

Max Beaumet

Année 2022 – 2023



Dossier Projet : World Wide Sneakers

Remerciement

Je tenais à exprimer ma plus sincère gratitude pour cette année que j'ai passée à l'IMIE Paris en tant qu'étudiant en développement web et web mobile. Grâce à votre enseignement, j'ai acquis une grande quantité de connaissances et de compétences qui me seront très utiles pour mon avenir professionnel.

Je voudrais tout particulièrement remercier l'ensemble des enseignants et des formateurs qui ont suivi mon parcours tout au long de l'année. Votre expertise et votre passion pour le domaine du développement web ont été une source d'inspiration pour moi, et je suis fier d'avoir eu la chance de travailler à vos côtés.

Je tiens également à exprimer ma gratitude envers l'équipe administrative de l'école pour leur soutien tout au long de l'année. Votre disponibilité et votre aide ont été précieuses pour moi et ont grandement contribué à ma réussite.

Enfin, je tiens à remercier mes camarades de classe. J'ai eu la chance de travailler avec des personnes incroyablement talentueuses et motivées, et je suis fier de les compter parmi mes amis.

Table des matières

Remerciement	3
Table des matières.....	4
Présentation de mon projet :	4
Spécifications Techniques.....	6
Langage de programmation.....	6
1.1) PHP :.....	6
1.2) HTML et CSS :.....	6
1.3) SQL :.....	6
1.4) JavaScript et Jquery :	6
1.5) Bootstrap	7
Logiciel :	7
2.1) Postman	7
2.2) Visual Studio Code	7
2.3) StarUML:.....	7
2.3) Draw.io :.....	7
Outil :	8
3.1) Trelo :.....	8
3.2) W3C et W3C CSS :.....	8
Spécifications fonctionnelles	8
1.1) Création de mon Dictionnaire de donnée, MCD, MLD et MPD :	8
1.2) Base de données :.....	13
1.3) Diagramme de cas d'utilisation / diagramme de séquence :	14
Développement/Réalisation	16
1) La partie back de mon site :.....	17
1.1) db_connect.php et function.php :.....	17
1.2) users.php :	18
1.3) sneakers.php :.....	22
1.4) register.php :	26
1.5) login.php / logout.php :	27
1.6) mailer.php / mailer_question.php :	28

1.7) order.php :	29
1.8) update_newsletter :	31
1.9) Admin.php :	31
2) La partie Front :	34
2.1) register.html :	35
2.2) create_product.html :	37
2.3) FAQ.html :	39
2.4) catalogue.html.....	40
2.5) admin.html :	42
2.6) account.html :.....	43
2.7) welcome.html :	44
3) Javascript (Jquery) :	45
3.1) register.js :	45
3.2) login.js :	46
3.3) create_product.js :.....	47
3.4) page_catalogue.js :	48
3.5) FAQ.js :.....	54
3.6) Fonctionnalité js de la partie admin :	56
RGPD :	58
1) Panorama de la SSI :	59
1.1) Un monde numérique hyperconnecté :	59
1.2) Un monde à risque :	59
1.3) Acteur majeur de la cybersécurité :	61
1.4) Protéger le cyberespace :	62
1.5) Les règles d'or de la sécurité :	63
2) Sécurité de l'authentification :	63
2.1) Principe de l'authentification :	64
2.2) Attaque sur les mots de passe :	65
2.3) Sécuriser ses mots de passe :	65
2.4) Gérer ses mots de passe :	66
2.5) Notion de cryptographie :	66
3) Sécurité sur Internet	66
3.1) Internet de quoi s'agit-il ? :	66

3.2) Les fichiers en provenance d'internet :	67
3.3) La navigation web :	68
3.4) La messagerie électronique :.....	68
3.5) L'envers du décor d'une connexion web :.....	68
4) Sécurité du poste de travail et nomadisme :	69
4.1) Application et mise à jour :.....	69
4.2) Option de configuration de base :.....	69
4.3) Configuration complémentaire :	69
4.4) Sécurité des périphériques amovible :	70
4.5) Séparation des usages :	70
Conclusion.....	71
Table des illustrations	72
Glossaire (liste mot compliquer)	74
Webographie (site web visitée).....	74
Annexes.....	75

Présentation de mon projet :

Mon projet est un site Web dynamique et responsive de vente et d'achat de sneakers entre particulier ou entre un particulier et la plateforme. Ce site permettra à un utilisateur de créer un compte ou de se connecter à son compte s'il en possède déjà un. Il pourra ajouter les produits qu'il désire vendre. Il pourra aussi regarder l'ensemble des produits mis en vente par les autres utilisateurs et les acheter.

Sur le site, un vendeur pourra aussi modifier et supprimer ses offres. Un acheteur pourra acheter via la page catalogue et pourra aussi ajouter le produit dans son "panier".

La vente de produits se fera en plusieurs étapes : Ceci passera par la création du compte de l'utilisateur et le choix du produit qu'il veut vendre. Du point de vue de l'acheteur ceci se passera différemment : choix du produit et ajout dans son panier. Si un utilisateur a une question, il pourra la poser dans la page FAQ.

L'objectif de ce site est de permettre la revente de sneakers entre particulier et l'achat auprès de la plateforme. La cible de ce projet est toutes les personnes recherchant une nouvelle paire de sneakers, quel que soit leur budget ou le modèle recherché. Le plus de ce site est le fait qu'il n'y a aucun frais de la part de la plateforme.

Spécifications Techniques

Langage de programmation

1.1) PHP :

J'ai utilisé le langage de programmation PHP pour la partie back-end de mon projet. PHP qui signifie Hypertext Preprocessor, il a été utile pour récupérer les données de la base de données. Utilisé sur différentes pages, par exemple la page inscription pour l'insertion d'un nouvel utilisateur dans la base de données. Mais aussi cela peut permettre d'obliger des codes de rédaction lors de l'insertion de son email ou du mot de passe par l'utilisateur. Pour ça j'utilise des expressions régulières. Mais aussi par exemple, la modification ou la suppression de compte ou d'information.

1.2) HTML et CSS :

Le langage de balisage HTML (Hypertext Markup Language) m'a permis de développer le front des pages web et de mettre les éléments de type images, textes, formulaires ou autre qui composent le site. Ensuite, l'utilisation du langage CSS (feuille de style en cascade) qui a pour objectif de placer les éléments comme souhaité. Pour mon projet, j'utilise une page de CSS globale pour le placement d'éléments qui se répètent sur l'ensemble des pages.

1.3) SQL :

Structured Query Langage qui a pour objectif de permettre d'exploiter la base de données relationnelle. Dans le cadre de mon projet, j'ai utilisé ce langage pour l'ensemble des modifications qui peuvent être apportées sur la base de données et qui est un élément clé utilisé pour la partie back de mon projet. J'ai utilisé les requêtes SQL, telle que SELECT UPDATE DELETE et INSERT.

1.4) JavaScript et Jquery :

Javascript est un langage de programmation de script principalement employé pour les sites dynamiques. Il permet de lier le HTML et le back.

Jquery est une bibliothèque javascript libre et multiplateforme qui permet de simplifier la façon de coder.

1.5) Bootstrap

Bootstrap est un ensemble d'outil utile à la création de site web responsive qui veut dire qu'il s'adapte à la taille de l'écran sur lequel l'utilisateur est en interaction avec le site. J'ai utilisé sa dernière version qui est bootstrap 5.3. Bootstrap est un framework CSS.

Logiciel :

2.1) Postman

Postman est une application qui a pour objectif d'effectuer des requêtes sur la partie back de mon projet même si la partie front est déjà créée. Cette application fonctionne de façon très intuitive.

On rentre le chemin vers le doc.php en lien localhost. Ensuite on vérifie la méthode (POST, GET) qui est utilisé sur la page HTML.

2.2) Visual Studio Code

C'est l'application sur laquelle j'ai codé l'ensemble du site. Elle permet de coder dans pleins de langages différents tant que l'on précise le langage dans l'extension du document (doc.php, doc.html, doc.css, doc.js). VS code est un environnement de développement intégré.

2.3) StarUML:

J'ai utilisé Star UML pour le développement de la conception front.

2.3) Draw.io :

J'ai utilisé Draw.io pour le développement de Merise.

Outil :

3.1) Trelo :

Trelo est un site internet de gestion de projet. Je l'ai utilisé pour mon projet, car il m'a permis d'y voir plus clair et de mieux me rendre compte du travail qu'il me restait à produire tout au long du développement de ce projet. J'ai séparé le projet en quatre parties sur ce site selon le langage (HTML/CSS, JS, PHP) enfin une dernière partie pour ce qui est relatif au projet mais pas à la partie de codage.

3.2) W3C et W3C CSS :

Le site W3C m'a permis de vérifier la syntaxe de l'ensemble de mon code HTML. Quant au site W3C CSS, il m'a permis de vérifier la syntaxe de mes pages CSS.

Spécifications fonctionnelles

1.1) Création de mon Dictionnaire de donnée, MCD, MLD et MPD :

J'ai commencé par rédiger mon dictionnaire de donnée :

Table Utilisateur

Nom symbolique	Description	Type	Contrainte
id_utilisateur	Identifiant Utilisateur	N	Unique et Obligatoire
Émail	Email de l'utilisateur	AN	Obligatoire et unique
Nom	Nom de l'utilisateur	A	Obligatoire et unique
Prénom	Prénom de l'utilisateur	A	Obligatoire et unique
nom_d_utilisateur	Nom de l'utilisateur	A	Obligatoire et unique
mot_de_passe	Mot de passe du compte utilisateur	AN	Obligatoire et Unique
newsletter	Abonnement ou non de l'utilisateur a la newsletter du site	Booléen	Obligatoire et Unique
asset	Permet de désactiver un compte en cas de non respect des règles du site	N	Obligatoire

Table Commande :

Nom symbolique	Description	Type	Contrainte
numero_de_commande	Numéro de commande	N	Obligatoire et Unique
état_commande	État de la commande	A	Obligatoire
users_id	Identifiant Utilisateur	N	Unique et Obligatoire
id_sneakers	Identifiant produit de la sneakers	N	Obligatoire et unique
date_de_commande	Date de la commande	N	Obligatoire

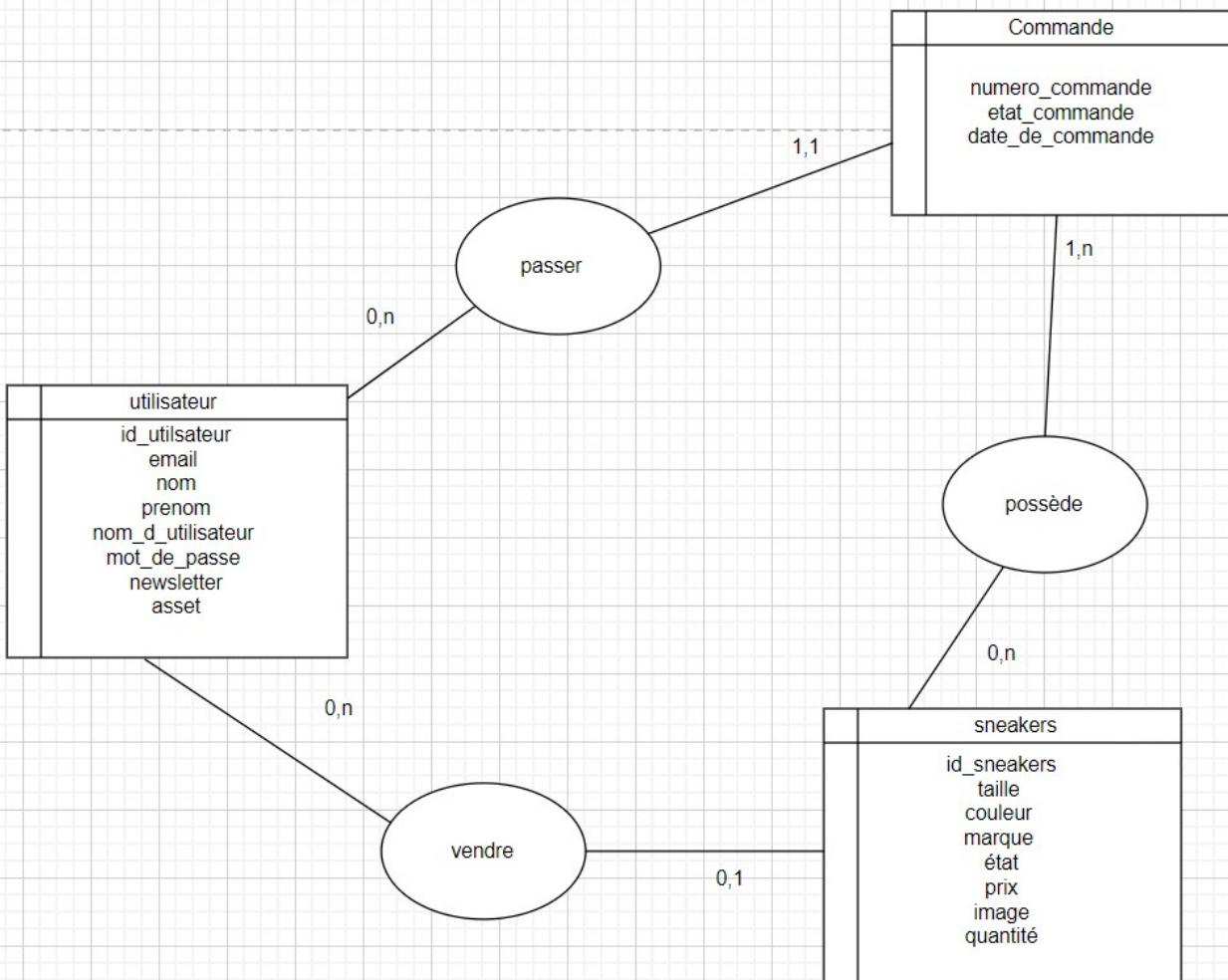
Table Sneakers :

Nom symbolique	Description	Type	Contrainte
Id_sneakers	Identifiant produit de la sneakers	N	Obligatoire et unique
Taille	Taille de la paire de sneakers	N	Obligatoire
Couleur	Couleur de la paire de sneakers	A	Obligatoire
Marque	Marque de la paire de sneakers	A	Obligatoire
Etat	Etat de la paire de sneakers	A	Obligatoire
Prix	Prix de l'article	N	Obligatoire
Image	Image de l'article	AN	NOT NULL
Quantité	Nombre de sneakers mise en vente	N	Obligatoire

Table Sneakers Order :

sneakers_id	Identifiant produit de la sneakers	N	Obligatoire et unique
numero_de_commande	Numéro de commande	N	Obligatoire et Unique

J'ai commencé par la méthode Merise avec la mise en place de mon mcd (modèle conceptuel de données) et mld (modèle logique de données). Pour créer ces deux modèles, j'ai utilisé l'application Draw.io



Mon modèle conceptuel de donnée (MCD) est composé de trois entités : utilisateur, commande et sneakers. L'entité utilisateur, comprend une liste de propriétés : id_utilisateur (propre à chaque utilisateur), email, nom, prénom, nom d'utilisateur, mot de passe, newsletter et admin.

Cette entité comprend deux cardinalités :

- La première cardinalité, est avec l'entité : commande. Elle est forte / faible car 0,n du côté utilisateur et 1,1 côté commande
- La seconde cardinalité, avec l'entité sneakers. Elle est forte / faible car 0,n du coté utilisateur et 0,1 du côté sneakers.

On s'intéresse maintenant à l'entité commande, qui comprend différentes propriétés : numéros de commande (unique), état de la commande, date de la commande. Cette table comprend deux cardinalités :

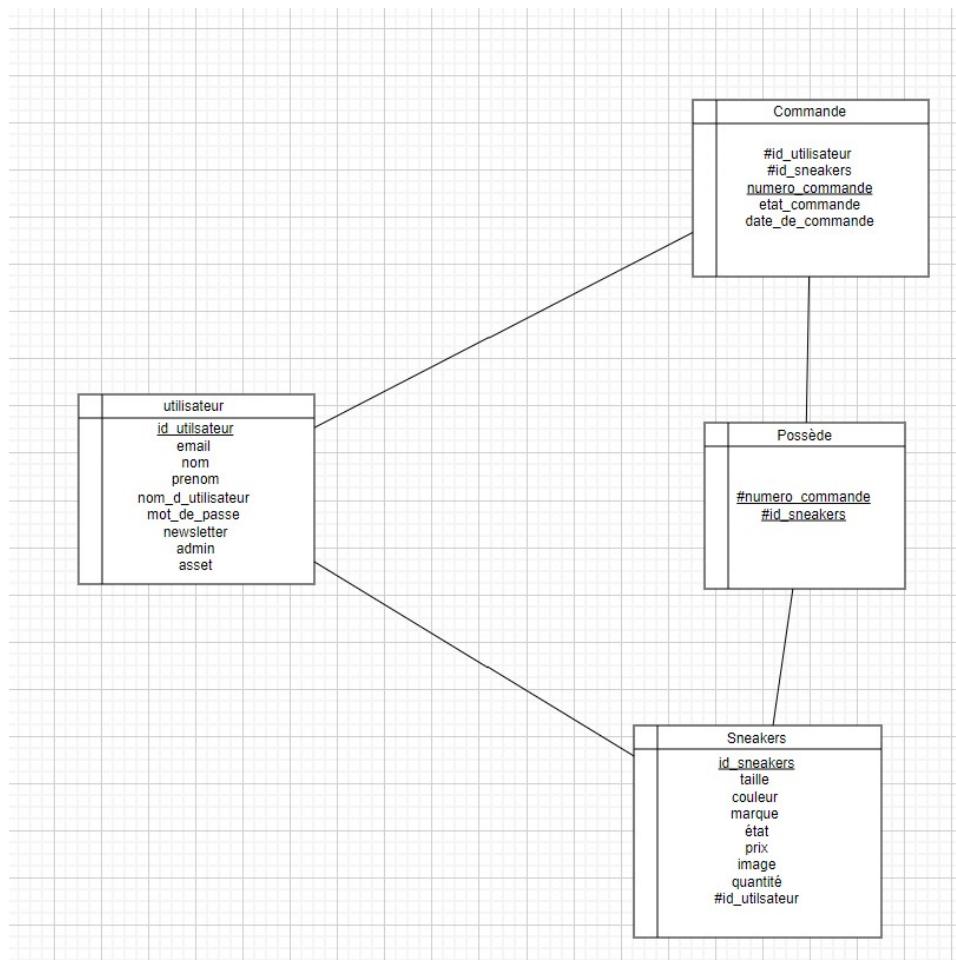
Une qui a déjà été évoquée plus haut, l'autre est avec l'entité sneakers. Cette seconde est forte/forte (1,n du côté commande /0,n du côté sneakers).

Enfin la dernière entité sneakers, qui possède un ensemble de propriété lié à une paire de sneakers comme : id_sneakers, taille, couleur, marque, état, prix, image et quantité.

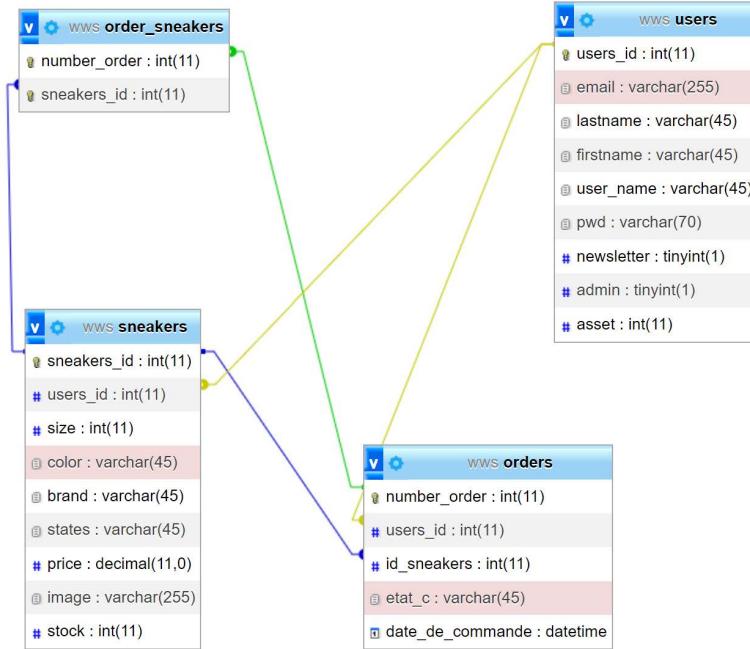
J'ai fait mon modèle logique de donnée (MLD), à partir du mcd. Il comprend une nouvelle entité qui se trouve entre les entités sneakers et commande. Elle se compose de deux propriétés :

- Numéro_commande (clé primaire et étrangère)
- id_sneakers (clé primaire et étrangère)

Une clé primaire est un champ ou un ensemble de champs d'une table qui contient des valeurs uniques. Une clé étrangère, une contrainte qui s'assure du respect de l'intégrité référentielle de la base de données.



Voici ci-dessous mon MPD (modèle physique de donnée) qui a pour but de préciser les types des colonnes de chaque entité.



J'ai choisi mon système de gestion de base de données relationnel (SGBDR) qui est MySQL.

1.2) Base de données :

Par la suite, j'ai développé ma base de données en SQL, a partir de mon MPD. Elle se compose de quatre tables dont une liant la table sneakers et commande. Mise en place des primary key et foreign key de chaque table. Mais aussi de chaque composante de la base de données.

```

28
29 CREATE TABLE orders (
30     number_order INT(11) auto_increment NOT NULL,
31     users_id INT(11) NOT NULL,
32     id_sneakers INT(11) NOT NULL,
33     etat_c VARCHAR(45) NOT NULL,
34     date_de_commande DATETIME NOT NULL,
35     PRIMARY KEY (number_order),
36     FOREIGN KEY (users_id) REFERENCES users(users_id),
37     FOREIGN KEY (id_sneakers) REFERENCES sneakers(sneakers_id)
38 );
39
40 CREATE TABLE Order_Sneakers (
41     number_order INT(11) NOT NULL,
42     sneakers_id INT(11) NOT NULL,
43     PRIMARY KEY (number_order,sneakers_id),
44     FOREIGN KEY (number_order) REFERENCES orders(number_order),
45     FOREIGN KEY (sneakers_id) REFERENCES sneakers(sneakers_id)
46 );
47

```

```

CREATE TABLE users (
    users_id INTEGER(11) auto_increment NOT NULL,
    email VARCHAR(255) NOT NULL,
    lastname VARCHAR(45) NOT NULL,
    firstname VARCHAR(45) NOT NULL,
    user_name VARCHAR(45) NOT NULL,
    pwd VARCHAR(70) NOT NULL,
    newsletter TINYINT(1) NOT NULL,
    asset INTEGER(11) NOT NULL,
    admin TINYINT(1) NOT NULL,
    PRIMARY KEY (users_id)

);

CREATE TABLE sneakers (
    sneakers_id INTEGER(11) auto_increment NOT NULL,
    users_id INTEGER(11) NULL,
    size INT(11) NOT NULL,
    color VARCHAR(45) NOT NULL,
    brand VARCHAR(45) NOT NULL,
    states VARCHAR(45) NOT NULL,
    price DECIMAL(11) NOT NULL,
    image VARCHAR(255) NOT NULL,
    stock INT(11) NOT NULL,
    PRIMARY KEY(sneakers_id),
    FOREIGN KEY(users_id) REFERENCES users(users_id)

);

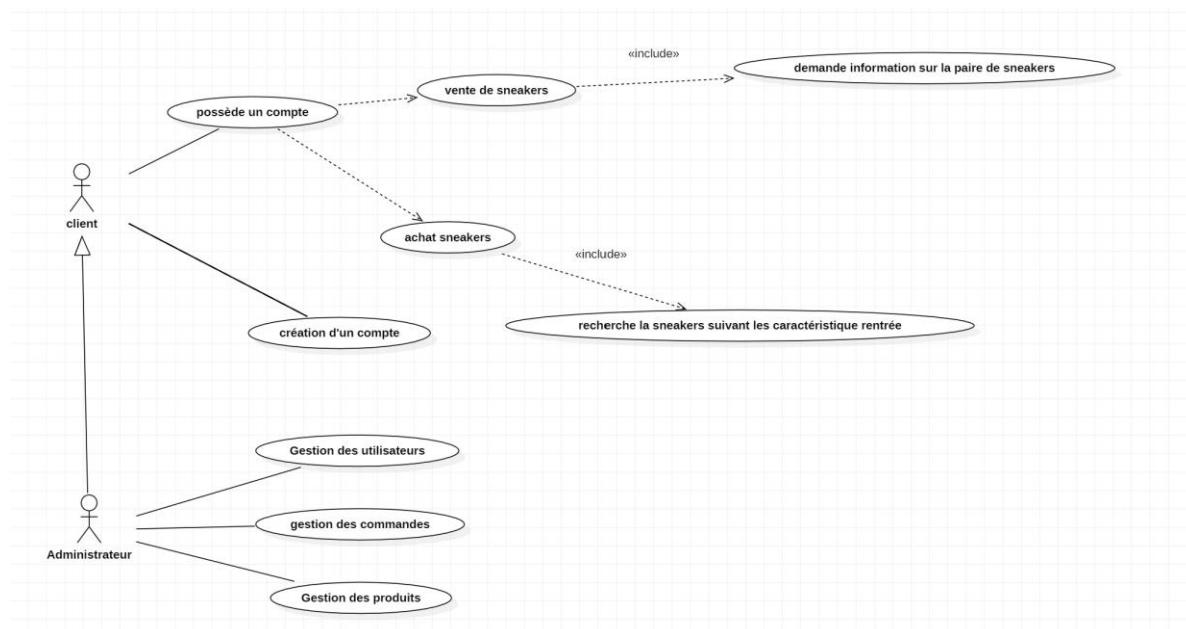
```

1.3) Diagramme de cas d'utilisation / diagramme de séquence :

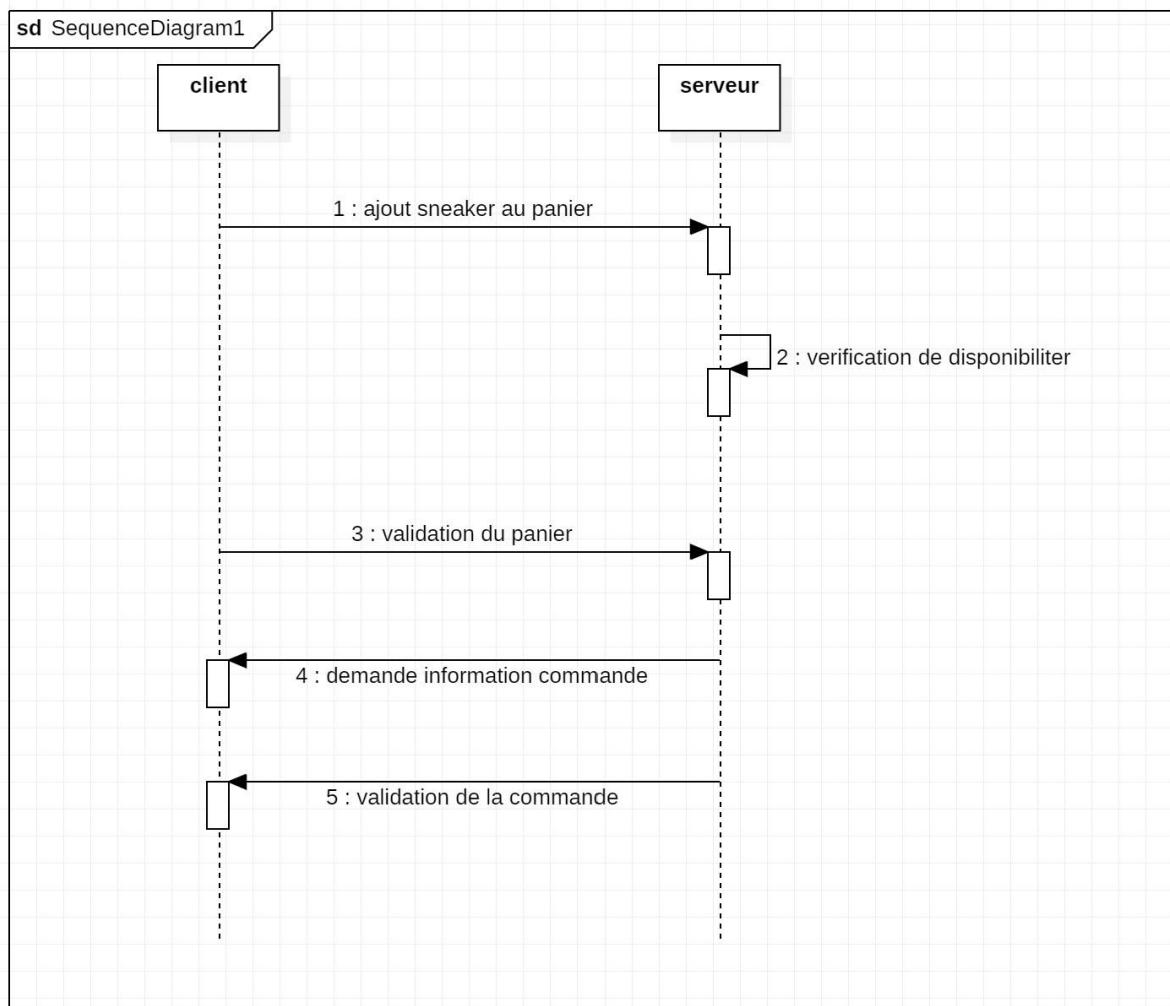
On débute cette partie en abordant la réalisation de mon UML (Unified Modeling Language). C'est un language de modélisation graphique.

Pour se faire j'ai réalisé mon diagramme de cas d'utilisation et diagramme de séquence.

Pour le diagramme de cas d'utilisation j'ai dégagé deux acteurs : client et administrateur. Un client peut posséder un compte ou s'en créer un. S'il en possède déjà un, il a la possibilité de vendre ou acheter un produit. En cas d'achat d'un produit ceci "inclus" la recherche de sneakers par son nom. S'il y a la vente d'une paire de sneakers ceci "inclus" la demande d'information sur la paire qui est vendue. Le second acteur, administrateur qui via une relation d'héritage peut faire l'ensemble des fonctionnalités de l'utilisateur et peut en plus gérer les utilisateurs, produits et commandes.



Mon diagramme de séquence, a pour but de voir un exemple d'interaction entre l'utilisateur et le site. On s'intéresse au passage de commande par le client. Il y a deux lignes de vie, une pour le client et une pour le serveur. Le diagramme débute par un message asynchrone (message envoyé qui n'attend pas de réponse) pour l'ajout au panier, puis un message réflexif de la part du serveur pour vérifier la disponibilité de la paire ajoutée au panier. Si cette condition est remplie alors un message asynchrone de la part du client a lieu pour la validation du panier. Puis, le serveur envoi deux messages auprès de l'utilisateur sur les informations commandes et enfin la validation de la commande.



Développement/Réalisation

Pour la réalisation de ce projet, j'ai débuté par la mise en place de ce qu'il y avait à faire. J'ai réparti les tâches, par langages de programmation. Pour cela j'ai utilisé Trelo une application en ligne de gestion de projet.

J'ai débuté en faisant les wireframes et maquettes de mon site internet évoqué précédemment puis j'ai codé la partie back du projet.

1) La partie back de mon site :

1.1) db_connect.php et function.php :

Ce fichier assure la connexion à la base de données.

🐘 db_connect.php > ...

```
1  <?php
2  $host = "localhost";
3  $username = "root";
4  $password = "";
5  $dbname = "WWS";
6
7  try {
8      $db = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
9  } catch (ErrorException $e) {
10      echo $e;
11 }
```

La page function.php a pour objectif de regrouper l'ensemble des fonctions dont j'aurais besoin dans plusieurs autres pages.

```

1  <?php
2  session_start();
3
4  function isAdmin(){
5      if(!$_SESSION["admin"]){
6          echo json_encode(["success"=>"false", "error"=>"vous n'êtes pas admin"]);
7          die;
8      }
9  }
10
11 function is_connected(){
12     if(!$_SESSION["connected"]){
13         echo json_encode(["success"=>"false", "error"=>"vous n'êtes pas connecté"]);
14         die;
15     }
16
17 }
18

```

```

19 function upload($file)
20 {
21     //? Si une image est transmise via le formulaire alors
22     if (isset($file["image"]["name"])){
23         /* Récupération du nom de fichier dans La superglobale FILES
24         $filename = $file["image"]["name"];
25
26         /* Chemin du fichier
27         $location = __DIR__ . "/../asset/$filename";
28
29         /* Récupération de L'extension du fichier
30         $extension = pathinfo($location, PATHINFO_EXTENSION);
31         $extension = strtolower($extension); /* Transformation de L'extension en minuscule
32
33         /* Liste des extensions possibles
34         $valid_extensions = ["jpg", "jpeg", "png"];
35
36         //? Si L'extension du fichier appartient au tableau des extensions valides alors
37         if (in_array($extension, $valid_extensions)) {
38             var_dump("location".$location);
39             var_dump("move".$file["image"]["tmp_name"]);
40             // var_dump($filename);
41             //? Si le fichier est bien enregistré à L'endroit souhaité alors
42             if (move_uploaded_file($file["image"]["tmp_name"], $location)) return $filename;
43             else return false;
44         } else return false;
45     } else return false;
46 }
47

```

1.2) users.php :

Ce fichier a pour objectif la gestion de l'ensemble des fonctionnalités que peut faire l'utilisateur.

Ce document débute par des appels vers les documents créés précédemment avec l'utilisation de `require_once` et `require`. La différence entre les deux sont le nombre d'appelle qu'on autorise vers ces documents.

Je défini en dessous un `if / else` qui permet de vérifier la méthode de récupération : information utilisateur. J'ai mis une condition qui test si la méthode de récupération d'information par le server est POST, alors la variable "méthode" prend cette valeur sinon elle est égale à la super globale GET.

Ensuite, j'ai mis en place un switch avec en paramètre la variable méthode et entre crochet l'option, choisi parmi celle qui composent le switch.

Le switch est composé de différentes options : `select_id`, `update_id`, `delete_id`, `update_newsletter` et `question_FAQ`.

- J'ai choisi que l'utilisateur puisse voir ses informations par rapport à son `users_id` avec le `select_id`.
- Ensuite, il y a le `update_id` qui permet à l'utilisateur de mettre à jour ses informations suivant son `users_id`.
- Aussi la possibilité pour l'utilisateur de s'inscrire à la newsletter
- Mais aussi la récupération et l'envoi en mail des questions posées par l'utilisateur sur la page FAQ.
- Enfin, le `delete_id` qui permet à un utilisateur de supprimer son compte.

Dans l'ensemble des requêtes de cette page, j'ai mis la condition qui assure que les informations qui sont affichées ou mise à jour ou supprimées de la base de données soit bien celle de l'utilisateur. Pour cela, je me base sur le users_id de la personne connectée (récupérer avec la superglobale \$_SESSION) et s'il concorde avec une des lignes de la base de données users.

Pour ce qui est de la case du switch : select_id, le code qui la compose est une condition qui vérifie que le users_id qui est récupéré avec la superglobale SESSION est bien existant et non vide.

Cette super globale SESSION a besoin pour son activation du code suivant "session_start()". Si les conditions sont remplies, alors la lecture du code continue avec une requête SQL préparée qui sélectionne les données avec l'id de l'utilisateur connectée, pour cela j'utilise une condition (WHERE) avec l'id_users récupérée de la superglobale SESSION. La requête est ensuite exécutée.

Sur une autre ligne, je crée une variable "user" qui va récupérer les données souhaitées et les insérer dans le tableau associatif grâce à l'utilisation du fetch(PDO::FETCH_ASSOC).

Si c'est un succès alors j'utilise echo json encode pour afficher le résultat.

Enfin, à l'extérieur de cette condition, je mets un break pour que le code soit directement arrêté et qu'il n'aille pas lire la suite du code de la page.

```
users.php > ...
1  <?php
2  require_once("db_connect.php");
3  require("function.php");
4 // is_Connected();
5
6 if ( $_SERVER["REQUEST_METHOD"] == "POST" ) {
7     $method= $_POST;
8 } else {
9     $method= $_GET;
10 }
11
12 switch($method["opt"]){
13     case "select_id":
14         if(isset($_SESSION["users_id"]) && !empty(trim($_SESSION["users_id"]))){
15             $req=$db->prepare("SELECT * FROM users WHERE users_id=?");
16             $req->execute([$_SESSION["users_id"]]);
17             $user=$req->fetch(PDO::FETCH_ASSOC);
18             echo json_encode(["success"=> true, "user"=>$user]);
19         }
20         break;
```

La prochaine case du switch est la case : update_id.

Elle commence par une condition qui s'assure que le prénom (firstname), nom de famille (lastname), email et users_name de l'utilisateur soit bien rentrées et définies. Sur la ligne suivante, je défini une variable vide password.

Ensuite, on a une condition qui vérifie que le password (mot de passe) est défini et non vide, via la superglobale POST dans le formulaire.

La variable définie précédemment est de nouveau appelée et on lui attribue la valeur du nouveau mot de passe. Cette valeur est ensuite vérifiée par la variable regex avec l'utilisation de preg_match qui s'assure que le mot de passe est bien conforme aux règles de sécurité relatives au mot de passe et pour limiter au maximum toute intrusion au sein du compte. Si le mot de passe est non conforme, alors j'affiche une erreur avec echo json_encode : "mot de passe au mauvais format" et alors on sort de la condition avec l'utilisation de die. Sinon le mot de passe est haché.

Dans le cas où le mot de passe rentré est resté vide, on a une requête préparée qui va modifier les données de l'utilisateur, sauf son mot de passe avec comme condition son users_id.

Dans le cas où le mot de passe est en accord avec les règles de sécurité mise en place par la regex alors on a une requête préparée avec une modification en plus du mot de passe toujours avec la même condition, celle du users_id.

A la suite de ceci, soit le message qui sera affiche est : changement effectué (en cas de succès) ou sinon erreur de maj (en cas d'échec).

```

21
22 case "update_id":
23     if(isset($_POST["firstname"], $_POST["lastname"], $_POST["email"],$_POST["user_name"]) && !empty(trim($_POST["firstname"])) && !empty(trim($_POST["lastname"])) &&
24     !empty(trim($_POST["email"])) && !empty(trim($_POST["user_name"]))) {
25         $password = "";
26
27         if(isset($_POST["pwd"]) && !empty(trim($_POST["pwd"]))) {
28             $password= "pwd";
29             $regex = "/^(?=.*\d)(?=.*[A-Z])(?=.*[a-z])[a-zA-Z0-9]{8,12}$/";
30             if (!preg_match($regex, $_POST["pwd"])) {
31                 echo json_encode(["success" => false, "error" => "Mot de passe au mauvais format"]);
32                 die;
33             }
34             $hash = password_hash($_POST["pwd"], PASSWORD_DEFAULT);
35         }
36
37         $req = $db->prepare("UPDATE users SET firstname = :firstname, lastname = :lastname, email = :email, user_name=:user_name WHERE users_id = :users_id");
38
39         $req->bindValue(":firstname", $_POST["firstname"]);
40         $req->bindValue(":lastname" , $_POST["lastname"]);
41         $req->bindValue(":email" , $_POST["email"]);
42         $req->bindValue(":user_name" , $_POST["user_name"]);
43         $req->bindValue(":users_id",$_SESSION["users_id"]);
44         $req->bindValue(":newsletter",$_GET["newsletter"]);
45         $req->execute();
46
47         if($password != "") {
48             $req = $db->prepare("UPDATE users SET firstname = :firstname, lastname = :lastname, email = :email, user_name=:user_name,pwd=:pwd WHERE users_id = :users_id");
49
50             $req->bindValue(":firstname", $_POST["firstname"]);
51             $req->bindValue(":lastname" , $_POST["lastname"]);
52             $req->bindValue(":email" , $_POST["email"]);
53             $req->bindValue(":user_name" , $_POST["user_name"]);
54             $req->bindValue(":pwd" , $hash);
55             $req->bindValue(":users_id",$_SESSION["users_id"]);
56
57             $req->execute();
58         }
59         echo json_encode(["success" => true, "msg"=> "changement effectué"]);
60     } else {
61         echo json_encode(["success" => false, "error" => "erreur de MAJ"]);
62     }
63     break;
64
65

```

Dans la dernière case du switch : delete_id. Celle-ci est la moins complexe, elle vérifie à travers une condition l'existence d'un users_id et qu'il est non vide.

Puis une requête préparée s'assure de la suppression de ces données puis l'utilisation d'un die pour arrêter la lecture du code qui suit. De façon à ce qu'elle ne rentre pas dans la case default qui est présente en cas de demande inconnue.

Après avoir fini de coder cette page, j'ai bien sûr testé chacune des cases qui la compose sur l'application de bureau : postman.

```

users.php > ...
66
67     case 'delete_id':
68         if (isset($_SESSION["users_id"]) && !empty(trim($_SESSION["users_id"]))) {
69             $req=$db->prepare("DELETE FROM users WHERE users_id=?");
70             $req->execute([$_SESSION["users_id"]]);
71             echo json_encode([{"success":true }]);
72
73         }else{
74             echo json_encode([{"success":false, "error":"erreur de suppression"}]);
75         }
76         die;
77
78     default:
79         echo json_encode([{"success":false, "error":"demand inconnu"}]);
80
81     break;
82 }
```

1.3) sneakers.php :

Cette page a pour objectif de permettre la sélection, l'insertion, la modification ou la suppression des données des sneakers en base de données. Les différentes données récupérées sont le nom/marque de la sneakers (brand), taille (size), couleur dominante (color), l'état des sneakers lors de sa vente (states), le prix (price), l'image du produit (image) et le nombre de paires qui vont être mise en vente (stock).

Dans cette page, j'ai choisi que le vendeur puisse faire différentes actions : ajout d'un produit, la mise à jour et la suppression.

Pour les deux dernières, j'ai bien sur mis comme condition l'user id de façon à être sûr que la modification ou la suppression d'une offre de sneakers soit bien faite par celui qui est à l'origine de cette offre.

Au début de cette page php, on y trouve deux appels vers les pages php :

- db_connect.php (page de connexion avec la base de données)
- function.php (regroupant les fonctions utiles dans les différentes page php).

Ensuite, on y trouve une condition qui s'assure que la méthode de récupération de données choisies soit la même que celle attendu pour le choix dans le switch de la page sneakers.php.

La première option du switch : select. Celle-ci a pour but de sélectionner toutes les données de la base de données. Pour arriver à ce résultat, j'utilise une requête non préparée (query) car pas de donnée provenant de l'extérieur donc pas de risque d'injection SQL. La requête est la suivante : "SELECT * FROM sneakers", l'étoile permet de sélectionner toutes les colonnes d'une table et par conséquent d'en extraire toutes ses données.

Sur la ligne qui suit, je crée une variable sneakers qui va me permettre de trier les données récupérées par la variable de la ligne précédente qui comprend la requête SQL. Pour ce tri de donnée, j'utilise la propriété php, fetchAll et PDO::FETCH_ASSOC qui permet la récupération et le tri des données.

J'ai ensuite codé une nouvelle case du switch le : select_id, qui peut être utile si un utilisateur souhaite voir seulement ses produits. Pour cela j'ai mis une condition qui s'assure de l'existence d'un users_id dans la superglobale SESSION. Si celui ci existe : j'ai codé une requête similaire avec une condition qui s'assure que le users_id est bien existant en bdd. Sinon, j'affiche une erreur avec comme message : "donnée manquantes".

La case suivante est identique à la précédente sauf qu'elle a comme condition le sneakers_id récupéré avec la superglobale POST.

```

1  <?php
2
3  require_once("db_connect.php");
4  require("function.php");
5  is_connected();
6
7  if ($_SERVER["REQUEST_METHOD"] == "POST") $method = $_POST;
8  else $method = $_GET;
9
10
11 switch($method["opt"]){
12     case 'select':
13
14         $req = $db->query("SELECT * FROM sneakers ");
15         $sneakers = $req->fetchAll(PDO::FETCH_ASSOC);
16         echo json_encode(["success" => true, "sneakers" => $sneakers]);
17         break;
18
19     case 'select_id':
20         if (isset($_SESSION["users_id"])){
21
22             $req=$db->prepare("SELECT * FROM sneakers WHERE sneakers_id=?");
23             $req->execute([$_SESSION["users_id"]]);
24             $users=$req->fetch(PDO::FETCH_ASSOC);
25             echo json_encode(["success"=>true,"sneakers"=>$users]);
26
27         } else{
28             echo json_encode(["success"=>false,"error"=>"erreur données manquantes"]);
29         }
30
31         break;
32
33     case 'select_sneakers_id':
34
35         if (isset($_POST["sneakers_id"])){
36             $req=$db->prepare("SELECT * FROM sneakers WHERE sneakers_id=?");
37             $req->execute($_POST["sneakers_id"]);
38             $sneakers=$req->fetch(PDO::FETCH_ASSOC);
39             echo json_encode(["success"=>true,"sneakers"=>$sneakers]);
40
41         } else{
42             echo json_encode(["success"=>false,"error"=>"erreur données manquantes"]);
43         }
44
45         break;
46

```

La case suivante est : insert, elle permet d'ajouter les données d'une paire de sneakers. Pour cela, on s'assure d'abord que l'ensemble de ces données soit entrées (size, color, brand, states, price, image, stock).

Pour s'assurer que la récupération d'images est bien réussie, j'utilise isset qui s'assure que la variable est existante et !empty() qui vérifie que la variable est non vide et trim pour enlever des espaces en trop au début et à la fin de la valeur.

Si la vérification est réussie, alors on va utiliser la méthode de PDO : bindValue(), qui permet de faire passer les valeurs rentrées par l'utilisateur dans la requête sql.

Pour l'image, ceci se passe différemment, car il y a plusieurs vérifications à faire avant

:

- Qu'il n'y a pas d'erreur lors de l'envoi du fichier
- Le document est bien une image
- L'extension du fichier

Si tous les tests sont passés, alors l'image est déplacée vers le dossier prévu pour les stocker.

Pour vérifier l'envoi du fichier, je m'assure que le nombres d'erreurs est égal à zéro. Après avoir fait toutes ces vérifications, on contrôle qu'il y est bien une image récupérée puis on crée une variable img qui va récupérer le résultat de la fonction upload().

Cette fonction est codée dans le fichier function.php qui permettra de récupérer le chemin de l'image qui sera ensuite envoyé en base de données. Pour faire ceci j'utilise la méthode bindValue() et exécute la requête php. Si tout cela se passe bien, alors on affiche un message de succès.

```
51 case 'insert':  
52     if (isset($_POST["size"],$_POST["color"],$_POST["brand"],$_POST["states"],$_POST["price"],$_POST["stock"]) && !empty(trim($_POST["size"])) && !empty(trim($_POST["color"])) && !empty(trim($_POST["brand"])) && !empty(trim($_POST["states"])) && !empty(trim($_POST["price"])) && !empty(trim($_POST["stock"]))) {  
53  
54         // Requête préparée avec l'utilisation d'une requête SQL INSERT qui permet d'insérer les information d'une sneakers au sein de la base de donnée d'après ceux qui  
55         // est rentrée par l'utilisateur dans le formulaire de création produit  
56         $req= $db->prepare("INSERT INTO sneakers (brand,color,price,size,states,image,stock,users_id) VALUES (:brand,:color,:price,:size,:states,:image,:stock,:users_id)  
57         ");  
58  
59         // brand -> $_POST[brand]  
60         $req->bindValue(":brand", $_POST["brand"]);  
61  
62         // color -> $_POST[color]  
63         $req->bindValue(":color", $_POST["color"]);  
64  
65         // price -> $_POST[price]  
66         $req->bindValue(":price", $_POST["price"]);  
67  
68         // size -> $_POST[size]  
69         $req->bindValue(":size", $_POST["size"]);  
70  
71         // states -> $_POST[states]  
72         $req->bindValue(":states", $_POST["states"]);  
73  
74     if (isset($_FILES['image'])) {  
75         // Récupère les informations sur le fichier  
76         $file_name = $_FILES['image']['name'];  
77         // print($file_name);  
78         $file_tmp = $_FILES['image']['tmp_name'];  
79         $file_size = $_FILES['image']['size'];  
80         $file_error = $_FILES['image']['error'];  
81  
82  
83         $img = ""; // Je défini img à vide  
84         if (isset($_FILES["image"]["name"])){  
85             $img = upload($_FILES); // Je récupère la réponse de l'upload  
86         } if ($img) $req->bindValue(":img", "..../asset/" . $img); // Je bind le chemin si la réponse de upload() n'est pas false  
87         else $req->bindValue(":img", null); // Je bind null sinon  
88  
89     }  
90 }
```

```

91     // Vérifie s'il y a une erreur lors de l'envoi du fichier
92     if ($file_error === 0) {
93         // Vérifie si le fichier est une image
94         $file_ext = pathinfo($file_name, PATHINFO_EXTENSION);
95         $allowed_ext = array('jpg', 'jpeg', 'png', 'gif');
96         if (in_array($file_ext, $allowed_ext)) {
97             // Déplace le fichier téléchargé vers un dossier de stockage
98             $destination = 'asset/product_img' . $file_name;
99             if (move_uploaded_file($file_tmp, $destination)) {
100                 // Affiche l'image téléchargée
101                 echo "<img src='$destination'>";
102             } else {
103                 echo "Erreur lors de l'enregistrement du fichier.";
104             }
105         } else {
106             echo "Le fichier doit être une image de type JPG, JPEG, PNG ou GIF.";
107             break;
108         }
109     } else {
110         echo "Erreur lors de l'envoi du fichier.";
111         break;
112     }
113 }
114
115 // stock -> $_POST[stock]
116 $req->bindValue(":stock", $_POST["stock"]);
117
118 // users_id -> $_SESSION[users_id]
119 $req->bindValue(":users_id", $_SESSION["users_id"]);
120

```

```

125
126
127     $req->execute();
128     echo json_encode([{"success":true});
129
130 } else {
131     // print_r($_POST);
132     // print_r($_FILES);
133     echo json_encode([{"success":false, "error":"erreur lors de l'insertion"}]);
134
135 };
136
137 break;
138
139

```

Continuons avec le choix du switch : ‘search’, qui a pour objectif la recherche d’un élément selon les données de la base de donnée.

Cette page débute en vérifiant que la variable search soit bien récupérée et non vide avec la super globale GET. Je m’assure aussi qu’il n’y est pas d’espace en trop sur la variable search. Par une requête préparée, je sélectionne l’ensemble des données de la base de données sneakers qui sont similaires au données text contenu dans la variable search. Puis je mets en forme les données récupérer et j’affiche un message de succès sinon un message d’erreur : ‘error’.

```

148
149 case 'search':
150     if(isset($_GET['search']) && !empty(trim($_GET['search']))){
151         $req=$db->prepare("SELECT * FROM sneakers WHERE brand LIKE ?");
152         for ($i=0; $i<1; $i++){
153             $database[] = "%{$_GET['search']}%";
154             $req->execute($database);
155             $sneakers = $req->fetchAll(PDO::FETCH_ASSOC);
156             echo json_encode(["success"=>true,"sneakers"=>$sneakers]);
157         }
158
159     }else{
160         echo json_encode(["success"=>false,"msg"=>"error"]);
161     }
162
163     break;
164

```

Passons maintenant à la case “update_id”. Celle-ci permet à un utilisateur de mettre à jour un de ses produits qu’il a mis en ligne. Tout d’abord, vérifier que l’ensemble des données attendu soit non vide et qu’il n’y est pas d’espace en trop.

Ensuite, mise en place de la requête SQL, avec la condition que le users_id récupéré avec la superglobale SESSION soit le même que celui de la paire de sneakers.

Puis, on utilise la méthode bindValue() sur les valeurs, ce qui signifie : associer les valeurs à un nom correspondant.

Si ceci fonctionne, alors un message de succès est affiché. Sinon un message d’erreur : “erreur lors de la mise à jour” est affiché.

La case qui la suit est update_sneakers_id, celle-ci est assez similaire, elle s’assure juste que le sneakers_id est bien défini.

```

167
168 case 'update_id':
169     if (isset($_POST["size"],$_POST["color"],$_POST["brand"],$_POST["states"],$_POST["price"],$_POST["image"],$_POST["stock"]) && !empty(trim($_POST["size"])) && !empty(trim($_POST["color"])) && !empty(trim($_POST["brand"])) && !empty(trim($_POST["states"])) && !empty(trim($_POST["price"])) && !empty(trim($_POST["image"]))) {
170         $req= $db->prepare("UPDATE sneakers SET size=:size,color=:color,brand=:brand,states=:states,image=:image,price=:price,stock=:stock WHERE users_id = :users_id");
171         $req->bindValue(":size",$_POST["size"]);
172         $req->bindValue(":color",$_POST["color"]);
173         $req->bindValue(":brand",$_POST["brand"]);
174         $req->bindValue(":states",$_POST["states"]);
175         $req->bindValue(":price",$_POST["price"]);
176         $req->bindValue(":image",$_POST["image"]);
177         $req->bindValue(":stock",$_POST["stock"]);
178         $req->bindValue(":users_id",$_SESSION["users_id"]);
179         $req->execute();
180         echo json_encode([{"success":true});
181     }else{
182         echo json_encode([{"success":false, "error":"erreur lors de la mise à jour"});
183     }
184
185     break;

```

Maintenant, intéressons-nous à la case : “delete_id”. Tout d’abord on vérifie que le users_id soit non vide et définis pour s’assurer que l’article qui désire être supprimer soit bien celui de l’utilisateur connecté. On met ensuite la requête avec comme condition son users_id.

Il reste juste la case “default”, au cas où le choix fait, ne fasse pas parti de l’un des choix vu avant. Avec un message, “demande inconnue” pour bien spécifier que la demande est inconnue.

```

187
188 case 'delete_id':
189     if (isset($_SESSION["users_id"]) && !empty(trim($_SESSION["users_id"]))) {
190         $req=$db->prepare("DELETE FROM sneakers WHERE users_id=?");
191         $req->execute([$_SESSION["users_id"]]);
192         echo json_encode([{"success":true }]);
193     }else{
194         echo json_encode([{"success":false, "error":"erreur de suppression"}]);
195     }
196
197     break;
198
199 default:
200     echo json_encode(["success" => false, "error" => "Demande inconnue"]);
201     break;
202 }

```

1.4) register.php :

Par la suite, je me suis intéressé à la partie back d'une des fonctionnalités principales de mon site : l'inscription / création de compte. Au début de mon code, j'appelle les deux fichiers : db_connect.php, mailer.php.

- Le premier étant d'avoir la connexion avec la base de données.
- Le second permet lors de l'inscription, d'envoyer un mail à l'adresse entrée lors de l'inscription.

Je vérifie ensuite que l'ensemble des données utilisateur soit non vide. Je crée une variable regex qui me permettra de vérifier la forme de l'email rentré par l'utilisateur.

Puis, je réitère cette action pour vérifier la forme du mot de passe. Ces deux vérifications ont pour objectif de sécuriser au maximum le compte de l'utilisateur.

Si ces deux vérifications sont validées, alors une requête SQL va permettre d'enregistrer les données utilisateur en base de données. J'utilise une requête préparée pour des questions de protections des données et de sécurité.

Ensuite, j'ai utilisé la méthode bindValue() sur l'ensemble des valeurs récupérées. En dernière ligne, je fais l'envoi de mail si l'inscription a fonctionnée et pour cela j'utilise le système de mailer.

```

register > register.php > ...
1  <?php
2  require_once('../db_connect.php');
3  require("../mailer.php");
4
5  if (isset($_POST["firstname"], $_POST["lastname"],$_POST["email"],$_POST["user_name"],$_POST["pwd"])) && !empty(trim($_POST["firstname"])) && !empty(trim($_POST["lastname"])) && !empty(trim($_POST["email"])) && !empty(trim($_POST["user_name"])) && !empty(trim($_POST["pwd"]))) {
6
7  } else {
8      echo json_encode(["success" => false, "error" => "Donnée vide"]);
9      die;
10 }
11
12 $regex = "/^([a-zA-Z0-9-+_.]+@[a-zA-Z0-9-]{2,}\.[a-zA-Z]{2,}$)/";
13 if (!preg_match($regex, $_POST["email"])){
14     echo json_encode(["success" => false, "error"=> "email au mauvais format"]);
15     die;
16 }
17
18 $regex = "/^(?=.*\d)(?=.*[A-Z])(?=.*[a-z])[a-zA-Z0-9]{8,12}$/";
19
20 if (!preg_match($regex, $_POST["pwd"])) {
21     echo json_encode(["success" => false, "error" => "Mot de passe au mauvais format"]);
22     die;
23 } else {
24     $hash = password_hash($_POST["pwd"], PASSWORD_DEFAULT);
25     $req = $db->prepare("INSERT INTO users (firstname, lastname, email, user_name, pwd) VALUES (:firstname, :lastname, :email, :user_name, :pwd)");
26     $req->bindValue(":firstname", $_POST["firstname"]);
27     $req->bindValue(":lastname", $_POST["lastname"]);
28     $req->bindValue(":user_name", $_POST["user_name"]);
29     $req->bindValue(":email", $_POST["email"]);
30     $req->bindValue(":pwd", $hash);
31     $req->execute();
32     echo json_encode(["success" => true, "msg" => "inscrit"]);
33     mailer("m.beaumet@gmail.com", "Bienvenu {$_POST["firstname"]} {$_POST["lastname"]}", "Merci de ton inscription");
34 }
35

```

1.5) login.php / logout.php :

Passons maintenant à la partie connexion / déconnexion en php. Commençons d'abord sur le fichier login.php.

Ce fichier débute avec un session start(), qui permet d'initialiser une session data. On vérifie ensuite que soit le username ou le mail et le mot de passe sont bien non vide.

On utilise une requête SQL pour comparer les données entrées par l'utilisateur avec celle dans la base de données. Si des valeurs dans la base de données concordent avec celle rentrées par l'utilisateur, on récupère dans le localstorage le users_id, si l'utilisateur est un administrateur et s'il est abonnée a la newsletter. Si le mot de passe est erroné alors on affiche une erreur "mot de passe erroné".

```

login_logout > login.php > ...
1  <?php
2  session_start();
3  require_once("../db_connect.php");
4
5 // Si les éléments username_email et pwd sont remplis et non vides dans le formulaire alors le code à l'intérieur du if est alors exécuter.
6 if(isset($_POST["username_email"])) && isset($_POST["pwd"]) && !empty(trim($_POST["username_email"])) && (!empty(trim($_POST["pwd"]))) {
7 // requête SQL qui permet de sélectionner dans la table user les données : mot de passe, l'id user et enfin si l'utilisateur est un admin.
8 $req = $db->prepare("SELECT pwd,users_id,admin FROM users WHERE email=? OR user_name=?");
9 // exécution de la requête SQL pour préparer avec la récupération des données du formulaire. Ceci permettra de voir si les données formulaire existe dans la base de donnée.
10 $req->execute($_POST["username_email"],$_POST["username_email"]);
11 // Permet de récupérer l'ensemble des infos de la requête
12 $user = $req->fetch(PDO::FETCH_ASSOC);
13
14 // Si la variable user est non vide et que le mot de passe de la base de donnée est le même que celui entrer dans le formulaire alors le code à l'intérieur du if est executé
15 if ($user && password_verify($_POST["pwd"], $user["pwd"])) {
16     // Cette ligne permet de passer la valeur de connecter à true pour savoir si l'utilisateur est bien connecter.
17     $_SESSION["connected"] = true;
18     // Je passe la variable id de l'utilisateur de la session à la valeur de l'id de l'utilisateur
19     $_SESSION["users_id"] = $user["users_id"];
20     // Je passe la variable admin à la valeur de l'utilisateur qui se connecte soit 0 ou 1.
21     $_SESSION["admin"] = $user["admin"];
22
23     unset($user["pwd"]);
24
25     // Renvoie un message de success à true
26     echo json_encode(["success" => true, "user" => $user]);
27     die;
28 } else {
29     echo json_encode(["success"=>false, "error"=>"mot de passe erroné"]);
30 }
31 } else {
32     $_SESSION = [];
33     echo json_encode(["success" => false, "error" => "donnée manquante"]);
34 }

```

On s'intéresse maintenant au fichier logout.php: ce fichier ne contient que quatres lignes de code : tout d'abord un session_start() et ensuite un session_destroy() qui détruit la session. Alors un message est envoyé si la session est bien détruite.

```

1  k?php
2
3  session_start();
4
5  session_destroy();
6
7  echo json_encode(["success" => "true", "msg" => "deconnecter"]);

```

1.6) mailer.php / mailer_question.php / mailer_commande.php :

Ce fichier a pour objectif l'envoi de mail et débute avec deux lignes qui appelle deux autres fichiers : PHPMailer et Exception qui ont été créé lors de l'appel dans le terminal de : “composer”. Composer est un gestionnaire de dépendance utiliser pour l'installer php mailer.

Ensuite, on a une fonction qui reçoit trois paramètres en entrée : \$to (pour qui est le mail), \$subject (objet du mail) et enfin \$body (contenu du mail). Cette fonction permet donc l'envoi de mail automatique notamment utilisé lors de l'inscription.

```

1 <?php
2
3 use PHPMailer\PHPMailer\PHPMailer;
4 use PHPMailer\PHPMailer\Exception;
5
6 require("vendor/autoload.php");
7
8 function mailer($to, $subject, $body)
9 {
10     $mail = new PHPMailer();
11
12     try {
13         $mail->IsSMTP();
14         $mail->SMTPDebug = 0;
15         $mail->SMTPAuth = true;
16         $mail->SMTPSecure = "tls";
17         $mail->Host = "smtp-mail.outlook.com";
18         $mail->Port = 587;
19         $mail->Username = "Max.beaumet@imie-paris.fr";
20         $mail->Password = "Papa2002!";
21         $mail->SetFrom("max.beaumet@imie-paris.fr", "Sujet Test");
22
23         $mail->Subject = $subject;
24         $mail->Body = $body;
25         $mail->AddAddress($to);
26
27         $mail->send();      █
28     } catch (Exception $e) {
29         echo "error: $e";
30     }
31 }
32
33

```

1.7) order.php :

Cette page a pour objectif, la gestion des différentes fonctionnalités en rapport avec la création, la sélection, la modification et la suppression de commande en php.

Il permet pour l'utilisateur de passer une commande. Pour cela, on a toujours les deux mêmes fichiers nommés en début de document : function.php et db_connect.php.

Par la suite, on vérifie superglobale utilisée et ensuite on a un switch avec trois choix possibles : select_users_id, select_num et insert.

- La première case, vérifie si le users_id est non vide. Si la condition est remplie alors s'en suit une requête SQL qui a pour objectif de sélectionner l'ensemble des commandes passées par l'utilisateur selon son users_id. C'est donné récupérées sont triées et stockées dans la variable order. Pour faire se trie, j'utilise PDO::FETCH_ASSOC. Je termine ce choix du switch avec un message de succès. Sinon un message d'erreur, "pas de commande avec cette id".

```

1 <?php
2 require_once("db_connect.php");
3 require("function.php");
4 //isAdmin();
5
6 if ($_SERVER["REQUEST_METHOD"] == "POST") $method = $_POST;
7 else $method = $_GET;
8
9 switch($method['opt']){
10
11     // - selection de commande par id : permet de voir les commande selon l'id client
12     // - vérification de l'id user ou l'id sneakers
13     case 'select_users_id':
14         if (isset($_SESSION["users_id"])){
15             $req=$db->prepare("SELECT * from orders where users_id = ?");
16             $req->execute([$_SESSION["users_id"]]);
17             $order=$req->fetch(PDO::FETCH_ASSOC);
18             echo json_encode(['sucess'=>true,'order'=>$order]);
19         }else{
20             echo json_encode(['sucess'=>false,'msg'=>"pas de commande avec cette id"]);
21         }
22     }
23
24     break;
25

```

- La seconde case de ce switch, permettra de recuperer les informations d'une commande suivant son numéro. Pour cela, on débute par la vérification que le numéro de commande soit bien renseigné (non-vide). Si c'est le cas, alors je fais appel à une requête SQL qui va chercher ce numéro de commande dans la base de données. Deux finalités sont alors possibles :
 - o Soit le numéro existe et alors les données vont être triées (même façon que précédemment) et afficher.
 - o Sinon le message d'erreur suivant s'affiche : "pas de commande avec cet id".

```

25
26     case 'select_num':
27         if (isset($_POST["number_order"])){
28             $req=$db->prepare("SELECT * from orders where number_order = ?");
29             $req->execute([$_POST["number_order"]]);
30             $order=$req->fetch(PDO::FETCH_ASSOC);
31             echo json_encode(['sucess'=>true,'order'=>$order]);
32         }else{
33             echo json_encode(['sucess'=>false,'msg'=>"pas de commande avec cette id"]);
34         }
35
36     break;

```

- La dernière case est dans le cadre du passage d'une nouvelle commande et de son insertion en base de données. Pour que celle-ci est lieu, on vérifie quatre choses :
 - o État de la commande
 - o Sa date
 - o l'users_id récupéré avec la superglobale SESSION
 - o l'id_sneakers récupéré avec la même superglobale.

Si cette vérification passe, alors on aura une requête SQL préparée qui fera l'insertion de ces données en base de données. Deux messages peuvent être affichées à la fin de la lecture de cette option du switch : commande ajoutée en cas de réussite ou sinon champs non-renseigné.

```

37
38 // insertion
39 case 'insert':
40     //print_r($_POST);
41     //print_r($_SESSION);
42     if(isset($_POST["etat_c"],$_POST["date_de_commande"],$_SESSION["users_id"],$_POST["id_sneakers"]) && !empty(trim($_POST["etat_c"])) && !empty(trim($_POST
43     ["date_de_commande"])) && !empty(trim($_SESSION["users_id"])) && !empty(trim($_POST["id_sneakers"]))) {
44         $req=$db->prepare("INSERT INTO orders (etat_c,date_de_commande,users_id,id_sneakers) VALUES (:etat_c,:date_de_commande,:users_id,:id_sneakers)");
45         $req->bindValue(":etat_c",$_POST["etat_c"]);
46         $req->bindValue(":date_de_commande",$_POST["date_de_commande"]);
47         $req->bindValue(":users_id",$_SESSION["users_id"]);
48         $req->bindValue(":id_sneakers",$_POST["id_sneakers"]);
49         $req->execute();
50         echo json_encode(["sucess"=>true,"msg"=>"commande ajouter"]);
51     }else{
52         echo json_encode(["sucess"=>false,"msg"=>"champs non renseigner"]);
53     }
54     break;
55
56 default:
57     echo json_encode(["sucess"=>false, "msg"=>"demande inconnu"]);
58

```

1.8) update_newsletter :

Cette fonctionnalité php a pour objectif de permettre à une personne de s'abonner à la newsletter.

```

76
77 case 'update_newsletter':
78     if(isset($_SESSION["users_id"]) && !empty(trim($_SESSION["users_id"]))){
79         $req=$db->prepare("UPDATE users SET newsletter=:newsletter WHERE users_id = :users_id");
80
81         $req->bindValue(":newsletter", $_GET["newsletter"]);
82         $req->bindValue(":users_id" , $_SESSION["users_id"]);
83         $req->execute();
84         echo json_encode(([{"success"=>true}]);
85
86     }else{
87         echo json_encode(([{"success"=>false, "error"=>"error"}]));
88     }
89     break;
90

```

Cette option du switch vérifie que le users_id est défini et sans espace inutile avec la superglobale users_id. Si cette condition est respectée alors je fais une requête sql qui UPDATE la donnée de la newsletter en base de données puis enfin on exécute cette requête après avoir utilisé la méthode bindValue(). Si ceci ne fonctionne pas on renvoi une valeur "error".

1.9) Admin.php :

Cette sous partie, se compose de trois fichier php permettant la gestion du site par les administrateurs ils sont :

- users_admin.php

- sneakers_admin.php

-order_admin.php

Ce fichier se compose du CRED (create, read, edit, delete). Tout d'abord le fichier users pour la gestion par les administrateurs. J'appelle deux fichiers function.php et db_connect.php. Ensuite je vérifie la superglobal soit GET ou POST. Ensuite, je code un switch qui va avoir comme choix :

- select : sélection de toutes les données de la base de données.

- select_id : sélection de toutes les données de la base de données selon un id récupérer en POST.

- delete_id : possibilité de supprimer un compte utilisateur en cas de conflit avec les règles de la plateforme.

Les deux premiers choix sont assez identiques. Pour le choix "select", j'effectue une requête "query" qui va récupérer toutes les données utilisateurs à part le mot de passe, newsletter et admin. Le mot de passe ne figure pas dans les données récupérées car on n'en a pas besoin dans le fait de vouloir supprimer un utilisateur ou afficher tous les utilisateurs.

Ensuite pour ce qui est de la newsletter ou du fait que si un utilisateur est admis ceci n'a aucun intérêt ce qui nous importe sont les informations générales des utilisateurs (email, users_name, users_id, lastname, firstname). Puis elles sont triées. En cas de succès, j'affiche dans la partie network de la console les données utilisateurs dans la variable users.

Pour ce qui est du select_id, il y a deux différences que l'on peut prendre en compte qui sont le users_id qui permet une sélection selon un users_id et donc par conséquent son ajout en condition de la requête préparée.

```

case "select":
    $req=$db->query("SELECT users_id, email, lastname, firstname, user_name FROM users");
    // $req->execute();
    $users=$req->fetch(PDO::FETCH_ASSOC);
    echo json_encode(["success"=> true, "user"=>$users]);
break;

case "select_id":
    if(isset($_SESSION["users_id"]) && !empty(trim($_SESSION["users_id"]))){
        $req=$db->prepare("SELECT * FROM users WHERE users_id=?");
        $req->execute([$SESSION["users_id"]]);
        $user=$req->fetch(PDO::FETCH_ASSOC);
        echo json_encode(["success"=> true, "user"=>$user]);
    }
break;

```

Le troisième choix possible est celui du delete_id : suppression d'un utilisateur suivant son users_id. Pour cela on vérifie d'abord que le users_id est bien récupéré et sans espace inutile. Puis ensuite je mets en place une requête préparée avec comme condition le users_id pour éviter une suppression de toutes les données utilisateurs. Puis j'exécute la requête en précisant par qu'elle superglobale je récupère le users_id.

```

case 'delete_id':
    if(!isset($_POST["users_id"])) && !empty(trim($_POST["users_id"])){
        $req=$db->prepare("DELETE users WHERE users_id=?");
        $req->execute($_POST["users_id"]);
        echo json_encode(["sucess" => true, "msg" => "votre compte a été supprimer malheureusement ! "]);
    }
break;

```

Pour ceux qui est des pages sneakers et order elles se composent de façon similaire. La seule différence reste dans les données qui sont récupérées.

Donc voici les cases du switch de la page admin_sneakers.php :

```

11
12     case 'select':
13         $req = $db->query("SELECT s.* FROM sneakers");
14         $articles = $req->fetchAll(PDO::FETCH_ASSOC);
15         echo json_encode(["success" => true, "articles" => $articles]);
16         break;
17
18     case 'select_id':
19         if (isset($_SESSION["users_id"])){
20
21             $req=$db->prepare("SELECT * FROM sneakers WHERE users_id=?");
22             $req->execute([$_SESSION["users_id"]]);
23             $sneakers=$req->fetch(PDO::FETCH_ASSOC);
24             echo json_encode(["success"=>true,"sneakers"=>$sneakers]);
25
26         } else{
27             echo json_encode(["success"=>false,"error"=>"erreur données manquantes"]);
28         }
29
30         break;

```

```

73
74     case 'delete_id':
75         if (isset($_POST["users_id"], $_POST["sneakers_id"]) && !empty(trim($_POST["users_id"])) && !empty(trim($_POST["sneakers_id"]))){
76             $req=$db->prepare("DELETE FROM sneakers WHERE users_id=? AND sneakers_id = ?");
77             $req->execute([$_POST["users_id"],$_POST["sneakers_id"]]);
78             echo json_encode(([{"success"=>true}]));
79
80         }else{
81             echo json_encode(([{"success"=>false, "error"=>"erreur de suppression"}]));
82         }
83
84         break;

```

Enfin les cases du switch de la page order_admin.php :

```
11 // - selection de commande : permet de voir toutes les commandes en cours
12 case 'select':
13     $req = $db->query("SELECT * from orders");
14     $order = $req->fetchAll(PDO::FETCH_ASSOC);
15     echo json_encode(["sucess"=>true, "orders"=> $order]);
16
17 break;
18
19
20 // - selection de commande par id : permet de voir les commande selon l'id client
21 // - vérification de l'id user ou l'id sneakers
22 case 'select_id':
23     if (isset($_SESSION["users_id"])){
24         $req=$db->prepare("SELECT * from orders where users_id = ?");
25         $req->execute([$SESSION["users_id"]]);
26         $order=$req->fetch(PDO::FETCH_ASSOC);
27         echo json_encode(['sucess'=>true,'order'=>$order]);
28     }else{
29         echo json_encode(['sucess'=>false,'msg'=>"pas de commande avec cette id"]);
30     }
31
32 break;
33
```

La seule particularité pour cette dernière case du switch : delete_number_order c'est que l'on supprime une commande selon un numéro de commande. Ce choix a été fait car il permet d'assurer qu'il n'y est qu'une commande qui soit supprimer car le numéro est unique.

```
67
68 case 'delete_number_order':
69     if(isset($_POST["number_order"]) && !empty(trim($_POST["number_order"]))){
70         $req=$db->prepare("DELETE FROM orders WHERE number_order = ?");
71         $req->execute[$_POST["number_order"]];
72         echo json_encode(["sucess"=>true, "msg"=>"commande supprimer"]);
73     }else{
74         echo json_encode(["sucess"=>true, "msg"=>"commande n'a pas été supprimer"]);
75     }
76
77 break;
```

2) La partie Front :

J'ai ensuite codé la partie la partie front de mon site. J'ai utilisé deux langage HTML et CSS. L'ensemble du front est responsive. Toutes les pages commencent par une partie head qui est composée de lien vers les pages CSS, bibliothèque Jquery ou Bootstrap.

2.1) register.html :

Cette page a pour objectif de permettre l'inscription de l'utilisateur. Celle-ci se compose d'un header, d'un formulaire et d'un footer.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>World Wide Sneakers - Page d'Inscription</title>
6      <link rel="stylesheet" href="register.css">
7      <link rel="stylesheet" href="../global_style.css">
8      <script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXy0G+800xUf+/Im1MZjXxxg0cBQBXU=" crossorigin="anonymous"></script>
9      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Naao0YzIztcQlfWSpd3yD65VohpuCoMLASjC" crossorigin="anonymous">
10     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBi4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+014" crossorigin="anonymous"></script>
11
12 </head>
13
14 <body>
15     <header>
16
17         
18
19         <div class="container">
20             <div class="row justify-content-center">
21                 <div class="col-12 w-100 col-lg-4 col-md-5 bordure">
22                     <!-- <div class="row justify-content-center"> -->
23                     <!-- <div class="row-xl justify-content-center"> -->
24                     <h1>World Wide Sneakers</h1>
25                     <h2> Buy & Sell Your Sneakers </h2>
26                 </div>/.col-12.w-100.col-lg-4.col-md-5.bordure
27
28
29             </div>/.row.justify-content-center
30         </div>/.container
31     </!-- </div> -->
32
33 </header>
```

Le header regroupe différentes balises dont notamment :

- Une balise img
- Des divisions représentées par la balise div
- Une balise h1 et h2

La première balise évoquée, est une balise image qui contient le logo sauvegardé en local avec l'utilisation du src qui permet de connaître la source de l'image.

Puis dans un ensemble successif de division, je mets mes deux balises h1 et h2. Ces balises permettent de décider de la taille d'un texte, h1 étant pour un titre et h2 pour le slogan du site.

Ces trois éléments sont centrés. Le premier élément utilise des propriétés css : margin.

Pour ce qui est du h1 et h2, ils sont au sein de trois divisions qui ont chacune des objectifs bien précis :

- La première défini un container
- La seconde division permet de définir sur qu'elle row (ligne) on veut se placer. Sur cette division, j'ai appliqué un justify-content-center qui permet de centrer du contenu.
- La troisième div, qui définit la colonne avec l'utilisation de col. Je précise la colonne en fonction de la taille de l'écran et aussi la width (largeur) pour que le texte soit bien centré.

```
35 <div class="background_color" class="marge">
36
37     <h3> Inscription </h3>
38
39
40     <!-- Formulaire d'inscription -->
41     <form method = "post">
42
43         <div class = "form_saisi" class="style_form">
44             <label for="firstname">Prénom: </label>
45             <input class="style_form" required type="text" class="size_block_nickname" id="firstname" name="user_name">
46         </div>/.style_form
47
48         <div class = "form_saisi" class="style_form">
49             <label for="lastname">Nom: </label>
50             <input class="style_form" type="text" class="size_block_name" id="lastname" name="user_name">
51         </div>/.style_form
52
53         <div class = "form_saisi" class="style_form">
54             <label for="Username">nom d'utilisateur : </label>
55             <input class="style_form" type="text" class="size_block_username" id="username" name="user_name">
56         </div>/.style_form
57
58
59         <div class = "form_saisi" >
60             <label for="email">e-mail : </label>
61             <input class="style_form" type="text" id="email" name="user_name">
62         </div>/.form_saisi
63
64         <div class = "form_saisi">
65             <label for="pwd">Mot de passe : </label>
66             <input class="style_form" type="password" class="size_block_mdp" id="pwd" name="user_name">
67         </div>/.form_saisi
68
69         <div class = "form_saisi" class="marge_save">
70             <input id="sauvegarde" type="submit" value="Inscrivez-vous">
71         </div>/.marge_save
72
73     </form>
74
75     <p>
76         <!-- Lien de redirection vers la page de connexion -->
77         <a href="#">Se connecter</a>
78     </p>
79 </div>/.marge
80
81     <script src="register.js"></script>
```

Maintenant, intéressons-nous au formulaire qui se trouve en dessous du header. Ce formulaire débute par une division (balise div), qui contient une balise h3, qui permet de mettre en avant le texte par rapport au reste du formulaire.

Pour le délimité, on utilise une balise form. A l'intérieur de cette balise on trouve un ensemble de div pour chacune des parties du formulaire : prénom, nom, nom d'utilisateur, email et mot de passe.

L'ensemble de ces div, regroupe deux balises :

- Une balise label qui a pour objectif d'afficher le texte / nom du champ. Une balise input qui est la zone de saisi pour les informations de l'utilisateur. Cette dernière balise évoquée, comprend plusieurs attributs :
 - o Un type (ici text)
 - o Un id
 - o Un name qui permet de l'identifier
 - o Une ou plusieurs classes

On finit ce formulaire avec le bouton de soumission contenu dans une div et qui n'a pas de balise label mais seulement une balise input qui possède trois attributs :

- o Un id
- o Une valeur
- o Un type qui n'est pas text mais submit

A l'extérieur du formulaire, on trouve une balise qui permet de lier du texte a un lien avec l'attribut : href.

Ensuite, on trouve une balise script qui permet de faire l'appel vers la page Javascript. Enfin, on termine par le footer qui englobe une seule balise "p" qui contient le texte contre le copyright.

Cette page a un block central qui regroupe trois éléments :

- La balise h3
- Le formulaire
- Le lien de redirection.

Celui-ci est défini par des propriétés css, comme margin qui met de la marge vers l'intérieur de la page, background-color qui permet d'avoir la couleur de fond. Ensuite la page login, qui a une composition similaire à la page précédemment évoquée.

2.2) Login.html :

Cette page a pour objectif de permettre à l'utilisateur de se connecter à son compte. Le code HTML de la page connexion se compose d'un header, du formulaire de connexion et enfin du footer.

Le header comprend le logo, le nom et le slogan du site. Puis s'en suit le formulaire de connexion composé de deux champs à remplir par l'utilisateur pour son authentification : email/users_name et son mot de passe. En bas du formulaire j'ai codé un lien de redirection vers la page inscription. Il y a aussi le bouton de soumission du formulaire. En dessous du formulaire se trouve le footer dans la balise du même nom.

```
</header>
<body>
    <header>
        

        <div class="container">
            <div class="row xl-mb-5-ml-2 justify-content-center">
                <div class="col-xl-5 col-lg-6 col-md-8 bordure">
                    <!-- <div class="row justify-content-center"> -->
                    <div class="row-xl justify-content-center">
                        <h1>World Wide Sneakers</h1>

                    </div>/.row-xl.justify-content-center
                    <div class="col-xl-ml-4-ml-3" id="div_h2">
                        <h2> Buy & Sell Your Sneakers </h2>
                    </div>/#div_h2.col-xl-ml-4-ml-3

                </div>/.col-xl-5.col-lg-6.col-md-8.bordure
            </div>/.row.xl-mb-5-ml-2.justify-content-center
        </div>/.container

    </header>
```

```

35
36 <div class="container fluid">
37   <div class="row justify-content-center">
38
39     <div class="col-xl-4 col-lg-6 col-md-8">
40
41       <div class="background_color">
42
43         <h3 class="title_h3"> Connexion </h3>
44
45         <!-- Formulaire de connexion -->
46         <form action="login.php" method="post">
47
48           <div class = "form_saisi">
49             <label for="name">nom d'utilisateur/email:</label>
50             <input class="style_form" type="text" id="username_email" name="id">
51           </div>/.form_saisi
52
53           <div class = "form_saisi">
54             <label for="name">Mot de passe : </label>
55             <input class="style_form" type="password" id="pwd" name="user_name">
56           </div>/.form_saisi
57
58           <div class = "form_saisi">
59             <input id="sauvegarde" type="submit" value="Connectez-vous">
60           </div>/.form_saisi
61
62           <!-- Lien de redirection vers le site d'inscription -->
63         </form>
64
65         <p>
66           <a href="../register/register.html">Inscrivez vous</a>
67         </p>
68
69         <script src="login.js"></script>
70         <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFY1zLA8Nl+NtUV0sA7MsXsP1llyJ0Mp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
71
72       </div>/.background_color
73
74     </div>/.col-xl-4.col-lg-6.col-md-8
75
76   </div>/.row.justify-content-center
77 </div>/.container.fluid
78
79 <footer>
80   <p>&copy;W&S</p>
81 </footer>
82 </body>
83 </html>

```

2.2) create_product.html :

L'objectif de cette page est de permettre à l'utilisateur de vendre sa paire. Elle se compose d'un header d'un formulaire et d'un footer.

Cette page a comme particularité par rapport aux deux dernières : l'utilisation d'une balise hr qui permet de faire une ligne horizontale entre le header et le formulaire. La différence majeure est sur la couleur de fond du formulaire qui est en blanc.

```

create_product.html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <link rel="stylesheet" href="create_product.css">
6     <link rel="stylesheet" href="../global_style.css">
7     <title>Page création produit</title>
8     <script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upPUjgXYG+800xUf+TmIMZJXxg0CBo8XU=" crossorigin="anonymous"></script>
9     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azrP1zp1Q+8aQXQ6B8JzIyQD9BjQj0oDh1LmQzZJl1ztcQTwfspd3yD65VohpuuCOMLASjC" crossorigin="anonymous">
10    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenUIFdBe4zVF0s0G1M5b4hcpxy09F7jl+jjXkk+Q2h455rYXK/7HauoJl+014" crossorigin="anonymous"></script>
11
12 </head>
13 <body>
14
15
16   <header>
17
18     <div class="header">
19       
20       <div class="container">
21         <div class="row justify-content-center">
22           <div class="col-xl-4 w-100 col-lg-8 col-md-5 bordure">
23             <h1>World Wide Sneakers</h1>
24             <h2>Buy & Sell Your Sneakers </h2>
25
26             </div>/.col-xl-4.w-100.col-lg-8.col-md-5.bordure
27           </div>/.row.justify-content-center
28         </div>/.container
29
30       </div>/.header
31
32   <hr>
33
34

```

```

34
35     <div class="container">
36         <div class="row justify-content-center">
37             <div class="col-lg-3 col-md-5">
38
39                 <div class="block">
40
41                     <form action="../sneakers.php" method="post">
42                         <!-- Formulaire de création d'un produit sur le site et de son ajout --&gt;
43                         <!-- Ajouter le rectangle d'ajout d'image avec prévisualisation de l'image--&gt;
44
45                         &lt;div class="title_form"&gt;
46                             &lt;h2&gt;Création produit&lt;/h2&gt;
47                         &lt;/div&gt;/.title_form
48
49                         &lt;div id="marge_form"&gt;
50                             &lt;label class="marge"&gt;Nom du modèle:&lt;/label&gt;
51                             &lt;input type="text" id="brand" class="size_input_brand" name="name"&gt;
52                         &lt;/div&gt;/#marge_form
53
54                         &lt;div id="marge_form"&gt;
55                             &lt;label class="marge"&gt;Taille:&lt;/label&gt;
56                             &lt;input class="size_input_size" id="size" type="text" name="size"&gt;
57                         &lt;/div&gt;/#marge_form
58
59                         &lt;div id="marge_form"&gt;
60                             &lt;label&gt;Couleurs:&lt;/label&gt;
61                             &lt;input class="size_input_color" id="color" type="text" name="color"&gt;
62                         &lt;/div&gt;/#marge_form
63
64                         &lt;div id="marge_form"&gt;
65                             &lt;label&gt;Etat:&lt;/label&gt;
66                             &lt;input class="size_input_state" id="state" type="text" name="state"&gt;
67                         &lt;/div&gt;/#marge_form
68
69                         &lt;div id="marge_form"&gt;
70                             &lt;label&gt;stock:&lt;/label&gt;
71                             &lt;input type="text" class="size_input_stock" id="stock" name="stock"&gt;
72                         &lt;/div&gt;/#marge_form
73
74                         &lt;div id="marge_form"&gt;
75                             &lt;label&gt;prix:&lt;/label&gt;
76                             &lt;input type="text" class="size_input_price" id="price" name="price"&gt;
77                         &lt;/div&gt;/#marge_form
78
79                         &lt;div id="marge_form"&gt;
80                             &lt;label for="image_produit"&gt;Image&lt;/label&gt;
81                             &lt;input type="file" id="image" name="image_produit"&gt;
82                         &lt;/div&gt;/#marge_form
83
84                         &lt;div id="marge_form"&gt;
85                             &lt;input id="sauvegarde" type="submit" value="Met en ligne ton article"&gt;
86                         &lt;/div&gt;/#marge_form
87
88                 &lt;/form&gt;
89
90             &lt;/div&gt;/.block
91
</pre>

```

```

97
98     <script src="create_product.js"></script>
99
100    <footer>
101        <p>&copy;WWS</p>
102    </footer>
103
104    </body>
105

```

2.3) FAQ.html :

Cette page a pour objectif de permettre de répondre aux questions de l'utilisateur et qu'il puisse s'abonner à la newsletter.

Elle se compose d'un header composé du logo, du nom du site et du slogan du site. A l'exterieur on trouve un ensemble de cases dynamiques répondant aux questions les plus fréquentes.

Pour ceci, j'ai utilisé trois balises details. Dans chacune d'elle, j'ai mis une balise summary qui contient la question et une balise p pour la réponse.

En bas de la page, on trouve une zone de texte où l'utilisateur peut poser sa question si elle ne fait pas partie des questions déjà présentent dans les cases au-dessus. En dessous de celle-ci, on trouve deux boutons : un pour s'abonner à la newsletter et un pour l'envoi de la question.

On retrouve tout en bas de la page le footer qui est similaire à celui des autres pages.

```
1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="UTF-8">
5     <title>FAQ World Wide Sneakers</title>
6     <link rel="stylesheet" href="FAQ.css">
7     <link rel="stylesheet" href="../global_style.css">
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3Q0gpJLIm9Nao0Yz1ztCQTwsPd3yD65VohppuC0mLASjC" crossorigin="anonymous">
9     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenfIdbI42VF0s0G1M5b4hpcxyO9F7Jl+jjXkk+Q2h455rYXK/7HAuoJl+0I4" crossorigin="anonymous"></script>
10    <script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgHXY0G+800xUf+/Im1MZjXxxg0cBQ8XU=" crossorigin="anonymous"></script>
11  </head>
12
13  <body>
14    <header>
15      <div class="container-fluid">
16        <div class="row-justify-content-right d-flex">
17          <div class="col">
18            
19            <h1>World Wide Sneakers</h1>
20            <h2>Buy & Sell Your Sneakers</h2>
21            </div>/.col
22        </div>/.row-justify-content-right.d-flex
23      </div>/.container-fluid
24    </header>
```

```

26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
```

```

<div>
  <h3>Foire Aux Questions</h3>

  <details>
    <summary>Quelle sont les délais de livraison?</summary>
    <p>Le temps de livraison est prévu entre 5 jours et 2 semaines.</p>
  </details>

  <details>
    <summary>Comment sont authentifier les produits?</summary>
    <p>Les produits sont autentifiés par notre équipe d'expert.</p>
  </details>

  <details>
    <summary>Peut-on vendre des bootlegs sur ce site?</summary>
    <p>La vente de bootlegs sur le site est interdite comme tout produit contrefait.</p>
  </details>
</div>

<form>
  <div>
    <textarea id="question" name="question">Pose ta question</textarea>
  </div>
  <input type="submit" id="question" value="Envoyez la question" name="question">
  <input type="submit" id="newsletter" value="Abonnez-vous à la newsletter" name="Newsletter">
</form>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8NI+RtUVF0sA7MsXsP1UYJ0Mp4YLEuNSFAP+3cXn/tWtTaxVXM" crossorigin="anonymous"></script>
<script src="FAQ.js"></script>

<footer>
  <p>&copy;WS</p>
</footer>

</body>
</html>
```

2.4) catalogue.html

Cette page a pour objectif que l'utilisateur est une vue globale de l'ensemble des paires disponibles sur le site ou qu'il recherche la paire de sneakers selon son nom.

Le code de la page html du catalogue, se compose d'un header avec deux balises : h1 et h2 qui sont misés dans plusieurs div avec des class qui rende le header responsive. Ensuite, j'ai fait une navbar qui permet une recherche pour l'utilisateur.

Pour cela j'utilise des fonctionnalités bootstrap. Cette navbar se compose d'une balise nav dans laquelle on a une div puis une balise form qui contient une balise input (la barre de recherche) et une balise button. Cette dernière balise permettra à l'utilisateur d'effectuer une nouvelle recherche. A l'extérieur un bouton permettra d'afficher toutes les sneakers.

Puis on a le footer similaire aux autres pages.

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5     <title>Catalogue Produit</title>
6     <link rel="stylesheet" href="page_catalogue.css">
7     <link rel="stylesheet" href="../global_style.css">
8     <script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXY0G+800xUf+/Im1MZjXxxgOcBQBXU=" crossorigin="anonymous"></script>
9     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWFspd3yD65Voh"
10    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenUIKFdBi4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYKK/7HAuoJl+0I
11    <!-- <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script> -->
12 </head>
13
14 <body>
15     <header>
16         <div class="header">
17             
18             <div class="container">
19                 <div class="row justify-content-center">
20                     <div class="col-xl-4 w-100 col-lg-8 col-md-5 bordure">
21                         <h1>World Wide Sneakers</h1>
22                         <h2> Buy & Sell Your Sneakers </h2>
23
24                     </div>/.col-xl-4.w-100.col-lg-8.col-md-5.bordure
25                 </div>/.row.justify-content-center
26             </div>/.container
27         </div>/.header
28     </header>
29
30
31     <!-- Barre de recherche avec BoodStrap ou sans -->
32     <nav class="navbar-body">
33         <div class="container-fluid">
34             <form method="GET" class="d-flex" id="navbar" role="nav-bar-search">
35
36
37                 <input class="form-control me-2" type="search" id="nav_bar" name="nav_bar" placeholder="Pour rechercher écrivez et appuyez sur entrée" aria-label="Search">
38                 <button id="new_search">Nouvelle recherche</button>
39
40             <div id="result_search">
41
42             </div>#result_search
43
44
45         </form>/#navbar.d-flex
46     </div>/.container-fluid
47 </nav>/.navbar-body
48
49     <button id="buttonProduct">Voir les sneakers</button>
50
51
52     <div id="article">
53         <div id="sneakers_id"></div>
54     </div>/#article
55
56     <div id="form_update"></div>
57
58
59     <script src="page_catalogue1.js"></script>
60
61
62     <footer>
63         <p>&copy WWS</p>
64     </footer>
65
66
67     </body>
68
69 </html>
70
71

```

2.5) admin.html :

Les pages HTML admin, ont pour objectif la gestion des utilisateurs, sneakers et commande par les administrateurs. Les trois pages sont similaires au niveau du code html mais les données diffèrent :

- Users

- Sneakers
- Order

Au sein du header, on trouve une balise h1.

Puis l'élément principale est un tableau qui se compose des différentes balises suivantes : <table> puis dedans une balise <tr> qui représente une cellule d'une ligne du tableau et à l'intérieur des balises <th> qui définis une cellule comme une cellule d'en tête pour un groupe de cellule.

```

1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta charset="utf-8">
5      <title>Tableau Utilisateur</title>
6      <link rel="stylesheet" href="../../global_style.css">
7      <link rel="stylesheet" href="admin_style.css">
8      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
9      <script src="https://code.jquery.com/jquery-3.6.3.min.js"
10         integrity="sha256-pvW+upLUpjgbXY0G+800xUF+/Im1MZjXxxg0cBQ8XU=" crossorigin="anonymous"></script>
11     <script src="../../auth.js"></script>
12 </head>
13
14 <body>
15     <script src="users.js"></script>
16
17     <header>
18         <h1>Tableau de gestion des utilisateurs</h1>
19     </header>
20
21
22     <table>
23         <tr>
24             <th class="th_firstname">Nom</th>
25             <th class="th_lastname">Prénom</th>
26             <th class="th_user_name">Username</th>
27             <th class="th_email">Email</th>
28             <!-- <th class="mdp">Mot de passe</th> -->
29             <th class="newsletter">Newsletter</th>
30             <th class="th_modify">Modifier</th>
31             <th class="th_delete">Supprimer</th>
32         </tr>
33
34         <tr class="t-body"></tr>
35     </tr>
36
37     </table>
38
39     <script src="users.js"></script>
40
41     <footer>
42         <p>&copy;WWS</p>
43     </footer>
44 </body>
45 </html>
```

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8">
5     <title>Tableau Utilisateur</title>
6     <link rel="stylesheet" href="../../global_style.css">
7     <link rel="stylesheet" href="../user/admin_style.css">
8     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
9     <script src="https://code.jquery.com/jquery-3.6.3.min.js"
10        integrity="sha256-pvPw+upLPUjgMXY0G+800xUf+/Im1MZjXXg0cBQBXU=" crossorigin="anonymous"></script>
11     <script src="../../auth.js"></script>
12 </head>
13
14 <body>
15     <header>
16         <h1>Tableau de gestion des Sneakers disponible sur le site</h1>
17     </header>
18
19     <table>
20         <tr>
21             <th class="th_user_name">Brand</th>
22             <th class="th_firstname">Size</th>
23             <th class="th_lastname">Color</th>
24             <th class="th_email">States</th>
25             <th class="th_price">Price</th>
26             <th class="th_modify">Supprimer</th>
27         </tr>
28
29         <tr class="t-body"></tr>
30     </tr>
31
32 </table>
33

```

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8">
5     <title>Tableau Utilisateur</title>
6     <link rel="stylesheet" href="../../global_style.css">
7     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
8     <script src="https://code.jquery.com/jquery-3.6.3.min.js"
9        integrity="sha256-pvPw+upLPUjgMXY0G+800xUf+/Im1MZjXXg0cBQBXU=" crossorigin="anonymous"></script>
10    <script src="../../auth.js"></script>
11 </head>
12
13 <body>
14     <header>
15         <h1>Tableau de gestion des Commandes</h1>
16     </header>
17
18     <table>
19         <tr>
20             <th class="th_user_name">Numéro de commande</th>
21             <th class="th_firstname">Id utilisateur</th>
22             <th class="th_lastname">Id sneakers</th>
23             <th class="th_email">Nom</th>
24             <th class="th_price">État de la commande</th>
25             <th class="th_modify">date de commande</th>
26             <th class="th_modify">Supprimer</th>
27         </tr>
28
29         <tr class="t-body"></tr>
30     </tr>
31
32 </table>
33
34     <script src="users.js"></script>
35
36     <footer>
37         <p>&copy;W5</p>
38     </footer>
39
40 </body>
41 </html>

```

2.6) account.html :

La page account a pour objectif de permettre à l'utilisateur de voir toute ses infos personnelles à part son mot de passe pour des raisons de sécurité. On trouve aussi les infos sur les produits dans le panier.

Pour ce qui est du header, à gauche on trouve le logo qui est mis sur la page grâce à l'utilisation de la balise img. Au centre, on a le nom du site et son slogan.

La page se compose ensuite de deux blocs blancs l'un à côté de l'autre. Ceux-ci sont codé grâce à des propriétés CSS. Celui de gauche contient les informations personnel de l'utilisateur.

Celui de droite, contient le panier de l'utilisateur.

```
1 <!DOCTYPE html>
2 <html lang="fr">
3
4     <head>
5         <meta charset="UTF-8">
6         <title>Account</title>
7         <link rel="stylesheet" href="../global_style.css">
8         <link rel="stylesheet" href="account.css">
9         <script src="https://code.jquery.com/jquery-3.6.3.min.js"
10            integrity="sha256-pvPw+upLPUjgMY0G+800xUf+/Im1MZjXXg0c8QBXu=" crossorigin="anonymous"></script>
11        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3Q0gpJLIm9Naod0Yz1ztTcQtWfspd3yD65VohppuOCmLASjC" crossorigin="anonymous">
12        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenUlKFD8Ie4zVF0s0GIM5b4hcpxyD9F7jl+jXkk+Q2h455rYXK/7HAuoJl+0I4" crossorigin="anonymous"></script>
13
14     </head>
15
16     <body>
17         <header>
18             <div class="header">
19                 
20                 <div class="container">
21                     <div class="row justify-content-center">
22                         <div class="col-11-4 w-100 col-lg-8 col-md-5 bordure">
23                             <h1>World Wide Sneakers</h1>
24                             <h2>Buy & Sell Your Sneakers </h2>
25                         </div> .col-11-4.w-100.col-lg-8.col-md-5.bordure
26                     </div> .row.justify-content-center
27                 </div> .container
28             </div> .header
29         </header>
30
31
32         <div id="block_text_img">
33             <div class="merge_div">
34                 <div class="block" id="text">
35                     <div class="container">
36                         <div class="row">
37                             <div class="col">
38                                 <h3>My Account</h3>
39                                 <div class="title_part">
40                                     <h4>Information Personnel</h4>
41                                     </div> .title_part
42                                 </div> .col
43                             </div> .row
44                         </div> .container
45
46             </div>
47
48         <div>
```

```

<div>
  <div id="marg">
    <label for="firstname">Prénom</label>
    <input class="size_input_firstname" type="text" id="username_firstname" name="id">
  </div> #marg

  <div id="marg">
    <label for="lastname">Nom</label>
    <input class="size_input_lastname" type="text" id="username_lastname" name="id">
  </div> #marg

  <div id="marg">
    <label for="email">Email</label>
    <input class="size_input_email" type="text" id="username_email" name="id">
  </div> #marg

  <div id="marg">
    <label for="Username">UserName</label>
    <input class="size_input_UserName" type="text" id="username" name="id">
  </div> #marg
</div>
</div> #text.block

</div> /.marge_div

<div id="block_text_img">
  <div>

    <div class="block2" id="text">
      <div class="container">
        <div class="row">
          <div class="col">
            <div class="title_part">
              <div>Commande</div>
            </div> .title_part
          </div> .col
        </div> .row
      </div> .container
      <div id="order"></div>
    </div> #text.block2
  </div>
</div> #block_text_img
</div> #block_text_img

<script src="account.js"></script>

<footer>
  <p>&copy;W5</p>
</footer>
</body>
</html>

```

2.7) welcome.html :

Cette page a comme objectif d'accueillir l'utilisateur après avoir recherché le site ou après sa connexion.

Cette page débute avec un header. A haut à gauche de la page on a le logo du site. Puis on trouve les balise h1 et h2 dans une division (balise div).

En haut à droite de la page on a un menu déroulant. Pour faire cet élément, j'utilise d'abord une balise button. Puis en dessous, on utilise une balise ul dans laquelle on inclut des balises li qui ont pour valeur des liens de redirection, balise a. Le header est séparer du reste de la page avec une balise hr.

Ensuite, on trouve une div avec dedans des balises <figure> qui permettent de faire apparaître une image avec en dessous un texte de façon alignée.

La balise figure se compose d'une balise a qui permet de rediriger l'utilisateur vers la page cible elle-même contenant l'image (balise img). Ceci permet qu'au clic sur l'image celle-ci redirige l'utilisateur.

La balise figcaption est utilisée pour l'affichage du texte. A l'extérieur de la div on a une balise script pour le lien vers la page js. On clôt cette page avec le footer similaire aux autres pages.

```
38 <hr>
39 <!-- Texte de présentation du site -->
40 <p>
41     L'entreprise World Wide Sneakers a été créée en juin 2022. La raison de sa création est la suivante : on désire une plateforme qui ne prenne pas de marge lors de la vente ou achat de sneakers. Cette idée est
42     bénéfique que ce soit pour le resealer ou bien pour l'acheteur.
43 </p>
44
45 <div class="image_categorie">
46
47     <figure>
48         <a href="../create_product/create_product.html">
49             
50         </a>
51         <figcaption>Vends ta paire</figcaption>
52     </figure>
53
54     <figure>
55         <a href="../page_catalogue/page_catalogue.html">
56             
57         </a>
58         <figcaption>Sneakers</figcaption>
59     </figure>
60
61     <figure>
62         <a href="../FAQ/FAQ.html">
63             
64         </a>
65         <figcaption>Question</figcaption>
66     </figure>
67
68 </div>/.image_categorie
69
70 <script src="welcome_page.js"></script>
71
72 <footer>
73     <p>©copyWWS</p>
74 </footer>
75
76 </body>
77 </html>
```

3) Javascript (Jquery) :

Pour la réalisation de mon projet, j'ai ensuite débuté la partie javascript pour ça j'utilise la bibliothèque Jquery. J'ai utilisé des appels ajax (Asynchronous Javascript) qui est un appel XMLHttpRequest vers une page PHP. Elle prend en paramètre un url vers la page php visée. Mais on précise aussi qu'elle superglobale on utilise pour recuperer les données. Puis le type avec le qu'elle on va récupérer les données. Enfin les data qu'on souhaite prendre en entrée de cette appel ajax.

3.1) register.js :

Pour commencer la page register (inscription) qui s'occupe de la fonctionnalité d'inscription, cette page comporte un appel ajax lors de la soumission du formulaire d'inscription.

Avant l'appel Ajax, j'évite la recharge automatique de la page comportement par default du formulaire.

L'appel Ajax est relié à la page php register.php, ensuite je précise avec quel superglobal sont récupéré les données puis les datatypes ici json et les data qui représente l'ensemble des données que l'on veut récupérer.

Pour cette page, on récupère les données suivantes : firstname, lastname, email, user_name et le password. Ensuite, si ceci est réussi alors on redirige l'utilisateur vers la page de connexion sinon on affiche dans la console : "error".

```

1 $("form").submit((event) => {
2   event.preventDefault();
3
4   $.ajax({
5     url:"register.php",
6     type: "POST",
7     dataType: "json",
8     data: {
9       firstname: $("#firstname").val(),
10      lastname: $("#lastname").val(),
11      user_name: $("#username").val(),
12      email: $("#email").val(),
13      pwd: $("#pwd").val()
14    },
15    success: (res) => {
16      console.log(res);
17      if (res.success) window.location.replace("../login_logout/login.html");
18      else console.log("error");
19      // alert(res.error);
20    }
21  });
22 });

```

3.2) login.js :

Je commence par créer une variable urlParams qui permet de récupérer un paramètre dans l'url.

Si l'url contient l'argument logout alors il y a l'exécution d'un appel ajax qui dans ce cas-là vide les données utilisateur (user_id, admin, newsletter) du localStorage.

Ensuite, lorsque le formulaire est soumis, alors on a un appel ajax avant cela on évite le fonctionnement par défaut du formulaire. Pour ce qui est de l'appel ajax, on le connecte à la page login.php, on précise avec quel superglobale on récupère les données ici en POST.

Puis, je précise le type avec lequel je récupère les données du formulaire ici type : json. Enfin on récupère les datas qui sont : email ou user_name et le mot de passe. Si cette dernière étape est un succès, alors j'insere dans la case user du localStorage les données utilisateur (users_id,admin et newsletter) puis je redirige l'utilisateur vers la page d'accueil.

```

1 const urlParams = new URLSearchParams(window.location.search);
2
3 if(urlParams.get("logout")){
4   $.ajax({
5     url: "logout.php",
6     type:"GET",
7     dataType: "json",
8     success: (res) => {
9       //if (res.success)
10       localStorage.removeItem("user");
11     }
12   });
13 }
14
15 $("form").submit((event) => {
16   event.preventDefault();
17   $.ajax({
18     url:"login.php",
19     type: "POST",
20     dataType:"json",
21     data: {
22       username_email: $("#username_email").val(),
23       pwd: $("#pwd").val(),
24     },
25     success: (res) => {
26       if (res.success) {
27         console.log(res);
28         console.log(res.user);
29         localStorage.setItem("user",JSON.stringify(res.user));
30         window.location.replace("http://localhost/projet_fin_annee/site/welcome_page/welcome_page.html");
31       }else{
32         alert(res.error);
33       }
34     }
35   });
36 });

```

3.3) create_product.js :

On débute comme pour la page login.js avec la récupération des données dans l'url.

Ensuite, j'ai codé la fonction insertSneakers qui prend en paramètre l'ensemble des données dont on a besoin de récupérer sur la sneakers (size, color, brand, states, price image et stock).

Cette fonction, part tout d'abord de la création d'une constante fd qui contient la création de la formdata, celle-ci récupère toutes les données dites plus tôt. Ensuite, on passe avec un appel ajax, on débute avec l'url vers la page sneakers.php, puis je précise avec quelle superglobale on récupère les données qui est : POST.

Les datas qui sont récupérées avec la constante fd puis au défini à false : contentType, processData et le cache.

A la soumission du formulaire, d'abord on empêche le fonctionnement de base du formulaire a la soumission.

On donne aux constantes les valeurs rentrées par l'utilisateur puis on fait un appel de la fonction créée au début de la page : insertSneakers() en y ajoutant l'ensemble des noms en paramètre de fonction.

Ceci permet d'envoyer et d'insérer les données en bdd. Puis à la soumission du formulaire on lui demande de vider toutes les cellules de texte rempli.

```

1 // Je récupère les paramètres de mon url
2 const ParamUrl = new URLSearchParams(window.location.search);
3
4 console.log("ok")
5 // On définit une variable qui servira à savoir si nous voulons insérer ou mettre à jour un article
6 let article_id = null
7
8 // fonction pour insérer une paire de Sneakers
9 function insertSneakers(size,color,brand,states,price,image,stock){
10    const fd = new FormData();
11    fd.append('opt','insert');
12    fd.append('size',size);
13    fd.append('color',color);
14    fd.append('brand',brand);
15    fd.append('states',states);
16    fd.append('price',price);
17    fd.append('image',image);
18    fd.append('stock',stock);
19
20    $.ajax({
21        url: "../sneakers.php",
22        type:"POST",
23        dataType:"json",
24        data:fd,
25        contentType: false,
26        processData:false,
27        cache:false,
28        success: (res)=>{
29            console.log(res);
30            console.log("ok");
31        }
32    })
33 }
34
35 $("form").submit((event)=>{
36     event.preventDefault(); // J'empêche le recharge automatique de la page lors de la soumission du formulaire
37     console.log("submit");
38     // Je récupère les valeurs rentrer dans le formulaire
39     const brand = $("#brand").val();
40     const color = $("#color").val();
41     const price = $("#price").val();
42     const states = $("#state").val();
43     const size = $("#size").val();
44     const stock = $("#stock").val();
45     const image = $('#image')[0].files[0];
46
47     insertSneakers(size,color,brand,states,price,image,stock);
48 })
49
50
51 $("form").submit(function() {
52     $("form")[0].reset();
53 });

```

3.4) page_catalogue.js :

Au début de cette page js je récupère dans une variable (let) nommée users je récupère les donnée (users_id, admin, newsletter) de l'utilisateur connectée dans le LocalStorage.

La première fonction de cette page, c'est celle de la suppression d'articles du catalogue. Bien sûr seul l'utilisateur qui a posté son offre produit peut la supprimer. La fonction supprimer_article prend en paramètre sneakers_id. Tout d'abord on vérifie dans cette fonction que l'utilisateur veux vraiment supprimer son offre produits avec l'utilisation de "confirm" qui fait apparaitre une boite de dialogue pour la confirmation de cette action. Si cette condition est remplie alors un appel ajax est exécuter celui-ci va se connecter avec le fichier sneakers.php récupérer les données dont il a besoin avec la superglobale POST.

C'est donnée sont le choix pour le switch de la page sneakers.php (ici : delete_id) et le sneakers_id qui a été récupérer au préalable en paramètre de la fonction. Si toute c'est condition sont respecté alors on va supprimer l'offre en bdd.

Mais aussi à sa suppression sur la page html avec l'utilisation du ".remove" sur l'id_sneakers qui est pris comme id de la div html de l'offre sur le site.

La fonction qui suit sur cette page est la fonction d'ajout d'un produit dans son panier par l'utilisateur. On veut que lorsque l'utilisateur click sur le bouton ajoutez au panier il puisse ajouter le produit au panier.

Pour cela au click du bouton on enregistre dans le LocalStorage les informations du produit suivante : id_sneakers et la quantité de produit que l'utilisateur souhaite commander. Avec l'utilisation d'une boucle foreach on va afficher ses données dans la page account.html. Si la quantité est égal à 0 alors on supprime l'article du panier. L'utilisateur via deux boutons différents peut soit augmenter ou diminuer la quantité de produit.

Si l'utilisateur a rempli son panier et alors il peut le valider. Ceci créera alors une commande en base de données. Pour ceci il faut qu'au click du bouton de validation du panier un appel ajax est déclencher vers la case insert de la page php order.php. Si cette appel est un succès alors envoi d'un mail à l'utilisateur pour lui confirmer que sa commande est bien validée.

La fonction qui suit est celle de l'enregistrement d'une modification d'un article enregistrer en base de données. Cette fonction prend en paramètre le sneakers_id. Ensuite je crée six constantes qui récupère les valeurs qui sont entrée dans le formulaire de modification. Puis un appel ajax, est alors exécuté vers la page sneakers.php. Pour ce faire, les données sont récupérées avec la SuperGlobale POST. Je défini le type des data qui seront récupéré : json. Les data récupérés sont l'opt (update_sneakers_id) du switch de la page sneakers.php et les valeurs de l'ensemble des constantes vu précédemment. En cas de succès, j'affiche dans la console le texte suivant : "success".

Si ces éléments peuvent être enregistrés, il faut bien une fonction qui permet d'afficher sur la page HTML le formulaire de modification. Cette fonction nommée modifier article qui prend en paramètre le sneakers_id de l'offre qui doit être modifiée. Le formulaire de modification, se compose d'une balise h3 qui est créé dans une constante title_formulaire avec comme texte "Modifier votre Offre sneakers". Ensuite, je crée une balise form avec comme méthode POST. Dans celle-ci on trouve une div par élément de l'offre qui peuvent être modifiée, chacune des div se compose d'un label et d'un input. L'input est la zone de modification pour l'utilisateur et le label qui permet l'affichage du nom de chacune des données présentent en input. Les données en input sont récupérées par l'appel ajax. Petite différence pour l'image, celle-ci est gérer via un bouton. A la fin j'ajoute un bouton de validation. Au moment où ce bouton est clicker par l'utilisateur, alors j'empêche le recharge de la page et ensuite j'appelle la fonction qui va récupérer les données du formulaire.

La page continue avec la fonction d'affichage des sneakers en HTML. Pour ceci, je récupère en paramètre les données de la base de données sneakers. Par la suite, je crée une première div qui comptera deux class style_div et container. Ensuite je lui attribue l'id div_ + le sneakers_id défini pour l'offre (différent pour toutes les offres). Ensuite dans une balise h3 je fais figurer le nom de la paire de sneakers mis en ligne. Puis dans une balise img j'ajoute l'image produit. On ajoute à ceci les infos produit dans des balises p (balise de texte). Puis je mets l'ensemble dans une div à laquelle j'attribue l'id de l'utilisateur qui a proposé cet article.

```

197 // affichage sneakers
198 function sneakers(data){
199   data.forEach(el=>{
200     // console.log(el.sneakers_id);
201
202     const div= $("<div></div>");
203     div.addClass("style_div");
204     div.addClass("container");
205     div.attr("id","div_"+el.sneakers_id);
206
207
208     // on récupère La brand de La n Ligne du tableau de donnée dans une balise h3
209     const brand_html = $("<h3></h3>").text(el.brand);
210
211
212     // div pour L'image + info produit
213     const div_image = $("<div></div>"); // Je récupère Le fichier du formulaire
214     div_image.addClass("style_produit")
215     const img= $("<img src='../logo.png'></img>")
216     // div_img_info_produit.append(img);
217
218     // let img; // Je déclare La variable img sans valeur;
219     // console.log(res.articles[i].image);
220     // if (res.articles[i].image) img = $("<img>").attr("src", "../"+res.articles[i].image); // Je crée une image et j'ajoute
221
222     div_image.append(img);
223
224     |
225     // div information produit
226     const div_info_produit = $("<div></div>");
227     div_info_produit.attr("id", "div_"+el.users_id);
228     div_info_produit.addClass("margin_info_product");
229

```

Ensuite je crée trois boutons qui seront tous dans une même balise div .ils ont la particularité d'être ajoutés selon des conditions : la première pour le bouton supprimé et modifier. On compare le users_id de l'utilisateur connecté avec celui sous lequel est enregistré l'offre de la paire de sneakers. Et aussi via la fonctionnalité qui s'active en cas de click de l'utilisateur (.click). Elle permet soit de faire un appel de fonction vers la fonction supprimer ou modifier suivant le bouton sur lequel clique l'utilisateur. Si les deux users_id ne sont pas les mêmes alors j'affiche un bouton d'ajout aux commandes avec aussi un appel de fonction en cas de click sur le bouton.

```

251 const div_button=$("<div></div>");
252
253
254
255 // Si users_id du LocalStorage == users_id de l'offre
256 if(users.users_id == el.users_id){
257
258     //bouton pour modifier une annonce
259     const supp_btn = $("<button></button>").text("supprimer");
260     supp_btn.addClass("style_button_supp")
261     div_button.append(supp_btn);
262     // réaction on click sur le bouton supprimer
263     $(supp_btn).click(function(){
264         supprimer_article(el.sneakers_id);
265         // window.Location.reload(true);
266     })
267
268
269
270     const modif_btn = $("<button></button>").text("modifier");
271     modif_btn.attr("id","style_button");
272
273
274     // réaction on click sur le bouton modifier
275     $(modif_btn).click(function(){
276         // sneakers id en paramètre de la fonction car besoin dans les data attendus pour l'appel ajax
277         // chercher comment enlever les offre sneakers et mettre à jour
278         modifier_article(el.sneakers_id);
279     })
280
281     div_button.append(modif_btn);
282
283 } else {
284     //bouton pour modifier une annonce
285     const add_shopping_btn = $("<button></button>").text("ajoutez au panier");
286     add_shopping_btn.addClass("style_button_panier");

```

Enfin je met l'ensemble des données dans leur div respective et les envoie vers la page HTML order.html dans une div avec l'id "article".

Je souhaite que si l'utilisateur veut voir tous les produits, il doit cliquer sur un bouton. Pour cela j'utilise la fonctionnalité de réaction au click. Si cela se produit, un appel ajax est envoyé vers la page sneakers.php et avec la superglobal GET récupère l'ensemble des données de cette base de donnée (sneakers). Si ceci est un succès, alors un appel de la fonction d'affichage des sneakers est fait.

```

298
299     div_info_produit.append(price_html,states_html,size_html,stocks_html,color_html);
300     div_image.append(div_info_produit);
301     div.append(brand_html,div_image,div_button);
302     // on envoi l'ensemble de c'est balise et donnée sur la page.HTML
303     $("#article").append(div);
304
305 }
306 }
307
308 $("#buttonProduct").click(()=>{
309     $.ajax({
310         url: "../sneakers.php",
311         type: "GET",
312         dataType: "json",
313         // async:false,
314         data:{
315             opt:"select",
316         },
317         success : (res)=>{
318             sneakers(res.sneakers);
319         }
320     })
321 })
322

```

Si un utilisateur ne souhaite voir qu'un produit en particulier, alors il peut utiliser la barre de navigation qui est activée lorsqu'une touche est appuyée. Ceci va créer une constante search qui va récupérer les lettres sur lesquelles l'utilisateur a appuyé. Si le nombre de lettre enregistré dans la constante est supérieur ou égal à trois, alors se déclenche un appel ajax, vers la page ssneakers.php. On Récupère les données a partir de la case search du switch de la page php : sneakers.php et par l'utilisation de la superglobale GET.

Mais cela prend aussi la valeur de la constante search pour la comparer au différent nom des sneakers enregistrés dans la base de données : sneakers. Si ceci est un succès alors un appel de la fonction d'affichage des sneakers est enclenché avec comme paramètre d'entrée le tableau de valeur qui concordait avec les touches sur lesquel avait appuyer l'utilisateur. Si a la fin de sa recherche, l'utilisateur souhaite renouveler cette expérience, il clique sur le bouton nouvelle recherche qui va alors recharger la page et donc la rendre vierge de toute offre de sneakers.

```

322
323     $("#nav_bar").keyup(() => {
324         const search = $("#nav_bar").val(); // Je récupère La valeur de La barre de recherche
325         // Si La recherche contient 3 caractères ou plus alors
326         // console.log(search);
327         // console.log(search)
328         if ($.trim(search).length >= 3) {
329             $.ajax({
330                 url: "../sneakers.php",
331                 type: "GET",
332                 dataType: "json",
333                 data: {
334                     opt: "search",
335                     search
336                 },
337                 success: (res) => {
338                     // console.log(res.sneakers);
339                     sneakers(res.sneakers);
340
341                 }
342             })
343         }
344     })
345
346     $("#new_search").click(function(){
347         window.location.reload(true);
348     })
349
350

```

3.5) FAQ.js :

Terminons par la page FAQ. Sur celle-ci, on a différentes fonctionnalités faites en js : on a l'inscription à la newsletter et l'envoi de questions auquel cas, elle ne serait pas renseignée dans la FAQ sur la page html.

```

1  $("#newsletter").click((event)=>{
2      event.preventDefault();
3      let users = JSON.parse(localStorage.getItem("user"));
4      let val_newsletter = users["newsletter"]
5      const val_newsletter_1 = 1;
6      const val_newsletter_0 = 0;
7
8      if (val_newsletter == 0){
9          submit_faq(users,val_newsletter_1);
10     }
11     if (val_newsletter == 1) {
12         submit_faq(users,val_newsletter_0);
13     }
14
15 })
16
17 function submit_faq(users,newsletter){
18     console.log(newsletter);
19     $.ajax({
20         url:"../users.php",
21         type: "GET",
22         dataType: "json",
23         async:false,
24         data:{
25             opt : "update_newsletter",
26             users_id : users["users_id"],
27             newsletter:newsletter,
28         },
29         success: (res) => {
30             console.log("ok");
31
32         }
33     })
34
35
36 }
37

```

Pour la fonctionnalité d'inscription à la newsletter tout d'abord, j'utilise une action lors de la soumission du bouton qui est dans une balise form. Dans cet évènement, j'empêche le rechargement de la page avec event.preventDefault(). Je crée ensuite une première variable users qui récupère les données en localhost dont le users_id, newsletter et admin. Puis je récupère la valeur de la newsletter qui est stockée en LocalStorage en variable val_newsletter. Je crée ensuite deux constantes qui gardent les deux valeurs possibles pour la newsletter.

Ensuite, je fais deux ifs qui suivant la valeur récupérée du LocalStorage l'a compare avec 0 et 1. Ceci déclenche alors un appel de fonction. La fonction appelée est submit_faq qui prend en argument l'id_users et la nouvelle valeur de la newsletter. Ensuite, je fais un appel ajax, qui appelle le fichier : users.php, récupère les données : opt (option du switch de la page users.php), users_id et enfin newsletter avec la superglobale GET. L'option choisie est update_newsletter pour mettre à jour la valeur.

```

38
39 $("form").submit((event)=>{
40   event.preventDefault();
41   console.log("ok");
42   let question = $("textarea").val();
43
44   $.ajax({
45     url:"../users.php",
46     type: "GET",
47     dataType: "json",
48     async:false,
49     data:{
50       opt : "question_FAQ",
51       question : question,
52     },
53     success: (res) => {
54       alert("Votre question a bien été envoyée au service de communication merci")
55     }
56   })
57 })
58
59
60 })
61

```

Passons maintenant à la deuxième fonctionnalité qui est gérée par le js dans cette page HTML. C'est l'envoi de questions de la part des utilisateurs qui n'ont pas trouvés leur réponse sur la page.

Pour réaliser cette fonctionnalité, j'ai utilisé : submit du formulaire en appuyant sur le bouton : "Envoie ta question".

Au clic, dans une variable "question" je récupère la valeur qui est entrée par l'utilisateur.

Ensuite je fais un appel : ajax vers le fichier users.php, qui par la superglobal GET récupère les datas :

- opt : "question_FAQ" (un des choix du switch du fichier users.php)
- La variable question

Si cette récupération de données fonctionne, alors j'appelle le système de mailer_question.php.

3.6) Fonctionnalité js de la partie admin :

La partie admin contient différentes fonctionnalités js :

Affichage des informations utilisateurs / des offres produits / commandes

Suppression d'un utilisateur / paire de sneakers / suppression d'une commande

Pour aborder ces fonctionnalités jQuery, on va se concentrer sur la partie utilisateur. Au début de chacune de ces trois pages javascript, on va vérifier que l'utilisateur connecté est administrateur.

```
1 users = JSON.parse(localStorage.getItem("user"));
2 console.log(users.admin);
3
4
5 if(users.admin == 0) window.location.replace("../login_logout/login.html");
6
```

Pour créer le tableau de donnée, je crée une fonction qui va contenir différente balise HTML (<tr> et <td>) dans une balise <tr> j'ajoute des balises <td>. Cette fonction prend en paramètre les données qui sont récupérées dans l'appel js qui suit. Cet appel ajax est fait vers la page php admin avec la superglobale GET avec comme option "select".

RGPD :

Durant cette année, j'ai suivi sur le site secnumacademie.gouv.fr des cours autour du sujet des RGPD. Ces cours se découpent en 4 parties différentes :

- Panorama de la SSI
- Sécurité de l'authentification
- Sécurité sur internet
- Sécurité du Poste de travail et nomadisme.

The screenshot shows the 'secnumacademie.gouv.fr' website interface. It features two main modules: 'MODULE 1 Panorama de la SSI' and 'MODULE 2 Sécurité de l'authentification'. Each module has a progress bar, a 'Temps passé' (Time spent) counter, a 'Score' (Score), and a 'Afficher les unités' (View units) button. The background of each module contains icons related to its respective topic.

MODULE 1
Panorama de la SSI
⌚ Temps passé : 22:07:58 ★ Score : 80%
Afficher les unités

MODULE 2
Sécurité de l'authentification
⌚ Temps passé : 03:06:14 ★ Score : 84%
Afficher les unités

The screenshot shows the 'secnumacademie.gouv.fr' website interface. It features two more modules: 'MODULE 3 Sécurité sur Internet' and 'MODULE 4 Sécurité du poste de travail et nomadisme'. Each module has a progress bar, a 'Temps passé' (Time spent) counter, a 'Score' (Score), and a 'Afficher les unités' (View units) button. The background of each module contains icons related to its respective topic.

MODULE 3
Sécurité sur Internet
⌚ Temps passé : 07:57:44 ★ Score : 84%
Afficher les unités

MODULE 4
Sécurité du poste de travail et nomadisme
⌚ Temps passé : 03:40:27 ★ Score : 86%
Afficher les unités

Nous allons maintenant rentrer plus en détail dans ces quatres parties, tout d'abord le panorama de la SSI (sécurité des systèmes d'information).

1) Panorama de la SSI :

Cette première partie se décompose en cinq sous unité :

- Un monde numérique hyperconnecté
- Un monde à risque
- Les acteurs de la cybersécurité
- Protéger le cyberespace

- Les règles d'or de la sécurité

1.1) Un monde numérique hyperconnecté :

Cette sous partie commence par un point crucial qui crée une des raisons de la multiplication des facteurs de risques des attaques cybercriminel qui est la diversité d'équipement que chacun a sous son utilisation. Puis le point de vue s'élargie, en abordant le cyberspace qui regroupe un espace sans frontière concrète. Mais ceci reste à nuancer par le fait que les infrastructures sont et restent technique et physique.

Et enfin, quel sont les règles dans cette espace. Ceci passe par des organisations à différentes échelles qui peuvent avoir un rôle de censure et de contrôle sur les informations qui circulent. En France, l'organisation qui est chargée de cette mission est la CNIL, commission nationale informatique & liberté. Leurs quatre missions principales sont :

- Informer et protéger les particuliers et les professionnels
- Accompagner et conseiller
- Contrôler et sanctionner (financièrement et/ou juridiquement)
- Anticiper
-

1.2) Un monde à risque :

Un monde à risque par notamment la multiplication des facteurs à haut risque et des portes d'entrées dont disposent les hackers pour rentrer dans un réseau d'entreprise. La seconde partie s'intéresse aux différents types d'attaque, elles sont soit massive ou ciblée. Pour ce qui est des attaques massives, elles sont par exemple :

- le phishing
- Ddos : le déni de service
- Logiciel malveillant

L'autre type d'attaque sont les attaques ciblées, celle-ci sont ciblées dans le cadre majoritairement de l'espionnage industriel ou économique ou atteintes à la propriété intellectuelle ou encore pour des raisons politiques. On compte de nombreux autres exemples d'attaque cybercriminel comme :

- Ver : C'est un logiciel malveillant qui se reproduit sur plusieurs ordinateurs en utilisant un réseau informatique comme Internet. Il peut se dupliquer une fois qu'il a été exécuté.
- Phishing : Consiste à envoyer des mails malveillants conçus pour tromper et escroquer les utilisateurs. L'objectif est la divulgation d'informations sensibles par les personnes recevant ces mails.
- Spyware : Fonctionne par infiltration et la discréption de ce virus qui va récupérer des informations sensibles : identifiant de connexion, activité en ligne et identifiant de compte. Il reste inconnu jusqu'à ce qu'il prenne contrôle de l'ensemble du système.
- Botnet : Le botnet est un ensemble d'ordinateur qui sont sous le contrôle d'un attaquant, ceci lui permet de faire des attaques avec une force d'attaque plus importante.
- Virus : Programme malveillant qui a pour objectif de perturber le bon fonctionnement d'un ordinateur.
- Cheval de troie : C'est un type de virus qui est installé sous la forme d'un logiciel totalement légitime.
- DDOS : Ce type d'attaque fonctionne via un envoi massif de requête auprès d'un serveur pour le mener à saturation.
- Cryptovirus / Ransomware : Empêche les utilisateurs d'accéder à leurs fichiers ou système personnel et exige le paiement d'une rançon en échange du rétablissement de leur accès.

1.3) Acteur majeur de la cybersécurité :

Pour assurer la cybersécurité en France en 2013, est rédigé une nouvelle version du livre blanc pour la défense et la sécurité nationale qui inclut dedans des règles sur la cybersécurité. Ceci permet aussi de remarquer l'importance du risque de menace que représente les cyberattaques classées deuxième derrière le terrorisme. Cela permet aussi de visualiser que les menaces concernent tous les acteurs Français administratif et économique. Le premier livre précisait déjà l'importance des attaques cyber criminel et avait permis la création de l'ANSSI (Agence National de la sécurité des systèmes d'information).

2015, a été l'année de mise en place de la première stratégie pour la sécurité numérique. Les 5 axes de la sécurité numérique sont :

- Les questions de sécurité numériques auprès des OIV (liste regroupant les opérateurs d'importance vitale en France)
- Le rôle des entreprises non OIV
- La formation à la cybersécurité
- Le rôle des industries
- La stratégie du numérique dans le contexte international

Intéressons-nous maintenant au rôle de l'ANSSI et a ces différentes missions :

- Prévoit des mesures en cas de crise des systèmes d'information vitaux de la Nation
- Coordonne les missions gouvernementales en matière de défense des systèmes d'informations
- Anime et coordonne les travaux interministériels en termes de sécurité numérique
- Conçoit et fait réaliser des systèmes de communication interministérielles hautement sécurisé
- Délivre des agréments aux produits et aux prestations de service destinées à protéger les systèmes d'information des entreprises

Depuis 2015, mise en place d'un système territorialisé chargé de relayer et de coordonner les actions de l'ANSSI au niveau du territoire.

L'ANSSI n'est pas le seul et unique agent chargé de s'occuper de cette sécurité il y a aussi :

- Ministère des armées : S'occupe de potentiel guerre numérique, menace étatique étrangère.
- Ministère de la défense avec deux organisations: CALID qui est chargé de surveillance des différences SI (système d'information) des ministres et partage d'outil de détection de menace avec l' ANSSI (Centre d'analyse en lutte Informatique Défensive), CERT (Computer Emergency Response Team)

Le Comcyber, le commandement opérationnel de cybersécurité (ComCyber) assure la conduite des opérations de défense des SI. À ce titre, il commande le CALID, mais aussi plus généralement toutes les opérations militaires dans le cyberspace.

On compte aussi de nombreux acteurs qui aident des entreprises dans la protection de leur SI en faisant appel à un audit de code, test d'intrusion.

1.4) Protéger le cyberspace :

La cybersécurité compte certaines règles d'or que toute personne se doit d'appliquer sur ses différents appareils:

- Choisir ses mots de passe selon des règles précises de sécurité même si beaucoup de sites aiguille l'utilisateur sur la difficulté de cracker son mot de passe et n'autorise pas les mots de passe trop faible, ceci permet de limiter les intrusions et usurpation d'identité.
- Mettre à jour ses logiciels régulièrement, ceci permet d'avoir un logiciel qui aura plus de mal à être corrompu et donc à limiter les portes d'entrée des cyberattaquants.
- Effectuer des sauvegardes régulières de ses données et si possible multiplier la diversité des espaces de stockage des données
- Télécharger ses programmes sur les sites officiels des éditeurs

- Être vigilant lors d'un paiement internet
- Prendre soin de son identité numérique

1.5) Les règles d'or de la sécurité :

Tous d'abord intéressons nous à définir ce qu'est une donnée qui englobe plusieurs caractéristiques :

- Un type
- Un critère
- Des droits
- Moyen d'accès

Il existe trois différents niveaux de risque pour nos données :

- Disponibilité : pas disponible pour un utilisateur autorisé
- Intégrité : modification non autorisée d'une donnée
- Confidentialité : divulgation non autorisée de donnée

Tout salarié d'une entreprise a comme devoir de protéger le patrimoine informationnel, mais aussi la réputation et l'image de l'entreprise. L'entreprise met aussi en place au préalable des sécurités mais la non-vigilance d'une personne peut gravement nuire à la société. Il faut rester vigilant par rapport au risque : d'hameçonnage, rançonnage et de compromission par des virus.

2) Sécurité de l'authentification :

Cette deuxième partie se décompose en quatre sous partie :

- Principe de l'authentification
- Attaque sur les mots de passe
- Sécuriser ses mots de passe

- Gérer ses mots de passe
- Notion de cryptographie

2.1) Principe de l'authentification :

L'authentification est une étape de contrôle indispensable puisqu'elle vérifie l'identité communiquée par un utilisateur lors de sa connexion sur un service. C'est une étape essentielle de la sécurité. Deux principes entre en compte : d'authentification et d'autorisation.

Les preuves d'authentifications peuvent prendre différentes formes le plus souvent un mot de passe a caractère multiple et différent mais peut aussi prendre comme forme :

- Code pin
- Réponse a une question secrète
- Un schéma de déverouillage

La méthode la plus prometteuse de l'authentification, est la reconnaissance biométrique. Elle est aussi compliquée a mettre en place. Elle peut être sous différentes formes :

- Empreinte rétinienne
- Structure de la main
- Empreinte digitale
- Voix
- Structure faciale

Malheureusement cette méthode a ces limites : son prix pour l'entreprise et la mise en place de lourd moyen mais aussi sur un aspect juridique elle pose problème sur le fait de stocker des caractéristiques morphologiques des personnes.

Le mot de passe reste la méthode la plus répandu mais pas sans faille car sa divulgation par négligence (mot de passe faible, diffusion a un tiers) ou malveillance (hameçonnage ou phishing, attaque par ingénierie sociale ou bien attaque par force brute)

2.2) Attaque sur les mots de passe :

Il y a deux types d'attaque sur les mots de passe :

- Attaque directe : Attaque ciblée vers un système d'authentification ou un mot de passe précis. L'objectif étant de se connecter sous votre identité numérique. (Par exemple, l'attaque force brute qui consiste à créer un code qui va tester un ensemble de combinaisons possible pour cracker votre mot de passe, attaque la moins rapide)
- Attaque indirecte : Ce type d'attaque est basé sur la ruse par la technique du fishing par exemple. Elle fonctionne via la création d'un site similaire en tout point au site officiel mais qui ne l'est pas l'objectif, c'est d'en tirer des informations personnelles sensibles.

2.3) Sécuriser ses mots de passe :

Comment s'assurer d'avoir un mot de passe fort. Pour cela, il doit pouvoir être compliqué à être révélé par un attaquant dans un temps raisonnable. Il doit aussi pouvoir résister à différentes attaques :

- Attaque par dictionnaire
- Attaque par brute force
- Attaque par permutation
- Attaque distribuée
- Attaque de proximité

Un mot de passe fort doit être composé :

- Un ou plusieurs caractères spéciaux
- Lettre majuscule
- Lettre minuscule et chiffre

A éviter les dérives de mot du dictionnaire qui sont fortement déconseillés comme mot de passe.

2.4) Gérer ses mots de passe :

La multiplication des services en ligne demandant une authentification, crée une multiplication des mots de passe qui rend de plus en plus compliqué de s'en rappeler si on veut éviter de mettre le même mot de passe partout. Il existe des solutions à ce problème :

- Le coffre-fort de mot de passe : Souvent sous forme d'application verrouillée par un mot de passe et contient un tout nouveau mot de passe de façon crypté.
- SSO (single sign on) : Se crée une authentification via un service comme google, Facebook, par exemple.

2.5) Notion de cryptographie :

Cette sous partie nous montre deux types de chiffrement :

- Chiffrement symétrique : La cryptographie symétrique est également appelée cryptographie à clé secrète. Une même clé secrète est partagée entre deux interlocuteurs, et ceux-ci doivent connaître la clé pour pouvoir communiquer.
- Chiffrement asymétrique : Cette approche est basée sur l'utilisation de 2 clés: une clé publique et une clé privée qui sont dépendantes l'une de l'autre.
 - o La clé « publique » est une clé qui permet de chiffrer le message.
 - o La clé « privée » est une clé qui permet, quant à elle, de déchiffrer le message.

Le risque avec le chiffrement asymétrique, c'est le man in the middle qui consiste à intercepter une communication sans que ni l'envoyer ou le récepteur ne s'en rende compte.

3) Sécurité sur Internet

3.1) Internet de quoi s'agit-il ? :

Ensemble de réseaux mondiaux interconnectés qui permet à des ordinateurs et à des serveurs de communiquer efficacement au moyen d'un protocole de communication commun (IP).

Maintenant que nous avons défini ce qu'est internet, intéressons-nous à la cyber malveillance, notamment avec des attaques informatiques de grande ampleur. Mais internet a permis à ces groupes, de se créer de nouvelles sources de revenu mais aussi à des personnes curieuses de tester des attaques sur des personnes de leur entourage.

Désormais se pose aussi un concept déjà évoqué précédemment et qui se nomme l'ingénierie social ensemble des attaques informatiques mettant l'accent sur les vulnérabilités humaines. La responsabilité humaine a une grande importance et ne peut pas être négligé. Pour contrer ce système, il existe un processus en trois étapes :

- Informer : par des campagnes d'informations ou d'exercice à une attaque par l'ingénierie sociale
- Sensibiliser
- Intégrer l'utilisateur

Des organisations en France sont prévues pour aider en cas d'attaque comme l'ANSSI.

3.2) Les fichiers en provenance d'internet :

Chaque fichier dispose d'un format et d'une extension. Aucun format ne dispose d'aucune faille. Les formats les plus exploités par les pirates sont-ils des textes formatés (PDF, DOC(X) / XLS(X)). Les raisons sont premièrement le fait que ce sont les extensions de documents les plus utilisées. Deuxièmement, ces formats de fichier sont intrinsèquement plus complexes et donc plus vulnérables que des fichiers texte brut car peuvent contenir des scripts exécutables, exemple: Javascript pour PDF.

Extension	Développeur / Editeur	Format	Logiciels associés
.docx .pptx etc.	Microsoft	Microsoft Excel Open XML Spreadsheet	Microsoft Excel 2007, LibreOffice, Open Office, Oxygen Office Professional (Linux)
.ai .psd etc.	Adobe Systems	Portable Document Format	Adobe Viewer, Ghostscript, Ghostview Xpdf, gPDF, Acrobat Reader
.amv .mpeg etc.	Moving Picture Experts	MPEG-4 Video Stream	Real media Player, WinAmp, Windows Media Player, iTunes

Les sources les plus fiables sont le fait de télécharger ses logiciels depuis les sites officiel des utilisateurs. Des pirates craquent des logiciels payants et désactivent la demande de clé logiciel lors du démarrage du logiciel. Ceci reste illégal et dangereux car le pirate a pu introduire dedans un code qui lui rapportera infinie de l'argent. En cas de rançonnage, alors la seule solution pour contrer celle-ci est d'avoir plusieurs sauvegardes de ses données sur différents espaces de stockage en organisant des sauvegardes mensuelles et hebdomadaires.

3.3) La navigation web :

Tout d'abord, la navigation web fonctionne par une adresse compréhensible et si possible mémorisable par l'homme.

Voyons maintenant ce qu'est la typosquatting, qui consiste à créer un site dont l'url se rapproche d'un site officiel.

Par exemple, en changeant deux lettres ou en inversant deux caractères. Un moteur de recherche est la porte d'entrée du web.

Le moteur de recherche parcourt des bases de données gigantesques et propose une liste de site issu des caractères rentrés dans la barre de recherche. Ceux qui différencient les moteurs de recherche sont les algorithmes, différence de philosophie et une question de goût et d'utilisation. Maintenant les cookies, qui ont pour objectif de préserver la vie privée des internautes. Vous êtes alors informés que le site va récupérer certaines de vos données. Ceci a aussi pour autre objectif de vous amener par la suite à avoir des publicités ciblées.

3.4) La messagerie électronique :

On va s'intéresser aux bonnes pratiques de messagerie. Dans la partie précédente, on avait vu que la première étape importante était de protéger tous les comptes avec un mot de passe fort et via un système d'authentification numérique. Il est recommandé d'avoir une adresse mail par fonction :

- Adresse pour les communications
- Adresse pour les achats en ligne
- Adresse professionnelle
- Adresse démarche administrative

Permettra d'identifier plus facilement les courriels frauduleux. Les messageries se divisent en deux catégories :

- Client léger : messagerie en ligne aussi appeler WebMail
- Client lourd : logiciel de messagerie installé sur l'ordinateur et qui permettra de regarder notre messagerie hors ligne si récupérer lors d'une précédente connexion

On peut aussi remarquer une autre catégorie qui sont les messageries instantanées qui sont des applications mobiles qui sont des applications lourdes. Mais ces conversations restent peu sécurisées.

3.5) L'envers du décor d'une connexion web :

Les fonctionnalités basiques d'une connexion web, lorsqu'on tape l'adresse la première étape est l'obtention de l'adresse ip qui y est relié ceci s'appelle la résolution DNS. Le DNS a un rôle d'annuaire. L'url va ensuite être lu de droite à gauche. Tout à droite, la donnée de l'AFNIC, ensuite savoir qui héberge le site, puis le nom de domaine.

Un serveur mandataire filtre les sites Web que vous consultez. Il reçoit les requêtes de votre navigateur pour récupérer les pages Web demandées avec leurs éléments et dans le respect des règles éditées et=écide de vous les transmettre ou non.

4) Sécurité du poste de travail et nomadisme :

4.1) Application et mise à jour :

Débutons par le concept de vulnérabilité. Les personnes cherchant de façon journalière comment connaître mieux les systèmes informatiques sont nombreux que ce soit des criminels service de renseignement mais aussi des chercheurs dans le domaine privé ou universitaire. La sécurité par obscurité, est une des premières méthodes de sécurité, c'est le fait de maintenir un attaquant potentiel

dans l'ignorance du fonctionnement interne du fonctionnement internet auquel il s'attaque. Ceci consiste plus concrètement à ne pas donner le code de fonctionnement.

4.2) Option de configuration de base :

La configuration d'un nouveau appareil passe par des étapes similaires à tous les appareils. Tout d'abord premier démarrage : celui-ci commence par la création d'un compte permettant de créer une première identité numérique sur l'ordinateur qui pourra aussi peut-être servir pour des services en ligne. S'ensuit des mises à jour obligatoire pour assurer la sécurité de l'appareil. Puis ensuite, il est important de protéger son appareil par un mot de passe fort. Sur un ordinateur le mieux restera de mettre un mot de passe au niveau du BIOS. De nombreux appareils proposent aujourd'hui d'activer le chiffrement du stockage (disque dur, mémoire flash, carte SD). Cette étape permet de protéger les données contre un accès illégitime en cas de tentative de vol des données.

4.3) Configuration complémentaire :

Dans les configurations complémentaires, une d'entre elle est la gestion des comptes utilisateurs. On rattache généralement au compte l'accès à des documents mais également des comptes chez des fournisseurs multimédia, des applications, des moyens de paiement. C'est là toute l'importance d'avoir un compte par personne utilisant l'appareil pour éviter toute action sans autorisation comme un paiement en ligne. Cette possibilité reste réservée majoritairement aux ordinateurs. Il est aussi possible de mettre en place des UCA (users access control) mais surtout de les abaisser en activant par exemple l'ouverture automatique d'une session ce qui n'est pas du tout conseiller.

4.4) Sécurité des périphériques amovible :

Ce périphérique externe comporte un risque lors de sa connexion à un ordinateur, c'est pour ça qu'il est fortement déconseillé de brancher tout périphérique externe sans être sûr qu'il ne soit pas corrompu. Ce périphérique compromis peut réaliser différentes actions sur la machine compromettant la chaîne de démarrage de l'ordinateur, télécharger et exécuter du contenu malveillant ou même reprogrammer d'autres périphériques USB connectés à l'ordinateur. Mais ces périphériques ne sont pas que des clés USB ou disque dur externe mais cela peut être une cigarette électronique qu'on recharge sur son PC.

En cas de perte ou de vol d'un périphérique amovible externe, il est conseillé pour éviter la fuite de donnée, de crypter les données présentent sur l'ordinateur.

4.5) Séparation des usages :

Débutons par le mélange des usages, celui-ci est la disparition progressive de la barrière entre le monde professionnel et personnel. L'exemple qui montre le mieux ce concept, est le travail dans son milieu personnel avec son matériel personnel. Cette activité aussi appeler BYOD (bring your own device), doit être proscrit car elle pose des problèmes concernant la sécurité sur plusieurs points :

- En cas de vol ou perte
- Intrusion
- Manque de contrôle sur l'utilisation des appareils par les collaborateurs
- Fuite des données lors du départ d'un collaborateur

La situation inverse existe et n'est pas sans risque non plus pour la protection de donnée personnel.

Conclusion

En conclusion de ce projet de développement de site web dynamique et responsive portant sur la vente / achat de sneakers. Mon objectif été de crée une plateforme de vente de sneakers facile d'utilisation.

Les fonctionnalités que je souhaitais réaliser son réussi, j'ai aussi su gérer le temps qui était à ma disposition pour développer ce projet. J'ai pu utiliser l'ensemble de ce que j'ai appris cette année. Que ce soit les langages de programmation que j'ai utilisé : PHP, HTML, CSS, jQuery.

Ce projet représente un réel accomplissement pour moi dans la gestion de problème mais aussi de concrétiser ce que j'ai appris pendant cette année.

Table des illustrations

page 11 : Dictionnaire de donnée

page 12 :MCD

page 13 : MLD

page 14: MPD

page 15 : code SQL base de donnée

page 16 : diagramme de cas d'utilisation

page 17 : Diagram de séquence

page 18 : Screen code db_connect.php

page 19 : Screen code function.php

page 21-23 : Screen code users.php

page 25-28 : Screen code sneakers.php

page 30 : register.php

page 31 : Screen code login.php / logout.php

page 32 : Screen code mailer.php

page 33-34 : Screen code order.php

page 34 : Screen code update_newsletter.php

page 36-38 screen page admin.php

page 39-40 register.html

paage 42-44 login.html

page 43-44: create_product.html

page 45-46 : FAQ.Html

page 47 : catalogue.html

page 48-49 : admin.html

page 50-51 : account.html

page 52 : welcome.html

page 54 : register.js

page 54 : login.js (bas de page)

page 56 : create_product.js

page 58-61 : catalogue.js

page 62-63 : FAQ.js

page 64 : admin.js

page 65 : RGPD

page 75 : RGPD tableau

Glossaire (liste mot compliquer)

- RGPD : Règlement Général sur la protection des données
- Table : Les tables de base de données relationnel sont un ensemble de données organisées sous forme d'un tableau ou les colonnes correspondant à des catégories d'information.
- base de données : Permet de stocker et de retrouver des données structurées sous un rapport avec un thème ou une activité.
- superGlobale : Les Superglobales sont des variables array, internes et prédéfinies par PHP, qui sont toujours disponibles, quel que soit le contexte
- requete SQL : Les requêtes SQL regroupent quatres types de requête : INSERT, UPDATE, DELETE, SELECT. Elle permette de récupérer des données en base de données.
- Requête préparer / Query : Ces deux types de requête sont utilisées pour des choses différentes :
 - Une requête préparée permet d'éviter les infections SQL lors de l'exécution de la requête.

- Switch : Un switch est une fonctionnalité php qui est similaire à une série d'instruction de condition "if". Elle permet d'avoir des choix multiples.
- LocalStorage : Technique d'enregistrement de donnée dans un navigateur web.

Webographie (site web visitée)

Site de vérification de la syntaxe CSS (w3c css validator) : <https://jigsaw.w3.org/css-validator/>

Site de vérification de la synthaxe HTML (W3C validator) : <https://validator.w3.org/>

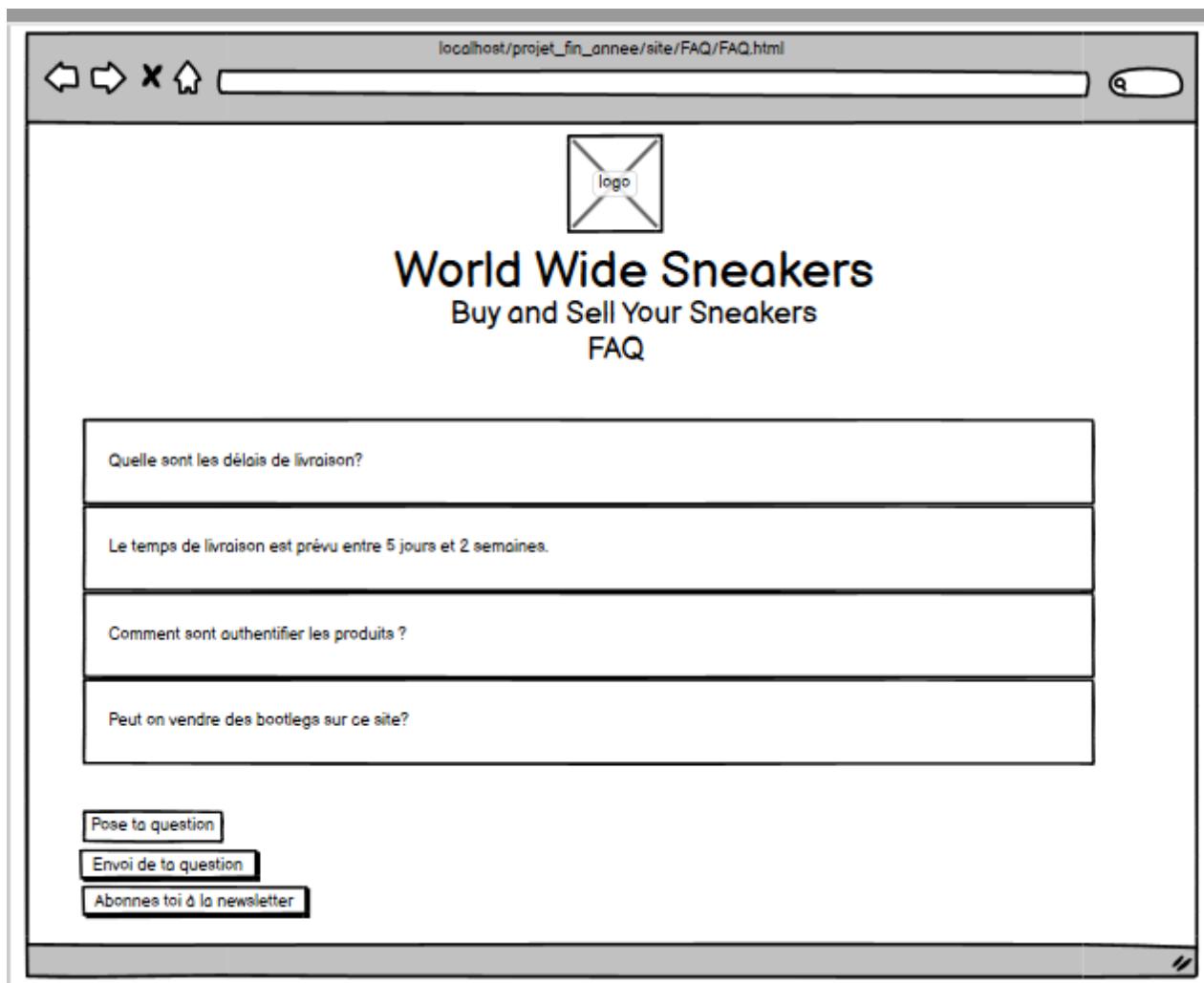
Site du manuel PHP : <https://www.php.net/manual/fr/index.php>

Site de la documentation de Bootstrap : <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

Site de formation RGPD : secnumacademie.gouv.fr

Annexes

Wireframe et Maquette Page FAQ :





World Wide Sneakers Buy & Sell Your Sneakers FAQ

Qu'elles sont les délais de livraison ?

Comment sont authentifier les produits ?

Peut on vendre des bootlegs sur ce site ?

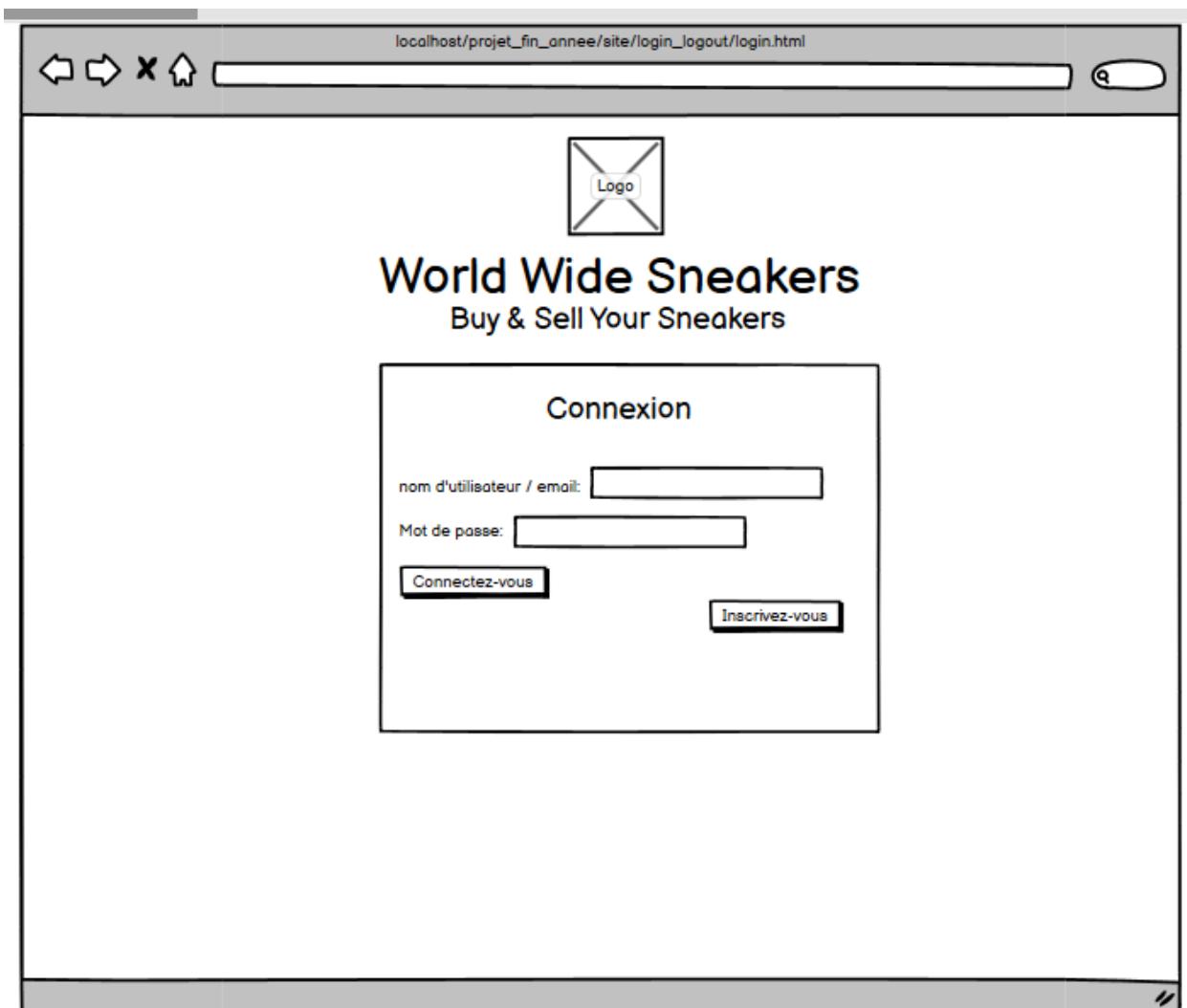
Pose ta question

[Envoi ta question](#)

[Abonne-toi à la newsletter](#)

©WWS

Wireframes et Maquette de la page Connexion (Login) :





World Wide Sneakers
Buy & Sell Your Sneakers

Connexion

Nom d'utilisateur / email: _____

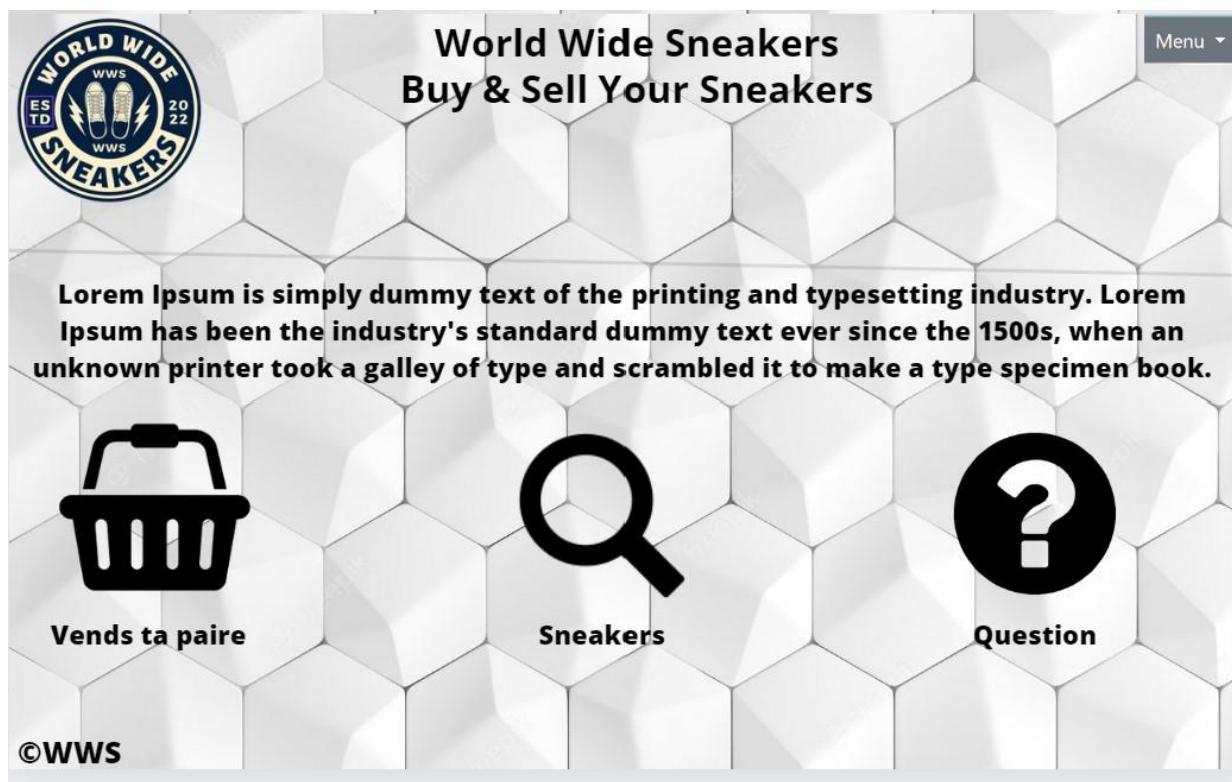
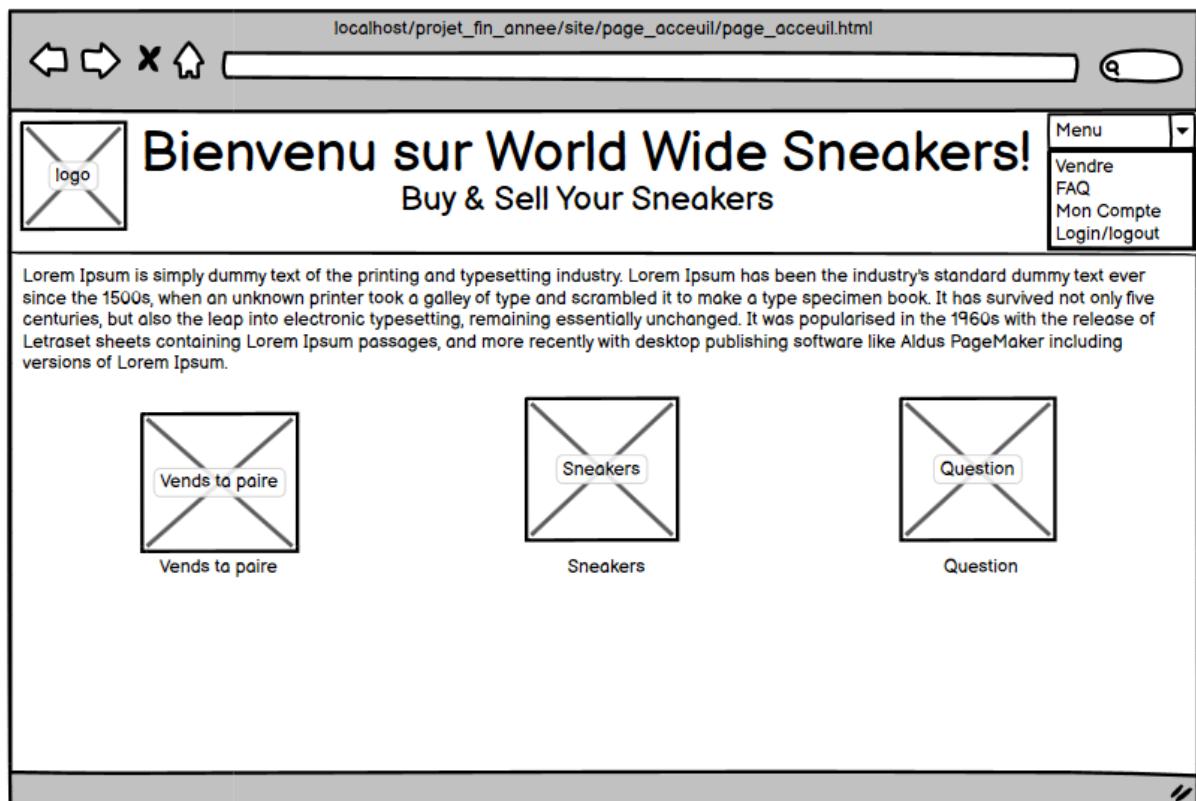
Mot de Passe : _____

[Connectez-vous](#)

[Inscrivez-vous](#)

©WWS

Wireframe et Maquette page Accueil :



Wireframe et Maquette page Compte :

Account Page

The wireframe shows a standard web browser interface with a header containing back, forward, and search buttons. Below the header is the website logo 'World Wide Sneakers' with the tagline 'Buy & Sell Your Sneakers'. The main content area is divided into two main sections: 'Mon compte' on the left and 'Panier' on the right. The 'Mon compte' section contains input fields for Name, First Name, Email, and Username. The 'Panier' section displays two items from the cart, each with a delete icon, followed by a 'Valider commande' button.

World Wide Sneakers
Buy & Sell Your Sneakers

Mon compte

Nom

Prénom

Email

Username

Panier

Nom de la sneakers
Prix
Quantité

Nom de la sneakers
Prix
Quantité

Valider commande

WWS

The mockup is set against a background of light gray hexagonal tiles. It features a circular logo on the left with the text 'WORLD WIDE SNEAKERS' and 'ESTD 2022'. The main title 'World Wide Sneakers' and tagline 'Buy & Sell Your Sneakers' are centered above the account sections. The layout is identical to the wireframe, with 'Mon compte' on the left and 'Information Commande' on the right, which includes a list of items and a 'Valider commande' button.

World Wide Sneakers
Buy & Sell Your Sneakers

Mon compte

Nom

Prénom

Email

UserName

Information Commande

Nom de la sneakers
Prix
Quantité

Nom de la sneakers
Prix
Quantité

Valider commande

WWS

Wireframe et Maquette page catalogue :

Page Catalogue

The wireframe shows a browser window titled "Page Catalogue". At the top left is a logo consisting of a square with an X. To its right is the title "World Wide Sneakers" and the subtitle "Buy & Sell Your Sneakers". Below the title is a search bar with a placeholder "Nouvelle recherche" and a "Voir les sneakers" button. The main content area contains two identical item cards, each labeled "Brand". Each card features a square icon with an X, a list of filters ("Prix", "Etat", "Taille", "Color", "Stock"), and three buttons: "Modifier", "Supprimer", and "Ajoutez aux commandes". A footer bar at the bottom is labeled "WWS".

The stylized maquette of the Catalogue page is set against a background of large, light-grey hexagons. It features a circular logo on the left with the text "WORLD WIDE SNEAKERS" and "ESTD 2022". The main header "World Wide Sneakers" and "Buy & Sell Your Sneakers" is centered above a search bar and a "Nouvelle Recherche" button. A "Voir les sneakers" button is located below the search bar. The main content area displays two items, each with a large "BRAND" heading. The first item shows a black and white sneaker with a checkered pattern and a list of filters. The second item shows a red and white high-top sneaker with a similar list of filters. Both items have "Modifier", "Supprimer", and "Ajoutez aux commandes" buttons. The footer bar at the bottom is labeled "WWS" and "090".

Wireframe et Maquette page ajout produit :

The image shows a wireframe and a mockup of a web page titled "Création Produit" (Product Creation). The wireframe is on the left, showing the overall layout with a header, logo, title, and form fields. The mockup is on the right, showing the same layout with specific field details.

Header: localhost/projet_fin_annee/site/create_product/page_creation_produit.html

Logo: Logo (represented by a square icon with an 'X' inside)

Title: World Wide Sneakers
Buy & Sell Your Sneakers

Form Fields (Wireframe):

- Nom du modèle: [Input Field]
- Taille: [Input Field]
- Couleurs: [Input Field]
- Etat: [Input Field]
- stock: [Input Field]
- prix: [Input Field]

Image Section:

- Image [Label]
- Choose file [Input Field] No file chosen
- [Submit Button] Met en ligne ton article



World Wide Sneakers

Buy & Sell Your Sneakers

Creation produit

Nom du modèle:

Taille:

Couleurs:

Etat:

Stock:

Prix:

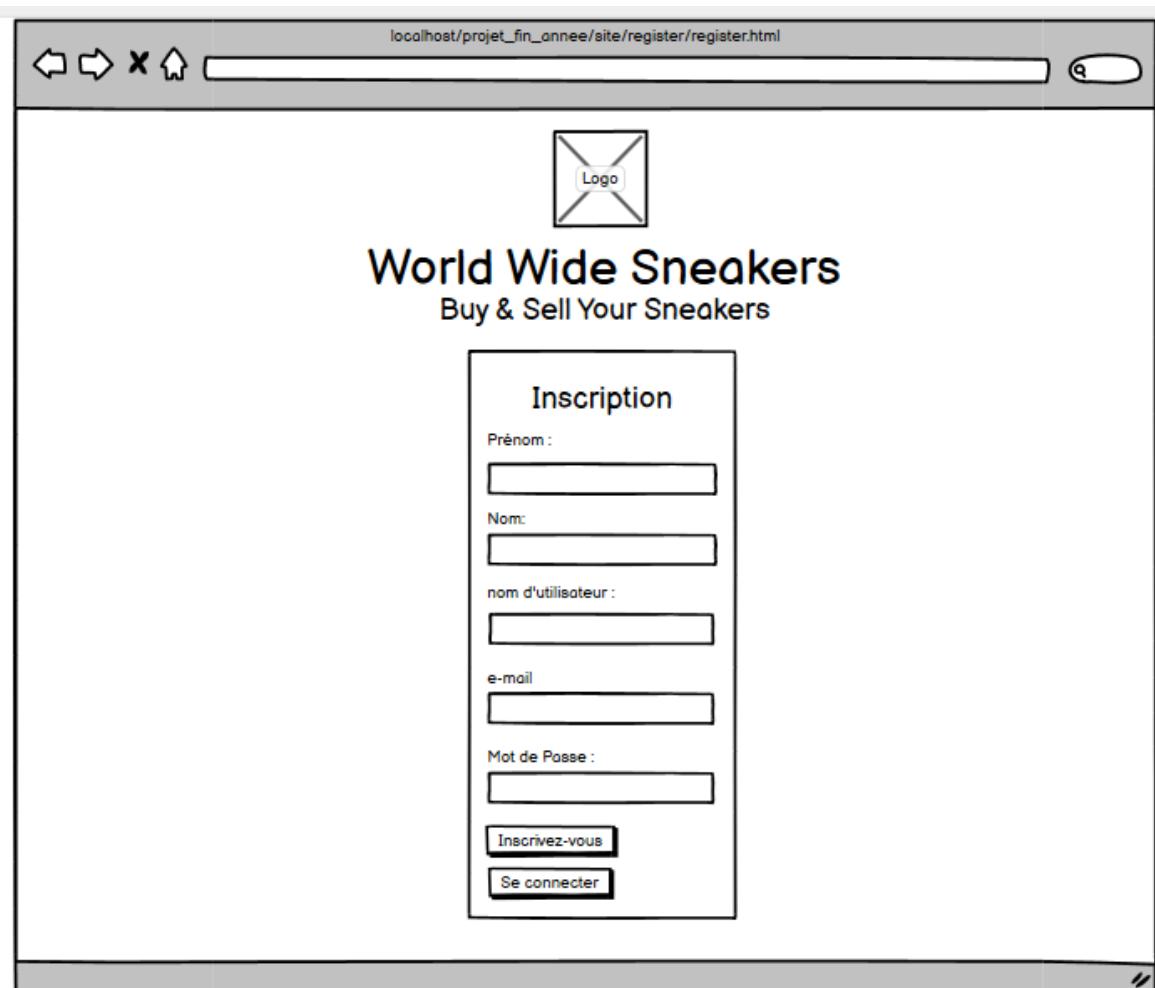
Image:

No file chosen

[Met en ligne ton article](#)

©WWS

Wireframe et Maquette page Inscription :





World Wide Web
Buy & Sell Your Sneakers

Inscription

Prenom :

Nom:

Nom d'utilisateur :

Email:

Mot de passe:

[Inscrivez-vous](#)

[Se connecter](#)

©WWS

Diagramme de cas d'utilisation :

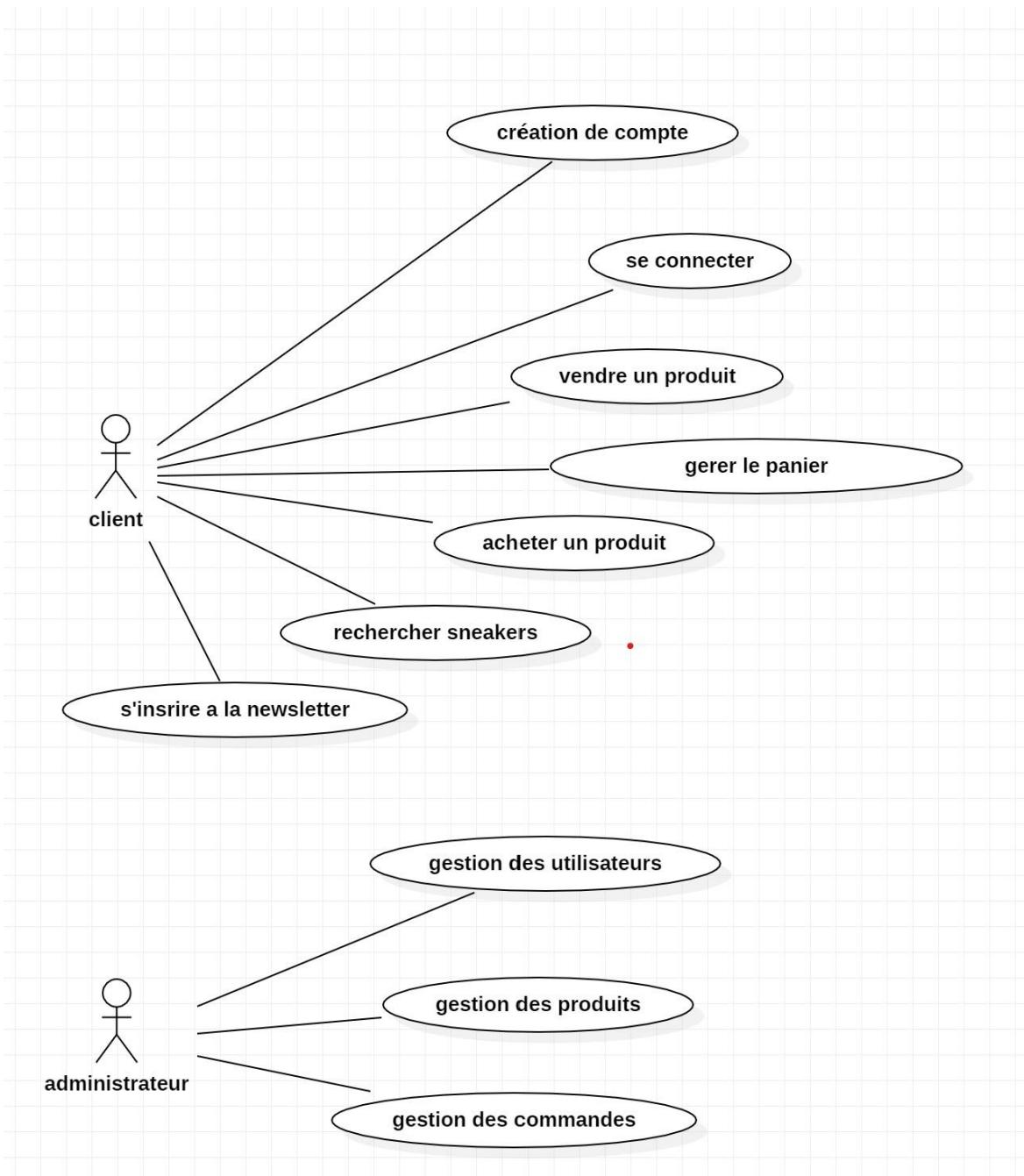


Diagramme de séquence :

