Dossier de projet professionnel

Création d’une boutique en ligne

Présentée par Morgane Maréchal

Sommaire

Introduction :

Le projet de boutique en ligne est un projet réalisé en équipe dans le cadre de la formation développeur web et mobile de la formation LaPlateforme_ C'est un titre RNCP de niveau V.

Liste des compétences du référentiel couvertes :

Ce projet couvre les compétences nécessaires à l'obtention du titre répartie en deux activités:

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.

Spécificités du projet

Contexte de création

Le projet de boutique en ligne est un projet de groupe dans la cadre de la formation développeur web et mobile de l'école La Plateforme. Nous étions dans un groupe de quatres personnes avec un tech lead plus expérimenté. Nous nous sommes organisés afin de travailler en groupe.

Périmètre et cible

Le site est une boutique en ligne d'instruments de musique. La cible regroupe des utilisateurs sur un panel d'âge et de situation sociale assez large. Cela comprend donc l'utilisation, de la part des clients, de supports différents (téléphones mobiles, ordinateurs, tablettes) qu'il fallait prendre en compte pour le responsiv de la solution. Les utilisateurs peuvent aussi bien surfer sur le site et commander des instruments de musique sur leur ordinateur ou sur leur smartphone sans perdre de l'expérience qualité. Ce qui implique une version mobile aussi orientée expérience utilisateur que la version laptop.

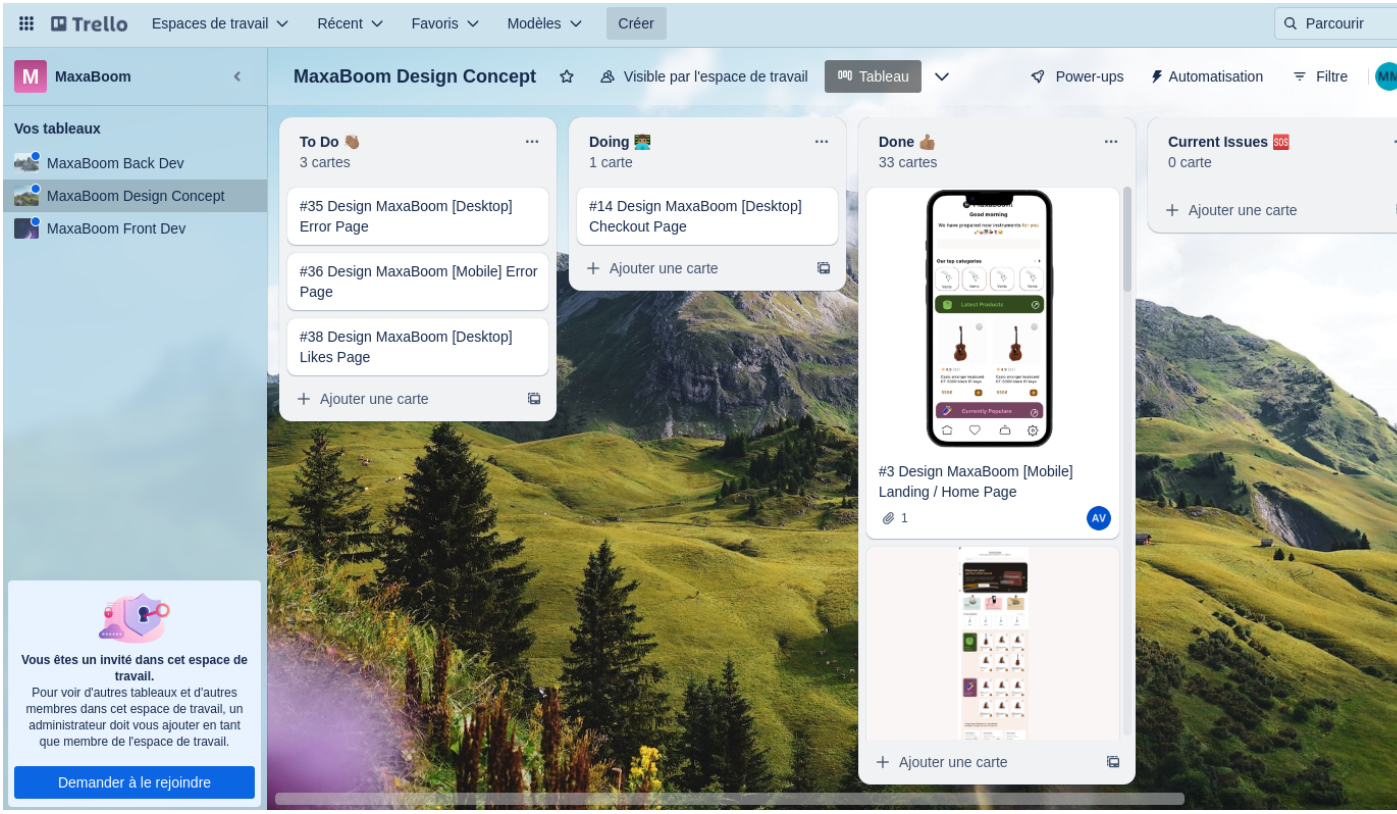
Pour rendre cela possible, nous avons utilisé le concept "mobile first". Mobile First est un concept de Web Design optimisé pour le mobile qui va au-delà du Responsive Web Design. Il consiste à concevoir un site en mettant la priorité sur la version mobile et en adaptant progressivement le web design pour les écrans plus large.

C'est le contraire de l'approche qui consistait à dégrader progressivement un site web pour l'adapter à un affichage sur Smartphones. Les pages d'un site web sont tout d'abord créées pour les smartphones et les tablettes puis évoluent ensuite progressivement pour s'adapter aux ordinateurs.

Méthode de travail

Trello pour l’organisation des tâches

Trello est un outil de gestion de projet en ligne. Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement. Les cartes sont rédigées en anglais.



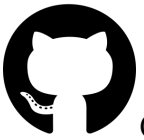
Les tâches sont réparties en trois colonnes, celles qui sont à faire, celles qui sont en cours et celles qui sont terminées. Cette méthode permet de suivre plus facilement l’état d’avancement du projet en cours. Chacun peut rédiger de nouvelles étiquettes ou s’en attribuer selon les besoins du projet.

Figma pour le travail commun sur le design

Figma est un éditeur de graphiques vectoriels et un outil de prototypage. C’est un outils collaboratif qui permet à plusieurs personnes de travailler sur un même projet

Chaque utilisateur peut travailler sur une page ou un composant du projet en suivant la charte graphique.

Git et Github pour le versionning


Github est une plateforme de versionning qui permet de centraliser un projet et d'y participer à plusieurs. Le répertoire a été créé sous le nom de Maxaboom (le nom de la boutique en ligne). Qui a ensuite été cloné en local par les membres du groupe.


Git est un logiciel de gestion de version décentralisé. C'est le plus populaire parmi les développeurs. Il permet de communiquer entre la version local de l'application et la version remote qui est centralisée sur github.

Pour chaque feature du projet nous avons créé des branches afin que chacun puisse travailler sur un élément différent sans empiéter sur le travail des autres. Je réalise régulièrement des commits pour ne pas perdre mon travail en cours. Quand une feature est terminée, j'envoie une pull request sur la branche en remote qui est ensuite validée par un membre de l'équipe. Le code est ainsi mutuellement vérifié et cela permet d'éviter les conflits de versioning lors du merge entre les versions..

Descriptif fonctionnel

Le projet de boutique en ligne comprend une page d'accueil attractive avec plusieurs sections dont une mise en avant des produits phares sur la page d'accueil.
Le design est contemporain et respecte la charte graphique de l'entreprise. Le site est responsive.
Le site doit comprendre une barre de recherche avec autocomplétion.
La boutique présente tous les produits avec la possibilité de les filtrer par catégorie / sous-catégories sans rechargement de page.
Un clic sur chaque produit renvoie à une page "détail" complète générée dynamiquement avec le nom, l'image, le prix, la description et un bouton ajouter au panier.
L'utilisateur peut se créer un compte avec un module Inscription / Connexion en asynchrone avec javascript.

L’administrateur possède son propre tableau de bord ce qui permet une gestion des produits à l’aide de back office (Ajout / Suppression /Modifications de produits, stocks...), une gestion des catégories et des sous catégories de produits (Ajout / Suppression / Modifications...)

La boutique comprend aussi un système de validation du panier avec ou non une vraie solution de paiement.

Arborescence du projet

- Page d’accueil (home)
- Page de connexion
- Page d’inscription
- Page recherche pour trouver un produit spécifique
- Page pour chaque produit
- Page mon compte (account)
- Page addresses
- Page panier (cart)
- Page paiement

L’ensemble de l’arborescence du projet a été maquetté sur Figma avec la version mobile et la version desktop.

Spécification technique et conception :

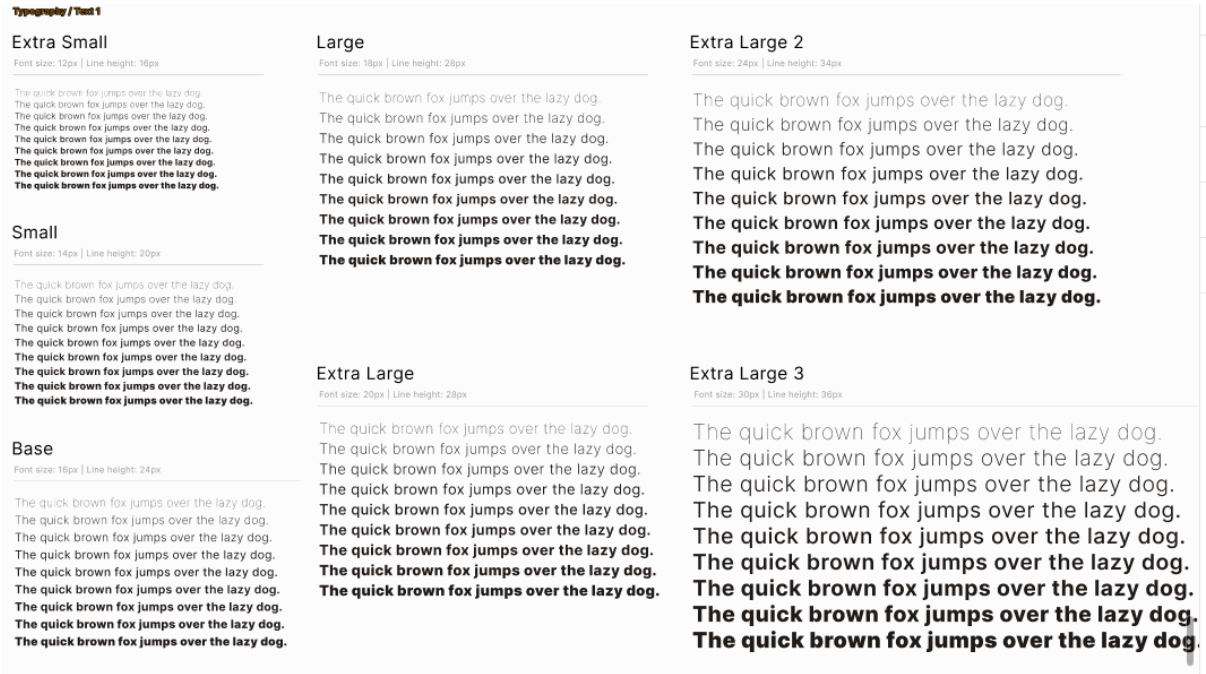
Charte graphique

Une charte graphique web est un ensemble de règles et de normes graphiques qui constituent l'identité visuelle d'un site web. La charte graphique web contient des règles d'utilisation de tous les éléments graphiques, et représente donc un élément essentiel du site.
La charte graphique a été explicitée sur le Figma afin que chaque membre de l'équipe puisse s'y référer pour créer des éléments graphiques de la maquette.

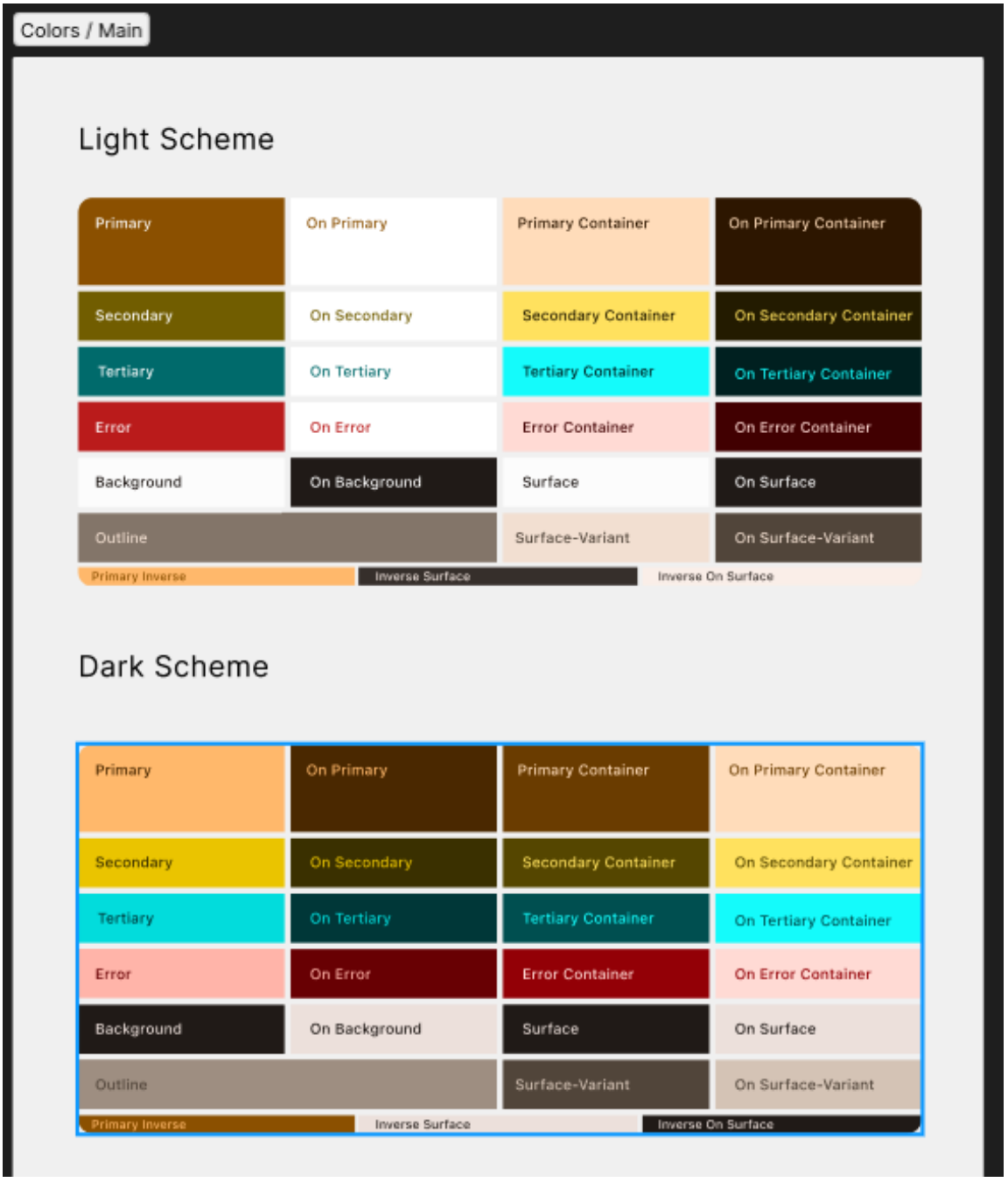
Elle comprend plusieurs éléments :

Le logo : la charte doit contenir le logo de l'entreprise ainsi que ses variantes, ses références couleurs et ses différentes tailles.

La typographie : la charte doit contenir la liste de toutes les polices utilisées, et ce pour chaque niveau de titre.
Les polices utilisées pour le site sont Mulish et Roboto.

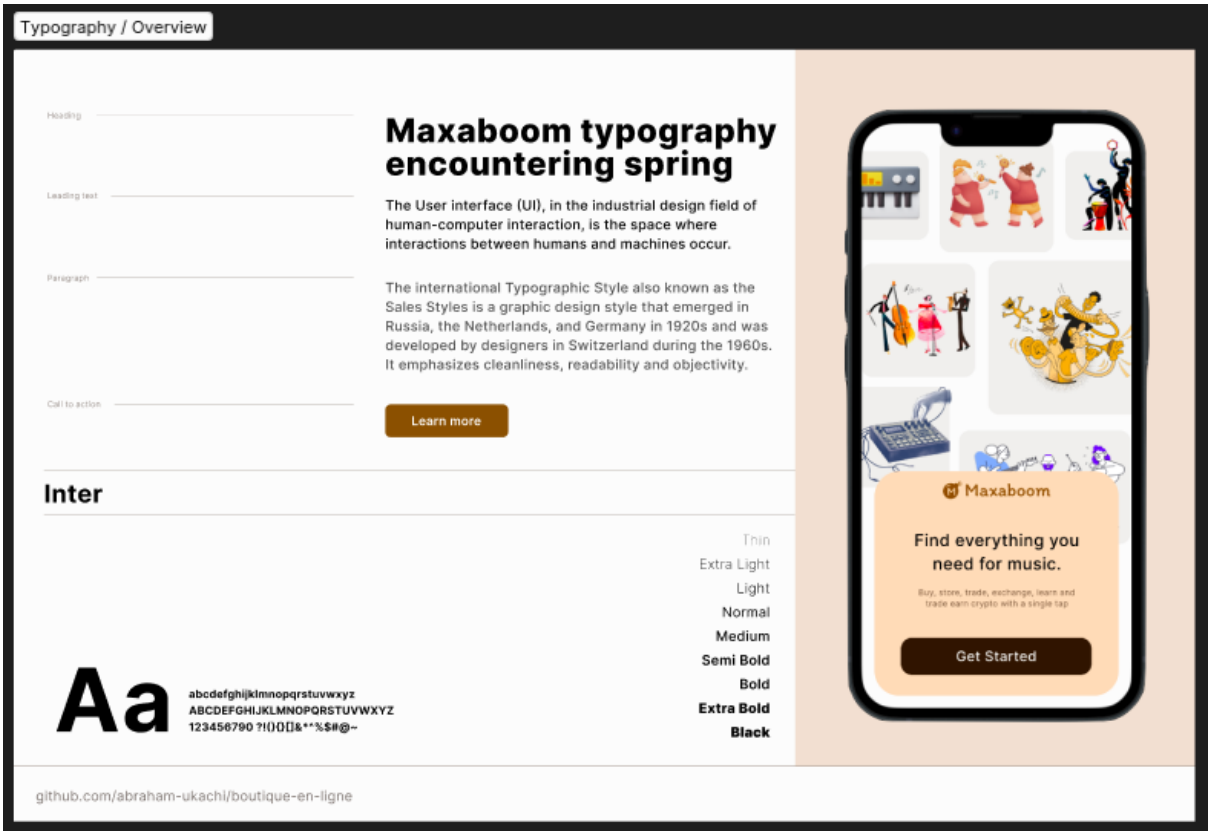


Les couleurs : la charte doit contenir les références des couleurs utilisées ;



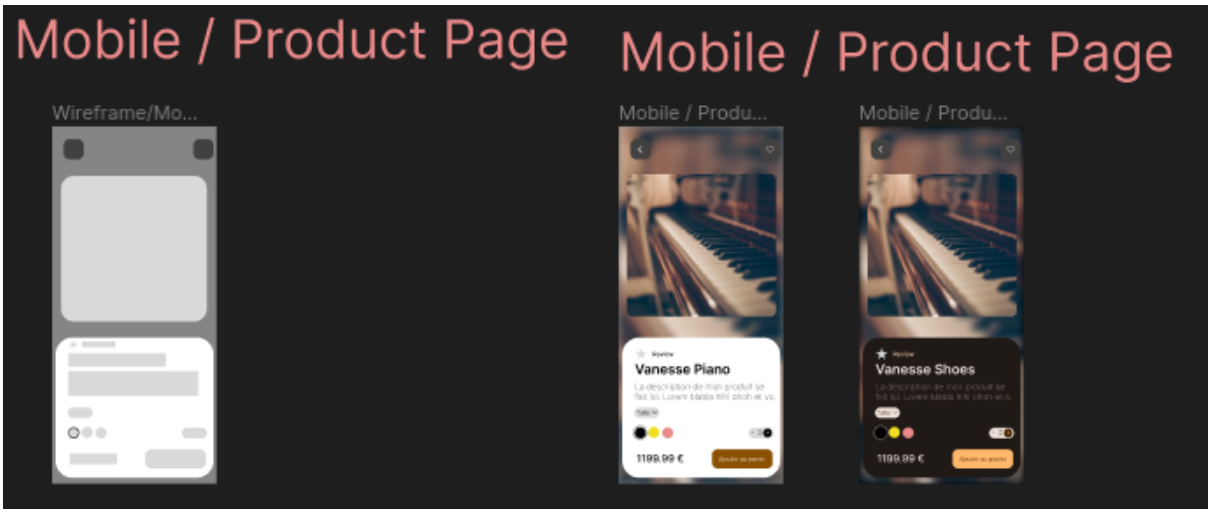
Les boutons et les icônes :
Les icônes et les boutons ont au préalable été créés sur Figma afin d'avoir une idée précise du rendu final.
Ci-dessous voici une partie des boutons utilisés.


Icon Button
Taille

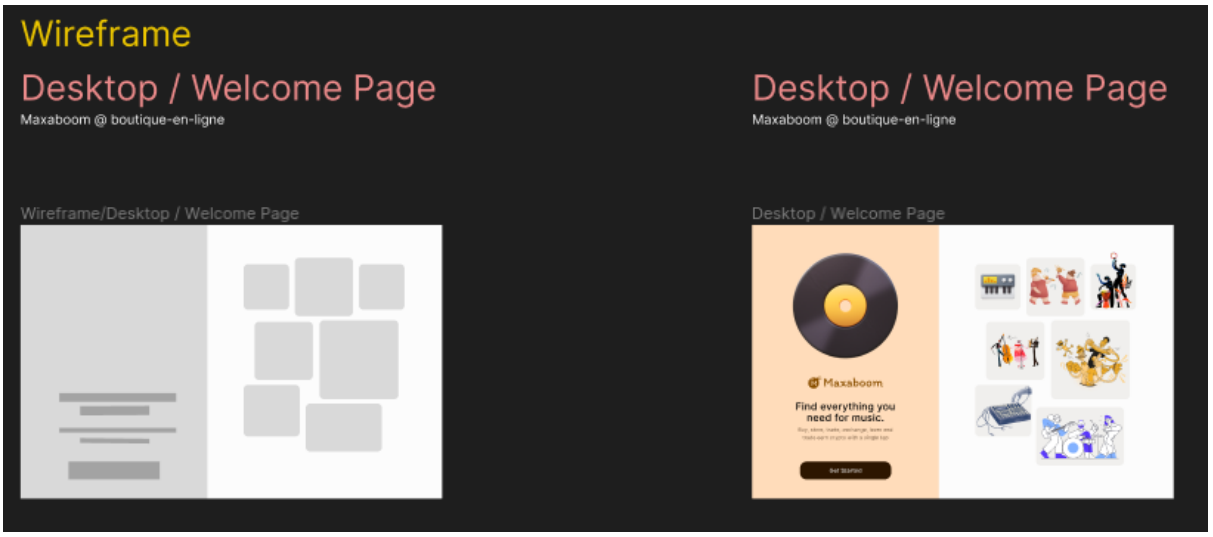

Typography / Overview
Heading
Leading text
Paragraph
Call to action
Maxaboom typography encountering spring
The User interface (UI), in the industrial design field of human-computer interaction, is the space where interactions between humans and machines occur.
The international Typographic Style also known as the Sales Styles is a graphic design style that emerged in Russia, the Netherlands, and Germany in 1920s and was developed by designers in Switzerland during the 1960s. It emphasizes cleanliness, readability and objectivity.
Learn more
Inter
Aa
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
123456790 ?!(){}[]&**%$#@~
Thin
Extra Light
Light
Normal
Medium
Semi Bold
Bold
Extra Bold
Black
Maxaboom
Find everything you need for music.
Get Started
github.com/abraham-ukachi/boutique-en-ligne

Maquette

Pour chaque page, nous avons fait sur le figma une maquette basse fidélité et une maquette haute fidélité.



Nous avons fait cela aussi bien pour la version mobile (ci-dessus) que desktop (ci-dessous).



Nous avons également créé des composants réutilisables et détaillés à assembler sur les différentes maquettes (voir ci-dessous).

Card/Product

❖ Card/Product

GUITARE

199,99 €

Corde

+
1
-

Card/Product/Desktop

❖ Card/Product/Desktop

Saxophone

Détails

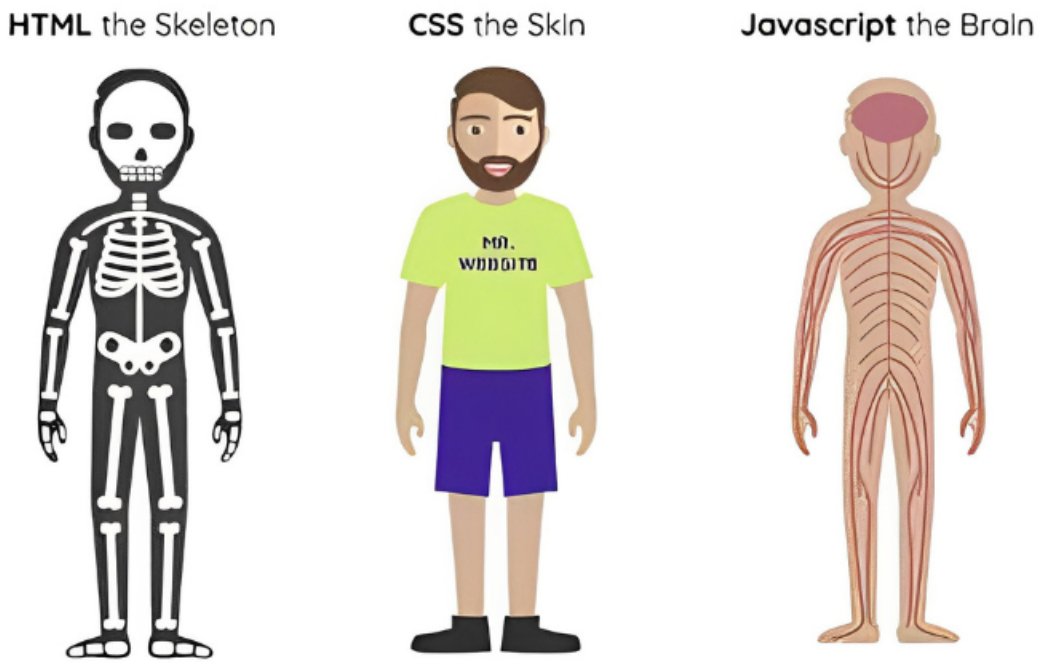
‹ 1 ›

399,99 €

X

Les langages utilisés

Pour ce projet, nous avons utilisés HTML, CSS, Javascript, PHP et SQL.
Html : Le langage HTML (HyperText Markup Language) est un langage de balisage utilisé pour la création de pages web, permettant notamment de définir des liens hypertextes. Il sert à structurer une page et son contenu.
CSS : c'est l'acronyme de « Cascading Style Sheets » ce qui signifie « feuille de style en cascade ». Sur le plan de la conception d'une page Web, le CSS permet par ailleurs de séparer la présentation d'une page HTML et sa structure. Ses standards sont définis par le World Wide Web Consortium (W3C).
Javascript : JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web. Cela permet notamment d'afficher dynamiquement du contenu selon les actions de l'utilisateur.



PHP : Le PHP, pour Hypertext Preprocessor, désigne un langage informatique, ou un langage de script, utilisé principalement pour la conception de sites web dynamiques. Il s'agit d'un langage de programmation sous licence libre qui peut donc être utilisé par n'importe qui de façon totalement gratuite. Avec le système d'exploitation Linux, il fait partie intégrante de la suite de logiciels libres LAMP. que j'ai utilisé.

SQL : SQL ou « Structured Query Language » est un langage de programmation permettant de manipuler les données et les systèmes de bases de données relationnelles. Ce langage permet principalement de communiquer avec les bases de données qui est dans PHPMyAdmin.

PhpMyAdmin : phpMyAdmin est une application Web de gestion pour les systèmes de gestion de base de données MySQL et MariaDB, réalisée principalement en PHP et distribuée sous licence GNU GPL (licence publique générale).

LAMP : LAMP est un acronyme du système d'exploitation Linux, du serveur web Apache, du serveur de base de données MySQL et du langage de programmation PHP.

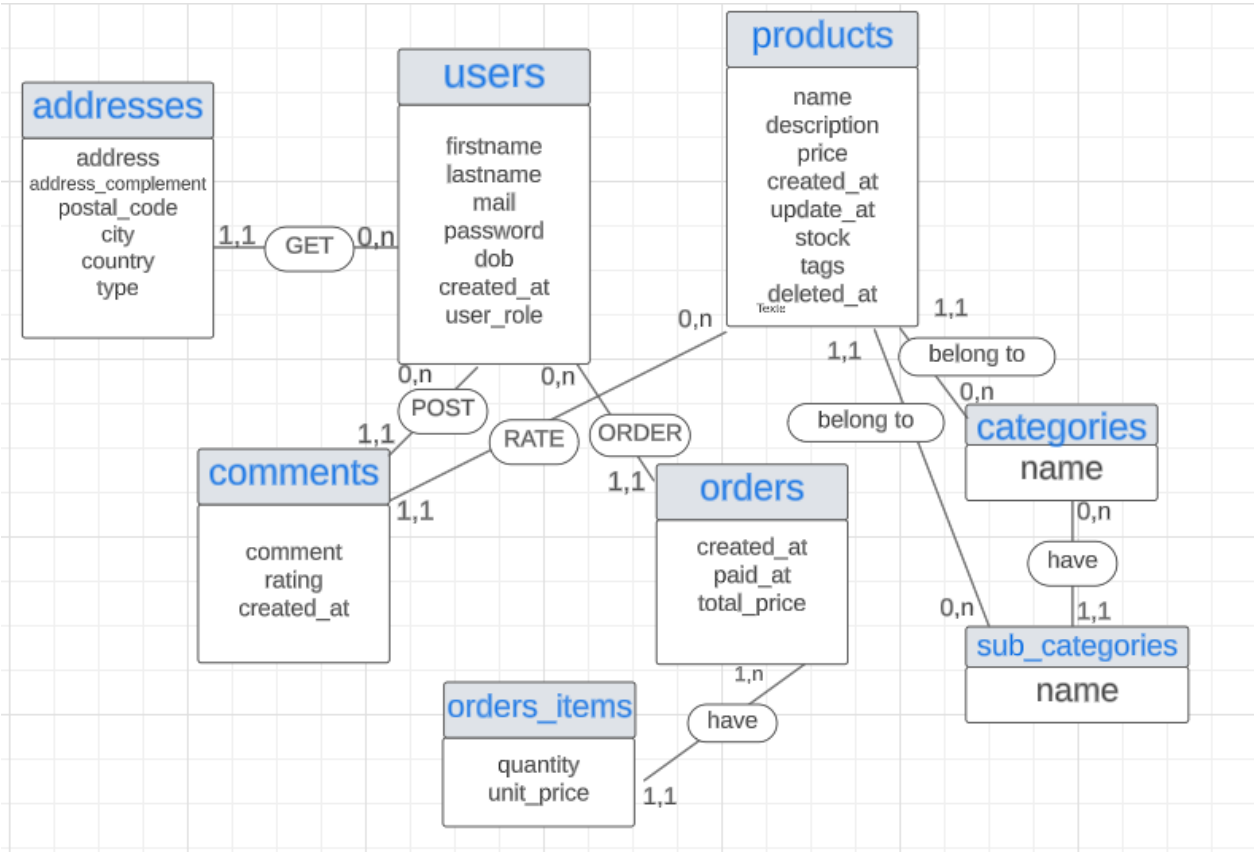
La base de données

Base de données : MCD, MLD, MPD

Le MCD (modèle conceptuel de données) fournit une description graphique pour représenter des modèles de données sous la forme de diagrammes pouvant contenir des entités ou des associations. Il peut être utilisé pour décrire les besoins en information ou par exemple le genre d'information nécessaire à l'élaboration du cahier des charges.

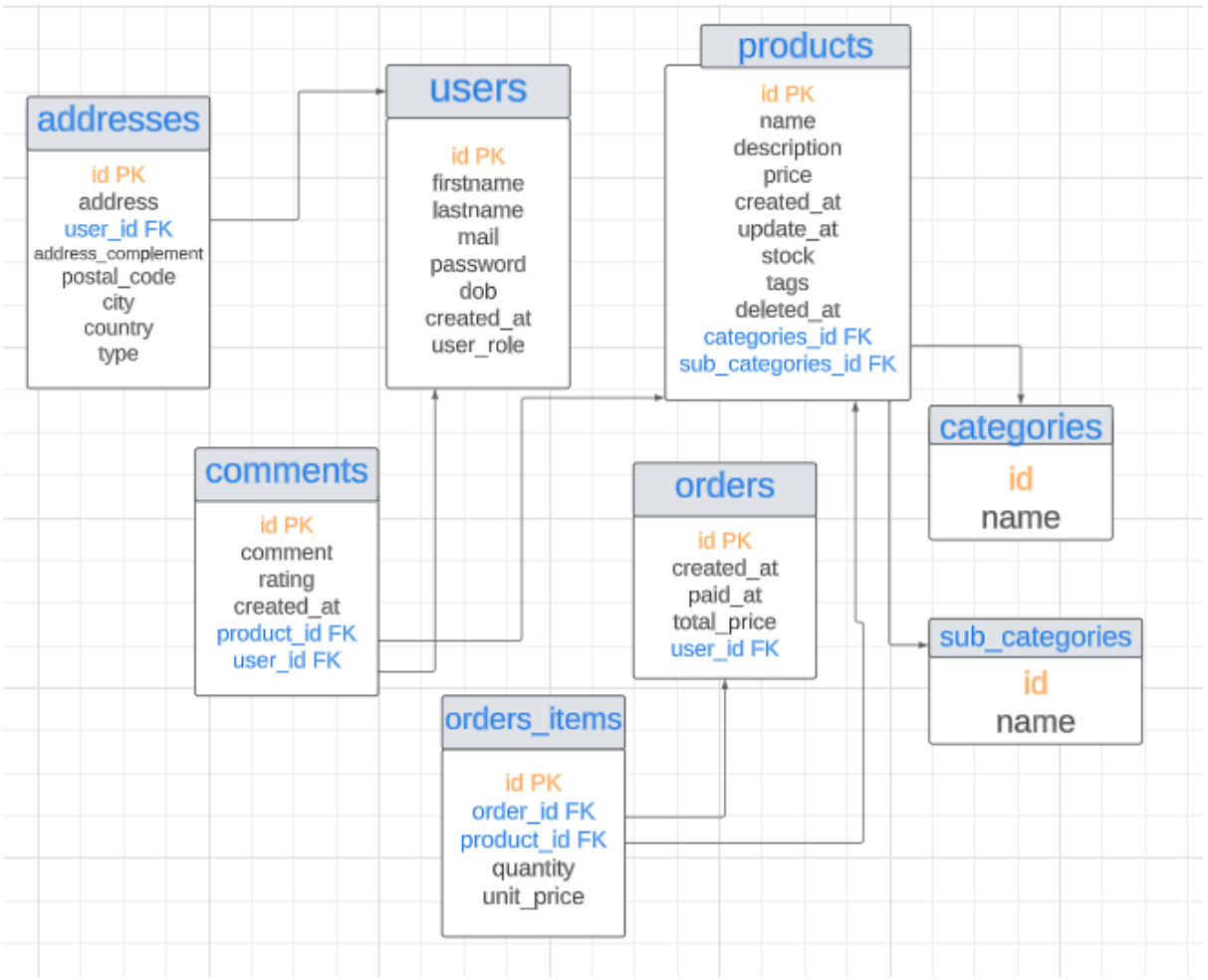
Le MCD comprend les tables des entités dont on aura besoin, les relations entre les tables, les cardinalités et les attributs.

Le modèle conceptuel de données (MCD) ci-dessous :

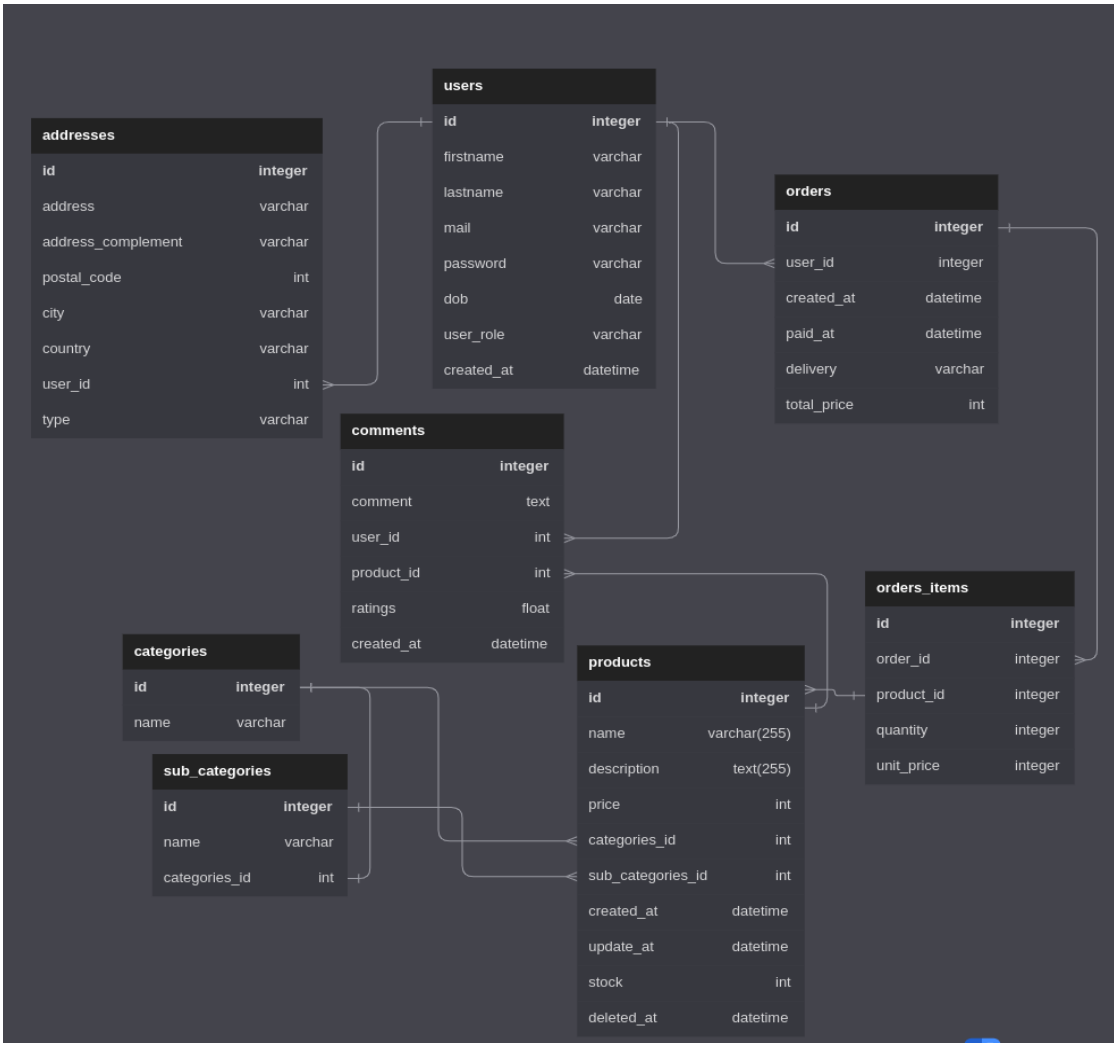


Le modèle logique de données (MLD) ci-dessous :

Pour le modèle logique de données, on ajoute les clés primaires et les clés étrangères. Si la relation entre deux table sont ‘many to many’, il faut rajouter une table intermédiaire.



Le modèle physique de données (MPD)



Réalisation

Architecture MVC

Le code a été organisé selon le design pattern MVC. Le MVC est un design pattern qui signifie Modèle - Vue - Contrôleur. Ainsi chaque fichier créé a un rôle bien défini. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

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

Vue : cette partie se concentre sur l'affichage. Elle récupère la variable qui contient tout ce qu'on doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste d'utilisateurs comme ci-dessous.

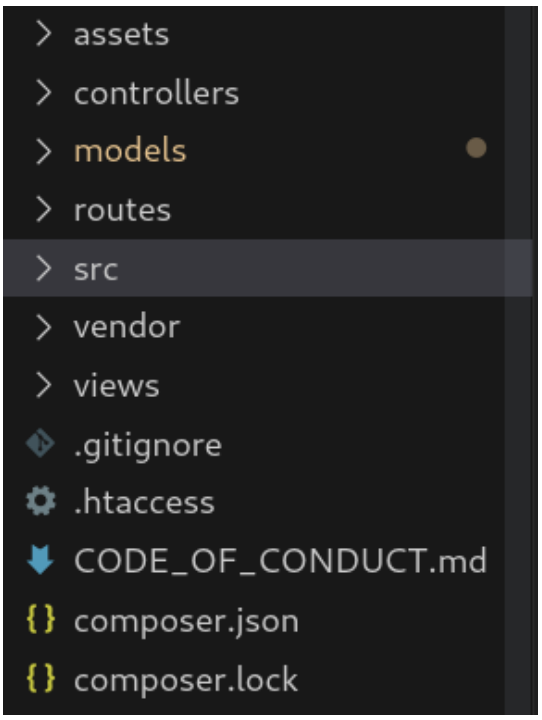
```
<h1>Gestion des utilisateurs</h1>
<?php
  echo "<br>";
  for ($i = 0; $i < count($allUsers); $i++) {
      echo "<div id=".$allUsers[$i]['id']." data-user-
        id=".$allUsers[$i]['id']." class='update-
        user'>".$allUsers[$i]['firstname'].
        " ".$allUsers[$i]['lastname']." ".$allUsers[$i]
        ['mail']." ".$allUsers[$i]['dob']."
        ".$allUsers[$i]['user_role'].
       "<a href='admin/user/".$allUsers[$i]
        ['id']."'>Modifier</a> <button id='".$allUsers[$i]
        ['id']."' class='deleteUser'>Supprimer</button>
        </div><br>";
  }
?>
```

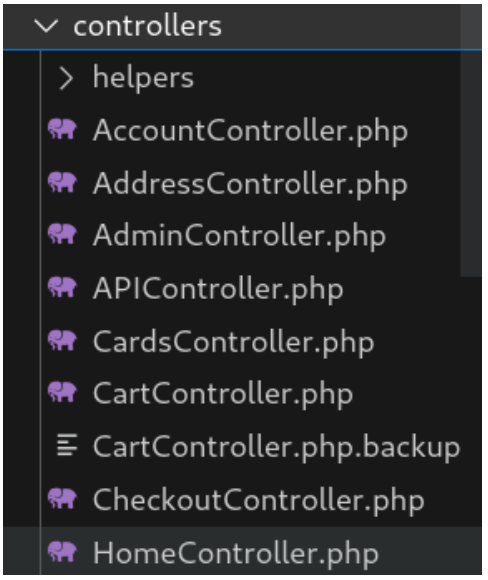
Contrôleur : cette partie gère les échanges avec l'utilisateur. C'est en quelque sorte l'intermédiaire entre l'utilisateur, le modèle et la vue. Le contrôleur va recevoir des requêtes de l'utilisateur. Pour chacune, il va demander au modèle d'effectuer certaines actions (ajouter un nom en base de donnée, supprimer un article du panier, …) et de lui renvoyer les résultats (un message pour dire que l'inscription est réussie, ajouter l'article dans le panier). Puis il va adapter ce résultat et le donner à la vue. Enfin, il va renvoyer la nouvelle page HTML, générée par la vue, à l'utilisateur.
Exemple, ci-dessous le controleurs renvoie une variable avec tous les utilisateurs ainsi que la page où ils pourront être affiché :

```
public function showAllUsers(){
    $users = new User();
    $allUsers = $users->displayUsers();
    require __DIR__ . '/../views/admin-users-page.php';
}
```

Exemple d'un dossier avec toutes les pages controllers (idéalement un controller par page affichée):

Dans Maxaboom, les fichiers relatifs au MVC étaient réparties dans les dossiers models, controllers et views comme ci-dessous..


assets
controllers
models
routes
src
vendor
views
.gitignore
.htaccess
CODE_OF_CONDUCT.md
composer.json
composer.lock


controllers
helpers
AccountController.php
AddressController.php
AdminController.php
APIController.php
CardsController.php
CartController.php
CartController.php.backup
CheckoutController.php
HomeController.php

Composer

Composer est un logiciel gestionnaire de dépendances libres écrit en PHP pour les bibliothèques. Composer nous aide à gérer les dépendances du projet. Grâce à ce logiciel, on peut intégrer et gérer des paquets open source en un seul endroit.

On install composer grâce aux commandes suivantes :

```
php composer.phar install
php composer.phar update
```

L'installation de composer nous a permis ensuite d'installer un autoloader et un router pour le projet.

L'Autoloader

L'autoloader sert à auto charger des classes. En effet, comme on a séparé nos classes dans des fichiers différents, on est obligé de faire beaucoup de require afin de charger les différentes classes.

```php
<?php
  namespace Maxaboom\Controllers;
  use Maxaboom\Models\Product;
  use Maxaboom\Models\Category;
  use Maxaboom\Models\User;

  class AdminController{
    ......
   }
?>
```

Voici un exemple d'autoloading à l'entrée d'une classe (ici AdminController), on utilise plusieurs autres classes(Product, Category, User) pour le bon fonctionnement des méthodes de celle-ci.
Le namespace définit un nom de package que l'on pourra ensuite charger de manière automatique grâce à un autoloading.

Tous les autoload sont définis dans le fichier composer.json comme l'exemple ci-dessous :

```
"autoload": {
    "psr-4": {
        "Maxaboom\\Routes\\": "routes/",
        "Maxaboom\\Models\\": "models/",
        "Maxaboom\\Models\\Helpers\\": "models/helpers/",
        "Maxaboom\\Views\\": "views/",
        "Maxaboom\\Views\\Components\\": "views/components/",
        "Maxaboom\\Controllers\\": "controllers/",
        "Maxaboom\\Controllers\\Helpers\\":
        "controllers/helpers/"
    }
},
```

A chaque fois qu'on rentre un nouveau autoload, il faut entrer la commande suivante `php composer.phar dump-autoload`

Le Router

Le routeur est un composant du code qui a pour rôle de recevoir toutes les requêtes de l'application et de router chacune vers le bon contrôleur. Pour ce projet, nous avons utilisé alto router.

Un routeur permet de centraliser l'application sur la page index.php. Cela permet de canaliser l'application par une entrée et d'y inclure toutes les fonctions. Le routeur va se charger d'appeler le bon contrôleur. Sur index.php, nous avons écrit :

```
$router = new AltoRouter();
$router->setBasePath(MAXABOOM_HOME_DIR);
```

Les routes sont rangées par fichier, chaque fichier faisant référence à une page (Views). Ces fichiers sont appelés sur index.php comme suit :

```
include __DIR__ . '/routes/home-route.php';
include __DIR__ . '/routes/shop-route.php';
include __DIR__ . '/routes/admin-route.php';
```

Voici quelques exemples de requêtes possibles avec un routeur.

```php
use Maxaboom\Controllers\AdminController;

$router->map('GET', '/admin/products', function () {
    $adminController = new AdminController();
    $adminController->showProductsPage();
});

$router->map('GET', '/admin/product/[i:productId]',
function($productId) {
    $adminController = new AdminController();
    $adminController->showOneProductPage($productId);
});

$router->map('DELETE', '/admin/product/[i:productId]',
function($productId) {
    $adminController = new AdminController();
    $response = $adminController->delete($productId);
    echo $response;
});
```

Le routeur permet également de renommer (mécanisme de rewriting) les URLs et par extension une meilleure gestion des URLs. Comme les URLs sont redirigées vers le routeur, cela permet un meilleur contrôle des pages créées. La centralisation de l'information permet un meilleur travail en équipe.

Ajouter du dynamisme avec Javascript

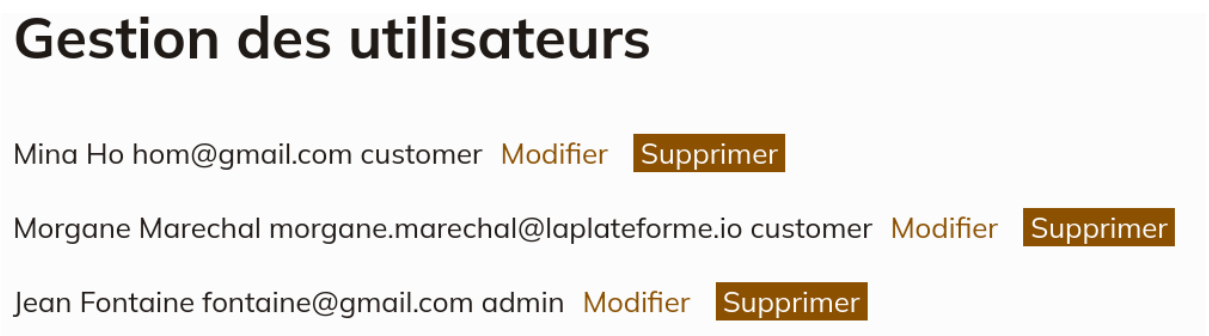
Nous avons utilisé javascript pour une meilleure fluidité du site et ainsi améliorer l'expérience utilisateur.
Pour cela nous avons utilisé l'API fetch(). Nous utilisons la méthode globale fetch() qui permet l'échange de données avec le serveur de manière asynchrone.
Pour récupérer le corps de la réponse, nous allons pouvoir utiliser les méthodes de l'interface Response en fonction du format qui nous intéresse, il y a entre autres text() et json() qui sont les deux méthodes utilisées dans le projet.

Dans l'exemple ci-dessous, nous avons utilisé json(), car nous voulions récupérer un fichier json afin de pouvoir traiter les données à l'intérieur.

Ici, si nous récupérons la valeur "ok", un élément de la page va être supprimé de manière dynamique.

```
async function deleteUser(userId){
    let response = await fetch(`admin/user/${userId}`, {method:
    'DELETE'});
    let responseData = await response.json();
    if(responseData.response == 'ok'){
        let userElement = document.getElementById(userId);
        console.log(userElement);
        userElement.remove();
    }
}
```

Concrètement, si nous appuyons sur l'un boutons "supprimer", la div html avec les informations de l'utilisateur ciblé et ce qui lui est associé sera supprimée.



API REST
Exemples action avec MVC, Router et Autoloader

Sécurisation du site Maxaboom

La prévention des attaques XSS

Cross-site scripting (XSS) est une faille de sécurité qui permet à un attaquant d'injecter dans un site web un code client malveillant. Pour prévenir les attaques XSS, il faut partir du principe que les données reçues par une application web ne peuvent pas être considérées comme « toujours » sûres.

Il est donc important d'implémenter des mesures de protection pour traiter toutes les données venant de l'extérieur. Ainsi, tout contenu doit être filtré, validé et encodé avant d'être utilisé par l'application. On utilise en particulier l'encodage d'entités HTML. En effet, certains caractères ont des significations spéciales en HTML. Htmlspecialchars() remplace tous ces caractères par leur équivalent dans la chaîne string. Cela est pratique pour éviter que des données fournies par les utilisateurs contiennent des balises HTML. Cette fonction prend un deuxième argument optionnel, qui indique comment doivent être traités les guillemets doubles et simples.

Dans le site Maxaboom, htmlspecialchar est ajouté dans la partie model, avant que les données fournies ne soient traitées pour effectuer des requêtes SQL.

Voici un exemple d'utilisation :

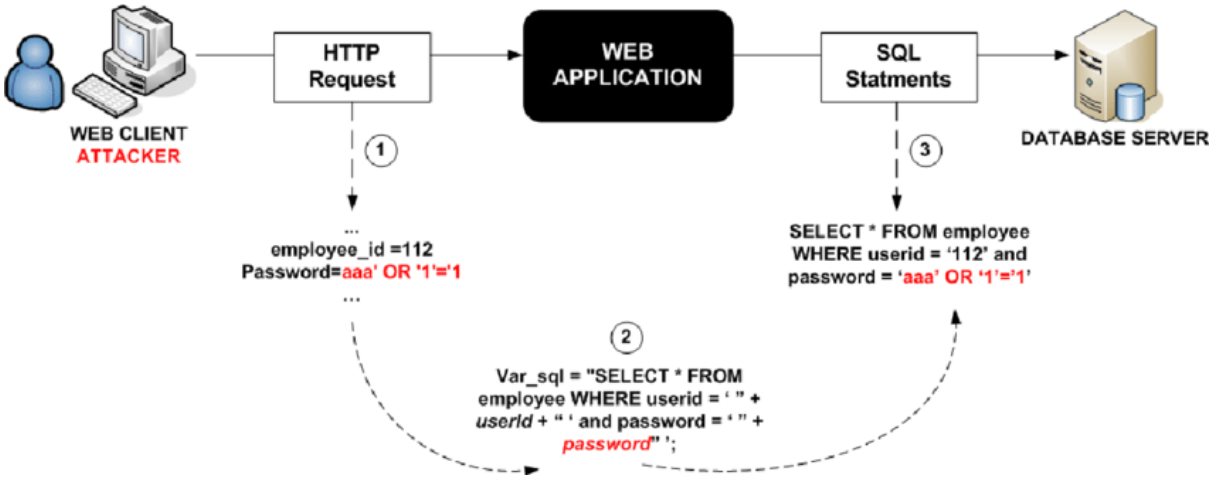
```php
public function registerGuest($firstname, $lastname, $mail)
{
    if (!$this->verifGuest()) {
        $created_at = $this->getCurrentDate();
        $sql = "INSERT INTO users (firstname, lastname, mail,
        user_role, created_at) VALUES (:firstname, :lastname,
        :mail, :user_role, :created_at)";
        $sql_exe = $this->db->prepare($sql);
        $sql_exe->execute([
            'firstname' => htmlspecialchars($firstname),
            'lastname' => htmlspecialchars($lastname),
            'mail' => htmlspecialchars($mail),
            'user_role' => 'guest',
            'created_at' => $created_at
        ]);
}
```

Ici, les entrées ‘firstname’, ‘lastname’ et ‘name’, du formulaire register sont protégées par htmlspecialchars() pour éviter les injections malveillantes de code.

Prévention des injections SQL

L'essentiel d'une injection SQL (et d'autres types de vulnérabilités d'injection) est l'utilisation de données non vérifiées provenant de l'extérieur de l'application. Il existe plusieurs mesures d'atténuation que les développeurs devraient implémenter.

Voici comment fonctionne une injection SQL :



Pour Maxaboom, nous avons utilisé des instructions préparées avec des requêtes paramétrées. En PHP, les requêtes préparées avec PDO correspondent à une façon de créer et d'exécuter nos requêtes selon trois étapes : une étape de préparation, une étape de compilation et finalement une dernière étape d'exécution.

Dans le code, cela se présente comme l'exemple ci-dessous :

```
1    public function registerSubCategory($name, $title, $categoryId){
2         $sql = "INSERT INTO sub_categories (name, title,
          category_id) VALUES (:name, :title, :category_id)";
3         $sql_exe = $this->db->prepare($sql);
4         $sql_exe->execute([
5              'name' => htmlspecialchars($name),
6              'title' => htmlspecialchars($title),
7              'category_id' => $categoryId,
8         ]);
9    }
```

Nous avons ici une méthode registerSubCategory qui va enregistrer une nouvelle sous-catégorie dans la base de données.

Dans la première phase de préparation nous allons créer un schéma de requête, où on ne précise pas les valeurs réelles dans notre requête mais des marqueurs nommés avec deux petits points (l.2 avec :name, :title et :category_id). Une cela créé, la base de données va analyser, compiler, et stocker le résultat sans l'exécuter. Finalement, nous allons lier des valeurs à nos marqueurs et la base de données va exécuter la requête.
Ici, on commence par préparer notre requête SQL grâce à la méthode prepare() (l.3) qui appartient à la classe PDOStatement. On place le résultat dans un objet $sql_exe. On remplace les valeurs dans notre requête SQL par nos marqueurs nommés entourés d'apostrophes.
Avec ce formatage, le risque d'injection SQL est limité.

ANNEXES

Sitographie :

Altorouter - PHP router that supports rest, dynamic and reversed routing. (s. d.). https://altorouter.com/

Basic usage - Composer. (s. d.). https://getcomposer.org/doc/01-basic-usage.md#autoloading

L'autoloading — formation la POO en PHP. (s. d.). Grafikart.fr. https://grafikart.fr/tutoriels/autoload-561

Using the Fetch API - web APIS | MDN. (2023, 7 juillet). https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Pavlutin, D. (2023). How to use Fetch() with JSon. *Dmitri Pavlutin Blog*. https://dmitripavlutin.com/fetch-with-json/

Javascript template Literals. (s. d.). https://www.w3schools.com/js/js_string_templates.asp

SQL Injection | OWASP Foundation. (s. d.). https://owasp.org/www-community/attacks/SQL_Injection

Cross Site Scripting (XSS) | OWASP Foundation. (s. d.). https://owasp.org/www-community/attacks/xss/