**CONCEPTION D’UN OUTIL D’AIDE A LA REMÉDIATION KYC :**

**CAHIER DE CONCEPTION**

**CONTROLE**

**M. AYISSI AYISSI Ulrich Antoine**

Organisateur Fonctionnel junior à BC-PME SA

**COORDINATION**

**M. NDJOMO EKO Rodrigue Armel**

Chef Service de l’Organisation à BC-PME SA

Banque Camerounaise des Petites et Moyennes Entreprises



**SOMMAIRE**

[**LISTE DES FIGURES** 2](#_Toc75347233)

[**LISTE DES TABLEAUX** 3](#_Toc75347234)

[**INTRODUCTION** 4](#_Toc75347235)

[**I.** **DOMAINE D’APPLICATION** 4](#_Toc75347236)

[**1.** **Objectifs du système** 4](#_Toc75347237)

[**2.** **Interfaces** 4](#_Toc75347238)

[a. Interfaces utilisateurs 4](#_Toc75347239)

[b. Interfaces logicielles 5](#_Toc75347240)

[**3.** **Contraintes générales de conception** 5](#_Toc75347241)

[**II.** **DOCUMENT DE REFERENCE** 5](#_Toc75347242)

[**III.** **NORMES STANDARTS ET OUTILS** 6](#_Toc75347243)

[**1.** **Méthodes de conception** 6](#_Toc75347244)

[**2.** **Environnement et outils de développement** 7](#_Toc75347245)

[**3.** **Standards de programmation** 7](#_Toc75347246)

[**IV.** **ANALYSE ET CONCEPTION (ETUDE PRELIMINAIRE)** 7](#_Toc75347247)

[**1.** **Capture des besoins fonctionnels** 7](#_Toc75347248)

[a. Cas d’utilisation 8](#_Toc75347249)

[b. Séquence 19](#_Toc75347250)

[c. Collaboration 26](#_Toc75347251)

[**2.** **Analyse du Système** 28](#_Toc75347252)

[a. Diagramme de classe 28](#_Toc75347253)

[b. Diagramme d’états transition 29](#_Toc75347254)

[**3.** **Capture des besoins techniques** 31](#_Toc75347255)

[a. Description des spécification ergonomiques 31](#_Toc75347256)

[b. Cas d’utilisation technique 31](#_Toc75347257)

[**4.** **Conception générique** 32](#_Toc75347258)

[**5.** **Conception préliminaire et détaillé** 33](#_Toc75347259)

[**5.1** **Conception préliminaire** 33](#_Toc75347260)

[**5.2** **Conception détaillée** 34](#_Toc75347261)

[**CONCLUSION** 38](#_Toc75347262)

# **LISTE DES FIGURES**

[Figure 1: différentes étapes de la méthode 2TUP 6](#_Toc75120801)

[Figure 2: diagramme de cas d’utilisation : module Administration 9](#_Toc75120802)

[Figure 3: digramme de cas d'utilisation : module extraction et analyse 12](#_Toc75120803)

[Figure 4: diagramme de cas d'utilisation : module correction 14](#_Toc75120804)

[Figure 5: diagramme de cas d'utilisation : module alerte et emails 16](#_Toc75120805)

[Figure 6: diagramme de cas d'utilisation "paramétrer un champ" 17](#_Toc75120806)

[Figure 7: digramme de cas d’utilisation : module reporting 18](#_Toc75120807)

[Figure 8: diagramme de séquence "s'authentifier" 20](#_Toc75120808)

[Figure 9: diagramme de séquence "créer un compte" 21](#_Toc75120809)

[Figure 10: diagramme de séquence "modifier un compte" 22](#_Toc75120810)

[Figure 11: diagramme de séquence "consulter information" 23](#_Toc75120811)

[Figure 12: diagramme de séquence "effectuer correction" 24](#_Toc75120812)

[Figure 13: diagramme de séquence "paramétrer champ" 25](#_Toc75120813)

[Figure 14: diagramme de séquence "générer reporting 26](#_Toc75120814)

[Figure 15: diagramme de collaboration" s'authentifier" 26](#_Toc75120815)

[Figure 16: diagramme de collaboration "consulter données clientèles" 27](#_Toc75120816)

[Figure 17: diagramme collaboration" modifier données clientèles" 27](#_Toc75120817)

[Figure 18: diagramme de collaboration "générer reporting" 28](#_Toc75120818)

[Figure 19: diagramme de classe 29](#_Toc75120819)

[Figure 20: diagramme d'état-transition créer et modifier un compte 30](#_Toc75120820)

[Figure 21: diagramme d'état-transition corriger anomalie 31](#_Toc75120821)

[Figure 22: diagramme de cas d'utilisation besoin technique 31](#_Toc75120822)

[Figure 23: diagramme de déploiement technique 32](#_Toc75120823)

[Figure 24: diagramme de composants 33](#_Toc75120824)

[Figure 25: diagramme de séquence système 34](#_Toc75120825)

[Figure 26: diagramme de collaboration système 35](#_Toc75120826)

[Figure 27: diagramme d'activité 36](#_Toc75120827)

# **LISTE DES TABLEAUX**

**Aucune entrée de table d'illustration n'a été trouvée.**

# **INTRODUCTION**

Ce document constitue le cahier de conception du logiciel nommé **HELPKYC,** projet de fin d’études portant sur la remédiation KYC (**Know Your Customer**) pour le compte de la BC-PME SA. Ce projet est réalisé par WAMBA Gabin et MADA Vanelle. Il est effectué dans le cadre de la cinquième année d’études au Département Informatique de l’Institut Saint Jean (Yaoundé). L’objet de ce projet est d’automatiser le processus d’aide à la remédiation KYC au sein de la BC-PME SA, car ce dernier est encore traité manuellement. Ce document vient compléter l’ensemble de la documentation rédigée au cours de ce projet. Alors que les autres documents sont des ressources destinées à décrire les éléments développés et faciliter une reprise et la maintenance du projet, ce document se concentre davantage sur l’aspect technique de la conception du projet. Seront abordés les points suivants :

* Domaine d’application ;
* Document de référence ;
* Normes, standards et outils ;
* Conception générale
* Conception détaillé.

# **DOMAINE D’APPLICATION**

# **Objectifs du système**

L’objectif de ce projet est de fournir un système permettant de détecter toutes les anomalies sur les informations clientèles de faire des corrections automatiques ou semi-automatiques et de faire un reporting sur l’ensemble des données des clients.

# **Interfaces**

## Interfaces utilisateurs

L’interface de l’application sera une interface WEB (via un navigateur WEB) contenant les modules tel que : modules gestions des utilisateurs, gestions des informations clients et le module reporting ; le tout agencé par une navigation par onglet. Cette interface web permettra l’affichage et la saisie de données.

Cette interface prendra en compte les contraintes du PDA : technologie disponible, taille de l’écran, résolution.

## Interfaces logicielles

Le concept d'interface est la pierre angulaire de la programmation [modulaire](https://fr.wikipedia.org/wiki/Modularit%C3%A9), un précurseur et un élément standard de la [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet). Dans la programmation orientée objet, l'interface d'un objet consiste en un ensemble de méthodes que l'objet doit suivre. C'est le « contrat de service » du composant. Elle est interne au système et c’est avec elle qu’il y’aura interaction (SQL, installé sur le serveur de base de données) au moyen de pilotes dédiés, de manière synchronisée. Cette base de données rassemble toutes les données manipulées par l’application.

# **Contraintes générales de conception**

Plusieurs contraintes provenant de différentes sources sont à prendre en compte dans la phase de conception du système. Ci-dessous un récapitulatif des contraintes imposées par le maitre d’ouvrage dans le cahier de charges :

* L’architecture 3-tiers sera utilisé pour l’application avec le pattern architectural MVC (Modèle- Vue- Contrôleur) ;
* La base de données doit être type Oracle ;
* L’application doit être en français ;
* La connexion au système doit être sécurisées par mot de passe. Chaque catégorie de personnes disposera de droits d’administrateurs/utilisateurs

# **DOCUMENT DE REFERENCE**

Les documents suivants sont en référence avec lecture de ce document :

* Le rapport d’analyse : qui présente la situation actuelle et ressort les différents besoins ;
* Le cahier de charges dans sa version finale : il contient les spécifications initiales des exigences.

Les cours suivants seront utilisés pour améliorer les concepts introduits dans ce dossier :

* Cours UML MIAGE ;
* Cours de gestion de projet avec la méthode 2TUP

# **NORMES STANDARTS ET OUTILS**

# **Méthodes de conception**

Les méthodes de conception sont utilisées afin d’améliorer la qualité de la conception finale.

La méthode de conception **2TUP** (2 Tracks Unified Process) sera utiliser pour mettre en place le modèle métier du système. La méthode 2**TUP** propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire qui consiste essentiellement à identifier les acteurs qui vont interagir avec le système à construire, les messages qu'échangent les acteurs et le système, à produire le cahier de conception et à modéliser le [contexte](https://www.techno-science.net/glossaire-definition/Contexte.html) (le système est une boîte noire, les acteurs l'entourent et sont reliés à lui, sur l'axe qui lie un [acteur](https://www.techno-science.net/definition/4786.html) au système on met les messages que les deux s'échangent avec le sens). La figure ci-dessous nous montre les étapes du processus 2TUP avec les différents diagrammes à chaque niveau de conception.

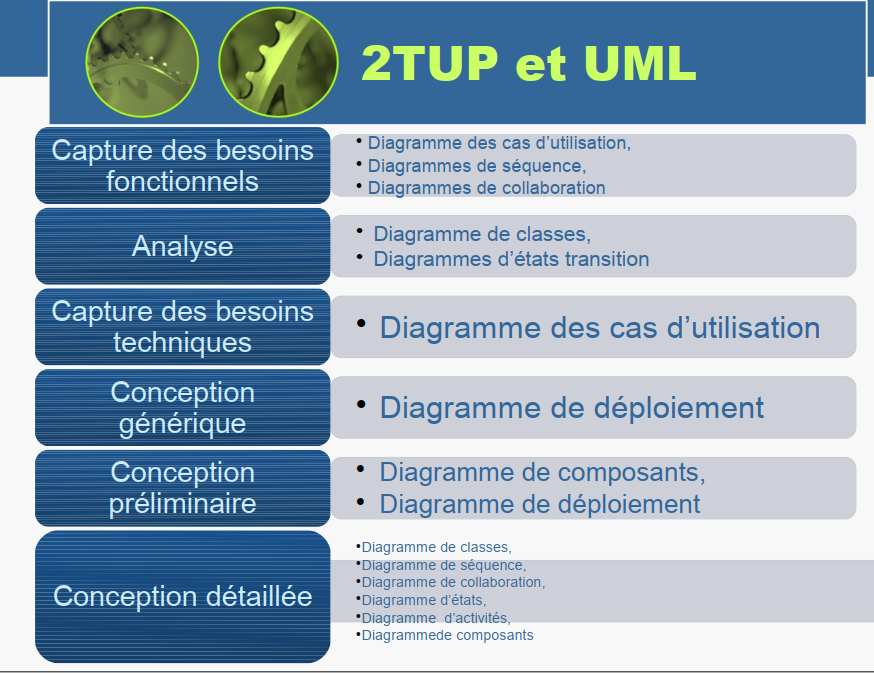


Figure 1: différentes étapes de la méthode 2TUP

**Les designs patterns :** sont utilisés pour améliorer l’architecture du logiciel. Ce sont des modèles de conception réutilisables qui correspondent à des problématiques courantes de conception indépendamment de tout langage. Ces modèles de conception fournissent un support fort pour la mise en œuvre de principes chers à l’approche par objets : la flexibilité, la réutilisabilité, la modularité, la maintenabilité…

# **Environnement et outils de développement**

Le matériel de développement utilisé est une machine préparée pour chaque développeur, équipée de **Windows 10 Professionnel** et d’une quantité suffisante de mémoire vive (le minimum a été fixé à 1Go pour avoir une qualité de développement acceptable, en partie en raison des nombreux services à exécuter).

Les deux membres de l’équipe de développement exécutent l’application sur leurs propres machines. La bases de données Oracle sera dans un premier temps situé sur leur machine personnelle ayant le rôle de serveur central. **Deux machines** sont donc nécessaires pour le développement du projet.

L’outil de développement **NetBeans** est mis à disposition de l’équipe de développement.

La **base de données Oracle** est manipulée et testée grâce à l’outil SQL Développer (lors de la phase de développement, un ensemble de données tests est utilisé afin d’avoir un support convenable pour les différents services à créer). L’environnement de conception **Entreprise Architect** est quant à lui utilisé pour la réalisation des modèles.

# **Standards de programmation**

L’équipe de développement suivra un ensemble de conventions de codage qui permettront une homogénéisation des sources :

* Les **commentaires** doivent être rédigés dans le code pour chaque classe, chaque méthode et chaque script en respectant la norme ;
* Les **conventions de codage Java WEB** utilisées sont celles recommandées par des lignes directrices de style pour la programmation ;
* Les **conventions de codage SQL** sont également celles recommandées ;
* Les **conventions approuvées par le W3C** sont également appliquées dans le cadre du développement WEB (HTML et CSS).

# **ANALYSE ET CONCEPTION (ETUDE PRELIMINAIRE)**

## **Capture des besoins fonctionnels**

## Cas d’utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés  
pour donner une vision globale du comportement fonctionnel d'un  
système logiciel. Ils sont utiles pour des présentations auprès de la direction ou  
des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont  
plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction  
entre un utilisateur (humain ou machine) et un système. Il est une unité  
significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs  
sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use  
cases).

Dans le cahier des charges, un certain nombre de fonctionnalités a été identifié.  
Nous les avons regroupés en cinq modules en fonction de leurs objectifs qui  
sont les suivants :

* Module d’Administration ;
* Module d’Extraction et Analyse ;
* Module de Correction ;
* Module Alertes et Emails ;
* Module de paramétrage ;
* Module de Reporting.

Nous allons présenter une description détaillée des modules ci-avant identifiés. Ainsi, pour chacun d’eux, nous présenterons :

* L’objectif du module ;
* Les acteurs qui interviennent directement dans le module ;
* Les fonctionnalités du module ;
* Les diagrammes de cas d’utilisation mettant en exergue de façon schématique comment les acteurs interviennent dans la réalisation des fonctionnalités identifiées.
* **Module d’Administration**

Objectif : Ce module sera en charge de la gestion des utilisateurs. Il s’agira précisément de la création et la gestion des différents comptes utilisateurs.

Acteurs : les acteurs concernés sont les Administrateurs.

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* Créer des comptes utilisateurs ;
* Modifier des comptes utilisateurs ;
* Supprimer des comptes utilisateurs ;
* Consulter l’historique des modifications effectuées.

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :

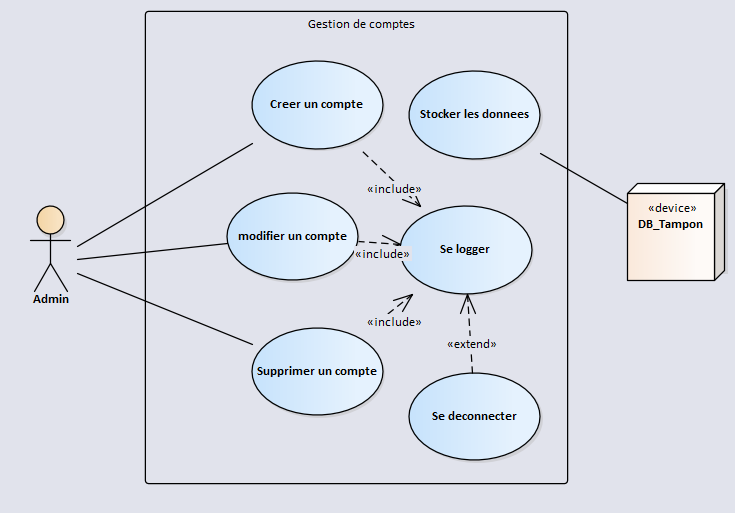


Figure 2: diagramme de cas d’utilisation : module Administration

Description textuelle des cas d’utilisation :

* Se loger

|  |  |
| --- | --- |
| *Cas d’utilisation* | Se loger |
| *Acteur primaire* | Administrateur, Utilisateur |
| *Précondition* | Existence de la page d’authentification sur la plateforme et d’une combinaison d’user et passe Word stockés dans la base de données |
| *Post condition* | Utilisateur connecté |
| *Description* | 1. L’administrateur ou l’utilisateur saisi son nom d’utilisateur et son mot de passe 2. Valide sur connexion 3. Le compte de l’administrateur ou de l’utilisateur est ouvert |
| *Exception* | Le compte erroné ou n’existe pas |

* Créer un compte

|  |  |
| --- | --- |
| *Cas d’utilisation* | Créer compte |
| *Acteur primaire* | Administrateur |
| *Précondition* | L’administrateur est connecté  Existence de la page de création de compte |
| *Post condition* | Compte créé |
| *Description* | 1. L’administrateur saisi les informations requises pour la création du compte 2. Valide sur créer 3. Le compte est créé |
| *Exception* | Le compte existe déjà ou champ obligatoire non rempli |

* Supprimer un compte

|  |  |
| --- | --- |
| *Cas d’utilisation* | Supprimer compte |
| *Acteur primaire* | Administrateur |
| *Précondition* | L’administrateur est connecté  Existence du compte à supprimer |
| *Post condition* | Compte supprimé |
| *Description* | 1. L’administrateur va sur la liste des comptes 2. Clique sur le bouton supprimé du compte à supprimer 3. Le compte est supprimé |
| *Exception* | Le compte n’existe pas |

* **Module d’extraction et d’analyse**

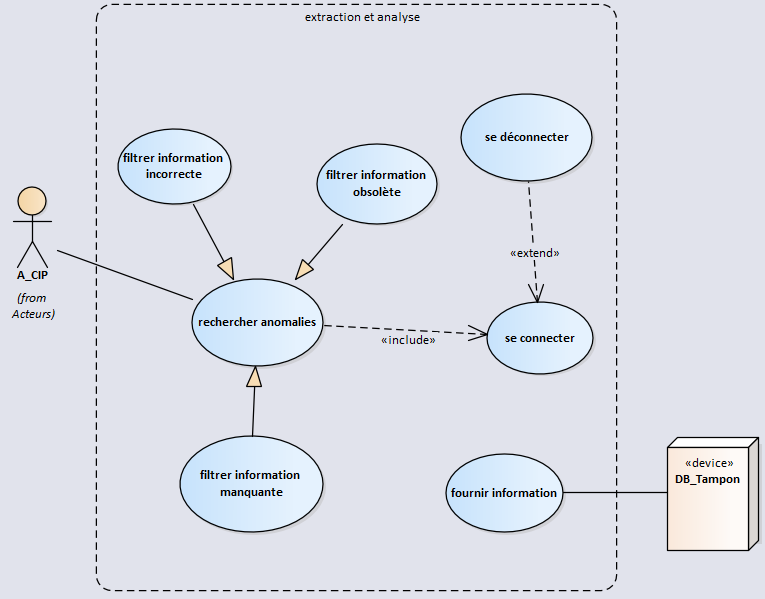
Objectif : c’est dans ce module que se passeront la détection, l’extraction et l’analyse des comptes clients possédant des anomalies sur leurs informations d’EER (Entrées en Relation)

Acteurs : les acteurs concernés sont les utilisateurs exploitant du système.

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* D’effectuer une recherche d’anomalies sur les informations clientèles selon un ou plusieurs critères (champ vide, information incorrecte, information obsolète)
* D’effectuer une recherche d’anomalies sur les informations clientèles selon les contrôles sur le nom du père ou le nom de la mère (nom du père ou le nom de la mère= **INCONNU/INCONU**)
* D’effectuer une recherche d’anomalies sur les informations clientèles dont le numéro de patente est mal renseigné (nombre de caractères en moins ou en plus) ;
* D’effectuer une recherche des anomalies sur les informations clientèles dont le numéro d’identité sociale est mal renseigné ;
* D’effectuer une recherche des anomalies sur les informations clientèles dont le numéro d’identité fiscale est mal renseigné ;
* D’effectuer une recherche des anomalies sur les informations clientèles dont le numéro du registre de commerce est expiré ou mal renseigné ;
* D’effectuer une recherche des anomalies sur les informations clientèles dont le numéro de la CNI est expiré ou mal rempli ;

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :



**Cas d’utlisation imcomplet**

Figure 3: digramme de cas d'utilisation : module extraction et analyse

Description textuelle des cas d’utilisation :

* Se déconnecter

|  |  |
| --- | --- |
| *Cas d’utilisation* | Se déconnecter |
| *Acteur primaire* | Administrateur, utilisateur |
| *Précondition* | L’administrateur ou l’utilisateur est connecté |
| *Post condition* | Administrateur ou Utilisateur déconnecté |
| *Description* | 1. L’administrateur ou l’utilisateur clique sur le bouton se déconnecter 2. Valide sur le bouton de confirmation 3. Le compte est déconnecté |
| *Exception* | Le compte n’existe pas |

* Rechercher anomalies

|  |  |
| --- | --- |
| *Cas d’utilisation* | Rechercher anomalies |
| *Acteur primaire* | Agent CIP |
| *Précondition* | Existence du module d’extraction et d’analyse |
| *Post condition* | Listes des comptes clients et leur anomalies respective |
| *Description* | 1. L’agent CIP se dans le module d’extraction et d’analyse 2. Clique sur le bouton rechercher anomalies 3. Une liste des comptes en anomalies avec leur critères respectives est affichée |
| *Exception* | Aucune anomalie constatée |

* **Module de correction**

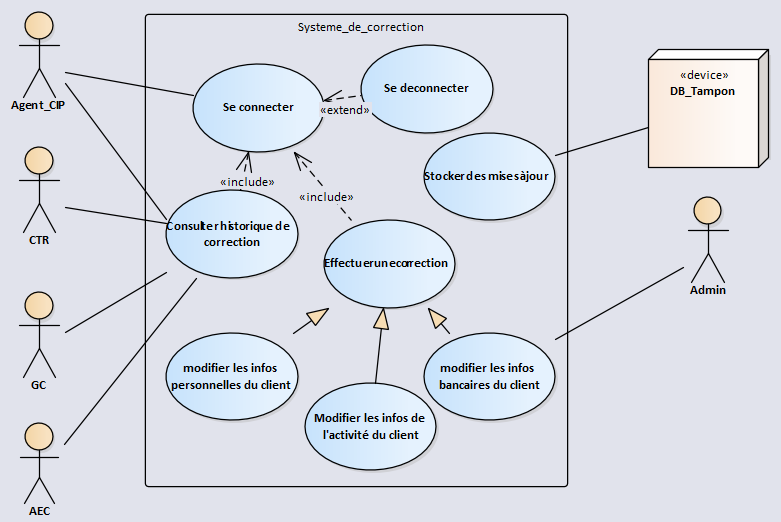
Objectif : c’est dans ce module que se passeront la correction des différentes anomalies détectées par le système sur les informations clientèles.

Acteurs : les acteurs concernés sont les utilisateurs exploitant du système (agent\_CIP, gestionnaire de compte (GC), l’agent d’exploitation Commercial (AEC), le contrôleur et l’administrateur).

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* De mettre à jour les contacts des clients (téléphone, emails) ;
* D’effectuer des corrections (automatiques et semi-automatiques) sur les champs en anomalies et sauvegarder ;

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :



**Cas d’utilisation à revoir**

Figure 4: diagramme de cas d'utilisation : module correction

Description textuelle des cas d’utilisation :

* Effectuer une correction

|  |  |
| --- | --- |
| *Cas d’utilisation* | Effectuer une correction |
| *Acteur primaire* | Agent CIP |
| *Précondition* | Existence des clients en anomalies |
| *Post condition* | Compte mis à jour |
| *Description* | 1. L’agent CIP se positionne sur le client à corriger 2. Effectue les mises à jour sur le client 3. Client mis à jour |
| *Exception* | Aucun client en anomalie |

* Consulter historique de correction

|  |  |
| --- | --- |
| *Cas d’utilisation* | Consulter historique de correction |
| *Acteur primaire* | Agent CIP, GC, CTR, AEC |
| *Précondition* | Existence des comptes mis à jour, l’utilisateur chargé des mails doit être connecté à l’outil. |
| *Post condition* | Liste de l’historique de correction |
| *Description* | 1. L’utilisateur clique sur le bouton historique de correction 2. Sélectionne la période ou la date dont il souhaite afficher les historiques 3. Une fenêtre présentant l’historique des corrections est affichée |
| *Exception* | Aucune mise à jour effectué |

* **Module Alertes et emails :**

Objectif : c’est dans ce module que seront effectué les alertes sur les comptes présentant des anomalies et seront également géré les mails pour demande de correction.

Acteurs : les acteurs concernés sont les utilisateurs exploitant du système.

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* De faire des alertes, notifications et propositions sur les anomalies

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :

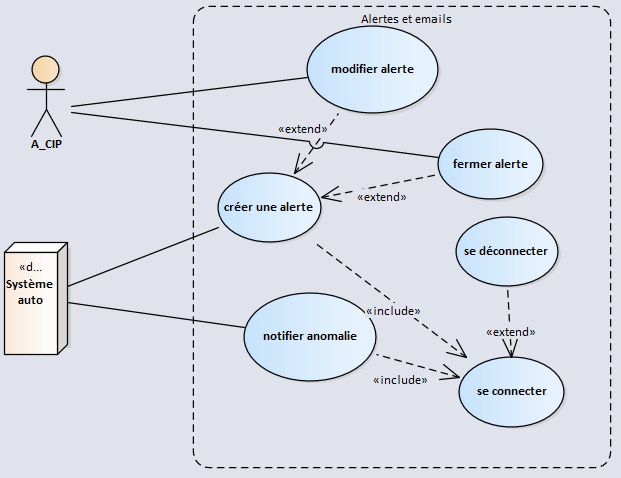


Figure 5: diagramme de cas d'utilisation : module alerte et emails

*Je ne vois nulle part dans le cas d’utilisation où un mail est envoyé*

Description textuelle des cas d’utilisation :

* Créer une alerte

|  |  |
| --- | --- |
| *Cas d’utilisation* | Créer une alerte |
| *Acteur primaire* | Système automatique |
| *Précondition* | Présence d’anomalie |
| *Post condition* | Liste des alertes |
| *Description* | 1. Le système auto détecte une anomalie 2. Il créé une alerte 3. Il notifie un utilisateur |
| *Exception* |  |

* **Module de paramétrage :**

Objectif : ce module permet d’effectuer les modifications sur les différents champs contenant les informations clientèles.

Acteurs : les acteurs concernés sont les utilisateurs exploitant du système.

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* Effectuer les modifications sur les différents champs.

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :

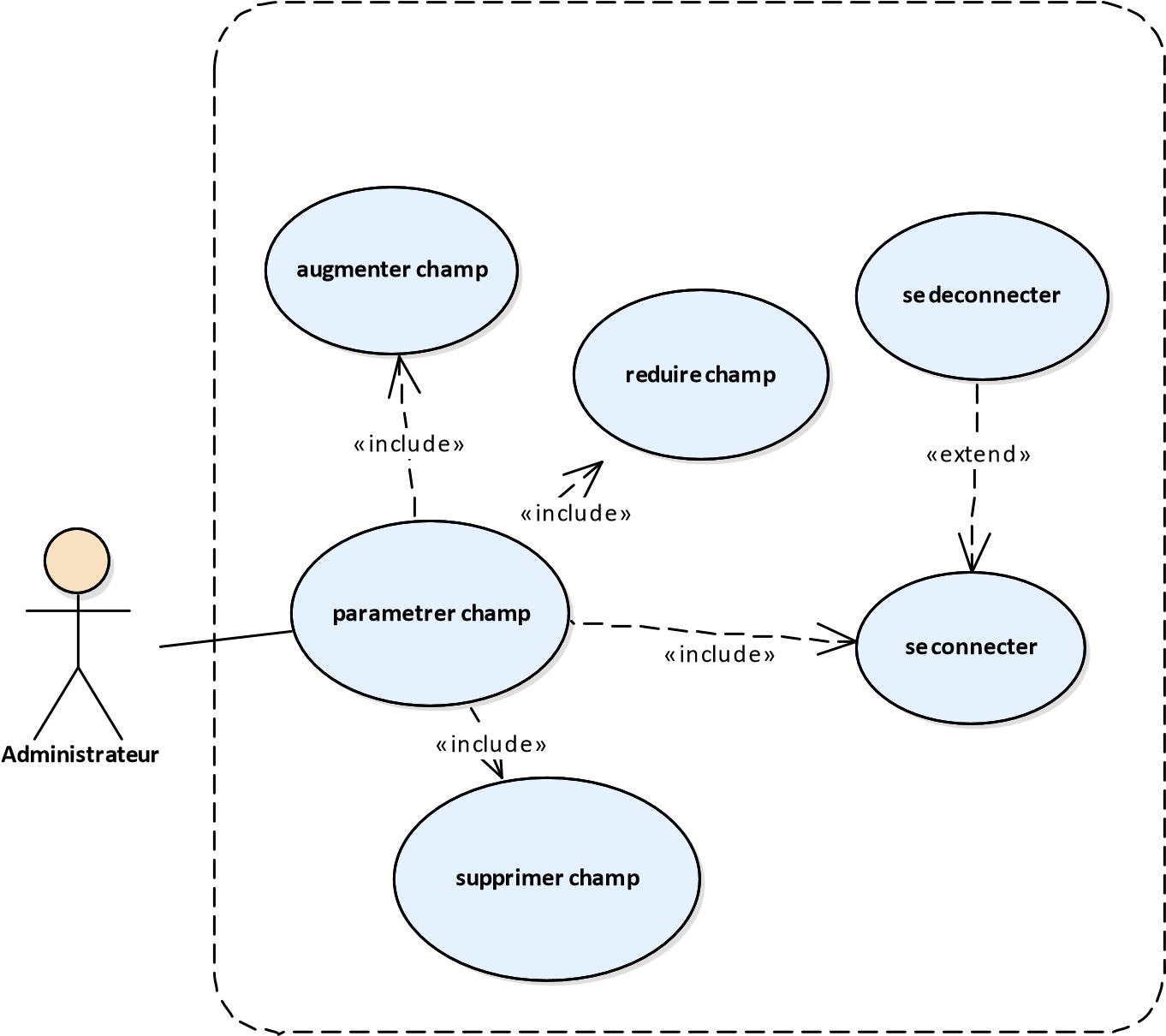


Figure 6: diagramme de cas d'utilisation "paramétrer un champ"

* **Module Reporting :**

Objectif : c’est dans ce module que seront effectués les repporting sur les anomalies et sur d’autres données qui permettre d’apprécier l’évolution de ses dernières, dans le but non seulement de voir l’utilité de notre outil mais aussi de faciliter la prise de décision.

Acteurs : les acteurs concernés sont les utilisateurs exploitant du système (agent\_CIP).

Fonctionnalités : ce module est constitué des cas d’utilisation suivants :

* Se connecter/se déconnecter sur l’outil ;
* Générer un reporting ;
* Publier/partager le reporting

Diagramme de cas d’utilisation : à partir des acteurs et des cas d’utilisations identifiés, nous déduisons le diagramme suivant :

Reporting

**A\_CIP**

*from*

*(*

*Acteurs)*

**se connecter**

**se déconnecter**

**generer reporting**

**reporting sur anomalie**

**autre reporting**

**CTR**

*(*

*from*

*Acteurs)*

**publier**

«extend»

«includ

»

~~e~~

«extend»

Figure 7: digramme de cas d’utilisation : module reporting

* Faire un reporting

|  |  |
| --- | --- |
| *Cas d’utilisation* | Faire un reporting |
| *Acteur primaire* | Agent CIP, CTR |
| *Précondition* | Existences des données (informations sur les clients) dans la base de données |
| *Post condition* | Etats sur l’évolution des corrections |
| *Description* | 1. L’utilisateur clique sur le bouton générer un reporting 2. Précise les indicateurs qu’il voudrait apprécier puis valide 3. Une fenêtre présentant les graphes est affichée |
| *Exception* | Aucune donnée dans le système |

## Séquence

Les cas d’utilisation décrivent les interactions des acteurs avec l’application que nous voulons spécifier et concevoir. Lors de ces interactions, les acteurs produisent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci. Nous allons isoler ces messages et les représenter graphiquement sur des diagrammes de séquence.

Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent à l'aide de messages.

Les différents diagrammes de séquence représentés ci-dessous sont liés aux différents cas d’utilisations de nos diagrammes de cas d’utilisation.

* Se logger

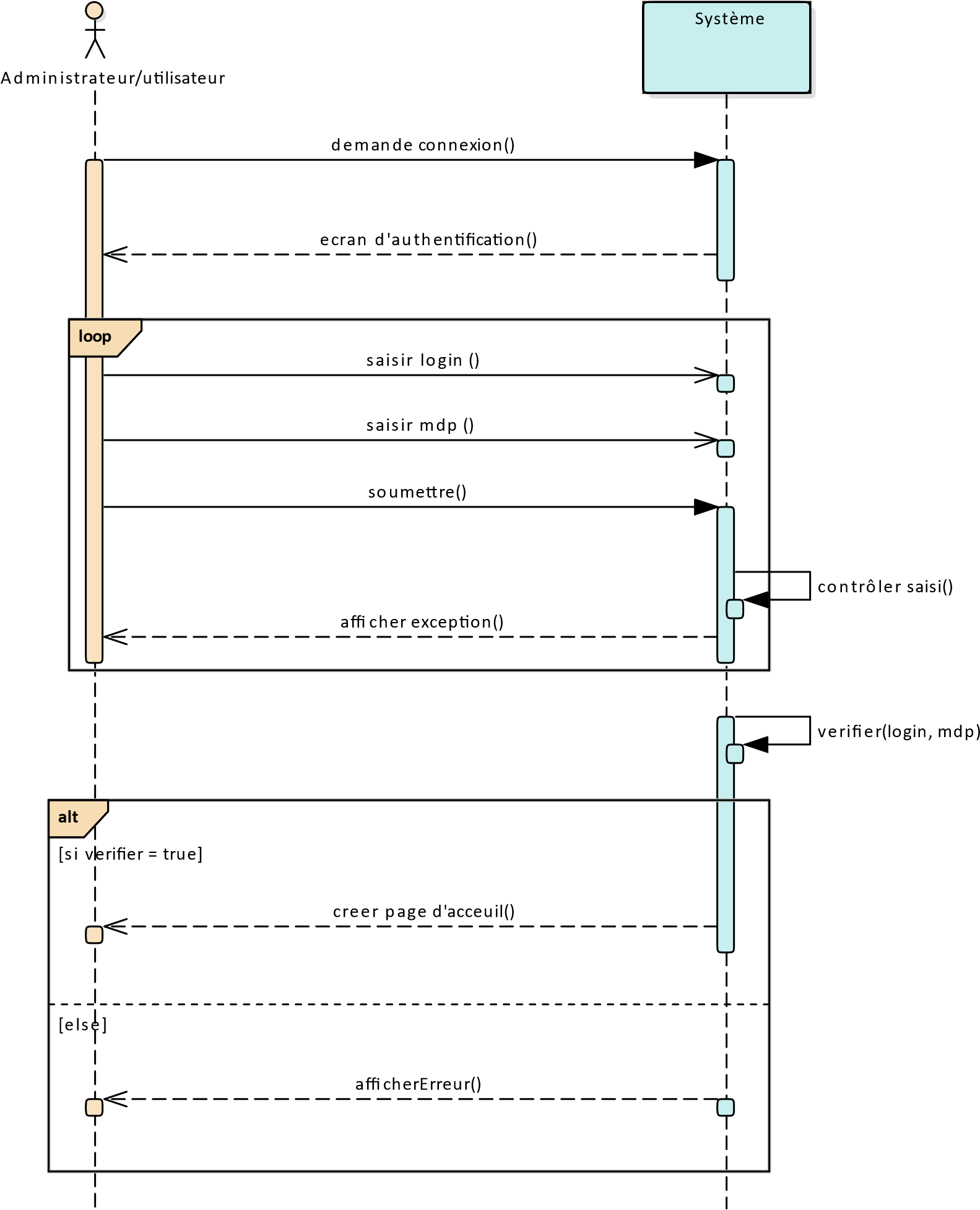


Figure 8: diagramme de séquence "s'authentifier"

* Créer un compte

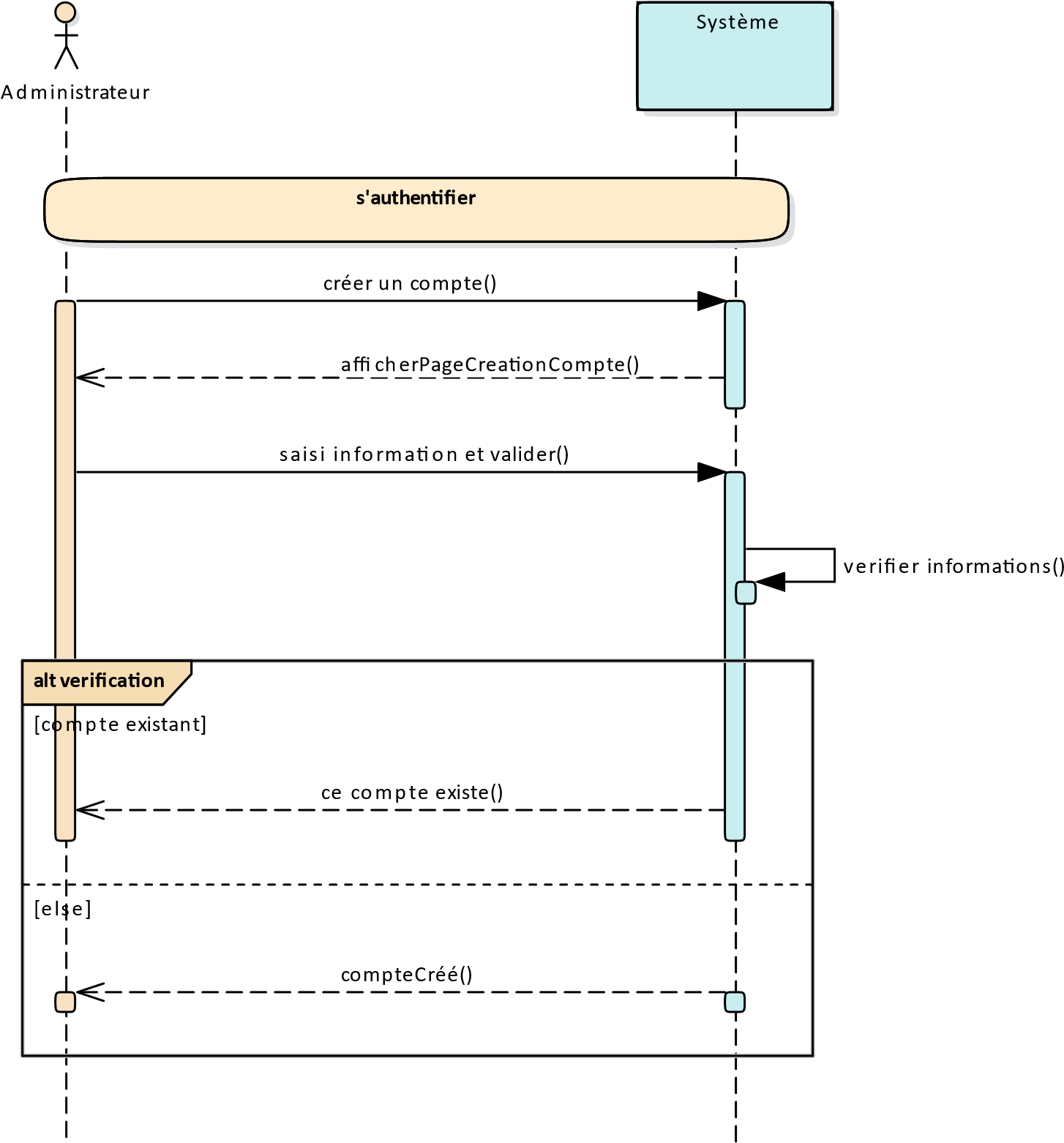


Figure 9: diagramme de séquence "créer un compte"

* Modifier un compte

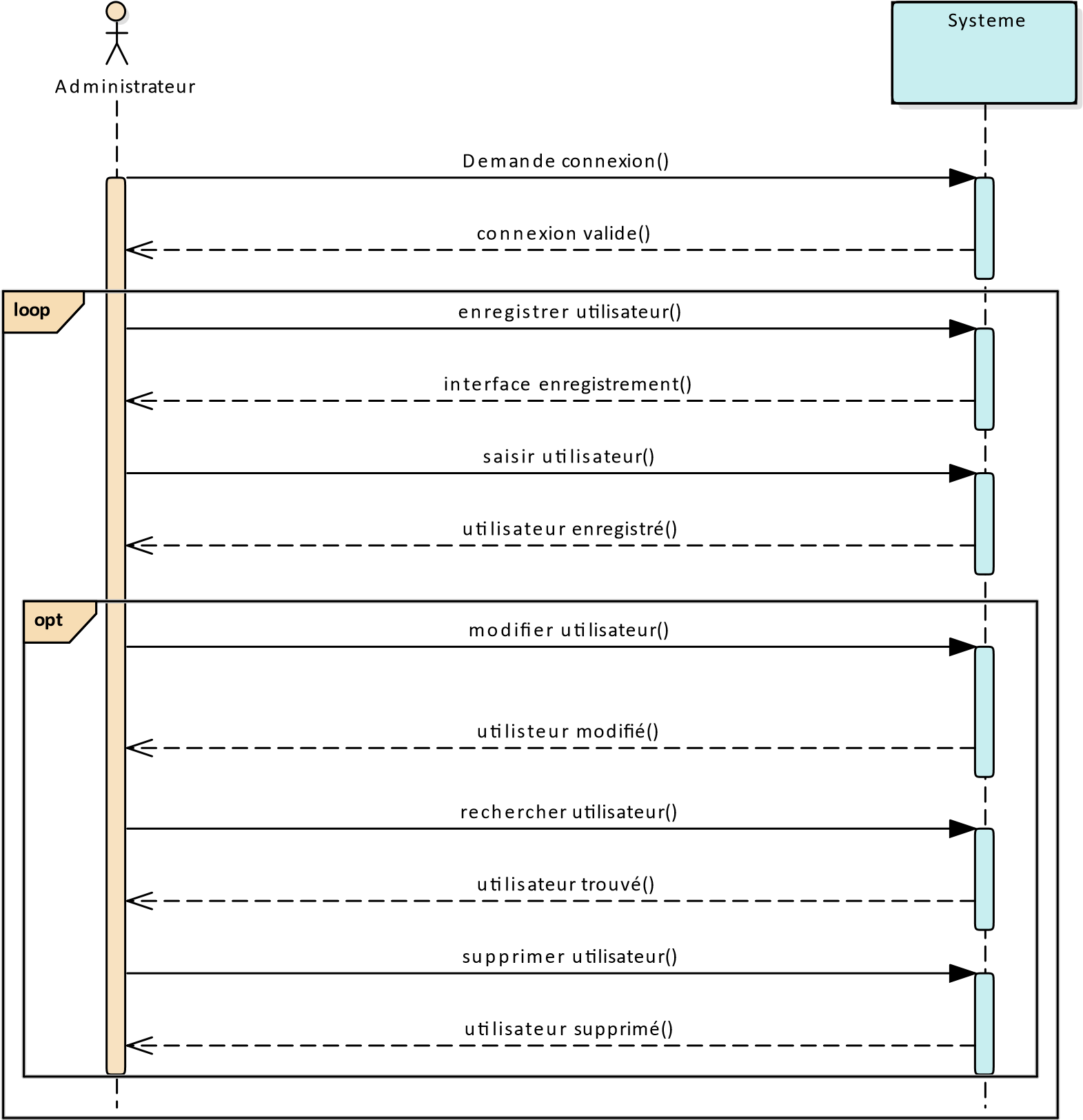


Figure 10: diagramme de séquence "modifier un compte"

* Consulter information

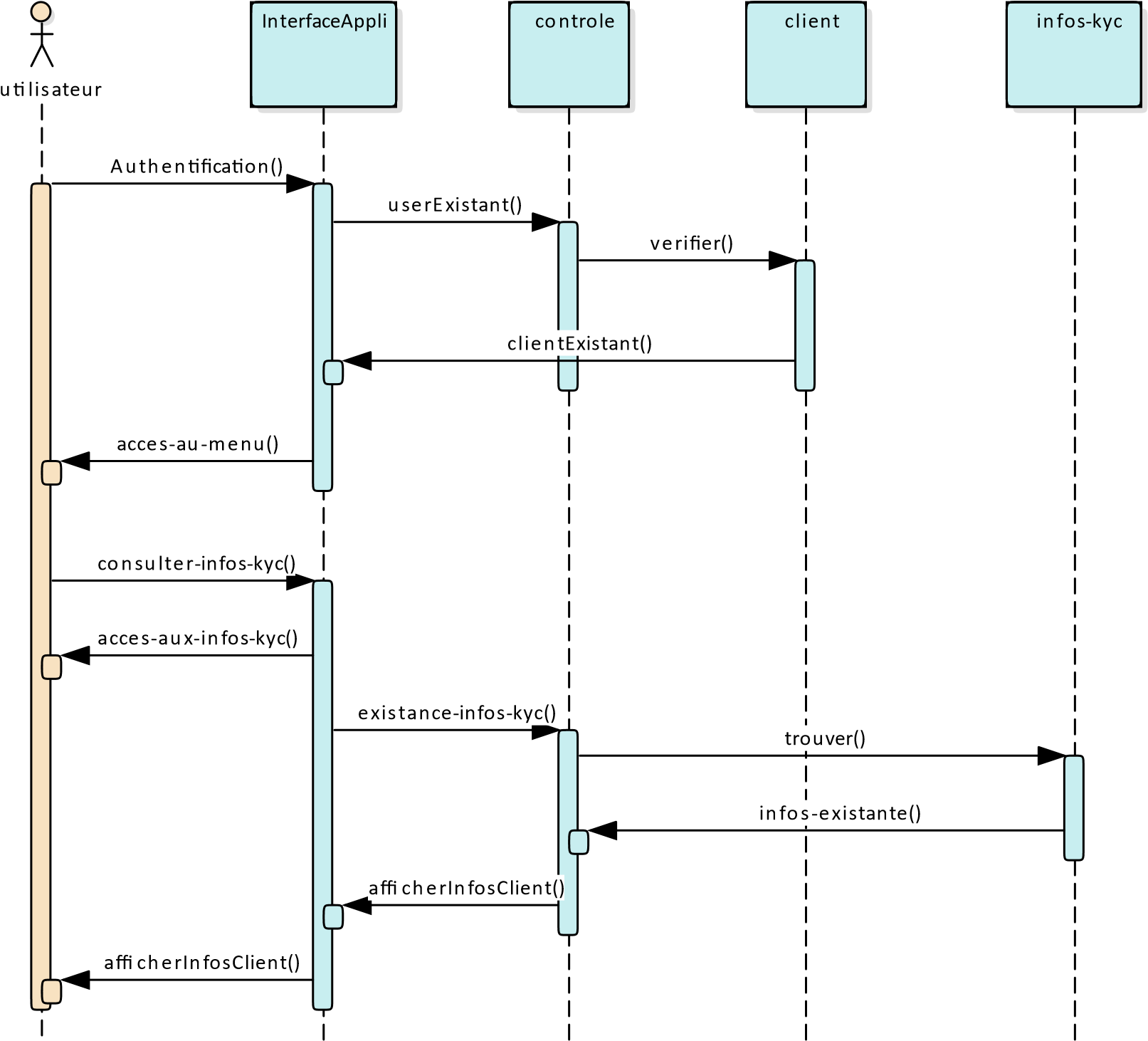


Figure 11: diagramme de séquence "consulter information"

* Effectuer une correction sur les données du client

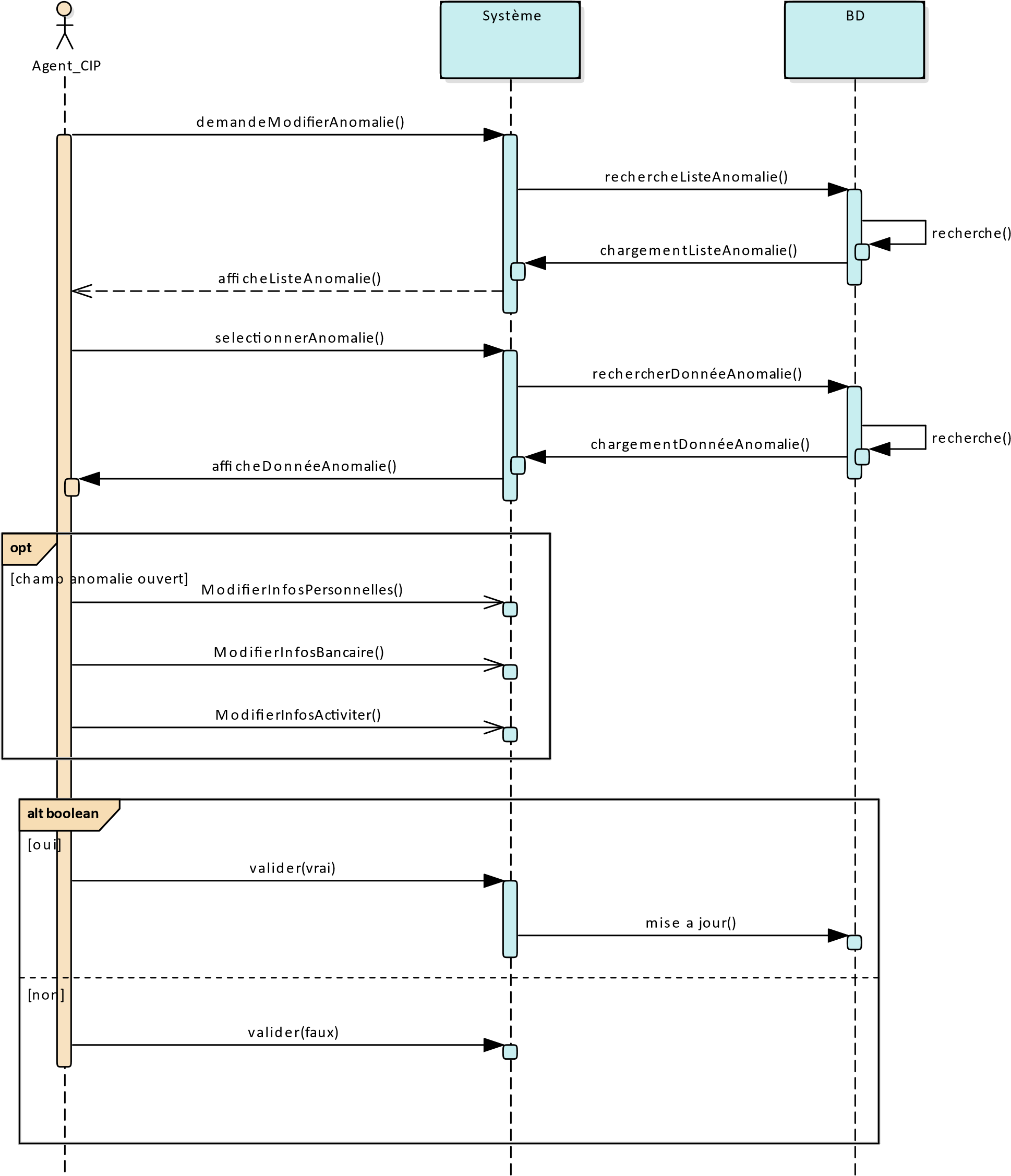


Figure 12: diagramme de séquence "effectuer correction"

* Paramétrer un champ

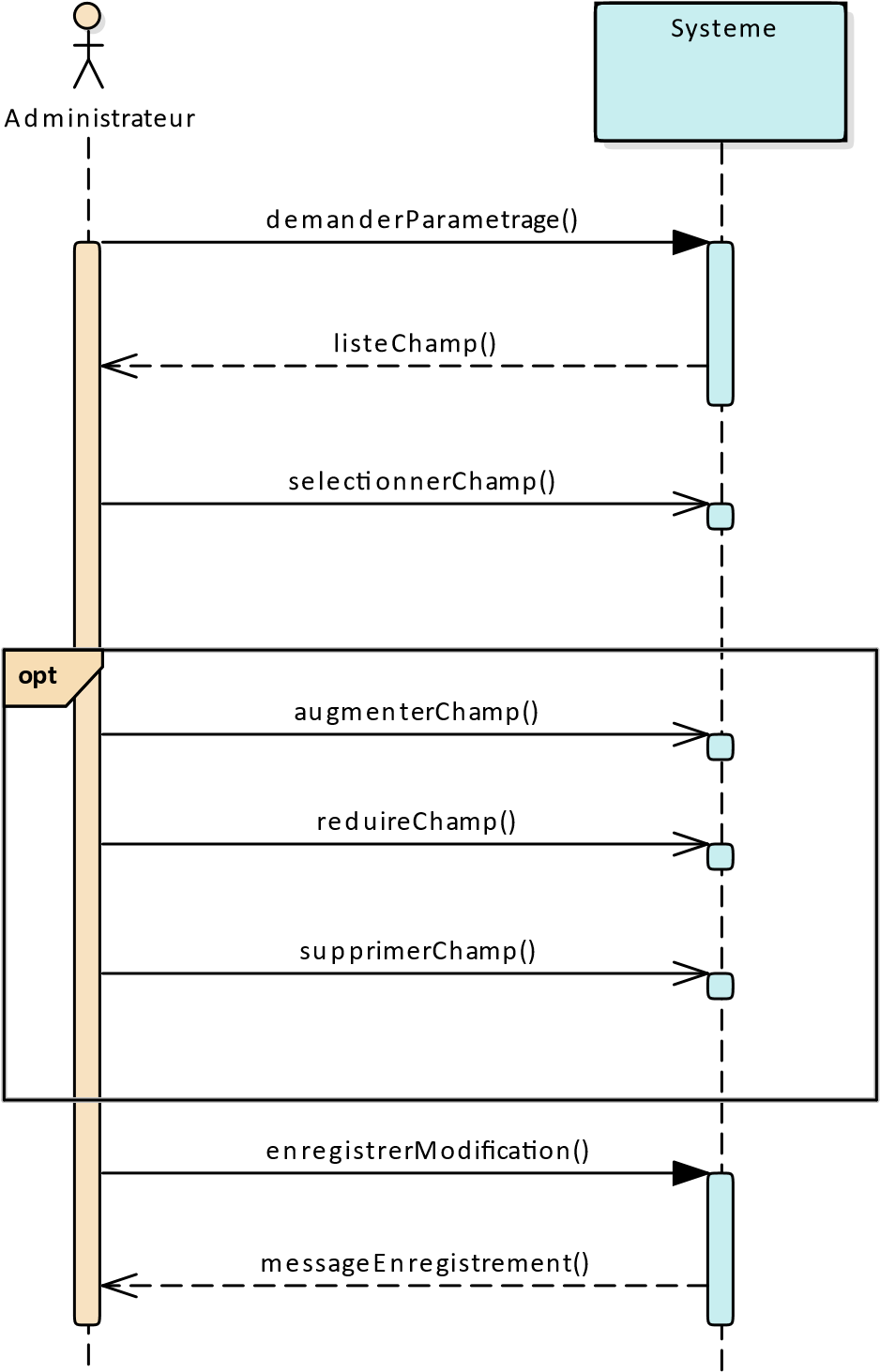


Figure 13: diagramme de séquence "paramétrer champ"

* Générer un reporting

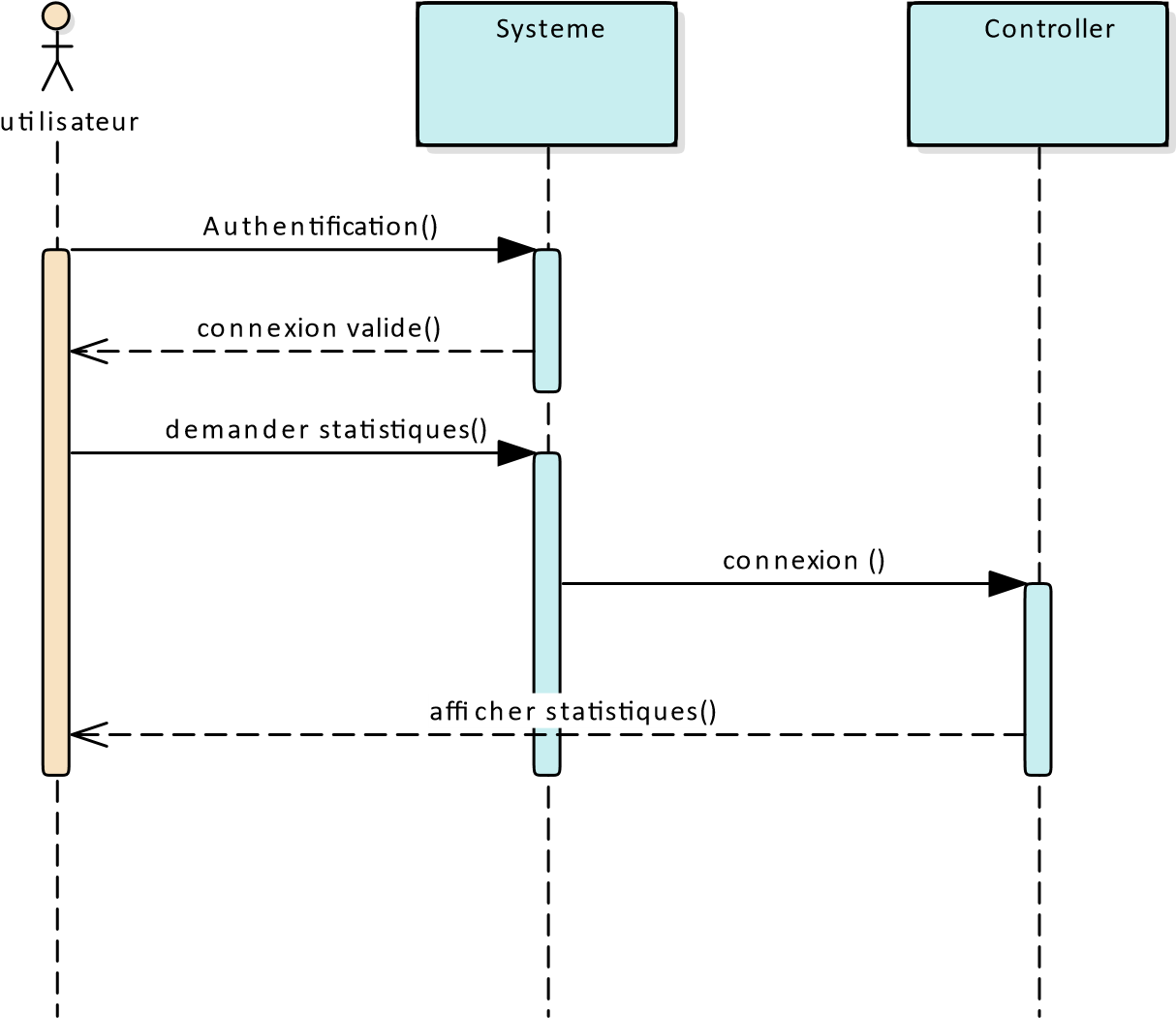


Figure 14: diagramme de séquence "générer reporting

## Collaboration

* Objectifs :
  + Ce type de diagramme montre des interactions entre objets (instances de classes et acteurs). Il s'intéresse à la structure de collaboration entre objets (séquencement, itération, concurrence, etc.)
  + Il permet de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.
  + Ce diagramme est équivalent au [diagramme de séquences](http://www.iro.umontreal.ca/~dift6825/diagrammeSequences.htm). Cependant, l'aspect temporel n'apparaît pas, mais l'aspect chronologique est présent.
* Éléments de base :
  + Un tel diagramme est formé d'un sous-ensemble du [diagramme de classes](http://www.iro.umontreal.ca/~dift6825/diagrammeClasses.htm) et les messages échangés entre objets à travers les liens.
  + Pour représenter l'aspect chronologique (ordre d'envoi des messages), il est important de numéroter les messages.
  + Le sens des messages nous permet de déterminer l'envoyeur et le receveur.

  Dans le cadre de notre travail, nous allons représenter les diagrammes de collaboration ayant un impact important dans le fonctionnement de notre application. Ces différents diagrammes concerneront en particulier les cas d’utilisation s’authentifier, consulter les données, effectuer une correction sur les données et enfin générer un reporting.

* S’authentifier

**persone**

**:**

**Interface utilisateur**

**:**

**compte utilisateur**

**:**

**:**

**User**

)

. saisir (login,mdp

1

)

4

. reponse(result+droit d'acces

5

. reponse retourné

. resultat affiché

6

3

. rechercher (login,mdp

)

2

. envoyer(login,mdp

)

Figure 15: diagramme de collaboration" s'authentifier"

* Consulter les données clientèles

**Banque de données**

**User**

**Peter : User**

**Client**

**C : Client**

1:

getUser

()

getClient

()

1.1:

Figure 16: diagramme de collaboration "consulter données clientèles"

* Modifier les données clientèles

**Banque de données**

**Client**

**C : Client**

**Jean : AgentCIP**

**AgentCIP**

1:

SetClient

()

1.1:

SetUser

()

Figure 17: diagramme collaboration" modifier données clientèles"

* Générer reporting

**:**

**personne**

**:**

**Serveur Application**

**client**

**:**

**:**

**serveur BD**

1

. GenererReporting(condition

)

.ReportingPublié

6

3

.recupererInfos

(

condition

)

4

. statistiqueAnomalie

2

.rechercherIndicateur

()

5

. resultat sur anomalie

Figure 18: diagramme de collaboration "générer reporting"

## **Analyse du Système**

## Diagramme de classe

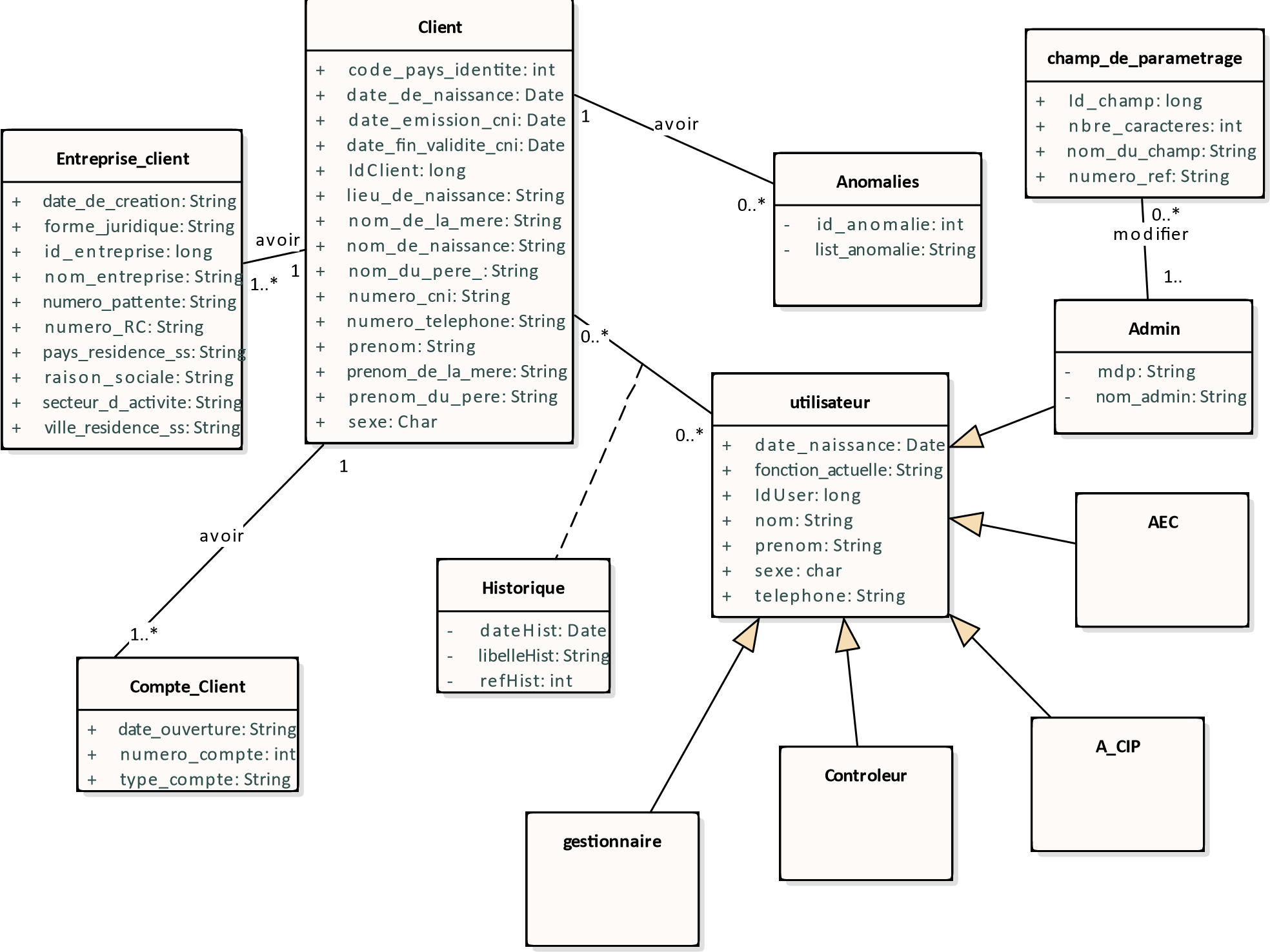
La conception objet demande principalement une description structurelle et statistique du système à réaliser sous un ensemble de classes logicielles, éventuellement regroupées en packages. Les meilleures classes candidates sont celles issues d’une analyse du domaine (souvent appelée analyse métier), c’est-à-dire des concepts manipulés par les experts du domaine.

Il est question de créer une représentation visuelle des objets du monde réel dans un domaine donné. Si l’on emploie la notation UML, il s’agit d’un diagramme dans lesquels ont fait figurer les éléments suivants :

* Les classes conceptuelles ou les objets du domaine. Une classe représente la description abstraite d’un ensemble d’objets possédant les mêmes caractéristiques. Exemples : rôle, utilisateur etc.
* Les associations représentant les relations sémantiques durables entre les classes.
* Les attributs des classes conceptuelles ; un attribut représente un type d’information contenu dans une classe. Exemples : nom magasinier, adresse administrateur
* Les méthodes : ce sont les différentes opérations possibles d’une classe exemples : se connecter, créer un compte etc.

Le diagramme de classe correspondant à notre analyse fonctionnelle suivant la méthode 2TUP est alors le suivant :

Figure 19: diagramme de classe



## Diagramme d’états transition

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode).

Ils spécifient habituellement le comportement d'une instance de classeur (classe ou composant), mais parfois aussi le comportement interne d'autres éléments tels que les cas d'utilisation, les sous-systèmes, les méthodes.

Le diagramme d'états-transitions est le seul diagramme, de la norme UML, à offrir une vision complète et non ambiguë de l'ensemble des comportements de l'élément auquel il est attaché. En effet, un diagramme d'interaction n'offre qu'une vue partielle correspondant à un scénario sans spécifier comment les différents scénarii interagissent entre eux.

La vision globale du système n'apparaît pas sur ce type de diagrammes puisqu'ils ne s'intéressent qu'à un seul élément du système indépendamment de son environnement.

Concrètement, un diagramme d'états-transitions est un graphe qui représente un *automate à états finis*, c'est-à-dire une machine dont le comportement des sorties ne dépend pas seulement de l'état de ses entrées, mais aussi d'un historique des sollicitations passées.

Les diagrammes d’état-transitions ci-dessous sont les ceux correspondant au scénario de création et modification de compte puis celle de la correction d’anomalie.

* Créer et modifier un compte

Initial

**En Attente**

**Validé**

**Privilégié**

*(*

*from corrigerAnomalie*

*)*

**mise à jour**

ExitPoint

AjouterRôleUtilisateur()

désactiverUtilisateur

AjouterUtilisateur()

validation

Annulation

ModifierUtilisateur()

Figure 20: diagramme d'état-transition créer et modifier un compte

* Corriger anomalie :

**infos client affiché**

Initial

**infos à modifier**

**modifié**

Final

**client trouvé**

Final

**Chercher client**

**validé**

listeModification()

client introuvable

]

[

Modifier Anomalie

validation

Figure 21: diagramme d'état-transition corriger anomalie

## **Capture des besoins techniques**

La capture des besoins techniques recense toutes les contraintes et les choix dimensionnant la conception du système.

Les outils et les matériels sélectionnés ainsi que la prise en charges des contraintes d’intégration conditionnant généralement des prérequis d’architecture générale.

## Description des spécifications ergonomiques

L’application **Help KYC** aura un menu principal de profondeur 2. Afin de faire valoir l’utilisabilité de l’outil, Nous aurons des écritures noires sur fond blanc de code **rgb(255, 255, 250)** dans le cadre général. Au niveau du menu nous aurons le vert de code **rgb(50, 168, 115) ** et rouge bordeaux de code **rgb(199, 0, 20) **. Ces couleurs ont été choisies en vue de permettre aux utilisateurs d’avoir un aspect visuel agréable pendant l’utilisation de l’outil.

Le temps maximal de chargement d’une page sera de 2 secondes. L’interface utilisateur sera très simple à utiliser. Cette interface sera adaptée tant pour les informaticiens que pour les non informaticiens. Il ne sera pas obligatoire d’avoir une connaissance évoluée sur l’outil informatique pour pouvoir utiliser notre logiciel.

## Cas d’utilisation technique

La capture des spécifications se présente comme suit :

* Capture des spécifications logicielles ;
* Capture des spécifications liés à la configuration matérielle.

La figure ci-contre est un diagramme de cas d’utilisation pour les besoins techniques :

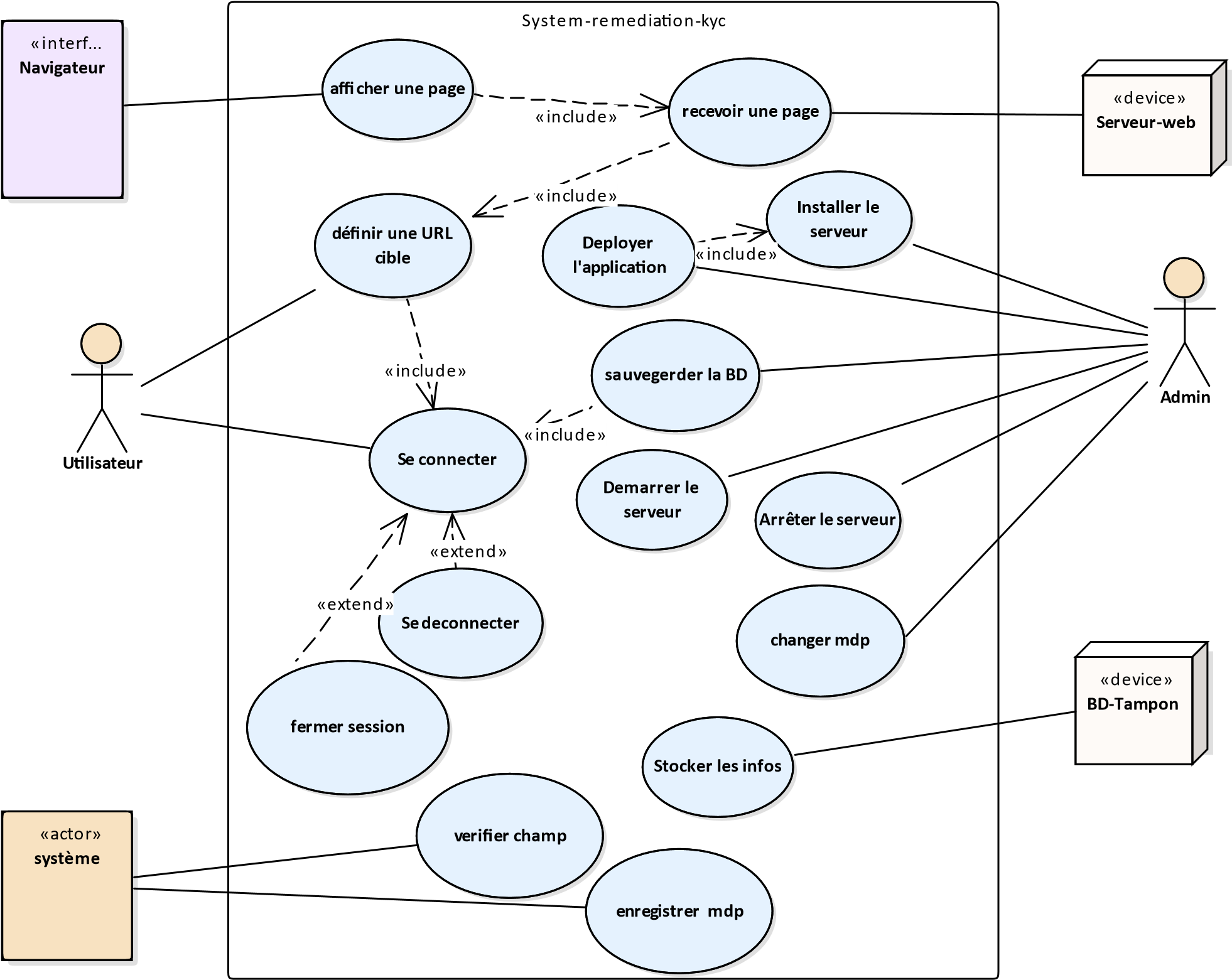


Figure 22: diagramme de cas d'utilisation besoin technique

## **Conception générique**

Il s’agit du découpage en composants nécessaires à la construction de l’architecture technique. Cette étape permet de minimiser l’incapacité de l’architecture technique à répondre aux contraintes opérationnelles. Le diagramme qui ressort suite à cette phase est le diagramme de déploiement.

Un diagramme de déploiement modèle l’architecture de temps d’exécution d’un système. Il affiche la configuration des éléments matériels (nœuds) et affiche comment des éléments logiciels et des artefacts sont mappés sur ces nœuds.

La figure ci-dessous est la représentation du diagramme de déploiement de la partie technique de notre logiciel.

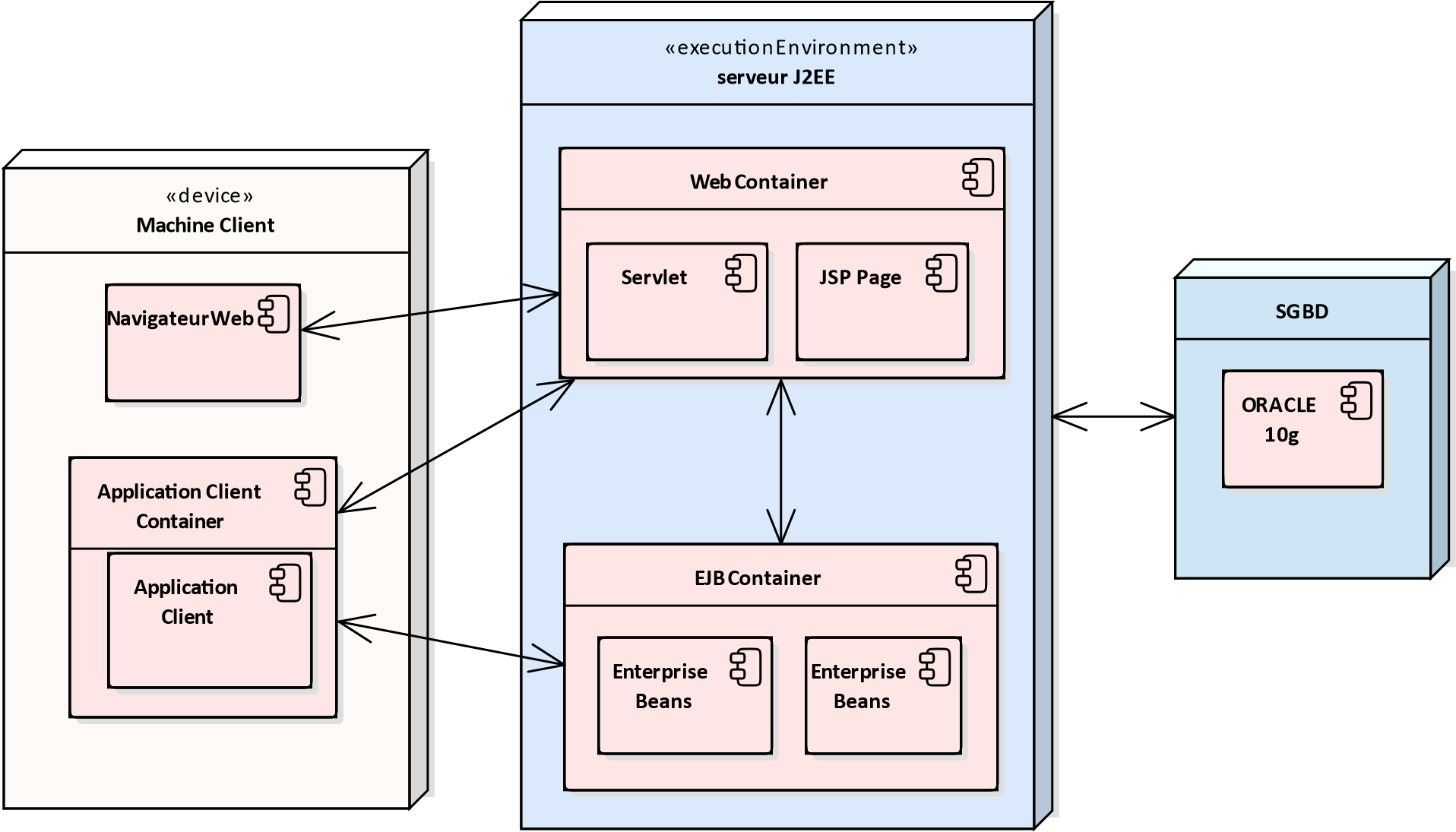


Figure 23: diagramme de déploiement technique

## **Conception préliminaire et détaillé**

Cette phase est la fusion de la Branche fonctionnelle et technique et mène à la conception applicative et à la solution adaptée aux besoins des utilisateurs. Elle concerne les étapes de la conception préliminaire, la conception détaillée, le codage et les tests puis l'étape de recette. On y construit les classes, les vues, les interfaces, les tables et les méthodes qui vont donner une image **« prête à coder »** de la solution. Conception préliminaire et détaillée

## **Conception préliminaire**

Cette étape est la plus importante du processus 2TUP vu qu’elle représente le cœur. En effet, dans cette étape nous allons enfin quitter les deux branche droite et gauche afin de faire la fusion entre les deux études technique et fonctionnelle.

Cependant, nous allons développer les catégories d’analyse en couches logicielles conformément au modèle retenu de la solution technique tout en restant le plus indépendant possible des outils de développement.

Afin d’aboutir à cela, nous allons s’organiser selon les diagrammes suivants :

* Diagrammes de composant

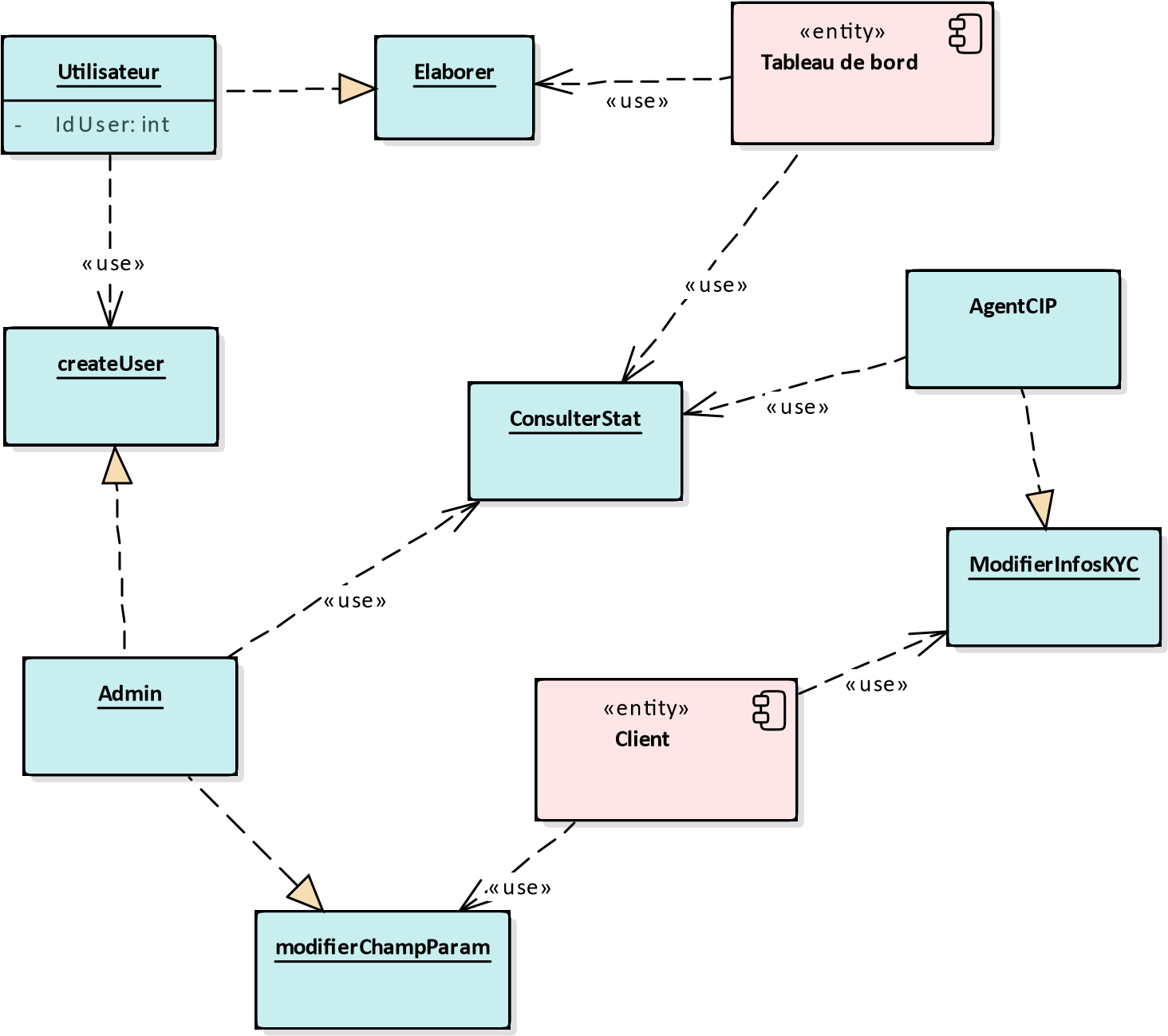


Figure 24: diagramme de composants

* Diagrammes de déploiement

## **Conception détaillée**

Conception de chaque fonctionnalité. Etape de codage : phase de programmation de ces fonctionnalités, avec des tests au fur et à mesure. Etape de recette : phase de validation des fonctions du système développé

Les différents diagrammes qui le concernent sont :

* Diagramme de classe système
* Diagrammes de séquence système

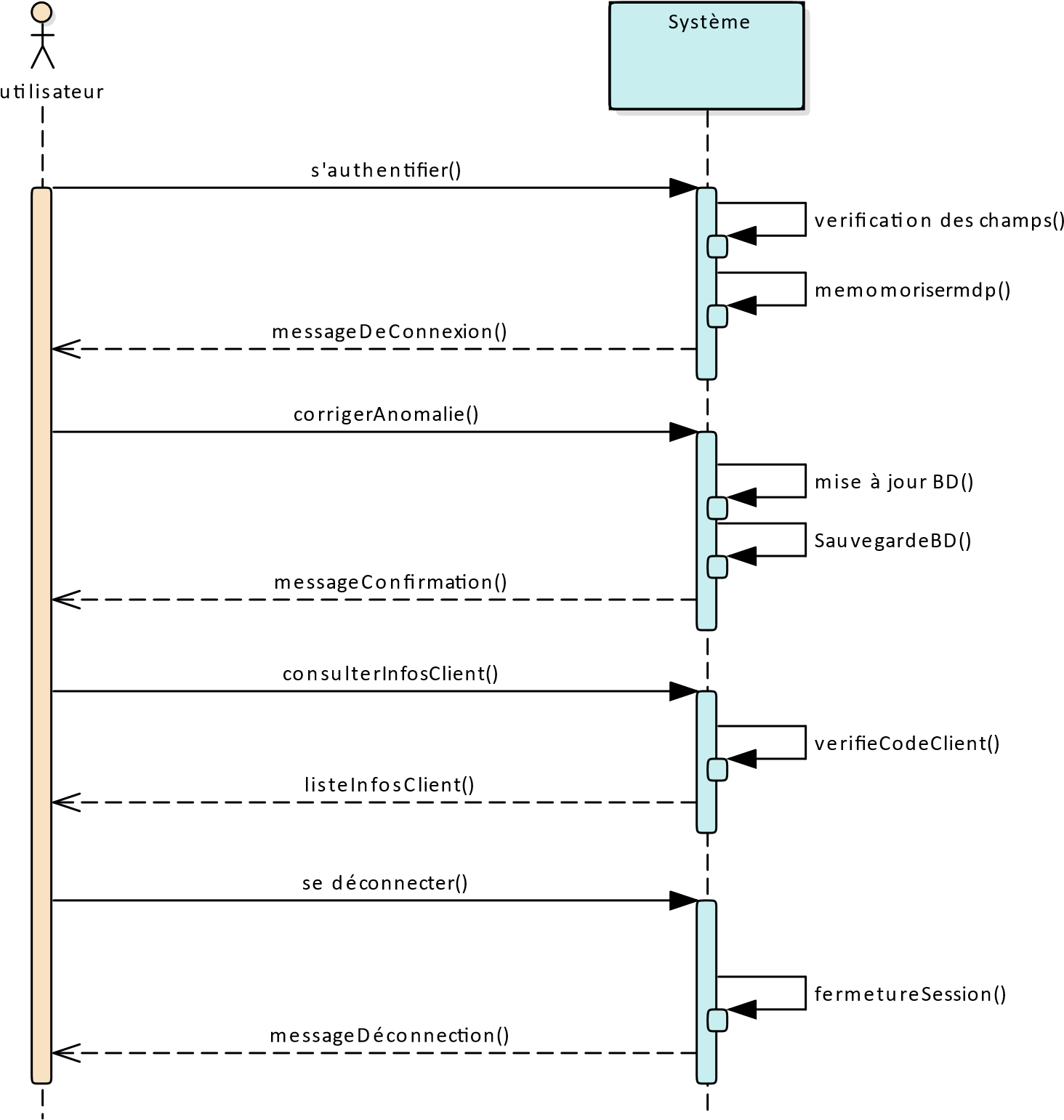


Figure 25: diagramme de séquence système

* Diagramme de collaboration

**user**

**)**

**interface (navigateur**

**:**

**:**

**client**

**:**

**list\_Client**

**clt:Client**

**BD**

**:**

**statistique**

**champ\_parametrage**

accéderDonnéesClient()

emettre\_reporting ()

consulter\_Client(code)

s'authentifier(login,mdp)

modifier\_client(code)

modifier\_champ()

mise\_a\_jourBD(client)

Figure 26: diagramme de collaboration système

* Diagramme d’état

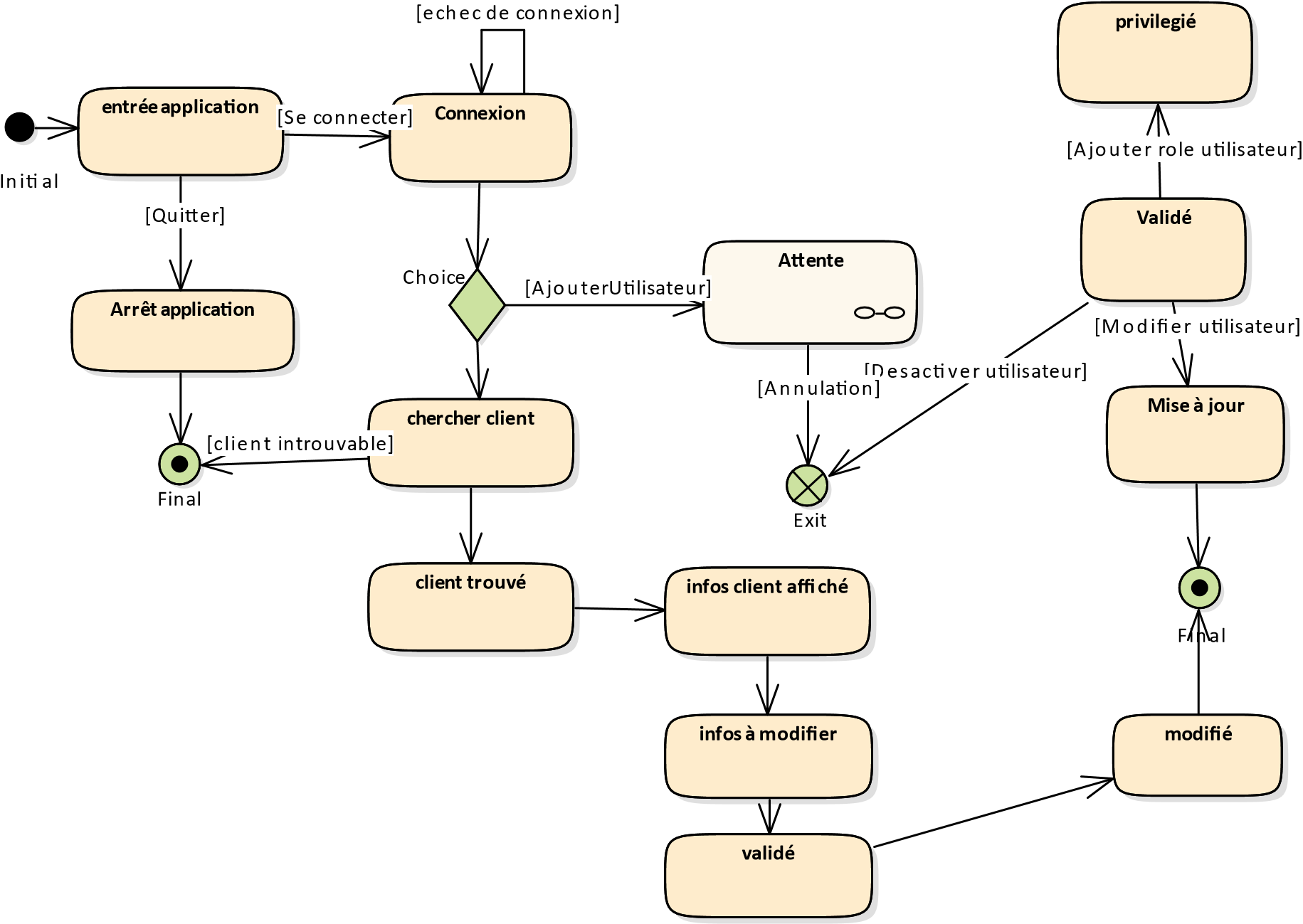


Figure 27: diagramme d’état-transition Système

* Diagramme d’activité

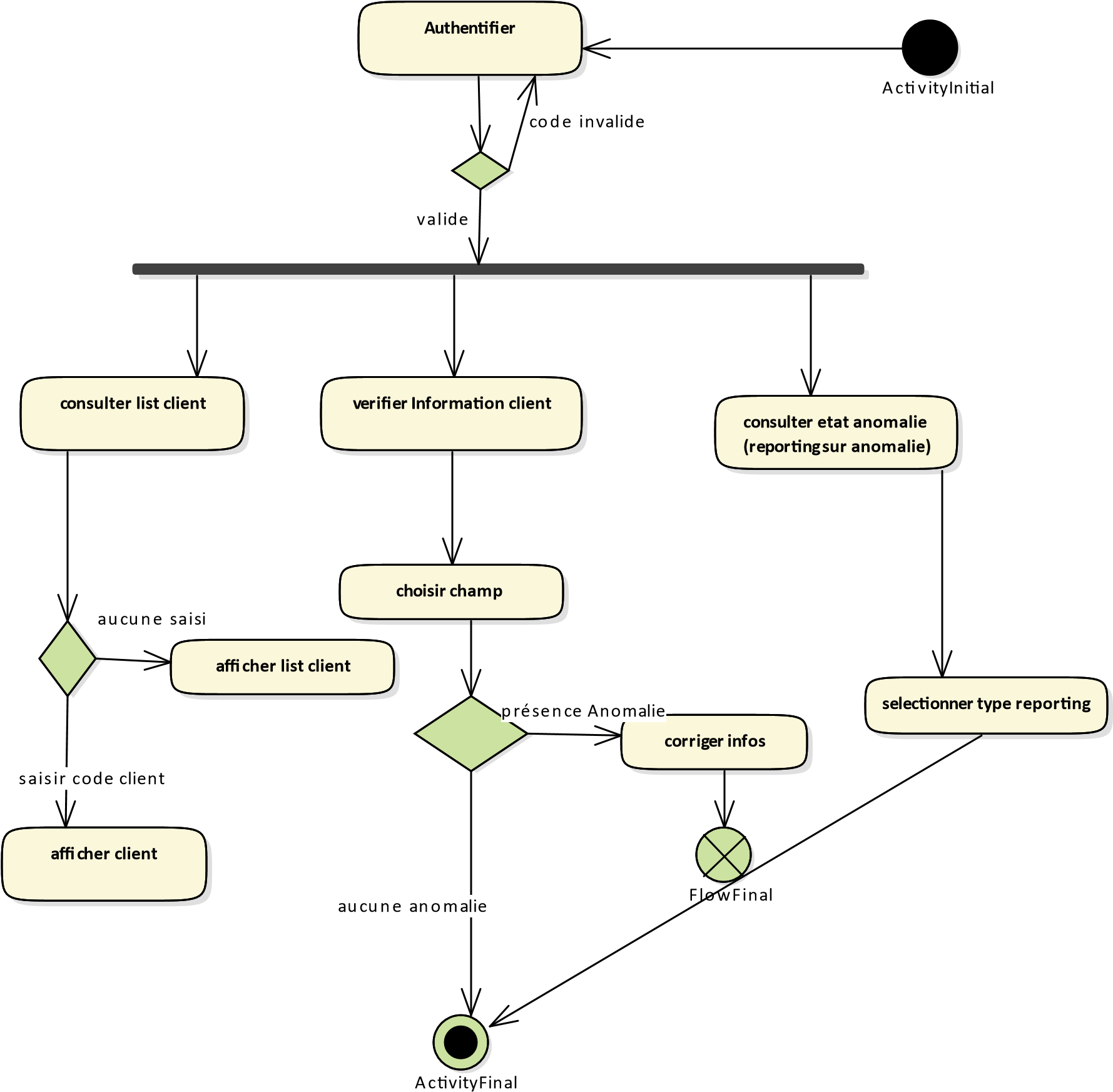


Figure 28: diagramme d'activité

* Diagramme de composants

# **CONCLUSION**

Parvenue au terme de notre travail où il était question d’établir un cahier de conception devant nous servir à faciliter la réalisation de notre application d’aide à la remédiation KYC, nous avons eu à modéliser notre système suivant la logique de la méthode 2TUP qui nous a permis de ressortir les différents diagrammes UML suivant les deux branches à savoir la branche fonctionnelle et technique. Toutefois, ces derniers nous aiderons dans la phase de réalisation de notre logiciel qui nous permettra de prendre en compte toutes les spécifications de notre application.