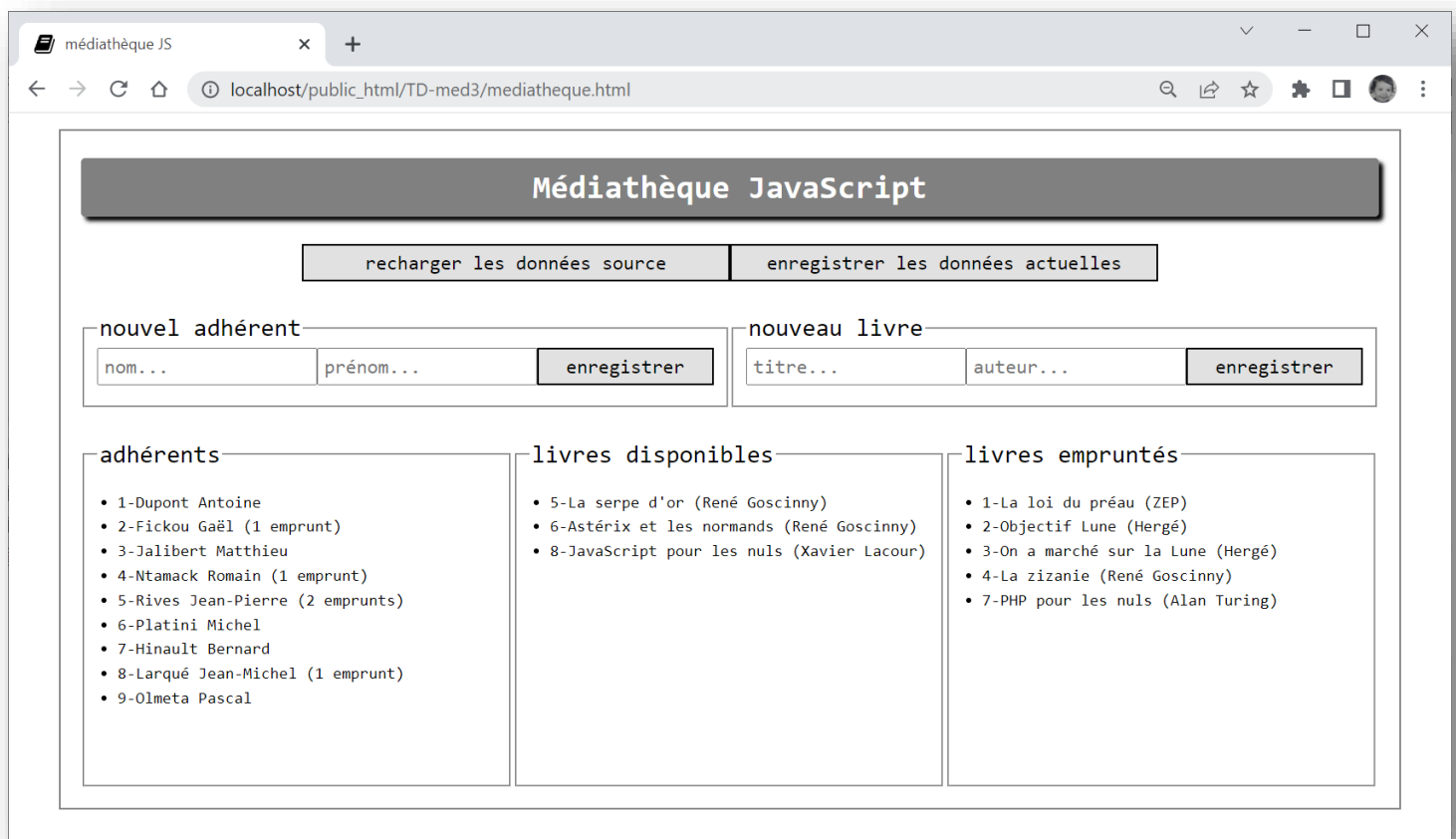


TP4 – Médiathèque JavaScript

INTRODUCTION

Dans cette évaluation, on programme le fonctionnement d'une médiathèque JavaScript. Ci-dessous une capture d'écran du produit fini :




Capture 1 – médiathèque JavaScript


Description des données de la base MySQL

Cette interface fonctionne en lien avec deux tables MySQL :

- Une table des adhérents :

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut
numAdherent 	int(11)			Non	<i>Aucun(e)</i>
nom	varchar(50)	utf8_general_ci		Non	<i>Aucun(e)</i>
prenom	varchar(50)	utf8_general_ci		Non	<i>Aucun(e)</i>

- Une table des livres :

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut
numLivre 	int(11)			Non	<i>Aucun(e)</i>
titre	varchar(100)	utf8_general_ci		Non	<i>Aucun(e)</i>
auteur	varchar(50)	utf8_general_ci		Non	<i>Aucun(e)</i>
numEmprunteur	int(11)			Oui	<i>NULL</i>
estEmprunte	tinyint(1)			Non	0

Un fichier donnees.sql est fourni pour avoir un jeu de données initial.

Vous noterez que par défaut, l'attribut numEmprunteur est à NULL, cet attribut fait référence au numAdherent de l'éventuel emprunteur.

Vous noterez aussi que le champ estEmprunte est un booléen (0 ou 1) :

- 0 signifie que le livre n'est pas emprunté (et dans ce cas numEmprunteur est à NULL),
- 1 signifie que le livre est emprunté (et dans ce cas numEmprunteur désigne l'emprunteur).

Description des fonctionnalités de l'interface

La médiathèque a un fonctionnement dicté par JavaScript : tous les changements se font sans modification de la base de données, mais vont modifier les listes et donc l'affichage.

Seuls deux boutons font fonctionner la médiathèque avec la base de données :

- Le bouton « enregistrer les données actuelles », qui permet de sauvegarder l'état actuel de la médiathèque.
- Le bouton « recharger les données source » qui permet, sans tenir compte des changements qui sont apparus dans les colonnes de l'interface, de revenir aux données de la base.

Au chargement de la page, les données MySQL sont récupérées. Elles permettent de remplir les trois colonnes :

- La colonne de gauche contient tous les adhérents (nom et prénom). L'information est complétée par le nombre d'emprunts en cours.
- La colonne centrale contient les livres (titre et auteur) qui sont disponibles à l'emprunt (donc non empruntés).
- La colonne de droite contient la liste des livres (titre et auteur) en cours d'emprunt.

Dans ces trois colonnes, les items des listes sont cliquables :

- Si on clique sur un adhérent, une popup nous donne la liste de ses emprunts actuels.
- Si on clique sur un livre disponible, une popup demande quel adhérent souhaite emprunter le livre en question.
- Si on clique sur un livre emprunté, une popup nous annonce qui emprunte actuellement ce livre, et nous demande si ce livre est rendu par l'emprunteur. Si oui, alors il est sorti de la liste des livres empruntés, retourne dans la liste des livres disponibles, et l'adhérent est mis à jour dans la colonne de gauche (son nombre d'emprunts).

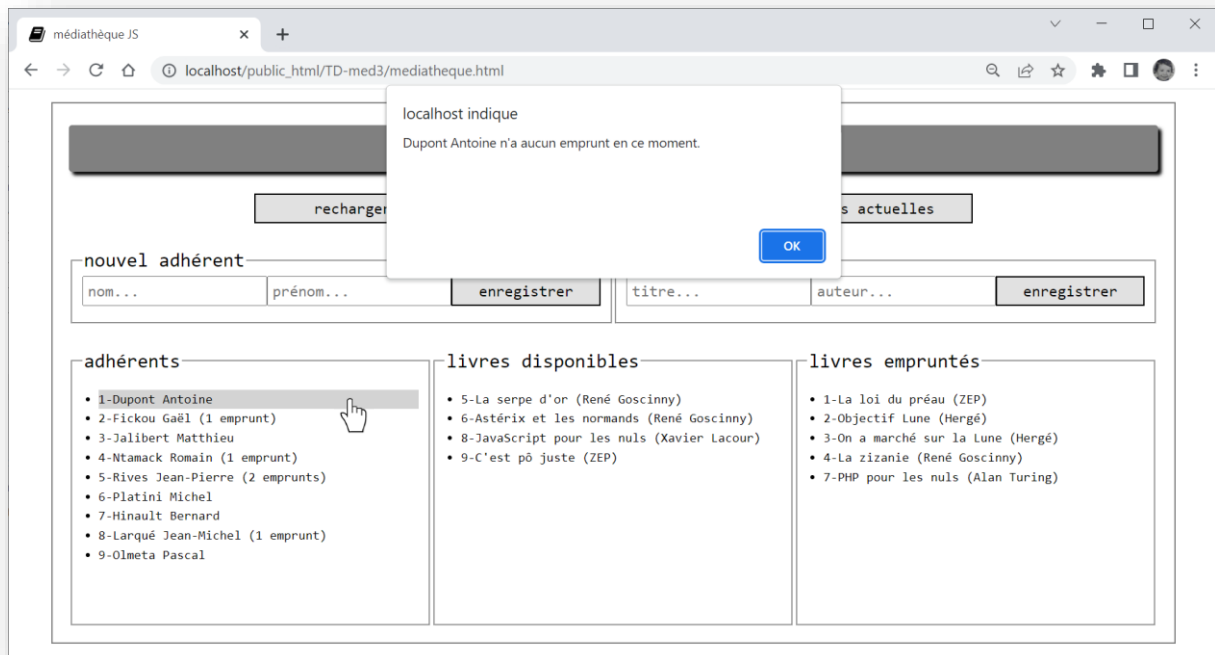
Il y a deux autres zones, qui servent à enregistrer de nouveaux adhérents et de nouveaux livres. Pour un nouvel adhérent ou un nouveau livre, une fois les deux champs de saisie remplis, le bouton « enregistrer » permet d'insérer le nouvel objet :

- Dans la colonne de gauche si c'est un nouvel adhérent,
- Dans la colonne centrale si c'est un nouveau livre.

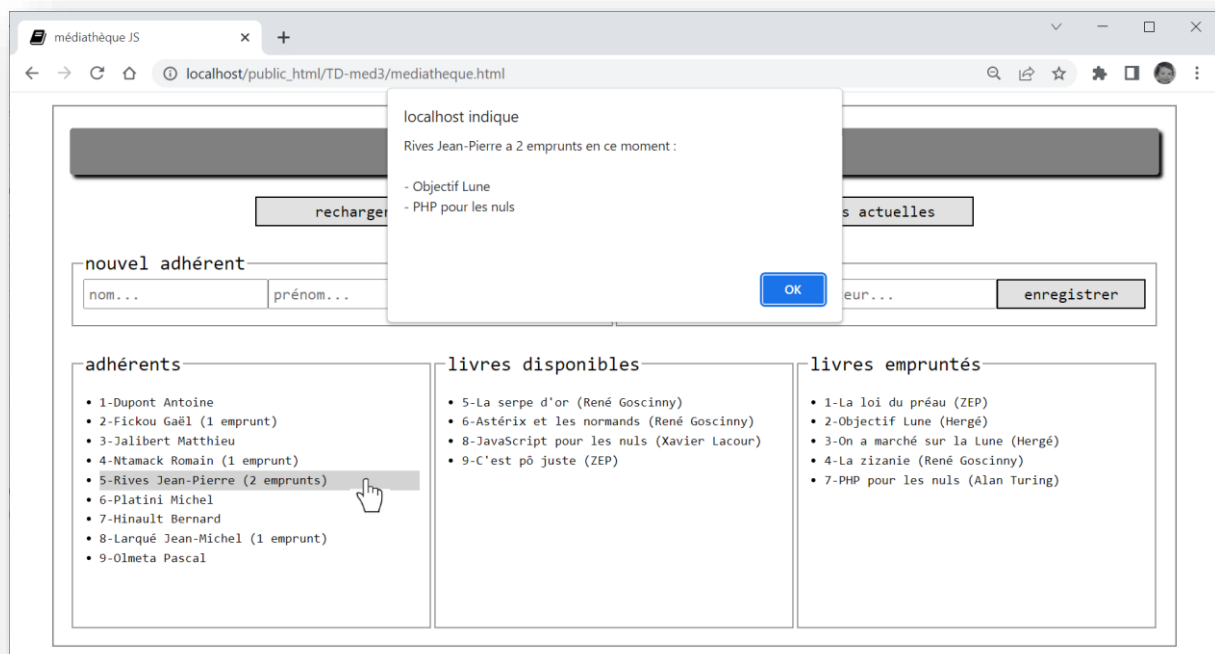
Bien sûr, la sauvegarde sur la base enregistrera les nouveaux adhérents et livres, et de même le rechargement des données source les efface s'il n'y a pas eu au préalable une sauvegarde.

Tout changement des données initiales (emprunt, retour, création d'adhérent ou de livre) provoque un changement d'aspect du bouton « enregistrer les données actuelles » pour rappeler à l'utilisateur le décalage existant.

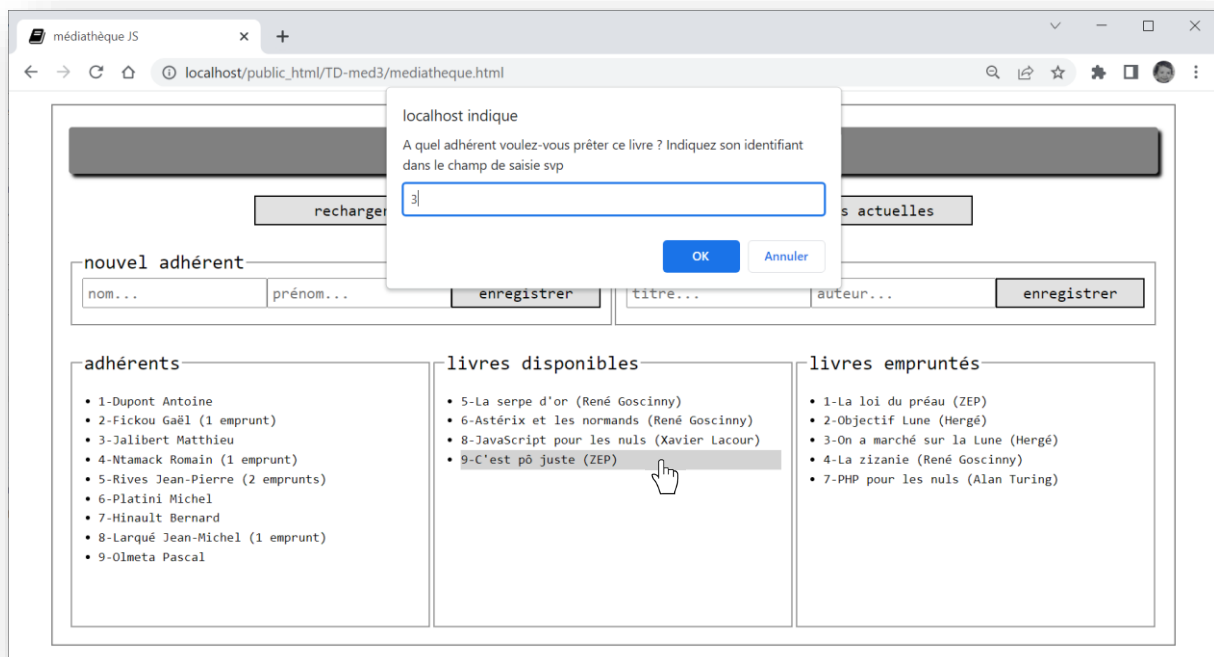
Quelques captures d'écran



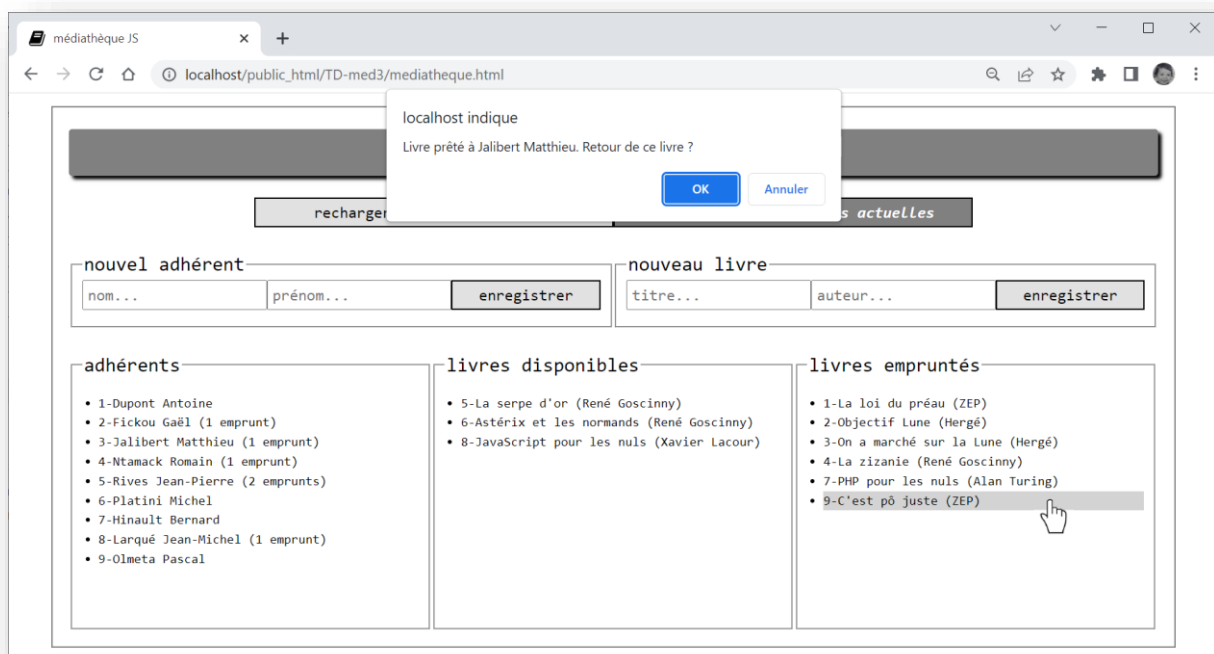
Capture 2 - clic sur un adhérent sans emprunt



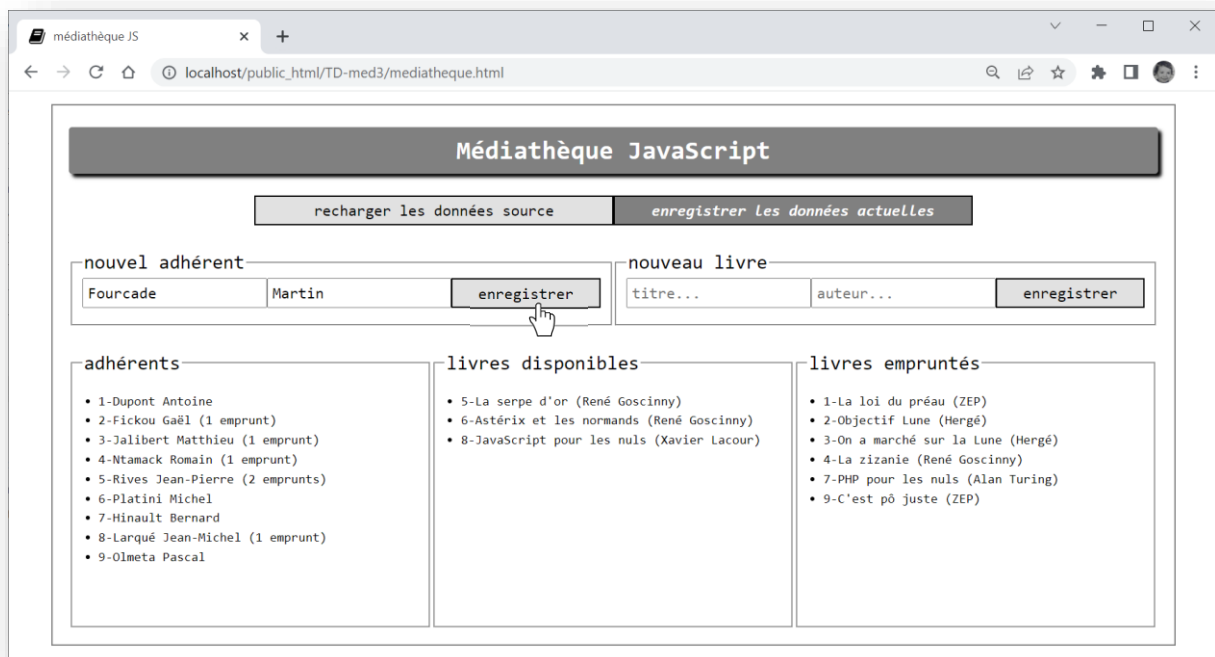
Capture 3 - clic sur un adhérent avec des emprunts



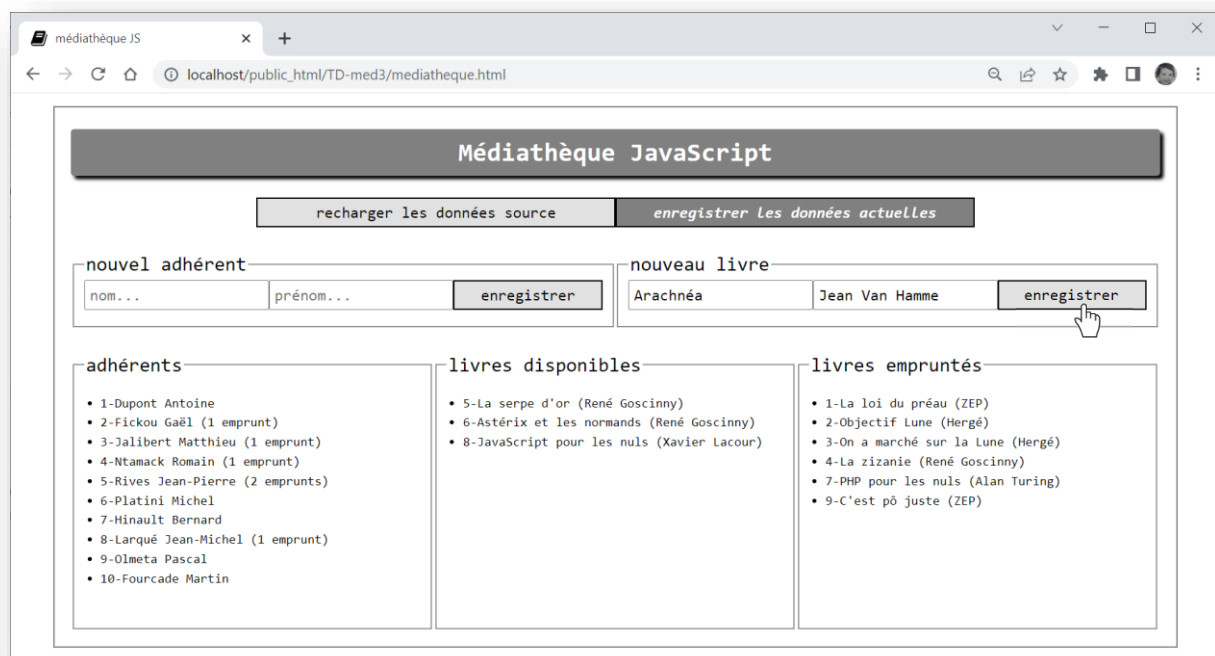
Capture 4 - clic sur un livre disponible à l'emprunt



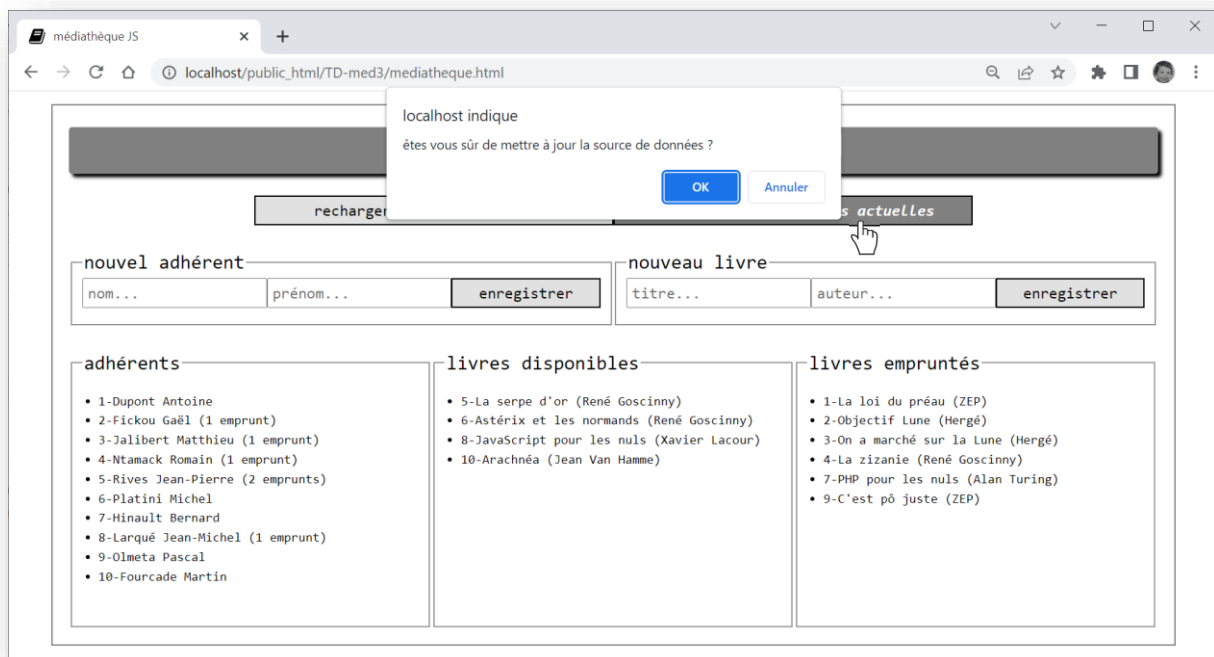
Capture 5 - clic sur un livre emprunté



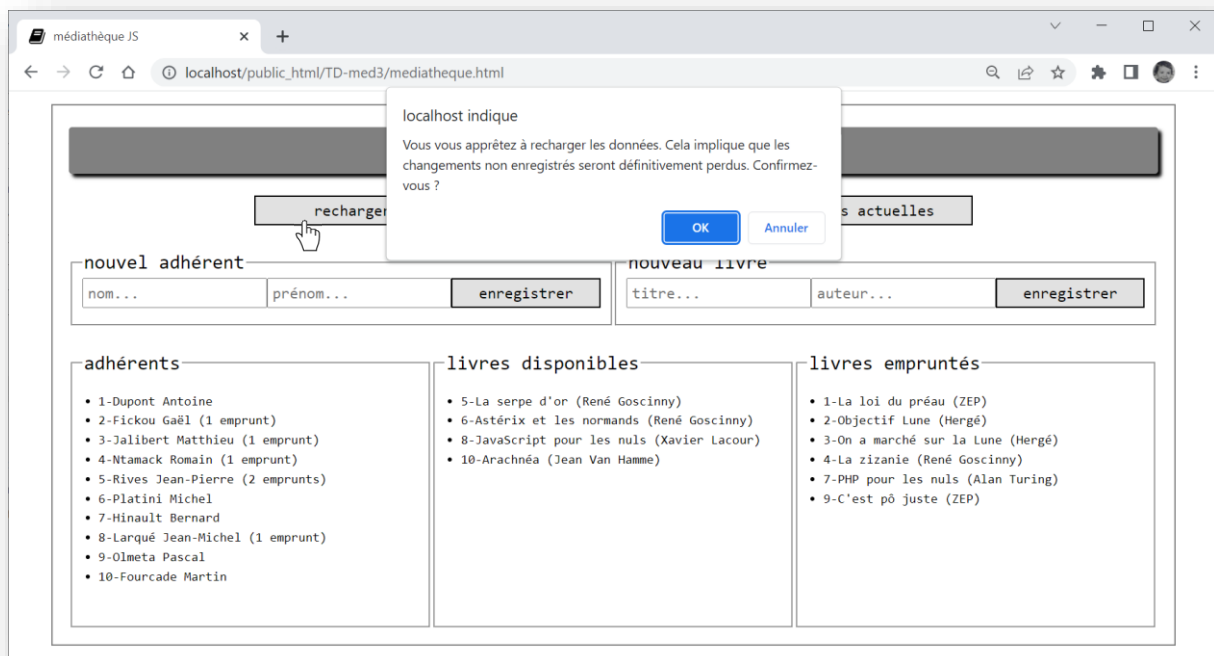
Capture 6 – enregistrement d'un nouvel adhérent



Capture 7 – enregistrement d'un nouveau livre

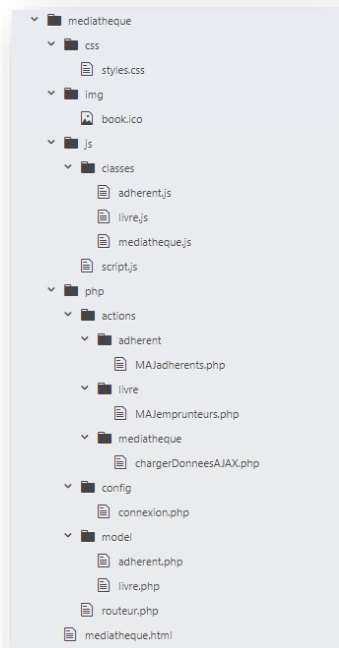


Capture 8 – clic sur le bouton d'enregistrement des données



Capture 9 – clic sur le bouton de rechargement des données

L'architecture de fichiers



PHP

Le modèle de conception MVC n'est pas utilisable dans ce cas, car c'est la vue qui déclenche les actions au travers d'événements. Il faut donc s'adapter.

L'organisation des fichiers PHP a été définie pour présenter de façon classique les fichiers modèle. Ils vous sont donnés, à compléter. Ce ne sont que des catalogues de méthodes statiques. Pas d'attribut, car tout va se passer au niveau client, par JavaScript.

Donc c'est au niveau du codage JavaScript qu'on va instancier des objets. Les classes PHP ne servent donc qu'à lister des méthodes de requête.

Le répertoire « actions » est une nouveauté. Ces fichiers sont en quelque sorte des contrôleurs, chacun étant dédié à une unique tâche. En exemple, vous avez le fichier chargerDonneesMySQL.php.

JavaScript

Le répertoire js regroupe les « classes » JavaScript, et un fichier de fonctions script.js, à compléter.

Les trois classes js servent à instancier des livres, des adhérents et une médiathèque.

Le fichier script.js regroupe des fonctions utilitaires qui servent essentiellement à gérer les événements qui surviennent au niveau de l'interface.

Exercice 1 – Livre

Un Livre a pour attributs :

- `numLivre`, qui correspond à l'identifiant de la table de données,
- `titre` et `auteur`, qui sont des chaînes de caractères,
- `numEmprunteur`, qui fait référence au `numAdherent` d'un adhérent,
- `estEmprunte`, qui correspond au booléen de la table de données, mais à qui on donnera les valeurs 0 ou 1, ce qui évite certaines erreurs pas très intuitives par la suite.

Complétez la classe `Livre` définie dans le fichier `js/classes/livre.js`, avec comme seules méthodes :

- un constructeur de signature
`constructor(numLivre, titre, auteur, numEmprunteur, estEmprunte)`
- une méthode `toString()` qui retourne une chaîne de la forme

```
Livre[numLivre = 1, titre = La serpe d'or, auteur = René Goscinny, numEmprunteur = 2, estEmprunte = 1]
```

Exercice 2 – Adherent

Un Adherent a pour attributs :

- `numAdherent`, qui correspond à l'identifiant de la table de données,
- `nom` et `prenom`, qui sont des chaînes de caractères,
- `tabEmprunts` qui est un tableau (qui stockera les livres empruntés).

Complétez la classe `Adherent` définie dans le fichier `js/classes/adherent.js`, avec comme méthodes :

- un constructeur de signature
`constructor(numAdherent, nom, prenom)`
- une méthode `toString()` qui retourne une chaîne de la forme
`Adhérent[numAdherent = 2, nom = Fickou, prénom = Gaël]`
- une méthode `emprunte(livre)` qui ajoute le livre à la fin du tableau `tabEmprunts`,
- une méthode `retourne(livre)` qui supprime le livre du tableau des emprunts. Si le livre est « au milieu » du tableau, le tableau sera « resserré » pour ne pas laisser de case vide,
- une méthode `listeEmprunts()` qui servira à afficher dans la popup la liste des emprunts de l'adhérent cliqué. Voir en exemples les captures 1 et 2.

Exercice 3 – Mediatheque

La classe Mediatheque dispose d'un constructeur sans argument qui initialise les deux attributs de la nouvelle médiathèque :

- un attribut `tabLivres` qui sera un tableau de livres ;
- un attribut `tabAdherents` qui sera un tableau d'adhérents.

Une médiathèque dispose de méthodes :

- `getLivreByNumLivre(numLivre)` qui retourne le livre du tableau `tabLivres` dont l'identifiant est égal au `numLivre` passé en paramètre (et retourne `null` s'il n'y en a pas),
- `getAdherentByNumAdherent(numAdherent)` qui retourne l'adhérent du tableau `tabAdherents` dont l'identifiant est égal au `numAdherent` passé en paramètre (et retourne `null` s'il n'y en a pas),
- `ajouteLivre(livre)` qui ajoute le livre à `tabLivres` (en dernière position du tableau),
- `ajouteAdherent(adherent)` qui ajoute l'adhérent à `tabAdherents` (en dernière position du tableau),
- `prete(livre,adherent)` qui passe l'attribut `estEmprunte` du livre à 1, l'attribut `numEmprunteur` à la valeur du `numAdherent` de l'adhérent, et enfin appelle la méthode `emprunte` de l'adhérent, avec en argument le livre,
- `recupere(livre)` qui récupère l'adhérent dont le `numAdherent` est égal à l'attribut `numEmprunteur` du livre passé en paramètre, puis appelle la méthode `retourne(livre)` pour cet adhérent, puis met à jour les attributs `numEmprunteur` et `estEmprunte` du livre (nouvelles valeurs : `null` et 0),
- `insererLivres(tabLivres)` qui appelle la méthode `ajouteLivre(livre)` pour chaque livre du tableau passé en paramètre,
- `insererAdherents(tabAdherents)` qui, pour chaque élément du tableau passé en paramètre, construit un nouvel adhérent, et appelle la méthode `ajouteAdherent(adherent)` pour cet adhérent,
- `insererEmprunts(tabLivres)` qui, pour chaque livre du tableau passé en paramètre et dont le `numEmprunteur` n'est pas `null`, récupère l'adhérent emprunteur et appelle la méthode `prete(livre,adherent)`.

Codez cette classe Mediatheque.

Exercice 4 – Le fichier de scripts

Le fichier `script.js` initialise des variables JavaScript qui représentent, par l'intermédiaire de l'interface DOM, les balises importantes du fichier `mediatheque.html`.

Il initialise également une variable `M` qui sera l'équivalent JavaScript de la médiathèque.

Ce fichier regroupe aussi des fonctions qui permettent de :

- gérer les divers événements (des clics)
- recharger ou de sauvegarder les données
- remplir et mettre à jour l'interface matérialisant la médiathèque

Les signatures des fonctions sont toutes données, et certaines fonctions sont écrites.

Vous devez coder toutes les fonctions. Il est conseillé de les tester au fur et à mesure dans la console.

Exercice 5 – les fichiers PHP

Le routeur

Il y a un fichier `routeur.php` qui gère les appels AJAX : il récupère l'objet, l'action, et appelle le bon fichier. Ce routeur est donné et à analyser.

Les fichiers Modèle

Ils sont donnés. Il y a `adherent.php` et `livre.php`. Chacun regroupe 3 méthodes statiques :

- une méthode de sélection de toutes les entrées de la table (en format objet),
- une méthode d'ajout d'une entrée dans la table,
- une méthode de suppression de toutes les entrées de la table.

Les fichiers « actions »

Un exemple de fichier PHP est fourni : `chargerDonneesMySQL.php`.

Il vous servira à imaginer comment doivent être complétés les deux autres fichiers : `MAJadherents.php` et `MAJemprunteurs.php`.

A vous de compléter ces fichiers pour qu'ils remplissent leur mission.

Exercice 6 – Améliorations diverses

Parmi les améliorations possibles :

1. Gestion des données par un fichier local json qu'on peut lire et réécrire (recharger et sauvegarder),
2. Ajout de fonctionnalités diverses : bouton d'impression de la page, changements de css, ...
3. Gestion plus fine des adhérents et des livres : possibilité de supprimer un livre ou un adhérent, éventuellement les gérer dans des onglets spécifiques, ...