



MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance

- Roullet

Nom d'usage

-

Prénom

- Jacques

Adresse

- 567 chemin du vieux Reynier, 83500 La Seyne-sur-mer

Titre professionnel visé

Cliquez ici pour entrer l'intitulé du titre professionnel visé.

MODALITÉ D'ACCÈS :

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

DOSSIER PROFESSIONNEL (DP)

Présentation du dossier

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

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

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

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

Pour prendre sa décision, le jury dispose :

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

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

Ce dossier comporte :

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

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

DOSSIER PROFESSIONNEL^(DP)



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

DOSSIER PROFESSIONNEL (DP)

Sommaire

Exemples de pratique professionnelle

Développer une application sécurisée

- Installer et configurer son environnement de travail en fonction du projet p. 7
- Développer des interfaces utilisateur p. 14
- Développer des composants métiers p. 24
- Contribuer à la gestion d'un projet informatique p. 34

Concevoir et développer une application sécurisée organisée en couches

- Analyser les besoins et maquetter une application p. 38
- Définir l'architecture logicielle d'une application p.
- Concevoir et mettre en place une base de données relationnelle p.
- Développer des composants d'accès aux données SQL et NoSQL p.

Préparer le déploiement d'une application sécurisée

- Préparer et exécuter les plans de tests d'une application p.
- Préparer et documenter le déploiement d'une application p.
- Contribuer à la mise en production dans une démarche DevOps p.

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.

DOSSIER PROFESSIONNEL ^(DP)

DOSSIER PROFESSIONNEL ^(DP)

EXEMPLES DE PRATIQUE

PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°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 :

Au cours de ma formation, j'ai installé et configuré mon environnement de travail dans le cadre de l'application FlashCash. Il s'agit d'une application de gestion de transactions financières, permettant d'effectuer différentes opérations telles que : ajout de fonds, virement entre utilisateurs, ou transfert vers une banque externe.

Sur ma machine:

- Installation de la JDK, choix de la JDK pour le projet
- Installation d'IntelliJ, environnement de développement
- Installation de Maven, gestion de projet et de dépendances respectivement grâce aux cycles de vie (clean, compile, package...) et du fichier pom.xml
- Installation de Postman, pour tester les endpoints des API REST
- Installation de MySQL Workbench pour gérer et visualiser la base de données
- Installation de Git pour le suivi de version.

En ligne:

- Initialisation du projet avec Spring Initializr avec choix des dépendances directement intégrées dans le projet (Spring WEB, Spring Security, Spring Data, Lombok, ...)

2. Précisez les moyens utilisés :

- JDK 17 : installée localement pour compiler et exécuter les applications Spring Boot dans un environnement compatible et à jour.
- Spring Initializr : pour générer les projets avec les dépendances nécessaires dès l'initialisation.
- IntelliJ IDEA (IDE): environnement de développement, utilisé pour écrire, organiser et exécuter le code.
- Maven : outil de gestion de projet utilisé pour la compilation, la gestion des dépendances, les profils d'environnement et le packaging.
- Postman : utilisé pour tester les endpoints de l'API REST (authentification, création d'utilisateurs, manipulation de patients...).
- MySQL Workbench : utilisé pour gérer et visualiser la base de données, analyser les données persistées, et exécuter des requêtes SQL de contrôle.
- Git : utilisé pour le suivi de version et dépôt du projet sur GitHub.

DOSSIER PROFESSIONNEL (DP)

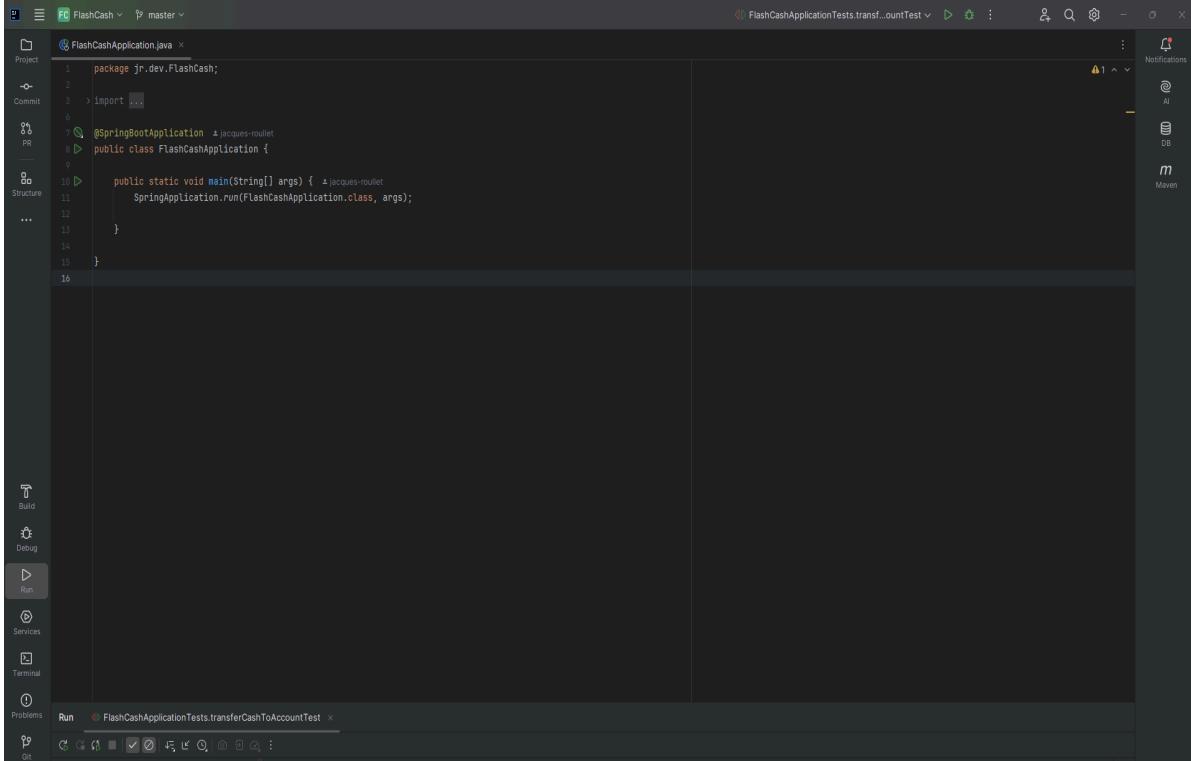
The screenshot shows the Spring Initializr interface at start.spring.io. The configuration is set for a Java Spring Boot application:

- Project:** Maven
- Language:** Java
- Spring Boot:** 3.4.4
- Project Metadata:**
 - Group: com.example
 - Artifact: demo
 - Name: demo
 - Description: Demo project for Spring Boot
 - Package name: com.example.demo
- Packaging:** Jar
- Java:** 21 (selected)

Dependencies: Spring Web (selected)

Buttons: GENERATE (CTRL + D), EXPLORE (CTRL + SPACE), ...

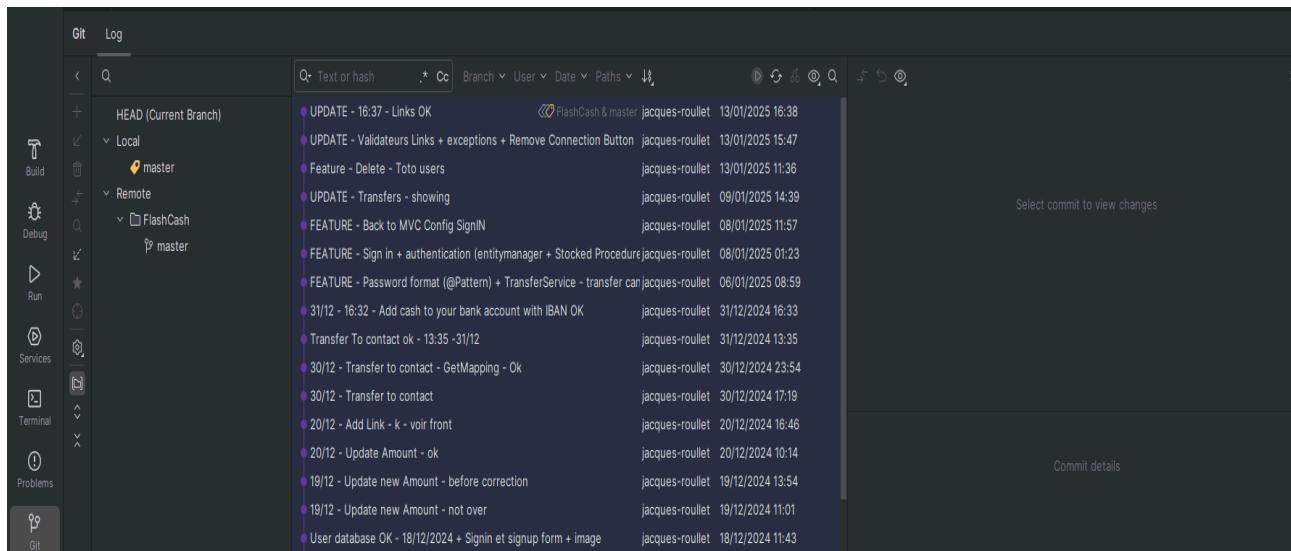
DOSSIER PROFESSIONNEL (DP)



The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** FlashCash, master
- Project Tool Window:** Shows the project structure for "FlashCashApplication.java".
- Code Editor:** Displays the Java code for `FlashCashApplication.java`, which includes imports, annotations, and a main method.
- Toolbars and Buttons:** Includes standard IntelliJ buttons for Build, Run, and Debug.
- Bottom Navigation:** Shows tabs for "Run" and "FlashCashApplicationTests.transferCashToAccountTest".
- Right Sidebar:** Contains sections for Notifications, AI, DB, and Maven.

IDE - IntelliJ



The screenshot shows the Git log in IntelliJ IDEA with the following details:

- Log Tab:** Selected, showing the commit history.
- Commit List:** Lists commits from HEAD (Current Branch) to the master branch of the FlashCash project. Commits include various updates, feature additions, and bug fixes.
- Toolbar:** Includes icons for Build, Run, Services, Terminal, and Problems.
- Right Panel:** Shows "Select commit to view changes" and "Commit details" sections.

Git - versioning

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a Java development environment with the following interface elements:

- Project View:** Shows the project structure with files like `FlashCashApplication.java` and `pom.xml`.
- Maven Tool Window:** Located on the right, it displays the Maven project tree and various build lifecycle phases (clean, validate, compile, test, package, verify, install, site, deploy) along with their corresponding Maven plugin details.
- Code Editor:** The main area shows the content of `pom.xml`, which defines the project's dependencies, including Lombok and various Spring Boot starters.
- Bottom Bar:** Shows tabs for "Maven" and "Git".

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>jr.dev</groupId>
  <artifactId>FlashCash</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>FlashCash</name>
  <description>Spring Flash Cash</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
  </dependencies>

```

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections, environments, flows, and histories. The main workspace displays a POST request to 'localhost:8070/user/create'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "username": "{{$randomFirstName}}",  
3   "password": "{{$randomLastName}}",  
4   "role": "user"  
5 }
```

The response section shows a successful '201 Created' status with a response time of 381 ms and a body size of 232 B. The response JSON is:

```
1 {  
2   "id": 1,  
3   "username": "Colby",  
4   "password": "Thompson",  
5   "role": "user"  
6 }
```

At the bottom, there are navigation links for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Postman

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'Local instance MySQL80'. The main area displays a query results grid for a table named 'user' with one row of data: id=1, email=fred@gmail.com, firstname=Fred, lastname=Fred, password=\$2a\$10\$QK6muTPUx0pESC..., account_id=NULL. Below the grid is a SQL history pane showing numerous database operations, mostly SELECT and INSERT statements, with timestamps ranging from 14:06:32 to 10:47:38. The schema selected is 'flashcash'.

Mysql workbench

3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie sous la supervision de mon formateur pour l'installation et la configuration des outils de travail.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - FlashCash

Période d'exercice - Du 10/12/2024 au 10/01/2025

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°2 - Développer des interfaces utilisateur

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

Dans le cadre du projet FlashCash, j'ai été amenée à concevoir et développer des interfaces utilisateur destinées à faciliter l'interaction entre l'usager et l'application. J'ai tout d'abord créé des templates HTML permettant à l'utilisateur de remplir des formulaires pour diverses opérations (inscription, connexion, création de compte, ajout de patients, etc.).

Ces interfaces sont reliées à des contrôleurs Spring Boot via des mappings HTTP (@GetMapping pour l'affichage, @PostMapping pour la soumission de données), afin de gérer le cycle complet des interactions : affichage, saisie, traitement, et retour utilisateur.

Les formulaires ont été pensés pour transmettre les données au back-end sous forme de JSON, automatiquement converti en objets Java grâce à la sérialisation/désérialisation assurée par Spring Boot.

J'ai accordé une attention particulière à la cohérence UX/UI, en m'appuyant sur des principes simples d'ergonomie : clarté des messages, limitation du nombre de champs, visibilité des erreurs de saisie, et réutilisation de composants communs (boutons, alertes, titres...) dans l'optique d'une expérience utilisateur fluide et intuitive.

Les pages ont été développées en conformité avec la charte graphique définie en début de projet, et les formulaires ont intégré des règles de validation des entrées, côté front et/ou côté back. Enfin, j'ai testé manuellement chacune des interfaces à l'aide de jeux d'essai réels directement depuis l'interface.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

Pour construire les interfaces, j'ai utilisé le langage HTML afin de structurer les pages et les formulaires, tout en respectant les bonnes pratiques de sémantique et d'accessibilité. J'ai veillé à attribuer correctement les labels aux champs de formulaire, et à utiliser une hiérarchie de titres cohérente, dans l'esprit du référentiel RGAA (Référentiel Général d'Amélioration de l'Accessibilité).

Pour le design, j'ai utilisé le framework Bootstrap (version MDB), qui m'a permis de styliser les composants de l'interface tout en assurant une mise en page responsive, adaptée à tous les types d'écrans (ordinateur, tablette, mobile).

Côté intégration avec le back-end, j'ai travaillé avec Thymeleaf, un moteur de templates compatible avec Spring Boot. Il m'a permis d'insérer dynamiquement des données dans les pages HTML (affichage de listes, messages d'erreur, champs préremplis...), en m'appuyant sur ses balises (*th:text*, *th:if*, *th:each*, etc.) qui restent proches de la syntaxe HTML classique.

J'ai également utilisé JavaScript pour améliorer l'expérience utilisateur. En manipulant le DOM (Document Object Model), j'ai mis en place, dans le projet *HealthCare*, une modale (popup) qui s'affiche automatiquement lors de la modification d'une note. Cette modale est déclenchée via un écouteur d'événement attaché au chargement de la page ou à un clic, ce qui permet une interaction fluide sans recharger toute l'interface.

Les formulaires ont été reliés à la logique métier via les annotations Spring Boot (@GetMapping, @PostMapping). Les données saisies sont automatiquement injectées dans des objets Java via le binding Spring, ce qui facilite leur traitement côté serveur.

Enfin, j'ai intégré les principes de sécurité web, notamment en validant systématiquement les entrées côté serveur pour me prémunir contre les attaques XSS (Cross-Site Scripting) ou les CSRF (Cross-Site Request Forgery). J'ai également veillé à l'affichage des mentions légales et à la prise en compte des obligations liées au RGPD (Règlement Général sur la Protection des Données), notamment dans les interfaces liées à la création de comptes utilisateurs.

DOSSIER PROFESSIONNEL (DP)

[LOGOUT](#)[ACCOUNT FEATURES](#)

WELCOME TO FLASHCASH

Fred Fred

Current Balance

158955.625

[CASH TRANSFERS](#)[ACCOUNT OPTIONS](#)

HomePage

A screenshot of a code editor showing the template for add-to-flashcash.html. The code is written in Thymeleaf and HTML. It includes a form for depositing cash, with fields for amount and submit/cancel buttons.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>Add To FlashCash Form</title>
        <link th:href = "@{/css/mdb.min.css}" rel="stylesheet"/>
        <link th:href = "@{/css/index.css}" rel="stylesheet" />
    </head>
    <body class = "signin-form" >
        <section class="vh-100" style="background-color: #e5f2ff;">
            <div class="container py-5 h-100">
                <div class="row d-flex justify-content-center align-items-center h-100">
                    <div class="col-12 col-md-8 col-lg-6 col-xl-5">
                        <div class="card shadow-2-strong" style="border-radius: 1rem; border: none; border-bottom: 1px solid #f5f5f5; padding: 1.5rem; margin-bottom: 1.5rem;">
                            <div class="card-body p-5 text-center">
                                <div class="logon-card">
                                    
                                </div>
                                <h2 class="card-title text-center mb-5 fw-bold fs-5">Deposit your Cash </h2>
                                <form th:action="@{/add-to-flashcash}" th:object="${addToFlashCashForm}" method="POST">
                                    <div data-mdb-input-init class="form-outline mb-4">
                                        <input type="number" th:field="#{amount}" id="floatingInputAmount" class="form-outline form-control-color" placeholder="Amount" step="0.01" />
                                    </div>
                                    <button data-mdb-button-init data-mdb Ripple-init class="btn btn-primary btn-lg btn-block" type="submit">Deposit your cash</button>
                                    <button href="/home" type="button" onclick="window.location.href='/home'" data-mdb-button-init data-mdb Ripple-init class="btn btn-danger btn-lg btn-block">Cancel</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>
    </body>
</html>
```

HomePage - template

DOSSIER PROFESSIONNEL (DP)



ACCOUNT SETTINGS

[HOME](#)

Balance
158955.625

Iban
DE89370400440532013000

Connection List

Firstname	Lastname
toto	toto

[REMOVE CONNECTION](#)

Transfer List

ID	Amount Before Fee	Amount After Fee	Date	Description	Receiver

Template - Account Settings

The screenshot shows a Java code editor with the file `TransferController.java` open. The code defines a controller for handling cash transfers. It imports necessary packages, initializes services, and defines two main methods: `showCashForm()` and `addCashToAccount()`. The `showCashForm()` method returns a `ModelAndView` with view name "add-to-flashcash" and model name "addToFlashCashForm". The `addCashToAccount()` method processes a form submission, logs the action, checks for errors, transfers cash to an account, and returns a `ModelAndView` with view name "account" and model name "user". It also handles a withdrawal endpoint `/cash-to-bank`.

```
package jr.dev.FlashCash.controller;

import ...;

@Controller  * jacques-roulet*
@RequiredArgsConstructor
@RequestMapping(*)
public class TransferController {

    private static final Logger logger = LoggerFactory.getLogger(TransferController.class);  4 usages

    private final LinkService linkService;
    private final TransferService transferService;
    private final SessionService sessionService;

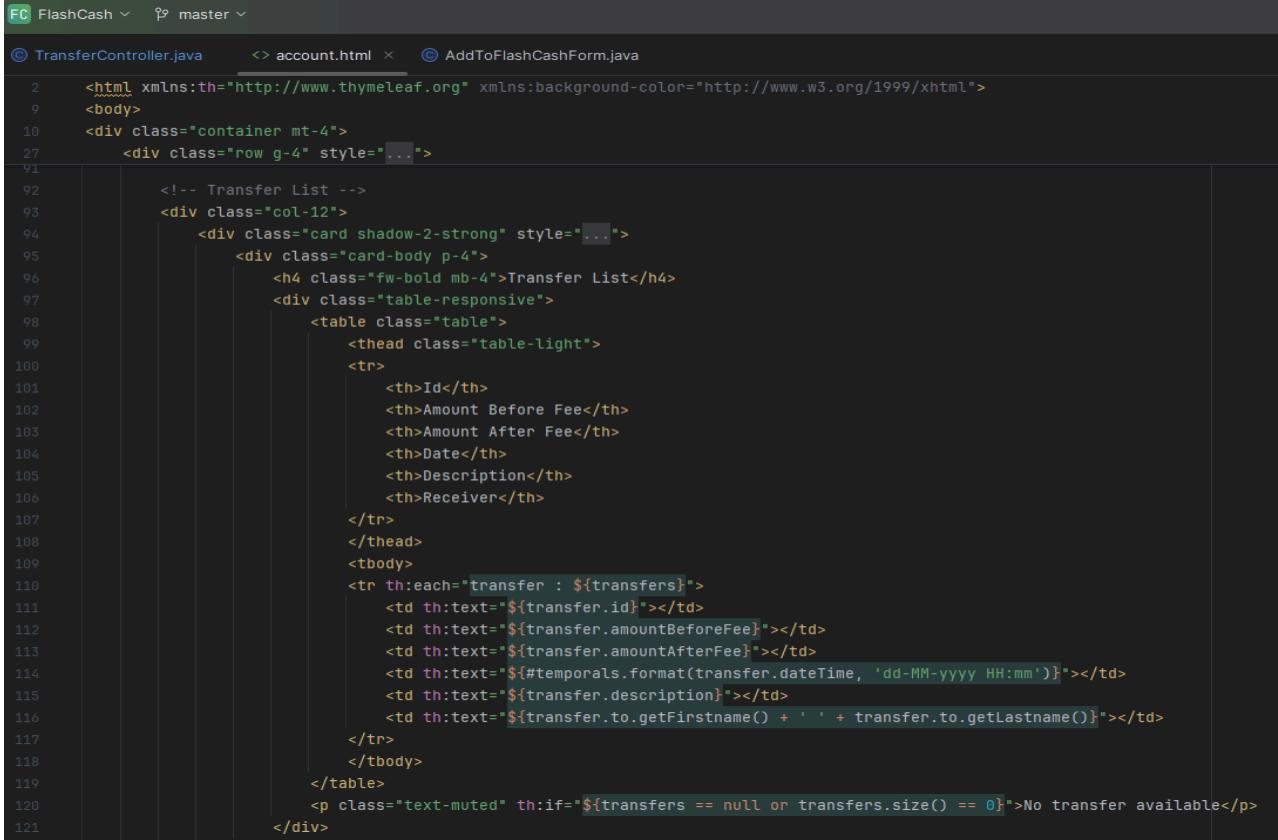
    @GetMapping(*"/add-to-flashcash")  * jacques-roulet
    public ModelAndView showCashForm(){
        logger.info("Cash adding display form");
        return new ModelAndView(  viewName: "add-to-flashcash",  modelName: "addToFlashCashForm",  new AddToFlashCashForm());
    }

    @PostMapping(*"/add-to-flashcash")  * jacques-roulet
    public ModelAndView addCashToAccount(
        @ModelAttribute("addToFlashCashForm") AddToFlashCashForm form,
        BindingResult result
    {
        logger.info("Adding cash");
        if(result.hasErrors()){
            return new ModelAndView(  viewName: "add-to-flashcash",  modelName: "addToFlashCashForm",  form);
        }
        transferService.transferCashToAccount(form);
        User user = sessionService.sessionUser();
        return new ModelAndView (  viewName: "account",  modelName: "user",  user);
    }

    @GetMapping(*"/cash-to-bank")  * jacques-roulet
    public ModelAndView showWithdrawCashForm(Model model){
        logger.info("Withdraw display form");
        String linkedIban = transferService.findIban();
        model.addAttribute( attributeName: "linkedIban",  linkedIban);
        return new ModelAndView(  viewName: "cash-to-bank",  modelName: "cashToBankForm",  new CashToBankForm());
    }
}
```

Contrôleur - formulaires - endpoints

DOSSIER PROFESSIONNEL (DP)

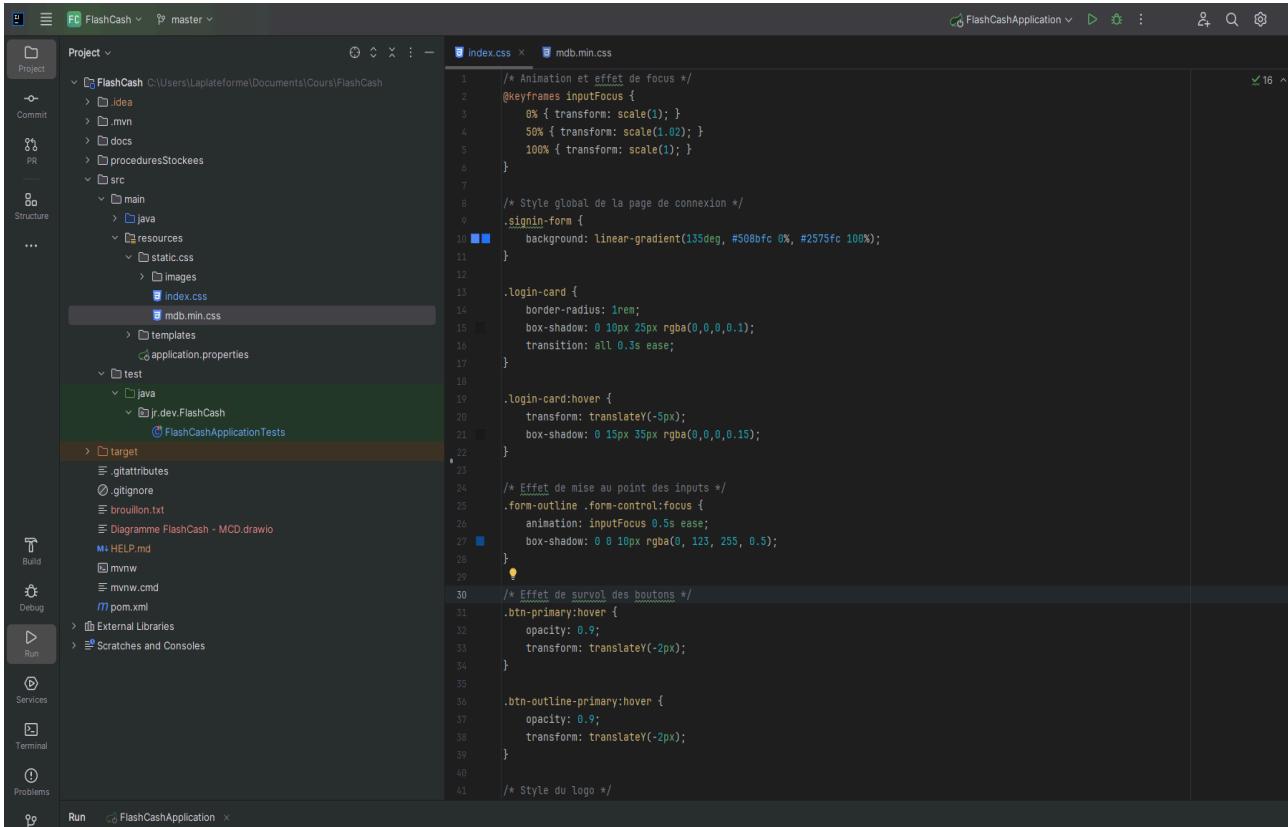


The screenshot shows a code editor with a dark theme. The title bar says "FC FlashCash master". The editor has three tabs: "TransferController.java", "account.html", and "AddToFlashCashForm.java". The "account.html" tab is active and contains the following Thymeleaf template code:

```
2 <html xmlns:th="http://www.thymeleaf.org" xmlns:background-color="http://www.w3.org/1999/xhtml">
9 <body>
10 <div class="container mt-4">
27   <div class="row g-4" style="...">
91     <!-- Transfer List -->
92     <div class="col-12">
93       <div class="card shadow-2-strong" style="...">
94         <div class="card-body p-4">
95           <h4 class="fw-bold mb-4">Transfer List</h4>
96           <div class="table-responsive">
97             <table class="table">
98               <thead class="table-light">
99                 <tr>
100                   <th>Id</th>
101                   <th>Amount Before Fee</th>
102                   <th>Amount After Fee</th>
103                   <th>Date</th>
104                   <th>Description</th>
105                   <th>Receiver</th>
106                 </tr>
107               </thead>
108               <tbody>
109                 <tr th:each="transfer : ${transfers}">
110                   <td th:text="${transfer.id}"></td>
111                   <td th:text="${transfer.amountBeforeFee}"></td>
112                   <td th:text="${transfer.amountAfterFee}"></td>
113                   <td th:text="#{temporals.format(transfer.dateTime, 'dd-MM-yyyy HH:mm')}"></td>
114                   <td th:text="${transfer.description}"></td>
115                   <td th:text="${transfer.to.getFirstname() + ' ' + transfer.to.getLastname()}"></td>
116                 </tr>
117               </tbody>
118             </table>
119             <p class="text-muted" th:if="${transfers == null or transfers.size() == 0}">No transfer available</p>
120           </div>
121     </div>
```

Thymeleaf - template liste des transferts

DOSSIER PROFESSIONNEL (DP)



The screenshot shows the IntelliJ IDEA interface with the 'FlashCash' project open. The left sidebar displays the project structure, including modules like 'FlashCash', 'Idea', 'mvn', 'docs', 'proceduresStockees', and 'src'. The 'src' module contains 'main', 'resources', 'test', and 'target' sub-folders. Under 'resources', there are 'static.css' and 'images' folders, which contain 'index.css' and 'mdb.min.css'. The right panel shows the content of the 'index.css' file, which includes CSS rules for various UI components like login forms and buttons, utilizing Material Design for Bootstrap (MDB) styles.

```
/* Animation et effet de focus */
@keyframes inputFocus {
    0% { transform: scale(1); }
    50% { transform: scale(1.02); }
    100% { transform: scale(1); }
}

/* Style global de la page de connexion */
.sigin-form {
    background: linear-gradient(135deg, #5088fc 0%, #2575fc 100%);
}

.login-card {
    border-radius: 1rem;
    box-shadow: 0 10px 25px rgba(0,0,0,0.1);
    transition: all 0.3s ease;
}

.login-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 15px 35px rgba(0,0,0,0.15);
}

/* Effet de mise au point des inputs */
.form-outline .form-control:focus {
    animation: inputFocus 0.5s ease;
    box-shadow: 0 0 10px rgba(0, 123, 255, 0.5);
}

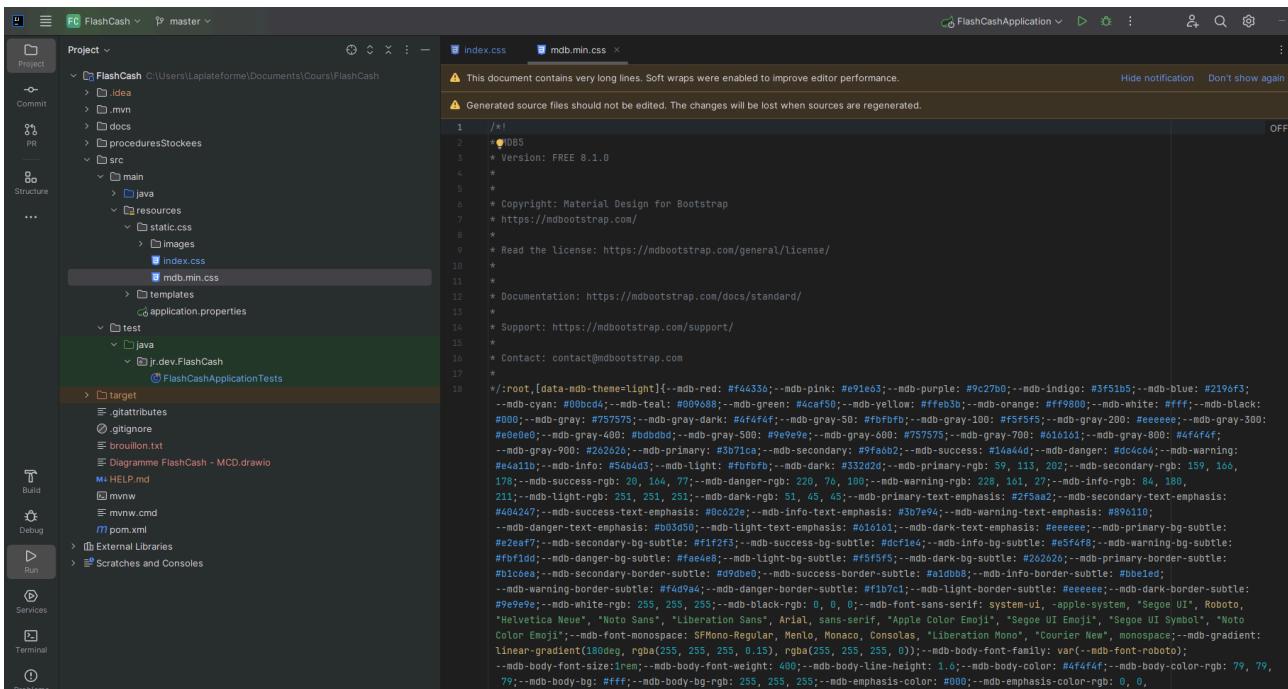
/* Effet de survol des boutons */
.btn-primary:hover {
    opacity: 0.9;
    transform: translateY(-2px);
}

.btn-outline-primary:hover {
    opacity: 0.9;
    transform: translateY(-2px);
}

/* Style du logo */

```

Style - Index.css



The screenshot shows the IntelliJ IDEA interface with the 'FlashCash' project open. The left sidebar displays the project structure, including modules like 'FlashCash', 'Idea', 'mvn', 'docs', 'proceduresStockees', and 'src'. The 'src' module contains 'main', 'resources', 'test', and 'target' sub-folders. Under 'resources', there are 'static.css' and 'images' folders, which contain 'index.css' and 'mdb.min.css'. The right panel shows the content of the 'mdb.min.css' file, which is a large, highly compressed CSS file containing numerous styles for various UI components, primarily using the Material Design for Bootstrap (MDB) framework.

```
/*
 * This document contains very long lines. Soft wraps were enabled to improve editor performance.
 * Generated source files should not be edited. The changes will be lost when sources are regenerated.
 */

/*
 * Read the license: https://mdbootstrap.com/general/license/
 *
 * Documentation: https://mdbootstrap.com/docs/standard/
 *
 * Support: https://mdbootstrap.com/support/
 *
 * Contact: contact@mdbootstrap.com
 */

/*
 * [data-mdb-theme=light]--> .mdb-pink: #f6436d;--> .mdb-purple: #9c27b0;--> .mdb-indigo: #3f51b5;--> .mdb-blue: #2196f3;--> .mdb-cyan: #00bcd4;--> .mdb-teal: #009688;--> .mdb-green: #4caf50;--> .mdb-yellow: #ffeb3b;--> .mdb-orange: #ff9800;--> .mdb-white: #fff;--> .mdb-black: #000;--> .mdb-gray: #757575;--> .mdb-gray-dark: #4f4f4f;--> .mdb-gray-50: #f0f0f0;--> .mdb-gray-100: #f5f5f5;--> .mdb-gray-200: #eeeeee;--> .mdb-gray-300: #e0e0e0;--> .mdb-gray-400: #bdbdbd;--> .mdb-gray-500: #777777;--> .mdb-gray-600: #616161;--> .mdb-gray-700: #616161;--> .mdb-gray-800: #4f4f4f;--> .mdb-gray-900: #262626;--> .mdb-primary: #3071ca;--> .mdb-secondary: #f0a0a2;--> .mdb-success: #14a44a;--> .mdb-danger: #c4cae4;--> .mdb-warning: #e0a1b0;--> .mdb-info: #5cb85c;--> .mdb-light: #fbfbfb;--> .mdb-dark: #332222;--> .mdb-primary-rgb: 59, 113, 202;--> .mdb-secondary-rgb: 159, 166, 178;--> .mdb-success-rgb: 20, 164, 77;--> .mdb-danger-rgb: 220, 70, 100;--> .mdb-warning-rgb: 228, 161, 27;--> .mdb-info-rgb: 84, 180, 211;--> .mdb-light-rgb: 251, 251, 251;--> .mdb-dark-rgb: 51, 45, 45;--> .mdb-primary-text-emphasis: #2f5fa2;--> .mdb-secondary-text-emphasis: #404247;--> .mdb-success-text-emphasis: #00c02e;--> .mdb-info-text-emphasis: #3b7e94;--> .mdb-warning-text-emphasis: #969610;--> .mdb-danger-text-emphasis: #003d50;--> .mdb-light-text-emphasis: #d10101;--> .mdb-dark-text-emphasis: #eeeeee;--> .mdb-primary-bg-subtle: #e0e0e0;--> .mdb-secondary-bg-subtle: #f0f0f0;--> .mdb-success-bg-subtle: #dcf1e4;--> .mdb-info-bg-subtle: #e5f4f8;--> .mdb-warning-bg-subtle: #ffefdd;--> .mdb-danger-bg-subtle: #f4e0e0;--> .mdb-light-border-subtle: #ff97c1;--> .mdb-primary-border-subtle: #b1cbea;--> .mdb-secondary-border-subtle: #a1d8d8;--> .mdb-success-border-subtle: #a1d8d8;--> .mdb-info-border-subtle: #bfebd5;--> .mdb-warning-border-subtle: #f4d9d4;--> .mdb-danger-border-subtle: #f1f7c1;--> .mdb-light-border-subtle: #eeeeee;--> .mdb-dark-border-subtle: #e6e6e6;--> .mdb-white-rgb: 255, 255, 255;--> .mdb-black-rgb: 0, 0, 0;--> .mdb-font-sans-serif: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", "Noto Sans", "Liberation Sans", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";--> .mdb-font-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;--> .mdb-gradient: linear-gradient(180deg, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0));--> .mdb-body-font-family: var(--mdb-font-robot);--> .mdb-body-font-size: 1rem;--> .mdb-body-font-weight: 400;--> .mdb-body-line-height: 1.6;--> .mdb-body-color: #4f4f4f;--> .mdb-body-color-rgb: 79, 79, 79;--> .mdb-body-bg: #fff;--> .mdb-body-bg-rgb: 255, 255, 255;--> .mdb-emphasis-color: #000;--> .mdb-emphasis-color-rgb: 0, 0, 0;
```

Style - mdb.min.css

DOSSIER PROFESSIONNEL (DP)

```
package jr.dev.FlashCash.controller;

> import ...;

@Controller
@RequiredArgsConstructor
public class HomeController {

    private final SessionService sessionService;
    private final LinkService linkService;
    private final TransferService transferService;

    @GetMapping("/home")
    public String home(Model model) {

        User user = sessionService.sessionUser();
        List<Link> links = linkService.getLinksForUser(user);
        List <Transfer> transfers = transferService.findTransactions();

        model.addAttribute("links", links);
        model.addAttribute("transfers", transfers);
        model.addAttribute("user", user);
        return "home";
    }

    @GetMapping("/account")
    public String getAccount(Model model) {

        User user = sessionService.sessionUser();
        List<Link> links = linkService.getLinksForUser(user);
        List <Transfer> transfers = transferService.findTransactions();

        model.addAttribute("user", user);
        model.addAttribute("links", links);
        model.addAttribute("transfers", transfers);
        return "account";
    }
}
```

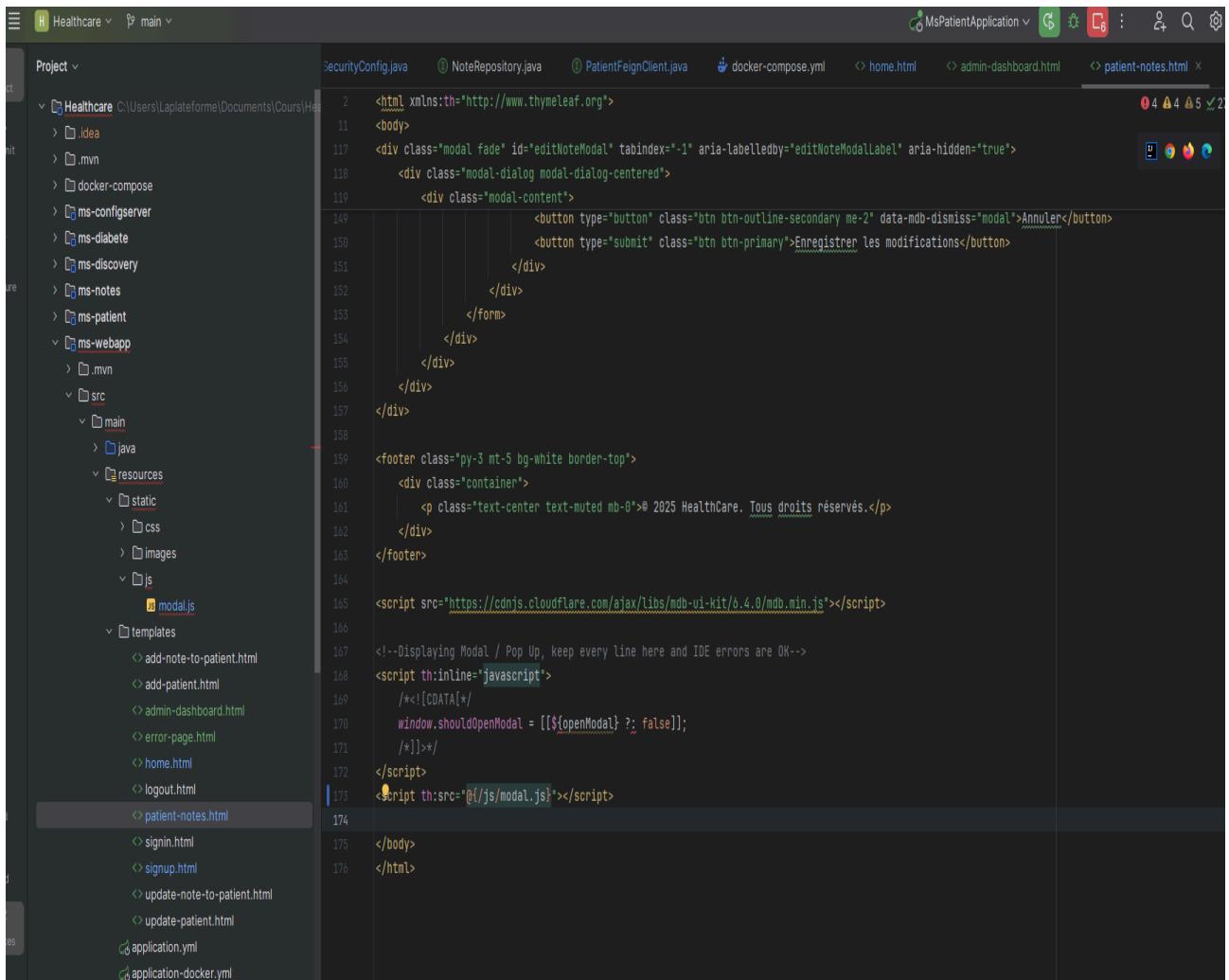
Contrôleur affichage dashboards

```
js modal.js x © AuthenticationService.java     ⓘ PatientFeignClient.java     © mspatient...|User.java     <> home.html
1  document.addEventListener('DOMContentLoaded', listener: function () : void {
2      const shouldOpen : boolean = [[${openModal} ?; false]];
3      console.log(">> openModal =", shouldOpen); //💡 To check in navigator
4      if (shouldOpen) {
5          const modalElement : HTMLElement = document.getElementById('editNoteModal');
6          const modalInstance : mdb.Modal = new mdb.Modal(modalElement);
7          modalInstance.show();
8      }
9  });


```

Modal - JS event

DOSSIER PROFESSIONNEL (DP)

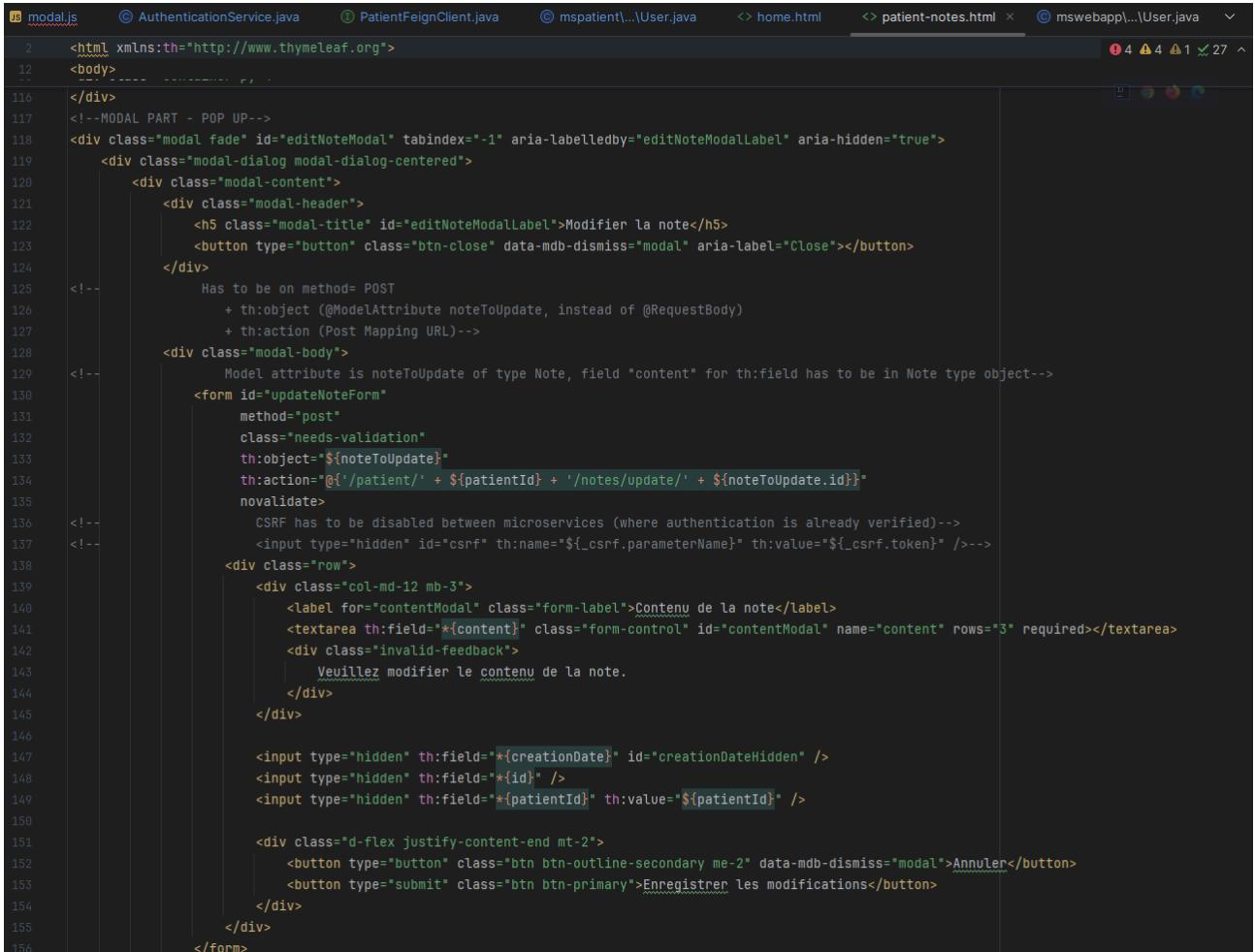


The screenshot shows a Java-based web application project named "Healthcare" in an IDE. The project structure includes modules like ms-configserver, ms-diabète, ms-discovery, ms-notes, ms-patient, and ms-webapp. The ms-webapp module contains a "src" directory with "main", "java", "resources", "static", "css", "images", "js", and "templates" sub-directories. The "templates" directory contains several Thymeleaf templates such as add-note-to-patient.html, add-patient.html, admin-dashboard.html, error-page.html, home.html, logout.html, patient-notes.html, signin.html, signup.html, update-note-to-patient.html, and update-patient.html. The "patient-notes.html" template is currently selected in the code editor. The code in this template uses Thymeleaf syntax to generate HTML. It includes a modal dialog for editing notes, with buttons for "Annuler" (Cancel) and "Enregistrer les modifications" (Save changes). The footer of the page contains a copyright notice for 2025 HealthCare. The code also imports a script from a CDN and includes inline JavaScript to handle modal opening logic.

```
2     <html xmlns:th="http://www.thymeleaf.org">
11    <body>
117      <div class="modal fade" id="editNoteModal" tabindex="-1" aria-labelledby="editNoteModalLabel" aria-hidden="true">
118        <div class="modal-dialog modal-dialog-centered">
119          <div class="modal-content">
149            <button type="button" class="btn btn-outline-secondary me-2" data-mdb-dismiss="modal">Annuler</button>
150            <button type="submit" class="btn btn-primary">Enregistrer les modifications</button>
151          </div>
152        </div>
153      </form>
154    </div>
155  </div>
156</div>
157</div>
158
159<footer class="py-3 mt-5 bg-white border-top">
160  <div class="container">
161    <p class="text-center text-muted mb-0">© 2025 HealthCare. Tous droits réservés.</p>
162  </div>
163</footer>
164
165<script src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/6.4.0/mdb.min.js"></script>
166
167<!--Displaying Modal / Pop Up, keep every line here and IDE errors are OK-->
168<script th:inline="javascript">
169  /*<![CDATA[*/
170  window.shouldOpenModal = [[${openModal} ? true : false]];
171  /*]]>*/
172</script>
173<script th:src="@{/js/modal.js}"></script>
174
175</body>
176</html>
```

Modal - import du script js dans la template

DOSSIER PROFESSIONNEL (DP)



A screenshot of a code editor showing a Thymeleaf modal template named 'modal.js'. The code is a modal dialog for updating a note. It includes a header with a title 'Modifier la note', a body with a text area for content, and a form with hidden fields for creation date, ID, and patient ID. It also includes buttons for canceling or saving changes.

```
1 js modal.js ① AuthenticationService.java ② PatientFeignClient.java ③ mspatient...\\User.java ④ home.html ⑤ patient-notes.html ✘ ⑥ mswebapp...\\User.java ⑦
2 <html xmlns:th="http://www.thymeleaf.org">
3 <body>
4
5 </div>
6 <!--MODAL PART - POP UP-->
7 <div class="modal fade" id="editNoteModal" tabindex="-1" aria-labelledby="editNoteModalLabel" aria-hidden="true">
8   <div class="modal-dialog modal-dialog-centered">
9     <div class="modal-content">
10       <div class="modal-header">
11         <h5 class="modal-title" id="editNoteModalLabel">Modifier la note</h5>
12         <button type="button" class="btn-close" data-mdb-dismiss="modal" aria-label="Close"></button>
13       </div>
14     <!--
15       Has to be on method= POST
16       + th:object (@ModelAttribute noteToUpdate, instead of @RequestBody)
17       + th:action (Post Mapping URL)-->
18     <div class="modal-body">
19       Model attribute is noteToUpdate of type Note, field "content" for th:field has to be in Note type object-->
20     <form id="updateNoteForm"
21       method="post"
22       class="needs-validation"
23       th:object="${noteToUpdate}"
24       th:action="@{/patient/' + ${patientId} + '/notes/update/' + ${noteToUpdate.id}}"
25       novalidate>
26       CSRF has to be disabled between microservices (where authentication is already verified)-->
27       <input type="hidden" id="csrf" th:name="${_csrf.parameterName}" th:value="${_csrf.token}" />-->
28     <div class="row">
29       <div class="col-md-12 mb-3">
30         <label for="contentModal" class="form-label">Contenu de la note</label>
31         <textarea th:field="*{content}" class="form-control" id="contentModal" name="content" rows="3" required></textarea>
32         <div class="invalid-feedback">
33           Veuillez modifier le contenu de la note.
34         </div>
35       </div>
36
37       <input type="hidden" th:field="*{creationDate}" id="creationDateHidden" />
38       <input type="hidden" th:field="*{id}" />
39       <input type="hidden" th:field="*{patientId}" th:value="${patientId}" />
40
41       <div class="d-flex justify-content-end mt-2">
42         <button type="button" class="btn btn-outline-secondary me-2" data-mdb-dismiss="modal">Annuler</button>
43         <button type="submit" class="btn btn-primary">Enregistrer les modifications</button>
44       </div>
45     </div>
46   </form>
47 </div>
48
49
50
51
52
53
54
55
56
```

Thymeleaf - modal template

3. Avec qui avez-vous travaillé ?

J'ai principalement travaillé en autonomie sur cette partie, tout en sollicitant régulièrement des retours de mon formateur et de mes camarades lors des phases de revue. J'ai également pris l'initiative de tester l'interface auprès de personnes extérieures au projet (famille, amis non-développeurs) afin de recueillir des retours sur l'intuitivité et la clarté des parcours utilisateurs.

DOSSIER PROFESSIONNEL^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association	-	La Plateforme
Chantier, atelier, service	-	FlashCash
Période d'exercice	-	Du 10/12/2024 au 10/01/2025

5. Informations complémentaires (*facultatif*)

Annexe:

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°3 - *Développer des composants métiers*

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

Au cours de ma formation, j'ai développé des composants métiers dans le cadre de l'application FlashCash.

FlashCash est une application de gestion de transactions financières, permettant d'effectuer différentes opérations telles que : ajout de fonds, virement entre utilisateurs, ou transfert vers une banque externe.

Pour ce projet, j'ai mis en place une architecture MVC (Modèle-Vue-Contrôleur), en respectant le cahier des charges fourni par le formateur.

Dans ce contexte, j'ai conçu des modèles de données (appelés aussi entités) pour représenter les objets enregistrés en base de données.

Une classe en Java est une structure qui permet de modéliser un concept métier (ex. : un utilisateur, un compte, une transaction).

Chaque objet est une instance concrète de cette classe, avec des valeurs propres à un cas d'usage précis. Par exemple, la classe User définit ce qu'est un utilisateur (nom, email, mot de passe...), et chaque utilisateur créé dans l'application devient un objet User.

J'ai développé des composants métiers au sein de la couche service, afin d'y centraliser la logique fonctionnelle : calculs de montants, vérifications de soldes disponibles, règles de gestion liées aux types de transfert...

Ces services ont été conçus pour être réutilisables et testables, et communiquent avec les couches controller (pour l'interface API) et repository (pour l'accès aux données). J'ai également intégré une couche de sécurité :

- Gestion des utilisateurs via Spring Security (authentification, sessions).
- Mise en place de validateurs (standard ou personnalisés) pour garantir la qualité des données lors de la création d'un compte utilisateur.

Enfin, j'ai documenté chaque méthode de la couche service à l'aide de commentaires explicites, afin de faciliter la relecture du code.

DOSSIER PROFESSIONNEL^(DP)

2. Précisez les moyens utilisés :

- Architecture en couches : Controller / Service / Repository
- Utilisation de DTO pour sécuriser les échanges de données
- Création de méthodes spécifiques exposées via API REST
- Rédaction de tests unitaires pour garantir la fiabilité du code
- Utilisation du mode debug pour l'identification des erreurs
- Respect des conventions de nommage (camelCase)
- Intégration de Spring Security pour l'authentification et les autorisations
- Mise en place de validateurs pour le contrôle des champs utilisateur

DOSSIER PROFESSIONNEL (DP)

```
public class TransferService {

    private final UserAccountRepository userAccountRepository;
    private final SessionService sessionService;
    private final TransferRepository transferRepository;
    private final UserRepository userRepository;

    public void transferCashToAccount(AddToFlashCashForm form) { 1 usage à jacques-roulet
        if(form != null){
            userAccountRepository.save(sessionService.sessionUser().getAccount().plus(form.getAmount()));
        }
    }

    public String transfer(TransferForm form) { // TO COMPLETE WITH ERROR MESSAGE ON CONSTRAINT 1 usage à jacques-roulet

        if(form!= null){
            User to = userRepository.findUserByEmail(form.getContactEmail())
                .orElseThrow(()-> new RuntimeException("User with email not found"));

            // fill transfer with information
            Transfer currentTransfer = new Transfer();
            currentTransfer.setFrom(sessionService.sessionUser()); // Gets whole user
            currentTransfer.setTo(to); // Gets whole user
            currentTransfer.setDateTime(LocalDateTime.now());
            currentTransfer.setDescription(form.getDescription());
            currentTransfer.setAmountBeforeFee(form.getAmount());
            currentTransfer.setAmountAfterFee(applyFee(form.getAmount()));

            if(sessionService.sessionUser().getAccount().getAmount() - form.getAmount() > 0){
                // Update Sender & Receiver accounts amounts and save in repo
                userAccountRepository.save(sessionService.sessionUser().getAccount().minus(currentTransfer.getAmountAfterFee())); // Get the transfer amount >> form amount
                userAccountRepository.save(to.getAccount().plus(currentTransfer.getAmountAfterFee()));
                transferRepository.save(currentTransfer);
                // Retour succès
                return "Transfer successful!";
            }
            else {
                // Retour échec si le solde est insuffisant
                return "You need more money";
            }
        }
        return "Invalid transfer form";
    }
}
```

Couche Service (couche métier) - usage Dto

DOSSIER PROFESSIONNEL (DP)

```
@Controller @jacques-roulet
@RequiredArgsConstructor
public class HomeController {

    private final SessionService sessionService;
    private final LinkService linkService;
    private final TransferService transferService;

    @GetMapping("/home") @jacques-roulet
    public String home(Model model) {

        User user = sessionService.sessionUser();
        List<Link> links = linkService.getLinksForUser(user);
        List <Transfer> transfers = transferService.findTransactions();

        model.addAttribute(attributeName: "links", links);
        model.addAttribute(attributeName: "transfers", transfers);
        model.addAttribute(attributeName: "user", user);
        return "home";
    }

    @GetMapping("/account") @jacques-roulet
    public String getAccount(Model model) {

        User user = sessionService.sessionUser();
        List<Link> links = linkService.getLinksForUser(user);
        List <Transfer> transfers = transferService.findTransactions();

        model.addAttribute(attributeName: "user", user);
        model.addAttribute(attributeName: "links", links);
        model.addAttribute(attributeName: "transfers", transfers);
        return "account";
    }
}
```

Création de méthodes spécifiques exposées via API REST - Structure MVC - Template d'affichage

DOSSIER PROFESSIONNEL (DP)

```
1 package jr.dev.FlashCash.interfaces.repository;
2
3 > import ...
4
5
6 @Repository 2 usages  ± jacques-roulet
7 public interface TransferRepository extends JpaRepository<Transfer, Integer> {
8     //    List<Transfer> findTransfersByFrom(User user);
9
10    // Stocked Procedure
11    @Query(value = "CALL PS_TransfersByUser(:userId)", nativeQuery = true) 1 usage  ± jacques-roulet
12    List<Transfer> returnTransfers(Integer userId);
13}
```

Couche repository - accès aux données - Procédure stockée

```
1 package jr.dev.FlashCash.interfaces.repository;
2
3 > import ...
4
5
6 @Repository 3 usages  ± jacques-roulet
7 public interface LinkRepository extends JpaRepository<Link, Integer> {
8
9     //    List<Link> findLinksByUser1Id(Integer id);
10
11     // Using a stocked procedure
12     @Query(value= "CALL PS_Links(:user1Id, :user2Id)", nativeQuery = true) 2 usages  ± jacques-roulet
13     List<Link> findLinkByUser1AndUser2StoredProcedure(@Param("user1Id") int user1Id, @Param("user2Id") int user2Id);
14
15
16     @Query("SELECT l FROM Link l WHERE l.user1.id = :userId") 2 usages  ± jacques-roulet
17     List<Link> findLinksByUserId(@Param("userId") Integer userId);
18
19 }
```

Couche repository - Requête paramétrée

DOSSIER PROFESSIONNEL (DP)

```
package jr.dev.FlashCash.config;

> import ...;

@Configuration + jacques-roulet
public class SpringSecurityConfig {

    @Bean + jacques-roulet
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean + jacques-roulet
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(( AuthorizationManagerRequestMat... requests) -> requests
                .requestMatchers( @ "/signin", @ "/authentication", @ "/signup", @ "/static/**", @ "/css/**", @ "/images/**") AuthorizedUrl
                .permitAll() AuthorizationManagerRequestMat...
                .anyRequest() AuthorizedUrl
                .authenticated()
            )
            .formLogin(( FormLoginConfigurer<HttpSecurity> form) -> form
                // customized login form
                .loginPage("/signin")
                // loginProcessingUrl("/authentication") // To go through your postmapping /authentication with spring security own means
                // username is default field and is changed here (> checked field in the HTML form login page)
                .permitAll().usernameParameter("email")
                // when successful, goes to /* URL, => always
                .defaultSuccessUrl( defaultSuccessUrl: "/home", alwaysUse: true)
                // Add an error code to the URL when login fails
                .failureUrl( authenticationFailureUrl: "/signin?authError=true")
            )
            // logout access
            .logout(( LogoutConfigurer<HttpSecurity> logout) -> logout
                .logoutUrl("/logout")
                .permitAll());
    }

    return http.build();
}
}
```

Intégration de Spring Security pour l'authentification et les autorisations

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure for "FlashCash" located at C:\Users\laplateforme\Documents\Code\FC. It includes .idea, .mvn, docs, proceduresStockees, src, main, java, jr.dev.FlashCash, config, controller, exceptions, interfaces, and repository.
- Code Editor:** Displays the content of `HomeController.java`. The code defines a `HomeController` class with a `@GetMapping("/home")` annotation. The method `home` retrieves the current user from `sessionService`, gets links from `linkService`, and finds transfers from `transferService`. It then adds these to a `Model` object and returns it.
- Debug Tool Window:** Shows a stack trace for a thread named "http-nio-8080-exec-3". The stack trace starts with `home:26, HomeController (jr.dev.FlashCash.controller)` and lists 92 hidden frames. The stack trace continues with:

```
    > @Target({ElementType.TYPE, ElementType.FIELD, ElementType.METHOD})\n@Retention(RetentionPolicy.RUNTIME = undefined\n    > this = {HomeController@13564}\n    @model = {BindingAwareModelMap@13568} size = 0\n    > transferService = {TransferService@13567}\n    > linkService = {LinkService@13566}\n    > sessionService = {SessionService@13565}
```

Utilisation du mode debug pour l'identification des erreurs

DOSSIER PROFESSIONNEL (DP)

```
import static org.mockito.Mockito.*;  
  
@SpringBootTest  ^jacques-roulet*  
class FlashCashApplicationTests {  
  
    @Test  ^jacques-roulet  
    void contextLoads() {  
    }  
  
    @Test  new *  
    void transferCashToAccountTest() {  
        // mocks  
        UserAccountRepository userAccountRepository = mock(UserAccountRepository.class);  
        SessionService sessionService = mock(SessionService.class);  
        TransferRepository transferRepository = mock(TransferRepository.class);  
        UserRepository userRepository = mock(UserRepository.class);  
  
        // constructor  
        TransferService service = new TransferService(userAccountRepository, sessionService, transferRepository, userRepository);  
  
        // Arrange  
        AddToFlashCashForm form = new AddToFlashCashForm();  
        form.setAmount(50.00);  
  
        UserAccount userAccount = new UserAccount();  
        userAccount.setAmount(50.00);  
        User user = new User();  
        user.setAccount(userAccount);  
  
        when(sessionService.sessionUser()).thenReturn(user);  
  
        // Act  
        service.transferCashToAccount(form);  
  
        // Assert / verify  
        verify(userAccountRepository).save(argThat( UserAccount acc -> acc.getAmount() == 100.00));  
    }  
}
```

Rédaction de tests unitaires pour garantir la fiabilité du code

DOSSIER PROFESSIONNEL (DP)

```
1 package jr.dev.FlashCash.model;
2
3 > import ...
4
5
6 @Data
7 @Entity
8 public class User {
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private Integer id;
12
13    @Column(unique = true, nullable = false)
14    private String email; //login
15
16    @Column
17    private String firstname;
18    private String lastname;
19
20    @Column
21    private String password;
22
23    > //...
24
25    @NotBlank(message = "Password is mandatory")
26    @StrongPassword
27    private String password;
28
29    @ManyToMany
30    private List<Link> links;
31
32    @OneToOne (cascade = CascadeType.ALL, fetch = FetchType.EAGER)
33    private UserAccount account;
34
35
36
37
38 }
39
```

Mise en place de validateurs pour le contrôle des champs utilisateur

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie tout en sollicitant régulièrement mon formateur, notamment pour améliorer la sécurité des processus d'inscription (confirmation de mot de passe, validation des champs).

J'ai également échangé avec mes camarades de promotion sur les choix d'interfaces et l'ergonomie utilisateur, afin d'adapter les fonctionnalités aux besoins réels.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme

Chantier, atelier, service

▶ FlashCash

Période d'exercice

▶ Du 10/12/2024 au 10/01/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°4 - Contribuer à la gestion d'un projet informatique

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

Dans le cadre du projet FlashCash, une application de gestion de transactions financières, j'ai été amenée à contribuer à la gestion du projet depuis la création, en m'appuyant sur une démarche structurée et collaborative.

J'ai participé à la définition des environnements de développement, en proposant une architecture technique adaptée à notre stack (Spring Boot, MySQL, sécurité avec Spring Security).

Nous avons opté pour une méthode de gestion de projet itérative, inspirée des principes Agile, avec un découpage en sprints hebdomadaires. J'ai assuré la planification des tâches sur l'outil collaboratif Trello, en décomposant les fonctionnalités principales (authentification, gestion des comptes, transferts bancaires) en sous-tâches détaillées, avec estimation de charge, échéance, et priorité.

Un suivi rigoureux de l'avancement était réalisé quotidiennement via Trello, ce qui permettait d'identifier rapidement les retards éventuels et de les signaler lors de nos points d'équipe.

J'ai également mis en place un dépôt GitHub pour assurer le versioning du projet. Les commits étaient réalisés localement au fil de la journée, puis poussés sur le dépôt distant en fin de journée, permettant ainsi un suivi des évolutions clair et centralisé.

2. Précisez les moyens utilisés :

- Méthodologie de gestion itérative (type Agile)
- Outil collaboratif de gestion de projet : Trello
- Planification des tâches et suivi de charge
- Points journaliers (formalisés avec points d'action)
- Communication écrite claire et structurée avec le reste de l'équipe
- Définition d'un environnement de développement cohérent avec les exigences du projet
- Prise en compte des retards ou blocages techniques, avec communication régulière au reste de l'équipe

DOSSIER PROFESSIONNEL (DP)

FlashCash Visible par l'espace de travail Tableau Power-ups Automatisation

The Trello board is organized into five columns:

- À faire**: Contains tasks like "Contrainte unique et différent du user lui même sur ajout du contact pour transfert" and "Remplacer requête SQL par procédure Stockée".
- En cours**: Contains tasks like "Login error message" (with a progress bar at 1/5) and "+ Ajouter une carte".
- Révision du code**: Contains tasks like "Validateur Transfer" (3/3), "Formulaire ajout d'argent", and "Renvoie de l'erreur "email déjà existant lors du sign up" vers la page sign in" (with a progress bar at 5/5).
- Test**: Contains tasks like "Ajout d'une feature { + / - } pour les transferts", "Création d'une classe de gestion des exceptions" (with a progress bar at 5/5), and "Erreur formulaire d'ajout de contact".
- Terminé**: Contains tasks like "Créer Base de Données", "Connecter JPA + Créer les Repositories", "Gestion de l'authentification", "Validateurs" (3/3), "Créer les models", "Formulaire transfert d'argent", "Création Logo", "Formulaire ajout de contact", "Logout", "Vue des transfers, ajout logo", and "migrer les services concernant le cash du accountcontroller vers le transferController".

Trello

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a GitHub repository page for 'FlashCash'. The repository is public and was created by 'JRoulet'. It has 19 commits, 1 star, 1 watch, 0 forks, and 1 pull request. The repository contains files like .mvn/wrapper, docs, src, .gitattributes, .gitignore, mvnw, mvnw.cmd, and pom.xml. A 'README' file is available for addition. The page also includes sections for About, Releases, Packages, Languages, and Suggested workflows.

About
No description, website, or topics provided.

Activity
1 star
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Languages
Java 54.4% • HTML 44.0%
CSS 1.6%

Suggested workflows
Based on your tech stack

SLSA Generic generator
Configure
Generate SLSA3 provenance for your existing release workflows

Git

3. Avec qui avez-vous travaillé ?

Le travail s'est effectué de manière autonome, le formateur jouait le rôle de l'équipe.

Nous faisions des points journaliers sur les tâches à planifier et les blocages ou bugs rencontrés

Une planification hebdomadaire est aussi mise en place en fin de semaine pour la semaine suivante pour fixer les échéances.

Nous avons également collaboré entre étudiants pour échanger sur les bugs rencontrés et les approches pour progresser dans le développement .

DOSSIER PROFESSIONNEL^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association	-	La Plateforme
Chantier, atelier, service	-	FlashCash
Période d'exercice	-	Du 10/12/2024 au 10/01/2025

5. Informations complémentaires (*facultatif*)

Annexes :

- Trello du projet
- Compte GitHub du projet

DOSSIER PROFESSIONNEL^(DP)

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

Exemple n°1 - 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 KundApp, l'analyse initiale a porté sur l'objectif principal : concevoir une plateforme de mise en relation entre des professionnels du bien-être et des clients, permettant la réservation de séances.

J'ai d'abord identifié les acteurs clés (client, professeur, administrateur) et les fonctionnalités attendues pour chaque rôle. Ces besoins ont ensuite été formalisés à travers des user stories, en adoptant le point de vue de l'utilisateur final pour garantir la pertinence de chaque fonctionnalité.

À partir de cette analyse, j'ai réalisé sur Figma les maquettes des écrans principaux de l'application (accueil, liste des séances, profil utilisateur, etc.). La conception de ces interfaces a suivi les bonnes pratiques d'ergonomie (UX/UI) pour assurer un parcours utilisateur simple et intuitif.

2. Précisez les moyens utilisés :

- Utilisation d'un outil de maquettage : Figma

DOSSIER PROFESSIONNEL (DP)



KundApp Admin

admin@gmail.com

DÉCONNEXION

Administration KundApp

Gestion des utilisateurs et du système



Recherche

TEACHER

CLIENTS

SESSIONS

Liste des teachers

NOUVEAU TEACHER

Prénom	Nom	Email	Téléphone	Statut	Crée le	Action
John	Doe	user@gmail.com	0629712167	Actif	21/11/1990	<button>MODIFIER</button>
John	Doe	user@gmail.com	0629712167	Actif	21/11/1990	<button>MODIFIER</button>

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)



KundApp Admin

admin@gmail.com

DÉCONNEXION

Administration KundApp

Gestion des utilisateurs et du système



Recherche

TEACHER

CLIENTS

SESSIONS

Liste des clients

NOUVEAU CLIENT

Prénom	Nom	Email	Téléphone	Statut	Crédits	Crée le	Action
John	Doe	user@gmail.com	0629712167	Actif	4	21/11/1990	MODIFIER
John	Doe	user@gmail.com	0629712167	Actif	4	21/11/1990	MODIFIER

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)

KundApp Admin

admin@gmail.com

DÉCONNEXION

Administration KundApp

Gestion des utilisateurs et du système



Recherche

TEACHER

CLIENTS

SESSIONS

Liste des sessions

Date/Heure	Lieu	Teacher	Sujet	Statut	Participant	Action
11/09/2025 12:00	■ En ligne	Jenna Watkins	YOGA	Annulé	2 participants 😊	VOIR
11/09/2025 12:00	Omvida 75015	Jenna Watkins	YOGA	Programmée	2 participants 😊	MODIFIER
11/09/2025 12:00	■ En ligne	Jenna Watkins	YOGA	Annulé	2 participants 😊	VOIR
11/09/2025 12:00	Omvida 75015	Jenna Watkins	YOGA	Programmée	2 participants 😊	MODIFIER
11/09/2025 12:00	■ En ligne	Jenna Watkins	YOGA	Annulé	2 participants 😊	VOIR

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)



KundApp Admin Sessions

4 crédits

user@gmail.com

DÉCONNEXION

Sessions

Découvrez et gérez vos séances

Toutes les catégories

EN LIGNE

EN PRÉSENTIEL

RECHERCHER PAR PROFESSEUR

SESSIONS DISPONIBLES

MES PROCHAINES SÉANCES

HISTORIQUE

MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

Prof Jenna

En ligne
0 participants (5 places)

S'INSCRIRE



MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

Prof Jenna

En ligne
0 participants (5 places)

S'INSCRIRE



MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

Prof Jenna

En ligne
0 participants (5 places)

S'INSCRIRE



© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)

KundApp Admin Sessions

4 crédits user@gmail.com DÉCONNEXION

Sessions

Découvrez et gérez vos séances

Toutes les catégories EN LIGNE EN PRÉSENTIEL RECHERCHER PAR PROFESSEUR

SESSIONS DISPONIBLES MES PROCHAINES SÉANCES HISTORIQUE

MEDITATION	Prof Jenna	Annulé	i
Jeudi 21 aout 2025 14:00 - 15h30	En ligne 0 participants (5 places)		
MEDITATION	Prof Jenna	Annulé	i
Jeudi 21 aout 2025 14:00 - 15h30	En ligne 0 participants (5 places)		
MEDITATION	Prof Jenna	Annulé	i
Jeudi 21 aout 2025 14:00 - 15h30	En ligne 0 participants (5 places)		

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)

KundApp Admin Sessions

4 crédits user@gmail.com DÉCONNEXION

Sessions

Découvrez et gérez vos séances

Toutes les catégories EN LIGNE EN PRÉSENTIEL RECHERCHER PAR PROFESSEUR

SESSIONS DISPONIBLES MES PROCHAINES SÉANCES HISTORIQUE

MEDITATION	Prof Jenna	En ligne	SE DÉSINScrire	Inscrit ✓	i
Jeudi 21 aout 2025 14:00 - 15h30	0 participants (5 places)				
MEDITATION	Prof Jenna	En ligne	SE DÉSINScrire	Inscrit ✓	i
Jeudi 21 aout 2025 14:00 - 15h30	0 participants (5 places)				
MEDITATION	Prof Jenna	En ligne	SE DÉSINScrire	Inscrit ✓	i
Jeudi 21 aout 2025 14:00 - 15h30	0 participants (5 places)				

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)

KundApp Admin Accueil

teacher@gmail.com

DÉCONNEXION

Bienvenue sur votre espace KundApp

Consultez et gérez vos prochaines séances

MES PROCHAINES SÉANCES

HISTORIQUE

CRÉER UNE SÉANCE

MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

En ligne

0 participants (5 places)

MODIFIER

MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

En ligne

0 participants (5 places)

MODIFIER

MEDITATION

Jeudi 21 aout 2025
14:00 - 15h30

En ligne

0 participants (5 places)

MODIFIER

© 2025 KundApp Admin. Tous droits réservés.

DOSSIER PROFESSIONNEL (DP)

Créer un teacher

Prénom *

Nom *

Email *

Téléphone *

Mot de passe *

Date de naissance *

Biographie

Adresse

Rue

Ville

Code Postal

Pays

[ANNULER](#)

[ENREGISTRER](#)

DOSSIER PROFESSIONNEL (DP)



Créer votre compte

Rejoignez notre communauté d'énergie positive

Prénom

Nom

Email

Mot de passe

➤ CRÉER MON COMPTE

Déjà membre? Connectez-vous

DOSSIER PROFESSIONNEL (DP)



Connexion

Email

Mot de passe

SE CONNECTER

Nouveau sur KundApp? Inscrivez-vous

Supprimer le client

X

Êtes-vous sûr de vouloir supprimer le client ?

ANNULER

CONFIRMER

DOSSIER PROFESSIONNEL (DP)

Modifier le User

Prénom *

John

Nom *

John

Email *

John@gmail.com

Téléphone *

0629712167

Date de naissance *

16/12/1991

Crédits

4

Adresse

Rue

567 chemin du Vieux Reynier

Ville

Houston

Code Postal

83500

Pays

USA

SUPPRIMER CE CLIENT

DÉSACTIVER CE CLIENT

ANNULER

CONFIRMER

DOSSIER PROFESSIONNEL (DP)

Modifier une séance

Type de cours *

Pilates ...

Description *

Type de séance

En présentiel En ligne

Vous pouvez changer de type de séances. Les champs spécifiques à l'ancien type seront effacés.

Lieu de la séance

Nom de la salle/ Lieu *

Omvida

Code Postal *

83500

Lien Google Map *

<https://www.google.com/maps>

Dates et horaires

Date et heure de début*

22/08/2024 14h00

Durée *

1h30

Participants et crédits

Nombre de places *

12

Crédits requis *

1

Entre 1 et 50 participants

Impossible de modifier : des participants se sont déjà inscrits à la séance. Annulez cette session et créez en une autre pour modifier le nombre de crédits.

Matériel

Les participants doivent apporter leur tapis

Cochez cette case si les participants doivent venir avec leur propre équipement

Participants inscrits à la séance

1 participant(s) inscrit(s)

[ANNULER CETTE SÉANCE](#)

[FERMER](#)

[SAUVEGARDER](#)

DOSSIER PROFESSIONNEL (DP)

Consultation de session annulée

Type de cours *

Yoga

Description *

Type de séance

En présentiel En ligne

Lien Zoom *

<https://zoom.us/fr/signin#/login>

Lien de la réunion Zoom pour se connecter à la session en ligne

Dates et horaires

Date et heure de début*

22/08/2024 14h00

Durée *



1h30

Participants et crédits

Nombre de places *

12

Crédits requis *

1

Entre 1 et 50 participants

Impossible de modifier : des participants se sont déjà inscrits à la séance. Annulez cette session et créez en une autre pour modifier le nombre de crédits.

Participants inscrits à la séance

Prénom	Nom	Email
John	Doe	user@gmail.com
John	Doe	user@gmail.com
John	Doe	user@gmail.com

[FERMER](#)

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé en autonomie sur l'analyse et le maquettage, en échangeant avec le formateur pour prioriser les fonctionnalités à intégrer. J'ai également recueilli des retours auprès de mes camarades pour tester l'intuitivité des parcours utilisateurs et identifier d'éventuels points de friction dans l'interface.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - Projet de fin d'études

Période d'exercice - Du 07/04/2025 au 31/08/2025

5. Informations complémentaires (*facultatif*)

Voir en annexe

DOSSIER PROFESSIONNEL (DP)

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

Exemple n°2 - Définir l'architecture logicielle d'une application

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

Dans le cadre de mon projet HealthCare, j'ai défini et mis en œuvre une architecture logicielle distribuée reposant sur une approche microservices, afin de garantir une bonne séparation des responsabilités, une modularité claire et une évolutivité future.

L'application est découpée en plusieurs services indépendants :

- ms-webapp (présentation) : service frontal en Thymeleaf, responsable de l'affichage et des interactions utilisateur
- ms-patient : service métier pour la gestion des patients et des utilisateurs
- ms-notes : service de stockage des notes médicales en base NoSQL (MongoDB)
- ms-diabète : service d'analyse automatisée du diabète à partir des contenus textuels des notes
- ms-configserver : serveur de configuration centralisé via Spring Cloud Config
- ms-discovery : annuaire de services basé sur Eureka pour le référencement et la découverte des différentes services

J'ai structuré chaque microservice selon une architecture multicouche classique (Controller / Service / Repository) afin de garantir la maintenabilité du code, avec des responsabilités bien séparées dans chaque couche. L'application fonctionne en environnement Dockerisé, chaque microservice étant déployé dans un conteneur dédié.

La stratégie de sécurité repose sur Spring Security, avec une authentification centralisée depuis ms-webapp. Cette dernière interroge ms-patient via API pour valider les identifiants. Une fois connecté, l'utilisateur peut naviguer dans l'application selon son rôle (USER ou ADMIN).

L'accès aux routes est protégé dans le filtre de sécurité (SecurityFilterChain) et des logiques d'affichage différentes sont déjà en place selon le rôle attribué.

Cette organisation me permet d'envisager à terme l'intégration d'un API Gateway pour renforcer la centralisation des appels et appliquer des règles transversales.

Enfin, j'ai commencé à réfléchir à des principes d'éco-conception, notamment via la mutualisation des dépendances entre services, la limitation des appels inter-microservices, et l'optimisation future des traitements lourds (comme le calcul de risque diabétique).

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

- Spring Boot & Spring Cloud : pour structurer les microservices, gérer la configuration (Config Server), et orchestrer les services (Eureka).
- Spring Security : utilisé dans ms-webapp pour l'authentification centralisée avec gestion des rôles (USER, ADMIN), en lien avec ms-patient.
- Spring Data JPA avec MySQL pour la gestion des données relationnelles (utilisateurs et patients).
- Spring Data MongoDB pour le stockage des notes médicales en base NoSQL.
- Feign Client pour faciliter la communication inter-services (ms-webapp ↔ ms-patient, ms-notes, ms-diabète).
- Docker pour conteneuriser les microservices et simuler un environnement distribué de production.
- Maven pour la gestion des dépendances et la structuration multi-modules.
- Thymeleaf pour l'interface web, avec des vues adaptées selon les rôles.

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a Java IDE interface with the following details:

- Project Bar:** Shows "Healthcare" as the active project.
- Structure View:** Displays the project structure under "Healthcare". The "ms-discovery" module is currently selected.
- POM.xml (ms-configserver) Editor:** The main editor pane displays the XML code for the POM file. The code includes dependencies for Spring Boot Actuator and Spring Cloud Config Server.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.3</version>
    <relativePath/> <!-- lookup parent from repository --&gt;
  &lt;/parent&gt;
  &lt;groupId&gt;jroulet&lt;/groupId&gt;
  &lt;artifactId&gt;ms-configserver&lt;/artifactId&gt;
  &lt;version&gt;0.0.1-SNAPSHOT&lt;/version&gt;
  &lt;name&gt;ms-configserver&lt;/name&gt;
  &lt;description&gt;configserver&lt;/description&gt;
  &lt;url/&gt;
  &lt;licenses&gt;
    &lt;license/&gt;
  &lt;/licenses&gt;
  &lt;developers&gt;
    &lt;developer/&gt;
  &lt;/developers&gt;
  &lt;scm&gt;
    &lt;connection/&gt;
    &lt;developerConnection/&gt;
    &lt;tag/&gt;
    &lt;url/&gt;
  &lt;/scm&gt;
  &lt;properties&gt;
    &lt;java.version&gt;17&lt;/java.version&gt;
    &lt;spring-cloud.version&gt;2024.0.0&lt;/spring-cloud.version&gt;
  &lt;/properties&gt;
  &lt;dependencies&gt; ⌂ Add Starters...
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
      &lt;artifactId&gt;spring-boot-starter-actuator&lt;/artifactId&gt;
    &lt;/dependency&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.cloud&lt;/groupId&gt;
      &lt;artifactId&gt;spring-cloud-config-server&lt;/artifactId&gt;
    &lt;/dependency&gt;
  &lt;/dependencies&gt;
</pre>
```

POM Configserver

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** Shows "Healthcare" as the current project.
- Toolbars:** Standard IntelliJ toolbars for Project, Commit, PR, Structure, Find, Build, Services, Terminal, Problems, and Git.
- Central Area:** Displays the content of `pom.xml (ms-discovery)`. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.3</version>
    <relativePath/> 
  </parent>
  <groupId>jroullet</groupId>
  <artifactId>ms-discovery</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ms-discovery</name>
  <description>Discovery Service - Eureka</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>17</java.version>
    <spring-cloud.version>2022.0.4</spring-cloud.version>
  </properties>
  <dependencies> ⚡ Add Starters...
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-config</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
  </dependencies>

```

Bottom Left: The text "POM Eurêka" is visible.

DOSSIER PROFESSIONNEL (DP)

The screenshot shows an IDE interface with the following details:

- Project View:** Shows a tree structure of a project named "Healthcare". It includes sub-modules like ".idea", ".mvn", "docker-compose", "ms-configserver", "ms-database", "ms-discovery", "ms-notes", "ms-patient", and "ms-webapp". The "ms-webapp" module is expanded, showing ".mvn", "src", "main", and "java" sub-folders. Inside "java", there are packages for "jroulet.mswebapp" containing "clients", "controller", "dto", "model", and "security.config". The "security.config" package contains two files: "CustomAuthenticationFailureHandler" and "SpringSecurityConfig".
- Code Editor:** The right pane displays the content of the "SpringSecurityConfig.java" file. The code defines a class "SpringSecurityConfig" with annotations like @Configuration, @EnableWebSecurity, and @EnableMethodSecurity. It sets up a "customAuthenticationFailureHandler" and defines a "passwordEncoder" using BCryptPasswordEncoder. The "http" block configures security filters, including "authorizeHttpRequests", "requestMatchers", "permitAll", "hasRole", and "authenticated" methods. It also handles form login, setting success URL to "/home", always using true, and failure URL to "/sign?authError=Invalid+email+or+password". It includes logout access and logout URLs.

Security Config

DOSSIER PROFESSIONNEL (DP)

The screenshot shows an IDE interface with the following details:

- Project:** Healthcare
- File:** pom.xml (ms-patient)
- Content (pom.xml):**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <scm>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2024.0.0</spring-cloud.version>
    </properties>
    <dependencies> ⌂ Add Starters...
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
    </dependencies>

```

JPA

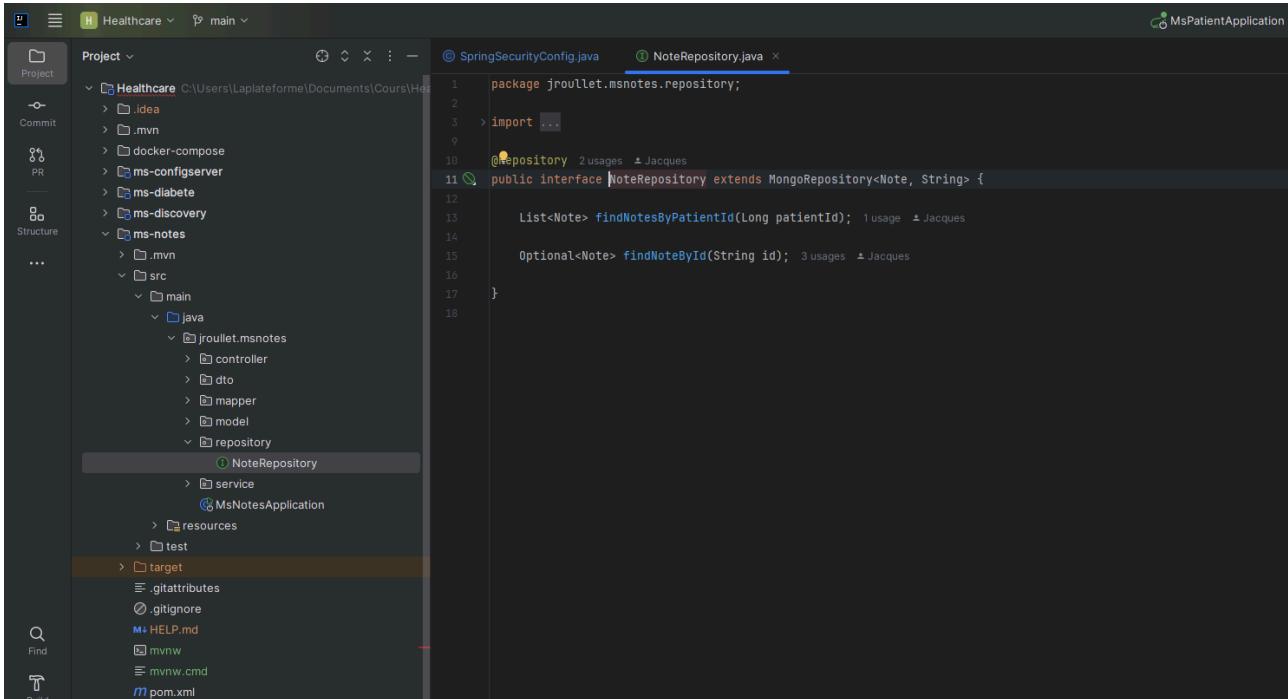
The screenshot shows the MongoDB Compass interface connected to the database "notes.notes" at "localhost:27018".

- Connections:** MongoDB Compass - localhost:27018/notes.notes
- Collection:** notes
- Documents:** 16
- Content:** The screenshot displays 16 documents in the notes collection, each representing a note with fields: _id, creationDate, lastUpdateDate, content, patientId, and _class.

_id	creationDate	lastUpdateDate	content	patientId	_class
ObjectId('67e67e9c7fef57739146ab91')	2025-03-27T23:00:00.000+00:00	2025-03-27T23:00:00.000+00:00	"pizza"	1	"jroulet.msnotes.model.Note"
ObjectId('67e67ea87fef57739146ab92')	2025-03-27T23:00:00.000+00:00		"fumeur"	1	"jroulet.msnotes.model.Note"
ObjectId('67e67ec87fef57739146ab93')	2025-03-27T23:00:00.000+00:00		"hemoglobine a1c"	1	"jroulet.msnotes.model.Note"
ObjectId('67e67ec87fef57739146ab94')	2025-03-27T23:00:00.000+00:00		"taille , poids"	1	"jroulet.msnotes.model.Note"

Mongo

DOSSIER PROFESSIONNEL (DP)

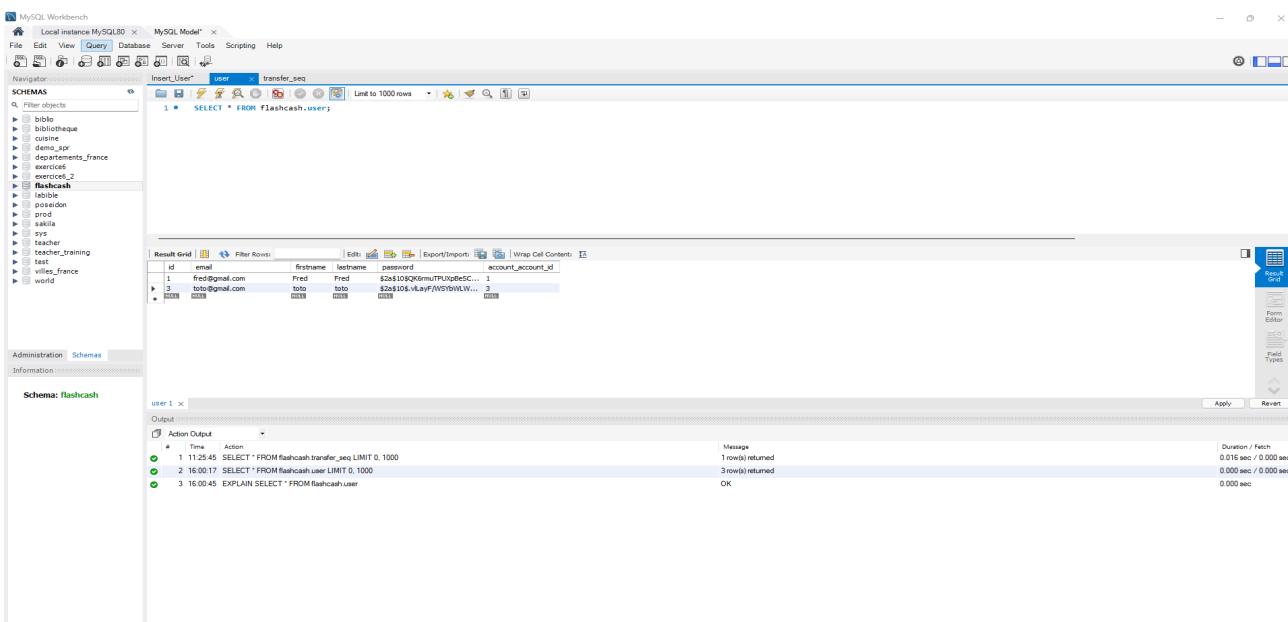


The screenshot shows a Java project structure for 'Healthcare' in a dark-themed IDE. The 'src/main/java' package contains several sub-packages: jrouillet.msnotes, controller, dto, mapper, model, and repository. The 'repository' package is currently selected. Inside it, the file 'NoteRepository.java' is open, showing the following code:

```
package jrouillet.msnotes.repository;
import ...;

@Repository
public interface NoteRepository extends MongoRepository<Note, String> {
    List<Note> findNotesByPatientId(Long patientId);
    Optional<Note> findNoteById(String id);
}
```

MongoRepository - accès aux données de la bdd non-relationnelle



The screenshot shows the MySQL Workbench interface. In the top query editor, a SELECT statement is run against the 'flashcash.user' table:

```
1 • SELECT * FROM flashcash.user;
```

The results are displayed in a grid:

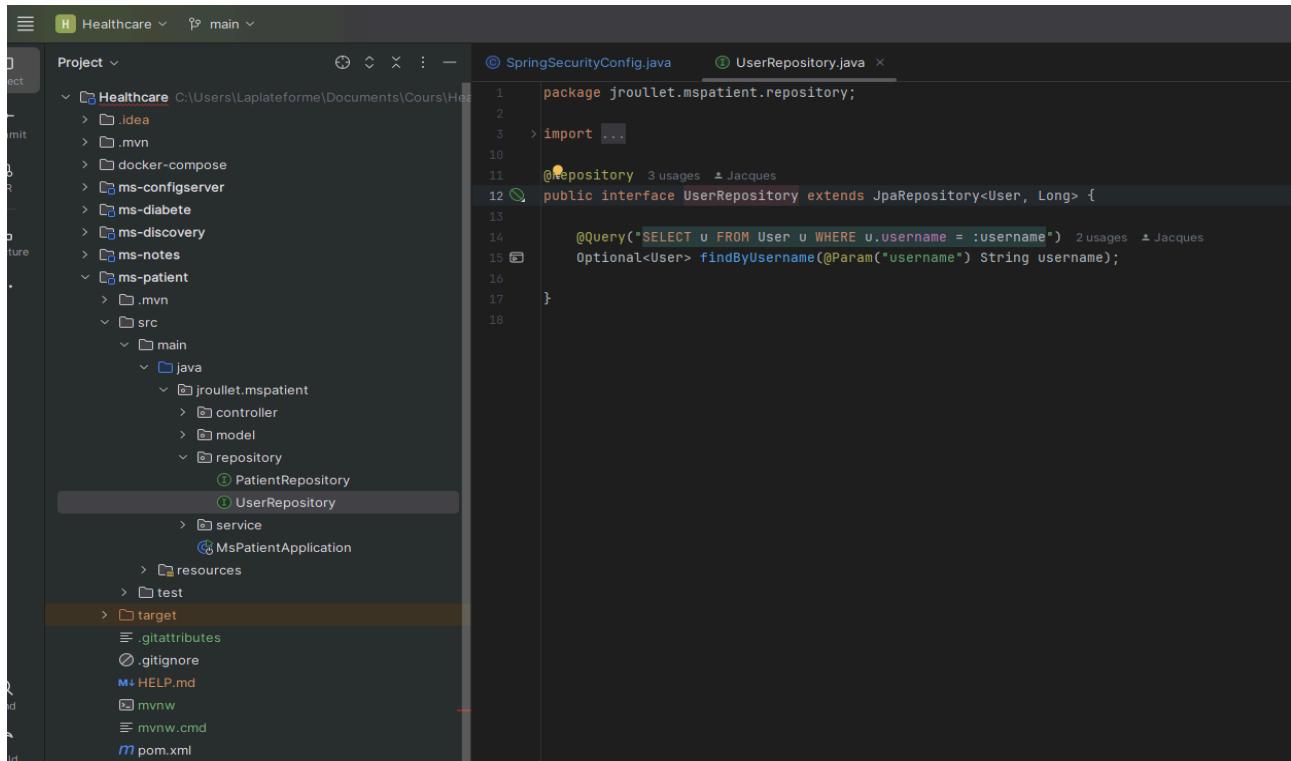
	id	email	firstname	lastname	password	account_account_id
1	1	fred@gmail.com	Fred	Fred	\$2a\$10\$QKgmuUTPUxpb5C...	1
2	3	toto@gmail.com	toto	toto	\$2a\$10\$vlayFIVStbWLV...	3
3	4	sotis	sotis	sotis	\$2a\$10\$...	4

In the bottom session history pane, three actions are listed:

Action	Time	Output	Message	Duration / Fetch
1	11:29:45	SELECT * FROM flashcash.transfer_seq LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
2	16:00:17	SELECT * FROM flashcash.user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
3	16:00:45	EXPLAIN SELECT * FROM flashcash.user	OK	0.000 sec

Mysql

DOSSIER PROFESSIONNEL (DP)

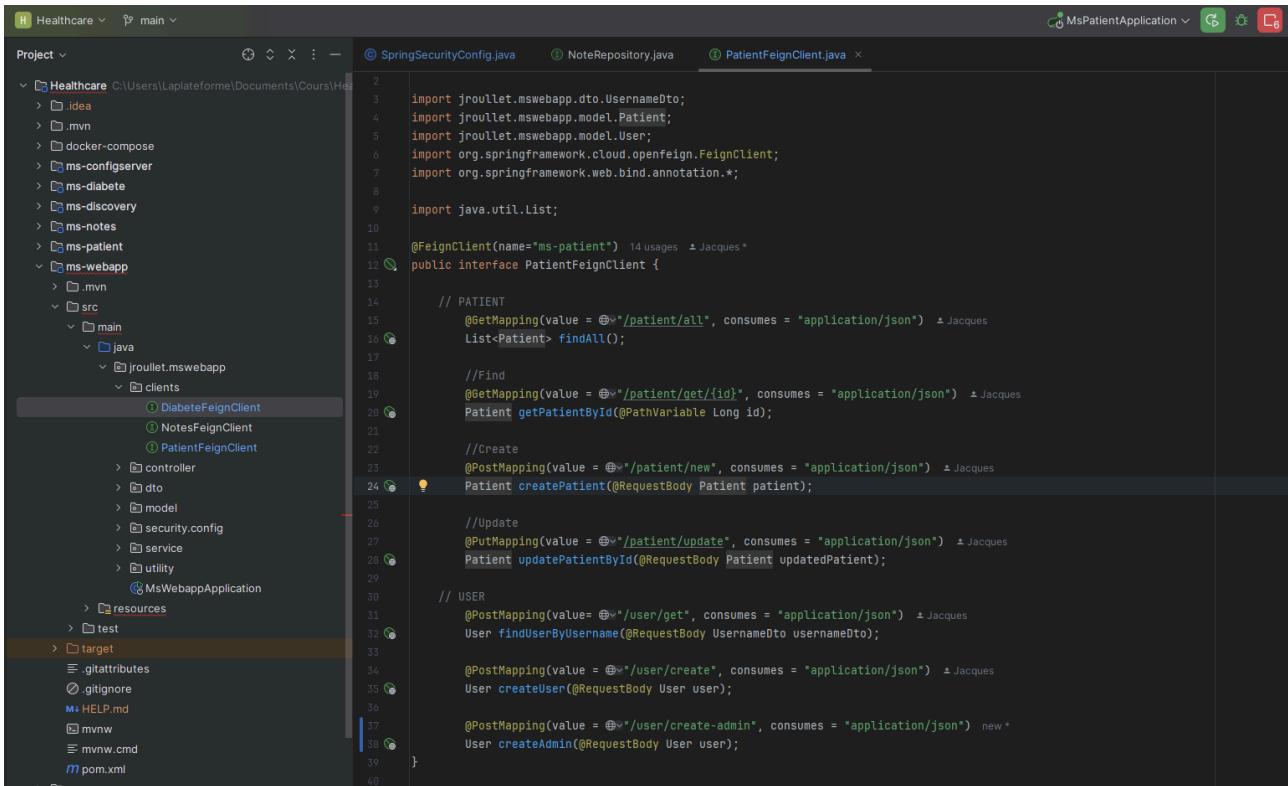


The screenshot shows a Java IDE interface with a dark theme. On the left, the project structure is displayed under the 'Project' tab, showing a 'Healthcare' project with various modules like ms-care, ms-configserver, ms-diabete, ms-discovery, ms-notes, ms-patient, and ms-notes. The 'ms-patient' module is expanded, showing its .mvn and src folders, which contain main, java, and resources sub-folders. The 'repository' package is selected, containing PatientRepository and UserRepository classes. The right panel shows the code for UserRepository.java:

```
1 package jrouillet.mspatient.repository;
2
3 import ...
4
5 @Repository 3 usages + Jacques
6 public interface UserRepository extends JpaRepository<User, Long> {
7
8     @Query("SELECT u FROM User u WHERE u.username = :username") 2 usages + Jacques
9     Optional<User> findByUsername(@Param("username") String username);
10
11 }
12
13
14
15
16
17 }
```

JpaRepository - accès aux données de la bdd relationnelle

DOSSIER PROFESSIONNEL (DP)



The screenshot shows a Java code editor in an IDE. The project structure on the left includes modules like Healthcare, ms-webapp, and ms-patient. The current file is PatientFeignClient.java, which contains the following code:

```
import jrouillet.mswebapp.dto.UsernameDto;
import jrouillet.mswebapp.model.Patient;
import jrouillet.mswebapp.model.User;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@FeignClient(name="ms-patient") 14 usages ▾ Jacques
public interface PatientFeignClient {

    // PATIENT
    @GetMapping(value = @v"/patient/all", consumes = "application/json") ▾ Jacques
    List<Patient> findAll();

    //Find
    @GetMapping(value = @v"/patient/get/{id}", consumes = "application/json") ▾ Jacques
    Patient getPatientById(@PathVariable Long id);

    //Create
    @PostMapping(value = @v"/patient/new", consumes = "application/json") ▾ Jacques
    Patient createPatient(@RequestBody Patient patient);

    //Update
    @PutMapping(value = @v"/patient/update", consumes = "application/json") ▾ Jacques
    Patient updatePatientById(@RequestBody Patient updatedPatient);

    // USER
    @PostMapping(value= @v"/user/get", consumes = "application/json") ▾ Jacques
    User findUserByUsername(@RequestBody UsernameDto usernameDto);

    @PostMapping(value = @v"/user/create", consumes = "application/json") ▾ Jacques
    User createUser(@RequestBody User user);

    @PostMapping(value = @v"/user/create-admin", consumes = "application/json") new *
    User createAdmin(@RequestBody User user);
}
```

FeignClient

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the Docker Desktop interface with the following details:

- Containers** tab selected.
- Container CPU usage:** 2.47% / 1200% (12 CPUs available).
- Container memory usage:** 2GB / 7.24GB.
- Show charts** button.
- Search bar** and **Only show running containers** filter.
- Table of running containers:**

Name	Container ID	Image	Port(s)	CPU (%)	Last start	Actions
prod	-	-	-	2.52%	26 secor	[...]
configserver-ms	7c4411183b8d	jroulet/healthcare-ms-configserver:latest	8888:8888	0.19%	1 minute	[...]
discovery-ms	e15e61724146	jroulet/healthcare-ms-discovery:latest	8761:8761	0.18%	56 secon	[...]
mongodb	1b439c9f7f24	mongo	27018:27017	0.64%	5 days a	[...]
notes-ms	17975987e8fb	jroulet/healthcare-ms-notes:latest	9000:9000	0.35%	26 secon	[...]
patient-ms	4137c96b3eb6	jroulet/healthcare-ms-patient:latest	8070:8070	0.18%	31 secon	[...]
patientdb	04ef8236ff05	mysql	3308:3306	0.77%	5 days a	[...]
webapp-ms	1a5d8f08a118	jroulet/healthcare-ms-webapp:latest	8090:8090	0.21%	36 secon	[...]

- Showing 8 items** message.
- Engine status:** Engine running.
- System stats:** RAM 5.50 GB, CPU 1.00%, Disk: 26.79 GB used (limit 1006.85 GB).
- Terminal** and **New version available** buttons.
- Docker** logo at the bottom left.

DOSSIER PROFESSIONNEL (DP)

```
2 <html xmlns:th="http://www.thymeleaf.org"
3 <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <title>Accueil | HealthCare</title>
6     <link th:href="@{css/mdb.min.css}" rel="stylesheet"/>
7     <link th:href="@{css/styles.css}" rel="stylesheet"/>
8 </head>
9 <body>
10    <nav class="navbar navbar-expand-lg">
11        <div class="container">
12            <div class="Logo me-3">
13                
14            </div>
15            <a class="navbar-brand" href="/home">HealthCare</a>
16            <button class="navbar-toggler" type="button" data-mdb-toggle="collapse" data-mdb-target="#navbarMain">
17                <span class="navbar-toggler-icon"></span>
18            </button>
19            <div class="collapse navbar-collapse" id="navbarMain">
20                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
21                    <li class="nav-item">
22                        <a class="nav-link active" href="/home">Patients</a>
23                    </li>
24                    <li class="nav-item">
25                        <a class="nav-link" href="/patient/new">Ajouter un patient</a>
26                    </li>
27                    <li class="nav-item" sec:authorize="hasRole('ADMIN')">
28                        <a class="nav-link" href="/admin">Admin</a>
29                    </li>
30                </ul>
31                <div class="d-flex align-items-center">
32                    <span class="me-3" th:if="${user != null}" th:text="${user.getUsername()}"/>
33                    <button type="button" class="btn btn-outline-primary" onclick="window.location.href='/logout'">Déconnexion</button>
34                </div>
35            </div>
36        </div>
37    </nav>
```

Thymeleaf - vues adaptées selon les rôles

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a Java development environment with the following details:

- Project Structure:** The project is named "ms-webapp" and contains a "src" folder with "main" and "java" subfolders. Inside "main", there are "resources" and "templates" folders containing various Thymeleaf templates like "add-note-to-patient.html", "add-patient.html", "admin-dashboard.html", etc.
- Code Editor:** A code editor window displays a Thymeleaf template named "admin-dashboard.html". The code includes meta tags, a title, and a main body section with an h1 header and a welcome message.
- Toolbars and Menus:** Standard IDE toolbars and menus are visible, including "File", "Edit", "Search", "Run", and "Help".

Vue Role Admin

The screenshot shows a web browser displaying the "Espace Administrateur" page at the URL "localhost:8090/admin". The page content is:

Bienvenue sur votre tableau de bord admin.

Vue Role Admin

DOSSIER PROFESSIONNEL (DP)

The screenshot shows the HealthCare application's homepage. At the top, there is a navigation bar with links for 'HealthCare', 'Patients', 'Ajouter un patient', and 'Admin'. On the right side of the nav bar, there is an email address 'toto@gmail.com' and a 'DÉCONNEXION' button. Below the nav bar, a blue header bar displays the text 'Bienvenue sur votre espace HealthCare' and 'Gérez vos patients et leurs informations médicales'. To the right of this bar is a blue button labeled 'AJOUTER UN PATIENT'. The main content area is titled 'Liste des patients' and contains a table with columns: Nom, Prénom, Date de naissance, Sexe, Email, Téléphone, and Actions. A message 'Aucun patient enregistré' is displayed in the center of the table area.

Vue Role Admin

This screenshot shows the same HealthCare application homepage as the previous one, but it is viewed by a user with the role 'User'. The navigation bar, header, and main content area are identical to the Admin view, including the 'Liste des patients' table and the 'Aucun patient enregistré' message. However, the email address shown on the right is 'tata@gmail.com' instead of 'toto@gmail.com'.

Vue Role User

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

J'ai réalisé cette architecture en autonomie dans le cadre de ma formation, en m'appuyant sur la documentation officielle de Spring et des ressources pédagogiques.

J'ai également sollicité ponctuellement l'aide de mon formateur pour valider certaines décisions techniques, notamment en matière de sécurité et d'orchestration des services.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme

Chantier, atelier, service ▶ HealthCare

Période d'exercice ▶ Du 03/03/2025 au 03/04/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

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

Exemple n°3 - Concevoir et mettre en place une base de données relationnelle

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

Lors de la conception du projet FlashCash

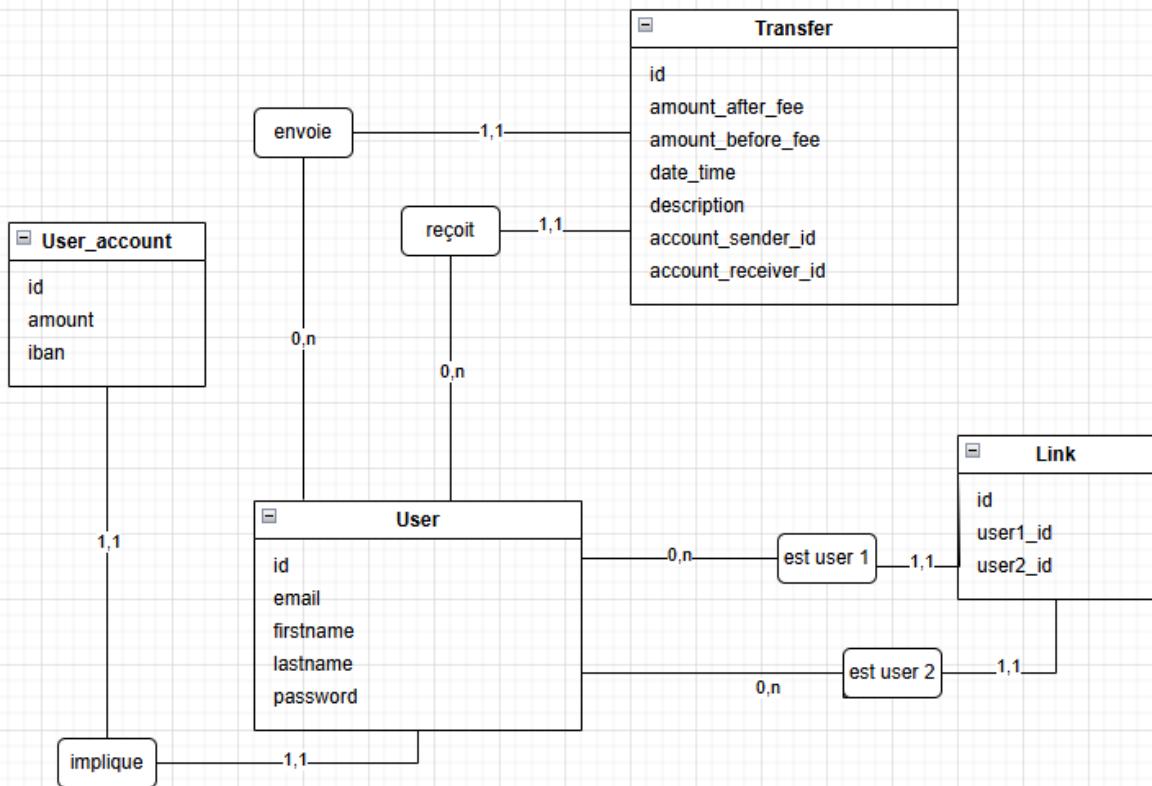
- Conception du MCD
- Conception du MLD
- Conception du MPD

Lors des projets BankApp et HealthCare

- Mise en oeuvre des instructions de création/modification/suppression de base de données
- Script de création de base de données
- Mise en oeuvre de contraintes/validateurs (voir points précédents)
- Exprimer les besoins de sécurité du système de gestion de base de données (SGDB), de gestion des comptes et de la politique de mots de passe

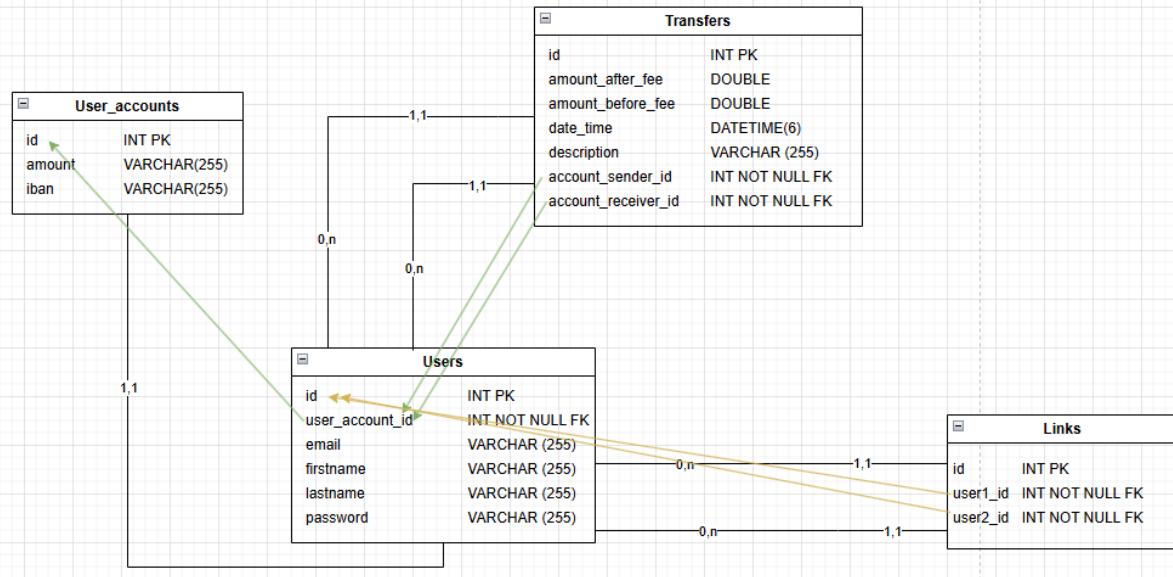
2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)



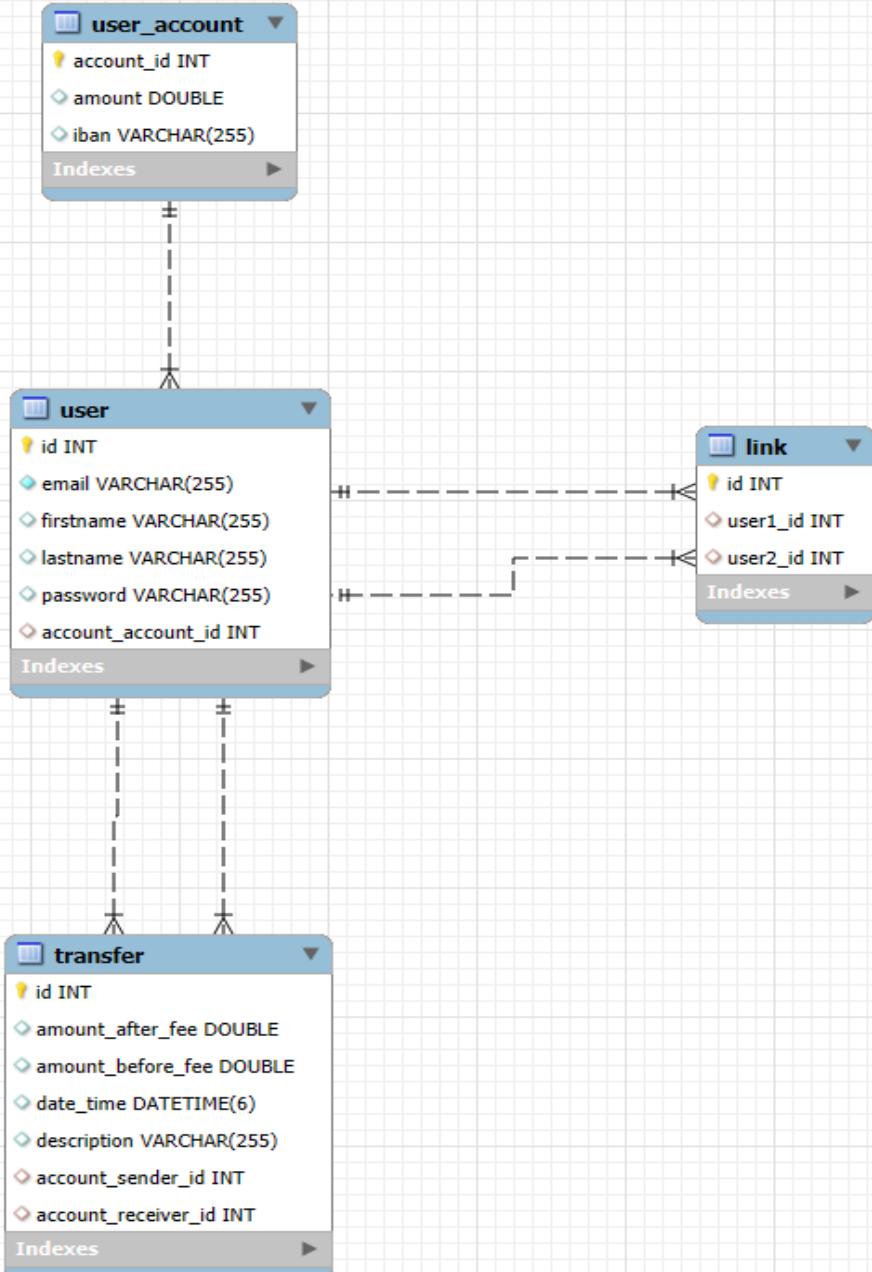
MCD

DOSSIER PROFESSIONNEL (DP)



MLD

DOSSIER PROFESSIONNEL (DP)



MPD

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a code editor interface with a dark theme. On the left, the project structure for 'BankApp' is displayed, including '.idea', '.mvn', 'src' (containing 'main' and 'msaccounts'), 'resources' (containing 'db' with 'data.sql' and 'schema.sql'), 'static', 'templates', and 'test'. In the center, three tabs are open: 'data.sql', 'application.yml', and 'schema.sql'. The 'schema.sql' tab is active and contains the following SQL code:

```
1 DROP TABLE IF EXISTS account;
2 DROP TABLE IF EXISTS customer;

3
4
5 CREATE TABLE `customer`
(
6     customer_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
7     name VARCHAR(100) NOT NULL,
8     email VARCHAR(100) NOT NULL,
9     mobile_number VARCHAR(20) NOT NULL,
10    create_dt DATE DEFAULT CURRENT_DATE
11);
12
13
14 CREATE TABLE `account`
(
15     account_number BIGINT PRIMARY KEY,
16     customer_id INT NOT NULL,
17     account_type VARCHAR(100) NOT NULL,
18     bank_address VARCHAR(200) NOT NULL,
19     create_dt DATE DEFAULT CURRENT_DATE,
20     CONSTRAINT fk_account_customer FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
21     ON DELETE CASCADE
22);
23
```

Script création base de données

The screenshot shows a code editor interface with a dark theme. The 'schema.sql' tab is active and contains the following SQL code:

```
1
2 INSERT INTO `customer` (name, email, mobile_number, create_dt)
3     VALUES ( 'john', 'john@gmail.com', '514-552-5555', create_dt CURRENT_DATE());
4
5
6 INSERT INTO `account` (customer_id, account_number, account_type, bank_address, create_dt)
7     VALUES ( 1, '12341646', 'Savings', '123 Main Street', create_dt CURRENT_DATE());
```

Script insertion données

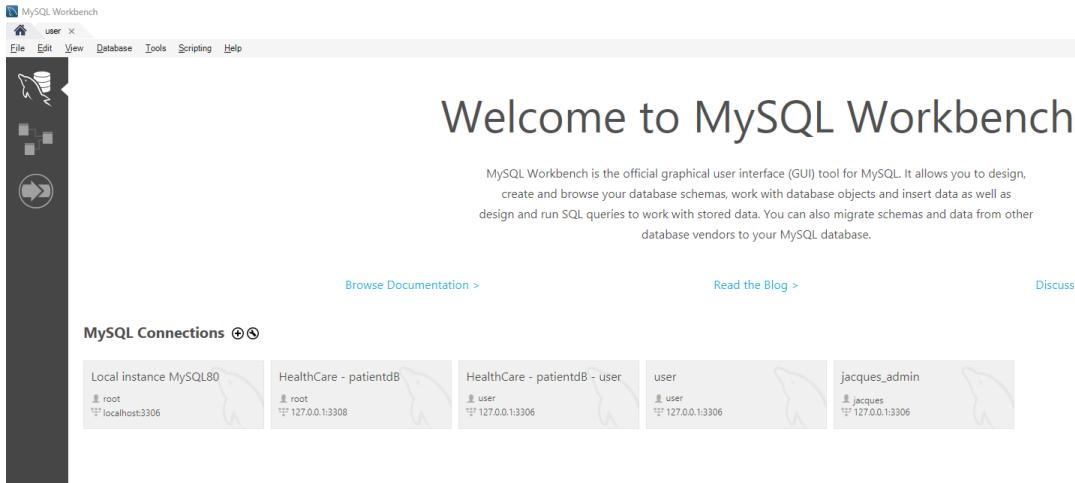
DOSSIER PROFESSIONNEL (DP)

```
application.yml ×

1  server:
2      port: 8080
3
4  spring:
5      application:
6          name: ms-accounts
7
8  datasource:
9      # url: jdbc:h2:mem:testdb
10     url: jdbc:h2:file:./data/msaccounts
11     # username: todo
12     # password: todo
13     driver-class-name: org.h2.Driver
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

Mise en oeuvre des instructions de création/modification/suppression de base de données

DOSSIER PROFESSIONNEL (DP)



Gestion des comptes et de la politique de mots de passe de la base de données

3. Avec qui avez-vous travaillé ?

J'ai configuré l'architecture en autonomie dans le cadre de ma formation, en m'appuyant sur les documentations et les ressources pédagogiques.

4. Contexte

Nom de l'entreprise, organisme ou association ➔ La Plateforme

Chantier, atelier, service ➔ FlashCAsh, BankApp, HealthCare

Période d'exercice ➔ Du 10/12/2024 au 03/04/2025

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

Concevoir et développer une application sécurisée organisée par couches

Exemple n°4 - Développer des composants d'accès aux données SQL et NoSQL

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

- Création des entités JPA à partir du modèle de données défini en amont (documents de conception) (screenshots 1 à 3)
- Mise en place des Repository Spring Data, sous forme d'interfaces étendant JpaRepository, afin de gérer facilement les opérations CRUD (Create, Read, Update, Delete) (screenshots 4 à 5)
- Mapping objet-relationnel (ORM, Hibernate) grâce aux annotations @Entity, @Id, @GeneratedValue, permettant de faire le lien entre les classes Java et les tables SQL. Hibernate gère les relations entre entités, la validation des contraintes, et garantit l'intégrité des données via l'annotation @Transactional.
- Configuration de la connexion sécurisée à la base de données via le fichier application.yml ou application.properties, en prenant soin de protéger les identifiants et de respecter les bonnes pratiques de sécurité (screenshots 6 à 7)
- Réflexion autour de la sécurité, avec une attention particulière portée à la protection contre les injections SQL, notamment par l'utilisation de procédures stockées et de requêtes paramétrées. (screenshots 8 à 9)

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)

```
© TransferController.java      © User.java ×  <> account.html      © HomeController.java      © AddToFlashCashForm.java
1 package jr.dev.FlashCash.model;
2
3 > import ...
4
5
6
7
8 < @Data
9 < * jacques-roulet *
10 < @Entity
11 < public class User {
12   <@Id
13   <@GeneratedValue(strategy = GenerationType.IDENTITY)
14   <private Integer id;
15
16
17   <@Column(unique = true, nullable = false)
18   <private String email; //login
19
20   <private String firstname;
21   <private String lastname;
22
23 > //...
24
25
26 < @NotBlank(message = "Password is mandatory")
27 < @StrongPassword
28   <private String password;
29
30
31
32   <@ManyToMany
33   <private List<Link> links;
34
35
36   <@OneToOne (cascade = CascadeType.ALL, fetch = FetchType.EAGER)
37   <private UserAccount account;
38 }
39
```

Création des entités JPA

DOSSIER PROFESSIONNEL (DP)

```
@Data  ✎ jacques-roulet
@Entity
💡
public class Transfer {

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

    private LocalDateTime dateTime;
    private Double amountBeforeFee;
    private Double amountAfterFee;

    //  @Size(max = 255, message = "Description must not exceed 255 characters")
    private String description;

    @ManyToOne
    @JoinColumn(name = "account_sender_id")
    private User from;
    @ManyToOne
    @JoinColumn(name = "account_receiver_id")
    private User to;
}
```

Création des entités JPA

```
✓ @Data  ✎ jacques-roulet
@Entity
💡
public class Link {

    ✓ @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Integer id;

    @ManyToOne
    User user1;

    @ManyToOne
    User user2;

}
```

Création des entités JPA

DOSSIER PROFESSIONNEL (DP)

```
① NoteRepository.java ×  
1 package jrouillet.msnotes.repository;  
2  
3 > import ...  
9  
10 @Repository 2 usages ▲ Jacques  
11 ② public interface NoteRepository extends MongoRepository<Note, String> {  
12  
13     List<Note> findNotesByPatientId(Long patientId); 1 usage ▲ Jacques  
14  
15     Optional<Note> findNoteById(String id); 3 usages ▲ Jacques  
16  
17 }  
18
```

Mise en place des Repository Spring Data

```
① UserRepository.java ×  
1 package jrouillet.mspatient.repository;  
2  
3 > import ...  
10  
11 @Repository 3 usages ▲ Jacques  
12 ② public interface UserRepository extends JpaRepository<User, Long> {  
13  
14     @Query("SELECT u FROM User u WHERE u.username = :username") 2 usages ▲ Jacques  
15     ③ Optional<User> findByUsername(@Param("username") String username);  
16  
17 }  
18
```

Mise en place des Repository Spring Data

DOSSIER PROFESSIONNEL (DP)

```
© Patient.java ×  
1 package jrouurret.mspatient.model;  
2  
3 > import ...  
13  
14 @Entity * Jacques *  
15 @Getter  
16 @Setter  
17 @ToString  
18 @NoArgsConstructor  
19 @AllArgsConstructor  
20 public class Patient {  
21  
22     @Id  
23     @GeneratedValue(strategy = GenerationType.IDENTITY)  
24     private Long id;  
25     @NotNull  
26     private String firstName;  
27     private String lastName;  
28     private LocalDate birthday;  
29     private String gender;  
30     private String address;  
31     private String phone;  
32     @Email  
33     private String email;  
34  
35  
36 }  
37
```

Mapping objet-relationnel (ORM, Hibernate) grâce aux annotations

DOSSIER PROFESSIONNEL (DP)

```
application.yml ×

1  server:
2    port: 8070
3  spring:
4    application:
5      name: "ms-patient"
6    profiles:
7      active: prod
8  ↴ datasource:
9    url: jdbc:h2:mem:mspatient
10   #   url: jdbc:h2:file:./data/mspatient
11   username: "TODO"
12   password: "TODO"
13   driver-class-name: org.h2.Driver
14 > # ...
15   jpa:
16     hibernate:
17       ddl-auto: update
18       show-sql: true
19       database-platform: org.hibernate.dialect.MySQL8Dialect
20     sql:
21       init:
22         mode: always
```

Configuration de la connexion sécurisée à la base de données

DOSSIER PROFESSIONNEL (DP)

```
1 • Ⓜ CREATE DEFINER='root'@'localhost' PROCEDURE `PS_TransfersByUser` (
2     IN userId INT -- Entry param
3 )
4 Ⓜ BEGIN
5     SELECT t.id,
6         t.amount_after_fee,
7         t.amount_before_fee,
8         t.date_time,
9         t.description,
10        t.account_sender_id,
11        t.account_receiver_id
12    FROM transfer t
13    WHERE t.account_sender_id = userId
14    OR t.account_sender_id IN(
15        SELECT l.user1_id
16        FROM link l
17        WHERE l.user2_id = userId
18        UNION
19        SELECT l.user2_id
20        FROM link l
21        WHERE l.user1_id= userId)
22    ;
23 END
```

```
1 package jr.dev.FlashCash.interfaces.repository;
2
3 > import ...
10
11 @Repository 5 usages ▲ jacques-roulet
12 Ⓜ public interface TransferRepository extends JpaRepository<Transfer, Integer> {
13 //    List<Transfer> findTransfersByFrom(User user);
14
15    // Stocked Procedure
16    @Query(value = "CALL PS_TransfersByUser(:userId)", nativeQuery = true) 1 usage ▲ jacques-roulet
17    List<Transfer> returnTransfers(Integer userId);
18 }
19
```

Réflexion autour de la sécurité - procédure stockée

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

J'ai développé ces composants en autonomie dans le cadre de ma formation, en m'appuyant sur la documentation officielle de Spring et des ressources pédagogiques.

J'ai également sollicité ponctuellement l'aide de mon formateur pour valider certaines décisions techniques afin de m'aider à anticiper les contraintes qui en découlent.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - FlashCash, HealthCare

Période d'exercice - Du 10/12/2024 au 03/04/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL^(DP)

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

Exemple n°1 - Préparer et exécuter les plans de tests d'une application

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

Créer un environnement de Tests

Implémentation de Swagger (test des endpoints automatisés)

Exécuter des tests unitaires en méthode Agile et en TDD (Test Driven Development)

Analyse du taux de couverture avec Jacoco

Tests d'intégration / Test de charge (de performance)

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)

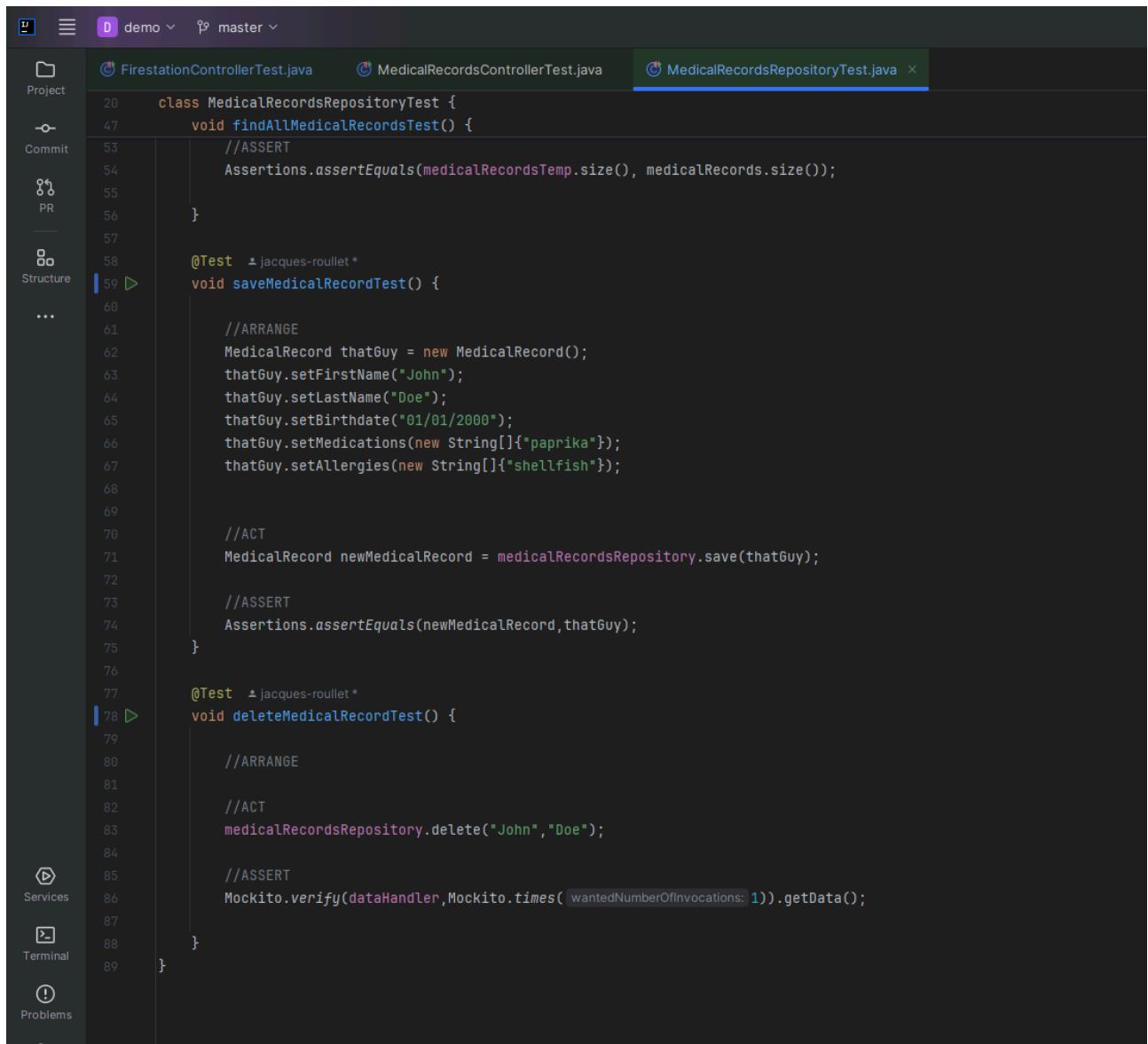
The screenshot shows a Java IDE interface with the following details:

- Project Bar:** Shows "demo" and "master" branches.
- Sidebar:** Includes sections for Project, Commit, PR, Structure, Services, Terminal, Problems, and Git.
- Code Editor:** The active tab is "MedicalRecordsRepositoryTest.java". The code is a unit test for a medical records repository using Mockito and JUnit. It includes setup methods for medical records and data handlers, and tests for finding all medical records and saving a medical record.

```
20 > class MedicalRecordsRepositoryTest {  
21     @Autowired  
22     MedicalRecordsRepository medicalRecordsRepository;  
23     @MockBean  
24     DataHandler dataHandler;  
25     List<MedicalRecord> medicalRecordsTemp = new ArrayList<>(); 3 usages  
26     @BeforeEach @jacques-roulet*  
27     void setUpMedicalRecordTest() {  
28         MedicalRecord medicalRecordGuy = new MedicalRecord();  
29         medicalRecordGuy.setFirstName("John");  
30         medicalRecordGuy.setLastName("Doe");  
31         medicalRecordGuy.setBirthdate("01/01/2000");  
32         medicalRecordGuy.setMedications(new String[]{"paprika"});  
33         medicalRecordGuy.setAllergies(new String[]{"shellfish"});  
34         medicalRecordsTemp.add(medicalRecordGuy);  
35     }  
36     Data mockData = Mockito.mock(Data.class);  
37     Mockito.when(dataHandler.getData()).thenReturn(mockData);  
38     Mockito.when(mockData.getMedicalrecords()).thenReturn(medicalRecordsTemp);  
39 }  
40  
41 @Test @jacques-roulet*  
42 void findAllMedicalRecordsTest() {  
43     //ARRANGE  
44     //ACT  
45     List<MedicalRecord> medicalRecords = medicalRecordsRepository.findAllMedicalRecords();  
46     //ASSERT  
47     Assertions.assertEquals(medicalRecordsTemp.size(), medicalRecords.size());  
48 }  
49  
50 @Test @jacques-roulet*  
51 void saveMedicalRecordTest() {  
52     //ARRANGE  
53     //ACT  
54     //ASSERT  
55 }  
56  
57 }  
58  
59 > class FirestationControllerTest {  
60     @Test @jacques-roulet*  
61     void testGetAllFirestations() {  
62         //ARRANGE  
63         //ACT  
64         //ASSERT  
65     }  
66 }  
67  
68 > class MedicalRecordsControllerTest {  
69     @Test @jacques-roulet*  
70     void testGetMedicalRecord() {  
71         //ARRANGE  
72         //ACT  
73         //ASSERT  
74     }  
75 }
```

- Bottom Bar:** Shows "Tests unitaires".

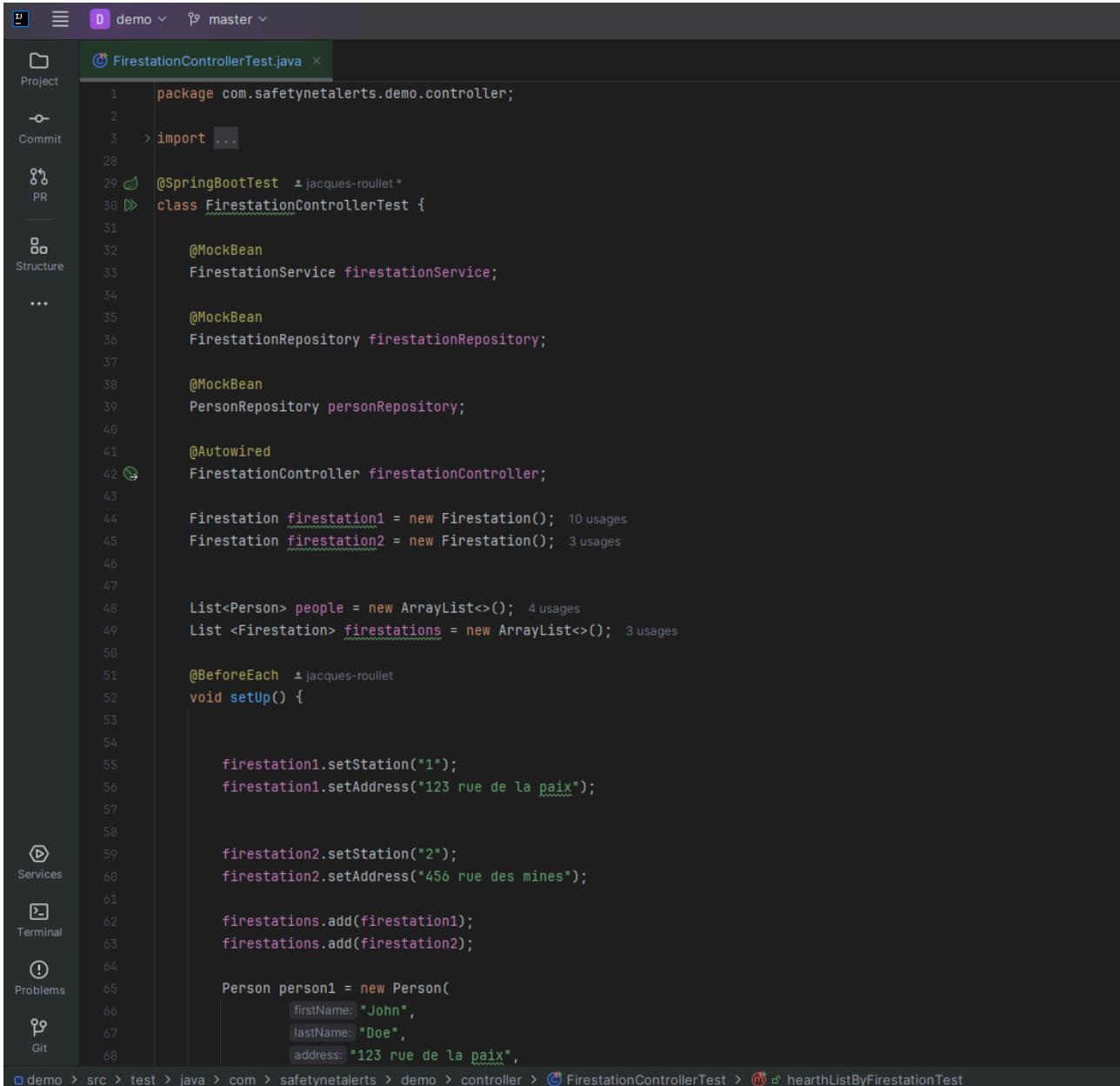
DOSSIER PROFESSIONNEL (DP)



The screenshot shows a Java IDE interface with the following details:

- Project Bar:** demo -> master
- File Tabs:** FirestationControllerTest.java, MedicalRecordsControllerTest.java, MedicalRecordsRepositoryTest.java (highlighted)
- Code Area:** The code is for a `MedicalRecordsRepositoryTest` class. It contains three test methods: `findAllMedicalRecordsTest`, `saveMedicalRecordTest`, and `deleteMedicalRecordTest`.
 - `findAllMedicalRecordsTest`: Checks if the size of the medical records in memory equals the size of the repository.
 - `saveMedicalRecordTest`:
 - ARRANGE: Creates a new `MedicalRecord` object named `thatGuy` with various properties set.
 - ACT: Calls `medicalRecordsRepository.save(thatGuy)`.
 - ASSERT: Checks if the saved record equals the original `thatGuy` object.
 - `deleteMedicalRecordTest`:
 - ARRANGE: Deletes a record from the repository using `medicalRecordsRepository.delete("John", "Doe")`.
 - ASSERT: Verifies that the data handler was called once using `Mockito.verify(dataHandler, Mockito.times(1)).getData()`.
- Sidebar:** Shows sections for Project, Commit, PR, Structure, ..., Services, Terminal, Problems, and Git.
- Status Bar:** Tests unitaires

DOSSIER PROFESSIONNEL (DP)



The screenshot shows a Java code editor with the file `FirestationControllerTest.java` open. The code is a unit test for a controller. It imports various dependencies and sets up two mock firestations. It then adds these firestations to a list and creates a person object with specific details.

```
package com.safetynetalerts.demo.controller;

import ...;

@SpringBootTest
class FirestationControllerTest {

    @MockBean
    FirestationService firestationService;

    @MockBean
    FirestationRepository firestationRepository;

    @MockBean
    PersonRepository personRepository;

    @Autowired
    FirestationController firestationController;

    Firestation firestation1 = new Firestation(); 10 usages
    Firestation firestation2 = new Firestation(); 3 usages

    List<Person> people = new ArrayList<>(); 4 usages
    List <Firestation> firestations = new ArrayList<>(); 3 usages

    @BeforeEach
    void setUp() {
        firestation1.setStation("1");
        firestation1.setAddress("123 rue de la paix");

        firestation2.setStation("2");
        firestation2.setAddress("456 rue des mines");

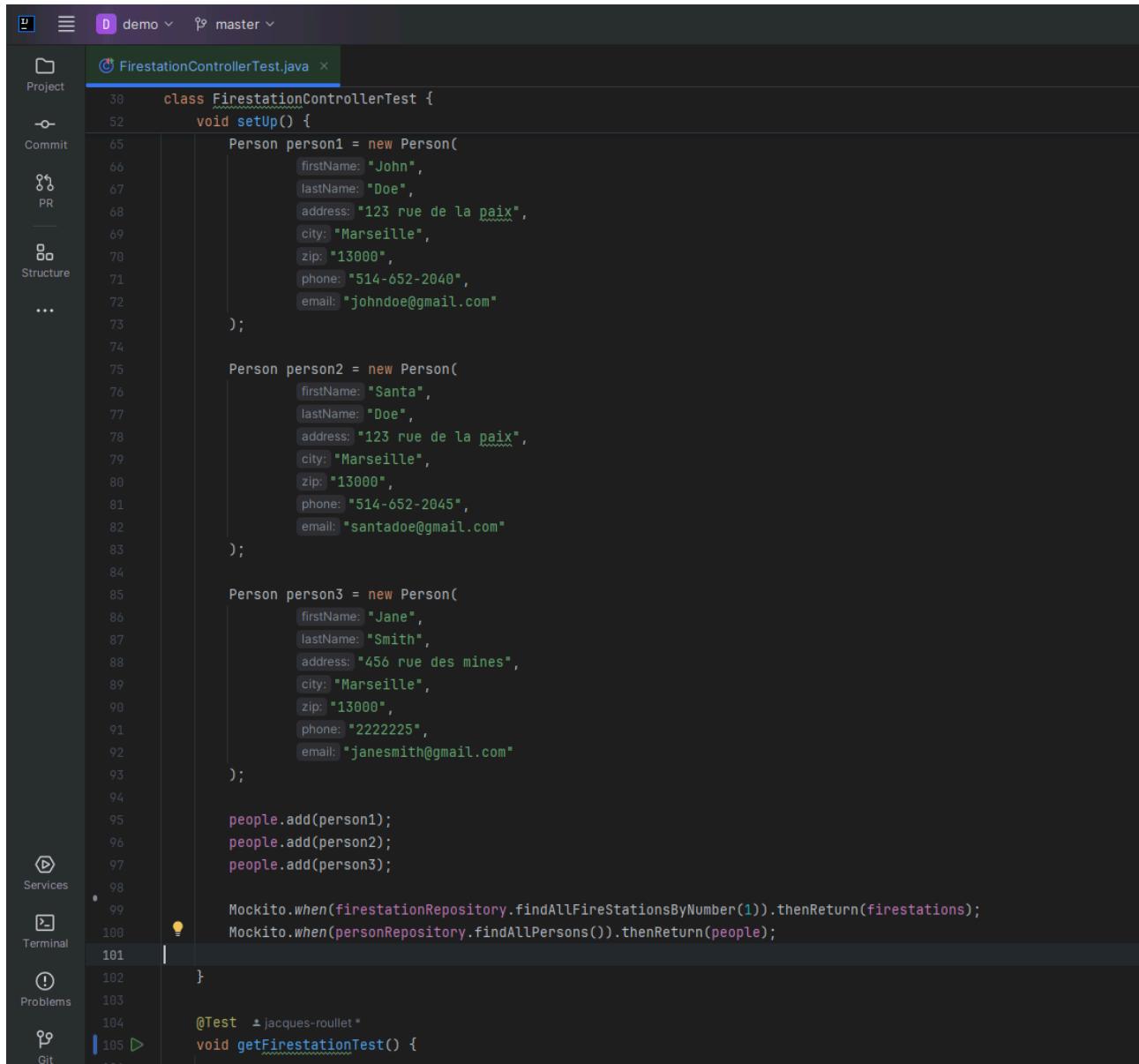
        firestations.add(firestation1);
        firestations.add(firestation2);

        Person person1 = new Person(
            firstName: "John",
            lastName: "Doe",
            address: "123 rue de la paix",
        );
    }
}
```

The sidebar on the left shows project navigation with sections like Project, Commit, PR, Structure, Services, Terminal, Problems, and Git. The status bar at the bottom shows the file path: demo > src > test > java > com > safetynetalerts > demo > controller > FirestationControllerTest > hearthListByFirestationTest.

Tests unitaires

DOSSIER PROFESSIONNEL (DP)

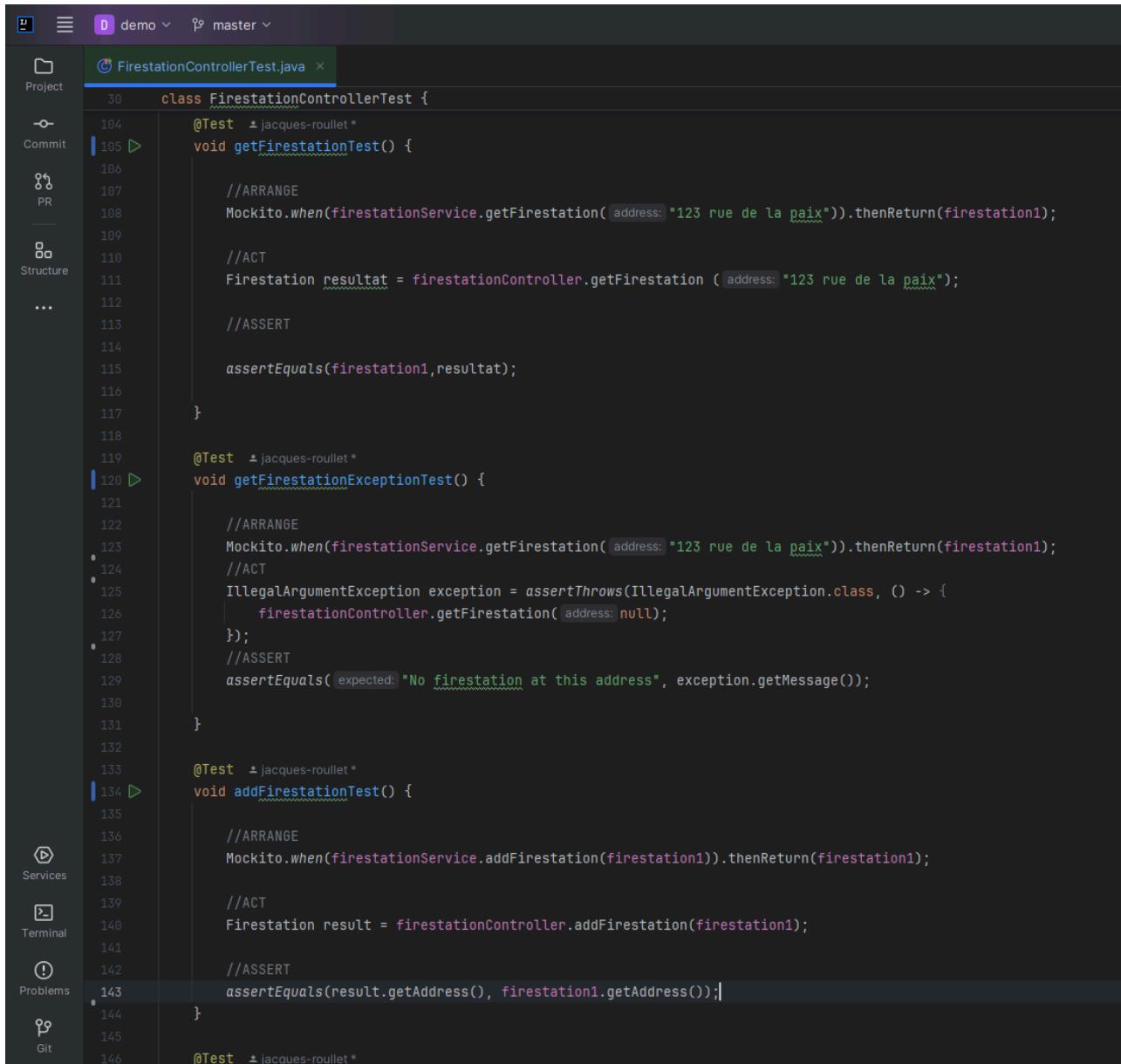


The screenshot shows a Java code editor with a dark theme. The file being edited is `FirestationControllerTest.java`. The code is a unit test for a `FirestationController`. It uses Mockito to mock repository objects and verify their interactions. The code creates three `Person` objects and adds them to a `people` list. It then calls methods on the `firestationRepository` and `personRepository` to check if they return the expected results.

```
class FirestationControllerTest {
    void setup() {
        Person person1 = new Person(
            firstName: "John",
            lastName: "Doe",
            address: "123 rue de la paix",
            city: "Marseille",
            zip: "13000",
            phone: "514-652-2040",
            email: "johndoe@gmail.com"
        );
        Person person2 = new Person(
            firstName: "Santa",
            lastName: "Doe",
            address: "123 rue de la paix",
            city: "Marseille",
            zip: "13000",
            phone: "514-652-2045",
            email: "santadoe@gmail.com"
        );
        Person person3 = new Person(
            firstName: "Jane",
            lastName: "Smith",
            address: "456 rue des mines",
            city: "Marseille",
            zip: "13000",
            phone: "2222225",
            email: "janessmith@gmail.com"
        );
        people.add(person1);
        people.add(person2);
        people.add(person3);
    }
    @Test
    void getFirestationTest() {
        Mockito.when(firestationRepository.findAllFireStationsByNumber(1)).thenReturn(firestations);
        Mockito.when(personRepository.findAllPersons()).thenReturn(people);
    }
}
```

Tests unitaires

DOSSIER PROFESSIONNEL (DP)



```
class FirestationControllerTest {  
    @Test // jacques-roulet*  
    void getfirestationTest() {  
        //ARRANGE  
        Mockito.when(firestationService.getFirestation( address: "123 rue de la paix")).thenReturn(firestation1);  
  
        //ACT  
        Firestation resultat = firestationController.getFirestation ( address: "123 rue de la paix");  
  
        //ASSERT  
        assertEquals(firestation1,resultat);  
    }  
  
    @Test // jacques-roulet*  
    void getFirestationExceptionTest() {  
        //ARRANGE  
        Mockito.when(firestationService.getFirestation( address: "123 rue de la paix")).thenReturn(firestation1);  
        //ACT  
        IllegalArgumentException exception = assertThrows(IllegalArgumentException.class, () -> {  
            firestationController.getFirestation( address: null);  
        });  
        //ASSERT  
        assertEquals( expected: "No firestation at this address", exception.getMessage());  
    }  
  
    @Test // jacques-roulet*  
    void addFirestationTest() {  
        //ARRANGE  
        Mockito.when(firestationService.addFirestation(firestation1)).thenReturn(firestation1);  
  
        //ACT  
        Firestation result = firestationController.addFirestation(firestation1);  
  
        //ASSERT  
        assertEquals(result.getAddress(), firestation1.getAddress());  
    }  
}
```

Tests unitaires

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The project is named "demo" and contains a "master" branch. It includes a "src" folder with "test", "java", "model", "repository", "service", and "applicationTests" subfolders.
- Code Editor:** The code editor shows a Java test class named `FirestationControllerTest`. The code imports various Spring Boot and Mockito annotations, and uses `FirestationController` and `FirestationService` from the `com.safetynetaerts.demo` package.
- Run Tab:** The "Run" tab is selected, showing the results of the `FirestationControllerTest`. It indicates 8 tests passed in 118ms.
- Terminal:** The terminal shows the command-line output of the test execution, including log messages from Spring Boot and the test results.

Tests unitaires

The screenshot shows the Swagger UI interface displaying the API documentation for the application. Key elements include:

- Header:** Shows the URL `localhost:8080/swagger-ui/index.html#`.
- Api Documentation:** A main heading with links to "Api Documentation", "Terms of service", and "Apache 2.0".
- Servers:** A dropdown menu set to `http://localhost:8080 - inferred Url`.
- API Endpoints:** A list of controllers and their methods:
 - basic-error-controller:** Basic Error Controller (no methods listed).
 - firestation-controller:** Firestation Controller (no methods listed).
 - medical-records-controller:** Medical Records Controller
 - POST /medicalRecord/add:** addPerson
 - DELETE /medicalRecord/delete:** deletePerson
 - GET /medicalRecord/get:** getMedicalRecords
 - PUT /medicalRecord/update:** updatePerson
 - person-controller:** Person Controller (no methods listed).

Swagger

DOSSIER PROFESSIONNEL (DP)

Swagger UI

localhost:8080/swagger-ui/index.html#/medical-records-controller/getMedicalRecordsUsingGET

GET /medicalRecord/get getMedicalRecords

Parameters

Name	Description
firstName <small>required</small>	firstName
string	(query)
John	
lastName <small>required</small>	lastName
string	(query)
Boyd	

Responses

Curl

```
curl -X GET "http://localhost:8080/medicalRecord/get?firstName=John&lastName=Boyd" -H "accept: */*
```

Request URL

<http://localhost:8080/medicalRecord/get?firstName=John&lastName=Boyd>

Server response

Code	Details
200	Response body

```
{
  "firstName": "John",
  "lastName": "Boyd",
  "birthdate": "03/06/1984",
  "medications": [
    "azmoli350mg",
    "hydрапеназол:100mg"
  ],
  "allergies": [
    "аллергия"
  ]
}
```

Execute Clear

Swagger

Swagger UI

localhost:8080/swagger-ui/index.html#/medical-records-controller/getMedicalRecordsUsingGET

200 Response body

```
{
  "firstName": "John",
  "lastName": "Boyd",
  "birthdate": "03/06/1984",
  "medications": [
    "azmoli350mg",
    "hydрапеназол:100mg"
  ],
  "allergies": [
    "аллергия"
  ]
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: 08 Apr 2025 14:43:50 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links
Media type	*/*	
Controls Accept header.		
Example Value	Schema	
	{ "allergies": [], "birthdate": "string", "firstName": "string", "lastName": "string", "medications": ["string"] }	
401	Unauthorized	No links
403	Forbidden	No links
404	Not Found	No links

Swagger

DOSSIER PROFESSIONNEL (DP)

```
2025-04-08T16:21:19.537+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.MsUserApplicationTests : Starting MsUserApplicationTests using Java 17.0.14
2025-04-08T16:21:19.540+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.MsUserApplicationTests : No active profile set, falling back to 1 default pr
2025-04-08T16:21:20.552+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.service.TourGuideService : TestMode enabled
2025-04-08T16:21:20.584+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.MsUserApplicationTests : Using custom MathContext: precision=256, rounding=M
2025-04-08T16:21:21.339+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.MsUserApplicationTests : Started MsUserApplicationTests in 2.17 seconds (prc
100000
2025-04-08T16:21:22.440+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.service.TourGuideService : TestMode enabled
highVolumeTrackLocation: Time Elapsed: 72 seconds.
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
> Task :test
BUILD SUCCESSFUL in 1m 2s
4 actionable tasks: 2 executed, 2 up-to-date
16:22:35: Execution finished ':test --tests "jroulet83.ms_user.MsUserApplicationTests.highVolumeTrackLocation"'.
```

Test de charge

```
2025-04-08T16:21:19.537+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.InternalTestHelper.java : Starting highVolumeGetRewardsIT
2025-04-08T16:21:19.540+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.InternalTestHelper.java : No active profile set, falling back to 1 default pr
2025-04-08T16:21:20.552+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.InternalTestHelper.java : TestMode enabled
2025-04-08T16:21:20.584+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.InternalTestHelper.java : Using custom MathContext: precision=256, rounding=M
2025-04-08T16:21:21.339+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.InternalTestHelper.java : Started highVolumeGetRewardsIT in 2.17 seconds (prc
100000
2025-04-08T16:21:22.440+02:00 INFO 17688 --- [ms-user] [Test worker] j.ms_user.UserService : TestMode enabled
highVolumeGetRewards: Time Elapsed: 66 seconds.
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
> Task :test
BUILD SUCCESSFUL in 1m 14s
4 actionable tasks: 2 executed, 2 up-to-date
16:49:27: Execution finished ':test --tests "jroulet83.ms_reward.highVolumeGetRewardsIT.highVolumeGetRewards"'.
```

Test de charge

DOSSIER PROFESSIONNEL (DP)

SafetyNetAlerts

localhost:63342/demo/target/site/jacoco/index.html?_jt=kjgsmh5eifiob06bv6utnrrfm&_ij_reload=RELOAD_ON_SAVE

SafetyNetAlerts

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.safetynetalerts.demo.service.dto	63 %		n/a	71 %	23	72	40	129	23	72	0	7
com.safetynetalerts.demo.repository	76 %			88 %	14	39	8	56	10	32	0	4
com.safetynetalerts.demo.service	99 %			100 %	8	74	3	207	0	38	0	3
com.safetynetalerts.demo.model	97 %			n/a	1	44	2	87	1	43	0	4
com.safetynetalerts.demo	37 %				1	2	2	3	1	2	0	1
com.safetynetalerts.demo.controller	100 %			100 %	0	23	0	36	0	22	0	3
Total	198 of 1 878	89 %	12 of 90	86 %	47	254	55	518	35	209	0	22

Jacoco - taux de couverture

localhost:63342/demo/target/site/jacoco/com.safetynetalerts.demo.service/FirestationService.java.html#L60

```

22.
23. @Service
24. public class FirestationService {
25.
26.     private final FirestationRepository firestationRepository;
27.     private final PersonRepository personRepository;
28.     private final MedicalRecordsRepository medicalRecordsRepository;
29.
30.     public FirestationService(FirestationRepository firestationRepository, PersonRepository personRepository, MedicalRecordsRepository medicalRecordsRepository) {
31.         this.firestationRepository = firestationRepository;
32.         this.personRepository = personRepository;
33.         this.medicalRecordsRepository = medicalRecordsRepository;
34.     }
35.
36.     public List<String> findPhoneNumbersByStationNumber(int number) {
37.         List<String> result = new ArrayList<>();
38.         List<Firestation> firestations = firestationRepository.findAllFireStationsByNumber(number);
39.         List<Person> persons = personRepository.findAllPersons();
40.
41.         for (Person person : persons) {
42.             if (personContainsFirestationAddress(firestations, person)) {
43.                 result.add(person.getPhone());
44.             }
45.         }
46.         return result;
47.     }
48.
49.     public boolean personContainsFirestationAddress(List<Firestation> firestations, Person person) {
50.         for (Firestation firestation : firestations) {
51.             if (firestation.getAddress().equals(person.getAddress())) {
52.                 return true;
53.             }
54.         }
55.         return false;
56.     }
57.
58.     public List<String> addressesDeliveredByFirestation(int number) {
59.         List<Firestation> firestations = firestationRepository.findAllFireStations();
60.         List<Person> people = personRepository.findAllPersons();
61.
62.         List<String> addresses = new ArrayList<>();
63.
64.         // Récupérer les adresses des stations de pompiers en fonction de leur numéro
65.         for (Firestation firestation : firestations) {
66.             if (firestation.getNumber().equals(String.valueOf(number))) {
67.                 for (Person person : people) {
68.                     // Vérifier si la personne a la même adresse que la station de pompiers et ajoutez cette adresse
69.                     if (person.getAddress().equals(firestation.getAddress()) && !addresses.contains(person.getAddress())) {
70.                         addresses.add(person.getAddress());
71.                     }
72.                 }
73.             }
74.         }
75.         return addresses;
76.     }
77.
78.     // test stream exemple
79.     public String fireStationNumberAtThisAddress(String address) {
80.         List<Firestation> firestations = firestationRepository.findAllFireStations();
81.         return firestations.stream().filter(f -> f.getAddress().equals(address))
82.             .map(Firestation :: getStation).findFirst().orElse(null);
83.     }
84.
85.     public FirestationDTO findPeopleCountByFirestation(int number) {
86.         FirestationDTO result = new FirestationDTO();
87.         List<FirestationPersonDTO> people = new ArrayList<>();

```

Jacoco - taux de couverture

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

J'ai développé ces composants en autonomie dans le cadre de ma formation, en m'appuyant sur la documentation officielle de Spring et des ressources pédagogiques.

J'ai également sollicité ponctuellement l'aide de mon formateur pour valider certaines décisions techniques afin de m'aider à anticiper les contraintes qui en découlent.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - SafetyNet, TourGuide

Période d'exercice - Du 15/11/2024 au 28/02/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

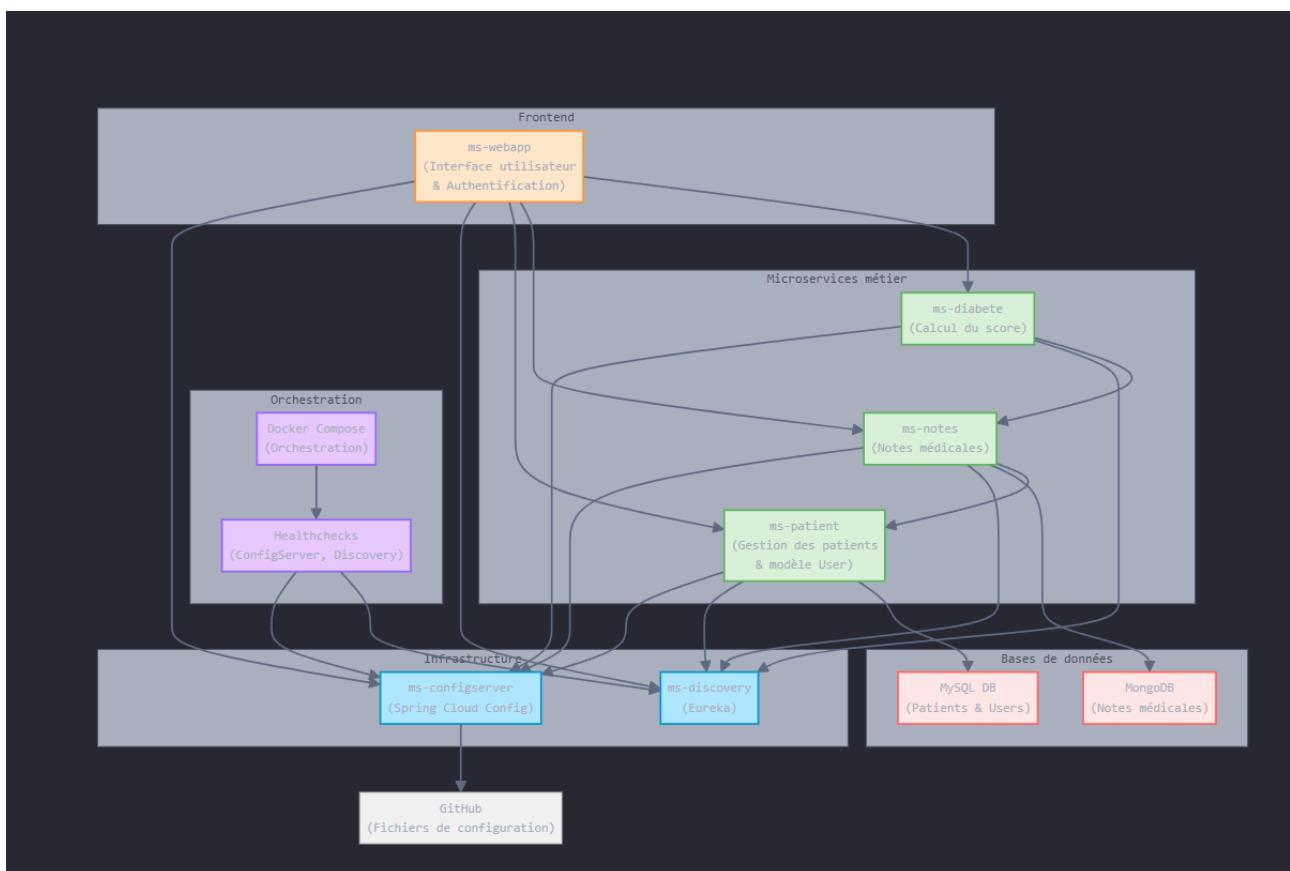
Activité-type 3 Préparer le déploiement d'une application sécurisée

Exemple n°2 - 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 de mon projet *HealthCare*, j'ai préparé et documenté le processus de déploiement d'une application distribuée en microservices, conteneurisée avec Docker.

Voici un aperçu de l'organisation des microservices :

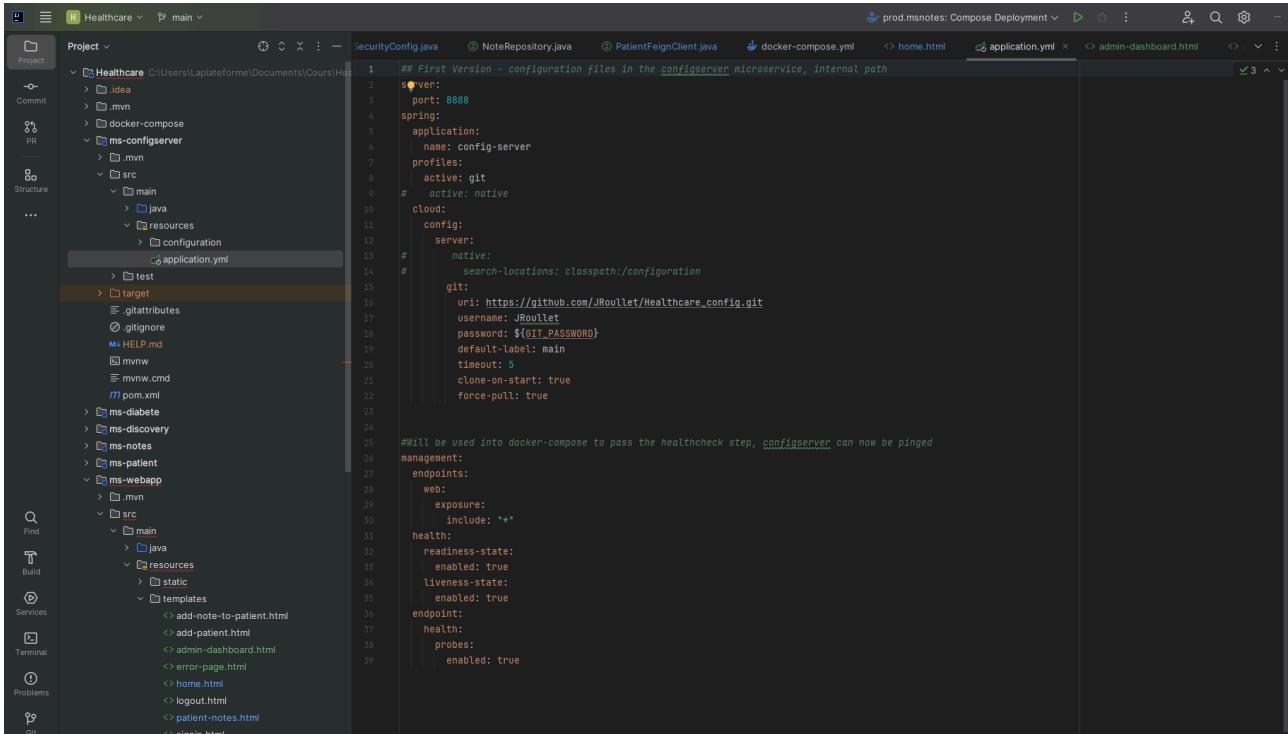


DOSSIER PROFESSIONNEL (DP)

- Configuration centralisée : récupération des configurations de chaque microservice depuis le Config Server (Spring Cloud)
- L'enregistrement des microservices dans Eureka pour pouvoir communiquer entre microservices (DiscoveryClient)
- Préparation des packages (.jar) validés et fonctionnels grâce aux cycles de vie fournis par Maven
- Construction des images avec JIB (Maven Plugin), sans nécessiter de Dockerfile
- Mise en place de healthchecks pour configserver et discovery
- Dépendances techniques (ex: ms-patient dépend de patientdB)
- Sécurité (identifiants pour Discovery et ConfigServer, secrets pour la connexion au serveur distant(Github))
- Orchestration du déploiement avec docker-compose (définition des ports et des dépendances)
- Commandes Docker
- Communication inter services : Feign

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)

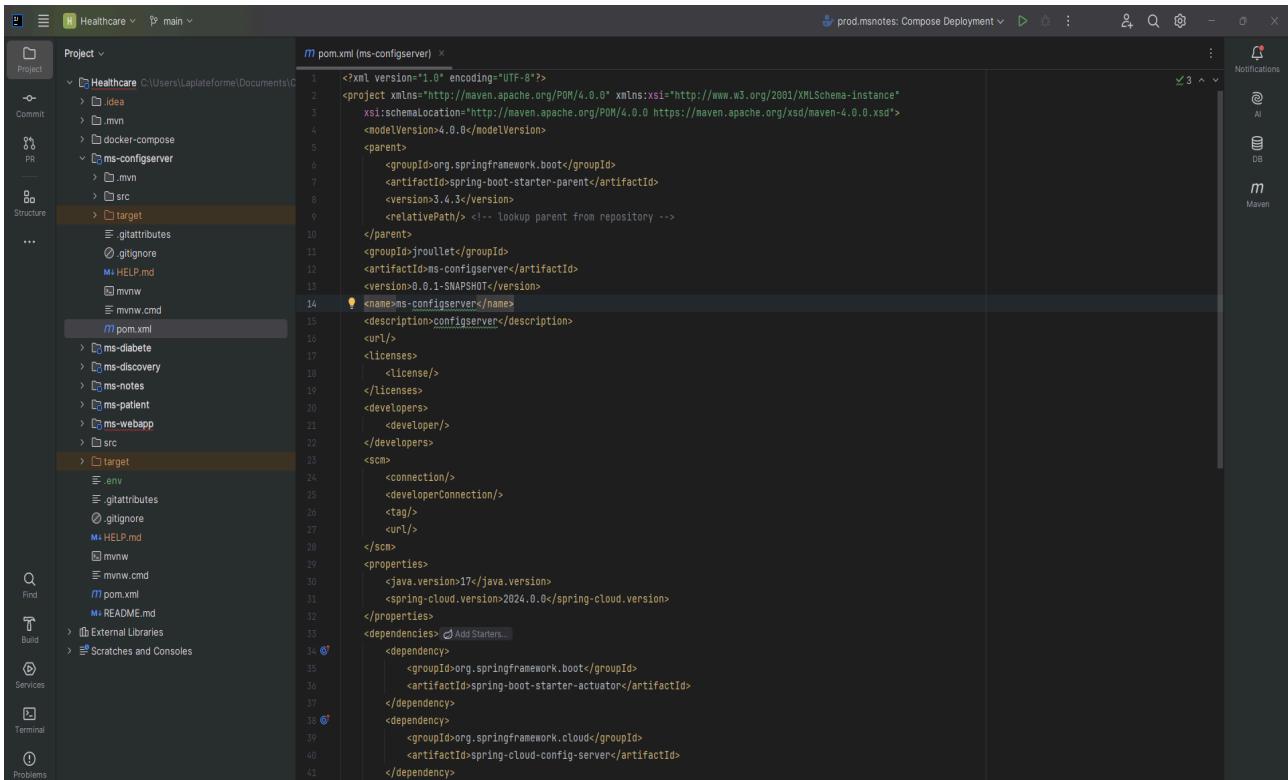


The screenshot shows a code editor with the file `application.yml` selected. The code defines a configuration server named `config-server` using Git as the active provider. It specifies port `8888`, a native cloud configuration, and a git endpoint for the `configserver` microservice.

```
## First Version - configuration files in the configserver microservice, internal path
server:
  port: 8888
spring:
  application:
    name: config-server
  profiles:
    active: git
  # active: native
  cloud:
    config:
      server:
        native:
          search_LOCATIONS: classpath:/configuration
  git:
    uri: https://github.com/JRoulet/Healthcare_config.git
    username: JRoulet
    password: $GIT_PASSWORD
    default_label: main
    timeout: 5
    clone_on_start: true
    force_pull: true

## Will be used into docker-compose to pass the healthcheck step, configserver can now be pinged
management:
  endpoints:
    web:
      exposure:
        include: ***
  health:
    readiness_state:
      enabled: true
    liveness_state:
      enabled: true
  endpoint:
    health:
      probes:
        enabled: true
```

Configuration : ConfigServer



The screenshot shows a code editor with the file `pom.xml` selected for the `ms-configserver` project. The pom file includes dependencies for Spring Boot Starter Parent, Spring Cloud Config Server, and Spring Cloud Actuator. It also defines a scm section for version control and properties for Java version and Spring Cloud version.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.3</version>
    <relativePath><!-- lookup parent from repository --&gt;
  &lt;/parent&gt;
  &lt;groupId&gt;jroulet&lt;/groupId&gt;
  &lt;artifactId&gt;ms-configserver&lt;/artifactId&gt;
  &lt;version&gt;0.0.1-SNAPSHOT&lt;/version&gt;
  &lt;name&gt;ms-configserver&lt;/name&gt;
  &lt;description&gt;configserver&lt;/description&gt;
  &lt;url&gt;&lt;/url&gt;
  &lt;licenses&gt;
    &lt;license&gt;&lt;/license&gt;
  &lt;/licenses&gt;
  &lt;developers&gt;
    &lt;developer&gt;&lt;/developer&gt;
  &lt;/developers&gt;
  &lt;scm&gt;
    &lt;connection&gt;&lt;/connection&gt;
    &lt;developerConnection&gt;&lt;/developerConnection&gt;
    &lt;tag&gt;&lt;/tag&gt;
    &lt;url&gt;&lt;/url&gt;
  &lt;/scm&gt;
  &lt;properties&gt;
    &lt;java.version&gt;17&lt;/java.version&gt;
    &lt;spring-cloud.version&gt;2024.0.0&lt;/spring-cloud.version&gt;
  &lt;/properties&gt;
  &lt;dependencies&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
      &lt;artifactId&gt;spring-boot-starter-actuator&lt;/artifactId&gt;
    &lt;/dependency&gt;
    &lt;dependency&gt;
      &lt;groupId&gt;org.springframework.cloud&lt;/groupId&gt;
      &lt;artifactId&gt;spring-cloud-config-server&lt;/artifactId&gt;
    &lt;/dependency&gt;</pre>
```

Spring Cloud configserver (pom.xml)

DOSSIER PROFESSIONNEL (DP)

The screenshot shows a code editor window with the file `ms-discovery.yml` open. The file contains configuration for an Eureka server, including details about instances, actuator endpoints, and health checks. The code is as follows:

```
server:
  port: 8761
#EUREKA Server -- Local Configuration
eureka:
  instance:
    hostname: localhost
  client:
    fetchRegistry: false
    registerWithEureka: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
  #ACTUATOR
  management:
    endpoints:
      web:
        exposure:
          include: "*"
      health:
        readiness-state:
          enabled: true
        liveness-state:
          enabled: true
      endpoint:
        health:
          probes:
            enabled: true
```

The sidebar on the left shows a project structure for "Healthcare_config" with files like `ms-discovery.yml`, `ms-patient.yml`, and `ms-webapp.yml`. The bottom status bar indicates the file is 1183 of 4000M.

Configuration à distance depuis Git (service Eurêka)

The screenshot shows a code editor window with the file `pom.xml` open for the `ms-webapp` module. The pom file includes dependencies for Spring Boot, Cloud, and Feign clients, along with various annotations and configurations. The code is as follows:

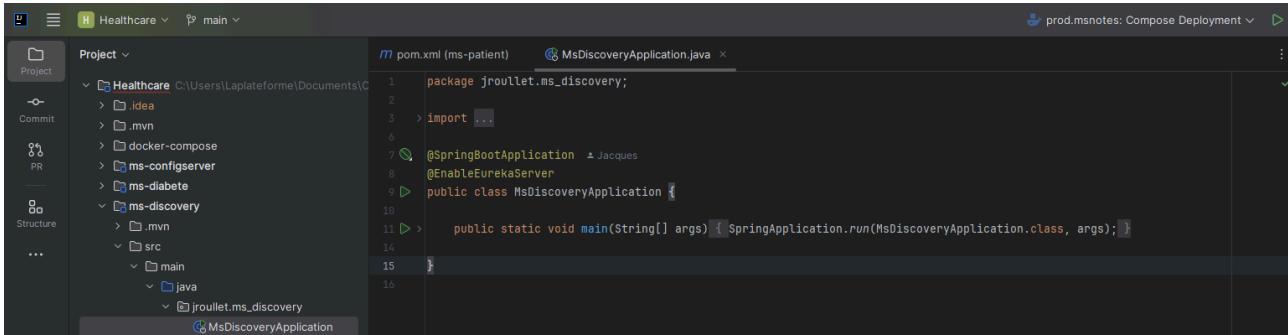
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.0.RELEASE</version>
    <relativePath>/</relativePath>
  </parent>
  <groupId>com.mswebapp</groupId>
  <artifactId>ms-webapp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ms-webapp</name>
  <description>Microservice Web Application</description>
  <packaging>jar</packaging>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-config</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>org.springframework.boot:spring-boot-devtools</exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <repositories>
    <repository>
      <id>central</id>
      <url>https://repo.maven.apache.org/maven2/</url>
    </repository>
  </repositories>
  <distributionManagement>
    <snapshotRepository>
      <id>ossrh-snapshots</id>
      <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
    </snapshotRepository>
    <releaseRepository>
      <id>ossrh-releases</id>
      <url>https://oss.sonatype.org/content/repositories/releases/</url>
    </releaseRepository>
  </distributionManagement>

```

The sidebar on the left shows a project structure for "Healthcare" with modules like `ms-discovery`, `ms-notes`, `ms-patient`, and `ms-webapp`. The bottom status bar indicates the file is 1834 of 4000M.

WebApp - Eurêka client + Feign Client - pom.xml

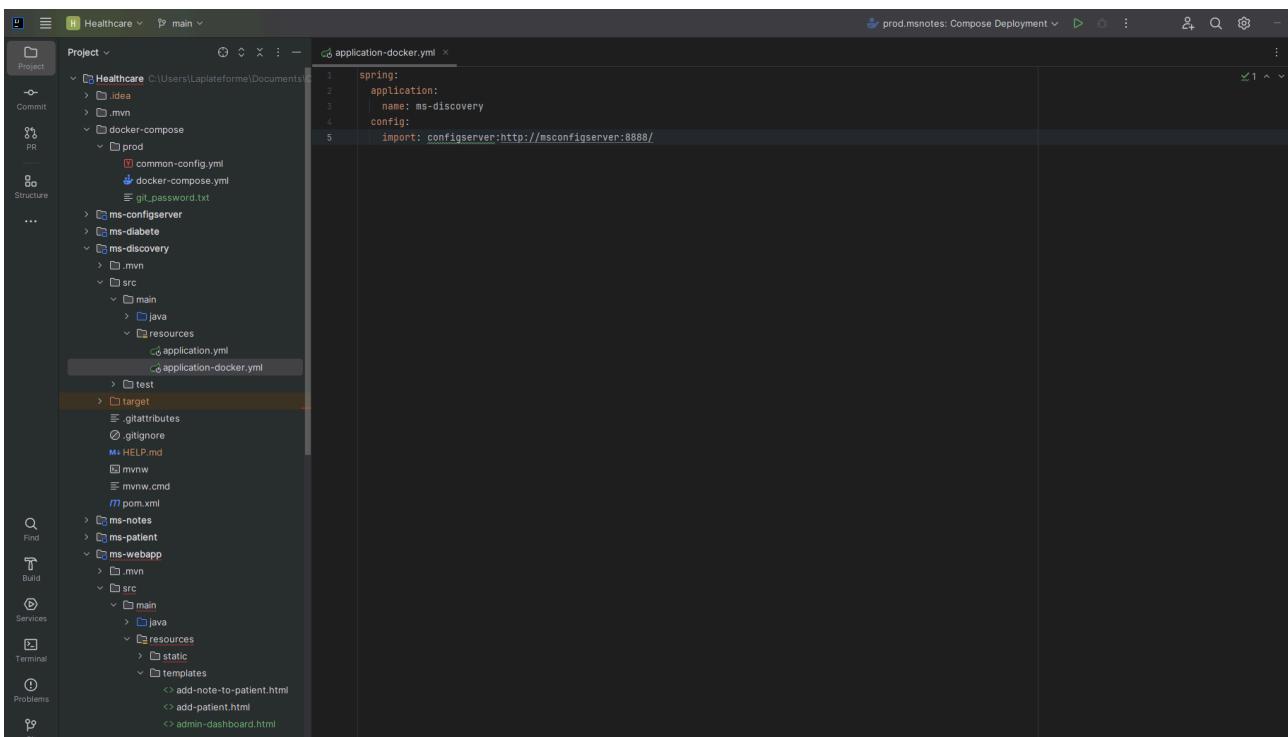
DOSSIER PROFESSIONNEL (DP)



MsDiscoveryApplication.java

```
package jrouillet.ms_discovery;
import ...;
@SpringBootApplication
@EnableEurekaServer
public class MsDiscoveryApplication {
    public static void main(String[] args) { SpringApplication.run(MsDiscoveryApplication.class, args); }
}
```

MsDiscovery - Eurêka Server

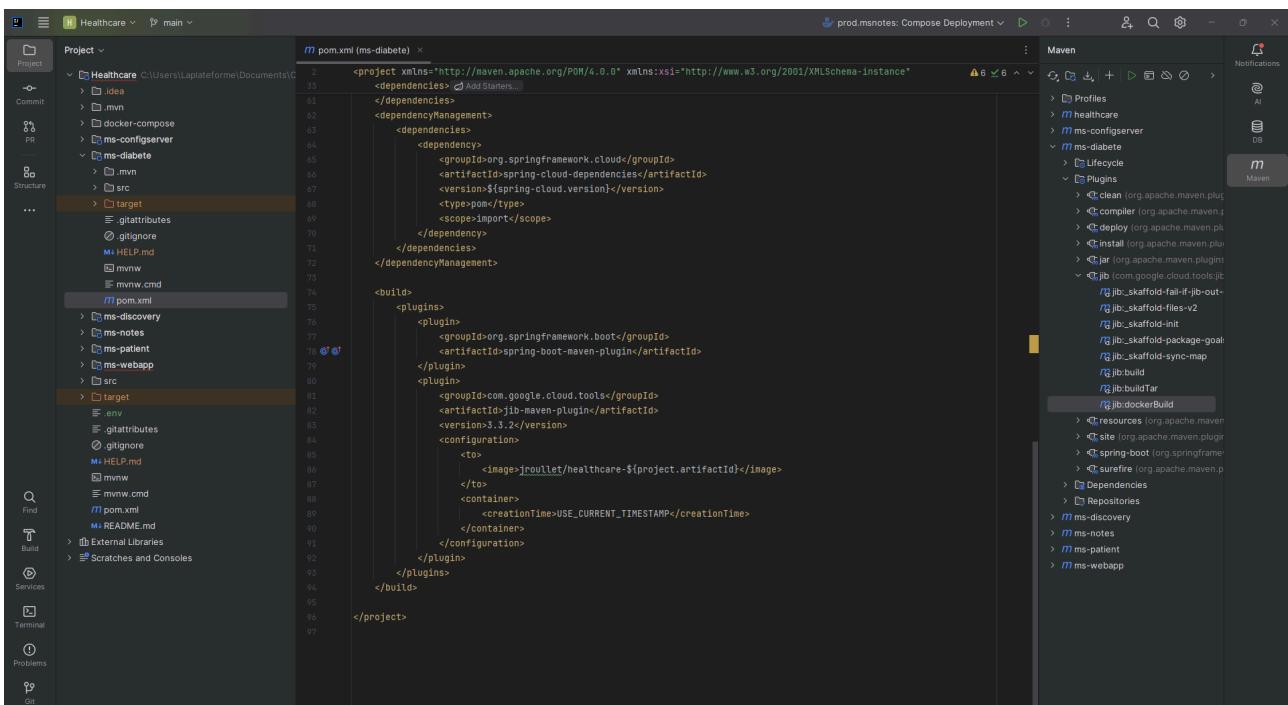


application-docker.yml

```
spring:
  application:
    name: ms-discovery
  config:
    import: configserver:http://msconfigserver:8888/
```

Configuration Eurêka (Docker)

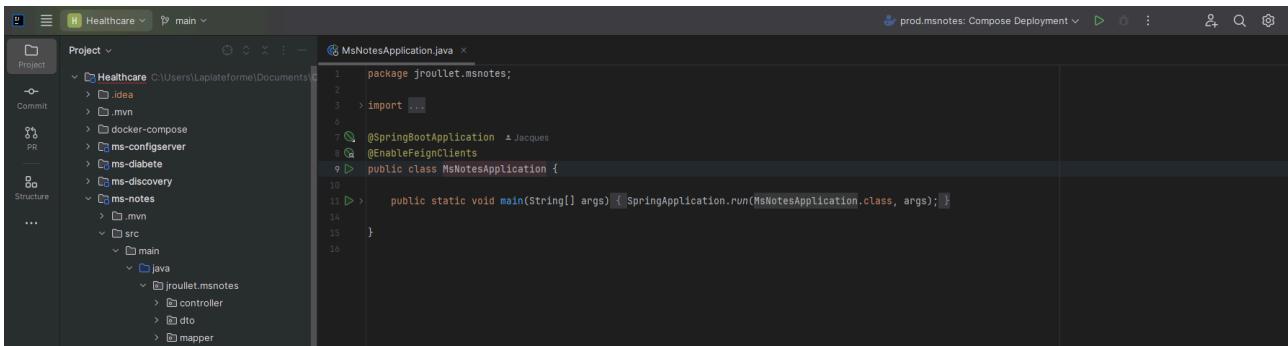
DOSSIER PROFESSIONNEL (DP)



The screenshot shows the IntelliJ IDEA interface with the project 'Healthcare' selected. The left sidebar displays the project structure, including modules like ms-diabète, ms-discovery, ms-notes, ms-patient, and ms-webapp. The right panel shows the content of the pom.xml file for the ms-diabète module. The code defines dependencies on org.springframework.cloud and org.springframework.boot-maven-plugin, and includes a build section with a jib-maven-plugin configuration for creating a Docker image named 'jrouillet/healthcare-\$[project.artifactId]'. The Maven tool window on the right shows various build-related options.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <dependencies> <!-- Add Starters... -->
    <dependencyManagement>
      <dependencies>
        <dependency>
          <groupId>org.springframework.cloud</groupId>
          <artifactId>spring-cloud-dependencies</artifactId>
          <version>${spring-cloud.version}</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
      </dependencies>
    </dependencyManagement>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>com.google.cloud.tools</groupId>
        <artifactId>jib-maven-plugin</artifactId>
        <version>3.3.2</version>
        <configuration>
          <to>
            <image>jrouillet/healthcare-$[project.artifactId]</image>
          </to>
          <container>
            <creationTime>USE_CURRENT_TIMESTAMP</creationTime>
          </container>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Création de l'image - JIB

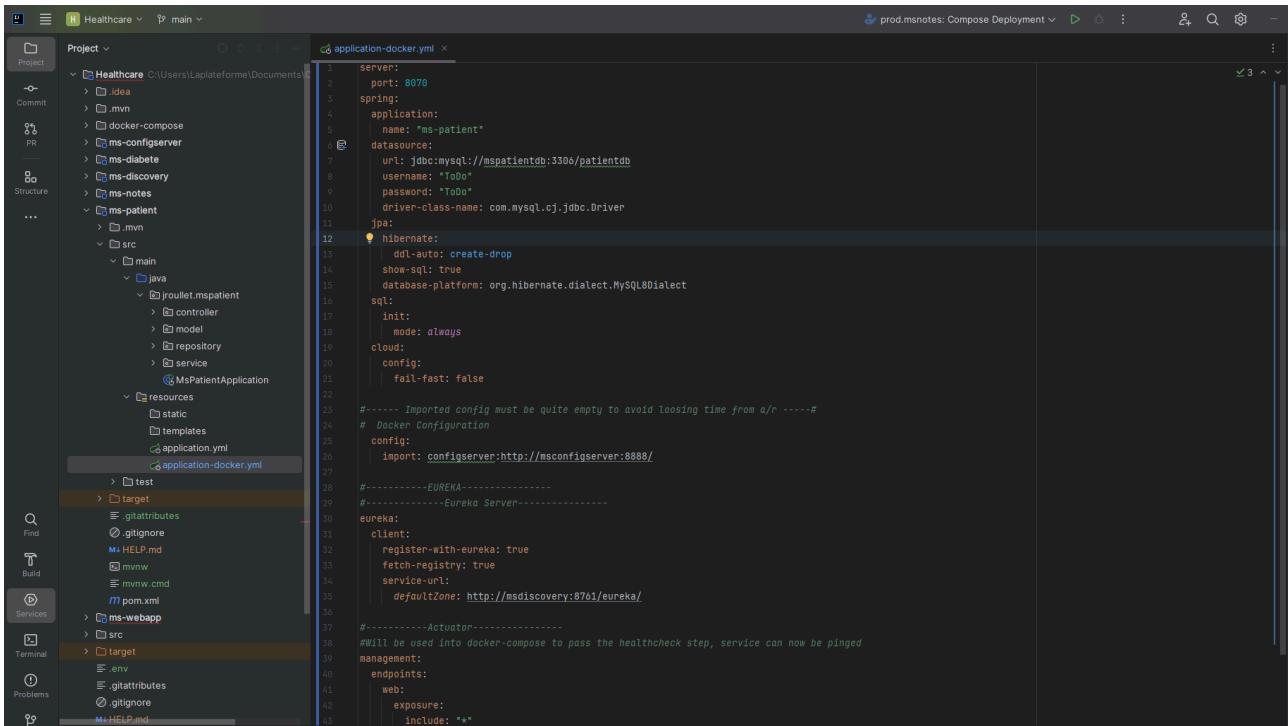


The screenshot shows the IntelliJ IDEA interface with the project 'Healthcare' selected. The left sidebar displays the project structure, including modules like ms-diabète, ms-discovery, ms-notes, ms-patient, and ms-webapp. The right panel shows the content of the MsNotesApplication.java file. The code defines a main class 'MsNotesApplication' with a static main method that runs the application using SpringApplication.

```
package jrouillet.msnotes;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class MsNotesApplication {
    public static void main(String[] args) { SpringApplication.run(MsNotesApplication.class, args); }
}
```

Feign Clients - communication inter services

DOSSIER PROFESSIONNEL (DP)

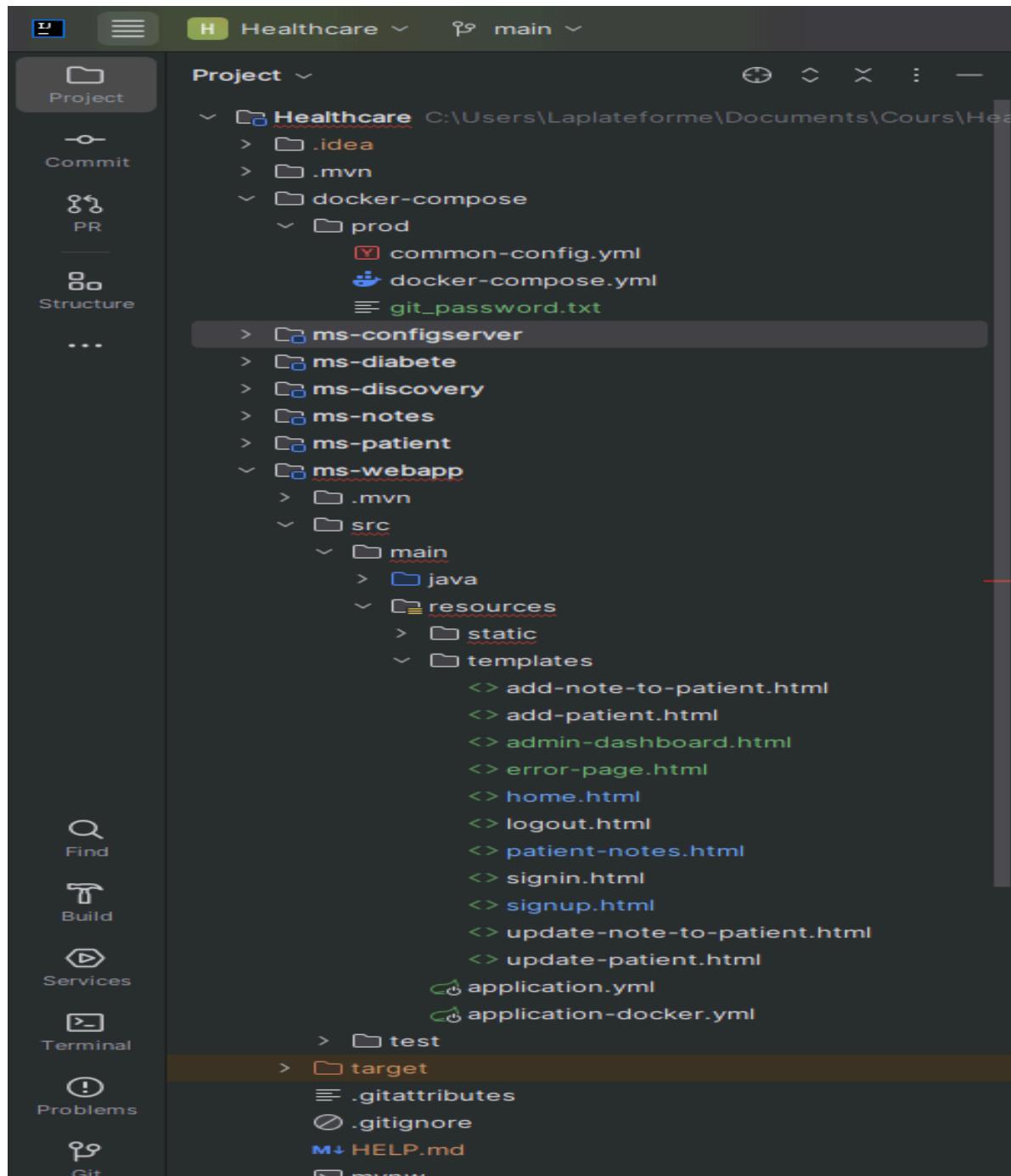


The screenshot shows a Java Spring Boot project structure in a code editor. The project is named "Healthcare" and contains several modules: ms-disease, ms-discovery, ms-notes, and ms-patient. The ms-patient module is expanded, showing its src directory which includes main, resources, and application.yml. The application-docker.yml file is open in the editor. It defines a server port of 8070, a spring application name of "ms-patient", and a datasource configuration for MySQL. It also includes configurations for JPA, hibernate, and Eureka. The Docker configuration section imports configuration from a configserver at http://msconfigserver:8888. The Actuator management section is also present.

```
server:
  port: 8070
spring:
  application:
    name: "ms-patient"
  datasource:
    url: jdbc:mysql://mspatientdb:3306/patientdb
    username: "Todo"
    password: "Todo"
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: create-drop
      show-sql: true
      database-platform: org.hibernate.dialect.MySQL8Dialect
    sql:
      init:
        mode: always
      cloud:
        config:
          fail-fast: false
#----- Imported config must be quite empty to avoid loosing time from s/r -----#
# Docker Configuration
config:
  import: configserver:http://msconfigserver:8888/
#-----EUREKA-----
#-----Eureka Server-----
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://msdiscovery:8761/eureka/
#-----Actuator-----
#Will be used into docker-compose to pass the healthcheck step, service can now be pinged
management:
  endpoints:
    web:
      exposure:
        include: "*"
```

application-docker.yml (configuration-docker)

DOSSIER PROFESSIONNEL (DP)



3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL^(DP)

J'ai développé ces composants en autonomie dans le cadre de ma formation, en m'appuyant sur la documentation officielle de Spring et des ressources pédagogiques.

J'ai également sollicité ponctuellement l'aide de mon formateur pour valider certaines décisions techniques afin de m'aider à anticiper les contraintes qui en découlent.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - HealthCare

Période d'exercice - Du 01/03/2025 au 03/04/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL (DP)

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

Exemple n°3 - Contribuer à la mise en production dans une démarche DevOps

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

- Rédaction d'un docker-compose
- Rédaction d'un common-config, certains segments du docker-compose vont hériter du common-config pour réduire les répétitions de code (sections extends)
- Git_password.txt (permet de ne pas afficher les informations d'authentification en clair dans le code)
- Commandes docker

2. Précisez les moyens utilisés :

```
docker-compose.yml ×
1  services:
2
3    msconfigserver:
4      image: jrouillet/healthcare-ms-configserver:latest
5      container_name: configserver-ms
6      ports:
7        - "8888:8888"
8      healthcheck:
9        test: "curl --fail --silent localhost:8888/actuator/health/readiness | grep UP || exit 1"
10     interval: 10s
11     timeout: 5s
12     retries: 10
13     start_period: 10s
14     extends:
15       file: common-config.yml
16       service: microservice-base-config
17     secrets:
18       - git_password
19     environment:
20       GIT_PASSWORD_FILE: /run/secrets/git_password
21
```

docker-compose - configserver + healthcheck endpoint

DOSSIER PROFESSIONNEL (DP)

```
21
22 C msdiscovery:
23     image: jrouillet/healthcare-ms-discovery:latest
24     container_name: discovery-ms
25     ports:
26         - "8761:8761"
27     depends_on:
28         msconfigserver:
29             condition: service_healthy
30     healthcheck:
31         test: "curl --fail --silent localhost:8761/actuator/health/readiness | grep UP || exit 1"
32         interval: 10s
33         timeout: 5s
34         retries: 10
35         start_period: 10s
36     extends:
37         file: common-config.yml
38         service: microservice-configserver-config
39
```

docker-compose - msdiscovery + healthcheck + héritage

```
40 C mspatientdb:
41     container_name: patientdb
42     ports:
43         - "3308:3306"
44     environment:
45         MYSQL_DATABASE: patientdb
46     extends:
47         file: common-config.yml
48         service: microservice-db-config
```

docker-compose : BDD MySQL

DOSSIER PROFESSIONNEL (DP)

```
0 C mswebapp:
1   image: jrouurret/healthcare-ms-webapp:latest
2   container_name: webapp-ms
3   ports:
4     - "8090:8090"
5   depends_on:
6     mspatientdb:
7       condition: service_healthy
8   extends:
9     file: common-config.yml
10    service: microservice-discovery-config
11   environment:
12     SPRING_DATASOURCE_URL: jdbc:mysql://mspatientdb:3306/patientdb
13
14
```

docker-compose : mswebapp

```
microservice-discovery-config:
  extends:
    service:
      microservice-configserver-config
  depends_on:
    msconfigserver:
      condition: service_healthy
    msdiscovery:
      condition: service_healthy
  environment:
    EUREKA_CLIENT_SERVICEURL_DEFAULTZONE: http://msdiscovery:8761/eureka/
```

common-config (docker-compose)

DOSSIER PROFESSIONNEL (DP)

```
services:  
  network-deploy-service:  
    networks:  
      - healthcare  
  
  microservice-base-config:  
    extends:  
      service: network-deploy-service  
    deploy:  
      resources:  
        limits:  
          memory: 700m  
  
  microservice-configserver-config:  
    extends:  
      service: microservice-base-config  
  environment:  
    SPRING_PROFILES_ACTIVE: docker  
    SPRING_CONFIG_IMPORT: configserver:http://msconfigserver:8888/  
    SPRING_DATASOURCE_USERNAME: "todo"  
    SPRING_DATASOURCE_PASSWORD: "todo"
```

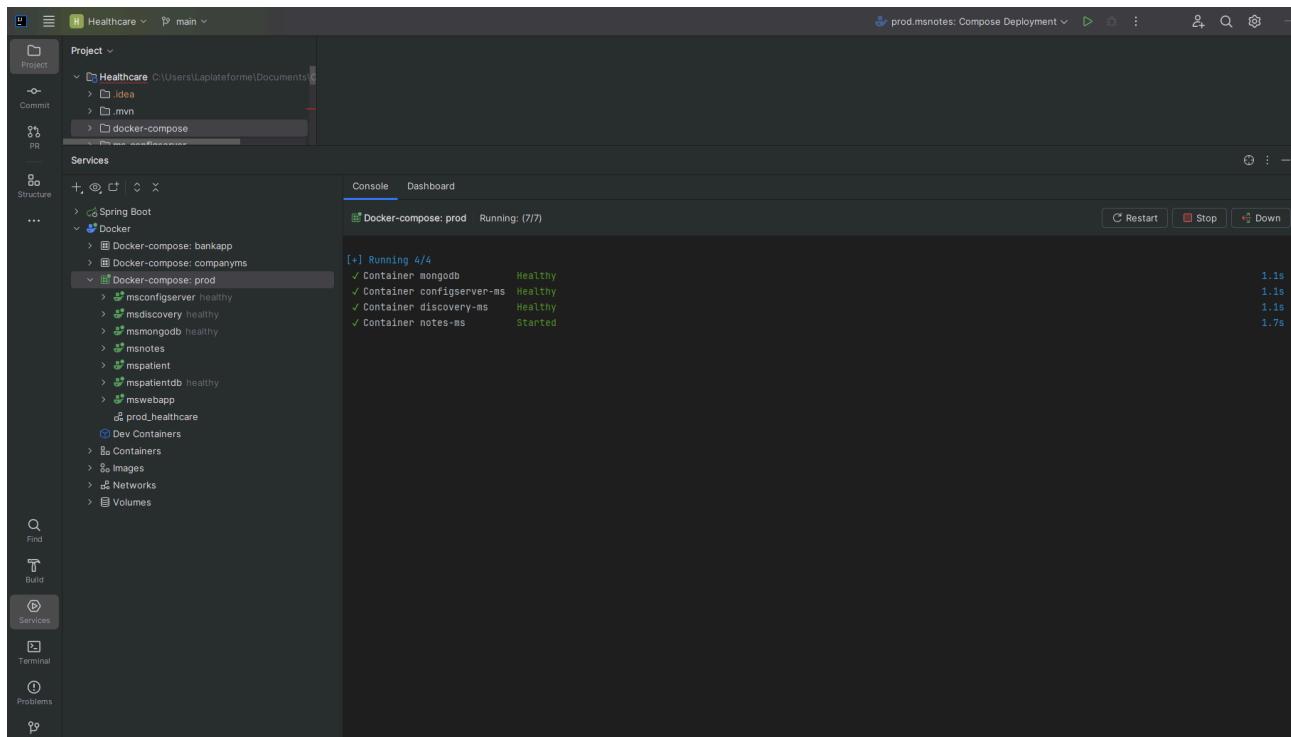
common-config (docker-compose)

DOSSIER PROFESSIONNEL (DP)

```
microservice-db-config:
  extends:
    service: network-deploy-service
  image: mysql
  healthcheck:
    test: [ "CMD", "mysqladmin" , "ping", "-h", "localhost" ]
    timeout: 10s
    retries: 10
    interval: 10s
    start_period: 10s
  environment:
    MYSQL_ROOT_PASSWORD: "todo"

microservice-mongo-config:
  extends:
    service: network-deploy-service
  image: mongo
  healthcheck:
    test: [ "CMD", "mongosh", "admin", "--username=todo", "--password=todo", "--eval", "db.runCommand({ ping: 1 })" ]
    interval: 10s
    timeout: 5s
    retries: 10
  start_period: 10s
```

common-config des bases de données



docker-compose status

DOSSIER PROFESSIONNEL (DP)

Docker Desktop - PERSONAL

Containers [Give feedback](#) [Learn more](#)

View all your running containers and applications.

Container CPU usage: 2.52% / 1200% (12 CPUs available) Container memory usage: 2.17GB / 7.24GB

Show charts

Name	Container ID	Image	Port(s)	CPU (%)	Last start	Actions
prod	-	-	-	2.03%	2 days ago	[...]
configserver-ms	7c4411183b8d	jroulet/healthcare-ms-configserver:latest	8888:8888	0.14%	2 days ago	[...]
discovery-ms	e15e61724146	jroulet/healthcare-ms-discovery:latest	8761:8761	0.14%	2 days ago	[...]
mongodb	1b439c9ff7f24	mongo	27018:27017	0.56%	7 days ago	[...]
notes-ms	17975987e8fb	jroulet/healthcare-ms-notes:latest	9000:9000	0.17%	2 days ago	[...]
patient-ms	4137c96b3eb6	jroulet/healthcare-ms-patient:latest	8070:8070	0.14%	2 days ago	[...]
patientdb	04ef8236ff05	mysql	3308:3306	0.72%	7 days ago	[...]
webapp-ms	1a5d8f08a118	jroulet/healthcare-ms-webapp:latest	8090:8090	0.16%	2 days ago	[...]

Showing 8 items

Engine running | RAM 6.65 GB CPU 0.75% Disk: 26.86 GB used (limit 1006.85 GB) Terminal [New version available](#)

docker desktop - containers

```
PS C:\Users\LaPlateforme\Documents\Cours\Healthcare> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
846b83b90236 mongo "docker-entrypoint.s..." 6 days ago Up 4 days (healthy) 0.0.0.0:27018->27017/tcp mongodb
da91d55b9ba9 mysql "docker-entrypoint.s..." 6 days ago Up 4 days (healthy) 3306/tcp, 0.0.0.0:3306->3306/tcp patientdb

PS C:\Users\LaPlateforme\Documents\Cours\Healthcare> cd ./docker-compose
PS C:\Users\LaPlateforme\Documents\Cours\Healthcare> cd ./prod
PS C:\Users\LaPlateforme\Documents\Cours\Healthcare> docker-compose up -d --wait --force-recreate
[+] Running 7/7
  ✓ Container configserver-ms Healthy
  ✓ Container mongodb Healthy
  ✓ Container patientdb Healthy
  ✓ Container discovery-ms Healthy
  ✓ Container notes-ms Healthy
  ✓ Container webapp-ms Healthy
  ✓ Container patient-ms Healthy
PS C:\Users\LaPlateforme\Documents\Cours\Healthcare> docker-compose down
```

Commandes docker

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

J'ai développé ces composants en autonomie dans le cadre de ma formation, en m'appuyant sur la documentation officielle de Spring et des ressources pédagogiques.

J'ai également sollicité ponctuellement l'aide de mon formateur pour valider certaines décisions techniques afin de m'aider à anticiper les contraintes qui en découlent.

4. Contexte

Nom de l'entreprise, organisme ou association - La Plateforme

Chantier, atelier, service - HealthCare

Période d'exercice - Du 01/03/2025 au 03/04/2025

5. Informations complémentaires (facultatif)

Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL (DP)

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.
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.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné Jacques Roullet, déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur des réalisations jointes.

Fait à Toulon le 10/04/2025

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé

ANNEXES

(Si le RC le prévoit)

