Dossier de projet professionnel

Titre: Développeur Web et Web Mobile

Création du site vitrine de L'association Happy Day.

SYLLA Ibrahim

Sommaire

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 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

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.

Résumé

Le collectif Happy Day est un collectif de jeunes adultes bénévoles issue de différents quartiers de Marseille, réunis afin de proposer différentes activités à but non lucratif, en direction de tous les publics (enfants, jeunes, adultes).
Mettre en avant le vivre ensemble par le biais du sport. Happy Day transmet à la génération qui va suivre tout son savoir pour permettre aux plus jeunes de trouver la voie dans laquelle ils excellent musique,

multimédia, sport, art, mode, cuisine. La plaquette de nos activités est diverse et variée. Nous intervenons dans les écoles, les associations d'entreprise et menons des actions sociales.

En plus de tout cela tous les ans depuis 2017 on organise un grand événement au nom de l'association HAPPY DAY qui constitue la journée convivial par excellence, tout public mélangé avec du sport des animations des grands jeux, des concours et des stands proposé par nos partenaires tout ça lors d'un grand repas partager.

L’association m’a sollicité pour pouvoir créer son site vitrine qui permettra aux personnes voulant adhérer à l’association de le faire directement sur le site et de ne pas passer par une autre plateforme, mais aussi de créer leur propre espace et d’être prévenu lorsqu'un nouvel événement arrive.

Spécifications fonctionnelles

Description de l’existant

Aucun site n’existait auparavant, seule une page Facebook permettait à l’asso de se présenter. Nous avons donc convenu lors de nos différents entretiens non seulement quels étaient les besoins de l'association, les fonctionnalités qui seraient développées mais également l’aspect graphique du site.

Périmètre du projet

Le site sera réalisé en français et ce dernier devra être accessible sur différents supports, à savoir mobile, tablette et ordinateur.

Cible adressée par le site internet

Le site de l’association Happy Day s’adresse à des particuliers.

Arborescence du site

L’arborescence du site se décline comme suit :

- ❖ Page d’accueil
- ❖ Page connexion
- ❖ Page inscription
- ❖ Page mon compte
- ❖ Page paiement
- ❖ Page de confirmation de paiement

Une partie back-office est également prévue afin de permettre la gestion du site.

Descriptifs des fonctionnalités

Authentification

Les utilisateurs doivent pouvoir s’inscrire pour adhérer à l'association, mais aussi se connecter pour accéder à leurs comptes.

Espace adhérent

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

Solution de paiement

La solution de paiement choisie est celle proposée par Stripe. Cette solution permettra au client de procéder au paiement de manière sécurisée par carte bancaire.

Notification par mail

Un mail de confirmation d'adhésion devra être envoyé au nouvel adhérent une fois le paiement effectué.

Back office

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

Spécification techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end :

- Le projet sera réalisé avec le langage PHP (Hypertext Preprocessor)
- Base de données SQL
- Gestionnaire de dépendance: Composer

Technologies utilisées pour la partie front-end:

- Le projet sera réalisé avec du HTML, CSS.
- 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

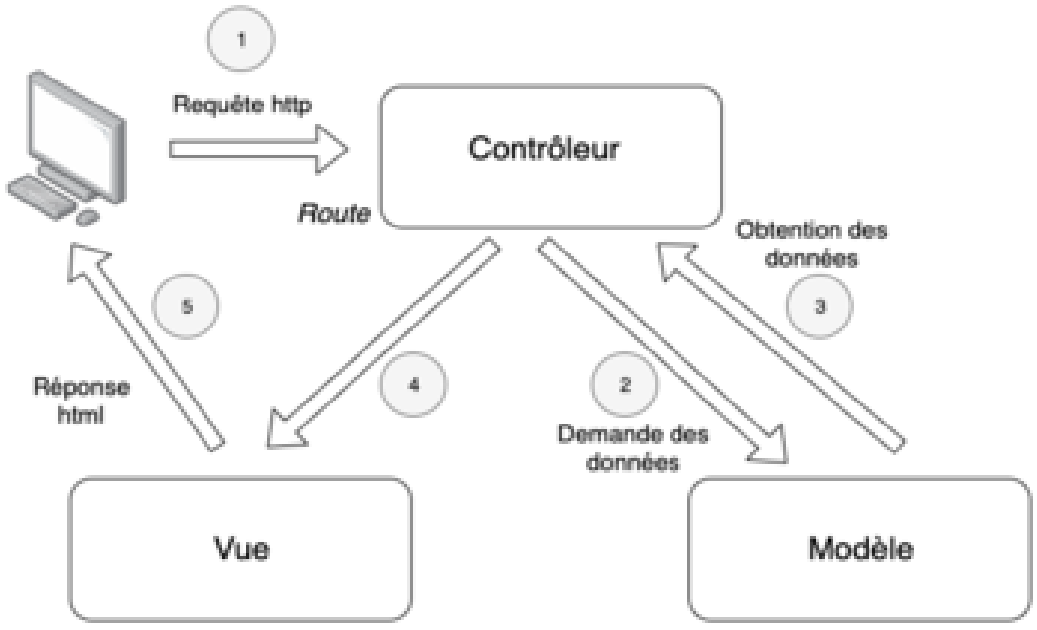
Pour l'organisation du projet j'ai utilisé Trello qui m'a permis de découper le projet en plusieurs tâches et de définir leur ordre de priorité.

Architecture du projet

Pour le développement du projet j'ai utilisé un design pattern de type MVC (Model-View-Controller).
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.
- Vue: 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.

Schéma du modèle MVC:



Source: Wikipédia

Réalisation

Charte graphique

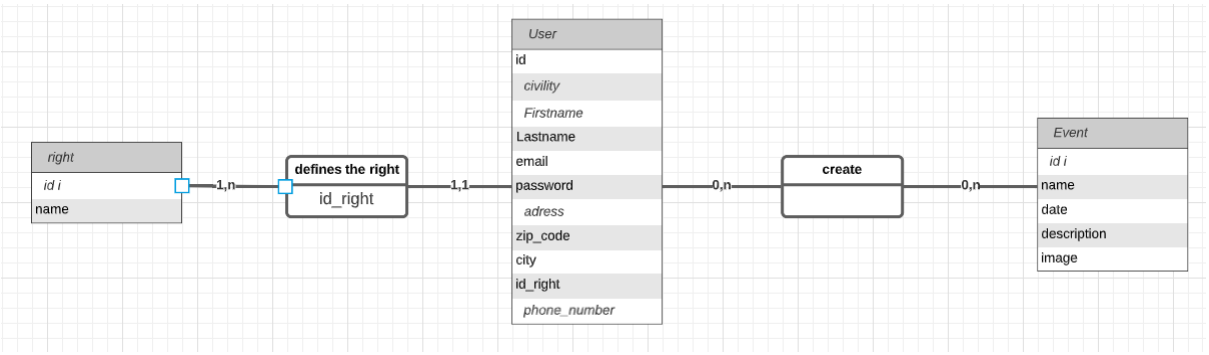
Cette couleur de base a été générée sur Coolors.com. Pour plus de détails sur la palette graphique, une section y est dédiée en aval.

Maquette

La maquette a été réalisée avec le logiciel gratuit Figma
Je me suis donc inspiré du design d'autres sites pour la réaliser.
Vous trouverez le wireframe de la maquette dans les annexes du dossier.

Conception de la base de données

Pour les fonctionnalités demandées par l'association, j'ai développé la base de données suivantes:



Model conceptuel de données

- Comme illustré ci-dessus la base de données se compose de 3 tables. Tout d'abord la table User qui va permettre à l'utilisateur de créer son compte adhérent mais aussi à l'association de recenser tous les adhérents et de les identifier. Cette table est liée aux tables droit et event.

- La table droits définit le droit d'un utilisateur (adhérent ou administrateur). La table events contiendra les évènements qui seront ajoutés par l'administrateur. Elles sont toutes les deux liées à la table User car l'utilisateur (l'administrateur) va gérer c'est deux table.

Extraits de code

Authentification

Pour s'authentifier il faut que l'utilisateur puisse s'enregistrer en base de données.

```php
if (isset($_POST['register'])){
    if (!empty($_POST['civility']) && !empty($_POST['firstname']) && !empty($_POST['lastname']) && !e
        !empty($_POST['password_conf']) && !empty($_POST['adress']) && !empty($_POST['zip_code']) &&

        $civility = security($_POST['civility']);
        $firstname = security($_POST['firstname']);
        $lastname = security($_POST['lastname']);
        $email = security($_POST['email']);
        $password = security($_POST['password']);
        $password_conf = security($_POST['password_conf']);
        $adress = security($_POST['adress']);
        $zip_code = security($_POST['zip_code']);
        $city = security($_POST['city']);
        $phone_number = security($_POST['phone_number']);
```

- Pour commencer je vérifie que les champs ne sont pas vide avec la fonction "empty" ensuite alors si ils ne sont pas vide j'utilise une fonction que j'ai créé qui sécurisent les champs à l'aide d'autres fonctions.

```php
if (preg_match("/^[^@]+@[^@]+\.[a-z]{2,6}$/i", $email) && preg_match('/^[0-9]{10}+$/', $phone_number)){

    $verify = $user->check_if_already_exists($email);
        // var_dump($verify);
    if ($password == $password_conf){
        $password = password_hash($password, PASSWORD_BCRYPT);
        if(count($verify) == 0){
            $user->register($civility, $firstname, $lastname, $email, $password, $adress, $zip_code, $city
        }
        else{
            echo 'user récurant';
        }
    }
```

- Ensuite j'utilise la fonction "preg_match" qui va effectuer une recherche de correspondance avec une expression régulière standard sur les champs demandés (ex: email) et grâce à ça on n'enverra pas de mauvaises informations en base de données (ex: email invalide).
- Si l'information est correcte, alors j'utilise la fonction password_hash() qui hache le mot de passe et j'appelle la méthode register() qui contient la requête SQL qui permet d'insérer en base de données.

```php
if (isset($_POST['connect'])){

    if (!empty($_POST['email']) && !empty($_POST['password'])){

        $email = security($_POST['email']);
        $password = security($_POST['password']);

        if (preg_match("/^[^@]+@[^@]+\.[a-z]{2,6}$/i", $email)){
            $user = new User();
            $verify = $user->check_if_already_exists($email);

            if(count($verify) > 0){
                if (password_verify($password, $verify[0]['password'])){
                    session_start();
                    $_SESSION['utilisateurs'] = $verify;
                    header('Location: ../Views/index.php');
                }
            }
```

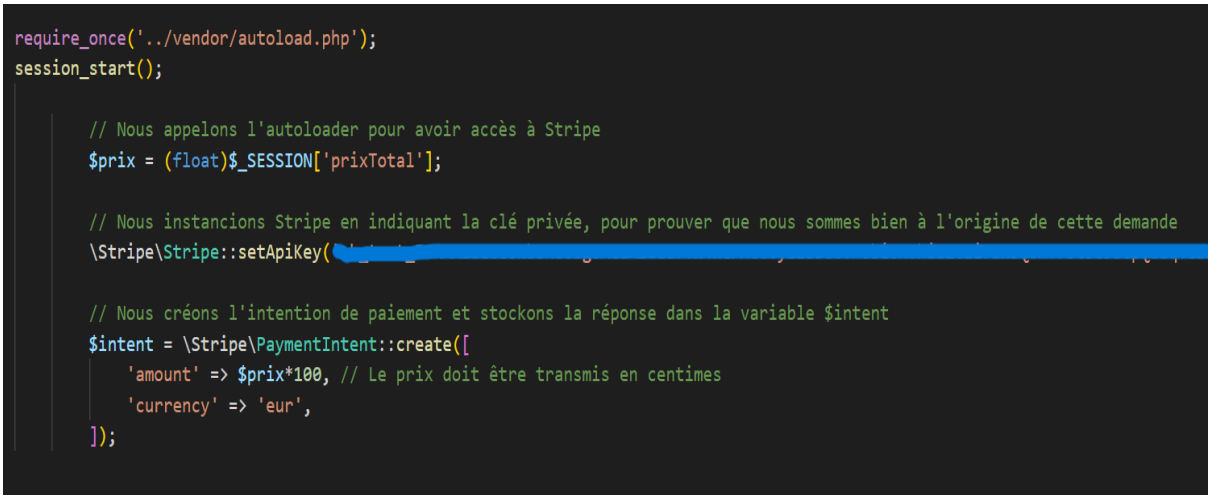
- Pour la partie connexion, je procède de la même manière que pour l'inscription pour la partie sécurité. Par la suite, j'appelle une méthode qui contient une requête SQL qui va permettre de sélectionner toutes les informations de l'utilisateur en fonction de

l'email qui a été donné dans le champ. Si l'email est trouvé en base de données, alors j'utilise la fonction password_verify() qui permet de décrypter le mot de passe haché. Si le mot de passe est trouvé dans la chaîne décryptée, alors je démarre une nouvelle session que je nomme "utilisateur".

Stripe

Partie PHP

Stripe est une solution de paiement qui va permettre à l'adhérent de régler les frais d'adhésion en carte bancaire Visa ou MasterCard.
Son intégration est organisée autour d'une API de type REST. Afin de réaliser l'intégration de la solution de paiement j'ai utilisé la checkout session de Stripe. Pour ce faire, j'ai créé un controller qui va gérer le paiement de l'adhérent.



- J'appelle l'autoloader créé par composer qui va me permettre d'avoir accès à l'API de stripe.

- ❖ Une fois que j'ai l'accès je l'instance en lui indiquant ma clé privée pour prouver que je suis bien à l'origine de cette demande.

- ❖ Enfin, pour finir la partie PHP, je crée l'intention de paiement et je stocke la réponse dans une variable puis je multiplie le prix par 100 car il doit être transmis en centimes.

L'autoloader de Composer:

```php
<?php

// autoload.php @generated by Composer

require_once __DIR__ . '/composer/autoload_real.php';

return ComposerAutoloaderInit42d605a30d1456f8f69511215a2d151b::getLoader();
```

Partie HTML

Dans la partie Views j'ai une page que je nomme "paiement.php" ou sera générer le formulaire de paiement de stripe.

```php
<form method="post">
    <div id="errors"></div>
    <input id="cardholder-name" type="text" placeholder="Titulaire de la carte">
    <div id="card-elements" class="test"></div>
    <div id="card-errors" role="alert"></div>
    <button name='payer' id="card-button" type="button" data-secret="<?= $intent['client_secret'] ?>">Valider le paiement</button>
</form>

<script src="https://js.stripe.com/v3/"></script>
<script src="https://code.jquery.com/jquery-2.0.2.min.js"></script>
<script src="js/stripe.js"></script>
```

- ❖ Le formulaire de paiement est généré grâce à une librairie JQuery et à une librairie Javascript de stripe.

- Je crée des div avec des id qui représentent des éléments spécifiques des librairies.

- Dans le bouton qui valide le formulaire je récupère ma variable qui contient l'intention de paiement.

Partie Javascript

Avec la partie JS je vais pouvoir traiter l'affichage des messages d'erreur, mais aussi récupérer les informations présentes dans le formulaire et les envoyer a Stripe.

```
window.onload = () => {
    // On instancie Stripe et on lui passe notre clé publique
    let stripe = Stripe(

    // Initialise les éléments du formulaire
    let elements = stripe.elements();

    // Récupère l'élément qui contiendra le nom du titulaire de la carte
    let cardholderName = document.getElementById('cardholder-name')

    // Récupère l'élement button
    let cardButton = document.getElementById('card-button');

    // Récupère l'attribut data-secret du bouton
    let clientSecret = cardButton.dataset.secret;


    // Crée les éléments de carte et les stocke dans la variable card
    let card = elements.create("card");

    card.mount("#card-elements");
```

- j'instancie Stripe et je lui fournit ma clé publique.

- Ensuite j'initialise les éléments du formulaire pour pouvoir récupérer les informations et le data-secret qui contient notre intention de paiement.

```
card.addEventListener('change', function(event) {
    // On récupère l'élément qui permet d'afficher les erreurs de saisie
    let displayError = document.getElementById("card-errors")

    // Si il y a une erreur
    if (event.error) {
        // On l'affiche
        displayError.textContent = event.error.message;
    } else {
        // Sinon on l'efface
        displayError.textContent = "";
    }
});
```

- Je crée un événement qui va gérer les messages d'erreur si une erreur est détectée.

```
cardButton.addEventListener("click", () => {
    // On envoie la promesse contenant le code de l'intention, l'objet "card" conter

    stripe.handleCardPayment(
        clientSecret, card, {
            payment_method_data: {
                billing_details: {
                    name: cardholderName.value
                }
            }
        }
    ).then((result) => {
        // Quand on reçoit une réponse

        if (result.error) {
            // Si on a une erreur, on l'affiche

            document.getElementById("errors").innerText = result.error.message
        } else {
            // Sinon on redirige l'utilisateur
            document.location.href = "commandes.php";
        }
    })
})
```

- Ensuite je crée un événement au clic sur le bouton de validation du formulaires.

- On envoie une promesse à l'API contenant l'intention de paiement et l'objet card qui contient les informations du formulaire.

- Enfin dès que l'on reçoit une réponse, si il y a une erreur on l'affiche, sinon on redirige l'utilisateur vers la page de validation de paiement.

CRUD utilisateur

L'acronyme CRUD signifie Create, Read, Update et Delete.

Le crud utilisateurs seras présent dans la partie admin qui pourra créer, lire les informations, modifier ou supprimer un adhérent

- Pour que l'administrateur puisse créer un adhérent, c'est la même procédure que l'inscription d'un adhérent sauf qu'il aura le champ droit car il pourra choisir si la personne est un administrateur ou un client.

- Pour lire je fait une requête qui vas chercher toute les informations des utilisateurs

- Pour modifier je sélectionne les informations d'un utilisateur en fonction de son id et je les affiche dans des champs pour les modifier. je récupère les informations modifier des champs et je les places dans ma requête d'update qui va modifier les info en base de données

- Pour supprimer, j'utilise une requête qui va supprimer l'utilisateur en fonction de son id.

Conception de la partie front-end

Hiérarchisation des Vues et du Style

Chaque vue (ex: connexion.php) intègre un *header* (en-tête) et un *footer* (pied de page), qui eux-même intègre les fichiers de scripts et de style.
Le dossier “Style” contient lui-même des dossiers pour les icônes, les images et les polices d’écriture. Le sous-dossier “CSS” est divisé en trois parties:

- La partie “Desktop” : contient les feuilles de style adaptées aux formats “grand-écran”, présents sur la plupart des ordinateurs ;
- La partie “Mobile” : contient les feuilles de style adaptées aux formats “mobile”, présents sur la plupart des smartphones ;
- La partie générale : contient les feuilles de style qui n’ont pas besoin de s’adapter aux tailles d’écran (ex: fonts.css, palette.css).

Ensuite, un seul fichier, nommé “style.css” est intégré dans le header.

Seulement, comment faire pour savoir quelle partie doit être intégrée ?

J’ai créé une simple fonction qui vérifie le support de l’utilisateur (mobile ou non). En fonction de la valeur booléenne retournée par la fonction, un opérateur ternaire se charge de la sélection du chemin relatif pour importer le fichier style.css.

```php
<?php
// vérifie si l'user est sur mobile ou non
function isMobile() {
    return preg_match("/(android|avantgo|blackberry|bolt|boost|cricket|docomo|fone|hiptop|mini|mobi|palm|p
}

isMobile() ? $cssDIR = "./Libraries/Style/CSS/Mobile/" : $cssDIR = "./Libraries/Style/CSS/Desktop/";
?>

<link rel="stylesheet" href="<?= $cssDIR ?>style.css">
```

extrait du code présent dans “header.php”

Source des icônes et des fontes de caractères

Pour une meilleure expérience utilisateur, j'ai récupéré des icônes gratuites sur le site web Flaticon.com et une fonte simple nommée *Prompt* sur Google Fonts.

Démonstartion de la police "Prompt"

Palette de couleurs

Telle que susmentionnée, la palette de couleurs à été générée sur Coolors.com, en essayant de rester le plus fidèle possible aux couleurs de bases du logo :

logo de la société "Team Happy Days"

```
:root {
    --primary: #F07900;
    --primary-lighter: #ffba6f;
    --primary-light: #F8A145;
    --primary-dark: #D35100;
```

couleurs primaires présentes dans le fichier palette.css

Élaboration des formulaires en plusieurs étapes

Les formulaires d'inscription et de connexion furent les premières fonctionnalités interactives présentes sur le site. Après avoir appliqué un style simple, je me suis trouvé face à un dilemme : le formulaire contient trop d'éléments pour que cela soit ergonomique. J'ai donc décidé d'en faire un formulaire en plusieurs étapes. Le site n'affichent qu'une seule partie du formulaire et passerait à la suivante lorsque l'utilisateur a rempli tous les champs.

Ce fut hélas plus facile à dire qu'à faire. Ainsi je me retrouve assez rapidement devant des erreurs qui, à mes yeux, n'avaient pas lieu d'être.

Après m'être démené pendant trois jours, je réussit enfin à résoudre tous les problèmes, et le formulaire fonctionnait comme un charme.

Veille sur les vulnérabilités de sécurité

Exposition des données sensibles

Comment éviter d'exposer les données stockées ?

- Sécurisez votre base de données avec le chiffrement.
- Utilisez des algorithmes de hachage sécurisés tels que **Argon5**, **Scrypt**, **Bcrypt** et **PBKDF2**.
- Le masquage des données peut être utilisé pour sécuriser les données sensibles d'une base de données.

L'intérêt des algorithmes de hachage est qu'ils permettent de calculer une empreinte (ou hash) d'une chaîne de caractères, par exemple. Cette empreinte est utile pour éviter de stocker en clair le mot de passe dans la base de données.

Dans le cas du projet, j'utilise **Bcrypt**:

```
if ($password == $password_conf){
    $password = password_hash($password, PASSWORD_BCRYPT);
    if(count($verify) == 0){
        $user->register($civility, $firstname, $lastname, $emai
    }
```

Injection SQL

- Cette vulnérabilité permet à un attaquant d'injecter des données non maîtrisées qui seront exécutées par l'application et qui permettent d'effectuer des actions qui ne sont normalement pas autorisées.

- Ce type d'attaque s'effectue généralement grâce aux champs présents dans les formulaires.

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

Comment contrer les injections SQL ?

Validez les entrées dans les champs

Cela consiste à limiter ce que l'utilisateur peut mettre dans la zone de texte. Cela n'empêchera pas l'injection, mais c'est une mesure que vous pouvez mettre en place pour limiter des attaques de base. En effet, les caractères spéciaux spécifiques à certains langages ne pourront pas être utilisés.

- Par exemple, on utilise la fonction preg_match() sur le champ email pour vérifier s' il est valide:

```
preg_match("/^[^@]+@[^@]+\.[a-z]{2,6}$/i", $email)
```

❖ Mais aussi des fonction système qui éliminent certains caracter qui n'ont rien à faire dans les champs:

```php
function security($data){
    //trim permet de supprimer les espaces inutiles
    $data = trim($data);
    //stripslashes supprimes les antishlashs
    $data = stripslashes($data);
    //htmlspecialchars permet d'échapper certains caractéres
    $data = htmlspecialchars($data);
    return $data;
}
```

Préparez les requête SQL

Ce sont des requêtes dans lesquelles les paramètres sont interprétés indépendamment de la requête elle-même, la query est verrouillée. De cette manière, il est impossible d'effectuer des injections.

exemple:

```php
public function register($civility, $firstname, $lastname, $email, $password, $adress, $zip_code, $city, $phone_number) {

    $sql = $this->bdd->prepare("INSERT INTO `users`(`civility`,`firstname`, `lastname`, `email`, `password`,
    `adress`, `zip_code`, `city`, `id_right`, `phone_number`) VALUES (?, ?, ?, ?, ?, ?, ?, ?, 1, ?)");
    $sql->execute(array($civility, $firstname, $lastname, $email, $password, $adress, $zip_code, $city, $phone_number));
}
```

Recherche effectuées à partir d'un site anglophone

Quand je voulais récupérer du texte que j'avais entré en base de donnée pour un event j'avais ce genre de caractère à la place des é ou à:

br?silien

J’ai entendu dire que c’était un problème d’utf8 alors j’ai effectuer une recherche sur stackoverflow et j’ai trouvé cet article qui m’a permit de trouver la solution:

With `PDO`: All you need to do is create a new `PDO` object. The constructor accepts parameters for specifying the database source `PDO`'s constructor mostly takes four parameters which are `DSN` (data source name) and optionally `username`, `password`.

Here I think you are familiar with all except `DSN`; this is new in `PDO`. A `DSN` is basically a string of options that tell `PDO` which driver to use, and connection details. For further reference, check PDO MySQL DSN.

```
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8', 'username', 'passwo
```

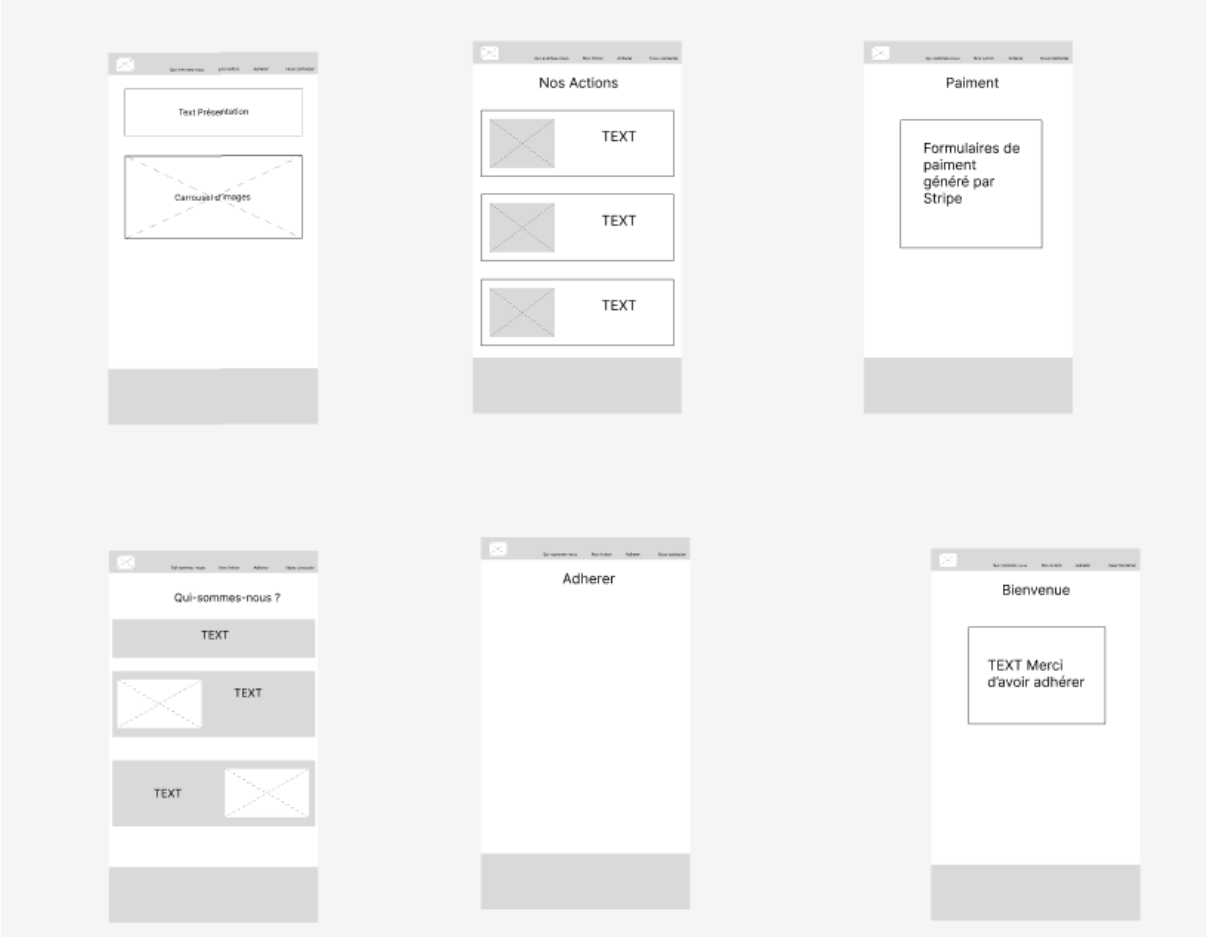
Note: you can also use `charset=UTF-8`, but sometimes it causes an error, so it's better to use `utf8`.

If there is any connection error, it will throw a `PDOException` object that can be caught to handle `Exception` further.

Il fallait rajouter le charset Utf8 dans la connexion a la base de données pour récupérer les informations correctement.

ANNEXES

Maquette Wireframe



Modèle Logique de données