الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
République Algérienne Démocratique et Populaire
وزارة التعليـم العالـي والبحـث العلمـي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf :……

Centre Universitaire
Abdelhafid Boussouf Mila

Institut des Sciences et Technologie **Département de Mathématiques et Informatique**

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Conception et Implémentation d'une Base de Données Distribuée

Préparé par : ALI MOUSSA Ikram
MEDJANI Sarra

Devant le jury

Mme. BENABDERRAHMANE Fatiha.	**MAA**	**C.U.Abd Elhafid Boussouf**	**Président**
M. SELMANE Samir.	**MAB**	**C.U.Abd Elhafid Boussouf**	**Rapporteur**
M. BESSOUF Hakim.	**MAB**	**C.U.Abd Elhafid Boussouf**	**Examinateur**

Année Universitaire : 2016/2017

Remerciement

Nous remercions dieu tout puissant pour nous avoir offert la force et la patience durant toutes ces années.

Nous tenons à exprimer notre profonde gratitude à notre encadreur :

Mr. Salmane samir, pour son soutien constant, son aide précieuse et ses conseils attentifs durant tout le projet. Et qui nous ont assuré l'environnement adéquat afin de réaliser notre travail.

Nous remercierons aussi les enseignants du département de l'informatique qui tout au long des années d'études nous ont transmit leur savoir sans réserves, et tous ceux qui nous ont apporté une aide pour la réalisation de ce projet.

Sans oublier bien-sûre tous les amis et collègues d'études pour leur enjouement et soutient spécialement Benmakhelouf Riyadh et Graiche Younes .

Ikram & Sara.

Dédicace

Je dédie ce mémoire à :

Ma mère et mon père pour l'éducation qu'ils m'ont prodiguée ; avec tous les moyens et au prix de toutes les sacrifices qu'ils ont consentis à mon égard, pour le sens du devoir qu'ils m'ont enseigné depuis mon enfance

A mes chères frères Yacine et Aymen,

A mes chères sœurs Nouzha, Soumia,Amina et Rahma,

A mon neveu Mohemed Anes, et mes nièces Nesrine et Nermine, ces petits anges qui nous comble de joie,

A mon binôme et mon amie Sara et toute sa famille,

A mes enseignants,

A tous mes amis(es), mes collègues de travail & mes collègues de 2eme année Master STIC spécialement B. riyad et G.youness .

Ikram

Dédicace

Au Début et avant tout, je veux remercier le DIEU qui ma donnée le courage à faire et finir ce modeste travail.

A mes très chers parents «MOHAMED» et «SAMIRA» qui n'ont jamais cessé de m'encourager que dieu les protège

A mes très chers frères« Mourad» et « Ishaq»

A mes très chères sœurs « Sabrina» et « Dounya »

A ma collègue de projet *et mon amie* « Ikram » *et toute sa famille*

à ma tante « SAMIA » qui ma aider pendant tous mes études

Dédier spécial a mon collègue B.riyadh Qui était avec nous et nous a aidés

A toute ma famille « Medjani»

A mes enseignants

A tous mes amis surtout khawla ,ibtissem ,asma

A mes petites layane , rahil ,aya

A toute ma famille de STIC 2, mes sœurs et frères,

promotion 2016/2017, pour leur aide et tous ces moments

inoubliables que nous avons passés ensemble,

Je dédie ce mémoire

A la fin, je remercie tous ceux qui ont aidé de près ou de Loin à réaliser notre travail.

Sara

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Le monde de l'informatique évolue très rapidement, alors que son but initial, était d'offrir des services satisfaisants, du point de vue vitesse d'exécution des tâches. Actuellement, de nouveaux besoins sont apparus notamment pour les grandes entreprises, celles-ci souhaitent stocker et échanger des informations qui sont géographiquement éloignées. La collecte et le traitement d'une grande quantité d'informations dispersées est une tâche très délicate. La solution qui s'impose est de distribuer les données et les organiser dans des bases de données sur différents sites de stockage. L'ensemble de ces sites constitue un système de bases de données réparties offrant la possibilité aux utilisateurs de manipuler les différentes bases via un réseau de manière transparente, comme dans une base de données unique.

Durant le stage pratique que nous avons effectués dans le cadre du projet de fin d'étude Master 2 qui s'est déroulé au niveau de la Commune de Mila (APC) précisément dans le bureau de la carte grise, nous avons passé un temps considérable au sein de la commune, cherchant et discutant avec le personnel des différents services. Finalement nous avons constaté que le suivi de mouvements des dossiers de la carte grise au niveau de cette commune est une opération réalisée manuellement concernant la vente externe, ce qui exerce une contrainte multiforme sur le fonctionnement de la commune, vu le grand nombre de dossiers à traiter et le temps qui se déroule pour extraire une carte grise d'une voiture vendue hors wilaya.

L'objectif de ce travail est d'essayer de résoudre les problèmes posés par le système actuel de gestion de la carte grise. Pour cela, nous avons conçu et mis en œuvre une base de données répartie sous *Oracle* assurant une gestion efficace et transparente de la carte grise.

Notre mémoire est structuré en deux parties essentielles. La première partie présente le domaine d'étude qui est en fait une synthèse de la documentation faite autour du domaine d'étude et qui contient des définitions et des concepts fondamentaux des bases de données répartie et leur conception.

La deuxième partie du mémoire est consacrée à l'étude de cas. Elle constitue l'essentiel du travail d'ingénierie des systèmes d'informations que nous avons effectué. Elle s'articule autour des phases essentielles de la méthode 2TUP, et qui sont :

Le chapitre 01 : « L'étude préliminaire» dans cette phase, nous présentons l'entreprise où nous avons effectué le stage et définissons les notions importantes qui la concernent ensuite nous allons élaborer le cahier des charges qui contient une représentation plus formelles des activités de capture des besoins fonctionnels et de capture des besoins techniques.

Le chapitre 02 : « Capture des besoins » Ce chapitre comporte deux étapes : la capture des besoins fonctionnels et celle des besoins techniques. La phase de capture des besoins fonctionnels formalise et détaille ce qui a été ébauché au cours de l'étude préliminaire, en donnant une description textuelle et une autre graphique pour chaque cas d'utilisation. La capture des besoins techniques couvre, par complémentarité avec celle des besoins fonctionnels, toutes les contraintes qui ne traitent ni de la description du métier des utilisateurs, ni de la description applicative.

Le chapitre 0 3 : « Analyse » Ce chapitre comporte les étapes de découpage en catégories, de développement du modèle statique et développement du modèle dynamique. Le découpage en catégories consiste de découper le modèle UML en blocs logiques les plus indépendants possibles. Le développement du modèle statique va nous permettre d'illustrer les principales constructions du diagramme de classes UML durant l'étape d'analyse. Le développement du modèle dynamique va nous permettre d'illustrer comment décrire des scénarios mettant en jeu un ensemble d'objets échangeant des messages.

Le chapitre 04 : « Conception » Ce chapitre comporte la conception générique, la conception préliminaire et la conception détaillée. Dans la phase de conception préliminaire on effectue la fusion des études fonctionnelles et techniques. La conception détaillée consiste à construire et à documenter précisément les classes, les interfaces, les tables et les méthodes qui constituent le codage de la solution.

Le chapitre 05 : « Implémentation » Dans ce chapitre on donne une description de l'application, les technologies de programmation utilisées, ainsi que les interfaces graphiques de l'application.

Conclusion générale : La conclusion générale résume les résultats de notre travail, et présente les perspectives que nous souhaitons réaliser dans le futur.

Partie I

présentation du domaine d’étude

Chapitre I

Généralités sur les bases de données réparties

1. Introduction

Le développement des techniques informatiques depuis ces dernières années a permis d'appliquer les outils informatiques dans l'organisation des entreprises. Vu, l'immense volume de données manipulées par ces dernières, la puissance des micro-ordinateurs, les performances des réseaux et la baisse considérable des coûts du matériel informatique ont permis l'apparition d'une nouvelle approche afin de remédier aux difficultés causées par la centralisation des données, et ce en répartissant les ressources informatiques tout en préservant leur cohérence.

Les bases de données réparties sont un moyen performant pour diminuer les problèmes provoqués par l'approche centralisée, mais ne restent pas sans failles.

2. Principe des bases de données réparties

Les bases de données distribuées permettent aujourd'hui de muter notre mode de travail traditionnellement centralisé en un mode décentralisé. Cette technologie est certainement l'un des plus importants développements du domaine des systèmes de gestion de bases de données.

2.1. Définition

Une base de données répartie (BDR) est une base de données dont différentes parties sont stockées sur des sites, généralement géographiquement distants, reliés par un réseau. La réunion de ces parties forme la base de données répartie.

Un système de bases de données réparti ne doit donc en aucun cas être confondu avec un système dans lequel les bases de données sont accessibles à distance (selon le principe client-serveur). Il ne doit pas non plus être confondu avec un système multi base. Dans ce dernier cas, chaque utilisateur accède à différentes bases de données en spécifiant leur nom et adresse, et le système se comporte alors simplement comme un serveur de BD et n'apporte aucune fonctionnalité particulière à la répartition.

Au contraire, un système de bases de données réparti est suffisamment complet pour décharger les utilisateurs de tous les problèmes de concurrence, de fiabilité, d'optimisation de requêtes ou de transaction sur des données gérées par différents SGBD sur plusieurs sites [1].

A Titre d'exemple, une banque peut posséder des agences à Mila et à Constantine. Dans une BD centralisée, le siège social de la banque va gérer tous les comptes des clients et les agences devraient communiquer avec le siège social pour avoir accès aux données.

Dans une BD répartie, les informations sur les comptes sont distribuées dans les agences et celles-ci sont interconnectées (entièrement ou partiellement) afin qu'elles puissent avoir accès aux données externes (Figure 1). Cependant, la répartition de la base de données bancaire est invisible aux agences en tant qu'utilisateurs, et la seule conséquence directe pour elles est que l'accès à certaines données est beaucoup plus rapide et très fiable.

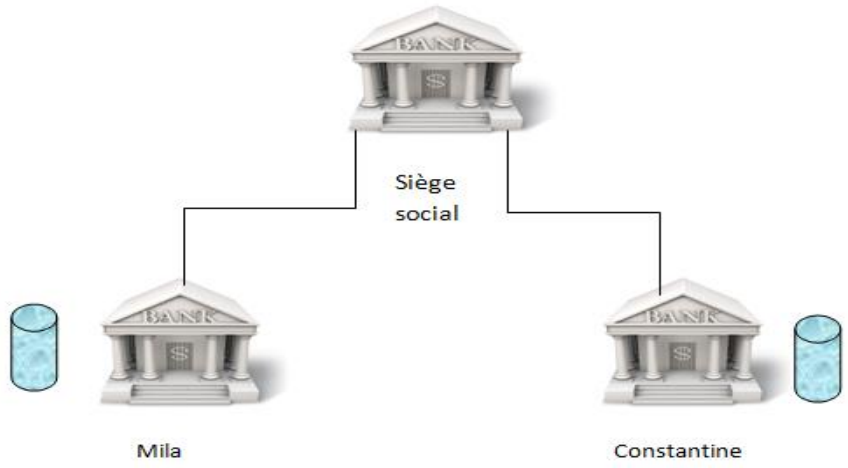

Figure I-1 : Exemple de BD Répartie.

2.2. Concepts des bases de données réparties

Une BDR reprend les mêmes principes que ceux d'une BD centralisée mais en étendant les techniques existantes ou en proposant certains concepts nouveaux qui sont particuliers à la répartition des données.

2.2.1. Schéma global

Le schéma global permet de définir l'ensemble des types de données de la base. Il ignore les concepts d'implémentation, chaque base locale en implémente une partie.

2.2.2. Schéma local

Une base de données locale comporte un schéma géré par le SGBD local.

Dans une BD répartie, chaque base locale rend visible toute ou une partie de la base aux sites clients.

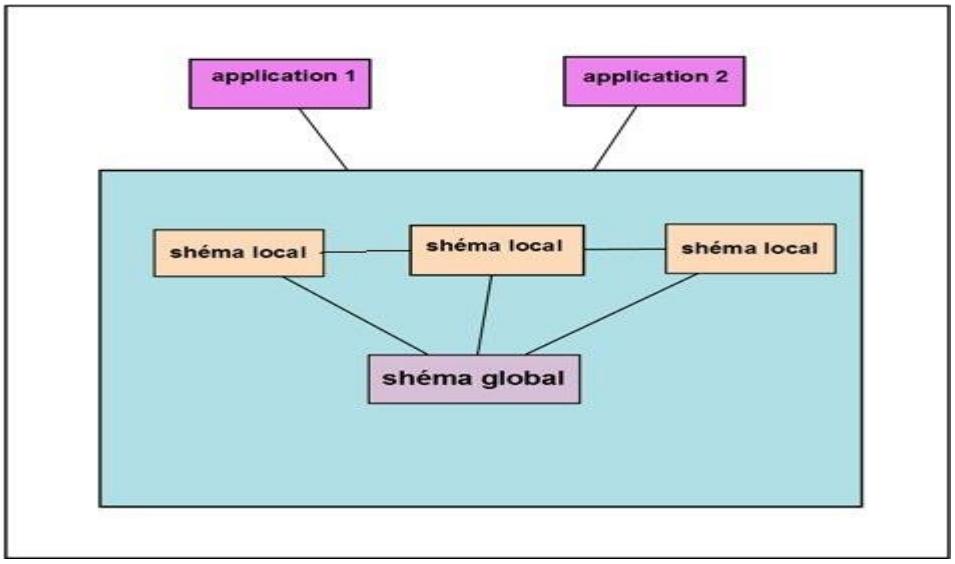

Figure I-2 : Schéma globale et local.

2.2.3. Schéma externe

Le niveau externe décrit les données sous forme de vues, chacune d'elles étant adaptée à une classe particulière d'utilisateurs ; un schéma externe, élaboré à partir du schéma conceptuel, peut naturellement mixer des données stockées dans différentes bases.

2.2.4. Schéma conceptuel

Où les données sont représentées sans prendre en compte les contraintes techniques ou de mise en forme ; toutes les données sont décrites dans ce schéma en utilisant un modèle de données, indépendamment de leur localisation dans le système réparti.

2.2.5. Schéma interne

Le niveau interne global n'a pas d'existence réelle mais fait place à des schémas internes locaux, répartis sur différents sites. Ces schémas correspondent à la description de l'organisation physique de la base, notamment la spécification de la fragmentation des données et la localisation de ces fragments. [2]

3. Avantages

Les bases de données réparties ont une architecture plus adaptée à l'organisation des entreprises décentralisées. [3]

- **Plus de fiabilité** : les bases de données réparties ont souvent des données répliquées. La panne d'un site n'est pas très importante pour l'utilisateur, qui s'adressera à un autre site.
- **Meilleures performances** : réduire le trafic sur le réseau est une possibilité d'accroître les performances. Le but de la répartition des données est de les rapprocher de l'endroit où elles sont accédées. Répartir une base de données sur plusieurs sites permet de répartir la charge sur les processeurs et sur les entrées/sorties.
- **Faciliter l'accroissement :** l'accroissement se fait par l'ajout de machines sur le réseau.

4. Inconvénients de la répartition des données

L'inconvénient majeur de la répartition des données d'une BD entre plusieurs sites est la complexité résultant de leur coordination.

Cette complexité se répartit de la façon suivante : [3]

- Le coût de mise au point du logiciel.
- Le nombre d'erreurs logicielles plus important.
- Echange de messages.
- Calcul supplémentaire.
- Récupération de système plus complexe après panne (Réintégration des sites ou liaison en pannes).

5. Les objectifs des bases de données réparties

Les objectifs de répartition d'une base de données peuvent être résumés ainsi : [4]

- Autonomie locale :
    - La base de données locale est complète et autonome (intégrité, sécurité, gestion), elle peut évoluer indépendamment des autres.
- Egalité entre sites :
    - Un site en panne ne doit pas empêcher le fonctionnement des autres sites.
- Fonctionnement continu :
    - La distribution permet la résistance aux fautes et aux pannes.
- Localisation transparente :
    - Accès uniforme aux données quel que soit leur site de stockage (fragmentation transparente).
    - Les données répliquées doivent être maintenues en cohérence.
- Requêtes distribuées :
    - L'exécution d'une requête peut être répartie (automatiquement) entre plusieurs sites (si les données sont réparties).
- Transactions réparties :
    - Le mécanisme de transactions peut être réparti entre plusieurs sites.
- Indépendance vis-à-vis du matériel :
    - Le SGBD fonctionne sur les différentes plateformes utilisées.
- Indépendance vis-à-vis du SE :
    - Le SGBD fonctionne sur les différents systèmes d'exploitation.

6. SGBD Réparti

6.1. Définition d'un Système de gestion de bases de données (Data Base Management System)

Le SGBD (Système de Gestion des Bases de Données) est l'outil principal de gestion d'une base de données. Il permet d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'informations. C'est une interface entre les utilisateurs et la mémoire de masse. Il facilite ainsi le travail des utilisateurs en leur donnant l'impression que l'information est organisée comme ils le souhaitent. Le SGBD est composé de plusieurs couches :

- Le SGBD externe (user interface handler). Sa tâche est d'interpréter les commandes utilisateurs.
- Le contrôleur sémantique des données (sémantic data controler). Il utilise les différentes contraintes définies sur la base de données afin de vérifier qu'une requête d'un utilisateur peut être effectuée.
- Le processeur de requêtes (query processor). Il détermine une stratégie afin de minimiser le temps d'exécution d'une requête.
- Le gestionnaire de transactions (transaction manager). Il assure la coordination des différentes demandes des utilisateurs.
- Le gestionnaire de reprise (recovery manager). Il s'occupe d'assurer la cohérence des données lorsque des pannes surviennent.
- Le système de gestion des fichiers (run-time support processor). Il gère le stockage physique de l'information. Il est dépendant du matériel utilisé.

Un SGBD réparti doit rendre la répartition des bases de données transparentes aux utilisateurs. La base de données étant répartie, il faut également répartir certaines fonctionnalités du SGBD. Le schéma d'un SGBD réparti est résumé dans la figure suivante [1].

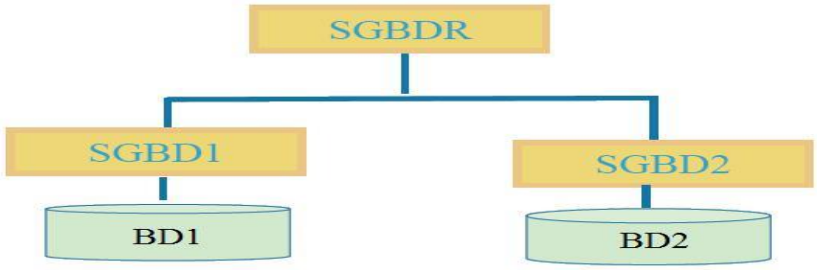

Figure I-3: Schéma d'un SGBD

6.2. Rôle d'un SGBD

Le logiciel de gestion d'un système de base de données (SGBD) à pour rôle :

- d'assurer la confidentialité des données : implémentation d'un mécanisme d'authentification par compte avec un mot de passe, attribution de rôles aux utilisateurs permettant d'ouvrir ou de réduire la surface d'exposition des données ;
- d'assurer la cohérence des données : vérifier les connaissances d'unicité (clés primaires) et les contraintes d'intégrité fonctionnelles (s'assurer qu'une clé étrangère référence bien un clé primaire et que la suppression d'une clé primaire ne crée pas d'enregistrement orphelins) ;
- d'assurer la gestion des incidents : le système doit s'assurer que l'échec d'une requête ne remet pas en cause l'intégrité des données. Il doit également pouvoir procéder aux reprises sur incident suite à une panne du serveur hébergeant la base de données, par exemple.[1]

7. Architecture des bases de données réparties

Pour implémenter une base de donnée répartie en est besoin de deux types d'architecture suivant :

7.1. Architecture Client-Serveur

Une application est bâtie selon une architecture client-serveur lorsqu'elle est composée de deux programmes, coopérant l'un avec l'autre à la réalisation d'un même traitement. La première partie, appelée module client, est installée sur le poste de travail alors que la seconde, appelée module serveur, est implantée sur l'ordinateur (ou même des ordinateurs

éventuellement situés dans des lieux géographiques différents) chargé de rendre le service (micro, mini ou grand système).

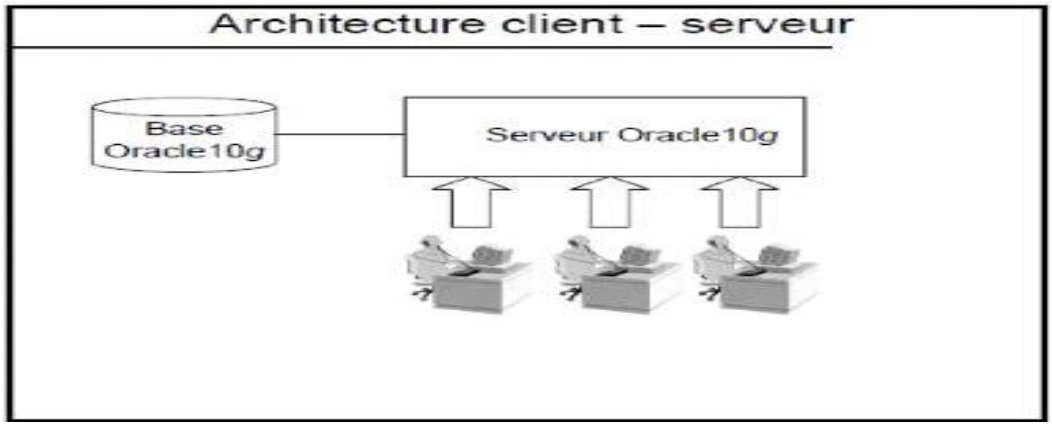

Figure I-4 : Architecture Client-Serveur

7.2. Architecture Pair-à-Pair (Peer-to-Peer, P2P)

C'est un type de communication pour lequel toutes les machines ont une importance équivalente. Il n'y a pas de machine qui a une importance hiérarchique par rapport aux autres. Dite aussi, l'architecture totalement répartie.

Chacune de ces architectures possède des avantages et des inconvénients. Le Client /Serveur avec sa structure plus hiérarchique est très sensible aux problèmes de panne des serveurs, bloquant ainsi les clients. En revanche, la prise de décision des serveurs est rapide.

Pour l'architecture Peer-To-Peer, comme les machines sont strictement équivalentes, la panne d'une machine peut rarement rendre le système un peu lent. Mais cette architecture engendre énormément de communication pour toute décision.

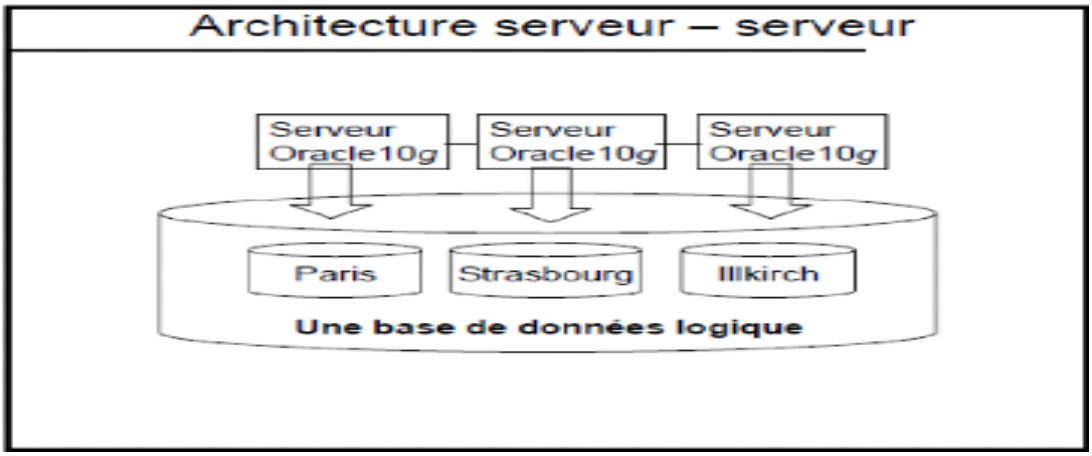

Figure I-5: Architecture serveur-serveur.

8. Conclusion

Ainsi, se termine cette première partie qui été une présentation générale des notions de base de données répartie qui représentent un domaine important pour la gestion de grand volume de données pour les entreprises réparties géographiquement. Dans la suite, nous allons exposer les techniques de conception et de gestion des bases de données réparties.

Chapitre II

Conception d’une base de données répartie

1. Introduction

Comme dans tous les mécanismes, la phase de conception est la plus importante et déterminante dans la mise en place d'une base de données reparties. Le rôle du concepteur est nécessaire de bien comprendre les étapes de conception, de définir les différents fragments de la base et leurs localisations ; d'évaluer les différents coûts de stockage et de transfert, et les priorités à respecter car différentes méthodes de conception existent et chacune d'elles nous offre une approche très différente de l'autre.

2. Conceptions d'une base de données répartie

La conception d'une base de données répartie peut être le résultat de deux approches totalement distinctes, soit le regroupement d'une multitude de bases de données déjà existantes (approche ascendante) soit construite du zéro (approche descendante).

Pour la conception des bases de données réparties il faut prendre des décisions en fonction de critères techniques et organisationnels pour l'objectif de minimiser le nombre de transferts entre sites, les temps de transfert, le volume de données transférées, les temps moyens de traitement des requêtes... etc.

2.1 Méthode de conception

2.1.1 Conception descendante (top down design)

On commence par définir un schéma conceptuel global de la base de données répartie, puis on distribue sur les différents sites en des schémas conceptuels locaux.

La répartition se fait donc en deux étapes, en première étape la fragmentation, et en deuxième étape l'allocation de ces fragments aux sites.

L'approche top down est intéressante quand on part du néant. [3]

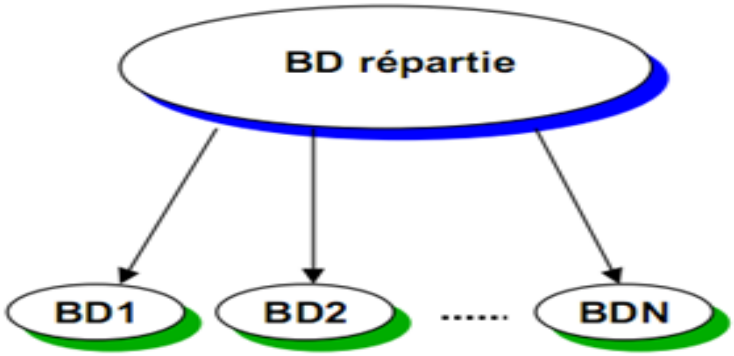

Figure II- 1 : Conception descendante d'une BDR

2.1.2 Conception ascendante (bottom up design)

L'approche se base sur le fait que la répartition est déjà faite, mais il faut réussir à intégrer les différentes BDs existantes en une seule BD globale. En d'autres termes, les schémas conceptuels locaux existent et il faut réussir à les unifier dans un schéma conceptuel global. Si les BDs existent déjà la méthode **bottomup** est utilisée.[3]

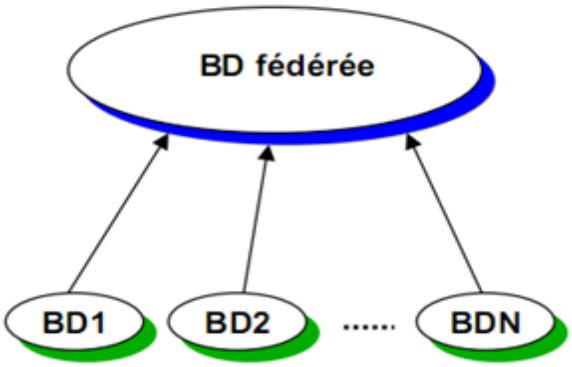

Figure II-2 : Conception ascendante d'une BDR.

3. Techniques utilisés pour la distribution des bases de données

La distribution de la base de données se fait en deux étapes, en première étape la fragmentation, et en deuxième étape l'allocation de ces fragments aux sites avec ou sans réplication. La distribution de la base de données est représentée dans la figure suivante :

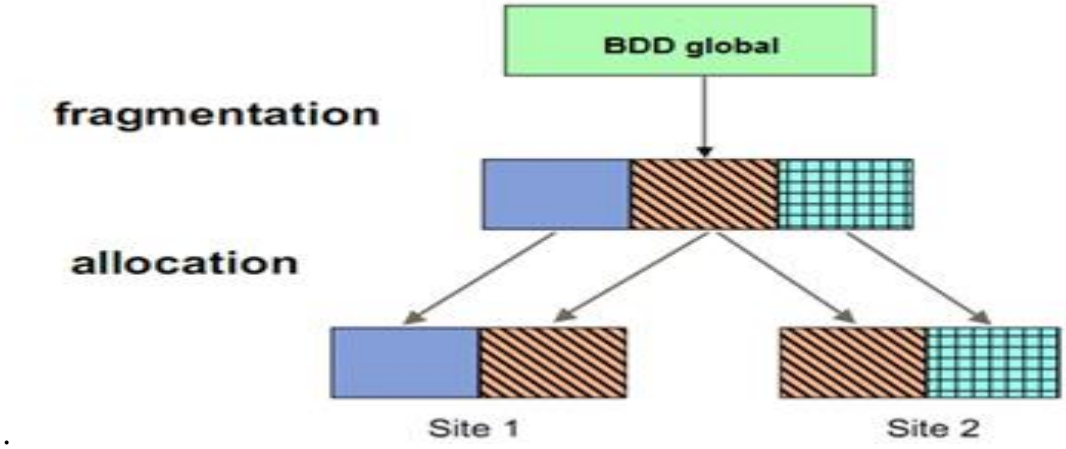

Figure II-3 : Décomposition d'une BDD global

3.1 la fragmentation

3.1.1 Définition

La fragmentation est le processus de décomposition d'une base de données en un ensemble de sous bases de données. Cette décomposition doit être sans perte d'information.

L'utilisation de petits fragments permet de faire tourner plus de processus simultanément, ce qui entraîne une meilleure utilisation des capacités du réseau d'ordinateurs.

Elle se base sur les règles suivantes :

- **La complétude :** Pour toute donnée d'une relation R, il existe un fragment Ri de la relation R qui possède cette donnée.
- **La reconstruction :** Pour toute relation décomposée en un ensemble de fragments Ri, il existe une opération de reconstruction.
- **Disjonction** : une donnée n'est présente que dans un seul fragment, sauf dans le cas de la fragmentation verticale pour la clé primaire qui doit être présente dans l'ensemble des fragments issus d'une relation. [3]

3.1.2 Objectif de la fragmentation

Les applications ne travaillent que sur des sous-ensembles des relations. Une distribution complète des relations générerait soit beaucoup de trafic, soit une réplication des données avec tous

les problèmes que cela occasionne : problèmes de mises à jour, problèmes de stockage. Il est donc préférable de mieux distribuer ces sous-ensembles.

L'utilisation de petits fragments permet de faire tourner plus de processus simultanément, ce qui entraîne une meilleure utilisation des capacités du réseau d'ordinateurs. [2]

3.1.3 Les problèmes de la fragmentation

La fragmentation peut être coûteuse s'il existe des applications qui possèdent des besoins opposés. On est en quelque sorte dans le cas d'une exclusion mutuelle qui empêche une fragmentation correcte.

Par ailleurs, la vérification des dépendances sur différents sites peut être une opération très longue. [2]

3.1.4 Types de fragmentation

Il existe 3 types de fragmentations :

a. la fragmentation horizontale

La relation est divisée en plusieurs sous-relations contenant chacune un sous-ensemble des tuples (lignes) de la relation.

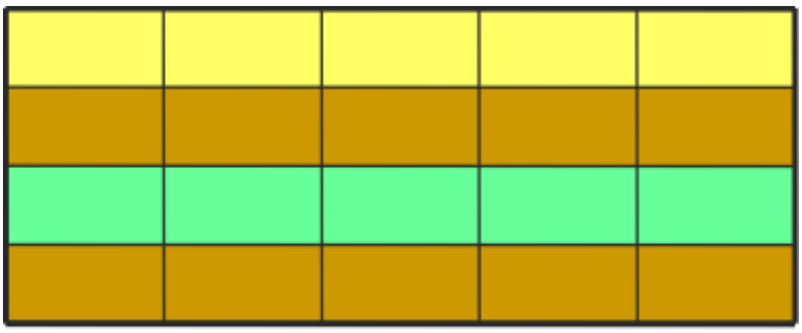

Figure II-4: Fragmentation horizontale

b. La fragmentation verticale

La relation est divisée en plusieurs sous-relations contenant chacune un sous-ensemble des attributs (colonnes) de la relation .La fragmentation horizontale a tout son intérêt pour une société dispersée aux quatre coins du globe et qui maintient une relation contenant ses employées. Cette relation logique, peut-être fragmentée horizontalement en plusieurs groupes contenant chaque fois les employés selon leur localisation.

La fragmentation verticale est plus complexe et moins intuitive. Les sous-relations ne contiennent pas tous les attributs mais tous les tuples. Un peu comme une vue permet de cacher les attributs inutiles selon le contexte, la fragmentation verticale peut-être utile d'un point de vue hiérarchique.[1]

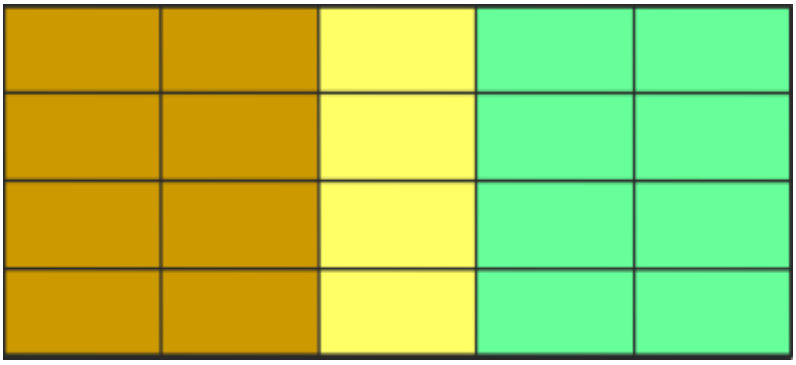

Figure II-5: Fragmentation verticale

c. La fragmentation mixte

Elle résulte de l'application successive d'opérations de fragmentation horizontale et verticale sur une relation globale. [2]

Figure II-6: Fragmentation mixte

3.2 Allocation des fragments aux sites

L'affectation des fragments sur les sites est décidée en fonction des requêtes qui ont servi à la fragmentation. Le but est de placer les fragments sur les sites où ils sont le plus utilisés, et ce pour minimiser les transferts de données entre les sites.

L'allocation peut se faire avec réplication ou sans réplication. Sachant que la réplication favorise les performances des requêtes et la disponibilité des données, mais est coûteuse en considérant les mises à jour des fragments répliquas.[3]

4. La réplication

4.1 Définition

La réplication consiste à copier les informations d'une base de données sur une autre. Elle peut être accompagnée d'une transformation des données sources, voir souvent d'une agrégation. Dans tous les cas, il s'agit d'une redondance d'information.

L'objectif principal de la réplication est de faciliter l'accès aux données en augmentant la disponibilité. Soit parce que les données sont copiées sur différents sites permettant de répartir les requêtes, soit parce qu'un site peut prendre la relève lorsque le serveur principal s'écroule. Une autre application tout aussi importante est l'amélioration des performances des requêtes sur les données locales, et ceci permet d'éviter les transferts de données et d'accroître la résistance aux pannes.[2]

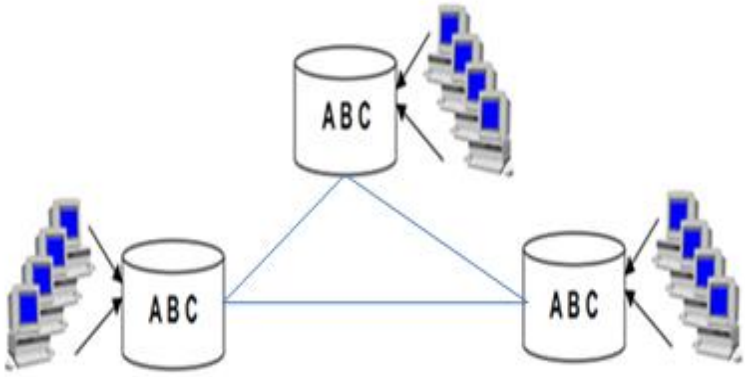

Figure II-7 : Réplication de données

4.2 Principe

Le principe de la réplication, qui met en jeu au minimum deux SGBDs, est assez simple et se déroule en trois étapes : [2]

1. La base maître reçoit un ordre de mise à jour (INSERT, UPDATE ou DELETE).

2. Les modifications faites sur les données sont détectées et stockées dans un fichier ou une file d'attente en vue de leur propagation.

3. Le processus de réplication prend en charge la propagation des modifications à faire sur une seconde base dite esclave. Il peut bien entendu y avoir plus d'une base esclave.

4.3 Les Types de réplication

4.3.1 La réplication synchrone

Aussi appelée « Réplication en temps réel » La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant la validation (commit) de la requête sur le serveur où la requête est exécutée. Ce type de réplication assure un haut degré d'intégrité des données mais requiert une disponibilité permanente des serveurs et de la bande passante. Ce type de réplication, fortement dépendant des pannes des systèmes, nécessite de gérer des transactions multi sites coûteuses en ressources. La réplication synchrone est utilisée dans le cas des compagnies aériennes, des banques . . . etc.

Deux types de réplication synchrone :La réplication synchrone asymétrique et la réplication Synchrone symétrique.

a. Réplication synchrone asymétrique

Elle utilise un site primaire (maitre) qui pousse les mises à jour en temps réel vers un ou plusieurs sites secondaires. La table répliquée est immédiatement mise à jour pour chaque modification par utilisation de trigger sur la table maître.

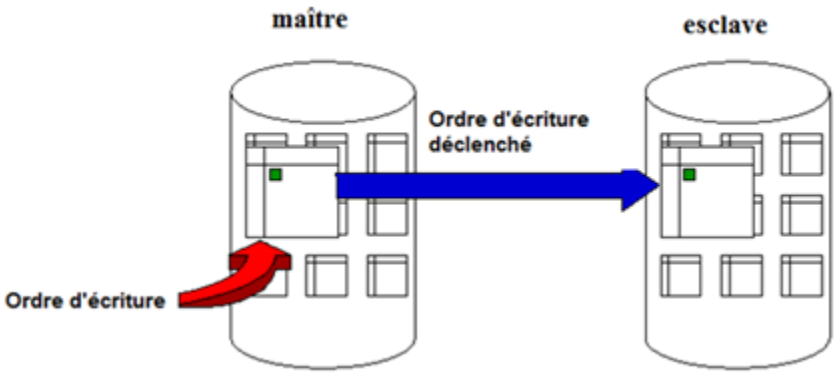

Figure II-8 : Réplication synchrone asymétrique

b. La réplication synchrone symétrique

Lors de la réplication synchrone symétrique, il n'y a pas de table maître. L'utilisation de trigger sur chaque table doit différencier une mise à jour client à répercuter d'une mise à jour par réplication. Cette technique nécessite l'utilisation de jeton.

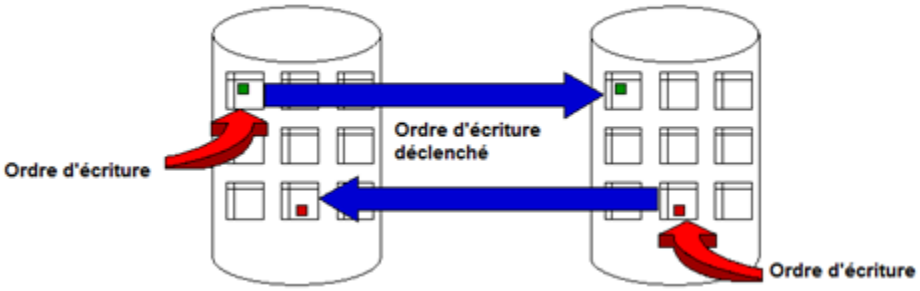

Figure II-9 : Réplication synchrone symétrique

4.3.2 La réplication asynchrone

La réplication asynchrone stocke les opérations intervenues sur une base de données dans une fille d'attente pour les propager plus tard à l'aide d'un processus de synchronisation. Ce type de réplication est plus flexible que la réplication synchrone. Il permet en effet de définir les intervalles de synchronisation, ce qui permet à un site de fonctionner même si un site de réplication n'est pas accessible. Si le site distant est victime d'une panne, l'absence de synchronisation n'empêche pas la consistance de la base maîtresse.

La réplication asynchrone est appropriée pour une équipe de développeurs travaillant à distance sur une application.

Il existe deux types de réplication asynchrone **:**La réplication asynchrone asymétrique et la réplication asynchrone symétrique.

a. La réplication asynchrone asymétrique

Elle pousse les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple. [2]

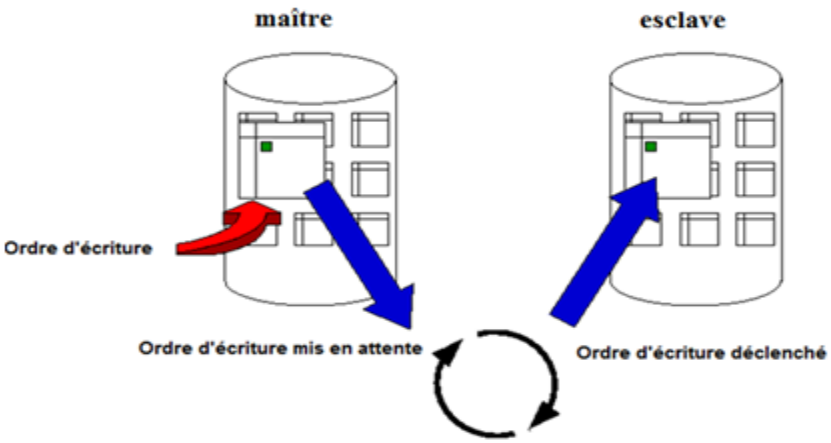

Figure II-10 : Réplication asynchrone asymétrique

b. La réplication asynchrone symétrique

Dans ce cas, la mise à jour des tables répliquées est différée. Cette technique risque de provoquer des incohérences de données.

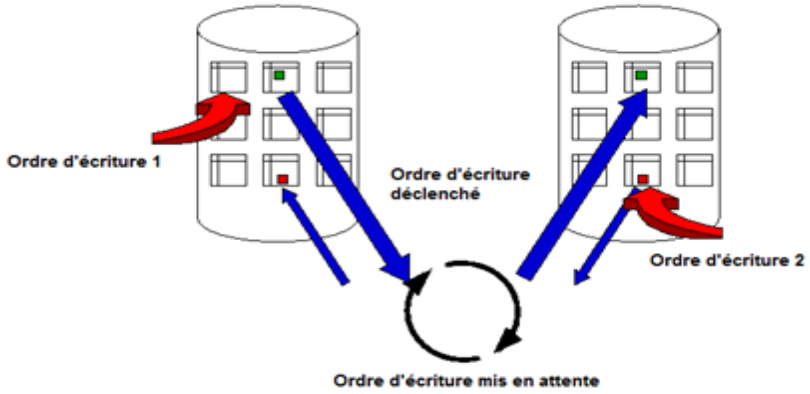

Figure II-11 : Réplication asynchrone symétrique

Il est bien évident que la réplication synchrone nécessite des moyens beaucoup plus coûteux que l'asynchrone, en raison principalement de la bande passante nécessaire pour assurer un échange parfait entre les différents serveurs d'un réseau. Les solutions asynchrones sont donc privilégiées par les entreprises dont l'exigence en "temps réel" et en continuité de service est réduite.

4.4 Les avantages de la réplication

Les avantages de la réplication sont assez nombreux, selon le type on trouve :

-Allégement du trafic réseau en répartissant la charge sur divers sites. Par conséquent, rapidité des accès aux données.

- Amélioration des performances des requêtes.

- Résistance aux pannes par l'augmentation de la disponibilité des données.

5. Gestion des Bases de données réparties

5.1 Mise à jour des BDDR

La principale difficulté réside dans le fait qu'une mise à jour dans une relation du schéma global se traduit par plusieurs mises à jour dans différents fragments.

Il faut donc identifier les fragments concernés par l'opération de mise à jour, puis décomposer en conséquence l'opération en un ensemble d'opération de mise à jour sur ces fragments.

- Insertion

Retrouver le fragment horizontal concerné en utilisant les conditions qui définissent les fragments horizontaux, puis insertion du tuple dans tous les fragments verticaux correspondants.

- Suppression

Rechercher le tuple dans les fragments qui sont susceptibles de contenir le tuple concerné, et supprimer les valeurs d'attribut du tuple dans tous les fragments verticaux.

- Modification

Rechercher les tuples, les modifier et les déplacer vers les bons fragments si nécessaire. [1]

6. Conclusion

Dans ce chapitre, nous avons présenté les principes de la répartition des données. Cette répartition peut se faire selon différents scénarios choisis par le concepteur, tout en prenant en compte les restrictions et les obligations de conception.

Nous avons vu également, comment gérer une base de données répartie avec les principes de réplication symétrique et asymétrique.

Partie II

Etude du cas

Chapitre I

Etude préliminaire

1. Introduction

Dans ce chapitre nous allons déterminer la phase d'étude préliminaire qui permet de recueillir les informations initiales sur le système. Il s'agit dans cette phase de définir le contour du système, les différents acteurs, ainsi que les messages d'interaction avec le système.

L'étude préliminaire est la toute première étape du processus 2TUP. Elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels, en utilisant principalement le texte, ou diagrammes très simples. Elle prépare les activités plus formelles de capture des besoins fonctionnels et de capture technique.

2. Élaboration du cahier de charge

2.1. Présentation de l'organisme d'accueil

La commune est la collectivité territoriale de base de l'État algérien, à la fois collectivité disposant de la personnalité morale, dotée de ses propres organes, délibératif et exécutif, et plus petite subdivision administrative de l'organisation territoriale de l'Algérie. Cette double compétence de la commune est exercée par le président de l'Assemblée Populaire Communale (A.P.C), qui est conjointement le représentant de la commune et le représentant de l'État au niveau communal. Mila la nouvelle ville a été fondée en 1877 et transformée en une commune par la publication 23/11/1890 La commune de Mila est l'un des Trente-deux commune de la wilaya algérienne de Mila.

La gestion administrative de la commune de Mila se répartie plusieurs services :

- Service d'administration et des moyens généraux.
- Service des affaires sociales, culturelles et sportives
- Service d'urbanisme et construction et équipement.
- Service de la route et réseaux divers biens.
- Service de réglementation générale.

Notre projet est à réaliser au niveau de service de réglementation générale, en particulierdans le bureau de carte grise. La figure suivante illustre l'organigramme de différents services de la commune du Mila:[5]

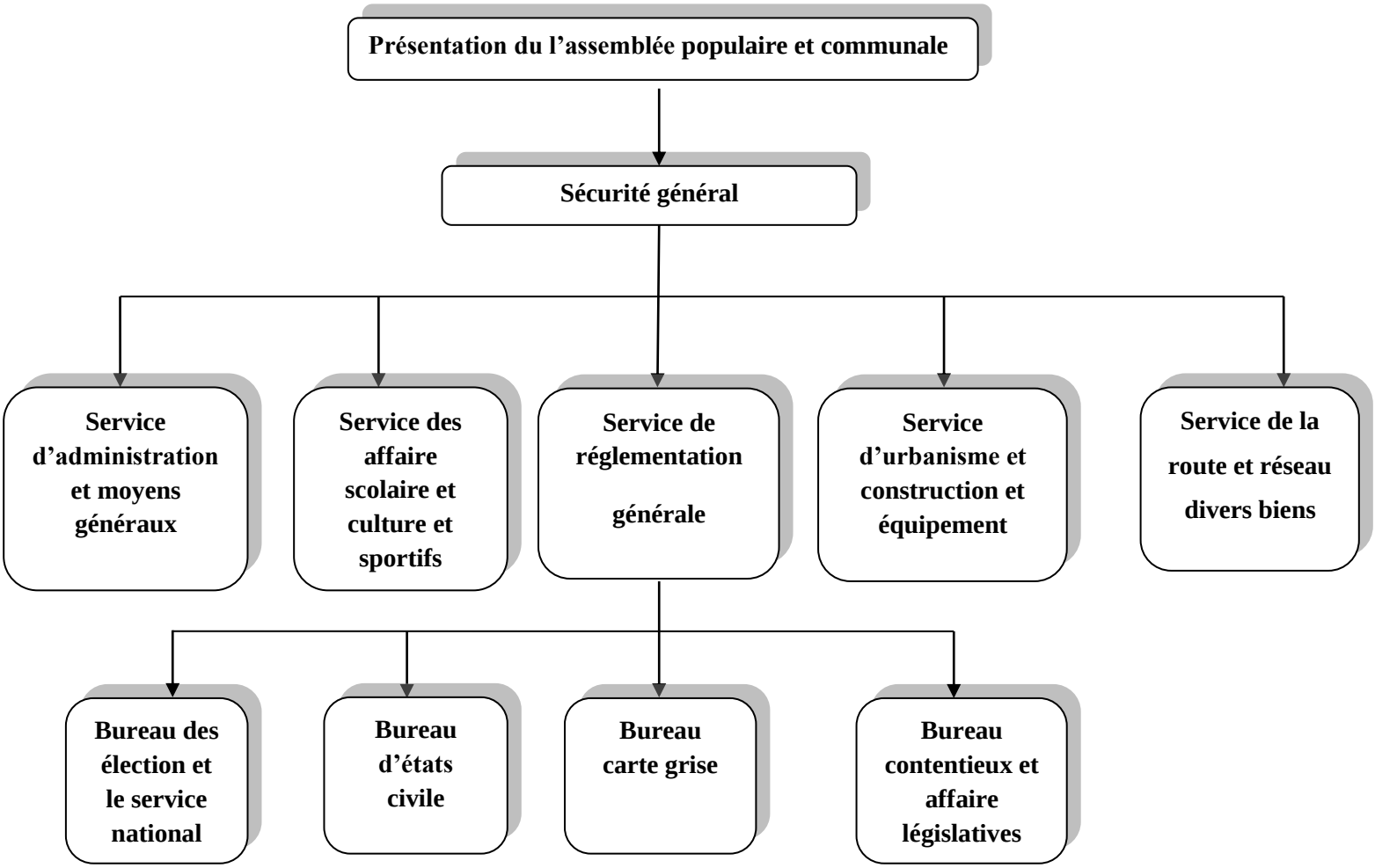

Figure I-1: L'organigramme de différents services de la commune du Mila

2.2. Présentation du projet

Notre projet consiste à un système d'information qui se base sur une base de données répartie pour la gestion des cartes grises au niveau de la commune de Mila.

Principalement les échanges d'informations est entre les différents systèmes de chaque wilaya lorsqu'il s'agit d'une vente externe et aussi avec le système central, lorsqu'il s'agit d'une mise à jour d'une carte grise ou bien une vente locale et aussi une nouvelle voiture.

Pour ce faire, il s'avère nécessaire de présenter l'organisme d'accueil qui est la commune de Mila afin de comprendre les activités principales qu'il exerce.

2.3. Problématique

Notre stage pratique a porté sur le suivie de la gestion de la carte grise au sein du bureau du carte grise de la commune du Mila.

Malgré la présence d'un système de gestion de la carte grise, certains problèmes restent toujours posés et peuvent être résumés comme suit :

- ✓ Manque d'un réseau national qui relie les différents systèmes de chaque wilaya et aussi avec le système central, donc pour effectuer une vente externe et extraire la nouvelle carte grise cela peut prendre une longue durée.

Notre travail intervenant dans le domaine de la gestion de la carte grise qui consiste à la mise en place d'une application base de données pour la distribution des tâches relative à la gestion de la carte grise de manière à rendre flexibles et souples l'accès et la manipulation des informations.

2.4. Objectifs de nouveau système

La gestion de la carte grise est une tâche très difficile plus qu'elle est trop compliquée et demande beaucoup du temps c'est dans ce sens qu'on présente notre sujet qui consiste à réaliser une application distribué fiable et robuste qui élimine la majorité des problèmes poser dans l'ancien système. Pour répondre aux besoins des citoyens ainsi que les employés du bureau pour la bonne gestion du ces tâches.

En peut résumer les objectifs de notre application grâce à ces points :

- ✓ Faciliter la communication entre les serveurs de chaque wilaya.
- ✓ Lorsqu'un serveur local et en panne les opérations effectuer sur ce serveur vont être effectué automatiquement sur le serveur central.
- ✓ Automatiser la vente externe d'une voiture par l'interconnexion entre les différentes applications de chaque wilaya.

2.5. Grands choix techniques

Pour réaliser ce projet nous allons utiliser une approche itérative et incrémentale, fondée sur le processus en Y.

Nous avons choisi aussi un certain nombre de techniques –clés.Ces techniquesclés sont principalement :

- ✓ Le langage de modélisation UML.
- ✓ Le processus de développement en Y (2TUP).
- ✓ L'environnement de développement NetBeans .
- ✓ Le langage de programmation java.
- ✓ Le SGBDR ORACLE .

2.6. Recueil des besoins fonctionnel

Un premier tour d'horizon des besoins exprimés par les employés de la carte grise permetd'établir un cahier des charges préliminaire ; durant notre stage pratique on a constaté qu'il y a différents cas pour extraire la carte d'immatriculions des véhicules :[5]

- ➢ Cas de ré-immatriculation du véhicule dans la même wilaya : le propriétaire doit disposer le dossier suivant :
  - Formulaire de demande d'immatriculation du véhicule signé et légalisé.
  - Carte grise barrée du véhicule.
  - Acte de vente signé et légalisé.
  - Timbre Fiscal.
  - Quittance de paiement de la taxe de transaction pour les véhicules concernés.
  - Extrait de naissance délivré sur la base du livret de famille.
  - Fiche de résidence.
  - Photocopie légalisée de la Carte Nationale d'Identité en cours de validité.

- Photocopie légalisée du statut (si l'acheteur est une personne morale).

➢ Cas de ré-immatriculation du véhicule transféré dans une autre wilaya : le propriétaire doit disposer le dossier suivant :

- Formulaire de demande d'immatriculation du véhicule signé et légalisé.
- Carte grise barrée du véhicule.
- Acte de vente signé et légalisé.
- Fiche de contrôle visée par l'ingénieur des Mines de la Wilaya d'accueil.
- Timbre Fiscal.
- Quittance de paiement de la taxe de transaction pour les véhicules concernés.
- Extrait de naissance délivré sur la base du livret de famille.
- Fiche de résidence.
- Photocopie légalisée de la Carte Nationale d'Identité en cours de validité.
- Photocopie légalisée du statut (si l'acheteur est une personne morale).

➢ Cas d'immatriculation de véhicule neuf :

- Formulaire de demande d'immatriculation du véhicule signé et légalisé.
- Carte d'immatriculation provisoire (carte jaune).
- Certificat de conformité du véhicule.
- Attestation de vente délivrée par le concessionnaire.
- Facture d'achat du véhicule.
- Timbre Fiscal.
- Quittance de paiement de la taxe de transaction pour les véhicules concernés.
- Extrait de naissance délivré sur la base du livret de famille.
- Fiche de résidence.

- Photocopie légalisée de la Carte Nationale d'Identité en cours de validité.
- Photocopie légalisée du statut (si l'acheteur est une personne morale).

2.7. Recueil des besoins opérationnels

Sécurité : Chaque utilisateur de l'application doit s'authentifier par un nom d'utilisateur et un mot de passe, pour qu'il puisse utiliser le système et accéder à l'application.

Temps de réponse : Doit être acceptable.

Interface graphique : Doit être simple à utiliser et convivial.

3. Description de contexte du système

3.1. Identification des acteurs

Les acteurs sont des entités externes qui interagissent avec le système, comme une personne humaine, un autre système ou un robot. Dans notre application on distingue deux catégories d'acteurs :

Administrateur : est le responsable de l'application, il a le droit d'ajouter un compte, activé/désactivé un compte et modifier un compte.

Agent employée : le responsable à gérer les opérations concernant la carte grise.

3.2. Identification des messages

On va détailler les différents messages échangés entre le système et l'utilisateur.

- Les messages émis par le système sont :

1. Les informations d'authentification.
2. Les informations d'une carte grise.
3. Les informations du gage /opposition/ incessibilité.
4. Les informations du gage /opposition/ incessibilité.
5. Les informations du duplicata.

6. Les informations de radiation carte grise.
7. Les informations de la vente locale.
8. Les informations d'une fiche de confirmation.
9. Les informations d'un avis de mutation.
10. Les informations de recherche une carte grise.
11. Les informations d'un employé.

- Les messages reçus par le système sont :

1. La confirmation d'authentification.
2. La confirmation d'ajout de la carte grise.
3. La confirmation de modification de la carte grise.
4. La confirmation de radier carte grise.
5. La confirmation d'ajout vente locale.
6. La confirmation d'annuler vente locale.
7. La confirmation d'ajout fiche de contrôle.
8. La confirmation de modifier fiche de contrôle.
9. La confirmation d'annuler fiche de contrôle.
10. La confirmation d'ajout de fiche de confirmation.
11. La confirmation d'ajout l'avis de mutation.

3.3. La méthode de conception

Notre système consiste à réaliser une application distribué pour la gestion de la carte grise, chaque wilaya possède son propre serveur qui contient tous les mis à jour de la gestion de la carte grise ,le serveur local de chaque wilaya correspond au schéma local de notre base de donnée ,à partir de ses schéma locaux on va construire le schéma global de la base de

donnée donc notre conception va êtreest une conception ascendante ,on obtient le schéma global par la réplication des bases de données locales de chaque wilaya .

3.4. Modélisation du contexte

Après les étapes précédentes, et à partir des informations obtenues, nous allons modéliser le contexte de notre application. Ceci va nous permettre dans un premier temps, de définir le rôle de chaque acteur dans le système.

3.3.1. Diagramme de contexte dynamique

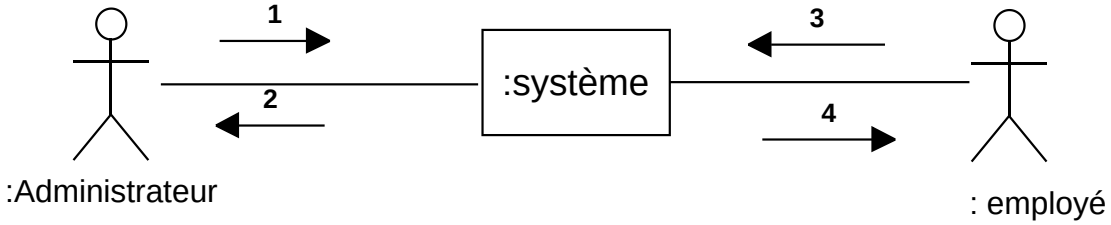

Figure I-2: Le diagramme de contexte dynamique du système

3.3.2. Description détaillée des messages

1 : l'administrateur ⟹ système
• Les informations concernant le compte d'authentification (login et mot de passe). • La mise à jour des informations des comptes (active/désactive, ajouter).

Tableau I-1: liste des messages d'administrateur

2 :système ⟹ l'administrateur
• Confirmation des informations d'authentification. • Confirmation des mises à jour concernant les comptes.

Tableau I-2: liste des messages du système vers l'administrateur

3 :employé ⟹ système
• Les informations d'authentification. • Les informations de la gestion de la carte grise. • Les informations concernant la vente locale. • Les informations concernant la vente externe.

Tableau I-3: liste des messages d'employé

4 : système ⟹ l'employé
• Le formulaire d'authentification. • Les formulaires de gestion de la carte grise. • Les formulaires concernant la vente locale et externe.

Tableau I-4: liste des messages du système vert l'employé

4. Conclusion

Après avoir dégagé les besoins fonctionnels et opérationnels et tous les critères qu'on doit prendre en considération, dans le prochain chapitre nous allons poursuivre la formalisation de ces besoins.

CHAPITRE II

CAPTURE DES BESOINS FONCTIONNELS ET TECHNIQUES

1. Introduction

Ce chapitre vient pour compléter l'étude fonctionnelle et technique ébauchée durant l'étude préliminaire dans le chapitre précédent. L'objectif de la capture des besoins consiste à déterminer ce que le système doit faire, c.à.d. le « quoi » a fourni aux développeurs une meilleure compréhension des fonctionnalités du système qu'ils doivent développer, elle comporte deux étapes : la capture des besoins fonctionnels et la capture des besoins techniques.

2. Capture des besoins fonctionnels

Elle représente un point de vue « fonctionnel » de l'architecture système. Et pour ce faire nous utiliserons la notion d'Use Case. Chaque Use Case sera identifié, décrit, et organisé, classé en fonction de son importance dans le projet.

La capture s'effectue sur plusieurs étapes

- Identification des cas d'utilisation.
- Le diagramme de cas d'utilisation pour les besoins fonctionnels.
- Description préliminaire les différents cas d'utilisation.
- Description détaillée des différents cas d'utilisation.
- Identifier les classes candidates du modèle d'analyse.

2.1. Déterminer les cas d'utilisations

- **Définition**

Un cas d'utilisation « use case » permet de décrire ce que le futur système devra faire, sans spécifier comment il le réalisera. Il permet d'exprimer le besoin des utilisateurs d'un système, il est donc une vision orientée utilisateur.

Liste des cas d'utilisations
• **Ajouter nouvelle carte grise.** • **Ajouter duplicata carte grise.** • **Annuler Gage/Opposition/Incessibilité.** • **Modifier carte grise.** • **Radier carte grise.** • **Ajouter vente locale.** • **Annuler vente locale.** • **Ajouter fiche de contrôle.** • **Modifier fiche de contrôle.** • **Annuler fiche de contrôle.** • **Etablir avis de mutation.** • **Rechercher carte grise.**

Tableau II-1 : La liste des cas d'utilisation du système de la carte grise

➢ **Identification des cas d'utilisation**

Pour constituer les cas d'utilisation, il faut considérer l'intention fonctionnelle de l'acteur par rapport au système dans le cadre de l'émission ou de la réception de chaque message. En regroupant les intentions fonctionnelles en unités cohérentes, on obtient les cas d'utilisation.

Cas d'utilisation	Acteur(s)	Message(s) émis/reçus
Ajouter nouvelle carte grise	**Employé**	**Emet :** Les informations concernant une nouvelle carte grise. **Reçoit :** La carte grise ajouter.
Ajouter duplicata carte grise	**Employé**	**Emet :** Les informations concernant le duplicata **Reçoit :** Le duplicata ajouter.
Annuler Gage/Opposition/Incessibilité	**Employé**	**Emet :** Les informations concernant la carte grise.

Modifier carte grise	**Employé**	**Emet :** Les informations concernant la carte grise. **Reçoit :** La carte grise modifier.
Radier carte grise	**Employé**	**Emet :** Les informations concernant la carte grise.
Ajouter vente locale	**Employé**	**Emet :** Les informations concernant le nouveau propriétaire. **Reçoit :** la vente locale ajouter.
Annuler vente locale	**Employé**	**Emet :** Les informations concernant le nouveau propriétaire.
Ajouter fiche de contrôle	**Employé**	**Emet :** Les informations concernant la fiche de contrôle. **Reçoit :** la fiche de contrôle ajouter.
Modifier fiche de contrôle	**Employé**	**Emet :** Les informations concernant la fiche de contrôle. **Reçoit :** La fiche de contrôle modifier.
Annuler fiche de contrôle	**Employé**	**Reçoit :** La fiche de contrôle annuler.
Etablir avis de mutation	**Employé**	**Emet :** Les informations concernant l'avis de mutation. **Reçoit :** L'avis de mutation ajouter.
Rechercher carte grise	**Employé**	**Emet :** Les informations de la carte grise. **Reçoit :** la liste des cartes grise.

Tableau II- 2 : La liste des acteurs et des messages par cas d'utilisation du système

Maintenant nous allons représenter notre diagramme de cas d'utilisation.

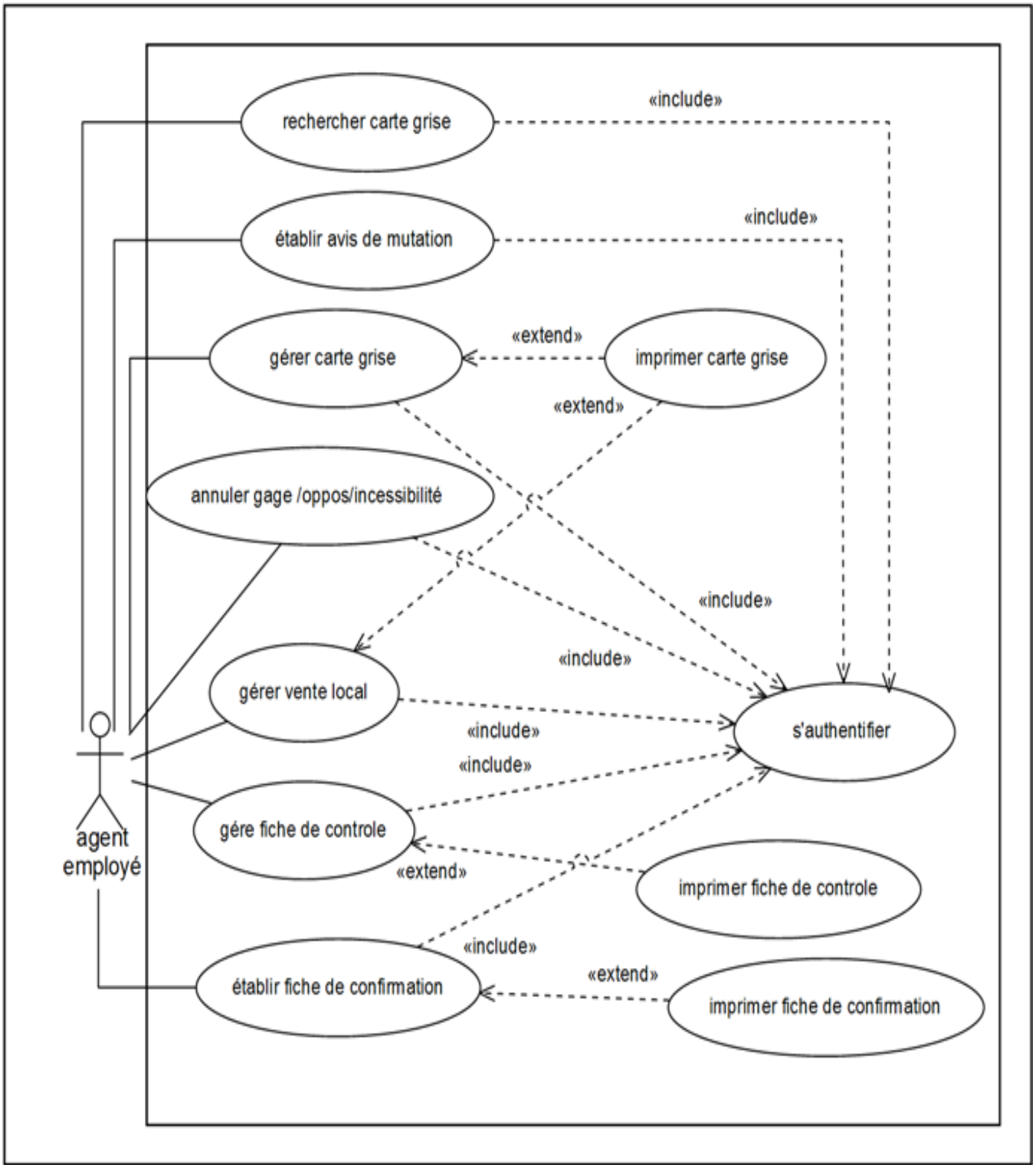

Figure II-1 : Diagramme de cas d'utilisation générale

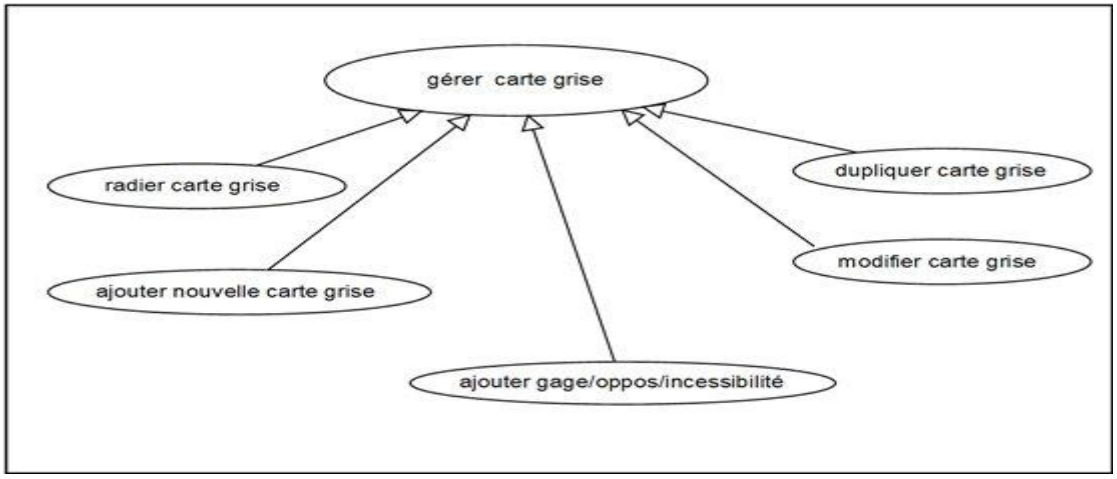

Figure II-2 : Diagramme de cas d'utilisation gérer carte grise

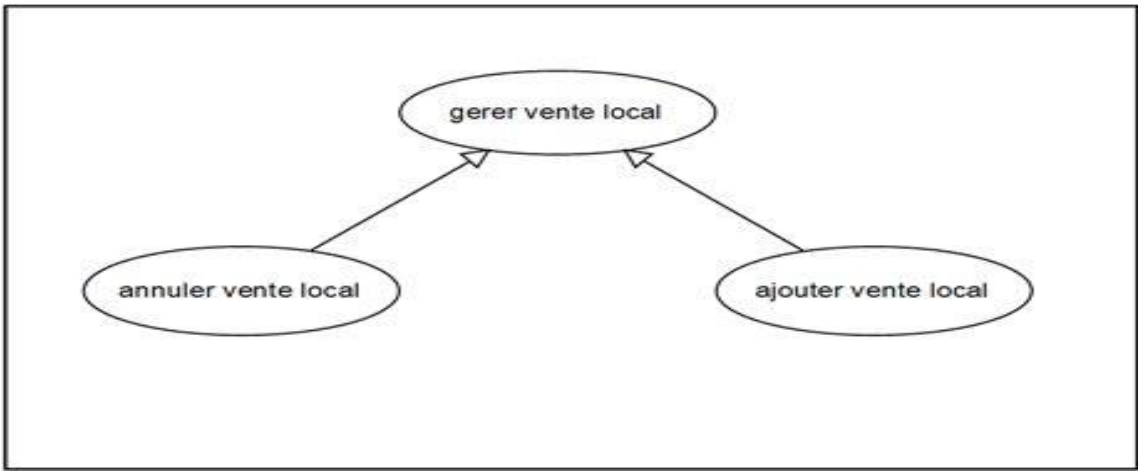

Figure II-3: Diagramme de cas d'utilisation gérer vente locale

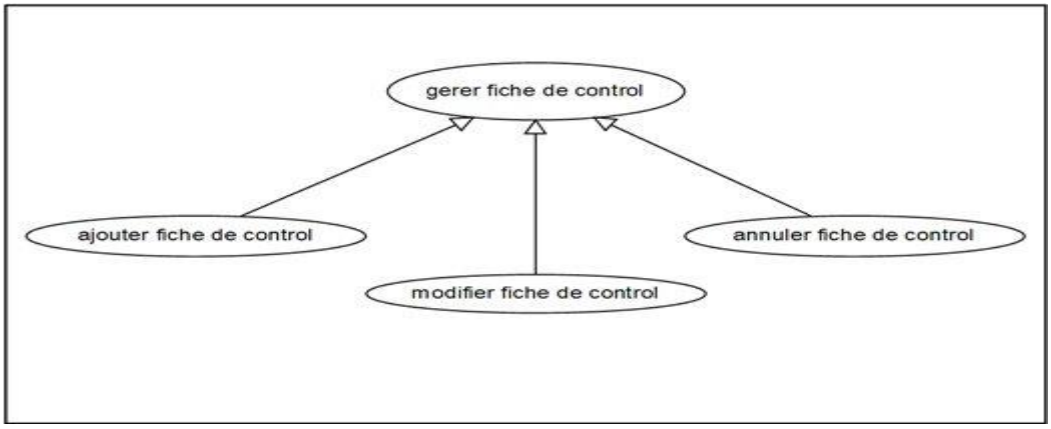

Figure II-4 : Diagramme de cas d'utilisation gérer fiche de contrôle

2.2. Description détaillée des cas d'utilisation

Nous allons maintenant détailler chaque cas d'utilisation et en donner une description plus formalisée.

Ajouter nouvelle carte grise

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Ajouter nouvelle carte grise.**
Acteur principale	L'employé.
Objectif	L'employé veut ajouter une nouvelle carte grise qui n'existe pas.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé a ajouté la nouvelle carte grise avec succès.
Scénario nominal	1. L'employé lance gérer carte grise. 2. Le système lui affiche un menu de gestion de carte grise. 3. L'employé choisi ajouter nouvelle carte grise. 4. Le système affiche un formulaire de la nouvelle carte grise contient des champs vides concernant la carte grise et le propriétaire. 5. L'employé remplis les champs vides et valides. 6. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	-Si la voiture est achetée avec l'un des cas suivant Gage/Opposition/Incessibilité : 1. L'employé lance une de ces cas Gage/Opposition/Incessibilité. 2. Le système affiche un formulaire avec des champs vides concernant l'une de ces cas Gage/Opposition/Incessibilité. 3. L'employé remplis les champs vides et valides. 4. Le système affiche un message « l'opération est terminée avec succès ». -Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	Aucun.

Tableau II-3 : Cas d'utilisation ajouter nouvelle carte grise.

- **Diagramme de séquence :**

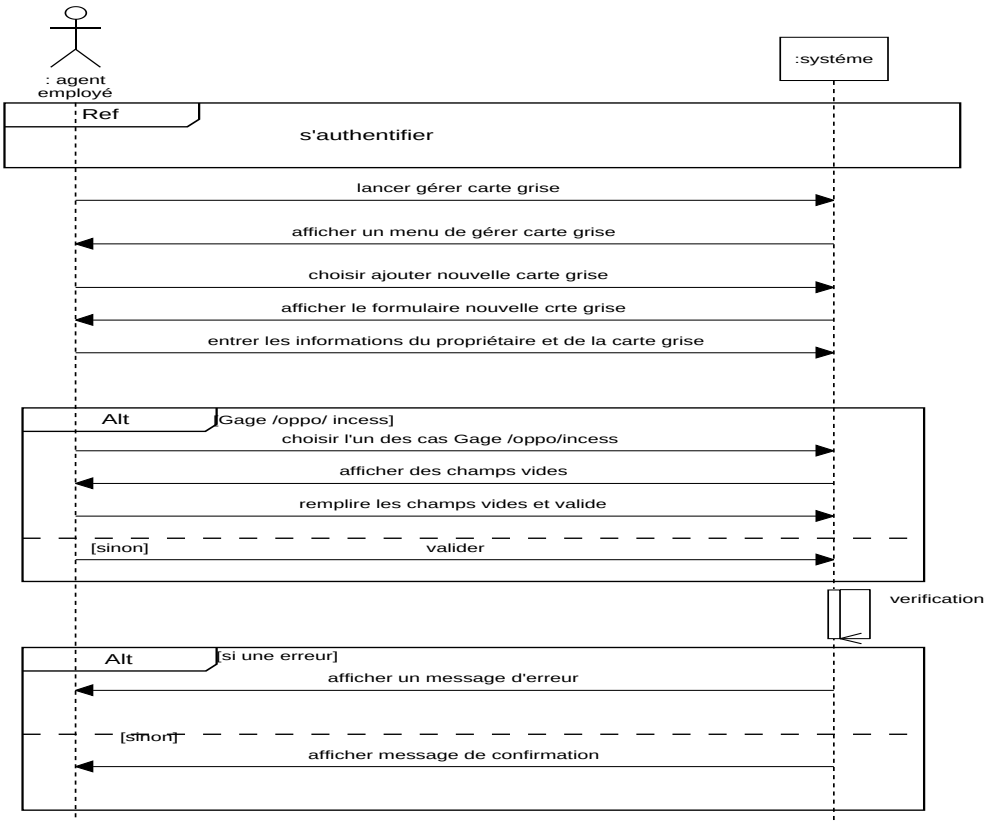

Figure II-5 : Diagramme de séquence système ajouter nouvelle carte grise

- **Diagramme d'activité :**

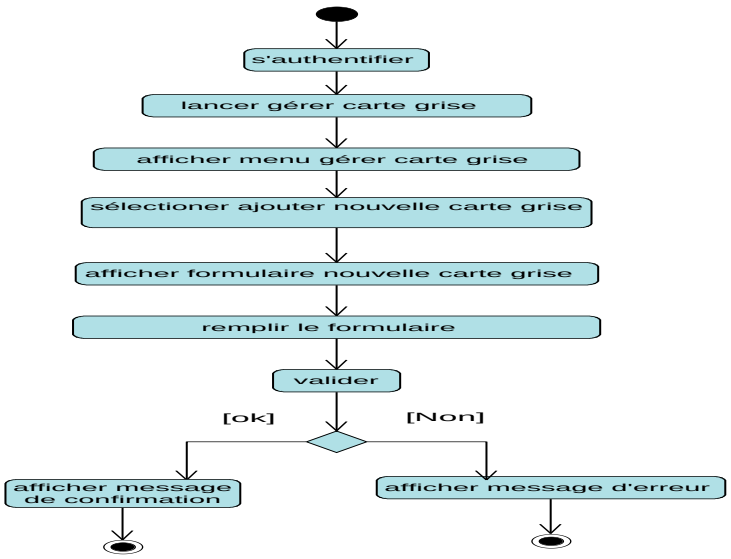

Figure II-6: Diagramme d'activité ajouter nouvelle carte grise

Ajouter duplicata de la carte grise

- **Description textuelle du cas d'utilisation :**

Cas d'utilisation	**Ajouter duplicata.**
Acteur principal	L'employé.
Objectif	Etablir une duplication d'une carte grise qui existe déjà.
Pré condition	L'employé doit être authentifié.
Poste condition	Un duplicata de la carte grise est établi avec succès.
Scénario nominal	1. L'employé lance gérer carte grise. 2. Le système lui affiche un menu de gestion de carte grise. 3. L'employé choisi duplicata carte grise. 4. Le système lui affiche un formulaire de duplicata de la carte grise. 5. L'employé entre la matricule de la voiture concernée. 6. Le système lui affiche les champs concernant le propriétaire et la carte grise remplis et grisées et les champs concernant le duplicata vides. 7. L'employé remplis les champs vides et valide. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-4 : Cas d'utilisation ajouter duplicata

- **Diagramme de séquence**

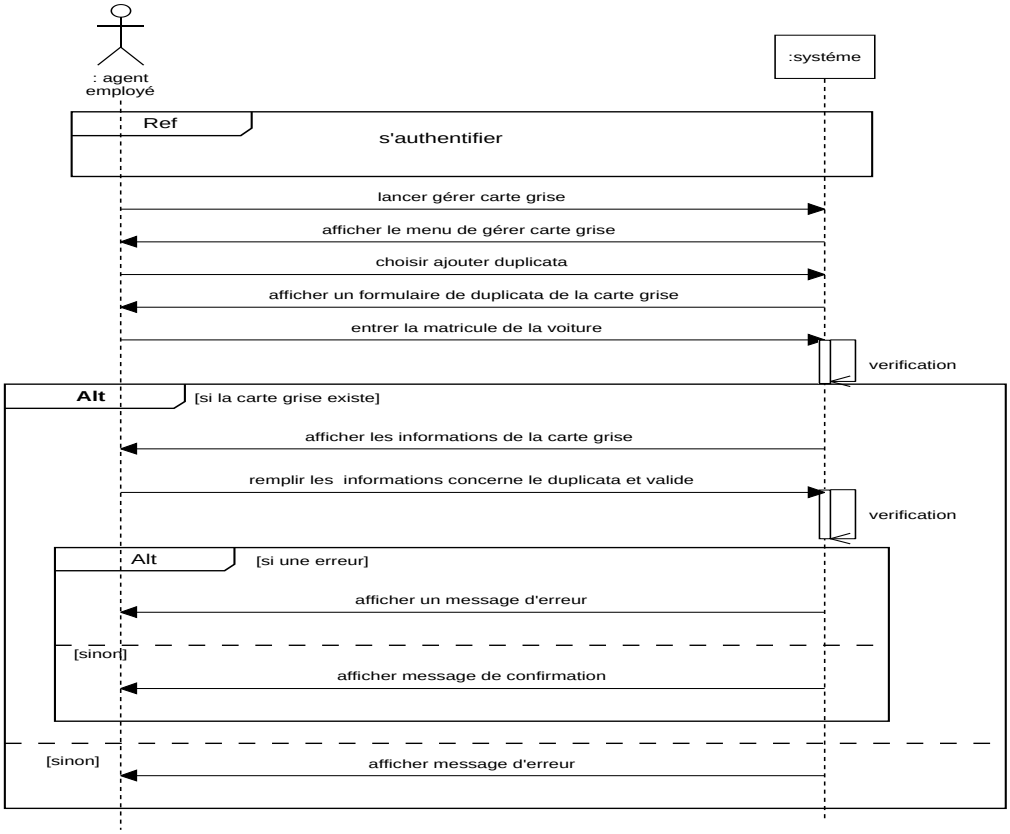

Figure II-7: Diagramme de séquence système ajouter duplicata

- **Diagramme d'activité**

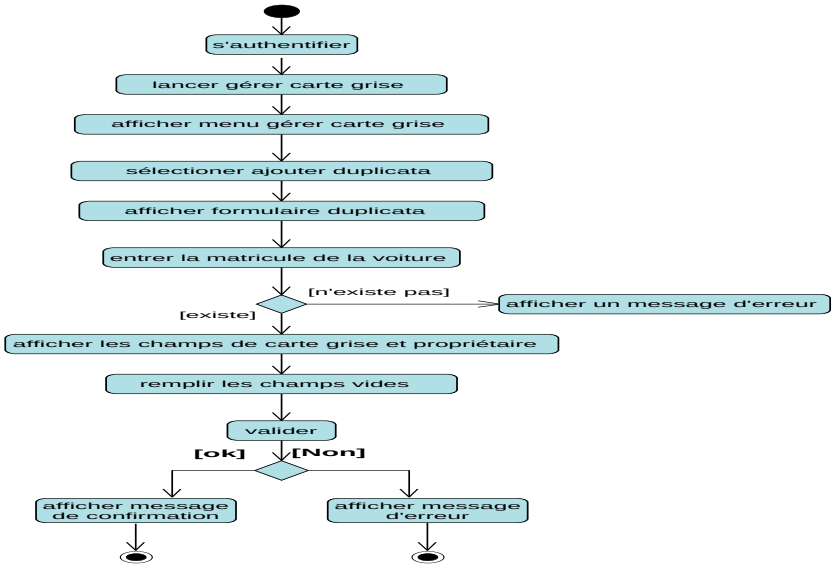

Figure II-8: Diagramme d'activité ajouter duplicata

Annuler Gage/Opposition/Incessibilité

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Annuler Gage/Opposition/Incessibilité.**
Acteur principale	L'employé.
Objectif	L'employé veut annuler une de ces cas Gage/Opposition/Incessibilité d'une carte grise qui existe déjà.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé à annuler une de ces cas Gage/Opposition/Incessibilité avec succès.
Scénario nominal	1. L'employé lance annuler Gage/Opposition/Incessibilité. 2. Le système affiche un formulaire d'annulation et attend l'employé entre la matricule. 3. L'employé entre la matricule de la voiture concernée. 4. Le système affiche un formulaire avec des champs vides concernant l'une de ces cas Gage/Opposition/Incessibilité et les informations concernant le propriétaire remplis et grisées. 5. L'employé remplis les champs vides et valides. 6. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-5 : Cas d'utilisation annuler Gage/Opposition/Incessibilité.

- **Diagramme de séquence**

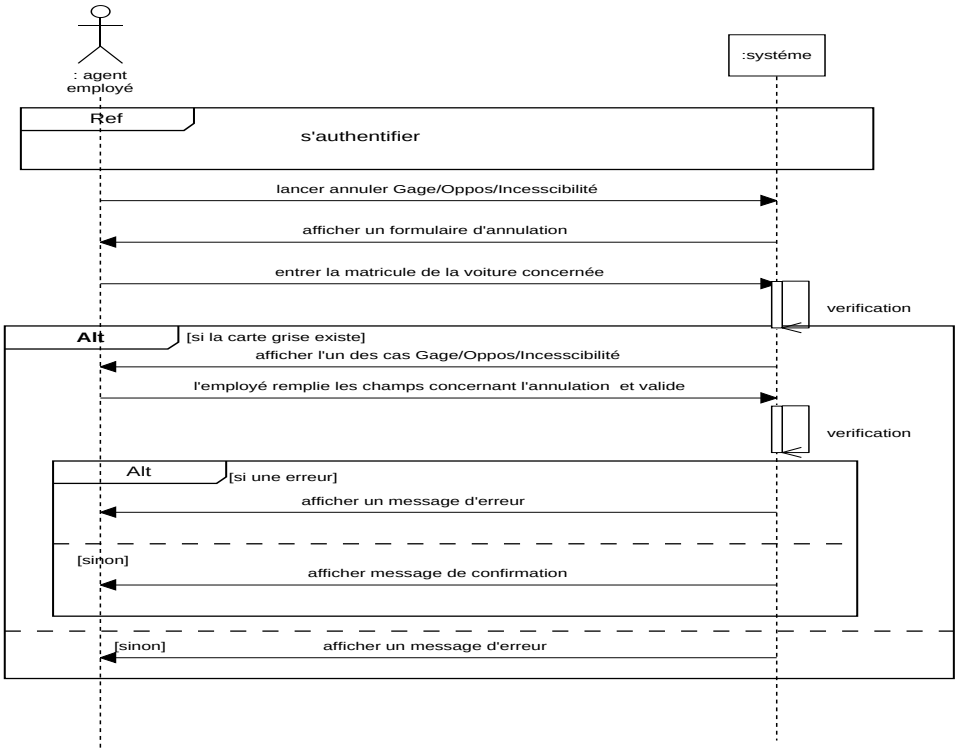

Figure II-9 : Diagramme de séquence système annuler Gage/Opposition/Incessibilité.

- **Diagramme d'activité**

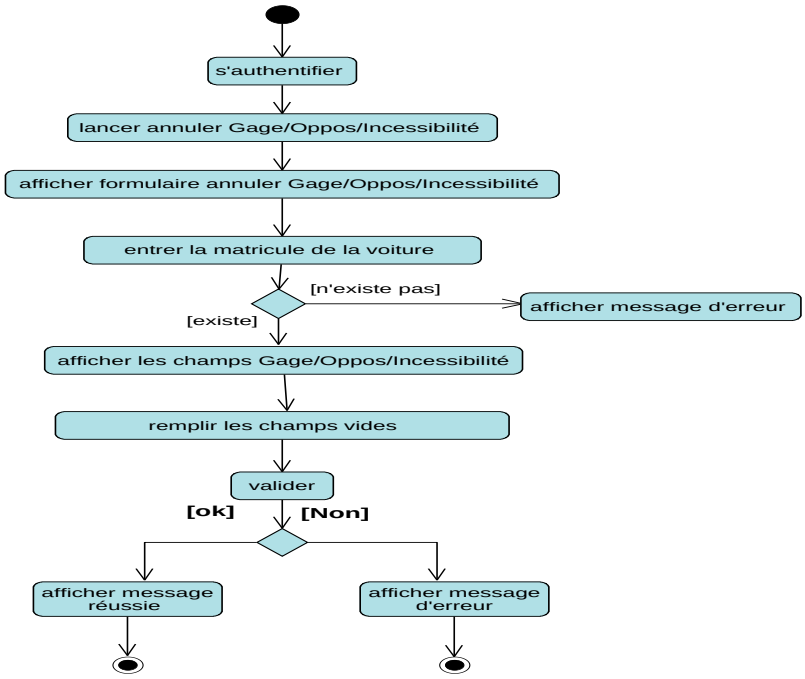

Figure II-10: Diagramme d'activité annuler Gage/Opposition/Incessibilité.

Modifier carte grise

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Modifier carte grise.**
Acteur principale	L'employé.
Objectif	L'employé veut modifier une carte grise déjà existe.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé à modifier la carte grise avec succès.
Scénario nominal	1. L'employé lance gérer carte grise. 2. Le système lui affiche un menu de gestion de carte grise. 3. L'employé choisi modifier une carte grise. 4. Le système affiche un formulaire de modification de carte grise. 5. L'employé entre la matricule de la voiture concernée. 6. Le système affiche les champs de la carte grise remplie avec possibilité de modification des champs. 7. L'employé change les champs qu'il veut dans le formulaire et valide ces changements. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-6 : Cas d'utilisation modifier carte grise

- **Diagramme de séquence**

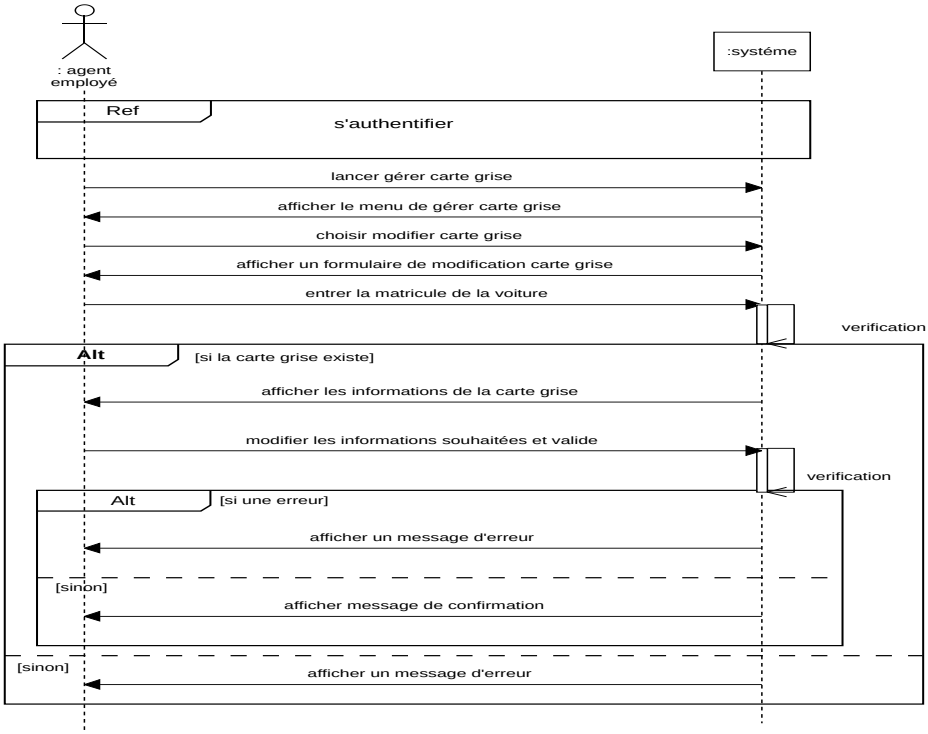

Figure II-11 : Diagramme de séquence système modifier carte grise

- **Diagramme d'activité**

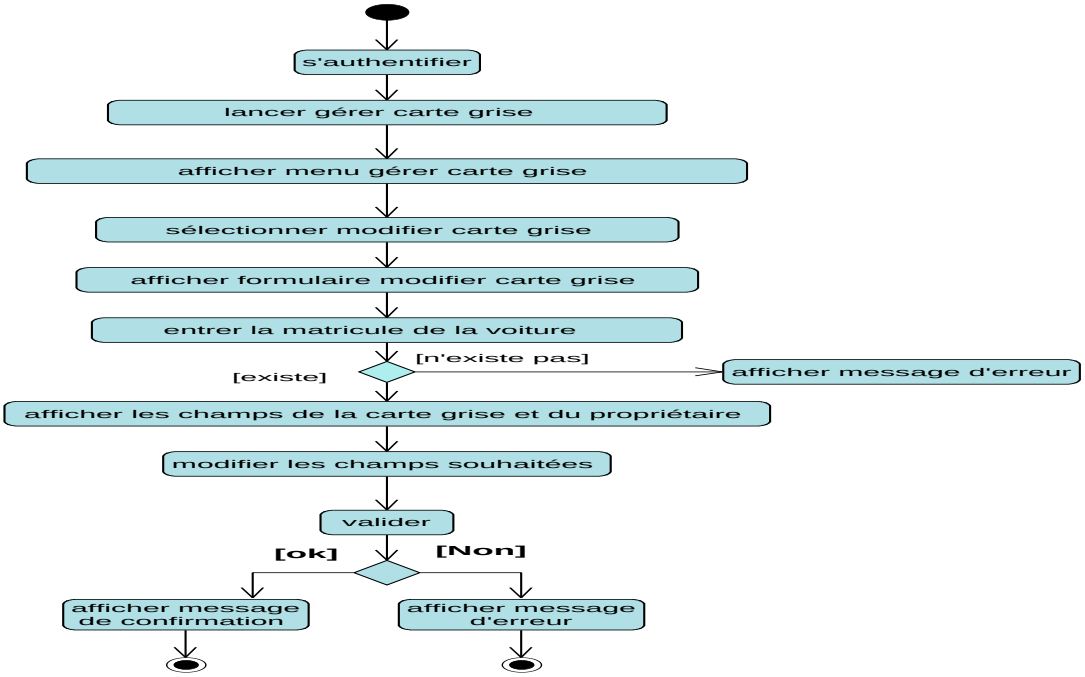

Figure II-12 : Diagramme d'activité modifier carte grise

Radier carte grise

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Radier carte grise.**
Acteur principal	L'employé.
Objectif	Retrait de la circulation d'un véhicule par radiation de la carte grise.
Pré condition	L'employé doit être authentifie.
Poste condition	Carte grise radiée avec réussite.
Scénario nominal	1. L'employé lance gérer carte grise. 2. Le système lui affiche un menu de gestion de carte grise. 3. L'employé choisi radiation carte grise. 4. Le système lui affiche un formulaire de radiation de carte grise. 5. L'employé entre la matricule de la voiture concernée. 6. Le système affiche les champs concernant le propriétaire remplis et grisées et les champs concernant la radiation vides. 7. L'employé remplis les champs vides et valide. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-7 : Cas d'utilisation radier carte grise

- **Diagramme de séquence**

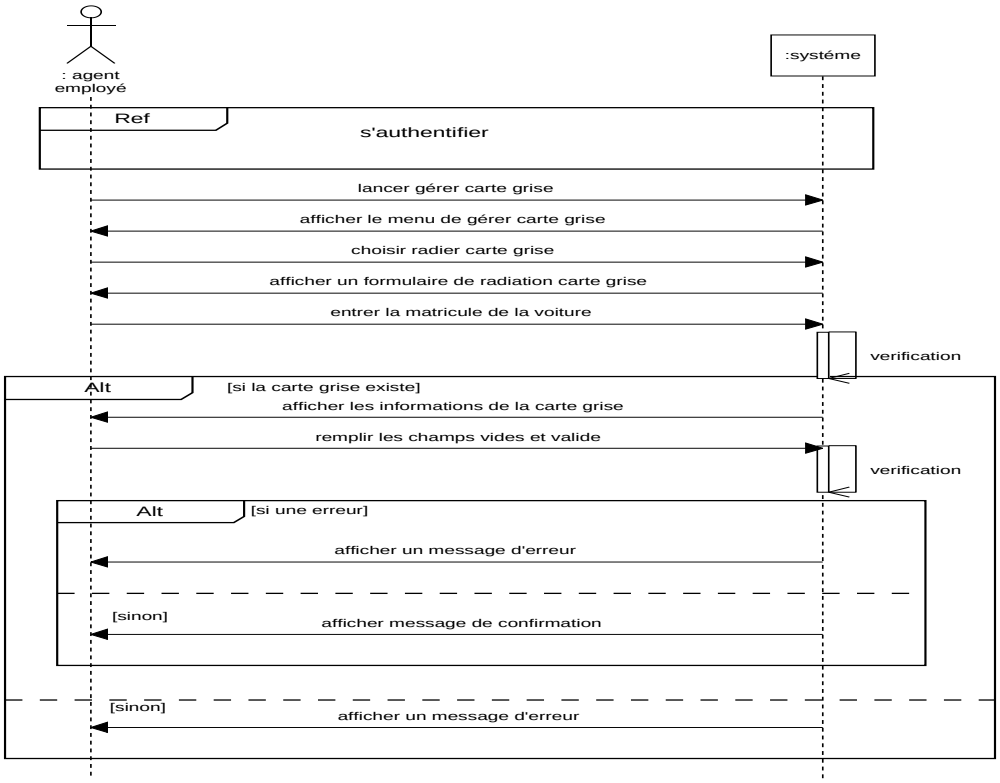

Figure II-13 : Diagramme de séquence système radier carte gris

- **Diagramme d'activité**

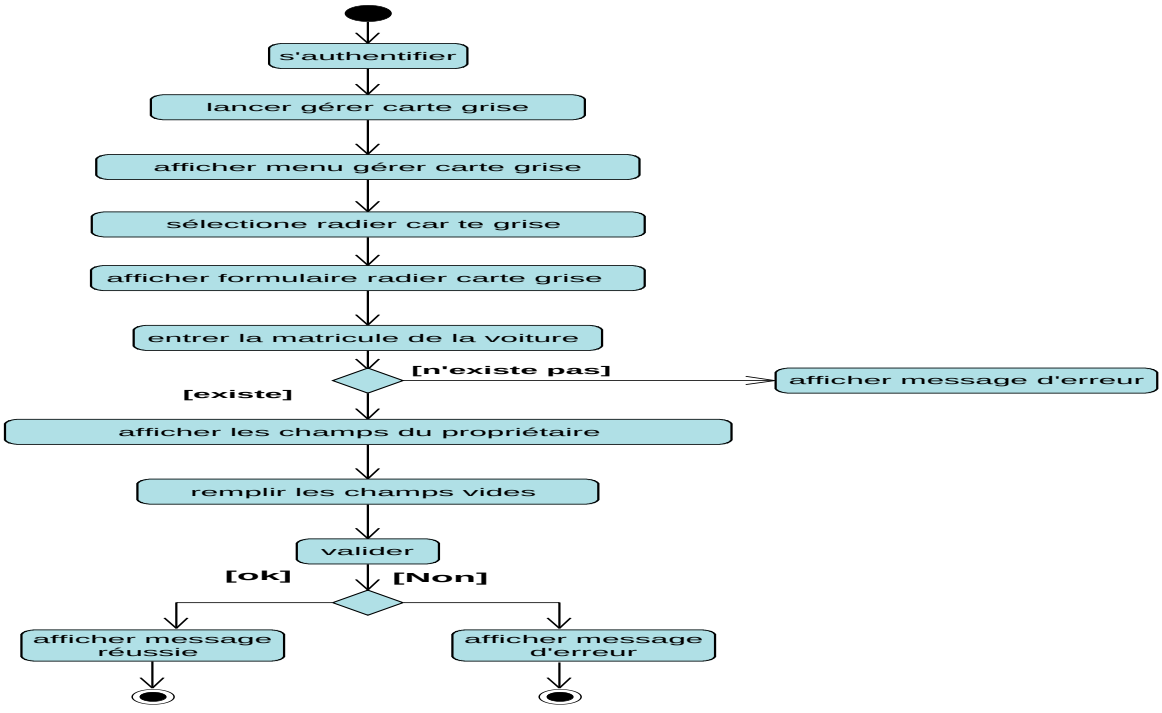

Figure II-14 : Diagramme d'activité radier carte grise

Ajouter vente locale

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Ajouter vente locale.**
Acteur principal	L'employé.
Objectif	La nouvelle carte grise d'une vente interne est ajoutée à la liste des cartes grises.
Pré condition	L'employé doit être authentifié.
Poste condition	Ajout d'une nouvelle vente locale avec succès.
Scénario nominal	1. L'employé lance gérer vente locale. 2. Le système lui affiche un menu de gestion de vente locale. 3. L'employé choisi ajouter vente locale. 4. Le système lui affiche le formulaire d'une vente locale. 5. L'employé entre la matricule de la voiture concernée. 6. Le système lui affiche les champs concernant l'ancien propriétaire remplis et grisées et les champs concernant le nouveau propriétaire plus les champs concernant la vente locale vide. 7. L'employé remplis les champs vides et valides. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-8 : Cas d'utilisation ajouter vente locale

- **Diagramme de séquence**

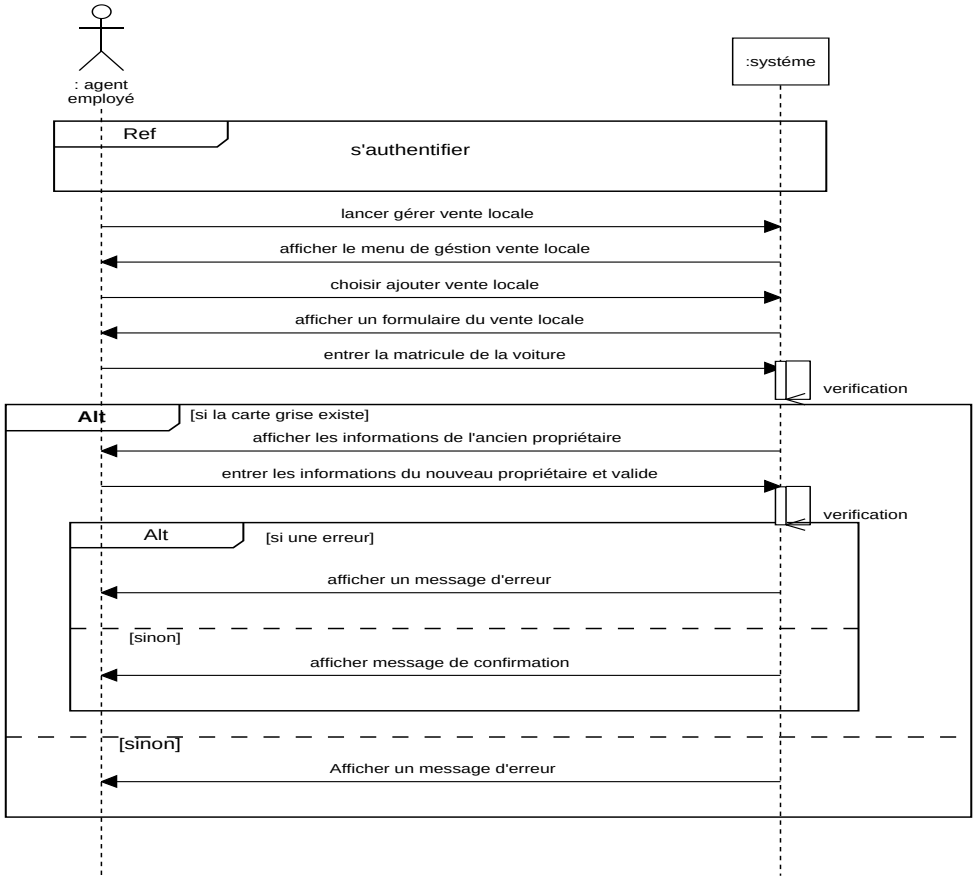

Figure II-15 : Diagramme de séquence système ajouter vente locale

- **Diagramme d'activité**

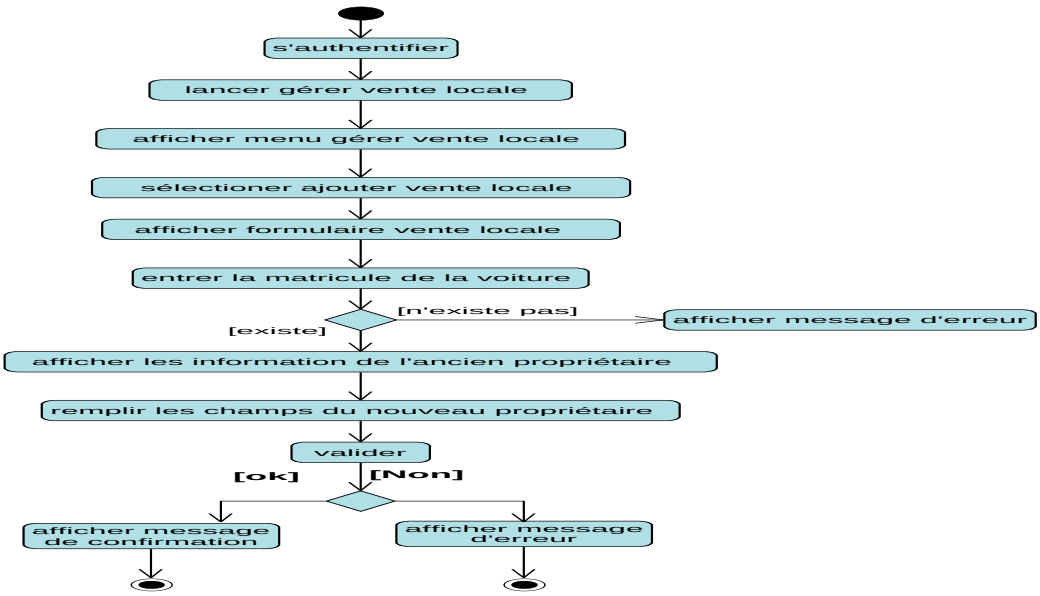

Figure II-16 : Diagramme d'activité ajouter vente locale

s

Annuler vente locale

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Annuler vente locale.**
Acteur principal	L'employé.
Objectif	Annuler une vente interne déjà créé.
Pré condition	L'employé doit être authentifié.
Poste condition	Annulation d'une vente interne avec succès.
Scénario nominal	1. L'employé lance gérer vente locale. 2. Le système lui affiche un menu de gestion de vente locale. 3. L'employé choisi annuler vente locale. 4. Le système lui affiche le formulaire d'annulation de vente locale. 5. L'employé entre la matricule de la voiture concernée. 6. Le système lui affiche les champs concernant l'ancien propriétaire remplis et grisées et les champs concernant le nouveau propriétaire plus les champs concernant la vente locale vides. 7. L'employé remplis les champs vides et valide. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	1-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-9 : Cas d'utilisation annuler vente locale

- **Diagramme de séquence**

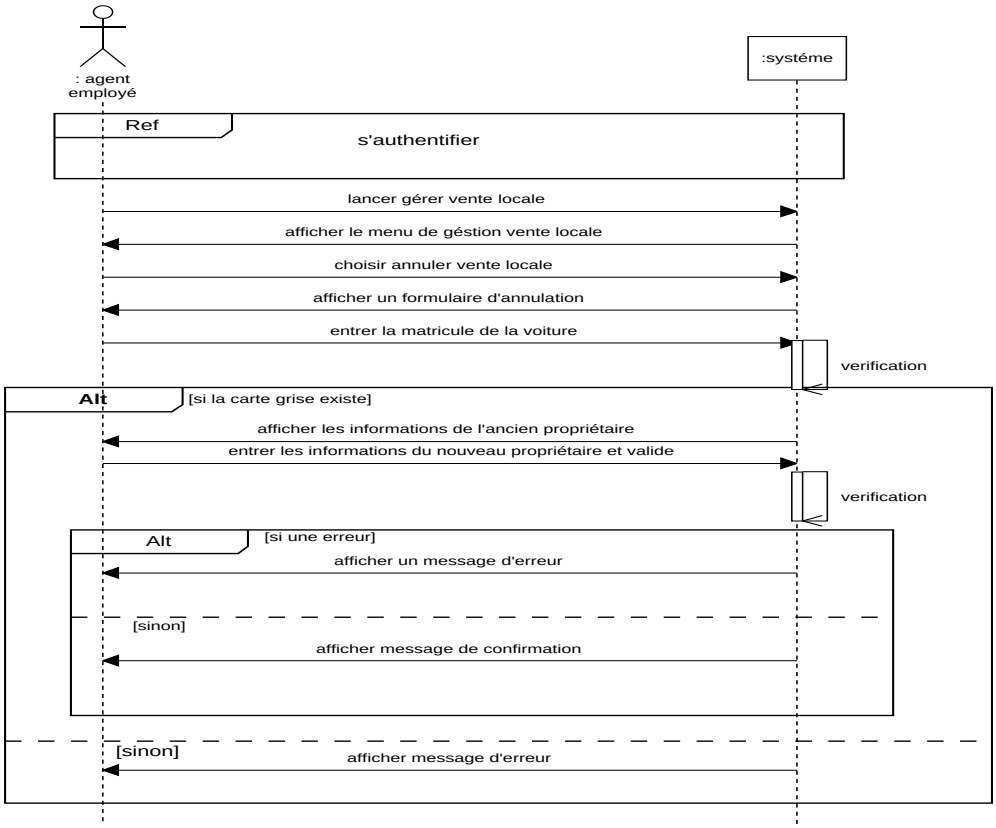

Figure II-17 : Diagramme de séquence système annuler vente locale

- **Diagramme d'activité**

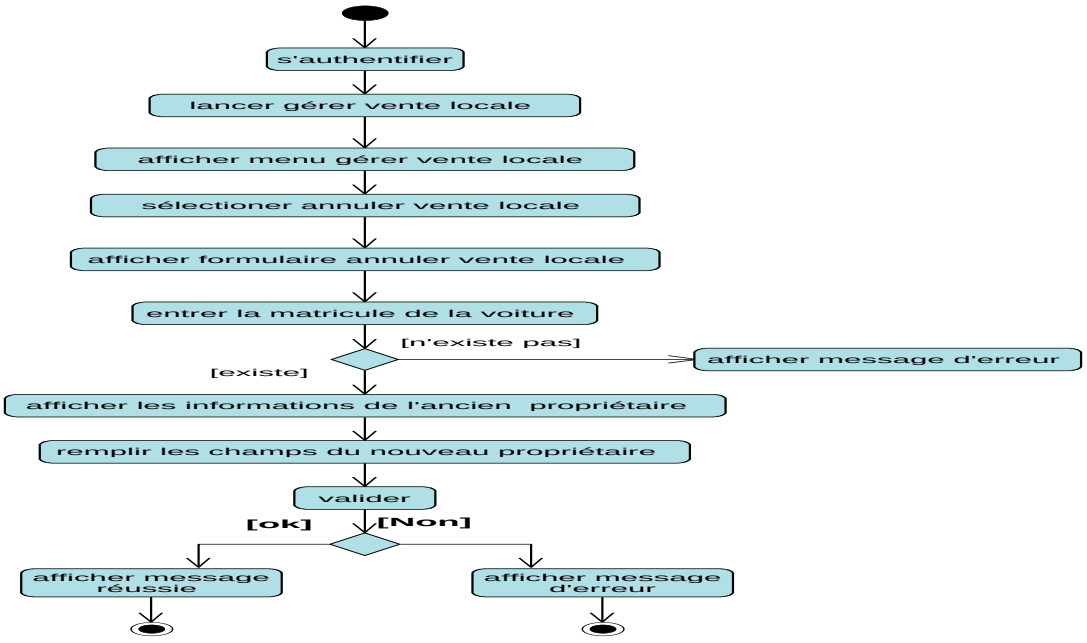

Figure II-18 : Diagramme d'activité annuler vente locale

Ajouter fiche de contrôle

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Ajouter fiche de contrôle.**
Acteur principal	L'employé.
Objectif	Ajouter une fiche de contrôle à la liste des fiches de contrôle.
Pré condition	L'employé doit être authentifié.
Poste condition	Ajout d'une fiche de contrôle avec succès.
Scénario nominal	1. L'employé lance gérer fiche de contrôle. 2. Le système lui affiche un menu de gestion de fiche de contrôle. 3. L'employé choisi ajouter fiche de contrôle. 4. Le système lui affiche le formulaire d'une fiche de contrôle. 5. L'employé entre la matricule de la voiture concernée. 6. Le système affiche les champs concernant la voiture et le propriétaire remplies et grisées et les champs concernant la fiche de contrôle vides. 7. L'employé remplis les champs vides et valide. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	1-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-10: Cas d'utilisation ajouter fiche de contrôle

- **Diagramme de séquence**

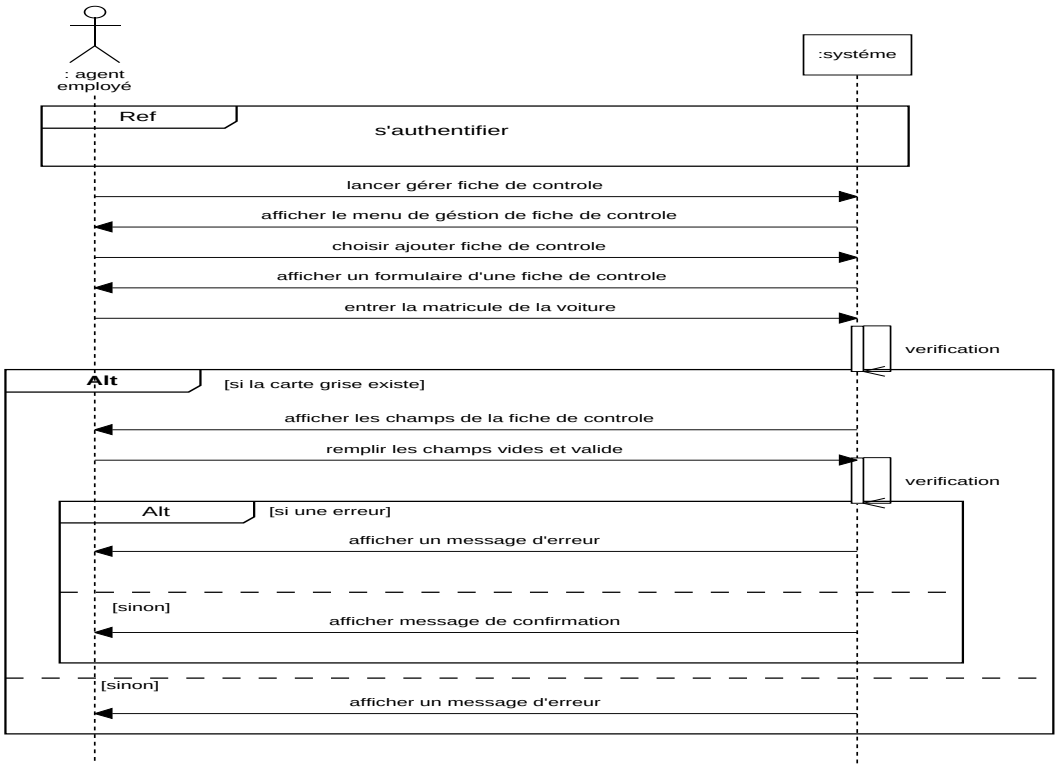

Figure II-19 : Diagramme de séquence système ajouter fiche de contrôle

- **Diagramme d'activité**

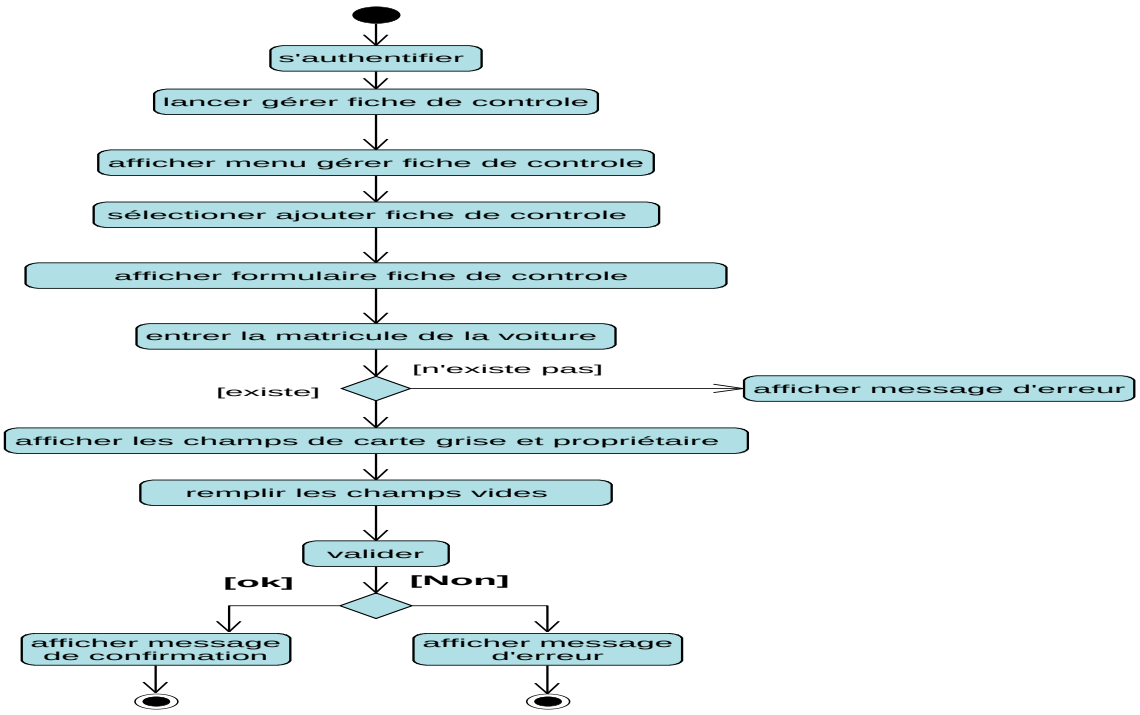

Figure II-20 : Diagramme d'activité ajouter fiche de contrôle

Modifier fiche de contrôle

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Modifier fiche de contrôle.**
Acteur principal	L'employé.
Objectif	L'employé veut modifier une fiche de contrôle déjà existe.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé à modifier la fiche de contrôle avec succès.
Scénario nominal	1. L'employé lance gérer fiche de contrôle. 2. Le système lui affiche un menu de gestion de fiche de contrôle. 3. L'employé choisi modifier une fiche de contrôle. 4. Le système affiche un formulaire de modification de fiche de contrôle. 5. L'employé entre la matricule de la voiture concernée. 6. Le système affiche les champs de la fiche de contrôle remplis avec possibilité de modification des champs. 7. L'employé change les champs qu'il veut dans le formulaire et valide ces changements. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	1.-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-11: Cas d'utilisation modifier fiche de contrôle

- **Diagramme de séquence**

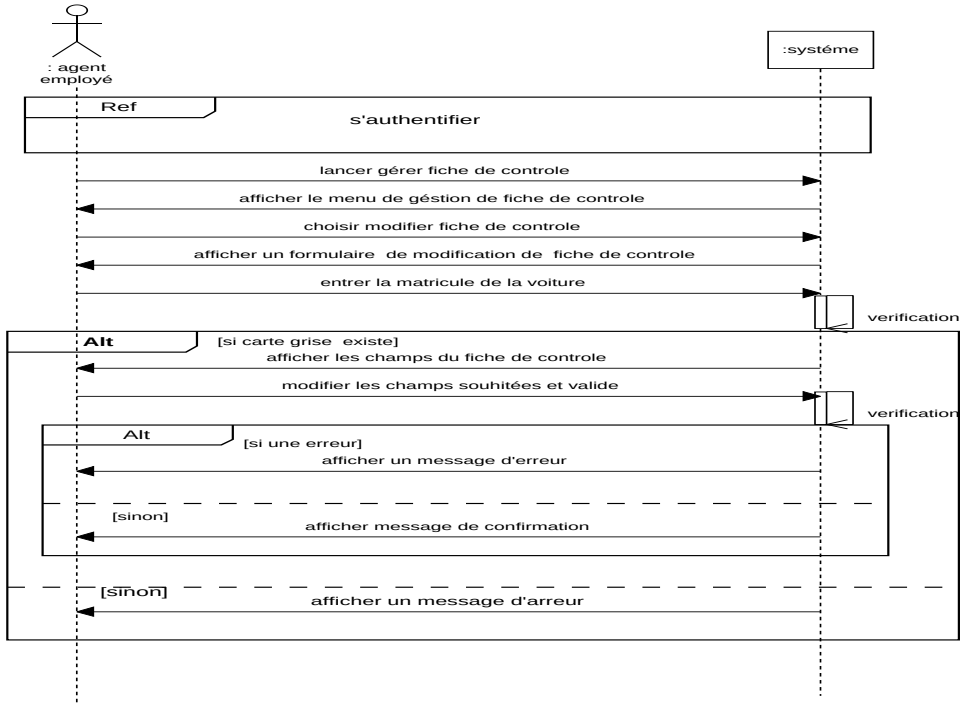

Figure II-21: Diagramme de séquence système modifier fiche de contrôle

- **Diagramme d'activité**

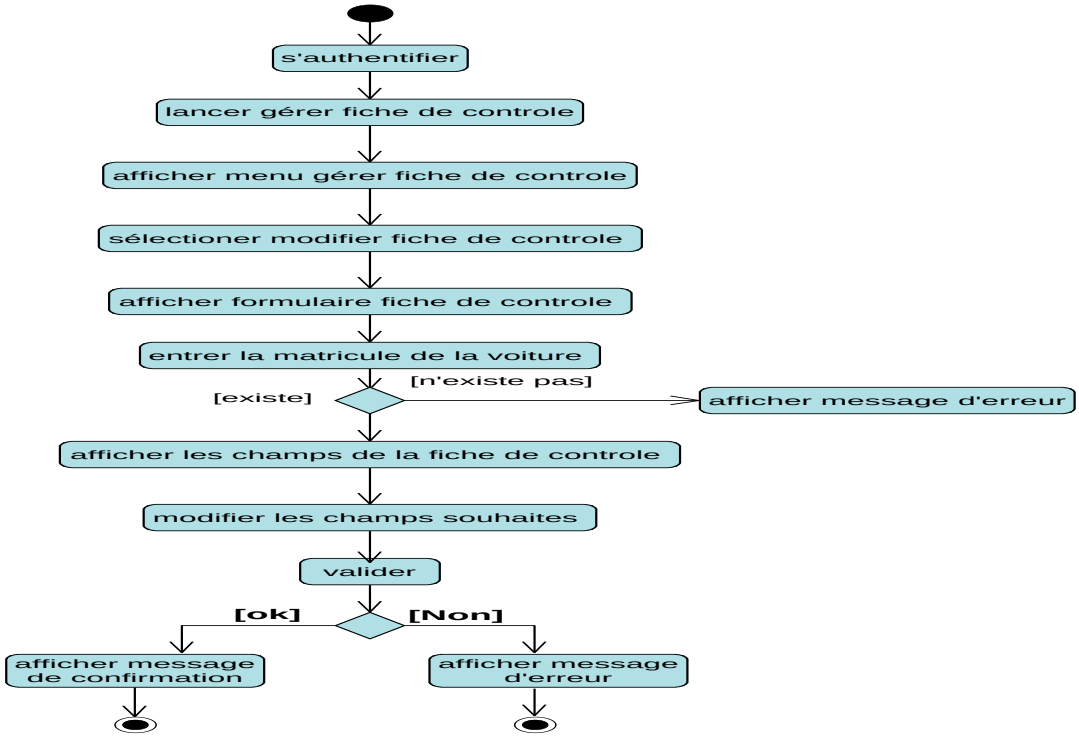

Figure II-22 : Diagramme d'activité modifier fiche de contrôle

Annuler fiche de contrôle

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Annuler fiche de contrôle.**
Acteur principal	L'employé.
Objectif	Annuler une fiche de contrôle déjà créé.
Pré condition	L'employé doit être authentifié.
Poste condition	Annulation d'une fiche de contrôle avec succès.
Scénario nominal	1. L'employé lance gérer fiche de contrôle. 2. Le système lui affiche un menu de gestion de fiche de contrôle. 3. L'employé choisi annuler fiche de contrôle. 4. Le système lui affiche le formulaire d'annuler une fiche de contrôle. 5. L'employé entre la matricule de la voiture concernée. 6. Le système affiche les champs concernant la voiture et le propriétaire plus les champs concernant la fiche de contrôle remplies et grisées. 7. L'employé vérifie les informations puis valide. 8. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	1-Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-12 : Cas d'utilisation annuler fiche de contrôle

- **Diagramme de séquence**

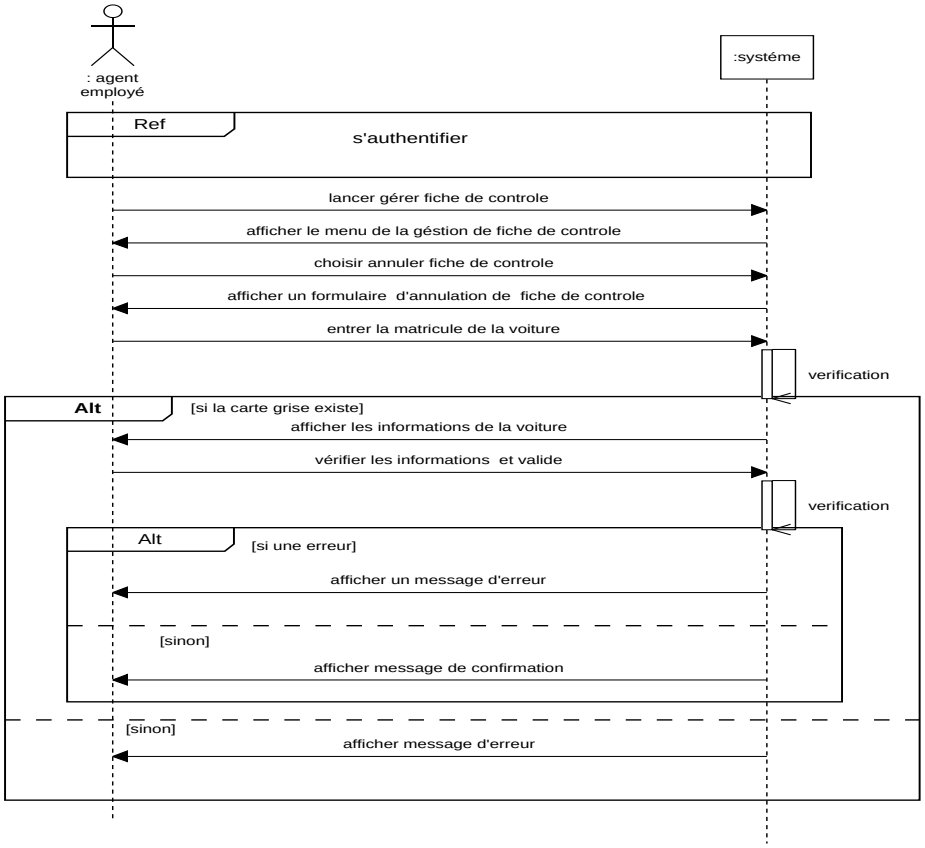

Figure II-23: Diagramme de séquence système annuler fiche de contrôle

- **Diagramme d'activité**

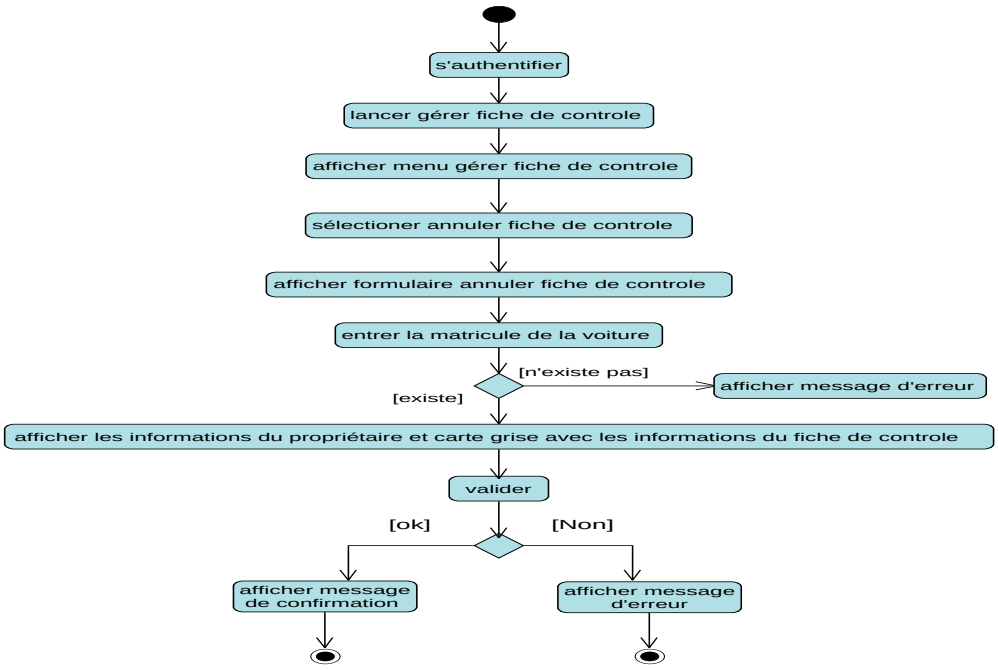

Figure II-24: Diagramme d'activité annuler fiche de contrôle

Etablir avis de mutation

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Etablir avis de mutation.**
Acteur principale	L'employé.
Objectif	L'employé veut établir un nouvel avis de mutation qui n'existe pas.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé a ajouté un nouvel avis de mutation avec succès.
Scénario nominal	1. L'employé lance établir avis de mutation. 2. Le système affiche un formulaire d'avis de mutation. 3. L'employé entre la matricule de la voiture concernée. 4. Le système lui affiche las champs concernant la carte grise remplis et grisées et les champs concernant l'avis de mutation vides. 5. L'employé remplis les champs vides et valide. 6. Le système affiche un message « l'opération est terminée avec succès ».
Scénario alternative	1. Si l'un des champs n'est pas rempli : le système affiche un message «il faut remplir tous les champs ».
Scénario d'exception	

Tableau II-13: Cas d'utilisation établir avis de mutation

- **Diagramme de séquence**

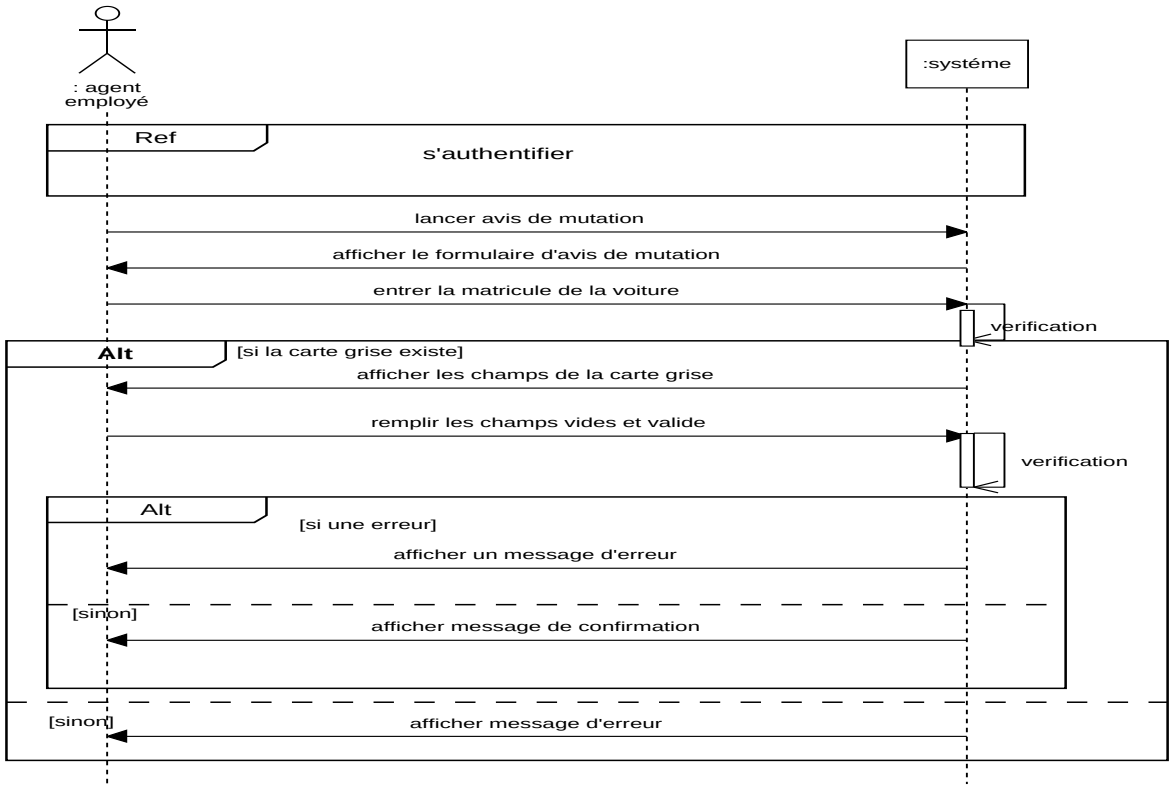

Figure II-25: Diagramme de séquence système établir avis de mutation

- **Diagramme d'activité**

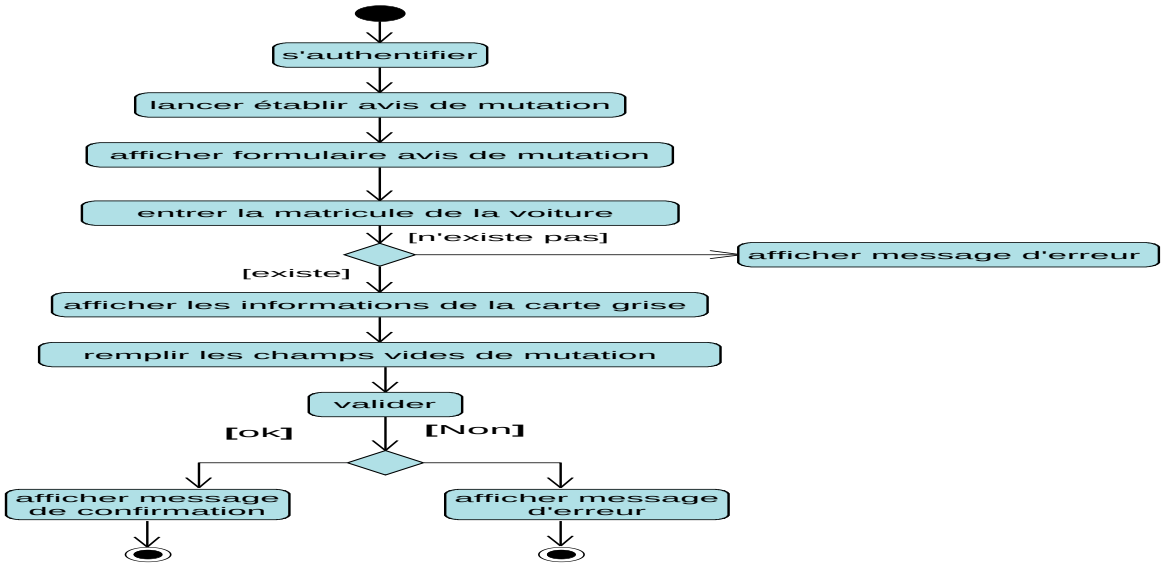

Figure II-26: Diagramme d'activité établir avis de mutation

Rechercher carte grise

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Rechercher carte grise.**
Acteur principale	L'employé.
Objectif	L'employé veut rechercher une carte grise.
Pré condition	L'employé doit être authentifié.
Post condition	L'employé à trouver la carte grise.
Scénario nominal	1. L'employé lance rechercher carte grise soi par matricule, par numéro de châssis ou bien par nom et prénom. 2. Le système affiche le formulaire de recherche. 3. L'employé entre les informations de la carte grise. 4. Le système affiche une page de résultats. 5. L'employé sélectionne une carte grise. 6. Le système lui présente la carte grise détaillé.
Scénario alternative	Si l'employé appuie sur annuler le système revient à la page d'accueil.
Scénario d'exception	

Tableau II-14: Cas d'utilisation rechercher carte grise

- **Diagramme de séquence**

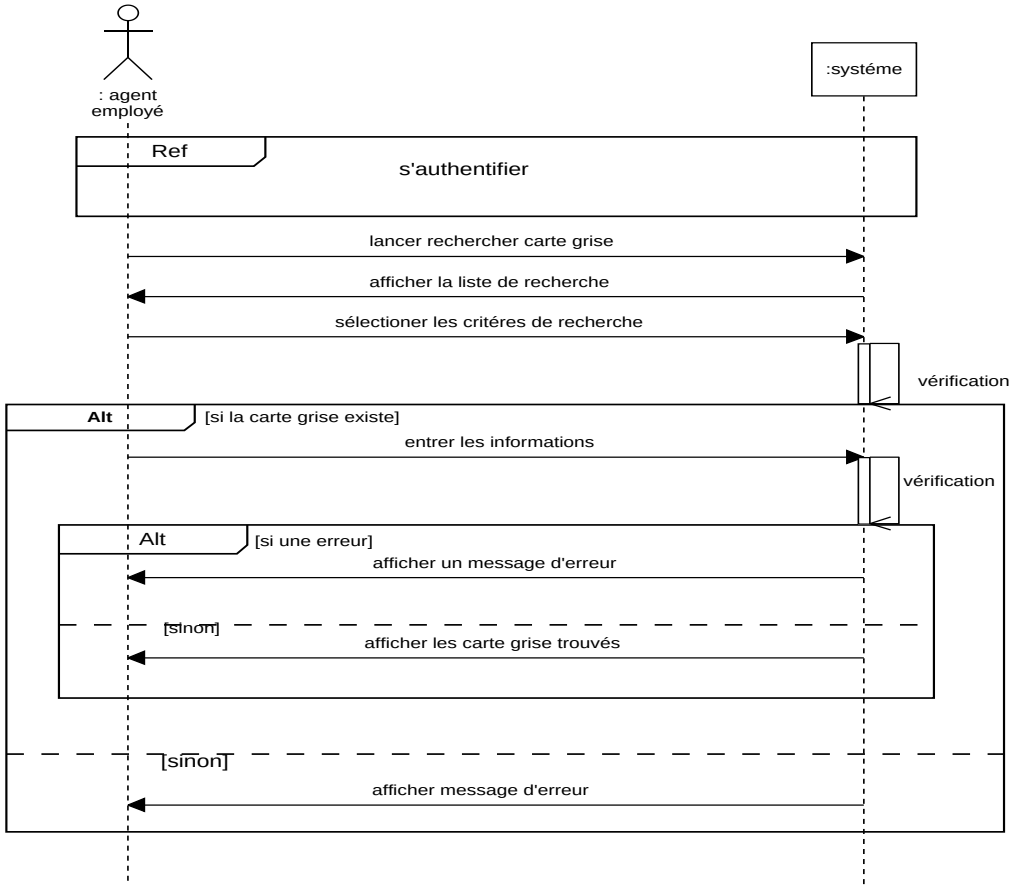

Figure II-27: Diagramme de séquence système rechercher carte grise

- **Diagramme d'activité**

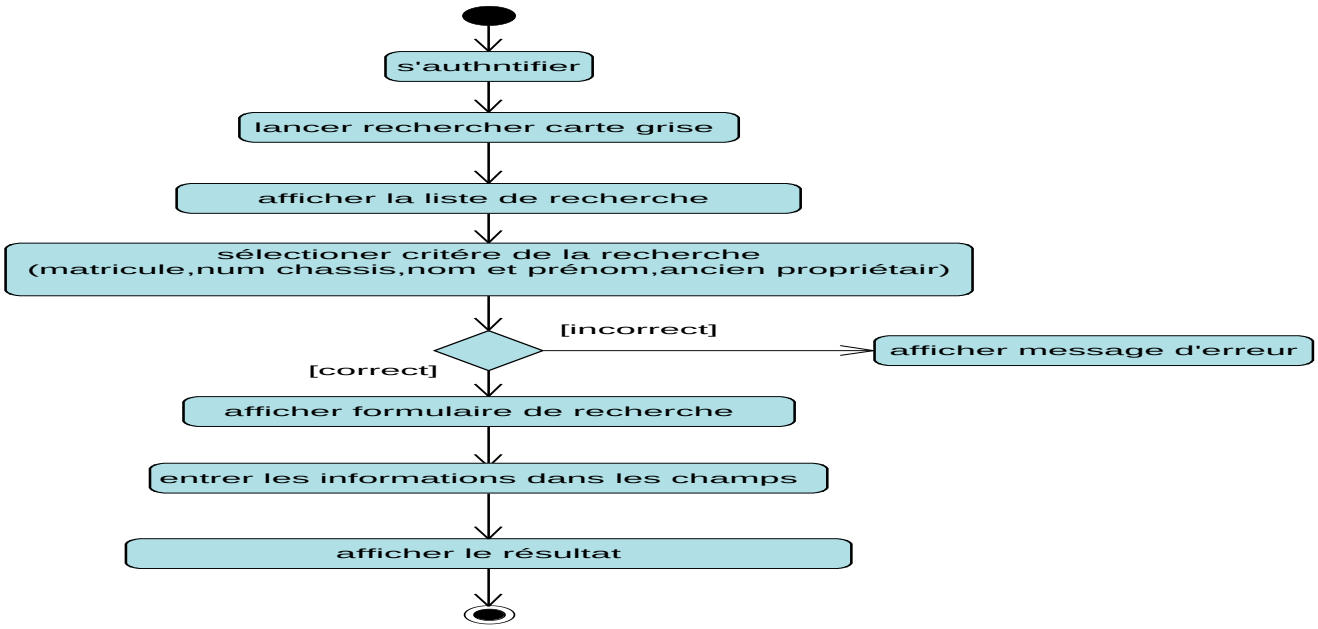

Figure II-28: Diagramme d'activité rechercher carte grise

2.3. Identification des classes candidates

Cette phase va préparer la modélisation orientée objet en aidant à trouver les classes principales du futur modèle statique d'analyse. La technique utilisée pour identifier les classes candidates est la suivante :

Chercher les noms communs importants dans les descriptions textuelles des cas d'utilisation.

Vérifier les propriétés « objet » de chaque concept (identité, propriétés, comportement), puis définir ses responsabilités.

2.3.1. Définition

Une responsabilité est une sorte de contrat, ou obligation, pour une classe. Elle se place à un niveau d'abstraction plus élevé que les attributs ou les opérations. **[6]**

2.3.2. La liste des classes candidates

La liste des classes candidates
• **Carte grise** • **Duplicata** • **Radiation** • **Propriétaire** • **Fiche de contrôle** • **Fiche de confirmation** • **Avis de mutation** • **Gage** • **Opposition** • **Incessibilité** • **Agent**

Figure II-29: La liste des classes candidates

2.3.3. Responsabilités des classes

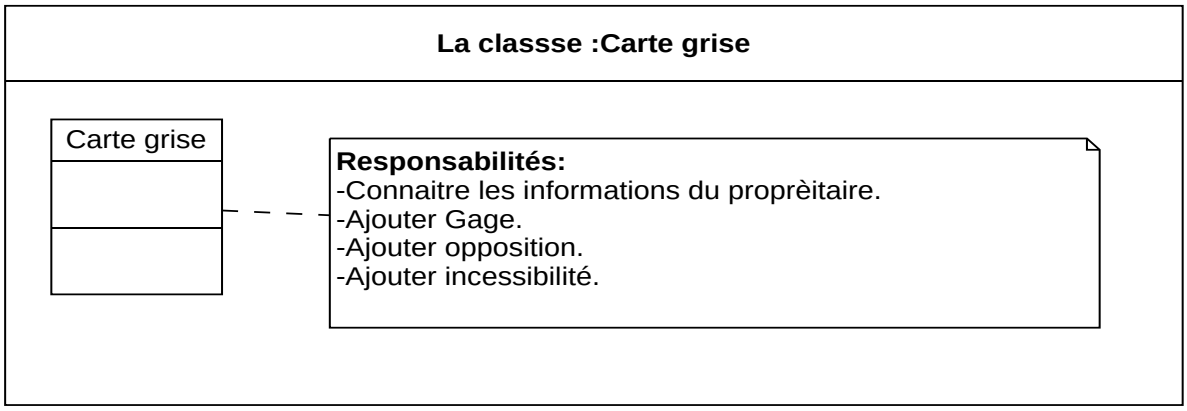

Figure II-30: Responsabilité de la classe carte grise

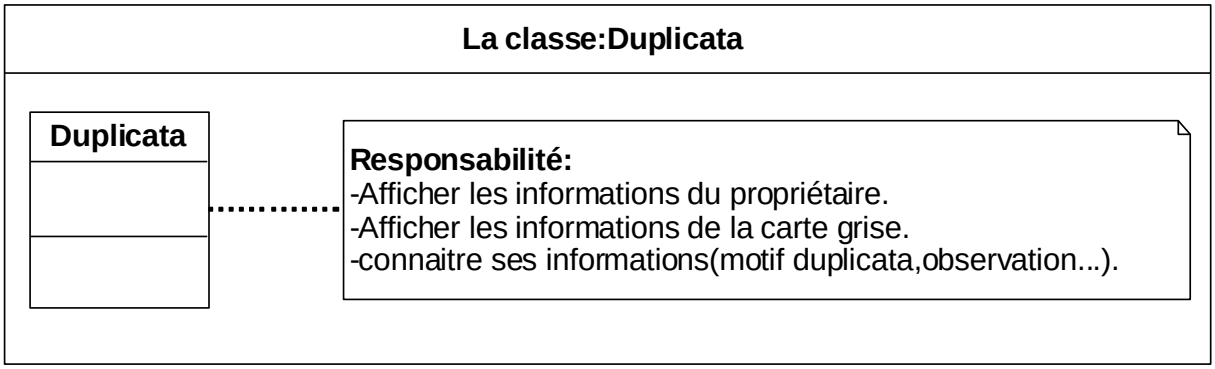

Figure II-31: Responsabilité de la classe duplicata

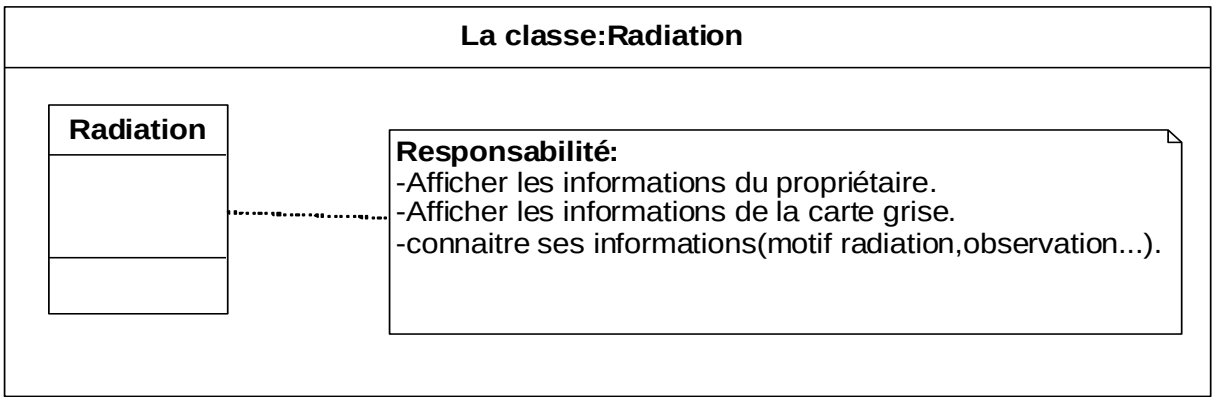

Figure II-32: Responsabilité de la classe radiation

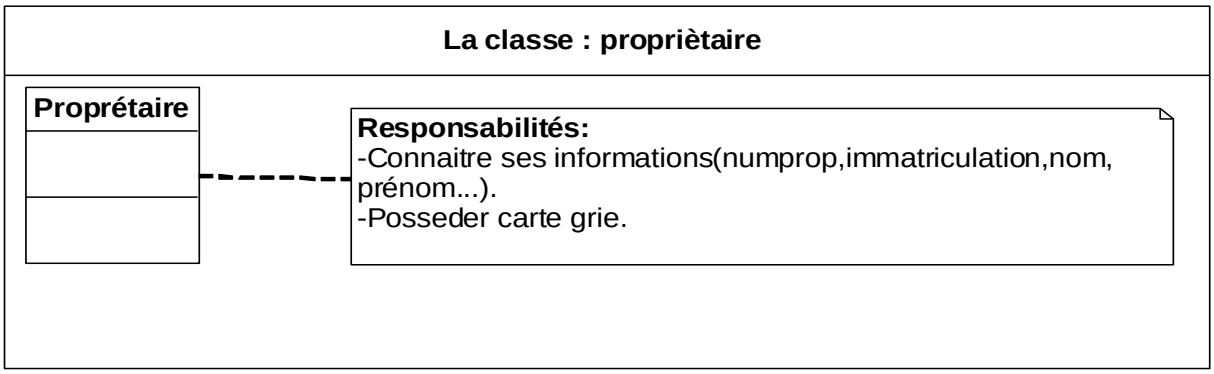

Figure II-33: Responsabilité de la classe propriétaire

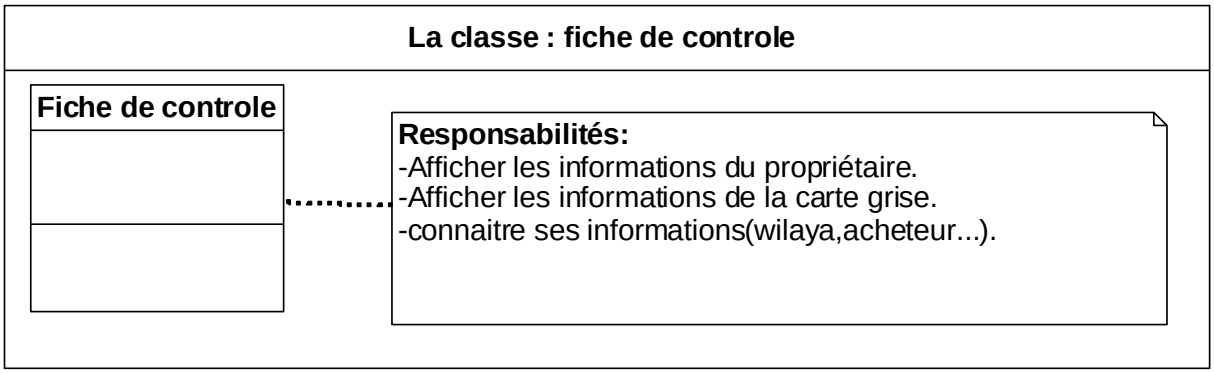

Figure II-34: Responsabilité de la classe fiche de contrôle

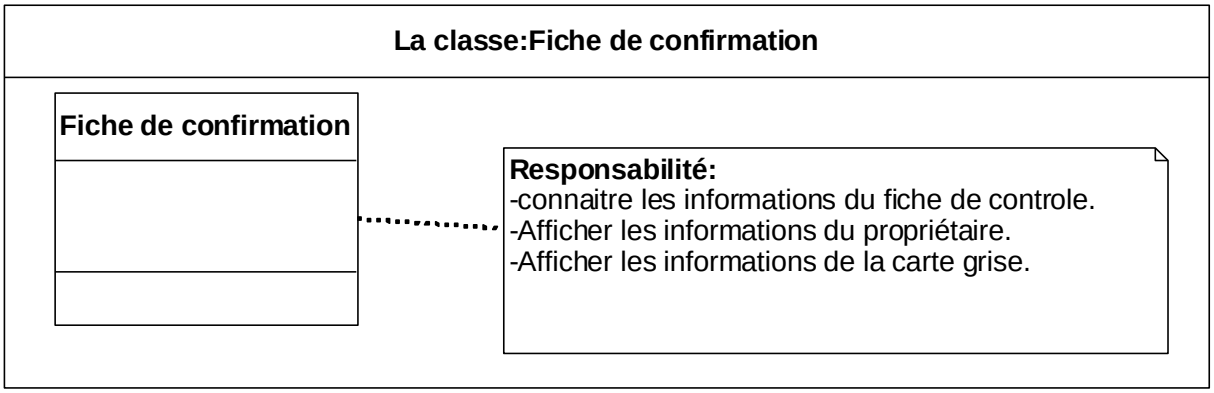

Figure II-35: Responsabilité de la classe fiche de confirmation

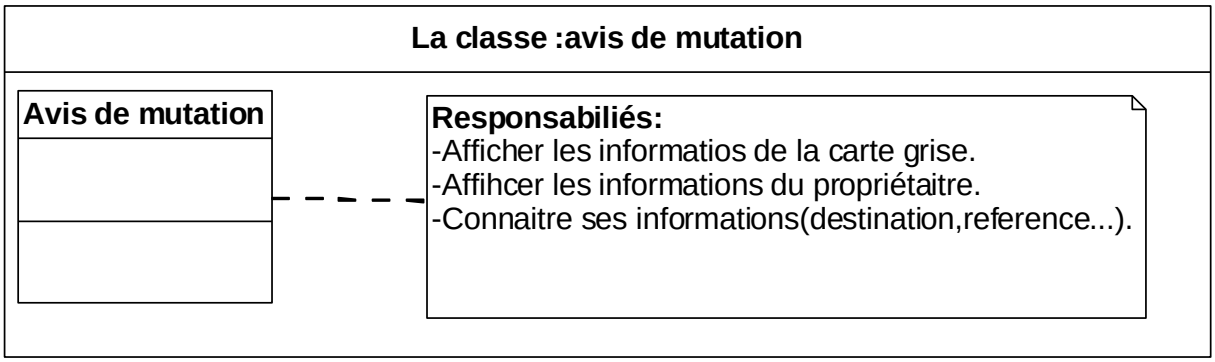

Figure II-36 : Responsabilité de la classe avis de mutation

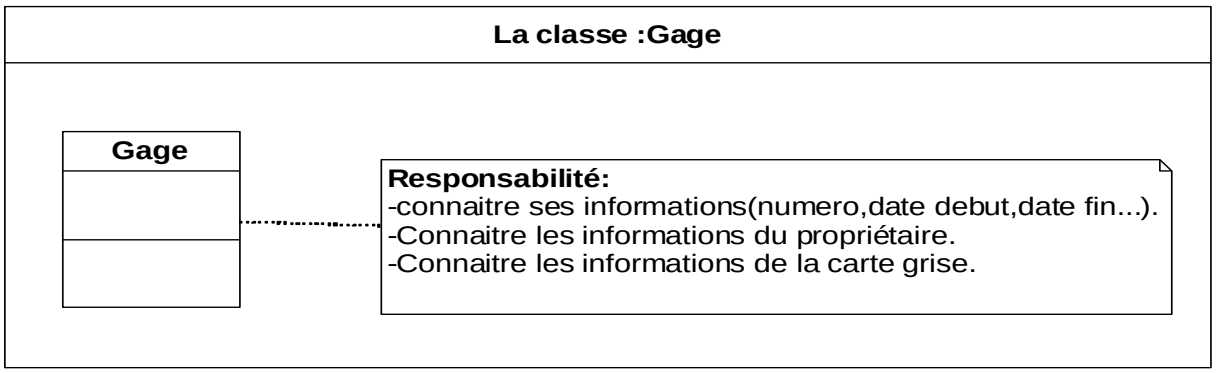

Figure II-37 : Responsabilité de la classe gage

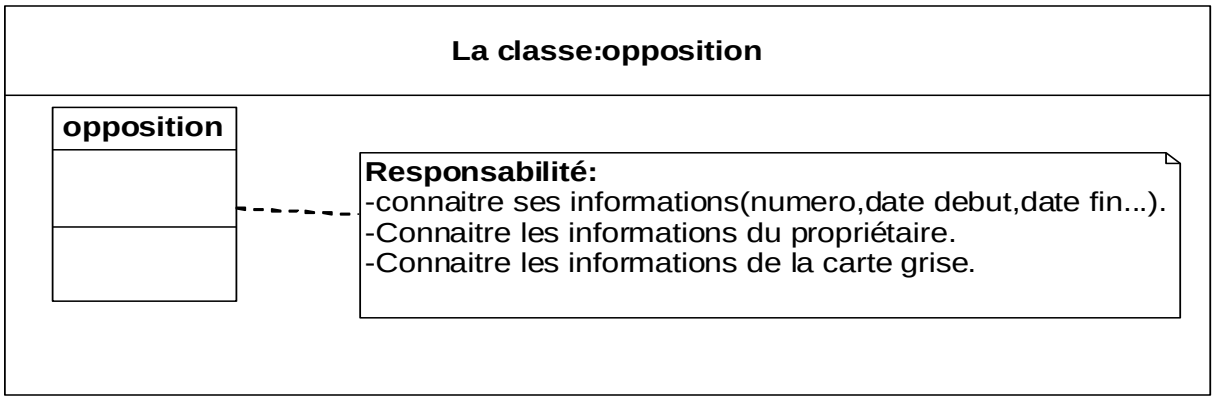

Figure II-38 : Responsabilité de la classe opposition

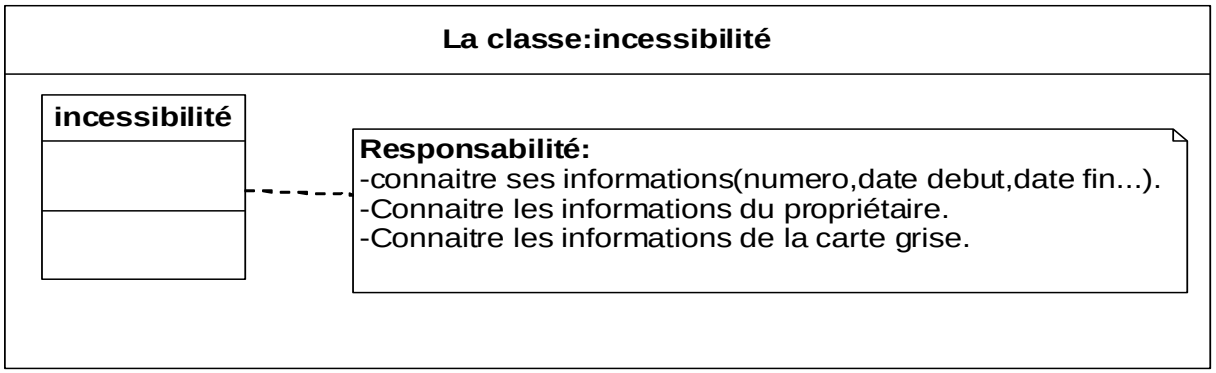

Figure II-39 : Responsabilité de la classe incessibilité

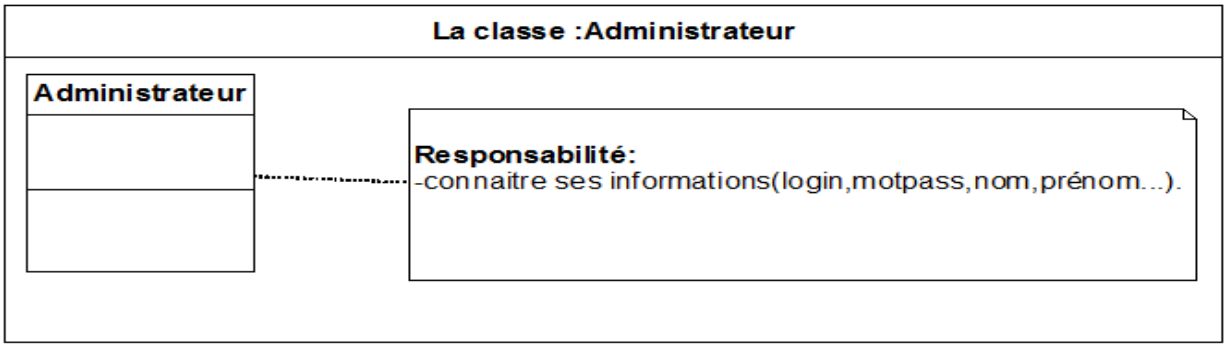

Figure II-40 : Responsabilité de la classe administrateur

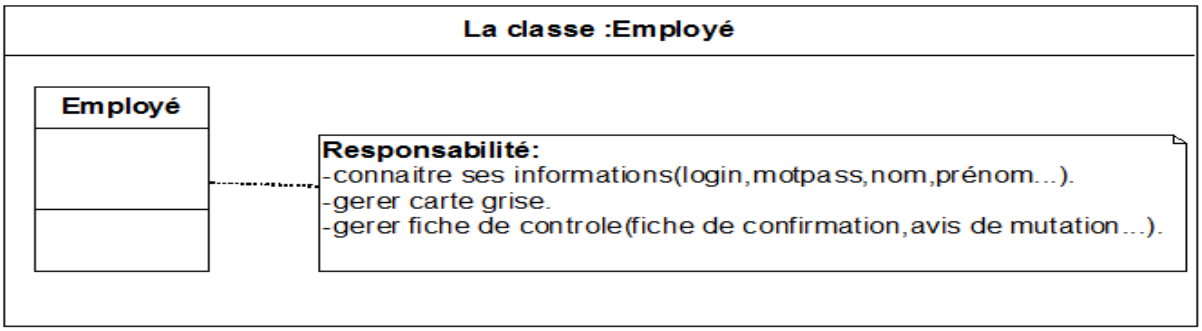

Figure II-41: Responsabilité de la classe employé

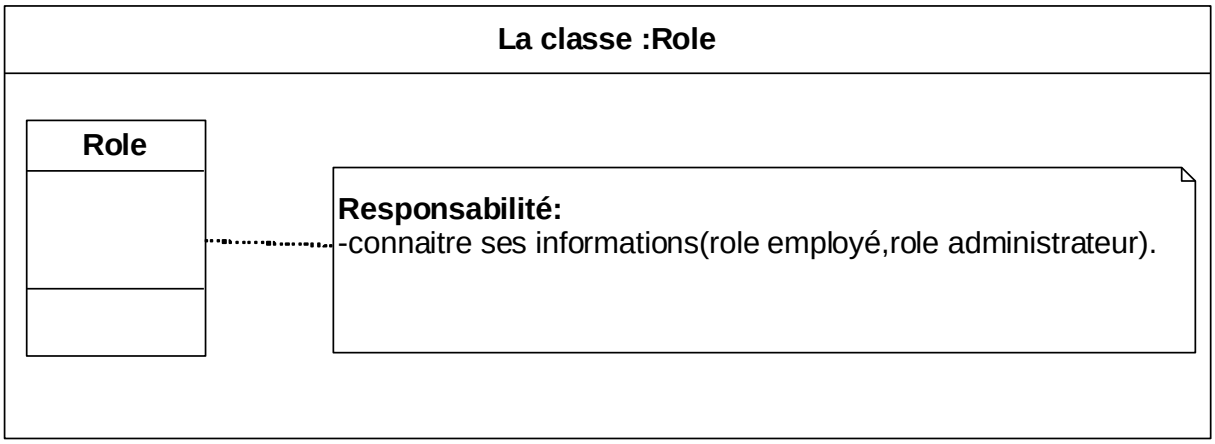

Figure II-42: Responsabilité de la classe rôle

3. Capture des besoins techniques

La capture des besoins techniques est la première étape de la branche droite du cycle en Y, c'est une étape de prise en compte des contraintes techniques et logicielles.

Elle se fait selon deux points de vue :

- La spécification technique du point de vue matériel.
- La capture des spécifications logicielles.

3.1. Spécification technique du point de vue matériel

Le matériel à mettre dans un système dépend de

- **Style d'architecture en niveaux :** qui spécifie le nombre de niveaux géographiques et organisationnels où vont se situer les environnements d'exécution du système. L'architecture peut être à deux niveaux, à trois niveaux, ou multi niveaux dans une organisation plus complexe. **[6]**
- **Contraintes techniques :** peuvent se manifester de différentes manières :
  - La sécurité.
  - La disponibilité
  - La performance.

3.2. Spécification d'architecture

L'application réalisée repose sur une architecture 2 tiers. Cette architecture est composée de deux type da liaison, une liaison pair à pair qui correspond à la communication entre le serveur local et le serveur central ou bien la communication entre deux serveurs locaux, et une autre liaison client-serveur correspond à la communication entre une machine cliente et un serveur local.

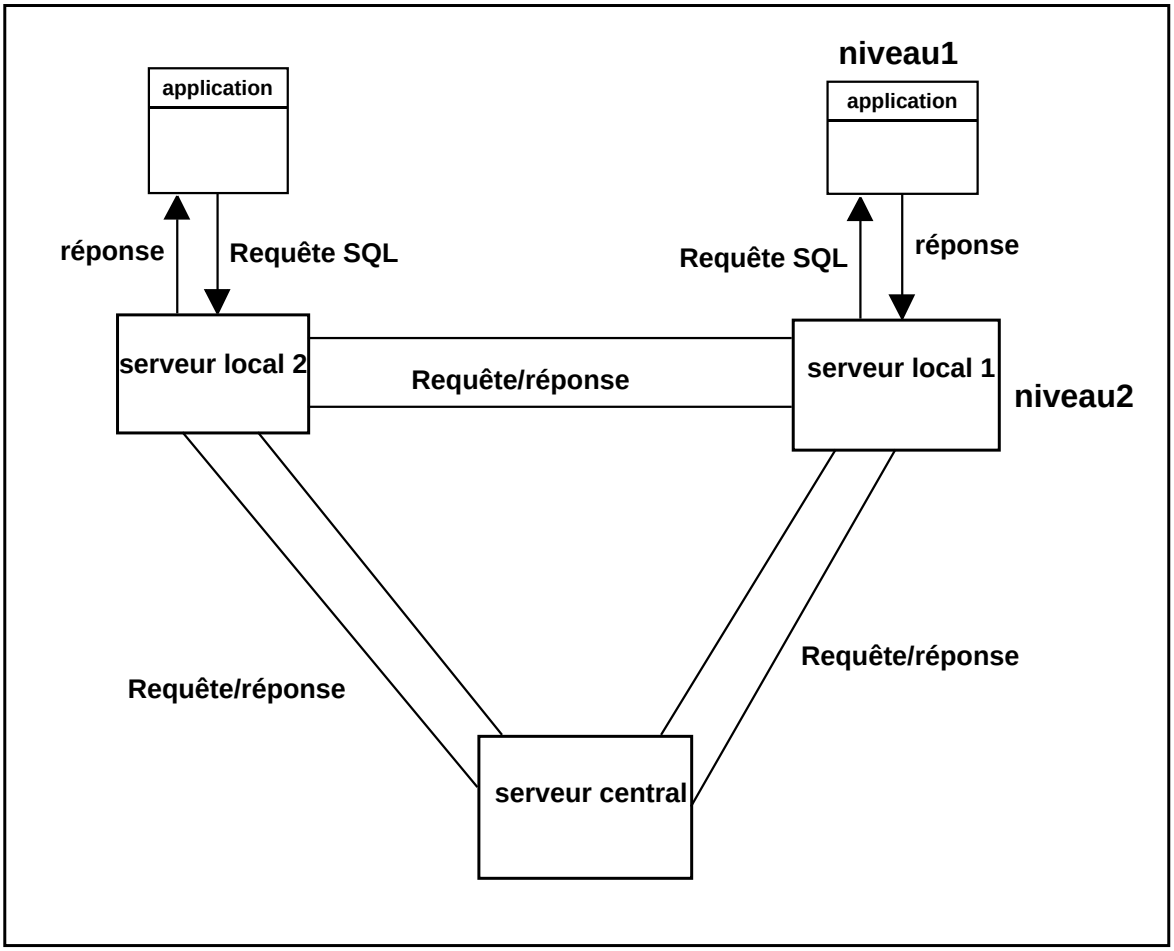

Figure II-43 : Architecture 2 niveaux de notre système

3.3. Capture des spécifications logicielles

Ce sont des fonctionnalités techniques que le système va s'assurer à l'utilisateur indépendamment des fonctionnalités métier.

- **Les exploitants :** Appelés aussi « acteurs techniques » du système, ils sont les acteurs qui bénéficient des fonctionnalités techniques du système : c'est l'administrateur.

Dans notre système les exploitants chargés de faire :

- La gestion de la sécurité.

3.3.1. Identification des cas d'utilisation techniques

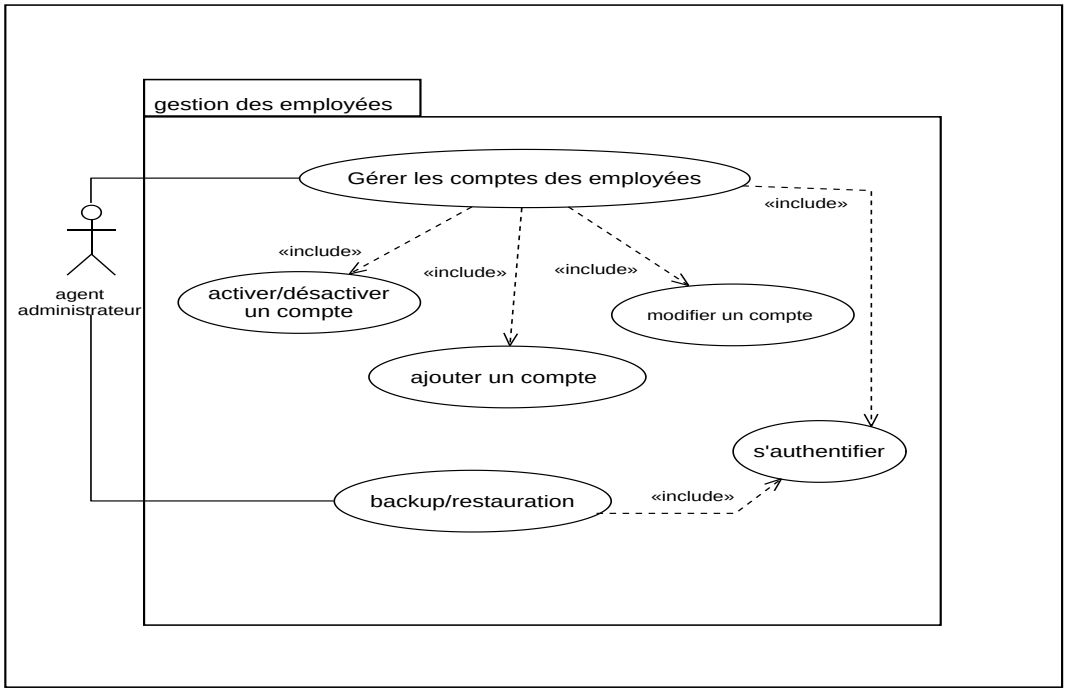

Figure II-44: Cas d'utilisation gestion des employées

3.3.2. Description des cas d'utilisation techniques

S'authentifier

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**S'authentifier.**
Acteur principal	Employé, Administrateur.
Objectif	Permet de saisir les informations (login, mot de passe) des comptes clients ou administrateurs du site.
Pré condition	L'existence d'un compte employé (ou administrateur).
Poste condition	Accès à l'application succès.
Scénario nominal	1-L'administrateur(ou employé) demande d'accéder à l'application. 2-Le système lui affiche le formulaire d'authentification. 3-L'employé/administrateur remplit le formulaire (nom d'utilisateur, mot de passe). 4-L'employé/administrateur valide les informations saisies. 5- Le système vérifie les informations saisies par l'utilisateur et le renvoie vers la page voulue.
Scénario alternative	-Le visiteur n'a pas remplie certains champs obligatoires. Le système détecte et lui propose de les remplir à nouveau. - Un message d'erreur est affiché sur l'écran avise l'employé (ou l'administrateur) que les informations saisies non valides.
Scénario d'exception	

Tableau II-15: Cas d'utilisation s'authentifier

- **Diagramme de séquence**

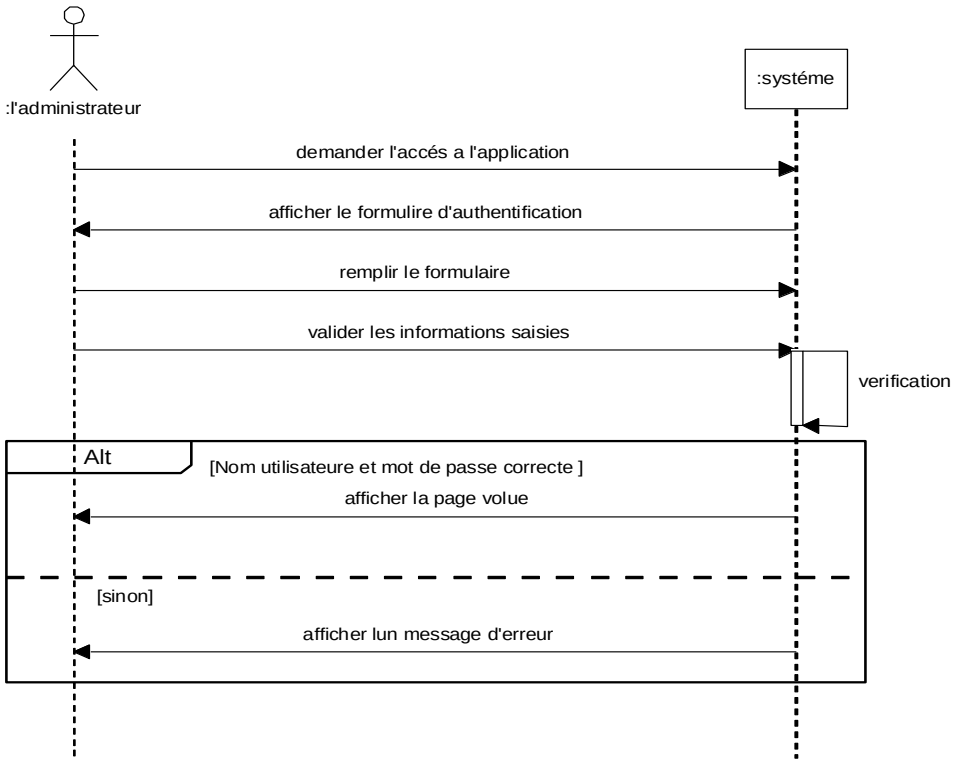

Figure II-45: Diagramme de séquence système s'authentifier

- **Diagramme d'activité**

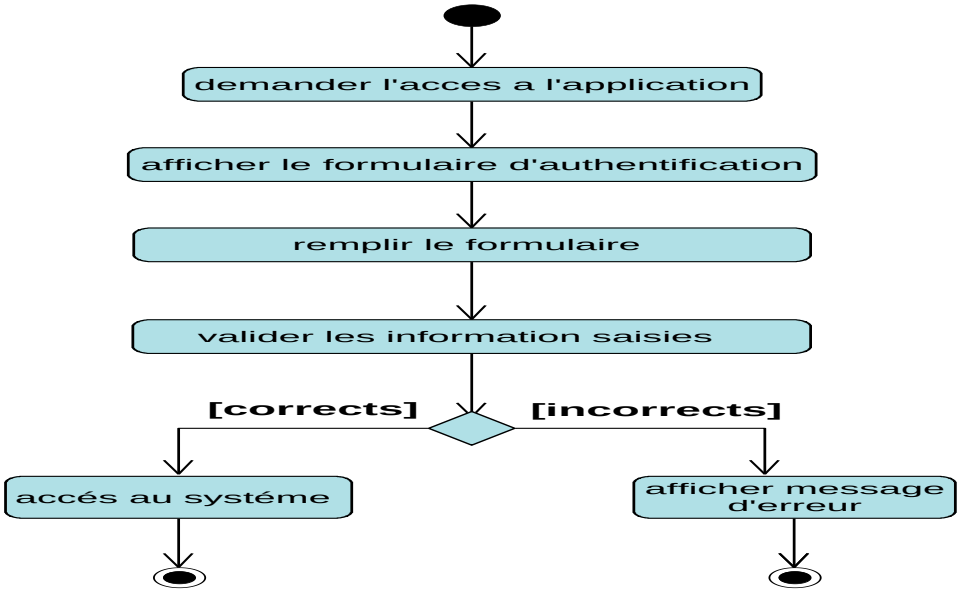

Figure II-46: Diagramme d'activité s'authentifier

Créer un compte

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Créer compte**
Acteur principal	Administrateur.
Objectif	Créer un compte pour qu'un employé soit identifiée par l'application, cet création se fait par le remplissage d'un certain formulaire.
Pré condition	Administrateur s'authentifier.
Poste condition	Compte créer avec succès.
Scénario nominal	1-L'administrateur demande de créer un compte. 2-Le système lui affiche le formulaire de création. 3- L'administrateur saisie les informations nécessaire (nom, prénom, mot de passe, etc.). 4- L'administrateur valide son formulaire. 5- Le système affiche la réussite de création.
Scénario alternative	1-Le visiteur n'a pas remplie certains champs obligatoires. Le système détecte et lui propose de les remplir à nouveau.
Scénario d'exception	

Tableau II-16: Cas d'utilisation créer un compte

- **Diagramme de séquence**

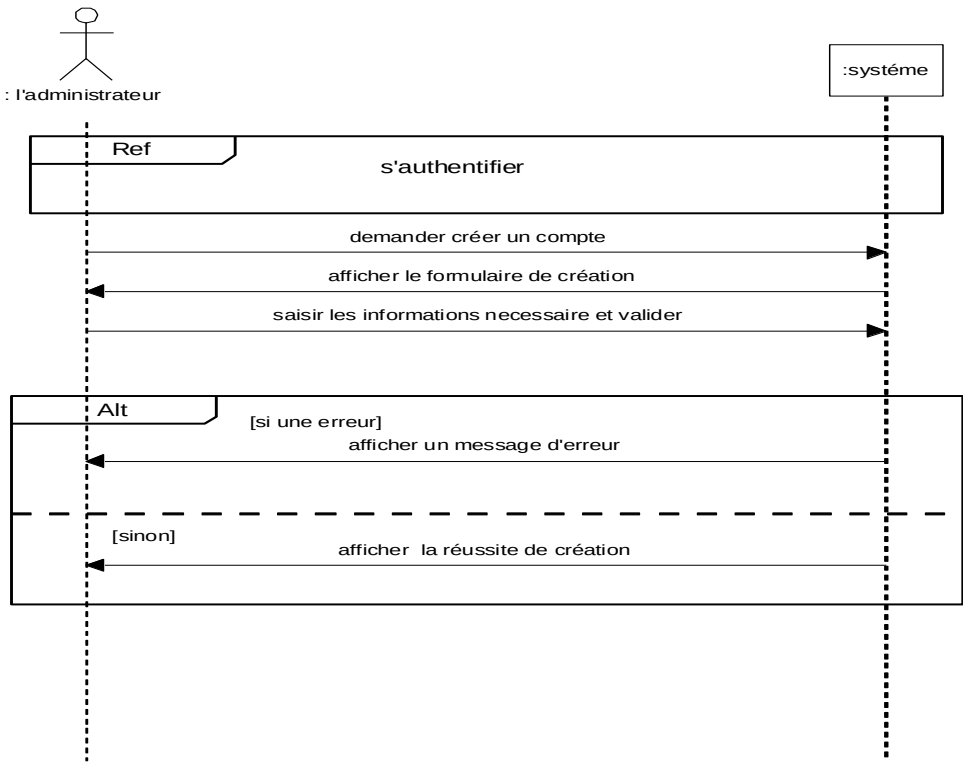

Figure II-47 : Diagramme de séquence système créer un compte

- **Diagramme d'activité**

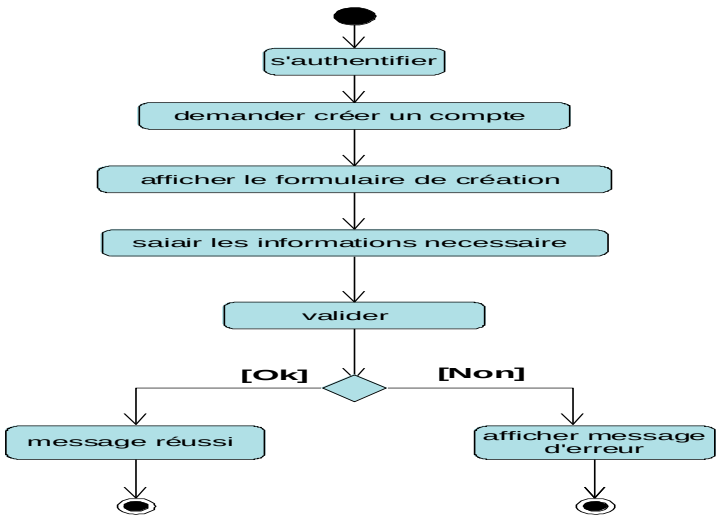

Figure II-48: Diagramme d'activité créer un compte

Activer/Désactiver un compte

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Activer/Désactiver compte.**
Acteur principal	Administrateur.
Objectif	Permet d'activer ou bien désactiver les comptes.
Pré condition	Administrateur s'authentifier.
Poste condition	Compte activer/désactiver avec succès.
Scénario nominal	1- L'administrateur demande d'activer/désactiver un compte. 2- Le système affiche la liste des comptes. 3-L'administrateur sélectionne le compte et choisir d'activer/désactiver le compte et valide. 4- Le système affiche la réussite d'activer/désactiver le compte.
Scénario alternative	-Un message d'erreur est afficher sur l'écran avise l'administrateur que la liste des comptes est vide.
Scénario d'exception	

Tableau II-17: Cas d'utilisation activer/désactiver un compte

- **Diagramme de séquence**

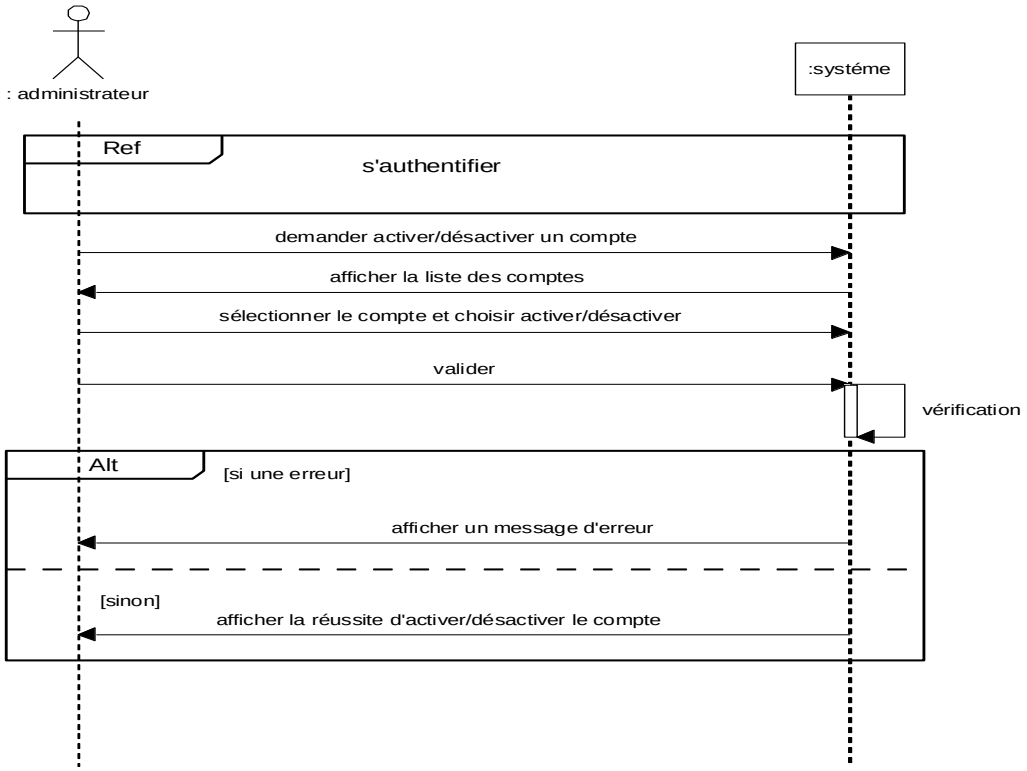

Figure II-49: Diagramme de séquence système activé/désactivé un compte

- **Diagramme d'activité**

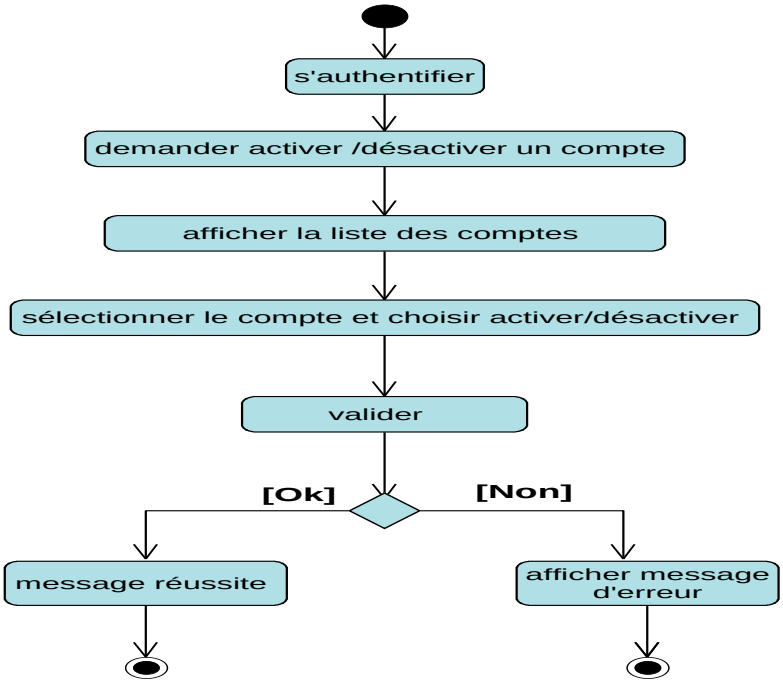

Figure II-50: Diagramme d'activité activé/désactivé un compte

Modifier un compte

- **Description textuelle du cas d'utilisation**

Cas d'utilisation	**Modifier un compte.**
Acteur principal	Administrateur.
Objectif	Permet de modifier un compte existe déjà.
Pré condition	Administrateur s'authentifier.
Poste condition	Modifier compte avec succès.
Scénario nominal	1-L'administrateur demande de modifier un compte. 2-Le système lui affiche les informations du compte. 3- L'administrateur modifie les informations qu'il veut et valide. 4- L'administrateur valide son formulaire. 5- Le système vérifie la validité des informations et affiche message de réussite.
Scénario alternative	1-Un message d'erreur est affiché avise le client que les informations saisies non valides (champ vide, information erronée, etc.).
Scénario d'exception	

Tableau II-18: Cas d'utilisation modifier un compte

➢ **Diagramme de séquence**

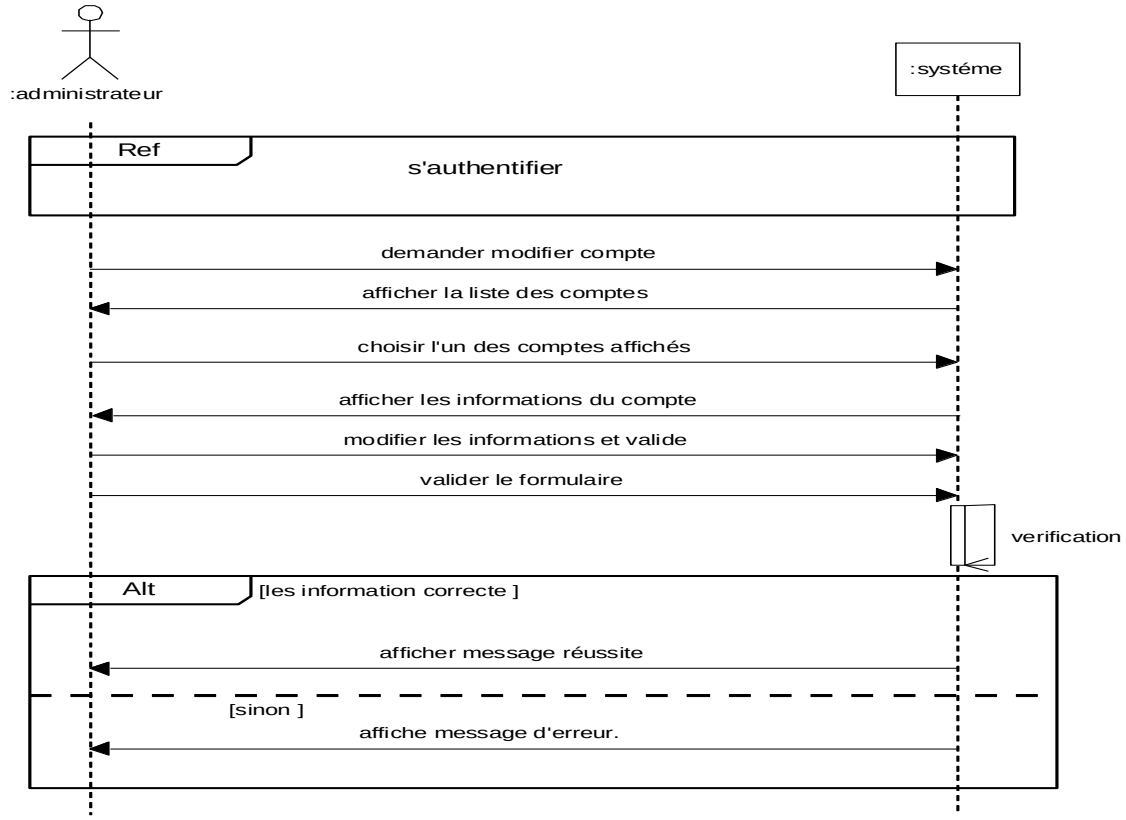

Figure II-51: Diagramme de séquence système modifier un compte

➢ **Diagramme d'activité**

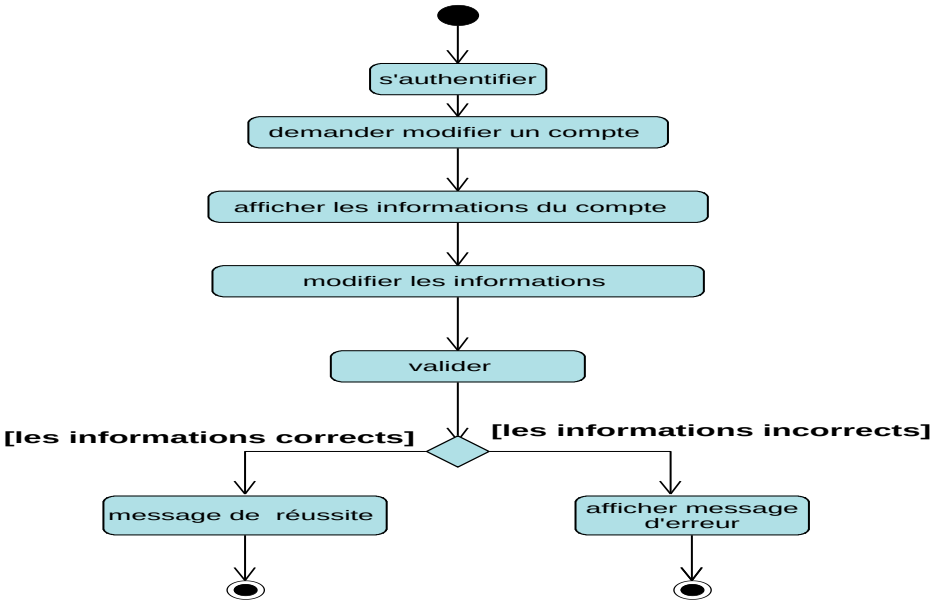

Figure II-52: Diagramme d'activité modifier un compte

3.3.3. Organisation en couche du modèle de spécification

Le développement de notre application repose sur un découpage en couches ou tiers, nous parlons alors d'applications multi-tiers. Trois grandes couches sont représentées :

– **Couche présentation :** C'est la face visible de notre application.

– **Couche métier :** C'est l'ensemble des règles métier de l'application.

– **Couche persistance de données :** Elle permet la manipulation et la conservation des données. [6]

4. Conclusion

La capture des besoins fonctionnels joue un rôle essentiel dans ce cadre d'une part, de compléter les recueils initiales des besoins opérés pendant l'étude préliminaire, et de l'autre part, elle donne une première vue pour le prochain chapitre concernant l'analyse afin d'identifier les classes du modèle statique qui présentent une des approches orientées objets.

La capture des besoins techniques couvre, par complémentarité avec celle des besoins fonctionnels, toutes les contraintes qui ne traitent ni de la description de métiers des utilisateurs, ni de la description.

CHAPITRE III

ANALYSE

1. Introduction

L'étape d'analyse est entamée à la suite de l'étape de capture des besoins fonctionnels, elle est située sur la branche gauche du processus 2TUP.

Dans ce chapitre là on va traiter la phase de l'analyse objet du système, en suivant les étapes suivantes :

- Le découpage en catégorie
- Le développement du modèle statique,
- Le développement du modèle dynamique.

2. Découpage en catégorie

Cette phase utilise la notion de package pour définir des catégories de classes d'analyse et découper le modèle UML en blocs logiques les plus indépendants possibles.

Une catégorie consiste en un regroupement logique de classes à forte cohérence interne et faible couplage externe. **[6]**

Le découpage en catégories se fait en 3 étapes :

2.1.La répartition des classes candidates en catégories

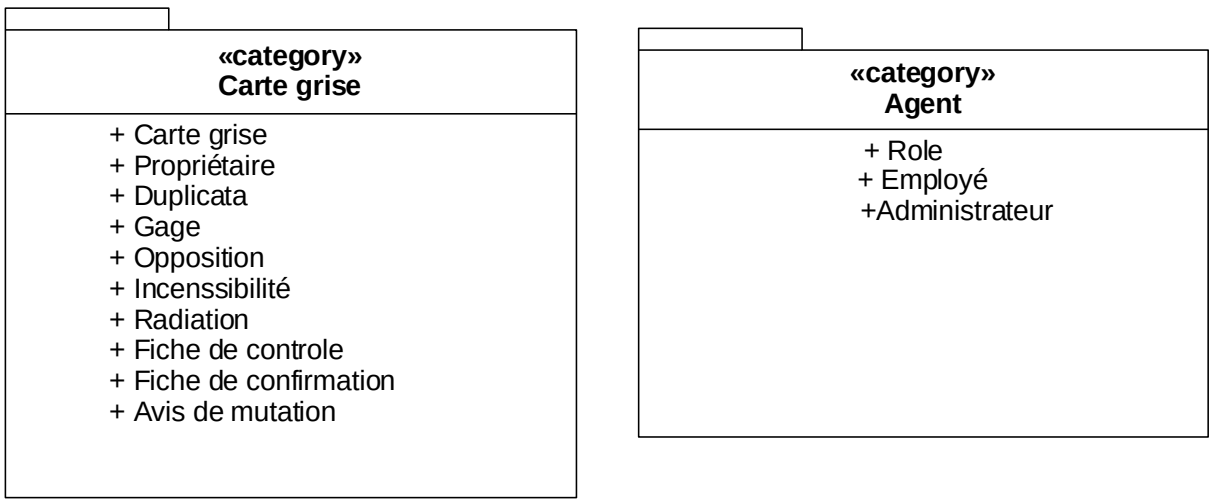

Figure III-1 : Découpage en catégorie de notre système

2.2. Elaboration des diagrammes de classes préliminaires par catégorie

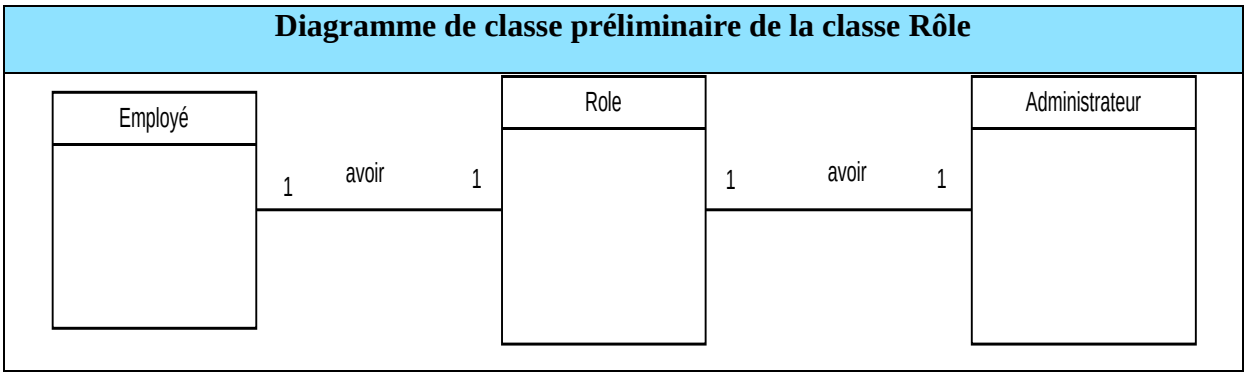

Figure III-2: Diagrammes de classes préliminaires de la classe rôle

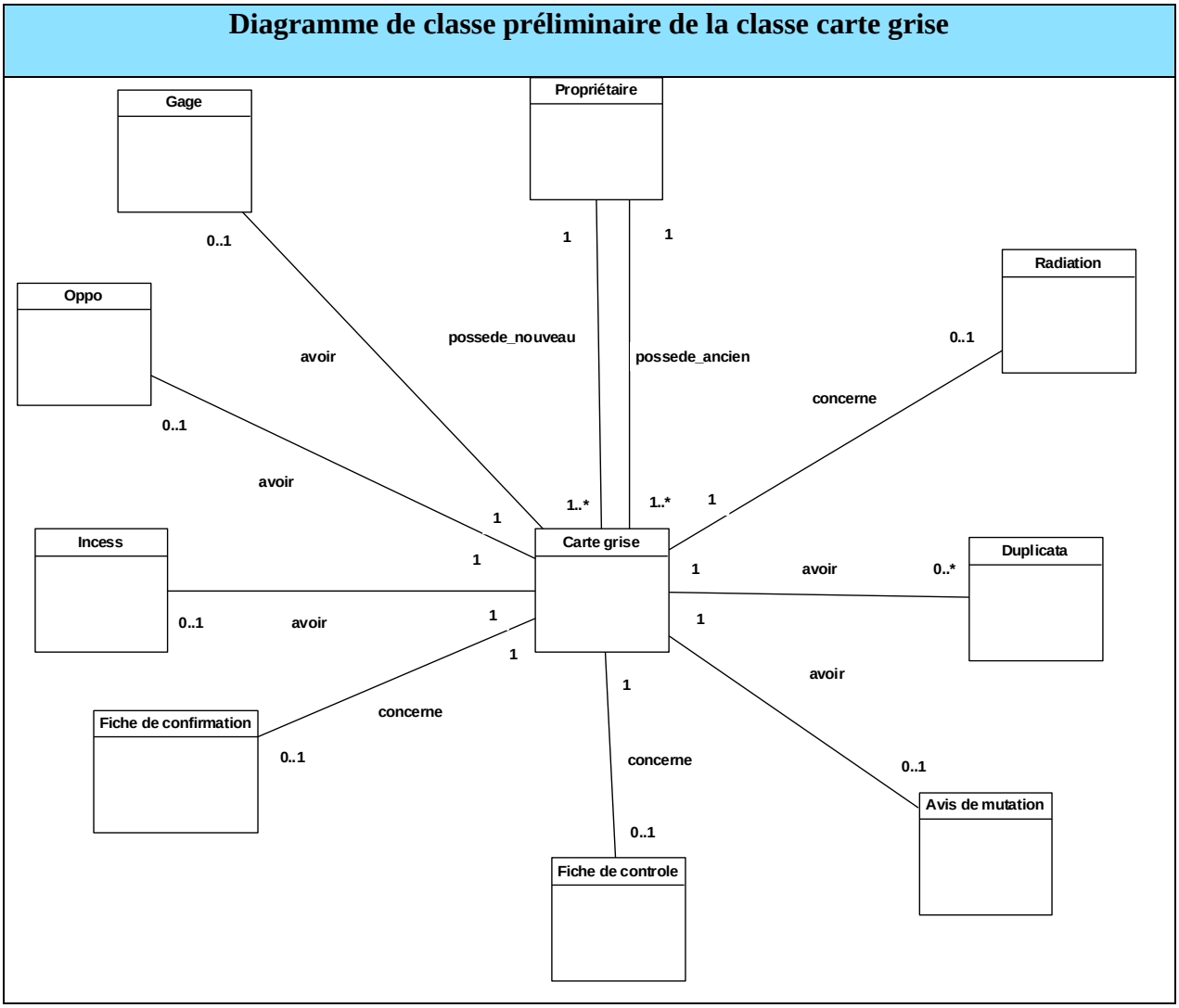

Figure III-3 : Diagrammes de classes préliminaires de la classe carte grise

2.3. Dépendance entre catégories

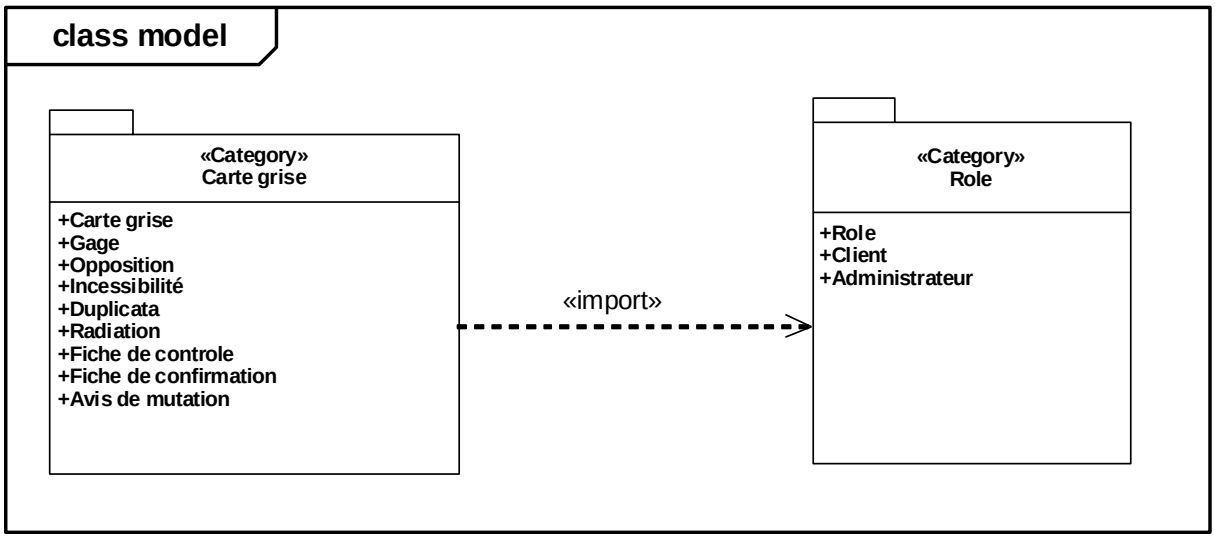

Figure III-4: Modèle structurel d'analyse

3. Développement du modèle statique

Le développement du modèle statique représente la deuxième activité de l'étape d'analyse. Lors de cette étape, nous reprenons les diagrammes organisés lors du découpage en catégories afin de les affiner en leur ajoutant des attributs.

Voici le diagramme de classe de la catégorie carte grise :

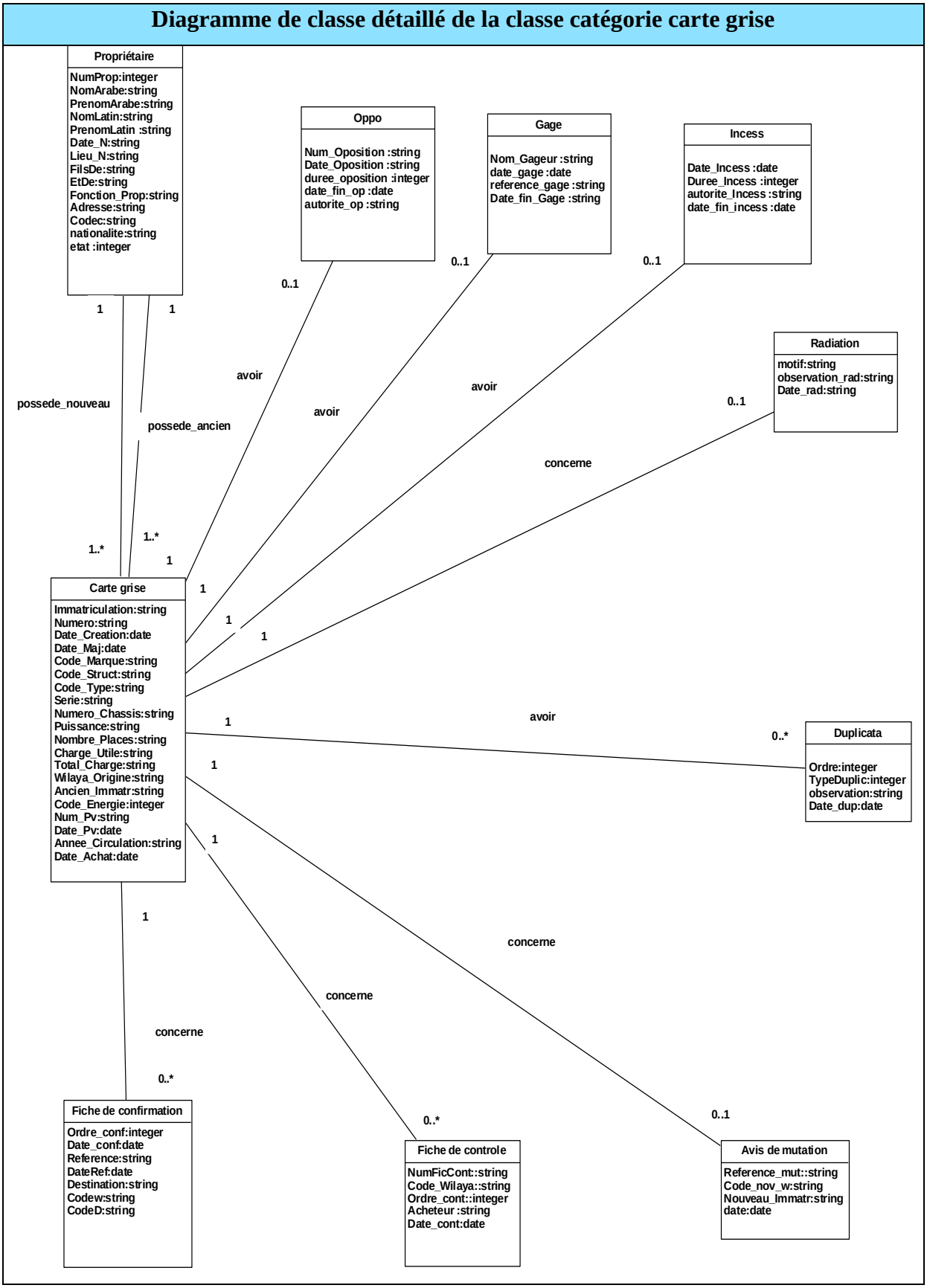

Figure III-5: Diagramme de classe détaillé de la classe catégorie carte grise

Voici le diagramme de classe de la catégorie Rôle :

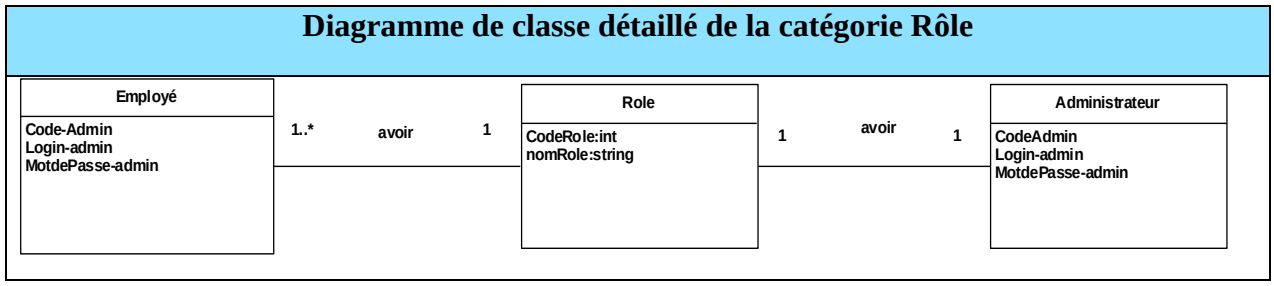

Figure III-6: Diagramme de classe détaillé de la catégorie rôle

4. Développement du modèle dynamique

Le développement du modèle dynamique est la troisième activité de l'étape d'analyse. Cette activité est en relation avec l'activité de modélisation statique.

Lors de cette étape, nous décrivons les différentes interactions entre les objets de notre application. En effet, nous avons utilisés le modèle dynamique : le diagramme de séquence détaillé.

4.1. Diagrammes de séquences

Le diagramme de séquence est un diagramme d'interaction entre les objets, qui met l'accent sur le classement des messages par ordre chronologique durant l'exécution du système. Un diagramme de séquence est un tableau dans lequel les objets sont rangés sur l'axe des abscisses et des messages par ordre d'apparition sur l'axe des ordonnées. Il est utilisé pour représenter certains aspects dynamiques d'un système : dans le contexte d'une opération, d'un système, d'un sous-système, d'un cas d'utilisation (un scénario d'un cas d'utilisation) selon un point de vue temporel. En effet dans cette phase, et après identification des cas d'utilisation, nous représentons à l'aide des diagrammes de séquences les différents cas d'utilisation.

Cat d'utilisation ajouter nouvelle carte grise

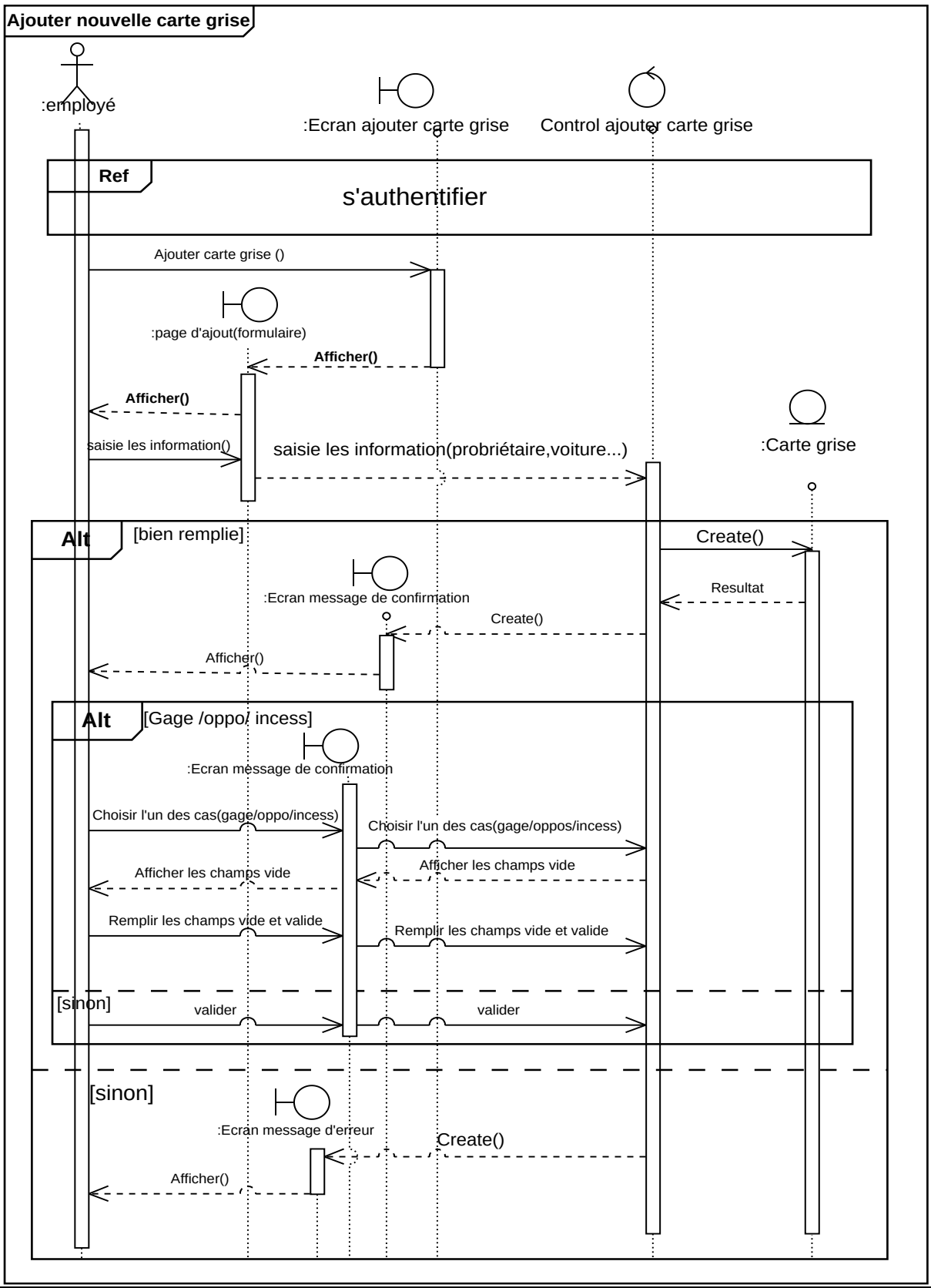

Figure III-7: Diagramme de séquence de cas d'utilisation ajouter nouvelle carte grise

Cas d'utilisation : Modifier carte grise

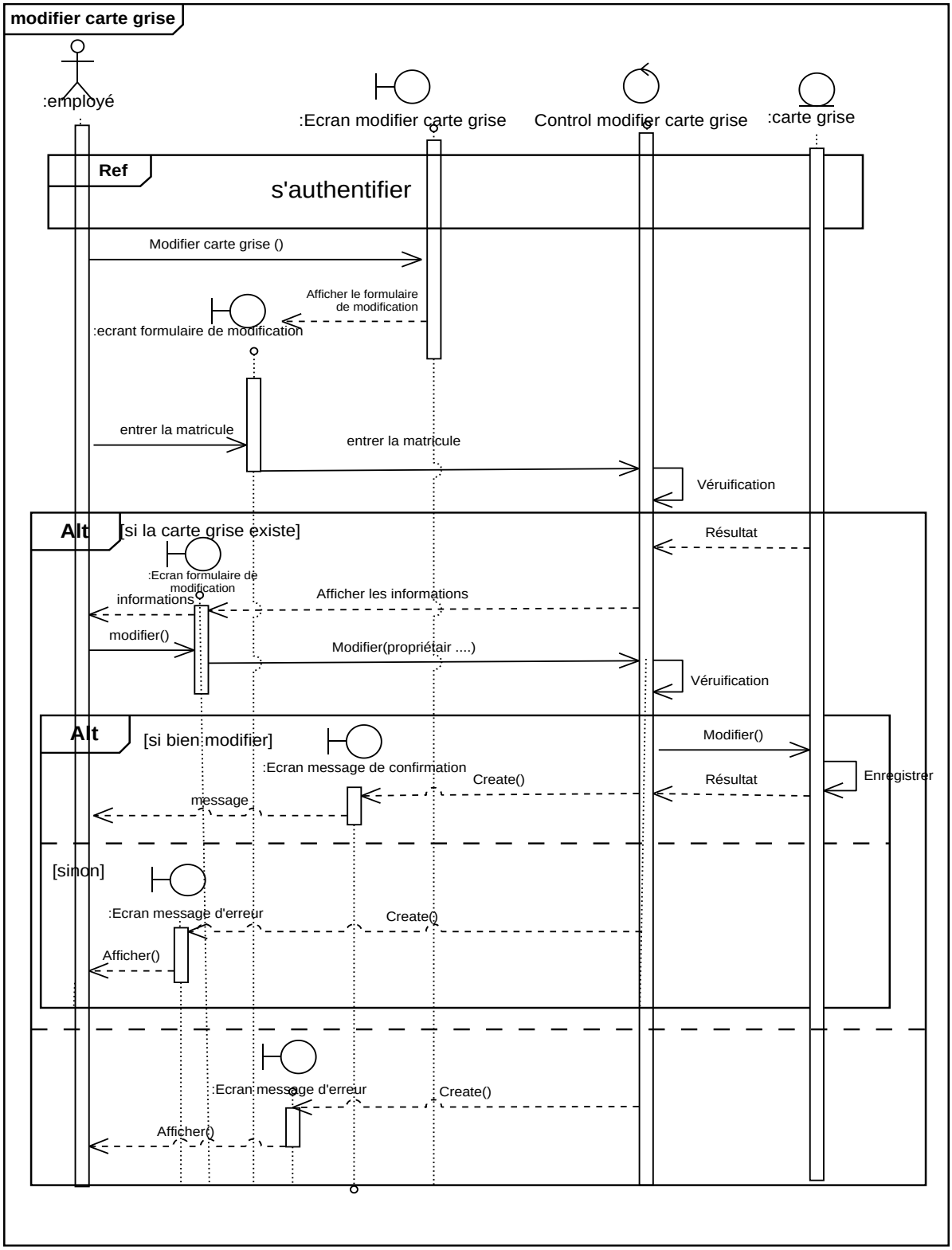

Figure III-8: Diagramme de séquence de cas d'utilisation modifier carte grise

Cas d'utilisation : Ajouter duplicata

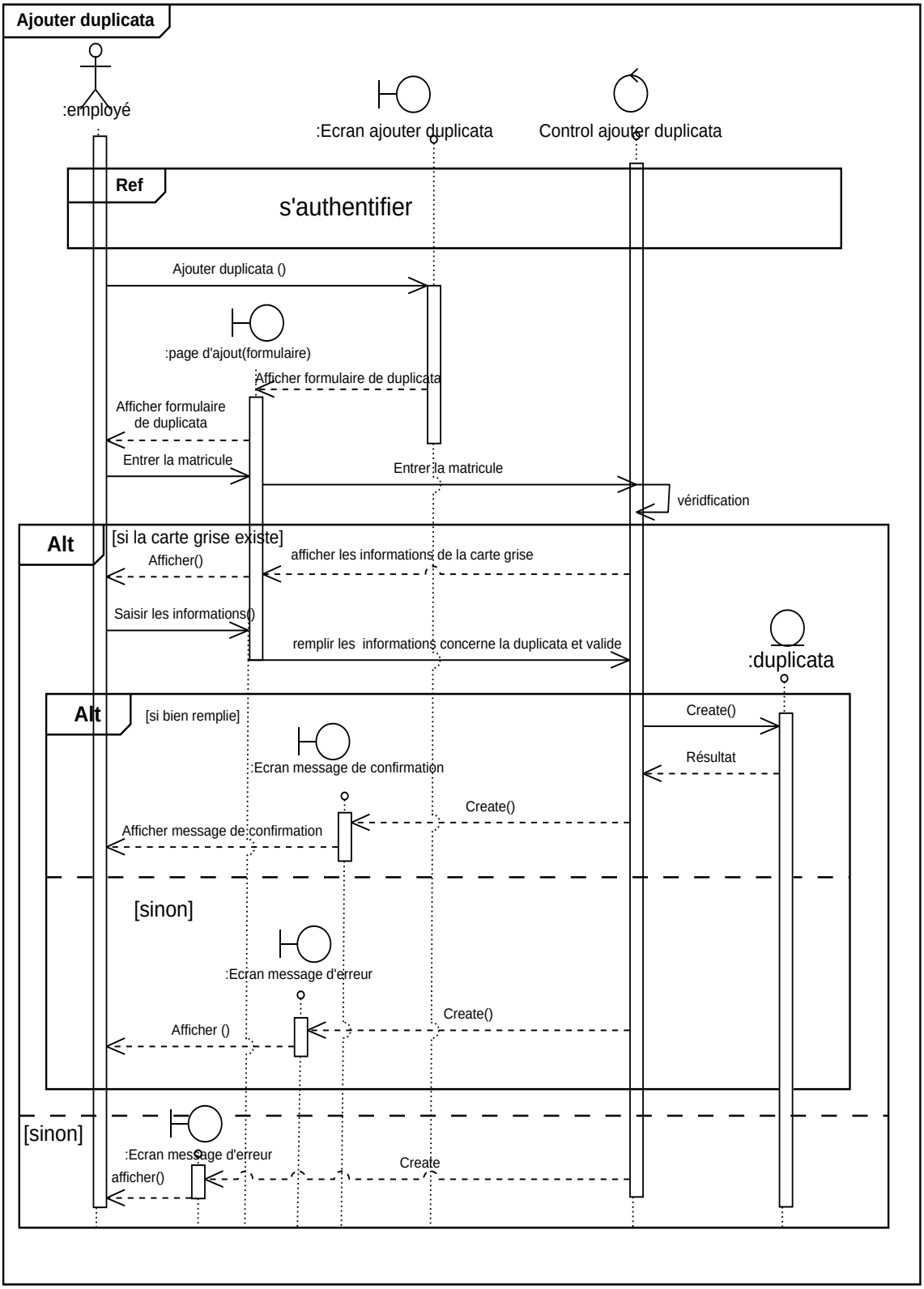

Figure III-9: Diagramme de séquence de cas d'utilisation ajouter duplicata

Cas d'utilisation : Annuler Gage/Opposition/incessibilité

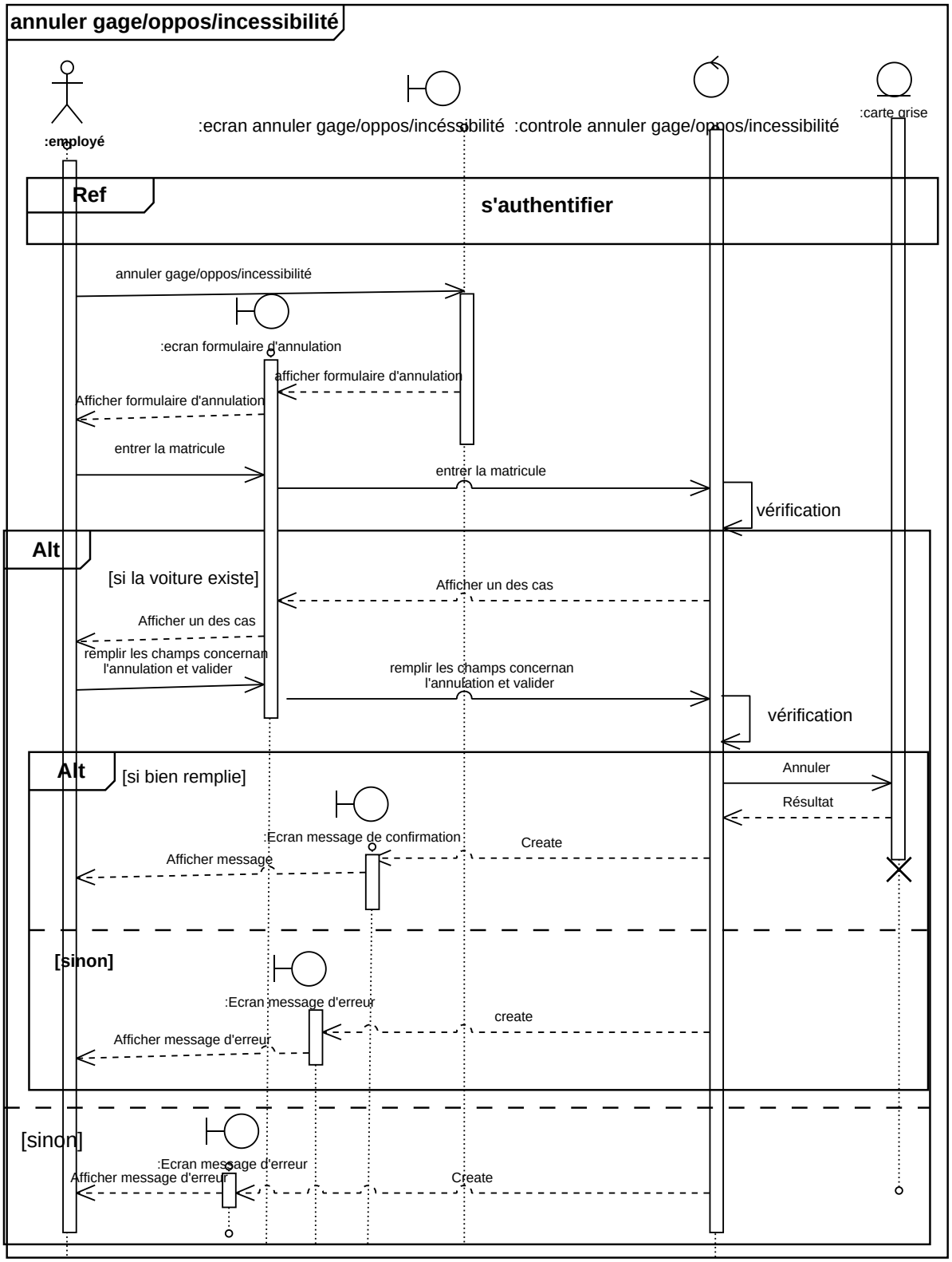

Figure III-10: Diagramme de séquence de cas d'utilisation annuler gage/opposition/incessibilité

Cas d'utilisation : Ajouter vente locale

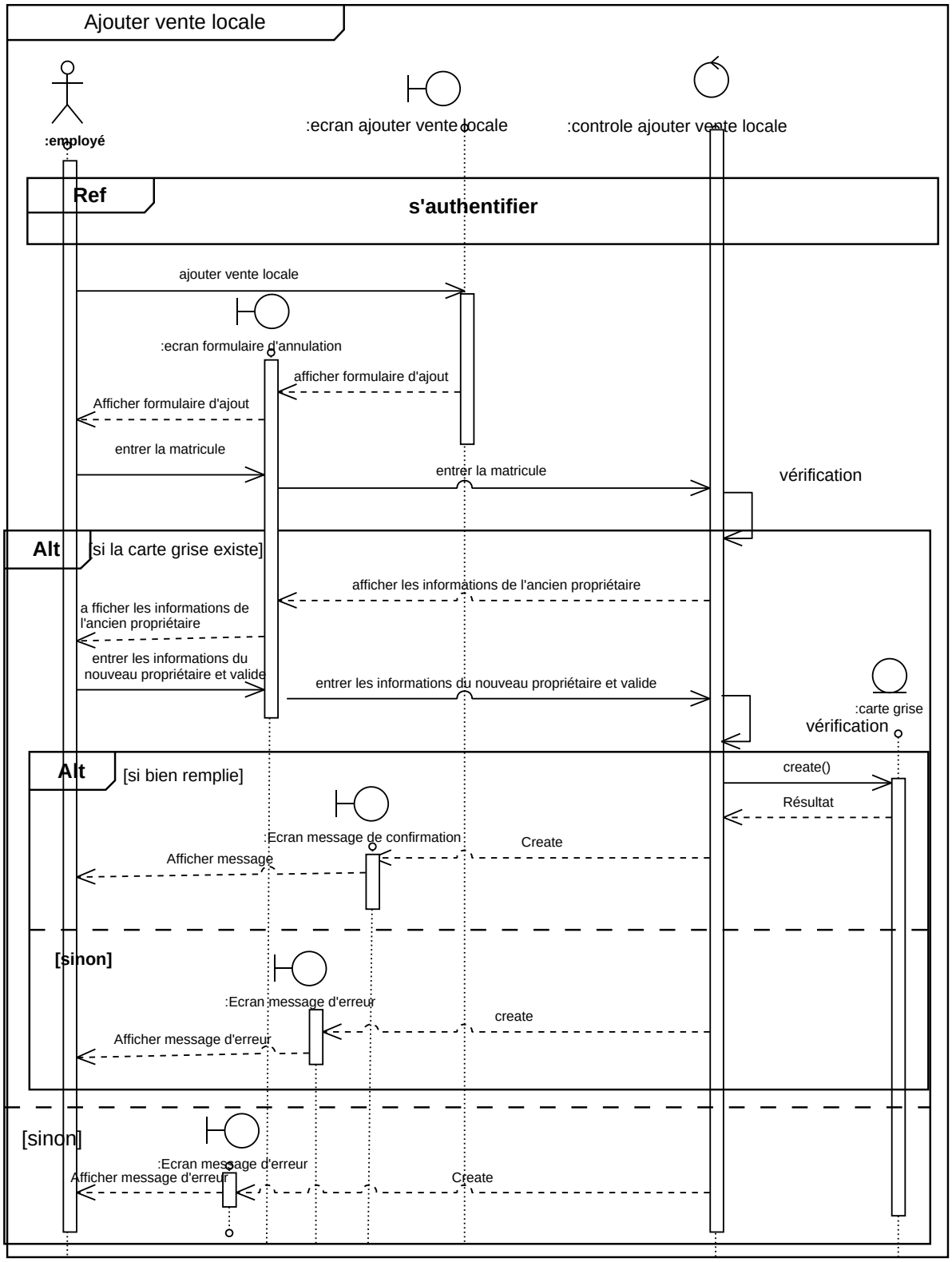

Figure III-11: Diagramme de séquence de cas d'utilisation ajouter vente locale

Cas d'utilisation : Annuler vente locale

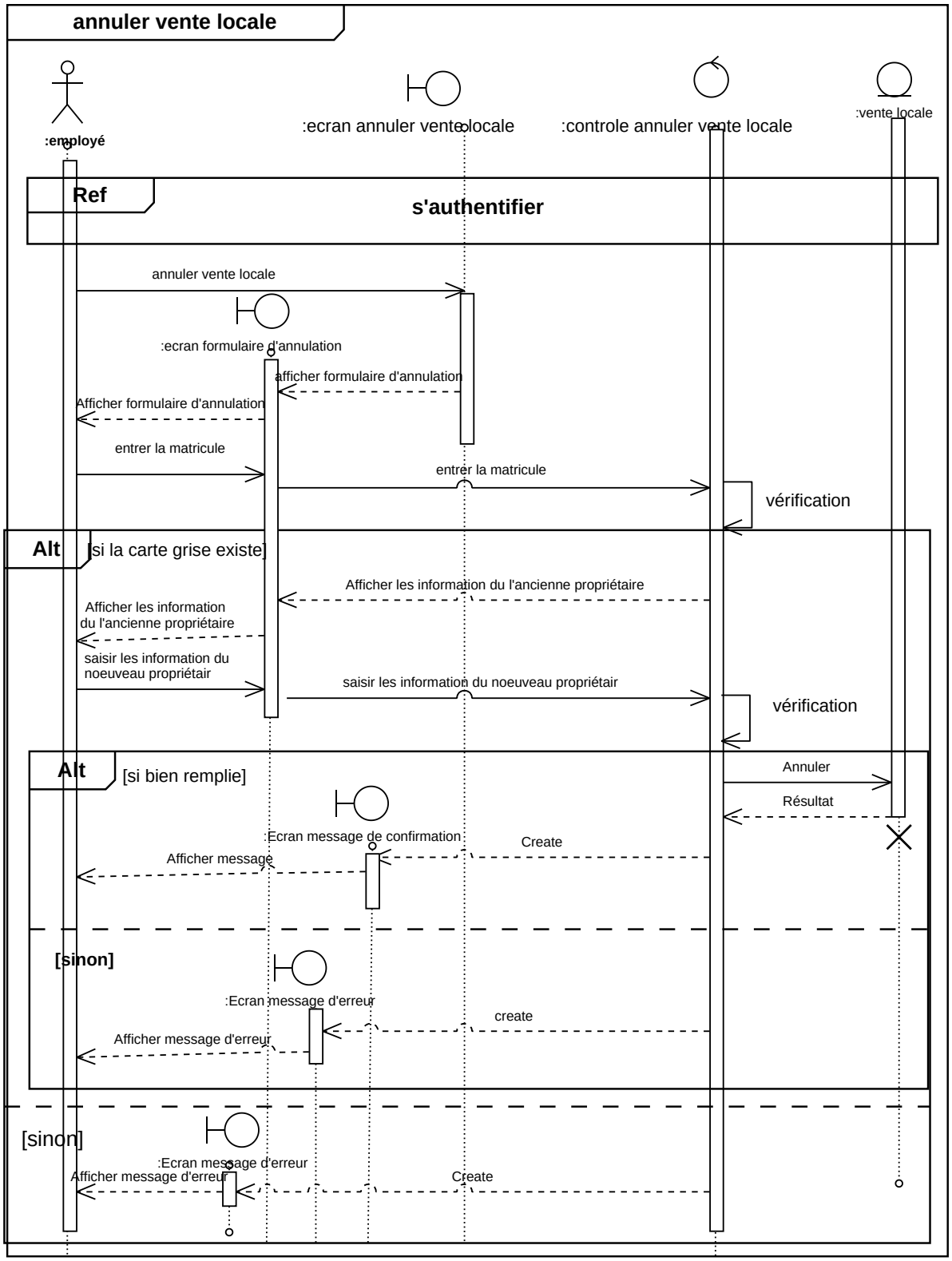

Figure III-12: Diagramme de séquence de cas d'utilisation annuler vente locale

Cas d'utilisation : Ajouter fiche de contrôle

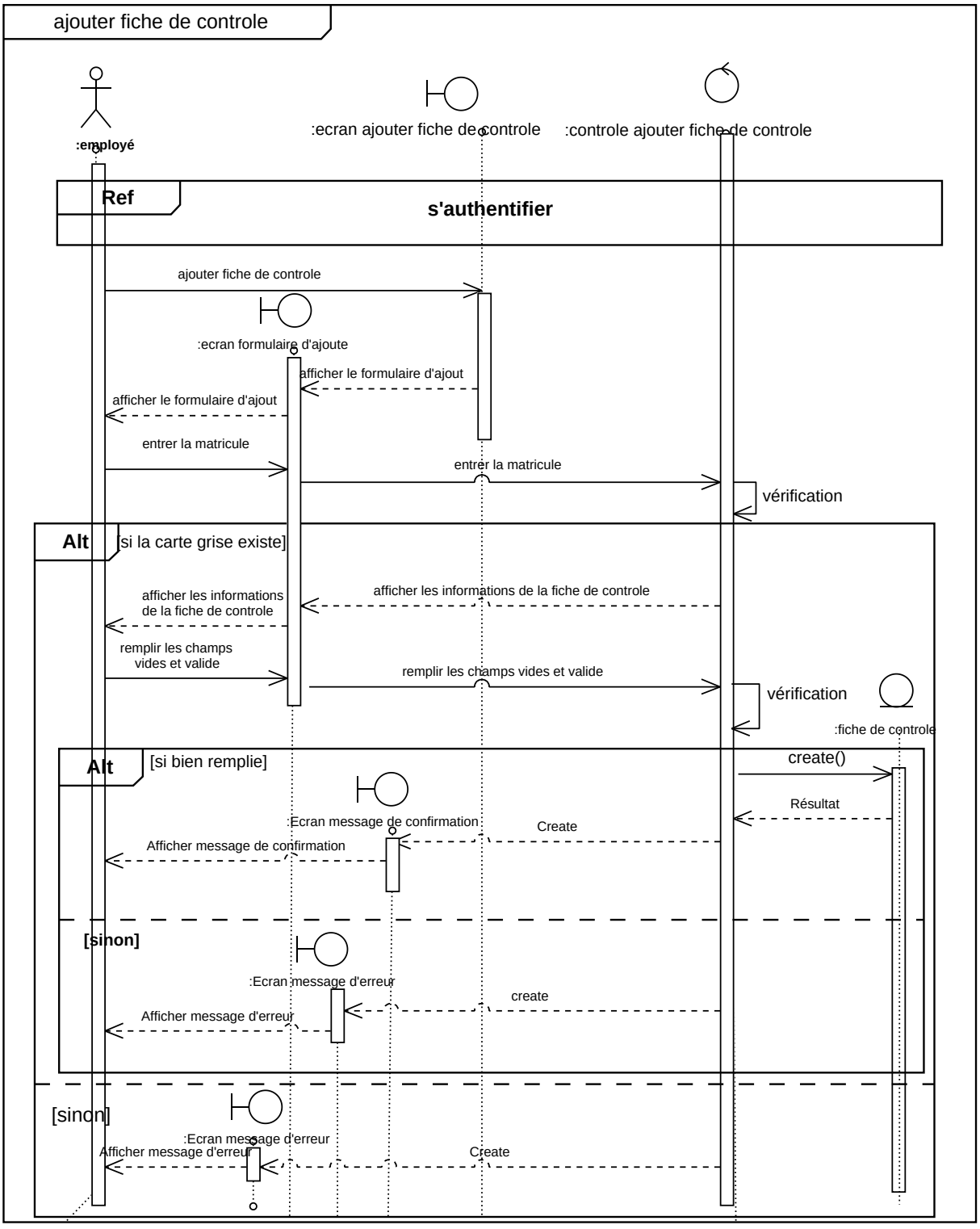

Figure III-13: Diagramme de séquence de cas d'utilisation ajouter fiche de contrôle

Cas d'utilisation : Modifier fiche de contrôle

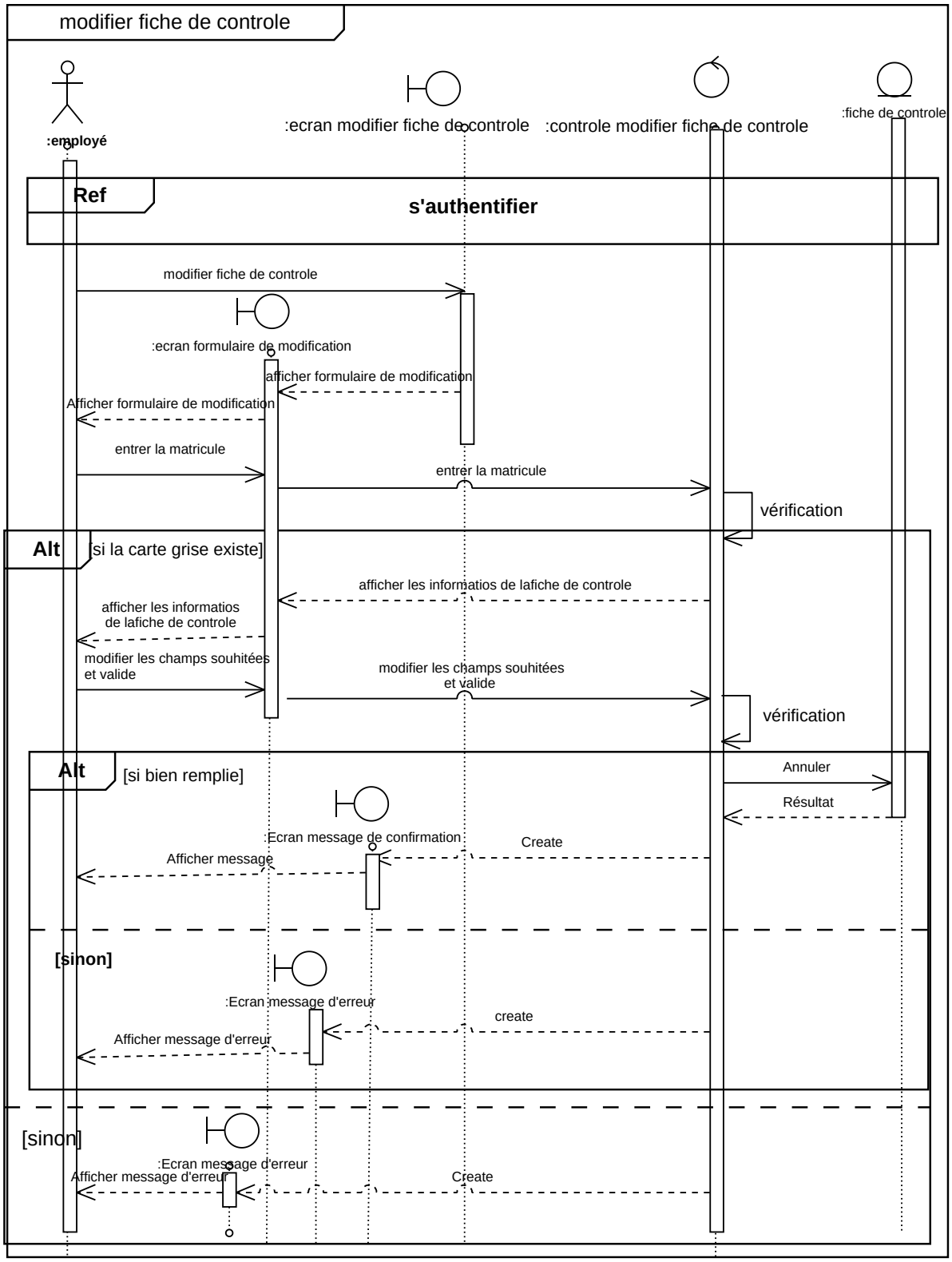

Figure III-14: Diagramme de séquence de cas d'utilisation modifier fiche de contrôle

Cas d'utilisation : Annuler fiche de contrôle

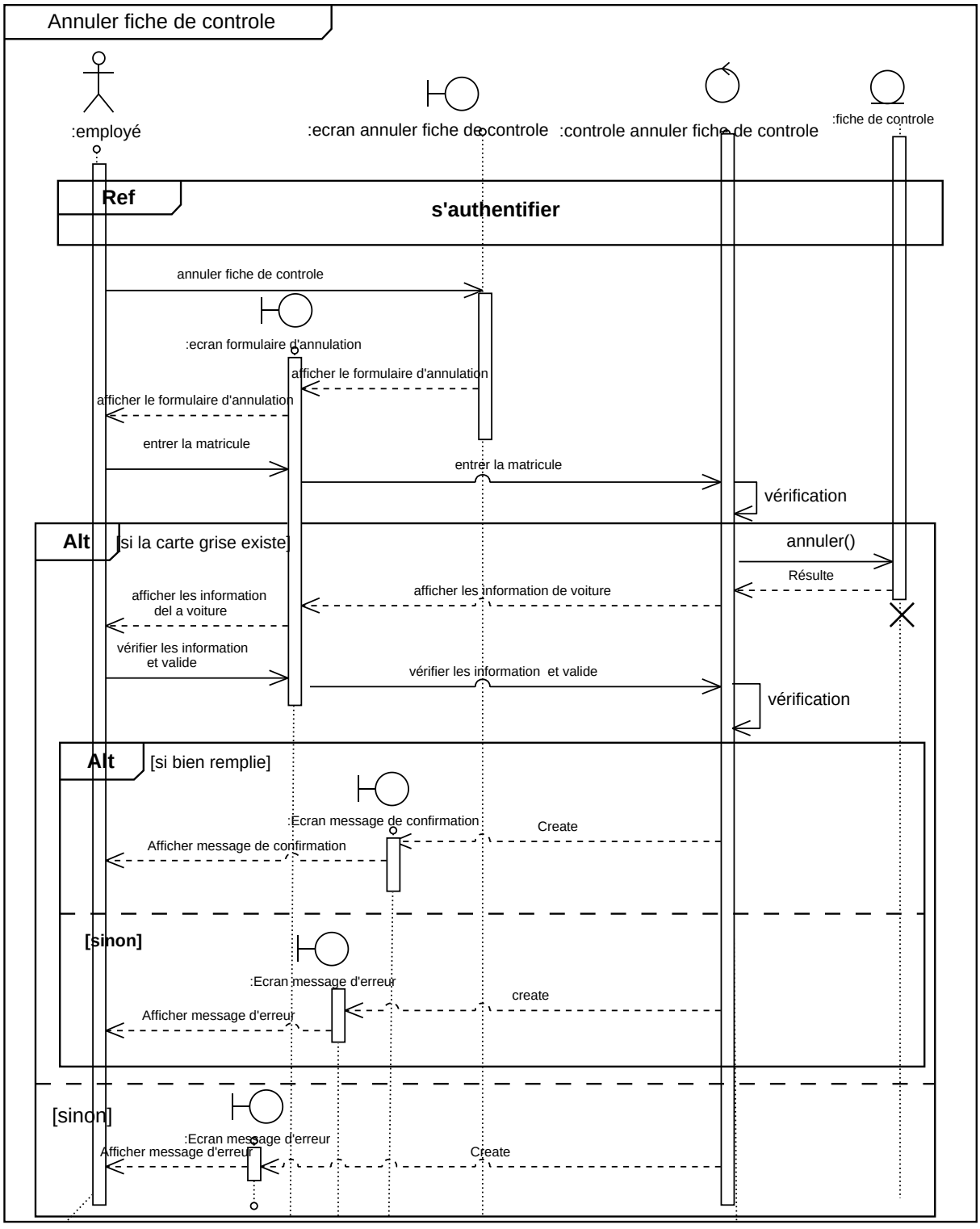

Figure III-15: Diagramme de séquence de cas d'utilisation annuler fiche de contrôle

Cas d'utilisation : Etablir avis de mutation

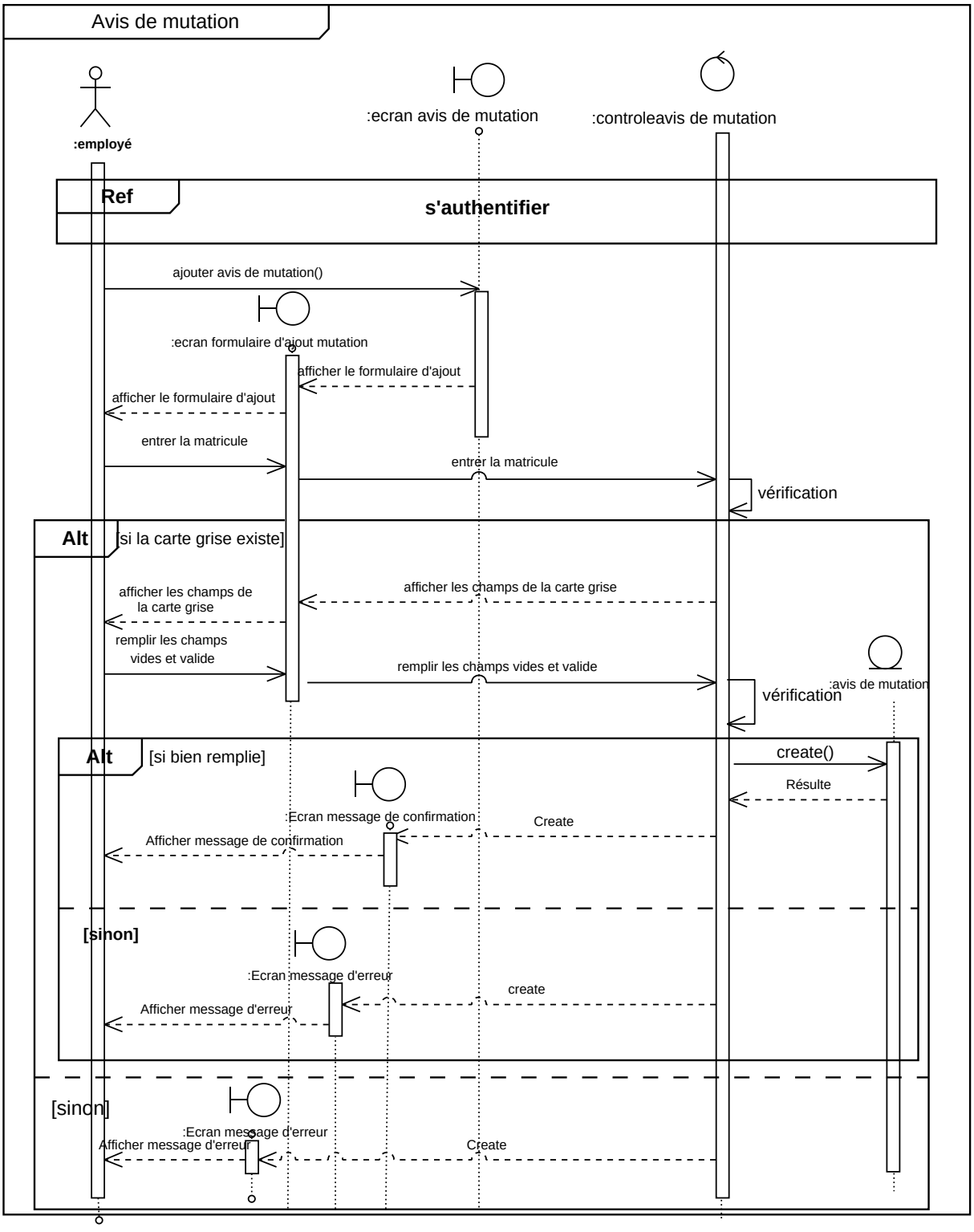

Figure III-16: Diagramme de séquence de cas d'utilisation établir avis de mutation

Cas d'utilisation : Radier carte grise

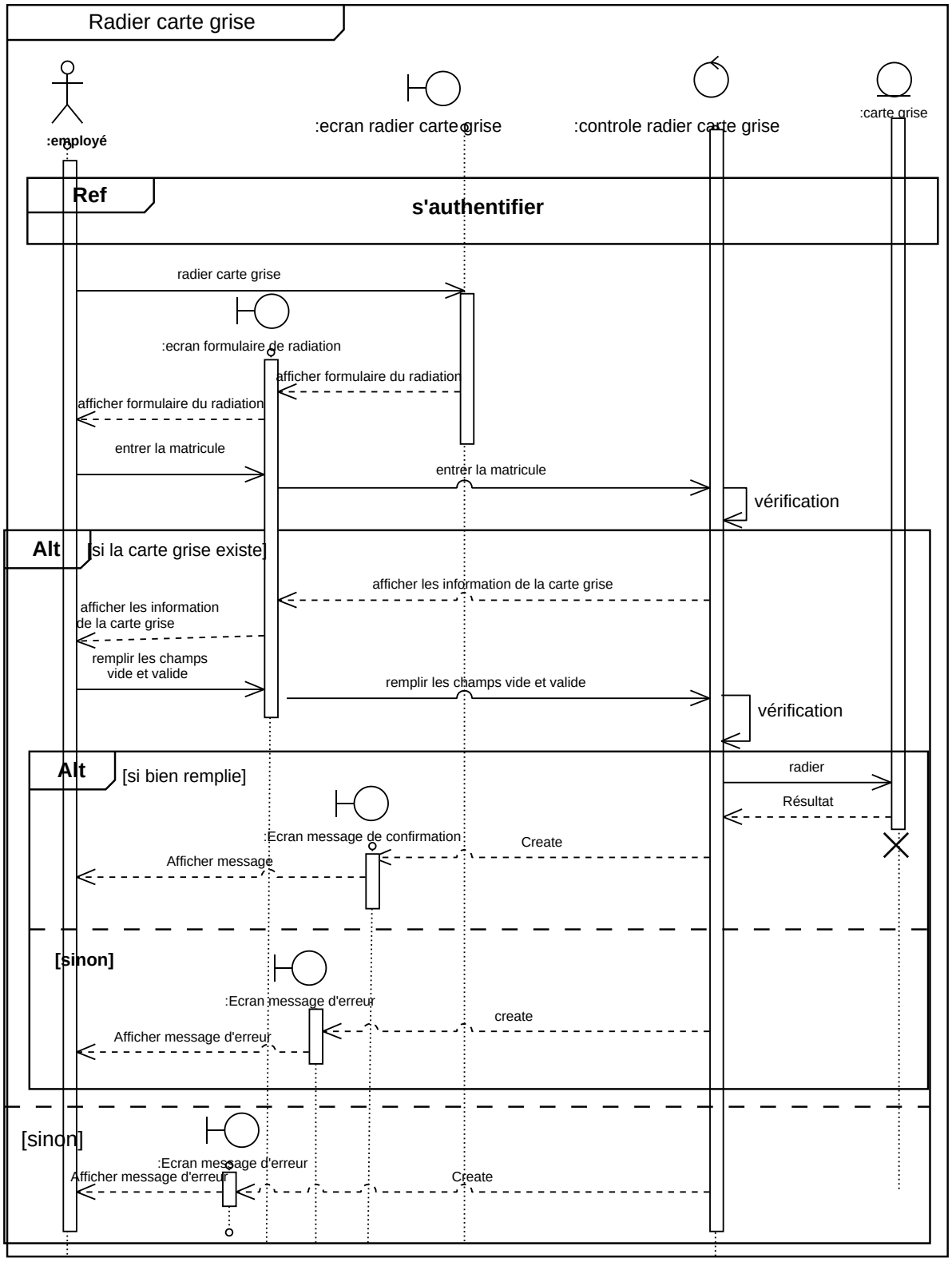

Figure III-17: Diagramme de séquence de cas d'utilisation radier carte grise

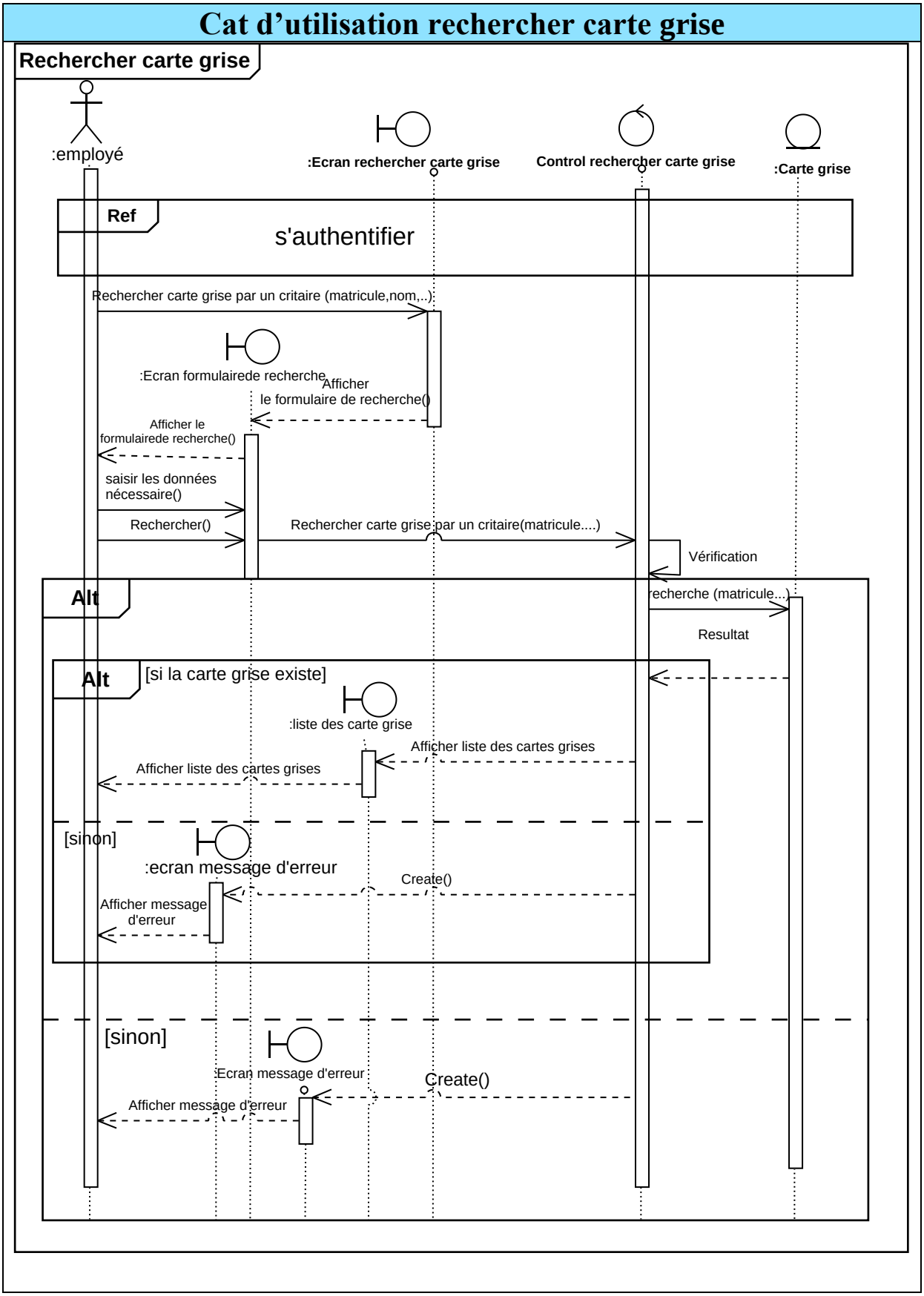

Figure III-18 : Diagramme de séquence de cas d'utilisation rechercher carte grise

Cas d'utilisation : S'authentifier

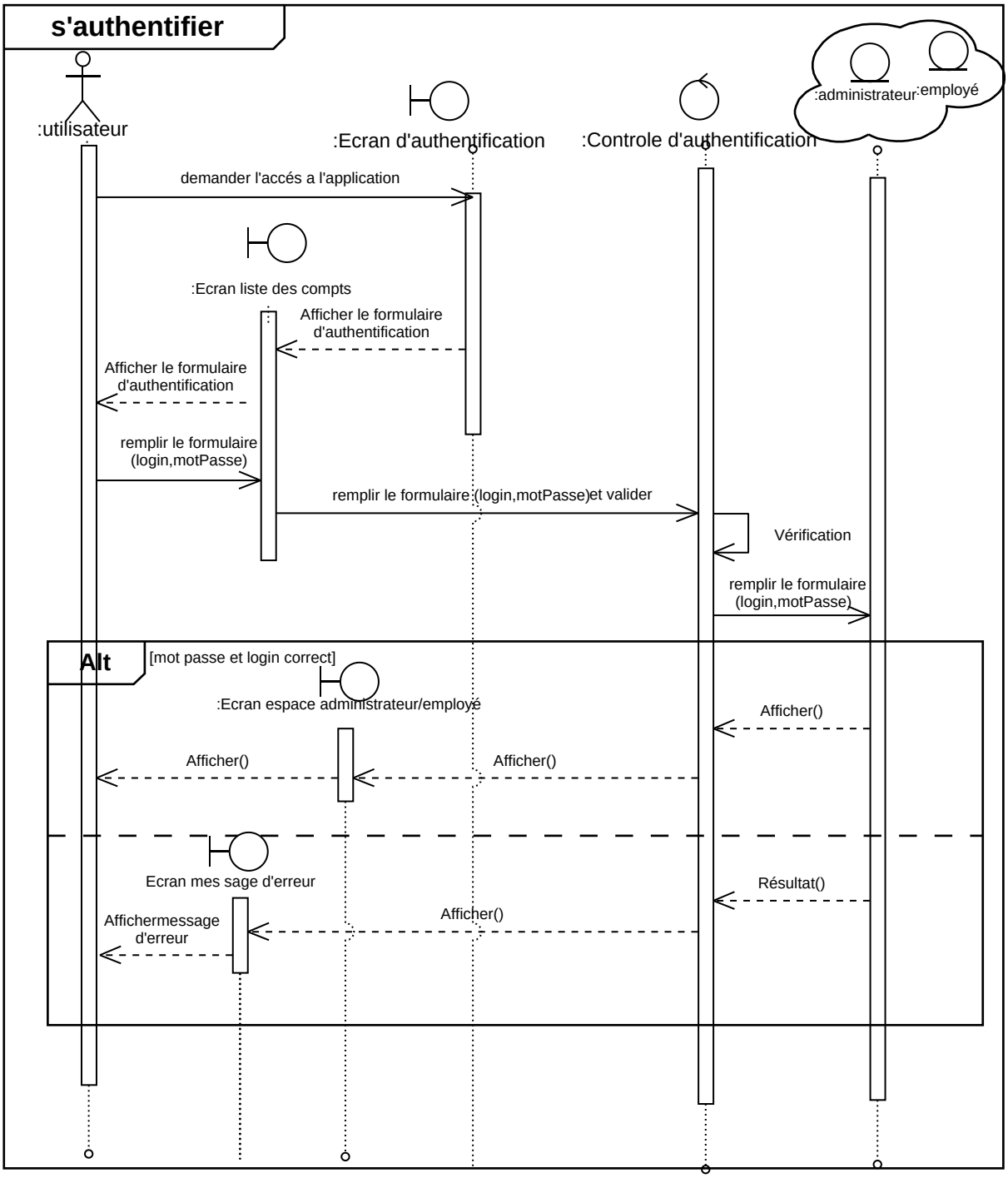

Figure III-19 : Diagramme de séquence de cas d'utilisation s'authentifier

Cas d'utilisation : Activer/Désactiver un compte

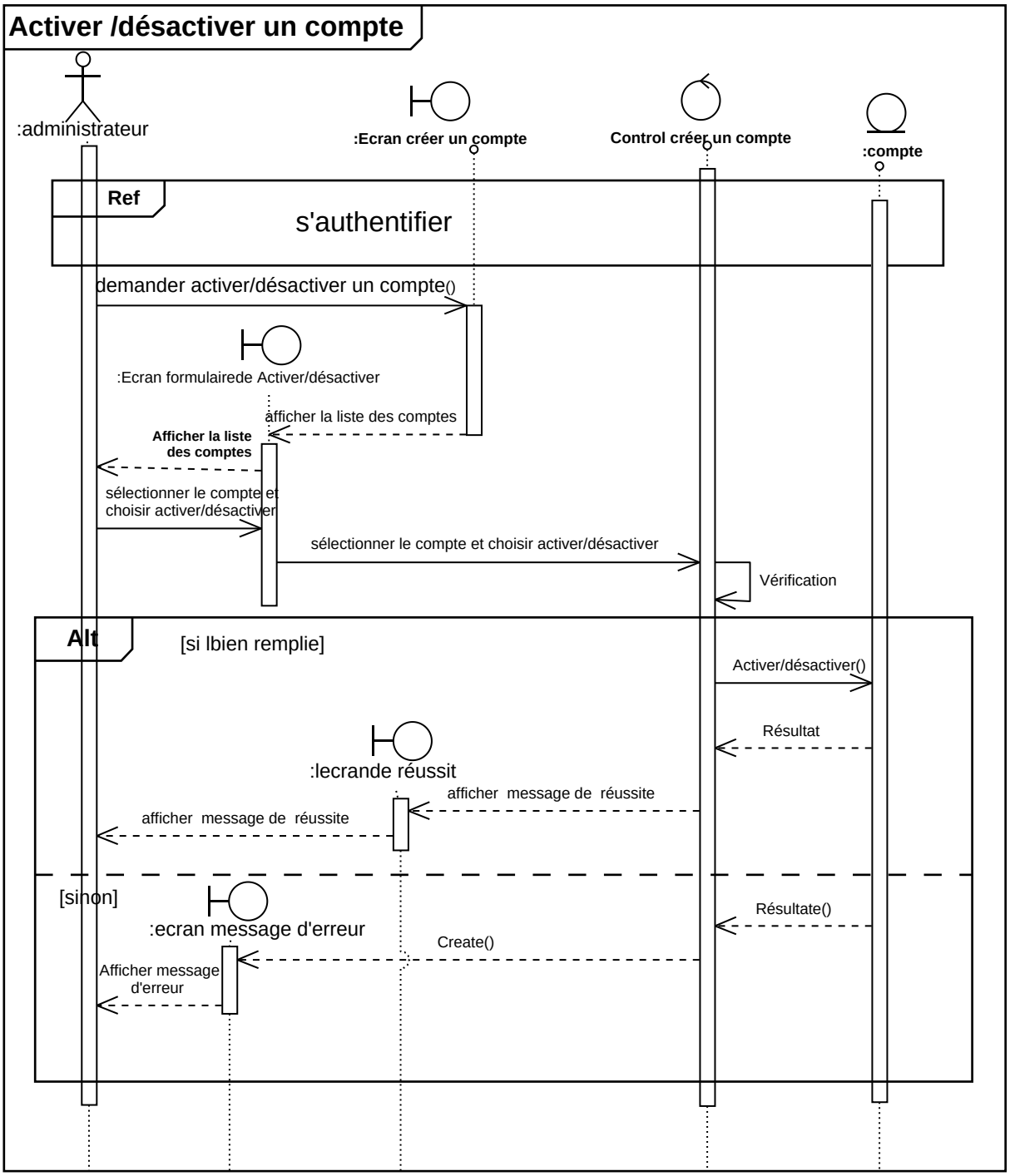

Figure III-20 : Diagramme de séquence de cas d'utilisation activer/désactiver un compte

Cas d'utilisation : Modifier un compte

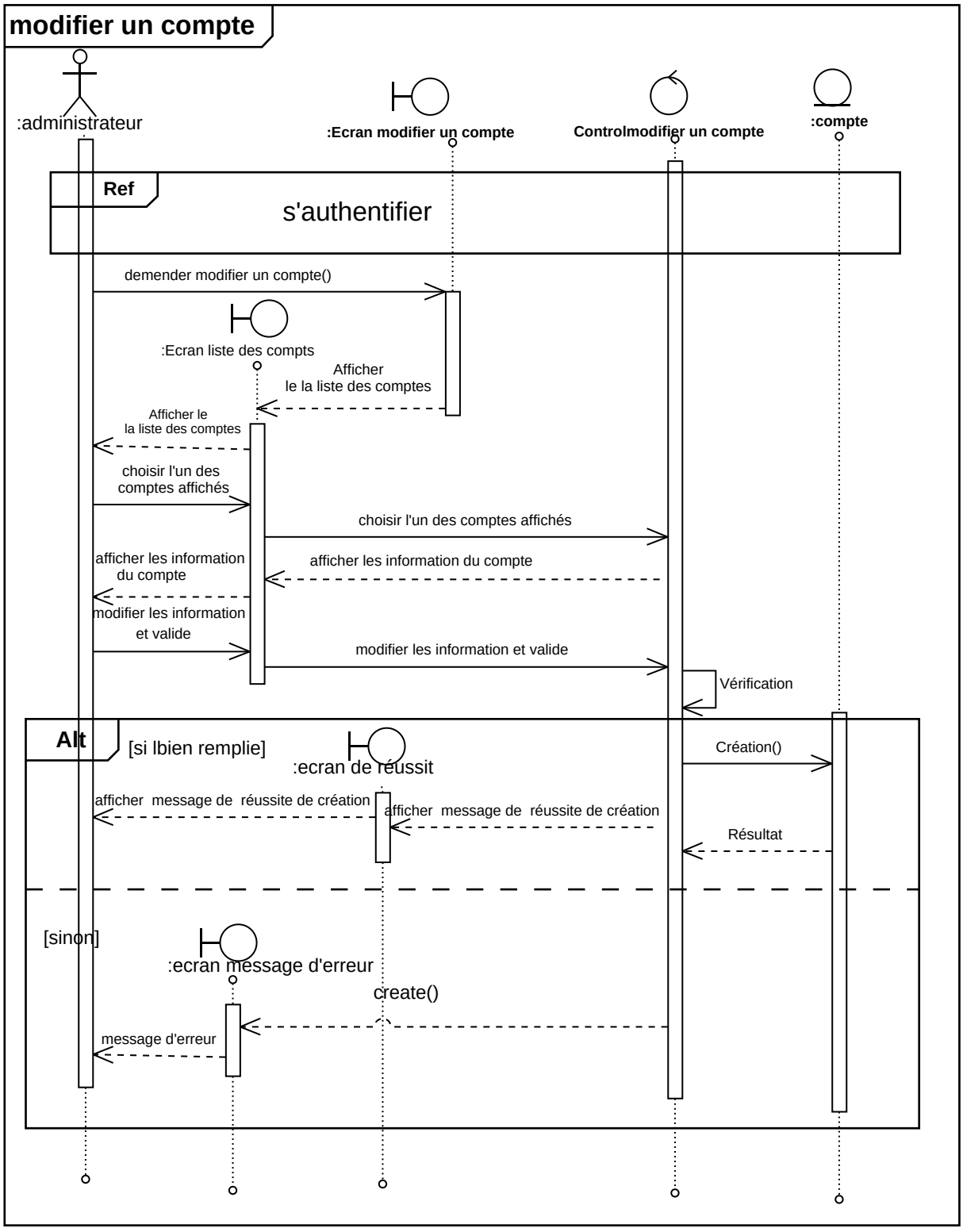

Figure III-21 : Diagramme de séquence de cas d'utilisation modifier un compte

Cas d'utilisation : Créer un compte

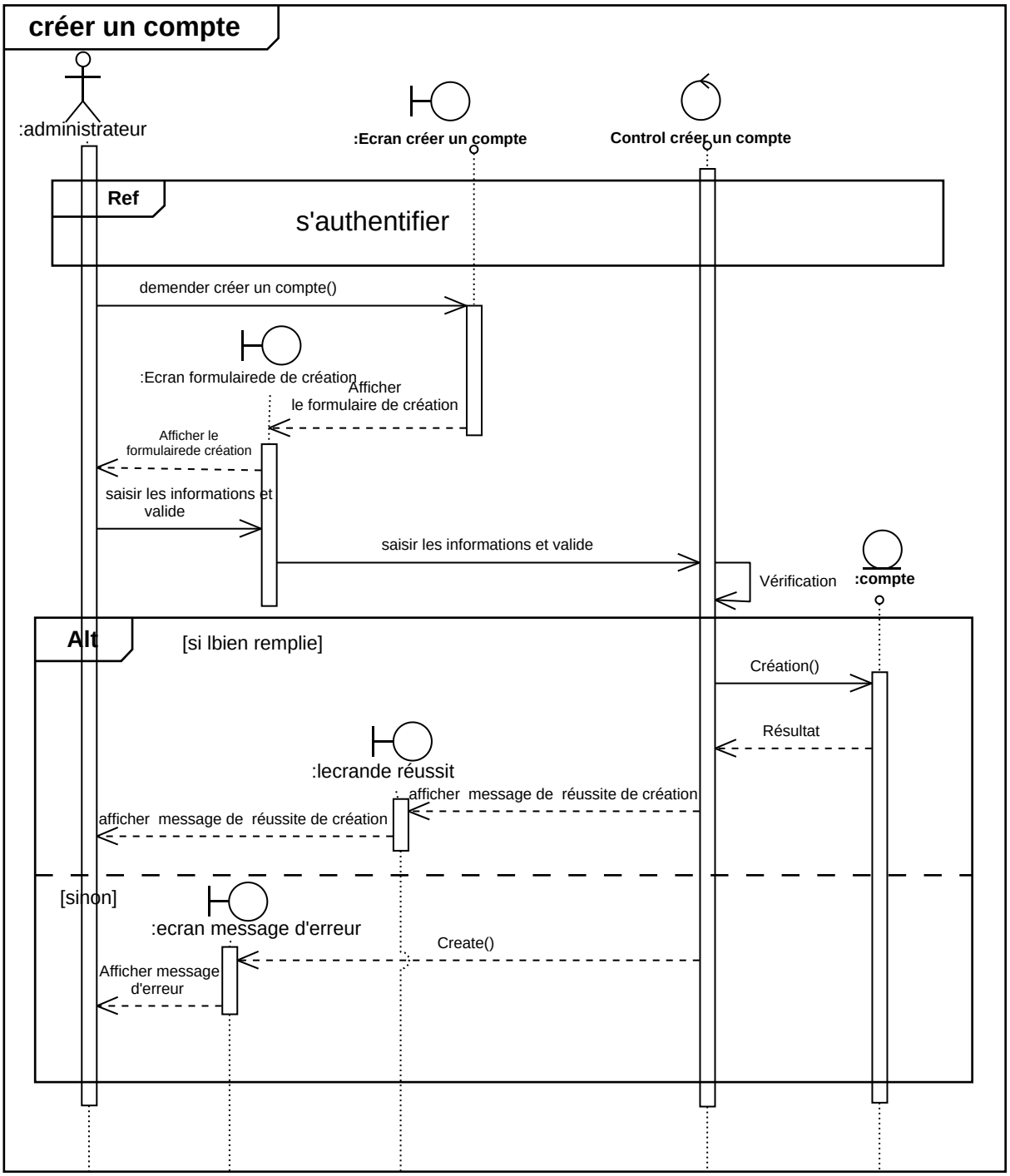

Figure III-22 : Diagramme de séquence de cas d'utilisation créer un compte

5. Conclusion

Dans ce chapitre nous avons présenté l'analyse objet du système, les classes issues des besoins fonctionnels sont regroupées en catégories pour organiser le modèle structurel d'analyse. Ce modèle nécessite un travail d'analyse détaillée de la structure des classes. Celui-ci est considéré comme une base pour le développement du modèle statique et dynamique. Nous allons décrire dans le chapitre suivant la conception du notre système.

CHAPITRE IV

CONCEPTION

1. Introduction

Nous arrivons maintenant à la dernière phase de la modélisation avec UML après la modélisation des besoins puis l'organisation de la structure de la solution. La conception consiste à construire et à documenter précisément les classes et les tables.

Dans ce chapitre nous allons effectuer une conception préliminaire et une conception détaillée.

2. Conception préliminaire

La conception préliminaire est certainement l'étape la plus délicate du processus 2TUPcar elle en représente le cœur. C'est en effet à cette occasion que s'effectue la fusion des études fonctionnelles et techniques. En conséquence, cette étape permet de passer de l'analyse objet à la conception, c'est-à-dire adapter la conception aux spécifications fournies par l'analyse et intégrer les fonctions métier et applicatives du système dans l'architecture technique. La première étape de ce chapitre est la conception du modèle de déploiement du système projeté. A partir du déploiement, nous pouvons définir les composants qui seront administrés par l'exploitant du système. **[8]**

2.1 Développement du modèle du déploiement

Dans cette étape on présente l'architecture physique supportant l'exploitation du système. Cette architecture comprend des nœuds correspondant aux supports physiques (serveurs, routeurs) ainsi que la répartition des artefacts logiciels (bibliothèques, exécutables…) sur ces nœuds, le déploiement se construit sur la définition des postes de travail.

- **Architecture adoptée**

Le choix de notre solution s'est porté sur une architecture 2 tiers :

Un client connecté à un serveur au niveau de la wilaya et cela en se basant sur un réseau MAN. Ce choix repose sur les arguments suivants :

- L'existence d'un réseau haut débit entre la commune et la wilaya.
- L'utilisation d'un serveur au niveau de la wilaya.

- Le serveur local de la wilaya connecte avec les autres serveurs locaux et aussi avec le serveur central.

➢ **Notre modèle de déploiement :**

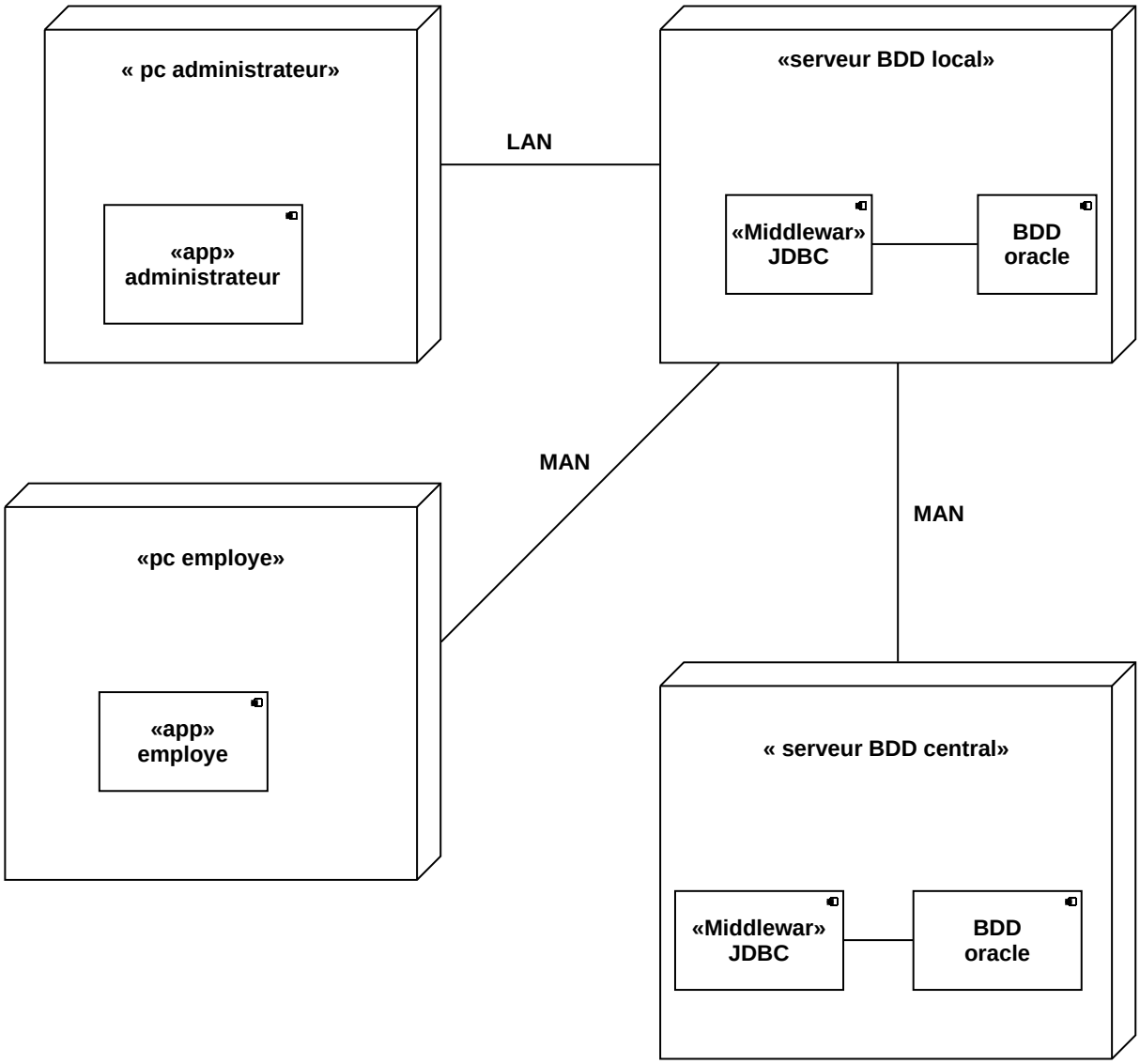

Figure IV-1 : Définition des applications dans le modèle d'exploitation.

2.2 Définition des interfaces

Espace	Interface	description
Administrateur	Gestion compte	Ajouter, modifier, activer/désactiver un compte.
		backup /restauration.
Employé	Gestion de la carte grise.	Ajouter nouvelle carte grise. Modifier carte grise. Ajouter duplicata. Radier carte grise
	Gestion de la vente locale.	ajouter vente locale. annuler vente locale.
	Gestion de la fiche de contrôle.	Ajouter fiche de contrôle. Modifier fiche de contrôle. Annuler fiche de contrôle.
	Etablir fiche de confirmation.	Etablir fiche de confirmation.
	Annuler Gage, Opposition, Incessibilité.	Annuler Gage, Opposition, Incessibilité.
	Etablir avis de mutation	Etablir avis de mutation

Tableau IV-1 : Les interfaces de notre système**.**

3. Conception détaillé

La conception détaillée est la phase ultime de la modélisation avec UML. Après la modélisation des besoins puis l'organisation de la structure de la solution, la conception détaillée vient construire et documenter précisément les classes, les interfaces, les tables et les méthodes qui constituent le codage de la solution.

3.1 Conception des classes

Classe	Description	Codification	Type
Carte grise	Numéro de matricule	Immatriculation	string
	La date de création de la carte grise	Date_Creation	date
	La dernière date de mise à jour de la carte	Date_Maj	date
	Le code marque de la voiture	Code_Marque	string
	Le code de la structure de la voiture	Code_Struct	string
	Le code type de la voiture	Code_Type	string
	Le numéro de châssis de la voiture	Numero_Chassis	string
	La puissance de la voiture	Puissance	string
	Nombre de places de la voiture	Nombre_Places	string
	La charge utile de la voiture	Charge_Utile	string
	La charge totale de la voiture	Total_Charge	string
	La wilaya origine de la voiture	Wilaya_Origine	string
	L'ancien matricule de la voiture	Ancien_Immatr	string
	Le code énergie de la voiture	Code_Energie	integer
	Le numéro du procès-verbal	Num_Pv	string
	La date du procès-verbal	Date_Pv	date
	Année de circulation de la voiture	Annee_Circulation	string
	Le montant du réception	Montant_Rec	double
Propriétaire	Numéro du propriétaire	NumProp	integer
	Nom arabe du propriétaire	NomArabe	string
	Prénom arabe du propriétaire	PrenomArabe	string
	Nom latin du propriétaire	NomLatin	string
	Prénom latin du propriétaire	PrenomLatin	string
	Date de naissance	Date_N	string

	Lieu de naissance	Lieu_N	string
	Prénom du père	FilsDe	string
	Nom et prénom du mère	EtDe	string
	La fonction du propriétaire	Fonction_Prop	string
	Adresse du propriétaire	Adresse	string
	Code de la commune du propriétaire	Codec	string
	Date d'achat de la voiture	Date_Achat	date
	Nationalité du propriétaire	nationalite	string
	Etat du propriétaire	etat	integer
Duplicata	Nombre de copie du duplicata	Ordre	integer
	Le type du duplicata	TypeDuplic	integer
	Observation du duplicata	observation	string
	La date du duplicata	Date_dup	date
Radiation	Motif de la radiation	motif	string
	Observation de la radiation	observation_rad	string
	La date de radiation	Date_rad	string
Opposition	Numéro opposition	Num_Oposition	string
	Date opposition	Date_Oposition	string
	Durée opposition	duree_oposition	integer
	Date fin opposition	date_fin_op	date
	Autorité opposition	autorite_op	string
Gage	Nom du gageur	Nom_Gageur	string
	Date du gage	date_gage	date
	Reference du gage	reference_gage	string
	Date fin du gage	Date_fin_Gage	string
Incessibilité	Date incessibilité	Date_Incess	date
	Durée incessibilité	Duree_Incess	integer
	Autorité incessibilité	autorite_Incess	string

	Date fin incessibilité	date_fin_incess	date
Avis de mutation	La référence du mutation	Reference_mut	string
	Le code du nouvelle wilaya	Code_nov_w	string
	La nouvelle matricule de la voiture	Nouveau_Immatr	string
	La date de mutation	date	date
Fiche de contrôle	Numéro de fiche de contrôle	NumFicCont	string
	Code wilaya	Code_Wilaya	string
	Ordre de fiche de contrôle	Ordre_cont	integer
	Acheteur de la fiche de contrôle	Acheteur	string
	Date de la fiche de contrôle	Date_cont	date
Fiche de confirmation	Ordre de fiche de confirmation	Ordre_conf	integer
	Date de fiche de confirmation	Date_conf	date
	Reference de fiche de confirmation	Reference	string
	Date de référence	DateRef	date
	Destination	Destination	string
	Code wilaya	Codew	string
	Code daïra	CodeD	string

Tableau IV-2: Dictionnaire de données avec Les classes et les attributs

3.2 Les opérations

Classe	Opération	Description
Carte grise	Ajouter	Ajouter une carte grise d'une vente locale ou bien d'une vente externe ou une nouvelle voiture.
	Modifier	Modifier une carte grise déjà créer.
Propriétaire	Ajouter	Ajouter un propriétaire d'une carte grise.
Duplicata	Ajouter	Ajouter un duplicata de la carte grise.
Radiation	Ajouter	Ajouter une radiation de la carte grise.
Opposition	Ajouter	Ajouter une opposition pour une carte grise.
Gage	Ajouter	Ajouter une Gage pour une carte grise.
Incessibilité	Ajouter	Ajouter une incessibilité pour une carte grise.
Avis de mutation	Ajouter	Ajouter un avis de mutation pour une carte grise.
Fiche de contrôle	Ajouter	Ajouter une nouvelle fiche de contrôle.
	Modifier	Modifier une fiche de contrôle déjà créer.
	Annuler	Annuler une fiche de contrôle déjà créer.
Fiche de confirmation	Ajouter	Ajouter une nouvelle fiche de confirmation.

Tableau IV-3: Dictionnaire de données avec Les opérations

4. Diagramme de classe détaillé

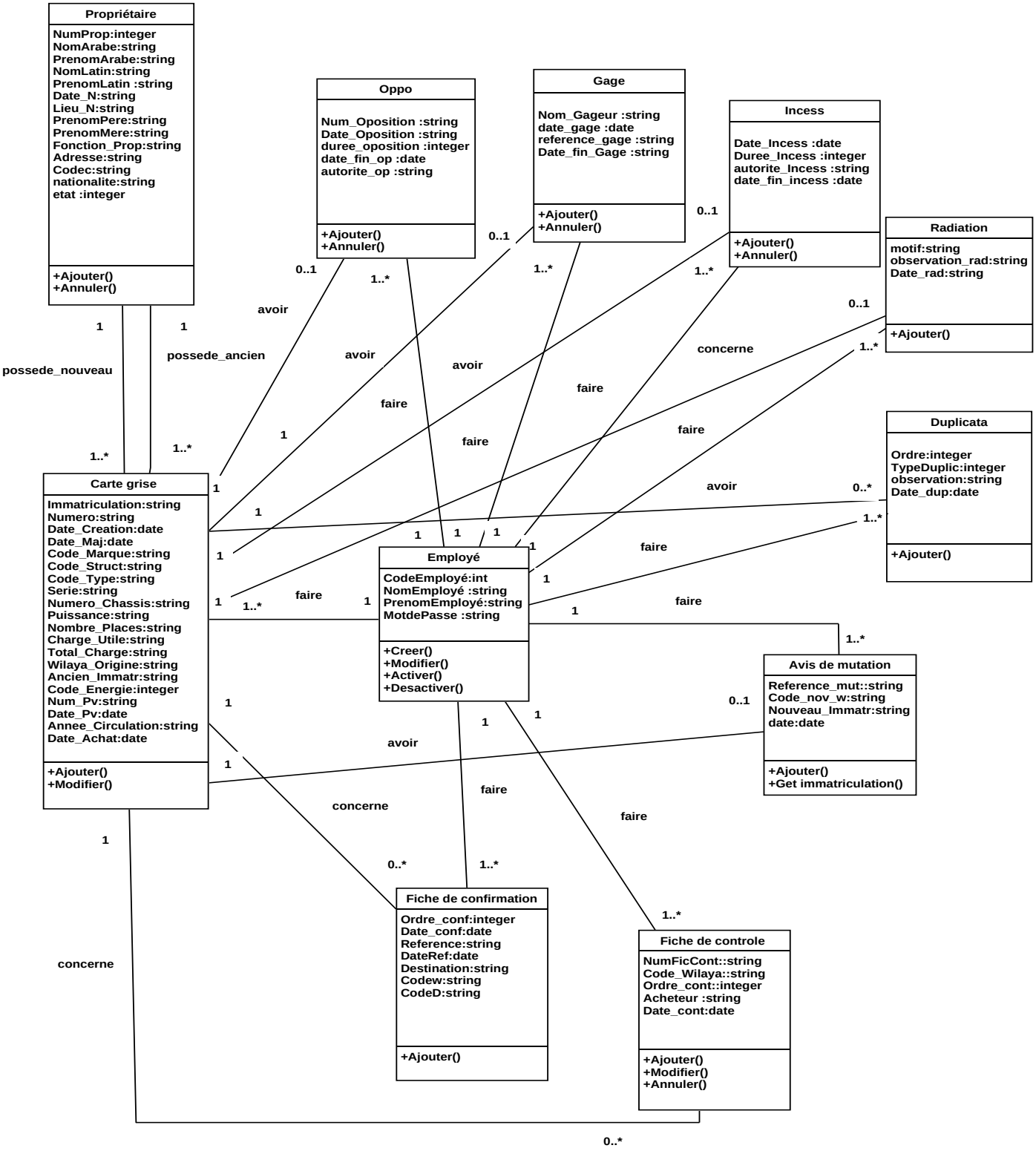

Figure IV-2 : Diagramme de classe détaillé

5. Le modèle relationnel

A partir de la description conceptuelle que nous avons effectuée, on peut réaliser le modèle relationnel ; vu que le système d'information ne peut pas le manipulé directement; et ça en utilisons des règles de passages de l'UML vers le relationnel.

Quelques notions essentielles :

- Domaine : c'est l'ensemble des valeurs d'un attribut.
- Relation : c'est un sous ensemble du produit cartésien d'une liste de domaines.
- C'est en fait un tableau à deux dimensions dont les colonnes correspondent aux
- Domaines et dont les lignes contiennent des tuples. On associe un nom à Chaque colonne.
- Attribut : c'est une colonne d'une relation, caractérisé par un nom.
- Tuples : c'est la liste des valeurs d'une ligne d'une relation.
- Cardinalité : elle permet de définir les conditions de participation d'une entité àune relation. Toutefois, une entité peut participer à plusieurs relations.
- L'arité : est le nombre d'attributs d'une relation.
- Clé : On distingue deux types de clés :

    - **Clé primaire :** Ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres (notion d'identifiant).

    - **Clé étrangère :** Attribut qui est clé primaire d'une autre entité.

NB : pour la notation, nous avons choisi de mettre en gras les clés primaires et de mettre # au début de chaque clé étrangère.

5.1 Les règles de passage

- **Règle 1 :** (Transformation des classes) chaque classe du diagramme UML devient une relation, il faut choisir un attribut de la classe pouvant jouer le rôle de clé.

- **Règle 2 :** (Association 1..*) il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.

- **Règle 3 :** (Association 1..1) il faut ajouter un attribut de type clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association. Si les deux multiplicités minimales sont à un, il est préférable de fusionner les deux classes en une seule.

- **Règle 4 :** cas héritage, transformer chaque sous classe en une relation, la clé primaire de la super classe devient clé primaire de chaque sous classe.

- **Règle 5 :** cas de composition, migration de clé primaire de la super classe composé clé étranger de la classe composant.

5.2 Les règles de gestion

- Le propriétaire possède une ou plusieurs carte grise .
- Le propriétaire possède une ou plusieurs ancien carte grise.
- La carte grise peut avoir zéro ou une opposition .
- La carte grise peut avoir zéro ou un gage.
- La carte grise peut avoir zéro ou une incessibilité.
- La carte grise peut avoir zéro ou une radiation.
- La carte grise peut avoir zéro ou plusieurs duplicatas.
- La carte grise peut avoir zéro ou un avis de mutation.
- Une fiche de confirmation concerne une et une seule carte grise.
- Une fiche de confirmation concerne une et une seule carte grise.

6. Les tables de la base de données

En se basant sur les règles ci-dessus, nous avons converti les classes entités et leurs associations, à des tables dans la base donnée. Les tables générées sont :

Les tables de la base de donnée
Carte grise (**Immatriculation**, **Numero_Chassis**, Date_Creation, Date_Maj, #NumProp, Code_Marque, Code_Struct, Code_Type, Puissance, Nombre_Places, Charge_Utile,Total_Charge, Wilaya_Origine, Ancien_Immatr, Code_Energie, Num_Pv, Date_Pv, Numero_Rec, Montant_Rec, Annee_Circulation,Date_Achat) ; **Propriétaire** (**NumProp**, #Immatriculation, NomArabe ,PrenomArabe,NomLatin,PrenomLatin, Date_N , Lieu_N, FilsDe, EtDe,Fonction_Prop, Adresse, Codec, nationalite) **Duplicata** (#**Immatriculation**, #**Numero_Chassis**, Date, TypeDuplic, NumProp, Observation) ; **Radiation** (#**Immatriculation**, #**Numero_Chassis**, Date, motif, NumProp, Observation) ; **Fiche de contrôle** (#**Immatriculation**, #**Numero_Chassis**, Ordre_conf, Date_conf, Reference, DateRef, Destination, Codew, CodeD) ; **Fiche de confirmation** (#**Immatriculation**, #**Numero_Chassis**, Ordre_conf, Date_conf, Reference, DateRef, Destination, Codew,CodeD) ; **Avis de mutation** (#**Immatriculation**, #**Numero_Chassis**, Reference_mut, Code_nov_w, Nouveau_Immatr, date) ; **Agent** (CodeAgent, NomAgent, MotdePasse ,type) ; **Opposition** (#**Immatriculation**, #**Numero_Chassis** ,Num_Oposition ,Date_Opositionduree_oposition ,date_fin_op ,autorite_op) ; **Gage (**Nom_Gageur ,date_gage,reference_gage ,Date_fin_Gage) ; **Incessibilité** (#**Immatriculation**,#**Numero_Chassis**,Date_Incess ,Duree_Incess, autorite_Incess,date_fin_incess) **;** **Ancien voit** (**Immatriculation**, **Numero_Chassis**, Date_Creation, Date_Maj, #NumProp, Code_Marque, Code_Struct, Code_Type, Puissance, Nombre_Places, Charge_Utile,Total_Charge, Wilaya_Origine, Ancien_Immatr, Code_Energie, Num_Pv, Date_Pv, Numero_Rec, Montant_Rec, Annee_Circulation,Date_Achat) ;

Tableau IV-4 : Les tables de la base de données

7. Conclusion

La phase de conception prend en compte les choix d'architecture technique retenue pour le développement et l'exploitation du système. Elle vise à trouver des solutions informatiques et techniques pour mettre en œuvre et construire le système analysé au cours des phases précédentes.

Il nous reste qu'une dernière étape qui est la phase de réalisation qu'on va l'aborder dans le chapitre suivant.

Chapitre V

Implémentation

1. Introduction

Ce chapitre est consacré à la réalisation et la mise en œuvre de notre application distribué pour la gestion de la carte grise.

En première partie nous allons présenter les outils de développement adoptés ,le système de gestion de base de données **Oracle**, le langage de programmation **Java**, ainsi que l'environnement utilisé qui est **Netbeans** aussi nous expliquons la démarche suivie, et les étapes de création de l'environnement du travail à savoir un réseau **Ad-hoc**.

Dans la seconde partie, nous allons présenter la solution proposée, et qui devra répondre aux exigences de la gestion de la carte grise en détaillant les étapes qui la composent ,et enfin nous montrons les principales interfaces de l'application.

2. Langage et outils de développement

2.1. Le langage de programmation java

Java est un langage de programmation évolué et orienté objet développé par Sun Micro Systems.il est très largement utilisé dans le domaine de développement notamment le développement d'application d'entreprises et mobiles.[9]

2.2. IDE NetBeans

Notre choix s'est porté vers l'IDE NetBeans. C'est un environnement de développement intégré, open source, très utile qui permet le développement en java. Son grand atout est le confort et la Simplicité qui offre pour un développement propre et rapide.[10]

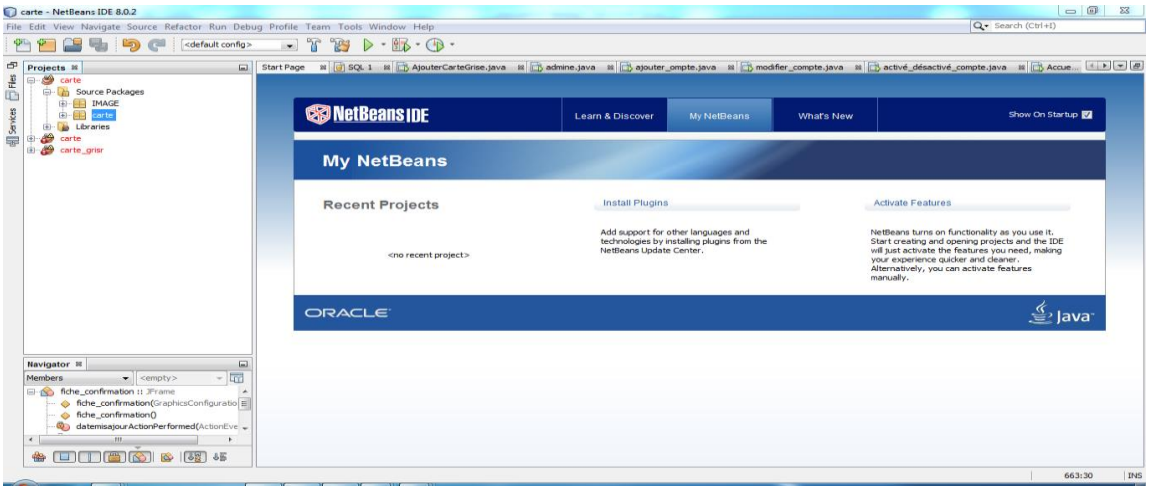

Figure V-1 : Capture écran de l'IDE NetBeans.

2.2.1. Connecteur

On a utilisé le connecteur (ojdbc7) pour connecter notre logiciel avec la base de données.

3. Système de gestion de base de données (Oracle)

Pour l'implémentation de la base de données que notre système utilise, nous avons sélectionné le SGBD Oracle, qu'est un système de gestion de bases de données édité par la société du même nom (Oracle Corporation), leader mondial des bases de données. Oracle se décline en plusieurs versions. Oracle Server **Standard**, une version comprenant les outils les plus courants de la solution Oracle.

Oracle Server **Enterprise Edition**, Oracle est un SGBD permettant d'assurer :[10]

- La définition et la manipulation des données.
- La cohérence des données.
- La confidentialité des données.
- L'intégrité des données.
- La sauvegarde et la restauration des données.
- La gestion des accès concurrents.

3.1. Oracle SQL Développer

Comme outil de développement de base de données on a utilisé « Oracle SQL Developer» qui est un environnement de développement intégré (EDI) multiplateformes, fourni gratuitement par Oracle Corporation et utilisant la technologie Java. C'est un outil graphique permettant d'interroger des bases de données Oracle à l'aide du langage SQL.[10]

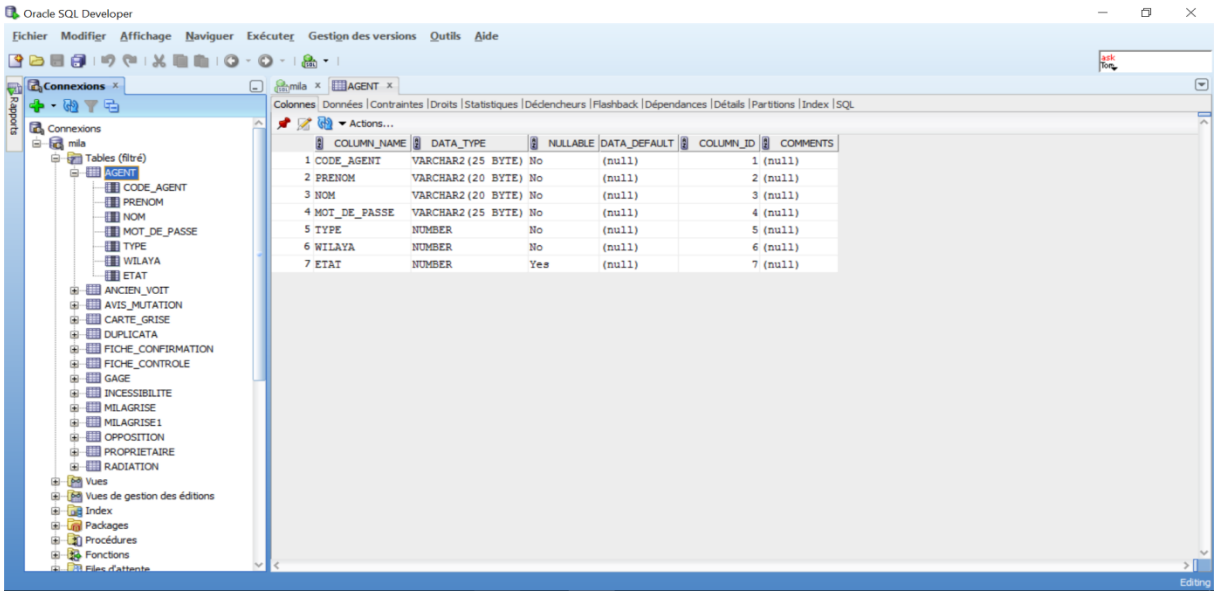

Figure V-2 : Interface Oracle SQL Developer.

4. Structure générale de la solution proposée

Notre solution consiste à créer sur chaque wilaya une base de données interfacée avec notre application de gestion de la carte grise. Chaque serveur de wilaya partage ses données avec les autres serveurs. Ainsi, ces différentes bases seront consolidées au niveau de la base centrale pour avoir une base globale qui contiendra toutes les données.

4.1.Justification des choix

1. La création des vues matérialisées « **materializedviews**» au niveau du base centrale, nous permet d'avoir une localisation physique des données au niveau de cette dernière et un rafraîchissement complet des données. Donc, on favorise les accès locaux qui ont pour but de réduire le trafic réseau.

2. En cas de panne d'un site ou d'indisponibilité du réseau, les données seront disponibles, car les accès aux données sont locaux.

3. Comme l'utilisateur a besoin de toutes les données de tous les sites, la création de vues «**views**» au niveau de la base centrale, nous permet d'avoir accès à toutes les données en toute transparence.

4. Avec cette solution, une grande disponibilité des données est assurée.

5. création des liens entre les serveurs qui permet l'interconnexion entre eux.

4.2. Configuration du réseau

Les ordinateurs dont nous disposons vont être reliés grâce à un réseau sans fil « **Ad hoc** » afin qu'ils puissent se connecter à distance à la base **ORACLE.**

4.3. Configuration ORACLE

Nous installons **ORACLE 12c**sur le poste centrale qui contient la base globale et sur chaque serveur des wilayas qui contient une base locale. Tandis que notre application sera installé sur chaque poste utilisateur.

Ensuite, nous configurons la couche réseau grâce au **Net Configuration Assistant** pour que les serveurs puissent se communiquer.

Une fois oracle installé, et notre base de données créée et démarrée, elle est prête à l'utilisation, mais elle n'est pas encore prête pour être accessible via le réseau.

4.3.1. Le processus d'écoute Oracle

Le processus d'écoute Oracle est un service permettant à des clients d'utiliser le protocole **TCP** pour accéder une base de données distante. Son port d'écoute par défaut est 1521.

4.3.2. Création des services de base de données

Le processus d'écoute autorise uniquement les demandes de connexions des utilisateurs pour les noms de services inscrits dans le **listener.ora** et chaque nom de service référence une base de données.

Nous prenons à titre d'exemple la création du service cgrisemila pour la BD Mila Pour ce faire, il faut saisir dans le **Oracle Net Manager** :

- Le nom du service : cgrisemila.

- Le protocole connexion réseau : TCP.

- Le nom de la machine ou l'adresse IP : *192.168.43.30*

- Saisir le N°du port : *1521* .

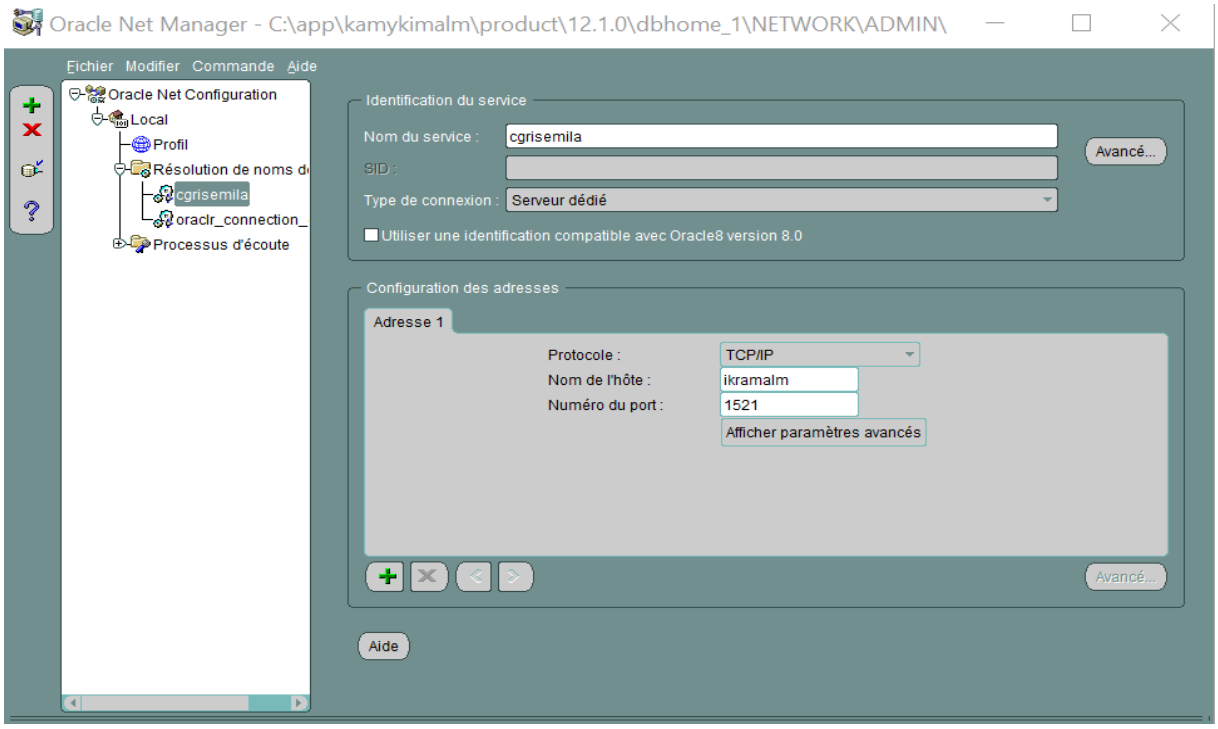

Figure V-3 : Oracle Net Manager.

5. Implémentation de la BDR

- **Au niveau de la wilaya de Mila**

-Création d'un compte utilisateur nommé cgrisemila :

Create user **cgrisemila** identified by **stic**; Grant all privileges to**cgrisemila**;

-Création du lien de BD entre la base Mila et le serveur de la base de données centrale :

Create database link **grisecntr** connect to **central** identified by **stic** using '(description=(address=(protocol=tcp)(host=serveur central)(port=1521))(connect_data=(sid=

cgrisecentral)))';

-Création du lien de BD entre le serveur locale du base Mila et le serveur locale de la base Constantine :

Create database link **grisec** connect to **constantine** identified by **stic** using '(description=(address=(protocol=tcp)(host=serveur Constantine)(port=1521))(connect_data=(sid=**cgriseconsta**)))';

-Création des tables de la base de données Mila Au niveau du serveur du base locale de la wilaya de Mila.

- **Au niveau de la wilaya du constantine**

Nous suivons le même processus d'implémentation de la base de donnée au niveau de la wilaya de Mila.

- **Création des vues matérialisées au niveau de la base central**

Pour implémenter la réplication des données de différentes bases locales sur la base centrale on a utilisé la notion des vues matérialisé. Il y a deux types de vues matérialisé :

a- Vue matérialisé à lecture seul : leur contenu n'est pas modifiable donc elles sont utilisées pour la consultation.

b- Vue matérialisé modifiable : les utilisateurs peuvent modifier le contenu de la table ainsi que la vue matérialisé, et le contenu des deux sera synchronisé.

Dans notre cas les utilisateurs doivent pouvoir travailler sur les tables locales ainsi sur les vues matérialisés, donc on va utiliser des vues matérialisés modifiables pour implémenter la réplication des données, donc pour chaque table de la base de données locale de chaque wilaya on va créer une vue matérialisé modifiable dans la base centrale.

Vu que les données ne seront pas modifiées plusieurs fois par jour, la synchronisation se fait une fois par jour à 10 :00.

Afin d'illustrer celles-ci, nous allons donner un exemple de création de deux vues matérialisées ; une pour la table carte grise et l'autre pour avis de mutation qui se trouvent respectivement dans la base de donnée de Mila :

Create materailized view mila_carte_grise

Refresh start with sysdate

Next (sysdate+1)/10/24

For update

*As select*from CARTE_GRISE@central;*

Create materailized view mila_avis_mutation

Refresh start with sysdate

Next (sysdate+1)/10/24

For update

*As select*from AVIS_MUTATION@central;*

Pour synchroniser les données entre les tables et les vue matérialisés modifiables on a utilisé la méthode de « réplication des données par vue matérialisé » proposé par le site officiel d'Oracle. [11]

Pour que les changements effectués sur une vue matérialisé modifiable soient recopiés dans la table maitre pendant la synchronisation elle doit appartenir à un groupe de vue matérialisé.

La déclaration suivante crée un groupe de vue matérialisé:

```
BEGIN
   DBMS_REPCAT.CREATE_MVIEW_REPGROUP (
gname => 'mila_repg',
master => 'cgrisemila',
propagation_mode => 'ASYNCHRONOUS');
END;
/
```

La déclaration suivante ajoute la vue matérialisé *mila_carte_grise* au groupe de vue matérialisé, faisant la vue matérialisé modifiable :

```
BEGIN
   DBMS_REPCAT.CREATE_MVIEW_REPOBJECT (
gname => 'mila_repg',
sname => 'mila_carte_grise',
oname=> 'cgrisemila',
type => 'SNAPSHOT',
min_communication => TRUE);
END;
```

6. Description de l'application

Dans cette partie nous allons présenter les interfaces principales de notreapplication.

- **L'interface d'authentification**

Cette interface permet aux utilisateurs d'accéder à l'application.

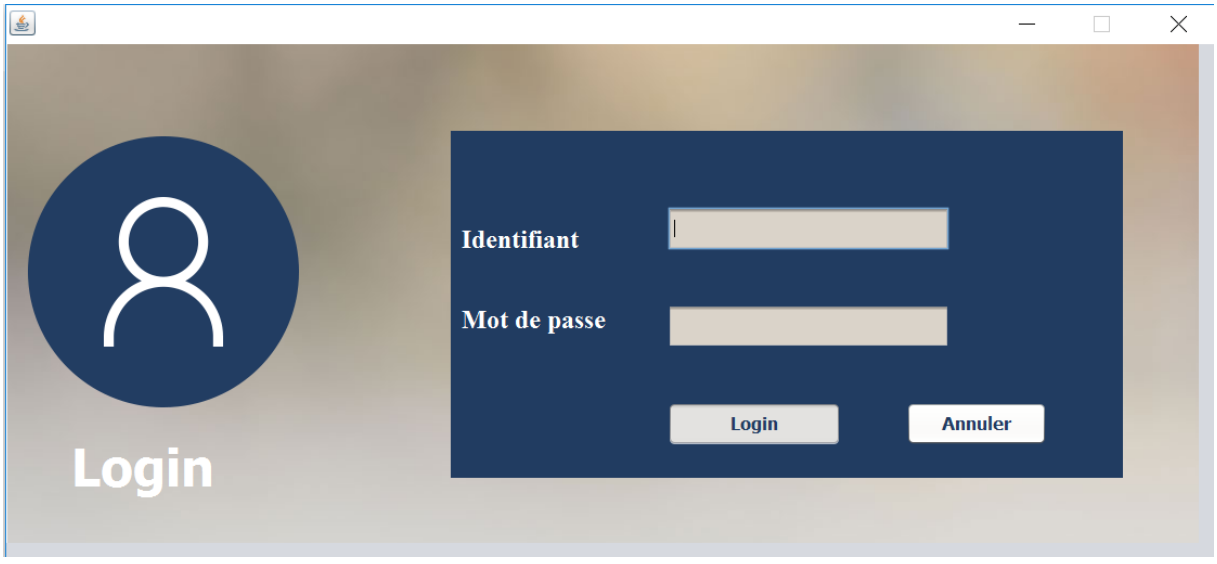

Figure V-4:L'interface d'authentification

- **Accueil employé**

Cette interface permet au employé de sélectionné l'un de ces cas pour la gestion du carte grise.

Figure V-5 : Accueil employé

- **Fenêtre ajouter nouvelle carte grise**

Cette fenêtre permet au l'employées d'ajouté une nouvelle carte grise.

Figure V-6 : Fenêtre ajouter nouvelle carte grise

7. Conclusion

La phase d'implémentation a été concrétisée par la réalisation de l'application, en respectant la modélisation qu'on a présentée dans le chapitre précédent pour la conception de notre projet .Nous avons aussi présenté l'environnement de développement et quelques interfaces du notre système.

Une base de données répartie est une collection de données logiquement unies et physiquement réparties sur plusieurs machines interconnectées par un réseau de communication. Elle permet d'intégrer et de partager des données gérées par des systèmes de gestion des bases de données réparties.

L'objectif que nous avons visé lors ce projet est la réalisation d'un système d'information en exploitant le réseau local existant. Pour cela, nous avons choisi **ORACLE 12c** comme SGBD. Ce choix est justifié par sa puissance et son efficacité, en terme de sécurité, volume de données traitées, ... etc.

Pour ce faire, nous avons commencé par la création des bases de données qui modélisent la carte grise de chaque wilaya, puis consolider ces bases au niveau du serveur central et nous avons construit les interfaces graphiques permettant de réagir avec cette base grâce à l'IDE NetBeans.

Ce projet nous a permis de se familiariser avec ORACLE, d'approfondir nos connaissances dans le domaine des bases de données réparties et d'acquérir des connaissances sur *Oracle* et NetBeans IDE.

Malgré ce qui a été fait, ce travail n'est pas encore terminé il peut être enrichi par d'autres fonctionnalités telles l'utilisation d'une architecture à trois tiers qui consiste à séparer logiquement l'architecture du système en trois couches logicielles. Enfin, nous espérons que ce modeste mémoire soit un modèle pour les autres étudiants et que sa lecture a été agréable et claire.

Résumé

Une base de données répartie est une collection de données logiquement reliées et physiquement réparties sur plusieurs machines interconnectées par un réseau de communication. L'objectif de ce travail est de réaliser une application de gestion de carte grise qui profitent des avantages de la répartition des bases de données et plus précisément l'augmentation des performances et tolérance aux pannes.

La conception et la mise en œuvre de notre base de données est répartie entre les différents serveurs locaux de chaque wilaya et le serveur central, nous avons utilisé UML comme langage de modélisation adopté au processus 2TUP qui répond le plus à notre besoin. Et Pour l'implémentation, nous avons porté notre choix respectivement sur l'environnement de développement Netbeans et le système de gestion de base de données Oracle 12C.

Abstract

A distributed database is a collection of logically linked and physically distributed data on multiple machines interconnected by a communication network. The objective of this work is to realize a management application of the gray card which take advantage of the advantages of the distribution of the databases and more specifically the increase of the performances and fault tolerance. The design and implementation of our database is split between the different local servers of each wilaya and the central server, we used UML as the modeling language adopted in the 2TUP process that best meets our need. And for implementation, we chose the Netbeans development environment and the Oracle 12C database management system.

BIBLIOGRAPHIE

❖ MÉMOIRES

[1] Bouanani Hanane ;Suivi et gestion répartie des clients d'une banque Cas de la Banque BADR Université Abou Bakr Belkaid – Tlemcen ;2013.

[2] Hakim MADI ; Conception et réalisation d'une base de données repartie sous oracle. Mémoire, Université A/Mira de Bejaia ;2009.

[3] Systèmes de Gestion de Bases de Données Réparties & Mécanismes de Répartition avec Oracle Cours, Par Rim Moussa ; Université 7 Nov - à Carthage ;2005-2006.

[4] Gestion d'hébergement des Résidences Universitaires de Tlemcen Sous Oracle. Mémoire de fin d'études pour l'obtention du diplôme de Master2 en Informatique , Université de Tlemcen 2012-2013.

[10] l. Nawal N. Imen ; Application client/serveur pour le suivi des dossiers au niveau de la CNR de Mila Centre universitaire de Mila ; 2015.

❖ OUVRAGES

[6] Pascal Roques & Frank Vallée : « UML en action de l'analyse des besoins à la conception en Java » édition Groupe Eyrolles, 2007.

[8] UML 2 en action de l'analyse des besoins à la conception 4e édition, Pascal Roque, Franck Vallée

[9] C.Herby, Apprenez à programmer en JAVA, 2011.

❖ SITES WEB

[5] http://www.wilaya-mostaganem.dz/fr/ecitoyen/vos-demarches/carte-grise

[7] http://niedercorn.free.fr/iris/iris1/uml/uml08.pdf

[11] https://docs.oracle.com/cd/B28359_01/server.111/b28324/tdpii_reppit.htm#TDPII102