Conception générale

Turing Gallery

ESI Digital Corporation

Version	Modifications apportées	Diffusion
1.0	-	01/12/2019

Table des matières

Table des figures

1. Introduction

1.1 Objet

L'objet du document « Conception générale » est de décrire en détail le fonctionnement interne du générateur de catalogue « Turing Gallery ». Ce document sera un complément aux autres documents (« Spécification des exigences », « Cahier des tests de recette », « Cahier des tests d'intégration » et « Manuel d'utilisation ») représentant les piliers du logiciel « Turing Gallery ». Ce document sera écrit en respectant la norme IEEE 830-1998 et en s'inspirant de la norme IEEE 1016-2009.

1.2 Portée

Ce document utilisera dans plusieurs situations un jargon informatique, ainsi que des diagrammes qui représentent les interfaces, les services, les modules et les communications, qui construisent le logiciel « Turing Gallery ». De ce fait, ce document est destiné exclusivement aux développeurs du logiciel.

1.3 Glossaire

Le tableau ci-dessous expliquera les différents vocabulaires techniques pouvant être susceptible de freiner la compréhension de ce document :

Terme	Définition
Framework	Un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.
Moteur de Template	Outil de modèle structurel qui simplifie la syntaxe pour assurer une bonne maintenabilité de son projet web.
Framaboard	Logiciel de gestion de tâches visuel. Il permet de gérer des projets de manière collaborative selon la méthode Kanban.
Camel Case	Une pratique qui consiste à écrire les mots en les liant sans espace ni ponctuation. Chaque mot commence par une majuscule à l'exception du premier.

JavaDoc	Outil développé par Oracle, permettant de créer une documentation d'API en format HTML depuis les commentaires présents dans un code source en Java.
Markdown	Langage de balisage léger conçu pour être facile à lire et à écrire. Un fichier Markdown fait office d'un fichier texte quand il s'agit d'afficher son contenu dans un dépôt Git.
Git Ignore	Un ensemble de modèles utilisé pour exclure certains fichiers d'un répertoire de travail Git. Il peut être local, global ou partagé à toute l'équipe.
Open Source	Un type de logiciel dans lequel le code source est publié sous une licence dans laquelle le détenteur des droits d'auteur accorde aux utilisateurs le droit d'étudier, de modifier et de distribuer le logiciel à qui que ce soit et à n'importe quelle fin.

Figure 1 : Termes et définitions

Acronyme	Définition
MVC	Modèle-Vue-Contrôleur, un Motif d'architecture logicielle visant à séparer le code source en 3 sous-parties. Ce modèle est très utilisé dans les applications Web.
JPA	Java Persistence API, une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.
REST	Representational State Transfer, un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services Web.
API	Application Programming Interface, un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

IDE	Integrated Development Environment, un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels.
OS	Operating System ou système d'exploitation, un ensemble de programmes qui dirige l'utilisation des ressources d'un ordinateur par des logiciels applicatifs.
UML	Unified Modeling Language, un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système.
IHM	Interface Homme-Machine, désigne les moyens et outils mis en œuvre afin qu'un humain puisse contrôler et communiquer avec une machine.

Figure 2 : Acronymes et définitions

1.4 Références

[IEEE Recommended Practice for Software Design Descriptions]

IEEE Computer Society

[IEEE Standard for Information Technology - Systems Design - Software Design Descriptions]

IEEE Computer Society

[Génie Logiciel – UML]

Faculté des Sciences et Techniques – François Jacquenet

[L'organisation du cycle de SCRUM]

Medium – lien vers la référence

[Cycle en V]

Wikipédia – lien vers la référence

1.5 Vue d'ensemble

Ce document, dans un premier temps, s'occupera de décrire les langages et les outils qui seront utilisés, la méthode de travail à appliquer, ainsi que les normes et standards qui doivent être respecté, lors du développement du logiciel « Turing Gallery ». Ensuite, il s'occupera de représenter les différents types de diagrammes qui décrivent le fonctionnement de ce logiciel. Enfin, il s'occupera d'afficher une première version des principales interfaces graphiques qui constituent ce logiciel.

2. Description générale

2.1 Langages de programmation

Le choix des langages de programmation pour le développement du logiciel « Turing Gallery » était fait en fonction d'un certain nombre de critères. Le besoin de l'application, les plateformes sur lesquelles elle tournera, les compétences et l'expérience des développeurs avec les différents langages de programmation, le budget réservé dans le projet « Turing Gallery » par l'entreprise « ESI Digtial Corporation » et quelques autres détails. La conjecture de tous ces critères a engendré les langages suivants :

- Côté client : HTML, JavaScript, CSS. Pour rendre le développement plus facile et rapide à faire, des Frameworks sont susceptible d'être utilisé comme Vue.JS mais également des moteurs de Template comme Thymeleaf.
- Côté serveur : Java EE. Il est indispensable d'utiliser le Framework Spring pour le développement du serveur du fait qu'il utilise un modèle MVC qui rendra l'architecture du projet plus facile à mettre en place, comprendre, modifier, tester et évoluer. Dans le cas des requêtes simples, il est envisageable d'utiliser JPA, tandis que les requêtes complexes seront faites en langage SQL.
- Base de données : En vue des récentes statistiques de performance des bases des données et en vue de notre budget actuel, nous avons opté pour la base de données MySQL.

2.2 Outils de développement

Les langages de programmation cités précédemment, sont une partie d'un ensemble de technologies et de méthodes de travail à utiliser pour assurer un développement fluide et cohérent du logiciel « Turing Gallery ». Cette partie mettra en avant les outils qui seront utilisés pour accompagner le développement :

- Gestionnaire de versioning : Git. Un outil indispensable quand il s'agit de développer une application à plusieurs collaborateurs. Non seulement que l'intégration des différents morceaux de code développés en parallèle est facile à faire, mais aussi on gardera un historique de toute l'évolution de l'architecture du projet. Cette fonctionnalité sera utile dans le cas où on veut revenir à une version ancienne pour une quelconque raison.
- Tests unitaires : Framework JUnit. Le développement du logiciel « Turing Gallery » se basera principalement sur le langage de programmation Java. Il est nécessaire de s'assurer que tous les services proposés par ce logiciel réagissent de la façon attendue, d'où l'importance de tester les différents modules dans différentes situations.
- Tests d'intégration : SoapUI. Le logiciel « Turing Gallery » utilisera le style d'architecture logicielle REST. Une séparation sera faite entre le côté client et le côté serveur et tous les services communiqueront à travers des API. L'application open source SoapUI réalisera des tests automatisés sur ces API pour garantir leurs fonctionnements.
- Intégration continue : Jenkins. Il sera envisageable d'utiliser cet outil en l'intégrant au dépôt Git, pour permettre un contrôle automatique et permanent sur les dépendances qui construisent le projet.
- Qualité logicielle : En fonction de nos exigences envers le développement du logiciel « Turing Gallery », l'outil open source SonarQube pourra être utilisé pour mesurer la qualité et la sécurité du code. Il se chargera de détecter la moindre anomalie qui se trouve dans le code analysé.

Aucun IDE, ni OS, ne sera imposé pendant le développement du logiciel « Turing Gallery ». En contrepartie, les développeurs doivent s'assurer que les librairies utilisées et le code écrit, fonctionnent indépendamment de l'OS. Ils doivent également s'assurer que les fichiers générés par les différents IDE resteront en local.

2.3 Méthode de travail

Le développement du logiciel « Turing Gallery » suivra une méthode agile pour la gestion de projet grâce au Framework de management de projet SCRUM. Un système de Sprint de 3 ou 4 semaines sera mis en place pour avancer sur des tâches spécifiques et avoir une partie du produit prête à être livré en fin de Sprint, comme la [Figure 3] le démontre en dessous. Le projet se basera également sur le modèle du cycle en V. La première phase consiste à bien préparer la base du projet en mettant en clair les besoins et les exigences du client, les contraintes imposées au logiciel, la conception générale. Puis la deuxième phase commencera par le codage des fonctionnalités définies auparavant et toutes les étapes de tests à faire, la [Figure 4] en dessous démontre tout le processus de développement. Le service en ligne Framaboard sera utilisé pour suivre le projet, gérer les tâches et avoir un accès à différents types d'analyses comme le diagramme de Gantt.

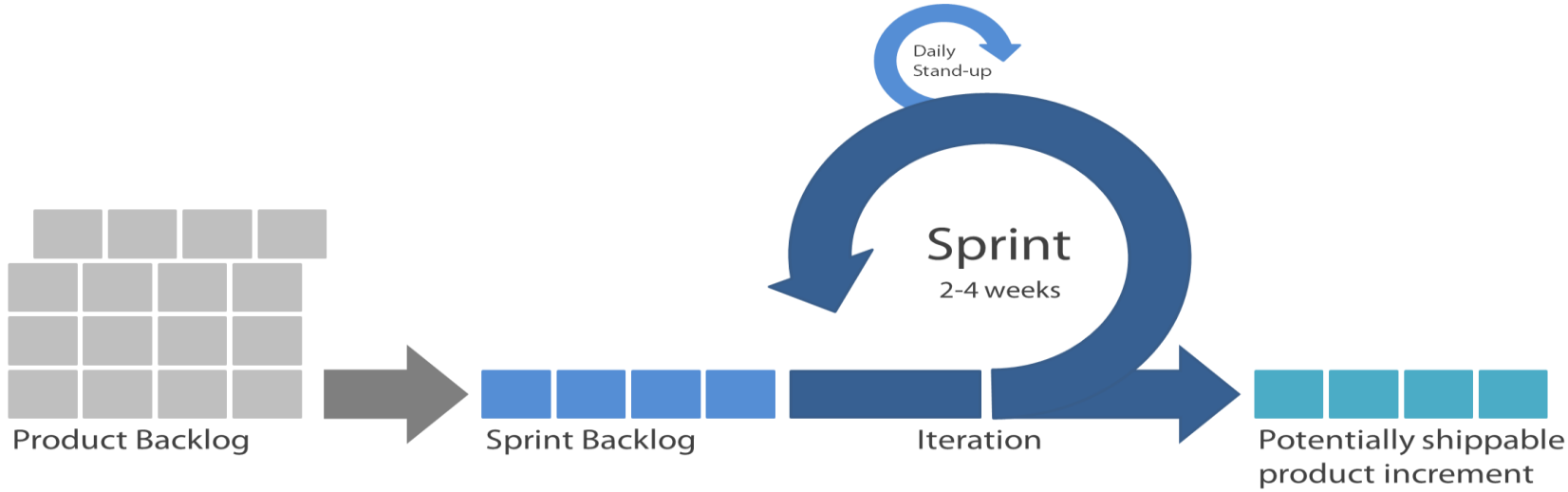

Figure 3 : Le processus d'un livrable en SCRUM

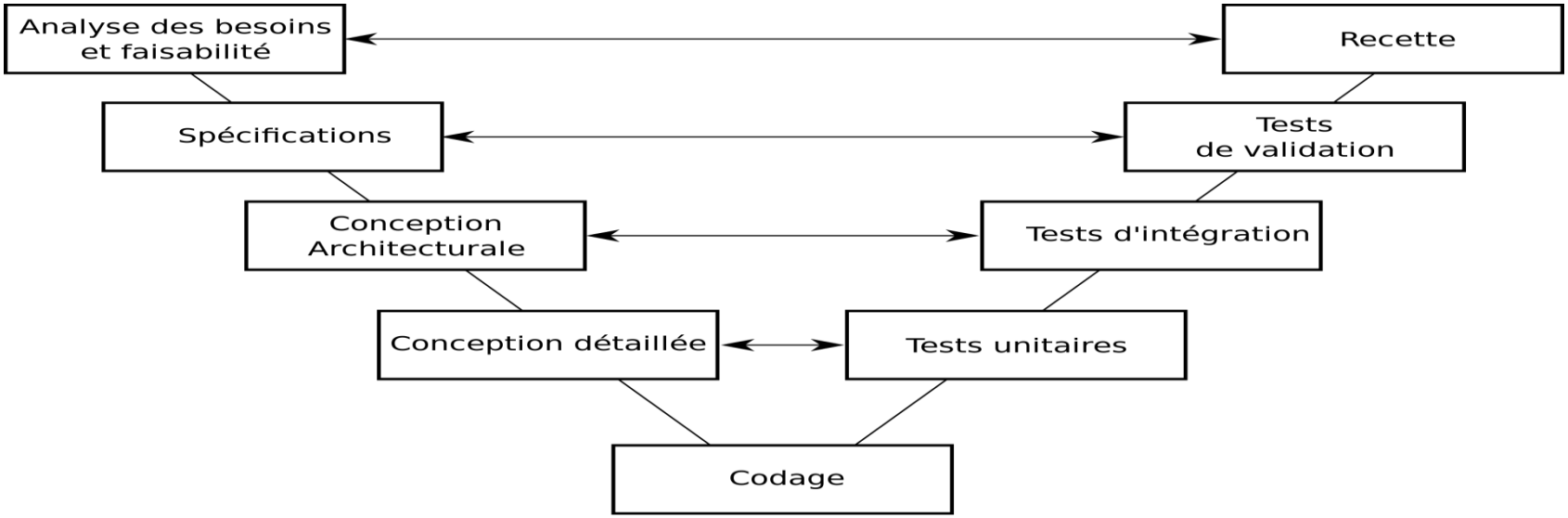

Figure 4 : Les phases du cycle en V

2.4 Normes et standards

Bien que la plupart des langages et outils seront utilisés en commun par tous les développeurs, il est essentiel d'imposer certaines règles pour rendre le code universel, et ainsi éviter de perdre du temps dans sa compréhension et son utilisation. Les normes qui seront citées peuvent être des règles générales utilisées dans la majorité des projets, mais aussi des règles personnelles qui seront spécifiques au développement du logiciel « Turing Gallery » :

- Utilisation de la pratique Camel Case lors du nommage des variables et des méthodes.
- Utilisation de l'outil JavaDoc pour la documentation du code source en Java.
- Utilisation des conventions de codage en Java.
- Indentation de code cohérente avec que des espaces, des outils peuvent être utilisés pour gérer cet aspect.
- Écriture du code et des commentaires en Anglais.
- Écriture des messages de commit de façon nette, précise et courte, en Anglais.
- Écriture des différents documents d'explication de projet en Markdown pour assurer un affichage correct sur GitHub.
- Utilisation d'un Git Ignore pour empêcher le commit des fichiers générés sur le dépôt Git.
- Toutes les ressources qui seront utilisées pendant tout le processus de développement doivent être en open source.

3. Diagrammes de conception

Ce document s'occupera dans cette partie de mettre en avant la philosophie derrière le fonctionnement du logiciel « Turing Gallery » dans des différents scénarios. Le Langage UML a été utilisé dans la conception des différents diagrammes. Ce document s'intéresse principalement aux diagrammes de cas d'utilisation, diagrammes d'activité, diagrammes de séquence et diagrammes de classes. La fin de cette partie sera consacrée à diffuser une première version des principales IHM de la version Web du logiciel « Turing Gallery ».

3.1 Diagrammes de cas d'utilisation

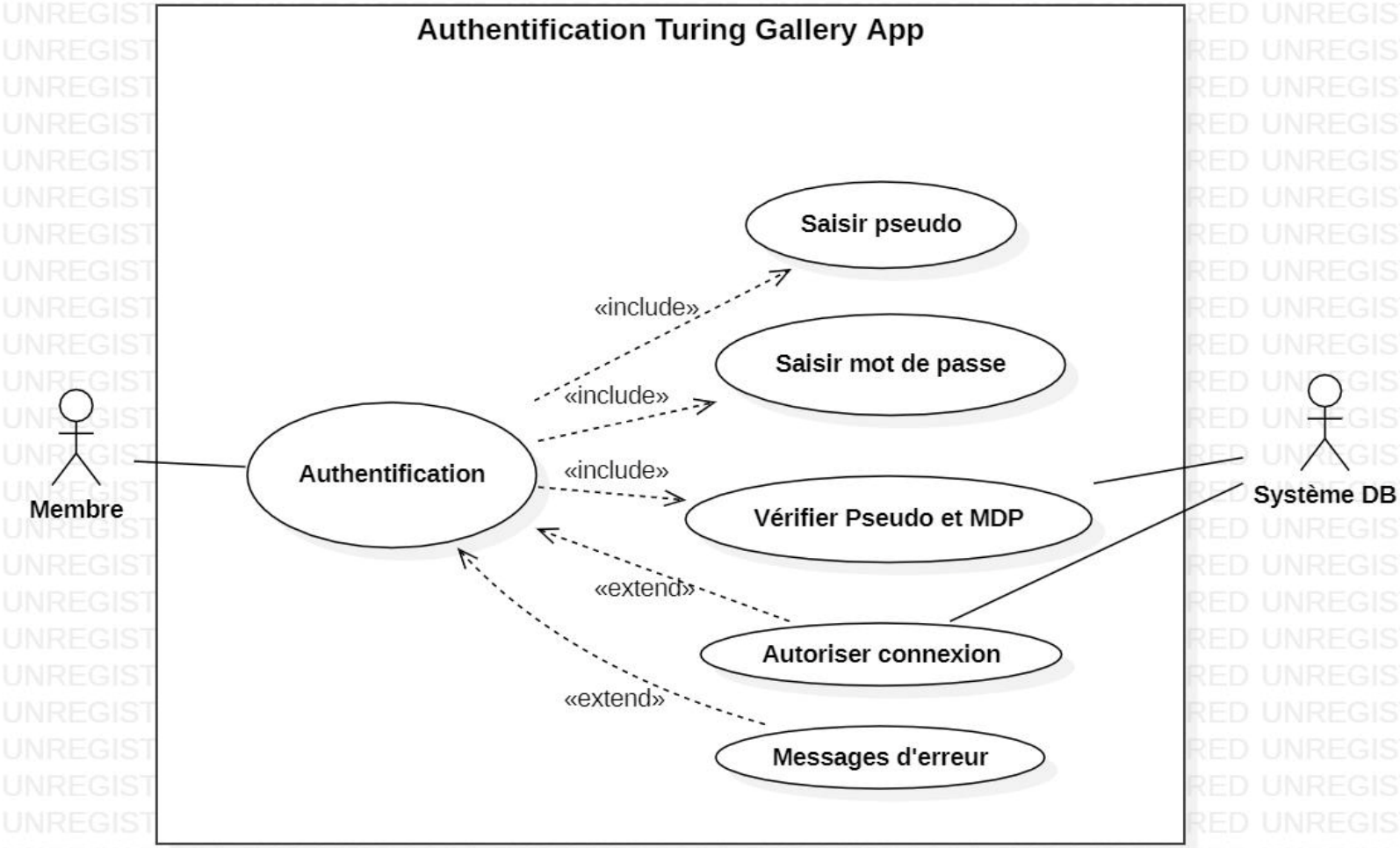

Figure 5 : Cas d'authentification

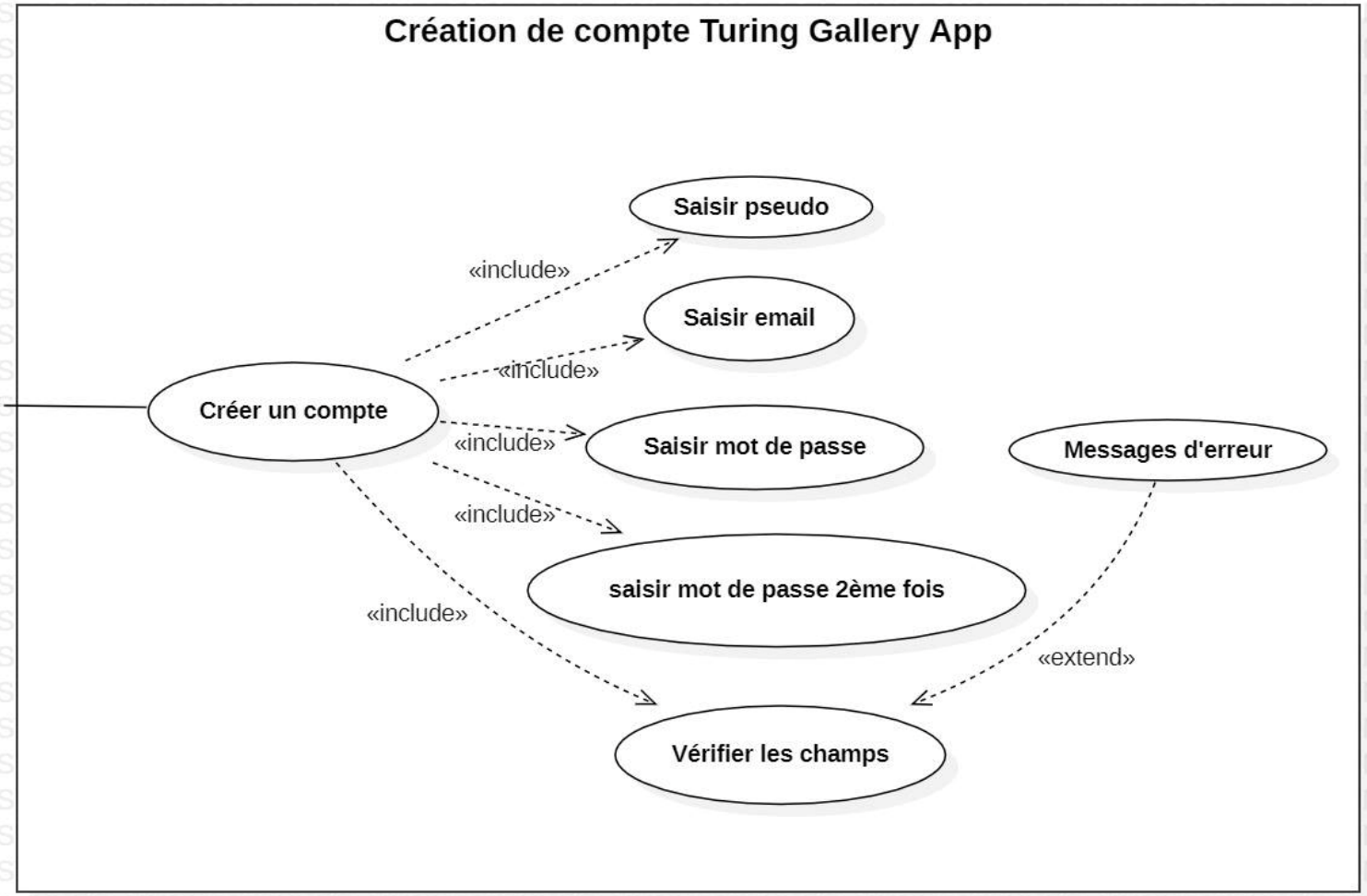

Figure 6 : Cas d'inscription

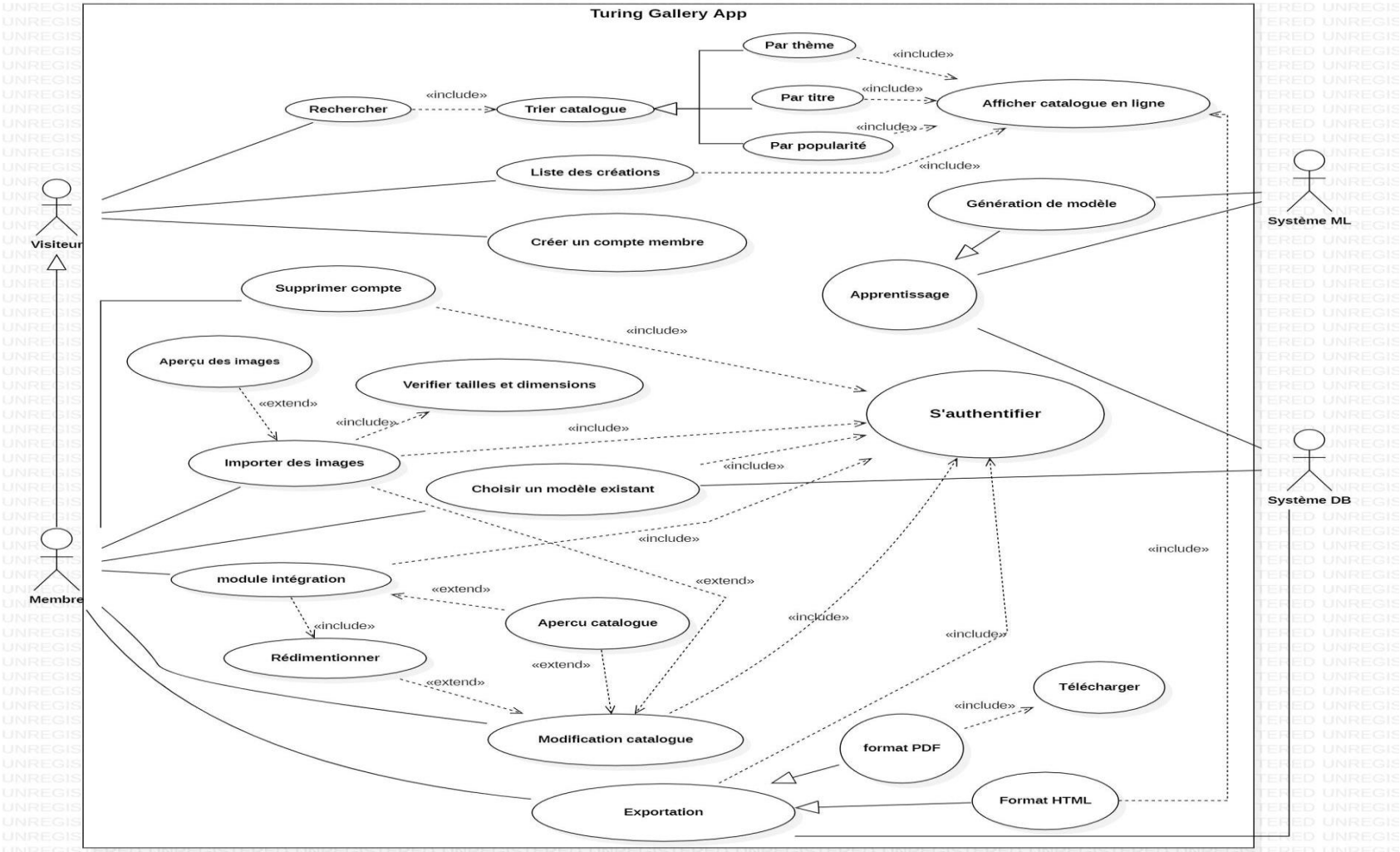

Figure 7 : Autres cas d'utilisation

3.2 Diagramme d'activité

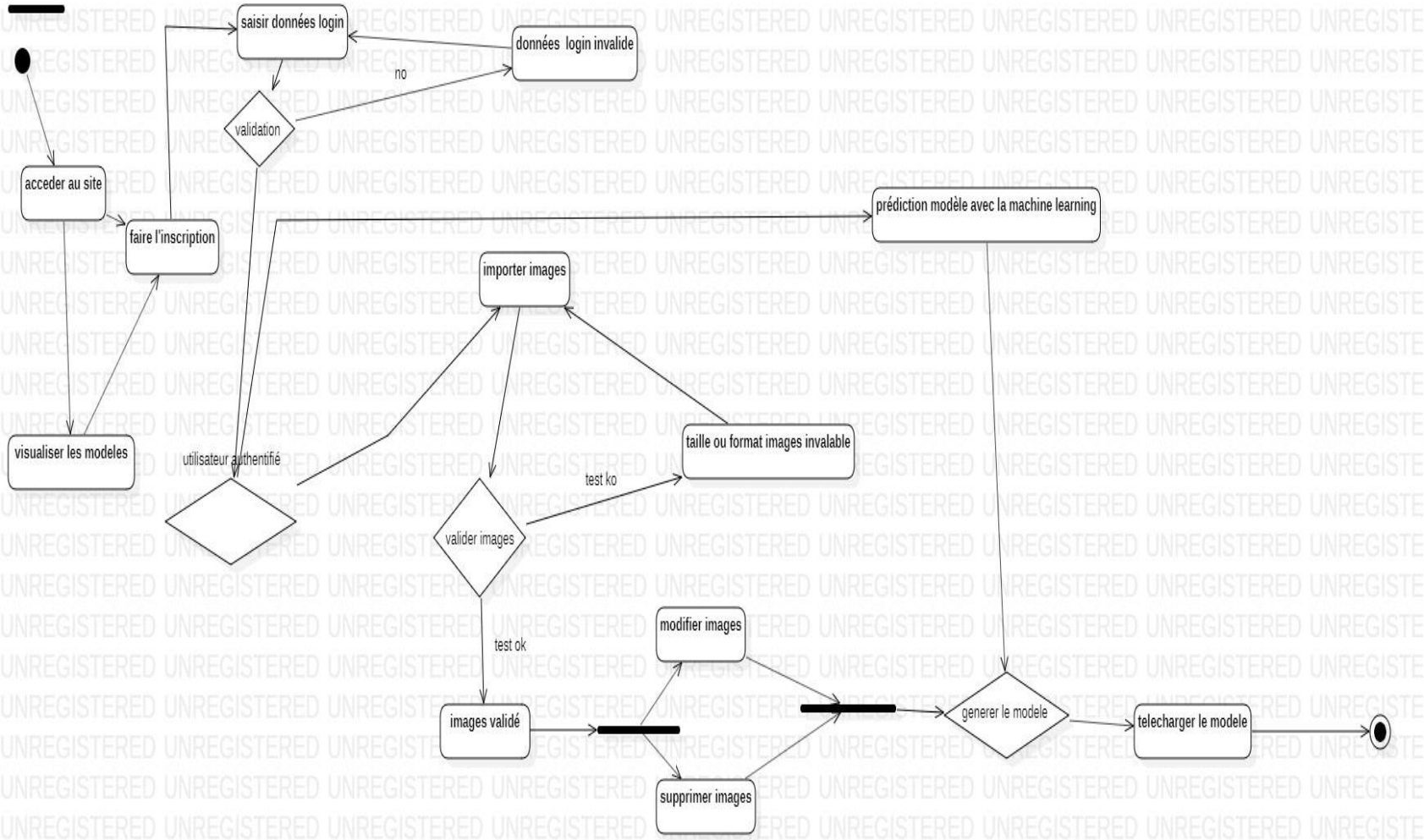

Figure 8 : Diagramme d'activité

3.3 Diagrammes de séquence

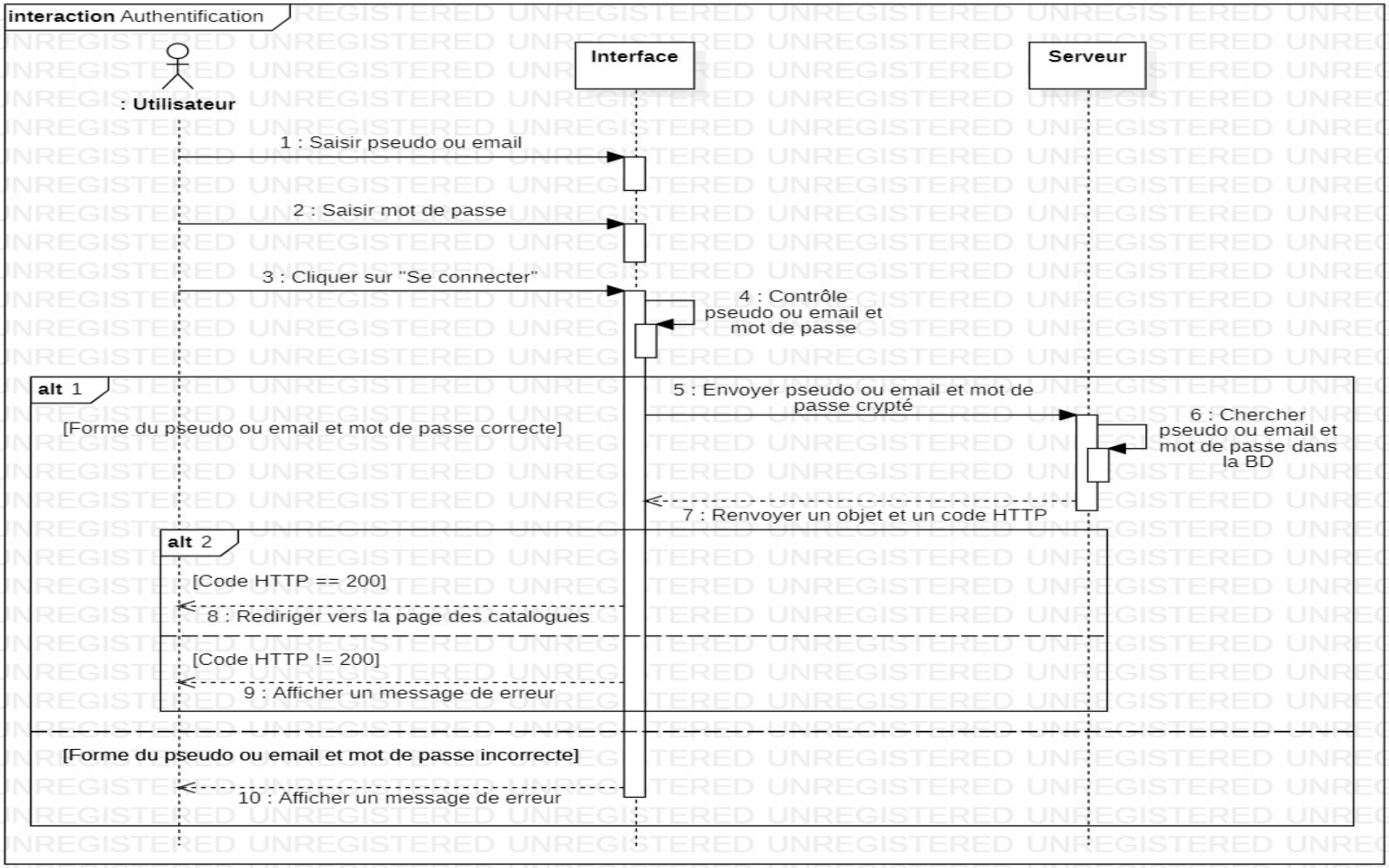

Figure 9 : Séquences d'authentification

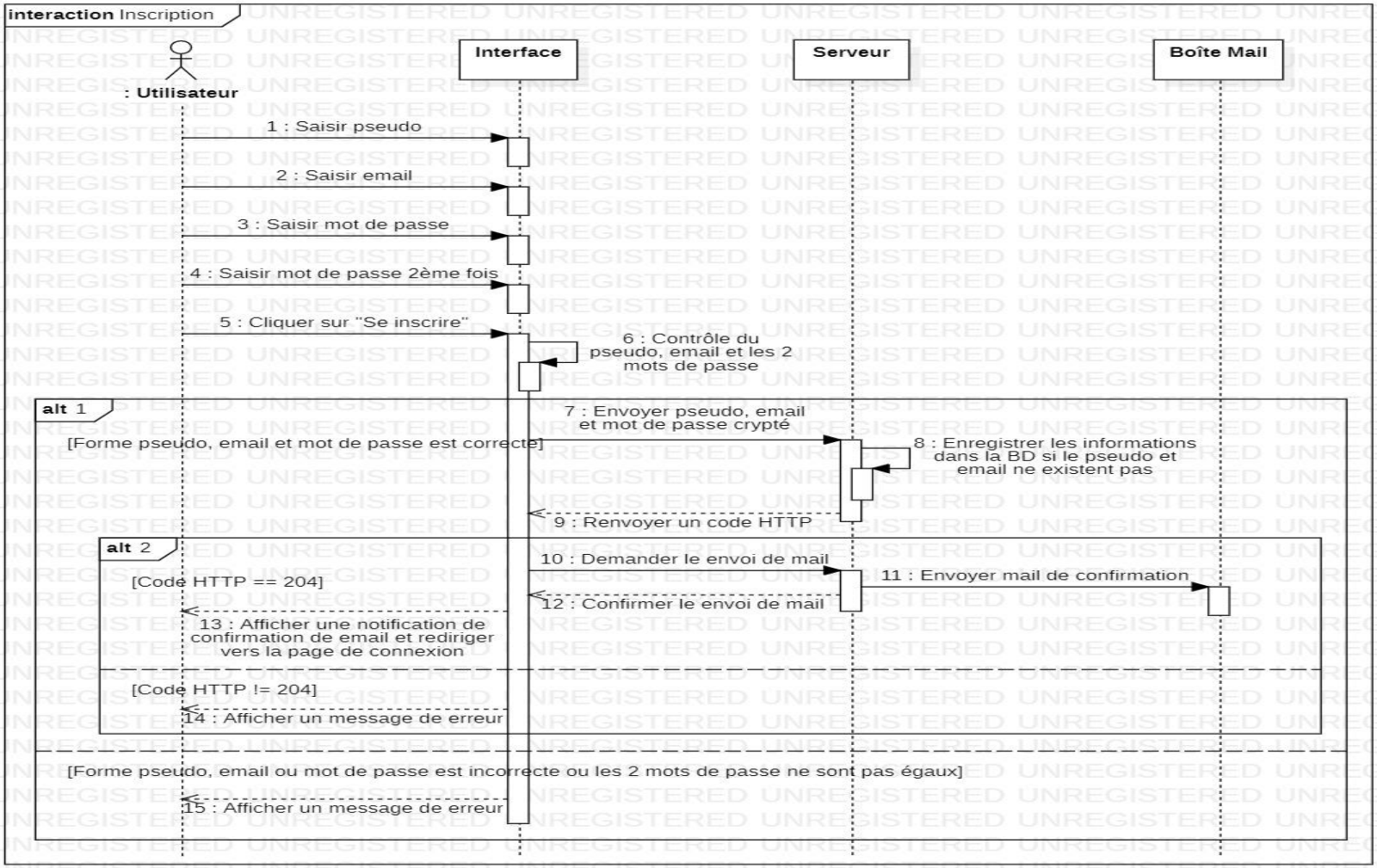

Figure 10 : Séquences d'inscription

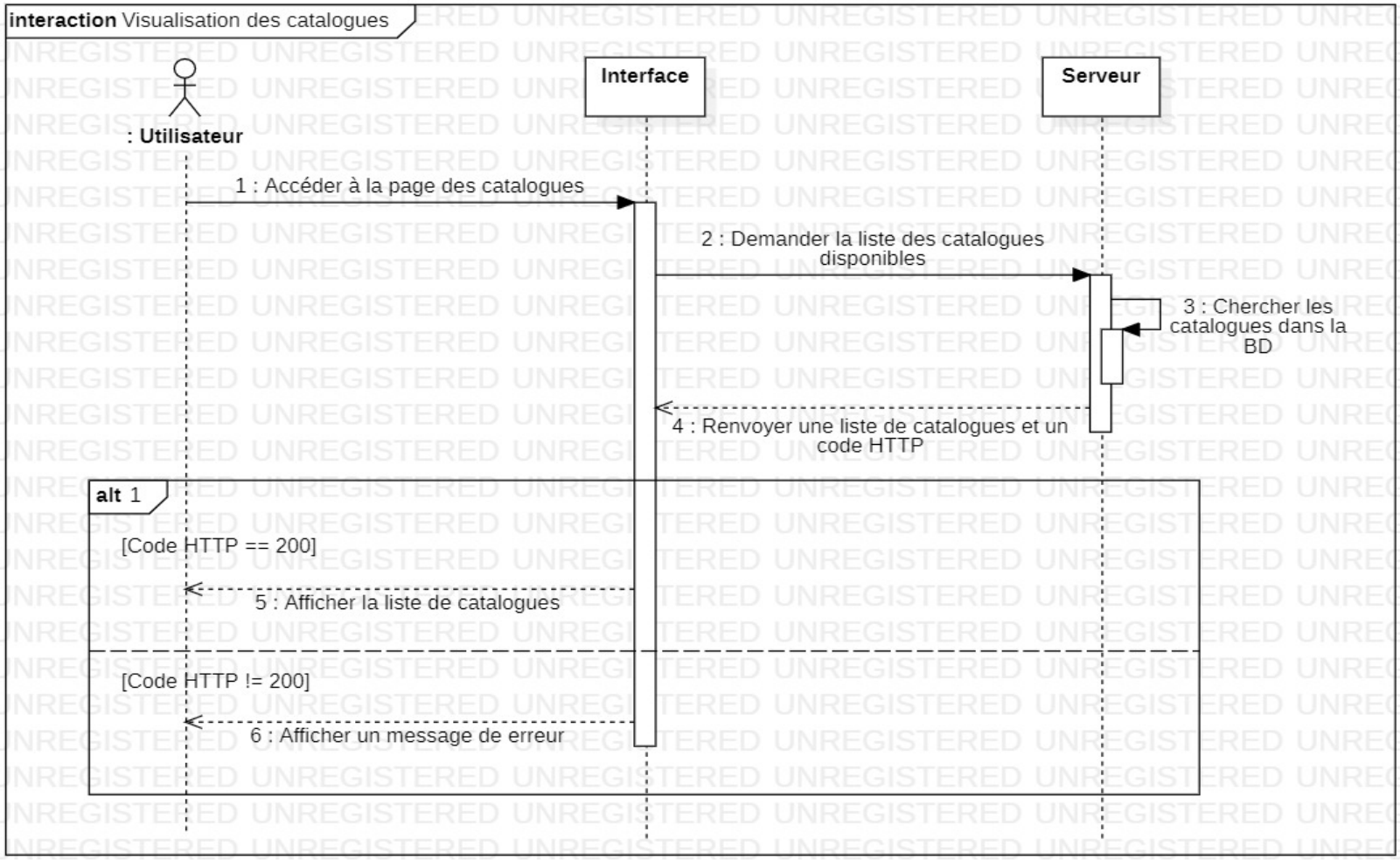

Figure 11 : Séquences de visualisation des catalogues

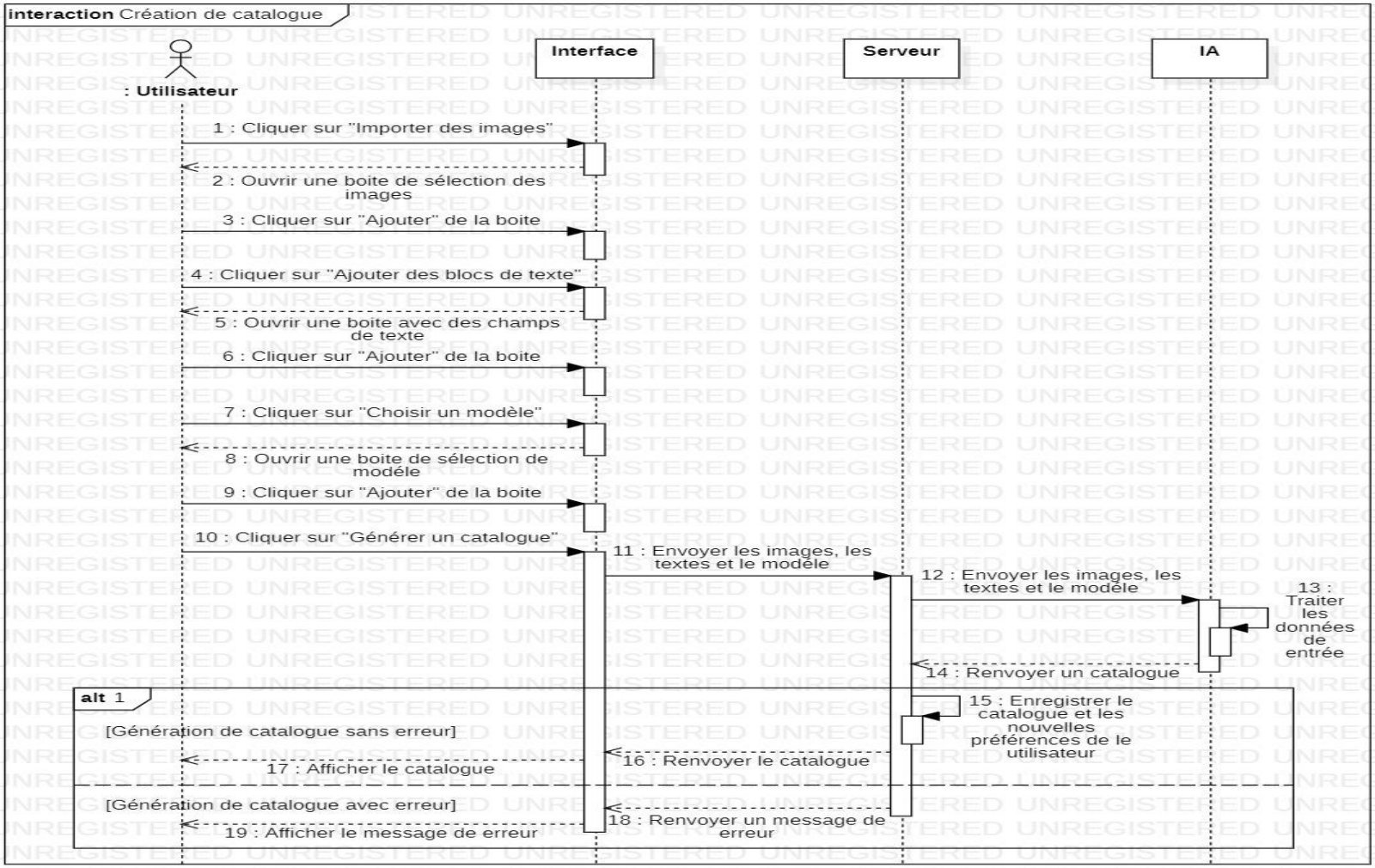

Figure 12 : Séquences de création de catalogue

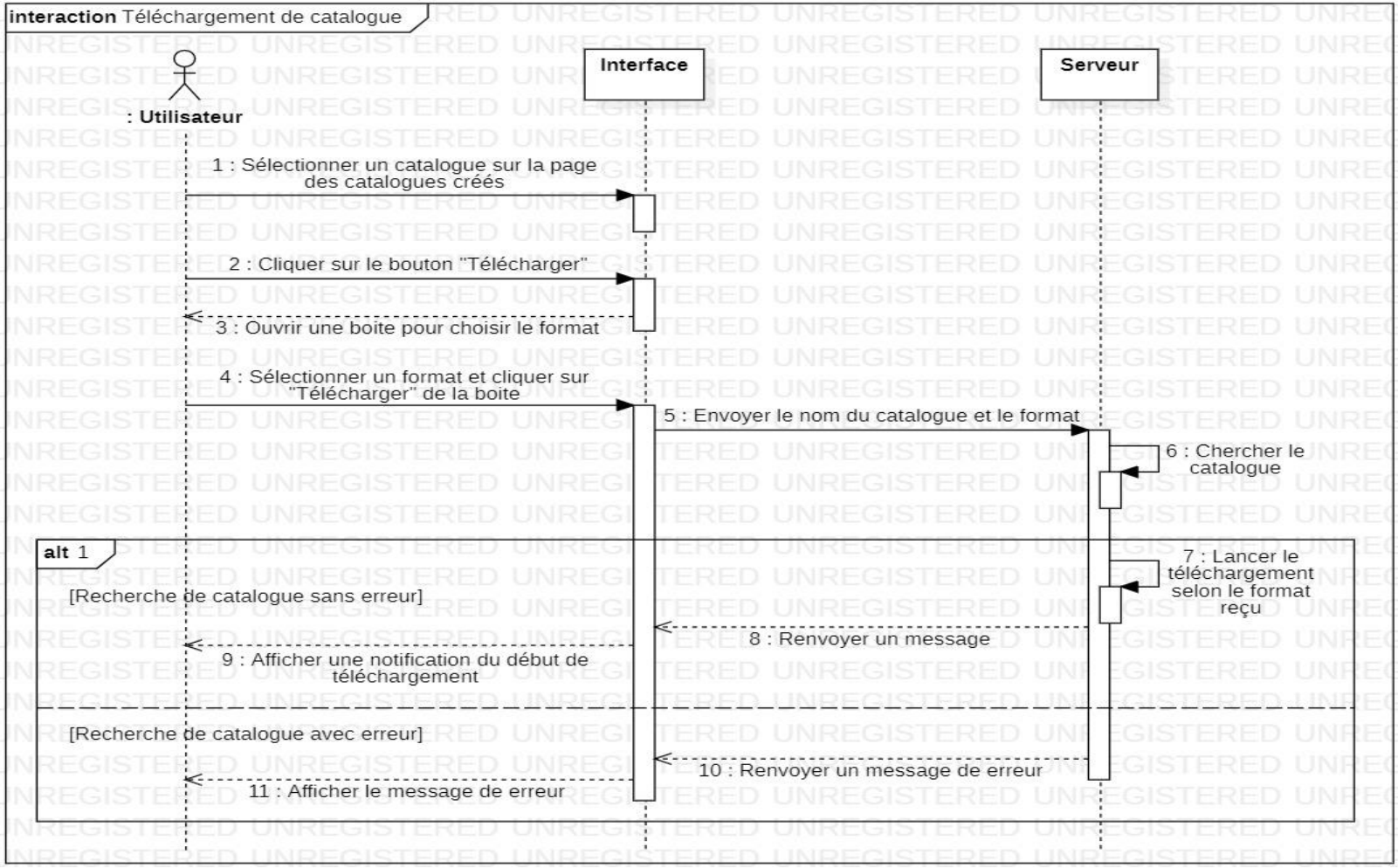

Figure 13 : Séquences de téléchargement de catalogue

3.4 Diagramme de classes

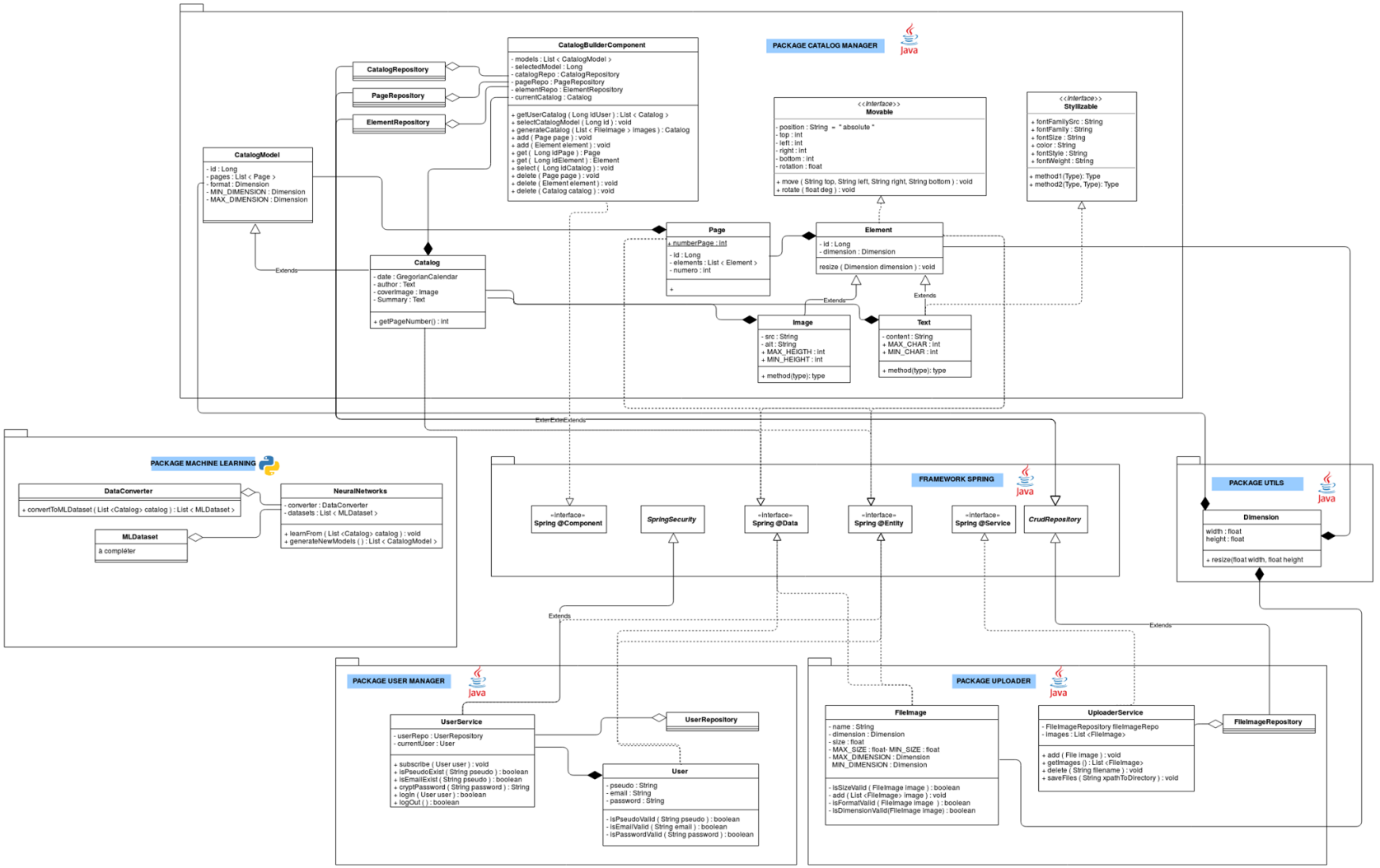

Figure 14 : Diagramme de classes