



# DOSSIER PROFESSIONNEL (DP)

*Nom de naissance*

▸ Kouotze

*Nom d'usage*

▸

*Prénom*

▸ Stephane

*Adresse*

▸ 5 Avenue de Grande Bretagne  
31300 Toulouse

## Titre professionnel visé

CONCEPTEUR DEVELOPPEUR D'APPLICATIONS

### MODALITÉ D'ACCÈS :

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

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 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>

## Sommaire

### Exemples de pratique professionnelle

|  |           |
|--|-----------|
| <b>Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité</b> | <b>p.</b> |
| - Intitulé de l'exemple n° 1 : Projet Rocket Labs  | p.        |
| - Intitulé de l'exemple n° 2   | p.        |
| - Intitulé de l'exemple n° 3   | p.        |
| <b>Concevoir et développer la persistance des données en intégrant les recommandations de sécurité</b>             | <b>p.</b> |
| - Intitulé de l'exemple n° 1 : Étude de cas infoPlus   | p.        |
| - Intitulé de l'exemple n° 2   | p.        |
| - Intitulé de l'exemple n° 3   | p.        |
| <b>Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité</b>   | <b>p.</b> |
| - Intitulé de l'exemple n° 1 : La grande braderie  | p.        |
| - Intitulé de l'exemple n° 2   | p.        |
| - Intitulé de l'exemple n° 3   | p.        |
| <b>Intitulé de l'activité-type n° 4</b>  | <b>p.</b> |
| - Intitulé de l'exemple n° 1   | p.        |
| - Intitulé de l'exemple n° 2   | p.        |
| - Intitulé de l'exemple n° 3   | p.        |
| <b>Titres, diplômes, CQP, attestations de formation <i>(facultatif)</i></b>  | <b>p.</b> |
| <b>Déclaration sur l'honneur</b>   | <b>p.</b> |
| <b>Documents illustrant la pratique professionnelle <i>(facultatif)</i></b>  | <b>p.</b> |
| <b>Annexes <i>(Si le RC le prévoit)</i></b>  | <b>p.</b> |

# **EXEMPLES DE PRATIQUE PROFESSIONNELLE**

## Activité-type 1

Exemple n°1 -

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

Projet Rocket Labs

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

Dans le cadre de la formation, il nous a été demandé de développer une interface utilisateur web, consommant des ressources trouvables librement sur internet.

Cette activité devait être réalisée en groupe, ainsi après recherche et délibération, notre choix s'est porté sur une API renvoyant des données au format JSON faite par un passionné (disponible sur GitHub) concernant la société Rocket Labs(NZ). Cette API délivre les informations des différentes missions.

Nous avons décidé de fonctionner en méthode Agile SCRUM, le formateur tenant le rôle de Product Owner. J'ai été élu Scrum master, ayant pour rôle d'animer les Daily meetings, mettre à jour le Trello et faire un rapport quotidien. J'ai aussi tenu le rôle de leader technique, apportant ainsi mon soutien à l'équipe en cas de difficultés. J'ai également créé un dépôt distant afin de faciliter le contrôle de version.

Les attendus étaient de livrer une application multi-composant qui soit responsive.

Nous avons donc, dans un premier temps, étudié l'API afin de déterminer quelles seraient les données pertinentes à afficher. Nous avons par la suite maqueté notre composant (tableau blanc) et ainsi pu assigner les différentes tâches aux affinités et ressenti de chacun.

Première étape: Analyse des données

```
{
  flight_number: 3,
  mission_name: "It's Business Time",
  upcoming: false,
  launch_year: "2018",
  launch_date_unix: 1541987200,
  launch_date_utc: "2018-11-11T01:51:00.000Z",
  launch_date_local: "2018-11-11T16:51:00+13:00",
  rocket: { _id: "5a1000200",
    telemetry: null,
  },
  launch_site: {
    name: "Mahia 10-1",
    name_long: "Rocket Lab Launch Complex 1"
  },
  launch_success: true,
  links: {
    articles: [ _ ],
    videos: [ _ ],
    flickr: [ _ ],
    mission_patch: [ _ ]
  },
  details: "The 11 November 2018 launch was successful; all cubesats planned to be deployed were deployed in orbit. controller issue. In October 2018, a nine-day launch window was announced starting 11 November 2018."
}
```

Figure 1: Données JSON

L'API JSON nous renvoie un tableau de données, celui-ci contenant plusieurs sous-tableaux et des liens de redirection.

Le choix a été porté sur certaines d'entre elles et aussi de remplacer les liens morts (essentiellement des images et/ou vidéo) par une image libre de droit trouvée sur internet.

# DOSSIER PROFESSIONNEL (DP)

## Deuxième étape: maquettage de l'interface

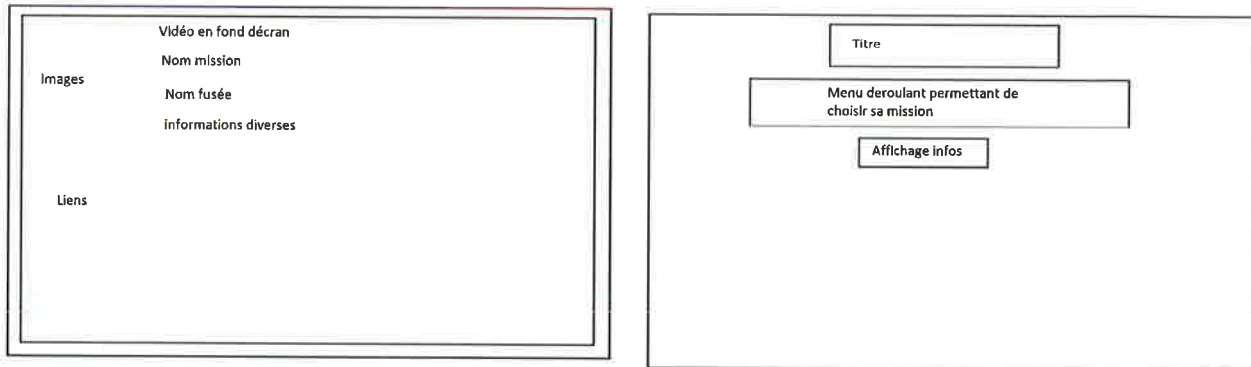


Figure 2: Maquettage de l'application

La maquette a été faite sur tableau blanc, puis présentée au PO, qui l'a validée.

## Troisième étape: Développement

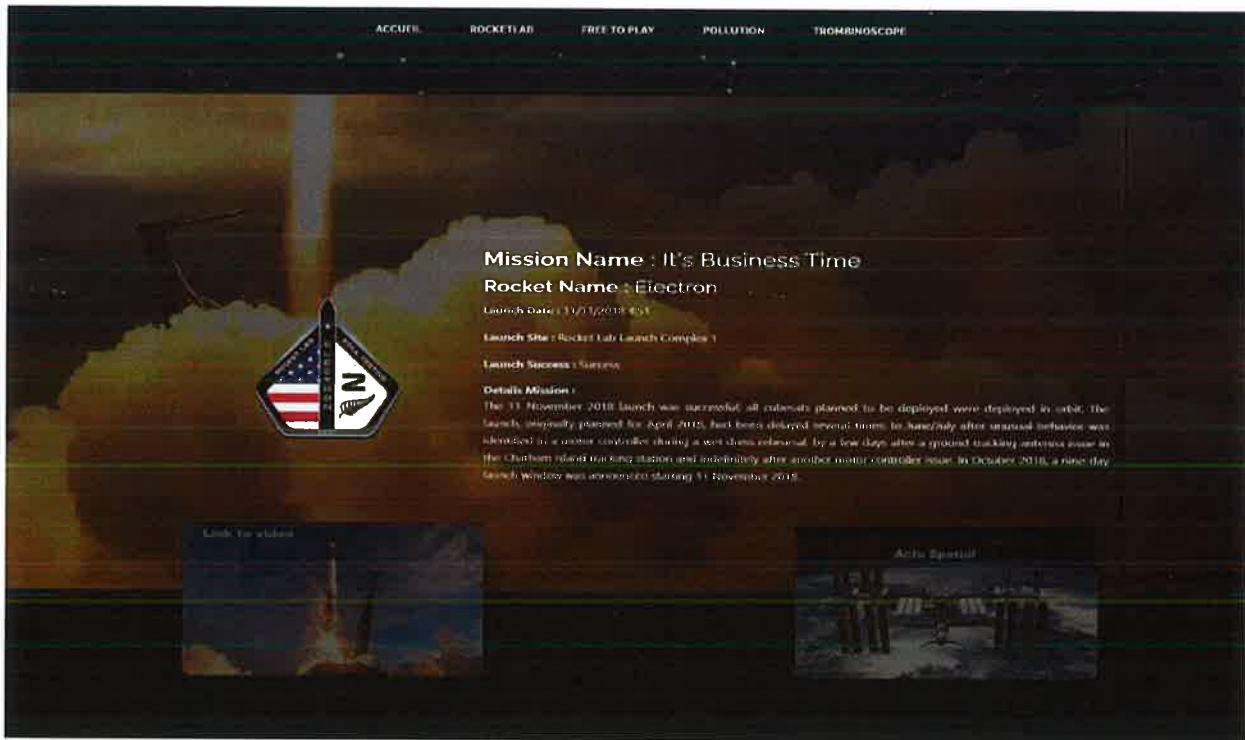


Figure 3: rendu final de l'application

Nous avons pour cela utiliser l'IDE Visual Code Studio avec le framework Angular 12+ ainsi que l'outil de contrôle de version GitLab.

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

Nous avons commencé le développement en y allant couche par couche:  
D'abord la couche service, qui s'occupe de récupérer les données.

```
export class ServiceNasaService {  
  
  //recuperation des données via l'url  
  url: string = 'https://rocketlabapi.herokuapp.com/v1/launches/';  
  
  //injection de dépendance  
  constructor(private http: HttpClient) { }  
  //permet de récupérer l'url  
  // type Observable : permet de récupérer des données dans le temps,  
  // c'est à dire soit à l'instanciation soit à chaque fois que l'on  
  // appelle la classe  
  public getNasaData() : Observable<any> {  
  
    // get retourne le fichier json courant (appelable depuis d'autre classe)  
    return this.http.get(this.url);  
  }  
}
```

Figure 4: Couche Service



## DOSSIER PROFESSIONNEL (DP)

Ensuite la partie Dynamique du composant.

Les informations, récupérées depuis la couche service, sont manipulées afin de renvoyer à l'interface utilisateurs celles que nous avons choisies.

```
constructor(private ServiceNasa: ServiceNasaService) { }

ngOnInit(): void {

    //la méthode subscribe permet de souscrire à un observable
    //et à être notifié des nouvelles valeurs erreur ou fin du stream
    this.ServiceNasa.getNasaData().subscribe(
        data => {
            this.dataView = data;
        }
    )
}

// evenement de type any qui recupere l'evenement
selectValue(event:any){
    return this.selectedOption = event.target.value;
}

afficherInfo(): void{
    console.log(this.missionpatchOK);
    this.missionpatchOK = true;
    this.missionName = this.dataView[this.selectedOption].mission_name;
    this.rocketName = this.dataView[this.selectedOption].rocket.name;
    this.dateLancement = this.dataView[this.selectedOption].launch_date_utc;
    this.launchSite = this.dataView[this.selectedOption].launch_site.name_long;
    this.launchSuccess = this.dataView[this.selectedOption].launch_success;
    this.missionPatch = this.dataView[this.selectedOption].links.mission_patch_small;
    this.detailsMission = this.dataView[this.selectedOption].details;
    this.link = this.dataView[this.selectedOption].links.articles[0];
    this.video = this.dataView[this.selectedOption].links.videos[0];
    this.showInfo = true;

    console.log(this.missionpatchOK);
    if(this.missionPatch == null || this.missionPatch.startsWith(this.wrongURL)){
        this.missionpatchOK = false;
    }
    console.log(this.missionpatchOK);
}
}
```

Figure 5: manipulation des données



# DOSSIER PROFESSIONNEL (DP)

Pour terminer, l'interface utilisera des fonctionnalités internes d'Angular.  
Les fonctionnalités utilisées sont la condition et la boucle.

```
<div *ngIf="showInfo" class="row" style="border-radius: 10px;">
  <div class="info-nasa">
    <video autoplay="true" loop="loop" poster="finVideo.png" id="decollage">
      <source src="assets/video/decollage.webm" type="video/webm">
    </video>
  </div>
  <select class="select-nasa" class="form-select" (change)=selectValue($event)>
    <option value="">--Please choose a rocket--</option>
    <option *ngFor="let item of this.dataView; let i = index" value="{{i}}"> {{item.flight_number}}
      -{{item.mission_name}} </option>
  </select>
  <h2><span>Mission Name :</span> {{ missionName }}</h2>
  <h3><span>Rocket Name :</span> {{ rocketName }}</h3>
  <!-- Modification de la date grace au pipe '|' et on lui donne le format que l'on veut
  <p><span>Launch Date :</span> {{ dateLancement | date: 'dd/MM/yyyy h:mm' }} </p>
  <p><span>Launch Site :</span> {{ launchSite }}</p>
```

Figure 6: diverses fonctionnalités du front end

La partie habillage de l'interface passe par une feuille de style.

Exemple, celle de la vidéo, où il est défini :

Sa largeur, la position du bloc par rapport à l'élément précédent, son décalage à gauche, sa profondeur, la position de l'élément en lui-même, la façon dont elle apparaît et son opacité.

```
.info-nasa video {
  width: 100%;
  margin-top: 27%;
  left: 50%;
  z-index: -100;
  transform: translateX(-50%) translateY(-50%);
  transition: 1s opacity;
  filter: brightness(50%);
}
```

Figure 7: CSS

## 2. Précisez les moyens utilisés:

Visual studio Code (Angular 12, NodeJS)  
Suivi de production (Trello)  
Contrôle de version (GitLab)

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 3. Avec qui avez-vous travaillé ?

Soulié Rosalie (DW), Ramuscello Romain (CDA), Clavier Jérôme (CDA)

## 4. Contexte

Nom de l'entreprise, organisme ou association ► AFPA

Chantier, atelier, service ► Site interne

Période d'exercice ► Du : 09/08/2021 au : 13/08/2021

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 2

Exemple n°1

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

Étude de cas infoPlus

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

Afin de valider les compétences acquises sur la persistance des données, notre formateur nous a soumis une étude de cas: concevoir et développer une BDD de gestion de stock pour la société fictive «Infoplus».

Le formateur, jouant le rôle du maître d'ouvrage, nous a fourni le cahier des charges déterminant les exigences et les attentes du projet.

Il était par exemple imposé de développer une base de données relationnelle.

Il nous éclairait également quand certaines directives étaient ambiguës.

#### Première étape: Analyse des données

Cette première phase a pour but d'identifier les différents éléments qui constitueront la BDD, comme les entités, les différents événements qui seront inclus....

Pour cela, j'ai établi un dictionnaire de données, via Microsoft Excel, illustrant les identifiants, les données qui seront persistées ou calculées, les actions lors de manipulations sur les données (triggers), les générations automatiques de données (fonctions).

| NOM             | DESCRIPTION                          | TYPE    | IDENTIFIANT | MINOR       | MODALITÉ DE | COMMENTAIRE                             |
|-----------------|--------------------------------------|---------|-------------|-------------|-------------|---|
|                 |                                      |         |             | D'OBTENTION | CALCUL      | DOMAINE DE VALEURS                      |
| <b>Division</b> |                                      |         |             |             |             |   |
| id_division     | Identifiant de l'Entreprise Infoplus | serial  | X           | mémorisé    |             |   |
| nom             | Nom de la division                   | varchar |             | mémorisé    |             | Infoplus Montreal, infoplus annecy      |
| adresse         | Adresse de la division               | varchar |             | mémorisé    |             | 1218 Boulevard pie IX H2P 2A2 Montréal, |
| <b>Salariés</b> |                                      |         |             |             |             |   |
| id_salaries     | Identifiant de la table salaries     | serial  | X           | mémorisé    |             |   |
| matricule       | matricule du salarié                 | int     |             | mémorisé    |             | employé N°12456                         |
| nom             | nom du salarié                       | varchar |             | mémorisé    |             | Maxley                                  |
| pre_nom         | prenom du salarié                    | varchar |             | mémorisé    |             | Bob                                     |
| numtel          | numtel du salarié                    | varchar |             | mémorisé    |             | 33-12-34-56-78-91                       |
| trigramme       | trigramme du salarié                 | char    |             | mémorisé    |             | Bob Maxley => BMA, John Lennon => JLE   |
| mail            | mail du salarié                      | varchar |             | mémorisé    |             | bob.maxley@infoplus.com                 |
| fonction        | Indiquer la fonction du salarié      | varchar |             | mémorisé    |             | dev, scrum master, PO                   |
| salaire         | Indiquer le salaire                  | char    |             | mémorisé    |             | 35K                                     |
| <b>Équipes</b>  |                                      |         |             |             |             |   |
| id_Equipe       | Identifiant de l'équipe              | serial  | X           | mémorisé    |             |   |
| pole_competence | spécialisation de l'équipe           | varchar |             | mémorisé    |             | R&D, support                            |

Figure 8: Dictionnaire des données

# DOSSIER PROFESSIONNEL (DP)

## Deuxième étape: Modèle Conceptuel de Données (MCD)

Cette phase consiste à établir un premier modèle de la BDD, en utilisant la méthode Merise et le logiciel Open Modèle Sphère.

Il s'agit d'une représentation abstraite de ce qui sera implémenté dans la base de données.

Les actions entre les tables sont identifiées et nommées à l'aide de verbes d'actions.

Les cardinalités sont également définis à cette étape.

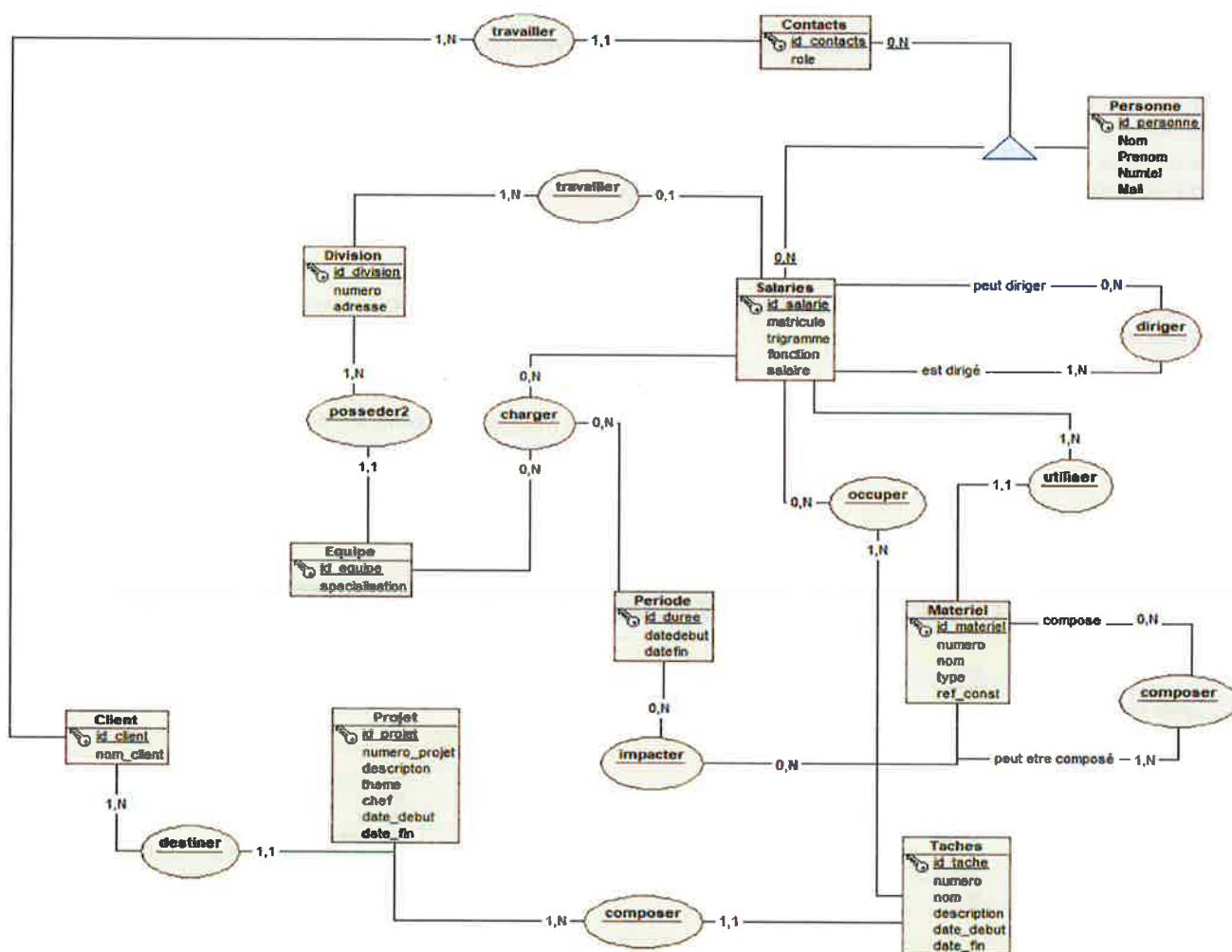


Figure 9: Modèle Conceptuel des Données

# DOSSIER PROFESSIONNEL (DP)

## Troisième étape: Modèle Logique de Données (MLD)

À partir du Modèle Conceptuel de Données, j'ai déterminé la création de table d'association et la migration des clés étrangères.

Le MLD est une représentation visuelle de ce qui sera implémenté dans la base de données.

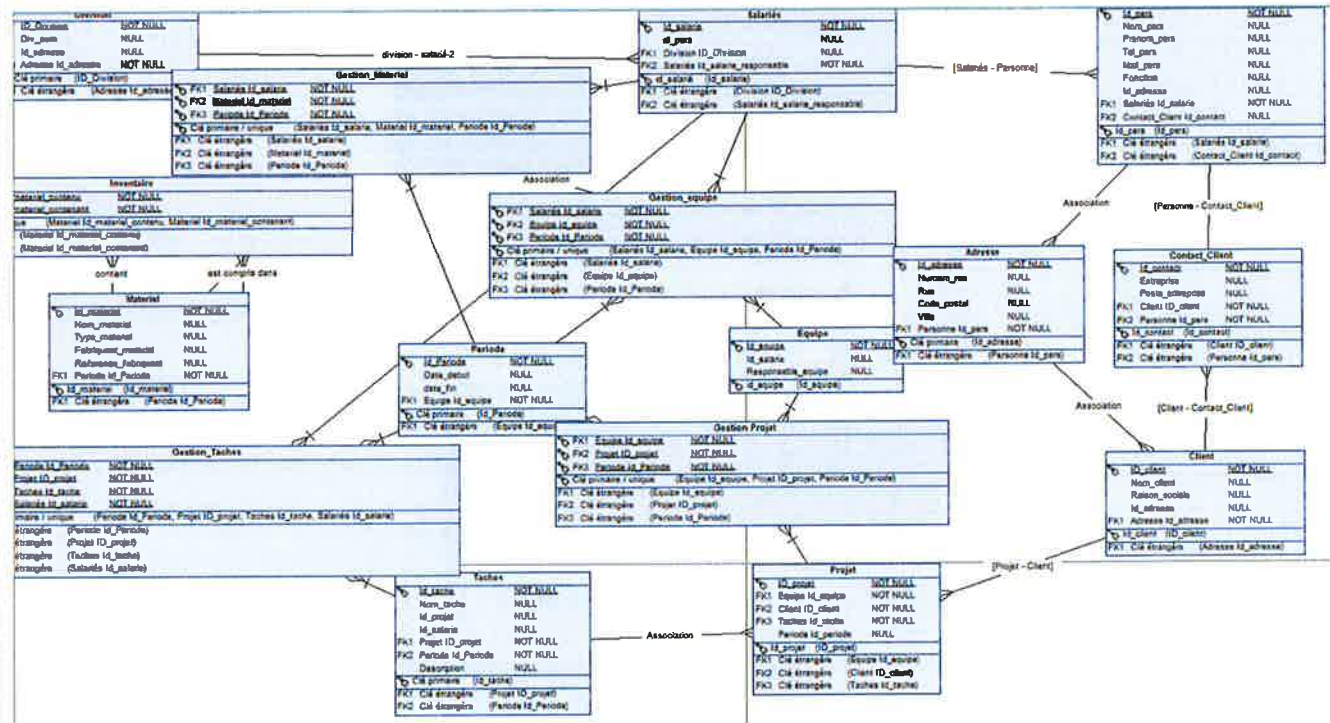


Figure 10: Modèle Logique des Données



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## Quatrième étape: Développement

Une fois le LMD établi, il est temps de développer la base de données.

Dans un premier temps, j'ai créé un profil utilisateur en s'assurant d'abord que celui-ci n'existe pas.

```
-- supprimer l'user si il existe déjà
DROP USER IF EXISTS entreprise;

-- créer l'user
CREATE USER entreprise ;

-- modifier son role
ALTER ROLE entreprise with
LOGIN
ENCRYPTED PASSWORD 'afpa123'
CREATEDB
```

Figure 11: Création d'un utilisateur

Une fois logger via l'utilisateur créé, j'ai implémenté une data base.

```
create database entreprise with
owner entreprise
connection limit -1
```

Figure 12: Création d'une base de données

La création des tables s'appuie sur le LDD (Langage de Définition des Données). J'ajoute également les contraintes liées aux clés primaires et étrangères.

```
DROP TABLE if exists salaries cascade;

CREATE TABLE IF NOT EXISTS salaries(
  id_salarie SERIAL NOT NULL,
  id_division INT,
  id_manager INT,
  matricule INT NOT NULL,
  trigramme CHAR(3),
  nom VARCHAR(20) NOT NULL,
  prenom VARCHAR (20) NOT NULL,
  numtel VARCHAR(12) NOT NULL,
  mail VARCHAR(30) NOT NULL,
  Fonction VARCHAR(25) NOT NULL,
  salaire INT NOT NULL
);

ALTER TABLE salaries ADD PRIMARY KEY (id_salarie);

-- Clé étrangère division sur la table salaries
ALTER TABLE salaries
ADD CONSTRAINT fk_salaries_division FOREIGN KEY (id_division) REFERENCES division (id_division)
ON DELETE CASCADE;
```

Figure 13: Création des tables



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

Une des contraintes imposées par le maître d'ouvrage était que lorsque l'on insère une nouvelle tâche, celle-ci devait être formaté en majuscule. J'ai donc créé un trigger qui avant insertion ou mise à jour permet de transformer la tâche.

```
-- creation du trigger avant l'insertion, car on modifie ce qui va etre insere
CREATE TRIGGER TRG02 BEFORE INSERT OR UPDATE of nom_taches
ON taches for each row execute function F_MAJ();

-- création de la fonction appelé par le trigger
CREATE OR REPLACE FUNCTION F_MAJ()
RETURNS TRIGGER AS $$

BEGIN
-- mise en forme (upper avant insertion)
    NEW.nom_taches = upper(NEW.nom_taches);
    RETURN NEW;

END;
$$
LANGUAGE PLPGSQL
```

Figure 14: Triggers

## Cinquième étape: Phase de tests

Afin d'assurer le bon fonctionnement de l'ensemble, la phase de tests à été effectuée à l'aide de script manuel. Les opérations de commit et de rollback ont été configurées sur manuel, donnant plus de liberté et de sécurité lors de cette phase.

Ils se découpent en 6 étapes:

- 1 – Création de la data base
- 2 – Création des triggers
- 3 – Effacement de la data base
- 4 – Réinsertion des données et vérification du déclenchement des triggers
- 5 – Création de la vue et exécution
- 6 – Création des requêtes

## 2. Précisez les moyens utilisés :

Modélisation Merise (outils: open Modelsphere).  
Microsoft Excel  
Langage PL/PGSQL  
PostgreSQL avec l'interface PgAdmin

## 3. Avec qui avez-vous travaillé ?

Seul

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 4. Contexte

Nom de l'entreprise, organisme ou association ▸ AFPA  
Chantier, atelier, service- Site interne  
Période d'exercice ▸ Du : /08/2021 au : 13/08/2021

## 5. Informations complémentaires (facultatif)

# DOSSIER PROFESSIONNEL (DP)

## Activité-type 3

Exemple n° 1 -

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

La grande braderie

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

Ce projet final était l'occasion de mettre en pratique toutes les compétences acquises lors de la formation.

Le formateur, toujours en tant que maître d'ouvrage, nous a soumis un projet full stack de plateforme de eCommerce, à faire en groupe. Il nous a également fourni un cahier des charges déterminant les exigences et les attentes du projet.

Les exigences étaient que les utilisateurs devaient pouvoir s'identifier et se déconnecter, qu'un utilisateur avec profil administrateur puisse avoir accès à des options supplémentaires. Aussi d'avoir un prix total qui s'affiche en permanence ainsi que les informations de l'article et sa quantité.

La page administrateur sert à modifier les articles présents dans la base de données.

#### Conception :

Le projet sera une API RESTful, fournissant des services sur lesquels notre partie frontend viendra s'appuyer pour récupérer les services fournis par le backend.

Le design pattern MVC sera appliqué, les modèles étant les objets contenant les données de l'application, la vue pour afficher les informations (ici, elle sera assurée par Angular) et les contrôleurs établissant la communication entre les parties backend et frontend.

Maven nous servira à structurer le projet et JUnit pour l'exécution des tests.

Les frameworks Spring Boot et Angular seront utilisés dans ce projet afin de faciliter le travail.

Concernant Spring Boot, nous utiliserons Spring Data JPA afin de faciliter les échanges avec la base de données (via son ORM ou Object Relational Mapper) et Spring MVC afin de gérer les requêtes HTTP.

Spring Boot contient un système d'annotations permettant de générer des beans (objets gérés automatiquement).

Notre application s'appuie sur 3 modèles, également présent en tant que table dans une base de données:

- le modèle Account comprenant les informations utilisateurs
- le modèle Article comprenant les informations des articles
- le modèle Panier comprenant des informations relatives aux 2 autres modèles

# DOSSIER PROFESSIONNEL (DP)

## Première étape: Spring data JPA

Pour la création des modèles, l'utilisation de Spring data JPA et de ses annotations permettent de rattacher le bean à la base de données.

```
@Data
@Entity
@Table(name = "T_Panier_SPRING")
public class Panier {

    /**
     * identifiant de la ligne de panier {@link Integer}
     */
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idPanier;

    /**
     * Relie un Panier a un utilisateur {@link Account}
     */
    @ManyToOne(targetEntity = Account.class, fetch = FetchType.EAGER)
    @JoinColumn(name="idUser", referencedColumnName = "idUser", nullable = false)
    private Account User;

    /**
     * Relie un Panier a un Article {@link Article}
     */
    @ManyToOne(targetEntity = Article.class, fetch = FetchType.EAGER)
    @JoinColumn(name="idArticle", referencedColumnName = "idArticle", nullable=false)
    private Article Article;

    /**
     * Permet de stocker la quantité commandé par l'utilisateur {@link Integer}
     */
    @Column(nullable = false, length = 5)
    private int quantite;

    /**
     * Valeur non sauve, mais utile pour le bon déroulement de l'appli {@link Integer}
     */
    @Transient
    private int prix;
```

Figure 15: Modèle

Les annotations nous servent à définir:

- si la table sera persistée (@Entity)
- son nom dans la BDD (@Table)
- sa clé primaire et comment elle est générée (@ID/@GeneratedValue)
- les cardinalités (@ManyToOne, @JoinColumn)
- les champs (@Column)
- l'indication de persistance d'un attribut (@Transient)

# DOSSIER PROFESSIONNEL (DP)

## Deuxième étape: Repository

JpaRepository est une interface qui permet d'implémenter les méthodes du CRUD, sans avoir à les définir explicitement, sauf en cas de requête particulière (@Query).

```
@Repository
public interface IArticleRep extends JpaRepository<Article, Integer> {
}
```

Figure 16: Couche DAO

## Troisième étape: Couche service

Les services ou couche métier, permettent d'appliquer une logique (calcul de prix, identification) entre les contrôleurs et les repository (et inversement). Elle inclut des méthodes non présentes dans le CRUD et augmente la séparation des responsabilités de l'application.

Ici, l'utilisation d'une annotation @Autowired permet d'utiliser un bean. Il est également préférable d'instancier une interface, diminuant la quantité de code à modifier en cas d'évolution.

```
@Service
public class ArticleService implements IArticleService{

    @Autowired
    IArticleRep hArticleRep;

    /**
     * Permet de retourner la liste des articles
     * @param idArticle {@link Integer}
     * @return list panier {@link Optional} {@link Article}
     */
    @Override
    public Optional<Article> getArticle(int idArticle) {

        Optional<Article> hArticle = null;

        hArticle = hArticleRep.findById(idArticle);

        return hArticle;
    }
}
```

Figure 17: Service

## DOSSIER PROFESSIONNEL (DP)

### Quatrième étape: Couche Contrôleur

La couche contrôleur est celle qui gère les services Restful, à l'aide d'annotations s'inspirant du CRUD. Elle s'appuie sur la couche métier lors des traitements.

```
@RestController
public class MagasinRestController {

    @Autowired
    ArticleService hArticleService;

    /**
     * Permet de retourner la liste des articles
     * @return list panier (@link List) (@link Article)
     */
    @GetMapping(path="/getAllArticle", produces= "application/json")
    public List<Article> getAllArticle(){

        return hArticleService.getAllArticle();
    }
}
```

Figure 18: Contrôleur

Pour vérifier le bon fonctionnement des services Rest, il a été décidé d'implémenter swagger. De par son interface, elle permet une utilisation facile et rapide des Rest tout en indiquant explicitement les URLs de connexion.

### JRSS Braderie Boot REST API documentation

REST APIs For Managing Article, cart in Braderie

Created by Team JRSS

See more at [les CDAZ en puissance!!!!](#)

[Contact the developer](#)

admin-rest-controller : Admin Rest Controller Show/Hide List Operations Expand Operations

caddie-rest-controller : Caddie Rest Controller Show/Hide List Operations Expand Operations

magasin-rest-controller : Magasin Rest Controller Show/Hide List Operations Expand Operations

**DELETE** /magasin/clear/{id} deleteCaddie

**GET** /magasin/getAllArticle getAllArticle

Response Class (Status 200)  
OK

Model Schema

```
{
  "description": "string",
  "idArticle": 0,
  "marque": "string",
  "prix": 0
}
```

Response Content Type **application/json**

Response Messages

| HTTP Status Code | Reason       | Response Model | Headers |
|------------------|--------------|----------------|---------|
| 401              | Unauthorized |                |         |
| 403              | Forbidden    |                |         |
| 404              | Not Found    |                |         |

Try it out!

Figure 19: swagger



## DOSSIER PROFESSIONNEL (DP)

### Autre: Tests

L'application a été développée selon le principe de TDD (test Driven Design), permettant ainsi de tester les différentes données traitées et comportements correspondant aux attentes.

```
@SpringBootTest
public class IUserRepTests {

    @Autowired
    IUserRep userRep;

    private Account createUser() {
        String login = "Robert";
        String pass = "robert123";
        int nbConnect = 0;
        String role = "user";

        Account newAccount = new Account(9, login, pass, nbConnect, role);

        return newAccount;
    }

    @Test
    public void saveUser() {

        Account userSaveAccount = createUser();

        userRep.save(userSaveAccount);
    }

    @Test
    public void readUser() {

        saveUser();

        Account expectedUser = userRep.getById(9);


        Account testUser = createUser();

        assertEquals(expectedUser.getIdUser(), testUser.getIdUser());
    }
}
```

Figure 20: tests

# DOSSIER PROFESSIONNEL (DP)

– Frontend –



Merci de vous connecter

Identifiant

Login

Mot de passe

Password

Connexion

### LISTE DES ARTICLES

| DESCRIPTION    | MARQUE         | PRIX(€) | QUANTITE                           | AJOUTER |
|----------------|----------------|---------|------------------------------------|---------|
| Stylo Bleu     | Bic            | 1       | <input type="text" value="min=1"/> | AJOUTER |
| Ramette 80 grs | ClaireFontaine | 5       | <input type="text" value="min=1"/> | AJOUTER |
| Montre         | Rolex          | 200     | <input type="text" value="min=1"/> | AJOUTER |
| Rameur         | Decathlon      | 290     | <input type="text" value="min=1"/> | AJOUTER |
| equere         | be all right   | 3       | <input type="text" value="min=1"/> | AJOUTER |

PANIER VIDE

RESTAURER PANIER SUPPRIMER PANIER SAUVEGARDER PANIER

### LISTE DES ARTICLES

| DESCRIPTION    | MARQUE         | PRIX(€) | MODIFIER | SUPPRIMER |
|----------------|----------------|---------|----------|-----------|
| Stylo Bleu     | Bic            | 1       | MODIFIER | SUPPRIMER |
| Ramette 80 grs | ClaireFontaine | 5       | MODIFIER | SUPPRIMER |
| Montre         | Rolex          | 200     | MODIFIER | SUPPRIMER |
| Rameur         | Decathlon      | 290     | MODIFIER | SUPPRIMER |
| equere         | be all right   | 3       | MODIFIER | SUPPRIMER |

RETOUR

CREER UN NOUVEL ARTICLE

Figure 21: IHM

A l'aide de la méthode post et l'URL définie dans le back end, nous identifions l'utilisateur et sauvegardons ses données le temps de la session.

```

getUserFromDB(login: any, pass: any): Observable<any>{
  return this.httpClient.post(this.urlUser, {login, pass}, httpOptions);
}

```

Figure 22: communication back et front

# DOSSIER PROFESSIONNEL (DP)

Une phase additionnelle à été l'intégration de SonarQube afin de vérifier la qualité du code.

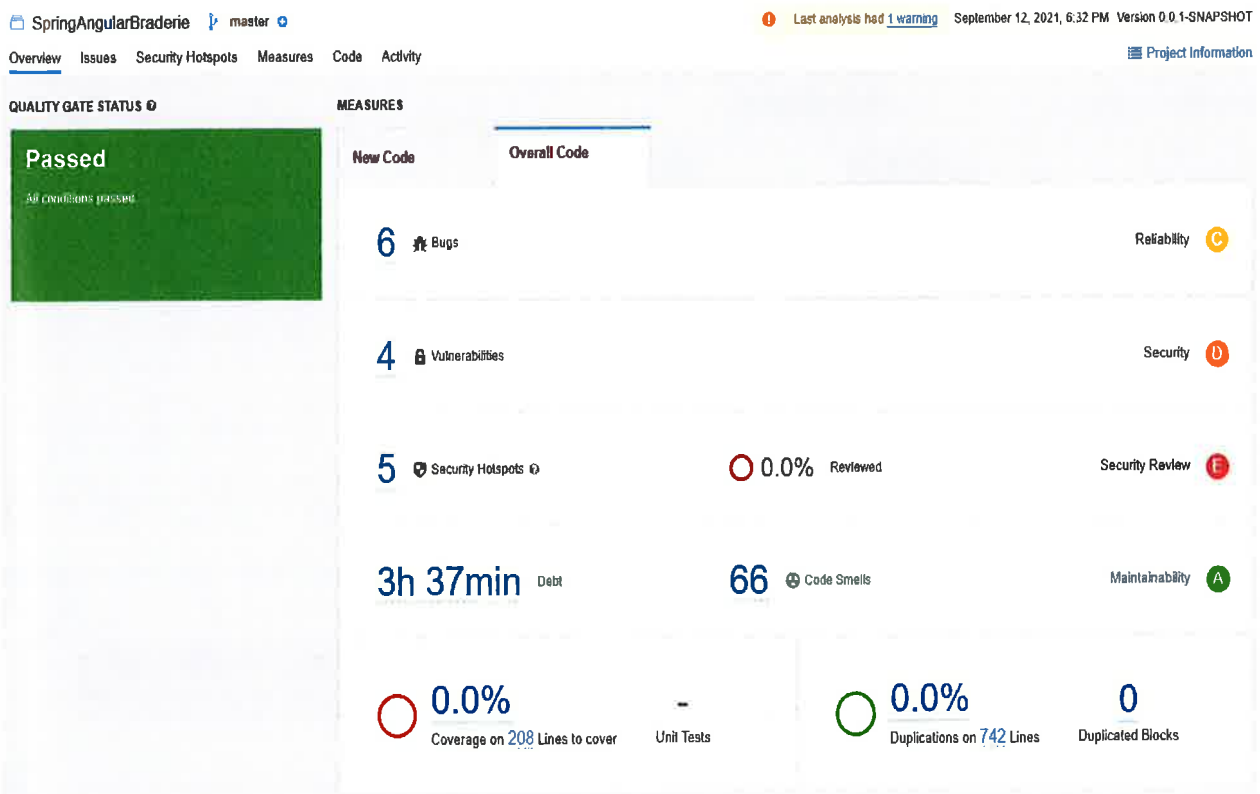


Figure 23: SonarQube

## 2. Précisez les moyens utilisés :

Eclipse IDE (Java 8 /Spring Boot)  
VS Code (Angular 12)  
PgAdmin (Postgresql)  
Revue de code (SonarQube)  
Intégration continue (GitLab)

## 3. Avec qui avez-vous travaillé ?

Cabrol Sandrine(CDA), Ramuscello Romain(CDA), Clavier Jérôme(CDA)

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## 4. Contexte

Nom de l'entreprise, organisme ou association - AFPA  
Chantier, atelier, service - projet interne  
Période d'exercice - Du : 12/07/2021 au : 26/07/2021

## 5. Informations complémentaires (facultatif)

Exemple de code page

## Titres, diplômes, CQP, attestations de formation

(facultatif)

| Intitulé     | Autorité ou organisme            |
|--------------|----------------------------------|
| Cliquez ici. | Cliquez ici pour taper du texte. |
|              |                                  |
|              |                                  |
|              |                                  |
|              |                                  |
|              |                                  |
|              |                                  |
|              |                                  |
|              |                                  |

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

|  |  |
|--|--|
|  |  |
|--|--|

## DOSSIER PROFESSIONNEL <sup>(DP)</sup>

### Déclaration sur l'honneur

Je soussigné Kouotze Stéphane ,

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 Toulouse

le 12/10/2021

pour faire valoir ce que de droit.

Signature :

A handwritten signature in blue ink, appearing to be 'Kouotze Stéphane', written over a horizontal line.



# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## Documents illustrant la pratique professionnelle

*(facultatif)*

| Intitulé |
|----------|
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |
|          |

# DOSSIER PROFESSIONNEL <sup>(DP)</sup>

## ANNEXES

*(Si le RC le prévoit)*