



# **BTS SIO – Option SLAM**

## **Documentation d'épreuve**

Rédacteur	Version	Date	Nb pages
Johann DIETRICH	1.5	18/04/2023	33

**Portfolio**

# SOMMAIRE

<b>1</b>	<b>CAHIER DES CHARGES .....</b>	<b>4</b>
1.1	Introduction .....	4
1.2	Expression fonctionnelle du besoin .....	5
1.3	Contraintes.....	6
1.4	Gestion des droits d'accès.....	7
<b>2</b>	<b>DESCRIPTION DES ENVIRONNEMENTS .....</b>	<b>8</b>
<b>3</b>	<b>METHODOLOGIE.....</b>	<b>9</b>
3.1	Méthodologie et versioning.....	9
3.2	Gestion de projet .....	10
<b>4</b>	<b>MISE EN OEUVRE .....</b>	<b>12</b>
4.1	Environnement de travail .....	12
4.2	Schématisation du projet .....	13
4.3	Début de la création du site .....	15
4.4	Création de la page principale et mise en commun avec la BDD.....	16
4.5	Formulaire de connexion .....	20
4.6	Page de documents/projets .....	23
4.7	Page d'affichage d'articles.....	25
4.8	Partie commentaires.....	27
4.9	Page administrateur .....	29
<b>5</b>	<b>GESTION DE LA MAINTENANCE (CORRECTIVE / EVOLUTIVE).....</b>	<b>32</b>
5.1	Mise à jour de la documentation du SI.....	32
5.2	Evaluation de la qualité de la solution.....	32
5.3	Procédure de correction d'un dysfonctionnement .....	32
5.4	Gestion des tests de mise à jour .....	32
<b>6</b>	<b>BILAN DU PROJET.....</b>	<b>33</b>
6.1	Validation des exigences point par point.....	33
6.2	Axes d'amélioration .....	33
6.3	Compétences acquises .....	33



# 1 Cahier des charges

---

## 1.1 Introduction

---

### Type de mission

Mission effectuée pour les cours, effectué partiellement durant des heures dédiées de cours de programmation et également en autonomie durant un temps libre.

### Contexte

Création du portfolio obligatoire pour le BTS afin de pouvoir afficher les projets effectuées durant l'année, les compétences obtenues. Ceci afin d'être utilisé comme projet présentable complet (E5) de fin de deuxième année.

### Demande du client

Portfolio avec toutes les informations nécessaires, tel qu'un CV, des articles et les projets effectués durant l'année. Celui-ci étant supposé être capable de montrer les compétences acquises au cours de l'année.

### Budget disponible

Le budget disponible est notre propre budget personnel, donc n'a pas vraiment de limite mais se doit d'être le moins coûteux possible.

## 1.2 Expression fonctionnelle du besoin

---

Liste des fonctionnalités attendues :

### Front office

Page d'accueil :

- Capable d'afficher des fonctionnalités graphiques, le CV, télécharger le CV et une présentation.

Page Projets :

- Afficher les différents projets E4 de manière sobre.

Page de connexion :

- Formulaire permettant la connexion a un compte

Page de création d'utilisateur :

- Formulaire permettant la création d'un compte utilisateur

Page articles :

- Capable d'afficher les articles et les commentaires de ceux-ci.

### Back Office

Page administrateur :

- Capable de modifier le cv, créer des articles et de voir les logs.

## 1.3 Contraintes

---

### Juridiques

Contrainte juridique due aux lois RGPD

Le RGPD établit plusieurs obligations pour les sites web, notamment :

- Obtenir le consentement explicite et informé des utilisateurs avant de collecter, traiter ou stocker leurs données personnelles.
- Informer clairement les utilisateurs de la finalité de la collecte, du traitement et du stockage des données personnelles.
- Mettre en place des mesures de sécurité appropriées pour protéger les données personnelles contre tout accès, utilisation, divulgation ou destruction non autorisée.
- Permettre aux utilisateurs d'exercer leurs droits, tels que le droit d'accès, de rectification, de suppression, de limitation et d'opposition au traitement de leurs données personnelles.

### Techniques

Utilisation de langages de programmation spécifiques tels que :

- PHP
- CSS
- HTML
- JS

Et de Frameworks tel que :

- BOOTSTRAP

## 1.4 Gestion des droits d'accès

---

<b>Administrateur</b>
Création de commentaires, création d'articles, modification du CV.

<b>Utilisateurs</b>
Création de commentaires, lecture des articles.

<b>Visiteurs</b>
Lecture des articles.

## 2 Description des environnements

---

### Environnement de développement

Utilisation actuelle d'un simple serveur live avec WAMP pour servir de test.

### Environnement de production

- Futur Serveur web OVH

Utilisation et modification d'un VPS (virtual private server) OVH tournant sur une machine DEBIAN 11 afin rendre le site web portfolio disponible sur le web et visible de part tout le monde. Ceci avec un nom de domaine également (dietrichportfolio.site).



## 3 Méthodologie

---

### 3.1 Méthodologie et versioning

---

GitHub :

Utilisation de GitHub qui nous sert à créer un dossier distant, accessible à tout moment. Celui-ci permettant de versionner, de mettre à jour, et de tester des versions du code du site web avant que celui-ci soit mis sur le serveur global.

Méthodologie :

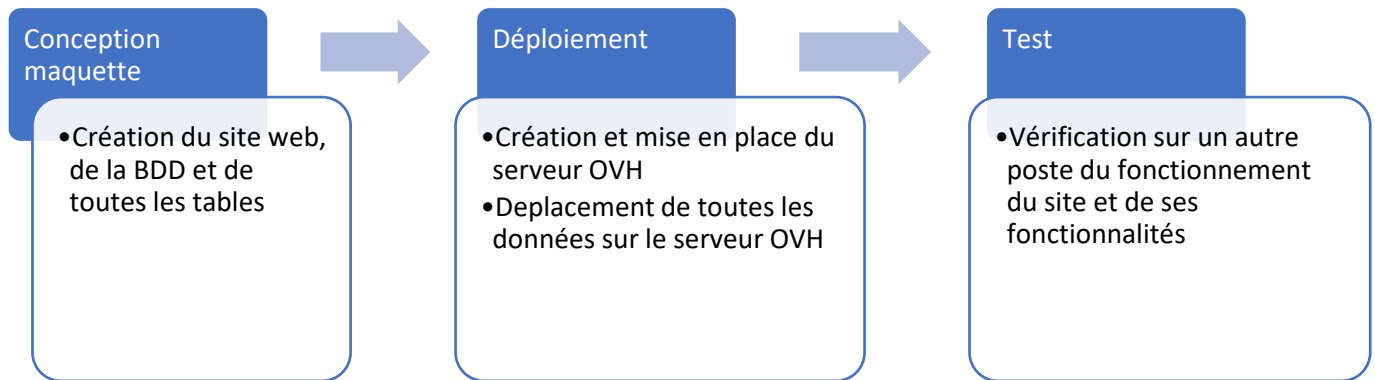
Dès qu'il y a une partie du portfolio qui a l'air fonctionnelle, une nouvelle version du projet est mise sur GitHub. Ceci sert à garder une version la plus stable possible en sauvegarde sur GitHub, tandis que des tests expérimentaux ne seront que fait en local, en physique ou il n'y aura aucun risque.

## 3.2 Gestion de projet

---

### 3.2.1 Planing de déploiement de la solution

---



### 3.2.2 Budget

---

Actuellement, un léger cout de 1 euro/mois qui sont utilisés pour acheter le server OVH nécessaire pour rendre le site web disponible sur internet et 1euro/ans pour le nom de domaine.

Le reste des outils étant gratuits, il n'y a donc besoin d'aucun budget supplémentaire en dehors de cela.

## 4 Mise en oeuvre

Au tout début du projet, il faut d'abord préparer son environnement de travail, ses outils et avoir l'idée globale de ce que l'on veut faire avec celui-ci.

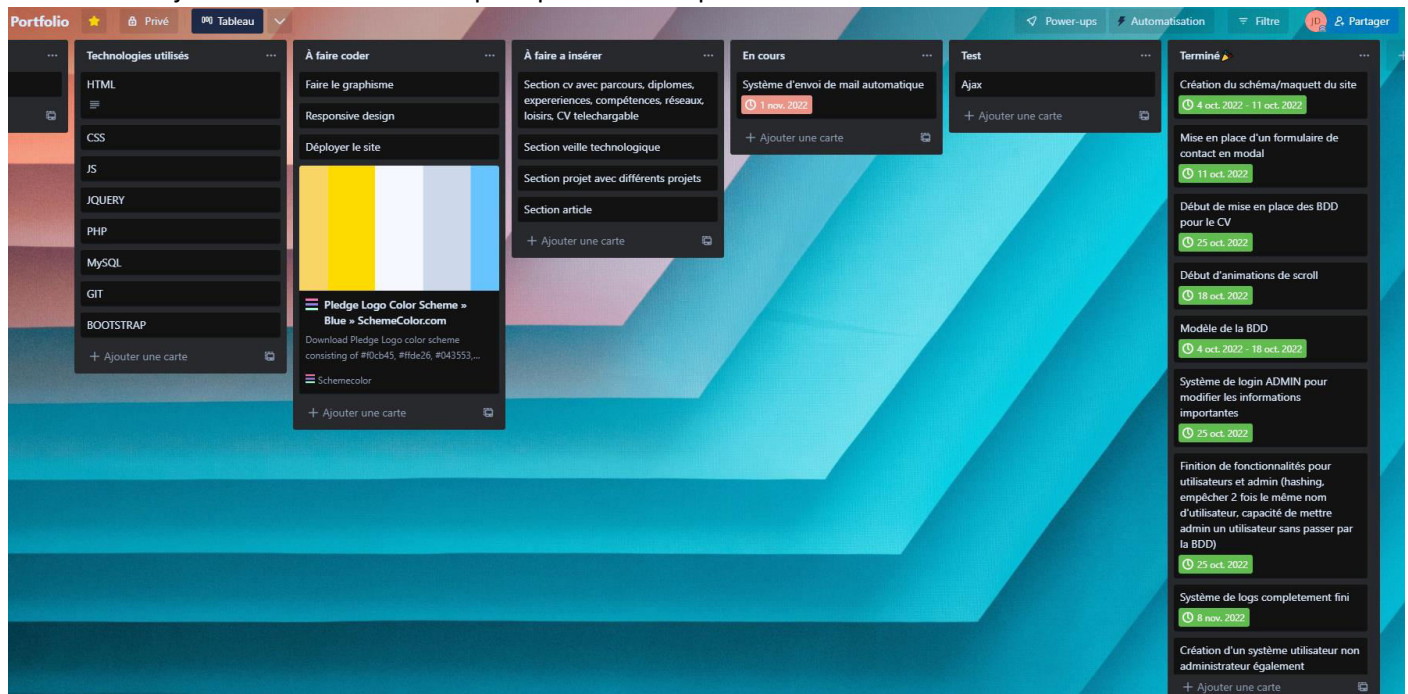
### 4.1 Environnement de travail

Pour l'instant, le projet de portfolio est sur ordinateur fixe, en local avec l'utilisation d'une base de données créée par WAMP afin de faciliter l'utilisation et la création de celle-ci.

L'installation de WAMP étant très facile (juste télécharger et exécuter) je n'en parlerai pas plus que cela.

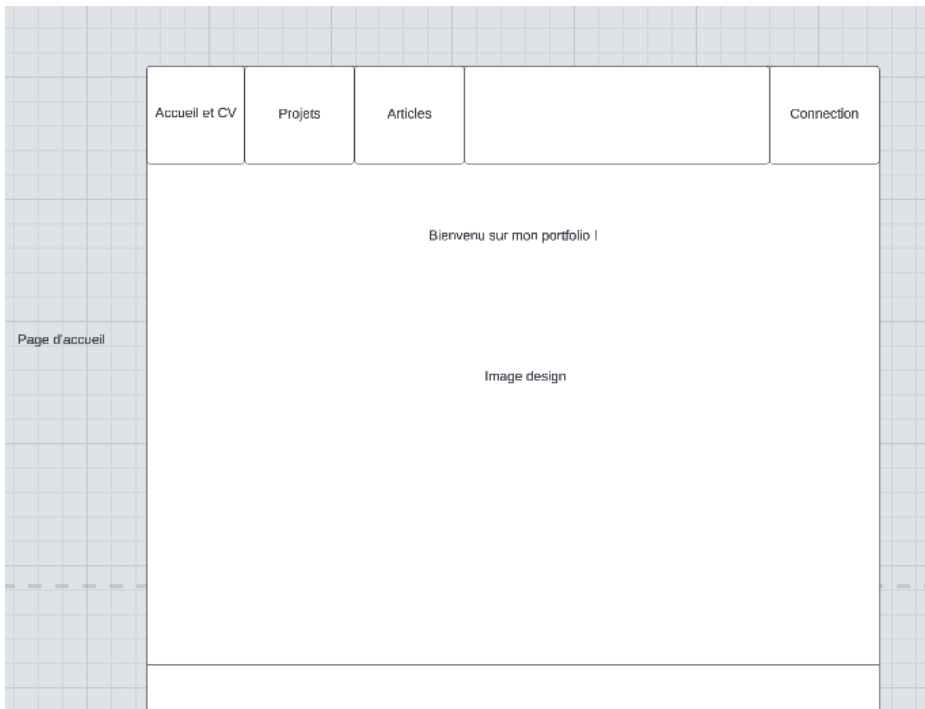
Je me suis servi de VScode pour programmer car c'est l'éditeur auquel je suis le plus habitué.

Et finalement je me suis servi de Trello pour planifier les opérations de telle sorte :



## 4.2 Schématisation du projet

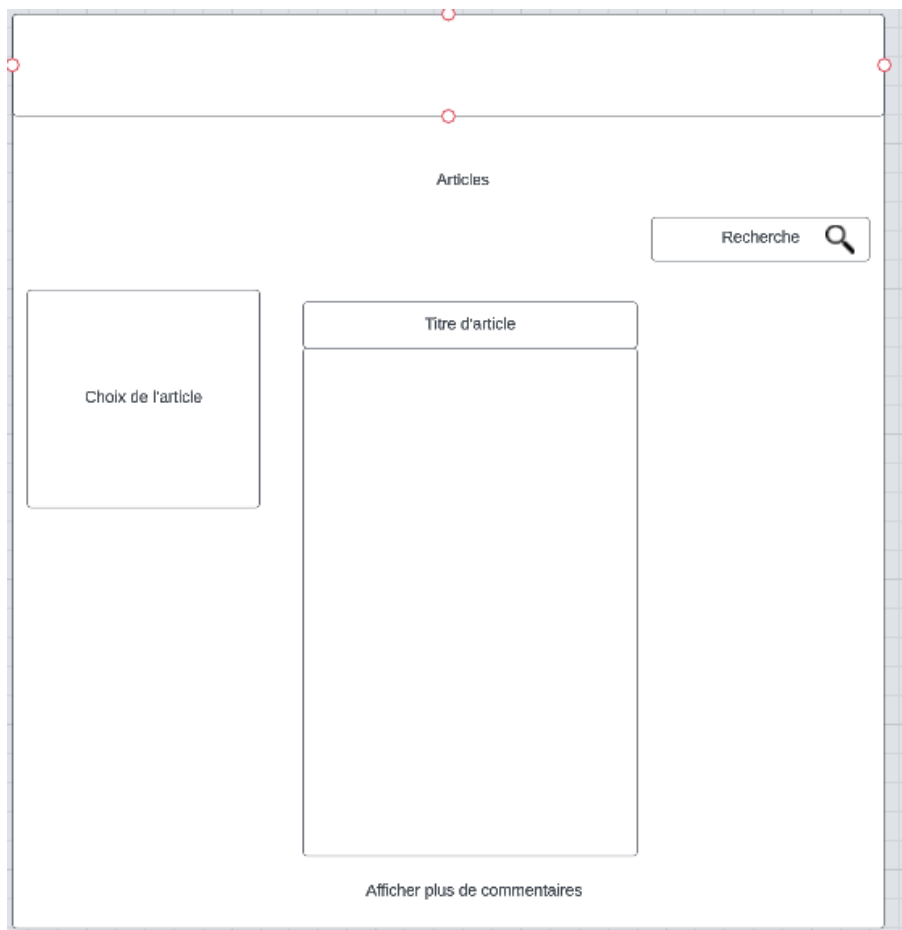
En effet, avant de commencer de programmer, j'ai d'abord mis en place un schéma basique permettant de voir l'apparence générale, et les fonctionnements voulus sur mon site. Ceux-ci étant étés faits sur LucidCharts et donnant un résultat comme vu ci-dessous avec la page d'accueil :



La page du cv :



Et la page des articles disponibles :



Une fois que tout est clair, on peut commencer la programmation du site !

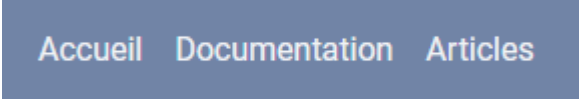
## 4.3 Début de la création du site

---

Comme départ, il faut déjà créer les pages qui seront requises partout, dans notre cas, le header qui nous permet de naviguer à travers les pages. Celui-ci sera appelé ensuite sur toutes les pages et stylisé grâce à Bootstrap.

```
<div class="navbar-nav">
  <a class="nav-item nav-link active"
href="http://localhost:3000/php/index.php">Accueil</a>
  <a class="nav-item nav-link active" href="http://localhost:3000/php/docu.php">
Documentation</a>
  <a class="nav-item nav-link active"
href="http://localhost:3000/php/articles.php">Articles</a>
</div>
```

Ce qui donne :



Accueil Documentation Articles

## 4.4 Création de la page principale et mise en commun avec la BDD

Comme tout site, il y faut la page d'accueil, celle-ci contient plusieurs pages chacune faites pour une partie de l'accueil. La page primaire, celle qu'on voit quand un utilisateur va sur le site, celle de transition, et la page qui sert à afficher le CV et les informations personnelles.

Mais avant tout cela, il faut faire les tables de la base de données permettant de stocker les données qu'on veut.

<input type="checkbox"/>	1	id	int		Non	Aucun(e)	AUTO_INCREMENT	Modifier	Supprimer	Plus
<input type="checkbox"/>	2	experiences	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus
<input type="checkbox"/>	3	presentation	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus
<input type="checkbox"/>	4	competences	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus
<input type="checkbox"/>	5	diplomes	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus

Et

<input type="checkbox"/>	1	id	int		Non	Aucun(e)	AUTO_INCREMENT	Modifier	Supprimer	Plus
<input type="checkbox"/>	2	sujet	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus
<input type="checkbox"/>	3	mail	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus
<input type="checkbox"/>	4	raison	text	latin1_swedish_ci	Non	Aucun(e)		Modifier	Supprimer	Plus

La page primaire demande pas beaucoup d'efforts, en effet la seule chose qu'il faut c'est un bon titre et une image fait de cette manière :

```
<body>
<center><h1><div class="premierblocktitre"> Portfolio johann Dietrich !</div></h1></center>
<br>
<section id="sec-1">

You, 5 hours ago • Uncommitted changes
```

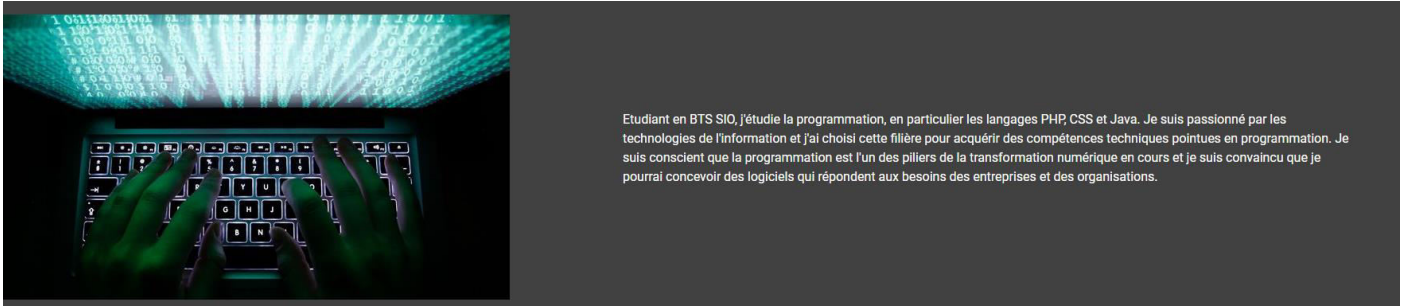




Il en va de même avec la page de transition, créée à l'aide d'une image, d'un texte et d'un peu de CSS.

```
<div class="imagindex"></div><br><br>
  <div class="center_paragraphe">Etudiant en BTS SIO, j'étudie la programmation, en
particulier... </div>
```

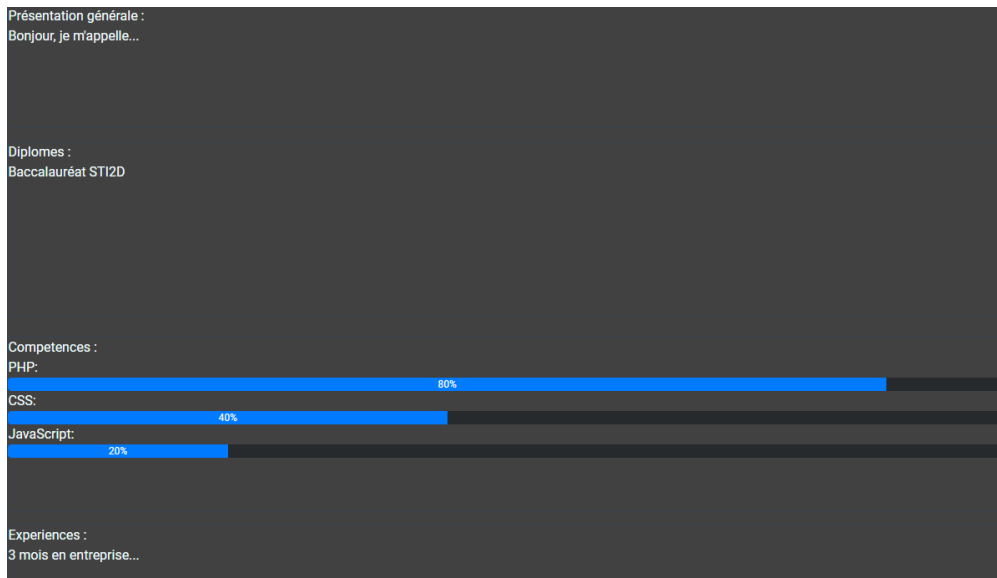
Ce qui donne un résultat ci-contre.



La partie CV diffère du reste. Celle-ci utilise une base de données afin de pouvoir être modifiée facilement et de manière dynamique. Pour cela il faut commencer par récupérer les données de la base de données, puis les récupérer et les insérer dans le CV qu'on veut afficher.

```
/* Recherche dans la BDD les données à insérer dans le CV */
$user = $conn->query("SELECT * FROM datacv ORDER BY id DESC LIMIT 1")->fetch();
$experiences = $user[1];
$presentation = $user[2];
$competences = $user[3];
$diplomes = $user[4];

<div class="cvprincipal">
  <div class="premiercontainercv">
    <div class="presentationgeneral"> Présentation générale :
      <br> <?php echo $presentation;?>
    </div>
    <div class="photo"></div>
  </div>
  <div class="deuxiemecontainercv">
    <div class="contincontcv">
      <div class="diplome"> Diplomes :
        <br> <?php echo $diplomes;?>
      </div>
    </div>
  </div>
</div>
```



En plus de tout cela, j'ai rajouté un formulaire de contact en modal, permettant aux visiteurs d'envoyer un message. Celui-ci demandant du PHP et du Javascript en plus permettant d'appeler pour montrer le formulaire et stocker les données dans la base de données.

```
<div id="myModal" class="modal">
  <div class="container1">
    <center>
      <div class="form_signup form-login">
        <div>Si vous voulez me contacter, veuillez insérer les informations
demandées ci-dessous et je vous joindrais aussi rapidement que possible !</div>
        <br><br>
        <form action="Index.php" name="form" id="form" method="post">
          <div class="form-group">
            <label for="sujet">Sujet</label>
            <input type="text" class="form-control" id="sujet"
name="sujet" placeholder="sujet" required>
          </div>
          <button type="submit" class="btn btn-primary">Envoyer</button>
        </form>
        <div><button class="fermer">fermer</button></div>
      </div>
    </center>
  </div>
</div>
```

```
if (isset($_POST['sujet'])) {
    $sujet = $_POST['sujet'];
    $mail = $_POST['mail'];
    $raison = $_POST['raison'];

    $sql = "INSERT INTO contact (id, sujet, mail, raison) VALUES (?, ?, ?, ?)";
    $conn->prepare($sql)->execute([null, $sujet, $mail, $raison]);
}
```

```
var modal = document.getElementById("myModal");

var btn = document.getElementById("myBtn");

var span = document.getElementsByClassName("fermer")[0];

btn.onclick = function() {
  modal.style.display = "block";
}

span.onclick = function() {
  modal.style.display = "none";
}
```

Si vous voulez me contacter, veuillez insérer les informations demandées ci-dessous et je vous joindrais aussi rapidement que possible !

Sujet

Mail
















Contenu

Envoyer

fermer

## 4.5 Formulaire de connexion

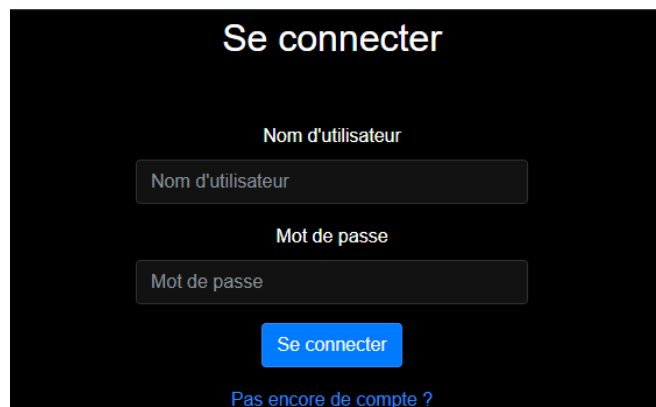
Afin de préparer le terrain pour un formulaire de connexion (et création de compte), il faut commencer par créer la table dans la base de données qui y sera adapté. J'ai choisi de prendre une table avec 5 valeurs.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 id	int			Non	Aucun(e)		AUTO_INCREMENT	 Modifier  Supprimer  Plus
<input type="checkbox"/>	2 username	text	latin1_swedish_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	3 mail	text	latin1_swedish_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	4 password	text	latin1_swedish_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	5 droits	tinyint(1)			Oui	NULL			 Modifier  Supprimer  Plus

L'ID de l'utilisateur, le nom de celui-ci, le mail, le mot de passe et les droits qu'il lui sont donnés. Dans notre cas, les droits vont uniquement servir afin de savoir si un utilisateur est un administrateur ou non.

Une fois cette partie faite, il reste donc la page de connexion à faire afin de pouvoir se servir de cette table.

La page est constituée donc d'un simple formulaire, appelant vers une fonction. Celle-ci permettant de vérifier si l'utilisateur existe bien, et donc s'il peut bien s'y connecter.



```
<div class="form_signup form-login">
  <div class="formtitle">Se connecter</div>
  <br><br>
  <form action="connection_util.php" name="form" id="form" method="post">
    <div class="form-group">
      <label for="username">Nom d'utilisateur</label>
      <input type="text" class="form-control" id="username"
name="username" placeholder="Nom d'utilisateur" required>
    </div>
    <div class="form-group">
      <label for="password">Mot de passe</label>
      <input type="password" class="form-control" id="password" name="password"
placeholder="Mot de passe" required>
    </div>
    <button type="submit" class="btn btn-primary">Se connecter</button>
  </form>
```

Ce petit formulaire appelant donc la fonction

```
se_connecter($conn)
qui elle-même fait :
```

```
$stmt = $conn->prepare("SELECT password FROM users WHERE username=?");
$stmt->execute([$_POST["username"]]);
$passwd = $stmt->fetch();
```

Ce qui récupère les données de la base de données afin de les comparer à ce que l'utilisateur a entré.

```
if (password_verify($_POST["password"], $passwd[0])){
    $_SESSION["username"] = $_POST["username"]; /* Crée les variables de sessions
permettant donc de confirmer la connection et plus */
}
else
{
    echo "Mauvais mot de passe ou nom d'utilisateur";
}
```

En cas de succès, il crée une session avec le nom de l'utilisateur et en cas d'erreur, il fait en sorte de donner un message d'erreur.

Maintenant, il nous faut également le moyen de créer efficacement des utilisateurs. Nous allons donc créer une page de création de compte.

Celle-ci fonctionne sous le même principe que celle de connexion avec un formulaire, qui cette fois-ci insère des données dans la base de données au lieu de les comparer.

Le formulaire aura donc l'air de cela

Créer un compte
Nom d'utilisateur :
<input type="text"/>
Adresse mail :
<input type="text"/>
Mot de passe :
<input type="password"/>
Confirmation du mot de passe :
<input type="password"/>

```
<center>
<div class="form_signup form-login">
  <div class="formtitle">Créer un compte</div>
  <br><br>
  <form action='creation_compte.php' name="form" id="form" method="post">
    <div class="form-group">
      <label for="username">Nom d'utilisateur :</label>
      <input type="text" class="form-control" id="username"
name="username" placeholder="Nom d'utilisateur" required>
    </div>
  </form>
</div>
```

Tout ceci lui permettant d'appeler une fonction, la fonction creercompte qui celle-ci insère les données comme montré ci-dessous :

```
$stmt = $conn->prepare("SELECT username FROM users WHERE username=:username");
$stmt->execute(['username' => $_POST["username"]]);
$user = $stmt->fetch();
if (empty($user)){
```

```

        $usernamebdd = "";
    }
    else { $usernamebdd = $user[0]; }

    $stmt = $conn->prepare("SELECT mail FROM users WHERE mail=:mail");
    $stmt->execute(['mail' => $_POST["mail"]]);
    $user = $stmt->fetch();
    if (empty($user)){
        $mailbdd = "";
    }
    else { $mailbdd = $user[0]; }

    if($_POST['password']== $_POST['confpassword']){ /* vérifie si les mots de passes
sont les memes */
        if ($_POST['username']== $usernamebdd || $_POST['mail']== $mailbdd) { /*
vérifie si le nom d'utilisateur ou l'adresse mail ne sont pas déjà utilisés */
            echo "nom d'utilisateur ou adresse mail déjà utilisés";
        }
        else {
            $mdphash = password_hash($_POST['password'], PASSWORD_DEFAULT);
            $sql = "INSERT INTO users (username, mail, password) VALUES (?, ?, ?)";
            $conn->prepare($sql)->execute([$_POST['username'], $_POST['mail'],
$mdphash]);
            header("location:articles.php");
            exit;
        }
    }
}

```

Code assez long, qui s'occupe de vérifier si le nom d'utilisateur ou l'adresse mail ne sont pas déjà utilisés, ceci en regardant dans la BDD si ceux-ci existent déjà. Il vérifie également si les deux mots de passes entrés sont les mêmes, et si le tout est bon, le mot de passe se fait hashé, puis inséré dans la base de données, dans la table « user » avec toutes les informations.

Avant de conclure la partie de connexion il y a une dernière chose à faire, et c'est définir quel utilisateur est un administrateur. Pour cela, rien de plus simple. Dans la BDD il suffit d'indiquer que les droits d'un utilisateur choisi sont égaux à 1, et non pas 0.

Une fois cela fait, il suffit de rajouter la ligne ci-dessous dans la fonction de connexion afin de définir une session qui possède justement la valeur des droits de l'utilisateur qui s'est connecté (0 ou 1)

```

$_SESSION["droits"] = $droits;

```

## 4.6 Page de documents/projets

La page de projets est probablement la moins technique et la plus facile à faire de toutes, elle a uniquement besoin d'un tableau contenant les informations sur les projets et d'une boucle permettant de les afficher correctement.

En effet, le tableau est représenté comme tel :

```
$projets = array(
    array("titre" => "E4 - Mise en place d'une page administrateur permettant d'éditer
les méta description des pages web.", "image" => "/images/seo.jpg", "lien" => "docs\E4 -
Mise en place d'une page administrateur permettant d'éditer les méta description des pages
web..pdf"),
    array("titre" => "E4 - Mise en place d'un système d'affichage, d'ajout, de
suppression et de modification d'articles de vente", "image" => "/images/articles.PNG",
"lien" => "docs\E4-
Mise_en_place_d'un_système_d'affichage,_d'ajout,_de_suppression_et_de_modification_d'artic
les_de_vente.pdf"),
)
```

Donc avec chaque projet ayant un titre, une image et un lien pour accéder à la documentation de celui-ci. Et ceci pour tous les projets qu'on veut mettre.

Et la deuxième partie étant des boucles

```
for ($i=0; $i < count($projets); $i+=4) {
    echo '<div class="row mb-5">';
    for ($j=0; $j < 4 && $i+$j < count($projets); $j++) {
        $projet = $projets[$i+$j];
        echo '<div class="col-md-3">';
        echo '<a href="'. $projet['lien'].'" class="thumbnail">';
        echo '';
        echo '<div class="caption">';
        echo '<h3>'. $projet['titre']. '</h3>';
        echo '</div>';
        echo '</a>';
        echo '</div>';
    }
    echo '</div>';
}
```

Celles-ci faisant en sorte que, grâce à des classes Bootstrap, mettent en place 4 articles par ligne permettant de les ordonner de façon propre. Sachant que chaque boucle met donc en place le titre de l'article, un lien cliquable vers celui-ci et une image de présentation. Le résultat final donnant donc un chose comme tel :

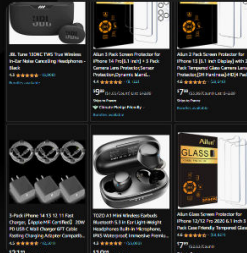
## Projets E4 et E5



E4 - Mise en place d'u...

Référence
81905453
80166979
82039106
81968500
74836295
67337361
76292500
81900760
73923500
74836246
74836372
73708264

E4 - Mise en place d'u...



E4 - Mise en place d'u...



E4 - Mise en place d'u...



E4 - Installation et séc...

full

☐ signe
 ☐ arrêt
 ☒ plein
 ☐ montre

E4 - Création d'une p...



E5 - Portfolio

	Référence	Prix
verre	81905453	8,90 €
verre de verre	80166979	21,80 €
bois	82039106	14,80 €
verre	81968500	12,80 €
plac	74836295	1,95 €
bois plac	67337361	5,10 €
bois plac	76292500	2,90 €
	81900760	40,00 €
	73923500	34,90 €
	74836246	3,20 €
	74836372	17,90 €
Forme	73708264	3,95 €

E5 - Bricobrac



## 4.7 Page d'affichage d'articles

Pour commencer notre page, il nous faut déjà avoir une table dans notre base de données afin de pouvoir stocker nos articles. Celle-ci ressemble donc à ça :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 id	int			Non	Aucun(e)		AUTO_INCREMENT	Modifier  Supprimer  Plus
<input type="checkbox"/>	2 titre	tinytext	latin1_swedish_ci		Non	Aucun(e)			Modifier  Supprimer  Plus
<input type="checkbox"/>	3 contenu	longtext	latin1_swedish_ci		Non	Aucun(e)			Modifier  Supprimer  Plus
<input type="checkbox"/>	4 auteur	tinytext	latin1_swedish_ci		Non	Aucun(e)			Modifier  Supprimer  Plus

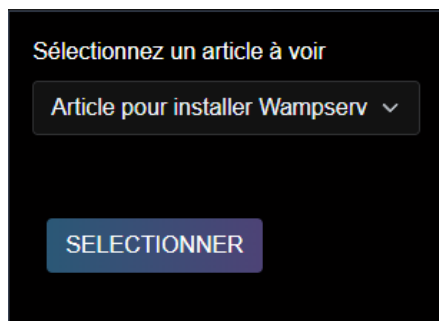
Permettant donc d'insérer le titre de celui-ci, le contenu et son auteur.

Maintenant que nous avons cela, nous pouvons faire les parties principales de cette page : Le sélectionnément et l'affichage d'articles.

Comme nous voulons faire en sorte que l'article n'apparaisse uniquement s'il est sélectionné, il paraît logique de faire le sélecteur en premier. Celui-ci est fait assez facilement :

```
<div class="selection_article">
<label>Sélectionnez un article à voir</label>
<br>
<form class="selecticle" method="POST">
    <select name="select" class="form-select" aria-label="Default select example">
        <?php
        $data = $conn->query("SELECT * FROM articles")->fetchAll();
        foreach ($data as $row)
        {
            echo "<option value=$row[id]> $row[titre] </option>";
        }
        ?>
    </select>
    <br><br><button type="submit" class="btn btn-form-color">Selectionner</button>
</form>
</div>
```

Celui-ci récupérant simplement les informations de la BDD permettant d'afficher le titre de chaque article en tant qu'option sélectionnable que l'on peut choisir avec le bouton.



Sélectionnez un article à voir

Article pour installer Wampserv ▼

SELECTIONNER

Une fois le bouton cliqué, on va simplement indiquer à la page qu'un formulaire avec une valeur a été envoyée et que si c'est effectivement le cas, que celui-ci lance plusieurs fonctions.

```

if (!empty($_POST))
{
    $article_choisi = choisir_article($conn);
    $infos_article = obtenir_article($conn,$article_choisi['id']);
}

```

Une première pour récupérer l'article voulu :

```

$idactuelle = $_POST['select'];
$stmt = $conn->prepare("SELECT * FROM articles WHERE id=:id");
$stmt->execute(['id' => $idactuelle]);
$article = $stmt->fetch();
return $article;

```

Et la seconde pour récupérer toutes les informations liées à cet article et de les mettre en tant que variables qui vont pouvoir être affichées :

```

$stmt = $conn->prepare("SELECT * FROM articles WHERE id=?");
$stmt->execute([$id_article_choisi]);
$infos_article = $stmt->fetch();

return ([
    'id' => $infos_article["id"],
    'titre' => $infos_article["titre"],
    'contenu' => $infos_article["contenu"],
    'auteur' => $infos_article["auteur"]
]);

```

Grâce à tout cela, il ne nous reste plus qu'à s'occuper de l'affichage lui-même de l'article sélectionné. Ceci fait assez facilement.

```

<?php
    if(!empty($_POST)){
        unset($_SESSION['article_id']);
        $_SESSION['article_id'] = $infos_article['id'];
    }

    <!-- Partie montrant l'article choisi -->
    <div class="main_article">
        <div class="titre_article">
            <?php echo '<h2>'. $infos_article['titre'].'</h2>'?>
        </div>
        <div class="contenu_article">
            <?php echo '<h4>'. $infos_article['contenu'].'</h4>' ?> <br><br>
        </div>
            <?php echo '<h5>'. $infos_article['auteur'].'</h5>'?>
        </div>
        <?php require_once("comm.php"); ?>
    </div>
    <?php }
    else{
        echo "<br><br><center><div class='white'><h2>Veuillez choisir un article à la
gauche de l'écran </h2></div></center>";
    }

```

En effet, il suffit de lui indiquer que si le formulaire a bien été envoyé et qu'un article a bien été choisi, qu'il affiche les parties qui sont remplies automatiquement avec le contenu des variables (justement récupérées de la BDD).

## 4.8 Partie commentaires

En supplément de nos articles, il y à également la mise en place d'un petit espace commentaire en dessous des articles permettant aux gens d'échanger.

Comme pour toutes informations qu'il faut stocker, il faut commencer par créer une table dans notre BDD qui nous servira à ressortir les informations que l'on veut.

<input type="checkbox"/>	1	id		int		Non	Aucun(e)	AUTO_INCREMENT		Modifier		Supprimer		Plus
<input type="checkbox"/>	2	contenu		varchar(128)	latin1_swedish_ci	Non	Aucun(e)			Modifier		Supprimer		Plus
<input type="checkbox"/>	3	article_id		int		UNSIGNED	Non	Aucun(e)		Modifier		Supprimer		Plus
<input type="checkbox"/>	4	user_id		int		UNSIGNED	Non	Aucun(e)		Modifier		Supprimer		Plus

Notre table ressemble à ça, l'ID d'article nous servant à savoir sur quel article le commentaire doit apparaitre et l'ID d'utilisateur, a savoir qui a écrit celui-ci.

Maintenant que cela est fait il nous faut continuer avec les deux grandes parties : La partie permettant d'afficher et la partie permettant d'ajouter un commentaire.

Pour la partie d'affichage, on appelle simplement une fonction :

```
<?php echo appel_com($conn);?>
```

Celle-ci récupérant l'ID de l'article actuel et s'en sert pour sélectionner tout les commentaires de cet article choisi.

Grâce a cela, on peut faire une boucle qui affiche tout les commentaires de l'article, ou qui affiche « pas encore de commentaires » s'ils n'y en a pas sur l'article choisi.

```
$idactuelle = $_POST['select'];
$stmt = $conn->prepare("SELECT * FROM commentaires WHERE article_id=:id");
$stmt->execute(['id' => $idactuelle]);
$commentaires = $stmt->fetchAll(); //selectionne les données de tous les
commentaires de l'article a l'id actuel
if (!empty($commentaires)){
    $var = "";
    foreach ($commentaires as $commentaire){
        $stmt = $conn->prepare("SELECT username FROM users WHERE id=?");
        $stmt->execute([$commentaire['user_id']]); //cherche le nom associé au
commentaire
        $nom_util= $stmt->fetch();
        $var .="<div class='boite_de_commentaire'> Commentaire de
".$nom_util['username']. " : <br>".$commentaire['contenu']. "</div>";
    }
    return $var;
}
else {
    return "<div class='boite_de_commentaire'> Pas encore de commentaires </div>";
}
```

Commentaire de admin :  
esreteerert

La prochaine partie est un peu différente, même si celle-ci est simple dans son fonctionnement, il faut mettre des restrictions au fait de pouvoir laisser un commentaire.

```
<div class="boite_de_commentaire">
    <?php if (isset($_SESSION["username"])){?>
        <form action="ajouter_com.php" name="form" id="form" method="post">
            <div class="form-group black">
                Un commentaire ?
                <input type="text" class="form-control" id="commentaire"
name="commentaire" placeholder="..." required>
            </div>
            <center><button type="submit" class="btn btn-
dark">Publier</button></center>
        </form>
    <?php } else { echo "veuillez vous connecter pour mettre des commentaires"; }?>
</div>
```

En effet, le code est assez simple, c'est un simple formulaire qui s'occupe à insérer des données dans la BDD. Mais il faut noter qu'il y a une restriction pour pouvoir mettre un commentaire, il faut être connecté et donc avoir un nom d'utilisateur utilisable.

```
$sql = "INSERT INTO commentaires (contenu, article_id, user_id) VALUES (?, ?, ?)";
$conn->prepare($sql)->execute($_POST['commentaire'], $id_article,
$_SESSION['id']);

header("location:articles.php");
exit;
```

Nom d'utilisateur qui sera donc utilisé dans le commentaire pour indiquer l'auteur de celui-ci.

## 4.9 Page administrateur

La dernière partie et celle qui est la plus utilitaire de tout le site, la page administratrice. C'est avec celle-ci que l'on va pouvoir créer des articles modifier le CV et voir les logs des comptes qui se sont connectés et déconnectés.

Il est important que cette page ne puisse pas être accessible par tous, il faut qu'uniquement les utilisateurs avec un certain droit puissent y accéder. Dans notre cas, c'est fait de telle manière :

```
if(isset($_SESSION["username"]))
{
    if($_SESSION["droits"] == 1)
    {
```

Ceci nous permettant simplement de vérifier si l'utilisateur est connecté et si celui-ci a bien les droits nécessaires pour entrer la page. A partir de là, le reste devient assez simpliste.

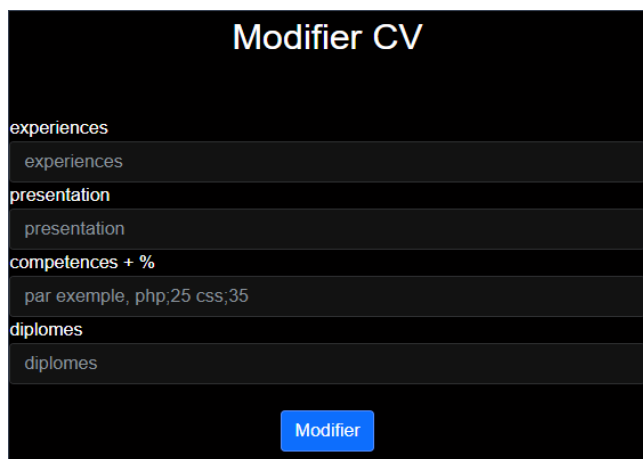
```
<div class="formtitle">Modifier CV</div> </center>
    <br><br>
    <form action="admin.php" name="form" id="form" method="post">
        <div class="form-group">
            <label for="experiences">experiences</label>
            <input type="text" class="form-control" id="experiences"
name="experiences" placeholder="experiences" required>
        </div>
        <br>
        <center><button type="submit" class="btn btn-primary">Modifier</button></center>
    </form>
</div>
```

On crée un formulaire permettant d'envoyer les informations directement dans notre BDD pour que le CV les réutilisent.

```
$experiences = $_POST['experiences'];
$presentation = $_POST['presentation'];
$competences = $_POST['competences'];
$diplomes = $_POST['diplomes'];

$sql = "UPDATE datacv SET experiences = ?, presentation = ?, competences = ?,
diplomes = ? WHERE id = 1";
$conn->prepare($sql)->execute([$experiences, $presentation, $competences,
$diplomes]);
```

Avec une simple commande SQL permettant de mettre à jour la BDD avec les informations données.



Notre prochaine parti, pour ajouter un article, n'est pas plus difficile que cela. En effet, il s'agit de la même chose. Un simple formulaire permettant d'envoyer des données a la BDD avec une commande SQL.

```
$sql = "INSERT INTO articles (titre, contenu, auteur) VALUES (?, ?, ?)";
$conn->prepare($sql)->execute([$_POST['titre'], $_POST['contenu'],
$_SESSION["username"]]);
header("location:articles.php");
exit;
```

## Ajouter article


titre

contenu de l'article

Créer l'article

Notre dernière partie, un peu plus intéressante est liée au logs. Servants donc a garder une trace dès qu'un utilisateur se connecte/se déconnecter. Il semble donc logique qu'il faille faire des changement également dans les fonctions gérant la connexion et la déconnexion, tout en ayant besoin d'une table dans la BDD afin de les stocker quelque part.

En premier lieu il faut donc créer notre table :


#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/>	1 id 	int			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/>	2 username	tinytext	latin1_swedish_ci		Non	Aucun(e)		
<input type="checkbox"/>	3 actions	tinytext	latin1_swedish_ci		Non	Aucun(e)		
<input type="checkbox"/>	4 pages	tinytext	latin1_swedish_ci		Non	Aucun(e)		
<input type="checkbox"/>	5 date_temps	datetime			Non	Aucun(e)		

Celle-ci permettant donc de créer des logs contenant l'utilisateur, l'action qu'il a faite, la page sur laquelle l'action a été faite et bien sur la date et l'heure ou l'action a été faite.

On commence donc par les parties de déconnexion et connexion :

```
$sql = "INSERT INTO logs (username, actions, pages, date_temps) VALUES (?, ?, ?, ?)";
//logs de déconnection
$conn->prepare($sql)->execute($_SESSION["username"], "Logout/Changement de compte",
"connection.php", time_now());
```

Les deux étant juste un bout de code a rajouter lors de l'appel des fonctions permettant d'insérer les valeurs qu'on veut dans la BDD avec les logs.

<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	72 uti	Logout/Changement de compte connection.php 2023-03-26 11:51:07
--------------------------	--	--	---	--------	--

Après cela il ne reste plus que l'affichage de ceux-ci sur la page admin. Se faisant assez facilement, encore une fois en récupérant les données de la BDD et en lui faisant faire une boucle afin d'afficher tous les éléments existants avec une légère tendance de style :

```

<div class="form_signup affichage_log">
    <ul>
        <?php
            $stmt = $conn->prepare("SELECT * FROM logs");
            $stmt->execute();
            $logs = $stmt->fetchAll();
            $logs = array_reverse($logs);
            foreach ($logs as $logs) {
                echo "<li>|| Utilisateur : " . $logs['username'] . " || Action : ".
$logs['actions']. " || Date et heure : ".$logs['date_temps'] . " || </li>";
            }
        ?>
    </ul>
</div>

```

Ce qui nous donne ceci :

- || Utilisateur : johann || Action : Login || Date et heure : 2023-04-20 15:10:07 ||
- || Utilisateur : johann || Action : Logout/Changement de compte || Date et heure : 2023-04-20 14:26:07 ||
- || Utilisateur : johann || Action : Login || Date et heure : 2023-04-20 14:25:42 ||
- || Utilisateur : johann || Action : Logout/Changement de compte || Date et heure : 2023-04-20 14:25:32 ||

## 5 Gestion de la maintenance (corrective / évolutive)

---

### 5.1 Mise à jour de la documentation du SI

---

Recueillir, analyser et mettre à jour les informations sur une version d'une solution

L'entièreté de la documentation et du fonctionnement peut être trouvé à l'intérieur de cette documentation, donc en cas de quelconque problème, il faudrait toujours s'adresser en priorité à cette documentation.

Mettre à jour des documentations technique et d'utilisation d'une solution

En cas de mise à jour de cette documentation, la version indiquée au début du document devra être changée afin de pouvoir savoir que celle-ci est à jour.

### 5.2 Evaluation de la qualité de la solution

---

Évaluer la qualité d'une solution

Le portfolio répond à tout les critères de qualités demandés, notamment sur la visibilité et son fonctionnement général.

### 5.3 Procédure de correction d'un dysfonctionnement

---

Analyser et corriger un dysfonctionnement

En cas de dysfonctionnement sur le site en direct, si le dysfonctionnement est mineur, le site peut continuer à tourner et une solution va être donc travaillée sur les fichiers en local.

Mais si le problème est majeur et mets en danger le site, alors un fichier « index.php » de remplacement sera mis à la place du portfolio, indiqué que le site a besoin d'une maintenance. Et ceci jusqu'à ce qu'une solution ait été trouvée, encore une fois, en local.

### 5.4 Gestion des tests de mise à jour

---

Élaborer et réaliser les tests des éléments mis à jour

Simplement l'utilisation de la procédure de test unitaire, d'abord en local, puis sur le serveur afin de vérifier le fonctionnement des deux et/ou régler un problème s'il y en a un.



## **6 Bilan du projet**

---

### **6.1 Validation des exigences point par point**

---

Besoin d'afficher le CV et de pouvoir le télécharger

Besoin d'un système de contact via le site directement

Besoin d'une page permettant la présentation de toutes les documentations effectuées

Besoin d'une page d'article pour montrer les compétences personnelles

Besoin d'un système d'utilisateur avec différentes fonctions selon les pages

Besoin d'un affichage graphique visible et compréhensible par tout le monde et sur tous les écrans

### **6.2 Axes d'amélioration**

---

Un ajout de potentiel personnalisation du site par l'utilisateur, tel un mode obscure et clair pour le site.

### **6.3 Compétences acquises**

---

Utilisation de PHP et MySQL pour la création de sites dynamiques et efficaces

Utilisation partielle de Bootstrap pour la mise en place d'éléments

Utilisation de CSS et de JS pour créer des animations