

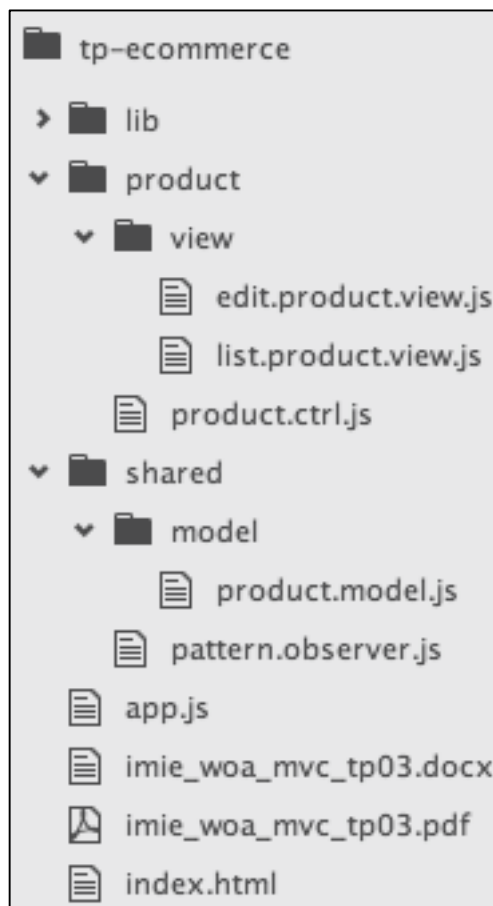
WOA & MVC – TP 03 – ECOMMERCE – PARTIE 1

OBJECTIF

- Appliquer le pattern MVC
- Utiliser les mécanismes de stockage du navigateur
- Communiquer avec un serveur

ETAPE 1 – STRUCTURE DE L'APPLICATION

Créer la structure suivante :






ETAPE 2 – LISTER LES PRODUITS

Un produit est composé des informations suivantes : id, label, price, imageUrl.

La vue liste produits est affichée par défaut.

Elle permet de :

- Créer un nouveau produit
- Modifier un produit
- Supprimer un produit

IMIECommerce				
<button>Nouveau</button>				
	Id	Libelle	Prix	Actions
	1	Sac à main	12.5	<button>Editer</button> <button>Supprimer</button>
	2	Sac à main 2	14.5	<button>Editer</button> <button>Supprimer</button>
	3	Sac à main 3	15.5	<button>Editer</button> <button>Supprimer</button>

A cette étape, les données sont stockées dans le modèle.

ETAPE 3 – CREER/EDITER/SUPPRIMER UN PRODUIT

- Implémenter la création d'un nouveau produit

IMIECommerce	
<button>Liste Produits</button>	
Id	<input type="text"/>
Libelle	<input type="text"/>
Prix	<input type="text"/>
Image URL	<input type="text"/>
<button>Valider</button>	

- Implémenter la modification d'un produit

IMIECommerce

Liste Produits

Id

Libelle

Prix

Image URL

Valider

- Implémenter la suppression d'un produit

ETAPE 4 – PERSISTANCE DANS LE NAVIGATEUR

Créer un fichier « shared/storage/product.storage.js » qui contient un composant « ProductStorage » qui gère la persistance des produits.

Utiliser le localStorage pour persister les modifications effectuées sur les produits.

ETAPE 5 – SERVICES SERVEUR

Créer une application Java EE qui expose les services :

- GET /api/products → retourne la liste des produits
- POST /api/products → crée un produit
- PUT /api/products → modifie un produit
- DELETE /api/products → supprime un produit

Côté client, implémenter la logique suivante :

- Lorsque l'utilisateur effectue une modification du modèle (création/modification/suppression)
 - La modification est persistée dans le localStorage
 - Les informations nécessaires à la génération de la requête serveur sont stockées également dans le localStorage
- Toutes les 10 secondes, l'application cliente se synchronise avec l'application serveur.

- Elle exécute chaque des requêtes correspondantes aux différentes modifications effectuées en local.
 - Si une requête est en erreur
 - Si code 4XX (Bad Request), l'application affiche les informations des requêtes en erreur et propose à l'utilisateur de refaire la saisie
 - Si code 5XX (Erreur côté serveur), l'application notifie qu'elle ne peut pas se synchroniser. Elle propose à l'utilisateur :
 - Soit de recommencer la saisie liée à l'erreur
 - Soit de conserver sa modification pour réessayer une prochaine synchronisation
- Elle récupère la liste de produits à jour depuis le serveur puis met à jour le localStorage et l'IHM.

ETAPE 6 – LOCALFORAGE

Remplacer l'utilisation du localStorage par la librairie localforage

(<https://github.com/mozilla/localForage>)