DOSSIER DE PROJET

« VATMINTON »

Titre professionnel

NIVEAU III

Développeur Web et Web Mobile

CHIV Vathana
LA MANU Soissons – Promotion n°3

SOMMAIRE

I. Résumé du projet

Passionné d’informatique et de badminton, mon projet portera sur la vente d’accessoires de badminton en ligne dont son identité sera « Vatminton » caractérisée par le logo d’un badiste en plein smash avec un « C » inversé pour faire référence à la première lettre du mot « compétiteurs».

Mon site sera aussi bien ciblé pour des particuliers, des professionnels que des collectivités. L’objectif sera de le rendre « responsive ». La création du site par un « en-tête» sera composée d’une barre de navigation avec trois catégories (« boutique », « compétition », « entraînement »). J’y intégrerai également un champ de recherche pour les produits ainsi que deux onglets supplémentaires (« mon compte », « mon panier ») accessible après la connexion de l’utilisateur.

La page d’accueil sera animée par l’actualité hebdomadaire, la meilleure offre promotionnelle du moment, le classement des meilleurs joueurs mondiaux.

Je conclurai que je mettrai au profit du site toutes mes connaissances de langages de programmation étudiés en cours de formation de « LA MANU » tels que HTML, CSS, JavaScript, jQuery, PHP,MySQL.

II. Cahier des charges

1) Descriptif du projet :

Il y a encore quelques années, les articles de sport (vêtements, équipements, …) se vendaient par tradition en point de vente physique ou lors d'événements. Aujourd'hui, l'e-commerce prend de plus en plus d'importance sur le marché des produits sportifs. Certains acteurs du retail sportif vont même jusqu'à vendre exclusivement en ligne !

Cette évolution est principalement due aux nouvelles habitudes de consommation des clients, à la multiplication du nombre d'écrans par foyer (ordinateurs, mobiles, tablettes, …), ainsi qu'aux améliorations logistiques.

Mon site web ciblera aussi bien des particuliers, des professionnels que des collectivités.

2) Leurs outils de développement :

- Lignes de commandes (cd, mkdir, mv, …)
- Trello (outils de gestion de projets)
- Git, Github (outils de programmation)
- Atom, Netbeans (éditeurs de texte)

3) Rôle utilisateur & Fonctionnalités associées :

a) Visiteurs

➔ Un utilisateur non enregistré sera considéré comme simple visiteur. Il n'aura donc qu'un droit de consultation sur l'ensemble des données (actualités, articles de la boutique, …). S'il désire faire un achat, il devra se connecter en tant que client.

b) Client

➔ Création d'un compte client :
Un utilisateur est considéré comme client dès lors qu'il dispose d'un compte utilisateur enregistré, ce qui lui permettra d'effectuer ses achats sur le site.

➔ Se connecter à son compte utilisateur :
A l'aide de son adresse mail et de son mot de passe, l'utilisateur pourra se connecter au site en tant que client.

➔ Lire et modifier les informations de son profil :
Un client aura la possibilité de voir les différentes informations qu'il aura saisies dans le formulaire d'inscription et de les modifier à tout moment.

➔ Suppression du compte utilisateur :
Un utilisateur aura la possibilité de supprimer son propre compte.

➔ Déconnexion :
Un utilisateur pourra se déconnecter à tout moment.

c) Administrateur

➔ Administration de la liste des produits :
L'administrateur a la possibilité de créer, de modifier et de supprimer les produits destinés à être mis en vente.

➔ Modification des comptes utilisateurs :
Il pourra à sa guise modifier le rôle de n'importe quel utilisateur sur le site, c'est à dire le faire passer de simple utilisateur à administrateur afin qu'il puisse bénéficier de toutes les fonctionnalités.
Suppression de compte :
Il pourra en outre supprimer n'importe quel compte utilisateur.

III. Liste des compétences

N° Fiche AT	Activités Types	N° Fiche CP	Compétences Professionnelles
1	Développer la partie Front-End d’une Application Web ou Web Mobile en intégrant les recommandations de sécurité	1	Maquetter une application
		2	Réaliser une interface utilisateur web statique et adaptable
		3	Développer une interface utilisateur web dynamique
		4	Réaliser une interface utilisateur avec une solution de gestion de contenu ou E-commerce
2	Développer la partie Back-End d’une Application Web ou Web Mobile en intégrant les recommandations de sécurité	5	Créer une base de données
		6	Développer les composants d’accès aux données
		7	Développer la partie Back-End d’une application web et web mobile
		8	Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou E-commerce

IV. Spécifications techniques

Logiciels

HTML 5	HyperText Markup Language Langage de base pour la création d'un site internet, il sert à structurer sa page web, c'est-à-dire le contenu, les images, …
CSS 3	Cascading Style Sheets Langage permettant de mettre en forme sa page web, c'est à dire de la styliser.
JavaScript	Langage de programmation de scripts très utilisé pour rendre ses pages web interactives.
PHP 7	HyperText PreProcessor Langage de programmation utilisé notamment pour faire la liaison entre l'utilisateur et la base de données.
MySql	Système de gestion de base des données relationnelles (SGBDR). Permet la réalisation de requêtes pour dialoguer avec la base de données.

Frameworks

Bootstrap 4	Framework CSS permettant de faciliter le développement de sites et d'applications réactives et responsives (adaptables à tout moment).
JQuery	Framework JavaScript permettant de rendre son site dynamique, gérer des évènements, animations, …

Outils

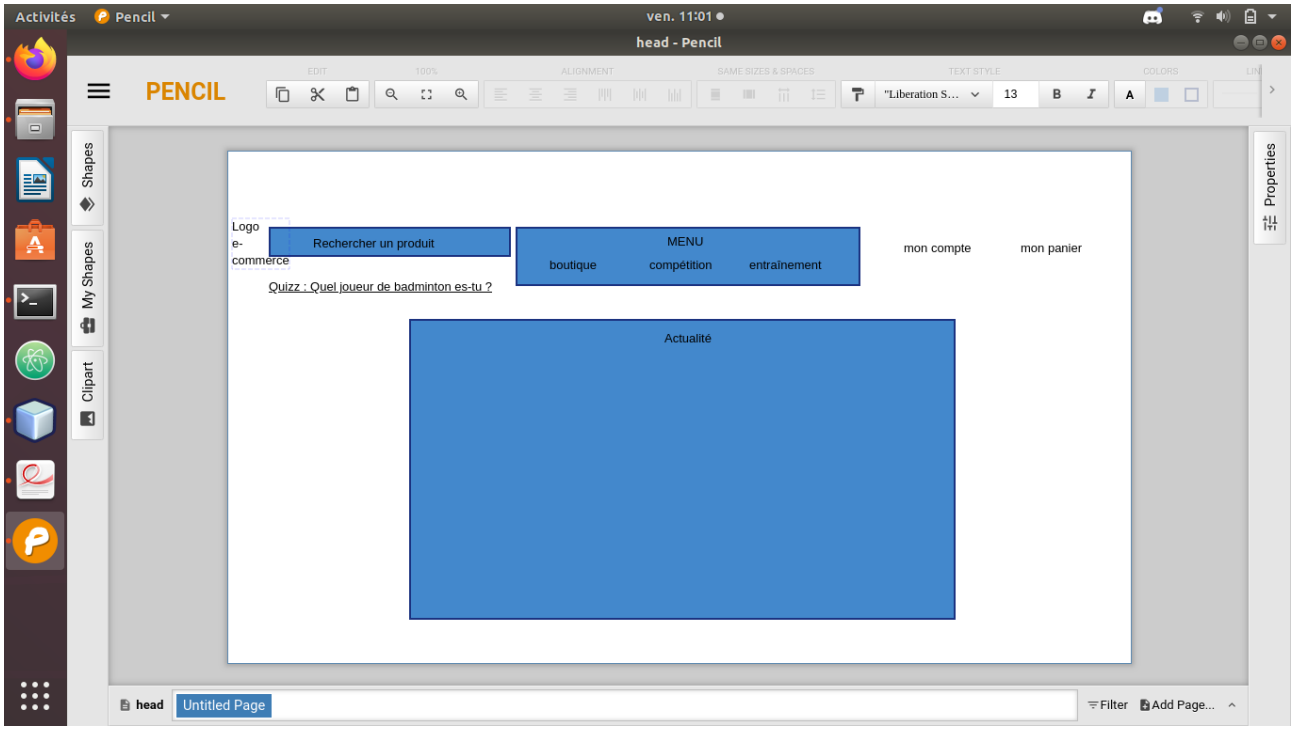
Netbeans	Environnement de développement intégré (IDE).
PhpMyAdmin	Système de gestion de base de données (SGBD) pour gérer la base de données ainsi que la visualiser.
Jmerise	Logiciel de modélisation permettant de schématiser une base de données, ses différentes relations et de générer son script SQL.
Pencil	Logiciel de maquettage.
Git	Logiciel de gestion de versions décentralisées, permettant la sauvegarde progressive de son avancée et de récupérer en cas de perte.

V. Réalisations (sécurité et web mobiles)

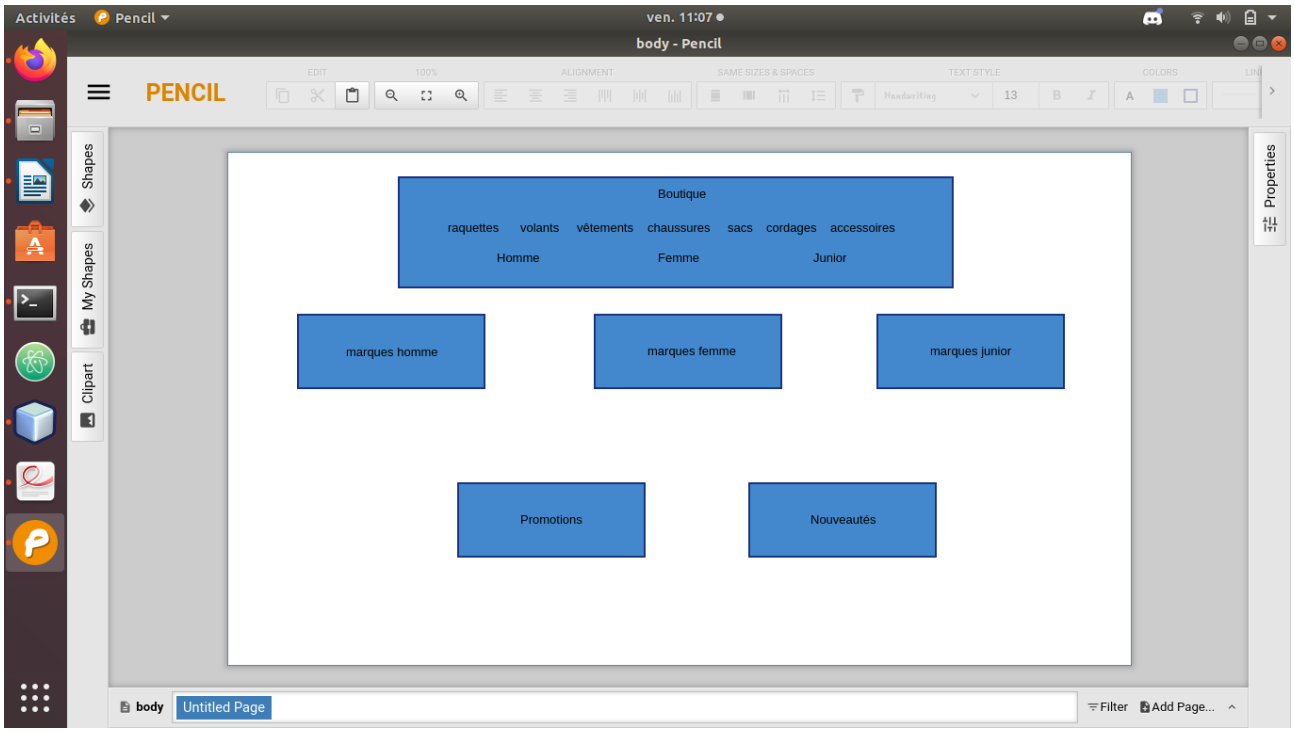
A) Partie Front-end

1. Maquetter une application

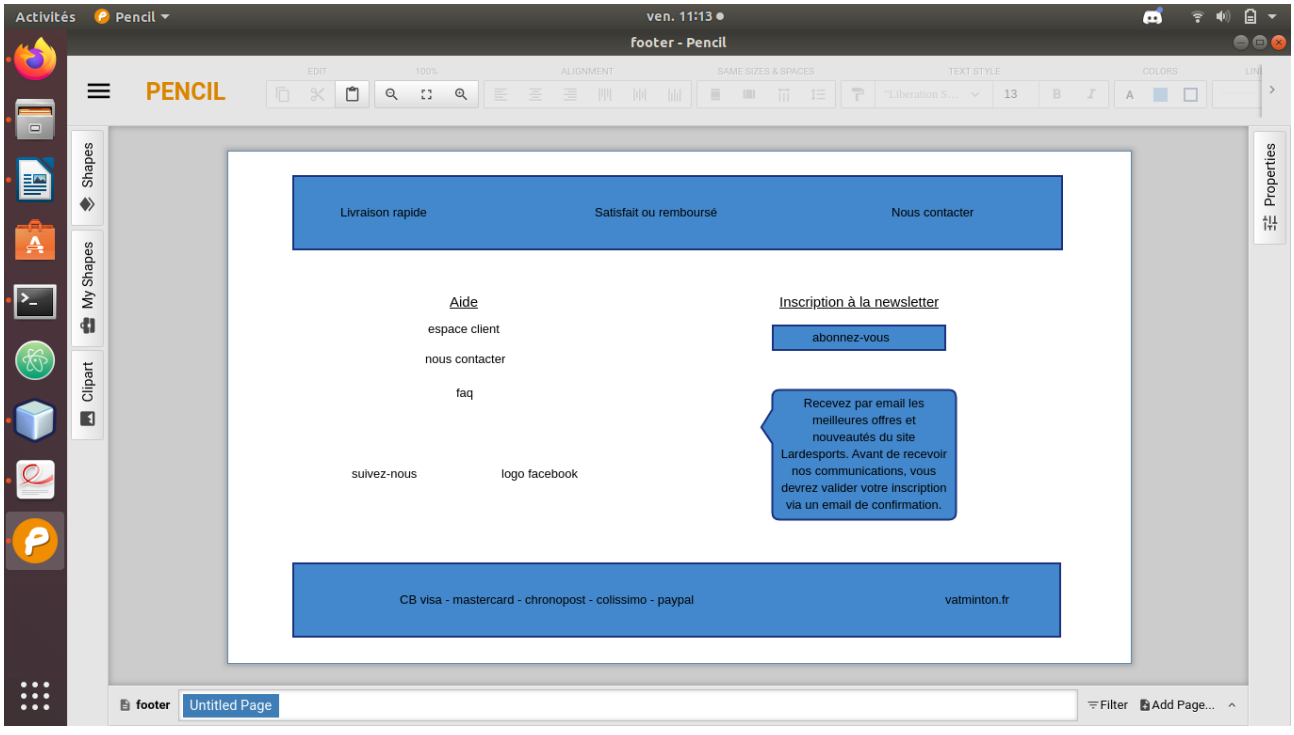
Après une mûre réflexion sur l’aspect visuel de mon site, j’ai réalisé un maquettage de mon projet sous le logiciel Pencil.



Vue de l’en-tête de ma page



Vue du corps de ma page



Vue du pieds de ma page

2. Réaliser une interface utilisateur web statique et adaptable, WEB MOBILE (responsive)

Le responsive Web design est une approche de conception Web qui vise à l'élaboration de sites offrant une expérience de lecture et de navigation optimales pour l'utilisateur quelle que soit sa gamme d'appareil (téléphones mobiles, tablettes, liseuses, moniteurs d'ordinateur de bureau).
Une expérience utilisateur "Responsive" réussie implique un minimum de redimensionnement (zoom), de recadrage, et de défilements multidirectionnels de pages.

Voici un échantillon de mon script :

```
<?php
<div class="row d-flex justify-content-center my-4">
<div class="col-lg-12">
<div class="news text-center my-3 p-0">Actualité n°1</div>
</div>
</div>
<!-- 1er article -->
<div class="row lee">
<div class="col-md-6 d-flex justify-content-center py-3">
<img src="../assets/img/leeChongWei.jpg" class="wei" alt="" /></div>
<div class="col-md-5">
<h1 class="display text-center">Atteint d'un cancer, Lee Chong Wei ne
participera pas aux Mondiaux</h1>
<p class="lead text-center">Atteint d'un cancer depuis l'été dernier, la star du
badminton Lee Chong Wei ne participera pas aux Mondiaux en Suisse au mois
 …
<p class="lead mt-2">Tombé à la 113e place mondiale, Lee Chong Wei a repris
l'entraînement au mois de janvier, mais a repoussé plusieurs fois depuis son
retour à la compétition.</p>
</div>
</div>

?>
```

Quelques explications :

on retrouvera le plus fréquemment des règles pour :

- agrandir la taille du texte
- agrandir la taille des contrôles et zones cliquables (pour une utilisation au doigt)
- faire passer le contenu sur une seule colonne
- masquer ou afficher des éléments spécifiques
- ajuster les dimensions et marges

Voici une partie de mon css :

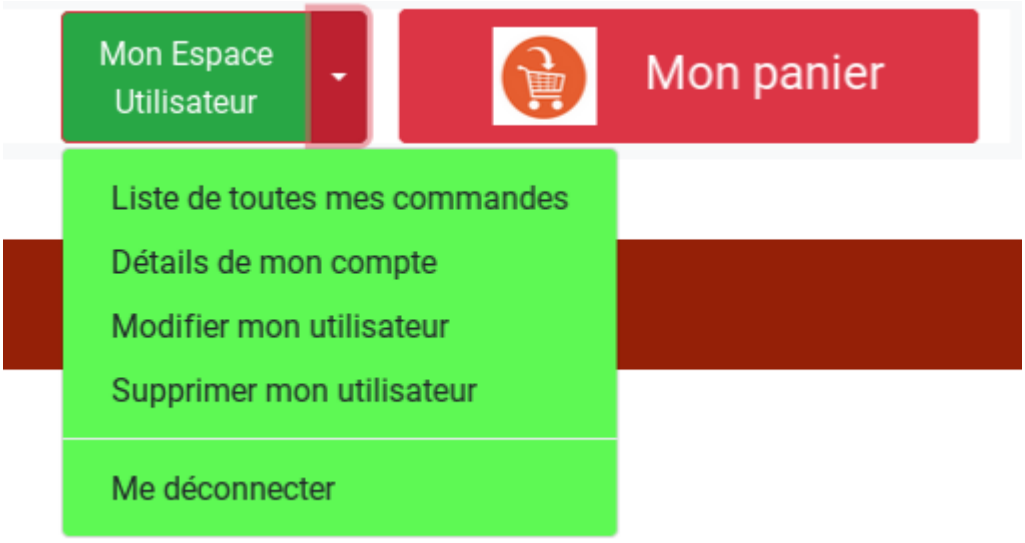
```
<?php
@media only screen and (max-width: 600px) {
img {
width:100%;
display: block;
float: none;
margin-left: auto;
margin-right: auto;
}
body {
width:100%;
background-color: #FFFFFF;
display: block;
float: none;
margin-left: auto;
margin-right: auto;
}
#carousel{
width: 85%;
display: block;
float: none;
margin-left: auto;
margin-right: auto;
}
}
?>
```

3. Développer une interface utilisateur web dynamique

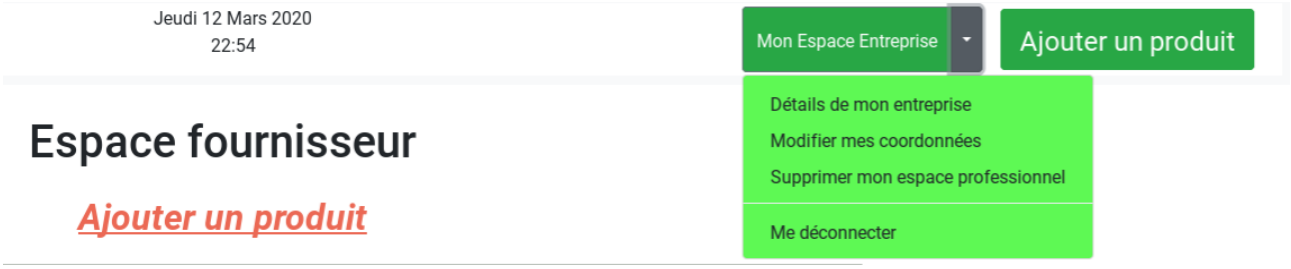
La barre de navigation est un élément essentiel du site puisqu’elle est présente sur chaque page.
Le contenu de ma page web dynamique peut donc varier en fonction d'informations (role utilisateur, entreprise ou administrateur, panier, formulaire rempli par l'utilisateur, etc.) qui ne sont connues qu'au moment de sa consultation. À l'inverse, le contenu d'une page web statique est *a priori* identique à chaque consultation.

Les 3 “navBar” différentes de mon site :

Barre de navigation de l’utilisateur



Barre de navigation du fournisseur



Mode administrateur avec tous les droits et ses contenus

Explications : J'ai créé un fichier "header.php" et "footer.php" afin de pouvoir ensuite l'inclure dans mes pages, ce qui évite de remettre le même code (allègement de la taille de mes fichiers). Pour l'intégrer, j'utilise la fonction PHP "include".

Comme les images ci-dessus, pour que mon application soit fonctionnelle, j'ai codé de telle sorte que chaque "role" à des fonctions différentes dès leur connexion au site.

Les 3 rôles sont :

- Administrateur
- Utilisateur
- Fournisseur

Un utilisateur par exemple ne pourra pas *ajouter un produit* ou un fournisseur ne pourra pas *ajouter dans son panier*.

Partie Back-end

4. Créer une base de données

Pour réaliser mon projet, j’ai réfléchi à la structure de ma base de données, aux tables et aux champs dont j’aurais besoin.

MCD : Modèle conceptuel de Données

Pour concevoir ma base de données, j’ai utilisé l’outil de modélisation JMerise.
J’ai tout d’abord créé les différentes tables et ajouté les relations entre elles.
Ma base de données comporte 8 tables, auxquelles j’ai ajouté un préfixe pour augmenter la sécurisation des données.

En effet, sachant que ma base de données va contenir la majeure partie des informations du site, ces failles peuvent avoir pour but de :

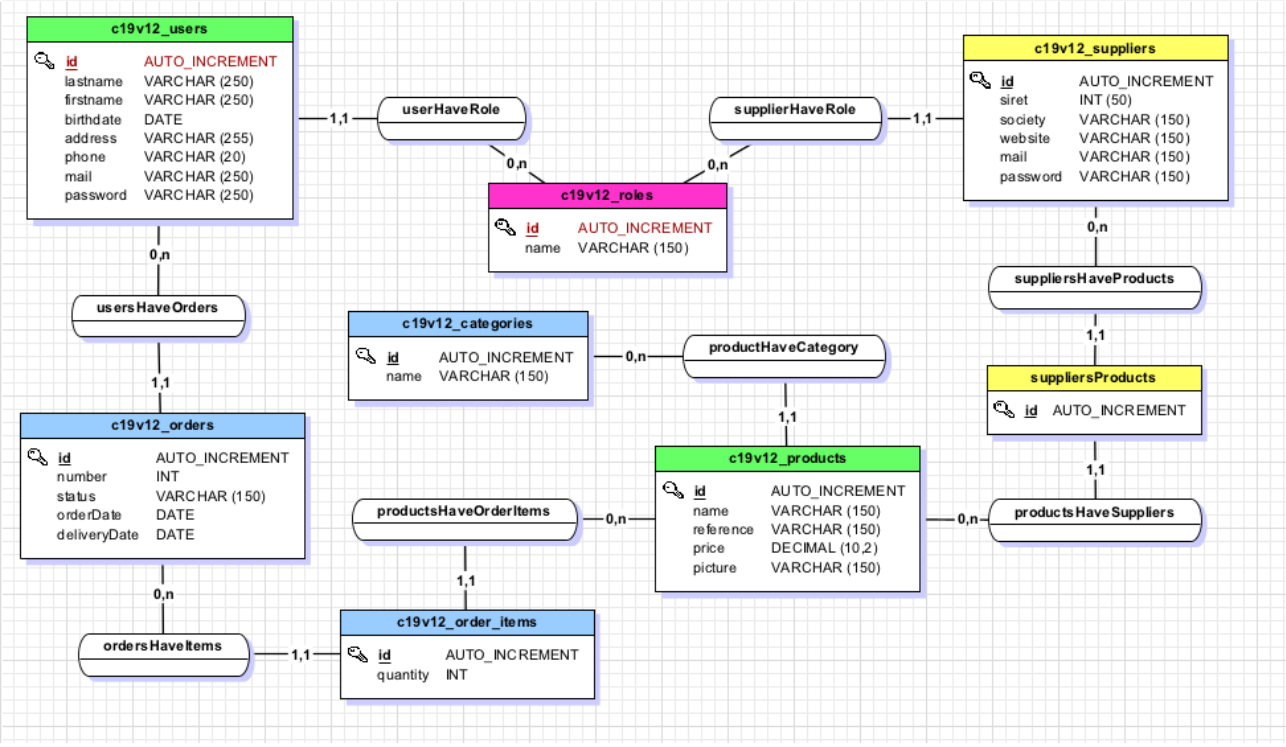
- Détruire le contenu du site
- Faire “planter” le site
- Ajouter des commentaires de spams
- Extraire des informations (ex : récupérer des adresses mails)
- Ajouter du contenu sur le site (ex : liens vers des sites illégaux)

Le site sera composé des tables suivantes :

- “c19v12_users” stocke les informations des utilisateurs
- “c19v12_roles” contient les différents rôles qui peuvent être attribués
- “c19v12_suppliers” stocke les informations des fournisseurs
- “c19v12_products” stocke les informations des produits
- “c19v12_suppliersProducts” stocke les informations des produits des fournisseurs
- “c19v12_catégories” stocke les informations du nom de catégorie
- “c19v12_orders” stocke les informations de commande
- “c19v12_order_items” stocke les quantités de produits

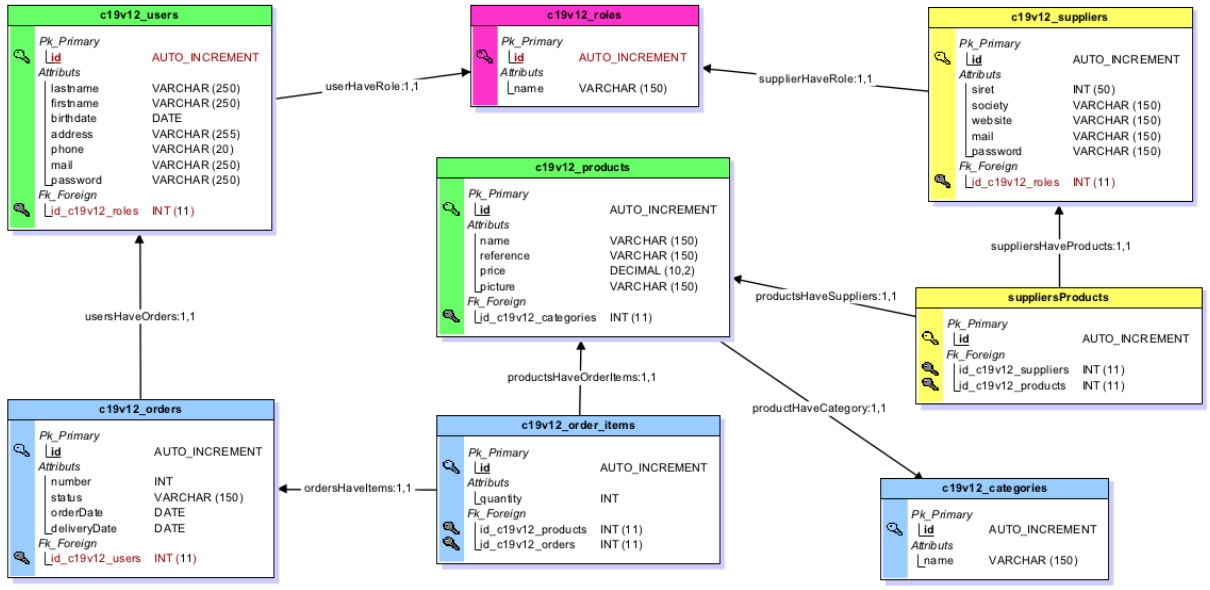
Mes tables auront des relations entre elles :

- Un utilisateur, comme le fournisseur et l'administrateur aura obligatoirement un rôle "role"
- Un utilisateur ne pourra pas ajouter, modifier ou supprimer un produit
- Un fournisseur n'aura pas accès au panier de la boutique pour y faire ces achats car l'espace est dédié à l'utilisateur
- Un produit appartient à une catégorie
- Un utilisateur aura accès au panier avec toutes les commandes en cours. Il pourra d'ailleurs accéder à toutes les coordonnées le concernant et les modifier, ou supprimer
- Un fournisseur aura un accès dédié pour ajouter un produit. Il pourra d'ailleurs accéder à toutes les informations de son entreprise et les modifier, ou supprimer
- Un utilisateur connecté pourra faire des commandes de vente en ligne
- Un panier peut avoir une ou plusieurs commandes du même utilisateur
- Un utilisateur pourra passer autant de commande qu'il le souhaite
- Un fournisseur pourra ajouter autant de produit qu'il le souhaite



MLD : Modèle logique de Données

Grâce à ces relations, il est possible de générer le MLD sur lequel vont apparaître les clés étrangères, permettant de lier les tables entre elles.



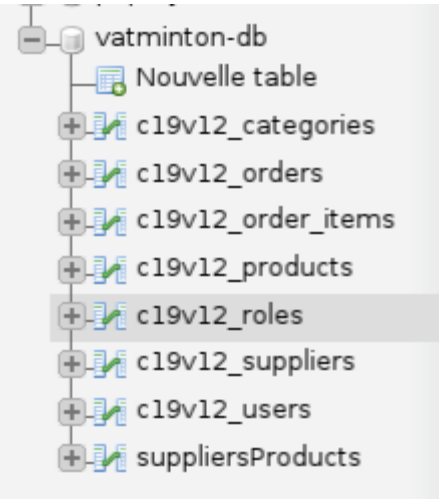
A partir de JMerise, j'ai généré un script SQL pour créer ma base de données que j'ai intégré à partir de la **console de linux** et par la suite avec **PhpMyAdmin**.

```
<?php
#------------------------------------------------------------
#        Script MySQL.
#------------------------------------------------------------
#------------------------------------------------------------
# Table: c19v12_roles
#------------------------------------------------------------
CREATE TABLE c19v12_roles(
        id   Int  Auto_increment  NOT NULL ,
        name Varchar (150) NOT NULL
      ,CONSTRAINT c19v12_roles_PK PRIMARY KEY (id)
)ENGINE=InnoDB;
#------------------------------------------------------------
# Table: c19v12_suppliers
#------------------------------------------------------------
CREATE TABLE c19v12_suppliers(
        id               Int  Auto_increment  NOT NULL ,
        siret            Int NOT NULL ,
        society          Varchar (150) NOT NULL ,
        website          Varchar (150) ,
        mail             Varchar (150) NOT NULL ,
        password         Varchar (150) NOT NULL ,
        id_c19v12_roles Int NOT NULL
      ,CONSTRAINT c19v12_suppliers_PK PRIMARY KEY (id)
      ,CONSTRAINT c19v12_suppliers_c19v12_roles_FK FOREIGN KEY (id_c19v12_roles)
REFERENCES c19v12_roles(id)
)ENGINE=InnoDB;
```

```
#------------------------------------------------------------
# Table: c19v12_categories
#------------------------------------------------------------
CREATE TABLE c19v12_categories(
        id   Int  Auto_increment  NOT NULL ,
        name Varchar (150) NOT NULL
       ,CONSTRAINT c19v12_categories_PK PRIMARY KEY (id)
)ENGINE=InnoDB;
#------------------------------------------------------------
# Table: c19v12_products
#------------------------------------------------------------
CREATE TABLE c19v12_products(
        id                   Int  Auto_increment  NOT NULL ,
        name                 Varchar (150) NOT NULL ,
        reference            Varchar (150) NOT NULL ,
        price                Decimal (10,2) NOT NULL ,
        picture              Varchar (150) ,
        id_c19v12_categories Int NOT NULL
       ,CONSTRAINT c19v12_products_PK PRIMARY KEY (id)
       ,CONSTRAINT c19v12_products_c19v12_categories_FK FOREIGN KEY
(id_c19v12_categories) REFERENCES c19v12_categories(id)
)ENGINE=InnoDB;
#------------------------------------------------------------
# Table: c19v12_users
#------------------------------------------------------------
CREATE TABLE c19v12_users(
        id              Int  Auto_increment  NOT NULL ,
        lastname        Varchar (250) NOT NULL ,
        firstname       Varchar (250) NOT NULL ,
        birthdate       Date NOT NULL ,
        address         Varchar (255) NOT NULL ,
        phone           Varchar (20) NOT NULL ,
        mail            Varchar (250) NOT NULL ,
        password        Varchar (250) NOT NULL ,
        id_c19v12_roles Int NOT NULL
       ,CONSTRAINT c19v12_users_PK PRIMARY KEY (id)
       ,CONSTRAINT c19v12_users_c19v12_roles_FK FOREIGN KEY (id_c19v12_roles)
REFERENCES c19v12_roles(id)
)ENGINE=InnoDB;
#------------------------------------------------------------
# Table: c19v12_orders
#------------------------------------------------------------
CREATE TABLE c19v12_orders(
        id              Int  Auto_increment  NOT NULL ,
        number          Int NOT NULL ,
        status          Varchar (150) NOT NULL ,
        orderDate       Date NOT NULL ,
        deliveryDate    Date NOT NULL ,
        id_c19v12_users Int NOT NULL
       ,CONSTRAINT c19v12_orders_PK PRIMARY KEY (id)
       ,CONSTRAINT c19v12_orders_c19v12_users_FK FOREIGN KEY (id_c19v12_users)
REFERENCES c19v12_users(id)
)ENGINE=InnoDB;
```

```
#------------------------------------------------------------
# Table: suppliersProducts
#------------------------------------------------------------
CREATE TABLE suppliersProducts(
        id                  Int  Auto_increment  NOT NULL ,
        id_c19v12_suppliers Int NOT NULL ,
        id_c19v12_products  Int NOT NULL
       ,CONSTRAINT suppliersProducts_PK PRIMARY KEY (id)
       ,CONSTRAINT suppliersProducts_c19v12_suppliers_FK FOREIGN KEY
(id_c19v12_suppliers) REFERENCES c19v12_suppliers(id)
       ,CONSTRAINT suppliersProducts_c19v12_products0_FK FOREIGN KEY
(id_c19v12_products) REFERENCES c19v12_products(id)
)ENGINE=InnoDB;
#------------------------------------------------------------
# Table: c19v12_order_items
#------------------------------------------------------------
CREATE TABLE c19v12_order_items(
        id                 Int  Auto_increment  NOT NULL ,
        quantity           Int NOT NULL ,
        id_c19v12_products Int NOT NULL ,
        id_c19v12_orders   Int NOT NULL
       ,CONSTRAINT c19v12_order_items_PK PRIMARY KEY (id)
       ,CONSTRAINT c19v12_order_items_c19v12_products_FK FOREIGN KEY
(id_c19v12_products) REFERENCES c19v12_products(id)
       ,CONSTRAINT c19v12_order_items_c19v12_orders0_FK FOREIGN KEY
(id_c19v12_orders) REFERENCES c19v12_orders(id)
)ENGINE=InnoDB;
?>
```

Voici la structure de ma base de données dans phpMyAdmin :



5. Développer les composants d’accès aux données

Pour une question pratique et économiser la taille des fichiers, j’ai placé mon en-tête et mon pied de page dans 2 fichiers distincts : “header.php” et “footer.php”. Il me suffit ensuite d’inclure ces 2 fichiers dans chacune de mes pages.

voici mon code :

```
<?php
session_start();
require_once '../init/credentials.php';
require_once '../models/database.php';
...
include 'header.php';
?>
…
<?php include 'footer.php'; ?>
```

Pour accéder aux données de ma base de données, je crée la méthode **“getInstance()”** dans mon modèle “database”.

La connexion à ma base de données s’effectue de la façon suivante :

```
<?php
class Database {
    // Objet de connexion PDO
    public $db;
    // Objet auto-instancié utilisé avec les méthodes de transaction SQL
    private static $instance;
    private $host     = DB_HOST;
    private $database = DB_NAME;
    private $username = DB_USER;
    private $password = DB_PWD;


    /**
    * Initialise une connection à la base de données
    **/
    public function __construct() {
      try {
        // -- IMPORTANT --
        // Les tables doivent être du type innoDB pour prendre en charge les
transactions.
        // Les tables de type MyISAM ne gèrent pas les transactions.
        $this->db = new PDO('mysql:host=' . $this->host .';dbname=' .
$this->database . ';charset=utf8',
                            $this->username,
                            $this->password);
```

```
        // En mode ERRMODE_EXCEPTION, si un échec survient, le script est
        // interrompu, la connexion fermée, et mysql effectue un roll back
        // sur la transaction
        $this->db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
      } catch (Exception $e) {
        die('Error : ' . $e->getMessage());
      }
    }
    /**
     * Méthode retournant une instance de classe PDO
     * @return object
     */
    public static function getInstance() {
      if ( is_null( self::$instance ) ) {
        self::$instance = NEW Database();
      }
      return self::$instance;
    }
    /**
     * Fermeture automatique de la connexion lorsqu'une instance de classe est
détruite
     */
    public function __destruct() {
      $this->db = NULL;
    }
  }
?>
```

Explications : La méthode getInstance permet de récupérer l'instance de connexion à ma base de données. Elle permet de créer une instance unique de mon objet.

6. Développer la partie Back-end d'une application web et web mobile

Pour développer la partie Back-end de mon application web, j'ai choisi de structurer mon code avec l'architecture (MVC Modèle Vue Contrôleur) de PHP.
-> Un modèle (Model) qui contient les requêtes qui vont me permettre de récupérer les données à afficher (en PHP)
-> Une vue (view) qui correspond à l'interface graphique (en HTML.CSS)
-> Un contrôleur (Controller) qui permet de faire dialoguer la vue et le modèle (en PHP)

Voici l’architecture de mes dossiers dans netbeans :

VI. Présentation CRUD “product”

Qu’est ce que le CRUD ?

L'acronyme informatique anglais **CRUD** (pour *create*, *read*, *update*, *delete*) (parfois appelé SCRUD avec un "S" pour *search*) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données.

Soit :

- **c**reate : créer
- **r**ead : lire
- **u**pdate : mettre à jour
- **d**elete : supprimer

Plus généralement, il désigne les opérations permettant la gestion d'une collection d'éléments.

Ce terme est aussi un jeu de mot en anglais sur l'adjectif **crude** (en français **brut** ou **rudimentaire**).

1) **C**reate : product

1.La première étape consiste à créer une table de base de données qui sera manipulée en utilisant l’accès CRUD. Pour cela, il faut soumettre la requête suivante dans votre base de données MySQL :

```php
<?php
public function addProductBySupplier() {
 //Soumission de la requête au serveur de la base de données
 $results = $this->pdo->db->prepare('INSERT INTO `c19v12_products` (`name`,
`reference`, `price`, `picture`, `id_c19v12_categories`)
 VALUES (:name, :reference, :price, :picture, :id_c19v12_categories)');
 // Associe une valeur à chaque marqueur nominatif de la requête
$results->bindValue(':name', $this->name, PDO::PARAM_STR);
$results->bindValue(':reference', $this->reference, PDO::PARAM_STR);
$results->bindValue(':price', $this->price, PDO::PARAM_STR);
$results->bindValue(':picture', $this->picture, PDO::PARAM_STR);
$results->bindValue(':id_c19v12_categories', $this->id_c19v12_categories,
PDO::PARAM_INT);
// Exécution de la requête
try {
return $results->execute(); // Méthode execute() toujours avec prepare() sinon
query()
}
catch (Exception $e) {
die('échec de la connexion :' . $e->getMessage());
}
}
?>
```

La table est utilisée pour collecter des informations sur l'utilisateur : le nom, la référence, le prix, l'imagetable. Chaque entrée reçoit automatiquement une **clé primaire** (*AUTO_INCREMENT PRIMARY KEY*), c'est-à-dire son propre ID unique.

2. Il est ensuite nécessaire de configurer la connexion et de supprimer la base de données. Il faut créer un fichier PHP sous le nom database.php, il est ensuite nécessaire d'insérer le script suivant avec la classe « Database » pour gérer les connexions de base de données :

« Database » pour gérer les connexions de base de données :

```php
<?php
class Product extends Database {
    public $pdo;
//attributs
//(seront utilisés lorsque l'on récupèrera des données à partir du formulaire)
    public $id;
    public $name;
    public $reference;
    public $price;
    public $picture;
    public $id_c19v12_categories;
    public $formErrors = array();
    public function __construct() {
        $this->pdo = Database::getInstance(); // Création de l'instance de
connexion
    }
    public function __destruct() {
        $this->pdo->db = NULL;
    }
    /*
     * Méthode démarrant une transaction SQL
     */
    public function beginTransaction() {
        return $this->pdo->db->beginTransaction();
    }
    /*     * Méthode appliquant la transaction courante et rendant ses
modifications permanentes     */
public function commit() {
        return $this->pdo->db->commit();
    }
    /*
     * Méthode faisant un rollback de la transaction courante et annulant ses
modifications
     */
    public function rollBack() {
        return $this->pdo->db->rollBack();
    }
    /*
     * Method retournant l'id de la dernière rangée insérée en base de données
     */
    public function lastInsertId() {
        return $this->pdo->db->lastInsertId();
    }
?>
```

Afin de pouvoir utiliser ici la classe définie pour l'accès à la base de données avec PDO, il faut **spécifier exactement les valeurs pour les quatre entrées $database** (nom de la base de donnée utilisée), **$host** (nom de l'hôte sur lequel la base de données est en cours d'exécution), **$username** (nom d'utilisateur) et **$password** (mot de passe de l'utilisateur).
Dans ce script, la classe de base de données comporte trois fonctions : *__construct()*, le **constructeur de classe** permet que la classe rappelle aux utilisateurs que l'initialisation (c'est-à-dire l'assignation d'une valeur initiale ou de départ) n'est pas autorisée. Ensuite *connect()* est la fonction principale de la classe qui régule la connexion enfin au contraire *disconnect()* est utilisée pour mettre fin à la connexion.

```
<?php
// Création d'une instance de classe Product. Instance de classe = OBJET
$product = new Product();
// Récupération des données du formulaire
// Affectation de chaque champ de formulaire à l'attribut auquel il est associé
$product->name = isset($_POST['name']) ? htmlspecialchars($_POST['name']) : '';
$product->price = isset($_POST['price']) ? htmlspecialchars($_POST['price']) :
'';
$product->reference = isset($_POST['reference']) ?
htmlspecialchars($_POST['reference']) : '';    $product->id_c19v12_categories =
isset($_POST['category']) && $_POST['category'] > 0 ? $_POST['category'] : '';
// input name = category
//Toutes regex pour le formulaire
$regexNameProduct = '/^[a-zA-Z0-9À-ÿ\' -]+$/';
$regexReference = '/^[A-Z]{1,5}[0-9]{1,5}$/';
$regexPrice = '/^\d+(\d{3})*(\.\d{1,2})?$/';
…
// Validation de picture
if (isset($_FILES['picture']) && $_FILES['picture']['error'] == 0) {
$filesInfo = basename($_FILES['picture']['name']);
$extension_upload = $filesInfo['extension'];
$extension_allowed = array('jpg','png','bmp');
$product->picture = $filesInfo;
if (in_array($extension_upload, $extension_allowed)) {
echo 'Le fichier transmis est bien un fichier jpg';
} else {
echo '/!\ vous devez sélectionner un fichier JPG /!\ ';
}
}
// Insertion du product dans la base de données
if (empty($product->formErrors)) {
$success = $product->addProductBySupplier();
?>
```

La page *createUser.php* est désormais activée, permettant ainsi d'entrer les informations utilisateur. Le script garantit que toutes les **données entrées et les messages d'erreur** sont enregistrés et que les messages d'erreur correspondants apparaissent bien quand une **entrée est incorrecte**, enfin le script assure aussi la **transmission des données à la base de données.**

2) **R**ead : product

La procédure Read lit les enregistrements d'une table sur la base de clé primaire indiqué en paramètre d'entrée. Mes 2 fichiers permettant de lire dans mon CRUD User les enregistrements d'une table sont "listUser.php" et "profilUser.php".

Ma vue "listProduct.php" permet d'afficher la liste des produits. Mon code :

```
<?php
session_start();
require_once '../init/functions.php';
require_once '../init/credentials.php';
require_once '../models/database.php';
require_once '../models/product.php';
require_once '../models/category.php';
require_once '../controllers/listProductCtrl.php';
//require_once '../controllers/listProductSearchCtrl.php';
require_once 'header.php';
?>
<div id="detailsProduct" class="h1 text-center mt-4">LISTE DES PRODUITS
</div>
<div class="row d-flex justify-content-center mt-5">

    <input type="hidden" name="id" value="<?= isset($_GET['id']) ? $_GET['id'] :
'' ?>" />
    <?php foreach ($productList as $product) { ?>

        <div class="col-md-4 d-flex justify-content-center mt-3">
            <div class="card text-black text-center bg-secondary mb-3"
style="max-width: 20rem;">
                <div class="card-header"><?= $product->id ?><?=
$product->productName ?></div>
                <div class="card-body">
                    <div><img src="../assets/img/<?= $product->picture ?>"
class="picture" alt="vatminton" /></div>
                    <h4 class="card-title">référence : <?= $product->reference
?></h4>
                    <p class="card">Catégorie de produit : <?=
$product->categoryName ?></p>
                    <p class="card">Prix : <?= $product->price ?> €</p>
                    <a class="px-3" href="detailsProduct.php?id=<?= $product->id
?>">Détails</a>
                    <?php if (isUser() || isAdmin()) { ?>
                        <a class="px-3" href="addToCart.php?id=<?= $product->id
?>">Ajouter à mon panier</a>
                    <?php } ?>
                </div>
            </div>
        </div>
    <?php }
    ?>
</div>
```

Mon contrôleur “listProductCtrl.php” permet de faire toutes les vérifications nécessaires avant d’envoyer les données à la vue. Mon code :

```php
<?php
$product = new Product();
$productList = $product->getProductList();
?>
```

3) Update : product

La procédure Update utilise la commande sql UPDATE sur la table en se basant bien sûr la clé primaire spécifié dans la clause WHERE. Tout comme la procédure Create il possède un paramètre pour chaque colonne dans la table. La requête *“prepare” évite l’injection de code SQL.*

Voici mon script permettant de modifier le profil de l’utilisateur et de le mettre à jour dans la base de données “product.php” :

```php
<?php
public function updateProfilProduct() {
        //Soumission de la requête au serveur de la base de données
        $results = $this->pdo->db->prepare('UPDATE `c19v12_products` SET  `name` = :name, `reference` = :reference, `price` = :price, `picture` = :picture
                                WHERE `id` = :id');
        // :mail est un marqueur nominatif, pour éviter l'injection de code SQL
        // Associe une valeur à chaque marqueur nominatif de la requête
        $results->bindValue(':id', $this->id, PDO::PARAM_INT);
        $results->bindValue(':name', $this->name, PDO::PARAM_STR); //dans On nomme ce qu'on veut dans bindvalue
        $results->bindValue(':reference', $this->reference, PDO::PARAM_STR);
        $results->bindValue(':price', $this->price, PDO::PARAM_STR);
        $results->bindValue(':picture', $this->picture, PDO::PARAM_STR);
        $results->bindValue(':name', $this->name, PDO::PARAM_STR);
        // Exécution de la requête
        try {
            return $results->execute(); // Méthode execute() toujours avec prepare() sinon query()
            // execute retourne true ou false
        } catch (Exception $e) {
            die('échec de la connexion :' . $e->getMessage());
        }
    }
?>
```

Ma vue “update.php” permet de modifier un produit. Mon code :

```php
<?php
session_start();
require_once '../init/functions.php';
require_once '../init/credentials.php';
```

```php
require_once '../models/database.php';
require_once '../models/product.php';
require_once '../models/category.php';
require_once '../models/role.php';
require_once '../controllers/updateProductCtrl.php';
include 'header.php';
?>
<div id="formulary" class="h1 text-center my-4">Espace fournisseur</div>
<h2 class="text-center mb-4">Modifier un produit</h2>
<div class="container">
    <div class="row">
        <div id="mainForm" class="col-md-12">
            <div class="row justify-content-center">
                <div class="col-md-12">
                    <div class="formA p-4 mt-2">
                        <form method="POST" action=""
enctype="multipart/form-data">
                            <!-- Nom -->
                            <div class="form-row">
                                <div class="form-group col-md-6 text-center">
                                    <label for="name">Nom : </label>
                                    <input type="text" class="form-control
text-center text-info" name="name" placeholder="" id="name" value="<?=
isset($detailsProduct->productName) ? $detailsProduct->productName : '' ?>" />
                                    <small class="text-warning"><?=
isset($profilProduct->formErrors['name']) ? $profilProduct->formErrors['name'] :
'' ?></small>
                                </div>
                                <!-- Price -->
                                <div class="form-group col-md-6 text-center">
                                    <label for="price">Prix : </label>
                                    <input type="text" class="form-control
text-center text-info" name="price" id="price" placeholder="" value="<?=
isset($detailsProduct->price) ? $detailsProduct->price : '' ?>" />
                                    <small class="text-warning"><?=
isset($profilProduct->formErrors['price']) ? $profilProduct->formErrors['price']
: '' ?></small>
                                </div>
                                <!-- Reference -->
                                <div class="form-group col-md-8 text-center">
                                    <label for="reference">Référence du produit
: </label>
                                    <input type="text" class="form-control
text-center text-info" name="reference" id="reference" placeholder="" value="<?=
isset($detailsProduct->reference) ? $detailsProduct->reference : '' ?>">
                                    <small class="text-warning"><?=
isset($profilProduct->formErrors['reference']) ?
$profilProduct->formErrors['reference'] : '' ?></small>
                                </div>
                                <!-- Categories -->
                                <div class="form-group col-md-4 text-center">
                                    <label for="categories">Catégorie : </label>
                                    <select id="categories" name="category"
class="form-control"> <!-- le name="category" est lié à $category dans le
controller -->
                                        <option value="<?=
$detailsProduct->id_c19v12_categories ?>" selected><?=
$detailsProduct->categoryName ?></option>
                                        <?php foreach ($categoryList as $item) {
?>
```

```
                                        <option class="text-white"
value="<?= $item->id ?>"><?= ucfirst($item->name) ?></option> <!-- uppercase
first letter -->
                                    <?php } ?>
                                </select>
                                <small class="text-warning"><?=
isset($profilProduct->formErrors['categories']) ?
$profilProduct->formErrors['categories'] : '' ?></small>
                            </div>
                            <div class="row">
                                <div class="form-group col-md-12
text-center">
                                    <input type="file" name="picture" />
                                </div>
                                <input type="submit" name="submit"
value="Enregistrer" />
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

Mon contrôleur “updateProductCtrl.php” permet de faire toutes les vérifications nécessaires avant d’envoyer les données à la vue. Mon code :

```
<?php
$product = new Product();
$category = new Category();
$product->id = isset($_GET['id']) ? $_GET['id'] : '';
$detailsProduct = $product->detailsProduct();
$categoryList = $category->getCategories();
if (isset($_POST['submit'])) {
// Récupération des données du formulaire
// Affectation de chaque champ de formulaire à l'attribut auquel il est associé
    $product->name = isset($_POST['name']) ? htmlspecialchars($_POST['name']) :
'';
    $product->price = isset($_POST['price']) ? htmlspecialchars($_POST['price'])
: '';
    $product->reference = isset($_POST['reference']) ?
htmlspecialchars($_POST['reference']) : '';
    $product->id_c19v12_categories = isset($_POST['category']) &&
$_POST['category'] > 0 ? $_POST['category'] : ''; // input name = category
//Toutes regex pour le formulaire
    $regexNameProduct = '/^[a-zA-Z0-9À-ÿ\' -]+$/';
    $regexReference = '/^[A-Z]{1,5}[0-9]{1,5}$/';
    $regexPrice = '/^\d+(\d{3})*(\.\d{1,2})?$/';
// Validation du nom du produit
    if (empty($product->name)) {
        $product->formErrors['name'] = 'Ce champ est vide';
    } elseif (!preg_match($regexNameProduct, $product->name)) {
        $product->formErrors['name'] = 'Merci de rentrer un nom de produit
valide';
    } elseif (strlen($product->name) < 1 || strlen($product->name) > 26) {
        $product->formErrors['name'] = 'Le nom doit comporter entre 2 et 25
caractères'; }
```

```
    // Validation des catégories (select)
    if (isset($_POST['categories'])) {
        if (empty($product->categories)) {
            $product->formErrors['categories'] = 'Ce champ est vide';
        } else {
            $product->formErrors['categories'] = 'champ est vide';
        }
    }
    // Validation du prix
    if (empty($product->price)) {
        $product->formErrors['price'] = 'Ce champ est vide';
    } elseif (!preg_match($regexPrice, $product->price)) {
        $product->formErrors['price'] = 'Merci de rentrer un prix valide';
    }
    // Validation de la référence
    if (empty($product->reference)) {
        $product->formErrors['reference'] = 'Ce champ est vide';
    } elseif (!preg_match($regexReference, $product->reference)) {
        $product->formErrors['reference'] = 'Merci de rentrer un numéro de
référence valide';
    }
    // Validation de picture
    if (isset($_FILES['picture']) && $_FILES['picture']['error'] == 0) {
        $filesInfo = basename($_FILES['picture']['name']);
        $extension_upload = $filesInfo['extension'];
        $extension_allowed = array('jpg','png','bmp');
        $product->picture = $filesInfo;
        if (in_array($extension_upload, $extension_allowed)) {
            echo 'Le fichier transmis est bien un fichier jpg, png, bmp';
        } else {
            echo '/!\ vous devez sélectionner un fichier JPG, PNG, BMP /!\ ';
        }
    }
    // Insertion du product dans la base de données
    if (empty($profilProduct->formErrors)) {
        $success = $product->updateProfilProduct();
        header('Location: ../../views/listProduct.php');
        //header('Location: ../../views/detailsProduct.php?id=' . $product->id);
        exit();
    }
}
?>
```

4) Delete : user

Cette procédure supprime une ligne spécifiée dans la clause WHERE.

Mon code dans le modèle product.php :

```
<?php
public function deleteProduct() {
        //Définition de la requête SQL
        $results = $this->pdo->db->prepare('DELETE FROM `c19v12_products` WHERE
`id` = :id');
        // :mail est un marqueur nominatif, pour éviter l'injection de code SQL
        // Associe une valeur à chaque marqueur nominatif de la requête
        $results->bindValue(':id', $this->id, PDO::PARAM_INT);
        //Soumission de la requête au serveur de la base de données
```

```
        try {
            return $results->execute();
        } catch (Exception $e) {
            die('échec de la connexion :' . $e->getMessage());
        }
    }
?>
```

Ma vue “delete.php” permet de modifier un produit. Mon code :

```
<?php
session_start();
require_once '../init/functions.php';
require_once '../init/credentials.php';
require_once '../models/database.php';
require_once '../models/product.php';
require_once '../controllers/deleteProductCtrl.php';
require_once '../views/header.php';
?>
<div class="row">
    <div class="col-md-12 text-center my-5">Êtes-vous sûr de vouloir supprimer
le produit ?</div>
</div>
<div class="row">
    <div class="col-md-6 text-center">
        <a class="btn  bg-success mx-5"
href="../views/detailsProduct.php">Annuler</a>
    </div>
    <div class="col-md-6 text-center">
        <a href="deleteProduct.php?id=<?= $product->id ?>&confirm=true"
class="btn bg-danger mx-5">Confirmer</a>
    </div>
</div>
```

Mon contrôleur “deleteProductCtrl.php” permet de faire toutes les vérifications nécessaires avant d’envoyer les données à la vue. Mon code :

```
<?php
if (isset($_GET['id']) && $_GET['id'] > 0 && $_GET['confirm'] == true) {
    $product = new Product();
    $product->id = isset($_GET['id']) ? htmlspecialchars($_GET['id']) : '';
    $success = $product->deleteProduct();
    if ($success) {
        $_SESSION['message'] = 'Le produit a bien été supprimé';
    } else {
        $_SESSION['message'] = 'Le produit n\'a pas été supprimé';
    }
    header('Location: ../views/listProduct.php');
    exit();
} else {
    $product = new Product();
    $product->id = isset($_GET['id']) ? htmlspecialchars($_GET['id']) : '';
}
?>
```

VII. Description de la veille

Les failles de sécurité

A) Le htmlspecialchars

Cette fonction de PHP convertit les caractères spéciaux en entités HTML et permet ainsi permet l'injection de code HTML et JS.

Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML pour conserver leurs significations. Cette fonction retourne une chaîne de caractères avec ces modifications.

Si la chaîne en entrée passée à cette fonction et le document final partagent le même jeu de caractères, cette fonction est suffisante pour préparer l'entrée pour une inclusion dans la plupart des contextes d'un document HTML.

B) La REGEX

La REGEX (expression régulière) et un fichier php que j'ai nommé "functions". Voici quelques lignes de mes REGEX :

```php
<?php
$regexName = '/^[a-zA-ZÀ-ÿ\' -]+$/';
$regexAddress = '/[0-9]{1,3}(?:(?:[,. ]){1,2}[-a-zA-ZÀ-ÿ]+)+$/';
$regexBirth = '/^(19|20)[0-9]{2}-[0-9]{2}-[0-9]{2}$/';
$regexPhone = '/^[0-9]{2}(-|\ ?)[0-9]{2}(-|\ ?)[0-9]{2}(-|\ ?)[0-9]{2}(-|\ ?)[0-9]{2}$/';
$regexMail = '/^[^\W][a-zA-Z0-9]+(.[a-zA-Z0-9]+)@[a-zA-Z0-9]+(.[a-zA-Z0-9]+).[a-zA-Z]{2,4}$/';
$regexPassword = '/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*\W).{3,}$/';
?>
```

Explications : La regex ou expression régulière est une chaîne de caractères, qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles. Une regex évite par exemple qu'un numéro de téléphone "alphanumérique" ne soit accepté car un numéro de téléphone n'est composé que de chiffres.

La partie validation des champs dans mon formulaire vient renforcer la sécurité du site. Vous remarquerez que ce code complète la regex créée en amont (code ci-dessus *$regexMail).* Voici mon code de validation de l'adresse email :

```
<?php
// Validation du mail
    if (empty($supplier->mail)) {
        $supplier->formErrors['mail'] = 'Ce champ est vide';
    } elseif (!preg_match($regexMail, $supplier->mail)) {
        $supplier->formErrors['mail'] = 'Merci de rentrer un mail valide';
    } elseif (strlen($supplier->mail) > 100) {
        $supplier->formErrors['mail'] = 'Le mail est trop long';
    } elseif (!$supplier->hasUniqueMail()) {
        // Verifie si l'adresse mail existe déjà en base de données
        $supplier->formErrors['mail'] = 'Mail existant, veuillez saisir une
adresse mail différente';
    }
?>
```

C)les fonctions

1) "function debug($data), debugPrint($data) et debugSession()"

- Ma fonction debug($data) m'a permis de faire un var_dump pour identifier une erreur dans mon code et ainsi trouver une solution. Les codes de retour pour ces exceptions ont des noms au format SQL_ERROR_XXX dont voici le script :

```
<?php
public function __construct() {
      try {
        // -- IMPORTANT --
        // Les tables doivent être du type innoDB pour prendre en charge les
transactions.
        // Les tables de type MyISAM ne gèrent pas les transactions.
        $this->db = new PDO('mysql:host=' . $this->host .';dbname=' .
$this->database . ';charset=utf8',
                            $this->username,
                            $this->password);
        // En mode ERRMODE_EXCEPTION, si un échec survient, le script est
        // interrompu, la connexion fermée, et mysql effectue un roll back
        // sur la transaction
        $this->db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
      } catch (Exception $e) {
        die('Error : ' . $e->getMessage());
      }
    }
?>
```

- Ma fonction debugPrint(), à la différence de la fonction debug() elle permet d'avoir des informations sommaires.

La balise <pre> représente du texte préformaté, généralement écrit avec une police à chasse fixe. Le texte est affiché tel quel, les espaces utilisés dans le document HTML seront retranscrits (mets à la ligne).

- "function debugSession()"

Cette fonction permet d'afficher l'id du *"role".*

2) les 3 fonctions : isAdmin(), isUser() et isSupplier()

Ses 3 fonctions me permettent d'afficher certains éléments sur une page pour un *"role"* et non pour les autres. Elles permettent également de **limiter les données à chaque *"role".***

a) isAdmin()

```php
<?php
// Méthode certifiant que l'utilisateur est un administrateur
function isAdmin() {
if (isset($_SESSION['user_role']) && $_SESSION['user_role'] == 1) {
return TRUE;
} else {
return FALSE;
}
?>
```

b) isUser()

```php
<?php
// Méthode certifiant que l'utilisateur est un client enregistré
function isUser() {
if (isset($_SESSION['user_role']) && $_SESSION['user_role'] == 2) {
return TRUE;
} else {
return FALSE;
}
}
?>
```

c) isSupplier()

```php
<?php
// Méthode certifiant que l'utilisateur est un fournisseur
function isSupplier() {
if (isset($_SESSION['user_role']) && $_SESSION['user_role'] == 3) {
return TRUE;
} else {
return FALSE;
}
}
?>
```

D) Le “hashage” de mots de passe via la fonction PHP password_hash()

Le “hashage” de mot de passe est l’une des pratiques de sécurité les plus basiques qui doit être effectuée. Sans cela, chaque mot de passe stocké peut être volé si le support de stockage est compromis. Ce mot de passe peut alors être immédiatement utilisé pour accéder frauduleusement non seulement à votre application mais aussi sur d’autres applications si l’utilisateur utilise le même mot de passe ailleurs.

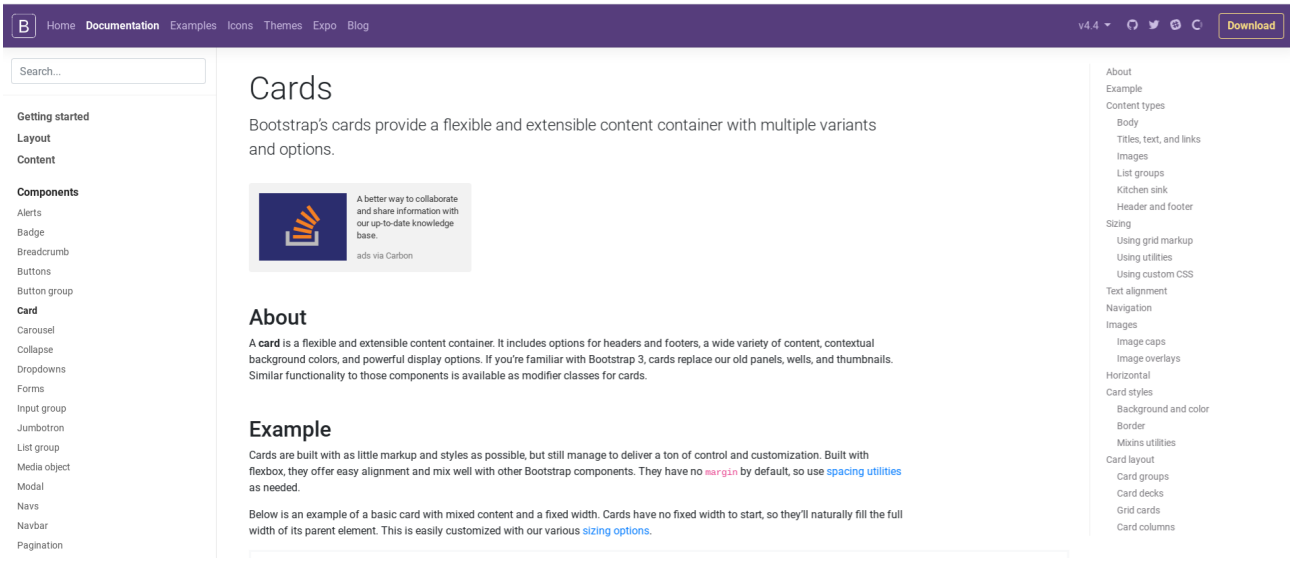
En appliquant un “hashage” sur le mot de passe avant de le stocker, vous rendez la tâche d’un attaquant très difficile pour connaître le mot de passe d’origine, et vous avez toujours la possibilité de comparer le mot de passe “hashé” à une chaîne reçue.

La fonction PHP password_hash() va créer une chaîne de caractères aléatoire, et c’est généralement la façon la plus sécurisée et la plus simple.

VIII. Description d’une recherche anglophone

Au cours de mes différentes recherches, j’ai été amené à consulter différents sites ou tuto entièrement en anglais et notamment le site https://getbootstrap.com/docs/4.4/components/card/ pour mes différentes vues (views) concernant l’affichage de la d’utilisateur, d’un fournisseur ou d’un produit.

Voici le modèle de “card” que j’ai utilisé :



Traduction :

“Cartes

Les cartes de Bootstrap offrent un conteneur de contenu flexible et extensible avec de multiples variantes et options.

À propos d'une carte est un conteneur de contenu flexible et extensible. Il comprend des options pour les en-têtes et les pieds de page, une grande variété de contenus, des couleurs de fond contextuelles et de puissantes options d'affichage. Si vous connaissez bien Bootstrap 3, les cartes remplacent nos vieux panneaux, puits et vignettes. Des fonctionnalités similaires à ces composants sont disponibles sous forme de classes de modification pour les cartes.

Exemples

Les cartes d'exemple sont construites avec le moins de balisage et de styles possible, mais parviennent à offrir une tonne de contrôle et de personnalisation. Construits avec la flexbox, ils offrent un alignement facile et se mélangent bien avec les autres composants du Bootstrap. Ils n'ont pas de marge par défaut, donc utilisez les utilitaires d'espacement si nécessaire.

Vous trouverez ci-dessous un exemple de carte de base avec un contenu mixte et une largeur fixe. Les cartes n'ont pas de largeur fixe au départ, elles rempliront donc naturellement toute la largeur de son élément parent. Il est facile à personnaliser grâce à nos différentes options de taille.”

IX. Remerciements

Je referme ce merveilleux chapitre en adressant mes remerciements aux formateurs Audrey DERLINCOURT et Guy LOUIS pour leur aide et leurs précieux conseils lors de cette formation.

Je remercie également toute l’équipe de l’école de laManu pour tout le travail effectué en amont, particulièrement les dirigeants et encadrants du campus de SOISSONS et NOYON.

Enfin merci à tous mes camarades apprenants pour la joie, la bonne ambiance, les bons moments et surtout l’esprit d’entraide.

IX. Conclusion

Ce projet a été pour moi un réel défi et une expérience très enrichissante car ce fût mon premier projet de E-Commerce.

Cela m’a permis de mettre en pratique l’enseignement reçu durant ses six mois de formation et d’approfondir davantage mes connaissances en développement web puis de me conforter dans mon choix de carrière.

Comme la plupart des projets qui nous tiennent à coeur, le temps nous manque… Les idées et les besoins du clients sont infinis. Cependant, je continuerai à développer mon site en y intégrant d’autres fonctionnalités, comme par exemple la mise en place d’un captcha pour améliorer la sécurité de mon site.