

# Dossier – Baccalauréat

VILMINOT Aurélien

INSTANTCRYPT

*Messagerie instantanée chiffrée*

## Tables des matières :

### **INTRODUCTION**

#### **I. PRÉSENTATION DU PROJET**

- A. Langages informatique utilisés
- B. Logiciels utilisés

#### **II. APPRENTISSAGE**

- A. Test d'un modèle de chat nazi
- B. Mise en ligne du site
- C. Communication entre deux ordinateurs

#### **III. RÉALISATION DU SITE FINAL**

- A. Création de la base de données
- B. Structure du site
- C. Pages d'identification et d'inscription
- D. Page des membres inscrits
- E. Page de chat

### **CONCLUSION**

- A. Les améliorations à apportées
- B. Conclusion générale

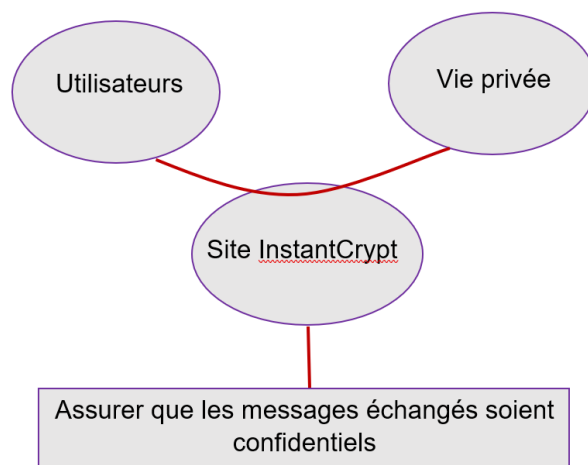
# INTRODUCTION

880 millions. Ce sont les revenus, en dollars, dégagés en 2017 par Axicom, l'une des principales sociétés spécialisées dans la collecte et la vente de données sur les consommateurs. D'après le régulateur américain du commerce, cette entreprise posséderait des informations sur plus de 700 millions de personnes dans le monde. Ceci n'est qu'un exemple parmi tant d'autres sur la collecte massive d'informations réalisée principalement par des entreprises localisées aux Etats-Unis. En effet, la législation en vigueur dans ce pays permet aux entreprises de revendre tout simplement les données personnelles que ce soit au gouvernement ou à d'autres entreprises. La protection des données personnelles sur internet est donc un enjeu crucial pour de nombreuses personnes. Ainsi, de plus en plus de logiciels ou d'applications chiffrés voient le jour. On peut citer, parmi elles, WhatsApp, Telegram ou encore TOR, un réseau informatique superposé mondial et décentralisé.

Nous en sommes donc venus à la problématique suivante : Comment communiquer de manière sécurisée ?

Afin de répondre à cette problématique, nous avons donc décidé de réaliser un chat chiffré en ligne. Pour ce faire, nous avons donc décidé de nous répartir les tâches : Léo s'occupe du design, Christophe réalise le chiffrement des messages et m'a aidé à coder le site enfin, pour ma part, je m'occupe du codage du chat.

Dans une première partie, je vais vous présenter de manière général le projet en lui-même. Ensuite, je parlerais des différentes recherches documentaires que j'ai faites ainsi que des premiers essais réalisés. Enfin, dans une dernière partie, j'expliquerai comment fonctionne le site final ainsi que les pages que j'ai créées.



# PARTIE I : PRÉSENTATION DU PROJET

## A. Langages informatique utilisés

On considère aujourd'hui qu'il existe deux types de sites web : les sites statiques et les sites dynamiques.

**Les sites statiques** sont des sites réalisés uniquement à l'aide des langages **HTML** et **CSS**. Les versions les plus récentes sont le HTML5 et le CSS3. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés. Ce n'est donc pas très pratique quand on doit mettre à jour son site plusieurs fois dans la même journée. Les sites statiques sont donc bien adaptés pour réaliser des sites dit « vitrine », pour présenter par exemple son entreprise, mais sans aller plus loin. Ce type de site se fait de plus en plus rare aujourd'hui, car dès que l'on rajoute un élément d'interaction (comme un formulaire de contact), on ne parle plus de site statique mais de site dynamique.

**Les sites dynamiques**, plus complexes, utilisent d'autres langages en plus de HTML et CSS, tels que **PHP** et **SQL**. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster. La plupart des sites web que vous visitez aujourd'hui (Amazon, YouTube...) sont des sites dynamiques.

Pour notre cas, nous avons décidé logiquement de créer un site dynamique à l'aide du PHP et MySQL.

Nous avons également dû faire appel à l'**Ajax** qui est un langage dérivé du PHP. Ce langage nous permet d'afficher en direct les messages reçus sans recharger totalement la page.

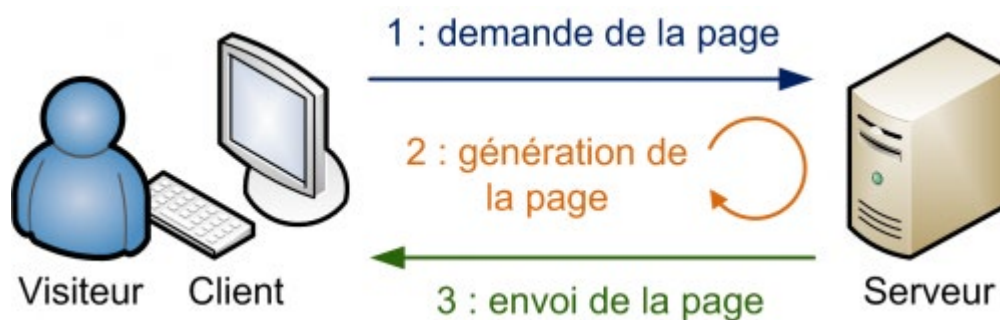
**JavaScript** est aussi de la partie. Ce langage permet de permettre aux messages de s'envoyer et également d'en recevoir. Il nous sert également à vérifier les formulaires d'inscription sont bien remplis avant d'être envoyés à la base de données. Dans notre cas, nous avons utilisé le langage de manière **interprétée** : dans ce cas, il n'y a pas de compilation. Le code source reste tel quel, et si on veut exécuter ce code, on doit le fournir à un interpréteur qui se chargera de le lire et de réaliser les actions demandées. Éventuellement, pour obtenir de significatifs gains de performances, le code peut être compilé à la volée pendant son exécution, c'est aujourd'hui ce que font la plupart des interpréteurs JavaScript.

## B. Logiciels utilisés

Lorsque vous voulez visiter un site web, vous tapez son adresse dans votre navigateur web, que ce soit Mozilla Firefox, Internet Explorer, Opera, Safari ou un autre. Il faut savoir qu'Internet est un réseau composé d'ordinateurs. Ceux-ci peuvent être classés en deux catégories :

*Les clients* : ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web. Dans les schémas qui vont suivre, l'ordinateur d'un client sera représenté par l'image suivante.

*Les serveurs* : ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du Web.



*Fonctionnement d'un site dynamique*

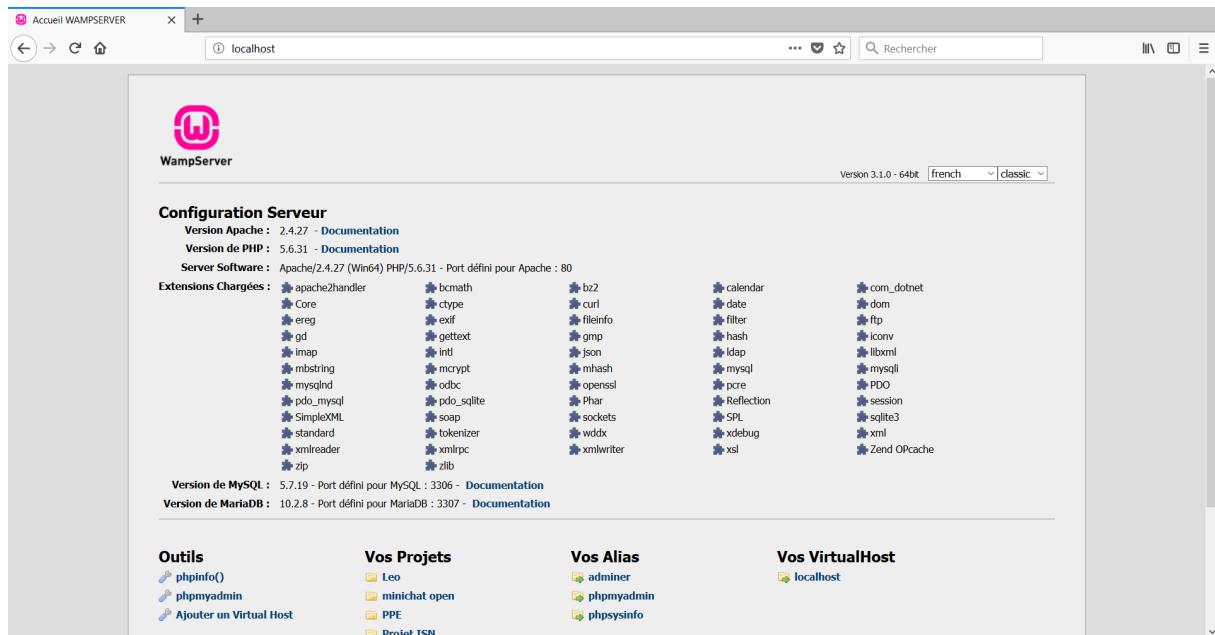
Pour que votre ordinateur puisse lire du PHP, il faut qu'il se comporte comme un serveur. Cependant nous n'allions pas acheter un serveur pour un simple projet. Il suffit simplement d'installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes. Ces programmes sont les suivants :

**Apache** : c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.

**PHP** : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP. En clair, en combinant Apache et PHP, l'ordinateur sera capable de lire des pages web en PHP.

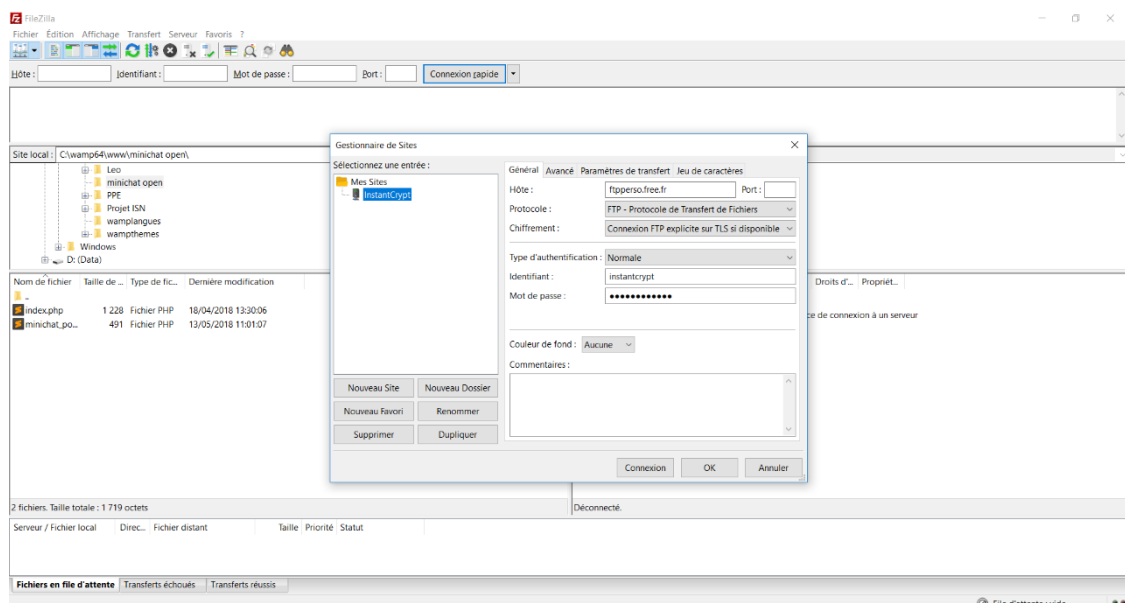
**MySQL** : c'est le logiciel de gestion de bases de données. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site).

Tous ces éléments qui vont nous aider à créer notre site dynamique sont libres et gratuits. Certes, il en existe d'autres (parfois payants), mais la combinaison Apache + PHP + MySQL est la plus courante sur les serveurs web, à tel point qu'on a créé des « packs » tout prêts qui contiennent tous ces éléments. L'un d'entre eux est WampServer. Nous avons donc installé ce logiciel sur un ordinateur.



*Interface de WampServer*

Nous avons également utilisé le client FTP Filezilla qui permet d'envoyer le site sur un serveur. FTP signifie « File Transfer Protocol » et, pour faire court et simple, c'est le moyen que l'on utilise pour envoyer nos fichiers.

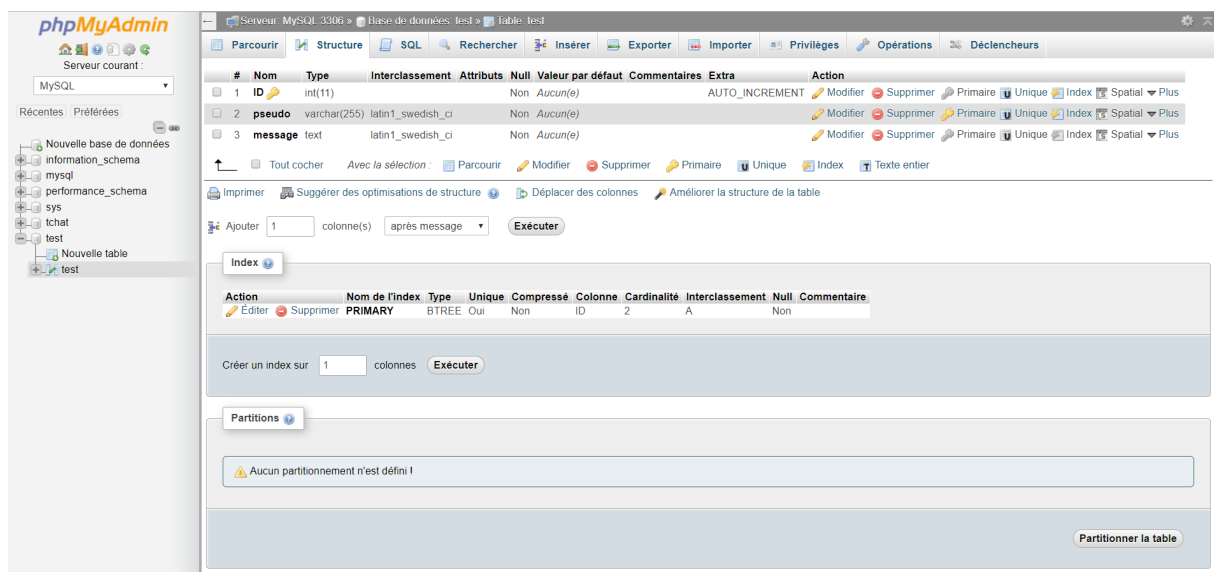


*Interface du client FTP Filezilla*

## PARTIE II : APPRENTISSAGE

### A. Test d'un modèle de chat

J'ai donc commencé le projet en analysant un modèle de minichat fournis par le site Open Classroom afin d'en comprendre le fonctionnement des différents langages utilisés. Dans ce premier modèle, seul le PHP et le SQL sont exploités. Je me suis également familiarisé avec la gestion des bases et donc avec l'interface **phpMyAdmin**.



Dans phpMyAdmin On distingue les trois champs suivants :

**ID** (type INT) : il nous permettra de savoir dans quel ordre ont été postés les messages. Il faudra le mettre en « auto\_increment » pour que les numéros s'écrivent tout seuls, et ne pas oublier de sélectionner « Primaire » (cela dit à MySQL que c'est le champ qui numérote les entrées).

**Pseudo** (type VARCHAR) : on indique la taille maximale du champ (ici, 255).

**Message** (type TEXT) : On indique le type TEXT ce qui nous permet d'envoyer des messages de longueurs illimités.

J'ai ensuite analysé le code du site composé de deux fichiers :

« minichat.php » : contient le formulaire permettant d'ajouter un message et liste les 10 derniers messages.

« minichat\_post.php » : insère le message reçu avec dans la base de données puis redirige vers « minichat.php ».

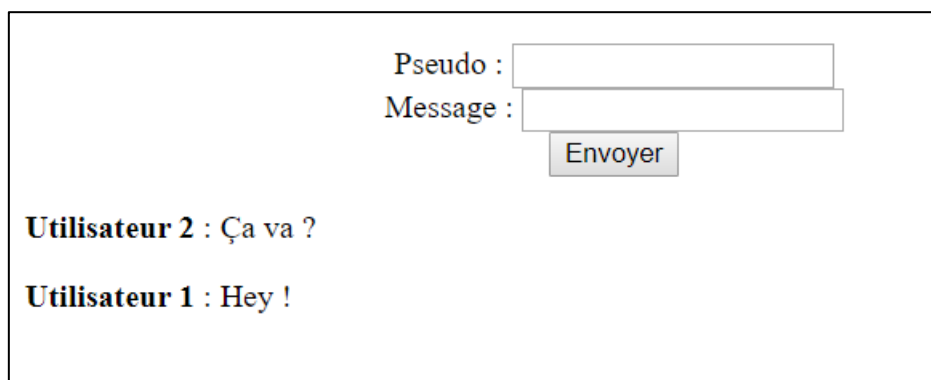
```

24 <?php
25 // Connexion à la base de données
26 try
27 {
28     $bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');
29 }
30 catch(Exception $e)
31 {
32     die('Erreur : '.$e->getMessage());
33 }

```

Dans la variable « \$bdd », on indique les identifiants requis pour la connexion de la base de données. Si la connexion échoue, un message est affiché. Dans un premier temps, j'ai dû faire face à de nombreux problèmes de connexion à la base de données. Cela était dû à un mauvais paramétrage de la base de données.

Une fois, la base de données configurée et le site paramétré et, j'ai ouvert ce dernier par l'intermédiaire du logiciel WampServer présenté précédemment.



Pseudo :

Message :

**Utilisateur 2 :** Ça va ?

**Utilisateur 1 :** Hey !

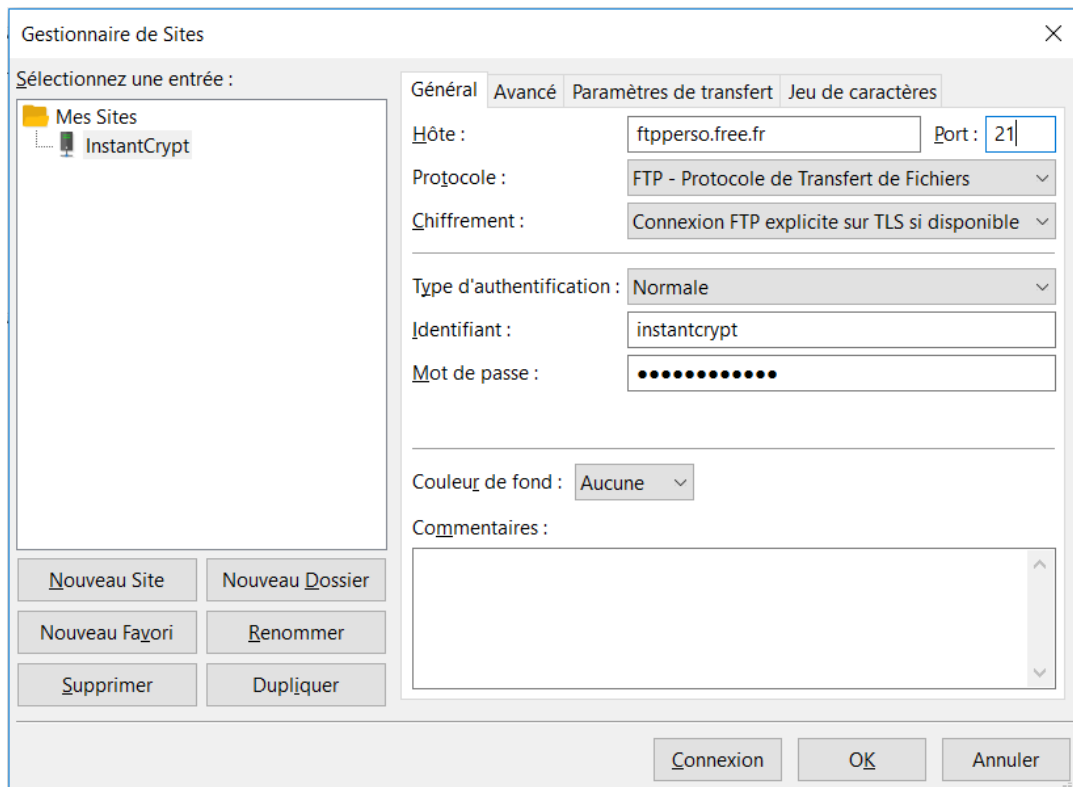
*Capture d'écran du site une fois ouvert*

## B. Mise en ligne du site

Nous avons ensuite décidé d'héberger le site sur un serveur afin qu'il soit accessible en ligne. J'ai donc choisi d'utiliser un serveur mis à disposition par un hébergeur professionnel. Possédant une Freebox, j'ai donc exploité le service d'hébergement fourni par l'opérateur. Afin d'envoyer le site sur le serveur proposé par Free, j'ai utilisé le client FTP Filezilla. Pour cela, il suffit de rentrer l'adresse du serveur Free - « ftperso.free.fr » - ainsi que l'identifiant et le mot de passe du compte Free.

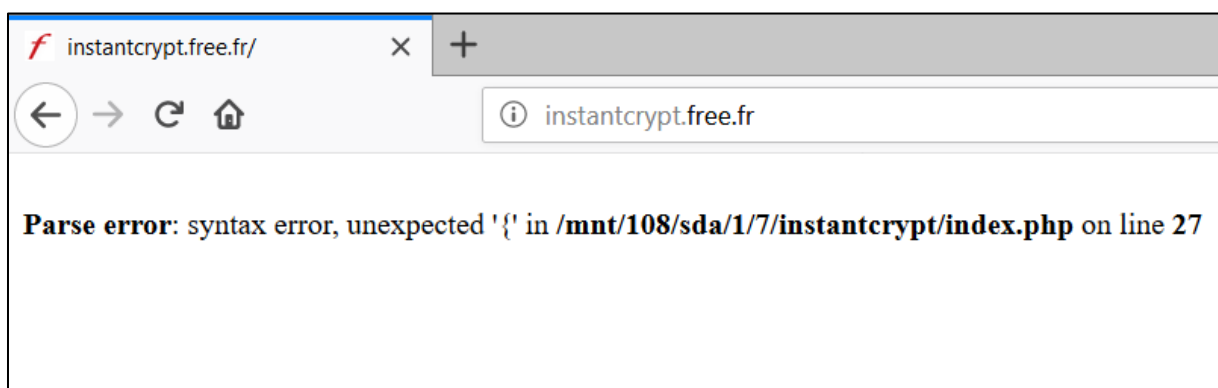






*Interface de connexion de Filezilla*

Une fois cette étape passée, j'ai tenté d'accéder au site en passant par un navigateur internet grâce à l'adresse fournie par Free. Cependant l'accès au site était impossible. En effet, un message d'erreur s'affichait dans le navigateur. Après de longues recherches, je me suis rendu compte que cela était dû au fait que la version de la base de données MySQL de l'hébergeur (free.fr) était trop vieille. Le code généré par la page PHP n'était donc pas supporté par l'interface phpMyAdmin.

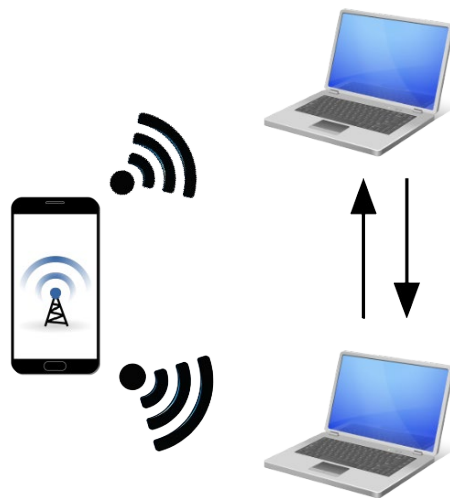


*Message d'erreur affiché par le navigateur*

Nous avons donc renié l'idée de mettre en ligne le site. Nous nous sommes donc concentrés sur la mise réseau local du site.

## C. Communication entre deux ordinateurs

Avec Christophe, nous avons voulu mettre nos deux ordinateurs en réseaux. J'ai donc effectué un partage de connexion avec mon smartphone. Nous étions donc connectés au même réseau.



*Schéma résumant la situation*

Nous avons ensuite voulu vérifier si les ordinateurs étaient bien dans le même réseau et si la connexion entre eux est possible. Pour cela, nous avons fait un ping entre les deux ordinateurs. Ping est une commande qui permet d'envoyer un signal vers une machine ou un nom de domaine et de recevoir, en echo, une réponse qui permet de savoir s'il y a bien une ressource qui répond à l'autre bout. En effet, un ordinateur envoie 4 paquets de données à l'ordinateur distant.

```
Invite de commandes
Description. . . . . : Intel(R) Dual Band Wireless-AC 8265
Adresse physique . . . . . : F8:5A:C2:26:77:14
DHCP activé. . . . . : Oui
Configuration automatique activée. . . . . : Oui
Adresse IPv6 de liaison locale. . . . . : fe80::a8ca:b9a0:8bac:7a6c%16(préfére)
Adresse IPv4. . . . . : 192.168.43.232(préfére)
Masque de sous-réseau. . . . . : 255.255.255.0
Bail obtenu. . . . . : mercredi 16 mai 2018 11:05:25
Bail expirant. . . . . : mercredi 16 mai 2018 14:32:11
Passerelle par défaut. . . . . : 192.168.43.1
Serveur DHCP . . . . . : 192.168.43.1
IAID DHCPv6 . . . . . : 133731522
DUID de client DHCPv6. . . . . : 00-01-00-01-21-74-BE-AA-2C-FD-A1-22-3C-24
Serveurs DNS. . . . . : 192.168.43.1
NetBIOS sur Tcpip. . . . . : Activé

C:\Users\chris>ping 192.168.43.98

Envoi d'une requête 'Ping' 192.168.43.98 avec 32 octets de données :
Réponse de 192.168.43.98 : octets=32 temps=7 ms TTL=128
Réponse de 192.168.43.98 : octets=32 temps=3 ms TTL=128
Réponse de 192.168.43.98 : octets=32 temps=3 ms TTL=128
Réponse de 192.168.43.98 : octets=32 temps=3 ms TTL=128

Statistiques Ping pour 192.168.43.98:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 3ms, Maximum = 7ms, Moyenne = 4ms

C:\Users\chris>
```

*CMD permettant de lancer le ping*

Cependant, nous nous sommes confrontés à de nombreux problèmes. En effet, mon ordinateur n'acceptait pas les paquets envoyés par Christophe. Après de nombreux essais, nous nous sommes rendu compte que cela était dû au Pare-Feu de l'antivirus. Une fois cette étape passée, nous avons donc pu afficher le

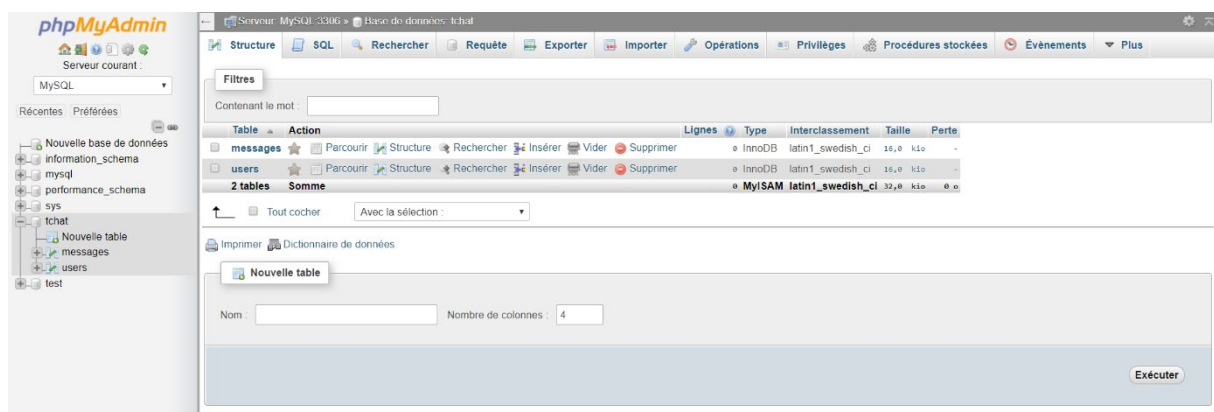
site sur les deux ordinateurs distants. Nous pouvions donc communiquer grâce au chat fourni à OpenClassroom.

En revanche, nous étions obligés de recharger la page à chaque fois que nous voulions afficher les nouveaux messages. De plus, nous devions rentrer à chaque fois le pseudo que l'on voulait adopter. Nous avons donc décidé de créer un site plus élaboré avec des comptes d'utilisateurs et des messages instantanés.

## PARTIE II : RÉALISATION DU SITE FINAL

### A. Création de la base de données

J'ai donc commencé par créer la base de données. Cette base comporte deux tables. « messages » s'occupe de la gestion des messages tandis que « users » liste tous les utilisateurs inscrits sur le site.



*Interface de la base données*


La table « messages » est structurée de la manière suivante :

- « id » : permet de lister par ordre tous les messages reçus et envoyés
- « sender » : utilisateur envoyant le message
- « receiver » : utilisateur recevant le message
- « message » : permet de stocker le message
- « date » : permet de stocker la date à laquelle le message a été envoyé

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/>	1	id			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/>	2	sender	varchar(255)	latin1_swedish_ci	Non	Aucun(e)		
<input type="checkbox"/>	3	receiver	varchar(255)	latin1_swedish_ci	Non	Aucun(e)		
<input type="checkbox"/>	4	message	text	latin1_swedish_ci	Non	Aucun(e)		
<input type="checkbox"/>	5	date	datetime		Non	Aucun(e)		

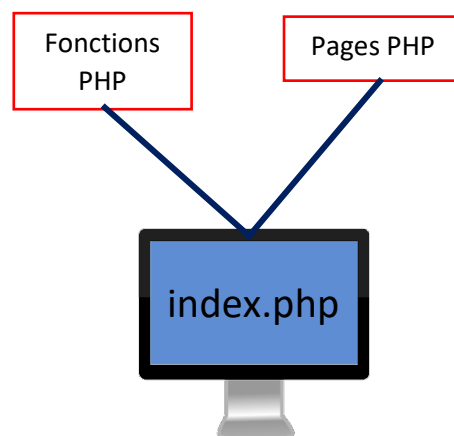
La table « users » est structurée de la manière suivante :

- « id » : permet de lister par ordre tous les utilisateurs inscrits
- « name » : permet de stocker les noms choisis par les utilisateurs
- « email » : stocke l'ensemble des adresses mail utilisées
- « password » : permet de stocker les mots de passe associés aux emails

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/> 1	<b>id</b> 	int(11)			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/> 2	<b>name</b>	varchar(255)	latin1_swedish_ci		Non	Aucun(e)		
<input type="checkbox"/> 3	<b>email</b>	varchar(255)	latin1_swedish_ci		Non	Aucun(e)		
<input type="checkbox"/> 4	<b>password</b>	varchar(255)	latin1_swedish_ci		Non	Aucun(e)		

## B. Structure du site

Le schéma suivant illustre la structure du site.



La page index.php, réalisée par Léo, récupère les deux pages et fonctions associées. Les pages incluses dans le dossier « fonctions » permettent d'exploiter les données incluses dans la base de données.

## C. Pages d'identification et d'inscription

### 1. Page d'inscription

J'ai tout d'abord créé la page d'inscription du site.

```
<?php
    if(isLogged() == 1){ // Si l'utilisateur est déjà connecté
        header("Location:index.php?page=membres"); // Alors il est redirigé vers la page des membres
    }
?>
```

Tout d'abord, on vérifie si l'utilisateur est déjà connecté sinon il est automatiquement redirigé vers la page de la liste des membres.

```
<?php
if(isset($_POST['submit'])){ // Créer une fonction pour chaque champs du formulaire lorsque l'utilisateur clique le bouton d'inscription
    $name = htmlspecialchars(trim($_POST['name'])); //Le 'trim' permet d'enlever autoatiquement les espaces de début et de fin
    $email = htmlspecialchars(trim($_POST['email']));
    $password = sha1(htmlspecialchars(trim($_POST['password'])));

    if(email_taken($email) == 1){ // Si l'adresse mail est déjà utilisée alors un message d'erreur s'affiche
        $error_email = "L'adresse email est déjà utilisée...";
    }else{
        register($name, $email, $password); // Sinon cela inscrit l'utilisateur
        $_SESSION['tchat'] = $email;
        header("Location:index.php?page=membres"); // Redirige l'utilisateur vers la page où sont affichés tous les membres inscrits
    }
}
}>
```

Ensuite, on crée une fonction pour chaque champ du formulaire lorsque l'utilisateur clique le bouton d'inscription. Le 'trim' permet d'enlever automatiquement les espaces de début et de fin entrés par inadvertance par l'utilisateur. On vérifie si l'adresse mail est déjà utilisée lors de l'inscription. Si l'adresse mail est déjà utilisée alors un message d'erreur s'affiche sinon cela inscrit l'utilisateur et le redirige l'utilisateur vers la page où sont affichés tous les membres inscrits.

```
<?php

function email_taken($email){ // Ouvre la fonction 'email'
    global $db;
    $e = array('email' => $email); //Créer un tableau contenant l'adresse email de l'utilisateur
    $sql = 'SELECT * FROM users WHERE email = :email'; // Créer une requête SQL
    $req = $db->prepare($sql); // Prépare la requête en la plaçant dans une variable 'req'
    $req->execute($e); // Exécute la requête avec le tableau créé précédemment
    $free = $req->rowCount($sql); // Cherche dans la base de données si l'adresse email est déjà utilisée ou pas grâce à la fonction PHP 'rowCount'

    return $free;
}
```

Dans le dossier fonction, le fichier « register.func.php » permet de vérifier si l'utilisateur utilise une adresse mail déjà utilisée ou pas. Pour cela, on effectue une requête à la base en lui demandant de scanner toutes les adresses mail et de vérifier s'il y a similitude.

```
function register($name, $email, $password){ // Fonction permettant d'associer chaque données fournies par l'utilisateur à une fonction
    global $db;
    $r = array(
        'name'=>$name,
        'email'=>$email,
        'password'=>$password
    );
    $sql = "INSERT INTO users(name,email,password) VALUES(:name,:email,:password)"; //Enregistre toutes ces données dans la base de données
    $req = $db->prepare($sql); // Prépare la requête en la plaçant dans une variable 'req'
    $req->execute($r); // Exécute la requête avec le tableau créé précédemment
}
```

Toujours dans le même fichier, j'ai créé une fonction permettant d'associer chaque termes saisis par l'utilisateur à la rangée correspondante dans la base de données. Ces informations sont donc enregistrées et pourront être réutilisées lors de la connexion.

## 2. Page de connexion

```
<?php
if(isLogged() == 1){ // Si l'utilisateur est déjà connecté
    header("Location:index.php?page=membres"); // Alors il est redirigé vers la page des membres
}
?>
```

Pour la page de connexion, tout on vérifie si l'utilisateur est déjà connecté ou non grâce au code précédent.

```
<?php

if(isset($_POST['submit'])){ // Créer une fonction pour chaque champs du formulaire lorsque l'utilisateur clique le bouton de connexion
    $email = htmlspecialchars(trim($_POST['email']));
    $password = sha1(htmlspecialchars(trim($_POST['password'])));

    if(user_exist($email,$password) == 1){ // Si l'adresse mail et le mot de passe utilisés sont bon alors l'utilisateur va être connecté
        $_SESSION['tchat'] = $email;
        header("Location:index.php?page=membres"); // Redirige l'utilisateur vers la page où sont affichés tous les membres inscrits
    }else{ // Sinon un message d'erreur s'affiche
        $error_user_not_found = "L'adresse email ou le mot de passe est incorrect";
    }
}
?>
```

Ensuite, on crée une fonction pour chaque champ du formulaire lorsque l'utilisateur clique le bouton de connexion. Comme pour l'inscription, le 'trim' permet d'enlever automatiquement les espaces de début et de fin entrés par inadvertance par l'utilisateur. On vérifie si l'adresse mail et le mot de passe utilisés sont bons. Si c'est le cas alors l'utilisateur va être connecté et redirigé vers la page où sont affichés tous les membres inscrits. Sinon un message d'erreur s'affiche.

```
<?php

function user_exist($email,$password){ // Fonction permettant d'associer chaque données fournies par l'utilisateur à une fonction
    global $db;
    $u = array(
        'email'=>$email,
        'password'=>$password
    );
    $sql = "SELECT * FROM users WHERE email=:email AND password=:password"; //Envoie toutes ces données dans la base de données
    $req = $db->prepare($sql); // Prépare la requête en la plaçant dans une variable 'req'
    $req->execute($u); // Exécute la requête avec le tableau créé précédemment
    $exist = $req->rowCount($sql); // Cherche dans la base de données si l'adresse email et le mot de passe existent ou pas grâce à la fonction PHP 'rowCount'
    return $exist;
}
```

Dans le fichier « signin.func.php », j'ai créé une fonction permettant d'associer chaque donnée fournie par l'utilisateur à une fonction. On envoie ensuite ces informations dans la base de données et on cherche dans la base de données si l'adresse email et le mot de passe existent ou pas grâce à la fonction PHP 'rowCount'.

## 3. Page de déconnexion

La page de déconnexion a pour simple but de fermer la session active et de rediriger l'utilisateur vers la page de connexion.

```
<?php
session_destroy(); // Ferme la session de l'utilisateur
header("Location:index.php?page=signin"); // Redirige l'utilisateur vers la page de connexion
?>
```

## D. Page des membres inscrits

Il a fallu ensuite créer la page listant l'ensemble des membres inscrits sur le site. Pour cela, on vérifie tout d'abord si l'utilisateur est bel bien connecté à une session. Dans le cas contraire, il est automatiquement redirigé vers la page de connexion.

```
<?php
if(isLogged() == 0){ // Si l'utilisateur n'est connecté à aucune session
    header("Location:index.php?page=signin"); // Alors il est redirigé vers la page de connexion
}
?>
```

On lit ensuite l'ensemble de contenu du tableau regroupant les utilisateurs et si l'adresse mail est différente de la session active alors on affiche le nom de l'utilisateur ainsi que son adresse mail.

```
<?php
foreach(get_membres() as $membre){ // Exploite tout le contenu du tableau
    if($membre->email != $_SESSION['tchat']){ // Si l'adresse email du membre est différente de celle de la session active
        ?>
```

Dans le fichier « membres.func.php », j'ai créé une fonction permettant de récupérer l'ensemble des utilisateurs listés dans la base de données et de les intégrer dans un tableau exploité précédemment.

```
<?php
function get_membres(){ // Créer une fonction sans paramètre
    global $db;
    $req = $db->query("SELECT * FROM users"); // Requête à la base de données sélectionnant tout ce qu'il y a dans la table utilisateurs
    $results = array(); // Création d'un tableau
    while($rows = $req->fetchObject()){ // Parcourt tous les résultats de la requête
        $results[] = $rows; // Permet d'afficher tous les résultats dans le tableau créé
    }
    return $results;
}
```

## E. Page de chat

Enfin, j'ai créé les fonctions nécessaires à la communication entre les différents utilisateurs. On vérifie dans un premier temps que l'utilisateur est bien connecté et que son contact existe bien.

```
<?php
if(!isset($_GET['user']) || isLogged() == 0 || user_exist() != 1){ // Si l'utilisateur n'est pas défini, n'est pas connecté ou la fonction créée est différente de 1
    header("Location:index.php?page=home"); // Alors l'utilisateur est redirigé vers la page d'accueil
}
```

Pour cela, on crée un tableau listant les différents utilisateurs inscrits sur le site et l'on cherche si l'email du correspondant existe bien et si cette dernière est différente de la session active grâce à la fonction PHP 'rowCount'.

```

<?php
function user_exist(){ // Créer une fonction sans paramètres
    global $db;
    $e = array('user' => $_GET['user'], 'session'=>$_SESSION['tchat']); // Créer une variable contenant un tableau qui contient les données à rechercher
    $sql = "SELECT * FROM users WHERE email =:user AND email != :session"; // Créer une variable contenant la requête SQL
    $req = $db->prepare($sql); // Prépare la requête en la plaçant dans une variable 'req'
    $req->execute($e); // Exécute la requête avec le tableau créé précédemment
    $exist = $req->rowCount($sql); // Cherche dans la base de données si les identifiants existe ou pas grâce à la fonction PHP 'rowCount'
    return $exist;
}

```

Pour terminer, on récupère tous les messages envoyés précédemment entre les deux correspondants.

En ce qui concerne l'envoi et la réception des messages, c'est une partie qui a été réalisée par Christophe.

## CONCLUSION

### A. Les améliorations à apportées

Nous avons donc réussi à concevoir entièrement un chat adapté pour un réseau local. Néanmoins, des améliorations sont à envisagées :

- Témoin permettant de voir que le message a bien été lu et reçu par le correspondant
- Notification avertissant l'arrivée d'un nouveau message
- Afficher l'heure sous les messages
- Créer une liste d'amis

### B. Conclusion générale

Ce projet m'a donc permis d'avoir une vision globale de ce qu'est concrètement le monde de l'informatique et plus précisément de la programmation. En effet, ce dernier englobe différents langages web tel que le PHP, HTML, ou encore le Javascript. La partie du projet que j'ai effectué m'a permis de me perfectionner dans le PHP, langage dans lequel j'étais novice en début d'année. J'en tire donc une expérience positive tant au niveau social que technologique.