|  |  |  |
| --- | --- | --- |
|  |  |  |
| *Nom de naissance* |  | cauche |
| *Nom d’usage* |  | *Entrez votre nom d’usage ici.* |
| *Prénom* |  | Anthony |
| *Adresse* |  | 01500 ambronay |
|  |  |

|  |  |
| --- | --- |
| **Titre professionnel visé** | |
|  | |
| DEVELOPPEUR WEB ET WEB MOBILE | |
|  | |
| **Modalité d’accès :** | |
|  | |
|  | Parcours de formation |
|  | Validation des Acquis de l’Expérience (VAE) |
|  | |

|  |  |
| --- | --- |
| **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**](http://travail-emploi.gouv.fr/titres-professionnels) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sommaire** | | | |
| **Exemples de pratique professionnelle** | | |
| **Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité : Dalle.e clone** | | **p.** | **5** |
|  |  Fenêtre Modale de connexion p. | p. | 5 |
|  |  api : backend p. | p. | 11 |
|  |  Input : barre de recherche p | p. | 16 |
|  |  |  |  |
| **Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité : quai antique** | | **p.** | **20** |
|  |  Diagramme et tableau kanban p. | p. | 20 |
|  |  Créer et développer les composants d’accès aux données p. | p. | 24 |
|  | CRUD Admin dashboard p | p. | 27 |
|  |  |  |  |
| **Titres, diplômes, CQP, attestations de formation** *(facultatif)* | | **p.** | 35 |
| **Déclaration sur l’honneur** | | **p.** | 36 |
| **Documents illustrant la pratique professionnelle** *(facultatif)* | | **p.** |  |
| **Annexes** *(Si le RC le prévoit)* | | **p.** |  |

**Exemples de pratique**

**professionnelle**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **1** | **Cliquez ici pour entrer l’intitulé de l’activité** | | | |
| ***Exemple n°1***  | | **Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité** | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Le but de ce projet était de créer un site qui permet de générer des images en se servant d’une intelligence artificielle. L’idée générale était de réaliser une application proche de dall-e. Un site ou on pourrait télécharger des images  Au préalable j’ai réalisé zoning et mockup à la main. Ensuite j’ai réfléchi à quoi est ce que mes tables ressembleraient, au mongoose schema que j’allais implémenter.  J’ai commencé par démarrer un projet react avec vite. Ensuite j’ai importé les librairies dont j’allais me servir pour ce projet, react-router-dom, file-saver, tailwind, firebase et bien d’autres.  J’allais me servir de tailwind parceque c’est une librairie front mobile-first ainsi que de bootstrap afin de faciliter grandement ma manière de faire le front.  -J’ai commencé par créer les pages du site et j’y ai implémenté les routes. Ensuite j’ai décidé de commencer par la navbar et le système d’authentification. Pour cela j’allais me servir de firebase et du servir d’authentification par email et password qu’il met à disposition.  Je m’occupe de créer les composants de bouton de connexion dans un header de l’app.jsx(la navbar)  -Je créer 2 fenêtres modales, bien plus simples et performantes que des pages, puis j’en fais une pour la connexion et une pour l’enregistrement.  Pour la modale de connexion par exemple je me sers de usestate, useref, usecontect que react met à disposition    const { modalState, toggleModals, signIn } = useContext(UserContext);    const navigate = useNavigate();    const [validation, setValidation] = useState("");    const inputs = useRef([]);    const addInputs = (el) => {      if (el && !inputs.current.includes(el)) {        inputs.current.push(el);      }    };    const formRef = useRef();    const handleForm = async (e) => {      e.preventDefault();      try {        const cred = await signIn(          inputs.current[0].value,          inputs.current[1].value        );        setValidation("");        console.log(cred);        toggleModals("close");        navigate("/create-post");      } catch {        setValidation("Wopsy, email and/or password incorrect")      }    };    const closeModal = () => {      setValidation("");      toggleModals("close");    };  Comme vous avez pu le constater je me sers d’un usercontext dans le usecontext, je vais donc le créer dans la foulée. L’idée sera de déterminer si l’utilisateur est connecté et de pouvoir agir sur le dom en fonction de sa présence ou non grâce à l’affichage conditionnel de react.  export function UserContextProvider(props) {    const signUp = (email, pwd) => createUserWithEmailAndPassword(auth, email, pwd)    const signIn = (email, pwd) => signInWithEmailAndPassword(auth, email, pwd)    const [currentUser, setCurrentUser] = useState();    const [loadingData, setLoadingData] = useState(true);    useEffect(() => {      const unsubscribe = onAuthStateChanged(auth, (currentUser) => {        setCurrentUser(currentUser)        setLoadingData(false)      })      return unsubscribe;    }, [])    // modal    const [modalState, setModalState] = useState({      signUpModal: false,      signInModal: false    })    const toggleModals = modal => {      if(modal === "signIn") {        setModalState({          signUpModal: false,          signInModal: true        })      }      if(modal === "signUp") {        setModalState({          signUpModal: true,          signInModal: false        })      }      if(modal === "close") {        setModalState({          signUpModal: false,          signInModal: false        })      }    }    return (      <UserContext.Provider value={{modalState, toggleModals, signUp, currentUser, signIn}}>        {!loadingData && props.children}      </UserContext.Provider>    )  }  Dans ce usercontext.provider je set :  modalState, toggleModals, signUp, currentUser, signIn    je créer des fonctions, des constantes et je mets ces valeurs en tant que value dans le usercontextProvider que j’exporte jusqu’en haut de l’arborescence dans le main.jsx.  Ensuite je créer le bouton de déconnexion qui aura une fonction async onclick qui utilisera la méthode signout que met à disposition firebase puis je me sers de usenavigate pour retourner dans le menu principal à la déconnexion. Le tout dans une try and catch function.  -Il ne me reste plus qu’à aller dans l’app.jsx pour tester tout ça et mettre tous ces composants.  -Pour la **sécurité** je n’ai pas omis de préciser que si quelqu’un se redirigeait vers la page createPost(l’url createpost) et qu’il n’était pas connecté, le contenu ne s’afficherait pas.  Ainsi, voilà à quoi ressemblait mon app :  const App = () => {    const {currentUser} = useContext(UserContext)    return (        <BrowserRouter>          <SignInModal />          <SignUpModal />          <header className='w-full flex justify-between items-center bg-white sm:px-8 px-3 py-2 border-b border-b-[#e6ebf4]'>            <Link to={"/"}>              <img src={logo} alt="logo" className='sm:w-28 w-20 pic object-contain' />            </Link>            { currentUser &&            <Link to={"/create-post"} className="font-inter sm:text-[20px] main-btn text-sm font-medium bg-[#6469ff] hover:bg-[#474ce7] mx-auto text-white px-4 py-2.5 rounded-md">                Génerer Image            </Link>            }            {            !currentUser  ?            <LoginButton /> :            <LogoutButton />            }          </header>          <main className='sm:p-8 pb-2 px-2 py-8 w-full bg-[#f9fafe] min-h-[calc(100vh-73px)]'>            <Routes>                <Route path='/' element={<Home/>} />                { currentUser &&                  <Route path='/create-post' element={<CreatePost/>} />                }            </Routes>            <Footer />          </main>        </BrowserRouter>    )  }  -Ensuite après avoir fait ceci il me restait plus qu’à m’occuper de l’apparence du site. Je me suis servi des paramètres basiques de tailwind en paramétrant la couleur, les width, color, bg color et aussi du responsif en m’aidant de sm ou m sur certains composants parce que tailwind mets à disposition des média queries mobile first en min-width et est donc une bonne pratique de nos jours.    -Pour simplifier l’ux, je vais créer un loader dont je pourrais me servir dans les différentes fonctions du site pour la génération d’images par exemple.  Dans la page de création de post, l’espace réservé à ceux qui se connectent, je vais mettre en place un système qui va aller chercher des idées de phrases à rentrer si l’utilisateur n’a pas d’idées. Pour cela je vais créer un dossier « constants » puis un fichier index.js et stocker une cinquantaine de phrases dans une fonction surpriseMePrompts que j’exporte dans un dossier utils puis dans un autre index.js.  import { surpriseMePrompts } from '../constants'  export function getRandomPrompt(prompt) {    const randomIndex = Math.floor(Math.random() \* surpriseMePrompts.length)    const randomPrompt = surpriseMePrompts[randomIndex]    if(randomPrompt === prompt) return getRandomPrompt(prompt)    return randomPrompt;  }  Puis je m’en sers dans le fichier createpost dans une fonction handlesurpiseme qui retournera un element du tableau de la fonction.    <FormField              labelName="Génerer texte aléatoirement"              type='text'              name="prompt"              placeholder="Un robot qui mange une pomme"              value={form.prompt}              handleChange={handleChange}              handleSurpriseMe = {handleSurpriseMe}              isSurpriseMe            />  Formfield est un composant que j’ai créé qui sert de modèle pour les inputs de mon app.  const FormField = ({labelName, type, name, placeholder, value, handleChange, isSurpriseMe, handleSurpriseMe}) => {    return (      <div>        <div className="flex items-center gap-2 mb-2">          <label htmlFor={name}            className="block text-sm font-medium text-gray-900 my-3"          >            {labelName}          </label>          {isSurpriseMe && (            <button              type='button'              onClick={handleSurpriseMe}              className="font-semibold text-xs bg-[#b4b4b6] hover:bg-sky-700 hover:text-white p-2 rounded-[50%] text-black"            >              <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-6 h-6">                <path strokeLinecap="round" strokeLinejoin="round" d="M16.023 9.348h4.992v-.001M2.985 19.644v-4.992m0 0h4.992m-4.993 0l3.181 3.183a8.25 8.25 0 0013.803-3.7M4.031 9.865a8.25 8.25 0 0113.803-3.7l3.181 3.182m0-4.991v4.99" />              </svg>            </button>          )}        </div>        <input          type={type}          id={name}          name={name}          placeholder={placeholder}          value={value}          onChange={handleChange}          required          className='bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg          focus:ring-[#4649ff] focus:border-[#4649ff] outline-none block w-full p-3'        />      </div>    )  }  Pour l’affichage des images dans la page principale je me suis servi de display grid.  Ensuite je suis allé m’occuper du backend ou j’ai créé une api fromscratch.  Mes librairies :  -dotenv pour la sécurité  -cloudinary pour stocker mes photos de manière éco résponsable et sécurisée  -cors  -express  -mongoose comme sgbd  -nodemon  -openai comme générateur d’image par IA  J’ai dû me connecter et créer un compte sur mongodb atlas, cloudinary, openai puis j’ai dû mettre les variables d’environnement dans le fichier dotenv dédié    -Ensuite je suis allé faire l’index.js qui allait initialiser les librairies puis les routes que j’allais définir plus tard.  -Ainsi, mon index.js ressemblait à ça :  import express from "express";  import \* as dotenv from "dotenv";  import cors from "cors";  import connectDB from "./mongodb/connect.js";  import postRoutes from './routes/postRoutes.js';  import dalleRoutes from './routes/dalleRoutes.js';  dotenv.config();  const app = express();  app.use(cors());  app.use(express.json({ limit: '50mb' }));  app.use('/api/v1/post', postRoutes);  app.use('/api/v1/dalle', dalleRoutes);  app.get('/', async (req, res) => {    res.send('Hello there')  })  const startServer = () => {    try {      connectDB(process.env.MONGODB\_URL);      app.listen(8080, () => console.log('Server has started on port http://localhost:8080'))    }    catch (error) {      console.log(error)    }  }  startServer();  J’ai ensuite créé mon model mongoose.schema de post  import mongoose from "mongoose";  const Post = new mongoose.Schema({    name: {type: String, required: true},    prompt: {type: String, required: true},    photo: {type: String, required: true},  });  const PostSchema = mongoose.model('Post', Post);  export default PostSchema;  Ce qui donne sur mongoose :    Puis mes routes, par exemple ici, voici le fichier postRoute.js  import express from "express";  import \* as dotenv from 'dotenv';  import {v2 as cloudinary} from 'cloudinary'  import Post from '../mongodb/models/post.js'  dotenv.config();  const router = express.Router();  cloudinary.config({    cloud\_name: process.env.CLOUDINARY\_CLOUD\_NAME,    api\_key: process.env.CLOUDINARY\_API\_KEY,    api\_secret: process.env.CLOUDINARY\_API\_SECRET,  })  //GET ALL POSTS  router.route('/').get(async (req, res) => {    try {      const posts = await Post.find({});      res.status(200).json({ success: true, data: posts });    } catch (err) {      res.status(500).json({success: false, message: 'fetching has failed'})    }  })  //CREATE A POST  router.route('/').post(async (req, res) => {    try {      const { name, prompt, photo } = req.body;      const photoUrl = await cloudinary.uploader.upload(photo);      const newPost = await Post.create({        name,        prompt,        photo: photoUrl.url,      });      res.status(200).json({ success: true, data: newPost });    } catch (err) {      res.status(500).json({success: false, message: "l'érreur est humaine"})    }  })  export default router;  -Mon backend était terminé il ne me restait plus qu’à introduire tout ça dans le front end.  Pour commencer je m’occupe de la fonction de génération d’image au clic. A noter que si le « headers » dans ma méthode post possède des doubles quotes le code est mal compilé.    const generateImage = async () => {      if(form.prompt) {        try {          setGeneratingImg(true)          const response = await fetch('http://localhost:5000/api/v1/dalle', {          method: 'POST',          headers: {            'Content-type': 'application/json'          },          body: JSON.stringify({ prompt: form.prompt }),          })          const data = await response.json();          setForm({...form, photo: `data:image/jpeg;base64,${data.photo}`})        } catch (error) {          alert(error);        } finally {          setGeneratingImg(false);        }      } else {        alert('veuillez remplir le champ !!')      }    }  Puis je me suis occupé de la fonction handlesubmit qui partagerait l’image à tout le monde.    const handleSubmit = async (e) => {      e.preventDefault();      if(form.prompt && form.photo) {        setLoading(true);        try {          const response = await fetch('https://aipicgenerator.onrender.com/api/v1/post', {            method: 'POST',            headers: {              'Content-Type': 'application/json',            },            body: JSON.stringify(form)          })          await response.json();          navigate('/');        } catch (error) {          alert(error)        } finally {          setLoading(false);        }      } else {        alert('Svp rentrez une image à générer avant de vouloir la partager')      }    }  -Je créer une fonction qui va get les posts dans la page principale home.    useEffect(() => {      const fetchPosts = async () => {          setLoading(true);          try {            const response = await fetch('https://aipicgenerator.onrender.com/api/v1/post', {              method: 'GET',              headers: {                'Content-Type': 'application/json'              }            })            if(response.ok) {              const result = await response.json();              setAllPosts(result.data.reverse());            }          } catch (error) {            alert(error)          } finally {            setLoading(false)          }      }      fetchPosts()    }, []);  -Puis je m’occupe de l’affichage de l’image qui sera affichée pour tout le monde avec les informations de l’utilisateur qui l’a partagé ainsi que la fonction de téléchargement de la photo.    Il ne restait plus qu’à m’occuper de la fonction de recherche. J’ai donc créé une nouvelle fonction handlesearchchange    const [searchText, setSearchText] = useState('');    const [searchedResults, setSearchedResults] = useState(null);    const [searchTimeout, setSearchTimeOut] = useState(null);    const handleSearchChange = (e) => {      clearTimeout(searchTimeout)      setSearchText(e.target.value);      setSearchTimeOut(      setTimeout(() => {        const searchedResults =        allPosts.filter((item) =>        item.name.toLowerCase().includes(searchText.toLowerCase())          ||        item.prompt.toLowerCase().includes(searchText.toLowerCase()));        setSearchedResults(searchedResults);      }, 500)      )    }  On distingue clairement que je set un timeout pour que l’affichage dynamique ne soit pas instantané mais qu’il charge au bout d’un certain temps pour laisser le temps à l’utilisateur d’écrire. Dans cette fonction filtre les posts par le nom et le prompt en précisant bien tolowercase car les recherches sont sensibles à la casse.  -Voilà comment j’implémente le tout dans le front :    return (      <section className='max-w-6xl mx-auto'>        <div className="">          <h1 className='rounded-md font-extrabold text-[35px] text-center p-5 mt-6 mb-14 bg-slate-400 text-white'>CréaTive</h1>          <p className='mt-2 text-[#666e75] text-[16px] max-w[500px]'>Parcourez une collection d'images visuellement époustouflantes générées automatiquements par une AI.</p>        </div>        <div className="mt-10">          <FormField            labelName="Rechercher un post :"            type="text"            name="text"            placeholder="Rechercher une image"            value={searchText}            handleChange={handleSearchChange}          />        </div>        <div className="mt-10">          {loading ? (            <div className="flex justify-center items-center">                <Loader />            </div>          ) : (            <>              {searchText && (                <h2 className='font-medium text-[#666e75] text-xl mb-3'>                   Résultats pour : <span className='text-[#222328]'>"{searchText}"</span>                </h2>              )}              <div className="grid lg:grid-cols-4 sm:grid-cols-3 xs:grid-cols-2 grid-cols-1 gap-3">                {searchText ? (                  <RenderCards                    data={searchedResults}                    title="Aucun résultat trouvé .. Réessayez"                  />                ) : (                  <RenderCards                    data={allPosts}                    title="Aucun post créé encore"                  />                )}              </div>            </>          )}        </div>      </section>    )  }  export default Home  Et le rendu en version mobile par exemple est le suivant    -Il ne me restait plus qu’à déployer l’application sur le web. La première étape fût de mettre à jour mon projet puis de push le projet sur github. Puis je me suis inscris sur render qui se met en lien automatiquement avec mes projets git pour mettre mon api en ligne.  -Je n’ai rencontré aucun souci, tout s’est bien passé du premier coup c’était simple et bien expliqué. J’ai pu mettre les variables d’environnement une par une dans une séction dédiée et sécurisée puisque non présentes dans le git à cause du gitignore.  -Ensuite J’ai récupéré l’url de l’api pour la mettre sur mes fonctions ou je me servais du localhost :8080. Une fois que cela a été fait il ne me restait plus qu’à créer un nom de domaine avec un hébergeur sécurisé que j’ai fait à l’aide d’hostinger. J’ai build mon app puis j’ai pu mettre mon projet en ligne en drag & droppant les fichiers dist. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Je me suis aidé de la doc technique de javascript. De stack overflow sur des problèmes techniques. J’ai utilisé des bouts de code de projets que j’avais fait précédemment.  J’ai utilisé le devtools pour comprendre pourquoi certaines requêtes marchaient ou non.  J’ai utilisé youtube et certains tutos pour arriver à comprendre à utiliser les api misent à disposition. | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| J’ai travaillé seul sur ce projet | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *studi* | |
| Chantier, atelier, service | | | *Cliquez ici pour taper du texte.* | | |
| Période d’exerciceDu : *01/01/2023*au :*30/01/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
| Mon site est disponible sur :  totywebsite.fr  Le git du projet est :  https://github.com/Toty-01/randompics  Mon api est dispo sur :  <https://aipicgenerator.onrender.com>  Les posts sont dispos sur :  https://aipicgenerator.onrender.com/api/v1/post | | | | | |
|  | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Activité-type** | **2** | **Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité** | | | |
| ***Exemple n° 1***  | | ***Création d’application web pour la gestion d’un restaurant*** | | | |
|  | | | | | |
|  | | | | | |
| **1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :** | | | | | |
|  | | | | | |
| Le projet «Le quai antique» est une mise en situation de notre école STUDI. L’idée était de créer une application web capable de gérer des réservations du côté client et développer la partie admin en y insérant des fonctionnalités de CRUD, le tout avec sécurité et bonnes pratiques.  J’ai donc commencé ce projet en faisant un tableau Kanban sur Trello comprenant tâches, conseils, démarches à suivre et deadline du projet.  Ensuite j’ai créé la charte graphique en m’aidant des supports donnés pour le sujet comme une idée de logo, les palettes de couleurs ainsi que les fonts. Sur ce même document, j’ai fait les wireframes version desktop et mobile. Voir photo ci-dessous.    Ensuite j’ai réalisé les diagrammes UML grâce à figma :    J’ai aussi réalisé les diagrammes de cas d’utilisation et de séquence grâce à gitmind qui propose un service de création gratuit de diagramme de toutes sortes. Exemple diagramme de cas d’utilisation :    Ensuite je me suis occupé du contenu en téléchargeant des images libres de droit sur unsplash que j’ai reformaté/redimensionné à l’aide de resizepixel.com pour pas qu’elles soient trop volumineuses en termes de performance (- de 800\*800 pixels) et qu’elles puissent être au même format si je les injecte dans un slider et ça permet d’alléger le code en css pour redimensionner l’image. Les avantages sont nombreux.  Ensuite j’ai regardé ce que proposait la concurrence en terme d’ux/ui design et de fonctionnalités pour me faire une idée plus précise du projet.  Ensuite j’ai créé le fichier de base de données pour pouvoir rentrer plus facilement mes requêtes dans le terminal et pour bien comprendre les entités et leur lien. Voir exemple ci-dessous :  CREATE DATABASE IF NOT EXISTS 'restaurant' SET utf8mb4 COLLATE utf8mb4\_general\_ci;  -- table `restaurant`  CREATE TABLE `restaurant` (    `id` int(11) PRIMARY KEY NOT NULL AUTO\_INCREMENT,    `nom` varchar(250) NOT NULL,    `couverts` int(11) NOT NULL,    `ouv\_midi` varchar(250) NOT NULL,    `ferm\_midi` varchar(250) NOT NULL,    `ouv\_soir` varchar(250) NOT NULL,    `ferm\_soir` varchar(250) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  INSERT INTO `restaurant` (`nom`, `couverts`, `ouv\_midi`, `ferm\_midi`, `ouv\_soir`, `ferm\_soir`) VALUES  ('Quai Antique', 60, '12h00', '15h00', '19h00', '22h00');  Après avoir réalisé le fichier de création de base de donnée j’ai paramétré mon environnement de travail en lançant la commande < symfony new restaurant > puis en le reliant à ma base de données dans le fichier dotenv dédié que j’ai ajouté dans le gitignore. J’ai ensuite installé les dépendances puis créé des fichiers scripts et css dans lesquels j’allais mettre le code qui allait englober l’application.  Puis j’ai installé le css et le script de bootstrap que je vais utiliser en librairie front pour ce projet.  J’ai créé ma route par défaut puis lancé la commande « symfony server :start » pour commencer à coder.  A ce moment je commence par voir si tout est bien lié, script et css puis je vais commencer par la balise méta description ainsi que la favicon pour qui doit forcément s’appeler favicon pour que ca marche. Pour cela j’ai dû aller sur un site et créer mon logo au préalable moi-même.    Le site canva.com m’a bien aidé pour ce logo.  Après avoir fais cela j’ai commencé par le header et le footer que j’allais utiliser dans l’index comme ceci :    {% block header %}            {% include 'header.html.twig' %}          {% endblock %}  Puis dans mon fichier footer.html.twig  {% block footer %}    <footer>      <div class="foot">  ……..    </div>    </footer>  {% endblock %}  J’ai appris à utiliser et à utiliser symfony en réalisant ce projet et j’ai découvert la particularité des ancres, des liens et des chemins qui sont définis grâce à asset à la racine du dossier public.  Exemples :                <a href="{{path ('reservation')}}">Réserver une table</a>          <a class="navbar-brand ancre-logo" href="{{path ('home')}}">            <img class="logo" src="images/logo.png" alt="Image Description">          </a>  Après avoir fais le header et le footer j’ai créé la nouvelle page carte que j’ai créé encore grâce à la commande « symfony console make:controller »  J’ai très vite compris le rôle de twig pour la vue, et le rôle des controllers qui redirige vers les pages dédiées et qui pouvait prendre en paramètre les entités ou les repository pour les entrer dans d’éventuelles fonctions que l’on peut transmettre à twig.  Ensuite j’ai créé les formulaire de connexion et d’enregistrement grâce à la commande « symfony make:auth » puis en suivant les instructions de composer. J’intègre ensuite les formulaires de connexion et d’enregistrement sur les pages dédiées.  -Je laisse les paramètres par défaut de 6 caractères minimum pour l’input password.  -Ensuite je vais créer mes premières lignes de code pour l’affichage conditionnel en fonction de l’utilisateur ou de l’admin par exemple            <button class="bouton reza">              {% if app.user %}                <a href="{{path ('reservation')}}">Réserver une table</a>              {% else %}                <a href="{{path ('app\_login')}}">Réserver une table</a>              {% endif %}            </button>  Ou encore pour accéder à la page d’administration            {% if app.user and is\_granted('ROLE\_ADMIN') %}              <li class="nav-item">                <button class='boutons'>                  <a href="{{ path('app\_admin') }}"  class="nav-link">Admin</a>                </button>              </li>            {% endif %}            {% if app.user %}              <li class="nav-item">                <button class='boutons'>                  <a class="nav-link" href="{{path ('app\_logout')}}">Déconnexion</a>                </button>              </li>              {% else %}              <li class="nav-item">                <button class='boutons'>                  <a class="nav-link logout" href="{{path ('app\_login')}}">  Connexion / inscription  </a>                </button>              </li>            {% endif %}  -J’ai ensuite créé le restaurant avec l’administrateur et un random user avec les datafixtures que j’ai envoyé à la base de données.      public function load(ObjectManager $manager): void      {          $admin = new Users();          $admin->setEmail('admin@admin.com');          $admin->setPassword(              $this->passwordEncoder->hashPassword($admin, 'imanadmin')          );          $admin->setRoles(['ROLE\_ADMIN']);          $manager->persist($admin);              $user = new Users();              $user->setEmail("toto@toto.com");              $user->setPassword(                  $this->passwordEncoder->hashPassword($user, 'totototo')              );            $manager->persist($user);          $manager->flush();      }  -J’ai ainsi pu donner le rôle admin à un utilisateur.  Puis la restaurantFixtures :  class RestaurantFixtures extends Fixture  {      public function load(ObjectManager $manager): void      {            $restaurant = new Restaurant();          $restaurant->setNom('Le Quai Antique');          $restaurant->setCouvert(60);          $restaurant->setOuvMidi('12h');          $restaurant->setFermMidi('15h');          $restaurant->setOuvSoir('19h');          $restaurant->setFermSoir('22h');          $manager->persist($restaurant);          $manager->flush();      }  }  -Ainsi j’ai pu retourner les premières données de la database sur le footer(horaires).                <p>                  {% for restau in restaurant %}                    {{ restau.ouvMidi }}                  {% endfor %}                  -                  {% for restau in restaurant %}                    {{ restau.fermMidi }}                  {% endfor %}                </p>  Et dans le controller :      #[Route('/', name: 'home')]      public function number(RestaurantRepository $restaurantRepository): Response      {          return $this->render('index.html.twig', [              'restaurant' => $restaurantRepository->findBy([],              ['ouv\_midi' => 'asc']),  }  Puis côté client :    Je me suis ensuite attelé à la page de réservation. J’ai créé mes formulaire à l’aide de la commande « symfony console make:form » puis je les ai personnalisé directement depuis le formtype comme cet exemple qui m’a été compliqué à codé par exemple :              ->add('heure', TimeType::class, [                  'widget' => 'choice',                  'hours' => [12, 13, 19, 20],                  'minutes' => range(0, 45, 15),                  'attr' => [                  'class' => 'form-control w-75 mx-auto'                  ],              ])  Ensuite j’ai créé mon controller et ma page dédiée dans laquelle j’y ai intégré le formulaire.  Ainsi, mon reservationController était codé comme cela :      public function index(RestaurantRepository $restaurantRepository, EntityManagerInterface $em, Request $request): Response      {          $reservation = new Reservation();          $reservationform = $this->createForm(ReservationFormType::class, $reservation);          $reservationform->handleRequest($request);          if($reservationform->isSubmitted() && $reservationform->isValid()){              $em->persist($reservation);              $em->flush();              $this-> addFlash('success', "réservation éffectuée");              return $this->redirectToRoute("home");          }          return $this->render('reservation.html.twig', [              'controller\_name' => 'ReservationController',              'reservationform' => $reservationform->createView(),              'restaurant' => $restaurantRepository->findBy([],              ['ouv\_midi' => 'asc'])          ]);      }  J’ai ajouté un flash personnalisé à chaque formulaire pour mettre au courant l’admin ou l’utilisateur de ses réservations.  Ensuite je me suis occupé de la page d’administration en créant un nouveau controller « admincontroller » et une nouvelle page twig que j’ai rangé dans un fichier réservé à l’admin pour y mettre les formulaires dont j’allais avoir besoin.  Pour la sécurité, je rentre la ligne de code qui rejette l’utilisateur qui se dirige sur cette page qui n’a pas le rôle « admin » grâce à cette ligne :          $this->denyAccessUnlessGranted('ROLE\_ADMIN');  Ensuite j’ai créé les formulaires en simplifiant certaines entités en mettant le choix à l’utilisateur de rentrer les entrées, les plats, les desserts les menus ainsi que les horaires qui seront modifiables.  La galerie d’image en slider avec le titre au survol à l’accueil m’avait été demandée mais je n’ai pas réussi à installer le bundle nécessaire pour y arriver.  A ce moment-là, j’ai créé la modification des horaires. Pour cela j’ai récupéré le restaurant par son id(2) dans la table restaurant puis je le flush, persist, redirige puis flash une alerte à la validation du formulaire et ça sera pareil pour chaque action générée par l’utilisateur qui aura un impact sur la base de données.          $restaurantRepository = $em->getRepository(Restaurant::class);          $restaurant = $restaurantRepository->find(2);          $horairesform = $this->createForm(HorairesFormType::class, $restaurant);          if($horairesform->isSubmitted() && $horairesform->isValid()){              $em->persist($restaurant);              $em->flush();              $this-> addFlash('success', "Changement horaire éffectuée");                return $this->redirectToRoute("app\_admin");          }  Ensuite j’ai créé les fonctions d’ajout de menu, d’entrées plats et desserts de la même manière sauf que je créais une new entity() à chaque validation de formulaire.         $plat = new Plat();         $platform = $this->createForm(PlatFormType::class, $plat);  Ensuite il fallait créer la fonctionnalité de suppression qui ne fût pas une chose aisée pour ma part.  J’ai dû dans un premier temps faire une route dans le controller avec une fonction qui prenait en paramètre l’id de la table que l’administrateur voudra supprimer.      #[Route('/admin/{id}/deleteentree', name:'deleteentree')]      public function removeentree(?int $id , EntityManagerInterface $em, EntreeRepository $entreeRepository): Response      {          $entrees = null;          if ($id !== null) {          $entrees = $entreeRepository->find($id);          $em->remove($entrees);          $em->flush();          }          $this-> addFlash('success', "Entrée supprimée avec succès");            return $this->redirectToRoute('app\_admin', [              'entrees' => $entrees,          ]);      }  Côté front ça donnait ça : (J’ai créé des cards avec l’id et le nom de chaque entité à supprimer)     <div class="reservation-cards">        {% for menus in menu %}          <div class="card mb-3 mx-3  pb-3" style="width: 18rem;">            <div class="card-body">            <h5 class="card-title">              Menu n°{{ menus.id }}            </h5>            <p class="my-4"><b>{{ menus.nom|upper }}</b></p>            <p>              Prix : {{ menus.prix }} €            </p>              <a href="{{ path('deletemenu', { 'id': menus.id }) }}" class="bouton">Supprimer</a>            </div>          </div>        {% endfor %}      </div>  Et côté client ça donnait ça :    Ensuite je sui allé dans la page carte ou j’ai retourné mes données en y mettant du style comme prédéfinit dans le cahier des charges :    Le site :    J’ai créé la carte du restaurant :    Ensuite j’ai créé un système de filtre dynamique qui permettait de trier les plats par catégories (entrée plat ou dessert) en css et avec des fonctions javascrypt.  Fichier scrypt :  filterSelection("all")  function filterSelection(c) {    var x, i;    x = document.getElementsByClassName("filterDiv");    if (c == "all") c = "";    for (i = 0; i < x.length; i++) {      RemoveClass(x[i], "show");      if (x[i].className.indexOf(c) > -1) AddClass(x[i], "show");    }  }  function AddClass(element, name) {    var i, arr1, arr2;    arr1 = element.className.split(" ");    arr2 = name.split(" ");    for (i = 0; i < arr2.length; i++) {      if (arr1.indexOf(arr2[i]) == -1) {element.className += " " + arr2[i];}    }  }  function RemoveClass(element, name) {    var i, arr1, arr2;    arr1 = element.className.split(" ");    arr2 = name.split(" ");    for (i = 0; i < arr2.length; i++) {      while (arr1.indexOf(arr2[i]) > -1) {        arr1.splice(arr1.indexOf(arr2[i]), 1);      }    }    element.className = arr1.join(" ");  }  // Add active class to the current button (highlight it)  var btnContainer = document.getElementById("myBtnContainer");  var btns = btnContainer.getElementsByClassName("bton");  for (var i = 0; i < btns.length; i++) {    btns[i].addEventListener("click", function(){      var current = document.getElementsByClassName("activ");      current[0].className = current[0].className.replace(" activ", "");      this.className += " activ";    });  }    Lorsque mon site fut terminé je l’ai build et il était près à être mis en ligne J’ai choisi hostinger car j’avais déjà des sites installés dessus.  J’ai donc acheté un nom de domaine puis j’ai transféré mes fichiers avec un protocole ftp.  Ensuite j’ai créé une base de données ou j’ai importé ma base de données depuis mon mysql en local.  Ensuite j’ai modifié mon fichier dotenv pour le relier à la base de données puis j’ai passé le projet en mode prod.  A ce moment-là, il y avait un problème dans mon style et dans mes liens, en réalité, l’architecture du site ne comprenait pas ou était la racine de mon projet ni où était le index.php. Après des heures de recherches, j’ai pu créer les lignes de code nécessaires dans un fichier htaccess à la racine du projet qui déterminaient tous ces critères nécessaires. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **2. Précisez les moyens utilisés :** | | | | | |
|  | | | | | |
| Les moyens utilisés étaient divers et variés.  Je me suis servi de Trello pour gérer les tâches et pour suivre l’évolution de mon travail en mode agile.  J’ai utilisé unsplash qui est une librairie d’images libres de droit.  Je me suis servi de resizepixel.com pour redimensionner les photos  J’ai utilisé le terminal et doctrine pour migrer les données la plupart du temps  J’ai utilisé le tooldev qui sert toujours pour trouver les failles des éléments css qui n’agissent as comme imaginé.  Je me suis servi de la doc technique de symfony bien fourni en termes d’éléments notamment pour les formtypes.  StackOverflow aussi m’a été d’une grande aide pour arriver à solutionner les problèmes.  Git comme outil de versionning.  canva.com m’a aidé pour le logo et ma favicon | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
| **3. Avec qui avez-vous travaillé ?** | | | | | |
|  | | | | | |
| J’ai travaillé seul et à l’aide de chat gpt pour répondre à des soucis d’ordre technique. | | | | | |
|  | | | | | |
|  | | | | |
|  | | | | | |
| **4. Contexte** | | | | | |
|  | | | |  | |
| Nom de l’entreprise, organisme ou association | | | | *Studi* | |
| Chantier, atelier, service | | | *Cliquez ici pour taper du texte.* | | |
| Période d’exerciceDu : *01/03/2023*au :*18/03/2023* | | | | | |
|  | | | | |
|  | | | | |
| **5. Informations complémentaires** *(facultatif)* | | | | |
|  | | | | | |
| Le lien de mon travail se trouve sur : <https://github.com/Toty-01/ecf>  Le lien figma de la charte graphique est disponible sur :  https://www.figma.com/file/2azPNSJ6JFIbHuKa895XIH/Untitled?t=TYpyI3fqyPL5QuTI-1  Le lien figma du diagramme uml : <https://www.figma.com/file/QUUlY847rElSJTa9gAtgyj/Untitled?t=TYpyI3fqyPL5QuTI-1>  Le lien du tableau Kanban est disponible sur : https://trello.com/invite/b/7BHZx2QD/ATTI6adb2be12657f7c0de0ffcb5920bc042F61F8167/modele-kanban | | | | | |
|  | | | | | |

|  |  |  |
| --- | --- | --- |
| **Titres, diplômes, CQP, attestations de formation** | | |
|  | | |
| *(facultatif)* | | |
| **Intitulé** | **Autorité ou organisme** | **Date** |
| **Dev web et web mobile** | **STUDI** | Cliquez ici pour sélectionner une date. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| **Déclaration sur l’honneur** |
|  |
|  |

Anthony cauche

Je soussigné(e) [prénom et nom] ,

déclare sur l’honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l’auteur(e) des réalisations jointes.

ambronay

20/03/2023

Fait à le

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)* |