

# DOSSIER PROFESSIONNEL (DP)



*Nom de naissance*

 NANQUETTE

*Nom d'usage*



*Prénom*

 Olivia

*Adresse*

 31 rue Droite  
83670 Fox Amphoux

## Titre professionnel visé

DEVELOPPEUR WEB et WEB MOBILE

### MODALITÉ D'ACCÈS :

✓ Parcours de formation



## Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**






Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE. Il est consulté par le jury au moment de la session d'examen.

### Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

*[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]*

### Ce dossier comporte :

-  pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
-  un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
-  une déclaration sur l'honneur à compléter et à signer ;
-  des documents illustrant la pratique professionnelle du candidat (facultatif)
-  des annexes, si nécessaire.

*Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.*



<http://travail-emploi.gouv.fr/titres-professionnels>



## Sommaire

### Exemples de pratique professionnelle

#### Activité type 1 : Développer la partie front-end d'une application web ou web mobile sécurisée

p. 5

🔒 Installer et configurer son environnement de travail en fonction du projet web ou web mobile.

p. 5

🔒 Maquetter des interfaces utilisateur web ou web mobile.

p. 10

🔒 Réaliser des interfaces utilisateur statiques web ou web mobile.

p. 16

🔒 Développer la partie dynamique des interfaces utilisateur web ou web mobile

p. 20

#### Activité type 2 : Développer la partie back-end d'une application web ou web mobile sécurisée

p. 30

🔒 Mettre en place une base de données relationnelle.

p. 30

🔒 Développer des composants d'accès aux données SQL et NoSQL.

p. 41

🔒 Développer des composants métier coté serveur.

p. 41

🔒 Documenter le déploiement d'une application dynamique web ou web mobile.

p. 46

#### Titres, diplômes, CQP, attestations de formation *(facultatif)*

p. \_\_\_\_\_

#### Déclaration sur l'honneur

p. 52

#### Documents illustrant la pratique professionnelle *(facultatif)*

p. \_\_\_\_\_

#### Annexes *(Si le RC le prévoit)*

p. \_\_\_\_\_

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

## Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°1

Installer et configurer son environnement de travail en fonction du projet web ou web mobile.

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

Dans ce rapport, j'expliquerai l'installation et la configuration de mon environnement de travail pour un nouveau projet web ou web mobile en admettant que je démarre sur une nouvelle machine selon les consignes données.

#### 1. Installation et configuration de Visual Studio Code

La première étape est d'installer un IDE (éditeur de code ou environnement de développement intégré), nous installerons Visual Studio Code (VSC) qui est un IDE multiplateforme largement utilisé pour sa fluidité et sa rapidité, doté d'un écosystème riche en extensions qui s'adapte à n'importe quel web stack (front ou back end).

Parmi toutes les extensions/plugins existantes, nous installerons les suivantes en allant sur notre VSC et l'onglet plugins. Il suffit de taper le nom de l'extension recherchée et cliquer sur installer:

- GitLens, qui est une version de Git qui intègre les fonctionnalités de GIT directement dans mon IDE. Il gère les différentes versions du code.
- Prettier : Ce plugin permet de mettre en forme notre code, par exemple au niveau de l'indentation. Il applique un style cohérent au code.
- ESLint: Ce plugin analyse le code pour proposer des corrections si besoin, on appelle cela le linting. Après son installation, nous devons l'initialiser sur notre projet en ouvrant le terminal intégré et en tapant la commande `npm install eslint --save-dev`

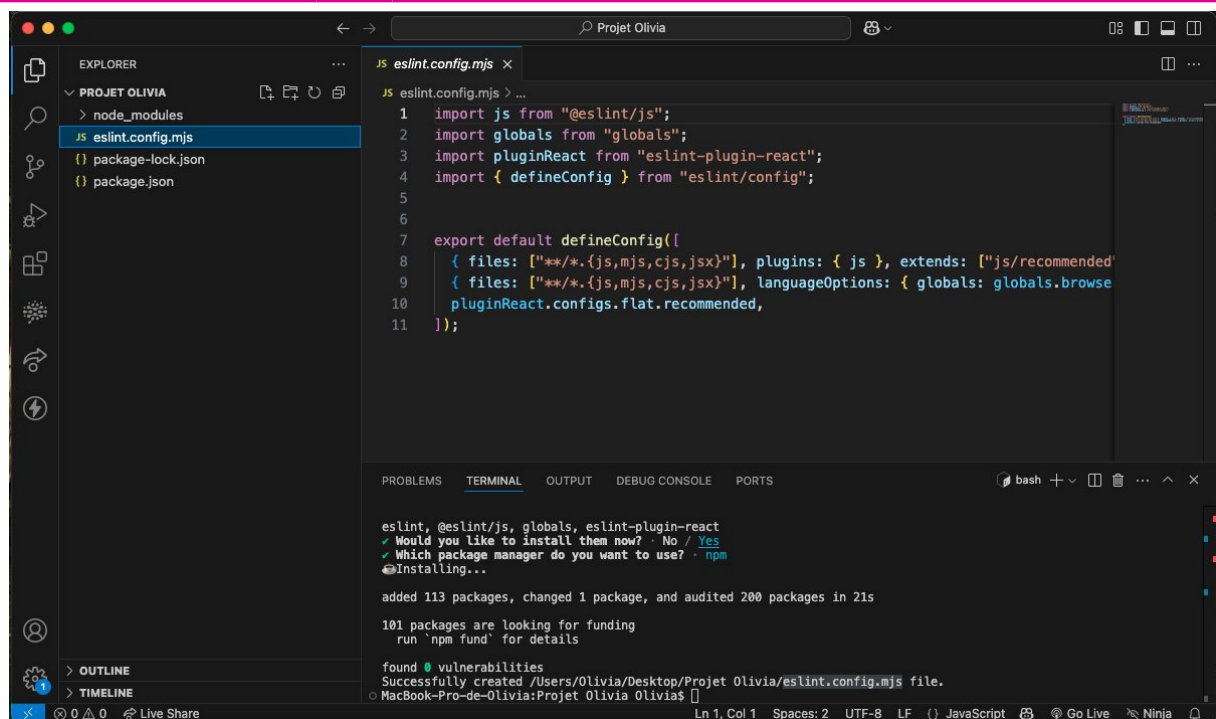
Ensuite nous devons générer le fichier de configuration avec la commande `npx eslint --init`

Pour ce projet, je choisis comme type de projet JavaScript, comme marche à suivre je choisis d'utiliser ESLint pour vérifier la syntaxe de mon code et trouver les problèmes. Puis je sélectionne le module JavaScript (import/export), et enfin j'opte pour le framework React, même si je ne vais pas l'utiliser dans ce projet.

Mon projet n'utilisera pas TypeScript, et sera visible sur mon navigateur.

Suite aux informations données ci-dessus, le programme me propose d'installer d'autres dépendances (eslint, @eslint/js, globals, eslint-plugin-react) sous le package manager npm

Mon projet contient à présent les fichiers suivants:



- J'installe ensuite PHP Intelephense qui est doté de plusieurs fonctionnalités essentielles pour un développement PHP productif (autocomplétion, détection d'erreurs, aide sur les fonctions et classes).

## 2. Configuration de GIT

Ensuite, nous allons installer GIT sur la machine via <https://git-scm.com/>.

Cela me propose la dernière version pour Mac 2.49.0

Git me permettra d'utiliser GitHub pour tous mes projets à venir et de sauvegarder chaque version du projet facilement.

GitHub quant à lui est une immense bibliothèque en ligne qui nous permet d'y déposer nos projets (appelés repository).

- Lors de la première installation de Git sur une machine, nous devons configurer notre identité avec les commandes :

```
git config --global user.name "Votre Nom"
```

```
git config --global user.email "votre.email@exemple.com"
```

- Ensuite, ayant déjà fait cela pour d'autres projets, je peux simplement initialiser ce nouveau projet dans Git avec la commande `git init`

J'en profite pour créer un fichier `.gitignore` dans lequel je pourrai noter le nom de chaque fichier ou dossier (suivi d'un `/`) que je ne souhaite pas faire apparaître dans mon repository GitHub, comme par exemple les fichiers inutiles ou les fichiers contenant des informations sensibles.

- A présent, je me rend sur mon compte sur <https://github.com/> pour y créer un nouveau repository que j'appellerai: Evaluation 1

- J'ajoute ensuite les fichiers créés avec la commande `git add` .
- Je fais ensuite mon premier message de commit, qui permet d'expliquer ce que j'ai fait avec `git commit -m "Add new project for evaluation"`
- Je sélectionne ensuite la branche principale de mon projet avec `git branch -M main`
- Ensuite je connecte mon repository local à mon repository distant que je viens de créer sur GitHub avec `git remote add origin https://github.com/Zitoone/Evaluation-1.git`
- Enfin j'envoie tout avec `git push -u origin main`

### 3. Mise en place d'un serveur local

Dans cette prochaine étape, nous allons installer un serveur local sur la machine. Nous avons plusieurs choix possibles : **XAMPP**, **WAMP**, **MAMP**. Pour un MAC je dois utiliser MAMP. Ce serveur permettra de visualiser l'avancement d'un site dynamique pendant sa construction avant de le mettre en ligne en lisant les fichiers HTML, PHP et gérer les bases de données MySQL. Ce serveur s'appelle APACHE.

Une fois MAMP installé, il me suffit de l'activer. Je pourrai l'activer avant chaque projet selon mes besoins.

Je crée ensuite le dossier de mon projet dans le dossier racine HTDOCS qui se trouve dans le dossier MAMP de ma machine. Je l'appelle: mon projet php

Pour y accéder j'utilise l'url suivant : <http://localhost:8888/mon-projet-php/>

### 4. Préparation d'un projet

4.1. Une fois mon environnement de travail installé complètement, je peux commencer à préparer mes projets. Nous allons commencer par la préparation d'un projet **Node.js**

JavaScript est un langage qui s'exécute côté client c'est-à-dire sur le navigateur.

Node js permet de l'exécuter côté serveur.

Après avoir téléchargé la version LTS qui est réputée pour sa stabilité sur <https://nodejs.org/> je vérifie son installation sur le terminal VSC de mon projet grâce à la commande `npm -v`

Je dispose actuellement de la version 10.8.2

Je crée ensuite un projet avec la commande `mkdir mon-projet-node`

Puis je remonte à la racine de ce projet avec la commande `cd mon-projet-node`

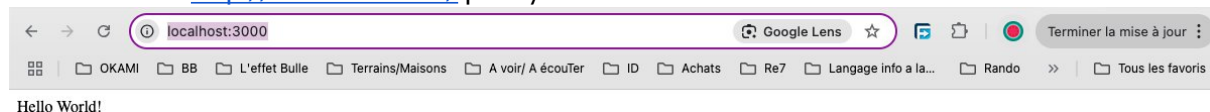
J'initialise et installe les packages nécessaires avec les commandes `npm init` et `npm install`

Je crée ensuite un fichier index.js dans ce nouveau dossier dans lequel je note le code suivant qui permet de lancer un serveur web Express lorsqu'on accède à l'URL racine :

```
const express = require('express');
const app = express();
app.get('/', (req, res) => res.send('Hello World!'));
app.listen(3000, () => console.log('Server running on port 3000'));
```

Je lance ensuite ce code pour que cela soit visible sur le port 3000 avec la commande `node index.js`

Je me rend sur <http://localhost:3000/> pour y voir le résultat suivant:

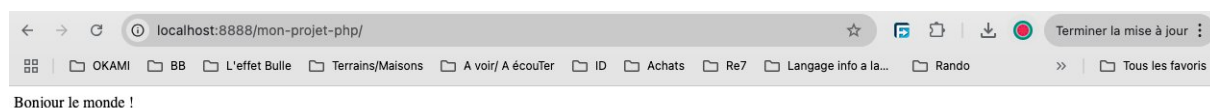


## 4.2. Nous allons à présent préparer le projet PHP

Je reprends mon dossier appelé “mon projet php” créé dans le dossier HTDOCS disponible dans le dossier MAMP. J'y crée un fichier index.php avec le code suivant:

```
<?php
echo "Bonjour le monde !";
?>
```

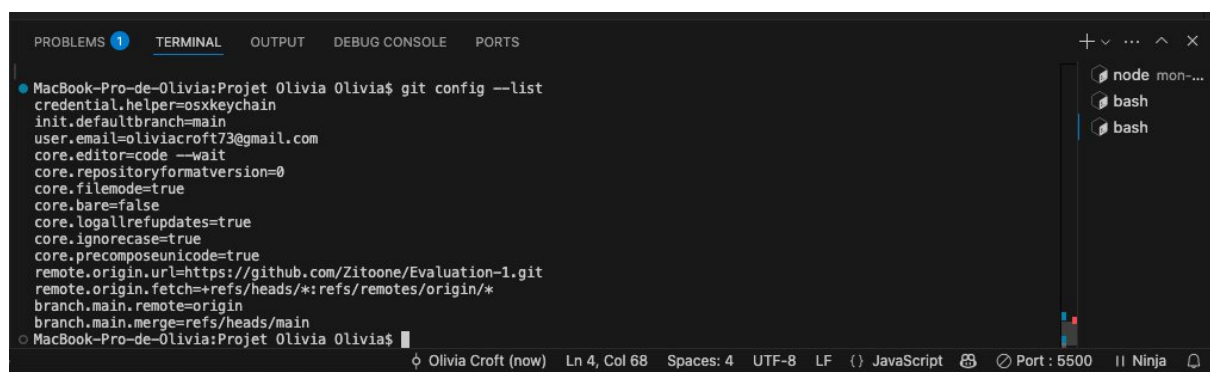
En retournant sur le localhost8888 suivi d'un / et le nom de mon dossier voici le résultat final:



## 5. Validation finale

A la suite de toutes les actions précédemment décrites je peux donc confirmer que:

- GIT est bien configuré :



Voici le lien de ce projet sur GitHub: <https://github.com/Zitoone/Evaluation-1.git>

- Le projet Node js accessible avec le port localhost:3000:

localhost:3000

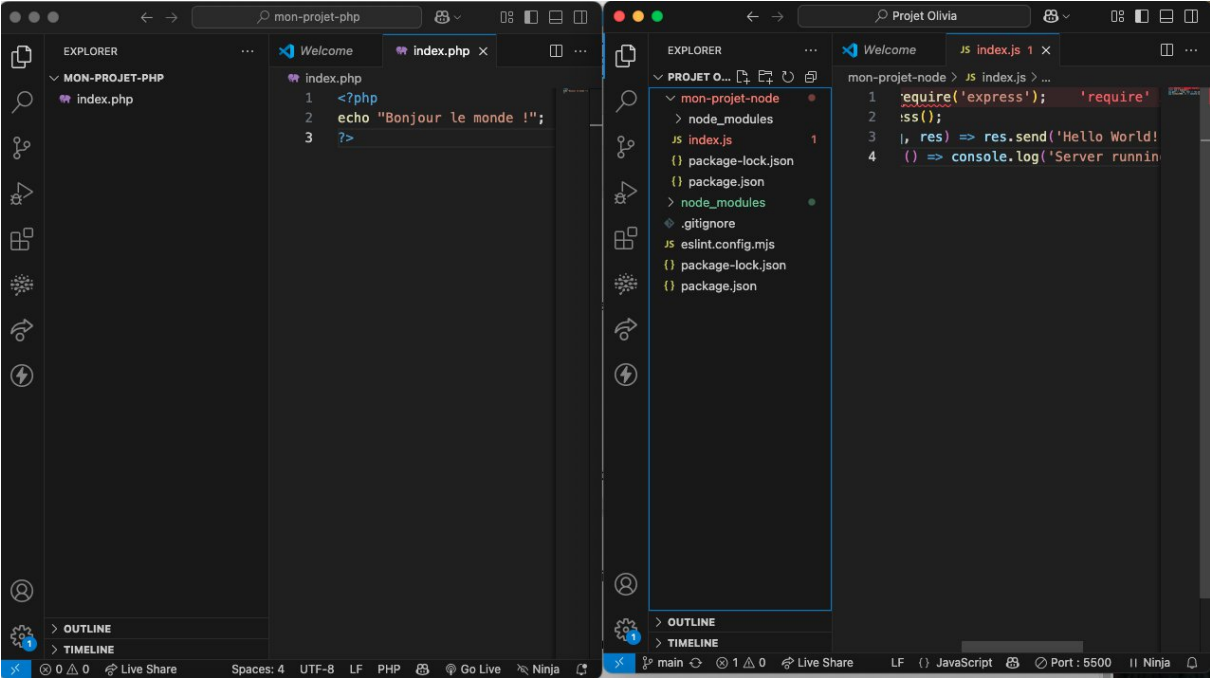
Hello World!

- Le projet PHP accessible via <http://localhost:8888/mon-projet-php/>:

localhost:8888/mon-projet-php/

Bonjour le monde !

- Ces 2 projets sont ouverts avec VSC



## Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°2

Maquetter des interfaces utilisateur web ou web mobile.

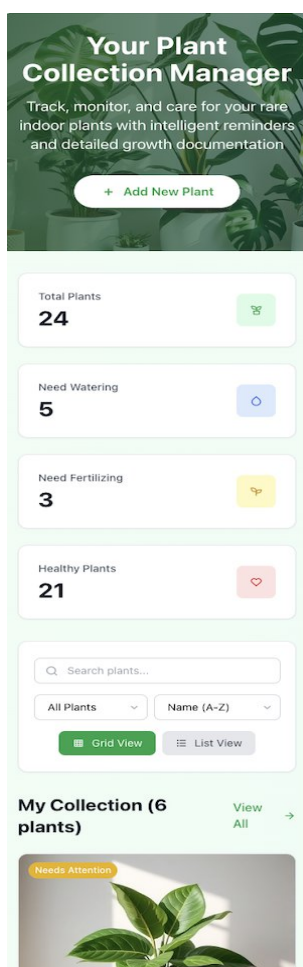
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour cette évaluation, nous devons concevoir les maquettes d'une application web ou web mobile destinée à la gestion et au suivi de collections de plantes d'intérieur rares, j'ai choisi de m'aider de 2 outils IA en ligne pour l'orientation du design [readdy.ai](https://readdy.ai) et [stitch.withgoogle.com](https://stitch.withgoogle.com)

Les 2 résultats m'ont convaincu et j'ai décidé d'allier les 2 propositions de design.

Comme dans les consignes nous avons le choix de concevoir une application web ou web mobile, j'ai choisi de créer la version web mobile qui sera sûrement plus utile à utiliser pour les utilisateurs (pour partager des photos de leurs plantes par exemple).

Je me suis calquée sur la page d'accueil proposé par readdy (ci-dessous)



J'y ai ajouté un header qui sera la barre persistante de toutes les pages, avec le logo qui permettra toujours de revenir sur la page d'accueil, un icône de notification et un icône de compte utilisateur. Le thème étant les plantes, j'ai pensé (comme readdy d'ailleurs) que la couleur dominante serait le vert.

Avec l'aide du site <https://paletadecolores.online/> j'ai recherché des déclinaisons de vert et de couleurs associées qui permettrait de disposer d'une palette de couleur harmonieuse sur l'ensemble de l'application.

Pour les autres pages, je me suis inspiré de Stitch avec un design plus épuré mais des sections et des pages

bien organisées.

Au fil de la conception des maquettes je me suis rendue compte que les couleurs trop orangées n'étaient pas idéales en couleurs de fond, j'en ai donc choisi une plus adéquate pour que l'application ait une ambiance chaleureuse, naturelle et douce.

J'ai d'abord décidé de mettre un header avec le logo et 2 icônes en haut de l'application, mais je me suis rendue compte bien plus tard que pour une application mobile il était préférable de mettre une barre fixe en bas de l'écran, ce qui est bien plus intuitif.

Ensuite j'ai construit chaque page selon les user stories présents dans les consignes avec l'idée que l'utilisateur est déjà passé par la phase de connexion à son compte personnel ou inscription.

#### *US1 : Créer une fiche plante*

- *En tant que collectionneur, je veux pouvoir ajouter une nouvelle plante à ma collection avec son nom, espèce, date d'acquisition et photo, afin de la cataloguer facilement.*

Sur la page d'accueil on trouve tout de suite le bouton ADD PLANT qui emmène sur la page de formulaire d'ajout d'une plante ou on peut ajouter le nom de la plante, l'espèce, la date d'acquisition et des photos. Une fois les informations remplies correctement, et après avoir appuyer sur SAVE PLANT on a un message de confirmation.

#### *US2 : Consulter le détail d'une plante*

- *En tant que collectionneur, je veux pouvoir consulter la fiche détaillée d'une plante (historique d'arrosage, de fertilisation, notes de croissance), afin de suivre son évolution dans le temps.*

Il y a plusieurs façons d'accéder à la fiche détaillée d'une plante, on peut par exemple sur la page d'accueil aller sur TOTAL PLANTS qui nous emmène sur la page de collection de toutes les plantes de l'utilisateur, et on choisit la plante que l'on veut pour avoir ses informations détaillées (historique d'arrosage, de fertilisation etc...)

#### *US3 : Modifier ou supprimer une plante de ma collection*

- *En tant que collectionneur, je veux pouvoir mettre à jour ou supprimer les informations d'une plante, afin de corriger ou retirer des plantes de ma collection.*

Toujours sur la fiche détaillée d'une plante, à la fin de la page on trouve 2 boutons d'actions: Btn EDIT pour modifier les informations de la plante ou DELETE pour supprimer complètement la plante.

Ici on peut également ajouter des photos de l'évolution de la plante; ce qui répond à l'US 6

#### *US4 : Recevoir des notifications d'entretien*

- *En tant que collectionneur, je veux recevoir des notifications personnalisées pour l'arrosage et la fertilisation, afin de ne pas oublier les soins de mes plantes.*

Sur n'importe quelle page, dans la barre des tâches, on retrouve l'icône habituelle de notifications et lorsqu'on va dessus on se retrouve sur la page des rappels. On y trouve les rappels à venir et les passés, et nous avons la possibilité d'en ajouter, ce qui nous amène la US suivante

#### US5 : Configurer les rappels d'entretien

- En tant que collectionneur, je veux pouvoir personnaliser la fréquence et l'heure des rappels d'entretien pour chaque plante, afin d'adapter les notifications à leurs besoins spécifiques.

Une page de configuration des rappels d'entretien. On peut choisir sa plante, sélectionner le type d'entretien, la fréquence et enfin l'heure des rappels désirée.

#### US6 : Ajouter des photos d'évolution

- En tant que collectionneur, je veux pouvoir ajouter régulièrement des photos à la fiche d'une plante, afin de documenter visuellement sa croissance au fil du temps.

Voir US 3

#### US7 : Filtrer ma collection de plantes

- En tant que collectionneur, je veux pouvoir filtrer les plantes par type, besoin en lumière ou date d'acquisition, afin de mieux organiser ma collection.

#### US8 : Trier ma collection de plantes

- En tant que collectionneur, je veux pouvoir trier mes plantes par ordre alphabétique, date d'acquisition ou dernier entretien, afin de naviguer facilement dans ma collection.

Sur la collection de plantes, tout en haut, on trouve un moyen de filtrer ou trier les plantes comme expliqué dans les US 7 et 8

#### US9 : Visualiser un historique global d'entretien

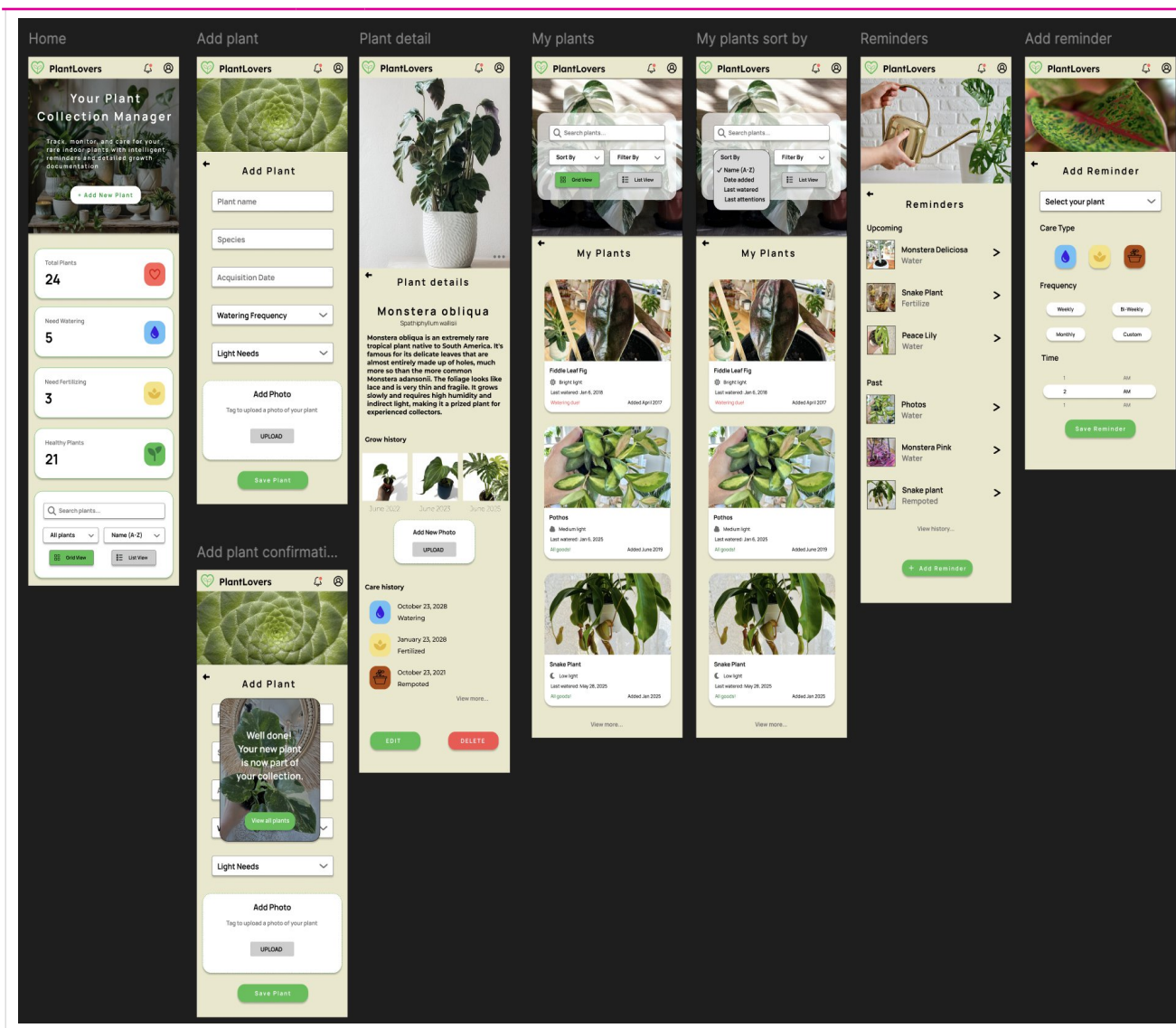
- En tant que collectionneur, je veux accéder à une page qui liste l'historique global des entretiens (arrosage, fertilisation, rempotage) pour toutes mes plantes, afin de suivre l'évolution de mes actions sur l'ensemble de ma collection.

Revenons sur la page de tous les rappels ou souvenez vous on y trouve les prochains et les anciens. Sous les rappels passés il y a un petit lien VIEW HISTORY... qui emmènera sur une page qui liste tous les entretiens pour toutes les plantes.

Le prototype est disponible sur ce lien :

<https://www.figma.com/proto/povwvTAZ7SWrWfSeR2DqH6/TP-Figma---PlantLovers-mobile-version?node-id=16-729&p=f&t=v1xwnKFIIJ3g9K3f-1&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=16%3A729>

Voici l'aperçu de toutes les maquettes créées :



## 2. Précisez les moyens utilisés :

Pour réaliser ces maquettes j'ai donc utilisé la version en ligne de FIGMA et tous les outils disponibles pour la création des différentes maquettes qui représentent les différentes pages. Une fois toutes les maquettes terminées, elles ont été liées entre elles pour présenter un parcours utilisateur réel grâce à la fonction prototype de Figma.

### 3. Avec qui avez-vous travaillé ?

Cette évaluation a été réalisée en solo.

## Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°3  Développement d'un portfolio personnel statique

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

Pour cette évaluation, nous devons réaliser un site vitrine statique présentant notre profil professionnel, nos compétences, nos projets et nos coordonnées.

Le site doit être conçu uniquement avec HTML5 et CSS3, sans frameworks ni bibliothèques externes (pas de Bootstrap ni JavaScript).

Cette présentation est divisée en plusieurs objectifs qui étaient demandés dans les consignes.

## 1. Structuration d'une page web en utilisant les balises sémantiques de HTML5

Afin que ce portfolio soit structuré de manière correcte, j'ai construit son HTML de façon claire et visible. La page d'accueil étant la page la plus fournie, en plus du <header> et <footer>, est séparée en 4 sections distinctes. On y trouve une section « Introduction », une section « A propos », une section « Projets » et une « Contact ». Chaque section sera détaillée dans les autres pages qui correspondent à chacune des sections (sauf pour la section Introduction, qui est une brève présentation de la page d'accueil et du site en général)

Chaque section est composée de titrage <h2> pour la présenter, et parfois de sous titrages (<h3>, <h4>) si besoin.

On y trouve également souvent la balise <article> qui permet de regrouper des informations de même nature ensemble. Cette balise doit toujours être accompagnée d'un titrage, chose que je n'ai pas faite pour la partie « Compétences » dans la section « A propos ».

Voici un exemple de structure de la page HTML « Projets » :

```
projects.html M X
<? projects.html > <? html > <? body
2 <html lang="fr">
11 <body>
12 <header>
13 <div>
14 <a href="./index.html"><h1>Olivia Nanquette</h1></a>
15 </div>
16 <nav class="navbar">
17 <ul>
18 <li><a href="index.html">Accueil</a></li>
19 <li><a href="about.html">A propos</a></li>
20 <li><a href="projects.html">Projets</a></li>
21 <li><a href="contact.html">Contact</a></li>
22 </ul>
23 </nav>
24 </header>
25 You, now * Uncommitted changes
26 <main id="page-projets">
27 <div class="titre">
28 <h2>Mes projets</h2>
29 </div>
30
31 <section>
32 <p>Voici une sélection des travaux menés durant ma formation, illustrant mes compétences techniques et ma progression.</p>
33
34 <div class="articles-projets">
35 <article>
36 
37 <div>
38 <h3>Brigitte Galerie d'Art</h3>
39 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nemo ullam, magni recusandae laboriosam officia eve
40 <div>
41 <ul class="liste-compet">
42 <li>HTML</li>
43 <li>CSS</li>
44 </ul>
45 </div>
46 <a href="#">Voir le projet </a>
47 </div>
48 </article>
49
50 <article>
51 
52 <div>
53 <h3>Site Foncière</h3>
```

Toutes ces balises sémantiques permettent aux navigateurs d'interpréter et d'afficher la page en y renseignant la structure et donc également de la rendre lisible pour tout utilisateurs.

## 2. Styliser et rendre attractif un site avec CSS3

Après l'intégration des pages HTML, j'ai réalisé la feuille de style associée appelée style.css qui se trouve dans le dossier CSS. Pour ce projet je décide de réaliser une seule feuille de style.

J'y importe en premier lieu l'URL qui me permettra d'utiliser les polices de caractère choisies.

Puis je crée certaines variables qui me serviront tout au long de la création de la feuille de style, comme les

couleurs par exemple.

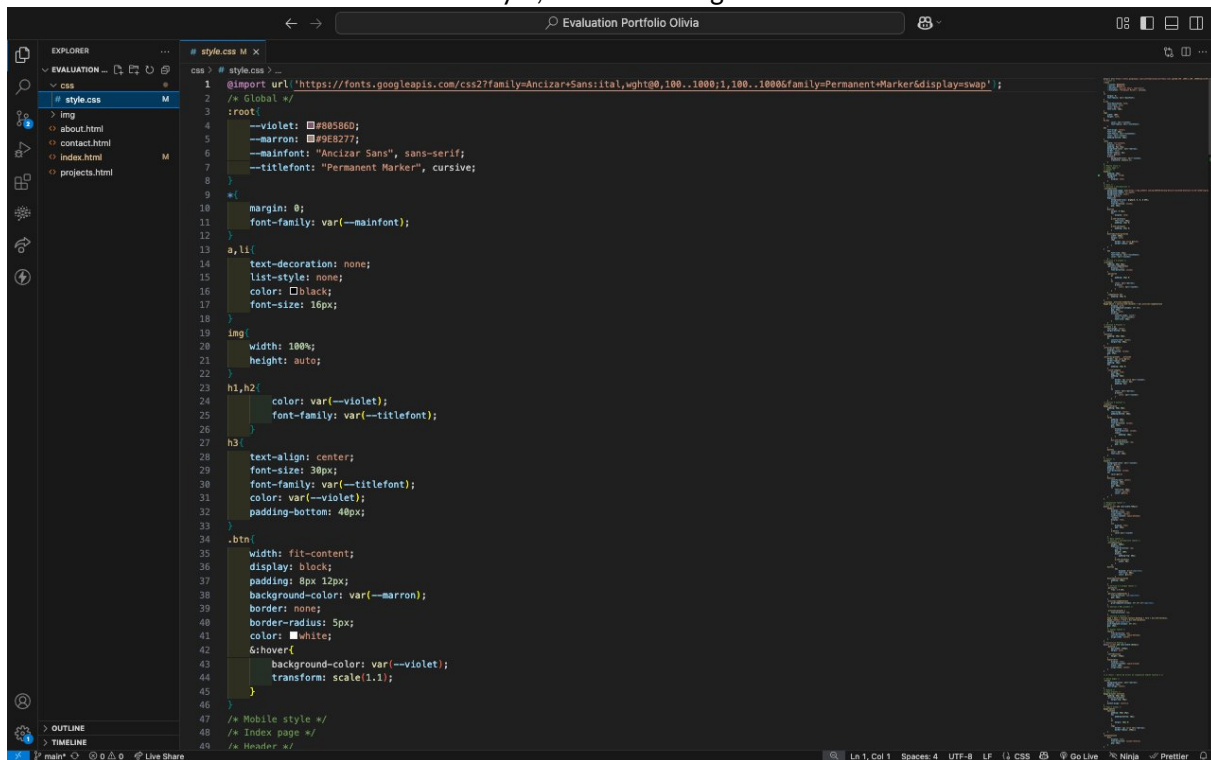
Ensuite j'y entre quelques propriétés globales que je souhaite appliquer à tout le portfolio grâce au caractère \* qui sélectionne absolument tous les éléments, comme par exemple un MARGIN à 0 pour pouvoir gérer moi même les marges des éléments.

Pour pouvoir positionner mes éléments mais aussi pour pouvoir les manipuler pour les différentes tailles d'écran, j'ai choisis principalement des FLEX BOX car cette méthode permet de distribuer l'espace entre les éléments et de les d'aligner facilement.

J'utiliserai tout de même la méthode GRID pour positionner mes différentes compétences plus facilement en contrôlant à la fois les lignes et les colonnes de cette partie.

J'ai utilisé le CSS pour mettre une image de fond sur ma première section car cela permet de bien distinguer le contenu et la présentation et permet de contrôler sa position, la taille, la répétition, le défilement.

Voici le début le début de ma feuille de style, elle fait 374 lignes :



```
1 @import url('https://fonts.googleapis.com/css2?family=Ancizar+Sans:ital,wght@0,100..1000;1,100..1000&family=Permanent+Marker&display=swap');
2 /* Global */
3 :root{
4   --violet: #80586D;
5   --maroon: #9E2777;
6   --mainfont: "Ancizar Sans", sans-serif;
7   --titlefont: "Permanent Marker", cursive;
8 }
9 *{
10  margin: 0;
11  font-family: var(--mainfont);
12 }
13 a,li{
14  text-decoration: none;
15  list-style: none;
16  color: black;
17  font-size: 16px;
18 }
19 img{
20  width: 100%;
21  height: auto;
22 }
23 h1,h2{
24  color: var(--violet);
25  font-family: var(--titlefont);
26 }
27 h3{
28  text-align: center;
29  font-size: 30px;
30  font-family: var(--titlefont);
31  color: var(--violet);
32  padding-bottom: 40px;
33 }
34 .btn{
35  width: fit-content;
36  display: block;
37  padding: 8px 12px;
38  background-color: var(--maroon);
39  border: none;
40  border-radius: 5px;
41  color: white;
42  &hover{
43    background-color: var(--violet);
44    transform: scale(1.1);
45  }
46 }
47 /* Mobile style */
48 /* Index page */
49 /* Header */
```

### 3. Concevoir un site responsive, accessible jusqu'à une largeur minimale de 320px

Pour ce projet j'ai décidé d'avoir une approche mobile first qui me semble une approche plus judicieuse car la plupart des sites Internet sont visités depuis un mobile de nos jours. J'ai donc commencé mon intégration à 320px puis une fois terminé j'y ai ajouté 2 breakpoints à l'aide de media query et un min width de 768px, puis 1025px pour la version ordinateur.

### 4. Intégrer des interactions visuelles au survol des éléments (effets hover)

J'ai ajouté quelques effets au survol de certains éléments comme sur le menu de navigation et les liens/boutons pour passer aux autres pages du portfolio.

### 6. Créer un repository Github, réaliser des commits réguliers et publier un site sur Github Pages

Afin de conserver les différentes versions de l'avancé de mon travail au fur et à mesure, je crée un nouveau repository avec lequel je ferai des commits régulier qui expliquent l'avancée de mon projet.

## 2. Précisez les moyens utilisés :

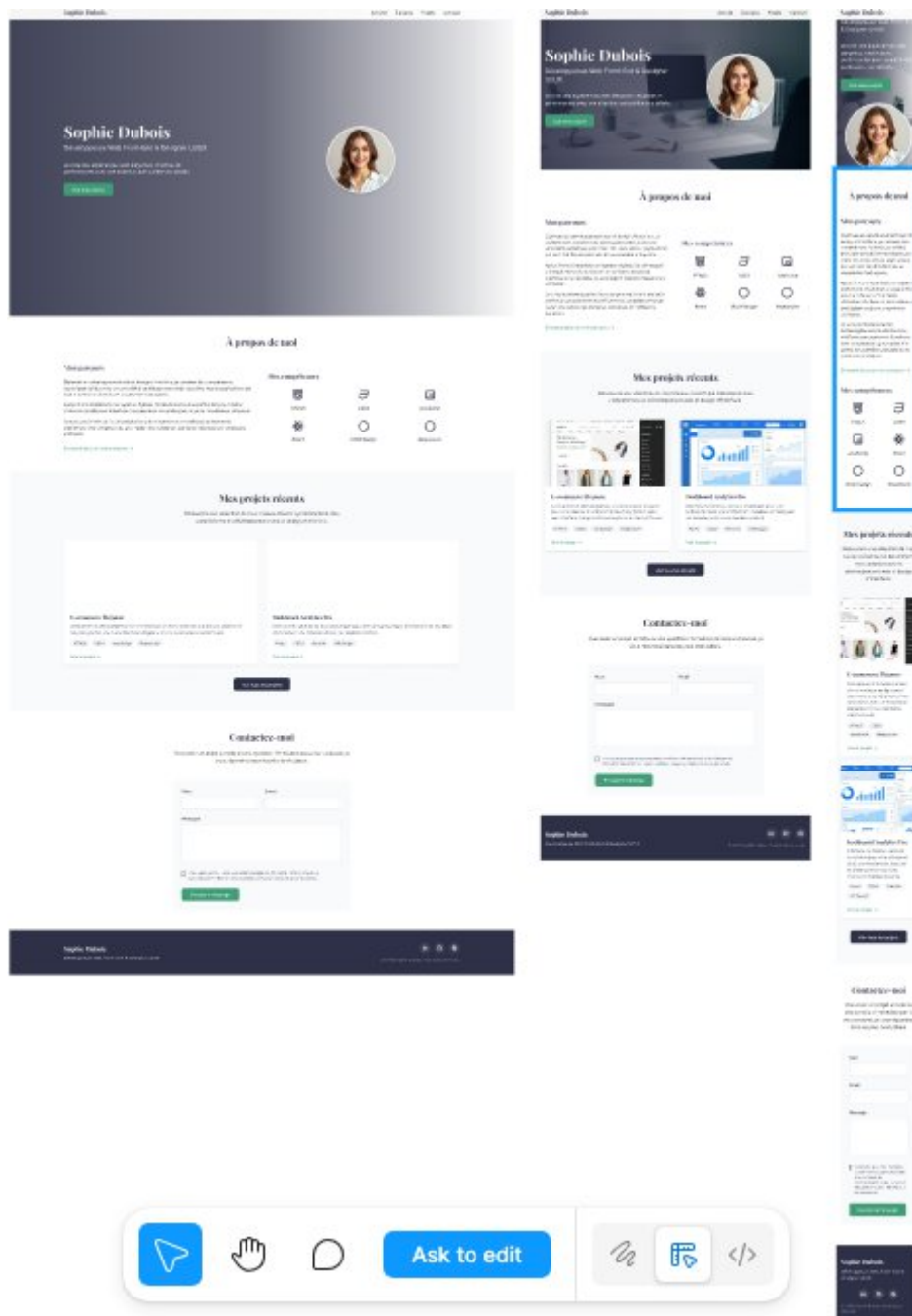
Pour réaliser ce portfolio j'ai utilisé HTML5 et CSS3.

Sur ce projet j'ai travaillé entièrement seule. Cette évaluation a été réalisé les 23 et 24 Juillet 2025, le 26 j'ai présenté à l'oral ce travail a mes camarades et mon professeur.

Lien du projet : <https://zitoone.github.io/Portfolio-Evaluation/>

Pour ce qui est du design, je me suis basée sur la maquette fournie pour l'évaluation en y modifiant les couleurs, les polices et quelques positions d'élèments.

Voici la maquette initiale :



## Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

## Exemple n°4 📁 Développer la partie dynamique des interfaces utilisateur web ou web mobile

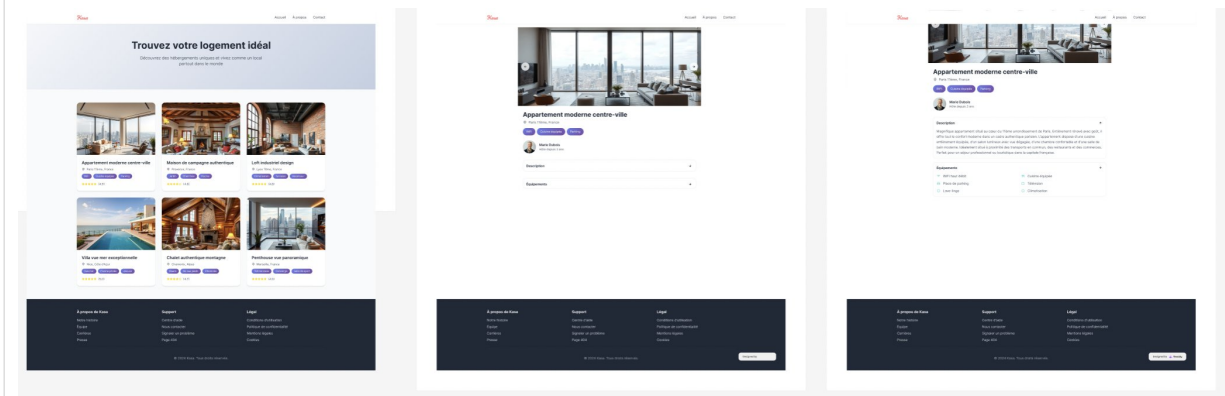
Nom du projet Kasa

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

Pour cette évaluation, nous devons reproduire un site de location d'hébergement type AirBnB simplifié avec React, Vite, TypeScript et Sass.

Pour réaliser ce projet nous avons certaines consignes à respecter comme la réalisation de 3 pages (Accueil, détails et erreur 404), mais aussi l'utilisation d'au moins 3 composants (Card, Carousel et Collapse).

Un design figma nous a été fournit :



### 2. Précisez les moyens utilisés :

Je commence tout d'abord par **initialiser mon projet** sur ma machine à l'aide des commandes suivantes:

```
npm create vite@latest
npm install
npm install -D sass-embedded
```

Avec la commande `npm run dev` je peux voir que mon projet est créé.

Puis je crée mon dossier `pages/` avec les 3 pages à créer pour ce projet:

— `pages/ # Home.tsx, Details.tsx, NotFound.tsx`

Je mets ensuite en place les routes pour créer une navigation entre les différentes pages du site en configurant d'abord le fichier `main.tsx`

```
import React from 'react' //Pour importer la bibliothèque React
import ReactDOM from 'react-dom/client' //Module pour interagir avec le DOM
import App from './App.tsx' //Importe le fichier App
import { BrowserRouter } from 'react-router-dom' //Bibliothèque pour gérer la navigation
entre les pages

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <BrowserRouter>
```

```

    <App />

  </BrowserRouter>

</React.StrictMode>,
)

```

Puis je termine en configurant les routes dans *app.tsx*

```

import {Routes, Route} from 'react-router-dom' // Pour la gestion de la navigation sur
l'appli
import Home from '../pages/Home'
import Details from '../pages/Details'
import NotFound from '../pages/NotFound'

function App() {
  return(
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/details" element={<Details />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
  )} export default App

```

Ensuite j'intègre le fichier *logements.json* fournit dans un nouveau dossier data (toujours dans le dossier *src*).

Ensuite nous allons créer nos composants réutilisables. Nous commençons par le composant *Header*. Pour cela nous créons le dossier *components* qui regroupera tous les composants de cette appli.

Dans le composant *Header*, j'importe les bibliothèques *Link* et *NavLink* dont je vais avoir besoin. J'installe aussi react icons avec la commande *npm install react-icons —save* qui me servira ici pour l'icône du burger menu en responsive et aussi plus tard dans le projet pour d'autres icônes.

Dans ce composant j'utilise un *useState* pour changer l'état du burger menu quand il sera ouvert ou fermé.

```

import {Link, NavLink} from 'react-router-dom'
import { CiMenuBurger } from "react-icons/ci";
import {useState} from "react";

function Header() {
  const [menuOpen, setMenuOpen]=useState(false);

```

```

return(
  <header className="desktop">
    <div>
      <Link to="/"><h1>Kasa</h1></Link>
    </div>
    <div className="burger" onClick={()=>setMenuOpen(!menuOpen)}>
      <CiMenuBurger />
    </div>
    <nav className={`navbar ${menuOpen ? "open" : ""}`>
      <ul>
        <li><NavLink to="/" className={({ isActive }) => isActive ?
"active" : ""}>Accueil</NavLink></li>
        <li><NavLink to="/projects">A propos</NavLink></li>
        <li><NavLink to="/contact">Contact</NavLink></li>
      </ul>
    </nav>
  </header>
);
}
export default Header;

```

J'importe le composant *header* dans le fichier *main.tsx* directement que je place au-dessus de *App* pour qu'il apparaissent sur toutes les pages du projet. Pareil pour le *footer*.

## Création du composant Card + affichage dynamique sur la Home

Voici le composant *Card*

```

import {Link} from 'react-router-dom'
import { MdOutlinePlace } from "react-icons/md";
import Rating from './Rating'

type AccomodationCardProps = {
  id: string;
  title: string;
  location: string;

```

```

pictures: string[];
tags: string[];
rating: string;
};

const AccomodationCard: React.FC<AccomodationCardProps>=({id,title, location, pictures,
tags, rating})=>(
  <Link to={`/${detail}/${id}`} >
    <article>
      <div>
        <img src={pictures[0]} alt={title} />
      </div>

      <div className='card-content'>
        <h3>{title}</h3>
        <p><MdOutlinePlace />{location}</p>
        <div className='tags'>
          {tags.map((tag, index)=>(
            <span key={index}>
              {tag}
            </span>
          ))}
        </div>
        <Rating rating={rating} />
      </div>
    </article>
  </Link>
)
export default AccomodationCard

```

Dans ce composant *Card*, je commence par déclarer les propriétés attendues et leur type pour ce composant.

Ensuite dans une variable *AccomodationCard* on indique que c'est un composant fonctionnel React qui reçoit les types de props définit juste avant et les déstructure dans les paramètres et on retourne ce que la carte doit contenir:

Elle doit être cliquable (pour emmener sur la page détail de l'hébergement) et va chercher toutes les propriétés nécessaires comme pour commencer, une photo (ici on sélectionne la 1ere du tableau photos)

puis le titre, le lieu, les tags associés et la note.

Pour les tags, comme c'est un tableau de tag on doit utiliser ici le `.map` pour le parcourir et afficher chaque tag dans un `<span>`. Et pour finir pour la note on a un autre composant *Rating* qui a été créé pour permettre de récupérer les notes de chaque hébergement et d'ajouter le nombre d'étoiles qui correspond à la note.

Voici le composant *Rating*:

```
type RatingProps = {
  rating: string;
};

const Rating = ({ rating }: RatingProps) => {
  const note = parseInt(rating);
  return (
    <div className="rating">
      {"★".repeat(note) + "☆".repeat(5 - note)} <span>({rating})</span>
    </div>
  );
};

export default Rating;
```

ici on récupère la propriété qui est de type string et on la convertit en chiffre grâce à `parseInt`, ce qui va permettre d'utiliser ce chiffre pour pouvoir répéter les étoiles pleines (la note) et les étoiles vides, soit la note maximale 5 - la note

Maintenant on peut créer la page *Home*

```
import { useEffect, useState } from "react"
import AccomodationCard from "../components/Card"
import data from '../data/logements.json'

type Accomodation = {
  id: string;
  title: string;
  location: string;
  pictures: string[];
  tags: string[];
  host: {
    name: string;
```

```

    picture: string;
  };
  rating: string;
  description: string;
  equipments: string[];
}

const Home=()=>{
  const [accomodations, setAccomodations]=useState<Accomodation[]>([])
  useEffect(()=>{
    setAccomodations(data as Accomodation[])
  }, [])
  return(
    <>
    <main>
      <div className="title">
        <h2>Trouvez votre logement idéal</h2>
        <p>Découvrez des hébergements uniques et vivez comme un local
partout dans le monde</p>
      </div>
      <div className='accomodation-card'>
        {accomodations.map((accomodation)=>(
          <AccomodationCard
            key={accomodation.id}
            id={accomodation.id}
            title={accomodation.title}
            location={accomodation.location}
            pictures={accomodation.pictures}
            tags={accomodation.tags}
            rating={accomodation.rating} />
        ))}
      </div>
    </main>
    </>
  )
}

export default Home

```

Dans cette page *Home*, on importe ce dont on va avoir besoin, puis on définit les props utilisées dans le composant et leur type. Puis on initialise un tableau vide des logements avec un *useState*, puis on peuple ce tableau grâce au *useEffect* en y ajoutant les données du json. On retourne ensuite ce qui apparaîtra sur la page et on parcourt ce nouveau tableau de logement, chaque carte reçoit les props nécessaires pour s'afficher correctement.

Maintenant passons au **composant *Carousel*** créé avec le plugin Slick et ses fichiers css intégrés

```
import Slider from "react-slick"
import "slick-carousel/slick/slick.css"
import "slick-carousel/slick/slick-theme.css"

type PicsProps = {
  pictures: string[];
};

const Carousel = ({ pictures }: PicsProps) => {
  const settings = {
    dots: true,
    arrows: true,
    speed: 300,
    slidesToShow: 1,
    slidesToScroll: 1,
    infinite: true,
    autoplay: false,
    autoplaySpeed: 1000,
  };

  return (
    <div className="carousel-container">
      <Slider {...settings}>
        {pictures.map((img, index) => (
          <div key={index}>
            <img
              src={img}
              alt={`Slide ${index + 1}`}/>
          </div>
        ))}
      </Slider>
    </div>
  )
}
```

```

    )))
  </Slider>
</div>
);
};

export default Carousel;

```

Donc ici les props nécessaires sont les pictures et se sont des chaînes de caractère.

Dans la création du composant on déstructure cette props et on définit les paramètres de notre carousel, on retourne ensuite le carousel avec ses paramètres, puis on parcourt le tableau des photos.

### Création du composant *Collapse*

```

import { useState } from "react"

type CollapseProps={
  title: string,
  children: React.ReactNode // pour accepter tous les types
}

const Collapse = ({title, children }:CollapseProps) => {
  const [isCollapsed, setIsCollapsed] = useState(false)

  return (
    <div className="collapse">
      <span>
        <h4>{title}</h4>
        <button className="collapse-button" onClick={() =>
setIsCollapsed(!isCollapsed)}>
          {isCollapsed ? " ↑ " : "↓"}</button>
        </span>
        {isCollapsed && (
          <div className="collapse-content">
            {children}
          </div>
        )}
      </div>
    )}
  </div>

```

```

)
}
export default Collapse

```

La on utilise le `useState` pour gérer l'état ouvert ou fermé.

On définit les props du composant et cette fois on utilise le type `React.ReactNode` qui permet d'accepter tous les types. (Ce qu'il convient d'utiliser pour rendre vraiment ses composants utilisables dans d'autres projets).

Ensuite on définit le composant une fois de plus avec les props qui seront utilisées et on définit l'état du composant à `false` : il n'est pas ouvert.

On retourne le résultat du composant en récupérant le titre et au clic du bouton on change l'état du composant qui s'ouvre. A chaque clic ça inverse l'état et le positionnement de la flèche.

Puis on indique que si l'état est vrai on peut afficher la propriété `children`.

### Connexion logique entre page *Home* et page *Detail* (via `:id`)

Sur la page Home on affiche donc tous les logements, chaque logement possède un ID unique. Quand on clique sur le logement, comme on a vu plus haut, on est redirigé vers les détails du logement. On y accède notamment grâce à la route définit dans `App.tsx`

```
<Route path="/detail/:id" element={<Details />} />
```

Sur la page Détails:

On récupère l'id du logement dans l'url grace a `useParams`

On parcourt le tableau de données (tout le json) avec la méthode `.find` pour trouver le logement correspondant à l'id, puis on retourne tout le contenu de cet hébergement, en y ajoutant le composant Carousel d'abord, puis toutes les autres infos avec l'ajout du composant Collapse pour la partie description et d'un nouveau petit composant équipements qui permet d'afficher dans une liste un icône et le nom associé.

Pour utiliser ce composant j'ai crée une fonction `getIcon()` avec un `switch` qui prend en charge tous les équipements possibles et leur icones (récupérés avec react icons)

### Gestion des erreurs (ID introuvable → redirection ou 404)

Si on tape un ID introuvable dans le navigateur, nous sommes redirigés vers une page Erreur 404 grâce à un `if` si l'ID n'existe pas

```

const { id } = useParams<{ id: string }>()
const accomodation = data.find((item: Accomodation)=>item.id === id)

if(!accomodation){

```

```
return <Navigate to="notFound" />  
}
```

Mais aussi grâce à la route créée dans App.tsx

```
<Route path="*" element={<NotFound />} />
```

### Mise en forme avec Sass

Pour le style, un dossier a été créé, il regroupe une feuille de style pour chaque composant, et une également pour chaque page + une feuille de style global du site et une pour y instaurer les variables (fonts + couleurs).

Sass permet d'écrire du CSS plus claire grâce aux variables, à l'imbrication et à la réutilisation du code.

## 5. Informations complémentaires (facultatif)

Le responsive est disponible pour cette application.

Le code est disponible sur mon Github <https://github.com/Zitoone/cp-kasa> et l'application est déployée sur <https://kasa-sage-nu.vercel.app/>

J'ai réalisé seule ce projet en 4 jours dans le cadre d'une de nos évaluations.

## Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

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

Pour cette évaluation, nous sommes chargés de construire la fondation d'une nouvelle plateforme en ligne de gestion d'événements. Cette application permettra à des utilisateurs de découvrir, de s'inscrire et de participer à divers événements tels que des conférences, des ateliers ou des rencontres professionnelles.

#### Description du schéma de la bdd:

Je commence par créer une table utilisateurs (users) qui regroupe les informations suivantes:

- un email unique
- un mot de passe haché pour la création de compte
- Un nom
- Un prénom
- Le rôle de l'utilisateur (participants, organisateurs ou administrateurs)

Je crée ensuite une table d'événements (events) qui indiquera:

- le nom de l'événement
- une description
- la date
- le lieu
- l'organisateur qui l'a créé

Une table de jointure appelée inscriptions sera ensuite créée pour relier les participants souhaitant s'inscrire à un événement choisi. La date et l'heure de leur inscription sera également présente sur cette table.

#### Justification des choix de conception:

J'ai choisi une structure simple, pour éviter par la suite les requêtes plus complexes. Je crée donc 2 tables avec les informations principales, ainsi qu'une table de jointure qui permettra à ces informations de se croiser.

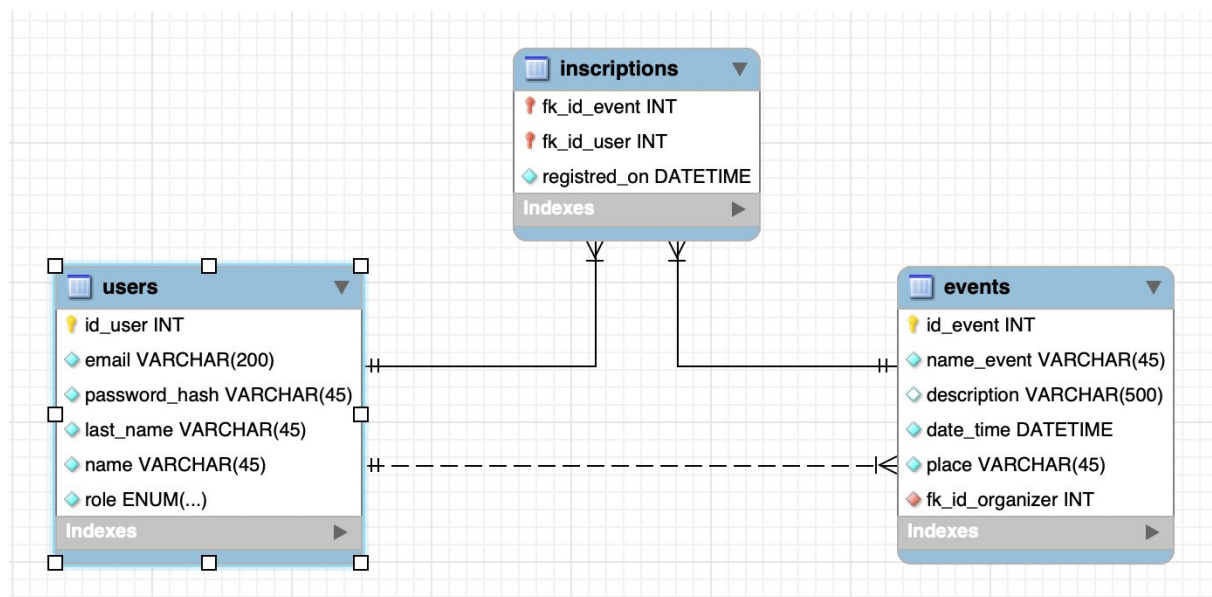
En plus de cela il y a une relation 1-n entre la table *users* et la table *events* afin de pouvoir connaître l'ID de l'utilisateur ayant un rôle d'organisateur lorsqu'il crée un nouvel événement.

Les types de données sont pour la plupart du VARCHAR, sauf les clés primaires et étrangères qui sont en INT, et la colonne role qui est une ENUM avec le choix entre 3 rôles différents.

La plupart des lignes créées ont toutes une contrainte NOT NULL, exceptée la colonne description dans la table des événements. L'organisateur pourra être libre d'écrire une description ou non, même si c'est évidemment préférable.

Tous les autres champs me semblent très importants d'être remplis pour y inclure les informations nécessaires à l'utilisation de cette bdd.

Voici l'illustration du diagramme relationnel créé avec MySQL Workbench:



Pour mettre en place cette bdd j'ai donc fait un export de ce diagramme qui m'a sorti tout une suite de requêtes permettant de créer la bdd et ses tables.

```
CREATE SCHEMA IF NOT EXISTS `events_manager` DEFAULT CHARACTER SET utf8 ;
USE `events_manager` ;

-- -----
-- Table `events_manager`.`users`
-- -----

CREATE TABLE IF NOT EXISTS `events_manager`.`users` (
  `id_user` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(200) NOT NULL,
  `password_hash` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `role` ENUM("Participant", "Organisateur", "Administrateur") NOT NULL,
  PRIMARY KEY (`id_user`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE)
```

```
ENGINE = InnoDB;

-- -----
-- Table `events_manager`.`events`
-- -----

CREATE TABLE IF NOT EXISTS `events_manager`.`events` (
  `id_event` INT NOT NULL AUTO_INCREMENT,
  `name_event` VARCHAR(45) NOT NULL,
  `description` VARCHAR(500) NULL,
  `date_time` DATETIME NOT NULL,
  `place` VARCHAR(45) NOT NULL,
  `fk_id_organizer` INT NOT NULL,
  PRIMARY KEY (`id_event`),
  INDEX `fk_id_organizer_idx` (`fk_id_organizer` ASC) VISIBLE,
  CONSTRAINT `fk_id_organizer`
    FOREIGN KEY (`fk_id_organizer`)
      REFERENCES `events_manager`.`users` (`id_user`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `events_manager`.`inscriptions`
-- -----

CREATE TABLE IF NOT EXISTS `events_manager`.`inscriptions` (
  `fk_id_event` INT NOT NULL,
  `fk_id_user` INT NOT NULL,
  `registred_on` DATETIME NOT NULL,
  PRIMARY KEY (`fk_id_event`, `fk_id_user`),
  INDEX `fk_id_user` (`fk_id_user` ASC) VISIBLE,
  INDEX `fk_id_event` (`fk_id_event` ASC) VISIBLE,
  CONSTRAINT `fk_id_event`
    FOREIGN KEY (`fk_id_event`)
      REFERENCES `events_manager`.`events` (`id_event`)
      ON DELETE NO ACTION
```

```

ON UPDATE NO ACTION,
CONSTRAINT `fk_id_user`
FOREIGN KEY (`fk_id_user`)
REFERENCES `events_manager`.`users` (`id_user`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

La bdd est donc disponible localement sur phpMyAdmin. Pour pouvoir m'y connecter je dois lancer mon serveur MAMP.

Voici l'interface ou on peut voir ici la table des évènements:

id_event	name_event	description	date_time	place	fk_id_organizer
1	Certified DDI Facilitator	Nondisp fx of base of nk of r femr, 7thC	2024-12-01 15:29:23	Tashang	11
2	Hydropower	Medial subluxation of proximal end of tibia, left ...	2025-06-21 00:20:47	Trondheim	13
3	MCSd	Nouvelle description de l'évènement	2025-04-26 23:37:36	Rio de Janeiro	13
4	Crossbeam XOS	Unsp fx shaft of r tibia, 7thQ	2024-08-11 19:40:47	Huijaying	12
5	Lesson Planning	Pellosis hepatitis	2025-03-05 11:00:18	Zastron	11
6	European Studies	Open bite of left little finger w damage to nail, ...	2025-07-15 01:50:45	Lameira	13
7	TSM Administration	Sprain of metatarsophalangeal joint of right lesse...	2024-10-17 06:03:33	Gagah	11
8	Konica	Implant cysts of iris, ciliary body or ant chamber...	2024-12-08 03:36:36	Segezha	13
9	Middle Office	Nondisp commnt fx shaft of rad, r arm, 7thE	2024-09-28 16:40:56	Hetoudian	11
11	Initiation au cheval	Introduction et explication sur l'utilisation d'un...	2025-10-17 11:00:00	Beauduc	11

J'ai peuplé cette bdd avec l'aide du site <https://www.mockaroo.com/> qui m'a permis de déterminer le nombre de champs dont j'avais besoin pour mes tables et le type de données qu'il devra générer.

Voici les données que cela m'a généré par exemple pour la table events (10 évènements ont été ajouté à la bdd)

```

--Table EVENTS

insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Certified DDI Facilitator', 'Nondisp fx of base of nk of r femr, 7thC', '2024-12-01
15:29:23', 'Tashang', 11);

```

```

insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Hydropower', 'Medial subluxation of proximal end of tibia, left knee', '2025-06-21
00:20:47', 'Trondheim', 13);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('MCSD', 'Unspecified fracture of lower end of left tibia', '2025-04-26 23:37:36', 'Vila
Nova de Santo André', 13);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Crossbeam XOS', 'Unsp fx shaft of r tibia, 7thQ', '2024-08-11 19:40:47', 'Hujiaying',
12);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Lesson Planning', 'Peliosis hepatitis', '2025-03-05 11:00:18', 'Zastron', 11);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('European Studies', 'Open bite of left little finger w damage to nail, subs', '2025-07-
15 01:50:45', 'Lameira', 13);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('TSM Administration', 'Sprain of metatarsophalangeal joint of right lesser toe(s)',
'2024-10-17 06:03:33', 'Gagah', 11);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Konica', 'Implant cysts of iris, ciliary body or ant chamber, unsp eye', '2024-12-08
03:36:36', 'Segezha', 13);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('Middle Office', 'Nondisp commnt fx shaft of rad, r arm, 7thE', '2024-09-28 16:40:56',
'Hetoudian', 11);
insert into events (name_event, description, date_time, place, fk_id_organizer) values
('GCIH', 'Inj right quadriceps muscle, fascia and tendon, subs encntr', '2025-04-06
12:32:40', 'Tanshan', 11);

```

J'ai également peuplé la table users (avec 16 utilisateurs) et la table inscriptions avec une dizaine d'inscriptions.

Grâce à ces mocks de données, j'ai pu tester les requêtes SQL à mettre en place pour la suite de cette évaluation grâce aux 11 user stories données dans les consignes.

Nous allons reprendre ici chaque user stories qui regroupe tous les types de requêtes (SELECT, INSERT, UPDATE, DELETE).

#### US 1

```

insert into users (email, password_hash, last_name, name, role)
values ('takakas0@opera.com', '$2a$0dfkzimUx5PFmlupJ', 'Simone', 'ShiTarik',

```

```
'participant');
```

Cette requête nous permet de créer un nouveau compte participant afin qu'il puisse s'inscrire à des évènements.































Elle insère dans la table users créée auparavant son email, son mot de passe haché, son nom de famille, son nom et son rôle parmi les 3 rôles possibles.

## US 2

```
SELECT name_event as nom_evenement, description, date_time as date_heure, place as lieu
FROM events;
```

Cette requête permet à un participant de consulter la liste complète des évènements (nom, description, date, lieu), afin de choisir celui qui l'intéresse.

Voici le résultat de cette requête:

<div><div><div><div></div><div></div></div><div></div></div></div>				nom_evenement	description	date_heure	lieu
<input type="checkbox"/>	 Edit	 Copy	 Delete	Certified DDI Facilitator	Nondisp fx of base of nk of r femr, 7thC	2024-12-01 15:29:23	Tashang
<input type="checkbox"/>	 Edit	 Copy	 Delete	Hydropower	Medial subluxation of proximal end of tibia, left ...	2025-06-21 00:20:47	Trondheim
<input type="checkbox"/>	 Edit	 Copy	 Delete	MCSD	Nouvelle description de l'évènement	2025-04-26 23:37:36	Rio de Janeiro
<input type="checkbox"/>	 Edit	 Copy	 Delete	Crossbeam XOS	Unsp fx shaft of r tibia, 7thQ	2024-08-11 19:40:47	Hujiaying
<input type="checkbox"/>	 Edit	 Copy	 Delete	Lesson Planning	Peliosis hepatis	2025-03-05 11:00:18	Zastron
<input type="checkbox"/>	 Edit	 Copy	 Delete	European Studies	Open bite of left little finger w damage to nail, ...	2025-07-15 01:50:45	Lameira
<input type="checkbox"/>	 Edit	 Copy	 Delete	TSM Administration	Sprain of metatarsophalangeal joint of right lesse...	2024-10-17 06:03:33	Gagah
<input type="checkbox"/>	 Edit	 Copy	 Delete	Konica	Implant cysts of iris, ciliary body or ant chamber...	2024-12-08 03:36:36	Segezha
<input type="checkbox"/>	 Edit	 Copy	 Delete	Middle Office	Nondisp commnt fx shaft of rad, r arm, 7thE	2024-09-28 16:40:56	Hetoudian
<input type="checkbox"/>	 Edit	 Copy	 Delete	Initiation au cheval	Introduction et explication sur l'utilisation d'un...	2025-10-17 11:00:00	Beauduc

## US 3

```
INSERT INTO inscriptions (fk_id_event, fk_id_user) VALUES (4, 7);
```

Celle-ci permet à un participant de pouvoir s'inscrire à un événement spécifique, afin de réserver sa place en insérant une nouvelle ligne dans la table inscriptions en y ajoutant l'ID de l'utilisateur et l'ID de l'évènement.

## US 4

```
SELECT name_event as evenements_inscrits FROM inscriptions i
LEFT JOIN events e ON e.id_event=i.fk_id_event
LEFT JOIN users u ON u.id_user=i.fk_id_user
WHERE last_name="Coletta" AND name="Spiller";
```

Cette US permet à un participant nommé ici Coletta Spiller de retrouver la liste de tous les évènements auxquels elle est déjà inscrite, afin de gérer son agenda personnel.

Cette requête joint les 3 tables de la bdd avec des LEFT JOIN.

Voici le résultat obtenu:

evenements_inscrits	date_heure	lieu
Certified DDI Facilitator	2024-12-01 15:29:23	Tashang
MCSD	2025-04-26 23:37:36	Rio de Janeiro
Crossbeam XOS	2024-08-11 19:40:47	Hujiaying
Konica	2024-12-08 03:36:36	Segezha

#### US 5

```
insert into events (name_event, description, date_time, place, fk_id_organizer)
values ('Initiation au cheval', 'Introduction et explication sur l'utilisation d'un
cheval', '2025-10-17 11:00:00', 'Beauduc', 11);
```

L'US 5 concerne cette fois l'organisateur qui veut pouvoir ajouter un nouvel évènement à la plateforme, afin de le proposer à la communauté.

Avec cette requête nous insérons un nouvel évènement dans la table events avec les valeurs attendues: le nom, la description, la date et l'heure, le lieu et l'ID de l'organisateur grâce à une clé étrangère qui fait référence à la table users avec laquelle on pourra retrouver le nom de cette personne.

#### US 6

```
SELECT CONCAT(last_name, " ", name) as nom_participant, email, name_event as evenement
FROM users u
LEFT JOIN inscriptions i ON i.fk_id_user=u.id_user
LEFT JOIN events e ON e.id_event=i.fk_id_event
WHERE fk_id_organizer= 13;
```

Ici l'organisateur peut obtenir la liste des noms et les emails associés de tous les participants inscrits à tous ses évènements.

Pour sélectionner uniquement un évènement, nous aurions pu modifier la clause WHERE:

```
WHERE name_event="European Studies";
```

#### US 7

```
SELECT COUNT(i.fk_id_user) as nombre_participant, name_event as evenement
FROM inscriptions i
LEFT JOIN events e ON e.id_event=i.fk_id_event
GROUP BY e.name_event
ORDER BY nombre_participant DESC;
```

Cette requête permet à un administrateur de pouvoir compter le nombre de participants pour chaque

événement, ce qui lui permettra d'identifier les sujets les plus populaires et d'orienter sa stratégie. On compte ici (avec la fonction d'agrégation COUNT) le total de participants inscrits pour chaque événement et on les trie par le nombre le plus élevé.

nombre_participant	1	evenement
4		Konica
4		Crossbeam XOS
2		Middle Office
2		Lesson Planning
2		Certified DDI Facilitator
2		European Studies
1		MCSD

#### US 8

```
SELECT CONCAT(last_name, " ", name) as nom_participant FROM users WHERE
role="participant" OR role="organisateur";
SELECT COUNT(id_user) as nombre_utilisateurs FROM users
WHERE role="participant";
```

La première requête aide l'administrateur du site à afficher la liste complète de tous les utilisateurs, ici trié par leur rôle de participant ou d'organisateur (ce qui exempt de la liste les éventuels autres administrateurs)

La deuxième requête permet de les compter afin d'avoir une vue d'ensemble de la communauté grandissante.

#### US 9

```
UPDATE events SET description="Nouvelle description de l'évènement", place="Rio de
Janeiro"
WHERE name_event="MCSD";
```

Cette requête permet à un organisateur de modifier les détails d'évènement créé afin de s'assurer que les informations communiquées aux participants sont toujours exactes.

Ici on modifie la description et le lieu de l'évènement nommé MCSD

#### US 10

```
DELETE FROM inscriptions WHERE fk_id_user=4 AND fk_id_event=10;
```

Avec cette requête, un participant annule sa réservation à un événement pour libérer la place s'il a un empêchement.

On annule donc son inscription grâce aux ID du participant et de l'évènement.

#### US 11

```
DELETE from inscriptions where fk_id_event=10;

DELETE FROM events WHERE name_event="GCIH";
```

Ici on permet à l'organisateur d'annuler un événement si besoin.

Cette requête se fera en 2 temps, d'abord une pour annuler les inscriptions présentent dans la table inscriptions grâce à l'ID de l'événement, puis ensuite pour annuler l'événement dans la table events grâce à son nom.

## 2. Précisez les moyens utilisés :

Pour concevoir, créer et manipuler ma base de données, j'ai mis en place un environnement de développement local en installant le serveur MAMP sur mon Mac. Celui-ci me permet d'accéder à phpMyAdmin, un outil web qui facilite la gestion des bases de données MySQL.

Sur macOS, il n'est pas nécessaire de configurer un nom d'utilisateur et un mot de passe pour se connecter à phpMyAdmin, contrairement à Windows où l'identifiant par défaut est souvent "root".

Pour la partie conception, j'utilise MySQL Workbench. Cet outil me permet de modéliser graphiquement la structure de ma base de données (tables, relations, types de données, etc.) et de générer automatiquement le script SQL correspondant. Une fois le design terminé, j'exporte la base de données pour l'importer dans phpMyAdmin via MAMP ou en copiant les requêtes SQL générées.

J'ai réalisée cette évaluation seule dans le cadre de cette formation le 28 Juillet 2025, la soutenance pour la présentation de mon travail a été faite dès le lendemain.

## Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

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

Cette évaluation a pour but de concevoir le cœur du système d'une application web de gestion de tournois e-sport, sans se préoccuper de l'interface utilisateur avec une base de données (bdd) fournie et peuplée.

J'ai commencé par créer mon fichier de dossiers avec les dossiers et fichiers de base:

- Dossier config avec un fichier *connect.php* pour la connexion PDO à la bdd incluant une gestion des erreurs
- Dossier database avec un fichier *db.sql* qui regroupe les requêtes données dans les consignes pour la création de la structure de la bdd et du peuplement de données
- Un fichier [readme.md](#) qui contient toutes les informations nécessaires au projet

Je crée ensuite un nouveau repository sur Github que j'initialise avec un premier commit

Je lance mon serveur MAMP pour pouvoir me connecter à phpMyAdmin et pouvoir créer et visualiser ma base de données.

Je crée la bdd et ses différentes tables avec les requêtes données. J'ajoute ensuite les requêtes pour peupler cette bdd.

Je crée également un compte utilisateur avec un mot de passe qui permet de s'y connecter via PDO (masterofesport - sport1234) que j'associe à la bdd esports et qui aura tous les privilèges possibles.

Je peux commencer à construire mon projet en me basant sur les US données dans les consignes.

*US1 : Création de compte*

- *En tant que nouveau participant, je veux pouvoir créer un compte avec email et mot de passe, afin de pouvoir rejoindre des équipes et participer à la plateforme.*

Pour cette US je crée un nouveau fichier intitulé *register.php* dans lequel j'initialise un doctype pour y insérer un formulaire d'inscription basique HTML qui permettra au nouvel utilisateur d'y entrer son nom d'utilisateur, son email et son mot de passe.

Je crée également un fichier *functions.php* qui abritera toutes mes fonctions créées liées à l'utilisation de la bdd.

Je commence le code PHP toujours au dessus du HTML avec d'abord une vérification du formulaire dans lequel j'intègre un message informatif pour l'utilisateur.

Explication du code

```

<?php
session_start();
require_once "config/connect.php";
require_once "functions.php"; //On commence par insérer à ce fichier le fichier de connexion à la bdd et celui des fonctions

$msg=""; // j'initialise le msg informatif que j'introduis sous le formulaire dans le HTML
$error=false; // Je crée une variable booléenne initialisée à FAUX pour les éventuelles erreurs

//Vérification du formulaire d'inscription:

if(!empty($_POST["register_submit"])){ // Si le formulaire nommé register_submit n'est pas vide
    if(empty($_POST["username"])){ // Si la case username est vide alors le msg informatif apparaît et l'erreur est VRAIE
        $msg= '<span style="color:red; font-weight:bold; font-size:120%;>Username is required</span>';
        $error=true;
    }
    if(empty($_POST["email"])){ // Pareil avec la case email
        $msg= '<span style="color:red; font-weight:bold; font-size:120%;>Email is required</span>';
        $error=true;
    }
    if(empty($_POST["password"])){ // Pareil avec la case mot de passe
        $msg= '<span style="color:red; font-weight:bold; font-size:120%;>Password is required</span>';
        $error=true;
    }
    if(!$error){ // Si l'erreur est finalement FAUSSE et que tous les champs sont OK alors on peut récupérer les infos avec la méthode POST
        $username = $_POST["username"];
        $email = $_POST["email"];
        $password = password_hash($_POST["password"], PASSWORD_DEFAULT); // On utilise ici la fonction de hachage de mot de passe avec la constante DEFAULT qui utilise
        l'algorithme bcrypt, il faut donc bien s'assurer que le type de résultat qui sera stocké dans la bdd a bien une longueur de 255 caractères

        $isAlreadyExists = checkIfEmailExists($pdo, $email); // On crée une variable qui retournera la réponse de notre fonction qui permet de vérifier si l'email existe déjà
        dans la bdd (voir fonction : 0 Faux ou 1 Vrai)
        if($isAlreadyExists){ // Si c'est vrai (1), un msg apparaît et l'erreur est VRAIE
            $msg= '<span style="color:red; font-weight:bold; font-size:120%;>Email already exists</span>';
            $error=true;
        }else{ //Si c'est faux, on peut ajouter le nouvel utilisateur à la bdd avec une nouvelle fonction
            $request=addNewUser($pdo, $username, $email, $password);
            if($request){ // si la requête de la fonction s'est bien passée (qu'elle retourne le nombre de ligne affectée) on a un msg de confirmation
                $msg= '<span style="color:green; font-weight:bold; font-size:120%;>Your account is create <a href="account.php">Account</a></span>';
                $error=false;
            }else{ // sinon un msg d'erreur
                $msg= '<span style="color:red; font-weight:bold; font-size:120%;>Registration failed</span>';
            }
        }
    }
}

```

## US2 : Connexion

- En tant qu'utilisateur enregistré, je veux pouvoir me connecter avec mon email et mot de passe, afin d'accéder à mes fonctionnalités personnalisées.

Je crée un nouveau fichier *connexion.php* avec un nouveau doctype et un nouveau formulaire de connexion simple avec email et mot de passe. De la même manière, en haut du fichier je commence le code php en y insérant toujours les fichiers de connexion à la bdd et celui lié aux fonctions.

Explication du reste du code php

```

<?php
session_start(); // On utilise les sessions ici pour que le navigateur enregistre certaines informations tant que le navigateur est ouvert
require_once "config/connect.php";
require_once "functions.php";

$msg = "";
// Je commence par une nouvelle vérification de ce nouveau formulaire, si les 2 champs sont remplis on récupère les données envoyées dont le mdp
if (!empty($_POST["submit"])) {
    if (!empty($_POST["email"]) && !empty($_POST["password"])) {
        $email = $_POST["email"];
        $password = $_POST["password"];

        $user = getUser($pdo, $email); // on appelle la fonction qui permettra de récupérer l'id et le mdp haché

        if ($user && password_verify($password, $user["password_hash"])) { //Ici on vérifie que l'utilisateur existe et on utilise une fonction
        (native) qui vérifie si les mots de passe correspondent, si c'est le cas on enregistre l'ID dans les sessions et on est envoyé sur la page du
        compte (sinon on a un msg d'erreur)
            $_SESSION['id'] = $user['id'];
            header("Location: account.php");
            exit(); // pour arrêter le script après redirection
        } else {
            $msg = '<span style="color:red; font-weight:bold; font-size:120%;>Username and password are not correct</span>';
        }
    } else {
        $msg = '<span style="color:red; font-weight:bold; font-size:120%;>Please enter your username and password</span>';
    }
}

```

Si l'email et le mdp correspondent, on arrive donc sur la page du compte utilisateur, c'est sur cette page

que l'on pourra répondre aux 2 prochaines US.

#### US3 : Déconnexion

- *En tant qu'utilisateur connecté, je veux pouvoir me déconnecter de mon compte, afin de sécuriser mon accès.*

#### US4 : Modifier mon profil

- *En tant qu'utilisateur, je veux pouvoir modifier mes informations personnelles, afin de garder mon profil à jour.*

Sur cette page on retrouvera les informations personnelles de l'utilisateur (pré saisies grâce aux sessions) modifiables avec le bouton UPDATE INFO et aussi un bouton de déconnexion qui ramène à la page de connexion (*logout.php*)

On retrouve aussi un lien vers une nouvelle page qui permet de créer une nouvelle équipe (page teams), ce qui nous emmène l'US 5

#### US5 : Créer une équipe

- *En tant que joueur, je veux pouvoir créer ma propre équipe, afin de participer à des compétitions avec mes coéquipiers.*

Sur cette page nous allons ajouter à la bdd une équipe avec un premier formulaire, chaque joueur connecté aura accès à cette page.

Dans la fonction d'ajout d'équipe il y a 2 requêtes, une qui permet d'ajouter l'équipe à la table teams et l'autre qui permet d'ajouter l'utilisateur connecté dans la table team\_members en tant que capitaine. C'est cette liaison de captain qui lui donne des droits sur ces équipes créées (car dans la table teams, on n'enregistre pas le nom de la personne qui l'a créée)

Avec un deuxième formulaire composé de 2 <select>, le joueur connecté peut sélectionner le nom d'une de ces équipes (grâce à son ID récupéré dans les sessions) puis le nom d'un des membres afin d'ajouter cette association à la table team\_members (ou de retirer un membre).

Ce qui répond à l'US 7 (ajout, suppression de joueurs)

#### US7 : Gérer les membres de mon équipe

- *En tant que capitaine, je veux pouvoir ajouter ou retirer des joueurs de mon équipe, afin d'organiser efficacement mes membres.*

#### US6 : Rejoindre une équipe

- *En tant que joueur, je veux pouvoir rejoindre une équipe existante, afin de jouer en groupe.*

Pour cette US le joueur connectée peut rejoindre une équipe dans la liste, la liste présente toutes les équipes disponibles, sauf les siennes, toujours sur la page teams, dans la section Join a team.

Passons ensuite à la page tournois

#### US12 : Lister les tournois ouverts

- *En tant que joueur, je veux voir tous les tournois ouverts*

*à l'inscription, afin de choisir où participer.*

Lorsqu'on arrive sur cette page(en tant que joueur) on retrouve tous les tournois avec leur informations sous forme de listes grâce à une fonction qui récupère les infos nécessaires.

Ensuite nous pouvons ajouter une équipe de l'utilisateur connecté (car il est captain) a un des tournois disponible, ce qui répond a l'US 11.

*US11 : Inscrire une équipe à un tournoi*

- *En tant que joueur/capitaine, je veux pouvoir inscrire mon équipe à un tournoi, afin de participer officiellement.*

Le formulaire et les fonctions ressemblent fortement a ceux utilisés pour ajouter un joueur a une équipe.

Maintenant nous allons rester sur cette page Tournois, mais nous allons nous connecter avec un autre niveau de compte pour s'orienter sur les US des utilisateurs qui ont un statut organisateur.

*US8 : Créer un tournoi*

- *En tant qu'organisateur, je veux pouvoir créer un tournoi avec nom, jeu, date et règles, afin d'organiser des compétitions*

*US9 : Modifier un tournoi*

- *En tant qu'organisateur, je veux pouvoir modifier les détails d'un tournoi que j'ai créé, afin de corriger ou mettre à jour les informations.*

*US10 : Supprimer un tournoi*

- *En tant qu'organisateur ou admin, je veux pouvoir supprimer un tournoi, afin de retirer un événement annulé ou terminé.*

Grâce aux sessions, si l'utilisateur a un rôle d'organisateur ou d'administrateur, certaines fonctionnalités deviennent accessibles : ajout d'un tournoi avec son descriptif, modification d'un tournoi (redirection vers `modify_tournament.php`) ou suppression d'un tournoi.

## 2. Précisez les moyens utilisés :

Cette évaluation a été réalisée, seule, sur les 3 jours dédiés à l'évaluation.

## Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n°3 

**Documenter le déploiement d'une application dynamique web ou web mobile.**

---

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

---

Le déploiement est une phase essentielle du cycle de vie d'une application web ou mobile. C'est le moment où l'application quitte l'environnement de développement local pour être mise à disposition des utilisateurs dans un environnement stable, sécurisé et accessible.

Une documentation de déploiement claire permet :

- Aux équipes techniques de reproduire l'installation ou la mise en ligne de façon fiable.
- Aux nouveaux développeurs de comprendre l'architecture générale et les choix techniques.
- Aux clients ou utilisateurs de connaître les modalités d'installation, de mise à jour ou de maintenance.

Il existe aujourd'hui plusieurs façons de déployer une application web, selon les besoins et les contraintes du projet :

- Hébergement "classique" sur un serveur mutualisé ou dédié (FTP, Apache, Nginx).
- Machines virtuelles ou VPS pour une maîtrise complète de l'environnement.
- Containers Docker pour un déploiement reproductible et portable.
- Plateformes PaaS (Platform as a Service) comme Railway, Render, Heroku, où l'infrastructure est gérée automatiquement.
- Déploiement serverless pour exécuter du code à la demande (ex : Vercel Functions, Netlify Functions, AWS Lambda).
- CI/CD automatisé, où chaque mise à jour du code déclenche automatiquement un build et un déploiement.

Dans le cadre de mon projet de formation OKAMI CONNECT, le backend a été déployé sur Railway, une plateforme PaaS simple à configurer et adaptée aux API Node.js, tandis que le frontend est déployé sur Vercel, une plateforme moderne optimisée pour les projets JavaScript/React.

## 2. Documentation de déploiement BACK-END OKAMI Connect

- Architecture et environnement

Le backend est une API développée en Node.js / Express, il a été construit sous un environnement de développement local (<http://localhost5001>) avec comme service tiers :

- MongoDB pour la base de données NoSQL managée
- Cloudinary pour un stockage et une optimisation des fichiers (images, PDFs)
- Gmail SMTP pour le service d'envoi d'emails (Nodemailer)
- GitHub pour la gestion de version et déploiement automatique
- Railway comme plateforme PaaS pour l'hébergement de l'API (<https://okami-back-production.up.railway.app>)

Tous ces services requiert des comptes pour leur utilisations et disposent de plan gratuit disponible. C'est une architecture 100% cloud, sans serveur dédié, ce qui simplifie la maintenance et améliore la disponibilité.

Variables d'environnement nécessaires :

Créer un fichier `.env` à la racine (voir le fichier `.env.example`)

Ne jamais committer le fichier `.env` (il doit se trouver dès le départ dans le `.gitignore`)

MONGO\_URI=URI de connexion MongoDB

JWT\_SECRET=Clé secrète pour les tokens JWT

CLOUDINARY\_CLOUD\_NAME=Nom du cloud Cloudinary

CLOUDINARY\_API\_KEY=Clé API Cloudinary  
CLOUDINARY\_API\_SECRET=Secret API Cloudinary  
MAIL\_HOST=Serveur SMTP  
MAIL\_PORT=Port SMTP  
MAIL\_USER=Email d'envoi  
MAIL\_PASS=Mot de passe d'application Gmail  
FRONT\_URL=URL de Front-end

- Procédure de déploiement
  - Etape 1 : Vérifier la structure

S'assurer que app.js écoute sur le port dynamique

```
const port = process.env.PORT || 5001
app.listen(port, () => {
  console.log(`Serveur démarré sur le port ${port}`)
})
```

Vérifier le package.json

```
{
  "type": "module",
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  }
}
```

- Etape 2 : Configuration Railway

Aller sur <https://railway.app>

Cliquer sur Login → Login with Github et autoriser Railway à accéder à vos repos

Cliquer sur New Project

Sélectionner Deploy from Github repo

Choisissez le nom de votre repo

Railway détecte automatiquement Node.js et lance le build

Aller ensuite sur le Dashboard pour générer un domaine public

→ Settings → Networking → Generate Domain, railway nous donne une URL

Retourner dans le Dashboard pour configurer ensuite les variables

→ Variables, puis ajouter toutes les variables listées au dessus

IMPORTANT : Il faut partager ces variables avec le service, pour chaque variable cliquer sur SHARE à côté de la variable, sélectionner le service et valider.

Ne pas créer de variable PORT, Railway la définit automatiquement.

Railway redéploie automatiquement après l'ajout des variables.

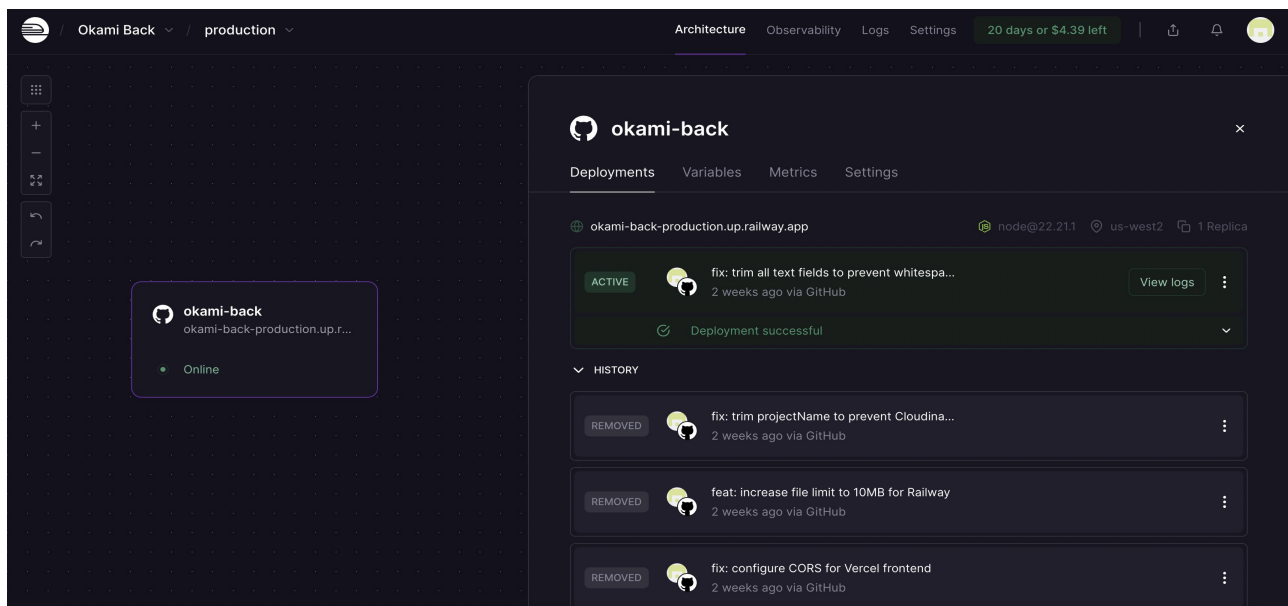
Nous pouvons ensuite cliquer sur le dernier déploiement pour consulter les logs en temps réel et voir si le déploiement est réussi.

Railway est configuré pour le déploiement continu et automatique.

Workflow :

1. Développeur push sur main
2. Railway détecte le changement

3. Railway exécute *npm install* et *npm start*
4. Railway redéploie



- Tests de validation :

Ces tests sont à effectuer après chaque déploiement pour s'assurer du bon fonctionnement de l'API

- Test des endpoints
- Vérification de la connexion MongoDB
- Vérification des uploads Cloudinary
- Test de l'envoi d'email SMTP

### 3. Documentation de déploiement FRONT-END OKAMI Connect

- Architecture et environnement

Le frontend est développé avec React + VITE. Il communique avec l'API backend et sera déployé via Vercel CLI. Un compte Vercel doit être créé et lié à GitHub. Git doit être configuré sur le projet.

- Variables d'environnement :

VITE\_API\_URL=URL de l'API Backend

- Procédure de déploiement :

- Construire le projet localement *npm run build*
- Installer Vercel CLI si ce n'est pas fait *npm install -g vercel*
- Se connecter à Vercel → *vercel login*
- 1er déploiement, *vercel*  
Répondre aux questions

```

olivia@MacBook-Pro-de-Olivia okami-front % vercel
Vercel CLI 48.10.6
? Set up and deploy "~/Desktop/Projet OKAMI/okami-connect/okami-front"? yes
? Which scope should contain your project? oliviacroft73-gmailcom's projects
? Found project "oliviacroft73-gmailcoms-projects/okami-front". Link to it? no
? Link to different existing project? no
? What's your project's name? okami
? In which directory is your code located? ./
Local settings detected in vercel.json:
- Build Command: npm run build
- Output Directory: dist
Auto-detected Project Settings (Vite):
- Development Command: vite --port $PORT
- Install Command: 'yarn install', 'pnpm install', 'npm install', or 'bun install'
? Want to modify these settings? no
? Do you want to change additional project settings? no
? Linked to oliviacroft73-gmailcoms-projects/okami (created .vercel)
? Detected a repository. Connect it to this project? no
? Inspect: https://vercel.com/oliviacroft73-gmailcoms-projects/okami/DMpEqH5kdo1oycqyZXyDgQmmw5Kq [2s]
? Production: https://okami-bsbkjgmet-oliviacroft73-gmailcoms-projects.vercel.app [28s]
? Deployed to production. Run 'vercel --prod' to overwrite later (https://vercel.link/2F).
? To change the domain or build command, go to https://vercel.com/oliviacroft73-gmailcoms-projects/okami/settings
olivia@MacBook-Pro-de-Olivia okami-front % vercel --prod
Vercel CLI 48.10.6
? Inspect: https://vercel.com/oliviacroft73-gmailcoms-projects/okami/51WcRMGFsV2PvAsEjWTxey5RpZK4 [3s]
? Production: https://okami-bt975q6mL-oliviacroft73-gmailcoms-projects.vercel.app [25s]

```

Vercel déploie en preview et fournit une URL de preview

- Ajouter les variables d'environnement via le dashboard Vercel → Settings → Environnement Variables → VITE\_API\_URL
- Déployer en production : `vercel --prod` → cette commande est à faire à chaque changement

Ajouter ensuite l'URL du frontend dans les variables d'environnement du backend

- Tests de validation :
  - Vérification des pages via navigateur
  - Vérification des appels API
  - Test responsive design et performance via Light House

## 5. Informations complémentaires (facultatif)

## Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

## Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] *Olivia Nanquette*..... ,  
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je  
suis l'auteur(e) des réalisations jointes.

Fait à *Brignoles*..... le *12.Décembre 2025*.....  
pour faire valoir ce que de droit.

Signature :



## Documents illustrant la pratique professionnelle

*(facultatif)*

### Intitulé

Cliquez ici pour taper du texte.

## ANNEXES

*(Si le RC le prévoit)*

