



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

- | | |
|-------------------------|---|
| <i>Nom de naissance</i> | ► <i>Cordier.</i> |
| <i>Nom d'usage</i> | ► <i>Cordier.</i> |
| <i>Prénom</i> | ► <i>Meryl.</i> |
| <i>Adresse</i> | ► <i>25 rue d'Amiens 80310 saint Vaast en chaussée.</i> |

Titre professionnel visé

Développeur Web et Web Mobile

MODALITE D'ACCES :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)



MINISTÈRE CHARGÉ
DE L'EMPLOI

(DP)

Dossier Professionnel

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>



(DP)

Dossier Professionnel

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. 10
- ▶ CP 3 Développer une interface utilisateur web dynamique. p. 16

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. 18

- ▶ CP 5 Créer une base de données. p. 18
- ▶ CP 6 Développer les composants d'accès aux données. p. 23
- ▶ CP 7 Développer la partie back-end d'une application web ou web mobile. p. 28

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

p. 31

Déclaration sur l'honneur

p. 32

Documents illustrant la pratique professionnelle (facultatif)

p. 33

Annexes (Si le RC le prévoit)

p. 34



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

EXEMPLES DE PRATIQUE PROFESSIONNELLE



(DP)

Dossier Professionnel

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 1 ► Maquetter une application

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

Pour toute l'activité de type 1 j'ai choisi de refaire la page d'accueil d'un site de voyage. On l'appellera "France voyage".

Le cahier des charges prévoit une barre de navigation, un carrousel d'image, et une section qui aura deux versions d'affichage :

- Une où l'on voit des destinations possibles, avec leur description
- Une autre où l'on voit les avis d'autres voyageurs.

Cette dernière sera filtrable en fonction de la note sélectionnée.

Par ailleurs, l'inscription à une newsletter est également nécessaire.

Dans un premier temps, j'ai défini les user stories

En tant que	Je veux	Pour
Visiteur	Que la newsletter soit masquée par défaut lorsque j'arrive sur la page.	Avoir une expérience de navigation moins encombrée dès l'ouverture de la page.
	Pouvoir afficher la newsletter lorsque je clique sur le lien "Newsletter".	Pouvoir accéder facilement aux informations de la newsletter si je le souhaite.
	Pouvoir refermer la newsletter en cliquant sur la croix.	Contrôler l'affichage de la newsletter selon mes besoins.
	Afficher l'image suivante dans le carrousel en cliquant sur le bouton "Suivant".	Visualiser les différentes images présentées dans le carrousel.
	Afficher l'image précédente dans le carrousel en cliquant sur le bouton "Précédant".	Visualiser les différentes images présentées dans le carrousel.
Administrateur	Que la newsletter se réaffiche lorsque le visiteur scroll au-delà de 500 pixels après l'avoir fermée une fois.	Inviter le visiteur à s'inscrire
	Que le système filtre les adresses email qui contiennent des domaines interdits lors de la soumission du formulaire de la newsletter.	M'assurer que seules des adresses valides sont enregistrées pour la newsletter.



(DP)

Dossier Professionnel

Dans un second temps, j'ai réalisé les wireframes dans les versions pour mobile, tablette et ordinateur :

- Pour les destinations possibles :

The image displays two wireframe prototypes side-by-side. On the left is the 'Version mobile' (mobile version), shown as a smartphone screen. On the right is the 'Version tablette' (tablet version), shown as a tablet screen. Both screens feature a header with a logo, the text 'Nom du client', and three navigation links: 'Favoris', 'Newsletter', and 'Avis'. A horizontal line labeled 'Navbar' indicates the top navigation bar. Below the header, both screens show a title 'Titre du site' and a large central image placeholder with arrows for navigation. Underneath this, there are two card-like sections, each with a title 'nom de la page', a small image placeholder, and a detailed description: 'titre de la carte' followed by the Latin text 'Quodsi haberent magnalia inter potentiam et divitias, et non illam'. Each card also features a heart icon and a 'Découvrir' button.



(DP)

Dossier Professionnel

Version ordinateur

The wireframe illustrates the layout of the Dossier Professionnel website on a desktop screen. At the top, there is a header bar with the French tricolor logo, the text "Liberté • Égalité • Fraternité", and "RÉPUBLIQUE FRANÇAISE". Below this, a secondary header contains the text "MINISTÈRE CHARGÉ DE L'EMPLOI". The main content area begins with a "Navbar" section featuring a "Logo", the text "Nom du client", and three links: "Favoris", "Newsletter", and "Avis". The main content area is titled "Titre du site" and includes a large image placeholder with navigation arrows (< and >) and a central image icon. Below this, there is a section titled "nom de la page" containing three card-like components, each with a placeholder image, a title "titre de la carte", and a descriptive subtitle "Quodsi haberent magnalia inter potentiam et divitias, et non illam". Each card also features a "Découvrir" button.



MINISTÈRE CHARGÉ
DE L'EMPLOI

(DP)

Dossier Professionnel

- Pour les avis :

Version mobile



Version tablette





(DP)

Dossier Professionnel

Version ordinateur

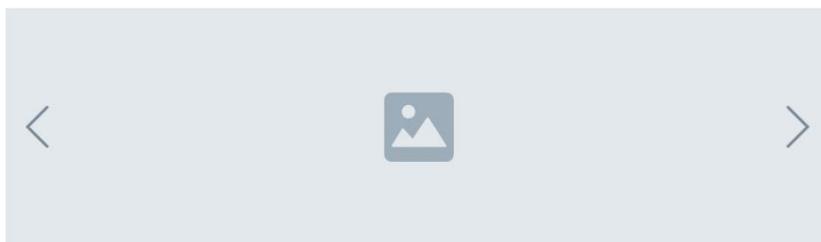


Nom du client

Favoris Newsletter Avis

Navbar

Titre du site



nom de la page

Checkbox Checkbox

Nom auteur



Sic de isto et tutius perducit ad actum ipsum, ut si dico "Ego autem vadam lavari, ut mens mea in statu naturae conformior." Et similiter circa alias res. Et sic, si contingit ex per se lavantem, et erit hoc paratus ut diceret, "Hoc non solum lavari ut desideravit, sed ut animus in statu naturae convenienter

Nom auteur



Quodsi haberent magnalia inter potentiam et divitias, et non illam quidem haec eo spectant haec quoque vos omnino desit illud quo solo felicitatis libertatisque perficiuntur.

2. Précisez les moyens utilisés :

Pour ce faire, j'ai utilisé Whimsical.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule

4. Contexte

Nom de l'entreprise, organisme ou association ► *Ecole O'Clock.*

Chantier, atelier, service ► *Projet personnel réalisé pendant la formation*

Période d'exercice ► Du : 11/09/2023 au : 28/02/2024

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 2 ▶ Réaliser une interface web statique et adaptable.

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

En me basant sur ces user stories et wireframes, je me suis lancée dans la réalisation de ce projet.

Le premier objectif était de faire un site responsive. J'ai donc codé le CSS en mobile-first et sans librairie. Ainsi les règles du BEM ont été respectées pour nommer les classes.

Le corps de la page est codé en HTML et intègre les noms de balises nécessaires à un bon SEO. L'interaction avec l'utilisateur est gérée avec JavaScript.

Voici un aperçu du site en question :



Voyages Restaurants Hôtels Vols Croisières

Partir en France



Voyages Restaurants Hôtels Vols Croisières

♡ Favoris 📧 Newsletter 🗞 Avis

Partir en France



Destinations



Visiter la capitale
Lorem ipsum dolor sit amet consectetur adipisciing elit. Magnam, minima!

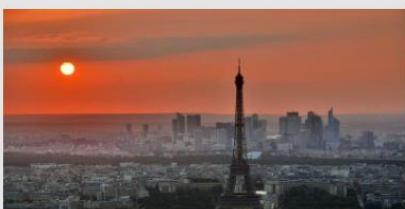


Partir au bord de mer
Lorem ipsum dolor sit amet consectetur adipisciing elit. Magnam, minima!



Promenade dans la nature
Lorem ipsum dolor sit amet consectetur adipisciing elit. Magnam, minima!

Destinations





(DP)

Dossier Professionnel

1. Structure de la page

Pour commencer, cette page a l'entête indispensable à tout fichier HTML. Celle-ci fait appel à deux fichiers CSS, le fichier de style et celui de reset.

Nous avons ensuite la première section de la page : le Header.

Il renferme la partie haute du wireframe : logo, nom du site et barre de navigation.

Cette dernière est divisée en deux containers. Comme vous pouvez le voir ci-dessous :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>France voyage</title>
<link rel="stylesheet" href="css/reset.css">
<link rel="stylesheet" href="css/styles.css">
<link rel="shortcut icon" href="img/favicon.ico" type="image/x-icon">
</head>
<body>

<header class="header">
<nav>
<div class="header__part">
<div class="container">
<div class="header__flex">
<a class="logo" href="index.html">

<span class="logo__title" href="index.html">France voyage</span>
</a>
<div class="menu">
<a class="menu__item" href="#"> Favoris</a>
<a id="newsletter-link" class="menu__item" href="#"> Newsletter</a>
<a class="menu__item" href="avis.html"> Avis</a>
</div>
</div>
</div>
<div class="header__part">
<div class="container">
<nav class="submenu">
<a class="submenu__item submenu__item--current" href="index.html">Voyages</a>
<a class="submenu__item" href="#"> Restaurants</a>
<a class="submenu__item" href="#"> Hôtels</a>
<a class="submenu__item" href="#"> Vols</a>
<a class="submenu__item" href="#"> Croisières</a>
</nav>
</div>
</div>
</nav>
</div>
</header>
```



(DP)

Dossier Professionnel

Puis, nous voyons la partie main qui inclus le titre du site, le carrousel d'image et (dans le cas de la page d'accueil) une section contenant les cartes de présentation des destinations.



```
<main class="container">
  <div class="content">
    <h1 class="content__title">Partir en France</h1>

    <section class="slider">
      <button class="btn slider__btn" id="previous_btn" type="button" aria-
label="Précédent">&lt;;</button>
      <div class="slider__nav--item">
        
        
        
        
        
        
      </div>
      <button class="btn slider__btn" id="next_btn" type="button" aria-
label="Suivant">&gt;;</button>
    </section>

    <section class="segment">
      <h2 class="segment__title">Destinations</h2>
      <div class="cards">
        <article class="card">
          <div class="card__content">
            
            <h3 class="card__title">Visiter la capitale</h3>
            <p class="card__desc">Lorem ipsum dolor sit amet consectetur adipisicing elit. Magnam, minima!</p>
          </div>
          <div class="card__action">
            <button class="btn btn__like" type="button" aria-label="J'aime"><i class="icon-
heart"></i></button> <a class="btn" href="">Découvrir</a>
          </div>
        </article>
      </div>
    </section>
  </div>
</main>
```

Nous pouvons remarquer ici la présence d'une hiérarchisation des titres avec les balises h1 et h2. Pour permettre une bonne accessibilité du site, les images ont un "alt" renseigné et les boutons ont un aria-label.



(DP)

Dossier Professionnel

Pour finir, le corps de cette page est clôturé avec un aside et un footer :

```
</div>
</main>

<aside class="newsletter newsletter--hidden">
  <button class="newsletter__close btn" aria-label="Fermer" type="button">&Cross;</button>
  <h2 class="newsletter__title">Inscription à la newsletter</h2>
  <form id="newsletter-form">
    <input id="subscriber-email" class="newsletter__field" type="email" required
placeholder="user@email.com" aria-label="Votre email">
    <button class="btn" type="submit">Valider</button>

    <div id="email-result" class="email-result"></div>
  </form>
</aside>

<footer class="footer">
  <div class="container">
    <p>Toute ressemblance avec d'autres contenus est purement fortuite.</p>
  </div>
</footer>

<script src=".js/app.js"></script>
</body>
</html>
```

En effet, une fenêtre de newsletter devant apparaître (selon les user stories), celle-ci est intégrée dans une balise aside. Nous pouvons remarquer la présence d'un formulaire ayant son aria-label également.

Vient enfin le footer et surtout le script faisant appel au fichier JavaScript qui gère les événements déclenchés par l'utilisateur.

2. Responsive et média queries

Le but étant d'avoir un style responsive, j'ai suivi la méthode mobile-first et utilisée des médias queries. Elles ont deux break points : un pour les tablettes à partir de 768px et un pour les ordinateurs à partir de 992px.

Cela me permet de gérer la direction de la flexbox du header, la largeur des cartes ainsi que celle de la newsletter pour que leurs alignements soient harmonieux.



MINISTÈRE CHARGÉ
DE L'EMPLOI

(DP)

Dossier Professionnel

```
● ● ●

@media screen and (min-width: 768px) {
  .header__flex {
    flex-direction: row;
    justify-content: space-between;
    align-items: center;
  }
  .segment {
    display: flex;
    flex-wrap: wrap;
    padding: 1rem;
    margin: 1rem 0;
  }
  .card{
    width: 47%;
  }
  .newsletter {
    width: 47%;
  }
}

@media screen and (min-width: 992px) {
  .card{
    width: 32%;
  }
  .newsletter {
    width: 35%;
  }
}
```

3. Interaction avec l'utilisateur en JavaScript

La première partie du main est un carrousel d'image. Celui-ci comporte un bouton suivant et un bouton précédent. Le code qui suit a pour but d'ajouter (ou enlever) une class aux images lors du clic de l'utilisateur. Cette dernière fixe le z-index à 1 et permet de modifier l'image à afficher en fonction de ce clic.



(DP)

Dossier Professionnel

```
const btnPrevious = document.querySelector("#previous_btn");
const btnNext = document.querySelector("#next_btn");
const img = document.querySelectorAll(".slider__img");
let imgIndex = 0;

btnNext.addEventListener("click", () => {

    img[imgIndex].classList.remove("slider__img--current");
    imgIndex++;

    if (imgIndex >= img.length) {
        imgIndex = 0;
    }

    img[imgIndex].classList.add("slider__img--current");
});

btnPrevious.addEventListener("click", () => {

    img[imgIndex].classList.remove("slider__img--current");
    imgIndex--;

    if (imgIndex < 0) {
        imgIndex = img.length - 1;
    }

    img[imgIndex].classList.add("slider__img--current");
});
```

Je sélectionne et stocke dans des variables les boutons ‘suivant’ et ‘précédant’ et les images.

Puis je crée une variable ‘imgIndex’ que j’initialise à zéro.

J’écoute le clic sur le bouton suivant.

J’enlève la class ‘slider__img—current’ de l’image actuellement affichée, et j’incrémente ‘imgIndex’.

Je le réinitialise lorsqu’il atteint la longueur du tableau de données des images.

J’ajoute la class class ‘slider__img—current’ à l’image qui doit être affichée.

J’écoute le clic sur le bouton précédent.

Ensuite, la logique est la même. Mais je décrémente le ‘imgIndex’ tout en ne le laissant pas devenir négatif.

2. Précisez les moyens utilisés :

J’ai utilisé VSCode comme environnement de développement intégré. La console de mon navigateur m’a permis de gérer les média queries. *Lighthouse* m’a indiqué la maîtrise des bonnes pratiques (100%) du SEO (90%) et de l’accessibilité (94%).

Quelques recherches google m’ont permis de fixer les break points des média queries et de récupérer les images et logo.

Puis ray.so m’a permis de faire les captures d’écran de code

3. Avec qui avez-vous travaillé ?

J’ai travaillé seule.

4. Contexte

Nom de l’entreprise, organisme ou association ► *Ecole O’Clock.*

Chantier, atelier, service ► *Projet personnel réalisé pendant la formation.*

Période d’exercice ► Du : 11/09/2023 au : 28/02/2024

5. Informations complémentaires (facultatif)



(DP)

Dossier Professionnel

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 :

Pour répondre aux exigences de sécurité, nous allons détailler la gestion de l'inscription à la newsletter. Elle est prévue pour rejeter une liste de mails jetables.

• • •

```
const formElement = document.getElementById("newsletter-form");
formElement.addEventListener("submit", (event) => {
  event.preventDefault();
  const emailInput = document.querySelector("#subscriber-email");
  const userEmail = emailInput.value;

  const forbiddenDomains = [
    '@mail-temporaire.fr',
    '@mail7.io',
    '@mailcatch.com',
    '@mailHazard.com',
    '@yopmail.com',
    '@yopmail.fr',
    '@yopmail.net',
  ];

  let isEmailOK = true;

  forbiddenDomains.forEach((forbiddenDomain) => {
    if (userEmail.includes(forbiddenDomain)) {
      isEmailOK = false;
    }
  });

  const resultElement = document.querySelector("#email-result");

  if (isEmailOK) {
    resultElement.textContent = "Votre inscription est bien prise en compte !";
    resultElement.classList.remove("error");
    resultElement.classList.add("success");
  } else {
    resultElement.textContent = "Le domaine fourni est interdit.";
    resultElement.classList.remove("success");
    resultElement.classList.add("error");
  }
});
```

Je stocke dans une variable le formulaire d'inscription.

Je récupère l'email saisi par l'utilisateur.

Je définis la liste des emails interdits par un tableau.

Je crée une variable gérant l'état de la réponse.

Je boucle sur le tableau des emails interdits pour comparer avec l'email saisi.

S'il y a concordance le booléen de la réponse passe à 'false'.

Je récupère la div contenant le message du résultat.

Si l'email est accepté, j'affiche un texte de succès en appliquant une classe qui le colore en vert.

Sinon, j'affiche un texte d'échec en appliquant une classe qui le colore en rouge.

Il est aussi possible de se connecter à une API tierce pour avoir accès à un plus grand nombre d'adresses email jetables.



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

```
const API_URL = "http://apitierce";  
  
async function fetchData() {  
    try {  
        const response = await fetch(API_URL);  
        const forbiddenDomains = await response.json();  
        return forbiddenDomains;  
    } catch (error) {  
        console.error(error);  
    }  
}  
  
fetchData();
```

Dans ce cas il aurait fallu :

- Se connecter à l'API grâce à son URL :
- Créer une fonction asynchrone pour appeler l'API.

La réponse est stockée sous la variable '*forbiddenDomains*' au format Json. Enfin, juste avant la gestion d'erreur, cette fonction retourne '*forbiddenDomains*' pour que nous puissions travailler avec.

2. Précisez les moyens utilisés :

J'ai fait le choix de rester sur du JavaScript Vanilla

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule

4. Contexte

Nom de l'entreprise, organisme ou association ► *Ecole O'Clock.*

Chantier, atelier, service ► *Projet personnel réalisé pendant la formation.*

Période d'exercice ► Du : 11/09/2023 au : 28/02/2024

5. Informations complémentaires (facultatif)



(DP)

Dossier Professionnel

Activité-type 2 Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.

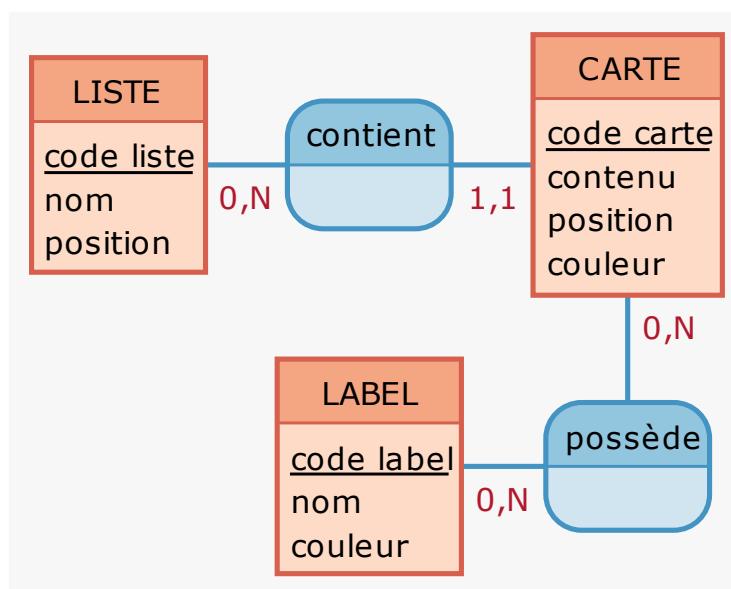
CP 5 ► Créer une base de données.

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

Pour travailler sur cette partie back end, j'ai choisi de reproduire un tableau de type Kanban. Pour ce faire, j'ai d'abord créé un MCD. Suivront le MLD et le MPD.

Mon projet comportera des listes dans lesquelles il pourra y avoir une ou plusieurs cartes. Elles-mêmes pourront avoir des labels ou non.

Pour mieux se rendre compte, voici le Model Conceptuel de données



Ici, nous avons trois entités ayant chacune un discriminant (souligné) et plusieurs attributs. Nous pouvons également repérer les cardinalités entre les différentes entités. En partant de ce MCD, j'ai fait un MLD.

Il peut être vu sous deux formes différentes :

- Modèle Logique de données sous forme textuelle :

```

list ( id, title, position)
card ( id, content, position, color, #list_id )
tag ( id, name, color )
card_has_tag ( id, #tag_id, #card_id )
  
```



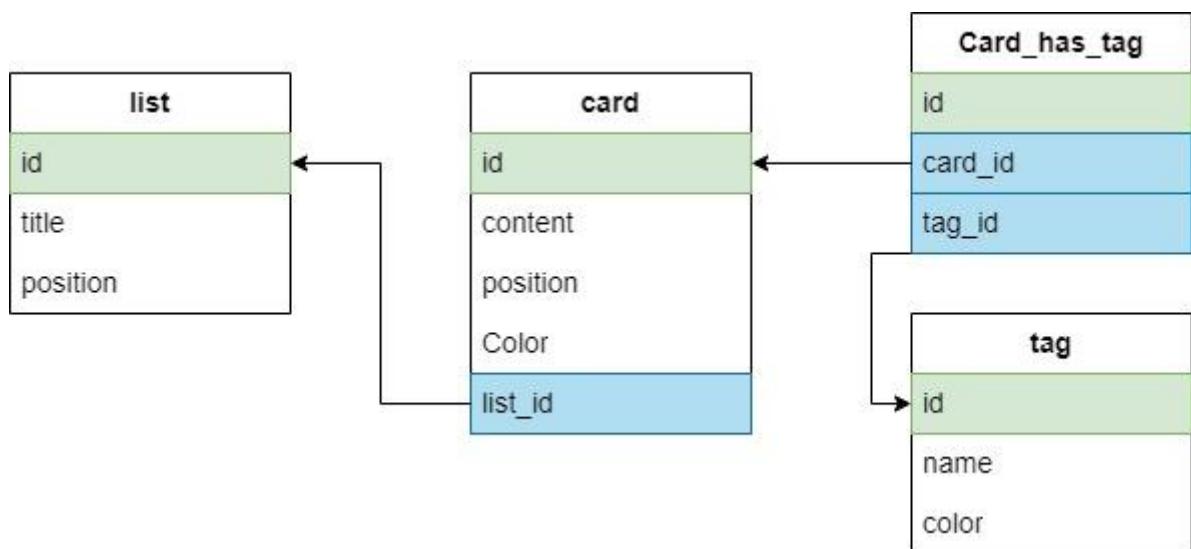
(DP)

Dossier Professionnel

Dans cette version textuelle, les champs sont écrits entre parenthèses. Les clés primaires, générées automatiquement, sont soulignées. Les clés étrangères sont précédées d'un signe #.

On remarque la présence d'une nouvelle table (card_has_tag) qui traduit l'association many to many entre card et tag.

2. Modèle Logique de données sous forme schématique :



Sur ce schéma, les clés primaires apparaissent en vert et les clés étrangère en bleu.

J'aurai pu coder le MPD (Modèle Physique de données) pour permettre la création de cette base de données avec un script SQL.



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

• • •

```
BEGIN;

DROP TABLE IF EXISTS "list", "card", "tag", "card_has_tag";

CREATE TABLE "list" (
    "id" INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    "title" TEXT NOT NULL,
    "position" INTEGER DEFAULT 1,
    "created_at" TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updated_at" TIMESTAMPTZ
);

CREATE TABLE "card" (
    "id" INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    "content" TEXT NOT NULL,
    "position" INTEGER NOT NULL DEFAULT 1,
    "color" VARCHAR(7),

    "list_id" INTEGER NOT NULL REFERENCES "list"("id") ON DELETE CASCADE,
    "created_at" TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updated_at" TIMESTAMPTZ
);

CREATE TABLE "tag" (
    "id" INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    "name" TEXT NOT NULL UNIQUE,
    "color" VARCHAR(7),
    "created_at" TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updated_at" TIMESTAMPTZ
);

CREATE TABLE "card_has_tag" (
    "id" INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

    "card_id" INTEGER NOT NULL REFERENCES "card"("id") ON DELETE CASCADE,
    "tag_id" INTEGER NOT NULL REFERENCES "tag"("id") ON DELETE CASCADE,

    UNIQUE ("card_id", "tag_id"),

    "created_at" TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    "updated_at" TIMESTAMPTZ
);

COMMIT;
```



(DP)

Dossier Professionnel

Mais, j'ai utilisé des modèles *Sequelize*. Par exemple voici celui permettant de créer une liste :

```
import { Sequelize } from "./sequelize-client.js";
import { Model, DataTypes } from "sequelize";

export class List extends Model {}

List.init({
  title: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  position: {
    type: DataTypes.INTEGER,
    defaultValue: 1
  }
}, {
  sequelize,
  tableName: "list"
});
```

Pour exécuter le MPD, il convient de créer la base de données.

En utilisant PostgreSQL comme Système de Gestion de Base de Données (SGDB), les commandes suivantes sont saisies dans le terminal.

- On exécute PostgreSQL en super-utilisateur :

```
sudo -i -u postgres psql
```

- On créer un rôle pour utiliser la base de données :

```
CREATE ROLE nomDeLutilisateur WITH LOGING PASSWORD 'motDePasse';
```

- On créer la base de données :

```
CREATE DATABASE nomDeLaBase OWNER nomDeLutilisateur;
```

- Après s'être déconnecté de super-utilisateur (**Ctrl + d**), on remplit la base de données :

```
psql -U nomUtilisateur -d nomBaseDeDonnees -f chemin/du/fichier.sql
```



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

2. Précisez les moyens utilisés :

Pour le MCD, j'ai utilisé Mocodo

Pour le MLD j'ai utilisé diagrams.net

Pour le MPD j'ai utilisé Sequelize

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole O'Clock.

Chantier, atelier, service ► Projet personnel réalisé pendant la formation.

Période d'exercice ► Du : 11/09/2023 au : 28/02/2024

5. Informations complémentaires (facultatif)

1. Les lignes saisie dans Mocodo pour créer le MCD :

LISTE : code liste, nom, position

contient, ON LISTE, 11 CARTE

CARTE : code carte, contenu, position, couleur

LABEL : code label, nom, couleur

possède, ON CARTE, ON LABEL

2. Le dictionnaire de données en rapport avec le MLD :

LISTE :

Code_Liste (int, Clé primaire) : Identifiant unique de la liste.

Nom (varchar) : Nom de la liste.

Position (int) : Position de la liste dans l'interface utilisateur.

CARTE :

Code_Carte (int, Clé primaire) : Identifiant unique de la carte.

Contenu (text) : Contenu de la carte.

Position (int) : Position de la carte dans la liste.

Couleur (varchar) : Couleur de la carte.

LABEL :

Code_Label (int, Clé primaire) : Identifiant unique du label.

Nom (varchar) : Nom du label.

Couleur (varchar) : Couleur du label.

Relations :

contient (Relation entre LISTE et CARTE) :

Code_Liste (int, Clé étrangère) : Clé étrangère faisant référence à l'identifiant de la liste.

Code_Carte (int, Clé étrangère) : Clé étrangère faisant référence à l'identifiant de la carte.

possède (Relation entre CARTE et LABEL) :

Code_Carte (int, Clé étrangère) : Clé étrangère faisant référence à l'identifiant de la carte.

Code_Label (int, Clé étrangère) : Clé étrangère faisant référence à l'identifiant du label.



(DP)

Dossier Professionnel

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 6 ▶ Développer les composants d'accès aux données.

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

Dans ce projet, pour me connecter à la base de données (BDD), j'ai utilisé *Sequelize*. Ainsi, je devais avoir un PG_URL dans mon fichier de variable d'environnement (".*env*"). Voici la syntaxe respectée :

```
PG_URL=postgres:// nomUtilisateur:mot_de_passe@localhost:port/nomBaseDeDonnees
```

Ensuite, j'ai créé l'instance de connexion à *Sequelize* avec un fichier '*sequelize-client.js*' :

```
import "dotenv/config";  
  
import { Sequelize } from "sequelize";  
  
export const sequelize = new Sequelize(process.env.PG_URL, {  
  logging: false,  
});
```

J'importe les variables d'environnements et *Sequelize*.

J'établis la connexion à la BDD

J'ajoute une option pour désactiver les journaux *Sequelize*. Cela allègera ma console.

Dans le cadre d'un projet en *Sequelize*, il n'est pas nécessaire d'avoir un *DataMapper*. Ainsi, le CRUD est géré dans le *Controller*. Plus précisément, je vais détailler ici, le *listController*. Il comporte cinq fonctions différentes :

- *getAllLists* et *getOneList* récupèrent respectivement toutes ou une liste de la base de données.
- *createList* permet de créer une nouvelle liste dans la base de données.
- *updateList* s'occupe de la modification d'une liste.
- *deleteList* supprime la liste en question de la base de données.

Admettons qu'un utilisateur veuille créer sa première liste dans son tableau Kanban.



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

Mon application va faire appel à la méthode `createList` :

```
import { List } from "../models/index.js";

export async function createList(request, response) {

  const title = request.body.title;
  const position = request.body.position;

  if (!title || typeof title !== "string") {
    response.status(400).json({ error: "Missing or invalid parameter: title" });
    return;
  }

  if (position && ( !Number.isInteger(position) || position < 0 )) {
    return response.status(400).json({ error: "Invalid type: position should be an integer above 0" });
  }

  const createdList = await List.create({
    title,
    position: position || 1
  });

  response.status(201).json(createdList);
}
```

J'importe le modèle `List` créé avec `Sequelize`.

Je crée et exporte la fonction asynchrone qui prend 2 paramètres : `request` et `response`.

Je récupère depuis le corps de la requête le titre et la position de la liste en les conservant dans des variables.

Je vérifie s'il existe bien un titre et si celui-ci est une string.

Je contrôle si l'utilisateur a bien saisi une position étant un nombre non négatif. Si ce n'est pas le cas, un message d'erreur est envoyé.

Je construis la nouvelle liste à partir du modèle. Je lui indique le titre et la position qu'elle doit avoir. Si aucune position n'existe je choisi la n°1 par défaut.

Enfin, je renvoie la nouvelle liste au format Json.

Ce code présente une insécurité. En effet, un utilisateur pourrait inscrire du code malveillant dans le corps de la réponse via le titre. Pour anticiper cela et nettoyer ce contenu, il convient d'installer 'sanitizeHtml' contre les attaques XSS :

```
import sanitizeHtml from 'sanitize-html';

const title = sanitizeHtml(request.body.title);
```



(DP)

Dossier Professionnel

Si notre utilisateur décide de modifier sa liste, je fais intervenir la fonction updateList :

```
...  
export async function updateList(request, response) {  
  const listId = parseInt(request.params.id);  
  if (isNaN(listId)) {  
    return response.status(400).json({ error: "List ID should be a valid integer" });  
  }  
  
  const body = request.body;  
  
  const list = await List.findByPk(listId);  
  if (!list) {  
    return response.status(404).json({ error: "List not found" });  
  }  
  
  const updatedList = await list.update({  
    title: body.title || list.title,  
    position: body.position || list.position  
  });  
  
  response.json(updatedList);  
}
```

Je récupère l'identifiant de la liste dans les paramètres de la requête en lui appliquant la méthode `parseInt`. S'il n'est pas un nombre entier, j'envoie une réponse avec une erreur 400.

J'extrais le corps de la requête

J'attends que `Sequelize` trouve la liste à mettre à jour dans la BDD avec l'identifiant de la requête.

S'il ne correspond pas à une liste existante, j'envoie une erreur 400.

Sinon je modifie la liste avec la méthode `update`. Le titre et la position conservent leur valeur ou prennent la nouvelle valeur.

La liste mise à jour est envoyée au format `Json`.

Pour sécuriser la récupération du corps de la requête, il est conseillé d'utiliser le module `joi`. En définissant un schéma de validation pour les données mise à jour, on peut ensuite les valider.

```
...  
import Joi from "joi";  
  
const updateListSchema = Joi.object({  
  title: Joi.string().min(1),  
  position: Joi.number().integer().min(1)  
}).min(1).message("Missing body parameters. Provide at least 'title' or 'position' parameter");  
  
const { error } = updateListSchema.validate(body);  
if (error) {  
  return res.status(400).json({ error: error.message });  
}
```



(DP)

Dossier Professionnel

Lorsque cette liste devient inutile, l'utilisateur peut la supprimer. C'est alors deleteList qui va agir.

```
● ● ●  
export async function deleteList(request, response) {  
  
  const listId = Number.parseInt(request.params.id, 10);  
  if (isNaN(listId)) {  
    return response.status(400).json({ error: "List ID should be a valid integer" });  
  }  
  
  const list = await List.findByPk(listId);  
  if (!list) {  
    return response.status(404).json({ error: "List not found" });  
  }  
  
  await list.destroy();  
  
  response.status(204).end();  
}
```

J'utilise “*Number.parseInt*” pour convertir l'identifiant, venant des paramètres de la requête, en un nombre entier décimal. S'il n'est pas reconnu comme un nombre j'envoie une erreur 400.

J'applique la méthode *findByPk* sur la *Class List* pour trouver la liste à supprimer. Si elle n'est pas trouvée j'en informe l'utilisateur.

Pour finir, la liste est détruite avec la méthode *destroy* et une réponse 204 est envoyée.

Pour afficher tout le tableau kanban de notre utilisateur, je fais appel à la fonction getAllLists.

```
● ● ●  
export async function getAllLists(request, response) {  
  const lists = await List.findAll({  
    order: [  
      ["position", "ASC"],  
      ["created_at", "DESC"],  
      ["cards", "position", "ASC"]  
    ],  
    include: {  
      association: "cards",  
      include: "tags",  
    },  
  });  
  
  response.json(lists);  
}
```

A l'aide de la méthode *findAll* j'ai accès à toutes les listes.

Je spécifie l'ordre dans lequel je souhaite avoir les résultats.

J'inclue l'association avec la table des cartes, qui elle-même est associée à celle des tags.

J'envoie la réponse au format Json.



(DP)

Dossier Professionnel

Dans sa version SQL, la requête aurait été :

```
...  
SELECT * FROM lists  
ORDER BY position ASC, created_at DESC;  
  
SELECT * FROM cards  
INNER JOIN lists ON cards.list_id = lists.id  
ORDER BY cards.position ASC;  
  
SELECT * FROM tags  
INNER JOIN cards_tags ON tags.id = cards_tags.tag_id  
INNER JOIN cards ON cards.id = cards_tags.card_id;
```

On notera que Sequelize nous permet d'éviter trois jointures en tout.

Par ailleurs, si l'utilisateur souhaite afficher une liste en particulier, il fera fonctionner la fonction `getOneList`. Afin d'expliquer une requête préparée, je vais la détailler dans sa version SQL.

```
...  
export async function getOneList(request, response) {  
  const listId = parseInt(request.params.id);  
  const query = `SELECT * FROM list WHERE id = $1`;  
  const results = await client.query(query, [listId]);  
  
  if (results.rows.length === 0) {  
    return response.status(404).json({ error: "List not found" });  
  }  
  
  response.json(results.rows[0]);  
}
```

Je paramètre la requête SQL qui sélectionne une liste spécifique en fonction de son identifiant.

J'exécute la requête avec le client qui interroge la BDD en passant l'identifiant en paramètre.

Je traite le cas où aucune liste n'est trouvée.

J'envoie les détails de la liste trouvée.

2. Précisez les moyens utilisés :

J'ai utilisé plusieurs modules :
Sequelize, Joi, et SanitizeHtml

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole O'Clock.

Chantier, atelier, service ► Projet personnel réalisé pendant la formation.

Période d'exercice ► Du : 11/09/2023 au : 28/02/2024

5. Informations complémentaires (facultatif)



(DP)

Dossier Professionnel

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 :

Nous venons de voir en détail la construction du listController. Nous allons voir comment celui-ci est utilisé dans l'application en suivant le chemin de son architecture.

Avant de créer mon router, un fichier *index.js*, à la racine du projet, permet de créer mon application.

```
import express from "express";
import cors from "cors";

import { router } from "./src/routers/index.js";
import { bodySanitizer } from "./middlewares/body-sanitizer.js";

const app = express();

app.use(cors({ origin: "*" }));

app.use(express.static("./dist"));

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

app.use(bodySanitizer);

app.use(router);

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

J'importe le module *Express* et mon routeur.

Je définis mon application grâce à la fonction *express()*.

Je donne accès aux fichiers static. J'appelle deux *bodyparser* : Le 1^{er} pour traduire le corps des requêtes http
Le 2^{ème} pour travailler en Json.

Je configure *Express app* pour qu'elle utilise mon routeur.

Je lui indique sur quel port est le serveur

La sécurisation de l'accès est apportée par le middleware CORS (Cross-Origin Resource Sharing). Il permet au serveur de spécifier explicitement quels types de requêtes http sont autorisées à accéder à ces ressources avec une origine particulière.

Le projet étant local, j'ai conservé toutes les origines possibles. En production, il faudra restreindre cette partie à notre domaine 'Front' en inscrivant par exemple : `origin: process.env.CORS`

De plus, j'utilise un middleware *bodySanitizer* pour nettoyer les données de requête entrantes avant qu'elles ne soient traitées par le routeur.



(DP)

Dossier Professionnel

Je peux donc coder mon routeur en toute sécurité. Celui-ci est divisé en trois fichiers, un pour chaque table de la BDD. Nous allons nous concentrer sur celui des listes.



```
import { Router } from "express";  
  
import * as listController from "../controllers/listController.js";  
import { controllerWrapper as cw } from "../controllerWrapper.js";  
  
export const router = Router();  
  
router.get("/lists", cw(listController.getAllLists));  
router.get("/lists/:id", cw(listController.getList));  
router.post("/lists", cw(listController.createList));  
router.patch("/lists/:id", cw(listController.updateList));  
router.delete("/lists/:id", cw(listController.deleteList));
```

J'importe le middleware *Router* depuis *Express*.

Puis, tous les Controller présents dans *listController*.

Je définis mon router.

Je déclare les différentes routes nécessaires à mon application. Elles font appel aux fonctions présentes dans *listController*.

On peut remarquer l'utilisation d'un controllerWrapper. Il me permet de centraliser la gestion d'erreur et de ne pas répéter les try/catch :



```
export function controllerWrapper(controllerMdw) {  
  
  return async (request, response, next) => {  
    try {  
      await controllerMdw(request, response, next);  
    } catch (error) {  
      console.error(error);  
  
      response.status(500).json({ error: "Unexpected server error.  
Please try again later." });  
    }  
  };  
}
```



Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

Par ailleurs, ce router nous permet d'utiliser quatre méthodes différentes :

1. La méthode GET :

La route 'router.get("/lists")' permet de récupérer toutes les listes présentent dans la base de données.

La route "/lists/:id" appelle une liste en fonction de son identifiant.

2. La méthode POST :

Elle permet de créer une nouvelle liste et de l'inclure dans la base de données grâce à la fonction *createList* du Controller.

3. La méthode PATCH :

Elle est utilisée lorsque nous avons besoin de modifier une liste déjà présente dans la base de données. C'est pour cela qu'elle a besoin que son identifiant soit précisé dans la route.

4. La méthode DELETE :

Elle supprime une liste de la base de données. Tout comme la route précédente, elle utilise l'identifiant de la liste visée.

2. Précisez les moyens utilisés :

On retrouve majoritairement Express.

Pour la sécurité j'ai également utilisé cors et bodysanitizer

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole O'Clock.

Chantier, atelier, service ► Projet personnel réalisé pendant la formation.

Période d'exercice ► Du : 11/09/2023 au : 28/02/2024

5. Informations complémentaires (facultatif)

Je vous remercie pour votre lecture attentive.



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

Titres, diplômes, CQP, attestations de formation

(facultatif)



MINISTÈRE CHARGÉ
DE L'EMPLOI

(DP)

Dossier Professionnel

Déclaration sur l'honneur

Je soussigné(e) [Méryl Cordier] ,

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 Amiens le 29/03/2024

pour faire valoir ce que de droit.

Signature :



(DP)

Dossier Professionnel

MINISTÈRE CHARGÉ
DE L'EMPLOI

Documents illustrant la pratique professionnelle

(facultatif)



MINISTÈRE CHARGÉ
DE L'EMPLOI

(DP)

Dossier Professionnel

Annexes

(Si le RC le prévoit)

Non prévu dans le RC