

Damjan Jovanovic

I.DA-P4A

2018



Table des matières

[1 Introduction 5](#_Toc514942627)

[2 Rappel du cahier des charges 5](#_Toc514942628)

[2.1 But 5](#_Toc514942629)

[2.2 Spécifications 5](#_Toc514942630)

[2.3 Restrictions 5](#_Toc514942631)

[2.4 Environnement 5](#_Toc514942632)

[2.5 Livrables 6](#_Toc514942633)

[2.6 Modèle conceptuel de base 6](#_Toc514942634)

[2.7 Planning initial 7](#_Toc514942635)

[2.8 Plan du site 7](#_Toc514942636)

[3 Analyse fonctionnelle 8](#_Toc514942637)

[3.1 Fonctionnalités 8](#_Toc514942638)

[3.1.1 Inscription 8](#_Toc514942639)

[3.1.2 Connexion 8](#_Toc514942640)

[3.1.3 Validations utilisateurs 8](#_Toc514942641)

[3.1.4 Ajout de produits 8](#_Toc514942642)

[3.1.5 Modification de produits 8](#_Toc514942643)

[3.1.6 Suppression de produits 8](#_Toc514942644)

[3.1.7 Commande 8](#_Toc514942645)

[3.1.8 Historisation des prix 8](#_Toc514942646)

[3.2 Interfaces 9](#_Toc514942647)

[3.2.1 Utilisateurs non-authentifiés 9](#_Toc514942648)

[3.2.2 Utilisateurs authentifiés 14](#_Toc514942649)

[3.2.3 Administrateur 17](#_Toc514942650)

[4 Analyse organique 23](#_Toc514942651)

[4.1 Diagramme de la base de données (MLD) 23](#_Toc514942652)

[4.2 Table articles 24](#_Toc514942653)

[4.3 Table articles\_concernant\_commande 24](#_Toc514942654)

[4.4 Table categories 24](#_Toc514942655)

[4.5 Table client\_passant\_commande 24](#_Toc514942656)

[4.6 Table clients 24](#_Toc514942657)

[4.7 Table factures 25](#_Toc514942658)

[4.8 Table panier\_commandes 25](#_Toc514942659)

[4.9 Table portemonnaie 25](#_Toc514942660)

[4.10 Table prixarticles 25](#_Toc514942661)

[4.11 Tables utilisateurs 25](#_Toc514942662)

[4.12 Plan du site final 26](#_Toc514942663)

[4.13 Fonctions php 27](#_Toc514942664)

[4.13.1 checkNameLastName($var) 27](#_Toc514942665)

[4.13.2 OnlyNumbers($var) 27](#_Toc514942666)

[4.13.3 MinPwd($var) 27](#_Toc514942667)

[4.13.4 Menu() 27](#_Toc514942668)

[4.13.5 getCategorie() 28](#_Toc514942669)

[4.13.6 getArticleByCategories($categorie) 28](#_Toc514942670)

[4.13.7 getArticleInfo($id) 28](#_Toc514942671)

[4.13.8 login($login,$pwd) 28](#_Toc514942672)

[4.13.9 RegisterClient ($nom, $prenom, $adresse, $codePostal, $ville, $pays, $telephone, $mail, $motPasse, $idUser) 28](#_Toc514942673)

[4.13.10 user\_exists($mail) 28](#_Toc514942674)

[4.13.11 registerUser($mail, $password1) 29](#_Toc514942675)

[4.13.12 createWallet($idUser) 29](#_Toc514942676)

[4.13.13 getUserInformation($email) 29](#_Toc514942677)

[4.13.14 getNotValidateUsers 29](#_Toc514942678)

[4.13.15 validateUser($idUser) 29](#_Toc514942679)

[4.13.16 addPrice($prixArticle, $dateFin, $idArticle) 29](#_Toc514942680)

[4.13.17 updatePrice($idArticle, $idPrixArticle) 30](#_Toc514942681)

[4.13.18 updatePriceArticle($idArticle, $idPrixArticle) 30](#_Toc514942682)

[4.13.19 addArticle($nomArticle, $imageArticle, $descriptionArticle, $stock, $idCategorie, $idPrixArticle) 30](#_Toc514942683)

[4.13.20 updateArticle($nomArticle,$imageArticle,$descriptionArticle,$stock, $idCategorie, $idPrixArticle, $idArticle) 30](#_Toc514942684)

[4.13.21 deleteArticle($idArticle) 30](#_Toc514942685)

[4.13.22 checkoutCart($numCommande, $idClient) 30](#_Toc514942686)

[4.13.23 addArticleToCart($taille, $idClient, $idArticle, $idPrixArticle) 31](#_Toc514942687)

[4.13.24 updateStock($idArticle) 31](#_Toc514942688)

[4.13.25 getCart($idClient) 31](#_Toc514942689)

[4.13.26 deleteArticleFromCart($idClient, $idCommande) 31](#_Toc514942690)

[4.13.27 getWallet($idUser) 31](#_Toc514942691)

[4.13.28 updateWallet($idUtilisateur, $solde) 32](#_Toc514942692)

[4.13.29 createBills($montant, $allIdCommand) 32](#_Toc514942693)

[4.13.30 getAllCommands($idUser) 32](#_Toc514942694)

[4.13.31 getDetailsCommand($idClient,$numCommand) 32](#_Toc514942695)

[4.13.32 getArticles($categorie,$order,$way) 33](#_Toc514942696)

[5 Tests 34](#_Toc514942697)

[6 Conclusion 39](#_Toc514942698)

[6.1 Planning final 39](#_Toc514942699)

[6.2 Difficulté rencontrée 39](#_Toc514942700)

[6.3 Mon ressenti 39](#_Toc514942701)

[6.4 Améliorations possibles 40](#_Toc514942702)

[7 Bibliographies 40](#_Toc514942703)

[7.1 Site utilisés 40](#_Toc514942704)

[8 Table des illustrations 41](#_Toc514942705)

[9 Annexes 41](#_Toc514942706)

# Introduction

Cette documentation a pour but de détailler le fonctionnement de l’application web développée dans le cadre du TPI (Travail Pratique Individuel). Elle est destinée aux experts évaluant le travail ainsi qu’aux personnes susceptible de reprendre le projet.

# Rappel du cahier des charges

## But

Le but du projet est de pouvoir vendre des vêtements sportwears depuis l’application web.

## Spécifications

* La base de données contient les clients, les vêtements, les factures, les prix et le stock.
* L'administrateur gère les produits (CRUD complet depuis le site).
* Le système permet au visiteur authentifié la gestion d'un panier d'achat (sélection de plusieurs articles).
* Le visiteur authentifié possède un porte-monnaie virtuel alimenté par l’administrateur.
* Le visiteur ajoute des produits dans un panier.
* Le panier peut être commandé, le stock est recalculé, le montant est débité.
* Les changements de prix sont historisés.
* Le site gère l'inscription et l'authentification selon les normes du WEB selon la séquence demande inscription -> validation par l’administrateur -> confirmation par email (simulé) ->confirmation d’inscription. Un mécanisme anti robot est à mettre en place.
* Les mots de passe sont encryptés.
* La base de données des produits sera réalisée et pré-remplie de quelques articles avant le début du travail, de même que l’utilisateur admin sera déjà dans la table utilisateurs.

## Restrictions

* Aucune

## Environnement

Matériels nécessaires avant le développement du site :

* Ordinateur de type PC
* Système d’exploitation : Windows 10 Entreprise
* Outil de développement : JetBrains PhpStorm, WAMP

## Livrables

En 1 exemplaire :

* Carnet de bord
* Documentation technique
* Mode d’emploi
* Code source

Par messageries aux experts et au maître d’apprentissage :

* PDF de la documentation technique
* PDF du mode d’emploi
* PDF des sources imprimés du logiciel

Par messagerie au maître d’apprentissage :

* Les sources (projet logiciel) et la version exécutable

## Modèle conceptuel de base

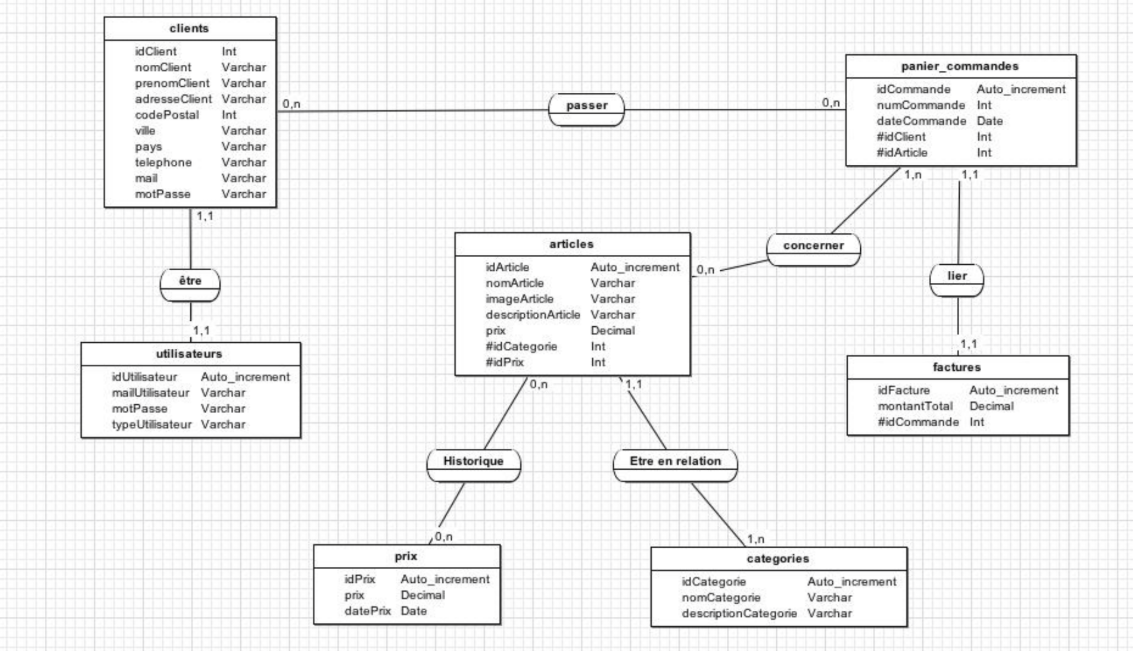


Figure 1 - Modele conceptuel de base

## Planning initial

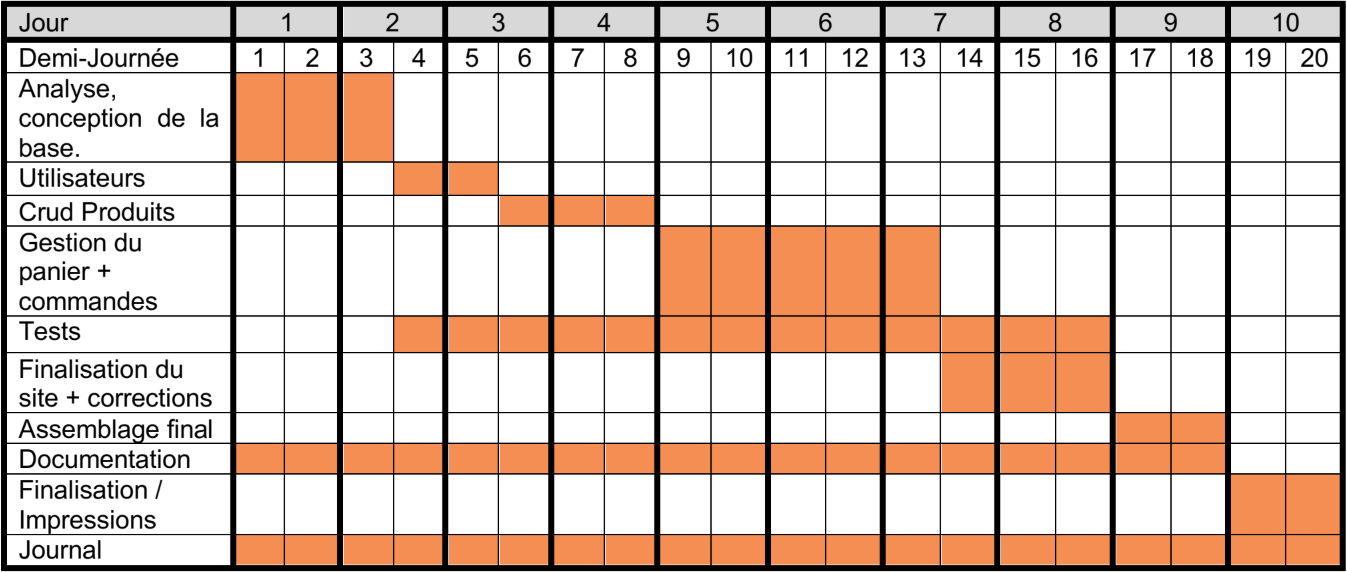


Figure 2 - planning initial

Ceci est le planning initial réalisé avant le début du projet. Le planning final peut être légèrement différent à celui-ci.

## Plan du site

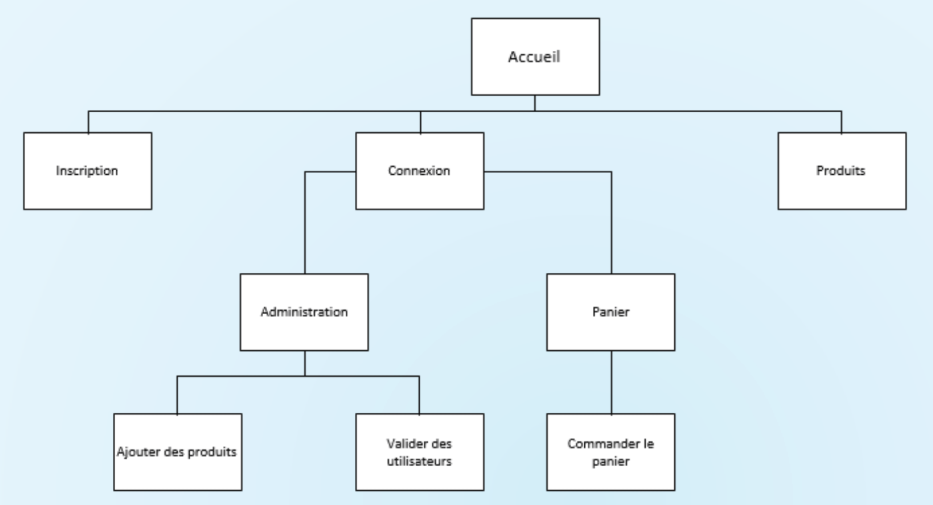


Figure 3 - plan du site

# Analyse fonctionnelle

## Fonctionnalités

### Inscription

Cette fonctionnalité permet à un client de créer un compte. Par la suite le compte sera validé ou non par l’administrateur permettant de se connecter sur le site et pouvoir ajouter des articles dans son panier.

### Connexion

Cette fonctionnalité permet à l’utilisateur de se connecter sur le site pour pouvoir ensuite ajouter des articles dans son panier.

### Validations utilisateurs

Cette fonctionnalité permet à l’administrateur de valider des utilisateurs qui se sont inscrits précédemment depuis le site.

### Ajout de produits

Cette fonctionnalité permet à l’administrateur d’ajouter des articles depuis l’administration du site web.

### Modification de produits

Cette fonctionnalité permet à l’administrateur de modifier un article.

### Suppression de produits

Cette fonctionnalité permet à l’administrateur de supprimer un article.

### Commande

Cette fonctionnalité permet à l’utilisateur de passer la commande des articles qui se trouvent dans son panier.

### Historisation des prix

Cette fonctionnalité évite à un article déjà présent dans notre panier, soit affecté par un changement de son prix.

## Interfaces

### Utilisateurs non-authentifiés

#### Accueil

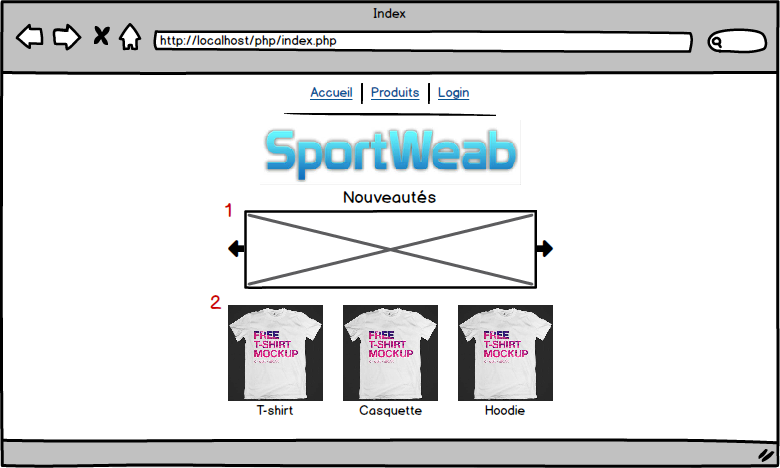


Figure 4 - page d’accueil du site web

1. Un slider qui affiche les tous les produits.
2. On peut apercevoir les trois catégories d’articles différents.

#### Connexion

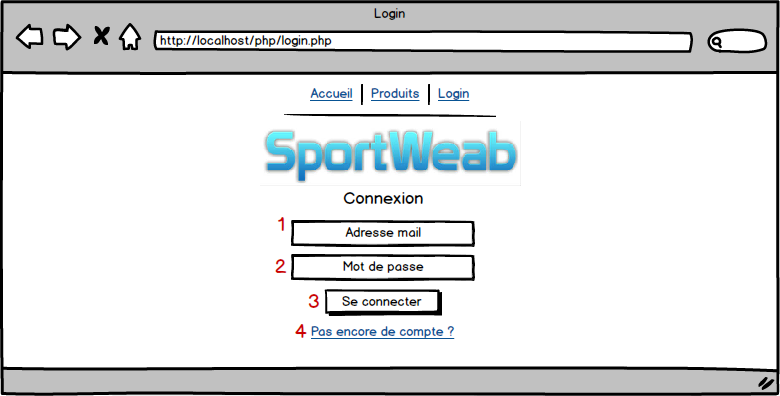


Figure 5 - Page de connexion

1. Champ pour l’email de connexion.
2. Champ pour le mot de passe lié au compte.
3. Bouton de connexion.
4. Lien qui redirige sur la page d’inscription. Voir **figure 3**.

#### Inscription

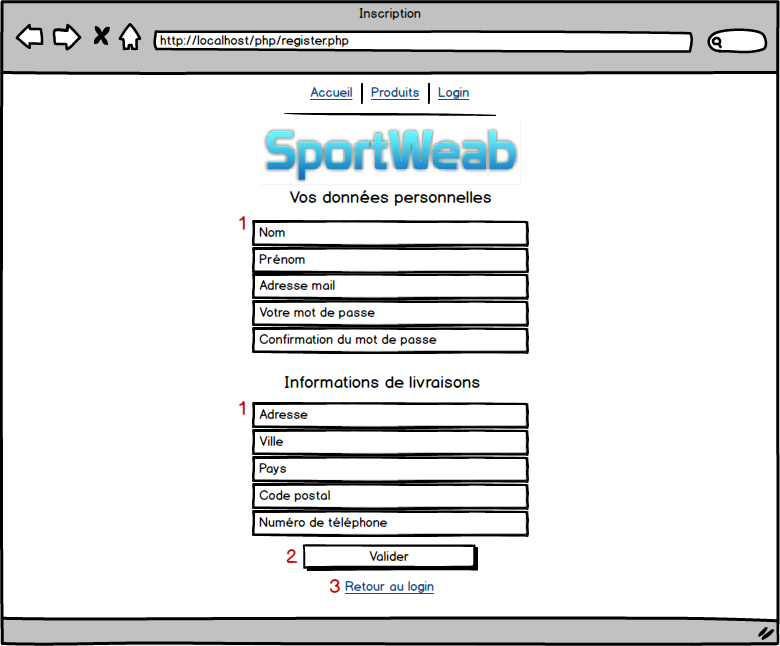


Figure 6 - Page d’inscription

1. Champs pour l’inscription.
2. Bouton d’inscription.
3. Lien qui redirige sur la page de connexion. Voir **figure 2**.

#### Produits



Figure 7 - PAGE DES PRODUITS

1. Lien permettant de trier les articles par prix ou par catégorie.
2. Les articles affichés triés par prix ou par catégorie. Si on clique sur un des articles, on est redirigé sur la page article qui affiche les informations du produit. Voir **figure 5**.

#### Page article

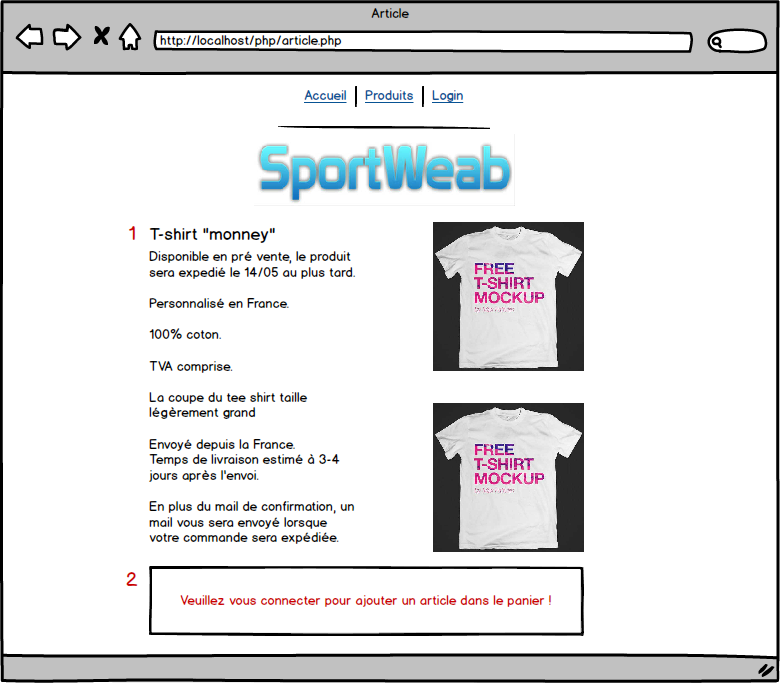


Figure 8 - Page article

1. Informations du produit.
2. Message indiquant qu’il faut être connecté pour pouvoir ajouter des articles dans son panier.

### Utilisateurs authentifiés

#### Article

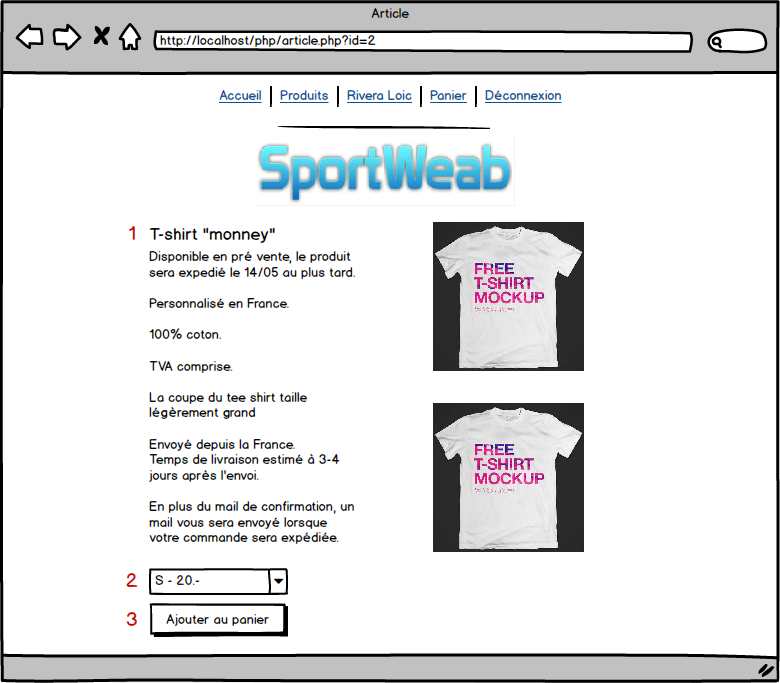


Figure 9 - Page de l’article

Quand on clique sur un article qui se trouve sur la page produit, la page article affiche les informations de l’article grâce à l’identifiant de l’article passé en paramètre.

1. Informations de l’article.
2. Choix de la taille de l’article.
3. Bouton qui permet d’ajouter l’article dans le panier.

#### Panier

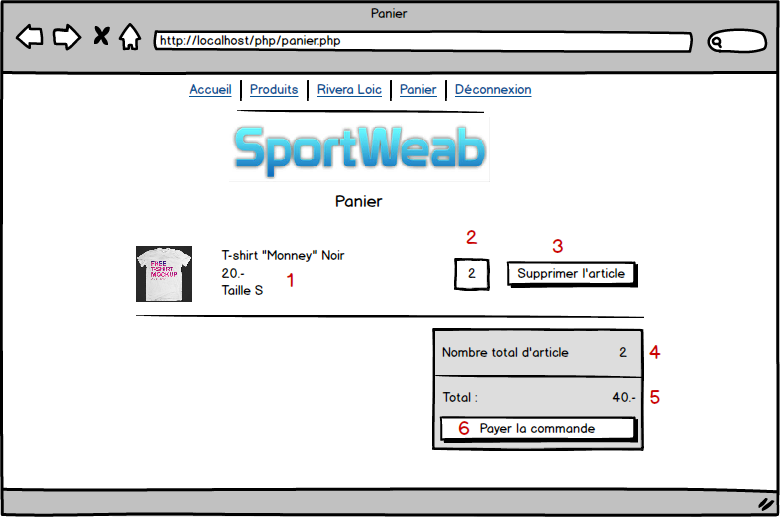


Figure 10 – panier

1. Information du produit.
2. Quantité du produit dans le panier.
3. Bouton permettant de supprimer l’article du panier.
4. Information concernant le nombre d’articles présent dans le panier.
5. Total du prix des articles présents dans le panier.
6. Bouton permettant de procéder à la page de paiement. Voir **figure 7**.

#### Page de paiement

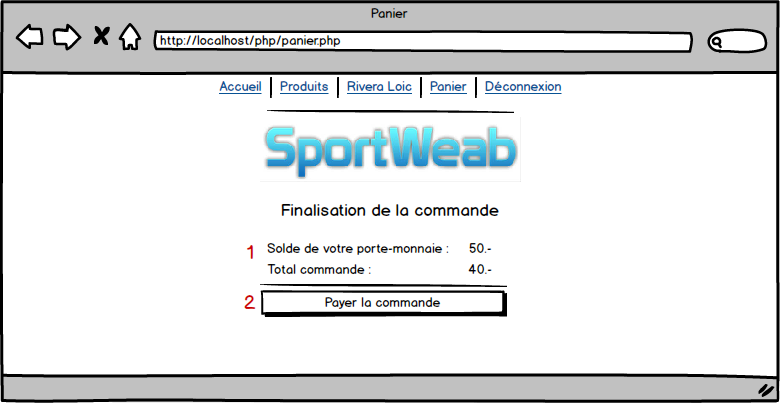
****

Figure 11 - Page de paiement

1. Information du solde du porte-monnaie de l’utilisateur et le prix total de la commande.
2. Bouton pour valider le paiement.

### Administrateur

#### Administration



Figure 12 - page d'administration

1. Bouton pour accéder à la page d’ajout des produits.
2. Bouton pour accéder à la page de validation des utilisateurs.

#### Validations d’utilisateurs

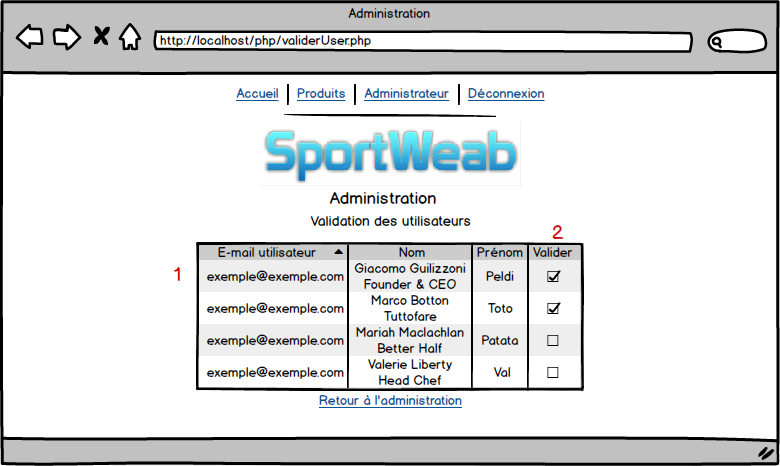


Figure 13 - page de validation des utilisateurs

1. Affichage des utilisateurs pas encore validé et leurs informations.
2. Bouton qui permet de valider les utilisateurs.

#### Page produits

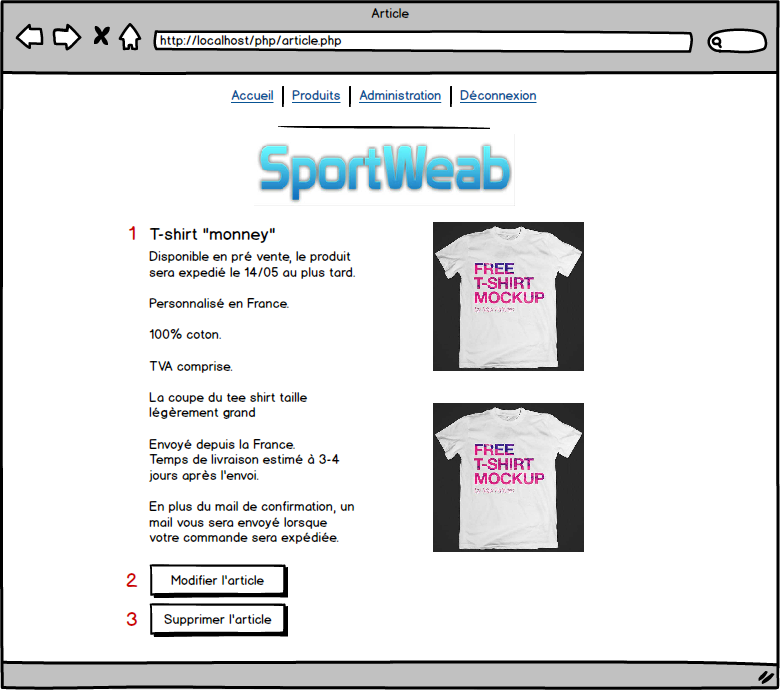


Figure 14 - Page produits administrateur

1. Description de l’article.
2. Bouton permettant de modifier un article et qui redirige sur la page « **modificationProduit.php** » voir figure 10.
3. Bouton permettant de supprimer un article.

#### Ajout de produits

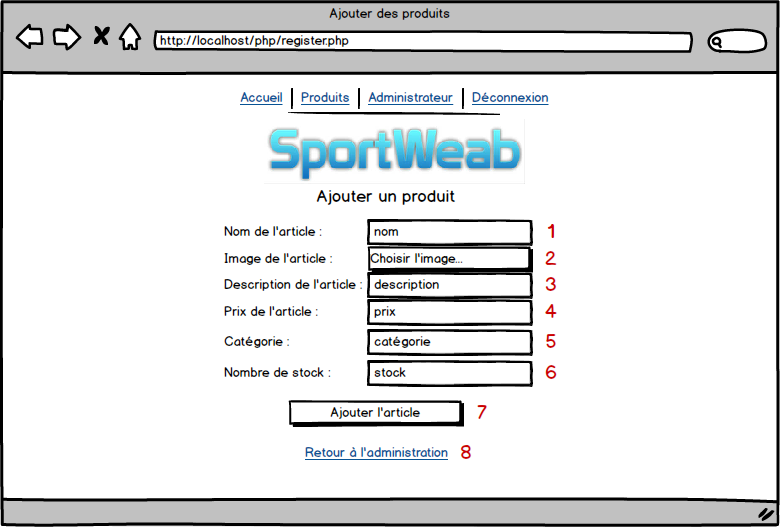


Figure 15 - PAGE D'AJOUT D'ARTICLE

1. Champ pour le nom de l’article.
2. Bouton permettant de sélectionner une image.
3. Champ pour la description de l’article.
4. Champ pour le prix de l’article.
5. Champ pour la catégorie de l’article.
6. Champ pour le nombre de stock.
7. Bouton permettant d’ajouter un produit.
8. Lien permettant de retourner sur la page de l’administration.

#### Modification d’un produit

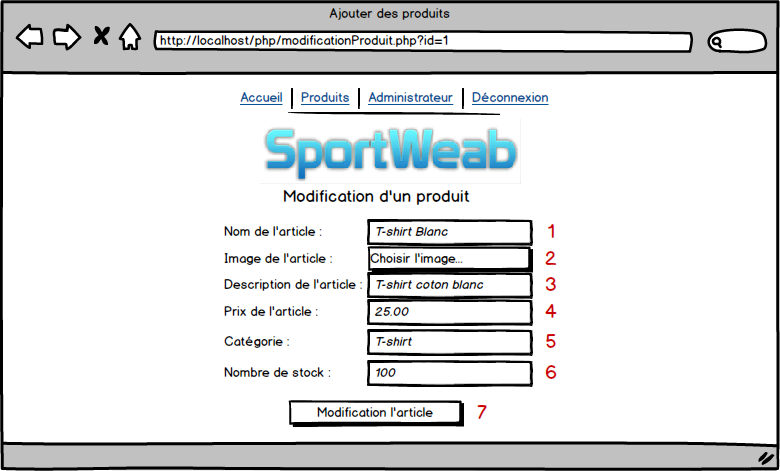


Figure 16 - Modification d'un produit

Les informations sont déjà pré-remplis par les informations de l’article grâce à l’identifiant passé en paramètre.

1. Champ pour le nom de l’article.
2. Bouton permettant de sélectionner une image.
3. Champ pour la description de l’article.
4. Champ pour le prix de l’article.
5. Champs pour la catégorie de l’article.
6. Champ pour le nombre de stock.
7. Bouton permettant de modifier le produit.

#### Suppression d’un produit

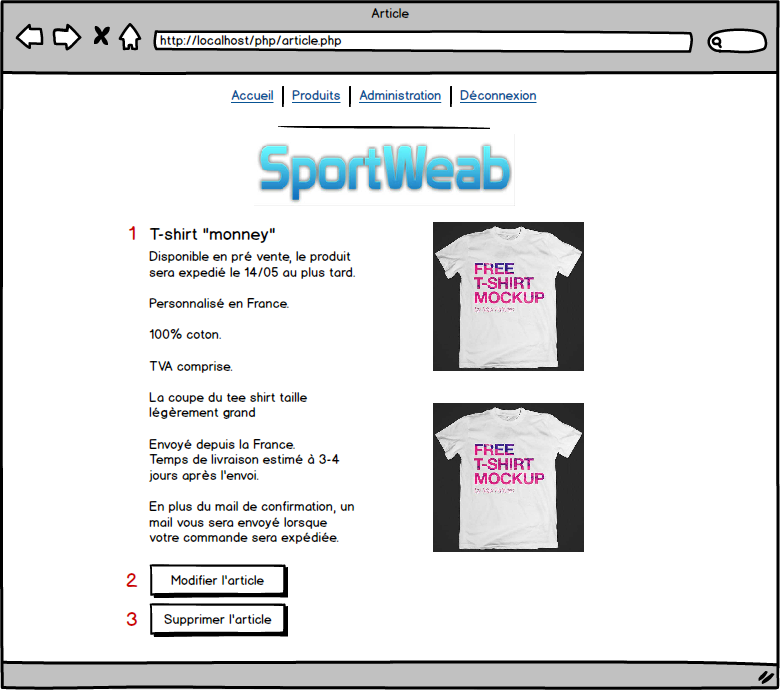


Figure 17 - page article administrateur

1. Decsription de l’article.
2. Bouton pour modifier le produit
3. Supprimer l’article.

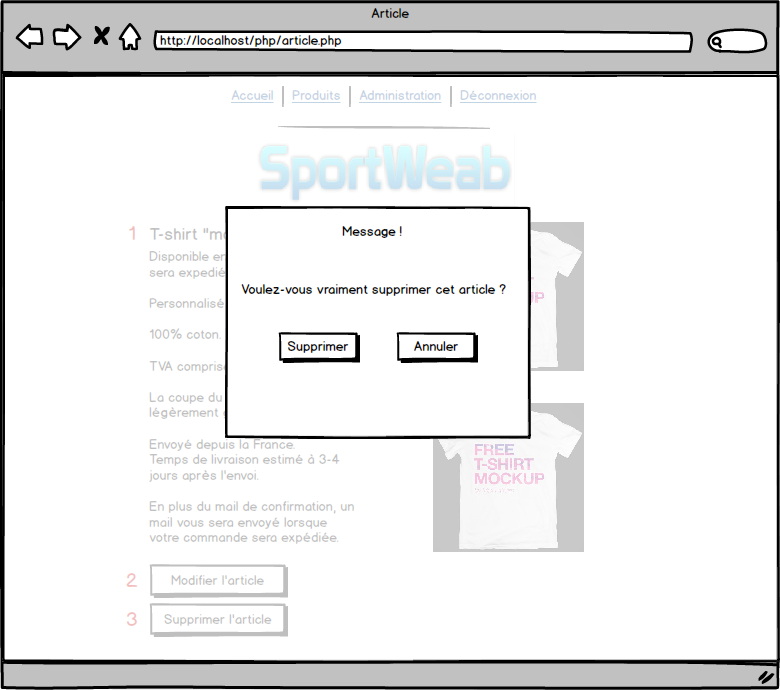


Figure 18 - modal de confirmation de suppression d'article

Ensuite une page modal apparaît pour confirmer la suppression de l’article.

# Analyse organique

## Diagramme de la base de données (MLD)

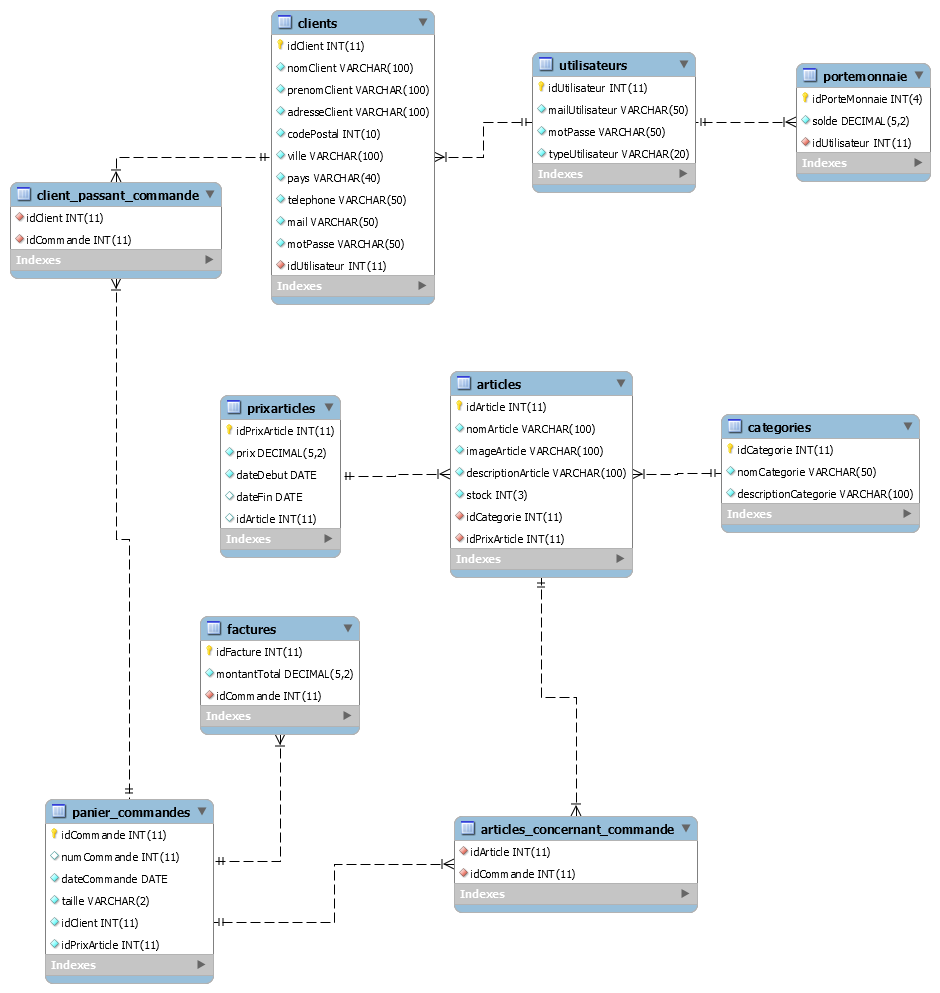


Figure 19 - Modele LOGIQUE de la base de donnees

Voici mon diagramme de la base de données, j’ai modifié la table car j’ai rencontré des problèmes lors de l’historisation des prix. J’ai donc décidé de supprimé la table « prix » et « prix\_historiques ». J’ai créé une nouvelle table « prixArticles ». La table prixArticles(**idPrixArticle**, prix, dateDebut, dateFin, #idArticle). J’ai ajouté une clé étrangère **#idPrixArticle** dans la table « articles ». J’ai ensuite supprimé la clé étrangère **#idClient** et **#idArticle** dans la table « panier\_commandes » car elles sont déjà lié dans les tables « articles\_concernant\_commandes » et « client\_passant\_commande ». Ensuite j’ai ajouté une clé étrangère **#idPrixArticle** pour savoir quel id a le prix quand il a été ajouté dans la base de données.

## Table articles

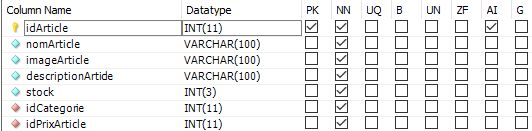


Figure 20 - Table articles

## Table articles\_concernant\_commande



Figure 21 - Table articles\_concernant\_commande

## Table categories

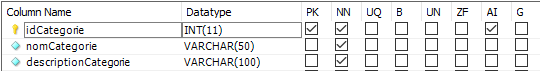


Figure 22 - Table categories

## Table client\_passant\_commande



Figure 23 - tABLE CLIENT\_PASSANT\_COMMANDE

## Table clients

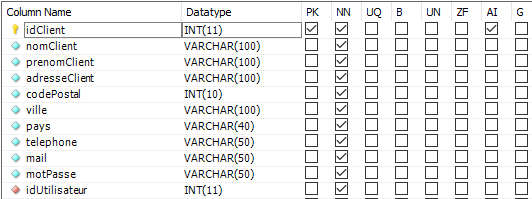


Figure 24 - table clients

## Table factures

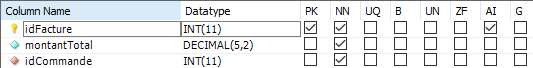


Figure 25 - table factures

## Table panier\_commandes

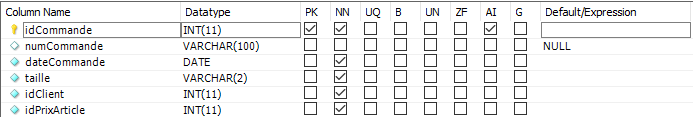


Figure 26 - Table panier\_commandes

## Table portemonnaie

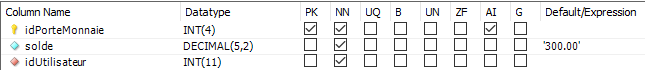


Figure 27 - table portemonnaie

## Table prixarticles

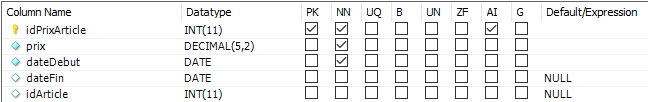


Figure 28 - table prixarticles

## Tables utilisateurs

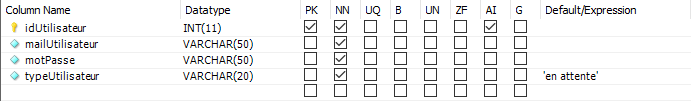


Figure 29 - table utilisateurs

## Plan du site final



Figure 30 - plan du site

## Fonctions php

### checkNameLastName($var)

Cette fonction permet de vérifier que la variable passée en paramètre contient que des lettres et seulement le tiret pour les prénoms et nom composé. Elle utilise une expression régulière.

|  |
| --- |
| **function** checkNameLastName($var) {  $pattern = **'/^[a-zA-Z]+(([\',. -][a-zA-Z ])?[a-zA-Z]\*)\*$/m'**;  **if** (*preg\_match\_all*($pattern, $var, $matches, ***PREG\_SET\_ORDER***, 0)) {  **return true**;  } **else** {  **return false**;  } } |

### OnlyNumbers($var)

Cette fonction permet de vérifier que la variable passée en paramètre contient que des chiffres. J’utilise la fonction php « **ctype\_digit()** » pour vérifier si la variable passée en paramètre contient bien que des chiffres.

|  |
| --- |
| **function** OnlyNumbers($var) {  **if** (*ctype\_digit*($var)) {  **return true**;  } **else** {  **return false**;  } } |

### MinPwd($var)

Cette fonction permet de vérifier que la variable contient plus que 4 caractères. J’utilise la fonction php « **strlen()** » pour vérifier que la variable passée en paramètre contient bien plus de quatre caractères.

|  |
| --- |
| **function** MinPwd($var) {  **if** (*strlen*($var) > 4) {  **return true**;  } **else** {  **return false**;  } } |

### Menu()

Cette fonction permet au menu d’être dynamique. On affiche l’onglet « Accueil », « Produits » et « Connexion » quand l’utilisateur n’est pas connecté.

Si l’utilisateur est connecté et que l’utilisateur est un administrateur, l’onglet « Accueil », « Produits », « Administration » et « Déconnexion » est affiché.

Si l’utilisateur est connecté et qu’il n’est pas administrateur, l’onglet « Accueil », « Produits », « Panier » et « Déconnexion » est affiché.

### getCategorie()

Permet de sélectionner les catégories.

Voici la requête :

|  |
| --- |
| **SELECT** *\** **FROM utilisateur;** |

### getArticleByCategories($categorie)

Cette fonction permet de récupérer les articles par catégories. On lui passe en paramètre la categorie.

Voici la requête :

|  |
| --- |
| **SELECT articles.idArticle, nomArticle, prix, nomCategorie FROM articles, categories, prixarticles WHERE articles.idCategorie = categories.idCategorie AND articles.idPrixArticle = prixarticles.idPrixArticle AND nomCategorie = :categorie;** |

### getArticleInfo($id)

Cette fonction permet de récupérer les informations concernant un article. On lui passe en paramètre l’identifiant de l’article.

Voici la requête :

|  |
| --- |
| **SELECT DISTINCT articles.idArticle, articles.nomArticle, articles.imageArticle, articles.descriptionArticle, articles.stock, categories.nomCategorie, articles.idPrixArticle, prixarticles.idPrixArticle, prixarticles.prix,** *MAX***(dateDebut), dateFin FROM articles, categories INNER JOIN prixarticles WHERE articles.idPrixArticle = prixarticles.idPrixArticle AND articles.idCategorie = categories.idCategorie AND prixarticles.idArticle = :idArticle ORDER BY prixarticles.dateDebut ASC;** |

### login($login,$pwd)

Cette fonction permet de faire à un utilisateur de se connecter. La requête recherche dans la table utilisateurs si un utilisateur existe avec le mail et le mot de passe qu’on lui passe en paramètre.

Voici la requête :

|  |
| --- |
| **SELECT** *\** **FROM utilisateurs WHERE mailUtilisateur = :nom AND motPasse = :mdp;** |

### RegisterClient ($nom, $prenom, $adresse, $codePostal, $ville, $pays, $telephone, $mail, $motPasse, $idUser)

Cette fonction permet l’insertion d’un nouveau client.

Voici la requête :

|  |
| --- |
| **INSERT INTO `clients` (`nom`, `prenom`, `adresse`, `codePostal`, `ville`, `telephone`, `mail`, `motPasse`) VALUES (:nom, :prenom, :adresse, :codePostal, :ville, :telephone, :mail, :motPasse)** |

### user\_exists($mail)

Cette fonction permet de séléctionner un compte utilisateur. Elle sert à vérifier si un utilisateur qui tente de s’inscrire utilise une adresse mail déjà utilisée par un autre utilisateur.

Voici la requête :

|  |
| --- |
| **SELECT** *\** **FROM utilisateurs WHERE mailUtilisateur = :email;** |

### registerUser($mail, $password1)

Cette fonction permet de créer un nouvel utilisateur lors de son inscription. Il faut lui donner en paramètre le mail de l’utilisateur ainsi que son mot de passe.

Voici la requête :

|  |
| --- |
| **INSERT INTO `utilisateurs` (`mailUtilisateur`, `motPasse`) VALUES (:mail, :motPasse);** |

### createWallet($idUser)

Cette fonction permet de créer un porte-monnaie virtuel pour un utilisateur. Je passe en paramètre l’identifiant de l’utilisateur. Une fois qu’il sera ajouté, son porte-monnaie sera créé avec comme solde 300CHF par défaut.

Voici la requête :

|  |
| --- |
| **INSERT INTO `portemonnaie` (`idUtilisateur`) VALUES (:idUtilisateur);** |

### getUserInformation($email)

Cette fonction permet de récupérer les données d’un utilisateur. Il faut lui donner en paramètre l’email de l’utilisateur.

Voici la requête :

|  |
| --- |
| **SELECT** *\** **FROM clients, utilisateurs WHERE clients.mail = :mail AND utilisateurs.mailUtilisateur = :mail;** |

### getNotValidateUsers

Cette fonction permet de sélectionner tous les utilisateurs qui ne sont pas encore validé.

Voici la requête :

|  |
| --- |
| **SELECT** *\** **FROM utilisateurs WHERE typeUtilisateur = 'en attente';** |

### validateUser($idUser)

Cette fonction permet de valider des utilisateurs qui sont en attente de validation. Il faut donner en paramètre l’identifiant de l’utilisateur a validé.

Voici ma requête :

|  |
| --- |
| **UPDATE utilisateurs SET typeUtilisateur = 'Utilisateur' WHERE idUtilisateur = :id;** |

### addPrice($prixArticle, $dateFin, $idArticle)

Cette fonction permet d’ajouter un prix pour un article. Il faut lui donner en paramètre le prix, la date de fin du prix et l’identifiant de l’article. La date du début du prix est par défaut à la date de l’ajout du nouveau prix.

Voici ma requête :

|  |
| --- |
| **INSERT INTO `prixarticles` (`prix`, `dateDebut`, `dateFin`, `idArticle`) VALUES (:prix,** *CURRENT\_DATE***(), :dateFin, :idArticle);** |

### updatePrice($idArticle, $idPrixArticle)

Cette fonction permet de modifier le prix d’un article. Il faut lui donner en paramètre l’identifiant de l’article et l’identifiant du prix.

Voici ma requête :

|  |
| --- |
| **UPDATE prixarticles SET dateFin =** *current\_date***() WHERE dateDebut = (SELECT MAX(dateDebut) WHERE idArticle = :idArticle AND idPrixArticle = :idPrixArticle);** |

### updatePriceArticle($idArticle, $idPrixArticle)

Cette fonction permet d’ajouter un article au nouveau prix ajouté.

Voici ma requête :

|  |
| --- |
| **UPDATE prixarticles SET idArticle = :idArticle WHERE idPrixArticle = :idPrixArticle;** |

### addArticle($nomArticle, $imageArticle, $descriptionArticle, $stock, $idCategorie, $idPrixArticle)

Cette fonction permet d’ajouter un nouvel article.

Voici ma requête :

|  |
| --- |
| **INSERT INTO `articles` (`nomArticle`, `imageArticle`, `descriptionArticle`, `stock`, `idCategorie`, `idPrixArticle`) VALUES (:nomArticle, :imageArticle, :descriptionArticle, :stock, :idCategorie, :idPrixArticle);** |

### updateArticle($nomArticle,$imageArticle,$descriptionArticle,$stock, $idCategorie, $idPrixArticle, $idArticle)

Cette fonction permet de modifier un article en fonction de l’identifiant de l’article.

Voici ma requête :

|  |
| --- |
| **UPDATE articles SET nomArticle = :nomArticle, imageArticle = :imageArticle, descriptionArticle = :descriptionArticle, stock = :stock, idCategorie = :idCategorie, idPrixArticle = :idPrixArticle WHERE idArticle = :idArticle;** |

### deleteArticle($idArticle)

Cette fonction permet de supprimer un article en fonction de l’identifiant de l’article.

Voici ma requête :

|  |
| --- |
| **DELETE FROM articles WHERE articles.idArticle = :idArticle;** |

### checkoutCart($numCommande, $idClient)

Cette fonction permet de commander un panier en ajoutant un numéro de commande. Le numéro de commande est composé de la date actuelle + un identifiant unique généré en php.

Voici ma requête :

|  |
| --- |
| **UPDATE panier\_commandes SET numCommande = :numCommande WHERE idClient = :idClient AND numCommande IS NULL;** |

### addArticleToCart($taille, $idClient, $idArticle, $idPrixArticle)

Cette fonction permet d’ajouter un article dans son panier. Entre temps quand on ajoute un article dans son panier il faut débiter le stock de l’article.

Dans un premier temps on débite le stock de l’article ajouté au panier.

|  |
| --- |
| **UPDATE articles SET stock = stock - 1 WHERE idArticle = :idArticle AND stock > 0 ;** |

Ensuite il faut ajouter l’article dans le panier de l’utilisateur.

|  |
| --- |
| **INSERT INTO `panier\_commandes` (`dateCommande`, `taille`, `idClient`, `idPrixArticle`) VALUES (***CURRENT\_DATE***(), :taille, :idClient, :idPrixArticle) ;** |

### updateStock($idArticle)

Cette fonction permet de recalculer le stock si un utilisateur supprime un article de son panier.

Voici ma requête :

|  |
| --- |
| **UPDATE articles SET stock = stock + 1 WHERE idArticle = :idArticle AND stock >= 0 ;** |

### getCart($idClient)

Cette fonction permet de récupérer le panier d’un client.

Voici ma requête :

|  |
| --- |
| **SELECT distinct panier\_commandes.idCommande, articles.idArticle, articles.nomArticle, articles.imageArticle , prixarticles.prix, panier\_commandes.taille, prixarticles.dateDebut, prixarticles.dateFin, panier\_commandes.dateCommande,panier\_commandes.idClient FROM panier\_commandes, articles, client\_passant\_commande, clients, prixarticles, articles\_concernant\_commande WHERE panier\_commandes.idCommande = articles\_concernant\_commande.idCommande AND articles\_concernant\_commande.idArticle = articles.idArticle AND panier\_commandes.idCommande = client\_passant\_commande.idCommande AND panier\_commandes.idPrixArticle = prixarticles.idPrixArticle AND client\_passant\_commande.idClient = clients.idClient AND articles.idArticle = prixarticles.idArticle AND prixarticles.dateDebut <= panier\_commandes.dateCommande AND (prixarticles.dateFin >= panier\_commandes.dateCommande) AND numCommande IS NULL AND client\_passant\_commande.idClient = :idClient;** |

### deleteArticleFromCart($idClient, $idCommande)

Cette fonction supprime un article dans le panier d’un utilisateur.

Voici ma requête :

|  |
| --- |
| **DELETE FROM panier\_commandes WHERE panier\_commandes.idClient = :idClient AND panier\_commandes.idCommande = :idCommande;** |

### getWallet($idUser)

Cette fonction sélectionne un porte-monnaie d’un utilisateur. Il faut donner en paramètre l’identifiant de l’utilisateur.

Voici ma requête :

|  |
| --- |
| **SELECT** *\** **FROM portemonnaie WHERE idUtilisateur = :idUtilisateur;** |

### updateWallet($idUtilisateur, $solde)

Cette fonction permet de modifier le solde d’un utilisateur après une commande. Il faut donner en paramètre l’identifiant de l’utilisateur.

Voici ma requête :

|  |
| --- |
| **UPDATE portemonnaie SET solde = :solde WHERE idUtilisateur = :idUtilisateur;** |

### createBills($montant, $allIdCommand)

Cette fonction permet de créer une facture pour une commande. Il insère le nombre d’identifiant de commande autant de fois que le nombre d’article qu’on a commandé. $allIdCommand contient un tableau d’identifiant de commande. Je fais une boucle **foreach** pour insérer ensuite comme valeur pour la requête.

Code :

|  |
| --- |
| $values = **null**; **foreach** ($allIdCommand **as** $key => $idCommand) {  $values .= **'(:montant,'**.$idCommand[**'idCommande'**].**' ),'**; } $values = *substr*($values,0,-1); |

Voici ma requête :

|  |
| --- |
| **INSERT INTO `factures` (`montantTotal`, `idCommande`) VALUES** $values**"**); |

### getAllCommands($idUser)

Cette fonction permet de récupérer toutes les commandes passées par un utilisateur.

Voici ma requête :

|  |
| --- |
| **SELECT DISTINCT** *\** **FROM panier\_commandes WHERE idClient = :idUtilisateur AND numCommande IS NOT NULL GROUP BY numCommande;** |

### getDetailsCommand($idClient,$numCommand)

Cette fonction permet de récupérer les détails d’une commande pour un utilisateur.

Voici ma requête :

|  |
| --- |
| **SELECT distinct articles.idArticle, articles.nomArticle, imageArticle,panier\_commandes.taille, prixarticles.prix FROM panier\_commandes, prixarticles, articles, articles\_concernant\_commande , factures WHERE panier\_commandes.idPrixArticle = prixarticles.idPrixArticle AND panier\_commandes.idCommande = articles\_concernant\_commande.idCommande AND articles\_concernant\_commande.idArticle = articles.idArticle AND panier\_commandes.idCommande = factures.idCommande AND panier\_commandes.idClient = :idUtilisateur AND panier\_commandes.numCommande = :numCommande AND panier\_commandes.numCommande IS NOT NULL;** |

### getArticles($categorie,$order,$way)

Cette fonction permet de construire une requête en fonction de ce que la fonction reçoit en fonction. Elle est utilisée pour l’affichage des articles et elle gère les filtres pour trier les articles. La variable $categorie contient la catégorie, la variable $order contient le champ à trier et $way contient l’ordre pour le tri.

Dans « $query » on a une requête. On a ensuite des tests pour la catégorie, pour trier les colonnes et l’ordre de tri.

|  |
| --- |
| $query = **"SELECT DISTINCT articles.idArticle, nomArticle, imageArticle, descriptionArticle,stock,nomCategorie,** *max***(prixarticles.dateDebut), prixarticles.prix FROM articles, categories, prixarticles WHERE articles.idArticle = prixarticles.idArticle AND articles.idCategorie = categories.idCategorie AND articles.idPrixArticle = prixarticles.idPrixArticle"**; **switch** ($categorie){  **case "Hoodie"** : $query .= **" AND nomCategorie = \"hoodie\" GROUP BY articles.idArticle ORDER BY "**;  **break**;  **case "Casquette"** : $query .= **" AND nomCategorie = \"casquette\" GROUP BY articles.idArticle ORDER BY "**;  **break**;  **case "T-shirt"** : $query .= **" AND nomCategorie = \"T-shirt\" GROUP BY articles.idArticle ORDER BY "**;  **break**;  **default** : $query .= **" GROUP BY articles.idArticle ORDER BY "**; } **switch** ($order){  **case "prix"** : $query .= **" prix "**;  **break**;  **case "nomCategorie"** : $query .= **" nomCategorie "**;  **break**;  **default** : $query .= **" articles.idArticle DESC "**; } **switch** ($way){  **case "ASC"** : $query .= **" ASC;"**;  **break**;  **case "DESC"** : $query .= **" DESC;"**;  **break**;  **default** :$query .= **";"**; } |

# Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **N° de test** | **A tester** | **Comment ?** | **Résultat** |
| **Connexion** | **T1** | Vérifier que l'on ne peut pas se connecter avec un compte inexistant | Connexion avec un compte jamais inscrit | Ok |
| **T2** | Vérifier que l'on ne peut pas se connecter avec un compte en attente de validation par l'utilisateur | Connexion avec un compte pas encore validé | Ok |
| **T3** | Vérifier que l'on ne peut pas se connecter sans avoir rentré le captcha | Ne rien mettre dans le captcha | Ok |
| **Inscription** | **T4** | Ne pas pouvoir s'inscrire en laissant des champs vides | Laisser des champs vides | Ok |
| **T5** | Ne pas pouvoir s'inscrire avec des chiffres dans le nom et le prénom | Mettre des chiffres dans le nom et prénom | Ok |
| **T6** | Ne pas pouvoir s'inscrire avec un email déjà existant dans la base | S'inscrire avec un email qui existe déjà | Ok |
| **T7** | Ne pas pouvoir s'inscrire sans avoir remplis le captcha | Ne pas remplir le captcha | Ok |
| **Insertion produit** | **T8** | Ne pas pouvoir ajouter un article en laissant des champs vides | Ne pas mettre d'image | Ok |
| **T9** | Ne pas pouvoir upload autre chose que des images | Sélectionner un fichier .docx à la place d'une image | Ok |
| **T10** | Ne pas pouvoir ajouter un article en mettant une valeur négative pour le stock | Mettre comme valeur -1 pour le stock | Ok |
| **T11** | Ne pas pouvoir ajouter un article en mettant une valeur plus grande que 150 pour le prix | Mettre 160 comme valeur pour le prix | Ok |
| **Modification produits** | **T12** | Ne pas pouvoir modifier un article si on laisse un champs vide autre que le champs de l'image | Ne pas mettre de prix à l'article | Ok |
| **T13** | Si l'on ne met pas d'image quand on modifie un article, on garde l'image déjà ajoutée | Ne pas sélectionner une autre image | Ok |
| **T14** | Si on ajoute une nouvelle image, il faut remplacer l'ancienne image par la nouvelle | Sélectionner une nouvelle image | Ok |
| **Suppression d'article** | **T15** | Pouvoir supprimer un article | Supprimer l'article depuis la page article en tant qu'administrateur | Ok |
| **Panier** | **T16** | Pouvoir ajouter un article dans son panier | Choisir un article et l'ajouter dans le panier et vérifier que l'article se trouve bien dans le panier | Ok |
| **T17** | Le stock est débité quand un article est ajouté au panier | Ajouter un article dans le panier et vérifier que sur la page de l'article que son stock a diminué de 1 | Ok |
| **T18** | Pouvoir supprimer un article de son panier | Supprimer un article du panier | Ok |
| **T19** | Le stock est recalculé quand on supprime l'article de son panier | Supprimer un article du panier et vérifier que le stock a été recalculé | Ok |
| **T20** | Quand on a ajouté un article dont le stock est égal à 1, un message affiche sur la page de l'article qu'il est en rupture de stock | Ajouter un article dans le panier dont le stock est égal à un | Ok |
| **Panier** | **T21** | Si on clique sur "Payer la commande" une fenêtre modal doit s'afficher pour confirmer le paiement | Cliquer sur le bouton "Payer la commande" | Ok |
| **T22** | Ne pas pouvoir commander les articles si l'on ne possède pas le solde du porte-monnaie suffisant pour payer la commande, un message d'erreur s'affiche | Ajouter des articles pour dépasser 300 CHF de commande et procéder au paiement | Ok |
| **T23** | Le bouton payer la commande est désactiver si le solde du porte-monnaie est insuffisant | Ajouter des articles pour que le total de la commande dépasse le solde du porte-monnaie | Ok |
| **T24** | Le prix d'un article ajouté dans le panier n'est pas affecté par le changement de son prix par la suite | Ajouter un article dans le panier, changé le prix de l'article avec l'administrateur et voir si le prix de l'article dans le panier a changé | Ok |
| **Commande** | **T25** | Le solde du porte-monnaie est mis à jour après une commande | Aller sur la page user.php et vérifier que le solde du porte-monnaie a été mis à jour | Ok |
| **T26** | La commande passée est ajoutée à l'historique des commandes et est accessible sur la page user.php | Aller sur la page user.php et vérifier que la commande à bien été ajoutée à l'historique de commande | Ok |
| **T27** | Vérifier que la commande qu'on vient d'effectuer contient bien les articles qu'on a commandé | Aller sur la page user.php et vérifier que la dernière commande contient tous les articles qu'on a ajouté | Ok |
| **Pagination** | **T28** | La page produits.php affiche 12 articles par page | Aller sur la page produits.php et vérifier qu'il y a bien 12 articles par pages et qu'il y a des liens en bas pour naviguer entre les pages de produits | Ok |
| **Filtres** | **T29** | Vérifier que trier les articles par prix trie les prix ascendant et descendant | Cliquer sur le lien "Prix" pour trier les prix ascendant, descendant et vérifier que le prix sont bien trié | Ok |
| **T30** | Les articles sont bien trié par catégories | Cliquer sur le lien "Catégorie" pour trier les articles par catégorie et vérifier que les articles sont bien trié par catégorie | Ok |

# Conclusion

## Planning final

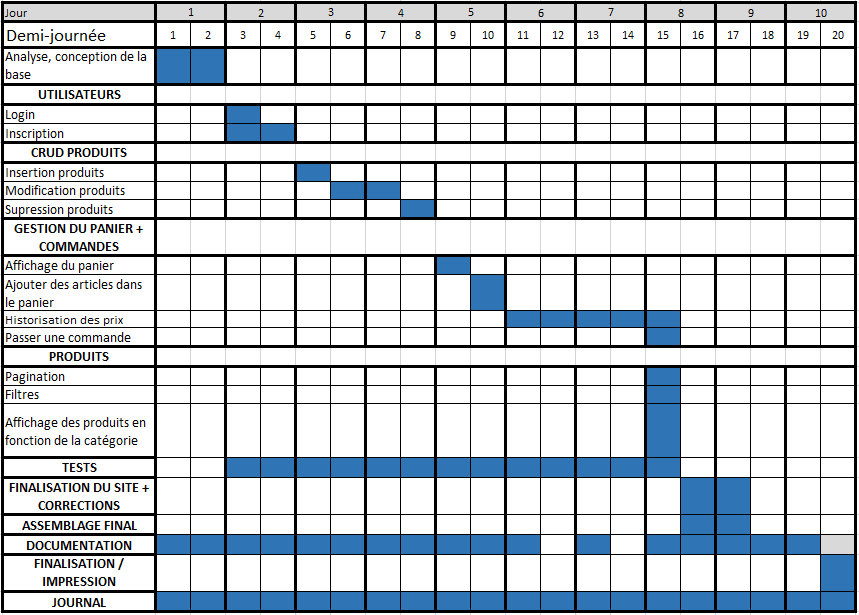


Figure 31 - Planning final

J’ai bien réussi à suivre le planning initial, jusqu’au moment où j’étais bloqué pour l’historisation des prix. Je devais finir le panier et les commande le 7ème jour mais cela m’a pris beaucoup plus de temps. J’ai fini la gestion de commandes et l’historisation le 8ème jour, car M. Bonvin m’a aiguillé sur une possible solution pour régler le problème de l’historisation des prix. J’ai par la suite rattrapé mon retard le mardi 22.05. Sinon en globalité j’ai bien été dans les temps.

## Difficulté rencontrée

La partie la plus compliquée à implémenter pour moi était, l’historisation des prix. J’ai prix énormément de temps pour faire la requête jusqu’à que je change un peu la base de données car il n’était pas possible d’afficher le prix avec sa date. J’ai réussi par la suite à rattraper le retard en changeant la base de données et en modifiant ma requête.

## Mon ressenti

Je suis satisfait du travail que j’ai produit. J’étais vraiment motivé pour réaliser ce travail, malgré que j’ai rencontré quelques problèmes qui m’ont beaucoup ralenti dans mon développement. Je trouve avoir bien résumé ces 4 années au sein du CFPT dans ce travail. Je tiens à remercier M. Bonvin pour avoir accepté de me suivre pour mon projet et de m’avoir aiguillé pour mon projet.

## Améliorations possibles

Il y a plusieurs choses qui pourraient être améliorées dans mon projet :

* Avoir une interface pour que l’administrateur puisse modifier le solde des utilisateurs.
* Que l’utilisateur puisse modifier ses données depuis le site.
* Avoir des messages de confirmations lors de la commande d’un panier.
* Pouvoir ajouter plus qu’un article lors de l’ajout d’un article dans le panier.

# Bibliographies

## Site utilisés

* <https://stackoverflow.com/>
* <http://php.net/>
* <http://getbootstrap.com/>
* <https://github.com/>
* <https://www.w3schools.com/>
* <https://openclassrooms.com/>

# Table des illustrations

[Figure 1 - Modele conceptuel de base 6](#_Toc514941933)

[Figure 2 - planning initial 7](#_Toc514941934)

[Figure 3 - plan du site 7](#_Toc514941935)

[Figure 4 - page d’accueil du site web 9](#_Toc514941936)

[Figure 5 - Page de connexion 10](#_Toc514941937)

[Figure 6 - Page d’inscription 11](#_Toc514941938)

[Figure 7 - PAGE DES PRODUITS 12](#_Toc514941939)

[Figure 8 - Page article 13](#_Toc514941940)

[Figure 9 - Page de l’article 14](#_Toc514941941)

[Figure 10 – panier 15](#_Toc514941942)

[Figure 11 - Page de paiement 16](#_Toc514941943)

[Figure 12 - page d'administration 17](#_Toc514941944)

[Figure 13 - page de validation des utilisateurs 17](#_Toc514941945)

[Figure 14 - Page produits administrateur 18](#_Toc514941946)

[Figure 15 - PAGE D'AJOUT D'ARTICLE 19](#_Toc514941947)

[Figure 16 - Modification d'un produit 20](#_Toc514941948)

[Figure 17 - page article administrateur 21](#_Toc514941949)

[Figure 18 - modal de confirmation de suppression d'article 22](#_Toc514941950)

[Figure 19 - Modele LOGIQUE de la base de donnees 23](#_Toc514941951)

[Figure 20 - Table articles 24](#_Toc514941952)

[Figure 21 - Table articles\_concernant\_commande 24](#_Toc514941953)

[Figure 22 - Table categories 24](#_Toc514941954)

[Figure 23 - tABLE CLIENT\_PASSANT\_COMMANDE 24](#_Toc514941955)

[Figure 24 - table clients 24](#_Toc514941956)

[Figure 25 - table factures 25](#_Toc514941957)

[Figure 26 - Table panier\_commandes 25](#_Toc514941958)

[Figure 27 - table portemonnaie 25](#_Toc514941959)

[Figure 28 - table prixarticles 25](#_Toc514941960)

[Figure 29 - table utilisateurs 25](#_Toc514941961)

[Figure 30 - plan du site 26](#_Toc514941962)

[Figure 31 - Planning final 39](#_Toc514941963)

# Annexes

* Manuel utilisateur
* Code source
* Journal de bord manuscrit