Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▸ Lenoir d'Espinasse
Nom d'usage ▸ Lenoir d'Espinasse
Prénom ▸ Guillaume
Adresse ▸ 5 rue de Lagréou, 31560 Nailloux

Titre professionnel visé

CONCEPTEUR DEVELOPPEUR D'APPLICATIONS

MODALITE D'ACCES :

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

Présentation du dossier

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

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

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

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

Pour prendre sa décision, le jury dispose :

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

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

Ce dossier comporte :

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

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

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

Sommaire

Exemples de pratique professionnelle

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1 DEVELOPPER DES COMPOSANTS D'INTERFACE

Exemple n°1 ▸ ***Création d'une interface web de gestion de scores de Golf***

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

Dans le cadre de mon parcours d'alternance au sein de l'entreprise ATOS sur le projet Mundi, j'ai eu l'occasion de découvrir un langage que nous n'avions pas vu pendant la formation. Il s'agit du langage Python. L'ayant utilisé à travers les Jupyter Notebook (une interface de développement en ligne), je n'avais pas eu l'occasion d'explorer le développement d'une application web comme nous avions pu le faire durant la formation avec Angular et Java.

De plus, j'avais connaissance que certains projets de Mundi pouvaient nécessiter de réaliser des applications web en utilisant le Framework FLASK. J'ai donc décidé de m'auto-former sur ce Framework Web et d'utiliser le cas d'un besoin personnel pour m'y exercer. En effet, étant golfeur, j'ai l'habitude de tenir à jour un fichier Excel avec mes scores de parties et les statistiques qui en découlent. J'ai donc décidé de basculer d'un fichier Excel à une interface Web.

Pour monter en compétence, j'ai utilisé différentes sources d'informations notamment des sites internet, et des vidéos sur Youtube. Je me suis formé aux domaines suivants : installation, création, fonctionnement, architecture, données dynamiques, base de données, ORM… Ceci m'a permis de démarrer le projet en avançant étape par étape.

Flask est ce qu'on appelle un microframework car il est très léger. Il suffit de quelques lignes de code pour créer une application après avoir installé le module via la commande « pip install flask » :

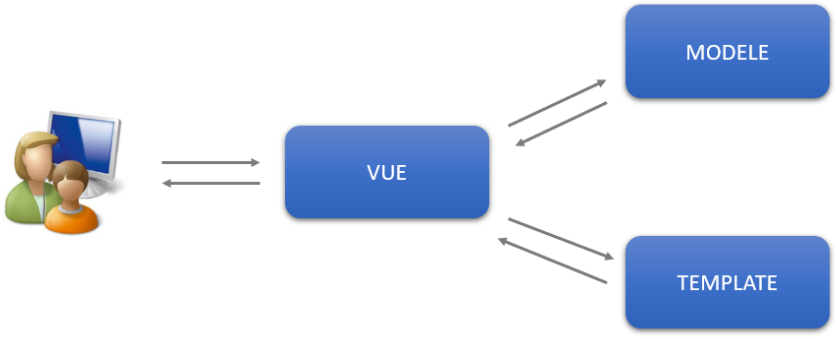
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "Hello world !"

if __name__ == "__main__":
    app.run()
```

D'un point de vue architecture, un projet Flask suit la structure MVT pour Modèle-Vue-Template propre aux Framework Python Django et Flask (cf. ci-dessous) :



La VUE joue le rôle central dans cette architecture car elle reçoit les requêtes http de l'utilisateur, interagit avec la base de données via le MODELE et l'utilisation d'un ORM (Object-Relational Mapping), puis sélectionne le bon TEMPLATE (fichier HTML) et ajoute les données et renvoi le tout en réponse de la requête. Voici ci-dessous un exemple concret de l'application :

Annotation contenant la route
Méthode appelée lorsque la route est demandée en requête HTTP
Utilisation MODELE pour recueillir des données en BDD via ORM
Appel du bon Template en associant les données puis retour en réponse à la requête

```
@app.route('/scoreslist')
def get_scoreslist():
    scores_list = db.session.query(score).all()
    return render_template("scoreslist.html", scores_list=scores_list)
```

L'utilisation d'un ORM est préconisée. Dans mon cas j'ai utilisé l'ORM SQLAlchemy qui est une extension directe du Framework Flask. Au niveau de la base de données, j'ai utilisé un Système de Gestion de Base de Donnée (SGBD), en l'occurrence SQLite qui est facile à installer et à utiliser. Pour le style des pages j'ai utilisé CSS et la librairie BOOTSTRAP.

Enfin, j'ai cherché à comprendre comment fonctionnait la couche MODELE. De manière assez similaire à ce qu'on avait vu avec le Framework Spring durant la formation il est nécessaire de créer des classes représentant les différentes tables de la base de données et de gérer les relations entre les tables par des annotations spécifiques.

Dans le cadre de cette application assez simple, j'ai eu besoin de créer deux classes (Score et golfCourse) et de définir les relations entre les deux :

- Un score n'est attribué qu'à un golfCourse
- Un golfCourse peut avoir plusieurs scores

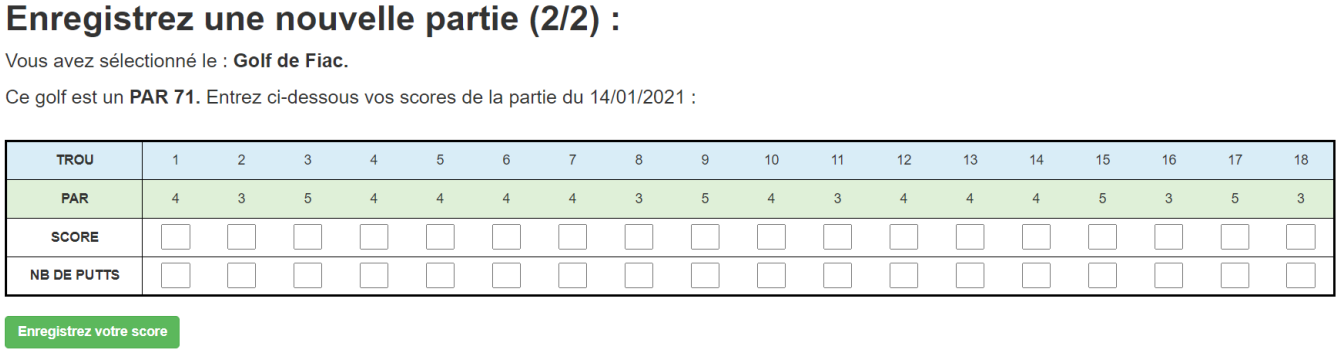
Il s'agit donc d'une relation 1 à plusieurs, où la table Score reçoit en clé étrangère golfCourse. Est venu ensuite le développement de l'ensemble des fonctionnalités.

1- Enregistrer une carte de score :

Pour cette fonctionnalité, j'ai eu besoin de créer un formulaire permettant d'enregistrer le score. Une fois la sélection du parcours de Golf par l'utilisateur définie, je récupère les données du golf et les affichent dynamiquement en passant en variable au template grâce à la méthode render_template() les données et ensuite en les affichant via un système de tag comme on peut le voir sur l'exemple ci-dessous. Il s'agit du principe de Data Binding qui permet d'associer des données d'un élément à un autre :

```
<td>{{ golf_user_choice.trou_1 }}</td>
```

Je récupère ensuite au niveau de la VUE les données du formulaire grâce à la méthode http POST et les enregistrent ensuite en base de données par l'intermédiaire d'une commande de l'ORM SQLAlchemy.



2- Visualiser le score d'une partie :

La deuxième fonctionnalité doit permettre de visualiser les données de la carte de score enregistrée et les statistiques qui vont avec. Pour ce faire, il suffit de récupérer l'ID du score passé en paramètre d'URL :

```
@app.route('/score/<id>')
```

Puis de récupérer les données de ce score grâce à la méthode SQLAlchemy qui permet de poster une requête SQL *SELECT * WHERE id= « la valeur du paramètre contenue dans l'URL » :*

```
score_to_show = db.session.query(score).get(id)
```

Et de retourner les valeurs contenues dans la variable score_to_show avec le bon template :

Détail du score de votre partie :

Résultat de la partie du 14/01/2021 au **Golf de Fiac**.

TROU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total
PAR	4	3	5	4	4	4	4	3	5	4	3	4	4	4	5	3	5	3	71
SCORES	4	3	4	6	5	5	5	4	7	4	3	5	4	3	5	3	7	3	80
PUTTS	2	1	1	2	2	2	1	2	2	2	2	2	2	1	1	2	2	2	31

Statistiques principales de la partie :

Score par rapport au PAR	Nombre de birdies	Nombre de par	Nombre de bogey	Nombre double bogey & +	Nombre greens régulations	Nombre d'approches/putt	Nombre de trois putts
9	2	8	5	3	8	5	0

Comme on peut le voir sur la visualisation de la page ci-dessus, j'ai « habillé » les données pour les rendre plus compréhensible notamment sur le type de score réalisé (Birdie, Par, Bogey, Double Bogey…). Pour y parvenir j'ai utilisé du code **JAVASCRIPT** permettant de gérer les différentes conditions :

```
for (let i = 0 ; i < 18 ; i++) {
    var strTrou = « trou_ »+i
    var strScore = « score_ »+i
    var trou = parseInt(document.getElementById(strTrou).innerText)
    var score = parseInt( document.getElementById(strScore).innerText)

    if (trou >score){
        document.getElementById(strScore).style.backgroundColor = « #A90000 » ;
        document.getElementById(strScore).style.color = « white » ;
    }
    else if (trou <score  && score <(trou+2) ){
        document.getElementById(strScore).style.backgroundColor = « #1361c0 » ;
        document.getElementById(strScore).style.color = « white » ;
    }
    else if (score >(trou +1)){
        document.getElementById(strScore).style.backgroundColor = « #0c3c78 » ;
        document.getElementById(strScore).style.color = « white » ;
    }
}
```

3- Visualiser l'ensemble des scores avec statistiques associées :

La dernière fonctionnalité doit permettre d'afficher l'ensemble des scores, de présenter des calculs de moyenne, et de voir ou supprimer un score. Pour y parvenir j'ai utilisé plusieurs méthodes de l'ORM SQLAlchemy permettant de poster différentes commandes SQL : SELECT * / SELECT * FROM SCORE WHERE ID=.. / DELETE FROM SCORE WHERE ID=..

Les calculs des moyennes sont réalisés via du code JAVASCRIPT. Ci-dessous la visualisation de ces éléments :

Ensemble des scores :

Retrouvez ci-dessous les statistiques principales de vos dernières parties :

Moyenne de l'ensemble des parties			12.4	6.2	0.9	7.3	7.0	2.8	32.5	4.8	1.4		
Moyenne des dix dernières parties			12.4	6.1	0.8	7.4	7.2	2.6	32.5	4.7	1.2		
Date	Lieu golf	Score brut	Score par rapport au par	Nb Greens régul	Nombre de Birdies	Nombre de Par	Nombre de Bogey	Nombre double Bogey et +	Nombre de putts	Nombre de 1 putt	Nombre de 3 putts	Détail du score	Supprimer partie
14/01/2021	Fiac	84	13	7	0	10	5	3	33	3	0	voir score	supprimer
14/01/2021	Fiac	82	11	5	1	7	8	2	29	8	1	voir score	supprimer
14/01/2021	Fiac	85	14	5	1	4	12	1	32	6	2	voir score	supprimer
14/01/2021	Fiac	85	14	5	1	5	9	3	34	4	2	voir score	supprimer
14/01/2021	Lou Verdaï	87	17	4	0	8	6	4	33	4	1	voir score	supprimer
14/01/2021	Fiac	81	10	9	1	8	7	2	36	2	2	voir score	supprimer
14/01/2021	La Ramée	84	14	4	0	8	7	3	32	5	1	voir score	supprimer
14/01/2021	Fiac	85	14	5	2	5	7	4	31	7	2	voir score	supprimer
14/01/2021	Fiac	80	9	8	2	8	5	3	31	5	0	voir score	supprimer
14/01/2021	Fiac	79	8	9	0	11	6	1	34	3	1	voir score	supprimer
14/01/2021	Seilh Rouge	88	16	7	1	6	7	4	36	5	5	voir score	supprimer
14/01/2021	Fiac	80	9	6	2	8	5	3	29	5	0	voir score	supprimer

En conclusion, ce projet personnel m'a permis à la fois de me former à une technologie que je ne connaissais pas encore mais également de revoir quasiment tous les grands principes de création d'une application WEB vus pendant la formation.

2. Précisez les moyens utilisés :

Pour réaliser ce projet, j'ai utilisé l'éditeur de code PyCharm. Ce projet ayant vocation à me former tout en « m'amusant » sur un sujet concret qui me passionne, je l'ai développé, et installé en local sur mon PC.

3. Avec qui avez-vous travaillé ?

Ce projet étant un projet personnel visant à développer mes compétences en Python et sur le Framework Flask, je l'ai réalisé tout seul sur mon temps libre, notamment pendant le second confinement.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *Projet Personnel*

Chantier, atelier, service ▸ *Cliquez ici pour taper du texte.*

Période d'exercice ▸ Du : *01/11/2020* au : *30/11/2020*

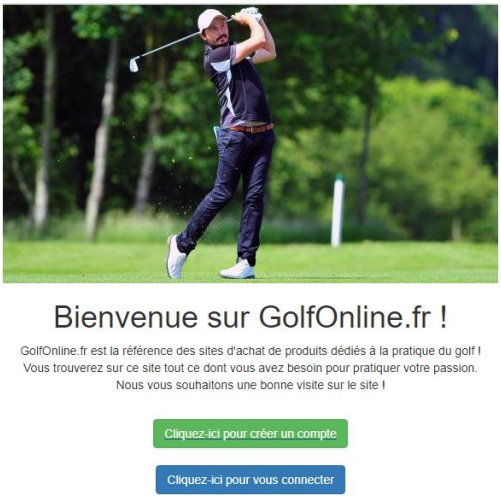
5. Informations complémentaires *(facultatif)*

Activité-type 1 : DEVELOPPER DES COMPOSANTS D'INTERFACE

Exemple n°2 : Projet Braderie : Front avec JSP, Back avec Spring Boot

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

Dans le cadre de la formation AFPA et du module de formation Développement d'application WEB, nous avons eu à mettre en application les compétences acquises sur un projet de construction d'une Braderie. L'idée étant de développer une application WEB représentant un site e-commerce avec la construction d'un panier, la possibilité d'ajouter des quantités ou de supprimer un article et enfin de le sauvegarder. J'ai choisi la thématique d'un site e-commerce de Golf.



L'environnement technique choisi était le suivant : Développement du Front avec la technologie Java Server Page (JSP), développement du Back en Java avec le Framework Spring.

1- Développement du FRONT-END en JSP :

La technologie JSP fait partie de la plateforme JAVA Entreprise Edition (JAVA EE) lancée pour sa première version en 1999. JSP permet de créer des pages web dynamiques en intégrant du code JAVA directement dans la page HTML grâce à un système de TAG. Elle permet de ce fait de faire passer des données de la partie BACK au FRONT.

Pour le développement du FRONT, j'ai utilisé HTML/CSS et le Framework CSS BootStrap.

Le principe des JSP est de faire passer des données entre le back et le front à partir de la couche Controller de l'application. Pour y parvenir j'ai utilisé l'interface Model. Grâce à cette interface il est possible de lier les données entre le back et le front. Exemple avec la liste des produits de cette braderie, au niveau du back :

Récupération de la liste produits en BDD en passant par les différentes couches
Ajout de la liste produits au model grâce à méthode addAttribute() et d'un identifiant
Renvoi de la page JSP adéquat

```
/*
 * Méthode qui permet d'afficher les produits sur la page listProducts:
 */
@GetMapping("/listproducts")
public String showAllProducts(Model model, HttpSession session) {
    User u = (User) session.getAttribute("userLogged");

    model.addAttribute("firstname", u.getNom());
    Collection<Produit> products = (Collection<Produit>) serviceProduct.getProducts();

    model.addAttribute("products", products);

    return "listProducts";
}
```

Puis on récupère les données au niveau de la page JSP grâce à l'identifiant (ici « products ») :

```
<div>
    <table border="1" class="table table-dark" style="margin-left: 10px;">
        <tr>
            <th class="bg-primary" scope="col">n° article</th>
            <th class="bg-primary" scope="col">Nom article</th>
            <th class="bg-primary" scope="col">Description</th>
            <th class="bg-primary" scope="col">Price</th>
            <th class="bg-primary" scope="col">Panier</th>
        </tr>
        <c:forEach items="${products}" var="products">
            <tr>

                <td>${products.unite}</td>
                <td>${products.name}</td>
                <td>${products.description}</td>
                <td>${products.price}E</td>
                <td class="btn btn-success"><a style="color: white;"
                    href="addpanier?id=<c:out value="${products.unite}"/>"> <c:out
                            value="Ajouter au panier" />
                </a></td>
            </tr>
        </c:forEach>
    </table>
</div>
```

Récupération des données de la liste grâce à l'identifiant

Possibilité d'afficher les attributs des produits de la liste

Ce qui donne ensuite cela au niveau visuel :

Me déconnecter du site

Nos produits

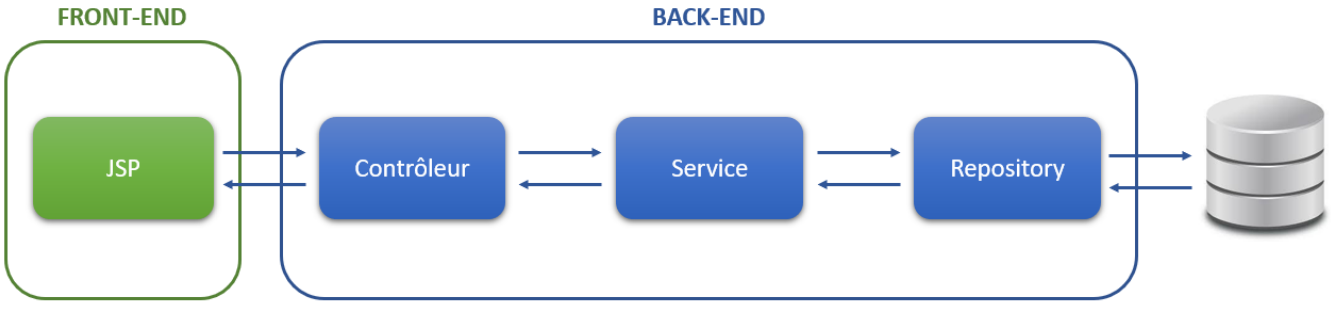
Bienvenue Guillaume ! Voici la liste des produits disponibles à la vente :

n° article	Nom article	Description	Price	Panier
21	Balle	Balle spécialement conçue pour jouer au golf. Balle alliant toucher doux et longue distance !	1E	Ajouter au panier
22	Fer de golf	Club spécialement conçu pour taper la balle à une distance précise en direction du green	99E	Ajouter au panier
23	Driver	Club spécialement conçu pour taper la balle du départ à une longue distance	390E	Ajouter au panier
24	Putter	Club spécialement conçu pour faire rouler la balle sur le green en direction du trou	290E	Ajouter au panier
25	Sac de golf	Sac de golf permettant de transporter tous les clubs ainsi que les balles et le matériel nécessaire à la pratique du golf	190E	Ajouter au panier

Visualiser mon panier | Visualiser mon panier sauvegardé

2- Développement du BACK-END en JAVA et Framework SPRING :

Pour le développement du Back-end, j'ai choisi d'appliquer un développement en couche et notamment d'adopter une architecture de type MVC (Modèle-Vue-Contrôleur => Cf exemple Activité 3). Au niveau du back pour bien séparer les responsabilités je sépare en 3 couches :



La couche **CONTRÔLEUR** gère la logique de l'application, récupère les requêtes de l'utilisateur et transmet la demande à la couche service. Dans l'autre sens, elle récupère les données de la part de la couche service, les transmet à la page JSP et se charge de transmettre la réponse à la requête de l'utilisateur en y associant la bonne page JSP.

La couche **SERVICE** fait l'interface entre la couche contrôleur et Repository. Elle gère la partie « business » de l'application et notamment tous les traitements à opérer sur les données avant envoi dans un sens ou l'autre (calculs par exemple).

La couche **REPOSITORY** se charge d'envoyer les données en base de données. Elle opère donc les requêtes SQL. Dans le cadre de cette application j'utilise l'interface CRUDRepository qui simplifie grandement le travail en utilisant des méthodes qui génèrent automatiquement les bonnes requêtes SQL.

On peut prendre l'exemple de la sauvegarde d'un panier en BDD.

Au niveau de la couche CONTRÔLEUR, on récupère les produits et l'utilisateur reliés à la session, puis on les transmet à la couche SERVICE :

```
@GetMapping("/save")
public String saveCart(HttpSession session) {

    List<Produit> myProductList = (List<Produit>) session.getAttribute("myProductsSelected");
    User UserLogged = (User) session.getAttribute("userLogged");

    // Appel de la méthode de mon service Cart qui permet d'ajouter en base mon
    // panier à partir du user loggé et de la liste de produits créés
    serviceCart.addCartInDB(myProductList, UserLogged);
    loger.info("L'utilisateur "+UserLogged.getNom()+" vient d'enregistrer un panier d'achat.");

    return "cartCreated";
}
```

Au niveau de la couche SERVICE, on crée un objet de type CART contenant l'ensemble des produits et l'utilisateur afin de pouvoir ensuite l'envoyer à la couche REPOSITORY. Comme expliqué précédemment, il existe des méthodes via l'instance CRUDRepository qui génère automatiquement une requête SQL. Dans ce cas précis la méthode .save() permet de créer une requête INSERT :

```
/*
 * Méthode qui permet d'ajouter le panier en créant un nouvel objet Cart qui est
 * sauvegardé en BDD:
 */
public void addCartInDB(List<Produit> myProductList, User UserLogged) {

    for (Produit p : myProductList) {

        Cart detail = new Cart();

        detail.setUser(UserLogged);
        detail.setQuantity(p.getQuantity());
        detail.setProduct(p);

        cartRepo.save(detail);

    }
}
```

Voici la visualisation d'un panier et de la possibilité de sauvegarde :

Votre panier :

N° Article	Nom	Description	Prix	Quantités	Ajouter quantités	Supprimer article
22	Fer de golf	Club spécialement conçu pour taper la balle à une distance précise en direction du green	99 Euros	1	ajouter 1 quantité	supprimer cet article du panier
21	Balle	Balle spécialement conçue pour jouer au golf. Balle alliant toucher doux et longue distance !	1 Euros	7	ajouter 1 quantité	supprimer cet article du panier

Coût total de votre panier :

106 Euros

Revenir à la liste des produits | Supprimer tous les articles du panier | Sauvegarder mon panier

Ce cas de la braderie m'aura permis de mieux appréhender le fonctionnement d'une application en couche ainsi que les différentes relations entre FRONT – BACK – BDD.

2. Précisez les moyens utilisés :

Pour ce projet réalisé pendant le 1er confinement, j'ai utilisé des moyens personnels (PC). Pour le développement j'ai utilisé l'IDE Eclipse.

3. Avec qui avez-vous travaillé ?

Ce projet de formation était un travail personnel. Il m'est arrivé cependant de demander de l'aide à des collègues de formation sur un point bloquant voir à notre formateur.

4. Contexte

Nom de l'entreprise, organisme ou association **:**
Projet de formation AFPA

Chantier, atelier, service :

Période d'exercice ▸ Du : *01/04/2020* au : *30/04/2020*

Activité-type 1 : DEVELOPPER DES COMPOSANTS D'INTERFACE

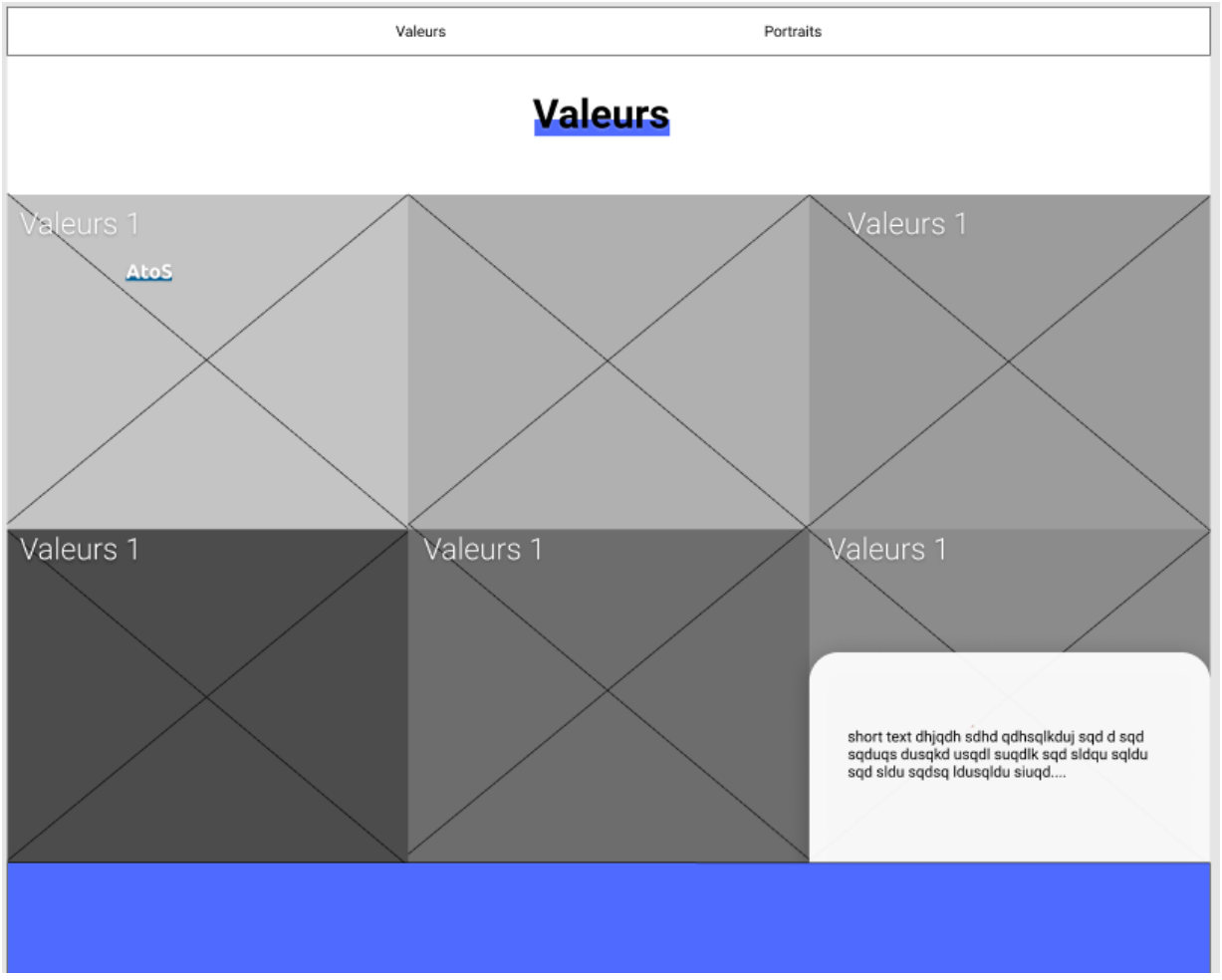
Exemple n°3 : Maquettage projet Culture Book

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

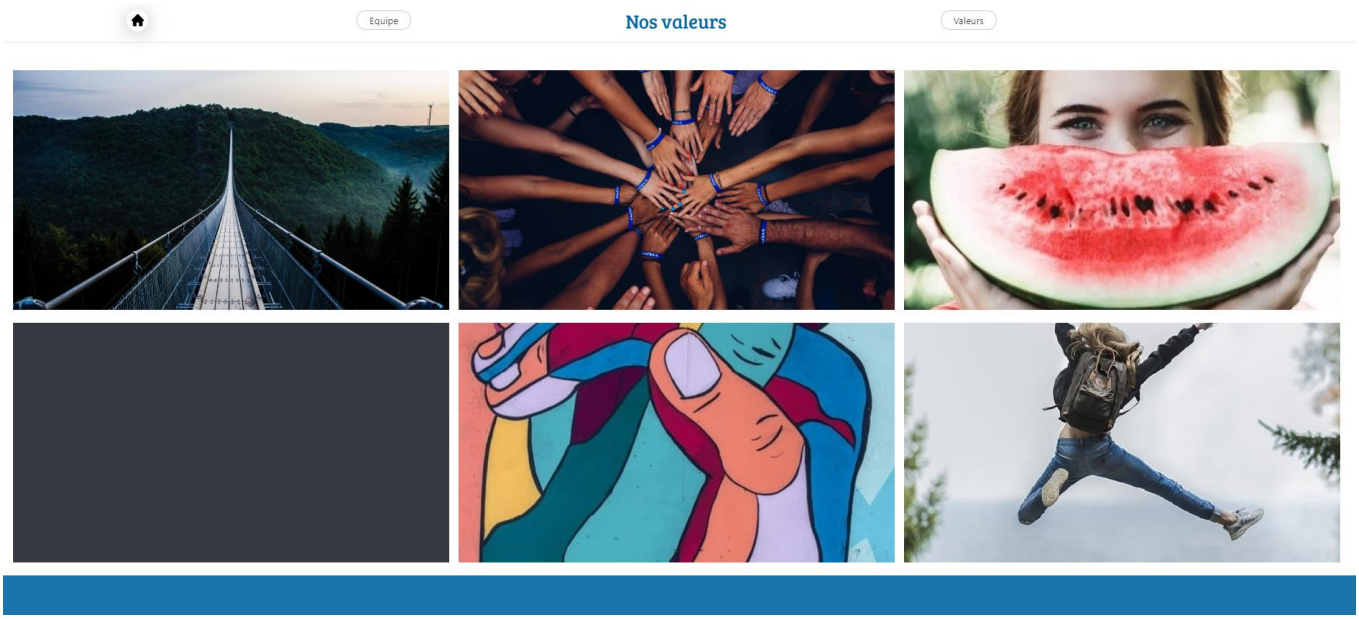
Dans le cadre de la formation, nous avons réalisé un culture book. Son objectif était de s'approprier la culture d'entreprise d'Atos et ses valeurs tout en réalisant collectivement un projet de développement d'application web.

Nous avons travaillé en mode Agile et nous sommes organisés par équipe de 2 ou 3. Dans les étapes en amont nous avons réalisé des maquettes de l'application pour valider tous ensemble l'orientation de design du site. Pour réaliser ces maquettes nous avons utilisé l'outil en ligne FIGMA.

Voici par exemple la maquette de la page d'accueil des valeurs :



Et voici le résultat final de cette page sur notre site en production :



La réalisation de maquette est donc très importante dans la phase de préparation/conception du projet car elle permet de donner une ligne directrice commune à l'ensemble des équipes d'un projet.

2. Précisez les moyens utilisés :

Logiciel en ligne FIGMA.

3. Avec qui avez-vous travaillé ?

Ce projet de formation était un travail de l'ensemble du groupe de formation AFPA.

4. Contexte

Nom de l'entreprise, organisme ou association **:**
Projet de formation AFPA

Chantier, atelier, service :

Période d'exercice ▸ Du : *01/02/2020* au : *10/02/2020*

Activité-type 1 : DEVELOPPER DES COMPOSANTS D'INTERFACE

Exemple n°4 : Projet Monnayeur, application de type Desktop

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

Dans le cadre de la formation à l'AFPA et de la découverte du langage JAVA, nous avons commencé par découvrir l'environnement JSE (Java Standard Edition). Cette plate-forme permet de construire des applications de type Desktop.

Pour découvrir à la fois le langage JAVA et nous perfectionner en algorithmie nous avons travaillé sur un projet de développement d'un Monnayeur. Le but étant de proposer différents choix de boissons et de rendre la monnaie.

Pour réaliser la partie IHM, nous avons utilisé la technologie SWING. Swing est une API qui permet de réaliser une interface graphique de type desktop avec du code JAVA.

La construction de l'IHM se fait à partir de classes spécifiques qui vont représenter des éléments graphiques. Exemple la classe JList permet de définir des listes non déroulantes ou encore la classe JButton qui permet de représenter un bouton.

2. Précisez les moyens utilisés :

Pour ce projet réalisé à l'AFPA avant le 1er confinement. Nous disposions d'un PC avec deux écrans. Nous avons utilisé l'IDE Eclipse pour développer ce projet.

3. Avec qui avez-vous travaillé ?

Ce projet de formation était un travail personnel. Il m'est arrivé cependant de demander de l'aide à des collègues de formation sur un point bloquant voir à notre formateur.

4. Contexte

Nom de l'entreprise, organisme ou association **:**
Projet de formation AFPA

Chantier, atelier, service :

Période d'exercice ▸ Du : *15/02/2020* au : *28/02/2020*

Activité-type 2 DEVELOPPER LA PERSISTANCE DES DONNEES

Exemple n° 1 ▸ ***Conception et création base de données projet réseau social***

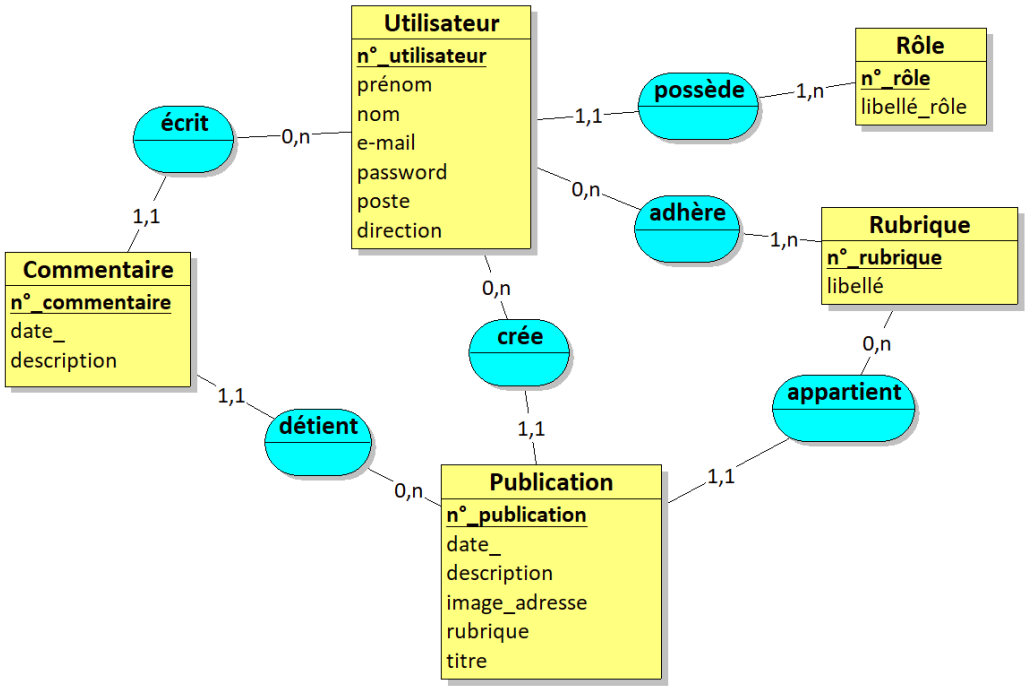
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Dans le cadre de la formation nous avons eu à réaliser un projet de groupe nous permettant de mettre en application l'ensemble des compétences acquises durant la formation. Ce projet a duré 1 mois et a été réalisé par groupe de 4. Dans notre groupe nous avons décidé de réaliser une application de type réseau social d'entreprise.

Après les étapes d'organisation, de définition de l'environnement de travail, de définition des besoins fonctionnels (cf. exemple 1 activité 3), nous avons eu à réaliser la conception et l'implémentation des données.

En partant de la réflexion menée sur les cas d'utilisation et leurs définitions dans le diagramme des « use case UML » (cf. exemple 1 activité 3), nous avons réfléchi à la conception et notamment à la structuration de notre base de données. Il est apparu rapidement évident que notre application se construirait autour de deux entités importantes : l'utilisateur et la publication. En effet, pour utiliser un réseau social il est obligatoire de posséder un compte utilisateur et enfin le but de notre réseau social était de pouvoir créer des publications.

Nous avons donc construit un modèle conceptuel des données (MCD) issu de la méthode Merise qui permet de représenter sur un schéma la structure des données et les relations entre elles qu'on appelle association. Pour caractériser ces associations on utilise les cardinalités qui sont des valeurs min et max qui expriment le nombre de fois qu'un élément d'une entité peut être engagé dans un élément de l'association, au minimum et au maximum. Il existe 3 valeurs possible 0,1,N (N étant l'infini). L'association est nommée par un verbe d'action qui symbolise le lien entre les deux entités. Voici le schéma MCD que nous avons construit :



Ainsi à titre d'exemple on peut expliquer l'association « possède » entre l'entité **Utilisateur** et **Rôle** de la manière suivante :

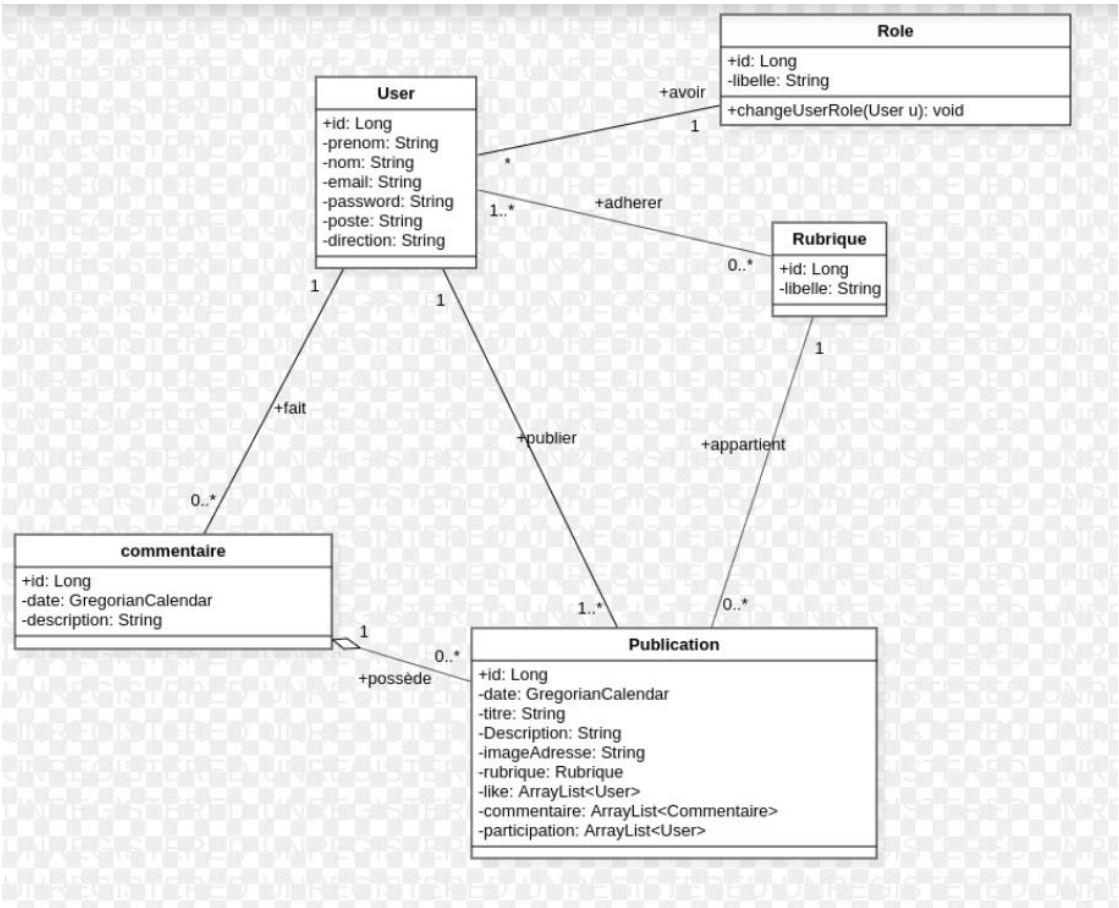
- Un utilisateur possède forcément un rôle (admin ou user) donc min = 1 et ne peut posséder qu'un rôle maximum donc max = 1
- Un rôle ne peut exister que si au moins un utilisateur est créé donc min = 1 et peut être attribué à une infinité d'utilisateurs donc max = N

Autre exemple avec les entités principales **Utilisateurs** et **Publication** et l'association « crée » :

- Un utilisateur peut décider de ne pas créer de publications (seulement lecteur) donc min = 0 mais peut par contre créer autant de publications qu'il le souhaite donc max = N
- Une publication ne peut exister que si au moins 1 utilisateur en créer une donc min = 1 et elle est forcément rattachée qu'à un seul utilisateur qui en est son propriétaire donc max = 1

À la suite de la création du MCD, nous avons décidé de réaliser un diagramme de classe UML. Un diagramme de classe permet contrairement au MCD d'entrer un peu plus dans la conception de l'application car il sert à représenter les classes, leurs attributs avec les types correspondants, les relations entre les classes (qui sont similaires à ce qu'on avait dans le MCD) et les méthodes de classe.

Les cardinalités sont aussi indiquées mais il faut les lire à l'inverse du MCD. N est également remplacé par « ..* ».



Pour une meilleure lisibilité nous avons volontairement enlevé de ce schéma les méthodes de classe pour les mettre à part.

A partir de là, nous avons la structure de nos données et de nos classes qui est parfaitement définie. On peut donc passer à l'étape de création du projet et de la base de données.

Le projet a été conçu au niveau Back-end en JAVA en utilisant le Framework SPRING BOOT et le gestionnaire de paquets MAVEN. Dans le cadre de ce projet nous avons décidé d'utiliser un Object Relational Mapping (ORM) qui s'appelle HIBERNATE et qui s'appuie sur l'API Java Persistence (JPA) pour la persistance de nos données. La base de données est une ORACLE de type relationnelle (SGBD).

Un ORM permet de s'appuyer sur la structure des classes et de ses attributs pour créer la base de données (d'où l'importance d'un diagramme de classe cohérent avec la structure des données). L'ORM permet de créer « un pont » entre le monde objet et le monde des données relationnelles.

Le principe d'Hibernate/JPA est d'utiliser des annotations avec le caractère @. Voici quelques-unes des principales annotations et leurs définitions :

- @Entity = définit la classe comme étant une entité persistante
- @Table = définit la table dans la BDD qui représente cette entité. On peut rajouter (name= « . ») qui sera le nom que prendre la table dans le BDD
- @Id = définit le fait que l'attribut qui se situe en dessous sera la clé primaire dans la table
- @Column = définit le fait que l'attribut situé en dessous sera une colonne dans la table

Voici par exemple la structure de la classe Rubrique avec ces annotations :

```
@Entity
@Table(name = "T_RUBRIQUE")
public class Rubrique {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column
    private Long id;

    @Column
    private String libelleRubrique;

    @Column
    private String typeRubrique;
```

Avec @GeneratedValue, on définit la stratégie de génération des clés primaires. Ici avec Auto, on laisse à Hibernate le choix de créer la séquence à l'implémentation de la BDD.

Enfin avec les annotations on peut gérer les différents cas de figure de relations entre les entités comme décrites dans les schémas précédents.

Il existe 3 possibilités d'annotations pour caractériser ces relations :

1. **@OneToOne :**

Une relation @OneToOne caractérise une relation de type 1-1. Ainsi si on prend l'exemple de la classe Commentaire, on met l'annotation @OneToOne au-dessus de l'attribut user car 1 commentaire ne peut être rattaché qu'à un seul user :

```
@OneToOne
private User user;
```

2. @OneToMany :

Une relation @OneToMany caractérise une relation de type 1-N. Cette annotation est donc inscrite dans l'entité du côté 1, au-dessus de l'attribut reflétant l'autre entité. Exemple avec la classe publication qui contient l'attribut comments. Une publication peut contenir à l'infini de commentaires donc on indique que c'est une relation @OneToMany :

```
@OneToMany
@JsonIdentityReference(alwaysAsId = true)
private Set<Commentaire> comments;
```

3. @ManyToOne :

Dans le cas de la relation @ManyToOne, cela définit une relation de type N-1. C'est quand l'entité dans laquelle on applique cette annotation peut se retrouver une infinité de fois dans la relation avec l'autre entité. Ainsi dans l'entité publication, un nombre infini de publications peuvent être crées par un utilisateur comme on peut le voir sur l'exemple ci-dessous avec l'attribut User poster :

```
@ManyToOne
private User poster;
```

4. @ManyToMany :

Enfin la dernière possibilité, c'est l'annotation @ManyToMany. Cela caractérise une relation de type N-N. C'est quand les possibilités sont infinies dans les deux entités. Ainsi dans l'exemple de la classe User, un utilisateur peut s'abonner à un nombre infini de rubriques et dans l'entité rubriques, une rubrique peut contenir un nombre infini d'utilisateurs :

```
@ManyToMany
@Column
private Set<Rubrique> rubriques;
```

Une fois l'ensemble de ces annotations positionnées sur les différentes classes, l'ORM Hibernate va s'appuyer dessus pour générer toutes les requêtes SQL permettant de créer les tables, colonnes, IDs, et relations entre les tables par le positionnement des clés étrangères ou une table d'association.

Nous avions créé un script permettant à chaque lancement de l'application d'effacer puis de créer les tables et d'ajouter des données spécifiques permettant de faire tourner l'application (rubriques, utilisateurs).

Ce projet fut très formateur et concret pour apprendre à concevoir et implémenter la persistance des données. Cette étape est cruciale dans la réussite d'un projet de développement d'application et doit donc être abordée de manière méticuleuse.

2. Précisez les moyens utilisés :

Réalisé pendant le second confinement, nous avons utilisé nos PC personnels. Nous avons installé dessus l'ensemble des logiciels indispensable comme l'IDE Eclipse, la base de données ORACLE, SQL DEVELOPER, un outil collaboratif qui était DISCORD et un outil de suivi de projet TRELLO.

3. Avec qui avez-vous travaillé ?

Pour ce projet, j'ai travaillé dans une équipe de 4 personnes. Il s'agissait de collègues de la formation AFPA.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *AFPA*

Chantier, atelier, service ▸ *Cliquez ici pour taper du texte.*

Période d'exercice ▸ Du : *01/05/2020* au : *01/06/2020*

5. Informations complémentaires *(facultatif)*

Activité-type 3 DEVELOPPER UNE APPLICATION N-TIERS

Exemple n° 1 ▸ ***Création d'une application réseau social d'entreprise***

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

Dans le cadre de la formation à l'AFPA, nous avons réalisé un projet de groupe d'une durée de 1 mois par groupe de 4 sur un sujet précis. Nous devions définir une organisation, une méthode de travail, définir ce que chacun allait faire…

Dans un premier temps, nous avons décidé de proposer un sujet sur lequel nous avions réfléchi. En effet, une liste de sujets nous a été proposé mais nous souhaitions apporter notre idée. Nous avons donc proposé de créer un réseau social d'entreprise.

La première étape a été de se mettre à la place d'un client (une entreprise) et d'imaginer les besoins fonctionnels qu'elle pourrait avoir :

1. **Définition du cahier des charges fonctionnels :**

Etant donné que le sujet a été défini par notre groupe nous avons dû proposer un cahier des charges fonctionnels à notre formateur afin qu'il le valide. Ce cahier des charges permet de définir plusieurs choses :

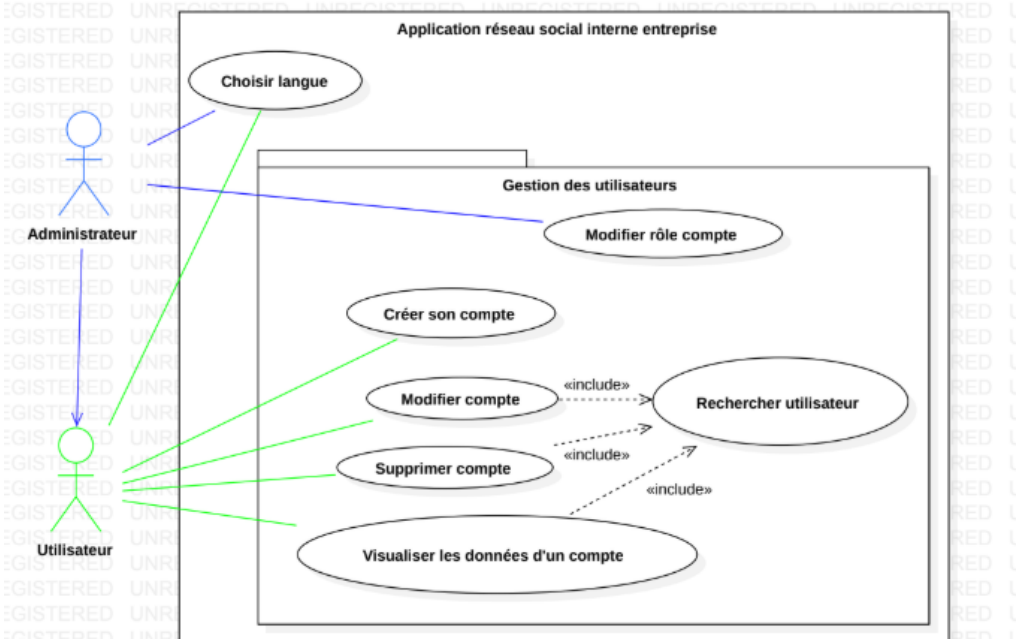
- Décrire le périmètre de l'application
- Lister les principales fonctionnalités
- Décrire les différents rôles d'utilisateurs de l'application
- Définir les besoins et les classer selon trois niveaux (A / B / C) :
    - Niveau A = Ce niveau définit les fonctionnalités primordiales qui doivent être absolument incluses dans l'application pour qu'elle soit opérationnelle.
    - Niveau B = Ce niveau définit les fonctionnalités importantes qui doivent être implémentées dans les futures versions de l'application.
    - Niveau C = Ce niveau définit des fonctionnalités utiles qui peuvent être incluses dans les versions futures de l'application. Ces fonctionnalités ne sont pas indispensables mais elles apportent un niveau de confort d'utilisation supérieur.

Ceci a abouti à la réalisation d'un document Excel recensant l'ensemble des besoins selon différentes rubriques et les niveaux de priorités.

2. **Réalisation diagramme de USE CASE UML :**

La deuxième étape de ce projet a été de réaliser un diagramme de USE CASE UML à partir du cahier des charges fonctionnels rédigés précédemment. Un diagramme de USE CASE UML a pour objectif de présenter l'ensemble des possibilités d'interaction entre l'application logicielle et les utilisateurs. Il vise à décrire l'ensemble des actions possibles sous forme de verbe d'action. Il peut créer une arborescence ou hiérarchie entre les cas d'usage comme par exemple le fait qu'un cas d'usage inclus le fait de réaliser un autre cas d'usage.

Voici un exemple de notre USE CASE UML pour la rubrique Gestion des utilisateurs avec les deux rôles administrateur et utilisateur :



3. Organisation du projet :

À la suite de ces étapes, nous avons pu nous pencher sur la définition de l'organisation de notre projet. En effet nous étions libres de nous organiser comme nous le souhaitions. La seule contrainte qui nous était imposée était de gérer notre projet en mode AGILE.

L'Agilité est un mode de gestion de projet apparu dans les années 90 qui vise à apporter de la souplesse dans la gestion d'un projet informatique. Il vient à contre-courant du modèle dominant qui était la gestion de projet en cycle en V. Un manifeste a été rédigé pour décrire ce qu'est l'agilité. Il se compose de 4 valeurs et de 12 principes.

L'idée centrale est de se rapprocher au maximum des besoins du clients en fonctionnant par des itérations courtes et surtout en mettant le client au centre du processus de développement. Il vise aussi à s'alléger de la rédaction de documentation et met le dialogue et la coopération au centre.

Dans le cadre de notre projet en mode agile, nous avons défini l'organisation suivante :

- 1 personne prend le rôle de Product Owner par sprint
- 1 personne par semaine prend le rôle de Scrum master
- 4 personnes prennent le rôle de développeur
- Le projet durant 1 mois nous avons décidé de réaliser 2 sprints de 2 semaines
- Un daily meeting avait lieu tous les jours permettant de faire le point sur l'avancement des uns et des autres.
- Le suivi du projet et notamment du Sprint Backlog (contenant les user stories du sprint) a été réalisé grâce à l'outil en ligne TRELLO avec 4 colonnes : BACKLOG PRODUCT / BACKLOG SPRINT / EN COURS / TERMINE SPRINT.

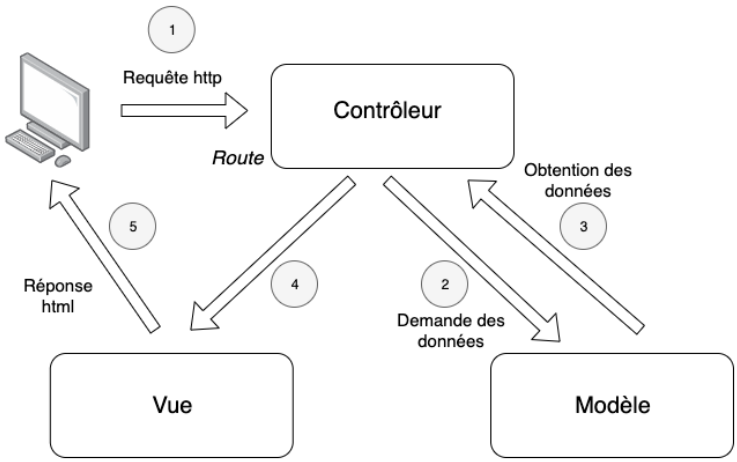
4. Choix techniques :

Pour réaliser ce projet nous avons fait plusieurs choix techniques :

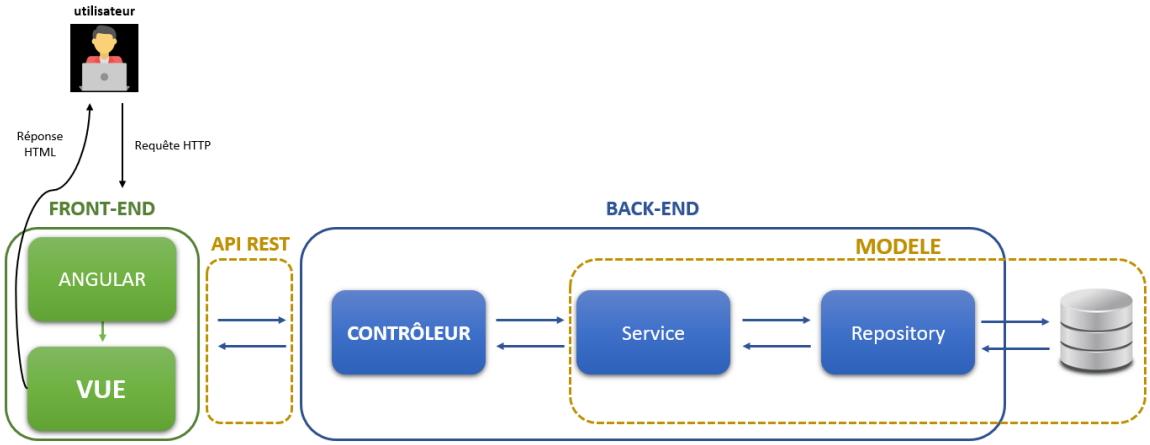
- La partie Front-end a été réalisé avec ANGULAR
- La partie Back-end a été réalisé avec JAVA et le Framework SPRING BOOT.
- Nous avons construit des API (Application Programing Interface) REST pour exposer nos données du BACK afin de séparer le client (interface web de l'utilisateur) du serveur. Ainsi les deux systèmes Front et back sont autonomes ce qui permettrait par exemple de développer une application mobile dans un second temps sans avoir besoin de redévelopper la partie Back-end. Elle consommerait juste l'API à partir des requêtes HTTP (GET, POST…).
- L'utilisation d'une base de données relationnelle (SGBD) ORACLE
- Utilisation de l'ORM Hibernate pour faciliter la création de la base de données
- Développement en couche et selon l'architecture Modèle-Vue-Contrôleur

5. Architecture du projet :

Nous avons donc décidé de suivre le modèle MVC. Le modèle MVC est une architecture logicielle crée en 1978 et souvent utilisé pour les applications web. Son fonctionnement est décrit dans le schéma ci-dessous.



Dans le cadre de notre projet, le fonctionnement est légèrement différent et peut être présenté de la manière suivante :



Ce projet nous a permis de se confronter à la réalité d'un projet de développement d'application web. Il nous a permis de voir à quel point la préparation et l'organisation avant même les phases de développement étaient fondamentales dans la réussite d'un projet. Il nous a permis d'expérimenter un projet en mode AGILE et d'apprendre à communiquer et travailler en équipe.

2. Précisez les moyens utilisés :

Réalisé pendant le second confinement, nous avons utilisé nos PC personnels. Nous avons installé dessus l'ensemble des logiciels indispensable comme l'IDE Eclipse, la base de données ORACLE, SQL DEVELOPER, un outil collaboratif qui était DISCORD et un outil de suivi de projet TRELLO.

3. Avec qui avez-vous travaillé ?

Pour ce projet, j'ai travaillé dans une équipe de 4 personnes. Il s'agissait de collègues de la formation AFPA.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *AFPA*

Chantier, atelier, service ▸ *Cliquez ici pour taper du texte.*

Période d'exercice ▸ Du : *01/05/2020* au : *01/06/2020*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] Guillaume Lenoir d'Espinasse ,

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 Nailloux le 20/01/2021

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

(Si le RC le prévoit)