Documentazione Query PHP

Di seguito tutte le *query* spiegate nel dettaglio che il *microservizio* in python dovrà effettuare e restituirne il contenuto del record al sito PHP per la gestione degli utenti, delle schede e degli esercizi. Per la gerarchia degli *url* da chiamare, preferirei sentirmi con Marco in modo da creare una sorta di omogeneità, visto anche l'utilizzo di un servizio simile da parte dell'applicazione Android.

Bisognerà concordare insieme la risposta che il *microservizio* manderà al sito PHP, considerando anche l'aspetto dei coockie (argomento già trattato con Marco, bisogna solo capire come gestirli lato sito web).

Glossario:

Utente: Atleta che si allena in palestra, che quindi consulta la propria scheda di allenamento per prendere visione degli esercizi da svolgere. Egli può solo prendere visione della suddetta scheda senza apportarvi modifiche. Può effettuare anche operazioni di lettura dei propri dati personali e modifica degli stessi. L'eliminazione è lasciata all'operatore/gestore nel momento in cui l'utente non prosegue il proprio abbonamento in palestra. Egli appartiene alla tabella del DB **'user'**.

Operatore/Gestore: Gestore della palestra o anche personal trainer, utenti quindi con privilegi maggiori, che possono svolgere operazioni di Lettura, Scrittura, Aggiornamento ed Eliminazione di tutti i record (utenti, esercizi, schede, invio messaggi etc.). Egli appartiene alla tabella del DB **'account**.

Autocomplete: Aiuto che viene dato all'Operatore/Gestore nel momento in cui compila il form per la ricerca di uno specifico utente oppure di uno specifico esercizio.

Negli script viene effettuata anche l'impaginazione dei record che vengono restituiti dal DB. Questa operazione si basa sul conteggio del numero di record presenti. Il microservizio quindi in questo caso si dovrebbe limitare a riportare il quantitativo totale nella response.

Di seguito le *query*:

- **login_website/login.php**

```
7   // Escape email to protect against SQL injections
8   $email = $mysqli->escape_string($_POST['email']);
9   $stmt = $mysqli->prepare( query: "SELECT first_name, last_name, password FROM account WHERE email=?");
10  $stmt->bind_param( types: 's', &var1: $email);
11  $result = $stmt->execute();
12  $stmt->store_result();
```

Serve per l'autenticazione dell'utente per poter usufruire delle funzioni del sito web.

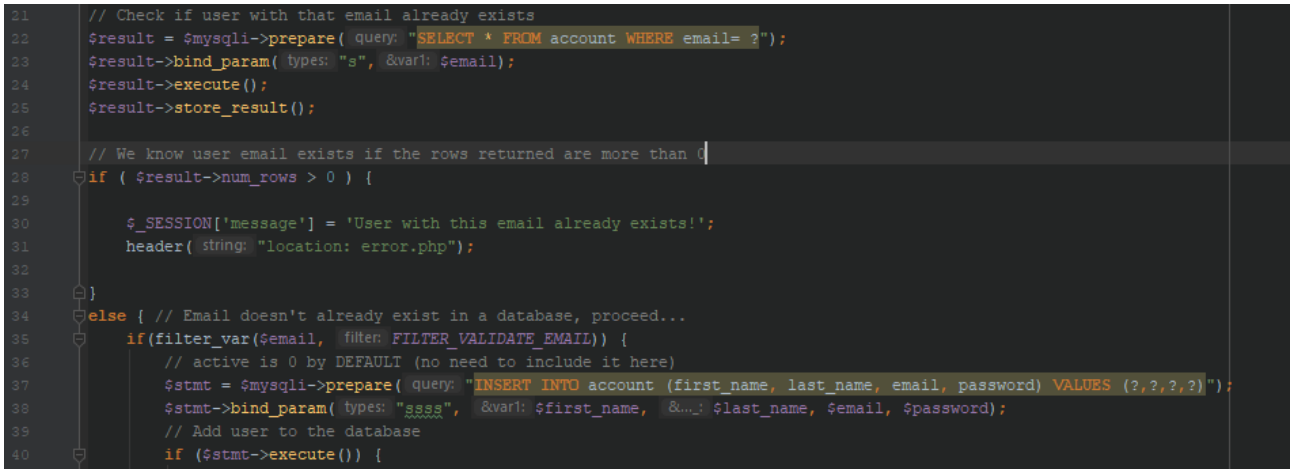
Input:

- email

Output:

- *first_name*
- *last_name*
- *password*

- **login_website/register.php**



Questo serve per la registrazione di un nuovo utente. Prima di tutto faccio fare un controllo che effettivamente l'email che l'utente sta inserendo non sia presente all'interno del DB, e dopo, solo una volta che viene validata l'email (FILTER_VALIDATE_EMAIL), avviene la scrittura sul DB del nuovo utente.

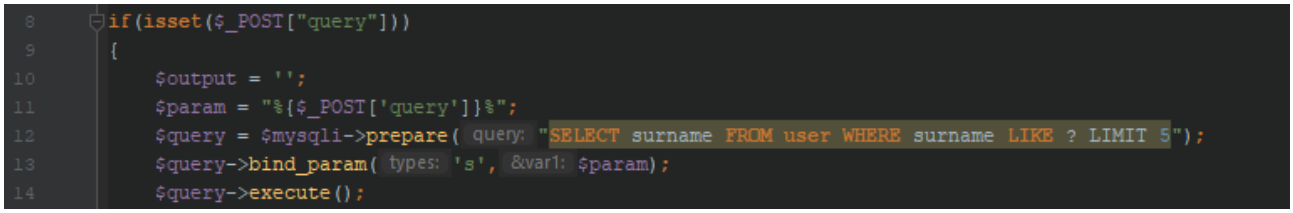
Input:

- *email*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/autocomplete.php**



Questo serve per *l'autocomplete* del form di ricerca dell'utente da parte dell'operatore.

Input:

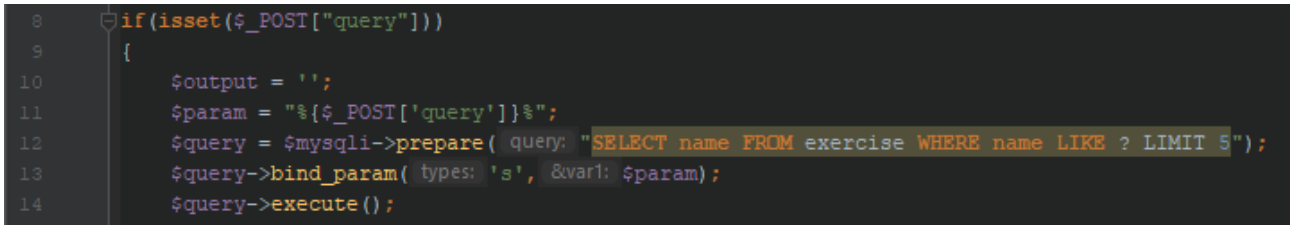
- *POST[query]* (ricerca che viene effettuata dall'operatore)

Output:

- *surname*

Stavo pensando che magari con gli *autocomplete* si potrebbero avere dei problemi andando ad utilizzare questo nuovo approccio del microservizio python. Andrebbe controllato. Appena abbiamo il codice python funzionante lo testiamo.

- **manage_schedules/autocomplete_exercise.php**



Stessa cosa dell'*autocomplete* soprariportato, solo che questo serve per gli esercizi e non per gli utenti. Stesse considerazioni fatte sopra circa l'utilizzo del microservizio python.

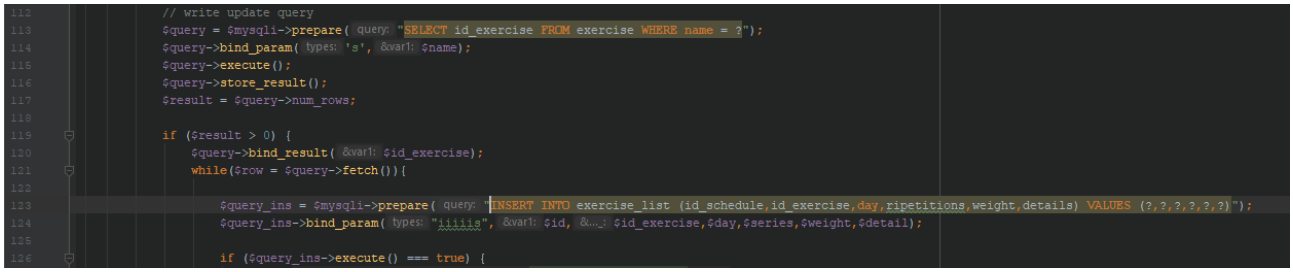
Input:

- *POST[query]* (ricerca che viene effettuata dall'operatore)

Output:

- *name*

- **manage_schedules/create_exercise.php**



Questo serve per la creazione di un nuovo esercizio per la scheda dell'utente.

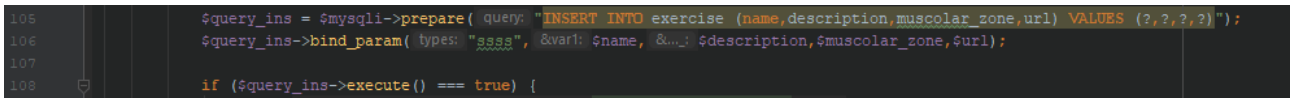
Input:

- *name*
- *id* (identificativo della scheda)
- *id_exercise* (identificativo dell'esercizio da inserire all'interno della scheda)
- *day*
- *ripetitions*
- *weight*
- *details*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/create_exercise_list.php**



Questo serve per la creazione di un nuovo esercizio all'interno del DB. Un esercizio per poter essere inserito in una scheda di un utente, deve essere presente nel DB.

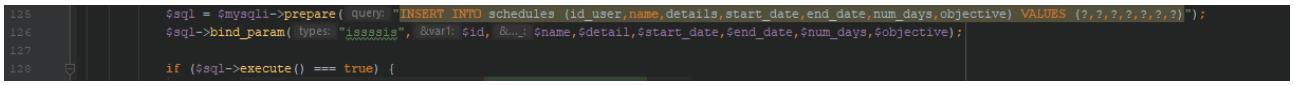
Input:

- *name*
- *description*
- *muscolar_zone*
- *zone_url*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/create_schedule.php**



Questo serve per la creazione di una scheda d'allenamento per l'utente.

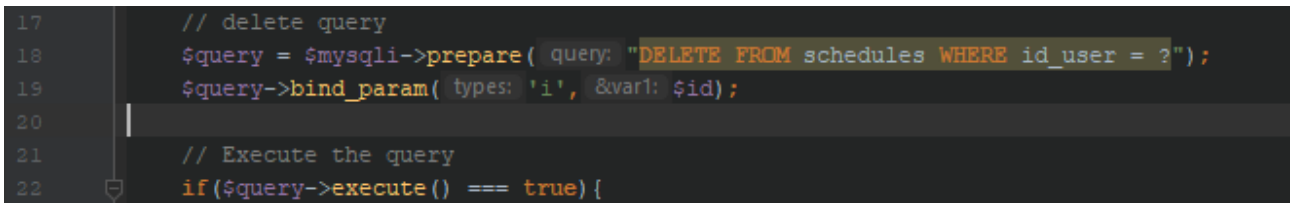
Input:

- *id_user* (identificativo dell'utente)
- *name*
- *details*
- *start_date*
- *end_date*
- *num_days*
- *objective*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/delete_schedule.php**



Questo serve per l'eliminazione di tutte le schede d'allenamento dell'utente.

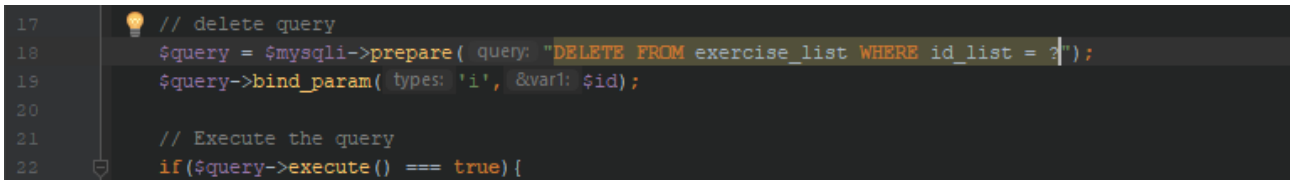
Input:

- *id_user* (identificativo dell'utente)

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/delete_single_exercise.php**



Questo serve per l'eliminazione di un esercizio dalla scheda d'allenamento dell'utente.

Input:

- *id_list* (identificativo della lista d'allenamento dell'utente)

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/delete_single_exercise_list.php**

Questo serve per l'eliminazione di un esercizio dalla base dati. L'esercizio sarà più consultabile e non potrà essere inserito in nessuna nuova scheda.

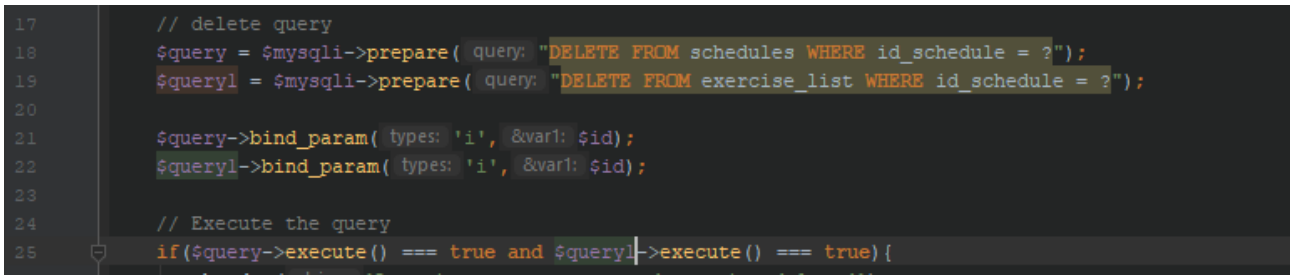
Input:

- *id_exercise* (identificativo dell'esercizio che si vuole eliminare)

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/delete_single_schedule.php**



Questo elimina la singola scheda d'allenamento dell'utente.

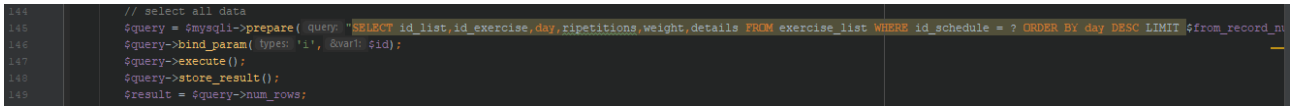
Input:

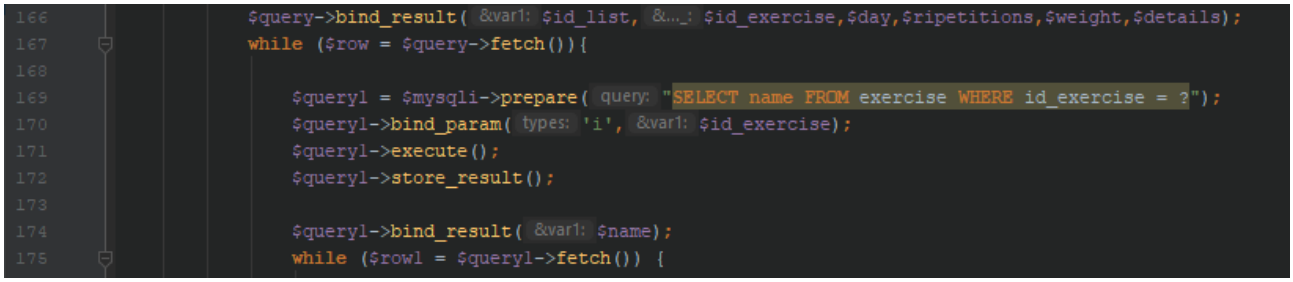
- *id* (identificativo della scheda d'allenamento che si vuole eliminare)

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_schedules/manage_exercises.php**





Questo serve per la gestione degli esercizi appartenenti alla scheda d'allenamento di un utente.

Input:

- *id* (identificativo della scheda dell'utente)
- *from_record_num*
- *record_per_page*

Output:

- *id_list*
- *id_exercise*
- *day*
- *name*
- *ripetitions*
- *weight*
- *details*

Per *from_record_num* e *record_per_page* vale la stessa considerazione fatta sopra.

```
201         // PAGINATION
202         // count total number of rows
203         $statement = mysqli_prepare($mysqli, query: 'SELECT * FROM exercise_list WHERE id_schedule = ? ');
204         mysqli_stmt_bind_param($statement, types: "i", &var1: $id);
205         mysqli_stmt_execute($statement);
206         mysqli_stmt_store_result($statement);
207         $total_rows = mysqli_stmt_num_rows($statement);
208         mysqli_stmt_close($statement);
209
210         // paginate records
211         $page_url="manage_exercises.php?";
212         include_once "paging.php";
```

Questo, analogamente a quanto riportato prima, serve per l'impaginazione.

Input:

- *id* (identificativo della scheda dell'utente)

Output:

- *ALL*

- **manage_schedules/manage_schedules.php**

```
140     $query_user = $mysqli->prepare( query: "SELECT name,surname FROM user WHERE id_user = ?");
141     $query_user->bind_param( types: "i", &var1: $id);
142     $query_user->execute();
143     $result = $query_user->get_result();
144     if ($query_user->execute() === true) {
```

```
156 // select all data
157
158 $query = $mysqli->prepare( query: "SELECT id_schedule, name, details, start_date, end_date, num_days, objective FROM schedules WHERE id_user = ? ORDER BY end_date DESC LIMIT $from
159 $query->bind_param( types: 'i', &var1: $id);
160 $query->execute();
161 $query->store_result();
162 $result = $query->num_rows;
163
164 //check if more than 0 record found
165 if ($result > 0) {
```

Questo serve per la gestione delle schede d'allenamento dell'utente.

Input:

- *id* (identificativo dell'utente)
- *from_record_num*
- *record_per_page*

Output:

- *id_schedule*
- *name*
- *details*
- *start_date*
- *end_date*

- *num_days*
- *objective*

Per *from_record_num* e *record_per_page* vale la stessa considerazione fatta sopra.

```
// PAGINATION
// count total number of rows
$statement = mysqli_prepare($mysqli, query: 'SELECT * FROM schedules WHERE id_user = ? ');
mysqli_stmt_bind_param($statement, types: "i", &var1: $id);
mysqli_stmt_execute($statement);
mysqli_stmt_store_result($statement);
$total_rows = mysqli_stmt_num_rows($statement);
mysqli_stmt_close($statement);
```

Questo, analogamente a quanto riportato prima, serve per l'impaginazione.

Input:

- *id* (identificativo dell'utente)

Output:

- *ALL*

- **manage_schedules/manage_users.php**

```
// select all data

$query = "SELECT id_user, name, surname, email FROM user ORDER BY id_user DESC LIMIT $from_record_num,$records_per_page";
$result = $mysqli->query($query);

//check if more than 0 record found
if (mysqli_num_rows($result)){
```

Questo serve per la gestione degli utenti.

Input:

- *from_record_num*
- *record_per_page*

Output:

- *id_user*
- *name*
- *surname*
- *email*

Per *from_record_num* e *record_per_page* vale la stessa considerazione fatta sopra.

```
// PAGINATION
// count total number of rows
$statement = mysqli_prepare($mysqli, query: 'SELECT * FROM user');
mysqli_stmt_execute($statement);
mysqli_stmt_store_result($statement);
$total_rows = mysqli_stmt_num_rows($statement);
mysqli_stmt_close($statement);
```

Questo, analogamente a quanto riportato prima, serve per l'impaginazione. Semplice SELECT.

- **manage_schedules/read_exercises.php**

```
137         // select all data
138         $query = "SELECT * FROM exercise LIMIT $from_record_num,$records_per_page";
139         $result = $mysqli->query($query);
140
141         //check if more than 0 record found
142         if (mysqli_num_rows($result)){
```

Questo serve per la lettura di tutti gli esercizi presenti nel DB.

Input:

- *from_record_num*
- *record_per_page*

Output:

- *name*
- *description*
- *muscolar_zone*

Per *from_record_num* e *record_per_page* vale la stessa considerazione fatta sopra.

```
168         // PAGINATION
169         // count total number of rows
170         $statement = mysqli_prepare($mysqli, query: 'SELECT * FROM exercise');
171         mysqli_stmt_execute($statement);
172         mysqli_stmt_store_result($statement);
173         $total_rows = mysqli_stmt_num_rows($statement);
174         mysqli_stmt_close($statement);
175
176         // paginate records
177         $page_url="exercises_list.php?";
178         include_once "paging.php";
```

Quest'operazione serve per l'impaginazione. Semplice SELECT.

- **manage_schedules/read_one_exercise.php**

```
73          // prepare select query
74          $query1 = $mysqli->prepare( query: "SELECT name, description, muscolar_zone FROM exercise WHERE id_exercise = ? LIMIT 0,1");
75          $query1->bind_param( types: 'i', &var1: $id);
76          $query1->execute();
77          $query1->store_result();
78          $query1->bind_result( &var1: $name, &...: $description, $muscolar_zone);
79          while ($row1 = $query1->fetch()) {
```

Questo serve per la lettura dei dettagli del singolo esercizio.

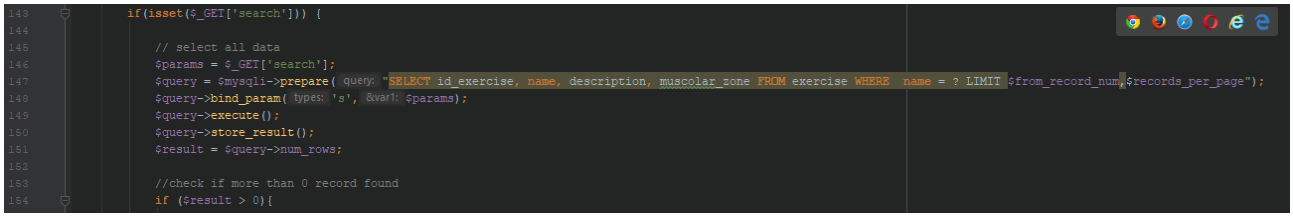
Input:

- *id* (identificativo dell'esercizio)

Output:

- *name*
- *description*
- *muscolar_zone*

- **manage_schedules/search_exercise_list.php**



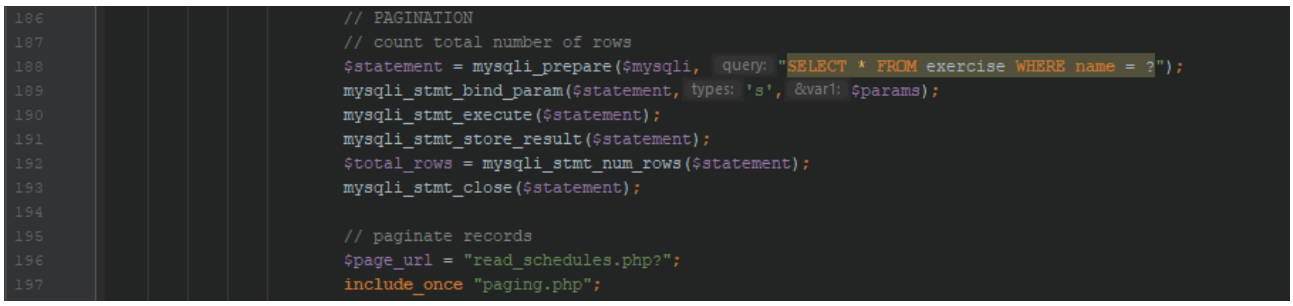
Questo serve per la ricerca del singolo esercizio da parte dell'operatore.

Input:

- *GET['search']* (è il nome dell'esercizio che viene ricercato)

Output:

- *Id_exercise*
- *name*
- *description*
- *muscolar_zone*



Quest'operazione serve per l'impaginazione.

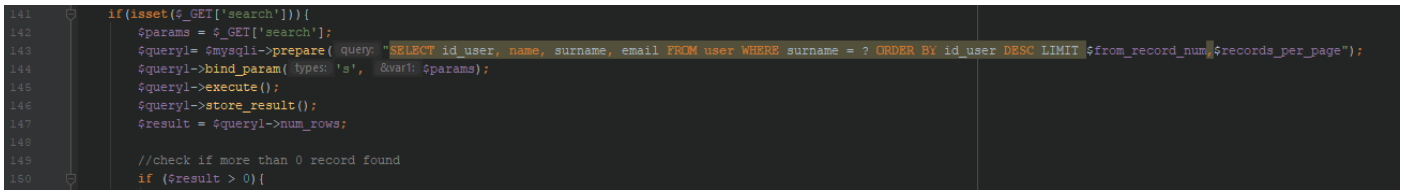
Input:

- *GET['search']* (è il nome dell'esercizio che viene ricercato)

Output:

- *ALL*

- **manage_schedules/search_user.php**



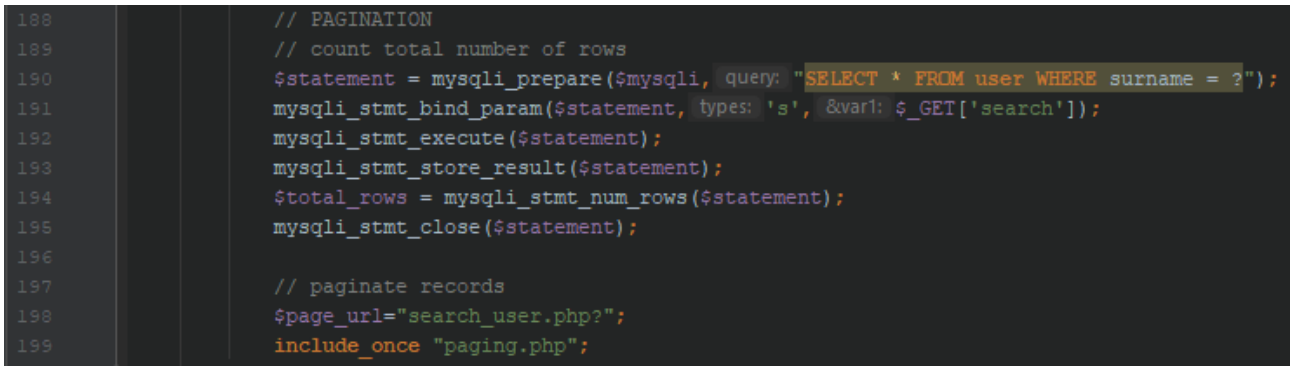
Questo serve per la ricerca del singolo utente da parte dell'operatore.

Input:

- *GET['search']* (è il nome dell'utente che viene ricercato)

Output:

- *Id_user*
- *name*
- *surname*
- *email*



Quest'operazione serve per l'impaginazione.

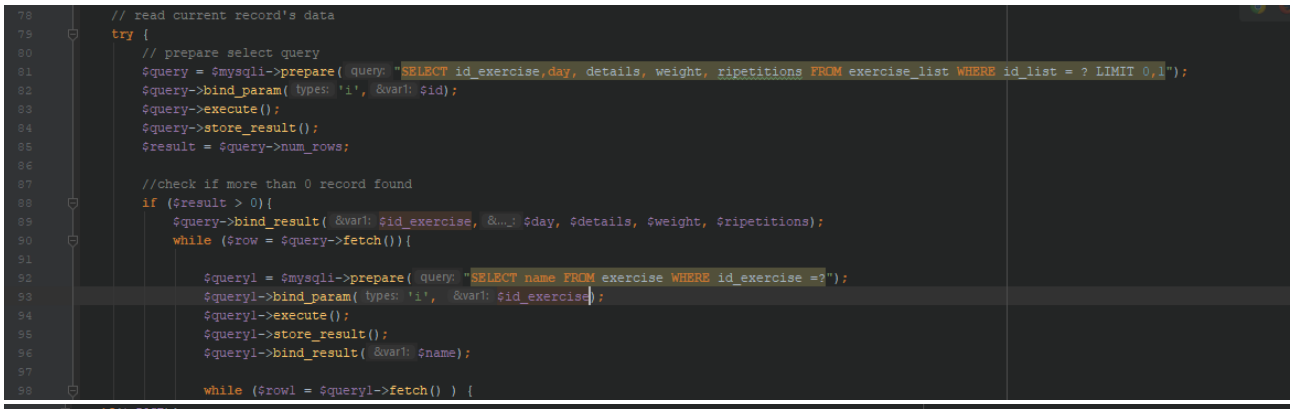
Input:

- *GET['search']* (è il nome dell'utente che viene ricercato)

Output:

- *ALL*

- **manage_schedules/update_exercise.php**



```php
// read current record's data
try {
    // prepare select query
    $query = $mysqli->prepare( query: "SELECT id_exercise,day, details, weight, ripetitions FROM exercise_list WHERE id_list = ? LIMIT 0,1");
    $query->bind_param( types: 'i', &var1: $id);
    $query->execute();
    $query->store_result();
    $result = $query->num_rows;

    //check if more than 0 record found
    if ($result > 0){
        $query->bind_result( &var1: $id_exercise, &...: $day, $details, $weight, $ripetitions);
        while ($row = $query->fetch()){

            $query1 = $mysqli->prepare( query: "SELECT name FROM exercise WHERE id_exercise =?");
            $query1->bind_param( types: 'i',  &var1: $id_exercise);
            $query1->execute();
            $query1->store_result();
            $query1->bind_result( &var1: $name);

            while ($row1 = $query1->fetch() ) {
```

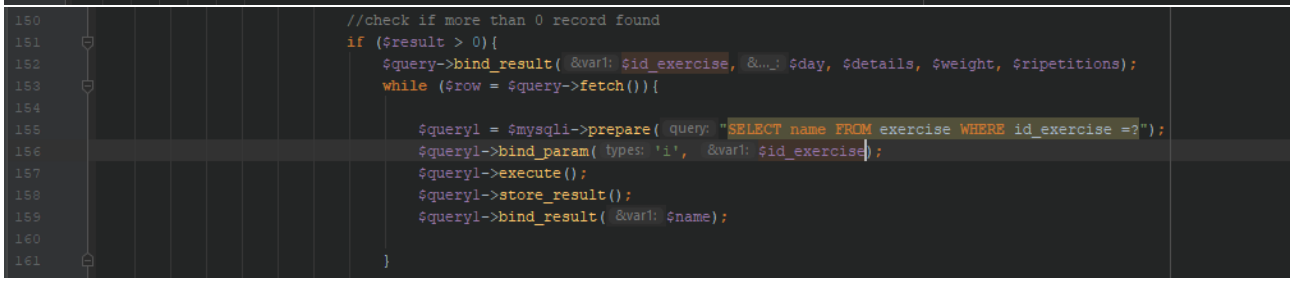
```php
if($_POST){

    try{
        if($_POST['day'] < 1 or $_POST['day'] > 7){
            echo "<div class='alert alert-danger'>Day not valid.</div>";
            throw new Exception();
        }

        // write update query
        $query = $mysqli->prepare( query: "SELECT id_exercise FROM exercise WHERE name = ?");
        $query->bind_param( types: 's',  &var1: $_POST['name']);
        $query->execute();
        $query->store_result();
        $result = $query->num_rows;
        if ($result > 0){
            $query->bind_result( &var1: $id_exercise);
            while ($row = $query->fetch()) {

                $query_up = $mysqli->prepare( query: "UPDATE exercise_list SET id_exercise= ?, day=?, ripetitions=?, weight=?, details=? WHERE id_list = ?");
                $query_up->bind_param( types: 'iiiiisi', &var1: $id_exercise, &...: $_POST['day'], $_POST['ripetitions'],$_POST['weight'],$_POST['detail'],$id);

                // Execute the query
                if ($query_up->execute() === true ) {
                    // read current record's data
                    try {
                        // prepare select query
                        $query = $mysqli->prepare( query: "SELECT id_exercise,day, details, weight, ripetitions FROM exercise_list WHERE id_list = ? LIMIT 0,1");
                        $query->bind_param( types: 'i', &var1: $id);
                        $query->execute();
                        $query->store_result();
                        $result = $query->num_rows;
```



```php
                //check if more than 0 record found
                if ($result > 0){
                    $query->bind_result( &var1: $id_exercise, &...: $day, $details, $weight, $ripetitions);
                    while ($row = $query->fetch()){

                        $query1 = $mysqli->prepare( query: "SELECT name FROM exercise WHERE id_exercise =?");
                        $query1->bind_param( types: 'i',  &var1: $id_exercise);
                        $query1->execute();
                        $query1->store_result();
                        $query1->bind_result( &var1: $name);

                    }
```

Questo serve per l'aggiornamento dell'esercizio presente all'interno della scheda d'allenamento dell'utente. Di fatto è come se sto andando a cambiare un esercizio di una scheda con un altro esercizio. Non devo obbligatoriamente essere una sostituzione ovviamente. Magari si vuole solo aggiornare qualche campo in particolare (es. *day, details, weight* etc.).

Input:

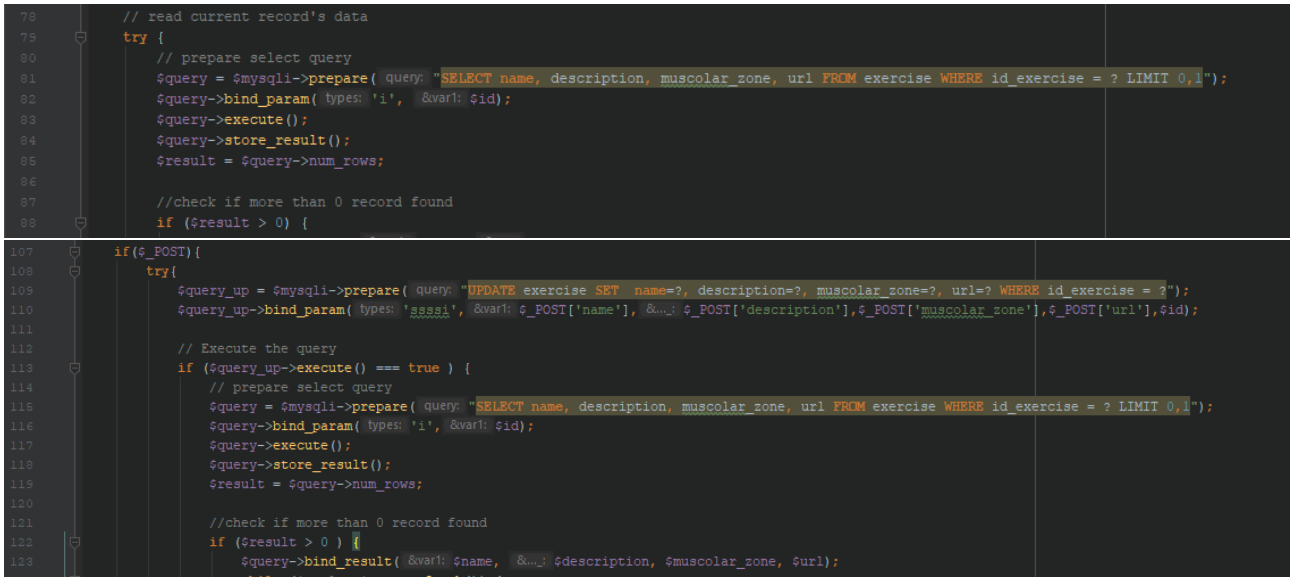
- *id_list* (identificativo della lista dell'utente)

Output:

- *id_exercise*
- *name*
- *day*
- *details*
- *weight*

- *ripetitions*

- **manage_schedules/update_exercise_list.php**



```
// read current record's data
try {
    // prepare select query
    $query = $mysqli->prepare( query: "SELECT name, description, muscolar_zone, url FROM exercise WHERE id_exercise = ? LIMIT 0,1");
    $query->bind_param( types: 'i',  &var1: $id);
    $query->execute();
    $query->store_result();
    $result = $query->num_rows;

    //check if more than 0 record found
    if ($result > 0) {
```

```
if($_POST){
    try{
        $query_up = $mysqli->prepare( query: "UPDATE exercise SET  name=?, description=?, muscolar_zone=?, url=? WHERE id_exercise = ?");
        $query_up->bind_param( types: 'ssssi', &var1: $_POST['name'], &...: $_POST['description'],$_POST['muscolar_zone'],$_POST['url'],$id);

        // Execute the query
        if ($query_up->execute() === true ) {
            // prepare select query
            $query = $mysqli->prepare( query: "SELECT name, description, muscolar_zone, url FROM exercise WHERE id_exercise = ? LIMIT 0,1");
            $query->bind_param( types: 'i', &var1: $id);
            $query->execute();
            $query->store_result();
            $result = $query->num_rows;

            //check if more than 0 record found
            if ($result > 0 ) {
                $query->bind_result( &var1: $name,  &...: $description, $muscolar_zone, $url);
```

Questo serve per l'aggiornamento dell'esercizio presente nel DB.

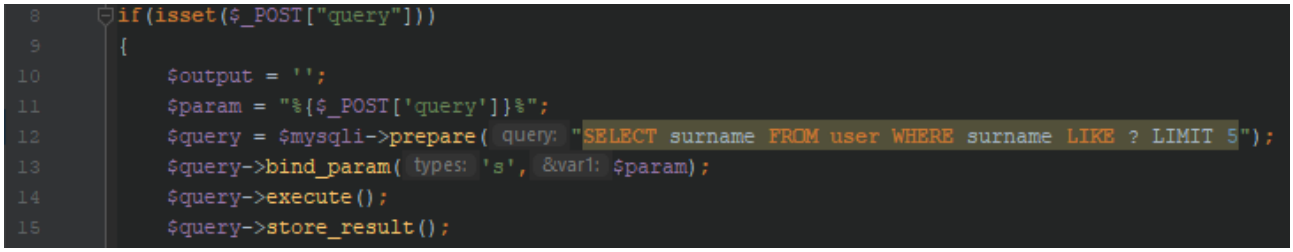
Input:

- *id_exercise* (identificativo dell'esercizio)

Output:

- *name*
- *description*
- *muscolar_zone*
- *url*

- **manage_user/autocomplete.php**



```
if(isset($_POST["query"]))
{
    $output = '';
    $param = "%{$_POST['query']}%";
    $query = $mysqli->prepare( query: "SELECT surname FROM user WHERE surname LIKE ? LIMIT 5");
    $query->bind_param( types: 's', &var1: $param);
    $query->execute();
    $query->store_result();
```

Questo serve per *l'autocomplete* dell'inserimento per la ricerca dell'utente da parte dell'operatore.

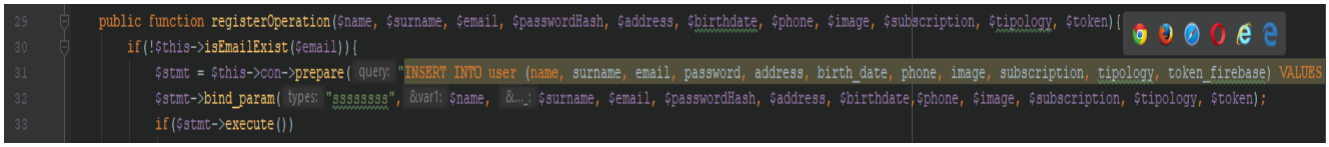
Input:

- *POST['query']* (l'utente che si sta cercando)

Output:

- *surname*

- **manage_user/create_user.php**



Questo script PHP (*create_user.*php) chiama una funzione definita all'interno della classe *DbOperation.php* che opera l'inserimento dell'utente sul DB. Questa soprariportata è la funzione in questione. I controlli dell'inserimento dei form vengono fatti sia lato HTML sia lato PHP.

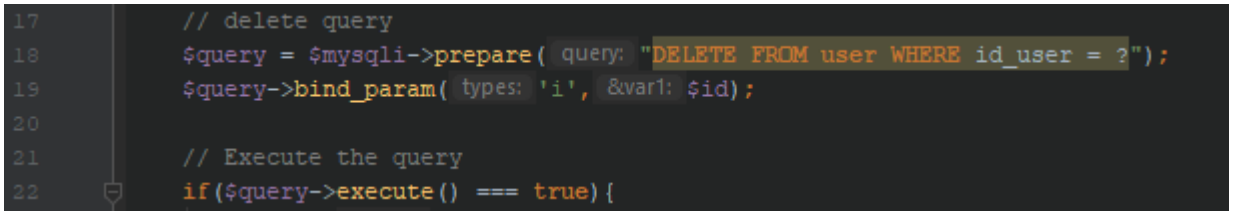
Input:

- *name*
- *surname*
- *email*
- *password*
- *address*
- *birth_date*
- *phone*
- *image*
- *subscription*
- *tipology*
- *token_firebase*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **manage_user/delete_user.php**



Questo elimina un utente dal DB.

Input:

- *id* (identificativo dell'utente in questione)

Output:

- *booleano* (*true o false*)

- **manage_user/read_one_user.php**

```
71      try {
72          // prepare select query
73          $query = $mysqli->prepare( query: "SELECT id_user, name, surname, email, birth_date, address, image, id_subscription FROM user WHERE id_user = ? LIMIT 0,1");
74          $query->bind_param( types: 'i', &var1: $id);
75          $query->execute();
```

Questo serve per la lettura dei dettagli del singolo utente.

Input:

- *id_user* (identificativo dell'utente)

Output:

- *name*
- *surname*
- *email*
- *birth_date*
- *address*
- *image*
- *id_subscription*

- **manage_user/update_user.php**

```
71  // read current record's data
72  try {
73      // prepare select query
74      $query = $mysqli->prepare( query: "SELECT id_user, name, surname, email, birth_date, address, id_subscription FROM user WHERE id_user = ? LIMIT 0,1");
75      $query->bind_param( types: 'i', &var1: $id);
76      $query->execute();
```

```
106             // write update query
107             $query_up = $mysqli->prepare( query: "UPDATE user SET name=?, surname=?, email=?,birth_date=?, address=?,id_subscription=? WHERE id_user = ?");
108             $query_up->bind_param( types: 'ssssssi', &var1: $new_name, &...: $new_surname,$new_email,$new_birth,$new_addr,$new_sub,$id);
109             // Execute the query
110             if($query_up->execute() === true){
111
112                 // read current record's data
113                 try {
114                     // prepare select query
115                     $query = $mysqli->prepare( query: "SELECT id_user, name, surname, email, birth_date, address, id_subscription FROM user WHERE id_user = ? LIMIT 0,1");
116                     $query->bind_param( types: 'i', &var1: $id);
117                     $query->execute();
118                     $result = $query->get_result()->fetch_assoc();
```

Questo serve per l'aggiornamento dei dati dell'utente.

Input:

- *id_user* (identificativo dell'utente)
- *name*
- *surname*
- *email*
- *birth_date*
- *address*
- *id_subscription*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **send_messages/autocomplete.php**

```
$param = "%{$_POST['query']}%";
$query = $mysqli->prepare( query: "SELECT title FROM messages WHERE title LIKE ? LIMIT 5");
$query->bind_param( types: 's', &var1: $param);
$query->execute();
```

Questo serve per *l'autocomplete* dell'inserimento per la ricerca del titolo del messaggio da parte dell'operatore.

Input:

- *POST['query']* (il titolo del messaggio che si sta cercando)

Output:

- *title*

- **send_messages/create_message.php**

```
if($_POST['destination'] == 'token'){
    $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?,?,?,?)");
    $query->bind_param( types: 'ssss', &var1: $title, &...: $body,$datatime, $argument[0]);
    if($query->execute()=== true){
        echo "<div class='alert alert-success'>Message has been saved.</div>";
        $load = true;
    }else{
        echo "<div class='alert alert-danger'>Unable to save the message. Please try again.</div>";
    }
}
elseif($_POST['destination'] == 'multicast'){
    $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?,?,?,?)");
    $query->bind_param( types: 'ssss', &var1: $title, &...: $body,$datatime, $argument[1]);
    if($query->execute()=== true){
        echo "<div class='alert alert-success'>Message has been saved.</div>";
        header( string: 'Location:read_messages.php');
    }else{
        echo "<div class='alert alert-danger'>Unable to save the message. Please try again.</div>";
    }
}
else{
    $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?,?,?,?)");
    $query->bind_param( types: 'ssss', &var1: $title, &...: $body,$datatime, $argument);
    if($query->execute()=== true){
```

Questo serve per l'inserimento del messaggio nel DB. Le tre diverse query sono dovute al fatto che a seconda del destinatario del messaggio, viene svolta un'operazione diversa e quindi una scrittura diversa nel campo *destination*.

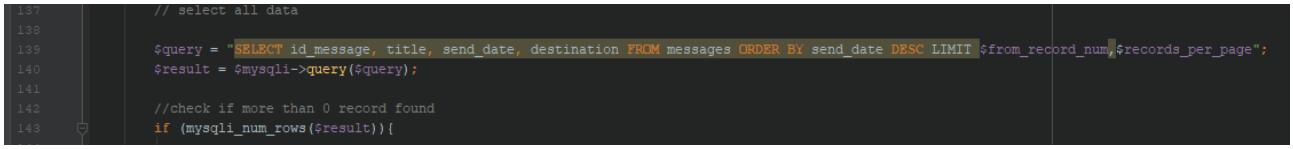
Input:

- *title*
- *body*
- *send_date*
- *destination*

Output:

- *booleano (true o false.* Questo perché i parametri li ho già attraverso la POST, quindi mi serve sapere solo se l'inserimento sul DB è andato a buon fine o meno).

- **send_messages/read_messages.php**



Questo serve per la lettura di tutti i messaggi presenti all'interno del DB.

Input:

- *from_record_num*
- *records_per_page*

Output:

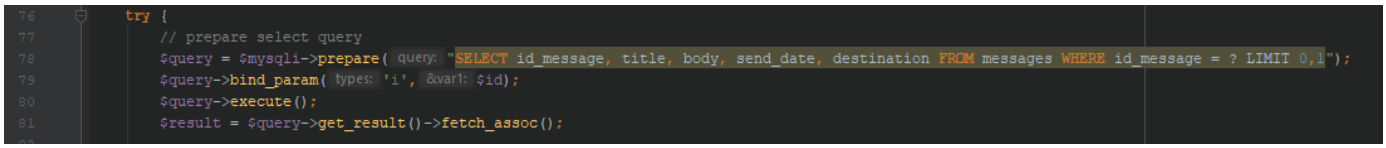
- *id_message*
- *title*
- *send_date*
- *destination*

```
// PAGINATION
// count total number of rows
$statement = mysqli_prepare($mysqli, query: 'SELECT * FROM messages');
mysqli_stmt_execute($statement);
mysqli_stmt_store_result($statement);
$total_rows = mysqli_stmt_num_rows($statement);
mysqli_stmt_close($statement);

// paginate records
$page_url="read_messages.php?";
include_once "paging.php";
```

Questo serve per l'impaginazione. Semplice SELECT.

- **send_messages/read_one_message.php**



Questo serve per la lettura dei dettagli del singolo messaggio.

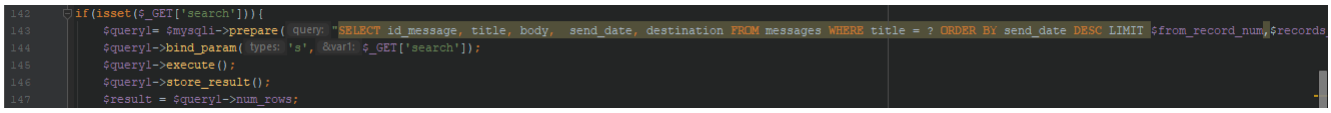
Input:

- *id_message*

Output:

- *title*
- *body*
- *send_date*
- *destination*

- **send_messages/search.php**



Questo serve per la ricerca di un messaggio in particolare da parte dell'operatore.

Input:

- *GET['search']* (è il titolo del messaggio che viene ricercato)

Output:

- *id_message*
- *title*
- *body*
- *send_date*
- *destination*

Questo serve per l'impaginazione.

Input:

- *GET['search']* (è il titolo del messaggio che viene ricercato)

Output:

- *ALL*

Anche qui è necessario soltanto il numero totale di record presenti nel DB.