Dossier de Projet professionnel

Le dossier de projet 30-35 pages Hors annexes. Environ 48750 caractères, espaces non-compris.

Un résumé du projet 200-250 mots.

Le site que je vais vous présenter est une boutique en ligne portant sur la vente de hardware, soit la vente de matériel informatique visant à être utilisé pour monter des PC. Ce site présente beaucoup de caractéristiques techniques notamment dû aux nombreuses catégories, gammes, générations, fabricants et éditeurs que comporte chaque produit.
L'enjeu de ce site est d'optimiser la navigation et la recherche entre les articles afin que les utilisateurs puissent trouver leur bonheur et monter leur ordinateur facilement.
Ce projet a été effectué en un mois avec la collaboration de Cindy Gauthier.
Il s'agit d'un site utilisant les technologies HTML/CSS JAVASCRIPT PHP/SQL sous une architecture MVC.

Compétences du référentiel couvertes par le projet

Le projet couvre les compétences énoncées ci-dessous.

Pour l'activité 1, "Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité":

● Maquetter une application
● Réaliser une interface utilisateur web ou mobile statique et adaptable
● Développer une interface utilisateur web dynamique
● Réaliser une interface utilisateur avec une solution de gestion de contenu ou E-commerce

Pour l'activité 2, "Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.":

● Créer une base de données
● Développer les composants d'accès aux données
● Développer la partie back-end d'une application web ou web mobile
● Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Sommaire

Spécification fonctionnelles

Périmètre du projet

Le site sera réalisé en français et ce dernier devra être accessible sur différents supports, à savoir mobile, tablette et ordinateur.
Le site s'adresse à des particuliers voulant monter une nouvelle configuration pc.

Arborescence du site

L'arborescence du site se décline comme suit :
- Page d'accueil
- Page inscription
- Page connexion
- Page mon profil
- Page tous les produits
- Page produit sélectionné
- Page tous les éditeurs
- Page éditeur sélectionné
- Page toutes les marques
- Page marque sélectionnée
- Page mention légale
- Page panier
- Page paiement
- Page de confirmation de commande
- Page administration du back-office

Fiche produit

Chaque article lorsqu'il est cliqué possède :
- Au minimum 2 photos ainsi que des images pré-défini si jamais il y en a moins
- Un titre d'article
- Un résumé du produit
- Une description du produit
- Un bouton ajouter au panier
- Un bouton acheter directement
- Une date pour rappeler quand est-ce que la fiche produit a été mis en ligne
- Un espace commentaire

I- Description des fonctionnalités

1. Inscription

L'utilisateur peut s'inscrire en renseignant son login, son email et son mot de passe qui sera haché automatiquement en base de données.
(utilisation de password_hash($value ,PASSWORD_BCRYPT))

2. Authentification Google

L'authentification est possible grâce à des modules installés à l'aide de composer. Google permet de récupérer un token contenant toutes les informations de l'utilisateur via une connexion avec n'importe quel compte qu'il possède. Son pseudo est alors transmis à la base de données ainsi que son id_google lui permettant de se ré-authentifier quand il le souhaite tout en sauvegardant les modifications éventuelles qu'il a pu faire sur son profil comme la modification de ses informations personnelles ou voir la liste de ses achats…

3. Authentification Classique

L'authentification classique s'effectue après l'inscription et nécessite le pseudo ainsi que le mot de passe de l'utilisateur. Un message d'erreur est émis si l'utilisateur se trompe de mot de passe. Cette authentification profite à ceux qui ne veulent pas partager leurs informations avec Google.

4. Affichage articles

Les articles peuvent tous être affichés en même temps si jamais l'on clique sur produits dans le header
Sinon ils s'affichent par type lorsque l'on clique sur les différentes options dans menu ou bien par type+marque/gamme si l'on clique dans l'option déroulante

5. Bar de recherche

La bar de recherche fonctionne à la fois avec de l'auto complétion et avec une recherche par mot clé. Un dictionnaire de mot est sauvegardé dans une table mot clé, celle-ci permet à l'auto complétion de proposer des résultats, ensuite un LOCATE en SQL permet de rechercher n'importe quel mot étant contenu dans la table article, ainsi, si un article possède dans son nom ou dans sa description un mot clé, alors les articles correspondant à la recherche seront affichés.

6. Side-nav-bar

La side-nav-bar se trouve dans la page qui présente tous les articles elle permet de trier les articles d'un même type suivant leur génération et leur marque.
Exemple je recherche une carte graphique construite par nvidia et provenant de la génération RTX 2000
La side-nav-bar produit ainsi des résultats répondant aux exigences de l'utilisateur.

7. Panier Client

Le panier fait figurer tous les articles qui ont été ajoutés au panier. Mais avant de payer il faut renseigner une adresse. Celle-ci peut être renseignée directement depuis la page du panier mais mènera vers une redirection sur le profil afin d'être enregistré puis il suffit de sélectionner une adresse parmi celles qui ont déjà été renseignées par l'utilisateur pour pouvoir accéder au bouton de paiement qui deviendra vert et cliquable.

8. Solution de paiement

Pour payer il suffit de confirmer l'achat en ayant validé l'adresse. On accède alors vers la page de paiement où il faut renseigner le nom du titulaire de la carte, son numéro de carte bleu et son code de sécurité. Stripe prend en charge le paiement et il sera directement reçu sur l'interface de stripe que possède l'administrateur du site/gérant. Stripe est une interface applicative de programmation (API) qui permet d'effectuer des paiements en ligne. Cette application est assez répandue puisqu'elle est utilisée par des entreprises comme DELIVEROO ou ASSOS et prend 2.9% de gain sur chaque vente. Il existe d'autres moyens de paiement cependant celui là est facile à implémenter et fonctionnel. Une fois que l'achat est effectué l'utilisateur est redirigé vers une page de succès de paiement où il est remercié pour sa fidélité puis deux boutons sont à sa disposition, soit pour continuer ses achats soit pour voir son historique d'achat. Si la carte bancaire renseigné n'existe pas alors un message d'erreur sera affiché et l'utilisateur devra entrer un code de carte bancaire valide pour effectuer son achat

9. Profil

Le profil des utilisateurs permet de renseigner des informations complémentaires comme le nom, prénom, une photo de profil et la date d'anniversaire. D'ailleurs lors de la première connexion, chaque utilisateur se voit attribuer une photo par défaut.
On peut également compléter jusqu'à trois adresses avec numéro de téléphone, identité l'adresse complète et des informations complémentaires… Sans celles-ci aucun paiement ne pourra être effectué

10. Likes

Un utilisateur peut choisir de liker un article, au début le cœur est vide avec des contours rouges. Une fois cliqué il devient rempli et tout rouge. Dans le même temps, plusieurs requêtes peuvent avoir lieu. Tout d'abord l'ajout d'une ligne avec l'id de l'utilisateur dans une table like pour avoir la trace du like de chaque utilisateur, puis un téléchargement de la somme des likes d'un article pour actualiser ce total dans la ligne article correspondante. Enfin si l'utilisateur clique sur le cœur remplit alors cela va annuler le like et donc effectuer une suppression du like en base de donnée et donc une nouvelle somme avec un like en moins dans la ligne de l'article correspondant.

11. Back-office

Le back-office permet d'administrer complètement les articles. On peut y créer de nouvelles gammes/génération marques/éditeurs et type de produit tout en suivant les avancées technologiques. Lorsqu'un nouveau produit sort on peut directement l'ajouter à son catalogue. On peut également ajouter les nouveaux produits au header afin de faciliter la navigation et à la side-nav-bar qui permet de filtrer les produits. Tout est automatisé en termes de téléchargement et affichage des données. Le back-office permet aussi de modifier et supprimer toute donnée figurant sur la page. Ce qui donne l'accès à la modification des droits des utilisateurs et les commentaires peuvent être supprimés par l'administrateur directement sur les pages articles afin d'effectuer de la modération.

12. Déconnexion

Un utilisateur peut se déconnecter à tout moment via un bouton dédié dans le footer. Cela a pour effet de supprimer sa session en cours

II - Spécification technique

Travail en amont inventé

Pour débuter ce projet nous avons créé un Trello afin de se partager les tâches cependant on a rapidement permuté sur des réunions papier car cela nous convenait mieux. On a aussi communiqué sur discord et google chat.

La première étape du projet a été de faire des recherches pour évaluer le nombre de catégories et sous catégories qu'allait nécessiter notre site. Au final nous avons trouvé tout un tas de chiffre que nous voulions implémenter, soit:

- 132 articles
- 9 types d'articles
- comptant ~30 gammes
- 11 générations
- 12 marques
- 8 éditeurs

En sachant que cette liste peut être amenée à évoluer mais qu'il s'agit ici de ce que l'on estime être le minimum pour avoir un site représentatif de notre thématique et pour que nos fonctionnalités de recherches soient pertinentes.

Afin de lister correctement chaque article, son type, sa gamme, sa génération, sa marque et son éditeur; nous avons créé tout un tas de dossiers et de sous-dossiers sur un google drive partagé. Le google drive présente d'abord les types d'articles (carte graphique, disque de stockage…) puis les gammes et les générations (des fois l'ordre est inversé selon le type de produit). Chaque produit est ensuite rattaché à un constructeur (une marque) qui lui-même est souvent rattaché à un éditeur (modèle customisé). En bout de file on essaie le plus souvent de copier la description d'un article ainsi que son image afin de pouvoir l'importer sur la boutique en ligne facilement. On ne stocke pas les images on importe directement l'url de l'image. C'est une mauvaise solution sur le long terme car un bon nombre d'url risque d'être invalide, cependant cela nous a permis d'introduire un grand nombre d'articles sans faire peser lourd notre projet en stockant les images dans un dossier.

Par la suite, il a fallu réfléchir aux fonctionnalités que l'on souhaitait utiliser sur notre site. Qu'elles soient développées ou non, nous avons essayé de penser à un maximum de

fonctionnalités qui pourraient être implémentées au cours de plusieurs versions du site et dans le temps.

Explication sur les produits

Un ordinateur moderne est composé d'une tour, de plus ou moins grande taille, avec des disques de stockage pour pouvoir installer tout ce dont l'utilisateur aura besoin. Il faudra également de la mémoire vive (RAM) pour pouvoir transmettre l'information aux différents composants. Celle-ci sera fixée à la carte mère qui centralise tous les composants; le choix du processeur, qui permet de calculer tous les processus doit être compatible avec la carte mère. Pour faire fonctionner le tout, on aura besoin d'une alimentation (300-+1000w) qui pourra également alimenter une carte graphique qui risque d'être indispensable selon l'utilité que l'on a de l'ordinateur. L'achat d'un clavier, une souris ainsi et un OS (window) que nous ne proposons pas dans notre boutique sont plus qu'indispensable
Enfin il faudra certainement des systèmes de refroidissement pour préserver les composants , ceux-ci peuvent être à base de ventilation ou ventilation combinée à un système liquide de refroidissement.
Il ne manque désormais plus qu'écran afin de compléter totalement la configuration de l'ordinateur.

Les produits sont nombreux et complexes en termes de gamme de génération et de personnalisation des éditeurs. Cela a eu pour conséquence de créer un système de tri inspiré des sites internet déjà existant comme LDLC ou bien matériel.net pour faciliter la navigation des utilisateurs qui doivent trouver tout un tas de composants compatibles et au prix qu'ils souhaitent.

Exemple de produits

Exemple complexe

Les cartes graphique sont actuellement produites par deux marques, NVIDIA et AMD

Il existe 3 générations de carte chez NVIDIA soit les GTX 1000 RTX 2000 et RTX 3000
Chaque génération possède des gammes différentes allant généralement de 50 60 70 80
Exemple GTX 1050 .. RTX 2070 et peuvent être éditées par des partenaires comme MSI ou GIGABYTE…

Ce qui nous fait donc une RTX 2070 MSI soit 4 critères à respecter pour afficher correctement une carte graphique

Exemple simple

Les disques de stockage sont divisés en trois générations, les disques dur , les disques ssd et les disques nvme. Eux même sont divisés en une multitude de gammes de stockage allant du 128 Giga à 4 Tera de données

Pour afficher un disque de stockage on aura donc besoin de sa marque, de sa génération et de sa gamme

Choix techniques et environnement de travail
Technologies utilisées pour la partie back-end :
● Le projet estréalisé avec le langage PHP (Hypertext Preprocessor)
● Base de données SQL
● Gestionnaire de dépendance: Composer

Technologies utilisées pour la partie front-end:
● Le projet est réalisé avec du HTML et CSS.
● Et enfin Javascript afin de dynamiser le site et d'améliorer l'expérience utilisateur.

L'environnement de développement est le suivant:
➢ Editeur de code: Visual Studio Code
➢ Outil de versioning: GIT, Github.
➢ Maquettage: Figma

Utilisation de composer pour le fonctionnement de *google authentification*

Du point de vue de l'organisation, j'ai utilisé Trello et de nombreuses feuilles volantes afin de dynamiser ma réflexion et mon travail d'équipe en espace de co-working. Puis j'ai repris Trello pour compléter les fonctionnalités qu'il manquait à implémenter

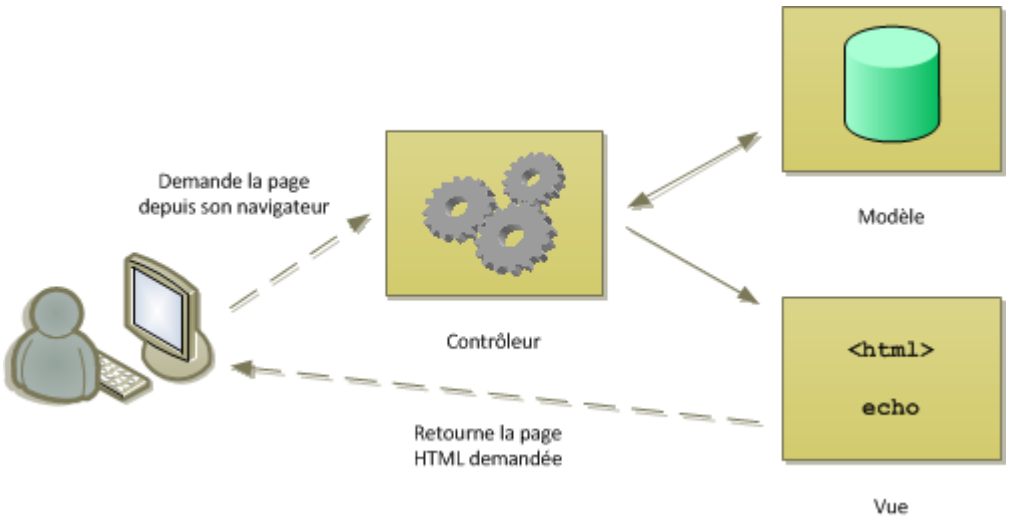
Architecture

Le projet comporte une architecture MVC (modèle vue contrôleur)

L'architecture MVC est l'une des architectures les plus utilisées pour les applications Web, elle se compose de 3 modules:
● Modèle: noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.

10

● Vue: composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur.
● Contrôleur: composant responsable des prises de décisions, gère la logique du code, il est l'intermédiaire entre le modèle et la vue.



Le fonctionnement de cette architecture est simple. Une vue, par exemple un formulaire de connexion, va envoyer les valeurs de son formulaire à un contrôleur. Le contrôleur est en général constitué de classes qui elles même possèdes des méthodes dans lesquelles transite les données envoyées. Une fois les données traitées, elles vont soit directement être renvoyées vers la vue soit le contrôleur va faire appel à un model pour envoyer ou vérifier les données en base de données. Par la suite le contrôleur renverra une réponse si tout s'est bien passé ou si le modèle lui a renvoyé une réponse.

On a donc 3 dossiers pour le MVC, les vues(envoie les données), les contrôleurs (traite les données et renvoie vers vues ou modèle) et les modèles (accès à la base de données) qui utilisent les données sécurisées par les contrôleurs.
Les dossiers contrôleur et modèle sont constitués de fichier contenant a chaque fois une classe tandis que les vues sont des pages simple qui sont appelé selon où l'on clique sur le site et qui sont paramétrés comme dit plus haut.

Tous les contrôleurs héritent d'une classe abstraite nommée Contrôleur et tous les modèles héritent également d'une classe abstraite nommée Model. Ces classes abstraites possèdent des méthodes (souvent génériques) qui sont utilisables par tous les enfants ce qui facilite grandement le développement du projet lorsqu'il faut ajouter rapidement de nouvelles fonctionnalités.

Réalisation (charte graphique, maquette, base de données)

Extrait de code significatif

Système de tri:
Ici on récupère tous les GET provenant de cette page et on l'envoie dans transitOneColumn

```php
if(isset($_GET['search_GC'])){
    $controllerDisplay = new \Controller\DisplayArticle();
    $result = $controllerDisplay->transitOneColumn($_GET);
```

On exclus l'input search de $_GET et on appel une fonction model une fonction display du controller

```php
class DisplayArticle extends Controller{
    public function transitOneColumn($tab_form){
        $tab = array();

        foreach($tab_form as $value){
            array_push($tab, $value);
            $key = array_search($tab_form['search_GC'], $tab);
            if (false !== $key) { // permet d'enlever la valeur de $_GET['search_GC'] qui renvoit les valeurs mal configurés
                unset($tab[$key]);
            }
        }

        $checkBoxSearch = new \Model\Display();
        $result = $checkBoxSearch->selectCheckBox($tab);
        if($result != "Ne correspond à aucun élément , il ne doit plus y avoir de stock ou l'article n'existe plus"){
            $this->displayArticlesByTab($result);
        }
        elseif($result == "Ne correspond à aucun élément , il ne doit plus y avoir de stock ou l'article n'existe plus"){
            echo "Ne correspond à aucun élément , il ne doit plus y avoir de stock ou l'article n'existe plus";
        }

        return $result;
    }
```

La fonction model va préparer une requête PDO en remplaçant les valeur du tableau par des ‘’?,’’

```php
public function selectCheckBox($tab){
    if($tab != null){

        $in  = str_repeat('?,', count($tab) - 1) . '?';
        $sql = "SELECT * FROM articles WHERE id_generation IN ($in)";
        $result = $this->pdo->prepare($sql);
        $result->execute($tab);
        $data = $result->fetchAll();

        return $data;
    }
    else{
        return $data = "Ne correspond à aucun élément , il ne doit plus y avoir de stock ou l'article n'existe plus";
    }
}
```

Les data vont être fetch pour repasser dans le controller du dessous dans la fonction displayArticlesByTab($tab)

Cette fonction sera elle même chargé d'afficher un rendu sur la vue
Les modulo de $temp permettent de faire un affichage des articles 3 par 3

```php
public function displayArticlesByTab($tab){

    $i = 0;

    $temp = 1;
    echo '<section class="rowSection">';
    foreach($tab as $value){
        if($temp == 4){
            $temp = 1;
        }
        echo "<form action='article.php' method='get' class='form_article'>";
        echo "<button type='submit' class='buttom_comp' name='articleSelected' value= '".$tab[$i]['id']."'>";
        echo "<p class='typo_comp'>". $tab[$i]['titre'] ."<u></u></p>";
        echo "<img src='".$tab[$i]['image']."' class='dimension_image'>";
        echo $tab[$i]['prix']."€<br />";
        echo "</button>";
        echo "</form>";

        if($temp % 3 === 0 ){
            echo '</section>';
            echo '<section  class="rowSection">';
        }
        $temp++;
        $i++;
    }
}
```

Header & footer

Le header et le footer posent les bases de l'identité de notre site. Après quelques recherches nous avons décidé de faire un header sobre et un footer assez large afin d'y faire figurer les informations complémentaires liées à notre site. On peut y trouver les différents modes de paiement, livraison, S.A.V, partenaires, et réseaux sociaux éventuels car tout cela dépend au final d'une activité sérieuse et professionnelle que nous avons essayé de simuler par tous les moyens.

Le header contient deux boutons déroulant pour naviguer vers les articles ou bien vers des pages de présentation des différents constructeurs/éditeurs. Il possède également un champ de recherche qui utilise l'autocomplétion et la recherche par mot clé. Enfin un bouton est dédié aux admin pour accéder au back-office mais reste invisible pour les utilisateurs lambda et deux boutons communs à tous permettent d'accéder au profil et au panier.

Le header quant à lui est quasiment entièrement affiché à l'aide de PHP. Beaucoup de variables permettent de prendre pour valeur le résultat de fonctions qui récupèrent des données en base de données. Cela permet d'effectuer une navigation en temps réel, si jamais un type d'article est supprimé via le back-office alors elle ne figurera plus dans la bar de navigation du header. Le header vérifie si un utilisateur est connecté pour lui afficher les boutons profil/panier et s'il est admin pour lui donner le chemin du back-office de l'admin. Il comporte également tout un tas de variables qui sont définies ailleurs dans les vues pour définir des chemins. Par exemple des variables peuvent être définies comme le css/le nom de la page, les liens vers les autres pages, puis une fois que le header est appelé, les variable qui sont défini dans les vues vont initialiser les variables du header pour faire les bon require (du css / nom de page / accès au autres pages etc.). Tout cela représente à peu près 30 lignes par page sans compter les require aux différents Model et Controller.

SQL

La base de données a été pensée par rapport aux fonctionnalités du site. J'utilise 15 tables.
4 tables qui concernent uniquement les produits
(Articles, Type, Gamme, Génération)
2 tables concernant les partenaires
(marque, editeur)
9 tables qui concernent les fonctionnalités & utilisateurs
(carte bleu, liste de souhait, commande, panier, commentaire, likes, auto-complétion, adresse, utilisateur)

J'utilise le moteur de stockage MyISAM qui est la par défaut

Mettre MCD, MLD

Les méthodes génériques.
Il va en exister pour toute situation qui peut se répéter dans différent contexte, cela peut donc être pour du front comme pour du back, et parfois les méthodes génériques sont elles même constituées de méthode générique

Par exemple la méthode générique verifyAndInsertOne() utilise deux méthodes génériques du model qui sont alreadyTakenCheck() pour savoir si une données est déjà utilisé par un autre utilisateur (par exemple un pseudo) et insertOneValue() pour insérer une nouvelle valeur dans la base de données. Cette fonction générique du contrôleur me permet ainsi de vérifier une valeur avant de l'insérer dans la base de donnée. Cette fonction existe sous plusieurs formats: verifyAndInsertTwo() verifyAndInsertThree() suivant le nombre de valeur que je souhaite insérer et si la fonction est utilisé plusieurs fois car le but de ces fonctions génériques est d'avoir une refactorisation des fonctions pour que le code soit plus claire et plus facile à utiliser comme à s'approprier.

Le contrôleur possède aussi des méthodes qui ne sont pas génériques mais inclassables donc pratiques à partager à toutes les autres classes. On y retrouve notamment une méthode qui va sécuriser les champs envoyés depuis les vues grâce à la méthode php htmlSpecialChars() (efficace contre les injections XSS) et deux méthodes qui vont limiter le nombre de caractères dans une chaîne de caractères par exemple pour afficher un résumé. Cette classe possède également une constante qui a pour valeur une image. Cette constante est utilisée lors de l'inscription des utilisateurs pour leur procurer une image par défaut puis ils peuvent la changer en allant dans leur profil.

La classe modèle quant à elle utilise une fonction externe dans son constructeur pour se connecter à la base de données, la fonction aurait directement pu être codée dans la classe mais pour un soucis de compréhension elle figure à l'extérieur dans un fichier nommé bdd.php.

Cette fonction permet d'initialiser une connexio*cteur exécuté, une propriété nommée $pdo va prendre le résultat de la fonction connect(). Cela va permettre d'utiliser $this->pdo à chaque fois que l'on aura besoin d'exécuter une connexion à la base de donnée.

La classe Model possède beaucoup d'autres fonctions, la plupart d'entre elles sont génériques.
Par exemple il y a 9 fonctions SELECT dont certaines pourraient être refactorisées mais si les déclinaisons existent c'est pour faciliter la lecture du code. Par exemple selectOneValue() et alreadyTakenCheck sont sensiblement les mêmes fonctions mais ne renvoie pas exactement le même résultat et lorsque la fonction se nomme alreadyTakenCheck() il devient plus facile de réfléchir au code avec des noms qui sont évoqués. On s'attend plus facilement au résultat que l'on pourra également conditionner plus facilement…

D'autres SELECT peuvent être similaires et c'est le fetch qui sera amené a changé, on peut avoir deux fonctions qui font le même travail mais l'une est en FETCH_ASSOC la deuxième en fetchAll. Tout dépend en fait du résultat attendu et de comment on souhaite le mettre en forme, donc pour un souci d'accessibilité certaines fonctions diffèrent.

Il existe 4 fonctions générique d'insertions allant d'une à quatre valeurs à insérer dans la base de données.

Une fonction générique de suppression deleteOneWhereId
3 fonction générique d'update allant de 1 à 4 valeurs dont une quatrième qui insère avec le typage INT sur toutes ses valeurs

Et enfin il existe une fonction qui permet d'effectuer des recherches par mot clé à l'aide de la fonction LOCATE en SQL pour ne citer que les fonctions ayant de l'importance dans model.

Un autre fichier model nommé Display.php contient une méthode de recherche par checkBox.
Il s'agit de selectCheckBox() (mettre annexe)
Cette méthode permet d'utiliser un tableau de variable et de l'insérer dans une requete sql en PDO. Pour ce faire, il faut utiliser une fonction str_replace qui va remplacer chaque valeur du tableau de variable par un "?,". Le nouveau tableau va alors pouvoir être inséré dans la requete SQL que cela ne créer de problème puis lorsque la requête est passé a ->pdo->prepare il suffira de mettre l'ancien tableau de variable dans le execute() pour que PDO reconnaisse a quelle variable correspond chaque "?," de la requête. Par la suite le resultat de la requête est fetch et est renvoyé vers le controleur qui lui même va le renvoyer vers une vue dans laquelle sera affiché les articles triés comme le voulait l'utilisateur.

La classe Article (Model) permet de fetch les articles les plus consultés, mais pour cela il faut utiliser la fonction qui créer des vues qui est contenue dans la même classe. Lorsqu'un utilisateur visite une page qui contient un article alors il va insérer une vue dans la table vues en SQL. Dès lors, un Count et un fetch et de la même fonction vont faire la somme des toutes les vues contenues dans la table vues pour émettre une somme de vues dans la table article afin que l'on puisse savoir quels sont les articles les plus vues pour ensuite pouvoir les fetch.

La classe Admin (Model) permet de modifier et d'insérer quand ce n'est pas générique toute chose dans le back-office, marque , éditeur , articles, type, utilisateur. Ainsi l'administrateur a la main sur tout ce qu'il peut se passer sur le site. Il peut également créer de nouvelles gammes/catégories d'articles sans encombre, au gré des nouvelles technologies.

Controleur :

La classe Display est la plus grande classe de ce projet, elle fait +1000 lignes et affiche toutes les données contenues dans le back-office. On va avoir l'ajout de type, gamme, génération, marque et éditeur avec l'ajout d'articles suivant les type, gamme, génération, marque et éditeur disponible. Ensuite l'administrateur pourra accéder aux données disponible et afficher notamment les différents articles, types, gammes , générations , marques , éditeurs et utilisateurs

Le dossier config contient bdd.php mais également deux autres fichier:

Le fichier utils.php qui est inclus dans toutes les vues comprend un session_start() qui est initialisé sur toutes les vues avec une vérification pour savoir s'il n'est pas déjà exécuté.*

Le fichier http.php contient une classe Http et une méthode qui permet de faire des redirections, le fait de faire les choses sous forme de fonction permet de simplifier l'écriture des redirections.

Lorsque que les redirections entrent en conflit car il y en a plusieurs qui peuvent être appelés sur la même page alors cela crée une erreur. Si l'on met son code entre un ob_start() et un ob_end_flush() alors le code sera lu différemment et l'ordinateur pourra prendre une décision entre les deux redirection pour savoir quelle page appeler

I - Failles de sécurité informatique

a - XSS

Le cross-site scripting (abrégé XSS) est un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, provoquant ainsi des actions sur les navigateurs web visitant la page. Les possibilités des XSS sont très larges puisque l'attaquant peut utiliser tous les langages pris en charge par le navigateur (JavaScript, Java...) et de nouvelles possibilités sont régulièrement découvertes notamment avec l'arrivée de nouvelles technologies comme HTML5. Il est par exemple possible de rediriger vers un autre site pour de l'hameçonnage ou encore de voler la session en récupérant les cookies.

Le principe est d'injecter des données arbitraires dans un site web, par exemple en déposant un message dans un forum, ou par des paramètres d'URL. Si ces données arrivent telles quelles dans la page web transmise au navigateur (par les paramètres d'URL, un message posté…) sans avoir été vérifiées, alors il existe une faille : on peut s'en servir pour faire exécuter du code malveillant en langage de script (du JavaScript le plus souvent) par le navigateur web qui consulte cette page.

La détection de la présence d'une faille XSS peut se faire par exemple en entrant un script Javascript dans un champ de formulaire ou dans une URL :

```
<script>alert('bonjour')</script>
```

Si une boîte de dialogue apparaît, on peut en conclure que l'application Web est sensible aux attaques de type XSS.

Basé sur le DOM ou local

Le problème est dans le script d'une page côté client.

XSS stocké (ou permanent)
Elle se produit quand les données fournies par un utilisateur sont stockées sur un serveur (dans une base de données, des fichiers, ou autre), et ensuite ré-affichées sans que les caractères spéciaux HTML aient été encodés.

XSS réfléchi (ou non permanent)
Il apparaît lorsque des données fournies par un client web sont utilisées telles quelles par les scripts du serveur pour produire une page de résultats. Si les données non vérifiées sont incluses dans la page de résultat sans encodage des entités HTML, elles pourront être utilisées pour injecter du code dans la page dynamique reçue par le navigateur client.

Solution
En PHP :
utiliser la fonction htmlspecialchars() qui filtre les '<' et '>' (cf. ci-dessus) ;
utiliser la fonction htmlentities() qui est identique à htmlspecialchars() sauf qu'elle filtre tous les caractères équivalents au codage HTML ou JavaScript.

b - Injection SQL:

Est un groupe de méthodes d'exploitation de faille de sécurité d'une application interagissant avec une base de données. Elle permet d'injecter dans la requête SQL en cours un morceau de requête non prévu par le système et pouvant en compromettre la sécurité.

La méthode *blind based:*

permet de détourner la requête SQL en cours sur le système et d'injecter des morceaux qui vont retourner caractère par caractère ce que l'attaquant cherche à extraire de la base de données. La méthode blind based, ainsi que la time based, se basent sur la réponse du serveur : si la requête d'origine renvoie bien le même résultat qu'à l'origine (et indique donc que le caractère est valide) ou ne renvoie pas le même résultat (et indique donc que le caractère testé n'est pas le bon). La time based a pour seule différence qu'elle se base sur le temps de réponse du serveur plutôt que sur la réponse en elle-même ;

error based:

Cette méthode profite d'une faiblesse des systèmes de base de données permettant de détourner un message d'erreur généré par le système de base de données et préalablement volontairement provoquée par l'injection SQL pour lui faire retourner une valeur précise récupérée en base de données

union based:

Cette méthode profite de certaines méthodes afin de détourner entièrement le retour de la requête SQL d'origine afin de lui faire retourner en une seule requête un important volume de données, directement récupéré en base de données. Dans ses exemples les plus violents, il est possible de récupérer des tables entières de base de données en une ou deux requêtes, même si en général cette méthode retourne entre 10 et 100 lignes de la base de données par requête SQL détournée

Stacked queries:

la plus dangereuse de toutes. Profitant d'une erreur de configuration du serveur de base de données, cette méthode permet d'exécuter n'importe quelle requête SQL sur le système ciblé, ce qui ne se limite pas seulement à récupérer des données comme les 3 précédentes. En effet, quand ce type de requête n'est pas désactivé, il suffit d'injecter une autre requête SQL, et elle sera exécutée sans problème, qu'elle aille chercher des données, ou en modifier directement dans la base de données.

Solution :

En PHP on peut utiliser pour cela la fonction mysqli_real_escape_string, qui transformera la chaîne ' -- en \' -- et ainsi bloque toute injection de code a la suite de la requete

Le fait d'utiliser une version de php récente permet aussi d'éviter les failles via des fonctions obsolètes

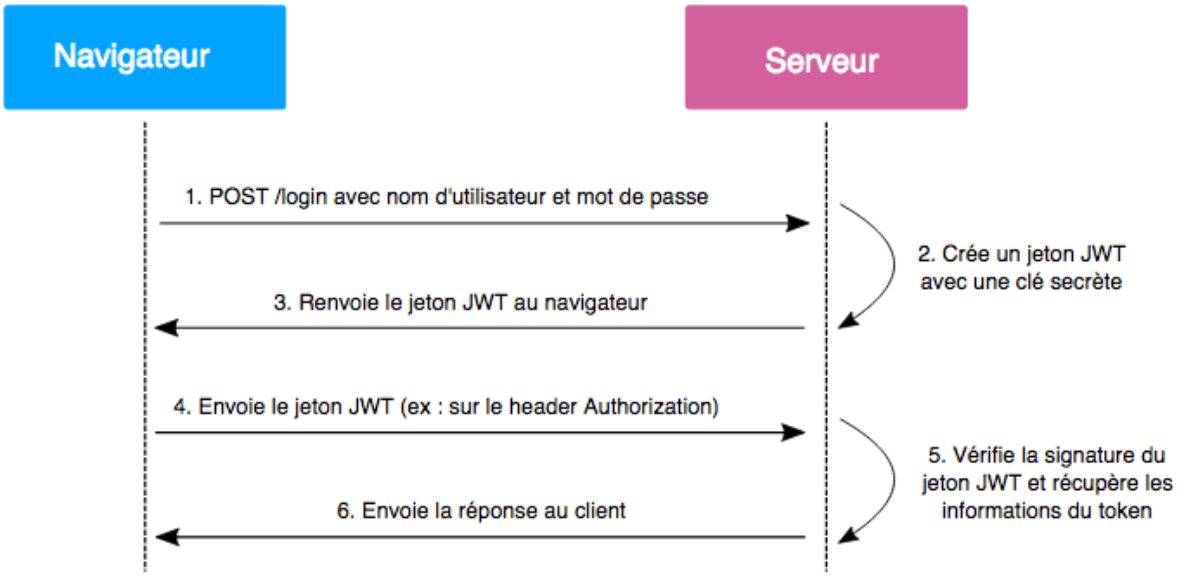
On peut aussi utiliser des requêtes préparées avec l'objet PDO : dans ce cas, une compilation de la requête est réalisée avant d'y insérer les paramètres et de l'exécuter, ce qui empêche un éventuel code inséré dans les paramètres d'être interprété. (solution a privilégier)

magic_quotes_gpc=on

Ce paramètre, lorsqu'il est mis sur *on*, ajoute le caractère "\" devant les apostrophes, les guillemets et le caractère nul. Il empêche ainsi le système d'interpréter une requête qu'un utilisateur mal intentionné aurait saisi dans un formulaire HTML. Cette variable tend aujourd'hui à ne plus être utilisée, au profit de la fonction *addslashes()* pour les requêtes SQL.

sécuriser les réponses du server à l'aide de Token

https://www.youtube.com/watch?v=tx93JqxFSeE&t=3s



c - faille RFI

Remote File Inclusion (RFI) est un type de vulnérabilité trouvé le plus souvent sur des sites web. Il permet à un attaquant d'inclure un fichier distant, généralement par le biais d'un script sur le serveur web. La vulnérabilité est due à l'utilisation de l'entrée fournie par l'utilisateur sans validation adéquate. Elle peut conduire à :

- L'exécution de code sur le serveur web
- L'exécution de code sur le côté client comme le JavaScript qui peut conduire à d'autres attaques comme Cross-site scripting (XSS)
- Déni de service (DoS)
- Le vol de données / Manipulation

II - Bonnes pratiques pour sécuriser une application

a/ Fichier de configuration PHP (php.ini)

chemin : */etc/phpX/apache/php.ini*
X représente la version de PHP utilisée

- Mettre la variable register_globals en off

Quand *register_globals* est à *on*, les variables EGPCS (Environnement, GET, POST, Cookie, Server) sont enregistrées comme des variables globales. Ainsi, une URL contenant une variable, par exemple *http://www.site.com/index.php?var=test* crée une variable *$var* sans besoin d'autre code.

- Limiter l'accès aux répertoires

Le *safe mode* est le mode de sécurité de PHP. Lorsqu'il est activé, il empêche un script d'accéder à des fichiers situés en dehors du dossier où se trouve le site. La vérification est faite en vérifiant le nom du propriétaire du fichier et celui du script
Parfois trop restrictive, une vérification sur le nom du groupe peut suffire, en utilisant la directive *safe_mod_gid*. La variable *Open_basedir* limite les manipulations aux fichiers situés dans les dossiers spécifiés. Il est également possible de désactiver individuellement des fonctions avec *disable_functions*. *Disable_classes* fonctionne de la même manière et permet de désactiver individuellement des classes.

- display_error=off et log_errors=on

Désactiver l'affichage des erreurs pour éviter d'afficher des informations aux utilisateurs en mettant la variable *display_error* en *off*. Il vaut mieux les inscrire dans un journal d'erreurs avec *log_errors=on*.

- Changer le répertoire temporaire des identifiants de session

Les identifiants de session sont enregistrés dans un répertoire temporaire. Par défaut, le paramètre *session.save_path* vaut */tmp* est accessible en lecture à tous. Il est donc plus sécurisé d'indiquer le chemin d'un répertoire situé ailleurs sur le système, et dont les droits auront été limités. De plus, il existe une fonction en PHP pour cripter les mots de passe enregistrés. Cette modification peut aussi se faire à partir du fichier de configuration d'Apache, à l'aide de la variable *php_value session.save_path*.

- Activer session.use_only_cookies

Cette variable indique si les identifiants de session doivent être utilisés seulement avec des cookies. Par défaut, cette variable est à *0*, elle est désactivée et autorise d'autres modes de lecture, par exemple avec les éléments *GET* ou *POST* des requêtes HTTP. En mettant *session.use_only_cookies* à *1*, le système lit les informations d'identifiant uniquement à partir des cookies.

b/ Bonnes pratiques : programmation défensive et environnement réseau

- L'injection de données

Pour éviter que des données non voulues soient injectées dans le code (requêtes SQL, commandes Shell, cross site scripting), il faut vérifier chaque donnée avant de les passer en

paramètres à des fonctions du système. Il s'agit de tester si les variables sont bien du type attendu. Il est aussi conseillé d'initialiser chaque variable.

- Journaliser toutes les erreurs

Le journal d'erreur est un bon indicateur pour repérer les attaques. Pour enregistrer toutes les erreurs d'exécution, ajouter la ligne *<?php error_reporting(E_ALL); ?>* au début de chaque page de code.

- Des outils comme les reverse proxy

Permettent d'écarter certaines requêtes faites sur le site (par exemple une requête demandant un chemin qui n'existe pas).

- Utiliser un pare-feu

Bloque les connexions sortantes depuis le serveur Web, afin d'éviter l'inclusion de fichiers PHP distants (php include).

- Utiliser un serveur dédié

Attention aux paramètres de sécurité des serveurs mutualisés. Pour une application manipulant des données sensibles nécessitant un haut niveau de sécurité, un serveur dédié sera plus adapté car la configuration sera totalement personnalisable.

C/ Utiliser un scanner de vulnérabilités pour vérifier la sécurité de son code

En sécurité informatique, un scanner de vulnérabilités est un programme conçu pour identifier des vulnérabilités dans une application, un système d'exploitation ou un réseau.

exemple de logiciel/Machine Virtuel:

- Dyn

Attaque Dyn DDoS - a provoqué la fermeture de nombreux sites Web, notamment Netflix, SoundCloud, Spotify, Twitter, PayPal, Reddit

- Acunetix
- Netspaker

… https://geekflare.com/fr/saas-web-vulnerability-scanner/

III - Documentation

a/ Sources

Journal du net
https://www.journaldunet.com/web-tech/developpeur/1007036-securiser-une-application-web-developpee-en-php/

ANSSI : https://www.ssi.gouv.fr/guide/recommandations-pour-la-securisation-des-sites-web/

ISO : https://www.iso.org/fr/isoiec-27001-information-security.html

b/ Actualités à suivre

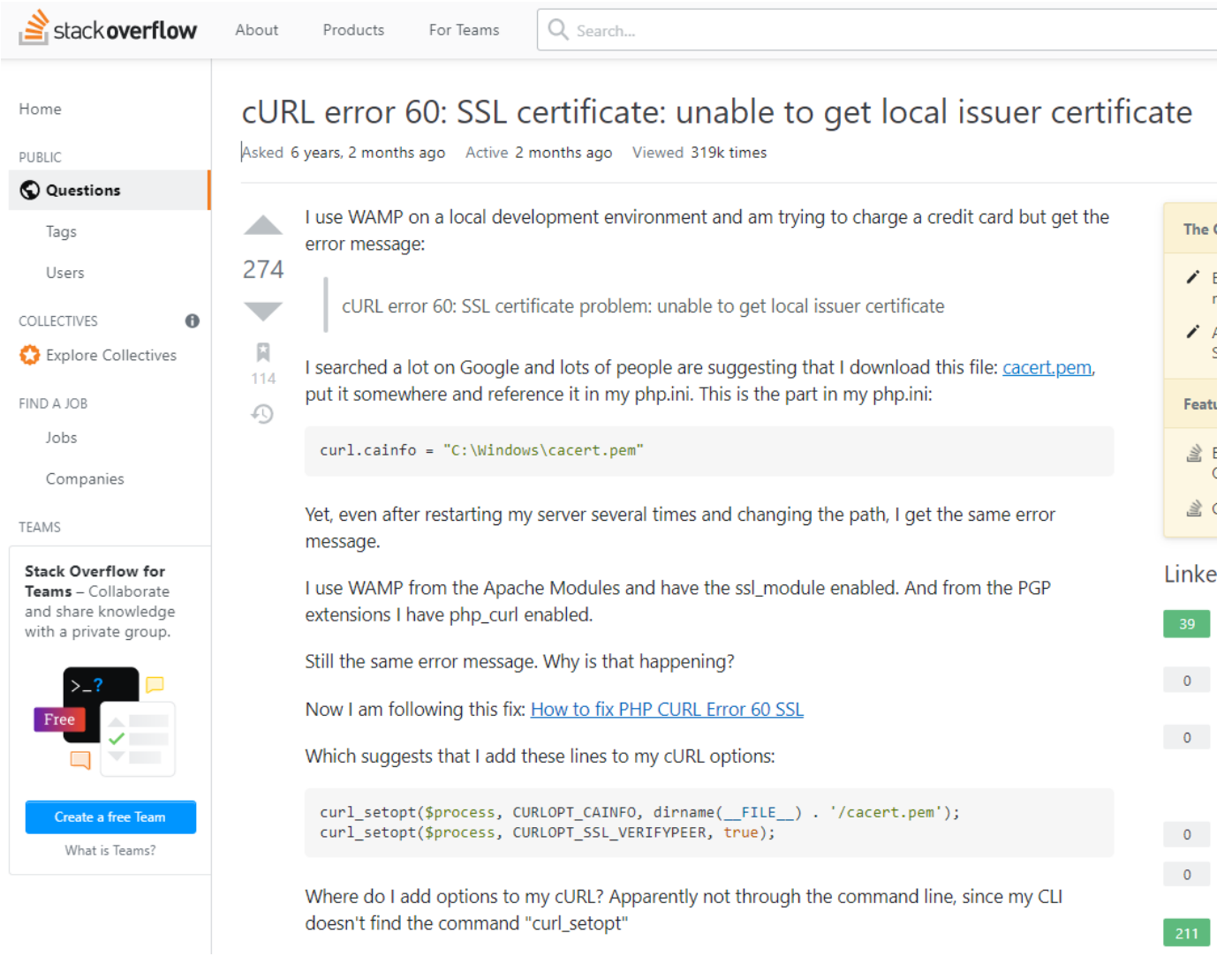
IT WATCH : https://sites.google.com/laplateforme.io/itwatch/lactu?authuser=1
CERT-FR : https://www.cert.ssi.gouv.fr/

c/ Pour aller plus loin

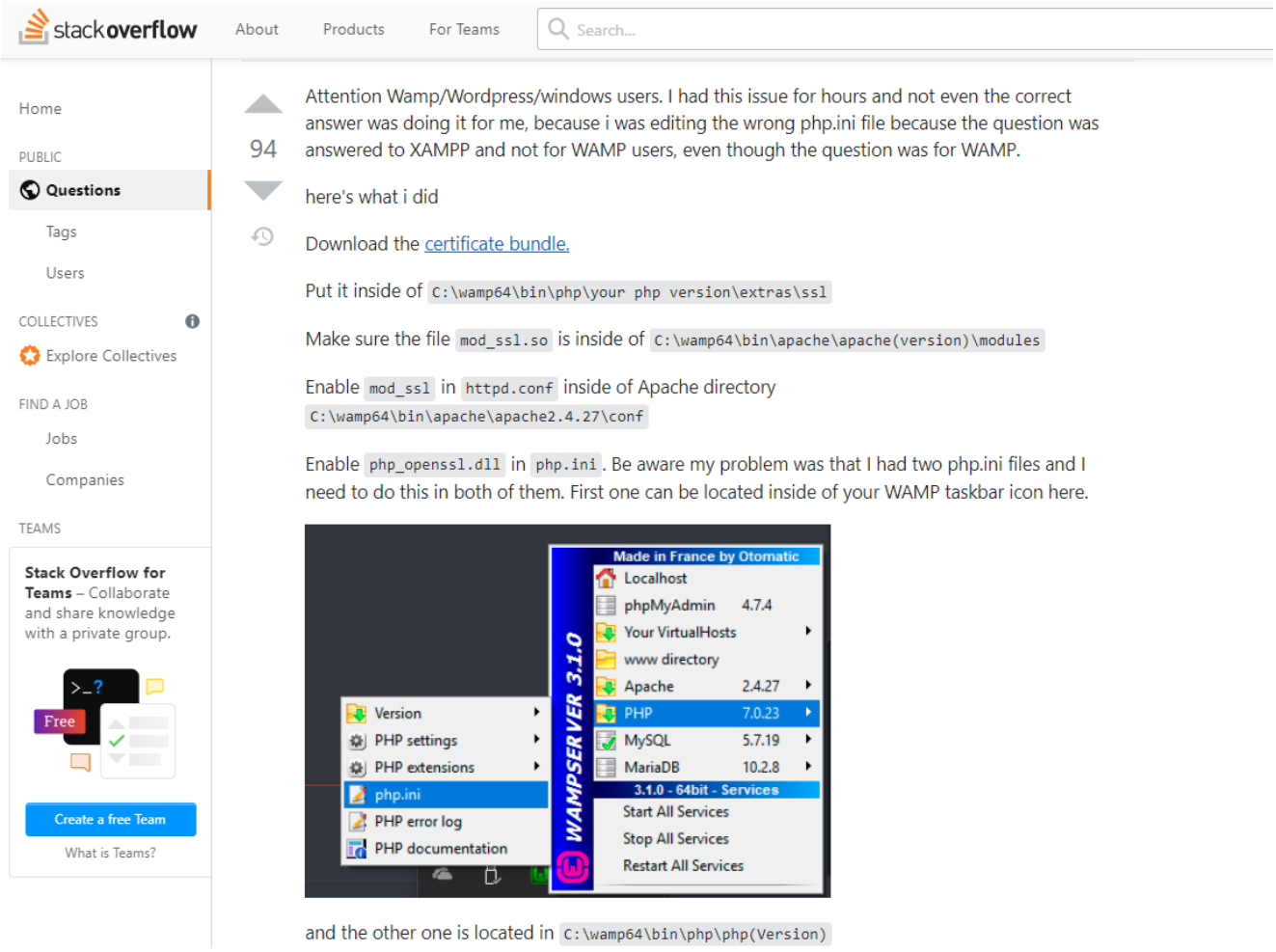
La veille technologique en sécurité
référence mondiale
Common Vulnerabilities and Exposures: https://cve.mitre.org/

Mettre en place les RGPD
https://www.cnil.fr/fr/rgpd-par-ou-commencer

Recherche en anglais

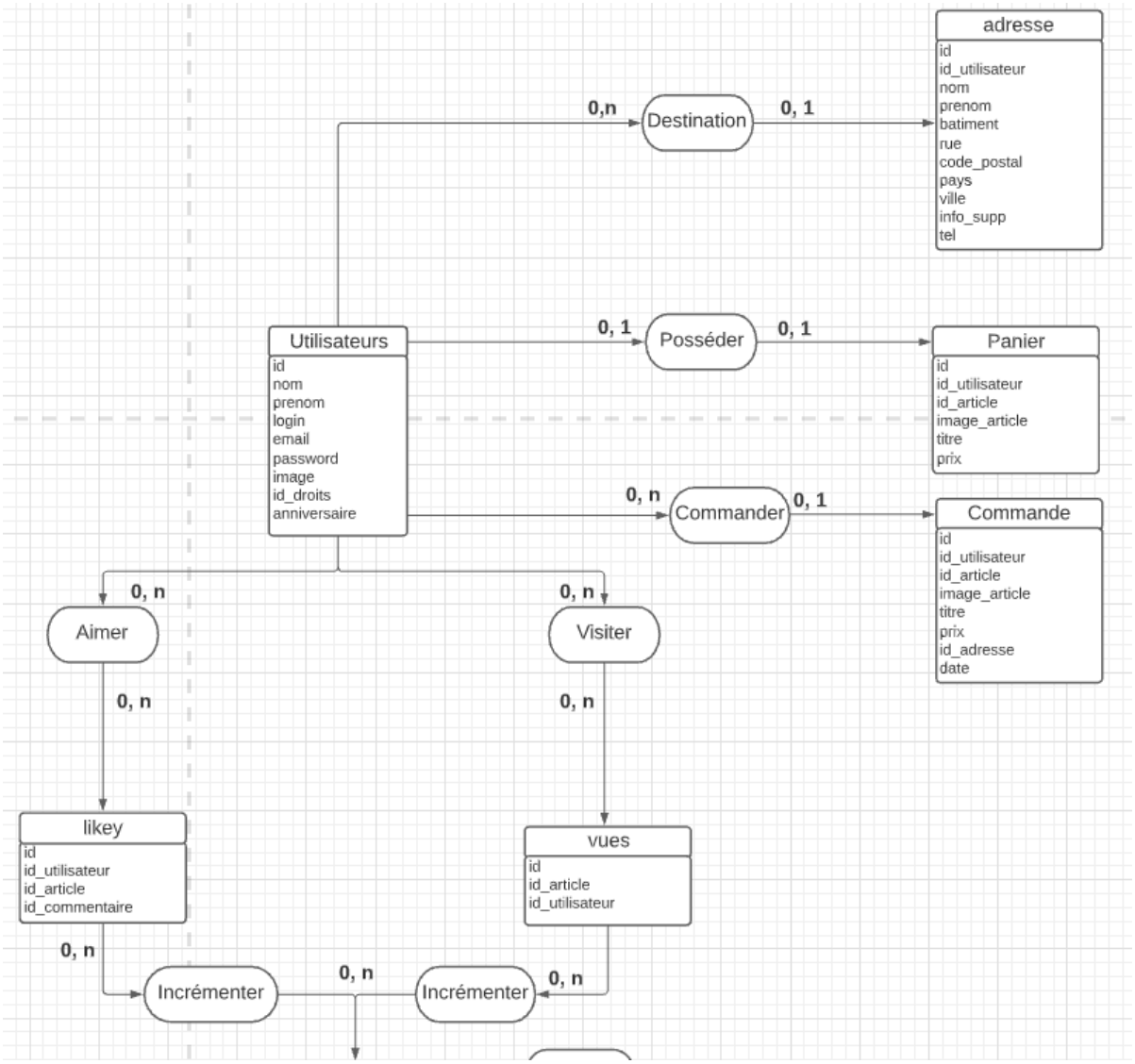


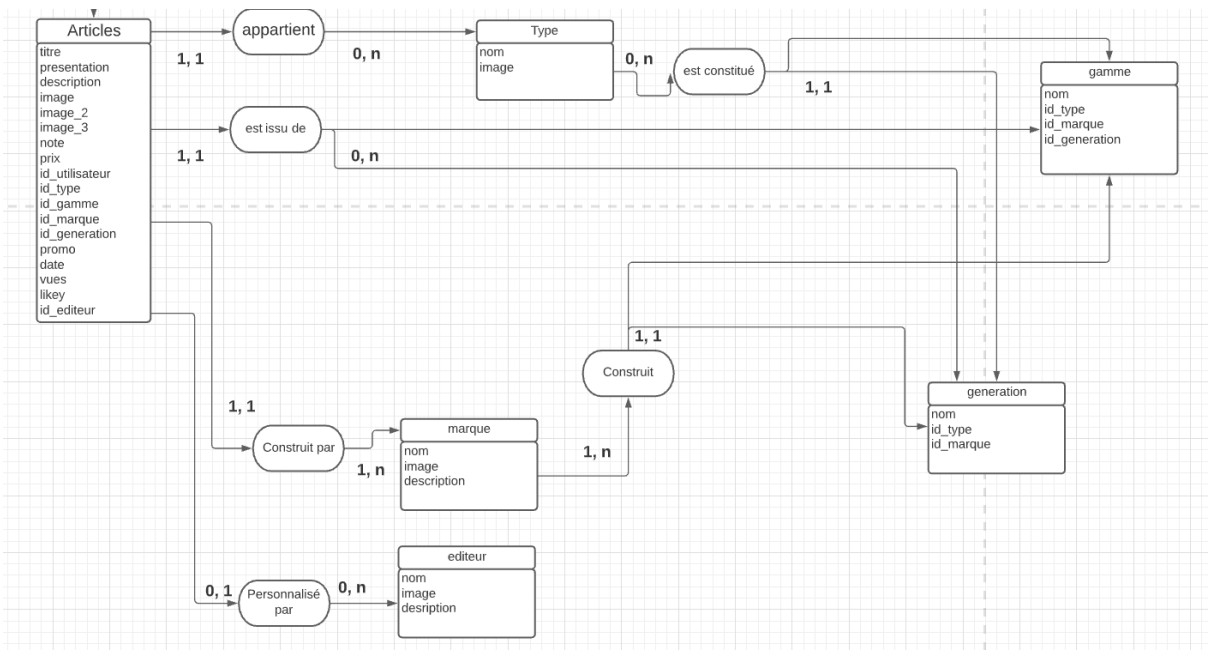
J'ai effectué cette recherche car l'authentification google nécessitait un certificat SSL pour donner l'accès à ses données (token). J'ai alors cherché dans ma config wamp ce qui n'allait pas et je suis tombé sur une explication avec les mod_ssl et l'importation d'un certificat standard cacert.pem



Dans l'explication suivante il montre ce qu'il faut faire pour que notre wamp reconnaisse le module ssl, j'ai déjà installé le fichier cacert.pem dans mon dossier modules. Puis je suis allé décommenter certaines lignes contenant le chemin d'accès au cacert.pem et enfin j'ai activé les mod_ssl de mon wamp afin que mon serveur apache puisse effectivement reconnaître les certificats SLL. Cela m'a donc permis d'avoir une url sécurisée et d'accéder aux données de google Oauth.

Maquette de boutique en ligne


adresse
id
id_utilisateur
nom
prenom
batiment
rue
code_postal
pays
ville
info_supp
tel
0,n
Destination
0, 1
Utilisateurs
id
nom
prenom
login
email
password
image
id_droits
anniversaire
0, 1
Posséder
0, 1
Panier
id
id_utilisateur
id_article
image_article
titre
prix
0, n
Commander
0, 1
Commande
id
id_utilisateur
id_article
image_article
titre
prix
id_adresse
date
0, n
Aimer
0, n
Visiter
0, n
0, n
likey
id
id_utilisateur
id_article
id_commentaire
vues
id
id_article
id_utilisateur
0, n
Incrémenter
0, n
Incrémenter
0, n


Articles
titre
presentation
description
image
image_2
image_3
note
prix
id_utilisateur
id_type
id_gamme
id_marque
id_generation
promo
date
vues
likey
id_editeur
appartient
1, 1
0, n
Type
nom
image
0, n
est constitué
1, 1
gamme
nom
id_type
id_marque
id_generation
est issu de
1, 1
0, n
1, 1
Construit
generation
nom
id_type
id_marque
1, 1
Construit par
1, n
marque
nom
image
description
1, n
0, 1
Personnalisé par
0, n
editeur
nom
image
desription

Annexes :
Maquettes

MCD MLD
Trello
Composer + google Oauth
Arborescence