|  |
| --- |
| 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

[3 Introduction 2](#_Toc485132434)

[4 Analyse préliminaire 2](#_Toc485132435)

[4.1 Organisation 2](#_Toc485132436)

[4.1.1 Inventaire d’applications 2](#_Toc485132437)

[4.2 Analyse concurrentielle 3](#_Toc485132438)

[4.3 Cahier des charges détaillé 3](#_Toc485132439)

[4.3.1 Objectifs 3](#_Toc485132440)

[4.3.2 Définition de l’audience 4](#_Toc485132441)

[4.4 Détermination des points faibles 4](#_Toc485132442)

[4.5 Plan de tests 4](#_Toc485132443)

[5 Conception 4](#_Toc485132444)

[5.1 Plan fonctionnel du site 5](#_Toc485132445)

[5.2 Maquettes préliminaires 5](#_Toc485132446)

[5.3 Base de données 7](#_Toc485132447)

[6 Réalisation 8](#_Toc485132448)

[6.1 Conception du code 8](#_Toc485132449)

[6.1.1 Classe « Gestock.php » 8](#_Toc485132450)

[6.1.2 Classe « DataToHTML.php » 9](#_Toc485132451)

[6.1.3 Script « addToCart.php » 9](#_Toc485132452)

[6.1.4 Script « addUser.php » 10](#_Toc485132453)

[6.1.5 Script « authentification.php » 10](#_Toc485132454)

[6.1.6 Script « deleteProductFromCart.php » 11](#_Toc485132455)

[6.2 Modèle de BDD final 11](#_Toc485132456)

[7 Bibliographie 11](#_Toc485132457)

# 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 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 Apache, PHP et MySql local.

**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

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

**Points positifs repris chez digitec :**

* Accès rapide à un produit désiré.
* Différents filtres de recherche pour un produit.
* Bonne description d’un produit, groupé par sous-catégories.
* Informations personnelles facilement accessibles.
* Bon listing des différentes commandes passées.
* Ajout/Suppression facile d’un produit dans le panier d’achats.
* La commande est facilement réalisable.

**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.

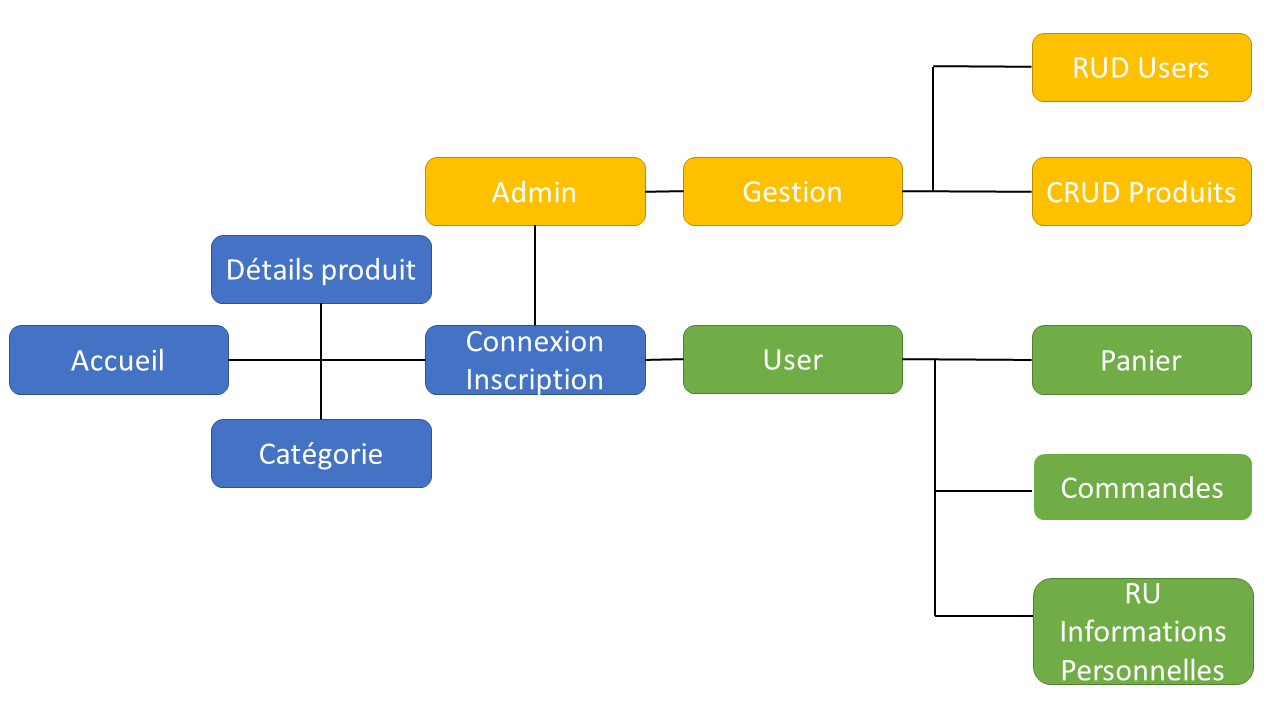
## Plan de tests

Plan de tests réalisé par mes soins.

# Conception

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

## Plan fonctionnel du site



## 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.

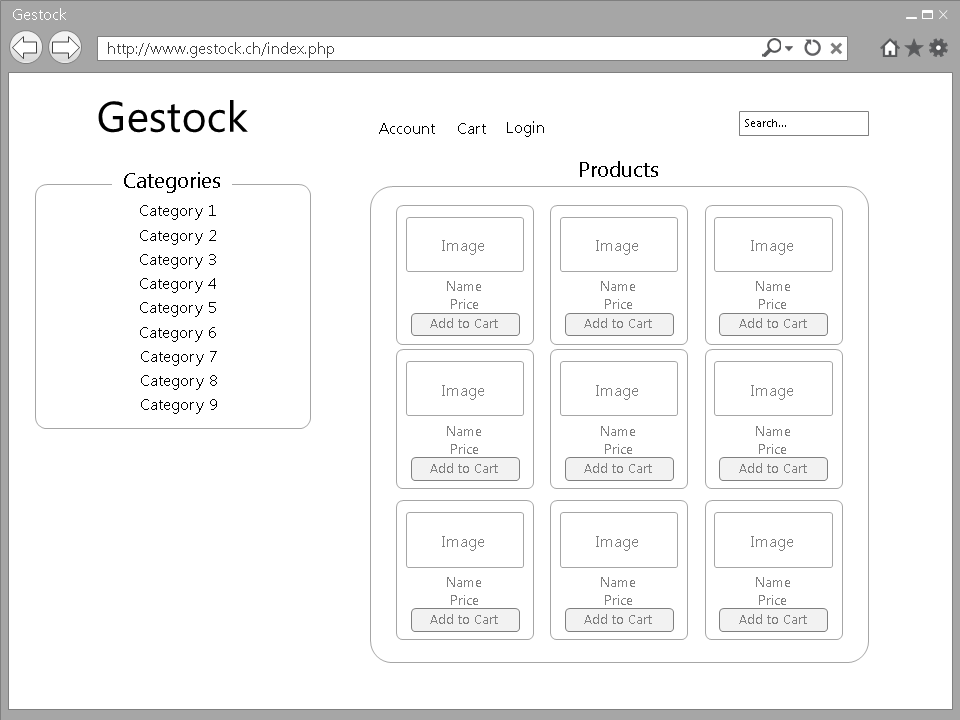


Figure 1

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.

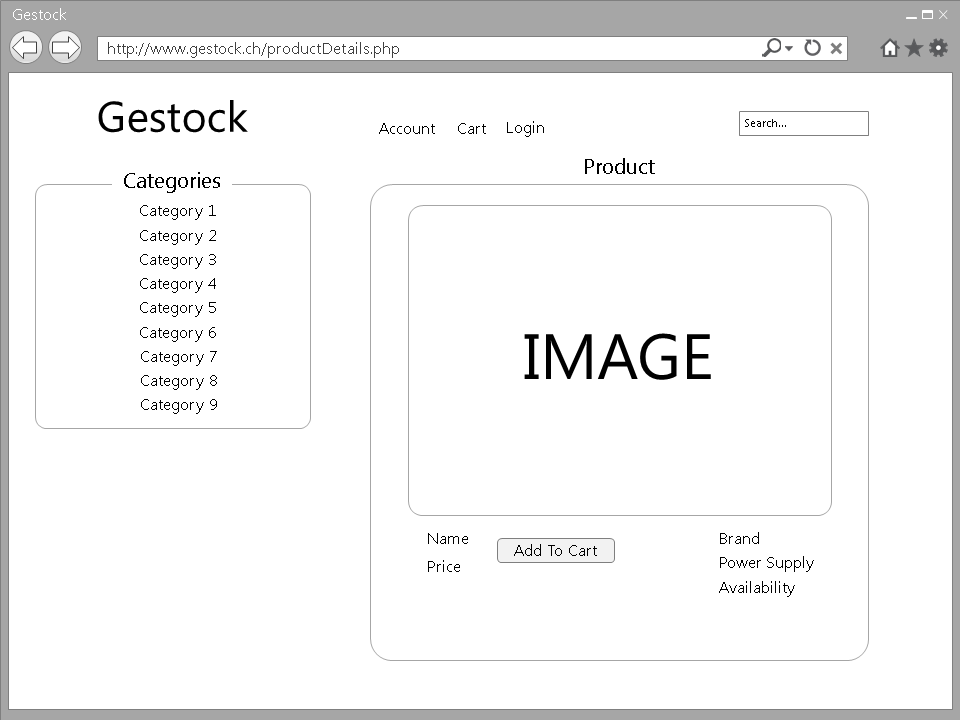


Figure 2

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.

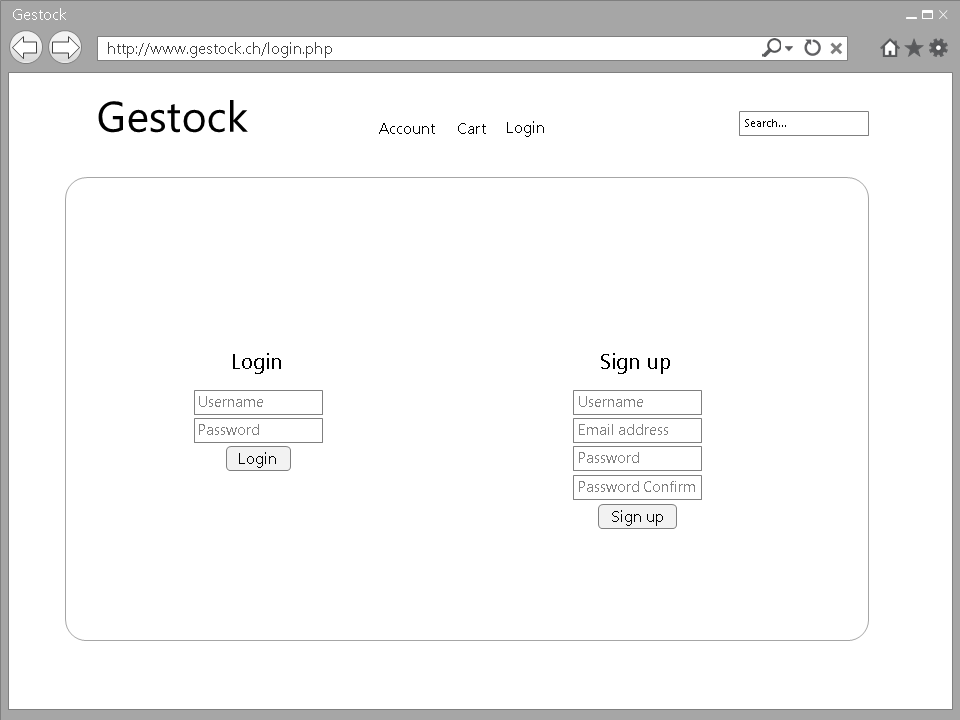
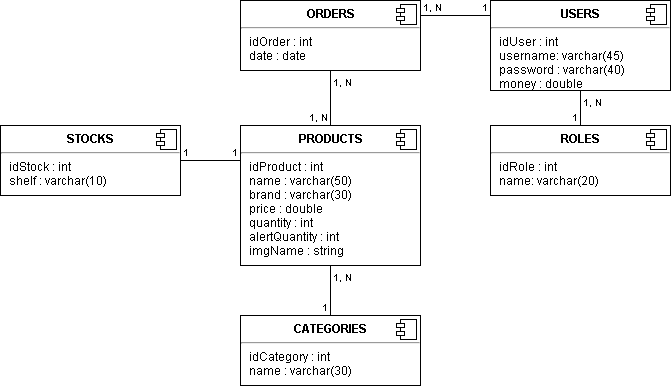


Figure 3

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.

## Base de données

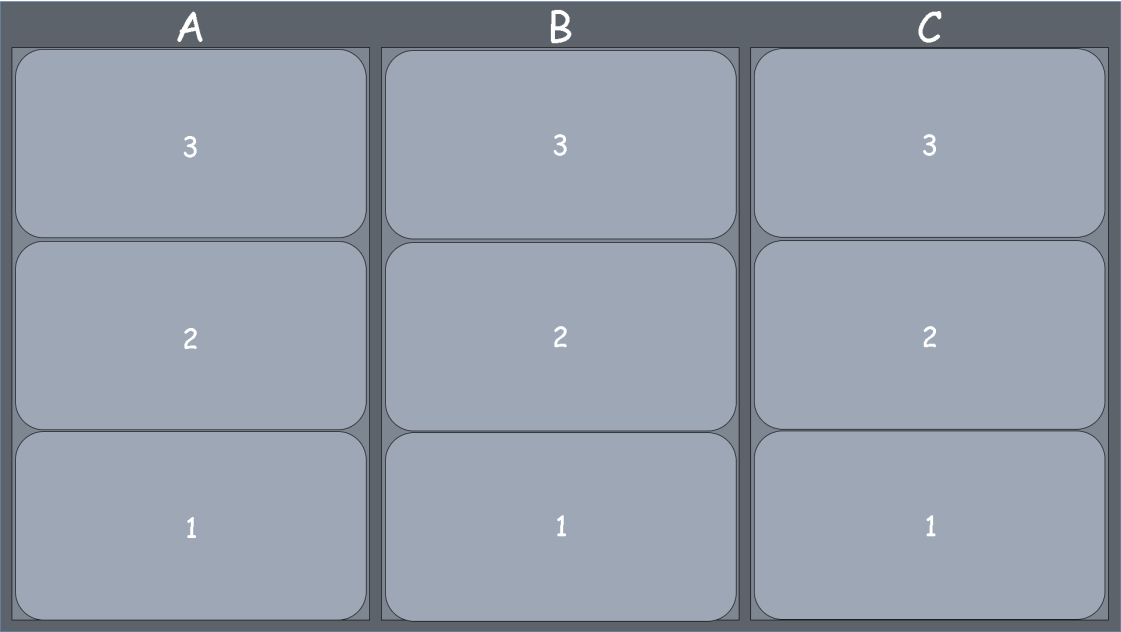


**Products :** Table contenant les données pour chaque produit. Un produit a également une catégorie, et une place dans un stock.

**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

## Conception du code

### 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.

### 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.

### Script « addToCart.php »



### Script « addUser.php »



### Script « authentification.php »



### Script « deleteProductFromCart.php »



## Modèle de BDD final

**[TO\_DO]**

# Bibliographie

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

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

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

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

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