



Candidat n°  
151481

Nom du candidat : Esther Pham

## WebMenu

### Sommaire

<b>Résumé du rapport du TPI.....</b>	<b>2</b>
1.1 Situation de départ.....	2
1.2 Mise en œuvre .....	2
1.3 Résultats.....	2
<b>1 Les grandes lignes du projet .....</b>	<b>3</b>
1.1 Analyse de la situation initiale .....	3
1.2 Analyse de l'état désiré .....	3
1.3 Cahier des charges / exigences du système .....	4
1.4 Organisation du projet.....	7
1.5 Travaux préparatoires .....	10
<b>2 Analyse .....</b>	<b>11</b>
2.1 Cas d'utilisation.....	11
2.2 Analyse de l'état actuel .....	12
2.3 Diagrammes d'activités .....	12
2.4 Diagrammes de séquence .....	15
2.5 Définition des endpoints.....	19
2.6 Analyse de l'état désiré / maquettes.....	20
2.7 Sécurité de l'information et protection des données .....	22
<b>3 Travaux préparatoires .....</b>	<b>23</b>
3.1 Librairie d'animation javascript .....	23
<b>4 Conception.....</b>	<b>26</b>

4.1	Exigences du système .....	26
4.2	Architecture du système.....	26
4.3	Concept d'implémentation.....	28
4.4	Concept de tests .....	31
<b>5</b>	<b>Réalisation .....</b>	<b>32</b>
5.1	Utilisation de l'application .....	32
5.2	Modification de la base de données .....	32
5.3	Design du système .....	32
5.4	Code par rapport aux diagrammes d'activités .....	33
<b>6</b>	<b>Test.....</b>	<b>46</b>
6.1	Procédure de test.....	46
6.2	Protocol de test .....	46
6.3	Signature du protocole de test .....	47
<b>7</b>	<b>Conclusion.....</b>	<b>48</b>
7.1	Améliorations possibles .....	48
7.2	Auto-évaluation .....	48
<b>8</b>	<b>Bibliographie: liste des sources et références.....</b>	<b>49</b>
<b>9</b>	<b>Glossaire .....</b>	<b>50</b>
<b>10</b>	<b>Signatures.....</b>	<b>52</b>
<b>11</b>	<b>Annexes .....</b>	<b>53</b>

# Résumé du rapport du TPI

## 1.1 Situation de départ

Lors d'événements tels que festivals ou concerts, l'affichage des menus peut poser des problèmes : supports papier qui tombent ou s'envolent et tableaux durs à lire. Le WebMenu résout ces défis avec un affichage numérique adaptable à tous les écrans, offrant une meilleure lisibilité grâce à des polices personnalisables en taille et couleur. L'application permet aussi d'indiquer en temps réel les articles épuisés. Elle repose sur des pages HTML pour le client et un serveur web connecté à une base de données pour gérer les demandes des utilisateurs.

## 1.2 Mise en œuvre

Pour ce projet, la gestion s'est déroulée en quatre phases : l'analyse, la conception, la réalisation et les tests.

Après avoir pris connaissance du cahier des charges, j'ai rencontré mon supérieur professionnel pour qu'il puisse répondre à mes différentes questions. Une fois ces interrogations éclaircies, j'ai pu démarrer le projet et planifier les activités associées à chacune des quatre phases. Dès le début du projet, des experts sont venus s'assurer que j'étais sur la bonne voie.

Après cette première visite, j'ai élaboré les différents schémas nécessaires à l'analyse et à la conception. Une fois ces schémas finalisés, j'ai entamé la phase de réalisation. Une seconde visite des experts a eu lieu pendant cette étape, afin de vérifier que le projet progressait correctement.

Enfin, après avoir terminé le projet, des tests de fonctionnement ont été réalisés pour s'assurer de sa conformité et de son efficacité.

## 1.3 Résultats

L'application répond pleinement au cahier des charges et est entièrement fonctionnelle. Cependant, aucune fonctionnalité supplémentaire n'a pu être implémentée.

L'application est disponible sous ces 2 url

- [Menu](#)
- [Gestion des articles](#)

Pour garantir la sécurité des données sensibles, le protocole HTTPS a été adopté. Tous les tests ont été réalisés avec succès, et les résultats sont conformes aux attentes. Cela confirme le bon fonctionnement de l'application.

# 1 Les grandes lignes du projet

## 1.1 Analyse de la situation initiale

Lors de festivals, manifestations, concerts ou autres événements, il est naturel de vouloir manger ou boire quelque chose. Cependant, les restaurateurs rencontrent souvent des difficultés pour afficher efficacement leurs menus de manière claire et accessible. Les menus imprimés, qu'ils soient sur des feuilles grandes ou petites, ont tendance à tomber ou à s'en-voler, tandis que les tableaux noirs ou blancs peuvent être difficiles à lire, surtout de loin ou en cas d'écriture peu soignée.

Le WebMenu propose une solution à tous ces problèmes grâce à un affichage numérique adaptable sur des écrans de toutes tailles. Il permet de personnaliser la taille et la couleur de la police pour une meilleure lisibilité. Avec cette application, fini les soucis d'un menu qui tombe ou d'une écriture illisible. De plus, il est possible d'indiquer directement sur le menu numérique lorsqu'un article est épuisé, permettant ainsi aux clients de savoir en temps réel ce qui est disponible.

Ce projet repose sur des pages HTML côté client et un serveur web qui héberge l'application. Ce dernier accède à une base de données relationnelle pour traiter les requêtes des utilisateurs.

## 1.2 Analyse de l'état désiré

Le projet final devrait permettre aux restaurateurs de mettre à jour et d'afficher leur menu de manière claire et simple grâce à une solution numérique. Ils auront la possibilité de personnaliser la police d'écriture, la couleur et les thèmes du menu via une base de données. Ils pourront également mettre à jour en temps réel les articles épuisés ainsi que toutes autres informations relatives aux produits.

Le système sera conçu pour être compatible avec des écrans de toutes tailles, qu'il s'agisse de tablettes, de télévisions ou d'autres dispositifs, afin de s'adapter aux différents besoins des utilisateurs.

Cette solution permettra aux restaurateurs de gagner un temps précieux en simplifiant la gestion de leurs menus, tandis que les clients bénéficieront d'une expérience utilisateur améliorée grâce à une lecture facilitée et des informations toujours à jour. En conséquence, la gestion optimisée des articles réduira les frustrations liées à la demande d'articles épuisés

### 1.3 Cahier des charges / exigences du système

Voici les exigences du système tirées du cahier des charges. Ces exigences correspondent aux différents éléments qui devront être mis en place pour ce projet :

Exigence fonctionnelle	Description	Priorité	Temps estimé
		Haute Moyenne Basse	(jours)
<b>Administration</b>	Administration, planning, visites des experts et séances avec le supérieur professionnel.	Haute	0.5 jour
Analyse / Conception	À partir de ce document et de la discussion avec le supérieur professionnel, analyse de la base de données existantes, la structuration du client et du serveur ainsi que la création des mock-ups. La documentation doit être traitée en parallèle avec l'analyse et la conception	Haute	1.5 jour
Réalisation des Web-Services GET (lecture des articles)	<p>L'implémentation de cette exigence doit permettre :</p> <ul style="list-style-type: none"> <li>• De créer la structure du serveur</li> <li>• De se connecter à la base de données</li> <li>• De créer les différents entrypoints (GET) du serveur</li> <li>• De lire les données de la base de données, les traiter et les mettre au format JSON</li> <li>• D'implémenter la logique des pages et des tris</li> </ul> <p>La documentation doit être traitée en parallèle avec les développements.</p>	Haute	2 jours

Réalisation des pages d'affichage des articles	<p>L'implémentation de cette exigence doit permettre :</p> <ul style="list-style-type: none"> <li>• De créer la structure du projet côté client</li> <li>• De récupérer les données du serveur et de les mettre en page</li> <li>• De gérer la rotation des pages</li> <li>• De gérer la relecture de des données et la mise à jour des pages</li> </ul> <p>La documentation doit être traitée en parallèle avec les développements.</p>	Haute	2 jours
Réalisation des Web-Services POST/UPDATE/DELETE (mise à jour des articles)	<p>L'implémentation de cette exigence doit permettre :</p> <ul style="list-style-type: none"> <li>• De gérer le login (mot de passe) et la sécurité des entrypoints</li> <li>• De créer les différents entrypoints serveur</li> <li>• De gérer les données avec les opérations CRUD sur la base de données</li> </ul> <p>La documentation doit être traitée en parallèle avec les développements.</p>	Haute	2 jours
Réalisation des pages de modification des articles	<p>L'implémentation de cette exigence doit permettre :</p> <ul style="list-style-type: none"> <li>• Se connecter avec le mot de passe</li> <li>• De créer une page pour modifier les articles existants (description, prix et disponibilité les données du serveur et de les mettre en page)</li> <li>• De créer une page pour ajouter un article dans un groupe</li> <li>• De créer une page pour effacer un article dans un groupe</li> </ul> <p>La documentation doit être traitée en parallèle avec les développements</p>	Haute	2 jours



Réalisation supplé- mentaire sur la modifi- cation des groupes	<ul style="list-style-type: none"> <li>• L'implémentation de cette exigence doit permettre : <ul style="list-style-type: none"> <li>• D'ajouter, de modifier ou d'effacer un groupe d'articles</li> <li>• De gérer les groupes sur les différentes pages</li> <li>• De gérer les couleurs de l'application, des groupes et des articles</li> <li>• De changer la vitesse de rafraichissement des pages.</li> </ul> </li> </ul> <p>La documentation doit être traitée en parallèle avec les développements.</p>	Basse	
--	--	-------	--

*Exigences fonctionnelles tirées du cahier des charges*



## 1.4 Organisation du projet

### 1.4.1 Méthode de gestion du projet

La méthode utilisée pour la gestion de projet est la méthode **Waterfall**. Elle se caractérise par une division en phases distinctes, où chaque phase ne peut commencer qu'une fois la précédente achevée. Cette méthode permet de déterminer précisément la durée et le moment où chaque tâche doit être réalisée. Pour ce projet, elle m'a permis de conserver une ligne directrice claire, évitant ainsi de m'égarer dans les tâches ou de perdre du temps inutilement.

Le planning ci-dessous illustre six phases distinctes de travail. Cependant, les phases de « Tests » et de « Documentation » seront menées en continu tout au long du projet. En effet, chaque tâche doit être soigneusement documentée, et chaque implémentation de nouvelles fonctionnalités sera testée dès son achèvement.

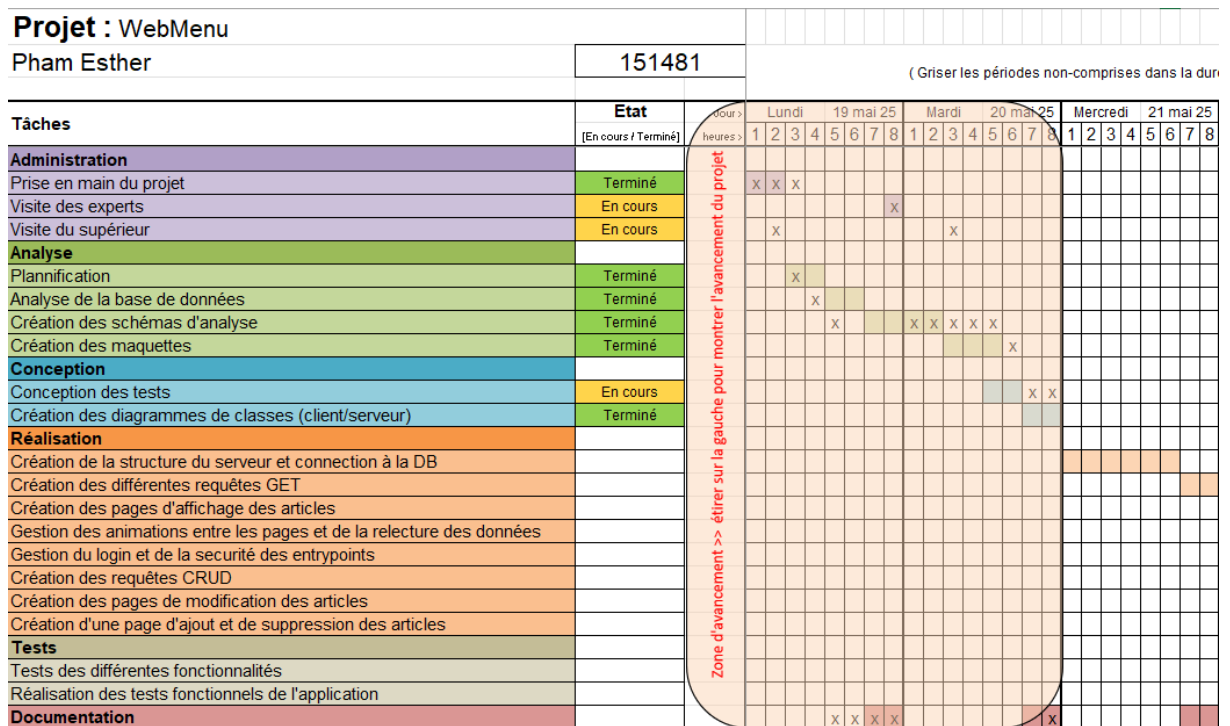


Figure 1 - Capture d'écran de mon planning



Toutes les tâches réalisées sont consignées dans le journal de travail, où sont également notés le temps passé et la date correspondante. Un code couleur a été instauré pour distinguer les différents événements importants :

**Projet :** WebMenu

Pham Esther		151481			
				Visites et décisions	Solutions
				Problèmes rencontrés	Absences
Date	Travail effectué	Temps [h.h]			
19.05.25	Prise en main du projet, lecture du CDC et des documents se trouvant sur PKOrg	2,0			
	Visite du supérieur : discussion des différentes documentations du projet	0,5			
	Création du planning, choix des différentes activités à réaliser pour le projet	0,5			
	Analyse de la base de données	1,0			
	Création du schéma use case	0,5			
	Documentation du chapitre 1 (Les grandes lignes du projet)	2,0			
	Documentation du chapitre 2.1.3 (Processus d'entreprise concernés)	1,0			
	Visite des experts : - Un seul expert présent, M Waelti - Documentation très importante, pour les photos il faut les mettre dans un dossier "Annexes" et indiquer le nom qui correspond à l'image dans la doc - Présentation finale se fait en 2 parties --> choisir si on veut que le supérieur vienne - La réalisation des requêtes CRUD prendra plus de temps	0,5			
	Réflexion personnelle ↓↓↓				
	Total >	8,0			
20.05.25	Pendant la visite des experts, on m'a expliqué qu'il faut que je documente bien et que ce soit complet. Il faut que je fasse attention aussi à bien remplir mon journal de travail et mon planning. Je pense que pour le planning, comme ils me l'ont fait remarquer, je dois faire attention pendant la création des services CRUD : cela va prendre plus de temps.				
	Création des diagrammes d'activités	2,5			
	Création des diagrammes de séquence	2,5			
	Visite du supérieur : Discussion des schémas d'analyse et des modifications à faire. Validation des sch	1,0			
	Création des maquettes	1,0			
	Conception des tests et documentation des tests	1,0			
	Réflexion personnelle ↓↓↓				
	Total >	8,0			

Figure 2 - Capture d'écran de mon journal de travail

#### 1.4.2 Liste des participants du projet

Voici les participants sur ce projet ainsi que leurs rôles :

Participant au projet	Rôle
<b>Frédéric Hertling</b>	Formateur en entreprise
<b>Albert Waelti</b>	Expert principal
<b>Loris Roubaty</b>	Expert secondaire
<b>Pierre-Alain Mettraux</b>	Supérieur professionnel
<b>Esther Pham</b>	Candidate

#### 1.4.3 Gestion des sauvegardes et du versioning

Il est impératif de sauver ses projets constamment pour ne pas perdre le travail en cas de problèmes. Voici mon plan de sauvegarde et de versioning :

- Une sauvegarde dans mon OneDrive personnel tous les soirs  
*Sauvegarde de la documentation, de la gestion du projet et du code*
- Une sauvegarde dans mon GitHub mis à jour tous les soirs

*Sauvegarde de la documentation, de la gestion du projet et du code*

- Une sauvegarde sur une clé USB tous les soirs

*Sauvegarde de la documentation, de la gestion du projet et du code*

OneDrive permet d'avoir un historique de version tout comme GitHub.

Ma clé USB possède toujours les dernières versions des différents éléments.

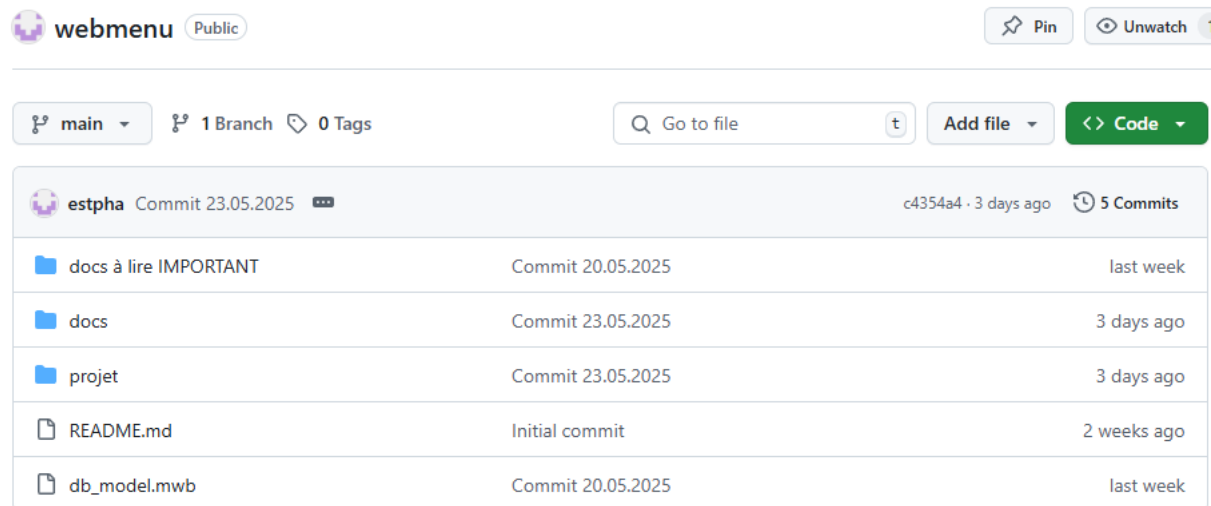


Figure 3 - <https://github.com/estpha/webmenu.git>

#### 1.4.4 Environnement de développement

Pour travailler sur ce projet un ordinateur est fourni par l'école sous Windows 11 :

Catégorie	Application	Version
Éditeur de code	VSCode	V1.100.0
GUI Git	GitHub Desktop	V3.4.19
Serveur web local	WAMP	V3.3.7
Gestion des bases de données	MySQL Workbench	V8.0
Navigateur web	Microsoft edge	V136.0.3240.76

## 1.5 Travaux préparatoires

Des travaux préparatoires ont été réalisés du 12.05.2025 au 16.02.2025 pour faciliter la réalisation du projet plus tard. Ces travaux préparatoires sont listés dans le cahier des charges (chapitres 5 – Travaux préparatoires) :

- Installation des outils de développement qui sont laissés au choix du candidat et des outils d'administration
- Mise en place d'outil de dépôt GIT pour gérer les versions et les sauvegardes
- Préparation d'un serveur avec un accès public pour l'hébergement de la partie serveur du projet
- Test d'une librairie JavaScript pour la rotation de pages avec si possible avec un effet agréable à l'œil au moment du changement de page
- Test d'un procédé permettant de gérer l'accès aux points d'entrée du serveur qui modifient les données de la base de données.

*Liste tirée du cahier des charges*

La réalisation de ces tests est documentée plus tard dans la documentation (chapitre 3 – Travaux préparatoires).

## 2 Analyse

### 2.1 Cas d'utilisation

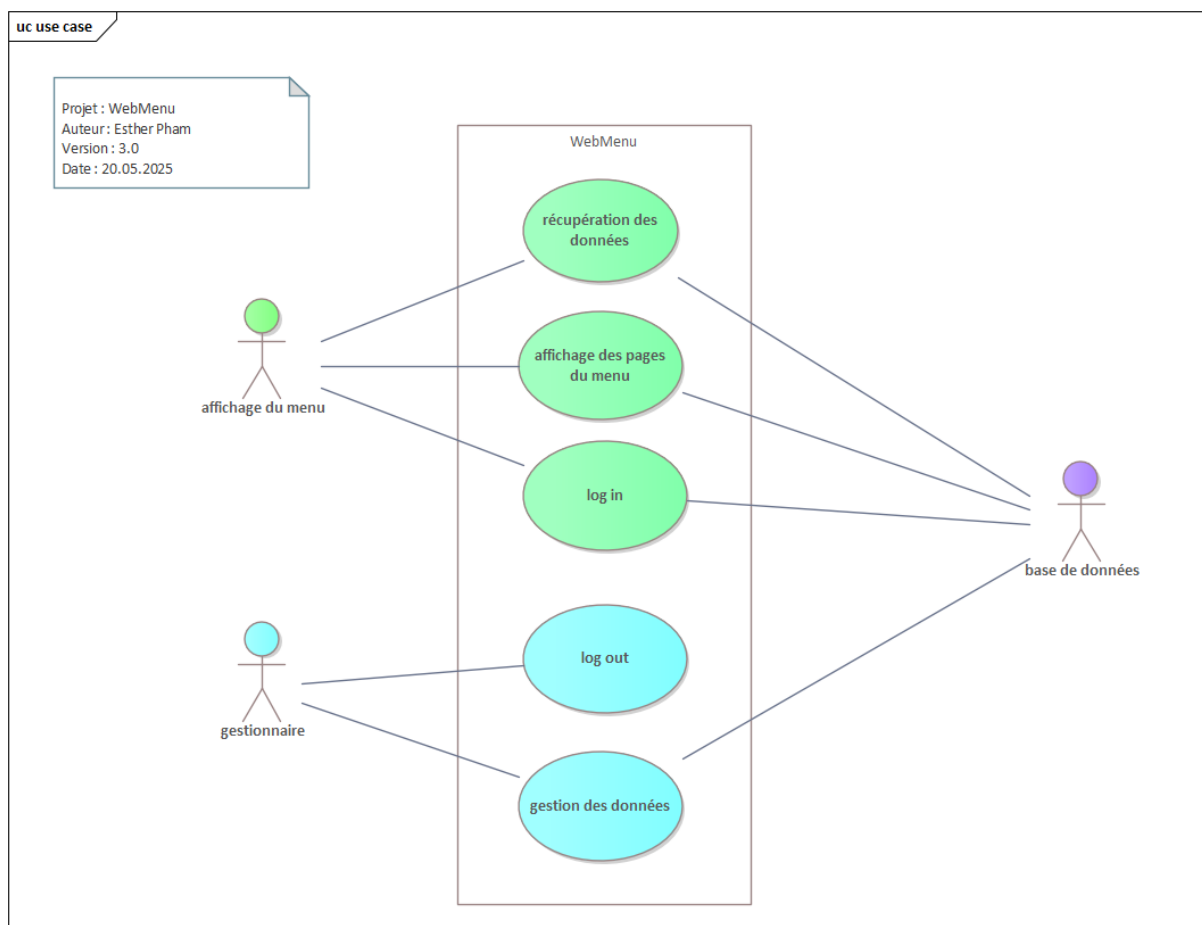


Figure 4 - useCase.png

#### 2.1.1.1 Description des acteurs

Acteurs	Description
<b>Affichage du menu</b>	L'affichage du menu correspond aux pages affichant le menu avec les différents articles.
<b>Gestionnaire</b>	Le gestionnaire correspond à l'admin de l'application. Pour accéder à ce statut, il faut se connecter.

#### 2.1.1.2 Description des actions

Action	Description
--------	-------------

<b>Récupération des données</b>	Le serveur effectue une requête GET pour récupérer la configuration et les données nécessaires à l'affichage du menu et de ses articles.
<b>Affichage des pages du menu</b>	Le client affiche le menu avec les différents groupes et leurs articles.
<b>Log in</b>	L'utilisateur passe de l'état « affichage du menu » à l'état « gestionnaire » en s'authentifiant avec un mot de passe.
<b>Log out</b>	L'utilisateur retourne à l'état « affichage du menu » en se désauthentifiant.
<b>Gestion des données</b>	Le gestionnaire peut ajouter, supprimer ou modifier les articles et les configurations présents dans la base de données.

## 2.2 Analyse de l'état actuel

Comme mentionné dans le chapitre 1.1 – Analyse de la situation actuelle, il reste difficile pour les restaurateurs d'afficher leurs menus lors de festivals, concerts et autres événements. Les méthodes traditionnelles, telles que les affiches imprimées et les tableaux blancs ou noirs, présentent plusieurs inconvénients dans ce type de contexte :

- Les affiches imprimées ont tendance à tomber ou à s'envoler facilement, et elles sont difficiles à lire si elles ne sont pas suffisamment grandes.
- Les tableaux blancs et noirs peuvent être difficiles à lire, surtout si l'écriture est petite ou peu soignée.

Ces limitations entravent une communication claire entre les restaurateurs et les participants, entraînant ainsi des pertes de temps considérables pour les deux parties.

## 2.3 Diagrammes d'activités

### 2.3.1 GET de la configuration

Ce diagramme représente le processus de requête GET pour récupérer les articles du menu. Lorsqu'on accède à l'URL du menu, le chargement de la page déclenche la requête GET de

la configuration. Le client envoie une demande au serveur, qui à son tour effectue une requête à la base de données. La base de données retourne les configurations, et le client les récupère pour les afficher.

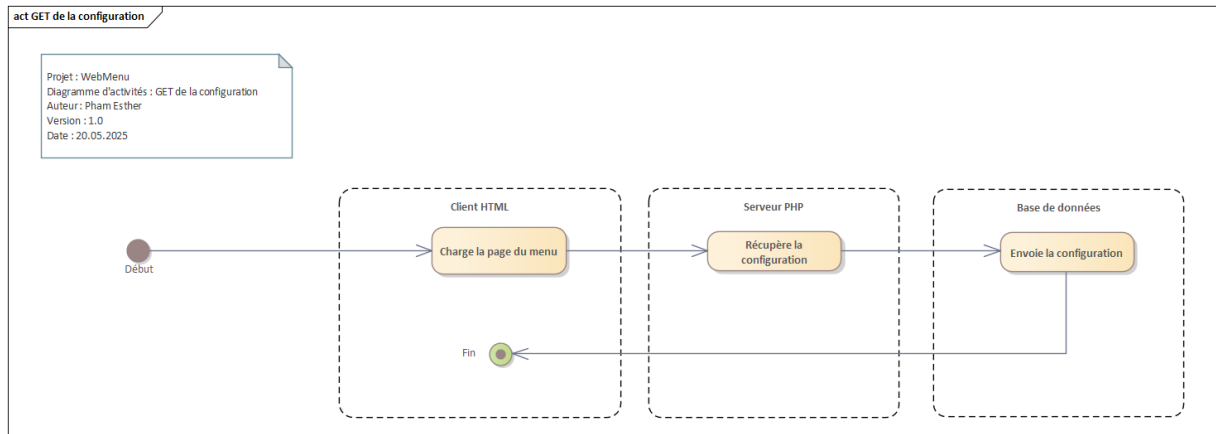


Figure 5 - diagAct\_getConf.png

### 2.3.2 GET des articles

Ce diagramme illustre le processus de requête GET pour récupérer les articles du menu. Lorsqu'on accède à l'URL du menu, le chargement de la page initie la requête GET des articles. Le client envoie une demande au serveur, qui effectue ensuite une requête à la base de données. La base de données retourne les articles, que le client affiche ensuite.

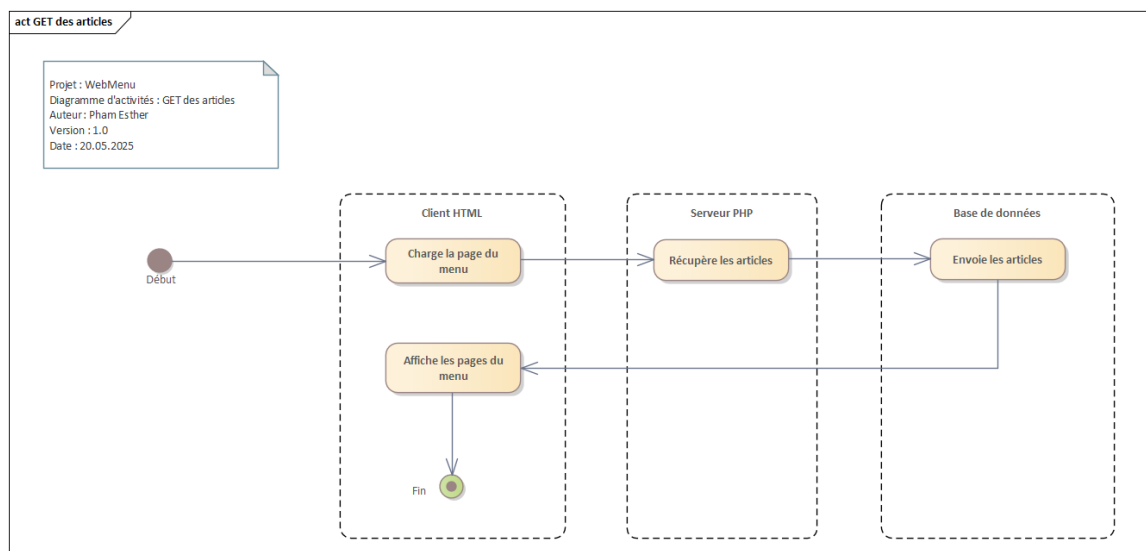


Figure 6 - diagAct\_getArt.png

### 2.3.3 CRUD des articles / de la configuration

Ce diagramme d'activité correspond à l'action d'ajouter, de supprimer ou de modifier les articles. Lorsque le bouton est appuyé, les données sont vérifiées avant l'ajout, la suppression ou la modification. Une fois qu'elles sont vérifiées, la base de données est modifiée.

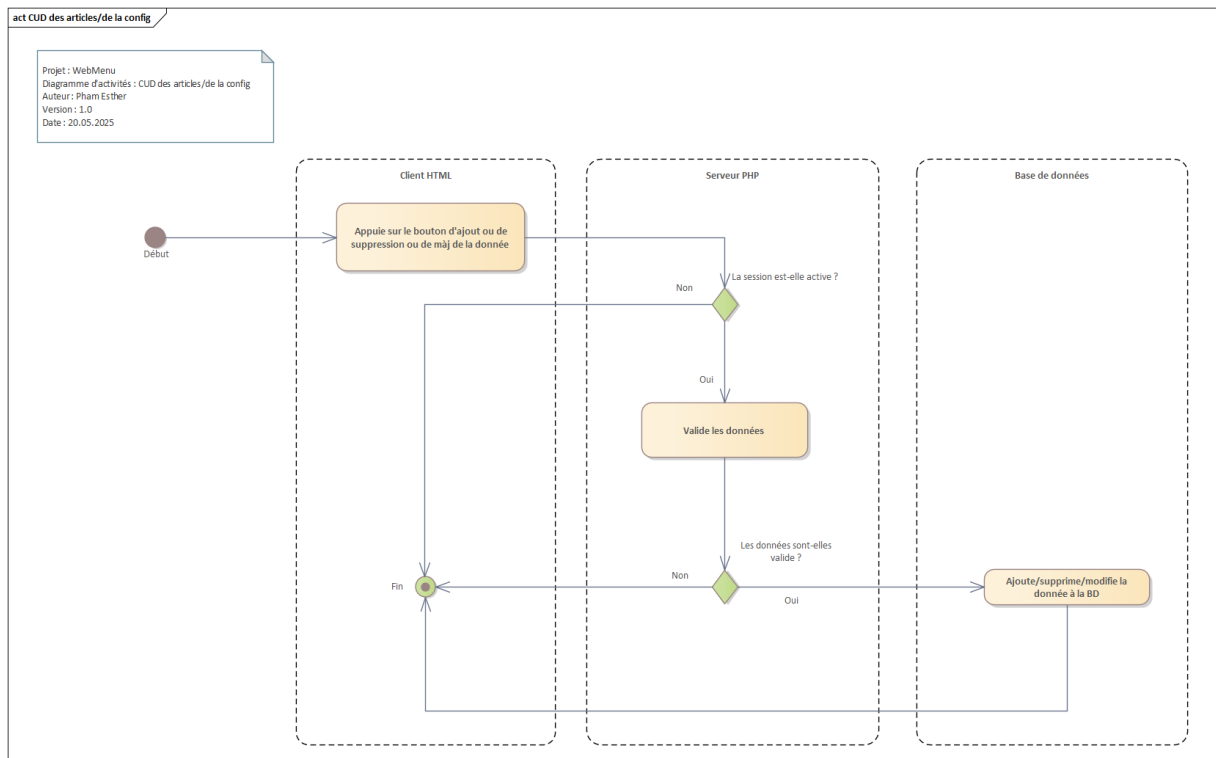


Figure 7 - diagAct\_cudArtConf.png

### 2.3.4 Rechargement de la page menu

Ce diagramme montre la répétition du chargement de la page d'article pour que les articles soient toujours à jours.

Tout d'abord, la couleur des activité GET dans ce schéma est différente des autres activités, car ce sont des activités qui correspondent aux diagrammes d'activités parlées auparavant (chapitres 2.3.1 – GET de la configuration et 2.3.2 - GET des articles). À chaque appel GET, un check du contenu du résultat de la requête est fait pour vérifier si le résultat est vide ou pas.

Une fois les GET effectués, le client affiche les différentes pages du menu avec les différents groupes et les articles y correspondant.

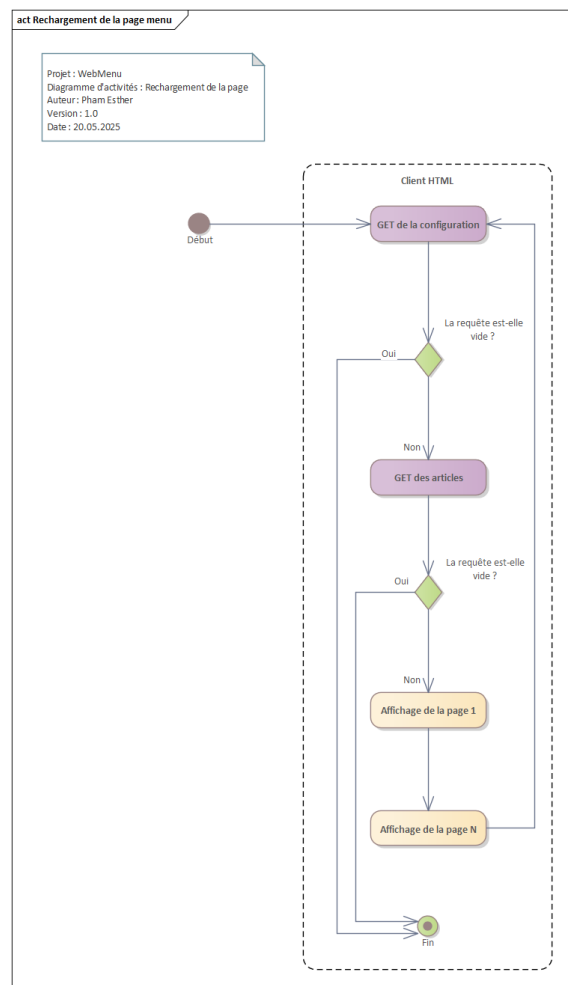


Figure 8 - diagAct\_rechargePage.png

## 2.4 Diagrammes de séquence

### 2.4.1 Lecture des données

Ce diagramme décrit en détail les différents diagrammes d'activités. Ce diagramme indique quel code d'erreur HTTP sera renvoyé si les données reçues sont vides.

Le code d'erreur renvoyé sera 400, il correspond le mieux à l'erreur.



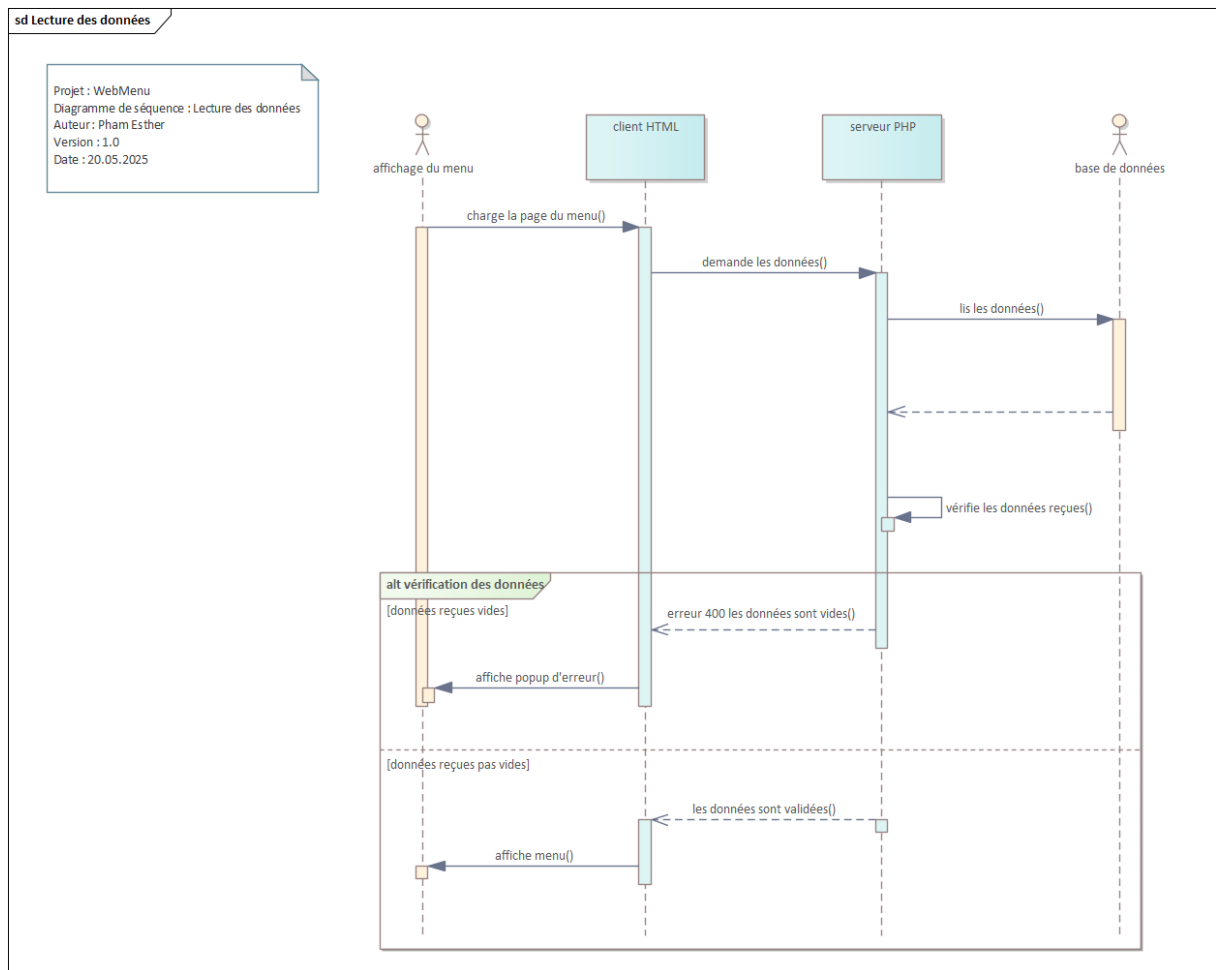


Figure 9 - diagSeq\_lectureDonnees.png

### 2.4.2 Ajout/suppression/modification des données

Pour l'ajout, la suppression et la modification, plusieurs codes d'erreurs sont renvoyés car il faut un login pour accéder à cette page. Voici les différents tests et les codes d'erreurs renvoyés pour chacun.

- **Vérifie l'état de la session** : Lorsque la demande de la modification de la base de données est envoyée, si l'utilisateur n'est pas connecté, le code d'erreur 401 est renvoyé au client.
- **Validation des données** : Lors de la validation des données, si celles-ci sont invalides le code d'erreur 400 est renvoyé.
- **Vérification de l'ajout/suppression/modification** : Pendant la vérification de la modification de la base de données, si une erreur est survenue le code d'erreur renvoyé est 503.

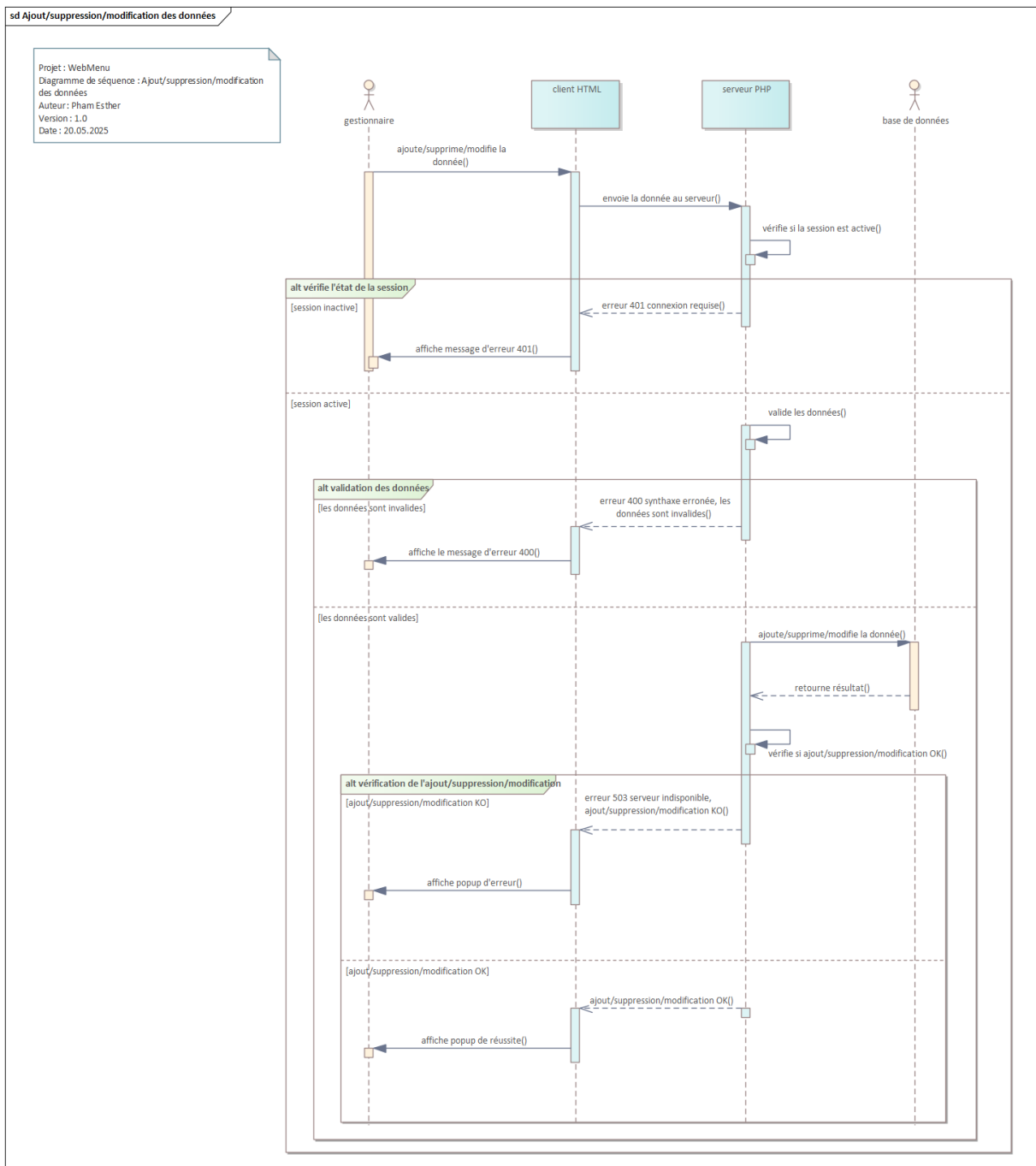


Figure 10 - diagSeq\_ajoutSuppModif.png

### 2.4.3 Rechargement de la page

Dans ce diagramme aucun code d'erreur n'est indiqué mais il sera affiché dans le popup d'erreur. Comme indiqué dans le schéma, la récupération des données fait appel à 2 autres schémas.

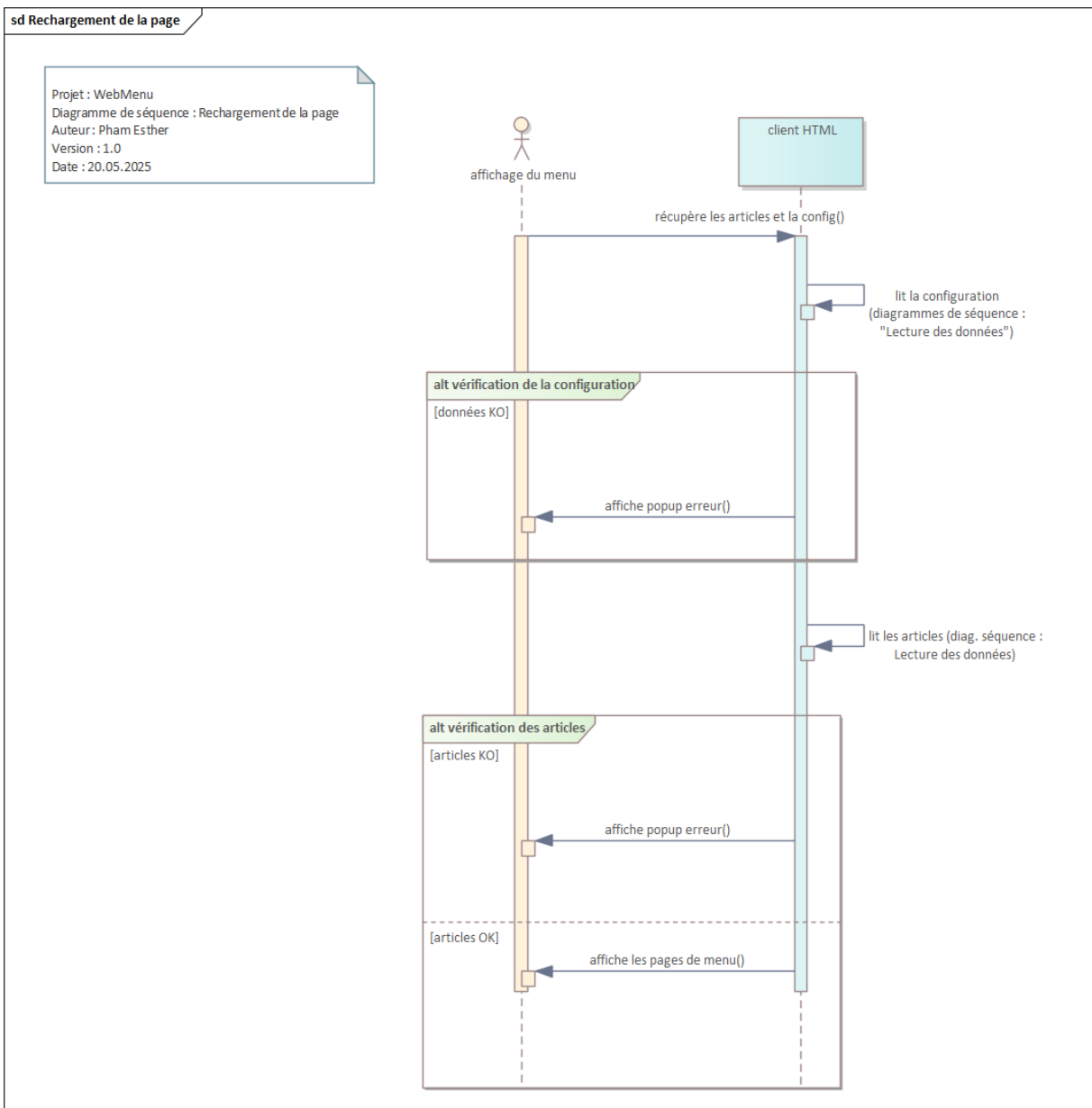


Figure 11 - diagSeq\_rechargePage.png

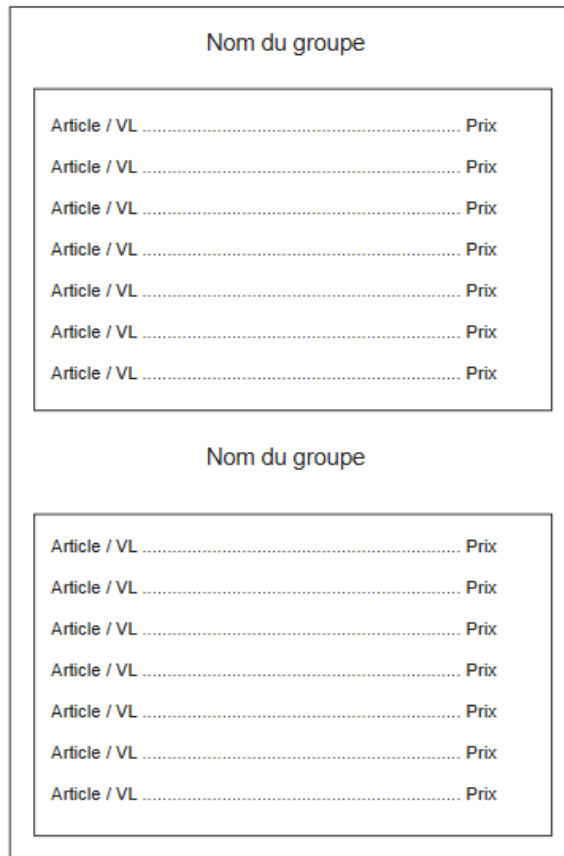
## 2.5 Définition des endpoints

Voici les différents endpoints qui vont être utilisés pour l'application entre le client et le serveur :

Fonction	Méthode	URL
Charger la configuration	<b>GET</b>	http://localhost/webmenu/server/controllers/config.php?action=get
Charger tous les articles par groupes	<b>GET</b>	http://localhost/webmenu/server/controllers/displayMenu.php?action=getArticlesByGroup
Charger la configuration pour la gestion des articles	<b>GET</b>	http://localhost/webmenu/server/controllers/config.php?action=getConf
Charger les groupes pour la gestion des articles	<b>GET</b>	http://localhost/webmenu/server/controllers/manageMenu.php?action=getGroup
Charger les articles pour la gestion des articles	<b>GET</b>	http://localhost/webmenu/server/controllers/manageMenu.php?action=getArticles&id=6
Connexion à la page de gestion des articles	<b>POST</b>	http://localhost/webmenu/server/controllers/config.php
Déconnexion de la page de gestion des articles	<b>POST</b>	http://localhost/webmenu/server/controllers/config.php
Ajout d'un article	<b>PUT</b>	http://localhost/webmenu/server/controllers/manageMenu.php
Modification d'un article	<b>PUT</b>	http://localhost/webmenu/server/controllers/manageMenu.php
Suppression d'un article	<b>DELETE</b>	http://localhost/webmenu/server/controllers/manageMenu.php

## 2.6 Analyse de l'état désiré / maquettes

Le projet final doit permettre d'afficher les pages du menu avec une animation entre chacune d'elles. Voici à quoi devrait ressembler l'affichage des articles du menu. Les maquettes qui vont suivre ne sont pas le résultat final des différentes pages de l'application. Cependant, elles permettent d'avoir un guide pour la réalisation de l'application :



La maquette illustre l'affichage des articles du menu, présentée sous deux versions identiques l'une au-dessus de l'autre. Chaque version est contenue dans un grand rectangle avec une bordure grise. À l'intérieur de ce rectangle, le titre "Nom du groupe" est centré en haut. Juste en dessous, une table à deux colonnes est présentée. La première colonne est intitulée "Article / VL" et la seconde "Prix". La table contient sept lignes de données, chacune avec des points de suspension à la fin de la première colonne. Les titres "Article / VL" et "Prix" sont alignés à gauche et à droite de la table, respectivement.

Article / VL	Prix
.....	
.....	
.....	
.....	
.....	
.....	
.....	





Figure 12 - maq\_affMenu.png

Il y aura également une page pour la gestion des articles. Pour accéder à la page d'administrateur, il faut connaître l'URL, sur cette page il sera possible de modifier les articles, les supprimer ou en ajouter.

Sur cette page se trouvent les icônes de modification et de suppression ainsi qu'un bouton d'ajout pour les articles. Il y a également une liste qui permet de filtrer les articles par groupe auxquels ils appartiennent :

Gestionnaire des articles

Choisir un groupe ▼

Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	<b>SOLDOUT</b>		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		





Ajouter un article

Figure 13 - maq\_indexAdmin.png

Lorsque l'on ajoute un article des champs se rajoute à la fin de la page pour indiquer le nom de l'article, sa quantité (ml/gr) ainsi que le prix de l'article. Il y aura aussi un bouton pour ajouter un article dans la liste :

Gestionnaire des articles

Choisir un groupe ▼

Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	<b>SOLDOUT</b>		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		

Nom de l'article :

Quantité :





Prix :

Figure 14 - maq\_ajoArt.png

Pour modifier un article, la page ressemble à la page d'ajout. Des champs se rajoute en bas de la page pour modifier l'article. Les champs modifiables sont le nom, la quantité (ml/gr), le prix et l'option d'indiqué si l'article est épuisé ou non :

Gestionnaire des articles

Choisir un groupe ▼

Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL ..... <b>SOLDOUT</b> .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		
Article / VL .....	Prix		

Nom de l'article : article
Sauver les modifications

Quantité : quantité

Prix : prix

☒ soldout

Figure 15 - maq\_modifArt.png

## 2.7 Sécurité de l'information et protection des données

Ce projet ne sera pas utilisé plus tard, aucune autre personne ne sera impactée étant donné que ce n'est pas un contrat. Ce projet sert de test uniquement, il y aura quand même HTTPS mis en place et un login pour se connecter à la page de gestion des articles.

## 3 Travaux préparatoires

### 3.1 Librairie d'animation javascript

La bibliothèque choisie pour l'animation entre les pages est **Slick.js**. Elle offre une variété de types d'animations pour les transitions entre les données. Le site officiel de la bibliothèque permet de tester ces différentes animations, tout en fournissant des démonstrations et des extraits de code permettant de les expérimenter facilement.

Voici l'adresse pour la librairie : [slick - the last carousel you'll ever need](https://kencherry.github.io/slick-carousel/slick/slick.js)

Voici un exemple que j'ai réalisé pour tester les animations : toutes les secondes, le menu change, et d'autres informations s'affichent. Ci-dessous, le code HTML illustrant cette démonstration :

Menu		
SPRITE   250 ml	Coca Cola   250 ml	<i>Caramel mou   270 gr</i>
2.-	5.-	5.5.-

```
<div id="container">
  <h1>Menu</h1>
  <div id="menu">
    <table>
      <tr>
        <td>FANTA | 250 ml</td>
        <td>Coca Cola | 250 ml</td>
        <td>Caramel mou | 270 gr</td>
      </tr>
      <tr>
        <td>5.-</td>
        <td>5.-</td>
        <td>5.-</td>
      </tr>
    </table>
    <table>
      <tr>
        <td>Jus d'orange | 250 ml</td>
        <td>Coca Cola | 250 ml</td>
        <td>Caramel mou | 270 gr</td>
      </tr>
      <tr>
        <td>2.-</td>
```



```
<td>5.-</td>
<td>5.5.-</td>
</tr>
</table>
</div>
</div>
```

Pour pouvoir créer une animation slick, il faut tout d'abord que certains script et liens soient indiqué dans la balise **head** de la page html :

```
<link rel="stylesheet" type="text/css" href="slick/slick.css" />
<link rel="stylesheet" type="text/css" href="stylesheets/main.css" />
<script type="text/javascript" src="scripts/jquery-3.7.1.min.js"></script>
<script type="text/javascript" src="slick/slick.min.js"></script>
```

Le fichier qui contiendra l'animation doit aussi y être indiquer :

```
<script type="text/javascript" src="scripts/indexCtrl.js"></script>
```

Dans ce script, c'est là où l'animation va être créé :

```
$(document).ready(function () {
  let menu = $("#menu");

  menu.slick({
    slidesToShow: 1,
    slidesToScroll: 1,
    autoplay: true,
    autoplaySpeed: 1000,
    pauseOnHover: false,
  });

  $.getScript("scripts/services/servicesHttp.js", function () {
    console.log("servicesHttp.js chargé !");
  });
});
```

### 3.1.1 Attributs de l'animation

Attribut	Utilité
<b>slidesToShow</b>	Permet d'indiquer combien d'élément à montrer, dans ce cas combien de table.
<b>slidesToScroll</b>	Permet d'indiquer combien d'élément à passer entre chaque table.

<b>autoplay</b>	Permet d'indiquer que l'animation se joue automatiquement, aucune action n'est requise de la part de l'utilisateur.
<b>autoplaySpeed</b>	Permet d'indiquer après combien de temps les menus changent, dans ce cas toute les 1 seconde.
<b>pauseOnHover</b>	Permet d'indiquer si oui ou non l'animation s'arrête quand la souris se trouve sur les éléments du menu.

## 4 Conception

### 4.1 Exigences du système

La page d’affichage du menu doit être conçue pour s’afficher en mode portrait, permettant ainsi de présenter une colonne d’articles. La mise en page doit être claire et lisible par le public. Le tri des articles par groupe, ainsi que l’organisation des groupes par page, est configurable dans la base de données. Les différentes pages du menu défilent automatiquement les unes après les autres.

L’application de gestion des articles doit offrir une interface permettant d’afficher la liste des articles par groupe, de les modifier, d’en ajouter ou d’en supprimer. Une fois les modifications effectuées dans l’interface administrateur, la page d’affichage du menu doit se mettre à jour automatiquement pour refléter les changements.

### 4.2 Architecture du système

L’application est hébergée sur cPanel sous les liens indiqués dans le schéma. L’application est divisée en 3 parties : le client, le serveur REST et la base de données. Les technologies utilisées sont indiquées ci-dessous :

- Pour la partie client, j’utilise css, javascript et html.
- Pour la partie serveur, j’utilise php.
- Pour la gestion de la base de données, j’utilise mariaDB.

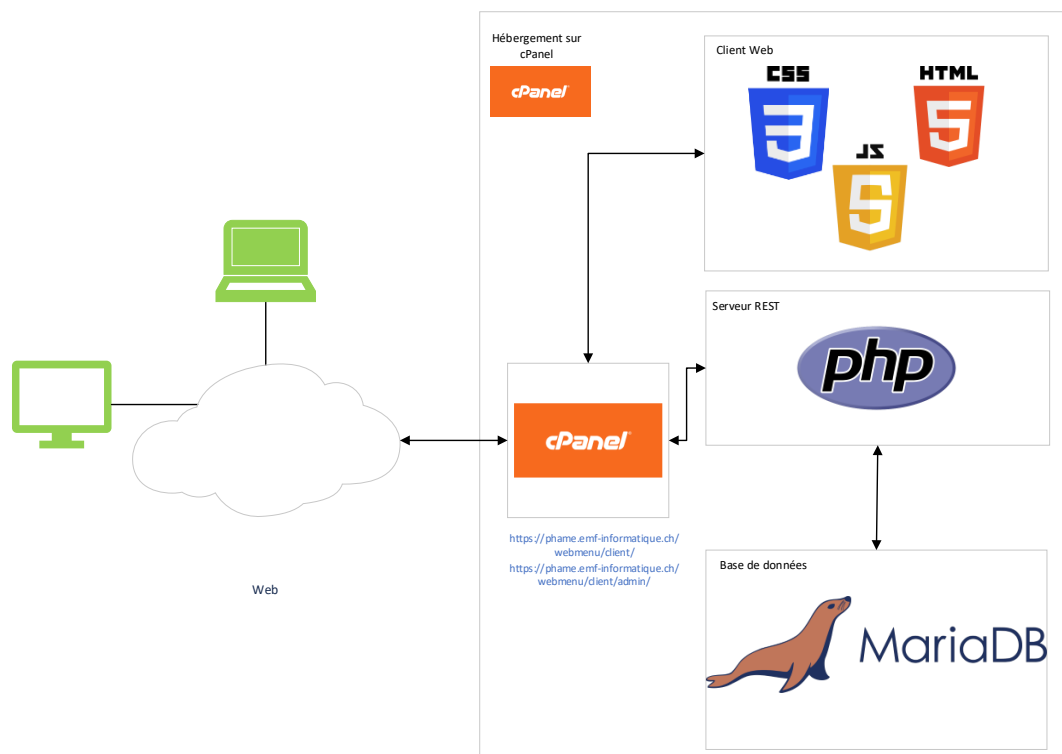


Figure 16 - architecture\_sys.jpg

#### 4.2.1 Base de données

La base de données a été fournie au début du projet, elle n'a pas été créée durant le projet. Voici le modèle de la base de données :

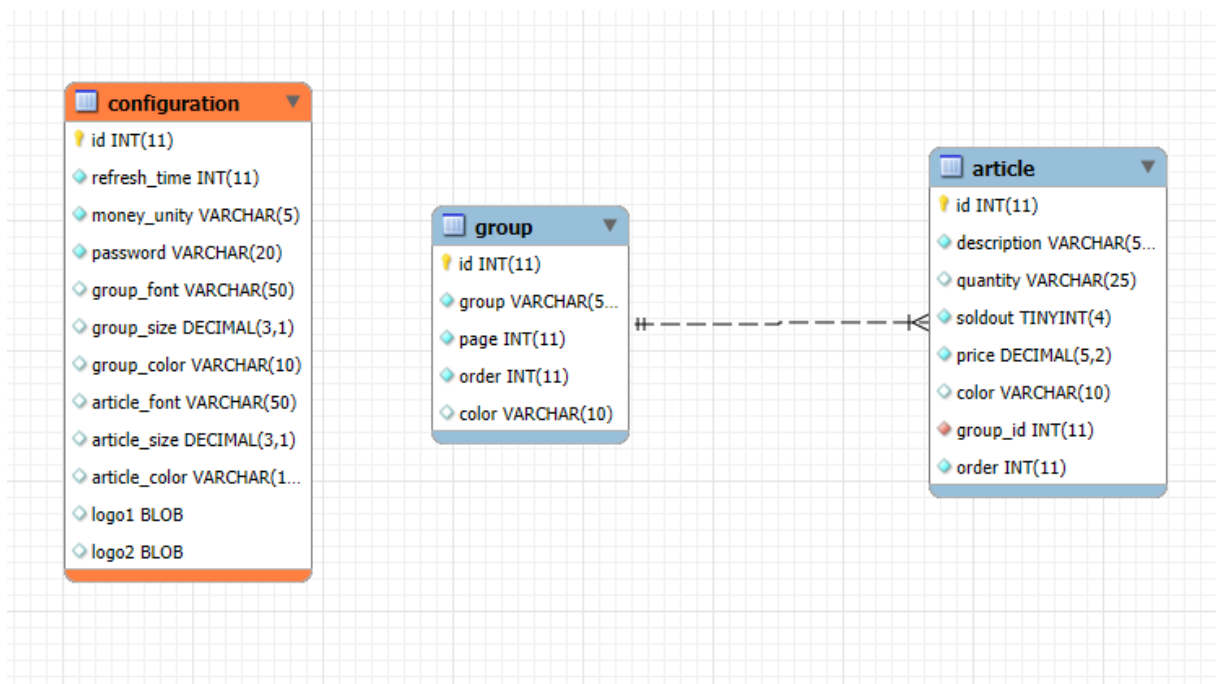


Figure 17 - schemaDB\_original.png

## 4.3 Concept d'implémentation

### 4.3.1 Diagramme de classes du client

Ce diagramme de classe correspond à la structure de la partie client de l'application. Chaque pages html possède un contrôleur, et chaque contrôleur est relié au fichier de service HTTP.

Les couleurs correspondent au modèle MVC. Les classes jaunes correspondent aux View, les classes vertes correspondent aux Controller et les classes bleues correspondent au Worker.

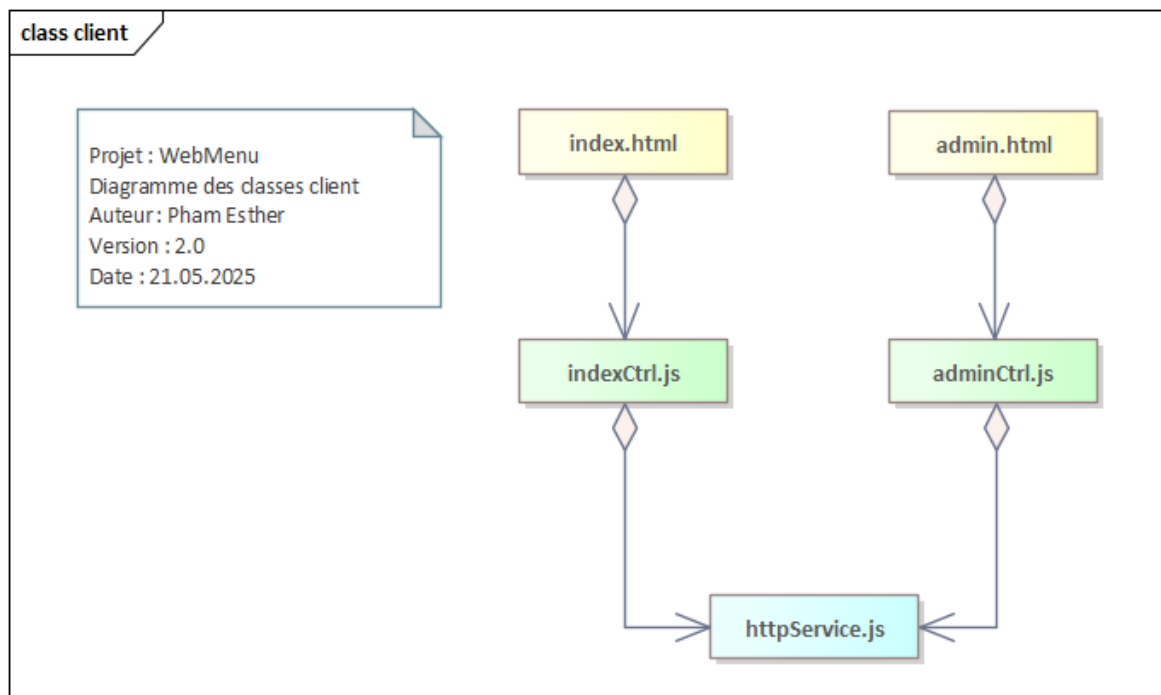


Figure 18 - diagCla\_client.png

Classe	Fonction
index.html	Permet la mise en page de la page d'affichage du menu.
admin.html	Permet la mise en page de la page d'ajout, de suppression et de modification.
indexCtrl.js	Correspond au contrôleur de la page index.html. Ce fichier permet le lien entre index.html et les services http.
adminCtrl.js	Correspond au contrôleur de la page admin.html. Ce fichier permet le lien entre la page admin.html et les services http.

httpService.js	Ce fichier contient toutes les requêtes http communiqué au serveur. C'est aussi dans ce fichier que les codes d'erreurs sont indiqués.
----------------	--

#### 4.3.2 Diagramme de classes du serveur

Dans ce diagramme est indiqué la structure du serveur de l'application. Comme pour le diagramme de classe du client, les couleurs correspondent au modèle MVC.

Les classes vertes correspondent aux Controller, les classes bleus aux Worker et les classes roses sont les Beans de l'application.

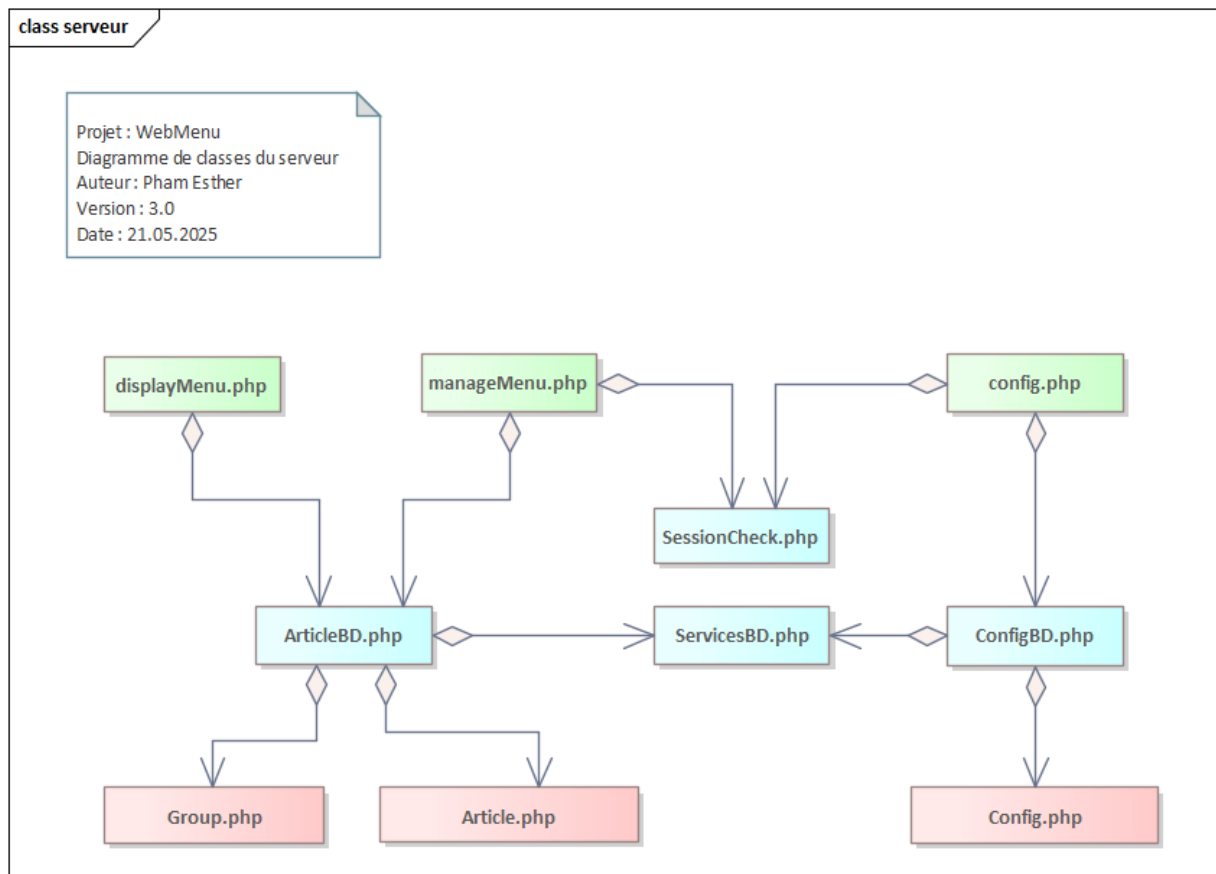


Figure 19 - diagCla\_serveur.png

Fichier	Fonction
displayMenu.php	Ce controller permet de gérer l'affichage des pages du menu. Ce fichier fait le lien entre les services http du client et les worker du serveur.

manageMenu.php	Ce controller gère les différentes requêtes http réalisées à partir de la page admin.
config.php	Ce controller gère les différentes requêtes http sur la table configuration.
SessionCheck.php	Ce worker gère la session, il permet de vérifier si la session est active ou non, de la détruire si besoin et de la créer.
ArticleBD.php	Ce worker fait les différentes requêtes CRUD sur la table article.
ConfigBD.php	Ce worker fait les différentes requêtes CRUD sur la table configuration.
ServicesBD.php	Ce worker permet la gestion de la connexion à la base de données et des requêtes.
Group.php	Ce bean est utilisé pour récupérer les données de la table group.
Article.php	Ce bean est utilisé pour récupérer les données de la table article.
Config.php	Ce bean est utilisé pour récupérer les données de la table configuration.

## 4.4 Concept de tests

Basé sur les différents diagrammes d'activités

Nr.	Objet testé	Description du test	Attente
1	Page d'affichage du menu	Accès aux pages du menu.	Les pages du menu s'affichent.
2	Connexion avec un mot de passe correct	Connexion à la page admin avec un bon mot de passe	La page d'admin s'affiche avec les différentes options de modification, d'ajout et de suppression.
3	Accès à la page admin sans s'authentifier	Accès à la page de gestion des articles sans s'authentifier	Un popup d'erreur s'affiche avec le bon code http.
4	Connexion avec mauvais mot de passe	Saisis d'un mauvais mot de passe	Un popup d'erreur s'affiche.
5	Notification d'information	Erreur lors des différents GET de données	Un popup indiquant que la liste d'article est vide s'affiche
6	Vérification du retour des requêtes ajout/suppression/modification	Erreur lors du retour des données de la requête	Un popup d'erreur s'affiche avec un code http.
7	Vérification des données avant l'envoi de la requête	Les données sont erronées	Un popup d'erreur s'affiche avec un code http.
8	Notification de modification, d'ajout et de suppression	Popup de confirmation de modification	Une notification s'affiche pour confirmer les modifications.
9	Ajout d'une couleur, d'une police d'écriture et d'une taille	Ajout des spécifications de la police dans la BD	Affichage des groupes et des articles selon les spécifications de la BD.
10	Modification des articles et mise à jour du menu	Ajout/modification/suppression d'un article et rechargement automatique de la page d'affichage du menu	La liste des articles modifier se mettent à jour sur la page de la gestion des articles et l'affichage du menu se met à jour.
11	Modification/ajout d'un article avec un ordre invalide	Modification/ajout d'un article avec un ordre plus petit que 1	Un popup d'erreur s'affiche indiquant que l'ordre est invalide



## 5 Réalisation

### 5.1 Utilisation de l'application

Pour accéder à la page d'affichage du menu, l'url est [Menu](#)

Pour accéder à la page de gestion des articles, l'url est [Gestion des articles](#). Le mot de passe pour accéder à la page est **Emf12345**.

### 5.2 Modification de la base de données

Une modification dans la base de données a été réalisé, car malheureusement le nom d'un champ dans la base de données originel posait un problème pour faire une requête sql. Étant donné que le nom du champ était le même que le nom de la table, la requête ne fonctionnait pas.

La table group possédait un champ « group » correspondant au nom du groupe et le nom du champ a été remplacé par « name ».

### 5.3 Design du système

#### 5.3.1 Backend

##### 5.3.1.1 Structure du serveur

La structure du serveur est la même qui est indiqué dans les schémas du chapitre 4.3.2 Diagramme de classe du serveur et le langage utilisé est PHP.

##### 5.3.1.2 Base de données schéma final

La différence entre le schéma de la base de données du chapitre 4.2.1 Plan d'intégration des systèmes est la colonne pour le nom du groupe. Avant, le nom du groupe correspondait à la colonne « group » mais maintenant il correspond à « name ». Ce changement est expliqué dans le chapitre 5.2.1 Modification de la base de données.

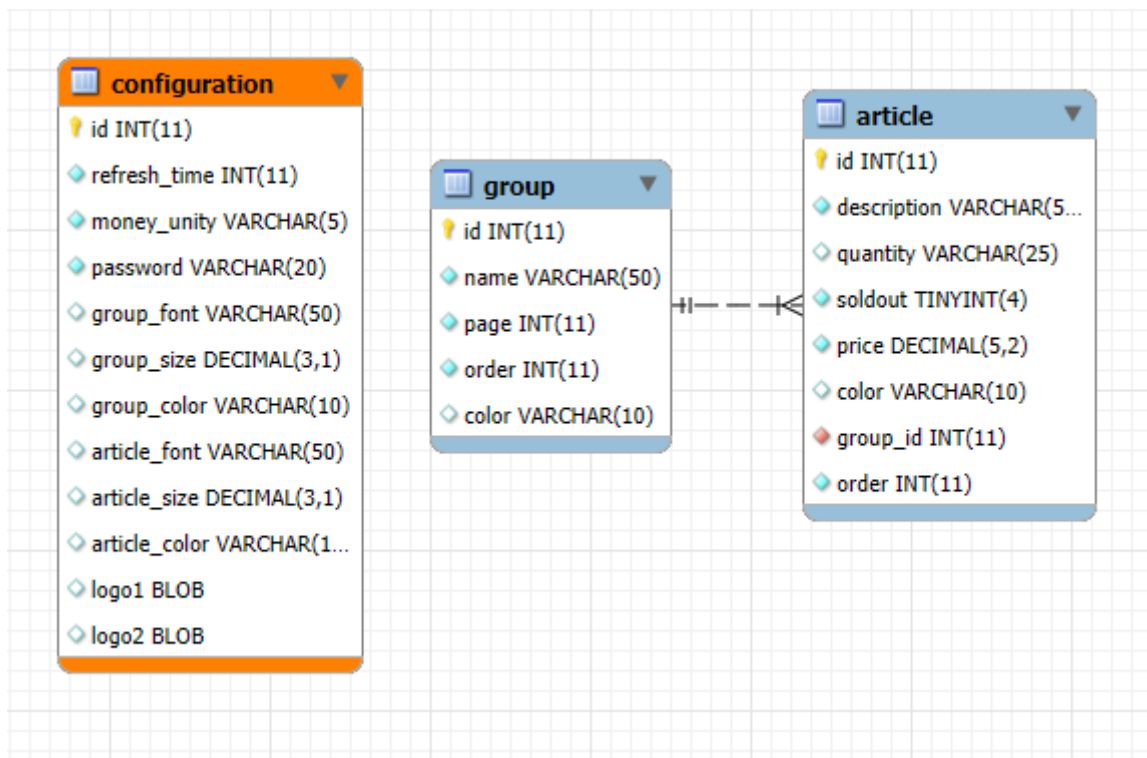


Figure 20 - schemaDB\_final.png

### 5.3.2 Frontend

#### 5.3.2.1 Structure client

La structure du serveur est la même qui est indiqué dans les schémas du chapitre 4.3.1 Diagramme de classe du client et les langages utilisés sont html pour les pages de base, css pour le style des pages et javascript pour les controller ainsi que les requêtes http.

#### 5.3.2.2 Bibliothèques utilisées

Les bibliothèques utilisées sont :

- slick.js : <https://kenwheeler.github.io/slick/>
- jquery.js : <https://jquery.com/download/>
- sweetAlert2.js : <https://sourceforge.net/projects/sweetalert2.mirror/>

## 5.4 Code par rapport aux diagrammes d'activités

Les différents bouts de code qui vont suivre correspondent aux différents diagrammes d'activités.

## 5.4.1 Récupération de la configuration

### 5.4.1.1 Client

Lorsque l'on charge la classe IndexCtrl dans le fichier indexCtrl.js, dans son constructeur, il faut d'abord initialiser la variable http qui sera utilisé pour faire tous les appels au fichier des services http. Une fois la variable http initialisé, on va appeler dans la méthode init(), la requête http correspondant au chargement de la configuration.

```
constructor() {  
  this.http = new HttpService();  
  this.http.centraliserErreurHttp((msg) => this.afficherErreurHttp(msg));  
  this.init();  
}  
  
init() {  
  this.http.chargerConf((data) => this.recuperationConfig(data));  
}
```

Une fois la configuration récupérée, la valeur du temps de rafraichissement de la page est récupérée pour l'utiliser plus tard dans l'animation qui sera expliqué dans le chapitre 5.4.3 Rechargement de la page d'affichage du menu.

```
recuperationConfig(data) {  
  let config = data;  
  this.#rechargementPage = data[0].refreshTime;  
  this.http.chargerArticlesParGroup((data) => this.affichageArticles(config, data));  
}
```

Dans le fichier httpService.js, la méthode chargerConf() permet d'envoyer la requête http au serveur :

```
chargerConf(successCallback) {  
  $.ajax({  
    type: "GET",  
    dataType: "json",  
    url: CONFIG_URL,  
    data: 'action=getConf',  
    success: successCallback  
  });  
}
```

### 5.4.1.2 Serveur

Dans le controller config.php, une vérification de l'action envoyé est faite pour confirmer la requête :

```
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    if ($_GET['action'] == "getConf") {
        $listeConf = new ConfigDB();
        echo $listeConf->getInJson();
    }
}
```

Une fois la requête confirmée, la requête sql est envoyé à la base de données :

```
public function getAll(): array
{
    $sql = "SELECT * FROM configuration";
    $params = array();
    $count = 0;
    $listeConfigs = array();
    $connect = ServicesDB::getInstance();
    $configs = $connect->selectQuery(sql: $sql, params: $params);
    foreach ($configs as $data) {
        $config = new Config(
            id: $data['id'],
            refreshTime: $data['refresh_time'],
            moneyUnity: $data['money_unity'],
            password: $data['password'],
            groupFont: $data['group_font'],
            groupSize: $data['group_size'],
            groupColor: $data['group_color'],
            articleFont: $data['article_font'],
            articleSize: $data['article_size'],
            articleColor: $data['article_color'],
            logo1: $data['logo1'],
            logo2: $data['logo2']
        );
        $listeConfigs[$count++] = $config;
    }
    return $listeConfigs;
}
```

Le résultat de la requête est converti en json, pour pouvoir manipuler les données récupérées plus facilement :

```
public function getInJson(): bool|string
{
    $listeConfigs = $this->getAll();
    $result = [];
    foreach ($listeConfigs as $config) {
        $result[] = json_decode(json: $config->toJson(), associative: true);
    }
    return json_encode(value: $result, flags: JSON_PRETTY_PRINT);
}
```

## 5.4.2 Récupération des articles

### 5.4.2.1 Affichage de la page du menu

#### 5.4.2.1.1 Client

Comme pour la récupération de la configuration, il faut appeler la méthode qui correspond à l'appel http dans le fichier httpService.js :

```
chargerArticlesParGroup(successCallback) {  
    $.ajax({  
        type: "GET",  
        dataType: "json",  
        url: DISPLAY_URL,  
        data: 'action=getArticlesByGroup',  
        success: successCallback  
    });  
}
```

#### 5.4.2.1.2 Serveur

displayMenu.php

Dans le fichier correspondant à DISPLAY\_URL, on va vérifier si l'action est la bonne. Une fois la vérification faite, on va appeler la méthode pour faire la requête sql :

```
if (isset($_SERVER['REQUEST_METHOD'])) {  
    if ($_SERVER['REQUEST_METHOD'] == 'GET') {  
        if ($_GET['action'] == "getArticlesByGroup") {  
            $listeArticles = new ArticleDB();  
            echo $listeArticles->getGroupedArticlesJSON();  
        }  
    }  
}
```

ArticleDB.php

La requête sql pour récupérer les articles par groupe est assez conséquente. La requête contient des inner join pour récupérer le nom du groupe de l'article, la couleur de la police du groupe, l'ordre du groupe dans les pages et l'ordre des pages :

```
$sql = "SELECT
    menu_display.article.id,
    menu_display.article.description,
    menu_display.article.quantity,
    menu_display.article.soldout,
    menu_display.article.price,
    menu_display.article.color AS couleurArticle,
    menu_display.group.name AS nomGroup,
    menu_display.group.color AS couleurGroup,
    menu_display.article.order AS ordreArticle,
    menu_display.group.page AS ordrePage,
    menu_display.group.order AS ordreGroup
FROM
    menu_display.article
INNER JOIN
    menu_display.group
ON
    menu_display.article.group_id = menu_display.group.id
ORDER BY
    ordrePage, ordreGroup, ordreArticle";
```

Aucuns paramètres, ne doit être indiqué car c'est un select. Ensuite, il faut créer une instance aux fichiers qui gère la connexion à la base de données. L'instance est utilisée pour faire la requête sql à la base de données.

```
$params = array();
$connect = ServicesDB::getInstance();
$articles = $connect->selectQuery(sql: $sql, params: $params);
```

Un fois la requête faites, il faut mettre en forme pour grouper les articles par groupe et les groupes par page.

Tout d'abord, un tableau qui sera utiliser pour regrouper les articles par groupe est créé. Ensuite, pour chaque article récupéré, on va récupérer le nom du groupe et on va indiquer que le nom du groupe est un nouveau tableau en vérifiant d'abord si le groupe existe déjà.

Dans le nouveau tableau du groupe, les informations de l'ordre de la page du groupe, de l'ordre du groupe sur la page et la couleur de la police du groupe est indiqué. Un tableau d'articles est également créé pour pouvoir y mettre la liste d'article qui sont associés au groupe :

```
$groupedArticles = [];
foreach ($articles as $data) {
    $groupName = $data['nomGroup'];

    if (!isset($groupedArticles[$groupName])) {
        $groupedArticles[$groupName] = [
            'ordrePage' => $data['ordrePage'],
            'ordreGroup' => $data['ordreGroup'],
            'couleurGroup' => $data['couleurGroup'],
            'articles' => []
        ];
    }
}
```

On crée l'article avec les data récupérés :

```
$article = new Article(
    id: $data['id'],
    description: $data['description'],
    quantity: $data['quantity'],
    soldout: $data['soldout'],
    price: $data['price'],
    color: $data['couleurArticle'],
    group: $data['nomGroup'],
    order: $data['ordreArticle']
);
```

On le rajoute dans la liste d'articles correspondant associés au groupe :

```
$groupedArticles[$groupName]['articles'][] = $article;
```

Une fois les articles associés à leur groupe respectif, il faut les mettre en forme pour rendre un tableau json en indiquant quels groupes correspond à quelle liste d'articles etc.

En premier, on récupère le tableau crée dans la méthode précédente et on va parcourir le tableau et transformer chacun des articles en tableau associatif json. :

```
public function getGroupedArticlesJSON(): bool|string
{
    $groupedArticles = $this->getAllByGroup();
    foreach ($groupedArticles as $groupName => $groupData) {
        $groupData['articles'] = array_map(callback: function ($article): mixed {
            return json_decode(json: $article->toJSON(), associative: true);
        }, array: $groupData['articles']);
    }
}
```

Il faut ensuite remettre les articles dans le tableau principal :

```
$groupedArticles[$groupName] = $groupData;
```

Finalement, le retour de la méthode est le tableau principal en json pour pouvoir lire plus facilement le retour du tableau :

```
return json_encode(value: $groupedArticles, flags: JSON_PRETTY_PRINT);
```

## 5.4.2.2 Gestion des articles

### 5.4.2.2.1 Client

Pour la gestion des articles du côté client, l'appel http est légèrement différent car l'appel se fait à chaque fois qu'un groupe est choisi dans le select de la page html :

Alcool fort  
 Bières  
 Boissons sans alcool

C'est pour cela que l'appel est différent dans le fichier httpService.js, l'id du groupe est donné comme paramètre :

```
chargerArticlesGestion(id, successCallback) {
  $.ajax({
    type: "GET",
    dataType: "json",
    url: MANAGE_URL,
    data: {
      action: 'getArticles',
      id: id
    },
    success: successCallback
  });
}
```

#### 5.4.2.2.2 Serveur

ArticleDB.php

La requête sql pour récupérer les articles par rapport au groupe se fait grâce à l'id donné en paramètre de la méthode. Dans le select le :id permet d'indiquer que cet élément correspond au paramètre fournit dans le tableau params :

```
public function getArticleByGroup($groupId): array
{
  $sql = "SELECT * FROM article where group_id = :id order by menu_display.article.order";
  $params = ['id' => $groupId];
```

Création du tableau retourné à la fin de la méthode et de l'index du tableau d'articles. :

```
$count = 0;
$listeArticles = array();
```

Connexion à la base de données et utilisation de cette connexion pour exécuter la requête sql :

```
$connect = ServicesDB::getInstance();
$articles = $connect->selectQuery(sql: $sql, params: $params);
```

Parcours des données reçues de la requête pour créer l'article et le mettre dans le tableau d'articles. Le tableau d'article est retourné un fois qu'il est rempli :



```
foreach ($articles as $data) {
    $article = new Article(
        id: $data['id'],
        description: $data['description'],
        quantity: $data['quantity'],
        soldout: $data['soldout'],
        price: $data['price'],
        color: $data['color'],
        group: $data['group_id'],
        order: $data['order']
    );
    $listeArticles[$count++] = $article;
}
return $listeArticles;
```

Finalement, il faut mettre en forme pour retourner un tableau json associatif de la liste d'articles récupéré précédemment :

```
foreach ($listeArticles as $article) {
    $result[] = json_decode(json: $article->toJson(), associative: true);
}
```

### 5.4.3 Rechargement de la page d'affichage du menu

IndexCtrl.js

Pour vérifier à chaque fin d'animation si les données des articles ont changé, il y a un timeout à la dernière slide qui permet de recharger la méthode d'initialisation de la classe :

```
}).on('afterChange', (event, slick, currentSlide) => {
    if (currentSlide === slick.slideCount - 1) {
        setTimeout(() => {
            this.init();
        }, this.#rechargementPage);
    }
});
```

Pour recharger la page de l'affichage du menu, j'utilise une empreinte MD5 qui me permet de tester si les articles récupérés sont différents des précédents. L'empreinte est créée par rapport aux articles récupérés :

```
affichageArticles(config, data) {
    let md5 = MD5(JSON.stringify(data));
    if (md5 === this.#empreinteMD5) {
        return;
    }
    this.#empreinteMD5 = md5;
```

#### 5.4.4 Modification / ajout d'un article

##### 5.4.4.1 Client

httpService.js

Pour la modification et l'ajout d'un article, la requête http envoyé est un PUT et elle contient les différents paramètres utiles à la requête sql :

```
modifierArticle(id, description, order, soldout, successCallback) {  
  $.ajax({  
    type: "PUT",  
    url: MANAGE_URL,  
    contentType: "application/json",  
    data: JSON.stringify({  
      action: 'modifArticle',  
      id: id,  
      description: description,  
      order: order,  
      soldout: soldout  
    }),  
    success: successCallback  
  });  
}
```

##### 5.4.4.2 Serveur

manageMenu.php

Tout d'abord on vérifie si la méthode http est un PUT :

```
if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
```

En php, la variable \$\_PUT[] n'existe pas alors il faut donner les différentes informations à traiter en json lorsque l'on envoie la requête http depuis le client. Ensuite dans le serveur on décode les données envoyées en tableau associatif :

```
$data = json_decode(json: file_get_contents(filename: "php://input"), associative: true);
```

On vérifie si l'action est la modification et on instancie la variable de modification de l'article. On utilise la variable qui vient d'être instanciée pour envoyer les paramètres utiles à la requête sql et récupérer le résultat de la requête :

```
elseif ($data['action'] === 'modifArticle') {
    $ajoutArticle = new ArticleDB();
    $result = $ajoutArticle->editArticle(
        id: $data['id'],
        description: $data['description'],
        order: $data['order'],
        soldout: $data['soldout']
    );

    if (isset($result['result']) && $result['result'] === 'true') {
        echo json_encode(value: $result);
    } else {
        http_response_code(response_code: 503);
    }
}
```

ArticleDB.php

Pour éviter les injections sql, on va échapper les caractères spéciaux du paramètre description :

```
$escapedDescription = htmlspecialchars(string: $description, flags: ENT_QUOTES, encoding: 'UTF-8');
```

Création de la requête sql avec les paramètres récupérés du client :

```
$sql = "UPDATE menu_display.article
    SET description = :description, menu_display.article.order = :order, soldout = :soldout
    WHERE id = :id";

$params = [
    'id' => $id,
    'description' => $escapedDescription,
    'order' => $order,
    'soldout' => $soldout
];
```

Création d'une instance pour la connexion à la base de données et exécution de la requête de modification :

```
$connect = ServicesDB::getInstance();
$resultat = $connect->executeQuery(sql: $sql, params: $params);
```

Test du résultat pour vérifier si la modification a été correctement exécutée :

```
if ($resultat && $resultat->rowCount() > 0) {
    return ['result' => 'true'];
} else {
    return ['result' => 'false', 'message' => 'No rows inserted'];
}
```

#### 5.4.4.2.1 Ajout

## manageMenu.php

L'ajout d'un article est aussi une méthode http PUT alors le test pour savoir quel méthode http la requête correspond est le même que pour la modification. Les données utilisées sont aussi décodées en un tableau associatif.

Comme pour la modification d'un article, on vérifie si l'action est l'ajout d'un article. On instancie la variable d'ajout de l'article. Une utilise la variable qui vient d'être instancier pour envoyer les paramètres utiles à la requête sql et récupéré le résultat de la requête :

```
if (isset($data['action']) && $data['action'] === 'ajoutArticle') {
    $ajoutArticle = new ArticleDB();
    $result = $ajoutArticle->addArticle(
        description: $data['description'],
        quantite: $data['quantite'],
        prix: $data['prix'],
        groupe: $data['groupe'],
        ordre: $data['ordre']
    );

    if (isset($result['result']) && $result['result'] === 'true') {
        echo json_encode(value: $result);
    } else {
        http_response_code(response_code: 503);
    }
}
```

## ArticleDB.php

Pour l'ajout, dans la requête, 2 paramètres de type string sont données alors il faut échapper ces 2 paramètres pour éviter les injections sql :

```
$escapedDescription = htmlspecialchars(string: $description, flags: ENT_QUOTES, encoding: 'UTF-8');
$escapedQuantite = htmlspecialchars(string: $quantite, flags: ENT_QUOTES, encoding: 'UTF-8');
```

Création de la requête d'insertion de l'article avec les différents paramètres utiles :

```
$sql = "INSERT INTO menu_display.article (description, quantity, price, group_id, menu_display.article.order)
      values (:description, :quantite, :prix, :groupe, :ordre)";

$params = [
    'description' => $escapedDescription,
    'quantite' => $escapedQuantite,
    'prix' => $prix,
    'groupe' => $groupe,
    'ordre' => $ordre
];
```

À la fin, il faut également tester le retour de la requête. Le test est le même que pour la modification.

## 5.4.5 Suppression d'un article

### 5.4.5.1 Client

httpService.js

Pour la suppression d'un article, la méthode http est un DELETE et il faut renvoyer l'id de l'article à supprimer :

```
supprimerArticle(id, successCallback) {
  $.ajax({
    type: "DELETE",
    url: MANAGE_URL,
    contentType: "application/json",
    data: JSON.stringify({
      action: 'suppArticle',
      id: id
    }),
    success: successCallback
  });
}
```

### 5.4.5.2 Serveur

manageMenu.php

Tout comme les requêtes PUT, il faut vérifier si la méthode http envoyé est de type DELETE. La variable \$\_DELETE[] n'existe pas non plus dans php, c'est pourquoi il faut également utiliser un tableau associatif des données envoyé :

```
if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
  $data = json_decode(json: file_get_contents(filename: "php://input"), associative: true);
```

Ensuite, il faut tester la valeur de l'action si elle correspond à la suppression d'un article. On crée l'instance qui va être utiliser pour la requête de suppression de l'article dans la base de données :

```
if (isset($data['action']) && $data['action'] === 'suppArticle') {
  $suppArticle = new ArticleDB();
  $result = $suppArticle->deleteArticle(id: $data['id']);
```

Le test du retour de la requête est le même que pour l'ajout et la modification d'un article

ArticleDB.php

Création de la requête par rapport à l'id donnée en paramètre :

```
$sql = "DELETE FROM menu_display.article WHERE id = :id";

$params = [
    'id' => $id
];
```

On crée une instance qui permet la connexion à la base de données et on exécute la requête :

```
$connect = ServicesDB::getInstance();
$resultat = $connect->executeQuery(sql: $sql, params: $params);
```

Test du résultat de la requête et renvoie d'une réponse :

```
if ($resultat && $resultat->rowCount() > 0) {
    return ['result' => 'true'];
} else {
    return ['result' => 'false', 'message' => 'No rows inserted'];
}
```

## 6 Test

### 6.1 Procédure de test

Tous les tests ont été définis dans le chapitre 4.4 Concept de tests, durant la phase conception du projet. Les tests sont réalisés depuis 2 ordinateurs sous Windows 11 avec un écran d'affichage en mode portrait.

- Pour la partie cliente les tests sont réalisés sur Microsoft Edge.
- Pour la partie serveur les tests sont réalisés sur Postman.

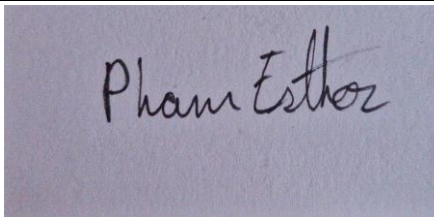
### 6.2 Protocol de test

Nr.	Objet testé	Description du test	Attente	Résultat	Visa
1	Page d'affichage du menu	Accès aux pages du menu.	Les pages du menu s'affichent.	OK	P.E.
2	Connexion avec un mot de passe correct	Connexion à la page admin avec un bon mot de passe	La page d'admin s'affiche avec les différentes options de modification, d'ajout et de suppression.	OK	P.E.
3	Accès à la page admin sans s'authentifier	Accès à la page de gestion des articles sans s'authentifier	Un popup d'erreur s'affiche avec le bon code http.	OK	P.E.
4	Connexion avec mauvais mot de passe	Saisie d'un mauvais mot de passe	Un popup d'erreur s'affiche.	OK	P.E.



5	Notification d'information	Erreur lors des différents GET de données	Un popup indiquant que la liste d'article est vide s'affiche	OK	P.E.
6	Vérification du retour des requêtes ajout/suppression/modification	Erreur lors du retour des données de la requête	Un popup d'erreur s'affiche avec un code http.	OK	P.E.
7	Vérification des données avant l'envoi de la requête	Les données sont erronées	Un popup d'erreur s'affiche avec un code http.	OK	P.E.
8	Notification de modification, d'ajout et de suppression	Popup de confirmation de modification	Une notification s'affiche pour confirmer les modifications.	OK	P.E.
9	Ajout d'une couleur, d'une police d'écriture et d'une taille	Ajout des spécifications de la police dans la BD	Affichage des groupes et des articles selon les spécifications de la BD.	OK	P.E.
10	Modification des articles et mise à jour du menu	Ajout/modification/suppression d'un article et rechargement automatique de la page d'affichage du menu	La liste des articles modifier se mettent à jour sur la page de la gestion des articles et l'affichage du menu se met à jour.	OK	P.E.
11	Modification/ajout d'un article avec un ordre invalide	Modification/ajout d'un article avec un ordre plus petit que 1	Un popup d'erreur s'affiche indiquant que l'ordre est invalide	OK	P.E.

### 6.3 Signature du protocole de test

Date	Nom	Signature
30.05.2025	Pham	



## 7 Conclusion

Les objectifs principaux du projet ont été atteints et sont pleinement fonctionnels. L'affichage du menu, ainsi que les animations entre chaque menu, fonctionnent parfaitement. Lorsqu'un article est modifié, ajouté ou supprimé, l'affichage du menu se met à jour automatiquement pour refléter les nouvelles informations.

Tout au long de ce projet, j'ai utilisé ChatGPT pour m'assister dans la résolution des différents problèmes rencontrés et pour commenter le code. Je l'ai également employé pour corriger l'orthographe et reformuler les phrases afin d'assurer une lecture plus fluide et agréable de mon rapport.

### 7.1 Améliorations possibles

Comme indiqué dans le cahier des charges, il serait souhaitable de pouvoir modifier d'autres éléments du menu, comme la gestion des groupes, via une interface web plutôt que directement dans la base de données. Malheureusement, faute de temps, je n'ai pas pu implémenter cette fonctionnalité. Cependant, cela constituerait une amélioration future intéressante, permettant d'offrir une expérience utilisateur plus fluide et conviviale.

### 7.2 Auto-évaluation

Je pense que mon projet aurait pu être plus abouti si j'avais consacré davantage de temps à sa réalisation, notamment pour la documentation. Cela constitue un point d'amélioration personnelle sur lequel je dois travailler afin d'éviter que ce type de situation ne se reproduise à l'avenir.

Ce projet m'a permis d'apprendre beaucoup sur moi-même, tant sur le plan de la gestion du travail que sur ma capacité à acquérir de nouvelles compétences de manière autonome. J'ai constaté que je manque parfois de persévérance lorsqu'il s'agit de résoudre un problème ou de mener des recherches approfondies.

Cela dit, j'ai apprécié l'opportunité de travailler sur un projet où j'étais responsable de la gestion de mon temps de manière optimale. Cette expérience m'a fait prendre conscience de l'importance de m'améliorer dans certains domaines, tant personnels que professionnels.

## 8 Bibliographie: liste des sources et références

Librairie d'animation utilisé slick.js : <https://kenwheeler.github.io/slick/>

Librairie utile pour le développement web jquery.js : <https://jquery.com/download/>

Librairie des popups sweetAlert2.js : <https://sourceforge.net/projects/sweetalert2.mirror/>

Icon js utilisé dans le schéma pour la structure du système : [JavaScript Logo, symbol, meaning, history, PNG, brand](#)

Icon css utilisé dans le schéma pour la structure du système : [Fichier:CSS3 logo and word-mark.svg — Wikipédia](#)

Icon html utilisé dans le schéma pour la structure du système : [File:HTML5 logo and word-mark.svg - Wikipedia](#)

Icon php utilisé dans le schéma pour la structure du système : [PHP: Download Logos](#)

Icon mariaDB utilisé dans le schéma pour la structure du système : [File:MariaDB colour logo.svg - Wikimedia Commons](#)

cPanel utilisé pour l'hébergement de l'application : [Identifiant cPanel](#)

Postman : [Postman API Platform](#)

WAMP : [Wampserver - Fichiers et Addons](#)

MySQL Workbench : [MySQL](#)

PHP : [PHP: Hypertext Preprocessor](#)

VSCode : [Visual Studio Code - Code Editing. Redefined](#)

Microsoft Edge : [Découvrez Microsoft Edge](#)

Github Desktop : [GitHub Desktop | Simple collaboration from your desktop](#)

Dépôt Github : [estpha/webmenu: projet final pour ma dernière année](#)

W3School utilisé pour la mise en forme des pages web : [W3Schools Online Web Tutorials](#)

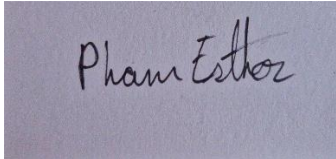
## 9 Glossaire

Terme	Signification
<b>backend</b>	Partie du développement d'une application qui gère la logique serveur, les bases de données et l'interaction avec le frontend.
<b>BD/DB</b>	<i>Base de Données</i> (Database) : Système de gestion d'informations structurées pour stocker, récupérer et manipuler des données.
<b>ChatGPT</b>	Modèle de langage développé par OpenAI pour comprendre et générer du texte, utilisé pour des conversations automatisées.
<b>controller</b>	En architecture MVC (Model-View-Controller), c'est la partie qui gère les entrées utilisateurs et les interactions avec le modèle et la vue.
<b>crud</b>	<i>Create, Read, Update, Delete</i> : Les quatre opérations de base pour manipuler des données dans une base de données.
<b>css</b>	<i>Cascading Style Sheets</i> : Langage pour décrire l'apparence des documents HTML (mise en page, couleurs, polices, etc.).
<b>frontend</b>	Partie d'une application qui concerne l'interface visible et l'expérience utilisateur (UI/UX).
<b>get</b>	Méthode HTTP pour récupérer des données d'un serveur, souvent utilisée dans les requêtes API.
<b>github</b>	Plateforme de développement pour héberger et gérer des projets de code source utilisant le système de versionnage Git.
<b>html</b>	<i>HyperText Markup Language</i> : Langage de balisage pour structurer le contenu d'une page web.
<b>https</b>	<i>HyperText Transfer Protocol Secure</i> : Version sécurisée du HTTP, utilisant le chiffrement SSL/TLS pour protéger les données.
<b>id</b>	Identifiant unique d'un élément dans un document HTML, utilisé pour le cibler dans le CSS ou JavaScript.
<b>javascript</b>	Langage de programmation pour créer des interactions dynamiques sur les pages web.
<b>json</b>	<i>JavaScript Object Notation</i> : Format léger pour échanger des données entre serveurs et applications web.
<b>onedrive</b>	Service de stockage en ligne de Microsoft permettant de sauvegarder, synchroniser et partager des fichiers dans le cloud.
<b>php</b>	<i>Hypertext Preprocessor</i> : Langage de programmation côté serveur pour créer des pages web dynamiques et interagir avec les bases de données.
<b>TPI</b>	<i>Travail Personnel Individuel</i> : Projet ou tâche individuelle souvent utilisé dans le cadre éducatif.

<b>view</b>	En architecture MVC, la vue (view) est responsable de l'affichage des données à l'utilisateur, correspondant à l'interface utilisateur.
<b>wamp</b>	<i>Windows, Apache, MySQL, PHP</i> : Environnement de développement pour configurer une machine Windows pour héberger des applications web.
<b>worker</b>	Un <i>web worker</i> est un script JavaScript exécuté en arrière-plan, permettant d'effectuer des tâches sans bloquer l'interface utilisateur.

## 10 Signatures

Je soussigné déclare que les informations contenues dans ce rapport de travail pratique individuel rendu ce jour le 02.06.2025 dans le cadre de la procédure de qualification de mon CFC d'informaticienne, ne sont pas plagiées. Toutes les informations de sources extérieures ainsi que les informations fournies par des tiers durant le déroulement du travail sont consignées.

Date	Nom	Signature
02.06.2025	Pham	

## 11 Annexes

- Le cahier des charges [/]
- Le journal de travail [/]
- Le planning [/]
- Les diagrammes de l'application et le schéma de la BD au format PNG/JPG  
[/images\_documentation]
- Le code source du projet [/code\_source]
- Le dump SQL de la BD avec les valeurs tests [/]
- Le modèle final et original de la BD au format MWB [/fichiers\_originaux]
- Les diagrammes de l'application au format QEA/VSDX[/fichiers\_originaux]