Présentation d'un site e-commerce thème WordPress personnalisé

ABADIE DAMIEN

TITRE PROFESSIONNEL DÉVELOPPEUR WEB ET WEB MOBILE 2025

GRETA CFA SUD CHAMPAGNE TROYES

Sommaire

- DESIGN DU SITE
- DÉVELOPPEMENT
  - Front-end
  - Back-end
- DÉPLOIEMENT
- CRUD Bougies

DESIGN DU SITE

CAHIER DES CHARGES, CROQUIS ET MAQUETTE

Cahier des charges

- Dans un premier temps, j'ai rédigé un **document de collecte d'informations** afin de réunir les éléments nécessaires à la conception du **cahier des charges** et de la **charte graphique** du projet.

- Mon tuteur m'avait fourni un **nuancier de couleurs** au début du projet. Cependant, vous constaterez que celui-ci a **évolué dans la version finale** du site.

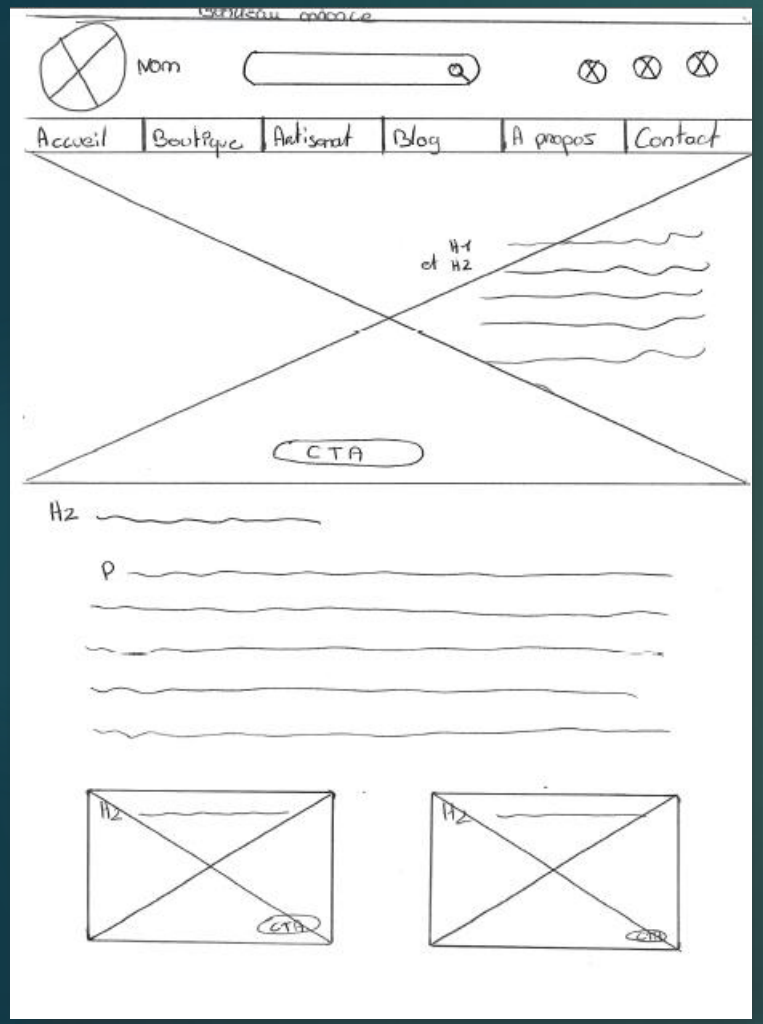
- J'ai élaboré le **cahier des charges** du mieux possible à partir des **informations disponibles** et des **attentes exprimées par le client**. J'y ai intégré les éléments essentiels, tels que la **charte graphique**, ainsi que les **besoins fonctionnels et esthétiques** relatifs à la conception de son site web.

Collecte d'informations

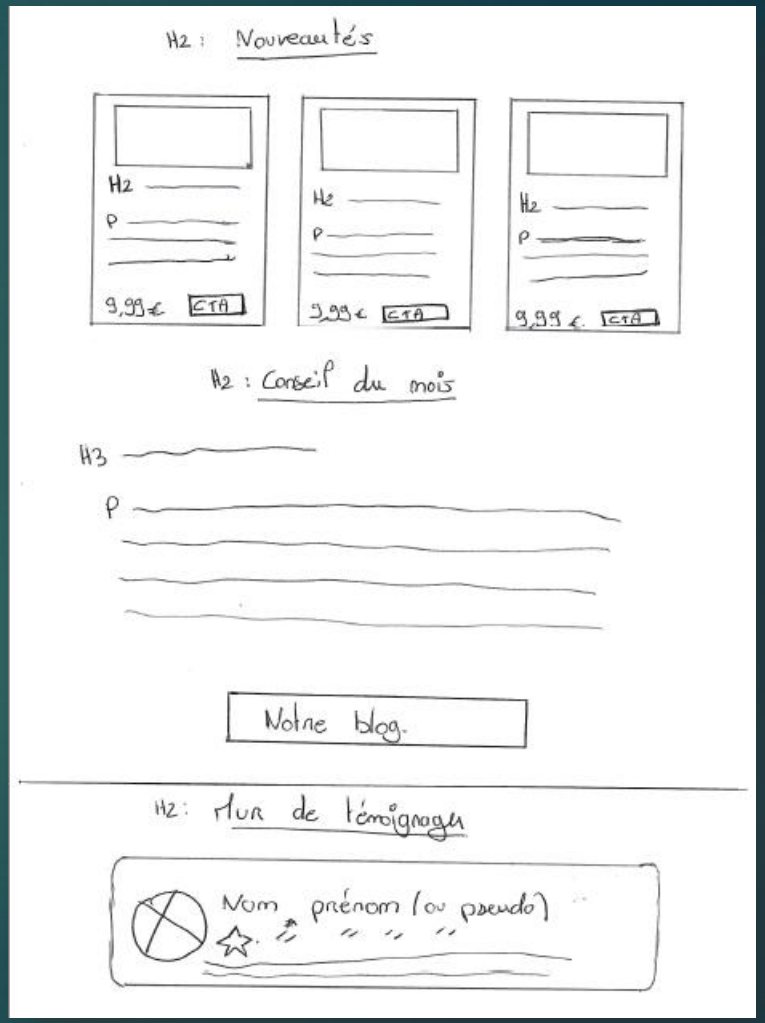
Nuancier de couleur

Cahier des charges

Croquis (wireframe)

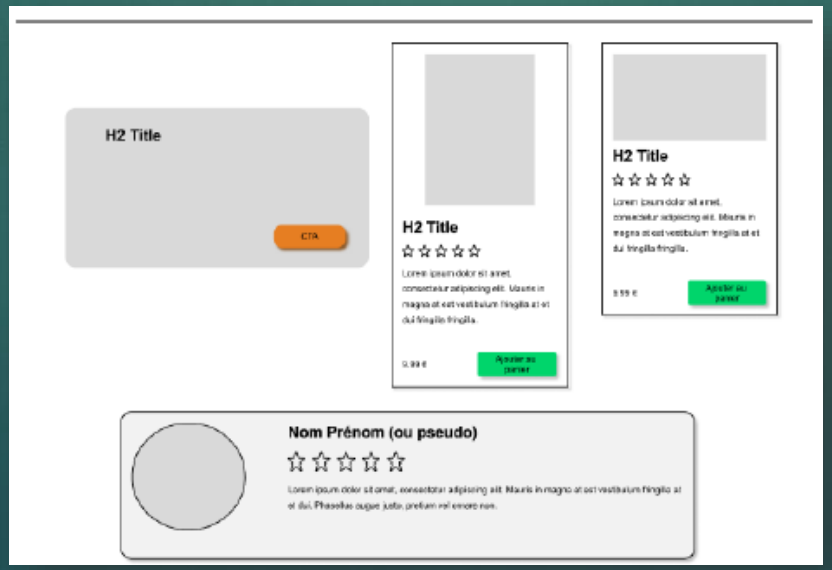


- J'ai commencé par **réaliser un croquis à la main** afin de matérialiser l'idée que j'avais pour le projet. J'ai conçu la **page d'accueil**, qui représentait la partie la plus complexe et la plus exigeante du travail.

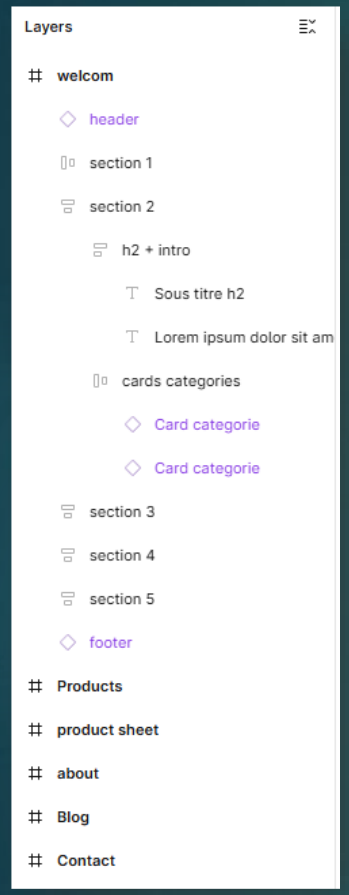


Maquette (design system)

- Avant de débuter la conception de la maquette, j'ai créé un **design system** afin de **faciliter la construction visuelle** du projet et de **gagner du temps** lors de la mise en page.
- Ce document regroupe l'ensemble des **éléments graphiques réutilisables**, tels que : (La typographie, le nuancier de couleur, le logo, des icônes, des boutons, la navigation et les cartes).
- Également des éléments à importer : certains **fonds de page ou de section**, notamment des **textures bois** 100 % réalistes, conformément à la demande du client, qui souhaitait rappeler la **matière première** qu'il travaille.

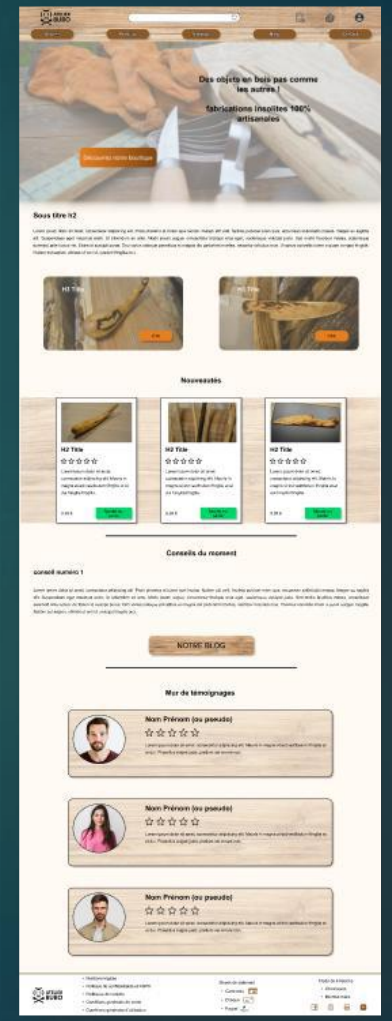


Maquette (desktop)



- Une fois le **design system** finalisé, j'ai pu entamer la **conception de la maquette desktop**, en procédant **page par page**. J'y ai intégré l'ensemble des **composants principaux**, tels que le **header**, le **footer**, les **cartes**, les **boutons** ainsi que les **icônes**, afin d'assurer une **cohérence visuelle** et une **uniformité d'expérience utilisateur** sur l'ensemble du site.

- J'ai veillé à **illustrer au maximum** les pages avec des **images pertinentes**, afin de permettre au client de **se projeter plus facilement** dans le rendu final de la maquette



Maquette (mobile)

- Après avoir terminé la **maquette Desktop**, je me suis ensuite consacré à la **maquette mobile**. Grâce à l'utilisation de la fonctionnalité **Auto Layout** sur **Figma**, j'ai pu **adapter facilement la disposition des éléments**.

- Cette étape m'a également permis de mieux comprendre les **principes de conception responsive**, essentiels pour garantir une **expérience utilisateur optimale** sur mobile.

Développement

FRONT-END

HTML (HyperText Markup Language)

- En tant que **débutant dans le développement front-end**, j'ai rencontré **plusieurs difficultés techniques** lors de la **reproduction exacte de la maquette**.

- Malgré ces obstacles, cette phase m'a permis d'**approfondir mes connaissances en intégration web**.

Section hero de la page d'accueil

```
<section class="hero">
  <div class="button">
    <button>Découvrez notre boutique</button>
  </div>
  <div class="title">
    <h1>
      Gravure sur bois artisanale
      Objets personnalisés et cadeaux
originaux en bois
    </h1>
  </div>
</section>
```

Statique VS dynamique

- Statique – cela illustre :

  - une zone **codée en dur**,
  - affichée **telle quelle**;
  - **non liée à la base de données**,
  - ne nécessitant aucune logique PHP ou WordPress.

```html
<section class="nouveautes">
  <div class="title">
    <div class="separateur"></div>
    <h2>Nouveautés</h2>
    <div class="separateur"></div>
  </div>
</section>
```

```php
$args = array(
  'post_type'      => 'product',
  'posts_per_page' => 3,
  'orderby'        => 'post_date',
  'order'          => 'DESC',
);

$loop = new WP_Query($args);

if ($loop->have_posts()) :
  while ($loop->have_posts()) : $loop->the_post();
```

- WordPress exécute une requête dynamique en allant chercher :

  - les **produits WooCommerce**,
  - **classés par date**,
  - **limités aux 3 plus récents**,
  - récupérés via la classe **WP_Query**.

CSS (Cascading Style Sheets)


```css
.hero {
  background: linear-gradient(
      rgba(255, 255, 255, 0.2),
      rgba(255, 255, 255, 0.9)
    ),
    url("../img/background-hero.jpg")
center/cover no-repeat;
  overflow: hidden;
  position: relative;
  height: 85vh;
  text-align: center;
  padding: 2rem;
}
```

- J'ai réalisé plusieurs fichiers **CSS**, un pour chaque page **HTML**.
- Le code suivant correspond à la **section "hero"** du site. Cette section permet de capter immédiatement l'attention de l'utilisateur grâce à une **image de fond**.

- Ce code correspond aux deux **cartes** de catégories situées après la section d'introduction sur la page d'accueil.
- Il définit principalement la mise en page et le **positionnement** des blocs au sein du site.

- **Bon à savoir :** La propriété position : **relative** déplace un élément par rapport à sa position d'origine, alors que position : **absolue** le positionne par rapport à son parent.

```css
.presentation .cards .card_1,
.presentation .cards .card_2 {
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  position: relative;
  width: 40%;
  height: 40vh;
  padding: var(--spacing-md);
  border-radius: var(--radius-20);
  background-color: var(--color-white);
  box-shadow: var(--shadow);
}
```

JS (JavaScript)

- **Document :** Représente **l'ensemble** de la page web au sein du **DOM**.
- **Écouteur d'événement :** Permet de **surveiller** un événement particulier
- **Fonction fléchée :** C'est une **synthaxe** moderne introduite avec la recommandation **ES6**, elle définit le bloc d'instructions à exécuter lorsque l'événement est déclenché.

```
document.addEventListener("DOMContentLoaded", () => {
  initHeaderEvents();
```

- **Initialisation des événements du header :** Appel d'une fonction qui s'exécute dès que la page est prête.

- **Fonction de rappel :** Interactions dans l'en-tête du site, ici pour le mobile.

- **Const :** Constantes utilisées pour sélectionner des éléments HTML à partir de leurs classes.

- **If (burger && nav) :** S'assure que les deux éléments existent bien dans le DOM avant d'exécuter le reste du code.
- **Écouteur d'événement « click » :** Permet d'exécuter une fonction au click.
- **Activation du menu burger :** Ajoute ou retire la classe CSS « active ».
- **Ouverture et fermeture du menu :** Ajoute ou retire la classe « open » sur le menu de navigation

```
function initHeaderEvents() {
  const burger = document.querySelector(".burger");
  const nav    = document.querySelector("nav");


  if (burger && nav) {
    burger.addEventListener("click", () => {
      burger.classList.toggle("active");
      nav.classList.toggle("open");
  }
    );
  }
}
```

- Cette fonction à pour objectif de gérer la galerie d'images présente sur la fiche produit.

- Ce if est une condition de sécurité afin d'interrompre l'exécution sur l'un des deux éléments essentiels est manquant.

```
function initProductGallery() {
    const galerie   = document.querySelector(".images_secondaires .track");
    const mainImage = document.querySelector(".image_principale img");
    if (!galerie || !mainImage) return;
```

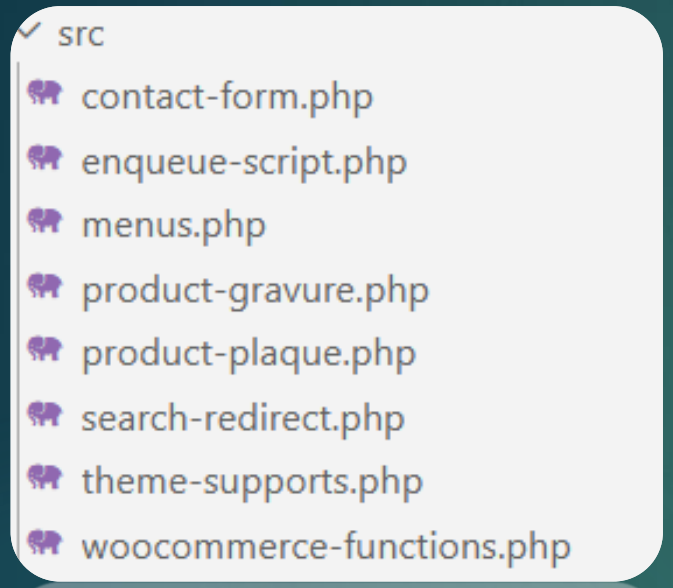
```
galerie.querySelectorAll(".thumb").forEach((thumb) => {
      const thumbImg = thumb.querySelector("img");
      thumb.addEventListener("click", () => {
        if (!thumbImg) return;
```

- On sélectionne tous les éléments possédant la classe « thumb » à l'intérieur de la galerie.
- La const « thumbImg » récupère l'image contenue dans chaque éléments « thumb »
- AddEvenListener analyse chaque clic.

Développement

BACK-END

Functions



- Pour améliorer la lisibilité et la maintenance du thème, j'ai divisé le fichier « function.php » en plusieurs fichiers spécialisés
- Quand a lui le fichier function.php est la pour chargé tout les fichiers divisés.
- Le fichier enqueue-script.php est un fichier clé au projet.
- Concernant le fichier theme-supports lui s'occupe d'activer certaines fonctionnalités native de WordPress.

Personnalisation

Pour la gestion des produits personnalisables (gravures), j'ai mis en place un ensemble de champs dynamiques sur les fiches produits WooCommerce.

J'ai utilisé plusieurs hooks WordPress : woocommerce_before_add_to_cart_button, woocommerce_add_to_cart_validation, woocommerce_add_cart_item_data, woocommerce_before_calculate_totals, woocommerce_get_item_data et woocommerce_add_order_item_meta.

Barre de recherche

- Avec $is_product = ‘product’ === get_post_type(); on verifie que la requete concerne un produit WooCommerce ou non.
- Puis avec $product = $is_product ? wc_get_product(get_the_ID()) : null; on affiche le produit et ses informations, si celui-ci n’est pas un produit, la valeur est donc nulle.

- Optimisation des requêtes :
  Filtrer les résultats de recherche
- **add_action('pre_get_posts', function ($query) { ... })**
- pre_get_posts est un **hook WordPress** qui permet de modifier la requête **avant** qu’elle ne soit exécutée.
- La fonction anonyme reçoit $query, qui représente la **requête courante** (ce que WordPress va aller chercher en base).

```php
add_action('pre_get_posts', function($query) {
    if (!is_admin() && $query->is_main_query() && $query->is_search()) {
        $query->set('post_type', ['post', 'product']);
    }
});
```

```php
function redirect_exact_content_match() {
    if (is_search() && !is_admin() && isset($_GET['s'])) {
        $search_query = trim(sanitize_text_field($_GET['s']));

        $args_page = array(
            'post_type'      => 'page',
            'title'          => $search_query,
            'post_status'    => 'publish',
            'posts_per_page' => 1,
        );

        $page = get_posts($args_page);
        if ($page && count($page) === 1) {
            wp_redirect(get_permalink($page[0]->ID));
            exit;
        }

        $args_product = array(
            'post_type'      => 'product',
            'title'          => $search_query,
            'post_status'    => 'publish',
            'posts_per_page' => 1,
        );

        $product = get_posts($args_product);
        if ($product && count($product) === 1) {
            wp_redirect(get_permalink($product[0]->ID));
            exit;
        }
    }
}
```

- Redirection automatique :

  Pour optimiser davantage l'expérience utilisateur, j'ai ajouté une logique de redirection intelligente.

  Si l'utilisateur recherche un terme correspondant exactement au titre d'une page ou d'un produit, il est automatiquement envoyé vers ce contenu au lieu d'arriver sur une simple liste de résultats.

  Grâce à cette approche, j'ai pu réduire les clics inutiles et rendre la recherche beaucoup plus fluide, en particulier pour les produits dont le titre est typiquement saisi tel quel par les clients.

- Ajout d'une classe au <body>
  pour le CSS/JS

Wp_enqueue_script() permet de charger un fichier JavaScript, il sert à WordPress pour identifier et gérer ce JS.

- Cette fonction ajoute une classe CSS au <body> uniquement lorsque l'on est sur une page de rechercher WooCommerce

```
function mon_theme_scripts() {
  wp_enqueue_script(
    'theme-header-js',
    get_template_directory_uri() . '/assets/js/header.js',
    array(),
    null,
    true
  );
}
function add_search_body_class( $classes ) {
    if ( is_search() && is_woocommerce() ) {
        $classes[] = 'search-woocommerce-page';
    }
    return $classes;
}
```

Déploiement

CI/CD

CI/CD

Continuous integration / Continuous delivery

- Création seconde branche Git : Test
- Branche main : Déploiement automatique

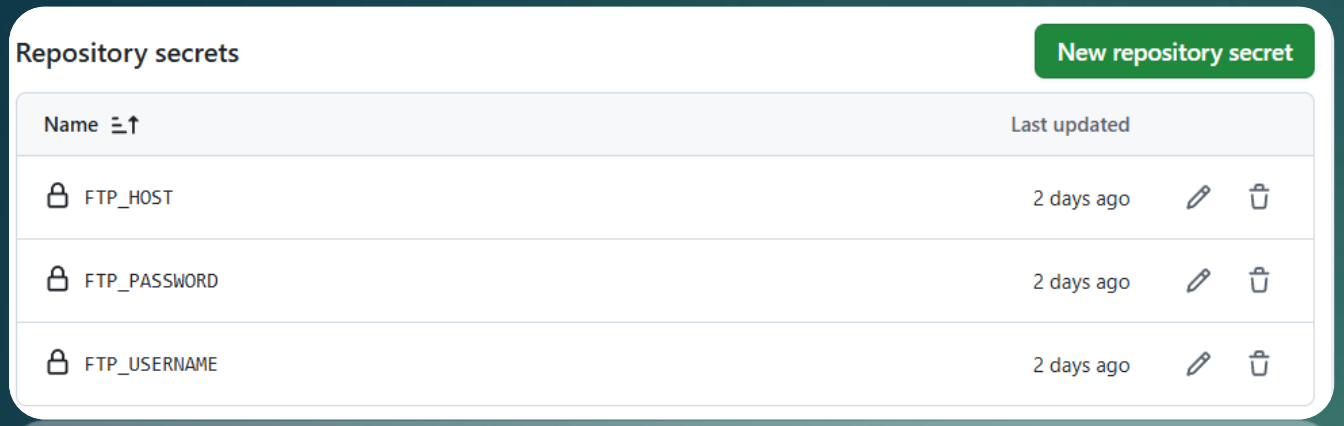
- Fichier de configuration YAML

  Dossier .github/workflows/ celui-ci permet la connexion FTP au serveur Cpanel, il permet d'automatiser chaque mise a jour de la branch main

- j'ai crée un **compte FTP dédié** via l'interface **cPanel**, afin de **limiter les accès** et de renforcer la **sécurité**, pour que **la navigation** en dehors du répertoire **soit refusé.**

- J'y ai ensuite installé **WordPress** et procédé à la **sauvegarde complète** du site à l'aide de la fonctionnalité **WP Tiger**, ce qui permet de **restaurer facilement** l'application en cas d'erreur ou de mauvaise manipulation.

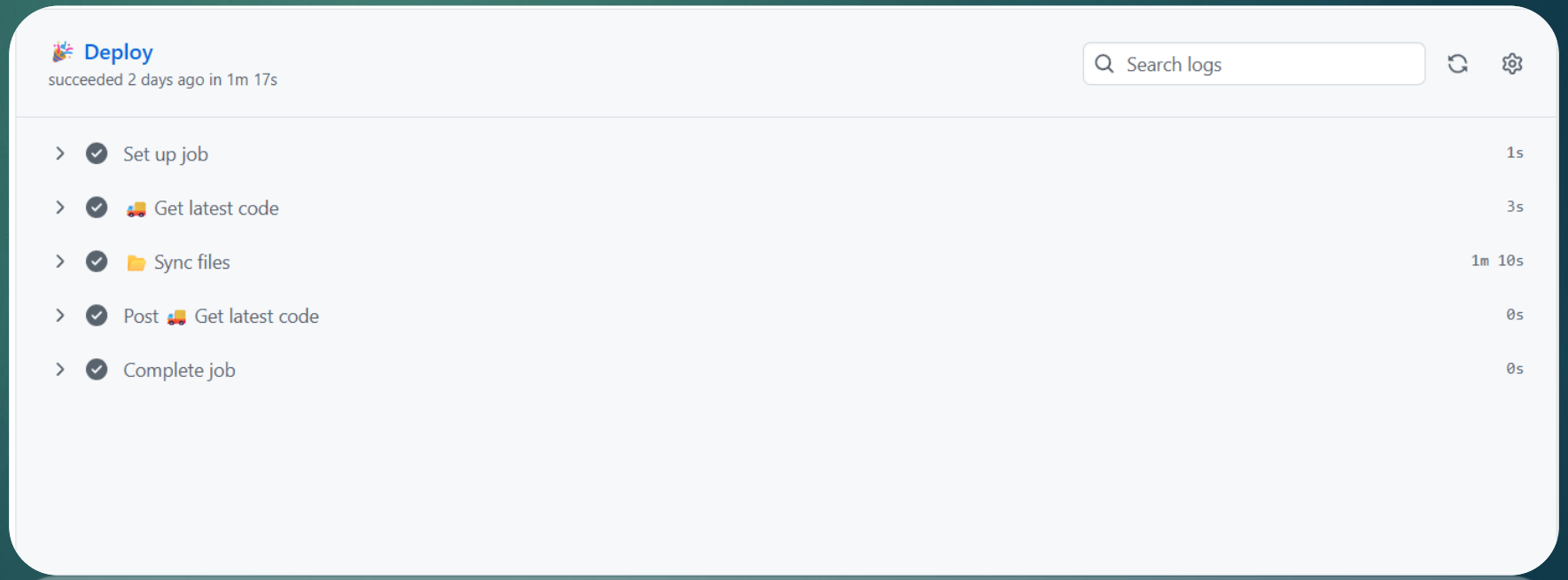
```yaml
on:
  push:
    branches:
      - main

name:  Deploy website on push

jobs:
  web-deploy:
    name:  Deploy
    runs-on: ubuntu-latest
    steps:
      - name:  Get latest code
        uses: actions/checkout@v4

      - name:  Sync files
        uses: SamKirkland/FTP-Deploy-Action@v4.3.5
        with:
          server:   ${{ secrets.FTP_HOST }}
          username: ${{ secrets.FTP_USERNAME }}
          password: ${{ secrets.FTP_PASSWORD }}
          protocol: ftp
          port: 21
          local-dir: ./
          server-dir: /wp-content/themes/mon-theme/
```



- Variables secrètes GitHub

  Dans les paramètres GitHub j'ai défini plusieurs variables secrètes, telles que FTP_HOST, FTP_USERNAME, FTP_PASSWORD

- Visualisation en temps réel du déploiement et vérification que le transfert s'effectue correctement.



Migration WordPress

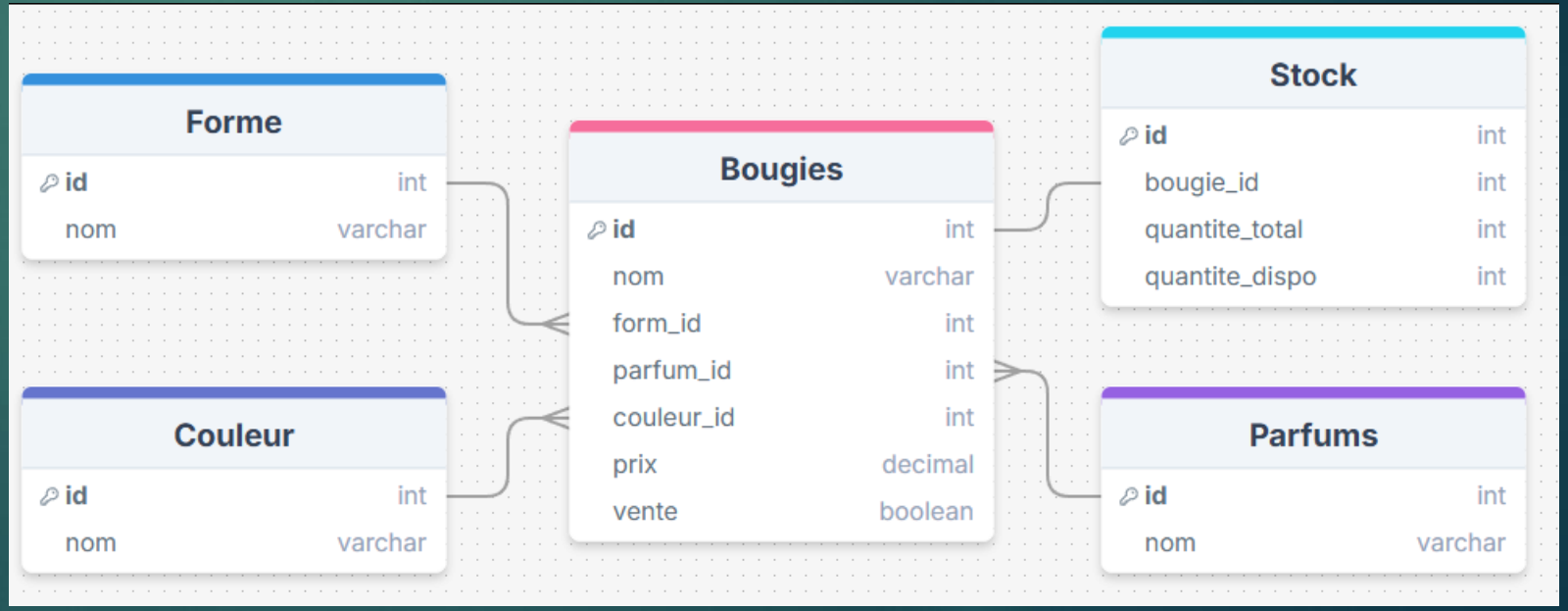
- Coté local :
- Plugin : UpdraftPlus : Sauvegarder maintenant
- Téléchargement des 5 fichiers à exporter

- Coté serveur :
- Plugin : UpdraftPlus : Téléverser des fichiers de sauvegarde
- Importer les 5 fichiers

Documentation (readme)

CRUD Bougies

CREATE, READ, UPDATE, DELETE



CRUD inventaire bougies

- Création de la base de donnée (gestion bougies) uniquement si elle n'existe pas.
- Création d'une table « bougies » à l'identifiant unique, référence de différentes caractéristiques (forme, parfum, couleur et prix), en vente oui ou non et assure que l'id existe bien dans sa table (foreign).
- Création d'une table « stock » quantité par défaut à 0 et une bougie est unique elle ne peut apparaitre plusieurs fois dans la table.

```sql
CREATE DATABASE IF NOT EXISTS gestion_bougies CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;
USE gestion_bougies;

CREATE TABLE Bougies (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(100) NOT NULL,
  forme_id INT,
  parfum_id INT,
  couleur_id INT,
  prix_id INT,
  vente BOOLEAN DEFAULT TRUE,
  FOREIGN KEY (forme_id) REFERENCES Forme(id),
  FOREIGN KEY (parfum_id) REFERENCES Parfums(id),
  FOREIGN KEY (couleur_id) REFERENCES Couleur(id),
  FOREIGN KEY (prix_id) REFERENCES Prix(id)
);

CREATE TABLE Stock (
  id INT AUTO_INCREMENT PRIMARY KEY,
  bougie_id INT NOT NULL,
  quantite_total INT DEFAULT 0,
  quantite_dispo INT DEFAULT 0,
  FOREIGN KEY (bougie_id) REFERENCES Bougies(id),
  UNIQUE (bougie_id)
);
```

Connexion à la base de donnée


```php
<?php
$host = 'localhost';
$db   = 'gestion_bougies';
$user = 'root';
$pass = '';

try {
  $pdo = new PDO("mysql:host=$host;dbname=$db;charset=utf8mb4", $user, $pass, [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
  ]);
} catch (PDOException $e) {
  die("Erreur de connexion : " . $e->getMessage());
}
?>
```

- Connexion en local
- Connexion grâce à PDO (PHP Data Object) en passant par new PDO()
- Catch : Erreur liées à PDO (connexion ou requêtes), si une erreur se produit le script casse directement
- Die : Le script casse et affiche un message d'erreur

Formulaire d'ajout


```php
$formes = $pdo->query("SELECT * FROM Forme ORDER BY nom")->fetchAll();
  foreach ($formes as $f) {
   echo "<option value='{$f['id']}'>" . htmlspecialchars($f['nom']) . "</option>";
   }
```

- PDO : connexion à la base de donnée
- Requête SQL pour récupérer toutes les colonnes de la table Forme triée par ordre alphabétique
- Analyse du tableau formes lignes par lignes
- Affiche le résultat

Tableau d'inventaire

- Récupère toutes les bougies
- Avec les jointures nous récupérons leurs formes, parfums, couleurs, prix et stocks

```php
$bougies = $pdo->query("
  SELECT b.id, b.nom, f.nom AS forme, p.nom AS parfum, c.nom AS couleur, pr.prix,
         s.quantite_total, s.quantite_dispo, b.vente
  FROM Bougies b
  LEFT JOIN Forme f ON b.forme_id = f.id
  LEFT JOIN Parfums p ON b.parfum_id = p.id
  LEFT JOIN Couleur c ON b.couleur_id = c.id
  LEFT JOIN Prix pr ON b.prix_id = pr.id
  LEFT JOIN Stock s ON b.id = s.bougie_id
  ORDER BY b.nom
")->fetchAll();
```

Supprimer

```php
<?php
// delete.php
include('config.php'); // Connexion PDO $pdo

session_start();

// --- CONFIG (par défaut false => desactiver) ---
$enable_csrf = false;          // true = vérifie token CSRF
$enable_auth = false;          // true = exige utilisateur loggué (voir commentaires)
$require_admin_role = false;   // true = exige $_SESSION['role'] === 'admin'
$check_exists_before_delete = true; // true = vérifie que la bougie existe avant
suppression
// ------------------------------------------------------------------------

if ($_SERVER['REQUEST_METHOD'] !== 'POST' || !isset($_POST['id'])) {
    header("Location: index.php");
    exit;
}

$id = (int) $_POST['id'];

// CSRF : si activé, vérifier le token
if ($enable_csrf) {
    if (!isset($_POST['csrf_token']) || !isset($_SESSION['csrf_token']) ||
$_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        // token invalide => redirection (comportement sûr)
        header("Location: index.php?message=forbidden");
        exit;
    }
    // optionnel: on peut supprimer le token après usage pour le rendre one-time
    unset($_SESSION['csrf_token']);
}

// Auth : si activé, s'assurer que l'utilisateur est connecté (adapter selon ton système)
if ($enable_auth) {
    if (!isset($_SESSION['user_id'])) {
        header("Location: index.php?message=login_required");
        exit;
    }
    if ($require_admin_role) {
        // adapte 'role' selon ton implémentation (admin, is_admin, etc.)
        if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
            header("Location: index.php?message=forbidden");
            exit;
        }
    }
}

try {
    // Optionnel : vérification d'existence avant suppression
    if ($check_exists_before_delete) {
        $stmtCheck = $pdo->prepare("SELECT id FROM Bougies WHERE id = ?");
        $stmtCheck->execute([$id]);
        if ($stmtCheck->rowCount() === 0) {
            header("Location: index.php?message=not_found");
            exit;
        }
    }

    // Suppression du stock lié (même logique qu'avant)
    $stmtStock = $pdo->prepare("DELETE FROM Stock WHERE bougie_id = ?");
    $stmtStock->execute([$id]);

    // Suppression de la bougie
    $stmtBougie = $pdo->prepare("DELETE FROM Bougies WHERE id = ?");
    $stmtBougie->execute([$id]);

    // Même message/redirect que l'original
    header("Location: index.php?message=suppression");
    exit;

} catch (PDOException $e) {
    // Comportement identique : arrêt et message d'erreur SQL
    die("Erreur : " . $e->getMessage());
}
```

- Vérification que la requête est bien une requête POST contenant un ID
- $id = (int) $_POST['id']; Conversion de l'ID en entier
- **Vérification CSRF :** Si activé, le code vérifie que le token envoyé correspond à celui en session.
- **Vérification connexion utilisateur et rôle admin :** Si activé, seuls les utilisateurs authentifiés (ou admin) peuvent supprimer.
- **Vérification que la bougie existe avant suppression :** Pour éviter de supprimer un enregistrement inexistant ou d'afficher une erreur SQL inutile.
- Supprimer le stock de la bougie
- Supprimer la bougie
- Redirection sur la page d'accueil
- Gestion des erreurs SQL
- **Gestion des erreurs SQL :** En cas d'erreur PDO, le script s'arrête et affiche le message d'erreur.

Conclusion

Je remercie l'équipe pédagogique ainsi que mon tuteur de stage pour cette formation et mon projet de stage

Merci pour votre attention

ABADIE Damien

Titre professionnel Développeur Web et Web Mobile 2025