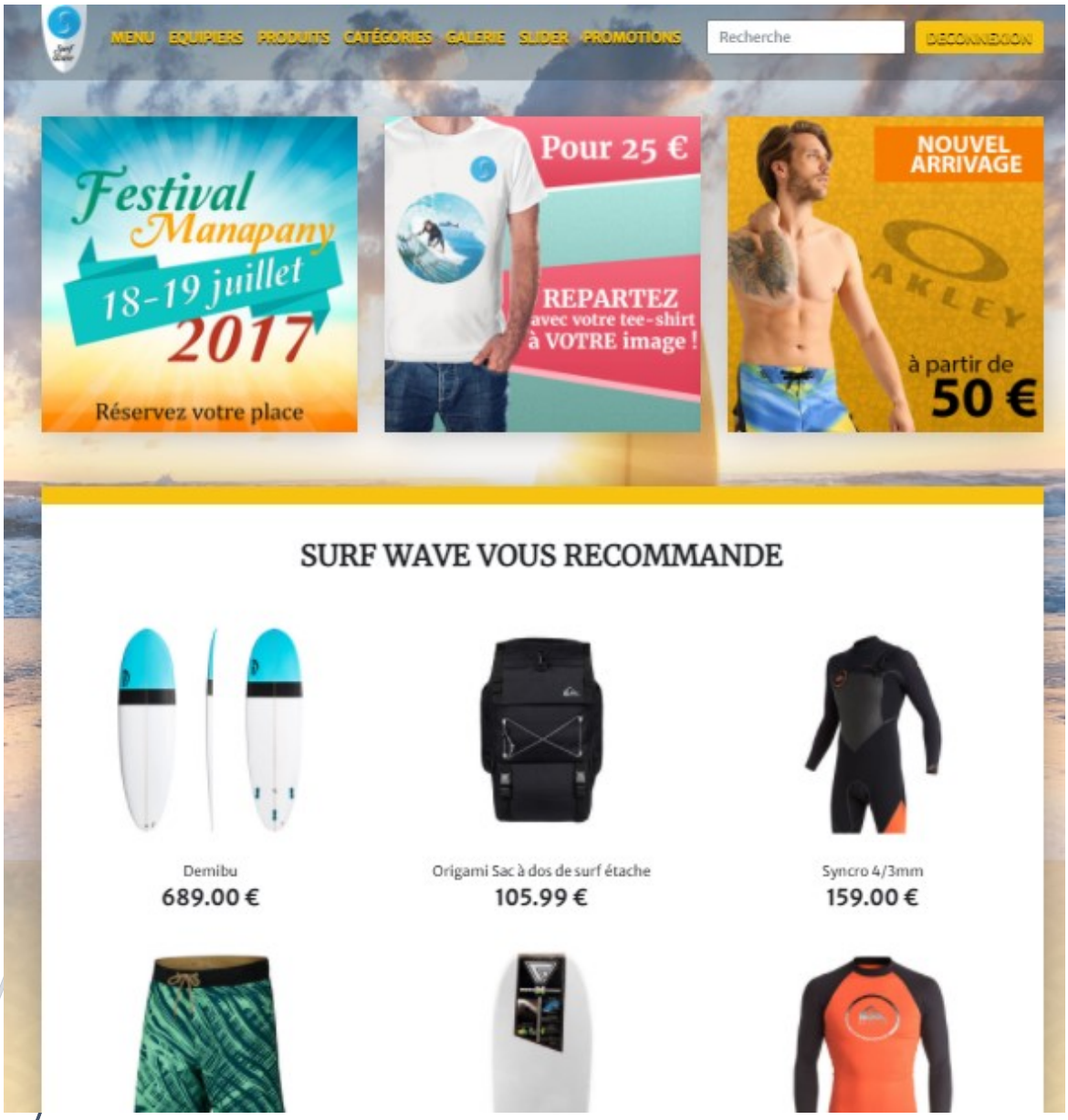
03/02/2022

Surfwave

SITE WEB MARCHAND version 2



CHRISTIAN MARAIS.

DWWM- AFPAR

TABLE DES MATIERES

SURFWAVE

Contexte :

On a reçu en demande la réalisation du backend et la transformation d'une page d'accueil statique du site marchand SURFWAVE en page d'accueil dynamique.

La partie graphique et le design ont déjà été réalisée et validé par des designer. Le backend reprendra ce design. Il a été imposer de réaliser le site suivant la méthode MVC. L'accès aux données seront faits par des fichiers nommés model qui transmettront les données aux contrôleurs chargés de les afficher quand ces derniers appelleront ces modèles. Il a été choisi d'utiliser des classes : abstraites groupant les méthodes de bases utilisées par les autres classes, des classes filles qui seront les contrôleurs appelés qui s'occuperont chacun de gérer leur propre activité. Pour les cheminements de fichier on a combiné l'utilisation d'un fichier HT Access et d'un routeur.

Il a été choisi de réaliser le site entièrement en PHP dans sa partie backend, aucun appel sera fait au javascript sur la partie dynamique du back. On a décidé de réutiliser certains composants graphiques du front pour le backend.

Le site comprend un CRUD produits, équipiers et catégorie, un système de gestion de photo et un système de gestion de slider.

Liste des outils utilisées :

Les technologies : Sql, Php8,SAAS

Les techniques : MVC en POO avec encapsulation et héritage de classe uniquement. Utilisation du rewrite engine de Htaccess, le moteur de réécriture d'url de Apache.

Les plateformes : LAMP(Mysql)

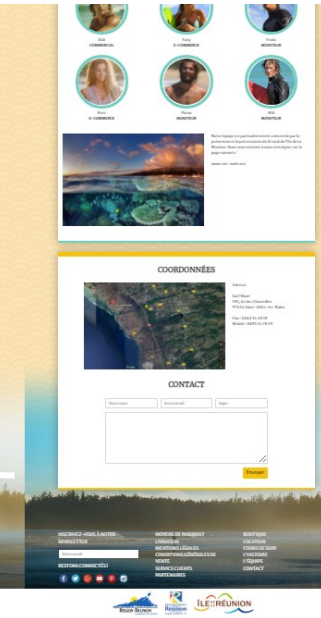
Légende :

Partie dynamique avec appel de BDD

Partie dynamique sans appel de BDD. Utilisation de méthode Php comme scanDir() pour obtenir tous les url des images d'un dossier qui seront ensuite inséré dans la page comme variables

View index

Composant FOOTER



CAHIER DES CHARGES :

Fonctionnalités :

Il a été réalisé les fonctionnalités *A,B,C,D2.*

1. A. Il a été codé la partie dynamique du slider. Il a été choisie d'utiliser la technique suggérée D2 ; Le slider sera alimenté par un listing des fichiers contenu dans un dossier slider avec des contraintes sur les images. Les images seront sélectionnées parmi les photos de la galerie présentes dans le dossier banque en suivant des critères de détails de fichiers(jpg, taille,dimensions).

2. B. Les données de la zone surfwave vous recommande sont désormais alimentées par la base de données sql.
3. Nous avons réalisé un crud produit avec des INSERT, Update, Delete.

Il a été choisie de réaliser le MVC du site en ayant recours uniquement à la méthode POST dans les process. Aucune méthode get n'a été utilisée. La variable de session

Correspondance arborescence des blocs html et des fichiers du site:

Ci-dessous le tableau des view correspondant à une url particulière :

HTML	→	*VIEW*
Accueil	→	*index.php*
Accueil >>login	→	*index.php+LOGIN.php*
Accueil>>login>>admin	→	*admin.php*
Accueil>>LOGIN>>Gestion administration>>produits	→	*produits.php (affichage+méthode delete)*
Accueil>>login>>gestion administration>>produits	→	*createArticle.php*
Accueil>>login>>gestion administration>>produits	→	*editArticle.php (delete et update)*
Accueil>>LOGIN>>Gestion administration>> equipiers	→	*equipiers.php (affichage+méthode delete)*
Accueil>>LOGIN>>Gestion administration>> equipiers	→	*createEquipier.php*
Accueil>>LOGIN>>Gestion administration>> equipiers	→	*editEquipier.php*

Accueil>>LOGIN>>Gestion administration>> categories → *catégories.php (affichage+méthode delete)*

Accueil>>LOGIN>>Gestion administration>> categories → *createCatégorie*

Accueil>>LOGIN>>Gestion administration>> categories → *editCategorie*

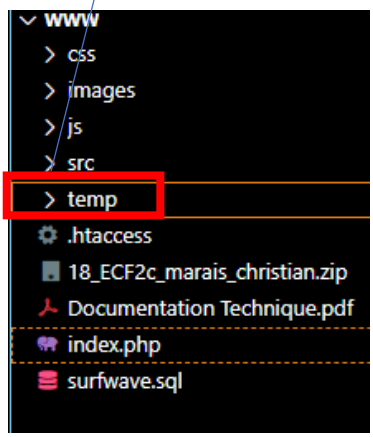
Accueil>>LOGIN>>Gestion administration>> galerie → *galerie.php (méthode upload)*

Accueil>>LOGIN>>Gestion administration>> slider → *slider.php>>(méthode upload, choix de fichiers et delete).*

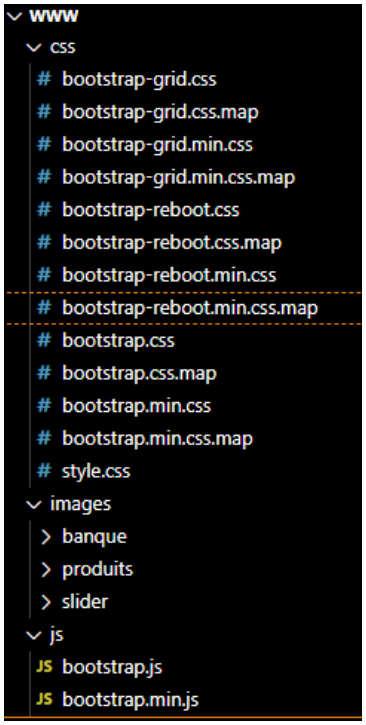
Arborescence des fichiers :

Arborescence pliée

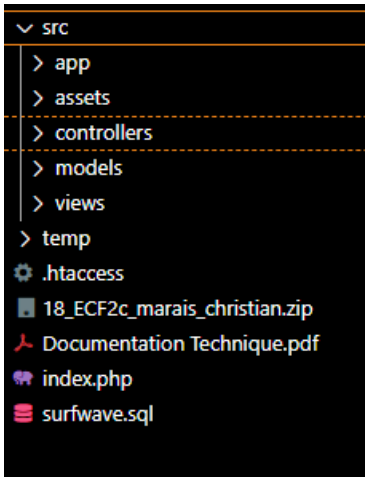
Les fichiers déplacés ou importés sont d'abord stocké dans le dossier temp en tant que Temp.ext(nom del'extension du fichier) puis déplacé vers son dossier de destination par la méthode gestion de fichier de controller.php



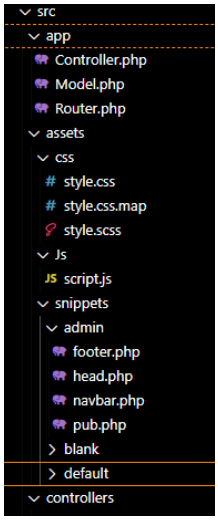
Arborescence des sources orginaux



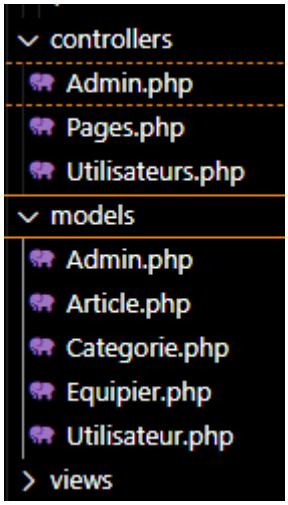
arborescence des sources php et assets personnalisées



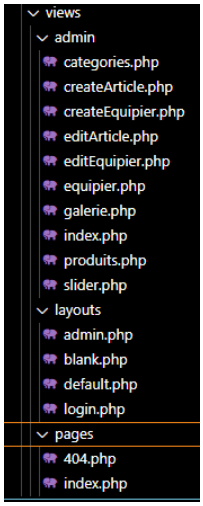
Arborescence des assets



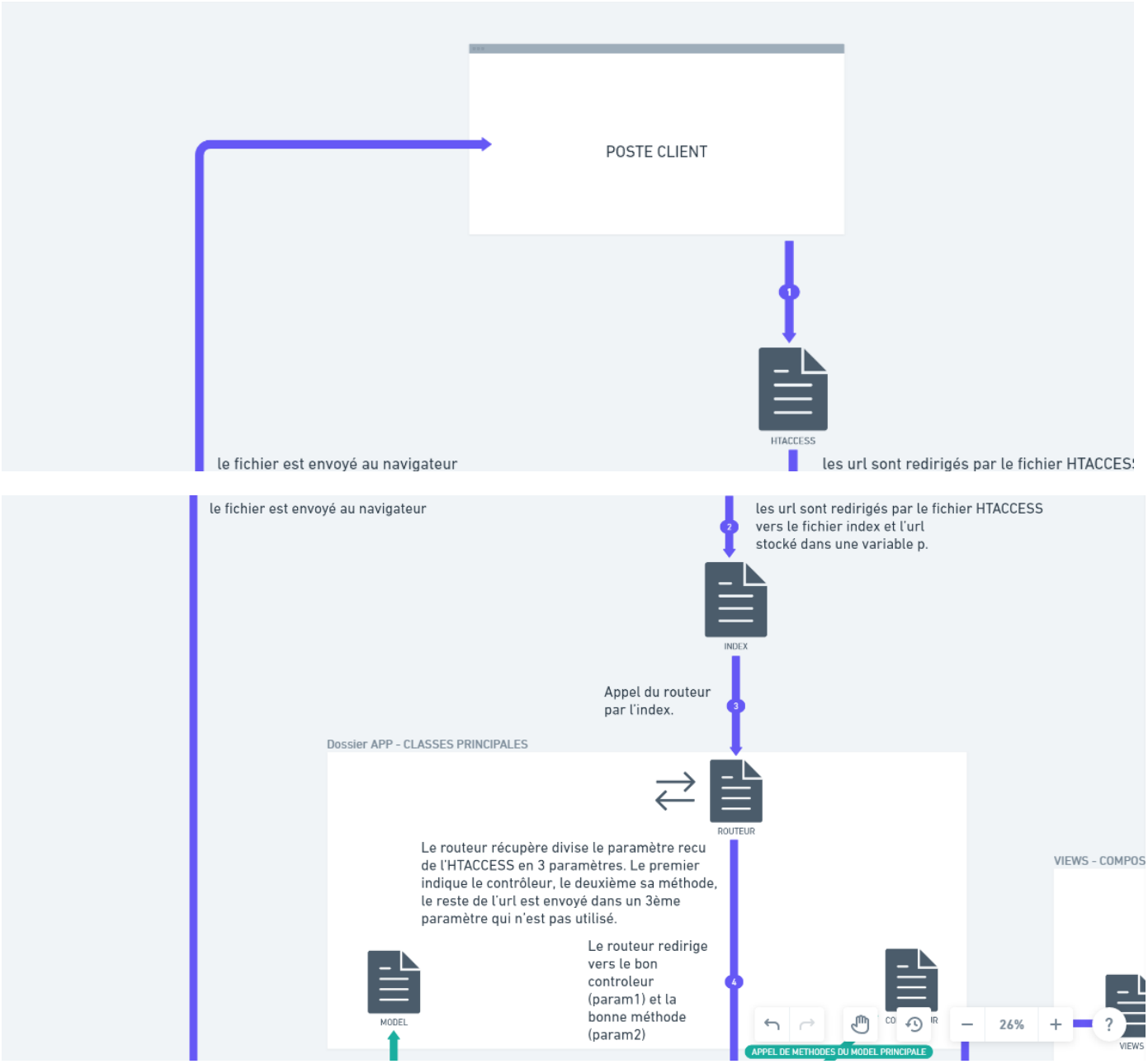
Arborescence des contrôleurs

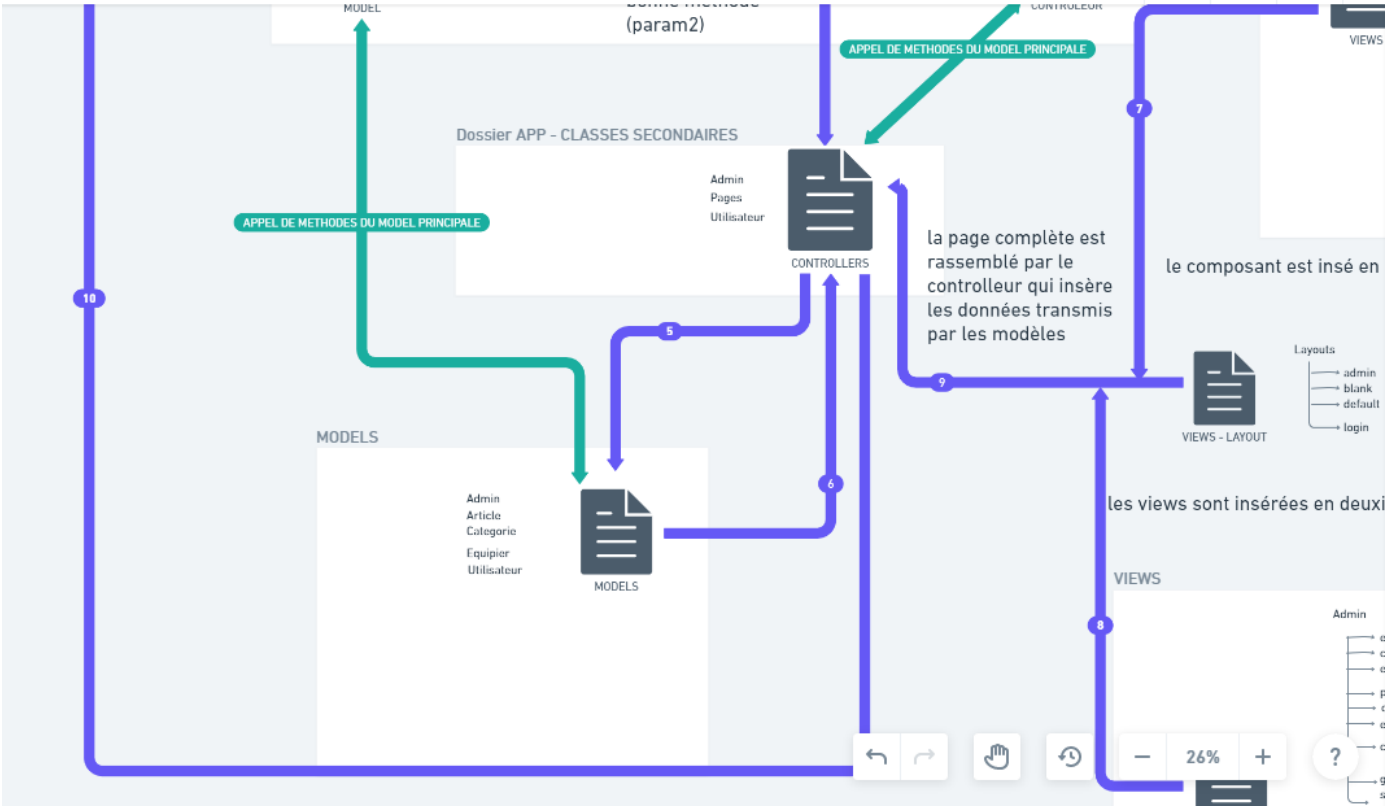


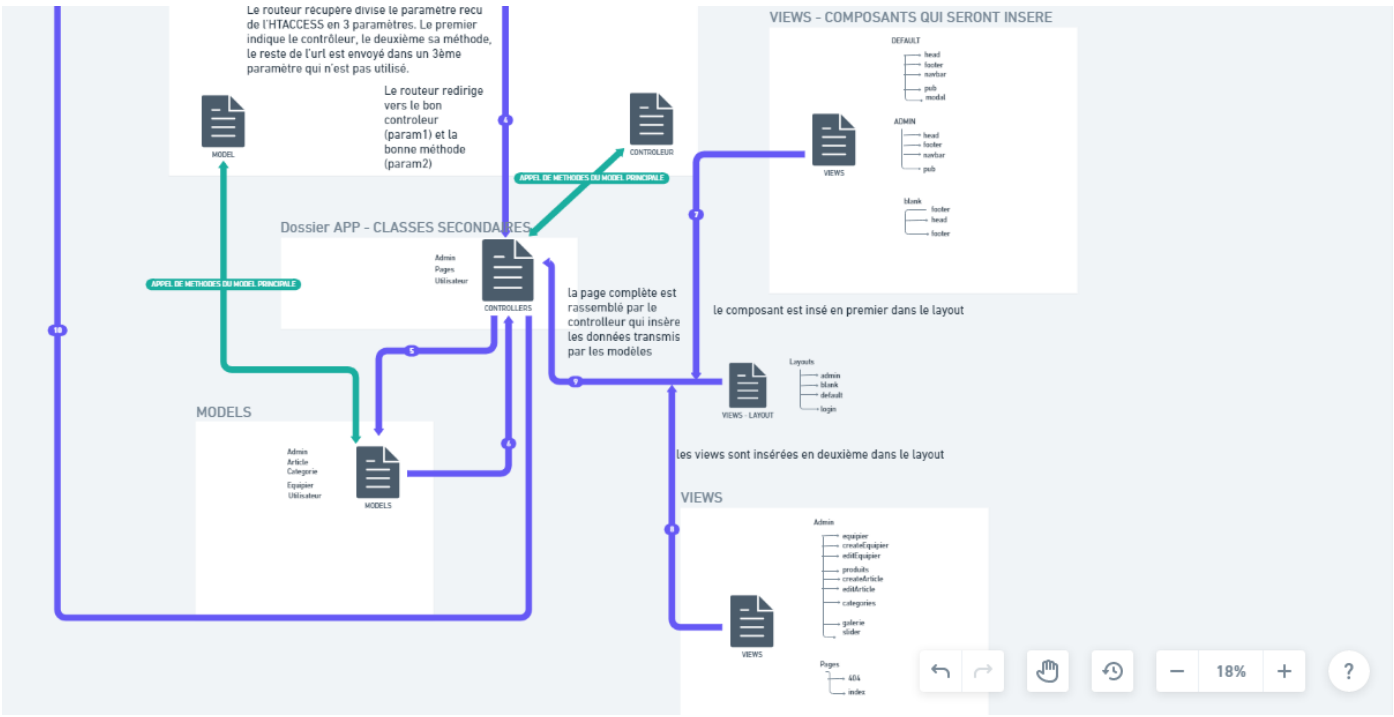
arborescence des views



LE MVC :


POSTE CLIENT
1
HTACCESS
le fichier est envoyé au navigateur
les url sont redirigés par le fichier HTACCES
le fichier est envoyé au navigateur
2
les url sont redirigés par le fichier HTACCESS
vers le fichier index et l'url
stocké dans une variable p.
INDEX
Appel du routeur
par l'index.
3
Dossier APP - CLASSES PRINCIPALES
ROUTEUR
Le routeur récupère divise le paramètre recu
de l'HTACCESS en 3 paramètres. Le premier
indique le contrôleur, le deuxième sa méthode,
le reste de l'url est envoyé dans un 3ème
paramètre qui n'est pas utilisé.
Le routeur redirige
vers le bon
controleur
(param1) et la
bonne méthode
(param2)
4
MODEL
VIEWS - COMPOS
26%
APPEL DE METHODES DU MODEL PRINCIPALE
VIEWS





Principes d'encapsulation utilisées pour les méthodes et variables :

Classe de App : protected et private. Cœur du système contient les méthodes de fonctionnement de l'affichage et plus sensibles les méthodes d'accès à la BDD. Le routeur a une méthode publique init qui sert de point d'entrée sinon la méthode redirection est private.

Classe de Controller : public car les méthodes seront appelés par le routeur qui est externe à la classe. Elles appelleront les méthodes et variables protected et private des classes du dossier app.

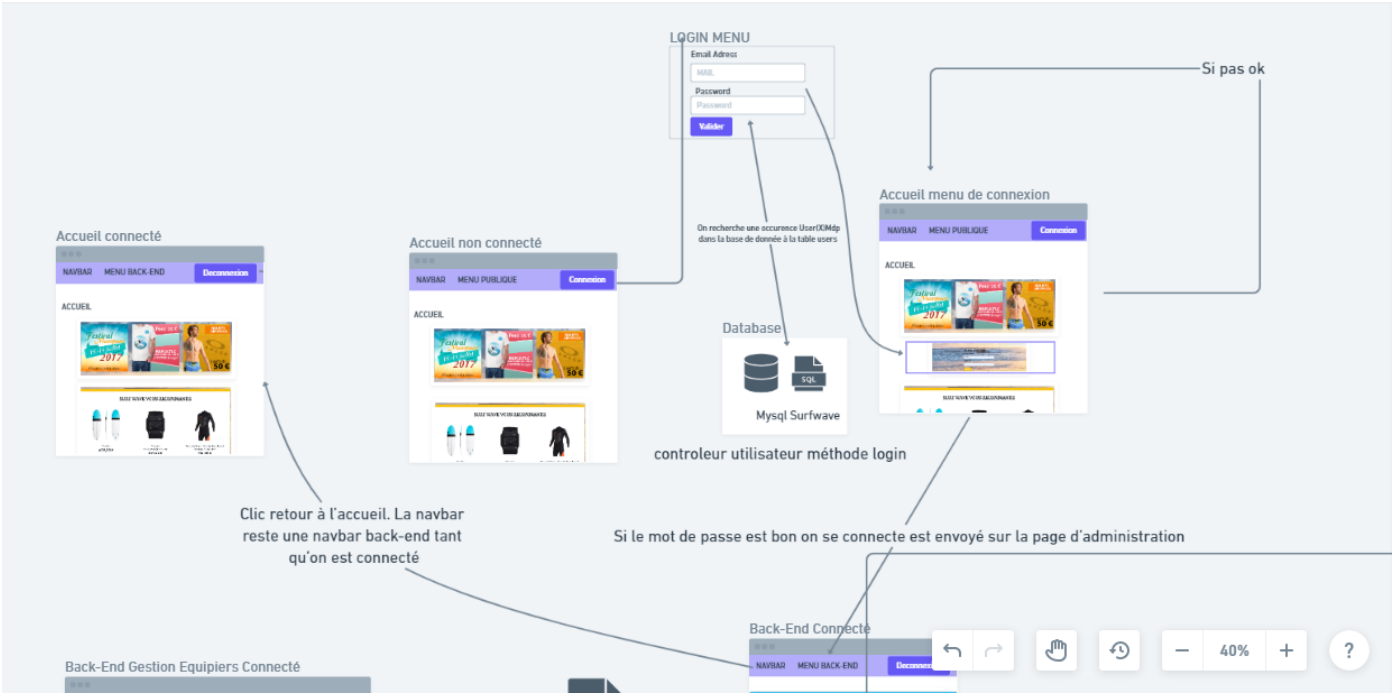
Classe de model : public car les méthodes seront appelés par les classes externes controlleurs. Les objets sont cependant séparés en plusieurs classes ou classes métiers pour avoir accès qu'aux données pertinentes au contexte sans exposer l'ensemble de la bdd.

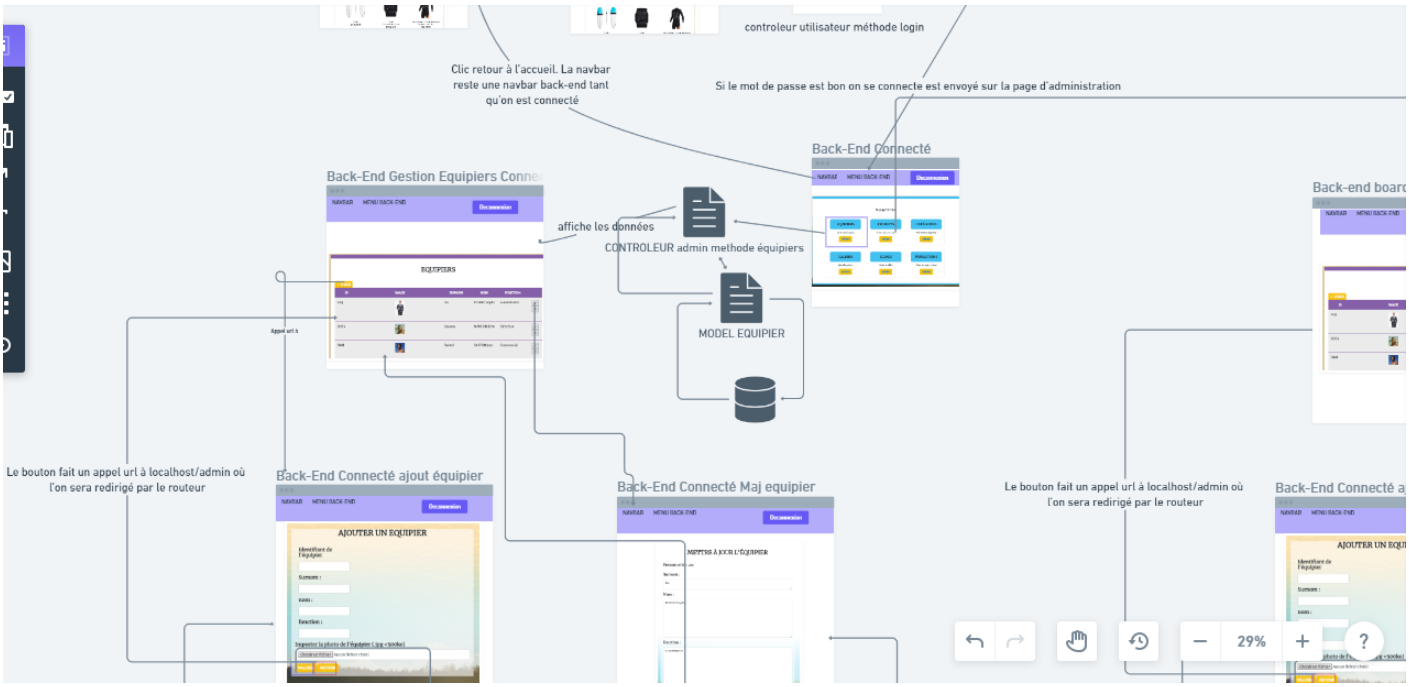
Fonctionnement de la méthode MVC utilisée :

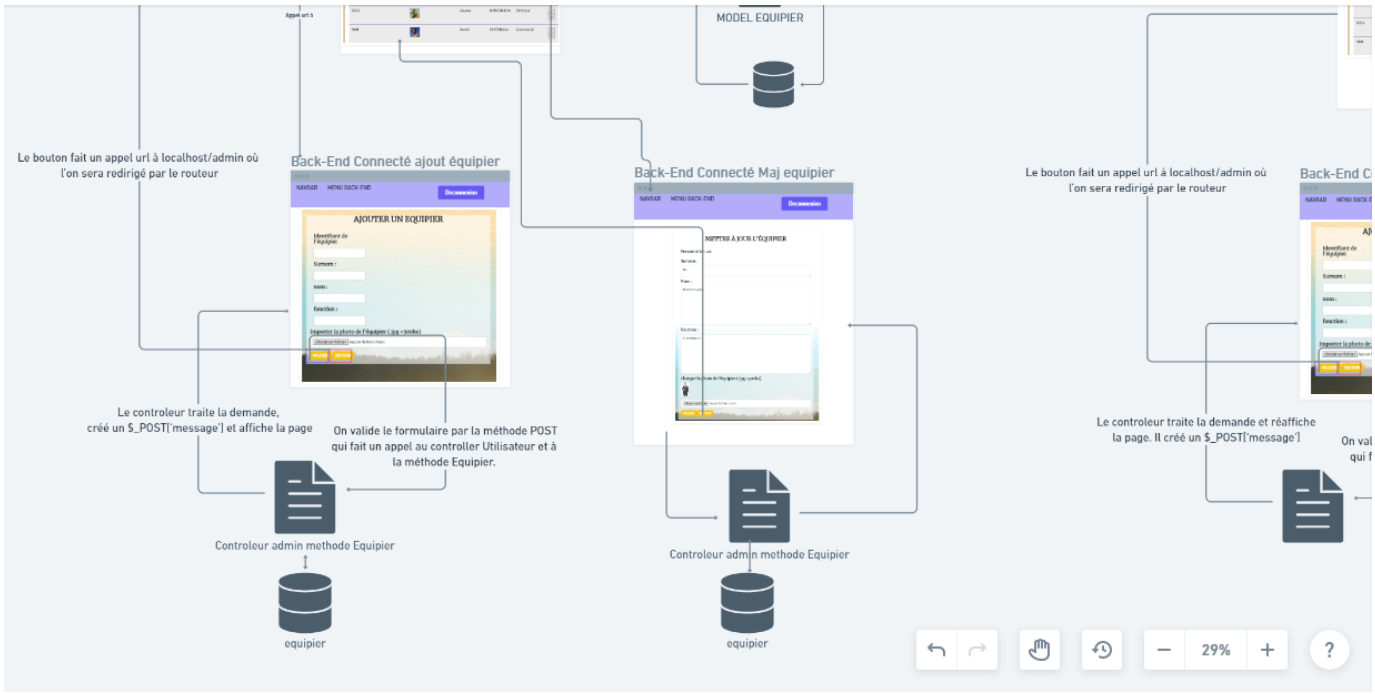
Le système de routage utilise deux moteurs : le fichier HTACCESS et le fichier ROUTEUR.PHP.

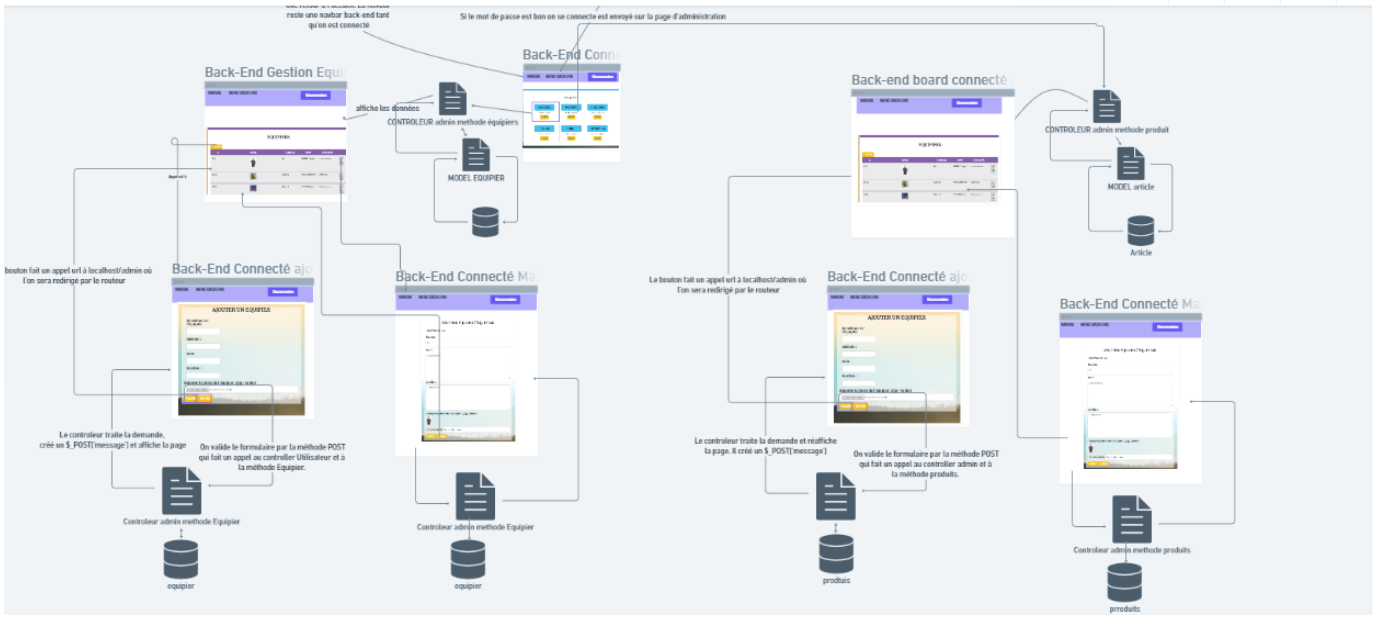
1. HtAccess :
    a. redirige tous les url vers le fichier racine index.php.
    b. url est envoyé en méthode get.
2. L'index.php inclut la classe routeur et l'instancie. C'est son seul contenu.
3. Le routeur.php inclut les cores ou un autoloader des classes.
    a. Il redirige vers le bon sous contrôleur qui ont chacun leur activité métiers. On a 3 : utilisateurs pour la gestion des login, Admin pour le backend, Pages pour l'affichage des page statiques comme la page d'erreur ou l'index.
    b. Stocke l'url dans un tableau p ou chaque dossier de l'url est séparé. Le tableau contiendra 3 éléments. L'élément 0 correspondant au contrôleur, l'élément 1 correspondant à la méthode et l'élement 3 correspondant aux paramètres supplémentaires pour un possible usage de la méthode get.
4. Les contrôleurs
    a. Ils font appels aux models avec la méthode loadModel
    b. gèrent l'appel des views utilisées pour l'affichage avec la méthode render
    c. Incluent les données dans les views en paramètre de render

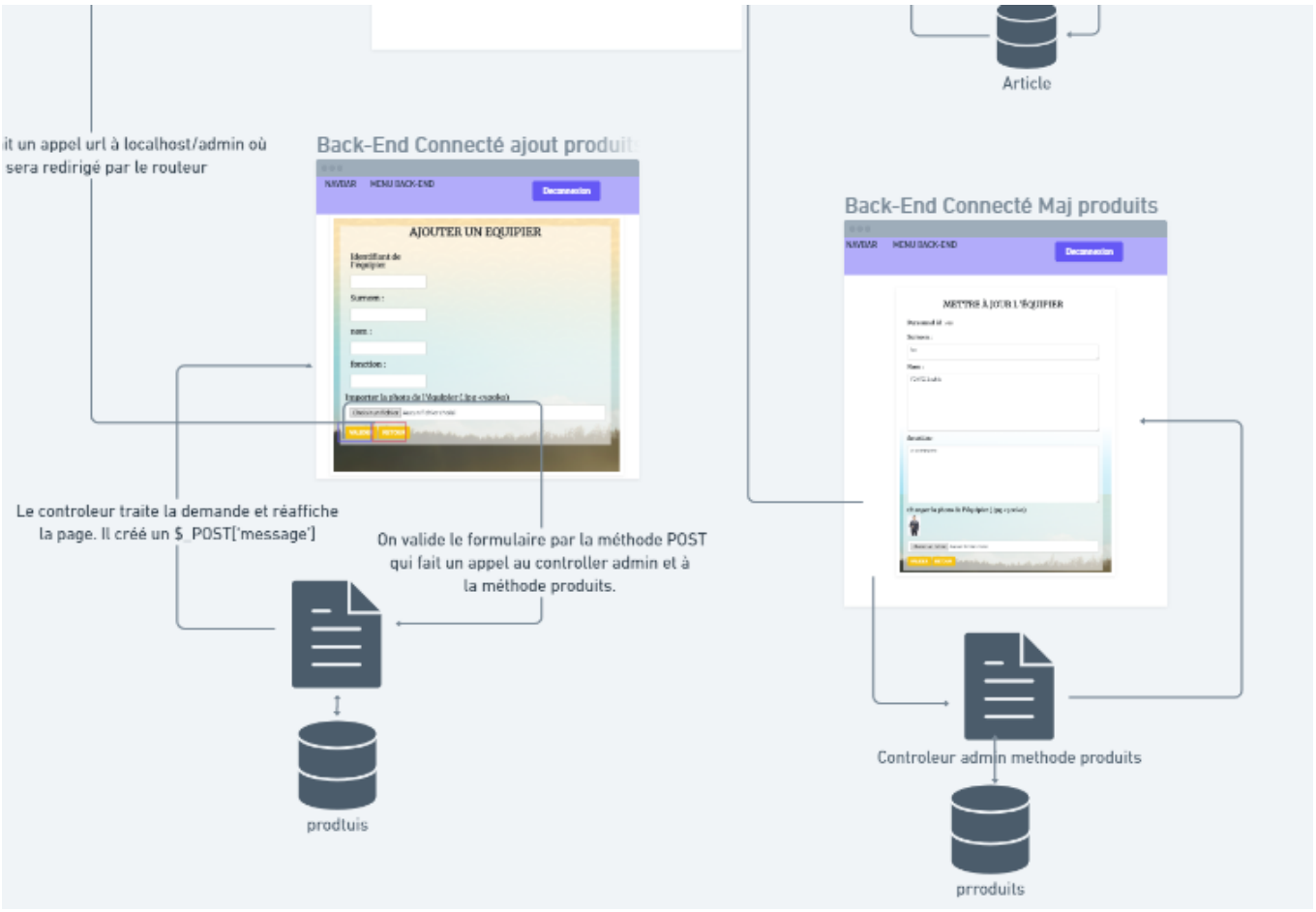
LE GRAPHE DE DIALOGUE:





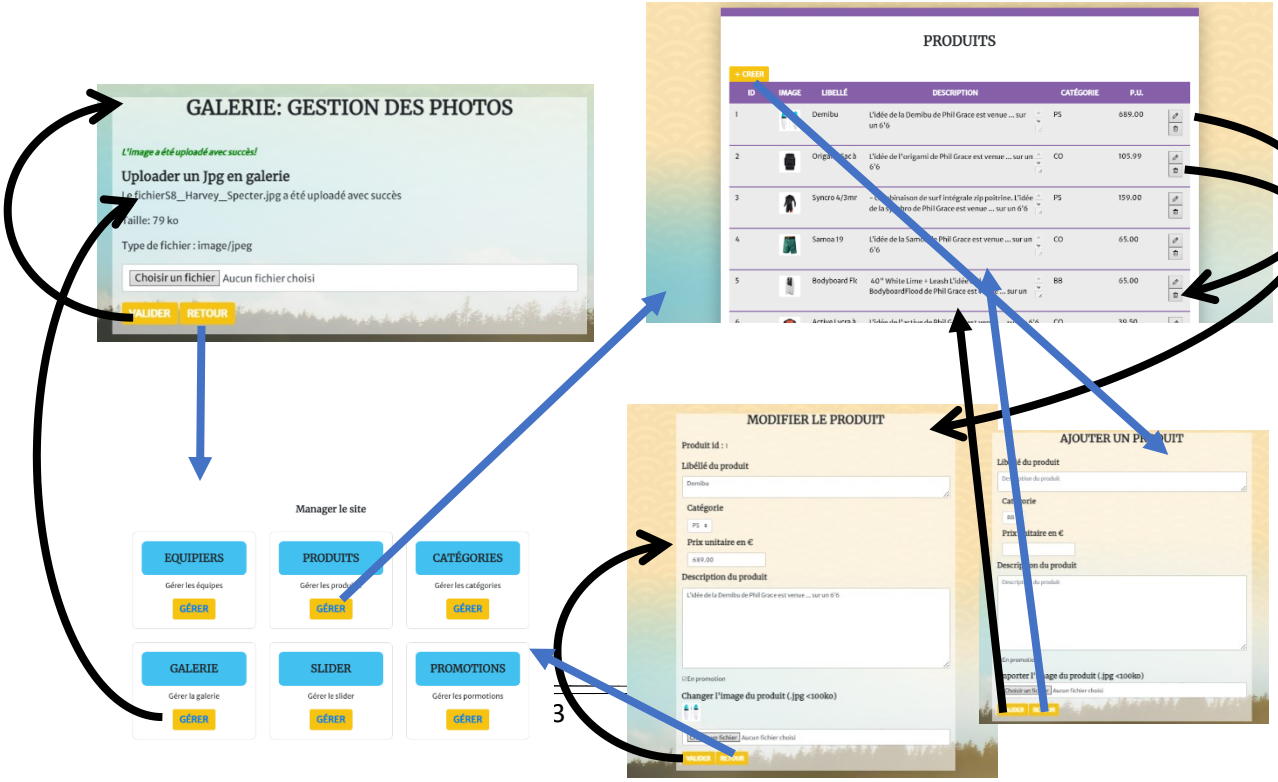

MODEL EQUIPIER
Le bouton fait un appel url à localhost/admin où l'on sera redirigé par le routeur
Back-End Connecté ajout équipier
AJOUTER UN EQUIPIER
Back-End Connecté Maj equipier
Le bouton fait un appel url à localhost/admin où l'on sera redirigé par le routeur
Le controleur traite la demande, créé un $_POST['message'] et affiche la page
On valide le formulaire par la méthode POST qui fait un appel au controller Utilisateur et à la méthode Equipier.
Controleur admin methode Equipier
equipier
Controleur admin methode Equipier
equipier
Le controleur traite la demande et réaffiche la page. Il créé un $_POST['message']
29%


Si le mot de passe est bon on se connecte est envoyé sur la page d'administration
Back-End Gestion Equi
Back-End Conn
affiche les données
CONTROLEUR admin methode équipiers
MODEL EQUIPIER
Back-end board connecté
CONTROLEUR admin methode produit
MODEL article
Article
Back-End Connecté ajo
Back-End Connecté Ma
Le bouton fait un appel url à localhost/admin où l'on sera redirigé par le routeur
Le controleur traite la demande, créé un $_POST['message'] et affiche la page
On valide le formulaire par la méthode POST qui fait un appel au controller Utilisateur et à la méthode Equipier.
Controleur admin methode Equipier
equipier
Controleur admin methode Equipier
equipier
Le controleur traite la demande et réaffiche la page. Il créé un $_POST['message']
On valide le formulaire par la méthode POST qui fait un appel au controller admin et à la méthode produits.
produits
Controleur admin methode produits
produits



NB :sur le dessin le model est d'abord représenté avec un intermédiaire le model pour le signaler mais par la suite il n'est plus représenté mais bien toutes accès à la bdd passe obligatoirement par un model dédié.

Légende schéma ci-dessous :
Bleu : sans envoie de données
Noir : avec envoie de données en $_POST



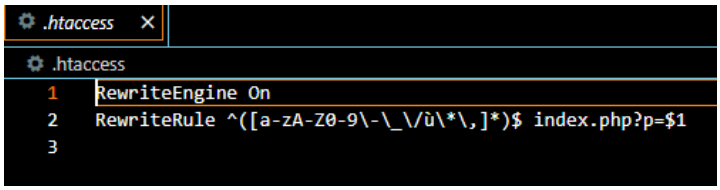
TODO LIST:

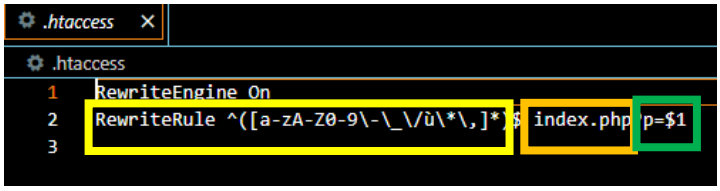
Ameliorer la méthode gestionfichier. Autoriser un mode de sélection de fichier par défaut et personnalisé. Les fichiers sont selectionnés avec des contraintes de width égale comme demandé, prévoir qu'on puisse respecter cette contrainte tout en autorisant sur d'autres pages l'upload de fichier sans taille défini en recourant à la même méthode.

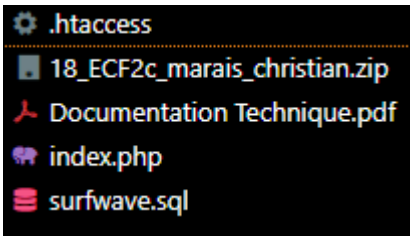
Un déboggage d'un bug qui s'est glissée dans la gestion de fichier suite une manipulation accidentelle.
Prévoir une méthode dans le routeur pour modifier personnaliser l'url sans modifier le nom de classes.
Commentaire à rajouter et indentation. Réduire le volume de code.
Durée : une demi journée.
Débogage réalisée.

HOW TO LIST:

Pour fonctionner le MVC nécéssite le fichier htaccess à la racine:







Réécrit tout caractère de l'url

En index.php

Transmet l'url par le get[p]