

# 151-Documentation de projet.

## Rapport de projet

Version 1 du 25.02.2024 au 06.10.2021

Santana Leandro



Module du 01 09 2023 au 06.10.2021

# Table des matières

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>2</b>	<b>Analyse .....</b>	<b>4</b>
2.1	Présentation du projet .....	4
2.2	Usecase .....	4
2.3	Maquette .....	5
2.3.1	Visiteur .....	5
2.3.1.1	S'inscrire.....	6
2.3.1.2	Se connecter .....	6
2.3.2	Utilisateur.....	7
2.3.2.1	Page de garde.....	7
2.3.2.2	Ajouter un monument .....	7
2.4	Diagramme d'activité .....	8
2.5	Diagramme de séquence système.....	9
2.6	Schéma ER .....	12
<b>3</b>	<b>Conception .....</b>	<b>13</b>
3.1	Diagramme de classe.....	13
3.1.1	Client .....	13
3.1.2	Serveur.....	13
3.2	Schéma relationnel .....	14
3.3	Diagramme séquence interactions.....	14
3.4	Conception des tests .....	15
<b>4</b>	<b>17</b>	

# **1 Introduction**

Lors de ce module nous avons comme travail à fournir qui se représente sous forme de site web. En ce qui me concerne j'ai décidé de baser mon site sur un « recueil » des bâtiments recensés à l'UNESCO

## 2 Analyse

### 2.1 Présentation du projet

Mon site est un rassemblement des bâtiments recensés à l'UNESCO. Chaque monument est constitué de ces éléments. Un pays dans lequel il se situe, une image du monument, une description, et une localisation et son nom.

Il y aura deux types d'utilisateurs sur le site en premier lieu les simples visiteurs qui pourront uniquement regarder le contenu du site. Le contenu du site se compose d'une liste de monuments avec leurs caractéristiques précédemment citées. Le second type d'utilisateur est l'utilisateur authentifié qui pourra quant à lui gérer les monuments ce qui veut dire qu'il pourra ajouter supprimer modifier les monuments et gérer les pays ce qui veut dire les ajouter les supprimer et les modifier. Si le temps me le permet j'aimerais ajouter une carte google maps qui permettra de localiser la position des lieux sur la carte du monde

### 2.2 Usecase

Pour chacun des acteurs présents ici ils auront des rôles et des activités possibles différentes

Rôle	Activités
Visiteur	Se connecter à son compte
	Créer son compte
	Visualiser le site sans pouvoir modifier un quelconque élément
Utilisateur	Il peut se déconnecter de son compte ce que le client ne peut pas faire
	Il peut modifier les monument
	Il peut modifier les pays

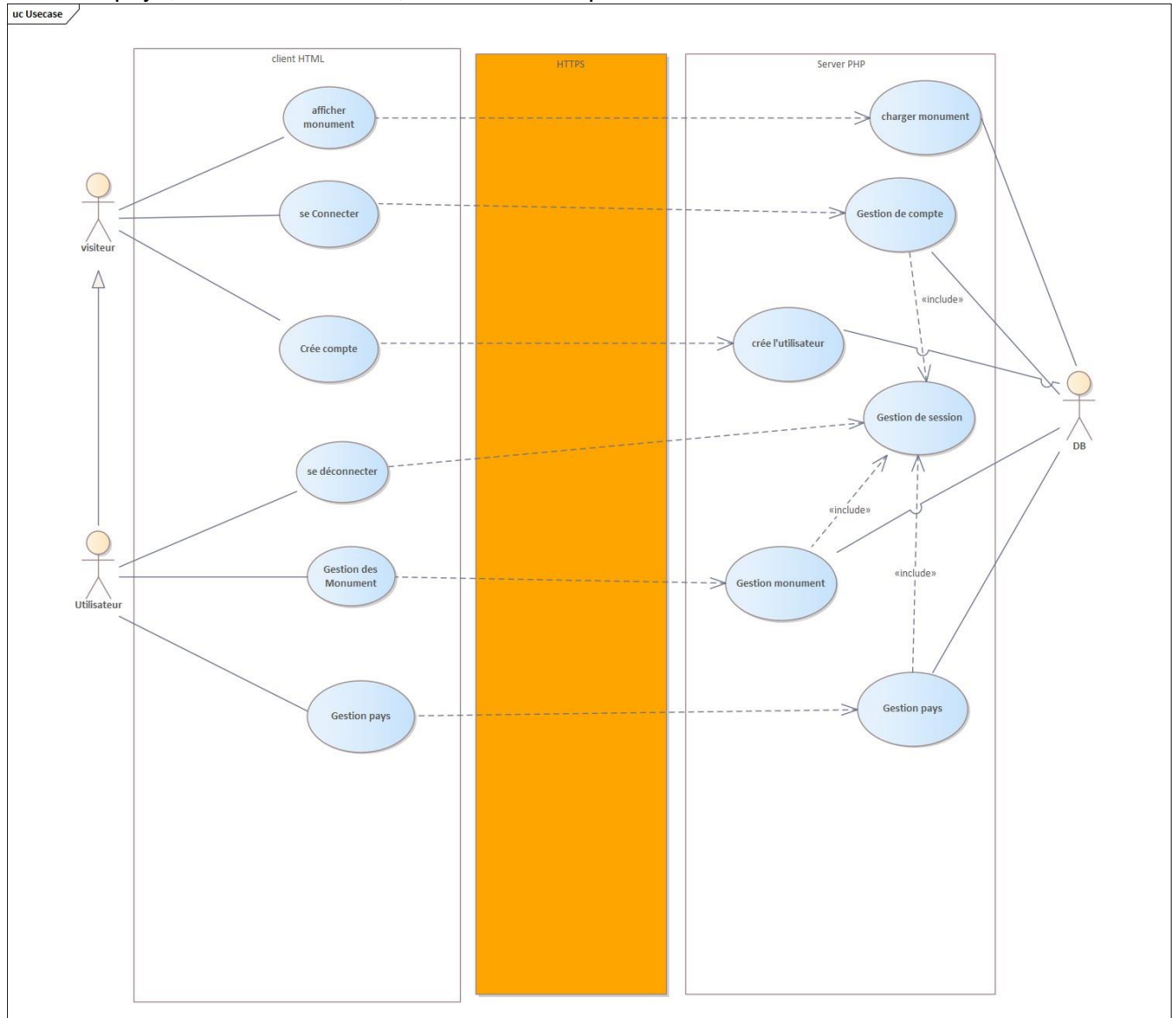
Les utilisateurs peuvent faire ce que les visiteurs font. De même pour les administrateurs qui ont tous ce que les rôles en dessous peuvent faire :

Visiter site -> afficher monument

Se connecter -> checkLogin

Créer un compte -> creation de compte

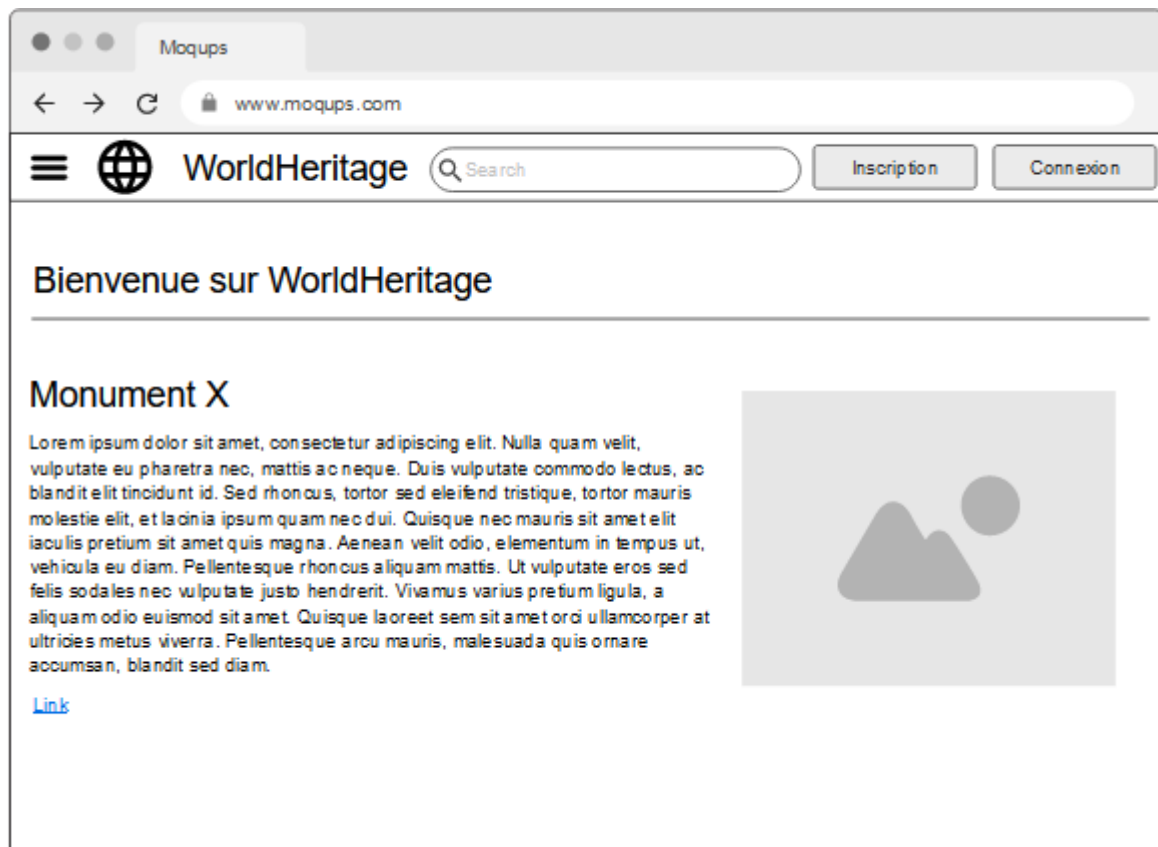
Gestion pays, Gestion monument, Gestion de compte -> DB



## 2.3 Maquette

### 2.3.1 Visiteur

Lorsqu'un visiteur arrive sur le site il a seulement les monuments puis il peut se connecter ou créer un compte pour pouvoir influencer le site



### 2.3.1.1 S'inscrire

Lors que l'utilisateur veut se connecter pour la première fois on lui propose de s'inscrire et de ce fait créer son compte il doit donner un nom d'utilisateur et un mot de passe. Si il possède déjà un compte il peut toujours s'y connecter

### Créer un compte

Nom d'utilisateur	<input type="text" value="Placeholder"/>	Mot de passe	<input type="text" value="Placeholder"/>
Adresse Email (optionel )	<input type="text" value="Placeholder"/>	Vérifier le mot de passe	<input type="text" value="Placeholder"/>

[Avez vous déjà un compte.](#)

### 2.3.1.2 Se connecter

Pour se connecter il faut juste entrer son nom d'utilisateur et son mot de passe

### Connexion

Label

Placeholder

Label


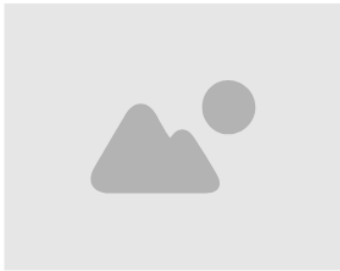
Placeholder

Se connecter

## 2.3.2 Utilisateur

### 2.3.2.1 Page de garde

Lors que l'utilisateur s'est connecté il peut maintenant ajouter un monument et avoir accès a une carte du monde. Il peut toujours se déconnecter

	<div><div>≡</div><div> WorldHeritage</div><div><input type="text" value="Search"/></div></div>
Page de base ▶	<h2>Bienvenue sur WorldHeritage</h2>
Ajouter monument	<div><h3>Monument X</h3><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam. Pellentesque rhoncus aliquam mattis. Ut vulputate eros sed felis sodales nec vulputate justo hendrerit. Vivamus varius pretium ligula, a aliquam odio euismod sit amet. Quisque laoreet sem sit amet orci ullamcorper at ultricies metus viverra. Pellentesque arcu mauris, malesuada quis ornare accumsan, blandit sed diam.</p><a href="#">Link</a></div> <div></div>
Carte du monde	
déconencter	

### 2.3.2.2 Ajouter un monument

Lors que l'on ajoute un monument il faut entrer les critères nécessaire que l'on voit dessous

Nom

Placeholder

Pays

Select

Coordonné X

Placeholder

Coordonné Y

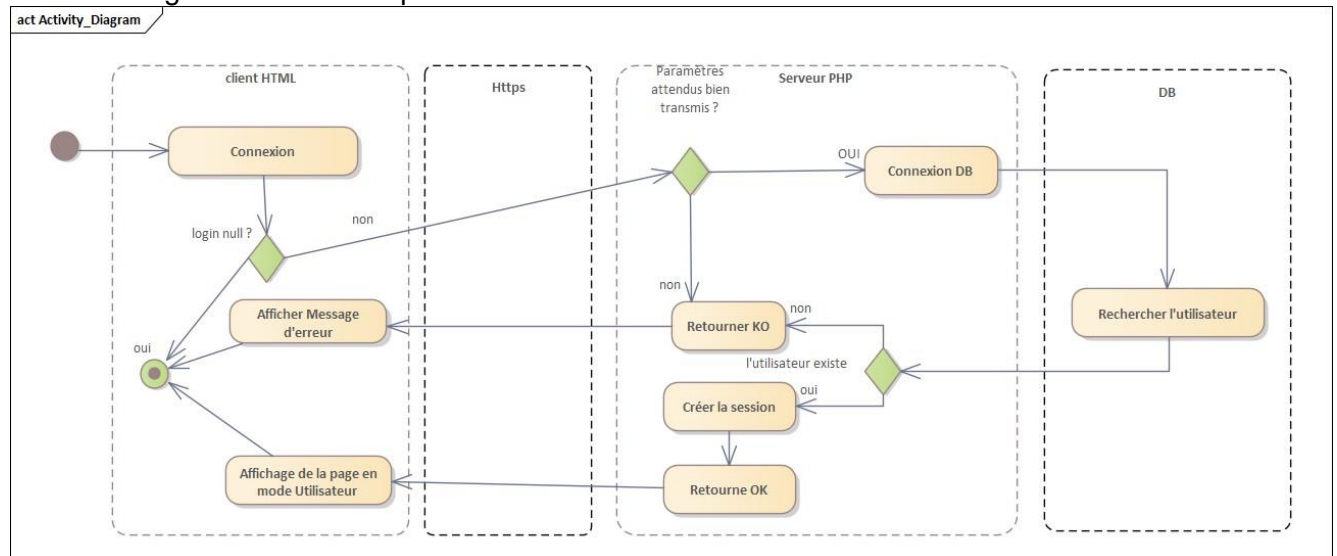
Placeholder

Image à ajouter

Créer

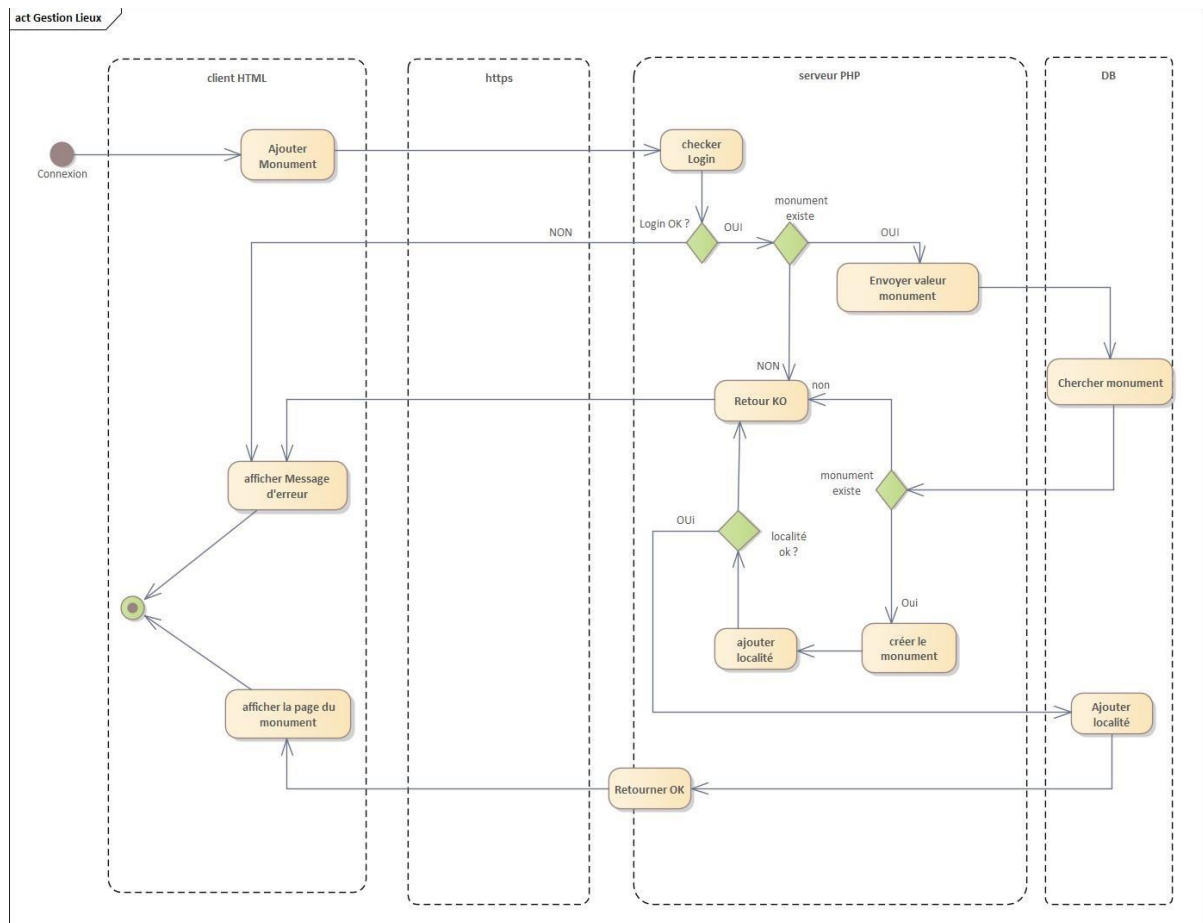
## 2.4 Diagramme d'activité

Voici le diagramme d'activité pour la connexion



Voici celui pour l'ajout d'un monument

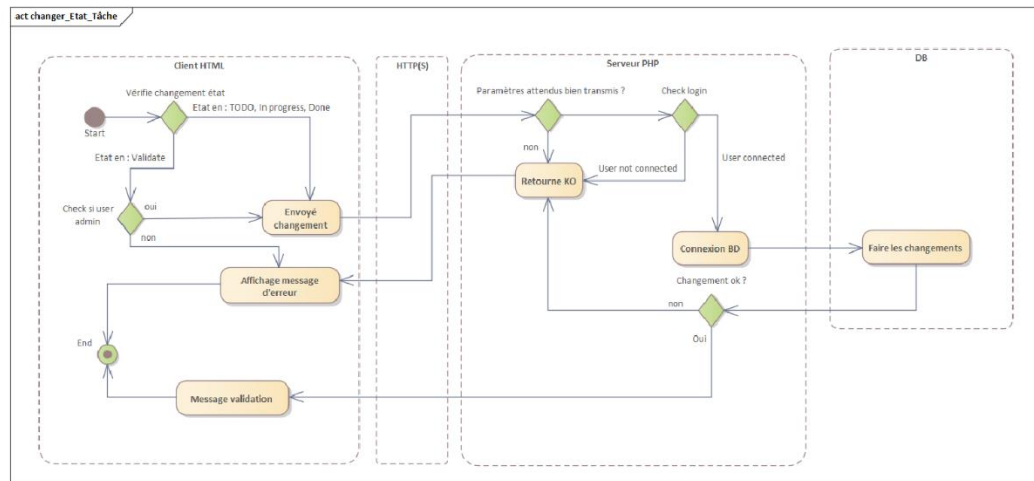




## 2.5 Diagramme de séquence système

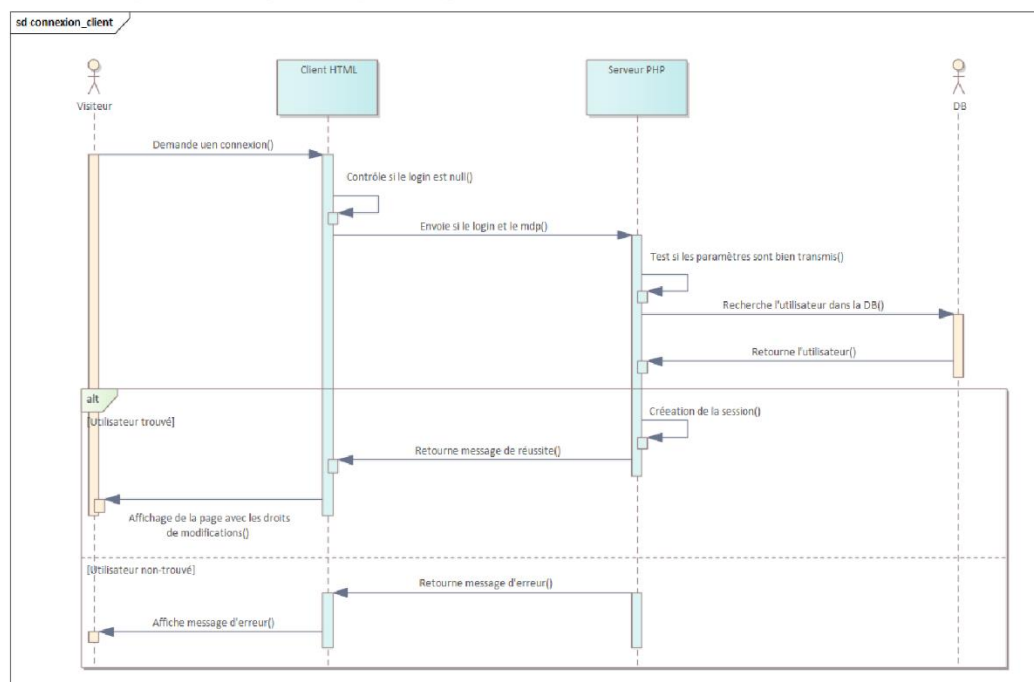
Voici le diagramme de séquence système pour la connexion

151-PaccaudS



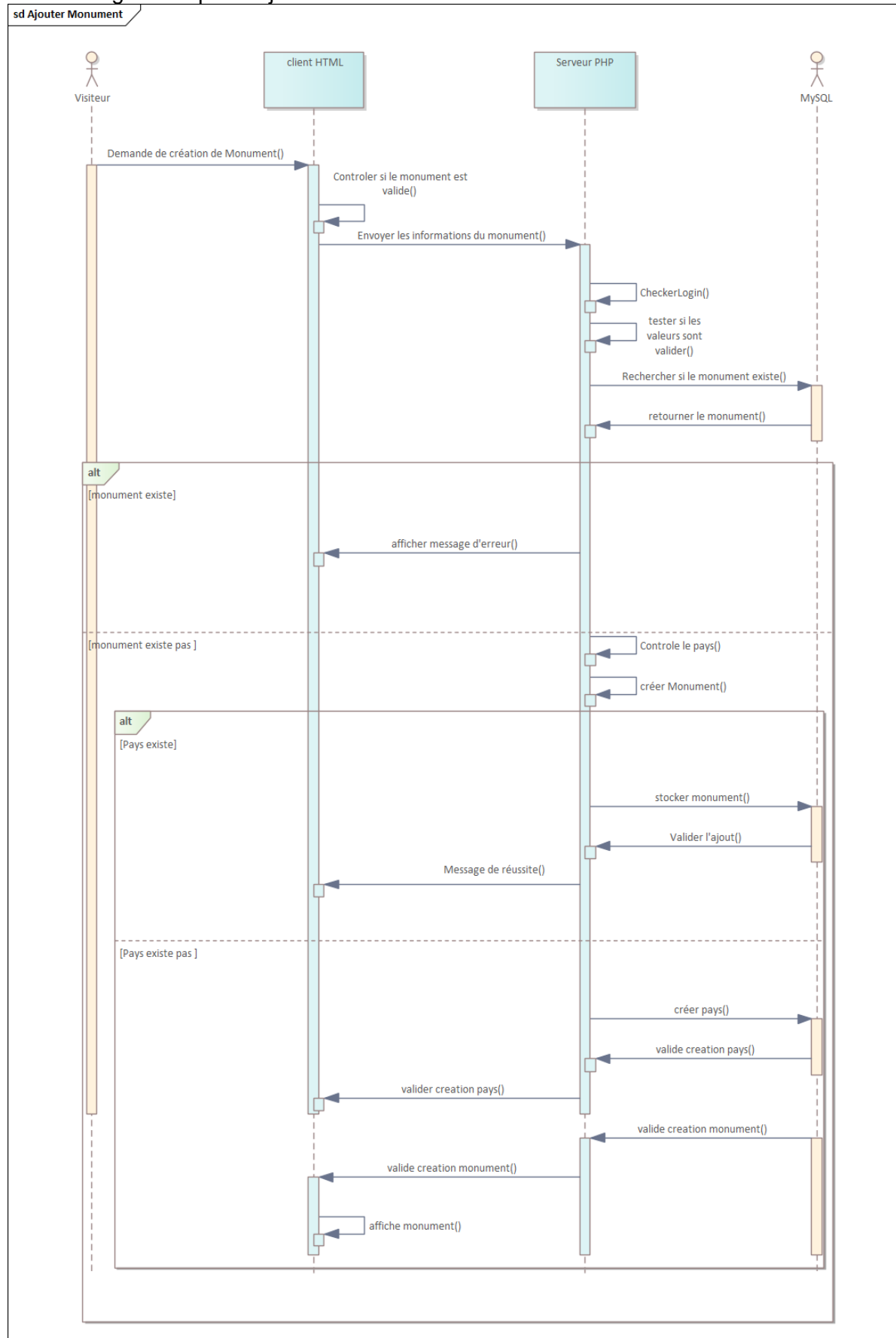
## Diagramme de séquence système

Voici le diagramme séquence système pour la connexion à un utilisateur.



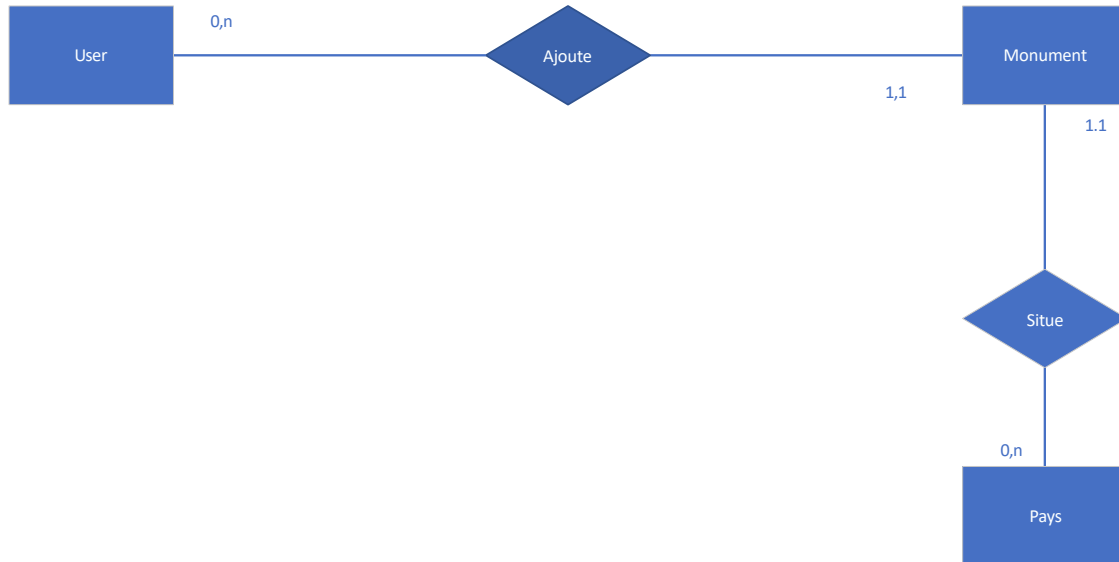
Voici le diagramme pour récupérer les projets d'un utilisateur.

Voici le diagramme pour l'ajout de monument



## 2.6 Schéma ER

Voici le schéma entité relation pour la basse de donnée

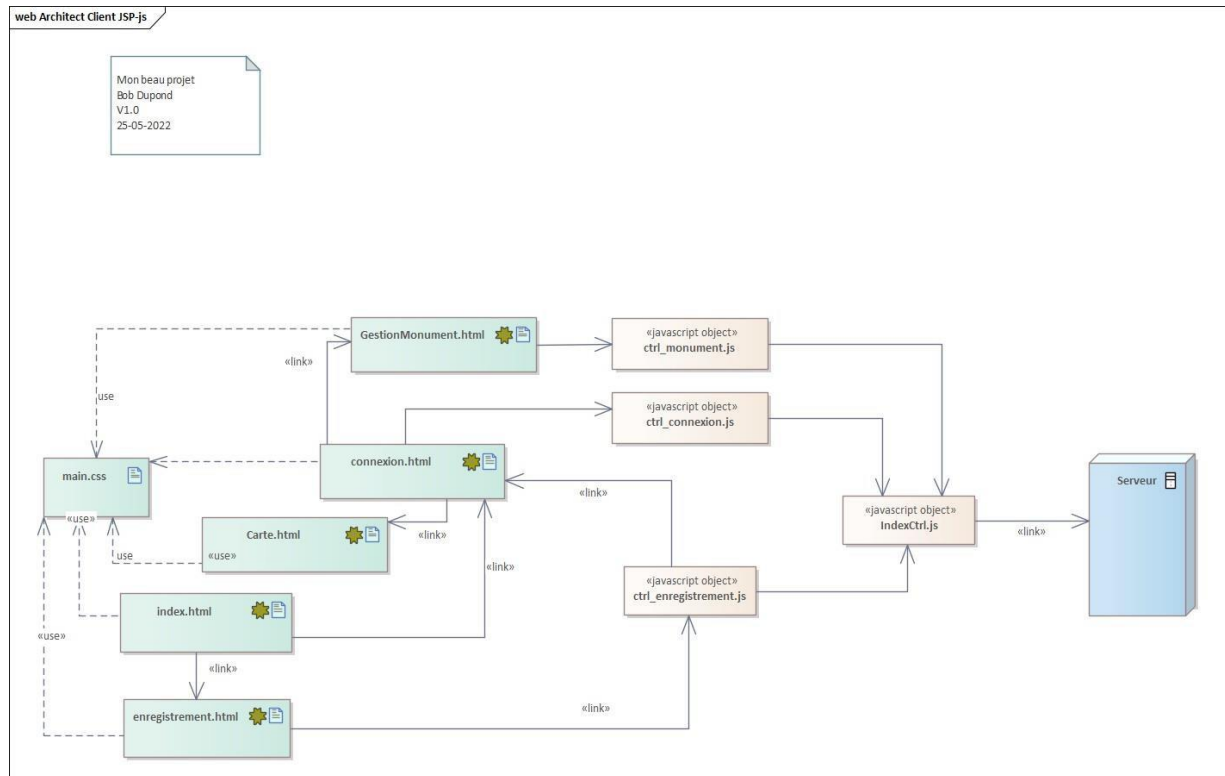


## 3 Conception

### 3.1 Diagramme de classe

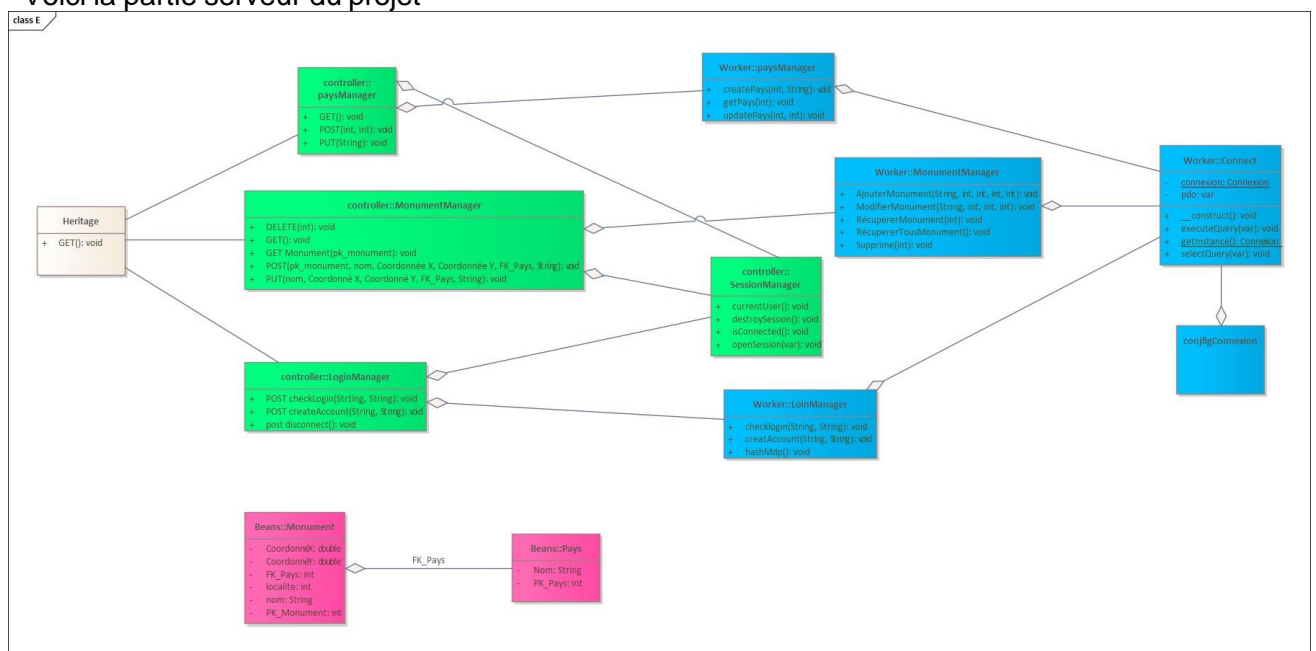
#### 3.1.1 Client

Voici la structure espérée de la partie cliente de l'application



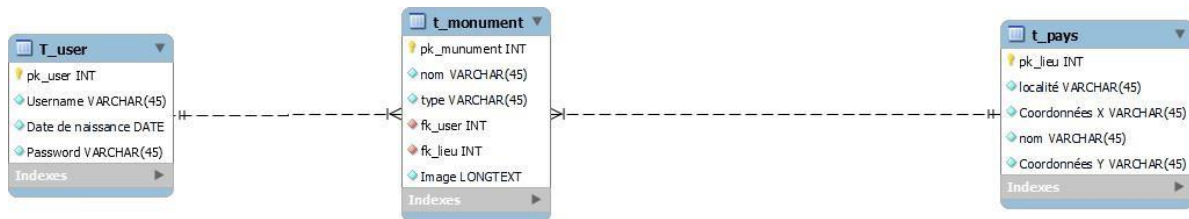
#### 3.1.2 Serveur

Voici la partie serveur du projet



## 3.2 Schéma relationnel

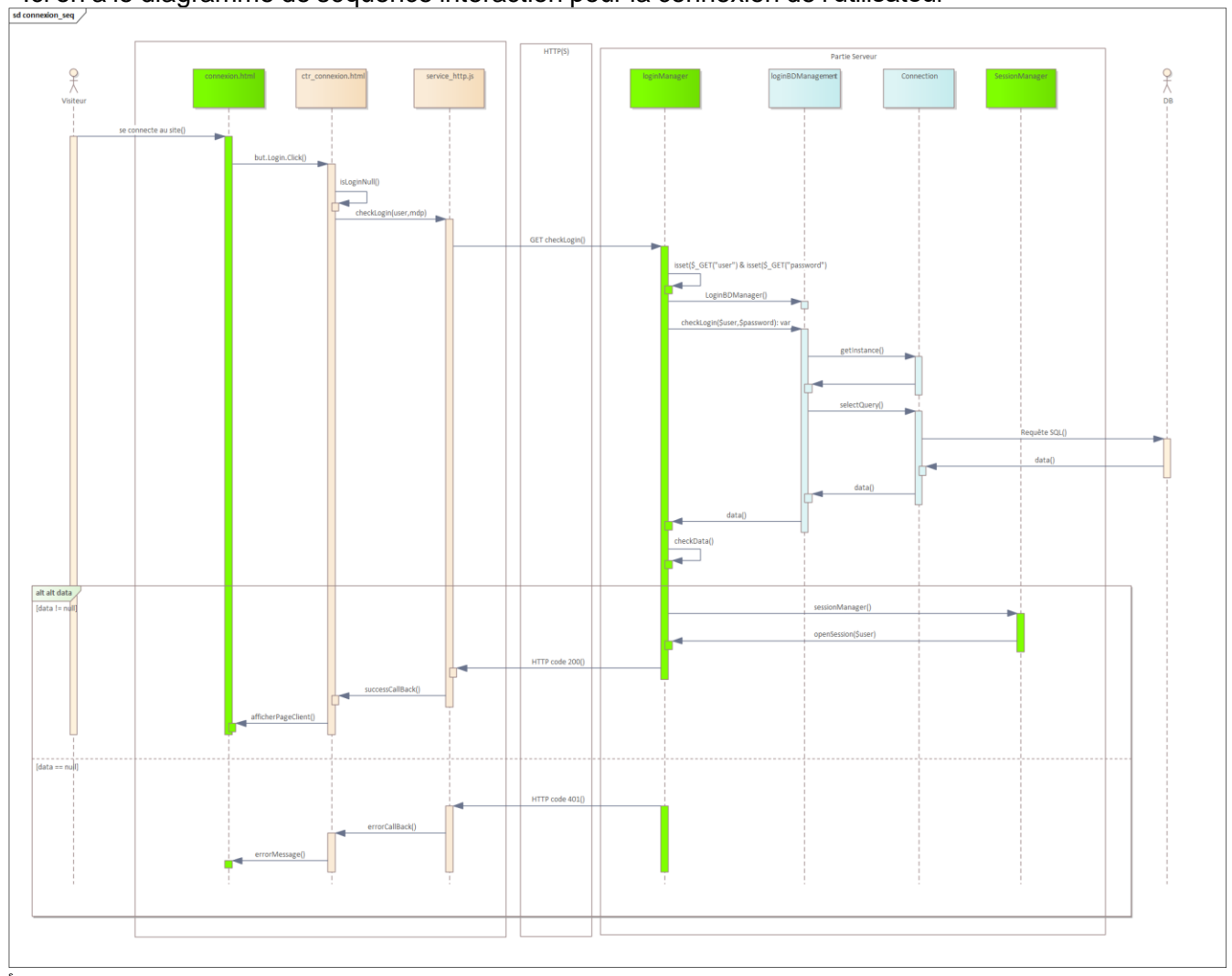
Ici nous avons le schéma relationnel basé sur le schéma ER



## 3.3 Diagramme séquence interactions

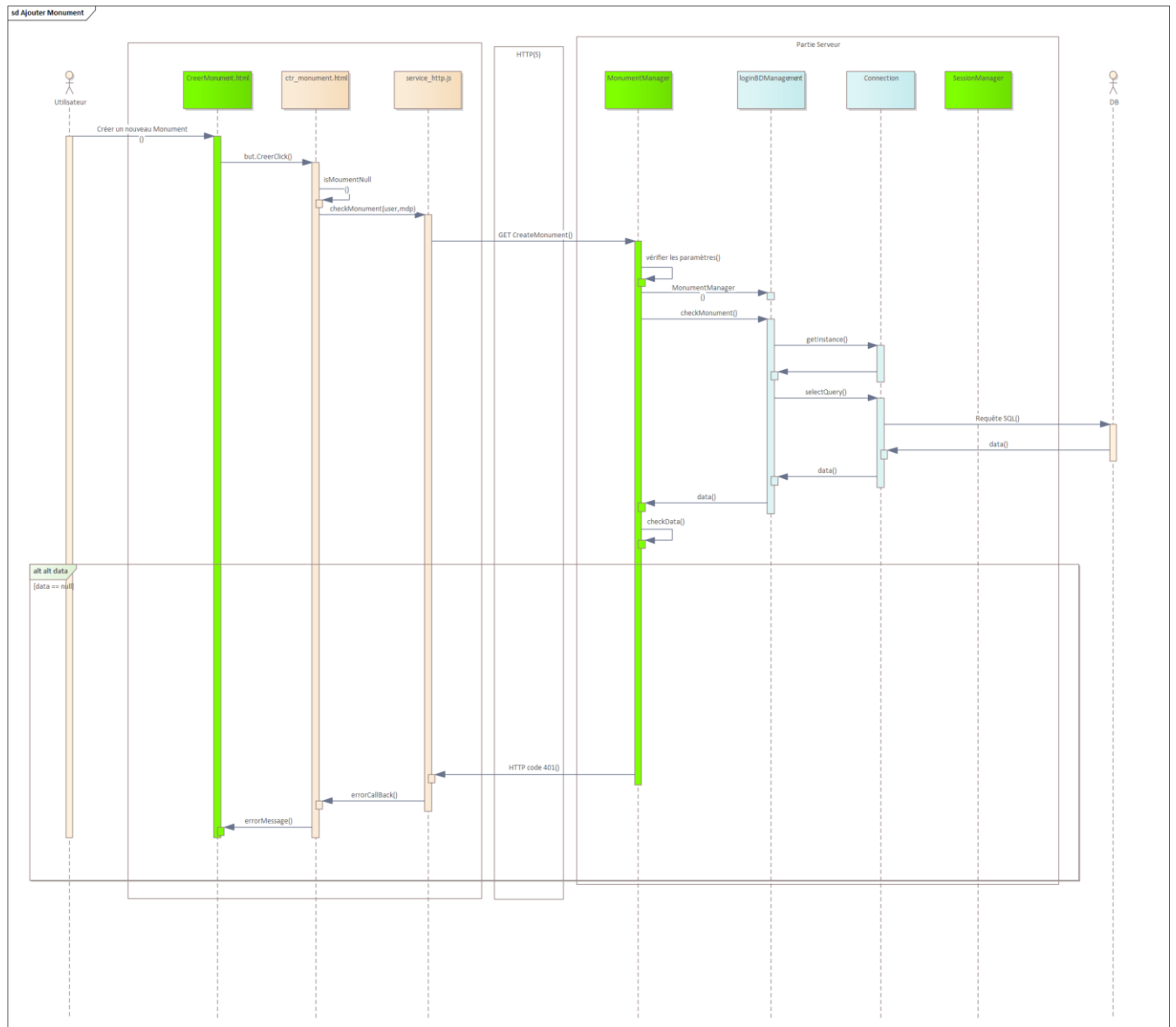
### 3.3.1 Se connecter

Ici on a le diagramme de séquence interaction pour la connexion de l'utilisateur



### 3.3.2 Ajouter Monument

Ici on a le diagramme d'activités pour ajouter les monuments



### 3.4 Conception des tests

Fonctionnement	Page ?	Résultat attendu	Résultat obtenu	Test validé ?
Visiter le site sans se connecter	Index.html	Le visiteur peut visionner les informations du site		
Crée un nouvel utilisateur	enregistrem.html	L'utilisateur est bien créé dans la base de données		
Crée un utilisateur qui est déjà existant	enregistrem.html	Un message d'erreur apparaîtra		
Se connecter à un compte existant avec les bonnes informations	connexion.html	La page d'accueil sera ouverte mais avec les options pour gérer le site		
Se connecter à un compte existant avec des informations fausses	Connexion.html	Un message d'erreur devrait apparaitre.		

<b>Ajouter un monument à la liste</b>	Monuiment.html	Le monument sera ajouter et afficher		
<b>Ajout d'un pays</b>	Pays.html	Un pays sera ajouter a la liste de tous les pays		
<b>Modifier le monument</b>	Monument.html	Les valeurs que l'on voudras changé du monument seront changé et l'affichage fera de même		
<b>Modifier un pays</b>	Pays.html	Le paramètre du monument qui sera changer sera afficher correctement		
<b>Déconnection d'un utilisateur.</b>	Index.html	L'utilisateur doit pouvoir se déconnecter.		



# 1. Implémentation

## Descente de code

Voici l'affichage pour ajouter le monument

Pour ce faire nous avons toute une infrastructure que je vais détailler ci-dessous :

L'ordre pour l'explication sera du serveur à l'interface

La méthode se nomme `ajouterMonument` elle permettra d'ajouter le monument à la base de données avec tous les paramètres qui sont présents dans la base de données

```
public function ajouterMonument($nom, $localite, $coordonnesX, $coordonnesY,
$username_user)
{
    $test = "";
    try {
```

en premier lieu on a la nécessité de faire une transaction car nous allons faire des modifications et pour éviter que plusieurs personnes produisent un conflit nous utilisons ce processus

```
$test = connexion::getInstance()->startTransaction();
```

Il faut en premier lieu déterminer les `fk_pays` et `fk_user` pour ce faire on va déterminer la pk dans sa table respective et on va l'assigner à une variable qui sera la valeur de la pk

```
$param = array(':nom' => $localite);
$query_pays = connexion::getInstance()->selectSingleQuery("SELECT pk_pays FROM
t_pays WHERE nom = :nom", $param);

$fk_Pays = $query_pays['pk_pays'];
$Session_username = $username_user;
$query_user = connexion::getInstance()->selectSingleQuery("SELECT pk_user FROM
t_user WHERE username = :username_user", array(":username_user" => $Session_username));
$fk_user = $query_user['pk_user'];
```

Après avoir récupéré les 2 fk on va créer un tableau qui recueillera tous les paramètres à ajouter

```
$array = array(
    ":nom" => $nom,
```

```

        ":localite" => $localite,
        ":fk_user" => $fk_user,
        ":fk_pays" => $fk_Pays,
        ":coordonnesY" => $coordonnesY,
        ":coordonnesX" => $coordonnesX
    );

```

Puis suite a la détermination de tous les paramètres on va exécuter la requête qui va nous permettre d'ajouter le monument à la base de données. Dans la partie values () on a toutes les valeurs déterminées au préalable

```

$query = "INSERT INTO `monumentheritage`.`t_monument`
        (`nom`, `localite`, `fk_user`, `fk_pays`, `coordonnesY`,
`coordonnesX`)
        VALUES (:nom, :localite, :fk_user, :fk_pays, :coordonnesY,
:coordonnesX)";
$connexion::getInstance()->executeQuery($query, $array);

```

maintenant selon le résultat on aura plusieurs actions possibles soit retourner la validité avec un code 200 soit une erreur avec un code 500 puis on finis notre transaction et soit on l'annule si il y a une erreur sinon on l'envoie comme prévue

```

if ($query) {
    http_response_code(200);
    $result = json_encode(array("isOk" => true, "message" => "ajout monument
OK"));
    $test = connexion::getInstance()->commitTransaction();
} else {
    http_response_code(500);
    $result = json_encode(array("isOk" => false, "message" => "ajout monument
NOK"));
    $test = connexion::getInstance()->rollbackTransaction();
}

```

Il ne faut pas oublier les potentiels erreur avec la requête et on décide quel message on va afficher

```

    } catch (PDOException $e) {
        connexion::getInstance()->rollbackTransaction();
        return json_encode(array("isOk" => false, "message" => "Erreur lors de l'ajout
du monument : " . $e->getMessage()));
    }
    return $result;
}

```

### Ctrl ajouter

Dans la partie ctrl nous ne ferons pas grand-chose a part transmettre vers le fichier « monument.php »

```

public function AjouterMonumentJSON($nom, $localite, $coordonneeX, $coordonneeY,
$username)
{
    $monument = $this->manager->ajouterMonument($nom, $localite, $coordonneeX,
$coordonneeY, $username);
    return $monument;
}

```

### Monument.php

Maintenant que nous sommes au sommet de la partie php nous allons devoir le transmettre à l'avant pour ce faire on vérifie que tous les paramètres sont fournis et ensuite on appelle la méthode pour ajouter le

monument avec les paramètres en \$\_POST ce qui voudra dire que depuis le client on va prendre le contenu de la requête httpService comme paramètres qu'on utilisera pour ajouter le monument donc si on ajoute un monument qui s'appelle test il faudra le mettre dans les datas de httpService

```
case 'POST':
    if (isset($_POST['nom']) && isset($_POST['localite']) &&
isset($_POST['coordonneeX']) && isset($_POST['coordonneeY']) && isset($_POST['username'])) {
        $login = new MonumentManager();
        echo $login->AjouterMonumentJSON($_POST['nom'], $_POST['localite'],
$_POST['coordonneeX'], $_POST['coordonneeY'], $_POST["username"]);
    } else {
        echo 'un des paramètres est manquant';
    }
    break;
```

### servicesHttp.js

Maintenant nous allons sur l'avant pour ce faire il faut créer la méthode qu'on a utilisé au part avant. Lors que l'on veut appeler la méthode du php on va utiliser celle-ci et on doit spécifier le type qui est ici un post donc comme expliqué au part avant nous allons devoir spécifier les paramètres dans la data. Il faut toujours faire un successCallback et un errorCallback c'est les deux possibilités de retour lorsque la méthode va être appelée.

```
function createMonument(name, pays, CoordonneeX, CoordonneeY, nom, successCallback, errorCallback) {

    $.ajax({
        type: "POST",
        dataType: "json",
        url: BASE_URL + "Monument.php",
        data: {
            "nom": name,
            "localite": pays,
            "coordonneeX": CoordonneeX,
            "coordonneeY": CoordonneeY,
            "username": nom,
        },
        xhrFields: {
            withCredentials: true
        },
        success: successCallback,
        error: errorCallback
    });
}
```

### Ctrl\_add.js

On fait quelques prérequis pour pouvoir faire communiquer l'utilisateur avec le service http. Donc on doit d'abord se lier à servicehttp et déterminer les variables que l'on utilisera.

```
$.getScript("javascripts/services/servicesHttp.js", function () {
    console.log("servicesHttp.js chargé !");
});
var addnameMonument;
var addnamelocalite;
var addCoordonneeX;
```

```
var addCoordonneeY;  
var user;
```

Quand la page est complètement chargée on va exécuter une action spécifique lors que l'on clique sur un bouton dans notre cas c'est « btnCreer »

```
$(document).ready(function () {  
    $("#btnCreer").click(function (event) {  
        event.preventDefault(); // pour afficher les echos / permission  
        console.log("Button créer presser");
```

on va déterminer quels champs présents dans l'interfaces seront utiliser comme base pour les paramètres utiliser par la suite dans le fonctionnement global

```
        addnameMonument = $("#name").val();  
        console.log("name = ", addnameMonument);  
        addnameLocalite = $("#pays").val();  
        console.log("pays = ", addnameLocalite);  
  
        addCoordonneeX = $("#CoordonneX").val();  
        console.log("CoordonneX = ", addCoordonneeX);  
        addCoordonneeY = $("#CoordonneY").val();  
        console.log("CoordonneY = ", addCoordonneeY);  
        user = sessionStorage.getItem("username");  
        console.log("utilisateur = ", user);
```

il faut maintenant appeler une méthode qui appellera la méthode du servicehttp les paramètres utiliser sont ceux récupérer par les champs textes

```
        AjouterMonument(addnameMonument, addnameLocalite, addCoordonneeX, addCoordonneeY,  
user, );  
    });  
})
```

Maintenant on appelle la méthode de servicehttp définis au part avant et on configure les success call back et les error call back en premier lieu on va commencer avec le success

```
function AjouterMonument(nom, pays, CoordonneX, CoordonneY, username, ) {  
    console.log("création du monument : " + nom + " " + pays + " " + CoordonneX + " " +  
CoordonneY + username);  
  
    // Appel de la fonction AJAX pour se connecter  
    createMonument(nom, pays, CoordonneX, CoordonneY, username, function (response) { //  
Callback de succès  
        console.log(response);
```

on récupérer ce qui est retourné par la requête d'ajout. Il faut surtout récupérer le « isOk » comme ça on va pouvoir passer dans le success. Ici on prend aussi le message pour pouvoir voir l'état de façon plus clair

```
        // Récupération des propriétés de la réponse JSON  
        var success = response.IsOk;  
        var message = response.message;  
        console.log(success);
```

maintenant que le success est passer on va charger le nouveau fichier que l'on veut ici c'est le « indexLogin.html » qui est une copie du index.html mais qui est uniquement accessible quand on est connecté

```
        if (success) {  
            alert("Monument connecté avec succès : ");  
  
            window.location.href = 'indexLogin.html';
```

dans le pire des cas on envoie un message d'erreur sous forme de pop up pour informer l'utilisateur que la personne n'a pas pu ajouter le monument

```
} else {  
    alert("Vous n'avez pas pu vous connecter : " + message);  
}  
, function (error) {  
    alert("Erreur lors de la création du monument");  
});  
}
```

### CreateMonument.html

```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Connexion et Création de Compte</title>  
    <link rel="stylesheet" href="stylesheets/connexion.css">  
    <script type="text/javascript" src="javascripts/helpers/jquery-  
1.10.2.min.js"></script>  
    <script type="text/javascript" src="javascripts/controllers/ctrl_add.js"></script>  
  
</head>
```

Ici on aura les styles nécessaires à l'affichage mais surtout les id qui seront les paramètres utilisés tout au long de la requête

```
<body class="containers">  
  
    <div class="login-container">  
        <form class="login-form">  
            <p class="heading">Ajouter le monument</p>  
            <div class="input-group">  
                <input required="" placeholder="nom" id="name" type="text" />  
            </div>  
            <div class="input-group">  
                <input required="" placeholder="pays" id="pays" type="text" />  
            </div>  
            <div class="input-group">  
                <input required="" placeholder="Coordonnées N°" id="CoordonneX"  
type="text" />  
            </div>  
            <div class="input-group">  
                <input required="" placeholder="Coordonnées E°" id="CoordonneY"  
type="text" />  
            </div>  
  
            <div class="input-group">  
                <input type="submit" value="Créer" id="btnCreer">  
  
            </div>
```

```

    </div>
</body>

</html>

```

## Problèmes rencontrés

Un problème que j'ai eu c'est le fait que je n'arrivais pas à afficher tous les monuments que je récupérer

```

foreach ($test as $row) {
    // Récupérez les valeurs spécifiques de chaque ligne
    $nom = $row['nom'];
    $localite = $row['localite'];
    // Concaténez les valeurs dans la chaîne HTML
    $data .= '<div class="text-holder" id="monumentInfo">';
    $data .= '<p>Nom: ' . $nom . '</p>';
    $data .= '<p>Localite: ' . $localite . '</p>';
    $data .= '</div><br>';
}

```

Du fait que j'ai certains styles sur les divs il fallait faire en sorte que tous les éléments soient récupérés et pas uniquement le premier comme je pensais puis il faut les mettre dans des balises directement. La suite se situe dans le JavaScript on va maintenant voir une autre spécificité on utilisera le `.innerHTML` qui va nous permettre d'ajouter au fichier html directement ici on va récupérer tous les éléments qui possèdent un « text-holder » qu'on a définis au part avant et on va déterminer si oui ou non le texte est court ou pas ce qui va seulement changer l'affichage.

```



monumentInfoElement.innerHTML = monumentsData;




// Ajouter des classes CSS en fonction de la longueur du contenu
var elements = monumentInfoElement.querySelectorAll('.text-holder');
elements.forEach(function (element) {
    // Récupérer la longueur du texte
    var textLength = element.textContent.length;

    // Ajouter des classes CSS en fonction de la longueur du texte
    if (textLength > 50) {
        element.classList.add('long-text');
    } else {
        element.classList.add('short-text');
    }
});
} else {

```

## Tests fonctionnels

Fonctionnement	Page ?	Résultat attendu	Résultat obtenu	Test validé ?
Visiter le site sans se connecter	Index.html	Le visiteur peut visionner les informations du site	Site visitable	
Crée un nouvel utilisateur	enregistre-ment.html	L'utilisateur est bien créé dans la base de données	Les utilisateurs sont tous dans la DB	

<b>Crée un utilisateur qui est déjà existant</b>	enregistrem ent.html	Un message d'erreur apparaîtra	Les utilisateurs sont créables même s'ils existent déjà	
<b>Se connecter à un compte existant avec les bonnes informations</b>	connexion.h tml	La page d'accueil sera ouverte mais avec les options pour gérer le site	L'utilisateur peut bien ajouter ou modifier les monuments	
<b>Se connecter à un compte existant avec des informations fausses</b>	Connexion. html	Un message d'erreur devrait apparaître.	Une erreur dit que les valeurs entrées ne sont pas valides	

## Hébergement

Le site web sera hébergé chez [santanal.emf-informatique.ch](http://santanal.emf-informatique.ch)

Quelques fonctionnalités n'ont pas encore d'applications graphiques

## **2. Synthèse**

### **Présentation réalisation**

Je n'ai pas eu la possibilité de réaliser une présentation devant la classe je fus bien trop en retard mais j'ai pris mon weekend pour rendre un travail des plus correctes

### **Différences entre planning et réalisation**

J'avais omis la journée d'informations de l'armée de ce fait j'ai eu un assez gros retard pour éviter de rendre un projet en lambeau j'ai pris du temps sur mes soirs et sur mon weekend

### **Conclusion**

J'ai beaucoup apprécié le contenu du module malgré les quelques problèmes il fut très conséquent mais pas pour autant trop dur il fallait juste prendre le temps nécessaire et demander a chatgpt les potentiels problèmes





