Journal de Stage – 1et 2 Juillet 2025 :

Pendant ces trois jours de stage, j'ai progressé dans ma compréhension du langage HTML et de la structure d'une page web professionnelle. J'ai travaillé principalement sur un projet de site vitrine pour l'entreprise GCT Gabès, et cela m'a permis d'apprendre plusieurs éléments essentiels du développement web.

Ce que j'ai appris et compris :

1. ***La balise `<span>` :***
   J'ai découvert que `<span>` est utilisée pour mettre en valeur une partie du texte sans sauter de ligne. Par exemple, dans un titre comme :
   ```
   <h1>GCT <span>Gabès</span></h1>
   ```
   le mot "Gabès" peut recevoir un style différent (couleur, taille, etc.). C'est une balise très utile pour le styliser localement
2. ***La balise `<i>` :***
   J'ai compris que `<i>` ne sert plus seulement à mettre du texte en italique, mais qu'elle est souvent utilisée pour afficher des icônes, surtout avec une bibliothèque appelée Font Awesome.
   Exemple :
   ```
   <i class="fas fa-phone"></i>
   ```
   affiche une icône de téléphone.
3. ***Lien CDN (Font Awesome via Cloudflare) :***
   J'ai appris que cette ligne
   ```
   <linkrel="stylesheet"href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.4.0/css/all.min.css">
   ```
   permet de charger des icônes depuis Internet sans les télécharger, grâce au réseau CDN de Cloudflare. Cela rend le site plus rapide et plus moderne.

Concrètement, ce que j'ai appliqué :

- J'ai utilisé des `<span>` pour styliser du texte dans les titres.
- J'ai inséré des icônes (téléphone, enveloppe, téléchargement…) grâce aux balises `<i>`.
- J'ai intégré un lien CSS externe (Font Awesome) pour importer des icônes dans ma page.

Compétences développées :

- Utilisation correcte de balises HTML simples mais puissantes.
- Capacité à analyser un site web existant et comprendre sa logique.
- Familiarisation avec les ressources en ligne externes (CDN, bibliothèques d'icônes).

Ce que je souhaite encore améliorer :

- Comprendre la différence entre balises inline (`<span>`, `<i>`) et balises block (`<div>`, `<section>`).
- Apprendre à connecter le HTML au CSS pour styliser mes éléments proprement.
- M'initier au JavaScript pour rendre la page interactive.

Journal de Stage –le 03 Juillet 2025 :

Aujourd'hui, j'ai commencé à travailler sur la partie CSS de la page HTML réalisée les deux jours précédents. J'ai appris à organiser les règles de style de manière propre et à définir une identité visuelle cohérente pour le site.Ce que j'ai fait aujourd'hui :

Ce que j'ai appris :

1. Palette de couleurs avec `:root`

J'ai découvert qu'on peut définir des variables CSS globales dans `:root` comme :

```
:root {
  --primary: #004d33; /* vert foncé */
  --secondary: #0077cc; /* bleu */
}
```

Ça m'a aidée à garder une cohérence visuelle dans tout le site, et ça rend les futures modifications beaucoup plus faciles.

2. Le reset CSS

Au début du fichier CSS, on trouve :

```
* {

  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Cela supprime les styles par défaut du navigateur. C'est une bonne pratique que j'ai retenue pour repartir sur une base propre.

3. Header et logo

J'ai agrandi le logo du site (`height: 70px`) pour qu'il soit plus visible. J'ai aussi utilisé `display: flex` pour bien aligner les éléments dans la barre de navigation, et j'ai placé le bouton « Demander un devis » en haut à droite avec style.

4. Section Produits

J'ai utilisé une grille CSS avec `display: grid` pour afficher plusieurs produits côte à côte, de façon propre :

```
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
}
```

J'ai forcé **toutes les images des produits** à avoir la même hauteur (height: 200px; object-fit: cover;) pour qu'elles soient alignées, peu importe leur format d'origine. Le résultat est beaucoup plus esthétique.

5. Boutons dynamiques

Les boutons changent de couleur au survol avec `:hover`, ce qui rend l'interface plus interactive et moderne. J'ai aussi utilisé `border-radius` pour les rendre arrondis.

6. Responsive Design

J'ai appris à utiliser les media queries pour que le site s'adapte bien aux téléphones :

```
@media (max-width: 768px) {
  .main-nav {
    flex-direction: column;
  }
}
```

C'est très important pour offrir une bonne expérience utilisateur sur mobile.

Journal de stage – le 04 juillet :

Activités réalisées :

- Étude approfondie du code CSS pour la mise en page d'un site web.
- Compréhension et application des styles globaux, notamment la remise à zéro des marges et paddings avec `* { margin: 0; padding: 0; }` et `box-sizing: border-box`.
- Définition des variables CSS dans `:root` pour gérer les couleurs du site de façon simple et organisée.
- Mise en place des styles pour le `body` : choix de la police, couleur de fond et couleur du texte.
- Adaptation des images pour qu'elles soient responsives avec `max-width: 100%` et `height: auto`.
- Création d'un conteneur `.container` centré avec largeur adaptable pour différentes tailles d'écran.

Compétences développées :

- Maîtrise des variables CSS pour un code plus propre et facile à modifier.
- Initiation au design responsive grâce à la gestion des tailles et marges.
- Compréhension du fonctionnement des sélecteurs globaux et spécifiques.

Difficultés rencontrées :

- Difficulté à bien comprendre comment fonctionne `box-sizing` et pourquoi il est important pour la mise en page.
- Ajustement des tailles et marges pour que le site soit joli sur plusieurs tailles d'écran.

Solutions apportées :

- Consultation de ressources et tutoriels sur CSS variables et `box-sizing`.
- Tests sur plusieurs navigateurs et écrans pour valider les styles.

Journal de stage – le 05 juillet

Activités réalisées :

- Ajout et stylisation de la barre de navigation `.main-nav` avec Flexbox pour une disposition horizontale et alignée.
- Conception du logo et du titre avec image et texte, application des couleurs et tailles via variables CSS.
- Mise en forme des menus de navigation (`nav ul` et `nav ul li a`), suppression des puces, ajout d'effets au survol (`hover`).
- Stylisation de la section héro avec image de fond, dégradé sombre, texte centré et boutons d'appel à l'action (CTA) avec transitions et effets au survol.
- Création d'une grille de produits (`.product-grid`) avec CSS Grid, et cartes produit (`.product-card`) avec effets de survol et ombres portées.
- Stylisation du pied de page `.sales-footer` avec grille responsive et liens sociaux.

Compétences développées :

- Utilisation avancée de Flexbox et Grid pour organiser le contenu efficacement.
- Création d'effets visuels et interactions utilisateurs grâce aux transitions CSS.
- Adaptation des styles pour les petits écrans avec media queries (`@media`).

Difficultés rencontrées :

- Gestion des espaces entre éléments dans Flexbox et Grid.
- Compréhension des niveaux d'empilement avec `z-index` pour que certains éléments restent visibles au-dessus des autres.

Solutions apportées :

- Expérimentation avec différentes valeurs de marges, paddings et gaps.
- Recherche et analyse d'exemples de design responsive.
- Discussions avec le superviseur pour valider le rendu visuel.

Le 06 juillet : Mise en place de l'environnement de travail et création du serveur

- Installation de Node.js et configuration de l'environnement de développement.
- Initialisation du projet avec la commande `npm init` et création du fichier `package.json`.
- Installation des dépendances nécessaires : `express` pour le serveur web, `mongoose` pour la connexion à la base de données MongoDB, `dotenv` pour gérer les variables d'environnement, `nodemon` pour faciliter le redémarrage automatique du serveur durant le développement.
- Création du fichier `.env` contenant les variables sensibles comme le port du serveur et l'URL de connexion MongoDB.

- Développement du fichier principal `index.js` qui configure Express, connecte la base de données et lance le serveur.
- Test du lancement du serveur et confirmation de la connexion réussie à MongoDB à travers les messages dans la console.

Le 07 juillet : Modélisation de la base de données et création du contrôleur d'inscription

- Analyse des besoins pour stocker les données utilisateurs : nom, email, mot de passe.
- Création du modèle Mongoose (`user.model.js`) définissant le schéma des utilisateurs avec les champs obligatoires.
- Écriture du contrôleur `registerController` dans `usercontroller.js` pour gérer l'inscription.
- Implémentation de la validation des données reçues (vérification des champs requis).
- Vérification de l'existence préalable d'un utilisateur avec le même email pour éviter les doublons.
- Sécurisation du mot de passe via le hashage avec la bibliothèque `bcryptjs`.
- Sauvegarde du nouvel utilisateur dans la base de données.
- Envoi d'une réponse JSON confirmant la réussite de l'inscription ou détaillant les erreurs.
- Tests via Postman pour valider le fonctionnement du point de terminaison API.

Le 08 juillet : création de la page frontend et intégration avec le backend

- Conception d'une page HTML simple contenant le formulaire d'inscription (nom, email, mot de passe).
- Ajout d'un script JavaScript pour intercepter la soumission du formulaire, collecter les données et les envoyer à l'API via `fetch()`.
- Gestion des réponses du serveur pour afficher des messages clairs à l'utilisateur (succès ou erreur).
- Configuration correcte de l'URL du serveur dans la requête `fetch` avec le bon port.
- Tests pratiques dans un navigateur pour s'assurer de la bonne communication entre le frontend et le backend.

Le 09 juillet :Résolution des problèmes et amélioration de l'expérience utilisateur

- Identification et résolution d'un problème CORS (Cross-Origin Resource Sharing) bloquant les requêtes entre le frontend et le backend.
- Étude du fonctionnement de CORS et intégration de la bibliothèque `cors` dans le serveur Express.
- Ajout du middleware `cors()` dans `index.js` pour autoriser les requêtes provenant de différentes origines.

- Amélioration des messages d’erreur affichés à l’utilisateur pour plus de clarté.
- Tests complets pour garantir que les données sont bien enregistrées en base et que le système fonctionne sans erreur.
- Vérification des données dans MongoDB à l’aide d’outils comme MongoDB Compass.
