La ligue 1&2 de

Football

Site web en PHP

OLLIVIER CHRIS

2022

# Page de service

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Auteur** | **Date** | **Description** |
| 1,0 | OLLIVIER Chris | 18/02/2022 |  |
|  |  |  |  |

Table des matiÈres

[Page de service 1](#_Toc100143205)

[Objet du document 2](#_Toc100143206)

[Nature du Site 2](#_Toc100143207)

[MISE EN PLACE 2](#_Toc100143208)

[La base de données PostgreSQL 3](#_Toc100143209)

[Class PHP 4](#_Toc100143210)

[Architecture 5](#_Toc100143211)

[Explication du code 6](#_Toc100143212)

[Affichage 12](#_Toc100143213)

# Objet du document

Ce document présente création et l’évolution d’un site en PHP sur le thème du foot et plus particulièrement de la Ligue 1 et 2. Dans le cadre de notre cours de gestion des bases de données et des sites web avec monsieur Gibert.

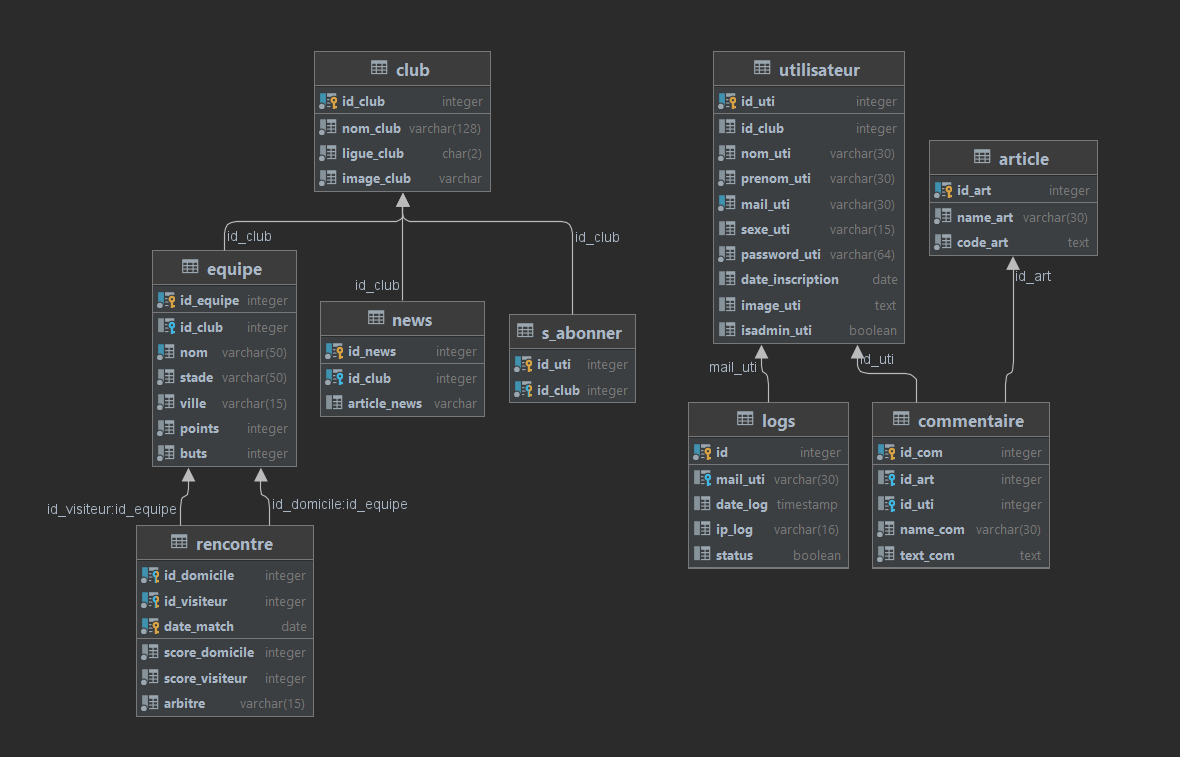
# Nature du Site

Ce site web est un site de fan de la Ligue de football. Il doit donc permettre à un utilisateur de créer un compte ou de se connecter s’il en possède déjà un. L’utilisateur a donc accès à plusieurs fonctionnalités comme lire des articles sur le foot et ajouter des commentaires à ces articles, retrouvez des informations sur les équipes qui jouent la Ligue 1 et 2. S’abonner à la Newsletter d’un des clubs de foot.

## MISE EN PLACE

Pour la mise en place de ce site web, nous avons utilisé des technologies comme PHP et PostgreSQL. La première étape était de construire le script SQL de création pour la base de données avec les contraintes. La 2de tâche a été de créer là l’inscription et la connexion en PHP, puis j’ai ajouté la possibilité à l’utilisateur de changer ses informations personnelles. J’ai pour finir par créer la partie article, commentaire et la visualisation des informations. Des informations des clubs.

# La base de données PostgreSQL

La base de données est modélisée sous la forme d’un diagramme de modèle d'objet de base de données en UML, illustre ci-dessous.

Toutes ces tables sont créées avec des contraintes pour éviter les errer dure la production du site et pour plus de sécurité.

Exemple 1 : On retrouve ci-dessous des contraintes de clef primaire et de clef étrangère ou même des valeurs pas défaut

create table équipe(  
 id\_equipe intègre not null constraint pk\_equipe primary key,  
 id\_club integer constraint fk\_club references club,  
 nom varchar(50) not null constraint uk\_nom unique,  
 stade varchar(50) not null,  
 ville varchar(15) not null,  
 points integer default 0 not null,  
 buts integer default 0 not null  
);

Exemple 2 : On y retrouve une contrainte de clef primaire qui est la concaténation de l’id du domicile, d’id du visiteur et de la date du match.

create table rencontre(  
 id\_domicile integer not null constraint fk\_id\_domicile references equipe on delete cascade,  
 id\_visiteur integer not null constraint fk\_id\_visiteur references equipe on delete cascade,  
 date\_match date not null,  
 score\_domicile integer not null,  
 score\_visiteur integer not null,  
 arbitre varchar(15) not null,  
 constraint pk\_rencontre primary key (id\_domicile, id\_visiteur, date\_match)  
);

# Class PHP

Ensuite, j’ai donc réalisé les classes PHP me permettant de communiquer avec ma base de données qui sont modélisées ci-dessous avec un digramme de classe en UML

Une image contenant texte, moniteur, capture d’écran, noir

Description générée automatiquement

Ce diagramme modéliser donc diffèrent classes ayant différentes utilités. On retrouve la class DotEnv qui permet de récupère des variables d’environnement dans un fichier en .env . Ensuite on remarque la class mère Dbh pour Data Base Host qui va elle crée la connexion à la base de données. Elle va donc être héritée sur les classes suivent : Article, Commentaire et Club qui sont des Object de la base de données, mais on y retrouver aussi des classes comme Login, Update, Signup qui sont eux des classes pour gère les utilisateurs. Toutes ces classes sont-elles même des classes mères de leur contrôleur.

## Architecture

Une image contenant texte

Description générée automatiquementLe fonctionnement de cette architecture respecte le modèle MVC (Model View Controller) on y retrouve donc nos classes model pour toutes les classes, mais un contrôleur uniquement pour les classes qui ont besoin de retour d’erreur comme Commentaire Login Signup et Update. Enfin on y retrouve nos view qui sont les ficher include qui vont eux intensifiez nos classes est appelé les méthodes.

# Explication du code

La class Dbh a uniquement une méthode

protected function connect(): PDO|null {  
 $dotEnv = new DotEnv(*\_\_DIR\_\_* . '../../.env');  
 $dotEnv->load();  
 try {  
 $dsn = getenv('DATABASE\_DNS');  
 $user = getenv('DATABASE\_USER');  
 $password = getenv('DATABASE\_PASSWORD');  
  
 $dbh = new PDO($dsn, $user, $password);  
 return $dbh;  
  
 }catch (PDOException $exception){  
 print 'Error :' . $exception->getMessage() .' <br>';  
 die();  
 }  
}

Cette méthode est protected se qu’il veut dire que l’on peut l’utilisera que si on hérite de cette class. Elle permet la connexion à la base de données grâce aux identifiants dans le fichier .env et retourne un Object PDO.

La class commentaire a deux méthodes :

public function getCommentaireData($idArt): bool|array {  
 $stmt = $this->connect()->query("SELECT \* FROM commentaire WHERE id\_art = $idArt");  
 return $stmt->fetchAll(PDO::*FETCH\_ASSOC*);  
}

Cette méthode est public, car on ne l’utilisera pas dans le contrôleur, elle permet de récupère un article a partie de son id, elle retourne un Array Associative .

protected function setCommentaire($idArt,$comName,$comText){  
 $stmt = $this->connect()->prepare('INSERT INTO COMMENTAIRE (id\_art,name\_com,text\_com) values (?,?,?)');  
 if (!$stmt->execute(array($idArt,$comName,$comText))){  
 $stmt = null;  
 header("location: ../frontend/php/index.php#popup-one?error=stmtFailed");  
 exit();  
 }  
 $stmt = null;  
}

Cette méthode est protected, car le contrôleur hérite de set class, elle permet d’ajouter un nouveau commentaire avec son id, nom et texte.

La class Club contient deux méthodes

public function getClubData*()*: bool|array *{* $stmt = $this->connect*()*->query*(*"SELECT \* FROM club"*)*;  
 return $stmt->fetchAll*(*PDO::*FETCH\_ASSOC)*;  
*}*

Cette méthode est public, car elle ne pas utiliser par le contrôleur. Elle permet de récupère toutes les clubs pour les affiches elle retour donc un array.

public function getClub*(*$id\_club*)* : bool|array *{* $stmt = $this->connect*()*->prepare*(*"SELECT \* FROM club where id\_club = ?"*)*;  
  
 if *(*!$stmt->execute*(*array*(*$id\_club*))) {* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php"*)*;  
 exit*()*;  
 *}* return $stmt->fetchAll*(*PDO::*FETCH\_ASSOC)*;  
*}*

Cette méthode est public, car elle ne pas utiliser par le contrôleur. Elle permet de récupère un club dans la base de donne en passent en paramètre l’id du club. Elle retourne un club dans un array.

La class article a une méthode

public function getArticleData*()*: bool|array *{* $stmt = $this->connect*()*->query*(*"SELECT \* FROM article"*)*;  
 return $stmt->fetchAll*(*PDO::*FETCH\_ASSOC)*;  
*}*

Elle permet de récupère toutes les articles de la base de données dans un array.

La class Signup comporte troie méthode.

private function encryptePass*(*$password*)*: string*{* $salt = "MySalt";  
 return crypt*(*$password,$salt*)*;  
*}*

Cette méthode est private car elle est utilisée dans la class uniquement. Elle permet de transformer un mot de passe en hash.

protected function setUser*(*$firstname,$lastname,$password,$mail,$date*){* $stmt = $this->connect*()*->prepare*(*'INSERT INTO UTILISATEUR (nom\_uti,prenom\_uti,mail\_uti,password\_uti,date\_inscription) values (?,?,?,?,?)'*)*;  
 $cryptPassWord = $this->encryptePass*(*$password*)*;  
  
 if *(*!$stmt->execute*(*array*(*$firstname,$lastname,$mail,$cryptPassWord,$date*))){* $stmt = null;  
 header*(*"location: ../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* $stmt = null;  
*}*

Cette méthode est protected se qu’il veut dire que l’on peut l’utilisera que si on hérite de cette class. Elle permet de créer un nouvel utilisateur en base. Elle prend toute le donne nessaicer pour en ajouter un.

protected function checkUser*(*$mail*){* $stmt = $this->connect*()*->prepare*(*'SELECT mail\_uti FROM utilisateur WHERE mail\_uti = ? '*)*;  
  
 if *(*!$stmt->execute*(*array*(*$mail*))){* $stmt = null;  
 header*(*"location: ../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* if *(*$stmt->rowCount*()* > 0*){* $resCheck =false;  
 *}*else*{* $resCheck =true;  
 *}* return $resCheck;  
*}*

Cette méthode permet d’assuré que le mail de l’utilisateur crée ne pas déjà pris. Elle prend un mail et retourne un booléen.

La class Update comporte troie méthode.

protected function updateUser*(*$firstname,$lastname,$sex,$idClub,$id*){* $stmt = $this->connect*()*->prepare*(*'UPDATE UTILISATEUR SET nom\_uti = ?, prenom\_uti = ?, sexe\_uti = ?, id\_club = ? WHERE id\_uti = ?'*)*;  
  
 if *(*!$stmt->execute*(*array*(*$firstname,$lastname,$sex,$idClub,$id*))){* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* $stmt = null;  
*}*

Cette méthode permet de modifier un utilisateur, elle ne retourne rien.

protected function updateUserImg*(*$file,$id*){* $stmt = $this->connect*()*->prepare*(*'UPDATE UTILISATEUR SET image\_uti = ? WHERE id\_uti = ?'*)*;  
  
 if *(*!$stmt->execute*(*array*(*$file,$id*))){* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* $stmt = null;  
*}*

Cette méthode permet de modifier l’image d’un utilisateur, elle ne retourne rien.

protected function updataNews*(*$club,$id*) {* $stmt = $this->connect*()*->prepare*(*'INSERT INTO s\_abonner (id\_uti,id\_club) VALUES (?,?)'*)*;  
  
 if *(*!$stmt->execute*(*array*(*$id,$club*))){* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* $stmt = null;  
*}*

Cette méthode permet de modifier la newsletter d’un utilisateur, elle ne retourne rien.

La class Login comporte 5 méthodes.

private function encryptPass*(*$password*)*: string*{* $salt = "MySalt";  
 return crypt*(*$password,$salt*)*;  
*}*

Cette méthode est private car elle est utilisée dans la class uniquement. Elle permet de transformer un mot de passe en hash.

private function passWordVerify*(*$pwd,$hashedPwd*)*: bool*{* if *(*$this->encryptPass*(*$pwd*)* == $hashedPwd*){* $res = true;  
 *}*else*{* $res =false;  
 *}* return $res;  
*}*

Cette méthode permet de vérifier que le hash en base de donne d’un mot de passe est bien le même que celui fourni.

private function addlogs *(*$mail,$ip,$status*){* $stmt = $this->connect*()*->prepare*(*'INSERT INTO LOGS (mail\_uti,ip\_log,status,date\_log) values (?,?,?,now())'*)*;  
  
 if *(*!$stmt->execute*(*array*(*$mail,$ip,$status*))){* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-one?error=stmtFailed"*)*;  
 exit*()*;  
 *}* $stmt = null;  
*}*

Cette méthode permet de créer une nouvelle logs en base de données. Elle ne retourne rien.

private function getSessionInfo*(*$mail*){* $stmt = $this->connect*()*->prepare*(*"SELECT \* FROM utilisateur WHERE mail\_uti = ?"*)*;  
  
 if *(*!$stmt->execute*(*array*(*$mail*))) {* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-two"*)*;  
 exit*()*;  
 *}* $allData = $stmt->fetchAll*(*PDO::*FETCH\_ASSOC)*;  
 $\_SESSION*[*'id'*]* = $allData*[*0*][*'id\_uti'*]*;  
 $\_SESSION*[*'idClub'*]* = $allData*[*0*][*'id\_club'*]*;  
 $\_SESSION*[*'nom'*]* = $allData*[*0*][*'nom\_uti'*]*;  
 $\_SESSION*[*'prenom'*]* = $allData*[*0*][*'prenom\_uti'*]*;  
 $\_SESSION*[*'mail'*]* = $allData*[*0*][*'mail\_uti'*]*;  
 $\_SESSION*[*'sex'*]* = $allData*[*0*][*'sexe\_uti'*]*;  
 $\_SESSION*[*'image'*]* = $allData*[*0*][*'image\_uti'*]*;  
 $\_SESSION*[*'isAdmin'*]* = $allData*[*0*][*'isadmin\_uti'*]*;  
*}*

Elle permet de récupère les données l’un utilisateur est de les fournir en variable de session

protected function getUser*(*$password, $mail,$ip*)  
{* $stmt = $this->connect*()*->prepare*(*"SELECT password\_uti FROM utilisateur WHERE mail\_uti = ?"*)*;  
  
 if *(*!$stmt->execute*(*array*(*$mail*))) {* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-two?error=stmtFailed"*)*;  
 exit*()*;  
 *}* if *(*$stmt->rowCount*()* == 0*){* $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-two?error=notfound"*)*;  
 exit*()*;  
 *}* $pwdHashed = $stmt->fetchAll*(*PDO::*FETCH\_ASSOC)*;  
 $checkPwd = $this->passWordVerify*(*$password,$pwdHashed*[*0*][*"password\_uti"*])*;  
  
 if *(*$checkPwd == false*){* $this->addlogs*(*$mail,$ip,'false'*)*;  
 $stmt = null;  
 header*(*"location: ../../frontend/php/index.php#popup-two?error=notfound"*)*;  
 exit*()*;  
 *}* else *{* session\_start*()*;  
 $this->addlogs*(*$mail,$ip,'true'*)*;  
 $this->getSessionInfo*(*$mail*)*;  
 *}* $stmt = null;  
*}*

Cette méthode permet de se connecte avec un utilisateur en base de données.

Toutes les classes contrôleurs permet d’ajouter des contrôles sur les données fournies par l’utilisateur.

# Affichage

Text

Description automatically generatedPour l’affichage du HTML, j’utilise des fichiers PHP classiques. Le site pour le frontend ne contient que 9 fichiers on peut les voir ci-dessous .

A screenshot of a computer screen

Description automatically generated with medium confidenceLes fichiers suivent sont séparé en 2 catégorie les Templates et les pages HTML classiques. Dans les fichiers suivent-on retrouver les fonctionnalités diffèrent comme :

* La NavBar qui est inclue dans toutes les pages de la même façon pour le Footer ces pages sont utilisées pour la navigation dans le site, la NavBar permet aussi l’inscription et la connexion.

Bar de navigation 🡪

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Création d’un compte 🡪

* Connexion à un compte

Graphical user interface

Description automatically generated with medium confidence

* La page Accourt qui permet à un utilisateur de voir toutes ses informations personnelles est de les modifier.

La page liste club permet d’affiche les diffèrent clubs dans la base de données en cliquent sur un club un popup s’affiche avec la description du club.

A picture containing shape

Description automatically generatedA picture containing graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generatedLa page article permet de visualiser les articles dans la base un utilisateur peut commentais un article

A picture containing graphical user interface

Description automatically generatedOn peut visualiser tous les commentaires à la fin de chaque article.

# Conclusion

Ce projet m’aura appris à utiliser le modèle mvc et la programmation-objet en PHP. Il aura permis aussi d’améliorer mes compétences en gestion de bases de données. Et d’acquérir de nouvelles compétences en termes de sécurité, que ce soit en PHP ou dans le milieu de la base de données.

Ce projet a pour ambition d’être continué avec l’ajout d’une API de foot et de géolocalisation.