

**BTS SERVICES INFORMATIQUES AUX ORGANISATIONS****SESSION 2023****Épreuve E5 - Conception et développement d'applications (option SLAM)****ANNEXE 7-1-B : Fiche descriptive de réalisation professionnelle (recto)**

<b>DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE</b>		N° réalisation : 1
Nom, prénom : VAQUIÉ Timothé		N° candidat : 02046968914
<input type="checkbox"/> Épreuve ponctuelle <input checked="" type="checkbox"/> Contrôle en cours de formation		Date : 07/04/2023
<b>Organisation support de la réalisation professionnelle</b>		
<b>Intitulé de la réalisation professionnelle</b> <b>Application Web de gestion d'un cimetière</b>		
<b>Période de réalisation :</b> Janvier – Février 2023 <b>Lieu :</b> Belvianes-et-Cavirac <b>Modalité :</b> <input checked="" type="checkbox"/> Seul(e) <input type="checkbox"/> En équipe		
<b>Compétences travaillées</b> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Concevoir et développer une solution applicative</li> <li><input checked="" type="checkbox"/> Assurer la maintenance corrective ou évolutive d'une solution applicative</li> <li><input checked="" type="checkbox"/> Gérer les données</li> </ul>		
<b>Conditions de réalisation<sup>1</sup> (ressources fournies, résultats attendus)</b> <ul style="list-style-type: none"> <li>- <b>Ressources fournies</b> : Classeurs d'informations, expression du contexte, expression des besoins.</li> <li>- <b>Résultats attendus</b> : Application web permettant le référencement et la gestion d'un cimetière pour une Mairie.</li> </ul>		
<b>Description des ressources documentaires, matérielles et logiciels utilisés<sup>2</sup></b> <ul style="list-style-type: none"> <li>- <b>Ressources documentaires</b> : Classeurs d'informations, documentations officielles de Bootstrap, PHP, JavaScript..</li> <li>- <b>Logiciels utilisés</b> : Xampp, StarUML.</li> <li>- <b>Environnement de travail</b> : Visual Studio Code</li> <li>- <b>Langages utilisés</b> : PHP, CSS, Javascript</li> <li>- <b>SGBD</b> : MySQL</li> <li>- <b>Framework</b> : Bootstrap.</li> </ul>		
<b>Modalités d'accès aux productions<sup>3</sup> et à leur documentation<sup>4</sup></b> <p>Dépôt Github : <a href="https://github.com/WallansDev/projet-1">https://github.com/WallansDev/projet-1</a></p>		

<sup>1</sup> En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

<sup>2</sup> Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

<sup>3</sup> Conformément au référentiel du BTS SIO « *Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve.*

<sup>4</sup> Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemples service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

## Sommaire

<b>I - Présentation.....</b>	<b>2</b>
1.1) Présentation de la société.....	2
1.2) Contexte.....	3
1.3) Description du contenu et des types utilisateurs.....	3
<b>II - Langages &amp; Logiciels.....</b>	<b>3</b>
2.1) Langages.....	3
2.2) Logiciels.....	3
2.3) Autres.....	3
<b>III - Base de données.....</b>	<b>4</b>
3.1) Informations pour la construction.....	4
3.2) Implémentation en ligne SQL.....	4
3.3) Schémas MERISE.....	6
3.3.1) MCD.....	6
3.3.2) MLD.....	7
3.4) Schéma UML.....	8
3.4.1) DCU.....	8
<b>IV - Architecture du projet.....</b>	<b>9</b>
4.1) Architecture générale.....	9
4.2) POO.....	9
<b>V - Programmation.....</b>	<b>10</b>
5.1) Inclusions.....	10
5.2) Fonctions.....	11
5.3) Sécurisation des mots de passe.....	11
5.4) Utilisation de Bootstrap.....	12
<b>VI - Triggers &amp; Fonctions SQL.....</b>	<b>12</b>
6.1) Triggers.....	12
6.2) Fonctions.....	15
<b>VII - L'application.....</b>	<b>17</b>
7.1) Vision globale.....	17
7.1.1) index.php.....	17
7.1.2) login.php :.....	17
7.1.3) account.php.....	18
7.1.4) logs.php :.....	18
7.1.5) plan.php :.....	19
7.1.6) concession.php :.....	19
7.1.7) concession-details.php.....	19
7.1.8) add-concession.php.....	20
7.1.9) update-concession.php.....	20
7.1.10) delete-concession.php.....	20
7.1.11) delete-concession-valid.php.....	20

7.1.12) tombecommunale.php.....	21
7.1.13) tombecommunale-details.php.....	21
7.1.14) add-tombecommunale.php.....	21
7.1.15) update-tombecommunale.php.....	21
7.1.16) delete-tombecommunale.php.....	22
7.1.17) delete-tombecommunale-valid.php.....	22
7.1.18) reposoir.php.....	22
7.1.19) reposoir-details.php.....	22
7.1.20) add-reposoir.php.....	23
7.1.21) update-reposoir.php.....	23
7.1.22) delete-reposoir.php.....	23
7.1.23) delete-reposoir-valid.php.....	23
7.1.24) mort.php.....	24
7.1.25) mort-details.php.....	24
7.1.26) add-mort.php.....	24
7.1.27) update-mort.php.....	25
7.1.28) delete-mort.php.....	25
7.1.29) delete-mort-valid.php.....	25
<b>VIII - Présentation du projet.....</b>	<b>25</b>
<b>IX - Possibilités d'ajout.....</b>	<b>26</b>

## I - Présentation

### **1.1) Présentation de la société**

La Mairie de Belvianes-et-Cavirac est une collectivité locale située dans la commune de Belvianes-et-Cavirac, une ville de l'Aude, en France. Elle est chargée de gérer les affaires municipales, de fournir des services publics locaux et de répondre aux besoins de la communauté locale.

Les missions de la Mairie de Belvianes-et-Cavirac sont variées et incluent notamment la gestion de l'état civil, l'urbanisme, la voirie, les transports, la gestion des équipements publics (parcs, bibliothèques, etc.), l'organisation des élections, la gestion du cimetière et des marchés, ainsi que la mise en place de projets de développement économique et culturel.

La Mairie de Belvianes-et-Cavirac est dirigée par un maire élu et un conseil municipal qui se réunit régulièrement pour prendre des décisions importantes pour la ville. Elle emploie également du personnel administratif et technique pour assurer le bon fonctionnement des services publics locaux.

En somme, la Mairie de Belvianes-et-Cavirac est une institution essentielle pour la vie de la communauté locale, fournissant une large gamme de services publics essentiels pour assurer le bien-être et la qualité de vie de ses habitants.

### **1.2) Contexte**

Afin de dématérialiser les documents concernant le cimetière de la commune et améliorer son administration, j'ai été missionné par le Maire de la commune, Mr Alain Chanaud, afin de réaliser une application web. La Mairie préférait faire appel à un étudiant stagiaire plutôt qu'à une entreprise professionnelle car les logiciels proposés étaient trop chers (~9000€ la licence).

### **1.3) Description du contenu et des types utilisateurs**

#### - Secrétaire :

- *Gérer les concessions.*
- *Gérer les tombes communales.*
- *Gérer les casiers du reposoir.*
- *Référencer les personnes décédées et enterrer sur la commune.*

#### - Maire :

- *Hérite des mêmes accès que 'secrétaire'.*
- *Regarder les logs.*

#### - Administrateur :

- *Tous les accès.*

## II - Langages & Logiciels

### **2.1) Langages**

- HTML 5, CSS, JavaScript, pour représenter les pages web.
- PHP 8.1 (POO), pour faire des pages web dynamiques.
- SQL, pour la base de données.

### **2.2) Logiciels**

- Visual Studio Code, IDE.
- StarUML, Construction des schémas relationnels.
- Xamp, logiciel pour mettre en place un serveur local.

### **2.3) Autres**

- Bootstrap, framework.

## III - Base de données

### 3.1) Informations pour la construction

Pour réaliser la base de données, j'ai disposé de classeurs d'informations. Ces informations m'ont permis de comprendre le fonctionnement et la gestion d'un cimetière.

- Une concession peut être une *tombe civile* ou un casier du *columbarium*<sup>1</sup>.
- Un mort est enterré dans une concession ou une tombe communale ou au reposoir.
- Celles-ci peuvent contenir un ou plusieurs morts.
- Chaque utilisateur verra ses actions journalisées.

*columbarium<sup>1</sup> : Un columbarium est un mobilier composé de cases. Il contient des urnes cinéraires renfermant les cendres des défunt, après crémation*

Table	Action	Lignes	Type	Interclassement	Taille	Perte
acheteur	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	3	InnoDB	utf8mb4_general_ci	16,0 kio	-
appartenir	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	4	InnoDB	utf8mb4_general_ci	48,0 kio	-
columbarium	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	2	InnoDB	utf8mb4_general_ci	32,0 kio	-
concession	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	6	InnoDB	utf8mb4_general_ci	32,0 kio	-
logs	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	889	InnoDB	utf8mb4_general_ci	96,0 kio	-
mort	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	5	InnoDB	utf8mb4_general_ci	64,0 kio	-
reposoir	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	1	InnoDB	utf8mb4_general_ci	32,0 kio	-
tombecivile	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	4	InnoDB	utf8mb4_general_ci	32,0 kio	-
tombecommunale	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	2	InnoDB	utf8mb4_general_ci	16,0 kio	-
users	<a href="#">Parcourir</a> <a href="#">Structure</a> <a href="#">Rechercher</a> <a href="#">Insérer</a> <a href="#">Vider</a> <a href="#">Supprimer</a>	3	InnoDB	utf8mb4_general_ci	16,0 kio	-

### 3.2) Implémentation en ligne SQL

#### - Table 'concession'

```

151 CREATE TABLE `concession` (
152     `IdConcession` int(11) NOT NULL,
153     `PrixConcession` int(11) DEFAULT NULL,
154     `PlaceDispo` int(11) DEFAULT NULL,
155     `TailleConcession` enum('individuelle','collective','familiale') DEFAULT NULL,
156     `TempsConcession` enum('Temporaire (5 ans - 15 ans)','Trentenaire (30 ans)','Cinquantenaire (50 ans)','Perpetuelle (pas de limite)') DEFAULT NULL
157 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
158

```

#### - Table 'tombecivile'

```

399 CREATE TABLE `tombecivile` (
400     `IdConcession` int(11) NOT NULL,
401     `NumeroPlan` int(11) NOT NULL
402 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
403

```

- Table 'columbarium'

```

132  CREATE TABLE `columbarium` (
133    `IdConcession` int(11) NOT NULL,
134    `IdCasier` int(11) NOT NULL
135  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

- Table 'reposeoir'

```

381  CREATE TABLE `reposeoir` (
382    `IdReposeoir` int(11) NOT NULL,
383    `PlaceDispo` int(11) NOT NULL
384  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

- Table 'tombecommunale'

```

420  CREATE TABLE `tombecommunale` (
421    `IdCommunale` int(11) NOT NULL,
422    `PlaceDispo` int(11) NOT NULL
423  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

- Table 'mort'

```

192  CREATE TABLE `mort` (
193    `IdMort` int(11) NOT NULL,
194    `NomMort` varchar(50) DEFAULT NULL,
195    `NomJeuneMort` varchar(50) DEFAULT NULL,
196    `PrenomMort` varchar(50) DEFAULT NULL,
197    `SexeMort` enum('homme','femme','n/a') DEFAULT NULL,
198    `DateNaissance` date DEFAULT NULL,
199    `DateMort` date DEFAULT NULL,
200    `DateObseques` date DEFAULT NULL,
201    `IdConcession` int(11) DEFAULT NULL,
202    `IdCommunale` int(11) DEFAULT NULL,
203    `IdReposeoir` int(11) DEFAULT NULL
204  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

- Table 'users'

```

439  CREATE TABLE `users` (
440    `idUser` int(11) NOT NULL,
441    `Username` varchar(30) DEFAULT NULL,
442    `Poste` enum('administrateur','maire','adjoint','secretaire') DEFAULT NULL,
443    `MdpHash` varchar(64) DEFAULT NULL
444  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

- Table 'logs'

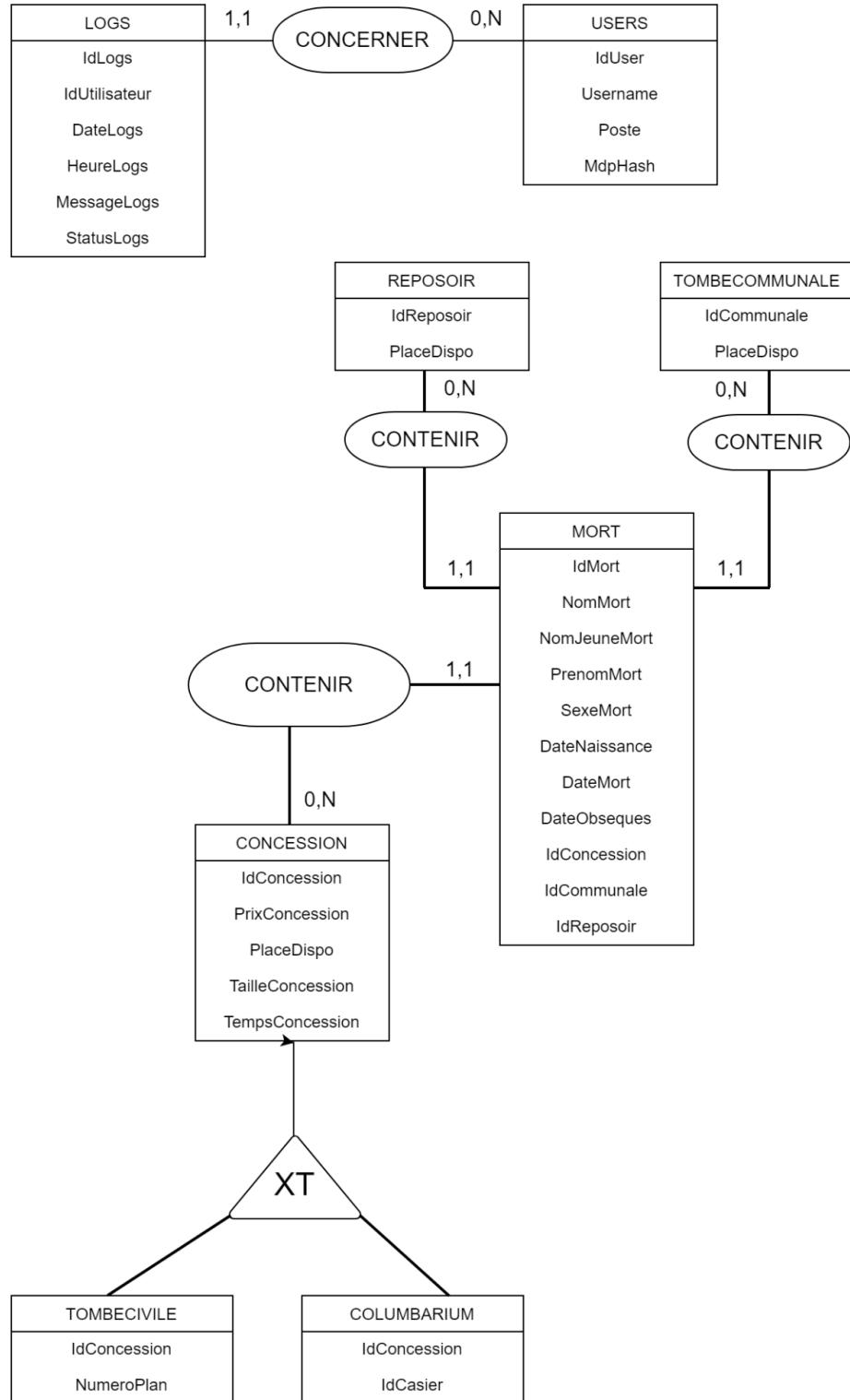
```

177  CREATE TABLE `logs` (
178    `IdLogs` int(11) NOT NULL,
179    `IdUtilisateur` int(11) NOT NULL,
180    `DateLogs` date NOT NULL,
181    `HeureLogs` varchar(8) NOT NULL,
182    `MessageLogs` varchar(500) NOT NULL,
183    `StatusLogs` varchar(7) NOT NULL
184  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

### **3.3) Schémas MERISE**

#### **3.3.1) MCD**



### **3.3.2) MLD**

**CONCESSION** : IdConcession, PrixConcession, PlaceDispo, TailleConcession, TempsConcession

**TOMBECIVILE** :#IdConcession, NumeroPlan

**COLUMBARIUM** :#IdConcession, IdCasier

**REPOSOIR** : IdCommunale, PlaceDispo

**TOMBECOMMUNAL** : IdCommunale, PlaceDispo

**MORT** : IdMort, NomMort, NomJeuneMort, PrenomMort, SexeMort, DateNaissance, DateMort, DateObseques, #IdConcession, #IdCommunale, #IdReposoir

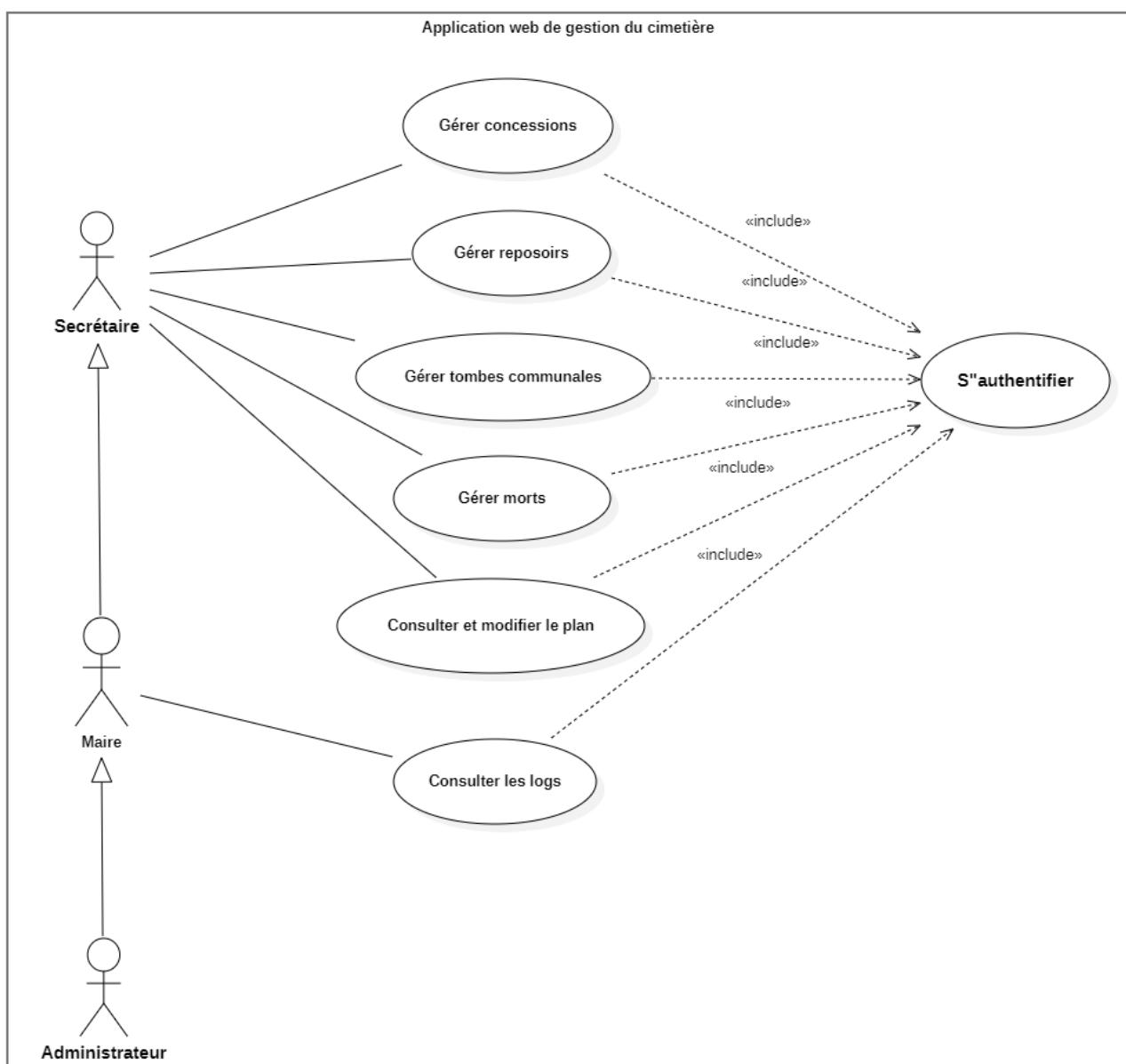
**USERS** : IdUser, Username, Poste, MdpHash

**LOGS** : IdLogs, DateLogs, HeureLogs, MessageLogs, StatusLogs, #IdUtilisateur

*Texte souligné = clé primaire | # = clé étrangère*

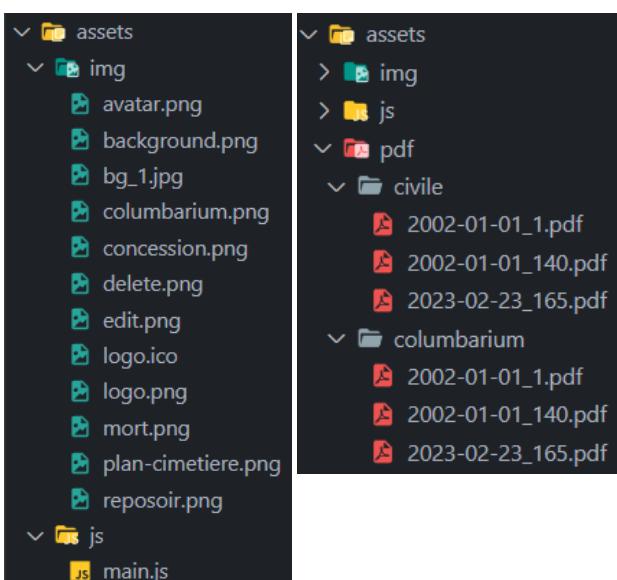
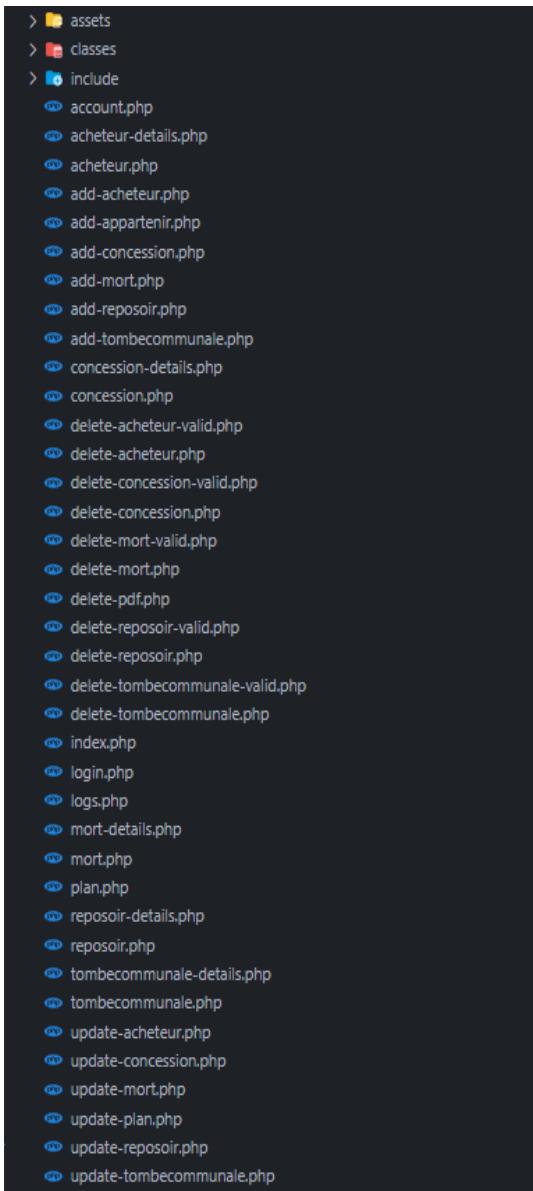
### 3.4) Schéma UML

#### 3.4.1) DCU



## IV - Architecture du projet

### 4.1) Architecture générale



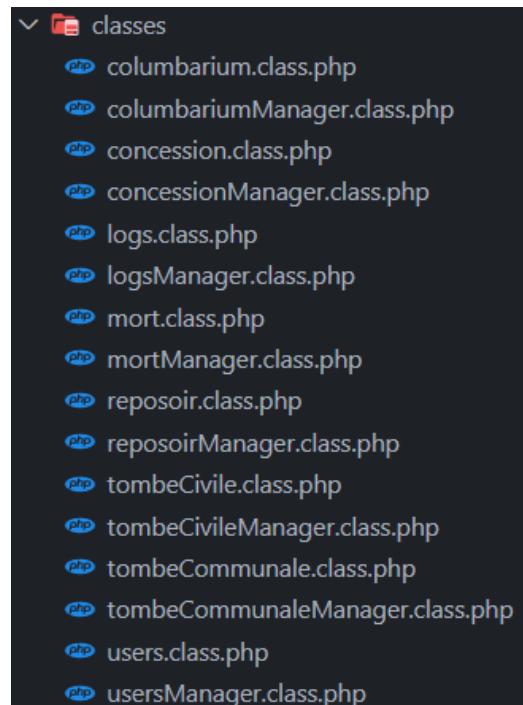
### 4.2) POO

Chaque classe correspond à une table SQL.

Le dossier classes contient tous les fichiers de classes PHP, chaque classe dispose de 2 fichiers *nom\_de\_classe.class.php* et *nom\_de\_classeManager.class.php*.

Le premier encapsule les propriétés et les méthodes nécessaires pour récupérer les données de la table.

Le manager est une classe qui est responsable de la gestion. Elle contient des méthodes pour effectuer des opérations, on y retrouve souvent le C.R.U.D<sup>1</sup>.



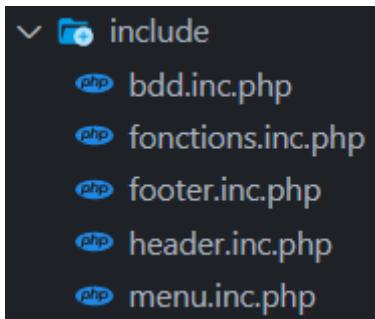
<sup>1</sup> C.R.U.D. : Create Read Update Delete, acronyme qui désigne les quatre opérations de base qui peuvent être effectuées sur une base de données

## V - Programmation

### 5.1) Inclusions

On y retrouve tous les fichiers qui vont être inclus dans d'autres fichiers .php

#### Dossier include



Il faut différencier include de require. Require interrompt l'exécution du script si le fichier inclus ne peut pas être trouvé ou a des erreurs, tandis qu'include génère une erreur et continue l'exécution du script.

Les inclusions permettent d'insérer le contenu d'un fichier dans un autre fichier.

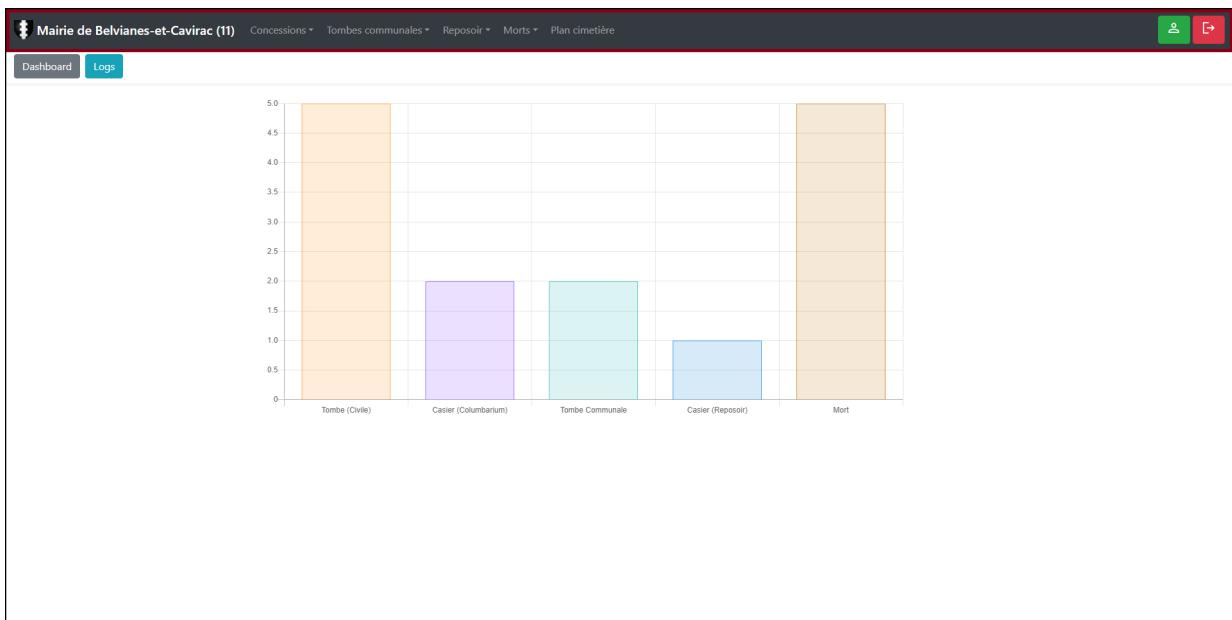
Les exemples suivants vont montrer l'impact que peut avoir l'utilisation d'une inclusion.

Il existe deux modes d'exécution pour une application, à savoir le **mode développement** et le **mode de production**.

- Le **mode développement** va afficher des erreurs à l'écran (pouvant être dangereux).
- Le **mode de production** préfère ne rien afficher et ainsi informer au minimum un utilisateur standard des erreurs potentielles.

Le choix entre ces deux modes peut être effectué dans le fichier de configuration *php.ini*. Ici, le mode est encore en développement en local.

#### Utilisation de 'header.inc.php' réussie :



## Utilisation de 'header.inc.php' ratée :

Fatal error: Uncaught Error: Failed opening required './include/headerf.inc.php' (include\_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\lgc2\account.php:14 Stack trace: #0 {main} thrown in C:\xampp\htdocs\lgc2\account.php on line 14

### 5.2) Fonctions

- `verifEntete()`, cette fonction initialise une fonction Javascript 'aff\_cach\_input', pour afficher une partie du menu si l'utilisateur dispose d'une \$\_SESSION['username'].

```

2  function verifEntete()
3  {
4      if (!empty($_SESSION['username'])) {
5          echo '<script>aff_cach_input("employe");</script>';
6      } else {
7          echo '<script>aff_cach_input("pasemploye");</script>';
8      }
9 }
```

- `alertBox()`, cette fonction permet d'afficher une alerte d'information. Celle-ci comporte en entrée une couleur (danger, info, success ... voir doc Bootstrap), un message.

Les autres arguments de la fonction, \$link et \$time, sont facultatifs et permettent de rediriger vers une page après un certain temps d'attente.

```

29  function alertBox($couleur, $message, $link, $time)
30  {
31      echo ("<div class='alert alert-$couleur' role='alert'>$message</div>");
32      if (strlen($link) > 1 && strlen($time) > 1) {
33          echo ("<script>setTimeout(function() { window.location.href = './$link';}, $time);</script>");
34      }
35 }
```

### 5.3) Sécurisation des mots de passe

D'après le R.G.P.D.<sup>1</sup>, le stockage des mots de passe en clair dans une base de données est interdit. Il est recommandé de stocker les mots de passe sous forme de hachage (hash) pour les protéger contre les accès non autorisés. De plus, il doivent respecter un certain format (min. 1 caractère spécial + 1 majuscule + 1 minuscule et au moins 8 caractères).

Cette méthode permet d'obtenir une chaîne de caractères de longueur fixe en sortie à partir d'une donnée en entrée. Il est de plus impossible de remonter jusqu'à la donnée initiale.

Les mots de passe enregistrés dans la base de données sont tous hachés avec la méthode `hash()`, en utilisant l'algorithme de hachage `SHA256`.

Pour la vérification d'une connexion on utilise la fonction `hash()`, native de PHP, où l'on rentre en entrée le nom de l'algorithme utilisé, ici `SHA256`, et le texte à hacher, ici le mot de passe. On vérifie ensuite si le texte haché est égal à celui stocké dans la base de données.

R.G.P.D.<sup>1</sup> : Règlement Général sur la Protection des Données est une réglementation européenne qui vise à renforcer la protection des données personnelles des citoyens de l'Union européenne.

## Code de connexion :

```

if ($user = $userManager->getUser($_POST['username'])) {
    if ($user->getMdpHash() == hash("sha256", $_POST['password'])) {
        session_start();
        $_SESSION['IdUser'] = $user->getIdUser();
        $_SESSION['username'] = $user->getUsername();
        $_SESSION['poste'] = $user->getPoste();
        $addLogs = new Logs(['IdUtilisateur' => $_SESSION['IdUser'], 'MessageLogs' => "s'est connecté", 'StatusLogs' => "debug"]);
        $logsManager->add($addLogs);
        header('Location: account.php');
    } else {
        echo '<script>alert("Adresse email ou mot de passe incorrect")</script>';
    }
} else {
    $_SESSION['login'] = false;
    echo '<script>alert("Adresse email ou mot de passe incorrect")</script>';
}

```

Table 'users' de la base de données :

	<input type="button" value="T"/>		IdUser	Username	Poste	MdpHash		
<input type="checkbox"/>		Éditer	<input type="button" value="Copier"/>	<input type="button" value="Supprimer"/>	1	maire	maire	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd1...
<input type="checkbox"/>		Éditer	<input type="button" value="Copier"/>	<input type="button" value="Supprimer"/>	2	admin	administrateur	9169bf3e501fea19614cacd6d646b50b63aa822bc2360a4db0...
<input type="checkbox"/>		Éditer	<input type="button" value="Copier"/>	<input type="button" value="Supprimer"/>	3	secretaire	secretaire	2cb4b1431b84ec15d35ed83bb927e27e8967d75f4bcd9cc4b2...

## 5.4) Utilisation de Bootstrap

J'ai utilisé le framework front-end Bootstrap, qui a facilité la création de l'application et son design.

### Ajout de Bootstrap :

```

1 <head>
2   <meta charset="utf-8">
3   <meta name="viewport" content="width=device-width, initial-scale=1">
4   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
5     integrity="sha384-ggOyR0iXcbMQv3Xipma34MD+dH/1fQ784/j6cYiJTQUOhcw7x9JvoRxT2M Zw1T" crossorigin="anonymous">
6
7   <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+9650z00rT7abK41JStQIAqVgRVzbzo5smXKp4YFrVH+8abTE1Pi6jizo" crossorigin="anonymous">
8   </script>
9   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-U02eT0CphQdSJQ6hJty5KVphtPhzWj9W01c1HTMGa3JDZwrnQq4sF86dIHNDz0W1"
10  crossorigin="anonymous">
11  </script>
12  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-JJSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM"
13  crossorigin="anonymous">
14  </script>
15  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js"
16  integrity="sha384-u0knCvxwv5kfmlBILk2hRnQC3Pr17a+RT6r1H17NnikvbZLhgTP00mMi466C8" crossorigin="anonymous">
17  </script>

```

## VI - Triggers & Fonctions SQL

### 6.1) Triggers

- **check\_places\_dispo**: Avant chaque insertion dans la table 'mort', on va venir vérifier si l'attribut 'PlaceDispo' est inférieur à 1. Si la proposition est vraie, il ne reste donc pas de place alors on crée une erreur. Sinon, on décrémente puis on effectue l'ajout initial.

check\_places\_dispo Éditer Exporter Supprimer BEFORE INSERT

```

1 BEGIN
2     DECLARE places_restantes INT;
3
4     IF NEW.IdReposoir IS NULL AND NEW.IdCommunale IS NULL THEN
5         BEGIN
6             SET places_restantes = PlacesRestantes_select_concession(NEW.IdConcession);
7             IF places_restantes < 1 THEN
8                 BEGIN
9                     SIGNAL SQLSTATE '20000';
10                    SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
11                END;
12            ELSE
13                BEGIN
14                    UPDATE CONCESSION SET PlaceDispo = PlaceDispo - 1 WHERE IdConcession = NEW.IdConcession;
15                END;
16            END IF;
17        END;
18
19    ELSEIF NEW.IdConcession IS NULL AND NEW.IdReposoir IS NULL THEN
20        BEGIN
21            SET places_restantes = PlacesRestantes_select_communale(NEW.IdCommunale);
22            IF places_restantes < 1 THEN
23                BEGIN
24                    SIGNAL SQLSTATE '20000';
25                    SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
26                END;
27            ELSE
28                BEGIN
29                    UPDATE TOMBECOMMUNALE SET PlaceDispo = PlaceDispo - 1 WHERE IdCommunale = NEW.IdCommunale;
30                END;
31            END IF;
32        END;
33
34    ELSE
35        BEGIN
36            SET places_restantes = PlacesRestantes_select_reposoir(NEW.IdReposoir);
37            IF places_restantes < 1 THEN
38                BEGIN
39                    SIGNAL SQLSTATE '20000';
40                    SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
41                END;
42            ELSE
43                BEGIN
44                    UPDATE REPOSOIR SET PlaceDispo = PlaceDispo - 1 WHERE IdReposoir = NEW.IdReposoir;
45                END;
46            END IF;
47        END;
48    END IF;
49 END;

```

- **delete\_place**: Après chaque suppression dans la table ‘mort’, on incrémente l’attribut ‘PlaceDispo’ selon l’identifiant ‘id’.

delete\_place Éditer Exporter Supprimer AFTER DELETE

```

1 BEGIN
2     IF OLD.IdCommunale IS NULL AND OLD.IdReposoir IS NULL THEN
3         BEGIN
4             UPDATE CONCESSION SET PlaceDispo = PlaceDispo + 1 WHERE IdConcession = OLD.IdConcession;
5         END;
6     ELSEIF OLD.IdConcession IS NULL AND OLD.IdReposoir IS NULL THEN
7         BEGIN
8             UPDATE TOMBECOMMUNALE SET PlaceDispo = PlaceDispo + 1 WHERE IdCommunale = OLD.IdCommunale;
9         END;
10    ELSE
11        BEGIN
12            UPDATE REPOSOIR SET PlaceDispo = PlaceDispo + 1 WHERE IdReposoir = OLD.IdReposoir;
13        END;
14    END IF;
15 END;

```

- **update\_delete\_place**: Après chaque modification sur la table ‘mort’, on incrémente l’attribut ‘PlaceDispo’ selon l’identifiant ‘id’ que l’on modifie.

```

1 BEGIN
2 IF OLD.IdCommunale IS NULL AND OLD.IdReposoir IS NULL THEN
3 BEGIN
4     UPDATE CONCESSION SET PlaceDispo = PlaceDispo + 1 WHERE IdConcession = OLD.IdConcession;
5 END;
6 ELSEIF OLD.IdConcession IS NULL AND OLD.IdReposoir IS NULL THEN
7 BEGIN
8     UPDATE TOMBECOMMUNALE SET PlaceDispo = PlaceDispo + 1 WHERE IdCommunale = OLD.IdCommunale;
9 END;
10 ELSE
11 BEGIN
12     UPDATE REPOSOIR SET PlaceDispo = PlaceDispo + 1 WHERE IdReposoir = OLD.IdReposoir;
13 END;
14 END IF;
15 END

```

- **update\_place**: Avant chaque modification sur la table ‘mort’, on va venir vérifier si l’attribut ‘PlaceDispo’ est inférieur à 1. Si la proposition est vraie, il ne reste donc pas de place alors on crée une erreur. Sinon, on décrémente puis on effectue la modification initiale.

```

1 BEGIN
2 DECLARE places_restantes INT;
3 IF NEW.IdReposoir IS NULL AND NEW.IdCommunale IS NULL THEN
4     IF NEW.IdConcession = OLD.IdConcession THEN
5         BEGIN
6         END;
7     ELSE
8         BEGIN
9             SET places_restantes = PlacesRestantes_select_concession(NEW.IdConcession);
10            IF places_restantes < 1 THEN
11                BEGIN
12                    SIGNAL SQLSTATE '20000'
13                    SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
14                END;
15            ELSE
16                BEGIN
17                    UPDATE CONCESSION SET PlaceDispo = PlaceDispo - 1 WHERE IdConcession = NEW.IdConcession;
18                END;
19            END IF;
20        END;
21    END IF;
22
23 ELSEIF NEW.IdConcession IS NULL AND NEW.IdReposoir IS NULL THEN
24     IF NEW.IdCommunale = OLD.IdCommunale THEN
25         BEGIN
26         END;
27     ELSE
28         BEGIN
29             SET places_restantes = PlacesRestantes_select_communale(NEW.IdCommunale);
30             IF places_restantes < 1 THEN
31                 BEGIN
32                     SIGNAL SQLSTATE '20000'
33                     SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
34                 END;
35             ELSE
36                 BEGIN
37                     UPDATE TOMBECOMMUNALE SET PlaceDispo = PlaceDispo - 1 WHERE IdCommunale = NEW.IdCommunale;
38                 END;
39             END IF;
40         END;

```

```

41    END IF;
42
43 ELSE
44     IF NEW.IdReposoir = OLD.IdReposoir THEN
45         BEGIN
46         END;
47 ELSE
48     BEGIN
49         SET places_restantes = PlacesRestantes_select_reposoir(NEW.IdReposoir);
50         IF places_restantes < 1 THEN
51             BEGIN
52                 SIGNAL SQLSTATE '20000'
53                 SET MESSAGE_TEXT = 'Il n y a pas assez de places disponibles';
54             END;
55         ELSE
56             BEGIN
57                 UPDATE REPOSOIR SET PlaceDispo = PlaceDispo - 1 WHERE IdReposoir = NEW.IdReposoir;
58             END;
59         END IF;
60     END;
61     END IF;
62 END IF;
63 END

```

## 6.2) Fonctions

- *NombreAcheteurs* : Permet de récupérer le nombre d'acheteurs enregistrés dans la base de données.

```

1 * CREATE DEFINER=`root`@`localhost` FUNCTION `NombreAcheteur`() RETURNS int(11)
2 BEGIN
3     DECLARE nombre_de_acheteur int;
4     SELECT COUNT(*) INTO nombre_de_acheteur
5     FROM ACHETEUR;
6     RETURN nombre_de_acheteur;
7 END

```

- *NombreColumbarium* : Permet de récupérer le nombre de casiers du columbarium enregistrés dans la base de données.

```

1 * CREATE DEFINER=`root`@`localhost` FUNCTION `NombreColumbarium`() RETURNS int(11)
2 BEGIN
3     DECLARE nombre_de_concession int;
4     SELECT COUNT(*) INTO nombre_de_concession
5     FROM COLUMBARIUM;
6     RETURN nombre_de_concession;
7 END

```

- **NombreConcession:** Permet de récupérer le nombre de concessions enregistrées dans la base de données

```

1 * CREATE DEFINER='root'@'localhost' FUNCTION `NombreConcession`() RETURNS int(11)
2 BEGIN
3     DECLARE nombre_de_concession int;
4     SELECT COUNT(*) INTO nombre_de_concession
5     FROM CONCESSION;
6     RETURN nombre_de_concession;
7 END

```

J'ai fait de même pour les fonctions suivantes :

- **NombreMort**
- **NombreTombeCivile**
- **NombreTombeCommunale**
- **NombreReposoir**

- **PlacesRestantes\_select\_concession** : Récupère la valeur de l'attribut 'PlaceDispo' selon un id émis en entrée pour une concession.

```

1 * CREATE DEFINER='root'@'localhost' FUNCTION `PlacesRestantes_select_concession`(`Id` INT(11)) RETURNS int(11)
2 BEGIN
3     DECLARE resultat varchar(66);
4     SELECT PlaceDispo into resultat FROM CONCESSION WHERE IdConcession = Id;
5     RETURN resultat;
6 END

```

- **PlacesRestantes\_select\_communale** : Récupère la valeur de l'attribut 'PlaceDispo' selon un id émis en entrée pour une tombe communale.

```

1 * CREATE DEFINER='root'@'localhost' FUNCTION `PlacesRestantes_select_communale`(`Id` INT(11)) RETURNS int(11)
2 BEGIN
3     DECLARE resultat varchar(66);
4     SELECT PlaceDispo into resultat FROM TOMBECOMMUNALE WHERE IdCommunale = Id;
5     RETURN resultat;
6 END

```

- **PlacesRestantes\_select\_reposoir** : Récupère la valeur de l'attribut 'PlaceDispo' selon un id émis en entrée pour un casier du reposoir.

```

1 * CREATE DEFINER='root'@'localhost' FUNCTION `PlacesRestantes_select_reposoir`(`Id` INT(11)) RETURNS int(11)
2 BEGIN
3     DECLARE resultat varchar(66);
4     SELECT PlaceDispo into resultat FROM REPOSOIR WHERE IdReposoir = Id;
5     RETURN resultat;
6 END

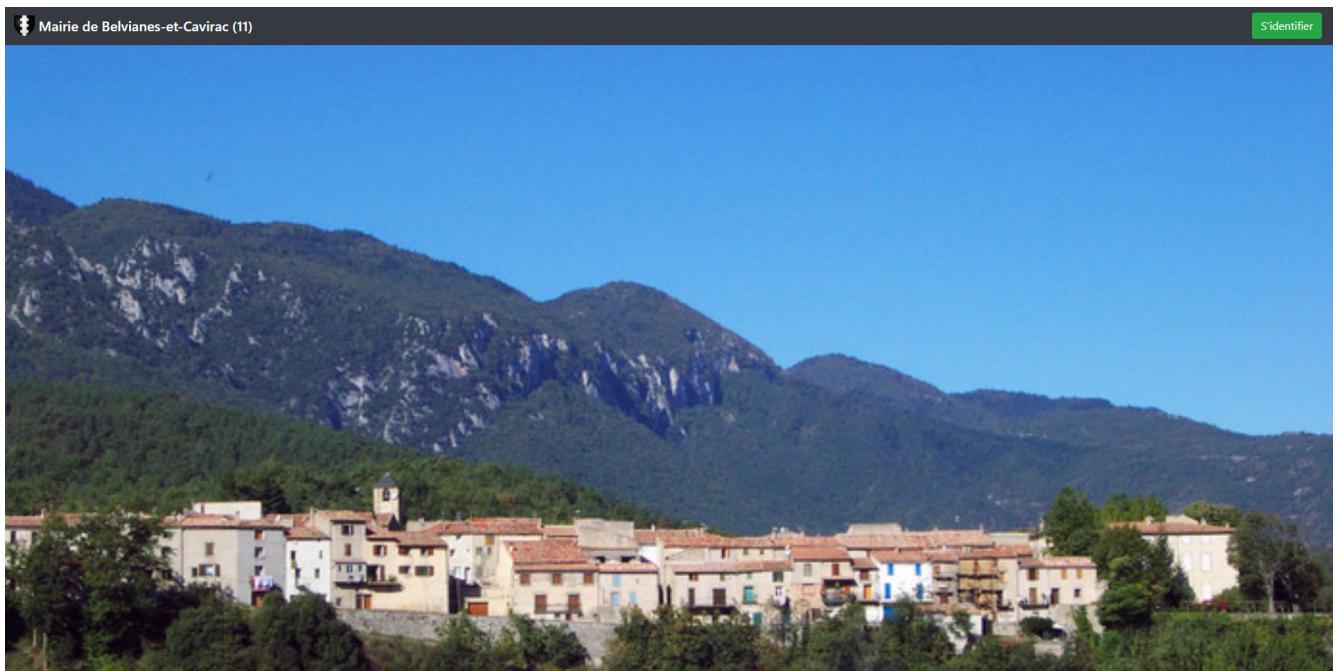
```

## VII - L'application

### 7.1) Vision globale

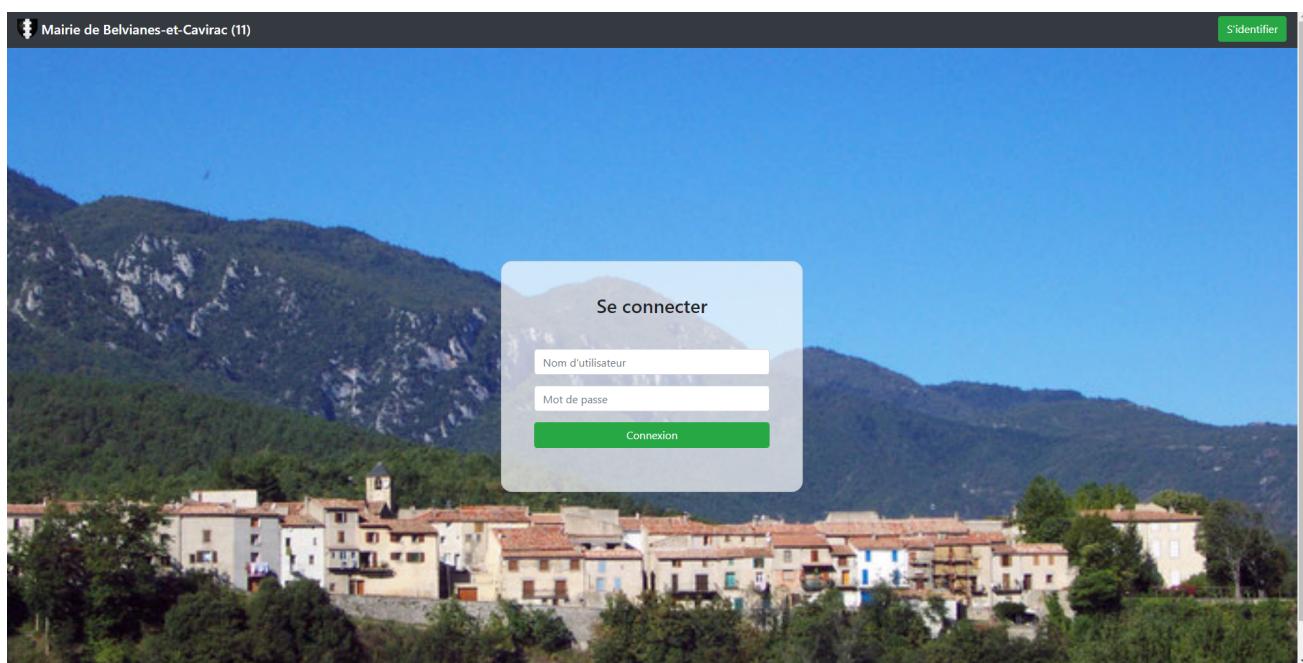
La page d'accueil simpliste avec la ville en fond, permet de se rendre sur la page de connexion.

#### 7.1.1) index.php



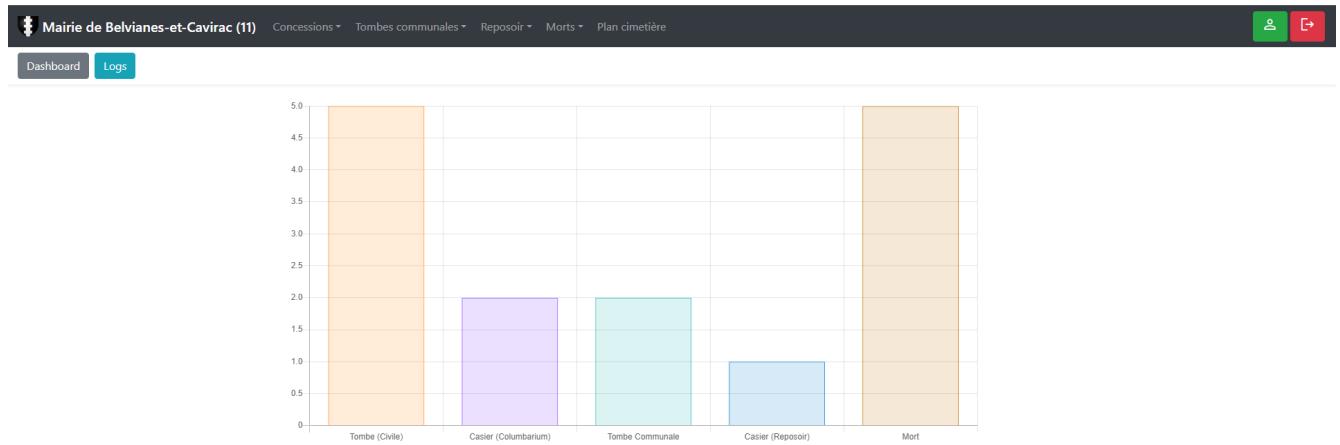
La page de connexion permet à l'utilisateur de s'authentifier pour utiliser l'application.

#### 7.1.2) login.php :



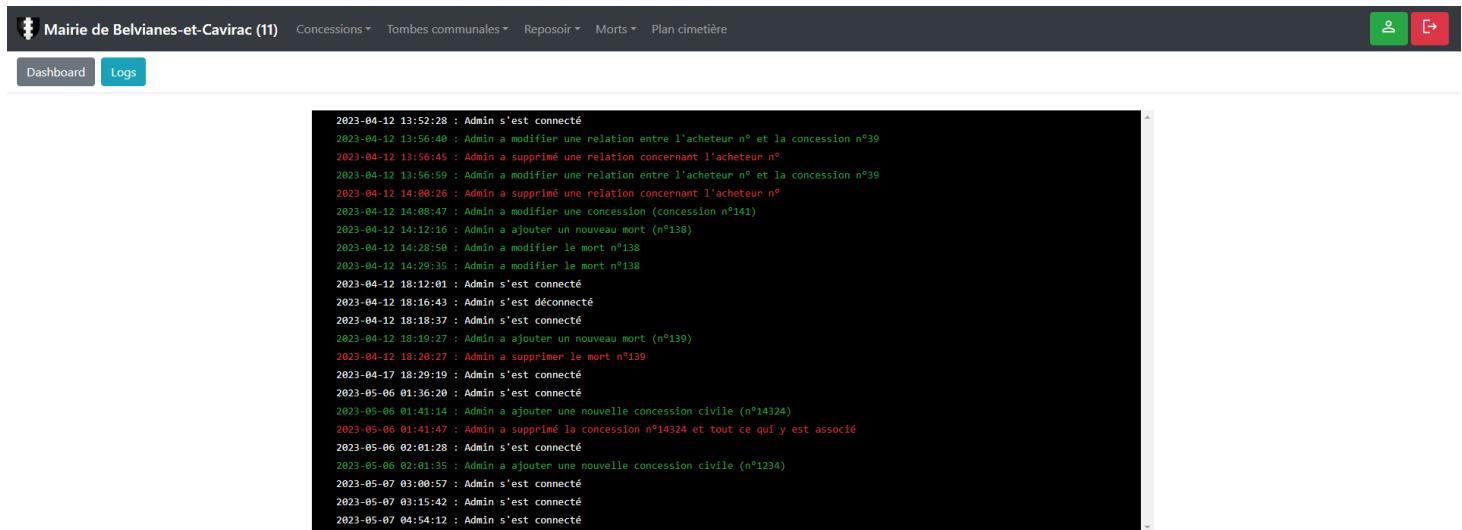
La page de compte dans un premier temps un diagramme en barre montrant le nombre des données rentré dans la base de données.

### **7.1.3) account.php**



La page de logs affiche les 23 dernières actions effectuées sur l'application. Seuls le maire et l'administrateur y ont accès en s'identifiant.

#### 7.1.4) logs.php :



Cette page affiche le plan du cimetière.

### 7.1.5) plan.php :



Page de recherche d'une concession par un formulaire.

### 7.1.6) concession.php :

Résultat de la recherche				
Numéro sur plan	Prix d'achat (F / €)	Places restantes	Taille	Temps d'acquisition
n°1	300	1	Familiale	Perpetuelle (pas de limite)

Page qui affiche toutes les informations d'une concession.

### 7.1.7) concession-details.php

Prix d'achat (F / €)	Places restantes	Taille	Temps d'acquisition	N° personnes décédées
300	1	Familiale	Perpetuelle (pas de limite)	n°131, 134, 135

Page d'ajout d'une concession.

### 7.1.8) add-concession.php

Page de modification d'une concession.

### 7.1.9) update-concession.php

Page de suppression d'une concession.

### 7.1.10) delete-concession.php

Page pour valider la suppression d'une concession.

### 7.1.11) delete-concession-valid.php

Page de recherche d'une tombe communale par un formulaire.

### 7.1.12) tombecommunale.php

Numéro plan	Places restantes
n°1	2

Page qui affiche toutes les informations d'une tombe communale.

### 7.1.13) tombecommunale-details.php

Places restantes	N° personnes décédées
2	n°

Page d'ajout d'une tombe communale.

### 7.1.14) add-tombecommunale.php

Ajout d'une tombe communale

Numéro de tombe *	Places disponibles *
-------------------	----------------------

\* Champs obligatoires

Ajouter

Page de modification d'une tombe communale.

### 7.1.15) update-tombecommunale.php

Modification de la tombe communale n°1

Places disponibles *
----------------------

\* Champs obligatoires

Ajouter

Page de suppression d'une tombe communale.

### 7.1.16) delete-tombecommunale.php

Mairie de Belvianes-et-Cavirac (11) Concessions ▾ Tombes communales ▾ Reposoir ▾ Morts ▾ Plan cimetière

Supprimer une tombe communale :

Supprimer

Page pour valider la suppression d'une tombe communale.

### 7.1.17) delete-tombecommunale-valid.php

Mairie de Belvianes-et-Cavirac (11) Concessions ▾ Tombes communales ▾ Reposoir ▾ Morts ▾ Plan cimetière

Êtes-vous sûr de supprimer la tombe communale n°1  
*Cela sera irréversible*

Supprimer
Annuler

Page de recherche d'un casier du reposoir par un formulaire.

### 7.1.18) reposoir.php

Mairie de Belvianes-et-Cavirac (11) Concessions ▾ Tombes communales ▾ Reposoir ▾ Morts ▾ Plan cimetière

Rechercher

1 résultat trouvé

Numéro de casier	Places restantes
n°1	0

Page qui affiche toutes les informations d'un casier du reposoir.

### 7.1.19) reposoir-details.php

Mairie de Belvianes-et-Cavirac (11) Concessions ▾ Tombes communales ▾ Reposoir ▾ Morts ▾ Plan cimetière

Informations casier n°1



Places restantes	N° personnes décédées
0	n°138

Page d'ajout d'un casier du reposoir.

### 7.1.20) add-reposoir.php

Page de modification d'un casier du reposoir.

### 7.1.21) update-reposoir.php

Page de suppression d'un casier du reposoir.

### 7.1.22) delete-reposoir.php

Page pour valider la suppression d'un casier du reposoir.

### 7.1.23) delete-reposoir-valid.php

Page de recherche d'une personne décédée.

### 7.1.24) mort.php

Numéro d'indentification	Nom de famille	Nom de jeune fille	Prénom	Sexe	Date de naissance	Date de décès	Date obsèques
n°131	SAVOIE	--	Norris	Homme	--	--	--

Page qui affiche toutes les informations d'une personne décédée.

### 7.1.25) mort-details.php

Date de naissance	Date de décès	Date des obsèques	Numéro de concession
--	--	--	1

Page d'ajout d'une personne décédée.

### 7.1.26) add-mort.php

Page de modification d'une personne décédée.

### 7.1.27) update-mort.php

Mairie de Belvianes-et-Cavirac (11) Concessions Tombes communales Reposoir Morts Plan cimetière

Modification du mort n°131

Nom de famille : SAVOIE      Prénom : Norris

Sexe :  Homme  Femme  Non Renseigner      Lieu d'enterrement :  Tombe civile  Columbarium  Tombe Communale  Reposoir

Choisir une concession : 1

Date Naissance : jj/mm/aaaa      Date Mort : jj/mm/aaaa      Date Obsèques : jj/mm/aaaa

\* Champs obligatoires

Modifier

Page de suppression d'une personne décédée.

### 7.1.28) delete-mort.php

Mairie de Belvianes-et-Cavirac (11) Concessions Tombes communales Reposoir Morts Plan cimetière

Supprimer un mort :

Entrer le numéro du mort à supprimer

Supprimer

Page pour valider la suppression d'une personne décédée.

### 7.1.29) delete-mort-valid.php

Mairie de Belvianes-et-Cavirac (11) Concessions Tombes communales Reposoir Morts Plan cimetière

Êtes-vous sûr de supprimer le mort n°131 ?  
Cela sera irréversible

Supprimer Annuler

## VIII - Présentation du projet

J'ai eu l'honneur de présenter ce projet à la Mairie à l'occasion d'un conseil municipal qui s'est tenu le Mercredi 12 Avril 2023. Durant ce conseil, j'ai présenté un powerpoint montrant les différents points positifs qu'apporte l'application web.

J'ai eu l'opportunité d'interagir avec les administrés, avec qui j'ai pu partager mes connaissances techniques et professionnelles. Cela a été une expérience enrichissante pour moi et m'a permis d'améliorer mes compétences et de mieux comprendre les besoins et les attentes des utilisateurs finaux de l'application que j'ai développée.

## IX - Possibilités d'ajout

Il serait possible dans l'avenir d'ajouter différentes fonctionnalités afin d'améliorer l'application :

- Possibilité pour les comptes ayant des droits de réinitialiser les mots de passe des utilisateurs.
- Créer de nouveaux utilisateurs
- Pouvoir stocker les informations de personnes ayant acheté une concession.