Dossier de projet Professionnel

Ribard - Lucas
2023

Titre : Développeur Web et Web mobile

Projet : Boutique-en-ligne

Table des matières

Compétences du référentiel couvertes par le projet

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

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 du projet

Ce projet a été réalisé en collaboration avec Félix Hauger.

Le but de ce projet est de mettre en place une boutique en ligne (Thème et produits au choix). Nous avons choisi de faire une boutique dont le thème est la vente de vêtements, avec un accent sur les saisons.

Voici une liste non exhaustive des fonctionnalités minimales que notre boutique devait respecter :

- Avoir une page d'accueil attractive avec plusieurs sections dont une mise en avant des produits phares / derniers produits mis en ligne.
- Avoir un design contemporain et respectant la charte graphique de votre entreprise, et le site doit être responsive !
- Avoir une barre de recherche de produits avec une autocomplétion en javascript asynchrone
- Accès à la boutique présentant tous les produits avec la possibilité de les filtrer par catégorie / sous-catégories
- Au clic sur chaque produit : une page "détail" complète générée dynamiquement
- (nom, image, prix, description, bouton ajouter au panier...)
- Un système de création de comptes d'utilisateurs
    - Module Inscription / Connexion
    - Page de gestion du profil utilisateur
- Un espace tableau de bord Administrateur:
    - Gestion des produits (Ajout / Suppression Modifications de produits, stocks...)
    - Gestion des catégories et des sous catégories de produits
- Système de validation du panier (une vraie solution de paiement n'était pas nécessaire, une simulation du processus suffit)

Spécifications techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end :

- Le projet sera réalisé avec le langage PHP (Hypertext Preprocessor)
- Base de données MySQL

Technologies utilisées pour la partie front-end:

- Le projet sera réalisé avec du HTML et CSS et Javascript afin de dynamiser le site et d'améliorer l'expérience utilisateur.

Organisation

Du point de vue de l'organisation, nous avons utilisé **Trello** afin de découper le projet en une multitude de tâches à réaliser et de définir leur ordre de priorité.
Nous avons aussi créé un **chat google** pour communiquer, ainsi qu'un **repository Github** pour le versioning de notre code.

Architecture du projet

Nous avons choisi d'utiliser un design pattern de type **MVC (Model-View-Controller)**.
L'architecture MVC est l'une des architectures les plus utilisées pour les applications Web, elle se compose de 3 modules:

- **Modèle :** noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- **Vue :** composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur.
- **Contrôleur :** composant responsable des prises de décisions, gère la logique du code, il est l'intermédiaire entre le modèle et la vue.

Partie 1 : Maquettage

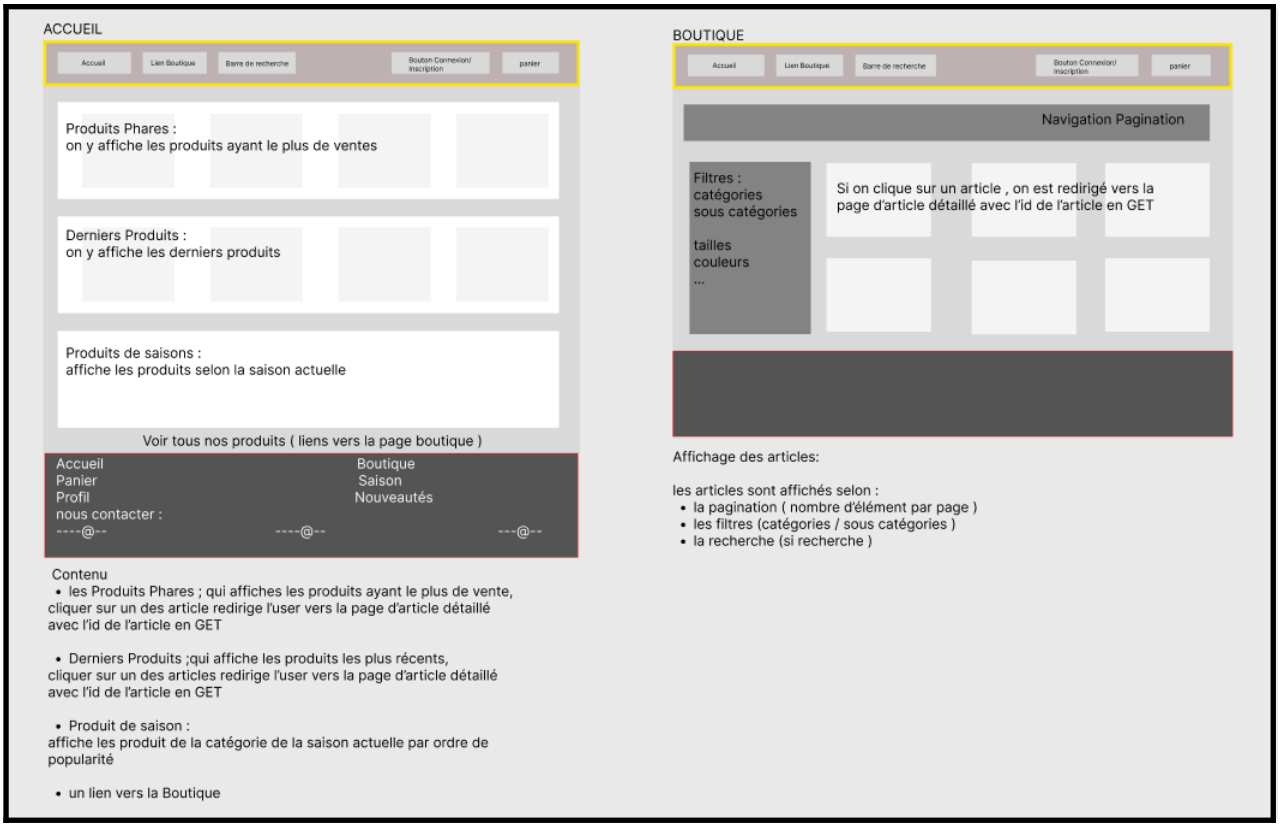
Nous avons commencé ce projet par le maquettage de notre boutique.

Le maquettage est souvent utilisé comme première étape dans un projet pour plusieures raisons :

1. Visualisation du concept : Il fournit une vision claire de ce à quoi ressemblera le projet final et aide à définir les objectifs et les attentes.
2. Communication et collaboration : La maquette facilite la communication entre les membres de l'équipe du projet, elle permet de partager et de discuter des idées, des fonctionnalités et des aspects visuels du projet.
3. Identification des problèmes potentiels : En créant une maquette, il est possible d'identifier les problèmes et les défauts du projet à un stade précoce, avant de passer à la mise en œuvre complète.
4. Validation des exigences : Le maquettage aide à valider les exigences du projet en les mettant en pratique. En créant une maquette interactive, il est possible de tester et de valider les fonctionnalités, les interactions utilisateur et l'expérience globale.

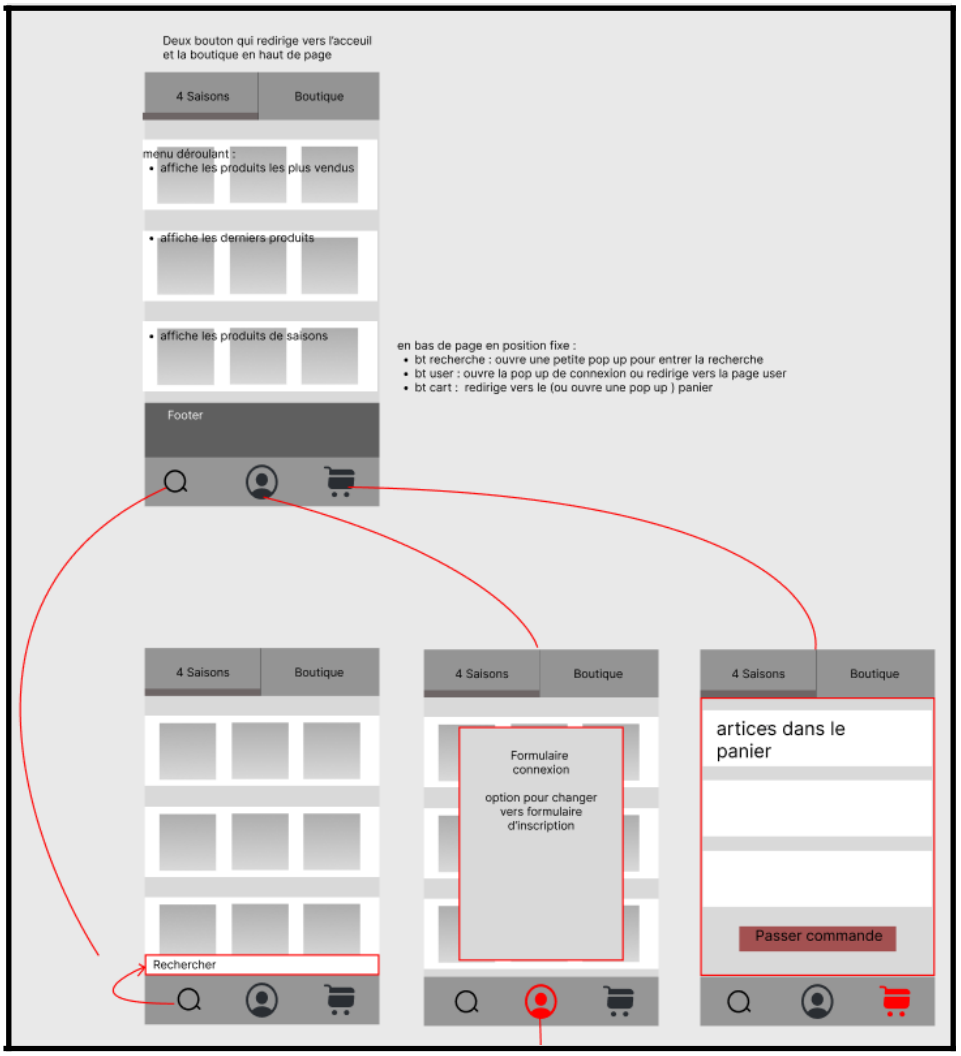
1 - Maquette Wireframe

Pour la maquette de notre projet nous avons utilisé Figma un outil de conception collaboratif de conception d'interfaces utilisateur (UI) et d'expérience utilisateur (UX) basée sur le cloud.



Sur cette partie de la maquette, on peut voir la conception de la page d'accueil et de la page boutique. Bien que leur organisation soit rudimentaire, toutes les fonctionnalités requises ont été soigneusement réfléchies et organisées.

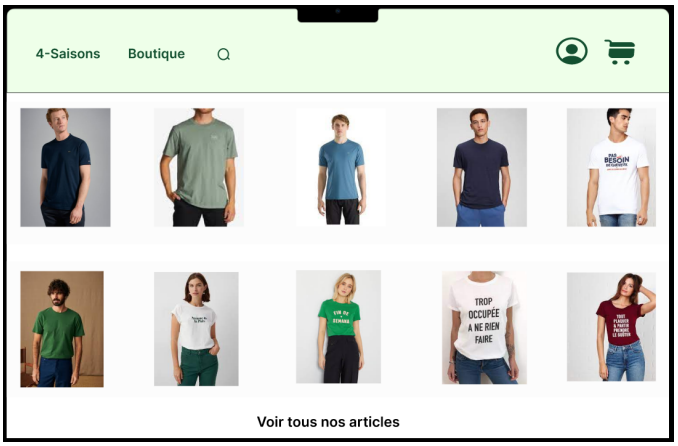
Nous disposons également d'une maquette dédiée à la version mobile, qui intègre tous les changements nécessaires et présente un système de navigation spécialement conçu pour cette version. Cette maquette prend en compte les contraintes et les exigences propres aux appareils mobiles, offrant ainsi une expérience de navigation fluide et optimisée pour les utilisateurs sur leurs smartphones ou tablettes.



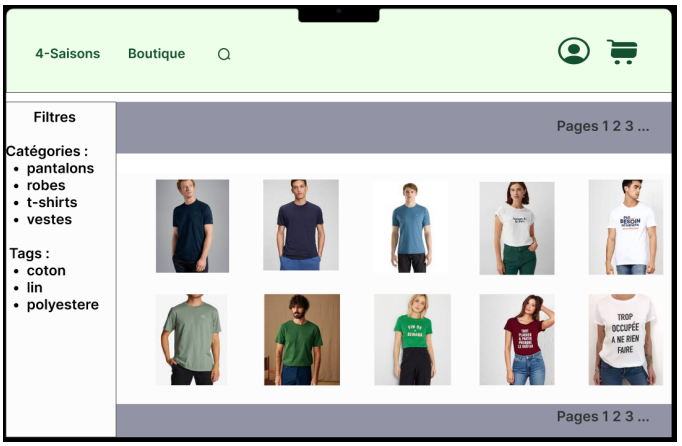
2 - Maquette haute fidélité

Une fois que nous avons établi le concept et défini les différentes fonctions du site, il était primordial de choisir une charte graphique et de développer un design plus poussé. Cette étape cruciale nous a permis de donner une identité visuelle distinctive à notre site et de créer une expérience utilisateur cohérente et attrayante.

Page D'accueil



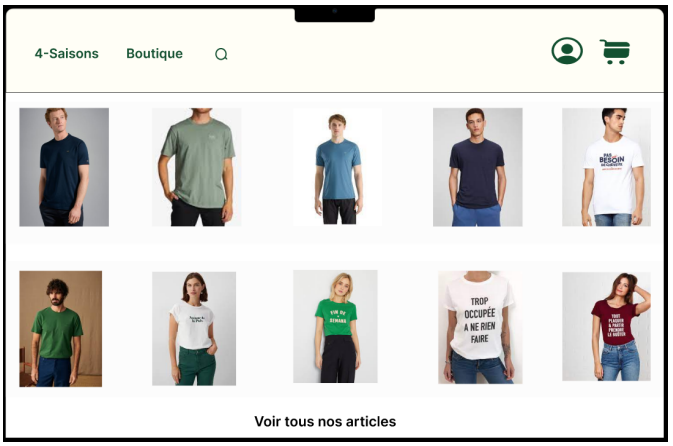
Page Boutique



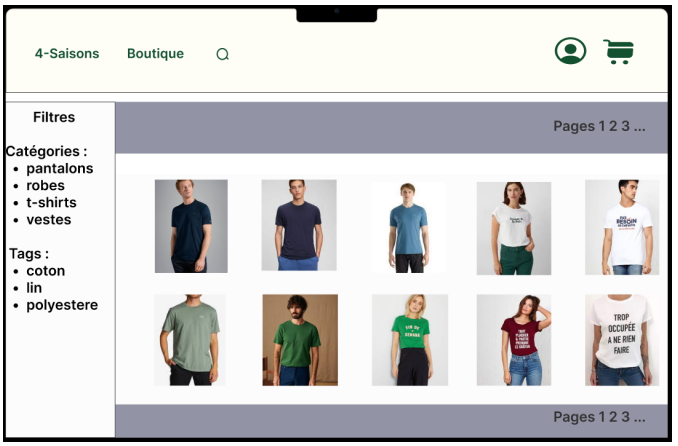
Ici on peut voir les mêmes deux pages que dans l'exemple précédent.

Étant donné que le thème du site repose sur son adaptation aux saisons, nous avons pris soin de créer plusieurs variations de thèmes de couleur pour représenter ces changements. Cette approche nous permet de capturer l'essence et l'ambiance de chaque saison, en offrant aux utilisateurs une expérience visuelle dynamique et immersive.

Page D'accueil



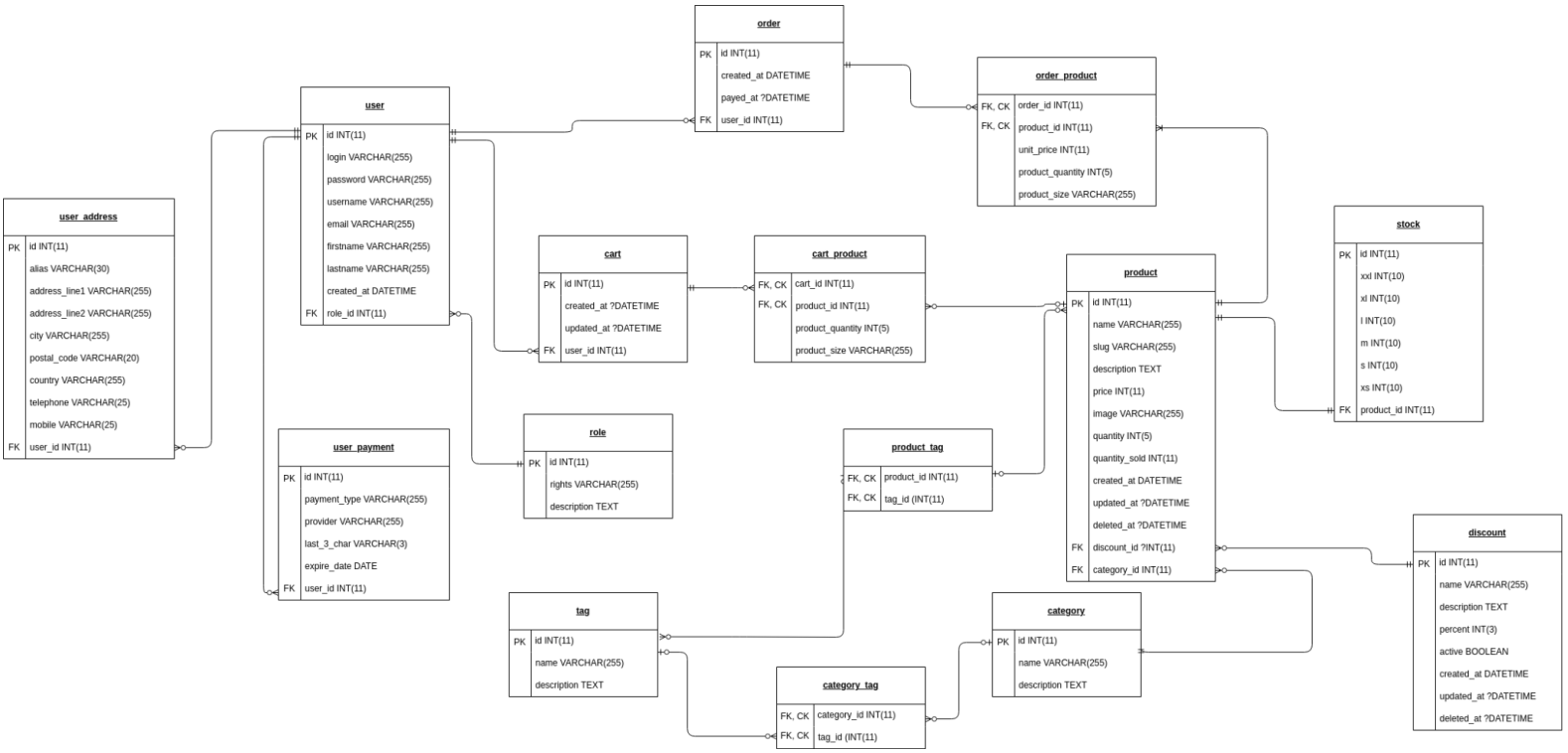
Page Boutique



Partie 2 : Conception de la base de donnée

Au vu des fonctionnalités exigées par le projet, nous avons élaborés la base de données suivante en réponse aux besoins identifiés :

schéma MPD : (Modèle Physique de Données)



La base de données comprend principalement **trois tables**, comme indiqué ci-dessus. La première table, appelée ***"users"***, permet d'identifier les clients. Cette table est liée à d'autres tables telles que *"user_address"* ou *"order"*, ce qui permet de, par exemple, associer une commande à un client spécifique.

Ensuite, il y a la table **"product"** qui est liée aux tables *"stock", "category"* et *"order_detail"*. Ces différentes liaisons permettent de déterminer le stock du produit, sa catégorie, d'afficher les commentaires associés, et lorsqu'une commande est passée, de savoir quels produits composent cette commande.

Enfin, il y a la table ***"order"*** qui permet de recueillir les informations relatives aux commandes passées. Cette dernière est reliée à la table users de manière à pouvoir identifier qui a passé commande. Elle est également reliée à la table *"order_product"* afin d'obtenir les détails de la commande et à la table *"user"* qui permet de récupérer les informations relatives à l'utilisateur qui a passé la commande.

En plus des tables mentionnées précédemment, il existe d'autres tables moins importantes, telles que la table *"cart",* qui est utilisée pour stocker les produits ajoutés au panier par un utilisateur connecté, afin qu'il puisse accéder à son panier même s'il utilise une autre machine.

schéma MCD : (Modèle Conceptuel de Données)

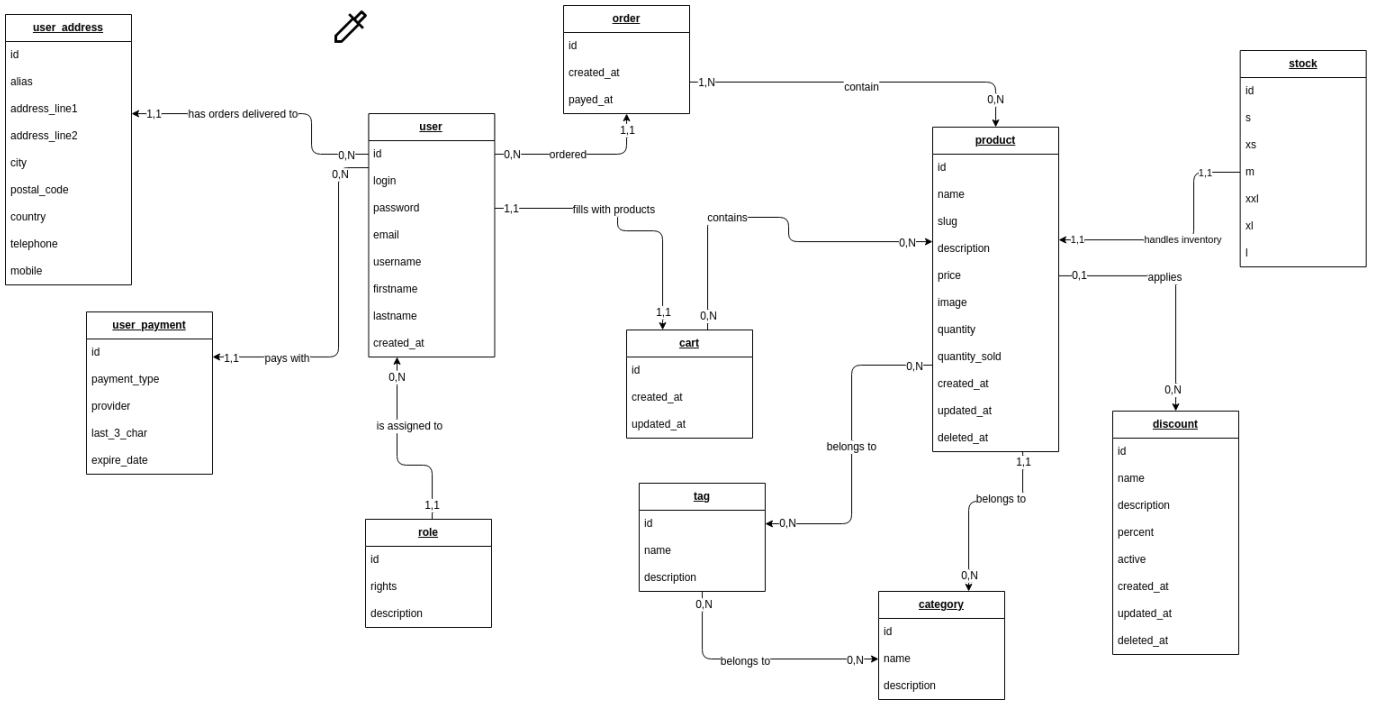
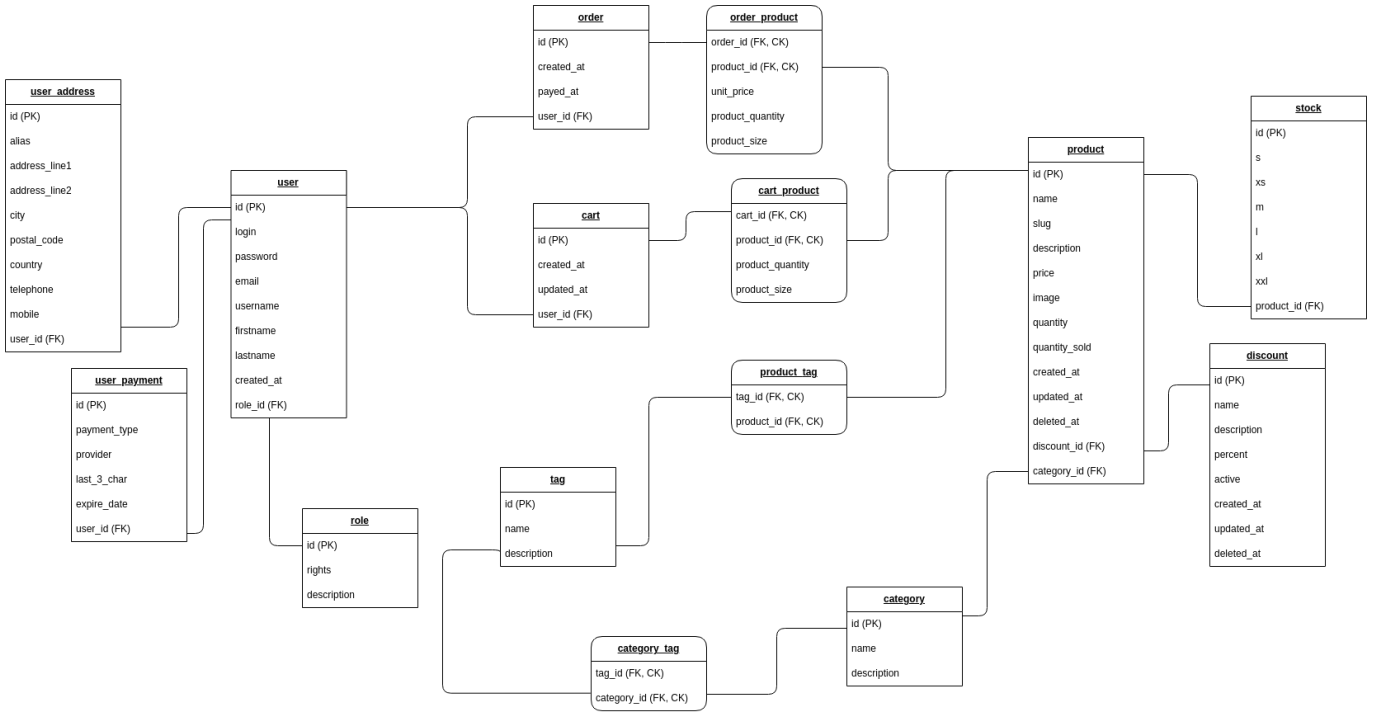


schéma MLD : (Modèle Logique de Données)



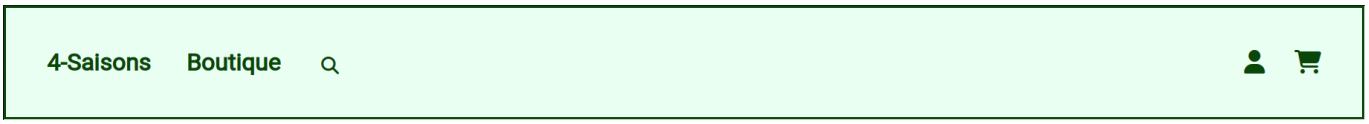
Partie 3 : Front-End

Partie 1 : Le Header

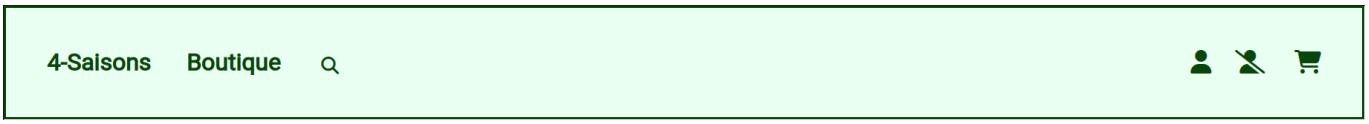
Pour la partie front j'ai commencé par le header qui sera commun à toutes les pages du site.
il est composé des éléments suivants :

liens vers la page d'accueil, la boutique et la barre de recherche

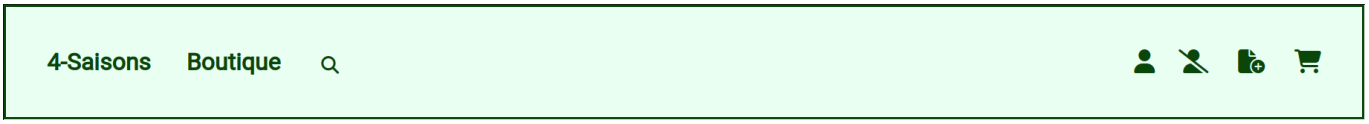
Affichage d'une pop up de connexion / inscription, et le panier



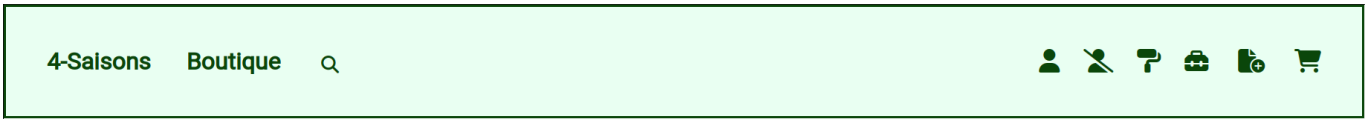
Si un utilisateur sans droits particuliers est connecté, le pictogramme qui affiche la pop up de connexion devient un lien qui mène vers la page gestion d'utilisateur, et un nouveau pictogramme apparaît, qui lui permet de se déconnecter.



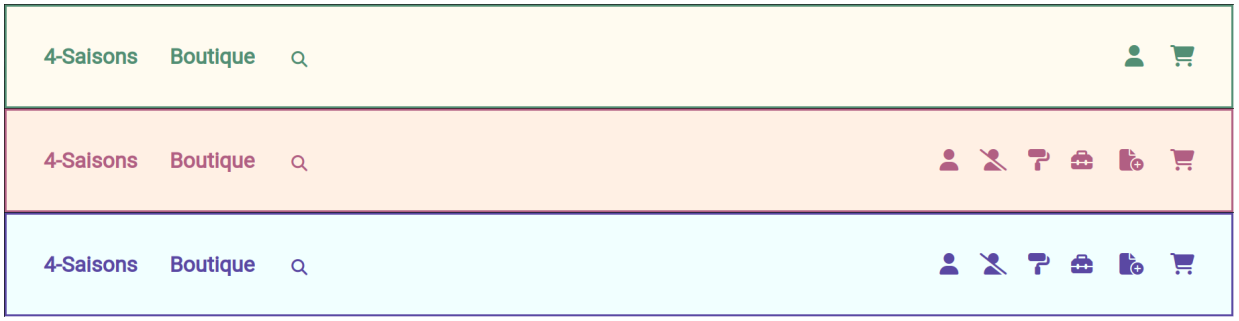
Si un utilisateur possède des droits de "commercial", un nouveau pictogramme apparaît et lui donne accès à une page qui lui permet d'ajouter des produits à la boutique.



Et enfin si un utilisateur à des droits d'administrateur , deux nouveaux pictogrammes apparaissent en plus de ceux mentionnés précédemment, le premier permet à l'administrateur de basculer entre les différentes apparences du site *(rappel : notre site possède un thème par saison)* pour réaliser des test,
et un second qui lui donne accès au panel d'administration.



La disposition du site reste la même selon les thèmes



été

automne

hiver

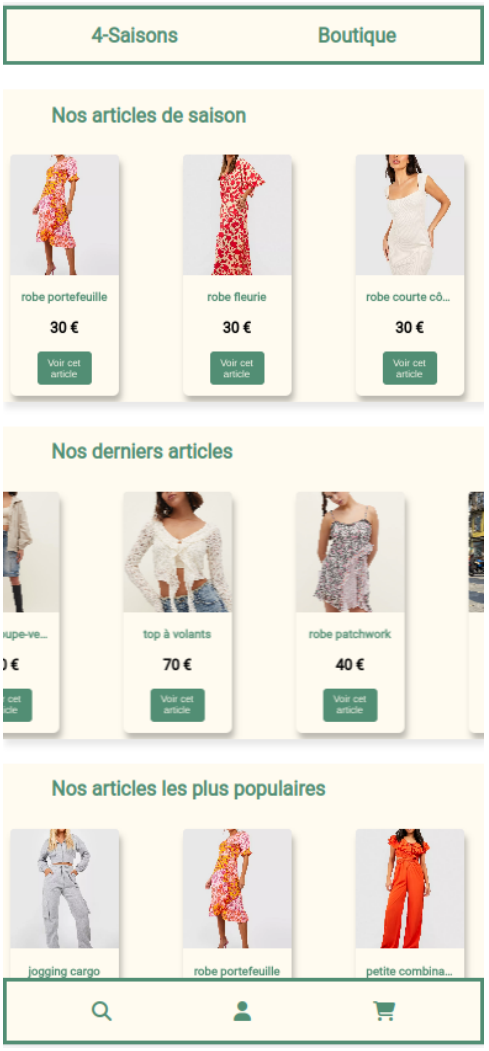
Partie 2 : Responsive

Un design responsive est important sur un site web car il garantit une expérience utilisateur optimale sur tous les appareils, améliore le référencement, élargit la portée du site, simplifie la maintenance et assure une accessibilité accrue pour les utilisateurs.

Pour notre boutique nous avons désigné un interface spécifique aux utilisateurs mobile, elle se différencie de la version pc de part son interface divisé en deux partie : le retour à la page d'accueil , et le lien vers la page d'accueil en haut de page , et la recherche , le lien vers la page profil et le panier fixé en bas de l'écran.

Pour la détection de la taille de l'écran utilisé, on se sert de media queries pour ensuite cibler différentes résolutions et appliquer des styles adaptés.

Les boutons, les icônes et les autres éléments interactifs sont plus grands pour une meilleure expérience tactile, et la police du texte à aussi été agrandie pour faciliter la lisibilité.

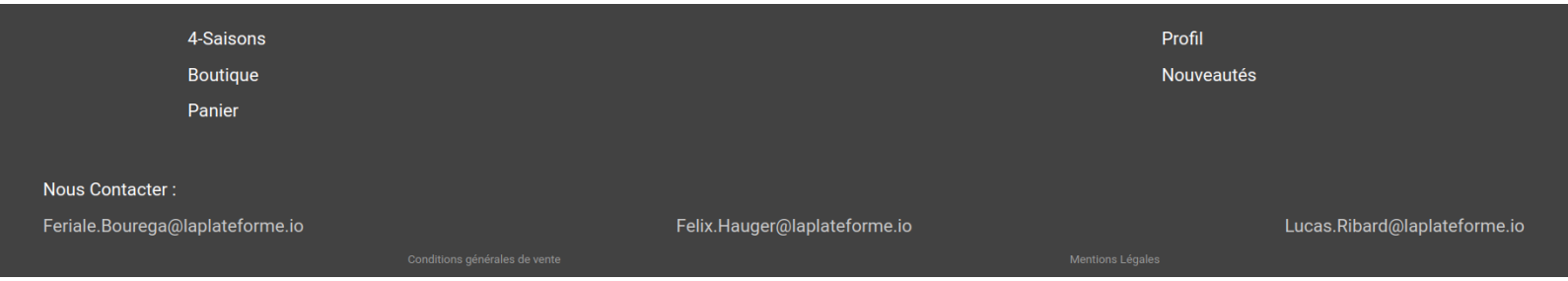


Partie 3 : Le Footer

Le footer d'un site internet est une section située en bas de page qui joue un rôle important en termes de conception et de fonctionnalité. Voici les principales utilisations du footer de notre boutique site :

1. Informations de contact : Le footer est un emplacement courant pour afficher les coordonnées de l'entreprise ou de l'organisation, telles que l'adresse, le numéro de téléphone, l'adresse e-mail, voire des liens vers les profils de réseaux sociaux. Notre boutique étant fictive, elle ne contient que nos adresses e-mails.
2. Liens utiles : Le footer contient des liens vers d'autres ressources utiles, comme des sites connexes, ou des parties du site qui ne sont pas accessibles directement depuis le contenu de la page.
3. Retour en haut de page : Dans les pages web longues, le footer peut inclure un lien "Retour en haut de page" ou une flèche qui permet aux utilisateurs de remonter rapidement en haut de la page sans faire défiler tout le contenu.
4. Copyright et mentions légales : Les informations relatives au droit d'auteur, aux droits de reproduction, conditions générales de vente et mentions légales sont souvent incluses dans le footer. Cela indique clairement les droits de propriété intellectuelle du site et fournit des informations légales importantes.

Le Footer étant un élément commun à toutes les pages, il est donc simple à comprendre et utiliser et possède des liens vers les pages les plus importantes du site.



Le Footer de notre boutique en ligne

Partie 4 : Back-End

Partie 1 : Le Panier

Tous les utilisateurs peuvent voir les articles de la boutique et ajouter des produits à leur paniers, mais l'achat de produits n'est disponible uniquement aux utilisateurs connectées.

Quand un utilisateur non connecté ajoute un produit à son panier, celui-ci est stocké sous forme de cookie, si l'utilisateur décide de se connecter avec un panier, celui-ci sera copié dans la base de données et associé au compte de l'utilisateur, lui permettant de continuer ses achats sans perdre son panier.

Le Panier en utilisateur non connecté (cookie)

Un cookie est composé de deux éléments, un nom et une valeur, mais pour stocker un article nous avons besoin de : l'id de l'article, la taille choisie de l'article et la quantité choisie.
Pour pouvoir enregistrer ces trois valeurs dans un seul cookie, on crée une variable "product", qui est un string composé de : "product_(*id du produit)_(taille choisie)".*

Cette variable deviendra le nom du cookie et la quantité deviendra la valeur du cookie.

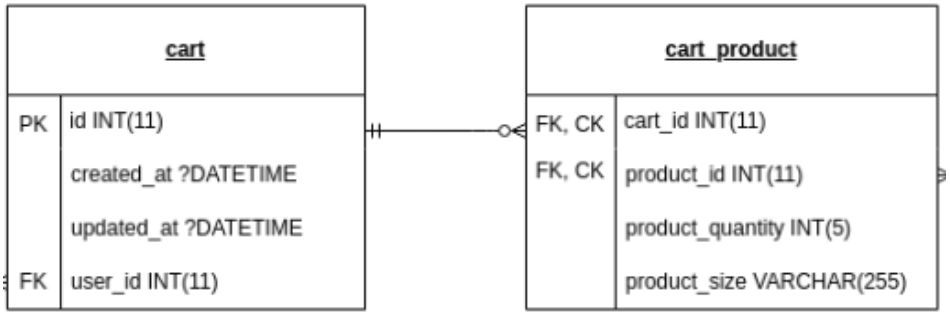
```
function Cookie(id_produit) {
    quantity = document.getElementById("InputQuantity").value
    if (typeof (Size) != "undefined") {
        product = 'product' + '_' + id_produit + '_' + Size;

        document.cookie = product + " = " + quantity + ";expires=Fri, 31 Dec 9999 23:59:59 GMT"
        CartCount()
    }
}
```

Pour récupérer les informations du cookie il suffit d'utiliser la fonction explode avec en paramètre le caractère "_", qui séparera le nom de cookie, et toute les info qu'il contient, et nous permet de récupérer id du produit et la taille du produit dans deux variable utilisable.

Le Panier en utilisateur connecté (bdd)

Dans la base de données, le Panier est formé avec la table “cart” et la table de liaison “cart_product”. Un Panier est ajouté automatiquement à chaque compte utilisateur créé, et une entrée dans la base “cart_product” est créé à chaque article ajouté au panier, ou supprimé quand retiré du panier.

Stocker le panier dans la base de données sert à avoir accès aux articles qui intéressent l’utilisateur même si il visite le site depuis un appareil différent, et l’invité à potentiellement acheter ces produits.



la table cart et cart_product

Transfert d’un panier en cookie vers un panier en bdd

Quand un utilisateur n’est pas connecté, il peut créer un panier, mais ne peux pas passer des achats, mais comme la création de panier en bdd se fait à l'inscription, si un utilisateur s'inscrit pour passer une commande, son panier sera supprimé et il perdra ses achats.
Pour éviter ce problème, à l’inscription on transfère le panier sous forme de cookie au panier qui vient d'être créé en base de données.

Pour transférer un panier de la forme de cookie en forme bdd, on récupère le cookie, on utilise la fonction “explode” pour séparer les informations stockées, et les réinsère dans la table “cart_product”.

```php
foreach($_COOKIE as $key => $value) {
        if (substr($key, 0, 8) === 'product_') {
                // 0 => id, 1 => size
                $product = explode('_', substr($key, 8));

                $quantity = $value;

                // Check if id & quantity are integers & if size has only alphanumeric characters
                if (preg_match('/\d{1,11}/', $product[0]) && preg_match('/[A-Za-z0-9]{1,5}/', $product[1]) && preg_match('/\d{1,5}/',
$quantity)) {
                        // Insert in cart_product
                        $cart_product->create($user_cart->getId(), $product[0], $product[1], $quantity);

                        // Remove cookie
                        setcookie($key, '', -1);
                }

        }
}
```

Partie 5 : Cyber-Sécurité

Les injections SQL

Pour nous protéger des injections SQL, nous avons utilisé PDO (PHP Data Objects) pour chaque requête faite à la base de données.

```php
public function findByUser(int $user_id)    {
        $sql =
            'SELECT
            id, created_at, updated_at, user_id
            FROM cart
            WHERE user_id = :user_id';

        $select = $this->_pdo->prepare($sql);

        $select->bindParam(':user_id', $user_id, PDO::PARAM_INT);

        $select->execute();

        return $select->fetch(PDO::FETCH_ASSOC);
    }
```

Exemple de requête SQL faite avec PDO

Utiliser PDO pour interagir avec une base de données dans PHP offre une protection contre les injections SQL grâce à plusieurs mécanismes intégrés :

1. Requêtes préparées : PDO permet d'utiliser des requêtes préparées, où les paramètres de la requête sont définis à l'avance et séparés du reste de la requête. Cela signifie que les valeurs des paramètres sont traitées de manière distincte des instructions SQL, ce qui empêche les attaques d'injection SQL. Les paramètres sont automatiquement échappés ou encapsulés par PDO selon le type de base de données utilisé.

2. Échappement automatique des données : PDO effectue automatiquement l'échappement des caractères spéciaux dans les valeurs insérées dans les requêtes, tels que les guillemets simples ou doubles. Cela garantit que les données sont correctement traitées et que les caractères potentiellement dangereux sont rendus inoffensifs.

3. Types de données stricts : PDO prend en charge la définition des types de données des paramètres, ce qui garantit que les valeurs fournies correspondent aux attentes de la base de données. Cela empêche également les attaques d'injection en s'assurant que les valeurs sont correctement formatées.

4. Séparation des données et des instructions : PDO encourage une approche où les données et les instructions SQL sont clairement séparées. Cela rend plus difficile pour un attaquant de manipuler les requêtes en insérant du code malveillant dans les données fournies par l'utilisateur.

Les failles XSS

Lorsqu'un utilisateur saisit des données dans un champ de saisie, tel qu'un formulaire, il est possible qu'il insère intentionnellement ou accidentellement des balises ou des caractères spéciaux qui peuvent être interprétés comme du code HTML ou JavaScript lorsqu'ils sont affichés sur une page web. Cela peut permettre à un attaquant d'injecter du code malveillant dans le site et d'exécuter des actions non autorisées dans le navigateur des autres utilisateurs.

L'utilisation de FilterSpecialCharacters() permet de convertir ces caractères spéciaux en entités HTML, rendant ainsi leur interprétation inoffensive lors de l'affichage sur une page web. Par exemple, les caractères <, >, &, ', " seront convertis en “&lt”;, “>”, “&”, “'”, “"”, respectivement.

```php
public function filterSpecialCharacters($string): string {
        return htmlspecialchars(trim($string), ENT_QUOTES);
    }
    public function filterMethodArgs(string $class_name, string $method_name, array $args): array{
        // Get the method information using reflection
        $method = new \ReflectionMethod($class_name, $method_name);

        // Filter each argument using the filterSpecialCharacters method
        $filtered_args = array_map([$this, 'filterSpecialCharacters'], $args);

        // Combine the argument names and filtered arguments into an associative array
        $args_names = array_map(fn($param) => $param->name, $method->getParameters());

        $filtered_args = array_combine($args_names, $filtered_args);

        return $filtered_args;
    }
```

En utilisant cette approche, les caractères spéciaux sont neutralisés et affichés simplement comme du texte brut sur la page web, ce qui empêche leur interprétation en tant que code exécutable.

Protection contre l’upload de fichiers malveillants.

Pour empêcher l’upload de fichier malveillant lors de l’ajout d’une image à la création d’un nouvel article, on limite l’upload a certains types de fichier, la taille maximale du fichier et le nom du fichier.

```php
public function getImageFile(array $image_file, string $destination_path) {
        // Test if file exists and has no error
        if (isset($image_file) && $image_file['error'] === 0) {

            $image_infos = pathinfo($image_file['name']);
            $image_name = $image_infos['filename'];

            if (!$this->checkUploadedFileNameLength($image_name)) {
                // Limit image file name length
                throw new Exception('Nom du fichier trop long. Longueur max : ' . Product::MAX_UPLOADED_FILE_NAME_LENGTH);

            } elseif (!$this->checkUploadedFileName($image_name)) {
                // Limit which characters image file name can contain
             throw new Exception('Seuls des caractères alphanumériques sans accent ni tirets sont acceptés pour le nom de l\'image');

            } elseif (!$this->checkUploadedFileSize($image_file['size'])) {
                // Limit image size
                throw new Exception('Taille maximum de l\'image : ' . static::$_max_uploaded_file_size / 1000000 . 'mo');
            }
```

broken access control

Pour empêcher n’importe quel utilisateur d'accéder aux parties protégées du site, certaines page contient des vérification de droit d'accès qui redirige l’utilisateur vers une autre page si il n’a pas les autorisations nécessaires.

```php
if (!isset($_SESSION['user'])) {
    http_response_code(403);
    header('Location: index.php');
    die();
} elseif ($_SESSION['user']->getRoleId() == 2) {
    http_response_code(403);
    header('Location: index.php');
    die();
}
```

Partie 6 : Situation de travail

Architecture MVC

Pour notre projet, nous avons décidé d'utiliser une architecture mvc, n'étant pas familier avec ce type d'architecture, j'ai dû me renseigner.

Un des sites que j'ai visité est le suivant : https://www.geeksforgeeks.org/mvc-framework-introduction/

Traduction de l'article :

(Certaines expressions ne sont pas aussi claires en français, ce qui a conduit à prendre certaines libertés lors de la traduction.)

Au cours des dernières années, les sites internets ont changé de simple pages HTML avec un peu de CSS vers des applications incroyablement complexes avec des milliers de développeurs y travaillant en même temps.
Pour travailler avec ces applications, les développeurs utilisent différentes structures de design, pour étaler leurs projets, rendre leur code moins complexe et faciliter leur mise en œuvre.
La structure la plus populaire est la structure MVC, aussi connue sous Model View Controller.

La structure MVC est une structure architecturale qui sépare une application en trois composants logiques principaux : ***Modèle Vue*** et ***Contrôleur*** (***Model***, ***View***, ***Controller***).
Chaque composant architectural est construit pour gérer un aspect spécifique du développement d'une application.
Elle sépare la logique métier et la couche de présentation. La structure MVC était traditionnellement utilisée pour les interfaces graphiques d'applications de bureau. Mais de nos jours, MVC est une des standard de l'architecture web les plus utilisées, pour créer des applications évolutives capables d'être facilement améliorées.
Elle est aussi utilisée pour construire des applications mobiles.

MVC à été créé par **Trygve Reenskaug.** Le but principal de cette architecture est de résoudre le problème des utilisateurs manipulant une application complexe, en séparant cette application en plusieurs sections spécifiques qui ont chacune leur utilité.

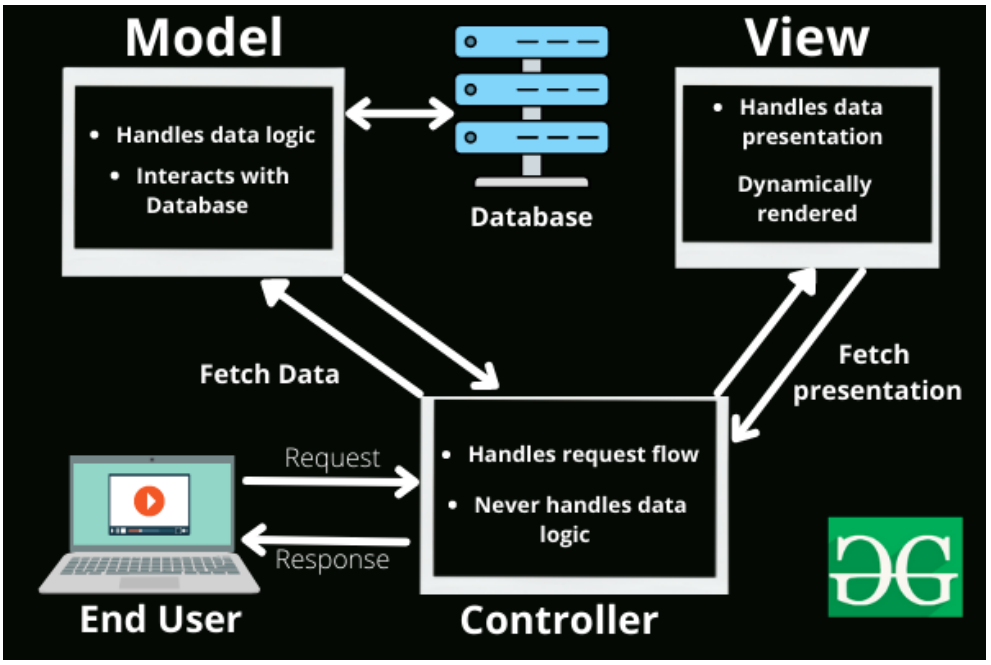
Caractéristiques du MVC :

- apporte une séparation claire entre la logique métier, la logique d'interface, et la logique d'entrée.
- Il offre un control total de votre HTML et notre URL, ce qui facilite la conception de l'architecture de notre application.
- C'est un composant de marquage d'URL puissant qui permet de concevoir des applications qui ont des URL compréhensibles et recherchables.
- Il supporte le développement piloté par les tests ou Test-Driven Development (TDD).

Composant du MVC :

La structure MVC inclut les 3 composant suivants :

- ***Contrôleur*** (Controller)
- ***Modèle*** (Model)
- ***Vue*** (View)



Contrôleur :
Le ***Contrôleur*** est le composant qui permet l'interconnexion entre la ***Vue*** et le ***Modèle***, il agit donc d'intermédiaire.Le ***Contrôleur*** n'a pas à se soucier de gérer la logique de donnée, il dit juste aux ***Modèle*** quoi faire.Il traite toute la logique metier et requête entrante, manipule les données en utilisant les composant du ***Modèle*** et interagit avec la ***Vue*** pour générer le rendu final.

Vue :
Le composant ***Vue*** est utilisé pour la logique d'interface graphique de l'application. Il génère une interface pour l'utilisateur. Les Vues sont créées à partir des données collectées par le composant ***Modèle***, mais ces données ne sont pas importées directement mais à travers le ***Contrôleur***. Il n'interagit qu'avec le ***Contrôleur***.

Modèle :
Le Composant ***Modèle*** correspond à toute la logique de donnée avec laquelle l'utilisateur interagit. Cela peut représenter soit les données transféré entre les composant ***Vue*** et ***Contrôleur*** soit n'importe quel autre données lié à la logique métier . Il peut ajouter ou récupérer des données depuis la base de données. Il répond au requête du ***Contrôleur*** car le ***Contrôleur*** ne peut pas interagir avec la base de données. Le ***Modèle*** interagit avec la base de données et renvoie les données au ***Contrôleur***.

Conclusion :

En conclusion, cet article offre une explication claire et accessible du fonctionnement de l'architecture Modèle-Vue-Contrôleur (MVC) et des relations entre ses différents composants. Le Modèle représente la partie de l'application qui gère les données et la logique métier, tandis que la Vue est responsable de l'interface utilisateur, présentant les données de manière conviviale. Le Contrôleur agit comme un médiateur entre le Modèle et la Vue, traitant les interactions de l'utilisateur et coordonnant les mises à jour nécessaires.

L'utilisation de l'architecture MVC est largement répandue dans le développement logiciel, car elle présente de nombreux avantages. En séparant la logique métier, l'interface utilisateur et le traitement des interactions, le MVC favorise la modularité et la maintenabilité du code. Il facilite également le travail en équipe, car différents développeurs peuvent se concentrer sur des parties spécifiques de l'application sans interférer avec les autres.

En comprenant les principes fondamentaux du MVC, les développeurs sont en mesure de créer des applications robustes et évolutives. Ils peuvent apporter des améliorations à la Vue sans altérer le Modèle, ou ajouter de nouvelles fonctionnalités dans le Contrôleur sans perturber l'interface existante. Cette séparation des responsabilités contribue à rendre le code plus propre, flexible et compréhensible.

En résumé, maîtriser l'architecture MVC est un atout précieux pour tout développeur souhaitant créer des applications efficaces et bien structurées. En combinant une bonne compréhension des principes de conception avec une pratique régulière, les développeurs peuvent exploiter pleinement le potentiel de cette architecture et créer des applications qui répondent aux besoins des utilisateurs de manière claire et intuitive.