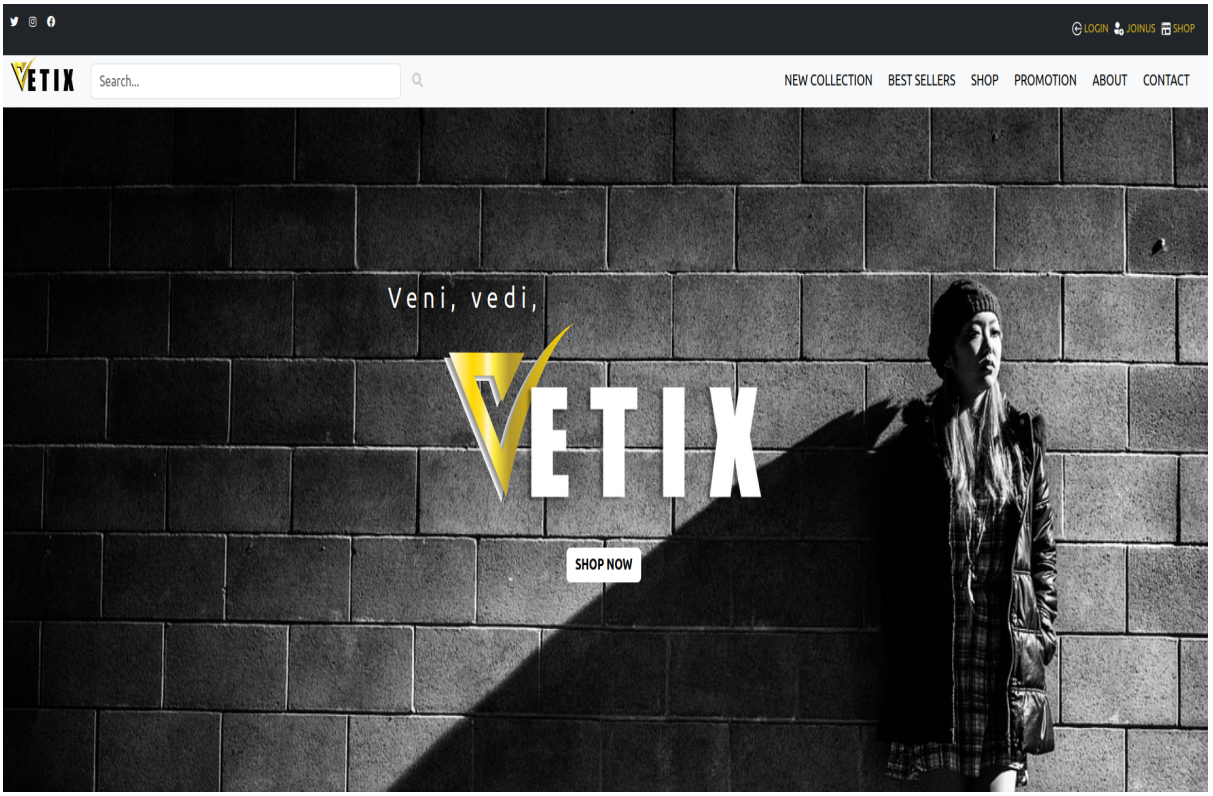
Titre : Développeur Web et Web Mobile

Dossier de projet



Création d’un site de e-commerce

Thomas Spinec

Table des matières

Introduction

Dans le cadre de ma formation à La Plateforme_, j'ai réalisé le projet de boutique en ligne. Ce projet à été réalisé par Jérémy Nowak, Nadia Hazem et moi-même sur une durée de un mois.

Pour le thème de la boutique, nous avons décidé de partir sur une boutique de prêt-à-porter pour femme. Nous nous sommes limités aux femmes car lors de quelques recherches préliminaires d'images pour le site, nous avons remarqué qu'il était plus difficile de trouver des photos professionnelles d'homme, donc dans un souci de gain de temps, nous avons fait le choix de se restreindre aux femmes. Nous avons aussi fait le choix de faire le site entièrement en anglais.

Pour l'organisation, nous avons créé un google chat, et pour le versionning nous avons utilisé github. Des branches ont été créées essentiellement par page sur lesquelles nous avons travaillé.

Les technologies que nous avons utilisées pour le back-end:

- PHP
- Base de données SQL (et utilisation de PDO pour les requêtes)

Les technologies que nous avons utilisées pour le front-end:

- HTML
- CSS
- Javascript pour dynamiser le site et d'améliorer l'expérience utilisateur

Ce projet m'a permis de valider ces compétences :
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.

Fonctionnalités

Front End

- Header adaptatif

Le header du site est adaptatif, il change selon l'état de connexion de l'utilisateur, mais également si la personne connectée est admin ou non.
Le header s'adapte aussi à la taille de l'écran .

- affichage de produits tendances/de la nouvelle collection sur l'accueil

La page d'accueil affiche quelques produits tendances, ainsi que quelques produits de la nouvelle collection, au hasard à chaque fois que l'on retourne sur cette page. Un lien vers le produit en question est disponible, son prix et le bouton d'ajout aux favoris si l'on est connecté.

- page des articles selon la catégorie sans rechargement de page

La page de la boutique affiche par défaut tous les articles, mais des boutons sont présents pour choisir une catégorie et le changement se fait sans rechargement de page. Les articles sont présentés par leur photo, leur prix. Si l'on clique sur la photo, les photos du produit s'affichent en plus grand. Comme sur la page d'accueil, si l'on est connecté, le bouton d'ajout ou de retrait des favoris apparaît.

- barre de recherche

Une barre de recherche avec auto complétion est présente dans le header afin de chercher un produit en particulier.

Back End

- module d'authentification

Le module d'authentification permet de se connecter ou de s'inscrire de manière classique via un formulaire.

- Ajout de Favoris

Lorsque l'utilisateur est connecté, un bouton en forme de coeur apparait sur les fiche produit, ce bouton permet d'ajouter le produits aux favoris grâce à une classe PHP, dans ce cas, le bouton change de couleur, et si l'on clique encore dessus, le produit est retiré.

- Ajout de commentaire

Sur la page de détail d'un produit, les commentaires laissés par les utilisateurs sont affichés, et il est possible de laisser un commentaire si l'on est connecté.

- Panier

Le panier n'est disponible que si l'utilisateur est connecté. Sur cette page, La quantité de chaque produit y est affiché, ainsi que leur prix (à l'unité et au total). On peut modifier la quantité d'un produit, voire le supprimer.

- Panel administrateur

Le panel administrateur permet de gérer les utilisateurs, les supprimer, ou modifier le rôle (user, ou admin). Il n'est pas possible de supprimer l'utilisateur actuellement connecté.

Le panel permet également de rajouter un produit, grâce à un formulaire. La première photo du produit est obligatoire, mais il est possible d'en mettre jusqu'à 3. Il faut choisir une catégorie, et renseigner une taille et le stock qui correspond. Il est possible de mettre plusieurs tailles (et donc plusieurs stocks), mais lorsqu'une taille est sélectionnée, il n'est pas possible de la sélectionner dans un autre select. Sous le formulaire d'ajout de produit, on trouve tous les produits, avec possibilité de trier par catégorie. Il est possible soit de supprimer un produit, soit d'ouvrir une popup qui permet de modifier le stock d'une taille liée à ce produit, ajouter une taille. Mettre le produit dans la catégorie des meilleures ventes et/ou à la catégorie nouvelle collection. Il est également possible de choisir une promotion à appliquer au produit (pas de promotion, 5%, 10%, 20%, 50% ou 75%), choisir une promotion met à jour le prix automatiquement.

Enfin, il est possible de gérer les catégories. On peut en supprimer, modifier, ou en ajouter.

Réalisations

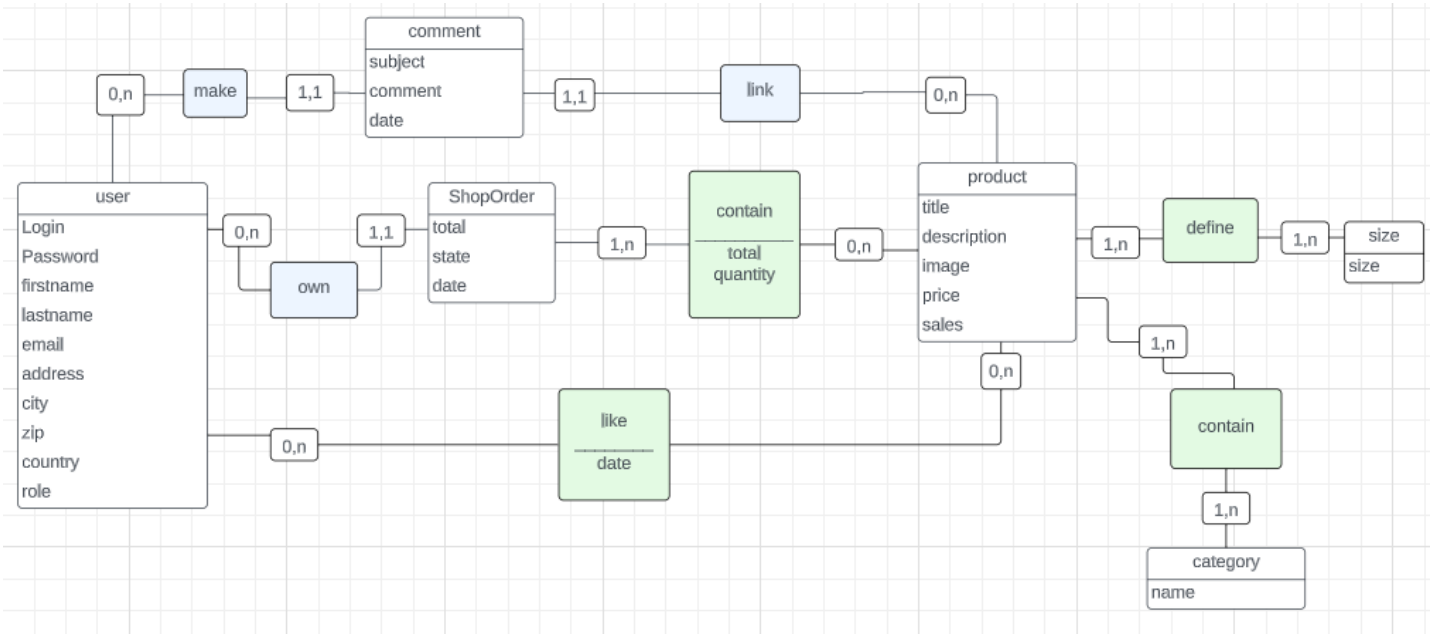
La première semaine de travail sur ce projet a été dédiée à deux étapes très importantes, la conception de la base de données, et la conception graphique.

Conception Base de données (BDD)

Pour cette étape, nous avons utilisé LucidChart. C'est une plateforme de collaboration en ligne permettant la création de diagrammes et la visualisation de données, ce qui en fait un bon outil pour travailler en groupe sur cette étape.

- MCD

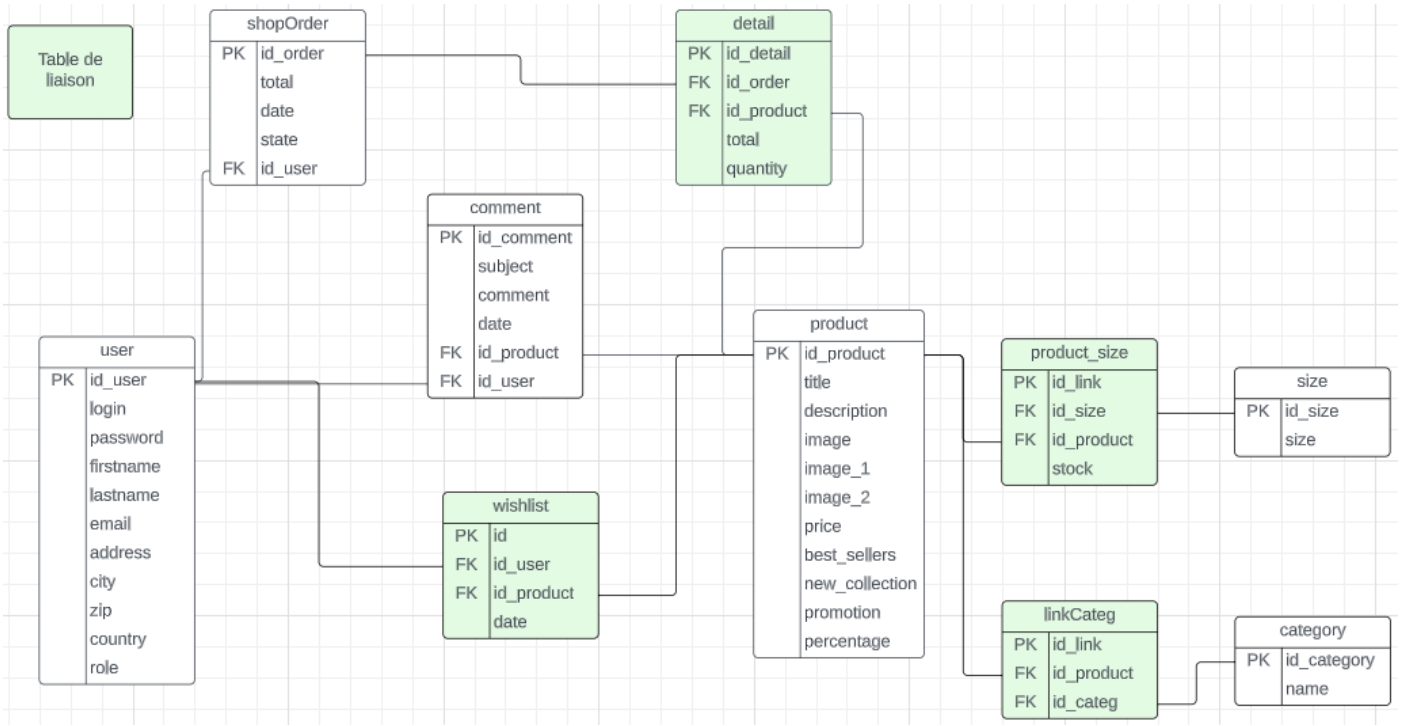
Pour la conception de la base de données, nous avons commencé par le modèle conceptuel de données présenté ci-dessous.



Les verbes qui ont été colorés en vert indiquent qu'il y a une relation many to many, il y aura donc une table de liaison.

- **MLD**

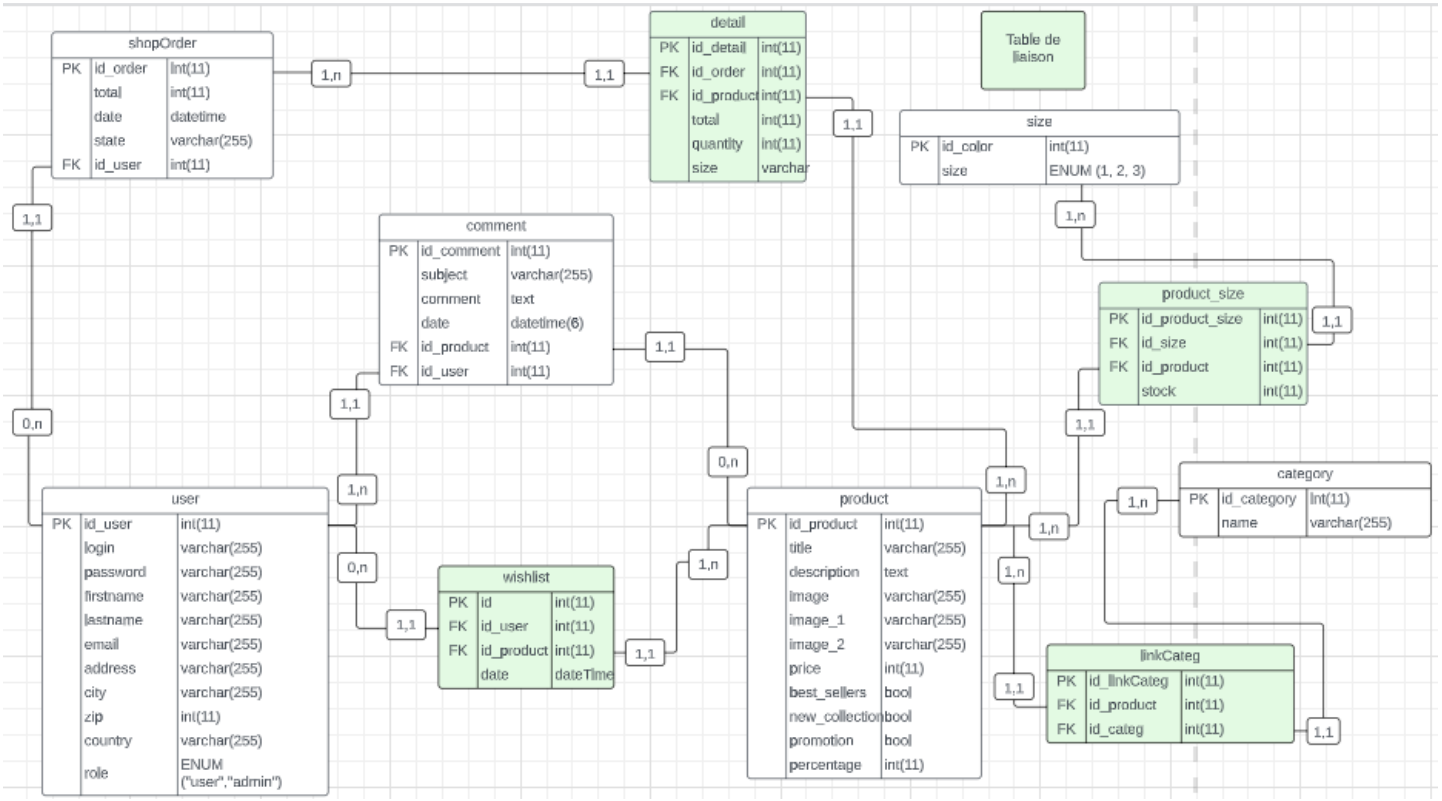
Nous avons ensuite réalisé le modèle logique de données présenté ci-dessous.



Pour le MLD, les cardinalités et les verbes disparaissent, les clés primaires et étrangères apparaissent. Les verbes qui étaient en vert deviennent les tables de liaison de la même couleur.

- **MPD**

Enfin, nous avons réalisé le modèle physique de données.



Comme on peut le voir sur le MPD, le site s'articule autour de 2 tables principales. Tout d'abord, il y a la table user, qui permettra de s'authentifier. Cette table est liée à différentes tables qui permettent de gérer certaines fonctionnalités du site. Il y a par exemple la table "wishlist" qui permet de gérer les favoris , ainsi que la table "comment" pour les commentaires.

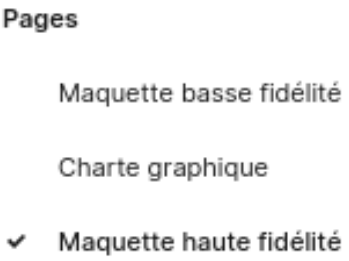
Ensuite la deuxième table principale est la table product. Cette table est reliée à la table "category" via une table de liaison. Elle est aussi liée à la table "size" également via une table de liaison.

Le panier correspond à la table "shopOrder" qui est liée à la table "user" d'un côté et la table "product" via une table de liaison.

Conception graphique

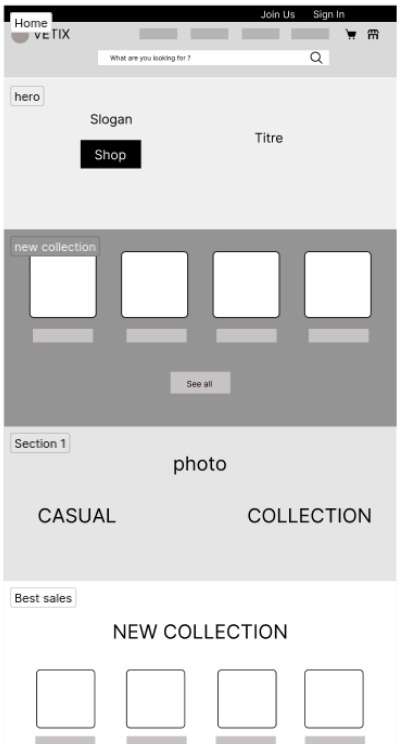
Pour la conception graphique, nous avons utilisé Figma, qui est un éditeur de graphiques vectoriels et un outil de prototypage

Nous avons réalisé la maquette basse fidélité, une charte graphique et la maquette haute fidélité.



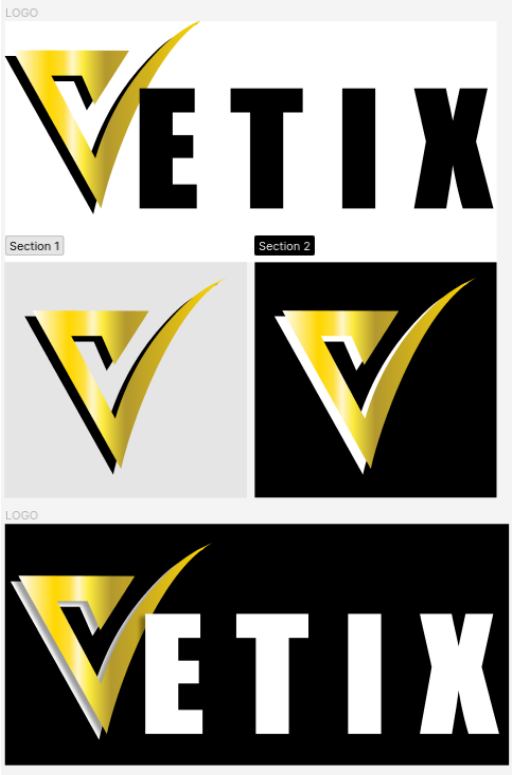
- **maquette basse fidélité**

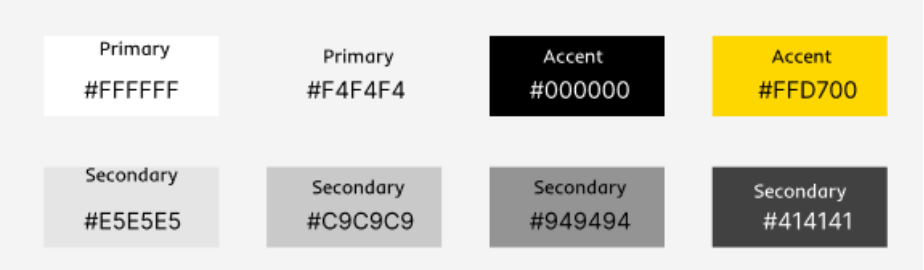
Pour la maquette basse fidélité, nous avons fait une partie mobile et une partie desktop par page, sans couleur. Par exemple avec la page d'accueil ci dessous:



- **charte graphique**

Dans la charte graphique, nous avons les différents logos du site, mais également les différentes typographies, les polices utilisées, etc …

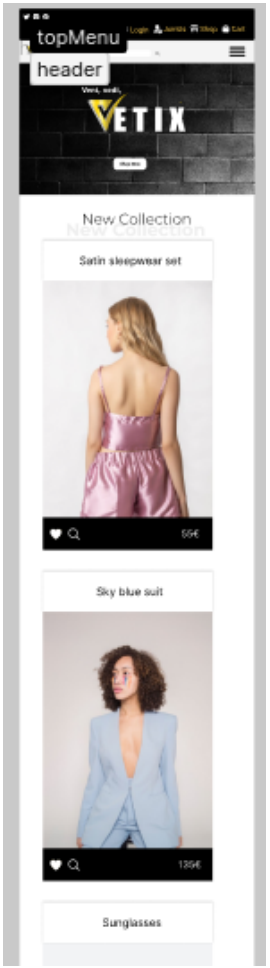


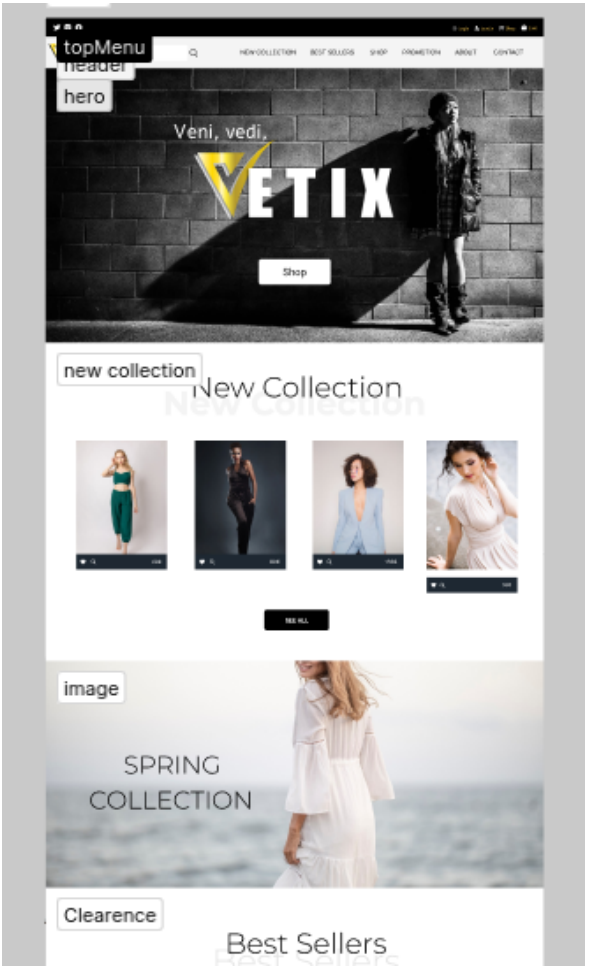


Pour le style, nous voulions qu'il soit assez sobre, mais moderne.

- **maquette haute fidélité**

La maquette haute fidélité reprend la basse, en y ajoutant les couleurs de la charte graphique, ainsi qu'une partie des images que nous retrouvons directement sur le site. Tout comme la basse fidélité, nous avons réalisé une partie mobile et une partie desktop.

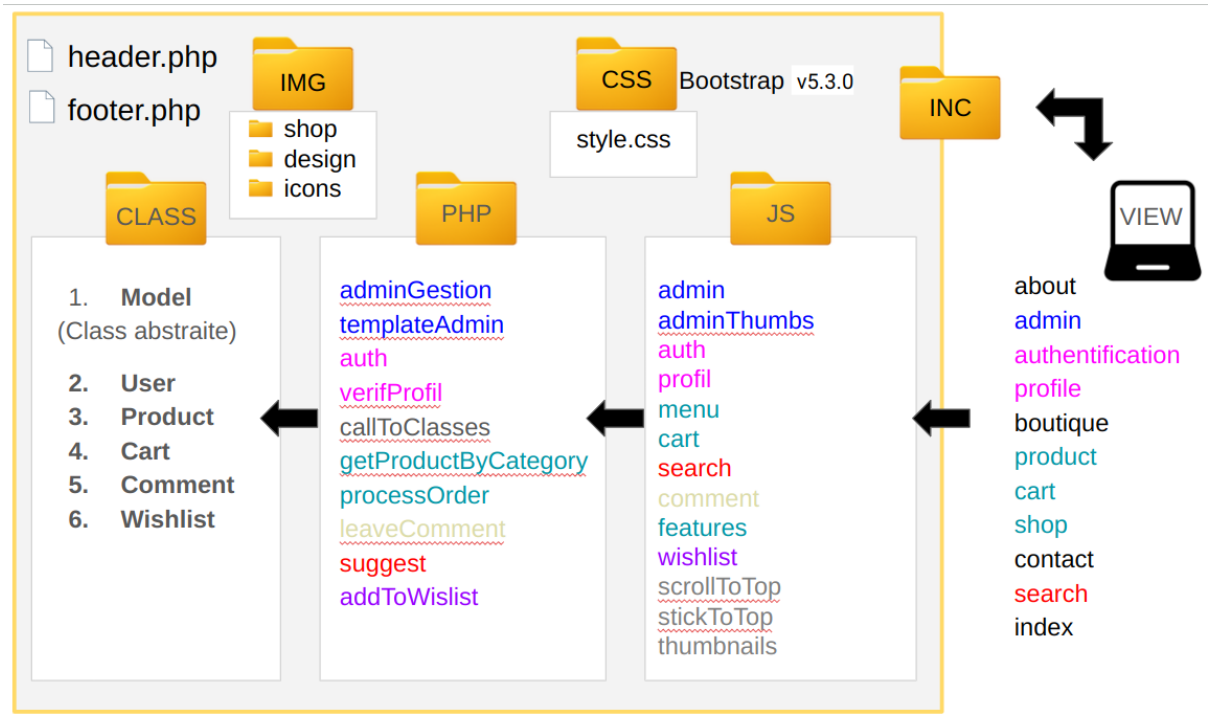




Code

- **Architecture**

Le projet ne suit pas de design pattern particulier, mais les fichier sont organisés comme présenté ci-dessous:



Les pages visibles pas un utilisateur (admin ou non) sont à la racine du projet, le reste est dans un dossier “inc”. Dans ce dossier, les fichiers que nous appelons dans nos pages via des *require* sont directement accessibles (le header ainsi que le footer). Puis nous avons un dossier pour chaque type de fichier (“php”, “js”, “img”, etc …).

- Panier

Pour la gestion du panier, nous avons décidé de le rendre accessible uniquement pour un utilisateur connecté. Nous avons créé une class Cart dans laquelle nous avons implémenté plusieurs méthodes comme par exemple :

- ajouter un produit
- supprimer un produit
- mettre à jour la quantité d’un produit

Exemple avec l’ajout de produit dans le panier :
Pour ajouter un produit dans le panier, nous avons utilisé du javascript pour un rendu dynamique notamment des messages d’erreur s’il y a besoin. Dans ce fichier JS, nous récupérons l’ID du produit via l’URL, puis nous avons un écouteur d'événement sur le bouton d’ajout au panier :

```
cart.addEventListener("click", function (e)
{
    e.preventDefault();
    let quantity = quantityInput.value;
    let size = sizeSelect.value;

    addToCart(id_product, quantity, size);
});
```

Lors du click, nous récupérons la quantité voulue ainsi que la taille, puis nous lançons la fonction “addToCart”. Dans cette fonction, on y retrouve un fetch :

```
function addToCart(id_product, quantity,
size) {
    let data = new FormData();
    data.append("id", id_product);
    data.append("quantity", quantity);
    data.append("size", size);
    data.append("addToCart", "ok");

  fetch("inc/php/process-order.php", {
      method: "POST",
      body: data,
    })
```

Ce fetch mène vers un fichier de traitement qui nous permet de faire plusieurs vérifications. Tout d’abord, on vérifie si l’utilisateur possède déjà un panier (vide ou non), grâce à une méthode de la classe.

```
$id_order = $cart->cartVerify($id);
```

Cette méthode permet de renvoyer l’id du panier de la personne connecté si elle en a déjà un, ou de créer un panier et de renvoyer l’id nouvellement créé si ce n’est pas le cas.

Après avoir récupéré cet ID, on récupère les informations du produit voulu (son prix, s’il y a une promo …).
Quand on a ces infos, on crée le détail dans le panier :

```php
$result = $cart->createDetail($id_order, $id_product, $quantity, $size, $total);
if ($result === "ok") {
    $cart->updateTotal($id_order);
};
```

Dans cette méthode, la première chose que l'on fait c'est de récupérer le détail du produit voulu :

```php
        $request = "SELECT * FROM detail WHERE id_order = :id_order AND id_product =
:id_product AND size = :size";
        $select = $this->bdd->prepare($request);
        $select->execute([
            ":id_order" => $id_order,
            ":id_product" => $id_product,
            ":size" => $size,
        ]);
        $result = $select->fetch(PDO::FETCH_ASSOC);
}
```

Si ce produit existe déjà dans le panier, dans ce cas, on réalise une UPDATE en mettant à jour la quantité. Dans le cas contraire, on fait un INSERT pour ajouter le produit.
Le produit a été ajouté dans le panier, ou alors sa quantité a été mise à jour.

- **favoris**

Nous avons mis en place une wishlist. Nous avons créé une classe du même nom. Lorsque l'on est connecté, un cœur est présent sur la carte d'un produit.

Comme présenté ci-dessus, de base le cœur est blanc, mais si le produit est déjà dans la wishlist, il est doré.

L'ajout dans la wishlist est contrôlé par du javascript, pour changer la couleur du cœur sans rechargement de page.

Tout comme l'ajout dans le panier, le js va lancer la méthode de la classe grâce à un fetch.

```php
    public function addToWishlist($user_id, $product_id)
    {
        $user_id = htmlspecialchars($user_id);
        $product_id = htmlspecialchars($product_id);

        // Check whether the product is already in the user's wishlist
        if ($this->isFavorite($user_id, $product_id)) {
            $this->removeFromWishlist($user_id, $product_id);
        } else {
            // Insert the product into the wishlist table
            $request = "INSERT INTO $this->tablename (id_user, id_product, date)
VALUES (:id_user, :id_product, NOW())";
            $select = $this->bdd->prepare($request);
            $select->execute([':id_user' => $user_id, ':id_product' =>
$product_id]);

            if ($select) {
                echo "ok";
            } else {
                echo "error";
            }
        }
    }     $result = $select->fetch(PDO::FETCH_ASSOC);
}
```

Cette méthode permet de vérifier si le produit est déjà dans la wishlist, si c'est le cas on le retire avec une autre méthode, qui va ni plus ni moins que faire un DELETE de la base de données. En revanche si le produit n'est pas dans la wishlist, un fait un INSERT.

- **commentaires**

Un utilisateur a la possibilité de poster un commentaire sur la page d'un produit, s'il est connecté, mais avant de poster un commentaire, nous récupérons ceux déjà postés (s'il y en a).

Comme montré sur la capture ci-dessous, la méthode récupère les commentaires liés à l'ID du produit. En outre, nous récupérons l'auteur du commentaire, le commentaire, et la date du commentaire que nous formattons grâce à la fonction "DATE_FORMAT". Nous ordonnons ensuite les commentaires par ordre décroissant.

Enfin, la fonction retourne les commentaires s'il y en a, et ces commentaires sont gérés par le js.

```php
    public function getComments($id)
    {
        $id = (int)$id;
        $request = "SELECT comment.*, DATE_FORMAT(comment.date, '- %d %m %Y %H:%i -
') as date, user.login as author
        FROM comment
        INNER JOIN user ON comment.id_user = user.id_user
        WHERE comment.id_product = :id
        ORDER BY date DESC
        ";
        // request
        $select = $this->bdd->prepare($request);
        // exec with params
        $select->execute([
            'id' => $id,
        ]);

        // get results
        $comments = $select->fetchAll(PDO::FETCH_ASSOC);
        // if $result produce an error, we return null
        if (!$comments) {
            return null;
        } else {
            // else we return the result
            return $comments;
        }
    }
```

Pour ajouter un commentaire, nous utilisons la méthode ci-dessous. On commence par passer les paramètres dans la fonction “htmlspecialchars” qui permet d'échapper certains caractères et ainsi se protéger des failles XSS, qui sont des injections de code malveillant.

Ensuite, on prépare la requête pour éviter les injections SQL. On l'exécute avec les paramètres du commentaire, sauf la date que l’on insère avec la requête grâce à la fonction “NOW”.

```php
public function addComment($subject, $comment, $id_product, $id_user)
    {
        // html special chars to avoid injections
        $subject = htmlspecialchars($subject);
        $comment = htmlspecialchars($comment);
        $id_product = htmlspecialchars($id_product);
        $id_user = htmlspecialchars($id_user);

        // request
        $request = "INSERT INTO $this->tablename (subject, comment, date,
id_product, id_user) VALUES (:subject, :comment, NOW(), :id_product, :id_user)";

        $insert = $this->bdd->prepare($request);

        // exec with params
        $insert->execute([
            'subject' => $subject,
            'comment' => $comment,
            'id_product' => $id_product,
            'id_user' => $id_user,
        ]);

        // echo "ok" if the request went well
        if ($insert) {
            echo "ok";
        } else {
            echo "erreur";
        }
        $this->bdd = null;
    }
```

- **ajout de produits**

Pour gérer l'ajout de produit, plusieurs étapes sont nécessaires.

Tout d'abord, nous indiquons le dossier dans lequel les images vont être enregistrées. Nous créons ensuite un tableau contenant les types d'image autorisés, afin de nous protéger des failles d'upload.

La première vérification des images est le type ainsi que sa taille, si cette vérification est bonne, on passe à la suivante.
La deuxième consiste juste à vérifier que le téléchargement s'est bien fait.

On réalise ces étapes pour les deux autres images si besoin, sinon on passe à la suite.

```php
$target_dir = "../img/shop/";

$allowedType = [
    'image/jpeg',
    'image/jpg',
    'image/png'
];

$fileName = $_FILES["imageProduct"]["name"];

$type = $_FILES["imageProduct"]["type"];
$size = $_FILES["imageProduct"]["size"];


if (in_array($type, $allowedType) && $size > 300000) {

    echo "Image too big or type image not allowed";
} else {
    if (!move_uploaded_file($_FILES["imageProduct"]['tmp_name'], $target_dir .
$fileName)) {
        echo "Sorry there was an error";
    }
}
```

Ensuite, nous mettons tous les paramètres dans un tableau.

```php
        $arrayProduct = [
            "title" => $title,
            "description" => $descritpion,
            "category" => $idCategory,
            "size" => $sizeProduct,
            "stock" => $stock,
            "priceEuro" => $priceEuro,
            "priceCentime" => $priceCentime,
            "imgName" => $imgName,
        ];
```

Si plusieurs images, tailles et stock ont été remplis, ils sont également ajoutés au tableau de données.

Une fois ce tableau créé, on lance la méthode de la classe Product.

```php
$product->addProduct($arrayProduct);
```

```php
$title = htmlspecialchars($product["title"]);
$description = htmlspecialchars($product["description"]);
$idCategory = htmlspecialchars($product["category"]);
$size = htmlspecialchars($product["size"]);
$stock = htmlspecialchars($product["stock"]);
$priceEuro = htmlspecialchars($product["priceEuro"]);
$priceCentime = htmlspecialchars($product["priceCentime"]);
$imgName = htmlspecialchars($product["imgName"]);

// change price into centimes
$realPrice = (int)$priceEuro * 100 + $priceCentime;
//Insertion of the product
$request1 = "INSERT INTO $this->tablename (title, description, image, price) VALUES
(:title, :description, :image, :price)";
$insert = $this->bdd->prepare($request1);
$insert->execute([
  ":title" => $title,
  ":description" => $description,
  ":image" => $imgName,
  ":price" => $realPrice,
]);
```

Dans cette méthode, nous utilisons encore une fois la fonction “htmlspecialchars”, puis nous transformons le prix en centime avant de faire la requête d’insertion dans la table Product.

```php
$request2 = "INSERT INTO link_categ (id_product, id_categ) VALUES (:id_product,
:id_categ)";
$insert2 = $this->bdd->prepare($request2);
$insert2->execute([
  ":id_product" => $lastId,
  ":id_categ" => $idCategory,
]);

$request3 = "INSERT INTO product_size (id_product, id_size, stock) VALUES
(:id_product, :id_size, :stock)";
$insert3 = $this->bdd->prepare($request3);
$insert3->execute([
  ":id_product" => $lastId,
  ":id_size" => $size,
  ":stock" => $stock,
]);
```

Vient ensuite les requêtes d’insertion dans les tables liées à la table précédente, la table de liaison entre les catégories et les produits et celle entre les tailles et les produits.

Conclusion

Le projet Vetix est l'aboutissement d'un mois de travail réalisé par Jérémy Nowak, Nadia Hazem et moi-même. Ce projet réalisé à l'école La Plateforme, nous a permis (en plus d'autres projets) de valider toutes les compétences du référentiel.

En outre, ce site possède plusieurs fonctionnalités, telles que:

- une page de boutique permettant de trier les produits par catégorie.
- un panier
- une page profil permettant également d'avoir accès à sa wishlist
- un panel administrateur pour gérer le contenu
- un page accueil mettant en avant quelques produits

En plus de ces fonctionnalités, nous voudrions en ajouter d'autres. Par exemple, une pagination sur la page de la boutique, une vérification du stock qui permettrait d'éviter de rajouter un produit en rupture dans son panier. Il faudrait également ajouter la modération des commentaires, les fonctions sont déjà implémentées, mais pas encore utilisées. Nous avons également d'autres idées qui relèvent plus du détail ou de l'amélioration de l'expérience utilisateur.