



## UNIVERSITÀ DEGLI STUDI DI MESSINA

DIPARTIMENTO DI SCIENZE MATEMATICHE E INFORMATICHE,  
SCIENZE FISICHE E SCIENZE DELLA TERRA

Corso di Laurea Triennale in Informatica

---

## Relazione Programmazione Web e Mobile

Luigi  
Villari  
Matricola 532465

Antonio  
Santagati  
Matricola 531795

---

ANNO ACCADEMICO, 2023/2024

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Tecnologie Impiegate</b>	<b>3</b>
2.1	Front end: HTML e CSS . . . . .	3
2.2	Front end: Javascript . . . . .	3
2.3	Back end: PHP . . . . .	3
2.4	Back end: Database MySQL . . . . .	3
2.5	Architettura RESTful . . . . .	3
<b>3</b>	<b>Interfaccia</b>	<b>4</b>
3.1	Homepage . . . . .	4
3.2	Creazione Utente . . . . .	5
3.3	Cancellazione Utente . . . . .	6
3.4	Modifica Utente . . . . .	7
3.5	Difficoltà . . . . .	8
3.6	Leaderboard . . . . .	9
3.7	Modalità: Training/Match . . . . .	10
3.8	Selezione Utente . . . . .	11
<b>4</b>	<b>Script</b>	<b>11</b>
<b>5</b>	<b>Database</b>	<b>12</b>
5.1	Definizione Schema E-R . . . . .	12
5.2	Struttura finale DB . . . . .	13
<b>6</b>	<b>Servizio Web: RESTful</b>	<b>14</b>
6.1	Metodo: GET . . . . .	15
6.2	Metodo: POST . . . . .	16
6.3	Metodo: PUT . . . . .	17
6.4	Metodo: DELETE . . . . .	18

# 1 Introduzione

Il progetto mira alla realizzazione di un gioco basato sul noto memory game. Diamo agli utenti la possibilità di poter immergersi in sessioni di gioco in single-player e in multiplayer permettendo di poter creare un proprio account sul quale salvare i progressi, sfidare amici e migliorare fino a scalare le vette più alte della classifica. Comprende anche diverse difficoltà per permettere a chiunque di giocare al proprio ritmo.

## 2 Tecnologie Impiegate

### 2.1 Front end: HTML e CSS

Per la gestione del gioco abbiamo optato per rendere responsivi i file CSS presenti all'interno del nostro progetto e, tramite uno scheletro fornito dai file HTML, siamo riusciti a creare un'infrastruttura chiara e facile da utilizzare.

### 2.2 Front end: Javascript

Javascript è stato impiegato per integrare funzionalità interattive , dinamiche e veloci. Ha consentito la creazione delle carte ,delle coppie e disporle nel campo da gioco in posizioni casuali. Il giocatore infatti , dopo aver guardato le carte per un lasso di tempo variabile in base alla difficoltà selezionata, è subito lanciato all'interno del gioco.

### 2.3 Back end: PHP

PHP è stato invece utilizzato per gestire le richieste del Server in quanto è in grado di comunicare direttamente col database MySQL. Ci ha permesso di salvare tutte le informazioni utili riguardanti il giocatore e le partite permettendoci un accesso e un utilizzo veloce ed efficiente dei vari dati.

### 2.4 Back end: Database MySQL

MySQL è stato utilizzato per memorizzare i dati dell'applicazione, consentendo a PHP di interagire con il database per gestire le informazioni necessarie.

### 2.5 Architettura RESTful

Utilizzata per progettare l'interfaccia tra i sistemi, consentendo la comunicazione e lo scambio di dati in un formato standardizzato e scalabile attraverso richieste HTTP ben definite.

### 3 Interfaccia

L’interfaccia del Memory Game è stata attentamente progettata per offrire un’esperienza d’uso intuitiva e fluida. Basata su un design responsivo, si adatta perfettamente a diversi dispositivi, garantendo un’esperienza ottimale sia su desktop che su smartphone. Le sue principali componenti sono:

- Homepage
- Creazione utente
- Cancellazione utente
- Modifica Utente
- Difficoltà
- Leaderboard
- Training/Match
- Selezione Utente

#### 3.1 Homepage

La Homepage è stata progettata per trasmettere al giocatore la sensazione di essere in un videogame Retro e con l’intento di renderla facile da usare e leggibile con un layout intuitivo. Creato in modo tale che anche un’utente da Telefono o da Tablet possa facilmente accedervi e utilizzarne tutte le sue funzioni.



(a) Home vista da Laptop.



(b) Home vista da Mobile.

### 3.2 Creazione Utente

La sezione di registrazione si trova all'interno del menù utente. Permette di creare il proprio account con la restrizione di inserire un nome unico e mai utilizzato così da poter essere differenziato dagli altri utenti.



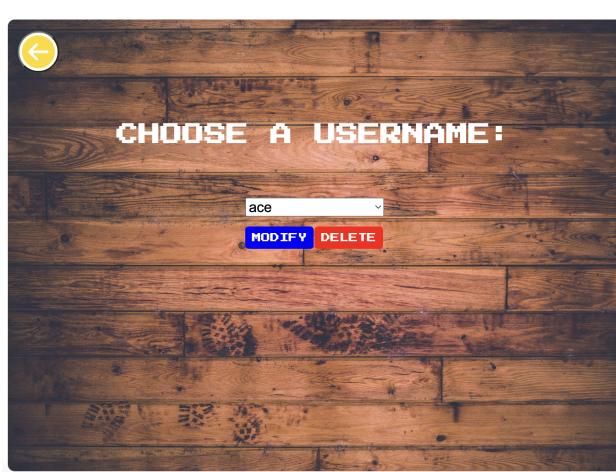
(a) Registrazione vista da Laptop.



(b) Registrazione vista da Mobile.

### 3.3 Cancellazione Utente

La sezione cancellazione utente è stata pensata per rimuovere definitivamente un username. Si trova all'interno del menù utente e tramite la funzione "Delete" sarà possibile rimuovere definitivamente e irreversibilmente un utente dal gioco. Questa sezione presenta anche un'altra funzione , la modifica. Essa permetterà dopo aver selezionato l'utente , attraverso il bottone "Modify", di venir reindirizzato verso la vera e propria pagina di modifica dell'utente.



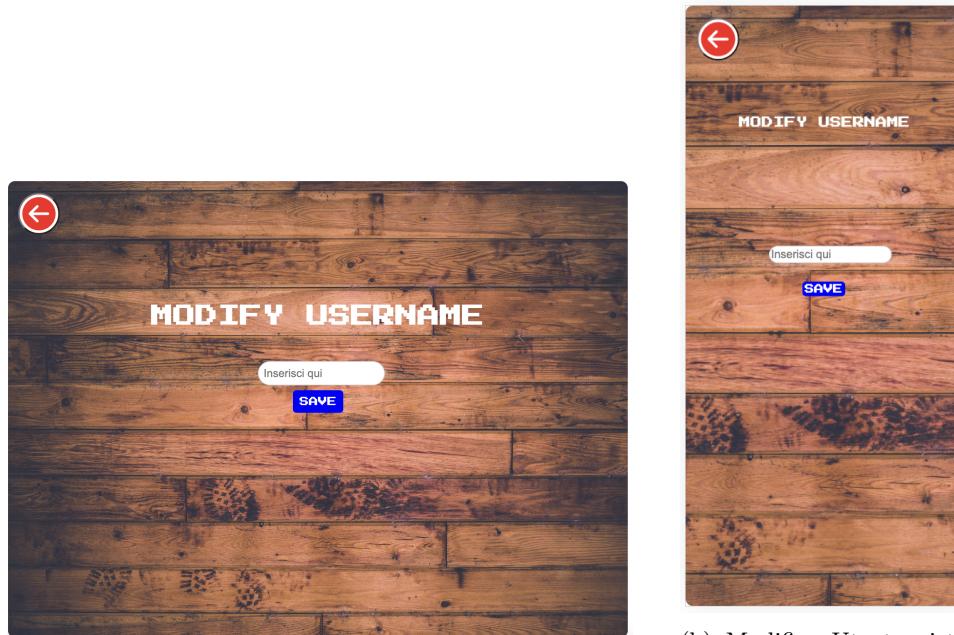
(a) Cancellazione vista da Laptop.



(b) Cancellazione vista da Mobile.

### 3.4 Modifica Utente

La sezione modifica utente permette di modificare l'utente precedentemente selezionato all'interno della sezione di manage. E' stata pensata per essere facile da utilizzare e immediata. Solo inserendo il nuovo username permetterà di sovrascrivere quello vecchio. Subito dopo aver effettuato questa operazione sarà possibile vedere i cambiamenti dalla schermata di selezione utente o dalla funzione "Leaderboard" nel caso in cui l'utente sia in classifica.



(a) Modifica Utente vista da Laptop.

(b) Modifica Utente vista da Mobile.

### 3.5 Difficoltà

Questa sezione è stata pensata per permettere ai giocatori di selezionare la difficoltà desiderata, così che durante la partita possano giocare con le impostazioni scelte. Il layout della pagina sfrutta dei bottoni che permettono un facile utilizzo e una semplice lettura. L'accesso a questa sezione è presente all'interno dell'homepage, rendendola ben visibile e permettendo a chiunque di trovarla e di accedervene in qualsiasi momento. Nel caso in cui un giocatore dovesse dimenticarsi di selezionare una difficoltà specifica, all'interno della partita, verrà selezionata una difficoltà di default.



(a) Scelta difficoltà vista da Laptop.



(b) Scelta difficoltà vista da Mobile.

### 3.6 Leaderboard

La leaderboard è una sezione fondamentale del gioco che permette a tutti i giocatori di confrontarsi tra di loro. La posizione in classifica è determinata dal numero di vittorie e permette a chiunque di vedere i migliori dieci giocatori.



Username	Vittoria	Sconfitta
sili	2	1
piero	1	1
peppa pig	1	0
ale	1	2
silvia	0	4
ciccio	0	0
papa	0	1
mamma	0	0
giovanni	0	0
Domenico	0	0

(a) Leaderboard vista da Laptop.

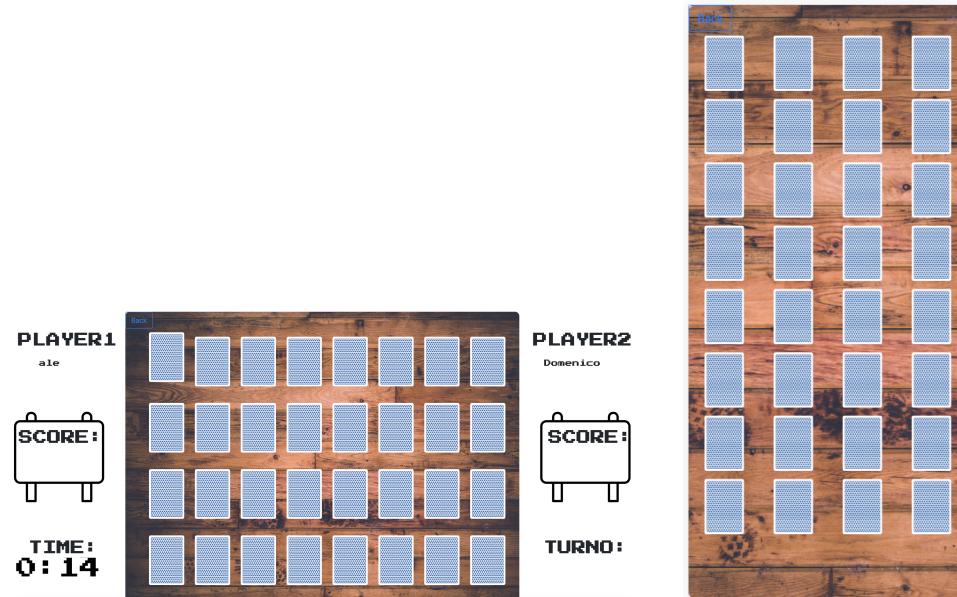


Username	Vittoria	Sconfitta
sili	2	1
piero	1	1
peppa pig	1	0
ale	1	2
silvia	0	4
ciccio	0	0
papa	0	1
mamma	0	0
giovanni	0	0
Domenico	0	0

(b) Leaderboard vista da Mobile.

### 3.7 Modalità: Training/Match

Le sezioni Training e Match sono dove avviene la vera e propria partita. La prima permette al giocatore di sfidare se stesso in una prova a tempo che gli permetterà di migliorare la sua memoria a breve termine per provare a memorizzare e trovare il maggior numero di coppie nel minor tempo possibile. Il Match, invece, permette a due giocatori registrati di sfidarsi. Il risultato della partita influirà sulla classifica. La prima modalità avrà un'impostazione semplice per permettere al giocatore di focalizzarsi sulla partita mostrando a schermo solo informazioni principali come il Turno attuale oppure il tempo trascorso dall'inizio della partita. La seconda oltre ad avere queste features, possiederà anche altre informazioni come gli username e il punteggio in base alle coppie trovate dai vari giocatori.



(a) Partita vista da Laptop.

(b) Partita vista da Mobile.

### 3.8 Selezione Utente

La selezione utente è la sezione creata per permettere ai due giocatori di scegliere il proprio username prima di uno scontro Match. Dopo la scelta di entrambi sarà possibile premere il tasto seleziona per poter effettivamente iniziare la partita.



(a) Selezione Utente vista da Laptop.



(b) Selezione Utente vista da Mobile.

## 4 Script

Il funzionamento del gioco si basa su diversi Script.

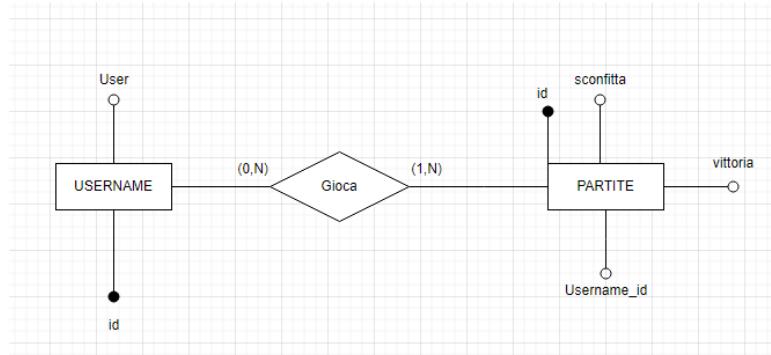
- **Home.js:** E' utilizzato per la gestione di tutte le funzionalità principali del gioco. Come se fosse un vero e proprio pannello di controllo. Offre l'accesso alle impostazioni dello User e a tutte le modalità di gioco.
- **difficulty.js:** Permette al giocatore di scegliere tra le varie difficoltà di gioco.
- **manage user.js:** Si occupa di gestire scelta tra la modifica e l'eliminazione di un utente preselezionato.
- **modify user.js:** Si occupa nello specifico di tutte le operazioni di modifica di un utente, permettendo di scegliere il nuovo username e tramite una richiesta AJAX effettua un "UPDATE" dello User.
- **username.js:** Si occupa della svolgimento di tutte le operazioni per la creazione di un utente.

- **leaderboard.js**: Permette di caricare tutte le informazioni riguardanti la classifica.
- **select user.js**: Permette attraverso una richiesta AJAX per ottenere una lista dei giocatori presenti e disponibili.
- **memory single.js**: E' lo script utilizzato per giocare alla modalità training.
- **memory.js**: E' lo script principale che permette a due user di sfidarsi nella modalità match, gestisce tutte le funzionalità di gioco.

## 5 Database

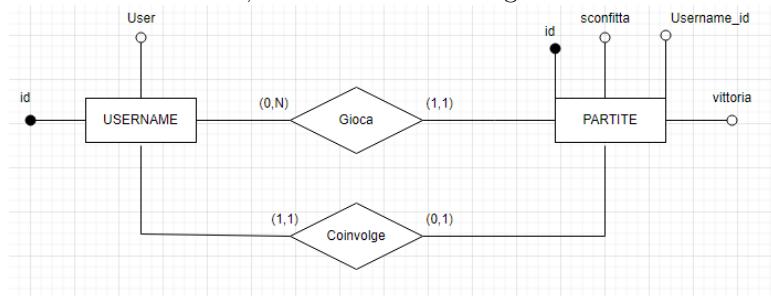
### 5.1 Definizione Schema E-R

Il database è stato implementato utilizzando MySQL. La progettazione è descritta dal seguente schema E-R:



(a) Schema E-R

Successivamente, attraverso un processo di ristrutturazione, lo schema è stato modificato, evolvendo verso la seguente forma:



(b) Schema E-R Ristrutturato

## 5.2 Struttura finale DB

Successivamente alla ristrutturazione sono state definite le tabelle andando ad inserire le adeguate foreign key. Di seguito la struttura finale delle tabelle:

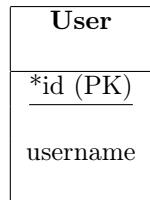


Tabella 1: Tabella User

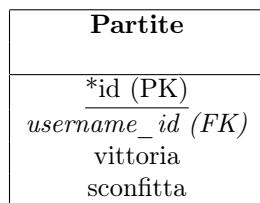


Tabella 2: Tabella Partite

Le foreign key sono indicate con \*, mentre le primary key sono sottolineate

## 6 Servizio Web: RESTful

All'interno del progetto, le API RESTful sono state ampiamente impiegate per semplificare e gestire le connessioni e le interazioni con il database, al fine di creare il servizio web. Le API RESTful rappresentano un metodo standard per sviluppare servizi web, offrendo un'architettura flessibile e scalabile per applicazioni web e mobile. Le funzionalità del gioco, come la creazione dell'utente, la selezione dei player per il match, la modifica dello user e altre operazioni, sono state implementate come risorse accessibili tramite API RESTful. Questo approccio ha consentito un accesso agevole e strutturato ai dati, garantendo una netta separazione tra il front-end e il back-end del sistema. Nell'architettura dei servizi RESTful, uno dei principi fondamentali è l'utilizzo delle URI (Uniform Resource Identifier) per identificare le risorse. Ogni richiesta inviata al backend tramite AJAX utilizza un URI che non solo si riferisce al file responsabile della gestione della richiesta, ma contiene anche informazioni aggiuntive

Di seguito un esempio del URI utilizzato:

```
req.open("POST", ".../php/add_win.php/Partite", true);
```

Un'altro dei vantaggi dei servizi Restful è l'utilizzo esplicito dei metodi HTTP. Nel progetto infatti ad ogni richiesta effettuata al database è associato il metodo HTTP adeguato.

## 6.1 Metodo: GET

```
function selected_user(){
    var req = new XMLHttpRequest(); //creo un oggetto XMLHttpRequest
    req.onload = function() { //definisce l'azione da eseguire una volta completata la richiesta
        if(req.status == 200 ){
            var data = JSON.parse(this.responseText); //parsing dei dati JSON
            visual(data); //invocazione funzione visual
        }
    }
    req.open('GET','../php/select_user.php/Username',true); //richiesta GET al server PHP
    req.send(); //invia richiesta
}
```

(a) Metodo GET

```
if ($method == "GET" && $table == "Username") {
    // Costruisci la query SQL per selezionare la colonna 'user' dalla tabella 'Username'
    $query = "SELECT user FROM Username";

    // Prepara la query
    $stmt = mysqli_prepare($conn, $query);
    // Esegui la query preparata
    mysqli_stmt_execute($stmt);

    // Ottieni il risultato della query
    $result = mysqli_stmt_get_result($stmt);

    // Inizializza un array per contenere i risultati della query
    $array = [];

    // Itera attraverso i risultati e aggiungi ciascuna riga all'array
    while ($row = mysqli_fetch_assoc($result)) {
        $array[] = $row;
    }

    // Converti l'array in formato JSON
    $json = json_encode($array);

    // Stampa il risultato JSON
    echo $json;
}
```

(b) Lato server associato al metodo GET

## 6.2 Metodo: POST

```
function recupero_dati_form(){
    //creazione dell'oggetto con i valori dei campi del form
    var data={};
    data.username = InputUsername.value; //salva nell'array data il valore di Username
    var JSONdata = JSON.stringify(data); //converte data in una stringa JSON
    var req = new XMLHttpRequest(); // Crea un oggetto XMLHttpRequest
    if(data.username!="") && data.username!=null){ //verifico che data non sia null
        req.onload = function(){ //definisce l'azione da eseguire una volta completata la richiesta
            console.log(req.responseText);
            var res = req.responseText; //metto all'interno della variabile res la responseText
            if(res.trim() == "ok"){ //verifica che tutto sia andato correttamente
                alert("Utente inserito correttamente");
                window.location.href = "../html/home.html"; //reindirizzamento alla pagina specificata
            }
            else{
                alert("Username non valido o già esistente");
            }
        }
        req.open("POST", "../php/add_user.php?Username", true); //richiesta POST al server php
        req.send(JSONdata); //invio richiesta con dati in formato JSON
    }
    else{
        const messageError = document.getElementById("alert"); // Gestisco il possibile caso in cui data sia null
        messageError.style.display = "block";
    }
}
```

(a) Metodo POST

```
// Se il metodo è POST e la tabella è "Username"
if ($method === "POST" && $table === "Username") {
    // Query per inserire un nuovo utente nella tabella 'Username'
    $query = "INSERT INTO Username (user) VALUES (?)";
    $stmt = mysqli_prepare($conn, $query); // Prepara la query
    mysqli_stmt_bind_param($stmt, 's', $input['username']); // Collega i parametri alla query preparata
    mysqli_stmt_execute($stmt); // Esegui la query preparata
}
```

(b) Lato server associato al metodo POST

### 6.3 Metodo: PUT

```

function modifyUser(){
    //creazione dell'oggetto con i valori dei campi del form
    var data={};
    data.username = user;//salva nell'array data il valore di Username
    data.new = Input.value;//salva nell'array data il valore di value
    var JSONdata = JSON.stringify(data);//converte data in una stringa JSON
    var req = new XMLHttpRequest(); // Crea un oggetto XMLHttpRequest
    if(data.username!="" && data.username!=null && data.new!=null && data.new!=""){
        //Eseguo una se
        req.onload = function(){ //definisce l'azione da eseguire una volta completata la richiesta
            console.log(req.responseText);
            var res = req.responseText;//metto all'interno della variabile res la responseText
            if(res.trim() == "ok"){
                //verifica che tutto sia andato correttamente
                alert("Utente modificato correttamente");
                window.location.href = "../html/home.html"; //reindirizzamento alla pagina specificata
            }
        }
        else{
            You, 2 days ago • last update, miss only control for register/modify...
            alert("Username non valido o già esistente");
        }
    }
    req.open("PUT", "../php/modify_user.php?Username", true); //richiesta POST al server php
    req.send(JSONdata); //invio richiesta con dati in formato JSON
}

```

(a) Metodo PUT

```

if ($method == "PUT" && $table == "Username"){
    $query = "UPDATE Username SET user = ? WHERE user = ?"; // Costruisce la query SQL per selezionare la colonna 'user'
    $stmt = mysqli_prepare($conn, $query); // Prepara la query
    mysqli_stmt_bind_param($stmt, 'ss', $input['new'], $input['username']); // Collega i parametri alla query preparata
    if(mysqli_stmt_execute($stmt)){ //Effettua un controllo sulla query eseguita per stampare un messaggio di conferma o
        echo 'ok';
    }
    else{
        mysqli_error($conn);
        echo 'error';
    }
}

```

(b) Lato server associato al metodo PUT

## 6.4 Metodo: DELETE

```

function delete_user(){
    var data={};
    data.username = userList.value; //salva nell'array data il valore di userList
    var JSONdata = JSON.stringify(data); //converte data in una stringa JSON
    var req = new XMLHttpRequest(); // Crea un oggetto XMLHttpRequest
    if(data.username!="") && data.username!=null){ //verifico che data non sia null
        req.onload = function(){ //definisce l'azione da eseguire una volta completata la richiesta
            if (req.status == 200){
                console.log(req.responseText);
                var res = req.responseText;
            }
            if(res.trim()=="ok"){ //verifica che tutto sia andato correttamente
                alert("Utente rimosso con successo");
                window.location.href="../html/home.html"; //reindirizza alla pagina specificata nell'url
            }
            else{
                alert("error");
            }
        }
        req.open("DELETE","../php/manage_user.php/Username",true); //richiesta DELETE al server php
        req.send(JSONdata); //invio richiesta con dati in formato JSON
    }
}

```

(a) Metodo DELETE

```

// Se il metodo è DELETE e la tabella è "Username"
if ($method == "DELETE" && $table == "Username") {
    // Query per selezionare l'id corrispondente all'username fornito
    $query1 = "SELECT id FROM Username WHERE user = ?";
    $stmt1 = mysqli_prepare($conn, $query1); //Prepara la query
    mysqli_stmt_bind_param($stmt1, 's', $input['username']); // Collega i parametri alla query preparata
    mysqli_stmt_execute($stmt1); //Esegui la query
    mysqli_stmt_bind_result($stmt1, $id); // Associa il risultato della query preparata ($stmt1) alla variabile $id
    You, 22 hours ago • comments code
    mysqli_stmt_fetch($stmt1); // Recupera il risultato
    mysqli_stmt_close($stmt1); // Chiudi la query
    $id = intval($id); // Converti $id in un intero, se necessario
    // Query per eliminare le righe corrispondenti nella tabella 'Partite'
    $query2 = "DELETE FROM Partite WHERE username_id = ? ";
    $stmt2 = mysqli_prepare($conn, $query2); //Prepara la query
    mysqli_stmt_bind_param($stmt2, 'i', $id); // Collega i parametri alla query preparata
    mysqli_stmt_execute($stmt2); //Esegui la query

    // Verifica se sono state eliminate righe da 'Partite'
    if (mysqli_stmt_affected_rows($stmt2) > 0) {
        // Query per eliminare l'utente dalla tabella 'Username'
        $query3 = "DELETE FROM Username WHERE id = ? ";
        $stmt3 = mysqli_prepare($conn, $query3); //Prepara la query
        mysqli_stmt_bind_param($stmt3, 'i', $id); // Collega i parametri alla query preparata

        // Esegui la query e verifica se l'eliminazione è avvenuta con successo
        if (mysqli_stmt_execute($stmt3)) {
            echo "ok";
        }
    }
}

```

(b) Lato server associato al metodo DELETE