



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance* - ARROUES  
*Nom d'usage* -  
*Prénom* - Bernard  
*Adresse* -

## 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>



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# 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. 12
- 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. 24

#### 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.

- CP 5 Créer une base de données. p. 25
- CP 6 Développer les composants d'accès aux données. p. 31
- CP 7 Développer la partie back-end d'une application web ou web mobile. p. 36
- CP 8 Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce . p. 44

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

p. 45

#### Déclaration sur l'honneur

p. 46

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

p. 47

#### Annexes (*Si le RC le prévoit*)

p. 48

# **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 :

Dans le cadre de ce projet, Sergei Brin, le cofondateur de Google m'a contacté pour réaliser son portfolio. N'ayant plus le temps de coder avec ses activités de philanthropes, et voulant tout de même réduire ses coûts au maximum, il a choisi de jeter son dévolu sur un développeur junior.

Après avoir eu un rapide échange de mails avec Sergei (que nous appellerons maintenant "le client"), il a souhaité apporter quelques précisions à son projet.

Le client souhaite que son portfolio présente les caractéristiques communes des portfolios. Il souhaite pouvoir montrer ses projets, afficher son CV et également un formulaire de contact. Il précise également que le site devra être en anglais, reprendre les couleurs de Google, et avoir une bonne accessibilité et de bonnes performances de SEO. Le client souhaite également que son site soit accessible sur les formats mobiles, tablette et desktop.

#### 1. Users stories

Suite à celà, pour valider les demandes du client, j'ai rédigé les users stories pour valider les actions que va pouvoir réaliser un visiteur sur le site. Les users stories permettent de comprendre clairement les besoins des utilisateurs de manière concrète, et de s'assurer que les fonctionnalités développées répondent aux attentes des utilisateurs.

## Users stories

En tant que	Je souhaite	Afin
Utilisateur	Consulter une page d'accueil simple et élégante	d'avoir un aperçu du portfolio
Utilisateur	Naviguer facilement à travers le portfolio	de trouver des informations sur les projets et les compétences
Utilisateur	Voir des images des projets réalisés et cliquer dessus pour obtenir plus d'informations	d'en savoir plus sur chaque projet
Utilisateur	Lire une description détaillée de chaque projet	afin de connaître le projet, son résumé du projet, les technologies utilisées
Utilisateur	Consulter le CV	de voir l'expérience professionnelle et les formations
Utilisateur	Envoyer un message ou une demande de contact à travers un formulaire	de poser des questions ou engager une collaboration

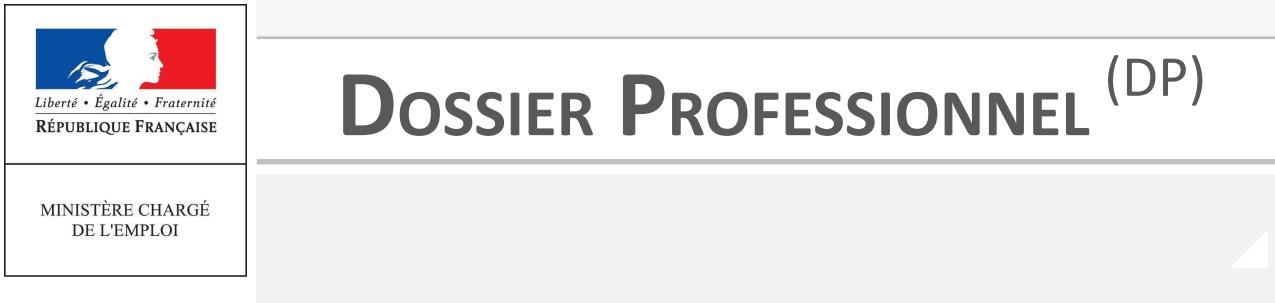
Après un rapide échange de mails avec le client pour vérifier que les users stories étaient bien conformes à ses attentes, j'ai travaillé sur les wireframes du projet.

## **2. Wireframes**

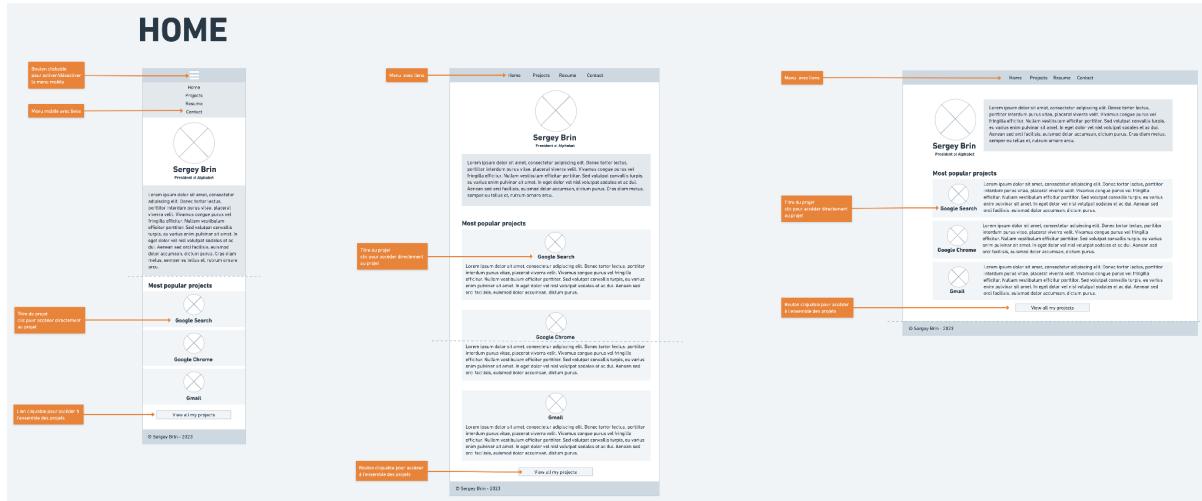
On peut dire que les wireframes sont des schémas simplifiés de la structuration d'un site web, qui permettent de valider visuellement l'idée que se fait le client de son projet. Les wireframes permettent également d'apporter des précisions sur certains éléments de la structuration d'une page, par exemple les boutons cliquables, les images,etc...

Le wireframe se doit d'être simple, généralement en nuance de gris, car il permet simplement de définir la structuration des pages et non pas l'aspect visuel des pages.

Pour ce projet j'ai donc réalisé pour chaque pages du site les wireframes aux formats mobile, tablette et desktop. Pour coller au maximum à l'idée de design mobile-first, j'ai choisi de toujours commencer par le wireframe de la page mobile, puis de continuer sur le format tablette et desktop.



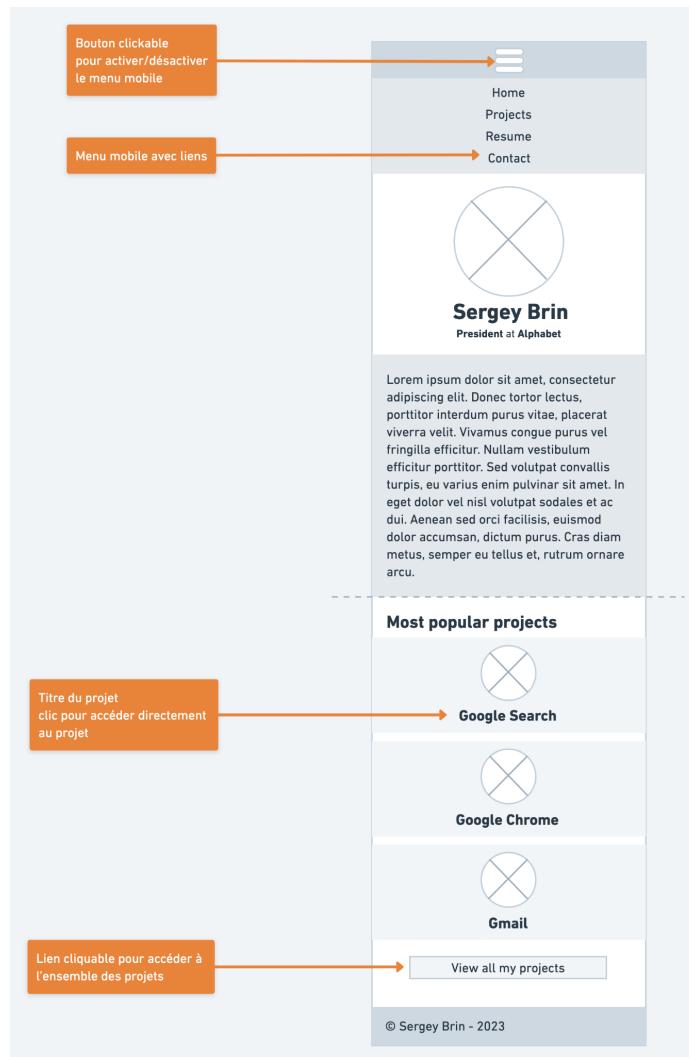
### Wireframe général de la page "Home"



On remarque sur cet exemple la représentation des trois formats spécifiés par le client. Je me suis basé sur le format mobile, puis j'ai décliné la structure de la page mobile aux formats tablette et desktop.

On remarque que les éléments ne changent pas, ils sont simplement réorganisés.

## Wireframe de la page “Home” - format mobile



On peut remarquer plusieurs choses sur ce wireframe:

- L'utilisation d'un séparateur horizontal en pointillés en milieu de page: il représente la fin de la page au format mobile. Pour afficher le contenu situé en dessous de ce séparateur, l'utilisateur devra scroller.
- L'utilisation de “tooltip”: ces informations permettent de préciser des éléments qui pourraient ne pas être clairs lors de la lecture de ce wireframe, notamment les évènements qui peuvent être rattachés à un élément.
- L'affichage du menu mobile: j'ai choisi d'afficher sur la première page la représentation du menu mobile, ainsi que les évènements qui permettent de l'afficher ou de le cacher. Cette représentation du menu mobile n'apparaît pas sur les wireframes.



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

## Wireframe de la page "Home" - format tablette

The wireframe illustrates the layout of the professional dossier home page for tablets. It features a header with a menu bar containing 'Menu avec liens', 'Home', 'Projects', 'Resume', and 'Contact'. Below the menu is a profile section for 'Sergey Brin', President at Alphabet, featuring a placeholder icon and a bio placeholder. A section titled 'Most popular projects' displays three items: 'Google Search' (with a placeholder icon), 'Google Chrome' (with a placeholder icon), and 'Gmail' (with a placeholder icon). Each project item includes a title, a detailed description placeholder, and a 'View all my projects' button. A large orange button on the left side of the page, labeled 'Titre du projet' (click to access directly) and 'Bouton cliquable pour accéder à l'ensemble des projets', points to the 'View all my projects' button.

Menu avec liens

Home Projects Resume Contact

Sergey Brin  
President at Alphabet

Titre du projet  
clique pour accéder directement  
au projet

Bouton cliquable pour accéder  
à l'ensemble des projets

Most popular projects

Google Search

Google Chrome

Gmail

View all my projects

© Sergey Brin - 2023

## Wireframe de la page "Home" - format desktop

This wireframe illustrates the layout of the home page for a desktop user. It features a header with a menu bar containing 'Home', 'Projects', 'Resume', and 'Contact'. Below the header is a profile section for 'Sergey Brin' with a placeholder image and the text 'President at Alphabet'. A callout box highlights the 'Menu avec liens' (links menu) in the top-left corner. The main content area displays 'Most popular projects' with three items: 'Google Search', 'Google Chrome', and 'Gmail', each accompanied by a placeholder icon and a brief description. Another callout box points to a link labeled 'Titre du projet clic pour accéder directement au projet'. A button labeled 'Bouton cliquable pour accéder à l'ensemble des projets' is shown next to a 'View all my projects' button. The footer contains the copyright notice '© Sergey Brin - 2023'.

## Autres exemples de wireframes

Three wireframes are shown side-by-side. The first wireframe, titled 'Project', shows a detailed view of a project page with sections for 'About', 'Stack', and 'Tools'. The second wireframe, titled 'About', shows a general 'About' page with a placeholder image and text. The third wireframe, also titled 'About', shows a more detailed 'About' page with sections for 'About me', 'Stack', and 'Tools'. Each wireframe includes a 'Menu avec liens' callout and a 'View all my projects' button.

Three wireframes are shown side-by-side. The first wireframe, titled 'Resume', shows a general resume page with sections for 'About', 'Experience', and 'Education'. The second wireframe, titled 'Experience', shows a detailed view of the 'Experience' section with two entries for 'Alphabet' and 'Google'. The third wireframe, titled 'Education', shows a detailed view of the 'Education' section with two entries for 'University of Stanford' and 'University of Maryland'. Each wireframe includes a 'Menu avec liens' callout and a 'View all my projects' button.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

Dans le cadre de ce projet, j'ai utilisé [Whimsical](#).

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

## 4. Contexte

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

Chantier, atelier, service ▶ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▶ Du : 01/12/2022 au : 09/02/2023

## 5. Informations complémentaires (facultatif)

Le repo de ce projet est accessible à l'adresse suivante: <https://github.com/elbernie/brin-portfolio>.

Les documents de conception (users stories et wireframes) sont disponibles dans le dossier “[docs](#)” de ce repo.

L'espace de travail Whimsical est disponible à cette adresse:

<https://whimsical.com/portfolio-6TsnN5msqqUAa6T96NcdU>

## 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 :

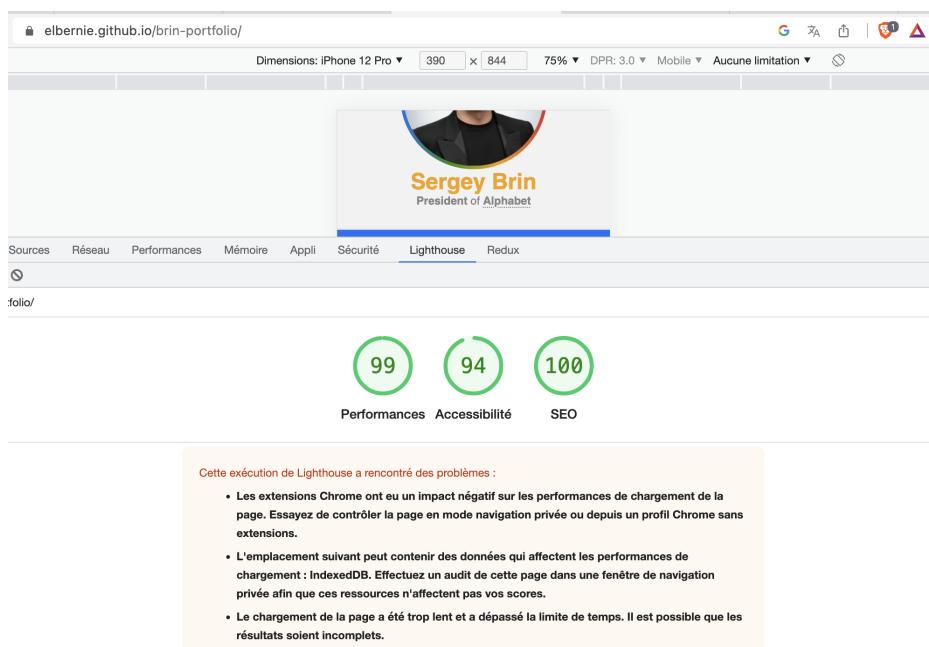
En reprenant les spécifications du projet (CP1), définies par les demandes du client, l'établissement des users stories et des wireframes, je suis passé à la réalisation du projet.

Ce projet définissait plusieurs objectifs. Le premier d'entre eux était d'être accessible et d'utiliser des bonnes pratiques de SEO. Le second objectif étant de réaliser un site responsive, accessible aux formats mobiles, tablettes et desktop.

La réalisation de ce projet se veut être la plus simple possible. J'ai choisi d'utiliser du HTML, du CSS sans frameworks. J'utilise cependant la librairie [normalize.css](#). Concernant l'utilisation de JavaScript, j'ai fait le choix d'en utiliser au minimum. Dans ce projet, JavaScript est uniquement utilisé pour gérer l'ouverture et la fermeture du menu sur mobile.

Pour réaliser ce projet, j'ai décidé d'adopter la stratégie du mobile-first. En effet, du fait de l'utilisation majoritaire et grandissante des mobiles pour naviguer sur le web, il semble judicieux d'adopter cette approche pour réaliser ce projet.

Pour vérifier que le projet répondait bien à ces impératifs, j'ai choisi d'utiliser l'outil Lighthouse intégré à la console de développement du navigateur.



outil de développement Lighthouse



# DOSSIER PROFESSIONNEL (DP)

## 1. Structure des pages

Pour structurer les pages du projet, pour favoriser l'accessibilité et le SEO, j'ai choisi d'utiliser une bonne sémantique HTML.

Voici par exemple la structure générale de la page d'accueil:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  >   <head>...
4  </head>
5
6  <body>
7  >   <header>...
8  </header>
9
10 <main>
11 >   <section class="about">...
12 </section>
13
14 >   <section class="projects">...
15 </section>
16 </main>
17
18 >   <footer>...
19 </footer>
20
21 <script src="/js/menu.js"></script>
22 </body>
23
24 </html>
```

On remarque ici que le corps de la page est divisé en deux grandes parties: la partie *header* et la partie *main*.

La partie *header* contient tout ce qui a trait à la partie supérieure de la page, notamment la navigation.

La partie *main* quant à elle, contient l'ensemble des informations principales de la page. On remarque qu'ici cette partie contient deux *sections*.

```

91      <section class="projects">
92          <h2>Most popular projects</h2>
93      >          <article class="project">...
94          </article>
95      >          <article class="project">...
96          </article>
97      >          <article class="project">...
98          </article>
99
100         <a href=".//projects.html" class="button">View all my projects</a>
101     </section>

```

*Contenu de la section “projets”*

Pour illustrer la structure HTML de la page, on remarque qu’ici la section “projets”, contient une balise *h2*, puis plusieurs articles et enfin un lien pour consulter l’intégralité des projets.

En précision, pour afficher ce bouton j’aurais pu choisir d’encapsuler une balise *button* au sein de la balise *a*, mais celà aurait pu nuire à l’accessibilité pour les personnes en situation de handicap. La balise *a* étant interprétée par les logiciels d’aide comme un changement de page, alors que les boutons sont prévus pour déclencher une action, par exemple soumettre un formulaire.

```

1  <article class="project">
2      <header>
3          
9          <a href=".//projects/search.html"><h3>Google Search</h3></a>
10     </header>
11     <div>
12         <p>
13             Google Search is the most widely used search engine in the world.
14         </p>
15         <p>
16             It's a powerful tool that allows users to quickly and easily find
17             information on the internet. With billions of web pages indexed,
18             Google's search algorithms use advanced techniques to provide the
19             most relevant results to users.
20         </p>
21         <p>
22             The interface is simple and user-friendly, with a search bar that
23             allows users to enter a query and receive results in a matter of
24             seconds. Google Search has become an essential part of our daily
25             lives, helping us find answers to questions, explore new topics,
26             and stay informed about the world around us.
27         </p>
28     </div>
29 </article>

```

*Structure du contenu d’une balise article*

On remarque dans la structure de l’article deux éléments importants, un *header* pour identifier les informations principales, puis une *div* contenant la description des projets.

Le *header* contient notamment une image avec un texte alternatif (*alt*) utile notamment pour les logiciels de lecture d’écran à destination des personnes en situation de handicap, ainsi qu’une balise *h3* pour respecter la sémantique HTML (*h1 > h2 > h3 > ...*)



# DOSSIER PROFESSIONNEL (DP)

## 2. responsive et media queries

Pour suivre la demande d'avoir un style responsive, et en suivant le mobile-first, j'ai utilisé des media queries. Pour avoir un site qui s'adapte au dark mode ou au light mode, ainsi qu'au mode de contraste élevé pour les personnes ayant des difficultés de lecture, j'ai également utilisé des média queries.

```
1 @media screen and (min-width: 1024px) {  
2   main {  
3     align-items: start;  
4     flex-direction: row;  
5     align-self: center;  
6     width: 90vw;  
7     gap: 3em;  
8   }  
9   .informations {  
10    flex-grow: 1;  
11    max-width: 25vw;  
12    margin: 0;  
13  }  
14  .informations address span {  
15    padding: 0;  
16  }  
17  
18  .resume {  
19    flex-grow: 3;  
20  }  
21 }
```

Ici la media query utilise un breakpoint. Ce breakpoint s'applique aux écrans ayant une largeur minimum de 1024px (desktop).

Le CSS change notamment la direction du *display:flex;* en *flex-direction:row;* à la place de *column*

J'ai également mis en place des média queries pour permettre à l'utilisateur d'avoir un thème sombre en fonction de la configuration de son matériel. J'utilise également des media queries si l'utilisateur souhaite avoir un contraste plus élevé, notamment au niveau du texte. Et enfin je combine les deux media queries pour les utilisateurs qui souhaitent avoir un thème sombre ou clair ET un contraste élevé.

```

1  @media (prefers-color-scheme: dark) { Ici la media querie applique un changement de couleur
2      .informations {
3          color: white;
4      }
5
6      .project {
7          background-color: #1f1f1f;
8          color: white;
9      }
10
11     .project h3 {
12         color: white;
13     }
14
15     .resume article {
16         background-color: #1f1f1f;
17         color: white;
18     }
19     .resume article h3,
20     .resume article header > span {
21         color: white;
22     }
23 }

1  @media (prefers-contrast: more) { Cette média querie s'applique lorsque l'utilisateur à un
2      h1 {
3          color: black;
4      }
5      p.about {
6          color: black;
7      }
8      h2 {
9          color: black;
10     }
11 }

1  @media (prefers-color-scheme: dark) and (prefers-contrast: more) { Cette media querie s'applique lorsque l'utilisateur
2      h1 {
3          color: white;
4      }
5      p.about {
6          color: white;
7      }
8      h2 {
9          color: white;
10     }
11     body {
12         background-color: black;
13     }
14 }

```

### 3. Menu mobile - javascript

La seule utilisation du javascript pour ce projet réside dans la gestion de l'ouverture ou la fermeture du menu mobile.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

```
1 const mobileMenuButton = document.querySelector('.burger-menu-container');
2 const menu = document.querySelector('.full-menu');
3
4 let showMenu = false;
5 mobileMenuButton.addEventListener('click', () => {
6     showMenu = !showMenu;
7     menu.style.display = showMenu ? 'none' : 'flex';
8 });


```

Dans ce script, un écouteur d'événement est appliqué sur l'élément `.burger-menu-container`. Si la variable `showMenu` est définie comme étant `false`, alors on inverse l'état de la variable en `true`. Puis on utilise un opérateur ternaire pour déterminer si le menu doit être affiché ou non.

## 2. Précisez les moyens utilisés :

Pour réaliser ce projet, j'ai utilisé **VSCode** avec **Prettier** pour formater le code

Pour vérifier l'accessibilité et les bonnes pratiques de SEO, j'ai utilisé l'outil **Lighthouse** de la console de développement de mon navigateur (Brave). J'ai également utilisé la **console** pour simuler les media queries.

Je me suis référé au site **MDN** (Mozilla Developer Network) pour mieux utiliser les media queries ayant attrait aux préférences de thème et de contraste de l'utilisateur.

Pour déployer le site, j'ai utilisé le système des **Github Pages**, puis j'ai utilisé l'outil **Google Search Console** pour permettre l'indexation du site.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

#### **4. Contexte**

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

Chantier, atelier, service ▶ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▶ Du : *01/12/2022* au : *09/02/2023*

#### **5. Informations complémentaires (facultatif)**

Le repo du projet est accessible ici: <https://github.com/ElBernie/brin-portfolio>

Le projet est accessible à cette adresse: <https://elbernie.github.io/brin-portfolio/>



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 3** - Développer une interface utilisateur web dynamique.

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

Dans le cadre de ce titre professionnel, j'ai réalisé une application de gestion de budget simplifiée. Cette application permet de gérer les dépenses et les revenus d'un utilisateur connecté. L'utilisateur peut créer différents budgets, ainsi que différentes catégories de dépenses. Il peut également ajouter des dépenses et des revenus à ces budgets.

Pour réaliser la partie front-end de l'application, j'ai utilisé la librairie React en y ajoutant les modules suivants:

- **Gestion des routes:** [React Router](#)
- **Gestion du state:** [React Redux](#), [Redux Toolkit](#)
- **Gestion des requêtes:** [axios](#), [React Query](#)
- **Gestion de la traduction:** [react-i18next](#)
- **Gestion des formulaires:** [React Hook Form](#)
- **Gestion des headers HTTP:** [React Helmet](#)
- **Librairie de composants:** [MUI](#)

## 1. Le router

```
1 // CREATION DU ROUTER
2 const router = createBrowserRouter([
3   {
4     path: '/',
5     element: <Root />, // Le composant Root est le composant qui encapsule les routes contenues dans le tableau children
6     children: [
7       {
8         path: '/',
9         /**
10          * Si un token est présent dans le state redux, on affiche la page DashboardPage,
11          * sinon on redirige vers la page de login
12          */
13         element: token ? <DashboardPage /> : <Navigate to="/login" />,
14       },
15       {
16         path: '/budget/:id',
17         element: token ? <BudgetPage /> : <Navigate to="/login" />,
18       },
19       {
20         path: '/categories',
21         element: token ? <CategoriesPage /> : <Navigate to="/login" />,
22       },
23     ],
24   },
25   /**
26    * Ces deux routes sont accessibles sans être authentifié, et ne sont pas encapsulées dans le composant Root
27    */
28   {
29     path: '/login',
30     element: !token ? <LoginPage /> : <Navigate to="/" />,
31   },
32   {
33     path: '/register',
34     element: !token ? <RegisterPage /> : <Navigate to="/" />,
35   },
36 ],
37 );
38 /**
39  * On vérifie que le chargement de l'application est terminé, avant de retourner le composant RouterProvider,
40  * sinon on affiche un message de chargement
41  */
42 /**
43 return loaded ? <RouterProvider router={router} /> : <div>Loading...</div>;
```

Pour créer le routeur, j'utilise React router et sa fonction *createRouterBrowser*. Je définis dans un premier temps les routes principales de l'application qui seront rendues dans le composant *Root*. Ces routes sont uniquement accessibles aux utilisateurs connectés.

Ensuite je gère les routes de connexion et d'inscription, qui sont uniquement accessibles aux utilisateurs non connectés.

Enfin je vérifie que l'application, et notamment le state, sont bien chargés avec la variable *loaded*. Si l'application est chargée, mon composant *App* retourne le routeur sinon il affiche une page de chargement.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## 2. Les requêtes

Pour effectuer les requêtes, j'ai commencé par définir une instance d'**Axios** qui sera réutilisable au sein de notre application, notamment par la librairie **React Query**.

```
1 import axios from 'axios';
2
3 /**
4  * Récupération du JWT dans le local storage
5  */
6 const getToken = () => localStorage.getItem('token') || null;
7
8 /**
9  * Création de l'instance axios.
10 * Cette instance est utilisée pour toutes les requêtes vers l'API
11 */
12 const api = axios.create({
13   baseURL: import.meta.env.VITE_API_URL,
14   headers: {
15     'Content-Type': 'application/json',
16     Accept: 'application/json',
17   },
18 });
19
20 /**
21  * Ajout du token dans le header de chaque requête
22  * Si le token n'est pas présent, on envoie un token vide.
23  * Le backend renverra une erreur 401 si le token est invalide
24 */
25 api.interceptors.request.use((config) => {
26   config.headers.Authorization = `Bearer ${getToken()}`;
27   return config;
28 });
29
30 export default api;
31
```

Ensuite cette instance peut être utilisée au sein des requêtes effectuées dans nos composants, un exemple ici avec l'utilisation de *useMutation* de React Query pour effectuer une requête POST lors de la création d'un nouveau budget.

```

1  function CreateBudgetButton() {
2    const { t } = useTranslation();
3    const [newBudgetName, setNewBudgetName] = useState(null);
4    const [showModal, setShowModal] = useState(false);
5
6    /**
7     * Gestion de requêtes de création d'un budget
8     */
9
10   // on récupère le queryClient avec le hook useQueryClient (react-query)
11   const queryClient = useQueryClient();
12
13   /**
14    * Utilisation du hook useMutation de react-query
15    * Ici on déstructure isLoading et mutate
16    * isLoading est un booléen qui nous permet de savoir si la requête est en cours ou non
17    * mutate est une fonction qui permet d'exécuter la requête de création d'un budget
18    */
19   const { isLoading, mutate } = useMutation({
20     // on définit la fonction de mutation, qui sera appelée avec mutate()
21     // On utilise notre instance axios définie précédemment pour effectuer une requête POST sur l'endpoint /budgets
22     mutationFn: (data) => api.post('/budgets', data),
23
24     // on définit ici ce qu'il se passe en cas de succès de la requête
25     onSuccess: () => {
26       setShowModal(false);
27
28       // on invalide la query 'budgets', gérée par react-query
29       // React-query va refaire la requête nommée 'budgets' et mettre à jour le cache
30       queryClient.invalidateQueries('budgets');
31     },
32   });
33
34   return (
35     <>
36       <Dialog
37         open={showModal}
38         fullWidth
39         maxWidth="sm"
40         onClose={() => setShowModal(false)}
41         <DialogTitle>{t('budget.create.title')}</DialogTitle>
42         <DialogContent>
43           <TextField
44             autoFocus
45             margin="normal"
46             id="name"
47             label={t('budget.create.name.field')}
48             type="text"
49             fullWidth
50             variant="standard"
51             onChange={(e) => setNewBudgetName(e.target.value)}
52           />
53         </DialogContent>
54         <DialogActions>
55           <Button onClick={() => setShowModal(false)}>
56             {t('budget.create.cancelButton')}
57           </Button>
58           <LoadingButton
59             variant="contained"
60             loading={isLoading} // on utilise isLoading pour afficher le loader
61             onClick={() =>
62               // on déclenche la requête de création de budget
63               // en passant les données à envoyer, à savoir le nom du nouveau budget
64               mutate({
65                 name: newBudgetName,
66               })
67             }>
68             {t('budget.create.button')}
69           </LoadingButton>
70         </DialogActions>
71       </Dialog>
72       <Button variant="contained" onClick={() => setShowModal(true)}>
73         {t('budget.create.title')}
74       </Button>
75     </>
76   );
77 }
78
79 export default CreateBudgetButton;

```



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

React Query devient très utile dans le sens où ce dernier gère le cache de nos requêtes. Ici dans le cadre de la création de notre nouveau budget, on indique à React Query d'invalider le cache des précédentes requêtes nommées "budgets", et rafraîchir les composants qui pourraient dépendre des données issues de cette requête.

## 2. Précisez les moyens utilisés :

Pour réaliser ce projet j'ai utilisé [Vite](#), ainsi que les documentations des différentes dépendances de mon projet, notamment les documentations de React Redux, et de React Query

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

## 4. Contexte

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

Chantier, atelier, service ➤ *Projet personnel réalisé en cours de formation.*

Période d'exercice ➤ Du : *01/12/2022* au : *09/02/2023*

## 5. Informations complémentaires (facultatif)

Le repo de ce projet est accessible à cette adresse: <https://github.com/ElBernie/yaba-front>

Le projet est accessible à cette adresse: <https://yaba.bernie.cool/> (email: demo@bernie.cool / mdp: demoL33T)

## Activité-type 1

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

**CP 4** ▶ Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.

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

**2. Précisez les moyens utilisés :**

**3. Avec qui avez-vous travaillé ?**

**4. Contexte**

Nom de l'entreprise, organisme ou association ▶

Chantier, atelier, service ▶

Période d'exercice ▶ Du : \_\_\_\_\_ au : \_\_\_\_\_

**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.

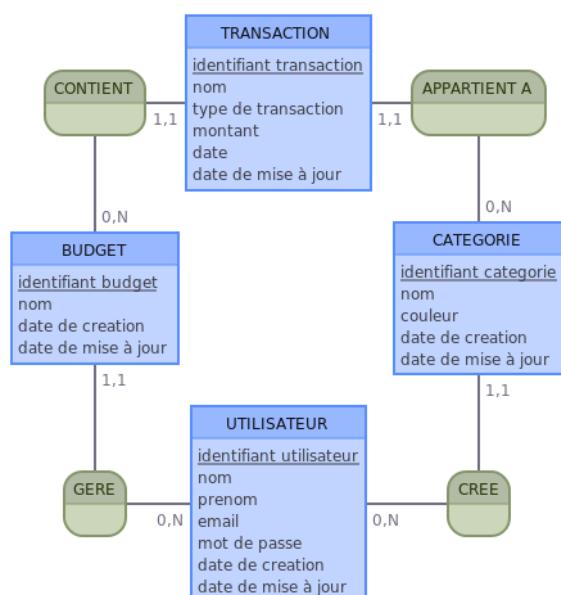
### 1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Comme présenté dans la compétence professionnelle 3, j'ai créé un petit gestionnaire de budget nommé **YABA** (*Yet Another Budget App*).

En premier lieu, il devait être possible pour un utilisateur de créer un compte. Puis chaque utilisateur devait pouvoir créer ses propres budgets, auquel il pouvait ajouter des transactions. Mais pour mieux comprendre les transactions au sein de ses budgets, l'utilisateur devait pouvoir créer et assigner des catégories à ces transactions.

Pour mieux comprendre la structure de mon projet, j'ai donc créé un MCD (*Modèle Conceptuel des Données*).

#### 1. Le MCD



On peut remarquer sur ce MCD les différentes cardinalités entre chaque tables, par exemple:

- Un utilisateur peut avoir *0 ou plusieurs* (*ON*) budgets, de même pour les catégories.
- Un budget et une catégorie peuvent avoir *au minimum et au maximum un* (*1,1*) utilisateur.
- Un budget et une catégorie peuvent avoir 0 ou plusieurs transactions.

On remarque que les cardinalités qui joignent la table budget et catégorie sont des cardinalités *many-to-many* (*ON*), c'est-à-dire que plusieurs budgets peuvent avoir plusieurs catégories et inversement. Dès lors, la table transaction devient de facto **une table de liaison**.

Après avoir défini la représentation visuelle de ma base de données avec le MCD, j'ai créé un **MLD** (*modèle logique des données*) pour représenter les relations entre les tables.

## 2. Le MLD

**BUDGET** (identifiant\_budget, nom, date\_creation,date\_mise\_a\_jour, #identifiant\_utilisateur)

**CATEGORIE** (identifiant\_categorie, nom, couleur, date\_creation, date\_mise\_a\_jour, #identifiant\_utilisateur)

**TRANSACTION** (identifiant\_transaction, type\_transaction, montant, date,date\_mise\_a\_jour, #identifiant\_budget, #identifiant\_categorie)

**UTILISATEUR** (identifiant\_utilisateur, nom, prenom, email, mot\_de\_passe, date\_creation, date\_mise\_a\_jour)

J'ai ensuite créé un **dictionnaire des données** pour représenter les champs des tables.

## 3. Le dictionnaire des données

Pour coller au plus près à la forme finale de ma base de données, j'ai choisi de traduire les champs des tables définis en français dans mon MCD vers l'anglais pour le dictionnaire des données.

user			
FIELD	TYPE	DETAILS	DESCRIPTION
id	INTEGER	PK, NOT NULL, GENERATED ALWAYS AS IDENTITY	user id
name	TEXT		User last name
first_name	TEXT	NOT NULL	User first name
email	TEXT	UNIQUE,NOT NULL	User email
password	TEXT	NOT NULL	User password, hashed
created_at	TIMESTAMPTZ	NOT NULL DEFAULT now()	creation date
updated_at	TIMESTAMPTZ		last update date
budget			
FIELD	TYPE	DETAILS	DESCRIPTION
id	INTEGER	PK, NOT NULL, GENERATED ALWAYS AS IDENTITY	budget id
user_id	INTEGER	NOT NULL REFERENCES "user"(id) ON DELETE CASCADE	foreign key on user table
name	TEXT		Budget name
created_at	TIMESTAMPTZ	NOT NULL DEFAULT now()	creation date
updated_at	TIMESTAMPTZ		last update date
category			
FIELD	TYPE	DETAILS	DESCRIPTION
id	INTEGER	PK, NOT NULL, GENERATED ALWAYS AS IDENTITY	category id
user_id	INTEGER	NOT NULL REFERENCES "user"(id) ON DELETE CASCADE	foreign key on user table
name	TEXT	NOT NULL	Budget name
color	TEXT	NOT NULL	Budget color
created_at	TIMESTAMPTZ	NOT NULL DEFAULT now()	creation date
updated_at	TIMESTAMPTZ		last update date
transaction			
FIELD	TYPE	DETAILS	DESCRIPTION
id	INTEGER	PK, NOT NULL, GENERATED ALWAYS AS IDENTITY	category id
budget_id	INTEGER	NOT NULL REFERENCES "budget"(id) ON DELETE CASCADE	foreign key on budget table
category_id	INTEGER	NOT NULL REFERENCES "category"(id) ON DELETE CASCADE	foreign key on budget table
name	TEXT		Budget name
type	TEXT	NOT NULL	Transaction type (CREDIT/DEBIT)
amount	NUMERIC	NOT NULL	Monetary amount of the transaction
created_at	TIMESTAMPTZ	NOT NULL DEFAULT now()	creation date
updated_at	TIMESTAMPTZ		last update date



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

Puis une fois le dictionnaire des données réalisé, je me suis servi de **PgAdmin** pour réaliser le MPD (*modèle physique des données*).

## 4. Le MPD

*Nota bene: pour des raisons d'organisation, les extraits de code sont séparés, mais représentent l'intégralité du MPD.*

```
1 -----  
2 -- Création des types de données custom  
3 -----  
4  
5 -- Vérification du format d'une adresse email  
6 CREATE DOMAIN email AS text  
7     CONSTRAINT valid_email CHECK ((VALUE ~* '^@[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,}$.')::text);  
8  
9  
10 -- Vérification d'un code couleur hexadécimal (ex: #FFFFFF)  
11 CREATE DOMAIN hex_color AS text  
12     CONSTRAINT hex_color CHECK ((VALUE ~* '^#[A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$.')::text));  
13  
14  
15 -- Vérification du type de transaction, CREDIT ou DEBIT  
16 CREATE DOMAIN transaction_type AS text  
17     CONSTRAINT transaction_type CHECK ((VALUE = ANY (ARRAY['CREDIT'::text, 'DEBIT'::text])));  
--
```

```

1  -- Création de la table budget
2  CREATE TABLE budget (
3      id integer NOT NULL,
4      name text,
5      created_at timestamp with time zone NOT NULL,
6      updated_at timestamp with time zone,
7      user_id integer NOT NULL
8  );
9
10 -- Ajout de la génération automatique de l'id, et de la valeur par défaut de la date de création
11 ALTER TABLE budget ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY;
12 ALTER TABLE budget ALTER COLUMN created_at SET DEFAULT now();
13
14
15
16 -- Création de la table category
17 CREATE TABLE category (
18     id integer NOT NULL,
19     user_id integer NOT NULL,
20     name text NOT NULL,
21     color hex_color NOT NULL,-- on utilise le type de données custom hex_color
22     created_at timestamp with time zone NOT NULL,
23     updated_at timestamp with time zone
24 );
25 ALTER TABLE category ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY;
26 ALTER TABLE category ALTER COLUMN created_at SET DEFAULT now();
27
28
29 -- Création de la table transaction
30 CREATE TABLE transaction (
31     id integer NOT NULL,
32     budget_id integer NOT NULL,
33     category_id integer NOT NULL,
34     name text,
35     type transaction_type NOT NULL,
36     amount numeric NOT NULL,
37     created_at time with time zone NOT NULL,
38     updated_at time with time zone
39 );
40
41 ALTER TABLE transaction ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY;
42 ALTER TABLE transaction ALTER COLUMN created_at SET DEFAULT now();
43
44
45 -- Création de la table user,
46 -- note: on utilise le mot clé "user" qui est un mot clé réservé en SQL, il faut donc l'entourer de guillemets
47 CREATE TABLE "user" (
48     id integer NOT NULL,
49     name text,
50     first_name text NOT NULL,
51     email email NOT NULL, -- on utilise le type de données custom email
52     password text NOT NULL,
53     created_at timestamp with time zone NOT NULL,
54     updated_at time with time zone
55 );
56
57 ALTER TABLE "user" ALTER COLUMN id ADD GENERATED ALWAYS AS IDENTITY;
58 ALTER TABLE "user" ALTER COLUMN created_at SET DEFAULT now();
59 -- on ajoute une contrainte d'unicité sur l'email
60 ALTER TABLE IF EXISTS "user" ADD CONSTRAINT email_unique UNIQUE (email);
61

```



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

```
1 -- Création des clés primaires
2 ALTER TABLE ONLY budget
3     ADD CONSTRAINT budget_pkey PRIMARY KEY (id);
4
5 ALTER TABLE ONLY category
6     ADD CONSTRAINT category_pkey PRIMARY KEY (id);
7
8 ALTER TABLE ONLY transaction
9     ADD CONSTRAINT transaction_pkey PRIMARY KEY (id);
10
11 ALTER TABLE ONLY "user"
12     ADD CONSTRAINT user_pkey PRIMARY KEY (id);
13
14
15 -- Création des clés étrangères
16 CREATE INDEX fk1_budget_fkey ON transaction USING btree (budget_id);
17
18 CREATE INDEX fk1_category_fkey ON transaction USING btree (category_id);
19
20 CREATE INDEX fk1_user_fkey ON budget USING btree (user_id);
21
22
23 -- création des relations au sein des tables
24 ALTER TABLE ONLY transaction
25     ADD CONSTRAINT budget_fkey FOREIGN KEY (budget_id) REFERENCES budget(id) ON DELETE CASCADE;
26
27 ALTER TABLE ONLY transaction
28     ADD CONSTRAINT category_fkey FOREIGN KEY (category_id) REFERENCES category(id) ON DELETE CASCADE;
29
30 ALTER TABLE ONLY budget
31     ADD CONSTRAINT user_fkey FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE;
32
33 ALTER TABLE ONLY category
34     ADD CONSTRAINT user_fkey FOREIGN KEY (user_id) REFERENCES "user"(id) ON DELETE CASCADE;
```

## 2. Précisez les moyens utilisés :

Pour réaliser le MCD, j'ai utilisé [Mocodo](#), puis j'ai utilisé PgAdmin pour m'assister dans la réalisation du MPD.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

#### **4. Contexte**

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

Chantier, atelier, service ▶ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▶ Du : *01/12/2022* au : *09/02/2023*

#### **5. Informations complémentaires (facultatif)**

Documents de conception: <https://github.com/ElBernie/yaba-back/tree/main/docs>

Le projet est accessible à cette adresse: <https://yaba.bernie.cool/> (email: demo@bernie.cool / mdp: demoL33T)



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 :

Toujours dans le cadre du projet YABA (CP3 et CP5), j'ai implémenté l'architecture avec NodeJS et Express. J'ai choisi de ne pas utiliser d'ORM et d'effectuer des requêtes SQL directement au serveur Postgres avec le module [node-postgres](#), en utilisant également le paquet [node-pg-format](#).

#### 1. Initialisation de la connexion

Pour initialiser la connexion au serveur Postgres, j'ai créé un fichier db.js gérant le pool de connexion.

```
1 import pg from 'pg';
2 /**
3  * Création du pool de connexion à la base de données
4  * =====
5  * Je ne passe pas de paramètre, node-postgres charge automatiquement
6  * les variables d'environnement définies dans le fichier .env
7  */
8 const pool = new pg.Pool();
9
10 // On exporte le pool de connexion pour le réutiliser
11 export default pool;
```

#### 2. Utilitaire de construction et d'exécution de requêtes.

Pour améliorer mon API, je souhaitais pouvoir interpréter les paramètres de la query string des requêtes pour la transformer en requête SQL. J'ai donc créé un utilitaire (*query.service.js*) qui permet d'analyser la query string, puis d'exécuter la requête, et ce de façon sécurisée pour éviter les injections SQL.

J'ai donc créé une première fonction *clauseBuilder*, qui permet de traiter les paramètres *sort*, *limit*, *order* et *where*.

```

1 clauseBuilder: (queryString) => {
2     // Je transforme la query string en objet
3     // les paramètres séparés par des virgules sont transformés en tableau
4     // ex: ?sort=-name,createdAt devient { sort: ['-name', 'createdAt'] }
5     const query = qs.parse(queryString, { comma: true });
6
7     /**
8      * SORT
9      */
10
11    // Je construis la clause ORDER BY, vide par défaut
12    let sort = '';
13    // Si la query contient un paramètre sort
14    if (query.sort) {
15        // Je construis un tableau de paramètres de tri
16        const sortItems = query.sort.map((item) => {
17            // Si le paramètre commence par un tiret, je le transforme en DESC
18            if (item.startsWith('-')) {
19                return `${item.slice(1)} DESC`;
20            }
21            // Sinon, je le transforme en ASC
22            return `${item} ASC`;
23        });
24        // Je construis la clause ORDER BY, en utilisant la fonction format de pg-format
25        // pour empêcher les injections SQL
26        sort = format.withArray('ORDER BY %s', sortItems);
27    }
28
29    /**
30     * LIMIT
31     */
32    let limit = '';
33    if (queryString.limit) {
34        limit = `LIMIT ${queryString.limit}`;
35    }
36
37    /**
38     * FILTERS
39     */
40
41    // Je supprime les paramètres sort et limit de l'objet query
42    // pour ne garder que les filtres
43    const queryFilters = query;
44    delete queryFilters.sort;
45    delete queryFilters.limit;
46
47    // Je construis la clause WHERE, vide par défaut
48    let filters = '';
49    // Si la query contient des filtres
50    if (queryFilters && Object.keys(queryFilters).length > 0) {
51        // Je construis un tableau de paramètres de filtre
52        const filterItems = Object.keys(queryFilters).map((item) => {
53            return `${item} = ${queryFilters[item]}`;
54        });
55
56        /**
57         Je construis la clause WHERE,
58         j'effectue un join sur le tableau de paramètres de filtre
59         pour obtenir une chaîne de caractères séparée par des AND (sql)
60         puis je formate la chaîne avec la fonction format de pg-format
61         pour empêcher les injections SQL
62         ex: ['name = "John"', 'age = 25'] devient 'name = "John" AND age = 25'
63        */
64
65        filters = format('WHERE %s', filterItems.join(' AND '));
66    }
67
68    return {
69        filters,
70        sort,
71        limit,
72        query: `${filters} ${sort} ${limit}`,
73    };
74},

```



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

La fonction *clauseBuilder* est utilisée au niveau du controller de requête. La fonction prend en paramètre la query string de la requête, puis obtient un objet en sortie.

Le paramètre *query* de cet objet peut ensuite être passé dans le paramètre *clauses* de la fonction *execute*.

La fonction *execute*, va concaténer la requête SQL de la couche d'accès avec les clauses de requête, en ajoutant les paramètres de la requête (*values*) puis exécute la requête de manière sécurisée.

```
1  execute: async ({ query, values, clauses }) => {
2    // je formate la requête avec les clauses
3    // pour obtenir une requête complète et éviter les injections SQL
4    const queryString = format(
5      `%s %s;
6      `,
7      query,
8      clauses
9    );
10   // j'exécute la requête, avec les valeurs associées si besoin
11   // j'utilise les queries paramétrées de node-postgres pour éviter les injections SQL
12   return pool.query(queryString, values);
13 },
14 }
```

### 3. La couche d'accès

Pour créer la couche d'accès aux données, j'ai créé un objet pour chaque entité de ma base de données, par exemple:

```
1  import queryBuilder from '../services/query.service.js';
2
3  const Budget = {
4    > getAll: async (options) => { ... },
5    > getOne: async (id) => { ... },
6    > create: async (budget) => { ... },
7    > update: async (id, budget) => { ... },
8    > delete: async (id) => { ... },
9  };
10
11 export default Budget;
```

Et chaque objet d'entité contient des méthodes pour modifier les données en base de données, exécutées avec la fonction `execute` définie précédemment.

```
4  getAll: async (options) => {
5      // je récupère uniquement les lignes de la table budget qui correspondent aux filtres
6      const { rows } = await queryBuilder.execute({
7          // j'ajoute une colonne amount à ma requête,
8          // c'est la somme des montants des transactions liées au budget, sur la table transaction
9          // si la somme est nulle, je retourne 0 (COALESCE)
10         query: `
11             SELECT *,
12                 (SELECT COALESCE(SUM(amount),0)
13                  FROM transaction
14                  WHERE transaction.budget_id = budget.id
15                  ) as amount
16             FROM budget
17             `,
18             // je passe les valeurs de la requête en paramètre, issue de queryBuilder
19             clauses: options.query,
20         });
21     // je retourne les lignes de la table budget
22     return rows;
23 },
```

*exemple de la fonction execute, avec des clauses*

```
41  create: async (budget) => {
42      const { rows } = await queryBuilder.execute({
43          query: 'INSERT INTO budget (user_id, name) VALUES ($1, $2) RETURNING *',
44          values: [budget.userId, budget.name],
45          clauses: undefined,
46      });
47      return rows[0];
48 },
```

*exemple de la fonction execute, sans clauses, mais avec des paramètres*

## 2. Précisez les moyens utilisés :

Dans le cadre de ce projet, et pour l'accès aux données j'ai utilisé les modules node-postgres, node-postgres-format, et [qs](#) pour parser la query string.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

## 4. Contexte

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

Chantier, atelier, service ➤ *Projet personnel réalisé en cours de formation.*



RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

Période d'exercice      ▶ Du : 01/12/2022      au : 09/02/2023

## 5. Informations complémentaires (facultatif)

Le repo du projet est accessible à cette adresse: <https://github.com/elbernie/yaba-back>

Le projet est accessible à cette adresse: <https://yaba.bernie.cool/> (email: demo@bernie.cool / mdp: demoL33T)

## 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 :

Pour m'exercer, j'ai souhaité réaliser un projet assez simple: un raccourcisseur d'URL. J'ai nommé ce projet YAUS (*Yet Another URL Shortener*).

Au niveau du back-end de cette application, j'ai également utilisé Express. Pour réaliser l'API de mon service, j'ai commencé par créer un routeur.

Le chemin de la requête au niveau des routes est le suivant: requête → routeur principal → router d'entité → middlewares → controller.

#### 1. Initialisation du serveur

```
7 dotenv.config();
8
9 const PORT = process.env.API_PORT ?? 3000;
10
11 // On déclare notre application express
12 const app = express();
13
14 // Utilisation du service de limitation des requêtes
15 app.use(Limiter.base);
16
17 app.use(express.json());
18
19 // On déclare notre routeur
20 app.use(router);
21
22 // middleware de gestion des erreurs
23 app.use(ErrorMiddleware);
24
25 // on fait écouter notre application sur le port défini
26 app.listen(PORT, () => {
27   console.log(`Server is running on port ${PORT}`);
28 });
29
30 export default app;
```

Ici on déclare notre serveur.

On remarque l'utilisation d'un *rate-limiter* pour limiter les requêtes sur notre serveur.

On remarque également l'utilisation de notre routeur principal en tant que middleware. Ce routeur va se charger de gérer les requêtes effectuées.

#### 2. Routeur principal

```
5 // Je déclare un router principal
6 const router = Router();
7
8 router.use('/links', linksRouter);
9 router.use('/auth', authRouter);
10
11 export default router;
```

Je déclare dans le fichier *src/routes/router.js* le routeur de notre API, puis au sein de ce fichier, je viens utiliser d'autres routeurs, liés aux entités, que je viens appliquer sur les endpoints.



# DOSSIER PROFESSIONNEL (DP)

### 3. Router d'entité

```
7 / On déclare un routeur, utilisable en tant que middleware
8 const linksRouter = Router();
9
10 linksRouter.get('/:alias', UserLinkController.getOneByAlias);
11 linksRouter.post(
12   '/',
13   Validation(LinkValidation.create, 'body'),
14   authentication({
15     authRequired: false,
16   }),
17   UserLinkController.create
18 );
19 linksRouter.patch(
20   ':id',
21   Validation(LinkValidation.update, 'body'),
22   authentication(true),
23   UserLinkController.update
24 );
25 linksRouter.delete('/:id', authentication(true), UserLinkController.delete);
26
27 export default linksRouter;
```

Dans ce fichier `src/routes/links.route.js` je vais déclarer les endpoints de la ressource **Links**, avec les méthodes associées (get, post, patch, delete,etc). On remarque l'utilisation de middlewares, que j'expliquerai plus tard.

Avec l'organisation de notre routeur, par exemple pour récupérer un lien, on va faire une requête GET à l'adresse suivante: [https://mon-domaine.fr/links/\[alias-du-lien\]](https://mon-domaine.fr/links/[alias-du-lien])

#### 4. Controller

Une fois la requête effectuée, c'est le controller qui va prendre le relais et traiter la partie "logique" de notre API, il va traiter les données de la requête puis interagir avec la couche des données de notre API et enfin renvoyer des données en retour. En voici un exemple avec le controller de création de lien, contenu dans le fichier `src/controllers/userlinks.controller.js`

```
29  create: async (req, res, next) => {
30    try {
31      const { url, alias } = req.body;
32      const link = await Link.getOneByUrl(url);
33      if (!link) {
34        const newLink = await Link.create(url);
35        const userLink = await UserLink.create({
36          user: req.user?.id || null,
37          link: newLink.id,
38          alias,
39        });
40        res.json(userLink);
41      } else {
42        const userLink = await UserLink.create({
43          user: req.user?.id || null,
44          link: link.id,
45          alias,
46        });
47        res.json(userLink);
48      }
49    } catch (error) {
50      // si une erreur est levée,
51      // Je passe l'erreur dans la fonction next
52      // Le middleware d'erreur va détecter automatiquement l'erreur et la traiter.
53      next(error);
54    }
55  },
```

Pour information, les liens et les alias des liens sont stockés dans deux tables différentes de la base de données. La table Link contient simplement l'URL, tandis que la table Userlink contient une référence à l'enregistrement de la table Link ainsi qu'une référence à l'utilisateur ayant créé le lien.

Ici le controller va simplement vérifier que ce lien est déjà présent dans la table Link. Si il est présent, le controller va créer une référence à ce lien dans la table Userlink, sinon il va créer le lien dans la table Link et créer une référence dans la table Userlink.



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

## 5. Middlewares

Pour assurer la gestion de l'authentification d'un utilisateur ou pour valider les données, j'ai créé des middleware que je passe au niveau du router, en voici un exemple:

```
11  linksRouter.post(
12    '/',
13    Validation(LinkValidation.create, 'body'),
14    authentication({
15      authRequired: false,
16    }),
17    UserLinkController.create
18  );
```

Pour gérer l'authentification de l'utilisateur, j'ai créé le middleware suivant:

```
4 /**
5  * Middleware d'authentification
6  * Il accepte un objet en paramètre, puis retourne une fonction middleware
7  */
8 const authentication =
9 ({ authRequired = true } ) =>
10 (req, res, next) => {
11   /**
12    * Si l'authentification est requise et que l'utilisateur n'a pas fourni de token
13    */
14  if (authRequired && !req.headers.authorization) {
15    res.status(401).send({ message: 'MISSING_TOKEN' });
16  }
17
18  /**
19   * Si l'authentification n'est pas requise et que l'utilisateur n'a pas fourni de token, on accepte la requête
20   */
21  if (!authRequired && !req.headers.authorization) {
22    next();
23  }
24
25  /**
26   * Si l'authentification est requise et que l'utilisateur a fourni un token, on vérifie que le token est valide
27   */
28  if (
29    req.headers &&
30    req.headers.authorization &&
31    req.headers.authorization.startsWith('Bearer ')
32  ) {
33    const jwt = req.headers.authorization.split(' ')[1]; // on découpe la string en 2 parties, à partir du premier espace, puis on sélectionne directement la 2ème partie découpée
34
35    try {
36      const verifiedJWT = Jwt.verify(jwt, process.env.JWT_SECRET); // on vérifie que le JWT est valide et n'a pas été modifié
37
38      // on attache les données du JWT décryptées à la requête, pour pouvoir l'utiliser dans nos routes
39      req.user = {
40        id: verifiedJWT.id,
41      };
42
43      next();
44    } catch (jwtVerificationError) {
45      // si le JWT n'est pas valide, on renvoie une erreur
46      res.status(401).json({ message: 'INVALID_TOKEN' });
47    }
48  }
49 }
50 export default authentication;
```

Ce middleware va vérifier si pour une route donnée l'authentification est obligatoire ou non. Si un token est passé dans les headers avec l'élément "Authorization", on vérifie tout d'abord que le token suit la forme "Bearer [jwt]".

On vérifie ensuite que le JWT est bien valide, avec la librairie [jsonwebtoken](#). Si le token est valide, on attache simplement à l'objet de requête l'id de l'utilisateur.

Pour sécuriser les données envoyées par l'utilisateur à l'API, j'ai choisi d'utiliser la librairie **Joi**.

J'ai créé un middleware, qui, tout comme le middleware d'authentification, accepte des paramètres. Le middleware de validation a deux paramètres: un schéma et une clé.

Le schéma est défini dans un autre fichier et permet de décrire et de valider les données envoyées, tandis que la clé nous permet de définir quel élément de la requête nous souhaitons valider. Par exemple, si je souhaite valider les données contenues dans req.body, je vais passer la clé "body".

```
1  /**
2   * Middleware de validation des données d'une requête
3   * Ce middleware prend en paramètre un schéma de validation et l'élément de la requête à valider
4   */
5
6  const Validation = (schema, key) => (req, res, next) => {
7    // si on ne trouve pas l'élément à valider dans la requête, on renvoie une erreur
8    if (!req[key]) {
9      return res.status(400).json({ error: `missing ${key} in request object.` });
10    }
11
12    // on valide les données de la requête avec le schéma de validation
13    const { error } = schema.validate(req[key]);
14    // si la validation échoue, on renvoie une erreur
15    if (error) {
16      return res
17        .status(400)
18        .json({ error: `${error.details[0].message} in req.${key}` });
19    }
20    // si la validation réussit, on passe au middleware suivant
21    next();
22  };
23
24  export default Validation;
```

Je définis ensuite les schémas au sein du dossier *validations*, voici un exemple du schéma de validation de la création d'un lien



# DOSSIER PROFESSIONNEL (DP)

```
4  create: Joi.object({
5      url: Joi.string()
6          .required()
7          .min(3)
8          .regex(/^(https?:\/\/)?([\da-z.-]+)\.([a-z.]{2,6})([/\w .-]*\/?$/)) // regex pour valider une URL
9          .messages({
10              'string.empty': 'URL is required',
11              'string.min': 'URL must be at least 3 characters',
12              'string.pattern.base': 'URL must be a valid URL',
13          }),
14      alias: Joi.string()
15          .max(20)
16          .regex(/^[\w-]+$/) // regex pour valider un alias (lettres, chiffres, tirets uniquement)
17
18          .required()
19          .messages({
20              'string.empty': 'Alias is required',
21              'string.max': 'Alias must be less than 20 characters',
22              'string.pattern.base':
23                  'Alias must contain only letters, numbers, and hyphens',
24          }),
25      }).required(),
```

Ensuite pour gérer de façon “propre” les erreurs de mon API, j’ai créé un middleware de gestion d’erreur

```
1  const ErrorMiddleware = (err, req, res, _next) => {
2      // Si l'erreur est une erreur de validation
3      if (err.name === 'ValidationError') {
4          // On renvoie un code 400 (bad request) et le message d'erreur
5          return res.status(400).json({ error: err.message });
6      }
7
8      // sinon on renvoie un code 500 (internal server error) et le message d'erreur
9      res.status(500).json({ error: err.message });
10 };
11 export default ErrorMiddleware;
12
```

Ce type de middleware est un peu particulier car il accepte 4 paramètres. Lors de la présentation de l’exemple de controller précédemment, on pouvait voir que dans ma gestion d’erreur je passais l’erreur au middleware suivant via la fonction `next()`. Express va alors détecter automatiquement le passage de l’erreur et va gérer l’erreur via mon middleware.

## 6. Rate limiting

Enfin, pour rajouter une couche supplémentaire de sécurité, j'ai décidé d'ajouter un rate limiter. Le rate limiter permet de limiter le nombre de requêtes à l'API. J'ai donc créé différents types de limiter en fonction des cas d'utilisation.

```
1 import rateLimit from 'express-rate-limit';
2
3 const Limiter = {
4   // Je crée un limiter par défaut pour toutes les routes
5   base: rateLimit({
6     windowMs: 60 * 1000, // 1 minute
7     max: 100, // 100 requêtes, par minute, par IP
8   }),
9
10  // limiter pour la création de compte, 1 par minute
11  accountCreation: rateLimit({
12    windowMs: 60 * 1000,
13    max: 1,
14    message: 'You have exceeded the 1 account creation per minute limit!',
15  }),
16
17  // limiter pour la connexion, 5 par minute
18  accountLogin: rateLimit({
19    windowMs: 60 * 1000,
20    max: 5,
21    message: 'You have exceeded the 5 account login attempts per minute limit!',
22  }),
23 };
24
25 export default Limiter;
```

J'ai créé un limiter généraliste que j'utilise généralement sur toutes les requêtes faites à mon serveur (cf: premier exemple de cette compétence - initialisation du serveur). Puis j'ai créé deux autres limiters que j'applique au niveau des routes `/auth/register` et `/auth/login`.

Le premier limiter concerne la limitation de création de comptes à une seule création de compte par minute, par IP. Le second permet de limiter les tentatives de connexion d'un utilisateur à 5 par minute et par IP, de façon à éviter les attaques par brute force.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

Pour construire ce projet j'ai utilisé les librairies suivantes:

- [Express](#) et sa documentation
- [express-rate-limit](#): pour gérer le rate limiting.
- [jsonwebtoken](#) pour construire et valider les tokens d'authentification
- [Joi](#) pour gérer la validation des données

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet

## 4. Contexte

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

Chantier, atelier, service ▶ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▶ Du : 01/12/2022 au : 09/03/2023

## 5. Informations complémentaires (facultatif)

Le repo de projet est accessible à cette adresse: <https://github.com/elbernie/yaus-back>

## Activité-type 2

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

**CP 8** ▶ *Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.*

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

**2. Précisez les moyens utilisés :**

**3. Avec qui avez-vous travaillé ?**

**4. Contexte**

Nom de l'entreprise, organisme ou association ▶

Chantier, atelier, service ▶

Période d'exercice ▶ Du : \_\_\_\_\_ au : \_\_\_\_\_

**5. Informations complémentaires (facultatif)**



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

## **Titres, diplômes, CQP, attestations de formation**

(facultatif)

## Déclaration sur l'honneur

Je soussigné(e) Bernard ARROUES ,

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

Fait à le 17/03/2023

pour faire valoir ce que de droit.

Signature :



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

## Documents illustrant la pratique professionnelle

*(facultatif)*

## **ANNEXES**

*(Si le RC le prévoit)*