RAPPORT DE STAGE

Stage de programmation à la mairie de Maing

Avril 2023 - Juin 2023

Axel GIRARD
Etudiant en Licence 3 Informatique

Enseignant référent - Emmanuel ADAM
Maître de stage - Julien NEF

Année scolaire 2022-2023

Sommaire

Remerciements

Je tiens à remercier toutes les personnes, qui par leurs conseils et leur accompagnement, m'ont permis de réaliser mon stage dans les meilleures conditions.

Je tiens tout d'abord à remercier mon maître de stage Mr Julien NEF, responsable de l'espace numérique de la mairie de Maing, pour la confiance accordée et ses conseils précieux qui m'ont permis de réaliser un stage non seulement intéressant, mais également instructif.

Je souhaiterai également remercier Mme Audrey BAERT ma mère, pour avoir pris le temps de lire mon rapport et d'en avoir corrigé une partie.

Enfin, je remercie la mairie de Maing et tout particulièrement les personnes s'occupant de l'administration pour leur accueil et leur bonne humeur communicative.

Introduction

Au cours de ma troisième année de licence informatique, j'ai eu l'opportunité d'effectuer un stage à la mairie de Maing, d'avril à juin 2023. Ce stage a été une occasion unique pour moi de mettre en pratique les connaissances acquises au cours de ma formation et de développer de nouvelles compétences dans le domaine de la programmation web.

L'objectif principal de ce stage était de concevoir et de programmer un site web au sein de l'intranet de la mairie, permettant la gestion et l'entretien des systèmes de sécurité. La mairie de Maing accorde une grande importance à la sécurité de ses infrastructures et de ses ressources, et il était donc essentiel de mettre en place un outil efficace pour assurer la surveillance et la maintenance de ces systèmes.

Après avoir présenté l'entreprise, je présenterai les différentes étapes du processus de développement du site web, en mettant l'accent sur les choix techniques et les fonctionnalités implémentées. Je décrirai également les différentes problématiques rencontrées et les solutions adoptées pour les résoudre.

Ce stage a été une expérience enrichissante à bien des égards. Il m'a permis de développer mes compétences en programmation web, en travaillant avec des technologies telles que HTML, CSS, JavaScript et PHP. J'ai également pu acquérir de nouvelles compétences en PHP, plus particulièrement dans l'utilisation de PDO et de phpMyAdmin

En conclusion de ce rapport, j'analyserai les résultats obtenus et je ferai part de mes réflexions sur les apprentissages réalisés durant ce stage. Je soulignerai également les perspectives d'évolution et d'amélioration du site web.

Présentation de l'entreprise

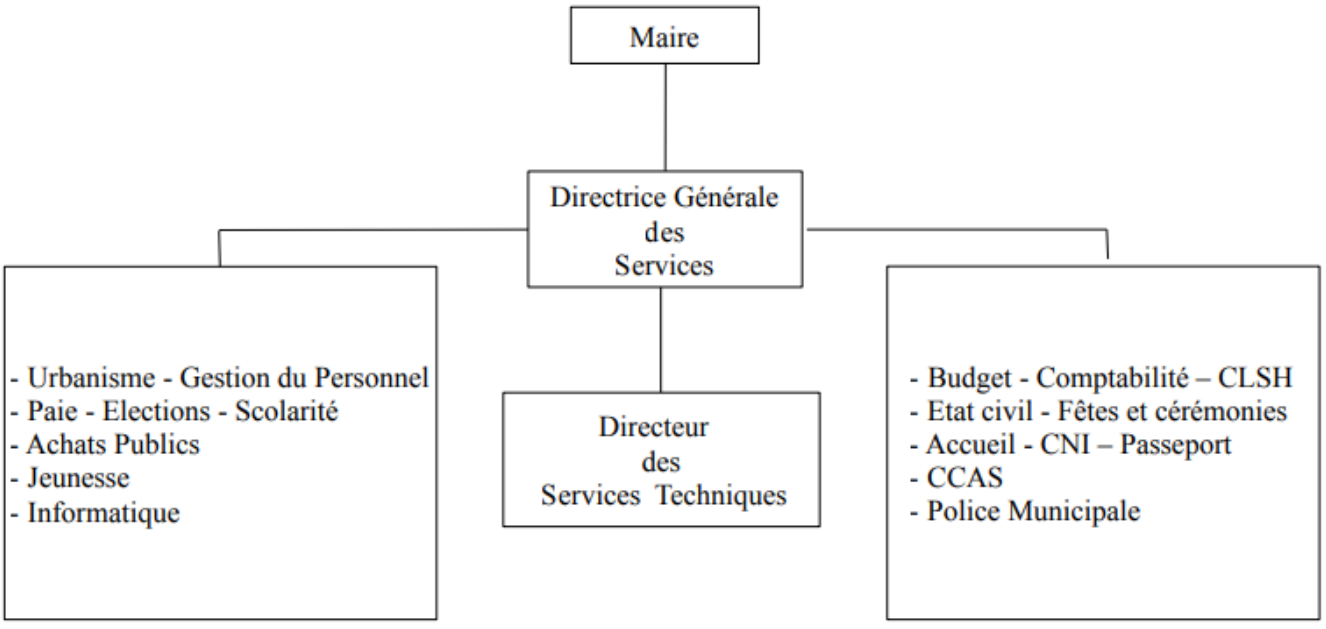
Maing est une petite ville située à 6 km au Sud-Ouest de Valenciennes. C'est une commune urbaine rattachée à Valenciennes Métropole.

La commune s'étend sur 11,7 km² et compte environ 4000 habitants depuis le dernier recensement de la population datant de 2006.

Ses habitants sont appelés les Maingeois et les Maingeoises.
Le maire de Maing se nomme M. Philippe BAUDRIN.

La Mairie a été construite rue Jean Jaurès par Jean-Baptiste Bernard en 1847. Elle se caractérise par sa façade du Second-Empire.

Elle est composée d'un pôle administratif dirigé par Mlle SERAFINI Directrice Générale des Services (DGS) et d'un pôle technique dirigé par M. BALBONA.



Organigramme des services de la Mairie

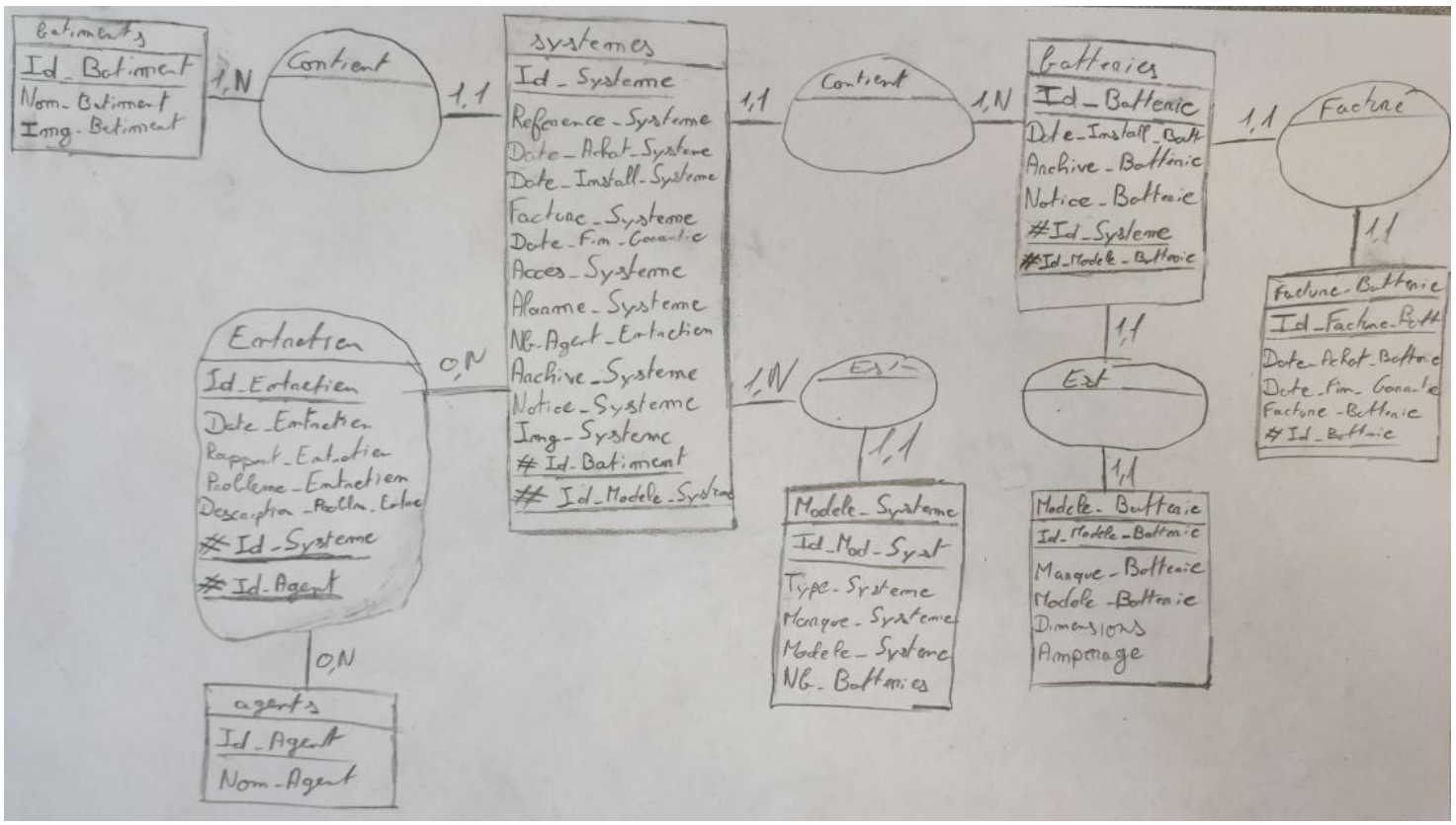
Le service informatique est assuré par M. NEF. Il gère le réseau, donne des cours à l'espace numérique, développe des applications via l'intranet et le site web de la commune (http://www.maing.fr).

Solution apportée

Lors des premiers jours du stage, une réunion a été organisée afin de discuter du projet et établir un cahier des charges, répertoriant toutes les fonctionnalités qui devront être mises en place :

- Archivage des batteries et systèmes
- Ajout entretien, batterie et système
- Ajout de modèles de système
- Ajout de modèles de batteries
- Ajout d'agents
- Accès au factures et aux notices
- Affichage de toutes les informations des systèmes
- Affichage de tous les entretiens pour chaque système
- Affichage de toutes les informations des batteries pour chaque système

Le modèle Entité/Association de la base de données, à ensuite été réalisé afin de faciliter la création de la base de données ainsi que son utilisation.



Après plusieurs modifications et la validation du modèle, une maquette du site détaillant les différents éléments qui y seront présents a été réalisée.

Par la suite,la base de données qui va contenir toutes les informations des bâtiments, systèmes et batteries à été créée.
Cette base de données se compose de 8 tables :

- agents
- batiments
- batteries
- entretien
- facture_batterie
- modele_batterie
- modele_systeme
- systemes

Une fois que la base de données a été créée, la création du site a pu commencer. Pour cela, les langages HTML, PHP, CSS et JavaScript ont été utilisés.
La première étape a été de créer une fonction PHP appelée **connexion.php**, qui permet de connecter le site à la base de données.

```
try{
    $pdo = new PDO("mysql:host=localhost;dbname=mairie", "root", "");
    }
    catch(PDOException $e){
        echo $e->getMessage();
}
```

La fonction, essaie de se connecter à la base de données *mairie* depuis “localhost” avec l’identifiant “root” et le mot de passe “ ”
Si la fonction n’arrive pas à se connecter à la base de données, un message d’erreur s’affichera.

Par la suite, la page **index.php** qui est la première page qui s'affiche à l'arrivée sur le site a été créée.
Sur cette page, se trouve plusieurs boutons, tous composés de l’image d’un des bâtiments et de son nom.

Ces informations, sont récupérées grâce à la requête SQL suivante :

```
$stmt = $pdo->prepare("SELECT * FROM batiments");
$stmt->execute();
```

On utilise ensuite le résultat de la requête afin d'afficher les données :

```
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
      $filename = $row['Img_Batiment'];
      $filepath = "Images/Batiments/$filename";
      $nom = $row['Nom_Batiment'];
      echo "<div class='photo'>
            <a href='ListeSysteme.php?id_batiment={$row['Id_Batiment']}'>
            <img src='$filepath'><div class='nom'>$nom</div></a></div>";
}
```

Ici, on assigne chaque ligne de résultat à la variable $row grâce à la fonction fetch(PDO::FETCH_ASSOC). Grâce à cette fonction, chaque ligne est retournée sous la forme d'un tableau associatif.

Ensuite, le nom du fichier contenant l'image d'un bâtiment est récupéré depuis la colonne *Img_Batiment* et est stocké dans la variable $filename.

Le chemin d'accès au dossier contenant les images des bâtiments est stocké dans la variable $filepath.
Les noms des bâtiments sont récupérés depuis la colonne *Nom_Batiment* et sont stockés dans la variable $nom.

Un bloc HTML est ensuite généré afin d'afficher l'image, le nom du bâtiment et un lien vers une page appelée **ListeSysteme.php** avec l'identifiant du bâtiment passé en paramètre grâce à <a href='ListeSysteme.php?id_batiment={$row['Id_Batiment']}'>.

Le chemin de l'image est utilisé comme source de l'élément <img>.
Depuis la page "index.php", deux autres pages sont accessibles, la première est **ListeSysteme.php**.

Cette page affiche la liste de tous les systemes reliés au bâtiment sélectionné précédemment grâce à la requête SQL suivante.

```
$id_batiment = $_GET['id_batiment'];
$stmt = $pdo->prepare("SELECT * FROM systemes WHERE Id_Batiment = :id_batiment
AND Archive_Systeme = 0");
$stmt->bindParam(':id_batiment', $id_batiment, PDO::PARAM_INT);
$stmt->execute();
```

Cette requête sélectionne tous les systemes dont l'id du bâtiment est celui récupéré dans l'URL de la page et dont la valeur de *Archive_Systeme* est égale à 0.

Les systèmes sont ensuite affichés grâce à la même méthode que pour les bâtiments de la page **index.php**.
La seconde page accessible est **Recherche.php**
Cette page affiche les systèmes par rapport à ce qui a été entré dans la barre de recherche de la page **index.php** grâce à la requête SQL et à la fonction php suivante :

```
if(isset($_GET['s']) AND !empty($_GET['s'])){
      $cherche = htmlspecialchars($_GET['s']);
      $References = $pdo->query('SELECT Reference_Systeme FROM systemes WHERE
Reference_Systeme LIKE "%'.$cherche.'%" ORDER BY Id_Systeme ASC');
    }
```

Cette fonction vérifie dans l'URL de la page si une variable "s" existe et n'est pas vide
Ensuite, on donne à la variable $cherche la valeur de "s" avant de sélectionner les *Reference_Systeme* qui correspondent à la valeur de la variable $cherche dans la variable $References.

```
$stmt = $pdo->prepare("SELECT * FROM systemes WHERE Reference_Systeme LIKE
:query");
$stmt->bindValue(':query', "%".$_GET['s']."%");
$stmt->execute();
```

Cette requête sélectionne les systèmes dont *Reference_Systeme* correspond à ce qui à été entré dans la barre de recherche et est stocké dans la variable $References.
Comme pour la page **ListeSysteme.php** et **index.php**, la méthode pour afficher les systemes est la même.

Depuis les pages **ListeSysteme.php** et **Recherche.php**, nous avons accès à la page **DetailsSystemes.php** du système précédemment sélectionné.

Sur cette page, les informations affichées proviennent de plusieurs tables de la base de données *mairie* grâce à plusieurs requêtes SQL

```
$id_systeme = $_GET['id_systeme'];
$stmt1 = $pdo->prepare('SELECT * FROM systemes WHERE Id_Systeme = :id_systeme');
$stmt2 = $pdo->prepare('SELECT * FROM batteries WHERE Id_Systeme =
:id_systeme');
$stmt3 = $pdo->prepare('SELECT * FROM entretien WHERE Id_Systeme =
:id_systeme');
$stmt4 = $pdo->prepare('SELECT * FROM systemes WHERE Id_Systeme = :id_systeme');
$stmt5 = $pdo->prepare('SELECT * FROM modele_systeme WHERE Id_Modele_Systeme =
:id_mod_sys');
$stmt6 = $pdo->prepare('SELECT Id_Modele_Systeme FROM systemes WHERE Id_Systeme
= :id_systeme');

$stmt1->bindParam(':id_systeme', $id_systeme, PDO::PARAM_INT);
$stmt2->bindParam(':id_systeme', $id_systeme, PDO::PARAM_INT);
$stmt3->bindParam(':id_systeme', $id_systeme, PDO::PARAM_INT);
$stmt4->bindParam(':id_systeme', $id_systeme, PDO::PARAM_INT);
$stmt5->bindParam(':id_mod_sys', $id_mod_sys_value, PDO::PARAM_INT);
$stmt6->bindParam(':id_systeme', $id_systeme, PDO::PARAM_INT);

$stmt1->execute();
$stmt2->execute();
$stmt3->execute();
$stmt4->execute();
$stmt5->execute();
$stmt6->execute();
$id_mod_sys_result = $stmt6->fetch(PDO::FETCH_ASSOC);
$id_mod_sys_value = $id_mod_sys_result['Id_Modele_Systeme'];
```

Les informations sont ensuites affichées sous forme de liste grâce aux balises <ul> et
Ces informations sont toutes affichées de la même façon:

```
while($row = $stmt5->fetch(PDO::FETCH_ASSOC)){
      $idModel = $row['Id_Modele_Systeme'];
      $TypeSyst = $row['Type_Systeme'];
      $MarqueSyst = $row['Marque_Systeme'];
      $ModeleSyst = $row['Modele_Systeme'];
```

```
        $NbBatt = $row['Nb_Batteries'];
        echo "<li><h4>Id du systeme : </h4>$idModel</li>";
        echo "<li><h4>Type de systeme : </h4>$TypeSyst</li>";
        echo "<li><h4>Marque du systeme : </h4>$MarqueSyst</li>";
        echo "<li><h4>Modele du systeme : </h4>$ModeleSyst</li>";
        echo "<li><h4>Nombre de batteries : </h4>$NbBatt</li>";
}
```

Chaque ligne de résultat de la requête $stmt5 est récupérée à l'aide de la méthode fetch(PDO::FETCH_ASSOC). À chaque itération de la boucle, les valeurs des colonnes de la ligne courante sont assignées à des variables correspondantes.
Les valeurs stockées dans les différentes variables sont affichées sous forme d'une liste HTML à l'aide de la fonction echo.

Depuis cette page, plusieurs pages sont accessibles, la première est :
AjoutEntretien.php.
Cette page permet d'ajouter un entretien à la liste de ceux ayant déjà été effectués sur le système grâce à plusieurs inputs et une fonction php.

Ces inputs font partie d'un formulaire et récupèrent les différentes valeurs qui ont été entrées.
Parmi ces inputs, un menu déroulant est présent afin de sélectionner l'agent ayant effectué l'entretien.
Ce menu à été créé en utilisant les balises <select> et <option>

```
<select id="agents" name="Id_Agent" required>
        <option disabled selected>Veuillez sélectionner un Agent</option>
        <?php
        $stmt = $pdo->prepare('SELECT Id_Agent, Nom_Agent FROM agents ORDER BY
Id_Agent ASC');
        $stmt->execute();
        while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                echo "<option value='" . $row["Id_Agent"] . "'>" . $row["Nom_Agent"]
        . "</option>";
        }?>
</select>
```

L'option sélectionnée par défaut dans le menu déroulant avec le texte est : "Veuillez sélectionner un Agent". Cette option est là pour guider l'utilisateur et l'inciter à faire un choix.

La requête $stmt récupère les colonnes *Id_Agent* et *Nom_Agent* depuis la table agents et les trie par ordre croissant par rapport à leur ID

La boucle while récupère chaque enregistrement sous la forme d'un tableau associatif grâce à la méthode 'fetch' et l'assigne à la variable $row

Une option est ensuite générée pour chaque enregistrement récupéré. Chaque option correspond au nom d'un agent lié à son ID.
Des fonctions JavaScript ont été ajoutées, la première est afficherDetails, elle permet d'afficher le champ Description_Probleme_Entretien si l'input radio "oui" est sélectionné.

```
let radioOui = document.getElementById("oui");
let radioNon = document.getElementById("non");
let sectionDetails = document.getElementById("detProb");
function afficherDetails() {
      sectionDetails.hidden = !radioOui.checked;
}
radioOui.addEventListener("change", afficherDetails);
radioNon.addEventListener("change", afficherDetails);
```

La seconde est la fonction permettant d'afficher le formulaire d'ajout d'un nouvel Agent

```
const boutonAgt = document.getElementById("boutonAgt");
const ajtAgt = document.getElementById("ajtAgt");
boutonAgt.addEventListener("click", function() {
      ajtAgt.removeAttribute("hidden");
});
```

À cette étape, un problème est survenu, comment forcer l'utilisateur à sélectionner un agent avant l'enregistrement ?
Le bouton d'enregistrement a alors été désactivé et une fonction permettant d'activer le bouton uniquement lorsqu'un agent est sélectionné à été créé.

```
const selectElement = document.getElementById('agents');
const Enregis = document.getElementById('AgentEnregist');
const initialValue = selectElement.value;
selectElement.addEventListener('change', function() {
      Enregis.disabled = selectElement.value === initialValue;
});
```

La fonction **AjouterEntretien.php** récupère toutes les données entrées dans les différents inputs afin des les ajouter à la base de données

```
if(isset($_POST['Id_Agent'])) {
      $IdAgent = $_POST['Id_Agent'];}

if(isset($_POST['Date_Entretien'])) {
      $Date = $_POST['Date_Entretien'];}

if(isset($_POST['Rapport_Entretien'])) {
      $Rapport = $_POST['Rapport_Entretien'];}

if(isset($_POST['Probleme_Entretien'])) {
      $Probleme = $_POST['Probleme_Entretien'];}

if(isset($_POST['Description_Probleme_Entretien'])) {
      $Detail = $_POST['Description_Probleme_Entretien'];}

if(isset($_POST['id_systeme'])) {
      $id_systeme = $_POST['id_systeme'];}

try {
      $req = $pdo->prepare("INSERT IGNORE INTO entretien (Date_Entretien,
Rapport_Entretien, Probleme_Entretien, Description_Probleme_Entretien,
Id_Systeme, Id_Agent) VALUES (?,?,?,?,?,?)");
      $req->execute(array($Date, $Rapport, $Probleme, $Detail, $id_systeme,
$IdAgent));}
      catch(PDOException $e) {
            echo $e->getMessage();}
header("Location: ../DetailsSystemes.php?id_systeme={$id_systeme}");
```

La fonction isset() est utilisée pour vérifier si certaines variables ont été envoyées via une requête POST. La fonction isset() renvoie true si la variable est définie et false sinon.
Si une variable est définie, la valeur est récupérée à partir de la requête POST et assignée à une variable qui lui est propre
La partie suivante utilise PDO pour préparer une requête d'insertion dans la table *entretien*.
La requête est préparée en utilisant des marqueurs de paramètres (?)
La méthode execute() est appelée sur l'objet $req avec un tableau contenant les valeurs à insérer. Les valeurs sont passées dans le même ordre que les marqueurs de paramètres dans la requête.

S'il y a une exception PDO lors de l'exécution de la requête (par exemple, une erreur de syntaxe SQL), elle est capturée et le message d'erreur est affiché à l'aide de la méthode getMessage().

L'autre fonction accessible depuis **DetailsSystemes.php** est **AjouterBatterie.php**, cette fonction effectue la même chose que **AjoutEntretien.php** mais pour les batteries.
Les fonctions supplémentaires sont les ajouts de fichiers pour la facture et la notice des batteries.
Deux autres problèmes ont alors fait leur apparition : comment enregistrer le fichier uploadé dans un dossier spécifique ? et comment éviter les doublons lors de cet enregistrement ?
La solution à ces problèmes a été d'utiliser les fonctions move_uploaded_file() et uniqid()

```
$dossier = "../Notices/";
$NoticeName = $_FILES['Notice']['name'];
$NoticeLoc = $_FILES['Notice']['tmp_name'];
$NoticeExt = pathinfo($NoticeName, PATHINFO_EXTENSION);
$NoticeNewName = uniqid().'.'.$NoticeExt;
$deplacerNotice = move_uploaded_file($NoticeLoc, $dossier.$NoticeNewName);
```

Dans un premier temps le chemin vers le dossier qui contiendra les fichiers envoyés grâce à la fonction move_uploaded_file()
Ensuite, le nom, le nom temporaire et l'extension du fichier sont stockés dans différentes variables.
La fonction uniqid est ensuite utilisée afin de créer un identifiant unique et en concaténant l'extension récupérée précédemment. Cela garantit que le nom du fichier sera unique pour éviter les conflits si plusieurs fichiers sont téléchargés avec le même nom.

Enfin, la fonction move_uploaded_file est utilisée pour déplacer le fichier depuis son emplacement temporaire vers le répertoire de destination spécifié par la variable $dossier. Le nouveau nom du fichier est donné par la variable $NoticeNewName.

La liste des batteries reliés au système est affichée sous la forme d'un tableau

Marque	Modèle	Dimensions	Amperage	Date Achat	Date d'installation	Date Fin Garantie	Facture Batterie	Notice Batterie		
Trojan	Modele N°5	Longueur : 150mm; Hauteur : 100mm; Profondeur : 70mm	12V 7AH	2023-06-05	2023-06-06	2023-06-16	64784ee4a569a.jpg	64784ee4a568c.	Archiver	Modifier
Duracell	Modele N°3	Longueur : 150mm; Hauteur : 100mm; Profondeur : 70mm	12V 7AH	2023-06-01	2023-06-01	2023-06-05	64784efa66e85.png	64784efa66e6c.	Archiver	Modifier

Lorsqu'une batterie est présente dans le système, deux nouvelles fonctions sont disponibles, **ArchiverBatt.php** et **ModifierBatt.php**.

La première permet d'archiver une batterie :

```
if(isset($_POST['archive_batterie'])){
    $id_systeme = $_POST['id_systeme'];
    $id_batterie = $_POST['id_batterie'];
    $req = $pdo->prepare('UPDATE batteries SET Archive_Batterie = 1 WHERE
Id_Batterie = :id_batterie');
    $req->bindParam(':id_batterie', $id_batterie, PDO::PARAM_INT);
    $req->execute();
    header("Location: ../DetailsSystemes.php?id_systeme={$id_systeme}");
    exit();
}
```

Si l'appuis sur le bouton "Archiver" est détecté, on récupère les valeurs des inputs *id_systeme* et *id_batterie* puis, grâce à la requête SQL, la valeur de *Archive_Batterie* est modifiée pour être égale à 1.
Après exécution de la fonction, le retour à la page **DetailsSystemes.php** est effectué grâce à la fonction PHP header()

La seconde donne l'accès à la page **ModifierBatt.php**
Cette page fonctionne comme la page d'ajouts de batterie, seule la fonction php reliée change, au lieu d'ajouter les valeurs entrés dans les inputs, on modifie les valeurs déjà stockées dans les tables *batteries* et *facture_batterie*.

```
try {
      $req = $pdo->prepare("UPDATE batteries SET Date_Installation_Batterie = ?,
Notice_Batterie = ?, Id_Systeme = ?, Id_Modele_Batterie = ? WHERE Id_Batterie =
?");
      $req->execute(array($DateInstall, $NoticeNewName, $id_systeme, $id_Modele,
$id_batterie));}
      catch (PDOException $e) {
            echo $e->getMessage();}
try {
      $req2 = $pdo->prepare("UPDATE facture_batterie SET Date_Achat_Batterie = ?,
Date_Fin_Garantie = ?, Facture_Batterie = ? WHERE Id_Batterie = ?");
      $req2->execute(array($DateAchat, $DateFinGar, $FactureNewName,
$id_batterie));}
      catch (PDOException $e) {
            echo $e->getMessage();}
```

La dernière fonction présente sur la page **DetailsSystemes.php** est **ArchiverSysteme.php**
Cette fonction est la même que pour **ArchiverBatt.php** mais elle modifie la valeur de *Archive_Systeme* à la place de *Archive_Batterie*.

Améliorations possibles

Plusieurs points peuvent être améliorés afin de permettre une meilleure utilisation du logiciel.

Une fonction de modification des données sur la page **DetailsSystemes.php** dans le cas où une erreur aurait été effectuée dans la saisie des informations lors de l'ajout d'un système. Un délai de 1 mois pourrait être mis en place pour autoriser la modification des données, au-delà de ce mois, la modification ne serait plus possible.

Une page permettant de consulter tous les systèmes ainsi que toutes les batteries ayant été archivés pourrait être mise en place car il est toujours utile de pouvoir voir les problèmes survenus sur d'anciens systemes dans le cas où ils se produisent sur les nouveaux.

Dans la fonction de recherche, la possibilité de chercher directement les batteries, ce qui affichera les systèmes ainsi que les batiments où la batterie recherchée se trouve, cela permettrai de s'y retrouver plus facilement dans le cas d'un oubli par rapport à où se trouve tel système ou telle batterie.

Dans la fonction de recherche, appliquer un minimum de caractères saisis de 3, ce qui permettra de réduire considérablement le temps d'exécution de la recherche dans le cas où beaucoup de données sont stockées dans la SGBD.

Conclusion

Avant la mise en place de la plateforme de suivi pour l'entretien des systèmes de sécurité, il n'existait aucun outil centralisé pour surveiller et diagnostiquer les problèmes rencontrés. Lorsqu'une anomalie se produisait, une entreprise spécialisée devait être appelée afin de détecter la cause du déclenchement de l'alarme. Cela pouvait être dû à un dysfonctionnement du système, à une défaillance de la batterie ou à une intrusion réelle.

Dans le cas où la batterie était identifiée comme la source du problème, il y avait un délai de commande si celle-ci n'était pas en stock, ce qui entraînait une période sans protection. En conséquence, les coûts d'intervention, de diagnostic et de commande de la batterie s'élevaient généralement entre 200 et 300 euros.

Cependant, grâce à notre logiciel de suivi, les diagnostics seront désormais effectués plus rapidement, ce qui permettra d'anticiper plus efficacement le stockage des batteries et de limiter les coûts liés à l'intervention d'une entreprise externe. En cas de défaillance de la batterie, la réparation pourra être effectuée directement par les agents de la mairie. Ainsi, le coût total de l'opération se situera entre 30 et 60 euros.

La mise en place de ce logiciel offre donc plusieurs avantages significatifs, tels qu'une résolution plus rapide des problèmes, une meilleure anticipation des stocks et une réduction des coûts globaux liés à l'entretien des systèmes de sécurité. Ces améliorations contribuent à renforcer l'efficacité de nos opérations de maintenance.