

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

Rapport de Projet

Date de création : 29.01.2024
Version 2 du 05.03.2024

Simon Gendre

EMF – Fribourg / Freiburg

Ecole des Métiers / Berufsfachschule
Technique / Technik

Module du : 29.01.2024

Au : 05.03.2024

Table des matières

1. INTRODUCTION.....	3
1.1. LE PROJET	3
1.2. OBJECTIFS DU MODULE	3
2. ANALYSE	4
2.1. LE PROJET	4
2.2. PLANNING.....	4
2.2.1. <i>Use Cases</i>	5
2.3. MAQUETTES.....	6
2.3.1. <i>Écran de login</i>	6
2.3.2. <i>Mode invité</i>	6
2.3.3. <i>Mode connecté</i>	7
2.4. DIAGRAMMES D'ACTIVITÉS (UNE ACTION).....	7
2.5. DIAGRAMMES DE SÉQUENCES (UNE ACTION)	8
2.6. SCHÉMA ER.....	8
3. CONCEPTION.....	9
3.1. DIAGRAMME DE CLASSE	9
3.1.1. <i>Client</i>	9
3.1.2. <i>Serveur</i>	9
3.2. DIAGRAMME DE SÉQUENCE INTERACTIONS	9
3.3. SCHÉMA RELATIONNEL DE LA BASE DE DONNÉES.....	10
3.3.1. <i>Table des utilisateurs</i>	11
3.3.2. <i>Table des salles de tchat</i>	11
3.3.3. <i>Table des messages</i>	11
4. RÉALISATION	13
4.1. DESCENTE DE CODE (ENVOI DE MESSAGES)	13
4.1.1. <i>Client</i>	13
4.1.2. <i>Serveur</i>	15
4.2. TESTS FONCTIONNELS	19
4.3. PROBLÈMES RENCONTRÉS.....	21
4.4. HÉBERGEMENT.....	21
5. SYNTHÈSE	23
5.1. DIFFÉRENCE DE TIMING.....	23
5.1.1. <i>Planning avec les croix</i>	23
5.1.2. <i>Analyse du planning</i>	23
5.2. DIFFÉRENCE AVEC LA CONCEPTION	23
5.3. CONCLUSION	24
5.3.1. <i>Ce que je retiens de ce module</i>	24
5.3.2. <i>Amélioration / proposition</i>	24
5.3.3. <i>Mes points forts et faibles</i>	24

1. Introduction

1.1. Le Projet

Durant ce module 151, nous allons créer une application web client-serveur en PHP et JavaScript.

1.2. Objectifs du 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. Le projet

Mon projet s'intitule Semaphor et consiste en une application de messagerie en direct. Les utilisateurs peuvent lire les anciens messages et doivent se logger pour en envoyer.

Il y aura une chatroom principale et éventuellement d'autres chatroom

2.2. Planning

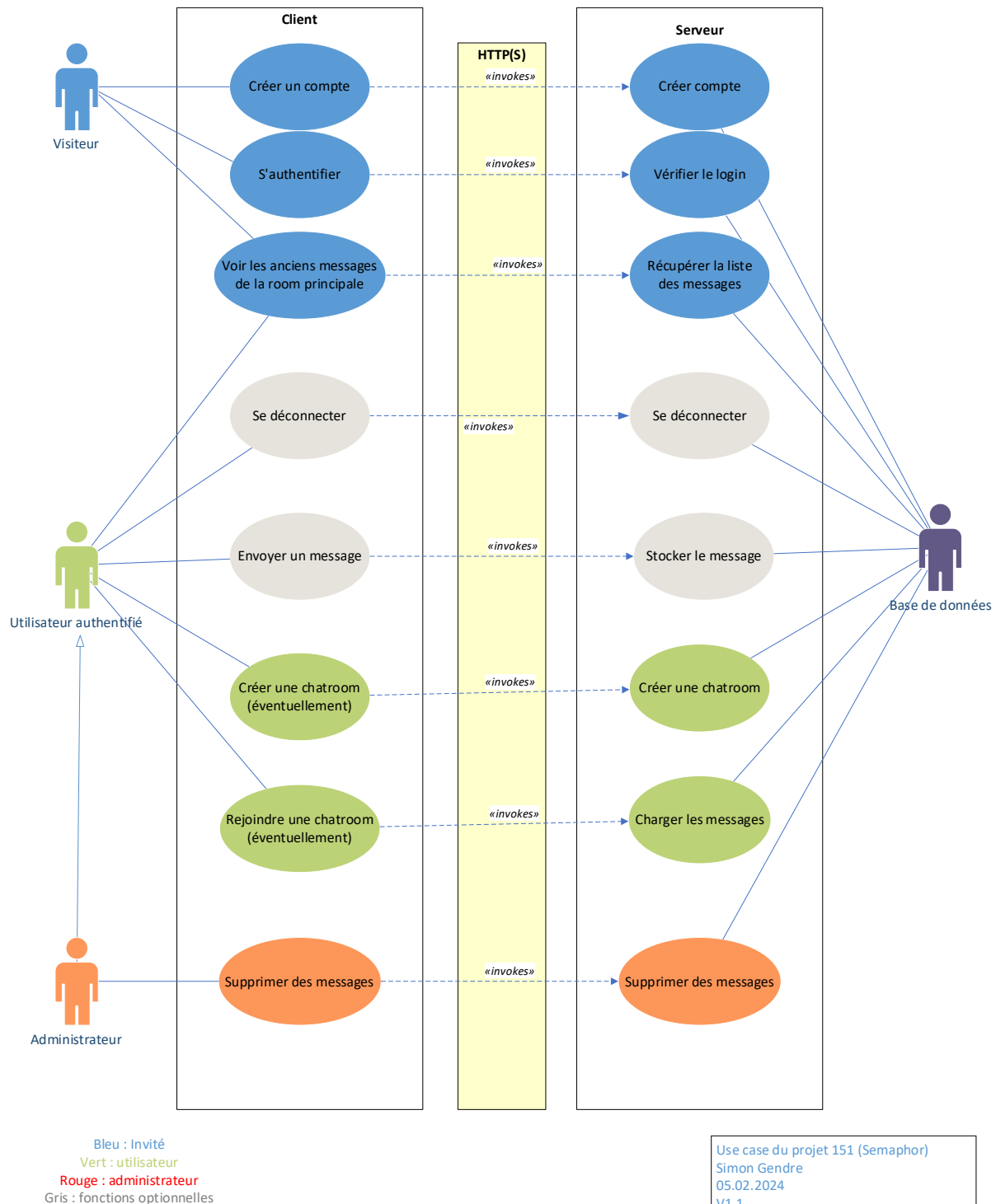
Voici le planning que j'ai estimé pour ce projet. L'analyse et la conception seront fait en semaine 1 et 2. La réalisation sera faite entre les semaines 2, 3, 4 et 5.

Tâches	29.01.2023 - MA	29.01.2023 - A-N	30.01.2023 - MA	05.02.2023 - MA	05.02.2023 - A-N	06.02.2023 - MA		19.02.2023 - MA	19.02.2023 - A-N	20.02.2023 - MA	26.02.2023 - MA	26.02.2023 - A-N	27.02.2023 - MA	04.03.2023 - MA	04.03.2023 - A-N	05.03.2023 - MA
Exercices / découverte							VACANCES DE CARNAVAL									
Analyse																
introduction																
use cases																
maquettes																
diagramme d'activités																
diagramme de séquences																
schéma ER																
Conception																
diagramme de classe serveur																
diagramme de classe client																
schéma relationnel DB																
diagramme de séquence interactions																
Réalisation																
Serveur - structure de base																
Serveur - login																
Serveur - récupérer les messages																
Serveur - envoyer des messages																
Serveur - création de comptes																
Serveur - créer et rejoindre des rooms																
Client - structure de base																
Client - login																
Client - récupérer les messages																
Client - - création de comptes																
Client - envoyer des messages																
Client - créer et rejoindre des rooms																
descente de code																
Hebergement																
Tests de l'application																

2.2.1. Use Cases

Voici les Use Cases de l'application. Il y aura trois acteurs différents : les visiteurs, les utilisateurs et les administrateurs. Voici ce qu'ils peuvent faire :

Acteur	Actions
Visiteur	Se loguer / créer un compte / voir les anciens messages de la room principale
Utilisateur	Se déconnecter / envoyer des messages / rejoindre et créer des room
Administrateur	Même chose que les utilisateur / supprimer des messages

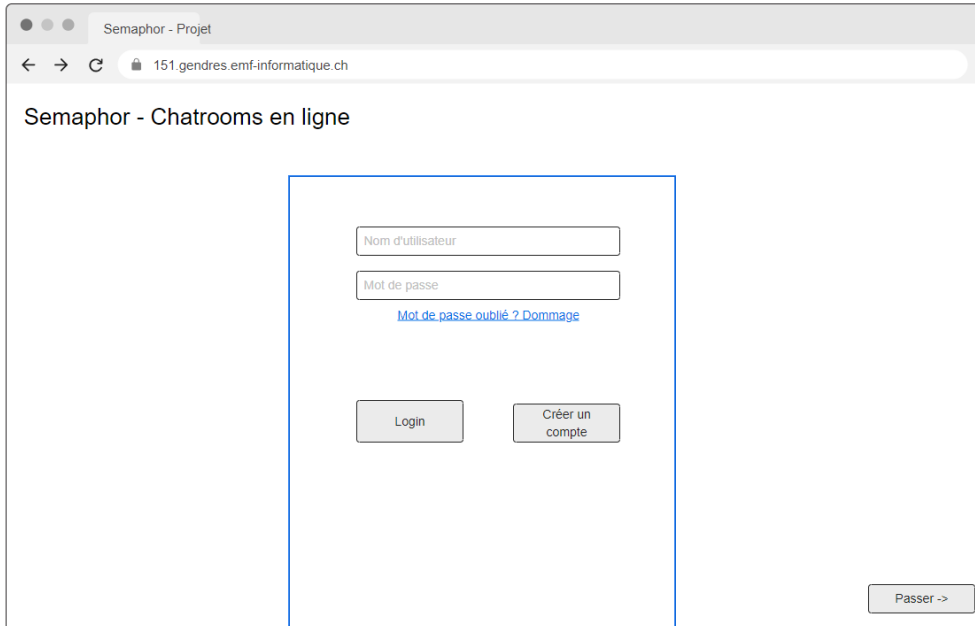


2.3. Maquettes

Voici les maquettes du site WEB. Ce ne sont que des squelettes qui permettent de se donner une idée du résultat final. Ce dernier risque d'être légèrement différent dans le design mais pas dans les options.

2.3.1. Écran de login

Une fois arrivé sur le site, les visiteurs seront accueillis par une page de login. Ils pourront, soit se connecter/créer un compte, soit accéder au site en mode invité.



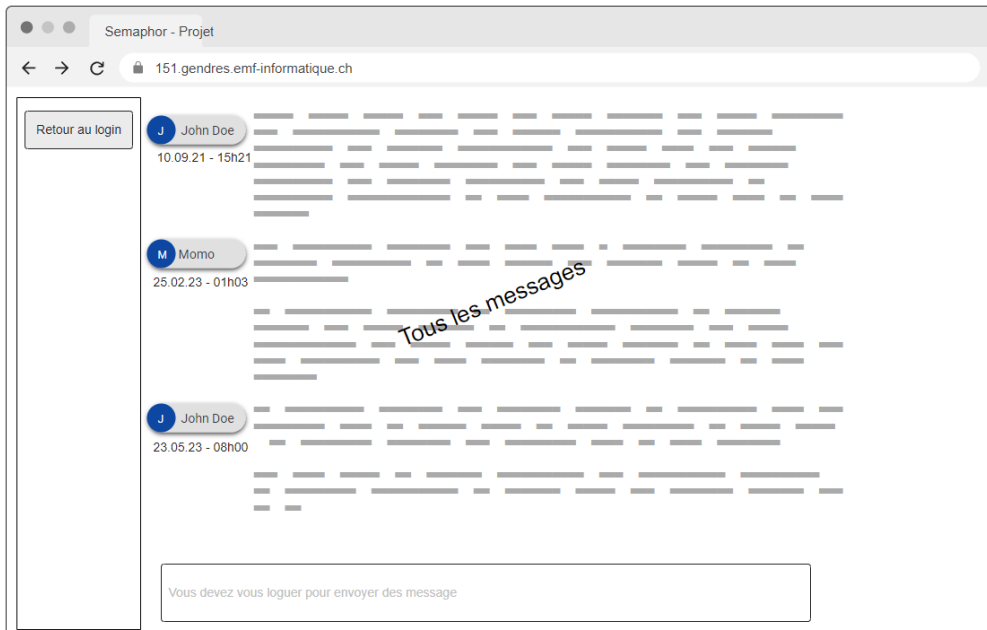
Maquette de l'écran de login. Le navigateur affiche "Semaphor - Projet" et l'URL "151.gendres.emf-informatique.ch". Le titre de la page est "Semaphor - Chatrooms en ligne". Le formulaire de login est centré et contient :

- Champ "Nom d'utilisateur"
- Champ "Mot de passe" avec un lien "Mot de passe oublié ? Dommage" en dessous.
- Deux boutons : "Login" et "Créer un compte".

En bas à droite, il y a un bouton "Passer ->".

2.3.2. Mode invité

Le mode invité permet de voir les messages envoyés dans la room principale. C'est tout.

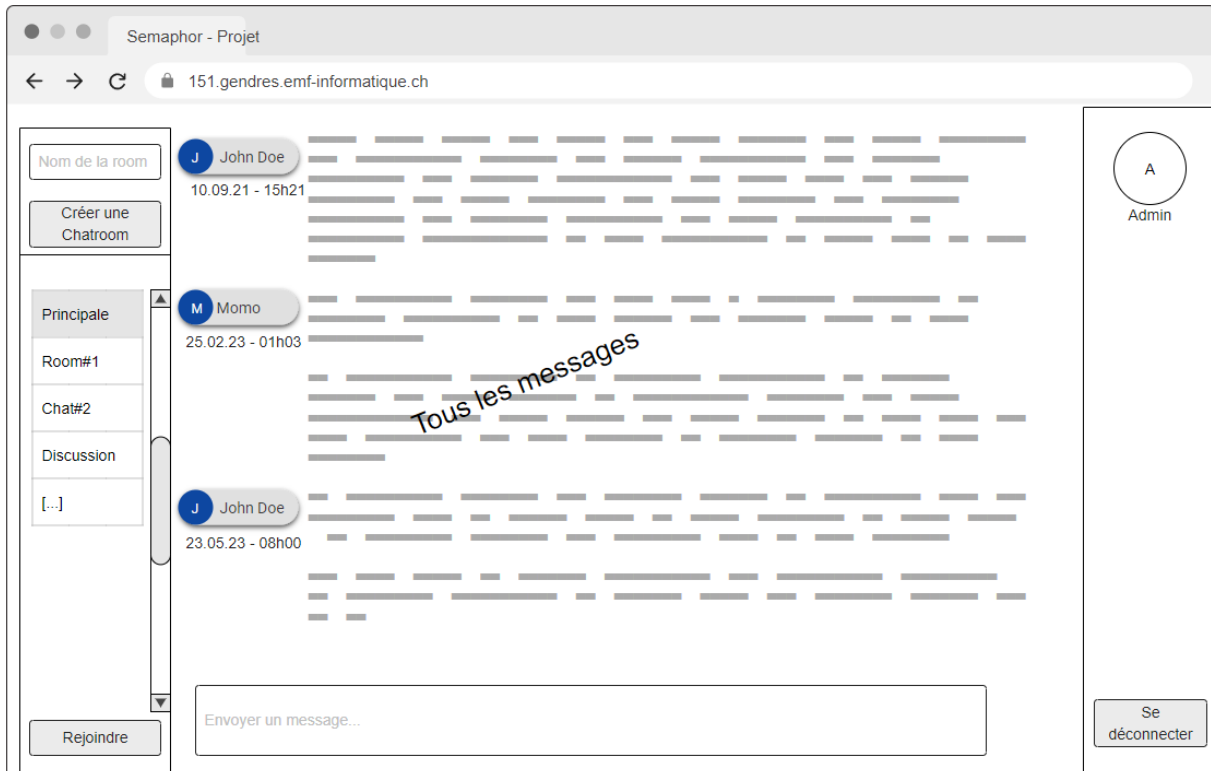


Maquette du mode invité. Le navigateur affiche "Semaphor - Projet" et l'URL "151.gendres.emf-informatique.ch". Le bouton "Retour au login" est en haut à gauche. Le chatroom principal affiche :

- Un bouton "Retour au login" en haut à gauche.
- Une liste de messages avec des avatars et des noms d'utilisateurs (John Doe, Momo) et des timestamps (10.09.21 - 15h21, 25.02.23 - 01h03, 23.05.23 - 08h00).
- Un message "Tous les messages" est superposé sur la liste.
- Un champ de saisie en bas avec le texte "Vous devez vous loguer pour envoyer des message".

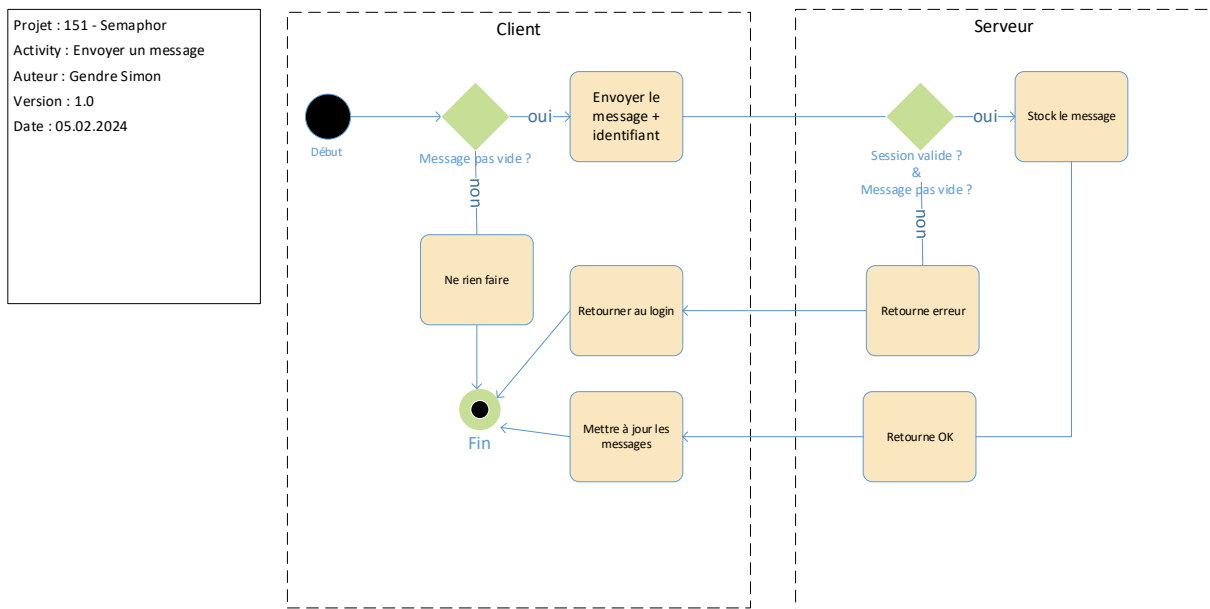
2.3.3. Mode connecté

Le mode connecté permet aussi de voir les messages et d'en envoyer mais il sera alors possible de changer de room en en créant une ou en rejoignant une.

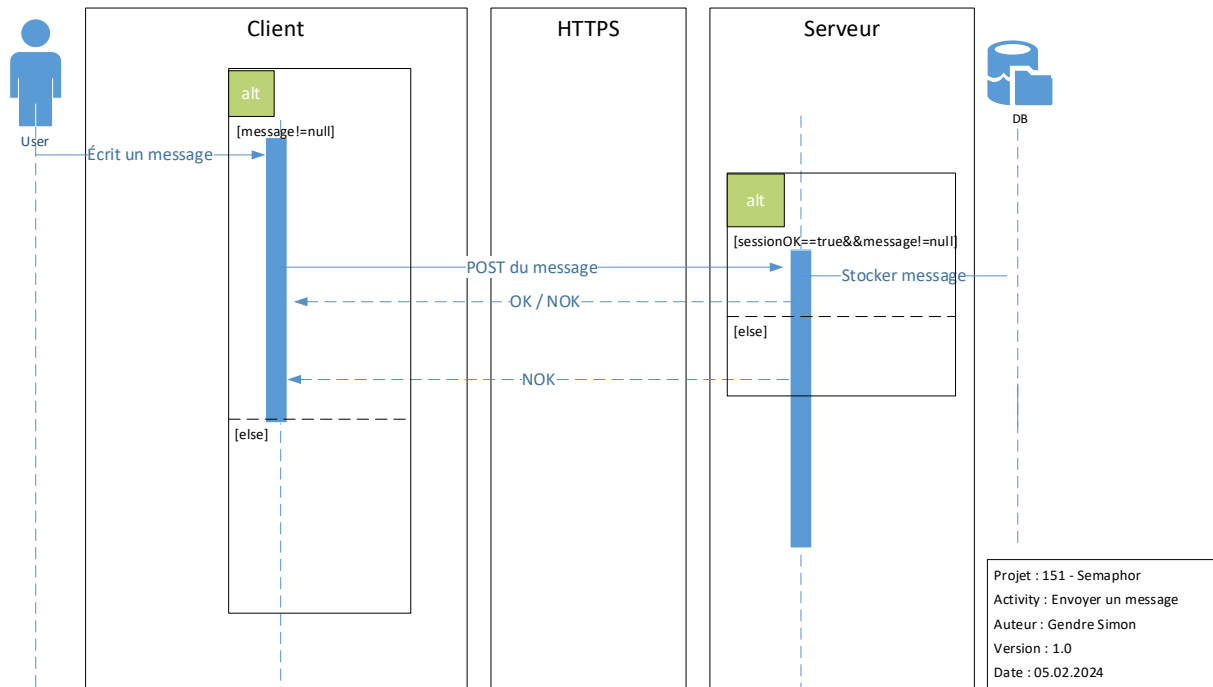


2.4. Diagrammes d'activités (une action)

Voici le diagramme d'activité pour envoyer un message.

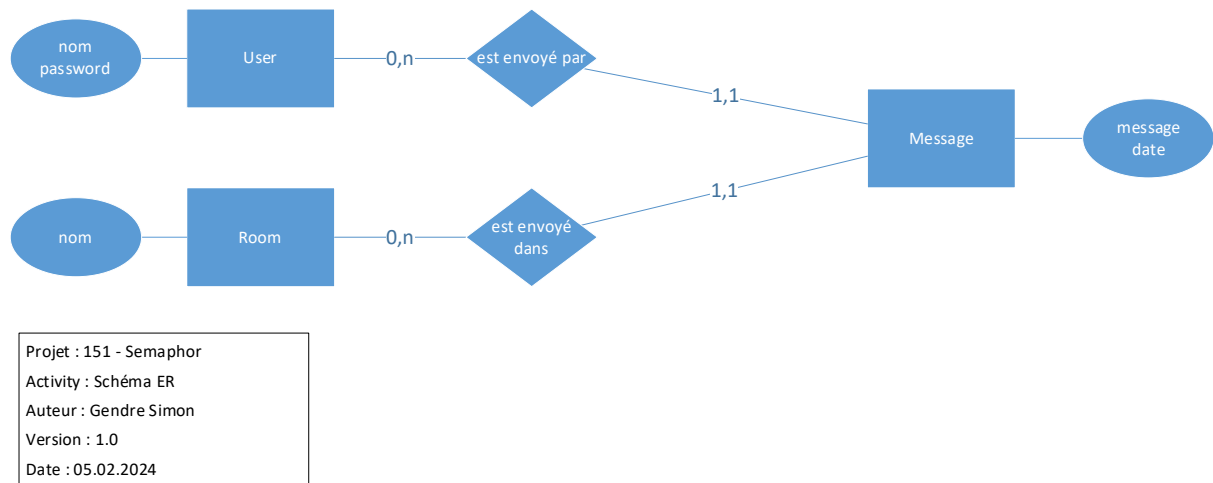


2.5. Diagrammes de séquences (une action)



2.6. Schéma ER

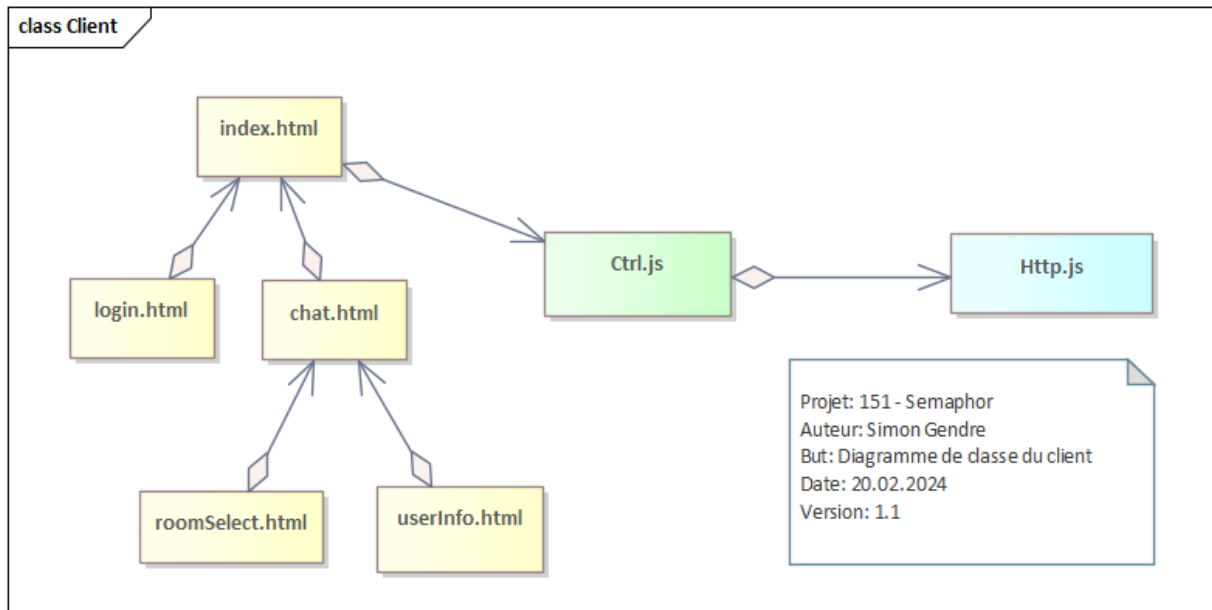
Voici à quoi ressemblera la base de données du serveur. Elle aura trois tables qui seront relié entre-elles. La table « user » contiendra les informations de login (nom et hash de mot de passe), la table room servira à séparer les messages en salles de chat différentes et pour finir, les messages contiendront un texte, une date et les informations qui les relie aux autres tables.



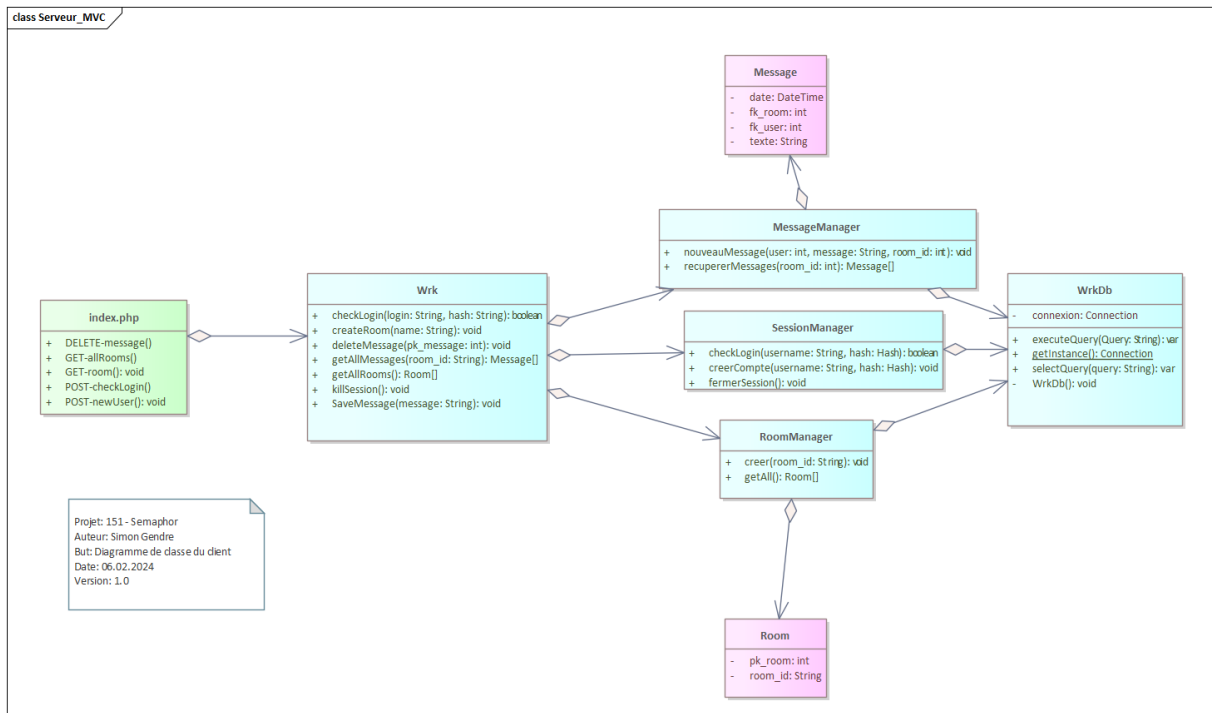
3. Conception

3.1. Diagramme de classe

3.1.1. Client

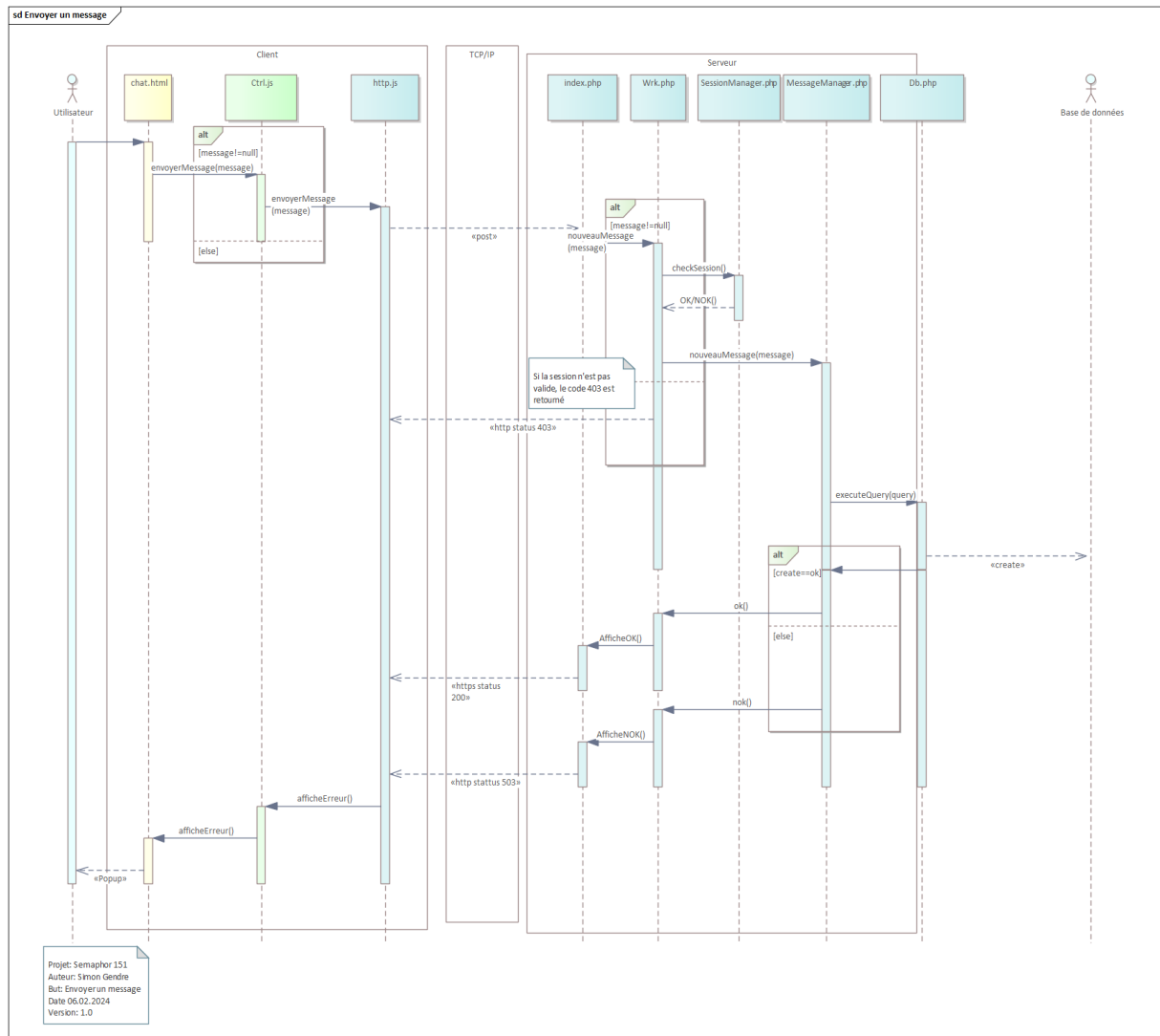


3.1.2. Serveur



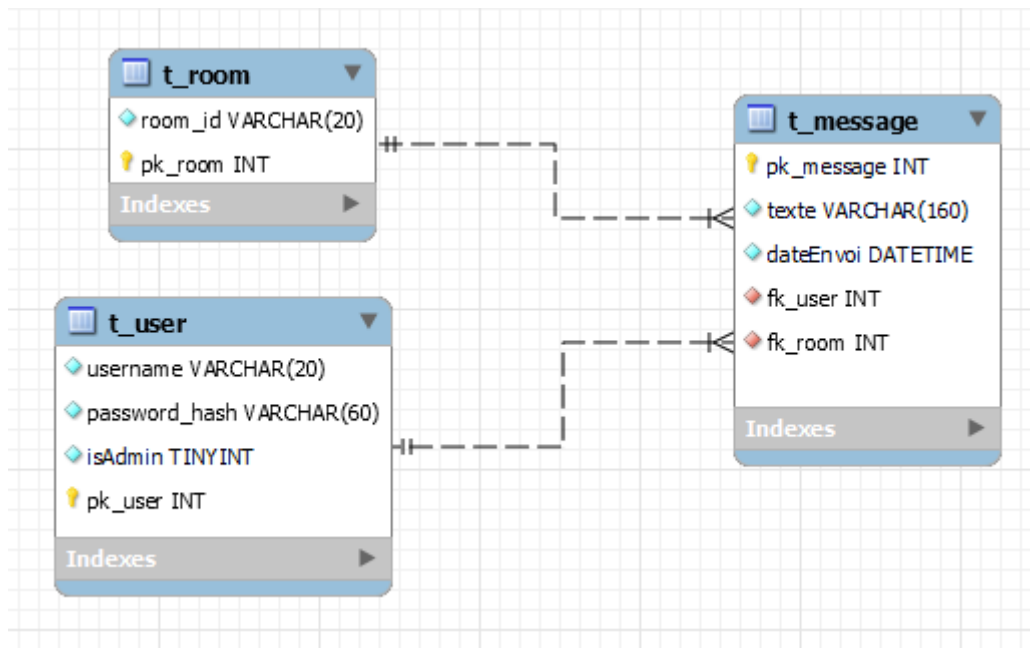
3.2. Diagramme de séquence interactions

151 - Intégrer des éléments de base de données dans des applications Web – Rapport Personnel



3.3. Schéma relationnel de la base de données

Voici les trois tables définies à l'analyse. La table centrale est celle des messages, elle est reliée avec un utilisateur et une salle de chat.



3.3.1. Table des utilisateurs

La table des utilisateurs contient simplement les champs, username, hash et un booléen pour les droits d'admin. Tout ces champs sont non-nul.








Table Name:

t_user

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 username	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 password_hash	VARCHAR(32)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 isAdmin	TINYINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 pk_user	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

3.3.2. Table des salles de tchat

La table de tchat ne contient qu'un nom qui est aussi une PK. J'ai décidé d'en faire une table à part (à la place d'un attribut dans le message) afin de pouvoir évoluer en ajoutant des champs liés aux salles dans le futur (comme le créateur).






Table Name:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 room_id	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 pk_room	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>






3.3.3. Table des messages

La table des messages stockera le texte du message, son auteur, la date d'envoi et la salle où il a été envoyé. Tous les champs sont non-nul.

151 - Intégrer des éléments de base de données dans des applications Web
– Rapport Personnel



Table Name: t_message

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
 pk_message	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 texte	VARCHAR(160)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 dateEnvoi	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 fk_user	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 fk_room	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Réalisation

4.1. Descente de code (envoi de messages)

4.1.1. Client

VueCtrl.js

Dans le contrôleur de l'interface, lorsque que le chat est chargé, deux listeners sont définis. Les deux font la même action, c'est-à-dire demander au contrôleur d'envoyer le message mais un des deux est sur le bouton « envoyer » et l'autre permet d'envoyer en appuyant sur « entrée »

```
loadChat() {  
  
  $("#send-btn").click((event) => {  
    var texte = $("#message-input").val();  
    $("#message-input").val("");  
    ctrl.sendMessage(texte, localStorage.getItem("currentRoom"));  
  });  
  $("#message-input").keypress(function(event) {  
    // le code 13 correspond à la touche entrée  
    if (event.which === 13) {  
      var texte = $("#message-input").val();  
      $("#message-input").val("");  
      ctrl.sendMessage(texte, localStorage.getItem("currentRoom"));  
    }  
  });  
  ctrl.loadRoom(localStorage.getItem("currentRoom"));  
  ctrl.autoReloadMessage();  
}
```

Ctrl.js

Dans le contrôleur, le message est vérifié pour qu'il ne soit pas vide et ne contienne pas d'injection HTML avant d'être transféré vers le service http. Les méthode « retour » sont aussi définies ici et seront appelé une fois que le serveur enverra une réponse.

```
sendMessage(texte, room_id) {  
  if (!texte || !texte.trim()) {  
    console.error("Text is null, empty, or contains only spaces.");  
    return;  
  }  
  if (room_id === null) {
```

```
        console.error("Room ID is null");
        return;
    }
    // empeche les injection HTML (merci chatGPT)
    texte = $("

").text(texte).html();
    console.log("Sending message:", texte, "to room:", room_id);

    this.http.sendMessage(
        texte,
        room_id,
        ctrl.sendMessageSuccess,
        ctrl.sendMessageError
    );
}


```

```
sendMessageSuccess(data, text, jqXHR) {
    console.log("message envoyé !");
    ctrl.loadRoom(localStorage.getItem("currentRoom"));
}
sendMessageError(data, text, jqXHR) {
    if (data.status == 413) {
        // Le message est trop long
        ctrl.vue.afficheStatut("Le message est trop long. La longueur maximale est de 160 caractères.", "red");

    } else if (data.status == 500) {
        // Une erreur s'est produite
        ctrl.vue.afficheStatut("Petit problème sur le serveur...", "red");

    } else if (data.status == 403) {
        // Pas logué
        ctrl.vue.afficheStatut("Pas autorisé ! Va te loguer petit chenapan.", "red");
    } else {
        ctrl.vue.afficheStatut("Le serveur ne répond pas...", "red");
    }
}
```

http.js

le service http permet d'envoyer des requêtes GET, PUT, POST, DELETE au serveur. Dans notre cas, c'est une requête POST qui est envoyée avec les paramètres suivants : action de type « message », puis le texte et la salle.

```
sendMessage(texte, roomId, successCallback, errorCallback) {
```

```
$.ajax({
  type: "POST",
  url: this.BASE_URL,
  data: {
    action: "message",
    texte: texte,
    room_id: roomId
  },
  xhrFields: {
    withCredentials: true
  },
  success: successCallback,
  error: errorCallback,
});
}
```

4.1.2. Serveur

Index.php

Sur l'index, un switch permet d'exécuter du code en fonction du type de requête (GET, PUT, POST, DELETE) puis dans le cas d'un POST, en fonction du paramètre « action » certains bouts de code sont exécutés. Voici un zoom sur le cas d'un « POST » avec l'action « message » qui permet de stocker un nouveau message.

```
case 'POST':
if (isset($_POST['action'])) {
$action = $_POST['action'];

switch ($action) {
case [...]
case 'message':
parse_str(file_get_contents("php://input"), $vars);
if (isset($vars['texte']) && isset($vars['room_id'])) {
$texte = $vars['texte'];
$room_id = $vars['room_id'];
// Envoyer un message
// Forward vers MessageManager.php

// test le message avant de le transférer.
// si il est trop long (>160 char) ça passe pas. (oui c'est une référence à la limite historique des SMS)
```

```
if (strlen($texte) > 160) {  
    http_response_code(413);  
    echo '<error>Le texte est trop long. Maximum 160 caractères. Longueur du message : ' . strlen($texte) . '</error>';  
} else {  
    // echo "enregistrer un message avec : texte ($texte), user ($user) et room_id ($room_id) .<br>";  
    $messageManager = new MessageManager();  
    echo $messageManager->send($room_id, $texte);  
}  
  
} else {  
    echo '<error>Paramètre texte, user ou room_id manquant pour un nouveau message</error><br>';  
}
```

MessageManager.php

Voici la méthode « send » du gestionnaire de messages qui est appelée par l'index. Cette méthode contrôle que la session actuelle a le droit d'envoyer un message (est loguée) avant d'écrire le message avec une autre méthode.

```
function send($room_id, $message)  
{  
    $session = SessionManager::getSessionInfo();  
    //check session  
    if ($session["isLogged"] == true) {  
        // user dans la session  
        $user = $session["username"];  
        http_response_code(200);  
        return $this->writeMessage($room_id, $user, $message);  
    } else {  
        //pas logué  
        http_response_code(403);  
    }  
}
```

La méthode « write » va sauvegarder le message dans la DB en évitant les injections SQL.


```
private function writeMessage($room_id, $user, $message)
{
    $connection = WrkDb::getInstance();
    //définit les champs manquant
    $fk_user = $connection->executeQuery("SELECT * FROM t_user WHERE username = ?", array($user))->fetch(PDO::FETCH_ASSOC)["pk_user"];
    $dateEnvoi = date('Y-m-d H:i:s'); // Format: YYYY-MM-DD HH:MM:SS

    // Preparation du SQL
    $sql = "INSERT INTO t_message (texte, dateEnvoi, fk_user, fk_room) VALUES (:texte, :dateEnvoi, :fk_user, :fk_room)";
    $params = array(':texte' => $message, ':dateEnvoi' => $dateEnvoi, ':fk_user' => $fk_user, ':fk_room' => $room_id);

    $query = $connection->executeQuery($sql, $params);

    return $query->rowCount();
}
```

SessionManager.php

Le gestionnaire de message appelle le gestionnaire de session pour savoir si l'utilisateur a le droit d'envoyer le message. Cette méthode retourne simplement les informations stockées dans la session.

```
static function getSessionInfo()
{
    if (isset($_SESSION["isLogged"])) {
        return array(
            'username' => $_SESSION['username'],
            'isLogged' => $_SESSION['isLogged'],
            'admin' => $_SESSION['isAdmin']
        );
    } else {
        return array('username' => "", 'isLogged' => "", 'admin' => "");
    }
}
```

WrkDb.php

La méthode `executeQuery` permet, comme son nom l'indique, d'exécuter une requête. Elle utilise les prepared statements pour contrer les injections SQL.

```
public function executeQuery($query, $params = array())
{
    try {
        $stmt = $this->pdo->prepare($query);
        $stmt->execute($params);
        return $stmt;
    } catch (PDOException $e) {
        http_response_code(500);
        error_log("Erreur lors de l'exécution de la requête: " . $e->getMessage());
        die("Erreur lors de l'exécution de la requête: " . $e->getMessage());
    }
}
```

4.2. Tests fonctionnels

Utilisateur	Cas	Tests	Attendu	Obtenu	OK/NOK
Visiteur	Envoyer un message	Un visiteur non connecté essaye d'envoyer un message	Un message s'affiche lui disant qu'il doit se connecter pour envoyer un message.	Un message apparait disant qu'il faut aller se loguer.	OK
	Envoyer un message	Un visiteur envoie un message en étant connecté	L'envoi du message est possible et il s'affiche à la suite des autres messages	Les Messages sont rafraichis et le nouveau est affiché dans la liste	OK
Client	Envoyer un message	Un visiteur envoie un message contenant une injection SQL	L'injection SQL n'est pas interprétée / un message d'erreur s'affiche	L'injection n'est pas exécutée mais est affiché comme n'importe quel autre message	OK
	Envoyer un message	Un visiteur envoie un message avec une injection HTML	L'injection HTML n'est pas interprétée / un message d'erreur s'affiche	L'injection n'est pas exécutée mais est affiché comme n'importe quel autre message	OK
	Envoyer un message	Un visiteur envoie un message vide	Le message n'est pas envoyé, il ne se passe rien (un message d'erreur s'affiche ?)	Une erreur (dans la console, rien n'est visible sur le site) nous dit que le texte est vide	OK
Administrateur	Envoi/Gestion d'un message	Un administrateur va sur la page de chat où figurent les messages	Il a la possibilité en plus d'envoyer lui aussi des messages de supprimer les messages qui figurent déjà dans le chat	Le bouton « passer en mode admin est en plus par rapport à un utilisateur	OK
	Envoyer un message	Un administrateur envoie un message contenant une injection SQL	L'injection SQL n'est pas interprétée / un message d'erreur s'affiche	L'injection n'est pas exécutée mais est affiché comme n'importe quel autre message	OK
	Envoyer un message	Un administrateur envoie un message vide	Le message n'est pas envoyé, il ne se passe rien (un message d'erreur s'affiche ?)	Les messages des admins sont traités de la même manière que les utilisateurs,	OK

151 - Intégrer des éléments de base de données dans des applications Web
– Rapport Personnel

				les messages vides ne sont pas envoyés	
--	--	--	--	--	--

4.3. Problèmes rencontrés

4.4. Hébergement

Pour héberger le site WEB sur notre domaine de l'école, nous devons faire quelques ajustements au code, comme les adresses autorisées par CORS, l'Endpoint du client et les paramètres de connexion à la DB.

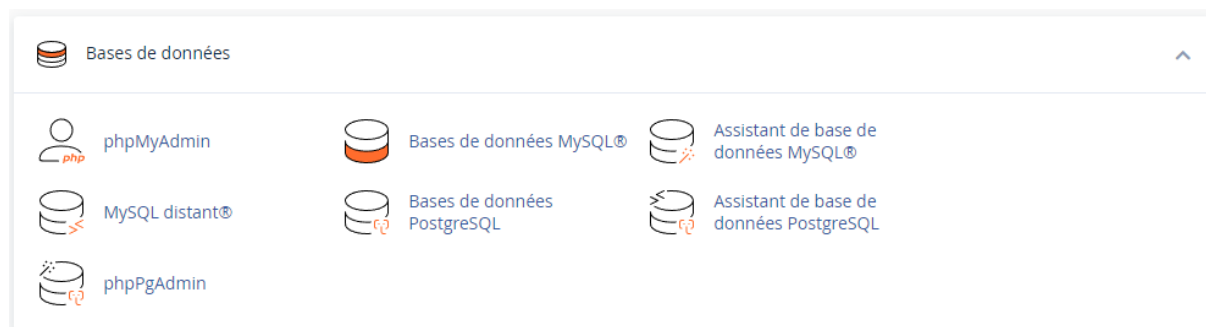
Ce ne sera plus localhost mais le FQDN du site WEB.

Dans mon cas : 151.gendres.emf-informatique.ch pour CORS et l'Endpoint et ces informations pour la base de données :

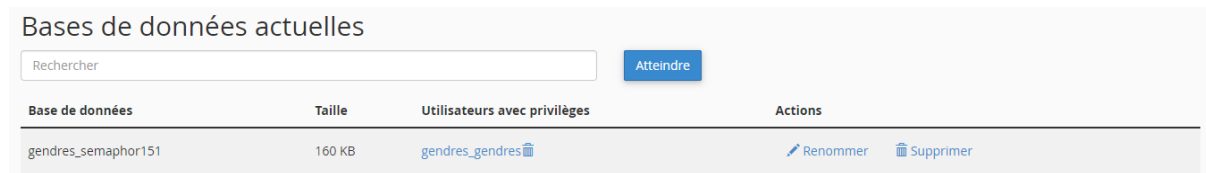
```
<?php
//Informations de connexion à la base de données
define('DB_TYPE', 'mysql');
define('DB_HOST', 'gendres.emf-informatique.ch:3306');
define('DB_NAME', '[REDACTED]');
define('DB_USER', '[REDACTED]');
define('DB_PASS', '[REDACTED]');
```

Il faudra aussi créer une DB et y mettre les données. Pour ce faire :

Nous pouvons nous connecter à <https://cpanel.emf-informatique.ch/> et nous rendre dans l'assistant de création de base de données.



Elle apparaîtra ensuite dans la liste des DB. Nous pouvons maintenant nous y connecter depuis MySQL Workbench pour y mettre des données.



C'est une connexion standard sur le FQDN de notre domaine. Les manipulations peuvent être faites comme nous l'avons fait pendant tout le module.

**151 - Intégrer des éléments de base de données dans des applications Web
– Rapport Personnel**

Connection Name:

Connection Remote Management System Profile

Connection Method: Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

5. Synthèse

5.1. Différence de timing

5.1.1. Planning avec les croix

	Bloc 1			Bloc 2				Bloc 3			Bloc 4			Bloc 5		
	29.01.2023 - MAT	29.01.2023 - A-M	30.01.2023 - MAT	05.02.2023 - MAT	05.02.2023 - A-M	06.02.2023 - MAT		19.02.2023 - MAT	19.02.2023 - A-M	20.02.2023 - MAT	26.02.2023 - MAT	26.02.2023 - A-M	27.02.2023 - MAT	04.03.2023 - MAT	04.03.2023 - A-M	05.03.2023 - MAT
Tâches																
Exercices / découverte	x	x	x	x				x	x							
Analyse																
Introduction	x	x														
use cases			x	x												
maquettes				x												
diagramme d'activités				x												
diagramme de séquences				x	x											
schéma ER				x												
Conception																
diagramme de classe serveur																
diagramme de classe client						x				x						
schéma relationnel DB					x											
diagramme de séquence interactions					x			x		x						
Réalisation																
Serveur - structure de base								x								
Serveur - login										x						
Serveur - récupérer les messages											x					
Serveur - envoyer des messages											x	x				
Serveur - création de comptes													x			
Serveur - créer et rejoindre des rooms																
Client - structure de base								x	x							
Client - login									x	x						
Client - récupérer les messages											x					
Client - création de comptes													x			
Client - envoyer des messages											x	x				
Client - créer et rejoindre des rooms																
descente de code														x		
Hebergement															x	x
Tests de l'application														x	x	

5.1.2. Analyse du planning

Dans le planning on peut voir que j'ai globalement mal estimé l'implémentation, il n'y a (presque) aucune croix qui s'aligne avec ce qui avait été prévu. Autrement, pour l'analyse et la conception, j'étais un peu plus juste.

5.2. Différence avec la conception

Côté client, le code correspond au diagramme de classe à la différence qu'un contrôleur de vue a été ajouté. Il permet de faire le lien entre le contrôleur principal et le chargement des HTML et CSS.

Et côté serveur, je n'ai pas du tout implémenté la classe « Wrk » et j'ai directement interfacé les sous-worker avec l'index PHP. Les beans que j'avais prévu ne sont pas trop utilisés (Je me suis forcé à quand même le faire même si j'aurais trouvé plus simple de travailler avec des tableaux)

Les maquettes sont globalement bien respectées.

5.3. Conclusion

5.3.1. Ce que je retiens de ce module

J'ai beaucoup aimé ce module. Surtout la partie projet une fois dans l'implémentation. Je m'étais toujours demandé comment fonctionnait le Web coté serveur et je pensais que c'était plus compliqué que ce ça l'est vraiment.

5.3.2. Amélioration / proposition

Si le module était à refaire, je proposerais de laisser plus de temps au projet / demander moins de fonctionnalités. Je pense que le temps en classe n'est clairement pas suffisant pour finir le projet.

5.3.3. Mes points forts et faibles

Comme toujours, mes éternels points faibles sont la documentation et la conception. J'ai toujours de la peine à me projeter sur *comment faire* au lieu de *faire* (j'ai envie de sauter tête baissée dans le code). Autrement, je pense avoir bien compris le fonctionnement du PHP et je n'ai pas eu trop de blocages dans le projet (à part Enterprise Architect Grrr).