|  |
| --- |
| CFPT |
| C:\Users\nunesr_info\Desktop\Gestock_TPI\Web\img\logo.png |
| Site de vente de matériel informatique |
|  |
| **Ricardo Nunes de Oliveira** |
| **19/06/2017** |

|  |
| --- |
|  |

Table des matières

[2 Introduction 3](#_Toc485630023)

[3 Analyse préliminaire 3](#_Toc485630024)

[3.1 Organisation 3](#_Toc485630025)

[3.1.1 Inventaire de technologies 3](#_Toc485630026)

[3.1.2 Inventaire d’applications 3](#_Toc485630027)

[3.2 Analyse concurrentielle 4](#_Toc485630028)

[3.3 Cahier des charges détaillé 6](#_Toc485630029)

[3.3.1 Objectifs 6](#_Toc485630030)

[3.3.1.1 Fonctionnels 6](#_Toc485630031)

[3.3.1.2 Académiques 7](#_Toc485630032)

[3.3.2 Définition de l’audience 7](#_Toc485630033)

[3.4 Détermination des points faibles 7](#_Toc485630034)

[3.5 Plan graphique 7](#_Toc485630035)

[4 Conception 8](#_Toc485630036)

[4.1 Plan fonctionnel du site 8](#_Toc485630037)

[4.2 Maquettes préliminaires 9](#_Toc485630038)

[4.3 Base de données 12](#_Toc485630039)

[5 Réalisation 14](#_Toc485630040)

[5.1 Arborescence du site 14](#_Toc485630041)

[5.1.1 Dossier « css » 15](#_Toc485630042)

[5.1.2 Dossier « fonts » 15](#_Toc485630043)

[5.1.3 Dossier « img » 15](#_Toc485630044)

[5.1.4 Dossier « inc » 15](#_Toc485630045)

[5.1.5 Dossier racine 15](#_Toc485630046)

[5.1.6 Dossier « js » 15](#_Toc485630047)

[5.2 Conception du code 15](#_Toc485630048)

[5.2.1 Fichier « gestock.js » 15](#_Toc485630049)

[5.2.2 Classe « Gestock.php » 16](#_Toc485630050)

[5.2.2.1 Fonctions complexes 17](#_Toc485630051)

[5.2.2.1.1 insertProductIntoCart() 17](#_Toc485630052)

[5.2.2.1.2 passOrder() 18](#_Toc485630053)

[5.2.3 Classe « DataToHTML.php » 18](#_Toc485630054)

[5.2.4 Scripts 19](#_Toc485630055)

[5.2.4.1 « addProduct.php » 19](#_Toc485630056)

[5.2.4.2 « addToCart.php » 19](#_Toc485630057)

[5.2.4.3 « addUser.php » 20](#_Toc485630058)

[5.2.4.4 « authentification.php » 20](#_Toc485630059)

[5.2.4.5 « deleteProduct.php » 21](#_Toc485630060)

[5.2.4.6 « deleteProductFromCart.php » 21](#_Toc485630061)

[5.2.4.7 « deleteUser.php » 21](#_Toc485630062)

[5.2.4.8 « modifyProduct.php » 21](#_Toc485630063)

[5.2.4.9 « modifyUser.php » 21](#_Toc485630064)

[5.2.4.10 « passOrder.php » 22](#_Toc485630065)

[5.3 Modèle de BDD final 22](#_Toc485630066)

[6 Tests 23](#_Toc485630067)

[7 Guide d’installation 28](#_Toc485630068)

[8 Conclusion 29](#_Toc485630069)

[8.1 Objectifs 29](#_Toc485630070)

[8.2 Points 29](#_Toc485630071)

[8.2.1 Forts 29](#_Toc485630072)

[8.2.2 Faibles 29](#_Toc485630073)

[8.3 Difficultés particulières 30](#_Toc485630074)

[8.4 Suite possibles pour le projet 30](#_Toc485630075)

[9 Bibliographie 30](#_Toc485630076)

# Introduction

Gestock est un site web permettant à des utilisateurs identifiés d’acheter du matériel informatique. Ceux-ci ont la complète gestion de leur panier d’achats qu’ils peuvent commander s’ils ont suffisamment d’argent dans leur compte, et si tous les produits sont en stock.

Les administrateurs du site auront accès à la gestion complète du site. Ils vont pouvoir ajouter, modifier ou supprimer un produit dans une page réservée uniquement à eux. Ils vont également pouvoir remplir les porte-monnaie des utilisateurs vu qu’il n’y aura pas de vrai système de paiement dans la première version du site.

J’ai choisi ce sujet car je pense que c’est la base de tout site e-commerce, et c’est à partir de là qu’on peut travailler avec une grande quantité de clients. À l’école, on fait souvent des projets divisés, c’est-à-dire qu’on travaille une chose un jour, quelque chose d’autre un autre jour, mais on assemble rarement toutes les parties pour en faire un tout. Les cas où ça arrive, c’est dans le cadre de travaux de groupe, mais voulant être indépendant plus tard, je voulais voir à quoi ressemble ce genre de travail pour une seule personne, et sans outils permettant d’accélérer le processus.

# Analyse préliminaire

Ce chapitre traitera toute la partie non-technique qui précède la réalisation du projet.

## Organisation

Je suis le seul à travailler dans ce projet.

Concernant la continuité du projet, je prendrai les tâches dans l’ordre logique pour un client. C’est-à-dire que j’essaierai de parcourir le chemin qu’un client fera (probablement) la première fois arrivé sur mon site. Par exemple, il arrivera dans la page index, donc c’est celle-ci que je ferai en premier. Mais elle ne peut pas fonctionner sans la BDD, donc cela sera ma première tâche. Après avoir été dans la page d’accueil, il va peut-être recherche un objet d’une certaine catégorie, ou alors voir les détails d’un produit. Dès qu’il est intéressé par l’achat d’un des produits, il voudra probablement se connecter pour commencer à faire son panier d’achat, donc ça sera la prochaine tâche. Je suivrai la réalisation des tâches dans cet ordre-là, jusqu’à la finalisation du site.

### Inventaire de technologies

* MySQL
* HTML / CSS
* PHP
* JavaScript / JQuery

### Inventaire d’applications

**MySqlWorkbench (6.3):** Utilisé pour créer le modèle de base de données et le mettre en place dans le server. Majoritairement utilisé pour les grands changements dans la BDD.

**EasyPHP (14.1VC11):** Server local Apache, PHP & MySql.

**VisualStudio Code (1.13.0):** IDE Principal utilisé pour le HTML, le CSS et le PHP. J’ai choisi cet IDE car il a une interface épurée avec peu de choses à l’écran, donc je n’ai la vision obstruée par un tas de fonctionnalités qui me sont inutiles. Certes, il n’y a pas de débuguer, mais je me sens plus à l’aise en le faisant moi-même, donc ce n’est pas un problème.

**Netbeans (8.0.2):** Uniquement utilisé pour ses raccourcis pour le PHPDoc.

**MS PowerPoint (14.0.7149.5000):** Création des maquettes pour mon site.

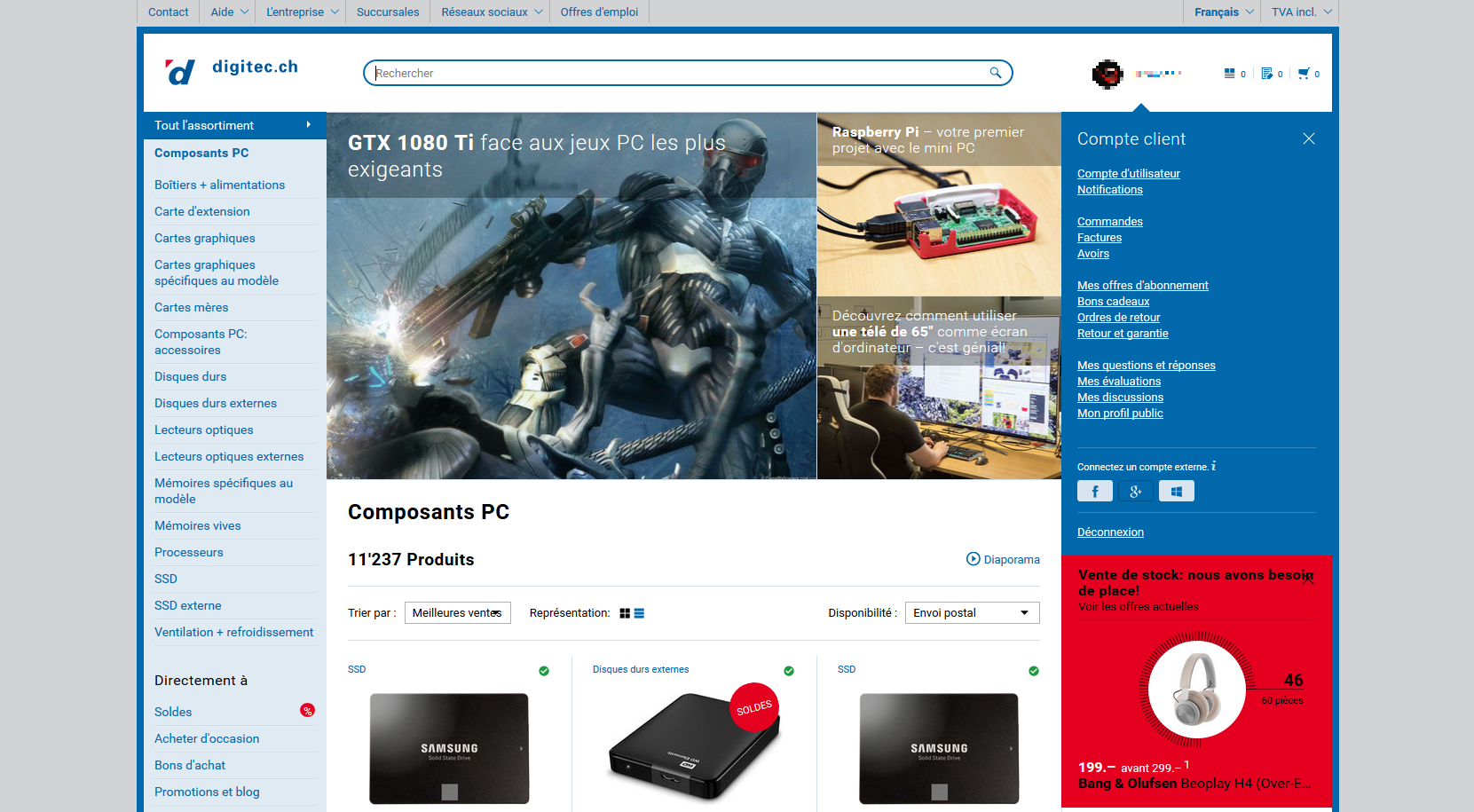
**Google Chrome (58.0.3029.110):** Navigateur principal pour tester mes scripts et mes pages web.

**Github (3.3.4.0):** Logiciel de versionning & de sauvegarde des documents.

## Analyse concurrentielle

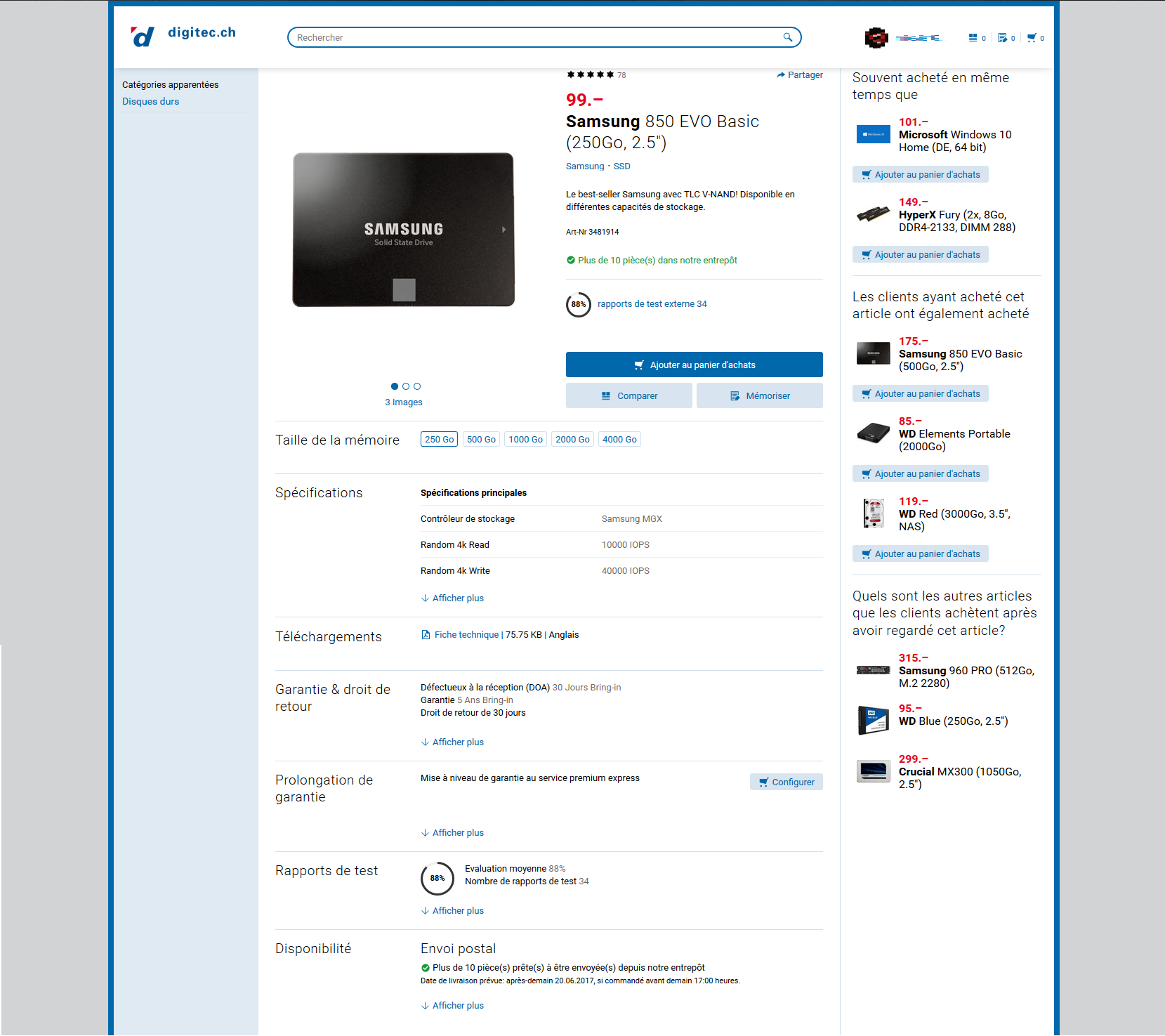
**Points positifs:**

Je vais principalement m’inspirer de « digitec.ch », je trouve qu’ils ont une bonne ergonomie, et l’information est à portée de main.

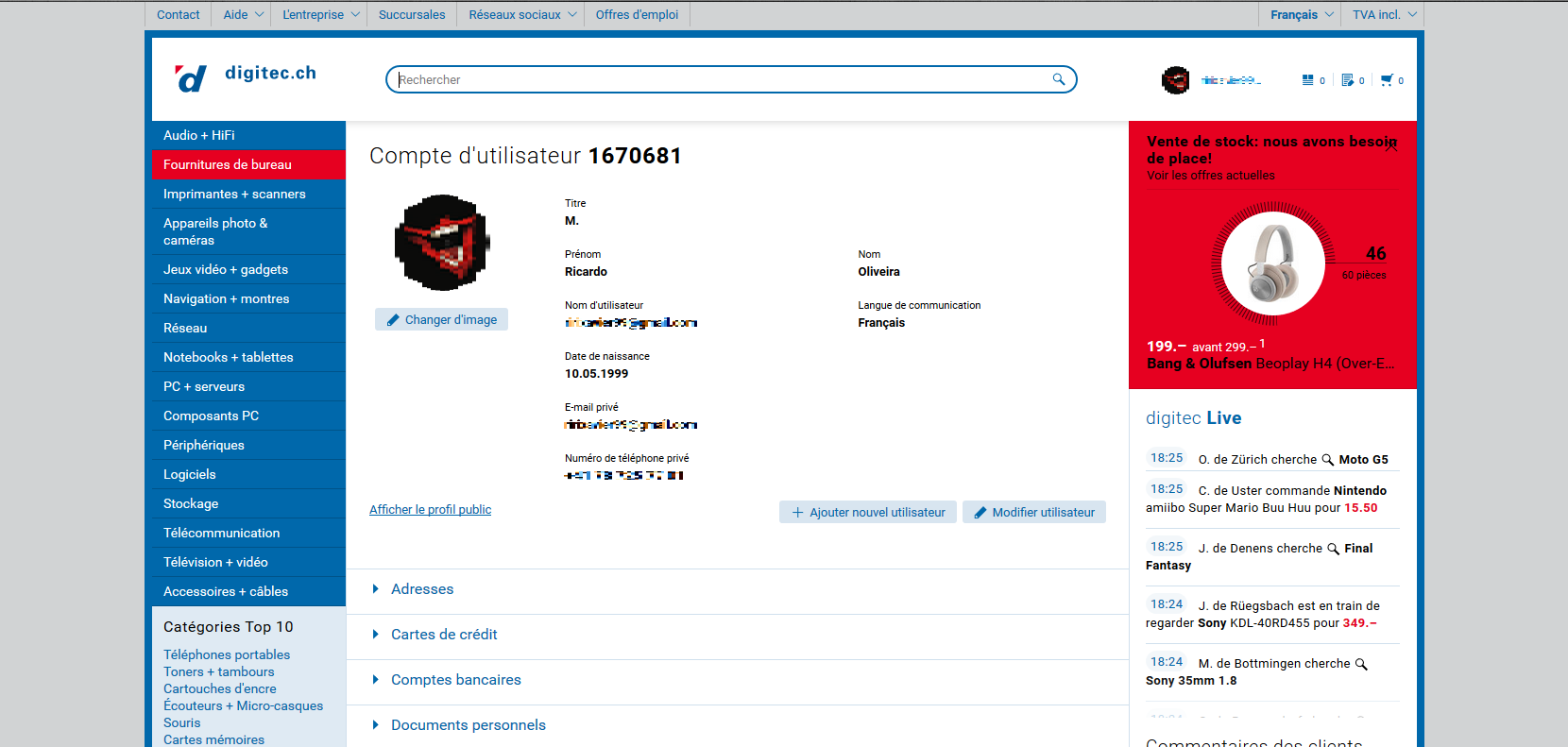


Comme on peut le voir à gauche, on a un listing de toutes les catégories, il est simple, mais c’est ce qui fait qu’on ne s’y perd pas. En bas nous avons des produits (en vedette dans ce cas-là) sur lesquels on peut cliquer pour accéder à leur page d’informations. De plus, on peut ajouter un produit sans aller dans sa page de détails.

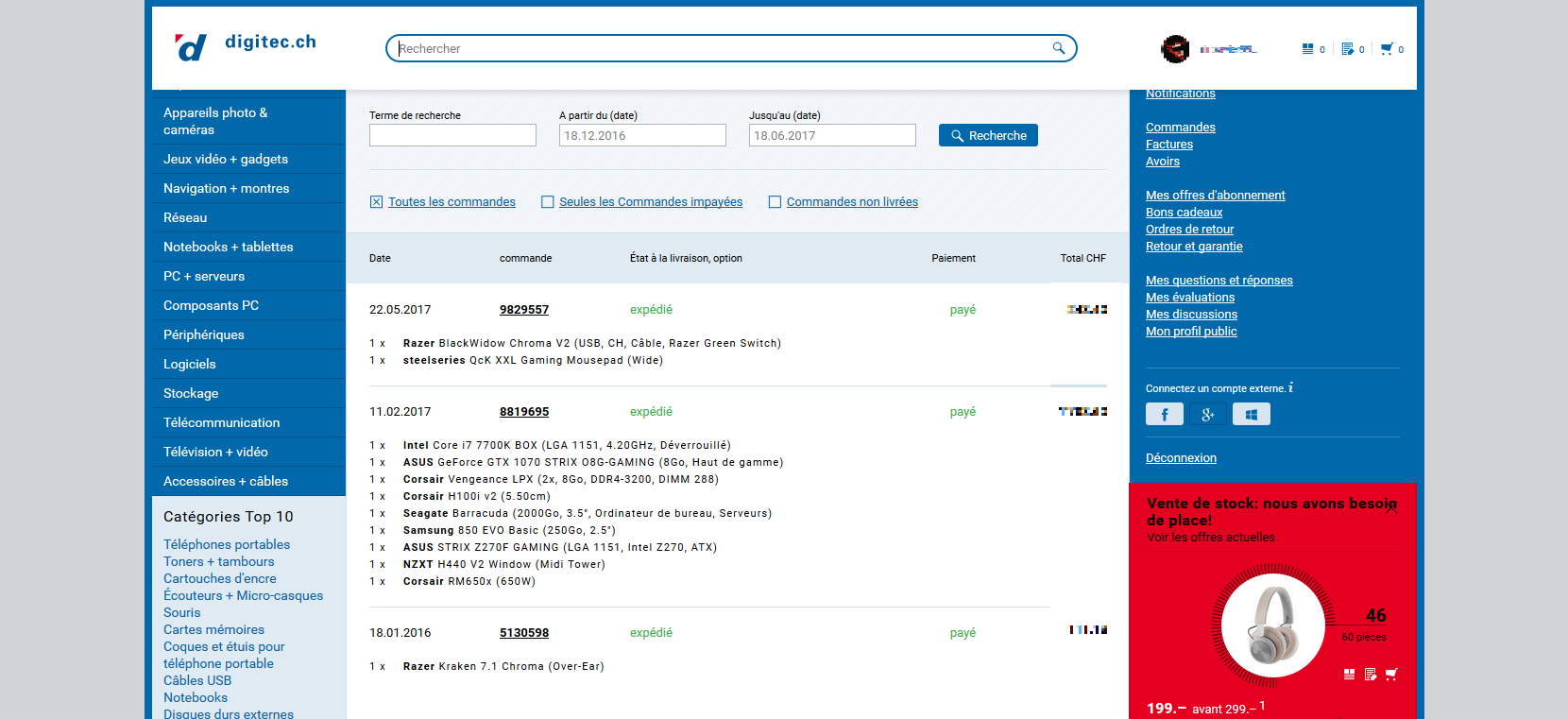
Quand on est connectés, on a, en haut à droite, notre image de compte ainsi que notre email. En cliquant dessus, on a un onglet qui s’ouvre, nous permettant de vite accéder aux informations qui pourraient nous être utiles. Bonne description d’un produit, groupé par sous-catégories.



On a là la vue complète sur un produit. On a ses informations regroupées par catégories (Je n’aurai pas le temps de faire autant de tri sur Gestock, surtout que je n’ai pas autant d’informations sur les produits.) et on peut voir sur si le produit est disponible à l’achat.



La page de l’utilisateur est complète et bien rangée. On a en haut les informations générales sur le compte, et on a, en bas, des informations plus spécifiques.



Les commandes antérieures sont affichées ici. Comme on peut le voir, on a la liste des produits qui on été commandés, ainsi que le coût total.

* Informations personnelles facilement accessibles.
* Bon listing des différentes commandes passées.
* **Points négatifs évités (Prodimex + 1000Ordi):**
* Impossibilité d’ajouter un produit sans aller dans la page de celui-ci. (1000Ordi)
* Site lent et laid. (Prodimex)
* Système de navigation entre catégories parfois confu. (Le deux)

## Cahier des charges détaillé

### Objectifs

#### Fonctionnels

J’ai porté une grande importance à la facilité d’accès à chaque information, afin que le client ait une bonne expérience utilisateur pour qu’il soit plus facilement tenté de réutiliser les services du site.

**Utilisateurs :**

* Trouver un produit recherché.
* Tri des produits par catégorie.
* Inscription/Authentification.
* Gestion du panier.
* Gestion des commandes.
* Interface sobre mais fonctionnelle.

**Administrateurs :**

* Gestion des produits. (CRUD)
* Gestion de l’emplacement des produits dans le stock.
* Gestion des utilisateurs. (RU)

#### Académiques

* Évaluer la charge de travail pour une seule personne.
* Me tester.
* Mettre en pratique tout ce qui a été appris pendant ma formation.

### Définition de l’audience

Mon projet s’adresse à toutes personnes recherchant à faire l’acquisition d’un, ou de plusieurs composants d’ordinateur

## Détermination des points faibles

Je me sens plutôt à l’aise avec les langages et les logiciels choisis, mais il y a tout de même certains points que je vais devoir approfondir afin de réaliser mon travail sans problèmes.

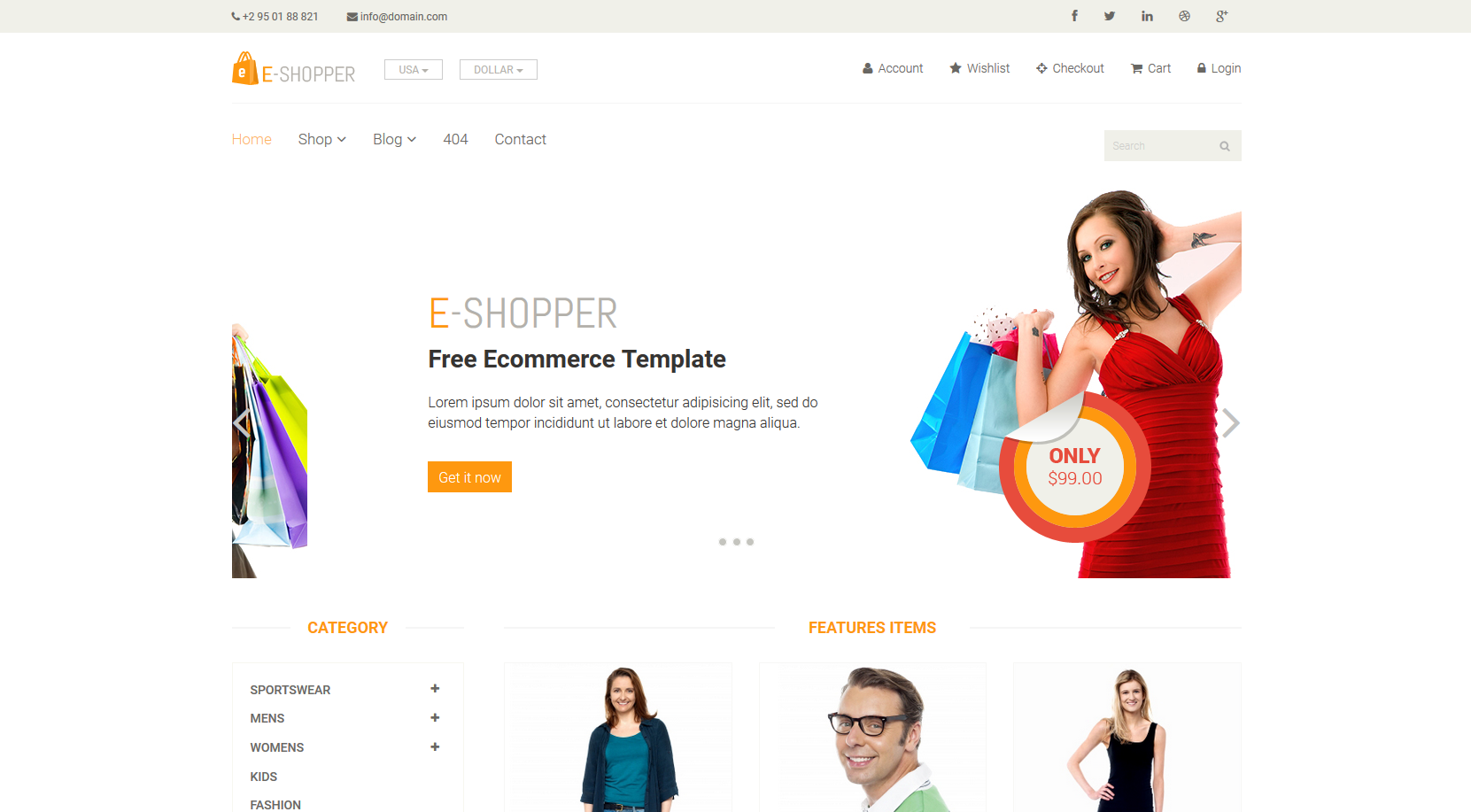
Premièrement il y a la pagination. Bien que je ne pense pas que cela soit compliqué à réaliser, il faut que je m’informe un peu plus à ce sujet. Ma piste serait d’utiliser « LIMIT » & « OFFSET » dans les requêtes SQL, donc je regarderai cela sur « sql.sh ».

Je pense que la requête concernant le panier d’achat peut être compliquée vu qu’un produit peut être dans plusieurs stocks en même temps, donc c’

Ensuite, il y a l’ajax. J’en ai déjà fait, mais ce n’est pas encore quelque chose avec lequel je suis complètement à l’aise dessus, mais ça devrait tout de même bien se passer. J’utiliserai l’ajax pour ajouter un produit dans le panier, et, si j’ai le temps, pour afficher les produits d’une certaine catégorie, au lieu de les afficher dans une page à part.

## Plan graphique

Je vais utiliser un template bootstrap que j’ai trouvé sur [ce lien](http://usebootstrap.com/theme/eshopper). Le voici :



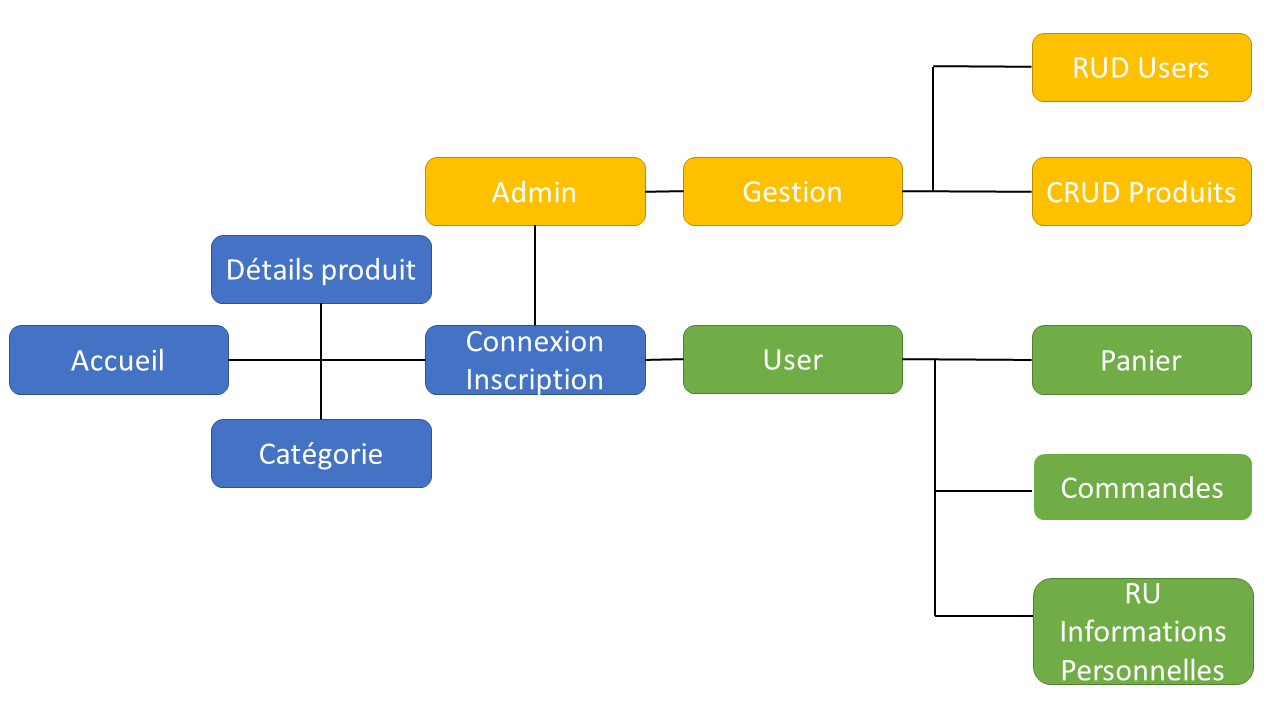
Je trouve le menu assez sobre et facile, c’est pour cela que je l’ai choisi. Il y a également certaines pages qui ont certains contrôles déjà positionnés et stylisés, comme la page de login ou encore l’affichage des produits. (Que vous pouvez voir sur cette page sous « Features Items »)

Cependant, beaucoup de choses ne me plaisent pas, comme par exemple la couleur et le fait qu’il y ait beaucoup de choses dont je n’ai pas besoin. Le changement de monnaie, le footer énorme, le slider d’images, ainsi que la majorité des onglets du menu en sont des exemples. Je vais donc le réadapter le template pour qu’il comble mes besoins, et je montrerai le résultat dans la rubrique « Style final ».

# Conception

Ce chapitre traitera toute la partie technique qui précède la réalisation du projet.

## Plan fonctionnel du site



Ce que vous voyez en bleu sont les fonctionnalités qui sont accessibles sans se connecter. On peut se balader sur le site, chercher les produits qui nous intéressent et se connecter/inscrire.

En vert c’est quand on est connecté en tant qu’utilisateur lambda. Celui-ci peut gérer son panier, passer une commande et commander. Il peut également aller dans sa page de détails afin de voir et modifier ses informations personnelles.

La partie d’administration est en jaune. Quand on est connectés en tant qu’administrateur, on a accès à la gestion complète des produits, c’est-à-dire qu’on peut modifier un produit, le supprimer, ou alors en ajouter un nouveau. On peut également mettre à jour les informations d’un utilisateur (notamment son porte-monnaie virtuel) ainsi que le bannir.

## Maquettes préliminaires

La barre de navigation sera présente sur toutes les pages et permettra d’accéder à la page d’accueil, le page de l’utilisateur, son panier, ainsi qu’à la page de connexion.

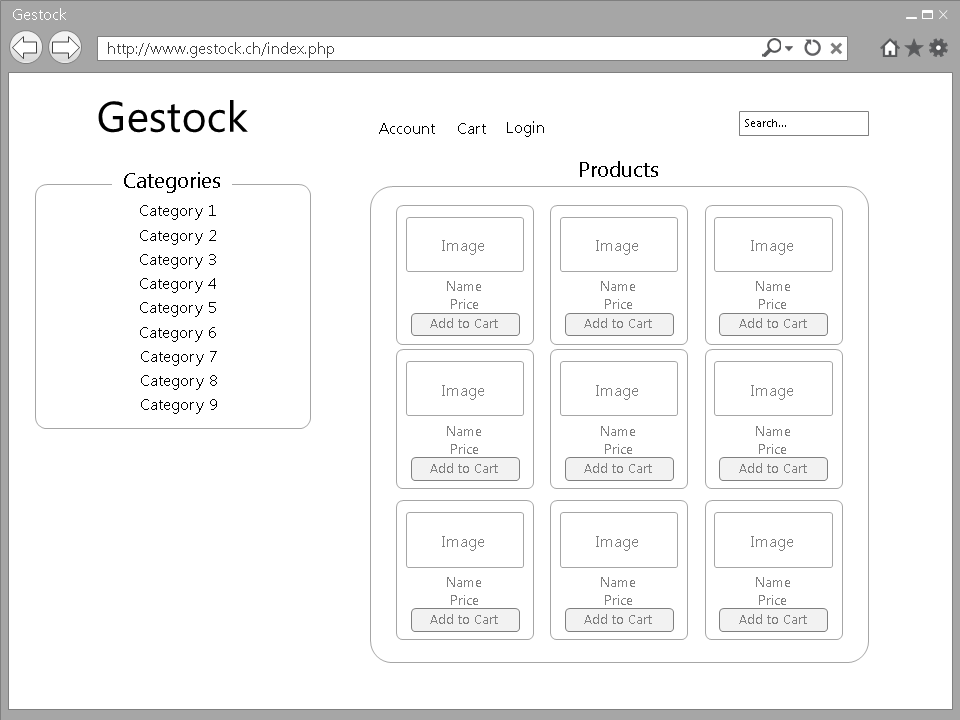


Figure 1

Dans cette image, nous avons la page principale du site, la page d’accueil. Ici nous aurons un aperçu des catégories à gauche, ainsi qu’un aperçu de 9 des produits présentés par le site, à droite. Si l’utilisateur est sûr qu’il veut commander un de ces produits, il peut directement l’ajouter au panier sans avoir à aller dans la page de détails de celui-ci. En cliquant sur un produit, nous arrivons à la page d’informations de celui-ci. La page de catégorie a le même aspect, mais ne montre uniquement les produits d’une certaine catégorie.

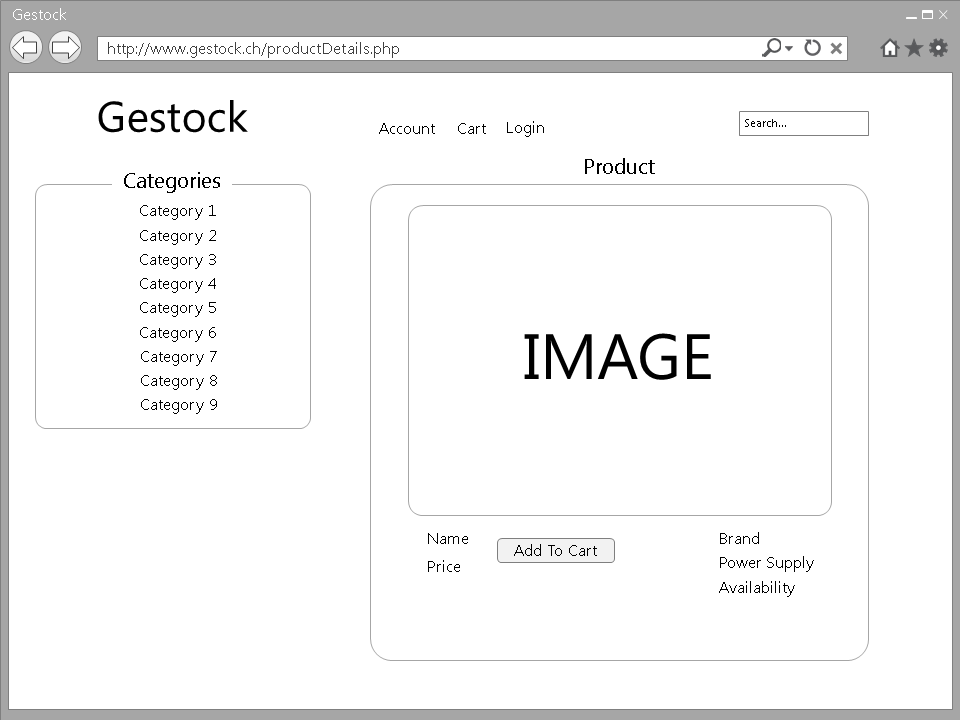


Figure 2

Nous avons ici la page des détails d’un certain produit. Celle-ci affiche le nom du produit, son prix, sa marque, sa catégorie ainsi que sa disponibilité. On retrouve également la liste des catégories à gauche.

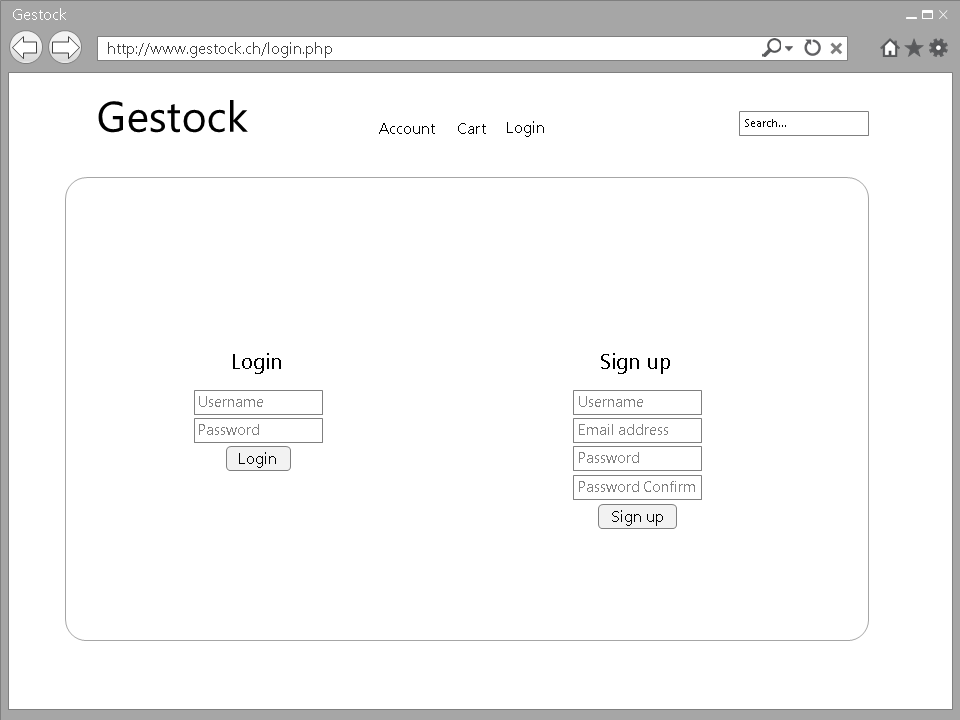
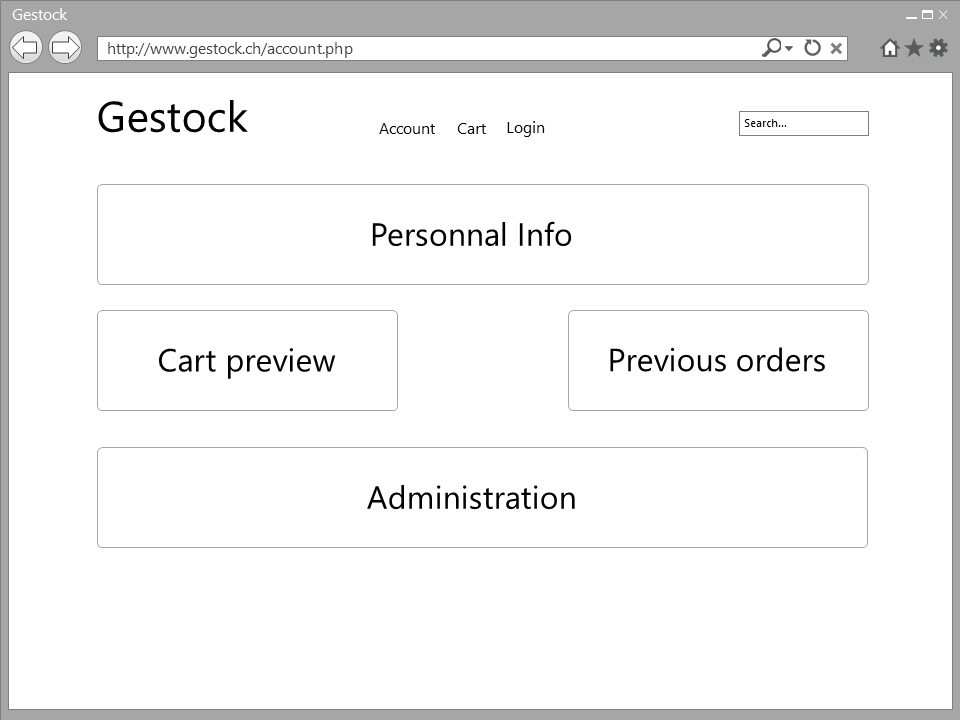


Figure 3

La page de login sera divisée en deux, une pour s’authentifier, et l’autre pour s’inscrire. Idéalement, j’aimerais également ajouter une case à cocher « Remember me » sous le login, afin que l’utilisateur n’ait pas à s’authentifier la prochaine fois qu’il vient sur le site.

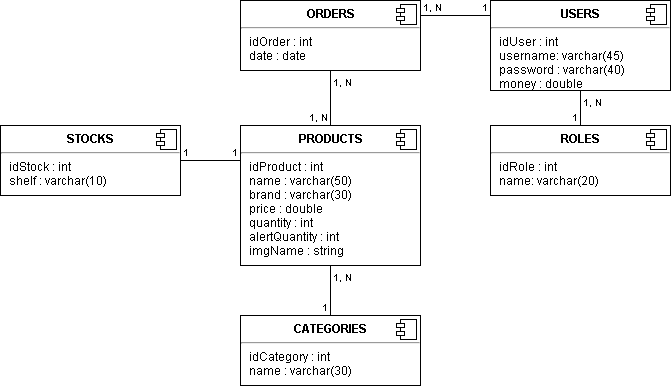


L’utilisateur aura accès à sa page personnelle depuis cette page. En haut, nous avons ses informations personnelles, son nom d’utilisateur, email, argent et la possibilité de changer son mot de passe.

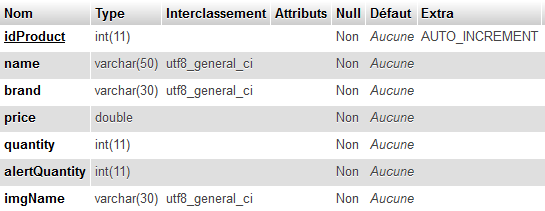
Au milieu nous avons deux preview. À gauche, cette du panier actuel. Les cinq produits les plus chers seront affichés ici. On pourra accéder à leur page de détails en cliquant dessus. Un lien en bas permettra d’accéder à la page du panier en complet. À droite, nous aurons la preview de la liste des commandes passées, avec les 5 dernières affichées. On pourra directement accéder aux informations d’une commande, ou accéder à la liste complète.

En bas, nous aurons une partie qui sera uniquement accessible par un administrateur. C’est ici que celui-ci aura les liens pour accéder aux pages de gestions, telles que la page de gestion des produits, celle de gestion des utilisateurs, ainsi que la page qui nous avertira des produits bientôt hors stocks.

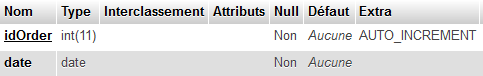
## Base de données



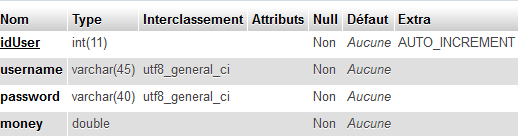
**Products :** Table contenant les données pour chaque produit. Un produit a également une catégorie, et une place dans un stock.



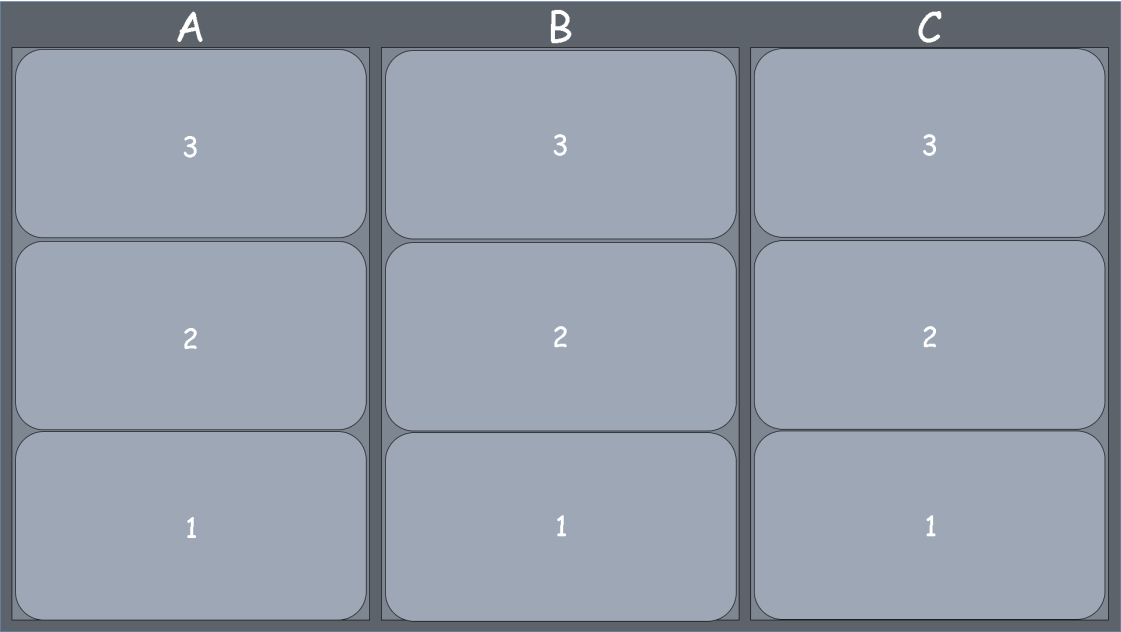
**Orders :** Table contenant les données pour une certaine commande. Il n’y a pas de table pour les paniers d’achats, donc c’est cette table-là qui s’en occupera. Vu que le champ « date » n’est pas obligatoire, le un panier devient une commande uniquement dès qu’il a une date. Cette table contiendra également l’id de l’utilisateur de la commande, ainsi que l’id des produits concernés.



**Users :** Table contenant les données pour chaque utilisateur. Un utilisateur a également un rôle.



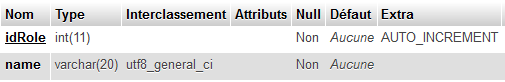
**Stocks :** Table contenant les données pour un stock. Un stock est un emplacement, là où sont stockés les produits. Voilà le système qui sera utilisé :



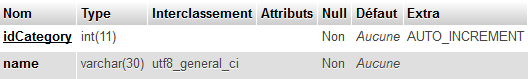
On dira donc, par exemple, que le produit X est stocké dans le stock (ou étagère) A2, ce qui veut dire qu’il est dans le deuxième étage de l’étagère A. Réalisation



**Roles :**

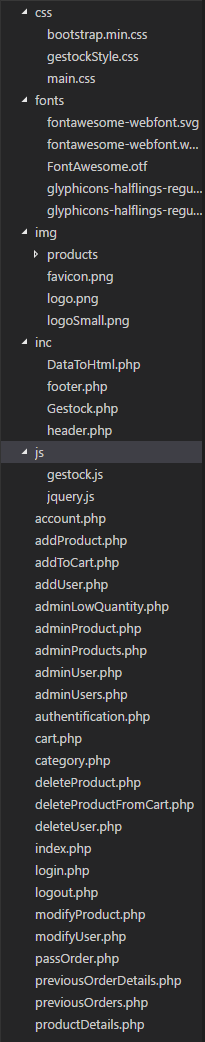


**Categories :**



# Réalisation

## Arborescence du site



Comme parlé avant, j’ai utilisé un template boostrap que j’ai modifié. Au début, il y avait bien plus de fichiers dans les dossiers CSS et JS. J’ai donc fait le tri et gardé uniquement ce dont j’avais besoin.

### Dossier « css »

Tout le style du site est stocké ici.

**« boostrap.min.css »** Classes Bootstrap utilisées dans le site.

**« main.css »** Style que j’ai décidé de garder du template initial.

**« gestockStyle.css »** Style que j’ai ajouté afin de combler des besoins spécifiques au site.

### Dossier « fonts »

Ceci contient les informations pour les quelques icônes que j’utilise dans le site. Allait à la base de pair avec un fichier CSS nommé « font-awesome.css » que j’ai mergé avec « gestockStyle.css » en y mettant uniquement ce qui était nécessaire.

### Dossier « img »

À la racine, on trouve les images du site : La petite icône pour l’onglet, la version petite du logo du site qui se trouve en haut à gauche de chaque page, ainsi que le logo du site dans sa taille originale.

Le dossier « products » contient les images de tous les produits. Les images des produits ajoutés depuis le site sont directement envoyées ici.

### Dossier « inc »

Ici se trouvent tous les fichiers PHP qui ne servent qu’à être ajoutés à des pages afin d’utiliser leu fonctions.

### Dossier racine

C’est ici que se trouvent tous les fichiers PHP de base, donc ceux d’affichage et les scripts.

**« DataToHtml.php »** Cette classe sert à mettre en forme certaines données. [Cliquez ici pour plus d’informations.](_DataToHTML.php#_Classe_)

**« footer.php »** Ce fichier est inclus à chaque fin de page d’affichage. C’est lui qui contient tout le footer, c’est-à-dire la partie du bas du site. (Barre d’informations, inclusions de fichiers JS, etc.)

**« Gestock.php »** C’est la classe principale du site. [Cliquez ici pour plus d’informations](_Gestock.php#_Classe_).

**« header.php »** Ce fichier est inclus à chaque début de page d’affichage. C’est lui qui contient tout le header, c’est-à-dire la partie du haut du site. (Barre de navigation, inclusions de fichiers css, etc.)

### Dossier « js »

Contient tous les scripts JS du site.

**« jquery.js »** Contient toutes les fonctions et script de la librairie JQuery.

**« gestock.js »** Contient uniquement trois fonctions faîtes spécialement pour le site. Plus plus d’informations, allez dans le point **5.2. Conception du code**.

## Conception du code

### Fichier « gestock.js »

Ceci est le seul fichier dont je parlerais dans la conception du code.

Ce fichier contient trois fonctions :

La première, « addPopup », crée un popup sur la page actuelle, avec le message entré. Cette fonction est notamment utilisée pour informer l’utilisateur de ce qui se passe sur le site. S’il a correctement ajouté un produit à son panier, s’il lui manque une information pour créer son compte, si sa commande s’est bien/mal déroulée, c’est cette fonction-là qui lui fera passer le message.

Le deuxième, « addToCart », est une fonction ajax qui est appelée quand on clique sur les boutons « Add to cart ».

### Classe « Gestock.php »

Cette classe est celle qui gère tous les appels à la BDD, c’est donc la classe principale du site.

Elle est conçue avec le design pattern du singleton. Celui-ci empêche la classe d’être instanciée, ce qui est pratique dans le cas où un seul objet doit gérer un système, ce qui est le cas dans mon site.

La première fois où le site sera ouvert, une instance de l’objet sera initialisée, et ce sera la seule. Dans le constructeur, beaucoup de choses se passent. Premièrement, on initialise un objet PDO afin de pouvoir faire les appels à la base de données. Ensuite, on prépare **toutes** les requêtes SQL qui seront utilisées dans le site. On pourrait croire que ce n’est pas optimisé, mais ça l’est, et voici pourquoi : Dans le cadre d’un site web, il y a plusieurs utilisateurs au même temps. Le fait de préparer les requêtes à l’avance fait que le server MySql n’a qu’à exécuter les requêtes, au lieu de les analyser à chaque fois, puis les exécuter.

Chaque requête est ensuite encapsulée dans une fonction, où on lie les paramètres avec les valeurs venant du site, puis on les exécute.

Toutes les exceptions jetées seront ajoutées au fichier log, cependant, dépendant du type d’exception relevé, elle sera traitée de manière différente. Celles qui ne sont pas bloquantes et qui doivent informer l’utilisateur qu’il a mal entré quelque chose (par exemple le cas où il a entré une combinaison utilisateur/mot de passe inexistant), sont retournées dans le site et sont affichées à l’utilisateur, afin de le guider. Cependant, celles qui sont bloquantes (dans le cas où la BDD n’est pas accessible, par exemple), on arrête le script et affichons une page blanche d’erreur. Un exemple serait dans la page d’accueil, si la BDD n’est pas disponible, les produits ne pourront pas être affichés, ce qui est embêtant vu que c’est la fonction principale de cette page.

**Attention,** la fonction « getNbProducts » est à utiliser **uniquement** après une requête SQL contenant la fonction « SQL\_CALC\_FOUND\_ROWS » comme par exemple la fonction « getProducts » ou encore, « getProductsOfCategory », sinon une erreur sera retournée. Cette fonction fonctionne de cette façon car j’utilise une méthode appelée « FOUND\_ROWS() », qui récupère la valeur calculée par « SQL\_CALC\_FOUND\_ROWS ». Si vous n’avez pas calculé la valeur, « FOUND\_ROWS() » n’aura rien à retourner.

#### Fonctions complexes

##### insertProductIntoCart()

Ajoute un produit dans le panier d’achat.



##### passOrder()

Cette fonction permet de « transformer » un panier en commande.



### Classe « DataToHTML.php »

Cette classé est utilisée pour mettre en forme les données avec les balises HTML. Elle est entièrement statique car il n’y a aucune valeur qui doit être stockée.

Elle n’est clairement pas nécessaire, vu que j’aurais pu faire le code dans les pages HTML, mais ça permet d’avoir celles-ci plus propres, ainsi qu’éviter une certaine redondance, vu que la même fonction peut parfois m’être utile dans 2-3 pages en même temps.

### Scripts

Ces pages exécutent leur fonction en récupérant les données des pages HTML pour les traiter à l’aide de la classe « Gestock.php ».

#### « addProduct.php »

Ajouter un produit dans la base de données.

Fonctionnement :

* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai d’insérer du produit.
* Redirige l’administrateur selon le résultat de l’insertion.

#### « addToCart.php »

Ajouter un produit au panier de l’utilisateur connecté.



#### « addUser.php »

Ajouter un utilisateur dans la base de données.



#### « authentification.php »

Authentifier l’utilisateur.



#### « deleteProduct.php »

Cette page permet de supprimer un produit avec les données envoyées depuis les pages HTML et traitées par la classe Gestock.php.

Fonctionnement :

* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai de supprimer le produit.
* Redirige l’administrateur selon le résultat de la suppression.

#### « deleteProductFromCart.php »

Cette page permet d’ajouter un produit avec les données envoyées depuis les pages HTML et traitées par la classe Gestock.php.



#### « deleteUser.php »

Fonctionnement :

* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai de supprimer l’utilisateur.
* Redirige l’administrateur selon le résultat de la suppression.

#### « modifyProduct.php »

Fonctionnement :

* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai de mettre à jour le produit.
* Redirige l’administrateur selon le résultat de la MàJ.

#### « modifyUser.php »

Fonctionnement :

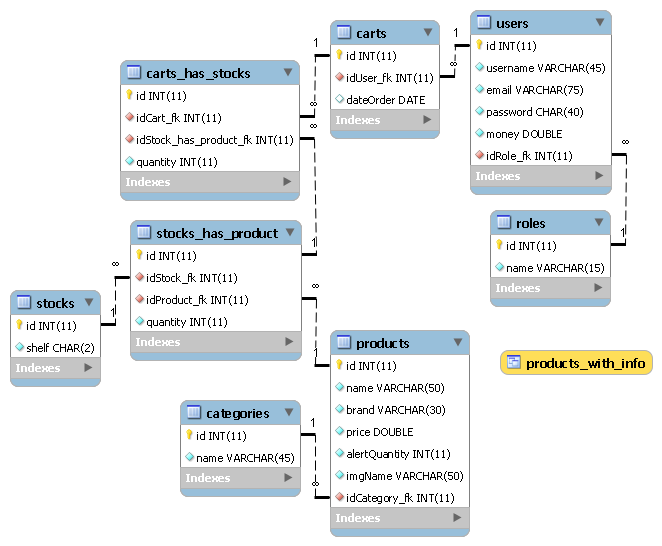
* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai de mettre à jour l’utilisateur.
* Redirige l’administrateur selon le résultat de la MàJ.

#### « passOrder.php »

Fonctionnement :

* Redirige l’utilisateur s’il n’a pas le droit d’être là.
* Toutes les variables prennent les bonnes valeurs.
* Essai de passer la commande.
* Redirige l’administrateur selon le résultat de la MàJ.

## Modèle de BDD final



Il y a eu quelques petits changements par rapport au modèle relationnel pensé au début.

Premièrement, la table « orders » dans le menu relationnel s’est faite renomme en « carts ». Le système pour définir quand le panier devient une commande reste le même, mais les liens entre les tables change. Avant, il y avant une table de liaison entre « orders » et « products », mais cela voulait dire que tout le stock d’un produit est à un seul endroit. Bien que dans ce projet-là, ce soit le cas, j’ai pensé au jour où on voudra ajouter d’autres emplacements pour le produit. Du coup, pour ce faire, j’ai ajouté une table de liaison entre « stocks » et « products » appelée « stocks\_has\_product » qui contient l’id du stock, celui du produit et sa quantité. Maintenant qu’il y a cette table, je l’ai liée avec la table « carts » à l’aide d’une autre table de liaison.

Deuxièmement, j’ai ajouté le champ ‘email’ dans la table users. C’est un champ logique à avoir, mais je l’avais oublié.

Ensuite, j’ai changé l’interclassement des colonnes ‘username’ & ‘password’ de la table users. L’interclassement est passé de UTF8\_CI à UTF8\_CC car j’ai trouvé plus logique que ces champs soient distingués pour chaque user par rapport aux majuscules.

De plus, j’ai ajouté un index FULLTEXT sur le champ ‘name’ afin de faciliter et d’accélérer les recherches sur ce champ.

Pour finir, j’ai ajouté une vue. En commençant, j’ai fait une vue avec tous les produits et leur catégories, appelée « products\_with\_category ». Je l’avais utilisé qu’une fois, jusqu’au moment où j’ai remarqué que j’avais également besoin de savoir la quantité de l’objet dans le stock, ce qui a fait que j’ai remodifié la vue pour afficher également la quantité. On se retrouve alors avec une vue appelée « products\_with\_info » qui contient tous les champs de products, ainsi que la catégorie du produit et sa quantité.

# Tests

J’ai testé mon site page par page, voici donc la liste complète des tests que j’ai pu faire, séparés par page.

**Index.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Aller sur l'index du site. | Se retrouver face à neuf produits avec la pagination. | OK |
| 2 | Cliquer sur suivant/précédent. | Passe à la page suivante/précédente. | OK |
| 3 | Cliquer sur un numéro de page. | Passe à la page correspondante. | OK |
| 4 | Se retrouver sur la première/dernière page. | Les flèches suivant/précédent sont correctement affichées. | OK |
| 5 | Taper index.php avec une page non-existante. | Retour à la page d'accueil. | OK |
| 6 | Cliquer sur "Add to cart" en étant connecté. | Ajoute le produit au panier. | OK |
| 7 | Cliquer sur "Add to cart" en n'étant pas connecté. | Redirige l'utilisateur sur la page de connexion. | OK |

**Category.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Aller sur une catégorie. | Se retrouver face aux produits de cette catégorie. | OK |
| 2 | Cliquer sur suivant/précédent. | Passe à la page suivante/précédente. | OK |
| 3 | Cliquer sur un numéro de page. | Passe à la page correspondante. | OK |
| 4 | Taper category.php avec une page ou un id non-existant. | Retour à la page d'accueil. | OK |
| 5 | Cliquer sur "Add to cart" en étant connecté. | Ajoute le produit au panier. | OK |
| 6 | Cliquer sur "Add to cart" en n'étant pas connecté. | Redirige l'utilisateur sur la page de connexion. | OK |

**ProductDetails.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Cliquer sur un produit. | Redirection sur sa page de détails. | OK |
| 2 | Taper productDetails.php avec un id non-existant. | Retour à la page d'accueil. | OK |
| 3 | Cliquer sur "Add to cart" en étant connecté. | Ajoute le produit au panier. | OK |
| 4 | Cliquer sur "Add to cart" en n'étant pas connecté. | Redirige l'utilisateur sur la page de connexion. | OK |
| 5 | Le produit est en stock. | Affiche "In stock" en vert. | OK |
| 6 | Le produit n'est bientôt plus en stock. | Affiche "Low stock" en orange. | OK |
| 7 | Le produit n'est plus en stock. | Affiche "Out of stock" en rouge. | OK |

**Login.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller en étant déjà connecté. | Redirection vers la page du compte d'utilisateur. |  |
| 2 | Appuyer sur "Login" ou "Sign up" sans avoir entré d'informations. | Le navigateur affiche ce qu'il faut entrer. | OK |
| 3 | Essayer de s'inscrire avec de mauvaises informations. | Popup d'erreur informant l'utilisateur. | OK |
| 4 | Essayer de s'inscrire avec un nom d'utilisateur/email déjà pris. | Popup d'erreur informant l'utilisateur. | OK |
| 5 | Essayer de s'inscrire avec des mots de passe différents. | Popup d'erreur informant l'utilisateur. | OK |
| 6 | Essayer de s'inscrire avec toutes les informations entrées correctes. | Utilisateur correctement ajouté à la base + popup d'erreur informant l'utilisateur. | OK |
| 7 | Essayer de s'authentifier avec une combinaison (username/email)/password non-existante. | Popup d'erreur informant l'utilisateur. | OK |
| 8 | Essayer de s'authentifier avec une combinaison existante. | Connexion + redirection sur la page du compte d'utilisateur. | OK |

**Account.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'utilisateur. | Affichage des informations personnelles, preview du panier, ainsi que celui des commandes passées. | OK |
| 3 | Y aller en étant connecté en tant qu'administrateur. | Affichage de la même page que pour les utilisateurs + l'onglet de gestion. | OK |
| 4 | Cliquer sur "Go to cart" dans la preview du panier. | Redirection sur la page du panier. | OK |
| 5 | Cliquer sur "Check details" dans la preview des commandes passées. | Redirection sur la page de la commande. | OK |
| 6 | Cliquer sur "Get the full list". | Redirection sur la page des commandes. | OK |
| 7 | En tant qu'administrateur, cliquer sur "Manage Products". | Redirection sur la page de gestion des produits. | OK |
| 8 | En tant qu'administrateur, cliquer sur "Manage Users". | Redirection sur la page de gestion des utilisateurs. | OK |
| 9 | En tant qu'administrateur, cliquer sur "Low quantity products". | Redirection sur la page de gestion des produits bientôt hors stock. | OK |

**Cart.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté. | Redirection sur la page de connexion avec message "Login first !" | OK |
| 2 | Y aller en étant connecté. | Affichage des produits dans le panier actuel. | OK |
| 3 | Cliquer sur un produit. | Redirection sur sa page de détails. |  |
| 4 | Appuyer sur "+" ou "-" à côté d'un produit. | Ajoute/Supprime une fois le produit du panier. | OK |
| 5 | Appuyer sur "-" alors que le produit n'est qu'une fois dans le panier. | Supprime le produit du panier. | OK |
| 6 | Appuyer sur "X" à côté d'un produit. | Supprime le produit du panier. | OK |
| 7 | Le panier n'a pas de produit. | Affichage du panier vide. | OK |
| 8 | Essayer de passer une commande avec un panier vide. | Popup d'erreur avec pour message "Your cart has no products !". | OK |
| 9 | Essayer de passer une commande avec un panier contenant un ou plusieurs produits qui ne sont pas en stock. | Popup d'erreur avec pour message "Some items aren't in stock !". | OK |
| 10 | Essayer de passer une commande alors qu'on n’a pas assez d'argent. | Popup d'erreur avec pour message "You do not have enough money !". | OK |
| 11 | Essayer de passer une commande quand tout va bien. | Le panier se vide et une nouvelle commande avec tous les produits sélectionnés est ajoutée. L'argent de l'utilisateur est réduit du montant de la commande. | OK |

**PreviousOrders.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté. | Affichage de la date et du lien des commandes liées à l'utilisateur connecté. | OK |
| 3 | Cliquer sur "Check details" à d'une commande. | Redirection sur sa page de détails. | OK |

**PreviousOrderDetails.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté. | Redirection sur la page de connexion. |  |
| 2 | Y aller en étant connecté et que l'id de la commande correspond à l'utilisateur. | Affichage des produits de la commande, ainsi que son prix total. | OK |
| 3 | Y aller en étant connecté mais que l'id de la commande ne correspond pas à l'utilisateur. | Redirection sur la page d'accueil. | OK |
| 4 | Cliquer sur un produit. | Redirection sur sa page de détails. | OK |

**AdminProducts.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté ou en étant connecté en tant qu'utilisateur. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'administrateur. | Affichage de la liste des produits. | OK |
| 3 | Cliquer sur "Add a product" | Redirection sur la page pour ajouter un produit. | OK |
| 4 | Cliquer sur un produit. | Redirection sur sa page de détails. | OK |
| 5 | Cliquer sur suivant/précédent. | Passe à la page suivante/précédente. | OK |
| 6 | Cliquer sur un numéro de page. | Passe à la page correspondante. | OK |
| 7 | Cliquer sur "Modifier" sur un produit. | Redirection sur la page de modification du produit. | OK |
| 8 | Cliquer sur "Supprimer" sur un produit. | Suppression du produit et rafraichissement de la page. | OK |
| 9 | Une erreur BDD est renvoyée. | Rafraichissement de la page avec un popup pour avertir l'administrateur. | KO |

**AdminProduct.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté ou en étant connecté en tant qu'utilisateur. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'administrateur sans ID. | Affiche le formulaire vide. | OK |
| 3 | Essayer d'ajouter un produit sans avoir rempli tous les champs. | Le navigateur bloque la redirection et affiche un message d'information. | OK |
| 4 | Essayer d'ajouter un produit en ayant entré du texte dans la champ prix ou quantité. | Informe l'administrateur de son erreur. | KO |
| 5 | Essayer d'ajouter un produit en ayant correctement entré les champs. | Ajoute le produit et redirige sur la page des produits en administrateur. | OK |
| 6 | Y aller en étant connecté en tant qu'administrateur avec un ID incorrecte. | Redirige sur la page des produits en administrateur. | OK |
| 7 | Y aller en étant connecté en tant qu'administrateur avec un ID correcte. | Affiche le formulaire rempli avec les valeurs du produit. | OK |
| 8 | Essayer de mettre à jour le produit sans avoir entré tous les champs texte. | Le navigateur bloque la redirection et affiche un message d'information. | OK |
| 9 | Essayer de mettre à jour le produit en ayant entré du texte dans le champ prix ou quantité. | Informe l'administrateur de son erreur. | KO |

**AdminUsers.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté ou en étant connecté en tant qu'utilisateur. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'administrateur. | Affichage des utilisateurs du site. | OK |
| 3 | Cliquer sur "Modifier" sur un utilisateur. | Redirection sur la page de modification de l'utilisateur. | OK |
| 4 | Cliquer sur "Supprimer" sur un utilisateur. | Supprimer l'utilisateur et rafraichis la page. | OK |
| 5 | Une erreur BDD est renvoyée. | Rafraichissement de la page avec un popup pour avertir l'administrateur. | KO |

**AdminUser.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté ou en étant connecté en tant qu'utilisateur. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'administrateur avec un ID invalide. | Redirection sur la page du listing des utilisateurs. | OK |
| 3 | Y aller en étant connecté en tant qu'administrateur avec un ID valide. | Affichage du formulaire avec les champs remplis. | OK |
| 4 | Essayer de modifier l'utilisateur avec des champs non remplis. | Le navigateur bloque la redirection et affiche un message d'information. | OK |
| 5 | Essayer de modifier l'utilisateur avec le champ "Money" avec du texte. | Informe l'administrateur de son erreur. | KO |

**AdminLowQuantity.php**

|  |  |  |  |
| --- | --- | --- | --- |
| **N°** | **Scénario** | **Résultat attendu** | **OK/KO** |
| 1 | Y aller sans être connecté ou en étant connecté en tant qu'utilisateur. | Redirection sur la page de connexion. | OK |
| 2 | Y aller en étant connecté en tant qu'administrateur. | Affichage des produits bientôt hors-stock ou hors-stock. | OK |
| 3 | Appuyer sur "Restock" sur un produit. | Redirection sur sa page de modifications. | OK |

# Guide d’installation

Il va premièrement falloir vous munir d’un serveur Apache/PhP/Mysql local. Dans mon cas, j’ai utilisé EasyPhp.

Allez dans la page d’administration du serveur, et rendez-vous dans la partie de la gestion de bases de données. Ensuite, importez le fichier « db\_gestockDB\_full.sql » et exécutez-le. Ce fichier contient le script pour créer la base de données et la remplir avec les informations nécessaires au début, comme des produits, leurs catégories, des utilisateurs, leurs rôles, ainsi que les stocks. Voilà qui est fait pour la base de données.

Concernant les sources du site WEB, allez à la racine de votre serveur afin d’y mettre le dossier « Web » du projet. Vous pouvez également laisser le dossier là où il est, et créer un alias dans votre page d’administration, si vous le préférez.

Vous êtes désormais prêts à utiliser le site web. L’utilisateur qui a les droits dans la base de données est directement installé avec le reste de la base de données, il n’y a donc pas de problèmes à se faire à ce niveau. Voici tous les identifiants à retenir :

**Base de données :**



**Site web :**





Je me suis permis d’ajouter les identifiants/mots de passe ici afin d’indiquer au correcteur comment s’y prendre. En temps normal, il faudrait lui les transmettre en privé.

# Conclusion

## Objectifs

Les objectifs du cahier des charges ont été entièrement accomplis.

Cependant, j’aurais aimé pourvoir améliorer la sécurité dans les formulaires administrateur. Je l’es ai mises sur ceux des utilisateurs lambda, mais je n’ai pas eu le temps de le faire pour les admins.

J’aurais également aimé pouvoir faire en sorte que l’utilisateur puisse changer ses informations personnelles. Peut-être pas toutes, mais au moins son mot de passe.

## Points

### Forts

* Interface agréable et intuitive.
* Rapide.
* L’information est à portée de main, rangé dans un ordre logique.
* Ajouter des produits dans le panier est simple.
* Passer une commande est simple.
* Gestion des produits efficace. (+Affichage des produits peu de stock.)
* Gestion des utilisateurs efficace.
* Sécurité du site

### Faibles

* Gestion d’éventuelles erreurs manquantes en tant qu’administrateur.
* Tri des produits uniquement par catégorie.
* Pas entièrement responsive.

## Difficultés particulières

1. Pagination « Universelle » : Bien que la pagination ait été facile à faire, j’ai passé pas mal de temps à essayer de faire en sorte d’avoir une seule fonction qui fait la pagination pour toutes les pages, quelle qu’elle soit.
2. Bien que le processus pour passer une commande n’ait pas réellement été compliqué à réaliser, il a été assez long car j’avais une bonne quantité de tests à faire.
3. Je m’attardais parfois sur des petites choses sans réelle importance alors que j’avais des fonctionnalités plus grosses à faire derrière. Je n’arrive pas à laisse quelque chose que je trouve lait dans mon code.

## Suite possibles pour le projet

Avant que Gestock soit un site fonctionnel, on doit encore lui ajouter pas mal de chose ! Voici ce que j’envisagerais à mettre en place si j’avais à le continuer :

1. Une cache à cocher « Se souvenir de moi » à la connexion afin de faciliter la vie d’un utilisateur qui viendrait souvent.
2. Un vrai système de paiement. Actuellement le site est géré avec un porte-monnaie virtuel, ça serait donc bien mieux de faire en sorte que le paiement soit réel.
3. Permettre à l’utilisateur de changer ses informations personnelles, ou au moins son mot de passe.
4. Mettre en place un système de récupération de mot de passe. Les utilisateurs qui perdent leur mot de passe arrivent souvent, et un système automatisé qui leur permet de les retrouver serait une bonne chose de faite.
5. Système de commentaires et de classement des produits.
6. Comme dit plus haut, les administrateurs doivent être bien attentifs quand ils gèrent le site, sous risque de faire une bêtise. Ce serait donc bien d’améliorer la sécurité liée à leur manipulation sur le site. Les utilisateurs en ont le droit, pourquoi pas eux ?
7. Actuellement, les produits ne peuvent être triés que par catégorie, mais il serait bien d’ajouter d’autres méthodes, comme par exemple le prix, la marque, ou encore la pertinence.
8. Informations de contact dans le footer.
9. Système d’inscription plus sécurisé avec confirmation par email et captcha « Je ne suis pas un robot » afin d’éviter les personnes malveillantes qui voudraient essayer de « détruire » le site.
10. Ajouter un système de commentaires/notation des produits afin d’aider les utilisateurs à choisir les bons produits.

# Bibliographie

<https://en.wikipedia.org/wiki/Singleton_pattern> (Informations sur le singleton)

<http://php.net/> (PHP)

<http://sql.sh/> (SQL)

<http://jquery.com/> (JQuery)

<http://sql.sh/cours/limit> (Pagination)