

TD : Introduction au développement web (Partie 1)

Vous allez travailler en autonomie pendant ces deux jours. Vous devrez suivre le canevas du document de TD, mais vous pouvez avancer à votre rythme.

Prenez le temps d'aller chercher des informations complémentaires et si vous connaissez déjà certaines parties, libres à vous d'aller plus loin que les consignes.

À chaque étape, vous devez m'envoyer des liens, des documents ou des captures d'écrans. Vous pouvez bien sur me contacter si vous ne comprenez pas un point particulier

But du TD

L'objectif de ce TD est de **concevoir une application client-serveur web simple**, en partant de zéro, pour comprendre les **fondamentaux du développement web**.

Vous apprendrez à :

- Créer une **interface web** avec HTML et CSS
- Gérer l'envoi de données avec un **formulaire HTML**
- Traiter ces données avec un **script PHP**
- Les **stocker dans une base de données** MySQL
- **Mettre en ligne** leur projet sur InfinityFree, un hébergeur web gratuit

À la fin du TD, chaque élève aura publié en ligne une **application web fonctionnelle**, accessible par un lien que je pourrais consulter pour vérifier le résultat.

Workflow

- Vous développerez en local sur une stack de type [wampserver](#)
- Après avoir testé et validé vous uploaderez vos fichiers sur une solution d'hébergement en ligne (infinityfree) avec un client FTP comme Filezilla

Compétences :

- Comprendre la typologie des sites web
- Réaliser une interface HTML/CSS
- Lire/Ecrire/Modifier des données dans une base de données avec PHP
- Gérer un formulaire

Prés-requis

- Utiliser un IDE (comme Visual Studio). Vous pouvez installer une extension PHP comme PHP Intelephense ou PHP Tools for Visual Studio Code
- Avoir Git Installé

- Avoir une stack MAMP, LAMP ou XAMP installée (Vous pouvez utiliser [Wampserver](#))
Vous les avez utilisé pour héberger un site wordpress

Intérêt d'utiliser WampServer dans un workflow de développement

Même si le projet final est hébergé sur un site distant comme InfinityFree, utiliser **WampServer en local** reste **fortement recommandé** pour plusieurs raisons pédagogiques et pratiques.

1. Développer hors-ligne, en toute sécurité

→ Pas besoin d'être connecté à Internet pour tester son code PHP ou sa base de données. Les erreurs ne sont visibles que localement, ce qui évite de "casser" la version en ligne.

2. Cycle de test rapide

→ Le code s'exécute immédiatement en local, sans passer par FTP. On gagne un temps énorme lors des tests.

3. Meilleur débogage

→ Les erreurs PHP s'affichent directement dans le navigateur local, souvent avec plus de détails que sur InfinityFree (où les erreurs sont parfois masquées).

4. Expérience réaliste de serveur web

→ Wamp simule un **serveur Apache avec MySQL** comme sur InfinityFree, donc les élèves travaillent dans **le même environnement technique**, sans les contraintes de l'hébergement distant.

5. Organisation professionnelle du travail

→ Le code est d'abord développé, testé, puis déployé. Cela enseigne un vrai **workflow propre** :

développement local → tests → mise en ligne via FTP

En pratique












👉 Créez un dossier dans le répertoire local (`www` ou `htdocs`) suivant le type de stack utilisée
Par exemple, si vous utilisez wampserver

```
C:\wamp64\www\intro_web\
```

Ce sera votre dossier local

vous pouvez créer ce dossier à côté de votre dossier WordPress il faudra simplement sélectionner le bon dossier

Index of /

Name	Last modified	Size	Description
 blog-version2/	2023-11-29 15:46	-	
 cake/	2023-12-15 11:39	-	
 cours/	2024-10-21 18:00	-	
 gto_goodies/	2023-12-01 14:01	-	
 initial/	2022-10-16 12:35	-	
 intro_web/	2025-06-25 11:17	-	
 moodle/	2024-09-13 18:01	-	
 my_app_name	2024-01-23 17:45	-	
 poo/	2024-11-04 16:31	-	
 test_pdo	2023-11-30 15:43	-	
 wordpress/	2023-09-27 14:22	-	

Dossier local

Apache/2.4.54 (Win64) PHP/8.2.0 mod_fcgid/2.3.10-dev Server at localhost Port 80

Consignes d'étapes

Quand vous voyez ce pictogramme :

! ... **Consignes à respecter**

Envoyez-moi l'objet de la consigne sur trentindev@gmail.com

Première journée — Découverte & mise en place

Objectifs :

- Créer un compte sur InfinityFree pour pouvoir créer un SSR avec du PHP
- Comprendre les architectures web (SSR, CSR, SPA)
- Configurer l'hébergement FTP + base de données
- Mettre en ligne une première page HTML

Contenus :

- Typologie des architectures web :
 - *Client/Serveur (SSR)* : réponse générée côté serveur (ex : PHP)
 - *SPA (Single Page Application)* : via JS (hors-scope ici)
 - *CSR (Client Side Rendering)* : HTML statique + AJAX (voir annexe)
 - *Hybride* : mélange des deux
- Création de compte sur <https://www.infinityfree.com/>

- Accès au cPanel : créer un FTP, une base MySQL, un sous-domaine
- Utilisation de Filezilla pour transférer un fichier HTML

Partie 1 : Création d'un compte sur InfinityFree

Objectif :

Disposer d'un espace web gratuit et accessible publiquement. Ceci vous permettra de vous familiariser avec le workflow d'un site web client/serveur.

Choix de l'hébergeur

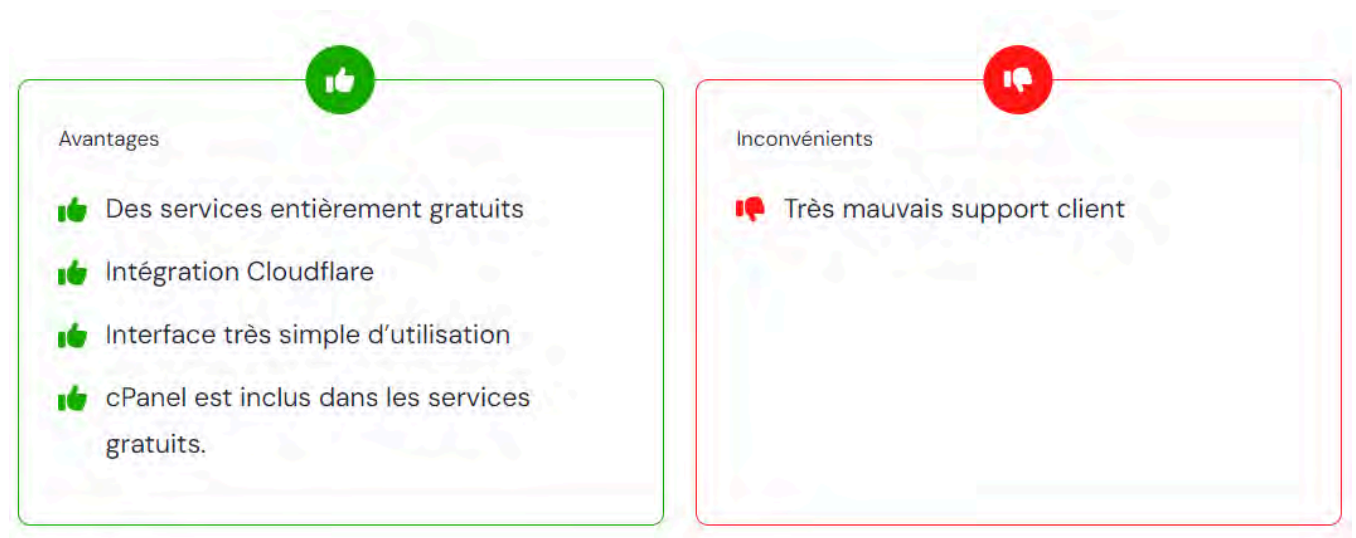
J'ai choisi infinityfree.fr pour répondre au deux besoins :

- **Gratuité**
- **Prise en charge de PHP et MySQL** : les services sont livrés avec le support PHP et MySQL ce qui signifie que vous pouvez également déployer votre site Web codé sur mesure.
- Outils d'administration (phpMyAdmin, compte FTP)

Ce n'est certainement pas la seule solution(Selon googiehost.com) est si vous voulez un site pérenne il vous faudra certainement choisir une solution payante.

Bien qu'InfinityFree garantisse une disponibilité de 99,9 %, ses pannes sont fréquentes et sa disponibilité est très mauvaise, loin d'atteindre 99,9 %. La note de 3 est inférieure à la moyenne, mais elle reste acceptable pour les amateurs de sites web. InfinityFree obtient un 10 pour ses tarifs, car il propose un hébergement gratuit.

Avantages, inconvénients et améliorations d'InfinityFre



- ✓ Hébergement entièrement gratuit et sans publicité
- ✓ Installateur Softaculous avec plus de 400 scripts
- ✓ Prise en charge PHP et MySQL idéale pour WordPress

- ✖ L'espace de stockage est limité à 5 Go
- ✖ Convient uniquement aux sites Web très basiques
- ✖ Aucune option pour acheter ou mettre à niveau les ressources
- ✖ Base de connaissances et support du forum uniquement



InfinityFree propose un hébergement gratuit (et sans publicité)

Comme tout les produits gratuits, vous allez être confronté à **pas mal de pubs sur les interfaces d'administration seulement, pas sur le site lui même**, donc :

Attention ou vous cliquez !!

Faites attention aux données que vous hébergerez

Pour ces services illimités, InfinityFree impose de nombreuses limites aux ressources serveur. Par exemple, l'espace de stockage gratuit n'est que de 5 Go, ce qui est insuffisant pour héberger un site web de nos jours.


Dans tout les cas je vous invite à supprimer vos sites (ou même votre compte) si vous n'en avez pas le besoin après les cours. Si vous désirez les garder, envisagez l'upload vers une version premium payante (~3€)

Si vous avez déjà une solution d'hébergement vous pouvez l'utiliser mais attention à la compatibilité de votre version de PHP avec les code donnés ici (8.3). Il vus faudra aussi adapter la partie base de données.


Création du compte


Vous allez créer un compte d'hébergement gratuit qui vous permettra d'utiliser PHP et HTML pour votre site

- 👉 Allez sur : <https://www.infinityfree.com/>
- 👉 Créez un compte (si possible avec identification via votre compte GitHub)



Login to your account

 Login with Google

 Login with GitHub

Email Address

you@example.com

Password

Password

[I forgot my password](#)


Keep me logged in

Free Website


Créez le compte

ACCOUNTS

Hosting Accounts

 Zipped folder

Download

 Download Images Here

Your Accounts

+ Create Account

No accounts yet. [Create an account now!](#)

Active Accounts: 0 / 3

Sélectionnez l'option gratuite

Choose Hosting Plan

InfinityFree is sponsored by iFastNet, a provider of powerful, low cost web hosting. Are you looking for more features, more server power, personal support and more? Then iFastNet is the service for you!

INFINITYFREE	STARTER PREMIUM	SUPER PREMIUM	ULTIMATE PREMIUM
\$0 forever	\$2.50 / month, billed yearly	\$3.65 / month, billed yearly	\$6.02 / month, billed yearly
5 GB Disk Space Unlimited Bandwidth Unlimited Hosted Domains ✗ Email Accounts ✗ cPanel Control Panel ✗ Unlimited Hits ✗ PHP Version Selection ✗ PHP mail() support ✗ Full DNS Management ✗ Remote MySQL Support ✗ Python/Node.js Support	5 GB Disk Space 250 GB Bandwidth 1 Hosted Domain 1 Email Account ✓ cPanel Control Panel ✓ Unlimited Hits ✓ PHP Version Selection ✓ PHP mail() support ✓ Full DNS Management ✓ Remote MySQL Support ✓ Python/Node.js Support	Unlimited Disk Space 250 GB Bandwidth 20 Hosted Domains 100 Email Accounts ✓ cPanel Control Panel ✓ Unlimited Hits ✓ PHP Version Selection ✓ PHP mail() support ✓ Full DNS Management ✓ Remote MySQL Support ✓ Python/Node.js Support	Unlimited Disk Space Unlimited Bandwidth Unlimited Hosted Domains Unlimited Email Accounts ✓ cPanel Control Panel ✓ Unlimited Hits ✓ PHP Version Selection ✓ PHP mail() support ✓ Full DNS Management ✓ Remote MySQL Support ✓ Python/Node.js Support
Create Now	Order Now	Order Now	Order Now

View more details on iFastNet's website

Want to know more about the plans? Or are you interested in Business Hosting, Reseller Hosting, VPS or Dedicated Servers? View all services and details on iFastNet's website.

Go to iFastNet

☞ Choisissez un nom de sub-domain et vérifiez avant de valider

Step 2: Choose a Domain Name

Enter the initial domain name for your account. You can add more domains after your account is created.

Subdomain

trentindv

Domain Extension

wuaze.com

← Back

Check Availability

⚠ Le mot de passe et l'identifiant sont générés automatiquement (notez le mot de passe et l'identifiant)

Step 3: Additional Information

Account Label

Website for trentindev.wuaze.com

A short description to help you identify the account.

Account Username

(generated automatically)

Used to login to FTP, MySQL, etc.

Account Password

.....



A unique password, between 8 and 15 characters, letters and numbers only.

Email Consent

(please select)

Is our supplier allowed to contact you about your hosting account?

← Back

+ Create Account

☞ Choisissez (**ou pas**) être contacté à propos de votre hébergement

Email Consent

I Approve

Is our supplier allowed to contact you about your hosting account?

☞ Dans votre profil :

- Choisissez votre langue de préférence
- Modifiez (si besoin) votre adresse mail
- Modifiez (si vous le désirez) l'authentification (avec ident-email) et l'auth double facteurs

PROFILE SETTINGS

Profile Information

Manage Email Address


Manage Authentication

Login History

Delete Profile

Your Profile

Profile Picture



Change your profile picture at [Gravatar](#).

Email Address and Password

Email Address

trentindev@gmail.com

Forum Name

To post in the community forum, you will need to set a forum name. This is the name that will be visible to other users.

Forum Name

Choose wisely! You can only change your forum name once per year.

Preferences

Language

French

Client Area Theme

Light

Choose which language you would prefer to use. Some systems support multiple languages and will use this setting.

PROFILE SETTINGS

Profile Information

Manage Email Address

Manage Authentication

Login History

Delete Profile

Manage Authentication

Enable and disable authentication methods for your InfinityFree account.

Password Login

Configure logging in with your email address and password. You can also enable two-factor authentication for extra security.

Login with Password

Not enabled

Enable Password

Two Factor Authentication

Only supported with password authentication

Enable Two-Factor Authentication

Social Logins

Login to your InfinityFree account with a social network of your choice.

Login with Google

Not enabled

Link Google

Login with GitHub

Linked to siladire@free.fr

Unlink GitHub

👉 Créer un nom de forum (obligatoire)

Forum Name


To post in the community forum, you will need to set a forum name. This is the name that will be visible to other users.

You cannot change your forum name right now. It can only be changed once per year.

Vous pourrez modifier tout cela plus tard.

Votre compte est créé

Your Accounts

 if0_39309459
Website for trentindev.wuaze.com

Active Accounts: 1 / 3

Il sera accessible sous trois jours maximum

Pour mon cas le site était disponible au bout de



It can take up to 72 hours for new domains to be accessible everywhere!

This is caused by DNS caching, which InfinityFree cannot control, but there are some workarounds you can try. [Learn more.](#)



Your domain is ready!

This page is loaded from the directory `"/htdocs"` of your InfinityFree hosting account. So you are ready to set up your website!

[Open Getting Started Guide](#)

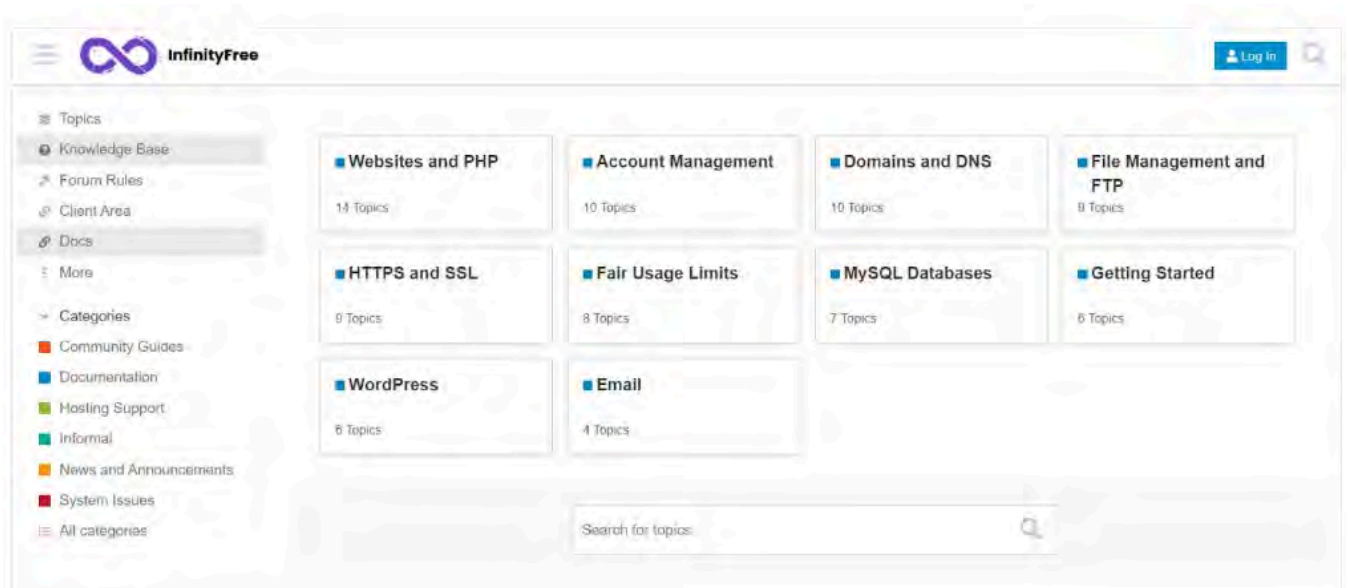
Ca fonctionne

Votre compte d'hébergement gratuit vous permet d'utiliser PHP et HTML pour votre site

Soutien

InfinityFree ne propose pas d'assistance personnalisée ; vous ne pouvez donc pas contacter d'assistance par téléphone ou par e-mail. L'adresse e-mail fournie n'est pas destinée à l'assistance.

Cependant, InfinityFree dispose d'une **solide base de connaissances et d'un forum rempli de questions et de solutions déjà résolues**. Le moteur de recherche d'InfinityFree est puissant, vous permettant de trouver rapidement la réponse. Si elle n'y figure pas, vous pouvez la poser et vous obtiendrez probablement une réponse.



Résumé :

1. Aller sur <https://infinityfree.com>
2. Créer un compte avec une adresse mail
3. Depuis le tableau de bord :
 - Créer un hébergement gratuit
 - Choisir un sous-domaine proposé (ex : `monprojet.epizy.com`)
 - Noter l'**URL**, les **identifiants FTP** et l'accès à **phpMyAdmin**

À noter : les élèves doivent activer leur compte par mail. Prévoir quelques minutes.

! Envoyez-moi le lien du site dès qu'il fonctionne.

Partie 2 : Typologie des architectures web

Objectifs :

- Comprendre les différences entre les types d'applications web
- Savoir reconnaître le type de projet qu'on va construire (ici : client-serveur PHP)

Objectif HTML

Pour qu'une page web soit affichée dans un navigateur, il faut du code HTML.

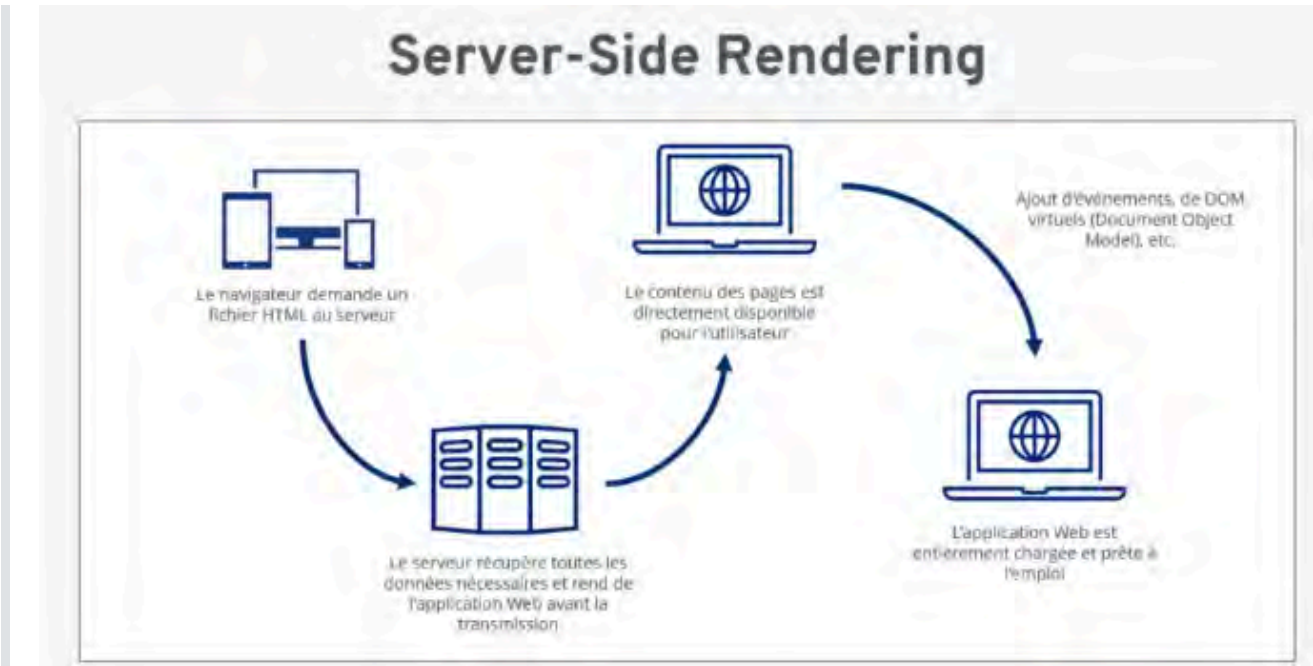
Ce code peut être **généré de deux façons** :

- **Côté serveur** : le serveur fabrique une page HTML complète, puis l'envoie au navigateur.
- **Côté client** : le navigateur construit la page à l'aide de JavaScript qui modifie le DOM.

Contenu vulgarisé :

Type	Description courte	Exemple
SSR (Server Side Rendering)	La page est générée sur le serveur en PHP ou NodeJS.	Votre projet
CSR (Client Side Rendering)	La page est statique, le contenu est injecté avec JavaScript	React, Vue.js
SPA (Single Page Application)	Une seule page, toute la navigation est gérée en JavaScript	Gmail

👉 Voir la description des différents types en annexe



Partie 3 : Mise en place du client FTP

Objectif :

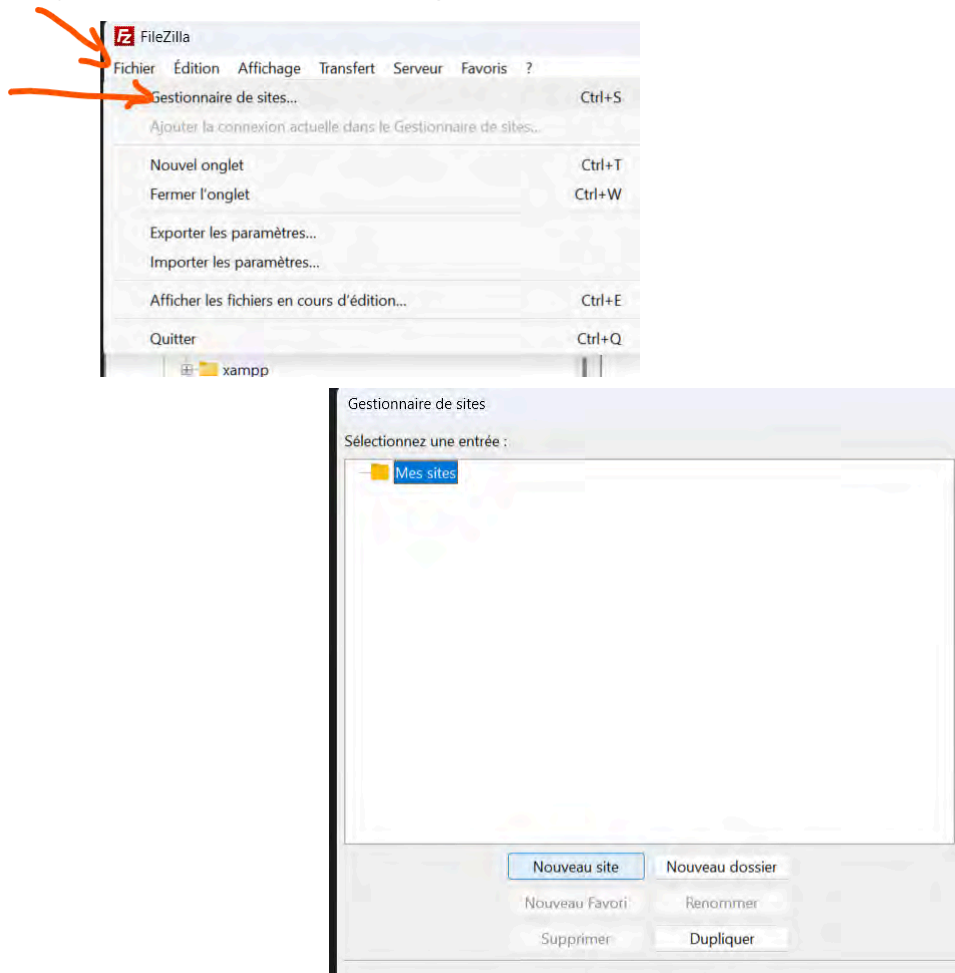
Apprendre à transférer des fichiers vers le serveur

- Soit avec un client FTP comme FileZilla
- Ou avec un client intégré à la solution d'hébergement

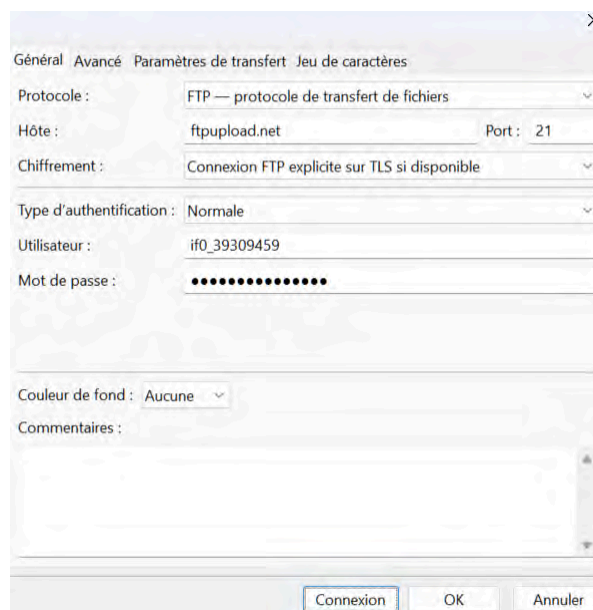
Outils :

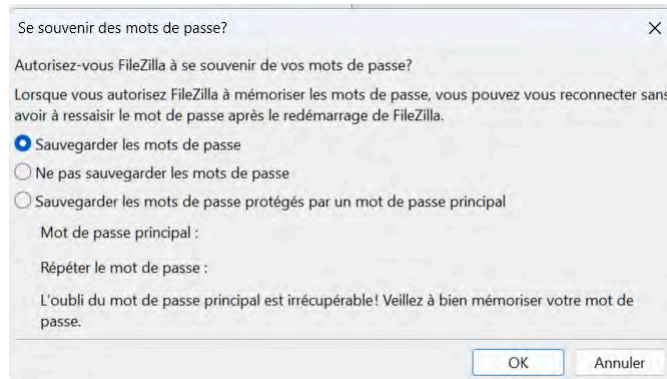
- 📄 Télécharger [FileZilla](#)

- 🖱️ Ajouter un nouveau site dans le gestionnaire de site :

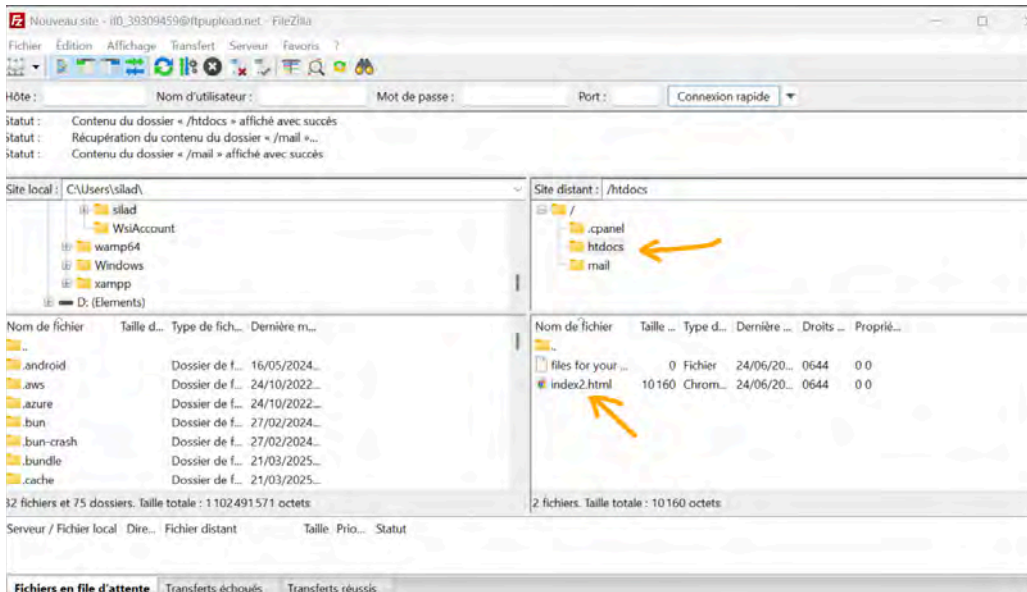


- - Hôte : ftpupload.net
 - Identifiants : ceux donnés par InfinityFree
 - Port : 21
 - Protocole : FTP simple





👉 Acceptez les warnings de connexion sans certificat



Le client FTP se déconnecte automatiquement assez souvent mais se reconnecte à la prochaine action ou suite à un rafraîchissement.

Exercice pratique :

- 👉 Avec un IDE positionnez vous dans votre dossier projet

```
c:\wamp64\www\intro_web\
```

- Créer un fichier `index.html` avec le contenu suivant :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Ma première page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Bienvenue sur mon site !</h1>
  <p>Ceci est ma première page en ligne.</p>
</body>
</html>
```

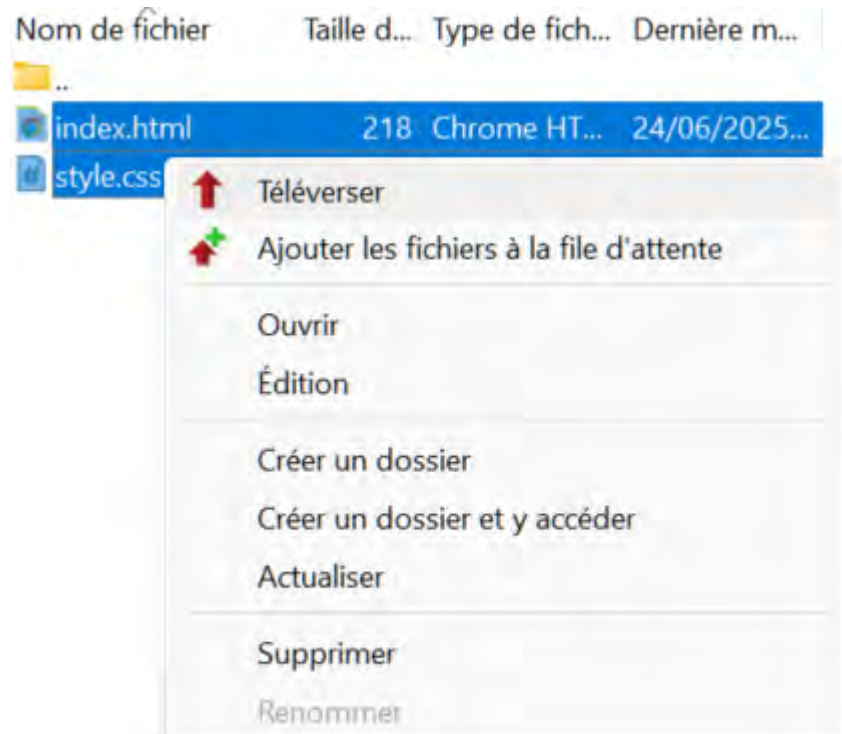
- 🗂 Dans le même dossier, créer un fichier `style.css`.

```
body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  color: #333;
  padding: 2rem;
}

h1 {
  color: #0066cc;
}
```

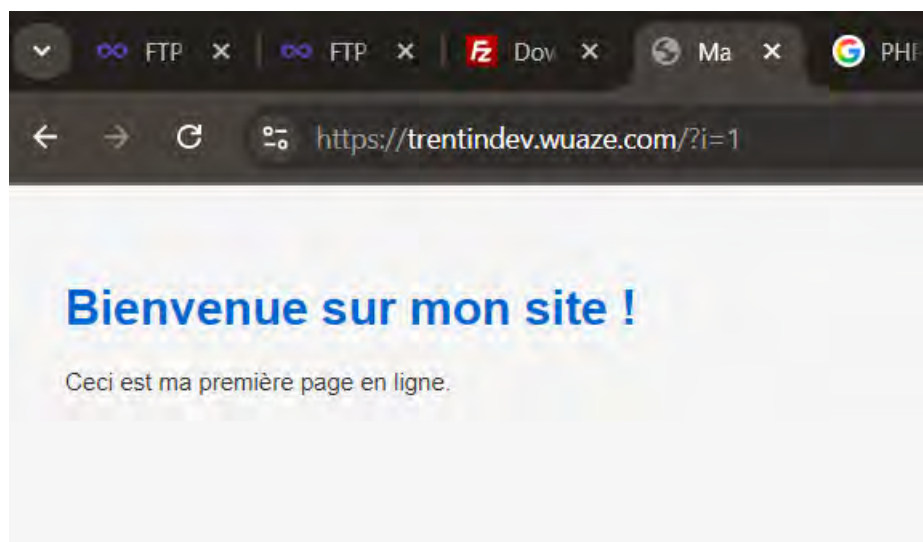
- 🗂 Transférer ces fichier dans le dossier `htdocs` sur infinityfree

Dans la colonne de gauche, naviguez jusqu'à votre dossier projet, puis cliquez droit sur les fichiers à télécharger



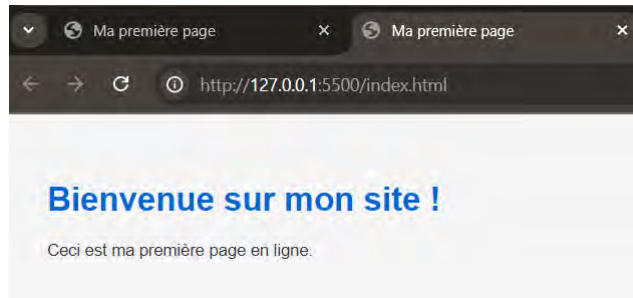
Nom de fichier	Taille ...	Type d...	Dernière ...	Droits ...	Proprié...
..					
files for your ...	0	Fichier	24/06/20...	0644	0 0
index.html	207	Chrom...	24/06/20...	0644	393094...
index2.html	10160	Chrom...	24/06/20...	0644	0 0
style.css	129	Fichier ...	24/06/20...	0644	393094...

- 🔄 Vérifier que l'URL du site affiche bien le contenu



Remarques

- Par défaut, le serveur est configuré pour "rendre" la page `index.html` ou `index.php`



Vous pouvez aussi installer une extension qui permet d'utiliser un client FTP directement dans Visual Studio Code, mais ce n'est pas toujours une pratique recommandée pour des problèmes de sécurité.

Mise en place du versionning

- 🐡 Dans le terminal de visual studio, vérifiez que vous êtes bien dans votre dossier projet :
- 🐡 Initialisez un dépôt git

```
git init
```

```
SORTIE  TERMINAL  PORTS  PROBLÈMES  CONSOLE DE DÉBOGAGE
PS C:\Users\silad\Documents\Projets\PHP_project\intro_web> git init
Initialized empty Git repository in C:/Users/silad/Documents/Projets/PHP_project/intro_web/.git/
```

- 🐡 Ajoutez les fichiers à l'index et committez

```
git status

>> On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
    index.html
    style.css
```

```
git add .

git commit -m "initialisation"
```

- 🐡 Créez une branche `dev` et basculez sur celle-ci

```
git branch dev
```

```
git switch dev
```

👉 Vérifiez

```
git status
```

```
>>
```

```
On branch dev
```

```
nothing to commit, working tree clean
```

⚠ **Consigne : Enrichir votre première page HTML**

Objectif : Utiliser les balises HTML de base pour structurer et enrichir votre page web.

Durée estimée : 10 minutes

Travail demandé :

À partir de votre fichier `index.html`, ajoutez les éléments suivants :

1. **Un titre de niveau 2** (`<h2>`) sous le titre principal (`<h1>`)
→ Ex. : "À propos de moi"
2. **Une liste à puces** (``) contenant au moins 3 éléments
→ Ex. : vos langages préférés, vos loisirs, vos films favoris...
3. **Une image** (``) libre de droits
→ Vous pouvez utiliser l'URL suivante :
`https://placehold.co/300x200`
(ou une autre image libre)
4. **Un lien cliquable** (`<a>`) vers un site que vous aimez
→ Ex. : [MDN Web Docs](https://developer.mozilla.org/fr/docs/Web)

Faites une capture d'écran

Bienvenue sur mon site !

À propos de moi

- J'aime coder en HTML, CSS et PHP
- Je joue de la guitare
- J'adore les jeux rétro

300 × 200

Pour apprendre le web, je recommande ce site : [MDN Web Docs](#)

Sitôt que l'apparence de votre site vous plait, téléversez les fichiers via FTP

Partie 4 : Création de la base de données

La persistance des données

Dans le cadre de ce TD client-serveur, la base de données va jouer un rôle essentiel : **stocker de manière durable les informations saisies par l'utilisateur** ce qui est une notion fondamentale du développement web

La plupart des applis modernes fonctionnent avec un serveur et une base.

But de la base de données dans ce projet

Dans ce TD, la base de données sert à **enregistrer de façon durable les informations saisies dans un formulaire web**, puis à les **réafficher dans une autre page**.

Concrètement, elle permettra :

- D'enregistrer un contact (nom, prénom, email, message)
- De conserver ces informations **même après avoir fermé la page**
- D'en afficher une **liste complète** accessible en ligne

Schéma du fonctionnement global

```
[Navigateur HTML]
|
| Remplissage du formulaire
V
[Script PHP (form.php)]
|
| Requête SQL (INSERT INTO)
V
[Base de données]
|
| Requête SQL (SELECT)
V
[Script PHP (liste.php)]
|
| HTML avec les résultats
V
[Navigateur - affichage de la liste]
```

Base de donnée et workflow de développement

Nécessité de travailler avec une base de donnée locale

InfinityFree comme toutes les solution SaaS, bloque les connexions MySQL venant de l'extérieur.
Seuls les **scripts hébergés sur leurs propres serveurs** (donc en ligne) peuvent accéder à la base de données.

Donc, **depuis votre machine locale (Wamp)** :

- ❌ Vous ne pouvez **pas** vous connecter sur la base de données distante, ce qui empêche d'avoir un workflow optimal
- ✅ Vous devez créer et utiliser **votre propre base MySQL locale** Il y aura donc deux configurations de projets distinct **une locale** et une distante.

Structure de la base de données (locale et distante)

Nous allons créer une table appelée `contacts` avec les champs suivants :

- `id` : identifiant unique de chaque contact
- `prenom`, `nom`, `email` : champs classiques d'un formulaire
- `message` : zone de texte libre
- `date_creation` : date automatique de l'enregistrement

Création de la base de donnée locale dans WampServer

Prérequis

- WampServer doit être **installé et lancé** (icône verte dans la barre des tâches)

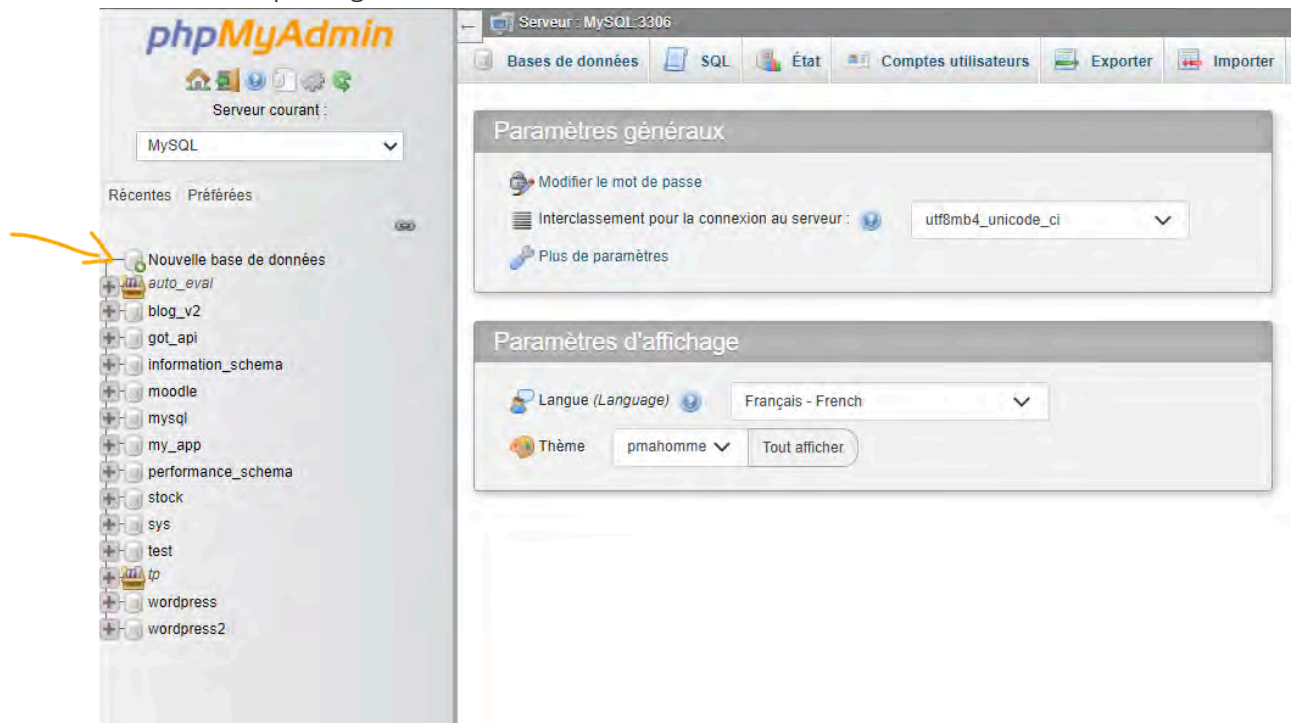
Étapes à suivre

1. Ouvrir phpMyAdmin

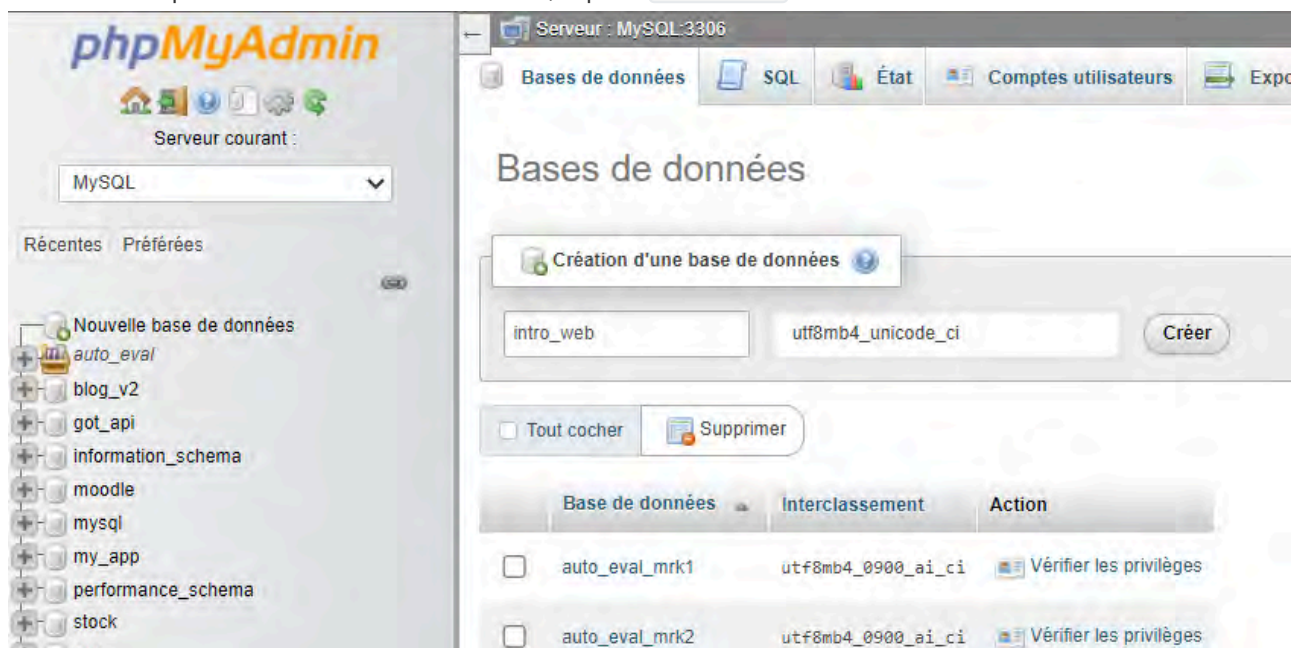
- Allez sur <http://localhost/phpmyadmin>
 - Identifiez-vous :
 - **Nom d'utilisateur** : `root`
 - **Mot de passe** : (*laisser vide*) par défaut sur Wamp
- Si vous les avez déjà modifié, mettez vos identifiants

2. Créer la base de données

- Une fois connecté, cliquez à gauche sur **Nouvelle base de données**



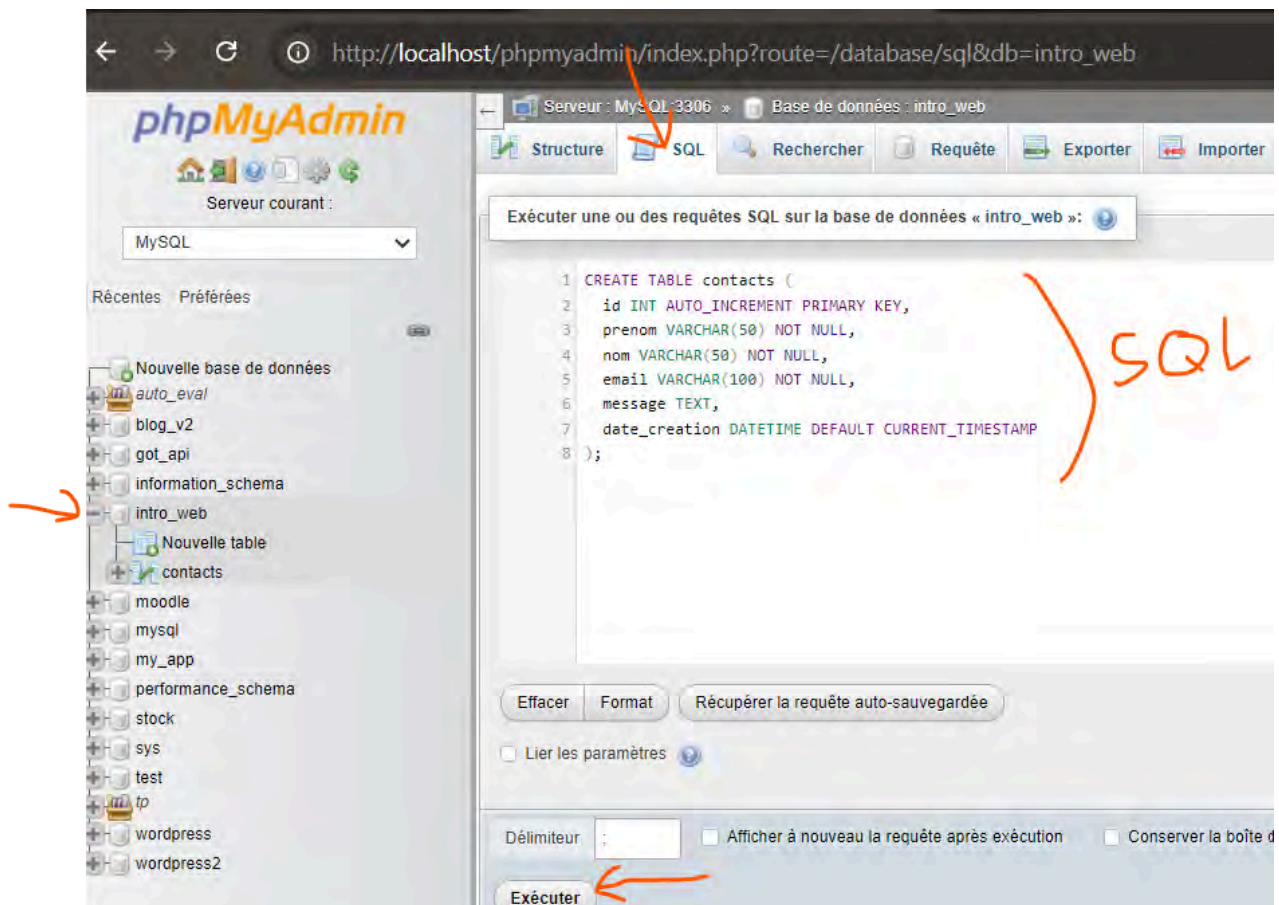
- Dans le champ **Nom de la base de données**, tapez : `intro_web`



- Choisissez le **classement** : `utf8mb4_general_ci` ou `utf8_general_ci`
- Cliquez sur **Créer**

✅ La base `intro_web` est maintenant créée !

3. Créer une table pour les contacts



- Cliquez sur votre base `intro_web` dans la colonne de gauche
- Cliquez sur **SQL** (onglet en haut)
- Collez le code suivant :

```
CREATE TABLE contacts (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  prenom VARCHAR(50) NOT NULL,  
  nom VARCHAR(50) NOT NULL,  
  email VARCHAR(100) NOT NULL,  
  message TEXT,  
  date_creation DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

- Cliquez sur **Exécuter**

✓ Votre table `contacts` est prête à recevoir des données.

Partie 5 - Introduction au PHP



Qu'est-ce que PHP ?

PHP (Hypertext Preprocessor) est un **langage de programmation côté serveur** conçu pour créer des **sites web dynamiques**.

Contrairement à HTML/CSS, qui décrivent l'affichage d'une page, **PHP exécute des instructions sur le serveur** avant que la page ne soit envoyée au navigateur.

Créé en 1995, il est aujourd'hui l'un des langages les plus utilisés dans le monde web, notamment parce qu'il est :

- **simple à apprendre**
- **gratuit et open source**
- **et compatible avec la majorité des hébergeurs web** (dont InfinityFree)

À quoi sert PHP concrètement ?

PHP est utilisé pour :

Application	Exemple concret
Gérer des formulaires	Traitement et enregistrement des données
Se connecter à une base de données	Insertion, lecture, modification (ex : MySQL)
Gérer des sessions et des connexions	Authentification, paniers, tableaux de bord
Générer du HTML dynamique	Afficher des données récupérées automatiquement
Envoyer des emails	Formulaires de contact, notifications
Construire des CMS	WordPress, Joomla, Drupal sont en PHP
Créer des APIs	Communication entre front et back

Exemple très simple

```
<?php
$nom = "Jean";
echo "Bonjour, " . $nom . " !";
?>
```

Cela génère la phrase **Bonjour, Jean !** dans le navigateur, **depuis le serveur**.À retenir

PHP est le langage **qui fait vivre les pages** : il **réagit** aux actions de l'utilisateur, **traite les données**, et **dialogue avec la base de données**.
C'est l'un des premiers outils pour passer d'un **site statique** à un **site dynamique**.

PHP vs JavaScript vs Python — Tableau comparatif

Vous avez déjà abordé le python et le JavaScript. Voilà quelques points de comparaison



Critère	PHP	JavaScript	Python
Rôle historique	Langage de serveur web	Langage de navigateur (client)	Langage généraliste
Fonction principale	Générer du HTML dynamique côté serveur	Gérer les interactions dans la page	Scripts, automatisation, IA, web, outils
Côté exécution	Serveur (fichier <code>.php</code>)	Client (dans le navigateur, <code>.js</code>)	Serveur ou local (fichier <code>.py</code>)
Syntaxe	Classique, verbeuse (<code>\$var;</code> , <code>echo</code>)	Souple mais parfois confuse	Très lisible et rigoureuse (indentation)
Utilisation web	Traitement de formulaire, base de données	DOM, formulaires, appels API, animations	Frameworks web (Flask, Django), back-end

Critère	PHP	JavaScript	Python
Popularité	Très utilisé pour le web (CMS, WordPress...)	Indispensable en web moderne	Très populaire en IA, automatisation, scripts
Paradigme principal	Impératif / Orienté Objet	Orienté événement, asynchrone possible	Multi-paradigme : impératif, objet, fonctionnel
Facilité d'apprentissage	Simple dans le web	Simple au début, plus complexe avec le temps	Très apprécié pour sa clarté
Hébergement facile	Oui (hébergeurs gratuits comme InfinityFree)	Non seul (nécessite Node.js ou framework)	Non sur hébergements classiques
Domaines forts	Web classique (formulaires, CMS)	Web moderne (React, animations, APIs)	IA, scripts système, traitement de données

Résumé

Langage	À retenir
PHP	Le plus utilisé pour les sites web dynamiques classiques , il génère du HTML et interagit avec la base de données côté serveur. Idéal pour débiter dans le back-end.
JavaScript	Langage indispensable côté client . C'est lui qui rend les pages interactives (menus, sliders, formulaires dynamiques). Il peut aussi fonctionner côté serveur avec Node.js .
Python	Langage généraliste très clair, utilisé dans de nombreux domaines (IA, scripts, automatisation, web). En web, il nécessite des frameworks comme Flask ou Django .

Exemples d'usage typique

- **PHP** : traitement d'un formulaire de contact sur un site.
- **JavaScript** : vérification des champs du formulaire avant envoi.
- **Python** : envoi d'un rapport automatique par mail chaque matin avec des données depuis une base.

Caractéristiques principales de la syntaxe PHP

Les balises PHP

Tout code PHP doit être écrit **entre balises spéciales** :

```
<?php
// code ici
?>
```

Le serveur exécute ce qu'il trouve **entre** `<?php ... ?>`, le reste (HTML) est envoyé tel quel au navigateur.

Les variables

- Une variable commence toujours par un `$` :

```
$nom = "Jean";  
$age = 25;
```

- Pas besoin de déclarer le type (PHP est faiblement typé)
- On peut stocker : des chaînes, des nombres, des tableaux, etc.

L'affichage

- La fonction la plus courante est `echo` :

```
echo "Bonjour, $nom";
```

- On peut aussi utiliser `print` (équivalent simplifié)

4. Les commentaires

```
// commentaire sur une ligne  
# Autre forme possible (comme en Bash)  
/* commentaire  
   sur plusieurs lignes */
```

Les instructions se terminent par un `;`

```
$nom = "Luc";    // OK  
echo $nom;      // OK
```

 Oublier le point-virgule provoque une **erreur fatale**.

6. Les structures de contrôle

Comme dans d'autres langages C-like (*qui ressemblent au langage C*) :

```
// condition if/else
if ($age >= 18) {
    echo "Majeur";
} else {
    echo "Mineur";
}
// La même chose mais avec un ternaire (plus concis et élégant)
echo $age >= 18 ? "Majeur" : "Mineur";

// Boucle for : de 0 à 4 avec un incrément de 1
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
```

Les fonctions

On définit une fonction avec `function` :

```
function direBonjour($prenom) {
    return "Bonjour, $prenom";
}
echo direBonjour("Laura");
```

Les tableaux

```
$fruits = ["pomme", "banane", "kiwi"];
echo $fruits[1]; // Affiche "banane"
```

On peut aussi utiliser des **tableaux associatifs** (clé,valeur) :

```
$personne = ["nom" => "Dupont", "age" => 30];
echo $personne["nom"];
```

Intégration HTML + PHP

PHP est souvent **mélangé à du HTML** :

```
<p>Bonjour <?php echo $nom; ?> !</p>
```

Fichiers `.php`

- Le code PHP est placé dans des fichiers avec l'extension `.php`
 - Il peut contenir aussi du HTML et du CSS
-

Fichiers de configuration

Les fichiers `config.php` permettent d'**éviter la répétition, de sécuriser votre code, et de mieux organiser votre projet**.

Ce sont des **outils simples mais puissants**, qu'on retrouve dans tous les projets web professionnels.

Le rôle des fichiers de configuration PHP

Ces fichiers sont utilisés pour **centraliser les paramètres de connexion à la base de données** (nom d'hôte, utilisateur, mot de passe, nom de la base).

- `config_local.php` est utilisé en **local**, avec WampServer (connexion à `localhost`)
- `config_infinity.php` est utilisé en **ligne**, sur InfinityFree (connexion à un serveur distant)

Avantages d'utiliser un fichier de configuration

Avantage	Explication
Centralisation	Toutes les infos de connexion sont au même endroit
Facilité de maintenance	On ne modifie la connexion qu'une seule fois, pas dans chaque fichier
Réutilisable	On peut inclure (<code>include</code>) le même fichier dans plusieurs scripts PHP
Séparation des environnements	On peut utiliser <code>config_local.php</code> en développement et <code>config_infinity.php</code> en production sans modifier le reste du code
Organisation professionnelle	C'est une bonne pratique utilisée dans la plupart des frameworks (Laravel, Symfony, etc.)

Précautions à prendre avec les fichiers de configuration

Risque	Précaution recommandée
Fuite d'identifiants	Ne jamais afficher ou exposer le contenu de <code>config.php</code> dans le navigateur
Fichier visible par erreur	Vérifier que le serveur est bien configuré pour exécuter le PHP et ne pas l'afficher
Mot de passe dans le code	Ne jamais laisser le fichier <code>config.php</code> sur un dépôt public (GitHub par exemple) sans le protéger
Fichier mal détecté	Toujours détecter automatiquement l'environnement (<code>localhost</code> ou domaine public) dans vos scripts

Bonnes pratiques

- Utiliser un script intelligent comme :

```
if (strpos($_SERVER['HTTP_HOST'], 'localhost') !== false) {  
    include 'config_local.php';  
} else {  
    include 'config_infinity.php';  
}
```

- **Ne pas inclure de fonctions ou de logique métier dans ces fichiers**, uniquement la connexion
- **Fermer la connexion avec** `mysqli_close()` une fois le traitement terminé



Créez un fichier `config_local.php`

Ce script :

- Initialise les paramètres de connexion à MySQL
- Tente de se connecter
- Laisse la **gestion des erreurs** au script qui inclura ce fichier (`include 'config_local.php';`)

```
<?php  
$host = 'localhost';  
$user = 'root';  
$password = '';  
$dbname = 'intro_web';  
  
$conn = mysqli_connect($host, $user, $password, $dbname);  
  
if (!$conn) {  
    // En cas d'erreur, on ne fait pas echo ici pour ne pas polluer les autres scripts  
    // On laisse les scripts inclure ce fichier et gérer les erreurs eux-mêmes  
}
```

Explications

`<?php` : Démarre un bloc de code PHP

`$host` = 'localhost' : Adresse du serveur MySQL : ici, on utilise le serveur **local** fourni par Wamp (localhost)

`$user` : Nom d'utilisateur MySQL en local

`$password` : Mot de passe vide par défaut pour `root` dans WampServer

`$dbname` : Nom de la base de données à laquelle on souhaite se connecter (elle doit déjà avoir été créée dans phpMyAdmin)

`$conn = mysqli_connect($host, $user, $password, $dbname);`

- Fonction PHP qui tente d'**ouvrir une connexion** à MySQL avec les infos données
- Si tout se passe bien, `$conn` contient un **objet de connexion**
- Si la connexion échoue, `$conn` vaut `false`

```
if (!$conn) { ... }
```

- On teste si la connexion a **échoué**
- Ici, on **ne fait rien volontairement**, car ce fichier est inclus dans d'autres scripts (`form.php`, `liste.php`, etc.)

Pourquoi ne rien afficher ici ?

Pour éviter que ce fichier **affiche un message d'erreur tout seul**, alors qu'un autre script s'en charge.

Cela permet de **garder le contrôle** sur l'affichage des erreurs.

Créez un fichier `config_infinity.php`

Utilisé pour la connexion distante

```
<?php
// Paramètres de connexion à la base de données
// Adaptez avec vos identifiants
$host = 'sql303.infinityfree.com';           // Exemple : sql303.infinityfree.com
$user = 'if0_XXXXXXX';                      // Votre Identifiant InfinityFree
$password = 'XXXXXXX';                      // Mot de passe MySQL
$dbname = 'if0_XXXXXXX_intro_web';          // Nom exact de la base

// Connexion
$conn = mysqli_connect($host, $user, $password, $dbname);

// Vérification de la connexion
if (!$conn) {
    die("Connexion échouée : " . mysqli_connect_error());
}
```

Explications

Élément	Rôle
Paramètres	Informations personnelles pour accéder à la base
<code>mysqli_connect()</code>	Tente la connexion avec ces infos
<code>if (!\$conn)</code>	Vérifie si la connexion a échoué
<code>die()</code>	Affiche un message d'erreur clair et arrête le script

Test en local

Pour vérifier que tout fonctionne :

📄 Créez un petit fichier `test_connexion.php` dans le dossier `www`

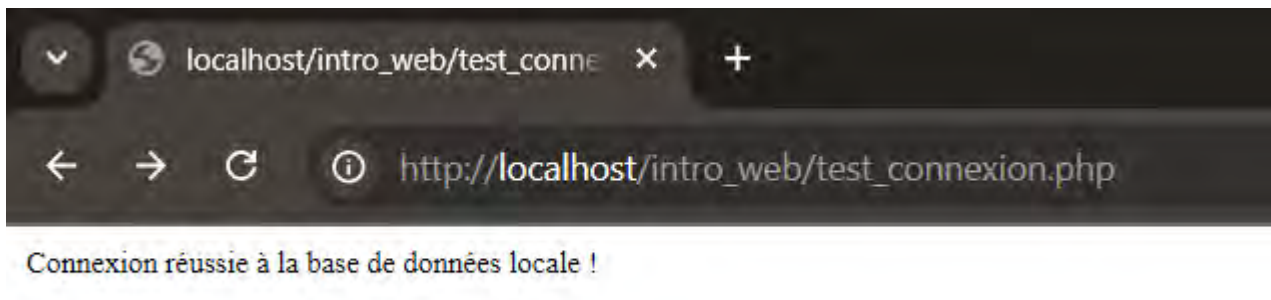
```
<?php
// on inclut le fichier config_local.php qui contient la connexion à la base
include 'config_local.php';

// si l'inclusion s'est bien passée, la variable $conn est déjà disponible
if ($conn) {
    echo "Connexion réussie à la base de données locale !";
} else {
    echo "Erreur de connexion : " . mysqli_connect_error();
}
```

La fonction `mysqli_connect_error()` fournit la cause de l'échec (mauvais identifiants, base inexistante, etc.).

Pour lancer le script, renseigner l'adresse suivante dans le navigateur.

`http://localhost/intro_web/test_connexion.php`



Erreurs courantes

En cas d'échec de connexion à la bdd locale

Type d'erreur	Cause fréquente	Message ou symptôme
Mauvais identifiant	Le nom d'utilisateur n'est pas <code>root</code> (rare en local)	<code>Access denied for user</code>
Mot de passe non vide	Certains Wamp sont configurés avec un mot de passe pour <code>root</code>	<code>Access denied for user 'root'@'localhost'</code>

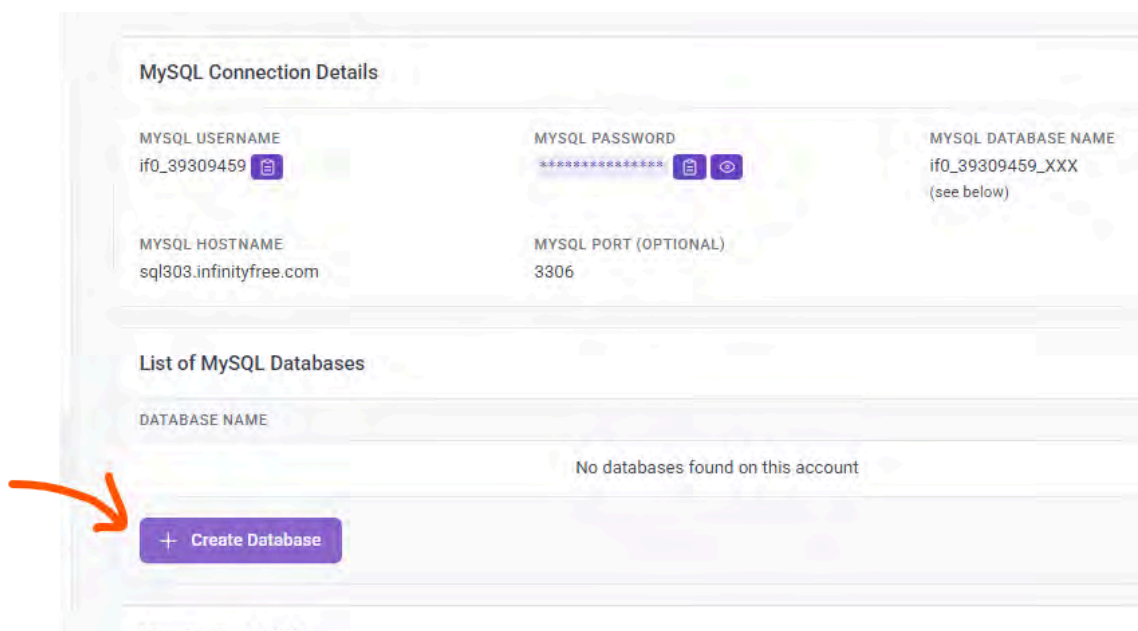
Type d'erreur	Cause fréquente	Message ou symptôme
Base non créée	La base <code>intro_web</code> n'existe pas dans phpMyAdmin	<code>Unknown database 'intro_web'</code>
Nom mal écrit	Une faute dans le nom de la base ou de l'hôte (ex : <code>localhost</code>)	<code>Unknown database</code> ou <code>php_network_getaddresses</code>
MySQL non démarré	Le service MySQL de Wamp n'est pas actif (icône orange ou rouge)	<code>Can't connect to MySQL server on 'localhost'</code>
Connexion déjà occupée	MySQL n'écoute pas sur le port 3306 ou est utilisé par un autre outil	Même message que ci-dessus
Fichier dans le mauvais dossier	Le script est exécuté en dehors de <code>www/</code>	<code>Not Found</code> ou page blanche

Nous allons maintenant créer une base de données distante.

Étapes détaillées pour créer une base de données sur InfinityFree

Étape 1 : Accéder au panneau de gestion

1. 🖱️ Connectez-vous à votre compte sur <https://app.infinityfree.net>
2. 🖱️ Cliquez sur votre hébergement actif
3. 🖱️ Cliquez sur **"MySQL Databases"**



Étape 2 : Créer la base

1. 🖱️ Choisissez un **nom pour la base** (ex : `intro_web`)
2. 🖱️ Cliquez sur **Create Database**

Create Database

Please choose the name of the new database.

Database Name

ifo_39309459_

MySQL Connection Details

MYSQL USERNAME ifo_39309459	MYSQL PASSWORD *****	MYSQL DATABASE NAME ifo_39309459_XXX (see below)
MYSQL HOSTNAME sql303.infinityfree.com	MYSQL PORT (OPTIONAL) 3306	

List of MySQL Databases

DATABASE NAME	ACTIONS
ifo_39309459_intro_web	phpMyAdmin Delete

3. 🖱️ **Notez les informations suivantes :**
 - **Nom de la base**
 - **Nom d'utilisateur** (souvent identique à l'utilisateur FTP)
 - **Mot de passe de base de données**
 - **Hôte MySQL** (ex : `sql303.infinityfree.com`)
 - **Port** ex : `3306`
 -

Étape 3 : Créer la table dans phpMyAdmin

1. 🖱️ Dans "MySQL Databases", cliquez sur **phpMyAdmin**
2. 🖱️ Une fois connecté, sélectionnez votre base à gauche

Serveur: sql303.infinityfree.com » Base de données: ifo_39309459_intro_web

Structure SQL Rechercher Requête Exporter Importer Opérations Procédures stockées Concepteur

⚠️ Aucune table n'a été trouvée dans cette base de données.

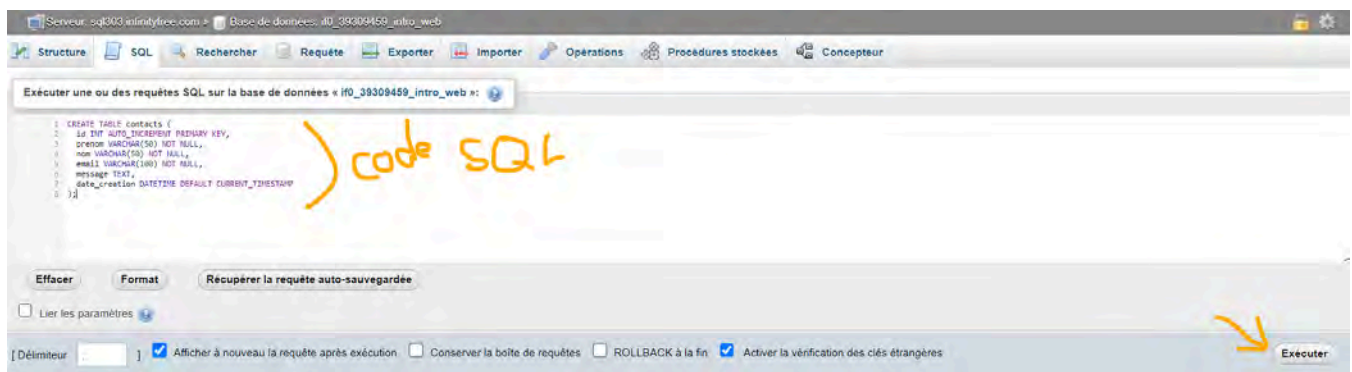
Nouvelle table

Nom : Nombre de colonnes :

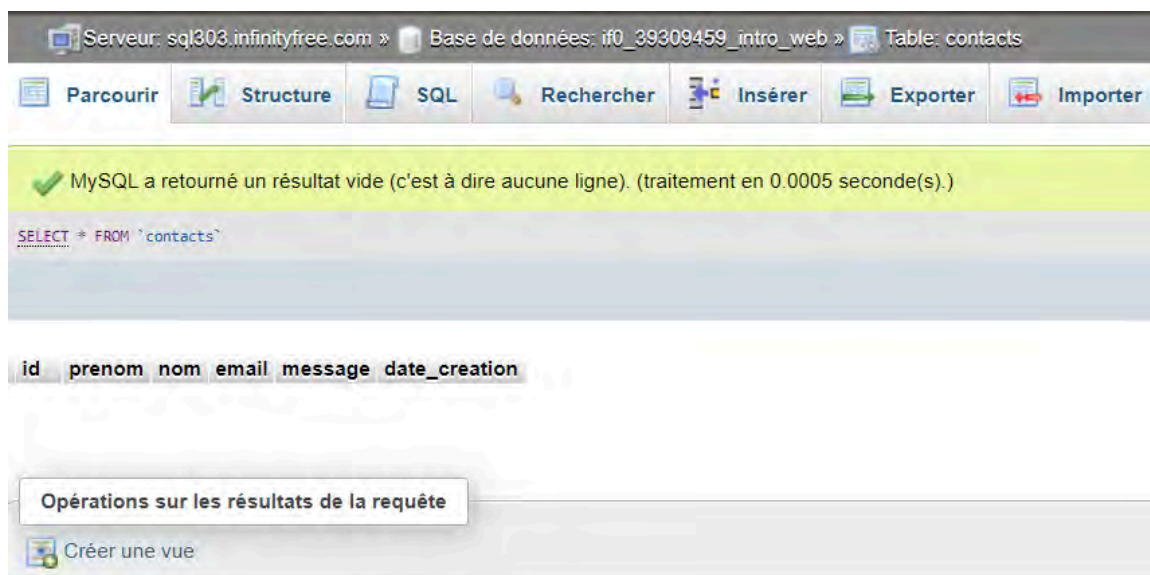
3. Cliquez sur **SQL**
4. Copiez-collez ce script SQL :

```
CREATE TABLE contacts (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  prenom VARCHAR(50) NOT NULL,  
  nom VARCHAR(50) NOT NULL,  
  email VARCHAR(100) NOT NULL,  
  message TEXT,  
  date_creation DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

1. Cliquez sur **Exécuter**



Vous avez maintenant une table prête à recevoir des données.



À retenir

- InfinityFree **n'accepte pas d'accès direct** à la base (vous devez passer par phpMyAdmin ou un script PHP).
- Le nom de la base est **précédé d'un préfixe** (par ex: if0_39309459_intro_web`) qui doit être utilisé dans votre script PHP.
- La connexion se fait via la fonction PHP `mysqli_connect()`. On le voit dans la partie suivante.

Partie 5 : Création d'un formulaire de contact

Objectifs

Créer une **petite application web** en PHP permettant de :

- Afficher un **formulaire de contact** (nom, prénom, email, message)
- **Enregistrer les données** dans une base MySQL
- **Afficher tous les contacts** enregistrés dans une autre page

Structure du projet dans le dossier `htdocs`

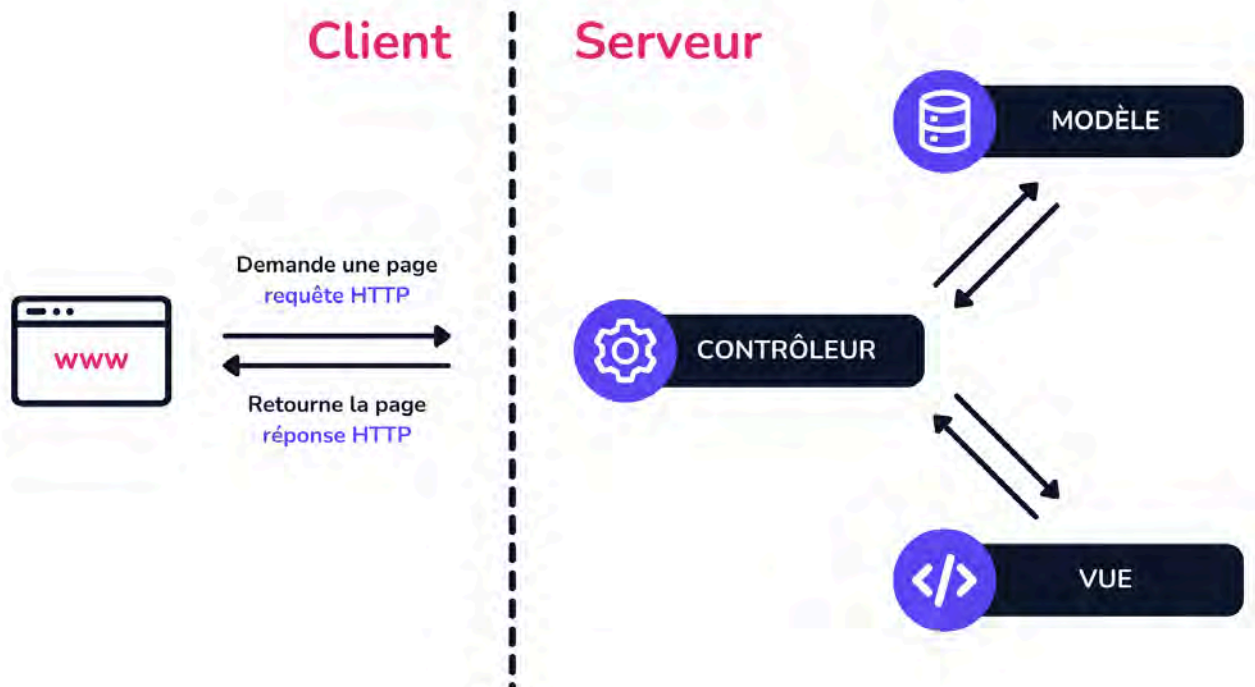
```
htdocs/      (ou /www)
|
├─ index.html      → Formulaire HTML
├─ form.php        → Enregistre les données dans la base
├─ liste.php       → Affiche les données enregistrées
├─ style.css       → Mise en page
├─ test_connexion.php → Vérifie la connexion à la base
├─ config_local.php → Connexion locale (wampServer)
└─ config_infinity.php → Connexion distante (InfinityFree)
```

Détail des fichiers

Fichier	Rôle
index.html	Page principale avec le formulaire HTML
form.php	Récupère les données du formulaire et les envoie à la base
liste.php	Lit les données de la base et les affiche dans un tableau
style.css	Ajoute du style à la page (typographie, couleurs, marges...)
config_XXX.php	Permet de centraliser les infos de connexion MySQL pour les inclure dans <code>form.php</code> et <code>liste.php</code> via <code>include</code>

Chaque fichier a une responsabilité claire : vue (HTML), traitement (PHP), données (SQL).

Cette structure est **idéale pour expliquer MVC plus tard** .



Le **MVC** (Model – View – Controller) est une architecture utilisée en développement web pour mieux organiser le code d'une application.

L'idée est de **séparer les responsabilités** :

- le **Modèle (Model)** gère les données et la logique métier (par exemple, les requêtes SQL),
- la **Vue (View)** s'occupe de l'affichage (HTML, CSS),
- le **Contrôleur (Controller)** fait le lien entre les deux (il récupère les données, appelle les bonnes fonctions, et affiche le bon contenu).

Cette structure permet de rendre le code **plus clair, plus modulaire et plus facile à maintenir**, car chaque partie est indépendante des autres. Le MVC est très utilisé dans les frameworks modernes comme Symfony, Laravel ou Ruby on Rails.

Modification du fichier `index.html`

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Formulaire de contact</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Formulaire de contact</h1>
  <form action="form.php" method="POST">
    <label>Prénom :</label><br>
    <input type="text" name="prenom" required><br><br>

    <label>Nom :</label><br>
```

```

<input type="text" name="nom" required><br><br>

<label>Email :</label><br>
<input type="email" name="email" required><br><br>

<label>Message :</label><br>
<textarea name="message"></textarea><br><br>

<button type="submit">Envoyer</button>
</form>
</body>
</html>

```

Explications :

Pour en savoir plus, voir Annexes

Élément	Rôle
<code><form></code>	Démarre le formulaire
<code>action</code>	Fichier PHP cible
<code>method</code>	Méthode d'envoi (<code>POST</code> ou <code>GET</code>)
<code>name</code>	Clé utilisée côté serveur
<code>required</code>	Champ obligatoire
<code>input</code> / <code>textarea</code>	Saisie utilisateur
<code>button type="submit"</code>	Envoi du formulaire

Modification du fichier `style.css`

```

body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  color: #333;
  max-width: 800px;
  margin: 2rem auto;
  padding: 1.5rem;
  line-height: 1.6;
}

h1, h2 {
  color: #0066cc;
  margin-bottom: 1rem;
}

form {

```

```
background-color: #fff;
padding: 1.5rem;
border: 1px solid #ddd;
border-radius: 8px;
}

label {
  display: block;
  margin-top: 1rem;
  font-weight: bold;
}

input[type="text"],
input[type="email"],
textarea {
  width: 100%;
  padding: 0.6rem;
  margin-top: 0.3rem;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

textarea {
  resize: vertical;
  height: 100px;
}

button {
  margin-top: 1.5rem;
  padding: 0.7rem 1.5rem;
  background-color: #0066cc;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #004a99;
}

table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 2rem;
  background-color: #fff;
}

th, td {
  padding: 0.8rem;
  border: 1px solid #ddd;
  text-align: left;
}
```

```
}  
  
th {  
  background-color: #e6f0ff;  
  color: #003366;  
}
```

Essayer de vous familiariser avec les sélecteurs CSS

<https://votre-site.wuaze.com/>

Formulaire de contact

Prénom :

Nom :

Email :

Message :

Envoyer

Ne le testez-pas maintenant !

Création du fichier `form.php`

```
<?php  
// Détection de l'environnement (local ou InfinityFree)  
if (strpos($_SERVER['HTTP_HOST'], 'localhost') !== false) {  
    include 'config_local.php';  
} else {  
    include 'config_infinity.php';  
}  
  
// Sécurisation des données reçues via POST  
$prenom = htmlspecialchars($_POST['prenom'] ?? '');  
$nom = htmlspecialchars($_POST['nom'] ?? '');
```



```

$email = htmlspecialchars($_POST['email'] ?? '');
$message = htmlspecialchars($_POST['message'] ?? '');

// Vérifie que les champs obligatoires ne sont pas vides
if ($prenom && $nom && $email) {
    // Requête SQL d'insertion
    $sql = "INSERT INTO contacts (prenom, nom, email, message)
            VALUES ('$prenom', '$nom', '$email', '$message')";

    // Exécution de la requête
    if (mysqli_query($conn, $sql)) {
        echo "✅ Données enregistrées avec succès !<br>";
        echo '<a href="liste.php">Voir la liste des contacts</a>';
    } else {
        echo "❌ Erreur SQL : " . mysqli_error($conn);
    }
} else {
    echo " ! Tous les champs obligatoires doivent être remplis.";
}

// Fermeture de la connexion
mysqli_close($conn);

```

Création de liste.php – Affichage des données enregistrées

```

<?php
// Détection de l'environnement
if (strpos($_SERVER['HTTP_HOST'], 'localhost') !== false) {
    include 'config_local.php';
} else {
    include 'config_infinity.php';
}

// Requête pour récupérer les contacts
$sql = "SELECT * FROM contacts ORDER BY date_creation DESC";
$result = mysqli_query($conn, $sql);

// Début de la page HTML
echo '<!DOCTYPE html><html lang="fr"><head><meta charset="UTF-8">';
echo '<title>Liste des contacts</title>';
echo '<link rel="stylesheet" href="style.css">';
echo '</head><body>';
echo '<h1>Liste des contacts enregistrés</h1>';

// Affichage du tableau si des résultats existent
if (mysqli_num_rows($result) > 0) {
    echo '<table>';
    echo '<tr><th>Prénom</th><th>Nom</th><th>Email</th><th>Message</th><th>Date</th></tr>';

```

```

while ($row = mysqli_fetch_assoc($result)) {
    echo "<tr>
        <td>{$row['prenom']}</td>
        <td>{$row['nom']}</td>
        <td>{$row['email']}</td>
        <td>{$row['message']}</td>
        <td>{$row['date_creation']}</td>
    </tr>";
}

echo '</table>';
} else {
    echo "<p>Aucun contact enregistré pour l'instant.</p>";
}

// Fin de la page HTML
echo '</body></html>';

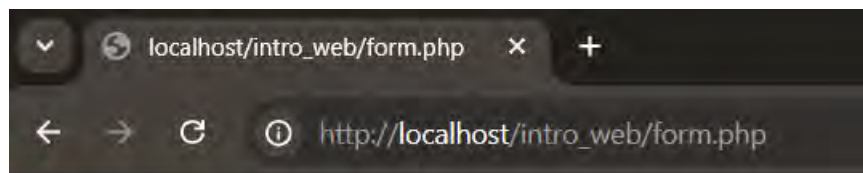
// Fermeture de la connexion
mysqli_close($conn);
?>

```

Remarques

- Ces fichiers doivent être placés dans le dossier `htdocs/intro_web` de Wamp ou sur InfinityFree
- Les **champs** `name` du formulaire doivent correspondre exactement aux colonnes SQL
- On utilise `htmlspecialchars()` pour éviter les injections HTML simples
- La connexion est **externalisée** dans les fichiers `config_*.php`, ce qui facilite la maintenance

⚠ Testez en local.



✓ Données enregistrées avec succès !
[Voir la liste des contacts](#)

Liste des contacts enregistrés

Prénom	Nom	Email	Message	Date
laurent	siladire	siladire@free.fr	coucou !	2025-06-25 13:41:35

⚠ **Testez sur la base de données distante**

nityFree cannot control, but there are some workarounds you can try. [Learn more.](#)

🔑 Control Panel

📁 File Manager

🗣️ Website Builder

📦 Script Installer

Domains

DOMAIN

trentindev.wuaze.com

➕ Add Domain

→ Manage

Account Details

USERNAME

if0_39309459

PASSWORD

Site Builders for if0_39309459

DOMAIN

trentindev.wuaze.com

ACTIONS

🗣️ Build Website

↑
votre
site

Formulaire de contact

Prénom :

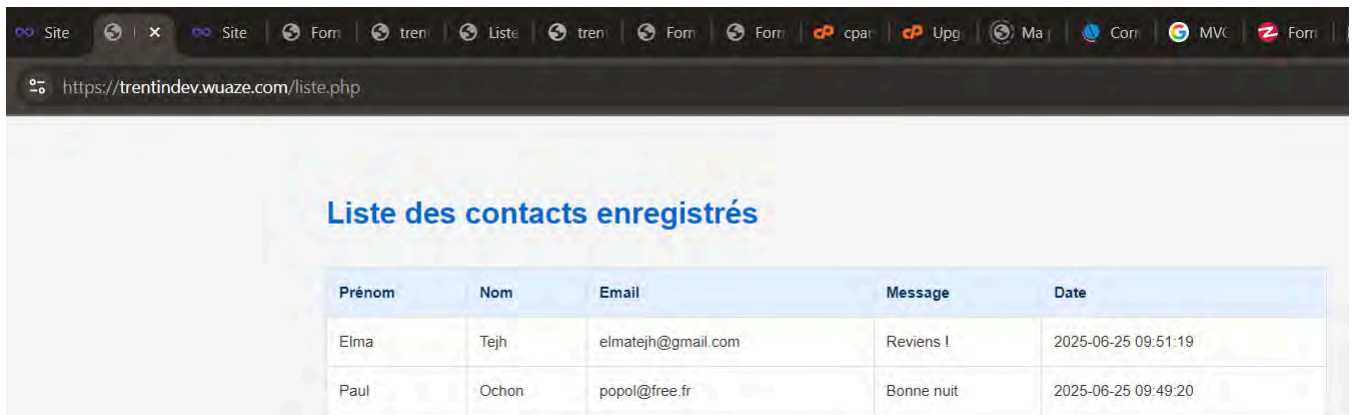
Nom :

Email :

Message :

Envoyer

<https://trentindev.wuaze.com/liste.php>



Prénom	Nom	Email	Message	Date
Elma	Tejh	elmatejh@gmail.com	Reviens !	2025-06-25 09:51:19
Paul	Ochoi	popol@free.fr	Bonne nuit	2025-06-25 09:49:20

Nous arrivons à la fin de la première partie

Vous avez maintenant un environnement de travail local et un environnement distant .

Vous pouvez valider votre branche git

```
git status
```

👉 Vérifiez que vous êtes bien sur la branche dev

```
>>
On branch dev
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html
```

```
modified:   style.css
```

Untracked files:

(use "git add <file>..." to include in what will be committed)

```
config_infinity.php
config_local.php
form.php
liste.php
nw_index.html
test_connexion.php
```

👉 Ajoutez tout à l'index

```
git add .
```

👉 Commitez vos changements

```
git commit -m "Première partie"
```

👉 Basculez sur la branche master

```
git switch master
```

👉 Fusionnez la branche dev sur la branche master

```
git merge dev
```

👉 Revenez sur la branche dev

```
git switch dev
```

⚠ ***Evaluation de la première partie***

Chaque élève :

- Envoie le **lien public de sa page** hébergée sur InfinityFree (URL)
- Envoie une **capture d'écran de sa base créée** avec la table `contact` visible dans phpMyAdmin

Exemple :

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0.0005 seconde(s).)

`ELECT * FROM `contacts``

☐ Tout afficher | Nombre de lignes : 25 | Filtrer les lignes : Chercher dans cette table | Trier sur l'index: A

Options

				id	prenom	nom	email	message	date_creation
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	Paul	Ochon	popol@free.fr	Bonne nuit	2025-06-25 09:49:20
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	Elma	Tejh	elmatejh@gmail.com	Reviens !	2025-06-25 09:51:19

☐ Tout cocher | Avec la sélection : Éditer | Copier | Supprimer | Exporter

Liens et ressources

Création de comptes sur infinityFree

- [infinityFree](#)

Faire un site Web HTML de base

Même si vous prévoyez d'utiliser une AI, c'est une bonne idée de vous familiariser avec le langage HTML :

- [Doc officielle sur MDN](#)
- [W3Schools](#)
- [OpenClassrooms](#)
- [Tuto interactif sur FreeCodeCamp](#)
- [Les formulaires sur MDN](#)

Langage Serveur : PHP

- [Manuel PHP](#)

- [PHP Sandbox](#) pour tester la syntaxe PHP sans installer d'environnement

SSR vs CSR

- [Grafikart.fr : Rendu côté client ou côté serveur ?](#)
- [SSR vs CSR Explained - FreeCodeCamp](#)

Annexes

Rendu côté serveur (SSR – Server Side Rendering)

Définition

Le SSR signifie que le **serveur est responsable de créer toute la page HTML** avant de l'envoyer. Cette page peut être :

- construite dynamiquement avec un langage serveur (PHP, Python, Ruby...),
- ou envoyée telle quelle si c'est un simple fichier HTML.

Avantages

- **Chargement initial rapide** : le navigateur affiche tout de suite une page prête.
- **Bon référencement (SEO)** : les moteurs de recherche voient directement le contenu HTML.
- **Liberté technologique** : on peut utiliser différents langages côté serveur.

Inconvénients

- À chaque clic, **toute la page est rechargée**, ce qui rend la navigation moins fluide.
- Impossible de **conserver des éléments actifs** entre deux pages (ex : un lecteur audio).
- **Le serveur travaille plus**, car il doit tout renvoyer à chaque requête.

Rendu côté client (CSR – Client Side Rendering)

Définition

Le CSR signifie que le **navigateur construit la page lui-même**, en exécutant du code JavaScript. Le serveur envoie seulement une **structure HTML minimale**, que le JavaScript remplit ensuite.

Avantages

- **Navigation fluide** : pas de rechargement complet de page.
- **Bonne réactivité** : les éléments peuvent changer sans recharger.
- **Facile à héberger** : la page principale est statique.

Inconvénients

- **Chargement initial plus lent**, car il faut attendre l'exécution du JavaScript.
- **Référencement plus difficile**, car le contenu n'est pas visible tout de suite par les robots.
- **Dépendance forte au JavaScript**, et peu de contrôle sur l'environnement d'exécution.

Autres approches hybrides

Dans la réalité, les applications modernes combinent souvent les deux approches pour profiter des avantages de chaque côté.

SSR avec amélioration progressive

- La page est d'abord rendue côté serveur.
- Ensuite, **du JavaScript améliore l'expérience utilisateur** : chargement partiel, interactions dynamiques, etc.
- Exemple d'outil : **AlpineJS**, pour des petits comportements sans framework complexe.

```
<div x-data="{ open: false }">
  <button @click="open = true">Voir le spoiler</button>
  <p x-show="open">Le héros meurt... ou pas !</p>
</div>
```

SSR avec hydratation partielle

- Le serveur génère une page HTML complète.
- **Certaines parties spécifiques** (comme les commentaires d'un article) sont rendues côté client, via du JavaScript.
- Exemple :

```
<post-comments post-id="42"></post-comments>
```

Cela permet de garder une bonne performance tout en rendant certains blocs dynamiques.

CSR avec rendu initial côté serveur

- On utilise un **serveur Node.js** pour générer une page HTML au moment de la première requête.
- Ensuite, le JavaScript prend le relais dans le navigateur.

Cette approche permet :

- Un **référencement efficace** (Google voit le HTML),
- Un **chargement initial plus rapide**.

Mais elle est plus complexe et nécessite des outils spécifiques :

- **Next.js** pour React
- **Nuxt** pour Vue
- **SvelteKit, Angular Universal**, etc.

CSR avec rendu dynamique (Dynamic Rendering)

- Le serveur détecte **qui visite le site**.
- Si c'est un humain, il envoie le JavaScript normal.
- Si c'est un moteur de recherche, il génère une **version HTML complète** avec un navigateur sans interface (headless browser).

Cela permet à un site JavaScript d'être bien référencé sans changer le fonctionnement pour les utilisateurs.

Qu'est-ce qu'AJAX ?

AJAX signifie **Asynchronous JavaScript and XML**.

C'est une **technique** qui permet à une page web de **communiquer avec un serveur sans être rechargée entièrement**.

Autrement dit : grâce à AJAX, on peut **envoyer ou récupérer des données** (comme depuis une base de données ou une API) **en arrière-plan**, sans que la page ne se réaffiche entièrement.

Concrètement, AJAX sert à :

- Envoyer un **formulaire sans recharger la page**
- Récupérer dynamiquement du contenu (ex : actualiser une liste, charger des commentaires...)
- Afficher des résultats en direct (ex : recherche instantanée)

Comment ça marche ?

1. Le **JavaScript** déclenche une action (clic, saisie...)
2. Une **requête HTTP** (GET ou POST) est envoyée au **serveur** via `XMLHttpRequest` ou `fetch()`
3. Le **serveur répond** (souvent avec des données au format **JSON**)
4. Le **JavaScript insère la réponse** dans la page sans la recharger

Exemple simple

```
fetch('liste.php')
  .then(response => response.text())
  .then(data => {
    document.getElementById('resultat').innerHTML = data;
  });
```

Cela va charger le contenu de `liste.php` dans un `<div id="resultat">` sans recharger la page.

À retenir

■ AJAX = une **page web plus réactive** : elle parle au serveur **sans se recharger**.

C'est ce qui rend le **CSR (Client Side Rendering)** fluide et moderne, en permettant de **charger uniquement ce qui change**.

Comprendre les formulaires HTML

Un **formulaire HTML** permet à un utilisateur de **saisir des informations** (texte, email, message, etc.) et de les **envoyer à un script serveur** (souvent en PHP) pour traitement.

Analyse du code

```
<form action="form.php" method="POST">
```

- `action="form.php"` : précise le **fichier PHP** qui va recevoir et traiter les données (ici, `form.php`)
- `method="POST"` : indique que les données seront **envoyées de façon invisible** (non visible dans l'URL). Plus sécurisé que `GET` pour les données personnelles.

Champs du formulaire

Chaque champ est associé à une **balise** `<label>` (titre lisible) et un **champ de saisie** (`<input>` ou `<textarea>`).

```
<input type="text" name="prenom" required>
```

- `type="text"` : champ de saisie classique
- `name="prenom"` : **nom du champ** (clé qui sera utilisée côté PHP avec `$_POST['prenom']`)
- `required` : empêche l'envoi du formulaire si le champ est vide

```
<input type="email" name="email" required>
```

- `type="email"` : le navigateur **vérifie automatiquement** que l'adresse est valide

```
<textarea name="message"></textarea>
```

- Champ de texte **multiligne**, pratique pour des messages ou commentaires

Bouton d'envoi

```
<button type="submit">Envoyer</button>
```

- Permet d'**envoyer le formulaire** au fichier PHP défini dans `action`

Résumé rapide à retenir

Élément	Rôle
<code><form></code>	Démarre le formulaire

Élément	Rôle
<code>action</code>	Fichier PHP cible
<code>method</code>	Méthode d'envoi (<code>POST</code> ou <code>GET</code>)
<code>name</code>	Clé utilisée côté serveur
<code>required</code>	Champ obligatoire
<code>input</code> / <code>textarea</code>	Saisie utilisateur
<code>button type="submit"</code>	Envoi du formulaire

Pour en savoir plus ([Les formulaires sur MDN](#)).
