

Titre : Développeur Web et Web Mobile



Mathieu Tatat

Sommaire:

Compétences du référentiel couvertes par le projet	4
Présentation du projet	5
Spécifications Fonctionnelles	6
Description de l'existant	6
Périmètre du projet	6
Cible adressée par le site internet	6
Arborescence du site	7
Description des fonctionnalités	8
1. Inscription	8
2. Connexion	8
3. Catégories produits	8
4. Fiche Produit	8
5. Évaluation produit	9
6. Évaluation établissement	9
7. Espace utilisateur	9
8. Back-end	9
8.1. Création produit	9
8.2. Suppression Produit	10
8.3. Modification Produit	10
Specifications Techniques	11
Choix techniques et environnement de travail	11
Architecture du projet	11
Réalisations	13
1. Charte graphique	13
2. maquette	13
3. Conception de base de données	14

4.	Extraits de code	15
4.1.	Inscription utilisateur	16
4.2.	Vérifications	17
4.3.	Connexion	18
4.4.	Class utilisateurs	19
4.5.	Commenter	20
4.6.	fonction moyenne	20
5.	Veille sur les vulnérabilités	21
5.1	Exposition des données sensibles	21
5.2	Comment éviter d'exposer les données en transit	21
5.3	Comment éviter d'exposer les données stockées	21
6.	Injection SQL	22
7.	Recherche effectuées sur un site anglophone	23

Annexes

Maquette	24
Modèle conceptuel de données	25
<u>Définitions</u>	26

Compétences du référentiel couvertes par le projet:

Le projet couvre les compétences suivantes:

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

- Maquetter une application.
- Réaliser une interface web ou web mobile statique adaptable.
- Développer une interface utilisateur Web dynamique.
- Réaliser une interface utilisateur avec une interface de gestion de contenu ou e-commerce.

Pour l'activité 2, du référentiel **"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.

Présentation du projet:

Paco Pizza est une entreprise créée en 2014 et basée à Aubagne 13400 qui est un service de restauration rapide. Mr Patrick Colonna confectionne et vend ses pizzas depuis son camion situé en centre ville.

Lui-même fils de pizzaïolo depuis plus de 8 ans, il lui tient à cœur de confectionner ses propres produits dans la plus pure tradition du métier et apporte un soin particulier à sélectionner des produits frais et de qualités provenant de tous horizons afin de satisfaire le plus grand nombre.

Il lui plaît également de créer de nouvelles recettes afin de conserver sa clientèle en renouvelant régulièrement sa carte et également attirer les curieux en recherche de nouvelles saveurs.

L'enjeu de ce projet est donc de lui permettre d'avoir une meilleure visibilité sur internet et de pouvoir apporter un visuel à ses dernières créations et aussi de pouvoir avoir un retour sur ce que ses clients pensent de son entreprise.

Spécifications Fonctionnelles:

Description de l'existant:

Mr Colonerra n'étant pas vraiment familiarisé avec internet, seule des pages Facebook et Instagram lui permettent d'avoir un peu de visibilité sur le web.

Nous avons donc convenu lors de nos différents entretiens tout ce dont l'entreprise aurait besoin, aussi bien niveau fonctionnalités qui seraient développées mais également de la charte graphique de la future application.

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 visée par le site internet:

Le site de l'entreprise Paco Pizza, s'adresse à des particuliers désirant se restaurer et souhaitant avoir un visuel des produits contenus dans la carte ainsi que des ingrédients qu'ils contiennent. Afin de pouvoir passer commande sur place ou par téléphone.

Arborescence du site:

L'arborescence du site se décline comme suit :

- Page Accueil
- Page Connexion
- Page Inscription
- Page Profil
- Page Produits par catégories
- Page Produit
- Page Contact
- Page Mentions légales
- Page Administrateur
- Page Gestion de produit

La partie back-end permettra de réaliser la gestion du site.

Description des fonctionnalités

1. Inscription:

Le but premier de Paco pizza étant principalement d'avoir une meilleur visibilité de leur établissement il n'est pas nécessaire que l'utilisateur puisse se connecter afin d'avoir un visuel de leurs produits en revanche il sera possible de créer un compte avec un simple formulaire requérant Nom, Prénom, Email, et Mot de passe à l'utilisateur afin d'avoir accès aux fonctionnalités disponibles.

2. Connexion:

Une fois que l'utilisateur aura créé un compte via son inscription il lui sera possible de se connecter en renseignant simplement dans le formulaire dédié son adresse email ainsi que son mot de passe.

3. Catégories produits

Directement accessible depuis la barre de navigation, l'utilisateur aura la possibilité de filtrer par catégorie les produits qu'il désire consommer en fonction de ses goûts, parmi les trois différentes catégories de produits disponibles.

4. Fiche produit

L'utilisateur aura la possibilité d'accéder à une fiche produit. Cette dernière comprendra:

- Le nom du produit
- La photo du produit
- Le prix
- La description du produit
- La note du produit

- Les commentaires clients

5. Évaluation des produits et commentaires clients

L'utilisateur sera en mesure d'évaluer un produit grâce à un système de notation sur 5 étoiles. 5 étoiles signifiant "Très satisfait du produit" et 1 étoile "Pas du tout satisfait du produit". Il sera également possible à l'utilisateur de commenter le produit et faire part de son expérience culinaire.

6. Évaluation établissement et commentaires clients

Reprenant le principe de notation de produit il sera également possible à un utilisateur de laisser une note allant de 1 à 5 étoiles afin d'évaluer son expérience globale dans l'établissement et laisser son avis dans un livre d'or.

7. Espace utilisateur

L'utilisateur aura la possibilité d'accéder à un espace client lui permettant de gérer ses informations personnelles. A savoir son Nom, son Prénom, son Email, et son Mot de passe .

Il lui sera également possible si il le désire de supprimer son compte et ses informations personnelles.

8. Partie Back-end

Le gérant du site aura la possibilité d'accéder à un espace sécurisé lui permettant d'administrer le site.

8.1. Création d'un nouveau produit:

Le gérant sera en mesure de créer un nouveau produit:

Lors de la création il devra renseigner dans le formulaire dédié:

- Le nom du produit
- Le Prix du produit
- Upload une photo du produit
- Donner une description du produit

8.2. Supprimer un produit existant:

Il sera possible de supprimer un produit que le gérant souhaite retirer de sa carte simplement en cliquant sur le bouton de suppression.

8.3. Modifier un produit existant:

En cas de changement de d'Ingrédient, de Prix, de Photo, ou de Nom il sera possible au gérant de mettre à jour les informations relatives à chaque produit.

Specifications Techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end:

- Ce projet sera réalisé avec le langage **PHP**.
- **SQL** me permettra de gérer les informations en base de données.

Technologies utilisées pour la partie front-end:

- Le projet sera réalisé en **HTML** et **CSS**
- **JAVASCRIPT** lui permettra de dynamiser le site et d'améliorer l'expérience utilisateur .

L'environnement de développement est le suivant:

- ❖ **Visual Studio** code comme éditeur de code
- ❖ **Github** et **Github Desktop** pour le versionning
- ❖ **Figma** pour le maquettage

Pour l'organisation bien qu'ayant travaillé seul sur ce projet j'ai mis en place une méthode Agile. Pour ce faire j'ai utilisé l'outil **Trello** afin de définir et ranger les tâches à réaliser en fonction de leur ordre de priorité.

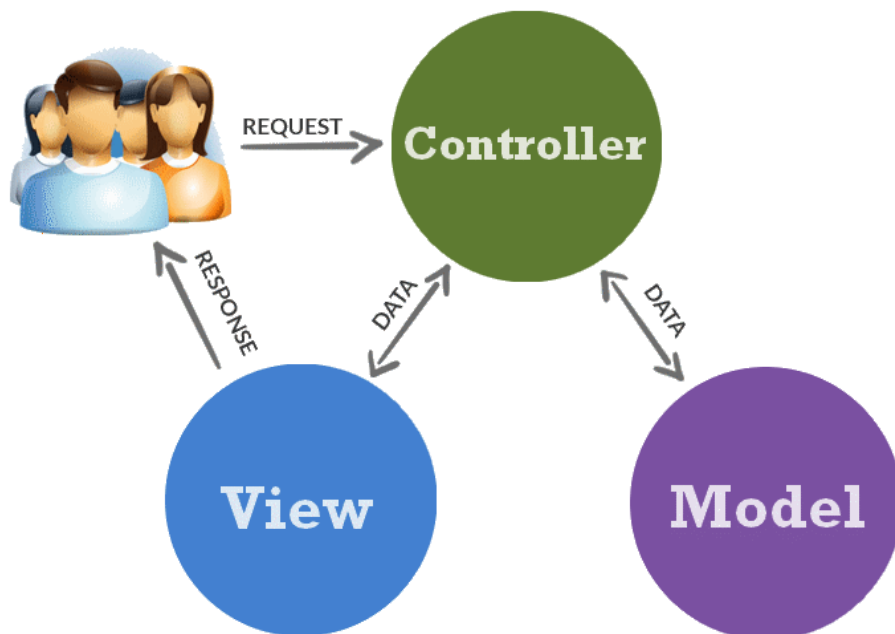
Architecture du projet

J'ai développé ce projet sous PHP, et bien que je n'ai pas utilisé de **router**, j'ai néanmoins voulu, dans un souci d'organisation de mes dossiers me baser sur le modèle **MVC** (Modèle, View, Controller).

- **Modèle** : Le *Modèle* représente le comportement de l'application : traitements des données, interactions avec la base de données etc. Il décrit les données manipulées par l'application et définit les méthodes d'accès.

- **View:** Elle correspond à l'interface avec laquelle l'utilisateur interagit. Les résultats renvoyés par le modèle sont dénués de toute présentation mais sont présentés par les vues. Plusieurs vues peuvent afficher les informations d'un même modèle. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle, et de permettre à l'utilisateur d'interagir avec elles.
- **Controller:** Il prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle. Il n'effectue aucun traitement, ne modifie aucune donnée, il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondante à la demande.

Voici un schéma représentant le modèle MVC




Source : formget.com

Réalisations:

1. Charte graphique:

Dans mon projet j'ai utilisé deux styles de police: Arial et Tai Heritage Pro Serif.

La couleur dominante est la suivante : #F2D675

Polices	Couleur
Arial Tai Heritage pro serif	#F2D675 

2. Maquettage:

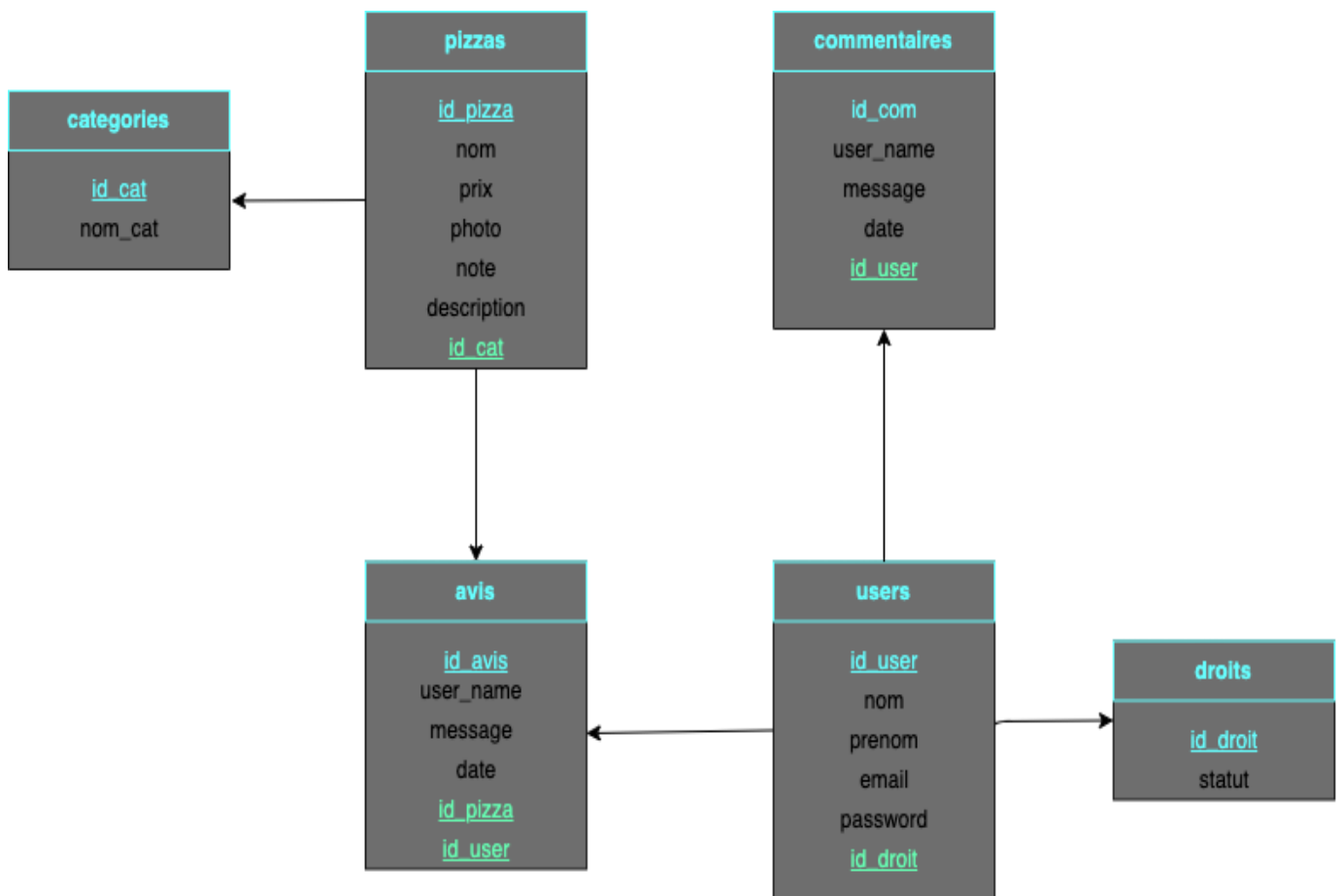
Pour réaliser la maquette j'ai utilisé le logiciel **Figma**.

Démarrant de zéro et le gérant n'ayant pas de Web designer, nous nous sommes mis d'accord ensemble sur l'aspect général, les couleurs, les polices et le rendu visuel que devra avoir le site une fois terminé.

En me basant sur tous ces points j'ai donc réalisé une maquette de l'application. Vous trouverez la dans les [annexes](#) du dossier.

3. Conception de la base de données.

En examinant les fonctionnalités demandées par l'entreprise, j'ai donc créé la base de données suivante.



Modèle logique de données

Je mets également en annexe [le modèle conceptuel de données](#).

Dans le schéma illustré ci-dessus on peut voir que ma base de données s'appuie sur deux tables principales.

1. La table User:

Elle permet aux clients de créer, modifier, et supprimer son compte ainsi que de permettre la connexion utilisateur. Elle est reliée à différentes tables comme la table commentaires qui permet de pouvoir commenter l'expérience globale vécu dans l'établissement. Elle est également reliée à la table Avis qui elle lui permet de donner un avis sur le produit que le client aura consommé.

2. La table Pizzas:

Elle permet à l'administrateur de pouvoir gérer les informations relatives à chaque produit, comme le prix, la description ou le nom. Elle permet aussi de créer ou supprimer des produits sur demande.

Elle est également reliée à la table Avis, car tout avis est propre à une pizza, et elle est aussi reliée à la table catégorie afin de pouvoir classer chaque produit dans la section qui lui est propre.

4. Extraits de code

4.1 Inscription d'un utilisateurs

Afin d'enregistrer son compte, un utilisateur doit remplir un formulaire en précisant son Nom, son Prénom, Son Email, choisir un mot de passe et confirmer celui-ci afin de pouvoir valider son inscription. j'effectue une vérification en base de données afin de m'assurer que l'email de l'utilisateur ne soit pas déjà enregistré en base de données et ainsi qu'il n'ai pas déjà un compte existant.

```
public function createUser(){
    if(isset($_POST['button_inscri'])){
        if (empty($_POST['nom']) || empty($_POST['prenom']) || empty($_POST['email'])
            || empty($_POST['password']) || empty($_POST['passwordConf'])) {
            echo"veuillez remplir tout les champs";
        }elseif($_POST['password'] != $_POST['passwordConf']){
            echo"Password et confirmation sont differents";
        }elseif(strlen($_POST['password'])< 6){
            echo"le mot de passe doit faire 6 caracteres minimum";
        }
        $modelUser= new Utilisateur;
        $verif = $modelUser->checkEmail(trim(htmlentities($_POST['email'])));
        if($verif>0){
            echo"email existant";
        }else {
            $nom = htmlentities($_POST['nom']);
            $prenom = htmlentities($_POST['prenom']);
            $email = htmlentities($_POST['email']);
            $password = htmlentities(password_hash($_POST['password'],PASSWORD_BCRYPT));
            $inscription = new Utilisateur;
            $inscription->inscription($nom,$prenom,$email,$password);
        }
    }
}
```


4.2 Vérification

Afin de pouvoir créer un compte un utilisateur doit respecter plusieurs formats afin de sécuriser au mieux ses données ainsi que de m'assurer qu'il renseigne de bonnes informations.

Ainsi si il laisse des champs vides il sera avertie et ne pourra pas enregistrer son compte, si son adresse email ne correspond pas au format email une regex le comparera et préviendra l'utilisateur du défaut de son format email.

De la même manière que pour l'email, une regex vérifiera le mot de passe de l'utilisateur s'assurant qu'il possède au minimum 8 caractères, et donne la possibilité à l'utilisateur d'insérer minuscules, majuscules, chiffres, et caractères spéciaux à son mot de passe .

```
// check for errors in user inputs and count them
if(empty($nom)){      array_push($errors, "Prenom is required");      }
if(empty($prenom)){   array_push($errors, "Prenom is required");      }
if(empty($email)){    array_push($errors, "Email is required");        }
if (!preg_match('/^[a-z0-9._-]+@[a-zA-Z0-9._-]+\.[a-z]{2,3}$/', $email)){
    array_push($errors, "Email format is wrong");      }
if(empty($password)){ array_push($errors, "Password is required"); }
    if ($pw !== $pwC) { array_push($errors, "The two passwords do not match"); }
if (!preg_match('/^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*{8,})/', $pw)) {
    array_push($errors, "Password format is wrong");    }

//check if user exists
$chkExists = $user->checkExists($email);
```

4.3 Connexion Utilisateur

Lors de sa visite sur le site, si l'utilisateur désire se connecter il pourra si celui-ci s'est déjà inscrit se connecter en renseignant simplement son email et son mot de passe. Si les deux informations correspondent bien aux informations renseignées en base de données alors celui ci se verra connecter et aura la possibilité de d'interagir avec diverses fonctionnalités présente sur le site.

```
public function verification(){  
  
    if(isset($_POST['button_con'])){  
  
        if (empty($_POST['email']) || empty ($_POST['mdp'])) {  
            echo"veuillez remplir tout les champs";  
        }  
        $modelUser= new Utilisateur;  
        $verif = $modelUser->checkEmail(trim( htmlentities($_POST['email'])));  
  
        if($verif < 1){  
            echo"aucun user ne possède cet email";  
        }else{  
            $modelUsers= new Utilisateur;  
            $connexionUser = $modelUsers->selectUser($_POST['email']);  
            if(password_verify(htmlspecialchars($_POST['mdp'], ENT_QUOTES, "ISO-8859-1"),  
                $connexionUser['password'])){  
                $_SESSION['user'] = $connexionUser;  
                // header('location: index.php');  
            }else{  
                echo"login ou password incorrect";  
            }  
            echo"bonjour ".$_SESSION['user']['nom'] ;  
        }  
    }  
}
```

4.4 Class Utilisateur

Afin de pouvoir gérer au mieux toutes les fonctionnalités relatives aux utilisateurs. J'ai créé une Class utilisateurs qui permet l'appel de la méthode désirée de créer, modifier, supprimer, etc, un utilisateur.

```
public function getUserByEmail($id){  
    $sql = 'SELECT * FROM utilisateurs WHERE email = :email';  
    $p = ['email' => $email];  
    $check = $this->selectQuery($sql,$p);  
    return $check->fetchAll();  
}  
  
function createNewUser($nom,$prenom,$email,$password,$id_droit){  
    $sql= 'INSERT INTO utilisateurs (nom, prenom, email, password)  
    VALUES (:nom, :prenom, :email, :password, id_droit = 1)';  
    $p = ['nom' => $nom, 'prenom' => $prenom, 'email' => $email, 'password' => $password];  
    $check = $this->selectQuery($sql,$p);  
    return $check->fetchAll();  
}  
  
function modifyUser($nom, $prenom, $email, $password){  
    $sql= 'UPDATE user  
    SET nom = :nom, prenom = :prenom, email = :email, password = :password  
    WHERE id = :id';  
    $p = ['name' => $name, 'prenom' => $prenom, 'email' => $email, 'password' => $password];  
    $check = $this->selectQuery($sql,$p);  
    return $check->fetchAll();  
}  
  
function deleteUser($id){  
    $sql= 'DELETE FROM user WHERE id = :id';  
    $p = ['id' => $id];  
    $check = $this->selectQuery($sql,$p);  
    return $check->fetchAll();  
}
```

4.5 Commenter

Pour laisser un commentaire sur l'expérience vécue au sein de l'établissement, l'utilisateur, s' il est connecté, devra remplir le formulaire soumis à cet effet. L'enregistrement en base de données prendra en compte le prénom, le message et la date de dépôt du message afin de le retranscrire sur le site.

```
<?php
//envoi des messages
if(isset($_POST['com'])){

    // recuperons le message
    $message = $_POST['message'];

    //verifions si le champs n'est pas vide
    if(isset($message) && $message != ""){
        //inserer le message dans la base de données
        $req = $con->prepare("INSERT INTO commentaires (`prenom`, `message`, `date`)
        VALUES (:prenom, :message, NOW())");
        $req->execute([
            "prenom" => $prenom,
            "message" => $message,

        ]);

        //on actualise la page
        header('Location:index.php');
    }else {
        // si le message est vide , on actualise la page
        header('Location:index.php');
        echo"erreur votre message ne peut pas etre vide";
    }

}

?>
```

4.6 Calcule de moyenne

Il sera également possible à l'utilisateur de noter son expérience en déposant une note allant de 1 à 5. Cette note sera enregistrée en base de données. Après l'enregistrement, elle sera calculée avec toutes les autres notes déjà données par tous les utilisateurs afin de renvoyer une note moyenne de l'expérience des clients, et ce grâce à une simple requête SQL.

```
public function moyenne()
{
    $moyenne = SELECT AVG(note) AS moyenne_note FROM notes

    return $moyenne;
}
```

5. Veille sur les vulnérabilités

5.1 Exposition des données sensibles

Lorsque vous surfez sur Internet, votre navigateur utilise le protocole HTTP pour afficher les pages web, et le protocole **TCP/IP** pour les transmettre.

Si le serveur web établit la **connexion TCP** avec le navigateur, une réponse avec le code status et le fichier demandé (généralement le fichier index.html pour la page web) sera transmise. Mais dans notre cas, les données transitent en HTTP et pas en HTTPS...

Les données transitant en HTTP peuvent être interceptées, car elles transitent en clair.

Comment éviter d'exposer les données sensibles en transit ?

- Utilisez le HTTPS pour l'ensemble de votre site, même s'il ne contient pas de données sensibles.
- Utilisez les requêtes GET pour récupérer les informations et POST pour modifier les informations.
- Sécurisez vos cookies pour qu'ils soient transmis par l'en-tête et via HTTPS.
- Sécurisez vos sessions en ajoutant une date d'expiration, en sécurisant l'ID et en ne mettant pas cet ID dans l'URL.

Les données sensibles ne sont pas seulement en transit, elles sont aussi stockées en base de données. Pour protéger certaines données stockées sur une application, il est possible d'utiliser des algorithmes de hachage.

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.

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.

6. Injections 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 se prémunir?

- Validez les entrées

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.

- Préparez les requêtes SQL

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

```
public function inscription($nom,$prenom,$email,$password){  
  
    $insertUser= $this->db->prepare(  
        "INSERT INTO utilisateurs(nom, prenom, email, password, id_droit)  
VALUES (:nom , :prenom, :email, :password, 1)";  
  
    $insertUser->execute(array(  
        ':nom'=>$nom, ':prenom'=>$prenom, ':email'=>$email, ':password'=>$password ));  
}
```

Pendant la création de de mon site j'ai été confronté au problème suivant comment calculer la somme moyenne de toutes les notes entrées en base de données. J'ai donc effectué un recherche sur le site anglophone Stack Overflow, et j'ai trouvé la solution suivante qui requiert une simple injonction SQL .

The first function we are going to see is the mean. To calculate the average in SQL, you must use the `avg(column_name)` function in your query. The example below shows its use:

```
SELECT avg(column_name) FROM `table_name`;
```

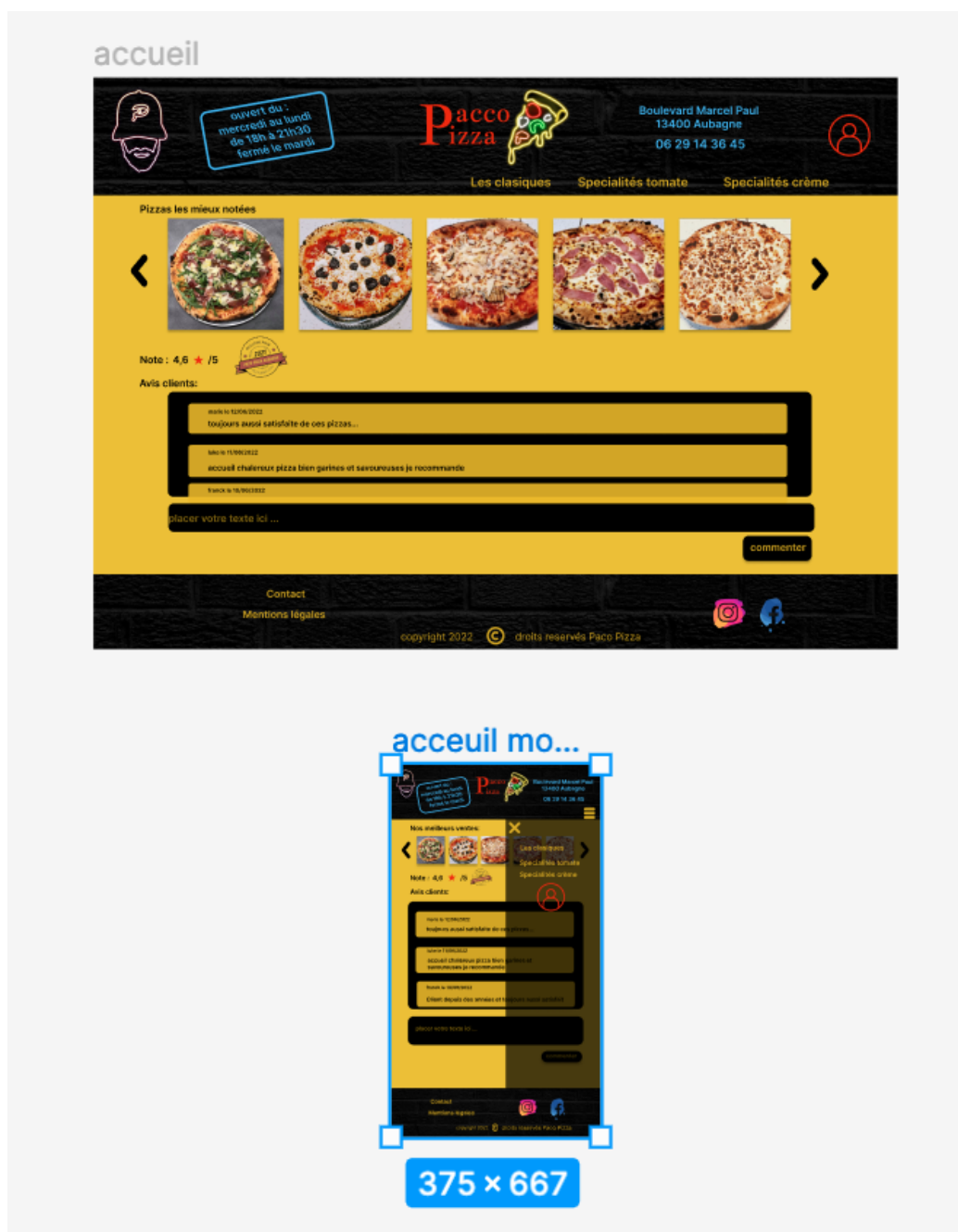
You must therefore use the "SELECT" function in which you will indicate the column on which you want to apply "avg". You still have to confirm the table where the column is located with the indication "FROM".

The result will look like the one below:

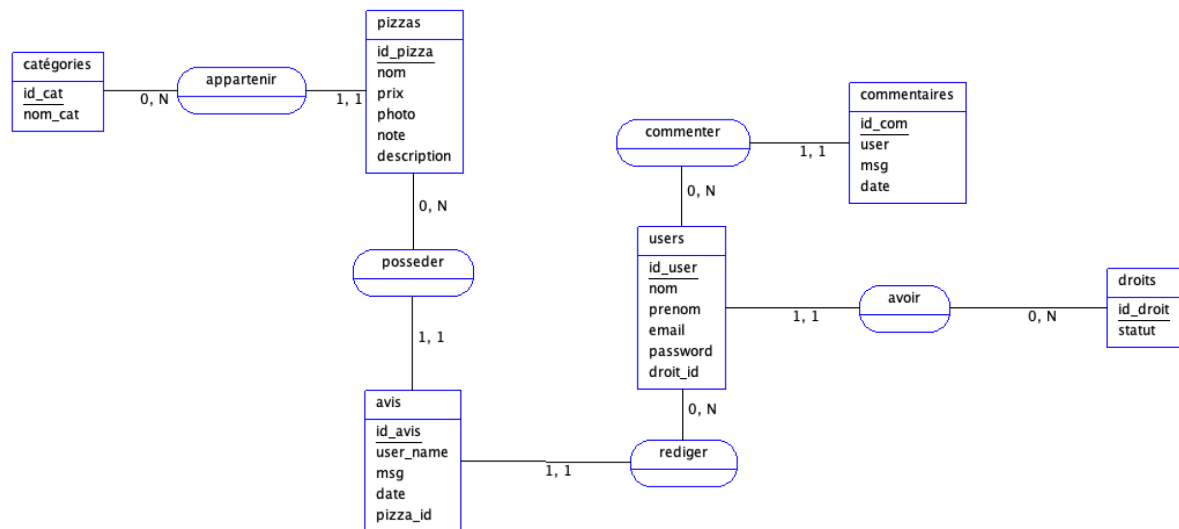
```
mysql> SELECT avg(nb_communes)
-> FROM `enquete`;
+-----+
| avg(nb_communes) |
+-----+
|          366.8300 |
+-----+
1 row in set (0.00 sec)
```

Annexes

Maquette



Modèle Conceptuel de base de données



Définitions:

PHP : Hypertext PreProcessor

SQL : Structured Query Langage

CSS : Cascading Style Sheets

HTML : Hyper Text Markup Langage

URL : Uniform Resource Locator

TCP : Transmission Control Protocol

IP : Internet Protocol