Projet B1 Informatique

-

Développement Web

Social Network

-

Sujet 1

Airbnb

Réalisé par :

GONZALEZ Marie

MAHENDRARASAN Mathursan

Suivi par :

EL HADDAD Auriles

ROMANET Matthias

SOMMAIRE

Pseudos GitHub :

GONZALEZ Marie – Clawchette

MAHENDRARASAN Mathursan – Rshzyyy

Dépôt GitHub du projet

GONZALEZ Marie
MAHENDRARASAN Mathursan

I. Les Choix du Projet

1. Le Choix du Sujet

Après avoir étudié les trois sujets proposés, nous avons remarqué qu'ils étaient tous plutôt similaires mais qu'ils avaient tous un élément difficile différent :

Le premier sujet, Airbnb, demande l'affichage d'une carte pour montrer l'emplacement de la location.
Le deuxième sujet, Blablacar, demande la possibilité d'avoir un nombre libre de ville intermédiaires pour chaque trajet.
Le troisième sujet quant à lui, Leboncoin, demande d'implémenter un système de messagerie interne.

La difficulté du premier sujet nous semblait la plus faisable pour nos compétences et le point bonus lié à la difficulté générale du sujet nous a d'autant plus convaincus, et nous avons donc choisi de le réaliser.

2. Le Choix des Langages et Logiciels

Nous avons choisi d'utiliser Xampp pour héberger localement notre site
Nous avons choisi l'HTML et le CSS pour le front-end ainsi que du PHP et du SQL pour le back-end.
Nous avons aussi utilisé un peu de JS pour effectuer des vérifications dans nos formulaires.

En effet, ces langages de programmation et ce logiciel sont ceux que nous avons étudiés en cours et donc ceux avec lesquels nous nous sentions le plus à l'aise.

Ce sont aussi les langages de programmations les plus utilisés pour le développement web, ce qui nous a permis d'avoir accès à de nombreuses aides et ressources en ligne pour nous aider à corriger nos erreurs et à avancer dans notre projet.

GONZALEZ Marie
MAHENDRARASAN Mathursan

II. La Base de Données (MySQL)

1. Exemples de Lignes de Commande

Nous avons utilisé la base de données pour pouvoir effectuer les fonctions CRUD nécessaires dans le projet.

Nous l'avons par exemple utilisé dans la gestion des annonces :

- **C** : Pour ajouter une annonce, nous avons effectué une ligne de commande de format "INSERT INTO *nomtable* (*nomcolonne1*, *nomcolonne2*, …) VALUE (*valeur1*, *valeur2*, …)" :

```
"INSERT INTO annonce (id_compte, titre, descript, nb_places, numerorue, nomrue, ville, codepostal, prix)
VALUE ($_SESSION[userID], '$_POST[titre]', '$_POST[description]', '$_POST[nbplaces]', '$_POST[numerorue]',
'$_POST[nomrue]', '$_POST[ville]', '$_POST[codepostal]', '$_POST[prix]')"
```

- **R** : Pour afficher une annonce, nous avons effectué une ligne de commande de format "SELECT *nomcolonne1*, *nomcolonne2*, … FROM *nomtable* WHERE *id_table* = *id*" :

```
"SELECT * FROM annonce WHERE id_compte='$_SESSION[userID]'"
```

L'astérisque (*) signifie "toutes les colonnes de la table".

- **U** : Pour modifier une annonce, nous avons effectué une ligne de commande de format "UPDATE *nomtable* SET *nomcolonne1* = *valeur1*, *nomcolonne2* = *valeur2*, … WHERE *id_table* = *id*" :

```
"UPDATE annonce SET titre='$_POST[titre]', descript='$_POST[description]', nb_places='$_POST[nbplaces]',
numerorue='$_POST[numerorue]', nomrue='$_POST[nomrue]', ville='$_POST[ville]',
codepostal='$_POST[codepostal]', prix='$_POST[prix]' WHERE id_annonce='$_GET[IDannonce]'";
```

- **D** : Pour supprimer une annonce, nous avons effectué une ligne de commande de format "DELETE FROM *nomtable* WHERE *id_table* = *id*" :

```
"DELETE FROM annonce WHERE id_annonce = '$_GET[IDannonce]'"
```

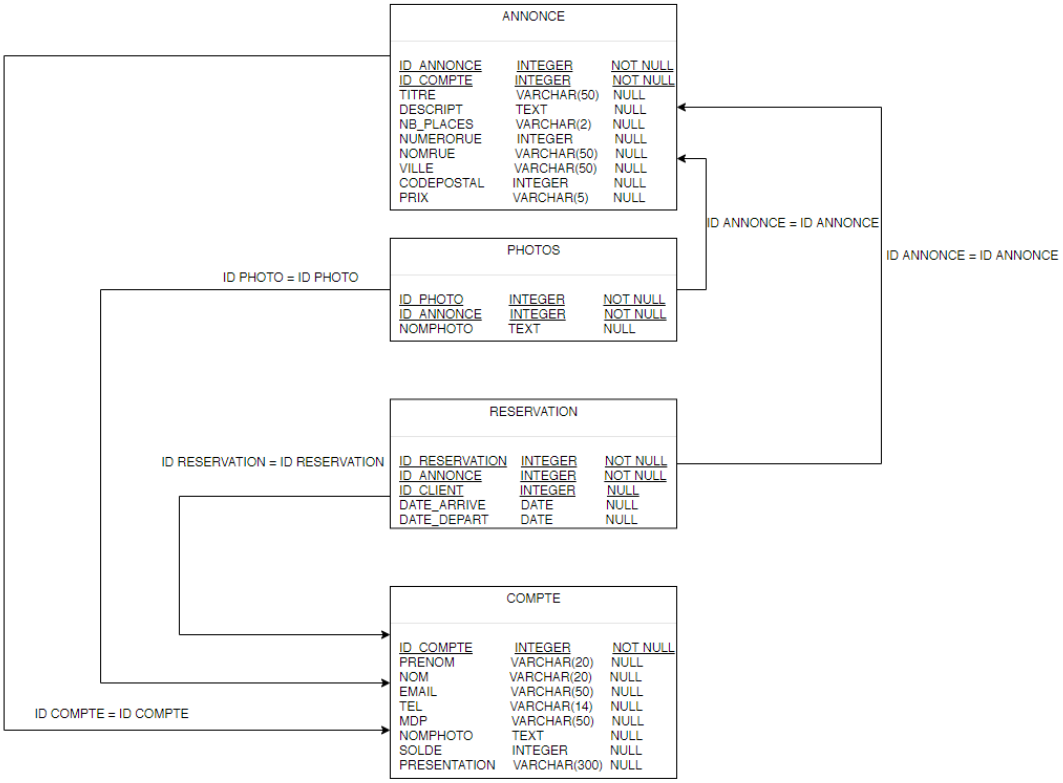
2. Diagramme de la Base de Données

La base de données est constituée d'une table *compte* qui stocke les informations des comptes des utilisateurs du site.

Les utilisateurs peuvent créer des annonces dont les informations seront stockées dans la table *annonce* qui est liée à la table compte par une clé étrangère, *id_compte*.

Ces annonces pourront avoir un nombre indéfinies de photos dont les noms seront stockés dans la table *photos* qui sera liée à la table *annonce* par une clé étrangère, *id_annonce*.

Les utilisateurs peuvent aussi réserver les biens mis en location dans la table *annonce*, les informations de ces réservations seront stockées dans la table *reservation* qui sera liée aux tables *annonces* et *compte* par des clés étrangères, respectivement *id_annonce* et *id_compte*.



GONZALEZ Marie
MAHENDRARASAN Mathursan

III. Le Front-End (HTML, CSS)

1. Le CSS

Le CSS a principalement été réalisé grâce aux classes proposées par *getbootstrap.com*.
Nous avons ainsi pu réaliser notre barre de navigation, nos boutons, nos formulaires, etc… facilement et proprement.

Lorsque nécessaire, nous ajoutions du CSS, soit dans le fichier *css/style.css*, soit directement dans les balises dont le contenu devait être modifié grâce à l'attribut *style*. Ainsi, nous pouvions adapter les éléments de Bootstrap à notre projet rapidement.

Nous avons par exemple utilisé les éléments color pour modifier la couleur du texte, margin pour décaler les éléments entre eux ou border pour ajouter une bordure autour d'un élément.

2. L'HTML

L'HTML nous a permis de créer la structure des pages de notre site :

Les formulaires ont été créés avec la balise <form> contenant des balises <label> pour les titres des champs et des balises <input> ou <textarea> pour les champs eux-mêmes et la balise <button> pour envoyer les valeurs entrées dans le formulaire.

Des balises telles que <h1>, <h3> ou <p> nous ont permis de rédiger le texte à afficher sur nos pages, parfois accompagnées de la balise nous permettant d'isoler une partie du texte si nous souhaitions, par exemple, le mettre d'une couleur différente.

Nous avons utilisé la balise <nav> pour créer une barre de navigation dans notre <header> et nous avons mis des balises <ul> et <li> à l'intérieur afin d'y créer une liste des éléments nécessaires.

D'autres exemples de balises utiles pour notre projet sont, par exemple, la balise <a> qui permet de créer un lien, la balise <img> qui permet d'afficher une image, la balise qui permet de rassembler ou d'isoler des éléments en fonction des besoins et la balise
 qui permet de sauter une ligne.

IV. Le Back-End (PHP)

1. Le Lien avec le SQL

Le PHP permet de faire le lien entre le site web et la base de données. Ce lien se créé avec la ligne de format :
$***variable*** = new PDO("mysql:host=***nomHébergeur***;dbname=***nomBaseDeDonnées***", "***identifiantPseudo***", "***identifiantMotDePasse***", ***option***);

```
$pdo = new PDO("mysql:host=localhost;dbname=db_airbnb","root","",array(PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION));
```

Ensuite, le PHP permet d'effectuer des commandes SQL sur la base de données ouverte avec la ligne précédente. Cela se fait avec les fonctions query(), qui permet d'effectuer une ligne de commande SELECT, et exec(), qui permet d'effectuer une ligne de commande INSERT INTO, UPDATE ou DELETE.

Le résultat de la fonction query() peut être récupéré dans une variable dans laquelle se trouvera donc toutes les lignes correspondant à la recherche réalisée par la commande SELECT. Pour récupérer ces informations sous forme d'objet et ainsi les rendre utilisables en PHP, il faut appliquer, sur cette variable, la fonction suivante :

```
->fetch(PDO::FETCH_OBJ);
```

Nous pouvons maintenant récupérer les valeurs stockées dans la base de données avec une ligne de format :
$*variable* = $*variableBDD*->*nomcolonne*;

Par exemple, nous avons une table "*compte*" dans laquelle sont stockées les informations de chaque compte créés sur le site, telles que le prénom de l'utilisateur (colonne "*prenom*") et l'ID du compte (colonne "*id_compte*"). Nous pouvons récupérer, dans la base de données mise dans la variable $*pdo* (donnée dans un exemple dans une capture d'écran plus haut) le prénom de l'utilisateur dont l'ID du compte est égal à 20 ainsi :

```
$utilisateur20 = $pdo->query("SELECT prenom FROM compte WHERE id_compte = 20");
$objetUtilisateur20 = $utilisateur20->fetch(PDO::FETCH_OBJ);
$prenomUtilisateur20 = $objetUtilisateur20->prenom;
```

Le résultat de la commande SQL est inséré dans la variable $*utilisateur20*. Il est ensuite mis sous forme d'objet dans la variable $*objetUtilisateur20* et, enfin, le prénom de l'utilisateur est récupéré dans la variable $*prenomUtilisateur20*.

Nous pouvons aussi lier le SQL et le PHP dans l'autre sens : on peut en effet insérer des variables de PHP dans des lignes de commandes SQL, il faut cependant les mettre entre guillemets pour que leur valeur puisse être lue par la commande SQL. Par exemple :

```
$utilisateur20 = $pdo->query("SELECT prenom FROM compte WHERE id_compte = '$IDcompte'");
```

2. Les Conditions et les Boucles

Le PHP nous permet aussi de réaliser des conditions (if, else if, else) et des boucles (while, foreach, etc). Ces conditions et ces boucles nous permettent d'obtenir un site qui s'adapte selon les besoins ou les éléments disponibles dans la base de données par exemple.

a. Les Conditions pour Vérifier

Les conditions peuvent permettre de vérifier des informations pour une question de sécurité.
Par exemple, quand un utilisateur se connecte à notre site, l'ID de son compte est stocké dans la variable *$_SESSION["userID"]*. Ainsi, sur les pages ne devant être accessibles que par des utilisateurs connectés, les éléments de la page sont à l'intérieur de la condition suivante :

```php
if(!empty($_SESSION["userID"])){
    //éléments de la page
}else{
    echo "<p style='color:red;'>Veuillez vous connecter pour accéder à cette page.</p>";
}
```

Si la variable n'est pas vide, et donc si l'utilisateur est connecté, les éléments de la page s'affichent alors. Sinon, l'utilisateur est invité à se connecter.

Les conditions peuvent aussi être utilisées pour que les éléments affichés à la page s'adaptent aux besoins de l'utilisateur. Par exemple, sur la page *gestionannonce.php*, l'utilisateur peut cliquer sur des boutons pour ajouter, modifier ou supprimer une annonce. Ces boutons ajoutent des éléments au lien après le lien principal, ce qui permet de rester sur la même page tout en ajoutant des éléments dans la variable $_*GET*. On peut ensuite utiliser une condition pour vérifier les éléments présents dans cette variable et afficher le contenu désiré en conséquence.

```php
if(empty($_GET)){
    //affiche toutes les annonces
    echo "<a href='gestionbiens.php?gerer=ajouter'>Ajouter une annonce</a>";
    echo "<a href='gestionbiens.php?gerer=modifier&...'>Modifier l'annonce</a>";
    echo "<a href='gestionbiens.php?gerer=supprimer&...'>Supprimer l'annonce</a>";
}else if($_GET["gerer"]=="ajouter"){
    //affiche le contenu permettant d'ajouter une annonce
}else if($_GET["gerer"]=="modifier"){
    //affiche le contenu permettant de modifier l'annonce choisie
}else if($_GET["gerer"]=="supprimer"){
    //supprimer l'annonce choisie
}
```

b. Les Boucles pour Répéter

Les boucles sont particulièrement utiles, par exemple, pour les annonces : en effet, n'importe quel utilisateur peut ajouter ou supprimer ses annonces à n'importe quel moment, le site doit donc s'adapter automatiquement au nombre d'annonces existantes à chaque moment donné.

On peut réaliser cela avec une boucle, et c'est à l'intérieur de la condition de cette boucle qu'une variable récupèrera, à chaque tour de la boucle, une nouvelle ligne sous forme d'objet de la table *annonce* :

```php
$annonces = $pdo->query("SELECT * FROM annonce");
while ($annonce = $annonces->fetch(PDO::FETCH_OBJ)) {
    //affiche les informations nécessaires de l'annonces
    //Exemple :
    echo $annonce->titre;
    echo $annonce->description;
}
```

Ainsi, chaque ligne existante dans la table *annonce* sera ajoutée, une à une sous forme d'objet et à chaque tour de la boucle, dans la variable $*annonce*, permettant d'afficher le nombre exact d'annonces avec leurs informations.

Une boucle est aussi utilisée pour l'affichage des photos des annonces puisque l'utilisateur à l'origine de l'annonce peut ajouter un nombre indéfini de photos à son annonce, rendant l'utilisation d'une boucle obligatoire pour les mêmes raisons.

3. Les Superglobales

Pour ce projet, nous avons utilisé trois superglobales de PHP :

- $_*POST* :
  Nous l'avons utilisé dans nos formulaires en mettant l'attribut " method='post' " à la balise <form>. Ainsi, lorsque l'utilisateur appuis sur le bouton affiché grâce à la balise <button> possédant l'attribut " type='submit' ", les informations entrées dans le formulaire par l'utilisateur sont enregistrées dans la variable de type liste nommée $_*POST*. Pour retrouver les éléments enregistrés dans cette liste, il fallait utiliser le nom donné à chaque champs du formulaire dans l'attribut name.

```
<form method="POST">
    <label for="prenom">Prénom</label>
    <input type="texte" name="prenom">

    <button type="submit">Valider</button>
</form>
```

  Dans cet exemple, quand l'utilisateur cliquera sur le bouton "Valider", la variable $_*POST["prenom"]* contiendra le prénom entré dans le champs par l'utilisateur qui pourra donc être réutilisé.

- $_*GET* :
  Comme $_*POST*, cette superglobale peut être utilisée pour récupérer les valeurs entrées par un utilisateur dans un formulaire. Cependant, la " method='get' " envoie les informations en clair dans l'adresse de la page. Cette méthode est donc moins sécurisée et adaptée.
  Cependant, cette superglobale peut être utilisée sans formulaire : puisque les éléments de cette variables sont écrites dans l'adresse de la page, ce qui est écrit dans le l'adresse peut aussi être stocké dans cette variable.
  La liste des éléments de la variable $_*GET* commence, dans le lien, par un point d'interrogation (?), chaque élément est séparé par une éperluette (&) et chaque élément contient l'index de la valeur dans la liste suivi par un égal (=) suivi par la valeur elle-même. Ainsi, avec des boutons ou des liens sur lequel l'utilisateur pourra cliquer, on peut ajouter et modifier le contenu de la superglobale (voir la partie **IV.2.a.** pour un exemple).

- $_*SESSION* :
  Cette dernière superglobale est peu utilisée mais primordiale dans notre projet. Elle est aussi différente des deux précédentes puisqu'elle requiert l'utilisation de la fonction session_start() pour pouvoir être utilisée. Nous avons lancé cette fonction dans le fichier *header.inc.php* afin qu'elle soit lancée sur toutes les pages.
  Cette superglobale stocke les informations qui y sont enregistrée sur le serveur jusqu'à ce qu'elle soit supprimée.
  Ainsi, on y stocke, dans notre projet, l'ID du compte de l'utilisateur lorsqu'il se créé un compte ou qu'il se connecte, et on le supprime quand l'utilisateur se déconnecte. Cela permet à l'utilisateur de rester connecté tant qu'il le souhaite et de lui afficher les pages nécessitant d'être connecté sans qu'il ait besoin de se reconnecter à chaque fois (voir la partie **IV.2.a.** pour un exemple de son utilisation).

V. Envoie de Mail

1. Paramétrages

Les paramètres à avoir dans le fichier *php.ini* :

```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP=smtp.gmail.com
; http://php.net/smtp-port
smtp_port=587

; For Win32 only.
; http://php.net/sendmail-from
sendmail_from = airbnbynov@gmail.com

; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
sendmail_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"
```

Les paramètres à avoir dans le fichier *sendmail.ini* :

```
[sendmail]

; you must change mail.mydomain.com to your smtp server,
; or to IIS's "pickup" directory.  (generally C:\Inetpub\mailroot\Pickup)
; emails delivered via IIS's pickup directory cause sendmail to
; run quicker, but you won't get error messages back to the calling
; application.

smtp_server=smtp.gmail.com

; smtp port (normally 25)

smtp_port=587

; SMTPS (SSL) support
;   auto = use SSL for port 465, otherwise try to use TLS
;   ssl  = alway use SSL
;   tls  = always use TLS
;   none = never try to use SSL

smtp_ssl=auto
```

```
auth_username=airbnbynov@gmail.com
auth_password=kbgerrdgmtrzrxax

; if your smtp server uses pop3 before smtp authentication, modify the
; following three lines.  do not enable unless it is required.

pop3_server=
pop3_username=
pop3_password=

; force the sender to always be the following email address
; this will only affect the "MAIL FROM" command, it won't modify
; the "From: " header of the message content

force_sender=airbnbynov@gmail.com
```

Les fichiers *php.ini* et *sendmail.ini* paramétrés sont disponibles dans le dépôt sur GitHub.

2. L'Utilisation

L'envoie de mail est possible grâce à la fonction de PHP mail(). Cette fonction nécessite quatre arguments :

- L'adresse email du destinataire
- Le sujet de l'email
- Le message de l'email
- L'adresse email de l'expéditeur

Un mail est envoyé quand un utilisateur créé un compte (voir fichier *inc/inscription.inc.php*) et quand un utilisateur réserve un bien mis en location (voir fichier *inc/reservation.inc.php*) :

```php
$subject = "Copie de votre inscription chez Airbnb";

$emailmessage = "Bonjour " . $prenom . $nom . ", voici la confirmation de votre inscription.\n\n";
$emailmessage .= "Prénom : ".  $prenom . "\n";
$emailmessage .= "Nom : ". $nom . "\n";
$emailmessage .= "Email : ". $email . "\n";
$emailmessage .= "Mot de passe : ". $mdp . "\n";
$emailmessage .= "Solde : ". $solde . "\n";

$headers = "From: airbnbynov@gmail.com";

mail($email, $subject, $emailmessage, $headers);
```