

## 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. 13
- CP 3 Développer une interface utilisateur web dynamique. p. 19

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

p. 22

- CP 5 Créer une base de données. p. 22
- CP 6 Développer les composants d'accès aux données. p. 25
- CP 7 Développer la partie back-end d'une application web ou web mobile. p. 30

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

p. 35

#### Déclaration sur l'honneur

p. 36

# **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 :

J'ai réalisé les wireframes et les users stories du Minimum Viable Product d'un site de recettes de cuisine en Responsive Web Design. Ces documents ont été créés d'après les attentes d'un client fictif.

#### Users stories:

Les user stories permettent de décrire les fonctionnalités du site à travers les actions que les utilisateurs du site pourront effectuer.

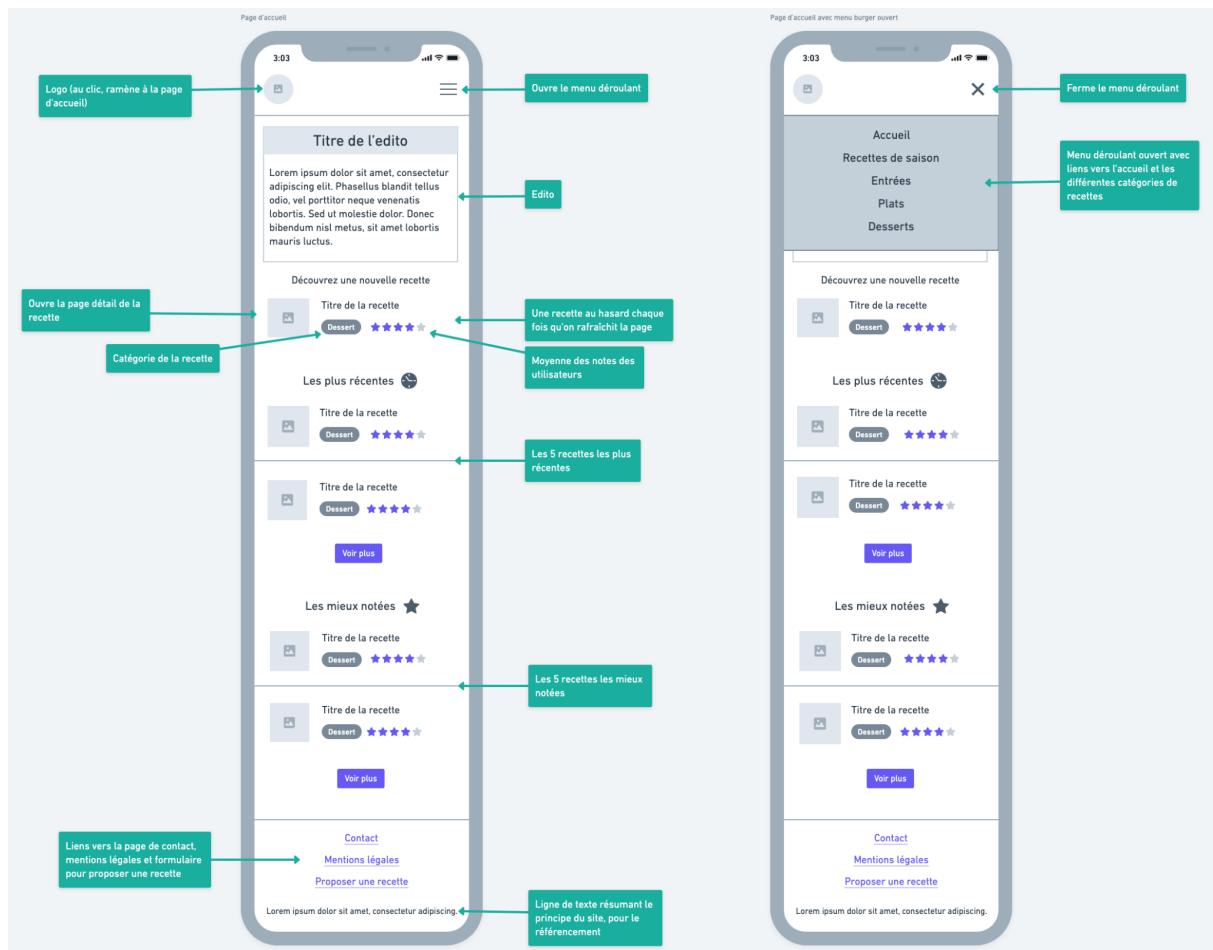
En tant que..	Je veux...	Afin de...
Visiteur	accéder à la liste des ingrédients d'une recette	vérifier si j'ai les ingrédients nécessaires ou les acheter
Visiteur	accéder aux étapes de la recette	réaliser celle-ci dans ma cuisine
Visiteur	cliquer sur le logo du header	retourner à la page d'accueil
Visiteur	cliquer sur les liens du header	naviguer sur les différentes pages de recettes du site
Visiteur	cliquer sur les liens du footer	naviguer sur les différentes pages du site
Visiteur	cliquer sur une recette	voir les ingrédients et étapes de cette recette
Visiteur	voir une recette au hasard différente à chaque fois sur la page d'accueil	essayer une nouvelle recette

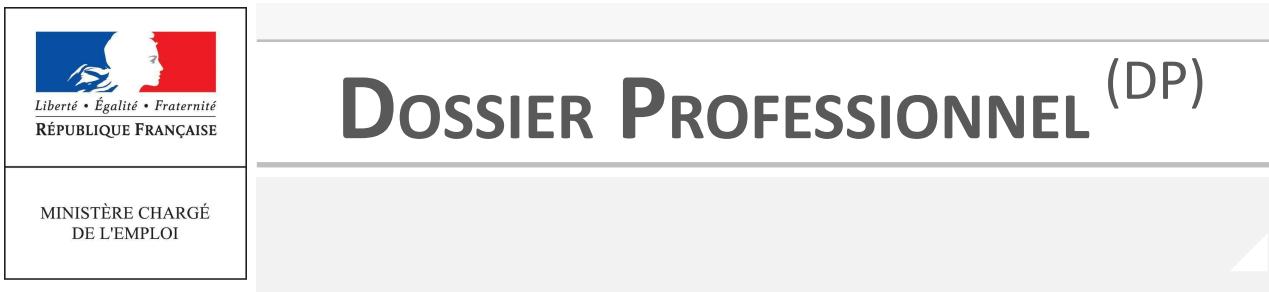
Visiteur	voir les 5 recettes les plus récentes sur la page d'accueil	découvrir des nouvelles recettes
Visiteur	voir les 5 recettes les mieux notées sur la page d'accueil	découvrir des recettes populaires
Visiteur	voir l'édition sur la page d'accueil	le consulter

### Wireframes:

Les wireframes ont pour but de présenter schématiquement le rendu visuel du site au client, afin de valider avec lui l'emplacement des éléments. Ce site est réalisé en Responsive Web Design, j'ai donc décliné ces wireframes en trois versions: mobile, tablette et desktop.

- Page d'accueil:





# DOSSIER PROFESSIONNEL (DP)

Page d'accueil

Accueil Recettes de saison Entrées Plats Desserts

**Titre de l'editho**

Accueil Recettes de saison Entrées Plats Desserts

**Titre de la recette**

Dessert ★★★★☆

**Les plus récentes**

**Les mieux notées**

**Liens vers la page de contact, mentions légales et formulaire pour proposer une recette**

**Liens vers l'accueil et les différentes catégories de recettes**

**Edito**

**Ouvre la page détail de la recette**

**Catégorie de la recette**

**Une recette au hasard chaque fois qu'on rafraîchit la page**

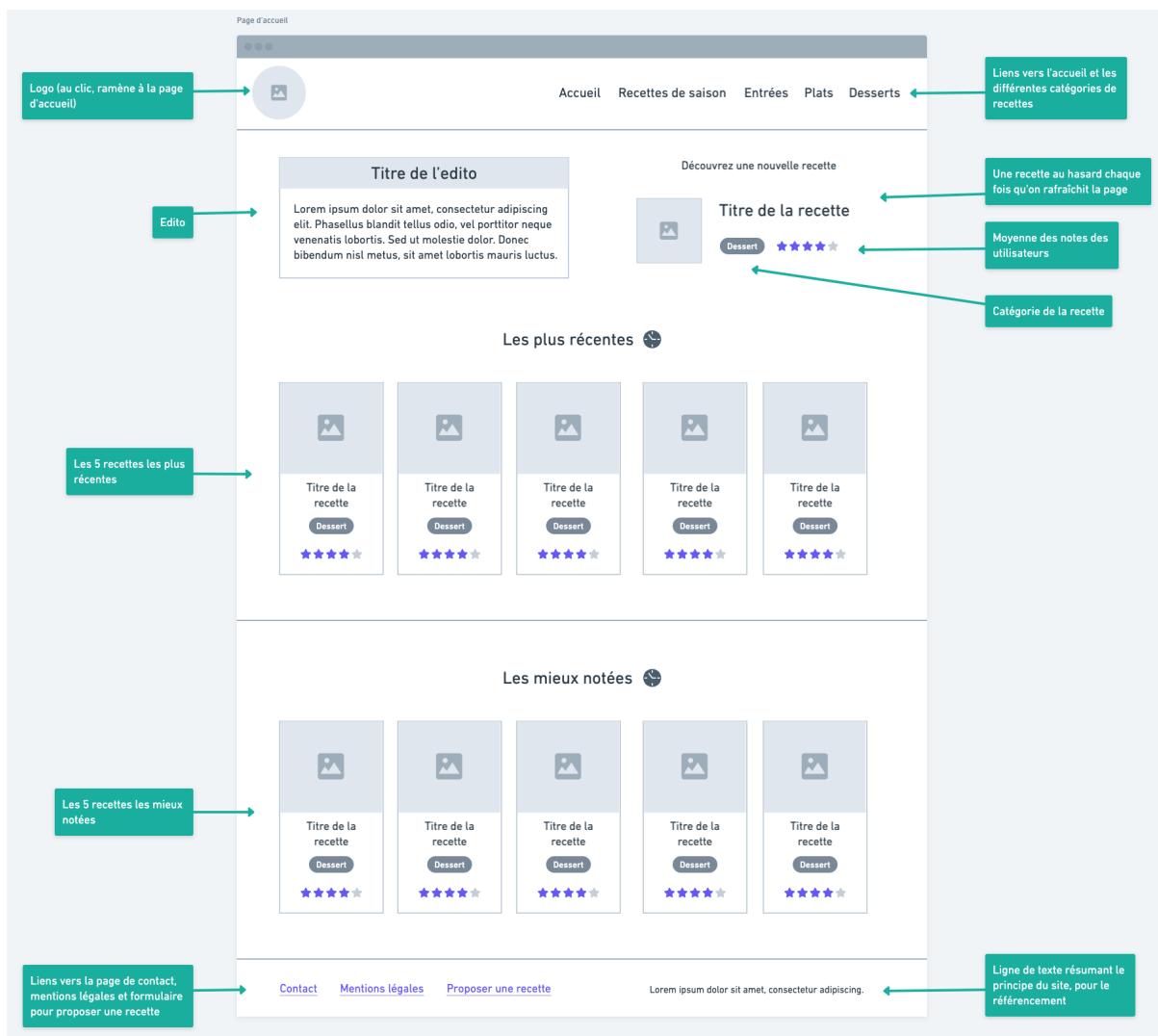
**Moyenne des notes des utilisateurs**

**Les 5 recettes les plus récentes**

**Les 5 recettes les mieux notées**

**Ligne de texte résumant le principe du site, pour le référencement**

Contact Mentions légales Proposer une recette Lorem ipsum dolor sit amet, consectetur adipiscing.



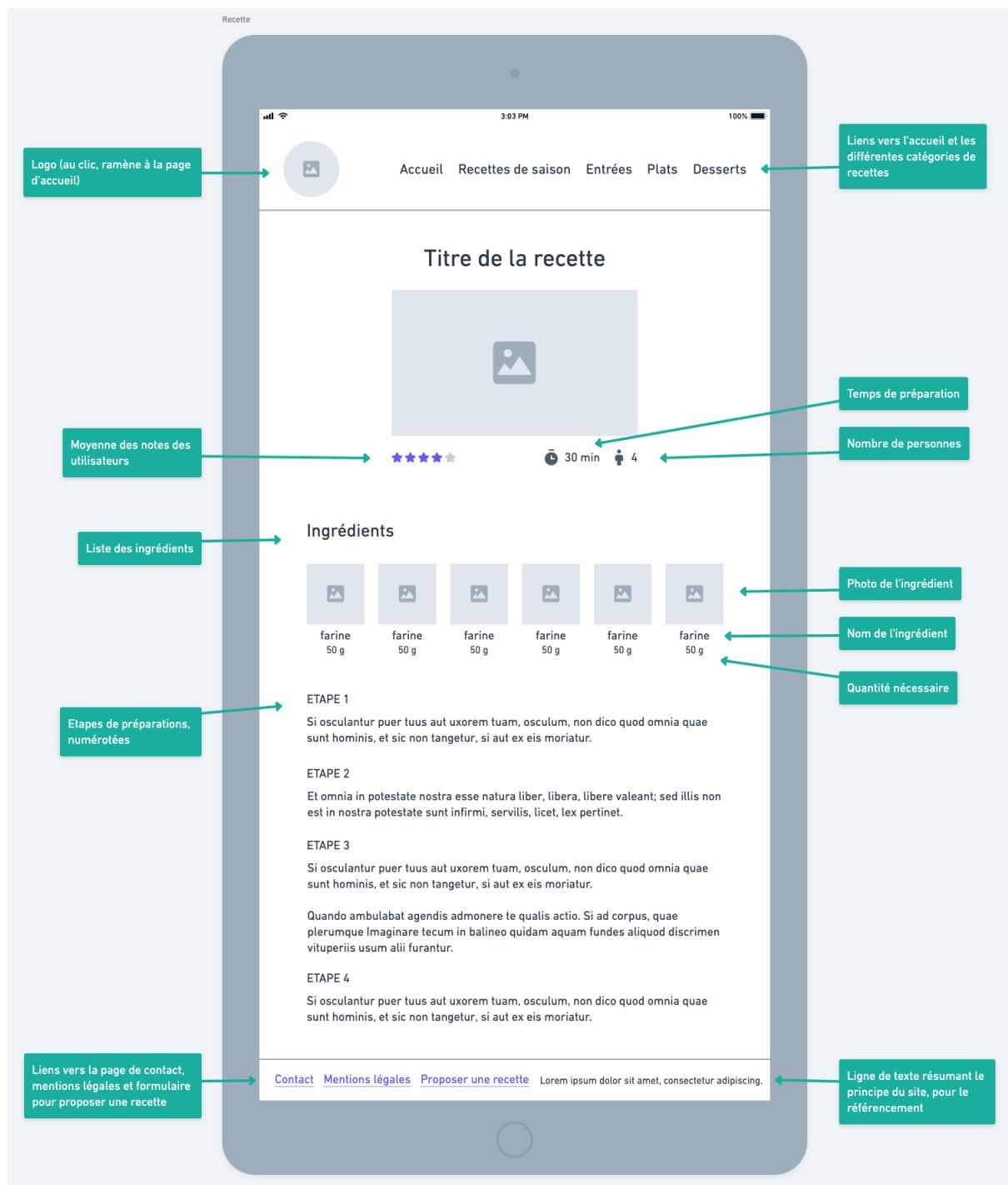


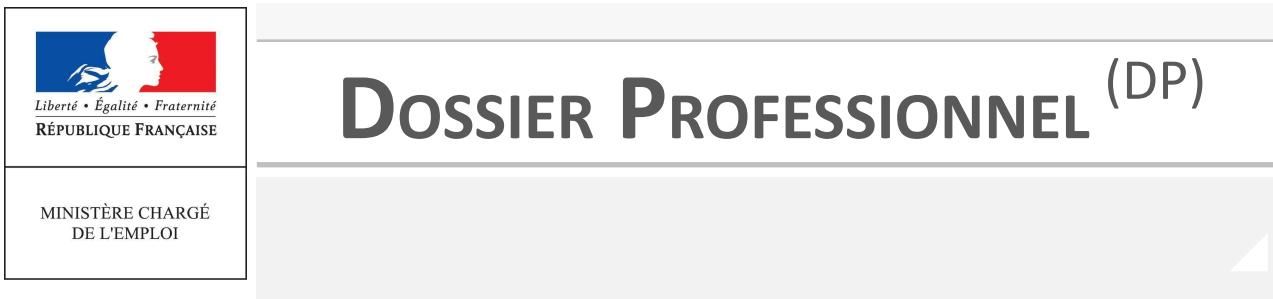
# DOSSIER PROFESSIONNEL (DP)

- Page de recette:

The image shows a smartphone displaying a recipe card. The card includes a logo at the top left, a menu icon at the top right, a title, a thumbnail image, user ratings, preparation time, serving size, a list of ingredients with photos, ingredient names, and quantities. It also features numbered steps, a footer with links like Contact and Mentions légales, and a summary line at the bottom.

- Logo (au clic, ramène à la page d'accueil)
- Ouvre le menu déroulant
- Moyenne des notes des utilisateurs
- Temps de préparation
- Nombre de personnes
- Liste des ingrédients
- Photo de l'ingrédient
- Nom de l'ingrédient
- Quantité nécessaire
- Etapes de préparations, numérotées
- ETAPÉ 1  
Si osculantur puer tuus aut uxorem tuam, osculum, non dico quod omnia quea sunt hominis, et sic non tangetur, si aut ex eis moriatur.
- ETAPÉ 2  
Et omnia in potestate nostra esse natura liber, libera, libere valeant; sed illis non est in nostra potestate sunt
- ETAPÉ 3  
Si osculantur puer tuus aut uxorem tuam, osculum, non dico quod omnia quea sunt hominis, et sic non tangetur, si aut ex eis moriatur.
- ETAPÉ 4  
Si osculantur puer tuus aut uxorem tuam, osculum, non dico quod omnia
- Liens vers la page de contact, mentions légales et formulaire pour proposer une recette
- Contact
- Mentions légales
- Proposer une recette
- Ligne de texte résumant le principe du site, pour le référencement





# DOSSIER PROFESSIONNEL (DP)

**Recette**

Logo (au clic, ramène à la page d'accueil)

Moyenne des notes des utilisateurs

Temps de préparation

Nombre de personnes

Etapes de préparations, numérotées

Liens vers l'accueil et les différentes catégories de recettes

Liste des ingrédients

Photo de l'ingrédient

Nom de l'ingrédient

Quantité nécessaire

Titre de la recette

Ingrediénts

ETAPE 1

ETAPE 2

ETAPE 3

ETAPE 4

Liens vers la page de contact, mentions légales et formulaire pour proposer une recette

Contact    Mentions légales    Proposer une recette

Lorem ipsum dolor sit amet, consectetur adipiscing.

Ligne de texte résumant le principe du site, pour le référencement

**2. Précisez les moyens utilisés :**

J'ai utilisé Whimsical (<https://whimsical.com/>) pour réaliser les wireframes.

**3. Avec qui avez-vous travaillé ?**

J'ai travaillé seule sur ces documents.

**4. Contexte**

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

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : *04/01/2021* au : *02/07/2021*



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 1

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

*CP 2 - Réaliser une interface web statique et adaptable.*

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

Aujourd'hui, lorsque l'on crée une application web, il est indispensable de prendre en compte tous les contextes d'utilisation, que ce soit sur mobile, tablette ou ordinateur, puisque de plus en plus d'utilisateurs consultent ces sites web via leur téléphone mobile.

Ce projet est une intégration d'un site web en HTML et CSS à partir d'un visuel fourni dans l'exercice, qui est donc réalisé en responsive. Voici des captures d'écran de la page réalisée, en 3 formats différents (mobile, tablette et desktop) :



**FINAL FANTASY CRISIS CORE**

**Aerith Gainsborough**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier **74,90 €**



**FINAL FANTASY VII**

**Cloud Strife Remake**

Figurine de 8 pouces - livrée avec son packaging

Ajouter au panier **74,90 €**



**FINAL FANTASY ADVENT CHILDREN**

**Cloud Strife**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier **98,90 €**



**FINAL FANTASY VII**

**Cloud Strife & Fenrir**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier **113,90 €**



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

**oFig** Figurines et statuettes

Figurines sur le thème *Final Fantasy*



FINAL FANTASY CRISIS CORE

**Aerith Gainsborough**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier 74,90 €



FINAL FANTASY VII

**Cloud Strife Remake**

Figurine de 8 pouces - livrée avec son packaging

Ajouter au panier 74,90 € 59 €



FINAL FANTASY ADVENT CHILDREN

**Cloud Strife**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier 98,90 €



FINAL FANTASY VII

**Cloud Strife & Fenrir**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier 113,90 €



FINAL FANTASY ADVENT CHILDREN

**Tifa Lockhart**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier 98,90 € 69 €



FINAL FANTASY VII

**Vincent Valentine**

Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier 74,90 €



FINAL FANTASY VII

**Barret Wallace**

Figurine de 8 pouces - livrée avec son packaging

Ajouter au panier 74,90 €



FINAL FANTASY VII

**Sephiroth**

Figurine de 10 pouces - livrée avec son packaging

Ajouter au panier 113,90 €



FINAL FANTASY ADVENT CHILDREN

**Nanaki - Red XIII**

Figurine de 5 pouces - livrée avec son packaging

Ajouter au panier 98,90 €

**oFig** - toutes les images sont sous copyright SquareEnix



**Aerith Gainsborough**  
FINAL FANTASY CRISIS CORE  
Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier    **74,90 €**



**Cloud Strife Remake**  
FINAL FANTASY VII  
Figurine de 8 pouces - livrée avec son packaging

Ajouter au panier    **74,90 €**    **59 €**



**Cloud Strife**  
FINAL FANTASY ADVENT CHILDREN  
Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier    **98,90 €**



**Cloud Strife & Fenrir**  
FINAL FANTASY VII  
Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier    **113,90 €**



**Tifa Lockhart**  
FINAL FANTASY ADVENT CHILDREN  
Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier    **98,90 €**    **69 €**



**Vincent Valentine**  
FINAL FANTASY VII  
Figurine de 7 pouces - livrée avec son packaging

Ajouter au panier    **74,90 €**



**Barret Wallace**  
FINAL FANTASY VII  
Figurine de 8 pouces - livrée avec son packaging

Ajouter au panier    **74,90 €**



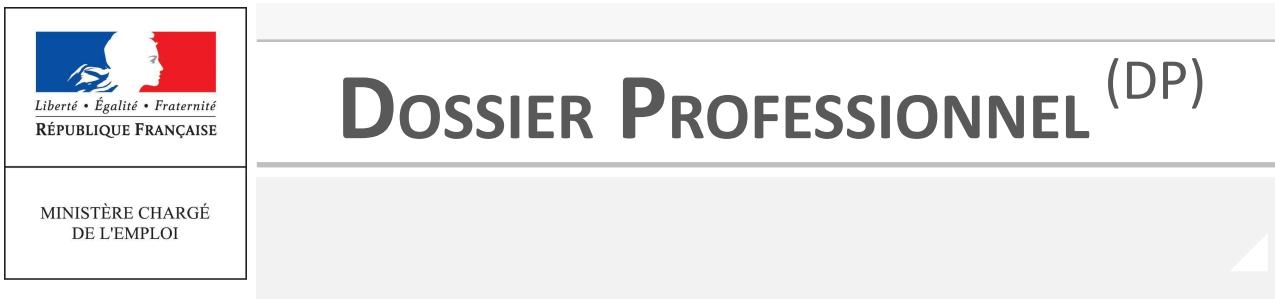
**Sephiroth**  
FINAL FANTASY VII  
Figurine de 10 pouces - livrée avec son packaging

Ajouter au panier    **113,90 €**



**Nanaki - Red XIII**  
FINAL FANTASY ADVENT CHILDREN  
Figurine de 5 pouces - livrée avec son packaging

Ajouter au panier    **98,90 €**



### Extrait du code HTML pour un article :

A screenshot of a web page showing a product listing. The page has a blue header bar with three dots (red, yellow, green). The main content area shows an article with the following HTML code:

```
<article class="product">
  
  <aside>
    Final Fantasy VII
  </aside>
  <h3>
    Cloud Strife Remake
  </h3>
  <p>
    Figurine de 8 pouces - livrée avec son packaging
  </p>
  <div class="add-to-cart">
    <div class="cart">
      Ajouter au panier
    </div>
    <div class="price-before">
      74,90 €
    </div>
    <div class="actual-price">
      59 €
    </div>
  </div>
</article>
```

Un article contient une image, un texte qui indique d'où la figurine vient, le nom de la figurine, une description contenant sa taille, et une div pour l'ajout au panier avec l'ancien prix (facultatif) et le véritable prix.

### Extrait du code CSS pour le responsive des articles :

Le premier **media query** s'applique aux appareils avec un écran avec un affichage d'une largeur inférieure à 1400px et change la largeur des articles (avec la classe `product`) pour les diminuer à 300px.

Le deuxième s'applique aux appareils avec un écran avec un affichage d'une largeur inférieure à 750px et crée plus de modifications, car il diminue la largeur des articles à 200px mais aussi la taille des différents textes, padding et margin et de la div d'ajout au panier, pour avoir des tailles d'éléments et des écarts entre eux plus adaptés et proportionnels à la taille de l'article dans son intégralité.

A screenshot of a CSS file showing media queries for responsive design. The code includes:

```
@media screen and (max-width: 1400px) {
  .product {
    width: 300px;
  }
}

@media screen and (max-width: 750px) {
  .product {
    width: 200px;
  }

  h3 {
    font-size: 15px;
  }

  p {
    font-size: .8em;
  }

  .add-to-cart {
    padding: 10px;
    font-size: .8em;
  }

  .actual-price {
    font-weight: bold;
    font-size: 15px;
  }

  .cart {
    margin-right: 2px;
  }
}
```

**2. Précisez les moyens utilisés :**

HTML, CSS, et afin de rendre le site responsive, j'ai utilisé des media queries.

**3. Avec qui avez-vous travaillé ?**

J'ai travaillé seule sur cette intégration.

**4. Contexte**

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

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : *04/01/2021* au : *02/07/2021*



# 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 :

Le projet réalisé consiste en une to do list en React. Il y a ici la possibilité d'ajouter et de supprimer une tâche et de la marquer comme effectuée.

De plus, un compteur de tâche affiche le nombre de tâches en cours (non cochées).

Enfin, les tâches sont ordonnées: celles qui n'ont pas encore été effectuées se trouvent en haut, et quand on marque une tâche comme effectuée, elle se déplace vers le bas.

Ajouter une tâche		
2 tâches en cours		
<input type="checkbox"/>	Coder une todolist en impératif	
<input type="checkbox"/>	Préparer des crêpes	
<input checked="" type="checkbox"/>	Appeler Jean Marc	

## Ajout d'une tâche :

```
const handleFormSubmit = (event) => {
  // Stores all tasks ids
  const ids = tasks.map((task) => task.id);
  // Stores the maximum id
  const maxId = ids.length > 0 ? Math.max(...ids) : 0;

  event.preventDefault();

  // Creates a new task
  const newTask = {
    // New id, superior to all other ids
    id: maxId + 1,
    // Label from the input value
    label: inputText,
    // Boolean to false because the task is not done yet when created
    done: false,
  };

  // Checks that the input isn't empty
  if (inputText === '' || inputText === ' ') {
    return;
  }

  // Add the task to the list of tasks
  const newTasks = [
    ...tasks,
    newTask,
  ];

  setTasks(newTasks);
};
```

Lorsque l'utilisateur ajoute une tâche, la fonction `handleFormSubmit` est appelée.

Elle vérifie que l'input n'est pas vide, puis crée une nouvelle tâche, avec un nouvel id, le titre de la tâche, et un booléen à faux qui signifie que la tâche n'a pas encore été effectuée.

Elle ajoute enfin le nouvel objet créé au tableau qui représente la liste des tâches.

## Changement d'état d'une tâche:

Quand l'utilisateur coche ou décoche une tâche, la fonction `handleTaskStatus` est appelée.

Celle-ci va modifier la propriété "done" de l'objet qui correspond à la tâche pour lui assigner la valeur inverse.

```
const handleTaskStatusChange = (taskId) => {
  // Looks for the task with the same id as the one that the user just clicked on
  const newTasks = tasks.map((task) => {
    if (task.id === taskId) {
      return {
        ...task,
        // Changes the boolean to the opposite
        done: !task.done,
      };
    }
    return task;
  });

  // Updates the tasks list with the new information
  setTasks(newTasks);
};
```

## Compteur de tâches:

```
const numberOfInProgressTasks = tasks.filter((task) => !task.done).length;
```

Pour compter les tâches en cours, on utilise la méthode `filter()` sur le tableau des tâches, qui retourne un nouveau tableau ne contenant que les tâches qui ont une

propriété "done" égale à faux, puis on les compte, et stocke le résultat dans la variable `numberOfInProgressTasks`, qui pourra ensuite être affichée.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## Tâches ordonnées:

Pour ordonner les tâches, on stocke le tableau des tâches non ordonnées dans une constante `unSortedTasks`, puis on utilise la méthode `sort()` pour trier `unSortedTasks` selon la propriété “done” de ces tâches. On stocke le tableau renvoyé dans la constante `sortedTasks`, qu'on pourra ensuite utiliser pour afficher les tâches dans l'ordre voulu.

```
● ● ●  
const unSortedTasks = [...tasks];  
const sortedTasks = unSortedTasks.sort((a, b) => a.done - b.done);
```

## 2. Précisez les moyens utilisés :

Pour cet exercice, j'ai utilisé React et Sass.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

## 4. Contexte

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

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : *04/01/2021* au : *02/07/2021*

## Activité-type 2

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

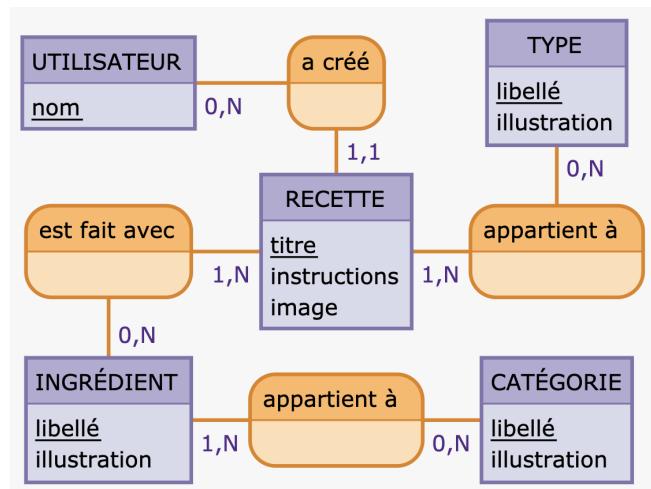
CP 5 - Créer une base de données.

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

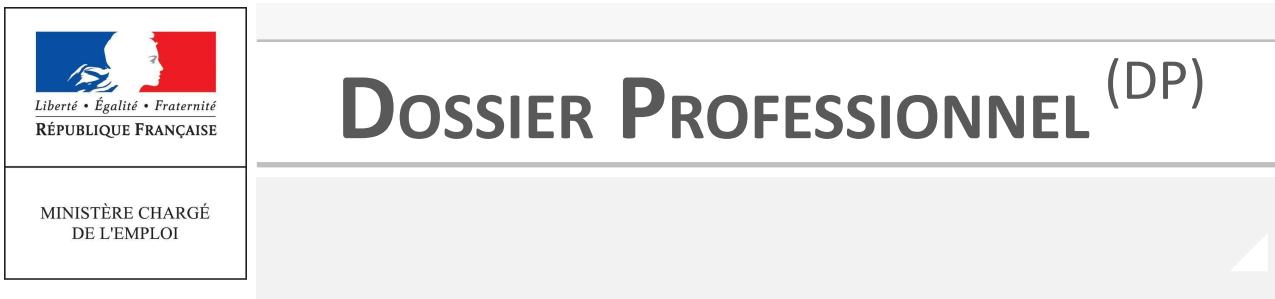
Dans le cadre d'un entraînement à la création d'une API, il fallait créer une base de données en partant de zéro sur le thème de notre choix. J'ai choisi de faire la base de données d'un site de recettes.

J'ai d'abord créé le **MCD** (Modèle Conceptuel de Données), qui représente la structure des données et ses relations. J'ai listé 5 entités différentes: recette, type (de recette), ingrédient, catégorie (pour les ingrédients), et utilisateur.

- Une recette a été créée par un seul utilisateur, et un utilisateur a pu créer au minimum 0 recettes et au maximum plusieurs.
- Une recette peut appartenir à un ou plusieurs types, et un type peut contenir de 0 à plusieurs recettes.
- Une recette est faite avec au moins 1 ingrédient ou plusieurs, et un ingrédient peut faire partie de 0 à plusieurs recettes.
- Un ingrédient peut appartenir à une ou plusieurs catégories, et une catégorie peut contenir de 0 à plusieurs ingrédients.



Ensuite, j'ai utilisé **Sqitch**, qui est un système de migrations permettant de versionner une base de données, pour créer la base de données, via le **MPD** (Modèle physique de Données), qui consiste à planter une base de données dans un SGBD (Système de Gestion de Base de Données), ici **PostgreSQL**:



Pour convertir le MCD en MPD, j'ai suivi ces 3 règles:

```

BEGIN;

CREATE DOMAIN url AS text
CHECK (VALUE ~ 'https?://\/(www\.)?[-a-zA-Z0-9@%._+~#=]{2,256}\
\.[a-z]{2,6}\b([-a-zA-Z0-9@%._+~#()?&/=]*')';
COMMENT ON DOMAIN url IS 'match URLs (http or https)';

CREATE TABLE "author" (
    "id" integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    "name" text NOT NULL UNIQUE,
    "created_at" timestamptz NOT NULL DEFAULT now(),
    "updated_at" timestamptz,
    "deleted_at" timestamptz
);

CREATE TABLE "recipe" (
    "id" integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    "title" text NOT NULL,
    "instructions" text NOT NULL,
    "image" url,
    "author_id" integer NOT NULL REFERENCES "author"("id"),
    "created_at" timestamptz NOT NULL DEFAULT now(),
    "updated_at" timestamptz,
    "deleted_at" timestamptz
);

CREATE TABLE "ingredient" (
    "id" integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    "label" text NOT NULL,
    "illustration" url,
    "created_at" timestamptz NOT NULL DEFAULT now(),
    "updated_at" timestamptz,
    "deleted_at" timestamptz
);

CREATE TABLE "type" (
    "id" integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    "label" text NOT NULL UNIQUE,
    "illustration" url,
    "created_at" timestamptz NOT NULL DEFAULT now(),
    "updated_at" timestamptz,
    "deleted_at" timestamptz
);

CREATE TABLE "category" (
    "id" integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    "label" text NOT NULL UNIQUE,
    "illustration" url,
    "created_at" timestamptz NOT NULL DEFAULT now(),
    "updated_at" timestamptz,
    "deleted_at" timestamptz
);

CREATE TABLE "recipe_type" (
    "recipe_id" integer NOT NULL REFERENCES "recipe"("id"),
    "type_id" integer NOT NULL REFERENCES "type"("id"),
    "created_at" timestamptz NOT NULL DEFAULT now(),
    PRIMARY KEY ("recipe_id", "type_id")
);

CREATE TABLE "recipe_ingredient" (
    "recipe_id" integer NOT NULL REFERENCES "recipe"("id"),
    "ingredient_id" integer NOT NULL REFERENCES "ingredient"("id"),
    "created_at" timestamptz NOT NULL DEFAULT now(),
    PRIMARY KEY ("recipe_id", "ingredient_id")
);

CREATE TABLE "ingredient_category" (
    "ingredient_id" integer NOT NULL REFERENCES "ingredient"("id"),
    "category_id" integer NOT NULL REFERENCES "category"("id"),
    "created_at" timestamptz NOT NULL DEFAULT now(),
    PRIMARY KEY ("ingredient_id", "category_id")
);

COMMIT;

```

- Toute entité du MCD devient une table.

- Si une des cardinalités maximum vaut 1, une clé étrangère est créée du côté de l'entité où se trouve le 1: ici par exemple une clé étrangère dans la table recipe fait référence à l'identifiant de la table author (*author\_id*).

- Si les deux cardinalités maximum sont n, donc une relation "plusieurs à plusieurs", la relation devient une table à part entière: ici les tables *recipe\_type*, *recipe\_ingredient* et *ingredient\_category*.

J'ai ajouté les colonnes *created\_at*, *updated\_at* et *deleted\_at* sur la plupart des tables. Le *deleted\_at* sert à faire un soft delete (suppression réversible). Sur les tables qui représentent les relations plusieurs à plusieurs, les colonnes *updated\_at* et *deleted\_at* n'existent pas: la seule modification possible est la suppression de la relation entre les 2 tables, et donc de cette table entière.

J'ai également créé un domaine url grâce à une expression régulière pour vérifier que les url des images et illustrations dans les différentes tables (*recipe*, *ingredient*, *type*, *category*) sont des url valides.

**2. Précisez les moyens utilisés :**

Pour le MCD j'ai utilisé Mocodo (<http://mocodo.wingi.net/>). Pour créer la base de données, j'ai utilisé le SGBD PostgreSQL et le système de migrations permettant de versionner une base de données Sqitch.

**3. Avec qui avez-vous travaillé ?**

J'ai travaillé seule sur ce projet.

**4. Contexte**

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

Chantier, atelier, service ▶ *Exercice de formation*

Période d'exercice ▶ Du : *04/01/2021* au : *02/07/2021*



# 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 6** - Développer les composants d'accès aux données.

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

Pour ce projet durant la formation, j'ai dû créer un site web affichant les différentes promotions d'une école ainsi que les étudiants qui les constituent. On devait également pouvoir ajouter un étudiant à une promotion.

Pour ce faire, je me connecte d'abord à la base de données:

```
const {Client} = require('pg');

const client = new Client(process.env.PG_URL);

client.connect();

module.exports = client;
```

On récupère le module pg.

Puis on crée la connexion SQL, en définissant les paramètres (présents dans le fichier .env).

Puis on ouvre la connexion.

Et on exporte enfin le module client pour pouvoir l'utiliser dans le dataMapper.

Dans le dataMapper, on récupère le module client ainsi:

```
const client = require('./database');
```

On crée ensuite dans ce dataMapper les requêtes SQL vers la base de données.

Par exemple:



```
getStudentsInPromo: (promoId, callback) => {
  const selectPromoStudentsQuery = `SELECT * FROM "student" WHERE "promo_id"=$1;`

  client.query(selectPromoStudentsQuery, [promoId], callback);
},
getPromoById: (promoId, callback) => {
  const selectPromoById = `SELECT * FROM "promo" WHERE "id"=$1;`

  client.query(selectPromoById, [promoId], callback);
},
addStudent: (studentInfo, callback) => {
  const addStudentQuery = `INSERT INTO "student"
("first_name", "last_name", "github_username", "promo_id")
VALUES
($1, $2, $3, $4);`

  client.query(addStudentQuery,
  [
    studentInfo.first_name,
    studentInfo.last_name,
    studentInfo.github_username,
    studentInfo.promo
  ],
  callback);
},
```

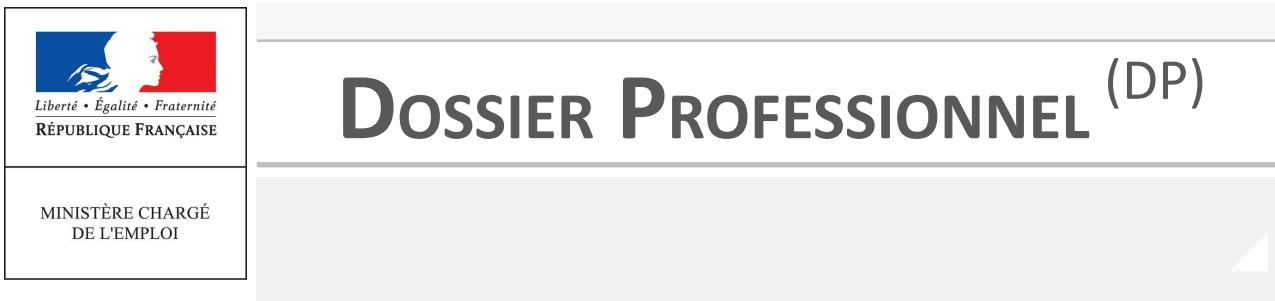
La méthode *getStudentsInPromos* permet de récupérer la liste de tous les étudiants d'une promo par son id.

La méthode *getPromoById* permet de récupérer les informations d'une promo par son id.

La méthode *addStudent* permet d'ajouter un étudiant avec ses infos (nom, prénom, pseudo github) à une promo (par son id).

Pour chaque méthode, on stocke la requête SQL. Ensuite, on donne en paramètre à `client.query` la requête et le callback (qui viendra du contrôleur).

Dans les contrôleurs, on va ensuite utiliser ces méthodes du dataMapper.



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

Pour récupérer tous les étudiants d'une promo:

```
showPromoStudents: (req, res) => {
    // Stores the id of the group of students
    const promoId = req.params.promoId;

    // To get all the students in this group of students
    dataMapper.getStudentsInPromo(promoId, (error, result) => {

        if (error) {
            console.log('error: ', error);
        } else {
            // Stores the result (the students)
            const studentsInPromoFromDatabase = result.rows;

            // To get all the information on this group of students
            dataMapper.getPromoById(promoId, (error, result) => {
                if (error) {
                    console.log('error: ', error);
                } else {
                    // Stores the result (the group of students)
                    const promoFromDatabase = result.rows[0];

                    // Renders the view with all the information
                    res.render('promo_students', {
                        students: studentsInPromoFromDatabase,
                        promoName: promoFromDatabase.name,
                        newStudentLastName: req.query.newStudentLastName,
                        newStudentFirstName: req.query.newStudentFirstName,
                    });
                }
            });
        }
    });
},
```

On récupère l'id de la promo par le paramètre d'url promoid.

On appelle la méthode `getStudentsInPromo` du dataMapper pour récupérer tous les étudiants de la promo.

S'il n'y a pas d'erreur, on appelle la méthode `getPromoById` pour récupérer les informations de la promo.

S'il n'y a pas d'erreur, on envoie la vue avec les informations nécessaires.

Pour ajouter un étudiant à une promo:

```
addStudent: (req, res) => {
    // To add a student to a group of students
    dataMapper.addStudent(req.body, (error) => {
        if (error) {
            console.log('error :', error);
        } else {
            // Redirects to the page of the group of students
            // with a message to notify that a new student was added
            res.redirect(`'/promo/${req.body.promo}/students?
newStudentFirstName=${req.body.first_name}&
newStudentLastName=${req.body.last_name}`);
        }
    });
},
```

On appelle la méthode `addStudent` du dataMapper pour ajouter l'étudiant. S'il n'y a pas d'erreur, on redirige vers la page de la promo, avec en paramètre de requête le nom et le prénom de l'étudiant ajouté, pour pouvoir l'afficher.

Enfin, les vues, faites avec EJS (Embedded Javascript Templates)

peuvent afficher les informations de la base de données. Par exemple, pour la page qui liste tous les étudiants d'une promo:



```
<%- include('partials/header.ejs') %>

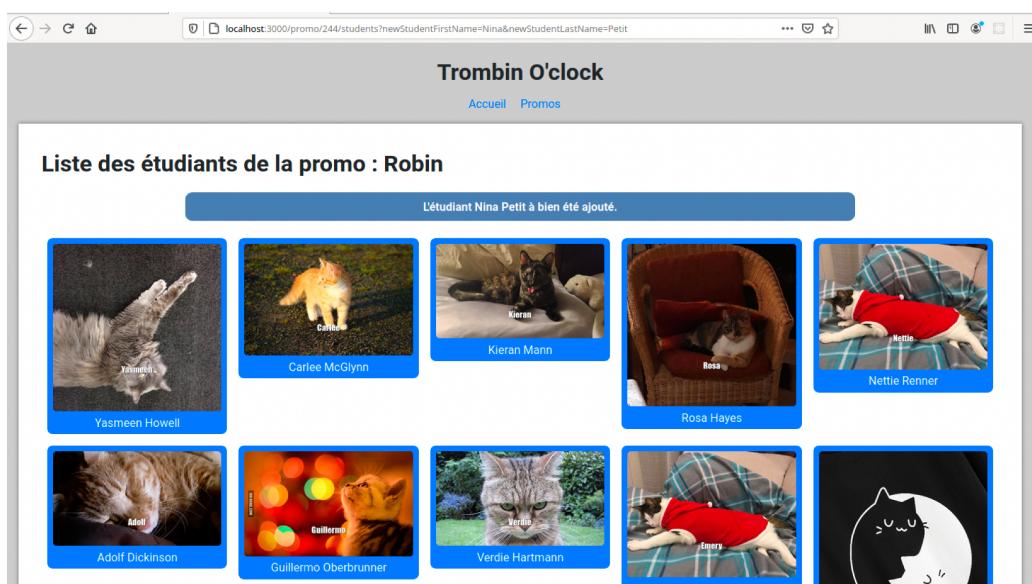
<h1>Liste des étudiants de la promo : <%= promoName %></h1>

<% if (newStudentFirstName && newStudentLastName) { %>
  <p class="student-added-msg">L'étudiant <%= newStudentFirstName %> <%= newStudentLastName %> à
  bien été ajouté.</p>
<% } %>

<ul class="cards">
  <% students.forEach((student) => { %>
    <li class="card">
      <a>
        <figure>
          
          <figcaption><%= student.first_name %> <%= student.last_name %></figcaption>
        </figure>
      </a>
    </li>
  <% }) %>
</ul>

<%- include('partials/footer.ejs') %>
```

Cette vue affiche une carte pour chaque étudiant, avec ses informations, et si un étudiant vient d'être ajouté, il y a un message pour prévenir l'utilisateur que l'ajout a bien été pris en compte.





MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## 2. Précisez les moyens utilisés :

Pour ce projet, j'ai utilisé Node.js, PostgreSQL, EJS et CSS.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

## 4. Contexte

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

Chantier, atelier, service ➤ *Exercice de formation*

Période d'exercice ➤ Du : 04/01/2021 au : 02/07/2021

## 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 ce projet, il fallait créer l'API d'un blog. Il fallait donc créer la base de données, puis des routes pour récupérer la liste des articles, la liste des catégories, un seul article par son id, les articles d'une catégorie par l'id de la catégorie, et il fallait également pouvoir ajouter un article.

Après s'être connecté à la base de données dans un fichier **client.js**, on peut créer les **dataMappers**, qui permettent de faire les requêtes vers la base de données. Par exemple, je crée **categoryDataMapper** pour récupérer les informations nécessaires par rapport aux catégories. (On créera ainsi un **postDataMapper** pour les requêtes qui concernent les articles):

```
● ● ●

const client = require('./client');

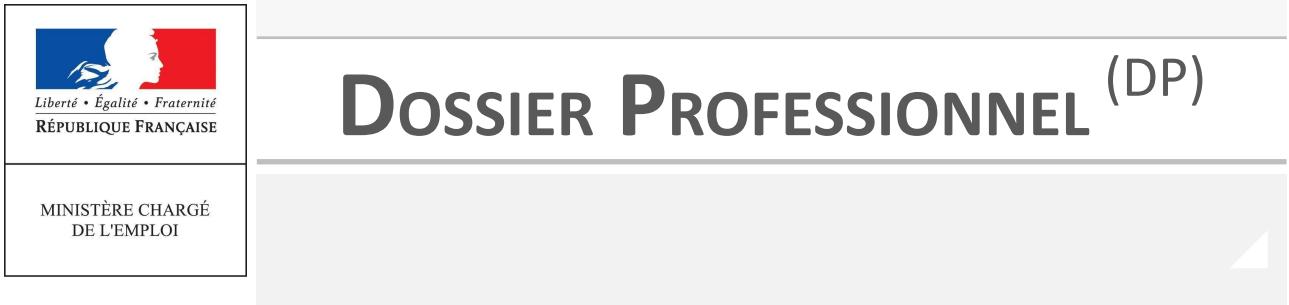
module.exports = {

    getAllCategories: async() => {
        const result = await client.query(`SELECT * FROM "category"`);
        return result.rows;
    },

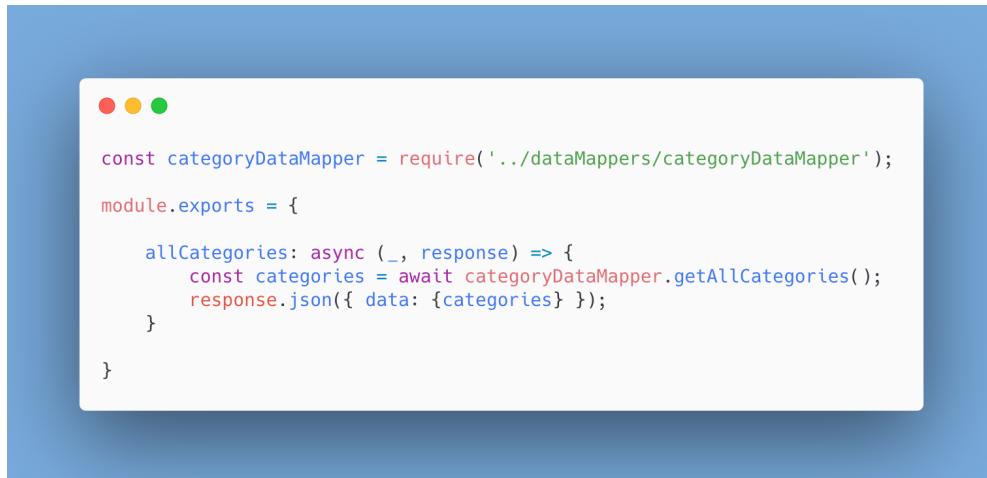
    getCategoryById: async(id) => {
        const result = await client.query(`SELECT * FROM "category" WHERE id = $1`, [id]);
        return result.rows[0];
    }
}
```

Je récupère ici le client, puis je l'utilise pour créer et exporter les méthodes qui me permettront de faire les requêtes voulues.

Je peux ensuite utiliser ces méthodes dans les **contrôleurs** (on crée de même un contrôleur par "thème", ici ce sera donc categoryController et postController).



categoryController:



```
● ● ●

const categoryDataMapper = require('../dataMappers/categoryDataMapper');

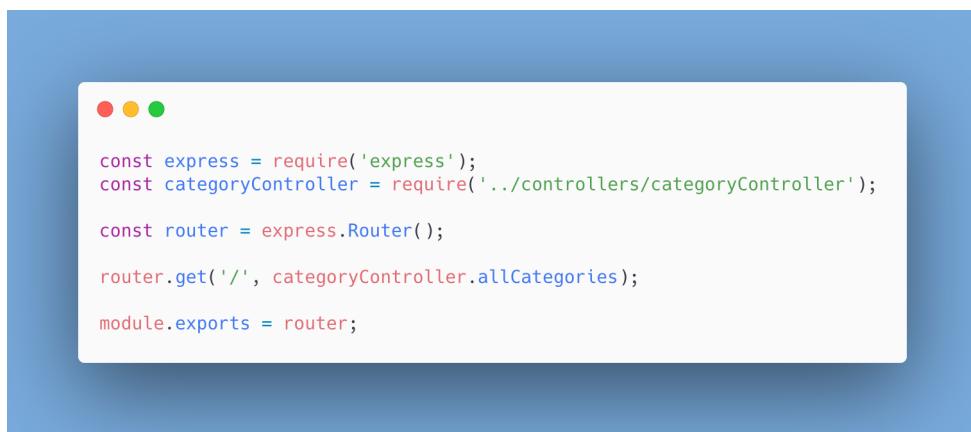
module.exports = {

    allCategories: async (_, response) => {
        const categories = await categoryDataMapper.getAllCategories();
        response.json({ data: {categories} });
    }
}
```

Je récupère le dataMapper, puis je l'utilise pour créer et exporter des méthodes qui me permettront de récupérer les informations nécessaires, en les envoyant dans un json.

Je peux ensuite utiliser ces méthodes dans les **routeurs** (ici un categoryRouter et un postRouter).

categoryRouter:



```
● ● ●

const express = require('express');
const categoryController = require('../controllers/categoryController');

const router = express.Router();

router.get('/', categoryController.allCategories);

module.exports = router;
```

Grâce à **Express.js**, je peux mettre en place un routeur, qui permettra d'utiliser la bonne méthode selon la route.

Enfin, dans un fichier **index.js**:

```
if (process.env.NODE_ENV !== 'production') {
    require('dotenv').config();
}

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

const categoryRouter = require('./app/routers/categoryRouter');
const postRouter = require('./app/routers/postRouter');

const app = express();
app.use(cors());
app.use(express.json());

app.use('/categories', categoryRouter);
app.use('/posts', postRouter);

const port = process.env.PORT || 3000;

app.listen(port, _ => {
    console.log(`http://localhost:${port}`);
});
```

On récupère **express** pour déterminer ensuite le format de récupération des données (ici json).

On utilise **cors** pour ajouter l'autorisation du cross-origin, pour que le service soit accessible au plus grand nombre.

On récupère les routeurs, et on définit un routeur avec une route racine différente de '/'.

Puis on définit le port.

## Sécurité du projet:

Injections SQL: Une injection SQL se produit lorsqu'une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Afin d'empêcher ces attaques, il faut préparer les requêtes SQL.

Par exemple:

```
insertPost: async post => {
    const result = await client.query(`INSERT INTO "post" (title, slug, excerpt, content, category_id)
        VALUES($1, $2, $3, $4, $5) RETURNING *`, [post.title, post.slug, post.excerpt, post.content,
    post.categoryId]);
    return result.rows[0];
}
```



# DOSSIER PROFESSIONNEL (DP)

MINISTÈRE CHARGÉ  
DE L'EMPLOI

Joi: J'utilise Joi pour vérifier les données. On décrit ce à quoi les données devraient ressembler. Par exemple, pour décrire ce à quoi un article devrait ressembler:

```
● ● ●

const Joi = require('joi');

const schema = Joi.object({
  title: Joi.string().required().min(2),
  slug: Joi.string().required().pattern(/^([0-9a-z]+)(-[|$])+/),
  excerpt: Joi.string().min(20),
  content: Joi.string().required().min(40),
  categoryId: Joi.number().integer().required().min(1)
}).required();

module.exports = schema;
```

Ainsi, on peut ensuite utiliser ce schéma pour vérifier les données lors de l'ajout d'un article. On crée un fichier validate.js, avec une méthode *body*:

```
● ● ●

body: (schema) => {

  return async (request, response, next) => {

    try {
      await schema.validateAsync(request.body);
      next();
    } catch (error) {
      return response.status(400).json({ error: error.details[0].message });
    }
  },
};
```

Et on peut ensuite l'utiliser dans le routeur:

```
router.route('/')
  .get(postController.allPosts)
  .post(validate.body(postSchema), postController.addPost);
```

## 2. Précisez les moyens utilisés :

Pour ce projet, j'ai utilisé Node.js, Express, Joi, PostgreSQL.

## 3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet

## 4. Contexte

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

Chantier, atelier, service ▶ *Exercice de formation*

Période d'exercice ▶ Du : *04/01/2021* au : *02/07/2021*



MINISTÈRE CHARGÉ  
DE L'EMPLOI

# DOSSIER PROFESSIONNEL (DP)

## Titres, diplômes, CQP, attestations de formation

(*facultatif*)

Intitulé	Autorité ou organisme	Date
Licence en école de commerce	SKEMA	2020
Maîtrise de la qualité en projet Web (Avancé)	Opquast	05/08/2021