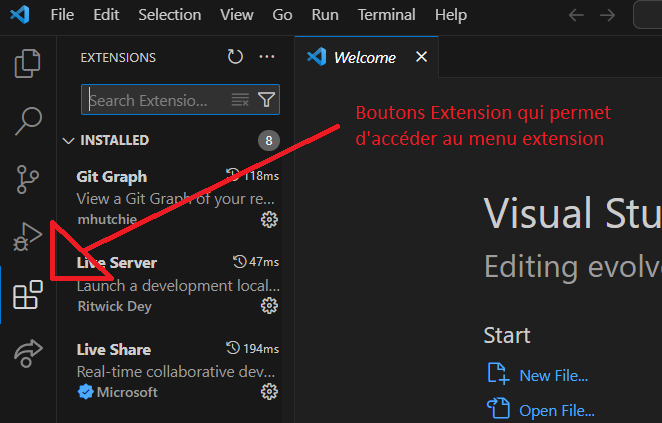
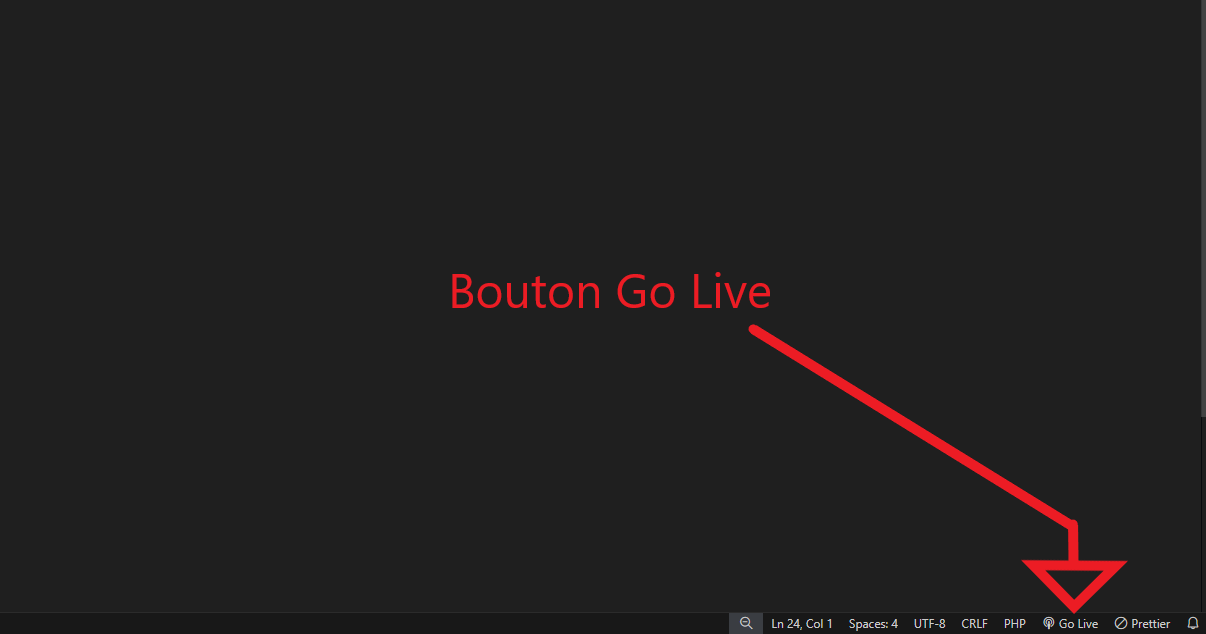
Contenu dossier professionnel

**Installation et Configuration de Mon Environnement de Travail**

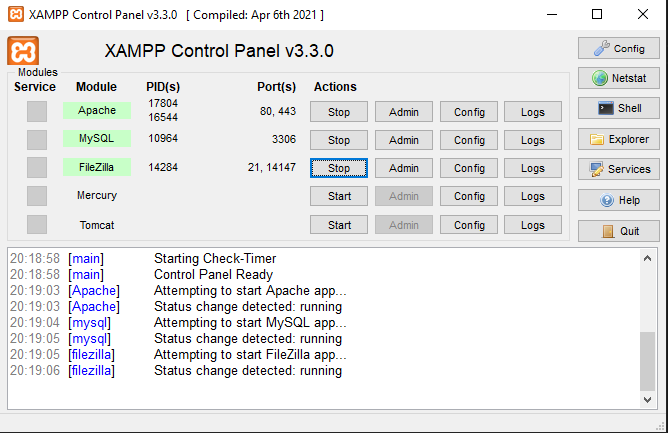
Pour mon éditeur de texte, j'ai choisi Visual Studio Code qui me permettra d’utiliser les langages (nécessaires à mes projets : HTML, CSS, JavaScript, PHP). J'ai téléchargé le programme d'installation depuis le site officiel et suivi les instructions pour l'installer sur mon système Windows.

Pour faciliter le développement web, j'ai ajouté l'extension "Live Server" à Visual Studio Code. Cette extension, disponible dans la Marketplace, me permet de lancer un serveur de développement en un clic grâce au bouton "Go Live" en bas à droite de l'éditeur. 

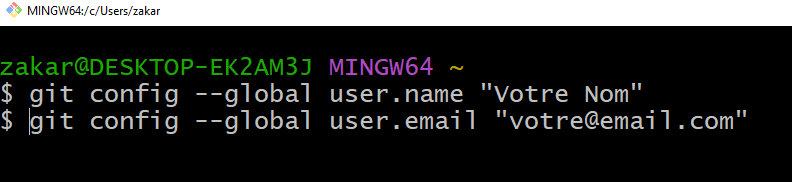


Pour le serveur web local et la gestion de base de données, j'ai opté pour XAMPP. J'ai téléchargé la dernière version depuis le site officiel, puis j'ai suivi les étapes d'installation. J'ai choisi d'installer Apache, MySQL, et PHP, et j'ai démarré les services correspondants depuis le panneau de contrôle XAMPP.

Comme XAMPP inclut MySQL, je me suis assuré que le service MySQL était démarré depuis le panneau de contrôle XAMPP.



Pour la gestion de versions, j'ai installé Git en utilisant le programme d'installation disponible sur [Git for Windows](https://gitforwindows.org/). Après l'installation, j'ai configuré mon nom et mon adresse e-mail via Git Bash en utilisant les commandes :



Pour la gestion de projet et la collaboration, j'ai choisi Trello. Je me suis inscrit sur [Trello](https://trello.com/) et créé un tableau dédié à mon projet. J'utilise Trello pour organiser mes tâches sous forme de listes et de cartes, facilitant ainsi la collaboration avec l'équipe.

Je veille à maintenir à jour régulièrement ces outils pour bénéficier des dernières fonctionnalités et corrections de sécurité.

**Installation et Configuration de Mon Environnement de Travail**

1. **Visual Studio Code :**

Pour mon éditeur de texte, j'ai choisi Visual Studio Code qui me permettra d’utiliser les langages (nécessaires à mes projets : HTML, CSS, JavaScript, PHP). J'ai téléchargé le programme d'installation depuis le site officiel et suivi les instructions pour l'installer sur mon système Windows.

2. **Live Server (Extension VS Code) :**

Pour faciliter le développement web, j'ai ajouté l'extension "Live Server" à Visual Studio Code. Cette extension, disponible dans la Marketplace, me permet de lancer un serveur de développement en un clic grâce au bouton "Go Live" en bas à droite de l'éditeur.

2. **XAMPP (Apache, MySQL, PHP) :**

Pour le serveur web local et la gestion de base de données, j'ai opté pour XAMPP. J'ai téléchargé la dernière version depuis le site officiel, puis j'ai suivi les étapes d'installation. J'ai choisi d'installer Apache, MySQL, et PHP, et j'ai démarré les services correspondants depuis le panneau de contrôle XAMPP.

3. **Git :**

Pour la gestion de versions, j'ai installé Git en utilisant le programme d'installation disponible sur [Git for Windows](https://gitforwindows.org/). Après l'installation, j'ai configuré mon nom et mon adresse e-mail via Git Bash en utilisant les commandes :

4. **MySQL :**

Comme XAMPP inclut MySQL, je me suis assuré que le service MySQL était démarré depuis le panneau de contrôle XAMPP.

6. **Trello :**

Pour la gestion de projet et la collaboration, j'ai choisi Trello. Je me suis inscrit sur [Trello](https://trello.com/) et créé un tableau dédié à mon projet. J'utilise Trello pour organiser mes tâches sous forme de listes et de cartes, facilitant ainsi la collaboration avec l'équipe.

**Maquetter des Interfaces Utilisateur avec Figma**

Étape 1 : Analyse des Besoins

Dans un premier temps, je m'attache à comprendre les besoins du projet en collaborant étroitement avec les parties prenantes. Cette phase implique une analyse approfondie des fonctionnalités requises et des objectifs spécifiques de l'interface utilisateur.

Étape 2 : Recherche et Inspiration

Je m'engage dans une recherche approfondie pour recueillir des inspirations et rester à jour avec les tendances actuelles en matière de conception. J'explore divers sites web, applications mobiles et autres sources pour identifier des éléments de design pertinents.

Étape 3 : Création d'un Wireframe Initial avec Figma

En utilisant l'outil de conception Figma, je conçois un wireframe initial. Mon objectif est de me concentrer sur la disposition des éléments principaux sans entrer dans les détails visuels. Figma facilite cette étape en permettant une création rapide et collaborative.

Étape 4 : Structuration de l'Information

J'organise l'information de manière logique en utilisant des groupes, des sections et des menus si nécessaire. Je veille à ce que la navigation soit intuitive en respectant les meilleures pratiques en UX (expérience utilisateur).

Étape 5 : Hiérarchisation des Éléments

Je définis une hiérarchie visuelle pour les éléments en fonction de leur importance. J'utilise les fonctionnalités de mise en forme de Figma pour guider l'attention de l'utilisateur à l'aide de différences de taille, de couleur et de style.

Étape 8 : Itération et Réajustement dans Figma

Grâce aux fonctionnalités de Figma, je peux itérer et réajuster ma maquette en fonction des retours des tests utilisateurs ou des discussions avec les parties prenantes. La collaboration en temps réel facilite le travail d'équipe.

**Remarques Supplémentaires**

* J'explore les fonctionnalités de collaboration en temps réel de Figma pour travailler efficacement avec les membres de l'équipe.
* Je m'assure que la maquette dans Figma est accessible à toutes les parties prenantes pour recueillir des commentaires supplémentaires.

En suivant ces étapes avec Figma, je parviens à créer une maquette d'interface utilisateur web ou web mobile bien pensée, alignée sur les besoins du projet et offrant une expérience utilisateur optimale.

**Maquetter le Site Web de Yann Couvreur avec Figma**

Je commence par une analyse approfondie des besoins du projet pour comprendre les exigences spécifiques du site web du pâtissier Yann Couvreur. Cette étape est cruciale pour cerner les fonctionnalités clés et les objectifs de l'interface utilisateur.

Habituellement, lors d’un maquettage je mène une recherche pour collecter des inspirations. Je parcours des sites web concurrents, des applications mobiles de référence, et des sources de design actuelles pour trouver des éléments pertinents à intégrer à ma maquette. Dans le cadre du maquettage du site de Yann Couvreur, je pars de son propre site web ce qui me permet de me concentrer sur la précision pour une mise en forme réussie et fidèle au projet

J'utilise Figma pour esquisser le wireframe initial du site web de Yann Couvreur. En me concentrant sur la disposition des éléments principaux, je construis une structure de base sans entrer dans les détails visuels. Figma se révèle être un outil idéal pour cette phase, permettant une création rapide et une collaboration efficace.

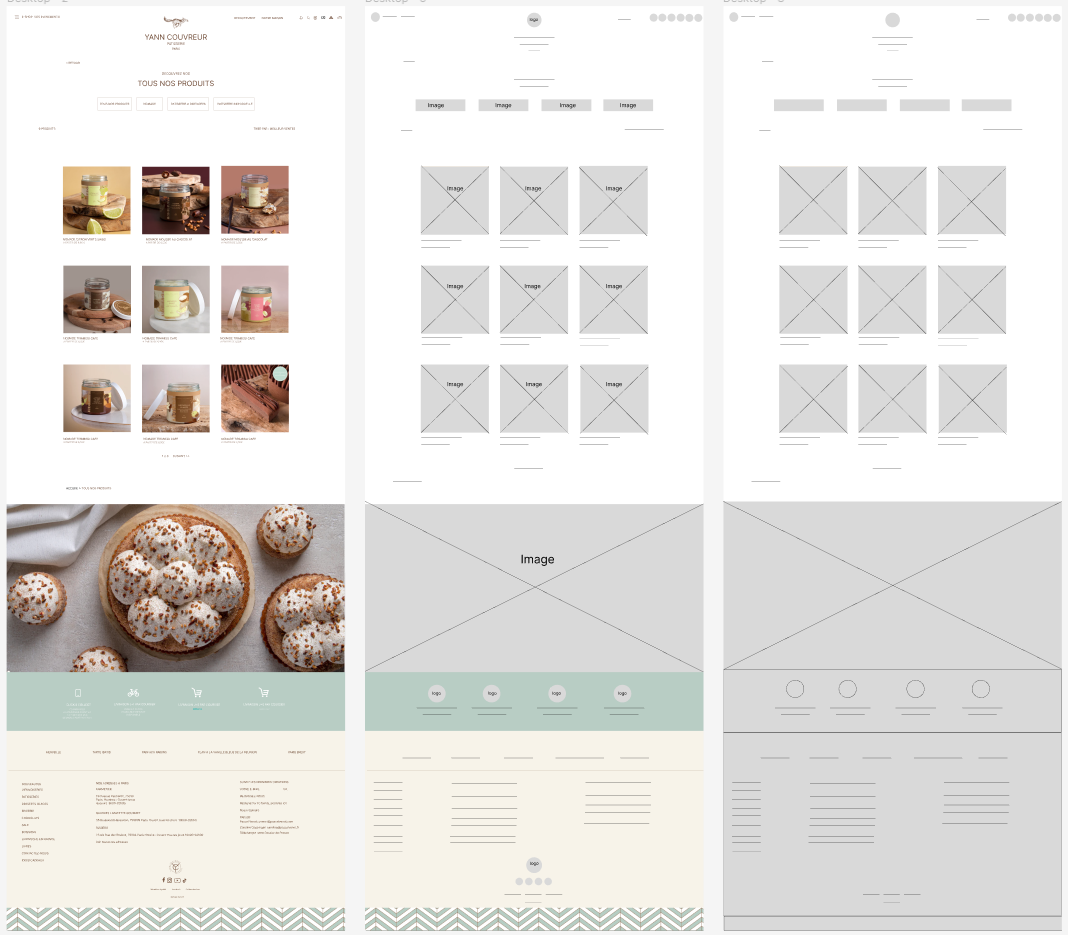
Avec le wireframe en place, je travaille sur la structuration de l'information. En utilisant les fonctionnalités de regroupement de Figma, je m'assure que l'information est organisée de manière logique, avec une navigation intuitive conforme aux meilleures pratiques en UX.

La hiérarchisation visuelle des éléments est cruciale pour guider l'attention de l'utilisateur. Dans Figma, j'utilise des différences de taille, de couleur et de style pour créer une hiérarchie visuelle claire et attirante.

Je profite des fonctionnalités de collaboration en temps réel de Figma pour itérer et réajuster ma maquette en fonction des retours des tests utilisateurs. Cette itération constante garantit que l'interface répond parfaitement aux besoins spécifiques du projet.

**Remarques Supplémentaires**

* Les fonctionnalités de collaboration en temps réel de Figma sont particulièrement utiles pour travailler efficacement avec les membres de l'équipe.
* Assurer que ma maquette dans Figma est accessible à toutes les parties prenantes facilite la collecte de commentaires supplémentaires et le processus de validation.



**Réalisation d'Interfaces Utilisateur Statiques Web ou Web Mobile avec HTML5 et CSS3 (Responsive, Mobile-First)**

1. **Analyse des Besoins et Utilisation de la Maquette Figma :**

* Actuellement, je travaille sur la reproduction du site de l'association Recycl'Optics,. Habituellement, je m'appuie sur la maquette finale créée dans Figma, mais cette fois-ci qui me sert de référence visuelle pour la conception des interfaces est le site de Recycl’Optics

2. **Choix des Technologies Front-End :**

* Pour la réalisation de l'interface statique, j'utilise les langages HTML5 et CSS3. HTML5 est employé pour structurer le contenu des pages, tandis que CSS3 me permet d'appliquer les styles nécessaires.

3. **Intégration de la Maquette avec HTML5 :**

* En me basant sur la maquette, j'utilise les balises HTML5 pour structurer le contenu du site du pâtissier Yann Couvreur. Les balises sémantiques comme **<header>**, **<main>**, **<section>**, et **<footer>** sont utilisées pour une organisation claire et compréhensible.

4. **Stylisation Fidèle avec CSS3 :**

* Le fichier CSS contient des règles spécifiques pour reproduire fidèlement les styles de la maquette. J'utilise des sélecteurs comme **.header**, **.accueil\_section1**, **.accueil\_section2**, etc., et j'applique des propriétés CSS telles que **background-color**, **box-shadow**, **border-radius** pour obtenir le rendu visuel attendu.

5. **Gestion des Médias et Responsive Design :**

* La maquette est adaptée pour garantir un design responsive en utilisant des règles de media queries. Le principe "mobile-first" est appliqué, assurant une expérience utilisateur optimale sur les petites résolutions d'écrans.

6. **Ajout de Micro-Interactions (si nécessaire) :**

* Si la maquette comprend des éléments interactifs, ces derniers sont implémentés en utilisant les fonctionnalités avancées de CSS3. Les boutons, les liens, et les images peuvent réagir de manière fluide au survol ou à d'autres interactions utilisateur.

7. **Tests et Débogage :**

* Les interfaces sont régulièrement testées sur différents navigateurs et appareils pour assurer une présentation uniforme. Les outils de débogage des navigateurs sont utilisés pour identifier et résoudre tout problème d'affichage ou de style.

**Remarques Supplémentaires :**

* Un style global est défini avec **\*** pour normaliser l'apparence à travers toutes les balises HTML.
* Des règles spécifiques sont ajoutées pour les balises **<a>**, **<h1>**, **<h2>**, **<h3>**, et **<i>** afin de maintenir une cohérence visuelle.
* Les couleurs, les ombres, et les bordures sont ajustées pour créer une esthétique moderne et professionnelle.
* La section **@media** sert à garantir une expérience utilisateur optimale sur les écrans de petite taille, en particulier avec une approche "mobile-first".

### Développement Dynamique avec JavaScript dans mon Projet

#### 3. ****Exercice 1 : Permutation d'Inputs****

Pour renforcer ma maîtrise de JavaScript et son application dans le développement dynamique, j'ai réalisés deux exercices pratiques. Le premier consistant en la permutation de valeurs entre deux inputs dans un formulaire. Voici comment j'ai procédé :

Dans cette démonstration, j'ai utilisé JavaScript pour détecter le clic sur le bouton "Permutter" et échanger les valeurs entre les deux champs de texte. Cela a renforcé ma compréhension des événements JavaScript et de la manipulation du DOM.

Image des inputs

Dans le code HTML, j'ai inclus deux champs de saisie de type "text" avec l'attribut "name" défini comme "input". Pour déclencher une action au clic sur l'un de ces champs, j'ai ajouté l'attribut "onclick" en lui associant la fonction "permut()".

Image JS de la recuperation de .input et de la fonction permut()

Nous débutons en déclarant la variable "input", qui forme un tableau en regroupant les deux champs de saisie grâce à leur attribut "name", grâce à la méthode "getElementsByName()".

Ensuite, la fonction "permut()" est créée. À l'intérieur de celle-ci, je récupère les valeurs de chaque champ du tableau "input" et les associe à deux variables distinctes, "input1" et "input2". Une variable temporaire, "input1Bis", est utilisée pour stocker la valeur de "input1". Enfin, je réaffecte la valeur de "input2" à "input1" et la valeur temporaire "input1Bis" à "input2". Ainsi, lorsqu'on appuie sur le bouton "Permuter", les valeurs des deux champs d'entrée sont échangées l'une avec l'autre. Voici le résultat visuel avant cette permutation.

Image avant et après permutation.

#### 4. ****Exercice 2 : Compteur de Caractères****

Le deuxième exercice pratique que j'ai réalisé concerne la création d'un compteur de caractères en temps réel dans une zone de texte d'un formulaire. Voici comment j'ai implémenté cette fonctionnalité :

Image de html

Ici, au sein du code HTML et toujours à l'intérieur d'un formulaire, nous retrouvons un champ de saisie de type texte. J'y ai ajouté l'attribut "onkeypress", qui déclenche l'exécution de la fonction "textcontrol()" à chaque pression de touche à l'intérieur du champ, en réagissant aux événements du clavier.

Notons également la présence d'un champ de saisie de type texte avec l'identifiant "caracRemain", qui jouera un rôle crucial dans la partie JavaScript du code.

Image du JS

Dans cette section, nous récupérons les identifiants "zoneTexte" et "caracRemain", les associant à deux variables homonymes à l'aide de la méthode "getElementById". Ensuite, la fonction "textcontrol()" est créée.

La valeur de 30 est assignée à une constante nommée "maxLength", définissant ainsi la longueur maximale de notre chaîne de caractères. Une autre variable, "remainingCharacters", est instanciée et équivaut à la différence entre "maxLength" (soit 30) et la longueur actuelle de la valeur de "zoneTexte". Cette dernière représente la longueur totale des caractères dans l'input où l'utilisateur est en train de saisir. Cela permet d'effectuer un décompte à partir de 30.

Enfin, une condition "if" vérifie si le nombre de caractères restants ("remainingCharacters") est inférieur ou égal à 0. Dans ce cas, un message est affiché indiquant que l'utilisateur a atteint le nombre maximal de caractères. De plus, un attribut "readonly" est ajouté à l'élément ayant l'identifiant "zoneTexte" et est configuré avec la valeur "true", empêchant ainsi l'ajout de caractères supplémentaires lorsque la limite de 30 est atteinte.

caracRemain.value = remainingCharacters;

La ligne "caracRemain.value = remainingCharacters;" assure l'affichage en temps réel du nombre de caractères restants, en assignant la valeur de "remainingCharacters" à l'attribut "value" de l'input identifié par "caracRemain". Cela affiche dynamiquement la valeur dans notre champ d'entrée.

Voici le résultat obtenu.

Image du rendu

En utilisant JavaScript, j'ai créé une fonction **textControl()** qui réagit à chaque frappe de touche dans la zone de texte. Elle compte le nombre de caractères restants et l'affiche dans un autre champ. Si la limite est atteinte, un message d'alerte est affiché, et la zone de texte devient en lecture seule.

### Développement de composant métier coté serveur PHP

1. Je vais à présent aborder les progrès que j'ai réalisés dans ma formation concernant le développement des composants métier côté serveur, en illustrant cela à travers l'exemple concret d'un formulaire de connexion.Dans cette démarche, nous partons du principe que la base de données a été préalablement alimentée via le processus d'inscription, incluant des informations telles qu'une adresse e-mail et un mot de passe. Le formulaire de connexion est structuré de la manière suivante :

Image html du formulaire

Nous rencontrons ici un formulaire qui redirige, via son attribut "action", vers la page traitement\_co.php, responsable de la gestion de cette connexion. Il utilise la méthode POST, une approche sécurisée pour l'envoi de données dans un formulaire. Passons désormais à l'analyse détaillée du fichier traitement\_co.php.

Image d $mail et $ password

Le processus débute par une vérification à l'aide d'une condition ternaire pour s'assurer que les variables $mail et $password sont définies et non vides. Si ces conditions ne sont pas remplies, elles sont alors assignées à la valeur NULL.

1. **Récupération des données du formulaire :**
   * On récupère les données de l'utilisateur (email et mot de passe) à partir du formulaire et on s'assure qu'ils sont définis et non vides.
2. **Connexion à la base de données :**
   * On crée un objet PDO et on le configure pour gérer les erreurs.
   * Les informations de connexion (hôte, nom de la base de données, nom d'utilisateur, mot de passe) sont utilisées pour établir la connexion.
3. **Requête SQL et Exécution :**
   * On prépare une requête SQL sécurisée qui sélectionne les utilisateurs actifs avec l'email fourni.
   * On exécute la requête en remplaçant les paramètres par les valeurs fournies par l'utilisateur.
   * On récupère les résultats de la requête.
4. **Vérification du Mot de Passe :**
   * On utilise **password\_verify** pour comparer le mot de passe fourni par l'utilisateur avec celui stocké dans la base de données (préalablement crypté avec **password\_hash** lors de l'inscription).
   * Si la vérification est réussie, l'utilisateur est redirigé vers la page **user\_home.php**, sinon un message d'erreur est affiché.

Ce processus illustre la manière dont les composants métier côté serveur sont implémentés pour gérer la connexion d'un utilisateur et vérifier la validité de ses informations d'identification.

Image if ($mail && $password){

Ensuite, le processus entre dans une condition qui débute lorsque les variables $mail et $password sont définies.

Image du try

À ce stade, les constantes de connexion sont intégrées, et un objet PDO est créé pour encapsuler toutes les informations liées à la connexion à la base de données, attribué à la variable $cnn. De plus, des attributs sont ajoutés à cette variable $cnn pour servir de gestionnaire d'erreur PDO en cas de difficultés de connexion avec la base de données.

Ensuite, la requête SQL à exécuter est préparée.

Image de la requete SQL

Dans cette étape, la requête SQL est attribuée à la variable $qry. Initialement, une tentative de connexion est lancée via l'objet $cnn, auquel est associée la méthode prepare. Cette méthode sécurise la requête contre d'éventuelles injections malveillantes. En tant qu'argument de cette méthode, la requête en langage SQL est spécifiée.

La requête SQL en question sélectionne tous les éléments de la table "users" avec un filtre correspondant aux adresses e-mail représentées par des points d'interrogation (correspondant à l'input utilisateur), et la condition "active = ?" qui est une valeur dans la base de données indiquant si l'adresse e-mail est considérée comme active sur le site.

La requête est ensuite exécutée en remplaçant les points d'interrogation par les arguments de notre tableau. Enfin, une variable $user est déclarée pour récupérer les résultats de la requête à l'aide de la méthode fetch(). Ensuite, le mot de passe de l'utilisateur est vérifié.

Haut du formulaire

Image de password\_verify

La méthode password\_verify() sert à confirmer la correspondance entre deux arguments : le mot de passe non crypté inséré par l'utilisateur dans le formulaire, et le mot de passe crypté stocké dans la base de données grâce à la méthode password\_hash().

Si la condition est satisfaite, l'utilisateur est redirigé vers la page user\_home.php, qui représente son espace de profil. En cas d'échec, un message "mot de passe incorrect" est affiché.

### Développer des composants d’accès aux données SQL et NoSQL

Je vais vous décrire brièvement la manière de développer des composants d'accès aux données SQL et NoSQL. Lorsque vous créez un projet, la connexion à différentes bases de données est souvent nécessaire pour développer les fonctionnalités d'un site web. En PHP, vous pouvez établir une connexion de la manière suivante :

Image du PDO

On débute en définissant des variables de connexion. La première, "host", représente l'"adresse" de la base de données, généralement localhost lorsqu'elle est située sur notre propre ordinateur via un logiciel de gestion de serveur local. Ensuite, une variable spécifie le nom exact de la base de données, correspondant à celui dans notre gestionnaire MySQL/NoSQL. Les variables "user" et "password" sont utilisées pour définir nos identifiants et le mot de passe de l'utilisateur pour la base de données.

Ensuite, nous déclarons **$pdo** comme un nouvel objet PDO. Nous lui attribuons les attributs **mysql:host** et **dbname**, auxquels nous associons les valeurs de nos variables précédemment créées, ainsi que le jeu de caractères (charset), similaire à celui utilisé dans le HTML. Enfin, nous spécifions le nom d'utilisateur et le mot de passe.

Cet objet PDO facilite la connexion à notre base de données à chaque requête SQL que nous exécuterons. Pour optimiser ce processus, nous attribuons différents paramètres à notre objet PDO. Ces paramètres prédéfinis en PHP nous permettent de gérer les éventuelles erreurs de connexion lors de nos tentatives d'accès.

Haut du formulaire