

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 cookie (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

```
21 // Check if user with that email already exists
22 $result = $mysqli->prepare( query: "SELECT * FROM account WHERE email= ?");
23 $result->bind_param( types: "s", &var1: $email);
24 $result->execute();
25 $result->store_result();
26
27 // We know user email exists if the rows returned are more than 0
28 if ( $result->num_rows > 0 ) {
29
30     $_SESSION['message'] = 'User with this email already exists!';
31     header( string: "location: error.php");
32
33 }
34 else { // Email doesn't already exist in a database, proceed...
35     if(filter_var($email, filter: FILTER_VALIDATE_EMAIL)) {
36         // active is 0 by DEFAULT (no need to include it here)
37         $stmt = $mysqli->prepare( query: "INSERT INTO account (first_name, last_name, email, password) VALUES (?, ?, ?, ?)");
38         $stmt->bind_param( types: "ssss", &var1: $first_name, &var2: $last_name, &var3: $email, &var4: $password);
39         // Add user to the database
40         if ($stmt->execute()) {
```

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

```
8 if(isset($_POST["query"]))
9 {
10     $output = '';
11     $param = "%".$_POST['query']."%";
12     $query = $mysqli->prepare( query: "SELECT surname FROM user WHERE surname LIKE ? LIMIT 5");
13     $query->bind_param( types: 's', &var1: $param);
14     $query->execute();
```

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`

```

8  if(isset($_POST["query"]))
9  {
10     $output = '';
11     $param = "%".$_POST['query']."%";
12     $query = $mysqli->prepare( query: "SELECT name FROM exercise WHERE name LIKE ? LIMIT 5");
13     $query->bind_param( types: 's', &var1: $param);
14     $query->execute();

```

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`

```

112 // write update query
113 $query = $mysqli->prepare( query: "SELECT id_exercise FROM exercise WHERE name = ?");
114 $query->bind_param( types: 's', &var1: $name);
115 $query->execute();
116 $query->store_result();
117 $result = $query->num_rows;
118
119 if ($result > 0) {
120     $query->bind_result( &var1: $id_exercise);
121     while($row = $query->fetch()){
122
123         $query_ins = $mysqli->prepare( query: "INSERT INTO exercise_list (id_schedule,id_exercise,day,ripetitions,weight,details) VALUES (?,?,?,?<?)");
124         $query_ins-&gt;bind_param( types: "iiiiis", &amp;var1: $id, &amp;var2: $id_exercise,$day,$series,$weight,$detail);
125
126         if ($query_ins-&gt;execute() === true) {
</pre

```

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**

```
105 $query_ins = $mysqli->prepare( query: "INSERT INTO exercise (name,description,muscolar_zone,url) VALUES (?, ?, ?, ?)");
106 $query_ins->bind_param( types: "ssss", &$var1: $name, &$var2: $description, &$var3: $muscolar_zone, &$var4: $url);
107
108 if ($query_ins->execute() === true) {
```

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**

```
125 $sql = $mysqli->prepare( query: "INSERT INTO schedules (id_user,name,details,start_date,end_date,num_days,objective) VALUES (?, ?, ?, ?, ?, ?, ?)");
126 $sql->bind_param( types: "issssis", &$var1: $id, &$var2: $name, &$var3: $detail, &$var4: $start_date, &$var5: $end_date, &$var6: $num_days, &$var7: $objective);
127
128 if ($sql->execute() === true) {
```

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**

```
17 // delete query
18 $query = $mysqli->prepare( query: "DELETE FROM schedules WHERE id_user = ?");
19 $query->bind_param( types: 'i', &$var1: $id);
20
21 // Execute the query
22 if ($query->execute() === true) {
```

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**

```
17 // delete query
18 $query = $mysqli->prepare( query: "DELETE FROM exercise_list WHERE id_list = ?");
19 $query->bind_param( types: 'i', &var1: $id);
20
21 // Execute the query
22 if($query->execute() === true){
```

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**

```
17 // delete query
18 $query = $mysqli->prepare( query: "DELETE FROM exercise WHERE id_exercise = ?");
19 $query->bind_param( types: 'i', &var1: $id);
20
21 // Execute the query
22 if($query->execute() === true){
```

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`

```

17 // delete query
18 $query = $mysqli->prepare( query: "DELETE FROM schedules WHERE id_schedule = ?");
19 $query1 = $mysqli->prepare( query: "DELETE FROM exercise_list WHERE id_schedule = ?");
20
21 $query->bind_param( types: 'i', &var1: $id);
22 $query1->bind_param( types: 'i', &var1: $id);
23
24 // Execute the query
25 if($query->execute() === true and $query1->execute() === true){

```

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`

```

144 // select all data
145 $query = $mysqli->prepare( query: "SELECT id_list,id_exercise,day,ripetitions,weight,details FROM exercise_list WHERE id_schedule = ? ORDER BY day DESC LIMIT $from_record_n
146 $query->bind_param( types: 'i', &var1: $id);
147 $query->execute();
148 $query->store_result();
149 $result = $query->num_rows;

```

```

166 $query->bind_result( &var1: $id_list, &var2: $id_exercise,$day,$ripetitions,$weight,$details);
167 while ($row = $query->fetch()){
168
169     $query1 = $mysqli->prepare( query: "SELECT name FROM exercise WHERE id_exercise = ?");
170     $query1->bind_param( types: 'i', &var1: $id_exercise);
171     $query1->execute();
172     $query1->store_result();
173
174     $query1->bind_result( &var1: $name);
175     while ($row1 = $query1->fetch()) {

```

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) {

155 // select all data
156
157 $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
158 $query->bind_param( types: 'i', &var1: $id);
159 $query->execute();
160 $query->store_result();
161 $result = $query->num_rows;
162
163 //check if more than 0 record found
164 if ($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.

```

203 // PAGINATION
204 // count total number of rows
205 $statement = mysqli_prepare($mysqli, query: 'SELECT * FROM schedules WHERE id_user = ? ');
206 mysqli_stmt_bind_param($statement, types: "i", &var1: $id);
207 mysqli_stmt_execute($statement);
208 mysqli_stmt_store_result($statement);
209 $total_rows = mysqli_stmt_num_rows($statement);
210 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*

```

138 // select all data
139
140 $query = "SELECT id_user, name, surname, email FROM user ORDER BY id_user DESC LIMIT $from_record_num,$records_per_page";
141 $result = $mysqli->query($query);
142
143 //check if more than 0 record found
144 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.

```

182 // PAGINATION
183 // count total number of rows
184 $statement = mysqli_prepare($mysqli, query: 'SELECT * FROM user');
185 mysqli_stmt_execute($statement);
186 mysqli_stmt_store_result($statement);
187 $total_rows = mysqli_stmt_num_rows($statement);
188 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*
- *muscular_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, muscular_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, $muscular_zone);
79 while ($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**

```
143 if(isset($_GET['search'])) {
144
145     // select all data
146     $params = $_GET['search'];
147     $query = $mysqli->prepare( query: "SELECT id_exercise, name, description, muscular_zone FROM exercise WHERE name = ? LIMIT $from_record_num,$records_per_page");
148     $query->bind_param( types: 's', &var1: $params);
149     $query->execute();
150     $query->store_result();
151     $result = $query->num_rows;
152
153     //check if more than 0 record found
154     if ($result > 0){
```

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*

```
186 // PAGINATION
187 // count total number of rows
188 $statement = mysqli_prepare($mysqli, query: "SELECT * FROM exercise WHERE name = ?");
189 mysqli_stmt_bind_param($statement, types: 's', &var1: $params);
190 mysqli_stmt_execute($statement);
191 mysqli_stmt_store_result($statement);
192 $total_rows = mysqli_stmt_num_rows($statement);
193 mysqli_stmt_close($statement);
194
195 // paginate records
196 $page_url = "read_schedules.php?";
197 include_once "paging.php";
```

Quest'operazione serve per l'impaginazione.

Input:

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

Output:

- *ALL*

- manage_schedules/search_user.php

```
141     if(isset($_GET['search'])){
142         $params = $_GET['search'];
143         $query1= $mysqli->prepare( query: "SELECT id_user, name, surname, email FROM user WHERE surname = ? ORDER BY id_user DESC LIMIT $from_record_num,$records_per_page");
144         $query1->bind_param( types: 's', &var1: $params);
145         $query1->execute();
146         $query1->store_result();
147         $result = $query1->num_rows;
148
149         //check if more than 0 record found
150         if ($result > 0){
```

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*

```
188     // PAGINATION
189     // count total number of rows
190     $statement = mysqli_prepare($mysqli, query: "SELECT * FROM user WHERE surname = ?");
191     mysqli_stmt_bind_param($statement, types: 's', &var1: $_GET['search']);
192     mysqli_stmt_execute($statement);
193     mysqli_stmt_store_result($statement);
194     $total_rows = mysqli_stmt_num_rows($statement);
195     mysqli_stmt_close($statement);
196
197     // paginate records
198     $page_url="search_user.php?";
199     include_once "paging.php";
```

Quest'operazione serve per l'impaginazione.

Input:

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

Output:

- *ALL*

- `manage_schedules/update_exercise.php`

```

78 // read current record's data
79 try {
80 // prepare select query
81 $query = $mysqli->prepare( "SELECT id_exercise, day, details, weight, repetitions FROM exercise_list WHERE id_list = ? LIMIT 0,1");
82 $query->bind_param( types: 'i', &var1: $id);
83 $query->execute();
84 $query->store_result();
85 $result = $query->num_rows;
86
87 //check if more than 0 record found
88 if ($result > 0){
89 $query->bind_result( &var1: $id_exercise, &var2: $day, $details, $weight, $repetitions);
90 while ($row = $query->fetch()){
91
92     $query1 = $mysqli->prepare( query: "SELECT name FROM exercise WHERE id_exercise =?");
93     $query1->bind_param( types: 'i', &var1: $id_exercise);
94     $query1->execute();
95     $query1->store_result();
96     $query1->bind_result( &var1: $name);
97
98     while ($row1 = $query1->fetch() ) {
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161

```

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`

```

78 // read current record's data
79 try {
80     // prepare select query
81     $query = $mysqli->prepare( query: "SELECT name, description, muscular_zone, url FROM exercise WHERE id_exercise = ? LIMIT 0,1");
82     $query->bind_param( types: 'i', &var1: $id);
83     $query->execute();
84     $query->store_result();
85     $result = $query->num_rows;
86
87     //check if more than 0 record found
88     if ($result > 0) {
89
107 if($_POST){
108     try{
109         $query_up = $mysqli->prepare( query: "UPDATE exercise SET name=?, description=?, muscular_zone=?, url=? WHERE id_exercise = ?");
110         $query_up->bind_param( types: 'ssssi', &var1: $_POST['name'], &var2: $_POST['description'], &var3: $_POST['muscular_zone'], &var4: $_POST['url'], &var5: $id);
111
112         // Execute the query
113         if ($query_up->execute() === true ) {
114             // prepare select query
115             $query = $mysqli->prepare( query: "SELECT name, description, muscular_zone, url FROM exercise WHERE id_exercise = ? LIMIT 0,1");
116             $query->bind_param( types: 'i', &var1: $id);
117             $query->execute();
118             $query->store_result();
119             $result = $query->num_rows;
120
121             //check if more than 0 record found
122             if ($result > 0 ) {
123                 $query->bind_result( &var1: $name, &var2: $description, &var3: $muscular_zone, &var4: $url);

```

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

Input:

- *id_exercise* (identificativo dell'esercizio)

Output:

- *name*
- *description*
- *muscular_zone*
- *url*

- `manage_user/autocomplete.php`

```

8 if(isset($_POST["query"]))
9 {
10     $output = '';
11     $param = "%".$_POST['query']."%";
12     $query = $mysqli->prepare( query: "SELECT surname FROM user WHERE surname LIKE ? LIMIT 5");
13     $query->bind_param( types: 's', &var1: $param);
14     $query->execute();
15     $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**

```
29 public function registerOperation($name, $surname, $email, $passwordHash, $address, $birthdate, $phone, $image, $subscription, $tipology, $token){
30     if(!$this->isEmailExist($email)){
31         $stmt = $this->con->prepare( query: "INSERT INTO user (name, surname, email, password, address, birth_date, phone, image, subscription, tipology, token_firebase) VALUES
32         $stmt->bind_param( types: "ssssssss", &var1: $name, &var2: $surname, $email, $passwordHash, $address, $birthdate, $phone, $image, $subscription, $tipology, $token);
33         if($stmt->execute())
```

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**

```
17 // delete query
18 $query = $mysqli->prepare( query: "DELETE FROM user WHERE id_user = ?");
19 $query->bind_param( types: 'i', &var1: $id);
20
21 // Execute the query
22 if($query->execute() === true){
```

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();
77
78     // write update query
79     $query_up = $mysqli->prepare( query: "UPDATE user SET name=?, surname=?, email=?, birth_date=?, address=?, id_subscription=? WHERE id_user = ?");
80     $query_up->bind_param( types: 'sssssi', &var1: $new_name, &var2: $new_surname, &var3: $new_email, &var4: $new_birth, &var5: $new_addr, &var6: $new_sub, &var7: $id);
81     // Execute the query
82     if($query_up->execute() === true){
83
84         // read current record's data
85         try {
86             // prepare select query
87             $query = $mysqli->prepare( query: "SELECT id_user, name, surname, email, birth_date, address, id_subscription FROM user WHERE id_user = ? LIMIT 0,1");
88             $query->bind_param( types: 'i', &var1: $id);
89             $query->execute();
90             $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**

```
11      $param = "%{$_POST['query']}%";
12      $query = $mysqli->prepare( query: "SELECT title FROM messages WHERE title LIKE ? LIMIT 5");
13      $query->bind_param( types: 's', &var1: $param);
14      $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**

```
97      if($_POST['destination'] == 'token'){
98          $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?, ?, ?, ?)");
99          $query->bind_param( types: 'ssss', &var1: $title, &var2: $body, &var3: $datetime, &var4: $argument[0]);
100          if($query->execute() == true){
101              echo "<div class='alert alert-success'>Message has been saved.</div>";
102              $load = true;
103          }else{
104              echo "<div class='alert alert-danger'>Unable to save the message. Please try again.</div>";
105          }
106      }
107      elseif($_POST['destination'] == 'multicast'){
108          $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?, ?, ?, ?)");
109          $query->bind_param( types: 'ssss', &var1: $title, &var2: $body, &var3: $datetime, &var4: $argument[1]);
110          if($query->execute() == true){
111              echo "<div class='alert alert-success'>Message has been saved.</div>";
112              header( string: 'Location:read_messages.php');
113          }else{
114              echo "<div class='alert alert-danger'>Unable to save the message. Please try again.</div>";
115          }
116      }
117      else{
118          $query = $mysqli->prepare( query: "INSERT INTO messages (title, body, send_date, destination) VALUES (?, ?, ?, ?)");
119          $query->bind_param( types: 'ssss', &var1: $title, &var2: $body, &var3: $datetime, &var4: $argument);
120          if($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**

```
137 // select all data
138
139 $query = "SELECT id_message, title, send_date, destination FROM messages ORDER BY send_date DESC LIMIT $from_record_num,$records_per_page";
140 $result = $mysqli->query($query);
141
142 //check if more than 0 record found
143 if ($mysqli_num_rows($result)) {
```

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*

```
175 // PAGINATION
176 // count total number of rows
177 $statement = mysqli_prepare($mysqli, query: 'SELECT * FROM messages');
178 mysqli_stmt_execute($statement);
179 mysqli_stmt_store_result($statement);
180 $total_rows = mysqli_stmt_num_rows($statement);
181 mysqli_stmt_close($statement);
182
183 // paginate records
184 $page_url="read_messages.php?";
185 include_once "paging.php";
```

Questo serve per l'impaginazione. Semplice SELECT.

- **send_messages/read_one_message.php**

```
76 try {
77     // prepare select query
78     $query = $mysqli->prepare( query: "SELECT id_message, title, body, send_date, destination FROM messages WHERE id_message = ? LIMIT 0,1");
79     $query->bind_param( types: 'i', &var1: $id);
80     $query->execute();
81     $result = $query->get_result()->fetch_assoc();
```

Questo serve per la lettura dei dettagli del singolo messaggio.

Input:

- *id_message*

Output:

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

- **send_messages/search.php**

```
143 if(isset($_GET['search'])){
144     $query1= $mysqli->prepare( query: "SELECT id_message, title, body, send_date, destination FROM messages WHERE title = ? ORDER BY send_date DESC LIMIT {$_from_record_num},$_records");
145     $query1->bind_param( types: 's', &var1: $_GET['search']);
146     $query1->execute();
147     $query1->store_result();
148     $result = $query1->num_rows;
```

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*

```
182 // PAGINATION
183 // count total number of rows
184 $statement = mysqli_prepare($mysqli, query: "SELECT * FROM messages WHERE title = ?");
185 mysqli_stmt_bind_param($statement, types: 's', &var1: $_GET['search']);
186 mysqli_stmt_execute($statement);
187 mysqli_stmt_store_result($statement);
188 $total_rows = mysqli_stmt_num_rows($statement);
189 mysqli_stmt_close($statement);
190
191 // paginate records
192 $page_url="search_user.php?";
193 include_once "paging.php";
```

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.