DULCA Constantin

Stage réalisé entre le 6 janvier 2025 et le 14 février 2025

BIJOUX FACTORY

I. Présentation

1) Présentation du stage

Dans le cadre de ma formation, j'ai effectué un stage de six semaines au sein de l'entreprise Bijoux Factory. Ce stage s'est déroulé du 6 janvier au 14 février, je l'ai trouvée en postulant à une offre d'emploi sur Indeed qui recherchait un développeur web pour le CMS Shopify. J'ai passé un premier entretien avec le gérant de l'entreprise et, suite à cela, il m'a fait passer un autre entretien avec l'informaticien engagé dans l'entreprise. L'objectif pour le patron était de savoir sur quoi il allait me faire travailler, lui n'ayant pas de compétence en informatique il a décidé de faire ainsi pour clarifier mes compétences. Les entretiens se sont déroulés en présentiel. J'ai choisi ce stage parce que je voulais apprendre à développer des sites web sur un CMS. Sur le marché du travail, de nombreuses offres d'emploi requièrent des compétences sur Shopify. Je pense donc que cette expérience me permettra d'acquérir des connaissances précieuses et d'ouvrir des opportunités pour mon avenir dans le développement web.

2) Présentation de l'organisation

Bijoux Factory est une entreprise spécialisée dans la fabrication de bijoux. Elle possède deux boutiques de vente en gros, situées rue Réaumur et rue du Temple à Paris, ainsi qu'une boutique de vente au détail.

L'entreprise gère également trois sites e-commerce :

- Niiki Paris et Jade & Julie, dédiés à la vente de bijoux sous leurs propres marques.
- Un site spécialisé dans le commerce B2B, destiné aux professionnels souhaitant acheter en gros.

3) Présentation de l'environnement réseau et des applications

L'environnement réseau et applicatif de Bijoux Factory est relativement simple et basé sur une infrastructure domestique :

L'entreprise fonctionne avec un modem internet fournissant une connexion Wi-Fi à ses PC.
Aucun serveur physique n'est utilisé pour héberger des applications ou des bases de données.
Google Drive est utilisé pour le stockage des données.
Un dossier spécifique est réservé à l'informaticien et à moi. Ce dossier contient du code et des fichiers liés au développement sur shopify et au CRM de l'entreprise.

L’entreprise utilise un CRM développé en interne par l’informaticien.
Ce CRM permet de gérer les prix, les ventes et les produits de l’entreprise.
Il est stocké sur Google Drive, Il est développée en Python, stockée sur Google Drive et utilisée pour la gestion des prix, ventes et produits de l’entreprise. Il fonctionne sans base de données et repose sur la génération de fichiers CSV pour stocker et manipuler les informations.

Shopify est utilisé pour la gestion des sites e-commerces c’est une plateforme e-commerce en SaaS (Software as a Service) qui permet de créer et de gérer facilement des boutiques en ligne. Elle est particulièrement prisée par les entreprises souhaitant vendre des produits sur internet sans avoir à gérer une infrastructure technique complexe.

Sur Shopify, la personnalisation avancée des pages et des fonctionnalités se fait à l’aide de Liquid, un langage de templating qui sert d’intermédiaire entre le backend et l’affichage du site. Liquid permet d’intégrer du code dynamique dans les thèmes Shopify, facilitant l’affichage et la modification des informations sans accès direct aux bases de données.

Contrairement à d’autres plateformes, Shopify ne donne pas d’accès direct à une base de données relationnelle. À la place, la gestion des informations produit, commandes et clients se fait via l’interface Shopify et peut être automatisée grâce à l’import/export de fichiers CSV. Ce système permet d’ajouter, modifier ou supprimer des produits en traitant des fichiers structurés, facilitant ainsi la gestion des catalogues à grande échelle.

II. Mission

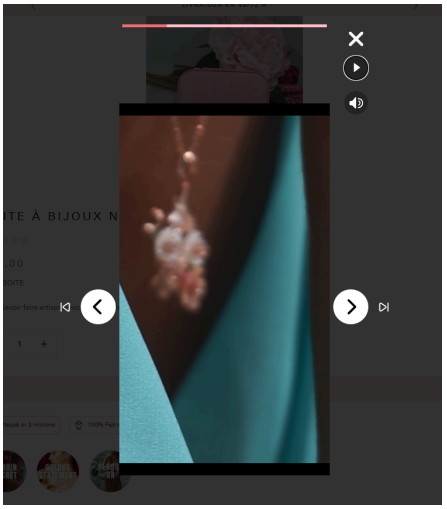
1) Développement d’un carrousel de vidéos façon instagram

a. Objectif

Sur chaque page produit, un système interactif de bulles vidéo a été mis en place afin d'améliorer la présentation des produits. Chaque produit dispose de plusieurs bulles, contenant chacune une vidéo différente permettant de mieux visualiser les détails et les caractéristiques du produit.

Chaque bulle est associée à un ensemble de vidéos que l’utilisateur peut faire défiler à l’aide de boutons blancs situés sous la vidéo. Des boutons transparents permettent de naviguer entre les différentes bulles, en passant à la précédente ou à la suivante.

L’utilisateur a également la possibilité de contrôler la lecture des vidéos en les mettant en pause ou en coupant le son. Une barre de progression, placée en haut de l’écran, indique l’avancement de la vidéo en cours.



b. Matériels, logiciels utilisés

Pour réaliser cette fonctionnalité, j’ai travaillé dans une balise liquid de shopify. Qu’est ce que c’est concrètement ? Shopify comme d'autres CMS propose un système de création de page intuitive permettant à un utilisateur d’ajouter une section, une image, ou un lien etc… mais il propose aussi une balise liquid qui permet d'injecter du code html css javascript et liquid, c’est dans cette balise que j’ai développé le système.

Pour structurer mon code avant de le copier coller j’ai utilisé Visual Studio Code.

voilà à quoi ressemble du code liquid :

```
{% if product.tags contains "faith" %}
```

```
{% if product.tags contains "faith" %}
    <div class="video-thumbnail">
        <a class="video-link">
            <img src="https://cdn.shopify.com/s/files/1/0795/4743/3286/files/faith_miniature.gif?v=1714750131" alt="Video Thumbnail" class="rounded-image">
        </a>
    </div>
{% endif %}
```

il s'exécute côté serveur, dans cette exemple on a une condition qui dit que si le produit demander par le client contient le tag “faith” on ajoute le code html avant le endif.

c. Description des tâches réalisées

Le code Liquid permet d'afficher des **bulles vidéo** sur les pages produits de Shopify en fonction des tags associés à chaque produit. Lorsqu'un produit contient un tag spécifique (comme "faith", "Jardin Secret", ou "Colors Statement"), une **bulle vidéo** correspondante est affichée, avec une image cliquable qui mènera à la vidéo associée. Les **GIFs** à afficher dans ces bulles seront décidés ultérieurement, mais le code est conçu pour accueillir ces vidéos dynamiquement, une fois les GIFs choisis. Si un produit possède un tag défini dans le code mais sans GIF associé, une bulle vide sera affichée. En outre, une bulle générique pour les avis clients est incluse à la fin du code, s'affichant pour tous les produits, indépendamment des tags. Ce système permet d'automatiser l'affichage des vidéos en fonction des tags des produits, et d'ajouter facilement les GIFs une fois sélectionnés.

côté javascript j'ai mis un événement au click sur les bulles

```
var lien = document.querySelectorAll('.video-link');
```

```
    lien.forEach((element,index) => {
        console.log(element);
        element.addEventListener('click',() => {
            openVideo(lesReels[index]);
        });
    });
```

Ce code est composé d'une variable qui contient un tableau récupéré en manipulant le DOM et après le foreach parcours chaque élément du tableau. et ajoute l'événement de clic qui lui-même exécute la procédure openVideo() qui prend en paramètre un autre tableau, alors oui lesReels est un tableau de tableau et ces tableaux contiennent les liens des vidéo qui sont sur shopify.

La procedure openVideo() a pour fonction de changer des variables déclaré en dehors de celle-ci qui elles même serviront à faire fonctionner la suite du programme

```
898     function openVideo(videoUrls) {
899         indexReel = lesReels.findIndex(function(reel, i) {
900         // Comparer les deux tableaux (vidéos)
901             return JSON.stringify(reel) === JSON.stringify(videoUrls);
902         });
903
904
905         videoPlaylist = videoUrls;
906         isPlaying = true;
907         if(currentVideoIndex == undefined){
908             currentVideoIndex = 0;
909         }
910
911         displayNavbar.style.display = 'none';
912         document.querySelector('body').style.overflowY = 'hidden';
913         playNextVideo();
914
915     }
```

La variable indexReel prend l'indice du premier élément dans lesReels qui correspond à videoUrls. videoPlaylist prend la valeur passé en paramètre. isPlaying s'initialise à true;

```
    function playNextVideo() {

miniatureIconeVideoStart.children[0].src=iconeVideo[indexReel].children
[0].children[0].currentSrc

console.log(iconeVideo[indexReel].children[0].children[0].currentSrc);
        if(isPlaying) {
            var modal = document.getElementById('video-modal');
            var videoPlayer = document.getElementById('video-player');
            var sourceElement =
document.getElementById('video-source');
            updateProgressBars();
            var videoUrl = videoPlaylist[currentVideoIndex];
            videoPlayer.src = videoUrl;
            sourceElement.setAttribute('src', videoUrl);
            videoPlayer.autoplay = true;
            modal.style.display = 'block';
        }
    }
```

la procédure playNextVideo() à pour fonction de mettre le lien de la vidéo sur la balise source faire apparaître l'interface de vidéo pour l'utilisateur c'est à dire les boutons et la bar de progressions et lancer la vidéo.

```
    function updateProgressBars() {
        var progressBarsContainer =
document.getElementById('progress-bars');
        progressBarsContainer.innerHTML = '';

        for (var i = 0; i < videoPlaylist.length; i++) {
            var progressBar = document.createElement('div');
            progressBar.classList.add('progress-bar');
            if (i === currentVideoIndex) {
                progressBar.classList.add('active');
            }
            progressBarsContainer.appendChild(progressBar);
        }
        document.documentElement.style.setProperty('--num-videos',
videoPlaylist.length);
```

```
    }
```

La procédure updateProgressBar à pour rôle de créer la bar de progression des vidéo

```
videoDisplay.addEventListener('timeupdate',barProgression);
```

```
    function barProgression(){
        videoEnCoursBarre = document.querySelector('.active');
        const currentTime = videoDisplay.currentTime; // Temps écoulé
        const duration = videoDisplay.duration; // Durée totale de la
vidéo

        // Calculer le pourcentage écoulé de la vidéo
        const elapsedPercentage = (currentTime / duration) * 100;

        // Mettre à jour la barre de progression avec un gradient
dynamique

        videoEnCoursBarre.style.background = `linear-gradient(to right,
#F47174 ${elapsedPercentage}%, #ffc0cb 0%)`;
    }
```

la procédure barProgression() calcule le temps restant de la vidéo et attribue une propriété css qui se met à jour pour faire progresser la ligne sur la bar de progression cette fonction est exécutée avec l'événement timeupdate sur l'élément html video il a pour rôle d'exécuter la fonction quand la vidéo est en cours de lecture.

```
btnPause.addEventListener('click',takePauseVideo);
    function takePauseVideo(){
        var iconeSuspens = document.querySelector('.suspend-btn');
        var iconePlay = document.querySelector('.play-btn');
        if(videoDisplay.paused == false){
            videoDisplay.pause();
            iconeSuspens.style.display = 'none';
            iconePlay.style.display = 'block';
        }else{
            videoDisplay.play();
            iconeSuspens.style.display = 'block';
            iconePlay.style.display = 'none';
        }
    }
```

La procédure takePauseVideo() met la vidéo en pause lors du click sur le bouton et change également l'icône pour faire comprendre à l'utilisateur que le bouton change de fonction quand la vidéo est en pause

```
    function btn_right() {
        console.log(currentVideoIndex <= videoPlaylist.length-1);
        if (currentVideoIndex <= videoPlaylist.length-2) {
            currentVideoIndex++;
        }else{
            currentVideoIndex = 0;
            btn_right_reel()
        }
        console.log(videoPlaylist.length -1);
        console.log("currentVideo = "+currentVideoIndex);
        console.log("indexReel = "+indexReel);
        console.log(iconeVideo[indexReel].children[0]);
        clearTimeout(timeoutID)
        playNextVideo()
    }
```

la procédure btn_right va changer la vidéo en passant à la suivante si il n'y a plus de vidéo pour une "storie" elle va passer à la suivante c'est la procédure btn_right_reel qui fait cela

il y a aussi la procédure btn_left et btn_left_reel elles font exactement la même chose mais en passant à la vidéo précédente et à la storie précédente. les procédure btn_right_reel sont aussi associées aux bouton gauche et droit de l'interface qui permettent de changer de storie automatiquement

```
    function closeVideo() {
        clearTimeout(timeoutID)
        isPlaying = false;
        currentVideoIndex = undefined;
        var modal = document.getElementById('video-modal');
        var videoPlayer = document.getElementById('video-player');
        videoPlayer.src = '';
        modal.style.display = 'none';
        displayNavbar.style.display = 'grid';
        document.querySelector('body').style.overflowY = 'scroll';
        document.querySelector('marquee-text').style.display = 'block';
        document.querySelector('.content-tabs').style.visibility =
'visible';
    }
```

et la procédure closeVideo va fermer l’interface interactive et va supprimer le lien de la vidéo

2) Correction de bug d’affichage

a. Objectif

Sur la page du site Niiki Paris, l'ancien stagiaire avait tenté d’ajouter une section dynamique pour afficher les dates de livraison estimées en utilisant une balise Liquid. Cependant, cela a entraîné plusieurs problèmes d’affichage. Le principal souci venait du fait que la section contenant la carte bleue était une section Shopify prédéfinie, tandis que la section de livraison était codée en Liquid. Cette différence de structure a causé un mauvais alignement des éléments, rendant la mise en page incohérente. De plus, l’image du camion de livraison était beaucoup trop grande par rapport à celle de la carte bleue.

b. Matériels, logiciels utilisés

Pour identifier et corriger ces problèmes d’affichage, j’ai utilisé le débogueur du navigateur ainsi que Visual Studio Code afin d’analyser plus précisément le code de la balise Liquid. Grâce aux outils de développement intégrés au navigateur, j’ai pu inspecter les éléments de la page, comprendre la structure des sections et repérer les différences entre la section Shopify prédéfinie et celle générée en Liquid.

c. Description des tâches réalisées

Sur les pages produits, un problème était présent avec une icône qui ne s'affichait pas correctement. Pour corriger cela, il a fallu intervenir dans le code Liquid de la section et attribuer une classe CSS déjà définie dans le fichier principal du thème Shopify. Ce fichier centralise les styles utilisés sur l'ensemble du site. En appliquant cette classe existante à l'icône sur les pages produits, son affichage a été restauré sans nécessiter d'ajout de code CSS supplémentaire.

Un autre problème à était résolu sur deux icônes qui n'étaient pas bien alignées cela était dû au fait que les propriété CSS ne s'appliquait pas cela était dû à la feuille de style principal du projet shopify theme.css des propriété était appliqué avec le l'élément ‘!important’ ce qui masquait toute propriété qui pouvaient venir de classe prioritaire, pour réparer cela il a fallu mettre le mot clé ‘!important’ sur les propriétés qui ne s'appliquaient pas.

3) Développement de la page de la marque

a. Objectif

Il a été demandé de développer une page web spécialement pour la marque Niiki Paris. La page initiale était statique, sans sections distinctes, se limitant à des paragraphes accompagnés de quelques images. L'objectif de cette refonte était de donner plus de dynamisme à la page en y ajoutant des animations et des éléments interactifs, afin de la rendre plus vivante et attrayante pour les visiteurs.

b. Matériels, logiciels utilisés

Visual studio code pour développer la page.

c. Description des tâches réalisées

Sur Shopify, il est possible de créer des pages grâce à une interface dédiée qui permet d'ajouter du contenu de manière intuitive. Cette interface propose un éditeur où l'on peut insérer du texte, des images et d'autres éléments sans avoir à coder directement. Cependant, j'ai rencontré un problème : chaque page affiche obligatoirement un titre par défaut, sans option pour le masquer via l'interface.

Pour résoudre ce problème, j'ai manipulé le DOM en JavaScript en insérant des lignes de code avant mon HTML, CSS et JavaScript principal. J'ai d'abord supprimé les balises inutiles, puis j'ai remonté la balise concernée afin d'adapter l'affichage du contenu et d'éviter que le titre imposé par Shopify ne perturbe la mise en page souhaitée.

J'ai également ajouté un événement au scroll permettant de détecter lorsque l'utilisateur atteignait une section spécifique de la page. À ce moment-là, les textes apparaissaient progressivement pour donner un effet dynamique et interactif. Ce code a été implémenté en utilisant les méthodes du DOM, notamment la manipulation des classes en JavaScript. En ajoutant et supprimant des classes selon la position de l'utilisateur sur la page, j'ai pu contrôler l'animation des éléments et améliorer l'expérience visuelle.

4) Sélection et Assemblage de PC sur Mesure

a. Objectif

Ayant constaté que les PC de l'entreprise étaient trop lents – il s'agissait de modèles tout-en-un avec l'écran intégré comme unité centrale – j'ai su qu'il était possible de monter des PC sur mesure offrant de bien meilleures performances pour un budget

équivalent. J’ai donc proposé cette solution à mon maître de stage, qui a accepté et m’a fait confiance. Après avoir sélectionné les composants adaptés et recommandé les pièces à acheter, j’ai pu assembler deux nouveaux PC pour l’entreprise, améliorant ainsi considérablement leur efficacité.

b. Matériels, logiciels utilisés

j’ai utilisé un navigateur web afin de rechercher et comparer les composants sur différents sites de e-commerce. Une fois les pièces sélectionnées et commandées, j’ai utilisé un tournevis pour assembler les PC et une clé usb pour installer windows 11.

c. Description des tâches réalisées

Pour le choix des composants, j’ai sélectionné un processeur AMD Ryzen 5 4600G, un CPU intégrant un chipset graphique. Cela a permis d’éviter l’achat d’une carte graphique dédiée, qui n’était pas nécessaire pour l’usage prévu par l’entreprise, tout en optimisant le budget.

Concernant la carte mère, plusieurs critères ont été pris en compte : la compatibilité du socket avec le processeur et la version du BIOS, qui doit être à jour pour supporter le CPU. Pour éviter tout problème de compatibilité, j’ai opté pour une carte mère disposant de la fonctionnalité BIOS Flashback. Cette option est très pratique, car elle permet de mettre à jour le BIOS directement via une clé USB et un bouton dédié, sans avoir besoin d’installer un processeur plus ancien pour effectuer la mise à jour. Finalement il n’ y a pas eu de problème de compatibilité le BIOS était à jour. Le dissipateur thermique et le ventilateurs était inclus avec le CPU, la pâte thermique était également présente il a ensuite fallu prendre un boitier PC, une alimentation et des barrettes de ram j’ai pris 16 go de ram j’ai également pris des carte wifi parce que les ordinateurs à la base sont connectés au wifi et j’ai fait en sorte que l’entreprise n’aie pas à changer ses habitudes j’ai pris des carte wifi pour les PC.

III. Conclusion

En conclusion, ce stage m’a permis d’acquérir une expérience en travaillant avec le CMS Shopify, notamment dans le développement et la personnalisation de pages. J’ai également amélioré mes compétences en JavaScript côté client, en appliquant des solutions dynamiques pour améliorer l'interaction avec les utilisateurs. Ce stage m’a aussi permis de me familiariser avec la relation entre le développeur et le client, en apprenant à comprendre ses besoins et à anticiper ceux qu’il pourrait avoir. Cela m’a conduit à adapter mon code de manière flexible, afin d’éviter de devoir tout refaire et d’optimiser ainsi la pérennité de mes développements.

De plus, ce stage m’a permis d’améliorer mes compétences dans l’analyse des besoins matériels pour une entreprise. En choisissant des PC à la fois moins chers et plus puissants, j’ai appris à évaluer les meilleures options en fonction des exigences spécifiques de l'entreprise. Cela m’a aussi donné l’occasion de proposer des solutions adaptées, en optimisant les coûts tout en améliorant les performances des équipements.