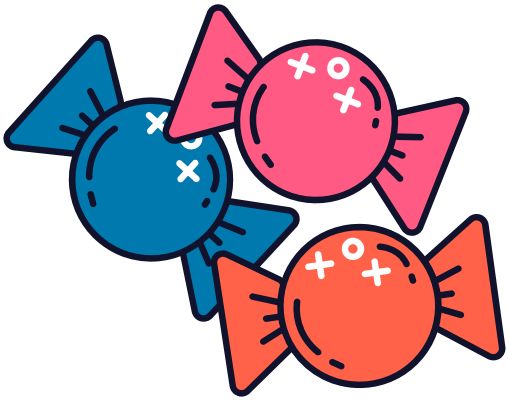
Candy shop

DOSSIER DE PROJET

Titre: Développeur Web et Web Mobile

Création d'une boutique en ligne



Julie Lambert

Table des matières

Compétences du référentiel couvertes par le projet

Le projet couvre les compétences énoncées ci-dessous.

Activité 1: **"Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurités"**:

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

Activité 2: **"Développer la partie back-end d'une applications 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
- Elaborer et mettre en oeuvre des composants dans une application de gestion de contenu ou e-commerce

Introduction

Pour le projet de fin d'année nous avons eu le projet "Boutique en ligne". Nous avons donc eu en charge la création d'un site de E-commerce en totalité.
Ce projet était a réaliser en groupe.

Le choix du thème étant libre nous avons décider de créer notre site sur le thème des confiseries. Nous avons pris le parti de faire un design simple et ludique avec une interface agréable pour les utilisateurs pour notre boutique de bonbons

Les compétences visées par ce projet étaient:

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou
e-commerce
- Créer une base de données
- Développer les composants d’accès aux données
- Développer la partie back-end d’une application web ou web mobile
- Elaborer et mettre en œuvre des composants dans une application de gestion de
contenu ou e-commerce.

Les langages utilisées sont PHP, Javascipt , et SQL.
Nous avons aussi utilisé le logiciel Figma pour le maquettage de l'application et Diagrams.net pour le maquettage de notre base de données.

Spécifications fonctionnelles

Description de l'existant

Dans ce projet le but était de tout créer a partir de zéro , en partant du maquettage , la base de données ainsi que toutes les fonctionnalités liées a notre site. Il n'existait aucune base de départ.

Périmètre du projet

Le site est réalisé en Français et il doit s'adapter à différents support tel que mobiles, tablettes et ordinateurs.

Cible visée par le site internet

Notre site vise les particuliers mais surtout les amoureux de bonbons ainsi que leur produits dérivés.

Arborescence du site

L'arborescence du site est définie telle qu'elle :

- Page d'accueil
- Page d'authentification : Connexion et Inscription
- Page catalogue
- Page catégories
- Page produit
- Page profil
- Page administrateur

Descriptions des fonctionnalités

1. **Authentification**

L'utilisateur à accès à une page d'authentification avec formulaire de connexion/inscription, où il peut soit se connecter s'il a déjà un compte ou s'inscrire s'il le souhaite. Il pourra alors enregistrer toutes les informations nécessaires à la créations de son compte.

2. **Catalogue produit**

La page catalogue permet d'afficher l'ensemble des produit présent sur le site et comprend certaine informations liées aux produits tel que : la photo, le nom, le prix ainsi qu'un bouton "Ajouter au panier"

3. **Catégories**

Le site comporte un menu qui permet de trier les différents articles par catégories : Bonbons , Chocolats , Goodies .

4. **Fiche produit**

La fiche fiche produit pemet d'afficher les informations relative au produit elle contient :

- La photo du produit
- Le nom du produit
- Le prix
- La description

5. **Profil**

La page profil récupère toutes les informations correspondantes à un utilisateur comme son nom, prénom, pseudo, email et mot de passe et lui permet de pouvoir modifier une information si nécessaire.

6. **Back office**

L'administrateur du site à accès a un espace qui lui est dédié afin qu'il puisse facilement administrer le site

6.1 **Gestion des produits**

L'administrateur peut , grâce a son espace ajouter ou supprimer un produit du catalogue .
Lors de l'ajout d'un produit il peut entrer les informations suivantes:

- Le nom du produit
- Un sous titre
- Uploader la photo du produit
- Le prix du produit
- Le stock de produit disponible
- Une description du produit
- Choisir la catégorie du produit

6.2 **Gestion des utilisateurs**

L'administrateur du site a également accès à la liste complète de tous les utilisateurs du site ainsi qu'a leurs informations. Il peut, si c'est nécessaire, supprimer un utilisateur de la liste et donc supprimer son accès au site.

Spécifications techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end:

- PHP est principalement utilisé pour ce projet
- Base de données SQL

Technologies utilisées pour la partie front-end:

- HTML
- CSS pour le style ainsi que le reponsive
- Javascript quipermet de rendre le site dynamique

Environnement de développement:

- Editeur de code: VScode
- Outil de versioning: GIT et Github
- Maquettage: Figma
- Base de donées: Diagrams.net

En ce qui concerne l'organisation de travail nous avons simplement utilisé le chat google car nous étions seulement deux à travailler sur ce projet. Nous nous sommes, d'un commun accord, répartis les tâches au fur et à mesure de l'avancement du projet.

Architecture du projet

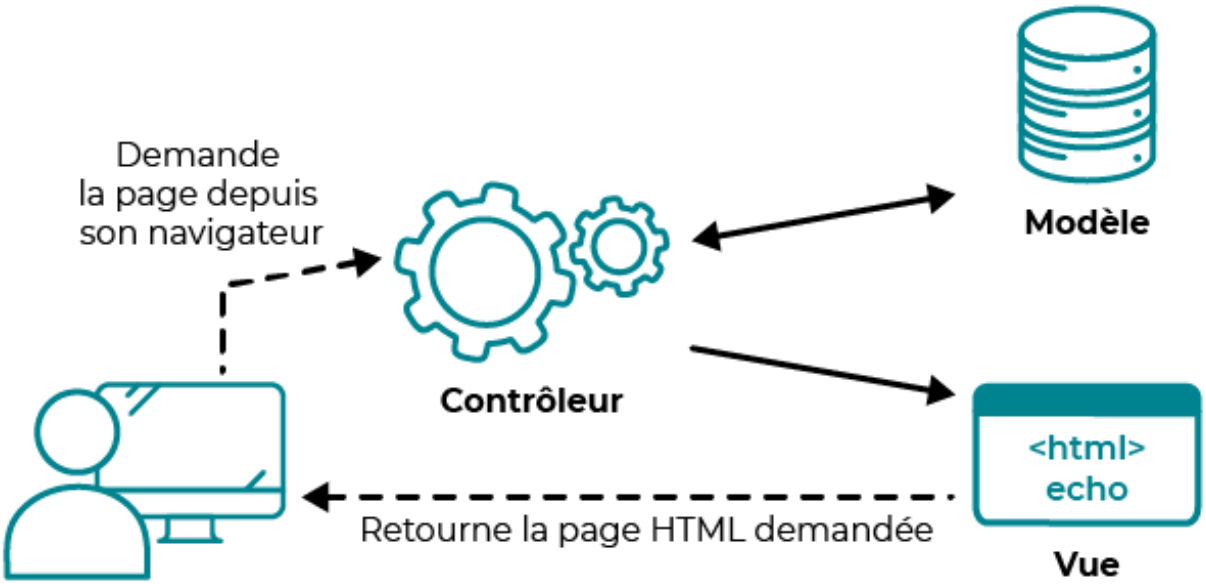
Pour ce projet nous avons voulus nous inspirer de l'architecture MVC (Model - View - Controller) afin de mieux structuré notre code.

Le design pattern MVC est l''un des patterns les plus utilisés. Le but du MVC est de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

- **Modèle**: cette partie gère ce qu'on appelle la logique métier du site. Elle comprend notamment la gestion des données qui sont stockées, mais aussi tout le code qui prend des décisions autour de ces données. Son objectif est de fournir une interface d'action la plus simple possible au contrôleur. On y retrouve donc entre autres des algorithmes complexe est des requêtes SQL.

- **Vue**: Cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.

- **Contrôleur**: Cette partie gère les échanges avec l'utilisateur. C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue. Le contrôleur va recevoir des requêtes de l'utilisateur. Pour chacune, il va demander au modèle d'effectuer certaines actions et de lui renvoyer les résultats. Puis il va adapter ce résultat et le donner à la vue. Enfin , il va renvoyer la nouvelle page HTML, générée par la vue, à l'utilisateur.



Le client et l'architecture MVC Source: Openclassrooms

Réalisations

1. **Maquette**

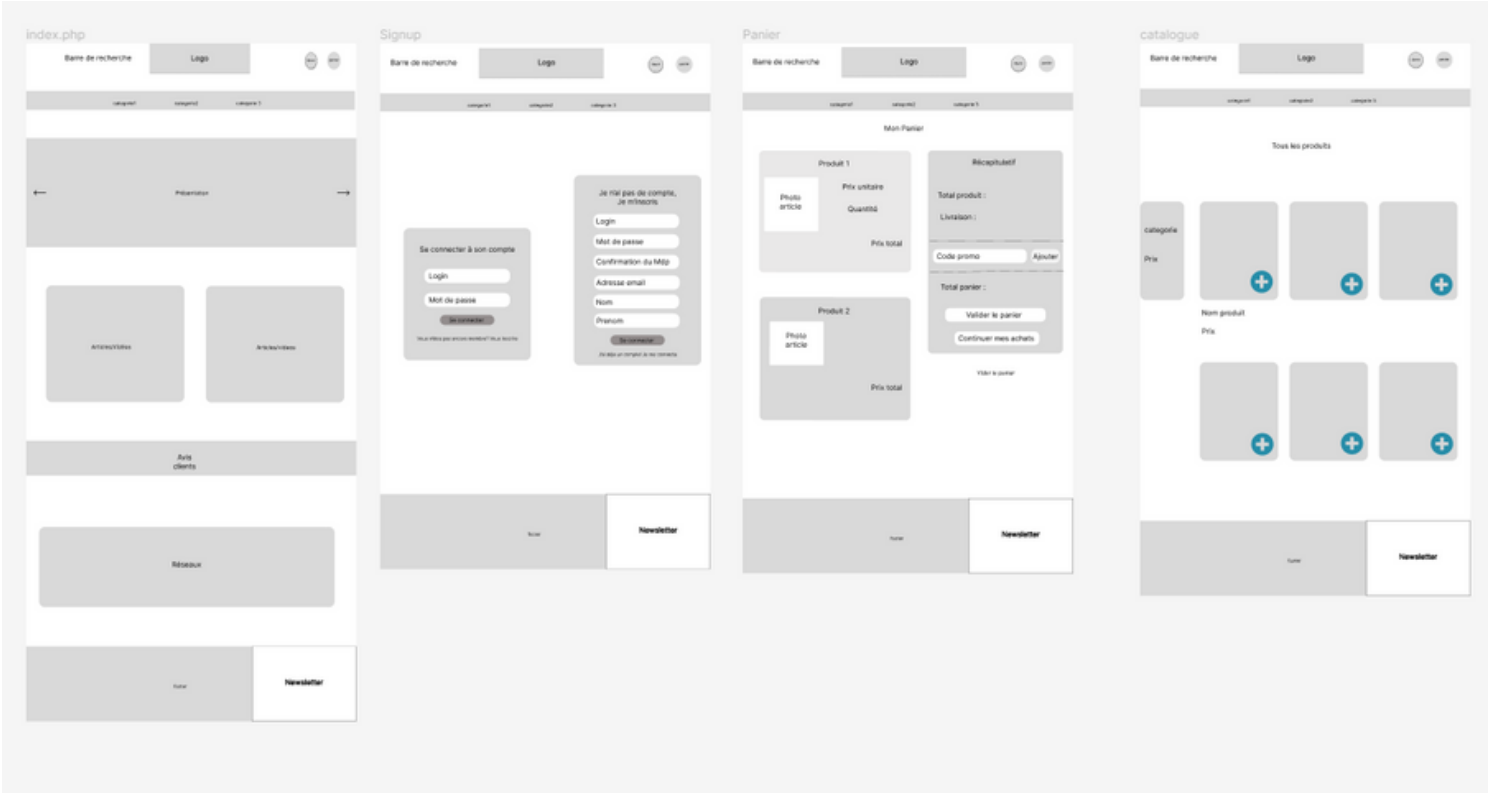
Le maquettage est une étape importante de la création du site. Il nous a permis d'avoir une vue d'ensemble de ce à quoi ressemblerai notre site, de voir toutes les fonctionnalités dont nous avions besoin et de déterminer la charte graphique. Nous l'avons réalisé avec le logiciel Figma .

Nous avons donc commencé par créer le Wireframe Basse fidélité qui est en noir et blanc ce qui permet de se concentrer sur les fonctionnalités essentiel sans être dérangé par le design.

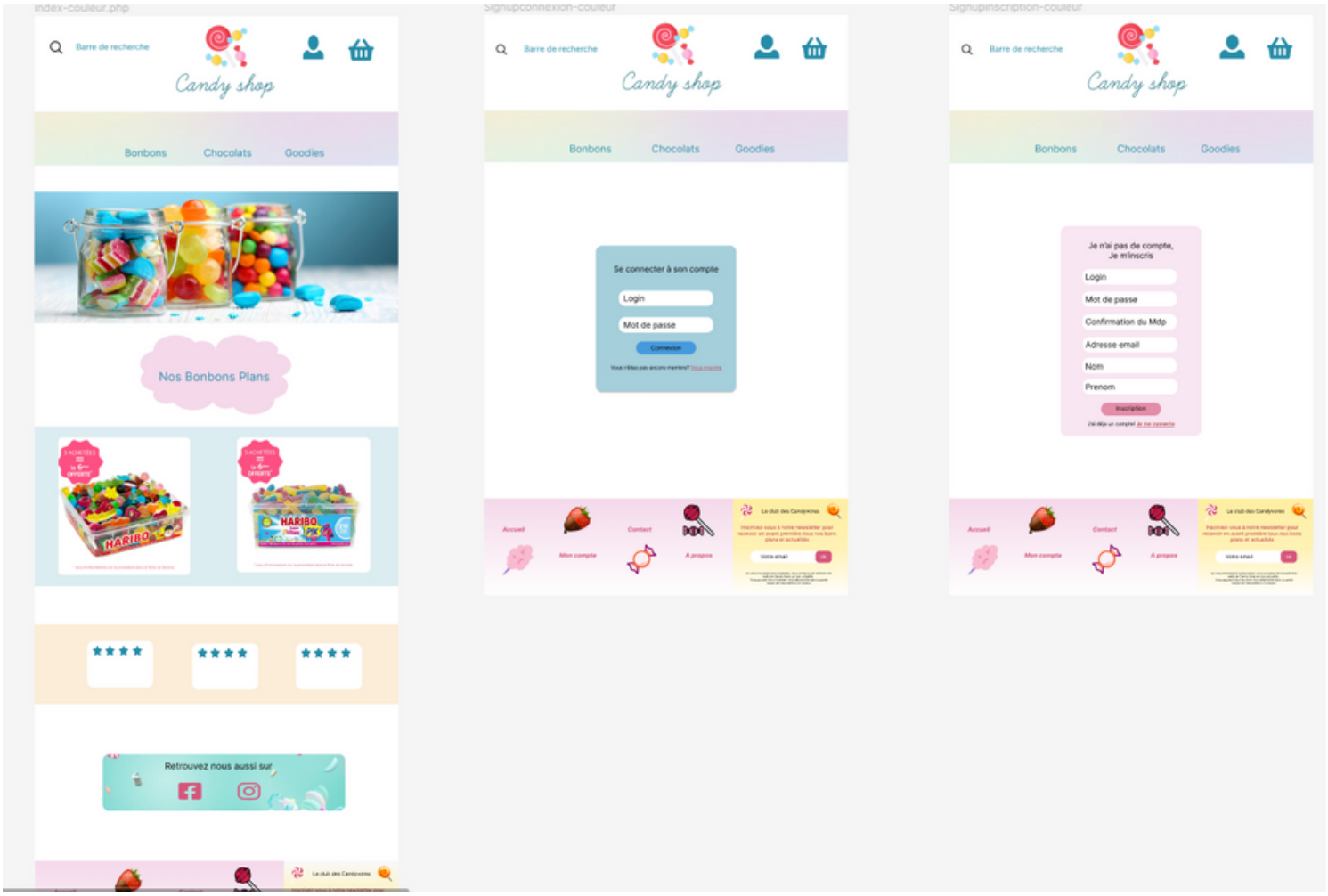
Nous avons ensuite fait la charte graphique qui nous a permis de créer le wireframe haute fidélité où cette fois nous nous sommes concentré sur le design du site.

Nous avons aussi créer la version mobile du site.

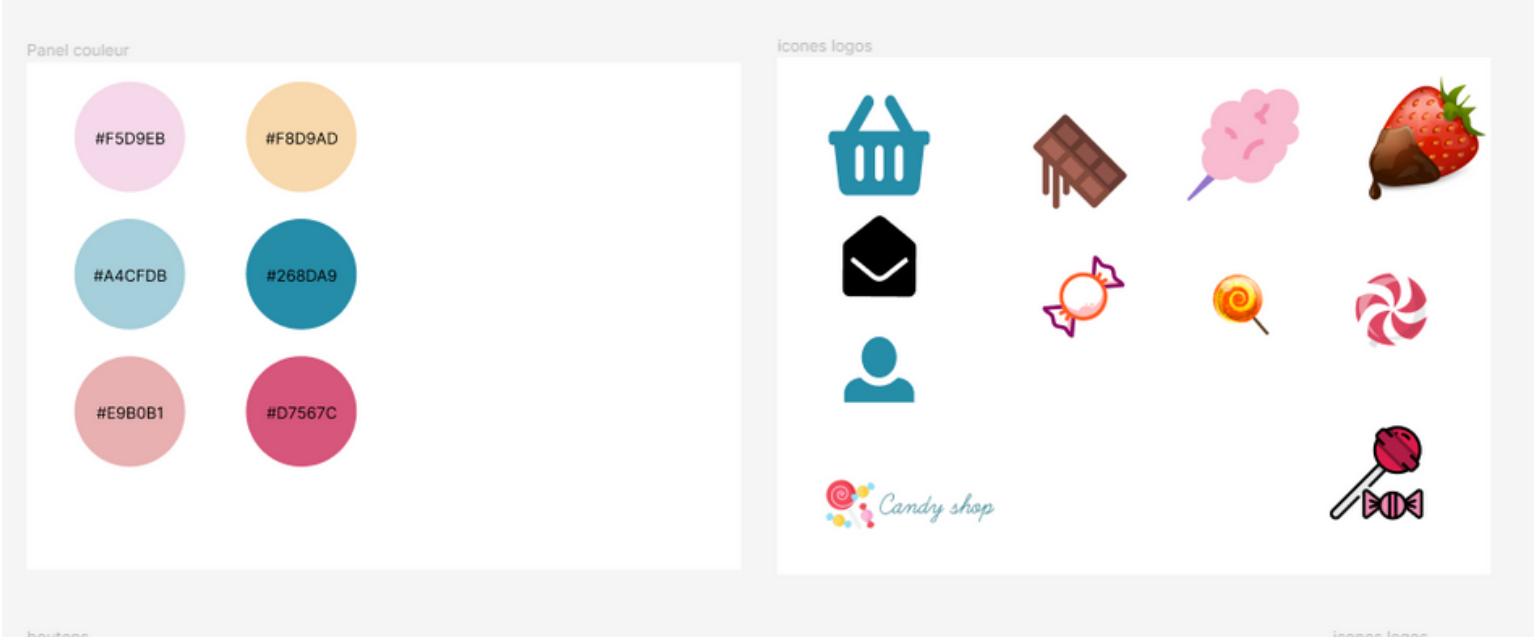
Wireframe basse fidélité



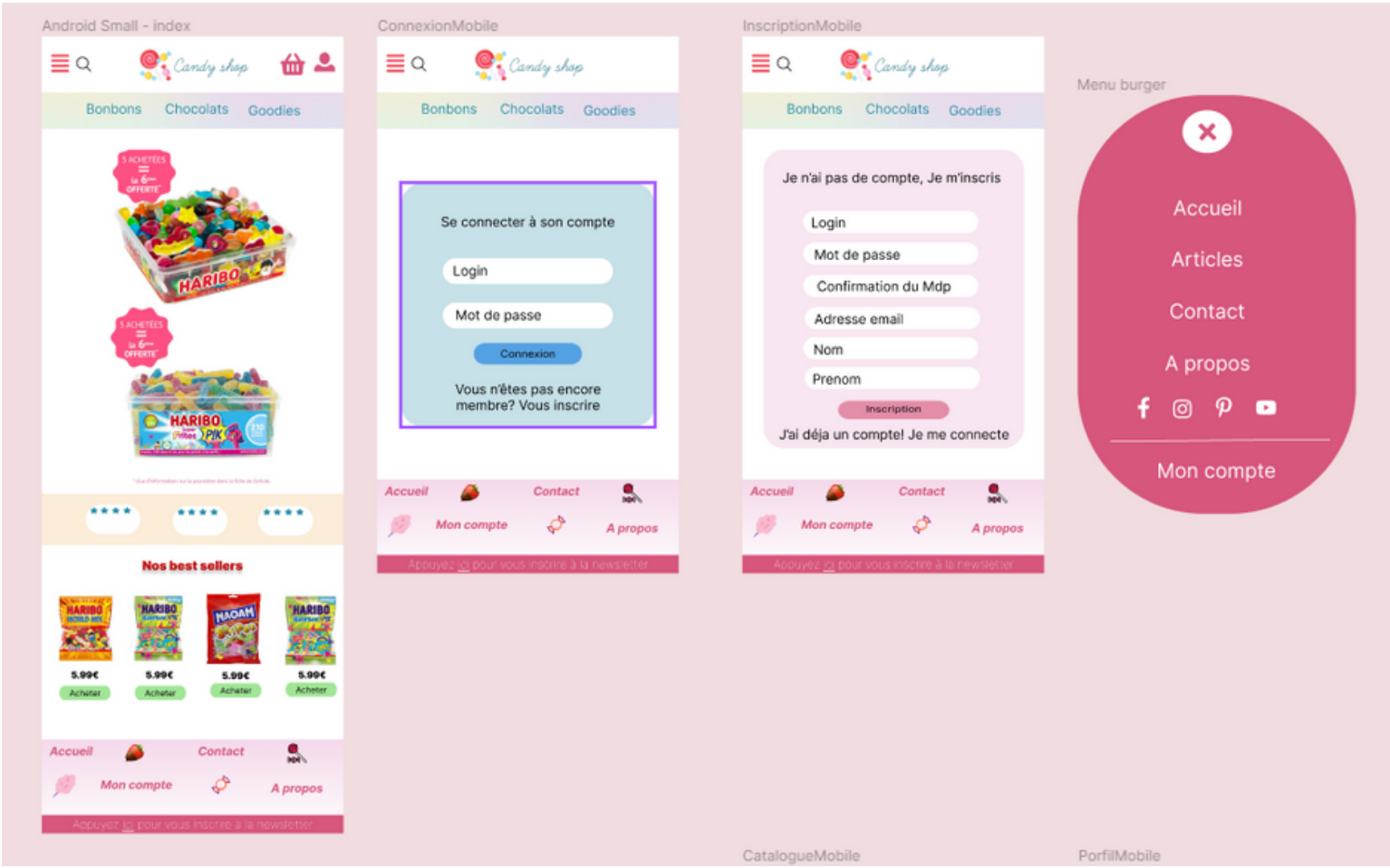
Wireframe haute fidélité



Charte graphique

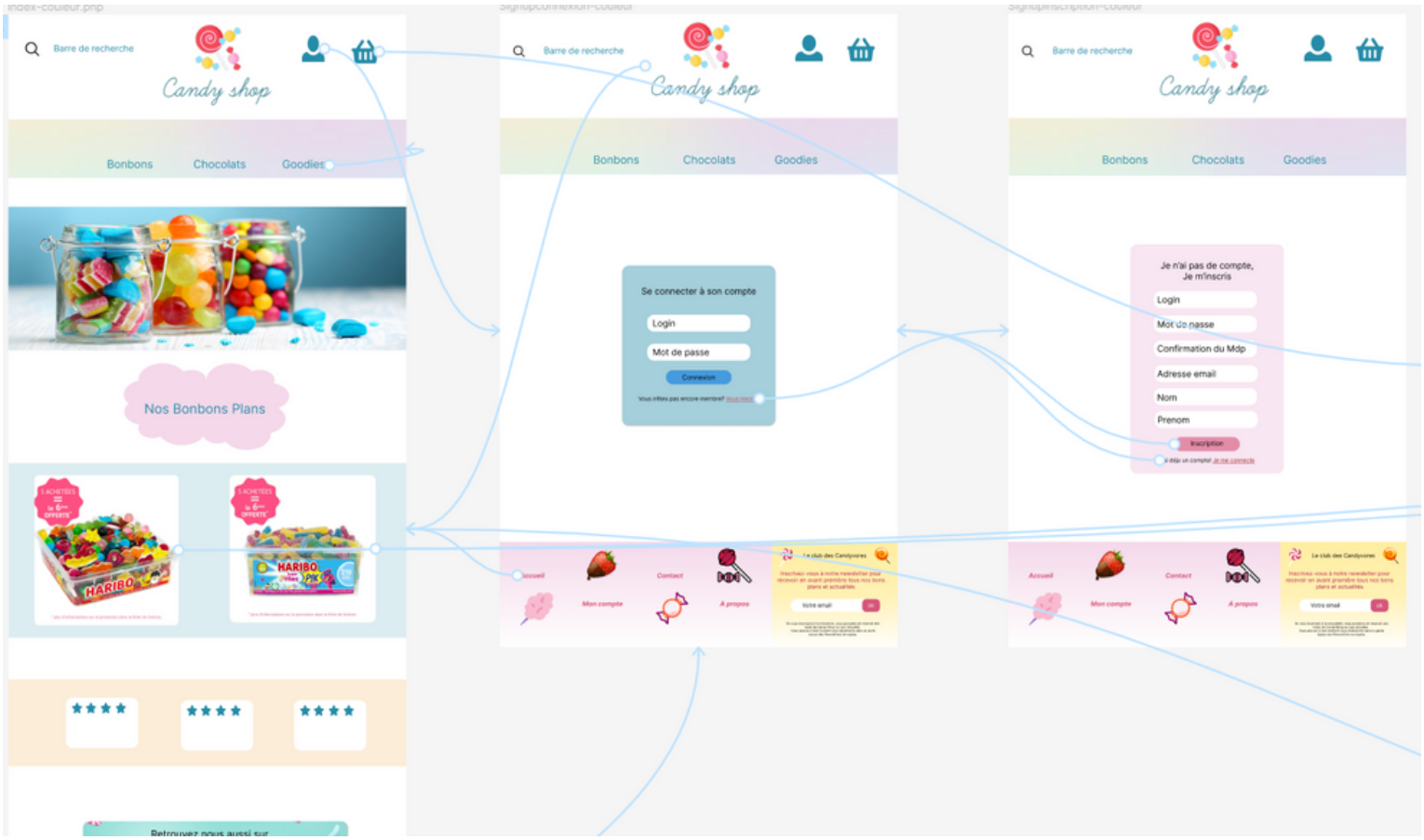


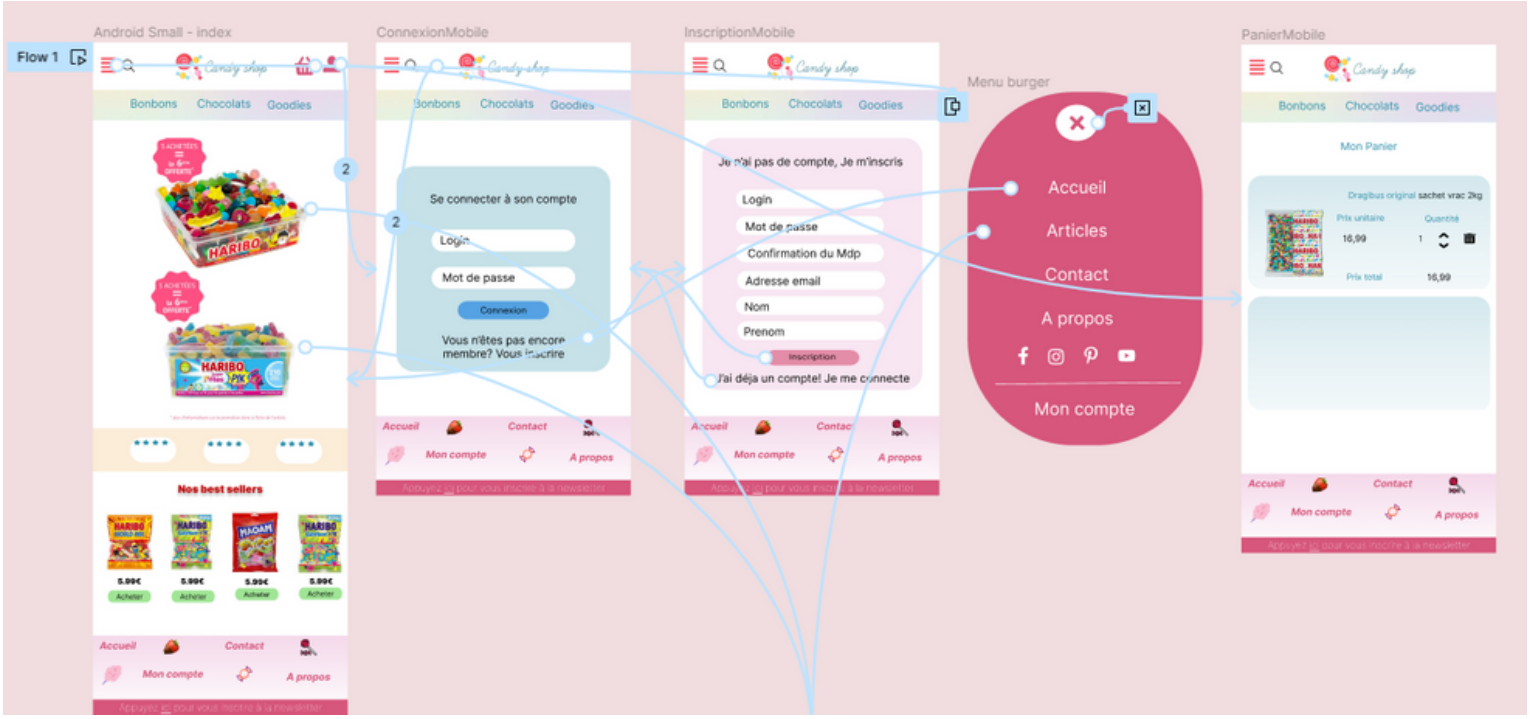
Version Mobile



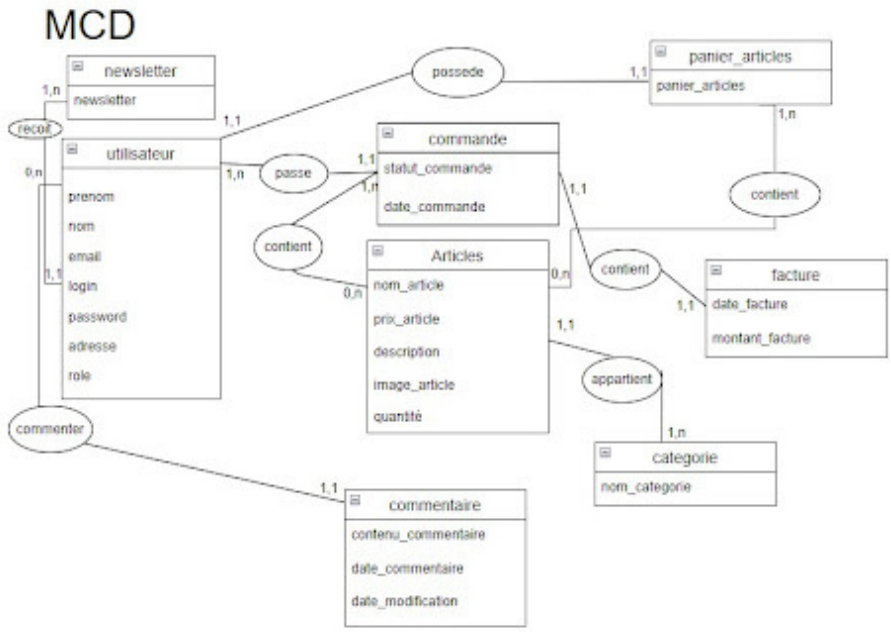
Prototypage

Le prototypage nous permet de voir l'interaction entre les pages selon le bouton sur lequel on clique.





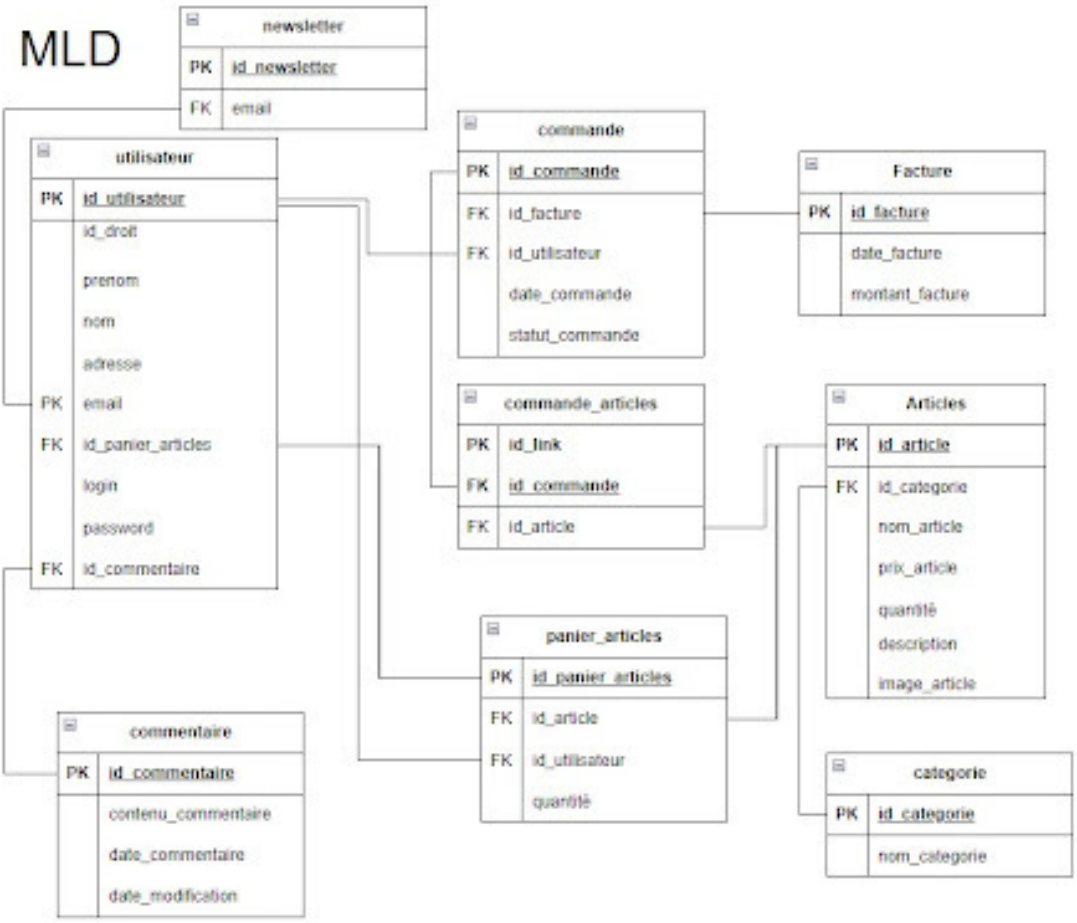
2. **Base de données**



Le MCD (Modèle Conceptuel de Donnée) est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux.

Il fait apparaitre :

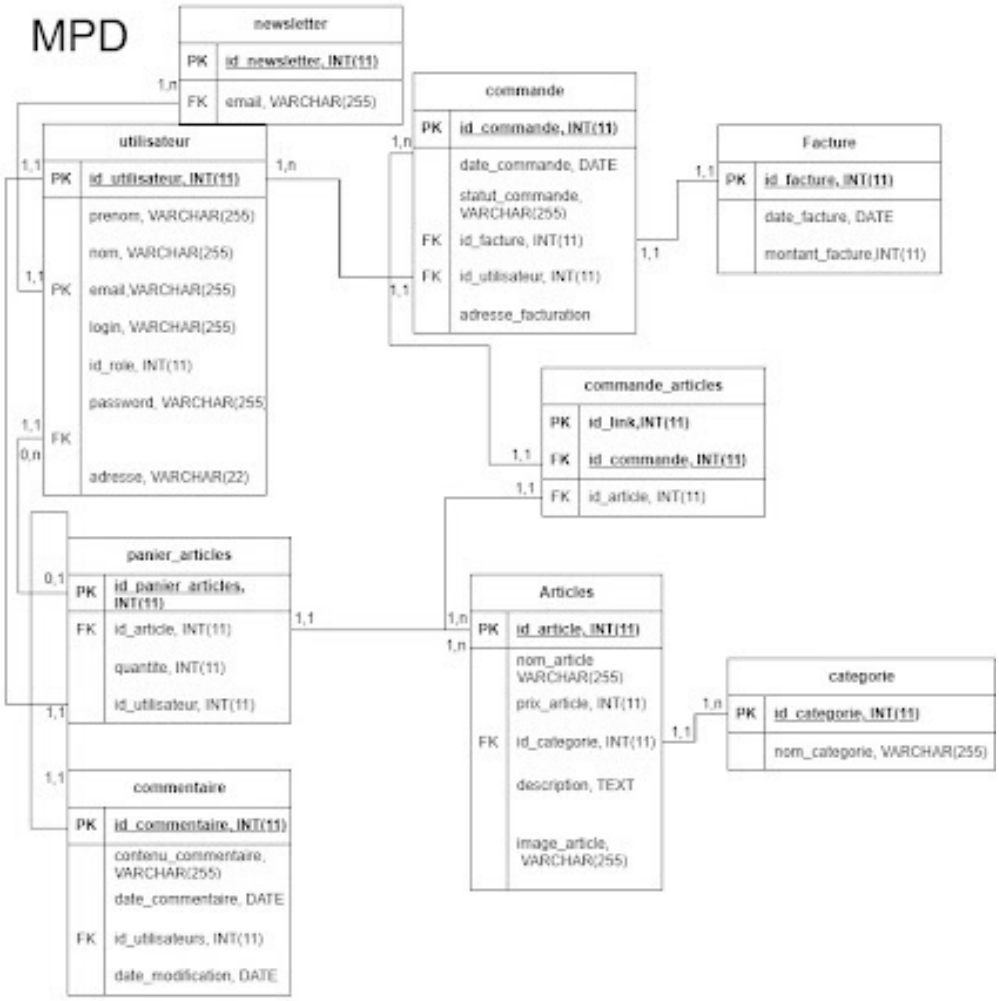
- Les entités
- Les propriétés
- Les relations qui expliquent et précisent comment les entités sont reliées entre elles
- Les cardinalités



Le MLD (Modèle Logique de Données) est la représentation textuelle du MPD. Il représente la structure de la base de donnée.

Il fait apparaitre:

- Les tables
- Les tables de relations
- Les clés primaires
- Les clés étrangères



Le MPD (Modèle Physique de Données)fait suite aux MCD et MLD.

Cette étape permet de construire la structure finale de la base de données avec les différents liens entre les éléments qui la composent

3. **Extraits de codes significatifs**

3.1 Affichage conditionnel

```php
<div id="Profilpic">
    <a href="#"><img class="profilpic" src="../assets/img/profil.png"></a>
    <div class="infobulle-profil">
        <?php if(empty($_SESSION['login'])) : ?>
            <a href="../viewer/signup.php?form=connexion" id="connexion">Se connecter</a>
            <a href="../viewer/signup.php?form=inscription" id="inscription">Inscription</a>
        <?php elseif($_SESSION['login']=='admin') : ?>
            <h3>Bienvenue <?php echo $_SESSION['login']; ?></h3>
            <a href="../viewer/panelAdmin.php">Panel Admin</a>
            <a href="../controller/logout.php" id="logout">Déconnexion</a>
        <?php else : ?>
            <h3>Bienvenue <?php echo $_SESSION['login']; ?></h3>
            <a href="../viewer/updateprofil.php">Modifier le Profil</a>
            <a href="#">Mon compte</a>
            <a href="../controller/logout.php" id="logout">Déconnexion</a>
        <?php endif; ?>
    </div>
</div>
```

Ici nous avons décidé de créer un affichage conditionnel qui permet d'avoir un menu différent selon que l'on soit connecter ou non et si c'est un administrateur ou un simple utilisateur qui est connecté.

Pas connecté: Se connecter/Inscription
Admin: panel admin/Déconnexion
Utilisateur: Modifier le profil/Mon compte/Déconnexion

3.2 Ajouter un article

```php
public static function addArticle($nom_article, $prix, $description, $photo, $stock, $categorie){

    require_once('../config/db.php');
    global $bdd;

    $request= $bdd->prepare('INSERT INTO article(nom_article, prix, description, photo, stock, id_categorie) VALUES (:nom_article, :prix, :description, :photo, :stock, :id_categorie)');
    $request->bindParam(':nom_article', $nom_article);
    $request->bindParam(':prix', $prix);
    $request->bindParam(':description', $description);
    $request->bindParam(':photo', $photo);
    $request->bindParam(':stock', $stock);
    $request->bindParam(':id_categorie', $categorie);
    $request->execute();
}
```

```php
<?php
require_once('../src/Article.php');
$article = new Article;
$mess_done = "L'article a été ajouté";
$mess_error= "Veuillez remplir tous les champs";

if(!empty($_POST['nom_article'])&& !empty($_POST['prix'])&& !empty($_POST['description']) &&!empty($_POST['photo']) &&!empty($_POST['stock'])&& !empty($_POST['categorie'])){

    $nom_article=htmlspecialchars($_POST['nom_article']);
    $prix=htmlspecialchars($_POST['prix']);
    $description=htmlspecialchars($_POST['description']);
    $photo=htmlspecialchars($_POST['photo']);
    $stock=htmlspecialchars($_POST['stock']);
    $categorie=htmlspecialchars($_POST['categorie']);

  $article->addArticle($nom_article,$prix,$description,$photo,$stock,$categorie);

  return $mess_done;
}else{
  return $mess_error;
}
?>
```

Pour ajouter un article à la liste des produits j'ai d'abord créé un formulaire HTML avec tous les champs nécessaire aux informations du produit.

Dans la classe Article j'ai ensuite créé une fonction qui permet d'ajouter les informations a la base de données grâce a une requête SQL

Au niveau du contrôleur j'effectue toutes les vérifications nécessaires à l'insertion de l'article en base de données.

3.3 Affichage de tous les articles + pagination

```php
<?php

// Determine la page sur laquelle nous sommes

if(isset($_GET['page']) && !empty($_GET['page'])){
    $currentPage = (int) strip_tags($_GET['page']); //
}else{
    $currentPage=1;
}

require_once('../config/db.php');

//détermine le nombre d'articles total

$request = $bdd->prepare('SELECT COUNT(*) AS nb_articles FROM `article`');
$request->execute();
$result = $request->fetch();

$nbArticles = $result['nb_articles'];


// Détermine le nombre d'article par page
$parPage = 12;

// Calcule le nombre de pages total
$pages = ceil($nbArticles / $parPage); // ceil est une fonction qui  arrondis à l'entier supérieur

// Calcul du 1er article
$premier = ($currentPage * $parPage) - $parPage;

$request = $bdd->prepare('SELECT * FROM `article` ORDER BY `nom_article` LIMIT :premier,:parpage');
$request->bindValue(':premier', $premier, PDO::PARAM_INT);
$request->bindValue(':parpage', $parPage, PDO::PARAM_INT);
$request->execute();
$articles = $request->fetchAll(PDO::FETCH_ASSOC);

?>
```

```php
<div class="container">
   <?php foreach($articles as $value) : ?>
    <p id="mess_form"></p>
    <a href= "oneArticle.php?id-product=<?=$value['id'] ?>">
        <div id ="display_articles">
            <img src="../assets/img/<?= $value['photo'] ?>" alt="img">
            <h2><?= $value['nom_article'] ?></h2>
            <p class="sous_titre"><?= $value['sous_titre'] ?></p>
            <p class ="prix"><?= $value['prix'] ?>€</p>
            <button type="submit" name ="addPanier" id="addPanier" >Ajouter au Panier</button>
        </div>
    </a>
     <?php endforeach ?>
</div>
<ul class="pagination">
    <!-- Lien vers la page précédente (désactivé si on se trouve sur la 1ère page) -->
    <?php if($currentPage === 1 ):?>
        <!-- <a href="?page=">Précédente</a> -->
        <?php else : ?>
                    <a href="?page=<?= $currentPage - 1 ?>" class="page-item">Précédente</a>
        <?php endif; ?>

    </li>
     <?php for($page = 1; $page <= $pages; $page++): ?>
    <!-- Lien vers chacune des pages (activé si on se trouve sur la page correspondante) -->
    <li class="page-item <?= ($currentPage == $page) ? "active" : "" ?>">
         <a href="?page=<?= $page ?>"><?= $page ?></a>
    </li>
    <?php endfor ?>
    <!-- Lien vers la page suivante (désactivé si on se trouve sur la dernière page) -->
    <?php if($currentPage == $pages):?>

        <?php else : ?>
                    <a href="?page=<?= $currentPage + 1 ?>" class="page-item">Suivante</a>
        <?php endif; ?>

</ul>
<?php require_once('../include/footer.php') ?>
</body>
</html>
```

Cette partie du code montre comment j'ai procédé pour effectuer l'affichage de tous les produits sur la page catalogue et faire une pagination. J'ai d'abord à l'aide de requêtes SQL récupéré tous les articles qui se trouvent en base de données puis je boucle sur chaque articles pour afficher toutes les informations que je souhaite tel que le photo, le nom, le prix...
Pour la pagination avec une requête j'ai déterminé le nombre total d'articles puis le nombre d'article que je souhaite par page. je détermine ensuite le nombre de page nécessaire arrondi à l'entier supérieur pour éviter quil manque une page. pour l'affichage j'ai fais un affichage conditionnel : le bouton suivant disparait lorsque l'on arrive sur la dernière page et le bouton précdent disparait lorsque l'on se trouve sur la première page.

3.4 Supprimer un article

```php
Public function deleteArticle($id) {
    require_once('../config/db.php');
    global $bdd;
    $mess_done = "Cet artice a bien été supprimé";
    $req = $bdd->prepare('DELETE FROM article WHERE id = :id');
    $req->bindParam(':id', $id);
    $req->execute();
    return $mess_done;
}
```

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['id'])) {
    $articleId = $_POST['id'];
    $admin = new Admin;
    $result = $admin->deleteArticle($articleId);
}
```

```php
?>
<div class="table-container-articles">
  <table id="tableArticles">
    <tr>
      <th>id</th>
      <th>Nom de l'article</th>
      <th>Prix</th>
      <th>Stock</th>
      <th>id catégorie</th>
      <th></th>
    </tr>
    <?php foreach($showArticle as $value) : ?>
    <tr>
      <td><?= $value['id'] ?></td>
      <td><?= $value['nom_article'] ?></td>
      <td><?= $value['prix'] ?>€</td>
      <td><?= $value['stock'] ?></td>
      <td><?= $value['id_categorie'] ?></td>
      <td>
        <form action="#" method="POST">
          <input type="hidden" name="id" value="<?= $value['id'] ?>">
          <button type="submit" name="deleteArticle">Supprimer</button>
        </form>
      </td>
    </tr>
    <?php endforeach ?>
  </table>
</div>
```

La suppression d'un article s'effectue de manière similaire à l'ajout d'un article.
On affiche les informations de chaque produits dans un tableau grâce la fonction displayArticle.
Dans le tableau on intègre une formulaire avec un bouton supprimer. Puis grâce à la fonction deleteArticle, créée dans la classe Admin, dans laquelle on effectue une requête de suppression on peut supprimer n'importe quel article de la base de données.

3.5 Catégories

```php
public static function displayBonbon(int $id)
{
    require_once('../config/db.php');
    global $bdd;

    $request = $bdd->prepare('SELECT * FROM `article` Where id_categorie = :id');
    $request->bindParam(":id", $id);
    $request ->execute();
    $result = $request->fetchAll(PDO :: FETCH_ASSOC);
    return $result;
}
```

```php
<div class="container">
   <?php foreach($allBonbon as $value) : ?>
    <p id="mess_form"></p>
            <a href= "oneArticle.php?id-product=<?=$value['id'] ?>">
                <div id ="display_articles">
                    <img src="../assets/img/<?= $value['photo'] ?>" alt="img">
                    <h2><?= $value['nom_article'] ?></h2>
                    <p class="sous_titre"><?= $value['sous_titre'] ?></p>
                    <p class ="prix"><?= $value['prix'] ?>€</p>
                    <button type="submit" name ="addPanier" id="addPanier" >Ajoute
                </div>
            </a>
    <?php endforeach ?>
</div>

<?php require_once('../include/footer.php') ?>
</body>
</html>
```

```php
<div class="container">
   <?php foreach($allBonbon as $value) : ?>
    <p id="mess_form"></p>
            <a href= "oneArticle.php?id-product=<?=$value['id'] ?>">
                <div id ="display_articles">
                    <img src="../assets/img/<?= $value['photo'] ?>" alt="img">
                    <h2><?= $value['nom_article'] ?></h2>
                    <p class="sous_titre"><?= $value['sous_titre'] ?></p>
                    <p class ="prix"><?= $value['prix'] ?>€</p>
                    <button type="submit" name ="addPanier" id="addPanier" >Ajoute
                </div>
            </a>
    <?php endforeach ?>
</div>

<?php require_once('../include/footer.php') ?>
</body>
</html>
```

Cette partie du code sert a trié tous les articles par catégories. Ici par exemple la catégorie "Bonbons".
Pour cela j'ai crée dans ma classe Article une fonction qui récupère les articles selon leurs "id_catégorie". Une fois récupéré j'affiche mes articles et leur informations sur ma page bonbon à l'aide d'un "foreach".

3.6 Javascript

```javascript
//Récupérer la valeur du paramètre Get "form"
const urlParams = new URLSearchParams(window.location.search);
const form =urlParams.get('form');

//Afficher le formulaire correspondant
if (form === 'connexion'){
    //Charger le formulaire de connexion dans la div "container-form-sign"
    fetch('../viewer/connexion.php')
        .then(response => response.text())
        .then(data => {
            document.getElementById('container-form-sign').innerHTML = data;
        });
    } else if (form === 'inscription'){
    //Charger le formulaire de connexion dans la div "container-form-sign"
    fetch('../viewer/inscription.php')
        .then(response => response.text())
        .then(data => {
            document.getElementById('container-form-sign').innerHTML = data;
        });
    }
```

Dans ce projet nous avons utilisé du Javascript afin de dynamiser notre site . Dans cette partie du code nous l'avons utilisé afin de pouvoir, lors de l'authentification, afficher le formulaire d'inscription ou de connexion sans rechargement de page.

Nous avons fait de même pour l'affichage des messages d'erreurs qui apparaissent lors de la soumission du formulaire si un problème survient ou si une information rentrée est incorrecte.

4. **Veille sur les vulnérabilités de sécurité**.

4.1 Failles XSS

La faille XSS (Cross Site Scripting) est un type de faille de sécurité des sites Web, quel'on peut retrouvé des les applications Web mal sécurisé.

Le principe de cette faille est d'injescter un code malveillant en Javasxipt dans un site Web vulnérable. par exemple en déposant un message dans un forum qui redirige l'utilisateur vers un faux site (Phishing) ou qii voleses informations.

La faille XSS permet d'exécuter des scripts du coté client.

Plusieurs techniques permettent de se protéger des failles XSS:

- La fonction htmlspecialchars() qui convertit les caractères spéciaux en entités HTML
- La fonction htmlentities() qui est identique à htmlspecialchars() sauf qu'elle filtre tous les caractères équivalents au codage html ou javascript.
- La fonction strip_tags() qui supprime toutes les balises.

```
$login = trim(htmlspecialchars($login));
$email = trim(htmlspecialchars($email));
```

4.2 Injections SQL

Les attaques par injections SQL exploitent les failles de sécurité d'une application qui interagit avec une base de données.

L'attaque SQL consiste à modifier une requête SQL en cours par l'injection d'un morceau de requête non prévue, souvent par le biais d'un formulaire. On peut ainsi accéder à la base de données, mais aussi en modifier le contenue et donc compromettre la sécurité du système.

L'utilisation de requêtes préparés est donc recommandé afin d'éviter les injections.

```php
$request = $bdd->prepare('SELECT * FROM `article` Where id = :id');
$request->bindParam(":id", $id);
$request ->execute();
$result = $request->fetchAll(PDO :: FETCH_ASSOC);
return $result;
```

4.3 Hachage des mots de passe

Afin de renforcer la sécurité sur notre site nous avons choisi de hacher les mots de passe de nos utilisateurs.

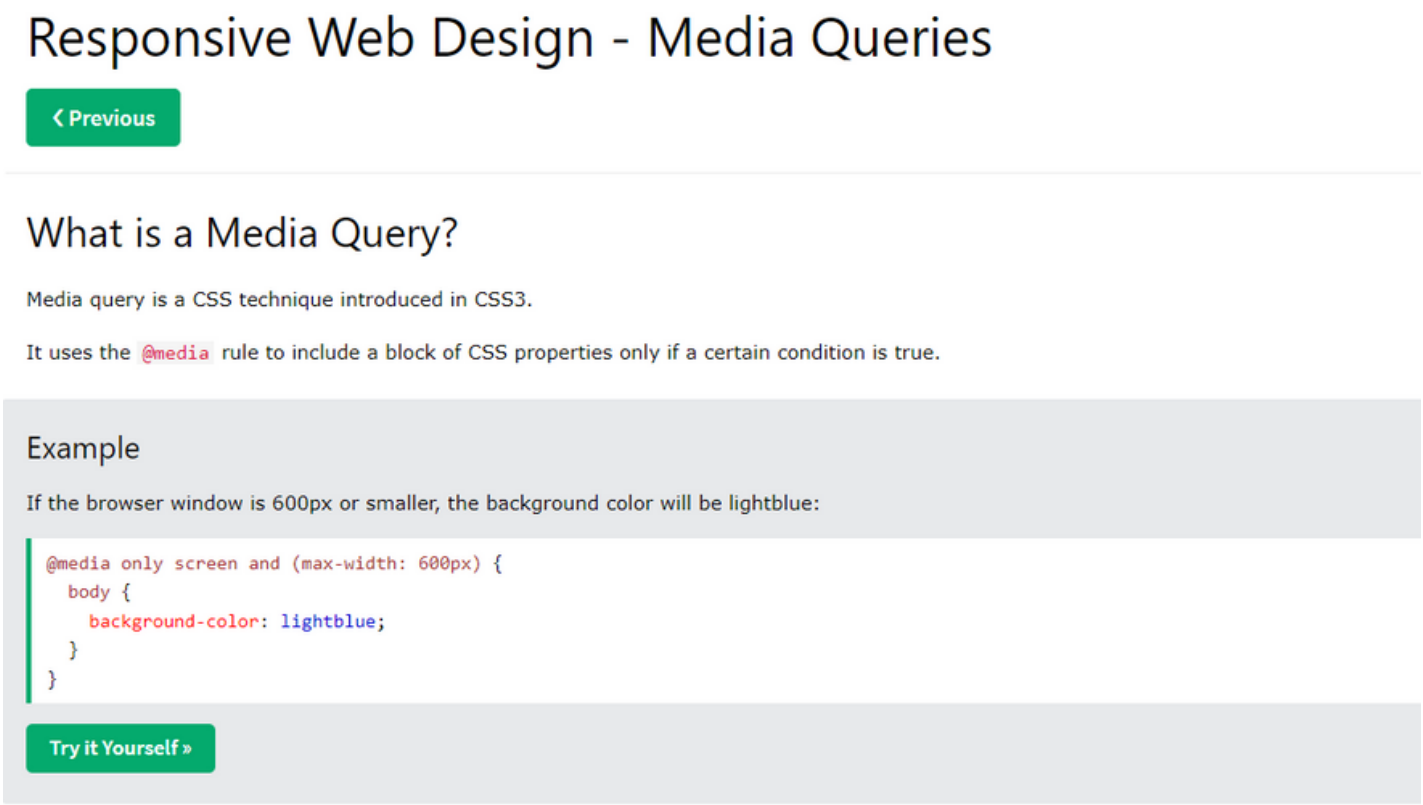
Nous avons utilisé l'algorithme "PASSWORD_BCRYPT" qui est une fonction de hachage de type Blowfish pour stocker de manière sécurisée les mots de passe dans la base de données.

```php
$pass_hash = password_hash($password, PASSWORD_BCRYPT);
```

5. Recherche à partir d'un site anglophone

Durant le projet j'ai dû effectué certaines recherche comme par exemple comment utiliser les media queries pour que mon site soit responsive. J'ai effectué ces recherches en anglais.

Voici un extrait du site "W3 School" qui m'a été utile pour la réalisation du responsive.



L'exemple ci-dessus nous explique que les media queries ont étaient introduits avec la version CSS3.

La condition utilisée dans cet exemple est que si la taille de l'écran est de 600 px ou est plus petit l'arrière plan sera de couleur bleue.