CFPT-I

Catal'info

TPI - 2015

SEEMULLER Julien

15/05/2015

Table des matières

1. Introduction

Dans le cadre de mon travail pratique individuel, le but est de réaliser un site web pour qu'un revendeur puisse fournir un catalogue en ligne des produits informatiques disponibles dans son établissement.

Les utilisateurs peuvent rechercher et visualiser les informations relatives aux composants ou produits informatique. Le catalogue informatique est géré par les administrateurs du site web et permet de fournir un maximum de documentation à l'utilisateur tel que des images, des descriptions du produits, le prix des produits ou encore des documentations sur les produits présentés.

Le site possède aussi différents moyens de classer les produits :

- Trier par popularité
- Trier par le choix des administrateurs
- Trier par date d'ajout

En plus d'une fonction de classement, il est aussi possible de rechercher des produits et de les afficher par catégories.

2. Etude d'opportunité

Il existe de nombreux sites de revendeurs de produits informatique, en voici quelques exemples :

2.1. ① PRODIMEX

http://www.prodimex.ch/

Points positifs :

- Images représentatives du produit
- Recherche précise du produit
- Médias dans descriptif du produit
- Titre du produit possède une description
- Changement de devises CHF vers EUR et inversement
- Affichage disponibilité des produits à l'aide d'un icone

Points négatifs :

- Menu supérieur confus (trop de termes & catégories vagues)
- Redondance dans les menus
- Interface peu intuitive par moments
- Ne possède pas de site mobile

2.2. ② 1000 ORDI

http://www.1000ordi.ch/

Points positifs :

- Interface intuitive
- Site disponible en deux langues différentes
- Menu supérieur propre
- Images représentatives du produit
- Recherche précise du produit
- Médias dans descriptif du produit
- Titre du produit possède une description
- Affichage disponibilité des produits à l'aide d'un icone
- Liste de comptabilité avec un produit en particulier

Points négatifs :

- Ne possède pas de site mobile

2.3. ③ STEG

http://www.steg-electronics.ch/fr/Default.aspx

Points positifs :

- Site disponible en trois langues différentes
- Catégories claires
- Affichage disponibilité des produits à l'aide d'un icone
- Images représentatives du produit
- Recherche précise du produit
- Médias dans descriptif du produit

Points négatifs :

- Ne possède pas de site mobile
- Interface peu intuitive par moments
- Site « trop long » (information pas instantanément accessible)

Il existe bien sûr beaucoup d'autres sites de revendeurs de produits informatique, j'ai décidé de limiter mon étude d'opportunité sur des sites populaires suisses.

Le site web partage donc des similitudes avec ces différents sites web. Par exemple, le site web possède une interface intuitive à l'utilisateur, un système de recherche par mots-clés, un affichage graphique des produits ou encore un affichage de la disponibilité des produits à l'aide d'icônes.

Une des problématiques présente sur la totalité des sites de mon échantillon est le manque d'une interface disponible pour téléphone mobile. De nos jours, les utilisateurs utilisent de plus en plus leur smartphone pour accéder à des sites web. Je pense qu'il est donc important d'implémenter une interface adaptable au format mobile.

3. Analyse fonctionnelle

3.1. Généralités

Voici le fonctionnement général de Catal'info :

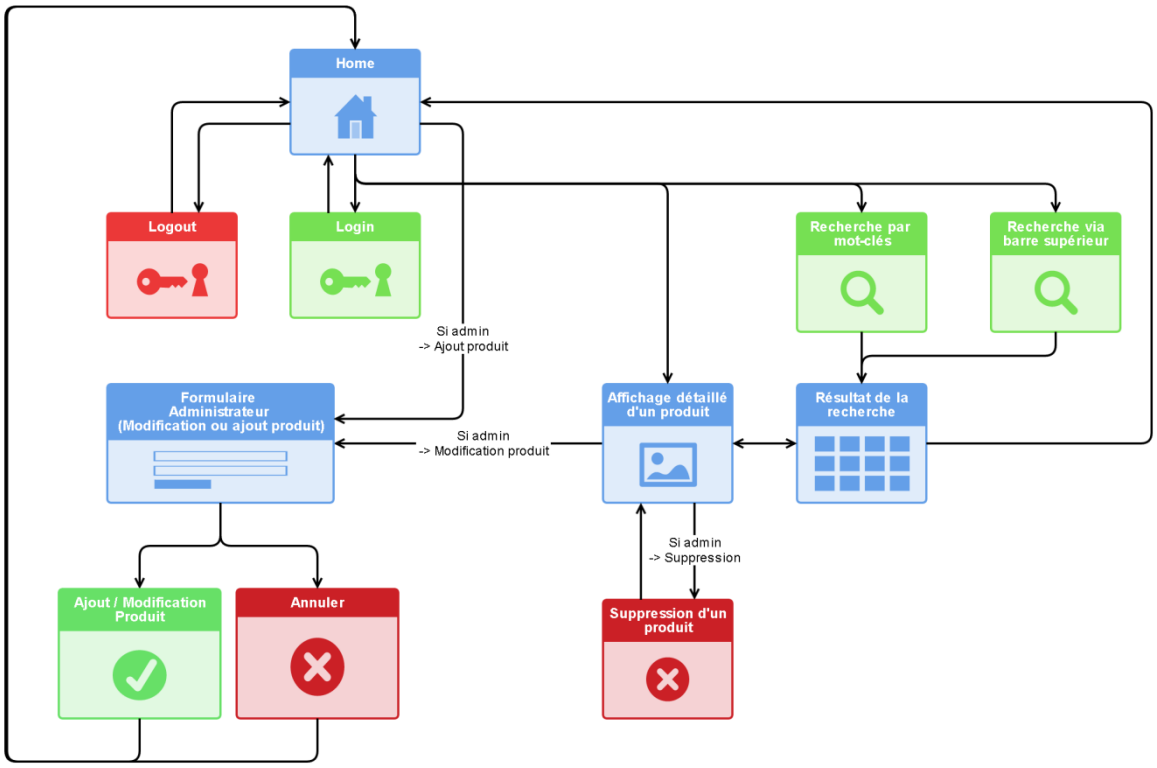
L'utilisateur est tout d'abord accueilli sur une page contenant les produits suggérés par les administrateurs du site, les produits les plus populaires auprès des utilisateurs et les produits ajoutés récemment.

Cette page possède aussi une barre de recherche permettant à l'utilisateur d'entrer des critères tels que le titre du produit, la marque ou encore des éléments de la description, puis on mène l'utilisateur sur une page contenant le résultat de sa recherche. Sur le côté gauche de la page, une liste des mots clés est affichée qui permet à l'utilisateur d'accéder rapidement à une catégorie (ex : RAM, Affiche une page avec la totalité des produits contenant le mot clé « RAM »).

Si un visiteur ou un administrateur clique sur un produit, on affiche les informations détaillées du produit selectionné.

Il existe deux types d'utilisateurs consultant le site. Le premier est le visiteur, il peut consulter les différentes informations des produits du site. Le deuxième est l'administrateur, il possède les mêmes privilèges que le visiteur mais peut aussi ajouter, modifier ou encore supprimer des produits. Quand un administrateur est connecté, il peut accéder à la modification ou suppression d'un produit en cliquant sur un icone d'engrenage se trouvant dans le coin de l'objet à modifier. Pour ajouter un produit, l'administrateur passe par une interface accessible via un lien dans la barre de navigation du site web.

Voici un schéma représentant les différentes pages du site web et les interractions entre ces dernières :

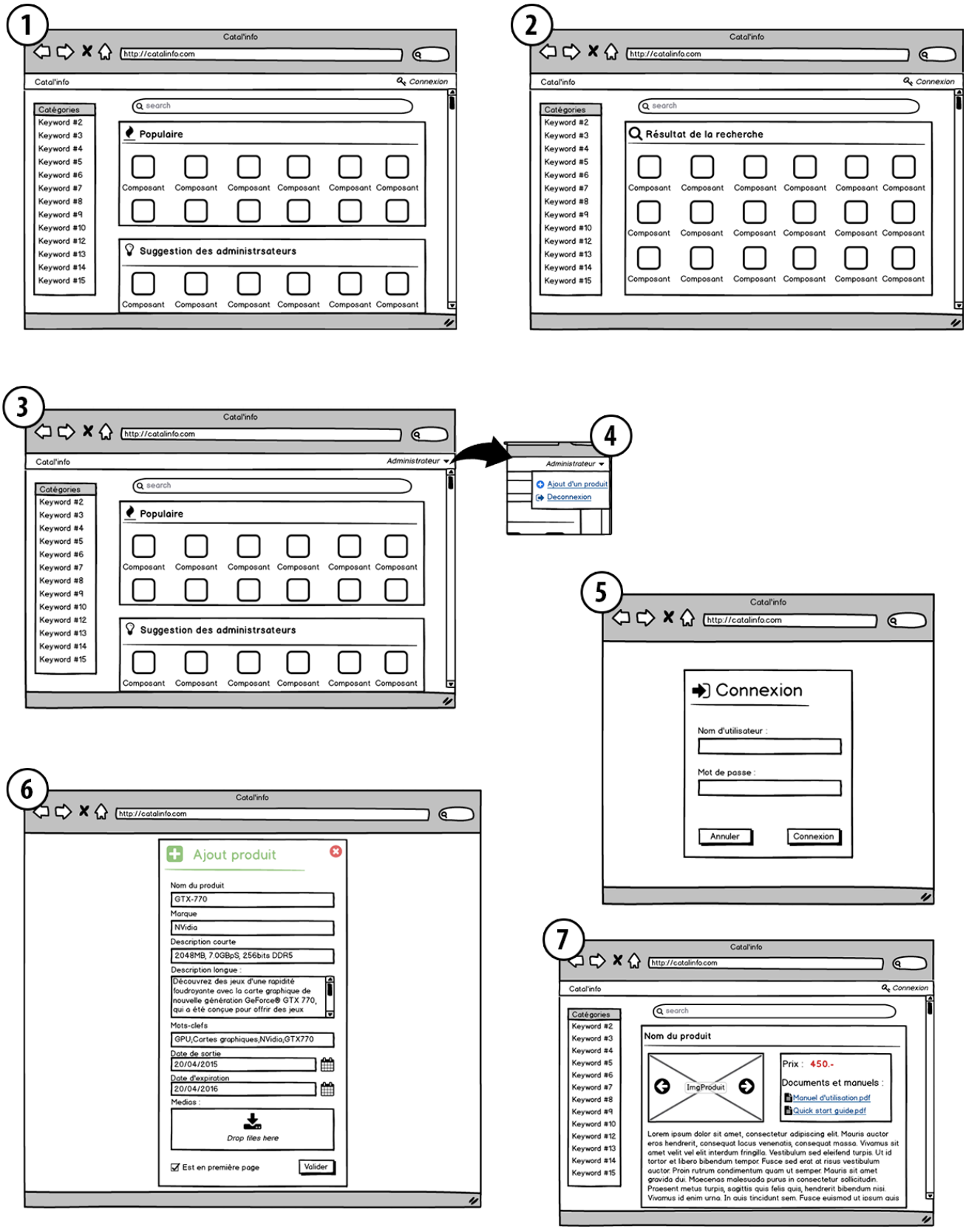


3.2. Description des fonctionnalités

Voici les fonctionnalités globales que possède Catal'info :

- Afficher les produits populaires et les sélections des administrateurs sur la page d'accueil
- Afficher la liste des produits selon une recherche de l'utilisateur
- Afficher la liste des produits selon un mot clé
- Les administrateurs peuvent :
    - Ajouter un produit et ses médias
    - Modifier un produit et ses médias
    - Supprimer un produit et ses médias
- Affichage détaillé des produits

3.3. Description détaillée de l'interface



3.3.1. ① Page principale (Visiteur)

La page principale du site web est la page dont on accède en arrivant sur le site web, elle peut être résumée en cinq parties principales :

- L'affichage des produits les plus populaires
- L'affichage des produits recommandés par les administrateurs du site web
- Le bouton de connexion pour administrateur
- La barre de recherche
- La liste des mots clefs pour la recherche par mots clefs

Ces différents éléments sont tous interactifs en renvoient l'utilisateur à une partie différente du site web. Cliquer sur un des produits populaire ou recommandé par l'un des administrateurs ouvre les « ⑦ Détails du produit ». Cliquer sur le bouton de connexion renvoie l'utilisateur vers la page de « ⑤ Connexion administrateur ». Cliquer sur l'un des mots clefs présent sur la partie gauche du site web permet à l'utilisateur de filtrer la liste des produit selon un critère, il est renvoyé vers la page du « ② Résultat de recherche ». Finalement, si l'utilisateur valide une recherche, on le renvoie sur la page du « ② Résultat de recherche », puis on filtre les résultats affichés selon les mots entrés par l'utilisateur.

3.3.2. ② Résultat de recherche

La page de résultat de recherche est accessible de deux façons, on y accède quand l'utilisateur clique sur l'un des mots clefs. Il est aussi possible d'y accéder après avoir validé une recherche à l'aide de la barre de recherche.

Le but principal de cette page est de permettre à l'utilisateur d'afficher des produits regroupés par critères tel que leur titre, leur description, leur marque ou encore à l'aide d'un des mots clefs présent sur la partie gauche de l'interface.

Il existe plusieurs éléments avec lesquels on peut interagir sur cette page. Cliquer sur l'un des produits affichés nous renvoie sur la page des « ⑦ Détails du produit ». Cliquer sur le texte « Catal'info » présent dans la barre supérieure renvoie l'utilisateur à la « ① Page principale (Visiteur) » ou à la « ③ Page principale (Administrateur) » en dépendant du fait que l'utilisateur soit connecté ou non.

3.3.3. ③ Page principale (Administrateur)

La page principale de l'administrateur est uniquement accessible à un administrateur connecté à Catal'info. Cette dernière est identique à la « ① Page principale (Visiteur) » à l'exception de quelques éléments :

- La page possède un « ④ Menu déroulant en session administrateur »
- La page permet à l'administrateur de gérer des produits

Le but principal de cette page est de permettre à l'administrateur de modifier ou de supprimer rapidement des produits. L'interface est intuitive car elle est presque identique à la « ① Page principale (Visiteur) ».

Un icone d'engrenage placé dans le coin du cadre d'un produit permet à l'administrateur d'accéder au formulaire de modification du produit.

3.3.4. ④ Menu déroulant en session administrateur

Le menu déroulant est accessible uniquement depuis une session administrateur et permet à l'administrateur de se déconnecter ou encore d'accéder à la page « ⑥ Ajout d'un produit / Modification produit ».

Pour accéder au menu déroulant, l'administrateur clique tout simplement sur son nom en haut à droite de la « ③ Page principale (Administrateur) ».

3.3.5. ⑤ Connexion administrateur

On peut accéder à cette page depuis n'importe quelle autre page du site tant que l'on n'est pas connecté (statut visiteur). L'accès au formulaire est simple, il suffit de cliquer sur le lien « Connexion » se trouvant dans la barre supérieure de la majorité des pages du site.

Ce formulaire est uniquement utile à un administrateur de Catal'info, car il lui permet de se connecter et d'accéder aux formulaires de gestion de produits.

Après avoir validé son nom d'utilisateur et son mot de passe, l'administrateur est redirigé vers la « ③ Page principale (Administrateur) ». Si des informations incorrectes sont rentrées à l'intérieur des champs, on affiche un message d'erreur au visiteur et on lui propose d'entrer à nouveau ses informations.

3.3.6. ⑥ Ajout d'un produit / Modification produit

Cette page est affichée après qu'un administrateur ait cliqué sur le bouton « Ajout d'un produit » dans le menu déroulant. Une version légèrement différente de cette page est affichée lorsque on modifie les informations d'un produit, le titre est changé à « Modification produit » et les informations du produit sélectionné sont préinscris dans les différents champs du formulaire. Un produit possède les informations suivantes et il est donc nécessaire de remplir tous les champs suivants :

- Un titre
- Une marque
- Une description courte du produit
- Une description longue du produit
- La date d'ajout au magasin du produit
    - La date ou le produit va être disponible dans le catalogue en ligne
- La date « d'expiration » du produit
    - La date ou le produit est retiré automatiquement du catalogue en ligne
- Une ou des images du produit
- Des documents téléchargeables (.pdf, .xls etc…)
    - Par exemple : Manuel d'utilisateur
- Recommandation des administrateurs (« Est en première page »)
    - Cocher ce champ pour afficher le produit dans les suggestions des administrateurs

Après avoir entré la totalité des champs ou après avoir modifié l'un de ces derniers, on peut cliquer sur valider pour ajouter le produit ou appliquer les changements potentiellement apportés.

3.3.7. ⑦ Détails du produit

Les détails d'un produit sont affichés après qu'un utilisateur / administrateur ait cliqué sur l'image d'un produit. Le rôle principal des détails du produit est, comme son nom l'indique, d'afficher en détail les différentes informations du produit. Cette page reste identique pour un administrateur et un utilisateur.

Une image du produit est affichée dans le coin gauche de la boite du produit et si le produit possède différentes images, les images défilent les unes après les autres. Une description est aussi affichée en dessous de l'image du produit, des documents téléchargeables (.pdf, .xml etc…) sont aussi disponibles, voir « ⑥ Ajout d'un produit / Modification produit ».

3.4. Description des éléments de sécurité

Ici seront indiqués les différents éléments composant la sécurité de Catal'info.

3.4.1. Compte administrateur

Il est possible d'ajouter, de modifier ou encore de supprimer des produits mais uniquement à l'aide d'un compte administrateur protégé par une combinaison de nom d'utilisateur et de mot de passe.

3.4.2. Connexion en tant qu'Administrateur

Pour sécuriser la connexion à Catal'info, les mots de passes des administrateurs seront encryptés à l'aide d'une combinaison de « SHA-1 » et de « MD5 ». On s'assure que si le mot de passe est intercepté à mi-chemin avec le serveur, que la personne malveillante ne puisse rien faire avec le mot de passe.

3.4.3. Avertissement avant suppression d'un produit

Avant de supprimer un produit, une fenêtre pop-up demande à l'administrateur de confirmer son action, cela permet de réduire les chances de fausses manouvres et la suppression accidentelle d'un produit.

4. Analyse organique

4.1. Généralités

Le site web Catal'info peut être résumé en cinq parties principales :

1. La page d'accueil, pour afficher les produits les plus vus et les produits recommandés par les administrateurs.
2. Le détail produit, qui permet à une personne d'obtenir toutes les informations d'un produit. La page possède aussi un support visuel et des documents téléchargeables.
3. La partie recherche du site web qui permet à l'utilisateur de rechercher des produits à l'aide de mots clefs ou d'une barre de recherche multicritères.
4. L'identification utilisateur, qui permet à une personne de se connecter au site web
5. La partie réservée aux administrateurs pour ajouter, modifier ou supprimer des produits.

Les différentes fonctionnalités décrites seront donc classées par parties du site web. Pour plus d'informations sur chaque page de Catal'info, se rendre dans la « Description détaillée de l'interface »

Je parlerais aussi d'éléments importants tels que la base de données du site ou encore des fonctionnalités supplémentaires du site web.

4.1.1. Backups du site web

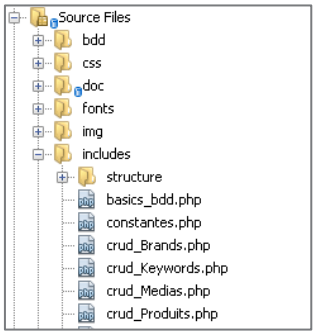
Des backups du projet sont effectués après l'ajout de chaque fonctionnalité importante du site web. Pour effectuer des backups, on utilise « gitHub » et le client « git » intégré à « NetBeans IDE ». « gitHub » me permet aussi de faire du « versioning » et de gérer plusieurs versions du projet à la fois.

En plus de ces dernières fonctions, « gitHub » permet un accès facile aux documents ou scripts du projet depuis leur site web. Cela rend le projet amovible et ajoute de la flexibilité à l'environnement de travail.

Par exemple : « Le disque dur utilisé sur le poste de travail est défectueux. Le projet n'est par contre pas en danger, car il est sauvegardé sur la plateforme de « gitHub » et peut être récupéré pour travailler sur un autre poste. »

4.1.2. Structure du site web

Les fichiers du site web sont triés et structurés de manière à être auto-intuitifs à l'utilisateur et à permettre une organisation optimale du site web.



Des dossiers sont utilisés pour trier chaque partie individuellement. Les scripts PHP sont stockés à l'intérieur du dossier « includes », les feuilles de style css à l'intérieur du dossier « css » et ainsi de suite.

4.2. Base de données

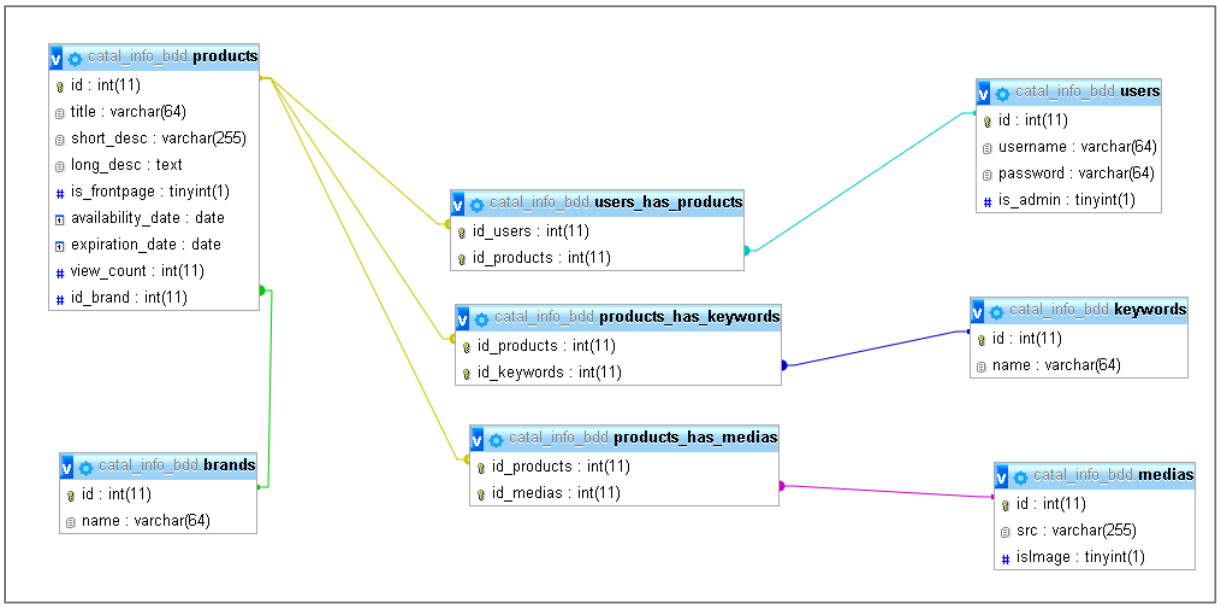
4.2.1. Introduction

Catal'info est un site de gestion de catalogue en ligne pour des revendeurs informatique, il est donc nécessaire de stocker et organiser les données dans une base de données. Pour la création de la base de données, j'ai utilisé « phpMyAdmin » (MySQL).

La base de données « catal_info_bdd » possède un total de huit tables permettant de gérer des produits, des utilisateurs, des mots clefs et même des medias.

4.2.2. Modèle relationnel de la base de données

Le modèle relationnel de la base a évolué tout au long du projet, voici la version finale du modèle relationnel de Catal'info :



4.2.3. Dictionnaire de données & Description des tables

Ci-dessous est décrit le contenu des tables et l'utilité de chaque champs pour Catal'info.

4.2.3.1. Table « brands »

La table brands contient les informations relatives aux marques des produits disponibles, elles sont identifiées par un id et possèdent un nom.

Exemple : « 1 – Asus »

Colonne	Type	Null	Défaut	Commentaires	MIME
id	int(11)	Non		L'identifiant unique de la marque.	
name	varchar(64)	Non		Le nom de la marque.	

4.2.3.2. Table « keywords »

La table des « keywords » contient les informations relatives aux mots clefs des produits ou aussi appelées « catégories ». Les mots clefs possèdent chacun un « id » pour les rendre unique et un nom.

Exemple : « 4 – Carte graphique »

Colonne	Type	Null	Défaut	Commentaires	MIME
id	int(11)	Non		L'identifiant unique du mot clef.	
name	varchar(64)	Non		Le mot clef.	

4.2.3.3. Table « medias »

La table des medias contient les informations relatives aux médias des produits (images, documents etc…). Un média possède un identifiant pour le rendre unique, une source indiquant ou se trouve le média sur le serveur et un champ « isImage » permettant de définir si le média concerné est une image.

Exemple : « 1 – up-content/img/test.png – 1 »

Colonne	Type	Null	Défaut	Commentaires	MIME
id	int(11)	Non		L'identifiant unique du média.	
src	varchar(255)	Non		Le chemin d'accès du média.	
isImage	tinyint(1)	Non		Décrit si le média est une image.	

4.2.3.4. Table « products »

La table principale de Catal'info. La table produit contient les informations relatives aux produits du site web. Chaque produit du site possède un identifiant unique (id), un titre, une description courte du produit, une description longue du produit, un champ permettant de dire si le produit doit être affiché en première page du site web, une date de disponibilité permettant de savoir quand le produit est disponible sur le catalogue, une date d'expiration permettant de savoir quand le produit va être retiré du catalogue, un nombre de vues, et une marque.

La marque du produit est en relation avec la table « brand » car un produit ne peut posséder qu'une seule marque et les marques sont des informations différentes des produits.

Exemple : « 1 – GeForce 770 – shortDesc – longDesc – 1 – 25.02.2005 – 14.03.16 – 99 – 1 »

Colonne	Type	Null	Défaut	Relié à	Commentaires	MIME
id	int(11)	Non			L'identifiant unique du produit	
title	varchar(64)	Non			Le titre du produit	
short_desc	varchar(255)	Non			Une courte description	
long_desc	text	Non			Une longue description	
is_frontpage	tinyint(1)	Non			Décrit si le produit est en première page	
availability_date	date	Non			Date de disponibilité du produit	
expiration_date	date	Non			Date d'expiration du produit	
view_count	int(11)	Non			Le nombre de vues du produit	
id_brand	int(11)	Non		brands -> id	L'id de la marque du produit (FK)	

4.2.3.5. Table « products_has_keywords »

Table intermédiaire permettant de faire une relation « many to many » entre les produits et leurs mots clefs (plusieurs produits possèdent plusieurs mots clefs).

Colonne	Type	Null	Défaut	Relié à	Commentaires	MIME
id_products	int(11)	Non		products -> id		
id_keywords	int(11)	Non		keywords -> id		

4.2.3.6. Table « products_has_medias »

Table intermédiaire permettant de faire une relation « many to many » entre les produits et leurs medias (plusieurs produits possèdent plusieurs medias).

Colonne	Type	Null	Défaut	Relié à	Commentaires	MIME
id_products	int(11)	Non		products -> id		
id_medias	int(11)	Non		medias -> id		

4.2.3.7. Table « users »

Table contenant les informations relatives aux utilisateurs et administrateurs de Catal'info. Chaque utilisateur possède un identifiant unique (id), un nom d'utilisateur utilisé pour se connecter, un mot de passe (encrypté en md5 & sha1), et un champ permettant de définir si l'utilisateur est un administrateur.

Exemple : 12 – Admin – 38758f53d77d5217d477433658b49a301f435feb – 1 »

Colonne	Type	Null	Défaut	Commentaires	MIME
id	int(11)	Non			
username	varchar(64)	Non			
password	varchar(64)	Non			
is_admin	tinyint(1)	Non			

4.2.3.8. Table « users_has_products »

Table intermédiaire permettant de faire une relation « many to many » entre les utilisateurs et leurs produits (plusieurs utilisateurs possèdent plusieurs produits). Cette table n'est actuellement pas utilisée, mais peut être utilisée pour faire un système de produits favoris ou de chariots.

Colonne	Type	Null	Défaut	Relié à	Commentaires	MIME
id_users	int(11)	Non		products -> id		
id_products	int(11)	Non		users -> id		

4.2.4. Script des fonctions basiques relatives à la base de données

Pour gérer la base de données, un script nommé « basics_bdd » a été créé. Ce dernier possède les fonctions basiques relatives à la base de données. Voici des exemples de fonctions que possède « basics_bdd » :

- Connexion à la base de données (PDO)
- Retourner le nombre d'enregistrements dans une table donnée
- Retourner un enregistrement par rapport à son « id »
- Retourner tous les enregistrements d'une table
- Supprimer un enregistrement d'une table par rapport à son « id »

« basics_bdd » permet aussi la simplification de la création de fonctions relatives à la base de données. Voici un exemple d'une fonction crée à l'aide de « basics_bdd », retournant le nombre de produits dans la base de données :

```php
function countProducts() {
    global $tableProducts;

    return countFields($tableProducts);
}
```

La taille de la fonction a été grandement réduite grâce à la création de fonctions basiques.

4.2.5. Fonctions principales relatives à la base de données

Ici seront décrites et analysées les fonctions principales se trouvant à l'intérieur du script « basics_bdd » :

4.2.5.1. Connexion à la base de données :

Voici le fonctionnement pour la connexion à la base de données : On essaye de se connecter à la base avec les informations nécessaire à la connexion contenues à l'intérieur de constantes. Puis si cela réussit, on stocke et on retourne le résultat, sinon en renvoie l'erreur de PDO.

```php
function connection() {
    try {
        $bdd = new PDO('mysql:host=' . DB_HOST .
                ';dbname=' . DB_NAME, DB_LOGIN, DB_PASS,
                array(PDO::ATTR_PERSISTENT => true));

        return $bdd;
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}
```

4.2.5.2. Récupération de tous les champs d'une table

Pour récupérer tous les champs d'une table, on commence par se connecter à la base de données. Puis on protège la chaine reçue contre l'injection SQL à l'aide de la fonction « quote ». Par la suite on crée une requête permettant de récupérer tous les champs de la table reçue en paramètre. On prépare la requête avant de l'exécuter.

Finalement on récupère les données de la table à l'intérieur d'un objet et on renvoie ces informations à l'aide de « return ».

```php
function getAllFields($table) {
    $dbc = connection();
    $dbc->quote($table);
    $req = "SELECT * FROM $table";

    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();
    return $data;
}
```

4.2.5.3. Récupération d'un champ par son identifiant unique

Pour récupérer un champ par son identifiant, on commence par se connecter à la base de données, et comme précédemment, on protège la chaine reçue contre l'injection SQL avec la fonction « quote ». On crée notre requête, puis on la prépare avant de l'exécuter.

Finalement, on récupère le champ dans un type donnée en paramètre, le paramètre est surchargé, ce qui permet à la fonction d'être plus robuste. Par défaut, on renvoie un objet contenant les informations du champ récupéré.

```php
function getFieldById($id, $table, $type = PDO::FETCH_OBJ) {
    $dbc = connection();
    $dbc->quote($table);
    $req = "SELECT * FROM $table WHERE id=:id";
    $requPrep = $dbc->prepare($req);
    $requPrep->bindParam(':id', $id, PDO::PARAM_INT);
    $requPrep->execute();

    return $requPrep->fetch($type);
}
```

4.2.5.4. Compter le nombre de champs d'une table

Pour compter les champs d'une table, la procédure est très similaire à la récupération de tous les champs d'une table. On procède de la même manière en se connectant et en protégeant nos paramètres contre l'injection SQL.

Puis on crée notre requête en utilisant la fonction MySQL « COUNT » qui permet de compter les champs donnés en paramètre, dans notre cas, on compte tous les champs de la table. Finalement on prépare la requête, on l'envoie et on récupère les données au format « integer ».

```php
function countFields($table) {
    $dbc = connection();
    $dbc->quote($table);
    $req = "SELECT COUNT(*) FROM $table";

    $requPrep = $dbc->prepare($req);
    $requPrep->execute();

    $number = $requPrep->fetch();
    $requPrep->closeCursor();
    return $number[0];
}
```

4.2.5.5. Suppression d'un champ par son identifiant

Pour supprimer un champ à l'aide de son identifiant unique (id), on commence par se connecter à la base, puis on protège le paramètre reçu contre l'injection SQL.

On crée notre requête, ou l'on supprime un élément de la table donnée en paramètre où l'identifiant du champ à supprimer est égal à celui reçu en paramètre. Finalement on renvoie le résultat de la requête si tout s'est passé comme prévu.

```php
function deleteFieldById($id, $table) {
    $dbc = connection();
    $dbc->quote($table);
    $req = "DELETE FROM $table WHERE id=:id";

    $requPrep = $dbc->prepare($req);
    $requPrep->bindParam(':id', $id, PDO::PARAM_INT);
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();
}
```

4.3. Page d'accueil

4.3.1. Affichage des produits les plus populaires

Pour afficher des produits par popularité, le fonctionnement est le suivant :

On utilise une fonction « getMostViewedProducts() » se trouvant dans le « crud_Produits ». La fonction, à l'aide d'une requête MySQL, récupère depuis la base de données la liste des produits triés par nombre de vues ainsi que leur medias. Il est important de récupérer les medias des produits pour pouvoir afficher un aperçu du produit par la suite. Les produits récupérés sont stockés dans un tableau d'objets. On s'assure aussi dans la requête MySQL que le media récupéré soit bien une image grâce au champ « isImage ». On limite aussi le nombre de produits affichés à l'aide d'une constante nommée « NUMBER_OF_TOP_PRODUCTS » qui sera passée en paramètre. Voici la fonction utilisée :

```php
function getMostViewedProducts($nbProductsShown) {
    global $tableProducts;

    $dbc = connection();
    $dbc->quote($tableProducts);
    $req = 'SELECT DISTINCT p.id AS idProduct, p.title as productTitle,
            . 'p.short_desc, MIN(m.src) AS mediaSource,'
            . ' m.isImage, p.view_count, p.is_frontpage '
            . 'FROM products AS p '
            . 'INNER JOIN products_has_medias AS pm ON p.id = pm.id_products '
            . 'INNER JOIN medias AS m ON pm.id_medias = m.id '
            . 'WHERE m.isImage = 1 '
            . 'AND NOW() > availability_date '
            . 'AND NOW() < expiration_date '
            . 'GROUP BY p.title '
            . 'ORDER BY view_count DESC '
            . 'LIMIT ' . $nbProductsShown;

    $requPrep = $dbc->prepare($req); // on prépare notre requête
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();

    return $data;
}
```

Pour convertir la liste des produits triés vers un format HTML, on utilise la fonction « structMostViewedProducts() ». Cette fonction va tout d'abord récupérer la liste des produits à l'aide de « getMostViewedProducts() », puis, pour chaque produit, elle va ajouter un élément à la variable « $str ». On remplit la variable « $str » avec le code HTML pour l'affichage de chaque produits et on remplace certains éléments par les éléments récupérés de la base de données, tel que le titre du produit, et une courte description. Après avoir parcouru chaque produit, on renvoie la variable contenant le code HTML.

Voici a quoi pourrait ressembler la fonction « structMostViewedProducts() »

```
function structMostViewedProducts() {
    $products = getMostViewedProducts(NUMBER_OF_TOP_PRODUCTS);
    $str = '';

    foreach ($products as $product) {
        $str.= '<div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
                    <div class="panel panel-success">
                        <div class="panel-heading">
                            <a href="detail.php?id=' . $product->idProduct . '"></a>
                        </div>
                        <div class="panel-body">
                            <div class="product-container thumbnail">
                                <a href="detail.php?id=' . $product->idProduct . '"></a>
                            </div>
                            <hr/>
                            <div class="panel-footer" style="float: left;">
                                <p>
                                    ' . $product->short_desc . '
                                </p>
                            </div>
                        </div>
                    </div>
                </div>';
    }

    return $str;
}
```

Après avoir créé la fonction « structMostViewedProducts() », il suffit d'afficher le contenu

Voici à quoi pourrait ressembler la partie s'occupant de l'affichage des produits les plus vu dans la page principale du site web :

```
<div class="row">
    <?php
    echo structMostViewedProducts();
    ?>
</div>
```

4.3.2. Affichage des produits recommandés

Le fonctionnement derrière l'affichage des produits recommandés est très similaire à celui de l'affichage des produits les plus vus. Il est le suivant :

On utilise une fonction «getRecommendedProducts() » se trouvant dans le « crud_Produits ». La fonction, à l'aide d'une requête MySQL, récupère depuis la base de données la liste des produits recommandés par les administrateurs ainsi que leur medias. Comme indiqué précédemment, il est important de récupérer les medias des produits pour pouvoir afficher un aperçu du produit par la suite. Les produits récupérés sont stockés dans un tableau d'objets. Un produit recommandé par un administrateur est définit par un champ « is_frontpage » qui est égal à true. On s'assure aussi dans la requête MySQL que le media récupéré soit bien une image grâce au champ « isImage ». Comparé à l'affichage des produits les plus vus on ne limite pas le nombre de produits affichés, les administrateurs du site sont en charge de choisir quel produit mettre en première page.

Voici a quoi pourrait ressembler la fonction « getRecommendedProducts() » :

```php
function getRecommendedProducts() {
    global $tableProducts;

    $dbc = connection();
    $dbc->quote($tableProducts);

    $req = 'SELECT DISTINCT p.id AS idProduct, p.title as productTitle,
            . ' p.short_desc, MIN(m.src) AS mediaSource,'
            . ' m.isImage, p.view_count, p.is_frontpage '
            . 'FROM products AS p '
            . 'INNER JOIN products_has_medias AS pm ON p.id = pm.id_products '
            . 'INNER JOIN medias AS m ON pm.id_medias = m.id '
            . 'WHERE m.isImage = 1 '
            . 'AND is_frontpage=true '
            . 'AND NOW() > availability_date '
            . 'AND NOW() < expiration_date '
            . 'GROUP BY p.title '
            . 'ORDER BY p.title ASC ';

    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();

    return $data;
}
```

Comme précédemment, pour convertir la liste des produits triés vers un format HTML, on utilise la fonction « structRecommendedProducts() ». Cette fonction va tout d'abord récupérer la liste des produits à l'aide de « getRecommendedProducts () », puis, pour chaque produit, elle va ajouter un élément à la variable « $str ». On remplit la variable « $str » avec le code HTML pour l'affichage de chaque produits et on remplace certains éléments par les éléments récupérés de la base de données, tel que le titre du produit, et une courte description. Après avoir parcouru chaque produit, on renvoie la variable contenant le code HTML.

Voici a quoi pourrait ressembler la fonction « structRecommendedProducts() » :

```
function structRecommendedProducts() {
    $products = getRecommendedProducts();
    $str = '';

    foreach ($products as $product) {
        $str.= '<div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
                    <div class="panel panel-success">
                        <div class="panel-heading">
                            <a href="detail.php?id=' . $product->idProduct . '"></a>
                        </div>
                        <div class="panel-body">
                            <div class="product-container thumbnail">
                                <a href="detail.php?id=' . $product->idProduct . '"></a>
                            </div>
                            <hr/>
                            <div class="panel-footer" style="float: left;">
                                <p>
                                    ' . $product->short_desc . '
                                </p>
                            </div>
                        </div>
                    </div>
                </div>';
    }

    return $str;
}
```

Après avoir créé la fonction « structRecommendedProducts () », il suffit d'afficher le contenu

Voici à quoi pourrait ressembler la partie s'occupant de l'affichage des produits recommandés dans la page principale du site web :

```
<div class="row">
    <?php
    echo structRecommendedProducts();
    ?>
</div>
```

4.3.3. Affichage en-tête dynamique

Le fonctionnement de l'en-tête dynamique est simple : On affiche de différents en-têtes en fonction de l'utilisateur connecté.

Pour cela, j'ai créé plusieurs fonctions me permettent de savoir si un utilisateur est connecté ou encore si l'utilisateur connecté est un administrateur.

Pour savoir si un utilisateur est connecté, on regarde si un identifiant a été initialisé dans la session de l'utilisateur. Après avoir effectué le test, on renvoie « true » si l'utilisateur est connecté, et « false » si l'utilisateur n'est actuellement pas connecté.

Voici un exemple du code que pourrait contenir la fonction « isConnected() » :

```
function isConnected() {
    return (isset($_SESSION['id']));
}
```

Pour déterminer si un utilisateur est un administrateur, le principe est le même, mais on teste plusieurs éléments.

Il faut vérifier que l'élément « admin » soit initialisé dans la session de l'utilisateur. Puis on regarde si cet élément est égal à 1. Si ces deux conditions sont respectées, la fonction renvoie « true » pour indiquer que l'utilisateur est bien un administrateur, sinon elle renvoie la valeur « false ». Voici à quoi pourrait ressembler le code permettant de vérifier si l'utilisateur connecté est un administrateur :

```
function isAdmin() {
    if (isset($_SESSION['admin'])) {
        if ($_SESSION['admin'] == 1) {
            $result = true;
        }
        else {
            $result = false;
        }
    }
    else {
        $result = false;
    }
    return ($result);
}
```

Après avoir déterminé le statut actuel de l'utilisateur (déconnecté, connecté, admin), on va afficher un en-tête différent pour chaque statut.

Si l'utilisateur est déconnecté, on affiche un bouton pour se connecter. Pour plus d'informations sur la connexion de l'utilisateur, se référer à « Identification utilisateur>Connexion utilisateur ».

Si l'utilisateur est connecté mais n'est pas un administrateur, on lui affiche un menu déroulant portant son nom d'utilisateur et lui permettant de se déconnecter. Pour plus d'informations sur la déconnexion de l'utilisateur, se référer à « Identification utilisateur>Déconnexion utilisateur ».

Si l'utilisateur est un administrateur, on affiche le même menu déroulant que l'utilisateur standard, mais avec un lien supplémentaire pour rediriger l'administrateur vers l'ajout de produits (cf. doc. Administration>Ajout produit). Voici quelques lignes de code représentant le fonctionnement de la fonction « getHeader() » :

```
function getHeader(){
    if (isConnected()){
        //En tête utilisateur ici (Connecté)
        if (isAdmin(){
            //En tête administrateur ici (Administrateur)
        }
    }
    else{
        //En tête visiteur ici (Déconnecté)
    }
}
```

4.4. Détail produit

4.4.1. Affichage du détail produit

L'affichage détaillé du produit se fait en plusieurs parties :

On commence par envoyer un paramètre en « GET » contenant l'identifiant du produit à afficher. Puis grâce à cet identifiant, on va chercher à l'aide d'une requête MySQL, les informations détaillée du produit ainsi que sa marque et ses médias. Finalement, on affiche les informations récupérées à l'aide d'une fonction permettant de structurer la page du produit.

Si l'identifiant du produit récupéré n'est pas un produit existant on renvoie l'utilisateur à la page d'accueil. Voici le code pour la redirection :

```php
if ($product == NULL) {
    header('Location: index.php');
}
```

Voici la fonction « getProductDetailledById() » qui permet de récupérer tous les champs nécessaire à l'affichage du détail produit :

```php
function getProductDetailsById($id) {
    global $tableProducts;

    $dbc = connection();
    $dbc->quote($tableProducts);
    $req = "SELECT p.id AS idProduit, p.title, p.short_desc,"
            . " p.long_desc, p.is_frontpage,"
            . " p.availability_date, p.expiration_date, p.view_count,"
            . " p.id_brand, b.id AS idBrand, b.name AS brandName, "
            . "(NOW() < availability_date OR NOW() > expiration_date) AS isExpired"
            . " FROM $tableProducts AS p"
            . " INNER JOIN brands as b ON p.id_brand = b.id"
            . " WHERE p.id = $id";


    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $data = $requPrep->fetch(PDO::FETCH_OBJ);
    $requPrep->closeCursor();

    return $data;
}
```

Dans la requête présente ci-dessus, on récupère un champ « isExpired » qui nous indique si le produit est expiré. Pour vérifier si un produit est expiré, on regarde si la date actuelle se trouve entre la date de disponibilité et la date d'expiration du produit. On affiche cette information à l'utilisateur accordement suivit d'un icone pour faciliter la visibilité.

4.4.2. Gestion des médias du détail produit

Les medias des produits peuvent-être résumés en deux catégories :

1. Les images (.png, .gif, .jpg, .jpeg)
2. Les autres (.pdf, .docx, .xls etc...)

En dépendant de la catégorie du media, on les affiche dans le détail du produit à deux endroits différents. Les images sont disposées à l'intérieur d'un « carousel » à côté de la description du produit. Des flèches sont disposées à droite et à gauche du « carousel » pour permettre à l'utilisateur de naviguer entre les images et des indicateurs de positions sont disposés au centre de la partie inférieure de l'image.

Les medias classés dans la catégorie « autres » sont affichés à l'aide d'une « medialist » se trouvant en bas de la page et disposent chacun d'un lien de téléchargement.

Le détail produit possède malgré cela de petites subtilités. En effet, si le produit possède une seule image, le « carousel » ne possèdera ni de contrôles, ni d'indicateurs de position. Les différentes images disposées à l'intérieur du « carousel » doivent aussi toutes faire la même hauteur, tout en gardant les proportions initiales de l'image. On doit donc pouvoir garder une mise en page homogène. Pour cela, les images stockées à l'intérieur du carousel sont toutes stockées dans un « container » intermédiaire à hauteur fixe.

Pour différentier les medias, un champ dans la table medias nommé « isImage » sert à décrire si le produit est une image. Pour trier les medias on va donc procéder de la manière suivante :

On récupère chaque media, puis, si le media est une image, on le stocke dans un tableau d'images, sinon, on le stocke dans un tableau d'autres fichiers. Voici a quoi pourrait ressembler le code permettant de séparer les images en deux catégories :

```
foreach ($medias as $media) {
    if ($media->isImage) {
        $i++;
        $arrayImage[$i] = $media->mediaSource;
    } else {
        $j++;
        $arrayOthers[$j] = $media->mediaSource;
    }
}
```

On va par la suite parcourir les deux tableaux et créer le code HTML propre à chaque medias et contenant les informations récupérées. Voici un exemple de code permettant de remplir une variable avec le contenu des medias « autres » :

```
foreach ($arrayOthers as $other) {
    $filename = substr($other, strlen(OTHER_FOLDER));
    $otherMedia .= 'Nom media : '.$other->name.' etc...';
}
```

On procède de la même manière pour les images à l'exception que l'on retire les contrôles du « carousel » si le produit ne possède qu'une seule image. En supprimant les contrôles du carousel on évite d'induire en erreur l'utilisateur.

Notez que l'on crée un nom pour le produit qui servira de lien de téléchargement par la suite. Le nom est créé en soustrayant la longueur de l'arborescence du dossier où sont stockés les medias au chemin complet.

Par exemple : « up-content/others/test_2.pdf ». On soustrait « up-content/others/ » à notre première chaine de caractère et on récupère le nom final du fichier : « test_2.pdf ». C'est ce nom de fichier qui sera visible en tant que lien de téléchargement dans le conteneur à fichiers téléchargeables.

4.4.3. Ajout de vues

Pour ajouter des vues à un produit, on lance une fonction à chaque fois que la page est lancée. Cette fonction va récupérer le nombre de vues d'un produit grâce à son identifiant unique et va ajouter une vue au total grâce à une requête « UPDATE ». Après avoir récupéré le nombre de vues d'un produit, on affiche le nombre de vues en bas de la page du détail produit à l'aide d'une fonction de structure.

Voici une fonction permettant d'ajouter une vue à un produit en fonction de son identifiant :

```php
function addViewById($id) {
    global $tableProducts;

    $dbc = connection();
    $dbc->quote($tableProducts);
    $dbc->quote($id);
    $req = "UPDATE $tableProducts SET view_count = view_count+1 WHERE id = $id";
    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $requPrep->closeCursor();
}
```

Voici la fonction permettant de structurer les vues vers un format HTML :

```php
function structViewCount($id) {
    $product = getProductById($id);
    $str = '<span>Nombre de vues : '
             . $product->view_count .
             ' <span class="glyphicon glyphicon-eye-open"></span></span>';

    return $str;
}
```

L'ajout de vue s'effectue à chaque fois qu'un utilisateur charge ou recharge la page, une amélioration envisageable est de limiter le nombre de vues ajoutées à une seule par personne. Pour plus d'informations se rendre dans la catégorie « Améliorations envisageables » de la documentation technique.

4.5. Recherches

4.5.1. Liste déroulante des catégories

Un bouton dépliant une liste déroulante se trouve en dessous de l'en-tête et permet à l'utilisateur à accéder rapidement à la recherche par catégories du site web. Le menu est composé d'un bouton et d'une liste HTML cachée, un script JavaScript va par la suite afficher chaque élément caché de la liste quand on clique sur le bouton. La liste est créée dynamiquement avec la liste des mots-clefs provenant la base de données et triée par ordre alphabétique.

Voici la fonction s'occupant de l'affichage des mots clefs à l'intérieur de la liste déroulante.

```php
function structKeywordsList() {
    $keywords = getAllKeywordsSorted();
    $str = '';

    //On affiche un lien pour chaque keywords de la base
    foreach ($keywords as $keyword) {
        $str.= '<a href="search.php?querryKW=' . $keyword->name . '" class="list-group-item">
                        ' . strtoupper($keyword->name) . '
                    </a>';
    }

    return $str;
}
```

On ajoute aussi un lien « href » vers la page de recherche pour permettre la recherche par mots-clefs. Pour plus d'informations sur la recherche par mot clefs, se rendre dans la catégorie « Recherche par mot-clef » de la documentation technique.

4.5.2. Recherche par mot-clef

Pour la recherche par mot-clef, on commence tout d'abord par récupérer la requête de recherche de l'utilisateur. Dans le cas de la recherche par mot-clef, un paramètre envoyé en « get » contient le nom de la catégorie à rechercher (keyword).

On va par la suite exécuter une fonction qui va, à l'aide d'une requête MySQL, récupérer la liste des produits possédant le même mot-clef que la requête de l'utilisateur. Par exemple : L'utilisateur clique sur « ALIMENTATIONS », on va rechercher la liste des produits liés au mot-clef « ALIMENTATIONS » et on les affiche sur la page.

Voici la fonction utilisée pour récupérer les produits recherchés à l'aide d'un mot-clef.

```php
function searchForProductWithKeywords($querry) {
    global $tableProducts;
    $dbc = connection();
    $dbc->quote($tableProducts);
    $req = 'SELECT DISTINCT p.id AS idProduct, p.title as productTitle, '
            . 'p.short_desc, p.view_count, p.is_frontpage, b.name AS brandName, '
            . 'k.name AS keywordName '
            . 'FROM ' . $tableProducts . ' AS p '
            . 'INNER JOIN brands AS b ON p.id_brand = b.id '
            . 'INNER JOIN products_has_keywords AS pk ON p.id = pk.id_products '
            . 'INNER JOIN keywords AS k ON pk.id_keywords = k.id '
            . 'WHERE k.name LIKE "%' . $querry . '%" '
            . 'GROUP BY p.title '
            . 'ORDER BY p.title DESC';

    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();

    return $data;
}
```

Une des subtilités de cette procédure est le fait que l'on ne peut pas récupérer en une seule requête les medias du produit et les produits liés aux keywords. On va donc séparer cette procédure en deux parties, et créer une fonction pour structurer la liste des produits recherchés par mot-clef.

Une autre subtilité est l'affichage d'un aperçu du produit en image, en effet, un produit possède différents type de médias tous confondus. Pour résoudre ce problème, on commence par vérifier le type du media avec son champ « isImage », on récupère la totalité des images du produit dans un nouveau tableau que l'on va parcourir pour récupérer la première image du produit.

Voici comment se passe le tri des médias des produits recherchés par mot-clef :

```
foreach ($products as $product) {
    //On récupère la première image du produit
    $medias = getProductMediasById($product->idProduct);
    $imgProduct = '';
    foreach ($medias as $media) {
        if ($media->isImage) {
            $imgProduct[] = $media->mediaSource;
        }
    }
```

Après avoir trié les medias, il ne nous reste plus qu'à ajouter « $imgProduct[0] » dans une balise « img » pour que la première image du produit s'affiche.

Actuellement, la recherche de produit par mot-clef ne possède pas de pagination ou de limitation du nombre de produits par page, pour plus d'informations sur les amélioration envisageables du site web, se rendre dans la catégorie « Amélioration envisageables » de la documentation.

4.5.3. Recherche multicritères

Pour la recherche multicritères, on commence tout d'abord par récupérer la requête de recherche de l'utilisateur. Dans le cas de la recherche multicritères, un paramètre envoyé en « get » contient les critères entrés par l'utilisateur dans la barre de recherche.

On va par la suite exécuter une fonction qui va, à l'aide d'une requête MySQL, récupérer la liste des produits et les medias des produits ou les critères entrés par l'utilisateur sont présent dans un des différents champs des produits (titre, description courte & longue, marque). Pour réaliser cela, on va concaténer les différents champs et utiliser la commande « LIKE » pour comparer ces derniers à la requête de l'utilisateur.

Voici la fonction utilisée pour récupérer les produits recherchés à l'aide de la barre de recherche multicritères :

```php
function searchForProduct($querry) {
    global $tableProducts;

    $dbc = connection();
    $dbc->quote($tableProducts);
    $req = 'SELECT DISTINCT p.id AS idProduct, p.title as productTitle,'
            . ' p.short_desc, MIN(m.src) AS mediaSource,'
            . ' m.isImage, p.view_count, p.is_frontpage, b.name AS brandName '
            . 'FROM ' . $tableProducts . ' AS p '
            . 'INNER JOIN products_has_medias AS pm ON p.id = pm.id_products '
            . 'INNER JOIN medias AS m ON pm.id_medias = m.id '
            . 'INNER JOIN brands AS b ON p.id_brand = b.id '
            . 'WHERE Concat(p.title, p.short_desc, p.long_desc, b.name) '
            . 'LIKE "%' . $querry . '%" '
            . 'AND m.isImage = 1 GROUP BY p.title ORDER BY p.title DESC';

    $requPrep = $dbc->prepare($req);
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();

    return $data;
}
```

Après avoir récupéré la liste des produits avec leur medias et leurs marques respectives, on va les structurer et les convertir en code HTML à l'aide d'une fonction. Comparé à la recherche par mot-clef, il est possible de récupérer les produits et leurs medias en une seule requête, ce qui diminue grandement le nombre de contrôles à effectuer.

4.6. Identification utilisateur

4.6.1. Connexion utilisateur

Le fonctionnement de la connexion utilisateur est plutôt simple. On commence par récupérer les informations envoyé en « POST » par l'utilisateur. Puis on passe en paramètre ces informations dans une fonction nommée « userConnect ». Cette fonction va tout d'abord récupérer le pseudo entré par l'utilisateur et va « regarder » si le pseudo existe dans la base de données, si il existe, on récupère les informations de l'utilisateur dans une variable, sinon on retourne « FALSE ».

Après avoir récupéré les informations de l'utilisateur, on va « hacher » le mot de passe entré par l'utilisateur et le comparer à celui présent dans la base de données. Pour hacher le mot de passe on utilise la fonction « hashPerso » qui utilise une combinaison de « MD5 » et de « SHA-1 » pour encrypter et protéger le mot de passe de l'utilisateur, pour plus d'informations, se rendre ci-dessous pour le détail de l'algorithme de « hachage ».

Si les deux mots de passes sont identiques, on va stocker les informations de l'utilisateur dans une session. La session possède trois champs, « id » qui correspond à l'identifiant unique de l'utilisateur, « username » qui correspond au pseudo de l'utilisateur et « admin », un booléen qui indique si l'utilisateur est un administrateur ou non. Finalement, on renvoie « TRUE » si l'utilisateur s'est connecté correctement, et « FALSE » si une erreur s'est produite.

Voici la fonction permettant de connecter un utilisateur

```php
function userConnect($username, $password) {
    $connect = false;
    $_SESSION['username']= $username;
    $user = getUserbyPseudo($username);
    if ($user != NULL && $user->password === hashPerso($password)) {
        $_SESSION['id'] = $user->id;
        $_SESSION['username'] = $user->username;
        $_SESSION['admin'] = $user->is_admin;
        $connect = TRUE;
    }
    return $connect;
}
```

Voici la fonction qui permet de « hacher » les mots de passes des utilisateurs :

```php
function hashPerso($password) {
    return sha1(md5($password));
}
```

Déconnexion utilisateur

Pour déconnecter un utilisateur, on redirige l'utilisateur vers un page nommé « logout.php ». Dans cette page, on supprime la session de l'utilisateur avant de le rediriger vers l'accueil.

Voici à quoi ressemble le code pour déconnecter l'utilisateur :

```php
session_start();
session_unset();
session_destroy();

header('location: index.php');
```

4.7. Administration

4.7.1. Ajout produit

4.7.1.1. Introduction

Pour accéder au formulaire d'ajout de produit, il est nécessaire d'être connecté sur une session administrateur. Si un utilisateur ou un visiteur accède à la page d'ajout de produit, il est automatiquement redirigé vers l'accueil. Après avoir accédé à la page, on vérifie le mode de la page avec un paramètre envoyé en « GET », si « editMode » est égal à 1, on va afficher un formulaire de modification de produit, dans le cas contraire, on affiche le formulaire d'ajout. Dans notre cas, nous allons partir du principe que le paramètre « editMode » n'est pas égal à 1.

```
if ($editMode == 1) {
    //Modification du produit
    //On récupère les informations du produit grâce à son id...
}
else {
    //Ajout du produit
}
```

Après avoir bien vérifié le fait d'être en mode d'ajout, on affiche le formulaire accordement. Le formulaire permet à l'utilisateur de remplir les informations relatives à un produit tel que son nom, une description courte, une description longue etc… L'utilisateur peut aussi ajouter des médias, sélectionner une marque depuis un menu déroulant, et même sélectionner différents mots-clefs.

4.7.1.2. Ajout de la marque du produit :

Pour assigner une marque au produit, on commence par récupérer la liste des marques disponibles triées par ordre alphabétique depuis la base de données. Puis on les stocke à l'intérieur d'un menu déroulant (balise « select »).

Chaque élément de la liste déroulante possède un attribut « value » qui correspond à l'id unique de la marque. L'attribut « value » est aussi la valeur récupérée quand on envoie le formulaire. Il nous suffit donc de récupérer la valeur de l'id de la marque sélectionnée et de la stocker dans une variable.

Voici à quoi pourrait ressembler la liste des produits :

```
<select class="form-control" name="brands" required="">
    <option value="10">AMD</option>
    <option value="21">Apple</option>
    <option value="2">Asus</option>
    <option value="20">BitFenix</option>
    <!-- [...] -->
</select>
```

4.7.1.3. Ajout de mots-clefs au produit :

Pour ajouter des mots-clefs au produit, le principe est très similaire à celui de l'ajout de la marque. On commence par récupérer la liste de tous les mots clefs triés par ordre alphabétique depuis la base de données, puis on les stocke à l'intérieur d'un sélecteur multiple (balise « select » avec l'attribut « multiple »).

Chaque élément de la liste des mots-clefs possède un attribut « value » qui correspond à l'id unique des mot-clef sélectionnés. L'attribut « value » est aussi la valeur récupérée quand on envoie le formulaire. Il nous suffit donc de récupérer les différents identifiants des mots-clefs sélectionnés après l'envoi du formulaire, et de les stocker dans un tableau.

On ira par la suite parcourir ce tableau pour ajouter les mots clefs au produit (voir « Ajout du produit dans la base de données »).

4.7.1.4. Ajout de medias au produit :

Pour gérer l'ajout des médias, on utilise une balise « input » de type « file » avec les attributs « multiple » & « required », ce qui permet d'avoir un « dialog » d'ajout de fichiers multiples, l'attribut « required » empêche l'utilisateur d'envoyer un produit sans médias. Notez que le « dialog » d'ajout de fichier dépend du système d'exploitation et est compatible avec des périphériques mobiles.

Après avoir créé la balise de type « file », les fichiers seront automatiquement ajoutés au serveur dans un dossier temporaire à chaque fois qu'un utilisateur envoie le formulaire. Les chemins d'accès vers ces fichiers sont stockés à l'intérieur de la variable « $_FILE », on va donc vérifier si les fichiers sont valides avant de les renommer et de les déplacer dans le dossier « up-content ».

Pour vérifier les fichiers, on commence tout d'abord par vérifier si le media envoyé n'a pas renvoyé d'erreur. Puis on vérifie si le fichier à bien été déplacé dans le dossier temporaire du serveur. On va par la suite vérifier que le fichier ne dépasse pas la taille maximale imposée par le serveur, dans notre cas la limite est de 5MB.

Voici à quoi pourrait ressembler les différents tests effectués sur les fichiers :

```
//Pour chaque fichiers envoyés...
foreach ($uploadedFiles["medias"]["error"] as $key => $value) {
    //Si l'upload n'a pas eu d'erreurs
    if ($value == UPLOAD_ERR_OK) {
        //Si le fichier existe
        if (file_exists($uploadedFiles["medias"]["tmp_name"][$key])) {
            //On vérifie que la taille de fichier n'exède pas la taille maximale
            if (filesize($uploadedFiles["medias"]["tmp_name"] [$key]) < MAX_SIZE) {
            }
        }
    }
}
```

Après avoir vérifié les fichiers, on va vérifier le type MIME du fichier envoyé, si le type MIME du fichier correspond à celui d'une image, on le stockera par la suite dans le dossier « up-content/img », sinon, on le stocke dans le dossier « up-content/others ». Après s'être assuré du type du fichier, on va générer un identifiant unique qui sera ajouté au nom du fichier pour éviter tout conflits de nom de fichiers. Pour obtenir le nom final du fichier, on récupère les 10 premiers caractères du nom de fichier initial et on lui ajoute l'identifiant unique et son extension. Après avoir obtenu le nouveau nom du fichier, on déplace le fichier dans son dossier attribué et on lui assigne son nouveau nom à l'aide de la fonction « move_uploaded_file » :

```
move_uploaded_file($uploadedFiles["medias"]["tmp_name"][$key], $filename);
```

4.7.1.5. Récupération des données du produit :

Pour récupérer les données des produits après l'envoie du formulaire, on stocke dans des variables les informations récupérées de la variable « $_POST à l'aide de la fonction « filter_input ». Par exemple :

```
$title = filter_input(INPUT_POST, 'title');
```

4.7.1.6. Ajout du produit dans la base :

Après avoir récupéré la totalité des informations nécessaire à l'ajout du produit, on contrôle toutes les informations envoyées. Puis on regarde si on est en train d'éditer le produit (variable « editMode = 1 »), si c'est le cas on va appeler la fonction « updateProduct » (voir « Modification produit »), sinon on utilise la fonction « addProduct ». Voici le fonctionnement de cette dernière :

```
function addProduct($title, $shortDesc, $longDesc, $isFrontpage, $startDate,
$endDate, $idBrand) {
    global $tableProducts;
    $viewCount = 0;

    $dbc = connection();
    $req = "INSERT INTO $tableProducts (title, short_desc, long_desc, "
            . "is_frontpage, availability_date, expiration_date, "
            . "view_count, id_brand) "
            . "VALUES (:title, :shortDesc, :longDesc, :isFrontpage, "
            . ":startDate, :endDate, :viewCount, :idBrand)";

    $requPrep = $dbc->prepare($req); // on prépare notre requête
    $requPrep->bindParam(':title', $title, PDO::PARAM_STR);
    $requPrep->bindParam(':shortDesc', $shortDesc, PDO::PARAM_STR);
    $requPrep->bindParam(':longDesc', $longDesc, PDO::PARAM_STR);
    $requPrep->bindParam(':isFrontpage', $isFrontpage, PDO::PARAM_BOOL);
    $requPrep->bindParam(':startDate', $startDate, PDO::PARAM_STR);
    $requPrep->bindParam(':endDate', $endDate, PDO::PARAM_STR);
    $requPrep->bindParam(':viewCount', $viewCount, PDO::PARAM_INT);
    $requPrep->bindParam(':idBrand', $idBrand, PDO::PARAM_INT);
    $requPrep->execute();
    $requPrep->closeCursor();
    return $dbc->lastInsertId();
}
```

On passe toutes les informations entrées par l'utilisateur en paramètre, puis on se connecte à la base de données. On initialise par la suite le nombre de vue du produit manuellement, le produit ne possède pas de vue par défaut. On va ensuite créer notre requête d'insertion de donnée et grâce à la fonction « bindParam » de PDO, on va insérer toutes les variables passées en paramètre à l'intérieur de la requête MySQL. Finalement, on prépare et on exécute la requête MySQL. On renvoie à l'utilisateur l'id du produit ajouté si la requête s'est déroulée comme prévu.

4.7.2. Modification produit

4.7.2.1. Introduction

Pour accéder au formulaire de modification de produit, il est nécessaire d'être connecté sur une session administrateur, puis il faut se rendre sur le détail d'un produit et cliquer sur le bouton « Modifier le produit ». Si un utilisateur ou un visiteur accède à la page de modification de produit, il est automatiquement redirigé vers l'accueil. Après avoir accédé à la page, on vérifie le mode de la page avec un paramètre envoyé en « GET », si « editMode » est égal à 1, on va afficher un formulaire de modification de produit, dans le cas contraire, on affiche le formulaire d'ajout. Dans notre cas, nous allons partir du principe que le paramètre « editMode » est égal à 1. Quand on accède la page en mode « modification » on passe aussi en paramètre l'identifiant du produit à modifier.

```
if ($editMode == 1) {
    //Modification du produit
    //On récupère les informations du produit grâce à son id...
}
else {
    //Ajout du produit
}
```

Après avoir bien vérifié le fait d'être en mode modification, on affiche le formulaire accordement. Le formulaire pré remplit automatiquement la totalité des champs du formulaire, et affiche les medias du produit dans une liste. Chaque médias de la liste possède une image représentative, pour une image, on affiche sa miniature, pour un document autre on affiche un icone de document.

4.7.2.2. Pré remplissage des champs

Pour pré remplir chaque champ du formulaire, on commence tout d'abord par récupérer la totalité des informations relatives au produit grâce à son identifiant passé en « GET ».

```
$product = getProductDetailsById($id);
$productKeywords = getProductKeywordsById($id, PDO::FETCH_ASSOC);
$productBrand = $product->id_brand;
$productMedias = getProductMediasById($id);
```

Puis on va par la suite remplir les champs du formulaire avec la commande « echo » et les informations récupérées. Par exemple, remplir le champ titre du formulaire :

```
<label class="">Nom du produit :</label>
<input class="form-control" name="title" type="text" value="
      <?php echo $txtTitle; ?>
"required/>
```

La variable « txtTitle » est le texte contenu dans le champ « title » récupéré de la base de données.

4.7.2.3. Pré remplissage des marques

La procédure du pré remplissage des marques est similaire à celle des champs mais possède la subtilité d'être à l'intérieur d'un menu déroulant. Pour cela, il va falloir comparer l'id de la marque du produit qui est en train d'être modifié avec les marques contenues à l'intérieur de la liste déroulante.

On procède de la manière suivante : On sélectionne le produit au moment où la liste est créée pour éviter d'envoyer des requêtes en trop au serveur MySQL. Quand on crée la liste déroulante, on regarde si la marque parcourue est égale à celle du produit modifié, si cela est le cas, on lui ajoute l'attribut « selected » qui permet de définir quel élément de la liste est affiché par défaut.

Voici à quoi pourrait ressembler la procédure d'ajout des marques dans la liste déroulante.

```
//On compare les IDs des marques du produit avec ceux de la liste, on sélectionne
ceux du produit
foreach ($brands as $brand) {
    if ($brand->id == $productBrand) {
        echo '<option selected value="' . $brand->id . '">' . $brand->name .
'</option>';
    } else {
        echo '<option value="' . $brand->id . '">' . $brand->name . '</option>';
    }
}
```

4.7.2.4. Pré remplissage des mots-clefs

Le pré remplissage de la liste des mots-clefs du produit est similaire à celui des marques mais plus complexe. En effet, un produit peut posséder plusieurs mots-clefs en comparaison à la marque.

On procède donc de la même manière, mais en parcourant la totalité des mots-clefs du produit. On compare les IDs des keywords du produit avec ceux de la liste, puis on sélectionne ceux du produit. A cause du formatage des tableaux d'objets, on doit utiliser la fonction « array_column » pour récupérer uniquement les identifiants des mots-clefs du produit.

Puis on peut parcourir, comme précédemment, chaque mots-clefs et ajouter l'attribut « selected » à chaque mots-clefs égaux aux mots-clefs du produits modifié.

Voici la partie du code s'occupant de la pré sélection des mots-clefs :

```
$idProductKeywords = array_column($productKeywords, "idKeyword");
foreach ($keywords as $keyword) {
    if (in_array($keyword->id, $idProductKeywords)) {
        echo //Le mot-clef sélectionné
    }
    else {
        echo //Un mot-clef standard
    }
}
```

4.7.2.5. Gestion des medias du produit :

Pour créer le menu de gestion des medias du produit, on commence tout d'abord par récupérer les médias du produit (voir « Pré remplissage des champs »). Puis on va vérifier si on est bien en mode d'édition du produit.

On va par la suite vérifier si chaque medias est une image ou non. Ce test est utilisé pour savoir si on doit afficher la miniature de l'image ou une image représentant un document autre. Récupérer le type du media sert aussi à former le nom affiché du media. Pour former ce dernier, on substitue le chemin d'accès complet du media au chemin d'accès du dossier où il est stocké, grâce à cela, on récupère uniquement le nom du média.

Après avoir fait cela, on va afficher le media, son image et un bouton « Supprimer » associé à ce dernier.

```
if ($editMode == 1) {
    echo "<label>Gestion des medias :</label>";
    //Pour chaque medias, on vérifie si le il est une image, et on l'ajoute
accordement
    foreach ($productMedias as $productMedia) {
        $idMedia = $productMedia->idMedia;
        if ($productMedia->isImage) {
            $mediaName = substr($productMedia->mediaSource, strlen(IMG_FOLDER));
            echo //Media est une image, On ajoute le contenu
        }
         else {
            $mediaName = substr($productMedia->mediaSource, strlen(OTHER_FOLDER));
            echo //Media est un "autre", On ajoute le contenu
        }
    }
}
```

Pour supprimer le média, le bouton appelle une page « delete.php », passe en « GET » l'identifiant du média et celui du produit. La page appelée va récupérer ces données et les utiliser pour récupérer les informations complètes du média.

Elle va par la suite utiliser la fonction « unlink » pour supprimer le media du serveur avant de la supprimer de la base de données à l'aide de la fonction « deleteProductMediaById » (ne pas confondre avec « deleteProductMediasById » qui supprime la totalité des medias d'un produit). Cette fonction s'assure aussi de supprimer les champs représentant la relation entre le produit et le media pour éviter de laisser des champs orphelins. Finalement, on renvoie l'utilisateur sur la page ou il se trouvait précédemment. Voici à quoi ressemble la page de suppression du produit :

```
//On détruit le media envoyé en paramètre
$idMedia = filter_input(INPUT_GET, 'idMedia');
$idProduct = filter_input(INPUT_GET, 'idProduct');

//On recupère le "path" du media et on le supprime du server
$media = getMediaById($idMedia);
unlink($media->src);

deleteProductMediaById($idProduct, $idMedia);
header('Location: addProduct.php?edit=1&id='. $idProduct);
```

4.7.2.6. Modification du produit dans la base de données :

Après avoir récupéré la totalité des informations nécessaire à la modification du produit, on contrôle toutes les informations envoyées. Puis on regarde si on est en train d'éditer le produit (variable « editMode = 1 »), si c'est le cas on va appeler la fonction « updateProduct », sinon on utilise la fonction « addProduct » (voir « Ajout produit »). Voici comment fonctionne « updateProduct » :

```
function updateProduct($id, $title, $shortDesc, $longDesc, $isFrontpage,
$startDate, $endDate, $viewCount, $idBrand) {
    global $tableProducts;
    $dbc = connection();
    $dbc->quote($tableProducts);
    $req = "UPDATE $tableProducts SET title=:title, short_desc=:shortDesc, "
            . "long_desc=:longDesc, is_frontpage=:isFrontpage, "
            . "availability_date=:startDate, expiration_date=:endDate, "
            . "view_count=:viewCount, id_brand=:idBrand WHERE id=$id";

    $requPrep = $dbc->prepare($req); // on prépare notre requête
    $requPrep->bindParam(':title', $title, PDO::PARAM_STR);
    $requPrep->bindParam(':shortDesc', $shortDesc, PDO::PARAM_STR);
    $requPrep->bindParam(':longDesc', $longDesc, PDO::PARAM_STR);
    $requPrep->bindParam(':isFrontpage', $isFrontpage, PDO::PARAM_BOOL);
    $requPrep->bindParam(':startDate', $startDate, PDO::PARAM_STR);
    $requPrep->bindParam(':endDate', $endDate, PDO::PARAM_STR);
    $requPrep->bindParam(':viewCount', $viewCount, PDO::PARAM_INT);
    $requPrep->bindParam(':idBrand', $idBrand, PDO::PARAM_INT);
    $requPrep->execute();
    $requPrep->closeCursor();
    return $id;
}
```

Comme pour l'ajout du produit : on récupère en paramètre les informations nécessaire à la modification du produit. Puis grâce à la fonction « bindParam » de PDO, on insère chaque paramètre dans la requête. Finalement, on prépare et on exécute la requête, et on renvoie à l'utilisateur l'id du produit modifié pour lui confirmer que la procédure s'est passée comme prévue.

4.7.3. Suppression d'un produit.

Pour accéder à la suppression d'un produit, il faut se rendre sur le détail produit, puis être connecté en mode administrateur. Si un utilisateur n'est pas connecté en administrateur, le menu d'administration du produit ne s'affichera pas.

Après s'être connecté et s'être rendu sur le détail d'un produit, un bouton « Supprimer le produit » apparait. En cliquant sur le bouton, une fenêtre modale s'affiche et demande la confirmation de l'administrateur pour la suppression d'un produit.

Quand on clique sur le bouton, on récupère l'identifiant du produit, et on utilise la fonction « deleteProductById ». Pour supprimer le produit définitivement, on commence par supprimer les medias lié au produit. Pour cela on les supprime tout d'abord du serveur « apache » à l'aide de la commande « unlink ». Puis on supprime les médias lié au produit dans la base de données. Pour éviter de laisser des champs orphelins dans la base, on supprime aussi manuellement les relations du produit (produit & mot-clef, produit & medias). Finalement, on supprime le champ du produit dans la base à l'aide de la fonction « deleteFieldById » de « basics_bdd ».

Voici le code de la fonction « deleteProductById » :

```php
function deleteProductById($id) {
    global $tableProducts;
    //On supprime les medias lié au produit
    deleteProductMediasById($id);
    //On supprime la relation avec les medias
    deleteProductMediaRelation($id);
    //On supprime la relation avec les keywords
    deleteProductKeywordRelation($id);
    //On supprime le produit
    deleteFieldById($id, $tableProducts);
}
```

Voici la fonction supprimant les medias du produit :

On commence par parcourir les medias du produit, et de les supprimer du serveur. Puis on crée une requête qui supprime les médias du produit. Cette requête est peu conventionnelle car on utilise la fonction « SELECT » pour supprimer seulement les medias associés au produit.

```php
function deleteProductMediasById($id) {
    $medias = getProductMediasById($id);
    foreach ($medias as $media) {
        unlink($media->mediaSource);
    }
    $dbc = connection();
    $table = 'medias';
    $dbc->quote($table);
    $req = "DELETE FROM $table "
            . "WHERE id IN ("
            . "SELECT pm.id_medias "
            . "FROM products_has_medias "
            . "AS pm "
            . "WHERE pm.id_products = $id)";
    $requPrep = $dbc->prepare($req); // on prépare notre requête
    $requPrep->execute();
    $data = $requPrep->fetchAll(PDO::FETCH_OBJ);
    $requPrep->closeCursor();
}
```

4.8. Fonctionnalités supplémentaires

4.8.1. Site web mobile

Le site web est compatible avec différents périphériques mobiles tels que des tablettes ou des smartphones. Le Framework « bootstrap » et des modifications des feuilles de style « CSS » on permit d'obtenir un site web adaptable à différents formats. La technologie de grille de « bootstrap » a été utilisée pour l'affichage des produits recommandés et des produits les plus vus. Voici un exemple de code HTML utilisant le système de grilles de bootstrap :

```
<!--
Le container prend différentes tailles en dépendant du format
Mobile      : 12 Emplacements (1 Elément par ligne)
Tablette    : 6 Emplacements (2 Eléments par ligne)
Laptop      : 4 Emplacements (3 Eléments par ligne)
Desktop     : 3 Emplacements (4 Eléments par ligne)
 -->
<div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
    <div class="panel panel-success">
        <!-- CONTENU ICI -->
    </div>
</div>
```

Comme avec le système de grille, certains éléments du site web ont besoin de changer de propriétés CSS en dépendant de la taille de l'écran. En utilisant le sélecteur « @media » on peut changer les styles en fonction des périphériques de consultation. Dans notre cas, on change les propriétés suivantes quand on est au format mobile (767px de largeur) :

- On change la taille de la barre de recherche pour qu'elle prenne l'entièreté de la longueur de l'écran
- On change le menu des mots-clefs pour qu'il ne s'affiche pas au-dessus des produits.
- On réduit la taille des titres (h1)

Voici à quoi ressemble la partie de la feuille de style CSS permettant de gérer le format mobile :

```
@media (max-width: 767px) {
    .list-group{
        position: relative;
    }

    .mini-submenu{
        position: relative;
    }
    .search-bar{
        width: 100%;
    }

    h1{
        font-size: 25px;
    }
};
```

5. Tests

5.1. Affichage des produits

Test	Résultat attendu	Résultat obtenu
Si on est sur l'accueil	Les produits recommandés sont affichés	OK
Si on est sur l'accueil	Les produits recommandés sont triés par ordre alphabétique	OK
Si on est sur l'accueil	Les produits les plus vus sont affichés	OK
Si on est sur l'accueil	Les produits les plus vus sont triés par nombre de vues	OK
Si on est sur l'accueil	Les informations des produits affichés correspondent à celles de la base	OK
Si on est sur l'accueil	Les miniatures des produits sont affichées	OK
Si on est sur l'accueil	Un lien est présent sur chaque produit affiché pour accéder à l'affichage détaillé	OK

5.2. Menu déroulant

Test	Résultat attendu	Résultat obtenu
Si on est sur l'accueil	Le menu déroulant est présent sur le côté de la page	OK
Si on est sur le détail produit	Le menu déroulant est présent sur le côté de la page	OK
Si on est sur le résultat de recherche	Le menu déroulant est présent sur le côté de la page	OK
Si on clique sur le bouton ouvrir	Le menu déroulant se déroule	OK
Si on clique sur le bouton fermer	Le menu déroulant se referme	OK
Quand le menu est ouvert	Les informations affichées proviennent de la base de données	OK
Quand le menu est ouvert	Les informations affichées sont triées par ordre alphabétique	OK
Quand le menu est ouvert	Les informations affichées sont affichées en majuscule	OK
Quand on clique sur un élément de la liste	On renvoie la personne vers la page de recherche associée au mot-clef	OK

5.3. Affichage des produits recherchés

Test	Résultat attendu	Résultat obtenu
Si on a fait une recherche par mot-clef	Les produits correspondant au mot-clef sont affichés	OK
Si on a fait une recherche multicritères	Les produits correspondant aux critères entrés sont affichés	OK
Quand on affiche les résultats	Les miniatures des produits sont affichées	OK
Quand on affiche les résultats	Les informations des produits correspondent à celles présentes dans la base	OK
Quand on affiche les résultats	Un lien vers le détail du produit est présent	OK

5.4. Affichage du détail du produit

Test	Résultat attendu	Résultat obtenu
Quand on clique sur un produit depuis l'accueil	On est redirigé vers la page du détail du produit	OK
Quand le produit n'existe pas dans la base de données	On renvoie l'utilisateur sur la page d'accueil	OK
Quand l'utilisateur n'est pas un administrateur	On n'affiche pas le menu d'administration du produit	OK
Quand l'utilisateur est un administrateur	On affiche un bouton pour modifier et pour supprimer le produit	OK
Quand on affiche le détail produit	Les images des produits sont affichées dans un "carousel"	OK
Quand on affiche le détail produit	Les fichiers téléchargeables s'affichent en bas de page	OK
Quand on affiche le détail produit	Les informations du produit sont correctes à celle présentes dans la base	OK
Si le produit possède une seule image	Les contrôles du "carousel" ne s'affichent pas	OK

5.5. Connexion utilisateur

Test	Résultat attendu	Résultat obtenu
Quand on clique sur le bouton "Connexion"	On ouvre la fenêtre modale de connexion	OK
Quand on laisse des champs vides	On informe l'utilisateur qu'il faut remplir tous les champs	OK
Quand on envoie des informations incorrectes	On affiche un message d'erreur à l'utilisateur	OK
Quand on envoie des informations correctes	On connecte l'utilisateur	OK
Quand on clique sur le bouton "Fermer"	On ferme la fenêtre modale	OK
Quand on clique sur le bouton "x"	On ferme la fenêtre modale	OK
Quand on clique en dehors de la fenêtre modale	On ferme la fenêtre modale	OK

5.6. Déconnexion utilisateur

Test	Résultat attendu	Résultat obtenu
Quand on clique sur le bouton "Se déconnecter"	On déconnecte l'utilisateur	OK

5.7. Ajout d'un produit

Test	Résultat attendu	Résultat obtenu
Quand on n'est pas administrateur	On redirige l'utilisateur vers la page d'accueil	OK
Quand on laisse des champs vides	On informe l'utilisateur que certains champs sont encore vides	OK
Quand l'utilisateur entre des informations trop longues	On empêche l'envoi du formulaire, et on affiche un message d'erreur	OK
Quand l'utilisateur envoie un fichier trop gros	On empêche l'envoi du formulaire, et on affiche un message d'erreur	OK
Quand l'utilisateur envoie un formulaire valide	On ajoute le produit et ses medias dans la base	OK

5.8. Modification d'un produit

Test	Résultat attendu	Résultat obtenu
Quand on n'est pas administrateur	On redirige l'utilisateur vers la page d'accueil	OK
Quand on laisse des champs vides	On informe l'utilisateur que certains champs sont encore vides	OK
Quand l'utilisateur entre des informations trop longues	On empêche l'envoi du formulaire, et on affiche un message d'erreur	OK
Quand l'utilisateur envoie un fichier trop gros	On empêche l'envoi du formulaire, et on affiche un message d'erreur	OK
Quand on arrive sur la page	On pré remplis les champs du formulaire par rapport au produit modifié	OK
Quand on arrive sur la page	On affiche les medias du produit et des boutons pour les supprimer individuellement	OK
Quand l'utilisateur envoie un formulaire valide	On modifie le produit et ses medias dans la base	OK

5.9. Suppression du media d'un produit

Test	Résultat attendu	Résultat obtenu
Quand on clique sur le bouton supprimer	On redirige l'utilisateur sur une page de suppression	OK
Quand on clique sur le bouton supprimer	Le media est supprimé du serveur	OK
Quand on clique sur le bouton supprimer	Le media est supprimé de la base de données	OK
Quand on clique sur le bouton supprimer	La relation entre le media et le produit est supprimée dans la base de données	OK
Quand on clique sur le bouton supprimer	On redirige l'utilisateur vers la page initiale après la suppression du media	OK

5.10. Suppression d'un produit

Test	Résultat attendu	Résultat obtenu
Quand on clique sur le bouton supprimer	On affiche une fenêtre modale demandant de confirmer la suppression du produit	OK
Quand on clique sur en dehors de la fenêtre modale	On ferme la fenêtre modale	OK
Quand on clique sur la croix de la fenêtre modale	On ferme la fenêtre modale	OK
Quand on clique sur le bouton "fermer de la fenêtre modale	On ferme la fenêtre modale	OK
Quand on clique sur le bouton supprimer (modale)	On supprime les medias du produit sur le serveur	OK
Quand on clique sur le bouton supprimer (modale)	On supprime les medias du produit dans la base de données	OK
Quand on clique sur le bouton supprimer (modale)	On supprime les relations entre le produit et les medias dans la base de données	OK
Quand on clique sur le bouton supprimer (modale)	On supprime les relations entre le produit et les mots-clefs dans la base de données	OK
Quand on clique sur le bouton supprimer (modale)	On supprime le produit de la base de données	OK
Après la suppression	On redirige l'utilisateur vers l'accueil et on lui affiche un message de confirmation	OK

5.11. Mise à jour de l'en-tête dynamique

Test	Résultat attendu	Résultat obtenu
Quand on est pas connecté	On affiche l'en-tête de visiteur	OK
Quand on est connecté en tant qu'utilisateur	On affiche l'en-tête d'utilisateur	OK
Quand on est connecté en tant qu'administrateur	On affiche l'en-tête d'administrateur	OK

6. Conclusion

6.1. Bilan

Catal'info possède toutes les fonctions nécessaire à la création d'un catalogue de produit en ligne. Il offre la possibilité d'afficher les produits recommandés par les administrateurs et d'afficher les produits les plus vus. Il donne aussi accès a deux types de recherches, une recherche par catégorie ou « mot-clef » et un outil de recherche multicritères.

Catal'info permet aussi à un administrateur de gérer les produits du site depuis différentes interfaces. Les administrateurs peuvent se connecter au site web à l'aide d'un formulaire de connexion sécurisé. Un administrateur peut donc ajouter, modifier et supprimer des produits. Chaque produit possède des medias tels que des images ou des documents « pdf », ces derniers peuvent être gérés (Ajout et suppression) par un administrateur depuis la page de modification du produit.

Catal'info possède aussi une interface graphique complètement adaptable a différents formats, y compris le format mobile.

Cependant, le site web, n'est pas parfait et des améliorations sont envisageables.

6.2. Améliorations envisageables

6.2.1. Pagination sur la page de recherche des produits :

La page de recherche actuelle ne possède pas une limite de produits affichés par page, si une recherche rapporterais deux-mille résultat, la récupèrerais et afficherais les deux-mille produits sur la même page. Cela pose des problèmes au niveau du serveur et au niveau de l'utilisateur et cela pourrait être évité à l'aide de la pagination.

La pagination permettrait d'afficher un certain nombre de produit par page et de séparer les résultats en plusieurs pages au lieu de tout afficher sur une seule page.

6.2.2. Ajout de medias au produit sans rafraichir la page (AJAX) :

Après avoir ajouté un media au produit, on recharge la page. Il aurait été possible d'ajouter un formulaire préliminaire qui permettrait de d'ajouter certains medias et de les gérer avant de les envoyer au serveur.

6.2.3. Utilisation de « dropzone.js » pour ajouter des medias :

Dropzone aurait été une addition cosmétique à l'ajout des medias. Combiné avec AJAX, il aurait permis aux utilisateurs de « drag and drop » des medias à l'intérieur d'un container pour les envoyer au serveur. Cette option est plus intuitive et plus plaisante pour l'utilisateur.

6.2.4. Suppression des medias sans rafraichir la page (AJAX) :

Pour supprimer un média depuis le formulaire de modification de produits, on appelle actuellement une autre page qui s'occupe de la suppression du média. Cela engendre un rafraichissement de page qui pourrait être évité à l'aide d'AJAX. Les rafraichissements de page retirent les informations entrées au préalable par les utilisateurs.

6.2.5. Système de favoris :

Une table « user_has_products » est présente dans la base de données du site mais n'est actuellement pas utilisée. Il aurait été possible d'utiliser cette table pour permettre à des utilisateurs connectés d'ajouter des produits à leurs favoris. Cela permet aux utilisateurs de retrouver rapidement des produits sans passer par la fonction de recherche du site web.

6.2.6. Tri par catégories :

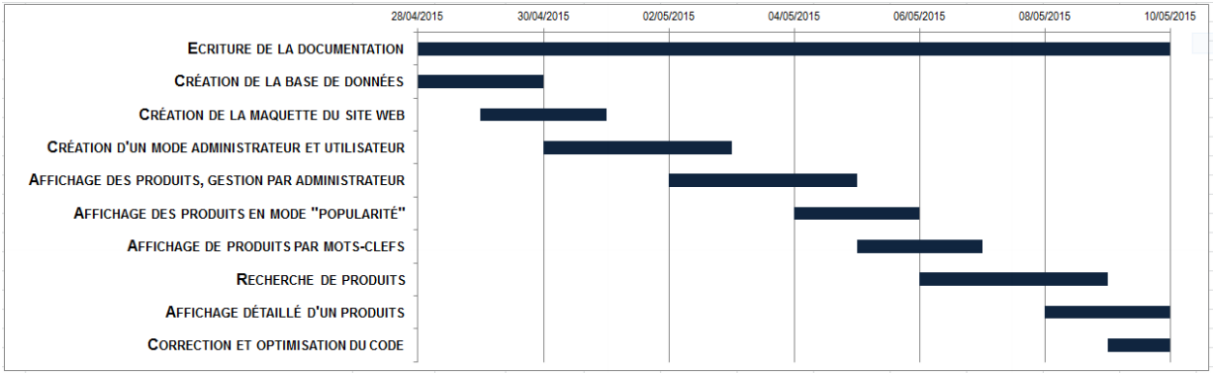
Une alternative à la pagination des produits après une recherche est le tri par catégorie. Si le nombre de produits affichés sur la page dépasse un seuil, on afficherait des catégories avec le nombre de produits disponible à l'intérieur de ces dernières. Cliquer sur une catégorie redirigerais l'utilisateur vers une page contenant la nouvelle liste de produits triés par critères et mot-clef combinés.

Exemple : Si les produits recherchés dépassent les 20, on affiche, « Alimentation (24), Cartes-mères (43), RAM (12), etc…

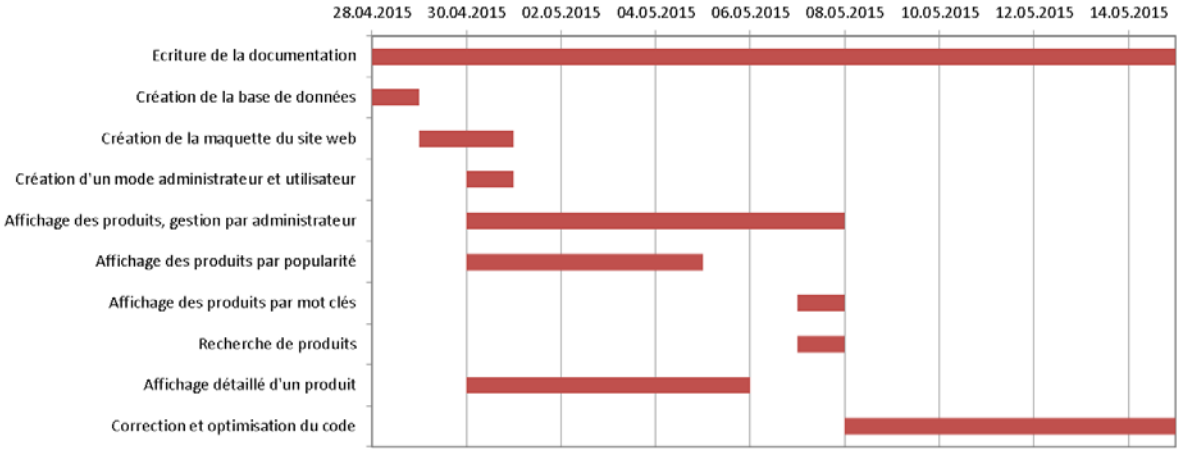
6.3. Comparaison journal et planning

Au fil du projet, j'ai rempli un planning représentant le déroulement actuel du projet. Voici une comparaison entre le planning « réel » et le planning des actions prévues :

6.3.1. Planning prévu :



6.3.2. Planning réel :



6.3.3. Conclusion :

En conclusion, j'ai trouvé très dur de respecter strictement un planning et malgré avoir inversé l'ordre de certaines actions durant le développement, le projet est arrivé à un état plus que satisfaisant et le cahier des charges a entièrement été complété.

7. Bibliographie

- Framework Bootstrap : http://getbootstrap.com/
- Serveur WAMP #1 : http://www.easyphp.org/
- Serveur WAMP #2 : www.wampserver.com
    - Ce serveur a été utilisé pour bénéficier de la fonction « array_column » qui n'est uniquement disponible à partir de PHP 5.5
    - EasyPHP VC11 étant trop instable, j'ai utilisé WAMPSERVER pour régler le problème.
- NetBeans IDE : https://netbeans.org/downloads/
- Schémas gliffy : https://www.gliffy.com
- Icones divers : http://iconmonstr.com/
- Aide au PHP : http://php.net/
- Aide au développement divers : http://www.w3schools.com/
- « basics_bdd » Fichier des fonctions de base pour gérer une base de données développée en classe en 4ème année.