151-Documentation Projet

Rapport personnel

Date de création : 09.02.2024
Version 2 du 03.03.2024

Ilhan Kaynak

E — M — F
Ecole des Métiers
Berufsfachschule
Fribourg
Informatique

Module du :
02.02.2024 au
08.03.2024

Table des matières

1 Introduction

Voici les objectifs de module :

1	Analyser les exigences d'une application Web et de la base de données, respectivement des éléments de données à lier, définir et documenter la technique de liaison.
2	Identifier les informations importantes de protection et de sécurité en tenant compte de la protection des données, et définir les mesures.
3	Réaliser l'intégration de l'application Web avec la base de données, respectivement aux éléments de données, en prêtant attention aux transactions, à la protection et la sécurité des données.
4	Mettre en œuvre les souhaits de modifications conformément au déroulement prescrit des modifications.
5	Définir et exécuter la procédure de test et de remise, la documenter dans un procès-verbal de tests. Si nécessaire, entreprendre les corrections.

2 Analyse

2.1 Objectifs projet

2.1.1 Consignes du projet

1. Le projet se fait individuellement.
2. Le projet devra être publié sur serveur emf-informatique.
3. Le projet doit comporter du côté client au moins deux zones différentes (visiteur et utilisateur connecté).
4. L'application cliente HTML (JS-AJAX-JQuery) et celle serveur PHPo doivent être réalisées en Objet.
5. Les données seront enregistrées dans une base de données mariaDB.
6. La réalisation du projet doit être documentée. La documentation (exempte de fautes d'orthographe) devra contenir tous les éléments demandés dans le RP.
7. Il faut réaliser le diagramme d'activité + le diagramme de séquences système + le diagramme de séquence interactions + la descente de code pour le login
8. Il faut réaliser le diagramme d'activité + le diagramme de séquences système + le diagramme de séquence interactions + la descente de code pour une fonctionnalité complète (hors login)

2.1.2 L'organisation, le fonctionnement et la structure du projet

1. Côté client :
   - Il y a une séparation des fichiers images, html, js, css
   - Il ne doit y avoir de scripts js dans les fichiers html
   - Le projet fonctionne parfaitement avec tous les navigateurs
   - L'application cliente doit s'initier par un login-password
2. Côté serveur :
   - A part les fichiers appelés par le client, les autres PHP sont des objets
   - Une séparation doit être faite entre les contrôleurs et les worker
   - Le serveur doit être appelé avec les bons paramètres (GET, POST, DELETE, PUT) et doit retourner les bons codes de réussite ou d'erreur

- Le serveur doit retourner des données au format XML ou JSON
- Le mot de passe doit être enregistré hacher dans la base de données
- L'information que l'utilisateur est bien logué doit être sauvée dans la session de l'utilisateur
- Le projet doit être robuste face aux injections HTML, JS et SQL

3. La qualité du code :
   - Tous les fichiers JS devront respecter le standard de documentation de JSDoc (http://usejsdoc.org/)
   - Toutes les classes PHPo devront respecter le standard de documentation de PHPDoc (http://www.phpdoc.org).

2.2 Présentation projet (TravelFinder)

2.2.1 Objectifs

TravelFinder est une plateforme en ligne conçue pour simplifier la recherche et la réservation d'hôtels dans diverses destinations. Le site offre aux utilisateurs la possibilité de rechercher des hôtels disponibles dans un lieu spécifique de leur choix, puis de réserver leur séjour en ligne.

2.2.2 Fonctionnalité

TravelFinder vise à simplifier le processus de recherche et de réservation d'hôtels en offrant une plateforme conviviale et intuitive. Les principaux objectifs du projet incluent :

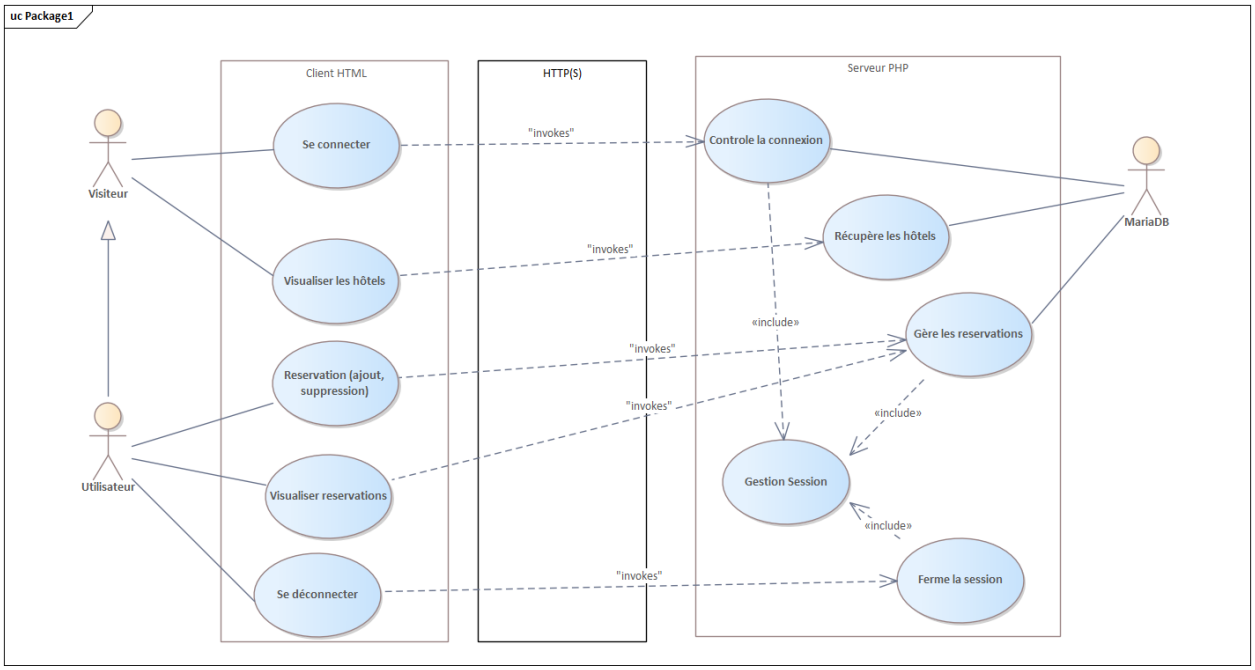
- Offrir une expérience utilisateur fluide et agréable lors de la recherche et de la réservation d'hôtels.
- Fournir des informations détaillées et précises sur chaque hôtel répertorié pour aider les utilisateurs à prendre des décisions informées.

2.2.3 Qui peut faire quoi

Le visiteur : il peut visuliser les hotels mais ne peux pas reserver, il peut se connecter.

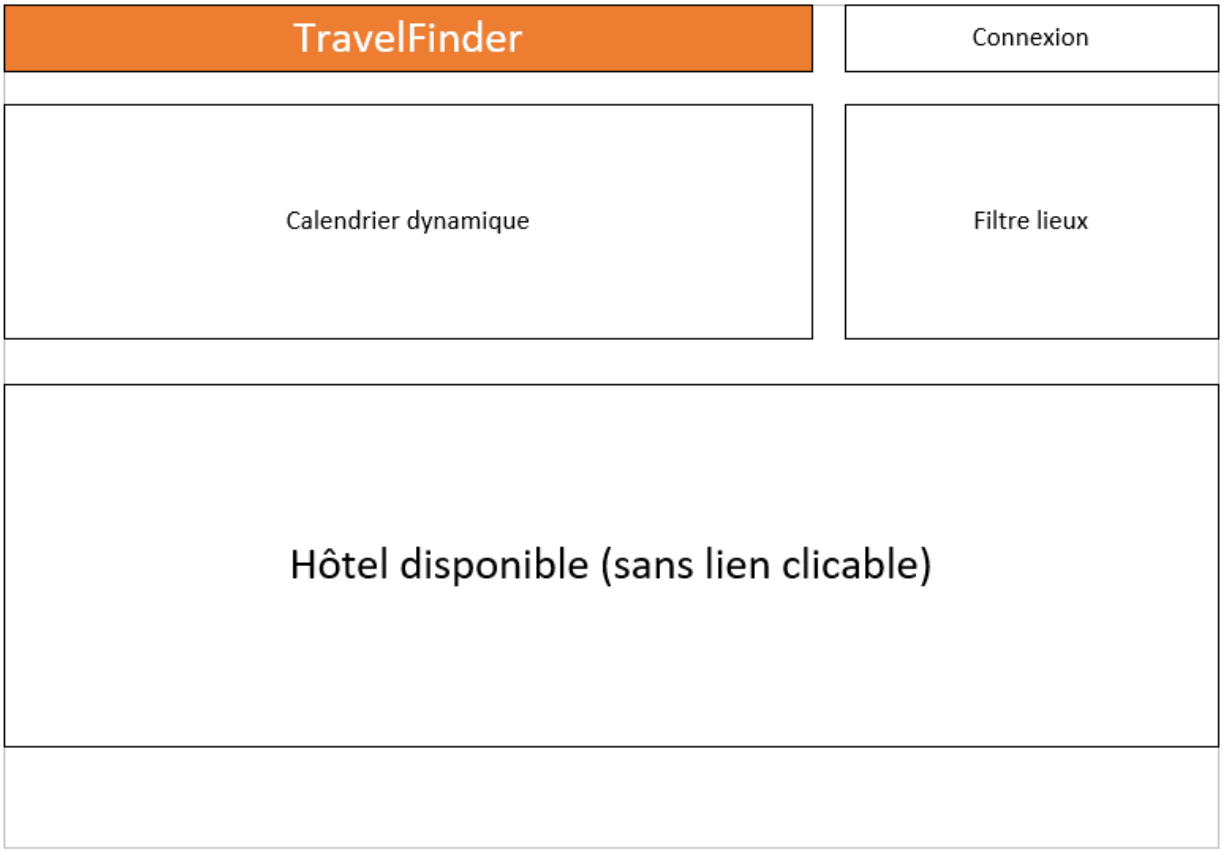
L'utilisateur : il peut visualiser et reserver les hotels.

2.3 Use cases



2.4 Maquettes

2.4.1 Vue visiteur

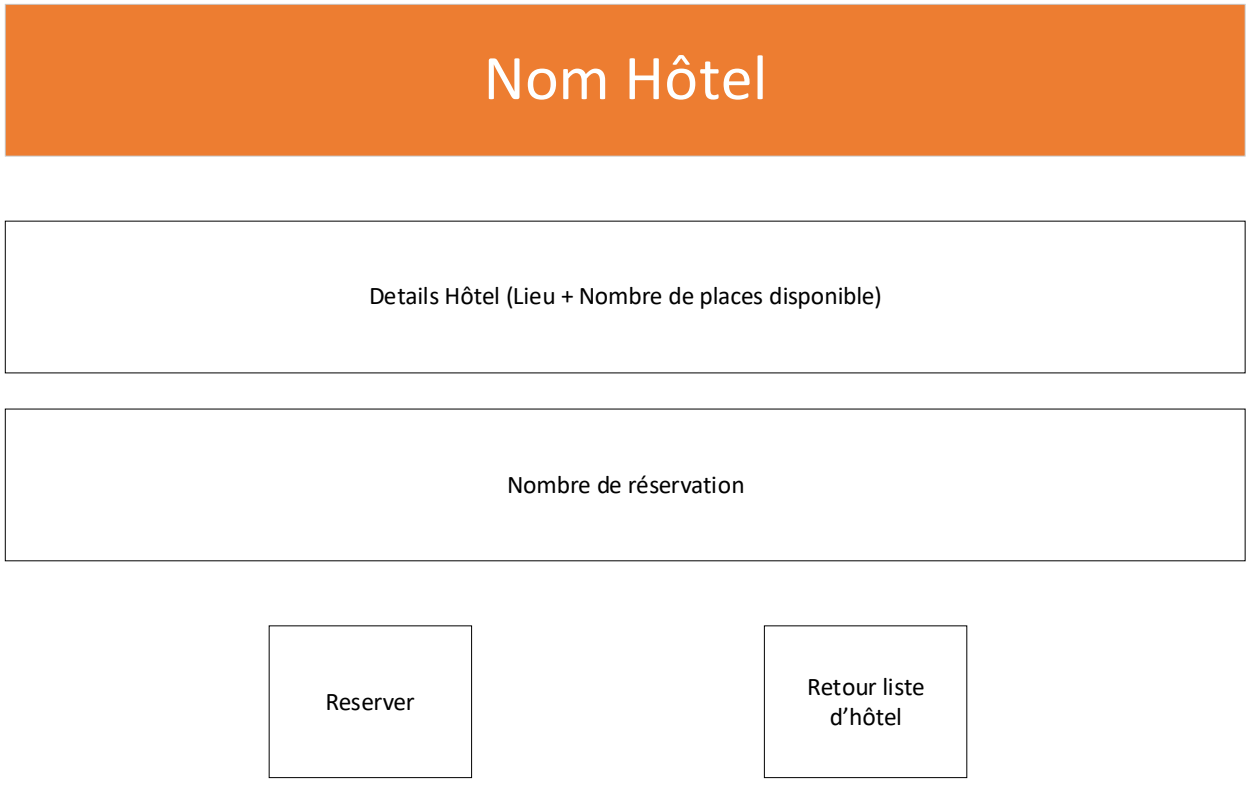


2.4.2 Vue connexion

2.4.3 Vue inscription

2.4.4 Vue utilisateur

2.4.5 Vue réservation



2.4.6 Vue mes réservations

TravelFinder

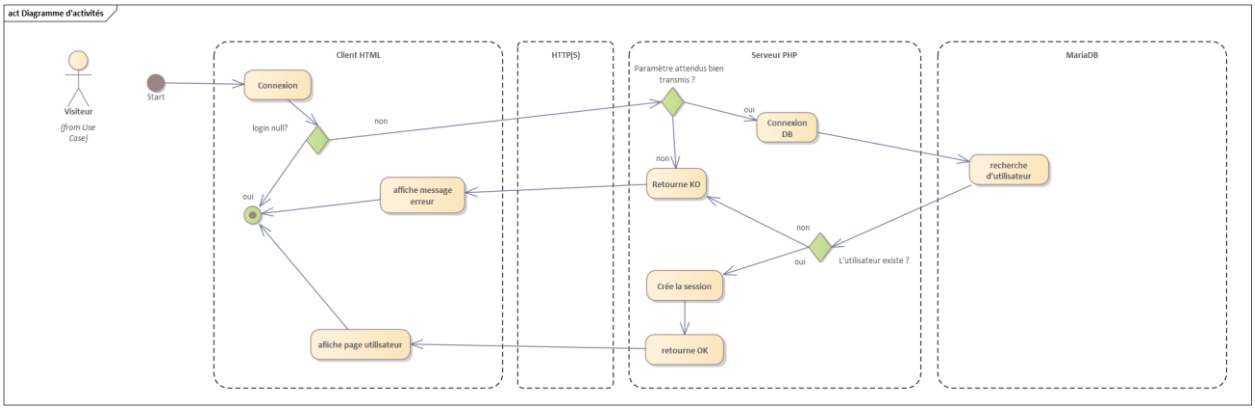
Mes reservations

Retour liste hotel

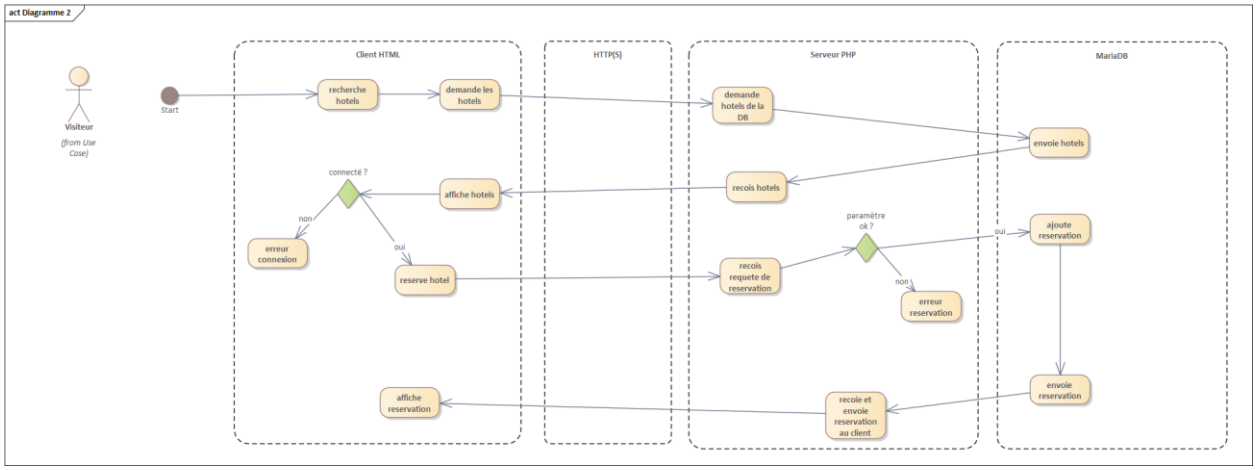
Supprimer reservation

2.5 Diagramme activités

2.5.1 Activités login

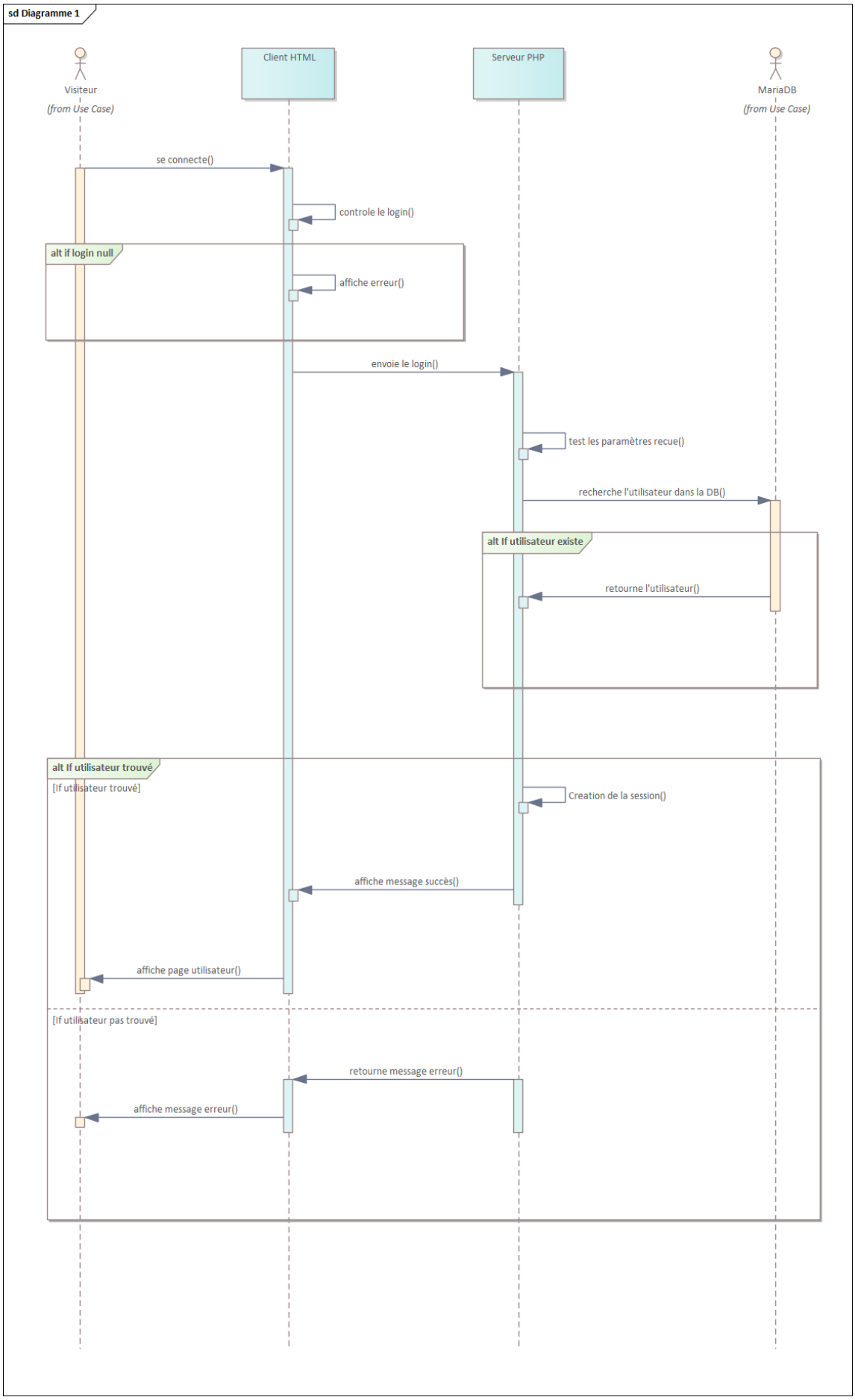


2.5.2 Activités réservation

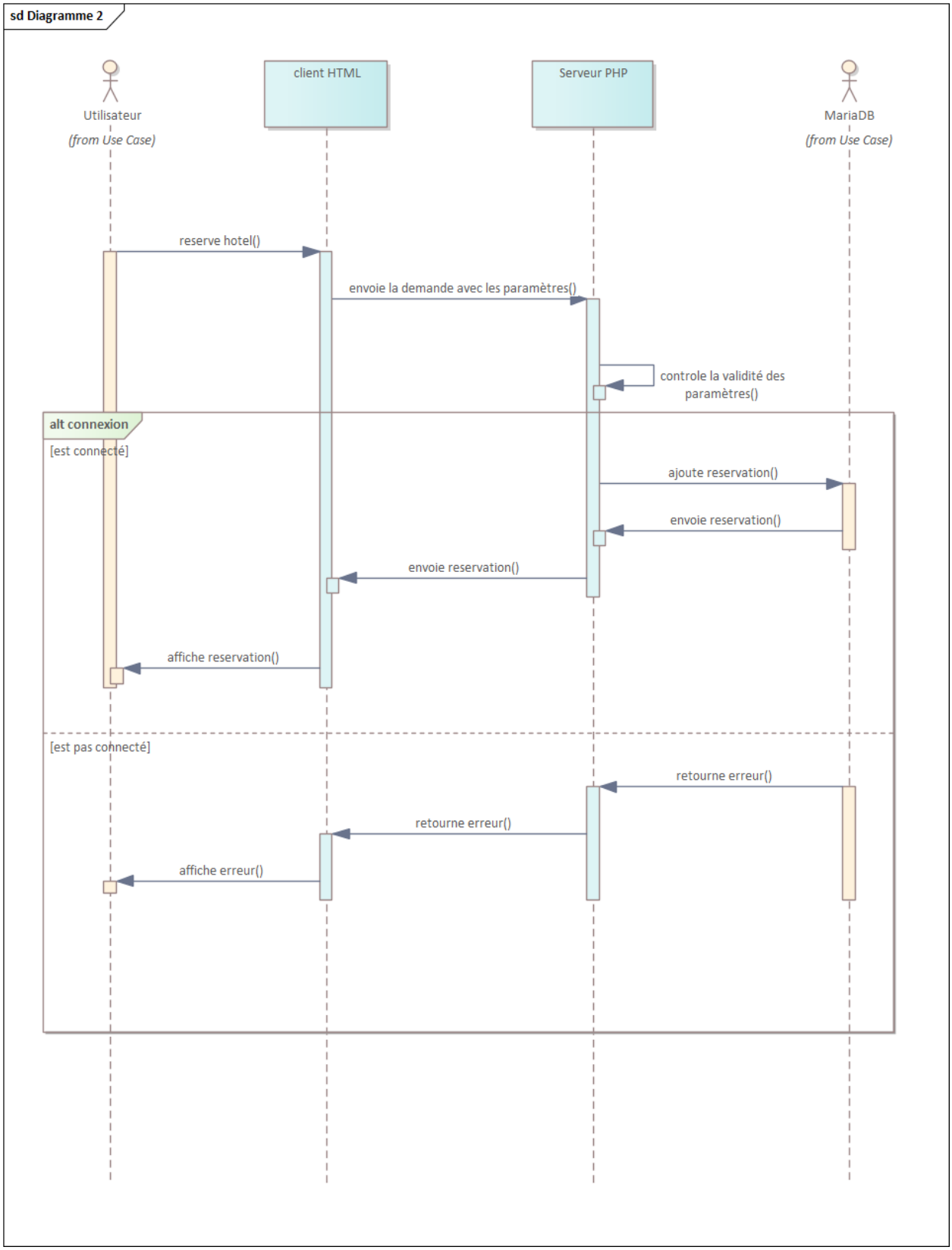


2.6 Diagrammes séquence

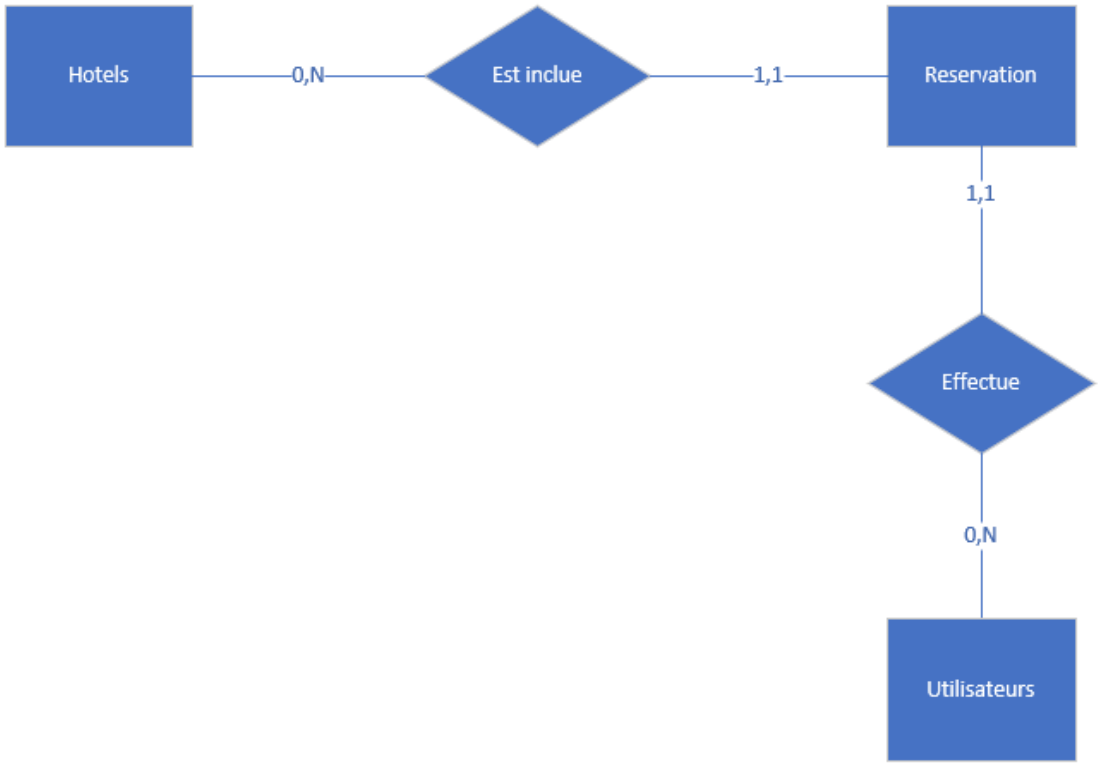
2.6.1 Séquence login


sd Diagramme 1
Visiteur
(from Use Case)
Client HTML
Serveur PHP
MariaDB
(from Use Case)
se connecte()
controle le login()
alt if login null
affiche erreur()
envoie le login()
test les paramètres recue()
recherche l'utilisateur dans la DB()
alt If utilisateur existe
retourne l'utilisateur()
alt If utilisateur trouvé
[If utilisateur trouvé]
Creation de la session()
affiche message succès()
affiche page utilisateur()
[If utilisateur pas trouvé]
retourne message erreur()
affiche message erreur()

2.6.2 Séquence réservation



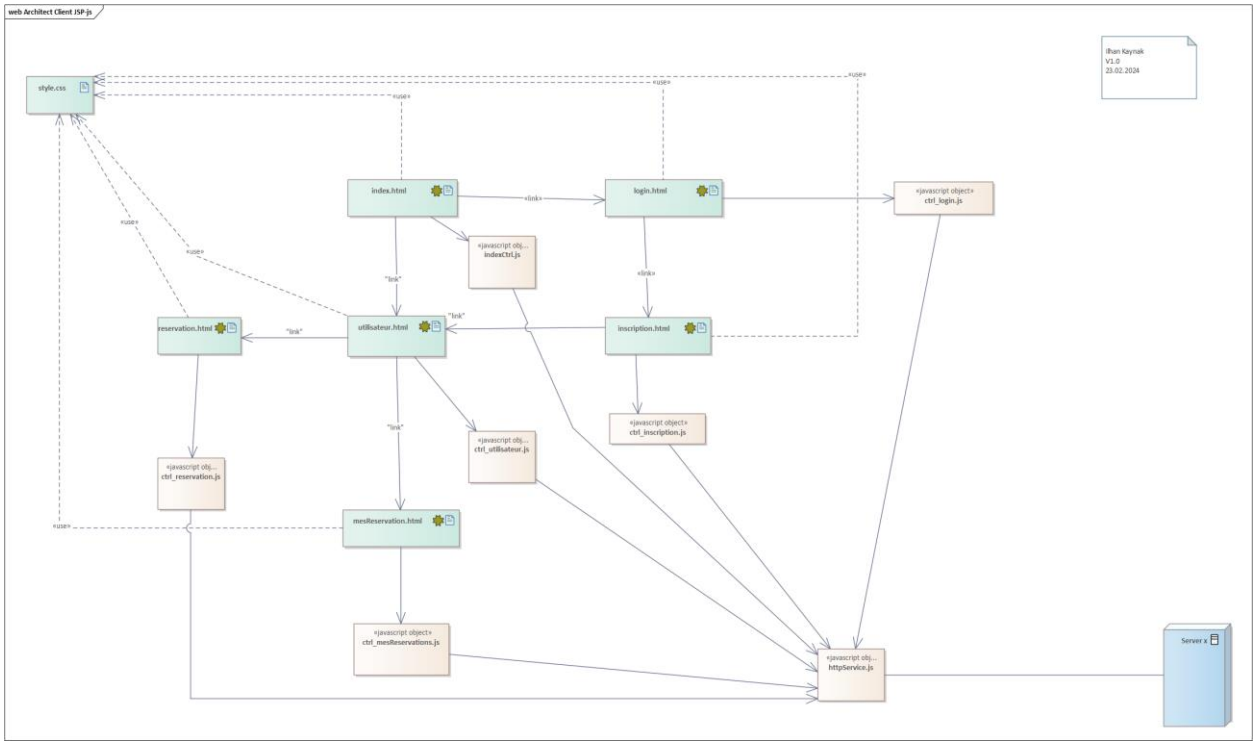
2.7 Schéma ER



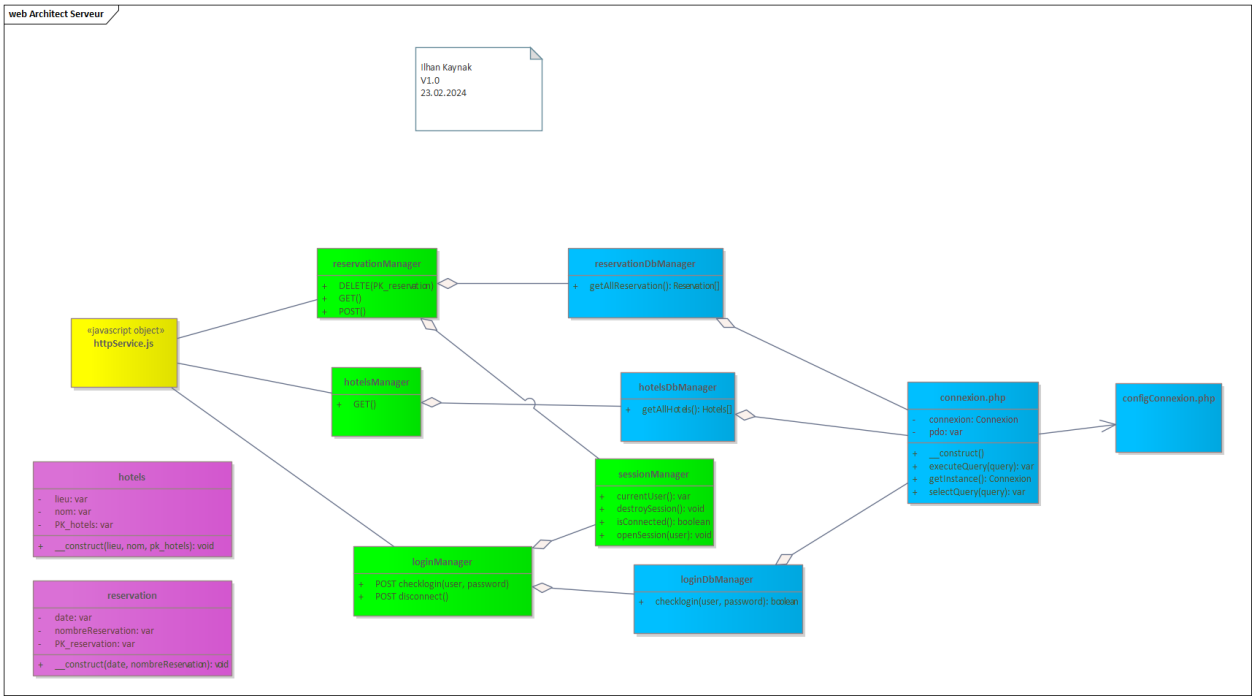
3 Conception

3.1 Diagramme de classe

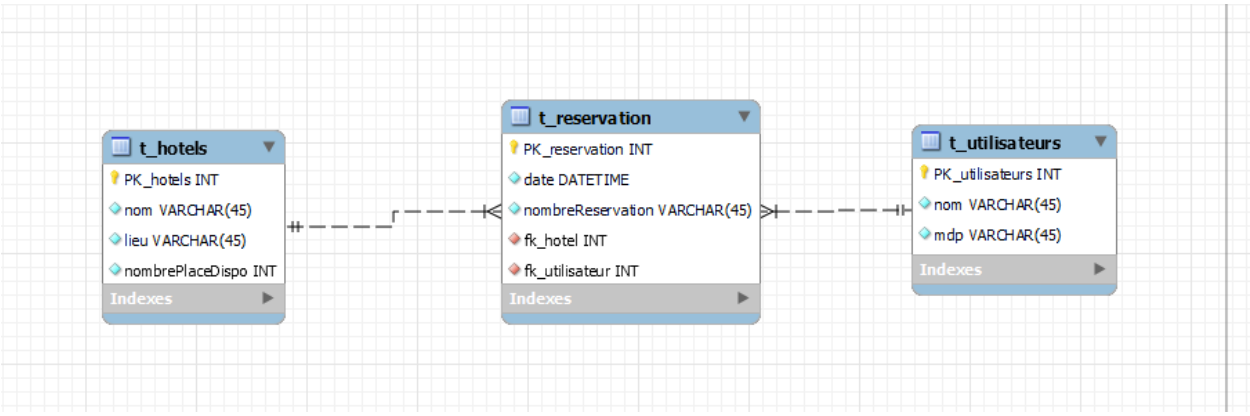
3.1.1 Client



3.1.2 Serveur

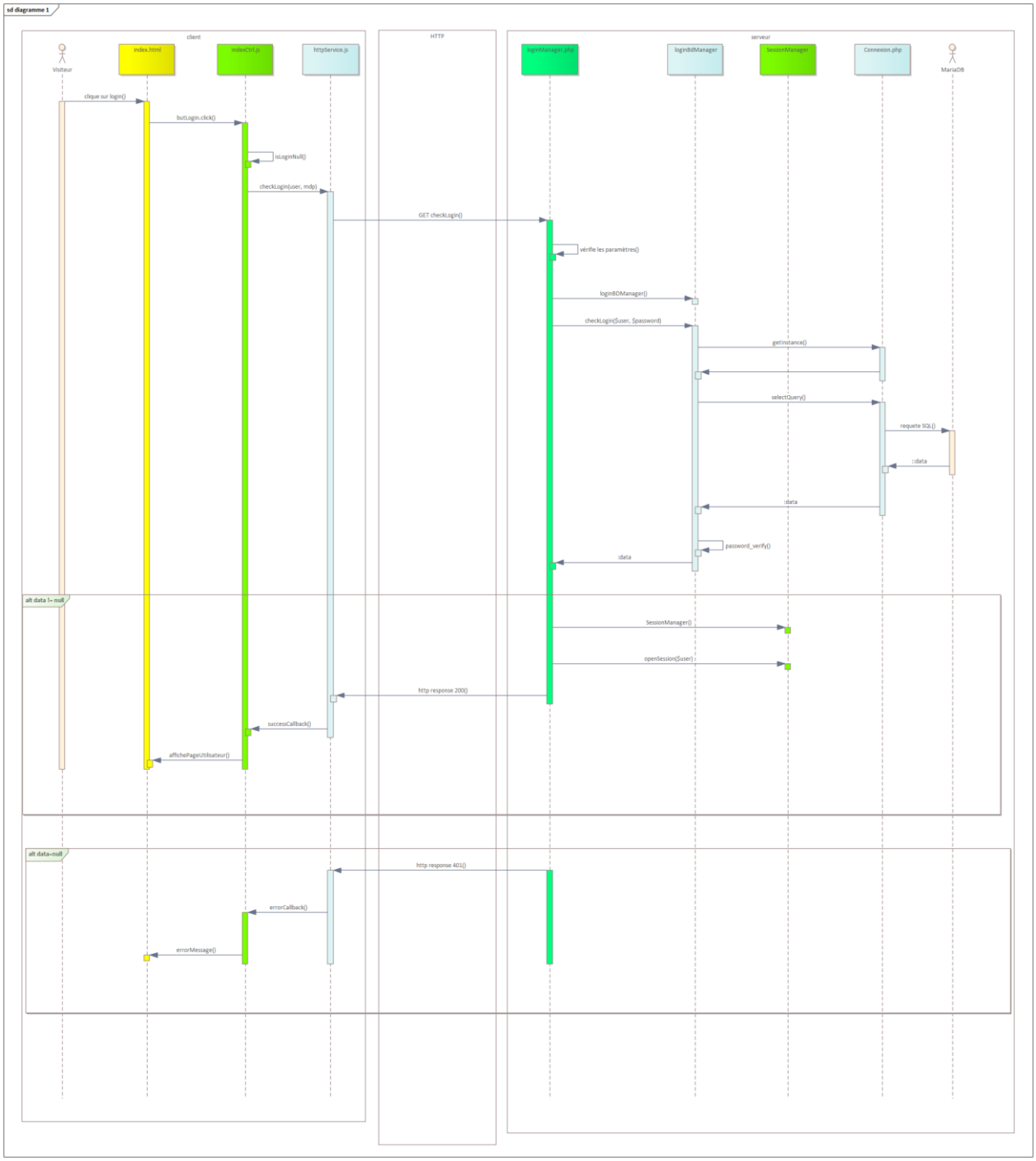


3.2 Schéma relationnel

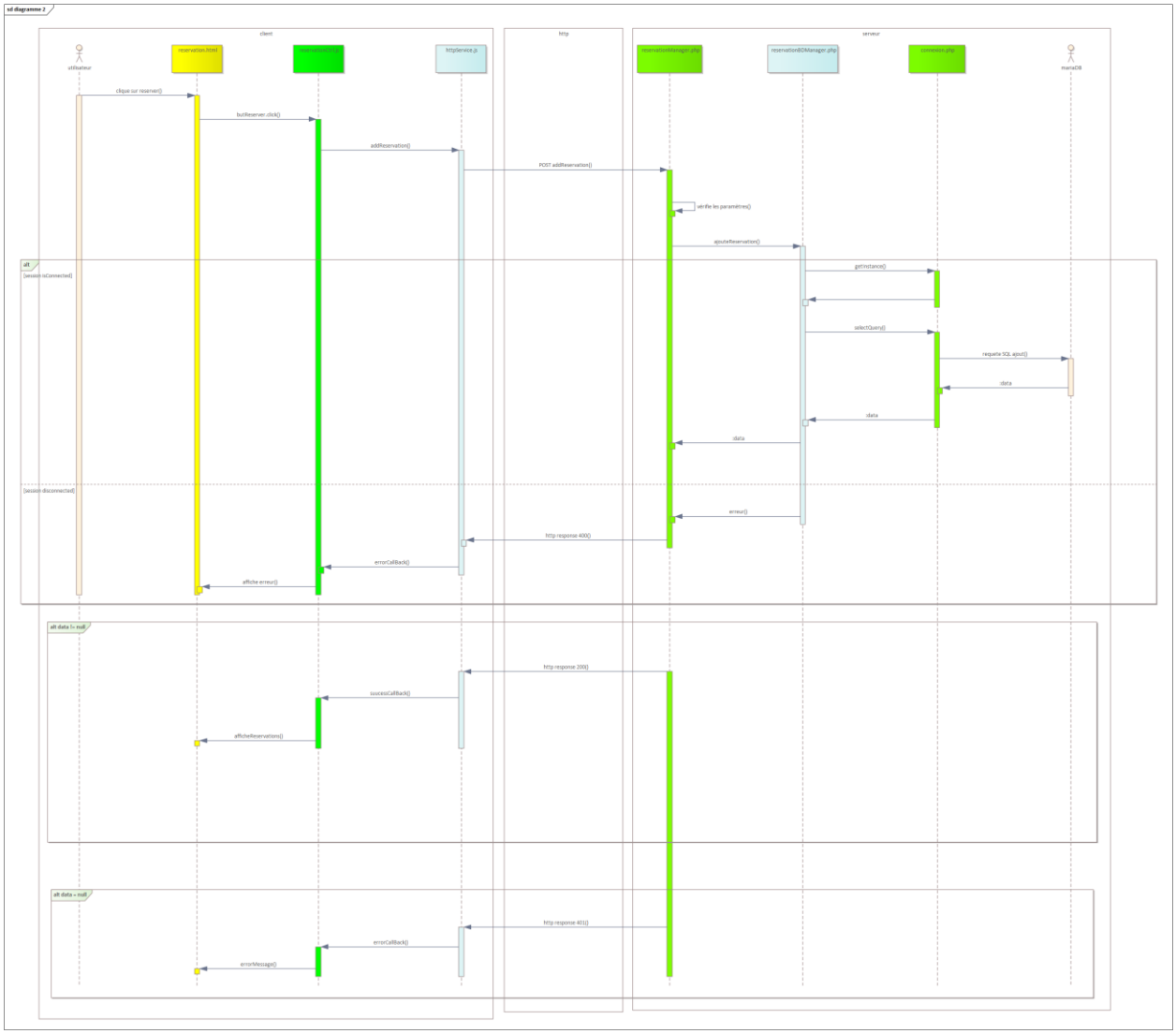


3.3 Diagramme séquence interactions

3.3.1 Login



3.3.2 Réservation



3.4 Conception de tests

Test	Résultat attendu	Résultat obtenue
Appuie sur le bouton connexion	Page de connexion s'ouvre et l'utilisateur peut soit se connecter soit s'inscrire.	
Le visiteur appuie sur un hôtel afin de le réserver	Rien de spécial ne se produit il doit se connecter.	
L'utilisateur appuis sur un hôtel afin de le réserver	Page de réservation s'ouvre, il doit entrer les paramètres désirés afin de réserver.	
L'utilisateur supprime une réservation	La réservation est supprimée.	
Le visiteur et l'utilisateur pourront filtrer les hôtels afin d'avoir ceux voulue. (Exemple Lieu = Londres)	Les hôtels filtrés s'affichent.	

L'utilisateur pourra se déconnecter	La page visiteur s'ouvre et le bouton de connexion réapparait.	

4 Implémentation

4.1 Descente de code

4.1.1 Wrk

Dans cette méthode j'effectue la requête me permettant d'insérer les paramètres données dans ma table t_reservation. Si l'action c'est bien effectué la méthode retourne true sinon false.

```
    public function addReservation($dateReservation, $nombreReservation,
$fk_hotel, $fk_utilisateur)
    {
        $requete = "INSERT INTO t_reservation (dateReservation,
nombreReservation, fk_hotels, fk_utilisateur) VALUES (?, ?, ?, ?)";


        $result = $this->connexion->executeQuery($requete,
[$dateReservation, $nombreReservation, $fk_hotel, $fk_utilisateur]);


        if ($result) {
            return true;
        } else {
            return false;
        }
    }
```

4.1.2 Ctrl

Dans cette méthode je commence par vérifier les paramètres, qu'ils ne soient pas vides. Ensuite j'appelle la méthode effectuer dans mon Wrk et j'insère les paramètres demandés. Si la méthode m'a retourné true alors je retourne la réponse http 200 sinon 401. Si les paramètres sont vides alors je retourne le code http 400.

```
    public function addReservation($date, $nombreReservation, $fk_hotel,
$fk_utilisateur)
    {
        if (!empty($date) && !empty($nombreReservation) && !empty($fk_hotel)
&& !empty($fk_utilisateur)) {
            $result = $this->manager->addReservation($date,
$nombreReservation, $fk_hotel, $fk_utilisateur);
            if ($result) {


                http_response_code(200);
            } else {
                http_response_code(401);
            }
```

```
        } else {
            http_response_code(400);
        }
    }
```

4.1.3 Server.php

Dans mon serveur.php je fais en sorte que si une session est start alors l'utilisateur peut effectuer cette action d'ajouter une réservation.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if ($_POST['action'] == "addReservation") {
        if ($sessionCtrl->get('nom')) {
            $dateReservation = $_POST['dateReservation'];
            $nombreReservation = $_POST['nombreReservation'];
            $fk_hotel = $_POST['fk_hotel'];
            $fk_utilisateur = $_POST['fk_utilisateur'];
            $reservationCtrl->addReservation($dateReservation,
$nombreReservation, $fk_hotel, $fk_utilisateur);
        }
    }
}
```

4.2 Problèmes rencontrés

J'ai eu des problèmes durant la création de ma DB. Quand j'ai créé ma DB depuis le modèle que j'avais effectué, j'avais oublié de changer le nom de ma DB donc c'étais rester mydb par défaut. Et quand je voulais insérer des données dans ma DB depuis MySQL Workbench cela me retourner une erreur. Malheureusement je n'ai pas pris de capture d'écran de celle-ci.

Un autre problème étais également en lien avec la création de la DB. Quand j'ai créé la table t_utilisateur je n'ai pas pris en compte la longueur du mot de passe haché. J'avais donc mis un nombre de 40 caractères mais le mot de passe haché en demandé plus. Donc durant mes tests avec POSTMAN j'ai eu quelques soucis avec cela.

4.3 Tests fonctionnels

Je n'ai pas encore implémenté le côté client donc les tests ne sont pas encore testables.

Test	Résultat attendu	Résultat obtenue
Appuie sur le bouton connexion	Page de connexion s'ouvre et l'utilisateur peut soit se connecter soit s'inscrire.	-
Le visiteur appuie sur un hôtel afin de le réserver	Rien de spécial ne se produit il doit se connecter.	-
L'utilisateur appuis sur un hôtel afin de le réserver	Page de réservation s'ouvre, il doit entrer les paramètres désirés afin de réserver.	-
L'utilisateur supprime une réservation	La réservation est supprimée.	-

Le visiteur et l'utilisateur pourront filtrer les hôtels afin d'avoir ceux voulue. (Exemple Lieu = Londres)	Les hôtels filtrés s'affichent.	-
L'utilisateur pourra se déconnecter	La page visiteur s'ouvre et le bouton de connexion réapparait.	-

4.4 Hébergement

L'hébergement de mon site se trouve sous l'URL ci-dessous. Je n'ai pas encore implémenté le côté client dans sur le visuel rien de fonctionne mais si nous effectuons les testes depuis POSTMAN les fonctionnalités sont opérationnels.

https://kaynaki.emf-informatique.ch

5 Conclusion

5.1 Ce que j'ai appris

Durant ce module j'ai appris à coder un backend avec le langage de programmation php.

5.2 Ce que j'ai aimé

Ce module m'a beaucoup plu, j'ai aimé travailler pour mon site et j'y ai pris du plaisir. Rien à voir avec le module mais j'ai aussi aimé les pauses personnelles. Les théories du cours n'étais pas ennuyeux, c'était claire et concis.

5.3 Ce que j'ai moins aimé

Le fait de ne pas avoir assez de temps pour le module mais passé plus de temps sur l'analyse et la conception étais assez embêtant.