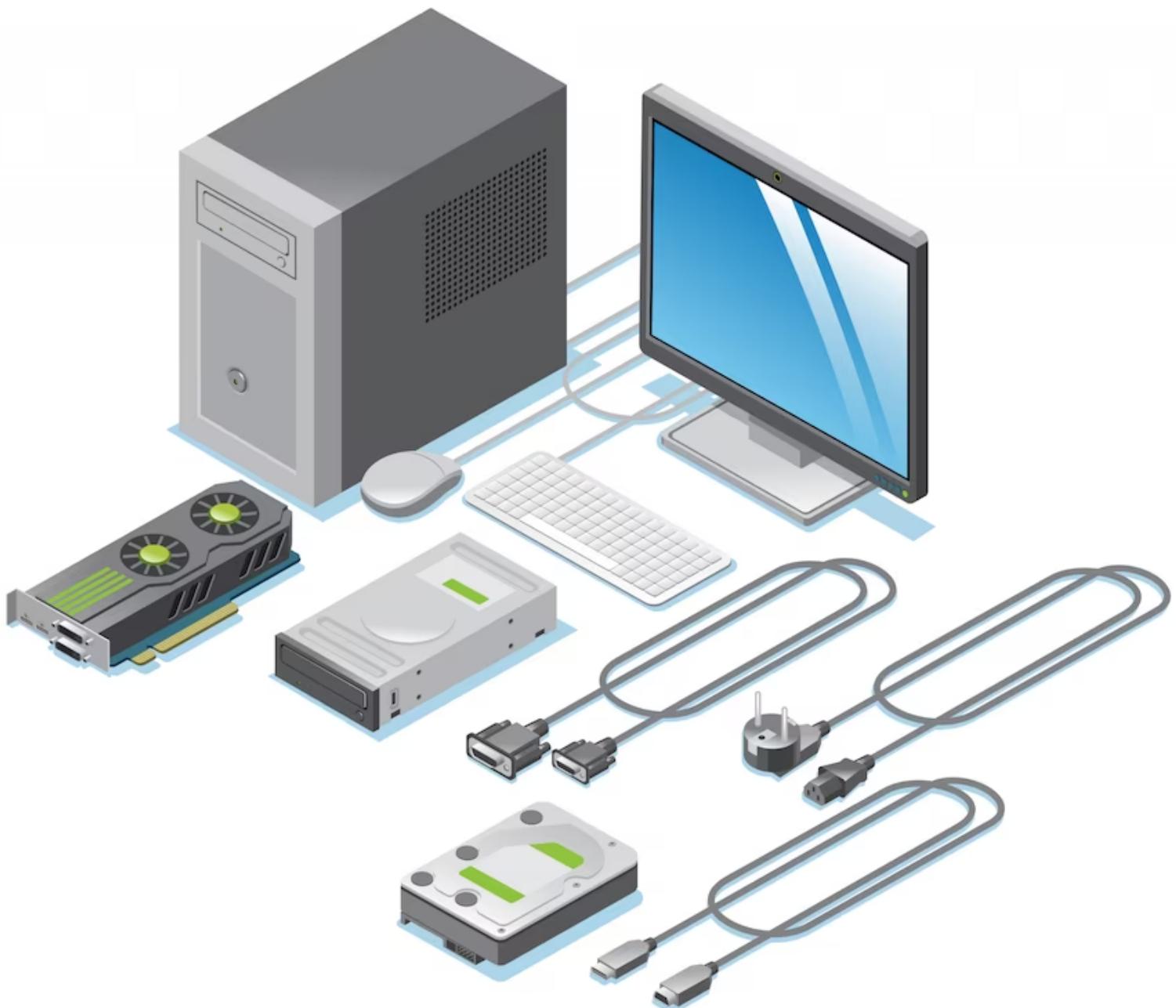


# *Dossier de projet*

---

**Titre : Développeur Web et Web Mobile**



**CALTAGIRONE Charles**

# Tables des matières

<b>Remerciements.....</b>	<b>3</b>
<b>Compétences du référentiel couvertes par le projet.....</b>	<b>4</b>
<b>Résumé.....</b>	<b>4</b>
<b>Spécifications fonctionnelles du projet.....</b>	<b>5</b>
1.    Présentation du projet.....	5
2.    Arborescence du site.....	5
3.    Description des fonctionnalités.....	6
3.1.    Authentification.....	6
3.2.    Accueil.....	6
3.3.    Catalogue produit et filtre.....	6
3.4.    Fiche produit.....	7
3.4.1.    Évaluation des produits et commentaires clients.....	7
3.5.    Panier Client.....	7
3.5.1.    Système de code promo.....	7
3.5.2.    Numéro de commande.....	7
3.6.    Header.....	8
3.6.1.    Fonctionnalité de recherche produit.....	8
3.6.2.    Affichage du nombre d'article dans le panier.....	9
3.6.3.    Affichage des catégories et sous-catégories.....	9
3.7.    Espace Client.....	9
3.7.1.    Modification de profil.....	9
3.7.2.    Gestion des adresses.....	9
3.7.3.    Historique de commandes.....	9
3.8.    Panel Admin.....	10
3.8.1.    Gestion des éléments.....	10
3.8.2.    Statistiques.....	10
<b>Spécifications techniques.....</b>	<b>11</b>
1.    Technologie utilisées.....	11
1.1.    Back-end.....	11
1.2.    Front-end.....	11
1.3.    Outils.....	11
2.    Charte graphique.....	12
2.1.    Logo.....	12
2.2.    Fonts.....	12
2.3.    Code couleurs.....	12
3.    Maquettage.....	13
3.1.    Wireframe.....	13
3.2.    Maquette.....	14
4.    Conception base de donnée.....	15
4.1.    Modèle Conceptuel de Données (MCD).....	15
4.2.    Modèle Logique de Données (MLD).....	16
5.    Jeu d'essai.....	17
5.1.    Autocompletion.....	17
5.2.    Module de Connexion.....	19
5.3.    Produits populaires et nouveaux produits.....	21
5.4.    Tous les produits.....	22
5.5.    Détail.....	26
5.6.    Panier.....	31

5.7.	Profil.....	34
5.8.	Panel Admin.....	37
6.	Responsive.....	40
7.	Sécurité.....	43
7.1.	Htmlspecialchars.....	43
7.2.	Injection SQL.....	43
7.3.	Password Hash.....	44
7.4.	Redirection.....	44
8.	Recherche effectuée depuis un site anglophone.....	45
9.	Axes d'amélioration.....	46
<b>Annexes.....</b>		<b>47</b>
	Maquette.....	47
	Github.....	47
	Boutique.....	47
	Portfolio.....	47
	MCD.....	48
	MLD.....	48

## Remerciements

Je tiens à exprimer ma profonde gratitude pour les projets pertinents auxquels j'ai participé au sein du centre de formation "La Plateforme". Ils ont été essentiels pour ma montée en compétence et ont renforcé mes connaissances dans le domaine du développement informatique.

Ces projets, soigneusement conçus, m'ont offert de bonnes opportunités de les mettre en pratique et de relever des défis stimulants. Grâce à eux, j'ai pu développer une vision plus complète et affiner mes compétences techniques.

Je suis enthousiaste à l'idée de continuer à appliquer les enseignements de ces projets à l'avenir. Ils ont joué un rôle déterminant dans ma progression professionnelle et m'ont donné la confiance nécessaire pour affronter de nouveaux défis.

Je tiens encore à remercier sincèrement le centre de formation pour cette expérience enrichissante.

# Compétences du référentiel couvertes par le projet

Le projet couvre les compétences énoncées ci-dessous.

Pour l'activité 1, “**Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité**”:

- Maquetter une application
- Réaliser une interface utilisateur web ou mobile statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Pour l'activité 2, “**Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité**”:

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

## Résumé

Pour ce projet, j'ai réalisé une boutique en ligne axée sur la vente de produits hardware tels que des processeurs, des cartes graphiques, ainsi que des accessoires PC tels que des chaises et des casques VR.

Le site comprend une page d'accueil présentant les produits les plus populaires et les nouveautés. J'ai mis en place un système de connexion asynchrone, permettant aux utilisateurs de se connecter facilement. De plus, j'ai créé une page de profil où les utilisateurs peuvent modifier leurs informations personnelles et consulter leur historique de commandes.

Une autre fonctionnalité importante est la page qui répertorie tous les produits, avec la possibilité de les trier par catégories et sous-catégories, ainsi que d'autres critères tels que le prix ou l'ordre alphabétique.

J'ai également créé une page dynamique pour chaque produit, affichant sa description, son prix et sa note. Les utilisateurs peuvent ajouter des produits à leur panier, laisser des commentaires et noter les produits. De plus, ils peuvent également répondre aux commentaires existants.

Pour les administrateurs, j'ai développé un panneau d'administration qui leur permet

d'ajouter, de modifier ou de supprimer des catégories, des produits et des utilisateurs.

Le site dispose également d'une barre de recherche avec un système d'autocomplétion intégré dans l'en-tête qui permet de trouver facilement un produit.

Enfin, j'ai veillé à ce que le site soit responsive, c'est-à-dire qu'il s'adapte aux différents formats d'écran tels que le 1920x1080, les ordinateurs de bureau, les tablettes et les smartphones.

## Spécifications fonctionnelles du projet

### 1. Présentation du projet

Il nous a été demandé de réaliser un site e-commerce, en choisissant son thème et ses produits à proposer, avec différentes exigences :

- Maquettage du site
- Réaliser une conception de base de données
- Programmation Orienté Objet
- Site responsive
- Le module d'inscription et de connexion en asynchrone
- Un tableau de bord pour l'administrateur
- Un système de validation de panier
- Des filtres par catégories sans recharge de page
- Une autocomplétion asynchrone

Ce projet devait être réalisé en groupe et sur un délai de 28 jours.

### 2. Arborescence du site

L'arborescence du site se décline comme suit :

- ❖ *Page d'accueil*
- ❖ *Page connexion*
- ❖ *Page inscription*
- ❖ *Page tous les produits*
- ❖ *Page produit*
- ❖ *Page panier*
- ❖ *Page profil*
- ❖ *Page administration*

Une partie back-office est également prévue afin de permettre la gestion du site.

### 3. Description des fonctionnalités

#### 3.1. Authentification

L'utilisateur à la possibilité de s'inscrire ou de se connecter s'il a déjà créé son compte.  
Pour l'inscription il doit remplir un formulaire avec plusieurs champs :

- Email
- Prénom
- Nom
- Mot de passe
- Confirmation du mot de passe

Il peut afficher ou masquer son mot de passe grâce à un bouton.

Une fois rempli, l'utilisateur doit appuyer sur le bouton "Valider", les champs sont alors vérifiés pour correspondre au format demandé, comme par exemple :

Le prénom et le nom ne doivent pas contenir de chiffres ou encore que les champs mot de passe et confirmation du mot de passe sont identiques.

De même pour la connexion, l'utilisateur doit remplir plusieurs champs :

- Email
- Mot de passe

Après avoir appuyé sur le bouton "Valider", les informations rentrées sont vérifiées pour voir s'ils correspondent à un utilisateur existant.

Si oui, il est alors connecté, sinon un message d'erreur apparaît.

#### 3.2. Accueil

Sur la page d'accueil, deux sections sont affichées, une qui permet de voir les produits de la boutique les plus vendus et l'autre les dernières nouveautés.

#### 3.3. Catalogue produit et filtre

Tous les produits sont affichés par défaut sur cette page avec leurs informations comme le prix, le stock, leur image ou encore leur description. Ensuite, l'utilisateur peut les trier par catégorie, sous-catégorie, mais aussi par ordre alphabétique, prix croissant, popularité ou encore nouveauté.

### 3.4. Fiche produit

Toutes les informations du produit sont affichées sur cette page :

- Son image
- Son nom
- Sa description
- Son stock
- Son prix
- Sa note moyenne
- Les commentaires
- Les réponses

Mais aussi un bouton qui permet à l'utilisateur d'ajouter ce produit à son panier.

#### 3.4.1. Évaluation des produits et commentaires clients

L'utilisateur, s'il est connecté, a la possibilité de poster un commentaire (de 2000 caractères maximum) avec une note allant de 1 à 5 étoiles.

Mais aussi de pouvoir répondre à des commentaires postés par d'autres utilisateurs.

Enfin, s'il le souhaite, il peut supprimer ses propres commentaires et/ou ses réponses.

### 3.5. Panier Client

L'utilisateur devra être en mesure de sélectionner un produit et de le mettre dans son panier dans le but final de passer une commande sur le site.

Il a la possibilité d'y ajouter des produits en étant connecté ou non.

Le panier affiche tous les produits ajoutés par l'utilisateur avec leurs informations :

- Son image
- Son nom
- Son prix
- Son stock
- La quantité demandée par l'utilisateur

Le client est en mesure d'augmenter ou de diminuer la quantité demandée grâce à des boutons  et .

Il aura également la possibilité de supprimer un article du panier ainsi que l'intégralité du panier.

Le prix HT, la TVA et le prix TTC du panier sont affichés à côté.  
Enfin l'utilisateur doit sélectionner une adresse qu'il aura au préalable enregistrée dans son profil pour valider le panier.

### 3.5.1. Système de code promo

L'utilisateur est en mesure d'ajouter un code promo. Une fois celui-ci entré, les prix HT, TTC et la TVA affichée sont mis à jour.

### 3.5.2. Numéro de commande

Un numéro de commande unique est généré automatiquement à la validation du panier.

## 3.6. Header

Le header contient :

- Le logo
- Une barre de recherche
- Une icône Utilisateur
- Une icône Panier
- La liste des catégories et sous-catégories

Pour l'icône Utilisateur, des liens apparaissent au survol de la souris. Ils changent en fonction du statut de l'utilisateur.

Si l'utilisateur n'est pas connecté :

- Connexion
- Inscription

S'il est connecté :

- Profil
- Déconnexion

Si l'utilisateur est un admin :

- Profil
- Panel Admin
- Déconnexion

### 3.6.1. Fonctionnalité de recherche produit

L'utilisateur a la possibilité de rechercher un produit qu'il souhaite grâce à une barre de recherche avec un système d'auto-complétion.

### **3.6.2. Affichage du nombre d'articles dans le panier**

Un nombre apparaît automatiquement au-dessus de l'icône panier dès que l'utilisateur ajoute un article à son panier.

Il s'incrémente jusqu'à 10 puis, au-delà, un + apparaît à côté.

### **3.6.3. Affichage des catégories et sous-catégories**

Toutes les catégories s'affichent puis, au survol de celles-ci avec la souris, un menu déroulant apparaît avec les sous-catégories associées.

Si l'utilisateur clique sur une catégorie, ou une sous-catégorie, il est redirigé sur la page "Tous les produits" avec le filtre pré-sélectionné de la catégorie choisie.

## **3.7. Espace Client**

Une page "Profil" est accessible à l'utilisateur pour lui permettre de voir et/ou modifier ses informations personnelles.

### **3.7.1. Modification de profil**

L'utilisateur a la possibilité de modifier ses informations personnelles ainsi que son mot de passe.

### **3.7.2. Gestion des adresses**

Il peut également consulter ses adresses enregistrées avec la possibilité, à l'aide de deux boutons, de modifier ou de supprimer l'adresse choisie.

### **3.7.3. Historique de commandes**

Un historique de commandes est affiché sur la page avec les détails de chacune d'entre elles comme :

- la date à laquelle elle a été passée
- le prix total
- le numéro de commande
- les produits commandés avec leur nom, description mais aussi leur quantité et leur prix

## 3.8. Panel Admin

### 3.8.1. Gestion des éléments

Un administrateur a la possibilité de gérer les produits, les catégories ainsi que les utilisateurs.

Sous forme d'un tableau, il peut les consulter, les modifier et les supprimer.

Il peut aussi ajouter des produits, des catégories et des sous-catégories grâce à un formulaire présent sur la page.

Il a également la possibilité de changer le rôle des utilisateurs pour ajouter ou supprimer des modérateurs et des administrateurs.

### 3.8.2. Statistiques

Différentes données lui sont mises à disposition. Ce sont les informations principales concernant le site.

Il peut ainsi contrôler :

- le nombre d'utilisateurs enregistrés
- le nombre de produits répertoriés
- le nombre de commandes effectuées
- le panier moyen de chaque commande
- le chiffre d'affaire total

# Spécifications techniques

## 1. Technologie utilisées

### 1.1. Back-end

 PHP est le langage imposé durant la formation et utilisé pour interagir avec la base de données.

 SQL est le langage utilisé pour gérer et manipuler les bases de données relationnelles.

### 1.2. Front-end

 HTML5 est un langage de description destiné à structurer et à afficher la page web.

 CSS3 est le langage qui définit le style et met en forme le contenu.

 JavaScript est le langage utilisé pour ajouter des fonctionnalités interactives et dynamiques aux pages web, en permettant la manipulation du contenu et de l'interactivité côté client.

### 1.3. Outils

 Virtual Studio Code est l'IDE que j'ai choisi car, d'une part, il est gratuit et populaire, et également qu'il propose un large choix d'extensions qui peuvent faciliter la programmation.

 Git est un système de contrôle de version qui permet de suivre et de gérer les modifications du code source d'un projet, facilitant ainsi le développement collaboratif et la gestion des différentes versions du logiciel.

 GitHub est une plateforme qui facilite l'hébergement de projets, tout en proposant une interface graphique intuitive pour simplifier la gestion et la collaboration.



MySQL et MariaDB sont des bases de données relationnelles pour stocker, gérer et récupérer efficacement les données dans les applications web et logicielles.



WampServer et XAMPP nous permettent d'héberger localement le projet et ainsi de le développer et de le tester.



Plesk nous permet d'héberger notre projet en ligne et de façon sécurisée.

## 2. Charte graphique

### 2.1. Logo

Il a été réalisé à partir du logiciel de conception graphique Canva.  
C'est un outil en ligne gratuit et très complet pour faciliter l'élaboration du logo.



### 2.2. Fonts

La police choisie est "Open Sans", importée depuis le catalogue de Google Fonts proposant du contenu gratuit.

Ceci est un commentaire

### 2.3. Codes couleur

La couleur principale, que l'on retrouve en fond des différentes sections est celle-ci :

#1d273d

#232b41

#262626

#2f374c

#444b5d

#4b5d8d

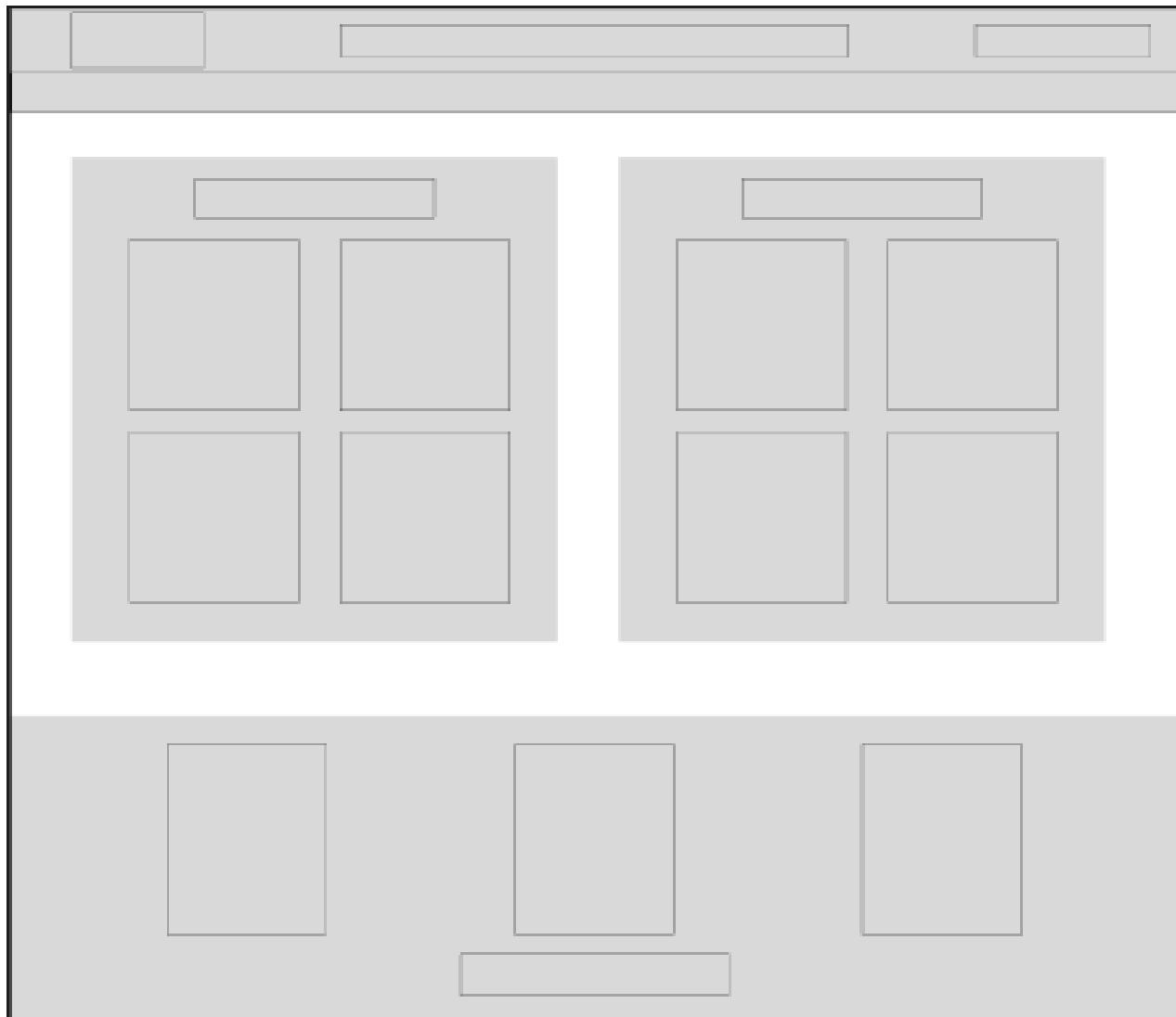
Pour le reste, que ce soit le header et sa navbar, le body, les champs de formulaire, etc..., voici les différentes couleurs utilisées :

### 3. Maquettage

L'outil utilisé pour cela a été Figma. Il est accessible depuis le navigateur directement et permet de travailler en collaboration en temps réel. Gratuit, pratique et complet pour travailler rapidement et efficacement.

#### 3.1. Wireframe

Nous réalisons dans un premier temps le wireframe qui est un schéma de la structure du site. Il n'a pas pour but de présenter le produit de façon aboutie mais plutôt comme une première approche afin de visualiser la mise en page.



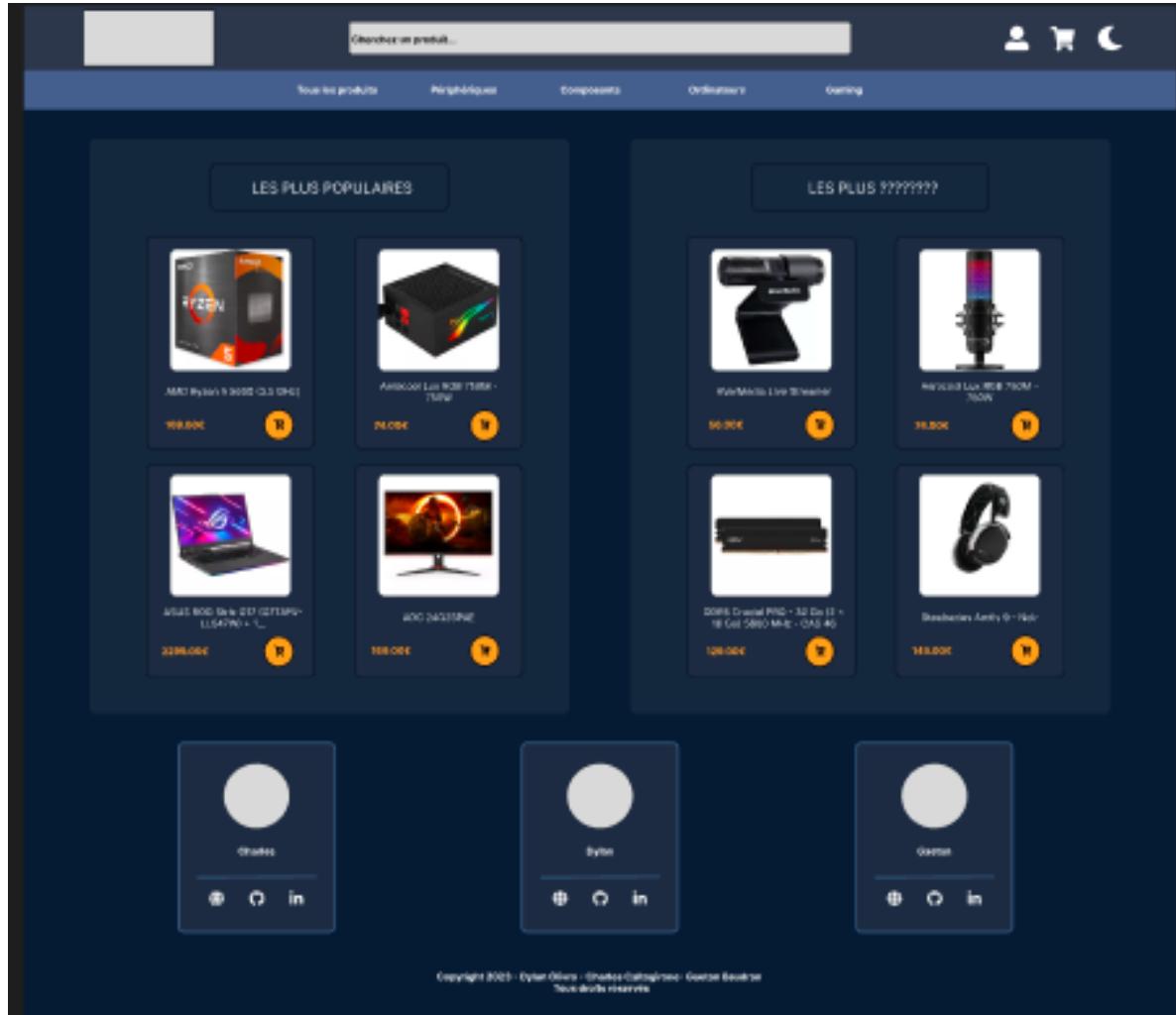
*wireframe de la page index*

### 3.2. Maquette

Vient ensuite la maquette qui nous présente le rendu final du site de façon bien plus précise.

On y voit alors les icônes choisies, du contenu concret et les codes couleurs qui seront utilisés par la suite.

Vous pouvez retrouver l'intégralité des maquettes réalisées dans la partie "[Annexes](#)".



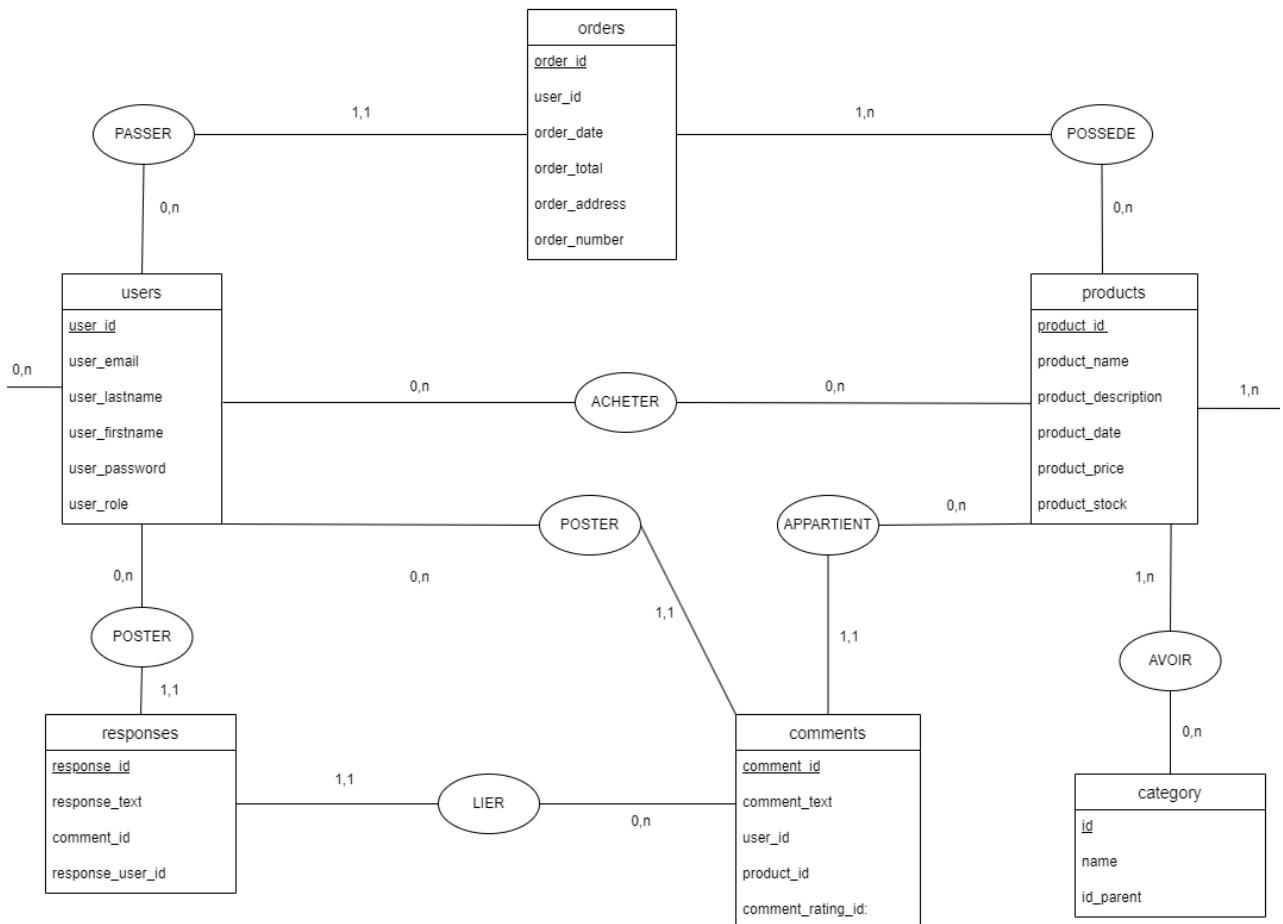
*maquette de la page index*

## 4. Conception base de donnée

Cette étape est très importante car elle permet d'avoir un visuel complet et précis sur notre base de données et à vérifier les relations entre les différentes tables (verbes et cardinalités).

Vous pouvez retrouver l'intégralité du MCD et MLD dans les "[Annexes](#)".

### 4.1. Modèle Conceptuel de Données (MCD)

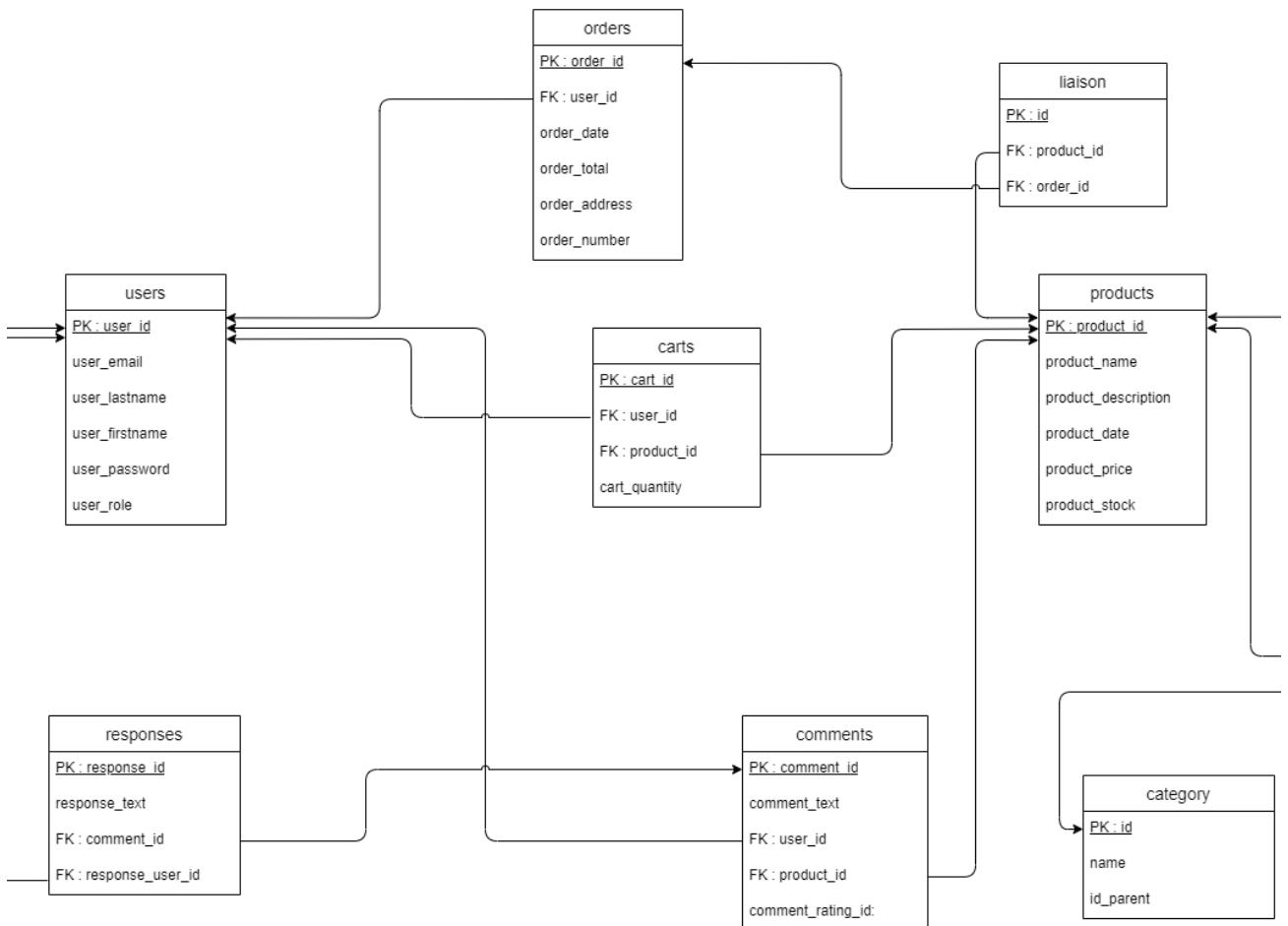


*partie du MCD montrant notamment la relation entre la table "users" et "products"*

On peut y voir par exemple qu'un "users" peut poster 0 à N (plusieurs) "comments" et qu'un "comments" ne peut être posté que par 1 "users".

Dans l'exemple entre la table "users" et "products", on peut distinguer la cardinalité commune "N". Ceci impliquera de créer une table de liaison entre elles, nous le verrons dans le MLD qui suit.

## 4.2. Modèle Logique de Données (MLD)



*partie du MLD montrant notamment la table de liaison entre “products” et “orders”*

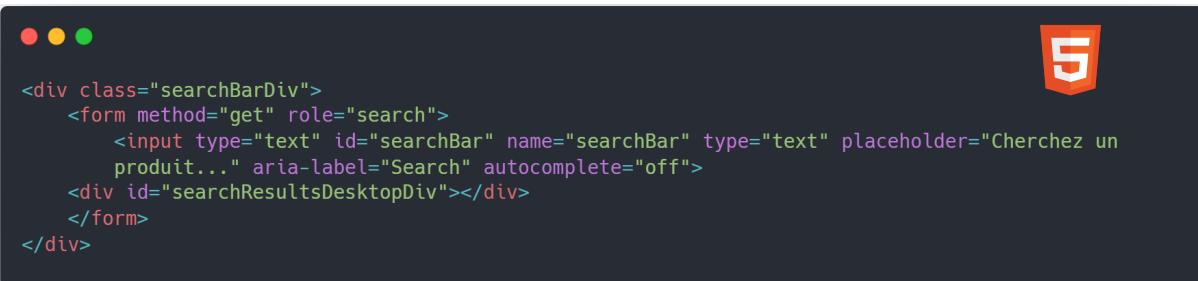
Il n'est plus question des verbes et des cardinalités mais nous pouvons donc voir ici la table de liaison “carts” nécessaire entre “users” et “products”.

On distingue également le lien entre les clés primaires (PK) et les clés étrangères (FK).

## 5. Jeu d'essai

### 5.1. Auto-complétion

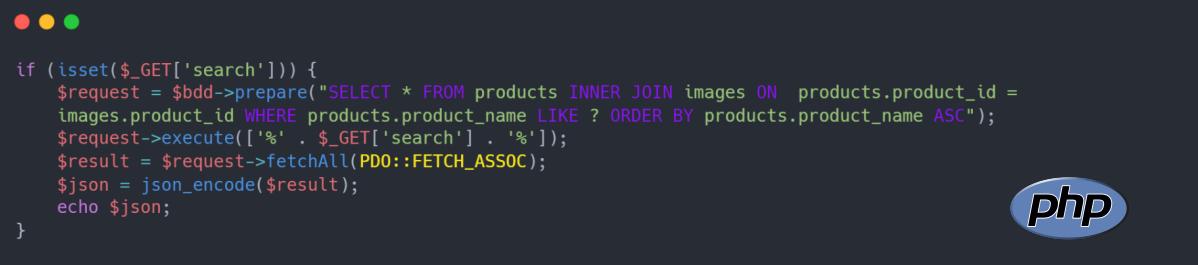
Une barre de recherche avec un système d'auto-complétion qui est composé d'un formulaire avec un **input** de type **text** et d'une **div** vide qui va nous servir à **afficher les résultats** de la recherche de l'utilisateur :



```
<div class="searchBarDiv">
  <form method="get" role="search">
    <input type="text" id="searchBar" name="searchBar" type="text" placeholder="Cherchez un
      produit..." aria-label="Search" autocomplete="off">
    <div id="searchResultsDesktopDiv"></div>
  </form>
</div>
```

Pour ce qui est de la partie Javascript, nous créons un “**event keyup**” qui prend donc en compte chaque tape de l'utilisateur sur son clavier. A chacune d'entre elles, nous allons alors effectuer une **requête fetch** vers la **page de traitement php**.

Celle-ci permet de récupérer les résultats de la base de données en les comparant avec la saisie de l'utilisateur. Que celle-ci corresponde exactement ou qu'en partie, grâce à l'opérateur **LIKE**.



```
if (isset($_GET['search'])) {
  $request = $bdd->prepare("SELECT * FROM products INNER JOIN images ON products.product_id =
  images.product_id WHERE products.product_name LIKE ? ORDER BY products.product_name ASC");
  $request->execute(['%' . $_GET['search'] . '%']);
  $result = $request->fetchAll(PDO::FETCH_ASSOC);
  $json = json_encode($result);
  echo $json;
}
```

Si la saisie correspond donc à des résultats, alors nous créons un ensemble d'éléments, composé de **div**, auxquels nous attribuons un nom de **class**, qui sera utilisé par la suite en CSS.

Nous terminons en intégrant le tout dans la **div finale** avec la fonction “**append()**”.

Nous entourons chaque résultat par un **lien** qui renvoie sur la page “**détail**” du produit.

A noter que l'on limite le nombre de résultats affichés à 5, à l'aide de la propriété **length**, pour ne pas trop encombrer la page.



```

searchResultsInput.addEventListener("keyup", () => {
  searchResultsDesktopDiv.innerHTML = "";
  fetch("./${getPage()}[0]autocomplete.php?search=${searchResultsInput.value}")
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      if (data.length == 0) {
        let noResult = document.createElement("p");
        noResult.innerText = "Aucun résultat trouvé";
        searchResultsDesktopDiv.append(noResult);
      }
      data.forEach((element) => {
        let resultsDiv = document.createElement("div");
        [...]
        let resultsLink = document.createElement("a");

        resultsDiv.className = "resultsDiv";
        [...]
        resultsLink.className = "resultsLink";

        resultsName.innerText = element.product_name;
        [...]

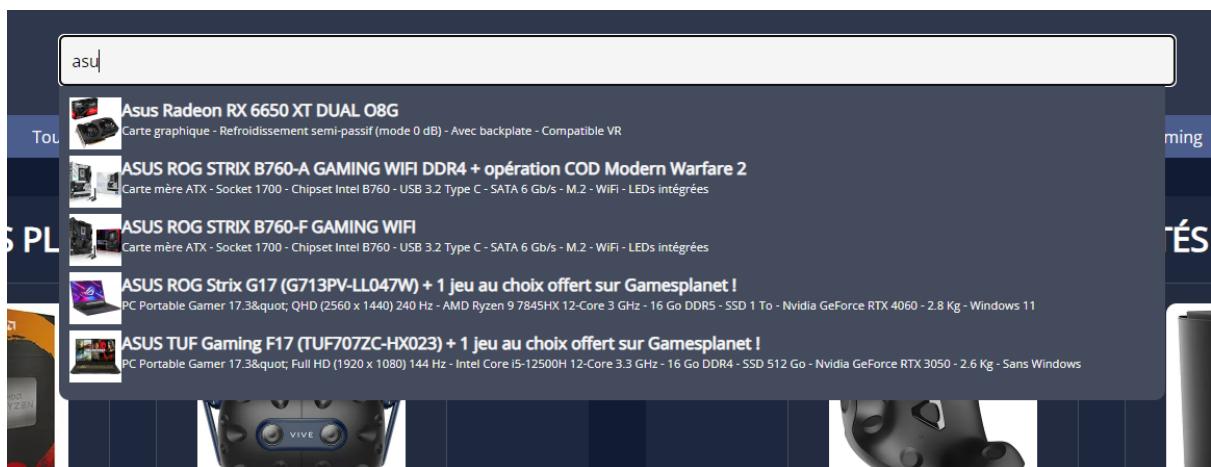
        resultsLink.href = './${getPage()}[0]detail.php?id=${element.product_id}';

        if (searchResultsDesktopDiv.children.length < 5) {
          resultsLink.append(resultsDiv);
          [...]

          searchResultsDesktopDiv.append(resultsLink);
        }
      });
    });
});

```

Nous pouvons voir ici le rendu final avec, pour chaque résultat, l'image du produit, son nom et sa description. Le nombre de résultats est bien limité à 5 comme expliqué précédemment.



*exemple de recherche de produits*

## 5.2. Module de Connexion

Pour l'inscription, nous l'avons fait de façon **asynchrone** en utilisant l'**API fetch** avec la méthode **POST**.

Ceci nous permet notamment d'afficher les messages d'erreur dynamiquement, c'est-à-dire sans rechargeement de page.

Nous créons un nouvel **objet formData** qui récupère la valeur des champs puis nous le convertissons en un **objet javascript** à l'aide de la méthode statique **Object.fromEntries()**.

On envoie ensuite le **body**, au format **JSON**, vers la page de traitement php.

On récupère la **promesse** par le biais du premier **then**, que l'on encode ensuite en **JSON**, puis, à l'aide du second **then**, nous pouvons alors **traiter** les données.



```
const formEl = document.querySelector("#FormRegister");
const message = document.querySelector("#message");

formEl.addEventListener("submit", (event) => {
  event.preventDefault();

  const formData = new FormData(formEl);
  const data = Object.fromEntries(formData);

  fetch("traitement/traitement_register.php", {
    method: "POST",
    headers: {
      "Content-Type": "application/json; charset=UTF-8",
    },
    body: JSON.stringify(data),
  })
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      message.style.color = "";
      if (data.REGISTER_ERROR) {
        message.innerHTML = data.REGISTER_ERROR;
      } else {
        window.location.href = `${getURL()}php/connect.php`;
        message.style.color = "green";
        message.innerHTML = data.REGISTER_SUCCESSION;
        formEl.reset();
      }
    })
    .catch((error) => console.log(error));
});
```

Après avoir reçu les données envoyées dans le corps (body), nous les récupérons puis les décodons afin de pouvoir les manipuler.

Nous créons une instance de la classe "User" en utilisant les différents champs pour initialiser ses propriétés.

Nous faisons différentes vérifications comme :

- Si les champs sont vides
- Si l'e-mail représente une adresse e-mail valide
- Si le nom et le prénom correspondent aux motifs définis par preg\_match
- Si les mots de passe correspondent
- Si l'e-mail n'est pas déjà utilisé

Si il y a une erreur, alors on l'encode en JSON ce qui va permettre au Javascript de l'afficher.

Puis, une fois que tous les champs sont valides, nous utilisons la fonction "**trim()**" pour supprimer les espaces en début et fin de saisie, "**ucfirst()**" pour forcer la majuscule sur la première lettre et "**"password\_hash()**" pour hacher le mot de passe.

Nous ajoutons alors le nouvel utilisateur à notre base de donnée à l'aide de la méthode "**register**" de la **classe User**.

```

● ● ●
$data = json_decode(file_get_contents('php://input'), true);

if (isset($data)) {
    $email = $data['email'];
    ...

    $user = new User(null, $email, $firstname, $lastname, $password, null);

    if (empty($email)) {
        $message['REGISTER_ERROR'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Email est vide.';
    } elseif (...) {
        ...
    } else {
        if ($user->isExist($bdd)) {
            $message['REGISTER_ERROR'] = '<i class="fa-solid fa-circle-exclamation"></i>Cette email est déjà utilisé';
        } else {
            $user->register($bdd);
            $message['REGISTER_SUCCES'] = '<i class="fa-solid fa-circle-check"></i>Données enregistrées avec succès';
        }
    }
} else {
    $message['REGISTER_ERROR'] = "Données manquantes";
}

header('Content-Type: application/json');
echo json_encode($message);
exit;

```



```

● ● ●
class User {

    public function register($bdd)
    {
        $email = trim($this->email);
        $lastname = ucfirst(trim($this->lastname));
        $firstname = ucfirst(trim($this->firstname));
        $password = password_hash(trim($this->password), PASSWORD_DEFAULT);

        $request = $bdd->prepare("INSERT INTO users (user_email,user_lastname,user_firstname,user_password)
                                VALUES (:user_email,:user_lastname,:user_firstname,:user_password)");
        $request->execute([
            'user_email' => $email,
            'user_lastname' => $lastname,
            'user_firstname' => $firstname,
            'user_password' => $password
        ]);
    }
}

```



Pour la connexion nous effectuons les mêmes étapes que l'inscription, si ce n'est la vérification du mot de passe grâce à la fonction “**password\_verify()**”.

Une fois les vérifications passées, nous envoyons les informations de l'utilisateur récupérées précédemment avec la méthode “**returnUserByEmail()**” dans une **SESSION**.

```

● ● ●
$bdd = new PDO('mysql:host=localhost;dbname=mon_projet', 'root', '');

$user = new User(null, $email, null, null, $password, null);

if (empty($email)) {
    $message['CONNECT_ERROR'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Email est vide.';
} elseif (...) { ... }
elseif ($user->isExist($bdd)) {
    $result = $user->returnUserByEmail($bdd);
}
if ($result) {
    if (password_verify($password, $result->user_password)) {
        $_SESSION['user'] = $result;
    }
}

```

```

● ● ●
class User {

    public function isExist($bdd): bool
    {
        $request = $bdd->prepare("SELECT user_email FROM users WHERE user_email = :user_email");
        $request->execute(['user_email' => $this->email]);

        return $request->rowCount() > 0 ? true : false;
    }

    public function returnUserByEmail($bdd)
    {
        $request = $bdd->prepare("SELECT * FROM users WHERE user_email = :user_email");
        $request->execute(['user_email' => $this->email]);
        $result = $request->fetch(PDO::FETCH_OBJ);
        return $result;
    }
}

```

### 5.3. Produits populaires et nouveaux produits

En ce qui concerne la requête PHP des produits populaires, nous utilisons la fonction SQL “**count**” qui nous permet de comptabiliser les produits les plus achetés.

Nous relierons, avec **INNER JOIN**, des tables entre elles afin de récupérer les informations souhaitées.

La commande **GROUP BY** permet de rassembler les résultats.

Aussi, nous ajoutons les commandes **ORDER BY** et **DESC** pour trier les données par nombre décroissant.

Nous limitons l'affichage à 4 résultats avec la commande **LIMIT**.

Concernant la requête PHP pour les nouveaux produits, nous récupérons les plus récents en les triant par leur date de mise en ligne.

```

● ● ●

$request = $bdd->prepare("SELECT *,count(*) FROM liaison_product_order INNER JOIN products
ON liaison_product_order.product_id = products.product_id INNER JOIN images ON images.product_id = products.product_id
WHERE image_main = 1 GROUP BY products.product_id ORDER BY count(*) DESC LIMIT 4");
$request->execute();
$result = $request->fetchAll(PDO::FETCH_OBJ);

$requestAllItems = $bdd->prepare("SELECT * FROM products INNER JOIN images ON products.product_id = images.product_id
WHERE image_main = 1 ORDER BY products.product_date DESC LIMIT 4");
$requestAllItems->execute();
$resultAllItems = $requestAllItems->fetchAll(PDO::FETCH_OBJ);

```

### LES PLUS POPULAIRES



NZXT H5 Flow - Blanc  
109.00€ 



Gainward GeForce RTX 4070 Ghost  
+...  
610.00€ 



Razer Basilisk v3  
69.00€ 



Blue Yeti USB Blackout  
139.00€ 

### NOUVEAUTÉS



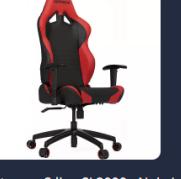
HTC Tracker 3.0  
139.00€ 



HTC Station de base 2.0  
199.00€ 



HTC VIVE PRO 2  
699.00€ 



Vertagear S-line SL2000 - Noir / Rouge  
214.00€ 

*aperçu des sections de la page index*

#### 5.4. Tous les produits

Pour cette page regroupant tous les articles du site, nous affichons la liste des catégories sur la partie latérale.

Pour cela, nous effectuons des requêtes PHP afin de les récupérer depuis la base de données. Une pour les catégories et une seconde pour les sous-catégories, en liant ces dernières à leur catégorie parente par le biais de l'**id** de celle-ci.

Côté visuel, seules les catégories sont affichées dans un premier temps et c'est à l'utilisateur de faire le choix d'en détailler ensuite s'il le souhaite, une seule ou bien plusieurs.

Pour ce faire, nous mettons en place un "**event click**" qui passera alors en "**display block**" les sous-catégories qui lui sont associées, et inversement s'il clique à nouveau en les passant en "**display none**", en utilisant la fonction "**toggle**".

```

● ● ●
$returnCategoryParent = $bdd->prepare('SELECT * FROM category WHERE id_parent = 0');
$returnCategoryParent->execute();
$resultCategoryParent = $returnCategoryParent->fetchAll(PDO::FETCH_OBJ);

foreach ($resultCategoryParent as $key) { ?>
    <div class="categoryParentDiv" data-parent-id=<?= $key->id; ?>>
        <ul>
            <li class="resultParent dropdown-toggle" id=<?= $key->id; ?>><input class="categoryParentRadio" type="radio" name="categoryParentRadio" id=<?= $key->id; ?>>
                <span class="categoryParentName" id=<?= $key->id; ?>><?= $key->name; ?></span>
                <span class="chevronCat" id=<?= $key->id; ?>>
                    <i class="chevronCatIcon fa-solid fa-chevron-down" id="chevronCatIcon"></i></span>
                </li>
            <ul class="categoryChildDiv" id="categoryChildDiv<?= $key->id; ?>" data-parent-id=<?= $key->id; ?>>

<?php
$returnCategoryChild = $bdd->prepare('SELECT * FROM category WHERE id_parent = ?');
$returnCategoryChild->execute([$key->id]);
$resultCategoryChild = $returnCategoryChild->fetchAll(PDO::FETCH_OBJ);
foreach ($resultCategoryChild as $key2) { ?>
            <li class="subCategoryName" id=<?= $key2->id; ?>>
                <input type="radio" name="subCategory" id=<?= $key2->id; ?>>
                <span id=<?= $key2->id; ?>><?= $key2->name; ?></span>
            </li>
        <?php } ?>
        </ul>
    </ul>
</div>
<?php }

```



```

● ● ●
for (let i = 0; i < categoryParentRadio.length; i++) {
    resultParent[i].addEventListener("click", () => {
        chevronCatIcon[i].classList.toggle("fa-chevron-down");
        chevronCatIcon[i].classList.toggle("fa-chevron-up");
        let childElement = document.querySelectorAll(
            "#categoryChildDiv" + categoryParentName[i].getAttribute("id")
        );
        childElement[0].classList.toggle("categoryChildDivBlock");
        allItems.innerHTML = "";
        fetchItems(fetchFilter + '?categoryParent=' + categoryParentRadio[i].id);
    });
}

```



**Périphériques**

- Processeur
- Carte Graphique
- Carte Mère
- Alimentation PC
- RAM
- Stockage
- Refroidissement

**Composants**

- Processeur
- Boitier PC
- Ventilateur boîtier
- Connectique

**Ordinateurs**

Cette liste de catégories représente notre premier système de filtre car, alors que l'utilisateur en sélectionne une, le contenu seul de celle-ci sera affiché.

Pour cela, nous effectuons une requête fetch vers une page de traitement qui nous récupérera tous les articles qui lui sont liés.

Afin de réaliser ces requêtes, qui seront semblables à toutes les catégories, hormis l'id de la celle choisie qui sera alors entré en paramètre, nous avons créé une fonction **"fetchItems(url)"** avec toute la mise en page établie, regroupant alors la création d'éléments, l'attribution d'un nom de classe pour le CSS et pour finir l'intégration totale dans la balise HTML.

```

● ● ●
for (let i = 0; i < categoryChild.length; i++) {
    subCategoryName[i].addEventListener("click", () => {
        allItems.innerHTML = "";
        categoryChild[i].checked = true;
        let urlGetSplitCategorie = urlGet.split "?";
        let urlGetCategorie = urlGetSplitCategorie[0];

        history.pushState({}, "", urlGetCategorie + "?subCategory=" + categoryChild[i].id);

        fetchItems(fetchFilter + '?subCategory=' + categoryChild[i].id);
    });
}

```



```

function fetchItems(url) {
    fetch(url)
        .then((response) => {
            return response.json();
        })
        .then((data) => {
            data.forEach((element) => {
                let divImg = document.createElement("div");
                ...
                let itemStock = document.createElement("p");

                itemName.className = "itemName";
                ...
                itemStock.className = "itemStock";

                itemImg.src = `../assets/img_item/` + element.image_name;
                itemLink.href = `./${getPage()[0]}detail.php?id=${element.product_id}`;

                ...
                itemPrice.innerText = element.product_price + " €";

                checkStock(element.product_stock, itemStock);

                divImg.append(itemImg);
                ...
                allItems.append(li);
            });
        });
    }
}

```

Une autre possibilité d'arriver sur cette page en regroupant déjà les résultats par catégorie et d'en choisir une directement dans la navigation du header. Nous envoyons alors son **id** dans le **GET** et le récupérons pour la requête.

Si aucune catégorie n'est sélectionnée et que nous arrivons donc par le lien général de la page, alors le filtre par défaut choisi pour afficher l'intégralité des produits est celui de la popularité.

```

if (urlGetSplit.length > 1) {
    let urlGetName = urlGetSplit[1].split("=")[0];
    let urlGetId = urlGetSplit[1].split("=")[1];
    if (urlGetName == "subCategory") {
        fetchItems(fetchFilter + `?subCategory=` + urlGetId);
        for (let i = 0; i < categoryChild.length; i++) {
            if (urlGetId == categoryChild[i].id) {
                categoryChild[i].setAttribute("checked", true);
                categoryChild[i].parentElement.parentElement.classList.toggle(
                    "categoryChildDivBlock");
            }
        }
    } else if (urlGetName == "categoryParent") {
        fetchItems(fetchFilter + `?categoryParent=` + urlGetId);
        for (let i = 0; i < categoryParentRadio.length; i++) {
            if (urlGetId == categoryParentRadio[i].id) {
                categoryParentRadio[i].setAttribute("checked", true);
                categoryChildDiv[i].classList.toggle("categoryChildDivBlock");
            }
        }
    } else {
        fetchItems(fetchFilter);
    }
}

```

Deux requêtes différentes sont établies, l'une pour afficher les résultats d'une **catégorie** et l'autre juste pour ceux d'une **sous-catégorie**.

```
● ● ●
if (isset($_GET['categoryParent'])) {
    $request = $bdd->prepare("SELECT * FROM products INNER JOIN liaison_items_category ON
products.product_id = liaison_items_category.id_item INNER JOIN images ON products.product_id =
images.product_id INNER JOIN category ON liaison_items_category.id_category = category.id WHERE
category.id_parent = ? AND image_main = 1 ORDER BY products.product_price DESC");
    $request->execute([$_GET['categoryParent']]);
}

} elseif (isset($_GET['subCategory'])) {
    $request = $bdd->prepare("SELECT * FROM products INNER JOIN images ON products.product_id =
images.product_id INNER JOIN liaison_items_category ON products.product_id =
liaison_items_category.id_item WHERE liaison_items_category.id_category = :id_category
AND image_main = 1 ORDER BY products.product_price DESC");
    $request->execute(['id_category' => $_GET['subCategory']]);
}

}elseif (...){...}
$result = $request->fetchAll(PDO::FETCH_ASSOC);
$json = json_encode($result);
echo $json;
```



Un second système de filtre est proposé, composé de plusieurs choix comme le prix, la nouveauté, etc...

Nous le créons sous forme d'un "**select**" composé de plusieurs "**option**".  
Un "**event change**" est mis en place sur chaque "**option**", réalisant alors une **requête fetch** suivant le filtre choisi.

```
● ● ●
<select name="triSelect" id="triSelect">
    <option value="Popularité">Popularité</option>
    <option value="Nouveauté">Nouveauté</option>
    <option value="Du - cher au + cher">Du - cher au + cher</option>
    <option value="Du + cher au - cher">Du + cher au - cher</option>
    <option value="Alphabétique A-Z">Alphabétique A-Z</option>
    <option value="Alphabétique Z-A">Alphabétique Z-A</option>
    <option value="Disponibilité">Disponibilité</option>
</select>
```



```
● ● ●
$(triSelect).change(function () {
    let triSelected = $("#triSelect option:selected").text();
    if (triSelected) {
        allItems.innerHTML = "";
        switch (triSelected) {
            case "Popularité":
                fetchItems(fetchTri + "?populaire" + urlGetId);
                break;
            case "Nouveauté":
                fetchItems(fetchTri + "?nouveau" + urlGetId);
                break;
            case ...
            default:
                break;
        }
    }
});
```



Les requêtes PHP de la page de traitement sont alors exécutées, comme par exemple ici pour le filtre “Popularité”.

```

if (isset($_GET['populaire'])) {
    if (isset($_GET['subCategory'])) {
        if ($_GET['subCategory'] < 5) {
            $requestAllItems = $bdd->prepare("SELECT *,count(*) FROM liaison_product_order INNER JOIN
liaison_items_category INNER JOIN products ON liaison_product_order.product_id =
products.product_id INNER JOIN images ON products.product_id = images.product_id INNER JOIN
category ON liaison_items_category.id_category = category.id WHERE category.id_parent =
:id_category AND images.image_main = 1 GROUP BY products.product_id ORDER BY count(*) DESC");
        } else {
            $requestAllItems = $bdd->prepare("SELECT *,count(*) FROM liaison_product_order INNER JOIN
products ON liaison_product_order.product_id = products.product_id INNER JOIN
liaison_items_category ON products.product_id = liaison_items_category.id_item INNER JOIN images
ON products.product_id = images.product_id WHERE liaison_items_category.id_category =
:id_category AND images.image_main = 1 GROUP BY products.product_id ORDER BY count(*) DESC");
        }
        $requestAllItems->execute(['id_category' => $_GET['subCategory']]);
    } else {
        $requestAllItems = $bdd->prepare("SELECT *,count(*) FROM liaison_product_order INNER JOIN products ON
liaison_product_order.product_id = products.product_id INNER JOIN images ON products.product_id =
images.product_id WHERE images.image_main = 1 GROUP BY products.product_id ORDER BY count(*) DESC");
        $requestAllItems->execute();
    }
} elseif (isset($_GET['nouveau'])) {...}

```

The screenshot shows a web-based product search interface. At the top right, there is a dropdown menu set to "Popularité". On the left, there are four filter dropdowns: "Périphériques" (Clavier, Souris, Ecran PC, Micro, Casque, Webcam), "Composants", "Ordinateurs", and "Gaming". Below these filters, there are three product cards displayed:

- Gigabyte M28U VRR + 1 jeu au choix offert sur Gamesplanet !**  
Moniteur 28" IPS 144 Hz - HDR 400 - 3840 x 2160 px (Ultra HD 4K) - 1 ms - DisplayPort / HDMI 2.1 (x2) - Pied réglable - Bords extra-fins - Hub USB - Switch KVM intégré  
**589.00 €**  
**EN STOCK**
- Asus TUF VG27AQ Adaptive Sync + 1 jeu au choix offert sur Gamesplanet !**  
Moniteur 27" IPS 165 Hz - HDR - 2560 x 1440 px (QHD) - 1 ms - DisplayPort / HDMI - Pied réglable + Rotation - Bords extra-fins - Compatible G-Sync  
**389.00 €**  
**EN STOCK**
- AOC CQ27G2U/BK Adaptive Sync (dalle incurvée) + 1 jeu au choix offert sur Gamesplanet !**  
Moniteur 27" VA 144 Hz - 2560 x 1440 px (QHD) - 1 ms - DisplayPort / HDMI  
**279.00 €**  
**EN STOCK**

## 5.5. Détail

C'est sur cette page que nous retrouvons tout ce qui concerne un produit, c'est-à-dire son nom, sa description, son image, son prix, son stock, sa note, son nombre d'évaluations. Ce sont ces requêtes qui nous permettent de récupérer ces informations :

```

$returnProduct = $bdd->prepare("SELECT * FROM products
WHERE product_id = :product_id");
$returnProduct->execute(['product_id' => $_GET['id']]);
$result = $returnProduct->fetch(PDO::FETCH_OBJ);

$image = new Image(null, $_GET['id'], null, null);
$result_images = $image->returnImagesByID($bdd);

```

```

class Image {

    public function returnImagesByID($bdd) {

        $recupImage = $bdd->prepare('SELECT * FROM images WHERE
product_id = :product_id');
        $recupImage->execute(['product_id' => $this->product_id]);
        $result = $recupImage->fetchAll(PDO::FETCH_OBJ);
        return $result;
    }
}

```

Dans la suivante, nous pouvons voir la fonction SQL “**AVG**” qui calcule directement la moyenne des notes puis nous l’arrondissons à l’aide de la fonction PHP “**ceil()**”.

```
$requestAverage = $bdd->prepare('SELECT AVG(comment_rating) AS avgComment, COUNT(comment_id)  
AS countRating FROM comments WHERE product_id = :product_id');  
$requestAverage->execute(['product_id' => $_GET['id']]));  
$resultAverageAndCount = $requestAverage->fetch(PDO::FETCH_OBJ);  
$averageComment = ceil($resultAverageAndCount->avgComment);  
$countRating = $resultAverageAndCount->countRating;
```



Nous pouvons désormais afficher ces résultats dans les balises souhaitées comme présenté ci-dessous :

```
<div class="BoxImg">  
      
</div>  
<div class="BoxDetail">  
    <p id="productName"><?= htmlspecialchars($result->product_name) ?></p>  
    <div class="BoxAverageCount">  
        <div class="AvgRating">  
            <input type="radio" id="star5Avg" value="5" disabled <?= $averageComment == 5 ? 'checked' : '' ; ?>  
            <label for="star5Avg" title="text"></label>  
            <input type="radio" id="star4Avg" value="4" disabled <?= $averageComment == 4 ? 'checked' : '' ; ?>  
            <label for="star4Avg" title="text"></label>  
            <input type="radio" id="star3Avg" value="3" disabled <?= $averageComment == 3 ? 'checked' : '' ; ?>  
            <label for="star3Avg" title="text"></label>  
            <input type="radio" id="star2Avg" value="2" disabled <?= $averageComment == 2 ? 'checked' : '' ; ?>  
            <label for="star2Avg" title="text"></label>  
            <input type="radio" id="star1Avg" value="1" disabled <?= $averageComment == 1 ? 'checked' : '' ; ?>  
            <label for="star1Avg" title="text"></label>  
        </div>  
        <span class="countRating">  
            <a href="#BoxCommentResponse"><?= $countRating ?> évaluation<?= $countRating > 1 ? 's' : '' ?></a></span>  
        </div>  
        <div id="description">  
            <p>Description </p>  
            <p><?= htmlspecialchars($result->product_description) ?></p>  
        </div>  
</div>
```



Lorsqu'un utilisateur visite la page d'un produit, il a la possibilité d'ajouter ce produit à son panier en cliquant sur un bouton situé sous le prix.

Une fois qu'il effectue cette action, nous procédons à une vérification pour déterminer si cet article est déjà présent dans le panier.

Si tel est le cas, nous augmentons simplement la quantité du produit existant. Si l'article n'est pas encore présent dans le panier, nous l'ajoutons alors.

Nous avons aussi la possibilité d'ajouter des articles dans notre panier sans être forcément connecté. Pour cela, nous nous servons de la **superglobal SESSION**.

Nous créons des tableaux pour les produits et les informations qui nous sont nécessaires comme leur **id**, leur **prix** et leur **quantité**.

Puis nous mettons à jour la quantité à chaque clic sur le bouton “*add item*”.

L'utilisateur peut consulter le nombre d'articles présents dans son panier en observant l'**icône "Panier"** située dans l'en-tête du site.

Le nombre d'articles affiché est **limité** à 10. Si le panier contient plus de 10 articles, un symbole "**10+**" sera affiché pour éviter toute perturbation de la mise en page.



```
$quantity = $bdd->prepare("SELECT `cart_quantity` FROM `carts` WHERE product_id = :product_id AND user_id = :user_id");
$quantity->execute([
    'product_id' => $result->product_id,
    'user_id' => $_SESSION['user']->user_id
]);
$result_quantity = $quantity->fetch(PDO::FETCH_OBJ);

if ($quantity->rowCount() > 0) {
    $updateQuantity = $bdd->prepare("UPDATE `carts` SET `cart_quantity` = :cart_quantity WHERE product_id = :product_id AND user_id = :user_id");
    $updateQuantity->execute([
        'cart_quantity' => $result_quantity->cart_quantity + 1,
        'product_id' => $result->product_id,
        'user_id' => $_SESSION['user']->user_id
    ]);
} else {
    $insertQuantity = $bdd->prepare("INSERT INTO `carts`(`user_id`, `product_id`, `cart_quantity`)
VALUES (:user_id,:product_id,:cart_quantity)");
    $insertQuantity->execute([
        'user_id' => $_SESSION['user']->user_id,
        'product_id' => $result->product_id,
        'cart_quantity' => 1
    ]);
}
```



```
$select = array();
$select['id'] = $result->product_id;
$select['qte'] = 1;
$select['prix'] = $result->product_price;
if (modif_qte($result->product_id, '+')) {
} else {
    ajout($select);
}
header('Location: detail.php?id=' . $result->product_id);
}
```

```
function ajout($select)
{
    array_push($_SESSION['panier']['id_article'],
    $select['id']);
    array_push($_SESSION['panier']['qte'],
    $select['qte']);
    array_push($_SESSION['panier']['prix'],
    $select['prix']);
}
```



### AMD Ryzen 5 5600 (3.5 GHz)

★★★★★ (2 évaluations)

Description :

Processeur Socket AM4 - Hexa Core - Cache 35 Mo - Vermeer - Ventirad inclus

169.00€

EN STOCK

Add Item +

rendu visuel de la description détaillée du produit

Nous avons aussi un système de commentaire. Si l'utilisateur est connecté, il a alors la possibilité de poster un commentaire ou une réponse à un commentaire existant.

```
● ● ●
```



```
if (isset($_POST['submitComment'])) {
    $comment = $_POST['comment'];
    if (...) ...
    } elseif (mb_strlen(str_replace("\n", '', $comment)) > 2000) {
        $COMMENT_ERROR = '<i class="fa-solid fa-circle-exclamation"></i>&ampnbspCommentaire trop long (2000max).';
    } elseif (...) ...
    } else {
        $addComment = $bdd->prepare('INSERT INTO comments (comment_text, user_id, product_id, comment_rating, comment_date) VALUES(:comment_text, :user_id, :product_id, :comment_rating, :comment_date)');
        $addComment->execute([
            'comment_text' => $_POST['comment'],
            'user_id' => $_SESSION['user']->user_id,
            'product_id' => $result->product_id,
            'comment_rating' => $_POST['rate'],
            'comment_date' => $date
        ]);
        header('Location: detail.php?id=' . $result->product_id);
    }
}
```

```
● ● ●
```



```
$addResponse = $bdd->prepare('INSERT INTO responses (response_text, comment_id, response_user_id, response_date) VALUES(:response_text, :comment_id, :response_user_id, :response_date)');
```

Plusieurs vérifications sont faites avant l'envoi, comme par exemple s'il est vide, s'il fait bien moins de 2000 caractères ou encore s'il a bien rentré une note sur 5.

Comme nous pouvons le voir ici, la limite du nombre de caractères est effectuée en **PHP** et en **JS**.

Pour calculer le nombre de caractères, nous avons mis en place un “**événement keyup**” sur les champs de commentaire et de réponse. Cela nous permet de détecter chaque fois que l'utilisateur appuie sur une touche de son clavier. De plus, nous avons ajouté une condition pour **changer la couleur** du compteur une fois qu'il dépasse 2000 caractères.

```
● ● ●
```



```
let textarea<?= $key->comment_id ?> = document.getElementById("TextareaResponse<?= $key->comment_id ?>");
let count<?= $key->comment_id ?> = document.getElementById("count<?= $key->comment_id ?>");

textarea<?= $key->comment_id ?>.addEventListener("keyup", () => {
    count<?= $key->comment_id ?>.innerText = `${textarea<?= $key->comment_id ?>.value.length}/2000`;
    if (textarea<?= $key->comment_id ?>.value.length > 2000) {
        count<?= $key->comment_id ?>.style.color = "#c7161d";
    } else {
        count<?= $key->comment_id ?>.style.color = "";
    }
});
```



Pour attribuer une note, l'utilisateur doit sélectionner l'un des cinq boutons radio disponibles. Le premier correspond à une note de 1/5, et le dernier représente une note de 5/5.



Les requêtes ci-dessous nous permettent de récupérer les commentaires avec leurs réponses associées.

```
$returnComments = $bdd->prepare('SELECT comments.*,users.user_firstname FROM comments INNER JOIN users ON comments.user_id = users.user_id WHERE product_id = :product_id ORDER BY comments.comment_id DESC');
$returnComments->execute(['product_id' => $result->product_id]);
$result_comments = $returnComments->fetchAll(PDO::FETCH_OBJ);

$returnResponses = $bdd->prepare('SELECT responses.*,users.user_firstname FROM responses INNER JOIN comments ON responses.comment_id = comments.comment_id INNER JOIN users ON responses.response_user_id = users.user_id WHERE product_id = :product_id AND comments.comment_id = :comment_id ORDER BY responses.response_id DESC');
$returnResponses->execute([
    'product_id' => $result->product_id,
    'comment_id' => $key->comment_id
]);
$result_responses = $returnResponses->fetchAll(PDO::FETCH_OBJ);
```



Enfin, l'administrateur, ou l'utilisateur, qui a créé un commentaire ou une réponse remarquera l'apparition d'un bouton supplémentaire.

Ce bouton permet de réaliser une **suppression en cascade** du commentaire avec ses réponses associées, ou simplement d'une réponse individuelle.

```
if (isset($_POST['deleteComment'] . $key->comment_id)) {
    $deleteResponseWithComment = $bdd->prepare('DELETE FROM responses WHERE comment_id = :comment_id');
    $deleteResponseWithComment->execute(['comment_id' => $key->comment_id]);

    $deleteComment = $bdd->prepare('DELETE FROM comments WHERE comment_id = :comment_id');
    $deleteComment->execute(['comment_id' => $key->comment_id]);

    header('Location: detail.php?id=' . $result->product_id);
}
```



The screenshot shows a comment form and its corresponding response section. At the top, there is a rating scale from 1 to 5 stars. Below it is a text input field labeled "Écrire un commentaire..." with a character count of 0/2000. A blue "Envoyer" button is located below the input field. In the response section, there is a comment by "Admin" from July 18, 2023, at 14:33:53. The comment text is "Ceci est un commentaire". There is a "Répondre" link. Below the comment, there is a reply by "Admin" from July 18, 2023, at 14:33:53. The reply text is "Ceci est une réponse". There is also a "Supprimer votre réponse" link.

## 5.6. Panier

Pour obtenir le contenu du panier de l'utilisateur, nous créons une instance de la classe "**Cart**" et utilisons la méthode "**returnCart()**". Pour récupérer son ou ses adresses, nous utilisons la classe "**Address**" et sa méthode "**returnAddressesByUser()**".

```
$cart = new Cart(null, $_SESSION['user']->user_id, null, null);
$result_cart = $cart->returnCart($bdd);

$stock = [];
foreach ($result_cart as $product) {
    array_push($stock, $product->product_stock);
}

$address = new Address(null, $_SESSION['user']->user_id, null, null, null, null, null, null, null);
$allUserAddresses = $address->returnAddressesByUser($bdd);
```



```
class Cart {
    public function returnCart($bdd) {
        $returnCart = $bdd->prepare("SELECT * from carts INNER JOIN products ON carts.product_id = products.product_id INNER JOIN images ON products.product_id = images.product_id WHERE user_id = :user_id AND image_main = 1");
        $returnCart->execute(['user_id' => $this->user_id]);
        $result = $returnCart->fetchAll(PDO::FETCH_OBJ);
        return $result;
    }
}
class Address {
    public function returnAddressesByUser($bdd)
    {
        $returnAddress = $bdd->prepare('SELECT * FROM addresses WHERE user_id = :user_id');
        $returnAddress->execute(['user_id' => $this->user_id]);
        $result = $returnAddress->fetchAll(PDO::FETCH_OBJ);
        return $result;
    }
}
```



Nous pouvons désormais afficher les informations souhaitées comme le nom, le prix, la quantité, etc...

```
foreach ($result_cart as $product) { ?>
<div class="cartDetail">
    <div class="cartProduct">
        <div class="cartImage">
            
        </div>
        <div class="cartInfo">
            <a href="./detail.php?id=<?= $product->product_id ?>">
                <p class="name"><?= htmlspecialchars(CoupePhrase($product->product_name)) ?></p>
            </a>
            <p class="stock"><?= htmlspecialchars($product->product_stock) ?></p>
        </div>
    </div>
    <p class="price"><?= htmlspecialchars($product->product_price) ?>€</p>
    <p class="quantity"><?= htmlspecialchars($product->cart_quantity) ?></p>
    <form action="" method="post">
        <button type="submit" name="add<?= $product->cart_id ?>" id="add" style="background-color: transparent">
            <i class="fa-solid fa-plus"></i>
        </button>
        <button type="submit" name="delete<?= $product->cart_id ?>" id="delete">
            <?= $product->cart_quantity <= 1 ? '<i class="fa-solid fa-xmark"></i>' : '<i class="fa-solid fa-minus"></i>' ?>
        </button>
    </form>
</div>
```



En cliquant sur les boutons "+" et "-", l'utilisateur peut ajuster la quantité du produit directement. Chaque modification entraîne une requête "UPDATE" dans la base de données pour mettre à jour la quantité du produit.

Lorsqu'il ne reste plus qu'un seul exemplaire du produit, le bouton "-" se transforme en "×". Dans cette situation, une requête "DELETE" est utilisée pour retirer le produit du panier.

Par ailleurs, l'utilisateur a également la possibilité de vider entièrement son panier en un seul clic grâce à un bouton représentant une corbeille.

```
● ● ●

if (isset($_POST['add'] . $product->cart_id))) {
    $requestUpdateCart = $bdd->prepare("UPDATE `carts` SET `cart_quantity` = :cart_quantity WHERE product_id = :product_id");
    $requestUpdateCart->execute([
        'cart_quantity' => $product->cart_quantity + 1,
        'product_id' => $product->product_id
    ]);
    echo '<i class="fa-solid fa-circle-check" style="color: #0cad00;"></i> Article ajouté au panier.';
    header('Location: cartPage.php');
}
if (isset($_POST['delete'] . $product->cart_id))) {
    if ((int)$product->cart_quantity > 1) {
        $requestUpdateCart = $bdd->prepare("UPDATE `carts` SET `cart_quantity` = :cart_quantity WHERE user_id = :user_id AND product_id = :product_id");
        $requestUpdateCart->execute([
            'cart_quantity' => $product->cart_quantity - 1,
            'user_id' => $_SESSION['user']->user_id,
            'product_id' => $product->product_id
        ]);
    } elseif ((int)$product->cart_quantity == 1) {
        $requestDeleteCart = $bdd->prepare("DELETE FROM `carts` WHERE user_id = :user_id AND product_id = :product_id ");
        $requestDeleteCart->execute([
            'user_id' => $_SESSION['user']->user_id,
            'product_id' => $product->product_id
        ]);
    }
    header('Location: cartPage.php');
}
```



Les prix de tous les produits sont enregistrés dans un tableau à l'aide d'une boucle et de la fonction "**array\_push()**".

Ensuite, pour obtenir la somme totale des prix, nous utilisons la fonction "**array\_sum()**".

Nous procédons ensuite au calcul de la TVA et du prix HT en utilisant notre fonction "**returnAmountTVA()**" et "**returnPriceHT()**".

```

$prices = [];
foreach ($result_cart as $cartProduct) {
    if ($cartProduct->cart_quantity > 1) {
        for ($i = 1; $i < (int)$cartProduct->cart_quantity; $i++) {
            array_push($prices, $cartProduct->product_price);
        }
    }
    array_push($prices, $cartProduct->product_price);
}

$total = array_sum($prices);

```

```

function returnPriceHT(float $priceTTC)
{
    $tva = 20 / 100;
    $priceHT = $priceTTC / (1 + $tva);
    $roundPriceHT = number_format($priceHT, 2, '.', '');
    return (float)$roundPriceHT;
}
function returnAmountTVA(float $priceTTC, float $priceHT)
{
    $amountTVA = $priceTTC - $priceHT;
    $roundAmountTVA = number_format($amountTVA, 2, '.', '');
    return (float)$roundAmountTVA;
}

```

On affiche alors le prix total du panier, hors taxe, puis le montant seul que représente la TVA, et enfin le prix total, toutes taxes comprises :

```

<p>HT : <?= returnPriceHT($total) ?>€</p>
<p>TVA : <?= returnAmountTVA($total, returnPriceHT($total)) ?>€</p>
<hr>
<p class="priceTotal">TTC : <?= number_format($total, 2) ?>€</p>

```

L'utilisateur a la possibilité d'ajouter un **code promo** qui lui accordera une réduction en **pourcentage** en fonction du code saisi.

Une fois le code promo appliqué, les prix affichés seront automatiquement mis à jour pour refléter la réduction appliquée.

```

$returnCode = $bdd->prepare('SELECT * FROM codes WHERE code_name = :code_name');
$returnCode->execute(['code_name' => $_POST['code']]);
$result_code = $returnCode->fetch(PDO::FETCH_OBJ);

if ($result_code) {
    $discount = (intval($result_code->code_discount) * $total) / 100;
    $total = $total - $discount;
    $CODE_MESSAGE = sprintf('Code %s appliqué (%d%%)', $result_code->code_name, $result_code->code_discount);
    $_SESSION['code'] = $result_code->code_discount;
} else {
    $CODE_MESSAGE = 'Code promo invalide';
    unset($_SESSION['code']);
}

```

Nous commençons par créer une instance de la **classe "Order"** et initialiser deux de ses propriétés. Ensuite, nous ajoutons la commande à notre base de données en utilisant la méthode **"addOrder()"** et récupérons son identifiant unique grâce à la fonction PHP **"lastInsertId()"**.

Les prix de tous les produits de la commande sont ensuite stockés dans un tableau pour calculer le montant final.

Par la suite, nous mettons à jour le stock des produits commandés en effectuant une requête "**UPDATE**", et nous insérons tous les produits dans une table de liaison pour permettre un suivi de l'historique de commande.

Enfin, nous mettons à jour l'**objet "Order"**, instancié précédemment, avec les nouvelles données disponibles en utilisant la méthode "**updateOrder()**" :

- L'id
- Le montant total
- L'adresse de livraison
- Le numéro de commande

Puis nous effaçons tout ce que contient le panier en utilisant la méthode "**deleteCart()**".



```
$date = date("Y-m-d H:i:s");
$order = new Order(null, $_SESSION['user']->user_id, $date, null, null, null);
$order->addOrder($bdd);
$lastInsertId = $bdd->lastInsertId();
// Récupère les prix dans un tableau
[...]

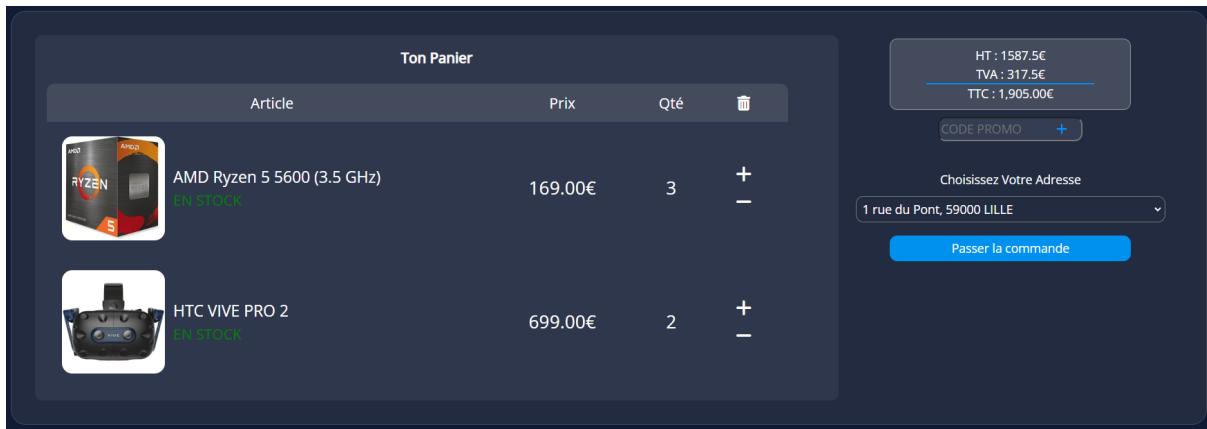
$updateStock = $bdd->prepare('UPDATE products SET product_stock = :product_stock WHERE product_id = :product_id');
$updateStock->execute([
    'product_stock' => $cartProduct->product_stock - (int)$cartProduct->cart_quantity,
    'product_id' => $cartProduct->product_id
]);

$insertLiaison = $bdd->prepare('INSERT INTO liaison_product_order (order_id,product_id, product_quantity) VALUES
                                (:order_id,:product_id,:product_quantity)');
$insertLiaison->execute([
    'order_id' => $lastInsertId,
    'product_id' => $cartProduct->product_id,
    'product_quantity' => $cartProduct->cart_quantity
]);
// Code promo
[...]
// Numéro de commande
[...]

$order->setId($lastInsertId);
$order->setTotal($total);
$order->setAddress($_POST['address']);
$order->setNumber($orderNumber);

$order->updateOrder($bdd);
$cart->deleteCart($bdd);
header('Location: cartPage.php');
```

Si l'utilisateur n'était pas connecté lorsqu'il a créé son panier, en se connectant ultérieurement, son panier sera **automatiquement mis à jour** avec le contenu du panier enregistré dans la **SESSION**.



*rendu visuel du panier*

## 5.7. Profil

L'utilisateur a la possibilité de consulter et modifier ses informations personnelles, que sont son nom, son prénom, son email et son mot de passe. Chaque élément peut être modifié individuellement s'il le souhaite.

```

class User {
    public function update($bdd, $password_bdd)
    {
        $email = trim($this->email);
        ...

        $request = $bdd->prepare("UPDATE users SET user_email = :user_email, user_firstname = :user_firstname,
        user_lastname = :user_lastname, user_password = :user_password WHERE user_id = :user_id ");
        $request->execute([
            'user_email' => $email,
            'user_firstname' => $firstname,
            'user_lastname' => $lastname,
            'user_password' => $password_bdd,
            'user_id' => $this->id
        ]);

        $_SESSION['user']->user_email = $email;
        $_SESSION['user']->user_firstname = $firstname;
        $_SESSION['user']->user_lastname = $lastname;
        $_SESSION['user']->user_password = $password;
    }
}

```

Par ailleurs, ses adresses sont également répertoriées et affichées. Il est important de noter que seules six adresses peuvent être enregistrées simultanément. Si l'utilisateur souhaite ajouter une nouvelle adresse, il doit préalablement en supprimer une en utilisant le bouton de corbeille.

```

● ● ●

$address = new Address(null, $_SESSION['user']->user_id, null, null, null, null, null, null, null);
$allUserAdresses = $address->returnAddressesByUser($bdd);

if (empty($numero)) {
    $INSERT_ADDRESS_ERROR = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Numero est vide.';
} elseif (...) {...}
elseif (count($allUserAddresses) >= 6) {
    $INSERT_ADDRESS_ERROR = '<i class="fa-solid fa-circle-exclamation"></i>Nombres maximum d\'adresse atteint (6).';
} else {
    $tel = $address->returnFormatTel($telephone);

    $address->setNumero($numero);
    $address->setName($name);
    $address->setPostcode($postcode);
    $address->setCity($city);
    $address->setFirstname($prenom);
    $address->setLastname($nom);
    $address->setTelephone($tel);
    $address->addAddress($bdd);
    header('Location: ../profil.php');
}

}


```



En outre, il est possible de simplement modifier une adresse existante en utilisant le bouton de crayon, sans nécessairement la supprimer.

```

● ● ●

$updateAddress = $bdd->prepare('UPDATE addresses SET address_numero = :address_numero, address_name = :address_name,
address_postcode = :address_postcode, address_city = :address_city, address_telephone = :address_telephone,
address_firstname = :address_firstname, address_lastname = :address_lastname WHERE address_id = :address_id AND
user_id = :user_id');


```

**Modifier ces infos personnelles**

Email  
charles@gmail.com

Prénom  
charles

Nom  
charles

Mot de passe  
Mot de passe 

**Enregistrer**

[Changer de mot de passe](#)

**Adresses enregistrées**

Nombres d'adresses maximum	
Robert Pires	 
1 rue rouge JESAISPAS, 59856 France 05 05 56 06 86	
Charlot Du métro	 
3 numéro et rue test TOULON, 83000 France 06 06 06 06 06	
Test nom Test prenom	 
1 test adresse TEST VILLE, 12323 France 06 66 66 66 66	
Encore tet	 
Encore tet	

L'utilisateur a la possibilité de consulter son historique de commandes, où différentes informations sont affichées. Pour obtenir ces informations, nous effectuons une requête vers deux tables de la base de données.

La première est la table "**orders**", qui nous fournit la date de la commande, le numéro de commande, l'adresse de livraison et le prix total.

```

$bdd = new PDO('mysql:host=localhost;dbname=ecommerce;charset=utf8', 'root', '');
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$order = new Order(null, $_SESSION['user']->user_id, null, null, null, null);
$allUserOrders = $order->returnOrdersByUser($bdd);

class Order {
    public function returnOrdersByUser($bdd)
    {
        $request = $bdd->prepare('SELECT * FROM orders WHERE user_id = :user_id ORDER BY order_date DESC LIMIT 5');
        $request->execute(['user_id' => $this->user_id]);
        $result = $request->fetchAll(PDO::FETCH_OBJ);
        return $result;
    }
}

```

La seconde est la table "**liaison\_product\_order**", qui nous permet de récupérer les produits associés à chaque commande, ainsi que leur quantité respective. De cette manière, l'utilisateur peut obtenir une vue complète de chaque commande passée et des produits qui y étaient inclus, accompagnés de leurs quantités.

```

$order->setId($userOrder->order_id);
$product = $order->returnContentOrder($bdd);

class Order {
    public function returnContentOrder($bdd) {
        $request = $bdd->prepare('SELECT * FROM orders INNER JOIN liaison_product_order ON orders.order_id = liaison_product_order.order_id INNER JOIN products ON liaison_product_order.product_id = products.product_id INNER JOIN images ON products.product_id = images.product_id WHERE user_id = :user_id AND orders.order_id = :order_id AND image_main = 1');
        $request->execute([
            'user_id' => $this->user_id,
            'order_id' => $this->id
        ]);
        $result = $request->fetchAll(PDO::FETCH_OBJ);
        return $result;
    }
}

```

COMMANDE EFFECTUÉE LE :		TOTAL :	NUMERO DE COMMANDE :
2023-07-16 00:02:54		725.00€	64B3178E807B00-01460190
	AMD Ryzen 5 5600 (3.5 GHz) 1352 € (169.00 x8)		
	Razer Ornata V3 X (AZERTY) 98 € (49.00 x2)		
COMMANDE EFFECTUÉE LE :		TOTAL :	NUMERO DE COMMANDE :
2023-07-11 14:50:47		1179.50€	64AD50271A7F81-39063846
	AMD Ryzen 5 5600 (3.5 GHz) 1521 € (169.00 x9)		
	HTC VIVE PRO 2 699 € (699.00 x1)		
	HTC Tracker 3.0 139 € (139.00 x1)		

## 5.8. Panel Admin

Cette page est réservée **exclusivement** à l'administrateur, qui bénéficie d'une vue d'ensemble sur le site, incluant de nombreuses informations.

L'administrateur a une vision directe du nombre d'utilisateurs inscrits, du nombre de produits répertoriés, du nombre de commandes effectuées, du panier moyen des clients, ainsi que du chiffre d'affaires de la société.



Pour cela nous avons effectué plusieurs **requêtes fetch** :

The figure displays two code snippets from browser developer tools. The top snippet is in PHP, showing a conditional block that prepares different SQL queries based on GET parameters. The bottom snippet is in JavaScript, showing a function that uses the Fetch API to send requests to 'traitement/traitement\_stats.php' with different table names ('product', 'user', 'order', 'orderAverage', 'salesRevenues'). It then processes the responses to update a 'countDiv' element with the appropriate count or average values.

```
if (isset($_GET['product'])) {
    $request = $bdd->prepare(
        "SELECT COUNT(*) as nb FROM products ");
} else if (...) {
} else if (isset($_GET['orderAverage'])) {
    $request = $bdd->prepare(
        "SELECT AVG(order_total) as avg FROM orders ");
} else if (isset($_GET['salesRevenues'])) {
    $request = $bdd->prepare(
        "SELECT SUM(order_total) as sum FROM orders ");
}

fetchCount("product", countProduct, 1);
fetchCount("user", countUser, 1);
fetchCount("order", countOrder, 1);
fetchCount("orderAverage", avgOrder, 2);
fetchCount("salesRevenues", salesRevenues, 3);
```

```
function fetchCount(table, countDiv, sql) {
    fetch(`traitement/traitement_stats.php?${table}`)
        .then((response) => {
            return response.json();
        })
        .then((data) => {
            if (sql == 1) {
                countDiv.innerText = data[0].nb;
            } else if (sql == 2) {
                let res = Math.round(data[0].avg * 100) / 100;
                countDiv.innerText = `${res}€`;
            } else if (sql == 3) {
                countDiv.innerText = `${data[0].sum}€`;
            }
        });
}
```

En outre, il a la possibilité d'effectuer un **CRUD**.  
Un formulaire dédié à l'ajout de produits et de

catégories est mis à sa disposition.

Ici nous ajoutons un nouveau produit à notre base de données, nous faisons la **liaison** avec la table prévue pour, puis nous ajoutons l'**image** qui sera reliée au produit en effectuant plusieurs vérifications notamment :

- La taille de l'image
- Son extension

Après cela, nous générerons un identifiant unique pour nommer le fichier en utilisant la fonction "`uniqid()`" et le déplaçons vers un emplacement désigné à l'aide de la fonction "`move_upload_file()`".

```

● ● ●
if (empty($name)) {...}
} elseif (...) {...}
} else {
    if ($file['error'] === UPLOAD_ERR_OK) {
        $product = new Product(null, $name, $description, $date, $price, $stock);
        $category = new Category(null, null, $category);
        $product->addProduct($bdd);
        $category->liaisonItemCategory($bdd);

        $returnLastID = $bdd->prepare("SELECT product_id FROM products ORDER BY products.product_id DESC");
        $returnLastID->execute();
        $resultID = $returnLastID->fetch(PDO::FETCH_OBJ);

        $image = new Image(null, $resultID->product_id, $file, 1);
        $image->addImage($bdd);

        $message['PRODUCT_SUCCES'] = '<i class="fa-solid fa-circle-check"></i> Produit ajouté.';
    }
}

```



```

● ● ●
class Image {
    public function addImage($bdd)
    {
        $tmpName = $this->name['tmp_name'];
        $name = $this->name['name'];
        $size = $this->name['size'];
        $error = $this->name['error'];

        $tabExtension = explode('.', $name);
        $extension = strtolower(end($tabExtension));
        $extensions = ['jpg', 'png', 'jpeg', 'gif', 'webp'];
        $maxSize = 2000000;
        $uniqueName = uniqid('', true);
        $file = $uniqueName . "." . $extension;

        if (in_array($extension, $extensions) && $size <= $maxSize && $error == 0) {
            move_uploaded_file($tmpName, '../../../../../assets/img_item/' . $file);

            $insertImage = $bdd->prepare('INSERT INTO images (product_id, image_name, image_main) VALUES (:product_id,:image_name,:image_main)');
            $insertImage->execute([
                'product_id' => $this->product_id,
                'image_name' => $file,
                'image_main' => $this->main
            ]);
        } else {
            echo "Mauvaise extension ou taille trop grande, Une erreur est survenue";
        }
    }
}

```



Pour les catégories nous les insérons à l'aide de la méthode **“addCategory()”**.

```

● ● ●
category = new Category(null, $name, $IDParent);
$category->addCategory($bdd);

```



### Ajouter un Produit

*Name*

*Price*

*Category*

*Stock*

*Description*

**Valider**

### Ajouter une Categorie

*Nom*

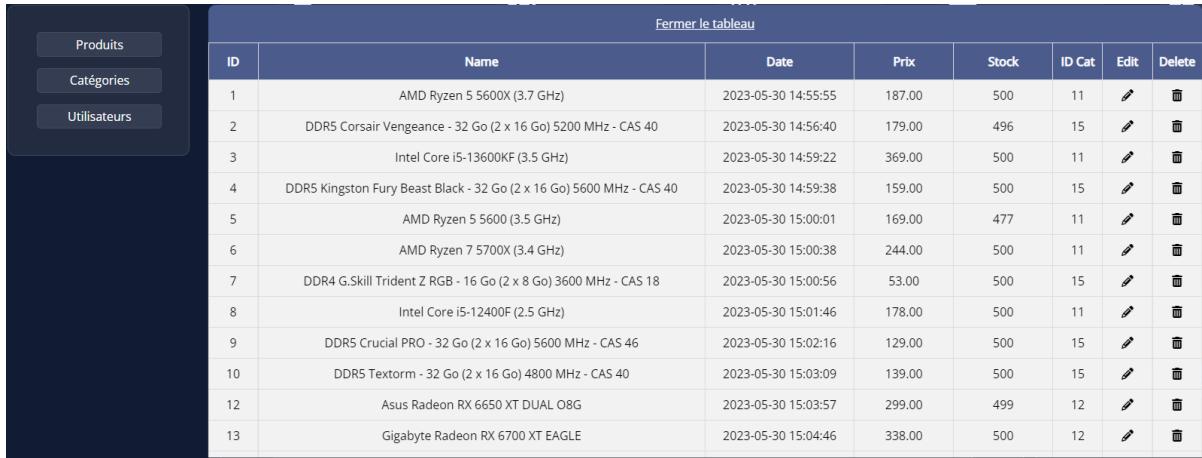
  

*Catégorie parent*

**Ajouter**

En ce qui concerne les produits, les catégories et les utilisateurs déjà présents, l'administrateur peut les consulter, les modifier et les supprimer via des tableaux spécifiques pour chacun de ces éléments.



The screenshot shows a dark-themed user interface for managing products. On the left, there's a sidebar with buttons for 'Produits', 'Catégories', and 'Utilisateurs'. The main area displays a table titled 'Fermer le tableau' (Close the table) with the following data:

ID	Name	Date	Prix	Stock	ID Cat	Edit	Delete
1	AMD Ryzen 5 5600X (3.7 GHz)	2023-05-30 14:55:55	187.00	500	11		
2	DDR5 Corsair Vengeance - 32 Go (2 x 16 Go) 5200 MHz - CAS 40	2023-05-30 14:56:40	179.00	496	15		
3	Intel Core i5-13600KF (3.5 GHz)	2023-05-30 14:59:22	369.00	500	11		
4	DDR5 Kingston Fury Beast Black - 32 Go (2 x 16 Go) 5600 MHz - CAS 40	2023-05-30 14:59:38	159.00	500	15		
5	AMD Ryzen 5 5600 (3.5 GHz)	2023-05-30 15:00:01	169.00	477	11		
6	AMD Ryzen 7 5700X (3.4 GHz)	2023-05-30 15:00:38	244.00	500	11		
7	DDR4 G.Skill Trident Z RGB - 16 Go (2 x 8 Go) 3600 MHz - CAS 18	2023-05-30 15:00:56	53.00	500	15		
8	Intel Core i5-12400F (2.5 GHz)	2023-05-30 15:01:46	178.00	500	11		
9	DDR5 Crucial PRO - 32 Go (2 x 16 Go) 5600 MHz - CAS 46	2023-05-30 15:02:16	129.00	500	15		
10	DDR5 Textorm - 32 Go (2 x 16 Go) 4800 MHz - CAS 40	2023-05-30 15:03:09	139.00	500	15		
12	Asus Radeon RX 6650 XT DUAL O8G	2023-05-30 15:03:57	299.00	499	12		
13	Gigabyte Radeon RX 6700 XT EAGLE	2023-05-30 15:04:46	338.00	500	12		

Lors de la modification d'un produit existant, si l'image associée au produit est mise à jour, l'ancienne image est supprimée du fichier de stockage à l'aide de la fonction "**unlink()**".

```
unlink('.../.../assets/img_item/' . $old_image);
```

Au moyen d'un formulaire et de boutons radio, l'administrateur a la capacité de modifier les **rôles** des autres utilisateurs. Cela offre une fonctionnalité permettant de gérer les priviléges et les droits d'accès des utilisateurs de manière flexible.



The screenshot shows a modal dialog with the title 'Role Actuel : Moderator'. It contains a label 'Role Actuel : Moderator' and three radio buttons for selecting a new role: 'Administrateur' (selected), 'Modérateur', and 'Membre'.

## 6. Responsive

Le site est conçu de manière responsive pour s'adapter à divers formats d'écran, allant du bureau avec une résolution de "1920px" jusqu'au mobile avec une résolution de "320px". Cette approche garantit une expérience utilisateur optimale quel que soit le dispositif utilisé pour accéder au site.

Pour cela nous nous sommes servis des "**Media Queries**". Ces requêtes nous permettent de donner un nouveau style au site en prenant le dessus sur les propriétés initiales en fonction des conditions renseignées. Nous précisons dans un premier temps le type de media, "screen" en l'occurrence, puis la **condition**, par exemple en jusqu'à une "width" de 1440px.

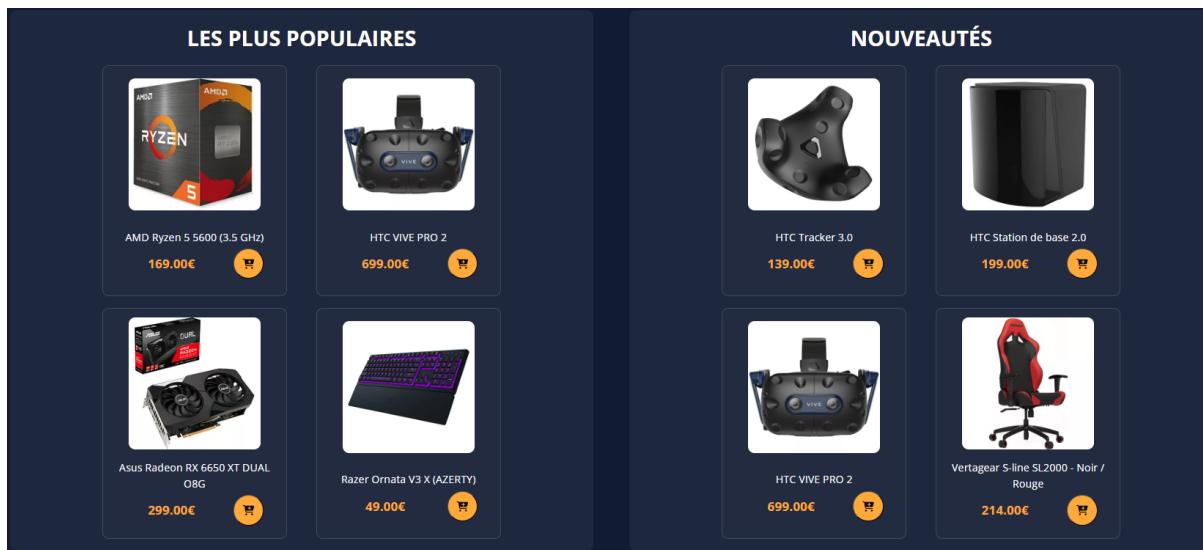
```
@media screen and (max-width: 1440px) {}
@media screen and (max-width: 1024px) {}
@media screen and (max-width: 768px) {}
@media screen and (max-width: 425px) {}
```

Dans cet exemple, nous avons effectué des modifications pour transformer deux sections qui étaient initialement côté à côté en les plaçant l'une en dessous de l'autre. De plus, nous avons ajusté la disposition des éléments en les plaçant à côté des images plutôt qu'en dessous.

Pour parvenir à ces changements, nous avons modifié la propriété "**flex-direction**" de "column" à "row", ainsi que la hauteur et la largeur pour obtenir le résultat souhaité. Ces ajustements permettent d'adapter la mise en page de manière plus appropriée en fonction des besoins spécifiques de l'affichage.

```
.CardProduct {
  background-color: var(--div-color);
  border: 1px solid var(--border-color);
  border-radius: 10px;
  margin-bottom: 2%;
  width: 280px;
  height: 350px;
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  align-items: center;
  padding: 1%;
}
```

```
.CardProduct {
  width: 40%;
  height: 250px;
  flex-direction: row;
  justify-content: space-between;
}
```



*Index au format 1920 px*



*Index au format 1024 px*

Nous avons également pris des mesures pour transformer notre auto-complétion et notre liste de catégories en un menu "burger" à partir de la résolution pour **tablette**.

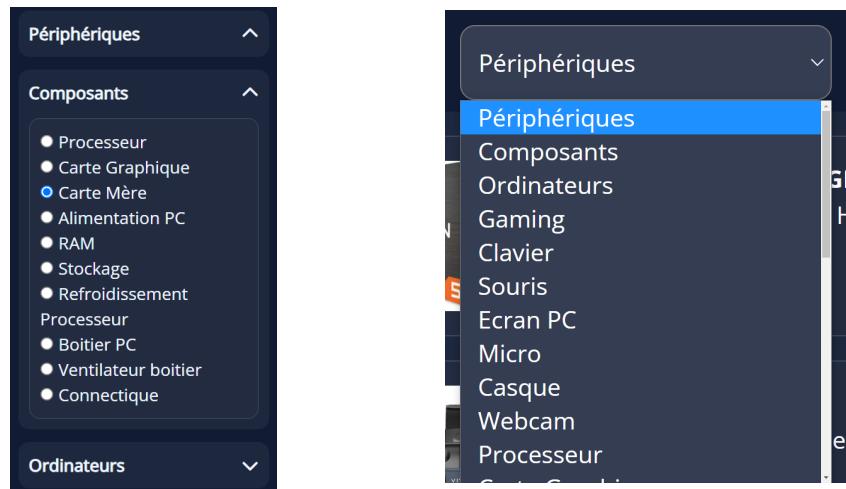
Grâce à cette modification, nous évitons que l'en-tête ne déborde sur les petites résolutions, tout en améliorant l'**expérience utilisateur**.

L'icône du menu burger change à l'ouverture de celui-ci.



Menu burger ouvert

Nous avons également effectué une **transformation** sur notre page "Tous les produits" en modifiant la façon de sélectionner la catégorie souhaitée. Nous sommes passés d'une liste constituée de balises "**ul**" et "**li**" à un menu déroulant "**select**" avec ses **options**.



Afin de faciliter le changement de couleur sur notre site, nous avons maintenant stocké ces valeurs dans des variables CSS. Cette approche nous permet de modifier la couleur une seule fois, et le changement sera appliqué à l'ensemble du site.

```

    :root {
        --background-color: #111b33;
        --section-color: #1d273d;
        --div-color: #232b41;
        --nav-color: #2f374c;
        --border-color: #444b5d;

        --input-color: #343d51;
        --button-color: #0091ee;
        --text-input-color: #90949b;

        --price-color: #f75b5b;
        --stock-color: #38b76e;
        --stockLimite-color: #ea8013;

        --main-color: #005078;
        --second-color: #007cb8;
    }

```

## 7. Sécurité

### 7.1. Htmlspecialchars

Cela nous permet d'empêcher l'utilisateur de rentrer du code dans les différents champs à leur disposition et qu'il soit exécuté en convertissant les caractères spéciaux en **entités HTML**.

Nous utilisons la fonction à l'affichage des données, comme le montre l'exemple ci-dessous :

```
● ● ●

<div class="InfoAddress">
    <p id="name">
        <?= htmlspecialchars($userAdress->address_lastname) . " " . htmlspecialchars($userAdress->address_firstname) ?>
    </p>
    <p>
        <?= htmlspecialchars($userAdress->address_numero) . " " . htmlspecialchars($userAdress->address_name) ?>
    </p>
    <p>
        <?= htmlspecialchars($userAdress->address_city) . ", " . htmlspecialchars($userAdress->address_postcode) ?>
    </p>
    <p>France</p>
    <p><?= htmlspecialchars($userAdress->address_telephone) ?></p>
</div>
```

### 7.2. Injection SQL

Les injections SQL permettent à l'utilisateur de modifier nos requêtes SQL.

Pour les empêcher, nous préparons nos requêtes à l'aide de la fonction "**prepare()**"

```
● ● ●

// Récupération du produit
$returnProduct = $bdd->prepare("SELECT * FROM products WHERE product_id = :product_id");
$returnProduct->execute(['product_id' => $_GET['id']]);
$result = $returnProduct->fetch(PDO::FETCH_OBJ);
```

Voici un exemple d'injection SQL, l'utilisateur ici rentre dans le champ son nom suivi de 2 tirets (--), ce qui permet de mettre en commentaire la suite de la requête et donc de ne pas prendre en compte la condition pour le mot de passe.

```
SELECT uid FROM Users WHERE name = 'Dupont';--' AND password = '4e383a1918b432a9bb7702f086c56596e';
```

### 7.3. Password Hash

Nous avons sécurisé les mots de passe des utilisateurs en les cryptant.

Nous avons utilisé la fonction “**password\_hash()**” qui permet de sécuriser les mots de passe en les cryptant à l'aide de l'algorithme “**Bcrypt**”.

```
$password = password_hash(trim($this->password), PASSWORD_DEFAULT);
```

Voici un exemple d'un mot de passe crypter :

```
user_password  
$2y$10$uFx8wvIAhgmw93DDzIno/O/w5g2JN20kXPvvC83HnKW...
```

Pour pouvoir utiliser ce mot de passe, comme par exemple dans un formulaire de connexion, nous devons pouvoir le comparer au mot de passe entré par l'utilisateur, pour cela nous utilisons la fonction “**password\_verify**” :

```
if (password_verify($this->password, $result->password)) {  
    .... }
```

### 7.4. Redirection

Nous avons ajouté des conditions pour accéder à certaines pages comme par exemple ce code ci qui redirige les utilisateurs qui n'ont pas le rôle d'administrateur vers l'index :

```
if ($_SESSION['user']->user_role == 0) {  
    header('Location: ../index.php');  
    exit();  
}
```

ou encore ce code ci qui redirige les utilisateurs déjà connectés vers l'index :

```
if (isset($_SESSION['user'])) {  
    header('Location:../index.php');  
}
```

Nous avons également mis en place une sécurité pour les changements d'**URL** sur les pages avec des paramètres en **GET**.

Ici si ce qui est rentré dans l'URL n'est pas un **ID** d'un produit existant alors l'utilisateur est redirigé vers l'index.

```
// Récupération du produit  
$returnProduct = $bdd->prepare("SELECT * FROM products WHERE product_id = :product_id");  
$returnProduct->execute(['product_id' => $_GET['id']]);  
$result = $returnProduct->fetch(PDO::FETCH_OBJ);  
  
// Empêche d'aller sur la page si il n'y a aucun produit de sélectionner  
if (!$result) {  
    header('Location: ../index.php');  
}
```

## 8. Recherche effectuée depuis un site anglophones

Nous étions à la recherche d'un moyen de récupérer l'URL de la page actuelle afin de l'utiliser pour nos redirections et appels de fichiers.

Finalement, nous avons trouvé une solution sur le site "**javatpoint**". Ils expliquent qu'en utilisant la superglobale **\$\_SERVER**, une variable intégrée de PHP, nous pouvons obtenir l'URL de la page courante.

Pour obtenir l'URL, il faut d'abord vérifier si le protocole est "**http**" ou "**https**".

Ensuite, dans leur exemple, ils montrent comment assembler l'**URL** avec l'opérateur "**.=**" qui va nous permettre de concaténer les différentes parties de l'URL.

To get the current page URL, PHP provides a superglobal variable `$_SERVER`. The `$_SERVER` is a built-in variable of PHP, which is used to get the current page URL. It is a superglobal variable, means it is always available in all scope.

If we want the full URL of the page, then we'll need to check the protocol (or scheme name), whether it is https or http. See the example below:

```
<?php  
if(isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on')  
    $url = "https://";  
else  
    $url = "http://";  
// Append the host(domain name, ip) to the URL.  
$url.= $_SERVER['HTTP_HOST'];  
  
// Append the requested resource location to the URL  
$url.= $_SERVER['REQUEST_URI'];  
  
echo $url;  
?>
```

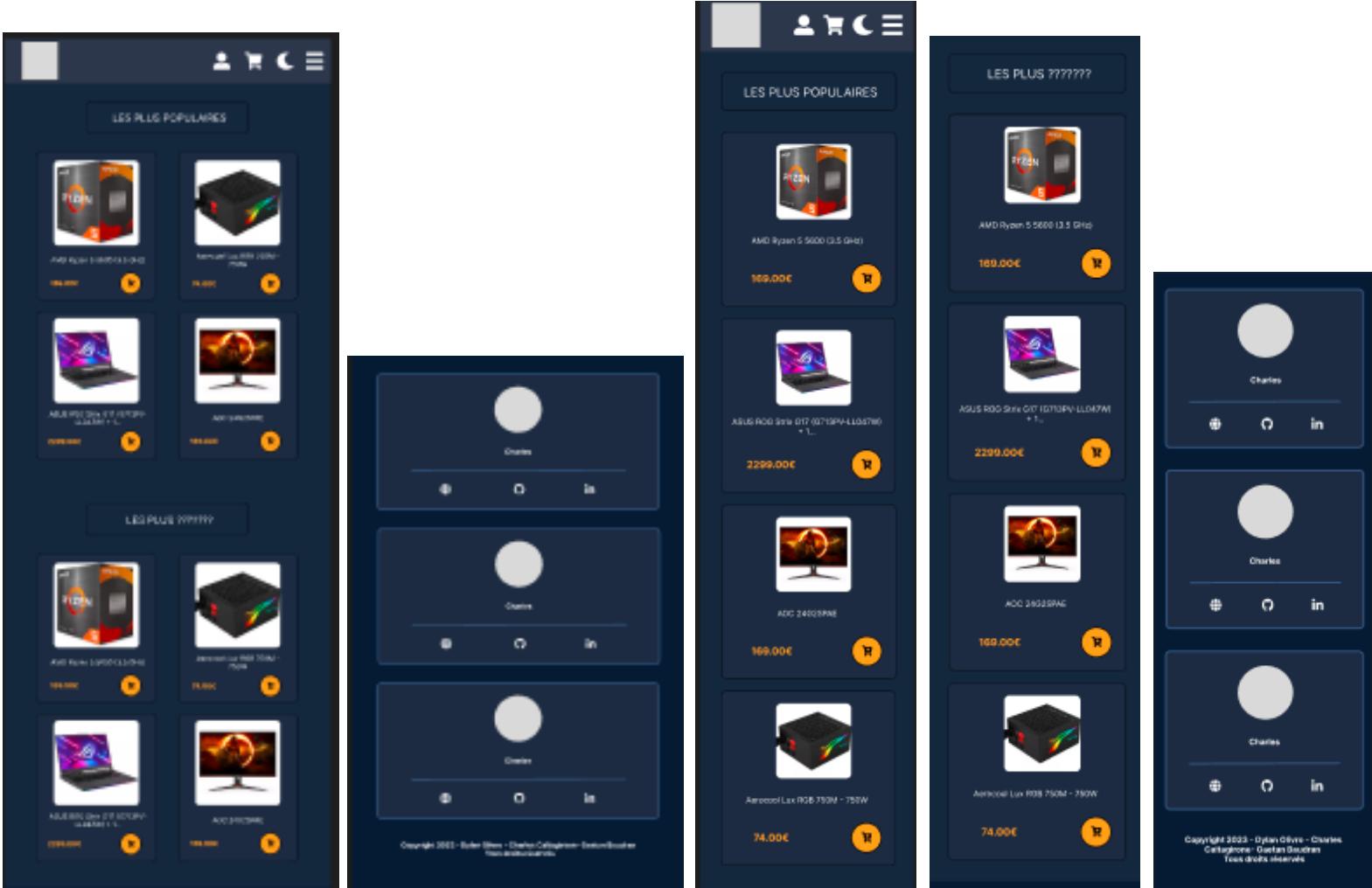
## 9. Axes d'amélioration

Voici une liste non exhaustive d'améliorations possibles à apporter au projet :

- Ajoutez une **pagination** sur la page “Tous les produits”.
- Ajouter la possibilité de mettre des **promotions**.
- Ajouter une **autocomplete** au panel admin afin de trouver plus facilement les produits, les catégories ou les utilisateurs.
- Ajouter une page **résultat** avec l'autocomplétion.
- Ajout d'une véritable **solution de paiement**, comme “**PayPal**” ou encore “**Stripe**”.

# Annexes

## Maquette



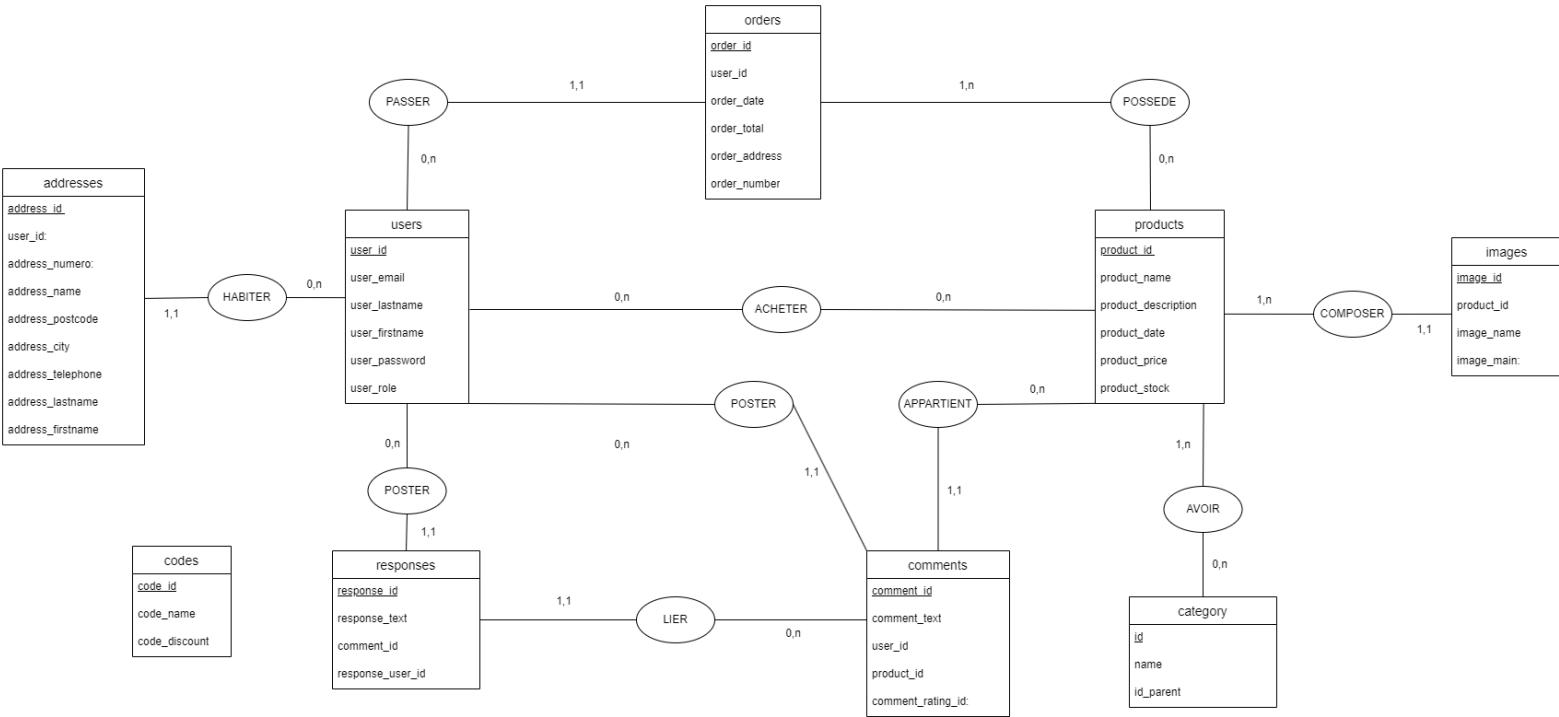
Nous retrouvons ici les autres maquettes réalisées, sous format tablette et mobile.

[Github](#)

[Boutique](#)

[Portfolio](#)

## MCD



## MLD

