



DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ Bare
Nom d'usage ▶
Prénom ▶ Esteban
Adresse ▶ 788 Avenue de la Libération

Titre professionnel visé

Concepteur développeur d'applications

MODALITÉ D'ACCÈS :

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

Présentation du dossier

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

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

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.
Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

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

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

Ce dossier comporte :

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

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

DOSSIER PROFESSIONNEL ^(DP)



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

Sommaire

Exemples de pratique professionnelle

Intitulé de l'activité-type n° 1: Développer une application sécurisée

p.

- CP 1 Installer et configurer son environnement de travail en fonction du projet
- CP 2 Développer des interfaces utilisateur
- CP 3 Développer des composants métier
- CP 4 Contribuer à la gestion d'un projet informatique

p.

p.

p.

p.

Intitulé de l'activité-type n° 2: Concevoir et développer une application sécurisée organisée en couches

p.

- CP 5 Analyser les besoins et maquetter une application
- CP 6 Définir l'architecture logicielle d'une application
- CP 7 Concevoir et mettre en place une base de données relationnelle
- CP 8 Développer des composants d'accès aux données SQL et NoSQL

p.

p.

p.

p.

Intitulé de l'activité-type n° 3: Préparer le déploiement d'une application sécurisée

p.

- CP 9 Préparer et exécuter les plans de tests d'une application
- CP 10 Préparer et documenter le déploiement d'une application
- CP 11 Contribuer à la mise en production dans une démarche DevOps

p.

p.

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.

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1 Développer une application sécurisée

CP 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 début de ma formation CDA, où je travaille majoritairement avec Java, j'ai eu besoin d'un éditeur de code performant avec des outils puissants. J'ai choisi **IntelliJ IDEA**, qui me permet de sélectionner la **JDK** adaptée à mon projet, d'exécuter et de compiler mes programmes Java, ainsi que d'accéder à divers outils facilitant le développement. Cet éditeur prend également en charge d'autres langages comme **HTML**, **CSS** et **JavaScript**, que j'utilise de manière occasionnelle.

Pour gérer les dépendances de mes projets, j'utilise **Maven** ou **Gradle**, bien que j'aie une préférence pour Maven. Ces outils permettent de récupérer facilement des dépendances sans problème de gestion des versions. **Maven**, étant plus ancien, est largement utilisé et compatible avec de nombreuses versions, tandis que **Gradle**, plus récent, est plus léger et optimisé pour le déploiement d'applications plus petites.

Très rapidement, j'ai commencé à travailler avec **Spring**, ce qui m'a conduit à utiliser **Spring Initializr**. Cet outil me permet, dès le début du projet, de choisir entre Maven et Gradle, de gérer les dépendances, et d'obtenir un squelette de projet Spring prêt à être développé.

Les étapes pour un projet

1. Choix des outils :

- J'opte généralement pour **IntelliJ IDEA**, qui offre un environnement productif avec une **JDK intégrée** et un **debugger performant**.
- Je sélectionne les langages nécessaires selon les besoins du projet (par exemple, si une interface utilisateur est requise).
- Je choisis entre **Maven** et **Gradle** pour gérer les dépendances, compiler et exécuter les tests.
- Je décide si mon projet sera basé sur **Spring** ou non.

2. Génération du projet :

- Une fois IntelliJ installé avec la version de Java et la JDK requise, je crée la structure de mon projet.
- Si c'est un projet Spring, j'utilise **Spring Initializr** directement depuis IntelliJ pour générer l'architecture de base.

3. Configuration et organisation :

- Je configure les versions des dépendances pour assurer la compatibilité du projet.

DOSSIER PROFESSIONNEL ^(DP)

- Je mets en place la structure des tests.
- Je crée les dossiers et composants métiers nécessaires pour débiter le développement.

2. Précisez les moyens utilisés :

J'ai utilisé un navigateur web pour télécharger les logiciels essentiels comme IntelliJ IDEA, la JDK, Maven et Gradle et IntelliJ IDEA pour configurer mes projets et gérer les dépendances.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association	La Plateforme
Chantier, atelier, service	Formation
Période d'exercice	Du 07/10/2024 au 11/07/2025

5. Informations complémentaires (facultatif)

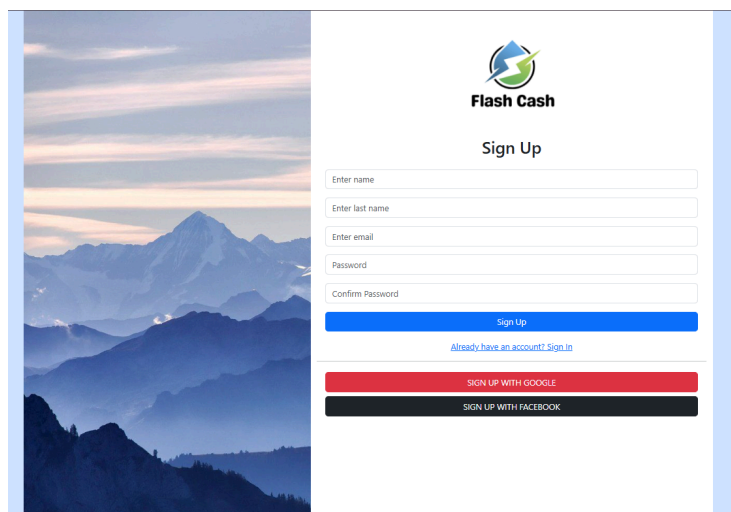
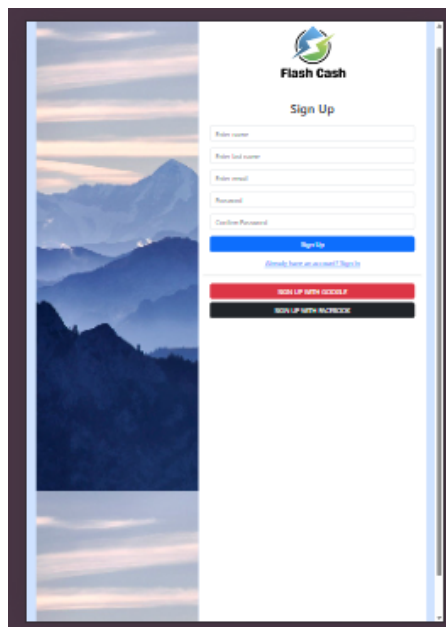
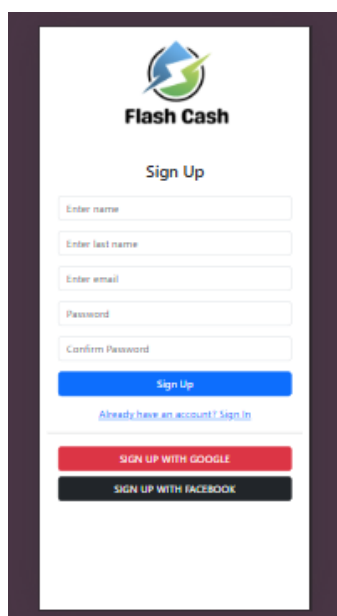
Cliquez ici pour taper du texte.

Activité-type 1 Développer une application sécurisée

CP n°2 - Développer des interfaces utilisateur

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

Pour cette compétence, j'ai deux exemples : un formulaire créé en début de formation avec HTML, CSS et Bootstrap. Ce formulaire est responsive grâce au grid de Bootstrap, ce qui fait que les champs du formulaire changent de taille ou que même le fond de la page apparaisse différemment en fonction de la résolution. Voici des images :



DOSSIER PROFESSIONNEL (DP)

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Form</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-QWTKZyjpPEjISv5MaRU9OFerPoke6YctnYmDr5pNlyT2bRjXh0JMHjV6HwH+ALEwIH" crossorigin="anonymous">
  <link rel="stylesheet" href=".../css/main.css">
</head>

<body class="bg-primary-subtle">
  <div class="container">
    <div class="row">
      <div class="col-md-5 col-sm-0 mountain"></div>
      <div class="col-md-7 col-sm-12 formContainer">
        
        <h2 class="text-center">Sign Up</h2>
        <form class="form">
          <div class="form-group my-3 mx-3">
            <input type="text" class="form-control" id="exampleInputName" aria-describedby="nameHelp"
                  placeholder="Enter name">
          </div>
          <div class="form-group my-3 mx-3">
            <input type="text" class="form-control" id="exampleInputLastName"
                  aria-describedby="lastNameHelp" placeholder="Enter last name">
          </div>
          <div class="form-group my-3 mx-3">
            <input type="email" class="form-control" id="exampleInputEmail" aria-describedby="emailHelp"
                  placeholder="Enter email">
          </div>
          <div class="form-group my-3 mx-3">
            <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
          </div>
          <div class="form-group my-3 mx-3">
            <input type="password" class="form-control" id="exampleInputPassword1"
                  placeholder="Confirm Password">
          </div>
          <div class="form-group my-3 mx-3">
            <button type="button" class="w-100 btn btn-primary">Sign Up</button>
          </div>
        </form>
        <a class="text-center" href="sign-in.html">Already have an account? Sign In</a>
        <hr>
        <div class="mx-3 my-1">
          <button type="button" class="w-100 btn btn-danger">SIGN UP WITH GOOGLE</button>
        </div>
        <div class="mx-3 my-1">
          <button type="button" class="w-100 btn btn-dark">SIGN UP WITH FACEBOOK</button>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

```
.purple {
  background-color: RGB(128, 80, 176);
}

.blue {
  background-color: #003668;
}

.mountain {
  background-image: url(.../Form/bg-montagne.jpg);
}

.imgSocial {
  width: 8.25%;
  height: auto;
}

.formContainer {
  height: 110vh;
  background-color: RGB(255, 255, 255);
  display: flex;
  flex-direction: column;
}

.mA {
  margin: auto;
}
```

On peut voir qu'avec Bootstrap et très peu de surcharge en CSS, j'ai réussi à créer un formulaire responsive pour ordinateur, tablette et mobile.

Le deuxième exemple est une page réalisée avec HTML, Thymeleaf et Bootstrap pour le style. Dans cet exemple, je vais montrer comment Thymeleaf, qui est un moteur de template de Spring, me permet de charger dynamiquement des informations du back-end dans mon template. Cela est possible parce que mon serveur web gère des routes dans lesquelles je mets à disposition des données accessibles via Thymeleaf.

```
@RequestMapping("/user/list")
public String home(Model model)
{
    model.addAttribute("users", userRepository.findAll());
    return "user/list";
}
```

```
<table class="table table-bordered">
  <thead>
    <tr>
      <th>Id</th>
      <th>Full Name</th>
      <th>User Name</th>
      <th>Role</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="user : ${users}">
      <td style="width: 10%" th:text="${user.id}"></td>
      <td th:text="${user.fullname}"></td>
      <td th:text="${user.username}"></td>
      <td style="width: 25%" th:text="${user.role}"></td>
      <td style="width: 15%" class="text-center">
        <a th:href="@{/user/update/{id}(id=${user.id})}">Edit</a> |
        <a th:href="@{/user/delete/{id}(id=${user.id})}">Delete</a>
      </td>
    </tr>
  </tbody>
</table>
```

User List

Add New				
Id	Full Name	User Name	Role	Action
3	ale	ale	USER	Edit Delete
4	esteban5	esteban5	ADMIN	Edit Delete
5	try	try	USER	Edit Delete
6	esteban	esteban	ADMIN	Edit Delete

Comme on peut le voir très facilement, j'accède aux données du serveur et même aux attributs du modèle utilisateur grâce à Thymeleaf. Avec Bootstrap, j'ai créé un tableau.

DOSSIER PROFESSIONNEL ^(DP)



2. Précisez les moyens utilisés :

Pour les deux exemple j' utilise IntelliJ pour code, pour le premier exemple j' utilise la documentation de Bootstrap pour code le formulaire et pour le deuxième exemple la documentation de Thymeleaf et Bootstrap, et le navigateur pour voire le rendu

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association		La Plateforme	
Chantier, atelier, service	Formation		
Période d'exercice	Du	07/10/2024	au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1 Développer une application sécurisée

CP n°3 - Développer des composants métier

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

Pour cette compétence, j'ai pris l'exemple d'un projet réalisé lors de ma formation, ressemblant à une banque en ligne. Dans cette application, l'utilisateur doit d'abord s'enregistrer via un formulaire, puis se connecter et être authentifié pour pouvoir accéder aux fonctionnalités de l'application. Cette authentification est réalisée avec **Spring Security**, qui interagit avec la base de données via les composants métier **Model** et **Repository**. Le **Model** représente l'utilisateur avec ses attributs, et le **Repository**, grâce à **JPA** (un ORM de Spring), permet d'accéder à la base de données **MySQL**. Une fois l'utilisateur connecté et authentifié, il pourra accéder aux pages de gestion de son compte et de son argent, grâce à la couche **Service** qui gère la logique métier, ainsi que le **Controller** qui charge les templates avec les données liées à l'utilisateur. Voici des images de mes composants:

```
@Bean
public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
        .csrf((CsrfConfigurer<HttpSecurity> csrf) -> csrf.disable())
        .authorizeHttpRequests((AuthorizationManagerRequestMatcher<> authorize -> authorize
            .requestMatchers("@"/, @"/login", @"/signin", @"/signup", @"/register", @"/css/**", @"/js/**", @"/images/**", @"/static/**").permitAll()
            .anyRequest().authenticated()
        )
        .formLogin((FormLoginConfigurer<HttpSecurity> formLogin -> formLogin
            .loginPage("/signin")
            .permitAll()
            .usernameParameter("email")
            .defaultSuccessUrl("/login", alwaysUse: true))
        .logout((LogoutConfigurer<HttpSecurity> logout -> logout
            .logoutUrl("/logout")
            .logoutSuccessUrl("/")
            .permitAll());

    return http.build();
}

@Bean
public AuthenticationManager authenticationManager(HttpSecurity http) throws Exception {
    AuthenticationManagerBuilder auth = http.getSharedObject(AuthenticationManagerBuilder.class);
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    return auth.build();
}
```

DOSSIER PROFESSIONNEL (DP)

Dans cette image, on peut voir comment, dans mon application, à part quelques routes ouvertes, le reste des routes demande une authentification. On peut voir aussi un encrypteur de mot de passe pour la sécurité des utilisateurs.

```
@Entity
@Table(name = "user", schema = "flashcash")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String email;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false, unique = true)
    private String username;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Account> accounts;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Friendship> friendships;

    public User() {}

    public User(String email, String password, String username) {
        this.email = email;
        this.password = password;
        this.username = username;
        this.accounts = new ArrayList<>();
    }
}
```

```
public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public String getUsername() { return username; }

public void setUsername(String username) { this.username = username; }

public List<Account> getAccounts() { return accounts; }

public void setAccounts(List<Account> accounts) { this.accounts = accounts; }

public List<Friendship> getFriendships() { return friendships; }

public void setFriendships(List<Friendship> friendships) { this.friendships = friendships; }
```

Ici, on peut voir mon modèle d'utilisateur qui me sert d'entité pour le repository. On peut voir aussi qu'il a des attributs, un constructeur ainsi que des getters et setters.

```
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByEmail(String email);
    Optional<User> findByUsername(String username);

    @Query("SELECT u FROM User u " +
        "WHERE u.id != :currentUserId " +
        "AND u NOT IN (" +
        "    SELECT f.friendUser FROM Friendship f WHERE f.user.id = :currentUserId " +
        "    UNION " +
        "    SELECT f.user FROM Friendship f WHERE f.friendUser.id = :currentUserId " +
        ")")
    List<User> findFriendsToAdd(@Param("currentUserId") Long currentUserId);
}
```

Dans cette image, on peut voir qu'avec JPA, je peux créer une interface que j'appelle *UserRepository* et qu'elle me donnera accès à la table *User* via des méthodes simples du CRUD, comme la création, la lecture, la mise à jour et la suppression. Mais aussi à des méthodes plus complexes, comme la recherche d'un utilisateur par son email, son surnom ou encore s'il a des relations entre les tables grâce aux relations et à la surcharge avec du SQL pur.

```
@Service 4 usages Esteban
public class AccountService {
    @Autowired
    private AccountRepository accountRepository;
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private SessionService sessionService;

    public Account saveAccount(String iban, User user) { 1 usage Esteban
        if (accountRepository.findByIban(iban).isPresent()) {
            throw new IllegalArgumentException("account already exists");
        }

        Account account = new Account(user, iban, balance: 0.0);
        accountRepository.save(account);
        return account;
    }

    public Account findById(Long id) { return accountRepository.findById(id).get(); }

    public void addMoney(double money, Account account) { 2 usages Esteban
        account.setBalance(account.getBalance() + money);
        accountRepository.save(account);
    }

    public void removeMoney(double money, Account account) { 1 usage Esteban
        account.setBalance(account.getBalance() - money);
        accountRepository.save(account);
    }
}
```

Ici, on peut voir un de mes services. Celui-ci est chargé de créer des comptes en utilisant le repository dédié à cette entité. Par exemple, il permet d'ajouter de l'argent ou d'en retirer (mise à jour).

En résumé, dans cet exemple, on peut voir que je respecte les normes de la POO, ainsi qu'une logique de travail en structurant mon application avec des *services*, *models* et *repositories*, ce qui permet une meilleure organisation et maintenabilité du code. De plus, je respecte les normes de nomenclature de Java : les classes en *CamelCase*, ainsi que les méthodes et les objets en *PascalCase*. J'ai également veillé à séparer les responsabilités pour rendre le code plus lisible et évolutif, tout en appliquant les bonnes pratiques du développement backend.

DOSSIER PROFESSIONNEL ^(DP)



2. Précisez les moyens utilisés :

Pour cette exemple j'utilise IntelliJ pour code et la documentation de Spring

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶		La Plateforme	
Chantier, atelier, service	▶	Formation	
Période d'exercice	▶	Du	07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1 Développer une application sécurisée

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

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

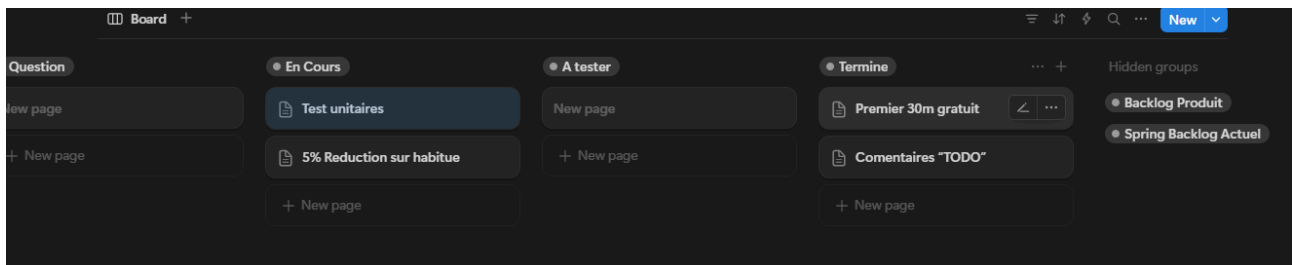
Pendant le développement de l'application **Parking System**, j'ai utilisé **Notion** pour organiser les tâches demandées durant le projet et m'assurer de compléter les fonctionnalités une par une en fonction de leur priorité

Pour cela, j'ai créé un **tableau Kanban** sur Notion afin de structurer les fonctionnalités sous forme de **cartes** contenant le titre de la fonctionnalité, une description détaillée et la priorité de la tâche

Les tâches sont réparties en plusieurs **colonnes** À faire, En cours, À tester, Terminé

J'ai également ajouté des **tags** pour identifier le type de tâche comme Test ou Nouvelle fonctionnalité

Grâce à cette organisation, j'ai pu assurer un suivi efficace de mon travail et disposer d'une **checklist claire** pour valider chaque étape du projet jusqu'à sa finalisation



DOSSIER PROFESSIONNEL (DP)

Test unitaires

Important

Important

Status

En Cours

Type

Test

+ Add a property

Comments

Add a comment...

corriger le code afin qu'il valide tous les tests unitaires

Premier 30m gratuit

Important

Tres important

Status

Termine

Type

Fonctionnalité

+ Add a property

Comments

Add a comment...

Ajouter une fonctionnalité de stationnement gratuit pour les 30 premières minutes

2. Précisez les moyens utilisés :

Pour cet exemple, j'utilise Notion

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

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

Chantier, atelier, service ▶ Formation

DOSSIER PROFESSIONNEL ^(DP)

Période d'exercice Du 07/10/2024 au 11/07/2025

5. Informations complémentaires (facultatif)

Cliquez ici pour taper du texte.

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

CP n°5 - Analyser les besoins et maquetter une application

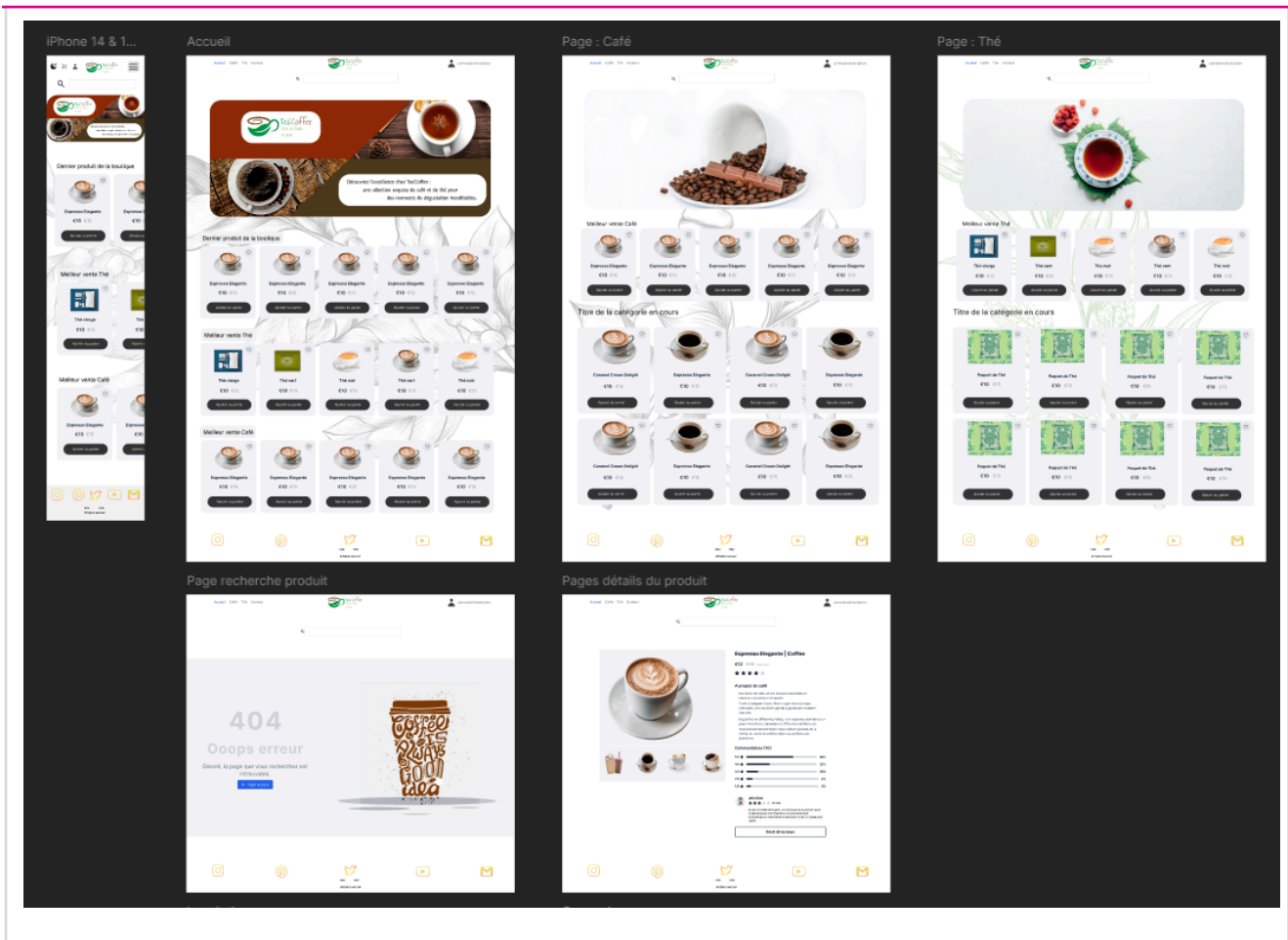
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pendant ma formation, on a dû faire un projet en groupe dans lequel l'objectif était de créer une boutique au choix. Pour cela, on a dû établir une liste ou un cahier des charges des fonctionnalités et des pages dont on avait besoin. Par exemple : une page d'accueil, une page avec tous les produits, une page pour afficher un produit, un dashboard client avec son historique d'achats, ses préférences, son profil, etc.

Pour réaliser la maquette et organiser les templates, on a décidé d'utiliser Figma, ce qui nous a permis d'avoir une vision claire de l'organisation avant le développement. En plus, Figma nous a aidés, grâce au wireframe, à montrer les liens entre les pages et le parcours logique de l'application

Voici des exemple de la maquette :

DOSSIER PROFESSIONNEL (DP)



2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL (DP)

Inscription

Veuillez vous inscrire

Nom:

Prénom:

Adresse:

Code postal:

Ville:

Mail:

Connexion

Veuillez vous connecter

Mail:

Mot de passe:

Profile

Veuillez modifier votre profil

Nom:

Prénom:

Adresse:

Code postal:

Ville:

Mail:

Modification du profile

Veuillez modifier votre profil

Nom:

Prénom:

Adresse:

Code postal:

Ville:

Mail:

Panier d'achats

Votre panier

Produit	Prix unit.	Quantité	Prix Total
Fudge Coffee 100g	€12.00	1	€12.00
Fudge Coffee 100g	€12.00	1	€12.00
Fudge Coffee 100g	€12.00	1	€12.00
Total			€36.00

Historique achats

Votre historique d'achat

Achat récent

Historique d'achats

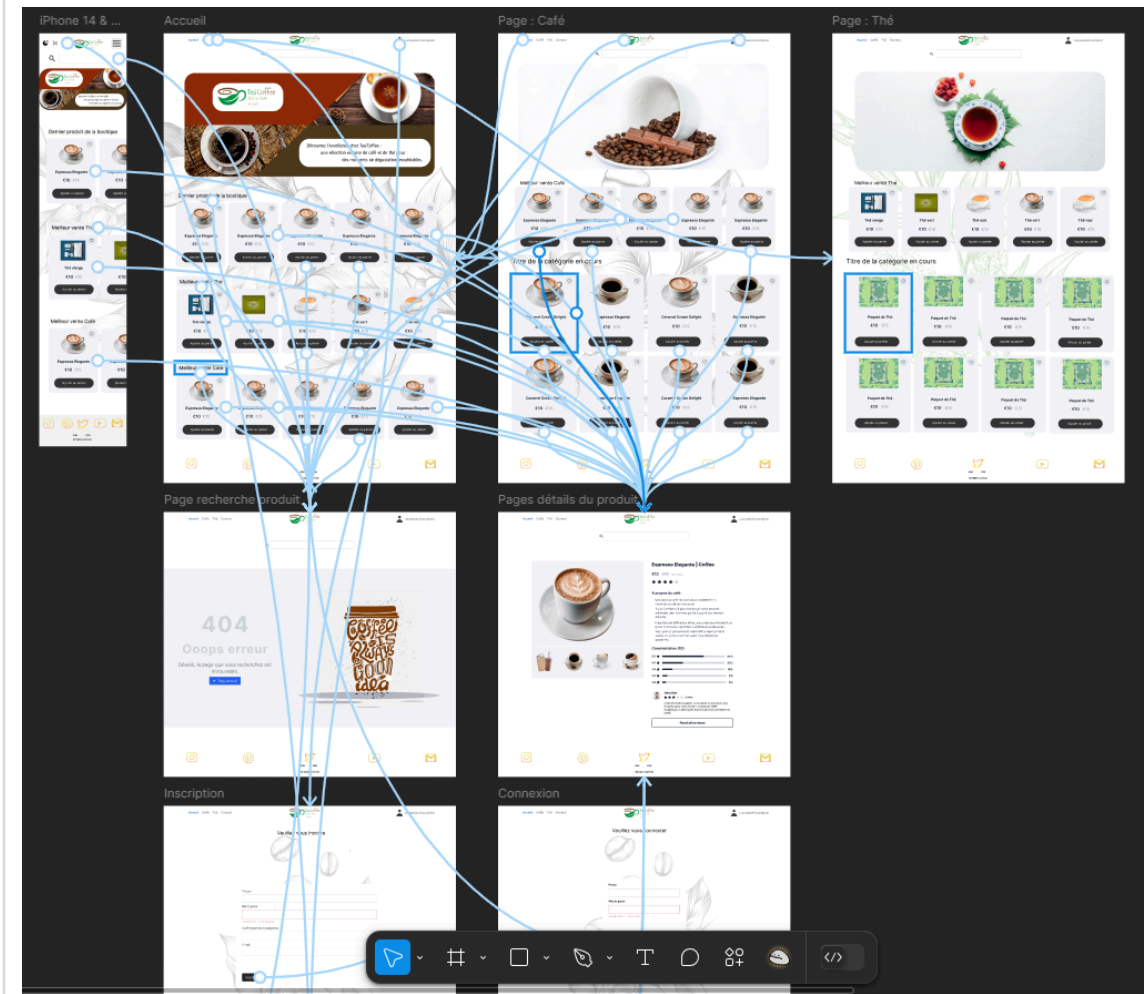
Achat 1

Achat 2

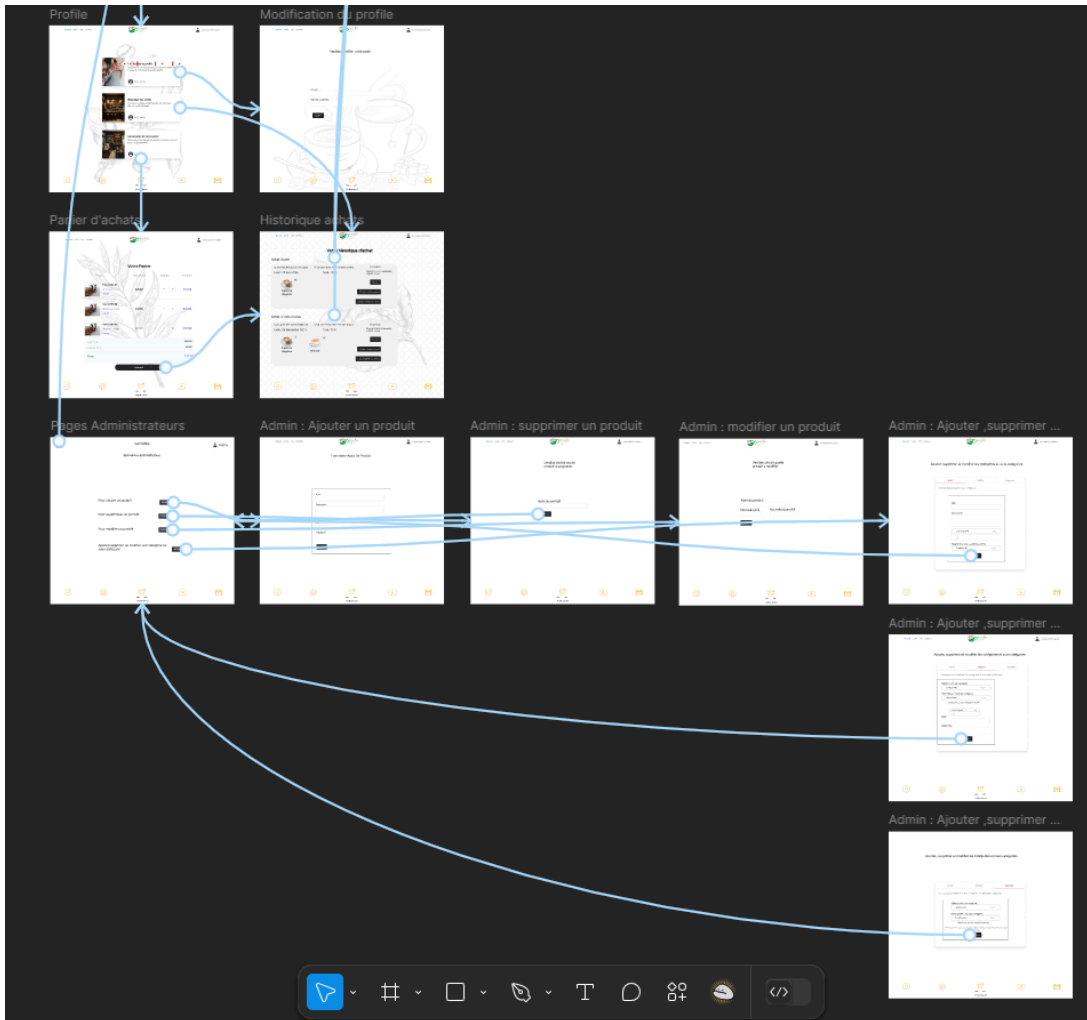
DOSSIER PROFESSIONNEL (DP)

On peut voir dans les images que l'on prend en compte l'adaptation des templates en responsive aussi.

Voici des exemples de wireframes qui montrent la logique de continuité de l'application:



DOSSIER PROFESSIONNEL (DP)



On peut voir que, grâce à Figma, j'ai la possibilité de transformer chaque objet de ma maquette en un composant, comme en HTML, et de créer des liens entre les composants et une template. Cela me permet également de présenter chaque template à un client en montrant la continuité de l'application.

En conclusion, on s'est assuré de finaliser la maquette avant le développement pour être sûrs d'avoir tout ce dont nous avons besoin pour le projet.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

Pour ce projet, j'ai travaillé en groupe durant un exercice de formation.

4. Contexte

Nom de l'entreprise, organisme ou association ▶		La Plateforme	
Chantier, atelier, service	▶	Formation	
Période d'exercice	▶	Du	07/10/2024 au 11/07/2025

5. Informations complémentaires (facultatif)

Cliquez ici pour taper du texte.

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

CP n°7 - 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 de formation, j'ai développé une API REST pour une application appelée BankApp. Cette API devait permettre la gestion de comptes bancaires, de cartes bancaires et de prêts. Le projet a été organisé en plusieurs microservices pour garantir une architecture modulaire et scalable, avec des couches distinctes : la couche de présentation (API), la couche logique métier (gestion des comptes, cartes et prêts) et la couche de persistance (gestion des données avec la base H2).

Pour la création des microservices, j'ai choisi Spring Boot, car il permet de développer des applications robustes et sécurisées. Afin de sécuriser mes microservices, j'ai intégré Spring Security et utilisé des tokens JWT pour assurer une authentification sécurisée des utilisateurs, conformément aux bonnes pratiques de sécurité des API REST.

En termes de persistance des données, et pour simplifier le déploiement, j'ai opté pour H2 en tant que base de données embarquée, avec Hibernate pour la gestion de la persistance. Cette combinaison m'a permis de stocker les données dans un fichier local tout en réduisant la complexité de l'installation et du déploiement, notamment dans un environnement de développement léger.

De plus, pour gérer de manière centralisée les configurations de chaque microservice et garantir une configuration cohérente et évolutive, j'ai utilisé Spring Cloud Config. Cela m'a permis de centraliser la configuration de mes microservices tout en assurant leur indépendance et leur capacité à être facilement mis à jour en fonction des environnements de déploiement.

Voici une image où l'on peut voir comment j'ai choisi les versions de Spring Boot et Spring Cloud, ainsi que quelques dépendances comme H2 pour la base de données, Eureka et OpenFeign pour la communication entre les microservices.

DOSSIER PROFESSIONNEL (DP)

```
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.4.2</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.test</groupId>
<artifactId>accounts</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>accounts</name>
<description>accounts</description>
<url/>
<licenses>
  <license/>
</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>21</java.version>
  <spring-cloud.version>2024.0.0</spring-cloud.version>
</properties>
<dependencyManagement>
  <dependencies>
    <dependency>
```

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</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-data-jpa</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.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

2. Précisez les moyens utilisés :

Pour concevoir et développer une application sécurisée organisée en couches, j'ai utilisé Spring Boot pour créer des microservices, Spring Security avec JWT pour sécuriser les API, et H2 pour la persistance des données. J'ai aussi intégré Spring Cloud Config pour centraliser la gestion des configurations et Eureka/OpenFeign pour la communication entre microservices.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL ^(DP)

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme	
Chantier, atelier, service	▶ Formation
Période d'exercice	▶ Du 07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 2

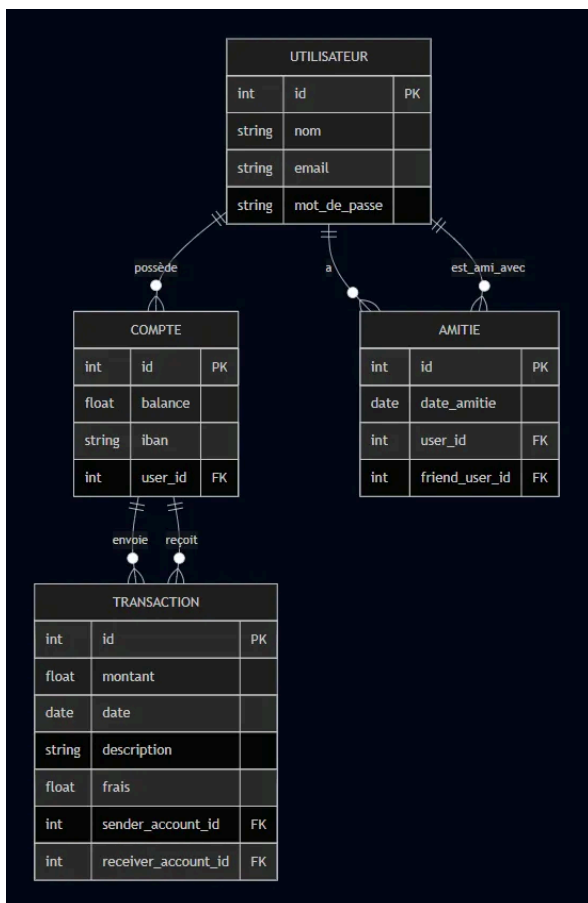
Concevoir et développer une application sécurisée organisée en couches

CP n°7 - Concevoir et mettre en place une base de données relationnelle

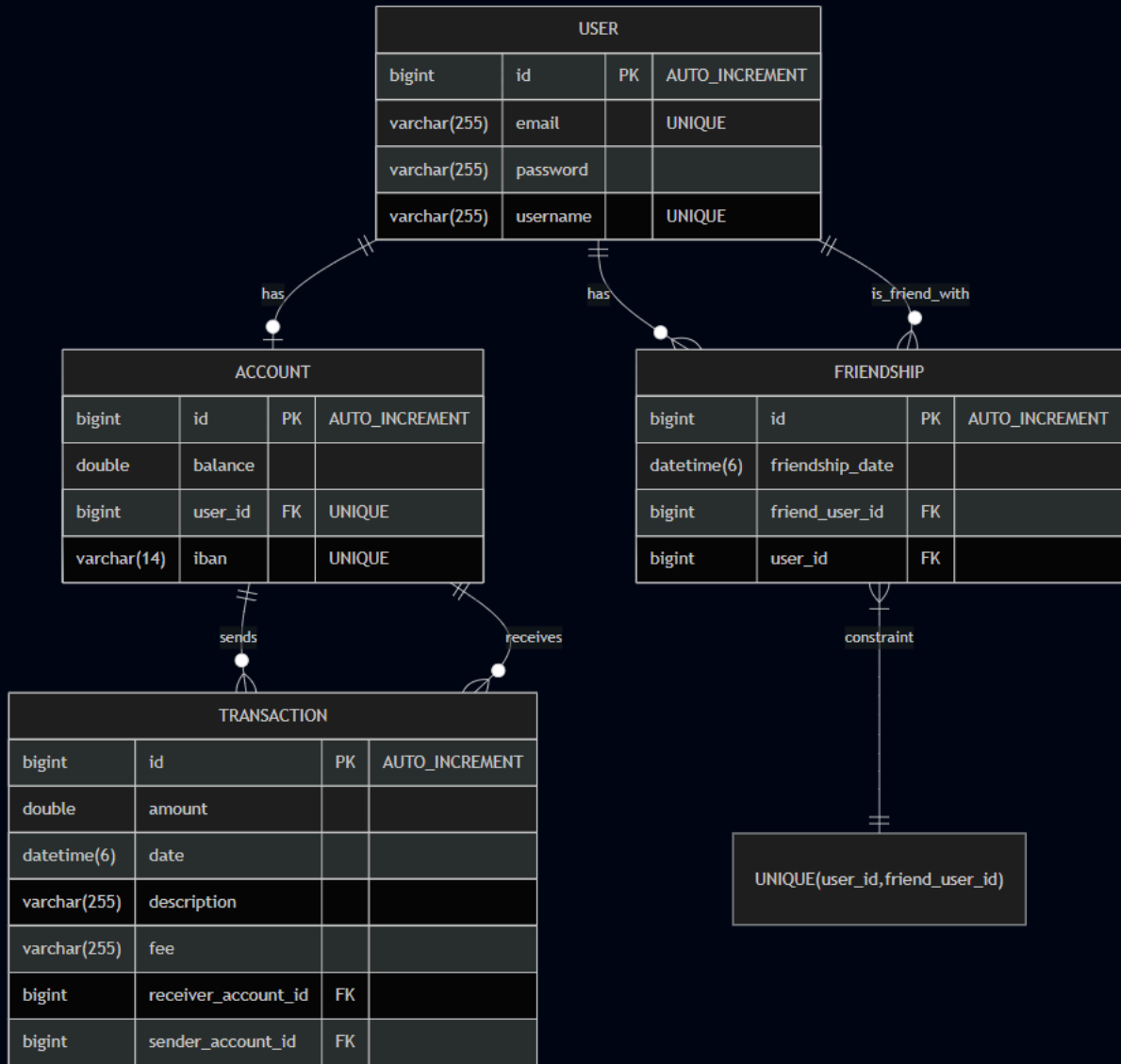
1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

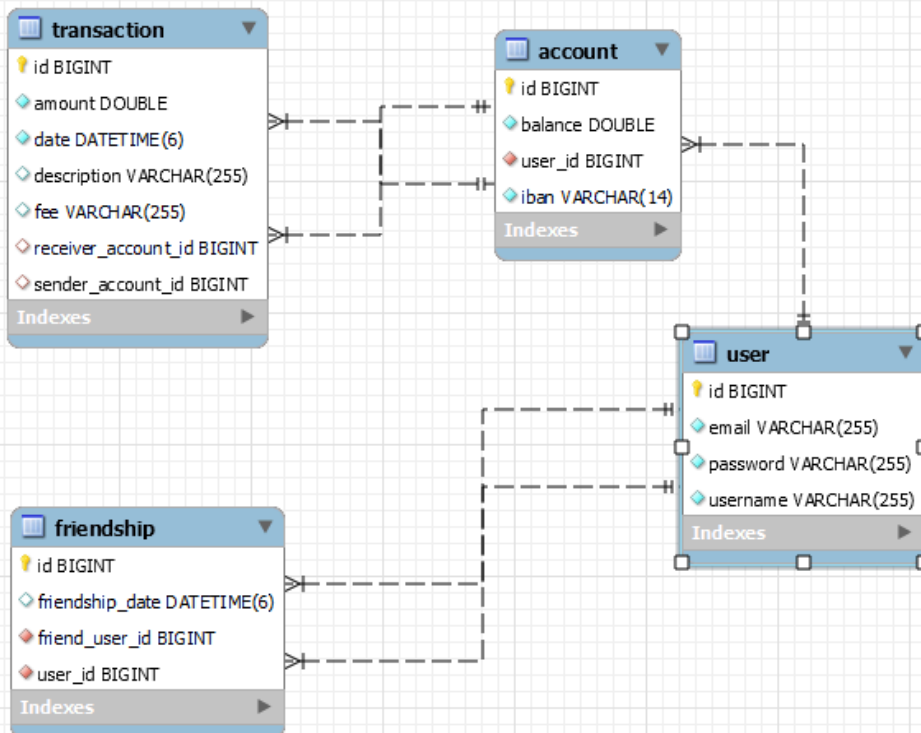
Dans mon projet FlashCash, j'ai conçu et mis en place une base de données relationnelle en suivant les étapes de modélisation. J'ai commencé par réaliser un **Modèle Conceptuel des Données (MCD)** en identifiant les principales entités et leurs relations, comme les **utilisateurs**, **comptes**, **transactions** et **cartes bancaires**. Ensuite, j'ai traduit ce modèle en **Modèle Logique des Données (MLD)** en définissant les types d'attributs et les clés primaires/étrangères pour assurer la cohérence des données. Enfin, j'ai généré le **Modèle Physique des Données (MPD)** sous **MySQL**, en créant les tables et en appliquant les contraintes d'intégrité.

Voici les modèles MCD, MLD et MPD que j'ai réalisés :



DOSSIER PROFESSIONNEL (DP)





2. Précisez les moyens utilisés :

Pour réaliser le MCD et le MLD, j'ai utilisé Mermaid pour créer les diagrammes. Pour le MPD, j'ai utilisé Workbench une fois que ma base de données était créée.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL ^(DP)

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme	
Chantier, atelier, service	▶ Formation
Période d'exercice	▶ Du 07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

CP n°8 - Développer des composants d'accès aux données SQL et NoSQL

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

Pendant ma formation, j'ai réalisé un projet appelé HealthCare, une application médicale permettant à des médecins de gérer leurs patients et d'ajouter des notes de suivi. Le projet était basé sur une architecture microservices, avec d'un côté un service qui gère la relation médecin-patient via une base de données MySQL, et de l'autre un service qui gère les notes médicales avec MongoDB, afin de pouvoir stocker une grande quantité de données non structurées. Pour les microservices, j'ai utilisé Spring Boot

- Du côté MySQL, j'ai utilisé la dépendance JPA/Hibernate pour générer les tables à partir des modèles de ma couche métier, gérer les opérations CRUD et écrire des requêtes plus complexes
- Pour MongoDB, j'ai utilisé la dépendance spring-boot-starter-data-mongodb, qui fonctionne de manière similaire à JPA et m'a permis d'effectuer des opérations CRUD sur des documents NoSQL

Voici quelques visuels montrant la mise en place et l'utilisation de mes composants d'accès aux données SQL et NoSQL dans ce projet:

Voici des images qui montrent l'utilisation de JPA/Hibernate pour créer les tables à partir d'un modèle et d'un repository, ainsi que quelques requêtes simples accessibles de base grâce à l'ORM, utilisé dans la couche service dédiée à l'utilisateur (médecin s'il a le rôle *doctor*)

```
@Entity
@Getter
@Setter
@ToString
@AllArgsConstructor
@NoArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    private String username;

    @Column(nullable = false, unique = true)
    private String email;

    @Column(nullable = false)
    private String password;

    @Enumerated(EnumType.STRING)
    @Column
    private Role role;

    @OneToMany(mappedBy = "user")
    private List<Patient> patients;
}
```

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByEmail(String email);
    Optional<User> findByUsername(String username);
}

@Service
public class UserService {
    @Autowired
    private UserRepository userRepository;

    public User registerUser(User user) {
        if (userRepository.findByUsername(user.getUsername()).isPresent()) {
            throw new UserAlreadyExistsException("User with username " + user.getUsername() + " already exists");
        }
        if (userRepository.findByEmail(user.getEmail()).isPresent()) {
            throw new UserAlreadyExistsException("User with email " + user.getEmail() + " already exists");
        }
        return userRepository.save(user);
    }

    public UserToLogDto loginUser(UserLogDto user) {
        Optional<User> user1 = userRepository.findByEmail(user.getEmail());
        if (user1.isPresent()) {
            return new UserToLogDto(user1.get().getEmail(), user1.get().getPassword(), user1.get().getRole().toString());
        }
        return null;
    }

    public Iterable<User> getAllUsers() { return userRepository.findAll(); }
}
```

DOSSIER PROFESSIONNEL (DP)

Et voici des images de l'utilisation de la dépendance MongoDB, qui fonctionne de la même façon que JPA/Hibernate

```
@Data 16 usages  ▲ Esteban
@NoArgsConstructor
@AllArgsConstructor
@Document(collection = "notes")
public class Note {

    @MongoId
    private String id;

    private String title;

    private String content;

    private String patientId;
}

@Repository 2 usages  ▲ Esteban
public interface NoteRepository extends MongoRepository<Note, String> {
    List<Note> findByPatientId(String patientId); 1 usage  ▲ Esteban
}

@Autowired
private NoteRepository noteRepository;

@Autowired
public MsPatientFeignClient msPatientFeignClient;

public List<Note> findAll() { return noteRepository.findAll(); }

public ResponseEntity<String> save(Note note) { 1 usage  ▲ Esteban
    try {
        ResponseEntity<String> response = msPatientFeignClient.patientExists(note.getPatientId());

        Note savedNote = noteRepository.save(note);
        return ResponseEntity.ok().body("Note saved");
    } catch (feign.FeignException.NotFound e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body(null);
    } catch (feign.FeignException e) {
        return ResponseEntity.status(e.status())
            .body("Error saving note");
    } catch (Exception e) {
        return ResponseEntity.badRequest().body("Error saving note");
    }
}
```

Dans les deux cas, on peut voir que j'ai spécifié les attributs des entités dans les modèles, et ensuite dans les repositories, en héritant de l'interface JpaRepository ou MongoRepository. Le moteur de Spring crée automatiquement les tables ou les met à jour grâce à l'ORM

J'ai aussi accès aux méthodes de base du CRUD via ces repositories, que j'utilise dans la couche service, comme findAll() pour récupérer toutes les données (read) ou save() pour insérer des données (create)

2. Précisez les moyens utilisés :

Pour cet exemple, j'ai utilisé Spring Boot, MongoDB, MySQL, Maven, ainsi que les ORM fournis par les dépendances JPA et Spring Data MongoDB Starter

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme	
Chantier, atelier, service	▶ Formation
Période d'exercice	▶ Du 07/10/2024 au 11/07/2025

5. Informations complémentaires (facultatif)

Cliquez ici pour taper du texte.

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

CP n°9 - Préparer et exécuter les plans de tests d'une application

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

Pour un projet de formation appelé "Parking System", mon objectif était de couvrir un pourcentage élevé du code avec des tests unitaires. Pour cela, j'ai utilisé Jacoco, un plugin Maven qui permet de mesurer la couverture du code. Avec des tests réalisés à l'aide de JUnit et Mockito, Jacoco m'a permis de calculer et visualiser les parties du code non couvertes par les tests. Cela m'a aussi permis d'identifier les lignes de code où aucun test n'avait été effectué

```
@Test  @ Esteban
void readSelectionInvalid(){
    String input = "invalid\n";
    InputStream inputStream = new ByteArrayInputStream(input.getBytes());
    System.setIn(inputStream);

    Scanner scanner = new Scanner(System.in);
    InputReaderUtil = new InputReaderUtil(scanner);

    int selection = inputReaderUtil.readSelection();

    assertEquals( expected: -1,selection);
}

@Test  @ Esteban *
void readVehicleRegistrationNumber() throws Exception {
    String regInput = "ABC123";
    InputStream in = new ByteArrayInputStream(regInput.getBytes());













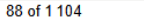

    System.setIn(in);

    Scanner scanner = new Scanner(System.in);
    inputReaderUtil = new InputReaderUtil(scanner);

    String regNumber = inputReaderUtil.readVehicleRegistrationNumber();

    assertEquals( expected: "ABC123",regNumber);
}
```

DOSSIER PROFESSIONNEL (DP)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
com.parkit.parkingsystem.service		89 %		84 %	4 35	11 151	0 17	0 5
com.parkit.parkingsystem		0 %		n/a	3 3	5 5	3 3	1 1
com.parkit.parkingsystem.config		81 %		50 %	3 9	4 26	0 6	0 1
com.parkit.parkingsystem.constants		77 %		n/a	2 3	2 6	2 3	2 3
com.parkit.parkingsystem.dao		99 %		50 %	4 13	0 79	0 9	0 2
com.parkit.parkingsystem.model		100 %		87 %	1 26	0 38	0 22	0 2
com.parkit.parkingsystem.util		100 %		75 %	1 6	0 18	0 4	0 1
Total	88 of 1 104	92 %	14 of 58	75 %	18 95	22 323	5 64	3 15

Dans la première image, on peut voir comment je teste un utilitaire qui valide les entrées du terminal de l'application. Dans la deuxième image, je montre comment Jacoco m'aide à voir à quel point mon code est testé, combien de lignes sont couvertes par les tests et lesquelles je dois encore tester.

Pour un deuxième exemple, je vais montrer un test d'intégration dans lequel j'ai été chargé de rendre plus performante une méthode de la couche service

```
@Test new *
public void highVolumeGetRewards() throws InterruptedException {
    ExecutorService executorService = Executors.newFixedThreadPool(nThreads: 200);
    Stopwatch stopWatch = new Stopwatch();
    stopWatch.start();

    Attraction attraction = rewardService.getRandomAttraction();
    List<User> users = userGateway.getUsers(numberOfUsers: 10000);
    Stopwatch stopWatch1 = new Stopwatch();
    stopWatch1.start();
    users.parallelStream().forEach( User user -> {
        user.addToVisitedLocations(new VisitedLocation(user.getUserId(), attraction, new Date()));
        rewardService.calculateRewardsAsync(user)
            .thenAccept( List<UserReward> userRewards -> userRewards.forEach(user::addUserReward));
    });
    stopWatch1.stop();
    System.out.println("time calculating " + stopWatch1.getTotalTimeSeconds() + " seconds.");
    for (User user : users) {
        while (user.getUserRewards().size() == 0) {
            TimeUnit.MILLISECONDS.sleep(timeout: 200);
        }
    }
    stopWatch.stop();
    for (User user : users) {
        assertTrue( condition: user.getUserRewards().size() > 0);
    }
    System.out.println("highVolumeGetRewards: Time Elapsed: " + stopWatch.getTotalTimeSeconds() + " seconds.");
    assertTrue( condition: 20 >= stopWatch.getTotalTimeSeconds());
}
```

Ce test est réalisé pour vérifier si une méthode, chargée de donner une récompense à un client d'un parc d'attractions, fonctionne correctement. Pour cela, il faut d'abord créer une quantité d'utilisateurs fictifs. Une fois ces utilisateurs créés, on lance un chronomètre qui s'arrêtera lorsque la méthode aura passé en revue chaque utilisateur et ajouté une récompense. Pour s'assurer que la méthode fonctionne correctement et que le test est valide, on vérifie, pour chaque

DOSSIER PROFESSIONNEL ^(DP)



utilisateur, qu'une récompense a bien été attribuée. Enfin, on vérifie que le temps écoulé est conforme aux exigences du chef de projet.

2. Précisez les moyens utilisés :

Pour cet exemple, j'ai utilisé les dépendances de test de Spring, JUnit, Mockito et Jacoco.,

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association	La Plateforme
Chantier, atelier, service	Formation
Période d'exercice	Du 07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

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

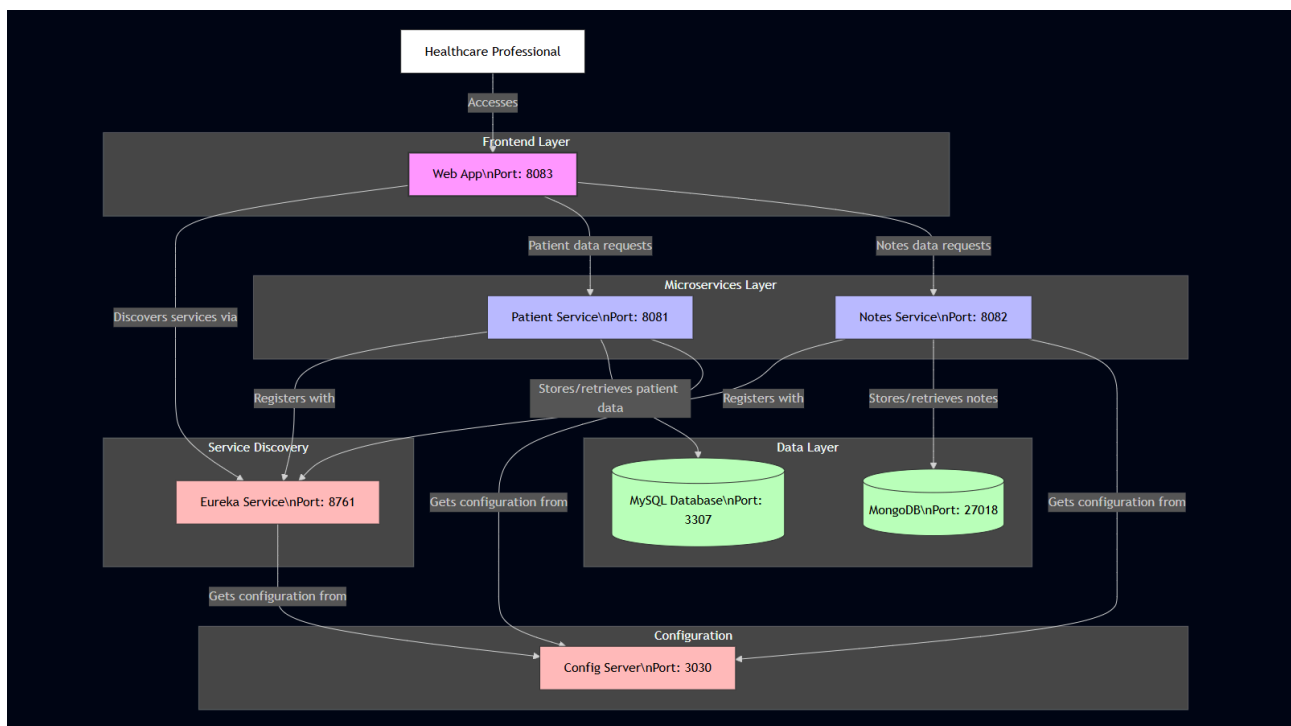
CP n°10 - Préparer et documenter le déploiement d'une application

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

Pour le projet HealthCare avec une architecture en microservices, j'ai utilisé Spring Cloud Config pour gérer différentes configurations selon les environnements (dev, test, prod). Chaque microservice récupère automatiquement sa configuration à partir d'un repository GitHub centralisé. Le microservice de configuration (Config Server) se charge de distribuer la bonne config à chaque microservice selon l'environnement. Pour le déploiement, j'ai utilisé Docker Compose. Dans le docker-compose.yml, je définis les dépendances entre les conteneurs pour que le Config Server et la base de données MySQL soient prêts avant de démarrer les autres services. Grâce à Spring Boot Actuator, je peux vérifier automatiquement que les services sont bien prêts avant de lancer les requêtes de configuration. Les images Docker de mes microservices sont générées avec Jib, un plugin Maven qui s'intègre avec les phases de build et qui exécute les tests automatiquement avant de créer l'image.

Ce système me permet de :

- garantir que chaque microservice démarre avec la bonne configuration
- tester automatiquement le projet avant la mise en production
- documenter facilement le processus de déploiement avec un fichier README.md qui explique le fonctionnement du Docker Compose et du Config Server
- switcher facilement d'un environnement à un autre (dev / test / prod)



HealthCareApp-CDA-2025: Quick Setup & Deployment

Overview

A microservices-based healthcare application for managing patient records and clinical notes with a web interface.

Prerequisites

- Docker Engine & Docker Compose
- Git
- 4GB RAM minimum

Preparation

1. Clone Repository

```
git clone https://github.com/Esteban-Bare/HealthCareApp-CDA-2025.git
cd HealthCareApp-CDA-2025
```

2. Configure Environment

Create a `.env` file in project root:

```
# Required configuration
CONFIG_SERVER_TOKEN=your-config-server-token
MYSQL_ROOT_PASSWORD=secure-password
MYSQL_DATABASE=patientdb
MYSQL_USER=healthcareapp
MYSQL_PASSWORD=secure-password
SPRING_DATASOURCE_URL=jdbc:mysql://mysql:3306/patientdb
SPRING_DATASOURCE_USERNAME=healthcareapp
SPRING_DATASOURCE_PASSWORD=secure-password
SPRING_DATA_MONGODB_URI=mongodb://mongodb:27017/notesdb
```

Deployment

1. Start Application

```
cd docker-compose
docker-compose up -d
```

2. Verify Deployment

```
docker-compose ps
```

All services should show "Up" with "healthy" status.

3. Access Application

- Web Interface: <http://localhost:8083>

Maintenance Commands

Service Management

```
# Stop services (preserve data)
docker-compose stop

# Remove containers (data lost)
docker-compose down

# View logs
docker-compose logs [service-name]

# Update services
docker-compose pull
docker-compose up -d
```

Troubleshooting

- Check service logs: `docker-compose logs [failing-service]`
- Verify environment variables in `.env`
- Ensure proper startup sequence (databases → config → discovery → services → web)

En conclusion, avec le schéma de mon architecture et la documentation du déploiement disponible dans le repository GitHub du projet, le déploiement de cette application se fait facilement et de manière claire.

2. Précisez les moyens utilisés :

Pour cet exemple, je me suis servi de Docker, Docker Compose, Jib, de la dépendance Spring Cloud pour le serveur de configuration, de Mermaid pour créer le schéma de l'architecture, et de GitHub pour héberger le projet et la documentation.

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL ^(DP)



J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme	
Chantier, atelier, service	▶ Formation
Période d'exercice	▶ Du 07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1 Développer une application sécurisée

CP 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 :

DOSSIER PROFESSIONNEL ^(DP)



2. Précisez les moyens utilisés :

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶		La Plateforme	
Chantier, atelier, service	▶	Formation	
Période d'exercice	▶	Du	07/10/2024 au 11/07/2025

DOSSIER PROFESSIONNEL ^(DP)



5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

Activité-type 1 Développer une application sécurisée

CP 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 :

2. Précisez les moyens utilisés :

DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ La Plateforme	
Chantier, atelier, service	▶ Formation
Période d'exercice	▶ Du 07/10/2024 au 11/07/2025

5. Informations complémentaires *(facultatif)*

Cliquez ici pour taper du texte.

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.

Déclaration sur l'honneur

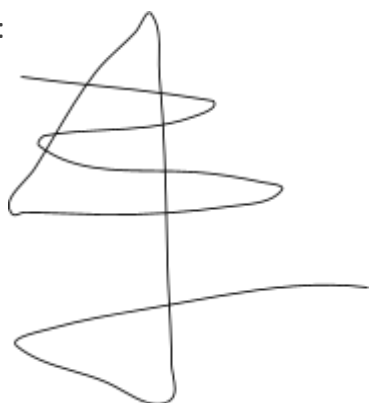
Je soussigné Esteban Bare *Cliquez ici pour taper du texte.* ,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je
suis l'auteur des réalisations jointes.

Fait à *Toulon.*

le *24/05/2025*

pour faire valoir ce que de droit.

Signature :

A handwritten signature in black ink, consisting of several loops and a long horizontal stroke at the bottom.

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES

(Si le RC le prévoit)

