#### ECOLE DES HAUTES ETUDES D’INGENIERIE – OUJDA

**CAISSE DE DÉPÔT ET DE GESTION PRÉVOYANCE - RABAT**

**Mémoire de Projet de Fin d’Etude** Présenté en vue de l’obtention du **DIPLÔME D’INGENIEUR D’ETAT**

FILIERE : Génie Informatique

Conception et

développement d'une application web - Le suivi des versements RCAR/RECORE

**Réalisé par :**

M BAKHALED Ibrahim

**Encadré par :**

M BOUHJRA Mouad

M SERGHINI Abdelhafid

**Stage effectué à :**

CDG Prévoyance CNRA/RCAR

**Soutenance le 21/07/2022 devant le Jury :**

M/MME ………………………………………….

M/MME ………………………………………….

M/MME ………………………………………….

#### Année universitaire : 2021 - 2022

**Dédicace**

Je dédié mon mémoire :

À mon père Belaid & ma mère Fouzia Vous êtes pour moi une source de vie car sans vos sacrifices, votre tendresse et votre *affection je ne pourrais arriver jusqu’au bout. Je me réjouis de cet* amour filial. Que Dieu vous garde afin que votre regard puisse suivre ma destinée.

.

À mon *frère* et *sœurs* qui ont été toujours présent pour moi.

*A mes amis avec qui j’ai partagé des moments des* plus agréables. A tous ceux qui sont chères, proches de mon cœur, et a tous ceux

*qui m’aiment et qui aurait voulu partager ma joie …*

**Remerciements**

Mes remerciements les plus sincères à toutes les personnes qui ont contribué de près ou de loin à l’élaboration de cette mémoire ainsi qu’à la réussite de cette formidable année universitaire.

Je remercie piètrement Allah le tout puissant de m’avoir donné le courage et là

volonté de mener à terme ce présent travail.

J’adresse mes vifs remerciements :

A mes encadrants Mr.BOUHJRA Mouad et Mr.SERGHINI Abdelhafid pour leurs encadrements, leur soutien sans faille et leur disponibilité. Leurs conseils, leurs suggestions de lecture, leurs commentaires, leurs corrections et leurs qualités scientifiques ont été précieux pour la réalisation de ce travail.

Je présente par la même occasion ma profonde gratitude à M. AHBACHOU Mustapha de m’avoir accordé ce stage.

Je tiens également à remercier et exprimer mon profond respect aux membres de jury d’avoir accepté de juger ce travail.

Sommaire

[INTRODUCTION GENERALE 1](#_bookmark0)

[Chapitre I : Cadre Générale de Le Suivi des Versements](#_bookmark1)

1. [Introduction 3](#_TOC_250041)
2. [Définition de la CDG Prévoyance 3](#_TOC_250040)
3. [L’amission, vision et le savoir-faire de l'entreprise 3](#_TOC_250039)
   1. [Notre mission 3](#_TOC_250038)
   2. [Notre vision 3](#_bookmark2)
   3. [Notre savoir-faire 3](#_bookmark3)
4. [Les 3 lignes métiers 4](#_bookmark4)
5. [Etude de l’existant 5](#_bookmark5)
6. [Critique de l’existant 5](#_TOC_250037)
7. [Solution à prévoir. 6](#_TOC_250036)
8. [Objectifs 7](#_TOC_250035)
9. [Prototype des interfaces 8](#_TOC_250034)
10. [Conclusion. 10](#_TOC_250033)

[Chapitre II : Analyse et spécification](#_TOC_250032)

1. [Introduction](#_bookmark6) 11
2. [Analyse des besoins 11](#_TOC_250031)
   1. [Les besoins fonctionnels](#_TOC_250030) 11
   2. [Les besoins non fonctionnels 12](#_TOC_250029)
3. [Diagramme des cas d’utilisation général 13](#_bookmark7)

[III.1 Définition d’uﾐ cas d’utilisation 13](#_bookmark8)

[III.2. Identification de l'acteur 14](#_bookmark9)

1. [Conclusion 14](#_bookmark10)

[Chapitre III : Etude Conceptuelle](#_bookmark11)

1. [Introduction](#_bookmark12) 16
2. [Description des diagrammes 15](#_TOC_250028)
   1. [Définition de diagramme de séquence 15](#_TOC_250027)
   2. [Définition du diagramme d’activité 15](#_TOC_250026)
3. [Capture des besoins fonctionnels 15](#_TOC_250025)
   1. [Description des cas d'utilisation 16](#_bookmark13)
      1. [Cas d’utilisation « gestion des relevés bancaires » 16](#_bookmark14)
      2. [Cas d’utilisation « gestion des workflows » 19](#_bookmark15)
      3. [Cas d’utilisation « Qualification automatique » 22](#_bookmark16)
      4. [Cas d’utilisation « Consultation de relevé bancaire » 25](#_bookmark17)
      5. [Cas d’utilisation « Qualification manuelle des produites » 26](#_bookmark18)
      6. [Cas d’utilisation « Qualification manuelle des acteurs » 29](#_bookmark19)
      7. [Cas d’utilisation « Enregistrement de relevé bancaire »](#_TOC_250024) 31
4. [Description de Les diagrammes des états transition 33](#_bookmark22)
   1. [Le classe « Relevé Bancaire » 33](#_bookmark23)
   2. [Le classe « Utilisateur » 34](#_bookmark24)
5. [Description des diagrammes de contexte 35](#_bookmark25)
   1. [Définition de diagramme de contexte 35](#_bookmark27)
   2. [Légende des messages 36](#_bookmark26)
6. [Description des diagrammes de classes 37](#_bookmark28)
   1. [Définition de diagramme de classes 37](#_bookmark30)
   2. [Le diagramme de classes 37](#_bookmark31)
7. [Description détaillée des attributs 39](#_bookmark32)
   1. [Les attributs des classes d'objets 39](#_bookmark33)
   2. [Les attributs des classes d'associations 40](#_bookmark34)
8. [Les méthodes des classes d'objets et des classes d'associations 41](#_bookmark36)
9. [Conclusion 42](#_TOC_250023)

[Chapitre IV : Réalisation](#_TOC_250022)

1. [Introduction 43](#_TOC_250021)
2. [Environnement du travail 43](#_bookmark38)
   1. [Environnement matériel 43](#_bookmark39)
   2. [Environnement logiciel 43](#_bookmark40)

[.II.2.1. Système d’exploitation 44](#_bookmark41)

* + 1. [Outil de modélisation UML 44](#_bookmark42)
    2. [Environnement de développement intégré 44](#_bookmark43)
    3. [Plateforme de gestion d’API : « Postman » 44](#_bookmark44)
    4. [Conteneurisation : « Docker » 45](#_bookmark45)
    5. [Orchestration des conteneurs : « Docker Compose » 45](#_bookmark46)
    6. [Hébergement et de gestion de développement de logiciels : « GitHub ». 45](#_bookmark47)
    7. [L'automatisation des Workflows avec CI/CD : « GitHub Actions » 46](#_TOC_250020)
    8. [La documentation des APIs : « Swagger v.3 » 46](#_TOC_250019)
    9. [Gestion de la qualité du code : « SonarQube »](#_bookmark50) [46](#_bookmark50)
    10. [Gestion de la qualité du code sur cloud : « SonarCloud » 47](#_TOC_250018)
    11. [Outil d’administration de la base de données RDBMS et DBMS 48](#_TOC_250017)
    12. [Langage de programmation](#_bookmark51) [48](#_bookmark51)
    13. [Environnement d'exécution 49](#_TOC_250016)
    14. [Gestionnaire de paquets. 50](#_TOC_250015)

1. L’architecture de l’application 51
   1. [Architecture 3-tiers 51](#_TOC_250014)
   2. [Micro services 51](#_TOC_250013)
   3. [Architecture Hexagonal 52](#_TOC_250012)
2. [La couche de la sécurité 5](#_bookmark59)2
3. [Les technologies utilisée -Plus détaillées 53](#_bookmark60)
4. [Les microsevices 54](#_bookmark61)
   1. [Angular+NGINX microservice 54](#_bookmark62)
   2. [Spring Cloud API Gateway micro service + LoadBalancer Ribbon 54](#_bookmark63)
   3. [Netflix Eureka Service Registration and Discovery 56](#_bookmark64)
   4. [Relevé Bancaires micro service (Hexagonal Architecture) 57](#_bookmark65)
      1. [Introduction à l'Architecture Hexagonale. 58](#_bookmark66)
      2. [L'architecture Hexagonale sur notre projet 62](#_bookmark67)
   5. [Activiti Workflow microservice. 64](#_bookmark68)
      1. [Introduction au Activiti Workflow 64](#_bookmark69)
      2. [Diagramme d'architecture de service d'Activiti 65](#_bookmark70)
      3. [Mise en place d'Activiti sur le projet 66](#_bookmark71)
   6. Sécurisation de l'application 67

.VI.6.1 Okta - Authentification et autorisation 67

VI.6.2 Okta - Sécuriser la communication avec HTTPS 70

* 1. Système de traçage 71

1. [Les interfaces graphiques 72](#_TOC_250011)
   1. [Interface Authentification 73](#_TOC_250010)
   2. [Interface Chargement de Relevé Bancaire 74](#_TOC_250009)
   3. [. Interface des Tâches de workflow Activiti 75](#_TOC_250008)
   4. [Interface des informations des relevés bancaires 76](#_TOC_250007)
   5. [Interface des informations les lignes de relevé bancaire 77](#_TOC_250006)
   6. [Interface de la qualification manuelle de l'acteur 78](#_TOC_250005)
   7. [Interface de la qualification manuelle de produite 79](#_TOC_250004)
   8. [Interface de système de traçage du serveur Zipkin 80](#_TOC_250003)
   9. [Interface de message broker RabbitMQ 81](#_TOC_250002)
   10. [Interface de qualité de code 82](#_TOC_250001)
   11. [Interface CI/CD avec GitHub Actions 83](#_TOC_250000)

CONCLUSION GENERALE 84

Liste de figure

Figure I.1. : Prototype - Interface de chargement de relevé bancaire **8**

Figure I.2: Prototype - Interface des tâches du Activiti workflow **8**

Figure I.3: Prototype - Interface de détails du relevé bancaire téléchargée **9**

Figure I.4: Prototype - Interface de détails des lignes de relevés téléchargée **9**

Figure II.1 : Diagramme de cas d’utilisation général de système **13**

Figure III.1 : Diagramme de cas d’utilisation « gestion des relevés bancaires » **16**

Figure III.2 : Diagramme d’activité « Gestion des relevés bancaires » **18**

Figure III.3 : Diagramme de séquence du scénario « Le chargement de relevé bancaire » **19**

Figure III.4 : Diagramme des cas d’utilisation « Gestion des Workflows » **19**

Figure III.5 : Diagramme d’activité « Création d'un workflow (processus métier) » **21**

Figure III.6 : Diagramme de séquence du scénario « Crée un processus métier » **22**

Figure III.7 : Diagramme d’activité « Qualification automatique » **22**

Figure III.8 : Diagramme d’activité « Qualification automatique » **24**

FigureIII.9 : Diagramme de séquence du scénario « Qualification automatique » **24**

Figure III.10 : Diagramme des cas d’utilisation « Consultation de relevé bancaire » **25**

Figure III.11 : Diagramme de séquence du scénario « Consultation de relevé bancaire » **26**

Figure III.12 : Diagramme des cas d’utilisation « Qualification manuelle des produites » **26**

Figure III.13 : Diagramme de séquence du scénario « ajouter un produit » **28**

Figure III.14 : Diagramme de cas d’utilisation « Qualification manuelle des acteurs » **29**

Figure III.15 : Diagramme de séquence du scénario « ajouter un acteur » **31**

Figure III.16 : Diagramme de cas d’utilisation « Enregistrement de relevé bancaire » **31**

Figure III.17 : Diagramme de séquence du scénario « Enregistrement de relevé bancaire » **32**

Figure III.18 : Diagramme état transition « relevé bancaire » **33**

Figure III.19 : Diagramme état transition « utilisateur » **34**

Figure III.20 : Diagramme de contexte **35**

Figure III.21 : Diagramme de classe **38**

Figure IV.1 : Les technologies utilisé **50**

Figure IV.2: Architecture 3 - tiers **51**

Figure IV.3 : Présentation de l'application **53**

Figure IV.4 : Mécanisme de Spring Cloud API Gateway **55**

Figure IV.5 : Mécanisme de Eureka Service Discovery **56**

Figure IV.6 : Interface de Eureka Service Discovery **57**

Figure IV.7 : Les zones de l'architecture **58**

Figure IV.8: Driving Side vs Driven Side **60**

Figure IV.9 : Les composants internes de l'architecture **61**

Figure IV.10 : L'aperçu de l'hexagone sur le projet **63**

Figure IV.11 : Diagramme d'architecture de service d'Activiti **65**

Figure IV.12 : Mise en place d'Activiti sur le projet **66**

Figure IV.13 : Okta Implémentation avec Angular **68**

Figure IV.14 : Page de connexion Okta avec application **69**

Figure IV.15 : L'implémentation de Self-Signed Certificat **71**

Figure IV.16 : Système de traçage **7*2***

Figure IV.17 : Interface Authentification **73**

Figure IV.18 : Interface Chargement de Relevé Bancaire **74**

Figure IV.19 : Interface des Tâches de workflow Activiti **75**

Figure IV.20 : Interface des informations des relevés bancaires **76**

Figure IV.21 : Interface des informations les lignes de relevé bancaire **77**

Figure IV.22 : Interface de la qualification manuelle de l'acteur **78**

Figure IV.23 : Interface de la qualification manuelle de produite **79**

Figure IV.24 : Interface de système de traçage du serveur Zipkin **80**

Figure IV.25 : Interface de message broker RabbitMQ **81**

Figure IV.26 : Interface de qualité de code **82**

Figure IV.27 : Interface CI/CD avec GitHub Actions **83**

Liste des tableaux

[TABLEAU III.1: Légende des messages **36**](#_bookmark29)

[TABLEAU III.2 : Description détaillée des attributs des classes d'objets](#_bookmark35)

(Relevé Bancaire micro service) **29**

TABLEAU III.3 : Description détaillée des attributs des classes d'objets

(Activi Workflow micro service) **40**

TABLEAU III.4 : Description détaillée des attributs des classes

D’associations (Relevé Bancaire micro service) **40**

[TABLEAU III.5 : Description des méthodes des classes d'objets et d'association (Relevé](#_bookmark55) Bancaire micro service) **41**

TABLEAU III.6: Description des méthodes des classes d'objets et d'association

(Activiti Workflow micro service) **42**

TABLEAU IV.1. Liste des micro services avec url et ports **53**

Page 9

**Introduction générale**

Personne ne peut plus douter que l’informatique est une révolution fondamentale et innovante qui a touché considérablement la vie humaine durant le dernier siècle. En effet, loin d’être un phénomène effervescent, ou une tendance passagère, l’informatique vient d’être exploitée dans tous les aspects de la vie. Aucun domaine n’est resté à l’abri de cette politique qui facilite les tâches aussi bien pour l’entreprise que pour le personnel.

En effet, les systèmes d’informations ont répondu à un besoin vif pour n’importe quel type d’organisation ; c’est la gestion d’information qui est parmi les enjeux les plus primordiaux pour les entreprises et touche pratiquement toutes les activités telles que la comptabilité, la production et la maintenance.

Dans le milieu financier marocain a été profondément réformé, depuis le début des années 90, autour de nombreux axes notamment le décloisonnement des marchés de capitaux par la transformation des relations entretenues entre les différents éléments constitutifs du système financier, la libéralisation des opérations financières et la réforme du cadre réglementaire des banques et du marché financier

L'Internet est un système de communication qui permet la communication et l’échange facile des informations. Ce dernier permet donc, de généraliser l'utilisation des outils informatiques (logiciel) plus performants avec des clients légers (navigateur web complet et sans demander l’installation de logiciel sur des machines individuelles). Ceci permet l’accès aux ressources sans contraintes particulières. Cette technologie permet le développement des applications pouvant tourner sous différents navigateurs, tout en assurant la sécurité que procure une application métier java.

A cet effet, notre travail consiste à développer une application de la gestion rcar et cnar des clientes assistées par ordinateur au sein du pôle prévoyance du groupe Caisse de dépôt et de Gestion (CDG). Afin de faciliter le travail aux utilisateurs de certaines taches manuelles et répétitives, et en fin d’arriver a élaboré des travaux pratiques aux étudiants pour qu’ils puissent être aptes et capable d’assurer au sein de l’environnement professionnelle.

Pour réaliser notre projet on va suivre la démarche suivante :

* Chapitre 1 : Le premier chapitre est une prise de connaissance et une analyse de l'existant pour mieux définir les besoins et les fonctions de notre application.
* Chapitre 2 : Analyse et spécification des besoins, cette partie consiste à collecter, analyser et définir les besoins de haut niveau (les besoins fonctionnels et les besoin non fonctionnels).
  + Chapitre 3 : Conception, dans ce chapitre sera consacrée à la conception de l'application : il s'agit d'une phase de modélisation théorique de l'application.
* Chapitre 4 : Réalisation, ce chapitre contient une description détaillée des outils utilisés pour développer l’application web, l’architecture du système et le matériel de déploiement de l’application.

# Chapitre I : Cadre Générale du

**Suivi des Versements**

### Introduction

Ce premier chapitre est consacré pour introduire des notions de base sur la gestion des retraites et son processus de gestion ainsi que les notions et intérêts relatifs à la gestion des retraites assistée par ordinateur.

### Définition de la CDG Prévoyance

La Branche Prévoyance de la Caisse de Dépôt et de Gestion, portée par CDG Prévoyance, incarne sa vocation de tiers de confiance et assure la gestion administrative, technique et financière de deux organismes : la CNRA et le RCAR et par conséquent, 159 milliards de dirhams de réserves correspondant à plus de 24 régimes de retraite et fonds de prévoyance

, caractérisés par une grande diversité en termes de population, d’architecture et de nature de prestations. Afin de remplir cette mission, CDG Prévoyance, opérateur global et intégré

, s’appuie sur une gouvernance, une plateforme organisationnelle moderne, un système d’information agile et extensible et un capital humain qualifié pour servir plus de 1 Million de citoyens.

### La mission, vision et le savoir-faire de l'entreprise

#### Notre mission

Assurer la gestion des régimes de retraite et des fonds de solidarité confies sous mandat tout en contribuant au développement économique, social et humain du pays.

#### Notre vision

Consolider le caractère de ≪Tiers de Confiance≫ en tant que gestionnaire de référence des régimes de retraite et des fonds de solidarité.

#### Notre savoir-faire

Une plateforme de gestion multi-régimes et multi-fonds capable d’offrir une qualité de service de haut niveau basée sur une structure organisationnelle moderne, un système D’ information agile aux meilleures normes, un capital humain de qualité et un dispositif de bonne gouvernance.

### Les 3 lignes métiers

On rencontre trois types (ou concepts) des lignes métiers :

###### Gestion des retraites

CDG Prévoyance œuvre pour la promotion et la gestion de produits et régimes de retraite obligatoires, facultatifs et pour le compte de l’Etat au profit des différentes couches socioprofessionnelles :

* + *RCAR :* Constitué d’un Régime Général et d’un Régime Complémentaire, le RCAR a pour objet de garantir des droits personnels au profit de l’affilié ou ses ayants-cause, au titre des risques de vieillesse, d’invalidité et de décès.
  + *RECORE :* Lancé en 1988, le Régime Complémentaire de Retraite est un produit de retraite complémentaire, à adhésion facultative, conçu pour couvrir certaines catégories socio-professionnelles non éligibles aux régimes de base, ainsi que celles ayant un besoin additionnel en matière de retraite.
  + *FRAM :* Lancé en 1991, le Fonds de Retraite des Avocats du Maroc est un produit dédié aux avocats du Maroc, à l’exception de ceux de Casablanca. Il assure à l’avocat, titulaire, stagiaire ou à ses ayants droit, des prestations de retraite, d’invalidité ou de décès. CRAC : Caisse de Retraite des Avocats de Casablanca
  + *RRMCR :* Régime de Retraite des Membres de la Chambre des Représentants

###### Fonds de solidarité

CDG Prévoyance intervient dans la gestion des rentes Accidents du Travail (AT) et Maladies Professionnelles (MP), des rentes Accidents de la Circulation (AC), des Fonds du Travail (FDT) et des arrérages des rentes AT au profit des ex-employés des Charbonnages du Maroc (CDM). CDG Prévoyance prend en charge également, des programmes d’aides directes visant l’inclusion sociale et la lutte contre la précarité, tels que le Fonds d’Entraide Familiale et Daam Al Aramil.

###### Gestion des investissements

CDG Prévoyance conçoit et met en œuvre des stratégies d’investissement adaptées aux besoins de ses fonds gérés, au contexte analyse et aux perspectives d’évolution des marchés financiers. Elle s’appuie à ce titre sur une structure de gouvernance financière aux meilleures normes et sur une proximité établie entre les activités ≪ Investissement

≫ et ≪ Actuariat ≫ pour une prise en compte efficiente des engagements propres à chaque régime ou produit gérés à travers le RCAR et la CNRA. L’épargne gérée par CDG Prévoyance représente, à fin 2020, 159 Milliards de Dirhams soit plus du tiers des fonds de la réserve des retraites à l’échelle nationale.

### Etude de l’existant

On effectue un stage dans l'entreprise CDG Prévoyance. Au cours de l’étude du processus de la qualification d’une grande quantité des relevés bancaires, on a pu relever quelques problèmes concernant les services de la gestion des relevés bancaires dans deux coté manuelle et automatique, il Ya plus applications qui offrent des services bien avancés ne sont pas gratuit et ils sont trop chère spécifiquement la qualification automatique.

### Critique de l’existant

* La perte du temps au niveau de la qualification à partir des méthodes traditionnelles.
* Redondance.
* Manque d’une archive qui contient les relevés bancaires déjà qualifier aux listes des relevés bancaires.
* Pas de modèle de présentation des relevés bancaires enregistrés à la base de données.
* Manque de service de contrôle des relevés bancaires par l’administrateur.
* Manque d’un service qui permet de contrôler le chargement des relevés bancaires par les employées.
* Manque de service de qualification automatique des relevés bancaires contrôlé par l’administrateur.
* Le risque de perte des données de relevés bancaires.
* Problème de sécurité des données des clients contre toute menace.

### Solution à prévoir

Pour résoudre ces différents problèmes, nous proposons les solutions suivantes :

* Développer une application pour offrir des services gratuits à la société tournant sur un réseau hétérogène.
* Qualifiez un grand nombre de relevés bancaires à la fois et en moins de temps possible.
* Les droits d’accès sont bien spécifiques et bien sécurisé.
* Gestion des relevés bancaires en utilisant aussi la fonctionnalité de chargement les relevés pour ne pas perde les données.
* Gestion de chargement et qualification automatique des relevés bancaires selon date et heure, et ce service reste contrôlé par l’administrateur.
* Gestion de la base de données avec sécurité surtout aux niveaux des données des clients.
* Contrôle les relevés bancaires qui ont été déjà qualifier ou rejeter.
* Minimum de temps aux niveaux de l’étape de qualification automatique vers la base de données.
* Etat de sortie de l’application c’est la déconnexion en toute sécurité.

### Objectifs

La mise en place d’un outil permet :

* Création des multi-micro services pour séparer et faciliter notre travaille.
* Création l'architecture hexagonale (architecture à base de ports et d'adaptateurs) pour notre backend micro service.
* Création des bases de données relationnelle et non-relationnelle.
* L'intégration de versionnage pour le suivi de notre travaille avec GitHub.
* Création le micro service de la localisation d'instances de services (micro services).
* Création un micro service comme le point d'entrée unique de notre application pour dispatcher, modifier et appliquer des filtres et des règles aux requêtes.
* L'intégration des tests, déploiement est l'automatisation de tous les workflows logiciels maintenant avec CI/CD depuis GitHub.
* Intégrer la qualité du code pour la maintenabilité, la lisibilité et moins de vulnérabilité.
* Création d'un service qui assurera la distribution des messages avec l’émetteur et le destinataire.

Sécuriser notre application avec l'autorisation, l'authentification et la communication entre tous les micro services ainsi qu’avec les bases de données.

* Création des interfaces qui repend aux critères ergonomiques, qui est simple à utiliser.
* Chargement et qualifier une grande quantité des fichiers relevés bancaires.
* Le Suivi des relevés bancaires qualifié et non qualifié par l'utilisateur.

Submit

sélectionnez votre fichier à télécharger

Nom Document

Date creation

Type

Taille

Status

CDG Logo

Upload

Inbox

User

tableau de bord des tâches

Identifier

Name

Create Time

Assignee

Description

Action

### Prototype des interfaces

User

CDG Logo

Upload

Inbox



**Figure.I.1 : Prototype - Interface de chargement de relevé bancaire**



**Figure.I.2 : Prototype - Interface des tâches d’Activiti workflow**

CDG Logo

Upload

Inbox

User

Section gestion des tâches

retourner aux tâches

Releve Bancaire

Les lignes de releve bancaire

releve bancaire ID

Date reception

Label

Nombres de lignes

Nombres des operation credit

Nombres des operation debit

Solde intial

Solde finale

Section gestion des tâches

retourner aux tâches

User

CDG Logo

Upload

Inbox

Releve Bancaire

Les lignes de releve bancaire

Date operation

Date value

Mode paiment

Montant

Acteur

Qualifier Acteur

Déja Q.SWP

Details

Qualifier SWP



**Figure.I.3 : Prototype - Interface de détails du relevé bancaire téléchargée**



**Figure.I.4 : Prototype - Interface de détails des lignes de relevés téléchargée**

### Conclusion

Dans ce chapitre nous avons spécifié à l’étude de la critique de l’existant, nous avons identifié nos objectives et présenté une étude préalable qui nous a aidés pour comprendre le contexte du notre applications web. Dans le chapitre suivant, on va déterminer les fonctionnalités et les acteurs le plus pertinents.

# Chapitre II : Analyse et spécification

### Introduction

La phase d'analyse et de spécification, est la première étape du processus de développement que nous avons adopté. En effet, elle formalise et détaille ce qui a été ébauché au cours de l'étude préliminaire, et permet de dégager l'étude fonctionnelle du système. Elle permet ainsi d'obtenir une idée sur ce que va réaliser le système en termes de métier (comportement du système).

Tout au long de ce chapitre, nous commencerons par définir les besoins fonctionnels et non fonctionnels de la solution que nous allons proposer, et présenter le diagramme général de cas d’utilisation.

### Analyse des besoins

Cette partie va servir à poser les bases du recueil des besoins du système à réaliser. Pour pouvoir clarifier les besoins des utilisateurs de notre application, nous allons présenter les besoins fonctionnels ainsi que les besoins non fonctionnels.

Voici en premier temps les besoins fonctionnels :

#### Les besoins fonctionnels

Il s'agit des fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du Système.

Le système à concevoir doit permettre à l’utilisateur d’effectuer les opérations suivantes :



* **Gestion des relevés bancaire** : Il s'agit d'un outil permettant de gérer tous les paiements bancaires obtenus et de les qualifier pour chaque opération de crédit ou de débit.
* **Gestion automatique du flux de travail :** l'application permet de :
  + La gestion du processus de chaque relevé bancaire.
  + La gestion des tâches progresser ou recule la tâche de chaque processus.
* **Chargement des fichiers d'extension .txt** : l’application fournit une vue pour qu’un utilisateur peut télécharger des fichiers txt dans la base de données.
* **L'authentification et l'autorisation** : L'application vérifie que l'utilisateur doit être authentifié et autorisé.

#### Les besoins non fonctionnels

Les besoins non fonctionnels concernent les contraintes à prendre en considération pour mettre en place une solution adéquate aux attentes des concepteurs des architectures dynamiques.

Notre application doit nécessairement assurer ces besoins :



* **L’extensibilité** : dans le cadre de ce travail, l'application devra être extensible, c'est-à-dire qu'il pourra y avoir une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités.
* **La sécurité** : l’application devra être hautement sécurisée, les informations ne devront pas être accessibles à tout le monde, c'est-à-dire que le site web est accessible par un identifiant et un mot de passe attribué à une personne physique.
* **L’interface** : avoir une application qui respecte les principes des Interfaces Homme/Machine (IHM) tels que l'ergonomie et la fiabilité.
* **La performance** : l'application doit être performante, c'est-à-dire que le système

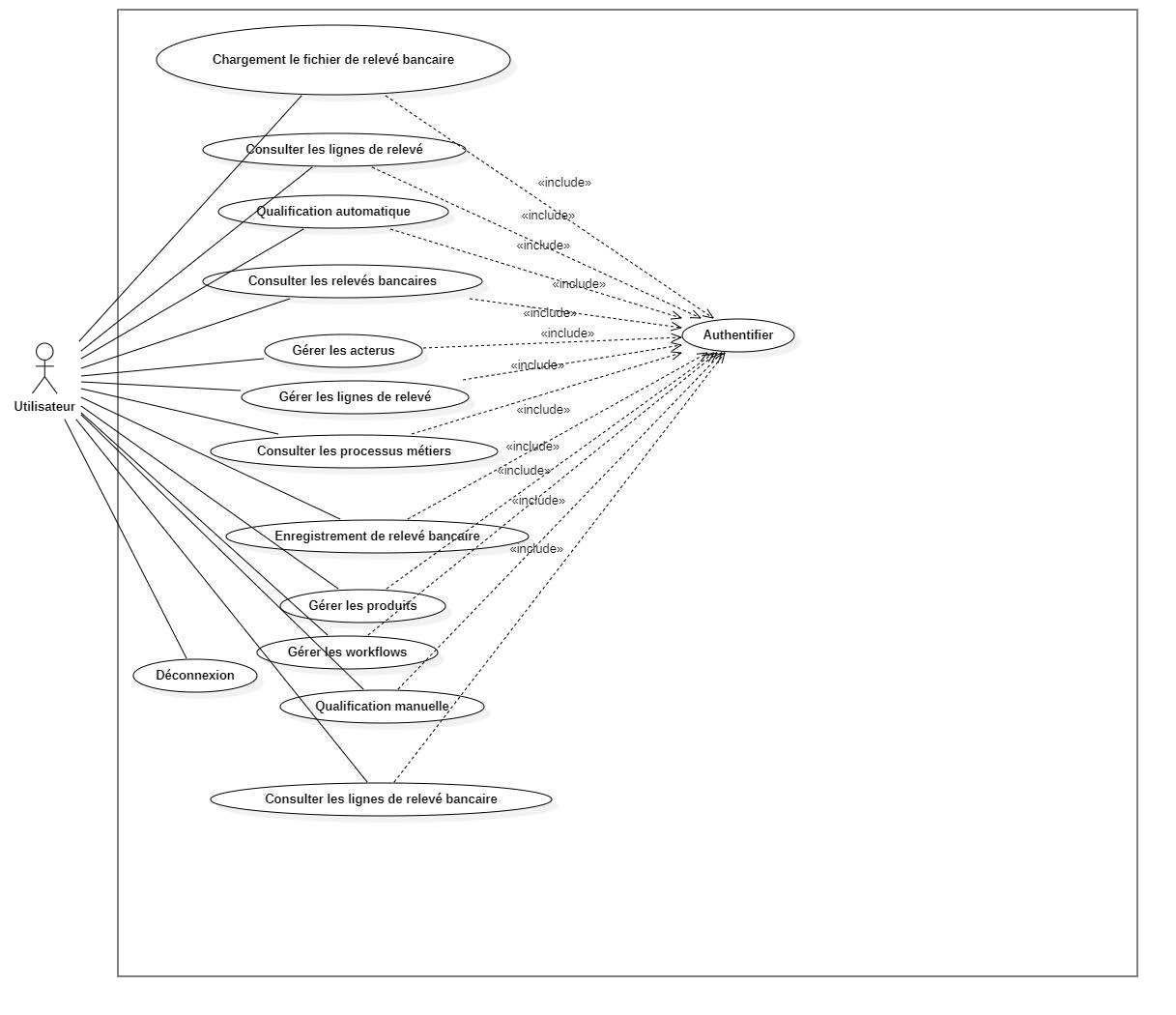
doit réagir dans un délai précis, quel que soit l'action de l'utilisateur.

* **La convivialité** : L'application doit être simple et facile à utiliser, même par des non-experts.
* **L’ergonomie** : le thème adopté par l'application doit être inspiré des couleurs et du logo type de l'entreprise d'accueil.

### Diagramme des cas d’utilisation général

#### Définition d’un cas d’utilisation

Le diagramme de cas d'utilisation décrit les utilisations requises d'un système, ou ce qu'un système est supposé faire. Les principaux concepts de ce diagramme sont les acteurs, cas d'utilisation et sujets. Un sujet représente un système avec lequel les acteurs et autres sujets interagissent.



**Figure II.1 : Diagramme de cas d’utilisation général de système**

#### Identification de l'acteur

Au niveau de cette section, nous présentons l'acteur susceptible d'interagir avec le système, mais avant tout, nous donnons une définition du concept d'acteur.

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié.

La mise en marche du système nécessite essentiellement quatre acteurs :

* **Utilisateur (employé)** : L'utilisateur doit être authentifié par l'administrateur, il a le droit de charger les fichiers relevés bancaires, ainsi que son rôle consiste à contrôler et clôturer les processus des workflows, qualifier et gérer tous les relevés bancaires et ses lignes et plus (nous verrons tout dans le prochain chapitre).

### Conclusion

Ce chapitre présente une phase indispensable pour l’étude et l’analyse de notre application. Nous avons défini les différents besoins fonctionnels et non fonctionnels, nous avons présenté le diagramme de cas d’utilisation général. Nous entamerons dans le chapitre suivant la conception de cette application qui comporte les diagrammes de cas d’utilisation détaillés, les différents diagrammes de séquence et le diagramme de classe.

# Chapitre III : Etude Conceptuelle

### Introduction

La phase de la conception est la phase initiale de la création et de la mise en œuvre de notre projet. En fait, elle représente une étape importante de réflexion dans le cycle de développement logiciel après la phase de l’analyse et de spécification.

Dans ce chapitre, nous allons présenter en détails la conception du projet à travers les diagrammes UML suivants : les diagrammes des cas d’utilisation, les diagrammes de séquence, le diagramme de classes et les diagrammes d’activités.

### Description des diagrammes

Dans le précédent chapitre, nous avons présenté le digramme de cas d’utilisation général qui décrit le comportement fonctionnel du système tel qu'il est vu par l'utilisateur, et ceci ne suffit plus, c’est pour cela, que nous allons traiter le digramme de séquence qui décrit le comportement dynamique entre l'acteur et les objets de système.

#### Définition de diagramme de séquence

Un diagramme de séquence est un document graphique qui montre pour des scénarios de cas d'utilisation précis, les événements générés et les interactions entre objets en se basant sur des messages ordonnés. Chaque message transitant sur un lien est symbolisé par une flèche porteuse d'une expression. La lecture se fait de haut en bas, et l'ordre chronologique doit respecter ce sens.

La réalisation de diagramme de séquence permet de lister les méthodes dont on aura besoin lors de la phase de développement. Pour ce faire, la description doit être suffisamment générale et exhaustive pour identifier tous les algorithmes [6].

#### Définition du diagramme d’activité

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements pouvant être parallèles. Le diagramme d'activité est également utilisé pour décrire un flux de travail.

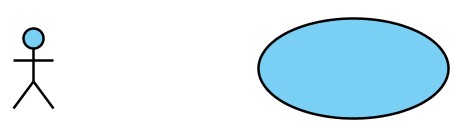
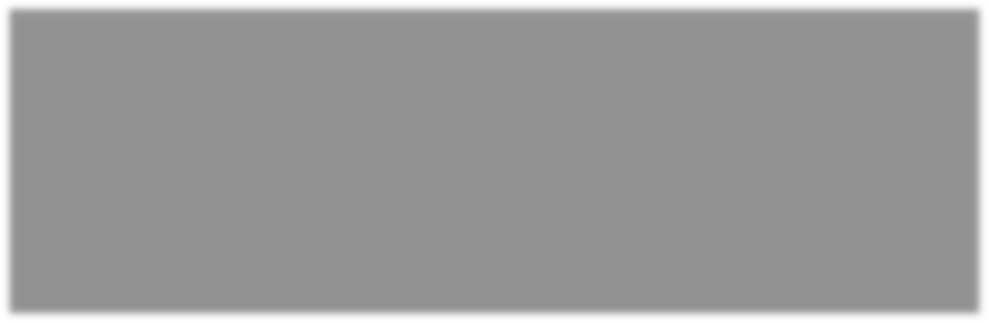
### Capture des besoins fonctionnels

La capture des besoins fonctionnels est la première étape de la branche gauche du cycle en

Y. Elle sera complétée au niveau de la branche droite du Y par la capture des besoins techniques.

#### Description des cas d'utilisation

###### Cas d’utilisation « gestion des relevés bancaires »



**Gestion des relevés bancaires**



**Figure III.1: Diagramme de cas d’utilisation « gestion des relevés bancaires »**

* **But :** Ce cas d'utilisation permet à l'utilisateur de gérer tous les relevés.
* **Les acteurs :** Utilisateur (employé).
* **Description textuelle :** Lors de la réception d'un versement, l'utilisateur devra télécharger le relevé bancaire, après qu'un message de confirmation s'affiche à l'écran pour informer l'acteur que le fichier a été téléchargé avec succès, cela créera un flux de travail (workflow) pour automatiser toutes les tâches, la qualification automatique parviendra à placer chaque chose à sa place spécifique, à moins que si nous ne connaissons pas certaines informations, ici la qualification manuelle entre en jeu.

#### Les scénarios

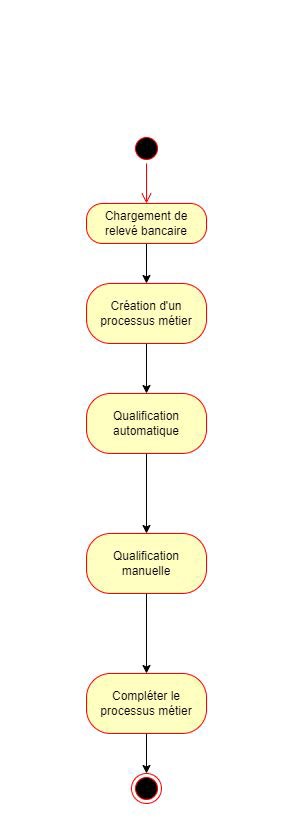
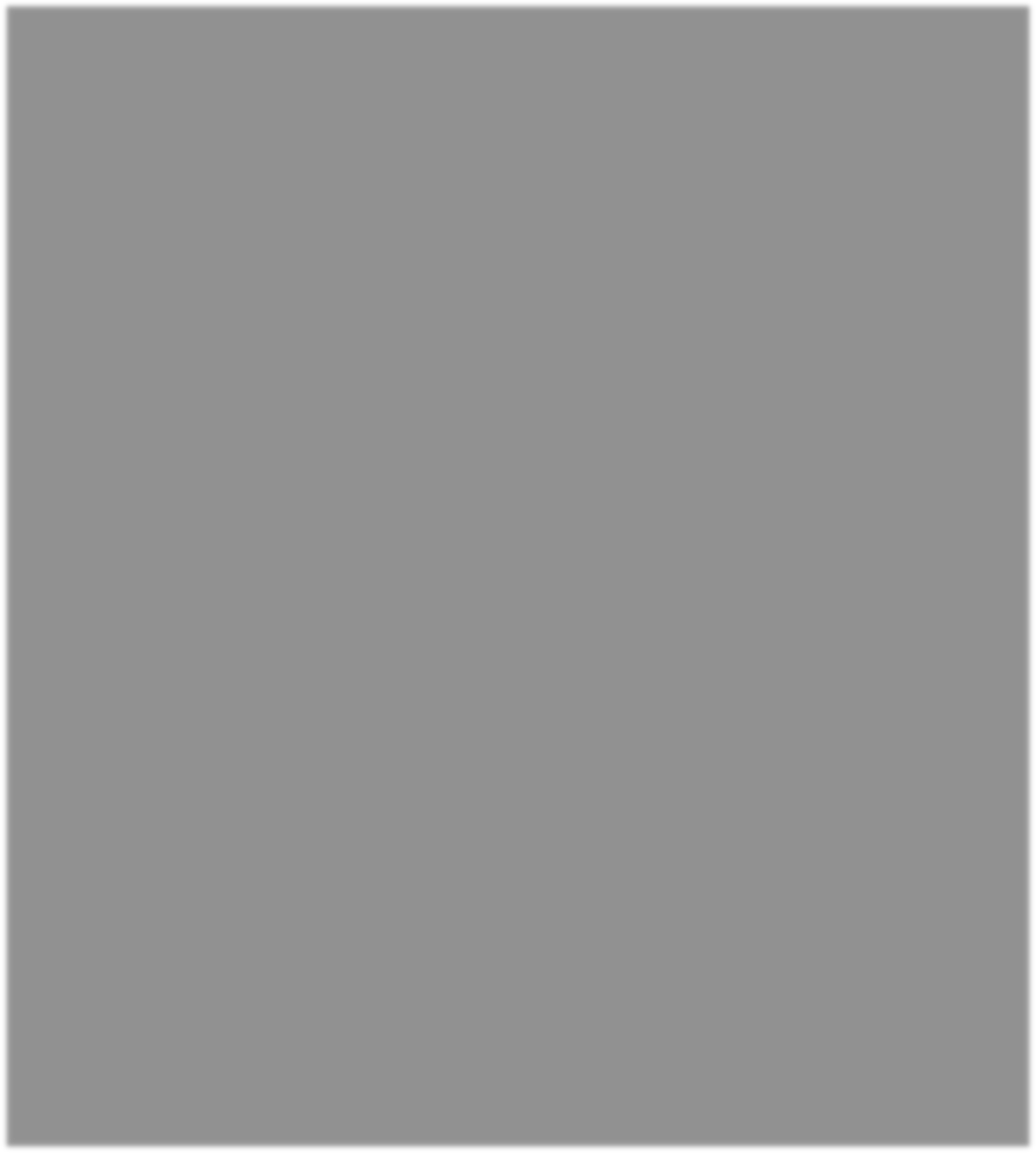
* La gestion d'un relevé bancaire : permet de gérer le relevé bancaire selon l’enchaînement suivant :
  + - 1. L’utilisateur doit être authentifié et autorisé par l'administrateur.
      2. Le système affiche un écran pour le chargement du relevé bancaire (chargement fichier texte).
      3. Un processus métier sera créé.
      4. Qualifier manuellement les lignes de relevés pour chaque acteur et chaque produite.
* Gestion d’un processus métier : permet de contrôler les processus selon l’enchaînement suivant :

1. L’utilisateur peut gérer des tâches pour chaque processus.
2. Le processus défini par séquence de tâches l'utilisateur peut passer à la tâche suivante ou précédente.
3. L’utilisateur passera à la qualification manuelle des versements.
4. Le système vérifie les champs du relevé bancaire une fois que l'utilisateur a terminé la tâche.

* Annulation d’un processus métier : permet d’annuler un processus selon l’enchaînement suivant :

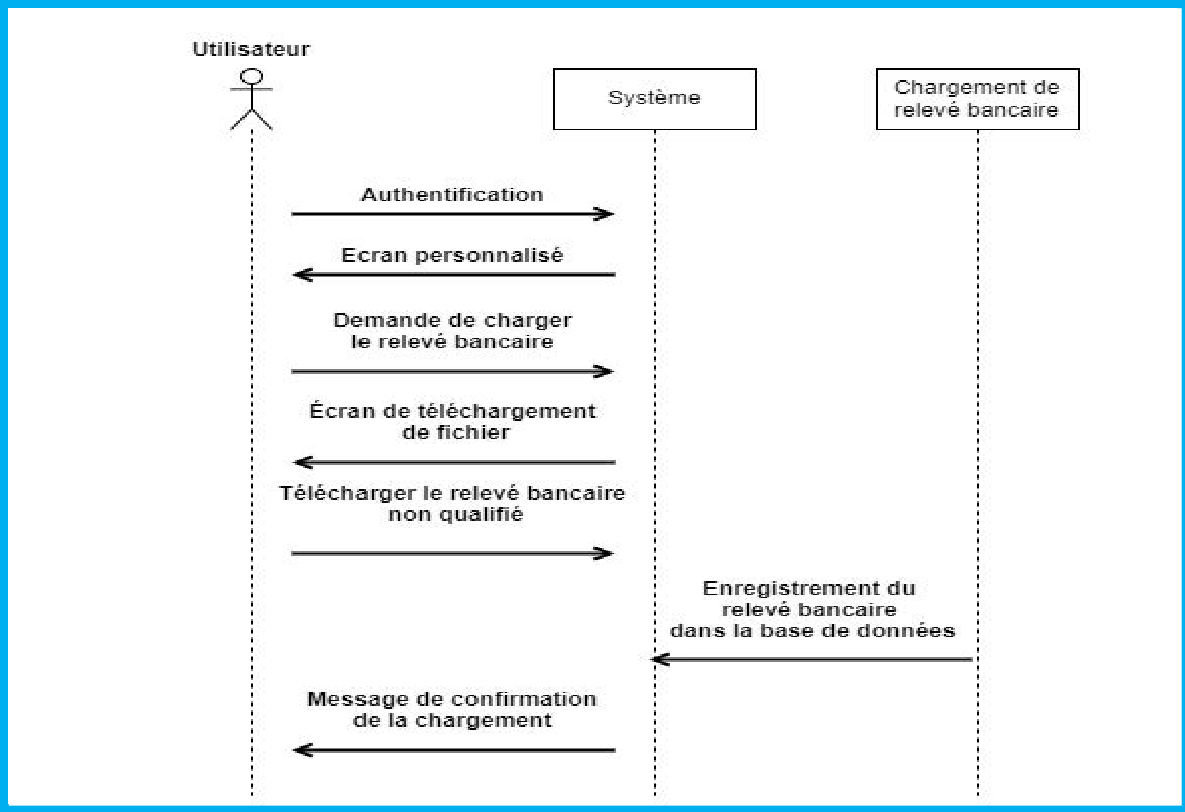
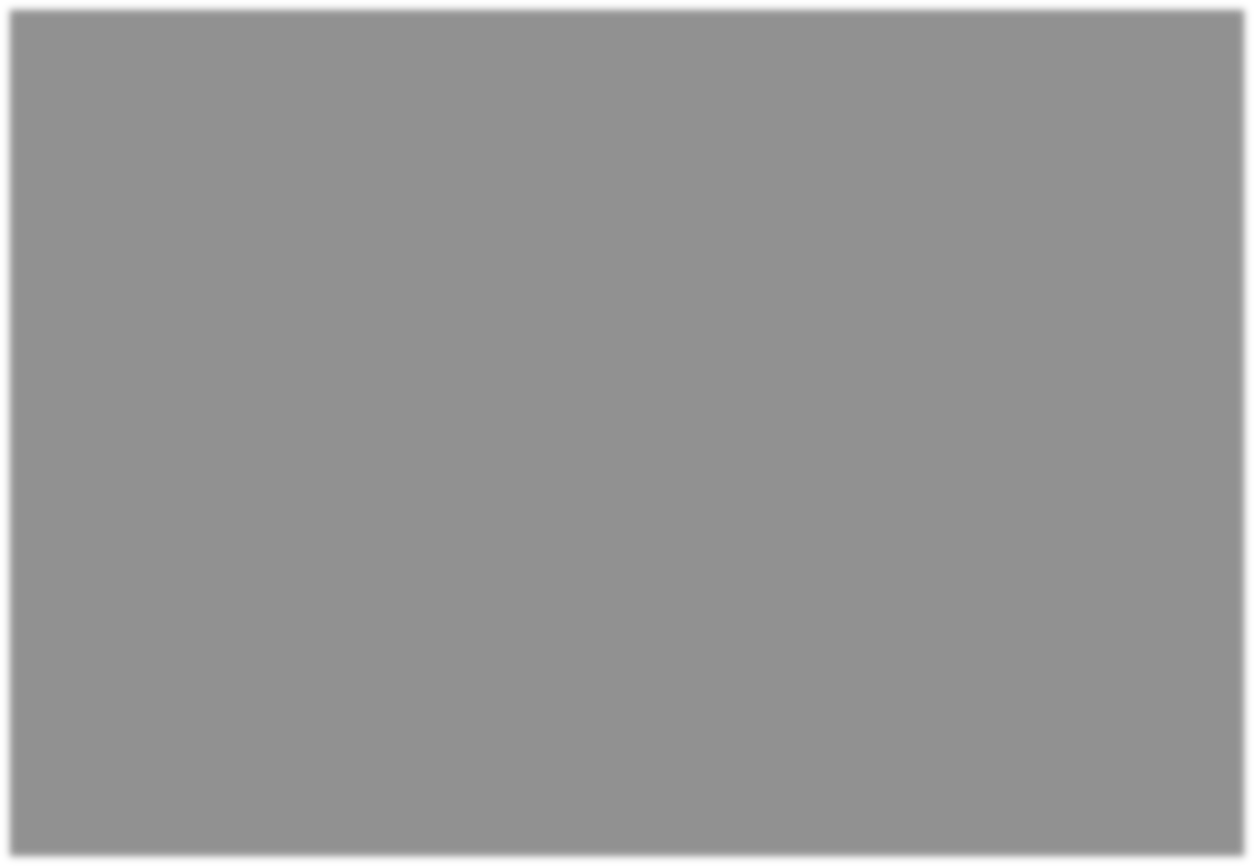
1. L’utilisateur accède à la page des tâches.
2. L’utilisateur sélectionne la tâche concernée.
3. Le système affiche les détails de relevé bancaire avec ses lignes.
4. Le système affiche un avertissement sur laquelle ce relevé déjà qualifier.
5. Dans le cas où l'utilisateur confirme l’annulation, le processus sélectionné sera terminé par le système.

###### Description Diagramme d’activité « Gestion des relevés bancaires »



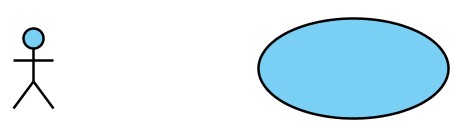
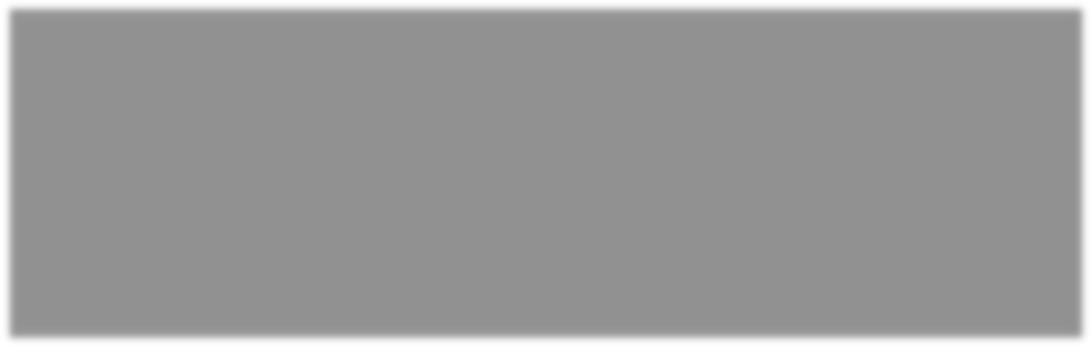
**Figure III.2 : Diagramme d’activité « Gestion des relevés bancaires »**

* + **Le scénario « le chargement d'un relevé bancaire »**



**Figure III.3 : Diagramme de séquence du scénario « Le chargement de relevé bancaire »**

###### Cas d’utilisation « gestion des workflows » :



**Gestion des workflows**



**Figure III.4 : Diagramme des cas d’utilisation « Gestion des Workflows »**

* + **But :** Ce cas d'utilisation permet à l'utilisateur du système passer à la tâche suivante, retour à la tâche précédente ou finissez la tâche. à travers ca on peut gérer les relevés bancaires avec ses lignes.
* **Les acteurs :** L'utilisateur.
* **Description textuelle :** Dès que le téléchargement du relevé bancaire est téléchargé avec succès, un flux de travail sera créé automatiquement, avec ce flux de travail, nous pouvons gérer notre relevé bancaire de manière moderne et automatique.

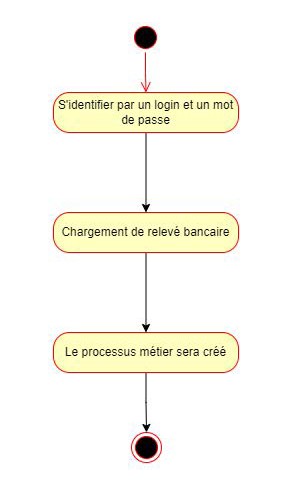
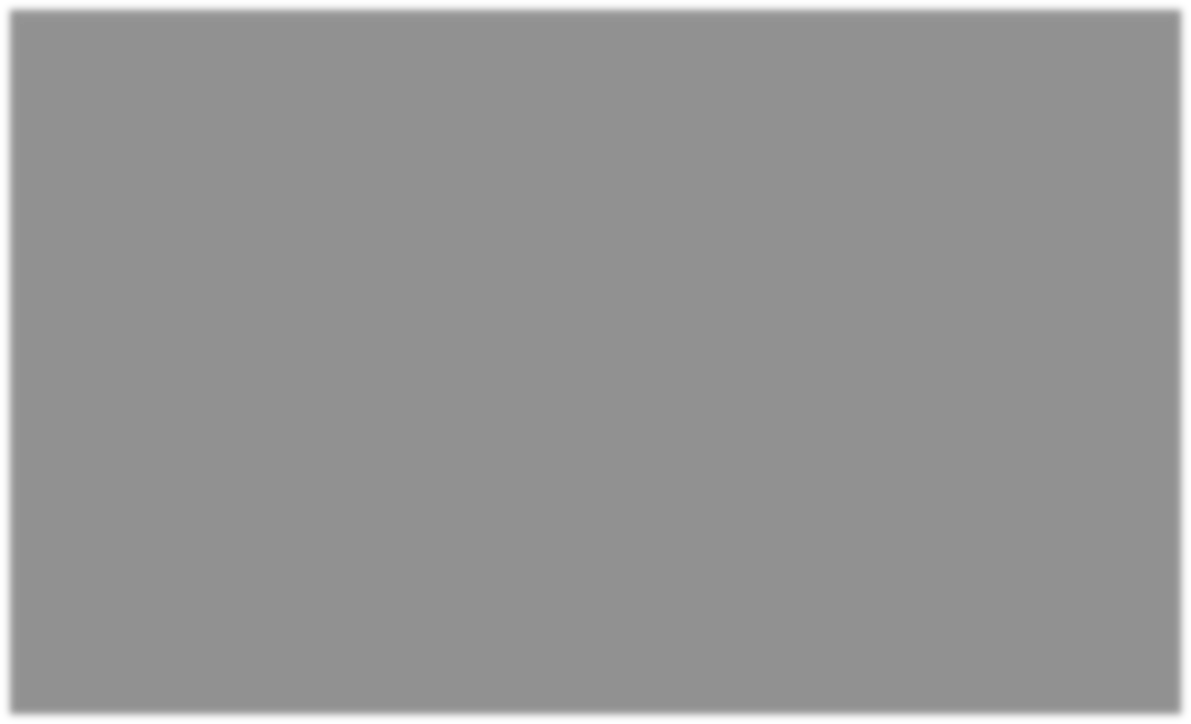
Lorsque le système a créé le workflow, nous pouvons accéder à la page des workflows et nous montrera les détails de ce workflow (identifiant, nom, date de création, cessionnaire, description et action) lorsque nous cliquons sur le bouton d'action vers lequel nous pouvons aller une autre page qui affichera les détails de ce personnel bancaire.

###### Les scénarios

* + **Création d'un workflow :** permet de crée un workflow à la liste des workflows selon l’enchaînement suivant :

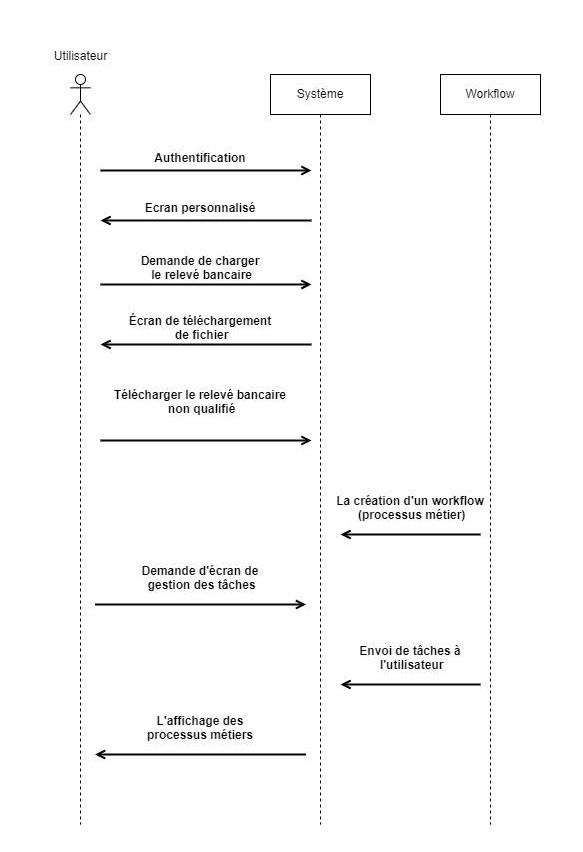
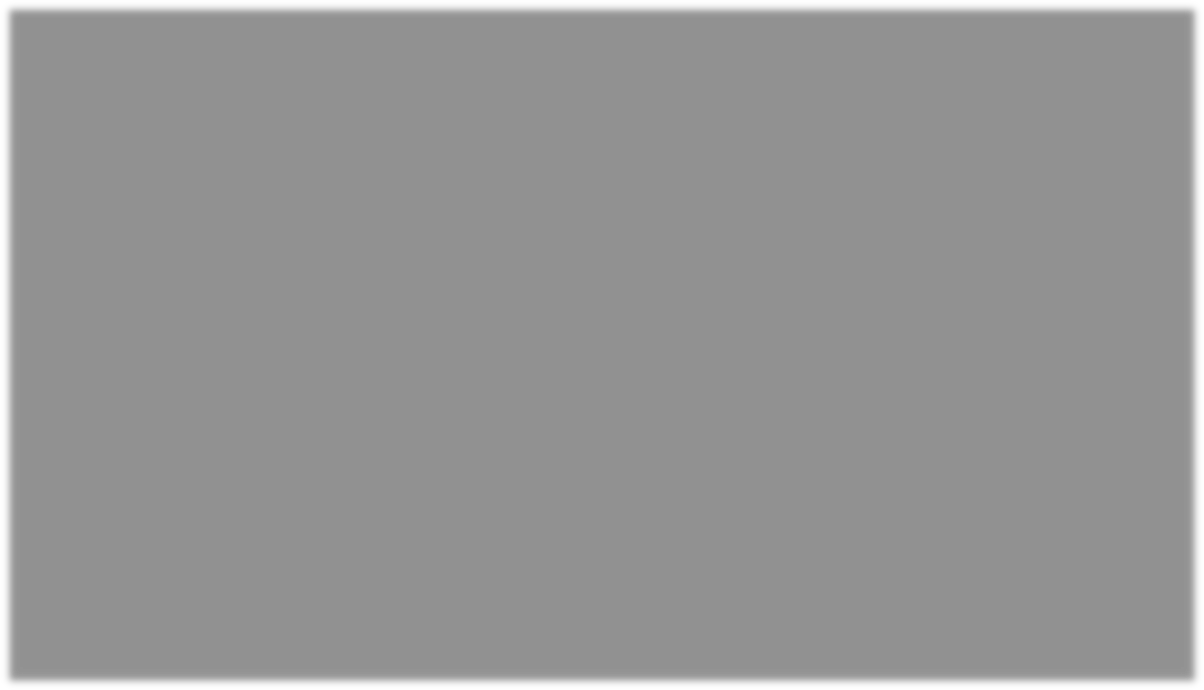
1. L’utilisateur doit être authentifié et autorisé par l'administrateur.
2. Le système affiche un écran pour le chargement du relevé bancaire (chargement fichier texte).
3. Un processus métier sera créé.
4. L’utilisateur demande la page de la boîte de réception.
5. Le système affiche un écran de boîte de réception qui contient toutes les tâches qui ne sont pas terminées.

###### Description du diagramme d’activité « gestion des workflows »



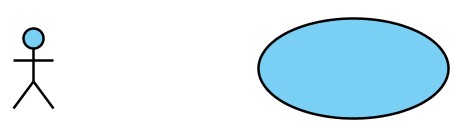
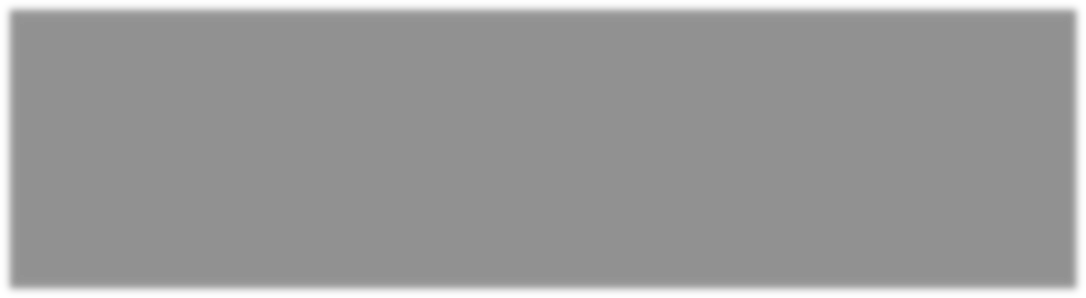
**Figure III.5 : Diagramme d’activité « Création d'un workflow (processus métier) »**

* + **Le scénario « Crée un processus métier »**



**Figure III.6 : Diagramme de séquence du scénario « Crée un processus métier »**

###### Cas d’utilisation « Qualification automatique »



Qualification automatique



**Figure III.7 : Diagramme d’activité « Qualification automatique »**

* + - **But :** Ce cas d'utilisation permet à l'utilisateur d'effectuer une qualification de manière automatique.
    - **Les acteurs :** Utilisateur (employé).
    - **Description textuelle :** Dès que le flux de travail (workflow) a été créé avec succès, cela fera une opération avec une manière automatique de placer chaque donnée à sa place spécifique dans la base de données.

#### Les scénarios

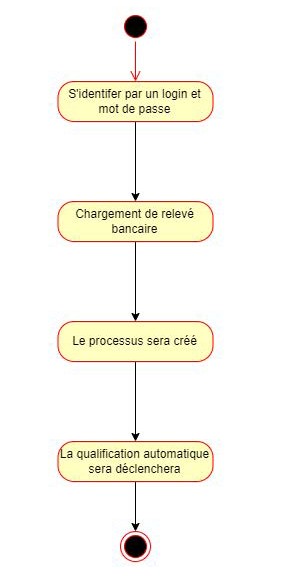
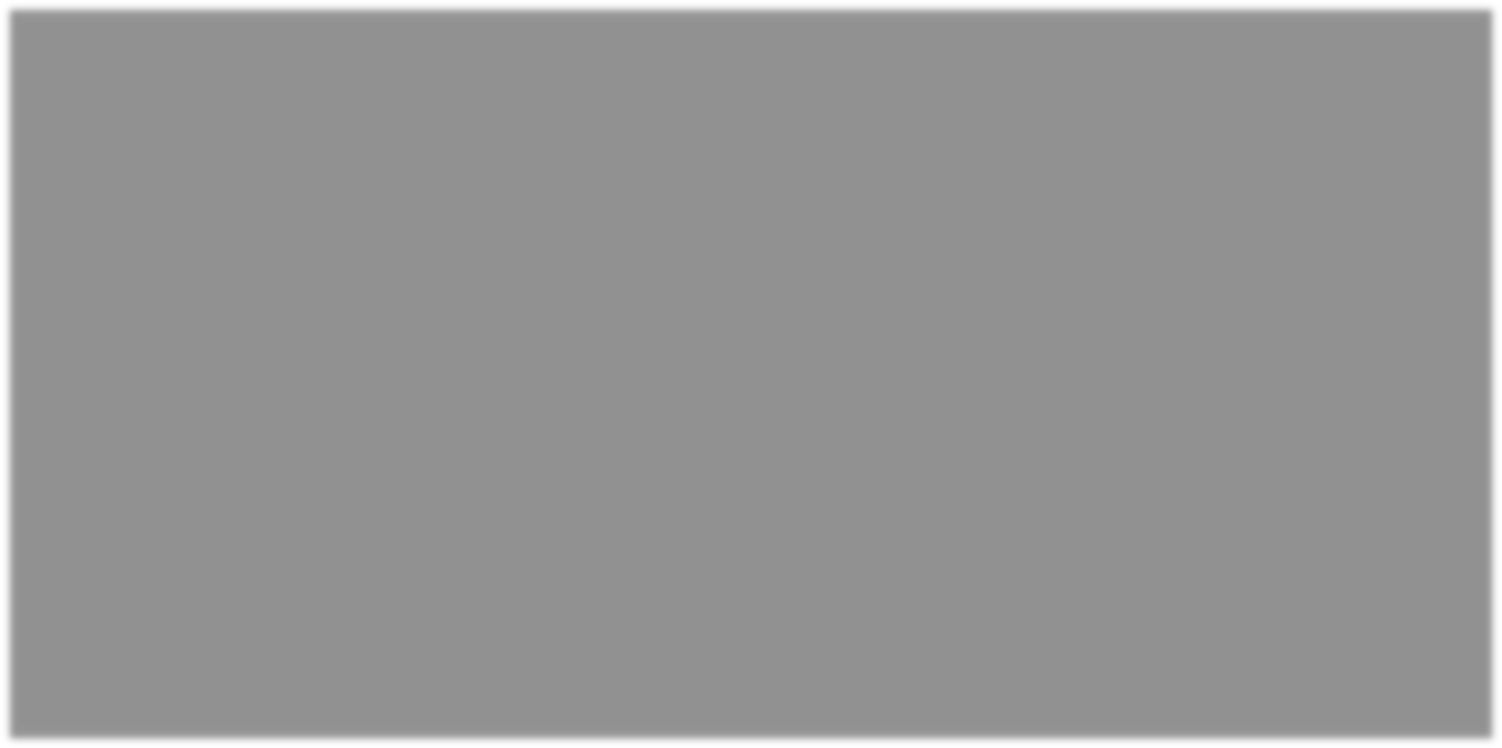
###### Planifier

1. L’utilisateur doit être authentifié et autorisé par l'administrateur.
2. Le chargement d'un relevé bancaire.
3. Au moment de création d'un workflow la qualification automatique se déclenchera.

###### Consulter le résultat de la qualification automatique

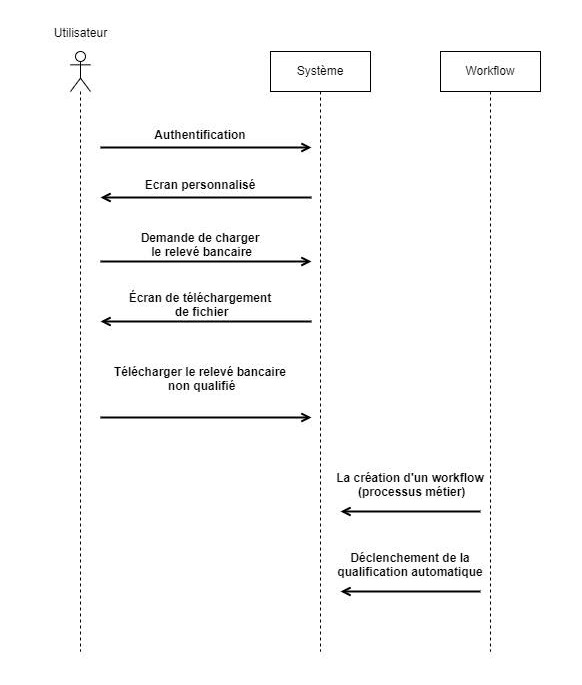
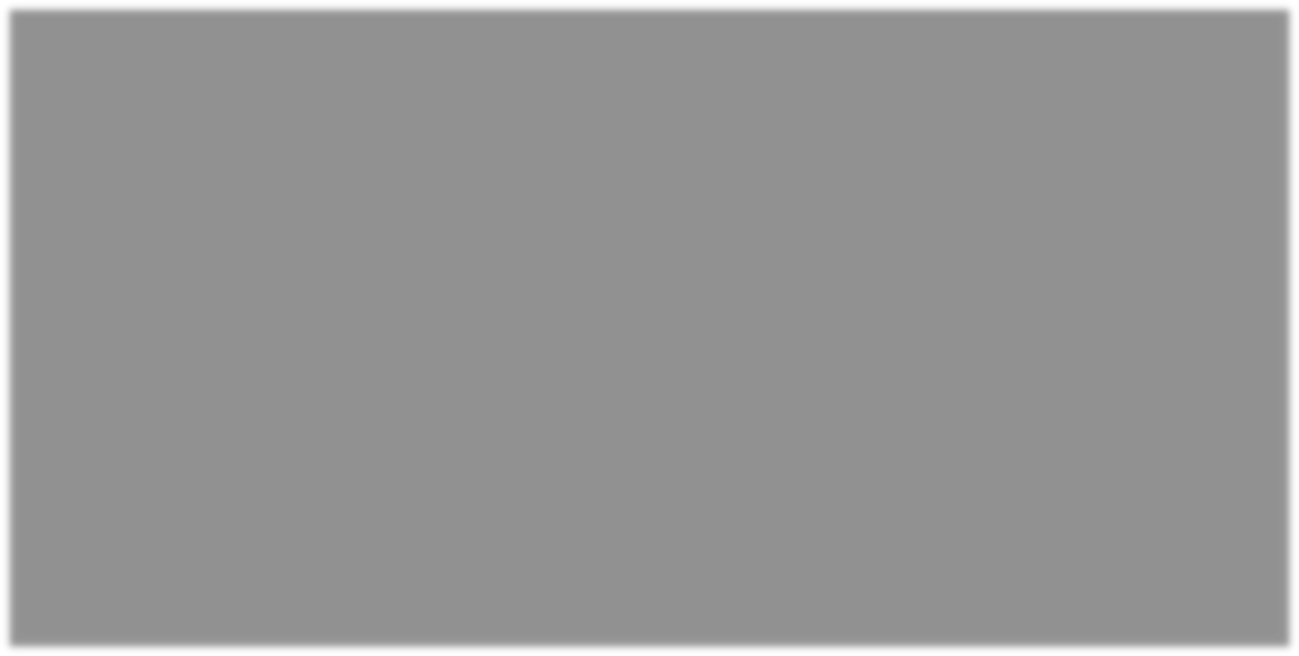
1. Nous pouvons consulter les données dans notre base de données.
2. Nous pouvons également afficher nos résultats avec la section de révision qui sera abordée sur ce sujet dans le cas suivant

###### Description Diagramme d’activité « Qualification automatique »



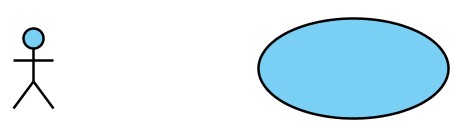
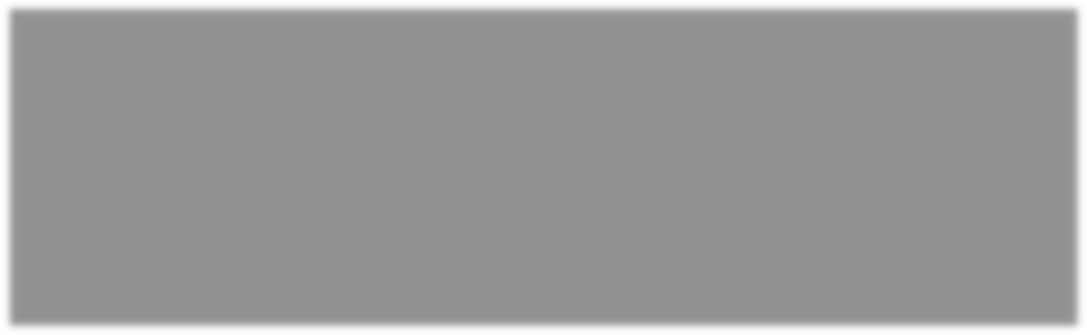
**Figure III.8 : Diagramme d’activité « Qualification automatique »**

* + **Le scénario « qualification automatique»**



**Figure III.9 : Diagramme de séquence du scénario « Qualification automatique »**

###### Cas d’utilisation « Consultation de relevé bancaire »



**Consultation de relevé bancaire**



**Figure III.10 : Diagramme des cas d’utilisation « Consultation de relevé bancaire »**

* + - **But** Ce cas d'utilisation permet à l'utilisateur de consulter le relevé bancaire chargé à travers le processus spécifique de ce relevé (Date réception, label, nombres de lignes, nombres des opération crédit, nombres des opération débit, solde initial, solde final).
    - **Les acteurs** L'utilisateur.
    - **Description textuelle** ce cas d'utilisation aide l'utilisateur à analyser le relevé bancaire et ses lignes pour prendre des décisions.

###### Les scénarios

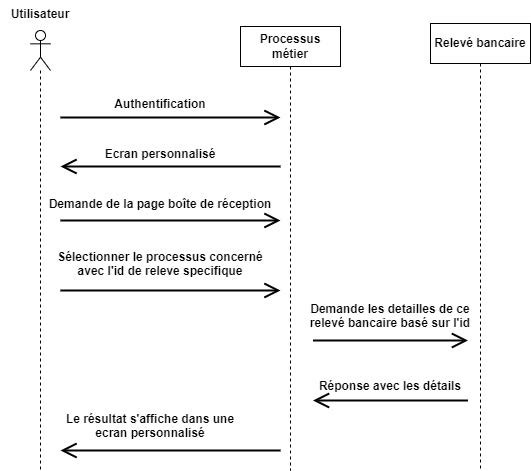
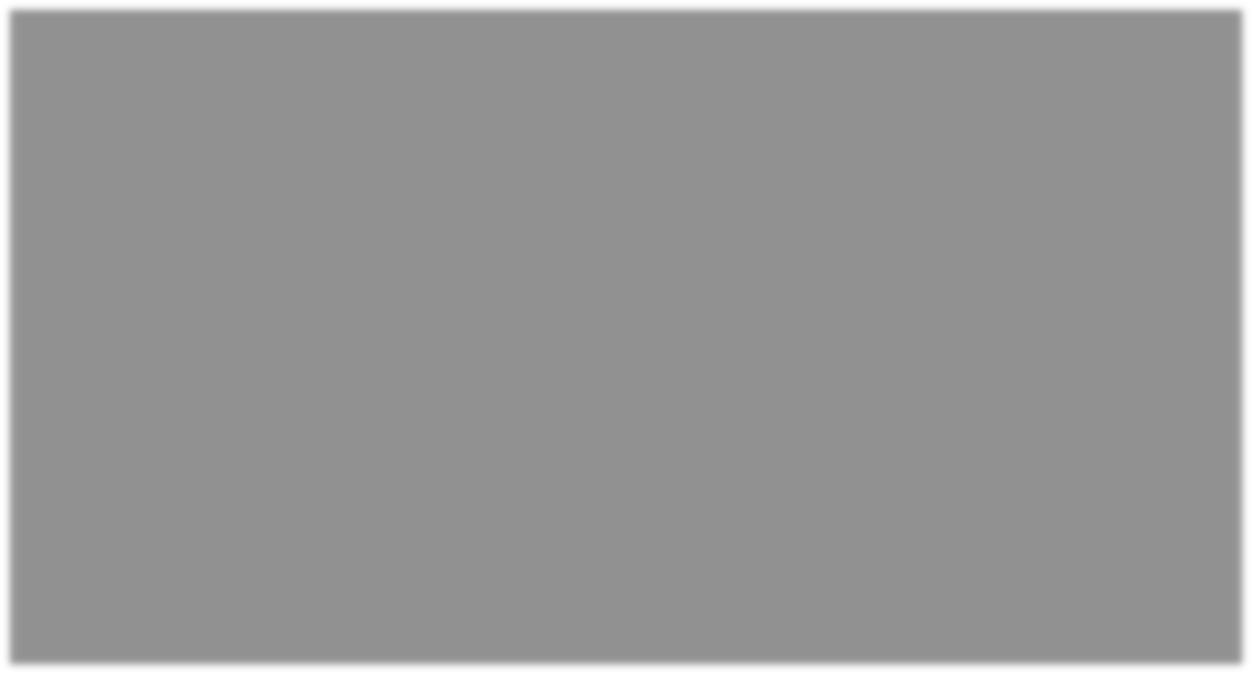
* **Consultation du relevé bancaire et les lignes :**

Permet d’enregistrer une facture de sous-traitance selon l'enchaînement suivant :

1. L'utilisateur demande au système de consulter les taches incomplètes.
2. Le système affichera un tableau des processus métier. 3- L'utilisateur sélectionne le processus spécifique.

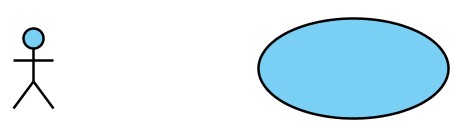
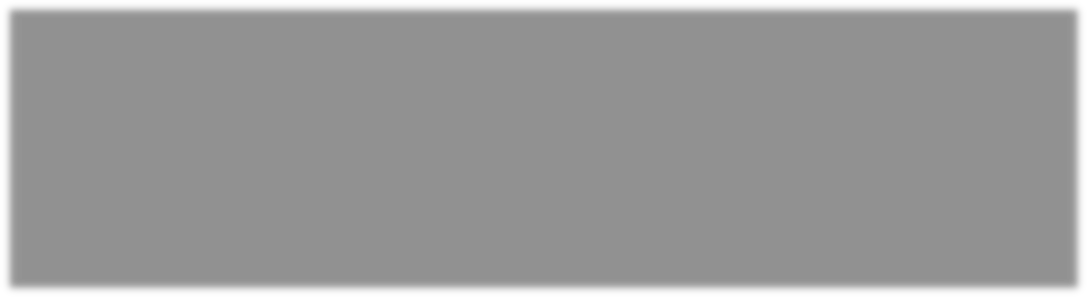
4- Le système affiche les détails de relevé bancaire avec ses lignes.

###### Le scénario « Consultation d'un relevé bancaire »



**Figure III.11 : Diagramme de séquence du scénario « Consultation de relevé bancaire »**

* + 1. **Cas d’utilisation « Qualification manuelle des produites »**



**Qualification manuelle des produites**



**Figure III.1*2* : Diagramme des cas d’utilisation « Qualification manuelle des produites »**

* **But** Ce cas d'utilisation permet à l'utilisateur de qualifier les lignes de relevé bancaire et spécifiquement la partie qui concerne les produits de la CDG Prévoyance (que ça soit RCAR-RG : Régime général, RCAR-RC : Régime complémentaire, RECORE : Régime complémentaire de retraite ou ADI : Assurance décès invalidité.) On va faire ça de manière manuelle.
* **Les acteurs** L'utilisateur.
* **Description textuelle** Ce cas d'utilisation permet à l'utilisateur de qualifier les produits a chaque acteur.

###### Les scénarios :

* + **Ajouter un produit** permet d’ajouter un produit selon l’enchaînement suivant :

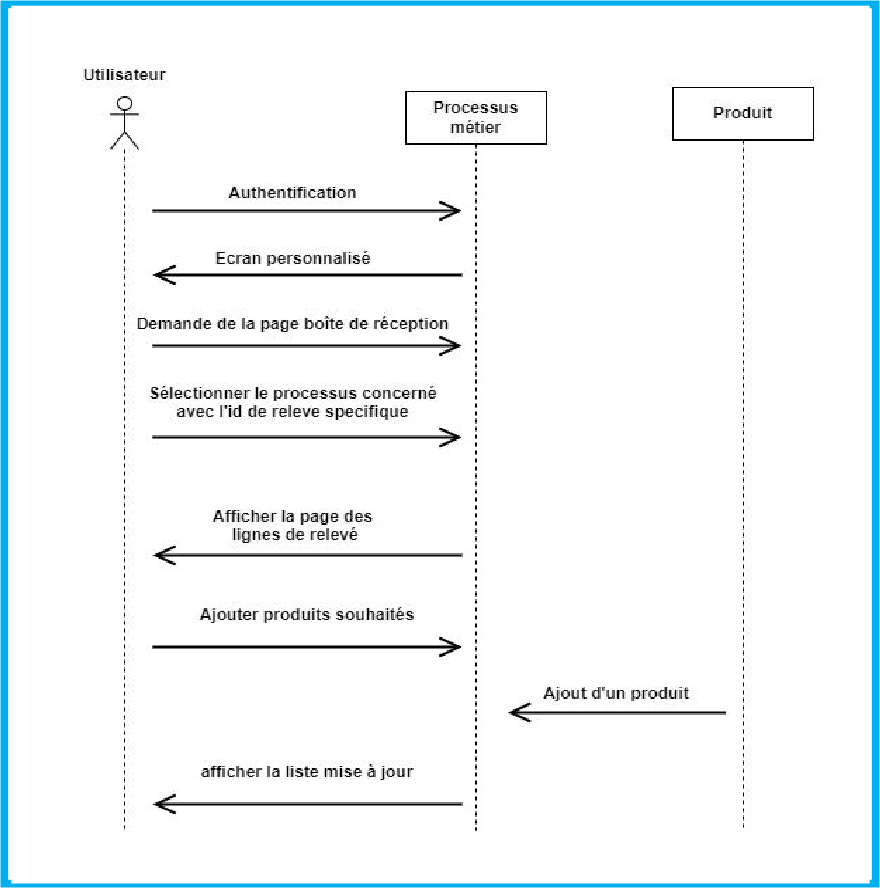
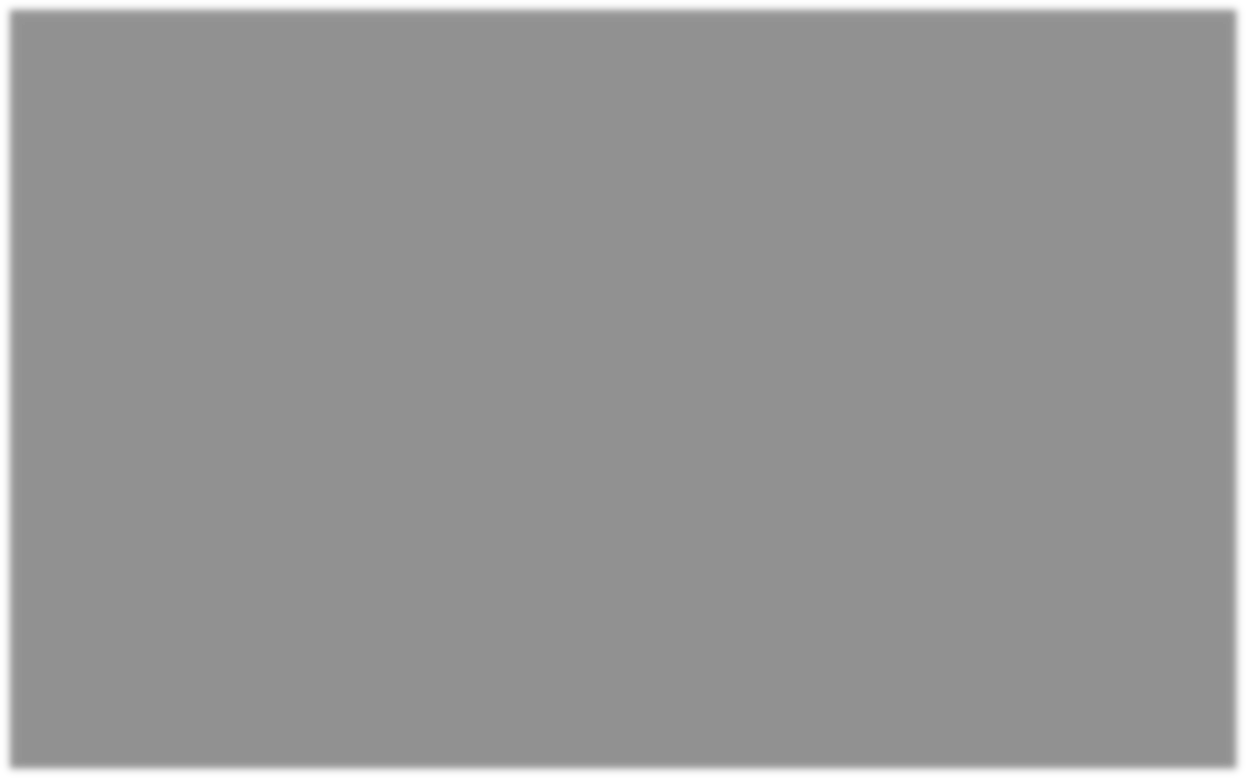
1. L’utilisateur demande au système d'afficher les workflows.
2. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
3. L’utilisateur sélectionne le workflow concerner.
4. Le système affiche les détails de relevé bancaire avec ses lignes.
5. L’utilisateur navigue aux les lignes de relevé bancaire.
6. Le système affiche un tableau des lignes.
7. L’utilisateur clique sur une icône sous la ligne Q.SWP pour ajouter un produit sur la ligne sélectionnée.
8. Une boîte de dialogue apparaît avec les produits disponibles (RCAR-RG : Régime général, RCAR-RC : Régime complémentaire, RECORE : Régime complémentaire de retraite ou ADI : Assurance décès invalidité).
9. Lors de la sélection du produit concerné sera finalement ajouté à cette ligne à cet acteur spécifique.
   * **Modification d'un produit** permet de modifier un produit selon l’enchaînement suivant :
10. L’utilisateur demande au système d'afficher les workflows.
11. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
12. L’utilisateur sélectionne le workflow concerner.
13. Le système affiche les détails de relevé bancaire avec ses lignes.
14. L’utilisateur navigue aux les lignes de relevé bancaire.
15. Le système affiche un tableau des lignes.
16. L’utilisateur clique sur une icône sous la ligne Q.SWP qui est déjà qualifiée 8- après cela, l'utilisateur sélectionne le produit qui doit être modifié.

9- le système affiche la liste modifiée.

* **Supprimer un produit** permet de supprimer un produit selon l’enchaînement suivant :

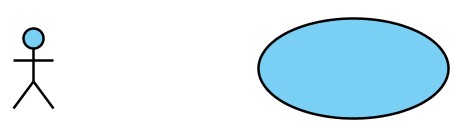
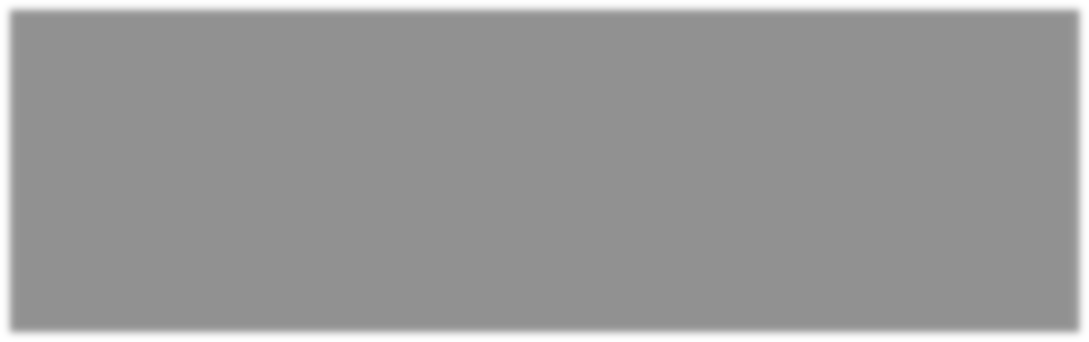
1. L’utilisateur demande au système d'afficher les workflows.
2. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
3. L’utilisateur sélectionne le workflow concerner.
4. Le système affiche les détails de relevé bancaire avec ses lignes.
5. L’utilisateur navigue aux les lignes de relevé bancaire.
6. Le système affiche un tableau des lignes.
7. Le système affiche les produits qui sont déjà qualifié a cote le nom de produit Il y a une icône de suppression.
8. L’utilisateur clique sur l'icône de suppression et le produit sera supprimé.
9. Le système affiche la liste modifiée.

#### Le scénario « Qualification manuelle : ajouter un produit »



**Figure III.13 : Diagramme de séquence du scénario « ajouter un produit »**

###### Cas d’utilisation « Qualification manuelle des acteurs »



**Qualification manuelle des acteurs**



**Figure III.14 : Diagramme de cas d’utilisation « Qualification manuelle des acteurs »**

* **But** Ce cas d'utilisation permet à l'utilisateur de qualifier les lignes de relevé bancaire et spécifiquement la partie qui concerne les acteurs de la CDG Prévoyance (rechercher un acteur et l'appliquer à un produit) On va faire ça de manière manuelle.
* **Les acteurs** L'utilisateur.
* **Description textuelle** Ce cas d'utilisation permet à l'utilisateur de qualifier les

acteurs à chaque produit.

###### Les scénarios :

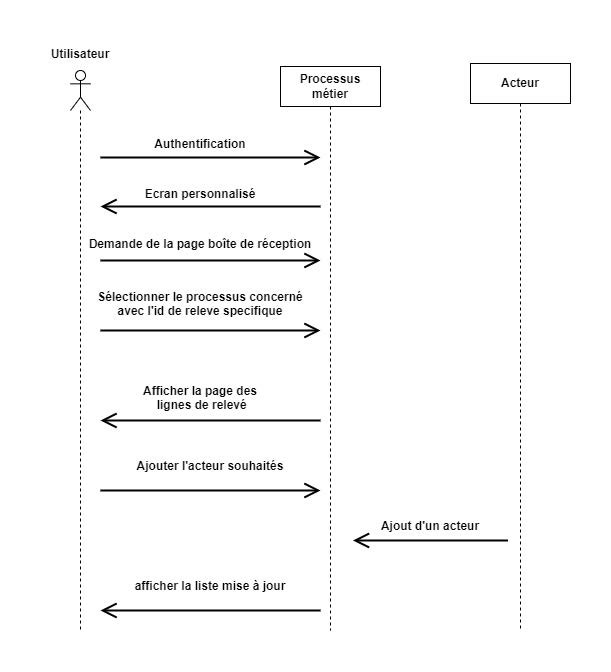
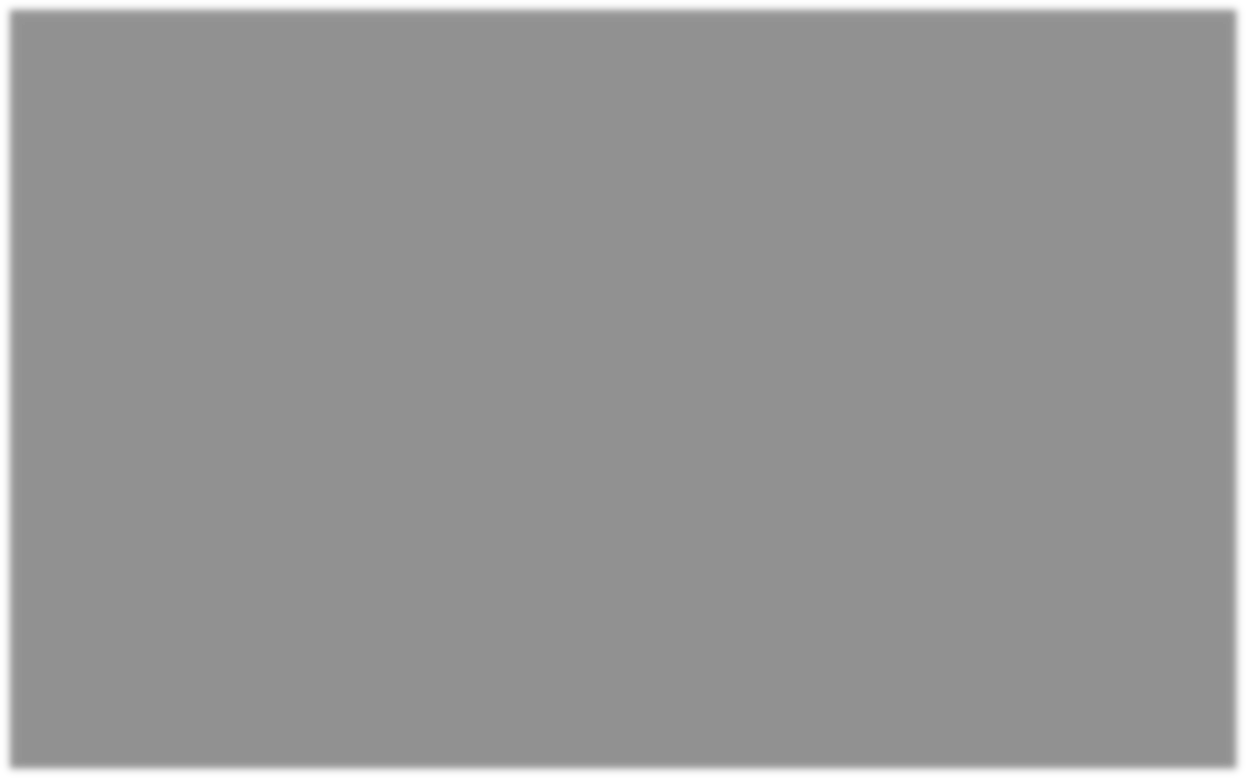
* + **Ajouter un acteur** permet d’ajouter un acteur selon l’enchaînement suivant :

1. L’utilisateur demande au système d'afficher les workflows.
2. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
3. L’utilisateur sélectionne le workflow concerner.
4. Le système affiche les détails de relevé bancaire avec ses lignes.
5. L’utilisateur navigue aux les lignes de relevé bancaire.
6. Le système affiche un tableau des lignes.
7. L’utilisateur clique sur une icône sous la ligne Q.Acteur pour ajouter un acteur

sur la ligne sélectionnée.

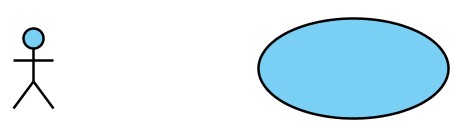
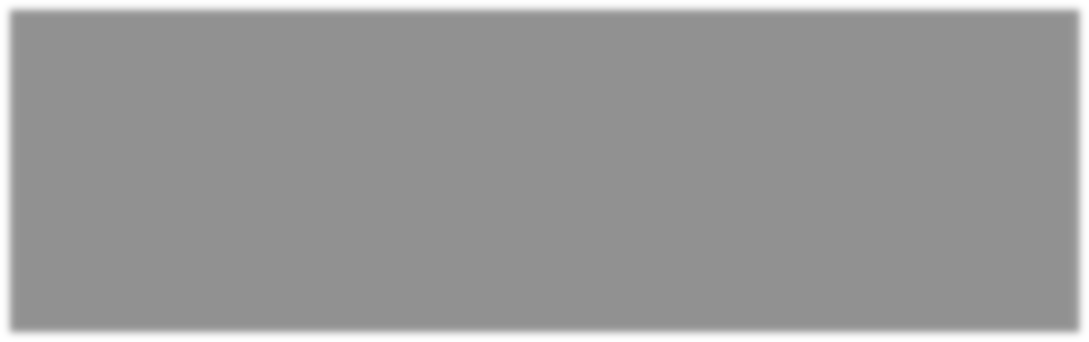
1. Une fenêtre de recherche apparaît.
2. L’utilisateur recherche l'acteur spécifique et l'applique à un produit déjà qualifié.
3. Le système affiche le tableau modifié.
   * **Modification d'un acteur** permet de modifier l'acteur selon l’enchaînement suivant :
4. L’utilisateur demande au système d'afficher les workflows.
5. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
6. L’utilisateur sélectionne le workflow concerner.
7. Le système affiche les détails de relevé bancaire avec ses lignes.
8. L’utilisateur navigue aux les lignes de relevé bancaire.
9. Le système affiche un tableau des lignes.
10. L’utilisateur clique sur une icône sous la ligne Q.Acteur qui est déjà qualifiée.
11. Après cela, l'utilisateur chercher et sélectionne l'acteur qui doit être modifié.
12. Le système affiche le tableau modifié.
    * **Supprimer un acteur** permet de supprimer un acteur selon l’enchaînement suivant :
13. L’utilisateur demande au système d'afficher les workflows.
14. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
15. L’utilisateur sélectionne le workflow concerner.
16. Le système affiche les détails de relevé bancaire avec ses lignes.
17. L’utilisateur navigue aux les lignes de relevé bancaire.
18. Le système affiche un tableau des lignes.
19. Le système affiche les acteurs qui sont déjà qualifié a cote le nom de l'acteur il y a une icône de suppression.
20. L’utilisateur clique sur l'icône de suppression et l'acteur sera supprimé.
21. Le système affiche la liste modifiée.

#### Le scénario « Qualification manuelle : ajouter un acteur »



**Figure III.15 : Diagramme de séquence du scénario « ajouter un acteur »**

###### Cas d’utilisation « Enregistrement de relevé bancaire »



**Enregistrement de relevé bancaire**



**Figure III.16 : Diagramme de cas d’utilisation « Enregistrement de relevé bancaire »**

* **But** Ce cas d'utilisation permet à l'utilisateur d’enregistrer le relevé bancaire après la qualification automatique et manuelle.
* **Les acteurs** L'utilisateur.
* **Description textuelle** ce cas d'utilisation aide l'utilisateur à terminer les opération crédit ou débit à chaque acteur et chaque produit.

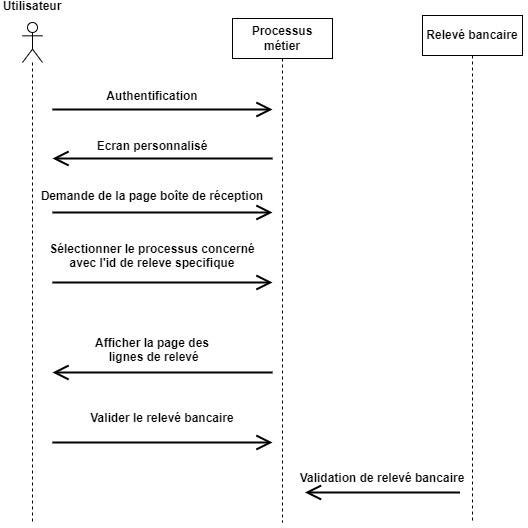
###### Les scénarios

* + **Enregistrement d'un relevé bancaire** permet d'enregistrer le relevé bancaire

Selon l’enchaînement suivant :

1. L’utilisateur demande au système d'afficher les workflows.
2. Le système affiche une liste des processus métiers de chaque relevé bancaire spécifique.
3. L’utilisateur sélectionne le workflow concerner.
4. Le système affiche les détails de relevé bancaire avec ses lignes.
5. L’utilisateur navigue aux les lignes de relevé bancaire.
6. Le système affiche toutes les lignes qualifiées.
7. L’utilisateur clique sur le bouton "Valider".

* **Le scénario « Enregistrement de relevé bancaire »**



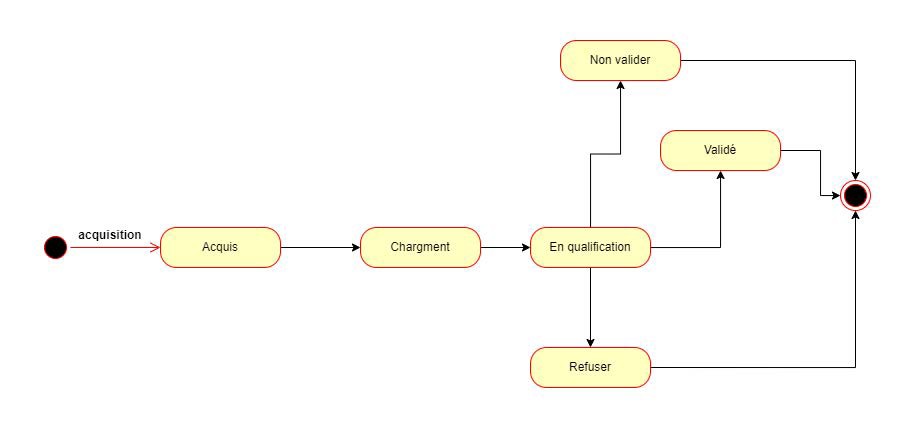
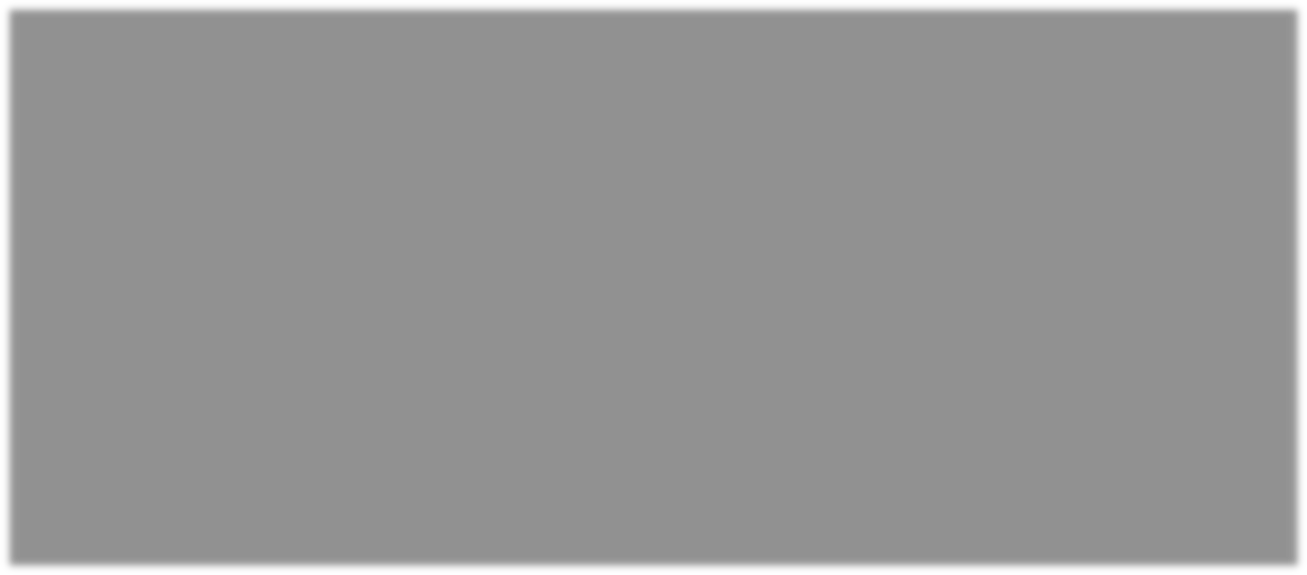
**Figure III.17 : Diagramme de séquence du scénario « Enregistrement de relevé bancaire »**

### Description de Les diagrammes des états transition

#### La classe « Relevé Bancaire »

Le relevé bancaire peut transiter entre plusieurs états différents :

Il passe par l’état **« chargement »** juste après son acquisition, il peut être **« validé »** s’il est affecté directement à un secteur, comme il peut être **« en qualification »** dans le cas si le processus n’est pas terminé ou **« non valider »**.

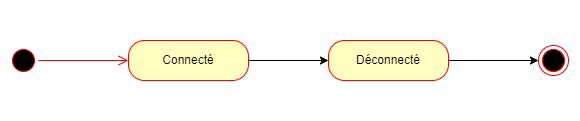
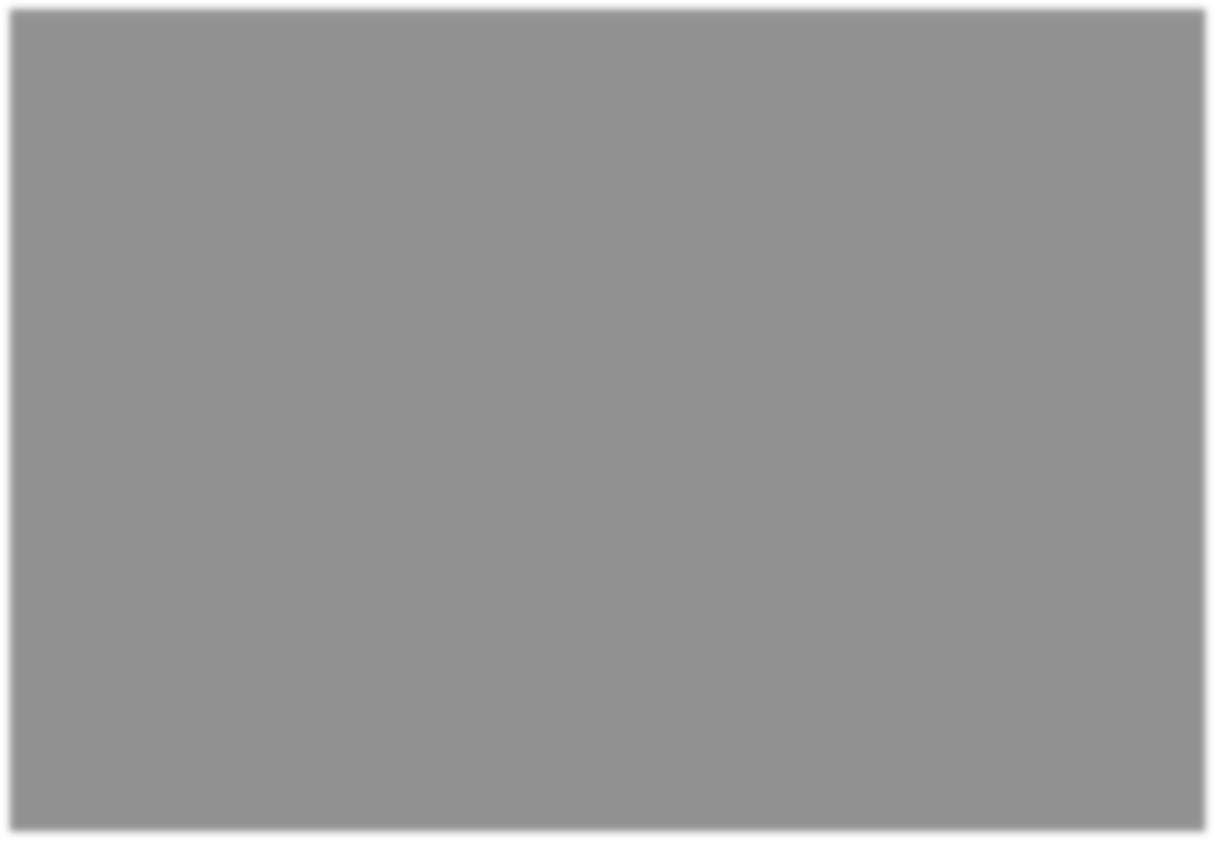


**Figure III.18 : Diagramme état transition « relevé bancaire »**

#### La classe « utilisateur »

Lorsqu’un utilisateur essaie de se connecter au système il peut être :

* Connecté
* Déconnecté

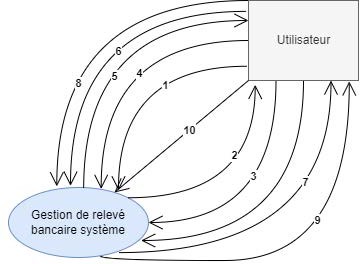


**Figure III.19 : Diagramme état transition « utilisateur »**

### Description des diagrammes de contexte

#### Définition de diagramme de contexte

Le diagramme de contexte sert à délimiter le contour du système en cours d'étude, nous définissons ici clairement ses frontières et l'acteur avec lesquels il communique. Ces derniers peuvent être soit des acteurs humains, soit d'autres systèmes avec lesquels le futur système communique.



**Figure III.20 : Diagramme de contexte**

#### Légende des messages

|  |  |  |  |
| --- | --- | --- | --- |
| **N° du message** | **L’émetteur** | **Le récepteur** | **Désignation** |
| **1** | Utilisateur | Système | Chargement de relevé bancaire. |
| **2** | Système | Utilisateur | Des confirmations sur le chargement |
| **3** | Utilisateur | Système | * Création des workflows. * Clôturer des processus métiers complété. * Suivre et contrôler les workflows. |
| **4** | Utilisateur | Système | Qualification des opérations crédit et débit. |
| **5** | Système | Utilisateur | * L’affichage de relevé bancaire. * L’affichage des lignes de relevé bancaire. * L’affichage des workflows. * L’affichage la qualification automatique. |
| **6** | Utilisateur | Système | Qualifier les produits manuellement. |
| **7** | Système | Utilisateur | Des confirmations sur la qualification des produits. |
| **8** | Utilisateur | Système | Qualifier les acteurs manuellement. |
| **9** | Système | Utilisateur | Des confirmations sur la qualification des acteurs. |
| **10** | Utilisateur | Système | Enregistrement de relevé bancaire |

**Tableau III.1: légende des messages**

### Description des diagrammes de classes

#### Définition de diagramme de classes

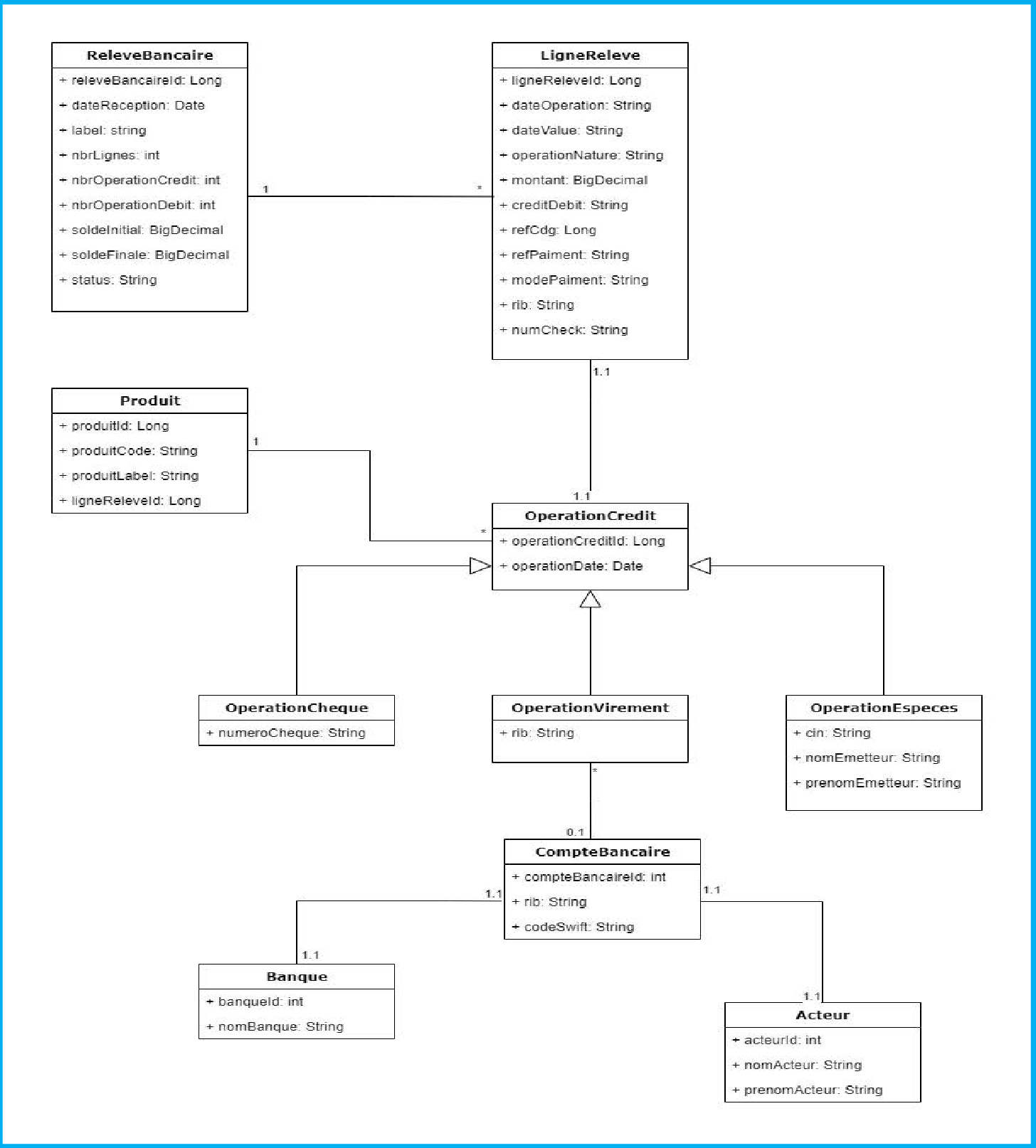
Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d’une telle modélisation. Le diagramme de classes montre la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d’utilisation. Il s’agit d’une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l’utilisation. Une classe-association possède les caractéristiques des associations et des classes : elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations. Une classe-association est caractérisée par un trait discontinu entre la classe et l’association.

Une classe est une description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations ou méthodes) et de relations avec d'autres objets (les associations et les agrégations).

Une classe de conception est composée par :

* **Attribut** chaque attribut d’une classe est le même pour chaque instance de cette classe.
* **Méthodes** elle définit le comportement d’une classe elle-même, et non le comportement de ses instances qui peut être différent. [6]

#### Le diagramme de classes





**Figure III.21 : Diagramme de classe**

### Description détaillée des attributs

#### Les attributs des classes d'objets

|  |  |  |
| --- | --- | --- |
| Classes | Attributs | Désignation |
| ReleveBancaire | * releveBancaireId * dateReception * label * nbrLignes * nbrOperationCredit * nbrOperationDebit * soldeInitial * soldeFinale | Identifiant de relevé bancaire.  La date de réception de relevé bancaire. Le nom de relevé bancaire.  Le nombre des lignes de relevé bancaire. Le nombre des opérations crédits.  Le nombre des opérations débits. Le solde initial de relevé bancaire.  Le solde final de relevé bancaire. |
| LigneReleve | * ligneReleveId * dateOperation * dateValue * operationNature * montant * creditDebit * refCdg * refPaiment * modePaiment * rib * numCheck | Identifiant de chaque ligne de relevé. La date d’opération d’un versement. La date valeur d’une opération.  La nature de versement.  Le montant d’un versement.  Identifiant d’opération (crédit ou débit). La référence de CDG.  La référence de paiement. Le mode de paiement.  Le RIB.  Le numéro de cheque. |
| OperationCredit | * operationCreditId * operationDate | Identifiant de l’opération crédit.  La date d’opération crédit. |
| OperationCheque | - numeroCheque | Le numéro de cheque. |
| OperationEspeces | * cin * nomEmetteur * prenomEmetteur | CIN.  Le nom d’émetteur.  Le prénom d’émetteur. |
| OperationVirement | - rib | RIB. |
| CompteBancaire | * compteBancaireId * rib * codeSwift | Identifiant de compte bancaire. RIB.  Le code Swift de compte bancaire. |
| Banque | * banqueId * nomBanque | Identifiant de banque.  Le nom de la banque. |
| Produit | * produitId * produitCode * produitLabel * ligneReleveId | Identifiant de produit. Le code de produit.  Le nom de produit.  Pour l’identification de chaque ligne de relevé. |
| Acteur | * acteurId * nomActeur * prenomActeur | Identifiant de l’acteur. Le nom de l’acteur.  Le prénom de l’acteur. |

**Tableau III.2 : Description détaillée des attributs des classes d'objets (Relevé Bancaire micro service).**

|  |  |  |
| --- | --- | --- |
| Classes | Attributs | Désignation |
| TaskDto | 1. Id | 1. Identifiant généré par Hibernate. |
|  | 2. releveBancaireId | 2. L’id de relevé bancaire. |
|  | 3. variableName | 3. Le nom de la variable. |
|  | 4. variableTypeName | 4. Le type de la variable. |
|  | 5. proccessesName | 5. Le nom de processus. |
|  | 6. taskId | 6. L’id de la tâche. |
|  | 7. assignee | 7. Le nom de l’assigné. |
|  | 8. name | 8. Le nom de la tâche. |
|  | 9. description | 9. Description de la tâche. |
|  | 10. prioprity | 10. Le nombre de la priorité. |
|  | 11. processDefinitionId | 11. Le nom unique du processus. |
|  | 12. processInstannceId; | 12. L’id de l’instance du processus. |
|  | 13. createTime | 13. La date de création de la tâche. |

#### Tableau III.3 : Description détaillée des attributs des classes d'objets (Activi Workfow microservice).

#### Les attributs des classes d'associations

|  |  |  |
| --- | --- | --- |
| Classes | Attributs | Attributs |
| ReleveBancaire | - ligneReleve | Liste des lignes de relevés bancaires. |
| LigneReleve | - releveBancaire | Identifiant du relevé bancaire auquel  appartiennent ces lignes. |
| OperationCredit | - ligneReleve | Identifiant du ligne de relevé. |
| CompteBancaire | * acteur * banque | Le compte bancaire d’acteur.  La banque d’acteur. |
| Banque | * banqueId * nomBanque | Identifiant de banque.  Le nom de la banque. |
| OperationEspeces | - hériter  d’opérationCredit | OperationCredit peut être en espèces. |
| OperationVirement | - hériter  d’opérationCredit | OperationCredit peut être en virement. |
| OperationCheque | - hériter  d’opérationCredit | OperationCredit peut être en chèque. |

**Tableau III.4: Description détaillée des attributs des classes d'associations (Relevé Bancaire micro service).**

### Les méthodes des classes d'objets et des classes d'associations

|  |  |  |
| --- | --- | --- |
| Classes | Méthodes | Descritpion |
| ReleveBancaire | 1. addReleveBancaire() 2. qualificationrelevebancaire() 3. removeReleveBancaire() 4. getReleveBancaires() 5. getReleveBancaireById() 6. releveBancaireStatus() 7. releveBancaireStatusQualifier() 8. deleteReleveBancaireById() | 1. Permet de créer un relevé bancaire 2. Permet de qualifier les lignes de relevé bancaire (virement, espèces ou chèque). 3. Permet de supprimer un relevé bancaire. 4. Permet d’obtenir tous les relevés bancaires. 5. Permet d’obtenir un relevé bancaire spécifique par son id. 6. Permet d'appliquer le statut “rejeter” au relevé bancaire. 7. Permet l'appliquer le statut “qualifier” au relevé bancaire. 8. Permet de supprimer un relevé bancaire spécifique par son id. |
| Acteur | 1. createacteur() 2. getActerus() 3. searchActeurs() 4. mockActeur() 5. getmockActeur() 6. getSearchedMockActeurs() | 1. Permet de créer un acteur. 2. Permet d’obtenir tous les acteurs. 3. Permet de chercher l’acteur. 4. Permet de simuler les données de l’acteur. 5. Permet d’obtenir tous les acteurs simulés 6. Permet d’obtenir un acteur spécifique par les mots recherché |
| Produit | 1. getProduits() 2. createProduit() 3. mockProduit() 4. getMockProduit() | 1. Permet d’obtenir tous les produits. 2. Permet de créer un produit. 3. Permet de simuler les données de produit. |

###### Tableau III.5: Description des méthodes des classes d'objets et d'association (Relevé Bancaire micro service).

|  |  |  |
| --- | --- | --- |
| Classes | Méthodes | Description |
| TaskDto | 1. startTheProcess() 2. processInformation() 3. getTasks() 4. getTaskById() 5. completeTask() | 1. Permet de démarrer le processus. 2. Permet d’obtenir les détails de processus. 3. Permet d’obtenir les informations sur la tâche. 4. Permet d’obtenir une tâche spécifique par son id. 5. Permet de terminer la tâche. |

**Tableau III.6: Description des méthodes des classes d'objets et d'association** **(Activiti Workflow micro service).**

### Conclusion

Dans ce chapitre, nous avons présenté le langage UML. Aussi nous avons fait la description des diagrammes des cas d’utilisation, de classe et de séquence et d’activité, et de contexte afin de délimiter le cadre de notre travail et de préparer un terrain favorable pour la prochaine étape.

Maintenant, notre application est prête à être codée. Dans le chapitre suivant, nous allons nous intéresser à l’implémentation de notre système en se basant sur la conception détaillée de ce chapitre.

# Chapitre IV : Réalisation

### Introduction

Dans chapitre précédent nous avons présenté les étapes de conception de l’application, ainsi que les différents diagrammes des cas d’utilisation, par la suite nous avons détaillé cette phase en établissant les diagrammes des classes ainsi que les diagrammes de séquences et d’activités.

Dans ce chapitre nous traitons les différentes étapes d’implémentation de l’application, et au fur et à mesure nous avons établi un ensemble de tests : ce sont les étapes de la phase de réalisation.

Nous allons commencer par la description de l’environnement matériel et logiciel tout en donnant par la suite un aperçu sur le travail accompli au cours de la période de développement.

### Environnement du travail

Dans cette partie, nous avons étudié le choix des outils matériels et surtout les outils logiciels du développement web.

#### Environnement matériel

Nous mentionnons les caractéristiques de l’ordinateur sur lesquelles nous avons développé l’application parce qu’elles peuvent donner une idée sur les conditions du travail.

L’application a été développée sur un ordinateur portable Acer Aspire qui se caractérise par :

* + - **Processus** : AMD Ryzen 3 3200U with Radeon Vega Mobile Gfx 2.60 GHz.
    - **Mémoire installé (RAM)** : 88.00 GB (5.93 GB usable).
    - **Type de système** : système d’exploitation 64 bits.
    - **Disque dur** : 250 Go SSD.
    - **Ecran :** 15,6 pouces.

#### Environnement logiciel

Nous avons énuméré au cours de cette partie les différents outils utilisés tout au long de ce projet pour l’étude et la mise en place de notre application.

###### Système d’exploitation

Nous avons utilisé comme système d’exploitation :

* Microsoft Windows 10 Pro.
* 21H2

###### Outil de modélisation UML :

* + - * **Visual Paradigm et Draw.io :**

Nous avons exploité pour la modélisation UML de l’application les outils Entreprise Architect qui sont flexible, complet et puissant, conçu pour les plateformes Windows. Visual Paradigm est un outil de création de modèles dont le langage est UML.

###### Environnement de développement intégré

* + - * **IntelliJ IDEA 2021.3 :**

Nous avons choisi IntelliJ comme Environnement de Développement Intégré (EDI). IntelliJ IDEA également appelé « IntelliJ », « IDEA » ou « IDJ » est un environnement de développement intégré (en anglais Integrated Development Environment - IDE) destiné au développement de logiciels informatiques reposant sur la technologie Java. Il est développé par JetBrains (anciennement « IntelliJ ») et disponible en deux versions, l'une communautaire, open source, sous licence Apache 2 et l'autre propriétaire, protégée par une licence commerciale. Tous deux supportent les langages de programmation Java, Kotlin, Groovy et Scala.

###### Plateforme de gestion d’API : « Postman »

Postman est officiellement présentée comme une plateforme API pour la création et l’utilisation d’API. D’une manière générale, Postman est une plateforme qui permet de simplifier chaque étape du cycle de vie des API et de rationaliser la collaboration, afin de créer, plus facilement et plus rapidement, de meilleures API.

###### Pourquoi utiliser Postman ?

La plupart des utilisateurs de Postman recourent à cette plateforme pour la construction et la formulation de requêtes, afin de tester des API sans avoir à renseigner de code. Parmi les nombreux points forts de Postman, on relève :

1. La possibilité d’utiliser la plateforme, quel que soit le langage utilisé pour la programmation des API.
2. Une interface utilisateur assez simple et facile à prendre en main.
3. L’absence de compétences nécessaires en codage.

###### Conteneurisation : « Docker »

Docker est une plateforme de conteneurs lancée en 2013 ayant largement contribué à la démocratisation de la conteneurisation. Elle permet de créer facilement des conteneurs et des applications basées sur les conteneurs. Il en existe d’autres, mais celle-ci est la plus utilisée. Elle est par ailleurs plus facile à déployer et à utiliser que ses concurrentes.

C’est une solution open source, sécurisée et économique. De nombreux individus et entreprises contribuent au développement de ce projet. Un large écosystème de produits, services et ressources sont développés par cette vaste communauté.

###### Les avantages des conteneurs Docker:

1. Modularité
2. Couches et contrôle des versions d'image
3. Restauration
4. Déploiement rapide

###### Orchestration des conteneurs : « Docker Compose »

Docker Compose est un outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs conteneurs comme un ensemble de services inter- connectés.

###### Hébergement et de gestion de développement de logiciels : « GitHub »

GitHub est considéré comme un outil essentiel pour les ingénieurs logiciels, et sa popularité est inégalée. Il accueille actuellement plus de 25 millions d’utilisateurs. C’est un nombre considérable de professionnels qui se tournent vers GitHub pour améliorer le flux de travail et la collaboration.

En bref, GitHub est un service basé sur le cloud qui héberge un système de contrôle de version (VCS) appelé Git. Il permet aux développeurs de collaborer et d’apporter des modifications à des projets partagés tout en gardant un suivi détaillé de leur progression

###### L'automatisation des Workflows avec CI/CD : « GitHub Actions »

GitHub Actions est l'outil d'intégration continue et de livraison continue construit et utilisé par GitHub. Il vous permet de créer, tester et déployer votre code directement à partir de GitHub, en prenant en charge toute l'automatisation qui permet que cela se fasse en douceur sans aucun outil CI/CD tiers. Les possibilités que vous pouvez créer et automatiser à l'aide de cela sont infinies, et avec la facilité de travailler directement à partir de l'endroit où votre code est stocké, GitHub ne peut pas être égalé.

###### La documentation des APIs : « Swagger v.3 »

Swagger est vite devenu la technologie la plus appréciée pour la documentation API. Plus précisément, pour documenter les API REST, très souvent utilisées. Swagger a été développé par Reverb, mais il s’agit désormais d’une solution indépendante et open source, sous la gouvernance de la fondation Linux, et plus précisément de l’OpenAPI Initiative.

Avec ce changement d’acteurs, Swagger a été rebaptisé « spécification OpenAPI », même s’il reste officieusement connu sous le nom plus accrocheur de « Swagger ».

###### Gestion de la qualité du code : « SonarQube »

Développé par SonarSource, SonarQube est un logiciel open source de gestion de la qualité du code. Il est utilisé pour inspecter le code source des logiciels et applications en développement et détecter des bugs, vulnérabilités de sécurités, instances de code dupliqué et autres anomalies pouvant nuire à la qualité du code source, et ainsi au fonctionnement de l’application qui en résulte. L’objectif est d’aider les développeurs à créer un code d’une meilleure qualité, tout en simplifiant le processus de développement.

SonarQube se base sur l’analyse statique du code, ce qui lui permet d’étudier le code source avant même l’exécution de l’application. Par opposition, l’analyse dynamique requiert l’exécution de l’application pour en étudier le comportement et détecter les problèmes dont elle souffre.

###### Les rapports SonarQube:

Après réception et analyse des fichiers du code, SonarQube génère 7 types de rapports chacun couvrant un aspect de sa qualité :

1. Architecture et design de l’application.
2. Test unitaire (analyse de parties précises du code).
3. Codes dupliqués.
4. Bugs potentiels.
5. Code complexe.
6. Règles de programmation (les bonnes règles du langage en question).
7. Commentaires.

###### Gestion de la qualité du code sur cloud : « SonarCloud »

Il faut savoir que SonarQube est développé par la société SonarSource et que ce logiciel est Open-Source. SI vous utilisez donc directement SonarQube sachez donc que ce sera gratuit. SonarCloud est directement basé sur SonarQube. Ils sont sensiblement pareils, mais pourquoi alors utiliser SonarCloud. Pour la facilité d’intégration avec d’autres outils cloud. Cet outil est gratuit si vous le connectez à un projet Open Source. Cela va dire que si vous comptez l’utiliser pour un dépôt que vous avez mis en public sur GitHub alors vous ne paierez pas.

Si par contre vous souhaitez connecter un dépôt privé alors vous devrez passer à la caisse.

###### La sécurité de votre code:

SonarCloud va vous aider à rendre votre code plus sécurisé. Il vous offre 2 indicateurs. Le premier c’est la vulnérabilité. SonarCloud est capable d’identifier et de vous dire les parties votre code qui pourraient être exploités par des hackers (oui oui vous lisez bien).

Et le deuxième paramètre ce sont les Security Hotspots. Cet indicateur va relever toutes les parties de code sensible en termes de sécurité et qui nécessite un examen manuel pour que vous puissiez évaluer s’il existe vraiment une vulnérabilité ou non.

###### II.2.12 Outil d’administration de la base de données RDBMS et DBMS

* + **MySQL Workbench (Relational Database Management System) :**

Nous avons appliqué MySQL Workbench comme un logiciel de gestion et d'administration de bases de données MySQL, il possède une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.

###### MongoDB (Database Management System) :

Nous avons ajouté mongodb en tant que base de données secondaire pour nous aider à récupérer et à ajouter des données volumineuses et massives de manière rapide et en moins de temps par rapport à MySQL.

###### Langage de programmation

* **Java**

L’application a été développée en JAVA pour lequel nous avons opté car il est de plus en plus utilisé dans le monde de la recherche scientifique ainsi que dans l'industrie. En effet, ce langage de programmation présente un large avantage car les programmes peuvent être exécutés sur différents systèmes d'exploitation et architectures matérielles. De plus, il permet de créer facilement des interfaces graphiques conviviales.

###### Backend Framework : Spring Boot

Spring Boot est un Framework développé par Pivotal en 2012 qui connaît depuis environ 4 ans, une explosion du nombre d’utilisateurs. C’est un point d’entrée unique vers tous les projets de la IO Fondation (batch, ligne de commande, web..) que vous pourrez donc utiliser dans votre application.

Grâce à une configuration par défaut, une application Spring Boot permet de développer plus rapidement en se focalisant essentiellement sur le code métier. On dira que Spring Boot a une « opinion ». Bien entendu, ces fonctionnalités par défaut peuvent être modifiées pour correspondre aux besoins. Cela implique de respecter les conventions et règles imposées par Spring afin d’éviter de perdre trop de temps sur de la configuration applicative. Cela nous permet notamment d’éviter d’écrire de multiples fichiers de configuration en XML.

###### Frontend Framework : Angular 13

Angular est un Framework open source JavaScript développé par Google. Ce Framework est utilisé pour développer des applications web et mobile. Avec cette technologie, on réalise des interfaces de type monopage ou “one page” qui fonctionnent sans rechargement de la page web.

Angular propose un ensemble de conventions et d’outils pour délimiter les bases d’une solution. Les développements sont ainsi optimisés, s’effectuent plus rapidement et de manière plus sûre.

En tant que langage open source, il est régulièrement alimenté et enrichi par une communauté importante de contributeurs. Ces derniers aident à l’amélioration du Framework dans le temps grâce des à des mises à jour récurrentes.

###### Les avantages de développer avec ce framework:

1. Rapidité d’exécution conséquente.
2. Des applications responsives.
3. Des compétences provenant de Google.
4. Interfaces haut de gamme.
5. Les directives et la declarative building.
6. La facilité de prise en main des projets.
7. Une forte maintenabilité.

###### Environnement d'exécution

* + **Node.js 16.14.0**

Node.js est un environnement d’exécution single-thread, open-source et multi-plateforme permettant de créer des applications rapides et évolutives côté serveur et en réseau. Il fonctionne avec le moteur d’exécution JavaScript V8 et utilise une architecture d’E / S non bloquante et pilotée par les événements, ce qui le rend efficace et adapté aux applications en temps réel.

###### Gestionnaire de paquets

* + - **NPM 8.6.0**

npm est le gestionnaire de paquets officiel de Node.js. Sa maîtrise est obligatoire pour tout développeur voulant travailler dans cet environnement car il est présent à toutes les étapes de la création, du développement et de la maintenance des applications Node.js.



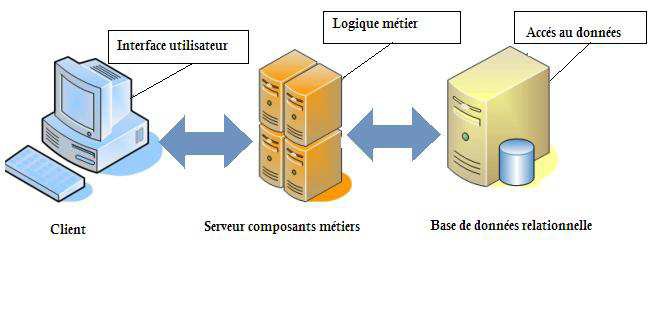
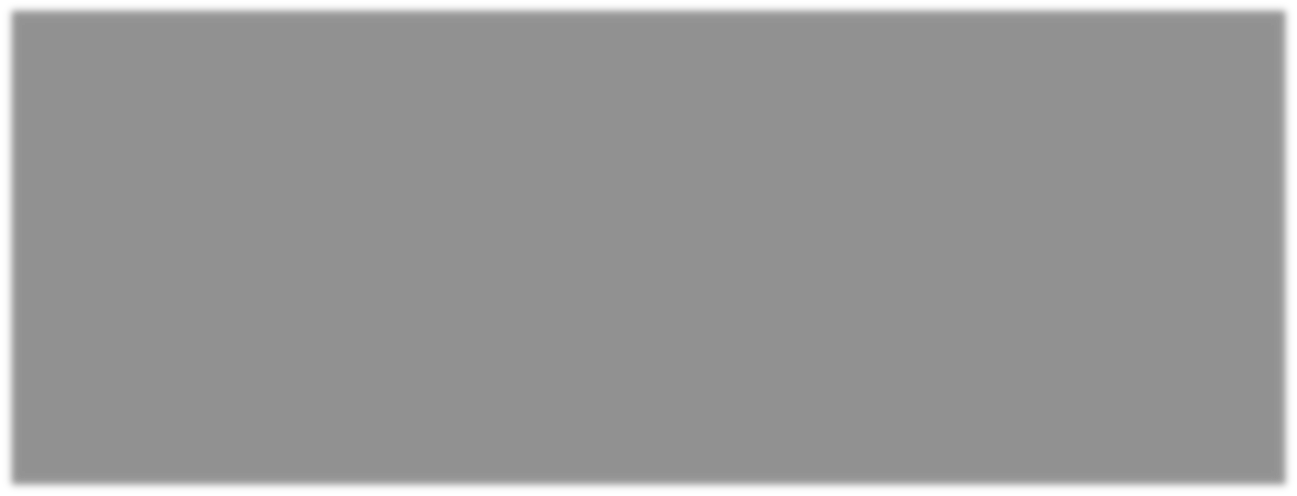
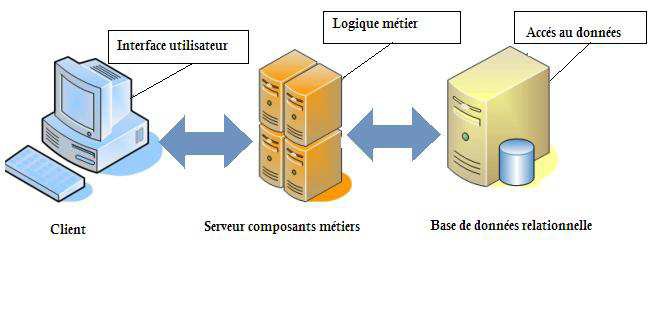
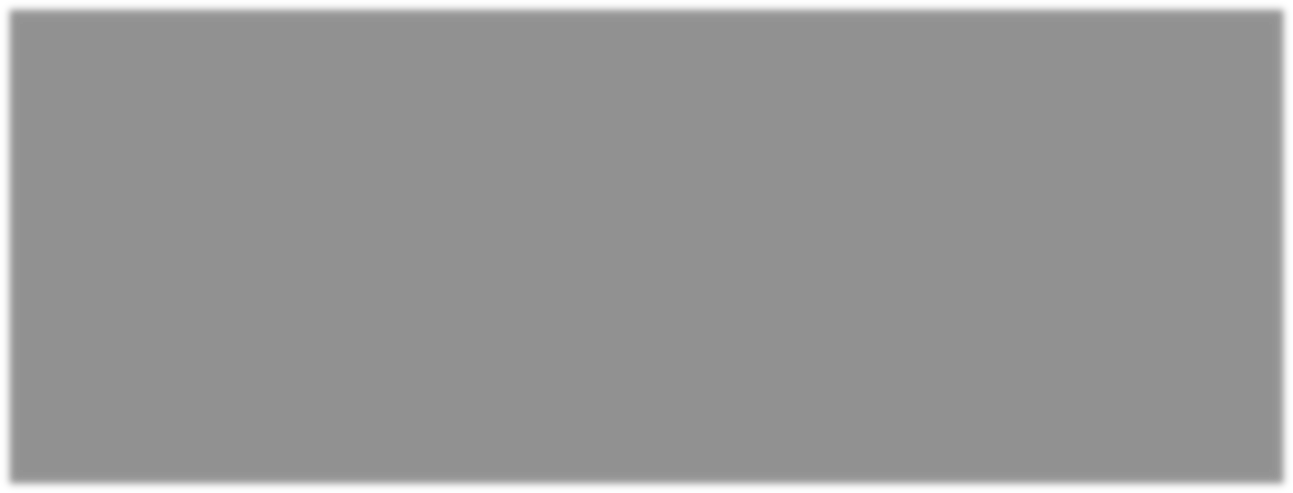
**Figure IV.1 : Les technologies utilisé**

### Les architectures de l’application

Nous présentons les architectures de l’application, puis nous détaillerons les différentes infrastructures qui le composent.

#### Architecture 3-tiers

**Nous avons développé l’application en nous basant sur l’architecture trois-tiers. En** adoptant cette architecture, le système sera divisé en trois couches (ou niveaux) différents : présentation des données, logique métier, et accès aux données.



**Figure IV.2 : Architecture 3 - tiers**

#### Microservices

L'architecture de microservices (ou microservices) désigne un style d'architecture utilisé dans le développement d'applications. Elle permet de décomposer une application volumineuse en composants indépendants, chaque élément ayant ses propres responsabilités Nous avons utilisé cette architecture pour de nombreux avantages dont certains sont :

* Les microservices sont des modules de déploiement autonomes et indépendants.
* Le coût de la mise à l'échelle est comparativement inférieur à celui de l'architecture monolithique.
* Il est possible de modifier ou de mettre à niveau chaque service individuellement plutôt que de mettre à niveau l'ensemble de l'application.
* Les microservices suivent le principe de responsabilité unique.
* Moins de dépendance et facile à tester.

Nous avons créé 6 microservices :

###### Pour notre frontend :

* Angular + NGINX.

###### Pour notre backend :

* Spring Cloud API Gateway microservice.
* Activiti Workflow microservice.
* Releve Bancaire microservice (Hexagonal architecture) + MySQL
* Netflix Eureka Server microservice.
* Relevé Bancaire microservice + MongoDB.

#### Architecture Hexagonal

L'architecture hexagonale, ou architecture à base de ports et d'adaptateurs, est un patron d'architecture utilisé dans le domaine de la conception des logiciels. Elle vise à créer des systèmes à base de composants d'application qui sont faiblement couplés et qui peuvent être facilement connectés à leur environnement logiciel au moyen de ports et d'adaptateurs. Ces composants sont modulaires et interchangeables ce qui renforce la cohérence des traitements et facilite l'automatisation des tests.

### La couche de la securité

##### Okta (authentification, autorisation)

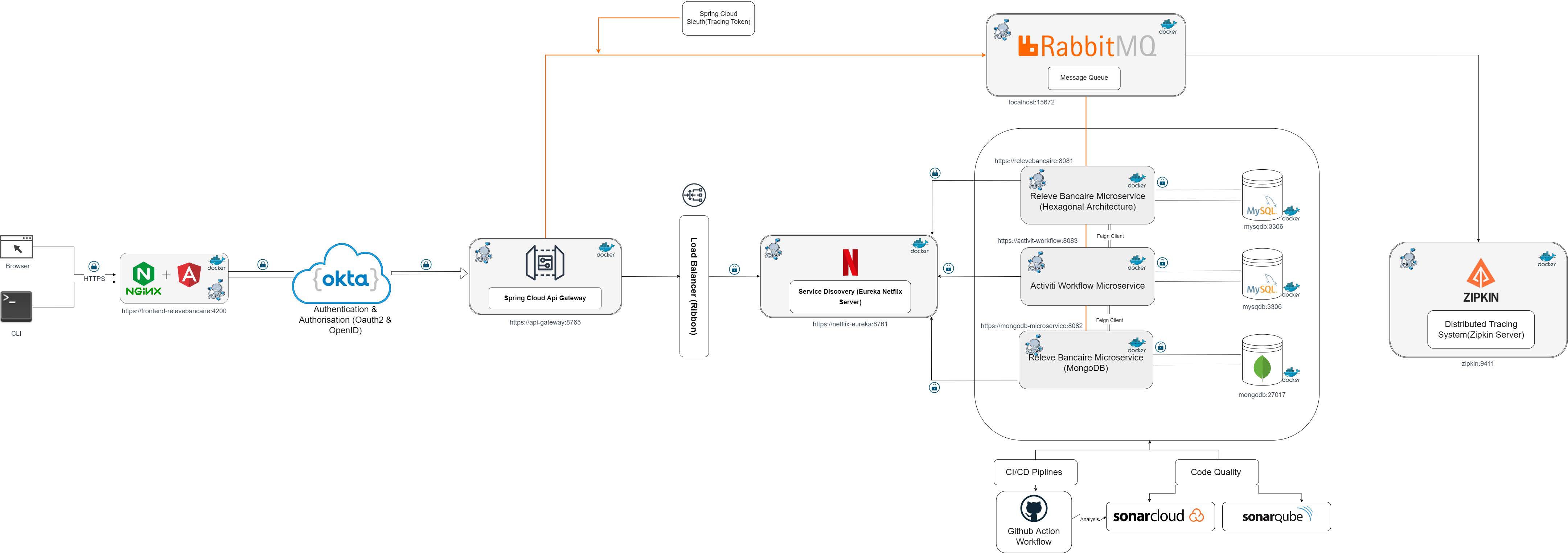
Okta, Inc. (anciennement Saasure Inc.) est une société américaine de gestion des identités et des accès basés à San Francisco. Il fournit un logiciel cloud qui aide les entreprises à gérer et à sécuriser l'authentification des utilisateurs dans les applications, et aux développeurs à intégrer des contrôles d'identité dans les applications, les services Web de sites Web et les appareils.

##### Le protocole de transmission (HTTPS)

Le protocole HTTPS (Hyper Text Transfer Protocol Secure) est une extension sécurisée du protocole HTTP, le « S » pour « Secured » (sécurisé) signifie que les données échangées entre le navigateur de l’internaute et le site web sont chiffrées et ne peuvent en aucun cas être espionnées (confidentialité) ou modifiées (intégrité).

### Les technologies utilisée - Plus détaillées

Cette image montre le tout et toutes les technologies utilisées dans notre application :



**Figure IV.3: Présentation de l'application**

|  |  |  |
| --- | --- | --- |
| Microservices | URL | Ports |
| Relevé Bancaire microservice (Hexagonal Architecture) + MySQL | Relevé Bancaire microservice (Hexagonal Architecture) + MySQL  URL | https://relevebancaire:8081/api/v1 |
| Relevé Bancaire microservice + MongoDB | Relevé Bancaire microservice + MongoDB  URL | https://mongodbmicroservice:8082/api/v2 |
| Activiti Workflow  microservice | Activiti Workflow  microservice URL | https://activiti-workflow:8083/api/v3 |
| Netflix Eureka Service Registry microservice | Netflix Eureka Service Registry microservice  URL | https://netflix-eureka:8761/ |
| Spring Cloud API Gateway microservice | Spring Cloud API Gateway microservice  URL | https://api-gateway:8765/ |
| Angular + NGINX  microservice | Angular + NGINX  microservice URL | https://localhost:4200/ |
| Zipkin Distributed Tracing System &  RabbitMQ | Zipkin Distributed Tracing System &  RabbitMQ URL | Zipkin: localhost:9411 RabbitMQ: 15672 |

**Tableau IV.1. Liste des microservices avec url et ports**

### Les microsevices

#### V.1.1 Angular + NGINX microservice

On va commencer par notre premier microservice c'est le microservice du Frontend:

* Nom du microservice : frontend-relevebancaire
* Fonctionne sur le port : 4200
* URL : https://localhost:4200/
* Nom du conteneur Docker : frontend-relevebancaire
* Nom de l'image Docker : frontend-relevebancaire
* Protocole de communication : HTTPS

Ce microservice a la version Angular 13.1.1 avec node.js version 16.14.0 et npm version 8.6.0 avec le serveur nginx exécuté sur le port 80. Ce microservice est sur un conteneur docker et géré par docker compose. Dans le projet, nous avons implémenté deux composants importants, le premier appelé upload qui gère le téléchargement des relevés bancaires et le second inbox a 4 sous-composants le composant inboxmanagement : gère tout a une relation avec la banque de relevés et ses linges.

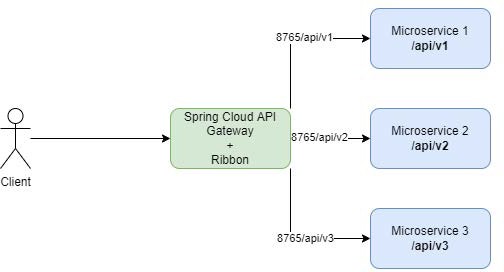
Le composant taskdetails gèrent tout a une relation avec Activiti Workflow. Le composant qualifieracteur gère la qualification des acteurs.

et le dernier composant qualifiersw : gère la qualification des produits.

#### V.1.2 Spring Cloud API Gateway microservice + Load Balancer Ribbon

* + Nom du microservice : api-gateway
  + Fonctionne sur le port : 8765
  + URL : https://api-gatewat:8765/
  + Nom du conteneur Docker : api-gateway
  + Nom de l'image Docker : api-gateway
  + Protocole de communication : HTTPS

Le modèle de passerelle est utilisé pour fournir une interface unique à un ensemble de microservices différents. Si nous disposons de plusieurs microservices fournissant des données pour votre API, nous ne souhaitons pas les exposer à notre client. Bien mieux pour eux d'avoir un seul point d'entrée, sans avoir à penser à quel service interroger quelles données. C'est aussi bien de pouvoir centraliser les traitements courants tels que l'authentification. Comme tout modèle de conception. Ainsi que la redirection à chaque microservice est fait par le loadbalancer ribbon qui va lire le chemin apartir le client et redirecté api a le microservice concerné. Tout ça va faire sur le port 8765. Pour notre projet, j'ai implémenté chaque microservice avec une information supplémentaire pour permettre au Ribbon de connaître le chemin correct, notre microservice qui a une architecture hexagonale est connu pour le Ribbon sous le nom de /api/v1 et notre microservice + mongodb a le chemin /api/v2 et le microservice Activiti Workflow a le chemin /api/v3 afin que le Ribbon connaisse chaque demande au endpoint spécifique.

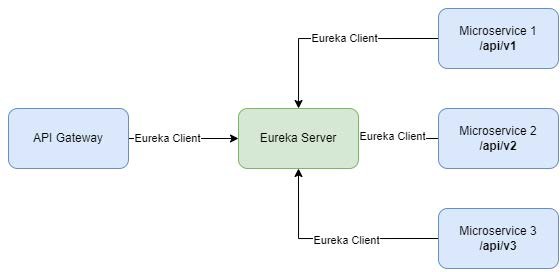


**Figure IV.4: Mécanisme de Spring Cloud API Gateway**

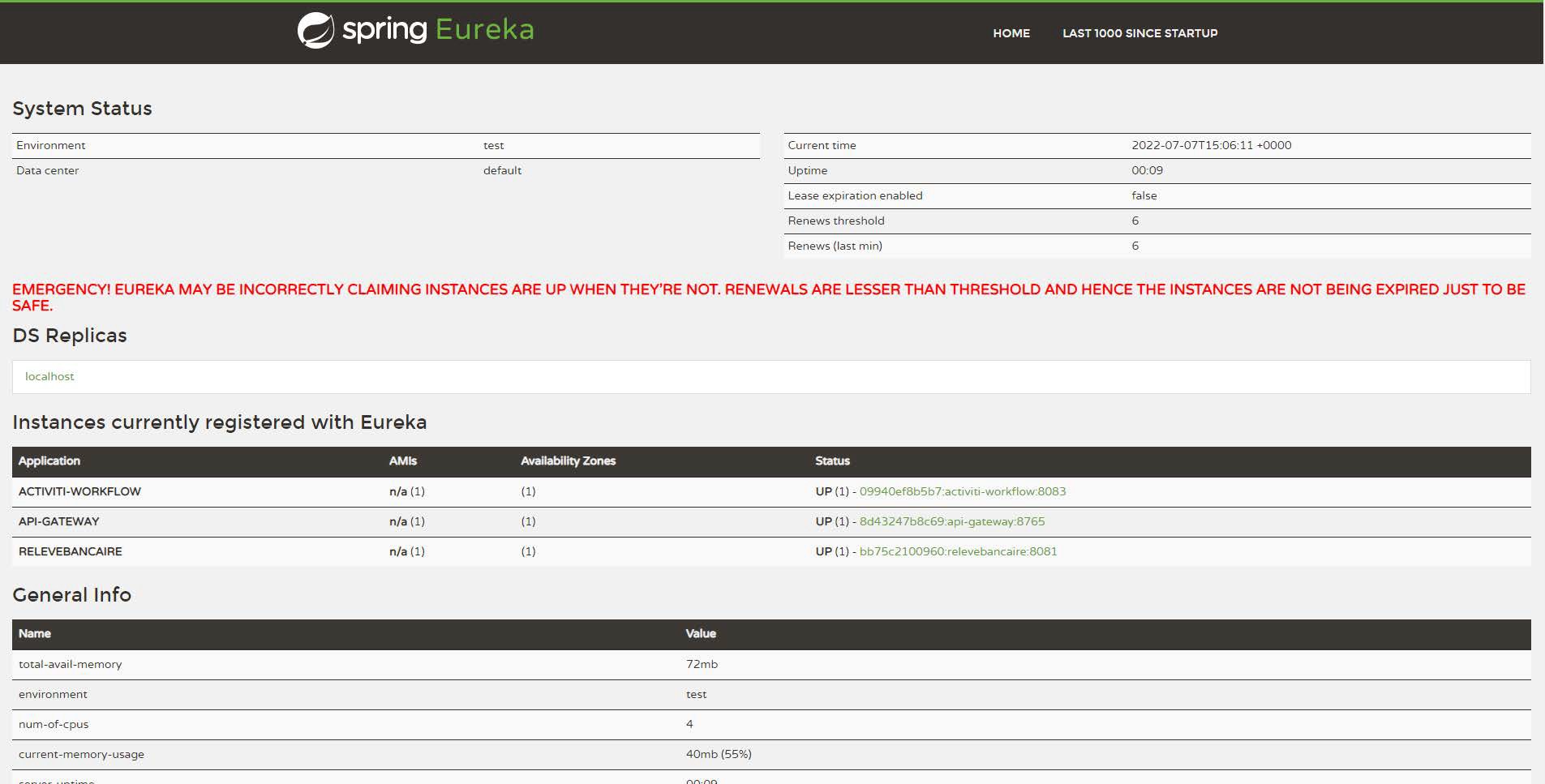
#### V.1.3 Netflix Eureka Service Registration and Discovery

* + Nom du microservice : netflix-eureka
  + Fonctionne sur le port : 8761
  + URL : https://netflix-eureka:8761/
  + Nom du conteneur Docker : netflix-eureka
  + Nom de l'image Docker : netflix-eureka
  + Protocole de communication : HTTPS

Eureka est une application permettant la localisation d’instances de services. Elle se caractérise par une partie serveur et une partie cliente. La communication entre les parties se fait via les API Web exposées par le composant serveur. Ces services doivent être créés en tant que clients Eureka, ayant pour objectif de se connecter et s’enregistrer sur un serveur Eureka. De ce fait, les clients vont pouvoir s’enregistrer auprès du serveur et périodiquement donner des signes de vie. Le service Eureka (composant serveur) va pouvoir conserver les informations de localisation desdits clients afin de les mettre à disposition aux autres services (service registry).



**Figure IV.5: Mécanisme de Eureka Service Discovery**



**Figure IV.6: Interface de Eureka Service Discovery**

#### V.1.4 Relvé Bancaires microservice (Hexagonal Architecture)

* + Nom du microservice : relevebancaire
  + Fonctionne sur le port : 8081
  + URL : https://relevebancaire:8081/
  + Nom du conteneur Docker : relevebancaire
  + Nom de l'image Docker : relevebancaire
  + Protocole de communication : HTTPS

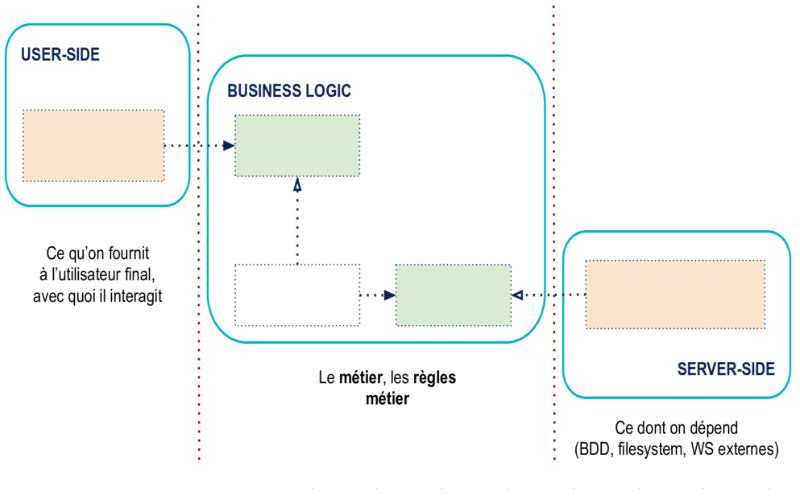
Sur ce microservice, il n'a pas d'architectures en couches traditionnelles, où l'application est divisée en couche de présentation, couche logique et couche de données.

#### V.1.4.1 Introduction à l'Architecture Hexagonale

L'architecture hexagonale, également appelée ports et adaptateurs, est un modèle architectural qui permet aux entrées des utilisateurs ou des systèmes externes d'arriver dans l’application à un port via un adaptateur, et permet d'envoyer la sortie de l'application via un port vers un adaptateur. Cela crée une couche d’abstraction qui protège le cœur d'une application et l'isole des outils et technologies externes - et en quelque sorte non pertinents.

#### Principe : Séparer User-Side, Business Logic et Server-Side

Le premier principe est de séparer explicitement le code en trois grandes zones formalisées.





**Figure IV.7: Les zones de l'architecture**

#### À gauche, la zone User-Side

C’est le côté par lequel **l’utilisateur** ou les programmes extérieurs vont **interagir avec l’application**. On y trouve le code qui permet ces interactions. Typiquement, votre code d’interface utilisateur, vos routes HTTP pour une API, vos sérialisations en JSON à destination de programmes qui consomment votre application sont ici.

C’est le côté où l’on retrouve les acteurs qui **pilotent** la Business Logic.

#### Au centre, la Business Logic

C’est la partie que l’on veut isoler de ce qui est à gauche et à droite. On y trouve tout le code qui concerne et implémente la **logique métier.** Le vocabulaire métier et la logique purement métier, ce qui se rapporte au problème concret que résout votre application, tout ce qui en fait la richesse et la spécificité est au centre. Dans l’idéal, un expert du métier qui ne sait pas coder pourrait lire un bout de code de cette partie et vous pointer une incohérence (true story, ce sont des choses qui pourraient vous arriver !).

#### À droite, la zone Server-Side

C’est ici qu’on va retrouver ce dont votre application a besoin, ce qu’elle pilote pour fonctionner. On y trouve les détails d’infrastructure essentiels comme le code qui interagit avec votre base de données, les appels au système de fichier, ou le code qui gère des appels HTTP à d’autres applications dont vous dépendez par exemple.

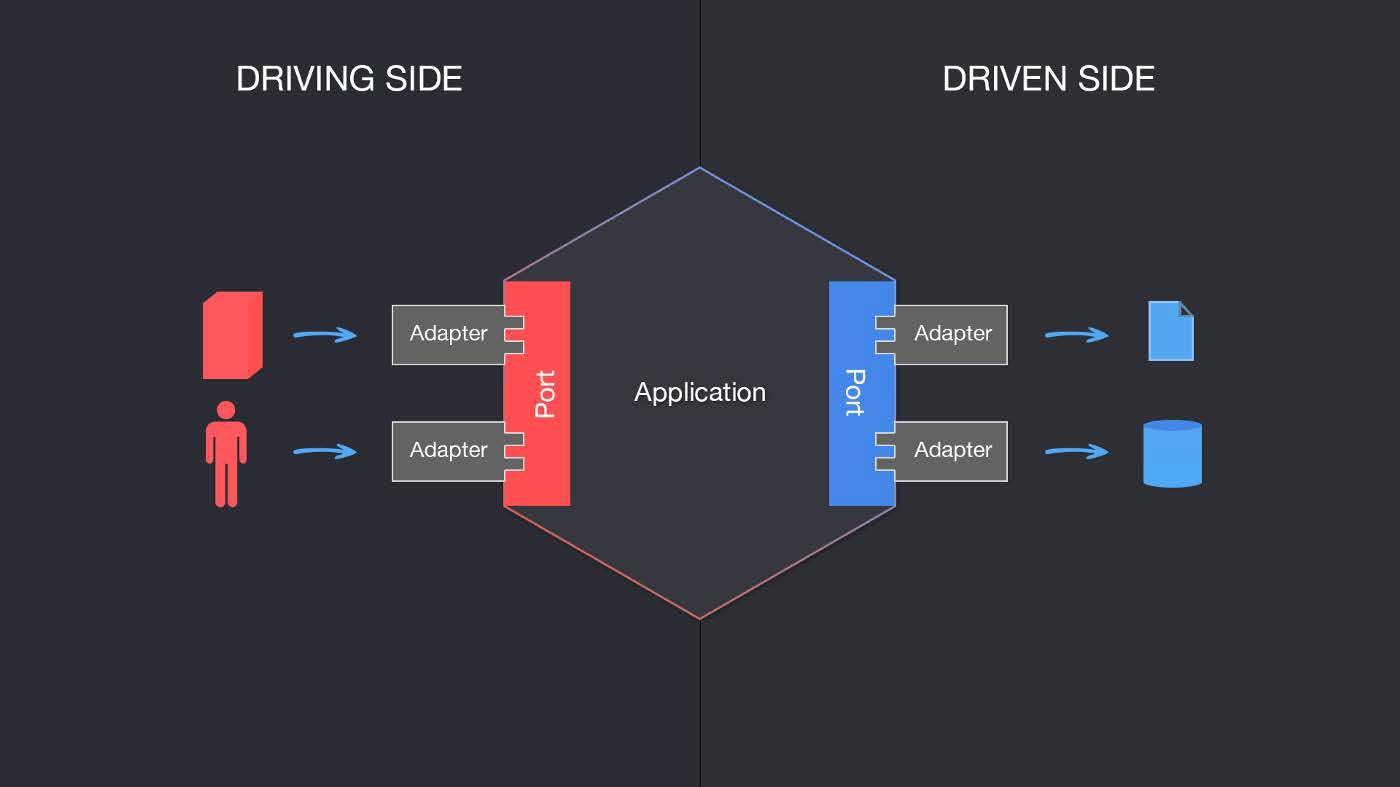
C’est le côté où l’on retrouve les acteurs qui sont **pilotés** par la zone Business Logic.

#### Ports

On peut voir un Port comme un point d'entrée technologiquement indépendant, il détermine l’interface qui permettra aux acteurs étrangers de communiquer avec l'Application, peu importe qui ou quoi implémentera ladite interface. Tout comme un port USB permet à plusieurs types d'appareils de communiquer avec un ordinateur tant qu'ils disposent d'un adaptateur USB. Les ports permettent également à l'application de communiquer avec des systèmes ou services externes, tels que des bases de données, des courtiers de messages, d'autres applications, etc.

#### Adaptateurs

Un adaptateur initiera l'interaction avec l'application via un port, en utilisant une technologie spécifique, par exemple, un contrôleur REST représenterait un adaptateur qui permet à un client de communiquer avec l'application. Il peut y avoir autant d'adaptateurs pour un seul port que nécessaire sans que cela représente un risque pour les ports ou l'application elle-même.



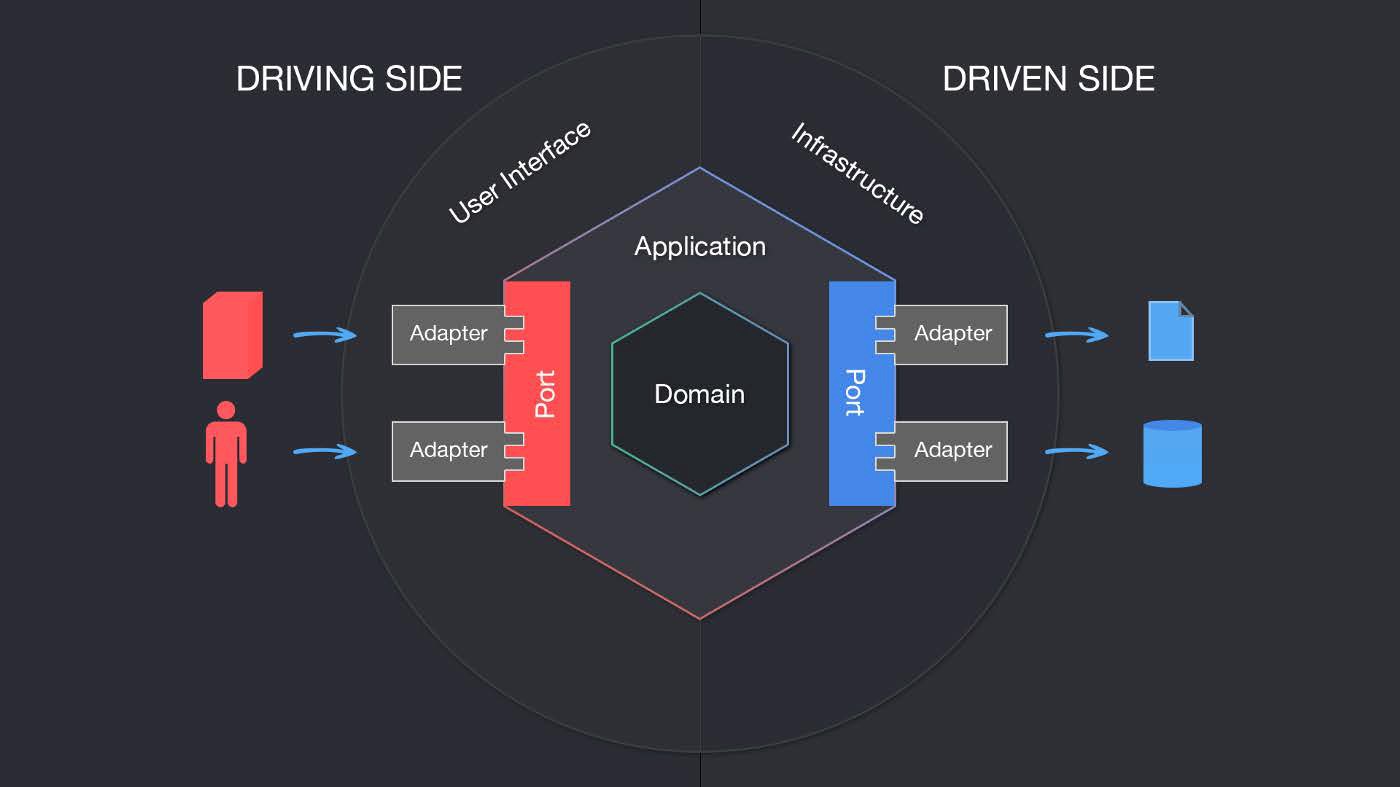


**Figure IV.8: Driving Side vs Driven Side**

#### Application

L'Application est le cœur du système, elle contient les Services applicatifs qui orchestrent les fonctionnalités ou les cas d'utilisation. Il contient également le modèle de domaine, qui est la logique métier intégrée dans les agrégats, les entités et les objets de valeur. L'application est représentée par un hexagone qui reçoit des commandes ou des requêtes des ports et envoie également des requêtes à d'autres acteurs externes, comme des bases de données, via des ports.

Lorsqu'elle est associée à la conception pilotée par le domaine, l'application, ou Hexagone, contient à la fois les couches d'application et de domaine, laissant les couches d'interface utilisateur et d'infrastructure à l'extérieur.





**Figure IV.9: Les composants internes de l'architecture**

#### Pourquoi c’est important ?

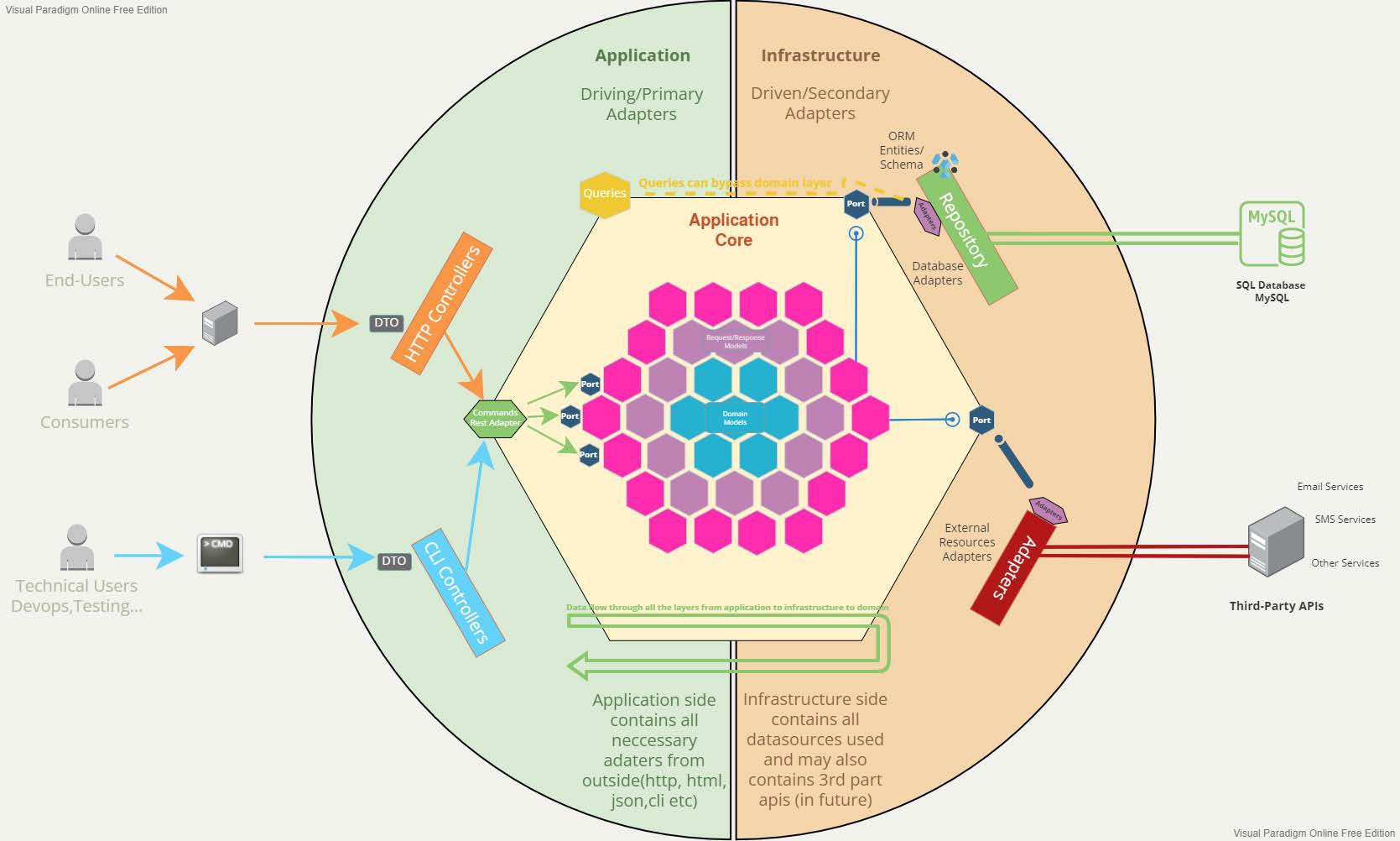
Une première caractéristique importante de cette séparation est qu’elle **sépare les problèmes**. À tout moment, on peut choisir de se concentrer sur une seule logique, presque indépendamment des deux autres : la logique applicative, la logique métier, ou la logique infrastructure. On les comprend plus facilement sans les mélanger, et les contraintes de chaque logique a moins d’impact sur les autres.

Une autre caractéristique est qu’**on met la logique métier en avant dans notre code**. On peut l’isoler dans un répertoire ou un module pour la rendre explicite pour tous les développeurs. On peut la définir, la raffiner et la tester sans embarquer la charge cognitive du reste du programme. C’est important car, au final, c’est la compréhension du métier par les développeurs qui part en production.

Et, pour finir, en termes de tests automatisés (comme on va le voir plus bas), on va réussir à tester avec un effort raisonnable :

Toute la Business Logic unitairement, L’intégration entre User-Side et Business Logic – sans le Server-Side L’intégration entre Business Logic et Server-Side – sans l’User-Side

#### V.1.4.2 L'architecture Hexagonale sur notre projet





**Figure IV.10: L'aperçu de l'hexagone sur le projet**

* **Conclusion**

L'architecture hexagonale ou ports et adaptateurs n’est pas la solution miracle pour toutes les applications. Cela implique un certain niveau de complexité qui, lorsqu'il est manipulé avec soin, apportera de grands avantages à votre système. Mais si les fenêtres brisées sont autorisées, cela pourrait causer beaucoup de maux de tête.

Lorsqu'ils sont correctement mis en œuvre et associés à d'autres méthodologies, telles que la conception pilotée par le domaine, les ports et les adaptateurs peuvent assurer la stabilité et l’extensibilité à long terme d'une application, apportant une grande valeur au système et à l'entreprise.

#### V.1.5 Activiti Workflow microservice

* + Nom du microservice : activiti-workflow
  + Fonctionne sur le port : 8083
  + URL : https://activiti-workflow:8083/
  + Nom du conteneur Docker : activiti-workflow
  + Nom de l'image Docker : activiti-workflow
  + Protocole de communication : HTTPS

Ce microservice est responsable de tous les flux de travail utilisant la solution Activiti, chaque fichier bancaire de relevé aura une gestion de processus, de cette façon nous aurons un contrôle total, une gestion rapide, fiable et flexible sur notre gestion de fichiers.

#### V.1.5.1 Introduction au Activiti Workflow

Activiti Activiti est une plateforme opensource Java de BPM, qui permet de concevoir, de déployer et de mettre en œuvre des workflows (ou procédures d’entreprise). Activiti, développé par les fondateurs de jBPM, est conforme au standard BPMN 2.0. Activiti s’adresse aux différents acteurs des processus de votre entreprise

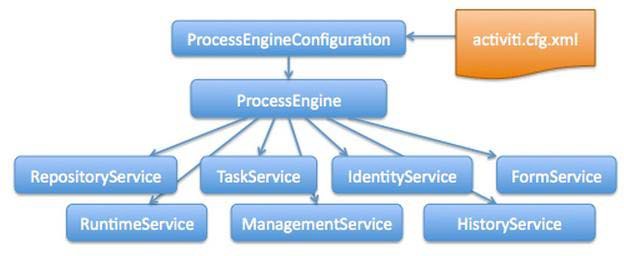
#### BPM

Le BPM (Business Process Management) désigne la gestion des processus d’entreprise. On pourra également utiliser le terme Workflow. Il consiste à identifier et modéliser des procédures d’entreprise (entrées, sorties, acteurs, etc. …), puis à les déployer dans un système de gestion des processus (BPMS). Exemple : le cycle de vie (rédaction – approbation – publication) d’un document

#### BPMN (Business Process Model And Notation)

BPMN (Business Process Model And Notation) - Les modèles et symboles de processus métier sont un ensemble de symboles de modélisation de processus métier standard développés par BPMI (Business Process Management Initiative). Les processus métier peuvent être créés à l'aide de symboles fournis par BPMN. La spécification BPMN 1.0 a été publiée en mai 2004. BPMI a été intégré à OMG (The Object Management Group Object Management Organization) en septembre 2005. OMG a publié la version finale de BPMN 2.0 en janvier 2011.

#### V.1.5.2 Diagramme d'architecture de service d'Activiti



**Figure IV.11: Diagramme d'architecture de service d'Activiti**

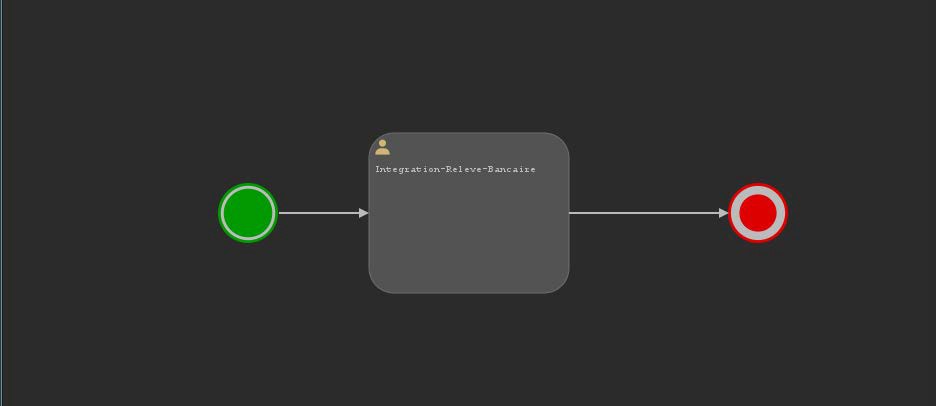
* **ProcessEngineConfiguration**

La classe de configuration du moteur de processus peut créer le moteur de flux de travail ProcessEngine via Process Engine Configuration.

#### ProcessEngine

Le moteur de workflow, qui équivaut à une interface de façade, crée un moteur de processus via Process Engine Configuration et des interfaces de service via Process Engine.

#### V.1.5.3 Mise en place d'Activiti sur le projet



**Figure IV.12: Mise en place d'Activiti sur le projet**

Sur notre exemple Nous avons implémenté une seule tâche que nous utiliserons pour télécharger le fichier et créer la tâche et effectuer les opérations nécessaires et la terminer

#### Comment cela fonctionnera-t-il dans notre projet ?

Notre objectif est d'abord de télécharger et de qualifier automatiquement et manuellement notre relevé bancaire, puis de l'enregistrer dans la base de données. À partir de ce moment, nous avons besoin d'une tâche qui gère cette opération, nous avons créé notre tâche appelée Integration-Releve- Bancaire et chaque tâche a besoin d'un assignée, nous lui avons donné un nom Gestionnaire-Intégration, nous devons donc communiquer avec l'autre microservice qui a tout ce dont nous avons besoin nous ferons une communication via un **Feign Client** qui parle du chemin du microservice relevebancaire à l'aide de **Ribbon** et **Eureka Netflix**, nous récupérons cette déclaration bancaire qui a téléchargé et récupérons son identifiant et faisons la qualification à travers elle.

#### V.1.6 Sécurisation de l'application

Pour rendre notre application plus sécurisée et flexible j'ai doublé la couche de sécurité

#### V.1.6.1 Okta - Authentification et autorisation

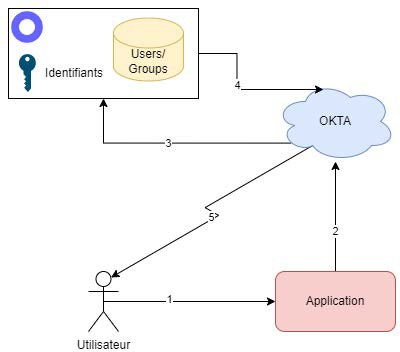
Okta est un service de gestion d'identité qui nous permet d'accéder à n'importe quel employeur à n'importe quelle application sur n'importe quel appareil. Il est hébergé sur un serveur sécurisé. Il utilise les technologies cloud pour aider les entreprises à gérer et à sécuriser l'authentification des utilisateurs dans les applications. Il vend divers services, dont l'authentification unique, qui est l'un des meilleurs programmes car il a la particularité unique de permettre aux utilisateurs de se connecter à de nombreuses applications via un seul processus centralisé. Il est sûr à utiliser et protège nos données sensibles même lorsque d'autres utilisent nos téléphones, car il ne faut que quelques minutes à la technologie développée pour voler nos informations personnelles.

#### Caractéristiques d'Okta:

1. Authentification unique
2. Authentification multifactorielle (MFA)
3. Gestion du cycle de vie
4. Annuaire Universel
5. Gestion des accès API
6. Accès avancé au serveur

#### Pourquoi nous utilisons OKTA pour l'authentification

Okta est basé sur OpenID Connect (OIDC), qui repose sur le protocole OAuth 2.0. Okta aide à créer, modifier et stocker en toute sécurité les données d'informations utilisateur à l'aide de l'API. Vous pouvez valider et connecter une ou plusieurs applications en les utilisant. Il permet de gérer facilement et en toute sécurité les utilisateurs et de sécuriser les données.



**Figure IV.13: Okta Implémentation avec Angular**

#### Implémenter l'authentification Okta dans Angular

L’implémentation d'okta dans notre application avec angular prend quelques mesures, nous en discuterons

Pour implémenter okta, nous devons suivre les étapes suivantes :

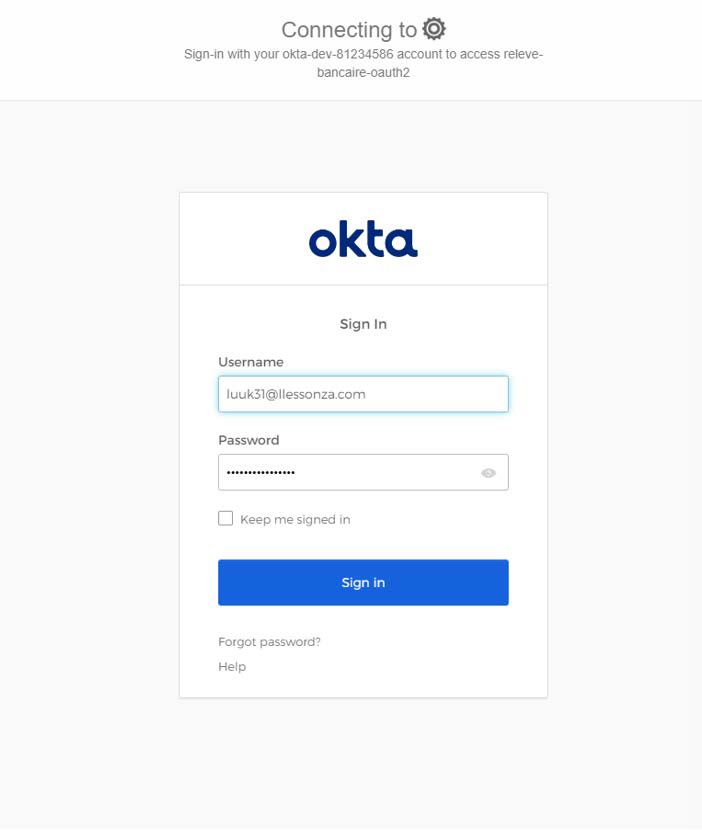
* Créer un compte busniness dans developer.okta.com.
* Ajoutez un client OpenID Connect dans Okta.

Pour intégrer okta à l'application angulaire, nous devons installer 2 dépendances :

* **okta-angular** cela indiquera à angular que nous utilisons okta

Comme service d'authentification.

* **okta-signin-widget** qui nous donnera l'interface okta pour se connecter. Généré par Okta
* Enfin, créez des routes pour la redirection vers la connexion et la déconnexion.



**Figure IV.14: Page de connexion Okta avec application**

#### V.1.6.2 Okta - Sécuriser la communication avec HTTPS

L’abréviation HTTPS signifie « Hypertext Transfer Protocol Secure », qui se traduit par « Protocole hypertexte de transmission sécurisé ». Le protocole de transmission est en quelque sorte le langage par lequel le client - en règle générale le navigateur - et le serveur Web se comprennent. Le HTTPS est la version du protocole de transmission qui opère en transmission chiffrée.

Le HTTPS remplit deux fonctions : La communication entre le client Web et le serveur Web est chiffrée, Le serveur Web est authentifié par le fait qu’en tout début de communication.

#### Comment migrer vers HTTPS (Self-Signed Certificate)

Tant que nous travaillons avec des conteneurs docker et utilisons java comme langage de programmation, nous avons besoin de 3 choses pour migrer de http à https :

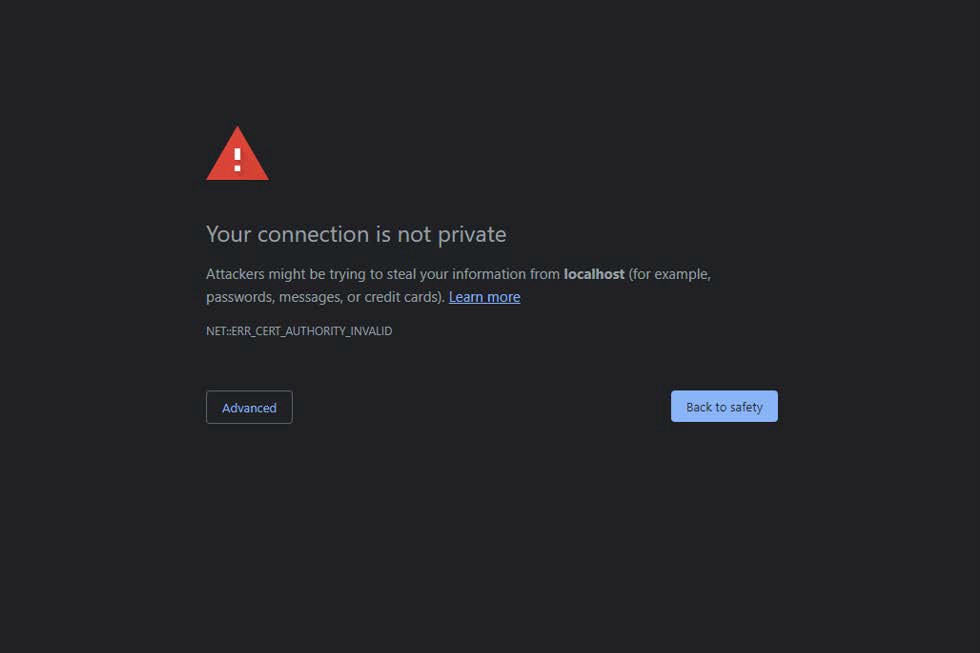
1. Générer le fichier .**crt** avec la commande **keytool -genkeypair**.
2. Copiez le fichier **.crt** dans **$JAVA\_HOME/lib/security.**
3. Et enfin importer et valider notre certificat. Avec la commande

#### keytool -importcert.

Nous l'avons fait dans chaque microservice backend qui utilise java

et pour le frontend, nous avons généré directement le fichier cert dans dockerfile dans le dossier **/etc/ssh**.

Lorsque nous terminons toutes les étapes et essayons d'accéder à notre application, cette image devrait apparaître.

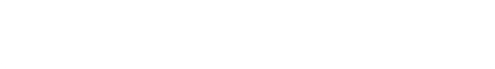
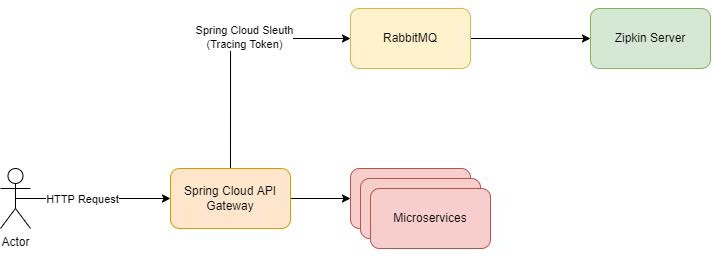


**Figure IV.15: L'implémentation de Self-Signed Certificat**

#### V.1.7 Système de traçage

Dans les grands systèmes de micro - services distribués, Un système est divisé en Plusieurs modules, Ces modules sont responsables de différentes fonctions, Ensemble dans un système, En fin de compte, il y a beaucoup de fonctionnalités disponibles, Les appels entre services sont complexes Pour pouvoir localiser rapidement les problèmes dans une architecture distribuée, Le traçage des liens distribués est la meilleure solution pour connaitre les problèmes.

Dans cette situation, nous utiliserons rabbitmq comme message broker et le producteur est spring cloud sleuth et le consommateur est le serveur zipkin.



**Figure IV.16: Système de traçage**

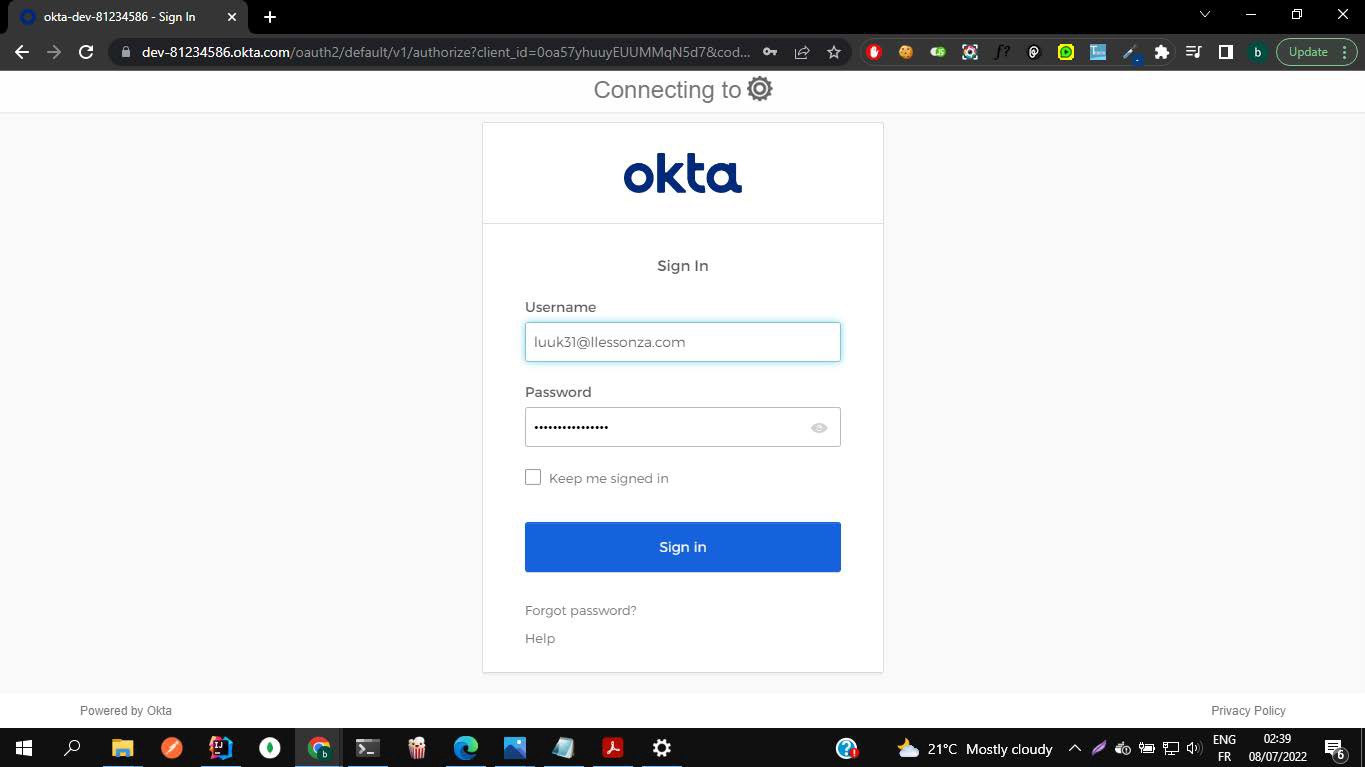
### Les interfaces graphiques

L’interface graphique est une partie très importante pour la réalisation d’une application Web convenable et conviviale offrant un certain plaisir à l’utilisateur lors de sa navigation. Ainsi ce critère peut faire la différence entre une application et un autre bien qu’elles aient les mêmes fonctionnalités.

Voici maintenant un ensemble de captures d’écrans sur les principaux points d’entrées de l’application :

#### Interface Authentification

Lorsque vous entrez dans l'application pour la première fois ou après la déconnexion, entrez à nouveau de l’interface home, une interface d’authentification est affichée. L’utilisateur doit introduire son Login et son mot de passe dans les champs correspondants pour pouvoir accéder aux différentes fonctionnalités de l’application.

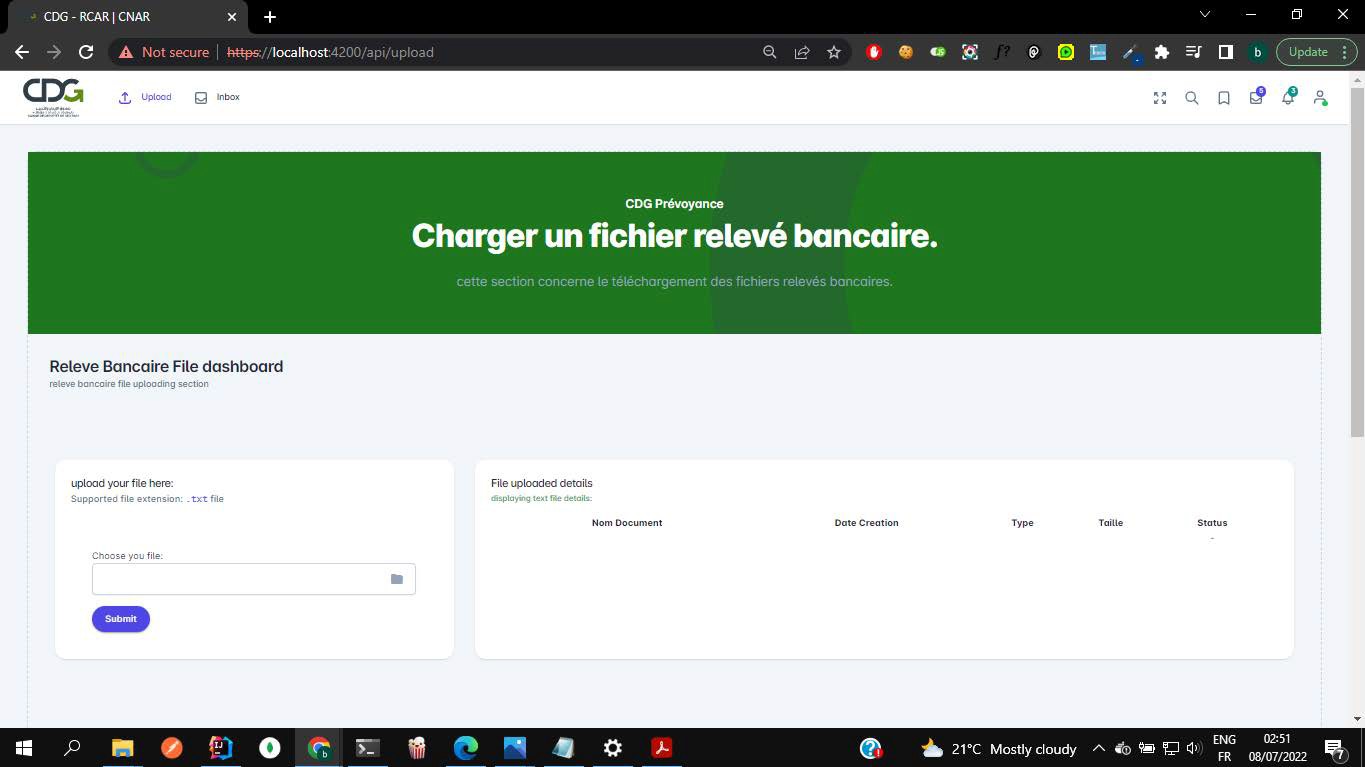


**Figure IV.17: Interface Authentification**

Une fois que le client a cliqué sur le bouton « Sign in », le système vérifie les données entrées.

En cas d’échec, il réaffiche la page d’authentification avec un message d’erreur. Si le Login et le mot de passe sont valides, le système passe au menu principal.

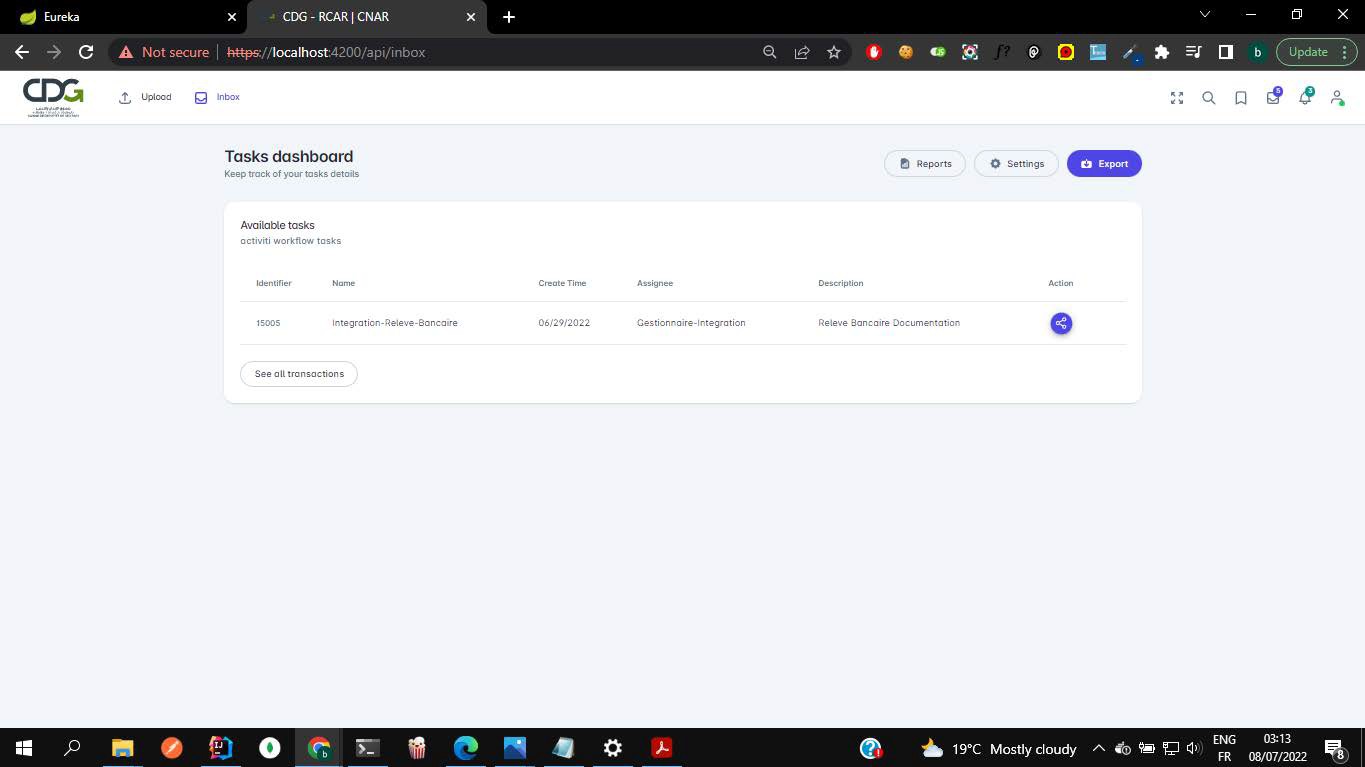
#### Interface Chargement de Relevé Bancaire



**Figure IV.18: Interface Chargement de Relevé Bancaire**

Cette interface permet à l'utilisateur de charger les relevés bancaires.

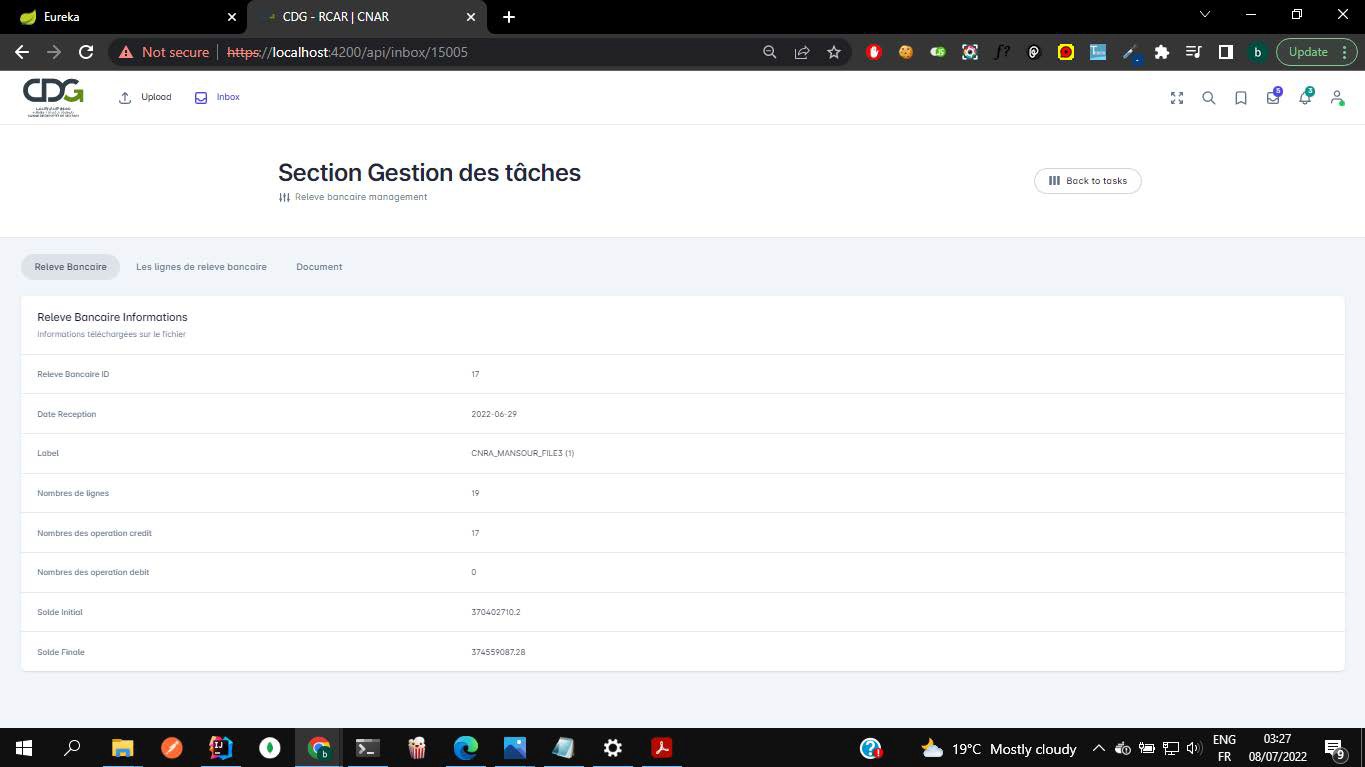
#### Interface des Tâches de workflow Activiti



**Figure IV.19: Interface des Tâches de workflow Activiti**

Cette interface affiche les tâches non terminées du Activiti.

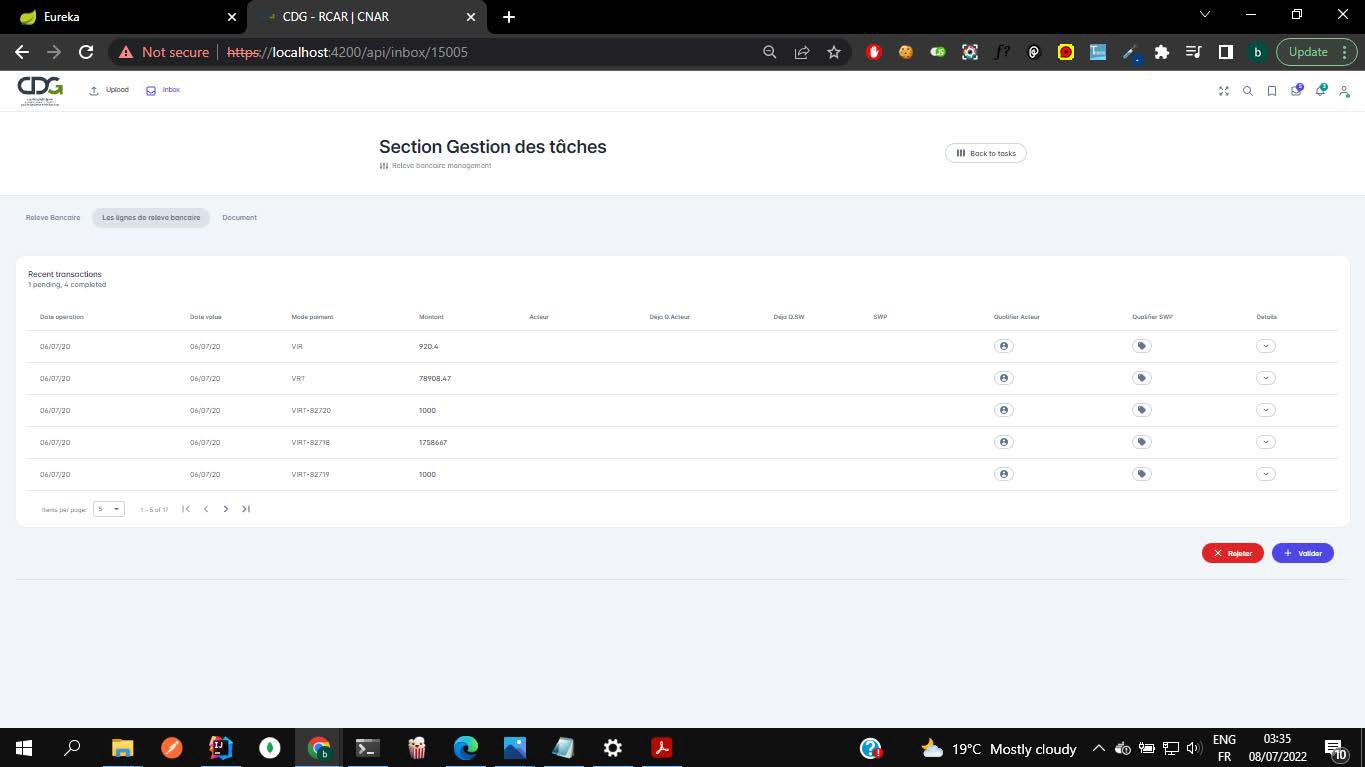
#### Interface des informations des relevés bancaires



**Figure IV.20: Interface des informations des relevés bancaires**

Cette interface affiche les détails sur le relevé bancaire.

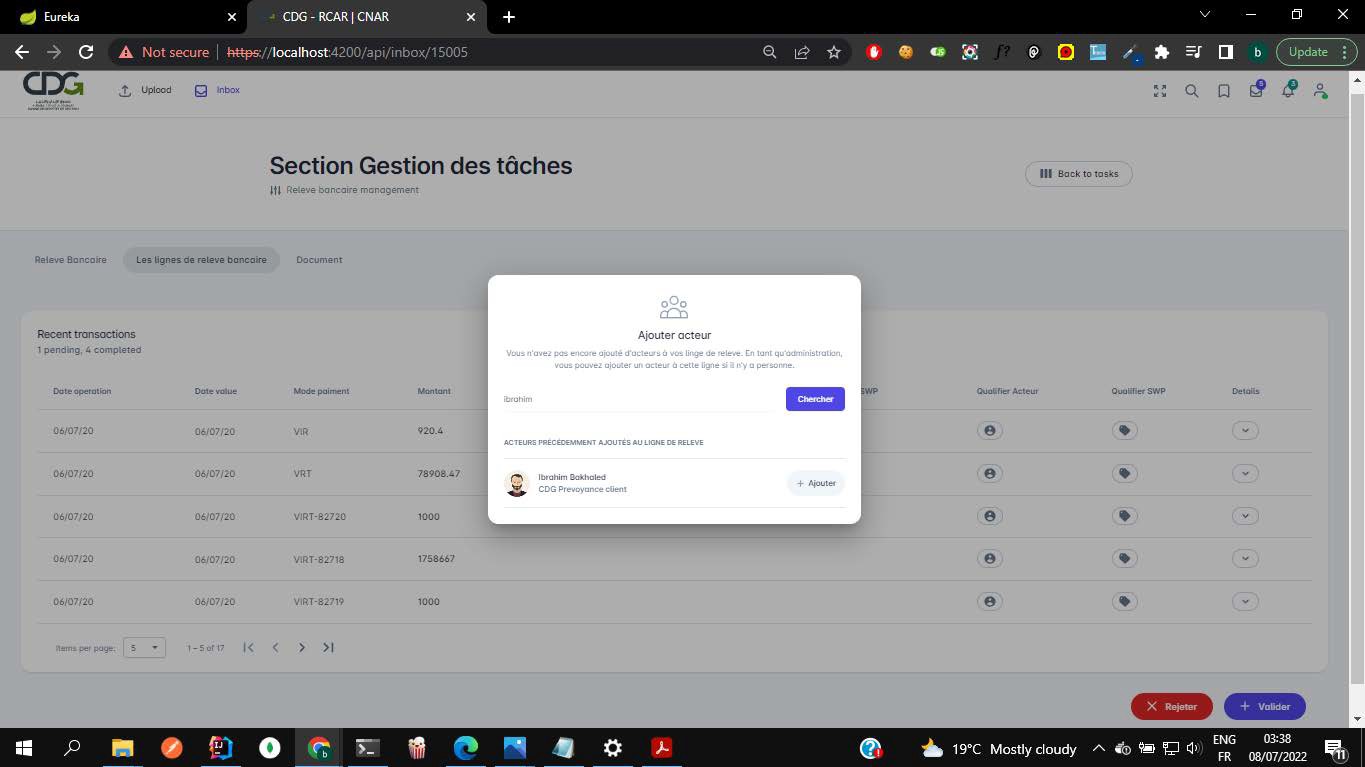
#### Interface des informations les lignes de relevé bancaire



**Figure IV.21: Interface des informations les lignes de relevé bancaire**

Cette interface affiche les détails sur les lignes de relevé bancaire

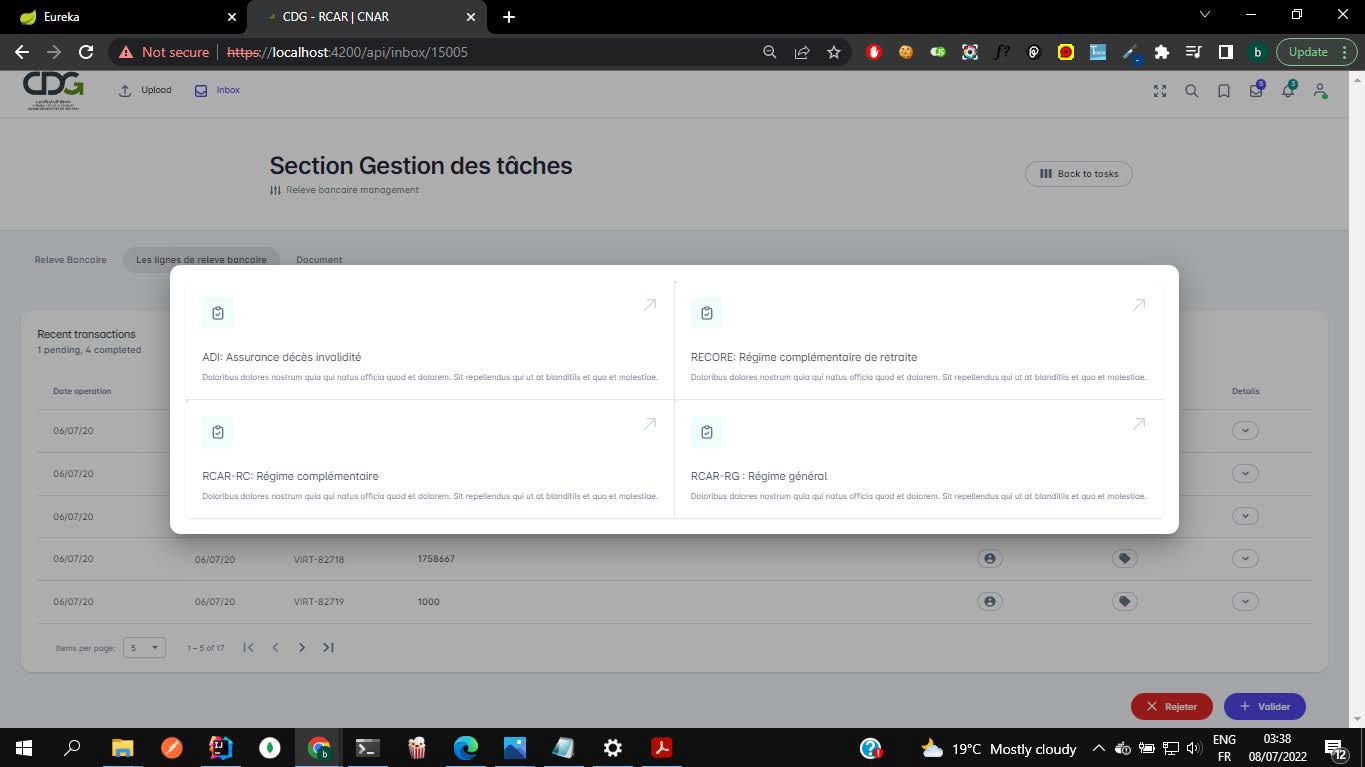
#### Interface de la qualification manuelle de l'acteur



**Figure IV.22: Interface de la qualification manuelle de l'acteur**

Cette interface affiche comment qualifier l'acteur d'une manière manuelle.

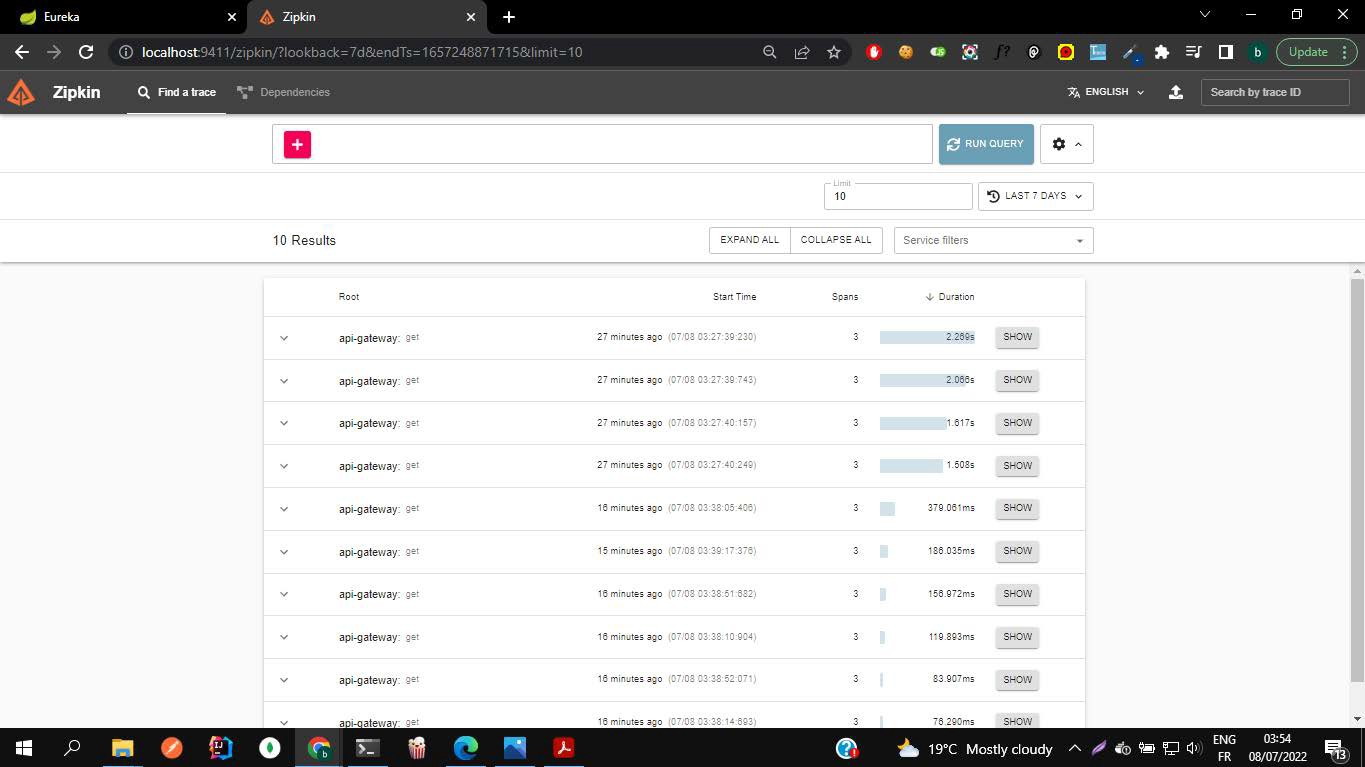
#### Interface de la qualification manuelle de produite



**Figure IV.23: Interface de la qualification manuelle de produite**

Cette interface affiche comment qualifier le produit d'une manière manuelle.

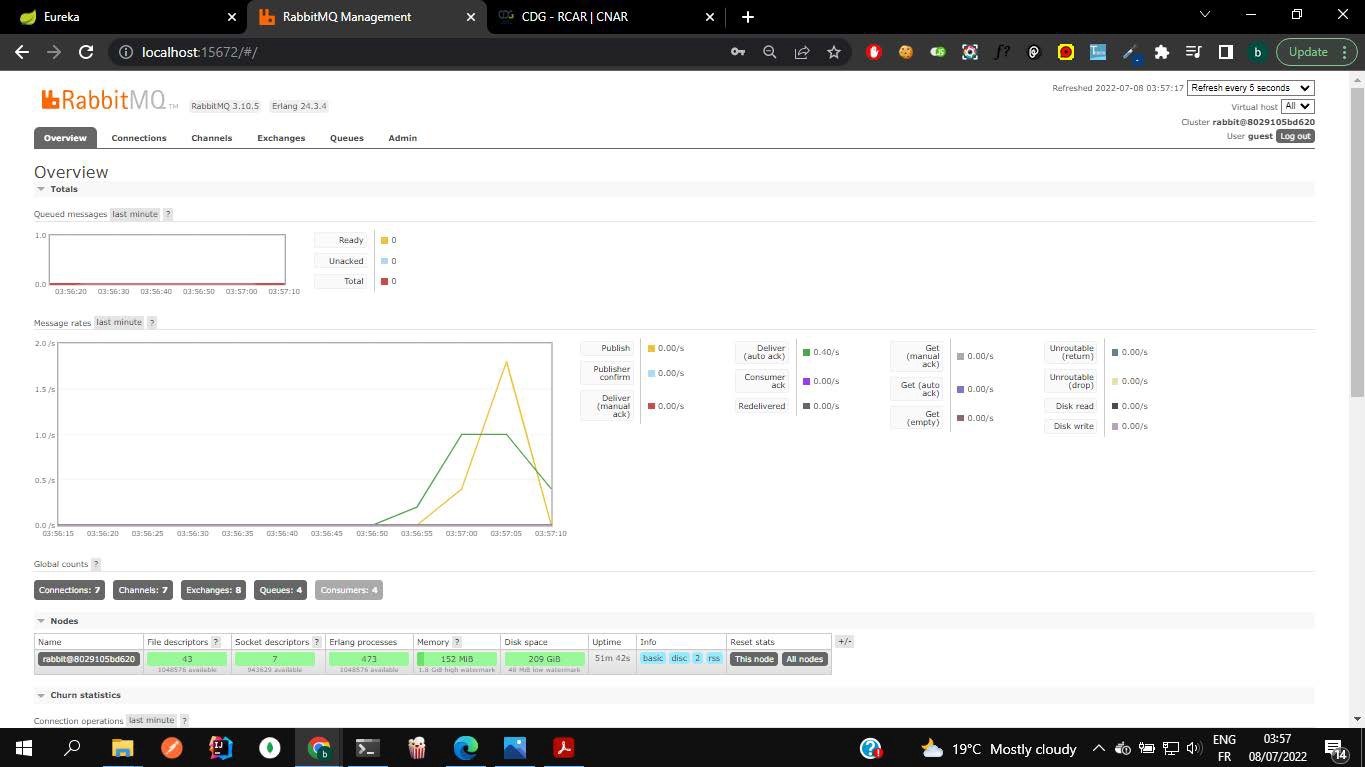
#### Interface de système de traçage du serveur Zipkin



**Figure IV.24: Interface de système de traçage du serveur Zipkin**

Cette interface affiche toutes les requêtes http qui ont été envoyées à chaque microservice

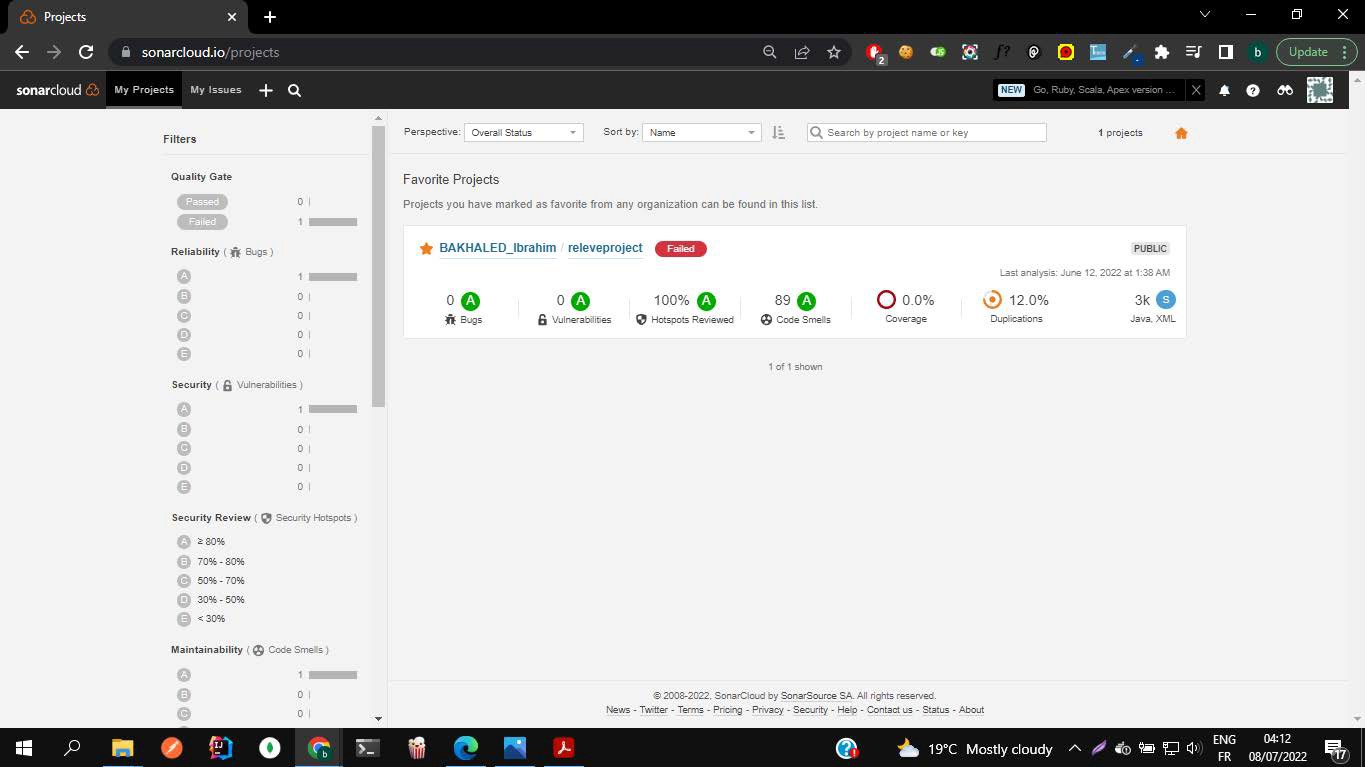
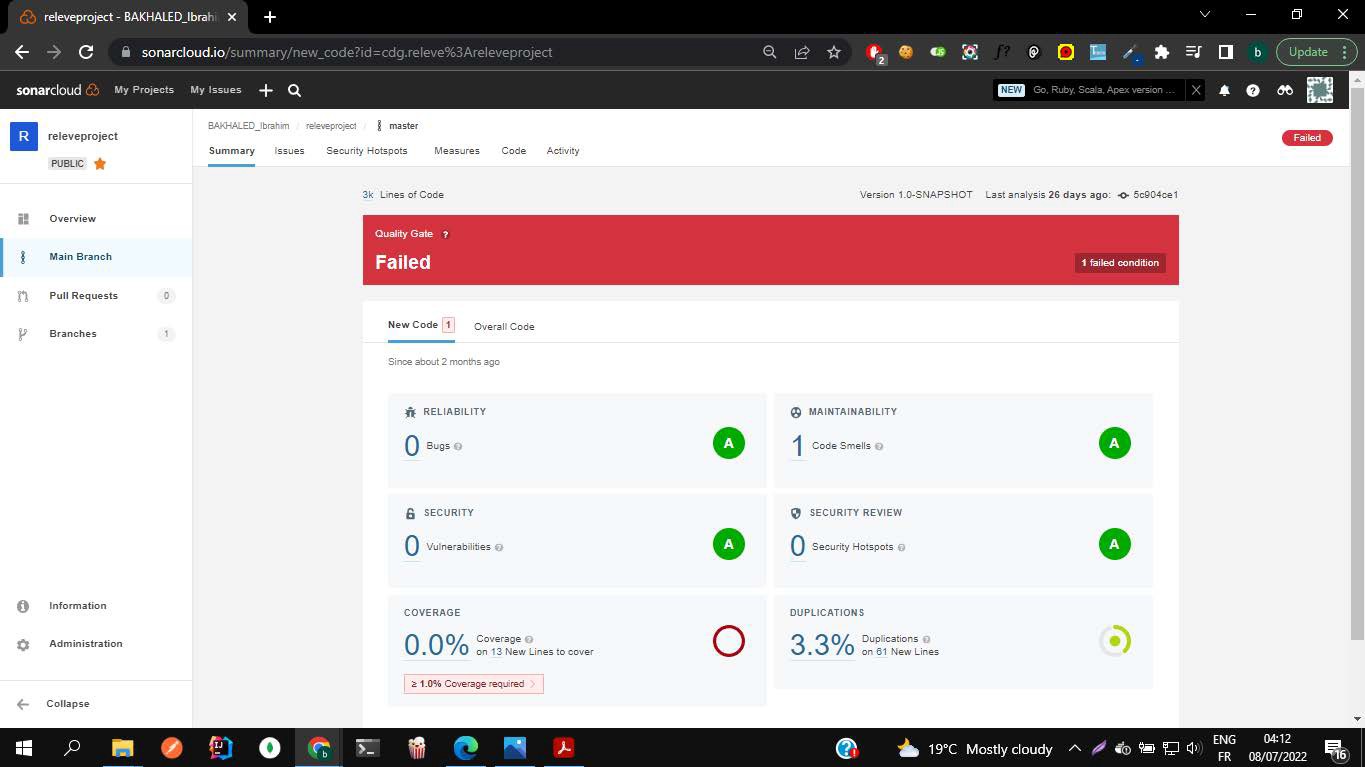
#### Interface de message broker RabbitMQ



**Figure IV.25: Interface de message broker RabbitMQ**

Cette interface affiche les détails de rabbitmq

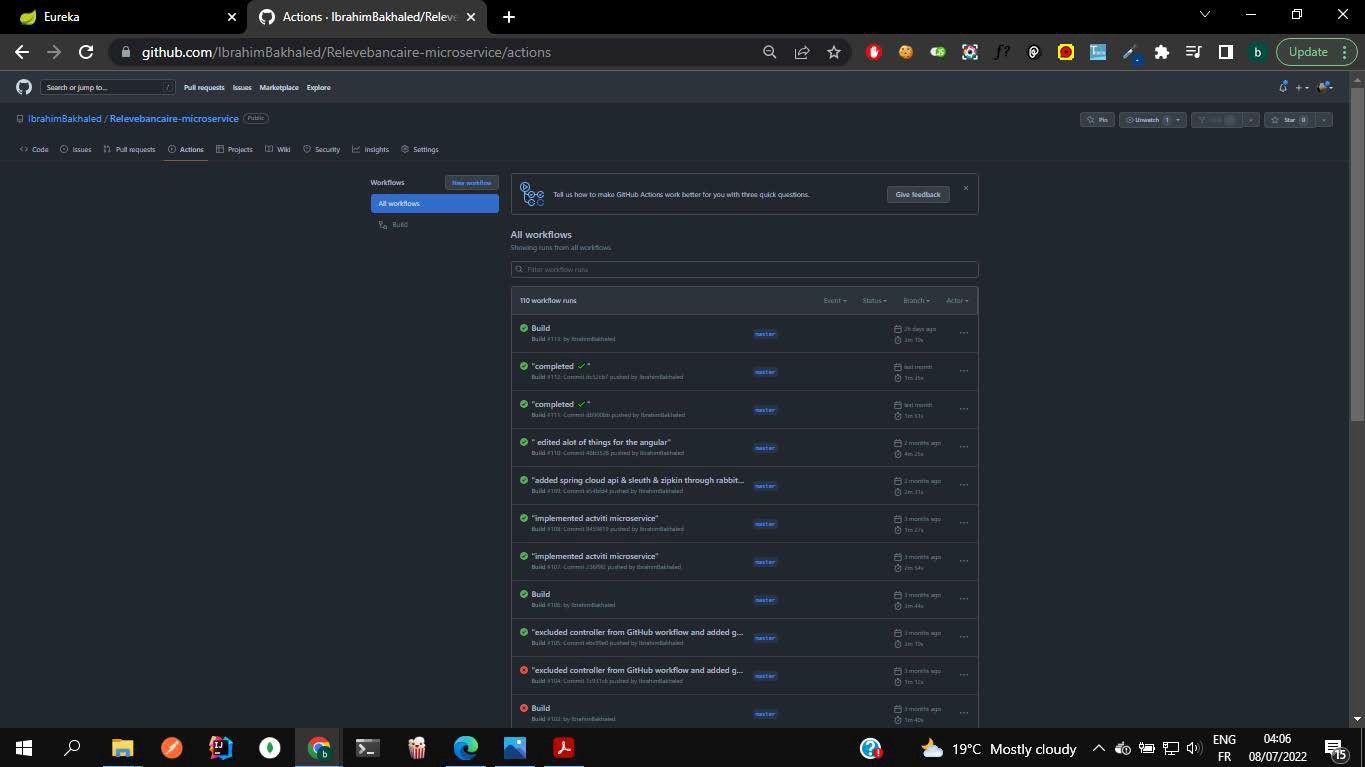
#### Interface de qualité de code



**Figure IV.26 Interface de qualité de code**

Cette interface affiche tous les détails sur la qualité du code du microservice relevebancaire.

#### Interface CI/CD avec GitHub Actions



**Figure IV.27: Interface CI/CD avec GitHub Actions**

Cette interface affiche toutes les constructions qui se sont terminées à chaque poussée vers le référentiel GitHub le vert signifie que la construction a réussi sans erreur le rouge signifie qu'il y a eu une erreur

**Conclusion générale**

**Conclusion Générale**

Tout au long de ce rapport, nous avons présenté les différentes étapes de réalisation de l’application de l'entreprise CDG Prévoyance. Pour le développement de ce projet la technique UML a été utilisée, ce qui a permis de mener correctement la tâche d’analyse des besoins à l’aide du diagramme de cas d’utilisation et la tâche de conception, ainsi les scénarios sont aussi détaillés afin d’expliquer tous tâches faite Puisque nous travaillons avec le Framework Spring Boot.

Ce projet nous a donné l’opportunité de s’initier à la vie professionnelle dans un milieu réel et avoir un début d’expérience significatif, et il nous a appris comment dès le matin on peut prendre le parti de la gaieté, comment réussir de bonnes relations pour assurer un travail de groupe, comment compter sur soi pour résoudre les problèmes au cas où ils se présentent, comment être méticuleuses dans notre travail, comment être attentives aux indications de nos supérieurs, comment être bien organisées pour accomplir dans les meilleurs délais, et meilleures conditions les tâches qui nous sont confiées.

Au cours de la réalisation de notre projet, nous avons étaient astreints par quelques limites notamment, la contrainte du temps qui était relativement un obstacle devant l’ajout de certaines autres fonctionnalités. Cependant, il était une occasion pour mettre en évidence et déployer sur le plan pratique nos connaissances en informatique.

Le projet peut être amélioré, et ce, en lui ajoutant quelques modules ou interfaces pour mieux l’adopter aux besoins de l’utilisateur et pour qu’il soit toujours fiable et au niveau des progrès atteint par la société.

**Ouvrage:**

**Bibliographie**

« Grokking The Java Developer Interview More Than 200 Questions To Crack The Java, Sping, Spring Boot & Hibernate Interview », [JATIN ARORA]

« The Complete Coding Interview Guide in Java » [ANGHEL LEONARD]

## Webographie

https://medium.com/ssense-tech/hexagonal-architecture-there-are-always- two-sides-to-every-story-bc0780ed7d9c

https://stackoverflow.com/ https://app.diagrams.net/ h[ttps://www.visual-paradigm.com/](http://www.visual-paradigm.com/)

https://github.com/IbrahimBakhaled/Relevebancaire-microservice [https://sonarcloud.io/project/overview?id](http://www.gmao-clarisse.com/FR/Interventions.htm)=cdg.releve%3Areleveproject [https://en.wikipedia.org/wiki/Okta,\_](http://www.gmao-clarisse.com/FR/Interventions.htm)Inc.#:~:text=It%20provides%20cloud

%[20](http://www.utc.fr/tsibh/public/spibh/99-00/Projets/GMAO/GMAO.htm)software%20that,website%20web%20services%20and%20devices.