151 - Le Classique 8

Documentation de Projet

Date de création : 28.01.2025  
Version 3 05.05.2025

Baptiste Schneider

|  |  |  |
| --- | --- | --- |
|  |  | Module du 28.01.2025 au 25.02.2025 |

Table des matières

[1 Introduction 5](#_Toc191414938)

[1.1 Description 5](#_Toc191414939)

[1.2 Planning 5](#_Toc191414940)

[2 Analyse 6](#_Toc191414941)

[2.1 Uses Case 6](#_Toc191414942)

[2.2 Maquettes 7](#_Toc191414943)

[2.3 Diagramme activité 8](#_Toc191414944)

[2.4 Diagramme de séquences systèmes 9](#_Toc191414945)

[2.5 Schéma ER 10](#_Toc191414946)

[3 Conception 11](#_Toc191414947)

[3.1 Diagrammes de classe 11](#_Toc191414948)

[3.1.1 Client 11](#_Toc191414949)

[3.1.2 Serveur 11](#_Toc191414950)

[3.2 Schéma relationnel 12](#_Toc191414951)

[3.3 Diagramme séquence interactions 13](#_Toc191414952)

[3.4 Test fonctionnels 15](#_Toc191414953)

[3.4.1 Conception des tests 15](#_Toc191414954)

[4 Implémentation 16](#_Toc191414955)

[4.1 Descente de code 16](#_Toc191414956)

[4.1.1 En-têtes http 16](#_Toc191414957)

[4.1.2 Démarrage de la session 16](#_Toc191414958)

[4.1.3 Récupération des paramètres 16](#_Toc191414959)

[4.1.4 Traitement des requêtes POST 16](#_Toc191414960)

[4.1.5 Actions disponibles 16](#_Toc191414961)

[4.1.5.1 Se connecter 16](#_Toc191414962)

[4.1.5.2 Réserver une table 17](#_Toc191414963)

[4.1.5.3 Vérifier une session utilisateur 18](#_Toc191414964)

[4.1.5.4 Se déconnecter 18](#_Toc191414965)

[4.2 Problèmes rencontrés 18](#_Toc191414966)

[4.2.1 Cache 18](#_Toc191414967)

[4.2.2 Mauvais départ 18](#_Toc191414968)

[4.3 Tests fonctionnels 18](#_Toc191414969)

[4.4 Hébergement 19](#_Toc191414970)

[5 Synthèse 20](#_Toc191414971)

[5.1 Présentation réalisation 20](#_Toc191414972)

[5.1.1 Login 20](#_Toc191414973)

[5.1.2 Home 20](#_Toc191414974)

[5.1.3 Contact 20](#_Toc191414975)

[5.2 Différences entre planning et réalisation 21](#_Toc191414976)

[5.3 Différence entre la conception et la réalisation 21](#_Toc191414977)

[5.4 Conclusion 22](#_Toc191414978)

# Introduction

## Description

Durant ce projet, je vais créer une application web de gestion pour une salle de billard appelée Le Classique 8. Cette application centralisera des fonctionnalités telles que la réservation des tables ou encore la visualisation des tables.

Elle sera composée d’un client web pour l’interface utilisateur, d’un serveur pour gérer la logique métier et les échanges, et d’une base de données pour stocker les informations essentielles.

L’objectif est de concevoir une solution intuitive, performante et évolutive, adaptée aux besoins spécifiques de la salle.

## Planning

Voici le planning pour mon projet :



# Analyse

## Uses Case

Ce diagramme use-case illustre l'architecture client-serveur de mon projet. Il représente les interactions entre deux types d’utilisateurs, Visiteur et Administrateur, avec un client HTML et un serveur PHP via HTTP(S). Le Visiteur peut se connecter et visualiser les tables, tandis que l’Administrateur peut également réserver une table et se déconnecter.

Chaque action déclenche une requête HTTP(S) vers le serveur PHP, qui gère les processus backend comme le contrôle du login, la réservation, la déconnexion et la fermeture de session, en interagissant avec une base de données (BD).

Ce schéma met en évidence la répartition des responsabilités entre le client et le serveur ainsi que le flux des interactions entre les différentes entités du système.

Une image contenant texte, capture d’écran, diagramme, cercle

Le contenu généré par l’IA peut être incorrect.

## Maquettes

Les maquettes présentent une interface utilisateur pour une plateforme de réservation de tables de billard. L'application comprend une page de connexion permettant aux utilisateurs de s'authentifier ou d'accéder directement à l'accueil via un bouton de passage.

La page d'accueil affiche les tables disponibles avec une interface de réservation où les joueurs peuvent saisir leur nom et réserver une session. Une section de navigation permet d'accéder à une page de contact fournissant des informations pour joindre le service client via téléphone, réseaux sociaux ou email.

L’ensemble des pages est structuré avec une barre de navigation facilitant le passage entre les sections principales.

Une image contenant texte, capture d’écran, diagramme, affichage

Description générée automatiquement

## Diagramme activité

Ce diagramme d'activité illustre le processus de réservation d'une table par un utilisateur via un client HTML interagissant avec un serveur PHP et une base de données MariaDB.

L'utilisateur clique sur "Réserver" ; si une erreur survient, un message s'affiche, sinon les informations de l'utilisateur sont présentées.

Les paramètres sont transmis au serveur PHP, qui vérifie leur présence ainsi que l'authentification de l'utilisateur avant de se connecter à la base de données.

Si la table est pleine, la réservation échoue, sinon elle est validée. MariaDB gère la recherche et l'enregistrement des réservations, garantissant une gestion efficace des demandes.

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

## Diagramme de séquences systèmes

Ce diagramme de séquence décrit le processus de réservation d'une table. Il illustre les interactions entre un visiteur, un client HTML, un serveur PHP et une base de données MariaDB.

Le processus commence par la demande de réservation, suivie de la validation des paramètres. Si ces derniers sont valides, le système vérifie si l'utilisateur est connecté. Si l'utilisateur est connecté, une autre vérification s'assure que le nombre de joueurs est inférieur à 4 pour autoriser la réservation.

En cas de succès, la demande est transmise au serveur PHP, qui interagit avec la base de données pour finaliser la réservation.

Si l'une des conditions (paramètres invalides, utilisateur non connecté ou nombre de joueurs insuffisant) n'est pas remplie, la réservation est refusée avec un message de non-validation (NOK).

Ce diagramme met en évidence la logique de validation et les échanges entre les différents composants du système pour assurer un processus de réservation structuré et fiable.

Une image contenant texte, diagramme, Parallèle, nombre

Le contenu généré par l’IA peut être incorrect.

## Schéma ER

Le diagramme entité-association présenté illustre la gestion des réservations de tables par des clients. Il comprend trois entités principales : Client, Table et Localité. Un client peut effectuer plusieurs réservations de tables, et une table peut être réservée par plusieurs clients

Chaque client est également associé à une localité via la relation habite, signifiant qu'un client réside dans une seule localité.

Ce diagramme structure les interactions essentielles pour la gestion des réservations et la localisation des clients, facilitant ainsi l'organisation et l'optimisation du service.



# Conception

## Diagrammes de classe

### Client

Le diagramme de classe illustre l'organisation des interfaces et leurs relations au sein de l'application cliente. Il se compose de plusieurs interfaces : login.js, IndexCtrl.js, et httpService.js, chacune définissant des attributs et des méthodes spécifiques.

Les connexions entre ces interfaces sont représentées par des associations et des relations d'utilisation. Les attributs sont indiqués avec leur type de données, tandis que les méthodes affichent leurs paramètres et types de retour. L'ensemble du diagramme structure l'interaction entre les différentes composantes du projet en mettant en évidence les dépendances et la répartition des responsabilités.

Il est important de savoir que ce diagramme de classe n’est pas une version complète de ce que mon application utilisera car je ne sais pas encore ce que j’utiliserais pour la faire fonctionner

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

### Serveur

Le diagramme de classe pour la partie serveur décrit la structure et les relations entre plusieurs interfaces. Il inclut serveur.php, Ctrl.php et WrkDB.php, chacune possédant ses propres attributs et méthodes.

Les relations entre ces interfaces sont représentées par des associations et une relation d'utilisation, indiquant les dépendances entre les composants. Les attributs sont précisés avec leurs types de données, et les méthodes affichent leurs paramètres et types de retour. Ce diagramme met en évidence l'organisation modulaire du système, définissant clairement les interactions entre les différentes parties du projet.

Il est important de savoir que ce diagramme de classe n’est pas une version complète de ce que mon application utilisera car je ne sais pas encore ce que j’utiliserais pour la faire fonctionner

Une image contenant texte, capture d’écran, diagramme, Police

Le contenu généré par l’IA peut être incorrect.

## Schéma relationnel

Ce diagramme représente la structure d'une base de données relationnelle pour la gestion des clients et de leurs associations avec des tables.

La table T\_Client stocke les informations des clients, y compris leur nom, prénom, email, mot de passe et une référence à leur localité via FK\_localite, qui est une clé étrangère pointant vers la table T\_Localite. Cette dernière contient des informations sur les localités, incluant un identifiant unique (PK\_Localite), le nom de la localité et son code postal.

La table de relation TR\_Client\_Table établit une relation entre T\_Client et T\_Table, permettant d'associer plusieurs clients à plusieurs tables. Enfin, T\_Table contient les tables disponibles, identifiées par PK\_Table et un numéro unique.

Ce modèle assure une gestion structurée des clients, de leur localisation et des tables auxquelles ils sont associés.

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

## Diagramme séquence interactions

Ce diagramme de séquence interaction illustre le processus de réservation d’une table dans un système client-serveur. Il met en évidence les interactions entre un visiteur, les différents composants du système et la base de données MariaDB.

Le processus débute avec l’appel à la fonction reserveTable(), suivi d’une vérification des paramètres et de l’authentification de l’utilisateur. Si les conditions sont remplies (paramètres valides, utilisateur connecté, nombre de joueurs approprié), la réservation est effectuée via une requête à la base de données. En cas de non-respect des conditions, des réponses négatives (NOK()) sont renvoyées à différentes étapes. Ce diagramme permet de comprendre la logique métier et les validations effectuées avant qu’une réservation ne soit confirmée.

Une image contenant texte, diagramme, Parallèle, Plan

Le contenu généré par l’IA peut être incorrect.

## Test fonctionnels

### Conception des tests

Ci-dessous, se trouve un tableau reprenant tous les tests que je vais effectuer afin de tester mon projet. Ils se base sur l’action réserver une table du [Uses Case](#_Uses_Case)



# Implémentation

## Descente de code

### En-têtes http

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Methods: POST, GET, PUT");

header("Access-Control-Allow-Headers: Content-Type, Authorization");

header("Access-Control-Allow-Credentials: true");

header("Content-Type: application/json");

Ces lignes définissent les en-têtes HTTP. Cela permet d'autoriser les requêtes CORS, spécifie les méthodes HTTP autorisées (POST, GET, PUT), et déclare que le contenu renvoyé est en JSON. Cela permet également de gérer les cookies avec Access-Control-Allow-Credentials: true

### Démarrage de la session

session\_start();

include\_once 'worker/Connexion.php';

La fonction session\_start() initialise une session PHP. Ensuite, le fichier Connexion.php est inclus pour gérer la connexion à la base de données, probablement en utilisant une classe Connexion.

### Récupération des paramètres

$json = file\_get\_contents('php://input');

$receivedParams = json\_decode($json, true);

Le code récupère le corps de la requête HTTP sous forme de chaîne JSON et le décode en un tableau associatif PHP grâce à json\_decode(). Cela permet de manipuler les données envoyées par le client.

### Traitement des requêtes POST

if (isset($\_SERVER['REQUEST\_METHOD'])) {

    if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

        if (isset($receivedParams['action'])) {

            $db = Connexion::getInstance();

Le script vérifie si la méthode HTTP est un POST et si un paramètre action est présent dans les données reçues. Ensuite, une instance de la connexion à la base de données est obtenue via la méthode getInstance() de la classe Connexion.

### Actions disponibles

#### Se connecter

if ($receivedParams['action'] == "connect") {

if (isset($receivedParams['username']) && isset($receivedParams['password'])) {

    $username = $receivedParams['username'];

        $password = $receivedParams['password'];

        $query = "SELECT NomUtilisateur, password

FROM T\_Client WHERE NomUtilisateur = :username";

        $params = [":username" => $username];

        $user = $db->selectSingleQuery($query, $params);

      if ($user) {

        if (password\_verify($password, $user['password'])) {

$\_SESSION['loggued'] = $username;

               $\_SESSION['status'] = 'loggued';

               http\_response\_code(200);

               echo json\_encode(["result" => true, "message" => "Connecté"]);

           } else {

            http\_response\_code(400);

               echo json\_encode(["result" => false, "error" => "Mot de passe incorrect"]);

           }

       } else {

           http\_response\_code(404);

           echo json\_encode(["result" => false, "error" => "Utilisateur non trouvé"]);

       }

   } else {

       http\_response\_code(400);

       echo json\_encode(["result" => false, "error" => "Paramètres manquants"]);

   }

}

* L'action connect permet à un utilisateur de se connecter. Elle vérifie que l'username et le mot de passe sont fournis.
* Une requête SQL est exécutée pour récupérer l'utilisateur correspondant au nom d'utilisateur.
* Si l'utilisateur existe, le mot de passe est vérifié avec password\_verify().
* Si la connexion est réussie, l'utilisateur est enregistré dans la session. Sinon, des messages d'erreur sont renvoyés.

#### Réserver une table

if ($receivedParams["action"] == "reserve") {

if (isset($receivedParams['tableNumber'])) {

$tableNumber = $receivedParams['tableNumber'];

        if (isset($\_SESSION['loggued'])) {

        $username = $\_SESSION['loggued'];

            $query = "SELECT COUNT(\*) as total FROM TR\_Client\_Table

WHERE FK\_Table = :tableNumber";

            $params = [":tableNumber" => $tableNumber];

            $result = $db->selectSingleQuery($query, $params);

            if ($result && $result['total'] < 4) {

            $insertQuery = "INSERT INTO TR\_Client\_Table (FK\_Client, FK\_Table)

                                VALUES ((SELECT PK\_Table FROM T\_Table

WHERE Numero = :tableNumber), :username)";

                $insertParams = [":tableNumber" => $tableNumber, ":username" => $username];

                if ($db->executeQuery($insertQuery, $insertParams)) {

                    http\_response\_code(200);

                    echo json\_encode(["result" => true, "message" => "Table réservée"]);

                } else {

                    http\_response\_code(500);

                    echo json\_encode(["result" => false, "error" => "Erreur lors de la réservation"]);

                }

            } else {

                http\_response\_code(409);

                echo json\_encode(["result" => false, "error" => "Table complète"]);

            }

        } else {

            http\_response\_code(401);

            echo json\_encode(["result" => false, "error" => "Utilisateur non connecté"]);

        }

} else {

    http\_response\_code(400);

    echo json\_encode(["result" => false, "error" => "Paramètres manquants"]);

   }

}

* L'action reserve permet de réserver une table. Le script vérifie si l'utilisateur est connecté (via la session).
* Une requête SQL est exécutée pour vérifier le nombre actuel de réservations sur la table. Si la table n'est pas complète (moins de 4 réservations), une nouvelle entrée est insérée dans la base de données.
* Si l'insertion est réussie, un message de confirmation est renvoyé. Sinon, des erreurs appropriées sont renvoyées.

#### Vérifier une session utilisateur

if ($receivedParams['action'] == "check\_session") {

if (isset($\_SESSION['loggued']) && $\_SESSION['status'] == 'loggued') {

        http\_response\_code(200);

        echo json\_encode(["result" => true]);

    } else {

        http\_response\_code(401);

        echo json\_encode(["result" => false]);

    }

}

L'action check\_session permet de vérifier si l'utilisateur est connecté en vérifiant la session. Si l'utilisateur est connecté, un code de réponse 200 est renvoyé, sinon un code 401 (non autorisé) est renvoyé.

#### Se déconnecter

if ($receivedParams['action'] == "disconnect") {

session\_unset();

    http\_response\_code(200);

    echo json\_encode(["result" => true]);

}

L'action disconnect détruit la session de l'utilisateur avec session\_unset(), ce qui le déconnecte. Un message de succès est renvoyé avec un code de réponse 200.

## Problèmes rencontrés

### Cache

Durant le projet, j'ai rencontré à plusieurs reprises des problèmes avec le cache de mon navigateur. En effet, surtout lors de la partie de réalisation du login, le cache jouait un rôle essentiel dans le fonctionnement de mon app.

Le cache de mon navigateur n'ayant pas été vidé, il ne prenait pas forcément en compte mes modifications, ce qui m'a fait perdre énormément de temps.

### Mauvais départ

Durant le projet, j'ai rencontré un problème qui venait de ma visualisation du code que je devrais produire. Je suis à plusieurs reprises parti dans une direction complètement absurde, qui m'a par conséquent fait perdre énormément de temps

## Tests fonctionnels

Voici mon protocole de tests fonctionnels du point [Conception des tests](#_Conception_des_tests) réalisé. Tous les tests ont été réalisé en local



## Hébergement

Pour l’hébergement de mon site web, j’ai premièrement téléversé tout mes fichiers sur cPanel dans le dossier 151 que j’ai créé. J’ai par la suite créé la base de données distante ainsi que la connexion vers cette base de données avec un utilisateur pour pouvoir la manager. J’ai terminé en insérant les données dans ma base de données

# Synthèse

## Présentation réalisation

### Login

Voici finalement ma page de login, sur cette page nous pouvons nous connecter avec nos identifiant ou bien passer si nous ne souhaitons pas se connecter :

Une image contenant texte, capture d’écran, logo, conception

Le contenu généré par l’IA peut être incorrect.

### Home

Voici la page home de mon site. Sur cette page, nous pouvons nous connecter/déconnecter. Nous pouvons aussi réserver une table avec notre nom d’utilisateur et visualiser les différentes pages

Une image contenant capture d’écran, logiciel, Logiciel multimédia, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

### Contact

Voici la page de contact, sur cette page nous pouvons simplement naviguer à travers les différentes pages du site

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.

## Différences entre planning et réalisation

Voici mon planning final. On peut voir qu’au début du module je le suivais plutôt bien mais qu’à partir du lundi après-midi du bloc 2 ça à pas mal changé. J’ai donc essayé de rattraper mon retard, on peut le voir dans les blocs 3-4-5



## Différence entre la conception et la réalisation

Pour faire une comparaison entre la conception et la réalisation, nous pouvons constater que mes fichiers ne sont pas organisés pareil, pas forcément les mêmes nom ou moins de fichier utiliser. De plus, j’utilise beaucoup plus de méthodes que je n’avais par forcément pensé utiliser lors de la conception.

De plus la logique pour réserver une table n’est pas la même que ce que j’avais imaginer lors de la réalisation.

Pour conclure, on peut voir un certain nombre de différences entre la conception et la réalisation

## Conclusion

Pour conclure, j’aurais pu produire un travail dans un état bien plus avancé avec de fonctionnalité plus poussée mais je me suis perdu sur des problèmes et des mauvais choix tout au long du module.   
Je suis tout de même content du travail fourni sur cette nouvelle technologie, mais j’aurais pu faire mieux si j’avais fait la réalisation dans un autre sens et avec un peu plus de temps