151 / Intégrer des éléments de base de données dans des applications Web

Rapport personnel

Date de création : 09.02.2024
Version 1 du 09.02.2024

Vallélian Enzo

E — Ecole des Métiers
— M Berufsfachschule
— F Fribourg
Informatique

Module du 09.02.2024
au jj.mm.aaaa

Table des matières

1 Introduction

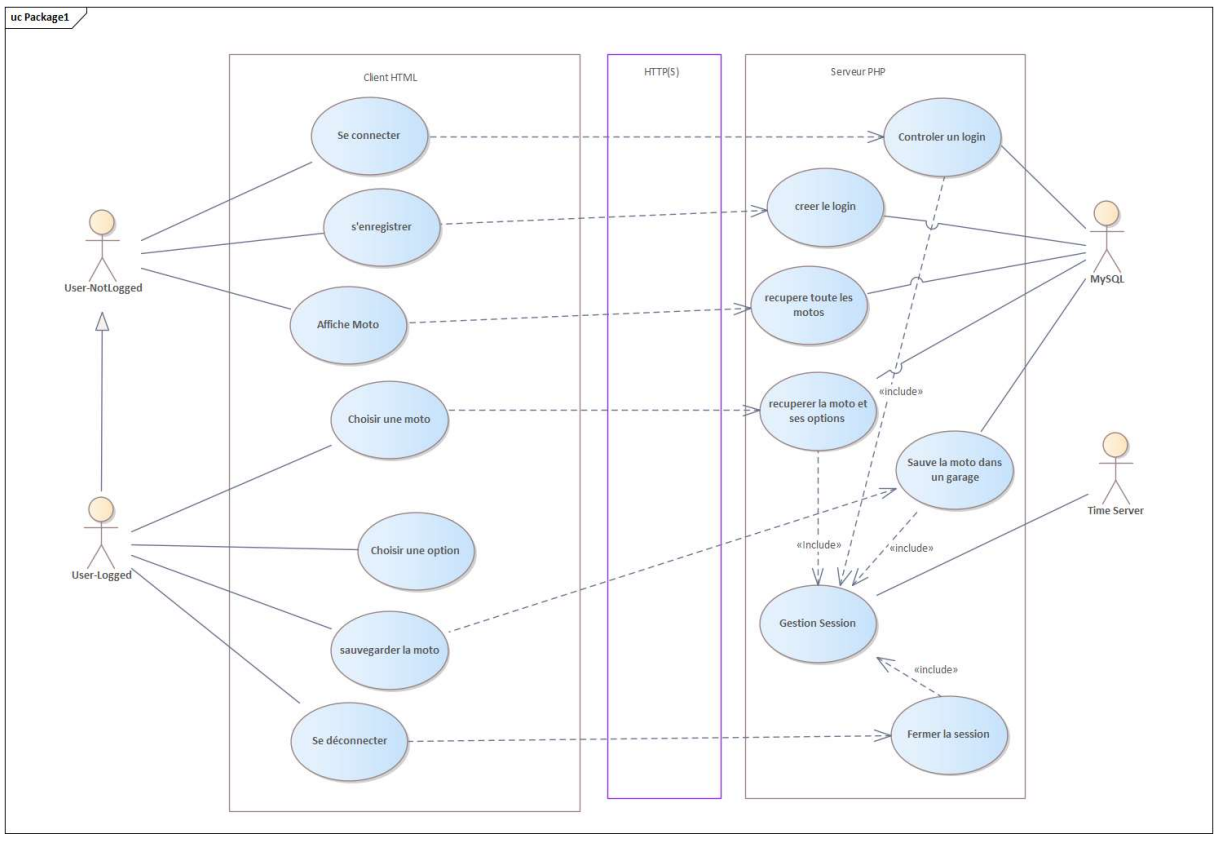
1.1 Analyse

1.1.1 Présentation du projet

Mon projet va être un configurateur de moto. La page d'accueil sera une sorte de showroom ou toutes les motos à configurer puis dès que le visiteur se connectera il pourra choisir des packs d'options. Enfin il pourra la mettre dans son garage qu'il récupèrera à chaque connexion.

Pour mieux comprendre mon projet, quand un utilisateur n'est pas connecté il va simplement voir la « vitrine » des motos. S'il est connecté alors il va pouvoir cliquer sur une moto. Il va être emmener sur la page « moto.html ». Sur cette page il va pouvoir configurer sa moto avec une seul option (paquet d'option) et appuyer sur le bouton (Make It), pour finir la sauvegarde et le stock dans un garage personnel qui se trouve sur la gauche de l'interface. Quand l'utilisateur va revenir sur le site son garage sera sauvegarde. Pour un utilisateur qui n'a pas de compte il est possible de s'enregistrer en allant sur « Sign in » puis « Register » et ensuite il suffira de mettre son login.

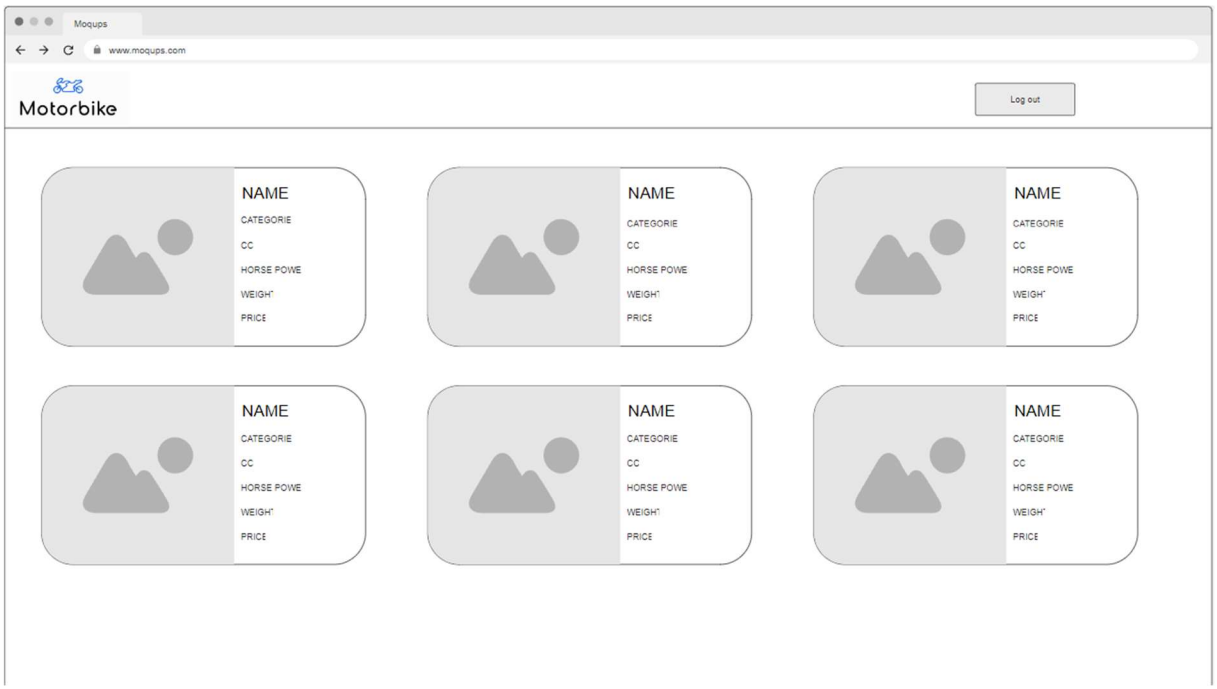
1.1.2 Uses Case



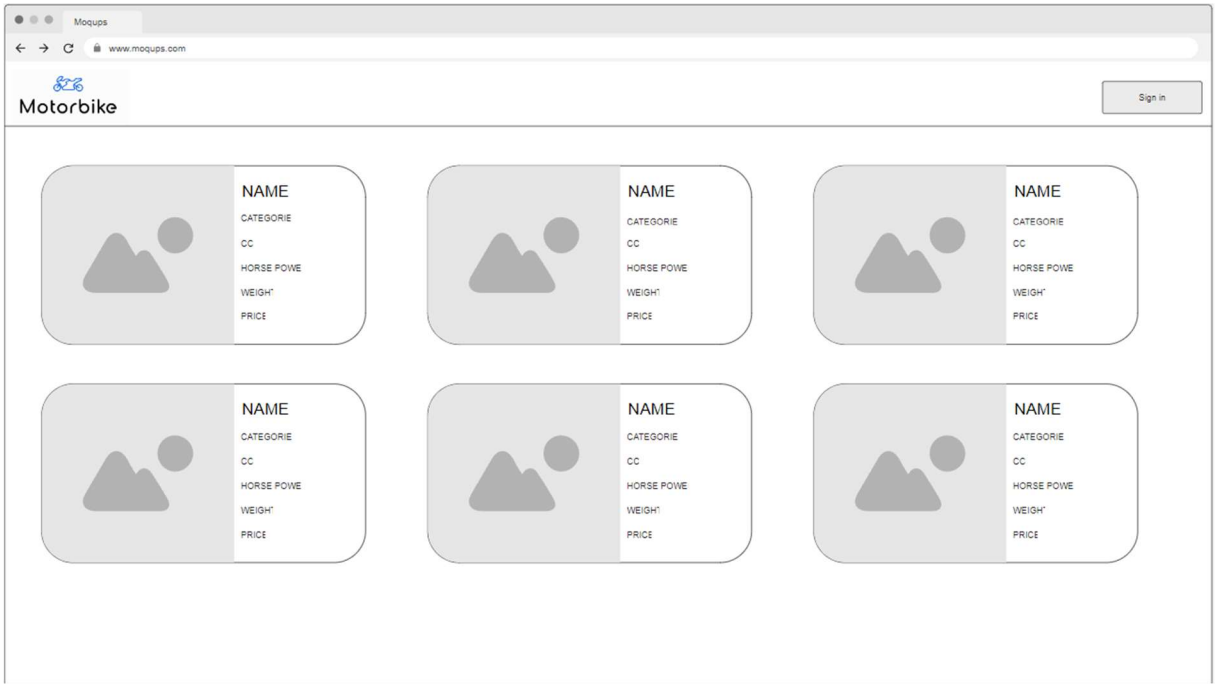
1.1.3 Maquettes

Voici mes maquettes :

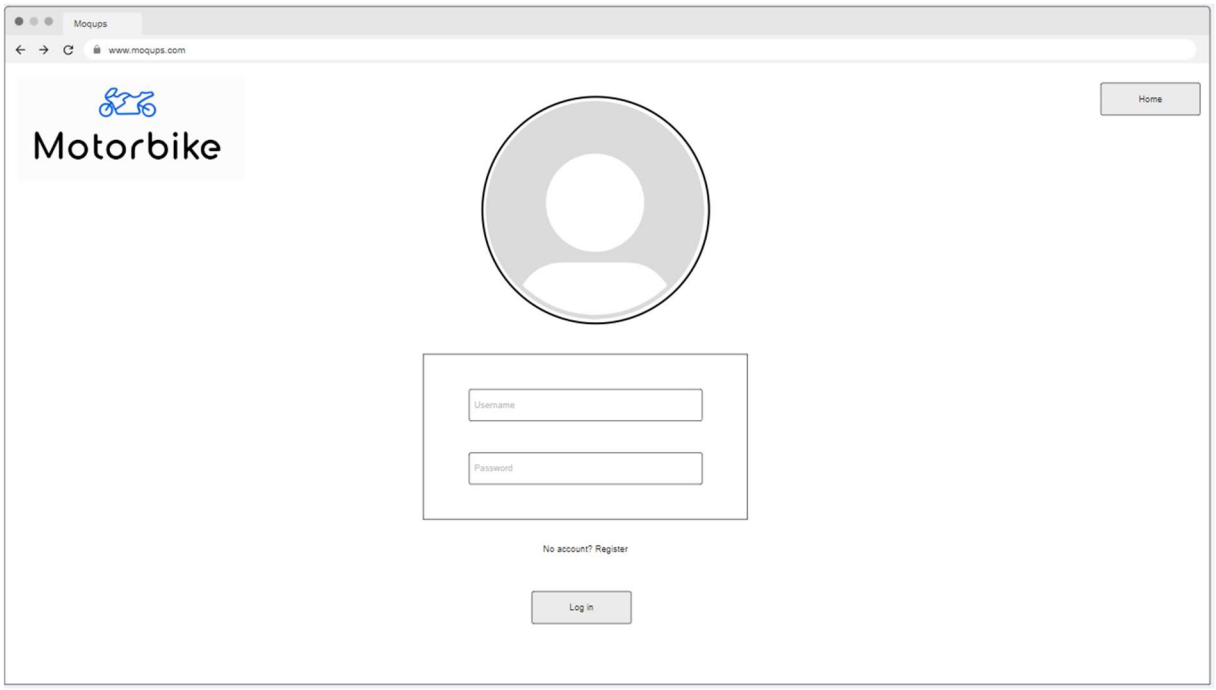
La page « Home si connecté » :



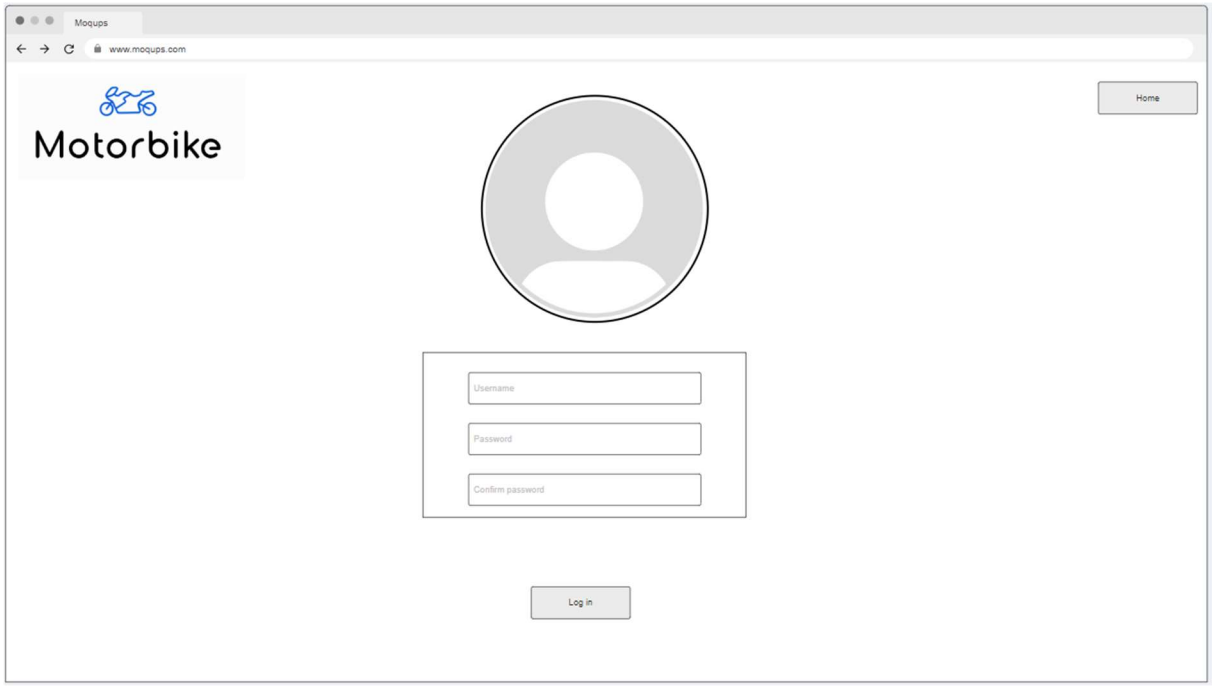
La page « Home si pas connecté » :



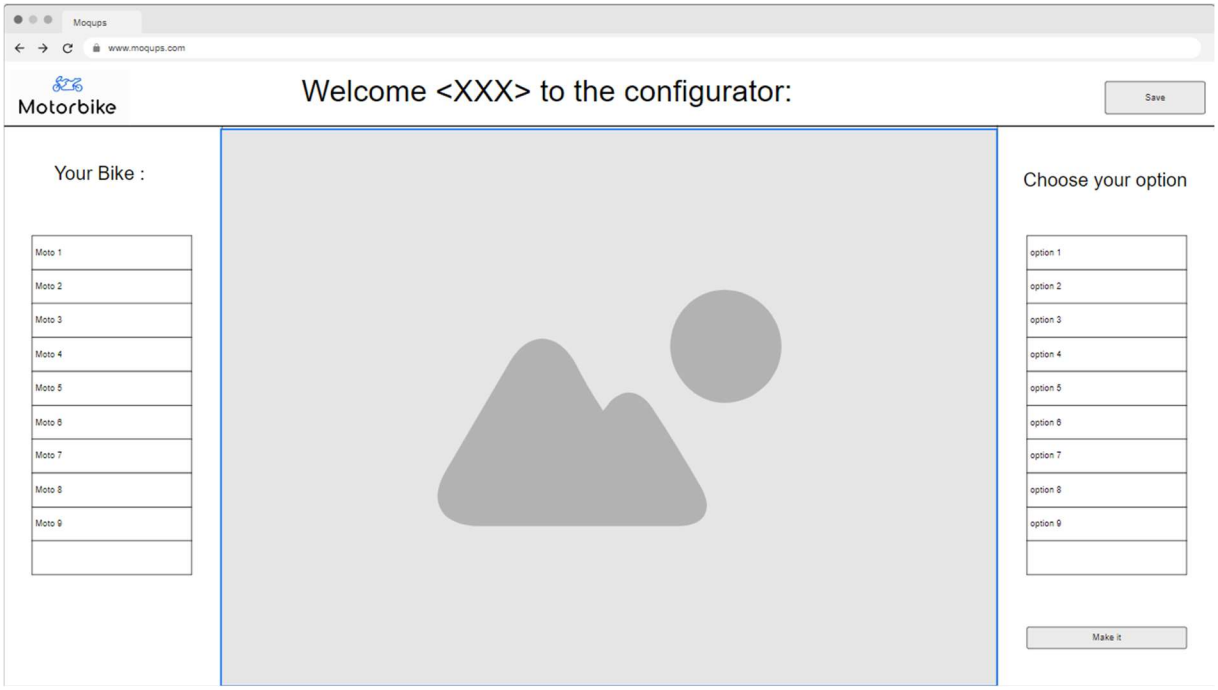
La page « Login » :



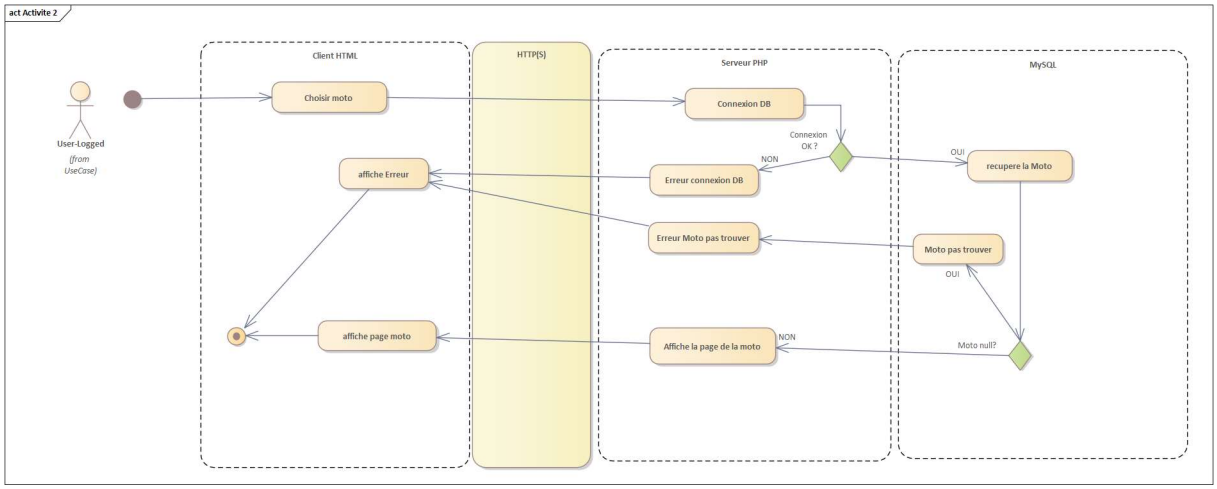
La page « Register » :

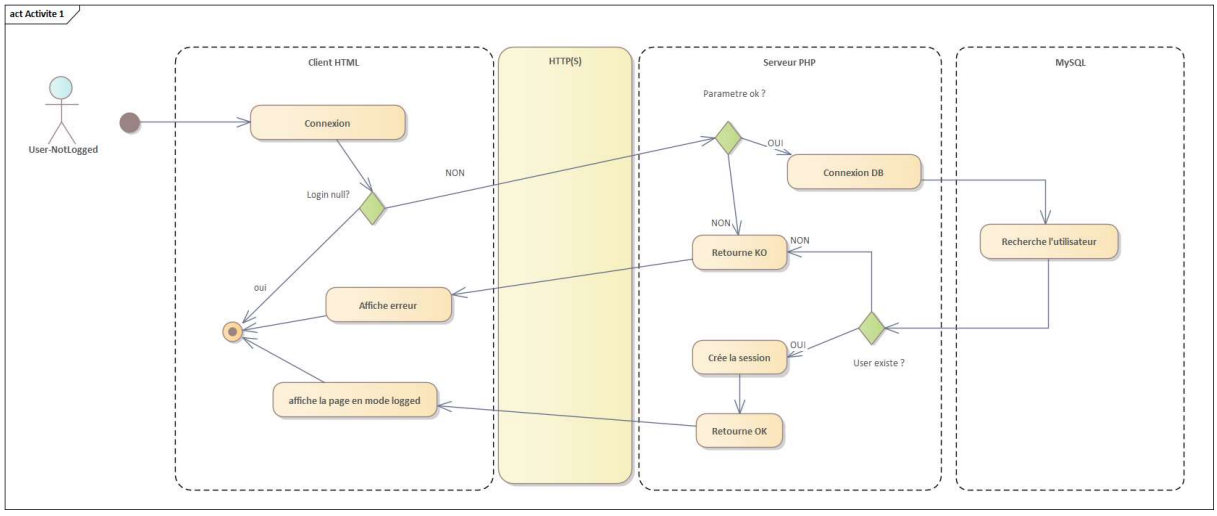


La page « Moto » :

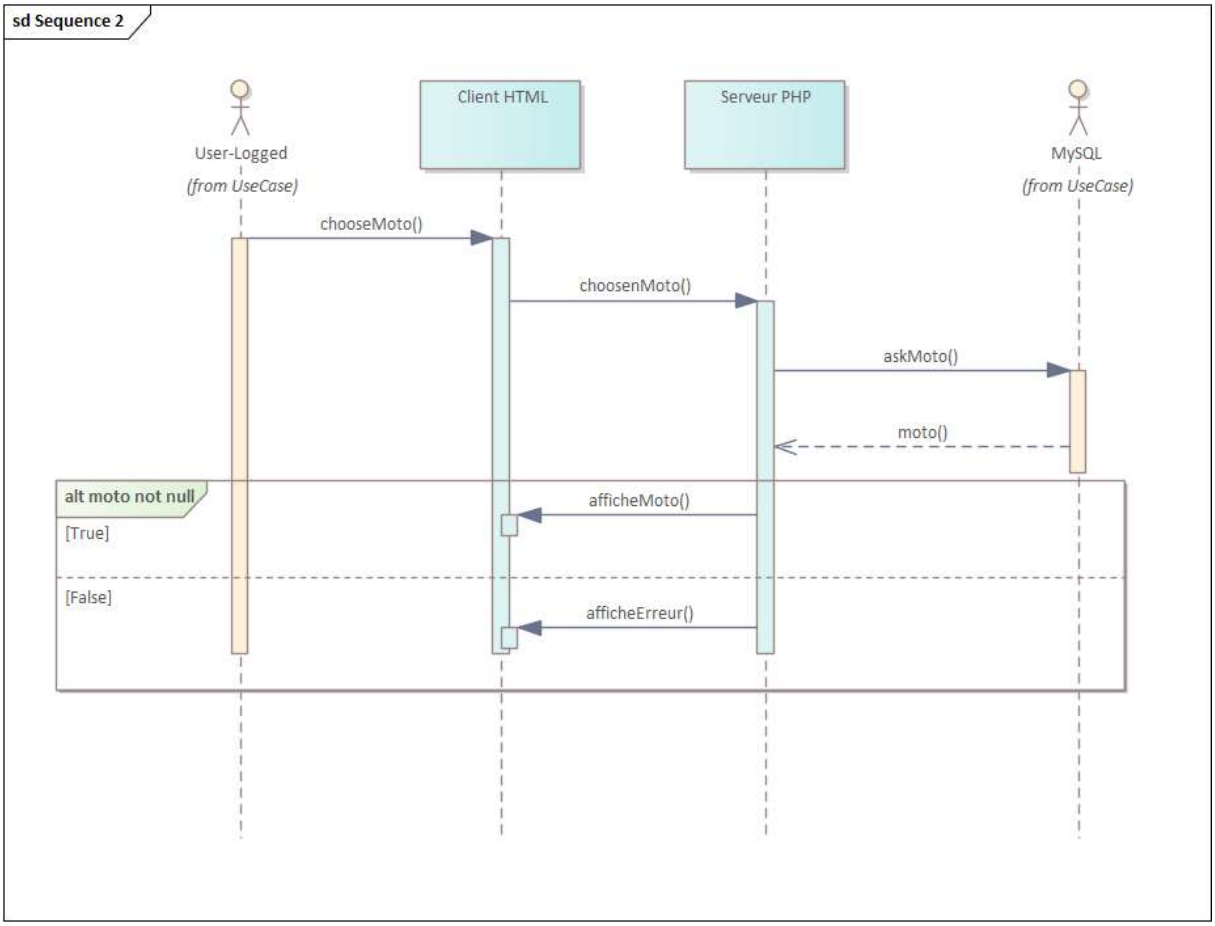


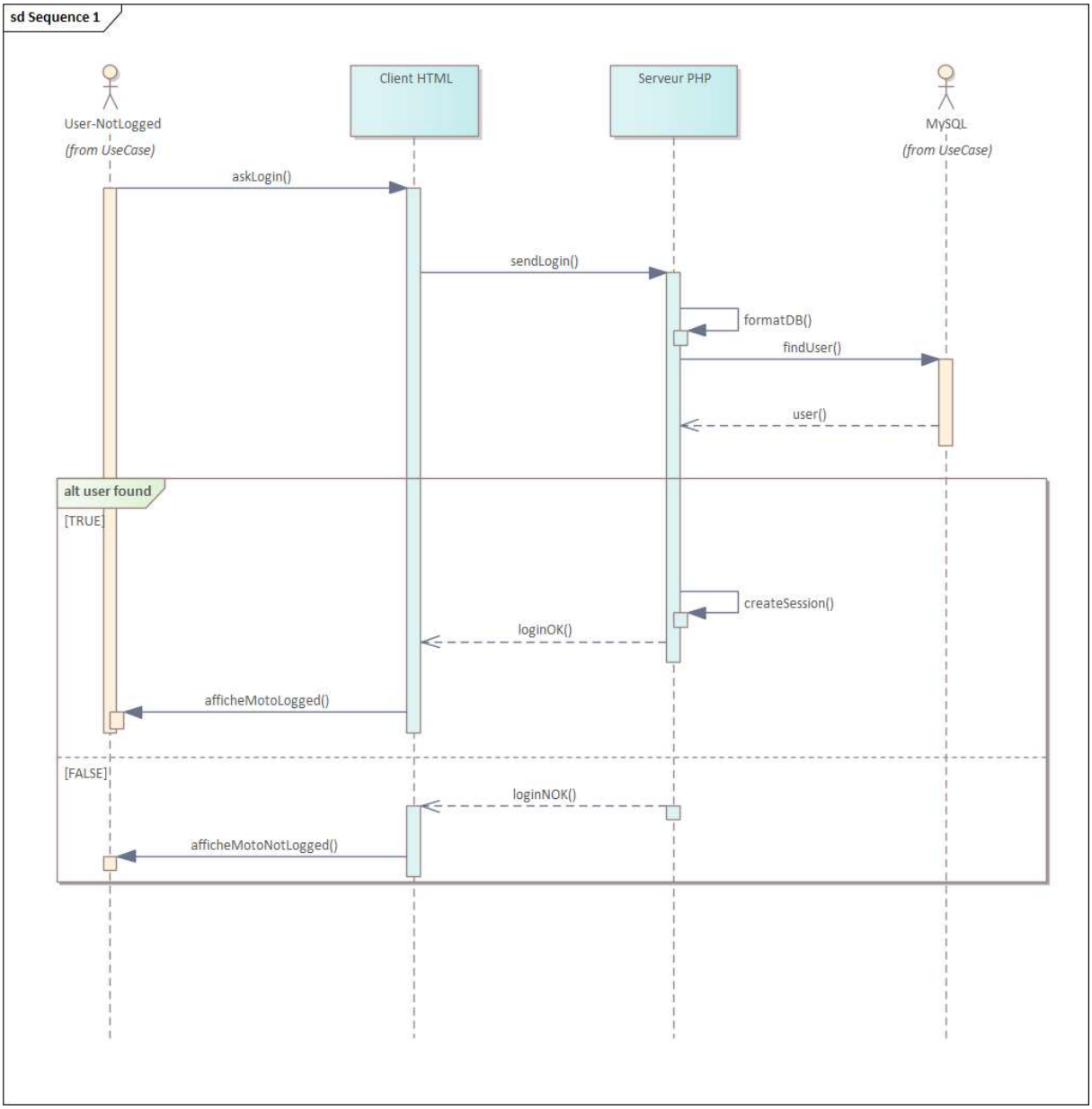
1.1.4 Diagramme activité



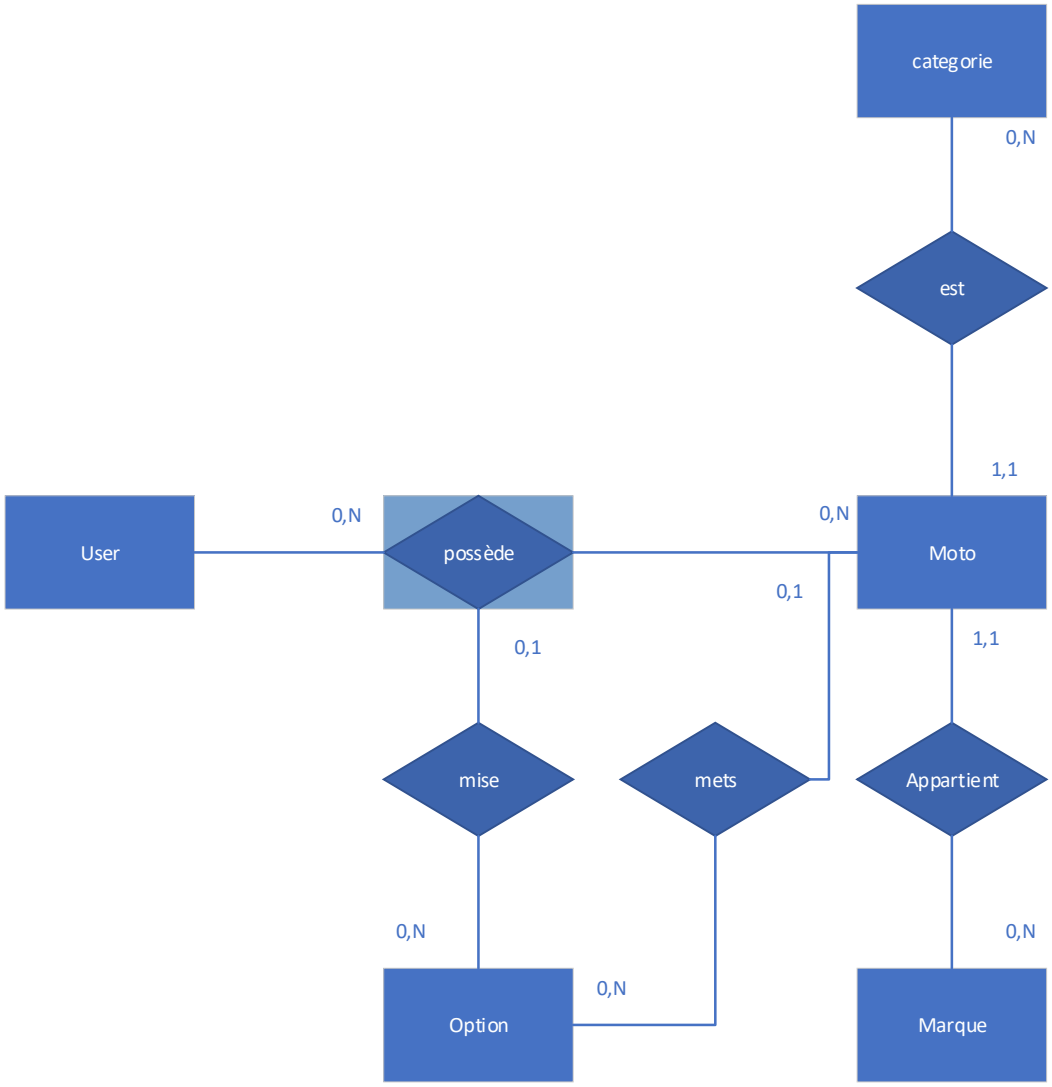


1.1.5 Diagramme de séquences systèmes





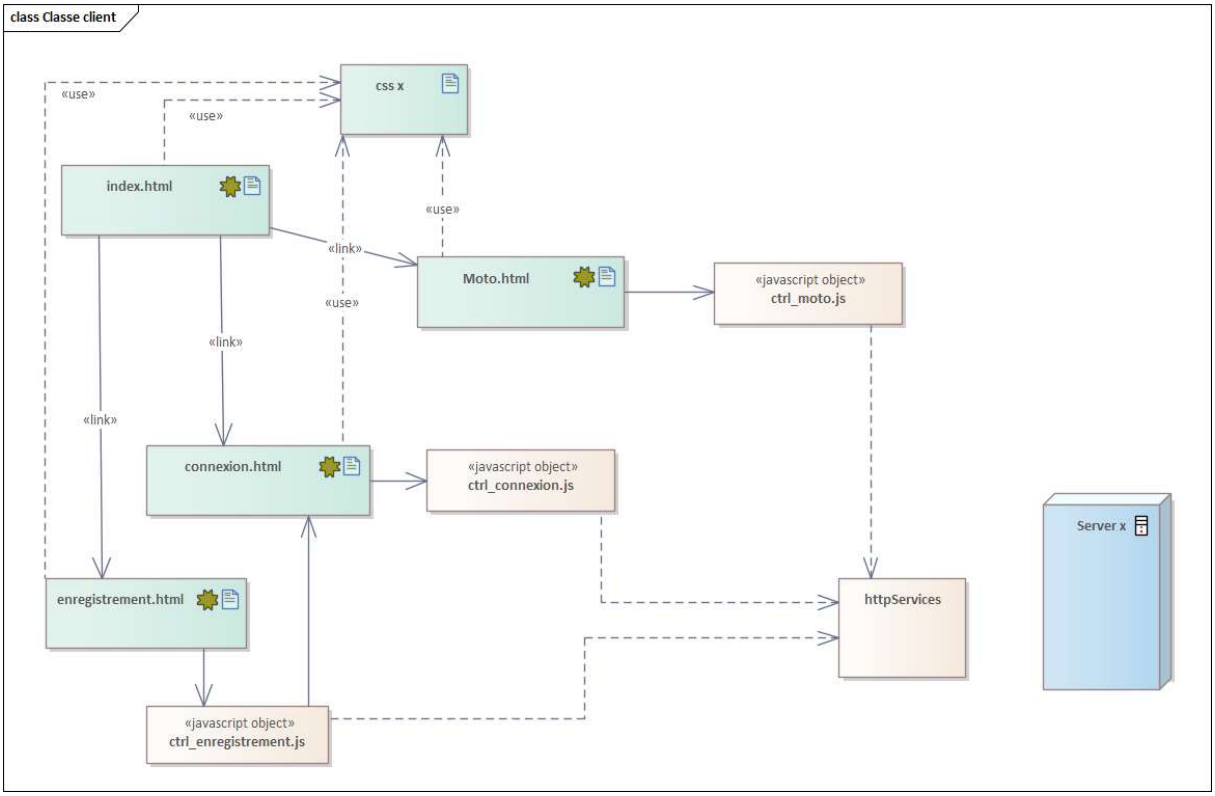
1.1.6 Schéma ER



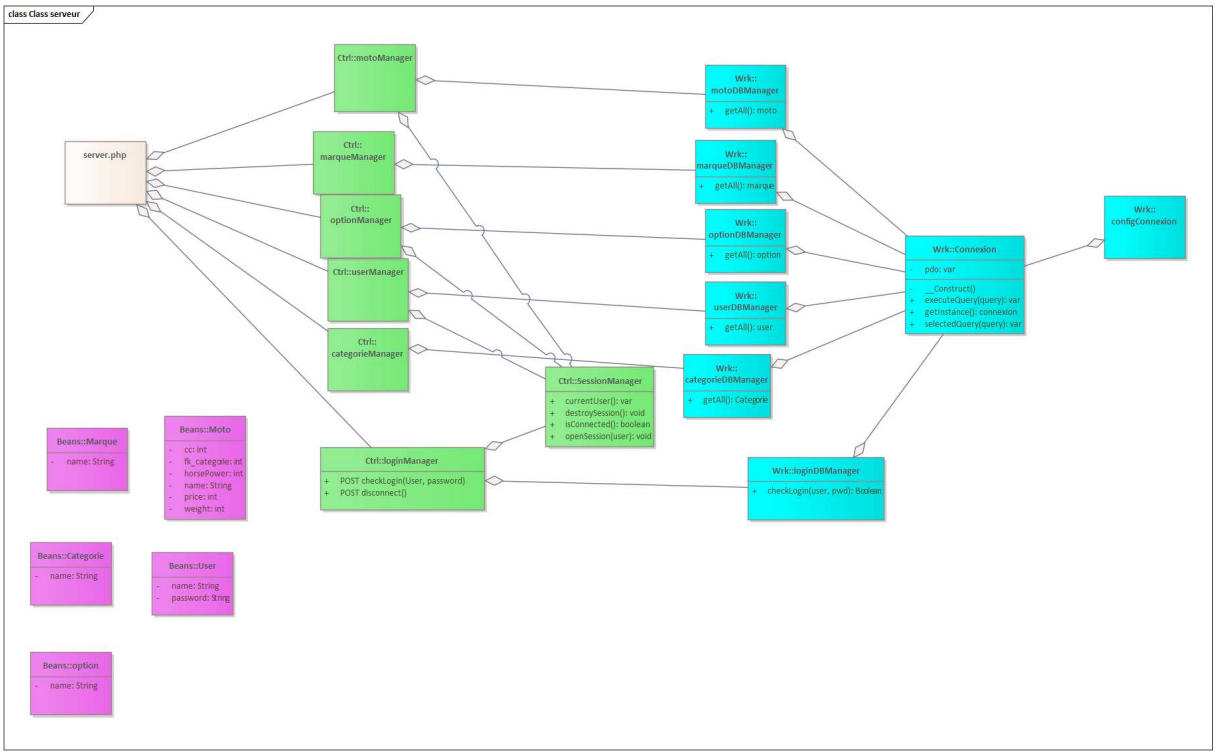
1.2 Conception

1.2.1 Diagrammes de classe

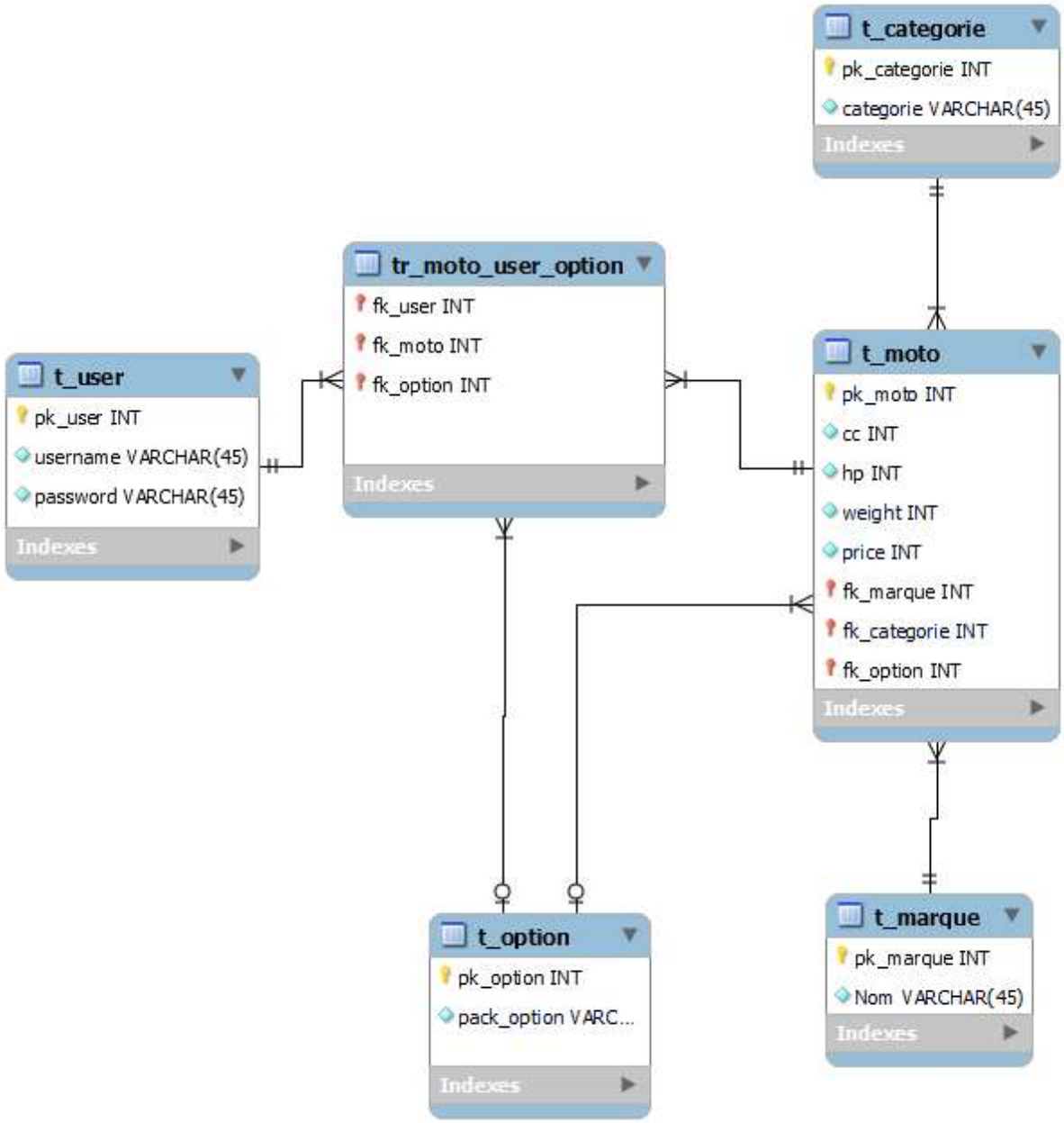
a) Client



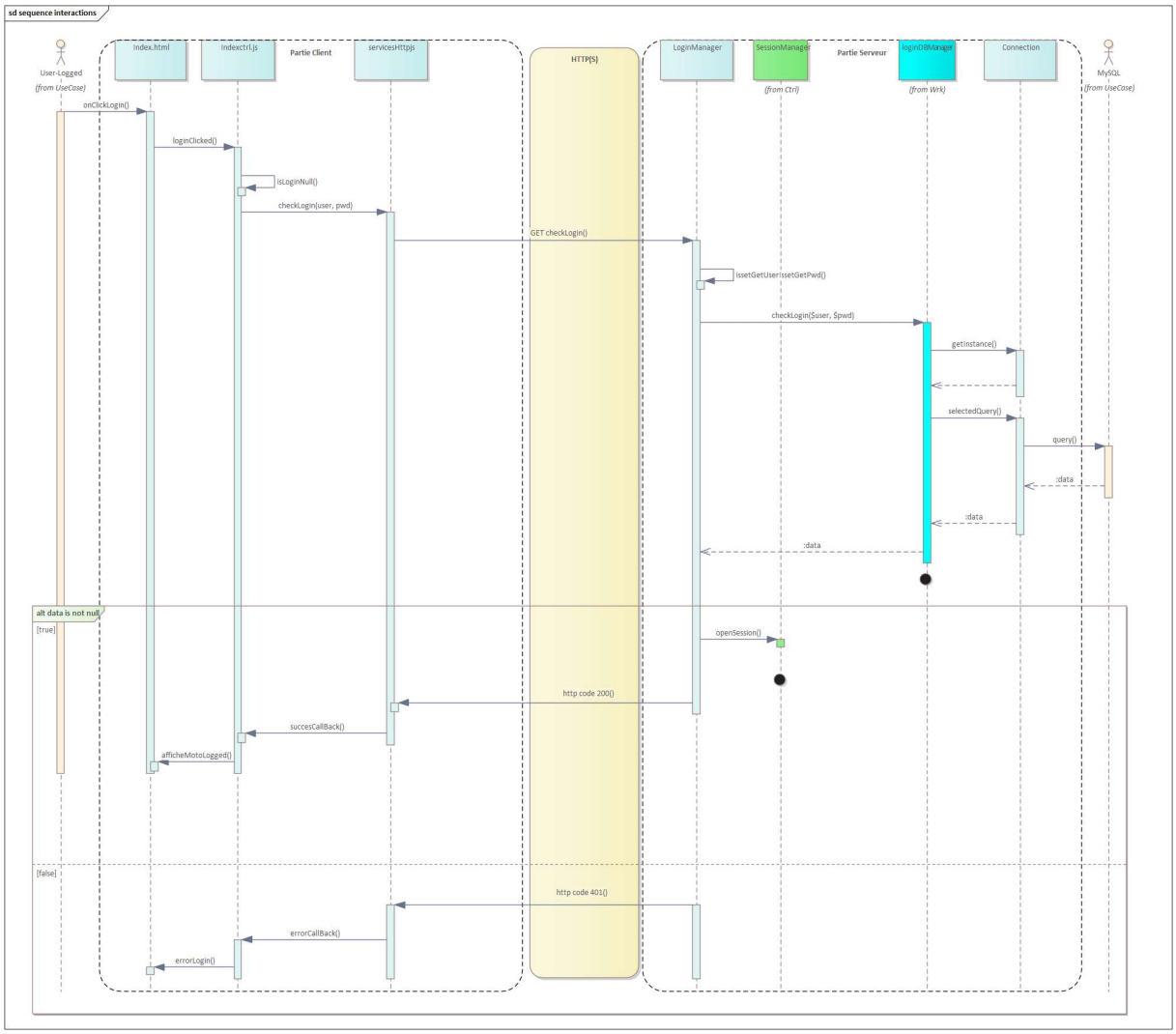
b) Serveur

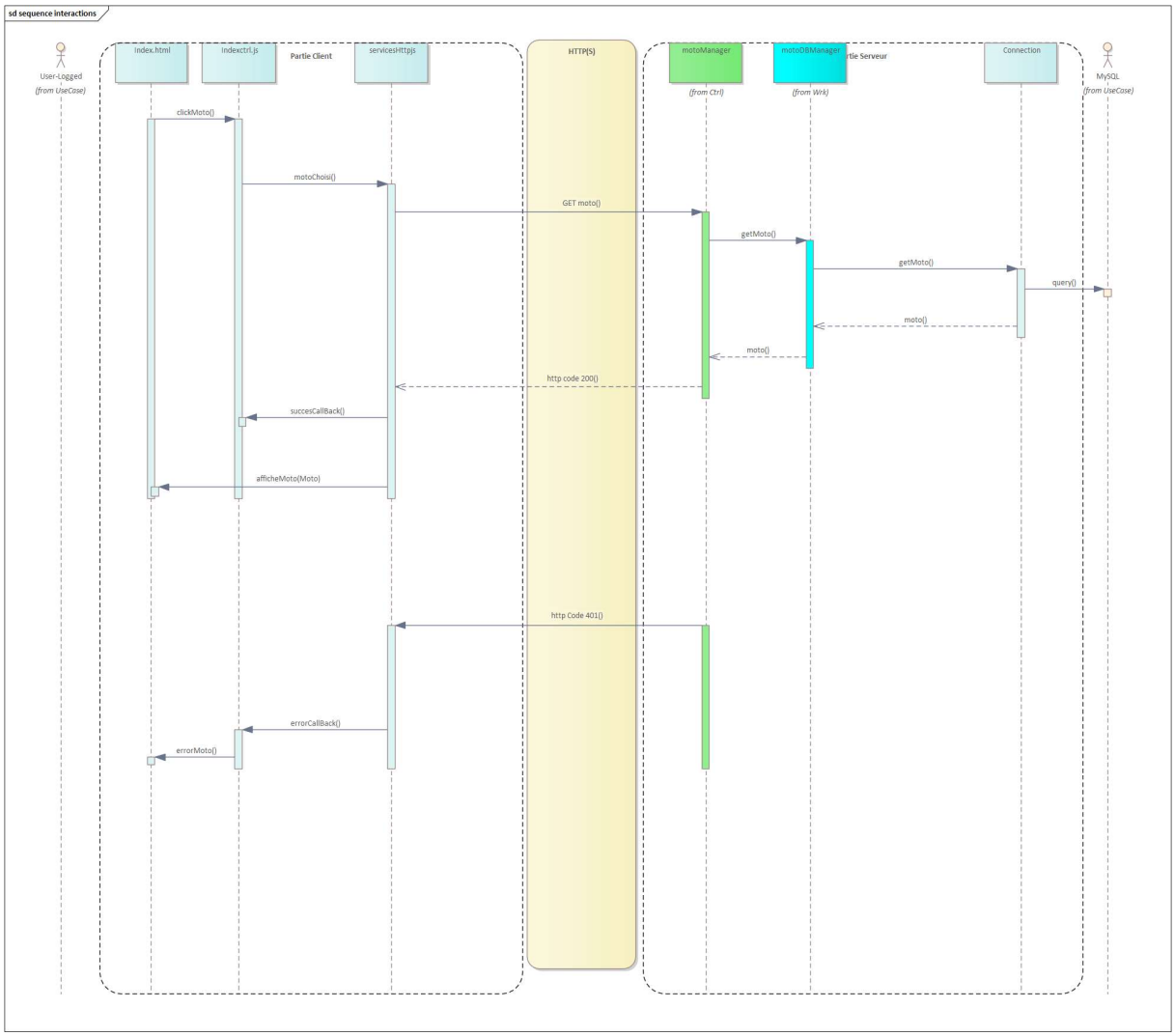


1.2.2 Schéma relationnel



1.2.3 Diagramme séquence interactions


sd sequence interactions
User-Logged
(from UseCase)
index.html
indexctrl.js
Partie Client
servicesHttpjs
HTTP(S)
LoginManager
SessionManager
(from Ctrl)
Partie Serveur
loginDBManage
(from Wrk)
Connection
MySQL
(from UseCase)
onClickLogin()
loginClicked()
isLoginNull()
checkLogin(user, pwd)
GET checkLogin()
issetGetUserissetGetPwd()
checkLogin($user, $pwd)
getinstance()
selectedQuery()
query()
:data
:data
:data
alt data is not null
[true]
openSession()
http code 200()
succesCallBack()
afficheMotoLogged()
[false]
http code 401()
errorCallBack()
errorLogin()



1.2.4 Conception des tests

1. Test de Navigation :

Vérifier que toutes les pages sont accessibles depuis le menu de navigation.

Tester les liens internes et externes pour s'assurer qu'ils dirigent vers les pages appropriées.

Vérifier la cohérence de la navigation sur différents appareils et résolutions.

2. Test d'Interface Utilisateur :

Vérifier que l'interface utilisateur connecte est la bonne

Tester les fonctionnalités interactives telles que les formulaires, les boutons, etc.

Vérifier la cohérence du design sur différents navigateurs et appareils.

3. Test de Fonctionnalités Spécifiques :

Vérifier le bon fonctionnement des fonctionnalités spécifiques

Tester les cas d'utilisation les plus courants ainsi que les scénarios exceptionnels.

4. Test de Sécurité :

Vérifier la sécurité des données en testant les entrées utilisateur pour prévenir les attaques XSS, CSRF, injection

Tester l'authentification et l'autorisation pour s'assurer que seuls les utilisateurs autorisés ont accès à certaines fonctionnalités.

1.3 Implémentation

1.3.1 Descente de code

Register :

Register.htlml :

```
<div class="login-page">
    <div class="form">
        <form class="register-form" id="registerForm">
            <input  id="username" type="text" placeholder="name" />
            <input id="password" type="password" placeholder="password" />
            <input id="passwordagain" type="password" placeholder="password
again" />
            <button>create</button>
            <p class="message">Already registered? <a href="login.html">Sign
in</a></p>
        </form>

        </form>
    </div>
</div>
```

Ctrl_register.js :

```
$(document).ready(() => {
    $("#registerForm").submit((e) => {
        handleSubmit(e);
    });
});

function handleSubmit(event) {
    event.preventDefault();

    const username = $("#username").val();
    const password = $("#password").val();
    const passwordagain = $("#passwordagain").val();

    if (password === passwordagain) {
        // Appeler la fonction createUserAjax avec les valeurs des champs
        createUserAjax(username, password, function (response) {
            // Succès de la requête
            console.log("Réponse du serveur:", response);

            window.location.href = "http://localhost:8083/indexlogged.html";
```

```
            // Afficher un message d'erreur si la création de compte a échoué


        }, function (xhr, status, error) {
            // Erreur de la requête
            console.error("Erreur de requête:", status, error);
            alert("Erreur lors de la requête AJAX. Veuillez réessayer.");
        });
    } else {
        // Afficher un message si les mots de passe ne correspondent pas
        alert("Les mots de passe ne correspondent pas.");
        $("#password").val('');
        $("#passwordagain").val('');
    }

    // Réinitialiser les champs de saisie
    $("#username").val('');
    $("#password").val('');
    $("#passwordagain").val('');
}
```

Httpservice.js :

```
function createUserAjax(username, password, successCallback, errorCallback) {
    $.ajax({
        type: "POST",
        contentType: "application/json", // Définir le type de contenu comme
JSON
        url: BASE_URL + "server.php",
        data: JSON.stringify({
            action: "createUser",
            username: username,
            password: password
        }), // Convertir les données en JSON
        xhrFields: {
            withCredentials: true
        },
        success: successCallback,
        error: errorCallback
    });
}
```

Server.php :

```
case 'createUser':
            $username = initVariableFromJson("username");
            $password = initVariableFromJson("password");

            if (isset($username)) {
                if (isset($password)) {
```

```
                //ctrl login
                $usermang = new UserManager($username, $password);




            }
        }
```

UserManager.php :

```
public function __construct( $username,$password)
    {
        // Créez une instance de UserDBManager
        $this->wrk = new UserDBManager();


        // Appelez la méthode pour créer un utilisateur avec le nom d'utilisa-
teur et le mot de passe fournis
        $this->wrk->createUser($username, $password);

    }
```

UserDBManager.php

```
public function createUser($username, $password)
    {
        $json = "";
        $query = "SELECT * FROM t_user WHERE username=:username";
        $params = array("username" => $username);
        $isUsernameTaken = $this->connexion->selectSingleQuery($query, $pa-
rams);
        if (!$isUsernameTaken) {
            $query = "INSERT INTO t_user (username, password) VALUES (:user-
name, :password)";
            $password = htmlspecialchars($password);
            $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
            $params = array('username' => htmlspecialchars($username), 'pass-
word' => $hashedPassword);
            $this->connexion->executeQuery($query, $params);

            $pkUser = $this->connexion->getLastId('t_user');
            $response = array(
                'success' => true,
                'pk' => $pkUser,
                'username' => $username,
                'message' => 'compte creer'
            );
            http_response_code(200);
```

```
            $json = json_encode($response, JSON_UNESCAPED_UNICODE);
        } else {
            http_response_code(401);
            $response = array(
                'success' => false,
                'message' => "Ce nom d'utilisateur n'est pas disponible"
            );
            $json = json_encode($response, JSON_UNESCAPED_UNICODE);
        }
        //  header('Content-Type: application/json; charset=utf-8');
        echo $json;
        return $response;
    }
```

1.3.2 Problèmes rencontrés

J'ai eu pas mal de problème lier au CORS et aussi au moment de l'hébergement.

Ce qui ma fait perdre pas mal de temps c'est l'installation de docker, vsc, Workbench … sur mon pc portable car celui de l'école avait été réinitialisez

1.3.3 Tests fonctionnels

La plupart de mes tests ont été fait en local.

Le login et la création d'utilisateur fonctionne.

La récupération de moto aussi

1.3.4 Hébergement

Mon site est héberge chez Tizoo :

Valleliane.emf-informatique.ch/151/client/index.html

1.4 Synthèse

1.4.1 Conclusion

J'ai apprécié ce module malgré le fait que je n'ai pas réussi à temps. Je pense que de commencer le projet plus rapidement ou de voir PHP et du MVC2 (web) avant ce module pourrait faciliter ce module.