

<b>Activité-type *</b>	ACT 1
<b>CP *</b>	Installer et configurer son environnement de travail en fonction du web ou web mobile

**1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :**

Pour le projets final je me suis pencher sur une boutique en ligne de posters, pour réaliser ce projet j'avais besoin d'un environnement stable ( qui me suivent tous au long de l'aventure ), c'est pour cela que j'ai fais une liste des ensemble de technologie qui me serais utile.

**VSCode :**

Pour commencer je me suis pencher sur Vscode un éditer de code simple, ergonomique et accessible, Vscode vas me permettre d'écrire du code avec plus de visibilité ( avec plusieurs couleur qui différencie plusieurs partie de mon code ), d'avoir un rapport de bug ( chaque erreur et annoter de sa ligne et de l'erreur ) et d'avoir un choix d'autocomplétion qui me permet d'aller plus vite et plus efficacement . Pour cela je me suis diriger sur le site officiel de Vscode (<https://code.visualstudio.com> ), puis j'ai installer le .dmg ( .dmg pour macOs ) pour avoir l'application Vscode Desktop sur ma machine.

**Docker :**

Ensuite j'avais besoin d'un environnement stable pour lancer et accéder a ma Base De Donnée surement et sainement, alors je me suis tourner vers docker qui me permet de créer des “Container” pour ensuite les lancer sans problème d'un seul clique ( ou au démarrage de ma machine ). Pour cela je suis aller sur le site officiel de docker ( <https://www.docker.com> ), puis j'ai cliquer sur “Download Docker Desktop”, se qui ma installer le .dmg qui me permet d'accéder directement a l'application desktop de Docker sur ma machine.

**Adminer :**

Après ma mise en action de Docker, je me suis tourner vers Adminer ( Adminer et un logiciel de gestion de base de données, comme par exemple PHPmyAdmin ) pour avoir un retour graphique et un accès complet a ma Base De Donnée MariaDB.

**Node :**

J'avais besoin d'un serveur pour lancer ma Base De Donnée et gérer les routes de mon API ( Coter “Backend” ), ect... . Alors je suis aller chercher du coter de Node. Pour l'instalation je suis

passer par Nvm avec les commande suivantes qui sont fournie sur le site officiel de Node (<https://nodejs.org/fr/download/current>),

```
1 # Télécharger et installer nvm :
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
3
4 # au lieu de redémarrer le shell
5 \. "$HOME/.nvm/nvm.sh"
6
7 # Télécharger et installer Node.js :
8 nvm install 24
9
10 # Vérifier la version de Node.js :
11 node -v # Doit afficher "v24.3.0".
12 nvm current # Doit afficher "v24.3.0".
13
14 # Vérifier la version de npm :
15 npm -v # Doit afficher "11.4.2".
```

ces commandes me permet d'installer Node sous sa version final qui est la 24.3.0 .

### Angular :

Pour le coter “Frontend” de mon application web ( Coter vue utilisateur ), je suis passer par un “FrameWork” ( definition :  Framework ), parmit les “FrameWork Frontend” je me suis tourner vers Angular qui est un “FrameWork” assez réputé et accessible, l'ayant dejas utiliser au paravent cela justifie encore plus mon choix. L'intallation d'Angular a etais tres simple, vu que précédament j'ai installer Node, ce service me permet grave a NPM d'intaller d'autre service tres facilement, donc grave a la commande suivante j'ai pu instantier Angular dans mon projet.

```
npm install -g @angular/cli
```

### 2. Précisez les moyens utilisés :

Matériel :

- Macbook pro 2019 sous MacOs Sequoia 15.6
- Processeur 1,4 Ghz Intel Core i5

Logiciel :

- MacOs
- Visual Studio Code
- Github
- Terminal
- Docker

**3. Avec qui avez-vous travaillé ?**

Pour installer mon environnement de travail j'ai étais seul.

**4. Contexte**

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
---	-------------------------

Chantier, atelier, service▶	
-----------------------------	--

Période d'exercice ▶ Du : <b>06/06/2025</b> au : <b>21/07/2025</b>
--

**5. Informations complémentaires (facultatif)**

<b>Activité-type *</b>	ACT 1
<b>CP *</b>	Maquetter des interfaces utilisateur web ou web mobile

## Introduction :

Dans le cadre de ce projet de formation, il nous a été demandé de maquetter une interface pour une application web. Ce travail a pour objectif de mettre en pratique les compétences acquises en conception d'interfaces utilisateur, en respectant les principes d'ergonomie, d'esthétique et de fonctionnalité. Il s'agit d'une étape essentielle dans le processus de développement, permettant de visualiser l'apparence et l'organisation de l'application avant sa réalisation technique.

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour ce projet de conception d'interface web, j'ai utilisé l'outil **Figma**, accessible en ligne, pour créer à la fois les **Wireframes** et la maquette finale de l'application.

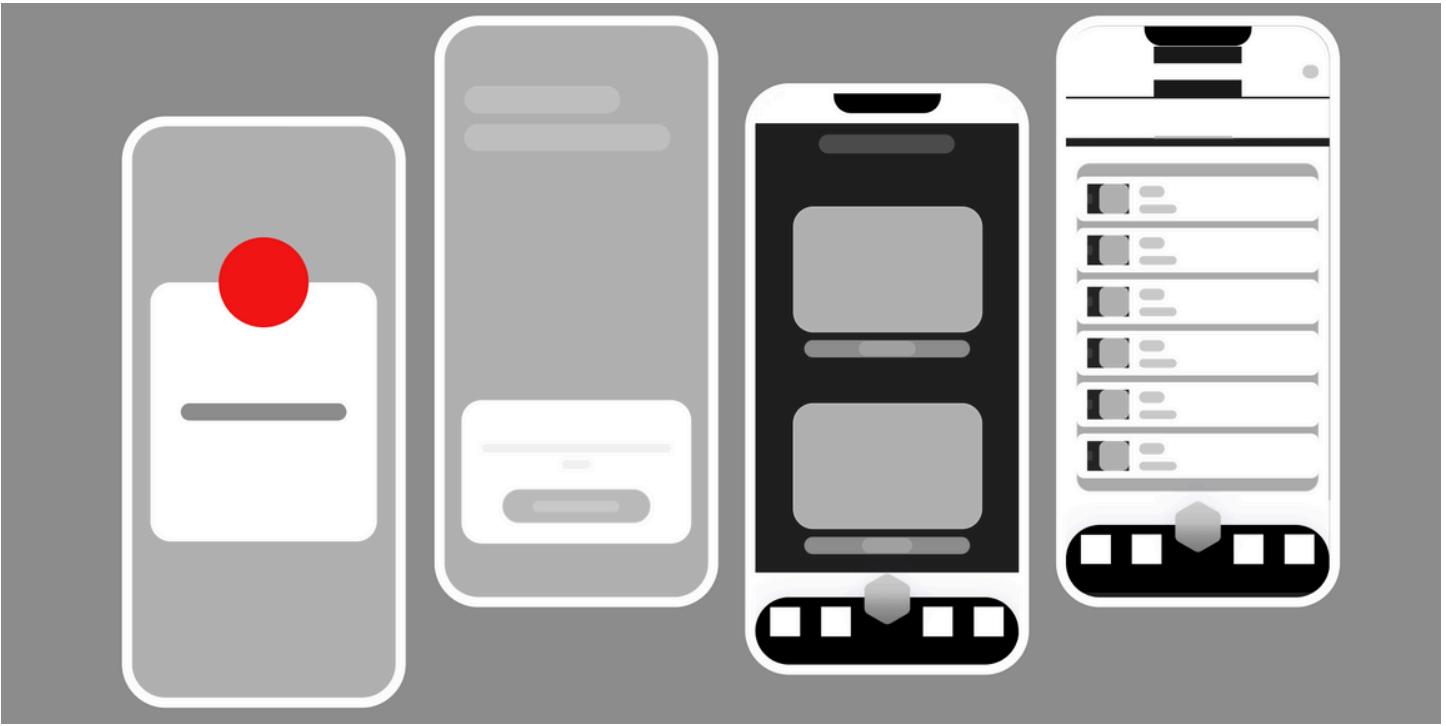
## Installation :

**Figma** ne nécessitant aucune installation lourde, je me suis inscrit directement sur le site officiel ([www.figma.com](http://www.figma.com)) et ai pu accéder à l'outil via un navigateur web. Pour plus de fluidité, j'ai également installé la version desktop.

## Réalisation :

### • Crédation des wireframes :

J'ai débuté le projet par la réalisation de **wireframes**, c'est-à-dire des maquettes en noir et blanc destinées à structurer l'interface de manière simple. Cette étape m'a permis de définir l'emplacement des éléments principaux (menus, boutons, images, zones de texte, etc.) sans me concentrer encore sur l'aspect graphique.

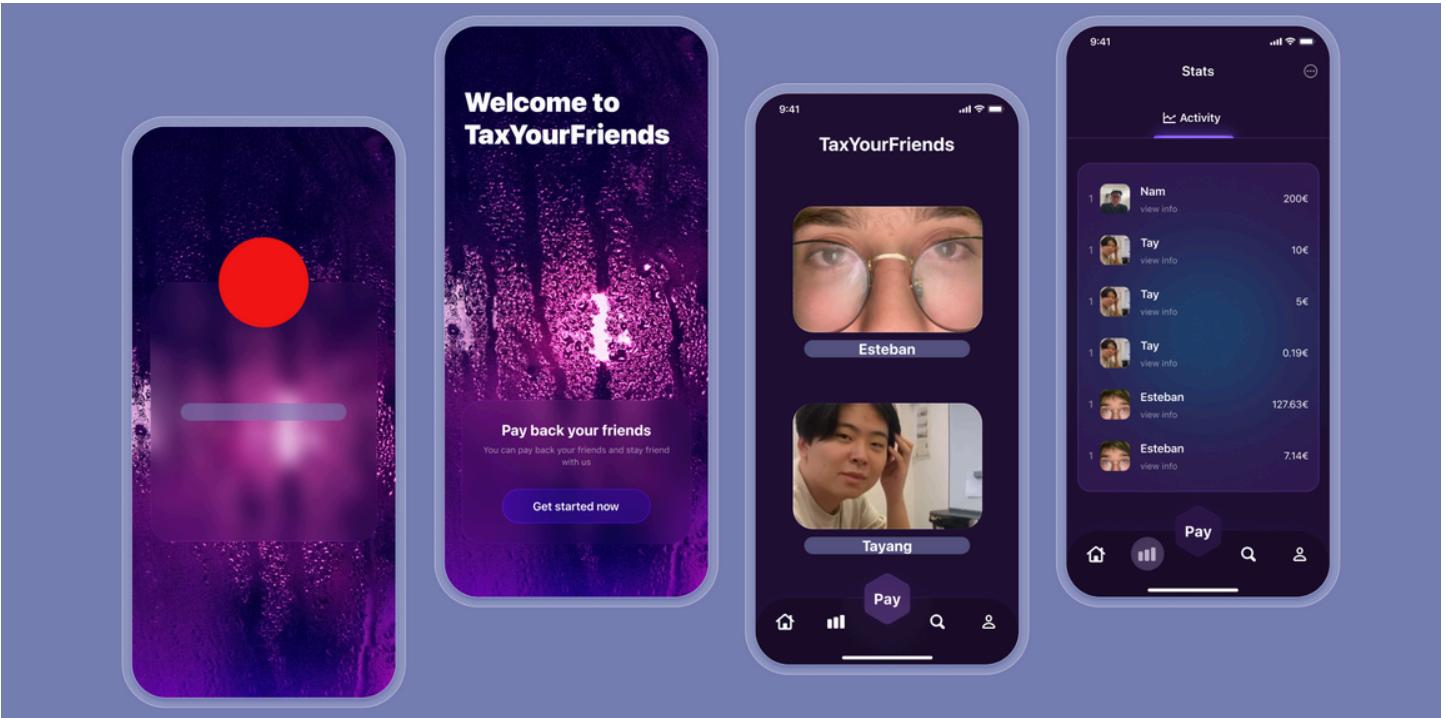


- **Maquette finale:**

Une fois les **wireframes** validés, je suis passé à la conception graphique de la maquette finale.

J'ai alors :

- choisi une **charte graphique** adaptée (couleurs, typographies, icônes),
- stylisé les composants (boutons, barres de navigation, zones de contenu),
- intégré des **photos et illustrations** pour rendre l'interface plus visuelle et attrayante.



- **Prototypage interactif :**

J'ai ajouté des **liens entre les pages** pour simuler la navigation dans l'application. Cela m'a permis de tester l'ergonomie et la fluidité du parcours utilisateur.

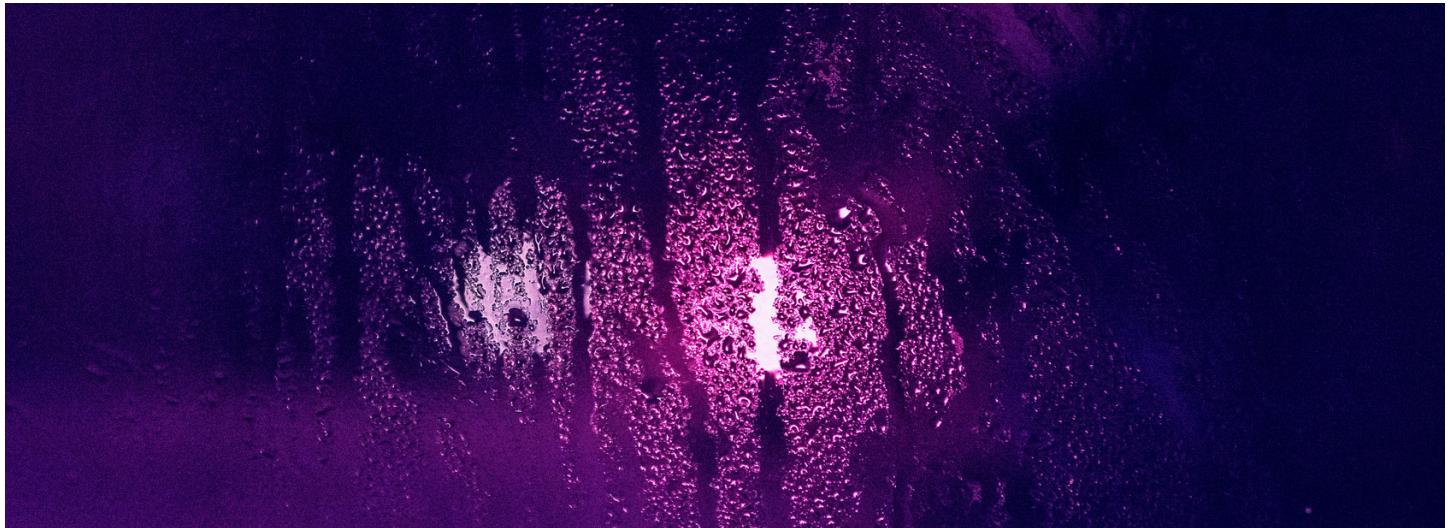
## 2. Précisez les moyens utilisés :

### Outils:

- **Figma** : outil principal utilisé pour concevoir les **wireframes** et la maquette finale. Il m'a permis de créer une interface interactive, de définir les composants graphiques, et de simuler la navigation entre les pages.
- **Navigateur web (Firefox)** : pour accéder à la version en ligne de Figma.
- **Macbook pro 2019 intel Core i5** : utilisé pour travailler sur l'ensemble du projet .

### Ressources graphiques :

- **Bibliothèques d'icônes intégrées à Figma** : pour gagner du temps et respecter les standards UI.
- **Banques d'images libres de droit (Unsplash)** : pour intégrer un background attrayant .



## 3. Avec qui avez-vous travaillé ?

Pour ce projet, j'ai travaillé seul. L'ensemble des étapes – de l'analyse du cahier des charges jusqu'à la réalisation finale de la maquette sur Figma – a été effectué en autonomie. Cela m'a permis de gérer moi-même l'organisation du travail

#### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
---	-------------------------

Chantier, atelier, service▶	
-----------------------------	--

Période d'exercice ▶ Du : **04/02/2025**au :**12/04/2025**

#### 5. Informations complémentaires (facultatif)

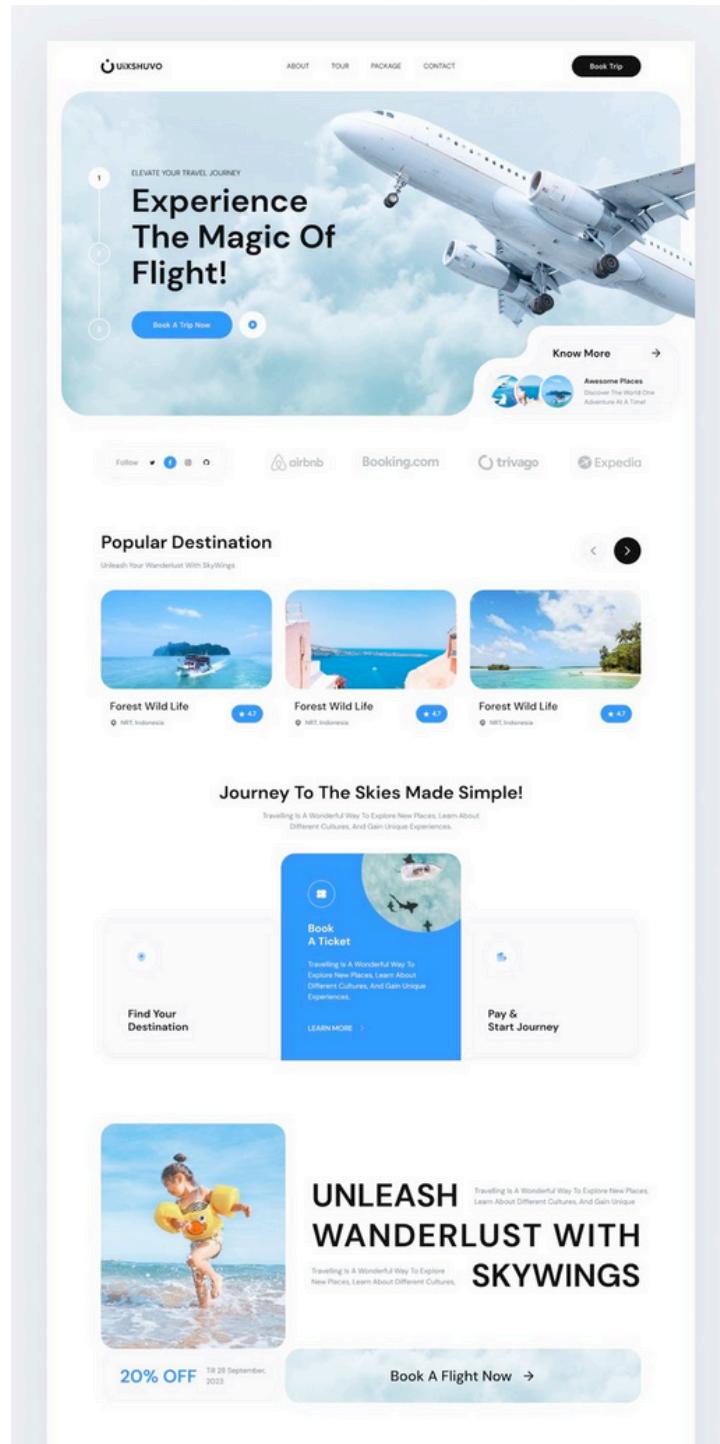
<b>Activité-type *</b>	ACT 1
<b>CP *</b>	Réaliser des interfaces statiques web ou web mobile

**1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :**

Dans le cadre de ce projet, il nous a été demandé de créer un site web statique à partir d'une maquette conçue sur Figma. Le thème que j'ai choisi pour mon site était l'aviation.

Voici les principales tâches que j'ai réalisées :

- **Étude de la maquette Figma** : analyse des différentes pages, disposition des éléments (header, images, blocs de texte, boutons), et respect de la charte graphique.



- **Création du code HTML** : structure de chaque page à l'aide de balises sémantiques (`<header>`, `<main>`, `<section>`, `<footer>`, etc.).



```
<header>
    <div class="logo">
        
    </div>
    <div class="burger-menu">
        <a href="#"><li>About</li></a>
        <a href="#"><li>Tour</li></a>
        <a href="#"><li>Package</li></a>
        <a href="#"><li>Contact</li></a>
    </div>
    <nav>
        <a href="#"><li>About</li></a>
        <a href="#"><li>Tour</li></a>
        <a href="#"><li>Package</li></a>
        <a href="#"><li>Contact</li></a>
    </nav>
    <button>Book trip</button>
    <button class="burger"><i class="fa-solid fa-bars"></i></button>
</header>
```



```
<div class="main">
    <h2>Popular Destination</h2>
    <p>Unleash Your Wanderlust With SkyWings</p>
    <br>
    <div class="case">
        <div class="case1">
            <a href=""></a>
            <h3>Forest Wild Life</h3>
            <p><i class="fa-solid fa-location-dot"></i> NRT, Indonesia</p>
        </div>
        <div class="case2">
            <a href=""></a>
            <h3>Forest Wild Life</h3>
            <p><i class="fa-solid fa-location-dot"></i> NRT, Indonesia</p>
        </div>
        <div class="case3">
            <a href=""></a>
            <h3>Forest Wild Life</h3>
            <p><i class="fa-solid fa-location-dot"></i> NRT, Indonesia</p>
        </div>
    </div>
</div>
```



```
<footer>
    <div class="OFF">
        <h2>20% OFF</h2>
    </div>
    <div class="text">
        <p>Till 28 Septembre,<br>2023</p>
    </div>
    <button>Book A Flight Now -></button>
</footer>
```

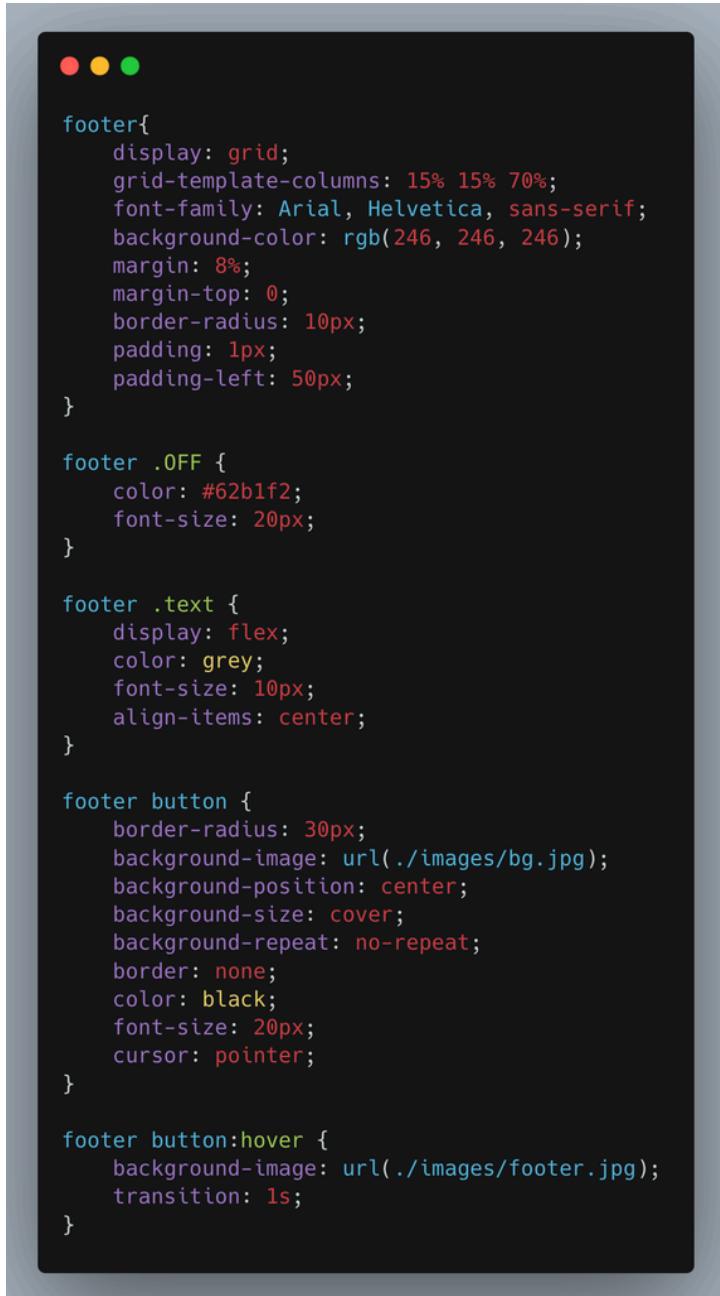
- **Mise en forme avec CSS** : application des styles pour respecter la maquette (typographie, couleurs, marges, mise en page responsive).



```
body {  
    padding: 1%;  
    background-color: white;  
    overflow-x: hidden;  
}  
  
nav {  
    display: grid;  
    grid-template-columns: 15% 15% 15% 15%;  
    justify-content: center;  
    align-items: center;  
    list-style-type: none;  
    font-size: 20px;  
}
```



```
header {  
    display: grid;  
    grid-template-columns: 1fr 2fr 1fr;  
    margin-bottom: 2%;  
}  
  
header button {  
    background-color: black;  
    color: white;  
    border: none;  
    border-radius: 50px;  
    font-size: 15px;  
    cursor: pointer;  
}
```



```
footer{  
    display: grid;  
    grid-template-columns: 15% 15% 70%;  
    font-family: Arial, Helvetica, sans-serif;  
    background-color: rgb(246, 246, 246);  
    margin: 8%;  
    margin-top: 0;  
    border-radius: 10px;  
    padding: 1px;  
    padding-left: 50px;  
}  
  
footer .OFF {  
    color: #62b1f2;  
    font-size: 20px;  
}  
  
footer .text {  
    display: flex;  
    color: grey;  
    font-size: 10px;  
    align-items: center;  
}  
  
footer button {  
    border-radius: 30px;  
    background-image: url("./images/bg.jpg");  
    background-position: center;  
    background-size: cover;  
    background-repeat: no-repeat;  
    border: none;  
    color: black;  
    font-size: 20px;  
    cursor: pointer;  
}  
  
footer button:hover {  
    background-image: url("./images/footer.jpg");  
    transition: 1s;  
}
```

- **Test dans le navigateur** : vérification du rendu dans plusieurs tailles d'écran (responsive, design de base).

ELEVATE YOUR TRAVEL JOURNEY

# Experience The Magic Of Flight!

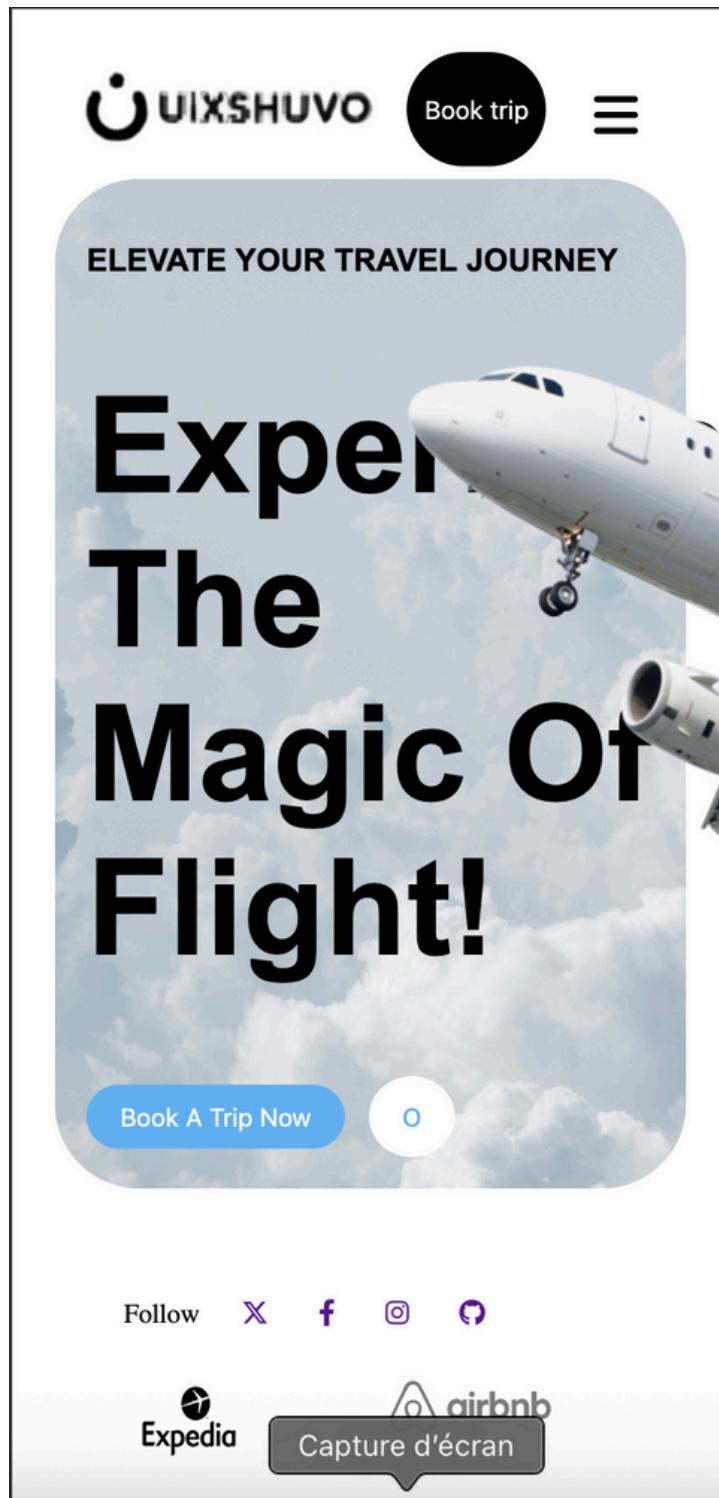
[Book A Trip Now](#)

Follow [X](#) [f](#) [i](#) [o](#)



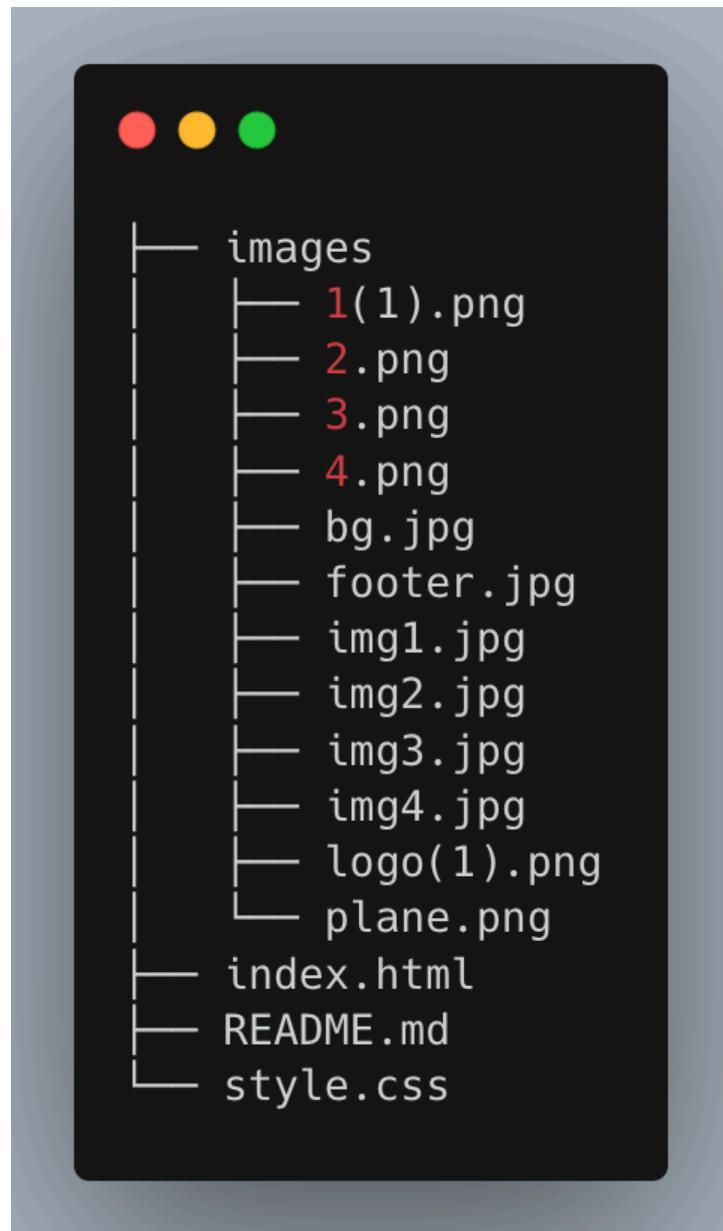
Booking.com





## 2. Précisez les moyens utilisés :

**Visual Studio Code** : éditeur de code utilisé pour écrire le HTML et le CSS du site. Il m'a permis d'organiser mes fichiers, de bénéficier de la coloration syntaxique et d'extensions utiles (comme Live Server).



**Figma** : utilisé en amont pour consulter la maquette fournie et reproduire fidèlement la structure et le style du site.

**Navigateur web (Firefox)** : pour visualiser le rendu du site pendant son développement et tester le responsive.

### 3. Avec qui avez-vous travaillé ?

J'ai travaillé **seul**, depuis mon **ordinateur personnel**, à l'aide de l'éditeur de code **Visual Studio Code**, d'un **navigateur web** (Firefox).

#### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
---	-------------------------

Chantier, atelier, service▶	
-----------------------------	--

Période d'exercice ▶ Du : **21/01/2025** au :**28/01/2025**

#### 5. Informations complémentaires (facultatif)

<b>Activité-type *</b>	ACT 1
<b>CP *</b>	Développer la partie dynamique des interfaces utilisateur web ou web mobile

Pour élargir mes compétences, je me suis donné un projet personnel intitulé "**PlanetCode**". Ce projet consiste à créer un **site interactif** permettant d'explorer les plus beaux endroits de la planète. Il m'a permis de mettre en pratique mes connaissances en **HTML, CSS, JavaScript**, tout en découvrant **Three.js**, une bibliothèque JavaScript dédiée à la création de scènes 3D dans le navigateur.

**1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :**

**Structure HTML du site** : création de plusieurs sections (accueil, destinations, carte 3D, contact) pour organiser les contenus.

```
Icon/favicon
index.css
index.html
index.js
main.js
models
├── earth
│   ├── license.txt
│   ├── scene.bin
│   └── scene.gltf
└── images
    ├── africa.jpg
    ├── africaMap.png
    ├── bg.jpg
    ├── china.jpg
    ├── france.jpg
    ├── kilimandjaro.avif
    ├── Namabi.avif
    ├── russia.jpg
    ├── Senegal.jpg
    ├── usa.jpg
    ├── user.jpeg
    └── user.JPG
        └── vietnam.jpg
package-lock.json
package.json
vite.config.js.png
index.html
README.md
style.css
```



```
<nav>
    <div class="logo">
        <h2>Planet<span>Code</span></h2>
    </div>
    <div class="links">
        <ul>
            <a href=".//index.html"><li>Home</li></a>
            <a href="#" id="Next2"><li>About Us</li></a>
            <a href="#" id="Gatewaybtn"><li>Gateway</li></a>
            <a href="#"><li>Contact</li></a>
        </ul>
    </div>
    <div class="user">
        <i class="fa-solid fa-circle-user"></i>
    </div>
</nav>
```



```
<div class="Gateway">
    <div class="CityContainer">
        <a href="#" id="africaOpen"><div class="case africa">
            <p>africa</p>
        </div></a>
        <div class="case france">
            <p>france</p>
        </div>
        <div class="case USA">
            <p>USA</p>
        </div>
        <div class="case China">
            <p>China</p>
        </div>
        <div class="case Russia">
            <p>Russia</p>
        </div>
        <div class="case Vietnam">
            <p>Vietnam</p>
        </div>
    </div>
</div>
```

```
<div class="fullAfrica">
    <i class="fa-solid fa-arrow-left africaArrow"></i>
    
    <div class="mapBtn">
        <button id="africaBtn1">1</button>
        <button id="africaBtn2">2</button>
        <button id="africaBtn3">3</button>
    </div>
    <div class="africaCaseSenegal">
        <p>Le lac Retba, lac rose du Sénégal</p>
    </div>
    <div class="africaCaseNamibia">
        <p>Le désert du Namib, un des plus anciens déserts du monde</p>
    </div>
    <div class="africaCaseTanzanie">
        <p>Le désert du Namib, un des plus anciens déserts du monde</p>
    </div>
</div>
```

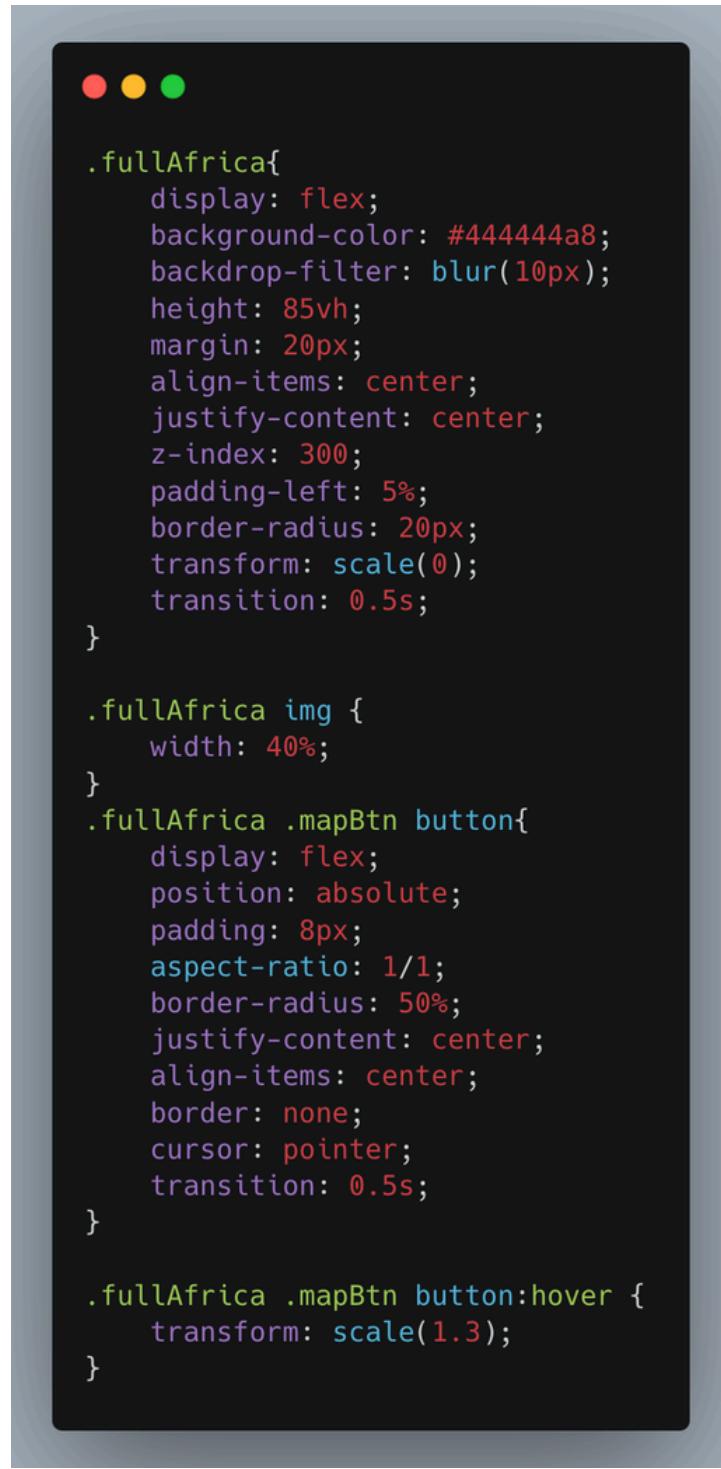
**Mise en page avec CSS** : stylisation de l'interface (typographie, couleurs, mise en page responsive) pour rendre le site agréable et adapté à tous types d'écrans.

```
canvas{
    cursor: grab;
    position: absolute;
    z-index: 200;
}

canvas:active{
    cursor: grabbing;
}

nav{
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    padding-top: 20px;
    z-index: 300;
}
```

```
.Gateway{  
    display: flex;  
    position: absolute;  
    backdrop-filter: blur(10px);  
    margin: 20px;  
    top: 10%;  
    width: 97%;  
    height: 80vh;  
    border-radius: 20px;  
    justify-content: center;  
    align-items: center;  
    transform: scale(0);  
    transition: 1s;  
    z-index: 300;  
}  
  
.CityContainer{  
    display: grid;  
    grid-template-columns: 1fr 1fr 1fr;  
    width: 90%;  
    height: 90%;  
    gap: 30px;  
}
```



```
.fullAfrica{  
    display: flex;  
    background-color: #444444a8;  
    backdrop-filter: blur(10px);  
    height: 85vh;  
    margin: 20px;  
    align-items: center;  
    justify-content: center;  
    z-index: 300;  
    padding-left: 5%;  
    border-radius: 20px;  
    transform: scale(0);  
    transition: 0.5s;  
}  
  
.fullAfrica img {  
    width: 40%;  
}  
.fullAfrica .mapBtn button{  
    display: flex;  
    position: absolute;  
    padding: 8px;  
    aspect-ratio: 1/1;  
    border-radius: 50%;  
    justify-content: center;  
    align-items: center;  
    border: none;  
    cursor: pointer;  
    transition: 0.5s;  
}  
  
.fullAfrica .mapBtn button:hover {  
    transform: scale(1.3);  
}
```

## Interactions avec JavaScript :

- Animation des boutons, effets de survol,
- Apparition des section,
- animation TreeJs et insertion,



```
function handleArrowBack() {
    const btn = elements.nextButtons[0];
    if (btn && elements.aboutUsWindow) {
        btn.style.display = "flex";
        setTimeout(() => {
            btn.style.opacity = "1";
            btn.style.right = "50%";
        }, 100);
        elements.aboutUsWindow.style.left = "-100%";
    }
}

function handleGatewayOpen() {
    if (elements.titleDiv) {
        elements.titleDiv.classList.add("scale-down");
    }
    setTimeout(() => {
        if (elements.gatewayWindow) {
            elements.gatewayWindow.classList.add("scale-up");
        }
    }, 500);
}
```



```
function toggleCaseActivity(caseElement) {
    if (caseElement) {
        caseElement.classList.toggle("active");
    }
}

if (elements.africa.btn1) {
    elements.africa.btn1.addEventListener("click", () => toggleCaseActivity(elements.africa.caseSenegal));
}
if (elements.africa.btn2) {
    elements.africa.btn2.addEventListener("click", () => toggleCaseActivity(elements.africa.caseNamibia));
}
```



```
// 2. Ajout de la 2eme lumiere
const light2 = new THREE.PointLight(0xffffffff, 10000);
light2.position.set(-130, 0, 30);
light2.castShadow = true;
scene.add(light2);

const loader = new GLTFLoader();
// Load model
let model;

loader.load('./models/earth/scene.gltf', function (gltf) {
    model = gltf.scene;
    model.scale.set(15, 15, 15);
    model.position.set(0, 0, 0);
    scene.add(model);
}, undefined, function (error) {
    console.error(error);
});
```

## Rendu final :

- Avex la commande ( npx vite preview )



# Welcome To PlanetCode

About Us

## 2. Précisez les moyens utilisés :

### Outils de développement :

- **Visual Studio Code** : éditeur de code utilisé pour écrire le HTML, le CSS, le JavaScript, et intégrer Three.js.
- **Navigateur web (Firefox)** : utilisé pour tester l'affichage du site, vérifier le responsive design et ouvrir la console de développement pour le débogage.

### Technologies utilisées :

- **HTML5** : structure des pages web.
- **CSS3** : mise en page et stylisation de l'interface, avec animations CSS simples.
- **JavaScript (vanilla)** : gestion de l'interactivité (menus dynamiques, chargement de contenu, événements utilisateurs).
- **Three.js** : création et gestion d'un **globe terrestre interactif en 3D**, ajout de lumières, de textures et d'effets de mouvement.

## 3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé **en autonomie**, sur mon **temps personnel**, avec mon ordinateur personnel, Visual Studio Code comme éditeur, et de nombreuses ressources en ligne

#### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
Chantier, atelier, service▶	
Période d'exercice ▶ Du : <b>30/04/2025</b> au : <b>05/05/2025</b>	

#### 5. Informations complémentaires (facultatif)

Repo github du project : <https://github.com/este0207/PlanetCode>

<b>Activité-type *</b>	ACT 2
<b>CP *</b>	Mettre en place une base de données relationnelle

Introduction :

Dans le cadre de cette compétence, j'ai développé un projet appelé **MiniShop**, une boutique en ligne de chaussures inspirée du style **Nike**.

Ce projet m'a permis de **concevoir et mettre en place une base de données relationnelle** avec **MariaDB**, en lien avec un back-end Node.js et une interface front Angular.

**1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :**

**Base de données (MariaDB) :**

- Mise en place d'un conteneur **MariaDB via Docker**.
- Crédation d'un **schéma relationnel** comprenant trois tables principales :
  - users : stockage des informations clients (nom, email, mot de passe...),
  - products : catalogue de chaussures (nom, prix, image, stock...),
  - user\_cart : panier d'achat lié à chaque utilisateur (produits ajoutés, quantités, total...).
- Définition des **relations entre les tables** (clés primaires/étrangères).
- Rédaction de **requêtes SQL** pour créer les tables et insérer des données de test.



```
MariaDB [shop]> SHOW TABLES;
```

Tables_in_shop
cart
cart_items
product
user

#### Back-end (Node.js + Express) :

- Création d'un **serveur Node.js** avec **Express** pour gérer les routes de l'application.
- Connexion du serveur à la base de données MariaDB.
- Mise en place d'**API REST** pour :
  - Récupérer la liste des produits (nom, Image, description)
  - Gérer les utilisateurs (création, authentification...),
  - Gérer les paniers (ajout, suppression, total...).
- Création d'un système de **stockage des images** côté serveur (chaussures, logos...).



```
async function main() {
    console.log("Démarrage du serveur sur " + host);
    const client = await mysql.createConnection({
        host : host,
        user : "root",
        password : "root",
        database: "shop"
    }).catch(console.error);
    return go(f, seed, [])
}
```



```
server.get("/products",async(req,res)=>{
    const products = await productModel.getAllProducts();
    res.json(products)
    console.log(products)
})

server.get("/product/:id",async(req,res)=>{
    const productId = req.params.id;
    const product = await productModel.getProductById(productId);
    res.json(product);
    console.log("Show product : "+productId);
    console.log(product);
})
```

```
// ----- Server listen ----- //
server.listen(PORT, () => {
    console.log(`Serveur démarré sur http://${host}:${PORT}`);
    console.log("Endpoints disponibles:");
    console.log("- POST /register");
    console.log("- POST /login");
    console.log("- GET /users");
});
```

## Front-end (Angular) :

Développement de l'interface utilisateur avec **Angular** :

- Page d'accueil avec affichage dynamique des produits,
- Pages produit, panier, formulaire d'inscription/connexion,
- Intégration du design responsive avec HTML/CSS.

Appels API via le **service HTTP Angular** pour interagir avec le back-end.

```
<app-startscreen></app-startscreen>
<app-navbar></app-navbar>
<app-form></app-form>
<app-home></app-home>
<app-card-container></app-card-container>
<app-kart></app-kart>
```

```
● ● ●  
  
<div class="nav">  
    <app-links logo="NIKE"></app-links>  
    <div class="login">  
        <a href="#" (click)="showKart()"><i class="fa-solid fa-cart-shopping"></i></a>  
        @if (!currentUser) {  
            <button (click)="showForm()">Login</button>  
        } @else {  
            <div class="user-info">  
                <span>{{ currentUser.name }} {{ currentUser.lastname }}</span>  
                <button (click)="logout()">Déconnexion</button>  
            </div>  
        }  
    </div>  
</div>
```

```
● ● ●  
  
<div class="home">  
    <div class="text">  
        <h1>JUST DO IT</h1>  
        <p (click)="scrollTo()"><i class="fa-solid fa-caret-down"></i></p>  
    </div>  
</div>
```

## 2. Précisez les moyens utilisés :

Le projet a été réalisé sur mon **ordinateur personnel**, en environnement local, à l'aide de :

- **Visual Studio Code** pour tout le développement,
- **Docker** pour gérer la base MariaDB,
- **Postman** pour tester les routes API,
- **Terminal** pour exécuter les scripts,
- **Navigateur web** pour les tests finaux de l'interface.

## 3. Avec qui avez-vous travaillé ?

Le projet a été réalisé **en autonomie**.

#### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
---	-------------------------

Chantier, atelier, service▶	
-----------------------------	--

Période d'exercice ▶ Du : **20/05/2025** au :**26/05/2025**

#### 5. Informations complémentaires (facultatif)

<b>Activité-type *</b>	ACT 2
<b>CP *</b>	Développer des composants d'accès aux données SQL et NoSQL

Dans le cadre de mon projet professionnel **MiniShop**, une boutique en ligne de chaussures, j'ai développé les **composants d'accès aux données** pour communiquer efficacement avec une base de données **SQL (MariaDB)**.

L'objectif était de créer un système permettant de **lire, insérer, modifier et supprimer** des données (CRUD) liées aux produits, utilisateurs et paniers.

#### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Tâches réalisées côté SQL (MariaDB) :

- **Création de la base et des tables relationnelles :**

- users : pour stocker les informations clients,
- products : pour le catalogue de chaussures,
- user\_cart : pour la gestion des paniers.

Ces tables ont été créées avec **clés primaires et étrangères** pour assurer la cohérence des données.



```
export class ProductModel{
    constructor(clientSQL){
        this.client = clientSQL;
        this.client.execute(`CREATE TABLE IF NOT EXISTS product(
            id INT PRIMARY KEY AUTO_INCREMENT,
            name TINYTEXT,
            price FLOAT,
            img TINYTEXT,
            cart BOOL
        )`).catch(console.error);
    }
}
```



```
await this.client.execute(`  
    CREATE TABLE IF NOT EXISTS user (  
        id INT PRIMARY KEY AUTO_INCREMENT,  
        name TINYTEXT NOT NULL,  
        lastname TINYTEXT NOT NULL,  
        email TEXT UNIQUE NOT NULL,  
        password TEXT NOT NULL  
    )  
`);
```

## Écriture de requêtes SQL personnalisées :

- Sélection de produits par id ou par nom,
- Sélection de tous les users ou le login
- Sélection de tous les produit pour affichage



```
async getAllUser(){  
    const [users] = await this.client.execute(`SELECT * FROM user`).catch(console.error);  
    return users;  
}
```



```
async getAllProducts(){  
    const [products] = await this.client.execute(`SELECT * FROM product`).catch(console.error);  
    return products;  
}  
  
async getProductById(id){  
    const [product] = await this.client.execute(`SELECT * FROM product WHERE id=?`,  
[id]).catch(console.error);  
    return product;  
}
```



```
async deleteUser(id){  
    const [users] = await this.client.execute(`DELETE FROM user WHERE id=?`, [id]).catch(console.error);  
    return users;  
}
```

## Création de routes :

```
server.get("/products",async(req,res)=>{
    const products = await productModel.getAllProducts();
    res.json(products)
    console.log(products)
})

server.get("/product/:id",async(req,res)=>{
    const productId = req.params.id;
    const product = await productModel.getProductById(productId);
    res.json(product);
    console.log("Show product : "+productId);
    console.log(product);
})
```



```
server.get("/users",async(req,res)=>{
    const users = await userModel.getAllUser();
    res.json(users)
    console.log(users)
})
```



```
server.delete("/delete-user/:id",async(req,res)=>{
    const userId = req.params.id;
    const users = await userModel.deleteUser(userId);
    // res.json(product);
    res.end("product id: "+userId+" are supp");
    console.log("product id: "+userId+" are supp")
})
```

## 2. Précisez les moyens utilisés :

Pour développer les composants d'accès aux données dans le projet **MiniShop**, j'ai utilisé plusieurs outils, technologies et ressources techniques :

### Technologies et outils utilisés :

- **MariaDB** : système de gestion de base de données relationnelle (SQL) déployé localement via **Docker**, utilisé pour stocker les utilisateurs, produits et paniers.
- **Docker** : pour créer un environnement de développement isolé et stable avec un conteneur MariaDB.

The screenshot shows the Docker Desktop application interface. On the left, there's a sidebar with navigation links: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers / frosty\_vaughan'. It displays a list of containers: 'frosty\_vaughan' (running, status 'Running (4 seconds ago)'), 'fd9cafe43bb3' (idle), and 'mariadb:latest' (idle). Below this is a table with tabs for Logs, Inspect, Bind mounts, Exec, Files, and Stats. The Logs tab is selected, showing a log output from a MariaDB container. The log entries are as follows:

```
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: 128 rollback segments in 3 undo tablespaces are active.
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the file full; Please wait ...
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: log sequence number 183751; transaction id 314
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] Plugin 'FEEDBACK' is disabled.
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] Plugin 'wsrep-provider' is disabled.
2025-07-21 09:19:59 2025-07-21 7:19:59 0 [Note] InnoDB: Buffer pool(s) load completed at 250721 7:19:59
2025-07-21 09:20:00 2025-07-21 7:20:00 0 [Note] Server socket created on IP: '0.0.0.0'.
2025-07-21 09:20:00 2025-07-21 7:20:00 0 [Note] Server socket created on IP: '::'.
2025-07-21 09:20:00 2025-07-21 7:20:00 0 [Note] mariadbd: Event Scheduler: Loaded 0 events
2025-07-21 09:20:00 2025-07-21 7:20:00 0 [Note] mariadbd: ready for connections.
2025-07-21 09:20:00 Version: '11.7.2-MariaDB-ubuntu2404' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary distribution
2025-07-21 09:19:57 2025-07-21 07:19:57+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.7.2+maria~ubu2404 started.
2025-07-21 09:19:58 2025-07-21 07:19:58+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2025-07-21 09:19:58 2025-07-21 07:19:58+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.7.2+maria~ubu2404 started.
2025-07-21 09:19:59 2025-07-21 07:19:59+00:00 [Note] [Entrypoint]: MariaDB upgrade not required
```

- **Node.js & Express** : back-end de l'application, utilisé pour créer une API REST et établir la communication avec la base de données via des composants personnalisés.
- **mysql2** : module Node.js permettant l'exécution sécurisée de requêtes SQL.
- **Postman** : pour tester les routes API et visualiser les données renvoyées par les composants SQL.
- **Visual Studio Code** : éditeur de code principal utilisé pour tout le développement (front, back, requêtes).
- **DBeaver** : pour visualiser, tester et manipuler graphiquement les données SQL.

DBeaver 25.1.3 - product

Database Navigator X Projects

Filter connections by name

shop 0.0.0:3306

Databases shop

- Tables cart (32K), cart\_items (32K), product (16K), user (32K)
- Views
- Indexes
- Procedures
- Packages
- Sequences
- Triggers
- Events
- sys
- Users
- Administrator
- System Info

Project - General X

Name | DataSource

- Bookmarks
- Dashboards
- Diagrams
- Scripts

Grid

id	name	price	img
1	DUNK LOW NEXT	120	1nike.jpg
2	DUNK LOW RETRO	78	nikebleu.webp
3	NIKE VAPORFLY 4	260	1nikerouge.jpg
4	DUNK LOW NEXT	120	1nikelow.jpg
5	AIR MAX PLUS	200	1TN.jpg
6	SB DUNK LOW	130	1vert.jpg

Value X

Record

Refresh Save Cancel Export data 200 6

6 row(s) fetched - 0.006s, on 2025-07-22 at 09:52:17

shop shop product

CET en

Home Workspaces Explore

Search Postman

Sign In Create Account

http://localhost:8090/images/Lil' Fella.png

Save

GET http://localhost:8090/products

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
-----	-------	-----------

Body Cookies Headers (8) Test Results

Status: 200 OK Time: 33 ms Size: 651 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": 1,
3     "name": "DUNK LOW NEXT",
4     "price": 120,
5     "img": "1nike.jpg"
6 },
7 {
8     "id": 2,
9     "name": "DUNK LOW RETRO",
10    "price": 78,
11    "img": "nikebleu.webp"
12 },
13 {
14     "id": 3,
15     "name": "NIKE VAPORFLY 4",
16     "price": 260,
17     "img": "1nikerouge.jpg"
18 },
19 {
20     "id": 4,
21     "name": "DUNK LOW NEXT",
22     "price": 120,
23     "img": "1nikelow.jpg"
24 }
```

Console Not connected to a Postman account

### 3. Avec qui avez-vous travaillé ?

Ce projet a été **réalisé seul**, dans un cadre professionnel.

J'ai travaillé **en autonomie** sur l'ensemble des composants, de la conception de la base de données à la mise en place du serveur Node.js et à la gestion des appels API.

### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
---	-------------------------

Chantier, atelier, service▶	
-----------------------------	--

Période d'exercice ▶ Du : <b>20/05/2025</b> au : <b>26/05/2025</b>
--

### 5. Informations complémentaires (facultatif)

<b>Activité-type *</b>	ACT 1
<b>CP 7</b>	► Développer des composants métier côté serveur

## 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de ce projet, j'ai développé les composants métier côté serveur pour l'application Post'hit, une boutique en ligne spécialisée dans la vente de posters personnalisables avec prévisualisation 3D. L'objectif était de gérer la logique métier de l'application, notamment la gestion des utilisateurs, des commandes et des interactions avec la base de données.

Les tâches réalisées incluent :

- **Création des routes API avec Node.js et Express :**

Développement d'endpoints pour gérer les fonctionnalités principales de l'application :

- Gestion des utilisateurs (inscription, connexion, profil),
- Gestion des commandes (création, modification, suppression),
- Gestion des interactions (ajout au panier, validation de commande).



```
import express from 'express';
import { getAllProducts,
    getProductById,
    addProduct
} from '../controllers/products.controller.js';

const router = express.Router();

router.get("/", getAllProducts);
router.post("/", addProduct);
router.get("/:id", getProductById);
router.put("/:id", getProductById);
router.delete("/:id", getProductById);

export default router;
```

- **Développement de la logique métier côté serveur :**

Mise en place de la logique pour :

- Validation des données reçues du front,
- Calcul des totaux des commandes,
- Gestion des erreurs et envoi de réponses appropriées au front-end.

```
● ● ●

import path from 'path';

export const serveImage = (req, res) => {
  const filename = req.params.filename;

  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'GET');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
  res.setHeader('Cache-Control', 'public, max-age=31536000');

  const options = {
    root: path.resolve('./public'),
  };

  res.sendFile(filename, options, (err) => {
    if (err) {
      if (!res.headersSent) {
        res.status(err.status || 500).send('Erreur lors de la récupération de l\'image');
      }
      return;
    }
  });
};
```

- **Interaction avec la base de données MariaDB :**

Connexion à la base de données pour effectuer des opérations CRUD sur les utilisateurs, les commandes et les produits.



```
import mysql from "mysql2/promise";
import dotenv from 'dotenv';
dotenv.config();

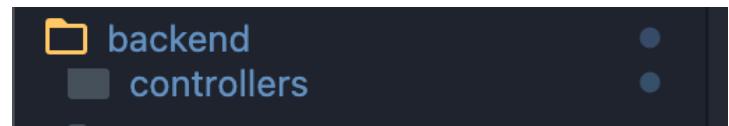
async function connectToDB() {
  try {
    const connection = await mysql.createConnection({
      host: process.env.DB_HOST,
      user: process.env.DB_USER,
      password: process.env.DB_PASS,
      database: process.env.DB_NAME,
    });
    console.log("Connexion à la base MySQL réussie");
    return connection;
  } catch (error) {
    console.error("Erreur de connexion à la base de données:", error);
    throw error;
  }
}

const client = await connectToDB();

export default client;
```

- **Gestion des fichiers et ressources statiques :**

Mise en place de la gestion des fichiers pour les images des produits, permettant leur stockage et leur récupération via le serveur.

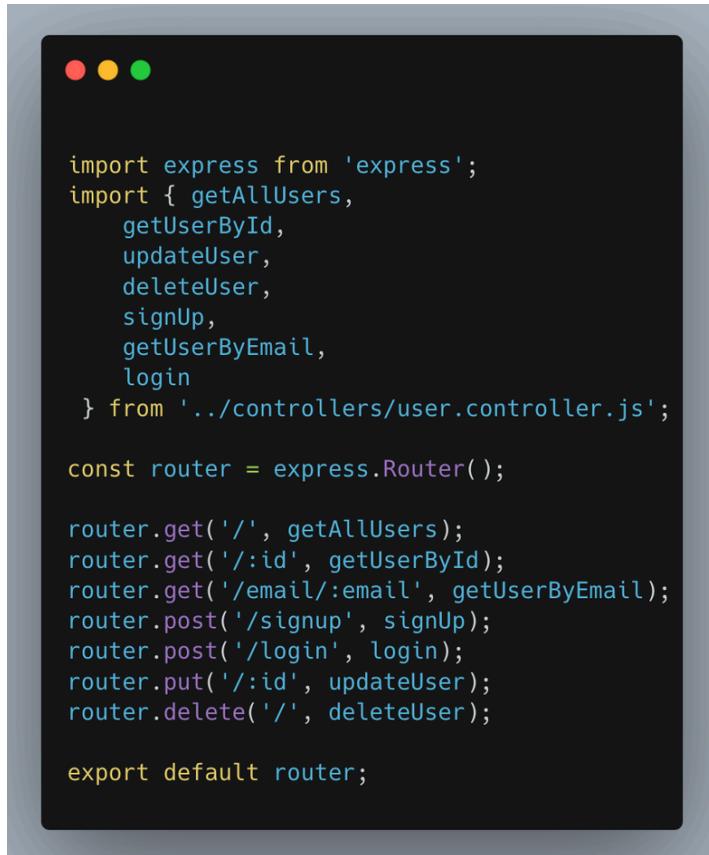


## 2. Précisez les moyens utilisés :

Pour développer les composants métier côté serveur de **Post'hit**, j'ai utilisé plusieurs outils, technologies et ressources :

### Outils et technologies :

- **Node.js & Express** : pour créer le serveur et gérer les routes API, permettant la communication entre le front-end Angular et la base de données.



```
import express from 'express';
import { getAllUsers,
        getUserId,
        updateUser,
        deleteUser,
        signUp,
        getUserByEmail,
        login
    } from '../controllers/user.controller.js';

const router = express.Router();

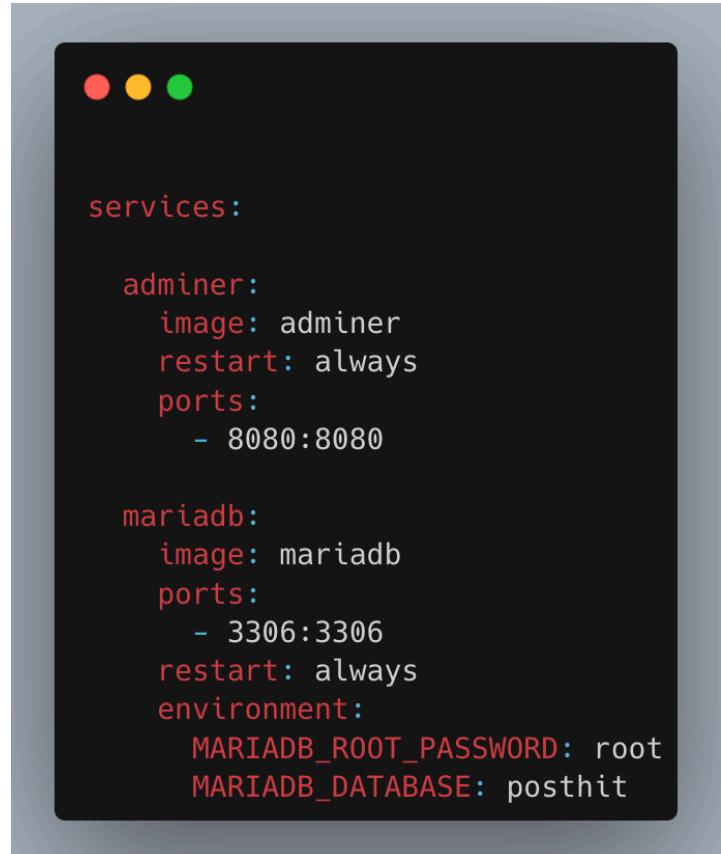
router.get('/', getAllUsers);
router.get('/:id', getUserId);
router.get('/email/:email', getUserByEmail);
router.post('/signup', signUp);
router.post('/login', login);
router.put('/:id', updateUser);
router.delete('/', deleteUser);

export default router;
```

- **MariaDB** : base de données relationnelle utilisée pour stocker les utilisateurs, produits et paniers.

Colonne	Type	Commentaire
<b>id</b>	int(11) <i>Incrément automatique</i>	
<b>product_name</b>	tinytext <i>NULL</i>	
<b>product_price</b>	double <i>NULL</i>	
<b>product_theme</b>	tinytext <i>NULL</i>	
<b>product_desc</b>	tinytext <i>NULL</i>	

- **Docker** : pour déployer la base MariaDB dans un environnement isolé et reproductible.

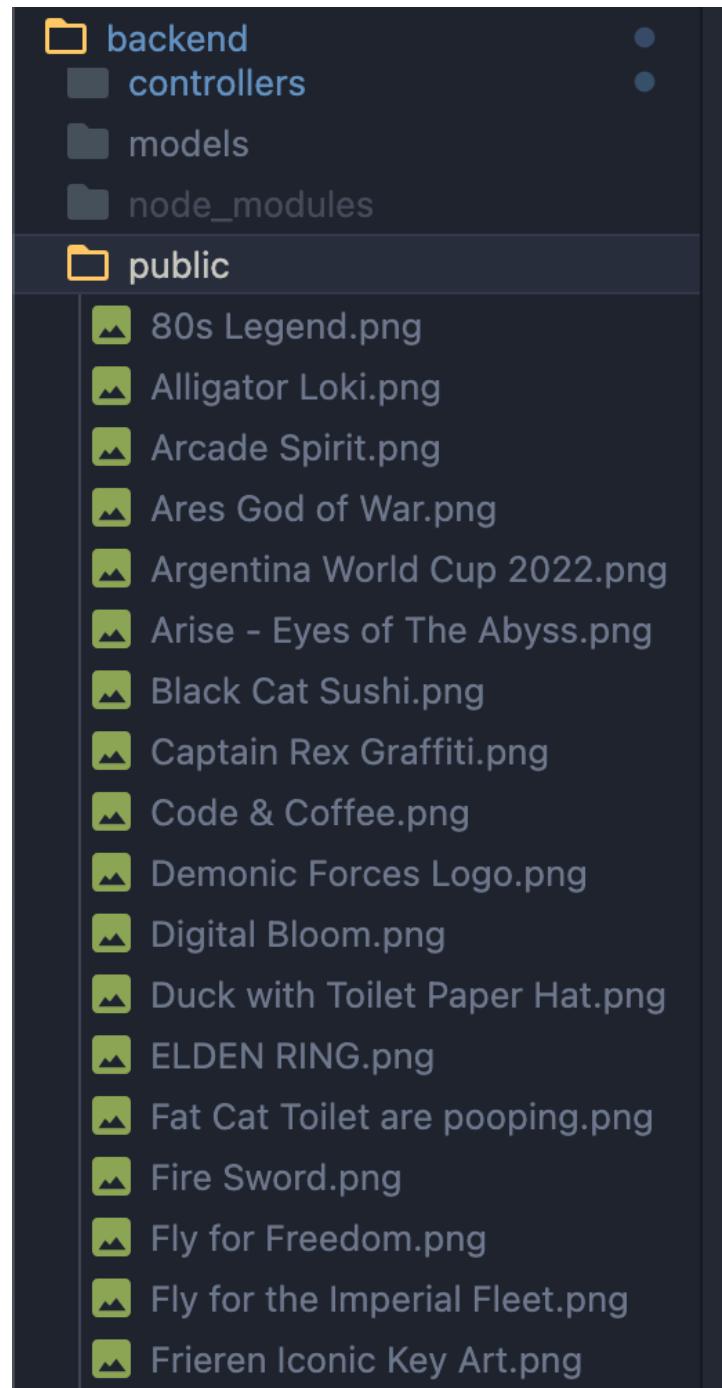


```
services:

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080

  mariadb:
    image: mariadb
    ports:
      - 3306:3306
    restart: always
    environment:
      MARIADB_ROOT_PASSWORD: root
      MARIADB_DATABASE: posthit
```

- **Visual Studio Code** : éditeur de code principal pour le développement du serveur et des composants métier.
- **Postman** : pour tester les endpoints API et vérifier le bon fonctionnement des composants côté serveur.
- **Dossier /public** : stockage des images des produits et utilisateurs, accessibles via des routes statiques du serveur.



### 3. Avec qui avez-vous travaillé ?

J'ai géré seul toutes les étapes du développement côté serveur : création du serveur Node.js, définition des routes API, développement de la logique métier, gestion des fichiers et interaction avec la base de données MariaDB.

#### 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
Chantier, atelier, service▶	
Période d'exercice ▶ Du : <b>20/05/2025</b> au : <b>10/07/2025</b>	

#### 5. Informations complémentaires (facultatif)

<b>Activité-type *</b>	ACT 2
<b>CP 8 ▶</b>	Documenter le déploiement d'une application dynamique

### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre du projet Post'hit, j'ai travaillé sur le déploiement de l'application dynamique, afin de rendre le site accessible en ligne avec un front-end et un back-end fonctionnels.

- **Déploiement du front-end sur Vercel :**

- Publication du projet Angular sur Vercel, accessible via l'adresse : [posthit.fr](http://posthit.fr)
- Configuration du domaine personnalisé et des variables d'environnement pour communiquer avec le back-end.
- Vérification que toutes les fonctionnalités dynamiques (affichage des posts, upload d'images) fonctionnent correctement avec le back-end distant.

- **Déploiement du back-end sur un VPS :**

- Installation de **Node.js** et configuration du serveur sur le VPS.
- Mise en place de **MariaDB** sur le VPS, avec les données nécessaires pour les utilisateurs, posts et interactions.
- Déploiement du serveur Node.js, gestion des routes API et liaison avec la base de données distante.

```

7 import { parseURLtoPath } from 'url';
8
9 // Import de la BDD
10 import db from './config/db.js';
11
12 // Import des routes
13 import productRoutes from './routes/products.routes.js';
14 import userRoutes from './routes/user.routes.js';
15 import cartRoutes from './routes/cart.route.js';
16 import categoryRoutes from './routes/categories.route.js';
17 import imageRoutes from './routes/image.routes.js';
18 import stripeRoutes from './routes/stripe.route.js';
19 import mailRoutes from './routes/mail.routes.js';
20 import bestsellingRoutes from './routes/bestselling.route.js';
21 import authRoutes from './routes/auth.routes.js';
22

```

Problèmes Sortie Console de débogage Terminal Ports 3 node - Posthit\_server + × ⌂ ⌂ ...

```

root@debian:~# ls
Posthit_server deploy_front.sh nginx-proxy
● root@debian:~# cd Posthit_server/
● root@debian:~/Posthit_server# ls
app.js controllers ecosystem.config.cjs init.sql node_modules package.json routes server.js
config docker-compose.yml import-csv.js models package-lock.json public save_BDD test-db.js
○ root@debian:~/Posthit_server# node server.js
Connexion à la base MySQL réussie
Server running on http://localhost:8090

```

## 2. Précisez les moyens utilisés :

**Vercel** : pour héberger le front-end Angular.

- Gestion du domaine personnalisé posthit.fr
- Configuration des variables d'environnement pour communiquer avec le back-end.

**VPS (Virtual Private Server)** : pour héberger le serveur Node.js et MariaDB.

- Installation et configuration de Node.js, MariaDB et des dépendances.
- Déploiement du serveur et liaison avec la base de données.

**MariaDB** : base de données relationnelle utilisée pour stocker utilisateurs, posts et interactions.

**Terminal SSH** : pour se connecter au VPS et déployer le serveur.

## 3. Avec qui avez-vous travaillé ?

Le déploiement a été réalisé **en autonomie**. J'ai géré seul le front-end sur Vercel et le back-end sur le VPS, ainsi que la base de données MariaDB.

## 4. Contexte

Nom de l'entreprise, organisme ou association ▶	La Plateforme Formation
Chantier, atelier, service▶	
Période d'exercice ▶ Du : <b>20/05/2025</b> au : <b>10/07/2025</b>	

## 5. Informations complémentaires (facultatif)