151

Documentation de projet

Date de création : 09.02.2024
Version 1

Jaquier Loïc

Ecole des Métiers
Berufsfachschule
Fribourg
Informatique

Table des matières

1 Analyse

1.1 Présentation du projet

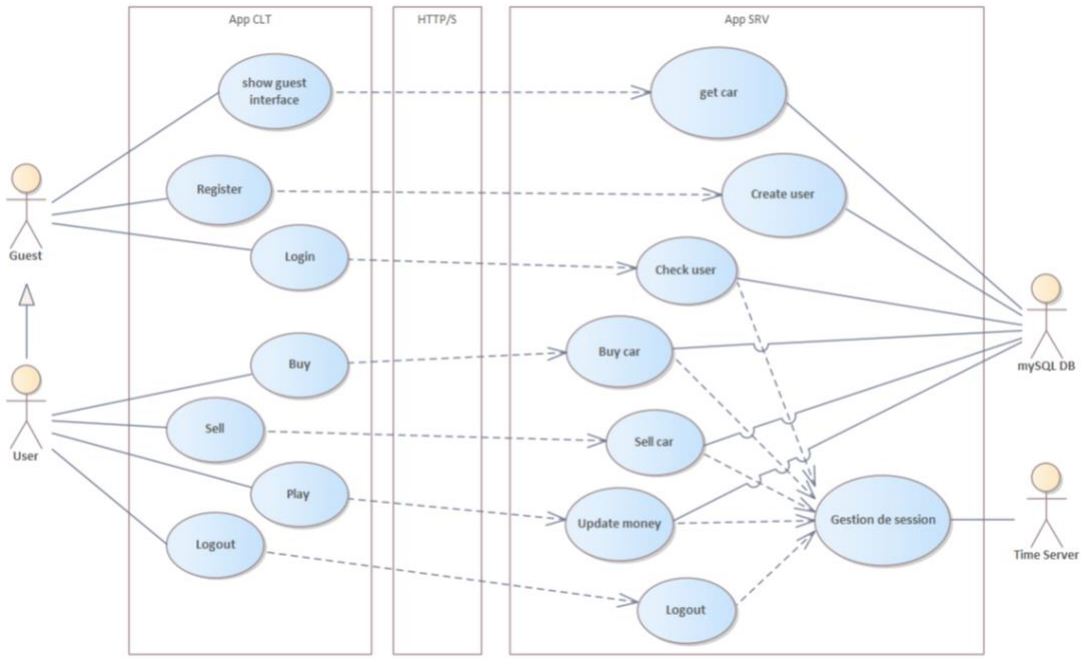
Mon projet consistera en une plateforme de vente de voiture. Il y aura donc plusieurs pages pour convenir à tous les besoins de l'application. Une page de connexion, une page d'enregistrement, un shop (accueil), un garage et finalement une page pour enregistrer de nouvelles voitures.

Lorsqu'un utilisateur arrive sur le site, il pourra voir le catalogues de voitures disponibles. Il aura la possibilité de se connecter et s'il ne possède pas de compte, il pourra en créer un. Une fois connecté, l'utilisateur verra alors sur la page son nom d'utilisateur ainsi que son argent disponible. Il aura également la possibilité d'accéder à son garage (pour voir les voitures qu'il possède) et à la page d'enregistrement de nouveau véhicules.

Un utilisateur connecté pourra également vendre et acheter des voitures sur le magasin.

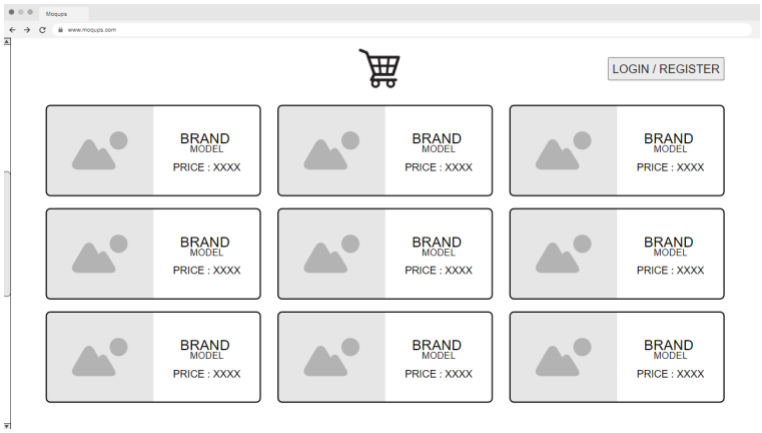
La DB contiendra les utilisateurs enregistrés, les transactions effectuées, les voitures ainsi que leurs marques correspondantes.

1.2 Uses Case

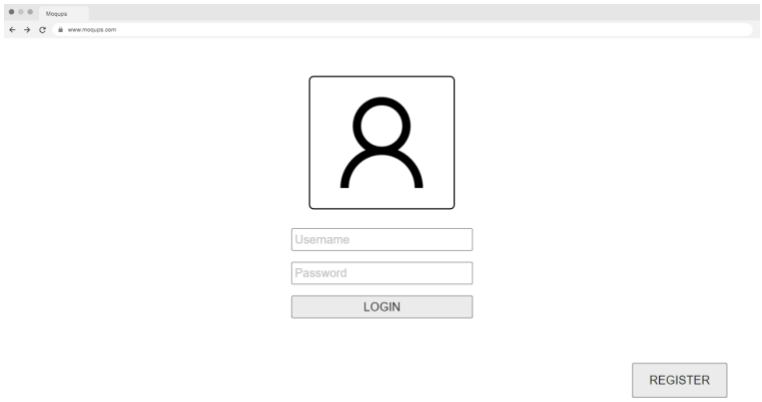


1.3 Maquettes

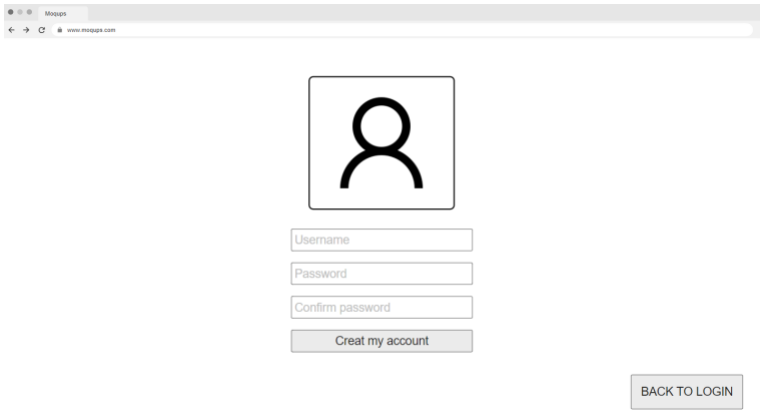
1.3.1 Shop - Home (Guest)



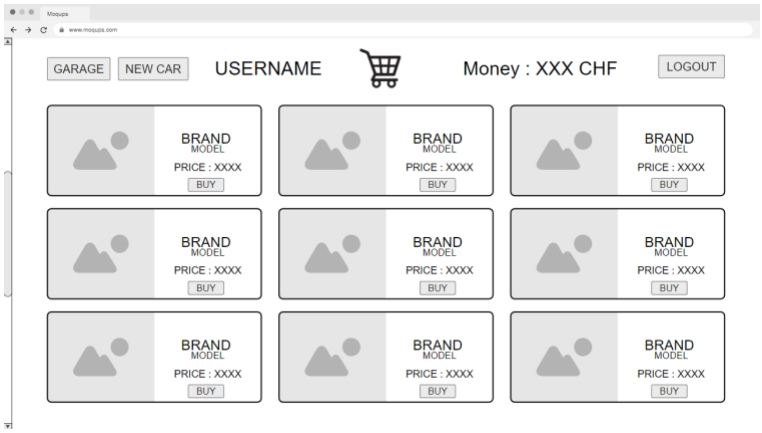
1.3.2 Login



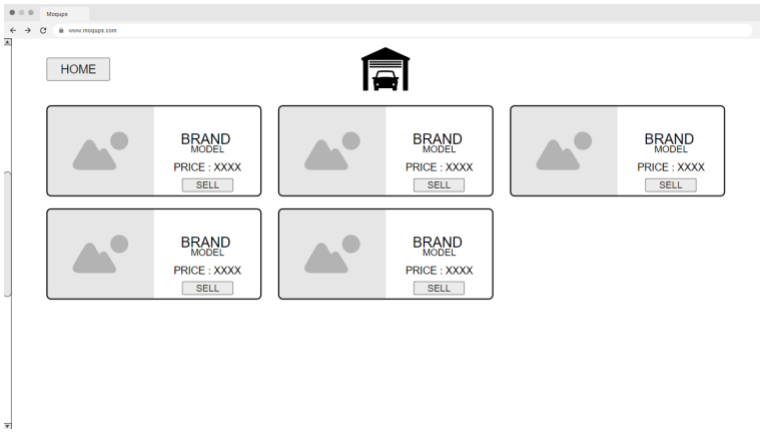
1.3.3 Register



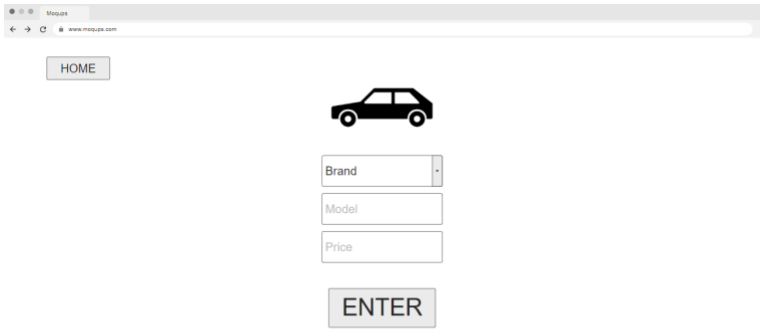
1.3.4 Shop - home (Connected)



1.3.5 Garage

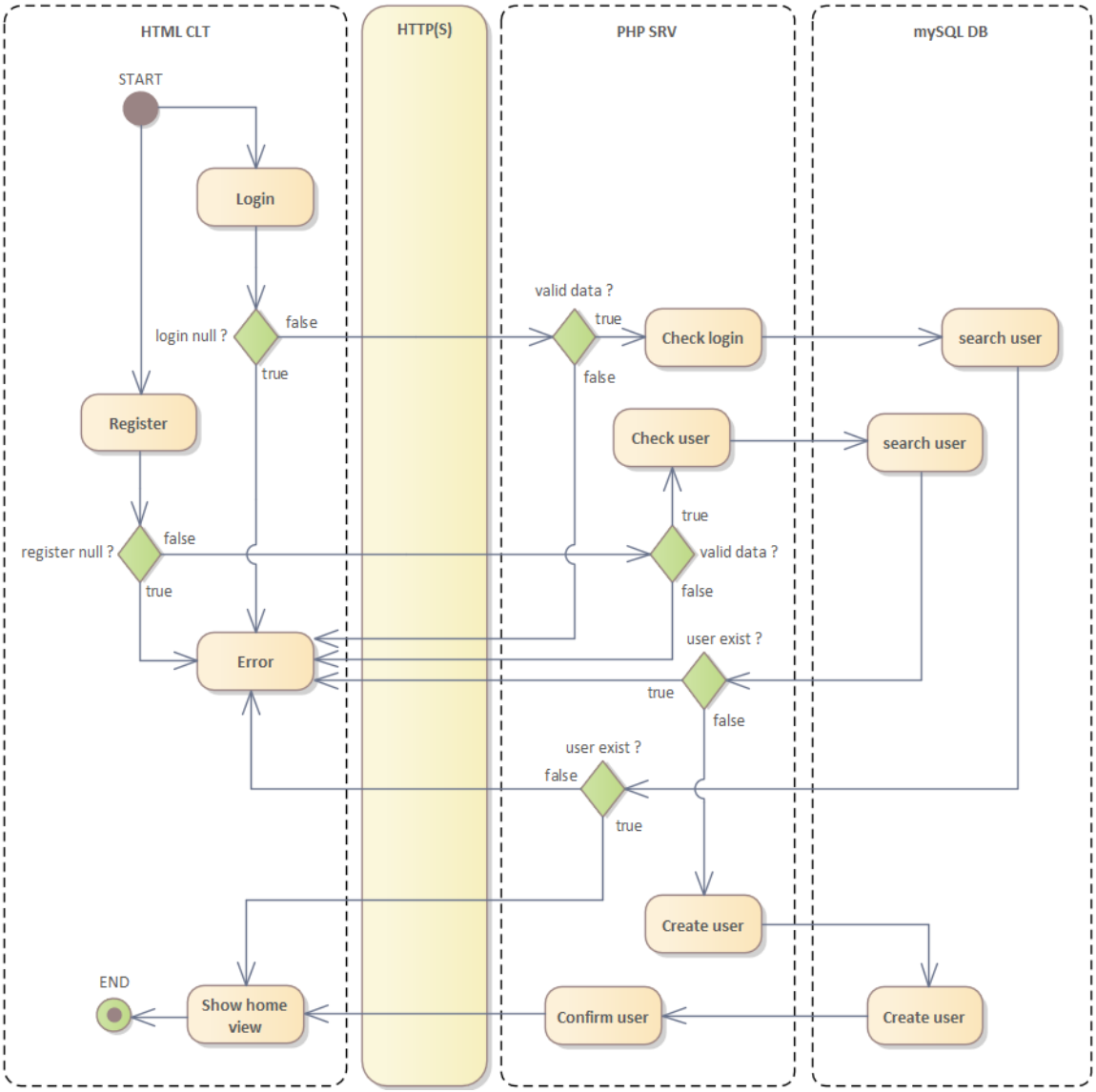


1.3.6 Enregistrement de voiture

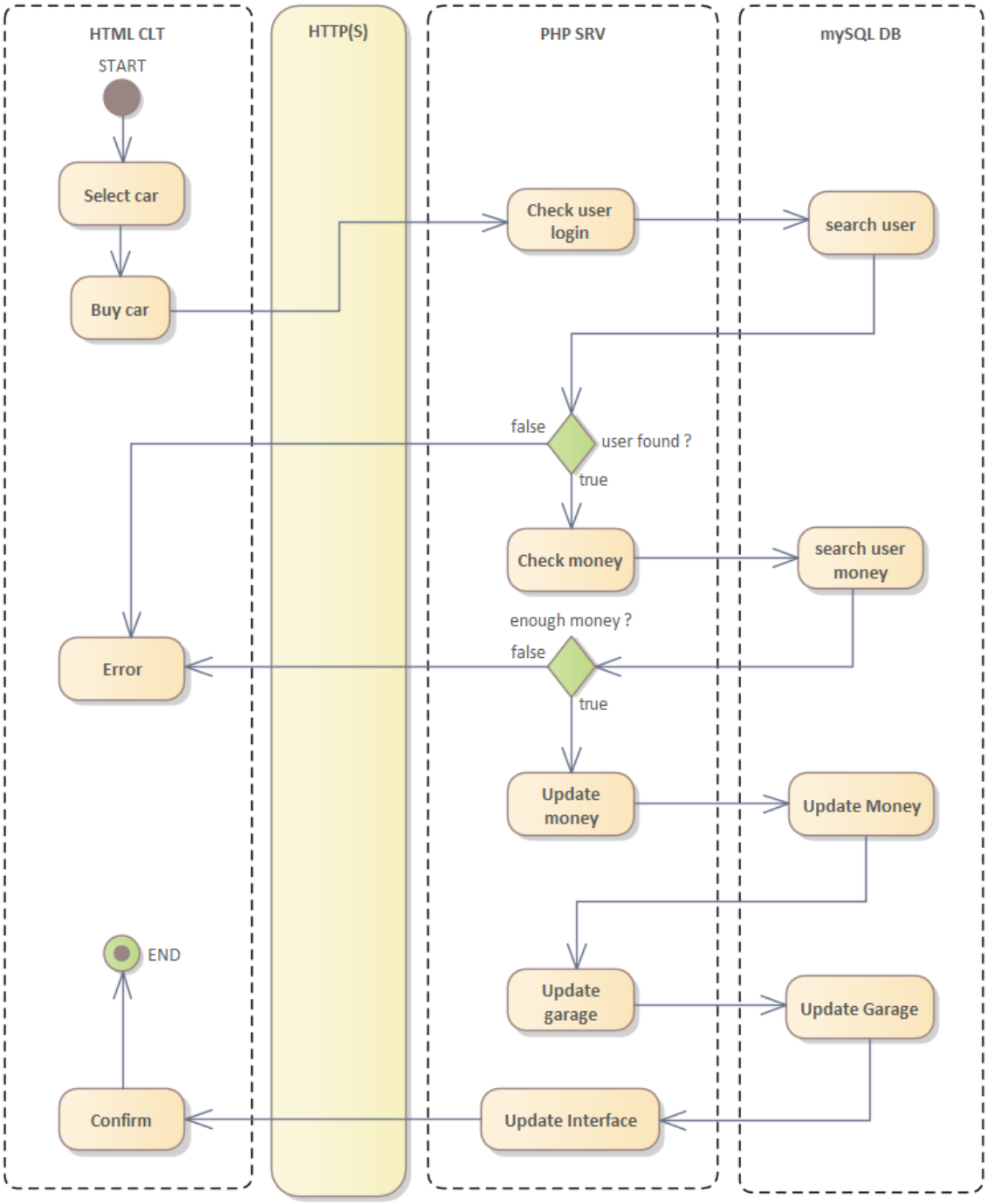


1.4 Diagramme activité

1.4.1 Connexion

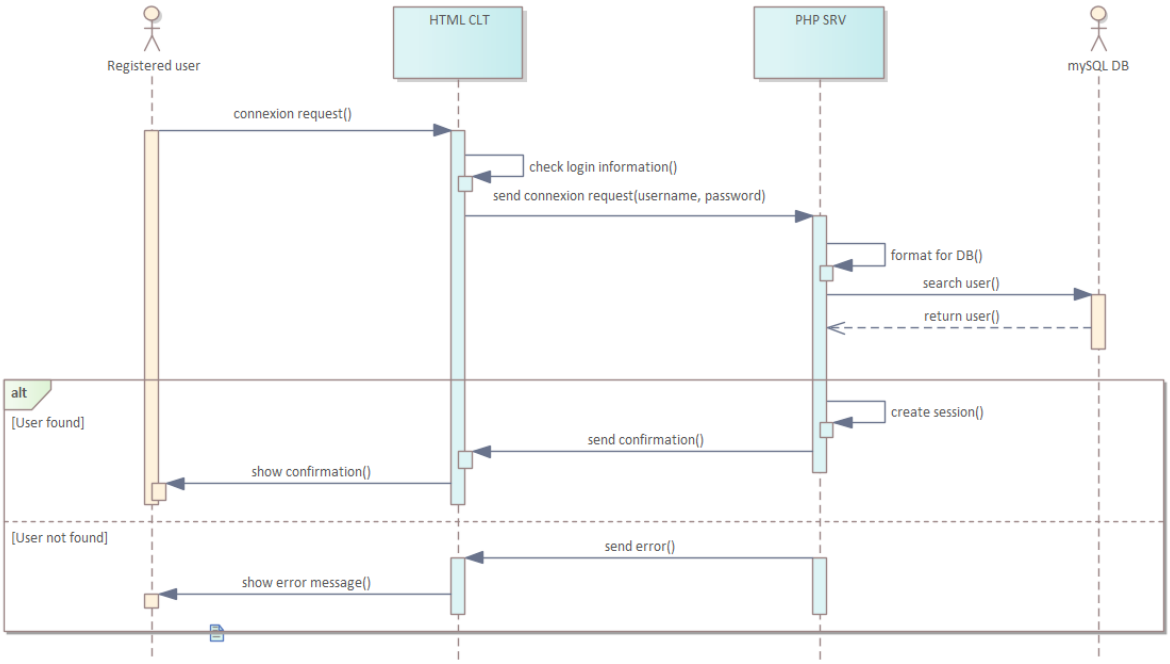


1.4.2 Achat voiture

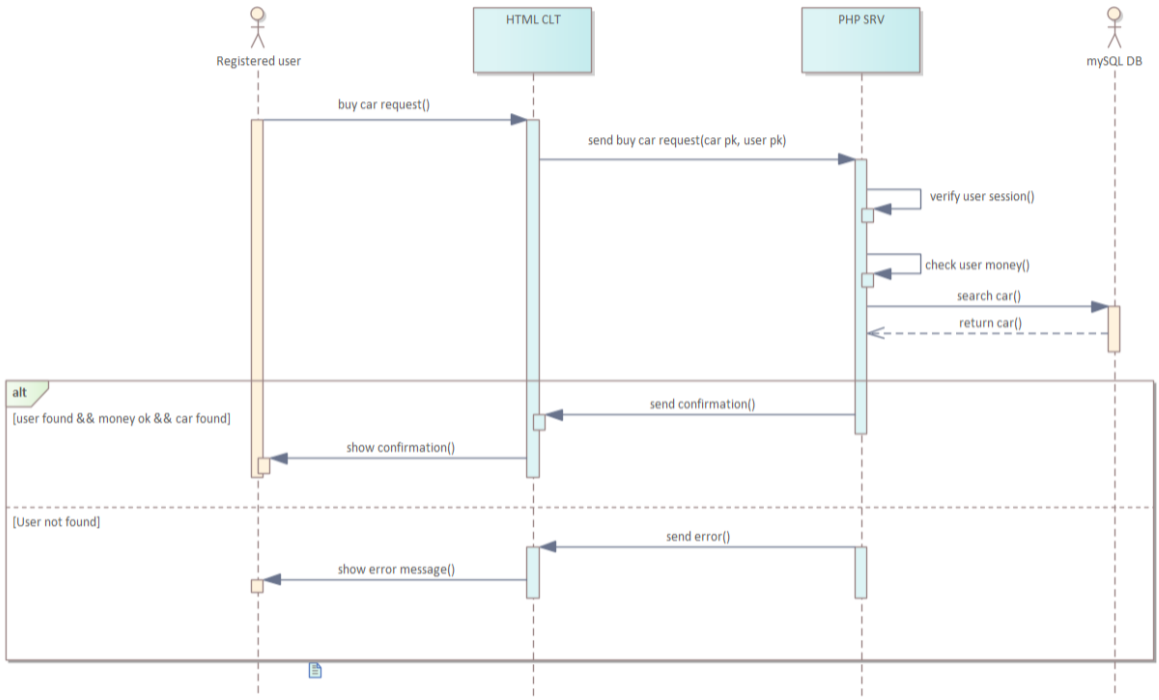


1.5 Diagramme de séquences systèmes

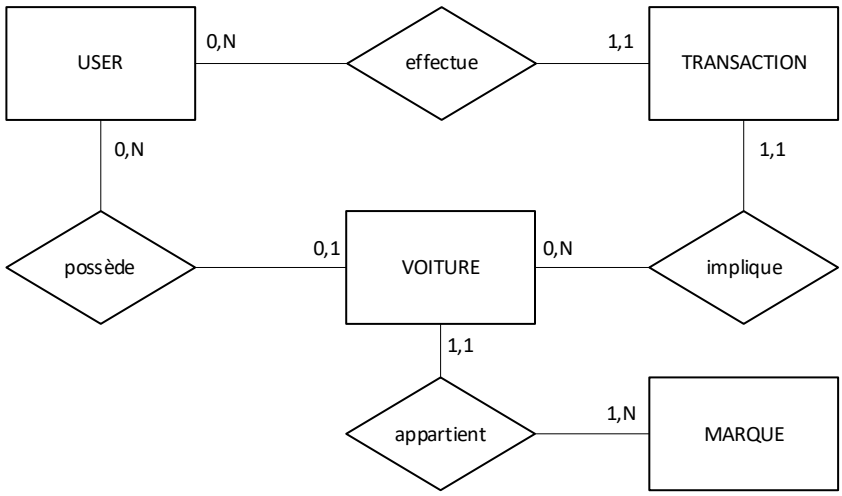
1.5.1 Connexion



1.5.2 Achat voiture



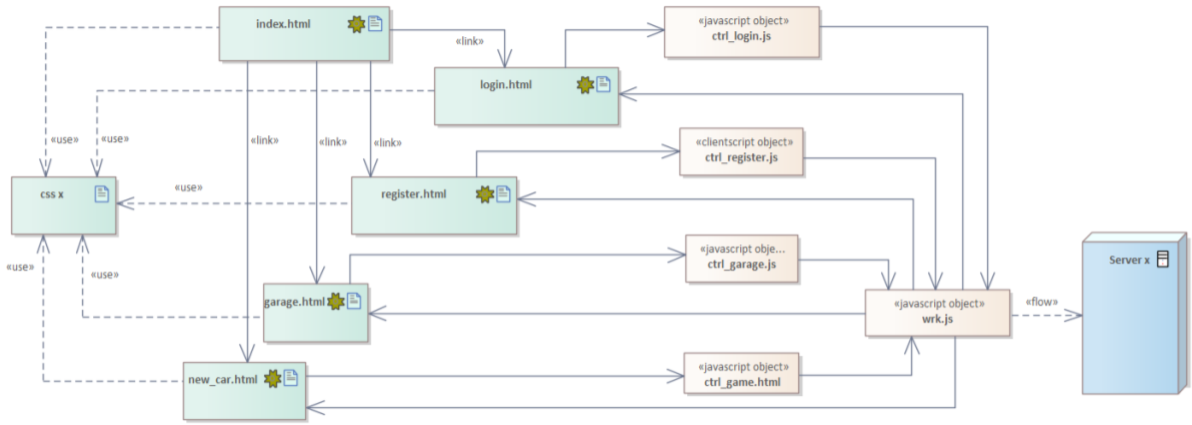
1.6 Schéma ER


USER
0,N
effectue
1,1
TRANSACTION
0,N
1,1
possède
0,1
VOITURE
0,N
implique
1,1
appartient
1,N
MARQUE

2 Conception

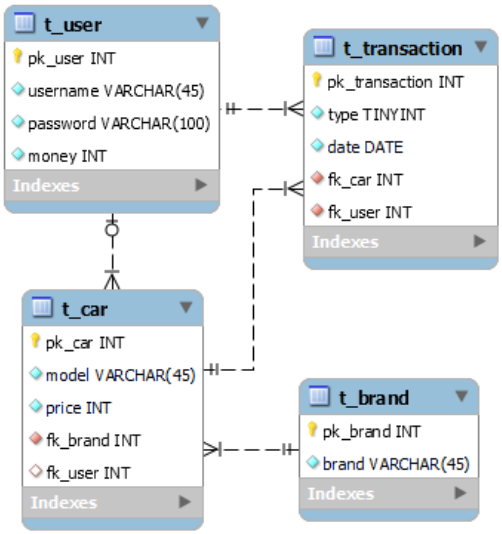
2.1 Diagrammes de classe

2.1.1 Client



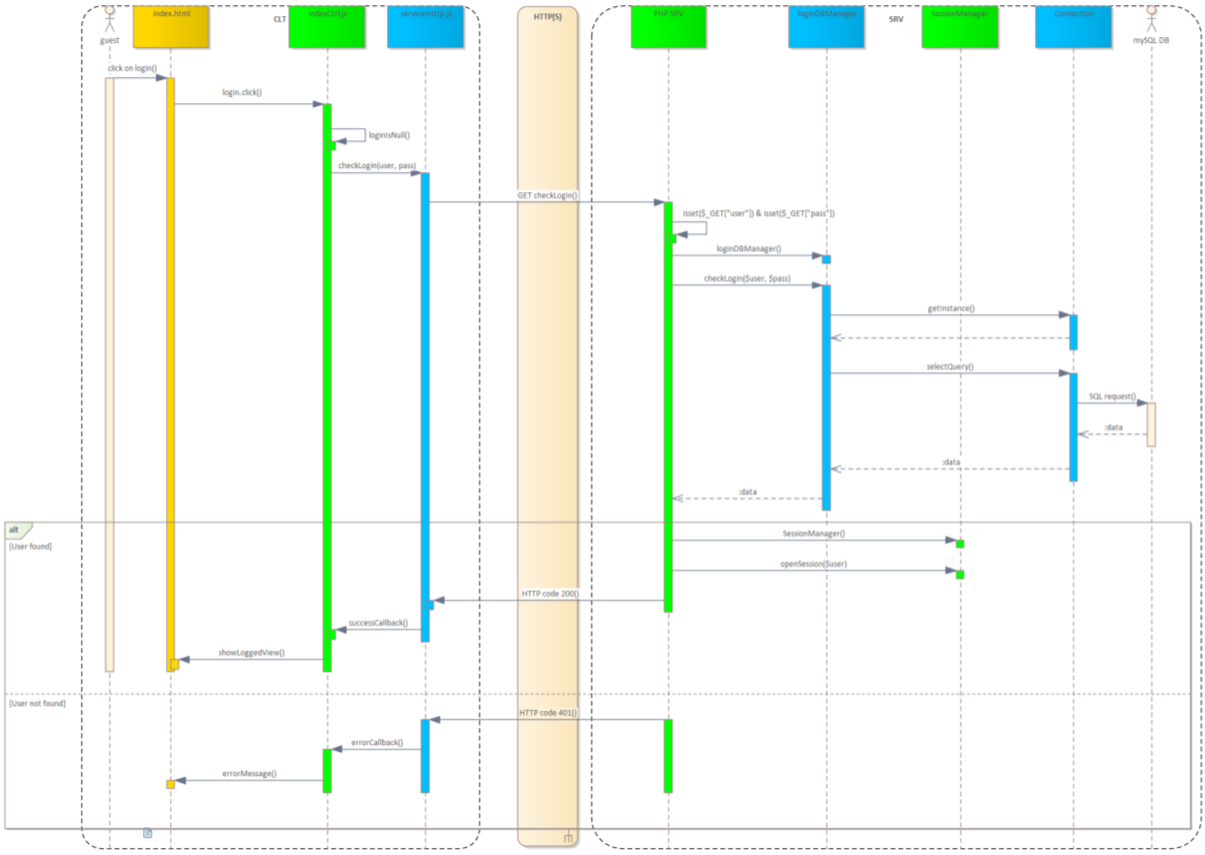
2.1.2 Serveur

2.2 Schéma relationnel

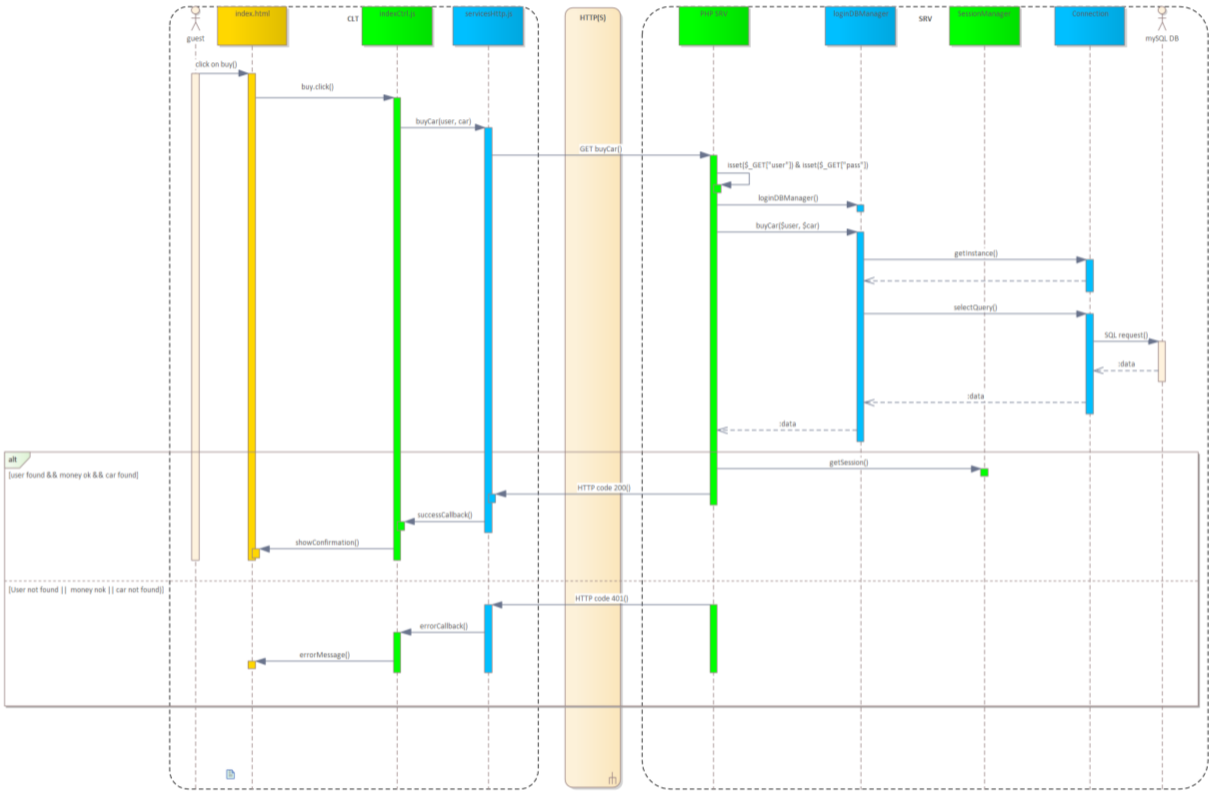


2.3 Diagramme séquence interactions

2.3.1 Connexion



2.3.2 Achat voiture



2.4 Conception des tests

2.4.1 Tests fonctionnels

N°	Description	Résultat attendu
1	Ajouter un utilisateur avec des champ vides	NOK
2	Ajouter un utilisateur avec « user » et « Pa$$w0rd »	OK
3	Ajouter un utilisateur avec « user » et « Pa$$w0rd »	NOK
4	Se connecter avec l'utilisateur « user » et « Pa$$w0rd »	OK
5	Se connecter avec l'utilisateur « user » et « password »	NOK
6	Se connecter avec l'utilisateur « abc » et « Pa$$w0rd »	NOK
7	Enregistrer une nouvelle voiture avec « Audi », « R8 » et « 100000 »	OK
8	Vendre cette voiture	OK
9	Acheter cette voiture	OK
10	Déconnexion	OK

2.4.2 Tests de sécurité

N°	Description	Résultat attendu
1	Injection SQL dans le login	NOK
2	Injection SQL dans register	NOK
3	Injection SQL dans la création de voiture	NOK
4	Injection JS dans le login	NOK
5	Injection JS dans register	NOK
6	Injection JS dans la création de voiture	NOK
7	Injection HTML dans le login	NOK
8	Injection HTML dans register	NOK
9	Injection HTML dans la création de voiture	NOK
10	Injection CSS dans le login	NOK
11	Injection CSS dans register	NOK
12	Injection CSS dans la création de voiture	NOK
13	Requêtes via Postman	NOK

3 Implémentation

3.1 Descente de code

3.1.1 Connexion

3.1.1.1 client

Index.html :

```
<button class="login-index-btn" on-
click="window.location.href='login.html'">Login</button>
```

Login.html :

```
<button class="login-btn"
onclick="loginCtrl(document.getElementById('user').value, docu-
ment.getElementById('pass').value)">Login</button>
```

ctrlLogin.js :

```
function loginCtrl(user, pass) {
    if (user.length > 0 && pass.length > 0) {
        checkUser(user, pass, successCallback, errorCallback)
    } else {
        alert('Les champs ne peuvent pas être vides !');
    }
}

function successCallback(data) {
    window.sessionStorage.setItem(StorageItems.pk_user, data.pk)
    window.sessionStorage.setItem(StorageItems.money, data.money)
    window.sessionStorage.setItem(StorageItems.username, data.username)
    getCarCtrl();
    window.location.href = 'main.html'
}

function errorCallback(data) {
    alert(data.responseJSON.message);
}
```

wrk.js :

```
function checkUser(user, pass, successCallback, errorCallback) {
    $.ajax({
        type: "POST",
        dataType: "JSON",
        url: BASE_URL,
        data: {
            "action": "checkUser",
```

```
            "user": user,
            "pass": pass
        },
        xhrFields: {
            withCredentials: true
        },
        success: successCallback,
        error: errorCallback
    });
}
```

3.1.1.2 server

index.php

```
        case 'checkUser':
            if (isset($_POST['user'])) {
                if (isset($_POST['pass'])) {
                    $user->checkUser($_POST['user'], $_POST['pass'], $session);
                }
            }
            break;
```

ctrlUser.php :

```
    public function checkUser($username, $password, $session)
    {
        return $this->manager->selectUser($username, $password, $session);
    }
```

wrkUser.php :

```
    public function selectUser($username, $password, $session)
    {
        $query = "SELECT * FROM t_user WHERE username = :username";
        $params = array("username" => htmlspecialchars($username));
        $response = $this->connexion->selectSingleQuery($query, $params);
        $obj = new User();
        if ($response) {
            $obj->initFromDb($response);
            $bool = password_verify($password, $response['password']);
            if ($bool) {
                http_response_code(200);
                echo json_encode(
                    array(
                        'success' => true,
                        'message' => 'Utilisateur connecté',
                        'username' => $obj->getUsername(),
                        'pk' => $obj->getPKUser(),
```

```
                        'money' => $obj->getMoney()
                    ),
                    JSON_UNESCAPED_UNICODE
                );
                $session->set('pk', $obj->getPKUser());
            } else {
                http_response_code(401);
                echo json_encode(
                    array(
                        'success' => false,
                        'message' => 'Identifiants incorrects !'
                    ),
                    JSON_UNESCAPED_UNICODE
                );
            }
        } else {
            http_response_code(401);
            echo json_encode(
                array(
                    'success' => false,
                    'message' => 'Identifiants incorrects !'
                ),
                JSON_UNESCAPED_UNICODE
            );
        }
    }
```

3.1.2 Achat voiture

3.1.2.1 client

main.html :

```
<button class="buy_button" id="carBtn${car.pk_car}"
onclick="buyCarCtrl(${car.pk_car}, ${car.price})">BUY</button>
```

ctrlMain.js :

```
function buyCarCtrl(carId, price) {
  buyCar(window.sessionStorage.getItem(StorageItems.pk_user), carId,
buyCarSuccessCallback, buyCarErrorCallback);
}

function buyCarSuccessCallback(data) {
  alert(data.message);
}

function buyCarErrorCallback(data) {
  alert(data.responseJSON.message);
```

```
  console.log(data);
}
```

wrk.js :

```
function buyCar(user, car, successCallback, errorCallback) {
    $.ajax({
        type: "POST",
        dataType: "JSON",
        url: BASE_URL,
        data: {
            "action": "buyCar",
            "user": user,
            "car": car
        },
        xhrFields: {
            withCredentials: true
        },
        success: successCallback,
        error: errorCallback
    });
}
```

3.1.2.2 server

index.php :

```
        case 'buyCar':
            if (isset($_POST['user'])) {
                if (isset($_POST['car'])) {
                    if ($session->get('pk')) {
                        $car->buyCar($_POST['user'], $_POST['car']);
                    } else {
                        http_response_code(401);
                        echo json_encode(
                            array(
                                'success' => false,
                                'message' => 'Erreur de session',
                            ),
                            JSON_UNESCAPED_UNICODE
                        );
                    }
                }
            }
```

ctrlCar.php :

```
    public function buyCar($user, $car)
    {
```

```
        echo $this->manager->updateBuyingCar($user, $car);
    }
```

wrkCar.php :

```
    public function updateBuyingCar($userId, $carId)
    {
        $getUserMoneyQuery = "SELECT money FROM db_151.t_user WHERE pk_user =
:userId";
        $getUserMoneyParams = array('userId' => $userId);
        $userData = $this->connexion->selectSingleQuery($getUserMoneyQuery,
$getUserMoneyParams);
        $getCarPriceQuery = "SELECT price, fk_user FROM db_151.t_car WHERE pk_car
= :carId";
        $getCarPriceParams = array('carId' => $carId);
        $carData = $this->connexion->selectSingleQuery($getCarPriceQuery,
$getCarPriceParams);
        $money = $userData['money'];
        $price = $carData['price'];
        $fk_user = $carData['fk_user'];
        if ($fk_user == null) {
            if ($money >= $price) {
                try {
                    $updateCarQuery = "UPDATE db_151.t_car SET fk_user = :userId
WHERE pk_car = :carId";
                    $updateCarParams = array('userId' => $userId, 'carId' =>
$carId);
                    $this->connexion->executeQuery($updateCarQuery,
$updateCarParams);
                    $newUserMoney = $money - $price;
                    $updateUserMoneyQuery = "UPDATE db_151.t_user SET money =
:newUserMoney WHERE pk_user = :userId";
                    $updateUserMoneyParams = array('newUserMoney' =>
$newUserMoney, 'userId' => $userId);
                    $this->connexion->executeQuery($updateUserMoneyQuery,
$updateUserMoneyParams);
                    $this->transaction->createTransactionInDB($userId, $carId, 0,
$this->connexion);
                    http_response_code(200);
                    return json_encode(
                        array(
                            'success' => true,
                            'message' => 'Voiture achetée avec succès.',
                        ),
                        JSON_UNESCAPED_UNICODE
                    );

                } catch (Exception $e) {
```

```
                    $this->connexion->rollbackTransaction();
                    http_response_code(401);
                    return json_encode(
                        array(
                            'success' => false,
                            'message' => 'Erreur lors de l\'achat de la
voiture.',
                        ),
                        JSON_UNESCAPED_UNICODE
                    );
                }
            } else {
                http_response_code(401);
                return json_encode(
                    array(
                        'success' => false,
                        'message' => 'Fonds insuffisants pour acheter la
voiture.',
                    ),
                    JSON_UNESCAPED_UNICODE
                );
            }
        } else {
            http_response_code(401);
            return json_encode(
                array(
                    'success' => false,
                    'message' => 'La voiture n\'est pas disponible à l\'achat (
rafraîchissez la page )',
                ),
                JSON_UNESCAPED_UNICODE
            );
        }
    }
```

3.2 Problèmes rencontrés

3.2.1 Problème CORS

Autoriser les requêtes provenant d'un autre domaine :

```
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Origin: http://localhost:8080');
header('Access-Control-Allow-Credentials: true');
```

3.2.2 Docker

Problèmes dans l'utilisation des commandes pour lancer les images

Problème lors de la mise en place de la configuration avec wsl, docker, vsc et github (chez moi)

3.2.3 Paramètre DB

host.docker.internal et non localhost ou 127.0.0.1

3.3 Réalisation des tests

3.3.1 Tests fonctionnels

N°	Description	Résultat obtenu
1	Ajouter un utilisateur avec des champ vides	NOK
2	Ajouter un utilisateur avec « user » et « Pa$$w0rd »	OK
3	Ajouter un utilisateur avec « user » et « Pa$$w0rd »	NOK
4	Se connecter avec l'utilisateur « user » et « Pa$$w0rd »	OK
5	Se connecter avec l'utilisateur « user » et « password »	NOK
6	Se connecter avec l'utilisateur « abc » et « Pa$$w0rd »	NOK
7	Enregistrer une nouvelle voiture avec « Audi », « R8 » et « 100000 »	OK
8	Vendre cette voiture	OK
9	Acheter cette voiture	OK
10	Déconnexion	OK

3.3.2 Tests de sécurité

N°	Description	Résultat obtenu
1	Injection SQL dans le login	NOK
2	Injection SQL dans register	NOK
3	Injection SQL dans la création de voiture	NOK
4	Injection JS dans le login	NOK
5	Injection JS dans register	NOK
6	Injection JS dans la création de voiture	NOK
7	Injection HTML dans le login	NOK
8	Injection HTML dans register	NOK
9	Injection HTML dans la création de voiture	NOK
10	Injection CSS dans le login	NOK
11	Injection CSS dans register	NOK
12	Injection CSS dans la création de voiture	NOK
13	Requêtes via Postman	NOK

3.4 Hébergement

https://jaquierl.emf-informatique.ch/151/client/index.html

4 Conclusion

Je ne suis habituellement pas très familier avec les modules de développement, mais j'ai beaucoup apprécié réaliser ce projet. Le planning étant assez serré, j'ai du travailler en grande partie chez moi mais je pense avoir réussi a atteindre les critères du module.

Au niveau du fonctionnement du module, il m'a fallu un peu de temps pour m'habituer au nouveaux outils tel que docker et WSL, mais une fois pris en main, il est très agréable de les utiliser pour réaliser les tests en local.