UNIVERSITÀ DEGLI STUDI DI MESSINA

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

Corso di Laurea Triennale in Informatica

Progetto Programmazione Web e Mobile

Francesco Maria
Russo
Matr. 526864

Domenico
Villari
Matr. 527221

ANNO ACCADEMICO 2023/2024

Indice

1 Introduzione

Il progetto mira a creare una piattaforma web dedicata al fitness, fornendo agli utenti uno strumento completo per tracciare i progressi e ottimizzare gli allenamenti. Funzionando come un diario personalizzato, consentirà agli utenti di registrare dettagli degli allenamenti, monitorare miglioramenti nel tempo e accedere a risorse utili. L'obiettivo principale è offrire un'esperienza intuitiva, consentendo agli utenti di pianificare, registrare e analizzare facilmente le attività fisiche.
Tutto il codice è possibile visionarlo nella relativa repository di GitHub.

2 Tecnologie Utilizzate

2.1 Front end: HTML e CSS

I file HTML e CSS sono derivati a partire dal template Bootstrap 4 denominato 'sb-admin-2' (disponibile su GitHub). L'utilizzo di questo template ha garantito un design responsivo, ottimizzando l'adattabilità e la visualizzazione del sito su varie piattaforme e dimensioni dello schermo. Questa scelta non solo ha assicurato una maggiore fruibilità su dispositivi diversi, ma ha anche accelerato significativamente il processo di sviluppo del sito.

2.2 Front end: JavaScript

JavaScript è stato impiegato per integrare funzionalità dinamiche e interattive, migliorando l'esperienza utente attraverso una serie di interazioni sul sito. Ha consentito la costruzione dinamica delle pagine e la personalizzazione dei contenuti in base alle azioni degli utenti.

2.3 Back end: PHP

PHP è stato impiegato per gestire le richieste al server, sfruttando la sua capacità di interagire direttamente con il database MySQL. Ciò ha permesso un'efficace manipolazione dei dati, consentendo l'archiviazione, l'accesso e la modifica efficiente delle informazioni fornite dall'applicazione

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 Stile architetturale: 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 di FitnessFolio 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 tablet e smartphone. Le sue principali componenti sono:

- Topbar. Essa include il pulsante di "Go Back" e il nome e cognome dell'utente. Se si clicca su questi ultimi è possibile modificare il profilo o effettuare il logout;
- Sidebar. Contiene i riferimenti a tutte le funzionalità del sito come:
  - Dashboard;
  - Schede;
  - Allenamenti;
  - Esercizi.
- Contenuto principale. Questo cambia da pagina a pagina in base a quello che l'utente deve compiere.

Il sito usa principalmente la classi "card" di Bootstrap 4 per organizzare i contenuti all'interno della pagina.

3.1 Register

La pagina della registrazione di un utente presenta un form con 6 campi:

- Nome;
- Cognome;
- Email;
- Numero di telefono(opzionale);
- Password;
- Conferma Password.

Nel campo email viene effettuata una validazione in tempo reale sulla base di quello che scrive l'utente.
La stessa cosa viene effettuata nei campi Password e Conferma Password dove vengono confrontati i valori presenti in essi, al fine di far immettere la stessa password due volte.

3.2 Index

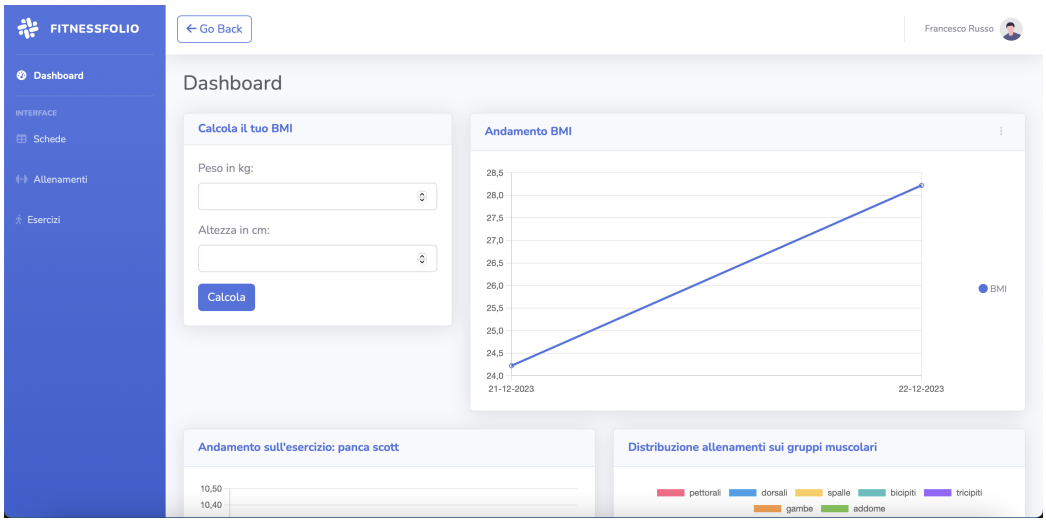

Figura 1: Screen della pagina Index

La pagina index contiene dei grafici per rendere più facile la visione dell'andamento dell'utente su diversi aspetti. Le prime due card riguardando il BMI(Body Mass Index). In particolare la prima permette di effettuare una misurazione, mentre la seconda mostra un grafico con l'andamento di esso.

I grafici a seguire riguardano più nello specifico gli allenamenti dell'utente. È possibile vedere la progressione di un determinato esercizio e la distribuzione dei gruppi muscolari sul totale degli allenamenti effettuati.

3.3 Schede

Nella pagina delle schede è possibile visionare le schede già terminate e, se presente, la scheda attuale. È anche possibile aggiungere una scheda nuova se non si è in possesso di una scheda non terminata.

Nella pagina della scheda attuale è possibile effettuare due operazioni: modificare la scheda o terminarla. La modifica permette sia di modificare le opzioni per gli esercizi già esistenti che aggiungerne di nuovi. Il tasto "Termina Scheda" permette di terminare la scheda attuale con la data odierna.

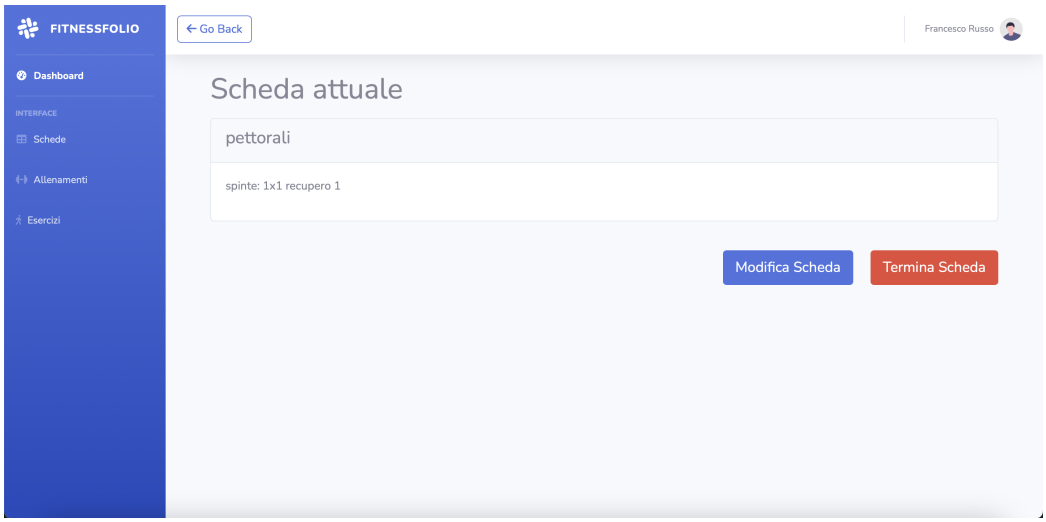

Figura 2: Screen della pagina della scheda attuale

3.4 Allenamenti

Nella pagina Allenamenti è possibile visionare gli allenamenti effettuati in passato. È anche possibile iniziare un nuovo allenamento.

Prima di iniziare un nuovo allenamento viene chiesto all'utente di selezionare i gruppi muscolari da allenare presenti nella sua scheda attuale.

Nella modalità Allenamento l'utente vede:

- Gruppo muscolare e esercizio corrente;
- Gif dell'esercizio corrente;
- Informazioni sull'esercizio;
- Bottone di recupero che avvia il timer sulla base del tempo di recupero;
- Progress-bar per il tempo di recupero;
- Bottone di fine allenamento.

Una volta terminate le serie dell'esercizio corrente, sarà visualizzato il campo per inserire il peso con cui si è effettuato l'esercizio e il bottone per passare all'esercizio successivo.

Figura 3: Screen della modalità allenamento

3.5 Esercizi

La pagina Esercizi dà una panoramica sugli esercizi presenti nel database.

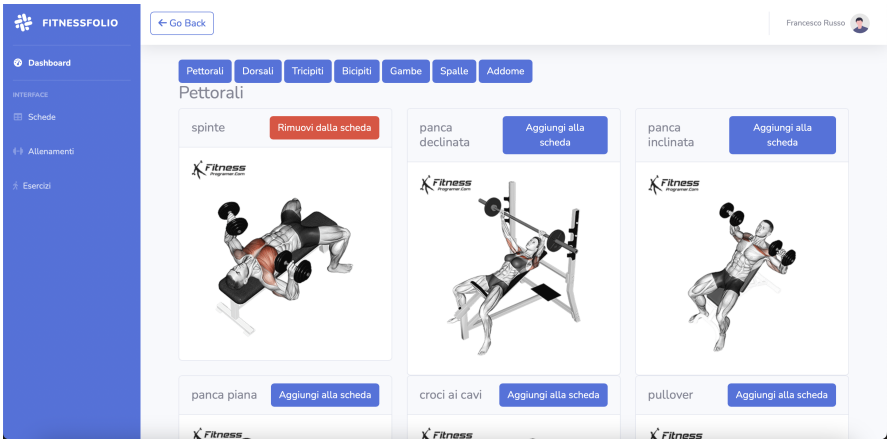

Figura 4: Screen della pagina Esercizi

In alto sono presenti dei bottoni per raggiungere velocemente un determinato gruppo muscolare.
Per ogni esercizio viene creata una card con:

- Nome dell'esercizio;
- Gif dell'esercizio;
- Bottone per aggiungere/rimuovere l'esercizio alla/dalla scheda corrente.

Quando si vuole aggiungere un esercizio apparirà un modal con i campi da riempire per le info dell'esercizio. La stessa cosa accadrà nel caso della rimozione per chiedere conferma all'utente.

4 Script

Per il funzionamento del sito vengono utilizzati vari script, quali:

- **login.js**: Gestisce il login al sito. Recupera i dati dal form, effettua una richiesta AJAX al server e gestisce la risposta: mostra un messaggio di errore in caso di fallimento o reindirizza alla pagina iniziale in caso di successo.

```
//funzione per il login
function loginProcedure (){

    //recupero valori inseriti nel form
    var email=document.getElementById("email").value;
    var password=document.getElementById("password").value;

    //modifico la password col sale e la cifro
    password=email.substring(0,5)+password;
    password=sha256(password);

    var req=new XMLHttpRequest();
    //console.log("req");
    req.onload=function () {

    // Se lo stato della richiesta è 200 (OK) e la risposta non è "ERROR"
    // Reindirizza l'utente alla pagina index.php
    if(req.status==200 && this.responseText!= "ERROR"){
            window.location.href="index.php";
        }
    else{
            var divElement = document.getElementById("messaggio_errore");
            // Aggiunta delle classi al div
            divElement.className = "alert alert-danger";
            divElement.innerHTML="Email o password non validi";
        }

    }

    //Richiesta AJAX al server (la password è cifrata)
    req.open("get","php/login.php/users/"+email+"/"+password+"/",true);
    console.log("connessione");
    req.send();
}
```

Figura 5: Funzione di Login

- **register.js**: Si occupa della registrazione dell'utente. Controlla la validità dei campi come email e password e invia una richiesta AJAX per l'inserimento nel database.
- **home.js**: Definisce una funzione utilizzata in tutti gli altri script per due scopi:
  1. L'inserimento del nome utente nella topbar.
  2. L'attivazione della procedura di logut nel momento in cui è invocata
- **bmi.js**: Questo script è responsabile della formattazione dinamica della pagina in base ai dati inseriti dall'utente. In particolare, calcola e visualizza il BMI inviando una richiesta AJAX per l'inserimento, o stampa un messaggio d'errore quando i dati inseriti dall'utente sono invalidi.
- **charts.js**: Utilizza la libreria `Chart.js` per creare grafici in javascript. Per ognuno dei 3 grafici è eseguita una richiesta AJAX al server per ottenere i dati

dal DB, che per necessità strutturali sono in 2 array, labels e data. Sulla base dei dati è costruito il grafico.

```
//assegno a degli array i valori di data e labels
var datadb=dati.data;
var labels=dati.labels;

//creo I dati per il grafico on labels e dati corrispondenti dati
const dataChart={
    labels: labels,
    datasets: [{
        label: "BMI",
        data: datadb,
        fill: false,
        borderColor: 'rgb(78, 115, 223)',
    }]
}

//configuro il grafico con il tipo line
const config = {
    type: 'line',
    data: dataChart,
    options: {
      plugins: {
          legend: {
              display: false, // Nascondi la legenda
          }
      }
    }
  };

const chart = new Chart(CanvasBMI, config); //crazione vera e propria del grafico
```

Figura 6: Esempio di creazione di un grafico

- **scheda.js** Lo script utilizzato nel file `schede.php` va a gestire la visualizzazione delle schede passate ed attuali dopo averle richieste al server.
- **pagina_scheda.js**: Uno dei più complessi tra gli script, gestisce diverse funzioni:
  1. Visualizza la scheda e i suoi esercizi con informazioni dettagliate.
  2. Permette l'aggiunta o la rimozione di esercizi quando è attivato l'eventListener sul relativo bottone. Va a creare delle checkbox per gli esercizi, e per quelli selezionati va ad aggiungere dei campi di input per i valori di serie,ripetizioni e recupero.
  3. Consente di terminare la scheda, inserendo automaticamente la data di fine scheda corrente.
- **add_scheda.js**: Lo script serve a creare una nuova scheda tramite pagina in cui sono visualizzabili delle card con al proprio interno delle checkbox con degli esercizi divisi per gruppo muscolare. Nel momento in cui una delle checkbox è selezionata vengono mostrati i campi di input relativi al numero di serie,ripetizioni e tempo di recupero.
- **esercizi.js**: lo script va a gestire la pagina `esercizi.php` andando a mostarare delle card con il nome dell'esercizio, una gif sull'esecuzione ed un bottone che può essere di due tipi: rimozione dalla scheda nel caso in cui sia presente

in quella attiva, e di inserimento nella scheda nel caso in cui non vi ci sia. Nel secondo caso sarà invocato un modal che richiederà le info necessarie alla registrazione nella scheda del nuovo esercizio.

- **allenamenti.js** Lo script è un gestore completo per l'applicazione di allenamento. Gestisce la transizione della pagina da una fase di selezione degli esercizi iniziale all'effettivo allenamento. Nasconde e mostra diversi elementi HTML in base allo stato dell'allenamento. Interagisce con l'utente per passare attraverso gli esercizi, mostrando una schermata per ogni esercizio con le relative informazioni, tra cui un timer descritto da una barra per tenere traccia del tempo di recupero tra le serie, per poi salvare i pesi utilizzati in un oggetto JSON. Fornisce anche la funzionalità di terminare l'allenamento e salvare i dati nel database tramite una richiesta AJAX

- **mostra _allenamento.js**: Utilizza XMLHttpRequest per fare una richiesta al server per ottenere dati in base all'id dell'allenamento fornito. Se la risposta indica un errore ("ERROR"), reindirizza l'utente a 'allenamenti.php'. Altrimenti, analizza la risposta JSON ottenuta dal server e la mostra nella pagina. Per ogni esercizio ricevuto, mostra la card corrispondente al gruppo di esercizi e crea un paragrafo per visualizzare le informazioni sull'esercizio (nome, serie, ripetizioni, recupero). Se è specificato un peso per l'esercizio, lo aggiunge alle informazioni visualizzate.

- **storico_allenamenti.js**: Viene effettuata una richiesta al server per ottenere i dati relativi allo storico degli allenamenti dell'utente.Una volta ottenuti i dati sono mostrati gli allenamenti. Per ciascun allenamento nell'insieme di dati ottenuti, viene creata una card con la data dell'allenamento formattata nel formato giorno-mese-anno. Queste card vengono visualizzate nello storico degli allenamenti sulla pagina, ciascuna con un link che porta a una pagina per vedere i dettagli specifici di quell'allenamento.

- **go_back.js** La top bar presenta un bottone per tornare alla pagina precedente dell'utente. Per garantire una navigazione coerente, se l'utente si trova su una delle pagine predefinite (come index.php, login.html o register.html), premere il bottone di ritorno lo reindirizzerà sempre a index.php. In caso contrario, il bottone tornerà alla pagina visitata precedentemente dall'utente.

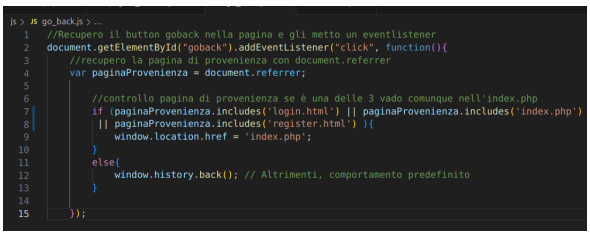

Figura 7: go_back.js

- **profile.js**: Questo script si occupa dell'aggiornamento del profilo utente. Viene effettuata una richiesta al server per ottenere le informazioni dell'utente corrente. Una volta ricevute le informazioni, vengono compilate nei campi del modulo del profilo. Quando il pulsante "modifica" è cliccato, il sistema verifica se almeno uno dei campi è stato modificato. Se una modifica è stata apportata, raccoglie i nuovi valori dei campi del modulo e li organizza in un oggetto. Successivamente, invia una richiesta al server per aggiornare le informazioni dell'utente con questi nuovi valori. La gestione della risposta del server è la seguente: se l'email è già associata ad un altro account, viene mostrato un avviso. Al contrario, se l'operazione va a buon fine, viene visualizzato un messaggio di successo e l'utente viene reindirizzato alla pagina iniziale. Infine lo script gestisce anche la possibilità di poter eliminare il proprio account.

5 Database

5.1 Definizione Schema E-R

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

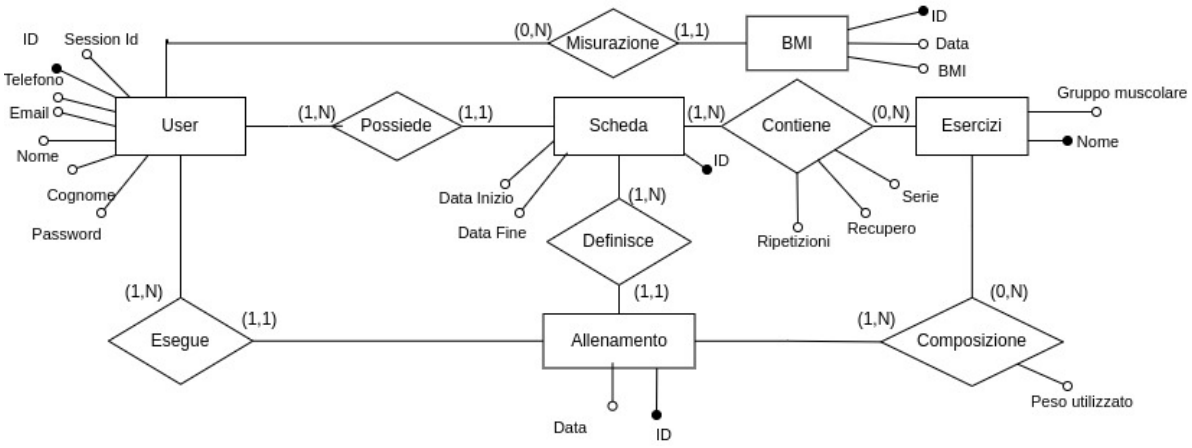

Figura 8: Schema ER iniziale

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

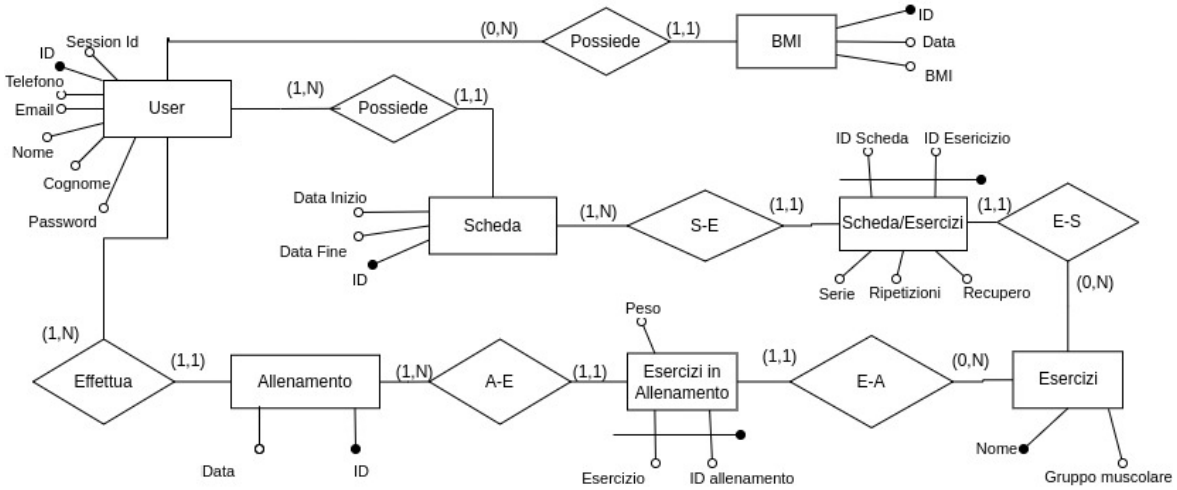

Figura 9: Schema ER finale

5.2 Struttura finale del DB

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

Tabella	Attributi
users	id,nome,cognome,email,pswrd,telefono,session_id
schede	id ,data_inizio,data_fine,user*
esercizi	nome,gruppo
allenamenti	id,data,scheda* ,user*
bmi	id,bmi,data,user*
a_e	allenamento*,esercizio*,peso
e_s	scheda*,esercizio*,serie,ripetizioni,recupero

Tabella 1: Le foreign key sono indicate con *, mentre le primary key sono sottolineate

5.3 Accorgimenti

Nella definizione del DB sono stati attenzionati anche alcuni aspetti che potrebbero rappresentare un rischio in ambito di sicurezza tra cui:

- La password non è salvata in chiaro, ma bensì in forma criptata usando l'algoritmo di hashing SHA256. Inoltre prima della criptazione va a subire una modifica tramite l'aggiunta di un sale, ossia una stringa aggiuntiva che viene appesa a quella originale così da avere nel db una forma potenzialmente sicura.

```
//modifico la password col sale e la cifro
password=email.substring(0,5)+password;
password=sha256(password);
```

Figura 10: Sezione di codice

6 Servizio Web: Restful

Nel contesto del progetto, le API RESTful sono state ampiamente impiegate per facilitare e gestire tutte le connessioni e le interazioni con il database al fine di creare il servizio web. Le API RESTful rappresentano un approccio standard per la creazione di servizi web, offrendo un'architettura flessibile e scalabile per lo sviluppo di applicazioni web e mobile.

Le principali funzionalità del sistema, come la gestione degli utenti, la creazione di schede di allenamento, la registrazione degli allenamenti, e altre operazioni cruciali, sono state implementate come risorse accessibili tramite API RESTful. Ciò ha consentito un accesso agevole e strutturato ai dati, garantendo una chiara separazione tra il front-end e il back-end del sistema.
Nell'architettura dei servizi RESTful, uno dei principi fondamentali è l'uso delle URI (Uniform Resource Identifier) per identificare le risorse. Ogni richiesta inviata al backend tramite AJAX utilizza un URI che non soltanto fa riferimento al file responsabile della gestione della richiesta, ma contiene anche informazioni aggiuntive. Ad esempio, un URI potrebbe avere il seguente formato:

```
file.php/nometabella/dato1/dato2
```

Di seguito un esempio di URI utilizzato:

```
req.open('POST', "php/logicaAllenamento.php/allenamenti/"+formattedDate, true);
```

Figura 11: Esempio di URI utilizzato, vi sono informazioni come la tabella (allenamenti) ed info aggiuntive come la variabile formattedDate

Un'altro dei vantaggidei servizi Restful è l'utilizzo esplicito dei metodi HTTP. Nel progetto infatti ad ogni richiesta effettuata al database è associato il metodo HTTP adegueato. Di seguito vi sono alcuni esempi.

6.1 Metodo GET

```
// Funzione per recuperare informazioni sugli esercizi
tabnine: test | explain | document | ask
function get_esercizi() {
    var req = new XMLHttpRequest();

    req.onload = function() {
        var data = JSON.parse(req.responseText);
        //FORMATO DELLA RISPOSTA: array con tutti fli esercizi nel formato
        //{0: "plank", 1: "addome", nome: "plank", gruppo: "addome"}
        //questo formato consente di accedere sia in con indice numerico che indice associativo
        //console.log(data);
        put_esercizi(data);
    }

    req.open('GET','php/esercizi.php/esercizi/attuale',true);
    req.send();
}
```

Figura 12: Metodo GET

```
//recupero gli esercizi dalla tabella esercizi
// Se il metodo è GET e la tabella è "esercizi"
if($method=="GET" && $table=="esercizi"){
    // Query per selezionare tutti i dati dalla tabella 'esercizi' ordinati per 'gruppo'
    $query = "SELECT * FROM esercizi ORDER BY gruppo";
    $res = mysqli_query($conn, $query);
    $rows = [];

    // Carica l'array 'rows' con i risultati della query
    if($res){
        while($row = mysqli_fetch_array($res)){
            $rows[] = $row;
        }
    }

    $rows = json_encode($rows);
    echo $rows;
}
```

Figura 13: Lato server associato al metodo GET, con una query di SELECT

6.2 Metodo POST

```
//funzione per inserire l'esrcizio nel DB
function insert_esercizio(nome, serie, ripetizioni, recupero) {

    var req = new XMLHttpRequest();

    req.onload = function(){
        console.log(this.responseText);
        if(this.responseText == 'ok'){
            location.reload();
        }
    };

    //richiesta AJAX post con i dati nell'URI
    req.open("POST", "php/logicaSchede.php/e_s/"+nome+"/"+serie+"/"+ripetizioni+"/"+recupero, true);
    req.send();
}
```

Figura 14: Metodo POST

```
//Inserimento dell'esercizio nella tabella e_s
elseif($method == "POST" && $table == "e_s"){
    $id_scheda = getSchedaFromUserID($conn, $id_user);

    //recupero valori dall'URI
    $nome = array_shift($request);
    $serie = array_shift($request);
    $ripetizioni = array_shift($request);
    $recupero = array_shift($request);

    //query
    $query="INSERT INTO `e_s`(`esercizio`, `scheda`, `serie`, `ripetizioni`, `recupero`) VALUES (?,?,?,?,?)";
    $stmt = mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt, "siiii", $nome, $id_scheda, $serie, $ripetizioni, $recupero);
    $res = mysqli_stmt_execute($stmt);

    //risposta
    if($res){
        echo 'ok';
    }
    else{
        echo 'error';
    }
}
```

Figura 15: Lato server associato al metodo POST, che esegue un INSERT

6.3 Metodo PUT

```javascript
function termina_scheda(){

    var req = new XMLHttpRequest();

    const today = new Date();
    const year = today.getFullYear();
    const month = String(today.getMonth() + 1).padStart(2, '0'); // Aggiunge lo zero iniziale se il mese è inferiore a 10
    const day = String(today.getDate()).padStart(2, '0'); // Aggiunge lo zero iniziale se il giorno è inferiore a 10
    const formattedDate = `${year}-${month}-${day}`;
    console.log(formattedDate);

    req.onload = function(){

        console.log("risposta server: "+this.responseText);

        if (this.responseText == 'ok'){
            window.location.href = "schede.php";
        }

    }

    req.open('PUT', "php/logicaSchede.php/schede/"+id+"/"+formattedDate, true);
    req.send();
}
```

Figura 16: Metodo PUT

```php
//modifica in cui settiamo la data di fine
elseif ($method == 'PUT' && $table == 'schede' && isset($request[0]) && isset($request[1]) && !isset($request[2])) {

    //recuepero variabili dall'URI
    $id_scheda=array_shift($request);
    $data_fine=array_shift($request);

    //echo "ID Scheda: $id_scheda, Data Fine: $data_fine, UserID: $userId";

    //  query
    $query="UPDATE `schede` SET `data_fine`=? WHERE id=? AND user=?";
    $stmt=mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt, "ssi",$data_fine, $id_scheda, $id_user);
    mysqli_stmt_execute($stmt);
    if(mysqli_affected_rows($conn) > 0){
        echo "ok";
    }
    else{
        echo "ERROR";
    }

}
```

Figura 17: Lato server associato al metodo PUT, esegue un UPDATE

6.4 Metodo DELETE

```
//funzione per eliminare gli esercizi della scheda
tabnine: test | explain | document | ask
function elimina_esercizio(){
    const id_esercizio = document.querySelector(".hidden_esercizio_elimina").value;

    var req = new XMLHttpRequest();

    req.onload = function(){
        if(this.responseText=='ok'){
            location.reload();
        }
    };

    req.open("DELETE", "php/logicaSchede.php/e_s/"+id_esercizio, true);
    req.send();

}
```

Figura 18: Metodo DELETE

```
//Eliminazione dell'esercizio dalla scheda
elseif ($method == "DELETE" && $table == "e_s" && isset($request[0])){
    $id_scheda = getSchedaFromUserID($conn, $id_user);
    $esercizio = array_shift($request);

    $query = "DELETE FROM e_s WHERE scheda=? AND esercizio=?";
    $stmt = mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt, "is", $id_scheda, $esercizio);
    mysqli_stmt_execute($stmt);

    if(mysqli_affected_rows($conn) > 0){
        echo "ok";
    }
    else{
        echo "ERROR";
    }

}
```

Figura 19: Lato server associato al metodo DELETE, esegue un DELETE