

```
{
  "_readme": [
    "This file locks the dependencies of your project to a known state",
    "Read more about it at https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies",
    "This file is @generated automatically"
  ],
  "content-hash": "85e07ea1e3620c3da1942492052d7d21",
  "packages": [
    {
      "name": "phpmailer/phpmailer",
      "version": "v6.1.5",
      "source": {
        "type": "git",
        "url": "https://github.com/PHPMailer/PHPMailer.git",
        "reference": "a8bf068f64a580302026e484ee29511f661b2ad3"
      },
      "dist": {
        "type": "zip",
        "url": "https://api.github.com/repos/PHPMailer/PHPMailer/zipball/a8bf068f64a580302026e484ee29511f661b2ad3",
        "reference": "a8bf068f64a580302026e484ee29511f661b2ad3",
        "shasum": ""
      },
      "require": {
        "ext-ctype": "*",
        "ext-filter": "*",
        "php": ">=5.5.0"
      },
      "require-dev": {
        "doctrine/annotations": "^1.2",
        "friendsofphp/php-cs-fixer": "^2.2",
        "phpunit/phpunit": "^4.8 || ^5.7"
      },
      "suggest": {
        "ext-mbstring": "Needed to send email in multibyte encoding charset",
        "hayageek/oauth2-yahoo": "Needed for Yahoo XOAUTH2 authentication",
        "league/oauth2-google": "Needed for Google XOAUTH2 authentication",
        "psr/log": "For optional PSR-3 debug logging",
        "stevenmaguire/oauth2-microsoft": "Needed for Microsoft XOAUTH2 authentication",
        "symfony/polyfill-mbstring": "To support UTF-8 if the Mbstring PHP extension is not enabled (^1.2)"
      },
      "type": "library",
      "autoload": {
        "psr-4": {
          "PHPMailer\\PHPMailer\\": "src/"
        }
      },
      "notification-url": "https://packagist.org/downloads/",
      "license": [
        "LGPL-2.1-only"
      ]
    }
  ],
}
```

```
51         "authors": [  
52             {  
53                 "name": "Marcus Bointon",  
54                 "email": "phpmailer@synchromedia.co.uk"  
55             },  
56             {  
57                 "name": "Jim Jagielski",  
58                 "email": "jimjag@gmail.com"  
59             },  
60             {  
61                 "name": "Andy Prevost",  
62                 "email": "codeworxtech@users.sourceforge.net"  
63             },  
64             {  
65                 "name": "Brent R. Matzelle"  
66             }  
67         ],  
68         "description": "PHPMailer is a full-featured email creation and transfer  
class for PHP",  
69         "funding": [  
70             {  
71                 "url": "https://marcus.bointon.com/donations/",  
72                 "type": "custom"  
73             },  
74             {  
75                 "url": "https://github.com/Synchro",  
76                 "type": "github"  
77             },  
78             {  
79                 "url": "https://www.patreon.com/marcusbointon",  
80                 "type": "patreon"  
81             }  
82         ],  
83         "time": "2020-03-14T14:23:48+00:00"  
84     }  
85 ],  
86 "packages-dev": [],  
87 "aliases": [],  
88 "minimum-stability": "stable",  
89 "stability-flags": [],  
90 "prefer-stable": false,  
91 "prefer-lowest": false,  
92 "platform": [],  
93 "platform-dev": [],  
94 "plugin-api-version": "1.1.0"  
95 }
```

```
1 {  
2     "require": {  
3         "phpmailer/phpmailer": "^6.1"  
4     }  
5 }
```

```
1 .gitignore
2 .vscode
3 /vendor/
4 /database/schema.mwb.bak
5 /views/assets/upload/
-
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : register.php
 * Date        : 26.05.2020
 * Description  : Formulaire d'inscription
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (IsLogged()) {
    header('Location: ./index.php');
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>Inscription</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    
    <div class="container">
        <div class="img">
```

```
54     
55 </div>
56 <div class="content">
57     <form method="post">
58         
59         <h2 class="title">Inscription</h2>
60         <div class="input-div one">
61             <div class="i">
62                 <i class="fas fa-user"></i>
63             </div>
64             <div class="div">
65                 <h5>Nom</h5>
66                 <input type="text" class="input" id="lastname" maxlength="50">
67             </div>
68         </div>
69         <div class="input-div one">
70             <div class="i">
71                 <i class="fas fa-user"></i>
72             </div>
73             <div class="div">
74                 <h5>Prénom</h5>
75                 <input type="text" class="input" id="firstname" maxlength="50">
76             </div>
77         </div>
78         <div class="input-div one">
79             <div class="i">
80                 <i class="fas fa-user"></i>
81             </div>
82             <div class="div">
83                 <h5>Nom d'utilisateur</h5>
84                 <input type="text" class="input" id="nickname" maxlength="50">
85             </div>
86         </div>
87         <div class="input-div one">
88             <div class="i">
89                 <i class="fa fa-envelope" aria-hidden="true"></i>
90             </div>
91             <div class="div">
92                 <h5>Email</h5>
93                 <input type="email" class="input" id="email" maxlength="100">
94             </div>
95         </div>
96         <div class="input-div one">
97             <div class="i">
98                 <i class="fas fa-phone-alt"></i>
99             </div>
100             <div class="div">
101                 <h5>Téléphone</h5>
102                 <input type="phone" class="input" id="phoneNumber" maxlength="50">
103             </div>
104         </div>
105         <div class="input-div pass">
106             <div class="i">
107                 <i class="fas fa-lock"></i>
108             </div>
```

```
109     <div class="div">
110         <h5>Mot de passe</h5>
111         <input type="password" class="input" id="password">
112     </div>
113 </div>
114 <div class="input-div pass">
115     <div class="i">
116         <i class="fas fa-lock"></i>
117     </div>
118     <div class="div">
119         <h5>Confirmation mot de passe</h5>
120         <input type="password" class="input" id="verifyPassword">
121     </div>
122 </div>
123 <input type="submit" id="btnRegister" class="btn" value="Inscription">
124 </form>
125 </div>
126 </div>
127
128 <!-- JAVASCRIPT INCLUDES -->
129 <?php include "../includes/footer.inc.html"; ?>
130
131 <script src="./js/register.js"></script>
132 </body>
133
134 </html>
135
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : profile.php
 * Date        : JJ.03.2020
 * Description  : Page qui affiche els données du profile ainsi que son formulaire de
modification.
 * Version     : 1.0.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
if (IsLogged() == false) {
    header('Location: ./login.php');
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">
    <title>Profile</title>
</head>

<body>
    <?php include './includes/navbar.inc.php' ?>

    
    <div class="container">
        <div class="img">
            
        </div>
    </div>
</body>
</html>
```



```
53
54     <div class="content" id="userData"></div>
55 </div>
56
57 <?php include './includes/footer.inc.html' ?>
58
59 <script src="./js/profile.js"></script>
60 </body>
61
62 </html>
--
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : manageEvents.php
 * Date        : 03.06.2020
 * Description  : Page qui permet de gérer les événements.
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (IsLogged() == false) {
    header('Location: ./login.php');
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <title>Gérer mes événements</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    <div id="events"></div>

    <?php include './includes/footer.inc.html'; ?>

    <script src="./js/events.js"></script>
```

```
54 </body>
55
56 </html>
--
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : login.php
 * Date        : 26.05.2020
 * Description  : Formulaire de connexion
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (IsLogged()) {
    header('Location: ./index.php');
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>Connexion</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    
    <div class="container">
        <div class="img">
            
```

```
54     </div>
55     <div class="content">
56         <form method="post">
57             
58             <h2 class="title">Bienvenue</h2>
59             <div class="input-div one">
60                 <div class="i">
61                     <i class="fas fa-user"></i>
62                 </div>
63                 <div class="div">
64                     <h5>Nom d'utilisateur ou Email</h5>
65                     <input type="text" class="input" id="authenticator">
66                 </div>
67             </div>
68             <div class="input-div pass">
69                 <div class="i">
70                     <i class="fas fa-lock"></i>
71                 </div>
72                 <div class="div">
73                     <h5>Mot de passe</h5>
74                     <input type="password" class="input" id="password">
75                 </div>
76             </div>
77             <input type="submit" id="btnLogUser" class="btn" value="Connexion">
78         </form>
79     </div>
80 </div>
81
82 <?php include './includes/footer.inc.html'; ?>
83
84 <script type="text/javascript" src="./js/login.js"></script>
85 </body>
86
87 </html>
88
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : index.php
 * Date        : 26.05.2020
 * Description  : Page d'accueil du site.
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>Accueil</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    <div id="events"></div>

    <?php include './includes/footer.inc.html'; ?>

    <script src="./js/events.js"></script>
</body>

</html>
```

2

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : event.php
 * Date        : 02.06.2020
 * Description  : Page qui affiche les informations d'un événement spécifique.
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title></title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    
    <div class="container">
        <div class="img">
            
        </div>

        <div class="content" id="eventData"></div>
    </div>
```



```
54
55 <?php include './includes/footer.inc.html'; ?>
56
57 <script src="./js/events.js"></script>
58 </body>
59
60 </html>
--
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : editProfile.php
 * Date        : 05.06.2020
 * Description  : Formulaire de modification du profile
 * Version     : 1.0
 */
require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idUser = filter_input(INPUT_GET, 'idUser', FILTER_SANITIZE_NUMBER_INT);

if (!IsLogged() || $idUser != $_SESSION['loggedIn']['idUser']) {
    header('Location: ./index.php');
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>Modifier mon profile</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    
    <div class="container">
        <div class="img">
```

```
54     
55 </div>
56 <div class="content">
57     <form method="post">
58         
59         <h2 class="title">Modifier mon profile</h2>
60         <div class="input-div one">
61             <div class="i">
62                 <i class="fas fa-user"></i>
63             </div>
64             <div class="div">
65                 <h5>Nom</h5>
66                 <input type="text" class="input" id="lastname" maxlength="50" value="<?=  
isset($_SESSION['loggedIn']['nom']) ? $_SESSION['loggedIn']['nom'] : '' ?>">
67             </div>
68         </div>
69         <div class="input-div one">
70             <div class="i">
71                 <i class="fas fa-user"></i>
72             </div>
73             <div class="div">
74                 <h5>Prénom</h5>
75                 <input type="text" class="input" id="firstname" maxlength="50" value="<?=  
$_SESSION['loggedIn']['prenom'] ?>">
76             </div>
77         </div>
78         <div class="input-div one">
79             <div class="i">
80                 <i class="fas fa-user"></i>
81             </div>
82             <div class="div">
83                 <h5>Nom d'utilisateur</h5>
84                 <input type="text" class="input" id="nickname" maxlength="50" value="<?=  
$_SESSION['loggedIn']['pseudo'] ?>">
85             </div>
86         </div>
87         <div class="input-div one">
88             <div class="i">
89                 <i class="fa fa-envelope" aria-hidden="true"></i>
90             </div>
91             <div class="div">
92                 <h5>Email</h5>
93                 <input type="email" class="input" id="email" maxlength="100" value="<?=  
$_SESSION['loggedIn']['email'] ?>">
94             </div>
95         </div>
96         <div class="input-div one">
97             <div class="i">
98                 <i class="fas fa-phone-alt"></i>
99             </div>
100             <div class="div">
101                 <h5>Téléphone</h5>
102                 <input type="phone" class="input" id="phoneNumber" maxlength="50"  
value="<?=  
isset($_SESSION['loggedIn']['telephone']) ? $_SESSION['loggedIn']  
['telephone'] : '' ?>">
```

```
103     </div>
104 </div>
105 <div class="input-div pass">
106     <div class="i">
107         <i class="fas fa-lock"></i>
108     </div>
109     <div class="div">
110         <h5>Mot de passe</h5>
111         <input type="password" class="input" id="password">
112     </div>
113 </div>
114 <div class="input-div pass">
115     <div class="i">
116         <i class="fas fa-lock"></i>
117     </div>
118     <div class="div">
119         <h5>Confirmation mot de passe</h5>
120         <input type="password" class="input" id="verifyPassword">
121     </div>
122 </div>
123 <input type="submit" id="btnRegister" class="btn" onclick="EditProfile(event)"
value="Modifier mon profile">
124 </form>
125 </div>
126 </div>
127
128 <!-- JAVASCRIPT INCLUDES -->
129 <?php include "../includes/footer.inc.html"; ?>
130
131 <script src="../js/profile.js"></script>
132 </body>
133
134 </html>
---
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : editEvent.php
 * Date        : 03.06.2020
 * Description  : Formulaire de modification d'un événement
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (!IsLogged()) {
    header('Location: ./index.php');
    exit();
}

$idEvent = filter_input(INPUT_GET, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
$idUser = $_SESSION['loggedIn']['idUser'];

if (IsCreator($idEvent, $idUser) == false) {
    header('Location: ./index.php');
    exit();
}

$eventData = GetEventData($idEvent, $idUser);

// dateDebut au format Y-m-d H:i:s
$eventBeginningDate = explode(' ', $eventData['dateDebut'])[0]; // Récupère la date au
format Y-m-d
$eventBeginningTime = explode(' ', $eventData['dateDebut'])[1]; // Récupère l'heure au
foormat H:m:s
$eventBeginningTime = explode(':', $eventBeginningTime)[0] . ':' . explode(':',
$eventBeginningTime)[1]; // Passe l'heure du format H:m:s au format H:m

// dateFin au format Y-m-d H:i:s
$eventEndDate = explode(' ', $eventData['dateFin'])[0]; // Récupère la date au format
Y-m-d
$eventEndTime = explode(' ', $eventData['dateFin'])[1]; // Récupère l'heure au foormat
H:m:s
$eventEndTime = explode(':', $eventEndTime)[0] . ':' . explode(':', $eventEndTime)[1];
// Passe l'heure du format H:m:s au format H:m

?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
```

```
49 <meta name="viewport" content="width=device-width, initial-scale=1.0">
50
51 <!-- STYLE DU SITE -->
52 <link rel="stylesheet" href="./css/style.css">
53 <link rel="stylesheet" href="./css/media.css">
54
55 <!-- SWEETALERT2 -->
56 <link rel="stylesheet" href="./css/sweetalert2.css">
57
58 <!-- GOOGLE FONTS -->
59 <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
60
61 <!-- FONTAWESOME -->
62 <script src="https://kit.fontawesome.com/a81368914c.js"></script>
63
64 <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">
65
66 <title>Modifier mon événement</title>
67 </head>
68
69 <body>
70 <?php include './includes/navbar.inc.php' ?>
71
72 
73 <div class="container">
74 <div class="img">
75 
76 </div>
77
78 <div class="content">
79 <form method="post">
80 
81 <h2 class="title">Modifier mon événement</h2>
82 <div class="input-div one">
83 <div class="i">
84 <i class="fa fa-calendar" aria-hidden="true"></i>
85 </div>
86 <div class="div">
87 <h5>Nom de l'événement</h5>
88 <input type="text" class="input" id="eventName" maxlength="50" value="<?=$eventData['nom'] ?>">
89 </div>
90 </div>
91 <div class="input-div one">
92 <div class="i">
93 <i class="fas fa-pen"></i>
94 </div>
95 <div class="div">
96 <h5>Descriptif de l'événement</h5>
97 <input type="text" class="input" id="eventDescription" maxlength="255"
value="<?=$eventData['descriptif'] ?>">
98 </div>
99 </div>
100 <div class="input-div one">
```

```
101         <div class="i">
102             <i class="fas fa-map-marker-alt"></i>
103         </div>
104         <div class="div">
105             <h5>Lieu</h5>
106             <input type="text" class="input" id="place" maxlength="255" value="<?=$eventData['lieu'] ?>" />
107         </div>
108     </div>
109     <div class="input-div one">
110         <div class="i">
111             <i class="fas fa-calendar-alt"></i>
112         </div>
113         <div class="div">
114             <h5>Date de début</h5>
115             <input class="input" id="beginningDate" type="text" onfocus="(this.type='date')" value="<?=$eventBeginningDate ?>" />
116         </div>
117     </div>
118     <div class="input-div one">
119         <div class="i">
120             <i class="fas fa-clock"></i>
121         </div>
122         <div class="div">
123             <h5>Heure de début</h5>
124             <input class="input" id="beginningTime" type="text" onfocus="(this.type='time')" value="<?=$eventBeginningTime ?>" />
125         </div>
126     </div>
127     <div class="input-div one">
128         <div class="i">
129             <i class="fas fa-calendar-alt"></i>
130         </div>
131         <div class="div">
132             <h5>Date de fin</h5>
133             <input class="input" id="endDate" type="text" onfocus="(this.type='date')" value="<?=$eventEndDate ?>" />
134         </div>
135     </div>
136     <div class="input-div one">
137         <div class="i">
138             <i class="fas fa-clock"></i>
139         </div>
140         <div class="div">
141             <h5>Heure de fin</h5>
142             <input class="input" id="endTime" type="text" onfocus="(this.type='time')" value="<?=$eventEndTime ?>" />
143         </div>
144     </div>
145     <div class="input-div one">
146         <div class="i">
147             <i class="fas fa-user"></i>
148         </div>
149         <div class="div">
150             <h5>Nombre maximal de participant</h5>
```

```
151         <input class="input" id="nbMaxGuest" type="number" min="0" value="<?=  
$eventData['nbMaxParticipant'] ?>" />  
152     </div>  
153 </div>  
154 <div class="input-div one">  
155     <div class="i">  
156         <i class="fas fa-images"></i>  
157     </div>  
158     <div class="div">  
159         <h5><label for="img">Image de l'événement</label></h5>  
160         <input class="input" name="img" id="img" type="file" accept="image/*" />  
161     </div>  
162 </div>  
163     <input type="submit" id="btnEditEvent" onclick="EditEvent(event, <?= $idEvent  
?>)" class="btn" value="Modifier l'événement">  
164 </form>  
165 </div>  
166  
167 <?php include './includes/footer.inc.html' ?>  
168 <script src="./js/events.js"></script>  
169 </body>  
170  
171 </html>  
---
```



```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : createEvent.php
 * Date        : 27.05.2020
 * Description  : Formulaire de création d'événement.
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (!IsLogged()) {
    header('Location: ./index.php');
    exit();
}

$datetime = new DateTime('tomorrow');
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>Créer un événement</title>
</head>

<body>
    <?php include './includes/navbar.inc.php' ?>

    
```

```
54 <div class="container">
55   <div class="img">
56     
57   </div>
58
59   <div class="content">
60     <form method="post">
61       
62       <h2 class="title">Cr  er un   v  nement</h2>
63       <div class="input-div one">
64         <div class="i">
65           <i class="fa fa-calendar" aria-hidden="true"></i>
66         </div>
67         <div class="div">
68           <h5>Nom de l'  v  nement</h5>
69           <input type="text" class="input" id="eventName" maxlength="50">
70         </div>
71       </div>
72       <div class="input-div one">
73         <div class="i">
74           <i class="fas fa-pen"></i>
75         </div>
76         <div class="div">
77           <h5>Descriptif de l'  v  nement</h5>
78           <input type="text" class="input" id="eventDescription" maxlength="255">
79         </div>
80       </div>
81       <div class="input-div one">
82         <div class="i">
83           <i class="fas fa-map-marker-alt"></i>
84         </div>
85         <div class="div">
86           <h5>Lieu</h5>
87           <input type="text" class="input" id="place" maxlength="255">
88         </div>
89       </div>
90       <div class="input-div one">
91         <div class="i">
92           <i class="fas fa-calendar-alt"></i>
93         </div>
94         <div class="div">
95           <h5>Date de d  but</h5>
96           <input class="input" id="beginningDate" type="text" onfocus="
97 (this.type='date')" value="<?= $datetime->format('Y-m-d'); ?>" />
98         </div>
99       </div>
100      <div class="input-div one">
101        <div class="i">
102          <i class="fas fa-clock"></i>
103        </div>
104        <div class="div">
105          <h5>Heure de d  but</h5>
106          <input class="input" id="beginningTime" type="text" onfocus="
107 (this.type='time')" value="<?= date('H:i'); ?>" />
108        </div>
109      </div>
110    </form>
111  </div>
112</div>
```

```
107     </div>
108     <div class="input-div one">
109         <div class="i">
110             <i class="fas fa-calendar-alt"></i>
111         </div>
112         <div class="div">
113             <h5>Date de fin</h5>
114             <input class="input" id="endDate" type="text" onfocus="(this.type='date')"
115         />
116         </div>
117     </div>
118     <div class="input-div one">
119         <div class="i">
120             <i class="fas fa-clock"></i>
121         </div>
122         <div class="div">
123             <h5>Heure de fin</h5>
124             <input class="input" id="endTime" type="text" onfocus="(this.type='time')"
125         />
126         </div>
127     </div>
128     <div class="input-div one">
129         <div class="i">
130             <i class="fas fa-user"></i>
131         </div>
132         <div class="div">
133             <h5>Nombre maximal de participant</h5>
134             <input class="input" id="nbMaxGuest" type="number" min="0" />
135         </div>
136     </div>
137     <div class="input-div one">
138         <div class="i">
139             <i class="fas fa-images"></i>
140         </div>
141         <div class="div">
142             <h5><label for="img">Image de l'événement</label></h5>
143             <input class="input" name="img" id="img" type="file" accept="image/*" />
144         </div>
145     </div>
146     <div class="radio-group">
147         <label class="radio">
148             <input type="radio" class="radio" name="type" id="public" value="public"
149             checked>
150             publique
151             <span></span>
152         </label>
153         <label class="radio">
154             <input type="radio" class="radio" name="type" id="private" value="private"
155             privé
156             <span></span>
157         </label>
158     </div>
159     <input type="submit" id="btnCreateEvent" class="btn" value="Créer un
160     événement">
161 </form>
```

```
158     </div>
159
160     <?php include './includes/footer.inc.html' ?>
161     <script src="./js/events.js"></script>
162 </body>
163
164 </html>
165
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : about.php
 * Date        : 08.06.2020
 * Description  : Page d'information sur le site
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/App/php/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- STYLE DU SITE -->
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="./css/media.css">

    <!-- SWEETALERT2 -->
    <link rel="stylesheet" href="./css/sweetalert2.css">

    <!-- GOOGLE FONTS -->
    <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

    <!-- FONTAWESOME -->
    <script src="https://kit.fontawesome.com/a81368914c.js"></script>

    <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">

    <title>But du site</title>
</head>

<body>
    <?php include './includes/navbar.inc.php'; ?>

    
    <div class="container">
        <div class="img">
            
        </div>
        <div class="content">
            <div class="container-about">
                <div class="div about">Ce site permet de créer des événements privés ou
```

```
    publiques afin que d'autres utilisateurs puissent y participer.</div>
54     <div class="div about">Si vous souhaitez en savoir plus sur le fonctionnement
    du site, le manuel utilisateur est disponible <a href="#">ici</a>.</div>
55
56     </div>
57 </div>
58 </div>
59
60 <?php include './includes/footer.inc.html'; ?>
61 </body>
62
63 </html>
..
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : 404.php
6  * Date        : 02.06.2020
7  * Description  : Page qui affiche l'erreur 404
8  * Version     : 1.0
9  */
10
11 require_once dirname(__DIR__) . '/App/php/backend.php';
12 ?>
13 <!DOCTYPE html>
14 <html lang="en">
15
16 <head>
17     <meta charset="UTF-8">
18     <meta name="viewport" content="width=device-width, initial-scale=1.0">
19     <!-- STYLE DU SITE -->
20     <link rel="stylesheet" href="./css/style.css">
21     <link rel="stylesheet" href="./css/media.css">
22
23     <!-- SWEETALERT2 -->
24     <link rel="stylesheet" href="./css/sweetalert2.css">
25
26     <!-- GOOGLE FONTS -->
27     <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
28     rel="stylesheet">
29
30     <!-- FONTAWESOME -->
31     <script src="https://kit.fontawesome.com/a81368914c.js"></script>
32
33     <link rel="shortcut icon" href="./assets/img/iconeWego.svg" type="image/x-icon">
34
35     <title>Page Not Found</title>
36 </head>
37 <body>
38     <?php include './includes/navbar.inc.php'; ?>
39
40     
41
42 </body>
43
44 </html>
45
```

```
$(document).ready(() => {
    $("#btnRegister").click(Register);
});

/**
 * @author Hoarau Nicolas
 * @date 26.05.20
 * @brief fonction qui filtre les données du formulaire d'inscription et les envoie au
serveur via un call Ajax
 * @param {*} event
 * @version 1.0
 */
function Register(event) {
    if (event)
        event.preventDefault();

    let lastname = $('#lastname').val();
    let firstname = $('#firstname').val();
    let nickname = $('#nickname').val();
    let email = $('#email').val();
    let phoneNumber = $('#phoneNumber').val();
    let password = $('#password').val();
    let verifyPassword = $('#verifyPassword').val();

    if (firstname.length == 0) {
        $('#firstname').focus();
        return;
    }

    if (nickname.length == 0) {
        $('#nickname').focus();
        return;
    }

    if (email.length == 0) {
        $('#email').focus();
        return;
    }

    if (password.length == 0) {
        $('#password').focus();
        return;
    }

    if (verifyPassword.length == 0) {
        $('#verifyPassword').focus();
        return;
    }

    $.ajax({
        type: "post",
        url: "../App/php/register.php",
        data: {
            lastname: lastname,
```



```
54     firstname: firstname,
55     nickname: nickname,
56     email: email,
57     phoneNumber: phoneNumber,
58     password: password,
59     verifyPassword: verifyPassword
60 },
61 dataType: "json",
62 success: (response) => {
63     switch (response.ReturnCode) {
64         case 0:
65             Swal.fire({
66                 title: "Inscription",
67                 text: response.Success,
68                 icon: "success",
69                 timer: 1300,
70                 showConfirmButton: false
71             }).then(() => {
72                 window.location.href = './login.php'
73             });
74             break;
75         case 1:
76         case 2:
77         case 3:
78         case 4:
79         case 5:
80             Swal.fire({
81                 title: "Inscription",
82                 text: response.Error,
83                 icon: "error",
84                 timer: 1500,
85                 showConfirmButton: false
86             });
87             break;
88     }
89 },
90 error: (error) => {
91     console.error(error);
92 }
93 });
94 }
```

```
$(document).ready(() => {
    let pageName = window.location.pathname.substring(
        location.pathname.lastIndexOf("/") + 1
    );

    switch (pageName) {
        case "editProfile.php":
            $("#lastname").focus();
            $("#firstname").focus();
            $("#nickname").focus();
            $("#email").focus();
            $("#phoneNumber").focus();
            break;
        case 'profile.php':
            GetProfileData();
            break;
    }
});

/**
 * @author Hoarau Nicolas
 * @date 03.06.20
 * @brief Fonction qui récupère les informations de l'utilisateur
 */
function GetProfileData() {
    $.ajax({
        type: "post",
        url: "../App/php/getUserData.php",
        dataType: "json",
        success: (response) => {
            switch (response.ReturnCode) {
                case 0:
                    ShowProfileData(response.Data);
                    break;
                case 1:
                    Swal.fire({
                        title: "Mon profil",
                        text: response.Error,
                        icon: "error",
                        timer: 1300,
                        showConfirmButton: false
                    });
                    break;
            }
        },
        error: (error) => {
            console.error(error);
        }
    });
}

/**
 * @author Hoarau Nicolas
 * @date 03.06.20
 */
```

```
55 * @brief Fonction qui affiche les informations de l'utilisateur
56 * @param {array} data
57 */
58 function ShowProfileData(data) {
59     document.title = data.pseudo;
60
61     let html = `<div class="event-data">
62     
63     <h2 class="title">${data.pseudo}</h2>
64     <div class="div">${data.nom == null ? '' : data.nom} ${data.prenom}</div>
65     <div class="div">${data.email}</div>
66     <div class="div place">${data.telephone == null ? '' : data.telephone}</div>
67     <button class="btn btn-edit" onclick="window.location.href =
68     './editProfile.php?idUser=${data.idUser}'">Modifier</button>
69     <button class="btn btn-delete"
70     onclick="DeleteProfile(${data.idUser})">Supprimer</button>`;
71
72     html += `</div>`;
73
74     $("#userData").html(html);
75 }
76 /**
77 * @author Hoarau Nicolas
78 * @date 04.06.20
79 * @brief Fonction qui affiche les informations de l'utilisateur
80 * @param {int} idUser
81 */
82 function DeleteProfile(idUser) {
83     $.ajax({
84         type: "post",
85         url: "../App/php/isProfileDeletable.php",
86         data: { idUser: idUser },
87         dataType: "json",
88         success: (response) => {
89             switch (response.ReturnCode) {
90                 case 0:
91                     Swal.fire({
92                         title: "Suppression du profile",
93                         text: response.Success,
94                         icon: "question",
95                         showConfirmButton: true,
96                         confirmButtonText: 'Valider',
97                         showCancelButton: true,
98                         cancelButtonText: 'Annuler',
99                         reverseButtons: true
100                     }).then((result) => {
101                         if (result.value) {
102                             $.ajax({
103                                 type: "post",
104                                 url: "../App/php/deleteProfile.php",
105                                 data: { idUser: idUser },
106                                 dataType: "json",
107                                 success: (data) => {
108                                     switch (data.ReturnCode) {
```

```
108         case 0:
109             Swal.fire({
110                 title: "Suppression du profile",
111                 text: data.Success,
112                 icon: "success",
113                 timer: 13000,
114                 showConfirmButton: false,
115             }).then(() => {
116                 window.location.href = '../App/php/logout.php';
117             });
118             break;
119         case 1:
120             Swal.fire({
121                 title: "Suppression du profile",
122                 text: data.Error,
123                 icon: "error",
124                 timer: 1500,
125                 showConfirmButton: false,
126             });
127             break;
128     }
129 },
130 error: (error) => {
131     console.error(error);
132 }
133 });
134 }
135 });
136 break;
137 case 1:
138     Swal.fire({
139         title: "Suppression du profile",
140         text: response.Error,
141         icon: "error",
142         timer: 1800,
143         showConfirmButton: false
144     });
145     break;
146 }
147 },
148 error: (error) => {
149     console.error(error);
150 }
151 });
152 }
153
154 /**
155  * @author Hoarau Nicolas
156  * @date 04.06.20
157  * @brief Fonction qui affiche les informations de l'utilisateur
158  * @param {*} event
159  */
160 function EditProfile(event) {
161     if (event)
162         event.preventDefault();
```

```
163
164 let lastname = $('#lastname').val();
165 let firstname = $('#firstname').val();
166 let nickname = $('#nickname').val();
167 let email = $('#email').val();
168 let phoneNumber = $('#phoneNumber').val();
169 let password = $('#password').val();
170 let verifyPassword = $('#verifyPassword').val();
171
172 if (firstname.length == 0) {
173     $('#firstname').focus();
174     return;
175 }
176
177 if (nickname.length == 0) {
178     $('#nickname').focus();
179     return;
180 }
181
182 if (email.length == 0) {
183     $('#email').focus();
184     return;
185 }
186
187 $.ajax({
188     type: "post",
189     url: "../App/php/editProfile.php",
190     data: {
191         lastname: lastname,
192         firstname: firstname,
193         nickname: nickname,
194         email: email,
195         phoneNumber: phoneNumber,
196         password: password,
197         verifyPassword: verifyPassword
198     },
199     dataType: "json",
200     success: (response) => {
201         switch (response.ReturnCode) {
202             case 0:
203                 Swal.fire({
204                     title: "Modification du profile",
205                     text: response.Success,
206                     icon: "success",
207                     timer: 1300,
208                     showConfirmButton: false
209                 }).then(() => {
210                     window.location.href = '../profile.php'
211                 });
212                 break;
213             case 1:
214             case 2:
215             case 3:
216                 Swal.fire({
217                     title: "Modification du profile",
```

```
218         text: response.Error,
219         icon: "error",
220         timer: 1500,
221         showConfirmButton: false
222     });
223     break;
224 }
225 },
226 error: (error) => {
227     console.error(error);
228 }
229 });
230 ,
```

```
1 // Conatantes login
2 const inputs = document.querySelectorAll(".input");
3
4 // Constantes navbar
5 const hamburger = document.querySelector('.hamburger');
6 const navlinks = document.querySelector('.nav-links');
7 const links = document.querySelectorAll('.nav-links li');
8
9 /**
10  * @brief ajoute le focus sur l'input cliqué
11  */
12 function addcl() {
13   let parent = this.parentNode.parentNode;
14   parent.classList.add("focus");
15 }
16
17 /**
18  * @brief fonction qui retire le focus de l'input
19  */
20 function remcl() {
21   let parent = this.parentNode.parentNode;
22   if (this.value == "") {
23     parent.classList.remove("focus");
24   }
25 }
26
27 // ajoute aux input les evnt focus et blur avec addcl et remcl en fontion
28 inputs.forEach(input => {
29   input.addEventListener("focus", addcl);
30   input.addEventListener("blur", remcl);
31 });
32
33 // ajoute l'event click au hamburger et fait l'affichage de la navbar
34 hamburger.addEventListener('click', () => {
35   let content = document.getElementsByClassName('content')[0];
36
37   navlinks.classList.toggle('open');
38
39   content.style.display == 'none' ? content.style.display = 'flex' :
40   content.style.display = 'none';
41
42   links.forEach(link => {
43     link.classList.toggle('fade');
44   })
45 .. ..
```

```
$(document).ready(() => {
    $("#btnLogUser").click(Login);
});

/**
 * @author Hoarau Nicolas
 * @date 20.03.20
 * @brief Fonction qui rcupère les données du formulaire et les envoie au serveur via
un call ajax
 * @param {*} event
 * @version 1.0
 */
function Login(event) {
    if (event)
        event.preventDefault();

    let authenticator = $('#authenticator').val();
    let password = $('#password').val();

    if (authenticator.length == 0) {
        $('#authenticator').focus();
        return;
    }

    if (password.length == 0) {
        $("#password").focus();
        return;
    }

    $.ajax({
        type: "post",
        url: "../App/php/login.php",
        data: { authenticator: authenticator, password: password },
        dataType: "json",
        success: (response) => {
            switch (response.ReturnCode) {
                case 0:
                    Swal.fire({
                        title: "Authentification",
                        text: response.Success,
                        icon: "success",
                        timer: 1300,
                        showConfirmButton: false
                    }).then(() => {
                        window.location.href = './index.php'
                    });
                    break;
                case 1:
                    Swal.fire({
                        title: "Authentification",
                        text: response.Error,
                        icon: "error",
                        timer: 1300,
                        showConfirmButton: false
                    });
            }
        }
    });
}
```



```
54         });  
55         break;  
56     }  
57 },  
58 error: (error) => {  
59     console.error(error);  
60 }  
61 });  
62 ,
```

```
const Toast = Swal.mixin({
  toast: true,
  position: 'top-end',
  showConfirmButton: false,
  timer: 1500,
  timerProgressBar: true
});

var pageName = window.location.pathname.substring(
  location.pathname.lastIndexOf("/") + 1
);

$(document).ready(() => {
  switch (pageName) {
    case "index.php":
    case "":
      GetPublicEvents();
      break;
    case "createEvent.php":
      $("#beginningDate").focus();
      $("#beginningTime").focus();
      $("#name").focus();
      break;
    case 'event.php':
      let urlParams = new URLSearchParams(window.location.search);
      let idEvent = urlParams.get('idEvent');

      GetEventData(idEvent);
      break;
    case 'manageEvents.php':
      GetManageEvents();
      break;
    case 'editEvent.php':
      $("#nbMaxGuest").focus();
      $("#eventDescription").focus();
      $("#beginningDate").focus();
      $("#beginningTime").focus();
      $("#endDate").focus();
      $("#endTime").focus();
      $("#place").focus();
      $("#eventName").focus();
      break;
  }

  $("#btnCreateEvent").click(CreateEvent);
});

// Change de couleur l'icone de l'image lorsqu'une image est sélectionné dans la
// création ou la modification de l'événement
$("#img").on('change', () => {
  $("#img")[0].parentNode.parentNode.children[0].style.color = "#38d39f";
});

/**
```

```
54 * @author Hoarau Nicolas
55 * @date 27.05.20
56 * @brief Fonction qui envoie les données pour créer un événement via un call ajax
57 * @param {*} event
58 * @version 1.0
59 */
60 async function CreateEvent(event) {
61   if (event)
62     event.preventDefault();
63
64   //#region Initialisation
65   let formdata = new FormData();
66   let eventName = $("#eventName").val();
67   let place = $("#place").val();
68   let eventDescription = $("#eventDescription").val();
69   let nbMaxGuest = $("#nbMaxGuest").val();
70   let type = $('input[name="type"]:checked').val();
71   let img = document.getElementById("img").files[0];
72   let beginningDate = new Date($("#beginningDate").val());
73   let endDate = new Date($("#endDate").val());
74   let beginningTime = $("#beginningTime").val();
75   let endTime = $("#endTime").val();
76   let now = new Date();
77
78   beginningDate.setHours(beginningTime.split(':')[0], beginningTime.split(':')[1], 0,
79 0);
80   endDate.setHours(endTime.split(':')[0], endTime.split(':')[1], 0, 0);
81   now.setHours(now.getHours(), now.getMinutes(), 0, 0);
82   //#endregion
83
84   //#region Conditions
85   if (eventName.length == 0) {
86     $("#eventName").focus();
87     return;
88   } else {
89     formdata.append("eventName", eventName);
90   }
91
92   if (eventDescription.length == 0) {
93     $("#eventDescription").focus();
94     return;
95   } else {
96     formdata.append("eventDescription", eventDescription);
97   }
98
99   if (place.length == 0) {
100     $("#place").focus();
101     return;
102   } else {
103     formdata.append("place", place);
104   }
105
106   if (nbMaxGuest == 0 || nbMaxGuest == "") {
107     $("#nbMaxGuest").focus();
108     return;
109   }
```

```
108 } else {
109     formdata.append("nbMaxGuest", nbMaxGuest);
110 }
111
112 if (beginningDate === null) {
113     $("#beginningDate").focus();
114     return;
115 } else if (beginningDate < now) {
116     Swal.fire({
117         title: "Création de l'événement",
118         text: "La date choisis est déjà passé",
119         icon: "error",
120         timer: 1500,
121         showConfirmButton: false
122     });
123     return;
124 } else {
125
126     formdata.append("beginningDate", beginningDate.toLocaleDateString());
127     formdata.append("beginningTime", beginningTime)
128 }
129
130 if (endDate === null) {
131     $("#endDate").focus();
132     return;
133 } else if (endDate < beginningDate) {
134     Swal.fire({
135         title: "Création de l'événement",
136         text: "La date choisis est avant le début de l'événement",
137         icon: "error",
138         timer: 1500,
139         showConfirmButton: false
140     });
141     return;
142 } else {
143     formdata.append("endDate", endDate.toLocaleDateString());
144     formdata.append("endTime", endTime);
145 }
146
147 if (typeof img !== 'undefined') {
148     formdata.append("img", img);
149 }
150
151 if (typeof type !== 'undefined') {
152     if (type == 'public') {
153         formdata.append("type", 0);
154     } else {
155         formdata.append("type", 1);
156         formdata.append('guestlist', await GetGuestList());
157     }
158 } else {
159     Swal.fire({
160         title: "Création de l'événement",
161         text: "Vous devez choisir si l'événement est public ou privé",
162         icon: "error",
```

```
163     timer: 1500,
164     showConfirmButton: false
165 });
166 return;
167 }
168 // #endregion
169
170 $.ajax({
171     type: "post",
172     url: "../App/php/createEvent.php",
173     // pour l'upload de fichier
174     contentType: false,
175     processData: false,
176     data: formdata,
177     dataType: "json",
178     success: (response) => {
179         switch (response.ReturnCode) {
180             case 0:
181                 Swal.fire({
182                     title: "Création de l'événement",
183                     text: response.Success,
184                     icon: "success",
185                     timer: 1300,
186                     showConfirmButton: false,
187                 }).then(() => {
188                     EventReminder();
189                     window.location.href = './index.php'
190                 });
191                 break;
192             case 1:
193             case 2:
194             case 3:
195             case 4:
196                 Swal.fire({
197                     title: "Création de l'événement",
198                     text: response.Error,
199                     icon: "error",
200                     timer: 1500,
201                     showConfirmButton: false
202                 });
203                 break;
204             }
205         },
206         error: (error) => {
207             console.error(error);
208         }
209     });
210 }
211
212 /**
213  * @author Hoarau Nicolas
214  * @date 27.05.20
215  * @brief Fonction qui récupère les tous les utilisateur à part le créateur de
216  * l'événement
217  * @return le tableau de guests
```

```
217 * @version 1.0
218 */
219 function GetGuestList() {
220     if (event)
221         event.preventDefault();
222
223     let guests = [];
224
225     // Récupère les invités
226     return fetch("../App/php/getGuestList.php", {
227         method: 'POST',
228         headers: {
229             'Accept': 'application/json',
230             'Content-Type': 'application/json'
231         }
232     }).then(response => response.json())
233     .then(async data => {
234         let html = `<div class="menu"><ul>`;
235
236         $.each(data, (key, user) => {
237             html += `<li>
238                 <label>
239                     <input type="checkbox" value="${user.idUser}">
240                     <span class="icon"></span>
241                     <span class="list">${user.pseudo}</span>
242                 </label>
243             </li>`;
244         });
245
246         html += `</ul></div>`;
247
248         // Attend la validation de la modal par l'utilisateur et récupère les invités
249         // coché
250         guests = await Swal.fire({
251             title: "Liste d'invités",
252             html: html,
253             showCancelButton: true,
254             cancelButtonText: 'Annuler',
255             confirmButtonText: 'Valider'
256         }).then((result) => {
257             let modalGuest = [];
258             // Récupère les case cochées de la modal
259             let pathToLi =
260                 Swal.getContent().childNodes[0].childNodes[0].children[0].children;
261
262             // Pour chaque <li>
263             for (let i = 0; i < pathToLi.length; i++) {
264                 const checkbox = pathToLi[i].children[0].children[0];
265
266                 if (checkbox.checked == true) {
267                     modalGuest.push(checkbox.value)
268                 }
269             }
270             return modalGuest;
271         });
272     });
273 }
```

```
270     return guests;
271 });
272 }
273
274 /**
275  * @author Hoarau Nicolas
276  * @date 28.05.20
277  * @brief Fonction qui récupère les événement publique qui ne sont pas encore passé
278  * @version 1.0
279  */
280 function GetPublicEvents() {
281     $.ajax({
282         type: "post",
283         url: "../App/php/getPublicEvents.php",
284         dataType: "json",
285         success: (response) => {
286             ShowPublicEvents(response);
287         },
288         error: (error) => {
289             console.error(error);
290         }
291     });
292 }
293
294 /**
295  * @author Hoarau Nicolas
296  * @date 28.05.20
297  * @brief Fonction qui récupère les événement publique qui ne sont pas encore passé,
298  * ainsi que les événements auxquels l'utilisateur participe et est invités
299  * @version 1.0
300  */
301 function GetManageEvents() {
302     $.ajax({
303         type: "post",
304         url: "../App/php/getManageEvents.php",
305         dataType: "json",
306         success: (response) => {
307             ShowManageEvents(response);
308         },
309         error: (error) => {
310             console.error(error);
311         }
312     });
313 }
314
315 /**
316  * @author Hoarau Nicolas
317  * @date 28.05.20
318  * @brief Fonction qui affiche les événements reçus
319  * @param {array} events
320  * @version 1.0
321  */
322 function ShowPublicEvents(data) {
323     console.log(data);
324 }
```

```
324 let html = `

7 sur 19



09.06.2020 à 16:20


```



```
376     html += `

Vous n'avez pas d'événements dans cette
catégorie</p></div>`;
377
378     for (let j = 0; j < eventZone.length; j++) {
379         const event = eventZone[j];
380
381         html += `

8 sur 19



09.06.2020 à 16:20


```

```
424     dataType: "json",
425     success: (response) => {
426         switch (response.ReturnCode) {
427             case 0:
428                 Toast.fire({
429                     icon: 'success',
430                     title: response.Success
431                 }).then(() => {
432                     if (pageName == "index.php" || pageName == "") {
433                         GetPublicEvents()
434                     } else if (pageName == "manageEvents.php") {
435                         GetManageEvents()
436                     } else {
437                         GetEventData(idEvent);
438                     }
439                 });
440                 break;
441             case 1:
442                 Toast.fire({
443                     icon: 'error',
444                     title: response.Error
445                 });
446                 break;
447         }
448     },
449     error: (error) => {
450         console.error(error);
451     }
452 });
453 }
454
455 /**
456  * @author Hoarau Nicolas
457  * @date 02.06.20
458  * @brief Fonction qui récupère les informations de l'id d'événement reçus
459  * @param {int} idEvent
460  */
461 function GetEventData(idEvent) {
462     $.ajax({
463         type: "post",
464         url: "../App/php/getEventData.php",
465         data: { idEvent: idEvent },
466         dataType: "json",
467         success: (response) => {
468             switch (response.ReturnCode) {
469                 case 0:
470                     ShowEventData(response.Data);
471                     break;
472                 case 1:
473                     window.location.href = './index.php'
474                     break;
475                 case 404:
476                     window.location.href = './404.php'
477                     break;
478             }
479         }
480     });
481 }
```

```
479     },
480     error: (error) => {
481         console.error(error);
482     }
483 });
484 }
485
486 /**
487  * @author Hoarau Nicolas
488  * @date 02.06.20
489  * @brief Fonction qui affiche les données d'un événement choisis
490  * @param {array} data
491  */
492 function ShowEventData(data) {
493     document.title = data.nom;
494
495     let html = `<div class="event-data">
496     
497     <h2 class="title">${data.nom}</h2>
498     <div class="div place">${data.prive == 1 ? 'Privé' : 'Publique'}</div>
499     <div class="div date">De ${data.dateDebut} à ${data.dateFin}<br> Par ${data.pseudo}
500     </div>
501     <div class="div">${data.descriptif}</div>
502     <div class="div place">${data.lieu}</div>
503     <div class="div">${data.nbGuest} sur ${data.nbMaxParticipant} participants <a
504     id="showGuests" onclick="GetEventGuestList(${data.idEvenement})">Voir les
505     participants</a></div>`;
506
507     if (data.isLogged === true) {
508         if (data.nickname === data.pseudo) {
509             if (data.prive == 1)
510                 html += `<div class="div"><a id="showInvited"
511                 onclick="GetInvited(${data.idEvenement})"> Voir les invités</a> <a id="showInvited"
512                 onclick="GetEditGuestList(${data.idEvenement})">Modifier la liste des invités</a>
513                 </div>`;
514
515             html += `<button class="btn btn-edit" onclick="window.location.href =
516             './editEvent.php?idEvent=${data.idEvenement}'">Modifier</button>
517             <button class="btn btn-delete" onclick="DeleteEvent(${data.idEvenement},
518             '${data.nom}', '${data.prive}')">Supprimer</button>`;
519         } else {
520             if (data.participating == 1) {
521                 html += `<button class="btn" onclick="Participate(${data.idEvenement},
522                 '${data.nom}', '${data.dateDebut}', '${data.dateFin}')">Se désinscrire</button>`;
523             } else {
524                 if (data.nbGuest < data.nbMaxParticipant) {
525                     html += `<button class="btn" onclick="Participate(${data.idEvenement},
526                     '${data.nom}', '${data.dateDebut}', '${data.dateFin}')">S'inscrire</button>`;
527                 }
528             }
529         }
530     } else {
531         Toast.fire({
532             icon: 'info',
533             title: "Vous ne pouvez pas participer à l'événement sans être connecté"
```

```
524     });
525   }
526
527   html += `</div>`;
528
529   $("#eventData").html(html);
530 }
531
532 /**
533  * @author Hoarau Nicolas
534  * @date 02.06.20
535  * @brief Fonction qui récupère les les participants de l'événement voulus
536  * @param {int} idEvent
537  */
538 function GetEventGuestList(idEvent) {
539   $.ajax({
540     type: "post",
541     url: "../App/php/getEventGuestList.php",
542     data: { idEvent: idEvent },
543     dataType: "json",
544     success: (response) => {
545       ShowEventGuestList(response.Data);
546     },
547     error: (error) => {
548       console.error(error);
549     }
550   });
551 };
552
553 /**
554  * @author Hoarau Nicolas
555  * @date 02.06.20
556  * @brief Fonction qui affiche participants d'un événement
557  * @param {array} guestList
558  */
559 function ShowEventGuestList(guestlist) {
560   let html = `<div class="menu">
561   <ul>`;
562
563   for (let i = 0; i < guestlist.length; i++) {
564     const guest = guestlist[i];
565     html += `<li>
566     <label>
567       <span class="list">${guest.pseudo}</span>
568     </label>
569   </li>`;
570   }
571
572   html += `</ul></div>`;
573
574   Swal.fire({
575     title: "Liste des participants",
576     html: html,
577     confirmButtonText: 'Valider'
578   });
```

```
579 }
580
581 /**
582  * @author Hoarau Nicolas
583  * @date 03.06.20
584  * @brief Fonction qui envoie un mail de rappel pour les événements
585  */
586 function EventReminder() {
587     $.ajax({
588         type: "post",
589         url: "../App/php/eventReminder.php",
590         dataType: "json",
591         success: (response) => {
592             console.log('email correctly send');
593         },
594         error: (error) => {
595             console.error(error);
596         }
597     });
598 }
599
600 /**
601  * @author Hoarau Nicolas
602  * @date 03.06.20
603  * @brief Fonction qui envoie les données pour supprimer un événement
604  * @param {int} idEvent
605  */
606 function DeleteEvent(idEvent, eventName, private) {
607     console.log(private);
608
609     $.ajax({
610         type: "post",
611         url: "../App/php/isEventDeletable.php",
612         data: {
613             idEvent: idEvent,
614             eventName: eventName
615         },
616         dataType: "json",
617         success: (response) => {
618             switch (response.ReturnCode) {
619                 case 0:
620                     Swal.fire({
621                         title: "Suppression de l'événement",
622                         text: response.Success,
623                         icon: "question",
624                         showConfirmButton: true,
625                         confirmButtonText: 'Valider',
626                         showCancelButton: true,
627                         cancelButtonText: 'Annuler',
628                         reverseButtons: true
629                     }).then((result) => {
630                         if (result.value) {
631                             $.ajax({
632                                 type: "post",
633                                 url: "../App/php/deleteEvent.php",
```

```
634         data: {
635             idEvent: idEvent,
636             eventName: eventName,
637             private: private
638         },
639         dataType: "json",
640         success: (data) => {
641             switch (data.ReturnCode) {
642                 case 0:
643                     Swal.fire({
644                         title: "Suppression de l'événement",
645                         text: response.Success,
646                         icon: "success",
647                         timer: 1300,
648                         showConfirmButton: false,
649                     }).then(() => {
650                         window.location.href = './index.php'
651                     });
652                     break;
653                 case 1:
654                     Swal.fire({
655                         title: "Suppression de l'événement",
656                         text: response.Error,
657                         icon: "error",
658                         timer: 1500,
659                         showConfirmButton: false,
660                     });
661                     break;
662             }
663         },
664         error: (error) => {
665             console.error(error);
666         }
667     });
668 }
669 });
670 break;
671 case 1:
672     Swal.fire({
673         title: "Suppression de l'événement",
674         text: response.Error,
675         icon: "error",
676         timer: 1500,
677         showConfirmButton: false
678     });
679     break;
680 }
681 },
682 error: (error) => {
683     console.error(error);
684 }
685 });
686 }
687
688 /**
```

```
689 * @author Hoarau Nicolas
690 * @date 04.06.20
691 * @brief Fonction qui envoie les données pour modifier un événement
692 * @param {*} event
693 * @param {int} idEvent
694 */
695 function EditEvent(event, idEvent) {
696     if (event)
697         event.preventDefault();
698
699     let formdata = new FormData();
700     let eventName = $("#eventName").val();
701     let place = $("#place").val();
702     let eventDescription = $("#eventDescription").val();
703     let nbMaxGuest = $("#nbMaxGuest").val();
704     let img = document.getElementById("img").files[0];
705     let beginningDate = new Date($("#beginningDate").val());
706     let endDate = new Date($("#endDate").val());
707     let beginningTime = $("#beginningTime").val();
708     let endTime = $("#endTime").val();
709     let now = new Date();
710
711     beginningDate.setHours(beginningTime.split(':')[0], beginningTime.split(':')[1], 0,
712 0);
713     endDate.setHours(endTime.split(':')[0], endTime.split(':')[1], 0, 0);
714     now.setHours(now.getHours(), now.getMinutes(), 0, 0);
715     formdata.append('idEvent', idEvent);
716     //endregion
717     //region Conditions
718     if (eventName.length == 0) {
719         $("#eventName").focus();
720         return;
721     } else {
722         formdata.append("eventName", eventName);
723     }
724
725     if (eventDescription.length == 0) {
726         $("#eventDescription").focus();
727         return;
728     } else {
729         formdata.append("eventDescription", eventDescription);
730     }
731
732     if (place.length == 0) {
733         $("#place").focus();
734         return;
735     } else {
736         formdata.append("place", place);
737     }
738
739     if (nbMaxGuest == 0 || nbMaxGuest == "") {
740         $("#nbMaxGuest").focus();
741         return;
742     } else {
```

```
743     formdata.append("nbMaxGuest", nbMaxGuest);
744 }
745
746 if (beginningDate === null) {
747     $("#beginningDate").focus();
748     return;
749 } else if (beginningDate < now) {
750     Swal.fire({
751         title: "Création de l'événement",
752         text: "La date choisis est déjà passé",
753         icon: "error",
754         timer: 1500,
755         showConfirmButton: false
756     });
757     return;
758 } else {
759
760     formdata.append("beginningDate", beginningDate.toLocaleDateString());
761     formdata.append("beginningTime", beginningTime)
762 }
763
764 if (endDate === null) {
765     $("#endDate").focus();
766     return;
767 } else if (endDate < beginningDate) {
768     Swal.fire({
769         title: "Création de l'événement",
770         text: "La date choisis est avant le début de l'événement",
771         icon: "error",
772         timer: 1500,
773         showConfirmButton: false
774     });
775     return;
776 } else {
777     formdata.append("endDate", endDate.toLocaleDateString());
778     formdata.append("endTime", endTime);
779 }
780
781 if (typeof img !== 'undefined') {
782     formdata.append("img", img);
783 }
784 // #endregion
785
786 $.ajax({
787     type: "post",
788     url: "../App/php/editEvent.php",
789     // pour l'upload de fichier
790     contentType: false,
791     processData: false,
792     data: formdata,
793     dataType: "json",
794     success: (response) => {
795         switch (response.ReturnCode) {
796             case 0:
797                 Swal.fire({
```



```
798         title: "Création de l'événement",
799         text: response.Success,
800         icon: "success",
801         timer: 1300,
802         showConfirmButton: false,
803     }).then(() => {
804         window.location.href = './index.php'
805     });
806     break;
807 case 1:
808 case 2:
809 case 3:
810 case 4:
811     Swal.fire({
812         title: "Création de l'événement",
813         text: response.Error,
814         icon: "error",
815         timer: 1500,
816         showConfirmButton: false
817     });
818     break;
819     }
820 },
821 error: (error) => {
822     console.error(error);
823 }
824 });
825 }
826
827 /**
828  * @author Hoarau Nicolas
829  * @date 05.06.20
830  * @brief Fonction qui récupère les invités à un événement
831  * @param {int} idEvent
832  */
833 function GetInvited(idEvent) {
834     $.ajax({
835         type: "post",
836         url: "../App/php/getInvited.php",
837         data: { idEvent: idEvent },
838         dataType: "json",
839         success: (response) => {
840             switch (response.ReturnCode) {
841                 case 0:
842                     ShowEventInvitedList(response.Data);
843                     break;
844                 case 1:
845                     Swal.fire({
846                         title: "Liste des invités",
847                         text: response.Error,
848                         icon: "error",
849                         timer: 1500,
850                         showConfirmButton: false
851                     });
852                     break;
```

```
853     }
854 },
855 error: (error) => {
856     console.error(error);
857 }
858 });
859 }
860
861 /**
862  * @author Hoarau Nicolas
863  * @date 05.06.20
864  * @brief Fonction qui affiche les invités à événement
865  * @param {array} invitedList
866  */
867 function ShowEventInvitedList(invitedList) {
868     let html = `<div class="menu"><ul>`;
869
870     for (let i = 0; i < invitedList.length; i++) {
871         const invited = invitedList[i];
872         html += `<li>
873             <label>
874                 <span class="list">${invited.pseudo}</span>
875             </label>
876         </li>`;
877     }
878
879     html += `</ul></div>`;
880
881     Swal.fire({
882         title: "Liste des invités",
883         html: html,
884         confirmButtonText: 'Valider'
885     });
886 }
887
888 /**
889  * @author Hoarau Nicolas
890  * @date 05.06.20
891  * @brief Fonction qui récupère les utilisateurs en disant s'il participe ou non
892  * l'événement donnée
893  * @param {int} idEvent
894  */
895 function GetEditGuestList(idEvent) {
896     $.ajax({
897         type: "post",
898         url: "../App/php/getEditGuestList.php",
899         data: { idEvent: idEvent },
900         dataType: "json",
901         success: (response) => {
902             switch (response.ReturnCode) {
903                 case 0:
904                     ShowEditGuestList(response.Data, idEvent);
905                     break;
906                 case 1:
907                     Swal.fire({
```

```
907         title: "Liste des invités",
908         text: response.Error,
909         icon: "error",
910         timer: 1500,
911         showConfirmButton: false
912     });
913     break;
914 }
915 },
916 error: (error) => {
917     console.error(error);
918 }
919 });
920 }
921
922 /**
923  * @author Hoarau Nicolas
924  * @date 05.06.20
925  * @brief Fonction qui affiche la modal pour la modification de la liste des invités et
926  *        envoie les données de modification
927  * @param {array} data
928  * @param {int} idEvent
929  */
929 function ShowEditGuestList(data, idEvent) {
930     let html = `<div class="menu"><ul>`;
931
932     $.each(data, (key, user) => {
933         html += `<li>
934             <label>
935                 <input type="checkbox" value="${user.idUser}" ${user.invited == 1 ? 'checked' :
936                 ''}>
937                 <span class="icon"></span>
938                 <span class="list">${user.pseudo}</span>
939             </label>
940         </li>`;
941     });
942     html += `</ul></div>`;
943
944     Swal.fire({
945         title: "Liste d'invités",
946         html: html,
947         showCancelButton: true,
948         cancelButtonText: 'Annuler',
949         confirmButtonText: 'Valider'
950     }).then((result) => {
951         if (result.value) {
952             let editGuestlist = [];
953             // Récupère les case cochées de la modal
954             let pathToLi =
955                 Swal.getContent().childNodes[0].childNodes[0].children[0].children;
956
957             // Pour chaque <li>
958             for (let i = 0; i < pathToLi.length; i++) {
959                 const checkbox = pathToLi[i].children[0].children[0];
```

```
959
960 // regarde els différences avec la liste de base
961 if (data[i].invited != checkbox.checked) {
962     editGuestlist.push([checkbox.value, checkbox.checked])
963 }
964 }
965
966 if (editGuestlist.length > 0) {
967     $.ajax({
968         type: "post",
969         url: "../App/php/editGuestList.php",
970         data: {
971             editGuetsList: editGuestlist,
972             idEvent: idEvent
973         },
974         dataType: "json",
975         success: (response) => {
976             switch (response.ReturnCode) {
977                 case 0:
978                     Swal.fire({
979                         title: "Liste d'invités",
980                         text: response.Success,
981                         icon: "success",
982                         timer: 1300,
983                         showConfirmButton: false
984                     });
985                     break;
986                 case 1:
987                     Swal.fire({
988                         title: "Liste d'invités",
989                         text: response.Error,
990                         icon: "error",
991                         timer: 1300,
992                         showConfirmButton: false
993                     });
994                     break;
995             }
996         },
997         error: (error) => {
998             console.error(error);
999         }
1000     });
1001 }
1002 }
1003 });
1004 ,
```

```
1 <nav>
2   <div class="hamburger">
3     <div class="line"></div>
4     <div class="line"></div>
5     <div class="line"></div>
6   </div>
7   <a href="./index.php"></a>
8   <ul class="nav-links">
9     <li><a href="./index.php">Accueil</a></li>
10    <?php if (IsLogged() == false) : ?>
11      <li><a href="./login.php">Connexion</a></li>
12      <li><a href="./register.php">Inscription</a></li>
13    <?php else : ?>
14      <li><a href="./profile.php">Mon profile</a></li>
15      <li><a href="./manageEvents.php">Gérer mes événements</a></li>
16      <li><a href="./createEvent.php">Créer un événement</a></li>
17      <li><a href="./App/php/logout.php">Déconnexion</a></li>
18    <?php endif; ?>
19      <li><a href="./about.php">But du site</a></li>
20    </ul>
21 </nav>
22
```

```
1 <!--    IMPORT JQUERY    -->
2 <script src="https://code.jquery.com/jquery.min.js"></script>
3 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js">
  </script>
4
5 <!--    IMPORT SWEETALERTS    -->
6 <script src="./js/sweetalert2.min.js"></script>
7
8 <!-- IMPORT JS FOR STYLE -->
9 <script type="text/javascript" src="./js/main.js"></script>
- ^
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : const.inc.php
6  * Date        : 27.05.2020
7  * Description  : fichier qui contient les constantes du projet
8  * Version     : 1.0
9  */
10
11 define('UPLOAD_PATH', dirname(dirname(__DIR__)) . '/views/assets/upload/');
12 define('DEFAULT_IMG', './assets/img/default.svg');
```

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

body {
  font-family: "Poppins", sans-serif;
  overflow-y: auto;
}

/* Chrome, Safari, Edge, Opera */
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}

/* Firefox */
input[type=number] {
  -moz-appearance: textfield;
}

input[type="file"] {
  display: none;
}

/***** LOGIN *****/
.wave {
  position: fixed;
  bottom: 0;
  left: 0;
  height: 100%;
  z-index: -1;
}

.container {
  width: 100vw;
  height: 85vh;
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-gap: 7rem;
  padding: 0 2rem;
}

.img {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}

.content {
  display: flex;
  justify-content: flex-start;
```



```
55 | align-items: center;
56 | text-align: center;
57 | }
58 |
59 | .img img {
60 |   width: 500px;
61 | }
62 |
63 | form {
64 |   width: 360px;
65 | }
66 |
67 | .content img {
68 |   height: 100px;
69 | }
70 |
71 | .content h2 {
72 |   margin: 5px 0;
73 |   color: #333;
74 |   text-transform: uppercase;
75 |   font-size: 2rem;
76 | }
77 |
78 | .content .input-div {
79 |   position: relative;
80 |   display: grid;
81 |   grid-template-columns: 7% 93%;
82 |   margin: 15px 0;
83 |   padding: 5px 0;
84 |   border-bottom: 2px solid #d9d9d9;
85 | }
86 |
87 | .content .input-div.one {
88 |   margin-top: 0;
89 | }
90 |
91 | .i {
92 |   color: #d9d9d9;
93 |   display: flex;
94 |   justify-content: center;
95 |   align-items: center;
96 | }
97 |
98 | .i i {
99 |   transition: 0.3s;
100 | }
101 |
102 | .input-div > div {
103 |   position: relative;
104 |   height: 38px;
105 | }
106 |
107 | .input-div > div > h5 {
108 |   position: absolute;
109 |   left: 10px;
```

```
110 top: 50%;
111 transform: translateY(-50%);
112 color: #999;
113 font-size: 15px;
114 transition: 0.3s;
115 }
116
117 .input-div:before,
118 .input-div:after {
119   content: "";
120   position: absolute;
121   bottom: -2px;
122   width: 0%;
123   height: 2px;
124   background-color: #https://www.wappalyzer.com/technologies/font-scripts;
125   transition: 0.4s;
126 }
127
128 .input-div:before {
129   right: 50%;
130 }
131
132 .input-div:after {
133   left: 50%;
134 }
135
136 .input-div.focus:before,
137 .input-div.focus:after {
138   width: 50%;
139 }
140
141 .input-div.focus > div > h5 {
142   top: -5px;
143   font-size: 15px;
144 }
145
146 .input-div.focus > .i > i {
147   color: #38d39f;
148 }
149
150 .input-div > div > input {
151   position: absolute;
152   left: 0;
153   top: 0;
154   width: 100%;
155   height: 80%;
156   border: none;
157   outline: none;
158   background: none;
159   padding: 0.5rem 0.7rem;
160   font-size: 1.2rem;
161   color: #555;
162   font-family: "poppins", sans-serif;
163 }
164
```

```
165 .input-div.pass {
166   margin-bottom: 4px;
167 }
168
169 a {
170   display: block;
171   text-align: right;
172   text-decoration: none;
173   color: #999;
174   font-size: 0.9rem;
175   transition: 0.3s;
176 }
177
178 a:hover {
179   color: #38d39f;
180 }
181
182 .btn {
183   display: block;
184   width: 100%;
185   height: 45px;
186   border-radius: 25px;
187   outline: none;
188   border: none;
189   background-image: linear-gradient(to right, #32be8f, #38d39f, #32be8f);
190   background-size: 200%;
191   font-size: 1.2rem;
192   color: #fff;
193   font-family: "Poppins", sans-serif;
194   text-transform: uppercase;
195   margin: 1rem 0;
196   cursor: pointer;
197   transition: 0.5s;
198 }
199
200 .btn:hover {
201   background-position: right;
202 }
203
204 ul {
205   text-decoration: none;
206   display: inline;
207 }
208
209 /***** NAVBAR *****/
210 nav {
211   height: 8vh;
212   background: transparent;
213   width: 100%;
214 }
215
216 .nav-links {
217   display: flex;
218   list-style: none;
219   width: 50%;
```

```
220 height: 100%;
221 justify-content: space-around;
222 align-items: center;
223 margin-left: auto;
224 }
225
226 /***** CREATE EVENTS *****/
227 .radio {
228     font-size: 18px;
229     text-transform: capitalize;
230     display: inline-block;
231     vertical-align: middle;
232     position: relative;
233     padding-left: 30px;
234     cursor: pointer;
235 }
236
237 .radio + .radio {
238     margin-left: 30px;
239 }
240
241 .radio input[type="radio"] {
242     display: none;
243 }
244
245 .radio span {
246     height: 18px;
247     width: 18px;
248     border-radius: 50%;
249     border: 3px solid #38d39f;
250     display: block;
251     position: absolute;
252     left: 0;
253     top: 7px;
254 }
255
256 .radio span:after {
257     content: "";
258     height: 8px;
259     width: 8px;
260     background-color: #38d39f;
261     display: block;
262     position: absolute;
263     left: 50%;
264     top: 50%;
265     transform: translate(-50%, -50%) scale(0);
266     border-radius: 50%;
267     transition: 300ms ease-in-out 0s;
268 }
269
270 .radio input[type="radio"]:checked ~ span:after {
271     transform: translate(-50%, -50%) scale(1);
272 }
273
274 /***** GUEST LIST *****/
```

```
275 ul li {
276   list-style: none;
277   padding: 5px 0;
278   font-size: 16px;
279 }
280
281 ul li input[type="checkbox"] {
282   display: none;
283 }
284
285 ul li span.list {
286   position: relative;
287   display: inline-block;
288   overflow: hidden;
289   padding: 0 5px;
290   transition: 0.5s;
291 }
292
293 ul li input[type="checkbox"]:checked ~ span.list {
294   color: #999;
295   transition-delay: 0s;
296 }
297
298 ul li span.icon {
299   position: relative;
300   top: -3px;
301   width: 18px;
302   height: 18px;
303   box-sizing: border-box;
304   border: 1px solid #262626;
305   display: inline-block;
306   margin-right: 5px;
307   overflow: hidden;
308 }
309
310 ul li span.icon:before {
311   content: "✓";
312   position: absolute;
313   color: #38d39f;
314   top: -5px;
315   left: 2px;
316   transform: translateY(-100%);
317 }
318
319 ul li input[type="checkbox"]:checked ~ span.icon:before {
320   transform: translateY(0);
321 }
322
323 /***** CARD *****/
324 .card-container {
325   width: 100vw;
326   height: 55vh;
327   display: flex;
328   align-items: center;
329   justify-content: center;
```

```
330   overflow: hidden;
331 }
332
333 .card {
334   display: grid;
335   grid-template-columns: 250px;
336   grid-template-rows: 150px 210px 80px;
337   grid-template-areas: "image" "text" "stats";
338
339   border-radius: 18px;
340   background: white;
341   box-shadow: 5px 5px 15px rgba(0,0,0,0.9);
342   text-align: center;
343
344   margin: 30px;
345   transition: 0.5s ease;
346   cursor: pointer;
347 }
348
349 .card-image {
350   grid-area: image;
351   border-top-left-radius: 15px;
352   border-top-right-radius: 15px;
353   background-size: cover;
354 }
355
356 .card-image img {
357   width: 180px;
358   height: 180px;
359 }
360
361 .card-text {
362   grid-area: text;
363   margin: 40px 25px 25px 25px;
364 }
365
366 .date {
367   color: #38d39f;
368   font-size: 14px;
369 }
370
371 .card-text p {
372   color: grey;
373   font-size: 13px;
374   font-weight: 300;
375 }
376
377 .card-text h2 {
378   margin-top: 0px;
379   font-size: 18px;
380 }
381
382 .card button {
383   grid-area: stats;
384   grid-template-columns: 1fr 1fr 1fr;
385   grid-template-rows: 1fr;
```

```
385 border-top-left-radius: 0;
386 border-top-right-radius: 0;
387 border-bottom-left-radius: 15px;
388 border-bottom-right-radius: 15px;
389 height: 80%;
390 background: #38d39f;
391 padding: 10px;
392 cursor: pointer;
393 }
394
395 .card:hover {
396   transform: scale(1.15);
397 }
398
399 #events {
400   overflow-y: auto;
401 }
402
403
404 .event-data {
405   width: 360px;
406 }
407
408 .event-data .place{
409   color: grey;
410 }
411
412 #showGuests {
413   float: right;
414   padding-top: 3px;
415   cursor: pointer;
416 }
417
418 #showInvited {
419   text-align: center;
420   cursor: pointer;
421 }
422
423 .btn-delete {
424   background-image: linear-gradient(to right, #d33838, #cf3636, #d33838);
425 }
426
427 .btn-edit {
428   background-image: linear-gradient(to right, grey, #999, grey);
429 }
430
431 .logo {
432   float: left;
433   height: 74px;
434   padding-left: 10px;
435 }
436
437 .container-about {
438   width: 750px;
439 }
```

```
440
441 .about {
442   font-weight: lighter;
443   color: #474b52;
444 }
445
```



```
/* ***** LOGIN ***** */
@media screen and (max-width: 1050px) {
  .container {
    grid-gap: 5rem;
  }
}

@media screen and (max-width: 1000px) {
  form {
    width: 290px;
  }

  .content {
    font-size: 2.4rem;
    margin: 8px 0;
  }

  .img img {
    width: 400px;
  }
}

@media screen and (max-width: 900px) {
  nav {
    height: 8vh;
  }

  .container {
    grid-template-columns: 1fr;
  }

  .img {
    display: none;
  }

  .wave {
    display: none;
  }

  .content {
    justify-content: center;
  }

  .one {
    margin-bottom: 10px;
  }

  .input-div {
    height: 42px;
  }

  .i {
    font-size: 1.5rem;
  }
}
```

```
55 |
56 | #formRegister {
57 |     margin-top: 0;
58 | }
59 |
60 |
61 | /***** NAVBAR *****/
62 | @media screen and (max-width: 768px) {
63 |     .line {
64 |         width: 30px;
65 |         height: 3px;
66 |         background: #999;
67 |         margin: 5px;
68 |     }
69 |
70 |     nav {
71 |         position: relative;
72 |     }
73 |
74 |     .hamburger {
75 |         position: absolute;
76 |         cursor: pointer;
77 |         right: 5%;
78 |         top: 50%;
79 |         transform: translate(-5%, -50%);
80 |         z-index: 2;
81 |     }
82 |
83 |     .nav-links {
84 |         position: fixed;
85 |         background: #38d39f;
86 |         height: 100vh;
87 |         width: 100%;
88 |         flex-direction: column;
89 |         clip-path: circle(100px at 90% -15%);
90 |         -webkit-clip-path: circle(100px at 90% -15%);
91 |         transition: all 0.7s ease-out;
92 |         pointer-events: none;
93 |     }
94 |
95 |     .nav-links.open {
96 |         clip-path: circle(110px at 90% -10%);
97 |         -webkit-clip-path: circle(110px at 90% -10%);
98 |         pointer-events: all;
99 |     }
100 |
101 |     .nav-links li {
102 |         opacity: 0;
103 |     }
104 |
105 |     .nav-links li a {
106 |         font-size: 25px;
107 |         color: #fff;
108 |     }
109 | }
```

```
110 .nav-links li:nth-child(1) {
111     transition: all 0.5s ease 0.2s;
112 }
113
114 .nav-links li:nth-child(2) {
115     transition: all 0.5s ease 0.4s;
116 }
117
118 .nav-links li:nth-child(3) {
119     transition: all 0.5s ease 0.6s;
120 }
121
122 li.fade {
123     opacity: 1;
124 }
125 }
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : config.php
6  * Date        : 26.05.2020
7  * Description  : Fichier qui contient les constantes de connexion à la base de donnée:
8  * Version     : 1.0
9  */
10
11 // Le Type de la base de donnée
12 define('DB_DBTYPE', "mysql");
13
14 // Connexion local
15 define('DB_HOST', 'localhost');
16 define('DB_DBNAME', 'wego');
17 define('DB_USER', 'root');
18 define('DB_PASS', 'Super');
19 define('DB_PORT', 3306);
20
21 // PHPMailer
22 define('MAIL_USERNAME', 'nicohoarau74@gmail.com');
23 define('MAIL_PASSWORD', "4<Rh#6F+}kPw(kRbGHGU");
24 define('MAIL_HOST', "smtp.gmail.com");
25 define('MAIL_PORT', 587);
26
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : willParticipate.php
 * Date        : 28.05.2020
 * Description  : Fichier qui met à jour si on est inscrit ou non à un événement
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
require_once __DIR__ . './backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idUser = $_SESSION['loggedIn']['idUser'];
$email = $_SESSION['loggedIn']['email'];
$idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
$eventName = filter_input(INPUT_POST, 'eventName', FILTER_SANITIZE_STRING);
$beginDate = filter_input(INPUT_POST, 'beginDate');
$endDate = filter_input(INPUT_POST, 'endDate');

$mailSubject = "WEGO: Validation de participation";

$mailMessage = <<<EX
Vous participez à l'événement "{$eventName}" prévu de {$beginDate} à {$endDate}.
EX;

$queryWillParticipate = <<<EX
INSERT INTO inscriptions (idUser, idEvenement)
VALUES (:idUser, :idEvent);
EX;

$queryWontParticipate = <<<EX
DELETE FROM inscriptions
WHERE idUser = :idUser
AND idEvenement = :idEvent;
EX;

$isParticipating = IsParticipating($idEvent, $idUser);

$query = $isParticipating == false ? $queryWillParticipate : $queryWontParticipate;
$successMessage = $isParticipating == false ? "L'inscription à l'événement réussi" :
"La désinscription à l'événement à réussi";

try {
    DatabaseController::beginTransaction();

    $requestParticipation = DatabaseController::prepare($query);
    $requestParticipation->bindParam(':idUser', $idUser, PDO::PARAM_INT, 11);
    $requestParticipation->bindParam(':idEvent', $idEvent, PDO::PARAM_INT, 11);
```

```
54 $requestParticipation->execute();
55
56 if ($isParticipating == false) {
57     if (SendMail($email, $mailMessage, $mailSubject) == false) {
58         DatabaseController::rollBack();
59         echo json_encode([
60             'ReturnCode' => 1,
61             'Error' => "Une erreur est survenue lors de l'envoi du mail."
62         ]);
63         exit();
64     }
65 }
66
67 DatabaseController::commit();
68
69 echo json_encode([
70     'ReturnCode' => 0,
71     'Success' => $successMessage
72 ]);
73 exit();
74 } catch (PDOException $e) {
75     DatabaseController::rollBack();
76
77     echo json_encode([
78         'ReturnCode' => 1,
79         'Error' => "Erreur est survenue lors de l'actualisation de la participation"
80     ]);
81     exit();
82 }
83 }
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : register.php
 * Date        : 26.05.2020
 * Description  : Fichier qui permet filtrer les données reçus du call Ajax afin de
créer un utilisateur.
 * Version     : 1.0
 */

require_once __DIR__ . '/backend.php';

$lastname = filter_input(INPUT_POST, 'lastname', FILTER_SANITIZE_STRING);
$firstname = filter_input(INPUT_POST, 'firstname', FILTER_SANITIZE_STRING);
$nickname = filter_input(INPUT_POST, 'nickname', FILTER_SANITIZE_STRING);
$email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
$phoneNumber = filter_input(INPUT_POST, 'phoneNumber', FILTER_SANITIZE_STRING);
$password = filter_input(INPUT_POST, 'password', FILTER_SANITIZE_STRING);
$verifyPassword = filter_input(INPUT_POST, 'verifyPassword', FILTER_SANITIZE_STRING);

$regex = "/^(?=\w{8,})(?=[^a-z]*[a-z])(?=[^A-Z]*[A-Z])(\w*\d)\w*$/"; // Faut que le mdp
contienne au minimum 8char, une majuscule et 1chiffre

// si tous les champs obligatoire sont remplis
if (strlen($nickname) > 0 && strlen($firstname) > 0 && strlen($email) > 0 &&
strlen($password) > 0 && strlen($verifyPassword) > 0) {
    if (preg_match($regex, $password)) {
        if (IsTaken(['userEmail' => $email]) == false) {
            if (IsTaken(['userNickname' => $nickname]) == false) {
                if ($password == $verifyPassword) {
                    if (Register($nickname, $firstname, $email, $password, $lastname,
$phoneNumber)) {
                        echo json_encode([
                            'ReturnCode' => 0,
                            'Success' => 'Le compte a bien été créé.'
                        ]);
                        exit();
                    } else {
                        echo json_encode([
                            'ReturnCode' => 1,
                            'Error' => 'Erreur est survenue lors de la création du compte'
                        ]);
                        exit();
                    }
                } else {
                    echo json_encode([
                        'ReturnCode' => 2,
                        'Error' => 'Les deux mots de passe données ne sont pas les mêmes.'
                    ]);
                    exit();
                }
            } else {
                echo json_encode([
```

```
51         'ReturnCode' => 3,  
52         'Error' => 'Ce nom d\'utilisateur est déjà utilisé'  
53     ]);  
54     exit();  
55 }  
56 } else {  
57     echo json_encode([  
58         'ReturnCode' => 4,  
59         'Error' => 'Cette adresse mail est déjà utilisé'  
60     ]);  
61     exit();  
62 }  
63 } else {  
64     echo json_encode([  
65         'ReturnCode' => 5,  
66         'Error' => 'Le mot de passe doit faire au moins 8 caractères et doit contenir au  
moins une majuscule et un chiffre.'  
67     ]);  
68     exit();  
69 }  
70 }  
--
```



```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet     : WE GO
5  * Page      : logout.php
6  * Date      : 27.05.2020
7  * Description : Déconnecte l'utilisateur
8  * Version    : 1.0
9  */
10
11 if (session_status() == PHP_SESSION_NONE) {
12     session_start();
13 }
14
15 unset($_SESSION['loggedIn']);
16
17 header('Location: ../../views/index.php');
18 exit();
19
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet     : WE GO
5  * Page       : login.php
6  * Date       : 26.05.2020
7  * Description : Fichier qui permet de filtrer les données reçus du call Ajax afin de
connecter l'utilisateur
8  * Version    : 1.0
9  */
10
11 require_once __DIR__ . '/backend.php';
12
13 if (session_status() == PHP_SESSION_NONE) {
14     session_start();
15 }
16
17 // Initialisation
18 $authenticator = filter_input(INPUT_POST, "authenticator", FILTER_SANITIZE_STRING);
19 $password = filter_input(INPUT_POST, "password", FILTER_SANITIZE_STRING);
20 $userLogged = null;
21
22 if (strlen($authenticator) > 0 && strlen($password) > 0) {
23     if (strpos($authenticator, '@')) {
24         $userLogged = Login(['userEmail' => $authenticator, 'userPwd' => $password]);
25     } else {
26         $userLogged = Login(['userNickname' => $authenticator, 'userPwd' => $password]);
27     }
28
29     if ($userLogged !== false) {
30         $_SESSION['loggedIn'] = $userLogged;
31
32         echo json_encode([
33             'ReturnCode' => 0,
34             'Success' => "Connexion réussi."
35         ]);
36         exit();
37     }
38
39     echo json_encode([
40         'ReturnCode' => 1,
41         'Error' => "Pseudo/Mot de passe invalide"
42     ]);
43     exit();
44 }
--
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : isProfileDeletable.php
6  * Date        : 04.06.2020
7  * Description : Fichier qui vérifie si le profile est supprimable
8  * Version     : 1.0
9  */
10
11 require_once __DIR__ . './backend.php';
12
13 $idUser = filter_input(INPUT_POST, 'idUser', FILTER_SANITIZE_NUMBER_INT);
14 $usersEvents = GetUsersEvent($idUser);
15
16 if ($usersEvents == 0 || CountUsersGuest($usersEvents) == 0) {
17     echo json_encode([
18         'ReturnCode' => 0,
19         'Success' => 'Voulez-vous vraiment supprimer votre profile ?'
20     ]);
21     exit();
22 } else {
23     echo json_encode([
24         'ReturnCode' => 1,
25         'Error' => "Il ne doit pas avoir de participant à vos événement pour supprimer
votre compte."
26     ]);
27     exit();
28 }
29
30
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : isEventDeletable.php
6  * Date        : 03.06.2020
7  * Description  : Vérifie si on peut supprimer l'événement.
8  * Version     : 1.0
9  */
10
11 require_once __DIR__ . './backend.php';
12
13 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
14 $eventName = filter_input(INPUT_POST, 'eventName', FILTER_SANITIZE_STRING);
15
16 if (GetEventNbGuest($idEvent) == 0) {
17     echo json_encode([
18         'ReturnCode' => 0,
19         'Success' => 'Voulez-vous vraiment supprimer ' . $eventName . '?'
20     ]);
21     exit();
22 } else {
23     echo json_encode([
24         'ReturnCode' => 1,
25         'Error' => "L'événement \"" . $eventName . "\" ne peut être supprimé car il y a des
participants"
26     ]);
27     exit();
28 }
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : getUserData.php
6  * Date        : 03.06.2020
7  * Description : Fichier qui envoie les données de l'utilisateur
8  * Version     : 1.0
9  */
10
11 require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
12
13 if (session_status() == PHP_SESSION_NONE) {
14     session_start();
15 }
16
17 $idUser = $_SESSION['loggedIn']['idUser'];
18
19 $query = <<<EX
20 SELECT idUser, pseudo, prenom, nom, email, telephone
21 FROM user
22 WHERE idUser = :idUser;
23 EX;
24
25 try {
26     $requestGetUserData = DatabaseController::prepare($query);
27     $requestGetUserData->bindParam(':idUser', $idUser, PDO::PARAM_STR);
28     $requestGetUserData->execute();
29
30     $_SESSION['loggedIn'] = $requestGetUserData->fetchAll(PDO::FETCH_ASSOC)[0];
31
32     echo json_encode([
33         'ReturnCode' => 0,
34         'Data' => $_SESSION['loggedIn']
35     ]);
36 } catch (PDOException $e) {
37     echo json_encode([
38         'ReturnCode' => 1,
39         'Error' => 'Erreur est survenue lors de la modification du compte'
40     ]);
41     exit();
42 }
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : getPublicEvents.php
6  * Date        : 28.05.2020
7  * Description : Fichier qui récupère les événements publics qui ne sont pas encore
8  * Version     : 1.0
9  */
10
11 require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
12 require_once __DIR__ . './backend.php';
13
14 if (session_status() == PHP_SESSION_NONE) {
15     session_start();
16 }
17
18 $dateMoreOneHour = date('Y-m-d H:i:s', strtotime(date('Y-m-d H:i:s') . '+1hours'));
19
20 $queryGetPublicEvents = <<<EX
21 SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.urlImage, u.pseudo
22 FROM evenement AS e
23 JOIN user AS u ON e.idOrganisateur = u.idUser
24 WHERE prive = 0
25 AND TIMESTAMP(dateDebut) >= :dateTime
26 ORDER BY dateDebut DESC
27 EX;
28
29 try {
30     $requestGetPublicEvents = DatabaseController::prepare($queryGetPublicEvents);
31     $requestGetPublicEvents->bindParam(':dateTime', $dateMoreOneHour);
32     $requestGetPublicEvents->execute();
33
34     $result['events'] = $requestGetPublicEvents->fetchAll(PDO::FETCH_ASSOC);
35
36     echo json_encode($result);
37 } catch (PDOException $e) {
38     echo json_encode([
39         'ReturnCode' => 1,
40         'Error' => 'Erreur est survenue lors de la récupération des événements'
41     ]);
42     exit();
43 }
44
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : getEvents.php
 * Date        : 28.05.2020
 * Description  : Fichier qui récupère les événements pour la gestion des événements s
on est connecté sinon récupère les événements publics.
 * Version     : 1.0 : Récupère les événements pour la gestion des événements si on
est connecté
 */

require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
require_once __DIR__ . './backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$dateMoreOneHour = date('Y-m-d H:i:s', strtotime(date('Y-m-d H:i:s') . '+1hours'));
$idUser = $_SESSION['loggedIn']['idUser'];

$queryGetEventsParticipating = <<<EX
SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.urlImage, u.pseuc
FROM inscriptions AS i
JOIN evenement AS e ON i.idEvenement = e.idEvenement
JOIN user AS u ON e.idOrganisateur = u.idUser
WHERE i.idUser = :idUser
AND e.idOrganisateur <> :idUser
AND TIMESTAMP(dateDebut) >= :dateTime
EX;

$queryGetPublicEventsNotParticipating = <<<EX
SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.urlImage, u.pseuc
FROM evenement AS e
JOIN user AS u ON u.idUser = e.idOrganisateur
WHERE idEvenement NOT IN (
    SELECT idEvenement
    FROM inscriptions
    WHERE idUser = :idUser )
AND prive = 0
AND e.idOrganisateur <> :idUser;
EX;

$queryGetInvitedEvents = <<<EX
SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.urlImage, u.pseuc
FROM invites AS i
JOIN evenement AS e ON e.idEvenement = i.idEvenement
JOIN user AS u ON u.idUser = e.idOrganisateur
WHERE prive = 1
AND TIMESTAMP(dateDebut) >= :dateTime
AND e.idEvenement NOT IN (
    SELECT idEvenement
    FROM inscriptions
```

```
53 WHERE idUser = :idUser )
54 AND i.idUser = :idUser
55 ORDER BY dateDebut DESC;
56 EX;
57
58 $queryGetEventCreated = <<<EX
59 SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.urlImage, u.pseuc
60 FROM evenement AS e
61 JOIN user AS u ON u.idUser = e.idOrganisateur
62 WHERE e.idOrganisateur = :idUser
63 AND TIMESTAMP(dateDebut) >= :dateTime
64 EX;
65
66 try {
67     $requestGetPublicEventsNotParticipating =
        DatabaseController::prepare($queryGetPublicEventsNotParticipating);
68     $requestGetPublicEventsNotParticipating->bindParam(':idUser', $idUser,
        PDO::PARAM_INT, 11);
69     $requestGetPublicEventsNotParticipating->bindParam(':dateTime', $dateMoreOneHour);
70
71     $requestGetPublicEventsNotParticipating->execute();
72     $result['eventsNotParticipating'] =
        $requestGetPublicEventsNotParticipating->fetchAll(PDO::FETCH_ASSOC);
73
74     $requestGetInvitedEvents = DatabaseController::prepare($queryGetInvitedEvents);
75     $requestGetInvitedEvents->bindParam(':idUser', $idUser, PDO::PARAM_INT, 11);
76     $requestGetInvitedEvents->bindParam(':dateTime', $dateMoreOneHour);
77
78     $requestGetInvitedEvents->execute();
79     $result['invitedEvents'] = $requestGetInvitedEvents->fetchAll(PDO::FETCH_ASSOC);
80
81     $requestGetEventsParticipating =
        DatabaseController::prepare($queryGetEventsParticipating);
82     $requestGetEventsParticipating->bindParam(':idUser', $idUser, PDO::PARAM_INT, 11);
83     $requestGetEventsParticipating->bindParam(':dateTime', $dateMoreOneHour);
84
85     $requestGetEventsParticipating->execute();
86     $result['eventsParticipating'] =
        $requestGetEventsParticipating->fetchAll(PDO::FETCH_ASSOC);
87
88     $requestGetEventCreated = DatabaseController::prepare($queryGetEventCreated);
89     $requestGetEventCreated->bindParam(':idUser', $idUser, PDO::PARAM_INT, 11);
90     $requestGetEventCreated->bindParam(':dateTime', $dateMoreOneHour);
91
92     $requestGetEventCreated->execute();
93     $result['eventsCreated'] = $requestGetEventCreated->fetchAll(PDO::FETCH_ASSOC);
94
95     $result['isLogged'] = true;
96     $result['nickname'] = $_SESSION['loggedIn']['pseudo'];
97
98     echo json_encode($result);
99     exit();
100 } catch (PDOException $e) {
101     echo json_encode([
102         'ReturnCode' => 1,
```



```
103     'Error' => 'Erreur est survenue lors de la récupération des événements'  
104   ]);  
105   exit();  
106 }  
107
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : getInvited.php
6  * Date        : 05.06.2020
7  * Description : Fichier qui récupère les invités à un événements
8  * Version     : 1.0
9  */
10
11 require_once __DIR__ . '/backend.php';
12
13 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
14
15 $data = (GetInvited($idEvent) != null) ? GetInvited($idEvent) : false;
16
17 if ($data != false) {
18     echo json_encode([
19         'ReturnCode' => 0,
20         'Data' => $data
21     ]);
22     exit();
23 } else {
24     echo json_encode([
25         'ReturnCode' => 1,
26         'Error' => 'Erreur est survenue lors de la récupération des invités'
27     ]);
28     exit();
29 }
30
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet     : WE GO
5  * Page      : getGuestList.php
6  * Date      : 27.05.2020
7  * Description : Récupère tous les utilisateurs de la base de données à part le
créateur de l'événement
8  * Version    : 1.0
9  */
10
11 require_once __DIR__ . '/backend.php';
12 require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
13
14 if (session_status() == PHP_SESSION_NONE) {
15     session_start();
16 }
17
18 $nickname = $_SESSION['loggedIn']['pseudo'];
19
20 $query = <<<EX
21     SELECT idUser, pseudo FROM user WHERE pseudo <> :pseudo;
22     EX;
23
24 try {
25     $requestGetUsers = DatabaseController::prepare($query);
26     $requestGetUsers->bindParam(':pseudo', $nickname, PDO::PARAM_STR, 50);
27     $requestGetUsers->execute();
28
29     $result = $requestGetUsers->fetchAll(PDO::FETCH_ASSOC);
30
31     echo json_encode($result);
32     exit();
33 } catch (PDOException $e) {
34     echo json_encode([
35         'ReturnCode' => 1,
36         'Error' => 'Erreur est survenue lors de la récupération de la liste des invités'
37     ]);
38     exit();
39 }
40
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : getEventGuestList.php
6  * Date        : 02.06.2020
7  * Description  : Fichier qui récupère participant d'un événement données.
8  * Version     : 1.0
9  */
10
11 require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
12
13 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
14
15 $query = <<<EX
16 SELECT u.pseudo
17 FROM inscriptions AS i
18 JOIN user AS u ON i.idUser = u.idUser
19 WHERE idEvenement = :idEvent;
20 EX;
21
22 try {
23     $requestGetGuestList = DatabaseController::prepare($query);
24
25     $requestGetGuestList->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
26     $requestGetGuestList->execute();
27     $result = $requestGetGuestList->fetchAll(PDO::FETCH_ASSOC);
28
29     echo json_encode([
30         'Data' => $result
31     ]);
32     exit();
33 } catch (PDOException $e) {
34     echo json_encode([
35         'ReturnCode' => 1,
36         'Error' => 'Erreur est survenue lors de la récupération de la liste des
participants'
37     ]);
38     exit();
39
40     exit();
41 }
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : getEventData.php
 * Date        : 02.06.2020
 * Description  : Fichier qui récupère els informations d'un événement donnée
 * Version     : 1.0
 */

require_once __DIR__ . './backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
$idUser = isset($_SESSION['loggedIn']['idUser']) ? $_SESSION['loggedIn']['idUser'] :
null;
$nickname = isset($_SESSION['loggedIn']['idUser']) ? $_SESSION['loggedIn']['pseudo'] :
null;

if (EventExist($idEvent)) {
    if ($idUser == null) {
        $data = GetEventData($idEvent);
        $data['isLogged'] = false;

        echo json_encode([
            'ReturnCode' => 0,
            'Data' => $data
        ]);
        exit();
    } else {
        if (IsPrivate($idEvent)) {
            if (IsInvited($idEvent, $idUser) || IsCreator($idEvent, $idUser)) {
                $data = GetEventData($idEvent, $idUser);
                $data['isLogged'] = true;
                $data['nickname'] = $nickname;

                echo json_encode([
                    'ReturnCode' => 0,
                    'Data' => $data
                ]);
                exit();
            } else {
                echo json_encode([
                    'ReturnCode' => 1,
                ]);
            }
        } else {
            $data = GetEventData($idEvent, $idUser);
            $data['isLogged'] = true;
            $data['nickname'] = $nickname;
        }
    }
}
```

```
53     echo json_encode([
54         'ReturnCode' => 0,
55         'Data' => $data
56     ]);
57     exit();
58 }
59 }
60 } else {
61     echo json_encode([
62         'ReturnCode' => 404
63     ]);
64     exit();
65 }
--
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : getEditGuestList.php
6  * Date        : 05.06.2020
7  * Description : Fichier qui récupère les utilisateurs en disant s'il participe ou non
8                  l'événement donnée
9  * Version     : 1.0
10 */
11 require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
12
13 if (session_status() == PHP_SESSION_NONE) {
14     session_start();
15 }
16
17 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
18 $nickname = $_SESSION['loggedIn']['pseudo'];
19 $query = <<<EX
20 SELECT DISTINCT u.idUser, u.pseudo,
21     CASE WHEN i.idUser IS NOT NULL AND i.idEvenement = :idEvent THEN TRUE ELSE FALSE END
22 AS invited
23 FROM user AS u
24 LEFT JOIN invites AS i ON i.idUser = u.idUser
25 WHERE i.idEvenement = :idEvent
26 OR i.idEvenement IS NULL
27 OR i.idUser NOT IN (
28     SELECT idUser FROM invites WHERE idEvenement = :idEvent
29 )
30 AND u.pseudo <> :nickname;
31 EX;
32
33 try {
34     $requestGetInvitedAndNotInvited = DatabaseController::prepare($query);
35     $requestGetInvitedAndNotInvited->bindParam(':nickname', $nickname, PDO::PARAM_STR);
36     $requestGetInvitedAndNotInvited->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
37     $requestGetInvitedAndNotInvited->execute();
38     $result = $requestGetInvitedAndNotInvited->fetchAll(PDO::FETCH_ASSOC);
39
40     echo json_encode([
41         'ReturnCode' => 0,
42         'Data' => $result
43     ]);
44     exit();
45 } catch (PDOException $e) {
46     echo json_encode([
47         'ReturnCode' => 1,
48         'Error' => 'Erreur est survenue lors de la récupération des utilisateurs'
49     ]);
50     exit();
51 }
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : eventReminder.php
6  * Date        : 03.06.2020
7  * Description : Fichier qui vérifie si un évènement est dans 24h et si c'est le cas
8                  envoie un mail de rappel
9  * Version     : 1.0
10 */
11 require_once __DIR__ . './backend.php';
12
13 $events = GetAllFutureEvent();
14 $now = date('Y-m-d');
15 $mailSubject = "WEGO: Rappel de participation";
16
17 foreach ($events as $event) {
18     $idEvent = $event['idEvenement'];
19     $eventName = $event['nom'];
20     $beginingDate = $event['dateDebut'];
21     $dayBeforeEvent = date('Y-m-d', strtotime($beginingDate . '-1 days'));
22     $mailMessage = <<<EX
23     N'oubliez pas que l'évènement "{$eventName}" se déroule demain.
24     EX;
25
26     if (strtotime($now) >= strtotime($dayBeforeEvent) && strtotime($now) <
27         strtotime($beginingDate)) {
28         if (GetEventNbGuest($idEvent) > 0) {
29             $eventUsersMail = GetEventUsersMail($idEvent);
30
31             foreach ($eventUsersMail as $mail) {
32                 if (SendMail($mail['email'], $mailMessage, $mailSubject) == false) {
33                     echo json_encode([
34                         'ReturnCode' => 1,
35                         'Error' => "Une erreur est survenue lors de l'envoi du mail."
36                     ]);
37                     exit();
38                 }
39             }
40         }
41     }
42 }
```



```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : editProfile.php
 * Date        : 05.06.2020
 * Description  : Fichier qui permet de filtrer les données reçues du call Ajax afin de
modifier le profil.
 * Version     : 1.0
 */

require_once __DIR__ . '/backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idUser = $_SESSION['loggedIn']['idUser'];
$email = $_SESSION['loggedIn']['email'];

$lastname = filter_input(INPUT_POST, 'lastname', FILTER_SANITIZE_STRING);
$firstname = filter_input(INPUT_POST, 'firstname', FILTER_SANITIZE_STRING);
$nickname = filter_input(INPUT_POST, 'nickname', FILTER_SANITIZE_STRING);
$email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
$phoneNumber = filter_input(INPUT_POST, 'phoneNumber', FILTER_SANITIZE_STRING);
$password = filter_input(INPUT_POST, 'password', FILTER_SANITIZE_STRING);
$verifyPassword = filter_input(INPUT_POST, 'verifyPassword', FILTER_SANITIZE_STRING);

$regex = "/^(?=\w{8,})(?=[^a-z]*[a-z])(?=[^A-Z]*[A-Z])(\w*\d)\w*\/"; // Faut que le mdp
contienne au minimum 8char, une majuscule et 1chiffre

// si tous les champs obligatoire sont remplis
if (IsTaken(['userEmail' => $email, 'idUser' => $idUser]) == false) {
    if (IsTaken(['userNickname' => $nickname, 'idUser' => $idUser]) == false) {
        if ($password != "") {
            if (preg_match($regex, $password)) {
                if ($password != $verifyPassword) {
                    echo json_encode([
                        'ReturnCode' => 4,
                        'Error' => 'Les deux mots de passe données ne sont pas les mêmes.'
                    ]);
                    exit();
                }
            } else {
                echo json_encode([
                    'ReturnCode' => 5,
                    'Error' => 'Le mot de passe doit faire au moins 8 caractères et doit contenir
au moins une majuscule et un chiffre.'
                ]);
                exit();
            }
        }
    }
}

if (EditProfile( $nickname, $firstname, $email, $password, $lastname,
```

```
$phoneNumber, $idUser, $oldEmail)) {
52     echo json_encode([
53         'ReturnCode' => 0,
54         'Success' => 'Le compte a bien été modifié.'
55     ]);
56     exit();
57 } else {
58     echo json_encode([
59         'ReturnCode' => 1,
60         'Error' => 'Erreur est survenue lors de la modification du compte'
61     ]);
62     exit();
63 }
64 } else {
65     echo json_encode([
66         'ReturnCode' => 2,
67         'Error' => 'Ce nom d\'utilisateur est déjà utilisé'
68     ]);
69     exit();
70 }
71 } else {
72     echo json_encode([
73         'ReturnCode' => 3,
74         'Error' => 'Cette adresse mail est déjà utilisé'
75     ]);
76     exit();
77 }
--
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet      : WE GO
5  * Page        : editGuestList.php
6  * Date        : 05.06.2020
7  * Description  : fichier qui modifie la liste des invités
8  * Version     : 1.0
9  */
10
11 require_once __DIR__ . '/backend.php';
12
13 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
14 $editGuestList = $_POST['editGuetsList'];
15
16 foreach ($editGuestList as $user) {
17     if ($user[1] == "false") {
18         if (DeleteInvitation($user[0], $idEvent) == false) {
19             echo json_encode([
20                 'ReturnCode' => 1,
21                 'Error' => 'Erreur est survenue lors de la modification de la base de données'
22             ]);
23             exit();
24         }
25     } else {
26         if (CreateInvite($user[0], $idEvent) == false) {
27             echo json_encode([
28                 'ReturnCode' => 1,
29                 'Error' => 'Erreur est survenue lors de la modification de la base de données'
30             ]);
31             exit();
32         }
33     }
34 }
35
36 echo json_encode([
37     'ReturnCode' => 0,
38     'Success' => 'La modification de la liste des invités a bien été effectué'
39 ]);
40 exit();
41
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : editEvent.php
 * Date        : 04.06.2020
 * Description  : Fichier qui permet de filtrer les données reçus du call Ajax afin de
modifier un événement.
 * Version     : 1.0
 */

require_once __DIR__ . '/backend.php';
require_once dirname(dirname(__DIR__)) . '/views/includes/const.inc.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$finfo = finfo_open(FILEINFO_MIME_TYPE);
$idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
$eventName = filter_input(INPUT_POST, "eventName", FILTER_SANITIZE_STRING);
$eventDescription = filter_input(INPUT_POST, "eventDescription",
FILTER_SANITIZE_STRING);
$place = filter_input(INPUT_POST, "place", FILTER_SANITIZE_STRING);
$beginningDate = str_replace('/', '-', filter_input(INPUT_POST, "beginningDate"));
$beginningTime = filter_input(INPUT_POST, "beginningTime");
$endDate = str_replace('/', '-', filter_input(INPUT_POST, "endDate"));
$endTime = filter_input(INPUT_POST, "endTime");
$nbMaxGuest = filter_input(INPUT_POST, "nbMaxGuest", FILTER_SANITIZE_NUMBER_INT);
$type = filter_input(INPUT_POST, "type", FILTER_SANITIZE_NUMBER_INT);
$img = isset($_FILES['img']) ? $_FILES['img'] : null;

$beginningDate = date("Y-m-d H:i", strtotime($beginningTime,
strtotime($beginningDate)));
$endDate = date("Y-m-d H:i", strtotime($endTime, strtotime($endDate)));

if (strlen($eventName) > 0 && strlen($eventDescription) > 0 && $nbMaxGuest > 0 &&
strlen($place) > 0) {
    if (strtotime($beginningDate) > strtotime(date("Y-m-d"))) {
        if (strtotime($endDate) >= strtotime($beginningDate)) {
            if (isset($img['tmp_name'])) {
                $mimeType = finfo_file($finfo, $img['tmp_name']);
                $fileType = explode('/', $mimeType)[0];

                if ($fileType == 'image') {
                    $fileExtension = pathinfo($img['name'], PATHINFO_EXTENSION);
                    $filename = uniqid() . '.' . $fileExtension;

                    if (move_uploaded_file($img['tmp_name'], UPLOAD_PATH . $filename)) {
                        if (EditEvent($eventName, $eventDescription, $place, $beginningDate,
$endDate, $nbMaxGuest, './assets/upload/' . $filename, $idEvent)) {
                            echo json_encode([
                                'ReturnCode' => 0,
                                'Success' => "L'événement " . $eventName . " a bien été modifié"
```

```
50         });
51         exit();
52     }
53     } else {
54         echo json_encode([
55             'ReturnCode' => 1,
56             'Error' => "Un problème est survenue lors de l'upload de l'image"
57         ]);
58         exit();
59     }
60     } else {
61         echo json_encode([
62             'ReturnCode' => 2,
63             'Error' => "Le fichier données n'est pas une image"
64         ]);
65         exit();
66     }
67 }
68
69 if (EditEvent($eventName, $eventDescription, $place, $beginningDate, $endDate,
70 $nbMaxGuest, null, $idEvent)) {
71     echo json_encode([
72         'ReturnCode' => 0,
73         'Success' => "L'événement " . $eventName . " a bien été modifié"
74     ]);
75     exit();
76 } else {
77     echo json_encode([
78         'ReturnCode' => 3,
79         'Error' => "L'événement ne peut pas se terminer avant d'avoir commencé"
80     ]);
81     exit();
82 }
83 } else {
84     echo json_encode([
85         'ReturnCode' => 4,
86         'Error' => "L'événement ne peut pas commencer avant aujourd'hui"
87     ]);
88     exit();
89 }
90 }
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : deleteProfile.php
 * Date        : 04.06.2020
 * Description  : Fichier qui vérifie si le profile est supprimable
 * Version     : 1.0
 */

require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
require_once __DIR__ . './backend.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idUser = filter_input(INPUT_POST, 'idUser', FILTER_SANITIZE_STRING);

$query = <<<EX
DELETE FROM user WHERE idUser = :idUser;
EX;

if ($_SESSION['loggedIn']['idUser'] != $idUser) {
    echo json_encode([
        'ReturnCode' => 2
    ]);
    exit();
}

$usersEventParticipating = GetEventsParticipating($idUser);
$usersEventInvited = GetEventsInvited($idUser);
$userEventCreated = GetUsersEvent($idUser);
$getUserInvitedToTheEvent = [];

if (count($usersEventParticipating) > 0) {
    foreach ($usersEventParticipating as $seventParticipating) {
        if (DeleteParticipation($idUser, $seventParticipating['idEvenement']) == false) {
            echo json_encode([
                'ReturnCode' => 1,
                'Error' => 'Une erreur est survenue lors de la suppression de votre profile.'
            ]);
        }
    }
}

if (count($usersEventInvited) > 0) {
    foreach ($usersEventInvited as $seventInvited) {
        if (IsInvited($seventInvited['idEvenement'], $idUser)) {
            if (DeleteInvitation($idUser, $seventInvited['idEvenement']) == false) {
                echo json_encode([
                    'ReturnCode' => 1,
                    'Error' => 'Une erreur est survenue lors de la suppression de votre profile.'
                ]);
            }
        }
    }
}
```

```
55     }
56   }
57 }
58 }
59
60 foreach ($userEventCreated as $event) {
61     $getUserInvitedToTheEvent = GetInvited($event['idEvenement']);
62
63     if (count($getUserInvitedToTheEvent) > 0) {
64         foreach ($getUserInvitedToTheEvent as $user) {
65             if (DeleteInvitation($user['idUser'], $event['idEvenement']) == false) {
66                 echo json_encode([
67                     'ReturnCode' => 1,
68                     'Error' => 'Une erreur est survenue lors de la suppression de votre profile.'
69                 ]);
70             }
71         }
72     }
73
74     if (DeleteEvent($event['idEvenement']) == false) {
75         echo json_encode([
76             'ReturnCode' => 1,
77             'Error' => 'Une erreur est survenue lors de la suppression de votre profile.'
78         ]);
79     }
80 }
81
82 try {
83     DatabaseController::beginTransaction();
84
85     $requestDeleteEvent = DatabaseController::prepare($query);
86     $requestDeleteEvent->bindParam(':idUser', $idUser, PDO::PARAM_INT);
87     $requestDeleteEvent->execute();
88
89     DatabaseController::commit();
90
91     echo json_encode([
92         'ReturnCode' => 0,
93         'Success' => "Votre compte à bien été supprimé."
94     ]);
95 } catch (PDOException $e) {
96     DatabaseController::rollBack();
97     echo json_encode([
98         'ReturnCode' => 1,
99         'Error' => 'Une erreur est survenue lors de la suppression de votre profile.'
100     ]);
101 }
102
```

```
1 <?php
2 /*
3  * Auteur      : Hoarau Nicolas
4  * Projet     : WE GO
5  * Page       : deleteEvent.php
6  * Date       : 03.06.2020
7  * Description : Fichier qui supprime un événement donnée
8  * Version    : 1.0
9  */
10
11 require_once __DIR__ . '/backend.php';
12
13 $idEvent = filter_input(INPUT_POST, 'idEvent', FILTER_SANITIZE_NUMBER_INT);
14 $eventName = filter_input(INPUT_POST, 'eventName', FILTER_SANITIZE_STRING);
15 $private = filter_input(INPUT_POST, 'private', FILTER_SANITIZE_NUMBER_INT);
16
17 if ($private == 1) {
18     $invited = GetInvited($idEvent);
19     foreach ($invited as $user) {
20         if (DeleteInvitation($user['idUser'], $idEvent) == false) {
21             echo json_encode([
22                 'ReturnCode' => 1,
23                 'Error' => "Erreur est survenue lors de la suppression de l'événement"
24             ]);
25             exit();
26         }
27     }
28 }
29
30 if (DeleteEvent($idEvent)) {
31     echo json_encode([
32         'ReturnCode' => 0,
33         'Success' => "L'événement \"" . $eventName . "\" est bien supprimé"
34     ]);
35     exit();
36 } else {
37     echo json_encode([
38         'ReturnCode' => 1,
39         'Error' => "Erreur est survenue lors de la suppression de l'événement"
40     ]);
41     exit();
42 }
```



```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : createEvent.php
 * Date        : 27.05.2020
 * Description  : Fichier qui permet de filtrer les données reçus du call Ajax afin de
créer un événement.
 * Version     : 1.0
 */

require_once __DIR__ . '/backend.php';
require_once dirname(dirname(__DIR__)) . '/views/includes/const.inc.php';

if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

$idUser = $_SESSION['loggedIn']['idUser'];
$finfo = finfo_open(FILEINFO_MIME_TYPE);

$eventName = filter_input(INPUT_POST, "eventName", FILTER_SANITIZE_STRING);
$eventDescription = filter_input(INPUT_POST, "eventDescription",
FILTER_SANITIZE_STRING);
$place = filter_input(INPUT_POST, "place", FILTER_SANITIZE_STRING);
$beginningDate = str_replace('/', '-', filter_input(INPUT_POST, "beginningDate"));
$beginningTime = filter_input(INPUT_POST, "beginningTime");
$endDate = str_replace('/', '-', filter_input(INPUT_POST, "endDate"));
$endTime = filter_input(INPUT_POST, "endTime");
$nbMaxGuest = filter_input(INPUT_POST, "nbMaxGuest", FILTER_SANITIZE_NUMBER_INT);
$type = filter_input(INPUT_POST, "type", FILTER_SANITIZE_NUMBER_INT);
$img = isset($_FILES['img']) ? $_FILES['img'] : DEFAULT_IMG;

$guestlist = isset($_POST['guestlist']) ? explode(',', $_POST['guestlist']): null;

$beginningDate = date("Y-m-d H:i", strtotime($beginningTime,
strtotime($beginningDate)));
$endDate = date("Y-m-d H:i", strtotime($endTime, strtotime($endDate)));

if (strlen($eventName) > 0 && strlen($eventDescription) > 0 && $nbMaxGuest > 0 &&
strlen($place) > 0) {
    if (strtotime($beginningDate) > strtotime(date("Y-m-d"))) {
        if (strtotime($endDate) >= strtotime($beginningDate)) {

            if (isset($img['tmp_name'])) {
                $mimeType = finfo_file($finfo, $img['tmp_name']);
                $fileType = explode('/', $mimeType)[0];

                if ($fileType == 'image') {
                    $fileExtension = pathinfo($img['name'], PATHINFO_EXTENSION);
                    $filename = uniqid() . '.' . $fileExtension;

                    if (move_uploaded_file($img['tmp_name'], UPLOAD_PATH . $filename)) {
                        if (CreateEvent($eventName, $eventDescription, $place, $beginningDate,
```

```
$endDate, $nbMaxGuest, './assets/upload/' . $filename, $type, $idUser, $guestlist)) {
51     echo json_encode([
52         'ReturnCode' => 0,
53         'Success' => "L'événement " . $eventName . " a bien été créé"
54     ]);
55     exit();
56 }
57 } else {
58     echo json_encode([
59         'ReturnCode' => 1,
60         'Error' => "Un problème est survenue lors de l'upload de l'image"
61     ]);
62     exit();
63 }
64 } else {
65     echo json_encode([
66         'ReturnCode' => 2,
67         'Error' => "Le fichier données n'est pas une image"
68     ]);
69     exit();
70 }
71 }
72
73 if (CreateEvent($eventName, $eventDescription, $place, $beginningDate, $endDate,
74 $nbMaxGuest, $img, $type, $idUser, $guestlist)) {
75     echo json_encode([
76         'ReturnCode' => 0,
77         'Success' => "L'événement " . $eventName . " a bien été créé"
78     ]);
79     exit();
80 } else {
81     echo json_encode([
82         'ReturnCode' => 3,
83         'Error' => "L'événement ne peut pas se terminer avant d'avoir commencé"
84     ]);
85     exit();
86 }
87 } else {
88     echo json_encode([
89         'ReturnCode' => 4,
90         'Error' => "L'événement ne peut pas commencer avant aujourd'hui"
91     ]);
92     exit();
93 }
94 }
--
```

```
<?php
/*
 * Auteur      : Hoarau Nicolas
 * Projet      : WE GO
 * Page        : backend.php
 * Date        : 26.05.2020
 * Description  : Fichier qui contient toutes les fonctionsdu site.
 * Version     : 1.0
 */

/* Import PHPMailer classes into the global namespace
   These must be at the top of your script, not inside a function*/

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

// Load Composer's autoloader
require dirname(dirname(__DIR__)) . '/vendor/autoload.php';
require_once dirname(__DIR__) . '/Controller/DatabaseController.php';
require_once dirname(dirname(__DIR__)) . '/config/config.php';

/**
 * @date 26.05.20
 * @author Hoarau Nicolas
 * @brief Fonction qui vérifie si l'utilisateur est connecté
 * @return boolean
 * @version 1.0
 */
function IsLogged(): bool
{
    return array_key_exists('loggedIn', $_SESSION) && $_SESSION['loggedIn'];
}

/**
 * @author Hoarau Nicolas
 * @date 27.05.2020
 * @brief Fonction qui récupère le salt de l'utilisateur
 * @param array $args
 * @return string
 * @version 1.0
 */
function GetSalt(array $args): ?string
{
    // initialise à null les colonnes du tableau vide/inexistante
    $args += [
        'userEmail' => null,
        'userNickname' => null
    ];

    extract($args); // extrait les données du tableau avec comme nom de varvariable son
    nom de colonne

    $clauseField = "";
```

```
54 $clauseValue = "";
55
56 if ($userEmail !== null) {
57     $clauseField = "email";
58     $clauseValue = $userEmail;
59 } elseif ($userNickname) {
60     $clauseField = "pseudo";
61     $clauseValue = $userNickname;
62 }
63
64 $query = <<<EX
65     SELECT salt
66     FROM user
67     WHERE `{ $clauseField }` = :clauseValue;
68     EX;
69
70 try {
71     $requestSalt = DatabaseController::prepare($query);
72     $requestSalt->bindParam(':clauseValue', $clauseValue, PDO::PARAM_STR);
73     $requestSalt->execute();
74
75     $result = $requestSalt->fetch(PDO::FETCH_ASSOC);
76
77     return $result !== false ? $result['salt'] : null;
78 } catch (PDOException $e) {
79     return null;
80 }
81 }
82
83 /**
84  * @author Hoarau Nicolas
85  * @date 26.05.2020
86  * @brief Fonction qui créer un nouvel utilisateur dans la base de données
87  * @param string $nickname
88  * @param string $lastname
89  * @param string $firstname
90  * @param string $email
91  * @param string $phoneNumber
92  * @param string $password
93  * @return boolean
94  * @version 1.0
95  *     Inscription sans salt
96  * @version 1.1
97  *     Inscription avec salt
98  */
99 function Register(string $nickname, string $firstname, string $email, string
    $password, string $lastname = null, string $phoneNumber = null): bool
100 {
101     $query = "";
102     $salt = hash('sha256', microtime());
103     $userPassword = hash('sha256', $password . $salt);
104
105     if ($lastname != null && $phoneNumber != null) {
106         $query = <<<EX
107             INSERT INTO user (pseudo, prenom, nom, email, password, telephone, salt)
```

```
108     VALUES (:nickname, :firstname, :lastname, :email, :password, :phoneNumber,
:salt);
109     EX;
110 } elseif ($phoneNumber != null) {
111     $query = <<<EX
112     INSERT INTO user (pseudo, prenom, nom, email, password, telephone, salt)
113     VALUES (:nickname, :firstname, :email, :password, :phoneNumber, :Salt);
114     EX;
115 } elseif ($lastname != null) {
116     $query = <<<EX
117     INSERT INTO user (pseudo, prenom, nom, email, password, salt)
118     VALUES (:nickname, :firstname, :lastname, :email, :password, :salt);
119     EX;
120 } else {
121     $query = <<<EX
122     INSERT INTO user (pseudo, prenom, email, password, salt)
123     VALUES (:nickname, :firstname, :email, :password, :salt);
124     EX;
125 }
126
127 try {
128     DatabaseController::beginTransaction();
129
130     $requestRegister = DatabaseController::prepare($query);
131     $requestRegister->bindParam(':nickname', $nickname, PDO::PARAM_STR, 50);
132     $requestRegister->bindParam(':firstname', $firstname, PDO::PARAM_STR, 50);
133
134     if ($lastname != null)
135         $requestRegister->bindParam(':lastname', $lastname, PDO::PARAM_STR, 50);
136
137     $requestRegister->bindParam(':email', $email, PDO::PARAM_STR, 100);
138     $requestRegister->bindParam(':password', $userPassword, PDO::PARAM_STR, 100);
139
140     if ($phoneNumber != null)
141         $requestRegister->bindParam(':phoneNumber', $phoneNumber, PDO::PARAM_STR, 50);
142
143     $requestRegister->bindParam(':salt', $salt, PDO::PARAM_STR, 64);
144
145     $requestRegister->execute();
146
147     DatabaseController::commit();
148     return true;
149 } catch (PDOException $e) {
150     DatabaseController::rollBack();
151     return false;
152 }
153 }
154
155 /**
156  * @author Hoarau Nicolas
157  * @date 26.05.2020
158  * @brief Fonction qui connecte l'utilisateur
159  * @param array les données de connexion de l'utilisateur
160  * @return array|false
161  * @version 1.0
```

```
162 *      Inscription sans salt
163 * @version 1.1
164 *      Inscription avec salt
165 */
166 function Login(array $args)
167 {
168     // initialise à null les colonnes du tableau vide/inexistante
169     $args += [
170         'userEmail' => null,
171         'userNickname' => null,
172         'userPwd' => null,
173     ];
174
175     extract($args); // extrait les données du tableau avec comme nom de variable son
    nom de colonne
176
177     $loginField = "";
178     $authenticator = "";
179     $salt = "";
180
181     if ($userEmail !== null) {
182         $salt = GetSalt(['userEmail' => $userEmail]);
183         $authenticator = $userEmail;
184         $loginField = "email";
185     } elseif ($userNickname !== null) {
186         $salt = GetSalt(['userNickname' => $userNickname]);
187         $authenticator = $userNickname;
188         $loginField = "pseudo";
189     } else {
190         return false;
191     }
192
193     if ($userPwd == null)
194         return false;
195
196     $pwd = hash('sha256', $userPwd . $salt);
197
198     $query = <<<EX
199     SELECT idUser, pseudo, prenom, nom, email, telephone
200     FROM user
201     WHERE `{ $loginField }` = :wayToConnectValue
202     AND password = :pwd;
203     EX;
204
205     try {
206         $requestLogin = DatabaseController::prepare($query);
207         $requestLogin->bindParam(':wayToConnectValue', $authenticator, PDO::PARAM_STR);
208         $requestLogin->bindParam(':pwd', $pwd, PDO::PARAM_STR);
209         $requestLogin->execute();
210
211         $result = $requestLogin->fetch(PDO::FETCH_ASSOC);
212
213         return $result !== false & $result > 0 ? $result : false;
214     } catch (PDOException $e) {
215         return false;
216     }
```

```
216 }
217 }
218
219 /**
220  * @author Hoarau Nicolas
221  * @date 26.05.2020
222  * @brief Fonction qui vérifie si le paramètre donnée(email, nickname) est déjà
    utilisé ou non
223  * @param array $args
224  * @return boolean|null
225  * @version 1.0
226  */
227 function IsTaken(array $args): ?bool
228 {
229     $args += [
230         'userEmail' => null,
231         'userNickname' => null,
232         'idUser' => null
233     ];
234
235     extract($args);
236
237     $authenticator = "";
238     $field = "";
239     if ($userEmail != null) {
240         $authenticator = $userEmail;
241         $field = "email";
242     } elseif ($userNickname != null) {
243         $authenticator = $userNickname;
244         $field = "pseudo";
245     }
246
247     $queryWithoutIdUser = <<<EX
248         SELECT `{ $field }`
249         FROM user
250         WHERE `{ $field }` = :authenticator;
251     EX;
252
253     $queryWithIdUser = <<<EX
254         SELECT `{ $field }`
255         FROM user
256         WHERE `{ $field }` = :authenticator
257         AND idUser <> :idUser;
258     EX;
259
260     $query = ($idUser != null) ? $queryWithIdUser : $queryWithoutIdUser;
261
262     try {
263         $requestIsUsed = DatabaseController::prepare($query);
264         $requestIsUsed->bindParam(':authenticator', $authenticator, PDO::PARAM_STR);
265
266         if ($idUser != null)
267             $requestIsUsed->bindParam(':idUser', $idUser, PDO::PARAM_INT);
268
269     }
```

```
270     $requestIsUsed->execute();
271     $result = $requestIsUsed->fetch(PDO::FETCH_ASSOC);
272
273     return $result !== false ? true : false;
274 } catch (PDOException $e) {
275     return null;
276 }
277 }
278
279 /**
280  * @author Hoarau Nicolas
281  * @date 27.05.2020
282  * @brief Fonction qui crée un événement
283  * @param string $eventName
284  * @param string $eventDescription
285  * @param string $place
286  * @param Date $beginningDate
287  * @param Date $endDate
288  * @param integer $nbMaxGuest
289  * @param FILES $img
290  * @param integer $idUser
291  * @return boolean
292  * @version 1.0
293  */
294 function CreateEvent(string $eventName, string $eventDescription, string $place,
    $beginningDate, $endDate, int $nbMaxGuest, $img, int $type, int $idUser, array
    $guestlist = null): bool
295 {
296     $creationDate = date('Y-m-d H:i');
297     $beginningDate = date("Y-m-d H:i", strtotime($beginningDate));
298     $endDate = date("Y-m-d H:i", strtotime($endDate));
299
300     $query = <<<EX
301     INSERT INTO evenement (nom, descriptif, dateDebut, dateFin, dateCreation, lieu,
    prive, nbMaxParticipant, urlImage, idOrganisateur)
302     VALUES (:name, :description, :beginningDate, :endDate, :creationDate, :place,
    :type, :nbMaxGuest, :urlImg, :idUser);
303     EX;
304
305     try {
306         DatabaseController::beginTransaction();
307
308         $requestCreateEvent = DatabaseController::prepare($query);
309         $requestCreateEvent->bindParam(':name', $eventName, PDO::PARAM_STR, 45);
310         $requestCreateEvent->bindParam(':description', $eventDescription, PDO::PARAM_STR,
    255);
311         $requestCreateEvent->bindParam(':beginningDate', $beginningDate);
312         $requestCreateEvent->bindParam(':endDate', $endDate);
313         $requestCreateEvent->bindParam(':creationDate', $creationDate);
314         $requestCreateEvent->bindParam(':place', $place, PDO::PARAM_STR, 255);
315         $requestCreateEvent->bindParam(':type', $type, PDO::PARAM_INT, 1);
316         $requestCreateEvent->bindParam(':nbMaxGuest', $nbMaxGuest, PDO::PARAM_INT, 11);
317         $requestCreateEvent->bindParam(':urlImg', $img, PDO::PARAM_STR, 255);
318         $requestCreateEvent->bindParam(':idUser', $idUser, PDO::PARAM_INT, 11);
319
```



```
320     $requestCreateEvent->execute();
321
322     if ($guestlist != null) {
323         $lastInsertId = DatabaseController::getInstance()->lastInsertId();
324
325         for ($i = 0; $i < count($guestlist); $i++) {
326             if ($idUser != $guestlist[$i]) {
327                 if (CreateInvite($guestlist[$i], $lastInsertId) == false) {
328                     DatabaseController::rollBack();
329                 }
330             }
331         }
332
333         DatabaseController::commit();
334         return true;
335     }
336
337     DatabaseController::commit();
338     return true;
339 } catch (PDOException $e) {
340     DatabaseController::rollBack();
341     return false;
342 }
343 }
344
345 /**
346  * @author Hoarau Nicolas
347  * @date 27.05.2020
348  * @brief Fonction qui crée une invitation pour un utilisateur à un événement
349  * @param integer $idGuest
350  * @param integer $idEvent
351  * @return boolean
352  * @version 1.0
353  */
354 function CreateInvite(int $idGuest, int $idEvent): bool
355 {
356     $query = <<<EX
357         INSERT INTO invites (idUser, idEvenement)
358         VALUES (:idGuest, :idEvent);
359     EX;
360
361     try {
362         DatabaseController::beginTransaction();
363
364         $requestCreateInvite = DatabaseController::prepare($query);
365         $requestCreateInvite->bindParam(':idGuest', $idGuest, PDO::PARAM_INT, 11);
366         $requestCreateInvite->bindParam(':idEvent', $idEvent, PDO::PARAM_INT, 11);
367         $requestCreateInvite->execute();
368
369         DatabaseController::commit();
370
371         return true;
372     } catch (PDOException $e) {
373         DatabaseController::rollBack();
374         return false;
375     }
376 }
```

```
375 }
376 }
377
378 /**
379  * @author Hoarau Nicolas
380  * @date 05.06.2020
381  * @brief Fonction qui supprime l'invitation d'un utilisateur à un événement
382  * @param integer $idGuest
383  * @param integer $idEvent
384  * @return boolean
385  * @version 1.0
386  */
387 function DeleteInvite(int $idGuest, int $idEvent): bool
388 {
389     $query = <<<EX
390     DELETE FROM invites WHERE idUser = :idUser AND idEvenement = :idEvent;
391     EX;
392
393     try {
394         DatabaseController::beginTransaction();
395
396         $requestDeleteInvite = DatabaseController::prepare($query);
397         $requestDeleteInvite->bindParam(':idUser', $idGuest, PDO::PARAM_INT, 11);
398         $requestDeleteInvite->bindParam(':idEvent', $idEvent, PDO::PARAM_INT, 11);
399         $requestDeleteInvite->execute();
400
401         DatabaseController::commit();
402
403         return true;
404     } catch (PDOException $e) {
405         DatabaseController::rollBack();
406         return false;
407     }
408 }
409
410 /**
411  * @author Hoarau Nicolas
412  * @date 02.06.2020
413  * @brief Fonction qui vérifie si l'événement existe
414  * @param integer $idEvent
415  * @return boolean
416  * @version 1.0
417  */
418 function EventExist(int $idEvent): bool
419 {
420     $query = <<<EX
421     SELECT COUNT(idEvenement) AS exist
422     FROM evenement
423     WHERE idEvenement = :idEvent
424     EX;
425
426     try {
427         $requestExist = DatabaseController::prepare($query);
428         $requestExist->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
429         $requestExist->execute();
```

```
430
431     $result = $requestExist->fetch(PDO::FETCH_ASSOC);
432
433     return $result['exist'] == 0 ? false : true;
434 } catch (PDOException $e) {
435     return false;
436 }
437 }
438
439 /**
440  * @author Hoarau Nicolas
441  * @date 02.06.2020
442  * @brief Fonction qui vérifie si l'utilisateur participa à l'événement
443  * @param integer $idEvent
444  * @param integer $idUser
445  * @return boolean
446  * @version 1.0
447  */
448 function IsParticipating(int $idEvent, int $idUser): bool
449 {
450     $query = <<<EX
451     SELECT COUNT(*) AS participating
452     FROM inscriptions
453     WHERE idUser = :idUser
454     AND idEvenement = :idEvent
455     EX;
456
457     try {
458         $requestIsParticipating = DatabaseController::prepare($query);
459         $requestIsParticipating->bindParam(':idUser', $idUser, PDO::PARAM_INT);
460         $requestIsParticipating->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
461         $requestIsParticipating->execute();
462
463         $result = $requestIsParticipating->fetch(PDO::FETCH_ASSOC);
464
465         return $result['participating'] == 0 ? false : true;
466     } catch (PDOException $e) {
467         return false;
468     }
469 }
470
471 /**
472  * @author Hoarau Nicolas
473  * @date 02.06.2020
474  * @brief Fonction qui vérifie si l'événement est privé
475  * @param integer $idEvent
476  * @return boolean
477  * @version 1.0
478  */
479 function IsPrivate(int $idEvent): bool
480 {
481     $query = <<<EX
482     SELECT prive
483     FROM evenement
484     WHERE idEvenement = :idEvent;
```

```
485     EX;
486
487     try {
488         $requestIsPrivate = DatabaseController::prepare($query);
489         $requestIsPrivate->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
490         $requestIsPrivate->execute();
491
492         $result = $requestIsPrivate->fetch(PDO::FETCH_ASSOC);
493
494         return $result['prive'] == 0 ? false : true;
495     } catch (PDOException $e) {
496         return false;
497     }
498 }
499
500 /**
501  * @author Hoarau Nicolas
502  * @date 02.06.2020
503  * @brief Fonction qui vérifie si l'utilisateur est invité à l'événement donnée
504  * @param integer $idEvent
505  * @param integer $idUser
506  * @return boolean
507  * @version 1.0
508  */
509 function IsInvited(int $idEvent, int $idUser): bool
510 {
511     $query = <<<EX
512     SELECT COUNT(*) AS isInvited
513     FROM invites
514     WHERE idUser = :idUser
515     AND idEvenement = :idEvent;
516     EX;
517
518     try {
519         $requestIsInvited = DatabaseController::prepare($query);
520         $requestIsInvited->bindParam(':idUser', $idUser, PDO::PARAM_INT);
521         $requestIsInvited->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
522         $requestIsInvited->execute();
523         $result = $requestIsInvited->fetch(PDO::FETCH_ASSOC);
524
525         return $result['isInvited'] == 1 ? true : false;
526     } catch (PDOException $e) {
527         return false;
528     }
529 }
530
531 /**
532  * @author Hoarau Nicolas
533  * @date 02.06.2020
534  * @brief Fonction qui vérifie si l'utilisateur est le créateur de l'événement donnée
535  * @param integer $idEvent
536  * @param integer $idUser
537  * @return boolean
538  * @version 1.0
539  */
```

```
540 function IsCreator(int $idEvent, int $idUser): bool
541 {
542     $query = <<<EX
543     SELECT COUNT(*) AS isCreator
544     FROM evenement
545     WHERE idOrganisateur = :idUser
546     AND idEvenement = :idEvent;
547     EX;
548
549     try {
550         $requestIsCreator = DatabaseController::prepare($query);
551         $requestIsCreator->bindParam(':idUser', $idUser, PDO::PARAM_INT);
552         $requestIsCreator->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
553         $requestIsCreator->execute();
554
555         $result = $requestIsCreator->fetch(PDO::FETCH_ASSOC);
556
557         return $result['isCreator'] == 1 ? true : false;
558     } catch (PDOException $e) {
559         return false;
560     }
561 }
562
563 /**
564  * @author Hoarau Nicolas
565  * @date 02.06.2020
566  * @brief Fonction qui récupère les information de l'événement
567  * @param integer $idEvent
568  * @param integer $idUser
569  * @return array|false
570  * @version 1.0
571  */
572 function GetEventData(int $idEvent, int $idUser = null)
573 {
574     $queryLogged = <<<EX
575     SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.lieu,
576     e.nbMaxParticipant, e.prive, e.urlImage, u.pseudo, (SELECT COUNT(*) FROM inscriptions
577     WHERE idUser = :idUser AND idEvenement = :idEvent) AS participating, (SELECT
578     COUNT(idUser) FROM inscriptions WHERE idevenement = :idEvent) AS nbGuest
579     FROM evenement AS e
580     JOIN user AS u ON e.idOrganisateur = u.idUser
581     WHERE e.idEvenement = :idEvent;
582     EX;
583
584     $queryNotLogged = <<<EX
585     SELECT e.idEvenement, e.nom, e.descriptif, e.dateDebut, e.dateFin, e.lieu,
586     e.nbMaxParticipant, e.prive, e.urlImage, u.pseudo, (SELECT COUNT(idUser) FROM
587     inscriptions WHERE idEvenement = :idEvent) AS nbGuest
588     FROM evenement AS e
589     JOIN user AS u ON e.idOrganisateur = u.idUser
590     WHERE e.idEvenement = :idEvent;
591     EX;
592
593     $query = $idUser !== null ? $queryLogged : $queryNotLogged;
```

```
590 try {
591     $requestGetEventData = DatabaseController::prepare($query);
592
593     if ($idUser !== null)
594         $requestGetEventData->bindParam(':idUser', $idUser, PDO::PARAM_INT);
595
596     $requestGetEventData->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
597     $requestGetEventData->execute();
598
599     $result = $requestGetEventData->fetch(PDO::FETCH_ASSOC);
600
601     return $result !== false ? $result : null;
602 } catch (PDOException $e) {
603     return false;
604 }
605 }
606
607 /**
608  * @author Hoarau Nicolas
609  * @date 02.06.2020
610  * @brief Fonction qui envoie un mail à un destinataire précit
611  * @param string $recipientMail
612  * @param string $message
613  * @return bool
614  * @version 1.0
615  */
616 function SendMail(string $recipientMail, string $message, string $subject): bool
617 {
618     // Instantiation and passing `true` enables exceptions
619     $mail = new PHPMailer(true);
620
621     try {
622         //Enable SMTP debugging.
623         $mail->SMTPDebug = 0;
624         //Set PHPMailer to use SMTP.
625         $mail->isSMTP();
626
627         //Set SMTP host name
628         $mail->Host = MAIL_HOST;
629
630         //Set this to true if SMTP host requires authentication to send email
631         $mail->SMTPAuth = true;
632
633         //Provide username and password
634         $mail->Username = MAIL_USERNAME;
635         $mail->Password = MAIL_PASSWORD;
636
637         //If SMTP requires TLS encryption then set it
638         $mail->SMTPSecure = "tls";
639
640         //Set TCP port to connect to
641         $mail->Port = MAIL_PORT;
642         $mail->From = MAIL_USERNAME;
643         $mail->FromName = "WEGO Service";
644     }
```

```
645     $mail->addAddress($recipientMail);
646
647     $mail->isHTML(true);
648
649     $mail->Subject = $subject;
650     $mail->Body = $message;
651
652     if ($mail->send())
653         return true;
654     else
655         return false;
656 } catch (Exception $e) {
657     return false;
658 }
659 }
660
661 /**
662  * @author Hoarau Nicolas
663  * @date 03.06.2020
664  * @brief Récupère tous les événements publics pas encore passé
665  * @return array|null
666  * @version 1.0
667  */
668 function GetAllFutureEvent(): ?array
669 {
670     $dateMoreOneHour = date('Y-m-d H:i:s', strtotime(date('Y-m-d H:i:s') . '+1hours'));
671
672     $query = <<<EX
673     SELECT idEvenement, nom, dateDebut
674     FROM evenement
675     WHERE TIMESTAMP(dateDebut) >= :dateTime
676     ORDER BY dateDebut DESC
677     EX;
678
679     try {
680         $requestGetAllFutureEvent = DatabaseController::prepare($query);
681         $requestGetAllFutureEvent->bindParam(':dateTime', $dateMoreOneHour);
682         $requestGetAllFutureEvent->execute();
683         $result = $requestGetAllFutureEvent->fetchAll(PDO::FETCH_ASSOC);
684         return $result;
685     } catch (PDOException $e) {
686         return null;
687     }
688 }
689
690 /**
691  * @author Hoarau Nicolas
692  * @date 03.06.2020
693  * @brief Fonction qui récupère le nombre de participant à un événement
694  * @param integer $idEvent
695  * @return integer|null
696  * @version 1.0
697  */
698 function GetEventNbGuest(int $idEvent): ?int
699 {
```

```
700 $query = <<<EX
701 SELECT COUNT(idUser) AS nbGuest
702 FROM inscriptions
703 WHERE idEvenement = :idEvent
704 EX;
705
706 try {
707     $requestGetEventNbGuest = DatabaseController::prepare($query);
708     $requestGetEventNbGuest->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
709     $requestGetEventNbGuest->execute();
710
711     $result = $requestGetEventNbGuest->fetch(PDO::FETCH_ASSOC);
712
713     return $result !== false ? $result['nbGuest'] : 0;
714 } catch (PDOException $e) {
715     return null;
716 }
717 }
718
719 /**
720  * @author Hoarau Nicolas
721  * @date 03.06.2020
722  * @brief Fonction qui récupère les addresses mail des participant à l'événement donr
723  * @param integer $idEvent
724  * @return array|null
725  * @version 1.0
726  */
727 function GetEventUsersMail(int $idEvent): ?array
728 {
729     $query = <<<EX
730     SELECT u.email
731     FROM inscriptions AS i
732     JOIN user AS u on i.idUser = u.idUser
733     WHERE idEvenement = :idEvent;
734     EX;
735
736     try {
737         $requestGetEventUsersMail = DatabaseController::prepare($query);
738         $requestGetEventUsersMail->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
739         $requestGetEventUsersMail->execute();
740
741         $result = $requestGetEventUsersMail->fetchAll(PDO::FETCH_ASSOC);
742
743         return $result;
744     } catch (PDOException $e) {
745         return null;
746     }
747 }
748
749 /**
750  * @author Hoarau Nicolas
751  * @date 04.06.2020
752  * @brief Fonction qui modifie un événement donné
753  * @param string $eventName
754  * @param string $eventDescription
```



```
755 * @param string $place
756 * @param Date $beginningDate
757 * @param Date $endDate
758 * @param integer $nbMaxGuest
759 * @param integer $idUser
760 * @param integer $idEvent
761 * @return boolean
762 * @version 1.0
763 */
764 function EditEvent(string $eventName, string $eventDescription, string $place,
    $beginningDate, $endDate, int $nbMaxGuest, $img = null, int $idEvent): bool
765 {
766     $query = <<<EX
767     UPDATE evenement
768     SET nom = :eventName, descriptif = :eventDescription, lieu = :place, dateDebut =
    :beginningDate, dateFin = :endDate, nbMaxParticipant = :nbMaxGuest
769     EX;
770
771     if ($img != null)
772         $query .= ", urlImage = :urlImg";
773
774     $query .= " WHERE idEvenement = :idEvent";
775
776     try {
777         DatabaseController::beginTransaction();
778         $requestEditEvent = DatabaseController::prepare($query);
779         $requestEditEvent->bindParam(':eventName', $eventName, PDO::PARAM_STR);
780         $requestEditEvent->bindParam(':eventDescription', $eventDescription,
    PDO::PARAM_STR);
781         $requestEditEvent->bindParam(':place', $place, PDO::PARAM_STR);
782         $requestEditEvent->bindParam(':beginningDate', $beginningDate);
783         $requestEditEvent->bindParam(':endDate', $endDate);
784         $requestEditEvent->bindParam(':nbMaxGuest', $nbMaxGuest, PDO::PARAM_INT);
785
786         if ($img != null)
787             $requestEditEvent->bindParam(':urlImg', $img, PDO::PARAM_STR, 255);
788
789         $requestEditEvent->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
790         $requestEditEvent->execute();
791         DatabaseController::commit();
792         return true;
793     } catch (PDOException $e) {
794         DatabaseController::rollBack();
795         return false;
796     }
797 }
798
799 /**
800 * @author Hoarau Nicolas
801 * @date 04.06.2020
802 * @brief Fonction qui récupère l'id des événements d'un utilisateur
803 * @param integer $idUser
804 * @return array|int|null
805 * @version 1.0
806 */
```

```
807 function GetUsersEvent(int $idUser)
808 {
809     $query = <<<EX
810     SELECT idEvenement FROM evenement WHERE idOrganisateur = :idUser;
811     EX;
812
813     try {
814         $requestGetUsersEvents = DatabaseController::prepare($query);
815         $requestGetUsersEvents->bindParam(':idUser', $idUser, PDO::PARAM_INT);
816         $requestGetUsersEvents->execute();
817
818         $result = $requestGetUsersEvents->fetchAll(PDO::FETCH_ASSOC);
819         return $result == false ? 0 : $result;
820     } catch (PDOException $e) {
821         return null;
822     }
823 }
824
825 /**
826  * @author Hoarau Nicolas
827  * @date 05.06.2020
828  * @brief Fonction qui compte le nombre de participant aux événement de l'utilisateur
829  * @param array $usersEvent
830  * @return integer
831  * @version 1.0
832  */
833 function CountUsersGuest(array $usersEvent): int
834 {
835     $result = 0;
836
837     foreach ($usersEvent as $event) {
838         $result += GetEventNbGuest($event['idEvenement']);
839
840         if ($result > 0)
841             break;
842     }
843
844     return $result;
845 }
846
847 /**
848  * @author Hoarau Nicolas
849  * @date 05.06.2020
850  * @brief Fonction qui récupère les événements auxquels un utilisateur participe
851  * @param integer $idUser
852  * @return array|null
853  * @version 1.0
854  */
855 function GetEventsParticipating(int $idUser): ?array
856 {
857     $query = <<<EX
858     SELECT idEvenement FROM inscriptions WHERE idUser = :idUser;
859     EX;
860
861     try {
```

```
862     $requestGetEventsParticipating = DatabaseController::prepare($query);
863     $requestGetEventsParticipating->bindParam(':idUser', $idUser, PDO::PARAM_INT);
864     $requestGetEventsParticipating->execute();
865
866     $result = $requestGetEventsParticipating->fetchAll(PDO::FETCH_ASSOC);
867
868     return $result;
869 } catch (PDOException $e) {
870     return null;
871 }
872 }
873
874 /**
875  * @author Hoarau Nicolas
876  * @date 05.06.2020
877  * @brief Fonction qui supprime la participation d'un utilisateur à un événement
878  * @param integer $idUser
879  * @param integer $idEvent
880  * @return boolean
881  * @version 1.0
882  */
883 function DeleteParticipation(int $idUser, int $idEvent): bool
884 {
885     $query = <<<EX
886     DELETE FROM inscriptions WHERE idEvenement = :idEvent AND idUser = :idUser;
887     EX;
888
889     try {
890         DatabaseController::beginTransaction();
891
892         $requestDeleteParticipation = DatabaseController::prepare($query);
893         $requestDeleteParticipation->bindParam(':idUser', $idUser, PDO::PARAM_INT);
894         $requestDeleteParticipation->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
895         $requestDeleteParticipation->execute();
896
897         DatabaseController::commit();
898
899         return true;
900     } catch (PDOException $e) {
901         DatabaseController::rollBack();
902         return false;
903     }
904 }
905
906 /**
907  * @author Hoarau Nicolas
908  * @date 05.06.2020
909  * @brief Fonction qui supprime l'invitation d'un utilisateur à un événement
910  * @param integer $idUser
911  * @param integer $idEvent
912  * @return boolean
913  * @version 1.0
914  */
915 function DeleteInvitation(int $idUser, int $idEvent): bool
916 {
```

```
917 $query = <<<EX
918 DELETE FROM invites WHERE idEvenement = :idUser AND idUser = :idUser;
919 EX;
920
921 try {
922     DatabaseController::beginTransaction();
923
924     $requestDeleteDeleteInvitation = DatabaseController::prepare($query);
925     $requestDeleteDeleteInvitation->bindParam(':idUser', $idUser, PDO::PARAM_INT);
926     $requestDeleteDeleteInvitation->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
927     $requestDeleteDeleteInvitation->execute();
928
929     DatabaseController::commit();
930
931     return true;
932 } catch (PDOException $e) {
933     DatabaseController::rollBack();
934     return false;
935 }
936 }
937
938 /**
939  * @author Hoarau Nicolas
940  * @date 05.06.2020
941  * @brief Fonction qui récupère les événements auxquels un utilisateur est invités
942  * @param integer $idUser
943  * @return array|null
944  * @version 1.0
945  */
946 function GetEventsInvited(int $idUser): ?array
947 {
948     $query = <<<EX
949     SELECT idEvenement FROM invites WHERE idUser = :idUser;
950     EX;
951
952     try {
953         $requestGetEventsParticipating = DatabaseController::prepare($query);
954         $requestGetEventsParticipating->bindParam(':idUser', $idUser, PDO::PARAM_INT);
955         $requestGetEventsParticipating->execute();
956
957         $result = $requestGetEventsParticipating->fetchAll(PDO::FETCH_ASSOC);
958
959         return $result;
960     } catch (PDOException $e) {
961         return null;
962     }
963 }
964
965 /**
966  * @author Hoarau Nicolas
967  * @date 05.06.2020
968  * @brief Fonction qui modifie un utilisateur donné
969  * @param string $nickname
970  * @param string $firstname
971  * @param string $email
```

```
972 * @param string $password
973 * @param string $lastname
974 * @param string $phoneNumber
975 * @param integer $idUser
976 * @param string $oldEmail
977 * @return boolean
978 * @version 1.0
979 */
980 function EditProfile(string $nickname, string $firstname, string $email, string
    $password = null, string $lastname = null, string $phoneNumber = null, int $idUser,
    string $oldEmail): bool
981 {
982     $salt = "";
983     $userPassword = null;
984
985     $query = <<<EX
986     UPDATE user
987     SET pseudo = :nickname, prenom = :firstname, email = :email
988     EX;
989
990     if ($lastname != null || $lastname != "")
991         $query .= ", nom = :lastname";
992
993     if ($phoneNumber != null || $phoneNumber != "")
994         $query .= ", telephone = :phoneNumber";
995
996     if ($password != null || $password != "") {
997         $salt = GetSalt(['userEmail' => $oldEmail]);
998         $userPassword = hash('sha256', $password . $salt);
999         $query .= ", password = :password";
1000     }
1001
1002     $query .= " WHERE idUser = :idUser;";
1003
1004     try {
1005         DatabaseController::beginTransaction();
1006
1007         $requestEditEvent = DatabaseController::prepare($query);
1008         $requestEditEvent->bindParam(':nickname', $nickname, PDO::PARAM_STR);
1009         $requestEditEvent->bindParam(':firstname', $firstname, PDO::PARAM_STR);
1010         $requestEditEvent->bindParam(':email', $email, PDO::PARAM_STR);
1011
1012         if ($lastname != null)
1013             $requestEditEvent->bindParam(':lastname', $lastname, PDO::PARAM_STR);
1014
1015         if ($phoneNumber != null)
1016             $requestEditEvent->bindParam(':phoneNumber', $phoneNumber, PDO::PARAM_STR);
1017
1018         if ($userPassword != null)
1019             $requestEditEvent->bindParam(':password', $userPassword, PDO::PARAM_STR);
1020
1021         $requestEditEvent->bindParam(':idUser', $idUser, PDO::PARAM_INT);
1022         $requestEditEvent->execute();
1023         DatabaseController::commit();
1024         return true;
    }
```

```
1025 } catch (PDOException $e) {
1026     DatabaseController::rollBack();
1027     return false;
1028 }
1029 }
1030
1031 function GetInvited(int $idEvent): ?array
1032 {
1033     $query = <<<EX
1034     SELECT u.pseudo, i.idUser
1035     FROM invites AS i
1036     JOIN user AS u ON i.idUser = u.idUser
1037     WHERE i.idEvenement = :idEvent
1038     EX;
1039
1040     try {
1041         $requestGetInvited = DatabaseController::prepare($query);
1042         $requestGetInvited->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
1043         $requestGetInvited->execute();
1044         $result = $requestGetInvited->fetchAll(PDO::FETCH_ASSOC);
1045
1046         return $result;
1047     } catch (PDOException $e) {
1048         return null;
1049     }
1050 }
1051
1052 /**
1053  * @author Hoarau Nicolas
1054  * @date 08.06.2020
1055  * @brief Fonction qui supprime un événement données
1056  * @param integer $idEvent
1057  * @return boolean
1058  */
1059 function DeleteEvent(int $idEvent): bool
1060 {
1061     $query = <<<EX
1062     DELETE FROM evenement WHERE idEvenement = :idEvent;
1063     EX;
1064
1065     try {
1066         DatabaseController::beginTransaction();
1067         $requestDeleteEvent = DatabaseController::prepare($query);
1068         $requestDeleteEvent->bindParam(':idEvent', $idEvent, PDO::PARAM_INT);
1069         $requestDeleteEvent->execute();
1070         DatabaseController::commit();
1071         return true;
1072     } catch (PDOException $e) {
1073         DatabaseController::rollBack();
1074         return false;
1075     }
1076 }
1077
```



```
<?php
//@source https://coderwall.com/p/rml5fa/nested-pdo-transactions

/**
 * This class extends native PDO one but allow nested transactions
 * by using the SQL statements `SAVEPOINT`, 'RELEASE SAVEPOINT' AND 'ROLLBACK
 * SAVEPOINT'
 */
class ExtendedPdo extends PDO
{
    /**
     * @var array Database drivers that support SAVEPOINT * statements.
     */
    protected static $_supportedDrivers = array("pgsql", "mysql");

    /**
     * @var int the current transaction depth
     */
    protected $_transactionDepth = 0;

    /**
     * Test if database driver support savepoints
     *
     * @return bool
     */
    protected function hasSavepoint()
    {
        return in_array($this->getAttribute(PDO::ATTR_DRIVER_NAME),
            self::$_supportedDrivers);
    }

    /**
     * Start transaction
     *
     * @return bool|void
     */
    public function beginTransaction()
    {
        if($this->_transactionDepth == 0 || !$this->hasSavepoint()) {
            parent::beginTransaction();
        } else {
            $this->exec("SAVEPOINT LEVEL{$this->_transactionDepth}");
        }

        $this->_transactionDepth++;
    }

    /**
     * Commit current transaction
     *
     * @return bool|void
     */
}
```



```
54 public function commit()
55 {
56     $this->_transactionDepth--;
57
58     if($this->_transactionDepth == 0 || !$this->hasSavepoint()) {
59         parent::commit();
60     } else {
61         $this->exec("RELEASE SAVEPOINT LEVEL{$this->_transactionDepth}");
62     }
63 }
64
65 /**
66  * Rollback current transaction,
67  *
68  * @throws PDOException if there is no transaction started
69  * @return bool|void
70  */
71 public function rollBack()
72 {
73
74     if ($this->_transactionDepth == 0) {
75         throw new PDOException('Rollback error : There is no transaction started');
76     }
77
78     $this->_transactionDepth--;
79
80     if($this->_transactionDepth == 0 || !$this->hasSavepoint()) {
81         parent::rollBack();
82     } else {
83         $this->exec("ROLLBACK TO SAVEPOINT LEVEL{$this->_transactionDepth}");
84     }
85 }
86
87 }
88
```

```
<?php

/**
 * @author dominique.aigroz@edu.ge.ch
 * Modify by: Hoarau Nicolas
 */
require_once dirname(dirname(__DIR__)) . '/config/config.php';
require_once __DIR__ . '/ExtendedPdo.php';

/**
 * @brief Helper class encapsulating
 *        the PDO object
 * @author dominique.aigroz@kadeo.net
 * @remark
 */
class DatabaseController
{
    private static $pdoInstance;

    /**
     * @brief Class Constructor - Create a new database connection if one doesn't exist
     *        Set to private so no-one can create a new instance via ' = new
     *        KDatabase();'
     */
    private function __construct()
    {
    }

    /**
     * @brief Like the constructor, we make __clone private so nobody can clone the
     *        instance
     */
    private function __clone()
    {
    }

    /**
     * @brief Returns DB instance or create initial connection
     * @return $objInstance;
     */
    public static function getInstance()
    {
        if (!self::$pdoInstance) {
            try {
                $dsn = DB_DBTYPE . ':host=' . DB_HOST . ';port=' . DB_PORT . ';dbname='
                . DB_DBNAME . ';charset=utf8';
                self::$pdoInstance = new ExtendedPdo($dsn, DB_USER, DB_PASS);
                self::$pdoInstance->setAttribute(ExtendedPdo::ATTR_ERRMODE,
                ExtendedPdo::ERRMODE_EXCEPTION);
            } catch (PDOException $e) {
                echo "KDatabase Error: " . $e->getMessage();
            }
        }
        return self::$pdoInstance;
    }
}
```

```
51     }
52
53     # end method
54     /**
55      * @brief Passes on any static calls to this class onto the singleton PDO instance
56      * @param $chrMethod The method to call
57      * @param $arrArguments The method's parameters
58      * @return $mix The method's return value
59      */
60     final public static function __callStatic($chrMethod, $arrArguments)
61     {
62         $pdo = self::getInstance();
63         return call_user_func_array(array($pdo, $chrMethod), $arrArguments);
64     }
65
66     # end method
67 }
68
```