



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance

- ▶ Gendrau

Nom d'usage

- ▶ Gendrau

Prénom

- ▶ Jean-Ely

Adresse

- ▶ 267 Quai Marmora Résidence Côté OUEST 83200 Toulon

Titre professionnel visé

Concepteur Développeur d'Applications (CDA) Session 2024-2025

MODALITÉ D'ACCÈS :

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

Présentation du dossier

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

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

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

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

Pour prendre sa décision, le jury dispose :

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

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

Ce dossier comporte :

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

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

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

DOSSIER PROFESSIONNEL (DP)

Sommaire

Exemples de pratique professionnelle

Intitulé de l'activité-type n° 1 Développer une application sécurisée	p.
▶ Intitulé de l'exemple n° 1 Installer et configurer son environnement (Poseiden)	p. 6
▶ Intitulé de l'exemple n° 2 Développer des interfaces utilisateur (Poseiden Thymeleaf views)	p. 10
▶ Intitulé de l'exemple n° 3 Développer des composants métier (Poseiden - Services - DTO)	p. 14
▶ Intitulé de l'exemple n° 4 Contribuer à la gestion d'un projet informatique (GitHub – Kanban & issues)	p. 20
Intitulé de l'activité-type n° 2 Concevoir et développer une application sécurisée organisée en couches	p.
▶ Intitulé de l'exemple n° 1 Analyser les besoins et maquetter une application (FlashCash)	p. 23
▶ Intitulé de l'exemple n° 2 Définir l'architecture logicielle (Poseiden MVC, HealthCare(Medica) - Microservices)	p. 25
▶ Intitulé de l'exemple n° 3 Concevoir et mettre en place une base de données (Poseiden - SQL + application.properties)	p. 27
▶ Intitulé de l'exemple n° 3 Développer des composants d'accès aux données (Poseiden - Repositories JPA)	p. 31
Intitulé de l'activité-type n° 3 Préparer le déploiement d'une application sécurisée	p.
▶ Intitulé de l'exemple n° 1 Préparer et exécuter les plans de tests (SafetyNet - Tests unitaires + JaCoCo)	p. 35
▶ Intitulé de l'exemple n° 2 Préparer et documenter le déploiement (HealthCare - Docker Compose)	p. 37
▶ Intitulé de l'exemple n° 3 Contribuer à la mise en production (HealthCare - Config, Kafka, Monitoring)	p. 39

DOSSIER PROFESSIONNEL (DP)

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

p. _____

Déclaration sur l'honneur

p. _____

Documents illustrant la pratique professionnelle (*facultatif*)

p. _____

Annexes (*Si le RC le prévoit*)

p. _____

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1 Développer une application sécurisée

CP 1 ▶ Installer et configurer son environnement de travail en fonction du projet

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

Dans le cadre du projet "Poseiden Skeleton", j'ai installé et configuré l'environnement de développement nécessaire pour la mise en œuvre d'une application sécurisée basée sur Spring Boot.

J'ai utilisé **IntelliJ IDEA** comme IDE principal, intégré les dépendances via **Spring Initializr**, puis configuré le projet en ajoutant les répertoires nécessaires (domain, controllers, templates, etc.) ainsi que les fichiers de configuration *application.properties*.

J'ai également installé et configuré le **JDK 1.8**, mis en place le SDK dans l'IDE, et structuré le projet avec Maven en important les dépendances nécessaires (Spring Web, Thymeleaf, Spring Security...). Enfin, j'ai préparé une base de données locale "demo", et exécuté un script SQL pour initialiser les tables.

2. Précisez les moyens utilisés :

- IDE : IntelliJ IDEA
- Langage : Java 8
- Framework : Spring Boot 2.0.4
- Outils de build : Maven (pom.xml)
- Base de données : MySQL (base "demo")
- Gestion de version : Git + GitHub
- Script d'initialisation : data.sql
- Système d'exploitation : Windows 11 Pro

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie, dans un cadre pédagogique, avec un encadrement ponctuel par un formateur. La gestion de projet a été simulée avec l'usage d'un tableau Kanban et la création d'issues GitHub pour simuler une organisation agile.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme Formation

► *Projet de formation – GitHub Repository "Java-Tp-Poseiden-skeleton"*

Chantier, atelier, service

► *Projet individuel – Environnement full-stack Java*

Période d'exercice ► Du : **30/11/2024** au : **02/12/2024**

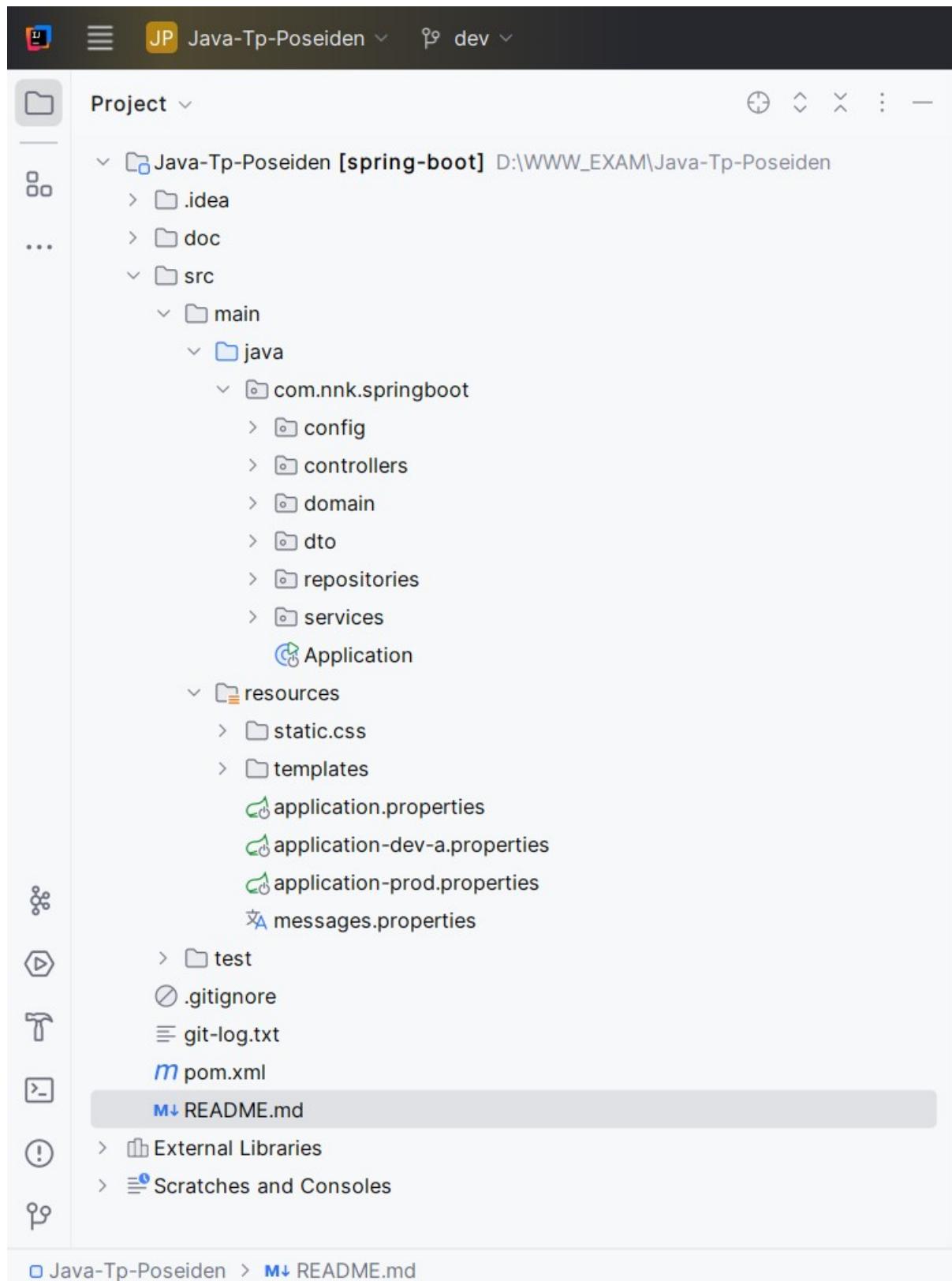
5. Informations complémentaires (facultatif) :

L'un des enjeux était de comprendre la structure d'un projet Spring Boot complet et de pouvoir le reconfigurer manuellement. Cela m'a permis de renforcer ma compréhension du fonctionnement des dépendances, de la configuration de base de données, et de l'organisation du code Java orienté MVC.

Extrait du fichier pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5    <modelVersion>4.0.0</modelVersion>
6
7    <groupId>com.nnk</groupId>
8    <artifactId>spring-boot</artifactId>
9    <version>0.0.1-SNAPSHOT</version>
10   <packaging>jar</packaging>
11
12   <name>spring-boot-skeleton</name>
13   <description>Java TP Poseiden Skeleton</description>
14
15  > <parent...>
16
17    <properties>
18      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19      <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
20      <java.version>1.8</java.version>
21    </properties>
22
23  >   <dependencies...>
24
25  >   <build...>
26
27
28  > </project>
```

Arborescence des packages du projet

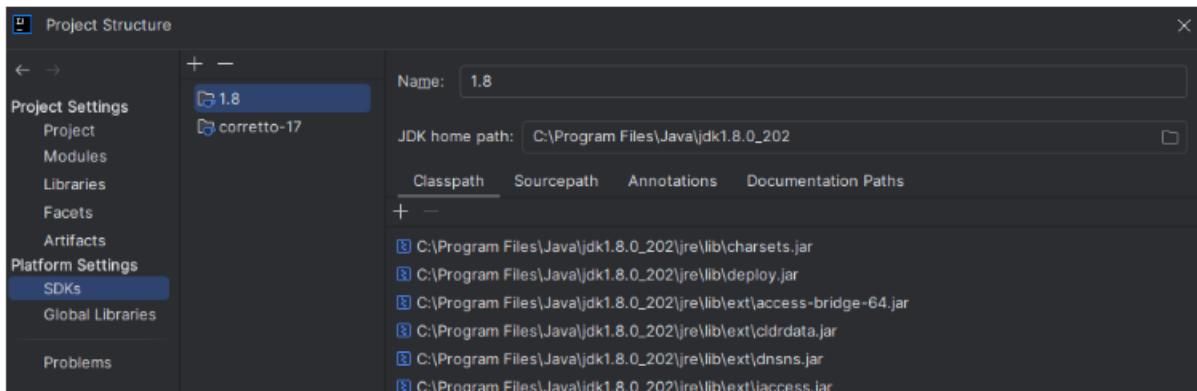


Extrait du fichier README.md

README

Setup with IntelliJ IDE

1. Configure a new project
 - i. Create project from Initializr: File > New > project > Spring Initializr
 - ii. Add lib repository into pom.xml
 - iii. Add folders
 - Source root: src/main/java
 - View: src/main/resources
 - Static: src/main/resource/static
 - iv. Create database with name "demo" as configuration in application.properties
 - v. Run sql script to create table doc/data.sql
2. Modify your configuration
 - i. Download and Install JDK 1.8 on [Oracle](#)
 - ii. In your IDE for this example use IntelliJ **ctrl+alt+shift+s**
 - iii. On Platform Settings menu click > SDKs and add new SDK
 - iv. Select SDK path in windows program files. For this example "****C:\Program Files\Java\jdk1.8.0_202**"
 - v. On Project Settings menu click > Project Select SDK 1.8 and change Language Level to 8 - Lambdas, type annotation, etc.



Activité-type 1

Développer une application sécurisée

CP 2 ▶

Développer des interfaces utilisateur

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

Dans le cadre du projet "Poseiden Skeleton", j'ai conçu et développé plusieurs interfaces utilisateur permettant la gestion d'objets métier (par exemple : utilisateurs, courbes, notations, etc.).

Ces interfaces ont été développées à l'aide du moteur de templates Thymeleaf, intégré à Spring Boot, en suivant une structure MVC. J'ai mis en place des formulaires HTML permettant la saisie et la modification des données, en veillant à la cohérence des validations (côté client et serveur).

L'interface graphique a été stylisée avec Bootstrap v4.3.1, dans le respect des bonnes pratiques UX/UI (accessibilité, responsive design, clarté).

Les vues sont organisées dans le dossier src/main/resources/templates et affichées via les contrôleurs Spring.

2. Précisez les moyens utilisés :

- Langages : HTML5, CSS3, Java (Thymeleaf), Bootstrap
- Outils : IntelliJ IDEA, Spring Boot, navigateur web (Chrome/Firefox pour tests)
- Structure : Templates HTML Thymeleaf intégrés aux contrôleurs Spring
- Librairies : Bootstrap v4.3.1 pour la mise en forme et la réactivité
- Validation : Champs obligatoires, types, messages d'erreur personnalisés

3. Avec qui avez-vous travaillé ?

Le développement de l'interface utilisateur a été réalisé de manière autonome, avec validation régulière des fonctionnalités via une grille d'issues GitHub. J'ai simulé un processus de recette utilisateur en testant chaque écran dans un navigateur.

4. Contexte

Nom de l'entreprise, organisme ou association

► *La Plateforme Formation.
Projet de formation – GitHub "Java-Tp-Poseiden-skeleton"*

Chantier, atelier, service ► *Projet individuel – Environnement full-stack Java*

Période d'exercice ► Du : **30/11/2024** au : **02/12/2024**

5. Informations complémentaires (facultatif)

Ce travail m'a permis de consolider ma compréhension du modèle MVC, d'améliorer mes compétences en intégration HTML/CSS dans un environnement Java, et d'adopter une logique de composant réutilisable pour les vues. J'ai également fait attention à la sécurité des formulaires (CSRF, validation serveur) et à l'ergonomie générale.

Poseiden Sign In

Username

Enter username

We'll never share your username with anyone else.

Password

Password

Sign In

Figure 1 (Vue ADMIN,USER): Page de connexion utilisateur (Login)Page de connexion sécurisée utilisant le module Spring Security, stylisée avec Bootstrap.

Add New Bid

account

account name

type

account type

Bid Quantity

10

Cancel **Add BidList**

*Figure 2 (Vue ADMIN,USER):
Formulaire d'ajout d'un élément dans le
Bid ListExemple de formulaire
dynamique pour ajouter un
enregistrement dans la base, avec
validation côté client.*

Add New Bid

account

type

Bid Quantity

Property bidQuantity threw exception; nested exception is java.lang.NullPointerException: bidQuantity

[Cancel](#) [Add BidList](#)

Figure 3(Vue ADMIN,USER): Message d'erreur sur la saisie d'un champ obligatoireIllustration d'une erreur générée via Spring en cas de champ manquant ou null. Extrait d'un formulaire intégré avec Thymeleaf.

Bid List

Bid List					
Add New		account	type	bidQuantity	
BidListId	Action			Action	
1		123456789	Individual Account	1000.0	Edit Delete

Figure 4(Vue ADMIN,USER): Affichage de la liste des offres (Bid List)Tableau de données généré dynamiquement, affichant les valeurs stockées en base, avec actions de modification et suppression.

[Home](#) | [Login](#)

User List

User List				
Add New				
Id	Full Name	User Name	Role	Action
1	Administrator	admin	ADMIN	Edit Delete
2	User	user	USER	Edit Delete

Figure 5: Liste des utilisateurs (Vue Admin)Interface affichant les utilisateurs existants, avec options d'édition et de suppression. Développée avec Thymeleaf et Bootstrap.

Activité-type 1 Développer une application sécurisée

*CP ** ► Développer des composants métier

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

Dans le cadre du projet "Poseiden Skeleton", j'ai développé plusieurs composants métier permettant la gestion d'éléments tels que les utilisateurs, les offres (BidList), et les notations.

Ces composants métier ont été implémentés dans des classes Java situées dans le package com.nnk.springboot.domain, conformément au modèle MVC.

J'ai utilisé les annotations Spring Boot pour la gestion des entités, les contraintes de validation (@NotNull, @Size, etc.), et la persistance des données.

J'ai également mis en place des services métier pour encapsuler la logique fonctionnelle (ex : création, modification, suppression, règles de gestion), séparant clairement la couche de présentation de la logique applicative.

Cette architecture m'a permis de garantir la réutilisabilité, la lisibilité et la testabilité du code.

2. Précisez les moyens utilisés :

- Langage : Java 8
- Framework : Spring Boot (Annotations @Entity, @Service, @Repository)
- Structure du code :
- Entités : domain/
- Services métier : services/
- Contrôleurs : controllers/
- ORM : Spring Data JPA
- Outils de validation : javax.validation.constraints

- Tests : JUnit

3. Avec qui avez-vous travaillé ?

Le projet a été réalisé individuellement. Toutefois, une démarche de simulation professionnelle a été adoptée : suivi des tâches via GitHub Issues, documentation du code, et logique orientée services pour faciliter un éventuel travail collaboratif ou la reprise par un tiers.

4. Contexte

Nom de l'entreprise, organisme ou association

▶ *La Plateforme Formation
Projet de formation – GitHub Repository "Java-Tp-Poseiden-skeleton"*

Chantier, atelier, service

▶ *Projet individuel – Environnement full-stack Java*

Période d'exercice

▶ Du : **30/11/2024** au : **02/12/2024**

5. Informations complémentaires (facultatif)

Composants métier et services – page suivante

Composants métier et services

BidListService :

service métier

Service Spring qui expose des méthodes métier pour récupérer, transformer et valider les entités BidList sous forme de DTO.

```
x
package com.nnk.springboot.services;

import com.nnk.springboot.domain.BidList;
import com.nnk.springboot.dto.BidListDTO;
import com.nnk.springboot.repositories.BidListRepository;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

/**
 * Service qui gère la logique métier liée aux BidList.
 *
 * Cette classe permet d'interagir avec le repository pour
 * récupérer, rechercher ou ajouter des BidList, et de les
 * transformer en objets DTO (Data Transfer Object).
 */
@Service
public class BidListService {

    private final BidListRepository bidListRepository;

    // Injection du repository via le constructeur
    public BidListService(BidListRepository bidListRepository) {
        this.bidListRepository = bidListRepository;
    }

    /**
     * Récupère toutes les BidList sous forme de DTO.
     *
     * @return une liste de BidListDTO
     */
    public List<BidListDTO> findAllBidListMapDTO() {
        return bidListRepository
            .findAll()
            .stream()
            .map(mapperBidList ->
                new BidListDTO(
                    mapperBidList.getBidListId(),
                    mapperBidList.getAccount(),
                    mapperBidList.getType(),
                    mapperBidList.getBidQuantity()
                )
            )
            .collect(Collectors.toList());
    }

    /**
     * Recherche une BidList par son identifiant.
     *
     * @param id identifiant de la BidList
     * @return l'objet BidList trouvé
     * @throws IllegalArgumentException si l'id n'existe pas
     */
    public BidList findBidListById(Integer id) {
        return bidListRepository
            .findById(id)
            .orElseThrow(
                () -> new IllegalArgumentException("Invalid bidList Id: " + id
            );
    }
}
```

CurvePointService :

traitement des entités

Service métier équivalent pour CurvePoint. Ce service illustre l'usage des Streams, le mapping vers des objets DTO, et la gestion de repository.

```
x
package com.nnk.springboot.services;

import com.nnk.springboot.domain.CurvePoint;
import com.nnk.springboot.dto.CurvePointDTO;
import com.nnk.springboot.repositories.CurvePointRepository;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

/**
 * Service métier pour gérer les CurvePoint.
 */
@Service
public class CurvePointService {

    private final CurvePointRepository curvePointRepository;

    public CurvePointService(CurvePointRepository curvePointRepository) {
        this.curvePointRepository = curvePointRepository;
    }

    /**
     * Récupère tous les CurvePoint et les convertit en DTO.
     *
     * @return liste de CurvePointDTO
     */
    public List<CurvePointDTO> findAllCurvePointMapDTO() {
        return curvePointRepository
            .findAll()
            .stream()
            .map(mapperCurvePoint ->
                new CurvePointDTO(
                    mapperCurvePoint.getId(),
                    mapperCurvePoint.getCurveId(),
                    mapperCurvePoint.getTerm(),
                    mapperCurvePoint.getValue()
                )
            )
            .collect(Collectors.toList());
    }

    /**
     * Recherche un CurvePoint par son identifiant ou lève une exception si non trouvé
     *
     * @param id identifiant du CurvePoint
     * @return CurvePoint trouvé
     */
    public CurvePoint findCurvePointByIdOrElseThrow(Integer id) {
        return curvePointRepository
            .findById(id)
            .orElseThrow(
                () -> new IllegalArgumentException("Invalid curve point Id: " +
            );
    }
}
```

TradeDTO :

exemple d'objet DTO

Objet de transfert de données (DTO) utilisé pour encapsuler des informations entre le service et le contrôleur sans exposer directement les entités JPA.

```
x
package com.nnk.springboot.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;

/**
 * Objet de transfert de données pour la classe Trade.
 *
 * Les DTO permettent de transporter les données
 * entre les couches (service ↔ contrôleur) sans exposer
 * directement les entités JPA.
 */
@Data
@AllArgsConstructor
@Builder
public class TradeDTO {

    private Integer tradeId;
    private String account;
    private String type;
    private Double buyQuantity;
}
```

UserService avec Spring Security

Cet extrait montre la logique métier liée à la gestion des utilisateurs et à la sécurité.

Ici, UserService implémente UserDetailsService, une interface clé de Spring Security.

Lorsqu'un utilisateur tente de se connecter, l'application va :

- Chercher l'utilisateur dans la base via le repository.

- Si l'utilisateur existe, il est transformé en objet UserPrincipal compréhensible par Spring Security.

- Si l'utilisateur n'existe pas, une exception est levée.

```
package com.nnk.springboot.services;

import com.nnk.springboot.config.UserPrincipal;
import com.nnk.springboot.domain.User;
import com.nnk.springboot.repositories.UserRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.userdetails.UserDetails;
import org.springframework.core.userdetails.UserDetailsService;
import org.springframework.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Optional;

/**
 * Service métier pour la gestion des utilisateurs et l'authentification.
 *
 * Cette classe implémente l'interface UserDetailsService de Spring Security,
 * ce qui permet d'intégrer la logique métier d'authentification directement
 * dans le projet.
 */
@Service
public class UserService implements UserDetailsService {

    private final UserRepository userRepository;

    private final Logger logger = LoggerFactory.getLogger(UserService.class);

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    /**
     * Chargement d'un utilisateur par son nom d'utilisateur (username).
     * Cette méthode est appelée automatiquement par Spring Security lors de la connexion
     *
     * @param username nom d'utilisateur saisi lors de la connexion
     * @return un UserDetails (adaptateur de l'entité User)
     * @throws UsernameNotFoundException si l'utilisateur n'existe pas
     */
    @Override
    public UserDetails loadUserByUsername(String username) {
        logger.info("Tentative de connexion pour l'utilisateur : {}", username);
        Optional<User> optionalUser = userRepository.findByUsername(username);

        if (optionalUser.isPresent()) {
            User user = optionalUser.get();
            logger.info("Utilisateur trouvé : {}", user.getUsername());
            return new UserPrincipal(user); // conversion vers l'objet reconnu par Spring Security
        } else {
            logger.error("Utilisateur introuvable : {}", username);
            throw new UsernameNotFoundException(username);
        }
    }
}
```

Activité-type 1 Développer une application sécurisée

CP 4 ► Contribuer à la gestion d'un projet informatique

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

Durant la réalisation du projet "Poseiden Skeleton", j'ai appliqué une gestion de projet inspirée de la méthode Agile. J'ai utilisé un tableau Kanban sur GitHub Projects pour structurer les tâches, les répartir par étapes (To Do, In Progress, Done), et suivre leur avancement.

Chaque tâche technique ou fonctionnelle (ex : création d'une entité, mise en place d'un formulaire, configuration de la base de données, validation...) faisait l'objet d'une issue **GitHub** décrivant précisément l'objectif, les critères d'acceptation, et les dépendances éventuelles.

Cette organisation m'a permis d'avoir une vision claire de l'état du projet à chaque instant, de prioriser les livrables, et de documenter les avancées tout au long du développement.

2. Précisez les moyens utilisés :

- Méthodologie : Kanban inspiré d'Agile
- Outils de gestion : GitHub Projects (tableau Kanban), GitHub Issues
- Outils de versioning : Git (branche principale + commits réguliers)
- Outils de développement : IntelliJ IDEA, Maven
- Documentation : README.md enrichi, commentaires dans le code

3. Avec qui avez-vous travaillé ?

Le projet a été réalisé en autonomie, sans équipe réelle. Toutefois, dans un esprit de professionnalisation, j'ai appliqué une organisation proche de celle d'un travail collaboratif en entreprise, en structurant mes tâches comme si je collaborais avec un ou plusieurs développeurs, avec relecture, suivi d'avancement et gestion de dépendances.

4. Contexte

Nom de l'entreprise, organisme ou association

La Plateforme Formation

Projet de formation – GitHub Repository "Java-Tp-Poseiden-skeleton"

Chantier, atelier, service

Projet individuel – Environnement full-stack Java

Période d'exercice

► Du : **30/11/2024** au : **02/12/2024**

5. Informations complémentaires (facultatif)

Cette démarche m'a permis de m'approprier les outils de gestion de projet modernes, d'optimiser mon temps, de clarifier mes priorités et de m'entraîner à une méthode de travail adaptable à un environnement professionnel.

J'ai également pris l'habitude de **committer régulièrement** avec des messages clairs, et de **documenter chaque étape importante**, ce qui facilite le travail en équipe et le suivi.

Cette annexe présente les éléments illustrant ma contribution à la gestion du projet Java-Tp-Poseiden. La gestion de projet a été réalisée principalement à l'aide de GitHub (issues, commits) et d'un tableau Kanban pour organiser les tâches de développement. Les bonnes pratiques de gestion de versions et de suivi de l'avancement ont été appliquées tout au long du cycle de vie du projet.

Suivi des tâches avec GitHub Issues

Le projet utilise des issues pour suivre les tâches de développement.

Chaque issue est affectée à une fonctionnalité spécifique, avec un statut (à faire, en cours, terminé).

Author	Labels	Projects	Milestones	Assignees	Newest
<input type="checkbox"/> Bug : Erreur de template lors de la génération de la page "add.html"	Form Front-End HTML Templates				
#58 · by jean-ely-gendrau was closed on Dec 8, 2024	Validation et Test...				
<input type="checkbox"/> Configuration : Ajout d'un constructeur surchargé dans les classes Domain	Back-End BidList CurvePoint Entity Form Rating RuleName Tests				
#51 · by jean-ely-gendrau was closed on Dec 8, 2024	Configuration et ...				
<input type="checkbox"/> Feature Request : Création de DTOs pour les services Tagger (#31)	Back-End BidList CurvePoint DTO enhancement Rating RuleName Service Injection Trade				
#47 · by jean-ely-gendrau was closed on Dec 5, 2024	Configuration et ...				
<input type="checkbox"/> [BUG] - Problème avec GenerationType.AUTO et l'auto-incrémation des ID	bug JPA				
#45 · by jean-ely-gendrau was closed on Dec 5, 2024	Configuration et ...				
<input type="checkbox"/> [BUG] Modifier le Driver MySQL pour permettre à JPA de créer les tables	bug Configuration				
#43 · by jean-ely-gendrau was closed on Dec 3, 2024	Configuration et ...				
<input type="checkbox"/> Configuration de Spring Security pour gestion du trafic	enhancement Security Setup SpringSecurity				
#41 · by jean-ely-gendrau was closed on Dec 12, 2024	Fonctionnalités P...				1
<input type="checkbox"/> Compléter les formulaires et affichages	BidList CurvePoint Form Front-End HTML Rating RuleName Templates Trade				
#40 · by jean-ely-gendrau was closed on Dec 10, 2024	Configuration et ...				
<input type="checkbox"/> Implémentation du contrôleur TradeController	Back-End CRUD JPA Logging Tests Trade				
#36 · by jean-ely-gendrau was closed on Dec 10, 2024	Validation et Test...				
<input type="checkbox"/> Implémentation du contrôleur RuleNameController	Back-End CRUD JPA Logging RuleName Service Injection Tests				
#35 · by jean-ely-gendrau was closed on Dec 10, 2024	Validation et Test...				
<input type="checkbox"/> Implémentation du contrôleur RatingController	Back-End CRUD enhancement JPA Logging Rating Service Injection Tests				
#34 · by jean-ely-gendrau was closed on Dec 10, 2024	Validation et Test...				
<input type="checkbox"/> Implémentation du contrôleur CurvePointController	Back-End CRUD CurvePoint JPA Logging Service Injection Tests				
#33 · by jean-ely-gendrau was closed on Dec 10, 2024	Validation et Test...				

@jean-ely-gendrau's Poseiden-Skeleton

Backlog | Priority board | Team items | Roadmap | In review | My items | View 7 | New view

Filter by keyword or by field

Backlog	Ready	In progress	In review	Done
2 / 5 Estimate: 1 This item hasn't been started Java-Tp-Poseiden-skeleton #47 Feature Request: Création de DTOs pour les services Tagger (#31) PO I S	5 Estimate: 1 This is ready to be picked up Java-Tp-Poseiden-skeleton #45 BUG - Problème avec GenerationType.AUTO et l'auto-incrémentation des ID PO	3 / 4 Estimate: 0 This is actively being worked on Java-Tp-Poseiden-skeleton #42 #41 Create WebSecurityConfig.java - granted free access to certain resources at startup PI I M	1 / 5 Estimate: 0 This item is in review Java-Tp-Poseiden-skeleton #40 Compléter les formulaires et affichages PI S	22 Estimate: 1 This has been completed Java-Tp-Poseiden-skeleton #50 Feature/32 controller bidist Java-Tp-Poseiden-skeleton #57 Feature/40 html form Java-Tp-Poseiden-skeleton #49 Feature/31 config init services Java-Tp-Poseiden-skeleton #51 Configuration : Ajout d'un constructeur surchargé dans les classes Domain Java-Tp-Poseiden-skeleton #53 Refactor/51 domain constructor Java-Tp-Poseiden-skeleton #54 Feature/40 html form Java-Tp-Poseiden-skeleton #55 Dev

Configuration et Base du Projet (Semaine 1 - Début) - #v1.1.0 17 Estimates: 2

Organisation des tâches via tableau Kanban

Un tableau de type Kanban a été mis en place pour visualiser l'avancement des tâches.

Les colonnes représentent les statuts : à faire, en cours, terminé.

Historique des commits Git

L'historique Git présente des commits réguliers et explicites, permettant de suivre l'évolution du projet par étapes fonctionnelles.

Les messages de commits respectent une convention simple et cohérente.

```
commit 6a031ffca7ca001cdc962102b7f68d01ff20eb74
Merge: 156fd88 5c5596d
Author: jean-ely.gendrau <150784000+jean-ely-gendrau@users.noreply.github.com>
Date: Tue Dec 10 15:55:58 2024 +0100

Merge pull request #65 from jean-ely-gendrau/fix/40-correctly-rename

fix(#40): minor adjustment to variable names passed to parameterized...
```

Activité-type 2 Concevoir et développer une application sécurisée organisée en couche

CP 5 ▶ Analyser les besoins et maquetter une application

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

Dans le cadre du projet “Flash Cash”, j’ai été chargé d’analyser les besoins d’une application de type “wallet numérique”, destinée à permettre aux utilisateurs de gérer leurs finances en ligne : dépôt, retrait, transferts, gestion des contacts, sécurité des transactions.

J’ai commencé par définir les fonctionnalités clés en m’appuyant sur les pratiques d’usage des applications bancaires : tableau de bord, formulaires de transaction, sécurité renforcée, etc.

J’ai ensuite structuré ces besoins sous forme de ***spécifications fonctionnelles*** : parcours utilisateur, champs nécessaires, règles métiers associées. Pour la partie visuelle, j’ai conçu des maquettes simples directement en HTML/CSS intégrées à des fragments Thymeleaf ([fragments/home.html](#), [panels.html](#), [selects.html](#)), tout en adoptant une approche responsive design avec Tailwind et classes utilitaires.

2. Précisez les moyens utilisés :

- Méthode : Expression des besoins via liste de fonctionnalités
- Outils de maquettage : Code HTML + Tailwind CSS + Thymeleaf (pour simulation)
- Techno d’appui : IntelliJ IDEA, Visual Studio Code (pour structuration front)
- Support projet : GitHub + gestion des tâches manuelle

3. Avec qui avez-vous travaillé ?

L’utilisation de Thymeleaf avec des composants graphiques basés sur Tailwind CSS m’a permis de concevoir rapidement des interfaces claires et réactives. Cette phase a également permis de mieux structurer les responsabilités entre les différents modules front-end, en prévision de la mise en œuvre côté back-end.

4. Contexte

Nom de l'entreprise, organisme ou association
► *La Plateforme Formation - Projet personnel de formation – GitHub "Java-Tp-FlashCash"*

Chantier, atelier, service ► *Conception et développement full-stack en autonomie*

Période d'exercice ► Du : **17/12/2024** au : **17/01/2025**

5. Informations complémentaires (facultatif)

L'utilisation de Thymeleaf avec des composants graphiques basés sur Tailwind CSS m'a permis de concevoir rapidement des interfaces claires et réactives. Cette phase a également permis de mieux structurer les responsabilités entre les différents modules front-end, en prévision de la mise en œuvre côté back-end.

Activité-type 2 Concevoir et développer une application sécurisée organisée en couche

CP 6 ► Définir l'architecture logicielle d'une application

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

Dans le cadre du projet "Flash Cash", j'ai défini et mis en œuvre une architecture logicielle respectant le modèle **MVC** (Model-Vue-Controller), complété d'une structure en **couches distinctes** pour séparer les responsabilités.

L'architecture comprend des packages dédiés aux **contrôleurs, services, repositories, modèles de données** (model), ainsi que des **DTO** pour les échanges entre couches.

J'ai également intégré une logique de configuration (config) et des classes utilitaires (utils) pour renforcer la clarté du code.

Cette organisation permet de garantir **la lisibilité, la maintenabilité et la testabilité** de l'application. Elle respecte les bonnes pratiques d'un projet professionnel basé sur Spring Boot.

2. Précisez les moyens utilisés :

- **Architecture logicielle** : Modèle MVC + séparation en couches (controller, service, repository, dto, config, etc.)
- **Technologies utilisées** :
 - Java 17
 - Spring Boot 3.4.4
 - Maven (gestion de projet)
- **Structure de projet** :
 - controller : gestion des routes et appels
 - service : logique métier
 - repository : accès aux données
 - dto : transfert de données
 - config : paramètres Spring Security, application.properties, etc.

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie, dans un cadre de formation. J'ai structuré l'architecture comme si elle devait être reprise par d'autres développeurs : noms de dossiers explicites, conventions de code, séparation nette entre les responsabilités, et commentaires techniques dans le code.

4. Contexte

Nom de l'entreprise, organisme ou association

► *La Plateforme Formation - Projet personnel de formation – GitHub "Java-Tp-FlashCash"*

Chantier, atelier, service

► *Conception et développement full-stack en autonomie*

Période d'exercice

► Du : **17/12/2024** au : **17/01/2025**

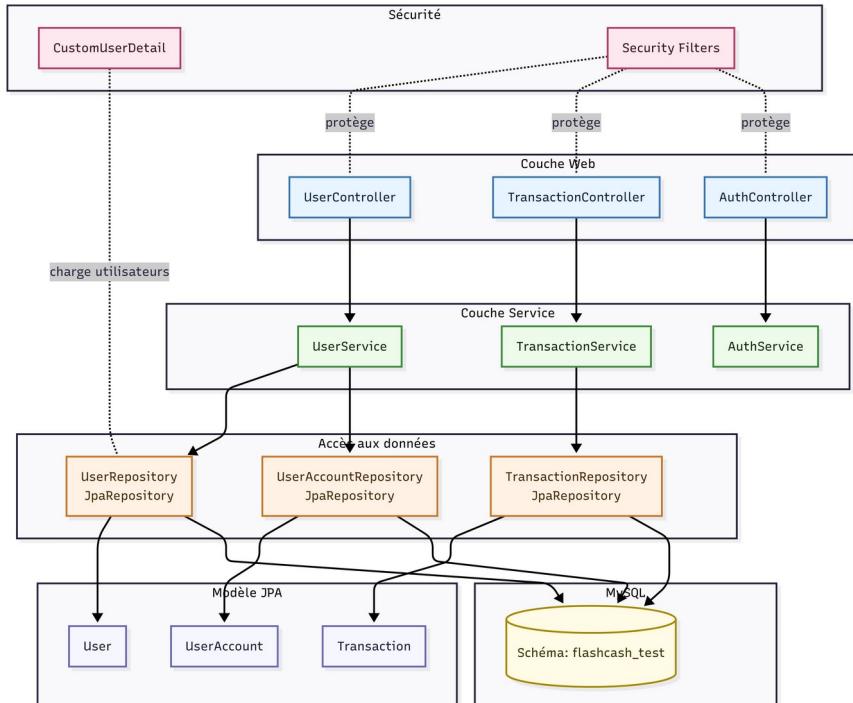
5. Informations complémentaires (facultatif)

Ce travail m'a permis de consolider mes compétences sur les **design patterns** utilisés dans Spring Boot, d'adopter une logique orientée service, et de mieux comprendre l'impact d'une architecture bien pensée sur la sécurité, la performance et la maintenabilité.

Le projet a été conçu pour pouvoir évoluer facilement avec des tests, une API REST ou une interface front-end découpée.

Composants Spring

Les contrôleurs REST gèrent les requêtes entrantes, déléguent la logique métier aux services, qui eux-mêmes accèdent aux données via les repositories Spring Data JPA et les entités persistées en base.



Activité-type 2 Concevoir et développer une application sécurisée organisée en couche

CP 7 ► Concevoir et mettre en place une base de données relationnelle

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

Pour le projet "Flash Cash", j'ai conçu et mis en place une base de données relationnelle MySQL dédiée au stockage des données liées aux utilisateurs, aux transactions, aux comptes, et aux fonctionnalités de gestion financière.

J'ai commencé par créer une base nommée `flashcash_test`, puis j'ai mis en place un utilisateur spécifique `flashcash-test`, avec un mot de passe dédié, afin de ne pas utiliser le compte `root`. J'ai ensuite attribué les droits nécessaires sur la base via un script SQL, tout en automatisant l'exécution avec Spring Boot (`spring.sql.init.mode=always`).

La structure de la base a été ensuite gérée via **JPA/Hibernate**, avec l'option `ddl-auto=create-drop` pour créer les tables à partir des entités Java à chaque démarrage de l'application.

2. Précisez les moyens utilisés :

- SGBD utilisé : MySQL
- Script SQL utilisé :
 - Crédit de la base : `CREATE DATABASE flashcash_test;`
 - Crédit de l'utilisateur et des droits
- Fichier de configuration : `application.properties`
 - `spring.datasource.url`, `username`, `password`, `init.mode`, `ddl-auto`
- Accès aux données : JPA / Hibernate (entités dans model, mapping automatique)
- Options dev : `spring.jpa.show-sql=true` pour visualiser les requêtes

3. Avec qui avez-vous travaillé ?

Le projet a été réalisé individuellement dans un cadre pédagogique. Néanmoins, la base de données a été conçue pour pouvoir évoluer et être **utilisable par d'autres développeurs**, grâce à l'utilisation d'un script SQL documenté et d'un fichier de configuration clair.

4. Contexte

Nom de l'entreprise, organisme ou association
► *La Plateforme Formation - Projet personnel de formation – GitHub "Java-Tp-FlashCash"*

Chantier, atelier, service ► *Back-end – intégration base de données relationnelle*

Période d'exercice ► Du : **17/12/2024** au : **17/01/2025**

5. Informations complémentaires (facultatif)

J'ai pris soin de séparer les fichiers de configuration pour différents environnements (dev-a, prod), ce qui facilite le déploiement futur.

La conception d'une base testable et sécurisée dès le départ m'a permis d'améliorer ma rigueur et ma maîtrise des outils professionnels.

Illustrations – Base de données relationnelle :



```
# Nom de l'application Spring Boot
spring.application.name=JAVA-Tp-FlashCash

# Activation du mode debug (utile en développement pour avoir plus de logs)
debug=true

# Niveau de journalisation pour le framework Spring
logging.level.org.springframework=INFO

# Niveau de journalisation plus détaillé pour le module Spring Security
logging.level.org.springframework.security=DEBUG

# Profil actif de l'application (utilisé pour charger des configurations spécifiques à l'environnement)
spring.profiles.active=dev-a

##### Configuration de la source de données #####
# Définition du driver JDBC pour MySQL
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

# URL de connexion à la base de données (ici en local, base flashcash_test)
spring.datasource.url=jdbc:mysql://localhost:3306/flashcash_test

# Nom d'utilisateur pour la connexion à la base de données
spring.datasource.username=flashcash-test

# Mot de passe associé à l'utilisateur de la base de données
spring.datasource.password=admin-password

##### Configuration d'Hibernate #####
# Définition du mode d'initialisation des scripts SQL (ici, toujours exécuté au démarrage)
spring.sql.init.mode=always

# Spécifie la plateforme SQL utilisée (ici MySQL)
spring.sql.init.platform=mysql

# (Optionnel) Emplacement des scripts SQL à exécuter au démarrage
# spring.sql.init.schemaLocations=classpath:import.sql

# Stratégie de gestion du schéma Hibernate : créer la base au démarrage et la supprimer à l'arrêt
spring.jpa.hibernate.ddl-auto=create-drop

# Affiche les requêtes SQL générées par Hibernate dans les logs (utile pour le debug)
spring.jpa.show-sql=true
```

Figure 1: Fichier application.properties (extrait)Paramétrage complet de la connexion à la base de données MySQL (flashcash_test), avec activation du profil dev-a, configuration Hibernate (ddl-auto=create-drop) et logs SQL.


```
-- Création de la base de données nommée "flashcash_test"
CREATE DATABASE flashcash_test;

-- Création d'un utilisateur "flashcash-test" avec un mot de passe "admin-password"
-- Cet utilisateur sera autorisé uniquement à se connecter depuis "localhost"
CREATE USER 'flashcash-test'@'localhost' IDENTIFIED BY 'admin-password';

-- Attribution de tous les privilèges sur la base "flashcash_test" à l'utilisateur
-- "flashcash-test"
-- Cela inclut les droits de lecture, écriture, modification et suppression
GRANT ALL PRIVILEGES
ON flashcash_test.*  
TO 'flashcash-test'@'localhost';

-- Application immédiate des modifications de privilèges
FLUSH PRIVILEGES;
```

Figure 2: Script SQL d'initialisation de la base de données. Script permettant de créer la base flashcash_test, un utilisateur flashcash-test, et de lui attribuer les privilèges nécessaires à l'environnement de développement local sécurisé.

Activité-type 2 Concevoir et développer une application sécurisée organisée en couche

CP 8 ► Définir l'architecture logicielle d'une application

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

Dans le cadre du projet “Flash Cash”, j’ai développé plusieurs composants d’accès aux données pour interagir avec la base MySQL via **Spring Data JPA**.

Pour chaque entité (ex. User, Transaction, Account), j’ai créé un **repository dédié**, en héritant de l’interface JpaRepository. Cela m’a permis de gérer facilement les opérations de base (CRUD), tout en ajoutant des méthodes personnalisées grâce aux **requêtes dérivées** (ex : findByEmail, findByUsername).

Les composants d’accès ont été isolés dans le package repository, et injectés dans les services via le système d’injection de dépendance de Spring.

Ce découplage m’a permis de respecter une architecture en couches, tout en rendant les tests unitaires plus simples à implémenter (même si non utilisés ici).

2. Précisez les moyens utilisés :

- **Base de données** : MySQL
- **Accès aux données** : Spring Data JPA
- **Classes impliquées** :
 - @Entity dans model
 - @Repository dans repository
- **Fonctionnalités utilisées** :
 - Méthodes CRUD automatiques (save, findAll, deleteById, etc.)
 - Requêtes personnalisées dérivées du nom de méthode
- **Technos** : Java 17, Spring Boot, Hibernate ORM

“L’usage de Spring Data JPA permet de coder rapidement des accès SQL robustes, avec moins de code et une meilleure lisibilité.”

3. Avec qui avez-vous travaillé ?

Le projet a été mené en autonomie dans un cadre pédagogique, mais l'architecture a été pensée pour un environnement professionnel avec une base SQL partagée. Chaque repository est documenté et conçu pour être facilement modifiable ou étendu par d'autres développeurs.

4. Contexte

Nom de l'entreprise, organisme ou association
▶ *La Plateforme Formation - Projet personnel de formation – GitHub "Java-Tp-FlashCash"*

Chantier, atelier, service ▶ *Couche d'accès aux données*

Période d'exercice ▶ Du : **17/12/2024** au : **17/01/2025**

5. Informations complémentaires (facultatif)

Même si le projet ne comprend pas de données NoSQL, la conception orientée entités/repositories pourrait facilement être adaptée à MongoDB ou un autre SGBD NoSQL via Spring Data Mongo. Ce travail m'a permis de mieux comprendre le mapping objet-relationnel, la gestion des transactions et la sécurité des accès aux données.

Illustrations des composants d'accès aux données



```
package org.example.javatpflashcah.repository;

import org.example.javatpflashcah.model.Transaction;
import org.example.javatpflashcah.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface TransactionRepository extends JpaRepository<Transaction, Integer>
{
    List<Transaction> findTransactionByFrom(User user);
}
```

Figure 1: Interface TransactionRepositoryExemple de repository JPA dédié à l'entité Transaction, avec une méthode personnalisée findTransactionByFrom(User user) dérivée du nom de la propriété.

```

    /**
     * Recherche un utilisateur à partir de son adresse email.
     *
     * @param email l'adresse email de l'utilisateur à rechercher
     * @return l'utilisateur correspondant à l'adresse email fournie
     * @throws RuntimeException si aucun utilisateur n'est trouvé avec cet email
     */
    public User findByEmail(String email) {
        // Appelle le repository pour tenter de retrouver l'utilisateur par son email.
        // Si aucun utilisateur n'est trouvé, une exception RuntimeException est levée.
        return userRepository.findByEmail(email).orElseThrow(
            () -> new RuntimeException("User not found")
        );
    }

```

Figure 2: Méthode métier `findByEmail` dans le serviceMéthode de service qui utilise `UserRepository` pour récupérer un utilisateur par `email`. Elle applique une gestion d'erreur en lançant une exception si l'utilisateur n'est pas trouvé.

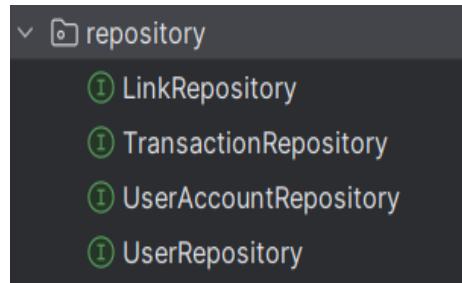


Figure 3: Arborescence des repositories dans l'IDE (IntelliJ)Vue d'ensemble des interfaces de repository JPA créées pour les entités principales du projet : utilisateur, transaction, liens et comptes.

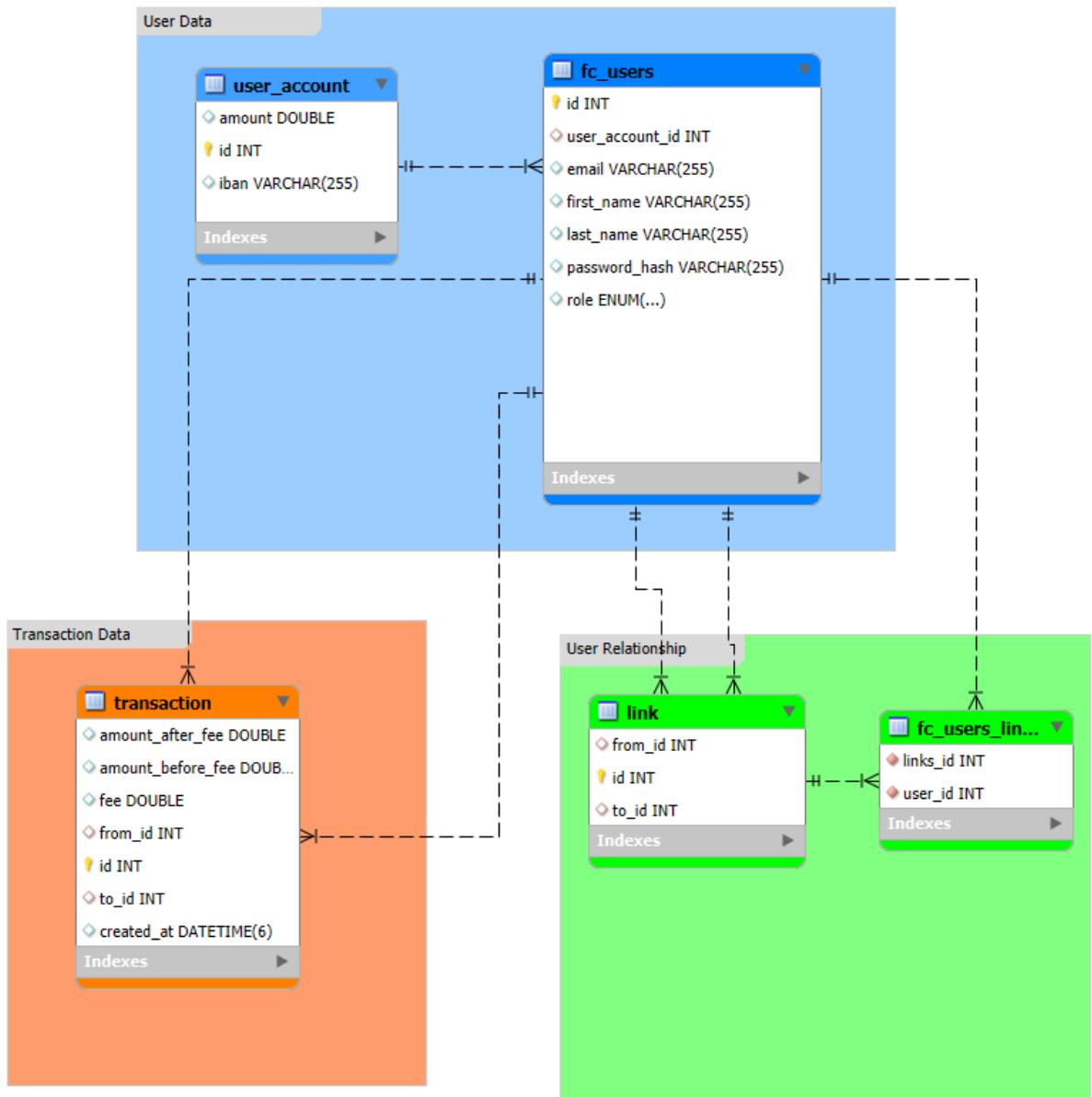


Figure 4: **Modèle physique de données (MPD) de l'application FlashCash**

Schéma relationnel représentant les tables **fc_users**, **transaction**, **user_account** et **link**, avec les relations et clés étrangères. Ce modèle illustre la structuration logique des données dans la base MySQL.

Activité-type 3 Préparer le déploiement d'une application sécurisée

CP 9 ▶ Préparer et exécuter les plans de tests d'une application

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

Dans le cadre du projet Java-Tp-SafetyNet, j'ai préparé et exécuté un ensemble de **tests unitaires et d'intégration** destinés à valider le bon fonctionnement des différents composants de l'application.

Ce projet a été développé avec **Spring Boot** et suit une architecture **MVC**. J'ai appliqué une stratégie de tests couvrant les *modèles, services, contrôleurs* et certaines **fonctions métier critiques**.

Les tests unitaires ont été réalisés à l'aide de **JUnit 5** et **Mockito**, avec injection de dépendances (@Mock, @InjectMocks) pour isoler la logique métier.

Les tests d'intégration sont exécutés via l'annotation **@SpringBootTest** et ciblent les cas réels d'utilisation, notamment les alertes incendie, les inondations et les urgences médicales.

Le projet intègre également JaCoCo pour mesurer la couverture de code, et produit un rapport automatique lors du build Maven (mvn test).

2. Précisez les moyens utilisés :

- **Outils et bibliothèques :**
 - **JUnit 5** pour les tests unitaires
 - **Mockito** pour les mocks de dépendances
 - **SpringBootTest** pour les tests d'intégration
 - **JaCoCo** pour la couverture de code
 - **Maven** pour l'automatisation (mvn test, rapport Surefire)
- **Organisation des tests :**
 - **src/test/java/.../model** : tests des entités
 - **.../service** : tests unitaires des méthodes métier
 - **.../controller** : tests unitaires et d'intégration des endpoints
 - **integration/controller** : cas réels complets simulant des scénarios utilisateurs (ex : incendie, inondation...)

3. Avec qui avez-vous travaillé ?

Le projet a été mené individuellement dans un cadre pédagogique. Cependant, l'ensemble des tests a été pensé dans une logique de production, avec une séparation claire entre les tests unitaires (vérification isolée des méthodes) et les tests d'intégration (vérification globale du comportement applicatif). Les résultats sont vérifiables et réutilisables par d'autres développeurs, et le projet respecte les principes SOLID.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ *La Plateforme Formation - Projet de formation – Java-Tp-SafetyNet*

Chantier, atelier, service ▶ *Tests applicatifs – Spring Boot back-end*

Période d'exercice ▶ Du : **15/11/2024** au : **30/11/2024**

5. Informations complémentaires (facultatif)

La couverture de code mesurée par JaCoCo dépasse les 80 %, conformément aux exigences du cahier des charges. Le rapport de test généré par Maven permet de tracer les classes couvertes, les méthodes testées, et les éventuelles faiblesses à corriger.

L'utilisation de mocks a facilité l'isolation des cas métier, et les tests d'intégration ont permis de simuler des cas concrets, garantissant la fiabilité de l'API en conditions réelles.

Activité-type 3 Préparer le déploiement d'une application sécurisée

CP 10 ▶ Préparer et documenter le déploiement d'une application

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

Dans le cadre du projet “*HealthCare*”, j'ai préparé le déploiement complet d'une application web structurée en microservices à l'aide de Docker et Docker Compose.

J'ai intégré plusieurs services indépendants (*patient*, *consultation*, *authentification*, *scoring diabète*) ainsi que les bases de données associées (**MySQL**, **MongoDB**, **PostgreSQL**) dans un environnement **orchestré localement**.

J'ai également mis en place un **Consul Server** pour le service discovery et un **Config Server** pour la gestion centralisée des propriétés de configuration.

Chaque service dispose de son propre `env_file` pour distinguer les environnements de développement.

La documentation du déploiement est fournie dans un `README.md`, avec une structure claire, et un script `install.sh` simplifie le démarrage de l'environnement.

2. Précisez les moyens utilisés :

- **Outils de déploiement :**
 - Docker
 - Docker Compose
 - fichiers `env` séparés pour chaque microservice
 - volumes persistants pour les bases de données
- **Structure du projet :**
 - `docker-compose-dev.yml` : orchestration des services
 - `common-config.yml` : configuration technique partagée
 - `common-config-service.yml` : gestion mémoire/CPU/restart/networks
 - `README.md` : instructions de lancement et d'utilisation
 - `install.sh` : script de démarrage simplifié

- **Services déployés :**

- config-server, eureka-server, gateway, authentication-service
- note-service (consultations), patient-service, diabetes-scoring-service
- bases : MongoDB, MySQL, PostgreSQL

“La séparation entre fichiers de config permet une adaptabilité rapide pour un environnement de prod, préprod ou local”

3. Avec qui avez-vous travaillé ?

Le projet a été réalisé individuellement dans un cadre pédagogique avancé. Les choix techniques et la documentation ont été conçus pour permettre une reprise facile du projet par d'autres développeurs, et faciliter l'industrialisation en équipe (environnement standardisé, configuration isolée, portabilité garantie).

4. Contexte

Nom de l'entreprise, organisme ou association ►	<i>La Plateforme Formation - Projet de formation – TP HealthCare (architecture microservices)</i>
Chantier, atelier, service	► <i>Préparation au déploiement avec Docker</i>
Période d'exercice	► Du : 03/03/2025 au : 10/04/2025

5. Informations complémentaires (facultatif)

L'environnement complet peut être démarré localement avec une seule commande Docker Compose, après configuration des variables d'environnement.

L'ensemble des services sont reliés à des réseaux Docker spécifiques (bridge, healthcare-overlay) assurant leur interopérabilité.

Des **tests de santé** (*healthcheck*) sont définis pour chaque service, garantissant un démarrage cohérent et fiable du système.

Activité-type 3 Préparer le déploiement d'une application sécurisée

CP 11 ▶ Contribuer à la mise en production dans une démarche DevOps

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

Dans le cadre du projet “*HealthCare*”, j'ai mis en œuvre une stratégie DevOps basée sur la conteneurisation complète des composants applicatifs à l'aide de Docker et Docker Compose.

Chaque microservice (authentification, patient, consultation, scoring diabète) est encapsulé dans son propre conteneur, avec ses variables d'environnement, ses ports exposés et ses dépendances correctement gérées via le fichier `docker-compose-dev.yml`.

Les bases de données utilisées (MySQL, PostgreSQL, MongoDB) sont elles aussi conteneurisées, avec des **volumes Docker persistants** pour assurer la conservation des données entre les redémarrages.

Des **fichiers de configuration partagés** (`common-config.yml`, `common-config-service.yml`) permettent de normaliser la configuration de ressources (CPU, mémoire, logs) et de simplifier l'administration des services.

L'orchestration repose sur des **réseaux Docker personnalisés** assurant une communication efficace entre les conteneurs.

2. Précisez les moyens utilisés :

- **Conteneurisation** : Docker, Docker Compose
- **Services déployés en conteneur** :
 - Microservices (auth, patient, note, scoring diabète)
 - Bases de données (MySQL, PostgreSQL, MongoDB)
 - Serveurs de configuration (Config Server), Service Discovery (Consul)
- **Infrastructure Docker** :
 - Volumes (`db_data_patient`, `postgres_data`, `mongo-data`)
 - Réseaux (`healthcare`, `healthcare-overlay`, `cluster_consul_consul-net`)
 - Healthchecks pour chaque service
 - Gestion des ressources (RAM, CPU, logs) via `common-config.yml`

- **Documentation et automatisation :**
 - README.md expliquant le lancement
 - Script `install.sh` pour initialisation rapide
 - Utilisation de fichiers `.env` par microservice pour isoler la configuration

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé individuellement. Toutefois, l'environnement a été conçu pour faciliter une mise en production collaborative : services modulaires, configuration externe, orchestration déclarative, et séparation claire des responsabilités.

Chaque service peut être repris, modifié, déployé ou monitoré indépendamment grâce à l'infrastructure mise en place.

4. Contexte

Nom de l'entreprise, organisme ou association ▶	<i>La Plateforme Formation - Projet de formation – TP HealthCare (architecture microservices)</i>
Chantier, atelier, service	▶ <i>Mise en production locale conteneurisée</i>
Période d'exercice	▶ Du : 03/03/2025 au : 10/04/2025

5. Informations complémentaires (facultatif)

Cette démarche DevOps constitue une base solide pour une évolution future **vers une intégration continue (CI) ou un déploiement continu (CD)** via GitHub Actions, Jenkins ou GitLab CI.

L'utilisation de services comme `healthcheck`, `depends_on`, `env_file`, ainsi que la séparation logique par services, assure une **reproductibilité totale** de l'environnement sur n'importe quelle machine compatible Docker.

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(*facultatif*)

Intitulé	Autorité ou organisme	Date
TAI	AFPA	2007
DWWM	La Plateforme	2024
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] *Jean-ely Gendrau*.....,

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 *Toulon*..... le *01/09/2025*.....

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

(Si le RC le prévoit)