Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance	Méthot
Nom d'usage	Méthot
Prénom	Margaux
Adresse	32 Place Charles de Gaulle 59491 Villeneuve d'Ascq

Titre professionnel visé

Développeur Web / Web Mobile

MODALITE D'ACCES :

☒ 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

Sommaire

Exemples de pratique professionnelle

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ▸ *Mon Curriculum Vitae en ligne*

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

Afin de mettre en pratique ce que nous avions appris concernant les langages HTML et CSS, et également d'avoir à disposition un support pour notre recherche d'emploi après la formation, il nous a été demandé de développer notre CV en ligne.

Dans un premier temps, j'ai effectué des recherches de CV d'autres développeurs, afin de savoir quelles informations étaient pertinentes sur un CV en ligne, et de quelle façon elles étaient présentées.

Après cette recherche, j'ai conçu le zoning, puis les maquettes de la version mobile et ordinateur, en gardant à l'esprit que le CV devait être « mobile first ». Après validation, j'ai établi la feuille de style, qui reprend les polices et les couleurs qui seront utilisées.

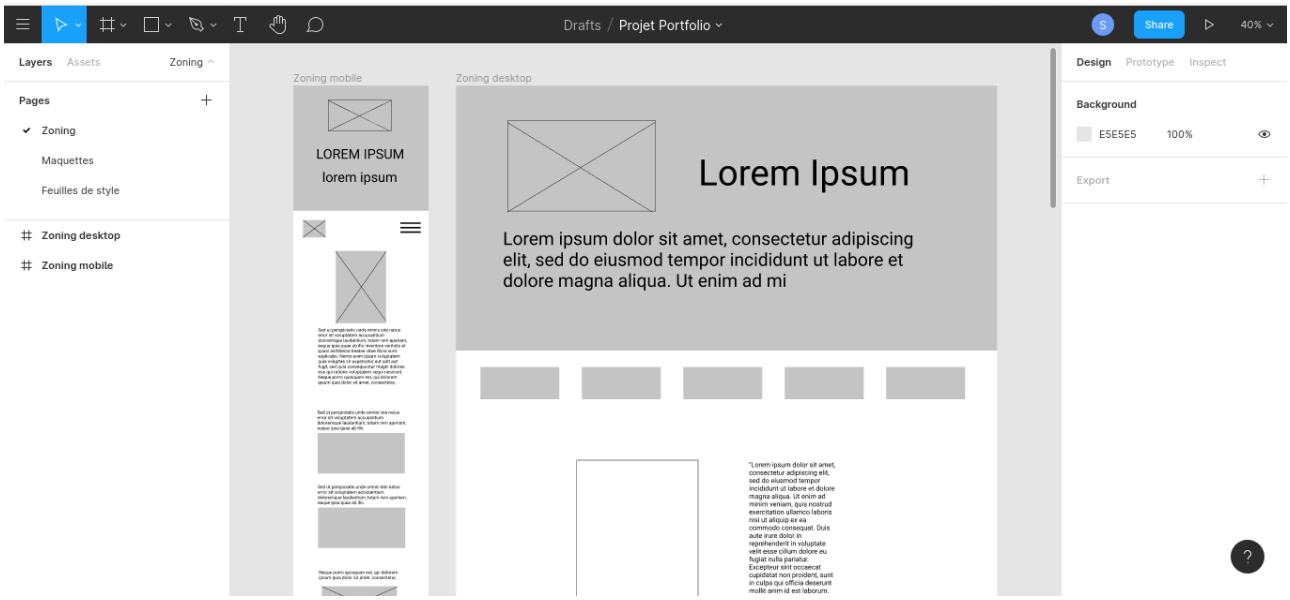
Côté développement, les consignes de cet exercice nous imposaient de respecter certains critères :

- avoir un dossier pour le CSS, un dossier pour les images et un dossier pour le SCSS
- utiliser le framework Bootstrap pour au moins la navbar responsive et la grid
- utiliser la méthodologie BEM afin de respecter le nommage des différentes classes et de leur arborescence
- utiliser les processeurs CSS : le pré-processeur SASS, la librairie POSTCSS pour les post-processeurs afin de minifier le CSS et de l'adapter afin d'avoir un rendu identique sur tous les navigateurs. Pour automatiser cela, nous devions utiliser des scripts pour le dev et la prod.

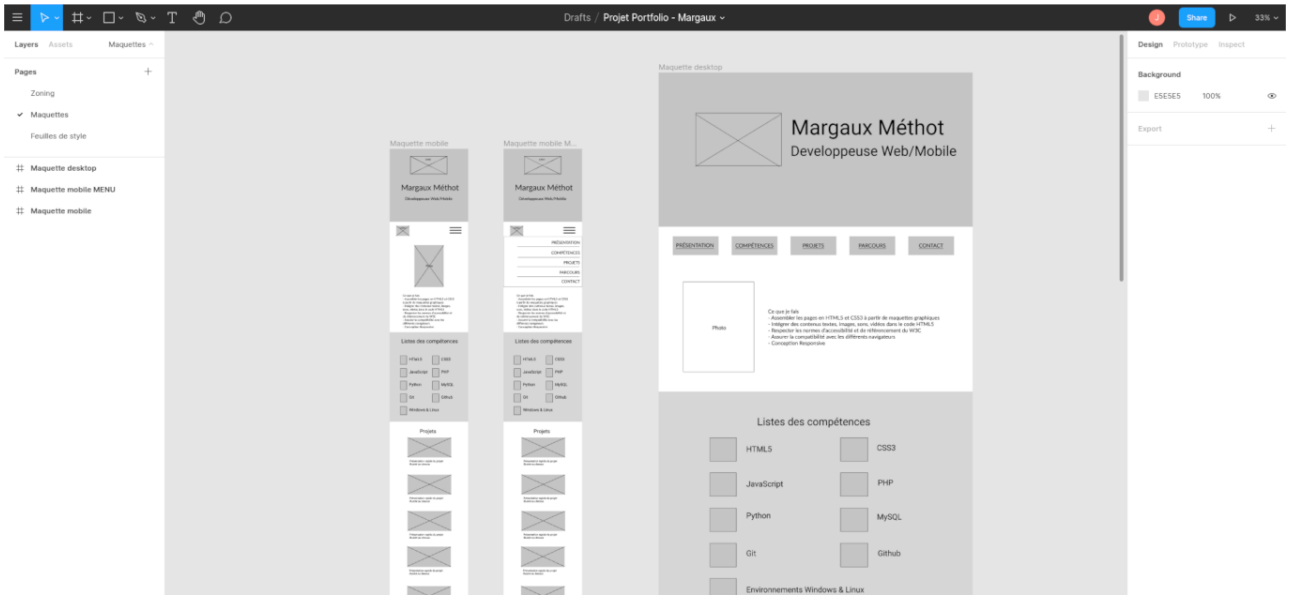
Pour le rendu final : voir annexe 1 page 43

2. Précisez les moyens utilisés :

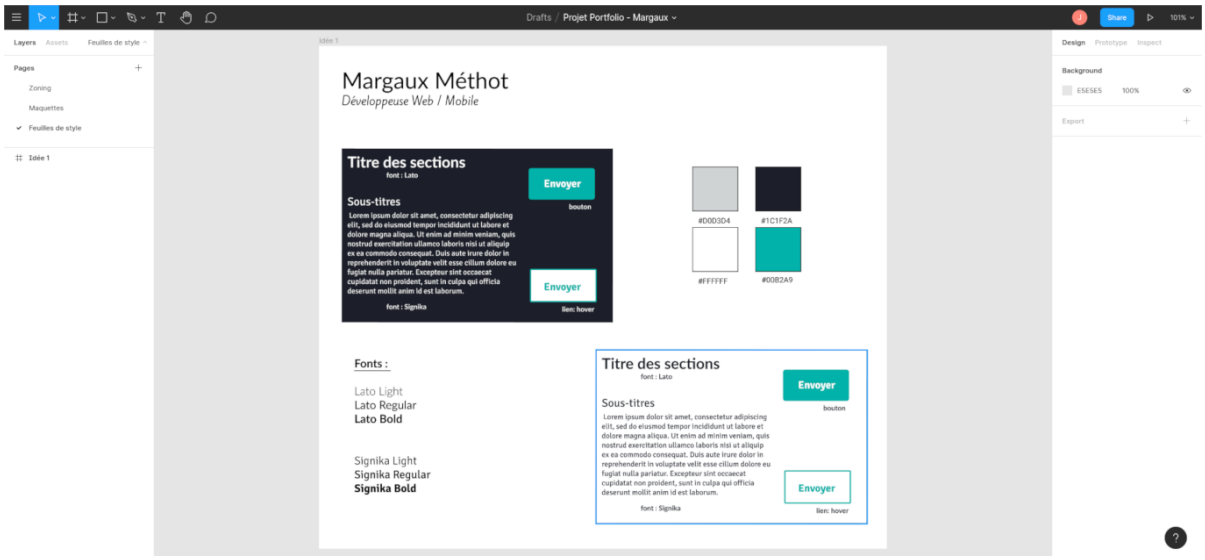
J'ai réalisé tout le travail de maquettage sur l'outil en ligne FIGMA.



Le zoning



Le wireframe



Le feuille de style

Concernant la partie développement, j'ai utilisé l'éditeur de code Visual Studio Code.

J'ai utilisé le framework Bootstrap en important les différentes lignes de code nécessaires comme indiquées dans la documentation.

```
<link
  rel="stylesheet"
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
  integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
  crossorigin="anonymous"
/>
```

Extrait de la balise <head> du fichier index.html

```
<script
  src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
  integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
  crossorigin="anonymous"
></script>
<script
  src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
  integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
  crossorigin="anonymous"
></script>
<script
  src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
  integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
  crossorigin="anonymous"
></script>
<script src="js/app.js"></script>
```

Scripts du fichier index.html

L'import de ce code m'a donc permis d'utiliser Bootstrap pour le développement de ma navbar : à l'aide de certaines balises et de certaines classes, la barre de navigation est développée et responsive en quelques minutes.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light fixed-top">
  <img class="navbar__logo" src="img/logomain.svg" alt="logo" />
  <button
    class="navbar-toggler"
    type="button"
    data-toggle="collapse"
    data-target="#navbarNavAltMarkup"
    aria-controls="navbarNavAltMarkup"
    aria-expanded="false"
    aria-label="Toggle navigation"
  >
    <span class="navbar__icon navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav ml-auto bg-light">
      <a
        class="navbar__item nav-item nav-link active"
        href="#presentation"
        >Présentation<span class="sr-only">(current)</span></a
      >
      <a class="navbar__item nav-item nav-link active" href="#competences"
        >Compétences<span class="sr-only"></span
      ></a>
      <a class="navbar__item nav-item nav-link active" href="#projects"
        >Projets<span class="sr-only"></span
      ></a>
      <a class="navbar__item nav-item nav-link active" href="#career"
        >Parcours<span class="sr-only"></span
      ></a>
      <a class="navbar__item nav-item nav-link active" href="#contact"
        >Contact<span class="sr-only"></span
      ></a>
    </div>
  </div>
</nav>
```

Code de la barre de navigation

L'utilisation de la méthodologie BEM facilite la lecture de l'arborescence d'un fichier HTML et de ses différentes classes.

```
<section class="career" id="career">
    <h3 class="career__title">Parcours</h3>
    <div class="career__descriptions">
        <div class="career__description career__description--right">
            <span class="careerdesc__date">Mars 2020 - Aujourd'hui</span>
            <span class="careerdesc__place">Studi</span>
            <span class="careerdesc__location">Distanciel</span>
            <span class="careerdesc__job">Formation Développeur Web/Mobile</span>
        </div>
```

Exemple d'utilisation de la méthodologie BEM

Pour finir, l'utilisation des processeurs CSS m'a permis d'optimiser le CSS, afin de gagner en performance et de permettre d'avoir un rendu visuel identique sur tous les navigateurs.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "npx node-sass scss/main.scss css/style.css",
  "prod": "node-sass scss/main.scss css/style.css && postcss css/style.css -o css/style.css -u autoprefixer && postcss css/style.css -o css/style.css -u cssnano"
},
```

Scripts permettant l'automatisation

3. Avec qui avez-vous travaillé ?

J'ai travaillé toute seule sur ce projet, mais en bénéficiant du suivi de mon formateur lorsque c'était nécessaire.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***Studi***

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du 14/04/2020 au 02/06/2020**

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°2 ▸ *Interface de gestion de commentaires*

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

Au cours de la formation, nous avons étudié le langage JavaScript, puis plus particulièrement la librairie React, librairie JavaScript qui permet de créer des interfaces utilisateurs.

Afin de mettre en pratique nos apprentissages, nous avons dû reproduire une interface de gestion de commentaires.
Elle devait contenir :

- un formulaire pour ajouter un commentaire
- un emplacement où s'affichaient les commentaires ajoutés
- un bouton qui permettait de passer en mode administration, qui lui seul permettait d'effacer les commentaires précédemment ajoutés.

Pour la mise en forme, nous devions utiliser le framework Bulma.

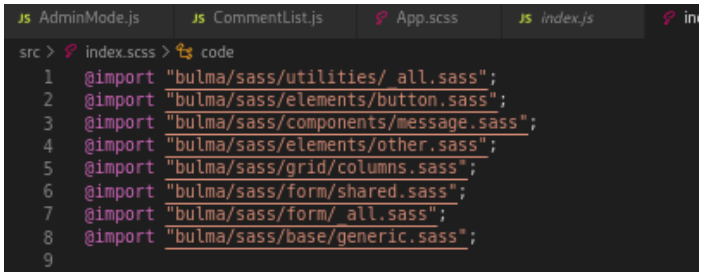
Pour le rendu final : voir annexe 2 page 44

2. Précisez les moyens utilisés :

Pour le développement, j'ai utilisé l'éditeur de code Visual Studio Code. J'ai installé Bulma grâce à la commande :

```
npm install bulma                                    copy
```

Ensuite, il suffit d'importer uniquement ce dont on a besoin dans le fichier index.scss.

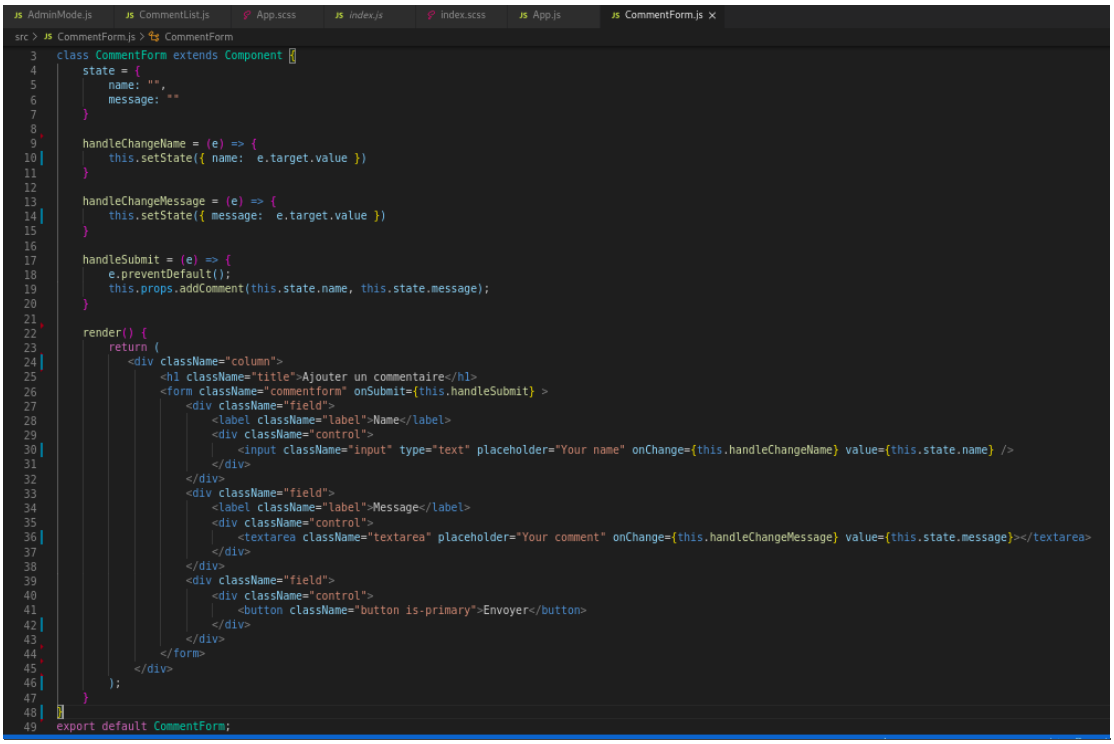


Import des composants Bulma nécessaires

Concernant le développement React, j'ai créé 3 composants :

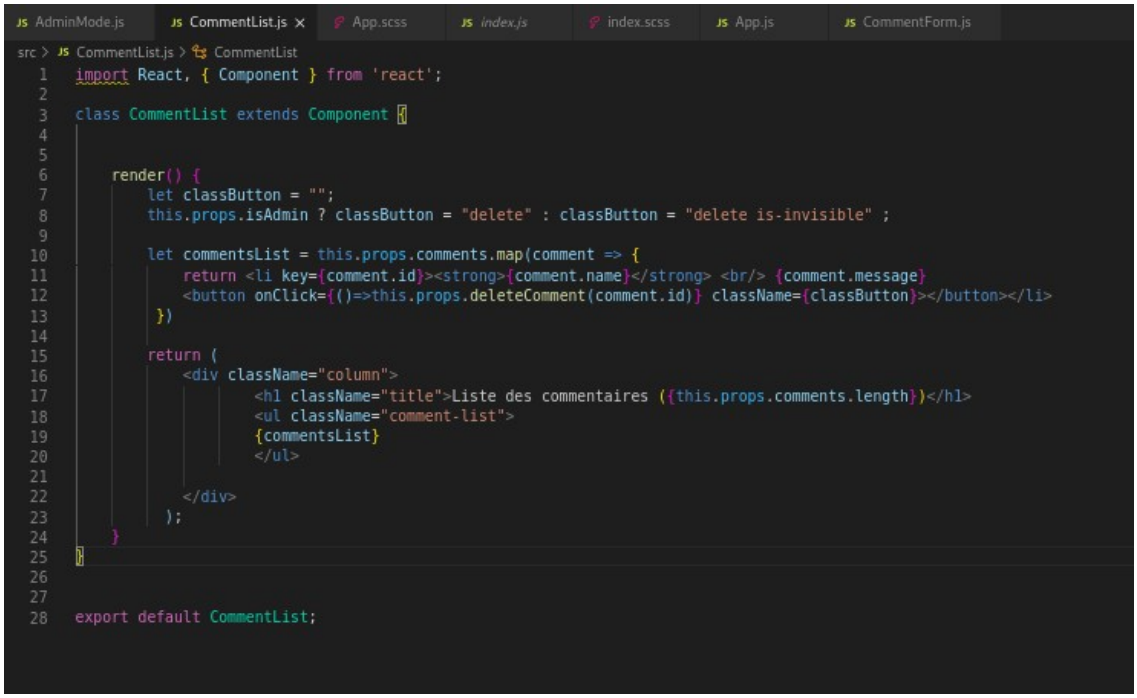
- <CommentList> qui s'occupera de l'affichage des commentaires et de leur bouton de suppression
- <CommentForm> qui s'occupera de l'affichage du formulaire et du traitement des données saisies
- <AdminMode> qui s'occupera de l'affichage du bouton qui activera et désactivera le mode admin

Le composant <CommentForm> traite les données enregistrées grâce aux fonctions handleChangeName, handleChangeMessage et handleSubmit, qui récupèrent et stockent les données saisies par l'utilisateur.



Composant CommentForm

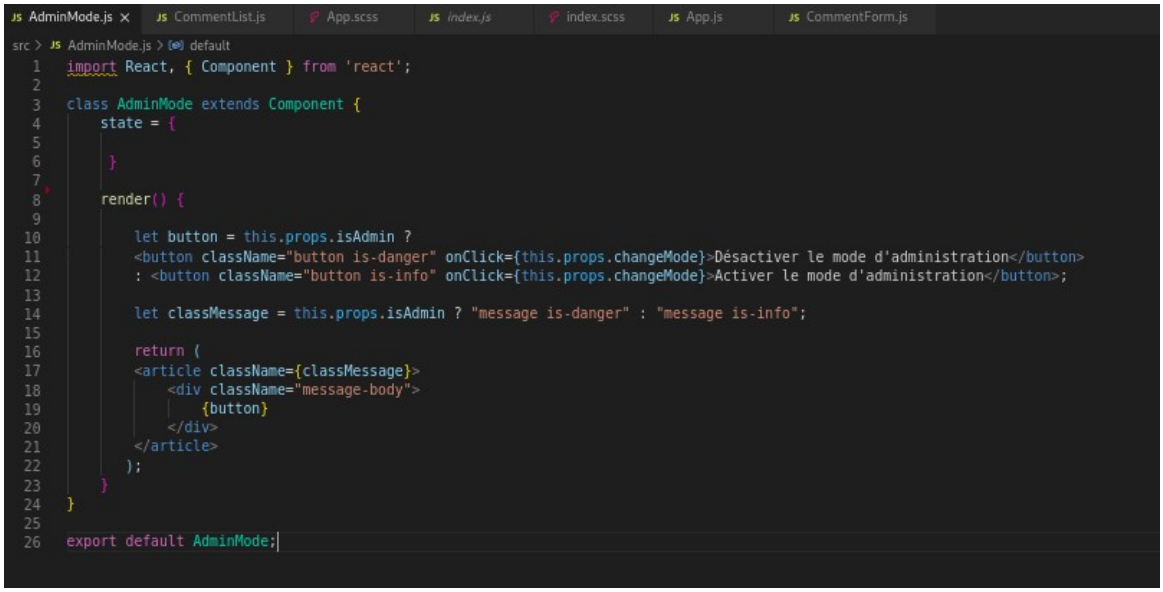
Dans le composant <CommentList> , l'affichage des boutons de suppression dépend de l'état de la variable isAdmin en ajustant la classe des boutons de suppression qui passe de « delete » à « delete is-invisible » qui sont des classes Bulma.



```
import React, { Component } from 'react';

class CommentList extends Component {


    render() {
        let classButton = "";
        this.props.isAdmin ? classButton = "delete" : classButton = "delete is-invisible" ;

        let commentsList = this.props.comments.map(comment => {
            return <li key={comment.id}><strong>{comment.name}</strong> <br/> {comment.message}
            <button onClick={()=>this.props.deleteComment(comment.id)} className={classButton}></button></li>
         })

        return (
            <div className="column">
                    <h1 className="title">Liste des commentaires ({this.props.comments.length})</h1>
                    <ul className="comment-list">
                    {commentsList}
                    </ul>

            </div>
          );
    }
}


export default CommentList;
```

Composant CommentList

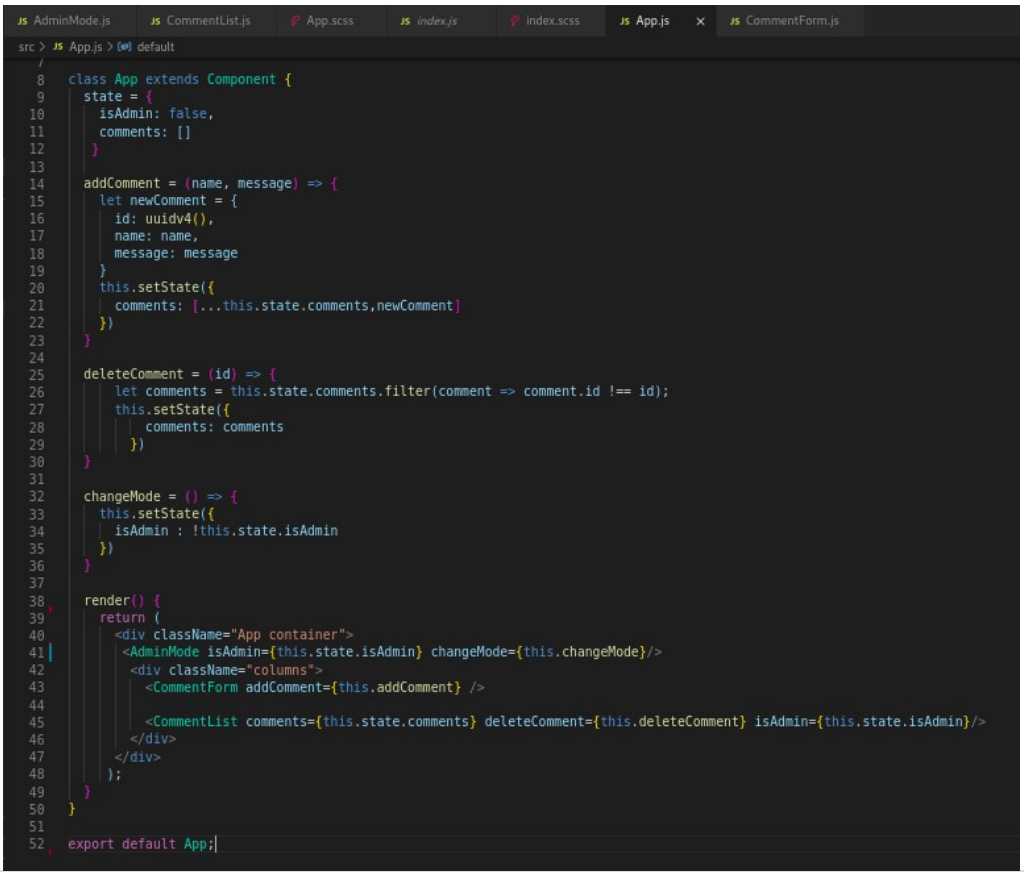
Le composant <AdminMode> adapte l'affichage du bouton en fonction de la variable « isAdmin » et en adaptant la couleur à l'aide de la variable « classMessage ».



```
import React, { Component } from 'react';

class AdminMode extends Component {
    state = {

     }

    render() {

        let button = this.props.isAdmin ?
        <button className="button is-danger" onClick={this.props.changeMode}>Désactiver le mode d'administration</button>
        : <button className="button is-info" onClick={this.props.changeMode}>Activer le mode d'administration</button>;

        let classMessage = this.props.isAdmin ? "message is-danger" : "message is-info";

        return (
        <article className={classMessage}>
            <div className="message-body">
                {button}
            </div>
        </article>
       );
    }
}

export default AdminMode;
```

Composant AdminMode

Une fois ces composants créés, il me suffit de les importer dans mon fichier App.js.
C'est dans ce fichier que sera géré l'état « isAdmin » dont dépendent les composants AdminMode et CommentList, qui ont un affichage conditionnel qui dépend de cette variable booléenne :



Composant App.js

3. Avec qui avez-vous travaillé ?

J'ai travaillé toute seule sur ce projet, mais en bénéficiant du suivi de mon formateur lorsque c'était nécessaire.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***Studi***

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du 22/06/2020 au 30/06/2020**

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°3 ▸ ***Boutique e-commerce sous WordPress***

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

Lors d'un projet réel réalisé lors de la formation, j'ai eu l'occasion de mettre en place une boutique e-commerce sur un site internet déjà existant et réalisé sous WordPress.

Monsieur Gauthier Royer, le client, gérant du bar « La relique », souhaitait ajouter à son site vitrine une partie boutique pour la vente de bières brassées par un partenaire.

Afin de procéder au développement, j'ai récupéré les données du site grâce à l'extension « All-in-One WP Migration » qui génère un fichier .wpress qui contient l'ensemble des données ainsi que la base de données.
J'ai installé un site WordPress vierge sur mon serveur privé, installé l'extension « All-in-One WP Migration » puis importé le fichier .wpress du site initial. Ainsi, je pouvais procéder au développement des fonctionnalités tout en gardant le site vitrine tel quel.

Avant de mettre en place la boutique, j'ai procédé à divers ajustements nécessaires afin de garantir au mieux la sécurité du site internet.
Aucune mise à jour n'avait été effectuée depuis la mise en ligne du site : pas de mise à jour de WordPress, ni du thème, ni des extensions installées.

De plus, les modifications de style avaient été faites sur le thème parent, sans création d'un thème enfant. J'ai donc procédé à la mise en place du thème enfant. Pour cela, j'ai créé un dossier oceanwp-child, dans lequel j'ai créé 2 fichiers :

- functions.php qui contient la fonction qui charge le style du thème parent et du thème enfant
- style.css qui contiendra les modifications du style appliquées au thème enfant

~ functions.php

```php
<?php

function oceanwp_child_enqueue_parent_style() {
    // Dynamically get version number of the parent stylesheet (lets browsers re-cache your stylesheet when you update your theme)
    $theme   = wp_get_theme( 'OceanWP' );
    $version = $theme->get( 'Version' );
    // Load the stylesheet
    wp_enqueue_style( 'child-style', get_stylesheet_directory_uri() . '/style.css', array( 'oceanwp-style' ), $version );

}
add_action( 'wp_enqueue_scripts', 'oceanwp_child_enqueue_parent_style' );

```

functions.php

Une fois le thème enfant créé, il apparait dans le tableau de bord WordPress, on peut donc l'activer comme thème du site.

Ensuite, j'ai procédé à l'ensemble des mises à jour, après avoir effectué une sauvegarde du site à l'aide de l'extension « UpdraftPlus ».

Pour sécuriser le site, j'ai procédé à l'installation de l'extension SecuPress, et suivi les recommandations indiquées.

Une fois le site mis à jour et sécurisé, j'ai pu commencer la mise en place de la boutique e-commerce.
Pour cela, j'ai utilisé l'extension WooCommerce qui permet de mettre en place une boutique e-commerce sur un site WordPress.

Monsieur Royer souhaitait que les commandes soient directement retirées sur place, et que le paiement se fasse en ligne par carte bancaire. J'ai donc mis en place le module de paiement Stripe.

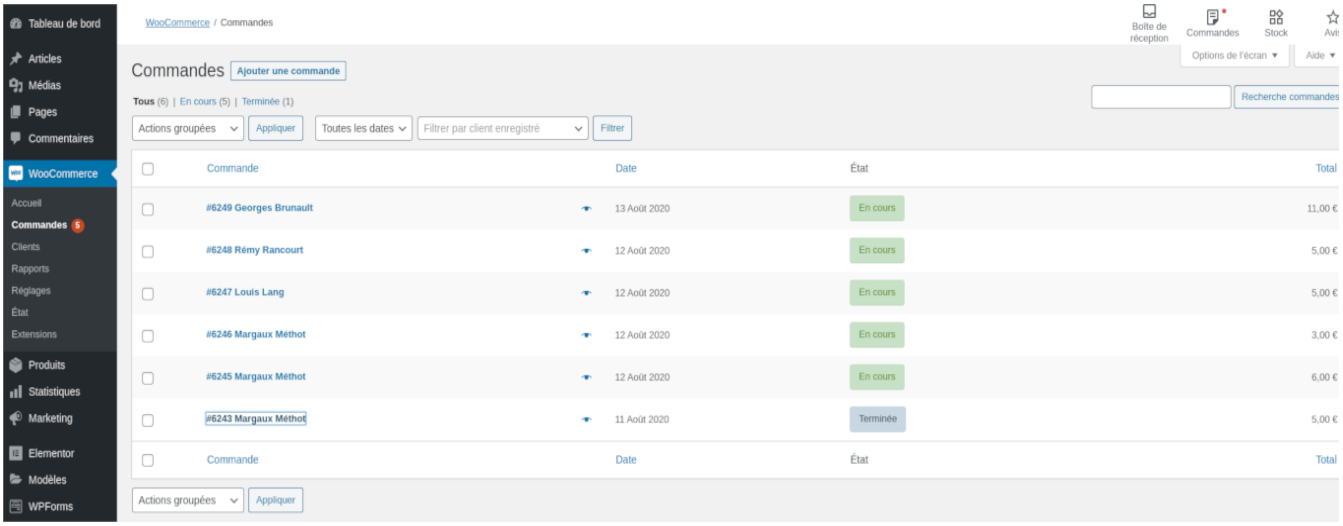
2. Précisez les moyens utilisés :

Une fois l'extension WooCommerce installée, nous avons accès à 2 nouveaux onglets « WooCommerce » et « Produits » dans le tableau de bord WordPress.

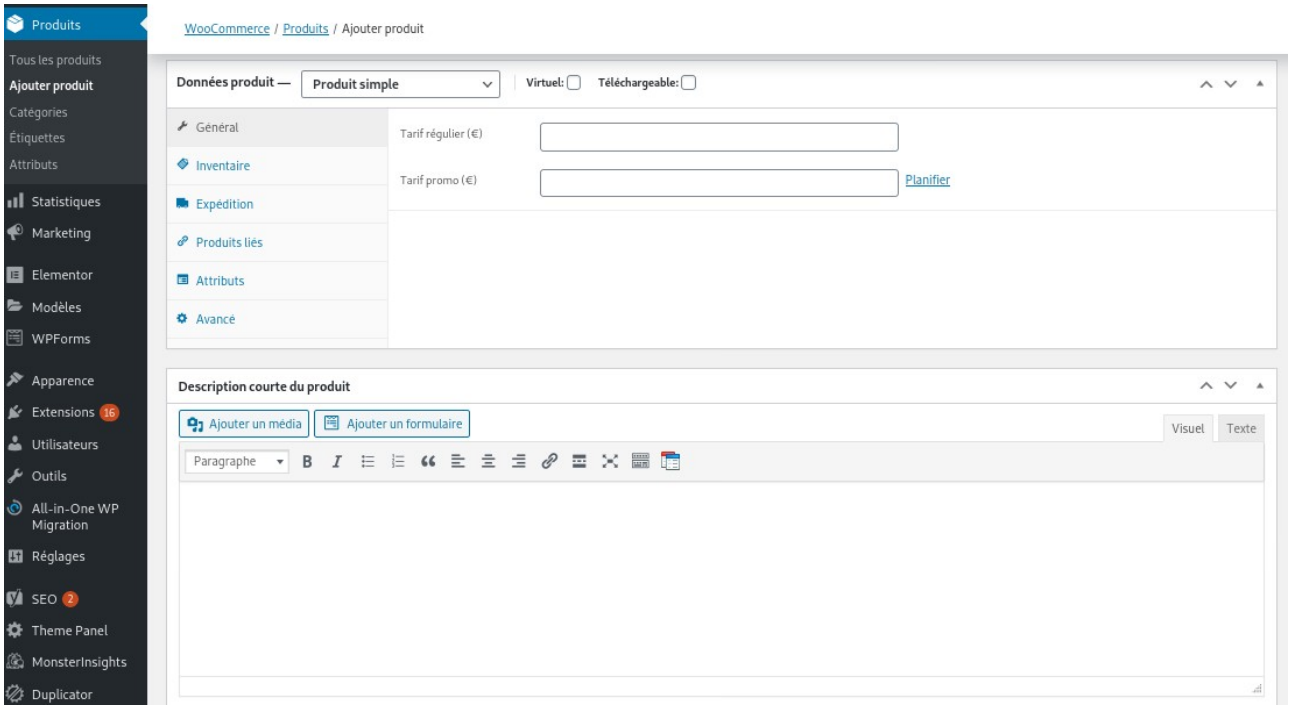
L'onglet « WooCommerce » nous donne accès notamment à la liste des commandes effectuées, à la liste des clients et également à un ensemble de paramètres, par exemple :

- Les types de paiements autorisés sur la boutique
- Les modalités de livraison (que j'ai désactivé car il n'y avait pas de livraison possible)
- Les modes de connexion des clients (s'ils peuvent commander sans compte par exemple)

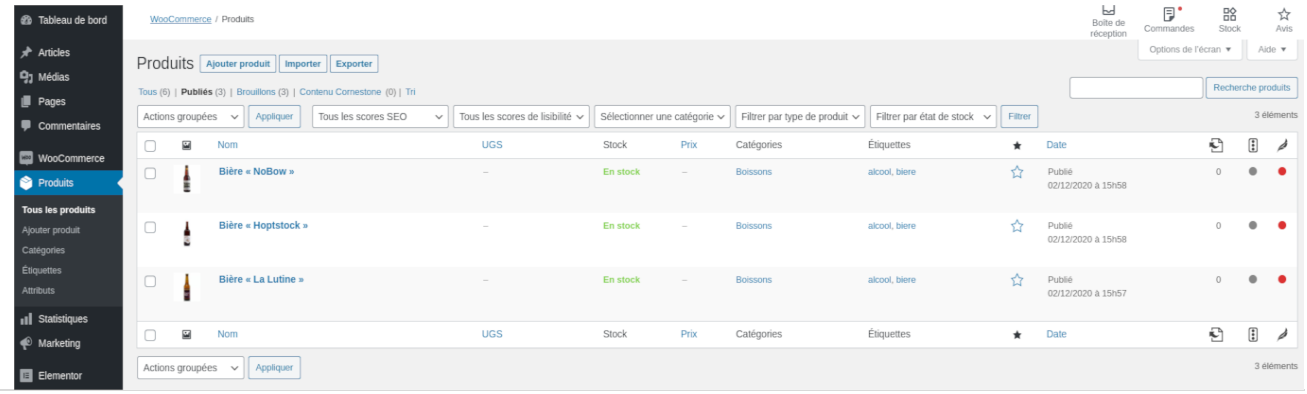
L'onglet « Produits » nous permet d'ajouter les produits qui seront mis en vente, de visualiser la liste des produits, d'ajouter des catégories et des étiquettes



Onglet WooCommerce, liste des commandes



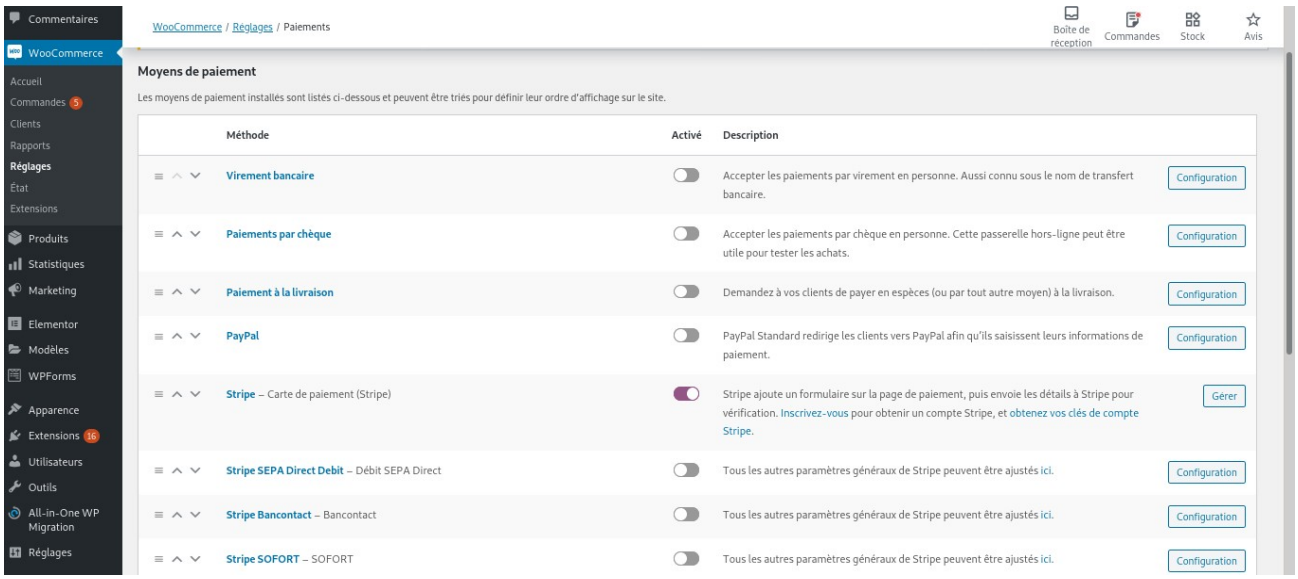
Onglet Produits, ajout d'un produit



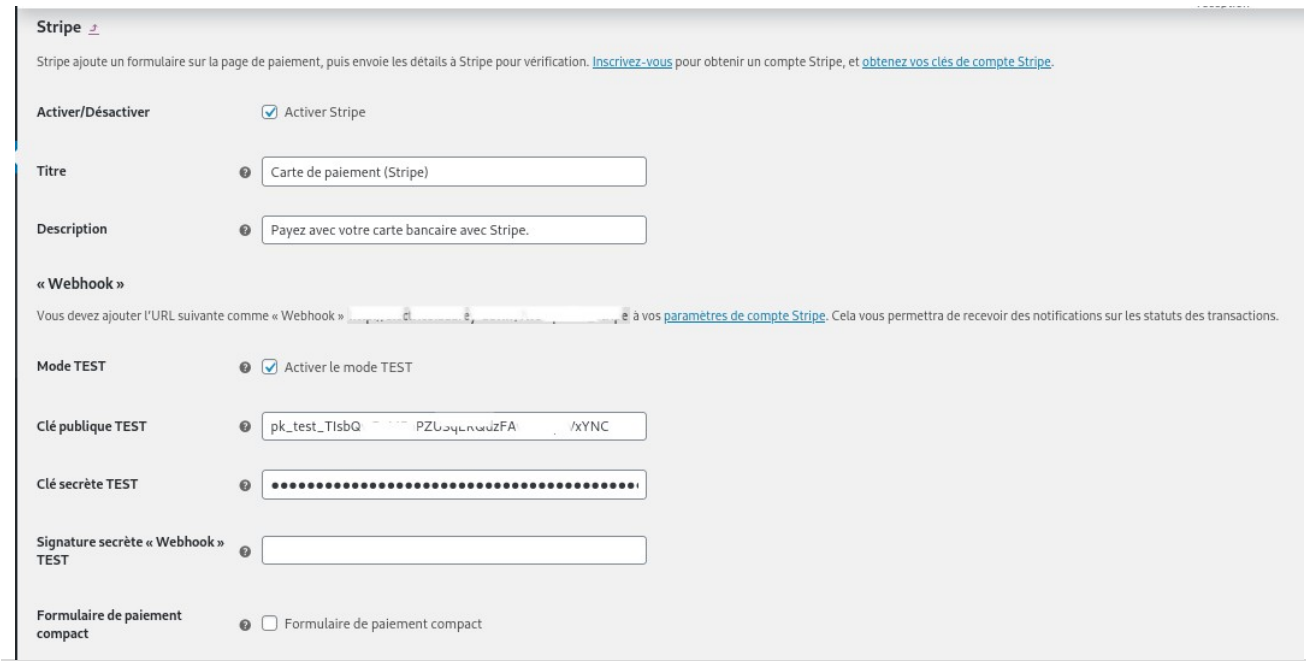
Onglet Produits, liste des produits disponibles dans la boutique

Le module de paiement Stripe, intégré à Wordpress, permet de mettre en place facilement le mode de paiement par carte bancaire.

Après enregistrement, il est possible de récupérer des clés de test, afin de simuler une vraie commande sur sa boutique.



Réglages des différents modes de paiement acceptés

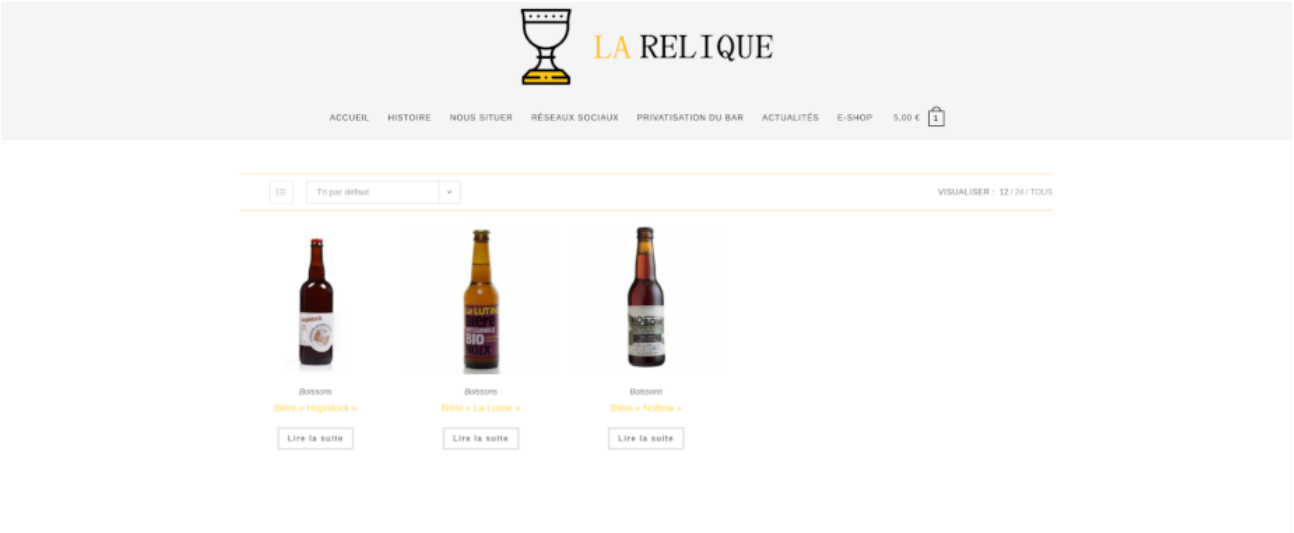


Configuration du module Stripe avec mise en place des clés de test

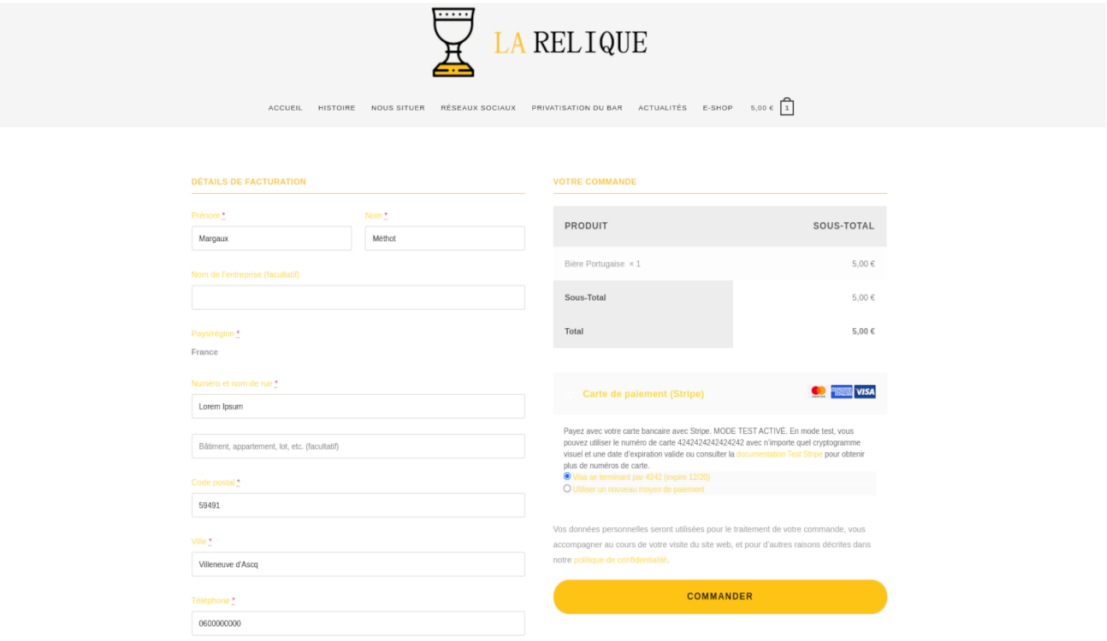
Une fois les différents réglages du module WooCommerce effectués, j'ai effectué quelques modifications de style à l'intérieur du fichier style.css afin d'avoir un rendu qui convenait à monsieur Royer.

```
/*
Theme Name: Thème enfant OceanWP
Theme URI: http://www.relique.margaux-methot.fr
Description: Thème enfant relique
Author: Margaux Méthot
Author URI: http://www.margaux-methot.fr/portfolio
Template: oceanwp
Version: 1.0.0
*/


.woocommerce .button:disabled{
    background-color: white;
    border : 1px solid gray;
    color: black;
}

.woocommerce-cart .wc-proceed-to-checkout a.checkout-button, form span input.input-text, form span textarea.input-text,
    color: black;
}

.woocommerce-cart .wc-proceed-to-checkout a.checkout-button:hover, .woocommerce-checkout #place_order:hover{
    color: white;
}

p.woocommerce-notice.woocommerce-notice--success.woocommerce-thankyou-order-received{
    text-align: center;
    color: black;
}
```

Fichier style.css



Boutique contenant des produits factices



Visuel de la page paiement, mise en place avec une carte factice

3. Avec qui avez-vous travaillé ?

J'ai travaillé toute seule sur ce projet, mais en bénéficiant du suivi de mes formateurs lorsque c'était nécessaire. Ils m'ont apporté leur aide concernant la méthodologie à employer lorsque l'on modifie un site WordPress existant, ainsi que lors de l'import du site sur mon serveur privé.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***Studi***

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du 01/07/2020 au 14/08/2020**

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ▸ *Conception d'une base de données*

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

Au cours de la formation, nous avons abordé la conception et le développement d'une base de données à l'aide de la méthode Merise. Pour mettre en application, ce que nous avions appris, il nous a été demandé de conceptualiser une base de données d'une application de multi-quizz, semblable à l'application en ligne « QuipoQuiz ».

Au sein de cette application, il existe plusieurs catégories, qui contiennent plusieurs quizz, qui eux contiennent des questions.
A la fin de chaque quizz, on obtient le score moyen obtenu par l'ensemble des autres joueurs.

2. Précisez les moyens utilisés :

J'ai commencé par élaborer le MCD, le modèle conceptuel de données, qui consiste à créer les différentes entités qui composeront le système d'information, et à établir des relations entre-elles.

Ainsi, j'obtiens les entités « Category », « Quizz », « Question » et « Result », composées de leurs propriétés.

Ensuite, j'établis les cardinalités :

- un quiz n'appartient qu'à une seule catégorie, mais une catégorie possède entre 0 et plusieurs quiz.
- un quiz comporte entre 1 à plusieurs questions, mais une question n'appartient à un seul quiz.
- un quiz possède entre 0 et plusieurs résultats, et un résultat n'appartient qu'à un seul quiz.

Une fois les cardinalités établies, je peux passer à l'élaboration du MLD, le modèle logique de données. Chaque entité devient une table, et leur identifiant devient la clé primaire de la table.

Pour les relations *,1 — *,n la clé primaire de la table côté *,n devient une clé étrangère de la table côté *,1 créant ainsi la relation entre les tables.

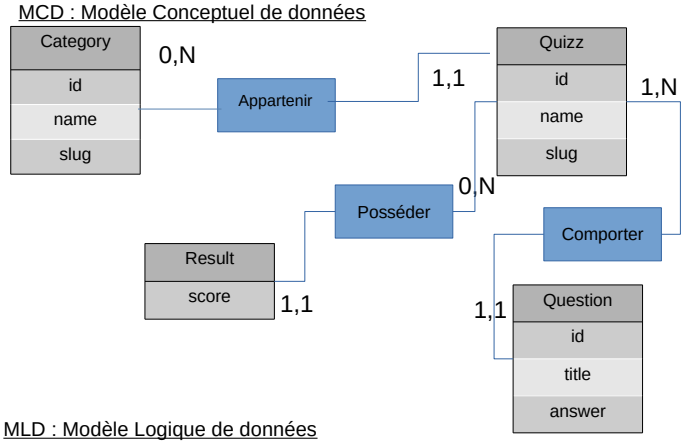
La table « Quizz » récupère le champ « category_id » clé étrangère qui fait référence à la clé primaire de la table « Category »

La table «Result» récupère le champ «quizz_id » clé étrangère qui fait référence à la clé primaire de la table «Quizz»

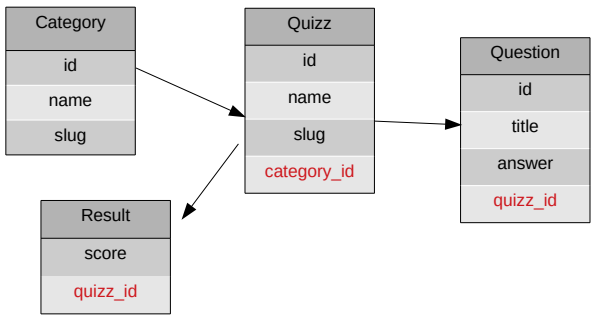
La table «Question» récupère le champ « quizz_id » clé étrangère qui fait référence à la clé primaire de la table «Question»

Maintenant, je peux élaborer le MPD, le modèle physique de données, qui consiste à préciser le type et les possibles restrictions appliquées pour chaque champ qui compose les tables.

MCD : Modèle Conceptuel de données



MLD : Modèle Logique de données



MPD : Modèle Physique de données

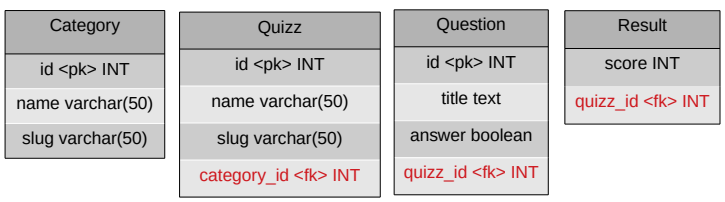


Schéma de la base de données

Une fois le MPD établi, je peux écrire le code SQL qui permettra de générer la base de données.

```sql
CREATE DATABASE IF NOT EXISTS quipoquiz

CREATE TABLE category (
    id INT UNSIGNED PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    slug VARCHAR(50) NOT NULL
)

CREATE TABLE quizz (
    id INT UNSIGNED PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    slug VARCHAR(50) NOT NULL,
    category_id INT UNSIGNED
)

ALTER TABLE quizz
ADD CONSTRAINT FK_category_quizz
FOREIGN KEY(category_id)
REFERENCES category(id)
ON DELETE RESTRICT
ON UPDATE RESTRICT;


CREATE TABLE question (
    id INT UNSIGNED PRIMARY KEY NOT NULL AUTO_INCREMENT,
    title text,
    answer BOOL NOT NULL,
    quizz_id INT UNSIGNED
)

ALTER TABLE question
ADD CONSTRAINT FK_question_quizz
FOREIGN KEY(quizz_id)
REFERENCES quizz(id)
ON DELETE RESTRICT
ON UPDATE RESTRICT;

CREATE TABLE score (
    score INT,
    quizz_id INT UNSIGNED
)

ALTER TABLE result
ADD CONSTRAINT FK_quizz_result
FOREIGN KEY(quizz_id)
REFERENCES quizz(id)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

Code SQL qui permet de générer la base de données.

Je créé d'abord les tables, en précisant le type et les restrictions de chaque colonne. Puis je modifie ces tables afin d'ajouter les contraintes de clés étrangères avec « ON DELETE » et « ON UPDATE » en « RESTRICT » afin d'éviter que la référence ne soit supprimée ou modifiée si cela entraine des données incohérentes.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule 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 18/06/2020 au 30/06/2020**

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°2 ▸ *Développement d'une API et de la base de données d'une application (Node.js)*

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

Lors de la formation, nous avons étudié node.js ainsi que le framework Express.js. Afin de mettre en pratique nos apprentissages, il nous a été demandé de développer une API REST qui permettait de faire tourner une application de quiz multi-choix. Cette API devait interroger une base de données, pensée et créée par nos soins.

2. Précisez les moyens utilisés :

J'ai procédé à l'installation d'Express au sein du projet.

```
npm install --save express
```

Pour développer la base de données, je devais utilisé le système de gestion de base de données « MongoDB » ainsi que l'ODM (« Object Document Mapping ») « Mongoose », qui permet une modélisation d'objets MongoDB sous Node.js et gère la validation des données grâce à des schémas.

J'ai donc procédé à l'installation de Mongoose :

```
npm install mongoose
```

Je peux ensuite rédiger les schémas :

- La collection « Category » possède les champs « title » et « slug », qui sont des chaînes de caractères et qui sont requises. Le champ « image » est aussi de type string, mais n'est pas requis.
- La collection « Quizz » possède les champs « title » et « slug », qui sont des chaînes de caractères et qui sont requises. Le champ « image » est aussi de type string, mais n'est pas requis. Le champ « category » est une référence à la collection « Category » et correspond à l'ID de celle-ci. Quant au champ « question », c'est un tableau de données.

models > JS category.js > ...

```
const mongoose = require("mongoose");

const CategorySchema = new mongoose.Schema({
    title : {
        type: String,
        required: [true, 'title required !']
    },
    slug : {
        type: String,
        required: [true, 'slug required !']
    },
    image : String,
})

const Category = mongoose.model('Category', CategorySchema);

module.exports = Category;
```

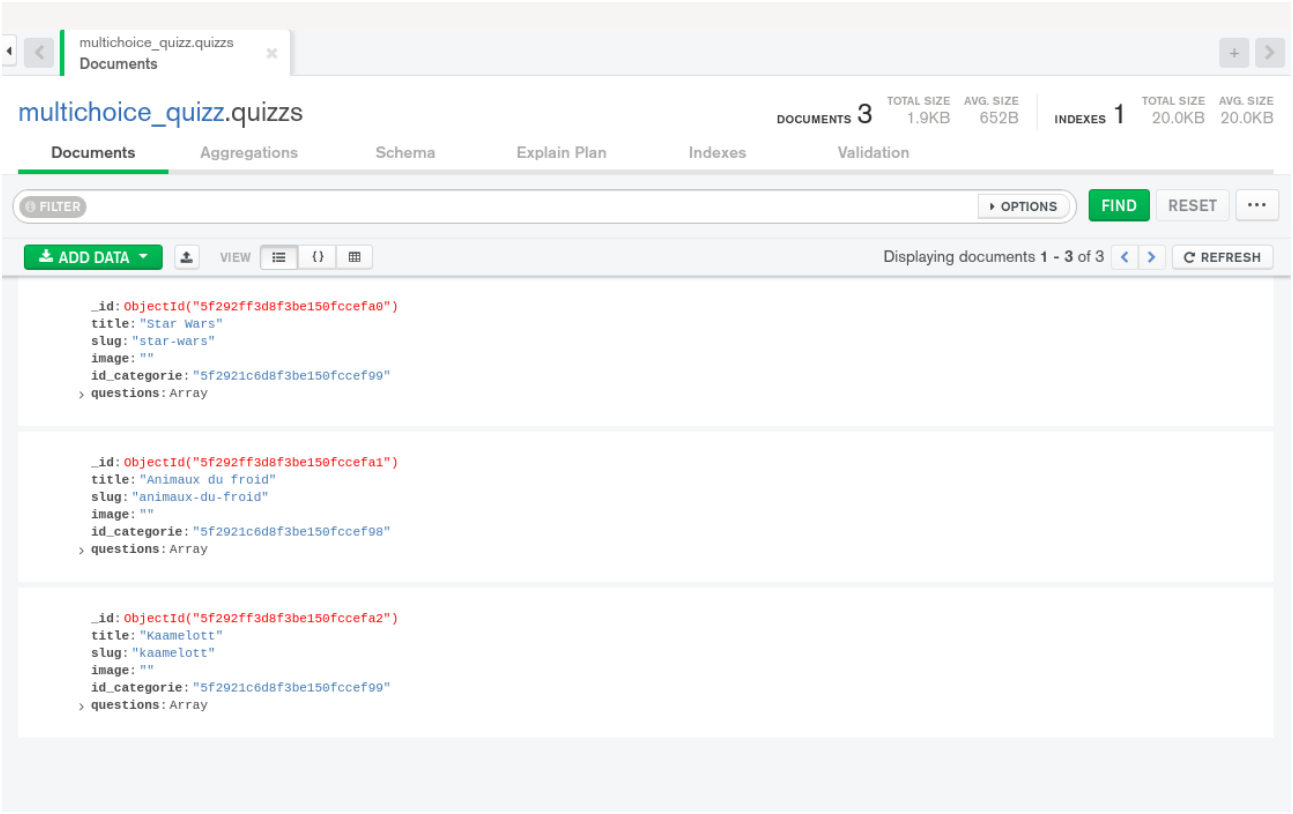
Schéma de l'entité Category

models > JS quizz.js > ...

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;


const QuizzSchema = new mongoose.Schema({
    title : {
        type: String,
        required: [true, 'title required !']
    },
    slug : {
        type: String,
        required: [true, 'slug required !']
    },
    image : String,
    category: {
        type: Schema.Types.ObjectId,
        ref: 'Category'
    },
    questions: Array
})

const Quizz = mongoose.model('Quizz', QuizzSchema);

module.exports = Quizz;
```

Schéma de l'entité Quizz

Une fois les collections créées, je peux importer des données dans la base de données, au format JSON.



Collection Quizz

Voici à quoi ressemble un item d'une collection, par exemple le quiz « Animaux du froid ».


```
_id: ObjectId("5f292ff3d8f3be150fccefa1")
title: "Animaux du froid"
slug: "animaux-du-froid"
image: ""
id_categorie: "5f2921c6d8f3be150fccef98"
questions: Array
  0: Object
    title: "Qu'étudie un arachnologue ?"
    answers: Array
      0: Object
        answer: "Rat"
        correct: "false"
      1: Object
        answer: "Serpent"
        correct: "false"
      2: Object
        answer: "Araignée"
        correct: "true"
      3: Object
        answer: "Aigle"
        correct: "false"
  1: Object
```


Item de la collection Quizz

Il possède donc un titre, un slug, le champ « image » a été laissé vide, et il possède un tableau de questions. Une question correspond à un objet au format JSON, comprenant la clé « title » ayant pour valeur une chaine de caractères, et la clé « answers », ayant pour valeur un tableau d'objets.

Chaque réponse possède la clé « correct » qui a pour valeur un booléen, qui permet de connaitre la bonne réponse et de traiter cette donnée en front.

Une fois la base de données développée, je peux passer au développement de L'API, en commençant par créer les routes qui correspondent aux différentes requêtes à la base de données.

routes > JS quizzsRouter.js > ...

```
1  const express = require('express');
2  const router = express.Router();
3  const Quizz = require('../models/quizz.js');
4
5  router.get('/', (req, res) => {
6      Quizz.find({})
7      .populate('category')
8      .exec((err, quizzs) => {
9          if(err) throw new Error(err);
10         else res.send(quizzs);
11     })
12 })
13
14 router.get('/:slug', (req, res) => {
15     Quizz.findOne({slug : req.params.slug}, (err, quizz) => {
16         if(err) res.status(400).send(err);
17         else res.send(quizz);
18     })
19 })
20
21
22 module.exports = router;
```

Router de la collection Quizz

Le router de la collection Quizz contient 2 requêtes : la première permet de récupérer l'ensemble des quizz de la collection, la seconde permet de récupérer les données d'un quiz passé en paramètre grâce à son slug. On accède au paramètre en récupérant le contenu de « req.params.slug ».

routes > JS categoriesRouter.js > ...

```
1  const express = require('express');
2  const router = express.Router();
3  const Category = require('../models/category.js');
4  const Quizz = require('../models/quizz.js');
5
6  router.get('/', (req, res) => {
7      Category.find({}, (err, categories) => {
8          if(err) throw new Error(err);
9          res.send(categories);
10     })
11 })
12
13 router.get('/:slug', (req, res) => {
14     Category.findOne({slug : req.params.slug}, (err, category) => {
15         if(err) res.status(400).send(err);
16         else Quizz.find({id_categorie : category.id}, (err, quizzs) => {
17             if(err) res.status(400).send(err);
18             else res.send(quizzs);
19         })
20     })
21 })
22
23
24 module.exports = router;
25
```

Router de la collection Category

Le router de la collection Category contient 2 requêtes : la première permet de récupérer l'ensemble des catégories de la collection, la seconde permet de récupérer tous les quizz qui appartiennent à la catégorie passée en paramètre.

Cette seconde requête est composée de 2 sous-requêtes. Dans un premier temps, on effectue une requête sur la collection Category afin de récupérer l'id de la catégorie passée en paramètre grâce au slug. Une fois l'id récupéré, il ne me reste plus qu'à effectuer une requête sur la collection Quizz afin de récupérer tous les quizz donc l'id de la catégorie à laquelle ils appartiennent, correspond à l'id de la catégorie passée initialement en paramètre.

Une fois les routers créés, il ne me reste plus qu'à initialiser le fichier App.js. Il faut importer Express, Mongoose et CORS (qui permet les requêtes multi-origines).

Je précise ensuite divers paramètres qui permettent de connecter l'API à la base de données, les routers utilisés pour chacune des collections, ainsi que leurs URL.

Ainsi, on sait que toutes les requêtes concernant les catégories commenceront par « {urlDeLaBaseDeDonnées}/categories} ».

app.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

mongoose.set('useFindAndModify', false);

const categoriesRouter = require('./routes/categoriesRouter.js');
const quizzsRouter = require('./routes/quizzsRouter.js');

const app = express();
app.use(express.json());
app.use(express.urlencoded({extended: true}));
app.use(cors());

const port = 3002;

app.use('/quizzs', quizzsRouter);
app.use('/categories', categoriesRouter);

mongoose.connect('mongodb://localhost:27017/multichoice_quizz',{
    useNewUrlParser: true,
    useUnifiedTopology: true
});

const db = mongoose.connection;

db.on('error',() => console.log('Erreur lors de la connexion'));
db.once('open',() => {
    console.log("Base de données dispo!");
    app.listen(port, () => {
        console.log(`App listening at http://localhost:${port}`);
    })
})
```

app.js

L'API est donc opérationnelle, nous pouvons démarrer notre serveur grâce à la commande :

node app.js

L'outil « Postman » permet d'effectuer des appels à l'API afin de vérifier si les réponses retournées correspondent à ce que nous souhaitons.

Voici un extrait d'un appel fait auprès de l'API : on précise la méthode utilisée (ici GET) ainsi que l'url cible (ici « localhost:3002/quizzs »). A cette URL, l'API est censée retourner l'ensemble des quizz, ce qui est bien le cas.

GET http://localhost:3002/quizzs

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
        ],
        "_id": "5f292ff3d8f3be150fccefa0",
        "title": "Star Wars",
        "slug": "star-wars",
        "image": "",
        "id_categorie": "5f2921c6d8f3be150fccef99"
    },
    {
        "questions": [
            {
                "title": "Qu'étudie un arachnologue ?",
                "answers": [
                    {
                        "answer": "Rat",
                        "correct": "false"
                    },
                    {
                        "answer": "Serpent",
                        "correct": "false"
                    },
                    {
                        "answer": "Araignée",
                        "correct": "true"
                    },
                    {
                        "answer": "Aigle",
                        "correct": "false"
                    }
                ]
            },
```


Extrait de la réponse de l'API grâce à l'outil Postman

L' API est donc fonctionnelle, et peut-être utilisée par plusieurs applications multi-quizz, telle qu'une application web ou une application mobile.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet avec le soutien de mon formateur si j'avais des questions.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***Studi***

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du 28/07/2020 au 10/08/2020**

Activité-type 2

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°3 ▸ ***Développement d'une API et de la base de données (Symfony)***

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

Lors de la formation, nous avons étudié Symfony, un framework PHP destiné à faciliter le développement d'applications.

Afin de mettre en application ce que nous avions appris, on nous a demandé de développer une API grâce au framework « API-Plateform » ainsi que la base de données, afin de gérer des groupes d'étudiants.

2. Précisez les moyens utilisés :

J'ai commencé par initialiser un nouveau projet Symfony, avec une arborescence étudiée pour le développement d'une API, à l'aide de la commande :

*composer create-project symfony/squeleton student-api 4.**

Puis j'ai installé la plateforme du serveur API :

composer require api

Après avoir créé le fichier .env.local qui contient les identifiants de connexion à la base de donnée, on crée celle-ci grâce à la commande :

php bin/console doctrine:database:create

Pour faciliter le développement, notre formateur nous a conseillé d'installer le bundle « maker-bundle » qui nous donne accès à une liste de commandes qui nous permettent d'automatiser certains développements sans avoir à écrire tout le code :

composer require symfony/maler-bundle –dev

J'ai pu ensuite commencer à créer les entités.
Nos étudiants appartiennent à des promotions et sont affectés à des projets, qui sont identifiés à l'aide de tags.
Nous devons donc créer plusieurs Entités : « Student », « SchoolYear », « Project » et « Tag ».
Nous créons ces entités Symfony grâce à la ligne de commande :

php bin/console –api-resource make:entity Student

Le « --api-resource » indique que l'entité doit être visible de l'API ». La saisie interactive nous permet de générer facilement nos entités, en précisant le nom des champs, leur type et s'il existe des relations avec d'autres Entités.

Ainsi, pour l'entité « Student » par exemple, nous générons grâce à cette commande, le fichier Student.php dans le dossier « Entity » et le fichier « StudentRepository.php » qui contiendra les requêtes dont nous aurons besoin.

Dans l'extrait de code ci-dessous, on remarque grâce aux annotations que nous avons différents champs, tels que l' « id », « firstname » ou « lastname ».
Le champ « schoolyears » quant à lui est une référence à l'entité « SchoolYear » avec laquelle elle partage une relation « Many to Many ». Cela signifie qu'une promotion peut contenir plusieurs élèves, et qu'un élève peut appartenir à plusieurs promotions.

src > Entity > Student.php

```php
<?php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;
use App\Repository\StudentRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ApiResource()
 * @ORM\Entity(repositoryClass=StudentRepository::class)
 */
class Student
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $firstname;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $lastname;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $email;

    /**
     * @ORM\ManyToMany(targetEntity=SchoolYear::class, inversedBy="students")
     */
    private $schoolyears;

    /**
     * @ORM\ManyToMany(targetEntity=Tag::class, inversedBy="students")
     */
```

Extrait du fichier Student.php

Une fois les entités créées, il faut générer le fichier de migrations et l'appliquer à la base de données. J'installe donc le bundle « doctrine-migrations-bundle » :

composer require doctrine/doctrine-migrations-bundle

Qui me permet d'utiliser la commande suivante, afin de générer le fichier de migrations :

php bin/console make:migration

Le fichier de migrations indique comment upgrade la base de données, et comment la downgrade si nécessaire. Le nom du fichier est important, car il permet de savoir quel est le dernier fichier de migration qui a été appliqué à la base de données.

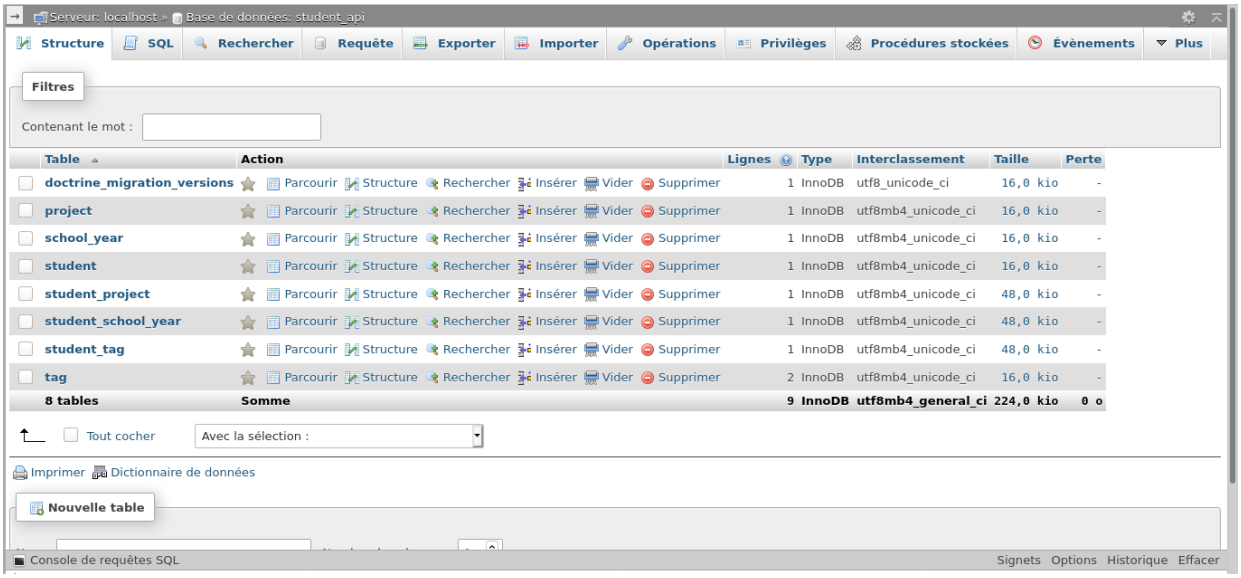
```
migrations > Version20200729101330.php
<?php

declare(strict_types=1);

namespace DoctrineMigrations;
use Doctrine\DBAL\Schema\Schema;
use Doctrine\Migrations\AbstractMigration;

final class Version20200729101330 extends AbstractMigration
{
    public function getDescription() : string
    {
        return '';
    }
    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->addSql('CREATE TABLE project (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, description LONGTEXT DEFAULT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `u
        $this->addSql('CREATE TABLE school_year (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, start_date DATE DEFAULT NULL, end_date DATE DEFAULT NULL, PRIMARY KEY(id)) DEFAULT CHARA
        $this->addSql('CREATE TABLE student (id INT AUTO_INCREMENT NOT NULL, firstname VARCHAR(255) NOT NULL, lastname VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT C
        $this->addSql('CREATE TABLE student_school_year (student_id INT NOT NULL, school_year_id INT NOT NULL, INDEX IDX_2D4AA1D9CB944F1A (student_id), INDEX IDX_2D4AA1D9D2EECC3F (school_year_id),
        $this->addSql('CREATE TABLE student_tag (student_id INT NOT NULL, tag_id INT NOT NULL, INDEX IDX_95F4B225CB944F1A (student_id), INDEX IDX_95F4B225BAD26311 (tag_id), PRIMARY KEY(student_id,
        $this->addSql('CREATE TABLE student_project (student_id INT NOT NULL, project_id INT NOT NULL, INDEX IDX_C2856516CB944F1A (student_id), INDEX IDX_C2856516166D1F9C (project_id), PRIMARY KEY(
        $this->addSql('CREATE TABLE tag (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) DEFAULT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci` ENGINE = InnoDB`
        $this->addSql('ALTER TABLE student_school_year ADD CONSTRAINT FK_2D4AA1D9CB944F1A FOREIGN KEY (student_id) REFERENCES student (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE student_school_year ADD CONSTRAINT FK_2D4AA1D9D2EECC3F FOREIGN KEY (school_year_id) REFERENCES school_year (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE student_tag ADD CONSTRAINT FK_95F4B225CB944F1A FOREIGN KEY (student_id) REFERENCES student (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE student_tag ADD CONSTRAINT FK_95F4B225BAD26311 FOREIGN KEY (tag_id) REFERENCES tag (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE student_project ADD CONSTRAINT FK_C2856516CB944F1A FOREIGN KEY (student_id) REFERENCES student (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE student_project ADD CONSTRAINT FK_C2856516166D1F9C FOREIGN KEY (project_id) REFERENCES project (id) ON DELETE CASCADE');
    }
    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->addSql('ALTER TABLE student_project DROP FOREIGN KEY FK_C2856516166D1F9C');
        $this->addSql('ALTER TABLE student_school_year DROP FOREIGN KEY FK_2D4AA1D9D2EECC3F');
        $this->addSql('ALTER TABLE student_school_year DROP FOREIGN KEY FK_2D4AA1D9CB944F1A');
        $this->addSql('ALTER TABLE student_tag DROP FOREIGN KEY FK_95F4B225CB944F1A');
        $this->addSql('ALTER TABLE student_project DROP FOREIGN KEY FK_C2856516CB944F1A');
        $this->addSql('ALTER TABLE student_tag DROP FOREIGN KEY FK_95F4B225BAD26311');
        $this->addSql('DROP TABLE project');
        $this->addSql('DROP TABLE school_year');
        $this->addSql('DROP TABLE student');
        $this->addSql('DROP TABLE student_school_year');
        $this->addSql('DROP TABLE student_tag');
        $this->addSql('DROP TABLE student_project');
        $this->addSql('DROP TABLE tag');
```

Extrait du fichier de migration

On peut maintenant appliquer le fichier de migration à la base de données, ainsi que ceux qui ont été générés mais pas encore appliqués :

php bin/console doctrine:migrations:migrate

On obtient donc notre base de données :

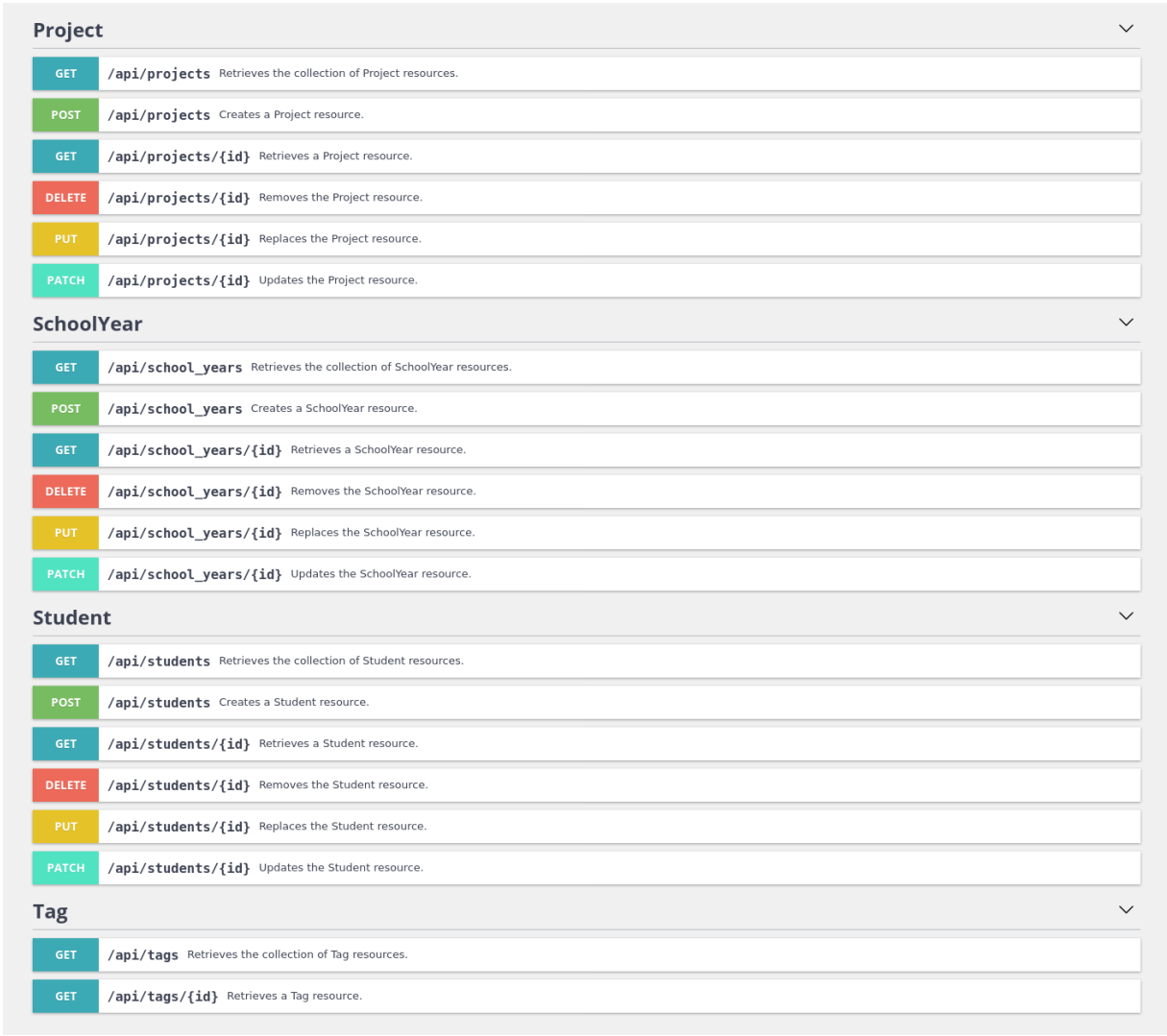


Base de données

Il est possible de n'autoriser qu'un certain type de méthode lors des requêtes sur nos entités. Nous avions comme consigne de n'autoriser que les méthodes « GET » pour l'entité « Tag ». Pour cela, dans l'entité « Tag », il est possible d'ajouter des paramètres dans l'annotation « ApiResource( ) » :

```php
<?php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;
use App\Repository\TagRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ApiResource(
 *     collectionOperations={"get"},
 *     itemOperations={"get"}
 * )
 * @ORM\Entity(repositoryClass=TagRepository::class)
 */
class Tag
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;
```

Extrait du fichier Tag.php

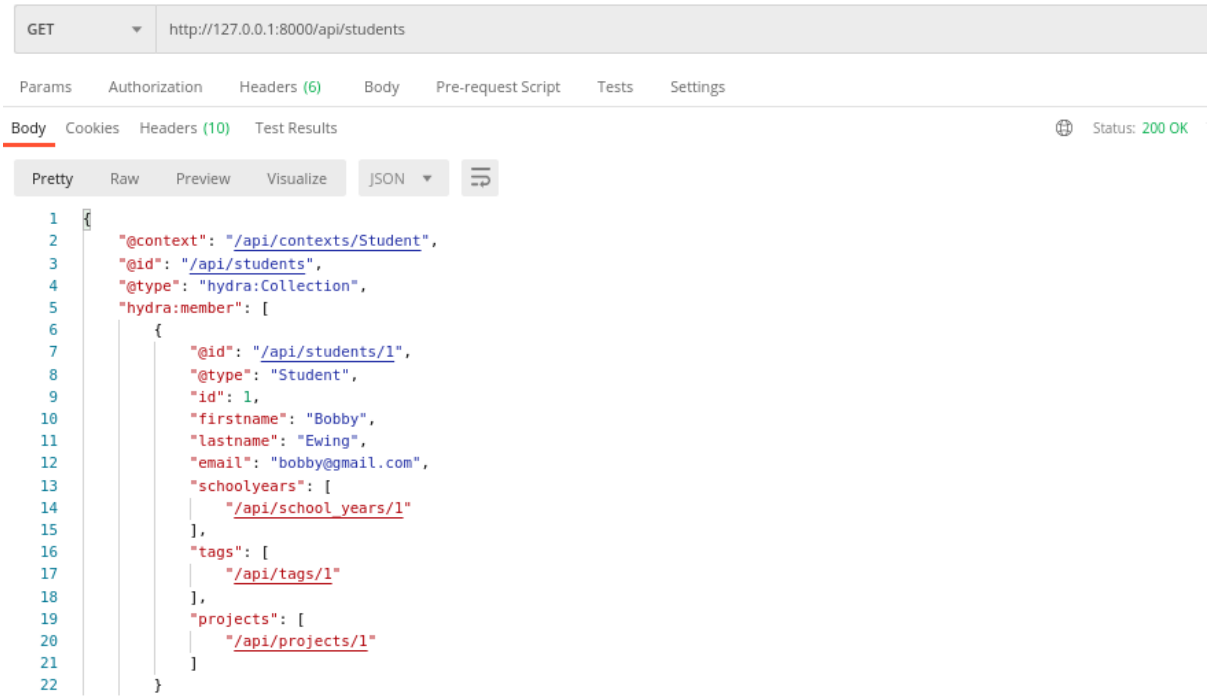


Interface d'API Platform

On constate bien que l'entité « Tag » n'est accessible qu'avec la méthode « GET ». A l'aide de « Postman », je vérifie que l'API retourne bien les données attendues. Voici la table « student » dans la base de données :

Table student

Voici la réponse de l'API lorsqu'on l'interroge avec Postman. On constate que les données renvoyées sont bien celles attendues.



Réponse de Postman

L' API est donc fonctionnelle, et peut-être utilisée par plusieurs applications, telle qu'une application web ou une application mobile.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet avec le soutien de mon formateur lorsque c'était nécessaire.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ ***Studi***

Chantier, atelier, service ▸ Cliquez ici pour taper du texte.

Période d'exercice ▸ **Du 28/07/2020 au 10/08/2020**

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Baccalauréat Scientifique spécialité mathématiques	**Lycée Sainte Clothilde - Lille**	**10/06/2017**
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] ..... ,

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 ..... le .....

pour faire valoir ce que de droit.

Signature :

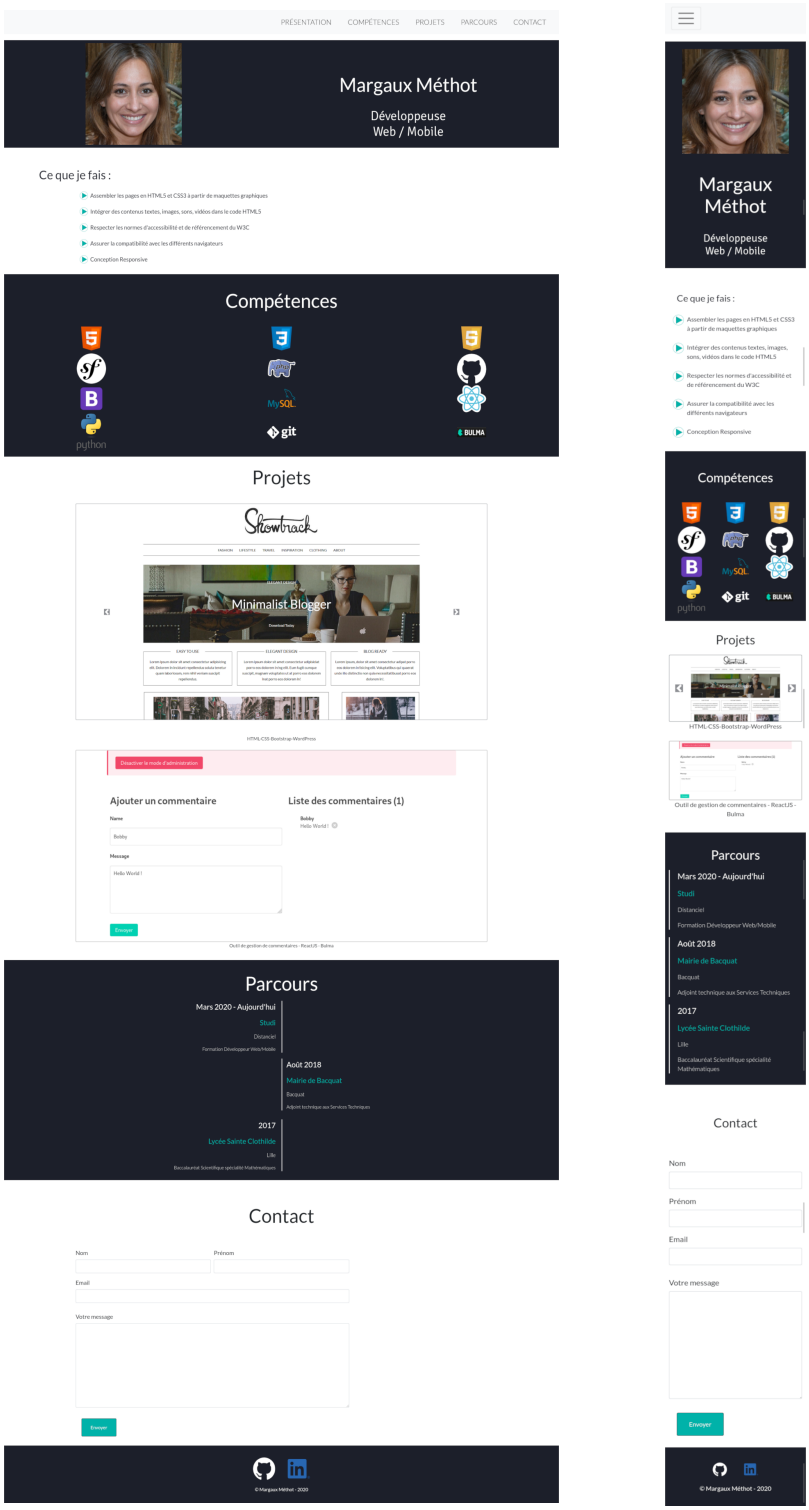
Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

Annexe 1 – Mon Curriculum Vitae



Vue ordinateur et vue mobile

Annexe 2 – Interface de gestion de commentaires

Mode administration désactivé

Mode administration activé