

DOSSIER PRO



Titre Professionnel Développeur Web/Web Mobile

Nom : RACHEDI
Prénom: Khellaf

Compétences couvertes par le projet	3
Présentation de l'entreprise	6
L'objectif	6
Cible	6
Périmètre du projet	7
UX / UI	7
Définition du parcours utilisateur (arborescence)	7
Définition des fonctionnalités	8
Charte graphique	9
Maquettes	10
Spécificités techniques ; Back-end	14
Environnement de développement et organisation	14
Technologies	14
Workflow	14
Modélisation de la base de données	15
Méthode MERISE	15
Modèle conceptuel de données (MCD)	15
Modèle Logique de Données (MLD)	15
Modèle Physique de Données (MPD)	16
Architecture logicielle: Backend	17
Extraits de code significatifs	17
Spécificités techniques: Front-end	22
Intégration	22
Responsive design	23
Javascript	25
Veille effectuée sur les vulnérabilités de sécurité	26
Résumé des recherches	26
Les failles de sécurité	27
Les injections SQL	27
Description	27
Solutions apportées	27
Cross-site scripting (XSS)	27
Description	27
La faille Upload	28
Description	28
Solutions apportées	28
Extrait anglophone	29
Traduction	29
UX/UI - Annexe 1	31
Workflow - Annexe 2	32
Modèle conceptuel de données (MCD) - Annexe 3	33
Modèle Logique de Données (MLD) - Annexe 4	34

1. Introduction

1.1. Compétences couvertes par le projet

Le projet que je vais vous présenter a été réalisé dans le cadre de ma formation au sein de la Coding School de la Plateforme_ à Marseille.

Je vais vous exposer ci-après les compétences du référentiel couvertes par le développement de mon projet dans sa totalité.

- **Activité type 1 :**

- **Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité**

- ✓ **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**

Les compétences liées à l'activité de type 1 sont abordées au travers de :

- **Réflexions apportées à l'UI/UX pour le parcours utilisateur**
- **La réalisation des maquettes**
- **L'intégration de la maquette dans le projet**

- **Activité type 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**

- ✓ **Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce**

Les compétences liées à l'activité de type 2 sont abordées au travers de :

- La conception et la modélisation de base de données utilisant la méthode **MERISE**.
- L'utilisation de la **POO** permettant l'interaction avec les données présentes en **BDD** pour l'utilisateur

1.2. Résumé du dossier de projet

Ce dossier présente l'ensemble du travail que j'ai effectué, à savoir la création du site e-commerce fictif Iced Watches, une entreprise qui développe une boutique en ligne. Le site vise à proposer l'acquisition de montres de haute horlogerie uniques. Le site est constitué d'une landing page présentant les services proposés par Iced Watches.

Tout visiteur du site aura la possibilité de consulter les catégories de produits et les fiches des produits et ainsi les ajouter au panier.

Cependant pour procéder à l'achat d'un ou plusieurs produits, il devra s'inscrire puis se connecter pour ainsi procéder à l'achat et finaliser sa commande.

Le visiteur devient un utilisateur et celui-ci peut en plus accéder à son espace client nommé "profil". Il aura la possibilité de consulter et modifier ses informations personnelles, mais aussi de consulter son historique de commande(s).

Le site contient aussi un panneau d'administration qui est entièrement réservé à l'administrateur du site qui lui confère la possibilité de gérer les utilisateurs du site, d'ajouter, modifier et supprimer des produits etc...

Ce site web comptant comme projet de mise en situation professionnelle pour le passage du titre professionnel de développeur web et web mobile. Il a été réalisé au cours de ma formation de développeur web à l'école La Plateforme à Marseille.

3. Cahier des charges

3.1. Présentation de l'entreprise

Iced Watches est le nom du e-commerce fictif spécialisé dans la vente au détail de montres de luxe. Iced Watches est née de la volonté d'un passionné de haute horlogerie de proposer à la vente des pièces d'exception de montres à d'autres passionnés.

En effet, les montres proposées à la vente sont issues de grands horlogers Suisses réputés pour la particularité de proposer un savoir-faire unique, des produits fiables et durables.

Techniquement ces montres sont surtout prisées pour leurs mouvements qui renferment des vrais trésors horlogers à travers les différentes complications.

Les matériaux utilisés sont nobles et travaillés et sont pour l'immense majorité assemblés manuellement.

Iced Watches n'est pas seulement de la vente de produits très haut de gamme, c'est aussi une véritable expérience offerte à ses clients.

A l'image des produits, Iced Watches propose des services de haute qualité.

Chaque commande est personnellement livrée par un commercial qui se déplace chez le client pour lui remettre en main propre le produit commandé tout en détaillant les fonctionnalités du produit.

Outre un service client d'exception, chaque produit est livré avec son certificat d'authenticité dûment complété par un maître horloger.

3.2. L'objectif

L'objectif est de mettre en place un marketplace permettant à des particuliers d'acquérir des pièces rares de montres. Prévue pour être utilisée sur un ordinateur, une grande importance est apportée à l'adaptation et à la compatibilité du service sur différents supports. Notamment les appareils mobiles représentant la plus grosse partie du trafic internet.

3.2.2. Cible

Comme présenté plus tôt, différents types d'utilisateurs sont présents sur le site. Il existe donc trois cibles:

❖ Visiteur :

- Utilisateur non authentifié sur le site qui peut consulter les produits. Potentiellement un nouvel utilisateur qui va s'inscrire. Il a accès au panier mais sans pouvoir finaliser une commande.

❖ Inscrit :

- Utilisateur authentifié sur le site, il peut consulter les produits, ajouter des produits au panier, finaliser sa commande en procédant au paiement et peut par la suite suivre sa commande via l'historique accessible depuis son espace profil. Enfin, il pourra modifier et mettre à jour ses informations personnelles dans le profil.

❖ **Administrateur :**

- Utilisateur authentifié avec des droits d'accès spécifiques au site. Il a accès à un espace qui lui est entièrement dédié : le panneau d'administration. Il pourra modifier, supprimer ou ajouter un ou plusieurs produits. Il peut aussi accéder à l'historique de toutes les commandes. Il peut également supprimer un utilisateur inscrit et modifier ses droits.

3.2.3. Périmètre du projet

Le site sera décliné en français, et sera complètement responsive pour être disponible sur desktop & mobile.

3.3. UX / UI

3.3.1. Définition du parcours utilisateur (arborescence)

Un parcours utilisateur est une visualisation des relations d'un individu avec un site web (dans le cadre de mon projet) au fil du temps.

Pour pouvoir le définir, il est nécessaire d'établir un scénario étape par étape des interactions possibles entre un utilisateur et le e-commerce.

Je l'ai modélisé sous forme d'arborescence.

L'arborescence du site se décline comme suit :

- Page d'accueil
- Page connexion
- Page inscription
- Page tous les produits
- Pages produits par catégories
- Page voir produit
- Page mon profil & historique de commande(s)
- Page panier
- Page choix livraison/facturation/mode de paiement
- Page paiement
- Page de confirmation de commande
- Page de contact

- Page Qui sommes-nous?
- Page Administration

La visualisation des actions de façon arborée permet d'avoir à la fois un niveau suffisant de détails afin de décrire l'action effectuée et une vision plus globale permettant d'identifier les incohérences en amont de l'élaboration d'une maquette. L'arborescence sous forme de schéma est présentée en annexe.

(Cf. UX/UI - Annexe 1)

Description des fonctionnalités

1.Catalogue produit et filtre

Une page doit permettre d'afficher l'ensemble des produits disponibles. Cet affichage devra comprendre la photo du produit, le nom du produit ainsi que son prix.

2.Fiche produit

L'utilisateur devra être en mesure d'accéder à une fiche produit. Cette dernière devra comprendre:

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

3.Panier client

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

Ce panier devra afficher les éléments suivants:

- Une photo de l'article
- Le prix de l'article
- La quantité demandée par le client
- Un bouton de validation du panier débouchant sur le processus de commande.

Le client devra être en mesure d'augmenter ou diminuer la quantité demandée. Il aura aussi la possibilité de supprimer le un article du panier, voire l'intégralité du panier.

4. Fonctionnalité de recherche produit

Le client devra être en mesure d'effectuer une recherche de produit ou de catégorie dans un champ prévu à cet effet. Afin de faciliter la recherche, un système de recherche avec autocomplétion doit être mis en place.

5.Espace client

L'utilisateur aura accès à un espace client dans lequel il lui sera possible de consulter et modifier ses informations. A savoir son nom, son prénom, son adresse email, son mot de passe mais également son adresse postale.

Il aura également la capacité de consulter ses précédentes commandes et de voir leur statut (annulée, en cours de traitement, expédiée).

6.Back office

L'administrateur du site devra avoir accès à un espace sécurisé lui permettant d'administrer le site.

6.1 Ajout des catégories produit

L'administrateur sera en mesure de gérer les catégories de produit c'est-à-dire créer des catégories et les supprimer.

7.2 Gestion des produits

L'administrateur aura également la capacité de créer des produits et de les supprimer.

Lors de la création du produit, ce dernier sera en mesure de:

- Donner un titre au produit
- Définir le prix du produit
- Décrire le produit
- Conférer une image au produit
- Établir les stocks du produit
- Décider si ce produit doit être mis en avant ou non.

8. Inscription/connexion

Un utilisateur peut s'inscrire sur le site et aura la capacité de se connecter pour :

- Accéder à son profil
- Finaliser une commande

9.Solution de paiement

La client aura le choix de procéder à un paiement sécurisé que ce soit par carte bancaire ou Paypal

3.3.2. Charte graphique

Pour la charte graphique, j'ai défini une palette de trois couleurs, un logo et les polices utilisées pour la typographie, pour que le site e-commerce ait sa propre identité visuelle.

Le site doit être attrayant visuellement pour l'utilisateur, tout en restant sobre et épuré.

Par conséquent, les couleurs utilisées sont le noir, le blanc et la couleur dorée pour faire référence aux matériaux nobles utilisés dans la fabrication des montres. La couleur dorée est aussi associée à l'image que l'on se fait du secteur du luxe.

Palette de couleurs:



Pour ce qu'il s'agit de la typographie, j'ai décidé d'utiliser la police Poppins de GoogleFonts. Son esthétique est agréable à voir car élégante, simple. Cela correspond au thème du e-commerce et à l'image renvoyée par Iced Watches.

Concernant le menu de navigation, j'ai opté pour la police Play de GoogleFonts car elle est épurée, attirante visuellement mais surtout elle détient une forte personnalité et rentre parfaitement dans un style flat design.

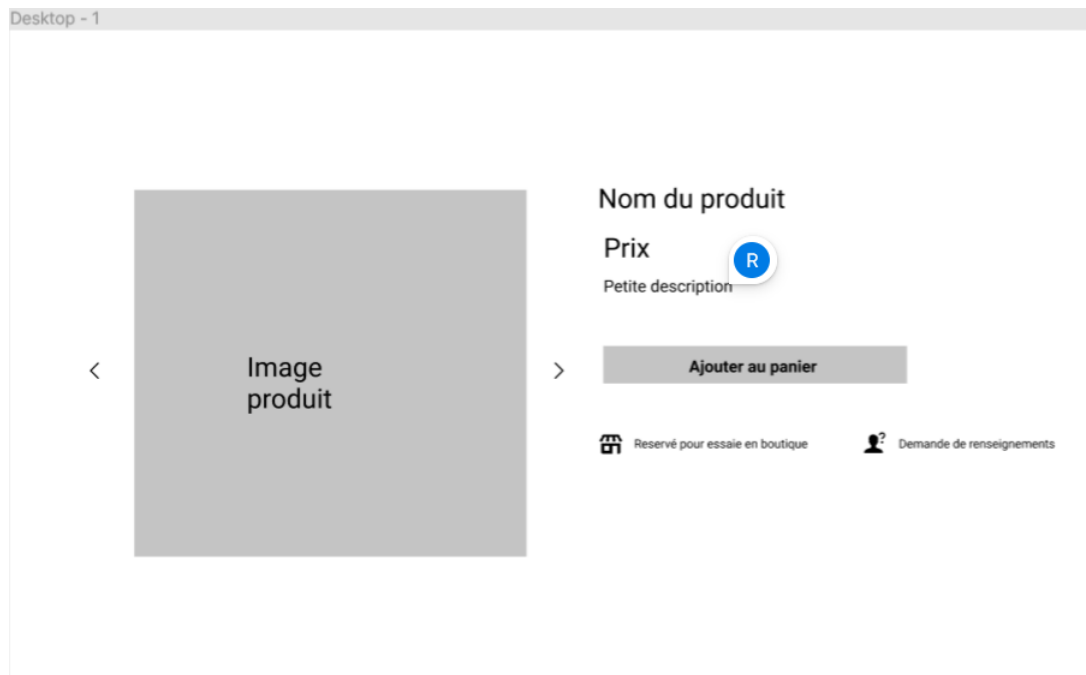
3.3.3. Maquettes

Les maquettes ont été réalisées sur le site gratuit **Figma**.

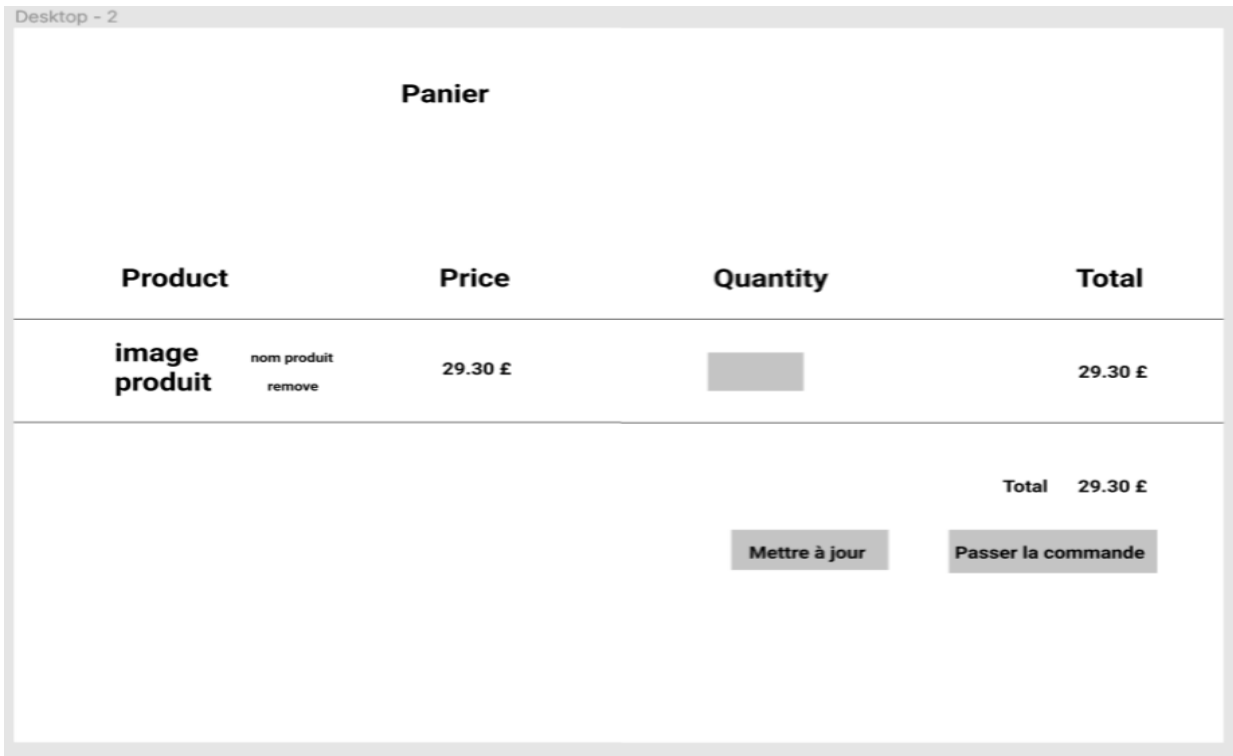
Les maquettes Wireframe permettent de visualiser comment agencer des éléments sur les différentes pages.

J'ai choisi un design minimaliste, pour rester dans le style Flat Design très en vogue dans le web.

Maquette page produit:

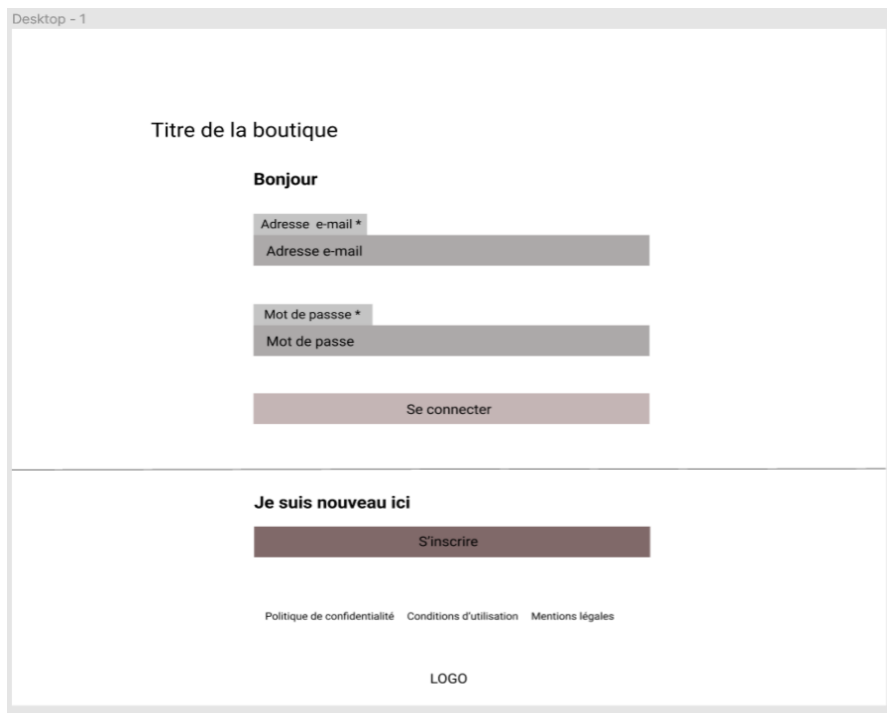


Maquette page panier:

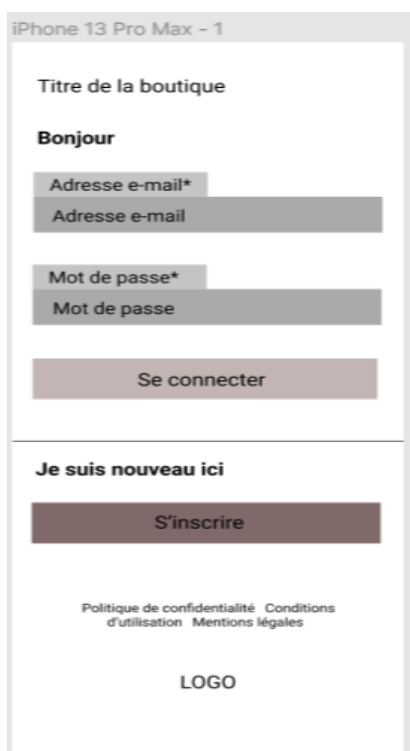


Par la suite, des maquettes Design ont été réalisées afin de mettre en avant une identité visuelle du site. Le but étant que le l'utilisateur puisse trouver le site attrayant et agréable car la première impression qu'un utilisateur fait sur un site est la partie front-end.

Maquette de la page de connexion version desktop :



Maquette de la page connexion version mobile:



4. Spécificités techniques : Backend

4.1. Environnement de développement et organisation

4.1.1. Technologies

Pour mener à bien le développement de ce projet, j'ai opté pour un environnement de développement sous la stack technique **WAMP**.

WAMP est un acronyme faisant référence à:

- **Windows**, le système d'exploitation de Microsoft le plus utilisé dans le monde.
- **Apache**, logiciel permettant de créer un serveur **HTTP**
- **MySQL**, le **SGBD**
- **PHP**, le langage de programmation informatique utilisé pour gérer les requêtes vers le serveur HTTP et générer des pages web dynamiques.

4.1.2. Workflow

En vue d'optimiser la gestion de projet et l'optimisation du temps de développement, il était nécessaire d'utiliser divers outils permettant de se répartir les tâches de développement et de versionner les modifications apportées sur le site.

Concernant la gestion des tâches, l'outil de gestion de projet en ligne utilisé se nomme Trello (**Cf. Workflow - Annexe 2**).

Il permet une organisation des projets selon un modèle de planches listant des cartes, chacune représentant des tâches. Chaque carte est assignable à

des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement.

Pour gérer les différentes modifications apportées dans le code et créer un historique de ces dernières.

Pour le versioning (gestion des modifications apportées), j'ai utilisé le logiciel de version de contrôle Github Desktop couplé à son homologue en ligne GitHub. Il s'agit d'une plateforme en ligne qui permet aux développeurs de stocker et de partager, publiquement ou non, le code généré pour différents projets.

Le workflow de ce projet e-commerce Iced Watches s'articule autour de deux branches principales, Main - anciennement Master - et Dev.

D'autres branches sont créées pour chaque nouvelle fonctionnalité qu'il faut développer. Une fois le développement de ces autres branches terminées, il faut les mergées sur la branche de référence Dev pour vérifier si le code ne comporte pas de bug ou de conflits suite à l'introduction de nouveaux morceaux de code.

4.2. Modélisation de la base de données

4.2.1. Méthode MERISE

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise).

Née à la fin des années 1970 en France, elle a pour objectif de définir une démarche permettant la modélisation et la conception de S.I.

Parmi les ressources informatiques de ces S.I., figurent en particulier les fichiers de données, bases de données et système de gestion de bases de données (S.G.B.D.). C'est sur ce dernier point que la méthode MERISE a été utile, car c'est en se basant sur ses principes de modélisation que la base de données de Iced Watches a été conçue.

4.2.2. Modèle conceptuel de données (MCD)

Il s'agit d'un modèle simplifié de la base de données, où les tables sont seulement au stade d'entités.

Entre les différentes entités, se trouvent des liaisons que l'on appelle associations, celles-ci représentent le verbe d'action qui décrit l'opération qui se fait entre les deux entités.

Un MCD est donc basé sur deux notions principales : les entités et les associations, d'où sa seconde appellation de schéma Entité/Association.

(Cf. Modèle conceptuel de données (MCD) - Annexe 3)

D'abord il a fallu imaginer tous les besoins des futurs utilisateurs et admin de Iced Watches. A partir de ces besoins, il a été possible d'établir les règles de gestion des données à conserver.

Ensuite il fallait définir le dictionnaire des données, c'est-à-dire toutes les données élémentaires qui vont être conservées en base de données et définir certaines caractéristiques qui figureront dans le MCD. Parmi ces caractéristiques, on retrouve par exemple la référence d'une donnée et notamment un identifiant unique, sa désignation, son type etc...

Étape finale à sa conception, on a pu à partir des informations précédemment recueillies, créer chaque entité, unique et décrite par un ensemble de propriétés; Et leur associations permettant de définir les liens et cardinalités entre les entités. Les cardinalités représentent le nombre d'interactions possibles entre deux entités.

4.2.3. Modèle Logique de Données (MLD)

Le MLD est une étape intermédiaire de la modélisation permettant la transition entre le modèle conceptuel de données et le modèle physique de données.

(Cf. Modèle Logique de Données (MLD) - Annexe 4)

Le passage d'un **MCD** en **MLD** s'effectue selon quelques règles de conversion bien précises :

- Une entité du **MCD** devient une relation, c'est-à-dire une table. Dans un SGBD de type relationnel, une table est une structure tabulaire dont chaque ligne correspond aux données d'un objet enregistré et où chaque colonne correspond à une propriété de cet objet. Ces colonnes font notamment référence aux caractéristiques définies dans le dictionnaire de données du **MCD**.
- Les identifiants respectifs de chaque entité deviennent des clefs primaires et toutes les autres propriétés définies dans le **MCD** deviennent des attributs. Les clefs primaires permettent d'identifier de façon unique chaque enregistrement dans une table et ne peuvent pas avoir de valeur nulle.
- Les cardinalités de type "0:n" | "1:n" sont représentés à travers le référencement de la clef primaire de la table qui la possède, en clef étrangère au sein de la table auquel elle est liée. Si deux tables liées possèdent une cardinalité de type "0:n|1:n", la relation sera traduite par la création d'une table de jonction/liaison, dont la clef primaire de chaque table deviendra une clef étrangère au sein de la table de jonction/liaison.
Les clés primaires de chaque table associée deviennent des clés étrangères dans chacune des tables qui reçoivent les données.

4.2.4. Modèle Physique de Données (MPD)

Étant presque une formalité, le MPD consiste à l'implémentation des modèles générés précédemment dans le Système de Gestion de Base de Données (SGBD) utilisé. Dans notre cas, ce sera le SGBD MySQL couplé au moteur de stockage InnoDB qui permet la gestion des contraintes des clefs étrangères en base de données.

4.3. Architecture logicielle: Backend

4.4. Extraits de code significatifs

Pour développer la partie back-end, j'ai choisi d'utiliser la technologie **PHP** et plus précisément la programmation orientée objet(**P.O.O**).

Les requêtes en base de données qui concernent l'utilisateur sont gérées par la classe "User". Cette classe possède des méthodes permettant d'exécuter des requêtes préparées.

L'utilisation de requêtes préparées, rendues possibles grâce à l'extension **PDO**, présentent un double avantage par rapport à l'exécution directe de requêtes **SQL**. Premièrement, peu importe le nombre d'exécution des requêtes, nous n'avons besoin de les préparer qu'une seule fois pour qu'elles soient réutilisables à volonté avec des paramètres identiques ou différents.

En un second temps, elles présentent un avantage majeur en termes de sécurité notamment pour prévenir des **injections SQL**.

Nos requêtes étant pré-formatées, elles empêchent le passage de code malicieux en paramètre(s), nous n'avons donc pas besoin de protéger nos paramètres ou nos valeurs manuellement.

La première étape est de créer une classe puis d'y ajouter les attributs que l'on désire.

```
class User extends Config
{
    private $_id;
    private $_nom;
    private $_prenom;
    private $_mail;
    private $_adresse;
    private $_codepostal;
    private $_ville;
    private $_password;
    private $_Malert;
    private $_Talert;
    private $_droits;
```

Un attribut est une variable qui est partagée par toutes les instances de la classe, on peut y stocker par exemples les paramètres de la connexion à la base de données pour ne pas avoir à les recopier à chaque fois que l'on a besoin de faire un appel à la base de données.

Pour que je puisse faire appel à mes fonctions sur cette page, il faut que je j'inclus ma page de fonction dans la page ou je souhaite utiliser ces fonctions et ensuite instancier la classe que je veux instancier, cela se fait de cette manière :

```
<?php
$header = new View;
$header->headerStyle();
?>
```

Connexion base de données et héritage

Concernant la connexion à la base de données j'ai procédé de la manière suivante.

J'ai codé une class **Bdd** qui se connecte à la base de données dans `_construct` directement, j'ai utilisé un **"try and catch"** la fonction php qui gère les erreurs. La gestion d'une erreur via une exception se fait en deux temps.

On va utiliser un bloc try dans lequel le code qui peut potentiellement retourner une **erreur** va être exécuté. On crée à l'intérieur une nouvelle connexion grâce à l'objet **new PDO**.

On va créer un bloc catch dont le but va être d'attraper l'exception si celle-ci a été lancée et de définir la façon dont doit être gérée l'erreur. La fonction `headerStyle` est appelée dans les autres classes afin de réaliser différentes requêtes.

la BDD quatre paramètres, le nom du serveur, la en l'occurrence localhost vu que je travaille en local, le nom de ma base de données qui est "boutique", l'identifiant et le mot de passe. ensuite je le stocke dans un attribut pour pouvoir faire appel dans toutes mes autres fonctions.

```

class Config
{
    protected $bdd;
    public function __construct()
    {
        try {
            $this->bdd = new PDO(
                'mysql:host=localhost;dbname=boutique',
                'root',
                ''
            );
        } catch (Exception $e) {
            die('Erreur : ' . $e->getMessage());
        }

        return $this->bdd;
    }
}

```

Jeu d'essai

Après avoir fait cela, on peut s'attaquer à l'écriture de toutes nos fonctions.

Les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données est nommée le **CRUD** (pour **create**, **read**, **update**, **delete**).

Je vais vous présenter l'ensemble des requêtes couvertes par le **CRUD** :

En ce qui concerne le **create**, cette opération est gérée par la requête "**INSERT**", pour envoyer des éléments en base de données.

Dans le cadre de ma boutique en ligne, je m'en suis servi pour envoyer en base de données les articles que je souhaite présenter.

Je commence déjà par récupérer les données en méthode 'POST' dans mon formulaire ci dessous :

Nom

Description

Prix

Quantité

Catégorie

Choisir une catégorie ▾

Image

Ajouter

Ici le code HTML de mon formulaire :

```
<form method="POST">
  <div>
    <label class="label" for="nom">Nom</label>
    <input class="inputtext" type="text" id="nom" name="nom" placeholder="Nom produit" required>
  </div>
  <div>
    <label class="label" for="description">Description</label>
    <input class="inputtext" type="text" id="description" name="description" placeholder="Ce produit est.." required>
  </div>
  <div>
    <label class="label" for="prix">Prix</label>
    <input class="inputtext" type="number" id="prix" name="prix" placeholder="En euro" required>
  </div>
  <div>
    <label class="label" for="quantite">Quantité</label>
    <input class="inputtext" type="number" id="quantite" name="quantite" placeholder="0" required>
  </div>
  <div>
    <label class="label" for="categorie">Catégorie</label>
    <select id="categorie" name="categorie" required>
      <option>Choisir une catégorie</option>
      <?=$cat->getCategories(); ?>
    </select>
  </div>
  <div>
    <label class="label" for="img">Image</label>
    <input class="inputtext" type="text" id="img" name="img" placeholder="Lien de l'image" required>
  </div>
  <div class="form-admin-butt">
    <button type="submit" name="submit">Ajouter</button>
  </div>
</form>
```

Donc pour envoyer des données en BDD, la méthode est la suivante, je commence par rédiger ma fonction que j'ai appelé ici "CreerProduits",

```

public function CreerProduits($nom, $prix, $img, $description, $quantite, $date, $categorie)
{
    $req = $this->bdd->prepare("INSERT INTO produits (blaze, prix, img, description, quantite, date, id_categorie)
VALUES (:blaze, :prix, :img, :description, :quantite, :date, :id_categorie)");
    $req->execute(array(
        ':blaze' => $nom,
        ':prix' => $prix,
        ':img' => $img,
        ':description' => $description,
        ':quantite' => $quantite,
        ':date' => $date,
        ':id_categorie' => $categorie,
    ));
    // var_dump($req);
    // var_dump($req->execute());
}

```

Je commence par déclarer ma méthode en public pour pouvoir y accéder en dehors de ma classe là où je vais instancier la méthode, après cela je récupère la connexion à ma base de données que j'ai stocké dans un attribut, dans un second temps je vérifie la cohésion des données envoyées par le formulaire (pour vérifier si les inputs ont bien été remplis et que aucun n'est vide). Une fois cela fait, je peux lancer ma requête SQL qui est un **INSERT**, dans cette INSERT je lui spécifie de cibler la table "produits" et les colonnes dans lesquelles je veux les stocker et pour finir les valeurs que je veux insérer.

Par la suite, je fais une requête préparée qui va exécuter ce que je lui passe en paramètre, donc le fait d'insérer en BDD les différentes données que je lui est ordonné d'insérer.

En ce qui concerne le **read**, cette opération est gérée par la requête **SELECT**, pour récupérer des éléments en base de données.

Dans le cadre de mon e-commerce, je m'en suis servi pour afficher les articles présents en base de données.

En ce qui concerne "l'**update**", cette opération est gérée par la requête qui porte le même nom **UPDATE**, pour modifier des éléments en base de données.

Dans le cadre de ma boutique en ligne, je m'en suis servi pour donner la possibilité à chaque utilisateur de mettre à jour leurs informations personnelles en base de données.

En ce qui concerne le "**DELETE**", cette opération est gérée par la requête qui porte le même nom **DELETE**, pour supprimer des éléments en base de données.

Dans le cadre de ma boutique en ligne, je m'en suis servi pour donner la possibilité à chaque administrateur de supprimer un ou des produits en base de données pour qu'ils ne soient plus présents sur le e-commerce.

5. Spécificités techniques : Frontend

5.1. Intégration

L'intégration de la maquette à été faite en utilisant le langage de balisage HTML. HTML étant limité à la création de simples représentations structurées des pages, c'est avec le langage CSS qu'a été défini le style désiré en accord avec la charte graphique du e-commerce Iced Watches évoqué précédemment dans le dossier.

Je le fais par le biais de feuilles de style personnalisées. L'espace admin détient sa propre feuille de style personnalisée.

Ce choix à l'avantage d'offrir une maintenabilité du code plus aisée car les modifications apportées impactent seulement la page liée à la feuille de style concernée, donc cela évite les conflits.

Pour ajouter du CSS à une balise HTML il suffit de lui donner une "class" que l'on nommera comme on le souhaite. (de préférence pour un nom explicite pour pouvoir mieux s'y retrouver après).

Après cela, il sera désormais possible de cibler cette balise via le nom qu'elle porte directement dans mon fichier CSS.

Il faut au préalable dans le <head> du fichier HTML créer un lien avec le fichier CSS avec une balise <link> à l'intérieur de la balise <head>.

```
<link rel="stylesheet" href="style2.css" />
```

Après avoir fait cela, on peut donner une classe à une balise, ici il s'agit d'une balise <h2> et grâce à cela il est désormais possible de lui attribuer du style.

Fichier HTML:

```
<h2 class="titre-vendues">Nos montres les plus vendues</h2>
```

Fichier CSS:

```
.titre-vendues{
  font-family:'Play', sans-serif;
  font-family: 'Share Tech Mono', monospace;
  color:wheat;
  font-weight: 600;
}
```

5.2. Responsive design

Une grande importance est portée sur l'adaptation de l'interface pour que le site web puisse être utilisé sur différents supports.

Les principaux supports sont : les ordinateurs, les tablettes et les smartphones.

Pour ce faire, j'utilise des **@Media Queries** dans le CSS.

Je commence par appeler les classes que je veux cibler puis je définis les largeurs d'écran maximales ou minimales.

Je peux facilement adapter le contenu affiché en changeant sa taille, masquer ou afficher du contenu ciblé.

Exemples de **@Media Queries**:

```
@media screen and (max-width:844px){
  .inner-container{
    padding: 80px;
  }
}

@media screen and (max-width:800px){
  .about-section{
    background-size: 100%;
    padding: 100px 40px;
  }
  .inner-container{
    width: 75%;
  }
}

@media screen and (max-width:600px){
  .about-section{
    padding: 0;
  }
  .inner-container{
    padding: 60px;
  }
}
```

Je définis que la largeur maximale que je souhaite est de 844px donc ces paramètres seront affectés aux classes ciblées qu'une fois que la largeur d'écran passe en dessous de 844px.

Exemple écran moins de 844px:

Iced Watches

≡

Bonjour

Adresse email

✉

Adresse email

Mot de passe

🔑

Mot de passe

Se connecter

Iced Watches

5.3. Javascript et AJAX

Dans le but d'agrémenter l'expérience utilisateur et d'apporter du dynamisme au site, l'utilisation du langage Javascript a été nécessaire. Utilisé côté client - dans sa version Vanilla - il permet de manipuler le Document Object Model (DOM), une représentation structurée sous forme d'arborescence de nœuds (node tree), de tout le contenu d'un document HTML.

Grâce à cette interface, il est possible d'exécuter du code en réponse à un événement déclenché par l'utilisateur, agir sur les différents éléments d'une page

et les modifier, ou récupérer des données d'un serveur distant en effectuant des requêtes asynchrone sans que cela n'interfère avec l'affichage et le comportement de la page existante.

Des requêtes asynchrone ont été réalisées en utilisant l'API Fetch. Elle permet d'effectuer des requêtes HTTP mais facilite la manière de récupérer et traiter la réponse. Fonctionnant avec le système de Promesses, Fetch renvoie un objet Response qui englobe entres autres, les données renvoyées par le serveur. Ces dernières sont alors directement exploitables.

Dans le cadre du e-commerce Iced Watches, Javascript a été utile pour permettre d'afficher/cacher (show/hide) l'historique de commande(s) dans la page profil. De plus, Javascript m'a permis de remplir automatiquement les informations de livraison lorsque les informations de facturation et de livraison sont les mêmes.

7. Veille effectuée sur les vulnérabilités de sécurité

7.1. Résumé des recherches

Une veille concernant les failles de sécurité web a été faite durant la réalisation du e-commerce. J'ai parcouru le web à travers différents sites internet (OWASP, MDN Webdoc, OpenClassRoom, StackOverflow) pour recouper les informations sur les failles web les plus connues.

- Failles de sécurité web (faille sécurité web, faille sécurité web openclassroom)
 - La sécurité d'un site Web - Apprendre le développement web | MDN
 - Sécurisez vos applications web avec l'OWASP
- Injections SQL (injection sql, web injection flaws)
 - What are injection flaws?
 - 6 réflexions au sujet de « Fonction PHP contre les injections SQL
- PDO, requêtes préparées (pdo requêtes préparées)
 - Les requêtes MySQL préparées avec PDO PHP
- Faille XSS (faille xss openclassroom, xss flaw owasp)
 - Stoppez le cross-site scripting (XSS) - Sécurisez vos applications web avec l'OWASP
 - XSS Filter Evasion Cheat Sheet
- Faille upload (upload vulnerabilities owasp)
 - Unrestricted File Upload | OWASP

7.2.3 Les failles de sécurité

7.3. Les injections SQL

7.3.1. Description

Cette vulnérabilité permet à un attaquant d'injecter des données non maîtrisées qui seront exécutées par l'application et qui permettent d'effectuer des actions qui ne sont normalement pas autorisées. Ce type d'attaque s'effectue généralement grâce aux champs présents dans les formulaires.

Dans le cas d'une attaque par injection SQL, au lieu de mettre un nom d'utilisateur et un mot de passe sur une page de connexion, un utilisateur malveillant entrera des données directement interprétées par le moteur SQL, ce qui lui permettra de modifier le comportement de votre application.

7.3.2. Solutions apportées

- Validation des formulaires et données en entrée de l'application
 - Validation frontend avec javascript
 - Validation HTML avec l'attribut type de l'input et l'attribut required
 - Validation Backend des données avant inscription en base de données
- Utilisation de requête préparées
 - Implémentation de la méthode de classe Request::send permettant l'utilisation de requêtes préparées. Ce type de requête, exécuté en utilisant l'objet PDO, permet d'interpréter les paramètres indépendamment de la requête. De cette manière, le risque de subir des injections SQL est nul.

7.4. Cross-site scripting (XSS)

7.4.1. Description

Avec une attaque XSS, un attaquant va essayer de prendre le contrôle de votre navigateur en injectant un script JavaScript dans l'application web. Il pourra l'injecter directement dans un formulaire, mais il peut également l'injecter dans l'URL, l'en-tête HTTP ou d'autres parties du framework utilisé. Contrairement aux injections SQL, il ne s'agit pas de requêtes et de commandes SQL sur une base de données. Une faille XSS s'exécute dans le code de l'application web.

7.4.2. Solutions apportées

- Validation des formulaires et données en entrée de l'application
 - Validation frontend avec javascript
 - Validation HTML avec l'attribut type de l'input et l'attribut required
 - Validation Backend des données avant inscription en base de données
- Encodage de données récupérées en entité de caractères HTML au moyen de la fonction PHP htmlspecialchars(). Cette dernière désactivant lors de l'affichage

des données tous les caractères spéciaux et notamment les caractères de balise
< > </ >.

7.5. La faille Upload

7.5.1. Description

Il existe deux types de problèmes provenant de l'upload d'un fichier sur le serveur. Le premier concerne les métadonnées du fichier, comme le chemin ou le nom du fichier. Ceux-ci sont généralement fournis par le transport, comme le codage HTTP en plusieurs parties. Ces données peuvent inciter l'application à écraser un fichier critique ou à stocker le fichier dans un mauvais emplacement. Les métadonnées de fichier doivent être validées par le serveur avant traitement. L'autre type de problème concerne la taille ou le contenu du fichier. L'éventail des problèmes dépend entièrement de l'utilisation du fichier.

7.5.2. Solutions apportées

- Validation des formulaires et données en entrée de l'application
 - Validation frontend avec javascript
 - Validation HTML avec l'attribut type de l'input et l'attribut required
 - Validation Backend des données avant inscription en base de données
- Restriction du type de fichier
- Restriction de la taille du fichier

8. Extrait anglophone

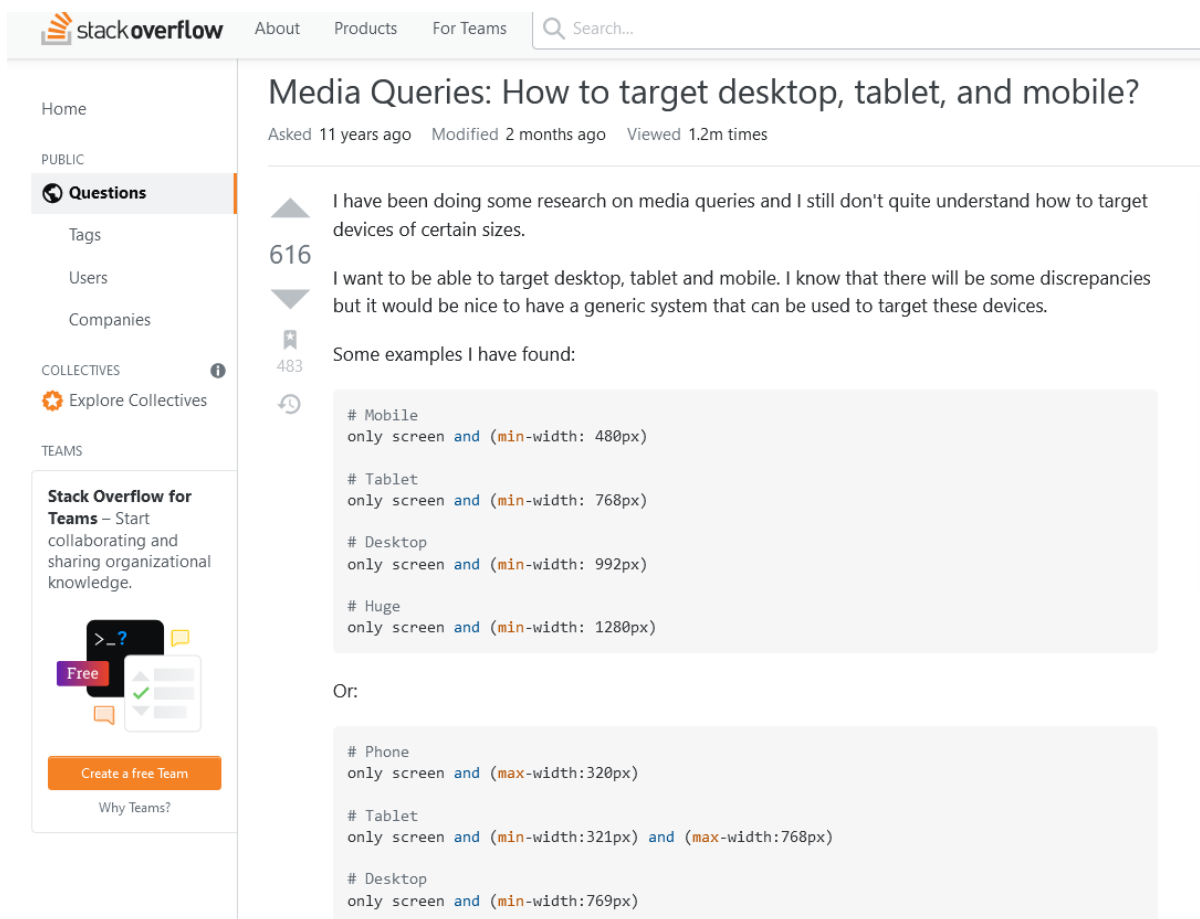
Situation de travail ayant nécessité une recherche à partir de site anglophone

Durant le développement de ce projet, j'ai rencontré de nombreuses situations qui ont nécessité une recherche d'informations. En grande majorité, les solutions recherchées trouvaient réponses sur des sites anglophones.

Pour décrire une de ces situations, je me suis tourné vers le forum anglo-saxon StackOverflow.com.

En effet, je recherchais des informations sur la manière de rendre responsive mon site e-commerce Iced Watches. Pour ce faire, j'ai tapé sur Google "how to make a website responsive".

Je me suis rendu sur le forum et j'ai réussi à obtenir les informations dont j'avais besoin afin de mettre en place des Media Queries pour optimiser le site sur différents supports.



The screenshot shows a Stack Overflow page for a question titled "Media Queries: How to target desktop, tablet, and mobile?". The question was asked 11 years ago, modified 2 months ago, and viewed 1.2m times. It has 616 votes and 483 answers. The user asks for help in targeting desktop, tablet, and mobile devices with media queries. The accepted answer provides the following CSS code:

```
# Mobile
only screen and (min-width: 480px)

# Tablet
only screen and (min-width: 768px)

# Desktop
only screen and (min-width: 992px)

# Huge
only screen and (min-width: 1280px)
```

Alternatively, the answer provides another set of media queries:

```
# Phone
only screen and (max-width:320px)

# Tablet
only screen and (min-width:321px) and (max-width:768px)

# Desktop
only screen and (min-width:769px)
```

Traduction:

Media Queries : comment cibler les ordinateurs, les tablettes et les mobiles ?

J'ai fait des recherches sur les Media Queries et je ne comprends toujours pas très bien comment cibler des appareils de certaines tailles.

Je veux pouvoir cibler les ordinateurs de bureau, les tablettes et les mobiles. Je sais qu'il y aura des divergences, mais ce serait bien d'avoir un système générique qui puisse être utilisé pour cibler ces appareils.

Quelques exemples que j'ai trouvé :

Mobile

only screen and (min-width: 480px)

Tablette

only screen and (min-width: 768px)

Ordinateur

only screen and (min-width: 992px)

Grand Écran

only screen and (min-width: 1280px)

Quels devraient être les points d'arrêt pour chaque appareil ?

Réponse:

@media (min-width:320px) smartphones, portrait iPhone, portrait 480x320 téléphones (Android)

@media (min-width:480px) smartphones, téléphones Android, iPhone

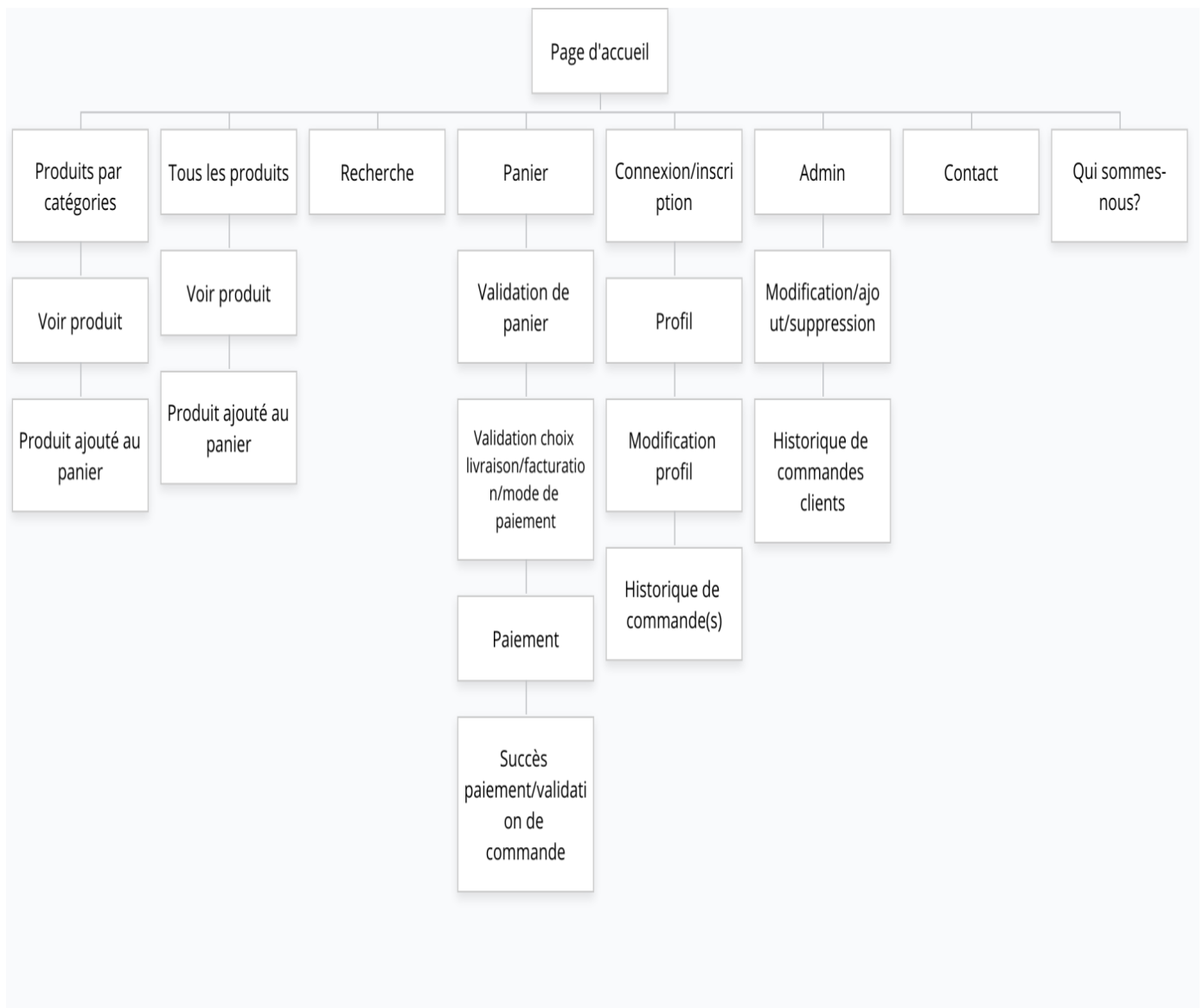
@media (min-width:600px) portrait tablettes, portrait iPad, tablette de lecture (Nook/Kindle)
800x480 phones (Android)

@media (min-width:801px) tablette, iPad, ordinateur portable et ordinateur de bureau

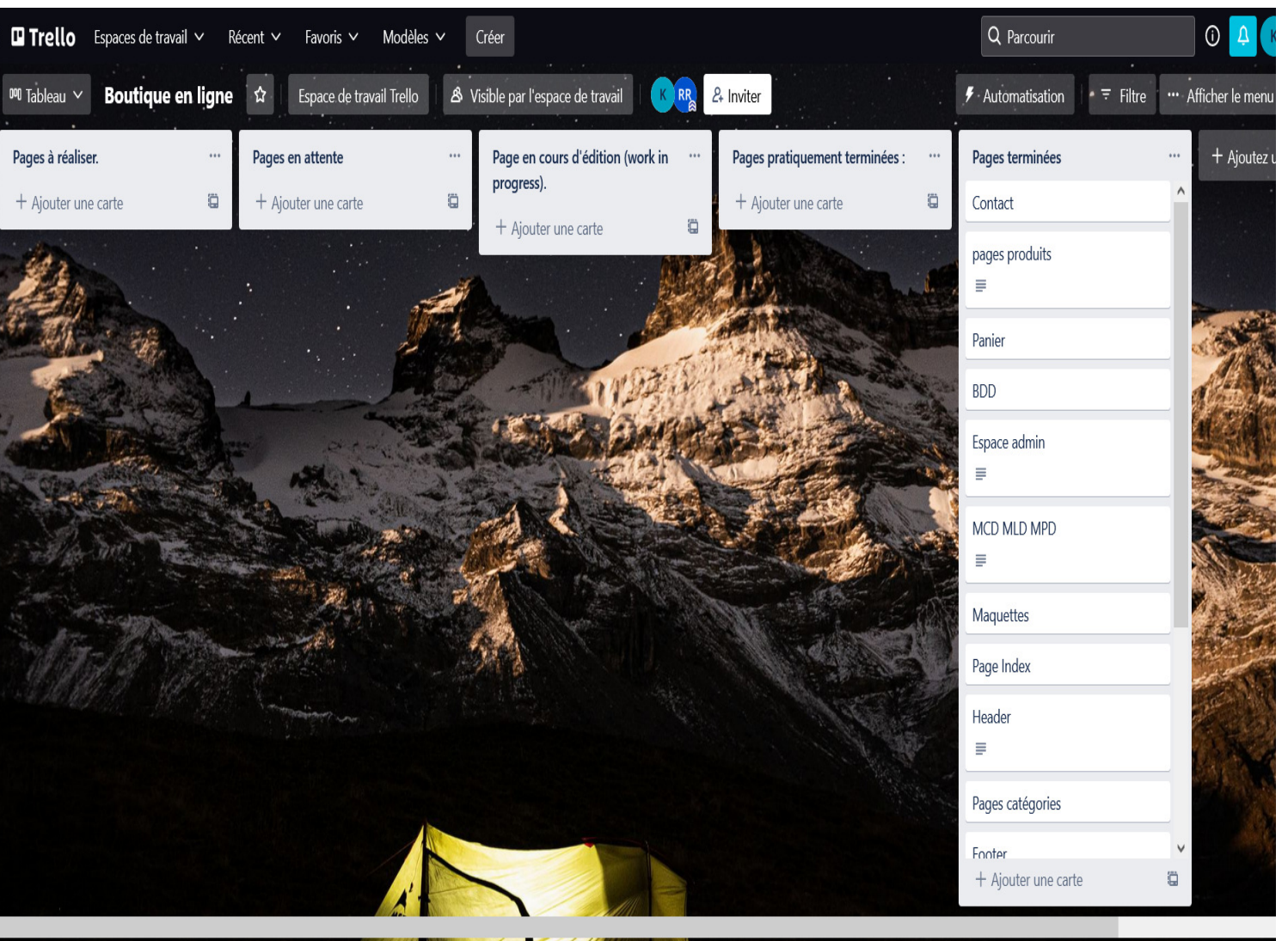
@media (min-width:1025px) grandes tablettes, ordinateur portable et ordinateur de bureau

@media (min-width:1281px) haute résolution ordinateur portable et ordinateur de bureau

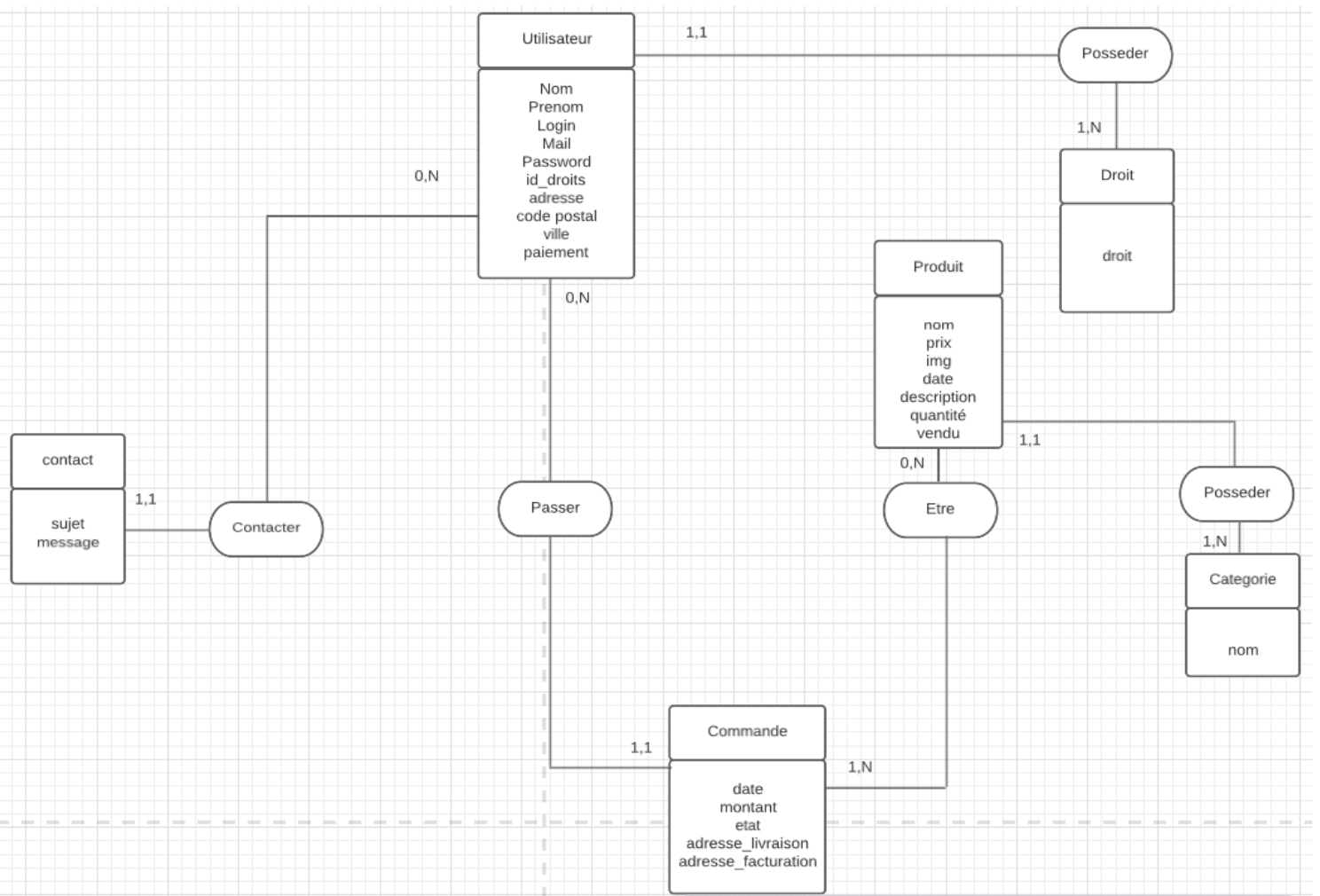
ANNEXE 1



ANNEXE 2



ANNEXE



ANNEXE 4

