Projet web « piscines municipales de Rennes »

Compte-rendu d'activité

Sommaire

1. Contexte et cahier des charges

Cet atelier professionnel est un travail collectif réalisé en deuxième année de BTS. Le client fictif, « les piscines municipales de Rennes », souhaite moderniser son site web pour y vendre des tickets donnant accès aux différentes piscines de la ville. Le cahier des charges reçu est le suivant :

« Les piscines municipales de Rennes vont changer de logiciel afin de générer directement des codes permettant d'accéder aux bassins. Ces codes pourront être achetés via le site internet de la piscine, en ligne, sans aucune inscription. Nous ne traiterons pas les problématiques de paiement en ligne, mais ce processus devra être simulé.

Le site permettra de générer des codes pour obtenir une leçon individuelle de nage sur un des créneaux libres ou un abonnement solo 10 entrées valable 10 mois ou un abonnement duo valable pour dix fois deux entrées simultanées, pendant un an. Vous pourrez envisager d'autres formules également.

En effet, le tarif, le nombre de places, la durée de validité des titres en vente pourront être définis en ligne par des administrateurs authentifiés. Vous chercherez à proposer une configuration la plus générique/modulable possible.

Vous aurez également à définir un formulaire permettant à n'importe qui de tester la validité d'un code et de voir son contenu actuel. »

Il n'y avait pas d'existant, et le projet devait donc être créé à partir de zéro.

D'après le cahier des charges, nous avons imaginé les pages nécessaires au minimum dans notre application pour respecter le cahier des charges :

- une page permettant l'obtention d'un code,
- une page permettant d'entrer un code pour vérifier ses informations,
- une page permettant de voir ses réservations en cours et d'en ajouter,
- une page permettant à l'administrateur de s'authentifier pour modifier les offres.

Parallèlement, nous avons élaboré le diagramme de cas d'utilisation qui suit, réalisé sur diagrams.net (anciennement draw.io) :

Projet web « Piscines municipales de Rennes » - Diagramme de cas d'utilisation

Notes :
- le cas « définir/modifier une offre » englobe la définition initiale et la modification d'une offre, et est dépendant d'un certain nombre de cas (valider un tarif…). On admet cependant que, lors de la modification, tous les paramètres d'une offre ne sont pas nécessairement modifiés par l'administrateur, qui se contente donc de les « valider » en laissant leur valeur telle quelle.
- on admet également que cette liste de cas n'est pas exhaustive. En effet, l'administrateur peut également définir/modifier le bassin ou le professeur associé à une séance individuelle, gérer les créneaux disponibles, voire ajouter une piscine ou un professeur...

2. Préparation et planification

2.1. Environnement de travail

Le client ayant demandé un site web, nous savions que nous allions devoir travailler en HTML et CSS, éventuellement JavaScript. Par ailleurs, PHP et MySQL étaient imposés.

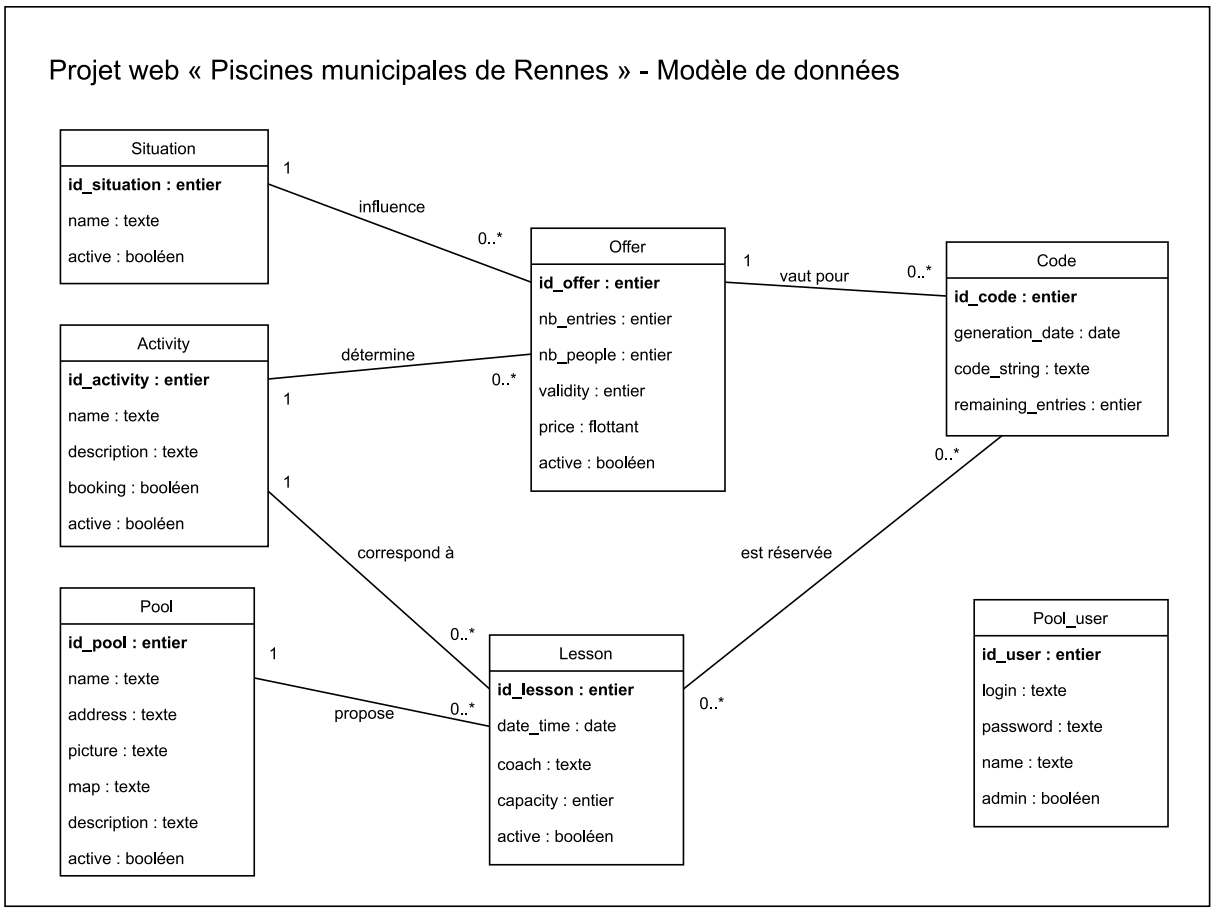
Un environnement de développement web devait être installé pour simuler un serveur web localement. Pour Windows, WampServer est une des possibilités, et c'est celle que j'ai choisi. Comme tout outil WAMP, il comprend un serveur Apache, un système de gestion de base de données MySQL/MariaDB, et un serveur PHP.

Pour ce qui est de l'éditeur de code, nous avons collectivement choisi Visual Studio Code.

L'installation détaillée des différents éléments de notre environnement de développement est expliquée dans un fichier annexe.

2.2. Modèle de données

Après avoir défini les cas d'utilisation à partir du cahier des charges, nous avons pu réfléchir aux données dont nous aurions besoin dans notre base de données, et avons conçu le modèle de données suivant :

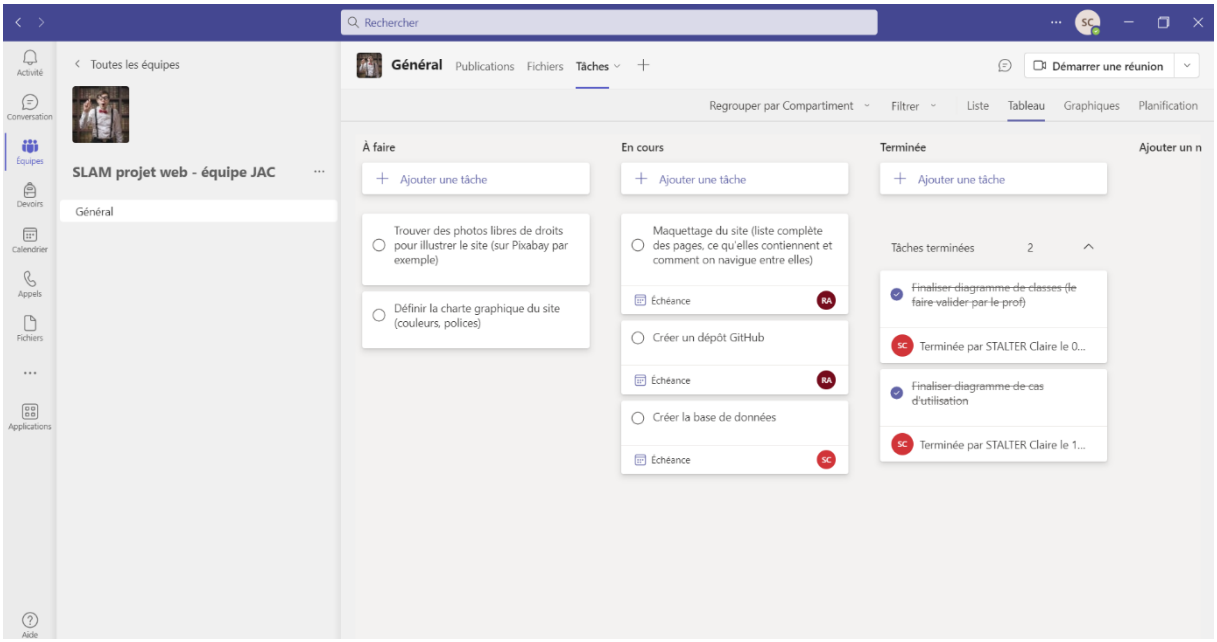


L'idée est qu'une formule soit définie selon une activité (cours de natation débutant, aquagym, bébé nageurs…) et une situation (adulte, moins de 18 ans…). Un code, qui donne accès aux piscines, est associé à une formule et une seule (mais qui peut comporter plusieurs entrées). Les séances, liées aux piscines, appartiennent également à une activité. La table Utilisateur permettra l'authentification de l'administrateur.

Ce diagrammes terminé, nous avons pu nous lancer dans la planification des tâches.

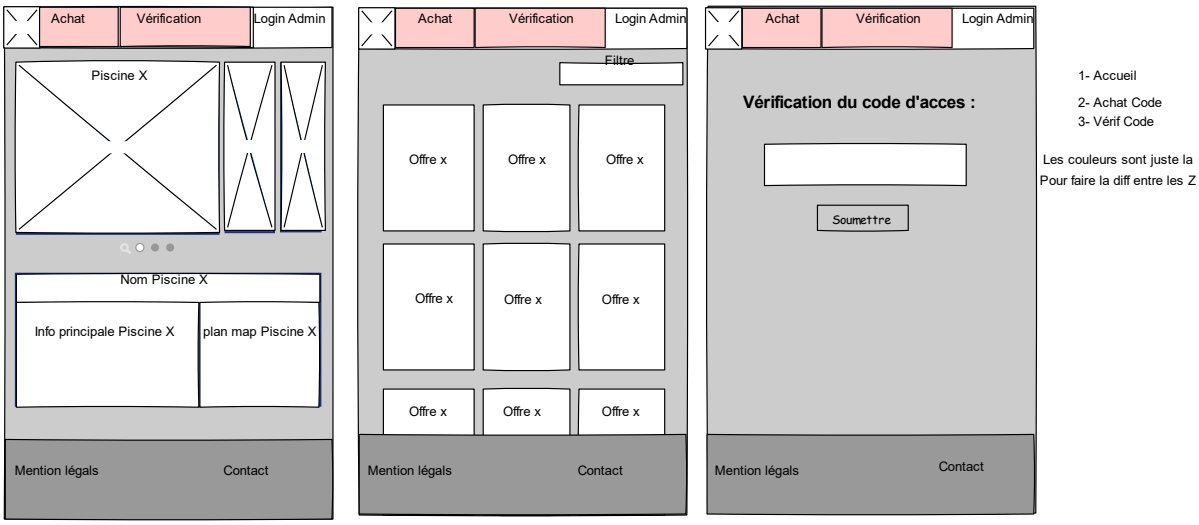
2.3. Organisation du projet

Dès le lancement du projet, nous avons décidé que nous aurions besoin d'un espace de travail collaboratif, et nous avons mis en place un groupe Teams. On peut y démarrer des conversations, organiser les fichiers liés au projet pour pouvoir les retrouver plus tard, mais aussi définir des tâches.



Par ailleurs, un dépôt GitHub de travail a été créé par l'un des membres de l'équipe, pour nous permettre de partager facilement notre code. Il se trouve à l'adresse https://github.com/DroVz/SitePiscineRennes.

Une ébauche de maquette du site a également été réalisée, avec le logiciel Pencil, pour nous donner une idée de ce qu'on chercherait à obtenir.



3. Réalisation

Cette partie fait d'abord un point sur l'ensemble des fonctionnalités du projet, puis détaille les plus emblématiques, celles qui ont nécessité une réflexion particulière.

3.1. Missions prévues et réalisées

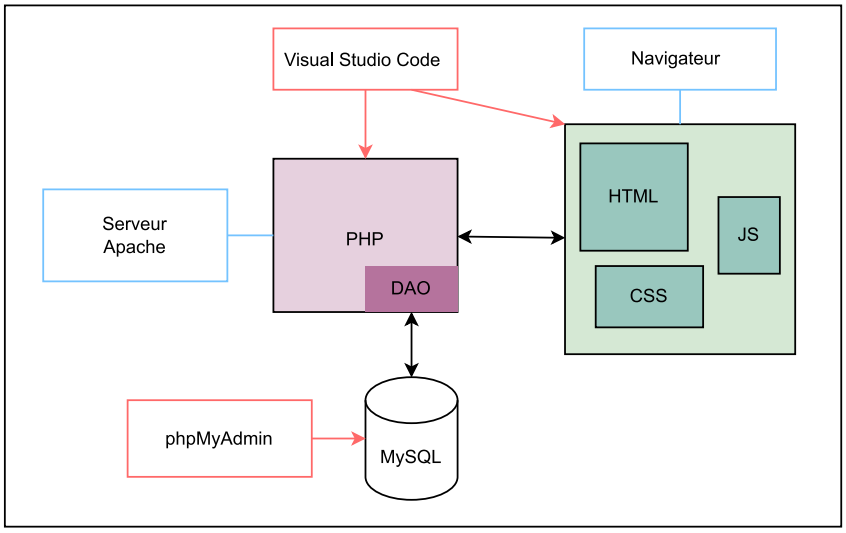
Dès le début, nous avons imaginé différentes fonctionnalités, certaines plus critiques que d'autres. Le tableau ci-dessous dresse le bilan des fonctionnalités développées (case marquée d'une croix), partiellement développées (case marquée d'un point d'interrogation) et abandonnées (case laissée vide).

Réalisé ?	
GÉNÉRAL	
- barre de navigation pour accéder aux pages achat/vérification de code, panier	x
- lien pour que l'administrateur puisse accéder au formulaire de connexion	x
- mot de passe de l'administrateur stocké chiffré dans la base	x
- les champs permettant d'entrer du texte ont une sécurité pour éviter les injections SQL	?
ACHAT DE CODE	
- possibilité de choisir une activité et une situation (adulte, moins de 18 ans…)	x
- les bonnes formules sont proposées en fonction des activité et situation choisies	x
- possibilité d'ajouter une formule au panier	x
- possibilité d'ajouter plusieurs fois la même formule au panier	
- possibilité d'ajouter plusieurs formules différentes au panier	x
- après ajout d'une formule au panier, possibilité de visualiser le panier ou bien de retourner à la sélection de la formule	x
- le panier de l'utilisateur est associé à la session, donc conservé tout au long de la navigation	x
- le panier affiche clairement le prix et nombre de chaque article	x
- le panier affiche clairement le prix total	x
- depuis le panier, possibilité de supprimer une formule	x

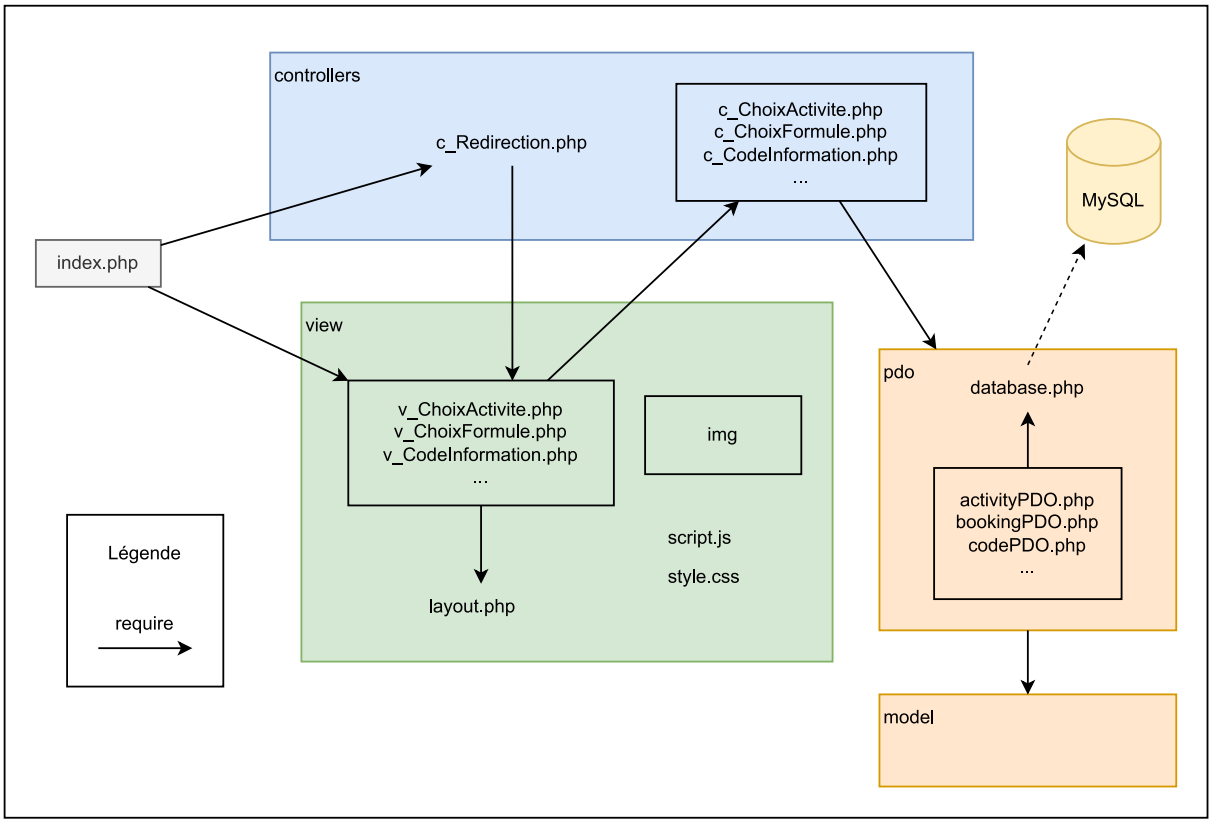
- depuis le panier, possibilité d'incrémenter une formule	
- simulation simple d'une page de paiement	x
- pour chaque formule, un code unique est généré	x
- après validation du paiement, les codes obtenus sont affichés dans le navigateur	x
- l'utilisateur peut exporter les codes au format PDF ou autre pour les conserver	
- une fois le(s) code(s) obtenu, le panier est vidé	x
VÉRIFICATION DE CODE	
- possibilité d'entrer un code pour en vérifier la validité	x
- si le code n'existe pas, cela est clairement indiqué	x
- si le code existe, ses informations sont affichées (formule, date d'achat, date de validité, entrées restantes…)	x
- si le code existe, on voit d'un coup d'œil s'il est encore valide ou non	
- si le code est valide et que l'activité nécessite une réservation, un bouton « Gérer mes réservations » est affiché	x
- possibilité de voir les séances pour lesquelles on a déjà une réservation	
- possibilité de réserver pour une séance, à condition que le code soit valide et que le maximum de réservations possible ne soit pas atteint	
- possibilité d'annuler une réservation, à condition que la date ne soit pas passée de plus de X heures/jours	
- possibilité de voir les séances disponibles correspondant à l'activité	
- possibilité de filtrer les séances en fonction de la piscine	
- pour chaque séance, on voit tout de suite s'il reste des places disponibles	
PARTIE ADMINISTRATEUR	
- possibilité de s'identifier pour accéder à la page de gestion	x
- possibilité d'ajouter une nouvelle formule	
- possibilité de modifier une formule	
- possibilité de désactiver une formule	
- les formules modifiées ou supprimées restent dans la base	

3.2. Architecture

La structure du projet, avec ses langages et son environnement, peut être schématisée comme suit :



Au sein des fichiers sources, le modèle MVC a été respecté, avec une séparation stricte entre les vues (affichage des données), le modèle (classes métier) et les contrôleurs (seuls habilités à interroger la base de données et à effectuer des traitements sur les données). L'architecture finale est la suivante :



Le dossier racine du site contient ainsi :

- un fichier *index.php*, point d'entrée du site, qui renvoie selon les cas vers le routeur *c_Redirection.php* ou directement vers une vue,
- un dossier *view*, qui contient les vues principale (*layout.php*) et secondaires, un fichier de styles CSS, un fichier JavaScript, et les images utilisées sur le site,
- un dossier *controllers*, qui contient des classes PHP chargées de récupérer les informations nécessaires aux vues plus le contrôleur de redirection *c_Redirection.php*,
- un dossier *pdo*, qui contient un fichier par classe entité et dont le rôle est d'interroger la base de données, plus un fichier *database.php* qui assure la connexion à la base de données,
- un dossier *model*, qui contient les classes entités correspondant aux tables de la base de données.

3.3. Panier

Pour comprendre le fonctionnement du panier, voyons le processus à partir du moment où l'utilisateur clique sur le bouton pour y ajouter une formule.

La page de sélection d'une formule comporte un formulaire avec un bouton « Ajouter au panier », formulaire dont le paramètre action vaut « index.php?action=panierRedirection&step=add ». Au clic sur ce bouton, le fichier *index.php* redirige la requête vers le routeur *c_Redirection.php*, avec en paramètres *action=panierRedirection* et *step=add*.

Le routeur appelle la vue correspondante, *v_PanierAjout.php*, qui elle-même importe son propre contrôleur, *c_PanierAjout.php*. C'est ce contrôleur qui s'occupe des traitements de données : il vérifie d'abord si la formule passée existe (à l'aide de sa propre fonction checkOfferExists). Puis, s'il n'existe pas encore, il crée un attribut nommé « cart » dans la variable de session $_SESSION de PHP, conservée tout au long de la navigation de l'utilisateur sur le site. Enfin, il vérifie si le numéro de formule à ajouter existe comme index de l'attribut *cart*. Si ce n'est pas le cas, il est créé puis passé à 1 (intentionnellement, une même formule ne peut pas être ajoutée plusieurs fois au panier).

```
1 reference | 0 overrides
function addToCart()
{
    if ($this->checkOfferExists()) {
        if (!isset($_SESSION['cart'])) {
            $_SESSION['cart'] = array();
        }
        if (!isset($_SESSION['cart'][$_POST["formule"]])) {
            $_SESSION['cart'][$_POST["formule"]] = 0;
        }
        $_SESSION['cart'][$_POST["formule"]] = 1;
```

Dans le panier, un bouton permet ensuite de supprimer une formule du panier. Au clic, *index.php* renvoie vers le routeur, qui appelle la vue *v_PanierSuppression.php*. Son contrôleur *c_PanierSuppression.php* utilise la fonction PHP *unset* pour simplement supprimer l'index correspondant à la formuler à enlever du panier.

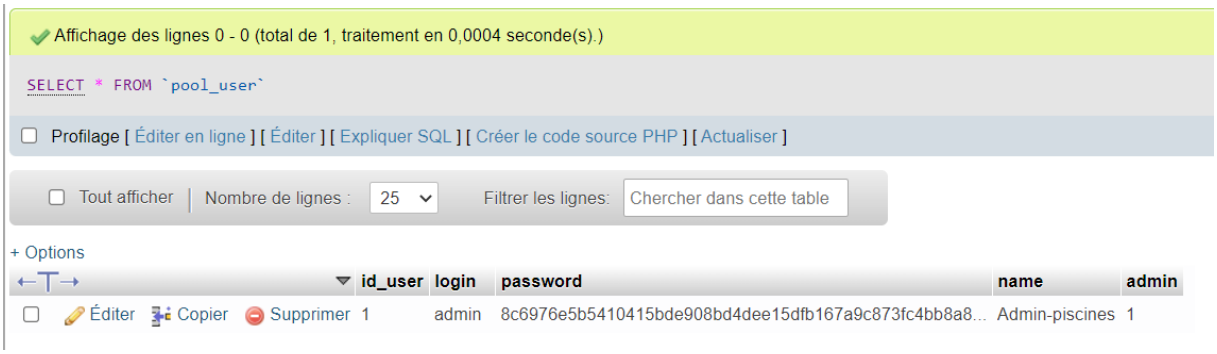
```
1 reference | 0 implementations
class PanierSuppression
{
    1 reference | 0 overrides
    function __construct()
    {}

    1 reference | 0 overrides
    function removeFromCart()
    {
        unset($_SESSION['cart'][$_POST["formule"]]);
        echo '<p>Le produit a été supprimé du panier</br><a href="index.php?ac
```

Une fois la simulation de paiement effectuée, c'est tout l'attribut *cart* de la globale $_SESSION qui est vidé par la fonction *emptyCart()* du contrôleur

```php
1 reference | 0 overrides
public function emptyCart(): void
{
    unset($_SESSION['cart']);
}
```

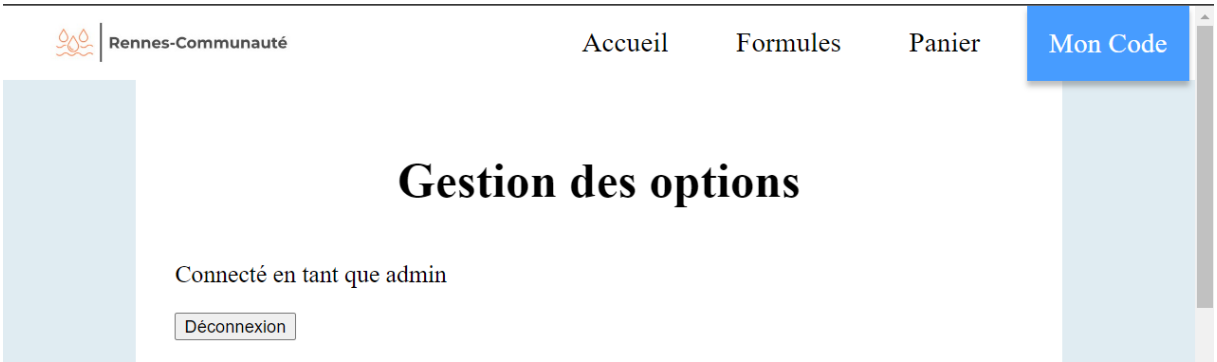
3.4. Partie administrateur

Tout d'abord, l'administrateur doit s'identifier pour accéder à la page de gestion. Son mot de passe est stocké chiffré (avec l'algorithme SHA-256) dans la table *pool_user*.



Une fois connecté, l'administrateur aura parfois besoin de changer de page : pour ajouter une activité, modifier une situation, etc. Pour que son statut de connexion soit conservé pendant toute la navigation, on utilise là encore la variable $_SESSION pour mémoriser son identifiant.

À tout moment, il peut se déconnecter.



Un autre élément important au sujet de la partie *back-office* du site concerne la suppression des activités, situations et formules présentes dans la base de données. Étant donné que des codes valables peuvent être liés à telle ou telle formule (et donc activité et situation), il est impensable de réellement supprimer ces éléments de la base de données. Pour cette raison, il n'y a pas de fonction *delete* dans les fichiers DAO de ces entités.

De la même façon, modifier une activité, situation ou formule doit se faire avec précaution. Il est inimaginable de modifier après coup le nombre d'entrées, le prix ou même l'intitulé d'une formule, alors que des codes valides peuvent y être associés. Nous n'avons donc pas non plus de fonction *update* dans les fichiers DAO. Lorsqu'on modifie une formule, la formule existante doit rester dans la

base mais être désactivée (les utilisateurs désirant acheter un code ne la verront plus dans les choix), tandis que la version modifiée doit être créée dans la base et activée par défaut.

Idéalement, on pourrait aller plus loin : une activité, situation ou formule devrait pouvoir être réellement supprimée ou modifiée dans la base, à la condition qu'aucun code valide n'y soit associé. Actuellement, toute suppression ou modification directe est impossible.

4. Documentation

Un guide d'installation du projet est fourni en annexe.

Le site final est hébergé à l'adresse piscines.clairedev.fr, et la section administrateur est accessible avec les identifiants « admin / PoivrEnGr1 ».

Mes fichiers sources finaux sont accessibles à l'adresse https://github.com/Cerlia/PiscinesRennes