Annexe 1 : Exercice Cinema

```php
public function listFilms()
{
    // Connexion à la base de données via la classe Connect
    $pdo = Connect::seConnecter();
    $requete = $pdo->query("
      SELECT
        film.id_film,
        film.titre,
        GROUP_CONCAT(genre.libelle_genre) AS tous_genre,
        affiche,
        CONCAT(DAY(film.date_sortie_fr), '-', MONTH(film.date_sortie_fr), '-', YEAR(film.date_sortie_fr)) AS release_date
    FROM film
    INNER JOIN genre_film ON film.id_film = genre_film.id_film
    INNER JOIN genre ON genre_film.id_genre = genre.id_genre
    GROUP BY film.id_film, film.titre
    ORDER BY titre ASC
    ");

    require "view/listFilms.php"; //necessaire pour récuperer la vue qui nous intéresse
}
```

```php
ob_start();
$film = $requete->fetch();
?>

<div class="intro">
  <p>Il y a <?= $requete->rowCount() ?> films.</p>
  <div class="group_btn">
    <button onclick="window.location.href='index.php?action=addFilm';">Ajouter un film</button>
    <button onclick="window.location.href='index.php?action=addCasting';">Ajouter un casting</button>
  </div>
</div>

<section class="princ">
  <?php
   foreach ($requete->fetchAll() as $film) { ?>
    <div class="group">
      <div class="text">
        <p><a href="index.php?action=detFilm&id=<?= $film['id_film'] ?>"><?= $film["titre"] ?></a></p>
        <p>Sorti le : <?= $film["release_date"] ?></p>
      </div>
      <p class="imgContain"><img class="img_aff" src="<?= $film["affiche"] ?>" alt="Affiche du film . $film['titre']" /></p>
    </div>

  <?php } ?>
</section>

<?php

$titre = "<h1 class='titreH1'>Liste des films</h1>";
$contenu = ob_get_clean(); //Fin de la vue
```

Responsive :

```css
.princ {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: 1fr;
    gap: 30px 10px;
}

@media (max-width: 1241px) {
.princ, .gpGenre {
    grid-template-columns: repeat(3,1fr);
    grid-template-rows: auto;
}
}
```

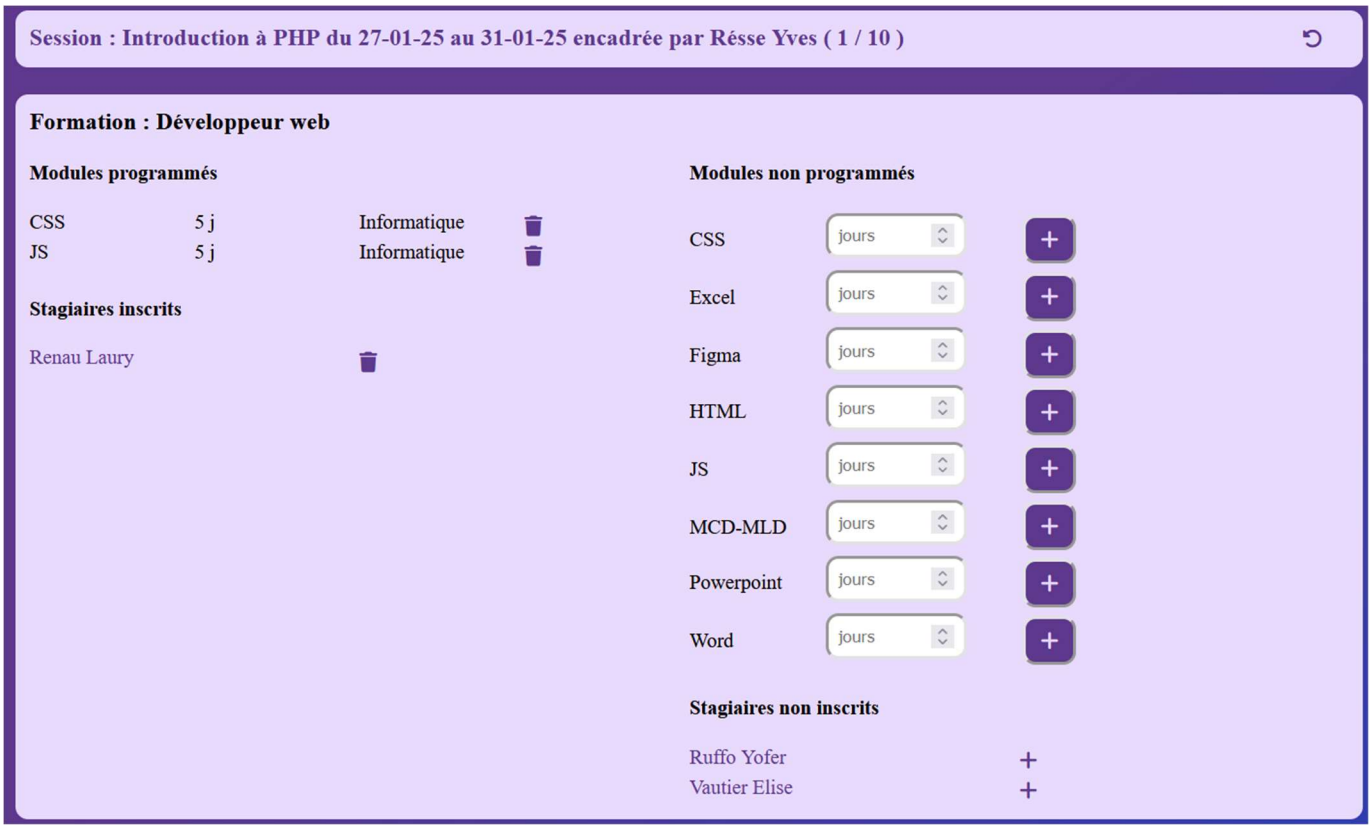
```css
@media (max-width: 720px) {
    .princ, .gpGenre {
        grid-template-columns: repeat(2,1fr);
        grid-template-rows: auto;
    }
```

```css
@media (max-width: 520px) {
    .princ, .gpGenre {
        grid-template-columns: 1fr;
        grid-template-rows: auto;
    }
```

Annexe 2 : Site internet pour un streamer

```
// useEffect : déclenche une fonction au montage du composant
useEffect(() => {
  async function fetchLastVideo() {
    try {
      // Étape 1 : Récupération d'un token d'accès via le client ID et le secret
      const tokenResponse = await axios.post('https://id.twitch.tv/oauth2/token', null, {
        params: {
          client_id: CLIENT_ID,
          client_secret: CLIENT_SECRET,
          grant_type: 'client_credentials'
        }
      });
      const accessToken = tokenResponse.data.access_token;

      // Étape 2 : Récupération des données utilisateur pour obtenir l'ID Twitch
      const userResponse = await axios.get('https://api.twitch.tv/helix/users', {
        headers: {
          'Client-ID': CLIENT_ID,
          'Authorization': `Bearer ${accessToken}`
        },
        params: {
          login: USER_LOGIN
        }
      });
      const userId = userResponse.data.data[0].id;

      // Étape 3 : Récupération de la dernière vidéo de l'utilisateur via son ID
      const videoResponse = await axios.get('https://api.twitch.tv/helix/videos', {
        headers: {
          'Client-ID': CLIENT_ID,
          'Authorization': `Bearer ${accessToken}`
        },
        params: {
          user_id: userId,
          first: 1 // On ne récupère que la dernière vidéo
        }
      });

      // Stockage de la vidéo dans l'état
      setLatestVideo(videoResponse.data.data[0]);
    } catch (error) {
      // Gestion des erreurs d'appel à l'API
      console.error('Erreur lors de la récupération des données de l\'API Twitch:', error);
      setError('Erreur lors de la récupération des données de l\'API Twitch');
    }
  }

  // Lancement de la fonction d'appel API
  fetchLastVideo();
}, []); // Le tableau vide indique que cet effet ne s'exécute qu'une fois (au montage)
```

Annexe 3 : Exercice Session




```
    {% block javascripts %}
        {% block importmap %}
            {{ importmap('app') }}
        {% endblock %}
    {% endblock %}
</head>


        <nav id="navbar">
            <div id="welcome">
                <p>Bienvenue <a href="{{ path('app_profile') }}">toi !</a></p>
                <p><a href="{{ path('app_login') }}">Se déconnecter</a></p>
            </div>

            <ol>
                <li><a href="{{ path('app_home') }}">Accueil</a></li>
                <li><a href="{{ path('app_session') }}">Sessions</a></li>
                <li><a href="{{ path('app_training') }}">Formations</a></li>
                <li><a href="{{ path('app_module') }}">Modules</a></li>
                <li><a href="{{ path('app_category') }}">Catégories</a></li>
                <li><a href="{{ path('app_trainer') }}">Formateurs</a></li>
                <li><a href="{{ path('app_intern') }}">Stagiaires</a></li>
            </ol>

            <div class="burger_button">
                <div class="line"></div>
                <div class="line"></div>
                <div class="line"></div>
            </div>
        </nav>



    {% block body %}{% endblock %}
```

Annexe 4 : Exercice Forum

```php
//on filtre
$nickname = filter_input(INPUT_POST, "nickname", FILTER_SANITIZE_FULL_SPECIAL_CHARS);
$email = filter_input(INPUT_POST, "email", FILTER_VALIDATE_EMAIL);
$password1 = filter_input(INPUT_POST, "password1", FILTER_VALIDATE_REGEXP, array("options" => array("regexp" => "/^(?=.*[A-Z])(?
=.*\d)(?=.*[\W_]).{12,}$/")));
// ^: debut de chaine, $ : fin de chaine, (?=.*[A-Z]) : lettres, (?=.*\d) : digit, (?=.*[\W_]) : caracteres speciaux, {12,} :
12 caracteres
$password2 = filter_input(INPUT_POST, "password2", FILTER_VALIDATE_REGEXP, array("options" => array("regexp" => "/^(?=.*[A-Z])(?
=.*\d)(?=.*[\W_]).{12,}$/")));
$terms = isset($_POST['terms']) ? true : false;  // Vérifie si la case est cochée

if (!$terms) {
    $errorMessage = "Vous devez accepter les termes et conditions pour vous inscrire.";
}

//on les vérifie
if (!$errorMessage && $nickname && $email && $password1 && $password2) {

    $verifyNickname = $user->findOneByNickname($nickname);
    $verifyEmail = $user->findOneByEmail($email);

    if (!$verifyNickname && !$verifyEmail) { //Vérification si l'un ou l'autre n'existe pas en bdd
        // add in bdd
        if ($password1 == $password2 && strlen($password1) >= 12) {
            $user->add([
                "nickname" => $nickname,
                "email" => $email,
                "password" => password_hash($password1, PASSWORD_DEFAULT)
            ]);

            Session::addFlash("success", "Compte créé avec succès. Veuillez vous connecter.");
            $this->redirectTo("security", "login");
            exit();
        } else {
```

```php
<?php if (isset($errorMessage)) { ?>
    <div class="error-message">
        <p><?php echo htmlspecialchars($errorMessage); ?></p>
    </div>
<?php } ?>

<div class="blockForm title ">
    <p class="mailLogin"><label for="email">Email</label></p>
    <input type="email" name="email" id="email">
</div>

<div class="blockForm title">
    <p class="mdpLogin"><label for="password">Mot de passe</label></p>
    <input type="password" name="password" id="password">
</div>
```

Annexe 5 : Exercice Cinema

```php
abstract class Connect
{
    const HOST = 'localhost';
    const DB = 'cinemalily';
    const USER = 'root';
    const PASS = '';
```

```php
public static function seConnecter()
{
    try {
        return new \PDO(   //le \ indique au framework que PDO est une classe native
            'mysql:host=' . self::HOST . ';dbname=' . self::DB . ';charset=utf8',
            self::USER,
            self::PASS
        );
    } catch (\PDOException $ex) {
        return $ex->getMessage();
    }
}
```

```php
public function detActeur($id)
{
    $pdo = Connect::seConnecter();

    $requete2 = $pdo->prepare("
      SELECT DISTINCT genre.id_genre, film.id_film,
            role.id_role,
            film.titre,
            film.affiche,
            role.personnage,
            role.photo,
            GROUP_CONCAT(genre.libelle_genre) AS tous_genre
        FROM film
        INNER JOIN casting ON film.id_film = casting.id_film
        INNER JOIN acteur ON casting.id_acteur = acteur.id_acteur
        INNER JOIN role ON casting.id_role = role.id_role
        INNER JOIN genre_film ON film.id_film = genre_film.id_film
        INNER JOIN genre ON genre_film.id_genre = genre.id_genre
        WHERE casting.id_acteur = :id
        GROUP BY film.id_film, film.titre, film.affiche, role.id_role, role.personnage, genre.id_genre
    ");

    $requete2->execute(["id" => $id]);

    require "view/detActeur.php";
```

```php
public function addActeur()
    {
        $pdo = Connect::seConnecter();

        if (isset($_POST['submit'])) {
            $name_acteur = filter_input(INPUT_POST, 'name_acteur', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            $firstname_acteur = filter_input(INPUT_POST, 'firstname_acteur', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            $sexe = filter_input(INPUT_POST, 'sexe', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            $ddn_acteur = filter_input(INPUT_POST, 'ddn_acteur', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            $url_acteur = filter_input(INPUT_POST, 'url_acteur', FILTER_VALIDATE_URL);


            if ($name_acteur && $firstname_acteur && $sexe && $ddn_acteur && $url_acteur) {
                $requete = $pdo->prepare("
                INSERT INTO personne (nom, prenom, sexe, date_naissance, photo)
                VALUES (:name_acteur, :firstname_acteur, :sexe, :ddn_acteur, :url_acteur)
                ");

                $requete->execute([
                    "name_acteur" => $name_acteur,
                    "firstname_acteur" => $firstname_acteur,
                    "sexe" => $sexe,
                    "ddn_acteur" => $ddn_acteur,
                    "url_acteur" => $url_acteur
                ]);

                $idPersonne = $pdo->lastInsertId();

                header("Location: index.php?action=listActeurs");
                exit();
            }
        }

        require "view/addActeur.php";
```

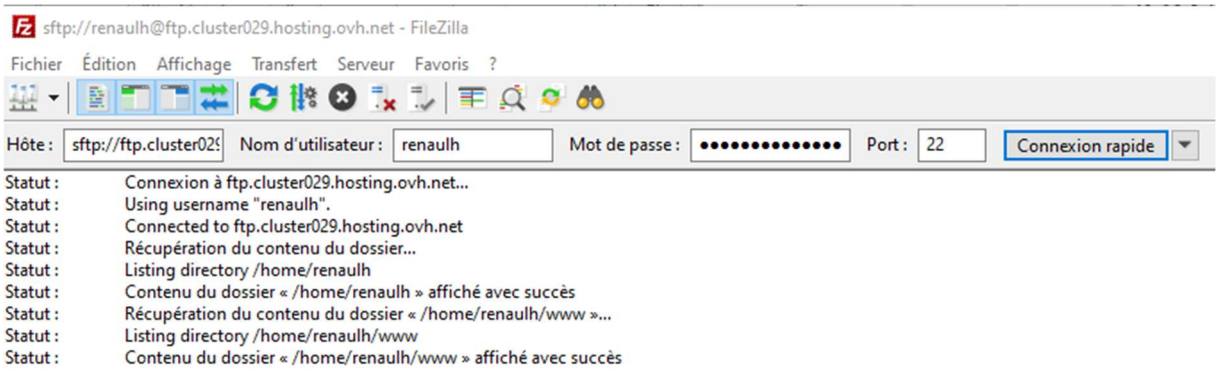
Annexe 6 : Déploiement d'un site internet

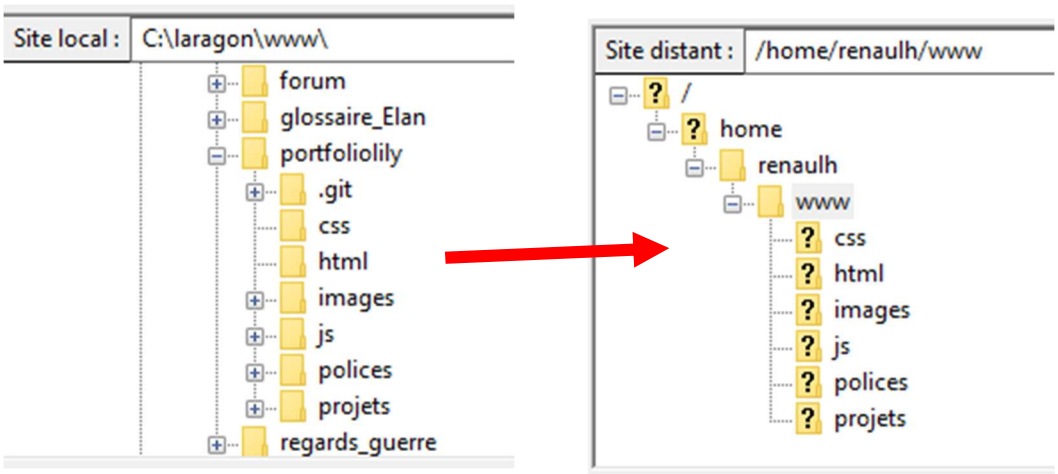
Serveur FTP et SFTP :
ftp.cluster029.hosting.ovh.net

Port FTP : 21
Port SFTP : 22

Lien FTP vers le cluster :
ftp://renaulh@ftp.cluster029.hosting.ovh.net:21/

Chemin du répertoire home :
/home/renaulh





https://www.renaulaury.fr