

# FORMATION PHP:

## PHP INTERMÉDIAIRE

### COURS 2



## 1. Concaténation et affectation de variable en PHP

- Utilisation de l'opérateur de concaténation « . ».
- Concaténation avec affectation à l'aide de « .= ».
- Utilisation de retours à la ligne avec <br> et \n.

## 2. Imbrication des tableaux en PHP

- Création de tableaux imbriqués.
- Parcours de tableaux avec foreach.

## 3. Formulaires, Sessions, Cookies, Authentification et Sécurité

- Gestion des formulaires : méthodes GET et POST.
- Sécurisation des formulaires avec htmlspecialchars() et filter\_input().
- Protection contre les attaques CSRF avec des tokens de sécurité.
- Sessions et Cookies

## 4. Authentification utilisateur

- Hachage de mot de passe avec password\_hash().
- Vérification du mot de passe avec password\_verify().

## 5. Fonctionnement des applications modernes avec PHP

- Architecture basée sur API RESTful et microservices.
- Utilisation de PHP pour créer des API RESTful.

## 6. Base de données et types de bases de données

- Comparaison entre bases de données relationnelles (SQL) et non-relationnelles (NoSQL).
- Connexion à MySQL avec PHP (mysqli et PDO).
- Exécution de requêtes MySQL et sécurisation des données.
- Gestion des erreurs en PHP

## 7. Utilisation de try-catch pour capturer et gérer les erreurs.

- Configuration du niveau d'erreur avec error\_reporting().

## 7. Création de table avec phpMyAdmin

## 8. Exercices pratiques

# QUIZ PHP : POINTS À AMÉLIORER

- Balises PHP (31%)
- Fonction isset() (62%)
- Tableaux Associatifs (62%)
- Sécuriser les Données contre XSS (56%)
- Afficher la Configuration PHP (56%)
- Gestion des Fichiers avec \$\_FILES (62%)

```
  
<?php  
    // Code PHP ici  
    $nom = "Jean";  
    if (isset($nom)) {  
        echo "La variable nom est définie."  
    }  
  
    echo htmlspecialchars($donnee); // Protège contre XSS  
  
    phpinfo();  
  
    if ($_FILES['fichier']['error'] == 0) {  
        move_uploaded_file($_FILES['fichier']['tmp_name'], 'destination/');  
    }  
  
?>
```

# CONCATÉINATION ET AFFECTATION DE VARIABLE EN PHP

## Concaténation

- La concaténation en PHP permet d'assembler plusieurs chaînes de caractères en utilisant l'opérateur « . ».
- Cela est utile pour combiner des textes et des variables dans une même chaîne.

## Exemple : Concaténation



```
$prenom = "Jean";  
$nom = "Dupont";  
echo $prenom . " " . $nom; // Résultat : Jean Dupont
```

# CONCATÉINATION ET AFFECTATION DE VARIABLE EN PHP

## Concaténation avec Affectation

- PHP propose également l'opérateur « `.=` » pour ajouter une chaîne à une variable existante.

Exemple : Concaténation et affectation combinée



```
$nom_complet = "Jean";  
$nom_complet .= " Dupont"; // Résultat : Jean Dupont  
echo $nom_complet;
```

# EXEMPLE AVEC RETOUR À LA LIGNE EN PHP

- Dans les pages HTML, utilisez `<br>` pour générer un retour à la ligne visible dans le navigateur.



```
echo "Bonjour !<br>";  
echo "Bienvenue sur notre site.<br>";  
echo "Voici quelques informations.<br>";
```

- Écriture dans un fichier avec retour à la ligne « `\n` »




```
$file = fopen("message.txt", "w");  
fwrite($file, "Bonjour !\nBienvenue sur notre site.\nVoici quelques  
informations.");  
fclose($file);
```

# IMBRICATION DES TABLEAUX EN PHP

- Les tableaux en PHP peuvent être simples ou multidimensionnels, c'est-à-dire des tableaux contenant d'autres tableaux.
- Les tableaux imbriqués sont utiles pour représenter des structures complexes, comme des données hiérarchiques.

## Exemple : Tableau Imbriqué



```
$utilisateurs = [  
    ["nom" => "Jean", "age" => 25],  
    ["nom" => "Marie", "age" => 22],  
    ["nom" => "Pierre", "age" => 30]  
];  
  
echo $utilisateurs[0]['nom']; // Résultat : Jean  
echo $utilisateurs[2]['age']; // Résultat : 30
```

# IMBRICATION DES TABLEAUX EN PHP

Exemple :



```
$utilisateurs = [  
    ["nom" => "Jean", "age" => 25],  
    ["nom" => "Marie", "age" => 22],  
    ["nom" => "Pierre", "age" => 30]  
];  
// Accès à l'âge de Marie  
echo $utilisateurs[1]['age']; // Résultat : 22  
echo $utilisateurs[0]['nom']; // Résultat : Jean  
echo $utilisateurs[2]['age']; // Résultat : 30
```



# IMBRICATION DES TABLEAUX EN PHP

Exemple :



```
$classes = [
    "classe_1" => [
        ["nom" => "Alice", "notes" => ["maths" => 15, "anglais" => 18]],
        ["nom" => "Bob", "notes" => ["maths" => 10, "anglais" => 14]]
    ],
    "classe_2" => [
        ["nom" => "Charlie", "notes" => ["maths" => 12, "anglais" =>
16]],
        ["nom" => "Diane", "notes" => ["maths" => 17, "anglais" => 19]]
    ]
];

// Accéder à la note d'anglais de Bob
echo $classes["classe_1"][1]["notes"]["anglais"]; // Résultat : 14

// Accéder à la note de maths de Diane
echo $classes["classe_2"][1]["notes"]["maths"]; // Résultat : 17
```

# PARCOURS D'UN TABLEAU AVEC FOREACH EN PHP

- La boucle foreach est idéale pour parcourir facilement les éléments d'un tableau en PHP.
- Elle fonctionne pour tous types de tableaux, qu'ils soient associatifs ou simples, et permet d'accéder aux clés et aux valeurs sans connaître leur position exacte dans le tableau.

```
$fruits = ["Pomme", "Banane", "Orange"];

foreach ($fruits as $fruit) {
    echo $fruit . "<br>"; // Affiche chaque fruit suivi d'un retour à la
    ligne en HTML
}
Pomme
Banane
Orange

$personne = [
    "nom" => "Jean",
    "age" => 30,
    "profession" => "Développeur"
];

foreach ($personne as $cle => $valeur) {
    echo $cle . ": " . $valeur . "<br>"; // Affiche chaque clé et valeur
    avec un retour à la ligne
}
nom: Jean
age: 30
profession: Développeur

$produits = [
    ["nom" => "Ordinateur", "prix" => 750],
    ["nom" => "Téléphone", "prix" => 500],
    ["nom" => "Tablette", "prix" => 300]
];

foreach ($produits as $produit) {
    echo "Produit: " . $produit["nom"] . " - Prix: " . $produit["prix"] .
    "€<br>";
}
Produit: Ordinateur - Prix: 750€
Produit: Téléphone - Prix: 500€
Produit: Tablette - Prix: 300€
```

# FORMULAIRES, SESSIONS, COOKIES, AUTHENTIFICATION, MYSQL ET SÉCURITÉ

## Objectifs :

- Comprendre la gestion des formulaires en PHP.
- Utiliser les sessions et cookies pour la gestion des utilisateurs.
- Implémenter un système d'authentification.
- Connecter PHP à une base de données MySQL et sécuriser les requêtes.

# GESTION DES FORMULAIRES AVEC PHP

- Un formulaire permet d'envoyer des données à un script PHP.
- Méthodes : GET (visible dans l'URL) et POST (non visible).
- Le traitement des données se fait avec les superglobales \$\_GET et \$\_POST.

Exemple : Formulaire simple (méthode POST):

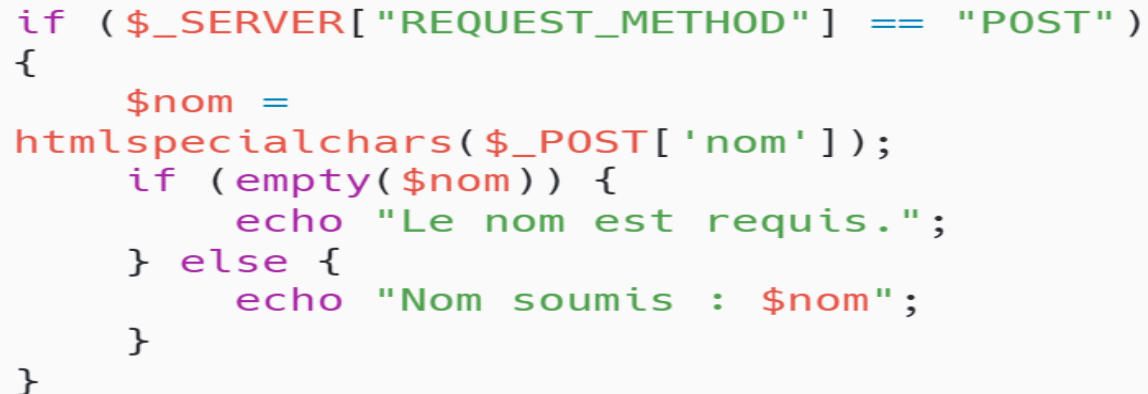


```
<form action="traitement.php"
method="POST">
  Nom : <input type="text" name="nom"><br>
  <input type="submit" value="Envoyer">
</form>
```

# SÉCURISER LES FORMULAIRES

- Utilisez htmlspecialchars() pour éviter l'injection de code HTML.
- Toujours valider et filtrer les données envoyées par l'utilisateur.

Exemple : Validation des données



```
if ( $_SERVER[ "REQUEST_METHOD" ] == "POST" )
{
    $nom =
htmlspecialchars( $_POST[ 'nom' ] );
    if ( empty( $nom ) ) {
        echo "Le nom est requis.";
    } else {
        echo "Nom soumis : $nom";
    }
}
```

# PROTECTION CONTRE LES ATTAQUES CSRF

- Les attaques CSRF (Cross-Site Request Forgery) peuvent être évitées avec un token de sécurité.
- Générer un « token » unique pour chaque formulaire et le vérifier à la soumission.

Exemple : Implémentation d'un token CSRF

```
// Génération du token
session_start();
$_SESSION['token'] =
bin2hex(random_bytes(32));
?>
<form action="traitement.php"
method="POST">
    <input type="hidden" name="token"
value="<?php echo $_SESSION['token']; ?>">
    Nom : <input type="text" name="nom"><br>
    <input type="submit" value="Envoyer">
</form>
```

# EXERCICE : FORMULAIRE SÉCURISÉ

## Objectif :

- Créer un formulaire de connexion sécurisé.
- Créez un formulaire avec des champs "nom d'utilisateur" et "mot de passe".
- Ajoutez une validation des données saisies et une protection CSRF.
- Affichez un message d'erreur si des champs sont manquants ou invalides.



Exercice 1.txt



Exercice 1 RSP.txt

# INTRODUCTION AUX SESSIONS EN PHP

- Les sessions permettent de stocker des informations sur l'utilisateur de manière persistante (au-delà d'une page).
- Utilisation de `session_start()` pour démarrer une session.
- Supprimer toutes les variables de session avec `$_SESSION = array();`.
- Détruire la session en appelant `session_destroy()`.

## Exemple : Utilisation de sessions



```
session_start();
$_SESSION['nom_utilisateur'] =
"JeanDupont";
echo "Utilisateur connecté : " .
$_SESSION['nom_utilisateur'];
```



# INTRODUCTION AUX COOKIES

Les cookies sont des petits fichiers stockés sur l'ordinateur de l'utilisateur par le navigateur web.

- Les cookies sont stockés dans le navigateur de l'utilisateur et peuvent être utilisés pour mémoriser des informations.
- Utilisation de `setcookie()` pour définir un cookie.

Exemple : Création d'un cookie



```
setcookie("nom_utilisateur", "JeanDupont", time() + 3600);  
// Expire après 1 heure
```

# SESSIONS VS COOKIES

## Sessions :

- Stockées côté serveur, plus sécurisées, expirent à la fermeture du navigateur (sauf session persistante).

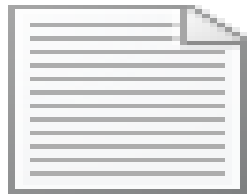
## Cookies :

- Stockés côté client (navigateur), utilisés pour mémoriser des informations sur une longue période (expiration personnalisable).

# EXERCICE : SYSTÈME DE SESSION ET COOKIES

## Objectif :

- Implémenter un système de session et de cookies.
- Créez une page de connexion.
- Utilisez une session pour garder l'utilisateur connecté.
- Ajoutez une case "Se souvenir de moi" pour créer un cookie et mémoriser l'utilisateur.



Exercice 2.txt



Exercice 2 - RSP.txt

# AUTHENTIFICATION UTILISATEUR

- Un système d'authentification consiste à vérifier si un utilisateur a des identifiants valides.
- Utilisation des sessions pour gérer l'état de connexion.
- Le mot de passe doit être stocké de manière sécurisée (haché).

Hachage de mot de passe avec password\_hash():

- password\_hash() permet de hacher un mot de passe de manière sécurisée.
- Utilisation de password\_verify() pour vérifier si un mot de passe correspond au hash.



```
$mot_de_passe_hache = password_hash("monMotDePasse", PASSWORD_DEFAULT);  
if (password_verify("monMotDePasse", $mot_de_passe_hache)) {  
    echo "Mot de passe correct.";  
} else {  
    echo "Mot de passe incorrect.";  
}
```

# FONCTIONNEMENT DES APPLICATIONS MODERNES AVEC PHP

Les applications modernes utilisent souvent une architecture basée sur des API (Application Programming Interface) et des microservices pour interagir avec diverses parties de l'application de manière modulaire.

PHP peut construire des API RESTful qui permettent à différents clients (applications web, mobiles) de communiquer avec le serveur.

## Principes clés :

- API RESTful : Permet l'échange de données en utilisant des requêtes HTTP (GET, POST, PUT, DELETE).
- Microservices : Architecture décomposant une application en petits services autonomes, chacun ayant sa propre responsabilité.
- Base de données : Généralement gérée via des ORM (Object-Relational Mapping) pour faciliter l'interaction avec les données.

## Exemple concret :

- Une application e-commerce où :
  - Le service d'authentification gère les utilisateurs.
  - Le service de produits fournit des informations sur les articles.
  - Le service de commande traite les achats.

# SCHÉMA DE L'ARCHITECTURE MODERNE

Client (Web/Mobile) : Interagit avec l'application via des requêtes HTTP.

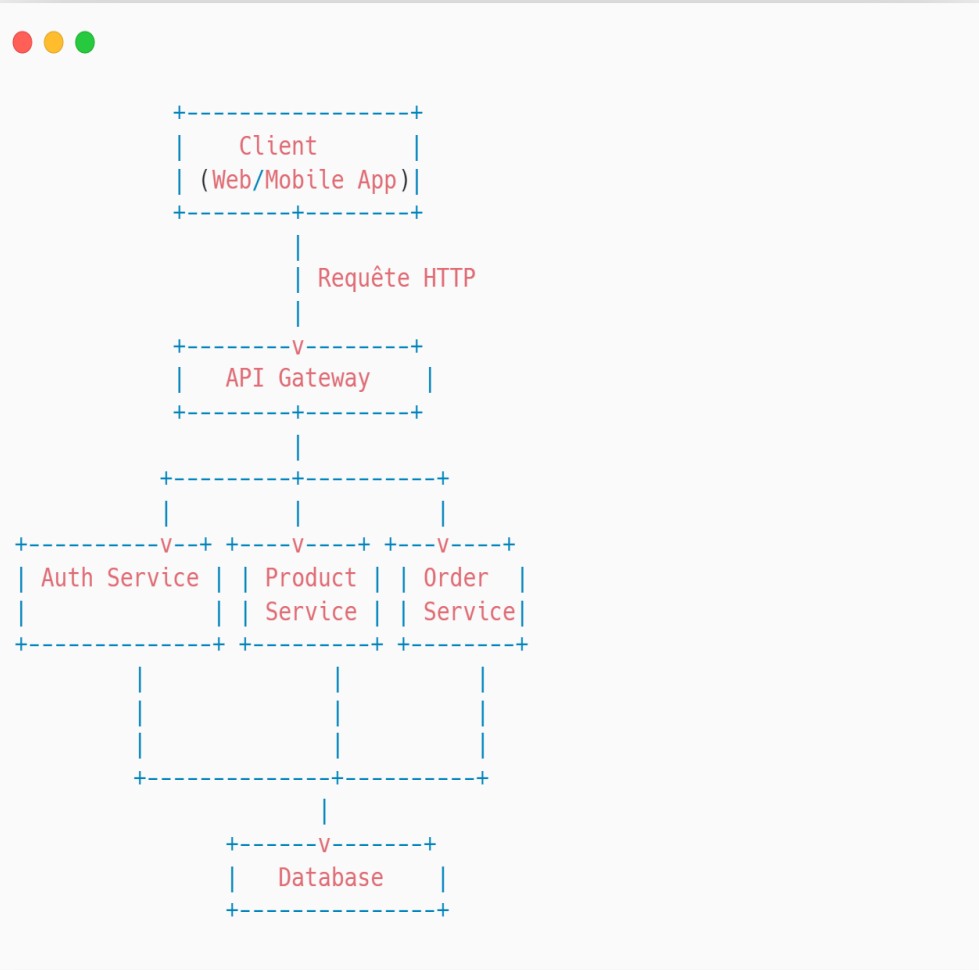
API Gateway : Route les requêtes vers les services appropriés.

Microservices :

- Service d'authentification : Gestion des utilisateurs.
- Service de produits : Récupération des informations de produits.
- Service de commandes : Traitement des achats et des paiements.

Base de données : Stockage des données pour chaque service, souvent en utilisant des bases de données différentes (SQL, NoSQL).

# SCHÉMA D'ARCHITECTURE MODERNE



Avantages de cette architecture :

- **Modularité** : Chaque service est indépendant et peut être développé ou mis à jour séparément.
- **Scalabilité** : Les services peuvent être mis à l'échelle individuellement en fonction des besoins.
- **Sécurité** : L'API Gateway permet de centraliser la gestion de la sécurité.

# BASE DE DONNÉES ET TYPES DE BASE DE DONNÉES

Les bases de données sont essentielles pour stocker, gérer et récupérer des données de manière efficace. On peut les classer en plusieurs catégories, chacune ayant ses propres caractéristiques et usages.

Types de Bases de Données :

- Bases de données relationnelles (RDBMS)
- Bases de données non relationnelles (NoSQL)
- Bases de données orientées objet

Bases de données relationnelles (RDBMS) :

- Utilisent des tables pour structurer les données.
- Les relations entre les données sont définies par des clés primaires et étrangères.
- Exemples :MySQL : Populaire pour les applications web, open-source. PostgreSQL, SQLite, Oracle DataBase...



# BASE DE DONNÉES ET TYPES DE BASE DE DONNÉES

## Bases de données non relationnelles (NoSQL) :

- Ne reposent pas sur un modèle de table fixe.
- Conviennent pour des données non structurées ou semi-structurées.
- Exemples :
  - MongoDB : parfait pour des données JSON.
  - Cassandra : Conçue pour gérer de grandes quantités de données distribuées.
  - Redis : Base de données clé-valeur, souvent utilisée pour le caching/Redis est idéal pour des applications nécessitant un accès rapide aux données.

## Bases de données orientées objet :

- Intègrent des concepts de programmation orientée objet pour stocker des données sous forme d'objets.
- Exemples :
  - db4o : Conçu pour les applications Java et .NET.

# BASE DE DONNÉES ET TYPES DE BASE DE DONNÉES

- Structure : Les données sont organisées en tables composées de lignes et de colonnes.
- Langage de requête : Utilisent SQL (Structured Query Language) pour les opérations CRUD (Create, Read, Update, Delete).
- Intégrité des données : Les contraintes d'intégrité assurent la cohérence des données à travers les relations.
- Transactions : Supportent les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) pour garantir la fiabilité des opérations.



Table: Utilisateurs

ID	Nom	Email
1	Alice	alice@email.com
2	Bob	bob@email.com

Table: Commandes

ID	UtilisateurID	Produit
1	1	Ordinateur
2	2	Téléphone

Dans cet exemple,  
la table Commandes a une clé étrangère  
« UtilisateurID »  
qui référence la table Utilisateurs,  
illustrant la relation entre les deux tables.

# CONNEXION À MYSQL AVEC PHP

- PHP offre deux interfaces pour interagir avec MySQL : mysqli et PDO.
- Les connexions doivent être sécurisées pour éviter les attaques.

Exemple : Connexion à une base de données avec mysqli



```
$conn = new mysqli("localhost", "utilisateur", "mot_de_passe",  
"ma_base");  
if ($conn->connect_error) {  
    die("Échec de connexion : " . $conn->connect_error);  
}  
echo "Connexion réussie";
```

# REQUÊTES MYSQL EN PHP

- Utilisation de requêtes SQL pour interagir avec la base de données (insertion, sélection, mise à jour, suppression).
- Les requêtes préparées protègent contre les injections SQL en séparant le code SQL des données utilisateur.

Exemple : Requête SQL simple (insertion)

```
● ● ●  
  
$sql = "INSERT INTO utilisateurs (nom_utilisateur, mot_de_passe) VALUES  
( 'Jean', 'motdepasse' )";  
if ( $conn->query($sql) === TRUE ) {  
    echo "Nouvel utilisateur ajouté";  
} else {  
    echo "Erreur : " . $conn->error;  
}
```

Exemple : Utilisation de requêtes préparées

```
● ● ●  
  
$stmt = $conn->prepare("INSERT INTO utilisateurs (nom_utilisateur,  
mot_de_passe) VALUES (?, ?)");  
$stmt->bind_param("ss", $nom_utilisateur, $mot_de_passe);  
$nom_utilisateur = "Jean";  
$mot_de_passe = password_hash("motdepasse", PASSWORD_DEFAULT);  
$stmt->execute();
```

# SÉCURISER L'AUTHENTIFICATION

- Valider les données reçues du formulaire.
- Utiliser les requêtes préparées pour vérifier les identifiants.



```
$stmt = $conn->prepare("SELECT mot_de_passe FROM utilisateurs WHERE  
nom_utilisateur = ?");  
$stmt->bind_param("s", $nom_utilisateur);  
$nom_utilisateur = $_POST['nom_utilisateur'];  
$stmt->execute();  
$stmt->bind_result($mot_de_passe_hache);  
$stmt->fetch();  
if (password_verify($_POST['mot_de_passe'], $mot_de_passe_hache)) {  
    echo "Connexion réussie.";  
} else {  
    echo "Identifiants incorrects.";  
}
```

# GESTION DES ERREURS EN PHP

- Utiliser try-catch pour capturer et gérer les erreurs.
- Utiliser `error_reporting()` pour configurer le niveau d'erreur.

Exemple : Gestion des exceptions



```
try {  
    // Code susceptible de provoquer une erreur  
} catch (Exception $e) {  
    echo 'Erreur capturée : ', $e->getMessage();  
}
```

# CRÉATION D'UNE TABLE AVEC PHPMYADMIN

Ouvrir phpMyAdmin :

- Accédez à <http://localhost/phpmyadmin>.
- Sélectionner la base de données : Cliquez sur la base de données où vous souhaitez créer la table.
- Créer une nouvelle table :
  - Dans le champ "Créer une table", entrez le nom de la table (par exemple, utilisateurs).
  - Spécifiez le nombre de colonnes (par exemple, 3).
  - Cliquez sur "Exécuter".
- Définir les colonnes : Remplissez les informations pour chaque colonne (nom, type, longueur, etc.) et définissez une clé primaire si nécessaire.
- Enregistrer : Cliquez sur "Enregistrer" pour créer la table.
- Création de table via script:

```
CREATE TABLE utilisateurs (  
    id INT(11) AUTO_INCREMENT PRIMARY KEY,  
    nom VARCHAR(100) NOT NULL,  
    email VARCHAR(100) NOT NULL UNIQUE  
)
```

# EXERCICE

1. Utilisez phpMyAdmin pour créer une table « utilisateurs » avec les colonnes id, nom, password .
2. Créez également une table commandes via un script PHP pour renforcer vos compétences.
3. Structure des fichiers :

```
auth-service/|
|— config/
|   |— database.php|
|— functions/
|   |— user_functions.php
|   |— auth_functions.php|
|— services/
|   |— auth_service.php
|   |— product_service.php (simulé)
|   |— order_service.php (simulé)|
|— api_gateway.php|
|— index.php
```

4. Configuration de la base de données
  - i. Complétez le fichier config/database.php avec vos informations de connexion
5. Implémentation des fonctions d'authentification
  - I. Complétez les fonctions dans functions/auth\_functions.php et services/auth\_service.php
  - II. Ajoutez une vérification pour voir si l'utilisateur existe déjà avant l'enregistrement
  - III. Implémentez la génération d'un token JWT simple pour la fonction de connexion
6. Test de l'API
  - I. Utilisez un outil comme Postman ou cURL pour tester les endpoints /auth avec les actions 'login' et 'register'
    1. Pour s'enregistrer :POST <http://votre-serveur/auth>
      1. action=register&username=test&password=password123
    2. Pour se connecter :POST <http://votre-serveur/auth>
      1. action=login&username=test&password=password123



# SUITE EXERCICE

Créez un fichier login.html

- Un formulaire simple avec des champs pour le nom d'utilisateur et le mot de passe.
- Du CSS intégré pour un style basique.
- Un script JavaScript qui gère la soumission du formulaire et l'appel à l'API.

FIN

