Site PHP ligue1

OBJET DU DOCUMENT

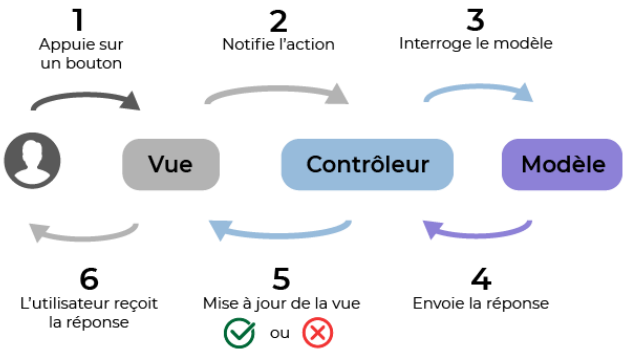
Ce document décrit l'implémentation PHP du projet de site internet Ligue1.
Ce projet vise à produire un site internet de football qui permet aux utilisateurs de s'inscrire et de lire des articles.

TABLE DES MATIERES

1 ARCHITECTURE, PRINCIPE DE FONCTIONNEMENT DU MODELE MVC

Conformément à l'analyse conception, la réalisation du site est structurée en troiscouches selon une structure qui ressemble à un design-pattern MVC (Model View Controler).

Le modèle MVC fonctionne selon le principe illustré par le schéma ci-dessous[1] :



Dans un site internet en PHP, la séparation des responsabilités entre le modèle, la vue et le contrôleur suit le principe du Modèle-Vue-Contrôleur (MVC). Voici comment ces composants sont généralement utilisés dans un site PHP :

Modèle :

Le modèle représente la logique métier et les données de l'application.
Il est responsable de l'accès et de la manipulation des données, généralement à l'aide de requêtes SQL vers une base de données.
Il définit les classes, les fonctions et les méthodes nécessaires pour interagir avec les données.
Il peut inclure des fichiers de configuration, des classes de connexion à la base de données et des classes d'accès aux données.
Il est indépendant de la présentation et ne contient pas de code HTML ou CSS.
Vue :

La vue représente l'interface utilisateur du site.
Elle est responsable de la présentation des données et de l'interaction avec l'utilisateur.
Elle contient généralement du code HTML, des balises PHP (pour afficher les données dynamiques) et du code CSS pour la mise en page.
Elle communique avec le contrôleur pour récupérer les données nécessaires à l'affichage.
Elle peut inclure des fichiers d'en-tête, de pied de page ou des modèles réutilisables pour une meilleure organisation.
Contrôleur :

Le contrôleur gère le flux de contrôle de l'application.
Il reçoit les requêtes HTTP de l'utilisateur (par exemple, en utilisant les méthodes GET ou POST).
Il interprète les actions de l'utilisateur et détermine les données et les vues nécessaires pour y répondre.
Il communique avec le modèle pour récupérer ou mettre à jour les données nécessaires.
Il sélectionne la vue appropriée pour afficher les données et les renvoie au navigateur de l'utilisateur.
Dans un site PHP, le contrôleur est généralement responsable de router les requêtes vers les bonnes actions ou méthodes du modèle et de la vue. Il peut utiliser des frameworks MVC existants tels que Laravel, Symfony ou CodeIgniter, qui fournissent une structure et des fonctionnalités supplémentaires pour faciliter le développement.

La séparation des responsabilités entre le modèle, la vue et le contrôleur dans un site PHP permet une meilleure organisation du code, une réutilisation plus facile, une maintenance simplifiée et une meilleure collaboration entre les développeurs travaillant sur différents aspects du site.

Chaque classe située dans le modèle possède une classe Manager qui regroupe toutes les Fonctions dont on a besoin pour celle-ci. Exemple pour la classe Article où l'on retrouve le constructeur les getters&setters :

```
class Article

{

   private int $id_article;

   private string $titre_article;
```

```php
private string $desc_article;

public function __construct(int $id_article, string $titre_article, string $desc_article)
{
    $this->id_article = $id_article;
    $this->titre_article = $titre_article;
    $this->desc_article = $desc_article;
}

/**
 * Get the value of id_article
 */
public function getId_article()
{
    return $this->id_article;
}

/**
 * Set the value of id_article
 *
 * @return  self
 */
public function setId_article($id_article)
{
    $this->id_article = $id_article;

    return $this;
}

/**
 * Get the value of titre_article
 */
public function getTitre_article()
{
    return $this->titre_article;
```

```
    }

    /**
     * Set the value of titre_article
     *
     * @return  self
     */
    public function setTitre_article($titre_article)
    {
        $this->titre_article = $titre_article;

        return $this;
    }

    /**
     * Get the value of desc_article
     */
    public function getDesc_article()
    {
        return $this->desc_article;
    }

    /**
     * Set the value of desc_article
     *
     * @return  self
     */
    public function setDesc_article($desc_article)
    {
        $this->desc_article = $desc_article;

        return $this;
    }
}
```

Il y a dans le modèle une classe ArticleManager qui regroupe toutes les fonctions pour communiquer avec la base de données :

```php
class ArticleManager
{

    private PDO $cnx;

    public function __construct(PDO $cnx)
    {
        $this->cnx = $cnx;
    }

    function getAllArticle(): array
    {
        $res = $this->cnx->query("select * from Article");
        $tableResult = [];
        while ($ligne = $res->fetch()) {
            $tableResult[] = new Article($ligne[0], $ligne[1], $ligne[2]);
        }

        return $tableResult;
    }

    function getArticleById(int $idArticle): Article
    {
        $res = $this->cnx->query("select * from Article where id_Article =" . $idArticle);
        $ligne = $res->fetch();
        $article = new Article($ligne[0], $ligne[1], $ligne[2]);

        return $article;
    }

    function getAllArticleByTitle(): array
    {
        $res = $this->cnx->query("select * from Article order by titre_article");
```

```
    $tableResult = [];
    while ($ligne = $res->fetch()) {
        $tableResult[] = new Article($ligne[0], $ligne[1], $ligne[2]);
    }

    return $tableResult;
}

function getAllArticleByInsertion(): array
{
    $res = $this->cnx->query("select * from Article order by id_article DESC");
    $tableResult = [];
    while ($ligne = $res->fetch()) {
        $tableResult[] = new Article($ligne[0], $ligne[1], $ligne[2],);
    }

    return $tableResult;
}


function insertArticle(Article $article)
{
    $this->cnx->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $statement = "insert into article values (default, ?, ?, ?)";

    $req_prepare = $this->cnx->prepare($statement);

    $titre = $article->getTitre_article();
    $desc = $article->getDesc_article();

    $req_prepare->bindParam(1, $titre);
    $req_prepare->bindParam(2, $desc);
    $req_prepare->bindParam(3, $auteur);

    $req_prepare->execute();
}
```

```
}
```

De plus, nous avons une page Index.php qui est considérée comme la page d'accueil ou la page principale du site. Elle joue un rôle central et est chargée en premier lorsqu'on accède à un site. Son rôle principal est de servir de point d'entrée et de coordonner l'affichage du contenu dynamique et des fonctionnalités du site.

Voici quelques principales utilisations et fonctions d'une page index.php :

Point d'entrée : La page index.php sert de point d'entrée principal pour le site web. Lorsque vous entrez l'URL du site dans votre navigateur, c'est cette page qui est chargée en premier.

Routage : La page index.php peut être utilisée pour gérer le routage des URL. Elle peut analyser l'URL demandée et déterminer quelle action ou quel contenu doit être affiché en fonction des paramètres de l'URL.

Affichage du contenu dynamique : La page index.php est souvent utilisée pour afficher le contenu dynamique du site. Elle peut interagir avec le modèle (la logique métier et les données) pour récupérer les informations nécessaires et les afficher dans la vue (l'interface utilisateur).

Inclusion d'autres fichiers : La page index.php peut inclure d'autres fichiers PHP ou ressources (comme des fichiers CSS ou JavaScript) qui sont nécessaires au fonctionnement du site. Par exemple, elle peut inclure un fichier de configuration, des bibliothèques, des modèles de mise en page ou des scripts communs.

Gestion des sessions et des utilisateurs : La page index.php peut gérer l'authentification des utilisateurs, le suivi des sessions et la gestion des autorisations. Elle peut vérifier si un utilisateur est connecté, rediriger vers une page de connexion si nécessaire, et afficher un contenu personnalisé en fonction de l'utilisateur connecté.

Gestion des erreurs : La page index.php peut également être utilisée pour la gestion des erreurs. Elle peut rediriger l'utilisateur vers une page d'erreur personnalisée si une erreur se produit lors du chargement ou du traitement des données.

Index.php :

```
<Body>
    <?php


  if (isset($_GET['Page'])) {
     $page = $_GET['Page'];
  } else {
     $page = 'Accueil';
  }
  switch ($page) {
     case "Accueil":
        include("./view/accueil.php");
        break;
     case "Page1":
        include("./view/page1.php");
        break;
     case "Page2":
        include("./view/page2.php");
```

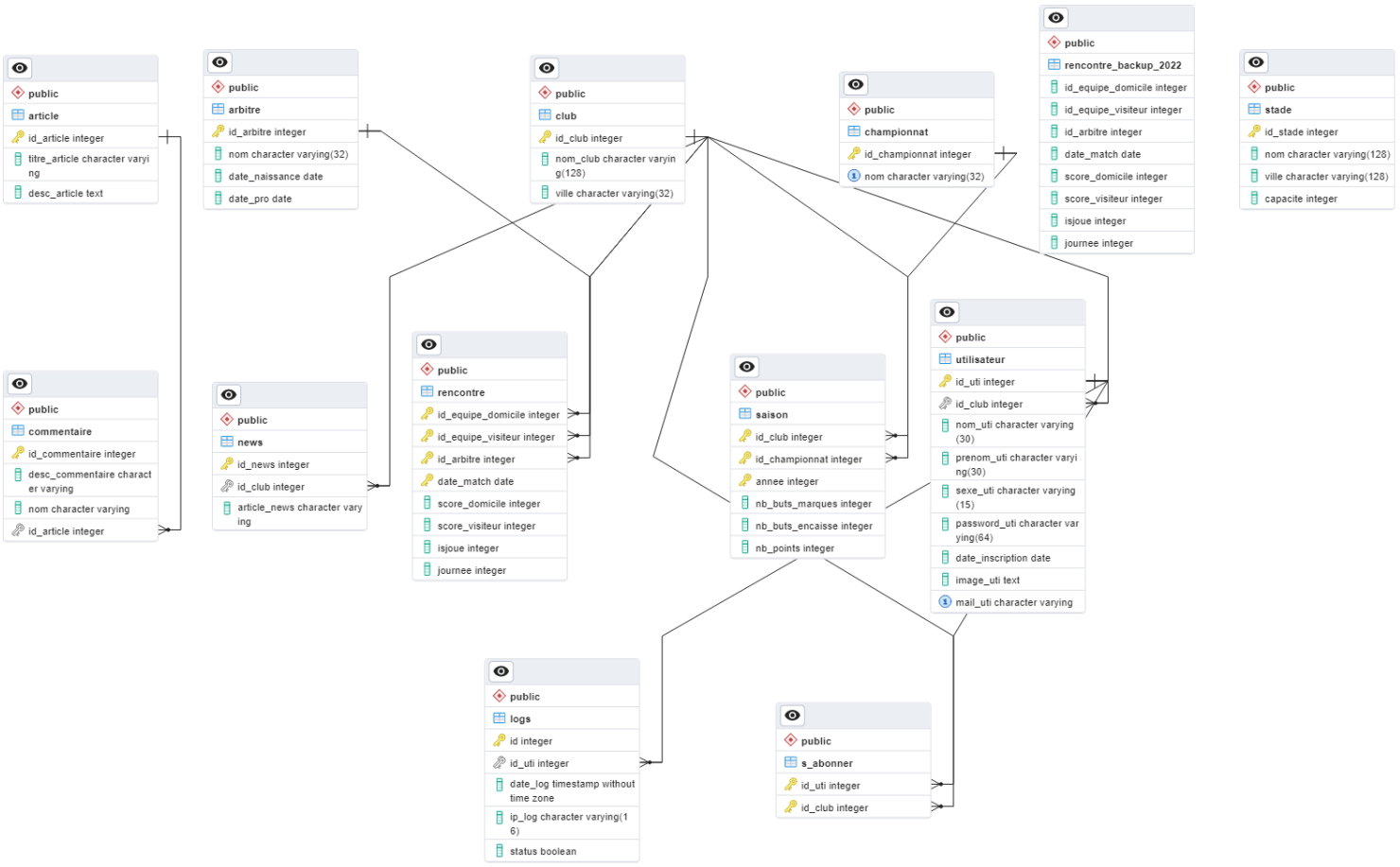
```
            break;
        case "Inscription":
            include("./Controller/c_inscription.php");
            break;
        case "Connexion":
            include("./Controller/c_connexion.php");
            break;
        case "Article":
            include("./Controller/c_article.php");
            break;
        case "Profil":
            include("./Controller/c_profil.php");
            break;
        case "Commentaire":
            if (isset($_GET['idArticle']) && empty($_GET['idArticle'])) {
                include('./Controller/c_accueil.php');
                break;
            } else {
                include('./Controller/c_commentaire.php');
                break;
            }
    }
    ?>
</body>
```

Ici la page Index.php sert de Point d'entrée et de Routage.

BASE DE DONNÉES :

DIAGRAMME DE MODELE CONCEPTUEL DE DONNÉES :

Voici le diagramme de modèle conceptuel de données de la base de données DB_Ligue :



ACCES AUX DONNEES ET TRAITEMENTS

PDO :

Lors de la création du site j'ai utilisé PDO (PHP Data Objects) qui est une extension PHP qui fournit une interface orientée objet pour accéder aux bases de données. Elle offre une abstraction de haut niveau permettant aux développeurs d'interagir avec différentes bases de données relationnelles, telles que MySQL, PostgreSQL, SQLite, etc., en utilisant un ensemble commun de méthodes et de conventions.

PDO nous offre :

Connexion à la base de données : PDO permet d'établir une connexion à la base de données en utilisant différentes méthodes de connexion, telles que PDO::__construct() ou PDO::dsn.

Utilisation de requêtes préparées : PDO facilite l'utilisation de requêtes préparées pour exécuter des instructions SQL. Les requêtes préparées permettent de séparer les instructions SQL de leurs paramètres, ce qui améliore la sécurité en prévenant les attaques par injection SQL.

Sécurité : PDO offre des fonctionnalités intégrées pour échapper les valeurs des paramètres et prévenir les vulnérabilités de sécurité courantes.

Gestion des erreurs : PDO gère les erreurs de base de données en lançant des exceptions PDOException qui peuvent être capturées et gérées par le code.

Support de différentes bases de données : PDO prend en charge plusieurs types de bases de données relationnelles, ce qui signifie que le même code peut être utilisé pour interagir avec différentes bases de données en utilisant les mêmes méthodes et conventions.

Gestion des transactions : PDO facilite la gestion des transactions, permettant d'exécuter des opérations qui doivent être atomiques (c'est-à-dire, soit toutes les opérations réussissent, soit aucune ne réussit).

Méthodes utiles : PDO fournit un ensemble de méthodes pour exécuter des requêtes SQL (par exemple, PDO::query() et PDO::exec()), récupérer des résultats (par exemple, PDOStatement::fetch()), compter les résultats (PDOStatement::rowCount()), etc.

L'utilisation de PDO dans le développement PHP offre plusieurs avantages, tels que la portabilité du code, la facilité de maintenance et la sécurité améliorée. Cela permet également de se conformer à l'architecture MVC (Modèle-Vue-Contrôleur) en séparant la logique d'accès aux données (le modèle) du reste de l'application.

CONTROLLER GESTIONBDD :

Pour se connecter à la base de données, j'ai créé une classe dans le controller qui se nomme GestionBDD qui gère et sert la connexion à la base de données :

```
class GestionBDD
{
    private string $user;
    private string $pass;
    private string $dns;
    private string $db;

    private PDO $cnx;
    function __construct(string $db, string $user = 'postgres', string $pass = 'root')
    {
        $this->user = $user;
        $this->pass = $pass;
        $this->db = $db;
        $this->dns = 'pgsql:host=localhost;dbname=DB_Ligue;';
    }

    /**
     *
     * @return PDO
     */
    function connect(): PDO
    {
        try {
            $this->cnx = new PDO($this->dns, $this->user, $this->pass);
            return $this->cnx;
```

```
    } catch (PDOException $e) {
        print "Erreur de connexion à la base !: " . $e->getMessage() . "<br/>";
        die();
    }
}
```

A chaque connexion à la base de données on instancie une nouvelle GestionBDD et on utilise la méthode connect() :

```
$BDD = new GestionBDD('DB_Ligue');
$cnx = $BDD->connect();
```

SUPERGLOBALES:

En PHP, **`$_SESSION`**, **`$_REQUEST`**, et **`$_POST`** sont des superglobales, des tableaux associatifs intégrés qui contiennent des données provenant des requêtes HTTP. Voici leur description et leurs utilisations :

`$_SESSION` :

`$_SESSION` est un tableau associatif qui stocke des variables de session côté serveur.

Les sessions permettent de stocker des informations spécifiques à un utilisateur tout au long de sa navigation sur le site.

Les valeurs stockées dans **`$_SESSION`** sont accessibles et persistantes sur plusieurs pages tant que la session utilisateur est active.

Pour utiliser **`$_SESSION`**, il faut d'abord démarrer la session en appelant la fonction **`session_start()`**.

Par exemple, vous pouvez stocker des informations telles que l'ID de l'utilisateur connecté, ses préférences ou tout autre type de données spécifiques à la session.

`$_REQUEST` :

`$_REQUEST` est un tableau associatif qui contient les données envoyées par l'utilisateur à travers une requête HTTP, indépendamment de la méthode (GET, POST, etc.).

Il combine les données des tableaux **`$_GET`**, **`$_POST`**, et **`$_COOKIE`**.

`$_REQUEST` peut être utilisé pour accéder aux paramètres de l'URL, aux données de formulaire et aux cookies.

Cependant, il est recommandé d'utiliser **`$_GET`**, **`$_POST`**, ou **`$_COOKIE`** directement lorsque vous connaissez la source des données, pour des raisons de sécurité et de lisibilité du code.

`$_POST` :

`$_POST` est un tableau associatif qui contient les données envoyées par l'utilisateur via la méthode POST d'une requête HTTP.

Il est couramment utilisé pour récupérer les données saisies dans un formulaire soumis par l'utilisateur.

Les données sont envoyées dans le corps de la requête HTTP, ce qui les rend invisibles dans l'URL.

Par exemple, vous pouvez récupérer les valeurs des champs de formulaire en utilisant **`$_POST['nom_champ']`**.

Il est important de noter que ces superglobales contiennent des données brutes et qu'il est nécessaire d'effectuer des validations et des filtrages appropriés avant de les utiliser pour garantir la sécurité de l'application. De plus, **`$_SESSION`** nécessite une configuration correcte du support de session dans PHP pour fonctionner correctement.

J'ai notamment dû utiliser ces Superglobales pour effectuer pour vérifier si les identifiants entrés par l'utilisateur sont bien inscrits dans la base de données :

```
if (isset($_POST['submit'])) {


   if (isset($_REQUEST['username']) && !(empty($_REQUEST['username'])) && isset($_REQUEST['password']) &&
!(empty($_REQUEST['password']))) {


      $BDD = new GestionBDD('DB_ligue');

      $cnx = $BDD->connect();


      $GU = new UserManager($cnx);


      $mailConnexion = $_REQUEST['username'];

      $mdpConnexion = $_REQUEST['password'];


      if ($GU->existUser($mailConnexion, $utilisateur)) {
```

FORMULAIRES :

Afin de récupérer les identifiants des utilisateurs et ensuite les insérer dans la base de données, j'ai créé un formulaire d'inscription ainsi qu'un formulaire de connexion en utilisant à la balise <form> ainsi Le traitement des données soumises par le formulaire peut être effectué en utilisant PHP :

Formulaire d'inscription dans la Vue :

```
<form action="./Controller/c_inscription.php" method="post">
                        <h1 class="box-title">Inscription</h1>
                        <br>Prénom : <input type="text" name="firstname" id="nom" required="" /> <br>
                        <br>Nom : <input type="text" name="name" /><br>
                        <br>Mail : <input type="email" id="idmail" name="email" /><br>
                        <br>Mot de passe : <input type="password" id="idpassword" name="password" minlength="8"><br>
                        <br> Sexe : <input type="radio" checked name="gender" value="H">Homme
                        <input type="radio" name="gender" value="F">Femme
                        <br>
                        <br> Photo de profil : <input type="file" id="idpdp" name="pic" accept="image/png,image/jpeg"><br>
```

```
                <br>
                <div class="form-group" name="team">
                    <label for="idClubFavori">Votre Club favori </label>
                    <select class="form-select" id="idClubFavori" aria-label="Choix du club favoris" name="favori">
                        <option value="1" selected>Choisi ton club favori</option>
                        <?php $BDD->getClubIntoSelect($cnx) ?>
                    </select>
                </div>
                <br>

                <input type="submit" value="Créer le compte" name="valid" class="box-button">

            </form>
```

Puis dans le Controller on traite les données entrées dans le formulaire :

```
if (isset($_REQUEST["valid"])) {

    $BDD = new GestionBDD('DB_Ligue');
    $cnx = $BDD->connect();

    $GC = new ClubManager($cnx);
    $tableResultEquipe = $GC->getListClubByName();

    if (isset($_REQUEST['firstname']) && !empty($_REQUEST['firstname'])) {
        $prenom = $_REQUEST['firstname'];
    }

    if (isset($_REQUEST['name']) && !empty($_REQUEST['name'])) {
        $nom = $_REQUEST['name'];
    }

    if (isset($_REQUEST['gender']) && !empty($_REQUEST['gender'])) {
        $sexe = $_REQUEST['gender'];
    }
```

```
    if (isset($_REQUEST['email']) && !empty($_REQUEST['email'])) {
        $email = $_REQUEST['email'];
    }

    if (isset($_REQUEST['password']) && !empty($_REQUEST['password'])) {
        $password = $_REQUEST['password'];
        $password = password_hash($_REQUEST['password'], PASSWORD_BCRYPT);
    }

    if (isset($_REQUEST['pic']) && !empty($_REQUEST['pic'])) {
        $pdp = $_REQUEST['pic'];
    }

    if (isset($_REQUEST['favori']) && !empty($_REQUEST['favori'])) {

        $clubfavori = $_REQUEST['favori'];
    }

    $dateInscription = date('d-m-Y');

        $userInscription = new User(0, (int) $clubfavori, $nom, $prenom, $sexe,  $password, $dateInscription, $pdp, $email);
        //(int $id_utilisateur , int $id_club_utilisateur, string $nom_utilisateur, string $prenom_utilisateur, string $sexe_utilisateur, string $password_utilisateur, string $date_inscription_utilisateur, string $mail_utilisateur)
        $BDD = new GestionBDD('DB_Ligue');
        $cnx = $BDD->connect();

        $GU = new UserManager($cnx);
        $GU->insertUser($userInscription);
        echo "Vous avez été inscrit !";
```

AFFICHER DES DONNEES

HTML :

Le HTML (HyperText Markup Language) est le langage de balisage standard utilisé pour structurer et présenter le contenu d'une page web. Il définit la structure logique et la mise en forme du contenu d'une page web en utilisant des balises et des attributs. Ici, je l'utilise pour permettre l'affichage et faire la mise en page des données. Exemple pour le menu :

```
<header class="menuh">
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <img class="logo" src="../Img/channels4_profile.jpg    "> >
      <ul class="navbar-nav">
        <li class="nav-item active">
          <a class="nav-link" href='index.php?Page=Accueil '> Accueil <span class="sr-only">(current)</span></a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href='index.php?Page=Page1'> Page1</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href='index.php?Page=Page2'> Page2</a>
        </li>
        <li class="nav-item">
          <a class="nav-link " href='index.php?Page=Article'> Article</a>
        </li>
        <div class="connecté">
          <?php
          if (!isset($_SESSION['id'])) { ?>
            <li class="nav-item">
              <a class="nav-link " href='index.php?Page=Inscription'> Inscription</a>
```

```
                </li>
                </li>
                <li class="nav-item">
                    <a class="nav-link " href='index.php?Page=Connexion'> Connexion</a>
                </li>
            <?php
            } else {
            ?>
                <li class="nav-item">
                    <a class="nav-link " href='/Controller/c_deconnexion.php'>Déconnexion</a>
                </li>
                <li class='navbar-item'> <a class="nav-link" href='index.php?Page=Profil'>Profil</a>
                </li>
            <?php
            }
            ?>
          </div>
        </ul>
      </div>
    </nav>
</header>
```

CSS :

CSS (Cascading Style Sheets) est un langage de feuilles de style utilisé pour définir la présentation et l'apparence d'un document HTML. Il permet de contrôler le style, la mise en forme, les couleurs, la disposition et d'autres aspects visuels d'une page web. Dans le cas du site web j'ai créer un fichier à part qui se nomme newCascadeStyleSheet.css qui me permet de personnaliser l'apparence du site et d'améliorer l'expérience visuelle des utilisateurs.

BOOTSTRAP :

Bootstrap est un framework front-end populaire utilisé pour la conception et le développement de sites web réactifs et mobiles. Il fournit une collection de styles CSS prédéfinis, de composants JavaScript et de mises en page responsives, ce qui facilite la création d'interfaces utilisateur

modernes et attrayantes. Pour le site je l'ai notamment utilisé pour obtenir un style prédéfini grâce aux balises <link> :

```
<base href="/">

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

<link href="./CSS/newCascadeStyleSheet.css" rel="stylesheet">
```