiti
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Il sito dovrà avere delle pagine di presentazione e informazioni generali della società
<b>002</b>	Deve esserci una pagina dove poter contattare la società
<b>003</b>	Deve esserci una pagina dove poter consultare il calendario degli eventi e delle partite
<b>004</b>	Dovrà esserci una pagina per le partite fatte e le fotografie collegate

ID: REQ-003	
<b>Nome</b>	Utenti
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Bisogna prevedere più livelli di accesso e dunque utenti differenti
<b>002</b>	Gli utenti sono i seguenti: ospite, utente semplice, allenatore, blogger, segretario, gestore e amministratore
<b>003</b>	L'amministratore ha la facoltà di poter modificare/creare/cancellare i vari utenti ed assegnarli.
<b>004</b>	Ogni utente, oltre ha anche un ruolo all'interno della società di basket (Giocatore, Staff, Allenatore, Supporters, Arbitro e Commissario di campo)
<b>005</b>	Per i giocatori si vogliono sapere le informazioni base

ID: REQ-004	
<b>Nome</b>	Pagine Private
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Questo requisito è la lista delle pagine accessibili solamente con login e per ogni pagina vi è segnato quale utente può accederci.
Sotto requisiti	
<b>001</b>	Gestione utenti (modifica/aggiunta/cancellazione assegnazione ruolo e registrazione nuovo utente) visibile solo dall'Amministratore.
<b>002</b>	Gestione eventi (aggiunt/modifica/cancellazione eventi o partite) visibile dal segretario, allenatore, gestore e amministratore.
<b>003</b>	Gestione partita (aggiunta/modifica/cancellazione convocazioi/arbitri/allenatori/luogo/ora/ritrovo) visibile dal segretario, allenatore, gestore e amministratore
<b>004</b>	Gestione pagamenti (modifica lista di chi ha pagato) visibile da segretario, gestore e amministratore
<b>005</b>	Gestione allenamenti (modifica/aggiunta/cancellazione di esercizi/allenamenti/luogo/orario) visibile da gestore, amministratore e allenatore
<b>006</b>	Gestione pagina report (cancellazione/modifica/aggiunta) visibile da amministratore, (aggiunta) visibile da gestore
<b>007</b>	Gestione annunci (cancellazione/modifica/aggiunta e invio degli annunci) visibile da segretario, gestore, allenatore e amministratore
<b>008</b>	Gestione palestra (cancellazione/modifica/aggiunta di unapalestra) visibile da gestore e amministratore
<b>009</b>	Gestione scheda personale dell'utente (modifica/aggiunta/cancellazione delle informazioni sull'utente) visibile dall'utente stesso (solo la propria) e dal gestore (di tutti gli utenti)
<b>010</b>	Gestione pagina delle statistiche sui vari giocatori e sulla società visibile dall'amministratore
<b>011</b>	Gestione impostazioni sito (aggiunta/modifica/cancellazione elementi dalle liste chi fa cosa/ruoli/categorie e impostazioni base) visibile dal gestore e dall'amministratore
<b>012</b>	Pagina in cui commentare le partite e aggiungere le foto visibile dal blogger

ID: REQ-005	
<b>Nome</b>	Interfaccia grafica del sito
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Il sito deve essere intuibile e facilmente navigabile
<b>002</b>	La società deve essere presentata anche graficamente
<b>003</b>	Si deve rispettare il tema del basket

ID: REQ-006	
<b>Nome</b>	Annunci
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Solamente gli utenti registrati ricevono gli annunci (partite, riunioni, eventi e comunicazioni varie)
<b>002</b>	Gli annunci sono personalizzabili dai vari utenti, ovvero se si ha bisogno di un messaggio per ricordare l'impegno ogni periodo e il metodo di ricezione
<b>003</b>	Gli annunci vanno fatti via email, sms e/o whatsapp

ID: REQ-007	
<b>Nome</b>	Allenamenti
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Un allenatore può preparare i propri allenamenti, inserendo pure i vari esercizi (nuovi o vecchi)
<b>002</b>	Un allenatore ha la possibilità di modificare l'allenamento (luogo, orario e altre informazioni)
<b>003</b>	A fine allenamento l'allenatore deve poter inserire la lista dei presenti/assenti
<b>004</b>	A fine allenamento l'allenatore deve poter inserire un commento all'allenamento
<b>005</b>	Bisogna prevedere un modo per prendere le varie statistiche degli allenamenti.

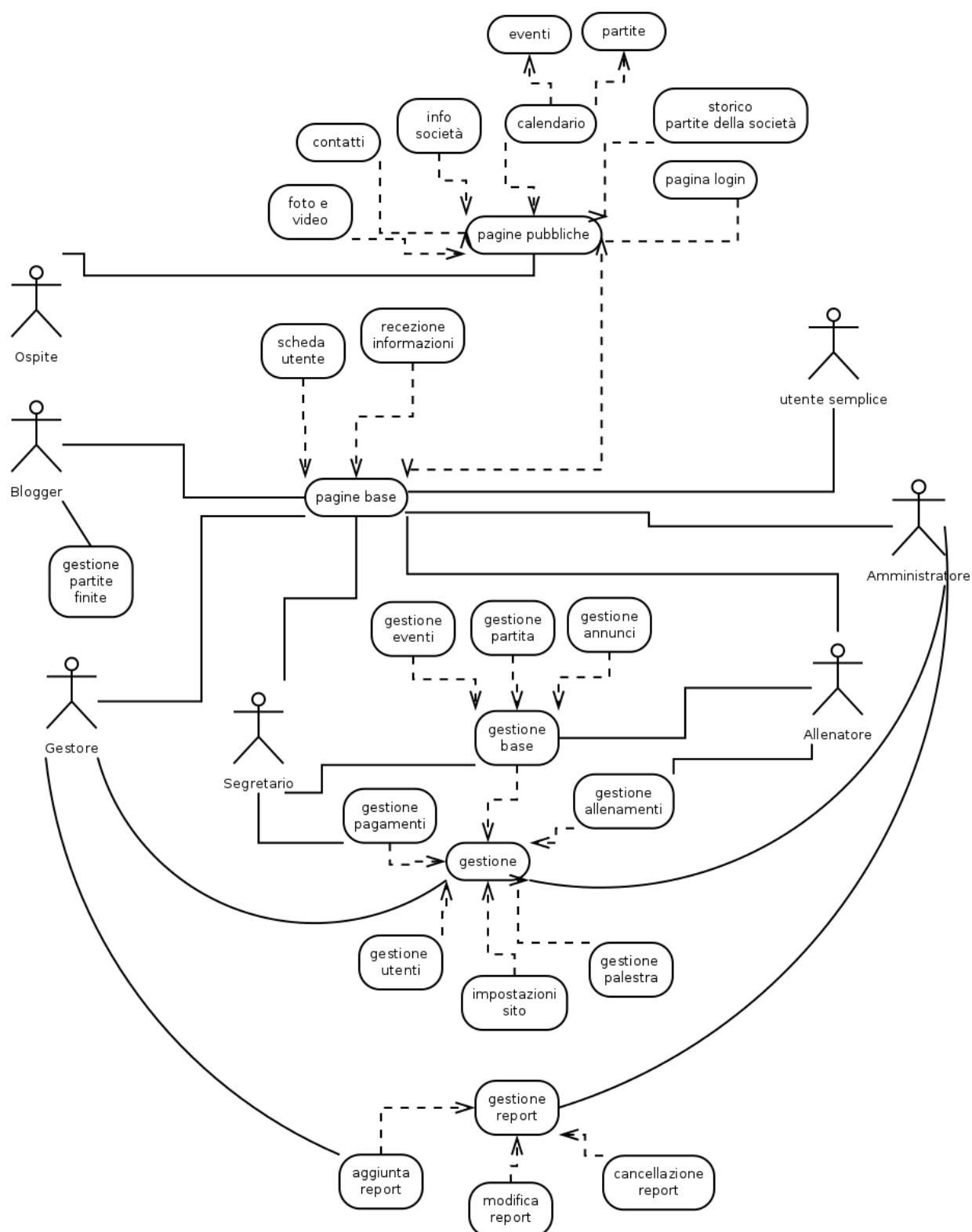


ID: REQ-008	
<b>Nome</b>	Report
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Bisogna prevedere una pagina dove poter preparare e stampare i report
<b>002</b>	I report stampabili devono essere: Liste membri (con possibilità di filtro), foglio di partita compilato, convocazioni e foglio allenamenti

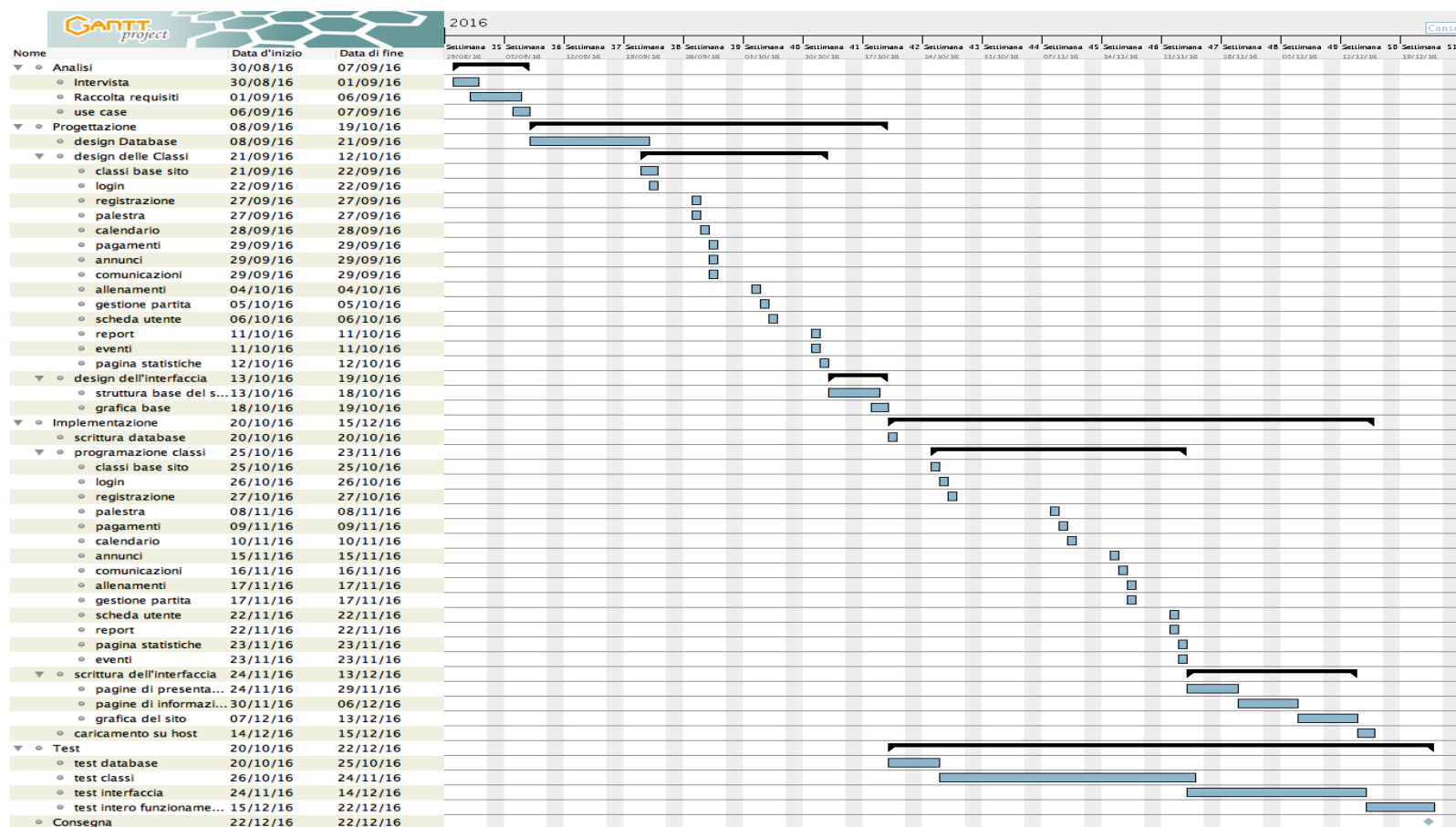
ID: REQ-009	
<b>Nome</b>	Calendario
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Deve esserci un calendario con tutti gli impegni della società
<b>002</b>	Se l'impegno è una partita, bisogna mostrare le informazioni base: Categoria, data, luogo, palestra e i vari ruoli degli accompagnatori
<b>003</b>	Il calendario è gestibile dal gestore e dall'amministratore

ID: REQ-010	
<b>Nome</b>	Partite
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	L'allenatore, il gestore o l'amministratore ha la facoltà di creare una partita e modificarne le varie informazioni (ora e luogo di ritrovo, allenatori, accompagnatori e arbitro)
<b>002</b>	Per ogni partita bisogna prevedere una lista convocati.
<b>003</b>	Per ogni convocato a quella partita bisogna poter preparare il tagliando di convocazione
<b>004</b>	Per ogni partita si tiene conto delle varie statistiche (canestri fatti, tiri fatti, ingressi dei giocatori)

## 2.3 Use case



## 2.4 Pianificazione



**Titolo del progetto:** Società di Basket  
**Alunno/a:** Simone Esposito  
**Classe:** Info 4  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Adriano Barchi

## **2.5 Analisi dei mezzi**

### **2.5.1 Software**

Per realizzare questo progetto ho utilizzato le seguenti librerie e con la loro annessa versione:

- MAMP – 3.0.7.3: per fare da web server alla macchina fisica. Questo pacchetto comprende php, mysql e phpmyadmin tutto preinstallato.
- Chart.js – 1.0.2: per la realizzazione dei grafici della pagina statistiche.php. Questa libreria scritta in javascript, offre la possibilità di creare facilmente dei grafici e di implementarli all'interno delle proprie pagine.
- FPDF – 1.81: questa libreria, scritta in php, offre la possibilità di creare dei file pdf online. Dunque ha tutto ciò che serve per realizzare un documento di quest'estensione, utilizzando dei comandi php. Comodo per realizzare i report stampabili.
- Bootstrap – 3.3.1: è stato utilizzato per tutto ciò che riguarda la grafica del sito, in quanto offre degli elementi già preconfezionati per il responsive e per essere graficamente puliti.
- Fullcalendar.io – 3.0.1: questa libreria javascript serve alla creazione di calendari dinamici, utilizzato per la visualizzazione degli eventi.

### **2.5.2 Hardware**

Dato che il progetto è interamente un sito web, non ho dovuto utilizzare delle macchine specifiche per fare funzionare il tutto. Il progetto è stato realizzato su un MacBook Pro del 2015, con un sistema operativo OS X El Capitan.

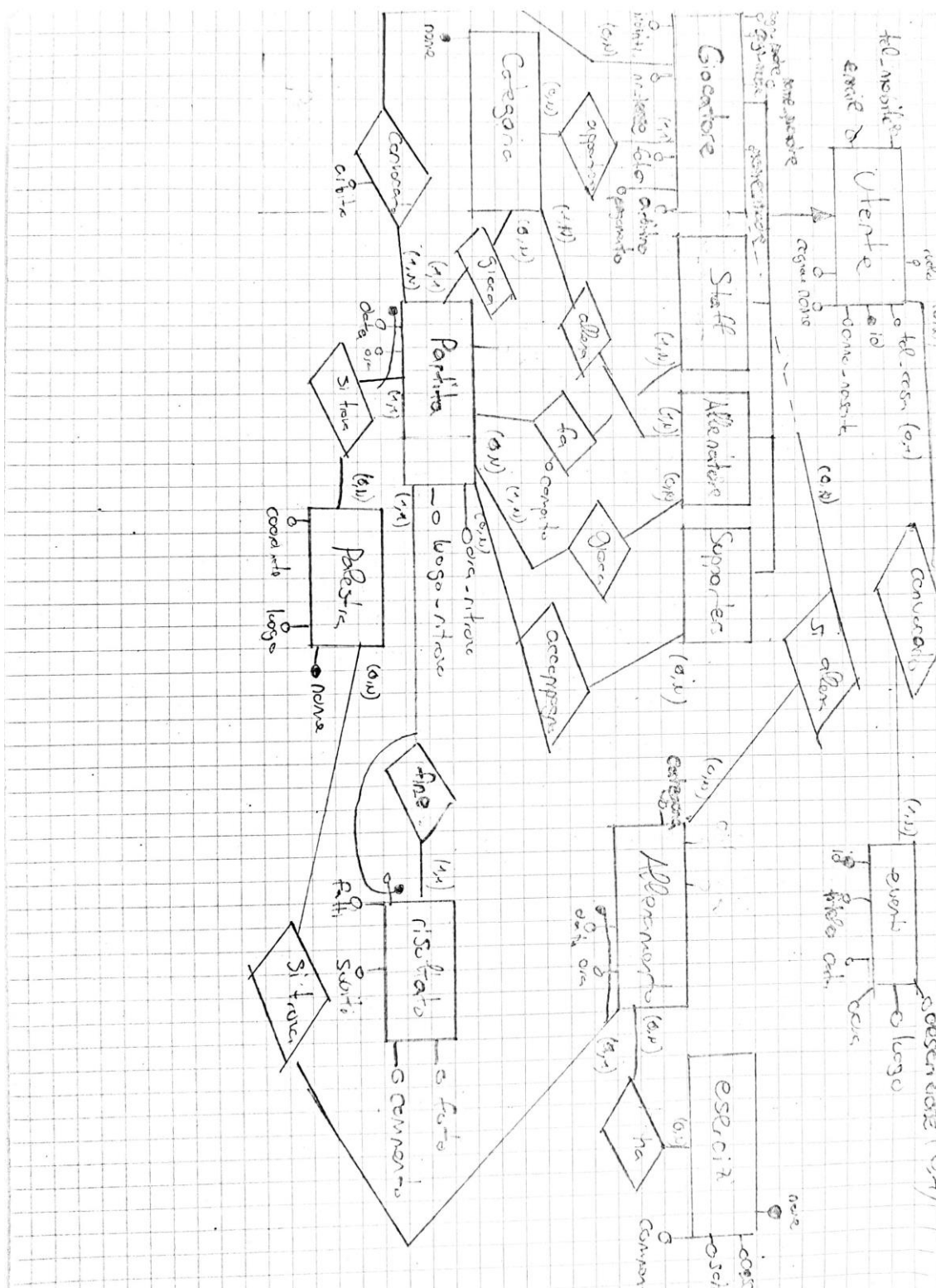
Il sito è stato inoltre pubblicato su un host, queste sono le credenziali:

- FTP:
  - User: gbasket
  - Pass: GestBasket\_Admin
  - Host: samtinfo.ch/gbasket
- Mysql:
  - User: gbasket
  - Password: GestBasket\_Admin
  - Database: samtinfoch38
  - Host: mysql.samtinfo.ch

**Titolo del progetto:** Società di Basket  
**Alunno/a:** Simone Esposito  
**Classe:** Info 4  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Adriano Barchi

### 3 Progettazione, Implementazione e Test

### 3.1 Database

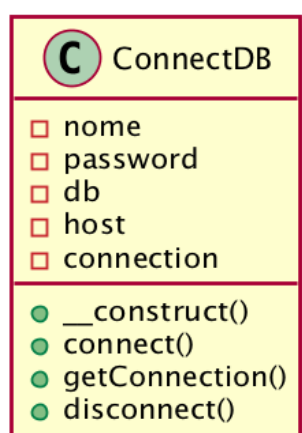


## 3.2 Connessione al Database

### 3.2.1 Introduzione

Questa classe è stata pensata per le connessioni con il database. Quindi essa ha lo scopo di interagire con il db e ritornare le varie connessioni, in modo da poterle utilizzare all'interno del sito, senza dover implementare la logica ad ogni pagina.

### 3.2.2 Progettazione – Diagramma UML



### 3.2.3 Spiegazione UML

- **nome**: nome dell'account con la quale effettuiamo l'accesso al db
- **password**: password dell'account con la quale effettuiamo l'accesso al db
- **db**: nome del nostro db
- **host**: nome dell'host che ospita il nostro db
- **connection**: oggetto connessione
- **\_\_construct()**: costruttore che richiama connect()
- **connect()**: esegue la connessione al db e setta la variabile connection con un oggetto connessione
- **getConnection()**: ritorna la variabile connection
- **disconnect()**: setta a null la variabile connection

### 3.2.4 Implementazione

La parte più importante della classe è il metodo **connect()** che esegue la connessione e **getConnection()** che ritorna la connessione effettuata. Grazie a questi due metodi possiamo gestire tutto il sito e le connessioni al database. Vediamo dunque il cuore di questa classe nel dettaglio

### 3.2.4.1 Connect()

Questo metodo, come spiegato in precedenza, si occupa di effettuare la connessione e salvarla all'interno della variabile connection.

```
/**
 * Metodo per eseguire manualmente la connessione
 */
public function connect() {
    if ($this->connection != null) {
        $this->connection = new mysqli($this->host, $this->nome, $this->password, $this->db) or die("error");
    }
}
```

### 3.2.4.2 getConnection()

Questo metodo si occupa semplicemente di prendere la variabile connection della classe e ritornarla.

```
/**
 * Metodo che ritorna l'oggetto connection
 */
public function getConnection() {
    if ($this->connection) {
        return $this->connection;
    } else {
        return false;
    }
}
```

### 3.2.5 Test Case

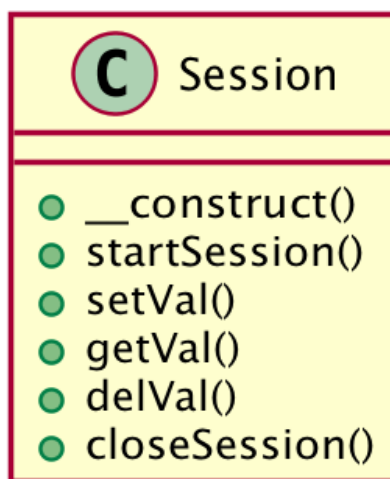
<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo che quando creo un nuovo oggetto ConnectDB, mi valorizza l'attributo connection richiamando getConnection()	Mi aspetto che utilizzando getConnection() posso eseguire la seguente query "select nome from account" senza errori.	Viene eseguita correttamente
2	Controllo che la funzione disconnect() interrompa l'attuale connessione	Mi aspetto che dopo aver chiamato la funzione disconnect() non sia più possibile eseguire una query, ovvero che ritorni un errore	Ritorna correttamente l'errore
3	Controllo che dopo la funzione disconnect() posso richiamare connect() e utilizzarla per una query	Mi aspetto che utilizzando getConnection() posso eseguire la seguente query "select nome from account" senza errori.	Viene eseguita correttamente

### 3.3 Sessioni

#### 3.3.1 Introduzione

Questa classe è pensata per lavorare in modo molto intuitivo e semplice con le sessioni, in modo da non dover sempre pensare alle piccolezze, che magari causano errori nel tempo, ma gestisce tutto ciò che serve richiamando semplicemente la classe. Inoltre offre la possibilità di avere metodi per salvare/recuperare le variabili all'interno della sessione.

#### 3.3.2 Progettazione – Diagramma UML



#### 3.3.3 Spiegazione UML

- `__construct()`: il costruttore richiama il metodo `startSession()`
- `startSession()`: questo metodo si occupa di startare le sessioni, in modo da poterci lavorare
- `setVal()`: riceve come parametri il nome della sessione e il valore, li salva all'interno di una variabile sessione
- `getVal()`: riceve come parametro il nome della sessione e ritorna il suo valore
- `delVal()`: riceve come parametro il nome della sessione da cancellare
- `closeSession()`: chiude e distrugge la sessione, utile per i logout

#### 3.3.4 Implementazione

Reputo che la parte più importante di questa classe sia il metodo `startSession()` che viene chiamato dal costruttore immediatamente. Molto utile, almeno non bisogna ricordarsi sempre in ogni pagina di startarle manualmente, ma basta creare un'oggetto di tipo `Session`. Inoltre i due metodi `set` e `get Val` sono molto intuitivi, utili per lavorare in qualsiasi progetto. Sono facilmente utilizzabili e funzionali



### 3.3.4.1 startSession()

Questo metodo si occupa solamente di startare una nuova session, altrimenti non sarebbero utilizzabili all'interno della pagina corrente. Però prima di startarne una nuova, controlla che non sia già stato fatto da qualche altra parte.

```
/**
 * Starto le sessini
 */
public function startSession(){
    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }
}
```

### 3.3.4.2 setVal()

Questo metodo riceve come parametri \$nome che sarà il nome della sessione e \$valore ovvero il suo valore.

```
/**
 * metodo che setta il valore
 * Param nome: nome della session, valore: valore della sessione
 */
public function setVal($nome, $valore){
    $_SESSION[$nome] = $valore;
}
```

### 3.3.4.3 getVal()

Riceve come unico parametro \$nome, esso è il nome della sessione che prende e ritorna, nel caso in cui è settata, altrimenti ritorna false.

```
/**
 * metodo che ritorna il valore di una sessione
 * Param nome: nome della sessione da ritornare
 */
public function getVal($nome){
    if(isset($_SESSION[$nome])){
        return $_SESSION[$nome];
    }else{
        return false;
    }
}
```

### 3.3.5 Test Case

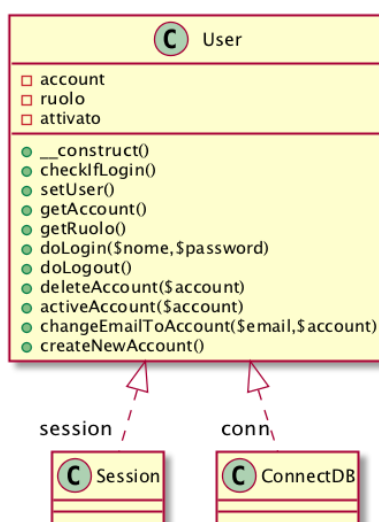
<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo che il costruttore funzioni correttamente	Al settaggio di un nuovo oggetto Session, il session_start() venga eseguito correttamente	Funziona correttamente secondo le aspettative
2	Controllo che se chiamo più volte il costruttore, non c'è nessun errore	Se chiamo più volte il costruttore, non deve esserci nessun errore di sessioni già attive, ma venga gestito correttamente	Viene gestito correttamente
3	Controllo che setVal setti correttamente il valore	Chiamando setVal e passandogli i due parametri, controllo che chiamando manualmente quella sessione, il valore sia settato correttamente	Viene settato correttamente
4	Controllo che getVal ritorni il valore della sessione	Passando il nome della sessione a getVal mi aspetto che ritorni correttamente il suo valore nel caso in cui sia stato settato in precedenza, altrimenti che ritorni false	Ritorna correttamente il valore in entrambi i casi
5	Controllo che la funzione delVal cancelli correttamente la sessione passata come parametro.	Passando il nome della variabile alla funzione delVal, mi aspetto che venga cancellata solamente quella e non tutte le altre	Viene cancellata correttamente
6	Mi aspetto che closeSession cancelli tutte le sessioni attuali	Chiamando la funzione closeSession voglio che tutte le sessioni vengano cancellate	Vengono cancellate correttamente

## 3.4 User

### 3.4.1 Introduzione

Questa classe ha il compito di gestire tutto ciò che concerne l'utente, sia quanto riguarda il login e logout, sia eventuali cambi di email, sia crearne uno nuovo o attivare l'account. Per far funzionare correttamente questa classe, essa implementa le altre due: ConnectDB e Session. Grazie a queste due la classe effettua i controlli e cambiamenti all'interno del DB e, ad esempio nel caso del login, setta poi le variabili di sessione che le pagine utilizzeranno per il corretto funzionamento delle varie logiche.

### 3.4.2 Progettazione – Diagramma UML



### 3.4.3 Spiegazione UML

- Account: nome dell'account con la quale effettuiamo l'accesso
- Ruolo: ruolo che ha l'account (utente normale, scrittore, allenatore, segretario...)
- Attivato: se l'account è già stato attivato o meno (se ha già effettuato il login)
- \_\_construct(): vengono salvate le due variabili conn e session
- checkIfLogin(): viene controllato se l'utente ha effettuato il login; ritorna true nel caso in cui è loggato e false nel caso in cui non lo è
- setUser(): questo metodo setta i tre valori, account, ruolo e attivato, dell'account. Per farlo fa prima un controllo se l'utente è loggato o meno. Ritorna false se è sloggato e altrimenti true
- getAccount(): ritorna il valore salvato in account
- getRuolo(): ritorna il valore salvato in ruolo
- getAttivato(): ritorna il valore salvato in attivato
- doLogin(): dopo aver cifrato in md5 il parametro password, fa un controllo assieme al nome dell'account, primo parametro, all'interno del database per vedere se esistono quei valori. Se esistono tira fuori le informazioni utili: account, ruolo e attivato che salva nelle rispettive variabili
- doLogout(): distrugge la sessione attuale, effettuando così il logout
- deleteAccount(): riceve come parametro il nome dell'account da cancellare all'interno del database
- activeAccount(): attiva l'account passato come parametro
- changeEmailToAccount(): cambia l'email che riceve come secondo parametro all'utente con il nome passato come primo parametro

- createNewAccount(): Crea un nuovo record nella tabella account all'interno del database

### 3.4.4 Implementazione

Reputo tre metodi molto importanti per questa classe, o meglio dire quelli a cui dare più importanza e maggiore attenzione, in quanto sono alla base di essa. Essi sono doLogin(), setUser() e activeAccount(), che implementano un po' tutto ciò che serve all'utente per navigare all'interno delle varie pagine. Altri metodi come deleteAccount o changeEmail sono molto simili, dunque evito di riportarli.

#### 3.4.4.1 doLogin()

Questo metodo accetta due parametri: account e password. Il secondo di questi, viene subito cifrato con md5, ovvero il metodo di cifratura con cui salviamo le password all'interno del db. Il primo dei due serve, assieme al secondo cifrato, a cercare nel database una corrispondenza. Se essa viene trovata, quindi esiste un record che ha il nome e la password uguali a quelle messe come parametro, allora vengono salvate tre nuove variabili di sessione: account, ruolo e attivato.

```
/**
 * Controllo se account e password sono corretti, se si setto le sessioni e ritorno true, altrimenti torno fa
 * Param: account: nome account, password: password dell'account
 */
public function doLogin($account,$password){
    $password = md5($password);
    $query = mysqli_query($this->conn, "select account,ruolo,attivato from account where account = '$account'");
    if($query->num_rows>0){
        $query = $query->fetch_assoc();
        $this->session->setVal("account",$query['account']);
        $this->session->setVal("ruolo",$query['ruolo']);
        $this->session->setVal("attivato",$query['attivato']);
        return true;
    }else{
        return false;
    }
}
```

#### 3.4.4.2 setUser()

Questo metodo viene richiamato per estrapolare dalle sessioni le tre informazioni utili, così che nelle varie pagine si possa mostrare il contenuto appropriato ad ogni utente. Come possiamo vedere, prima viene effettuato un controllo se l'utente ha fatto il login, se sì, allora salviamo le tre variabili all'interno di tre omonime di classe.

```
public function setUser(){
    if($this->checkIfLogin()){
        $this->account = $this->session->getVal("account");
        $this->ruolo = $this->session->getVal("ruolo");
        $this->attivato = $this->session->getVal("attivato");
        return true;
    }else{
        return false;
    }
}
```

### 3.4.4.3 activeAccount()

Riceve come paramtro l'account a cui cambiare il valore di attivato a 1. Questo perché se l'utente ha attivato a 0, allora appena effettua il login sarà costretto a inserire tutti i valori utili prima di poter procedere (come esempio numero telefono, via, ecc)

```
public function activeAccount($account){
    $query = $this->conn->prepare("UPDATE account set attivato=1 where account=?");
    $query->bind_param("s",$account);
    $query->execute();
}
```

### 3.4.5 Test Case

Nr Test	Descrizione	Aspettative	Risultato
1	Implementando un oggetto user non deve tornare alcun errore	Mi aspetto che implementando un oggetto di tipo user non vi siano errori all'interno della pagina	Non vi sono errori
2	Viene testato il metodo che controlla se l'utente ha fatto il login	Il metodo checkIfLogin che controlla se l'utente ha un login attivo, deve ritornare i valori corretti in entrambi i casi	Ritorna false quando l'utente non è loggato e true quanto invece lo è
3	Viene testato il metodo che setta i valori delle sessioni	Controllo che il metodo setUser imposti correttamente le tre variabili. Se non ha fatto login il metodo ritorna un valore false, altrimenti true	Le variabili vengono settate correttamente e il metodo ritorna i valori nel modo corretto
4	Vengono testati i tre getter delle variabili della classe	Controllo che getAccount, getAttivato, getRuolo ritornino il valore correttamente	Ritornano il valore correttamente
5	Viene testato il metodo per eseguire il login dell'account	Controllo che doLogin() effettua il login se trova utente e password nel db, in quel caso ritorna true, altrimenti false	Ritorna correttamente i due casi e se trova un account nel db, esegue il login salvando le tre sessioni
6	Viene testato il metodo che effettua il logout	Controllo che doLogout() svuota correttamente le sessioni.	Il metodo effettua correttamente il logout
7	Viene testato il metodo per cancellare un utente	Controllo che il metodo deleteAccount() cancelli correttamente il valore nel db e di conseguenza, si cancella anche dalla tabella di tipologia	Viene cancellato correttamente

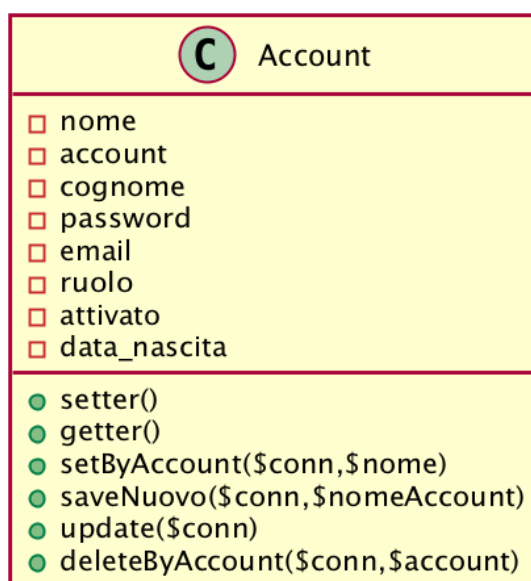
8	Viene testato il metodo per attivare un account	Controllo che activeAccount() attivi correttamente l'account, impostando il valore attivo a 1 nella tabella account	Viene impostato correttamente
9	Controllo che il metodo per cambiare l'email all'account funzioni correttamente	Utilizzando il metodo changeEmailToAccount() e passando come parametro la nuova mail e il nome account, ci si aspetta che venga cambiato il record	La nuova mail viene impostata correttamente
10	Controllo che il metodo per creare un nuovo utente funzioni e crei i record giusti	Utilizzando createNewAccount() e passandogli tutti i parametri, ci si aspetta che venga creato un nuovo account e la tipologia	Vengono creati correttamente

## 3.5 Account

### 3.5.1 Introduzione

Questa classe è molto simile a user, solamente che quella era dedicata più all'utente e la relazione con le varie sessioni e metodi utili per login/logout. Questa classe è invece la classe di relazione con database. Difatti facendo un confronto con lo schema ER del database, notiamo che le voci sono le stesse. I metodi della classe sono le relazioni base che si possono fare (INSERT, UPDATE, DELETE, SELECT).

### 3.5.2 Progettazione – Diagramma UML



### 3.5.3 Spiegazione UML

- Nome: nome dell'utente, ovvero nome della persona reale alla quale è associato l'account
- Account: nome dell'account per fare l'accesso al sito
- Cognome: cognome dell'utente, ovvero il cognome della persona reale alla quale è associato l'account
- Password: password dell'account per fare l'accesso al sito
- Email: email dell'utente alla quale si mandano le varie notifiche o annunci
- Ruolo: ruolo che ha quell'account (admin, gestore, segretario, ...)
- Attivato: è un valore booleano che controlla se l'utente ha già inserito tutte le informazioni utili (via, domicilio, nome padre e madre, numero telefono,...)
- Data\_nascita: è la data di nascita dell'utente
- Setter(): per ogni variabile di classe vi è un setter del valore. Esso riceve come parametro il valore da settare a quella variabile (Es setName(\$nome) setta la variabile di classe \$nome in base al parametro ricevuto)
- Getter(): come per il setter, per ogni variabile di classe c'è un getter che ritorna il valore
- setByAccount(): questo metodo riceve come parametro la connessione al db e il nome dell'account. Con questi due parametri setta tutte le variabili di classe in base al record che trova nel database.
- saveNuovo(): crea un nuovo record in account e ne salva il nome\_account. Il resto dei campi sono nulli e verranno settati in un secondo momento

- `update()`: esegue le modifiche all'interno del record selezionato, da `setByAccount`, in modo che tutti i valori cambiati grazie ai vari setter, vengano registrati e mantenuti dal database
- `deleteByAccount()`: riceve come parametro la connessione e il nome dell'account da cancellare

### 3.5.4 Implementazione

Questa classe è un po' la base di tutte le classi che si relazionano con il database. Difatti essa comprende una variabile per ogni attributo del db e ha i vari metodi per le operazioni base. Reputo che `setByAccount()`, `saveNuovo()` e `update()` siano molto importanti, in quanto grazie a questi tre eseguiamo tutte le operazioni utili che ci servono per la gestione di un account.

#### 3.5.4.1 `setByAccount()`

Come detto in precedenza questo metodo ha lo scopo di impostare tutte le variabili di classe in base ai record trovati all'interno del database, in base al parametro che riceve, ovvero il `$nome`. Possiamo notare come prima venga preparata la query e poi eseguita. Se nel database trova una compatibilità, allora setta tutte le variabili e poi ritorna `true`, altrimenti ritorna `false`.

```
public function setByAccount($conn, $nome) {
    $query = mysqli_query($conn, "select * from account where account='$nome' ");
    if ($query->num_rows > 0) {
        $query = $query->fetch_assoc();
        $this->nome = $query['nome'];
        $this->account = $query['account'];
        $this->cognome = $query['cognome'];
        $this->password = $query['password'];
        $this->email = $query['email'];
        $this->attivato = $query['attivato'];
        $this->ruolo = $query['ruolo'];
        $this->data_nascita = $query['data_nascita'];
        return true;
    } else {
        return false;
    }
}
```

#### 3.5.4.2 `saveNuovo()`

Questo metodo ha lo scopo di salvare un nuovo account, all'interno di account. Al momento di preoccupa solo di salvare il nome dell'account, il resto degli attributi verrà poi inserito manualmente dall'utente, quando farà l'accesso per la prima volta. È molto semplice questa funzione, all'inizio viene preparata la query, che è un semplicissimo insert all'interno della tabella account. Essa riceve come parametro `$nomeAccount` che sarà, appunto, il nome dell'account salvato. Nella seconda riga cambia il "?" con la variabile ricevuta come parametro e poi esegue la query.

```
public function saveNuovo($conn, $nomeAccount) {
    $query = $conn->prepare("INSERT INTO account (account) VALUES (?");
    $query->bind_param("s", $nomeAccount);
    $query->execute();
}
```



### 3.5.4.3 update()

Questa classe ha lo scopo di modificare tutti i valori dell'account. Per farlo bisogna utilizzare i vari setter della classe, in modo che questo metodo, alla fine, prende tutti i valori (es \$this->nome o \$this->cognome) e li salva all'interno del database. Notiamo come all'inizio viene salvato in una variabile \$account il nome dell'account dove eseguire le modifiche. Alla riga successiva viene preparata la query, con tutti i "?" che verranno poi sostituiti con i valori corretti. In ultimo il tutto viene eseguito e quindi salvato nel database.

```
public function update($conn) {
    $account = $this->account;
    $query = $conn->prepare("UPDATE account SET nome = ?,cognome = ?, password=?, email=?, attivato=?, ruolo=?,
    $query->bind_param("ssssiis", $this->nome, $this->cognome, $this->password, $this->email, $this->attivato,
    $query->execute();
}
```

### 3.5.5 Test Case

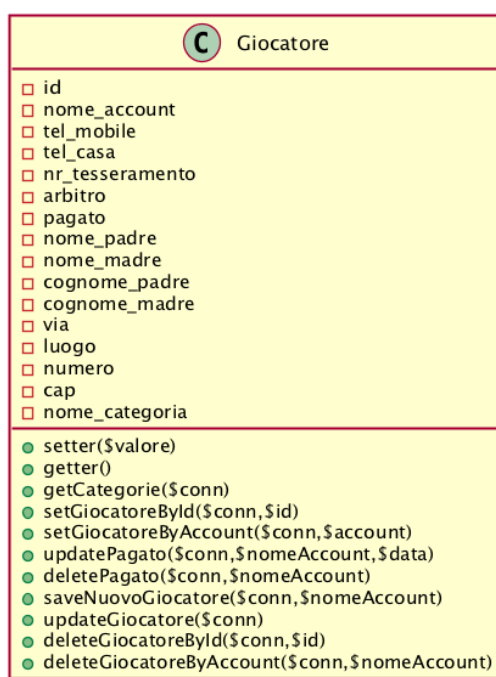
<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database grazie al metodo setByAccount	Mi aspetto che chiamando setByAccount e passandogli come valore il nome dell'account, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare un nuovo account	Chiamando saveNuovo il metodo salvi il nuovo account, passato come seconod attrivuto	Viene salvato un nuovo account all'interno del database, esso ha lo stesso nome passato come parametro
4	Controllo il metodo per effettuare dei cambiamenti all'account impostato (dopo averlo richiamato con i set) e quindi fare poi l'update su db	Richiamando il metodo update, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo il metodo per cancellare l'account	Richiamando il metodo delete() mi aspetto che l'account, passato come parametro, venga cancellato dal database	L'account viene cancellato correttamente

### 3.6 Giocatore

#### 3.6.1 Introduzione

Questa classe si occupa semplicemente di gestire tutto ciò che comporta un giocatore, quindi l'aggiunta/modifica delle varie informazioni e funzioni ad esso collegato. La classe giocatore è molto simile a classi già viste in precedenza, in quanto le varie operazioni che vengono fatte su database sono tutte molto uguali.

#### 3.6.2 Progettazione – Diagramma UML



#### 3.6.3 Spiegazione UML

- Id: id dell'account che viene salvato nel db, è un valore incrementale
- Nome\_account: nome dell'account alla quale è associata la tabella giocatore
- Tel\_mobile: numero di telefono mobile dell'utente
- Tel\_casa: numero di telefono di casa dell'utente
- Nr\_tesseramento: numero di tesseramento del giocatore
- Arbitro: valore booleano che segna se il giocatore è anche arbitro o no
- Pagato: valore che segna se il giocatore a pagato o meno la quota della stagione corrente
- Nome\_padre: Nome del padre del giocatore
- Nome\_madre: Nome della madre del giocatore
- Cognome\_padre: Cognome del padre del giocatore
- Cognome\_madre: Cognome della madre del giocatore
- Via: via dove abita il giocatore
- Luogo: luogo dove abita il giocatore
- Numero: numero della via di dove abita il giocatore

- Cap: cap del luogo in cui abita il giocatore
- Nome\_categoria: nome della categoria in cui fa parte il giocatore
- Setter(): tutti i vari setter degli attributi, dove riceve come parametro il valore che assegna alla variabile di classe, ad esempio (setNome(\$nome) setta la variabile di classe \$nome con il valore passato come parametro)
- Getter(): come per i setter, esiste un get per ogni variabile di classe, però essa ritorna il valore della variabile
- getCategorie(): ritorna la lista intera delle categorie
- setGiocatoreById(): questo metodo riceve come parametro la connessione e l'id del giocatore. Cerca all'interno del database una corrispondenza e quando la trova setta tutte le variabili in base al record trovato e ritorna true, altrimenti ritorna false
- setGiocatoreByAccount(): funziona nello stesso modo del setGiocatoreById, solamente che il controllo viene fatto in base al nome dell'account e non all'id
- updatePagato(): riceve come parametro la connessione, il nome dell'account e la data in cui è stato pagato. Setta l'attributo pagato nel database a 1 (ovvero che ha pagato) e poi crea un nuovo record all'interno della tabella pagamenti, che è lo storico di tutti pagamenti effettuati.
- deletePagato(): Setta l'attributo pagato all'interno del database a 0 (ovvero che non ha pagato) e cancella l'ultimo record associato al giocatore all'interno del database pagamenti.
- saveNuovoGiocatore(): salva un nuovo giocatore, riempiendo nel database solamente il nome dell'account, che viene passato come parametro. Gli altri attributi verranno settati in un secondo momento.
- updateGiocatore(): prende il valore di tutti gli attributi di classe e li salva all'interno del database. Quindi se questo metodo viene chiamato dopo vari set degli attributi, ne salva i cambiamenti all'interno del database.
- deleteGiocatoreById(): elimina il record del giocatore in base all'id
- deleteGiocatoreByAccount(): elimina il record del giocatore in base all'account.

### 3.6.4 Implementazione

Questa è una delle classi più grandi e complesse dell'intero progetto, in quanto comprende davvero molte funzioni e particolarità che non si ripetono per il resto delle classi. Dunque reputo che ci sono davvero molte funzioni che vanno commentate. Esse sono: updatePagato(), getCategorie(), setGiocatoreByAccount().

#### 3.6.4.1 updatePagato()

Questo metodo serve a registrare un pagamento effettuato dal giocatore. Prende come parametri una connessione, il nome dell'account del giocatore e la data del pagamento. Nella prima parte viene impostato l'attributo pagato, nella tabella giocatore, a 1, in modo da segnare che ha effettuato il pagamento per la stagione corrente. Nella seconda parte, effettua la registrazione del pagamenti, all'interno della tabella pagamenti, essa è lo storico di tutti i pagamenti fatti.

Per ogni esecuzione, prima viene effettuate un prepare statment dei valori, in modo da evitare sql injection. I valori vengono poi modificati e la query eseguita.

```
public function updatePagato($conn, $nomeAccount,$data){
    $query = $conn->prepare("UPDATE giocatore SET pagato = 1 where nome_account = ?;");
    $query->bind_param("s",$nomeAccount);
    $query->execute();

    $query = $conn->prepare("INSERT INTO pagamenti (account_giocatore,giorno) values (?,?);");
    $query->bind_param("ss",$nomeAccount,$data);
    $query->execute();
}
```

### 3.6.4.2 getCategoryie()

In alcune pagine del sito si necessita di avere l'intera lista delle categorie (U12, U10, U8,...). Per poterle utilizzare è stato creato questo metodo che si interfaccia alla tabella categoria, dove estrapola i nomi e li salva all'interno di un array, che a fine riempimento, viene ritornato. In questo caso non vengono usati dei prepare statement, in quanto non vengono passati dei parametri, ma è tutto preimpostato. All'inizio del metodo viene creato l'array categorie e in seguito si esegue la query. Per ogni valore trovato, si riempie l'array e poi, a fine riempimento, esso viene ritornato.

```
public function getCategoryie($conn) {
    $categorie = array();
    $query = mysqli_query($conn, "select nome from categoria ");
    while ($row = $query->fetch_assoc()) {
        $categorie[] = $row['nome'];
    }
    return $categorie;
}
```

### 3.6.4.3 setGiocatoreByAccount()

Questo metodo serve semplicemente a impostare tutte le variabili della classe in base al record che trova all'interno del database. Il record viene trovato in base al \$nome dell'account che viene passato come parametro. Se non viene trovato nessun record, il metodo ritorna false, altrimenti true.

```
public function setGiocatoreByAccount($conn,$nome){
    $query = mysqli_query($conn, "select * from giocatore where nome_account='$nome' " );
    if($query->num_rows>0){
        $query = $query->fetch_assoc();
        $this->id = $query['id'];
        $this->nome_account = $query['nome_account'];
        $this->tel_mobile = $query['tel_mobile'];
        $this->tel_casa = $query['tel_casa'];
        $this->nr_tesseramento = $query['nr_tesseramento'];
        $this->arbitro = $query['arbitro'];
        $this->pagato = $query['pagato'];
        $this->nome_padre = $query['nome_padre'];
        $this->nome_madre = $query['nome_madre'];
        $this->cognome_padre = $query['cognome_padre'];
        $this->cognome_madre = $query['cognome_madre'];
        $this->via = $query['via'];
        $this->numero = $query['numero'];
        $this->luogo = $query['luogo'];
        $this->cap = $query['cap'];
        $this->nome_categoria = $query['nome_categoria'];
        return true;
    }else{
        return false;
    }
}
```

### 3.6.5 Test Case

Nr Test	Descrizione	Aspettative	Risultato
1	Vengono controllati i vari setter e getter	Richiamando un set o un get esegue correttamente il suo scopo	Ritorna o imposta correttamente il valore

2	Viene controllato che chiamando il metodo per impostare i valori tramite id funzioni correttamente	Ci si aspetta che chiamando il metodo setGiocatoreById e passandogli come parametro una connessione al DB e l'id del giocatore, tutti gli attributi vengono settati correttamente	Tutti gli attributi vengono settati correttamente
3	Viene controllato che chiamando il metodo per impostare i valori tramite nome account funzioni correttamente	Ci si aspetta che chiamando il metodo setGiocatoreByAccount e passandogli come parametro una connessione al DB e il nome dell'account del giocatore, tutti gli attributi vengono settati correttamente	Tutti gli attributi vengono settati correttamente
4	Viene controllato che il metodo per creare un nuovo giocatore vuoto funzioni correttamente	Chiamando il metodo saveGiocatore() e passandogli come parametro la connessione ad un DB e il nome dell'account a cui associarlo, venga creato un nuovo record nel db con tutti i campi vuoti tranne id e nome_account	Viene creato correttamente con tutti i campi vuoti tranne id e nome_account
5	Viene controllato che il metodo per modificare un giocatore funzioni correttamente	Chiamando il metodo updateGiocatore dovrebbe salvare il nuovo giocatore nel caso in cui l'id è impostato.	Funziona correttamente la modifica al giocatore
6	Viene controllato che il metodo per cancellare un giocatore dal DB sia corretto	Chiamando il metodo deleteGiocatore e passandogli un id e la connessione, ci si aspetta che venga cancellato dal DB	Viene cancellato correttamente dal DB
7	Controllo che il metodo per ritornare la lista di categorie funzioni	Chiamando il metodo getCategorie mi aspetto che vengano ritornare tutte le categorie sotto forma di array	Vengono ritornate correttamente le categorie
8	Controllo che venga effettuato correttamente un pagamento	Mi aspetto che chiamando il metodo updatePagato() e passandogli i parametri, venga registrato correttamente il pagamento	Viene registrato correttamente il pagamento
9	Controllo il metodo per cancellare l'ultimo pagamento	Chiamando il metodo deletePagamento() mi aspetto che l'ultimo pagamento venga cancellato correttamente da entrambe le tabelle	Il pagamento viene cancellato correttamente da entrambe le tabelle

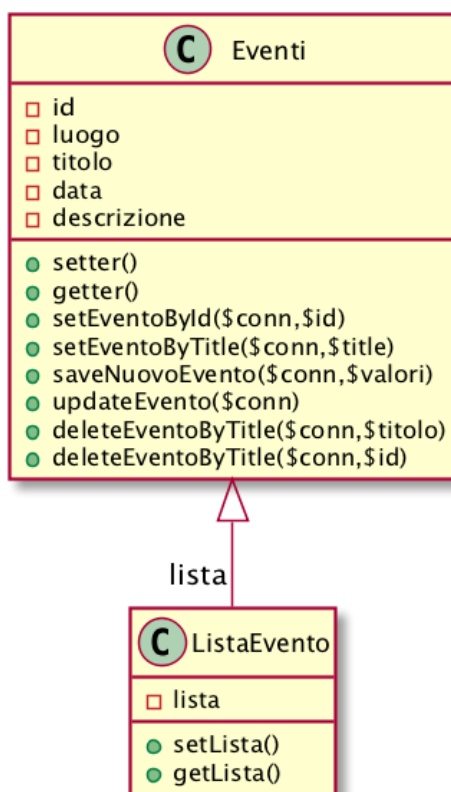
## 3.7 Eventi

### 3.7.1 Introduzione

Gli eventi sono divisi in tre parti, quella degli eventi, quindi il solito interfacciamento al database, la lista degli eventi, ovvero una lista di tutti gli eventi, in modo che questa lista possa poi essere gestita dalla terza parte, ovvero il calendario e la visualizzazione di queste informazioni. Per utilizzare quest'ultima parte mi sono appoggiato a uno dei software sopra descritti, ovvero fullcalendar.io molto funzionale e intuibile.

### 3.7.2 Progettazione

#### 3.7.2.1 Diagramma UML



#### 3.7.2.2 Spiegazione UML

Eventi:

- **Id:** questo è l'id di ogni singolo evento, esso è un valore numerico che viene definito in automatico dal sistema in modo incrementale
- **Luogo:** eventuale luogo dove si svolge l'evento
- **Titolo:** il titolo dell'evento
- **Data:** quando si svolge questo evento
- **Descrizione:** la descrizione dell'evento
- **Setter():** per ogni singolo attributo della classe, esiste un setter che, ricevendo un parametro come valore, lo imposta all'attributo
- **Getter():** per ogni singolo attributo della classe, esiste un getter che ritorna il valore dell'attributo
- **setEventoById():** setta tutti gli attributi della classe ricevendo come parametro l'id dell'evento

- `setEventoByTitle()`: setta tutti gli attributi della classe ricevendo come parametro il titolo dell'evento
- `saveNuovoEvento()`: salva un nuovo evento in base a tutti i parametri che riceve
- `updateEvento()`: salva il valore di tutti gli attributi di classe al record associato alla variabile id, quindi utile per delle modifiche ai vari eventi
- `deleteEventoById()`: elimina l'evento che ha l'id passato come parametro
- `deleteEventoByTitle()`: elimina l'evento che ha come titolo il titolo passato come parametro

ListaEventi:

- `lista`: la lista di tutti gli eventi
- `setLista()`: questo metodo fa un select su tutti gli eventi e riempie la variabile lista di un array dove ad ogni indice, si trova un oggetto di tipo evento.
- `getLista()`: questo metodo ritorna la variabile lista, in modo che possa essere utilizzata per molteplici funzioni, tipo il calendario.

### 3.7.2.3 Grafica

La pagina degli eventi viene divisa in più parti. Queste parti comprendono la normale visualizzazione dei vari eventi, il dettaglio di un evento, la modifica di un evento, la cancellazione e la creazione. Tali funzioni sono divise in finestre a parte.

Normale visualizzazione:

Questa visualizzazione si raggiunge semplicemente caricando la pagina `evento.php`, tutte le altre parti sono legate a questa.

**October 2016**

today < >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11 modific	12 tua ma	13 sml	14	15
16	17	18	19	20	21	22 Prova
23	24	25	26	27	28	29
30	31	1	2	3 Partita U10	4	5

#### Dettaglio, Modifica e Cancellazione Evento:

Questa sezione si raggiunge cliccando su un evento presente all'interno del calendario, per mostrare il dettaglio ho utilizzato i dialog in html, cambiandone la grafica con il css. Per la modifica basta cliccare su salva, mentre per la cancellazione vi è il tasto "cancella".

**Annuncio**

---

**Titolo:**

**Descrizione:**

**Luogo:**

**Data:**

#### Aggiunta Evento:

Questa sezione si raggiunge cliccando sul calendario ma non su un evento. La grafica è molto simile per quella del dettaglio.

**Annuncio**

---

**Titolo:**

**Descrizione:**

**Luogo:**

**Data:**

### **3.7.3 Implementazione**

#### **3.7.3.1 Eventi**

Questa classe ha il compito principale di offrire tutti metodi necessari per estrapolare e gestire i dati salvati nella tabella evento nel database. Molte di queste funzioni sono già state spiegate e viste nelle classi precedenti, però ritengo che un paio vadano comunque riviste, in quanto fondamentali.



I metodi sono i seguenti: `setEventoByTitle` e `saveNuovoEvento`

### **setEventoByTitle:**

Questo metodo si occupa principalmente di settare tutti gli attributi della classe evento, in base al record trovato all'interno del database. Il record viene trovato facendo un controllo sul titolo dell'evento e quello inserito come parametro. Poi, una volta eseguita la query, tutti gli attributi vengono salvate nelle rispettive variabili.

```
public function setEventoByTitle($conn, $title){
    $query = mysqli_query($conn, "select * from evento where titolo='$title' " );
    if($query->num_rows>0){
        $query = $query->fetch_assoc();
        $this->id = $query['id'];
        $this->titolo = $query['titolo'];
        $this->luogo = $query['luogo'];
        $this->descrizione = $query['descrizione'];
        $this->data = $query['data_evento'];
        return true;
    }else{
        return false;
    }
}
```

### **saveNuovoEvento:**

Questo è il metodo utilizzato per creare un nuovo evento. Riceve come parametro tutti gli attributi necessari per il database: titolo, descrizione, luogo e data. Essi vengono salvati all'interno delle variabili e poi sostituite ai punti di domanda. Dopodiché viene eseguita la query e quindi salvati i dati all'interno del database. L'id viene dato automaticamente dalla banca dati.

```
/**
 * Salva una nuova palestra nel db
 */
public function saveNuovoEvento($conn, $titolo, $luogo, $data, $descrizione){
    $query = $conn->prepare("INSERT INTO evento (luogo, titolo, descrizione, data_evento) VALUES (?, ?, ?, ?)");
    $query->bind_param("ssss", $luogo, $titolo, $descrizione, $data);
    $query->execute();
}
```

### **3.7.3.2 ListaEventi**

La lista Eventi è uguale a tutte le classi di tipo `Lista<Oggetto>`, ha come attributo di classe `lista`, ovvero un oggetto di tipo array che contiene tutti gli oggetti evento salvati all'interno del database. Così da poterli utilizzare, ad esempio, per il calendario.

### **setLista:**

Questo metodo esegue una query all'interno della tabella evento ed estrapola tutti i record salvati al suo interno. In questo modo poi popola l'array `lista` di oggetti evento con già tutte le variabili settate

```
public function setListaEvento() {
    $this->lista = array();
    $conn = new ConnectDB();
    $conn = $conn->getConnection();
    $query = mysqli_query($conn, "SELECT * from evento");
    if ($query->num_rows > 0) {
        while ($riga = $query->fetch_assoc()) {
            $evento = new Evento();
            $evento->setEventoById($conn, $riga['id']);
            $this->lista[] = $evento;
        }
    }
}
```

#### getLista:

Questo metodo non fa altro che ritornare l'attributo lista della classe, in modo da poter fare utilizzare all'utente tutti gli eventi.

```
public function getLista() {
    return $this->lista;
}
```

#### **3.7.3.3 Gestione\_evento**

Questa è la pagina vera e propria degli eventi, ovvero la visualizzazione grafica di tutta la logica impostata fin'ora. Per realizzare questa parte, ho utilizzato il plugin fullcalendar.io, molto utile e facile da usare. Questa pagina è puramente html e js. Alla creazione della stessa, la pagina fa una chiamata ajax a getLista che ritorna la lista in formato JSON che poi riutilizza per creare il calendario. Vediamo alcune parti fondamentali

#### Controllo permessi e login:

Questa pagina utilizza le classi create in precedenza, ovvero Session e ConnectDB. Difatti all'inizio della pagina viene fatto un controllo se l'utente ha i permessi per visualizzarla e soprattutto se ha fatto il login.

```
include_once 'Session.php';
include_once 'User.php';
$session = new Session();
$ruolo = $session->getVal("ruolo");
include_once 'Param.php';
if ($session->getVal('ruolo') != false or $session->getVal('ruolo')>-1) {
    include_once 'Menu.php';
}
```

#### Chiamata AJAX:

Questa funzione in jquery esegue una chiamata ajax alla pagina ListaEvento.php che ritorna la lista di eventi in formato JSON. Il risultato viene elaborato e poi salvato negli events del calendario, facendo così esso mostrerà tutti gli eventi.

```
$.ajax({
    url: "ListaEvento.php",
}).done(function (result) {

    $('#calendar').fullCalendar({
        defaultDate: new Date().toJSON(),
        editable: false,
        eventLimit: false,
        events: JSON.parse(result),
        eventClick: function (event) {
```

### **Click su evento:**

Questa funzione si apre nel momento in cui si clicca su un evento. Come possiamo notare i tag che hanno quegli id vengono prima svuotati del loro valore e poi ripopolati con quello nuovo e poi viene aperto il dialog. In questo modo possiamo visualizzare i dettagli.

```
dayClick: function (event) {
    $("#titolo").val("");
    $("#old_title").val("");
    $("#luogo").val("");

    $("#descrizione").val("");
    $("#pagina").val("nuovo");
    $("#data").val(event.format());
    document.getElementById("eventoDialog").showModal();
    return false;
}
```

## **3.7.4 Test Case**

### **3.7.4.1 Evento**

<b><u>Nr Test</u></b>	<b><u>Descrizione</u></b>	<b><u>Aspettative</u></b>	<b><u>Risultato</u></b>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database grazie ai metodi setEventoById e setEventoByTitle	Mi aspetto che chiamando setEventoById, passandogli come parametro un id, e setEventoByTitle, passandogli il titolo un evento, i vari attributi vengano settati correttamente	Richiamando i due metodi, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare un nuovo evento	Chiamando saveNuovoEvento e passandogli come parametro gli attributi, venga salvato un nuovo evento nel db	Viene salvato un nuovo evento nel db, avente gli stessi attributi di quelli passati come parametri
4	Controllo il metodo per effettuare dei cambiamenti all'evento impostato (dopo averlo richiamato con i set) e quindi fare poi l'update su db	Richiamando il metodo updateEvento, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare un evento sul db in base al titolo o all'id	Richiamando il metodo deleteById o deleteByTitle, mi aspetto che l'evento venga cancellato dal db	L'evento viene cancellato correttamente

### 3.7.4.2 Test ListaEvento

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo il metodo per settare la lista di tutti gli eventi	Mi aspetto che richiamando il metodo setListaEvento, l'attributo \$lista, contenga tutti gli eventi del calendario sotto oggetti Evento	La lista viene settata correttamente
2	Controllo il metodo per ritornare la lista	Mi aspetto che richiamando il metodo getLista ci venga ritornata l'attributo \$lista	La lista viene ritornata correttamente
3	Controllo il metodo per ritornare la lista in formato JSON	Mi aspetto che richiamando il metodo listaToJSON, la lista venga convertita in JSON e poi ritornata	La lista viene ritornata correttamente sotto forma JSON
4	Controllo le varie funzionalità della pagina ListaEvento.php	Mi aspetto che richiamando la pagina ListaEvento.php e passandogli come parametro, POST['pagina'] = nuovo venga creato un nuovo evento in base al resto dei parametri POST ricevuti.	Viene creato un nuovo evento con i parametri passati
5	Controllo le varie funzionalità della pagina ListaEvento.php	Mi aspetto che richiamando la pagina ListaEvento.php e passandogli come parametro, POST['pagina'] = modifica venga modificato l'evento in base al resto dei parametri POST ricevuti.	Viene modificato l'evento secondo i parametri passati
6	Controllo le varie funzionalità della pagina ListaEvento.php	Mi aspetto che richiamando la pagina ListaEvento.php e passandogli come parametro, POST['pagina'] = cancella venga cancellato l'evento passato come parametro	Viene cancellato l'evento passato come parametro
7	Controllo le varie funzionalità della pagina ListaEvento.php	Mi aspetto che richiamando la pagina ListaEvento.php e passandogli come parametro, POST['pagina'] = "un altro valore o nessuno", essa ritorni la lista degli eventi salvati nel db sotto forma di JSON	La pagina stampa la lista degli eventi sotto forma di JSON

### 3.7.4.3 Test Gestione\_Evento

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo l'accesso alla pagina	Mi aspetto che se non si ha fatto il login, il calendario non è accessibile	Ritorna l'errore se non si ha fatto l'accesso, quindi il calendario non è visibile
2	Controllo i permessi per effettuare le modifiche agli eventi (creare, cancellare e modificare)	Controllo che se non si hanno i permessi adeguati, non è possibile eseguire le operazioni sul calendario	Se non si hanno i permessi giusti, si possono solo visualizzare i dettagli del singolo evento
3	Controllo che si possa creare un evento	Controllo che se si hanno i permessi, cliccando su un giorno del calendario, si apra un form e dopo aver inserito i dati e salvato, venga creato un nuovo evento nel calendario.	Con i permessi è possibile creare un evento
4	Controllo che si possa cancellare un evento	Controllo che dopo aver cliccato su un evento si apra un form e dei dettagli e se si hanno i permessi, mi aspetto che il tasto cancella sia visibile e funzionante	Con i permessi è possibile cancellare gli eventi
5	Controllo che si possa modificare un evento	Mi aspetto che cliccando su un evento si apra un form contenente i dettagli modificabili, se si hanno i permessi, e cliccando sul tasto salva, esso modifichi i dati sul db	Con i permessi è possibile modificare gli eventi
6	Controllo che un evento si possa aprire e leggerne i dettagli	Controllo che cliccando su un evento si apra e mostri i dettagli dell'evento	Cliccando sull'evento si apre correttamente il form di dettaglio

## 3.8 Allenamento

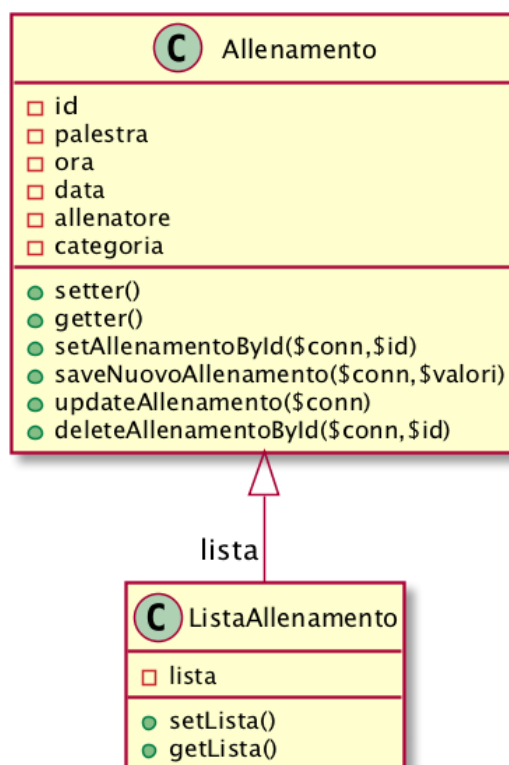
### 3.8.1 Introduzione

Come gli eventi, la parte degli allenamenti è suddivisa in tre parti, ovvero la classe che s'interfaccia con il database, la classe che recupera tutti gli allenamenti e li salva all'interno di un'array che dunque conterrà oggetti di tipo allenamento e l'ultima, la visualizzazione grafica di tutta questa parte.

La parte della listaAllenamenti non la spiego ulteriormente, in quanto uguale a quella degli eventi, con l'unica differenza che la query viene eseguita sulla tabella allenamento e non evento e la lista è popolata di oggetti di tipo allenamento.

### 3.8.2 Progettazione

#### 3.8.2.1 Diagramma UML



#### 3.8.2.2 Grafica

La pagina Allenamenti è divisa in tre parti fondamentali; la normale visualizzazione, ovvero quella che si vede appena si carica la pagina, la visualizzazione di modifica degli allenamenti, quella di inserimento di un nuovo allenamento e la cancellazione. Inoltre ci sono le parti di giocatore e quella di esercizi. Quest'ultima parte verrà approfondita in uno dei prossimi capitoli.

### Normale visualizzazione:

Per raggiungere questa visualizzazione, basta caricare la pagina gestione\_allenamento.php, essa sarà ciò che si vedrà, in caso si soddisfi i permessi e si ha fatto il login. Da questa pagina è poi possibile arrivare a tutte le funzioni a essa collegate, come anche la gestione degli esercizi.

Palestra	Allenatore	Data	Ora	Categoria	Giocatori	Esercizi	
Gmissss	si	2016-12-22	22:22:00	U10	<a href="#">Giocatori</a>	<a href="#">Esercizi</a>	<a href="#">modifica</a> <a href="#">cancella</a>
Gmissss	si	2017-01-12	10:10:00	U12	<a href="#">Giocatori</a>	<a href="#">Esercizi</a>	<a href="#">modifica</a> <a href="#">cancella</a>

[Aggiungi allenamento](#)  
[Gestione Esercizi](#)

### Modifica allenamento e Nuovo allenamento:

Cliccando sul bottone modifica oppure si aggiungi allenamento, si raggiungerà questa pagina. Nel caso di modifica i valori saranno già riempiti e modificando i campi e poi salvando, la modifica verrà mantenuta all'interno del database.

Palestra:

Categoria:

Allenatore:

Data:

Ora:

### Lista Giocatori:

Sempre passando dalla normale visualizzazione, possiamo accedere alla lista giocatori. Questa pagina ha la funzionalità di poter inserire come partecipante, un giocatore ad un determinato allenamento. Per farlo basta cercare il nome all'interno del campo di ricerca, selezionare e cliccare sul bottone.

Aggiungi Giocatore:

Nome	Cognome	Data di Nascita	Categoria	
simone	prova	1995-03-10	U12	<a href="#">cancella</a>

### 3.8.3 Implementazione

Per quanto riguarda l'implementazione a lato codice della classe Allenamento e ListaAllenamenti, essi sono stati scritti su modello di quella degli eventi, in quanto offrivano le stesse cose che necessita questa classe. Evito dunque di riportare codice doppione, ma preferisco concentrarmi sulla classe che cambia molto, ovvero gestione\_allenamento, più precisamente ciò che riguarda l'aggiunta dei giocatori agli allenamenti.

#### 3.8.3.1 Gestione\_allenamento – Aggiunta Giocatore

Reputo che questa parte della gestione allenamenti sia molto importante, in quanto ho comunque dovuto ideare da zero un campo di ricerca, che all'input dell'utente, cercasse dinamicamente all'interno del database il nome di ogni giocatore. Per farlo ho utilizzato jquery e ajax. Il primo per la gestione degli eventi, il secondo per fare richiesta ad una pagina php e attendere il risultato da stampare poi fuori.

#### Gestione degli input e chiamata ajax:

```

$("#giocatore").on('input',function(e){
    if(($(this).val() == "")){
        $("#aggiungiGiocatore").prop('disabled',true);
        $("#aggiungiConvocazione").prop('disabled',true);
        if($('#tendinaNomi').length){
            $('#tendinaNomi').remove();
        }
    }else{
        $("#aggiungiGiocatore").prop('disabled',false);
        $("#aggiungiConvocazione").prop('disabled',false);

        input2 = $(this).val();
        $.ajax({
            url: "searchGiocatore.php",
            type: 'GET',
            data: {input: input2},
            success: function(result){
                if($('#tendinaNomi').length){
                    $('#tendinaNomi').remove();
                }
                result = JSON.parse(result);
                selezionato = result[0][0];
                tendinaNomi(result);
            }
        });
    }
});

```

Come possiamo vedere dall'immagine, il campo input ha come id "giocatore". Ad ogni input da parte dell'utente viene chiamata la funzione e in base ai caratteri inseriti, esegue una determinata parte. Se l'input non ha caratteri, allora il bottone per aggiungere un giocatore viene disabilitato, come il menù a tendina, altrimenti viene abilitato il menù a tendina e il bottone. Subito dopo viene eseguita una chiamata ajax che interroga la pagina searchGiocatore.php, passandogli come parametro l'input. Prende la risposta e imposta il menù a tendina.



### Ricerca giocatore:

```
$input = $_GET['input'];
$conn = new ConnectDB();
$conn = $conn->getConnection();
$query = $conn->prepare("SELECT account.account, account.nome, account.cognome, giocatore.id from account join giocatori on account.id = giocatori.account_id where account.account = :ss");
$query->bind_param("ss", $input, $input);
$query->execute();

$query->bind_result($account, $name, $surname, $id);
$giocatori = array();
while ( $query-> fetch() ) {
    $giocatori[] = array($account, $name, $surname, $id);
}
echo json_encode($giocatori);
```

Come possiamo vedere dal codice qui sopra, la pagina prende l'input ricevuto come get, esegue una connessione al database e poi esegue la query, inserendo nel where, l'input in modo da fare un confronto con tutti i nomi dei giocatori. Ogni risultato che riceve, lo salva all'interno di un array che poi ritorna sotto formato json, in modo che sia gestibile da js nell'altra pagina.

### 3.8.4 Test Case

#### 3.8.4.1 Allenamento

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database grazie al metodo setAllenamentoByld	Mi aspetto che chiamando setAllenamentoByld, passandogli come parametro un id, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare un nuovo allenamento	Chiamando saveNuovoAllenamento e passandogli come parametro gli attributi, venga salvato un nuovo allenamento nel db	Viene salvato un nuovo allenamento nel db, avente gli stessi attributi di quelli passati come parametri
4	Controllo il metodo per effettuare dei cambiamenti all'allenamento impostato (dopo averlo richiamato con i set) e quindi fare poi l'update su db	Richiamando il metodo updateAllenamento, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare un allenamento sul db in base al titolo o all'id	Richiamando il metodo deleteByld mi aspetto che l'allenamento venga cancellato dal db	L'allenamento viene cancellato correttamente

### 3.8.4.2 ListaAllenamenti

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setLista	Mi aspetto che richiamando il metodo setLista() l'attributo \$lista, venga riempito con le informazioni necessarie, di ogni allenamento	Viene settato correttamente l'attributo
2	Controllo il metodo getLista	Mi aspetto che richiamando il metodo getLista venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.8.4.3 Gestione\_allenamenti

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti (almeno allenatore)	Se possiedo i permessi giusti, quindi da allenatore in su, posso visualizzare la pagina e fare le dovute operazioni. Altrimenti con privilegi minori, posso solo visualizzare la tabella e i dettagli	Solamente se si possiede i permessi dell'allenatore (in su) è possibile effettuare le operazioni sulla pagina, altrimenti si può solamente visualizzarla
3	Controllo la parte di modifica	Controllo che inserendo tutti campi correttamente e cliccando sul bottone invia, i dati vengano registrati e salvati	I dati vengono registrati correttamente
4		Controllo che non sia possibile inserire dati sbagliati	Solamente su chrome c'è il controllo. Altrimenti è possibile inserire dati sbagliati, ma comunque essi non vengono salvati all'interno del database

5		Controllo che non sia possibile non inserire i dati	Per salvare i dati bisogna riempire tutti i campi.
6	Controllo la parte di creazione	Controllo che inserendo tutti campi correttamente e cliccando sul bottone invia, i dati vengano registrati e salvati	I dati vengono registrati correttamente
7		Controllo che non sia possibile inserire dati sbagliati	Solamente su chrome c'è il controllo. Altrimenti è possibile inserire dati sbagliati, ma comunque essi non vengono salvati all'interno del database
8		Controllo che non sia possibile non inserire i dati	Per salvare i dati bisogna riempire tutti i campi.
9	Controllo la parte di cancellazione	Controllo che cliccando sul bottone cancella, venga chiesto all'utente di procedere. In caso di assenso viene cancellato quell'allenamento, in caso contrario allora si annulla l'operazione	Cliccando su sì, il record viene eliminato dal database
10	Viene controllata la parte dell'inserimento di un nuovo giocatore	Mi aspetto che selezionando un nuovo giocatore, sia possibile inserirlo solamente se non è già stato fatto in precedenza, altrimenti non lo inserisce.	L'inserimento avviene correttamente secondo le aspettative
11	Viene controllato la cancellazione di un giocatore dalla lista dell'allenamento selezionato	Mi aspetto che cliccando sul bottone cancella, il giocatore venga eliminato dalla lista	Il giocatore viene eliminato correttamente dalla lista

## 3.9 Esercizi

### 3.9.1 Introduzione

La classe è suddivisa in due parti, ovvero quella degli esercizi associati ad un allenamento e quella di gestione di ogni singolo esercizio. Per gli esercizi esistono dei file preimpostati, riempibili manualmente e poi da caricare sul web e da associare a un esercizio. Ogni singolo esercizio ha la possibilità di essere commentato e visualizzato nel dettaglio, in modo da poter essere sempre modificato e migliorato per gli allenamenti successivi.

### 3.9.2 Progettazione

#### 3.9.2.1 UML



#### 3.9.2.2 Grafica

Esistono due vie per raggiungere la parte degli esercizi. Una è quella di cliccare sulla voce di esercizi, all'interno di gestione allenamento. In quella vediamo tutti gli esercizi associati a quell'allenamento. È possibile inoltre, selezionando l'esercizio e cliccando su aggiungi, collegarlo all'allenamento. La seconda parte, accessibile sempre da gestione allenamento, ma cliccando sul link gestione esercizi, si arriverà alla pagina `gestione_esercizi.php`, accessibile solamente se si è almeno allenatore. In questa pagina è possibile fare le solite operazioni, modifica, cancellazione e aggiunta, e inoltre è possibile anche scaricare il file dell'esercizio e vedere il dettaglio.

### **Aggiunta Esercizio ad Allenamento:**

Per raggiungere questa visualizzazione basta cliccare sul bottone “Esercizi” di gestione allenamento. In questa pagina possiamo notare che vi è una tabella, contenente tutta la lista di esercizi associati all’allenamento corrente. Un allenatore (o chi ha più poteri di lui) può aggiungere altri esercizi dalla lista del menù a tendina, o eventualmente cancellarne cliccando sul bottone cancella.

**Aggiungi Esercizio:**

test

Aggiungi

Indietro

Titolo	Descrizione	Commento	File	
test	test	prova ciao cmiami tsmsi tiama eigims eigmgipfoe igmgimfie ifmigmid	33_SocietaBasket_SimoneEsposito_29_11.docx	cancella
prova	prova	pova	17_SocietaBasket_SimoneEsposito_12_10.docx	cancella

### **Gestione Esercizio – Normale Visualizzazione:**

Per raggiungere questa visualizzazione, bisogna cliccare sul link in gestione\_allenamento. Questo link ci porta a questa pagina, dove possiamo visualizzare i vari esercizi creati, crearne di nuovi, modificarli, scaricarli o vederne il dettaglio.

aggiungi esercizio

Indietro

Titolo	Descrizione	Commento	Download File	
test	test	prova ciao cmiami ts...	Download	modifica cancella dettaglio
prova	prova	pova...	Download	modifica cancella dettaglio
smi	mifn	fini...	Download	modifica cancella dettaglio

### Gestione Esercizio – Aggiungi e Modifica Esercizio:

Per raggiungere la parte di modifica/aggiungi, bisogna cliccare sull'apposito bottone in gestione\_esercizi. Arrivati a questo punto è possibile creare/modificare un esercizio. Caricando pure un file di tutti gli schemi necessari. Cliccando su indietro si torna alla visualizzazione precedente.

**Titolo:**

**Descrizione:**

**File Attuale:** 33\_SocietaBasket\_SimoneEsposito\_29\_11.docx  
 Nessun file selezionato

**Commento:**

### Gestione Esercizio – Dettaglio:

Questa pagina è quella di dettaglio di ogni singolo esercizio. Quindi è possibile aggiungere sempre note aggiuntive, in modo che ogni singolo esercizio possa essere migliorato per gli allenamenti successivi. Inoltre è molto utile per pescare esercizi non utilizzati per tanto tempo. Inoltre è pure possibile vedere quale tipo di file è associato a questo esercizio. Cliccando sul bottone indietro si torna alla gestione degli esercizi

**test**

**Commento:**

**Descrizione:**

**File:**

### 3.9.3 Implementazione

La logica che sta dietro alla modifica, cancellazione e aggiunta di qualcosa all'interno del database è molto simile alle classi precedenti, come la lista di questi oggetti. Così reputo molto più importante focalizzarmi su cambiamenti che caratterizzano questo tipo di classe. Tra queste novità, vi è il caricamento di un file su internet, e i suoi controlli, il download dello stesso e la visualizzazione in dettaglio di ogni esercizio. In ultimo, penso che sia molto importante spiegare pure l'aggiunta di un esercizio ad un allenamento.

#### Upload di un File:

Come possiamo vedere questa pagina riceve una variabile di tipo \$\_FILES e la elabora, salvando le varie informazioni utili, come il tipo, il nome, ... questo tipo di informazioni vengono poi collegate e poi viene chiamata la classe AggiungiFile. Essa si occupa semplicemente di copiare il file sul proprio server e poi di salvare nel database il valore. Se c'è stato un errore nell'upload, quindi potrebbe essere un formato sbagliato, allora stampa fuori l'errore.

```
$userfile_tmp = $_FILES['file']['tmp_name'];

//recupero il nome originale del file caricato
$userfile_name = $_FILES['file']['name'];

$userfile_type = $_FILES['file']['type'];

$aggiuntafile = new AggiungiFile();
$aggiuntafile->setTmp($userfile_tmp);
$aggiuntafile->setDir("esercizi");
$aggiuntafile->setFile($userfile_name);
$aggiuntafile->setType("docx");

if($aggiuntafile->uploadFile()){
    $esercizio->saveNuovoEsercizi($conn, $userfile_name, $_POST['descrizione'], $_POST['titolo'], $_POST['commento']);
}else{
    echo 'errore nell\'upload del file';
}
```

#### Download di un file:

Il download è molto semplice, basta mettere il file come link e allora parte automaticamente il download.

```
echo "<td><a href='esercizi/'. $lista[$i]->getFileSchema()."><button class='btn btn-success'>>
```

#### Dettaglio di un esercizio:

Il dettaglio dell'esercizio è semplicemente una pagina in cui si tirano fuori tutte le informazioni legate all'id dell'esercizio passato come post. Difatti vediamo che viene utilizzata la classe Esercizi, viene settato l'esercizio in base all'id e poi si tira fuori ogni informazione utile.

```
function dettaglio(){
    $esercizio = new Esercizi();
    $conn = new ConnectDB();
    $conn = $conn->getConnection();
    $esercizio->setEserciziById($conn, $_POST['esercizio']);

    echo '<div class="body_container row">';
    echo '<div id="users" class="col-md-10 body_left">';
    echo '<div class="table-responsive">';
    echo '<h2>'.$esercizio->getTitolo().</h2>';
    echo '<h4>Commento:</h4>';
    echo '<div id="box_details">'.$esercizio->getCommento().</div>';
    echo '<br>';
    echo '<h4>Descrizione:</h4>';
    echo '<div id="box_details">'.$esercizio->getDescrizione().</div>';
    echo '<br>';
    echo '<h4>File:</h4>';
    echo '<div id="box_details">'.$esercizio->getFileSchema().</div>';
    echo '</div>';
    echo '</div>';
    echo '<div class="col-md-2 body_right">';
    echo '<form action="gestione_esercizi.php"><input type="submit" value="indietro" class="btn btn-primary"/></form>';
    echo '</div>';
    echo '</div>';
}
```

### Aggiunta di un esercizio a un allenamento:

Per l'aggiunta esiste questa select che tira fuori ogni valore dalla lista di esercizi, classe lista, e ne tira fuori l'id (che è il valore di ogni option) e il titolo. Questi valori vengono poi generati e si crea così il menu a tendina.

```
echo '<div class="form-group "><label for="data">Aggiunga Esercizio:<select id="listaEsercizi" class="form-control" >';
for($i=0;$i<count($lista);$i++){
    echo "<option value='".$lista[$i]->getId().">".$lista[$i]->getTitolo()."</option>";
}
echo '</select>';
echo '<button id="aggiungiEsercizio" class="btn btn-success" >Aggiungi</button></div>';
```

Una volta selezionato l'esercizio e cliccato sul bottone aggiungi, allora entrerà in funzione jquery che prende l'evento e fa una chiamata ajax alla pagina listAllenamentiEsercizi. Essa si occupa semplicemente di prendere come valore l'id dell'allenamento e quello dell'esercizio e salvarli all'interno del database.

```
$("#aggiungiEsercizio").click(function(){
    allenamentoId = $('#allenamento').val();
    esercizio = $('#listaEsercizi').val();
    $.ajax({
        url: "ListaAllenamentiEsercizi.php",
        type: 'GET',
        data: {aggiungi: esercizio, allenamento: allenamentoId},
        success: function(result){
            location.reload();
        }
    });
});
```

### 3.9.4 Test Case

#### 3.9.4.1 Esercizi

Nr Test	Descrizione	Aspettative	Risultato
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database grazie al metodo setEsercizioById	Mi aspetto che chiamando setEsercizioById, passandogli come parametro un id, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare un nuovo esercizio.	Chiamando saveNuovoEsercizio e passandogli come parametro gli attributi, venga salvato un nuovo esercizio nel db	Viene salvato un nuovo esercizio nel db, avente gli stessi attributi di quelli passati come parametri



4	Controllo il metodo per effettuare dei cambiamenti all'esercizio impostato (dopo averlo richiamato con i set) e quindi fare poi l'update su db	Richiamando il metodo updateEsercizio, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare un esercizio sul db in all'id	Richiamando il metodo deleteByld mi aspetto che l'esercizio venga cancellato dal db	L'esercizio viene cancellato correttamente

### 3.9.4.2 ListaEsercizi

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setLista	Mi aspetto che richiamando il metodo setLista() l'attributo \$lista, venga riempito con le informazioni necessarie, di ogni esercizio	Viene settato correttamente l'attributo
2	Controllo il metodo getLista	Mi aspetto che richiamando il metodo getLista venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.9.4.3 Gestione Esercizi

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti (almeno allenatore)	Se possiedo i permessi giusti, quindi da allenatore in su, posso visualizzare la pagina e fare le dovute operazioni.	Solamente se si possiede i permessi dell'allenatore (in su) è possibile effettuare le operazioni sulla pagina.
3	Controllo la parte di aggiunta di un nuovo esercizio	Mi aspetto che inserendo tutti i valori corretti venga salvato un nuovo esercizio solamente in caso in cui venga caricato anche il file	Viene salvato un nuovo esercizio
4		Nel caso in cui manca un'informazione, non viene caricato nessun esercizio	Non viene salvato l'esercizio in mancanza di un dato

5	Controllo la parte di modifica di un esercizio	Mi aspetto che sia possibile salvare sempre le modifiche.	È possibile salvare sempre le modifiche
6	Controllo la parte di cancellazione di un esercizio	Mi aspetto che cliccando sul tasto cancella, all'utente venga chiesta conferma. In caso si continui, allora mi aspetto che l'esercizio venga cancellato, in caso contrario si annulla l'operazione	L'operazione continua correttamente, cancellando l'esercizio, in caso l'utente sceglie di continuare. In caso contrario si annulla tutto.
7	Controllo la possibilità di associare un esercizio ad un allenamento	Mi aspetto che cliccando sul bottone aggiungi, nella relazione allenamento-esercizio, venga associato	Viene associato correttamente l'esercizio ad un allenamento

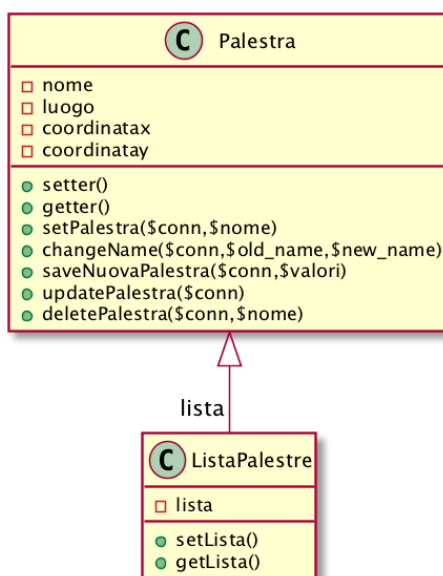
### 3.10 Palestra

#### 3.10.1 Introduzione

Questa è una classicissima classe di iterazione con il database, dove si eseguono tutte le operazioni utili alla palestra. Tutte queste funzioni sono già state approfondite nei capitoli precedenti, l'unica variante è la funzione changeName, che approfondirò qui in seguito. Per il resto evito di spiegarne il funzionamento, in quanto non c'è nulla di nuovo.

#### 3.10.2 Progettazione

##### 3.10.2.1 UML



##### 3.10.2.2 Grafica

Per accedere a questa pagina basta cliccare sul menù la voce Palestra e si accederà alla classica visualizzazione tabellare dei valori. Da questa pagina è possibile arrivare alle sottopagine di gestione; modifica, aggiunta e cancellazione. Era stato pensato di aggiungere pure il plugin di google maps per visualizzare su mappa dove si trovano le palestre, ma per questioni di tempo, non si è riusciti ad implementarlo.

##### Normale Visualizzazione:

Per arrivare a questa pagina si utilizza il menù. Una volta giunti qua si può visualizzare il dettaglio di ogni palestra. Inoltre è possibile eseguire le classiche operazioni di modifica/aggiunta e cancellazione. Per farlo basta cliccare sui rispettivi bottoni.

Nome	Luogo	Mappa		
Lambertenghi	Lugano	10;10	modifica	cancella
Nome_Palestra	Luogo	10;10	modifica	cancella

aggiungi palestra

### **Aggiungi e Modifica:**

Questa è la classica visualizzazione di aggiunta e modifica di una palestra. All'inizio si voleva avere, al posto dell'input di coordinate, la possibilità di avere una mappa di google, in modo da gestire lì sopra le coordinate reali. Attualmente quelle sono solo un'impostazione manuale, che verrà facilmente implementata in futuro. Cliccando su indietro, è possibile tornare alla gestione delle palestre.

**Nome Palestra:**

**Luogo:**

**Coordinate:**

### **3.10.3 Implementazione**

La logica utilizzata per questa pagina è un po' la base di tutte quelle di gestione, dato che non offre ancora la possibilità di aggiunta della mappa di google, non c'è nulla di nuovo. Per effettuare le operazioni base, vengono semplicemente chiamate i metodi della classe e poi salvate le informazioni all'interno del database, oppure cancellate.

### **3.10.4 Test Case**

#### **3.10.4.1 Palestra**

<b><u>Nr Test</u></b>	<b><u>Descrizione</u></b>	<b><u>Aspettative</u></b>	<b><u>Risultato</u></b>
1	Vengono controllati i vari getter e setter	Controllo che tutti i set e get delle variabili settino o riportino il loro valore correttamente	Il loro valore viene settato o ritornato correttamente
2	Controllo il metodo per settare tutti i valori del db in base al nome	Mi aspetto che passando il nome di una palestra come parametro a setPalestra le varie variabili vengono settate correttamente	Le variabili vengono settate correttamente
3	Viene controllato il metodo per creare un nuovo record palestra nel db	Mi aspetto che richiamando il metodo saveNuovaPalestra e passandogli i vari valori come parametro, venga registrato un nuovo record su db	Viene registrato un nuovo record sul DB

4	Viene controllato il metodo per cambiare il nome ad una palestra	Mi aspetto che richiamando il metodo changeName() e passando come parametro il vecchio nome e quello nuovo, avvenga la modifica nel db	Mi aspetto che la modifica avvenga correttamente
5	Controllo il metodo per fare le modifiche alle palestre già registrate	Richiamando il metodo updatePalestra() e passandogli come parametro i valori da cambiare e il nome della palestra, avvengano correttamente i cambiamenti	I cambiamenti avvengono correttamente
6	Controllo il metodo per cancellare una palestra dal db	Richiamando il metodo deletePalestra() mi aspetto che venga cancellata la palestra nel db in base al nome passato come parametro	La cancellazione avviene con successo

### 3.10.4.2 ListaPalestra

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setLista	Mi aspetto che richiamando il metodo setLista() l'attributo \$lista, venga riempito con le informazioni necessarie, di ogni palestra	Viene settato correttamente l'attributo
2	Controllo il metodo getLista	Mi aspetto che richiamando il metodo getLista venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.10.4.3 Gestione Palestre

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti (almeno allenatore)	Se possiedo i permessi giusti, quindi da allenatore in su, posso visualizzare la pagina e fare le dovute operazioni. Altrimenti con privilegi minori, posso solo visualizzare la tabella e i dettagli	Solamente se si possiede i permessi dell'allenatore (in su) è possibile effettuare le operazioni sulla pagina, altrimenti si può solamente visualizzarla

3		Mi aspetto che inserendo i dati corretti e cliccando su salva, i valori vengano cambiati e salvati correttamente	Vengono salvati correttamente i valori all'interno del database
4	Controllo la funzione di modifica	Mi aspetto che se un utente che tenta di impostare i campi nel modo non corretto, non possa continuare, dunque non viene salvato nessun record e venga stampato fuori un avviso	L'avviso non viene stampato fuori, ma all'interno del database non viene salvato nessun record
5		Mi aspetto che inserendo i dati corretti e cliccando su salva, i valori vengano salvati correttamente	Vengono salvati correttamente i valori all'interno del database
6	Controllo la funzione di aggiunta	Mi aspetto che se un utente che tenta di impostare i campi nel modo non corretto, non possa continuare, dunque non viene salvato nessun record e venga stampato fuori un avviso	L'avviso non viene stampato fuori, ma all'interno del database non viene salvato nessun record
7	Viene controllato la cancellazione di una palestra	Mi aspetto che cliccando sul bottone cancella, la palestra venga eliminata, solamente se l'utente accetta l'alert di continuare oppure o no	La palestra viene eliminata correttamente solamente dopo il consenso dell'utente, altrimenti si annulla l'operazione
8	Controllo i dati all'interno della tabella nella normale visualizzazione	Mi aspetto che all'interno della tabella i dati stampati siano completi e corretti	I dati sono completi e corretti

### 3.11 Partita

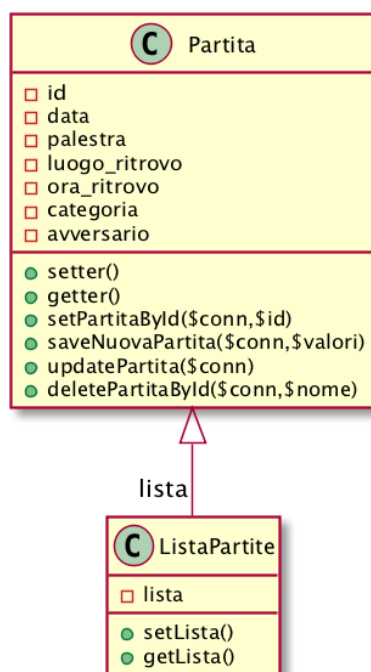
#### 3.11.1 Introduzione

La classe partita ha la funzione principale di interagire con la tabella partita all'interno del database. È stata pensata per registrare le partite da effettuare e anche quelle già finite, a cui poter associare poi delle immagini e dei commenti, ma questa sezione ne discuterò più a vanti nel capitolo Risultati.

Come possiamo notare, questa classe è molto simile alle altre, in particolare agli allenamenti, in quanto sono anche molto simili concettualmente, solamente che alle aprtite non associamo gli esercizi. Vedrò dunque di riportare alcune funzioni utili in gestione\_partite, più che alla classe

#### 3.11.2 Progettazione

##### 3.11.2.1 UML



Come possiamo notare dalla foto, la classe è il classico interfacciamento con il database. L'unica cosa nuova è la presenza dell'attributo avversario e della categoria. Il primo è il nome dell'avversario contro la quale si gioca, utile poi per mostrare i risultati di fine partita, mentre il secondo per vedere quale è la categoria giocante.

##### 3.11.2.2 Grafica

Graficamente, come tutto il resto, è molto simile agli allenamenti. Vi è la parte di modifica,aggiunta e cancellazione di una partita, oltre alla normale visualizzazione. Ritorna anche in questo caso la possibilità di aggiungere i vari giocatori che vengono convocati per la partita, un po' come la lista nell'altra classe. A fare una grande differenza qui è la possibilità di stampare le varie convocazioni alla partita, con un tasto annesso. Questo link genera un pdf che può essere stampato o scaricato. Un'altra piccola differenza è la possibilità è quella di aggiungere lo staff e assegnarli un ruolo.

### Normale Visualizzazione:

Questa pagina ricorda molto quella degli allenamenti, anche graficamente. È possibile arrivare alla normale visualizzazione semplicemente cliccando sul menù. Una volta fatto accesso a questa pagina, se si hanno i permessi, è possibile visualizzare anche tutte le funzioni per gestire una partita; dalle varie informazioni, all'aggiunta di staff e giocatori, oppure al commento finale (risultato, vedi capitolo successivo).

aggiungi partita

	Data	Ora	Palestra	Ora Partita	Cat.	Ritrovo	Avversario	Lista	Lista	Partita
<span style="background-color: #2196F3; color: white; padding: 2px 5px; border-radius: 3px;">modifica</span> <span style="background-color: #F44336; color: white; padding: 2px 5px; border-radius: 3px;">cancella</span>	2016-12-24	10:10	Lambertenghi	12:12:00	U10	Qui	Avversario	<span style="background-color: #2196F3; color: white; padding: 2px 5px; border-radius: 3px;">Staff</span>	<span style="background-color: #2196F3; color: white; padding: 2px 5px; border-radius: 3px;">Giocatori</span>	<span style="background-color: #2196F3; color: white; padding: 2px 5px; border-radius: 3px;">Dettaglio</span>

### Aggiungi e Modifica:

Per raggiungere queste due pagine, oltre ad avere i permessi speciali, bisogna cliccare sul pulsante “modifica” oppure su “aggiungi partita”. Da qui è possibile salvare o modificare le impostazioni di una partita. Queste impostazioni possono essere il nome dell'avversario, la categoria, l'ora e tutto ciò che serve. Cliccando su salva si torna alla normale visualizzazione.

**Data:**

**Ora di Ritrovo:**

**Ora Inizio Partita:**

**Palestra:**

**Categoria:**

**Luogo di ritrovo:**

**Avversario:**

Invia

Indietro

### Aggiungi Giocatore:

Questa parte delle partite è raggiungibile cliccando sul bottone “Giocatori”. Questa pagina è una copia di quello degli allenamenti, però ciò che differisce tanto è la possibilità di cliccare sul link a destra, ovvero il link che apre il pdf con la lista dei convocati. In questa pagina è comunque possibile aggiungere i giocatori e dunque convocarli.

Aggiungi

Stampa Convocati  
Indietro

Nome	Cognome	Data di Nascita	Categoria	Arbitro	
Simone	Esposito	1995-03-10	U12	Arbitro: <input type="text" value="No"/>	<span style="background-color: #F44336; color: white; padding: 2px 5px; border-radius: 3px;">cancella</span>



### Stampa Convocati:

Questa è una classica pagina che si apre contenente tutta la lista dei convocati.

## Convocazione per: Simone Esposito

Avversario: **Avversario**

Data: **2016-12-24**

Ora inizio partita: **12:12:00**

Luogo di Ritrovo: **Qui**

Ora di Ritrovo: **10:10**

In caso di problemi o assenza, chiamare il numero *0910001122*

### Aggiungi Staff:

L'aggiunta dello staff è molto simile a quella dei giocatori, solamente che una volta aggiunto un utente, ad esso si può assegnare un lavoro, compito, grazie al menù a tendina. Questa pagina è accessibile grazie al bottone Staff nella normale visualizzazione.

Nome	Cognome	Compito	
admin	admin	entrata ▾	<input type="button" value="cancella"/>

### 3.11.3 Implementazione

Le cose importanti di questa classe, o meglio quelle diverse da quelle scorse, sono; la creazione del file pdf, l'aggiunta di un compito ad un utente dello staff e la possibilità di settare un giocatore, come arbitro o meno. Questa funzione viene utilizzata quando un giocatore viene convocato non per giocare, ma piuttosto per arbitrare. Il resto della pagina è uguale a quella degli allenamenti su tutte e molto simile alle altre.

### Creazione della lista convocati:

Per creare il pdf in php ho utilizzato fpdf, una libreria gratuita e molto funzionale. Per fare funzionare questa parte ho esteso la classe base, facendo in modo che passandogli un giocatore e una partita, crea automaticamente il tagliando. I valori di tutti i giocatori e della partita li prende grazie alla funzione getConvocati, passandogli l'id della partita. Una volta ritornata la lista dei giocatori, per ogni giocatore, chiama il metodo Giocatore che prepara il tagliando.

```
$partita = $_POST['partita'];
$pdf = new PDF();
$pdf->AliasNbPages();
$pdf->AddPage();
$pdf->SetFont('Times', '', 12);
$convocati = $pdf->getConvocati($partita);
for ($i = 0; $i < count($convocati); $i++) {
    $pdf->Giocatore($convocati[$i][0], $partita, $convocati[$i][1]);
}
```

### Aggiunta opzione arbitro a giocatore:

Dopo aver aggiunto un giocatore a una partita, quindi impostato come convocato, si può anche mettere l'opzione se è stato convocato come arbitro o meno. Qui di seguito vediamo la cella del menù a tendina, contenente due valori, se "Si" allora significa che è arbitro per questa partita, altrimenti è "No". In base al valore selezionato all'interno del database, di default il giocatore avrà se arbitro o meno.

```
echo "<td>Arbitro: <select id='seArbitro' name='". $lista[$i][4]. "'>";
if($lista[$i][5]){
    echo '<option value="true" selected>Si</option>';
    echo '<option value="false">No</option>';
}else{
    echo '<option value="true">Si</option>';
    echo '<option value="false" selected>No</option>';
}
```

Una volta che si cambia il valore della select jquery prende l'evento e lo gestisce. Prende l'id della partita e del giocatore, oltre al suo valore e li manda, tramite una chiamata ajax, alla pagina ListaConvocati.php che si occupa semplicemente di salvare questi valori nel db.

```
$('#seArbitro').on('change', function() {
    partitaId = $('#partita').val();
    giocatore = $("#seArbitro").attr("name");
    $.ajax({
        url: "ListaConvocati.php",
        type: 'GET',
        data: {arbitro: this.value, giocatore: giocatore, partita:partitaId},
        success: function(result){
        }
    });
});
```

### Aggiunta compito a staff:

L'aggiunta del compito a uno dello staff è praticamente uguale alla selezione di arbitro o meno di un giocatore. Per eseguire questa funzione, la pagina esegue una select su tutti i lavori salvati nel db e ne stampa il contenuto nella select.

```
for($i=0;$i<count($lista);$i++){
    echo '<tr>';
    echo "<td>".$lista[$i][0]."</td>";
    echo "<td>".$lista[$i][1]."</td>";
    echo "<td><select id='selezionaCompito' name='".$lista[$i][3]."'>";
    if(null === $lista[$i][2]){
        echo "<option value='' selected></option>";
    }else{
        echo "<option value=''></option>";
    }
    for($l=0;$l<count($compiti);$l++){
        if($compiti[$l] === $lista[$i][2]){
            echo "<option value='".$compiti[$l]."' selected>".$compiti[$l]."</option>";
        }else{
            echo "<option value='".$compiti[$l]."'>".$compiti[$l]."</option>";
        }
    }
}
echo "</select></td>";
```

Una volta che si seleziona qualcosa di differente da quello già selezionato, quindi c'è un cambiamento, jquery prende questo evento ed esegue la solita chiamata ajax per salvare i valori.

```
$('#selezionaCompito').on('change', function() {
    partitaId = $('#partita').val();
    staff_partita = $("#selezionaCompito").attr("name");
    $.ajax({
        url: "ListaStaffPartita.php",
        type: 'GET',
        data: {compito: this.value, staff: staff_partita, partita:partitaId},
        success: function(result){
        }
    });
});
```

### 3.11.4 Test Case

#### 3.11.4.1 Classe Partita

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database grazie al metodo setPartitaByld	Mi aspetto che chiamando setPartitaByld, passandogli come parametro un id, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare una nuova partita	Chiamando saveNuovaPartita e passandogli come parametro gli attributi, venga salvata una nuova partita nel db	Viene salvata una nuova partita nel db, avente gli stessi attributi di quelli passati come parametri
4	Controllo il metodo per effettuare dei cambiamenti ad una partita	Richiamando il metodo updatePartita, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare una partita sul db in base all'id	Richiamando il metodo deleteByld mi aspetto che la partita venga cancellata dal db	La partita viene cancellata correttamente

#### 3.11.4.2 Classe Lista Partita

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setLista	Mi aspetto che richiamando il metodo setLista() l'attributo \$lista, venga riempito con le informazioni necessarie, di ogni partita	Viene settato correttamente l'attributo
2	Controllo il metodo getList	Mi aspetto che richiamando il metodo getList venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.11.4.3 Gestione Partita

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti	Controllo che la pagina offra una visualizzazione in base ai poteri che ha l'utente che la apre. In questo modo anche un utente normale può vedere le informazioni	La pagina risponde correttamente in base al tipo di utente che la chiama
3	Controllo la parte di modifica	Controllo che inserendo tutti campi correttamente e cliccando sul bottone invia, i dati vengano registrati e salvati	I dati vengono registrati correttamente
4		Controllo che non sia possibile inserire dati sbagliati	Solamente su chrome c'è il controllo. Altrimenti è possibile inserire dati sbagliati, ma comunque essi non vengono salvati all'interno del database
5		Controllo che non sia possibile non inserire i dati	Per salvare i dati bisogna riempire tutti i campi.
6	Controllo la parte di creazione	Controllo che inserendo tutti campi correttamente e cliccando sul bottone invia, i dati vengano registrati e salvati	I dati vengono registrati correttamente
7		Controllo che non sia possibile inserire dati sbagliati	Solamente su chrome c'è il controllo. Altrimenti è possibile inserire dati sbagliati, ma comunque essi non vengono salvati all'interno del database
8		Controllo che non sia possibile non inserire i dati	Per salvare i dati bisogna riempire tutti i campi.

9	Controllo la parte di cancellazione	Controllo che cliccando sul bottone cancella, venga chiesto all'utente di procedere. In caso di assenso viene cancellata quella partita, in caso contrario allora si annulla l'operazione	Cliccando su sì, il record viene eliminato dal database
10	Viene controllata la parte dell'inserimento di un nuovo giocatore	Mi aspetto che selezionando un nuovo giocatore, sia possibile inserirlo solamente se non è già stato fatto in precedenza, altrimenti non lo inserisce.	L'inserimento avviene correttamente secondo le aspettative
11	Viene controllato la cancellazione di un giocatore dalla lista dell'allenamento selezionato	Mi aspetto che cliccando sul bottone cancella, il giocatore venga eliminato dalla lista	Il giocatore viene eliminato correttamente dalla lista
12	Viene controllata la funzione di arbitro	Mi aspetto che cambiando il valore del menù a tendina, quello per selezionare se arbitro o meno, di un giocatore, tale cambiamento venga registrato nel database	Il valore viene cambiato correttamente all'interno del database
13	Viene controllato l'inserimento di un membro dello staff ad una partita	Selezionando un membro dello staff dal campo di ricerca e cliccando sul bottone, tale associazione deve essere salvata all'interno del database	L'associazione viene salvata correttamente all'interno del db
14	Viene controllato il cambiamento del ruolo di un membro dello staff	Mi aspetto che cambiando il valore del menù a tendina, quello per selezionare il tipo di compito, il nuovo valore venga salvato all'interno del database	Il nuovo valore viene salvato all'interno del database
15	Controllo il pdf per la convocazione dei giocatori	Mi aspetto che cliccando sul link per il file dei convocati, abbia tutte le informazioni corrette	Il file pdf ha tutte le informazioni corrette

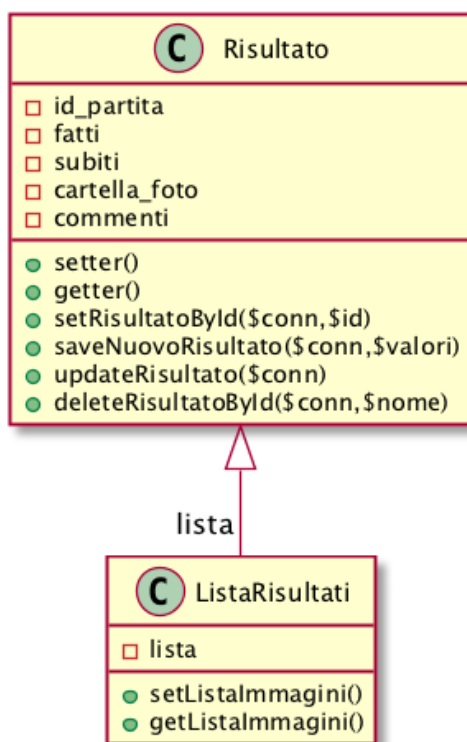
## 3.12 Risultato

### 3.12.1 Introduzione

Questa è la classe che tiene conto dei risultati di una partita e anche dei vari commenti, con la possibilità di aggiungere anche delle immagini e poi visualizzarle a fine partita. Un utente normale ha la possibilità di visualizzare le informazioni base di una partita, senza vedere i convocati o lo staff, ma può anche vedere il dettaglio. Questa parte è in sostanza il risultato della partita, comprende le foto e i commenti da parte del writer. Quindi è una funzione visibile a tutti, però gestibile da chi ha il poter di writer in su.

### 3.12.2 Progettazione

#### 3.12.2.1 UML



#### 3.12.2.2 Spiegazione UML

- `Id_partita`: è l'id della partita alla quale è associato il risultato
- `Fatti`: sono i canestri fatti dalla società
- `Subiti`: sono i canestri subiti dalla società
- `Cartella_foto`: la cartella in cui si trovano tutte le immagini
- `Commenti`: questo è il commento che scrive il writer, o chi commenta la partita
- `setListalmmagini()`: al posto del classico `setLista` di risultati, in questo caso la lista viene settata di tutte le immagini legate a quella partita, in modo che possano essere viste e consultate da un media.

### 3.12.2.3 Grafica

Tutti gli utenti possono visualizzare questa parte del sito, per farlo bisogna accedere prima alla pagina di gestione\_partita e poi cliccare su dettaglio. Una volta fatto accesso a questa pagina ci saranno due tipi di utenti. Quelli che sono utenti normali, quindi senza permessi speciali, e quelli che hanno i permessi da writer in su. Se una partita attualmente non ha ancora commenti allora ci sarà una visualizzazione base, dove si vede un singolo bottone da schiacciare, questo porta all'inserimento della partita. Mentre se la partita ha già un commento, allora è possibile vedere il dettaglio di fine partita.

#### Normale visualizzazione con permessi e senza commento:

Questa pagina è visualizzabile se non ci sono ancora commenti legato ad una partita. Per arrivarci basta cliccare sul bottone dettaglio di una partita. Il bottone indietro porta alla lista delle partite.

Aggiungi commento

Indietro

#### Normale visualizzazione con commento:

Se alla partita è già associato un commento, allora si possono vedere tutti i dettagli. Nella prima parte possiamo vedere la partita e il suo id, appena sotto il risultato della partita. Nel riquadro bianco si trova il commento del write e sotto uno slider che passa tutte le immagini associate alla partita.

modifica

Aggiungi Fotografie

Indietro

## Partita 1

Risultato Finale:

# Pura 10:20 Avversario

Commento:

Questo è il commento

Fotografie:

```

Function aggiungiModificaView() {
    if ($POST['titolo'] != "" AND $POST['descrizione'] != "" AND $POST['commento'] != "") {
        $esercizio = New Esercizio();
        $conn = new ConnectDB();
        $conn = $conn->getConnection();
        $esercizio->setEsercizioById($conn, $POST['id']);
        $esercizio->setTitolo($POST['titolo']);
        $esercizio->setDescrizione($POST['descrizione']);
        $esercizio->setCommento($POST['commento']);
        if(isset($FILES['file'])) {
            $userfile_tmp = $FILES['file']['tmp_name'];
            //recupero il nome originale del file caricato
            $userfile_name = $FILES['file']['name'];
            $userfile_type = $FILES['file']['type'];
            $aggiuntafile = new AggiungiFile();
            $aggiuntafile->setId($userfile_tmp);
            $aggiuntafile->setDir("esercizio");
            $aggiuntafile->setFile($userfile_name);
            $aggiuntafile->setType("docx");
        }
    }
}

```



### Aggiunta e Modifica Commento:

L'aggiunta e la modifica è una pagina semplicissima in cui poter aggiungere il commento in un input, mentre in quelli sotto si aggiungono i canestri fatti e subiti. Per raggiungere questa visualizzazione basta cliccare sul bottone "modifica" (nel caso in cui esiste già un commento) altrimenti su "aggiungi commento".

### Aggiunta di Fotografie:

Questa parte è accessibile solamente in caso che esista già il commento. Per arrivarci basta cliccare sul bottone Aggiungi Fotografie. Per caricare una foto, basta selezionarne una (o più) e poi cliccare su invia. Il programma si preoccuperà di caricare sulla cartella specifica tutte le immagini selezionate.

### **3.12.3 Implementazione**

La parte interessante di questa classe è quella del caricamento delle immagini, la visualizzazione delle stesse e la visualizzazione in base al commento. Il resto sono le classiche operazioni che si svolgono sul database, riprese già nelle scorse parti.

### Scelta di visualizzazione in base alla presenza di commento oppure no:

La logica che sta dietro alla visualizzazione in base alla presenza del commento o meno, è davvero molto semplice. In questo if cerco di settare tutti i valori presenti nel db all'interno della tabella risultati nel database in base all'id della partita, se non trova record, allora mi ritorna false e quindi entra nell'else. Altrimenti continua con la visualizzazione e quindi stampa fuori tutti i dettagli

```
if ($risultato->setRisultatoById($conn, $id_partita)) {
```

### Caricamento delle immagini:

La logica di caricamento delle immagini è molto simile a quella dei file. Nelle prime righe viene controllato se è stata inserita l'immagine. Poi esegue un controllo di tutte le immagini selezionate e ne tira fuori le informazioni. Il programma poi carica tutte le foto all'interno della cartella che ha l'id uguale a quella della partita. Prima di fare l'upload viene eseguita una scansione dell'estensione, per evitare file sbagliati.

```
function salvaImmagini() {
    if ($_POST['image_form_submit'] == 1) {
        $images_arr = array();
        foreach ($_FILES['images']['name'] as $key => $val) {
            $image_name = $_FILES['images']['name'][$key];
            $tmp_name = $_FILES['images']['tmp_name'][$key];
            $size = $_FILES['images']['size'][$key];
            $type = $_FILES['images']['type'][$key];
            $error = $_FILES['images']['error'][$key];

            $risultato = new Risultato();
            $conn = new ConnectDB();
            $conn = $conn->getConnection();
            $id_partita = $_POST['id_partita'];
            $risultato->setRisultatoById($conn, $id_partita);

            $target_dir = "media/" . $risultato->getCartellaFoto() . "/";

            if (strpos($type, 'png') OR strpos($type, 'jpg') OR strpos($type, 'jpeg')) {
                $target_file = $target_dir . $_FILES['images']['name'][$key];
                if (move_uploaded_file($_FILES['images']['tmp_name'][$key], $target_file)) {
                    $images_arr[] = $target_file;
                }
            }
        }
    }
}
```

### Visualizzazione delle immagini:

Per la visualizzazione delle immagini ho utilizzato il carousel di bootstrap, plugin molto comodo per mostrare tutte le immagini, qui di seguito è una piccola porzione del codice. Possiamo vedere come per ogni immagine si prepara il bottoncino per spostarsi a quella successiva o quella prima, in sostanza è una sorta di lista di immagini.

```
$lista->setListaImmagini($id_partita);
$immagini = $lista->getListaImmagini();
for ($i = 0; $i < count($immagini); $i++) {
    if ($i == 0) {
        echo '<li data-target="#myCarousel" data-slide-to="' . $i . '" class="active"></li>';
    } else {
        echo '<li data-target="#myCarousel" data-slide-to="' . $i . '"></li>';
    }
}
if ($i == count($immagini) - 1) {
    echo '</ol>';
    echo '<div class="carousel-inner" role="listbox">';
}
}
```

### 3.12.4 Test Case

#### 3.12.4.1 Classe Risultato

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database	Mi aspetto che chiamando setRisultatoById, passandogli come parametro un id di una partita, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare unna nuovo risultato	Chiamando saveNuovorisultato e passandogli come parametro gli attributi, venga salvato un nuovo risultato nel db	Viene salvato un nuovo risultato nel db, avente gli stessi attributi di quelli passati come parametri
4	Controllo il metodo per effettuare dei cambiamenti ad un risultato	Richiamando il metodo updateRisultato, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare un risultato sul db in base all'id	Richiamando il metodo deleteById mi aspetto che il risultato venga cancellato dal db	Il risultato viene cancellato correttamente

#### 3.12.4.2 Classe ListaRisultati

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setListalmmagini	Mi aspetto che richiamando il metodo setListalmmagini() l'attributo \$lista, venga riempito con tutte le immagini della cartella avente lo stesso id passato come parametro	Viene settato correttamente l'attributo lista
2	Controllo il metodo getListalmmagini	Mi aspetto che richiamando il metodo getListalmmagini venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.12.4.3 Pagina gestione Risultati

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti	Se possiedo i permessi giusti, quindi da writer in su, posso visualizzare la pagina e fare le dovute operazioni. Altrimenti con privilegi minori, posso solo visualizzare solo i dettagli	Solamente se si possiede i permessi del writer (in su) è possibile effettuare le operazioni sulla pagina, altrimenti si può solamente visualizzarla
3	Controllo la pagina se non è presente un commento	Se la partita non ha ancora un commento, allora mi aspetto che la visualizzazione, per chi ha i poteri almeno di writer, sia solo di un bottone	Viene visualizzato solamente il bottone aggiungi commento
4	Controllo la pagina se è presente un commento	Se la partita ha un commento già inserito, allora posso aggiungere le immagini oppure visualizzare i dettagli	Posso visualizzare i dettagli e inserire le fotografie
5	Viene controllata la creazione della cartella immagini	Controllo che all'inserimento di un commento, venga creata la cartella sull'host in cui poter inserire le fotografie	La cartella viene creata correttamente
6	Controllo l'inserimento dei dati nella modifica o aggiunta di un commento	Se inserisco dati sbagliati mi aspetto che non venga salvato nessun cambiamento	Non viene salvato nessun cambiamento
7	Controllo la visualizzazione dei dati	Controllo che tutti i del commento salvati sul db siano quelli che vengono visualizzati nella pagina	I dati sono gli stessi
8	Controllo le fotografie	Vengono controllate che tutte le fotografie caricate nella cartella della partita, vengano visualizzate all'interno del plugin per le foto	Le fotografie sono le stesse

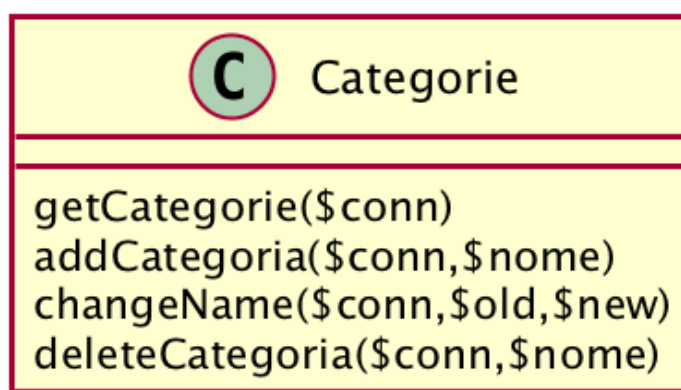
### 3.13 Categorie e Lavori

#### 3.13.1 Introduzione

La classe Categorie e Lavori sono uguali, l'unica cosa che cambia è la tabella su cui lavorano. Prendo quindi Categorie come esempio, così da poter analizzarle entrambe senza essere rindondanti nella spiegazione. La classe Categorie serve a sviluppare la lista di categorie, ovvero i nomi, da poter associare ai vari giocatori. Nel caso dei lavori, essi sono i compiti da associare allo staff quando viene convocato per una partita.

#### 3.13.2 Progettazione

##### 3.13.2.1 UML



##### 3.13.2.2 Spiegazione UML

- `getCategorie()`: ritorna l'intera lista di categorie sotto forma di lista
- `addCategoria()`: riceve come un parametro il nome della categoria da aggiungere nel database e la salva come nuovo record
- `changeName()`: riceve come parametro il nome vecchio da cambiare in quello nuovo, che è il terzo parametro
- `deleteCategoria`: riceve come parametro il nome della categoria da cancellare

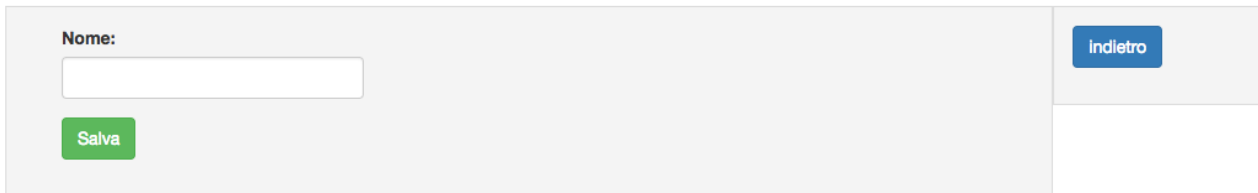
##### 3.13.2.3 Grafica

Questa pagina è accessibile solamente al getore e all'amministratore. Per farlo bisogna cliccare sulle "Impostazioni" del sito e poi sulla voce "Lista Categorie". Immediatamente arriveremo a visualizzare questa situazione.

The screenshot shows a web interface for managing categories. At the top left is a green button labeled 'aggiungi categoria'. At the top right is a blue button labeled 'indietro'. Below these is a table with two columns: 'Nome' and 'Azioni'.

Nome	Azioni
U10	<a href="#">modifica</a> <a href="#">cancella</a>
U12	<a href="#">modifica</a> <a href="#">cancella</a>
U14	<a href="#">modifica</a> <a href="#">cancella</a>

Da qui possiamo aggiungere modificare o cancellare il nome di una categoria, cliccando su uno dei primi due bottoni, ci apparirà un classico input dove inserire il nome.



### 3.13.3 Implementazione

Questa semplice classe offre la possibilità di gestire tutto ciò che serve a modificare una categoria, oppure ad aggiungerla. La pagina di impostazioni è accessibile solo con i permessi di gestore o amministratore. Dunque bisogna assicurarsi bene che questo controllo venga fatto bene. Reputo molto utile comunque spiegare il metodo `changeName` della classe e la visualizzazione tabellare delle varie categorie.

#### Metodo `changeName`:

Questo metodo ha lo scopo di cambiare il nome di una categoria, se caso essa è stata inserita precedentemente con un nome errato. Riceve come parametro la connessione, il vecchio nome da cambiare e il nuovo nome. Nella prima riga prepara la query da fare, ovvero fa un `UPDATE` della tabella categoria, cercando il record che ha come nome il parametro `$old`. Quando lo trova lo setta con il nuovo nome `$new`. Infine il metodo esegue questa query.

```
function changeName($conn,$old,$new){
    $query = $conn->prepare("UPDATE categoria set nome=? where nome=?");
    $query->bind_param("ss",$new,$old);
    $query->execute();
}
```

#### Visualizzazione tabellare delle categorie:

`$categorie` è una variabile in cui è salvata l'intera lista di nomi di categorie. Arrivato al `for`, esso continua fin quando ci sono elementi all'interno della categoria e al suo interno, ne stampa semplicemente il valore all'indice `$i`, il valore sarà il nome della categoria. In questo modo vengono stampate tutte le righe della tabella.

```
for($i=0;$i<count($categorie);$i++){
    echo '<tr>';
    echo "<td>".$categorie[$i]."</td>";
    echo '<td>';
```

### 3.13.4 Test Case

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo il metodo per ritornare la lista di nomi delle categorie	Mi aspetto che i nomi di tutte le categorie vengono ritornate correttamente	Vengono ritornati correttamente tutti i nomi
2	Controllo il metodo per aggiungere una nuova categoria	Mi aspetto che chiamando il metodo addCategoria e passandogli come parametro il nuovo nome, venga salvata una nuova categoria	Viene salvata una nuova categoria con il nome uguale a quello passato come parametro
3	Controllo il metodo changeName	Mi aspetto che chiamando il metodo changeName e passandogli come parametro il vecchio nome e quello nuovo, venga registrato il cambiamento nel database	Viene registrato correttamente il cambiamento all'interno del database
4	Controllo il metodo deleteCategoria	Mi aspetto che passando il nome della categoria a deleteCategoria, essa venga cancellata dal database	La categoria viene cancellata correttamente dal database

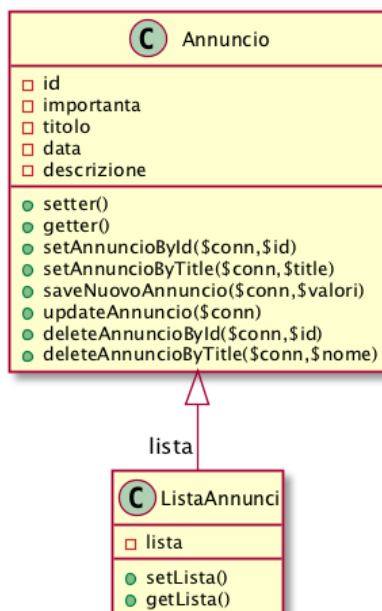
### 3.14 Annuncio

#### 3.14.1 Introduzione

La parte degli annunci è stata pensata per fornire la possibilità di scrivere degli annunci visibili agli utenti. In questo modo un utente che fa accesso al sito, potrà vedere se ci sono avvisi importanti da tenere d'occhio. È stata anche pensata per mandare dei remind ai vari utenti iscritti alle circolari via sms e whatsapp in modo automatico, ma è un'implementazione che deve ancora essere sviluppata.

#### 3.14.2 Progettazione

##### 3.14.2.1 UML



##### 3.14.2.2 Spiegazione delle Classe Annuncio

La classe annuncio ha un attributo chiamato importanza e data. Grazie a questi due valori, la pagina gestione annunci definisce l'ordine di lettura dell'utente, in modo che risalti all'occhio prima ciò che è importante e urgente e in seguito gli annunci secondari.

#### 3.14.3 Grafica

Questa parte del sito è forse quella che esce un po' di tutte dai soliti schemi, offrendo una grafica, seppur grezza, pensata molto a quello per cui è pensata. Difatti si è dovuto trovare una soluzione per dividere graficamente l'ordine e l'importanza dei vari annunci, oltre al fatto che fosse necessario anche trovare un modo per sistemare le varie informazioni. Accedendo alla pagina annunci, è possibile osservare la lista degli annunci non ancora scaduti, messi in ordine di importanza (rosso=importante, arancione=medio importante, verde= poco importante). Se si hanno i permessi è possibile eseguire le solite operazioni sugli annunci.



### Normale Visualizzazione:

Questa visualizzazione è accessibile cliccando sulla voce del menù “annuncio”. Potremmo dunque vedere tutti gli annunci che non sono ancora scaduti, messi in ordine per importanza. Come in questo caso, possiamo vedere in alto a destra, di ogni riquadro, la data. Appena sotto in centro il titolo e poi il contenuto dell’annuncio. Sotto è possibile cliccare su modifica o cancella l’annuncio, mentre in alto a destra è possibile aggiungere un annuncio. Il tutto è messo all’interno di un riquadro di colore diverso, che rispecchia l’importanza dell’annuncio.

### Modifica e Aggiungi:

Questa visualizzazione è accessibile cliccando sul bottone “aggiungi annuncio” oppure “modifica” nella normale visualizzazione. Da qui possiamo creare o modificare un annuncio inserendo le informazioni utili.

## 3.14.4 Implementazione

La parte più importante di questa classe, che differisce dalle altre, è semplicemente la query di chiamata dei vari annunci. Per tutto il resto è molto simile alla logica già proposta di aggiunta, modifica e cancellazione.

### 3.14.4.1 Query di chiamata e riordinamento

La seguente query è composta principalmente in due parti. La prima è la selezione che facciamo, ovvero in questo caso c’interessano tutti i campi degli annunci. La seconda parte, fondamentale per il riordinamento secondo importanza e data. Possiamo vedere che seleziono gli annunci solo più giovani della data attuale meno 3 ore, dunque continuo anche a vedere quelli già passati, ma non più di tre ore dalla chiamata. Infine questa selezione la facciamo ordinando per importanza decrescente (3 è importante e 1 poco importante)

```
"SELECT * from annunci WHERE data > DATE_ADD(now(),INTERVAL -3 HOUR) ORDER BY importanza desc";
```

### 3.14.5 Test Case

#### 3.14.5.1 Annuncio

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo i setter e i getter	Mi aspetto che richiamando i vari setter o getter, vengano settati correttamente gli attributi di classe	I setter e getter eseguono correttamente il loro funzionamento
2	Controllo che si riesca a recuperare i dati da database	Mi aspetto che chiamando setAnnuncioById, passandogli come parametro un id di una partita, i vari attributi vengano settati correttamente	Richiamando il metodo, gli attributi vengono settati correttamente
3	Controllo il metodo per salvare un nuovo annuncio	Chiamando saveNuovoAnnuncio e passandogli come parametro gli attributi, venga salvato un nuovo annuncio nel db	Viene salvato un nuovo annuncio nel db, avente gli stessi attributi di quelli passati come parametri
4	Controllo il metodo per effettuare dei cambiamenti ad un annuncio	Richiamando il metodo updateAnnuncio, mi aspetto che vengano registrati i cambiamenti su db	Vengono salvati i cambiamenti nel db, secondo il valore degli attributi
5	Controllo i metodi per cancellare un annuncio sul db in base all'id	Richiamando il metodo deleteById mi aspetto che l'annuncio venga cancellato dal db	L'annuncio viene cancellato correttamente

#### 3.14.5.2 Lista Annunci

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il metodo setLista	Mi aspetto che richiamando il metodo setList() l'attributo \$lista, venga riempito di tutti gli annunci dopo la data attuale -3 ore e nell'ordine corretto.	Viene settato correttamente l'attributo lista nell'ordine corretto
2	Controllo il metodo getLista	Mi aspetto che richiamando il metodo getLista venga ritornato correttamente l'attributo \$lista	Viene ritornato correttamente l'attributo lista

### 3.14.5.3 Gestione annunci

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo gli accessi alla pagina da utente non loggato	Mi aspetto che la pagina, se non ho fatto il login, non mi permetta di visualizzare nulla se non l'errore prestabilito	La pagina mi segnala l'errore e m'impedisce la visualizzazione della pagina e mi segnala l'errore
2	Controllo l'accesso con utenti con i permessi minori a quelli richiesti	Se possiedo i permessi giusti, quindi da allenatore in su, posso visualizzare la pagina e fare le dovute operazioni. Altrimenti con privilegi minori, posso solo visualizzare solo i dettagli	Solamente se si possiede i permessi di allenatore (in su) è possibile effettuare le operazioni sulla pagina, altrimenti si può solamente visualizzarla
3	Controllo il riempimento degli input	Mi aspetto che se si inserisce la data nel modo corretto, non venga salvato nel database. Gli altri dati, indipendentemente dal dato inserito, vengono accettati	Vengono accettati correttamente i dati e nel caso di un input di data sbagliato, allora non salva nel db
4	Controllo la funzione di modifica	Mi aspetto che modificando un annuncio e salvando il contenuto, i cambiamenti vengano registrati all'interno del database	I dati vengono registrati correttamente nel database
5	Controllo la funzione di aggiunta di un nuovo annuncio	Mi aspetto che creando un nuovo aggiunto venga creato un nuovo record all'interno del database	Viene creato correttamente un nuovo record all'interno del database
6	Controllo che gli annunci vengano disposti correttamente	Mi aspetto che gli annunci vengano mostrati nell'ordine corretto, secondo l'importanza e che essi abbiano il colore del riquadro del colore corretto	Il colore e l'ordine sono corretti.
7	Controllo la funzione di cancellazione	Mi aspetto che cliccando sul tasto cancella di un annuncio esso venga cancellato anche dal database	L'annuncio viene cancellato correttamente dal database

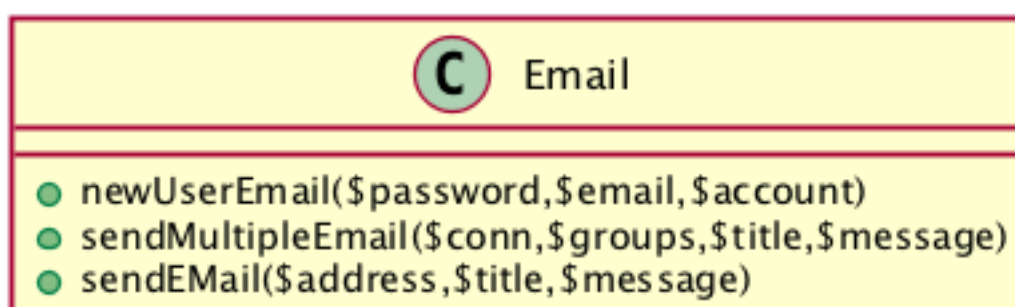
### 3.15 Email

#### 3.15.1 Introduzione

Questa classe viene utilizzata per tutti i tipi di invio delle email, dall'invio di una semplice email, a quello di una circolare a gruppi differenti. In questo modo è possibile spedire a tutti gli utenti desiderati. Questa classe viene utilizzata all'interno delle pagine di registrazione, recupero password e comunicazioni e non fa altro che spedire email.

#### 3.15.2 Progettazione

##### 3.15.2.1 UML



##### 3.15.2.2 Spiegazione UML

- `NewUserEmail()`: questo metodo viene chiamato quando si crea un nuovo utente. Riceve come parametro la password da spedire, l'indirizzo e il nome dell'account. Grazie a questi parametri, spedisce all'utente registrato i valori con cui fare l'accesso.
- `sendMultipleEmail()`: questo metodo viene chiamato principalmente nella pagina comunicazioni, difatti ha lo scopo di interpretare i vari gruppi a cui spedire le informazioni e poi mandare a tutti gli indirizzi il titolo e il messaggio della mail ricevuti. Alla fine chiama la funzione `sendEmail()`
- `sendEmail()`: questa funzione manda semplicemente una mail all'indirizzo che riceve come parametro.

#### 3.15.3 Implementazione

Essendo una classe che esce completamente dagli schemi, reputo molto importante spiegare nel dettaglio tutti e tre i metodi.

##### NewUserEmail:

Come possiamo vedere questo metodo è molto semplice, comprende solamente due righe che sono fondamentali. La prima si occupa semplicemente di preparare il testo dell'email con le varie informazioni utili, mentre la seconda chiama la funzione `sendEmail` passandogli tutti i parametri utili, questa spedisce la mail al nuovo utente.

```

public function newUserEmail($password, $email, $account) {
    $testo = "Account:". $account. ' Password:'. $password. ' Link: http://saminfo.ch/gbasket/login.php';
    $this->sendEmail($email, 'Nuovo Account Società di Basket', $testo);
}
  
```

### SendMultipleEmail:

Il controllo per ogni gruppo avviene in questo modo. Prima si decodifica quanto ricevuto dalla pagina comunicazioni e si prepara la listaEmail, che conterrà tutti i gruppi. Poi per ogni elemento di \$groups, si controlla quanti elementi ha e per ogni elemento, seleziono tutti gli account di quel gruppo e lo associo, ogni singolo indirizzo, alla variabile listaEmail. Quest'array verrà poi mandato alla funzione sendEmail.

```
$groups = json_decode($groups);
$listEmail = array();
for ($i = 0; $i < count($groups); $i++) {
    if ($groups[$i] == 'Tutti') {
        $query = mysqli_query($conn, "select email from account ");
        while ($row = $query->fetch_assoc()) {
            $listEmail[] = $row['email'];
        }
        $i = count($groups);
    }
}
```

### SendEmail:

Vediamo che il primo controllo che effettua questo metodo è quello degli indirizzi. Vede se il valore ricevuto è un array di indirizzi oppure un singolo indirizzo. Se è singolo allora spedisce l'email, altrimenti esegue un'altra operazione. Questa seconda operazione viene fatta perché se vengono superati 10 indirizzi nella stessa mail, allora il sistema lo percepisce come spam e quindi bisogna mandarli in gruppi da dieci.

```
public function sendEmail($address,$title,$message){
    if(is_array($address)){
        for ($i = 0; $i <= count($address); $i = $i + 10) {
            $destinatari = "";
            for ($j = 0; $j < 10; $j++) {
                if (($i + $j) < (count($address))) {
                    if (($i + $j) != 0) {
                        $destinatari.=",";
                    }
                    $destinatari = $destinatari . $address[$i + $j];
                }
            }
            mail($destinatari, $title, $message);
        }
    }else{
        mail($address, $title, $message);
    }
}
```

### 3.15.4 Test Case

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo che venga mandata la mail dopo la registrazione	Mi aspetto che registrando un nuovo utente, alla sua email vengono mandate tutte le informazioni	Le informazioni vengono ricevute correttamente
2	Controllo che venga mandata una mail a tutti i gruppi passati come parametro	Mi aspetto che tutte le email dei gruppi selezionati ricevano le email	Tutti gli indirizzi ricevono le email
3	Controllo che tutte le email vengano spedite correttamente	Mi aspetto che le email vengano spedite correttamente agli indirizzi	Le email vengono spedite con successo

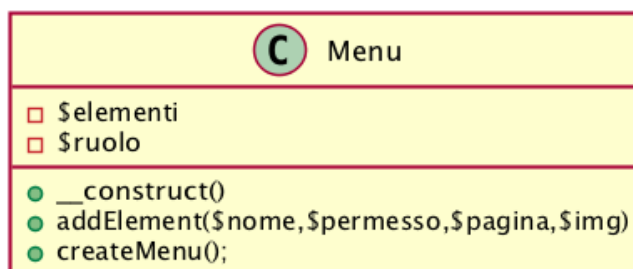
## 3.16 Menu

### 3.16.1 Introduzione

Questa classe si occupa solamente di creare il menù in modo intelligente e comodo. Difatti tutte le voci del menù, vengono dichiarate all'interno del costruttore, in modo che se un giorno va modificato qualcosa, si possa facilmente modificare qualcosa all'interno di quel metodo e poi tutto viene creato di conseguenza. Questa pagina viene implementata all'interno di quasi tutte le pagine, in modo che si possa navigare tra una e l'altra.

### 3.16.2 Progettazione

#### 3.16.2.1 UML

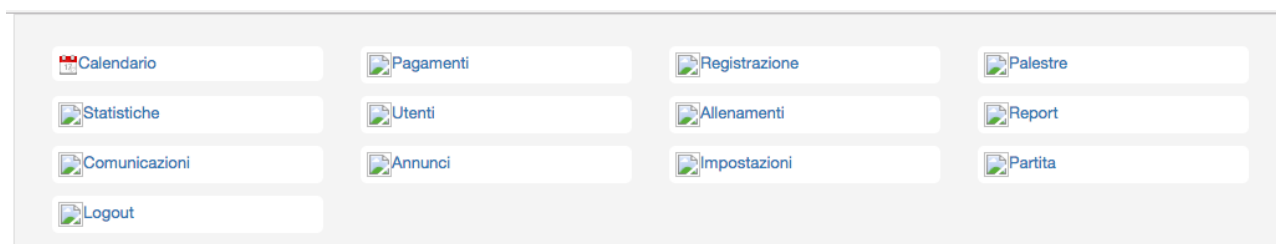


#### 3.16.2.2 Spiegazione UML

- Elementi: è un array contenente tutte le voci che viene utilizzato da createMenu() per creare il vero e proprio menù
- Ruolo: ruolo che ha l'utente che ha fatto login
- \_\_construct(): nel costruttore della classe vengono preparate tutte le voci del menù, passate a addElement, e poi viene settata la variabile \$ruolo
- addElement(): riceve i valori e prepara un array che inserisce come elemento dentro l'array "Elementi"
- createMenu(): prende l'array elemento e costruisce il menù in base anche al valore che è settato in ruolo

#### 3.16.2.3 Grafica

Il menù differisce in base al ruolo che lo consulta, in modo di non permettere l'accesso alle pagine a utenti che non sono abilitati. Dunque c'è una prima sicurezza nel lato del menù. Questa è la visualizzazione completa.



### 3.16.3 Implementazione

Questa classe è praticamente alla base della navigazione tra tutte le pagine, in quanto si tratta proprio del menù, dato che non fa riferimenti al database, ma solo alla sessione settata una volta fatto il login, differisce molto da tutte le altre, quindi è molto utile poter vedere nel dettaglio tutte le funzioni che offre.

#### Costruttore:

In questa funzione, nonché costruttore della classe, aggiungiamo manualmente tutti i valori del menù, in modo che essi vengano creati e visualizzati nel modo corretto. Vediamo che inizialmente salviamo nella variabile \$ruolo il valore della sessione ruolo, poi vengono aggiunte le voci. Le voci sono settate con il: "nome", il ruolo che può visualizzarlo in poi, il nome della pagina che porta e l'immagine icona della voce. Tutti questi parametri vengono passati a addElement.

```
public function __construct() {
    $session = new Session();
    $this->ruolo = $session->getVal('ruolo');
    $this->addElement("Calendario", "0", "gestione_evento.php", "calendario.png");
    $this->addElement("Pagamenti", "3", "gestione_pagamenti.php", "img.png");
    $this->addElement("Registrazione", "4", "registrazione.php", "img.png");
    $this->addElement("Palestre", "4", "gestione_palestra.php", "img.png");
    $this->addElement("Statistiche", "0", "statistiche.php", "img.png");
    $this->addElement("Utenti", "4", "gestione_utenti.php", "img.png");
    $this->addElement("Allenamenti", "0", "gestione_allenamenti.php", "img.png");
    $this->addElement("Report", "2", "report.php", "img.png");
    $this->addElement("Comunicazioni", "3", "comunicazioni.php", "img.png");
    $this->addElement("Annunci", "0", "gestione_annunci.php", "img.png");
    $this->addElement("Impostazioni", "4", "impostazioni.php", "img.png");
    $this->addElement("Partita", "0", "gestione_partite.php", "img.png");
    $this->addElement("Logout", "0", "logout.php", "img.png");
}
```

#### addElement:

Chiamando addElemento e passandogli come parametro i valori necessari, esso li prende e li salva in un array e lo imposta poi come elemento all'interno dell'array di classe elementi.

```
public function addElement($nome,$permesso,$pagina, $img){
    $this->elementi[] = array($nome,$permesso,$pagina,$img);
}
```

#### createMenu:

Questa funzione si occupa semplicemente di prendere tutti gli elementi e in base al numero dell'elemento che viene creato, s'impone un tipo di classe di div. Questo viene semplicemente fatto per renderlo responsive nel modo che si desidera e per creare una logica che crea tutto il menù. Per vedere se stampare l'elemento o meno viene fatto un controllo sul ruolo

```
$cnt = 0;
echo '<div class="form-container form_menu">';
for($i=0;$i<count($this->elementi);$i++){
    if($cnt%4==0 AND $cnt>0){
        echo '</div>';
        echo '<div class="row row_menu">';
    }else if($cnt==0){
        echo '<div class="row row_menu">';
    }
    if($this->ruolo>=$this->elementi[$i][1]){
        $cnt++;
        echo '<div class="col-sm-3"><div class="col-sm-12 colonna">';
    }
}
echo '</div>';
echo '</div>';
```

### 3.16.4 Test Case

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo la creazione del menù	Mi aspetto che il menù viene creato correttamente chiamando la voce createMenu	Il menù viene creato correttamente
2	Controllo gli elementi del menù	Mi aspetto che gli elementi del menù siano corretti in base all'utente che li visualizza, ovvero in base al parametro impostato sotto ruolo	Gli elementi sono visualizzati correttamente in base all'utente



### 3.17 Main

#### 3.17.1 Introduzione

Questa pagina viene utilizzata solamente per fare un controllo al primo login. Nel caso in cui un account fa per la prima volta l'accesso al sito, allora l'utente deve ancora inserire le informazioni utili. Per questo motivo, non si viene abilitati fin quando non si riempiono i campi. Questa pagina è raggiungibile solamente grazie al login, difatti non vi è la voce sul menù.

#### 3.17.2 Progettazione

##### 3.17.2.1 Grafica

Possiamo notare che il main, nel caso in cui l'account non è ancora attivato, mostra i seguenti input da riempire prima di poter proseguire. Riempiti tutti questi campi, l'account viene impostato come attivato e dunque, potrà navigare tranquillamente tra le pagine che gli sono permesse.

### Riempire i campi per poter proseguire

<b>Nome:</b> <input type="text" value="simone"/>	<b>Cognome:</b> <input type="text" value="prova"/>
<b>Email:</b> <input type="text" value="simone@samtrevano.ch"/>	<b>Data di nascita:</b> <input type="text" value="10.03.1995"/>
<b>Nome Padre:</b> <input type="text" value="Antonio"/>	<b>Cognome Padre:</b> <input type="text" value="Esposito"/>
<b>Nome Madre:</b> <input type="text" value="Ursula"/>	<b>Cognome Madre:</b> <input type="text" value="Esposito"/>
<b>Numero di Cellulare:</b> <input type="text" value="049494"/>	<b>Numero del Telefono:</b> <input type="text" value="3935959"/>
<b>CAP:</b> <input type="text" value="6981"/>	<b>Luogo:</b> <input type="text" value="Bedigliora"/>
<b>Via:</b> <input type="text" value="Bedea"/>	<b>Numero Via:</b> <input type="text" value="2"/>

salva

### 3.17.3 Implementazione

Questa pagina esegue un controllo sull'account se è attivato o meno, dunque sfrutta il valore della variabile \$\_Session['attivato']. Se il valore è a 0, vuol dire che bisogna ancora inserire i dati e quindi viene stampato fuori tutto l'input. Altrimenti la pagina comprende solamente il menù e il resto è bianco. In questo modo l'utente può navigare alle pagine che gli serve. Qui di seguito mostro il semplice controllo che esegue e l'attivazione dell'account.

```
$user = new User();
$user->setUser();
$session->getVal('ruolo');
if (!$user->getAttivato()) {
    if(isset($_POST['pagina'])) {
        aggiungiModificaView();
        include_once 'Menu.php';
    } else {
        mainView($user->getAccount());
        include_once 'head.php';
    }
} else {
    include_once 'Menu.php';
}
```

### 3.17.4 Test Case

<u>Nr Test</u>	<u>Descrizione</u>	<u>Aspettative</u>	<u>Risultato</u>
1	Controllo il primo accesso	Mi aspetto che se l'account esegue per la prima volta l'accesso, vengano mostrati gli input da riempire	Gli input vengono chiesti correttamente, in base al tipo di account.
2	Controllo l'attivazione	Mi aspetto che dopo aver inserito tutti i valori correttamente e salvandoli, l'account venga attivato correttamente	L'account viene attivato correttamente e dopo è possibile navigare tra le pagine
3	Controllo un account già attivato	Mi aspetto che se un account è già stato attivato, allora non gli viene mostrato il form ma può navigare tranquillamente tra le pagine	Un account già attivato può navigare tranquillamente tra le pagine

### 3.18 Login e Logout

Queste pagine sono le principi due per poter gestire tutto ciò che concerne la connessione e la disconnessione al sito. Il logout si occupa semplicemente di richiamare il metodo destroy session all'interno della classe sessione, mentre il login, utilizza le due classi account e session per impostare i valori nel modo corretto.

#### 3.18.1 Progettazione

##### 3.18.1.1 Grafica

Graficamente presento solamente il login, in quanto il logout è semplicemente una voce nel menù che richiama la pagina logout ed essa riporta alla pagina di login. Come possiamo vedere, la pagina di login è molto semplice, presenta due input, uno per l'utente e l'altro per la password, un bottone e infine un link alla pagina di recupero password. Inserendo i due valori e cliccando su Entra, possiamo accedere alla pagina main, solamente se quanto inserito è corretto.

#### 3.18.2 Implementazione

La parte importante di questa parte sono sicuramente le chiamate alla classe account e session, per quanto riguarda il login, mentre la chiamato al logout per ciò che concerne la pagina di logout.

##### Login:

Possiamo prima notare che alla pagina di login viene eseguito prima un controllo di connessione, per vedere se è possibile, altrimenti errore. Poi gestiamo gli input ricevuti dal form, ovvero utente e password. Immediatamente il programma manda queste due variabili al metodo doLogin, che ritorna true se è riuscito nell'impresa di login (dunque non ci sono stati errori e i valori inseriti sono corretti). Se è corretto allora cambia pagina nel main.php, altrimenti stampa l'errore e ripropone il form.

```
$conn = new ConnectDB();
if($conn->getConnection()){ //controllo che non vi siano errori di connessione
    $utente = $_POST['utente'];
    $password = $_POST['password'];
    if($user->doLogin($utente,$password)){ //controllo che il login avvenga correttamente
        //mando alla pagina main
        header('Location: main.php');
    }else{ //altrimenti stampo la non riuscita del login
        echo 'Account o password sbagliati';
    }
}else{ //altrimenti stampo avvenuta errore di connessione
    echo 'errore connessione';
}
```

### Logout:

La pagina di logout è davvero molto semplice, per funzionare si appoggia alle due classi Session e User. Notiamo come vengono prima settate e poi viene chiamata semplicemente la funzione per sloggare un utente, ovvero doLogout(). Eseguita la disconnessione richiamiamo la pagina login.php.

```
include_once 'Session.php';
include_once 'User.php';
$session = new Session();
$user = new User();
$user->doLogout();
header("location: login.php");
```

### 3.18.3 Test Case

#### 3.18.3.1 Login

Nr Test	Descrizione	Aspettative	Risultato
1	Controllo l'inserimento di valori sbagliati	Mi aspetto che inserendo valori sbagliati, non sia possibile fare l'accesso al sito.	Inserendo valori errati, non è possibile fare login, ma mostra l'errore e bisogna reinserire i dati
2	Controllo l'inserimento di valori corretti	Mi aspetto che inserendo valori corretti e cliccando su entra, la pagina ci fa loggare e ci rimanda alla pagina main.php	Ci logga e ci riporta alla pagina di main.php correttamente

#### 3.18.3.2 Logout

Nr Test	Descrizione	Aspettative	Risultato
1	Controllo il sistema di logout	Mi aspetto che cliccando sulla voce di logout nel menù, il nostro account venga sloggato e la pagina ci reindirizza alla pagina login.php	Ci reindirizza correttamente alla pagina dopo che ci disconnette dal sito.

### 3.19 Registrazione

#### 3.19.1 Introduzione

La registrazione degli utenti è possibile solo ai gestori e agli amministratori. Questa parte si limita semplicemente a creare un nuovo account e, in base alla tipologia scelta, crea un nuovo record, associato all'account. La registrazione è fondamentale per creare un nuovo account, in quanto non è possibile farlo esternamente, ovvero un utente guest non ha modo di registrarsi al sito, ma spetta solamente chi ha i permessi.

#### 3.19.2 Progettazione

##### 3.19.2.1 Grafica

La grafica di questa pagina è davvero molto semplice, si tratta semplicemente di 6 input e un bottone dove si chiedono le informazioni base, questo perché il resto verrà poi inserito dall'utente una volta che si collega. Nel menù a tendina possiamo scegliere che tipo di utente è: giocatore, allenatore, supporter o staff.

##### 3.19.2.2 Implementazione

La logica che sta dietro a questa pagina è davvero molto semplice, tutta la sua complessità sta nel mandare l'email al proprietario dell'account e la creazione randomica della password. Il resto sono semplici insert all'interno del database costruiti grazie a classi già fatte.

Nella prima riga creiamo una stringa di 5 caratteri che viene formata grazie ad un numero randomico che inseriamo dentro il metodo hash. In poche parole, questo metodo prende il tipo di hash, in questo caso sha512 e ne genera una stringa in base al numero randomico. Di questa stringa ne prendiamo solo i primi 5 caratteri. Poi viene creato un nuovo oggetto email e con tutte queste informazioni, spediamo l'email all'indirizzo. L'email contiene account e password.

```
$password = substr(hash('sha512', rand()), 0, 5);
$email = new Email();
$email->newUserEmail($password, $_POST['email'],$_POST['account']);
```

In questa riga invece creiamo semplicemente un oggetto di account e ne salviamo uno nuovo con tutte le informazioni immesse all'interno degli input.

```
$user->createNewAccount($_POST['account'], $_POST['email'], $_POST['tipologia'], $_POST['nome'], $_POST['cognome'],
```

### 3.19.3 Test Case

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo che la pagina non è accessibile se non si è fatto il login e se non si è almeno gestori	Se faccio l'accesso come gestore e amministratore, posso visualizzare i form, altrimenti ricevo l'errore	L'accesso viene controllato correttamente
2	Controllo che se non si inseriscono tutti i campi, non è possibile eseguire l'invio per la creazione di un nuovo utente	Solamente se riempio tutti i campi di registrazione è possibile effettuare la creazione di un nuovo utente, altrimenti ricevo l'errore	Viene stampato l'errore in caso manchi un elemento, altrimenti viene eseguito correttamente l'invio
3	Controllo che la riga di codice per creare una stringa casuale funzioni correttamente	Eseguendo <code>substr(hash('sha512',rand()),0,5)</code> mi aspetto che ogni volta mi ritorna una stringa differente, composta sempre da 5 caratteri	Ritorna sempre un valore differente di lunghezza 5
4	Controllo che l'email venga spedita correttamente	Eseguendo il metodo di invio di una mail della classe Email.php, venga mandata correttamente l'email.	L'email viene mandata correttamente
5	Controllo che l'utente venga creato correttamente	Eseguendo il codice dopo aver riempito i campi, controllo che venga creato un nuovo utente e soprattutto che i campi nel db siano uguali a quelli inseriti	Viene creato con successo un nuovo utente

## 3.20 Scheda

### 3.20.1 Introduzione

Questa pagina è stata implementata semplicemente per cambiare le informazioni base dell'account, informazioni come il luogo, la via, la data di nascita oppure la password. Un gestore o amministratore ha la possibilità di accedere alle schede di tutti gli utenti, mentre il singolo utente può accedere alla propria e modificare solamente la propria.

### 3.20.2 Progettazione

#### 3.20.2.1 Grafica

Per accedere alle schede esistono due modi: tramite la voce “Scheda” nel menù, accessibile da tutti gli utenti, oppure sulla voce “utenti” del menù e poi si clicca sull'utente desiderato, questa seconda opzione è possibile solamente ha chi ha i permessi, ovvero gestore o amministratore.

#### Visualizzazione base:

In questa visualizzazione possiamo vedere tutte le informazioni legate all'utente di cui abbiamo aperto la scheda e da qui, possiamo modificare la scheda oppure della password.

Nome:	Simone	<a href="#">modifica scheda</a>
Cognome:	Esposito	
Email:	simonenara90@gmail.com	
Data di Nascita:	1995-03-10	
Nr Cellulare:	0762801019	
Nr Tel. Casa:	0916081465	
Indirizzo:	6981 Bedigliora Bedea 2	

#### Modifica Scheda:

Per accedere alla modifica della scheda basta cliccare su “modifica scheda” e si vedrà questa visuale, per modificare le informazioni

Nome: <input type="text" value="Simone"/>	Cognome: <input type="text" value="Esposito"/>	<a href="#">Indietro</a>
Email: <input type="text" value="simonenara90@gmail.com"/>	Data di Nascita: <input type="text" value="10.03.1995"/>	
Numero Cellulare: <input type="text" value="0762801019"/>	Numero tel. Casa: <input type="text" value="0916081465"/>	
Cap: <input type="text" value="6981"/>	Luogo: <input type="text" value="Bedigliora"/>	
Via: <input type="text" value="Bedeà"/>	Numero Via: <input type="text" value="2"/>	
<a href="#">salva</a>		

Oppure questa per modificare la password

**Vecchia Password:**

**Nuova Password:**

**Ripeti Password:**

salva

### 3.20.3 Implementazione

Questa pagina è una semplice select sui dati salvati all'interno del database, nel caso della visualizzazione, e un update per le modifiche. Reputo che la parte nuova di questa pagina sia l'inserimento della password nuova e il controllo su quella vecchia. Vediamo nel dettaglio questa funzione.

All'interno della variabile \$account\_in viene salvato la scheda attuale alla quale si cerca di eseguire la modifica. Nell'if controlliamo che siano inseriti tutti i valori, altrimenti riporta l'errore e non fa modifiche alla pagina. Facciamo una select sull'account e la password, cifrata, all'interno del database, per vedere se esiste, se non è così allora ritorno che la password è errata, altrimenti prendo la password nuova, assicurandomi che siano uguali quelle inserite e la cifico, salvando il cambiamento all'interno dell'account.

```

$account_in = $_GET['account'];
if(isset($_POST['vecchia']) && isset($_POST['nuova']) && isset($_POST['nuova2'])) {
    $conn = new ConnectDB();
    $conn = $conn->getConnection();

    $vecchia = md5($_POST['vecchia']);
    $query = $conn->prepare("SELECT account FROM account WHERE account=? AND password=? LIMIT 1");
    $query->bind_param("ss", $account_in, $vecchia);
    $query->execute();

    $query->store_result();
    if($query->num_rows > 0) {
        if($_POST['nuova'] === $_POST['nuova2']) {
            $nuova = md5($_POST['nuova']);
            $query = $conn->prepare("UPDATE account SET password=? WHERE account=? AND password=? LIMIT 1");
            $query->bind_param("sss", $nuova, $account_in, $vecchia);
            $query->execute();
        } else {
            echo 'le due password nuove sono diverse';
        }
    } else {
        echo 'password vecchia errata';
    }
}

```



### 3.20.4 Test Case

<b>Nr Test</b>	<b>Descrizione</b>	<b>Aspettative</b>	<b>Risultato</b>
1	Controllo la visualizzazione di altre schede	Mi aspetto che se non hai i permessi non puoi visualizzare le schede degli altri, ma solo la tua.	Non puoi vedere le altre schede se non hai i permessi
2	Controllo le informazioni del database e quelle in scheda	Mi aspetto che le informazioni salvate nel database e quelle mostrate in scheda siano uguali	Le informazioni sono uguali
3	Controllo la modificabilità di una scheda	Mi aspetto che sia possibile modificare la scheda	È possibile modificare la scheda utente
4	Controllo i dati che si possono inserire	Mi aspetto che non sia possibile inserire dati da quelli richiesti, se ciò avviene, allora non vengono salvate le informazioni	Le informazioni non vengono salvate in caso di dati sbagliati

## 3.21 Report







### 3.21.1 Introduzione

Si necessitava di una funzione dove poter caricare e scaricare tutti report utili per la società, come fogli allenamenti, convocazioni, o altre documenti ufficiali. Questi, chiamati report, sono facilmente caricabili e scaricabili dal sito, tramite la pagina collegata. Questa è accessibile solamente da chi ha i permessi di allenatore in su, dunque non è visibile a tutti, in quanto alcuni documenti potrebbero essere importanti.

### 3.21.2 Progettazione

#### 3.21.2.1 Grafica

Questa pagina ha una sola visualizzazione, in quanto upload, download e cancellazione si trovano tutti nella stessa pagina. Come si può vedere dall'immagine, la parte sinistra è tutto ciò che concerne i file già caricati. Cliccando sul nome o sul bottone download è possibile scaricare il file selezionato, tale operazione partirà immediatamente. Mentre se clicchiamo sul tasto "cancella", verrà prima chiesto all'utente di procedere o meno. Infine a destra troviamo la parte di upload. Una volta selezionato il file e cliccato sul bottone, esso viene caricato sul sito.

Nome File:	Download	Scegli i File:
 27_SocietaBasket_SimoneEsposito_10_11.docx	 Download  Cancella	<input type="button" value="Scegli file"/> Nessun file selezionato <input type="button" value="upload"/>
 ProgettoBlueSky_SimoneEsposito.docx	 Download  Cancella	

### 3.21.3 Implementazione

Questa pagina riprende funzioni già trattate in precedenza; l'upload e download dei file. Per questo motivo ho deciso di evitare di riportare codice in più, in quanto uguale a quanto già spiegato. La pagina semplicemente si divide, per l'appunto in due parti: la prima si occupa di fare una select all'interno del db e controllare tutti i file caricati, poi controlla all'interno della cartella report e ne stampa fuori i vari collegamenti. Mentre la seconda parte è strettamente un form di upload, che rimanda alla pagina il file da caricare

#### 3.21.4 Test Case

Nr Test	Descrizione	Aspettative	Risultato
1	Controllo la possibilità di inserire i file	Mi aspetto che caricando un file non dia errori e che il file sia quello corretto	Il file caricato è quello corretto e non da errori
2	Controllo i file accettati	Mi aspetto che inserendo file con estensione non corretta dia errore	Da errore correttamente
3	Controllo i file da scaricare	Mi aspetto che cliccando sul nome o sul bottone, sia possibile scaricare il file	Il file viene scaricato
4	Controllo che il file scaricato	Mi aspetto che il file scaricato sia uguale a quello sul server	Il file è uguale

## 3.22 Statistiche

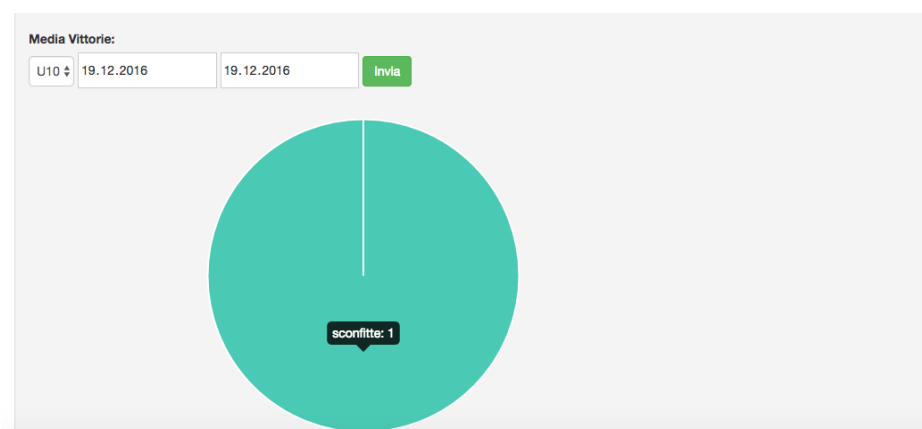
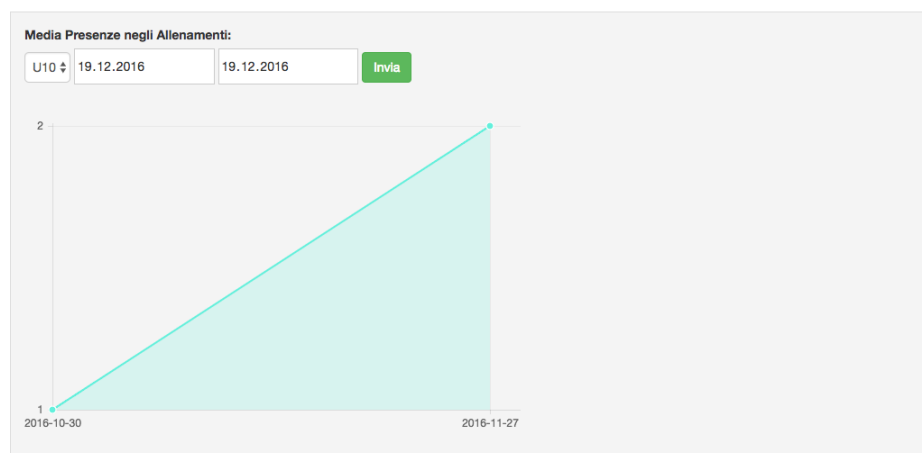
### 3.22.1 Introduzione

Era richiesto di mostrare pure delle statistiche base dei vari giocatori, ho scelto così di mostrarne tre che sono secondo me fondamentali per vedere un po' la partecipazione di ogni singolo giocatore e l'andamento. Difatti la pagina statistiche mostra la partecipazione dei giocatori ad un allenamento, ad una partita oppure mostra l'andamento di vittorie in un determinato periodo. Questa pagina è accessibile da tutti gli utenti, in quanto non necessita di modifiche o aggiunte, ma si basa sui valori trovati all'interno del database. Per realizzarlo ho dovuto utilizzare il plugin chart.js, molto comodo per creare grafici.

### 3.22.2 Progettazione

#### 3.22.2.1 Grafica

La pagina di statistiche è molto semplice, offre 3 grafici, uno sopra all'altro, tre form per grafico. In questi form è possibile scegliere quale tipo di periodo ci interessa sapere le statistiche, oppure quale categoria prendere in considerazione. I grafici sono due lineari, per l'andamento della presenza dei vari giocatori, e uno a torta per capire un po' nel complesso come sta andando quel periodo per quanto riguarda vittorie e sconfitte.



### 3.22.3 Implementazione

La logica che sta dietro questa pagina è molto semplice. Il plugin accetta i vari valori sotto forma di dato json e li stampa fuori. In questo esempio di codice vediamo una chiamata ajax alla pagina ListaStatistiche, che si limita semplicemente a stampare fuori i dati di una select. Questi dati ricevuti vengono inseriti all'interno di un array formato json, con tutte le intestazioni utili, come il nome delle colonne, il colore, i vari valori, ... infine all'interno del documento in cui importiamo questo file js, il grafico media\_presenze viene popolato da tutte le impostazioni inserite e si crea un nuovo grafico di tipo Line.

```
success: function (result) {
    result = JSON.parse(result);
    colonne = [];
    valore = []
    $.each(result, function () {
        valore.push($(this)[0]['numero']);
        colonne.push($(this)[0]['data_evento']);
    });
    var data = {
        labels: colonne,
        datasets: [
            {
                fillColor: "rgba(99,240,220,0.2)",
                strokeColor: "rgba(99,240,220,1)",
                pointColor: "rgba(99,240,220,1)",
                pointStrokeColor: "#fff",
                pointHighlightFill: "#fff",
                pointHighlightStroke: "rgba(220,220,220,1)",
                data: valore
            }
        ]
    };
    var ctx = document.getElementById("media_presenze").getContext("2d");
    var myLineChart = new Chart(ctx).Line(data);
}
```

### 3.22.4 Test Case

Nr Test	Descrizione	Aspettative	Risultato
1	Controllo che vengano formati correttamente i dati	Mi aspetto che inserendo correttamente i parametri e cliccando su invia, venga creato il grafico	Il grafico viene creato correttamente
2	Controllo i dati sul grafico	Mi aspetto che i dati risultanti sul grafico siano reali	I dati sono corretti
3	Controllo l'assenza di dati	Mi aspetto che in assenza di dati, venga mostrata una frase a rimpiazzare il grafico	Non viene mostrata la frase

#### 4 Test dei Requisiti

<b>Test Case:</b>	TC-001	<b>Nome:</b>	Realizzare un sito che permetta la gestione di una società di Basket
<b>Riferimento:</b>	REQ-001		
<b>Descrizione:</b>	Viene controllato che il sito offre delle pagine pubbliche per la presentazione della società di basket, e di pagine private per la gestione della società. Il sito deve poter essere raggiungibile in qualunque posto, dunque caricato su un host e deve essere accessibile grazie ad un login		
<b>Prerequisiti:</b>	Bisogna avere accesso a internet per testare la possibilità di raggiungere la macchina. Per entrare nelle pagine private bisogna inoltre avere un account.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedere al sito <a href="http://saminfo.ch/gbasket/login.php">http://saminfo.ch/gbasket/login.php</a> e vedere se il sito è accessibile</li> <li>2. testare la pagina di calendario.php, accessibile anche agli ospiti</li> <li>3. Accedere alla pagina <a href="http://saminfo.ch/gbasket/login.php">http://saminfo.ch/gbasket/login.php</a> e vedere se dispone del form di login</li> <li>4. Provare l'accesso al login con utente di prova (user: user.user, password: user) e vedere se offre delle pagine private per la gestione</li> </ol>		
<b>Risultati attesi:</b>	<p>Mi aspetto, per precedenti punti, le seguenti cose:</p> <ol style="list-style-type: none"> <li>1. Il sito sia caricato su un host raggiungibile da internet</li> <li>2. Vi siano pagine pubbliche, come il calendario.php, dunque si visualizzi la pagina correttamente</li> <li>3. La pagina login.php offre un form di login</li> <li>4. Sia possibile effettuare l'accesso al sito, tramite login e che esso offra delle pagine private. Queste pagine non possono essere raggiungibili se non si effettua il login</li> </ol>		

<b>Test Case:</b>	TC-002	<b>Nome:</b>	Il sito deve prevedere delle pagine pubbliche
<b>Riferimento:</b>	REQ-002		
<b>Descrizione:</b>	Il sito deve offrire delle pagine visibili anche dall'esterno, pagine utili per le informazioni e le comunicazioni		
<b>Prerequisiti:</b>	Bisogna semplicemente avere accesso a internet e poter raggiungere l'host del sito		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedendo al link <a href="http://saminfo.ch/gbasket/">http://saminfo.ch/gbasket/</a> deve essere possibile navigare esternamente al sito</li> <li>2. Controllare la pagina comunicazione e la possibilità di comunicare con la società</li> <li>3. Accedere alla pagina del calendario</li> <li>4. Accedere alla pagina delle partite</li> </ol>		
<b>Risultati attesi:</b>	Mi aspetto che il sito offra delle pagine consultabili anche esternamente, come le informazioni del sito o eventi legati alla società		

<b>Test Case:</b>	TC-003	<b>Nome:</b>	Il sito deve prevedere utenti differenti – guest
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente guest		
<b>Prerequisiti:</b>	Non bisogna avere nessun account, solo connessione al sito per testare le varie pagine.		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedere al form di login inserendo nulla nei campi</li> <li>2. Accedere alla pagina gestione_evento.php</li> <li>3. Accedere alla pagina gestione_pagamenti.php</li> <li>4. Accedere alla pagina registrazione.php</li> <li>5. Accedere alla pagina gestione_palestra.php</li> <li>6. Accedere alla pagina statistiche.php</li> <li>7. Accedere alla pagina gestione_utenti.php</li> <li>8. Accedere alla pagina gestione_allenamenti.php</li> <li>9. Accedere alla pagina report.php</li> <li>10. Accedere alla pagina comunicazioni.php</li> <li>11. Accedere alla pagina gestione_annunci.php</li> <li>12. Accedere alla pagina impostazioni.php</li> <li>13. Accedere alla pagina gestione_partite.php</li> <li>14. Accedere alla pagina scheda.php</li> </ol>		
<b>Risultati attesi:</b>	Mi aspetto che in tutte le pagine dia un messaggio che obbliga l'utente a fare il login e dunque non può visualizzare nulla		

<b>Test Case:</b>	TC-004	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente normale
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente normale		
<b>Prerequisiti:</b>	Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali <ul style="list-style-type: none"> <li>User: user.user</li> <li>Password: user</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>Accedere al form di login inserendo le credenziali</li> <li>Accedere alla pagina gestione_evento.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>Accedere alla pagina gestione_pagamenti.php</li> <li>Accedere alla pagina registrazione.php</li> <li>Accedere alla pagina gestione_palestra.php</li> <li>Accedere alla pagina statistiche.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare i vari campi</li> </ol> </li> <li>Accedere alla pagina gestione_utenti.php</li> <li>Accedere alla pagina gestione_allenamenti.php</li> <li>Accedere alla pagina report.php</li> <li>Accedere alla pagina comunicazioni.php</li> <li>Accedere alla pagina gestione_annunci.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>Accedere alla pagina impostazioni.php</li> <li>Accedere alla pagina gestione_partite.php <ol style="list-style-type: none"> <li>Visualizzare la pagina di partita</li> <li>Provare a modificare, aggiungere o cancellare una partita</li> <li>Accedere al dettaglio partita (risultato)</li> <li>Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>Accedere alla pagina scheda.php <ol style="list-style-type: none"> <li>Visualizzare la scheda</li> <li>Modificare i campi della scheda</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	Per i seguenti punti queste sono le aspettative: <ol style="list-style-type: none"> <li>È possibile fare il login</li> <li>Può visualizzare la pagina ma non eseguire le operazioni</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso e modificare i vari parametri</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso ma non può eseguire le operazioni</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso e vedere i risultati, ma non può fare le operazioni</li> <li>Può fare l'accesso ed eseguire le modifiche</li> </ol>		

<b>Test Case:</b>	TC-005	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente blogger
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente blogger		
<b>Prerequisiti:</b>	Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali <ul style="list-style-type: none"> <li>User: blogger.blogger</li> <li>Password: blogger</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>Accedere al form di login inserendo le credenziali</li> <li>Accedere alla pagina gestione_evento.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>Accedere alla pagina gestione_pagamenti.php</li> <li>Accedere alla pagina registrazione.php</li> <li>Accedere alla pagina gestione_palestra.php</li> <li>Accedere alla pagina statistiche.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare i vari campi</li> </ol> </li> <li>Accedere alla pagina gestione_utenti.php</li> <li>Accedere alla pagina gestione_allenamenti.php</li> <li>Accedere alla pagina report.php</li> <li>Accedere alla pagina comunicazioni.php</li> <li>Accedere alla pagina gestione_annunci.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>Accedere alla pagina impostazioni.php</li> <li>Accedere alla pagina gestione_partite.php <ol style="list-style-type: none"> <li>Visualizzare la pagina di partita</li> <li>Provare a modificare, aggiungere o cancellare una partita</li> <li>Accedere al dettaglio partita (risultato)</li> <li>Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>Accedere alla pagina scheda.php <ol style="list-style-type: none"> <li>Visualizzare la scheda</li> <li>Modificare i campi della scheda</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	<p>Per i seguenti punti queste sono le aspettative:</p> <ol style="list-style-type: none"> <li>È possibile fare il login</li> <li>Può visualizzare la pagina ma non eseguire le operazioni</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso e modificare i vari parametri</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso ma non può eseguire le operazioni</li> <li>Non può fare l'accesso</li> <li>Può fare l'accesso e vedere i risultati. Può eseguire le operazioni solamente sulla pagina risultati</li> <li>Può fare l'accesso ed eseguire le modifiche</li> </ol>		



<b>Test Case:</b>	TC-006	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente allenatore
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente allenatore		
<b>Prerequisiti:</b>	Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali <ul style="list-style-type: none"> <li>• User: allenatore.allenatore</li> <li>• Password: allenatore</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedere al form di login inserendo le credenziali</li> <li>2. Accedere alla pagina gestione_evento.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>3. Accedere alla pagina gestione_pagamenti.php</li> <li>4. Accedere alla pagina registrazione.php</li> <li>5. Accedere alla pagina gestione_palestra.php</li> <li>6. Accedere alla pagina statistiche.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare i vari campi</li> </ol> </li> <li>7. Accedere alla pagina gestione_utenti.php</li> <li>8. Accedere alla pagina gestione_allenamenti.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>9. Accedere alla pagina report.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo report</li> </ol> </li> <li>10. Accedere alla pagina comunicazioni.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Mandare una comunicazione</li> </ol> </li> <li>11. Accedere alla pagina gestione_annunci.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>12. Accedere alla pagina impostazioni.php</li> <li>13. Accedere alla pagina gestione_partite.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina di partita</li> <li>b. Provare a modificare, aggiungere o cancellare una partita</li> <li>c. Accedere al dettaglio partita (risultato)</li> <li>d. Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>14. Accedere alla pagina scheda.php               <ol style="list-style-type: none"> <li>a. Visualizzare la scheda</li> <li>b. Modificare i campi della scheda</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	Per i seguenti punti queste sono le aspettative: <ol style="list-style-type: none"> <li>1. È possibile fare il login</li> <li>2. Può visualizzare la pagina e può eseguire le operazioni</li> <li>3. Non può fare l'accesso</li> <li>4. Non può fare l'accesso</li> <li>5. Non può fare l'accesso</li> <li>6. Può fare l'accesso e modificare i vari parametri</li> <li>7. Non può fare l'accesso</li> <li>8. Può fare l'accesso ed eseguire le operazioni</li> <li>9. Può visualizzare la pagina e scaricare i report, il resto non può eseguirlo</li> <li>10. Può fare l'accesso ed eseguire le operazioni</li> <li>11. Può fare l'accesso ed eseguire le operazioni</li> <li>12. Non può fare l'accesso</li> <li>13. Può fare l'accesso alla pagina e eseguire le operazioni su entrambe le pagine</li> <li>14. Può fare l'accesso ed eseguire le modifiche</li> </ol>		

<b>Test Case:</b>	TC-007	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente segretario
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente segretario		
<b>Prerequisiti:</b>	Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali <ul style="list-style-type: none"> <li>• User: segretario.segretario</li> <li>• Password: segretario</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedere al form di login inserendo le credenziali</li> <li>2. Accedere alla pagina gestione_evento.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>3. Accedere alla pagina gestione_pagamenti.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>4. Accedere alla pagina registrazione.php</li> <li>5. Accedere alla pagina gestione_palestra.php</li> <li>6. Accedere alla pagina statistiche.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare i vari campi</li> </ol> </li> <li>7. Accedere alla pagina gestione_utenti.php</li> <li>8. Accedere alla pagina gestione_allenamenti.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>9. Accedere alla pagina report.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo report</li> </ol> </li> <li>10. Accedere alla pagina comunicazioni.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Mandare una comunicazione</li> </ol> </li> <li>11. Accedere alla pagina gestione_annunci.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>12. Accedere alla pagina impostazioni.php</li> <li>13. Accedere alla pagina gestione_partite.php               <ol style="list-style-type: none"> <li>a. Visualizzare la pagina di partita</li> <li>b. Provare a modificare, aggiungere o cancellare una partita</li> <li>c. Accedere al dettaglio partita (risultato)</li> <li>d. Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>14. Accedere alla pagina scheda.php               <ol style="list-style-type: none"> <li>a. Visualizzare la scheda</li> <li>b. Modificare i campi della scheda</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	Per i seguenti punti queste sono le aspettative: <ol style="list-style-type: none"> <li>1. È possibile fare il login</li> <li>2. Può visualizzare la pagina e può eseguire le operazioni</li> <li>3. Può visualizzare la pagina e può eseguire le operazioni</li> <li>4. Non può fare l'accesso</li> <li>5. Non può fare l'accesso</li> <li>6. Può fare l'accesso e modificare i vari parametri</li> <li>7. Non può fare l'accesso</li> <li>8. Può fare l'accesso ed eseguire le operazioni</li> <li>9. Può visualizzare la pagina e scaricare i report, il resto non può eseguirlo</li> <li>10. Può fare l'accesso ed eseguire le operazioni</li> <li>11. Può fare l'accesso ed eseguire le operazioni</li> <li>12. Non può fare l'accesso</li> </ol>		

	<p>13. Può fare l'accesso alla pagina e eseguire le operazioni su entrambi le pagine</p> <p>14. Può fare l'accesso ed eseguire le modifiche</p>
--	---

<b>Test Case:</b>	TC-008	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente gestore
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente gestore		
<b>Prerequisiti:</b>	<p>Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali</p> <ul style="list-style-type: none"> <li>• User: gestore.gestore</li> <li>• Password: gestore</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Accedere al form di login inserendo le credenziali</li> <li>2. Accedere alla pagina gestione_evento.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>3. Accedere alla pagina gestione_pagamenti.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>4. Accedere alla pagina registrazione.php</li> <li>5. Accedere alla pagina gestione_palestra.php</li> <li>6. Accedere alla pagina statistiche.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare i vari campi</li> </ol> </li> <li>7. Accedere alla pagina gestione_utenti.php</li> <li>8. Accedere alla pagina gestione_allenamenti.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>9. Accedere alla pagina report.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo report</li> </ol> </li> <li>10. Accedere alla pagina comunicazioni.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Mandare una comunicazione</li> </ol> </li> <li>11. Accedere alla pagina gestione_annunci.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina</li> <li>b. Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>12. Accedere alla pagina impostazioni.php</li> <li>13. Accedere alla pagina gestione_partite.php <ol style="list-style-type: none"> <li>a. Visualizzare la pagina di partita</li> <li>b. Provare a modificare, aggiungere o cancellare una partita</li> <li>c. Accedere al dettaglio partita (risultato)</li> <li>d. Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>14. Accedere alla pagina scheda.php <ol style="list-style-type: none"> <li>a. Visualizzare la scheda</li> <li>b. Modificare i campi della scheda</li> </ol> </li> </ol>		
<b>Risultati attesi:</b>	<p>Per i seguenti punti queste sono le aspettative:</p> <ol style="list-style-type: none"> <li>1. È possibile fare il login</li> <li>2. Può visualizzare la pagina e può eseguire le operazioni</li> <li>3. Può visualizzare la pagina e può eseguire le operazioni</li> <li>4. Può entrare nella pagina ed eseguire le operazioni</li> <li>5. Può entrare nella pagina ed eseguire le operazioni</li> <li>6. Può fare l'accesso e modificare i vari parametri</li> <li>7. Può entrare nella pagina ed eseguire le operazioni</li> </ol>		

	8. Può entrare nella pagina ed eseguire le operazioni 9. Può visualizzare la pagina ed eseguire l'inserimento dei report 10. Può fare l'accesso ed eseguire le operazioni 11. Può fare l'accesso ed eseguire le operazioni 12. Non può fare l'accesso 13. Può fare l'accesso alla pagina e eseguire le operazioni su entrambi le pagine 14. Può fare l'accesso ed eseguire le modifiche
--	---

<b>Test Case:</b>	TC-009	<b>Nome:</b>	Il sito deve prevedere utenti differenti – utente admin
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Esistono vari tipi di utenti che visualizzano parti diverse delle pagine. Bisogna dunque prevedere un modo per gestire più visualizzazioni e ruoli differenti. In questo caso testiamo l'utente admin		
<b>Prerequisiti:</b>	Per testare questa parte bisogna utilizzare un utente con le seguenti credenziali <ul style="list-style-type: none"> <li>User: admin.admin</li> <li>Password: admin</li> </ul>		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>Accedere al form di login inserendo le credenziali</li> <li>Accedere alla pagina gestione_evento.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a cancellare, aggiungere o modificare un evento</li> </ol> </li> <li>Accedere alla pagina gestione_pagamenti.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>Accedere alla pagina registrazione.php</li> <li>Accedere alla pagina gestione_palestra.php</li> <li>Accedere alla pagina statistiche.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare i vari campi</li> </ol> </li> <li>Accedere alla pagina gestione_utenti.php</li> <li>Accedere alla pagina gestione_allenamenti.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un allenamento</li> </ol> </li> <li>Accedere alla pagina report.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un nuovo report</li> </ol> </li> <li>Accedere alla pagina comunicazioni.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Mandare una comunicazione</li> </ol> </li> <li>Accedere alla pagina gestione_annunci.php <ol style="list-style-type: none"> <li>Visualizzare la pagina</li> <li>Provare a modificare, cancellare o aggiungere un nuovo annuncio</li> </ol> </li> <li>Accedere alla pagina impostazioni.php</li> <li>Accedere alla pagina gestione_partite.php <ol style="list-style-type: none"> <li>Visualizzare la pagina di partita</li> <li>Provare a modificare, aggiungere o cancellare una partita</li> <li>Accedere al dettaglio partita (risultato)</li> <li>Provare a modificare, aggiungere o cancellare il commento</li> </ol> </li> <li>Accedere alla pagina scheda.php <ol style="list-style-type: none"> <li>Visualizzare la scheda</li> <li>Modificare i campi della scheda</li> </ol> </li> </ol>		

<b>Risultati attesi:</b>	<p>Per i seguenti punti queste sono le aspettative:</p> <ol style="list-style-type: none"> <li>1. È possibile fare il login</li> <li>2. Può visualizzare la pagina e può eseguire le operazioni</li> <li>3. Può visualizzare la pagina e può eseguire le operazioni</li> <li>4. Può entrare nella pagina ed eseguire le operazioni</li> <li>5. Può entrare nella pagina ed eseguire le operazioni</li> <li>6. Può fare l'accesso e modificare i vari parametri</li> <li>7. Può entrare nella pagina ed eseguire le operazioni</li> <li>8. Può entrare nella pagina ed eseguire le operazioni</li> <li>9. Può visualizzare la pagina ed eseguire le operazioni</li> <li>10. Può fare l'accesso ed eseguire le operazioni</li> <li>11. Può fare l'accesso ed eseguire le operazioni</li> <li>12. Può fare l'accesso ed eseguire le operazioni</li> <li>13. Può fare l'accesso alla pagina e eseguire le operazioni su entrambi le pagine</li> <li>14. Può fare l'accesso ed eseguire le modifiche</li> </ol>
--------------------------	--

<b>Test Case:</b>	TC-010	<b>Nome:</b>	Livello di accesso
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	Bisogna prevedere livelli di accesso differenti		
<b>Prerequisiti:</b>	Account amministratore		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Bisogna fare accesso al sito come amministratore</li> <li>2. Andare in registrazione e creare un nuovo utente</li> <li>3. Andare in impostazioni e cliccare su gestione ruoli</li> <li>4. Visualizzare i ruoli degli utenti</li> <li>5. Andare in utenti e scegliere un utente per tipo</li> </ol>		
<b>Risultati attesi:</b>	<ol style="list-style-type: none"> <li>1. Si può fare l'accesso come amministratore</li> <li>2. Si può accedere nella pagina registrazione e creare un nuovo utente, scegliendo tra: giocatore, staff, supporter o allenatore. Arbitro e Commissario del campo sono una sotto categoria</li> <li>3. Possibile accedere alla pagina impostazioni e gestione ruoli</li> <li>4. Visualizzare i ruoli che devono essere: utente, writer, allenatore, segretario, gestore e amministratore</li> <li>5. Si possono visualizzare le schede degli utenti</li> </ol>		

<b>Test Case:</b>	TC-011	<b>Nome:</b>	Grafica Sito
<b>Riferimento:</b>	REQ-005		
<b>Descrizione:</b>	La grafica del sito deve rispecchiare lo scopo ed essere intuitiva		
<b>Prerequisiti:</b>	Avere un account		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Fare accesso al sito</li> <li>2. Visualizzare il menù e accedere alle varie funzioni</li> </ol>		
<b>Risultati attesi:</b>	Ci si aspetta che un utente riesca a giostrarsi tranquillamente all'interno del sito		

<b>Test Case:</b>	TC-012	<b>Nome:</b>	Comunicazioni
<b>Riferimento:</b>	REQ-006		
<b>Descrizione:</b>	Le comunicazioni agli utenti vengono fatte tramite annunci e scadenze degli stessi, tramite email, whatsapp e sms		
<b>Prerequisiti:</b>	Avere un account con permessi da allenatore in su		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Fare accesso al sito</li> <li>2. Creare un evento</li> <li>3. Attendere la scadenza dell'evento</li> <li>4. Controllare la ricezione dell'email e sms</li> </ol>		
<b>Risultati attesi:</b>	Mi aspetto che alla quasi scadenza di un evento, si riceva una comunicazione email e sms		

<b>Test Case:</b>	TC-013	<b>Nome:</b>	Allenamenti
<b>Riferimento:</b>	REQ-007		
<b>Descrizione:</b>	Un allenatore ha i permessi per creare, modificare, cancellare o visualizzare un allenamento. A questo allenamento ha pure la possibilità di aggiungere o cancellare un esercizio o un giocatore		
<b>Prerequisiti:</b>	Avere un account con i permessi di allenatore		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Fare accesso al sito con un account di allenatore o superiore</li> <li>2. Accedere alla pagina allenamento</li> <li>3. Cliccare sul bottone nuovo allenamento</li> <li>4. Inserire i dati, salvare e visualizzare se inserito un nuovo allenamento</li> <li>5. Modificare l'allenamento inserito cliccando su modifica</li> <li>6. Cambiare i dati e salvare</li> <li>7. Cliccare su Giocatori</li> <li>8. Visualizzare la lista e provare ad aggiungere o cancellare un giocatore</li> <li>9. Tornare su allenamento e cliccare su Esercizi</li> <li>10. Visualizzare la lista e provare ad aggiungere o a cancellare un esercizio dall'allenamento</li> <li>11. Tornare indietro e provare a cancellare un allenamento</li> </ol>		
<b>Risultati attesi:</b>	<p>Mi aspetto il seguente risultato dai punti:</p> <ol style="list-style-type: none"> <li>1. Sia possibile fare l'accesso al sito con un account di allenatore</li> <li>2. La pagina allenamento si aaccessibile</li> <li>3. Sia possibile cliccare sul bottone nuovo allenamento</li> <li>4. Si possa inserire dei dati e salvare il nuovo allenamento. I dati visualizzati devono essere quelli inseriti</li> <li>5. Cliccando sul bottone modifica, sia possibile modificare un allenamento</li> <li>6. Cliccando su salva i dati modificati devono essere uguali a quelli visualizzati</li> <li>7. Sia possibile cliccare su giocatori</li> <li>8. Si possa visualizzare la lista dei giocatori associati e gestirli</li> <li>9. Sia possibile cliccare sul bottone esercizi</li> <li>10. Si possa visualizzare la lista degli esercizi associati e gestirli</li> <li>11. Sia possibile cancellare un allenamento</li> </ol>		

<b>Test Case:</b>	TC-014	<b>Nome:</b>	Report
<b>Riferimento:</b>	REQ-008		
<b>Descrizione:</b>	Il sito deve prevedere una pagina dove poter caricare, scaricare e gestire i vari report		
<b>Prerequisiti:</b>	Avere un account con i poteri di gestore almeno		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Fare accesso al sito con l'account di gestore</li> <li>2. Andare in report</li> <li>3. Controllare che si possa inserire un nuovo report, tramite bottone upload</li> <li>4. Controllare che si possa scaricare un report, tramite bottone download</li> <li>5. Andare nella pagina di utenti e cliccare su lista convocati</li> <li>6. Andare in impostazioni e cliccare lista utenti</li> </ol>		
<b>Risultati attesi:</b>	<p>Mi aspetto i seguenti risultati per i vari punti:</p> <ol style="list-style-type: none"> <li>1. Sia possibile fare l'accesso</li> <li>2. La pagina report sia accessibile</li> <li>3. Sia possibile caricare un nuovo report con il bottone upload</li> <li>4. Sia possibile scaricare un report, tramite il bottone download</li> <li>5. Si presentino i vari tagliandi per le convocazioni</li> <li>6. Si presentit una lista di utenti</li> </ol>		

<b>Test Case:</b>	TC-015	<b>Nome:</b>	Calendario
<b>Riferimento:</b>	REQ-009		
<b>Descrizione:</b>	Il sito deve prevedere un calendario dove rappresentare tutti gli impegni della società, in questo modo anche gli utenti possono vedere questa pagina.		
<b>Prerequisiti:</b>	Bisogna avere un utente registrato		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Fare login al sito</li> <li>2. Accedere alla pagina calendario.php</li> <li>3. Cliccare su un evento: label colore marrone e vederne le informazioni</li> <li>4. Cliccare su una partita: label color rosa e vederne le informazioni</li> <li>5. Cliccare su un allenamento: label color blu e vederne le informazioni</li> <li>6. Provare a modificare il calendario</li> </ol>		
<b>Risultati attesi:</b>	<p>Mi aspetto i seguenti risultati per i punti:</p> <ol style="list-style-type: none"> <li>1. Sia possibile fare il login</li> <li>2. Sia possibile accedere alla pagina calendario.php</li> <li>3. Sia possibile cliccare su un evento e vederne le informazioni</li> <li>4. Sia possibile cliccare su una partita e vederne le informazioni</li> <li>5. Sia possibile cliccare su un allenamento e vederne le informazion</li> <li>6. Sia possibile modificare il calendario</li> </ol>		

<b>Test Case:</b>	TC-016	<b>Nome:</b>	Partite
<b>Riferimento:</b>	REQ-010		
<b>Descrizione:</b>	Il sito deve prevedere una parte in cui poter tenere le informazioni delle varie partite e i risultati delle stesse		
<b>Prerequisiti:</b>	Avere un account con i permessi almeno di allenatore		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Dopo aver fatto l'accesso al sito come allenatore, andare nella pagina partita.php e vederne le informazioni</li> <li>2. Cliccare su giocatori e provare a modificare la lista</li> <li>3. Provare a generare la lista dei convocati con il link sulla destra</li> <li>4. Tornare indietro e cliccare su staff e provare a modificare la lista</li> <li>5. Andare in dettaglio e inserire un commento alla partita</li> <li>6. Provare a modificare il commento</li> <li>7. Provare a inserire delle immagini</li> <li>8. Tornare indietro e cancellare la partita</li> </ol>		
<b>Risultati attesi:</b>	<p>Mi aspetto i seguenti risultati per i vari punti:</p> <ol style="list-style-type: none"> <li>1. Sia possibile fare l'accesso alla pagina dopo aver fatto login e le informazioni devono essere uguali a quelle inserire</li> <li>2. Sia possibile visualizzare e modificare la lista dei convocati, impostando anche se arbitro o meno</li> <li>3. Cliccando sul link sulla destra viene generato il pdf</li> <li>4. Sia possibile visualizzare e modificare la lista dello staff, impostando i vari ruoli tra cui anche commissario</li> <li>5. Sia possibile visualizzare il dettaglio della partita e aggiungere i vari commenti (risultato, commento e foto)</li> <li>6. Sia possibile modificar e un commento e salvarne le informazioni nuove</li> <li>7. Sia possibile caricare un immagine</li> <li>8. Sia possibile cancellare una partita</li> </ol>		

#### 4.1 Risultati dei Test Case

Test Case	Nr Passaggio	Risultato
TC-001	1	È possibile raggiungere il sito tramite il link
	2	La pagina calendario.php all'esterno del sito, non è accessibile, bisogna fare per forza l'accesso.
	3	La pagina presenta un login
	4	È possibile fare l'accesso al sito con quell'account
TC-002	Tutti i punti (1-4)	Non è stato testato, in quanto erroneamente sono stati scritti questi requisiti poi commentati con il relatore. Il sito di presentazione della società esiste già, quindi le pagine rimangono quelle private
TC-003	Tutti i punti dal 1-14	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.



TC-004	Punti: 3, 4, 5, 7, 9, 10, 12	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
	Punti: 2, 11, 13, 1	È possibile fare l'accesso alla pagina, ma senza eseguire le operazioni (modifica, cancellazione e aggiunta)
	Punti: 14, 16	È possibile fare l'accesso alla pagina e modificare gli input
	Punto 8	È possibile fare l'accesso, questo perché è stato poi deciso che l'utente può visualizzare i dettagli, ma non eseguire le operazioni. Inizialmente ci aspettava di negare completamente il permesso
TC-005	Punti: 3, 4, 5, 7, 9, 10, 12	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
	Punti: 2, 11, 1	È possibile fare l'accesso alla pagina, ma senza eseguire le operazioni (modifica, cancellazione e aggiunta)
	Punti: 14, 16, 13	È possibile fare l'accesso alla pagina e modificare gli input
	Punto 8	È possibile fare l'accesso, questo perché è stato poi deciso che l'utente può visualizzare i dettagli, ma non eseguire le operazioni. Inizialmente ci aspettava di negare completamente il permesso
TC-006	Punti: 3, 4, 5, 7, 12	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
	Punti: 1, 9	È possibile fare l'accesso alla pagina, ma senza eseguire le operazioni (modifica, cancellazione e aggiunta)
	Punti 2, 6, 8, 10, 11, 13, 14	È possibile fare l'accesso alla pagina e modificare gli input
TC-007	Punti: 4, 5, 7, 12	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
	Punti: 1, 9	È possibile fare l'accesso alla pagina e modificare gli input
	Punti 2, 3, 6, 8, 10, 11, 13, 14	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
TC-008	Punti: 12	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
	Punti: 1	È possibile fare l'accesso alla pagina e modificare gli input
	Punti: 2,3,4,5,6,7,8,9,10,11,13,14	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.

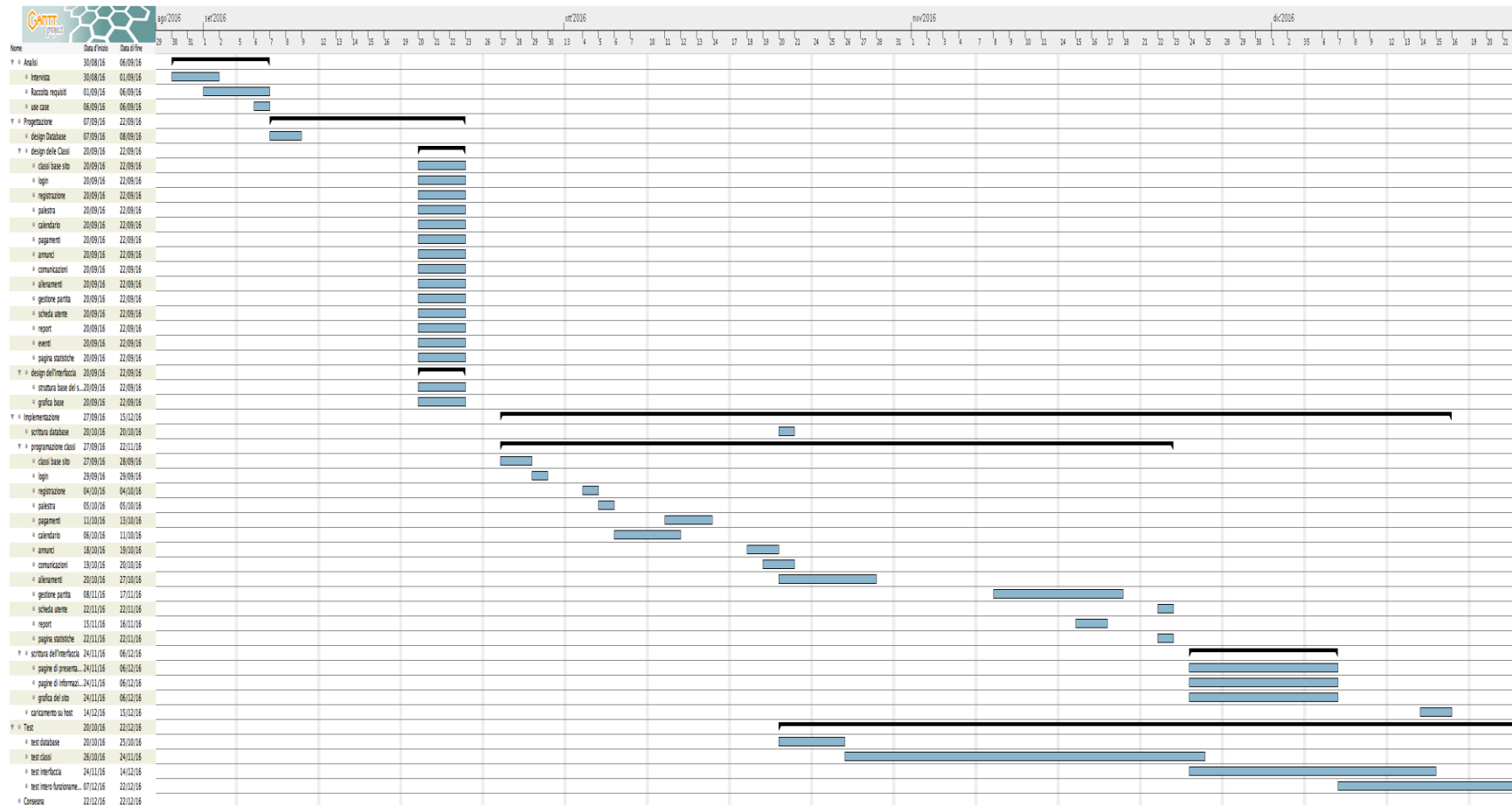
TC-009	Punti: 1	È possibile fare l'accesso alla pagina e modificare gli input
	Punti: 2,3,4,5,6,7,8,9,10,11,12,13,14	In tutte le pagine ritorna correttamente la frase d'errore, di dover fare il login.
TC-010	Punti: 1-5	Il sito prevede correttamente la possibilità di aggiungere e modificare i vari ruoli agli utenti. I ruoli sono quelli aspettati
TC-011	Punti:1-2	Il sito offre una grafica molto minimale, ma funzionale allo scopo per la quale è stato progettato
TC-012	Punti: 1-4	Il sito non offre la possibilità di fare delle comunicazioni direttamente con gli utenti, questo perché per questioni di tempo e di capacità, non si è riusciti a sviluppare questo automatismo. In cambio è stata sviluppata la pagina di comunicazioni, per mandare delle circolari ai vari gruppi di utenti
TC-013	Punti:1-11	Il sito offre la possibilità di gestire gli allenamenti e tutte le informazioni che stanno attorno, come gli esercizi, giocatori e varie date
TC-014	Punti:1-6	Il sito offre la possibilità di caricare, scaricare e cancellare i vari report utili. Inoltre i link e i pdf funzionano correttamente
TC-015	Punti:1-6	Il sito offre la possibilità di gestire le informazioni all'interno del calendario e fa visualizzare anche le partite e allenamenti.
TC-016	Punti: 1-8	Il sito offre la possibilità di gestire la partita e tutte le informazioni che sono a essere collegate, come ad esempio la lista convocati, lo staff e le varie info base

## **5 Consuntivo**

---

Possiamo notare che il consuntivo finale è molto differente a quello previsto durante l'analisi a inizio progetto, questo perché avevo dato troppa importanza alla parte di progettazione, senza pensare che avrei potuto trovare molti problemi durante la programmazione o comunque, che essa prendeva molto più tempo di quanto avessi previsto. Dopo aver discusso con il mio relatore, ho concordato di tagliare la progettazione e cominciare direttamente con l'implementazione delle pagine e della logica che sta dietro. Alcune parti, come ad esempio la comunicazione via email/sms/whatsapp automatica, non sono riusciti a implementarla perché abbiamo perso alcune giornate e nel complesso, mi sono visto costretto a dare importanza altre cose, così ho avviato questa parte creando una pagina di comunicazioni manuale.

Per il consuntivo finale vedere la pagina successiva.



**Titolo del progetto:** Società di Basket  
**Alunno/a:** Simone Esposito  
**Classe:** Info 4  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Adriano Barchi

## 6 Conclusioni

---

Questo sito offre tutto ciò che una società di Basket necessita per gestire tutta la parte cartacea o di statistiche fatte fino a ora a mano. Inoltre è un possibile punto di consultazione delle varie partite, commenti e fotografie per rivivere emozioni passate e magari, per tenere sott'occhio alcune future promesse. Questo sito sicuramente faciliterà tutto ciò che è la gestione della società o l'iterazione tra i vari utenti.

### 6.1 Sviluppi futuri

Il progetto era molto limitato, ma sicuramente come implementazioni future, si potrebbe aggiungere la possibilità di creare un live delle partite. Dunque ampliare ciò che è l'attuale figura di blogger, il writer, concedendogli la possibilità sul sito o su una applicazione mobile, di commentare in live la partita, inserendo i vari canestri, chi sta giocando e magari qualche fotografia, così da permettere a genitori, o amici, di vivere la partita anche se lontani dalla palestra.

Un'altra implementazione possibile è quella di inserire l'automatismo all'interno delle comunicazioni e creare anche qua un'applicazione per aiutare in questo lavoro, così si possa avere un rapporto più stretto fra tutti i membri della comunità, magari con un piccolo "social" di gestione.

Trovo che come ultima implementazione futura, sempre a lato web e app, sia comodo poter offrire delle statistiche più personalizzate per ogni utente e renderle visibili a chiunque necessiti di tali informazioni, questo per aiutare a far crescere i giocatori.

### 6.2 Considerazioni personali

Da questo progetto ho appreso molto l'importanza di effettuare un'accurata pianificazione e a dare l'importanza giusta alle varie parti, senza evitare di dover correre e magari eseguire male il lavoro, perché si aveva considerato di concludere tutto quanto.

Ho avuto finalmente il piacere di affrontare un intero sito internet lavorando a classi in php e devo ammettere che è stato molto soddisfacente, perché una volta create le basi, tutto il resto del sito comincia a svilupparsi molto facilmente.

## 7 Bibliografia

---

### 7.1 Sitografia

- <http://stackoverflow.com/>, *Stack OverFlow*, da 01.09 al 20.12
- <http://www.w3schools.com/>, *W3Schools*, da 01.09 al 20.12
- <http://php.net/>, *PHP*, da 01.09 al 20.12
- <http://www.chartjs.org/>, *Chart.Js*, Novembre 2016
- <https://fullcalendar.io/>, *FullCalendar*, da Settembre 2016 a Ottobre 2016
- <https://bitbucket.org/>, *BitBucket*, da 01.09 al 20.12

## 8 Allegati

---

Elenco degli allegati

- Diari di lavoro
- Codici sorgente
- Quaderno dei compiti
- Prodotto

**Titolo del progetto:** Società di Basket  
**Alunno/a:** Simone Esposito  
**Classe:** Info 4  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Adriano Barchi