



DOSSIER PROFESSIONNEL (DP)

Nom de naissance

► DANG

Nom d'usage

► CABROL

Prénom

► Sandrine

Adresse

► 10 allée des frênes – 31670 LABEGE

Titre professionnel visé

Concepteur développeur d'applications

MODALITE D'ACCES :

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

DOSSIER PROFESSIONNEL (DP)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du Dossier Professionnel (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'emploi]

Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

 <http://travail-emploi.gouv.fr/titres-professionnels>

DOSSIER PROFESSIONNEL (DP)

Sommaire

Exemples de pratique professionnelle

Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

p. 5

- ▶ Conception et développement d'un composant Pollution d'une application avec Angular ... p. 5

Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

p. 8

- ▶ Création d'une base de données InfoPlus (entreprise fictive) p. 8

Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

p. 12

- ▶ Conception et développement d'une application gérant un panier..... p. 12

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

p. 16

Déclaration sur l'honneur

p. 17

Documents illustrant la pratique professionnelle (facultatif)

p. 18

Annexes (Si le RC le prévoit)

p. 19

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exemple n°1 ▶ Conception et développement d'un composant pour une application avec Angular

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

Conception et réalisation d'un composant pour une application web avec Angular

1. Nous avons réalisé la maquette illustrant le composant à développer

cf : Annexe 1 – page 19 : Exemple de la maquette réalisée VS. Déploiement du composant Pollution – projet Angular

2. J'ai analysé le fichier Json (API public) dans l'objectif de savoir ce que nous pourrions exploiter comme informations mais aussi comment l'exploiter (compréhension de la construction du fichier Json)

3. Ensuite nous avons développé le **composant** Pollution. Pour ce composant principal, nous sommes partis sur la création de deux composants :

- **Composant PARENT** qui permet l'affichage d'une map avec leaflet et d'un tableau avec les données sur la composition de l'air relatif à une ville saisie par l'utilisateur.
- **Composant ENFANT** représentant une barre de recherche afin que l'utilisateur puisse taper le nom d'une ville. Le composant enfant 'barre de recherche' transmet la ville au composant parent pour que ce dernier puisse récupérer les datas correspondantes et nécessaires.

Pour cela, j'ai créé une **couche Service** qui récupère l'api et met à disposition les datas grâce à la méthode GET de HttpClient et en retournant un observable.

```
33 | url: string = 'https://public.opendatasoft.com/api/records/1.0/search/?dataset=worldwide-pollution&q=&facet=c
34 |
35 | constructor(private http: HttpClient) { }
36 |
37 | private getNomVille(ville: string = 'Balma'): Observable<any> {
38 |   return this.http.get(`${this.url}&refine.filename=${ville}`);
39 | }
```

Les observables permettent ainsi au composant Parent (mapLeaflet) de s'y souscrire et accéder par exemple à une valeur telle que le nom d'un polluant.

Ci-dessous, le code me permettant de récupérer des données dans l'api et de 'setter', l'objet Polluant déclaré en attribut de classe avec ses datas. Mais avant cela, je vérifie que le user a bien saisi une ville avec la condition 'IF'.

DOSSIER PROFESSIONNEL (DP)

```
127 // appel de la méthode pour récupérer les data du fichier Json
128 this.serviceRechercheVille.getDataVilleSaisie(villeOK).subscribe(
129
130   data => {
131
132     if (data.records.length === 0) {
133       this.error = 'Ville incorrecte';
134
135     } else {
136       // Set de l'objet Json avec les données récupérées
137       this.serviceRechercheVille.setPolluantsVille(
138         {
139           'value_pm25': data.records[0].fields.value_pm5,
140           'category_pm25': data.records[0].fields.category_pm25,
141           'description_pm25': data.records[0].fields.data_pollutants_pm25_descr
```

Afin que le composant puisse souscrire à l'observable créé dans le service, j'ai injecté ce service dans le constructeur du composant TS.

Enfin lorsque toutes les données ont été récupérées dans des variables, j'ai réalisé le **template HTML** pour afficher la carte et les données de la qualité de l'air dans un tableau HTML et un encart pop up sur la carte.

Ci-dessous, une partie du code permettant d'afficher le tableau avec la description de chaque polluant si la ville saisie est correcte et si error est vide (2 directives structurelles : *ngIf)

```
9   <div *ngIf="error == ''; else erreur">
10     <!-- Directive structurelle : if ville est correcte, affichage du tableau --
11     <table *ngIf=villeSaisie>
12       <thead>
13         <tr>
14           <th> POLLUANTS </th>
15           <th> PM25 </th>
16           <th> O3 </th>
17           <th> CO </th>
18           <th> NO2 </th>
19       </tr>
20     </thead>
21
22     <tbody>
23       <tr>
24         <td id="iddescription"> Description </td>
25         <td> {{ polluantsVille.description_pm25 }} </td>
26         <td> {{ polluantsVille.description_O3 }} </td>
27         <td> {{ polluantsVille.description_CO }} </td>
28         <td> {{ polluantsVille.description_NO2 }} </td>
29     </tbody>
```

DOSSIER PROFESSIONNEL (DP)

4. J'ai ensuite **intégré du CSS** à notre composant
5. Une fois le composant fonctionnel, nous l'avons intégré à l'application web
6. Et en dernier lieu, j'ai **fixé les bugs d'intégration** (imports manquants pour la carte leaflet, mise au propre du CSS par rapport à la charte graphique du site, suppression du scroll automatique sur la carte...)

Projet sur 2 semaines avec chaque matin un daily meeting.

2. Précisez les moyens utilisés :

IDE : Visual Studio Code pour utiliser le framework Angular (nodeJs), extension LiveShare pour l'intégration du composant développé dans l'application

Langages travaillés : HTML, CSS, Sass, TypeScript

Gestion de projet : Trello

3. Avec qui avez-vous travaillé ?

Composant Pollution : Farhan Barreh (CDA7), Emmanuel Ducournau (CDA7) et Cyril Sonam (DWWM7)
Application web : groupe CDA7 et groupe DWWM7

4. Contexte

Nom de l'entreprise, organisme ou association ► AFPA

Chantier, atelier, service ► Projet interne (Pollution de l'air)

Période d'exercice ► Du : 02/08/2021 au : 13/08/2021

5. Informations complémentaires (facultatif)

Projet permettant de travailler de la conception du projet jusqu'à son intégration.
Travail en équipe en mode agile

DOSSIER PROFESSIONNEL (DP)

Activité-type 2 Concevoir et développer la persistance des données en intégrant les recommandations de sécurité

Exemple n° 1 ▶ *Création d'une base de données InfoPlus (entreprise fictive)*

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

Conception et développement d'une base de données InfoPlus.

1. J'ai analysé toutes les informations données dans le cahier des charges dans le but de lister toutes les données dans un **dictionnaire de données**.
2. J'ai par la suite recensé les relations grâce aux règles de gestions en prenant en compte les **cardinalités minimum et maximum**.
3. Cela m'a permis d'élaborer graphiquement le **Modèle Conceptuel des Données (MCD)** en contrôlant les règles de normalisation (nom entité, association et attribut doivent être uniques, une entité doit avoir un identifiant, les cardinalités (0,1 ou n), pas d'attribut qui se calcule et pas d'association fantôme (cardinalité max 1 de chaque côté)
4. Une fois le MCD réalisé, je l'ai converti en MLD (**Modèle Logique des Données**)

cf : *Annexe 2 – page 20 : Exemple du MCD et MLD créés pour l'étude de cas InfoPlus*

5. J'ai ensuite élaboré le **Modèle Physique des Données (MPD)** c'est-à-dire implémenter le modèle dans le SGBD (PostgreSQL).

Pour cela, j'ai créé le **script créant les tables**.

Ici par exemple création de deux tables :

- **Personne** ayant 6 colonnes typées (serial pour l'idPersonne qui sera la clé primaire, varchar et char) et ajout de quelques contraintes comme not null ou le nombre maximum de char/varchar
- **Salarie** qui a 5 colonnes mais qui hérite aussi des attributs de Personne grâce au mot clé INHERITS(Personne)

```
1 -- Suppression des tables si elles existent
2 DROP TABLE IF EXISTS PERSONNE CASCADE;
3 DROP TABLE IF EXISTS SALARIE CASCADE;
4
5 CREATE TABLE PERSONNE (
6     idPersonne SERIAL NOT NULL,
7     nomPersonne VARCHAR(30) NOT NULL,
8     prenomPersonne VARCHAR (30) NOT NULL,
9     telPersonne CHAR(10) NOT NULL,
10    mailPersonne VARCHAR NOT NULL,
11    fonctionPersonne VARCHAR(30) NOT NULL
12 );
13
14 CREATE TABLE SALARIE (
15     idSalarie_dirige INT,
16     idDivision INT,
17     numMatricule INT NOT NULL,
18     trigramme CHAR (3) NOT NULL,
19     salaire INT NOT NULL
20 )
21 INHERITS (PERSONNE);
22 |
```

DOSSIER PROFESSIONNEL (DP)

Puis j'ai modifié les tables pour intégrer les clés primaires, clés étrangères et autres contraintes un peu plus « complexes » telles que des check ou des regex.

```
1 -- Création des clés primaires
2 ALTER TABLE SALARIE ADD PRIMARY KEY (idPersonne);
3
4 -- Création des clés étrangères
5 ALTER TABLE SALARIE
6 ADD CONSTRAINT fk_salarie_salarie FOREIGN KEY (idSalarie_dirige) REFERENCES SALARIE (idPersonne)
7 ON DELETE CASCADE;
8
9 -- Contraintes sur le trigramme qui doit contenir que des lettres dans SALARIE
10 ALTER TABLE SALARIE ADD CONSTRAINT chk_salarie_trigramme CHECK (trigramme~* '^[a-z]{3}');
```

Ici par exemple, je dis que idPersonne devient la clé primaire de la table Salarie.

La table Salarie ayant une relation réflexive, la clé étrangère idSalarie_dirige pointe sur la clé primaire idPersonne de la table Salarie.

6. Puis j'ai réalisé le script qui permet d'insérer des données dans les tables créées, ici dans la table Salarie.

```
1 INSERT INTO SALARIE (nomPersonne, prenomPersonne, telPersonne, mailPersonne, fonctionPersonne, idSalarie_dirige, idDivision, numMatricule, salaire) VALUES
2 ('Hunt', 'Owen', '0637383940', 'o.hunt@infoplus.com', 'PDG', NULL, NULL, 10000, 5500),
3 ('Grey', 'Meredith', '0601020304', 'm.grey@infoplus.com', 'Chef de projet', 6, 1, 11111, 3500);
```

7. Une fois la base de données créée, j'ai testé la BDD en extrayant des informations sous forme de requêtes, de triggers et de fonctions et procédures stockées.

Dans la requête ci-dessous, j'ai réalisé une natural join à la table Salarie dans le but de récupérer le nom de la personne. Idem pour récupérer le nom de la tâche dans la table Tache. Puis j'ai trié par nomPersonne.

```
1 -- REQ01 : requête permettant de connaître
2 -- les tâches par projet affectées à chaque salarié.
3
4 SELECT nomPersonne, nomTache, idProjet FROM SALARIE_TACHE_PERIODE
5 NATURAL JOIN SALARIE
6 NATURAL JOIN TACHE
7 ORDER BY nomPersonne;
```

DOSSIER PROFESSIONNEL (DP)

Dans l'exemple ci-dessous, j'ai créé un **trigger** qui se déclenchera avant l'insertion d'un salarié. Ce trigger appelle une **fonction stockée** qui génère le trigramme du salarié à partir de la 1^{ère} lettre de son prénom + la 1^{ère} lettre de son nom + la 2^{nde} lettre de son nom si trigramme n'existe pas, sinon on prend la lettre suivante.

```
1 -- TRG03 : trigger permettant de calculer et d'insérer le trigramme d'un salarié. Ce
2 -- trigger utilise la fonction FCT02.
3
4 -- FCT02 : fonction qui, pour un salarié donné, génère un trigramme à partir de son
5 -- prénom et de son nom de famille.
6
7 DROP TRIGGER IF EXISTS TR_VERIF_TRIGRAMME ON SALARIE CASCADE;
8 -- création du trigger --
9 CREATE TRIGGER TR_VERIF_TRIGRAMME BEFORE INSERT ON SALARIE
10 FOR EACH ROW
11 EXECUTE FUNCTION generer_trigramme ();
12
13
14 -- creation de la fonction appelée par le trigger --
15 CREATE OR REPLACE FUNCTION generer_trigramme ()
16 RETURNS trigger
17 AS $$
18 BEGIN
19     FOR i IN 1..(SELECT LENGTH (NEW.nomPersonne)) LOOP
20         NEW.trigramme = UPPER (SUBSTRING(NEW.prenomPersonne, 1,1)|||
21             SUBSTRING(NEW.nomPersonne, 1, 1)|||
22             SUBSTRING(NEW.nomPersonne, i+1, 1));
23         EXIT WHEN NEW.trigramme NOT IN (SELECT trigramme FROM SALARIE);
24     END LOOP;
25     RETURN NEW;
26 END;
27 $$ LANGUAGE plpgsql;
28
29 -- test et vérification du trigger
30 INSERT INTO SALARIE (nomPersonne, prenomPersonne, telPersonne, mailPersonne, fonctionPersonne, idSalarie_dirige, idDivision, numMatricule, salaire) VALUES
31 ('Grey', 'Lexie', '0550515253', 'l.grey@infoplus.com', 'Designer', 7, 3, 88888, 2300),
32 ('Grayson', 'Liv', '0654555657', 'l.grayson@infoplus.com', 'Chef de projet', 6, 1, 99999, 3100);
```

2. Précisez les moyens utilisés :

Conception de la BDD :

- Dictionnaires des données sous Excel
- MCD et MLD sous OpenModelSphere, AnalyseSI, StarUML

Mise en place de la BDD :

- PostgresSQL avec pgAdmin 4

Développement des composants :

- langage PLPGSQL

3. Avec qui avez-vous travaillé ?

Essentiellement en quasi autonomie et entraide avec stagiaires du groupe

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► AFPA

Chantier, atelier, service ► Projet interne (InfoPlus : entreprise fictive)

Période d'exercice ► Du : 18/02/2021 au : 19/03/2021

5. Informations complémentaires (*facultatif*)

Etude de cas où j'ai pu réaliser de A à Z la conception et le développement d'une base de données.

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité

Exemple n° 1 ▶ Conception et développement d'une application gérant un panier

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

En équipe, nous avons décidé de construire l'application en plusieurs couches :

- repository
- service
- controller
- et un package model contenant nos différents Beans, à savoir (user, article et panier) étroitement liés aux tables de la BDD.

Concernant la **couche repository** permettant d'accéder à la BDD, grâce à Spring et plus particulièrement Spring Data, les interfaces du package repository ont hérité de la classe JpaRepository. Nous avons aussi créé des méthodes spécifiques pour accéder aux données lorsque ces dernières n'étaient pas proposées dans la classe mère.



Par exemple : la méthode ci-dessous, permet de récupérer un objet Panier en fonction de l'idUser et l'idArticle passés en paramètre

```
34 // Méthode qui retourne une ligne de panier en fonction de l'user et de l'article ( utilisé pour l'update)
35 /**
36 * Permet de recuperer un panier par l'iduser et idArticle
37 * @param iduser {@link Integer}
38 * @param idArticle {@link Integer}
39 * @return panier {@link Panier}
40 */
41 @Query("select p from Panier p where p.User.idUser= :idUser and p.Article.idArticle= :idArticle")
42 Panier getByUserAndArticle(@Param("idUser")Integer iduser, @Param("idArticle")Integer idArticle);
```

La **couche service** qui appelle la couche repository via une injection de dépendance grâce à l'annotation @Autowired permet d'encapsuler la logique métier.

Par exemple dans PanierService, on va calculer le prix d'une ligne de panier pour un liste de panier donnée ou calculer le prix total d'un panier ou encore de vérifier l'existence d'une ligne de panier afin de soit augmenter sa quantité si on ajoute le même article, soit d'ajouter la ligne de panier dans le panier si l'article n'existe pas encore dans le panier.

La couche Service permet aussi d'appeler toutes les méthodes déclarées dans la couche Repository par

DOSSIER PROFESSIONNEL (DP)

exemple pour récupérer la liste de tous les articles, j'appelle le findAll().

La **couche RestController** quand à elle, me permet de créer des endpoints afin que mon IHM puisse appeler les APIs créées. En fonction du type de requêtes http, je vais utiliser Get/Post/Delete Mapping. GetMapping pour récupérer des données (les lire), par exemple pour récupérer un panier sauvegardé dans la BDD.

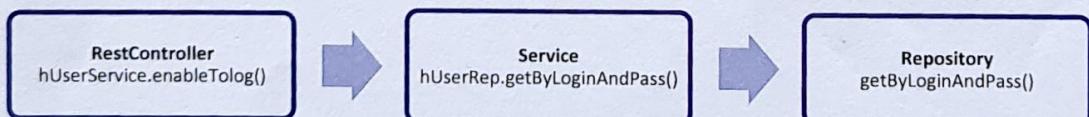
PostMapping pour envoyer des données par exemple lors de l'identification du user, on envoie au back le login et le password pour vérifier qu'il existe bien dans la BDD.

DeleteMapping permet par exemple de supprimer une liste de panier d'un user.

```
24 @Slf4j
25 @RestController
26 @RequestMapping("/index")
27 @CrossOrigin(origins = "*")
28 public class UserRestController {
29
30     @Autowired
31     IUserService hUserService;
32
33     @Autowired
34     IArticleService hArticleService;
35
36     /**
37      * Permet de logger un User
38      * @param user {@link Account}
39      * @return User {@link Account}
40      */
41     @PostMapping(path="/user", produces= "application/json")
42     public Account getCurrentConnectUser(@RequestBody Account user) {
43
44         Account p_user = hUserService.enableTolog(user.getLogin(), user.getPassword());
45
46         Log.info("user authentifié : " + p_user);
47
48         return p_user;
49     }
50 }
```

Le code ci-dessus est l'exemple qui me permet d'identifier un user via son login et son password.

J'ai annoté en premier lieu la classe avec @RestController et @RequestMapping("/index"). J'ai aussi annoté la méthode avec @PostMapping, en précisant le path de l'uri et ce qu'il va produire (sous format json). Et ensuite je vais faire appel aux différentes couches pour arriver jusqu'à la BDD et vérifier si ce que la requête envoyée correspond à un enregistrement dans la BDD.



En dernier lieu, le package model qui se compose des beans et comme évoqué précédemment étroitement liés avec les tables de la BDD car avec certaines annotations, les tables de la BDD peuvent être créées à partir de ces classes.

DOSSIER PROFESSIONNEL (DP)

```
26 @Data
27 @Entity
28 @Table(name = "T_Panier_SPRING")
29 public class Panier implements Serializable {
30
31     private static final long serialVersionUID = 2608804565680180566L;
32
33     /**
34      * identifiant de la ligne de panier {@link Integer}
35      */
36     @Id
37     @GeneratedValue(strategy = GenerationType.IDENTITY)
38     private int idPanier;
39
40     /**
41      * Relie un Panier à un utilisateur {@link Account}
42      */
43     @ManyToOne(targetEntity = Account.class, fetch = FetchType.EAGER)
44     @JoinColumn(name="idUser", referencedColumnName = "idUser", nullable = false)
45     private Account User;
```

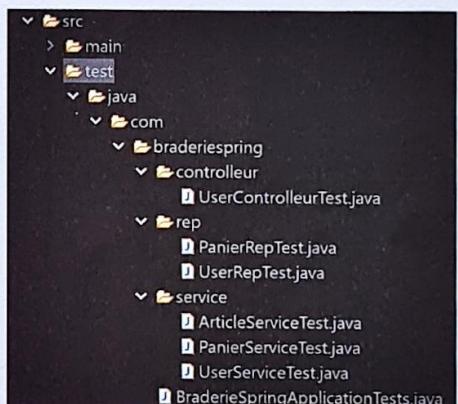
@JoinColumn indique une clé étrangère qui pointe sur idUser

Pour chacune des couches, nous avons mis en place des **tests unitaires** avec Junit et tester que chaque méthode de la couche fonctionnait bien avant de passer à la couche suivante.

Donc, après avoir développé la couche repository, nous avons mis en place les tests unitaires permettant de vérifier que l'on récupère bien ce qu'on attend. Cela permet ainsi d'exclure la couche testée s'il y a un bug.

Ci-contre l'arborescence du package test. On peut ainsi retrouver les 3 couches de l'application.

@Entity indique que la classe est persistente
@ Table pour renommer le nom de la table (optionnel)
@Id et **@GeneratedValue** indique que ce sera la clé primaire et que cette dernière sera générée à partir de l'identité propre au SGBD
@ManyToOne car plusieurs lignes de panier peuvent être associées à un User



Dans l'exemple ci-dessous, j'ai vérifié que la méthode **save** panier fonctionnait. Pour cela, j'ai créé un user, un article et j'ai utilisé la méthode **save** de la couche repository pour enregistrer le panier dans la BDD. Une fois la méthode **save** appelée, je viens vérifier grâce à la méthode **getListPaniers** qu'un panier existe bien en utilisant un **assertThat equals 0** qui doit retourner **false** car j'attends 1 en retour.

```
24 @SpringBootTest
25 public class PanierRepTest {
26
27     @Autowired
28     IPanierRep hPanierRep;
29
30     @Test
31     public void getListPaniers() {
32
33         User hUser = new User(1, "Marleyb", "marleyb123", 0);
34
35         Article hArticle = new Article(6, "Velo", "Decathlon", 190);
36
37         Panier hPanier = new Panier(hUser, hArticle, 6);
38
39         hPanierRep.save(hPanier);
40
41         List<Panier> result = hPanierRep.getListPaniers(1);
42
43         Integer expected = 0;
44
45         assertThat(expected.equals(result.size())).isFalse();
46
47     }
48
49 }
```

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

IDE : Eclipse

Langages travaillés : Java (Spring)

Plateforme d'intégration : GitLab

3. Avec qui avez-vous travaillé ?

Jérôme Clavier (CDA7), Stéphane Kouotze (CDA7) et Romain Ramuscello (CDA7)

4. Contexte

Nom de l'entreprise, organisme ou association ► AFPA

Chantier, atelier, service ► Projet interne (Braderie)

Période d'exercice ► Du : 05/07/2021 au : 21/07/2021

5. Informations complémentaires (facultatif)

Travail collaboratif de la conception au développement de l'application

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Master 2 Marketing Stratégique spécialisation Design Packaging	CEPE – IAE de Poitiers	2011
Master 1 Marketing	IAE Toulouse - TSM	2010
Licence 3 Marketing	IAE Toulouse -TSM	2009
DUT Gestion des Entreprises et des Administrations	IUT Toulouse – Paul Sabatier	2008
CAM Diploma Marketing Digital	Oxford College of Marketing	2017

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

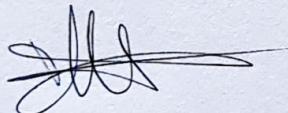
Je soussigné(e) [prénom et nom] Sandrine CABROL

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 BALMA le 13/12/2021

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

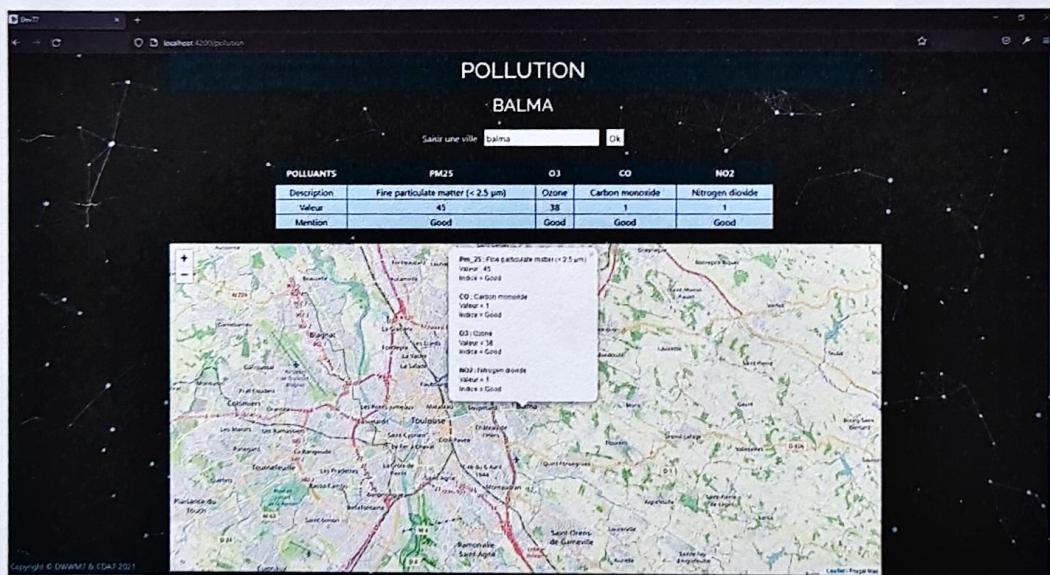
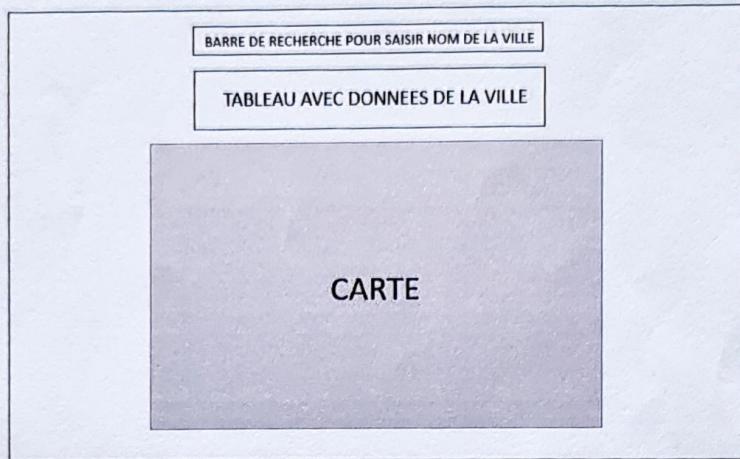
Intitulé

Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

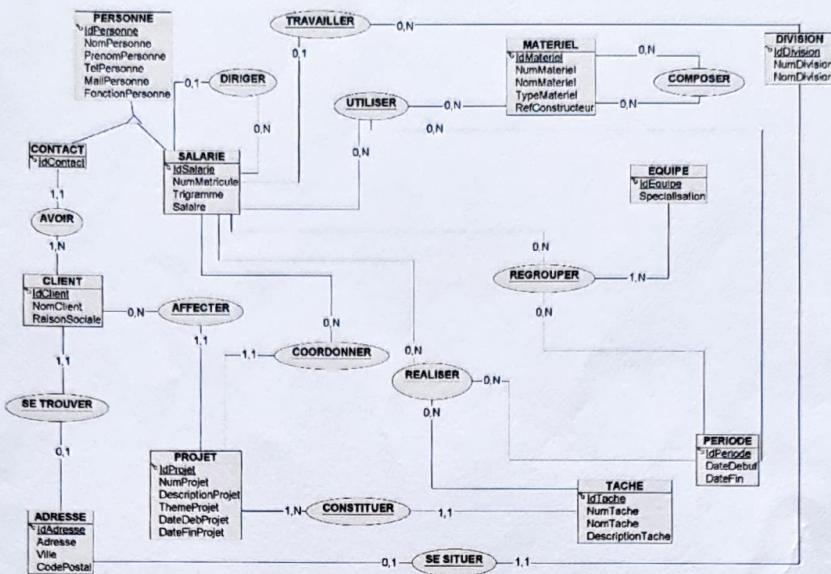
ANNEXES

Annexe 1 : Exemple de la maquette réalisée VS. déploiement du composant Pollution – projet Angular

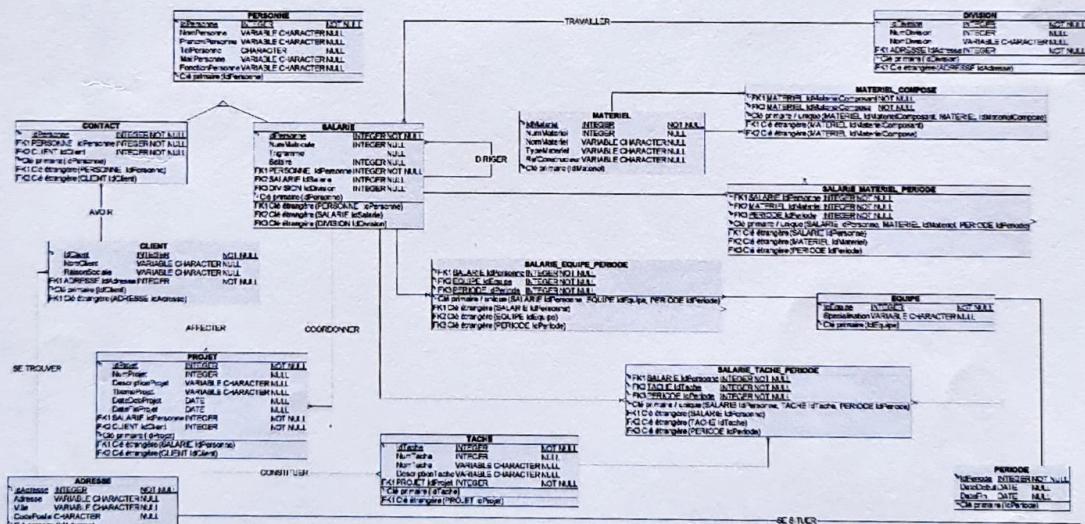


DOSSIER PROFESSIONNEL (DP)

Annexe 2 : réalisation du MCD et MLD pour la base de données InfoPlus



- Modèle Conceptuel de Données -



- Modèle Logique de Données -