



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

<i>Nom de naissance</i>	- M
<i>Nom d'usage</i>	- M
<i>Prénom</i>	- Marie
<i>Adresse</i>	- X, xxxx 00000 XXXXXXXX

Titre professionnel visé

Développeur 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>



DOSSIER PROFESSIONNEL (DP)

Sommaire

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.

p. 5

- CP 1 Maquetter une application. p. 5
- CP 2 Réaliser une interface statique et adaptable. p. 13
- CP 3 Développer une interface utilisateur web dynamique. p. 19
- CP 4 Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce. p. /

Intitulé de l'activité-type n° 2 : Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

p. 25

- CP 5 Créer une base de données. p. 25
- CP 6 Développer les composants d'accès aux données. p. 29
- CP 7 Développer la partie back-end d'une application web ou web mobile. p. 35
- CP 8 Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce . p. /

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

p. 42

Déclaration sur l'honneur

p. 43

Documents illustrant la pratique professionnelle (*facultatif*)

p. 44

Annexes (*Si le RC le prévoit*)

p. 45

EXEMPLES DE PRATIQUE

PROFESSIONNELLE



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

CP 1 - Maquetter une application.

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

Lors de la réalisation d'un exercice de l'école, il m'a été demandé de réaliser une partie des documents de conception d'une application de recettes de cuisine. J'ai conçu les *wireframes* et les *user stories* à partir du cahier des charges fourni. Ce dernier était simple et donnait le contexte et les spécifications fonctionnelles du Produit Minimum Viable (MVP).

User Stories

Les *user stories* sont les scénarios utilisateurs. Ce sont des descriptions courtes et simples d'attentes d'un utilisateur de l'application. Elles sont écrites de la manière suivante : "En tant que [qui], je veux [quoi] afin de [pourquoi]." Chaque *user story* est indépendante des autres et décrit une fonctionnalité. Les *user stories* permettent de s'assurer que les fonctionnalités développées répondent aux attentes des utilisateurs et que les demandes sont bien comprises.

Extrait de user stories

<u>En tant que</u>	<u>Je veux</u>	<u>Afin de</u>
Utilisateur	Accéder à la page d'accueil	Voir une recette au hasard, les dernières nouveautés et les meilleures recettes
Utilisateur	Accéder à la page des recettes de saison	Consulter toutes les recettes de saison
Utilisateur	Accéder au formulaire de soumission d'une recette	Soumettre une recette que j'ai élaborée
Utilisateur	Accéder à la page d'une recette	Voir la liste des ingrédients nécessaires et la préparation

Utilisateur	Cliquer sur le logo	D'être redirigé vers la page d'accueil
-------------	---------------------	--

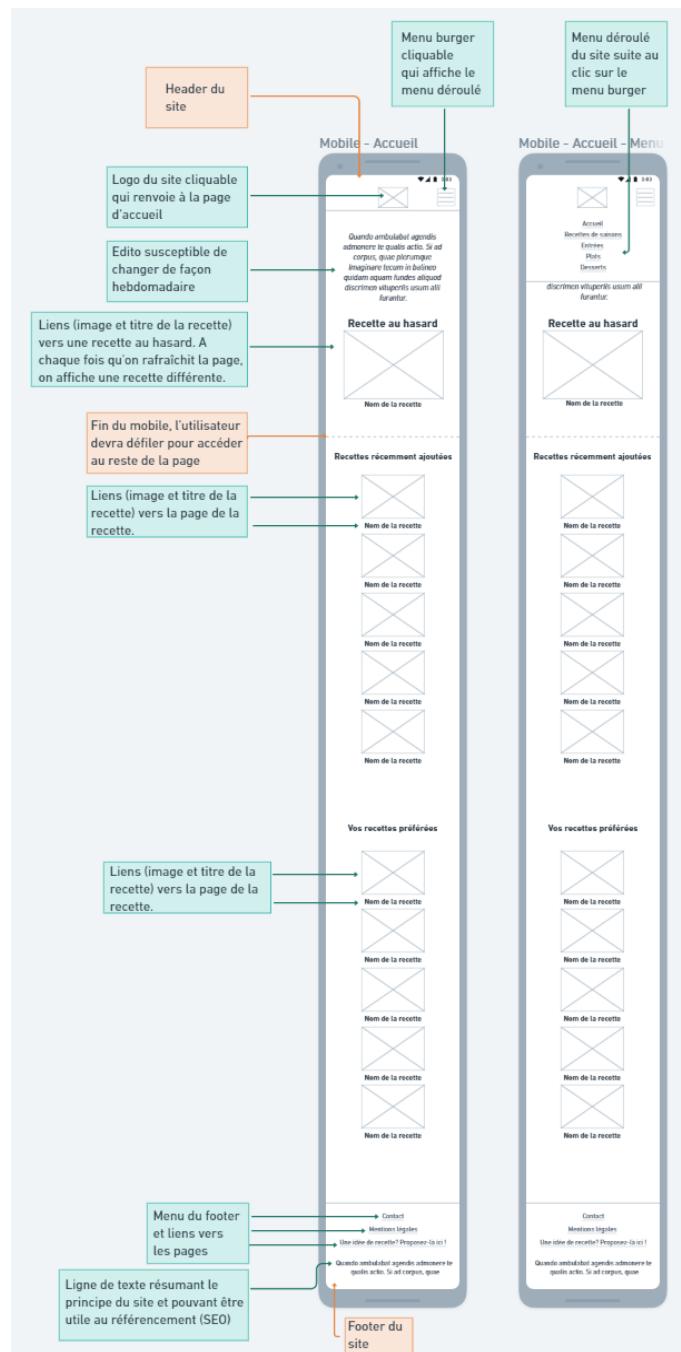
Wireframes

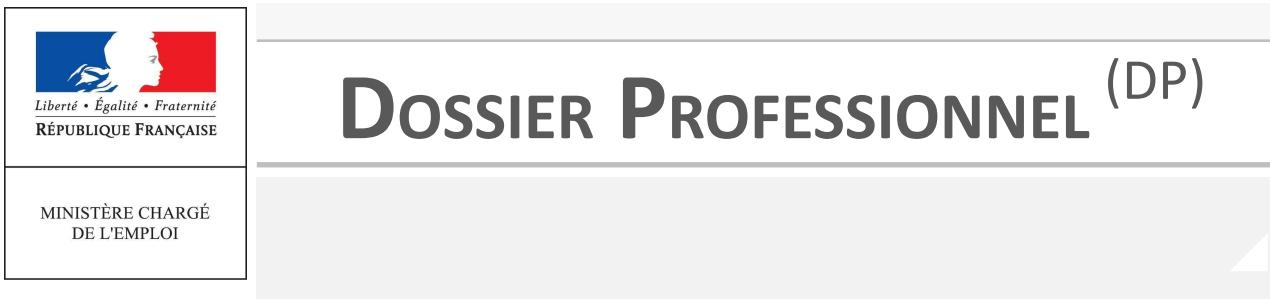
Les *wireframes* sont des maquettes en nuances de gris, légendées et très simples d'une application. Elles sont dénuées de tout choix graphique et servent à montrer de façon schématique la structure des pages.

Pour ce projet, j'ai utilisé l'approche *mobile-first* afin de réaliser les *wireframes*. En effet, la majorité des usages se fait aujourd'hui avec des smartphones et tablettes.

Wireframes au format mobile

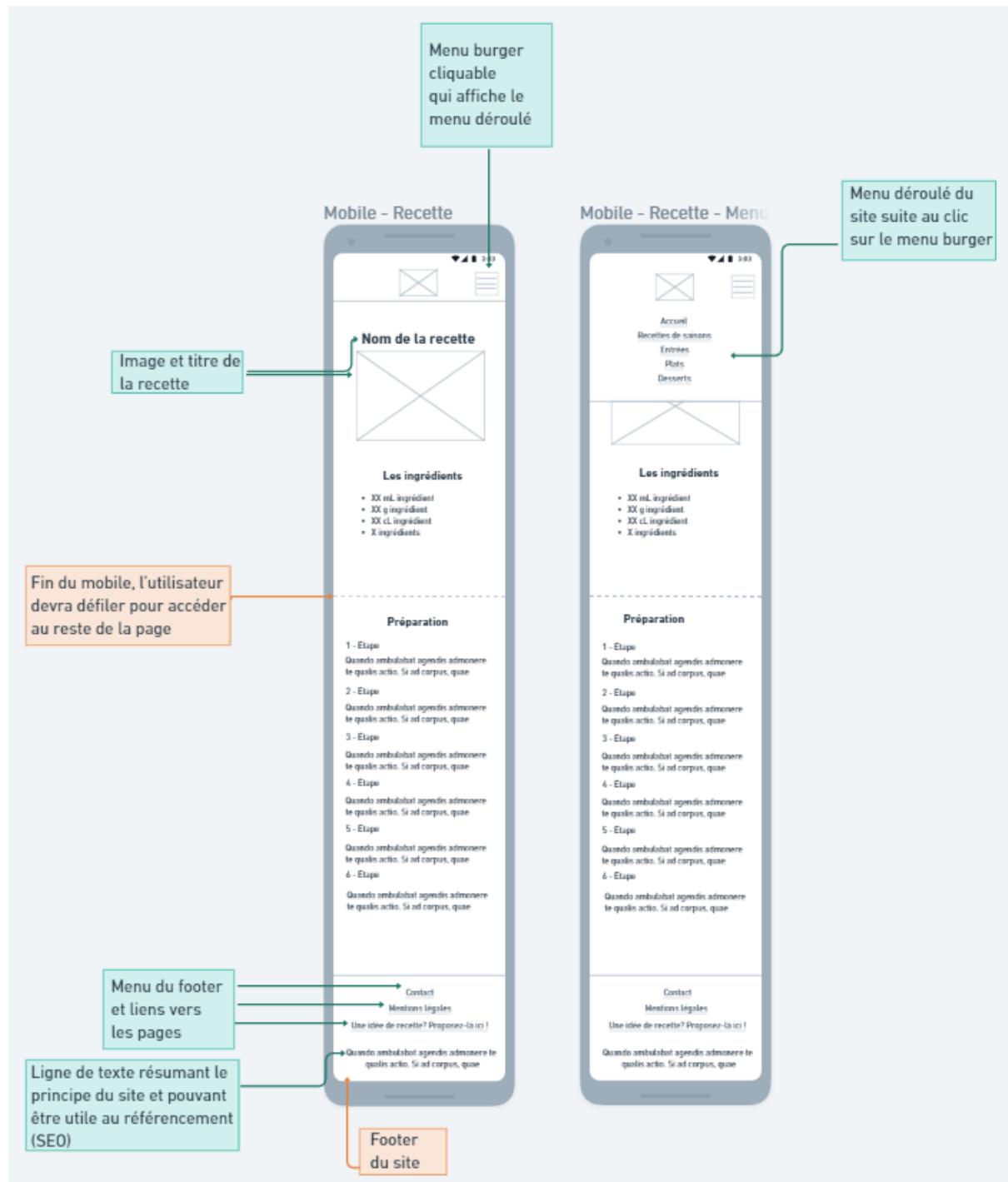
Page d'accueil





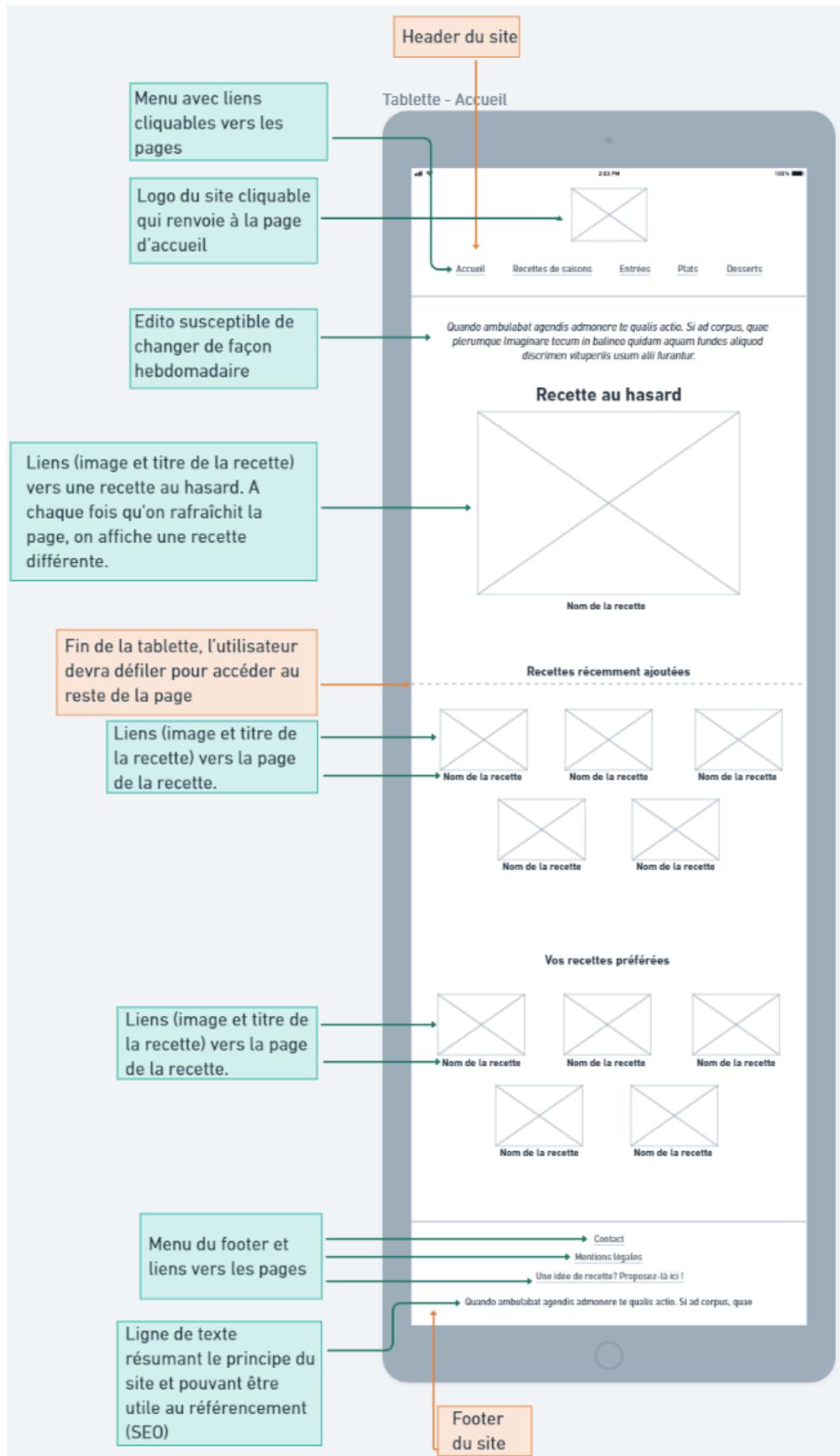
DOSSIER PROFESSIONNEL (DP)

Page de recette



Wireframes au format tablette

Page d'accueil

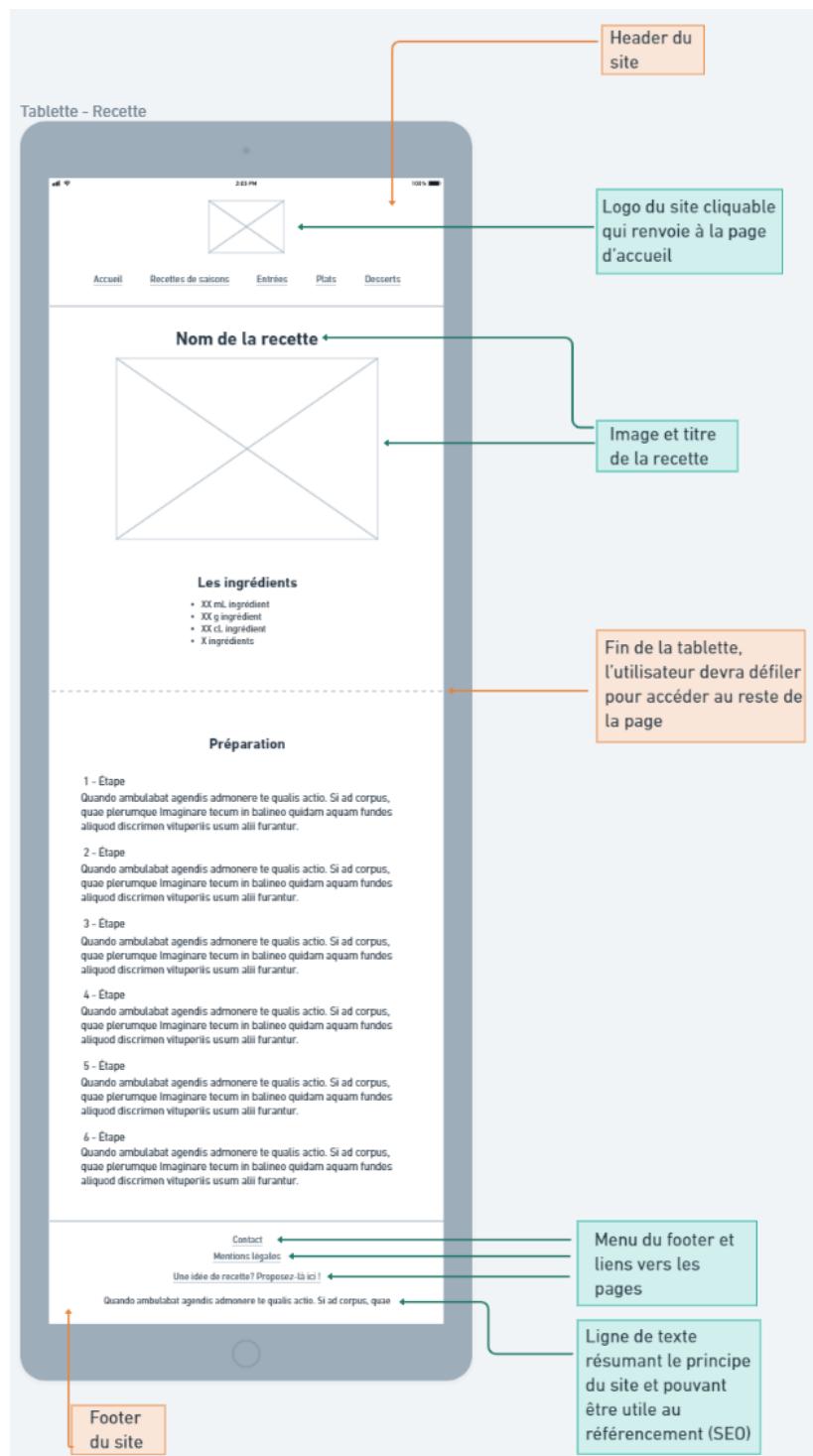




MINISTÈRE CHARGÉ
DE L'EMPLOI

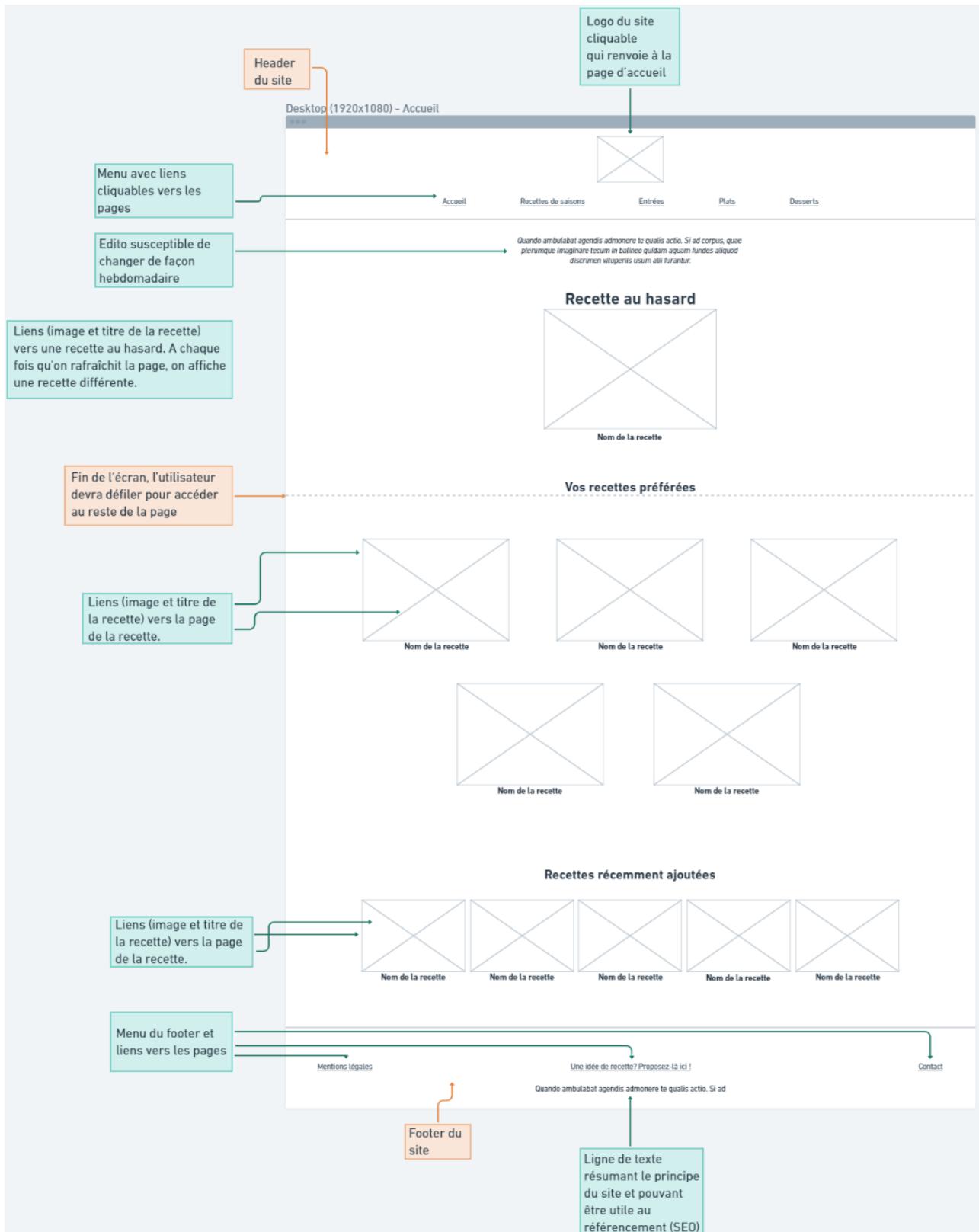
DOSSIER PROFESSIONNEL (DP)

Page de recette



Wireframes au format desktop

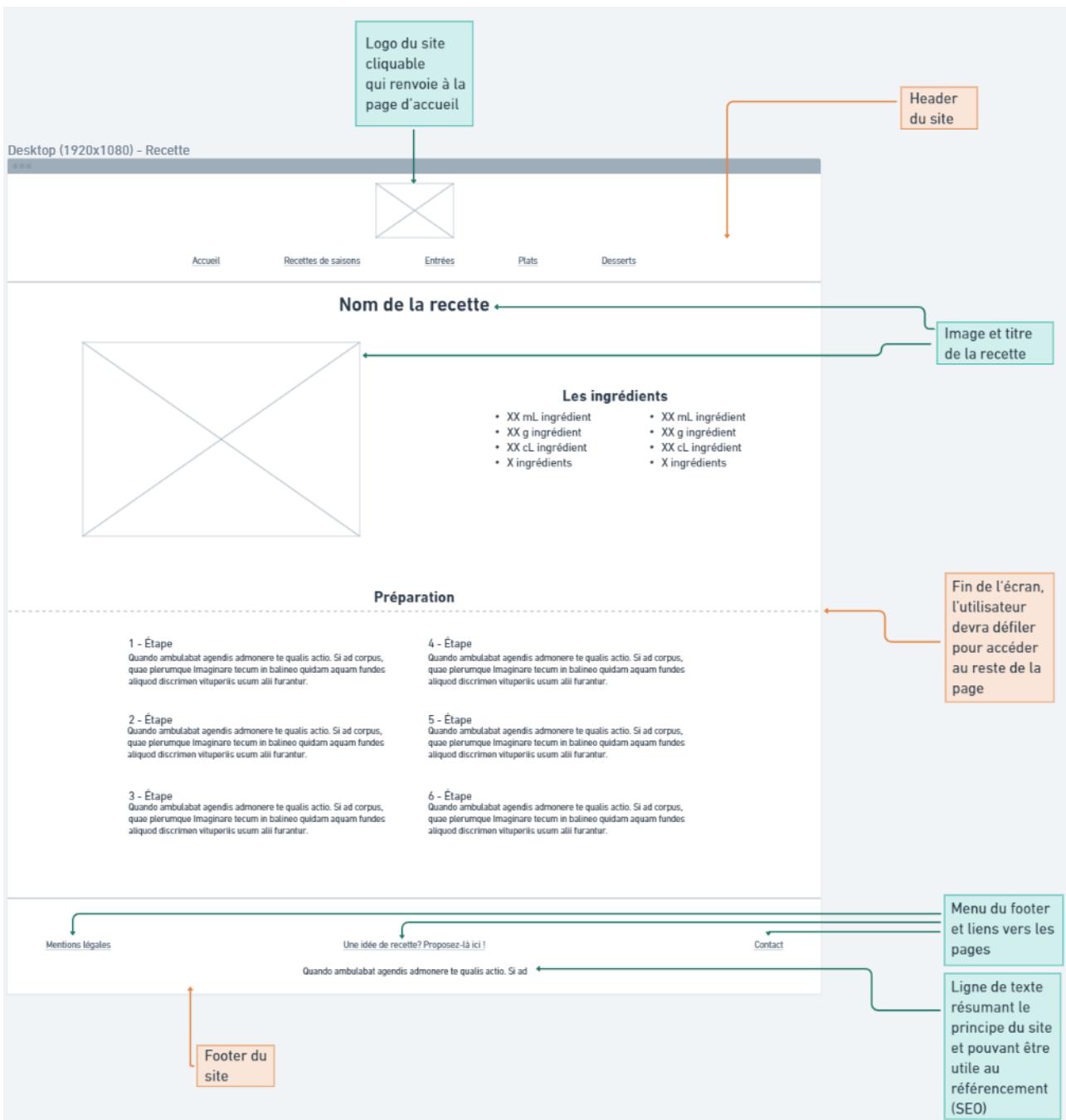
Page d'accueil





DOSSIER PROFESSIONNEL (DP)

Page de recette



2. Précisez les moyens utilisés :

Pour la réalisation des *user stories*, j'ai utilisé le logiciel de traitement de texte *Word* de Microsoft.

Pour la réalisation des *wireframes*, j'ai utilisé la plateforme de collaboration visuelle en ligne *Whimsical* (<https://whimsical.com/>).

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *École O'Clock*

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : 27/03/2023 au : 11/09/2023

5. Informations complémentaires (facultatif)

Bien qu'il fasse partie des documents de conception du projet, il ne m'était pas demandé de rédiger un cahier des charges pour cet exercice. Sa rédaction aurait inclus, en plus du contexte et de son MVP, les *user stories*, les *wireframes*, les spécifications techniques, le public ciblé, les rôles des utilisateurs, l'arborescence du site, les différentes routes et les évolutions possibles.



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

CP 2 - Réaliser une interface web statique et adaptable.

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

L'école nous a demandé de réaliser un journal d'étudiants (projet fictif dans le but de nous exercer). Il nous a été donné une image du rendu final attendu, avec la consigne d'intégrer la maquette de manière statique et de rendre le site responsive. Les éléments fournis, autres que la maquette, étaient des images (avatars, photo, logo au format SVG) et les codes hexadécimaux des couleurs.

Maquette du rendu attendu :

The image is a composite of three parts. On the left, a woman with glasses and long hair is sitting at a wooden table in a cafe, looking at a laptop screen that shows some code. On the right, there is a screenshot of a news website. The top part of the screenshot shows a woman sitting at a desk, similar to the one in the photo. Below this, the website has a header with 'O'CLOCK STUDENTS NEWS' and a sub-header 'Latest news from our students'. It features a large image of a person working on a laptop. Below the image, there is some placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque scelerisque suscipit nibh quis porttitor. Integer facilis mi urna, a pulvinar quam adipiscing ut. Vivamus vel vestibulum mauris.' At the bottom of the website screenshot, there are links for 'Plan du site', 'Mentions légales', and 'Contact'.

Latest news

NEWS	TEAM
<p>Lorem ipsum dolor sit amet</p> <p>John Marchill le 21 Juillet 2019</p> <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque scelerisque suscipit nibh quis porttitor. Voilà, je tenais à le dire.</p> <p>Continue reading</p>	<p>Le pair programming pour les n00bs</p> <p>Paul Darson le 2 Août 2019</p> <p> Le pair programming pour les dévs, c'est un peu comme certains fromages : c'est mieux à deux.</p> <p>Continue reading</p>
WORK	TEAM
<p>Pourquoi j'aime PHP</p> <p>Bruce Ékout le 7 Août 2019</p> <p> PHP est le langage le plus répandu dans le Web. Je vous propose de faire le tour de toutes ces petites choses simples qui lui donnent une saveur si unique.</p> <p>Continue reading</p>	<p>La situation de scribe</p> <p>Otis le 5 Septembre 2019</p> <p> Mais, vous savez, moi je ne crois pas qu'il y ait de bonne ou de mauvaise situation. Moi, si je devais résumer ma vie aujourd'hui avec vous, je dirais que c'est d'abord des rencontres, des gens qui m'ont tendu la main, peut-être à un moment où je ne pouvais pas, où j'étais seul chez moi. Et c'est assez curieux de se dire que les hasards, les rencontres forgent une destinée...</p>

Pour ce projet, j'ai créé une version mobile en utilisant l'approche *mobile-first*, bien que l'exercice ne le demandait pas.

J'ai donc commencé par coder la partie HTML, puis le CSS en commençant d'abord par la version mobile puis celle du desktop.

Pour pouvoir avoir une structure de page différente en fonction de ces deux types d'affichages, j'ai

utilisé les *media queries* avec les *breakpoints* “*max-width: 959px*” (tous les écrans dont la résolution est de maximum 959 pixels, qui correspond au format mobile) et “*min-width: 960px*” (tous les écrans dont la résolution est de minimum 960 pixels, qui correspond au format desktop).

Je suis partie du principe que la tablette aurait le même affichage qu'un mobile.

La page est divisée en deux parties : la partie avec la photo et celle avec les articles.

Pour la réaliser, j'ai utilisé différentes balises HTML *<header>*, *<nav>*, *<aside>*, *<main>*, *<section>*, *<article>*, *<footer>* afin d'avoir un code avec une bonne sémantique.

La mise en forme et la disposition des différentes pages ont été faites avec *Flexbox*.

La principale différence entre les formats d'affichages est, quand le site est vu depuis un smartphone, *flex-direction: column* (les articles sont les uns au-dessus des autres) et au format desktop, *flex-wrap: wrap* (les éléments sont les uns à côté des autres et vont à la ligne en-dessous lorsque le bord est atteint).

Les screenshots de la page d'accueil et de la page d'un article, aux formats mobile et desktop, sont fournis en annexe 1.

Structure de la partie de la page avec la photo :

```
● ● ●

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>oNews</title>
    <link rel="stylesheet" href="../css/reboot.css">
    <link rel="stylesheet" href="../css/style.css">
    <link rel="icon" type="image/x-icon" href="../images/onews.svg">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <div class="wrapper">
        <div class="left-side">
            <header>
                <a href="../html/index.html"></a>
                <h1>0'clock <br> Students <br> News</h1>
            </header>
            <aside>
                <h2>Latest news <small>from our students</small></h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras metus mi, euismod in vulputate sed, ultrices non elit. Nam ornare eu ipsum condimentum tempus. Duis placerat augue id ex dapibus suscipit. Aliquam pretium est ut elementum convallis. Suspendisse gravida hendrerit nunc sit amet scelerisque.</p>
            </aside>
            <footer>
                <nav>
                    <ul>
                        <li><a href="../html/plan_de_site.html">Plan du site</a></li>
                        <li><a href="../html/mentions_legales.html">Mentions légales</a></li>
                        <li><a href="../html/contact.html">Contact</a></li>
                    </ul>
                </nav>
            </footer>
        </div>
    </div>
```



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Structure de la partie de la page avec les articles :



```
<main class="right-side">
    <header>
        <h2>Latest news</h2>
    </header>
    <section id="articles">
        <article class="news">
            <a class="news-category news-category__news" href="#">News</a>

            <h3 class="news-title">Lorem ipsum dolor sit amet</h3>

            <div class="news-details">
                
                <span>John Marchill</span>
                <time datetime="2019-07-21">le 21 Juillet 2019</time>
            </div>

            <p class="news-content">
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus leo ante,
                tempor non augue semper, pellentesque blandit ligula. Fusce eu nisl eu justo aliquet imperdiet sed ac
                augue.
            </p>

            <a class="news-link" href="#">Continue reading</a>
        </article>
        <article class="news">
            <a class="news-category news-category__team" href="#">Team</a>

            <h3 class="news-title">Le pair programming pour les n00bs</h3>

            <div class="news-details">
                
                <span>Paul Darson</span>
                <time datetime="2019-08-02">le 2 Août 2019</time>
            </div>

            <p class="news-content">
                Le pair programming pour les devs, c'est un peu comme certains fromages : c'est
                mieux à deux.
            </p>

            <a class="news-link" href="#">Continue reading</a>
        </article>
    <!-- There are 4 more articles written with the same structure. Only the datas change in the <a
    class="news-category news-category__XXX", the <h3>, the <img> and its alt="", the name between the <span>
    and the date in <time> in the <div class="news-details">, the content of the <p class="news-content" and
    finally the path in the <a class="news-link" href="XXX">.It was cut for the screenshot. -->
    </section>
</main>
</div>
</body>
</html>
```

Extrait du CSS pour la partie avec la photo:

Format mobile :

```
/* LEFT SIDE LAYOUT */

/* the left side uses 60% of the screen's height /
@see https://developer.mozilla.org/en-US/docs/
Learn/CSS/Building_blocks/Values_and_units */

.wrapper .left-side {
    height: 60vh;

    display: flex;
    flex-direction: column;
    justify-content: space-between;

    background-image: url('../images/nicole.jpg');
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;

    padding: 2em;

    text-shadow: 0.1em 0.1em 0.3em black;
}

.left-side header {
    display: flex;
    justify-content: center;
    align-items: center;

    text-transform: uppercase;
    color: #FFFFFF;
    line-height: 1.5em;
}

.logo {
    width: 3.5em;

    margin-right: 1em;
}

.left-side aside {
    padding: 1em 2em;

    background-color: rgba(175, 174, 174, 0.48);

    color: #FFFFFF;
    text-align: center;
    font-size: large;
}
```

Format desktop :

```
/* LEFT SIDE LAYOUT */

/* the left side uses half of the screen's width /
@see https://developer.mozilla.org/en-US/docs/
Learn/CSS/Building_blocks/Values_and_units */
/* 16px = 1em , 1px = 0.0625em , 8px = 0.5em */

.left-side {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    width: 50vw;

    background-image: url('../images/nicole.jpg');
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;

    padding: 2em;

    text-shadow: 0.1em 0.1em 0.3em black;
}

.left-side header {
    display: flex;
    align-items: start;

    text-transform: uppercase;
    color: #FFFFFF;
    line-height: 2em;
}

.logo {
    width: 5em;
    margin-right: 1em;
}

.left-side aside {
    color: #FFFFFF;
    background-color: rgba(255, 255, 255, .3);
    padding: 1em 2em;
    text-align: right;
}
```



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

Extrait du CSS pour la partie avec les articles:

Format mobile :

```
/* RIGHT SIDE LAYOUT */

/* the right side uses 40% of the screen's height /
@see https://developer.mozilla.org/en-US/docs/
Learn/CSS/Building_blocks/Values_and_units */

.right-side {
    height: 40vh;
}

.right-side header {
    background-color: #f9f9f9;

    text-align: center;
    font-size: large;
    padding: 0.5em 1em;

    border-bottom: 0.06em solid #eaeaea;
}

.news {
    padding: 1em;

    border-top: 0.06em solid #eaeaea;
    border-right: 0.06em solid #eaeaea;
}

/* Article page */

.right__title {
    margin: 0;
    padding: 0.5em 1em;

    background-color: #f9f9f9;

    text-align: center;
    font-size: large;

    border-bottom: 0.06em solid #eaeaea;
}

.posts {
    margin: 0em;
    padding: 2em 1.5em;

    border-top: 0.06em solid #eaeaea;
    border-right: 0.06em solid #eaeaea;
}

.post__meta {
    margin-top: 1em;
}

.a.post__link:hover {
    color: #1602ac;
    font-weight: bolder;
    text-decoration: none;
    transition: 0.2s;
}
```

Format desktop :

```
/* RIGHT SIDE LAYOUT */

.right-side {
    width: 50vw;
    overflow-y: scroll;
}

.right-side header {
    position: sticky;
    top: 0;

    background-color: #f9f9f9;
    font-size: larger;
    padding: 0.5em 1em;

    border-bottom: 0.06em solid #eaeaea;
}

#articles {
    display: flex;
    flex-wrap: wrap;
}

#articles p {
    padding-top: 1em;
}

.news {
    width: 50%;
    padding: 2em;
    border-top: 0.06em solid #eaeaea;
    border-right: 0.06em solid #eaeaea;
}

/* Article page */

.right__title {
    margin: 0;
    padding: 0.5em 1em;

    background-color: #f9f9f9;

    text-align: center;
    font-size: larger;

    border-bottom: 0.06em solid #eaeaea;
}

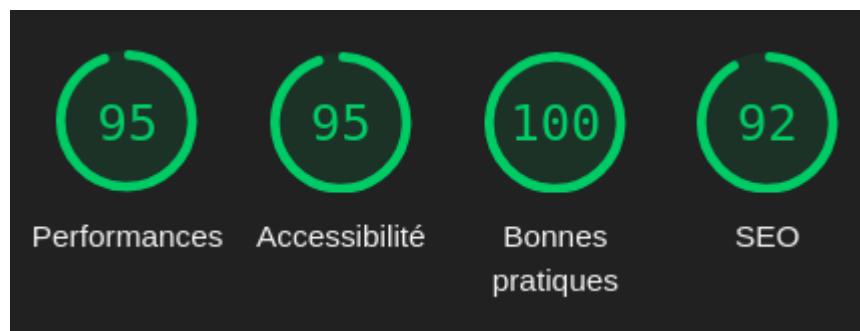
.posts {
    margin: 0em;
    padding: 2em 1.5em;

    border-top: 0.06em solid #eaeaea;
    border-right: 0.06em solid #eaeaea;
}

.post__meta {
    margin-top: 1em;
}

.a.post__link:hover {
    color: #1602ac;
    font-weight: bolder;
    text-decoration: none;
    transition: 0.2s;
}
```

Pour m'assurer que le site soit bien référencé et qu'il soit accessible au plus grand nombre, j'ai utilisé l'outil *Lighthouse* de mon navigateur.



2. Précisez les moyens utilisés :

J'ai utilisé l'Environnement de Développement Intégré (IDE) Visual Studio Code. J'ai également utilisé DevTools (outils disponibles sur les navigateurs via la touche F12) pour m'aider à peaufiner la mise en page.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *École O'Clock*

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : 27/03/2023 au : 11/09/2023

5. Informations complémentaires (facultatif)



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

Activité-type 1

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

CP 3 - Développer une interface utilisateur web dynamique.

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

Durant la formation, nous avons découvert Javascript par la création d'un site de voyage et par une liste interactive de tâches .

1 - Site de voyage :

The screenshot shows a travel website with a header "Trip O'dvisor" and navigation links for "Voyages", "Restaurants", "Hôtels", "Vols", and "Croisières". Below the header is a large image of a city skyline. A section titled "Partir en voyage" displays three travel destinations: "Partir à l'aventure" (desert landscape), "Partir à l'étranger" (city skyline), and "Promenade dans la nature" (forest landscape). Each destination has a brief description and a "Découvrir" button.

The screenshot shows the same travel website but in dark mode. The overall appearance is darker, with the city skyline image being the most prominent visual element. The "Partir en voyage" section and its three destination cards are visible, maintaining the same layout and content as the light mode version.

Lorsque l'utilisateur arrive sur le site, il peut choisir de passer le thème du site en mode sombre.



```
const app = {
  init: function()
  {
    slider.init();
    newsletter.init();
    theme.init();
    review.init();
    errorManager.init();
  }
}

document.addEventListener('DOMContentLoaded', app.init);
```

Le module du thème (*theme.js*) est initialisé au chargement du DOM grâce à *app.js* qui regroupe les *init* des différents modules du projet. En effet, nous allons utiliser le DOM (*Document Object Model*) pour modifier le HTML.

Pour pouvoir changer la couleur du thème, je place un écouteur d'événement sur le bouton dédié.

Pour cela, je repère le bouton et son id dans le fichier HTML.

index.html :

```
<button id="theme-switch" class="btn" type="button" aria-label="Changer le thème">
    <i class="icon-moon"></i>
</button>
```

Puis je récupère ce bouton grâce au *querySelector* qui me permet de récupérer l'*id* “*theme-switch*” (pour spécifier qu’on récupère un *id* il faut mettre “#” avant et pour une classe on met un “.”; comme pour le CSS)

theme.js :

```
● ● ●

const theme = {
    init: function()
    {
        console.log('initialisation du module theme');
        // theme dark / light button
        const switchThemeBtn = document.querySelector('#theme-switch');
        switchThemeBtn.addEventListener('click', theme.handleSwitchBtnClick)
    },
}
```

Au clic sur le bouton *<button id =”theme-switch”...>*, la méthode *handleSwitchBtnClick* est appelée.

```
● ● ●

handleSwitchBtnClick: function () {
    console.log('click sur le bouton du theme');

    // if the body element has a class 'theme-dark'
    // then toggle removes it, else toggle adds it
    const bodyElement = document.querySelector('body');
    bodyElement.classList.toggle('theme-dark');
```

Elle ajoute (ou retire) à l'aide du *toggle* la classe “*theme-dark*” à la balise *<body>*.

La classe “*theme-dark*” est ajoutée à la balise *<body>* dans *index.html* (ou retirée si elle était déjà présente). *<body class="theme-dark">*

```
/* dark mode */
.theme-dark {
    color: white;
    background-color: rgb(22, 22, 22);
    --border-normal: 1px solid rgba(255, 255, 255, .2);
    --color-bg: rgb(15, 15, 15);
    --color-opacity: rgba(255, 255, 255, .1);}
```

Dans *styles.css*, un style propre au thème sombre est activé lorsque la classe “*theme-dark*” est ajoutée et est désactivé lorsque cette dernière est retirée.



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

Pour une meilleure expérience utilisateur, le choix d'utiliser (ou non) le thème sombre est stocké dans le stockage local.

Si la balise <body> a une classe "theme-dark", alors on stocke un tableau avec la clé "theme-dark" et sa valeur "yes". S'il était déjà présent et que l'utilisateur repasse en thème clair, le tableau est retiré.

```
// if the user wants the dark theme,  
if (bodyElement.classList.contains('theme-dark')) {  
    // the information is stored in the local Storage of the browser  
    localStorage.setItem('theme-dark', 'yes');  
}  
else {  
    // nothing is stored in localStorage  
    localStorage.removeItem('theme-dark');  
}
```

Cookie dans stockage local :

theme-dark	yes
------------	-----

À chaque nouvelle visite de l'utilisateur, on vérifie la présence de "theme-dark : yes" dans le stockage local. Cela se produit lors du chargement de l'application, par une condition présente dans *theme.init*.

S'il a choisi le thème sombre lors de sa précédente visite, alors il y aura bien le cookie et dans la console du navigateur, le console.log affichera "yes".

La condition "si *darkThemeLover* est strictement égale à yes" sera vérifiée et appellera la méthode *handleSwitchBtnClick* qui appliquera le thème sombre.

```
// set the color theme by reading what is in local storage  
const darkThemeLover = localStorage.getItem('theme-dark');  
console.log(darkThemeLover);  
if (darkThemeLover === 'yes') {  
    // If there is the key 'theme-dark' with the value 'yes' in  
    // the local storage, then the theme is dark  
    theme.handleSwitchBtnClick();  
    // else it is the light theme  
}
```

Console :
yes theme.js:25

2 - Site de la liste de tâches :

Cette application de liste de tâches, réalisée avec le framework *Laravel*, permet à un utilisateur de se créer une “*To Do List*” et de la gérer.

The screenshot shows a simple web application titled "Task manager". At the top is a yellow header bar with the title. Below it is a dark blue button labeled "nouvelle tâche". The main area contains three task cards:

- Laver son assiette (Corvées) with edit and delete icons.
- Nouveau titre pour la tache (Loisirs) with edit and delete icons.
- Coucou (Loisirs) with edit and delete icons.

(*JavaScript Object Notation*) et non plus *XML*.

La mise en page est minimalist et l'affichage de la liste est fait uniquement par le *DOM*. Les données de la liste sont récupérées par une *API* avec des requêtes *AJAX*.

AJAX veut dire *Asynchronous JavaScript And XML* (*JavaScript* et *XML* asynchrones) et permet d'envoyer et récupérer des données vers et depuis un serveur de façon asynchrone, c'est-à-dire sans avoir à recharger la page. Aujourd'hui, on utilise le format d'échange de données *JSON*



```
const task = {
    init: function() {
        console.log("démarrage du module task");
        // On arrival on the page we retrieve all tasks
        task.loadFromAPI();
    },
    // Asynchronous method to collect all tasks from API (Javascript
    // doesn't wait for the method to be done to continue)
    loadFromAPI: async function() {
        // Use of fetch to connect to the API
        // "await" tells the method to wait for the end of the request
        // to continue
        const response = await fetch(app.APIBaseUrl + 'tasks');

        // Translation of the API's response in a Javascript array
        const taskList = await response.json();

        // To insert the created tasks in the DOM, each task
        // (currentTask) from the array of all tasks (taskList) is sent to the
        // create method, which insert them in the DOM
        for (const currentTask of taskList) {
            task.create(currentTask);
        }
    }
}
```

les tâches stockées sur un serveur.

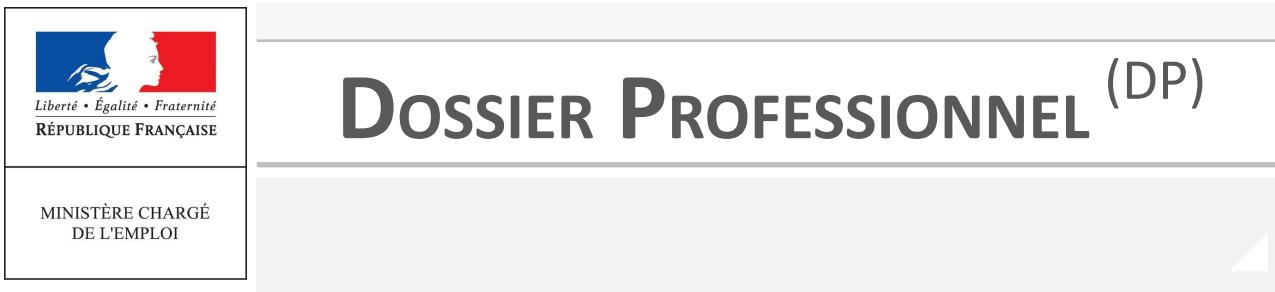
Pour en faire un objet *Javascript*, j'utilise *json()*.

Je parcours le tableau, stocké dans *taskList*, afin d'envoyer chaque tâche (*currentTask*) à la méthode *create*. Cette dernière est la méthode dédiée à l'ajout de chaque tâche dans le *DOM*.

À l'arrivée sur le site, je contacte l'*API* (*Application Programming Interface* ou « interface de programmation d'application ») afin de récupérer toutes les tâches déjà créées.

J'utilise la méthode *fetch* dans ma fonction asynchrone *loadFromAPI* (*JavaScript* n'attend pas que la fonction ait fini pour continuer).

response (promesse sous la forme d'un objet *Response* de *fetch*) contient une représentation de l'intégralité de la réponse *HTTP*. Dans notre cas,



```

create: function(taskFromAPI) {
    // Creation of the li, p, delete div and edit div elements
    const taskElement = document.createElement('li');
    const titleElement = document.createElement('p');
    const editElement = document.createElement('div');
    const deleteElement = document.createElement('div');
    const categoryElement = document.createElement('em');

    // li needs to have an data-id attribute with the id for the
    // task in creation
    taskElement.dataset.id = taskFromAPI.id;

    // The task's title is added to the p element
    titleElement.textContent = taskFromAPI.title;

    // titleElement is added to the li
    taskElement.append(titleElement);

    // The category is added to its em element
    categoryElement.textContent = taskFromAPI.category.name;

    // The category is added to the task
    taskElement.append(categoryElement);

    // The buttons edit and delete are given their classes
    editElement.classList.add('edit');
    deleteElement.classList.add('delete');

    // Event listeners are added to the buttons
    deleteElement.addEventListener('click',
        task.handleDeleteClick);
    editElement.addEventListener('click', task.handleEditClick);
    // Both buttons are added to the task in creation
    taskElement.append(deleteElement);
    taskElement.append(editElement);

    // The task taskElement is added to the ul with the class
    // "tasklist"
    const tasksContainer = document.querySelector('.tasklist');

    tasksContainer.append(taskElement);
}

```

du fichier *index.html* et y ajoute la tâche *taskElement* avec *append*.

```

▼<ul class="tasklist">
  ▼<li data-id="3"> [flex] == $0
    <p>Laver son assiette</p>
    <em>Corvées</em>
    <div class="delete"></div>
    <div class="edit"></div>
  </li>
  ...

```

Cette procédure sera répétée autant de fois qu'il y a des tâches contenues dans le tableau *taskList* de *loadFromAPI*.

Pour mettre les tâches, envoyées depuis *loadFromAPI*, sous forme de liste, je crée les éléments **, *<p>*, *<div>* et **.

J'ajoute un attribut *data-id* pour que chaque tâche *taskElement* ait son *id*. J'ajoute le titre de la tâche à *titleElement* (qui correspond à la balise *<p>*) que j'ajoute ensuite à *taskElement*. De même pour la catégorie avec *categoryElement*, qui correspond à la balise **.

J'ajoute ensuite la classe “*edit*” à l’élément *editElement* et la classe “*delete*” à l’élément *deleteElement*, qui sont tous les deux des *<div>*.

J'ajoute à ces derniers des écouteurs d'événement qui appellent les méthodes dédiées et enfin les ajoute à la tâche.

Pour finir, à l'aide de *querySelector*, je récupère la balise *<ul class="tasklist">*

2. Précisez les moyens utilisés :

J'ai utilisé Visual Studio Code et les outils de développements du navigateur, accessibles par la touche F12. Pour le site de la liste de tâches, j'ai également utilisé *Insomnia* pour tester mes requêtes HTTP.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet, en m'inspirant du cours du formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *École O'Clock*

Chantier, atelier, service ▶ *Exercice de formation*

Période d'exercice ▶ Du : 27/03/2023 au : 11/09/2023

5. Informations complémentaires (facultatif)



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

Activité-type 2

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

CP 5 - Créer une base de données.

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

Dans le cadre d'un exercice pratique de la formation, nous avons eu à réaliser un site de vente de chaussures avec un *back-office* permettant de gérer le contenu. Ce dernier n'étant accessible que si l'utilisateur a le rôle adéquat.

Afin de pouvoir réaliser le site, j'ai eu à créer sa base de données.

J'ai reçu le brief du client fictif et j'ai utilisé la méthode *Merise* (méthode informatique qui facilite la création de base de données et de projets informatiques). J'ai ainsi pu créer le Modèle Conceptuel de Données (MCD), le Modèle Logique de Données (MLD), le Modèle Physique de Données (MPD) et rédiger le dictionnaire de données.

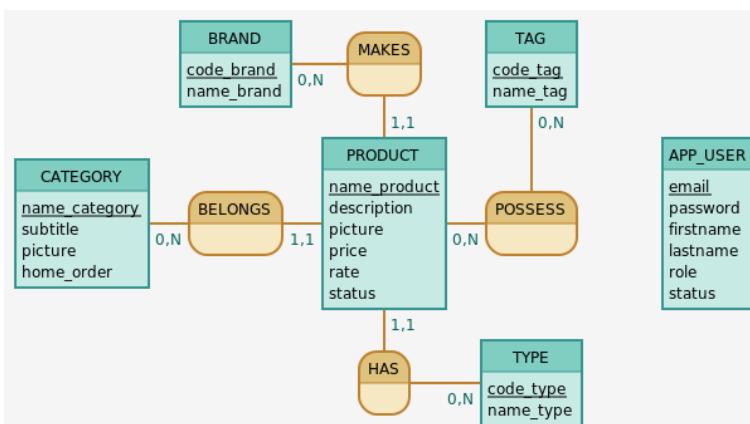
1 - MCD (Modèle Conceptuel de Données) :

Le MCD est une modélisation des données où on y représente rigoureusement un système de données sous la forme d'entités et leurs relations, ainsi que leurs cardinalités.

Dans le cas du site, j'ai listé 6 entités et leurs attributs.

Un produit ne peut avoir qu'une seule marque, catégorie et qu'un seul type. Les type, catégorie et marque peuvent avoir aucun ou plusieurs produits.

Les cardinalités ON:ON représentent une relation *ManyToMany* entre les produits (PRODUCT) et les tags (TAG). En effet, plusieurs produits peuvent avoir plusieurs tag.



L'entité *app_user* n'est pas reliée aux autres entités, mais est utilisée par l'application dans le cas de la gestion du site par le *back-office*.

2 - MLD (Modèle Logique de Données):

Le site *MoCoDo* m'ayant servi à créer le MCD a automatiquement généré le MLD.

On constate que la table *POSSESSES* a été créée dû à la relation *ManyToMany*, elle est une table de liaison avec deux clés étrangères pour relier les tables *PRODUCT* et *TAG*.

De plus, la table *PRODUCT* a 3 clés étrangères : #code_brand, #name_category, #code_type. Ceci est dû au fait qu'elle détient le "1" dans leurs relations.

APP_USER (email, password, firstname, lastname, role, status)

BRAND (code_brand, name_brand)

CATEGORY (name_category, subtitle, picture, home_order)

POSSESSES (#name_product, #code_tag)

PRODUCT (name_product, description, picture, price, rate, status, #code_brand,
#name_category, #code_type)

TAG (code_tag, name_tag)

TYPE (code_type, name_type)

3 - Dictionnaire de données (extrait) :

Une fois les MCD et MLD réalisés, j'ai rédigé le dictionnaire de données afin de représenter les champs de mes tables avec leurs types et spécificités.

PRODUCT			
Champ	Type	Spécificités	Description
id	INT(10)	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	Id du produit
name	VARCHAR(64)	NOT NULL	nom du produit
price	DECIMAL(10,2)	NOT NULL	prix du produit
description	TEXT	NULL	description du produit
image	VARCHAR(64)	NULL	image du produit
rate	TINYINT(1)	NOT NULL, DEFAULT 0	note du produit
brand	Entity	NOT NULL	marque du produit
category	Entity	NOT NULL	catégorie du produit
type	Entity	NOT NULL	type du produit
status	TINYINT(1)	NOT NULL, DEFAULT 0	état du produit (0 = désactivé, 1 = active)
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	date de la création du produit
updated_at	TIMESTAMP	NULL	date de mise à jour du produit

CATEGORY			
Champ	Type	Spécificités	Description
id	INT(10)	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	id de la catégorie
name	VARCHAR(64)	NOT NULL	nom de la catégorie
subtitle	VARCHAR(64)	NULL	sous-titre de la catégorie
picture	VARCHAR(255)	NULL	image de la catégorie
home_order	TINYINT(4)	NOT NULL, DEFAULT 0	de la catégorie
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	date de la création de la catégorie
updated_at	TIMESTAMP	NULL, ON UPDATE CURRENT_TIMESTAMP	date de mise à jour de la catégorie

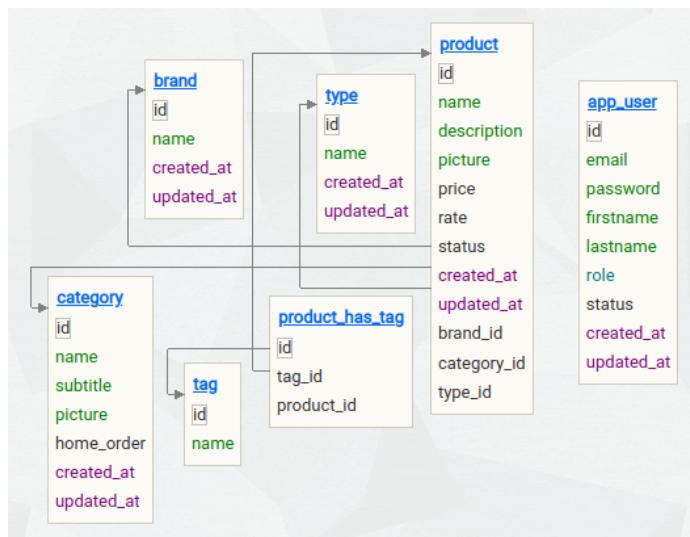
TYPE			
Champ	Type	Spécificités	Description
id	INT(10)	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	id du type
name	VARCHAR(64)	NOT NULL	nom du type
created_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP	date de la création du type
updated_at	TIMESTAMP	NULL, ON UPDATE CURRENT_TIMESTAMP	date de mise à jour du type

BRAND			
Champ	Type	Spécificités	Description
id	INT(10)	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO_INCREMENT	id de la marque
name	VARCHAR(64)	NOT NULL	nom de la marque



DOSSIER PROFESSIONNEL (DP)

4 - MPD (Modèle Physique de Données):



Une fois les tables générées dans mon gestionnaire SGBD, Adminer permet de visualiser le MPD.

Extrait des requêtes SQL pour la création des tables de la base de données :

```

DROP TABLE IF EXISTS `app_user`;
CREATE TABLE `app_user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `email` varchar(128) NOT NULL,
  `password` varchar(60) NOT NULL,
  `firstname` varchar(64) DEFAULT NULL,
  `lastname` varchar(64) DEFAULT NULL,
  `role` enum('admin','catalog-manager') NOT NULL,
  `status` tinyint(3) unsigned NOT NULL DEFAULT 0 COMMENT '1 => actif\n2 => désactivé/bloqué',
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email_UNIQUE` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

DROP TABLE IF EXISTS `brand`;
CREATE TABLE `brand` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(64) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NULL DEFAULT NULL ON UPDATE current_timestamp(),
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

DROP TABLE IF EXISTS `category`;
CREATE TABLE `category` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(64) NOT NULL,
  `subtitle` varchar(64) DEFAULT NULL,
  `picture` varchar(255) DEFAULT NULL,
  `home_order` tinyint(4) NOT NULL DEFAULT 0,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

DROP TABLE IF EXISTS `tag`;
CREATE TABLE `tag` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

```

Lors de leur rédaction, j'effectuais des tests pour vérifier leur bon fonctionnement.

Je me suis aperçue que le fait d'avoir d'abord écrit la requête pour la table de liaison PRODUCT/TAG produisait une erreur.

Je l'ai corrigée en modifiant l'ordre des requêtes (d'abord créer les tables tag et produit, et après la table de liaison).

DROP TABLE supprime la table si elle existe déjà

2. Précisez les moyens utilisés :

J'ai utilisé *MoCoDo* (<https://www.mocodo.net/>), *Word*, *Visual Studio Code* et *Admininer*.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet, en m'inspirant du cours du formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ➤ *École O'Clock*

Chantier, atelier, service ➤ *Exercice de formation - Projet fil rouge*

Période d'exercice ➤ Du : *27/03/2023* au : *11/09/2023*

5. Informations complémentaires (facultatif)

Bien que l'entité *app_user* ne soit liée à aucune autre, il est possible qu'elle le devienne par une évolution des besoins du site.



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

CP 6 - Développer les composants d'accès aux données.

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

Dans la continuité du CP 5, j'ai connecté sa base de données au site de vente de chaussures et son *back-office*. Le site est développé en PHP et utilise le package *Altorouter* (géré par *Composer*) pour la gestion des routes.

Pour communiquer avec la base de données, j'utilise les classes natives *pdo* (*PHP Data Objects*) et *pdoStatement*. Grâce à elles, je peux établir la connexion et faire mes requêtes, et manipuler les données. Les données du site sont gérées depuis le *back-office* et peuvent être lues, créées, modifiées et supprimées. C'est ce que l'on appelle un *CRUD* (*Create, Read, Update, Delete*).

```
● ● ●  
<?php  
namespace Oshop\Utils;  
  
use PDO;  
  
// Retenir son utilisation => Database::getPDO()  
// Design Pattern : Singleton  
class Database  
{  
    /** @var PDO */  
    private $dbh;  
    private static $_instance;  
    private function __construct()  
    {  
        // Récupération des données du fichier de config  
        // la fonction parse_ini_file parse le fichier et retourne un array associatif  
        $configData = parse_ini_file(__DIR__ . '/../config.ini');  
  
        try {  
            $this->dbh = new PDO(  
                "mysql:host={$configData['DB_HOST']};dbname=  
{$configData['DB_NAME']}";charset=utf8",  
                $configData['DB_USERNAME'],  
                $configData['DB_PASSWORD'],  
                array(PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING) // Affiche les erreurs  
SQL à l'écran  
        );  
        } catch (\Exception $exception) {  
            echo 'Erreur de connexion...<br>';  
            echo $exception->getMessage() . '<br>';  
            echo '<pre>';  
            echo $exception->getTraceAsString();  
            echo '</pre>';  
            exit;  
        }  
    }  
    // the unique method you need to use  
    public static function getPDO()  
    {  
        // If no instance => create one  
        if (empty(self::$_instance)) {  
            self::$_instance = new Database();  
        }  
        return self::$_instance->dbh;  
    }  
}
```

Je vais illustrer mes propos en parlant des produits.

Lors de mes cours, l'école O'Clock m'a fourni ce fichier *Database.php*. Je l'utilise tel quel dans mes projets car il est une bonne base. Je l'ai donc utilisé lors du développement de ce site.

Il gère uniquement la connexion à la base de données et me permet de faciliter les requêtes (factorisation).



```
<?php  
  
namespace App\Models;  
  
abstract class CoreModel  
{  
    /**  
     * @var int  
     */  
    protected $id;  
    /**  
     * @var string  
     */  
    protected $created_at;  
    /**  
     * @var string  
     */  
    protected $updated_at;  
  
    /**  
     * Get the value of id  
     *  
     * @return int  
     */  
    public function getId(): ?int  
    {  
        return $this->id;  
    }  
}
```

J'ai développé le site selon le pattern *Active Record* : chaque entité a une classe qui a accès aux données de la base de données.

Je crée la classe *CoreModel.php*, qui centralise toutes les propriétés et méthodes utiles à tous les modèles (en effet, il y a autant de modèles que d'entités).

Vu que toutes les tables de la base de données ont des champs *id*, *created_at* et *updated_at*, ces propriétés sont définies, avec leurs getters et setters, dans *CoreModel*. Cela évite de re-déclarer des propriétés qui seront identiques pour chaque classe(modèle). Cela est possible grâce à l'héritage (avec l'ajout de "extends *CoreModel*").

```
/* Here abstract methods are declared so all classes which inherit from CoreModel  
need to declare these methods (thus implements the CRUD methods) */  
abstract public function insert(): bool;  
abstract public function update(): bool;  
abstract public function delete(): bool;  
abstract public static function find($id);  
abstract public static function findAll();  
}
```

CoreModel impose aussi d'implémenter les méthodes du CRUD à toutes les classes qui héritent de lui. Cela est possible par la déclaration de méthodes abstraites.

Je crée ensuite la classe *Product.php* qui hérite de *CoreModel*. Elle correspond à ma table *PRODUCT* dans la base de données. Elle est composée de propriétés qui correspondent chacune à un champ de la table et de leurs *getters* et *setters* pour la lecture et la modification des données. Ces propriétés sont de type *Private* et ne sont accessibles qu'à l'intérieur de la classe.



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

```
<?php

namespace App\Models;

use App\Utils\Database;
use PDO;

class Product extends CoreModel
{
    /**
     * @var string
     */
    private $name;
    /**
     * @var string
     */
    private $description;
    /**
     * @var string
     */
    private $picture;
    /**
     * @var float
     */
    private $price;

    /**
     * @var int
     */
    private $rate;
    /**
     * @var int
     */
    private $status;
    /**
     * @var int
     */
    private $brand_id;
    /**
     * @var int
     */
    private $category_id;
    /**
     * @var int
     */
    private $type_id;
}
```

Column	Type	Comment
id	int(10) unsigned	<i>Auto Increment</i>
name	varchar(64)	
description	text	<i>NULL</i>
picture	varchar(255)	<i>NULL</i>
price	decimal(10,2) [0.00]	
rate	tinyint(1) [0]	
status	tinyint(1) [0]	
created_at	timestamp [current_timestamp()]	
updated_at	timestamp	<i>NULL</i>
brand_id	int(10) unsigned	
category_id	int(10) unsigned	<i>NULL</i>
type_id	int(11) unsigned	

On constate que les champs de la table *PRODUCT* sont identiques aux propriétés du modèle *Product.php*.

J'ai également implémenté les 5 méthodes héritées de *CoreModel* (*find*, *findAll*, *insert*, *update* et *delete* qui forment le *CRUD*).

Méthode *find* pour récupérer un objet produit grâce à son *id*:

```
/**
 * Method to collect one product given its id from the table PRODUCT
 *
 * @param int $productId
 * @return Product
 */
public static function find($productId)
{
    // PDO object for the connection to the DataBase
    $pdo = Database::getPDO();

    // SQL request for one product
    $sql = '
        SELECT *
        FROM product
        WHERE id = ' . $productId;

    // Use of query because we read
    $pdoStatement = $pdo->query($sql);

    // fetchObject() to collect one result
    $result = $pdoStatement->fetchObject('App\Models\Product');

    return $result;
}
```

C'est une requête, comme toutes celles qui veulent "lire" des données, qui s'écrit avec *SELECT*.

On indique dans quelle table on cherche et à quelle ligne grâce à l'*id*. (Par exemple ici: "je veux le produit dont l'*id* est égale à X")

Méthode `findAll` pour récupérer tous les objets produit :

```
/**  
 * Method to collect all products from the table PRODUCT  
 * with the name for corresponding type, category and brand  
 */  
* @param int $productId  
* @return Product  
*/  
public static function findAll()  
{  
    // PDO object for the connection to the DataBase  
    $pdo = Database::getPDO();  
  
    // SQL request for all products with the name for brand  
    // type and category  
    $sql = "SELECT `product`.* , `brand`.`name` AS brandName,  
    `type`.`name` AS typeName, `category`.`name` AS categoryName  
    FROM `product`  
    INNER JOIN `brand` ON `product`.`brand_id`=`brand`.`id`  
    INNER JOIN `type` ON `product`.`type_id`=`type`.`id`  
    INNER JOIN `category` ON `product`.`category_id`=`category`.`id`  
    ORDER BY id ASC";  
  
    // Use of query because we read  
    $pdoStatement = $pdo->query($sql);  
  
    // fetchAll() to collect many results  
    $results = $pdoStatement->fetchAll(PDO::FETCH_CLASS,  
    'App\Models\Product');  
  
    return $result;  
}
```

À l'origine, la requête de la méthode `findAll` était plus simple ("SELECT * FROM 'product'").

Or, je voulais pouvoir récupérer le nom de la marque, le type et la catégorie de chaque produit afin de mieux pouvoir les gérer depuis le *back-office*. J'ai donc décidé de faire une requête avec jointures.

Méthode pour ajouter un produit :

```
/**  
 * Méthode for adding a product  
 * @return bool  
 */  
public function insert(): bool  
{  
    $pdo = Database::getPDO();  
  
    // INSERT INTO request  
    $sql = "  
        INSERT INTO `product` (  
            name,  
            description,  
            picture,  
            price,  
            rate,  
            status,  
            category_id,  
            brand_id,  
            type_id  
        )  
        VALUES (  
            :name,  
            :description,  
            :picture,  
            :price,  
            :rate,  
            :status,  
            :category_id,  
            :brand_id,  
            :type_id  
        )  
    ";  
  
    $pdoStatement = $pdo->prepare($sql);  
  
    // The dynamic parts of the request are given their value with the aliases  
    // beginning with :  
    $pdoStatement->bindValue(':name', $this->name, PDO::PARAM_STR);  
    $pdoStatement->bindValue(':description', $this->description, PDO::PARAM_STR);  
    $pdoStatement->bindValue(':picture', $this->picture, PDO::PARAM_STR);  
    $pdoStatement->bindValue(':price', $this->price);  
    $pdoStatement->bindValue(':rate', $this->rate, PDO::PARAM_INT);  
    $pdoStatement->bindValue(':status', $this->status, PDO::PARAM_INT);  
    $pdoStatement->bindValue(':category_id', $this->category_id, PDO::PARAM_INT);  
    $pdoStatement->bindValue(':brand_id', $this->brand_id, PDO::PARAM_INT);  
    $pdoStatement->bindValue(':type_id', $this->type_id, PDO::PARAM_INT);  
  
    // Execution of the insertion request  
    $pdoStatement->execute();  
  
    // Number of row  
    $insertedRows = $pdoStatement->rowCount();  
  
    // If at least one row is added  
    if ($insertedRows > 0) {  
        // then id is retrieved  
        $this->id = $pdo->lastInsertId();  
  
        // true if it worked  
        return true;  
    }  
    //if not, false  
    return false;  
}
```

J'utilise *PDO* et *pdoStatement* pour préparer la requête et ses paramètres. Vu que les données proviennent de l'utilisateur, et qu'il ne faut jamais faire confiance



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

aux données des utilisateurs (NTUI), l'utilisation de `$pdo->prepare()` et `$pdoStatement->execute()` et `$pdoStatement->bindValue()` permettent de prévenir les attaques par injections SQL.

Méthode pour modifier un produit :

```
public function update(): bool
{
    $pdo = Database::getPDO();

    // UPDATE request
    $sql = "
        UPDATE `product`
        SET
            name = :name,
            description = :description,
            picture = :picture,
            price = :price,
            rate = :rate,
            status = :status,
            category_id = :category_id,
            brand_id = :brand_id,
            type_id = :type_id,
            updated_at = NOW()
        WHERE id = :id
    ";

    // Preparation of the request
    $pdoStatement = $pdo->prepare($sql);

    // Values of the dynamic parts of the request
    $pdoStatement->bindValue(':id', $this->id, PDO::PARAM_INT);
    $pdoStatement->bindValue(':name', $this->name, PDO::PARAM_STR);
    $pdoStatement->bindValue(':description', $this->description, PDO::PARAM_STR);
    $pdoStatement->bindValue(':picture', $this->picture, PDO::PARAM_STR);
    $pdoStatement->bindValue(':price', $this->price);
    $pdoStatement->bindValue(':rate', $this->rate, PDO::PARAM_INT);
    $pdoStatement->bindValue(':status', $this->status, PDO::PARAM_INT);
    $pdoStatement->bindValue(':category_id', $this->category_id, PDO::PARAM_INT);
    $pdoStatement->bindValue(':brand_id', $this->brand_id, PDO::PARAM_INT);
    $pdoStatement->bindValue(':type_id', $this->type_id, PDO::PARAM_INT);

    // Execution of the request
    $pdoStatement->execute();

    $updatedRows = $pdoStatement->rowCount();

    // true is return if at least one row was added
    return ($updatedRows > 0);
}
```

La méthode pour mettre à jour un produit est quasiment identique à celle de création. La différence se trouve dans la requête SQL avec, à la place de `INSERT, UPDATE`.

J'utilise à nouveau `$pdo->prepare()` et `$pdoStatement->execute()` pour me protéger des injections SQL.

Méthode pour supprimer un produit :

```
/**
 * Method to delete a product from the table
 */
public function delete(): bool
{
    $pdo = Database::getPDO();

    // DELETE request
    $sql = "
        DELETE FROM `product`
        WHERE id = :id
    ";

    // Preparation of the request
    $pdoStatement = $pdo->prepare($sql);
    $updatedRows = $pdoStatement->execute([':id' => $this->id]);

    return ($updatedRows > 0);
}
```

Et enfin, encore une fois pour les mêmes raisons citées plus haut, utilisation de `$pdo->prepare()` et `$pdoStatement->execute()`. Je supprime le produit dont l'`id` est égale à "X".

2. Précisez les moyens utilisés :

J'ai utilisé Visual Studio Code et le gestionnaire SGBD *Adminer*.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet, en m'inspirant du cours du formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *École O'Clock*

Chantier, atelier, service ▶ *Exercice de formation - Projet fil rouge*

Période d'exercice ▶ Du : 27/03/2023 au : 11/09/2023

5. Informations complémentaires (facultatif)



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

Activité-type 2

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

CP 7 - Développer la partie back-end d'une application web ou web mobile.

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

À la suite des CP 5 et CP 6, je continue de parler du *back-office* du site de vente de chaussures.

J'ai dit dans la CP 6 que l'application est développée en PHP avec *Altorouter* pour la gestion des routes.

```
▽ app
  > Controllers
  > Models
  > Utils
  > views
  ≡ config.ini
  ≡ config.ini.dist
  > docs
  ▽ public
    > assets
    > demo_sessions
    🐾 index.php
  > vendor
  ♦ .gitignore
  { } composer.json
  { } composer.lock
  $ import-external-repo.sh
  ⓘ README.md
```

Elle est développée selon le modèle de conception *MVC* (*Model-View-Controller*, ou Modèle-Vue-Contrôleur en français). Il permet une meilleure répartition du travail et d'améliorer la maintenance grâce à la séparation entre la logique métier et l'affichage.

- Le modèle va contenir les méthodes (CRUD) pour communiquer avec la base de données.
- La vue va gérer l'affichage des pages.
- Le contrôleur va contenir la logique qui concerne les actions effectuées par l'utilisateur. Il fait le lien entre les modèles, les vues et le client.

Ce projet a 7 contrôleurs, 7 modèles et de multiples vues (réparties dans divers sous-dossiers)

▽ Controllers	▽ Models	▽ views	▽ product
🐾 AuthenticationController.php	🐾 AppUser.php	> authentication	🐾 add.tpl.php
🐾 CategoryController.php	🐾 Brand.php	> category	🐾 list.tpl.php
🐾 CoreController.php	🐾 Category.php	> error	🐾 update.tpl.php
🐾 ErrorController.php	🐾 CoreModel.php	> layout	
🐾 MainController.php	🐾 Product.php	> main	
🐾 ProductController.php	🐾 Tag.php	> partials	
🐾 UserController.php	🐾 Type.php	> product	
		> user	

On est dans la Programmation Orientée Objet (POO) parce que j'utilise des classes (les modèles) qui ont des méthodes et propriétés qui me permettent d'interagir avec les données. Elles sont comme des "moules" qui, lorsqu'on les instancie, crée un nouvel objet (une instance de la classe) avec toutes les propriétés de cette dernière. De plus, mes contrôleurs et modèles héritent d'une classe abstraite (*CoreController* pour les contrôleurs et *CoreModel* pour les modèles).

L'utilisateur accède au site par un point d'entrée unique : *index.php* (ce fichier est dans le dossier *public*, seul dossier accessible aux utilisateurs depuis leur navigateur). C'est également lui qui démarre la session et qui contient toutes les routes.

Extrait des routes de index.php

```
// We map the routes for Altorouter
// 1 - HTTP Method , 2 - URL route after basePath, 3 - Target (method's name and
// controller's name, 4 - Name of the route, using the convention "controllerName-
// methodName")
$router->map(
    'GET',
    '/',
    [
        'method' => 'home',
        'controller' => 'MainController'
    ],
    'main-home'
);

$router->map(
    'GET',
    '/category/list',
    [
        'method' => 'list',
        'controller' => 'CategoryController'
    ],
    'category-list'
);
```

L'utilisateur arrive sur la page d'accueil du *back-office* qui est la route par défaut.

Comme précisé dans la CP 5, l'accès au *back-office* nécessite d'avoir le rôle adéquat. En effet, tout le monde ne peut pas y accéder et interagir avec la base de données.

J'ai donc mis en place des sécurités pour protéger mon application.

Sécurité:

[oShop](#) Accueil Catégories Produits Types Marques Tags Sélection Accueil Utilisateurs Connexion

Bienvenue dans le backOffice Dans les
shoe...

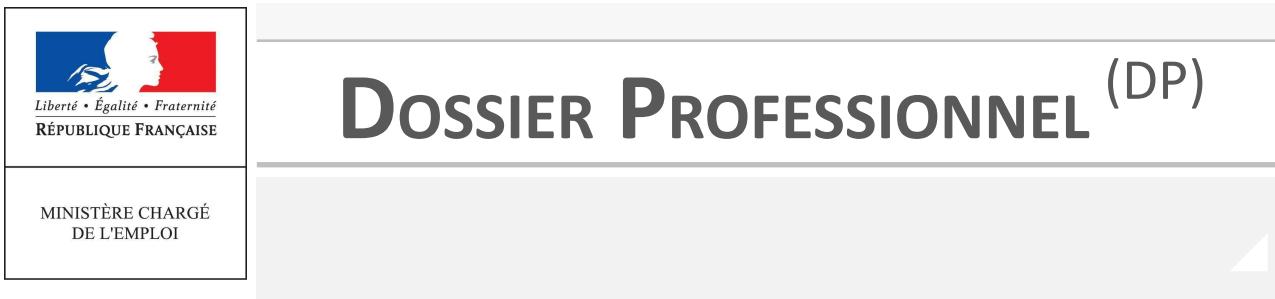
Liste des catégories	
#	Nom
1	Détente
2	Au travail
3	Cérémonie
Voir plus	

Liste des produits	
#	Nom
28	test
1	Kissing
26	Talon aubrille
Voir plus	

[oShop](#) Accueil Catégories Produits Types Marques Tags Sélection Accueil Utilisateurs Connexion

Connexion au back-office

Courriel	<input type="text" value="boite@meuh.cow"/>
Mot de passe	<input type="password" value="Meuh de passe"/>
Se connecter	



MINISTÈRE CHARGÉ
DE L'EMPLOI

```

<?php
namespace App\Controllers;
abstract class CoreController
{
    protected $router;

    public function __construct($router, $match)
    {
        $this->router = $router;
        $routeId = $match['name'];

        $accessControlList = [
            // 'main-home' => [], free access
            'user-list' => ['admin'],
            'user-add' => ['admin'],
            'user-create' => ['admin'],
            'category-list' => ['admin', 'catalog-manager'],
            'category-add' => ['admin', 'catalog-manager'],
            'category-create' => ['admin', 'catalog-manager'],
            'product-list' => ['admin', 'catalog-manager'],
            'product-add' => ['admin', 'catalog-manager'],
            'product-create' => ['admin', 'catalog-manager'],
            'product-create' => ['admin', 'catalog-manager'],
        ];

        if (array_key_exists($routeId, $accessControlList)) {
            $authorizedRoles = $accessControlList[$routeId];
            $this->checkAuthorization($authorizedRoles);
        }
        $this->checkCsrfToken($routeId);
    }

    public function checkCsrfToken($routeId)
    {
        $csrfTokenToCheckInPost = [
            'user-create',
            'category-choose-home-selection-post'
        ];
        if (in_array($routeId, $csrfTokenToCheckInPost)) {
            $formToken = filter_input(INPUT_POST, 'token');
            if (isset($_SESSION['token'])) {
                $sessionToken = $_SESSION['token'];
            } else {
                $sessionToken = ''; // default value if no token
            }
            if ($formToken === null || $formToken === '' || $sessionToken === '' || $formToken !== $sessionToken) {
                $errorController = new ErrorController();
                $errorController->err403();
            } else {
                unset($_SESSION['token']);
            }
        }
    }

    /**
     * Methode to display HTML code based on the views
     * @param string $viewName
     * @param array $viewData
     * @return void
     */
    protected function show(string $viewName, $viewData = [])
    {
        $router = $this->router;
        $viewData['currentPage'] = $viewName;
        $viewData['assetsBaseUrl'] = $_SERVER['BASE_URI'] . 'assets/';
        $viewData['baseUrl'] = $_SERVER['BASE_URI'];

        extract($viewData);
        require_once __DIR__ . '/../views/layout/header.tpl.php';
        require_once __DIR__ . '/../views/' . $viewName . '.tpl.php';
        require_once __DIR__ . '/../views/layout/footer.tpl.php';
    }

    /**
     * Method to redirect to an other page
     *
     * @param string $routeId Route id as defined in index.php
     * @return void
     */
    protected function redirect(string $routeId)
    {
        header('Location: ' . $this->router->generate($routeId));
        exit();
    }

    /**
     * Methode to check if the user has the required role
     *
     * @param array $authorizedRoles Authorized roles list
     */
    public function checkAuthorization(array $authorizedRoles)
    {
        if (isset($_SESSION['userId'])) {
            $user = $_SESSION['userObject'];
            $role = $user->getRole();

            if (in_array($role, $authorizedRoles)) {
                return true;
            } else {
                http_response_code(403);
                $this->show('error/err403');
                die();
            }
        } else {
            $this->redirect('user-login');
        }
    }
}

```

Dans *CoreController*, j'ai mis en place un *Access Control List (ACL)*.

C'est un tableau avec la liste des routes de l'application auxquelles j'attribue les rôles utilisateurs pouvant y accéder. J'ai créé deux rôles : *catalog-manager* et *admin*. L'administrateur peut accéder à toutes les routes tandis que le *catalog-manager* a un accès plus restreint : il ne peut pas gérer les utilisateurs du site. Grâce à *checkAuthorization*, s'il essaie d'y accéder, une erreur 403 (accès interdit) est retournée. Dans le cas d'un utilisateur non connecté, il est redirigé vers la page de connexion.

J'ai ensuite voulu protéger mon application des *CSRF (Cross-Site Request Forgery)*. Ce sont des attaques, transmises par des requêtes falsifiées, qui font exécuter une action à un utilisateur authentifié grâce à ses droits, et ce sans qu'il en ait conscience. Pour éviter cela, j'utilise des *tokens* (ou jetons en français).

Par exemple, j'ai mis en place un *token* pour le formulaire d'ajout d'un utilisateur.

Dans la méthode *create()* de *UserController*, je génère de manière aléatoire une chaîne de

caractères contenant 32 octets cryptographiquement sécurisés que j'ajoute à la session de l'utilisateur. Je le passe également à la vue *add.tpl.php* et le mets dans un *input* de type *hidden*.

Méthode *create* de UserController :

```
$token = bin2hex(random_bytes(32));
// token stocked in the user's session
$_SESSION['token'] = $token;

$this->show('user/add', [
    'user' => $userToInsert,
    'errors_list' => $errorsList,
    'token' => $token
```

Template add.tpl.php :

```
<form action="" method="POST" class="mt-5">
    <input type="hidden" name="token" value="<?= $token ?>">
    <div class="mb-3">
```

Formulaire dans l'inspecteur de page :

```
▼<form action method="POST" class="mt-5">
    <input type="hidden" name="token" value="3bb9a604202df2df351a347b
ec750507f4cb740c979cdbb28b19f12ee2f9af5b" == $0
    ▼<div class="mb-3">
```

Lorsque l'administrateur sera authentifié et voudra aller sur la page d'ajout d'un utilisateur, le jeton sera automatiquement généré. *CoreController* par sa méthode *__construct()* appellera *checkAuthorization()* qui vérifiera si l'administrateur a le rôle pour accéder à cette page.

En tant qu'*admin*, il a le droit.

La méthode *__construct()* continue et appelle alors la méthode *checkCsrfToken()*. Cette dernière vérifie si la route nécessite la vérification d'un *token CSRF*. Puisque l'utilisateur *admin* veut ajouter un utilisateur (qui est une route nécessitant un *token*), *checkCsrfToken()* va récupérer le *token* du champ *input* de type *hidden* du formulaire et le comparer à celui stocké dans la session de l'administrateur. S'ils sont différents, l'action s'arrête et retourne une erreur "403 - Accès interdit". S'ils sont identiques, on laisse l'action se dérouler et on supprime le *token* car ils sont à usage unique.

Enfin, lorsqu'un utilisateur veut se connecter, il entre son email et son mot de passe dans le formulaire de connexion et le soumet. De là, *UserController* appelle la méthode *loginPost* .

Je récupère l'email (*\$email*) et le mot de passe (*\$password*) saisis. J'appelle la méthode *findByEmail* de mon modèle *AppUser* et lui demande de trouver l'email entré dans la base de données. S'il existe, je compare ensuite le mot de passe saisi avec celui enregistré avec l'email. Les mots de passes étant *hashés* (cryptés) avec *password_hash()*, j'utilise donc *password_verify* pour les comparer.

Si l'email et le mot de passe donnés correspondent à ceux enregistrés en base de données, alors l'utilisateur existe et s'authentifie, et je lui ajoute les informations à sa session ("userId" : l'*id* de l'utilisateur connecté et "userObject" : l'objet *AppUser* de l'utilisateur connecté).

Pour effectuer un test du fonctionnement des rôles, je me suis créé un compte avec le rôle de *catalog-manager*. Une fois connectée, j'ai essayé d'accéder à la page de gestion des utilisateurs et j'ai reçu une page 403. Le test est concluant.

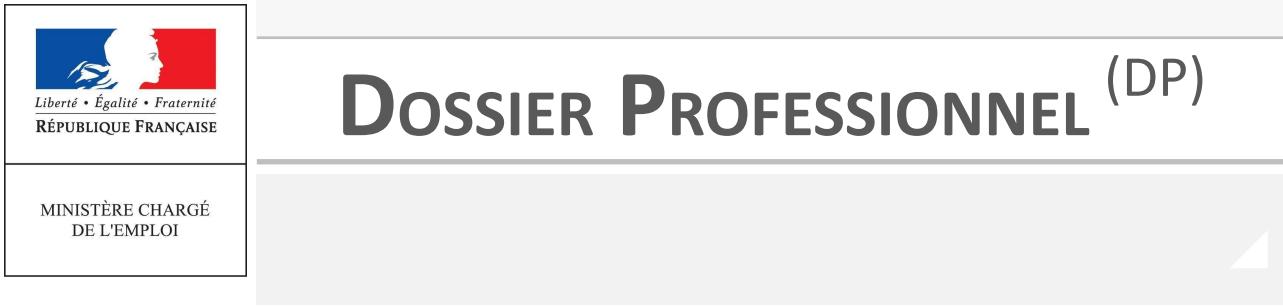
```
public function loginPost()
{
    $email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
    $password = filter_input(INPUT_POST, 'password');

    $appUser = AppUser::findByEmail($email);

    if ($appUser) {
        if (password_verify($password, $appUser->getPassword())) {

            $_SESSION['userId'] = $appUser->getId();
            $_SESSION['userObject'] = $appUser;

            $this->redirect('main-home');
        } else {
            echo 'Courriel ou mot de passe incorrect';
        }
    } else {
        echo 'Courriel ou mot de passe incorrect';
    }
}
```



DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ
DE L'EMPLOI

localhost:8080/user/list

oShop Accueil Catégories Produits Types Marques Tags Sélection Utilisateurs Bonjour Marie Accueil Déconnexion

403 Forbidden

Vous n'avez l'autorisation suffisante pour accéder à cette page.

[retour à l'accueil](#)

Circulation des données :

Je vais maintenant parler de la circulation des données en prenant l'exemple d'ajout d'une catégorie.

Une fois l'utilisateur arrivé sur la page d'accueil du *back-office* et qu'il s'est connecté (en tant que *catalog-manager* ou *admin*), il clique sur lien du menu de navigation menant aux catégories.

Le "clic" appelle la méthode *list()* du contrôleur *CategoryController*, celle-ci appelant à son tour la méthode *findAll()* du modèle *Category*, qui va récupérer la liste de toutes les catégories contenues dans le *back-office*.

oShop Accueil Catégories Produits Types Marques Tags Sélection Accueil Utilisateurs Bonjour Marie Déconnexion

Liste des catégories

#	Nom	Sous-titre	Ajouter
1	Détente	Se faire plaisir	
2	Au travail	C'est parti	
3	Cérémonie	Bien choisir	
4	Sortir	Faire un tour	
5	Vintage	Découvrir	
6	Piscine et bains		
7	Sport		

findAll() retourne les résultats, qui sont stockés dans la variable *\$categoriesList* de *CategoryController*. Ce dernier transmet ces données à la vue *list.tpl.php* et affiche ensuite la page de façon dynamique grâce à la méthode *show()*, héritée du *CoreController*.

```
<a href=<?= $router->generate('category-add') ?><?>
class="btn btn-success float-end">Ajouter</a>
<h2>Liste des catégories</h2>
```

L'utilisateur clique sur "Ajouter", ce qui appelle la méthode *generate()* du *router*. Celle-ci permet d'accéder à la route "*category-add*" dont l'URL est "<http://localhost:8080/category/add>" et par conséquent au formulaire d'ajout d'une catégorie.

oShop Accueil Catégories Produits Types Marques Tags Sélection Accueil Utilisateurs Bonjour Marie Déconnexion

Ajouter une catégorie

[Retour](#)

Nom	<input type="text" value="Nom de la catégorie"/>
Sous-titre	<input type="text" value="Sous-titre"/>
Image	<input type="text" value="image jpg, gif, svg, png"/> URL relative d'une image (jpg, gif, svg ou png) fournie sur cette page

[Valider](#)

Une fois le formulaire rempli, le “clic” sur le bouton “Valider” (bouton de type *submit*) lance une requête HTTP de type *POST* et appelle la méthode *create()* de *CategoryController*.

Je récupère *\$name*, *\$subtitle* et *\$picture* grâce à l’attribut *name* du formulaire. J’instancie un nouvel objet *Category* *\$categoryToInsert* et je vais modifier ses valeurs de propriétés en utilisant les setters de mon modèle *Category* avec *setName()*, *setSubtitle()* et *setPicture()*.

```
public function insert(): bool
{
    $pdo = Database::getPDO();
    $sql = "
        INSERT INTO `category` (`name`, `subtitle`, `picture`)
        VALUES (:name, :subtitle, :picture)
    ";
    $pdoStatement = $pdo->prepare($sql);
    $pdoStatement->bindValue(':name', $this->name, PDO::PARAM_STR);
    $pdoStatement->bindValue(':subtitle', $this->subtitle, PDO::PARAM_STR);
    $pdoStatement->bindValue(':picture', $this->picture, PDO::PARAM_STR);
    $pdoStatement->execute();
    $insertedRows = $pdoStatement->rowCount();
    if ($insertedRows > 0) {
        $this->id = $pdo->lastInsertId();
        return true;
    }
    return false;
}
```

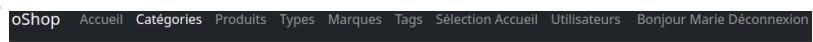
insert() retourne un *booléen*, elle retourne *vrai* car l’insertion a fonctionné. Ce qui permet à *create()* de rediriger vers la page “category-list” (dont l’URL est “<http://localhost:8080/category/list>”) qui liste toutes les catégories. À nouveau, *CategoryController* fait appel à la méthode *findAll()* du modèle *Category*, qui permet d’afficher maintenant la dernière catégorie ajoutée : “catégorie 2”.

```
public function create()
{
    $name = filter_input(INPUT_POST, 'name');
    $subtitle = filter_input(INPUT_POST, 'subtitle');
    $picture = filter_input(INPUT_POST, 'picture');

    $categoryToInsert = new Category();
    $categoryToInsert->setName($name);
    $categoryToInsert->setSubtitle($subtitle);
    $categoryToInsert->setPicture($picture);

    if ($categoryToInsert->save()) {
        $this->redirect('category-list');
    }
}
```

insert() du modèle *Category* envoie une requête SQL. Celle-ci est protégée contre les injections SQL par l’utilisation des méthodes *prepare()*, *bindValue()* et *execute()* (comme vu précédemment dans la CP 6). Étant donné que la méthode



Liste des catégories

#	Nom	Sous-titre	
1	Détente	Se faire plaisir	
2	Au travail	C'est parti	
3	Cérémonie	Bien choisir	
4	Sortir	Faire un tour	
5	Vintage	Découvrir	
6	Piscine et bains		
7	Sport		
9	catégorie 2	testest	



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

J'ai utilisé Visual Studio Code et le gestionnaire SGBD *Adminer*. Le projet est développé en PHP avec les packages *Altorouter* et *Altodispatcher* (gérés par *Composer*). Enfin, l'intégration a été faite avec *Bootstrap*.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet, en m'inspirant du cours du formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *École O'Clock*

Chantier, atelier, service ▶ *Exercice de formation - Projet fil rouge*

Période d'exercice ▶ Du : 27/03/2023 au : 11/09/2023

5. Informations complémentaires (facultatif)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
BTS Métiers de l'Audiovisuel, option Techniques d'Ingénierie et Exploitation des Équipements	xxx, XXXX(00)	00/00/2015



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné(e) Marie M

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 XXXXXXXXXXXXXXXXXXXX

le 12/10/2023

pour faire valoir ce que de
droit.

Signature :

X

Documents illustrant la pratique professionnelle

(*facultatif*)

Intitulé
Cliquez ici pour taper du texte.

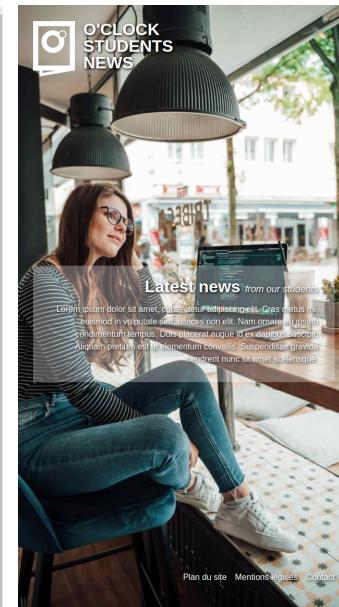
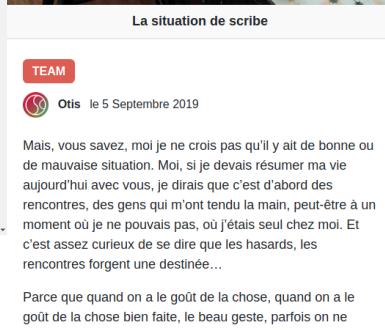
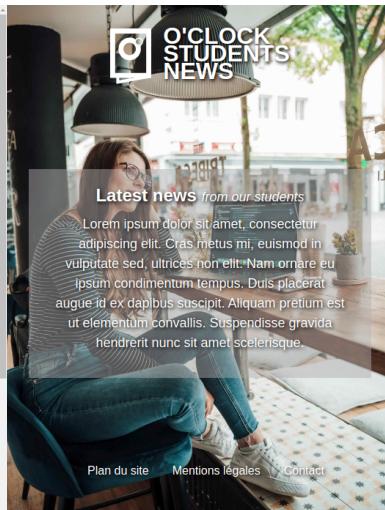


DOSSIER PROFESSIONNEL (DP)

ANNEXES

(Si le RC le prévoit)

Annexe 1 : CP 2 : Intégration Format mobile



Format desktop

