الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……..… /13

Centre Universitaire de Mila

Institut des sciences et de la technologie **Département de Mathématiques et Informatique**

SYSTÈME DE GESTION DU RÉFECTOIRE D'UN ÉTABLISSEMENT SCOLAIRE

Mémoire préparé en vue de l'obtention du diplôme de licence en Informatique

Préparé par :

1- Zaimeche Mourad

2- Harizi Rami

Encadré par :

- Talai Meriem

Filière : Informatique

Année universitaire : 2012/2013

REMERCIEMENT

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, nous tenons à remercier notre encadreur « Talai Meriem », son précieux conseil et son aide durant toute la période du travail.

Nos remerciements s'étendent également le directeur et le magasinier du lycée «Kamel abd allah bacha » pour ses bonnes explications qui nous ont éclairé le chemin de la recherche de ce modeste travail.

Nous tenons à exprimer nos sincères remerciements à tous les professeurs du « centre universitaire de mila » qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, et toutes les personnes qui ont participé de près ou de loin qui m'ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.

Rami et Mourad

DEDICACES

Au Début et avant tous, je veux remercier le dieu qui à permet le courage à faire et finir ce modeste travail.

C'est avec un grand plaisir et une réelle joie de fierté que je dédie ce travail.

Aux êtres les plus chers dans ma vie,

Mes parents à qui je dois les remercier beaucoup pour leur soutien moral

À mon frère Issam

À ma sœur Malak

À toute la famille Harizi et Bouarroudj

À mon binôme Mourad ainsi que toute sa famille

À tous les professeurs du « CUM » qui nous ont enseigné

A tous mes amis du centre universitaire de mila

Surtouts Raouf, Ilyas, Fateh

A la fin, nous remercions tous ceux qui ont aidé

De prés ou de loin à réaliser notre travail.

RAMI

DEDICACES

Je dédie ce modeste travail aux êtres les plus chers dans ma vie,

mes parents à qui je doit les remercier beaucoup pour leur soutien moral

À mes frères Abdelouahab, et Oussama.

À ma petite belle-sœur Iness.

À mes très très chères amies Khawla.

À tous mes amis en particulier Raouf, Ilyas

À touts mes enseignants du primaire jusqu'aujourd'hui.

À mon binôme Rami ainsi que toute sa amille.

À toute la famille Zaimeche.

À tous mes collègues de la section.

À tous ceux qui me connaissent.

MOURAD

Table des matières

Introduction général 7

Chapitre 01 : Réfectoire 9

1.1.Introduction 10

1.2.Restauration scolaire -réfectoire 10

1.3.Fonctionnement du réfectoire 10

1.4 . Tâches du magasinier 11

1.4.1 Préparation des listes de repas par semaine 11

1.4.2 Suivi du processus d'approvisionnement 11

a. Commande des aliments 11

b. Réception des aliments 12

c. Livraison de la commande 12

d. Stockage des aliments 12

1.4.3 Gestion de la consommation 12

1.5 Tâches de l'économiste 13

1.6 Tâches du Directeur de l'établissement 13

1.7 Tâches du cuisson 13

Chapitre 02 : Présentation UML et du processus unifié UP 14

2.1 Introduction 15

2.2 Langage de modélisation 15

2.2.1 Qu'est ce que un langage de modélisation ?.....15

2.2.2 Forme des langages de modélisation 15

2.2.3 Domaines Fonctionnels 15

2.3 Exemples de langage de modélisation 16

2.3.1 langage EXPRESS (ISO 10303 -11) 16

2.3.2 langage UML (Unified Modeling Language) 14

2.4 UML (Unified Modeling Language).....18

2.4. 1 Présentation d'UML.....18

2.4.2 Historique 18

2.4.3 Mise en œuvre d'une démarche a l'aide d'UML : les vues 19
2.4.4 Diagrammes d'UML 20
a-Diagrammes structurels.....20
b- Diagrammes comportementaux 21
c- Diagrammes d'interaction 22
2.4.5 Les éléments de modélisation d'UML 22
a. Les éléments structurels 22
b. Les éléments comportementaux 24
c. Les relations 25
2.5 Le processus unifié UP 25
2.5.1 Définition/Présentation 25
2.5.2 Les quatre 'P' du processus unifié 26
2.5.3 PU piloté par les cas d'utilisation 26
2.5.4 PU centré sur l'architecture d'utilisation 27
2.5.5 PU itératif et incrémental 27
Chapitre 03 : langage de programmation (Delphi) 29
3.1 INTRODUCTION 30
3.2 Langage de programmation 30
3.2.1 Classement du langage de programmation 30
3.3 Visual Basic (VB) 31
3.3.1 Historique 31
3.3.2 Fonctionnalités du langage 32
3.3.3 Caractéristiques de Visual Basic 32
3.4 Delphi 34
3.4.1 Présentation 34
3.4.2 Les caractéristiques de Delphi 34
3.4.3 Delphi et les autres 34
3.4.4 Principes de développement Delphi 35
3.4.5 Avantages du langage Delphi 36

3.4.6 Delphi et Windows 36
3.4.7 L'interface de Delphi 37
Chapitre 04 : Etude de Cas 38
4.1 Introduction 39
4.2 Identification des cas d'utilisations 39
4.3 Description des cas d'utilisation 39
4.3.1 Cas d'utilisation «authentification» 40
4.3.2 Cas d'utilisation « Etablir bon commande » 41
4.3.3 Cas d'utilisation « Réception des aliments » 42
4.3.4 Cas d'utilisation «Consommation des aliments» 43
4.3.5 Cas d'utilisation « renseigner des articles » 44
4.3.5.1 Cas Ajouter article 44
4.3.5.2 Cas Modifier article 45
4.3.5.3 Cas Supprimer article 46
4.3.6 Cas d'utilisation « Gestion fournisseur » 47
4.3.6.1 Cas Ajouter fournisseur 47
4.3.6.2 Cas Modifier fournisseur 48
4.3.6.3 Cas Supprimer fournisseur 49
4.3.7 Cas d'utilisation « Gestion des familles » 50
4.3.7.1 Cas Ajouter famille 50
4.3.7.2 Cas Modifier famille 51
4.3.7.3 Cas Supprimer famille 52
4.3.8 Cas d'utilisation « Etablir menu des repas » 53
4.3.8.1 Cas Etablir une liste des repas 53
4.3.9 Cas d'utilisation « Renseigner les informations administratives » 54

Chapitre 05 : Implémentation 85
5.1 Introduction 86
5.2 Modèle relationnel 86
5.2.1 Définition 86
5.2.2 Le passage du diagramme de classe au modèle relationnel 86
5.2.3 Représentation de notre base de données 87
5.3 La base de données ACCESS 2010 89
5.4 Les interfaces de l'application 90
5.4.1 Interface authentification 90
5.4.2 Interface menu principale 91
5.4.3 Interface Renseigner des articles.....91
5.4.4 Interface ajouter article 92
5.4.5 Interface recherche pour modifier article 93
5.4.6 Interface modifier article 93
5.4.7 Interface supprimer article 94
5.4.8 Interface établir bon commande 94
5.4.9 Interface réception des aliments 95
5.4.10 Interface consommation des aliments 95
5.4.11 Interface établir menu des repas 96
5.4.12 Interface informations administrative 96
5.4.13 Interface imprimer bon de commande 97
5.4.14 Interface imprimer consommation des aliments 97
Conclusion générale 98
Bibliographie 99

Liste des figures

Figure 1 : représentation de établissement par diagramme EXPRESS 17

Figure 2 :Vues d'UML 19

Figure 3 : la hiérarchie des diagrammes UML 2.0 sous forme d'un diagramme de classes 20

Figure 4 : Différentes représentations des classes en UML 23

Figure 5 : Eventuelles représentations des interfaces 23

Figure 6 : Représentations des objets 23

Figure 7 : Représentation des acteurs 24

Figure 8 : Représentation des autres éléments structurels 24

Figure 9 : Représentation des éléments comportementaux 24

Figure 10 : Représentation des éléments modélisation de type relation 25

Figure 11 : UP est piloté par les cas d'utilisation 26

Figure 12 : La vie du Processus unifié 28

Figure 13: L'interface de Delphi 37

Figure14 : Diagramme de cas d'utilisateur 55

Figure15 : Diagramme de séquence système du cas d'utilisation Authentification 56

Figure16 : Diagramme de séquence système du cas d'utilisation Etablir bon commande 57

Figure17 : Diagramme de séquence système du cas d'utilisation Réception des articles 58

Figure18 : Diagramme de séquence système du cas d'utilisation consommation des aliments 59

Figure19 : Diagramme de séquence système du cas d'utilisation renseigner des articles (ajouter articles).....66

Figure20 : Diagramme de séquence système du cas d'utilisation renseigner des articles (modifier articles).....61

Figure21 : Diagramme de séquence système du cas d'utilisation renseigner des articles (supprimer articles).....62

Figure22 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs

(ajouter fournisseur).....63
Figure23 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs (modifier fournisseur).....64
Figure24 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs (supprimer fournisseur).....65
Figure25 : Diagramme de séquence système du cas d'utilisation gestion des familles (ajouter famille).....66
Figure26 : Diagramme de séquence système du cas d'utilisation gestion des familles (modifier famille).....67
Figure27 : Diagramme de séquence système du cas d'utilisation gestion des familles (supprimer famille).....68
Figure28 : Diagramme de séquence système du cas d'utilisation établir menu des repas.....69
Figure29 : Diagrammes d'activités de navigation de cas « s'authentifier ».....70
Figure30: Diagrammes d'activités de navigation : Renseigner des articles (ajouter article).....71
Figure31: Diagrammes d'activités de navigation : Renseigner des articles (modifier article).....72
Figure32 : Diagrammes d'activités de navigation : Renseigner des articles (supprimer article).....73
Figure33 : Diagrammes d'activités de navigation : gestion de fournisseur (ajouter fournisseur).....74
Figure34 : Diagrammes d'activités de navigation : gestion de fournisseur (modifier fournisseur)...75
Figure35:Diagrammes d'activités de navigation:gestion de fournisseur (supprimer fournisseur)....76
Figure36 : Diagrammes d'interactions cas d'utilisation s'authentifier.....77
Figure37 : Diagramme de classes participantes cas d'utilisation s'authentifier.....78
Figure38 : Diagramme de classes participantes : Renseigner des articles (ajouter article).....79
Figure39 : Diagramme de classes participantes : Renseigner des articles (modifier article).....80
Figure40 : Diagramme de classes participantes Renseigner des articles (supprimer article).....81
Figure41 : Diagramme de séquence : Renseigner des articles (ajouter article).....82
Figure42 : Diagramme de séquence : Renseigner des articles (Modifier article).....83
Figure43 : Diagramme de classes.....84

Tableau 1 : Les cas d'utilisations.....39

Introduction Général

Introduction général

La société dépend aujourd'hui de la marée d'information, surtout après le cercle grandissant des connaissances et de la recherche dans divers domaines, et l'émergence de dispositifs électroniques utilisés dans les technologies de l'information. De même, la majorité des institutions peuvent résoudre leurs problèmes grâce à l'utilisation de solutions informatiques et réduire ainsi les coûts d'exploitation et améliorer l'efficience et l'efficacité des temps de réponse et la crédibilité des résultats.

Le sujet de ce rapport est d'une importance dans la conduite de stock des produits d'alimentation. L'objectif principal de notre travail est de sensibiliser à l'importance de l'évolution dans les règles qui aident à la prise de décision, il est des bases de données aux bases de connaissances, à mesure qu'ils deviennent ensemble des connaissances dans ce domaine grâce à la contribution des technologies de l'information et de la communication dans la gestion du cycle de vie des informations.

Étant donné la quantité d'informations manipulée dans l'institution de restauration et sa grande taille, il est utile de poser la question fondamentale suivante:

Comment le domaine de la gestion des stocks peut profiter du développement technologique pour la politique de stockage et d'améliorer les performances.

Nous répondons à cette question sur ce rapport de fin d'études de licence en informatique au long des chapitres structurés comme suit :

Le 1er chapitre est consacré à la présentation du fonctionnement du réfectoire ainsi que les responsabilités du personnel. Dans le 2ième chapitre, nous nous intéressons aux langages de modélisation, particulièrement le langage UML ainsi qu'au processus de modélisation UP. Le 3ième chapitre est consacré aux caractéristiques de Visual Basic et Delphi. Nous y effectuons une analyse comparative entre ces environnements. Le 4ième chapitre présente notre étude de cas, c'est-à-dire la phase de conception et analyse ainsi que la construction des différents diagrammes d'UML. Dans le 5iéme et dernier chapitre nous procédons à la phase d'implémentation. Et enfin, le mémoire se termine par une conclusion générale du travail présenté.

Chapitre 1

Restauration scolaire

1.1. Introduction

Dans ce premier chapitre, nous allons présenter la définition de la restauration scolaire et les fonctionnements du réfectoire.

1.2. Restauration scolaire - réfectoire - :

Appelé parfois cantine, la restauration scolaire est un des secteurs de la restauration collective. Elle remplit une fonction nutritionnelle de base et a un devoir de restitution calorique. Cette restitution calorique, particulièrement dans un établissement scolaire, doit être d'autant plus exacte qu'elle est destinée à une seule catégorie de «consommateurs», ce qui élimine les risques d'erreurs dus à la recherche d'équilibre pour différents types de consommateurs.

La restauration scolaire doit faire l'objet d'un consensus total entre des besoins nutritionnels des consommateurs et les moyens financiers mis à disposition puisqu'elle détermine à elle seule la taille des besoins et donc le dimensionnement et la capacité des moyens à mettre en œuvre. Cependant, elle ne représente qu'une quantité alimentaire partielle puisque ses consommateurs ne sont pris en charge que sur une partie de la journée.

La restauration scolaire, intégrée aux établissements scolaires (écoles, collèges et lycées), remplit aussi une fonction éducative : l'éducation au goût et à l'équilibre alimentaire de ses jeunes convives !

1.3. Fonctionnement du réfectoire:

Un réfectoire est un service municipal assuré par des agents municipaux (magasinier, cuisinier, …etc.) sous la gestion financière de l'économe de l'établissement et sous la responsabilité directe du directeur de l'établissement.

Tous le personnel du service se doit de respecter l'ordre et la discipline nécessaires au bon fonctionnement du service.

Dans ce qui suit, nous allons présenter les tâches communes aux réfectoires scolaires des établissements moyens(CEMs) ou secondaires (lycées). Ces taches sont classées solen leur premier responsable.

1.4. Tâches du magasinier

1.4.1. Préparation des listes de repas par semaine :

- Les magasiniers doivent préparer les éventuels menus des repas par semaine. Ces menus dépendent de plusieurs paramètres :
  - Leur quantification repose sur les normes établies par l'Inspection de l'alimentation scolaire : En effet, dès le commencement de l'année scolaire, l'inspection fournit, aux magasiniers des réfectoires une liste des repas mensuels en tant que canevas calorique à respecter. Elle leur fournit également le cahier de charge établit avec les fournisseurs conventionnés pour alimenter le stock.
  - Les composants d'un menu dépendent du budget prévisionnel par semaine.
  - Les aliments existants au magasin, puisque ces aliments diffère selon l'approvisionnement effectués réellement par les fournisseurs. Sachant que l'inspection de l'alimentation scolaire fournit aux magasiniers le cahier de charge établit avec les fournisseurs conventionnés pour alimenter le stock.

1.4.2. Suivi du processus d'approvisionnement :

L'approvisionnement des réfectoires scolaires est assuré en collaboration avec des fournisseurs choisis et accrédites par le conseil municipal des restaurants scolaire, il suit le processus suivant :

a. Commande des aliments :

Le gestionnaire lance la commande. Il établit un bon de commande par fournisseur par semaine : il doit y indiquer le nom du fournisseur, le numéro du son date d'établissement ainsi qu'une liste des aliments commandés ce bon est dupliqué en 3 exemplaire : Une 1ère copie sera adressée au fournisseur, la seconde à l'économe, le magasinier répertorie la 3éme copie.

b. Réception des aliments :

Lorsque le fournisseur reçoit le bon de commande, il doit y rependre en totalité, c'est-à-dire qu'il doit fournir tous les aliments mentionnés ainsi que les quantités correspondantes. Le magasinier qui supervise la réception de ces aliments, doit inventorier les éléments entrés. Il doit également rejeter tout article qui ne respecte pas le cahier des charges établit avec l'inspection de l'alimentation scolaire ou qui ne respecte pas le bon de commande. Ce pendant, tout dommage des articles après leur réception n'est pas pris en charge par le fournisseur. Dans son registre de réception, le magasinier doit indiquer le numéro du bon commande qui correspondant ainsi que la date de réception. Les aliments périssables doivent être reçus dans la journée de leur consommation.

c. Livraison de la commande :

Une fois que le magasinier valide les éléments reçus fournisseur doit établir un bon de livraison. Ce bon est daté du jour de réception et est signé par le fournisseur et le magasinier. Il doit également indiquer les quantités et les prix unitaires des produits livrés. Le fournisseur remet une copie de ce bon à l'économiste.

d. Stockage des aliments :

Le stockage des aliments doit respecter les règles d'hygiène et de santé (humidité, chaleur, …etc.). Il est également soumis aux conditions de stockage disponibles ainsi que les échéances de consommations. Le magasinier supervise cette opération et donnes les directives nécessaires pour éviter le dommage des aliments.

1.4.3. Gestion de la consommation :

- Les types et les quantités des aliments consommés sont fonction des menus préparés effectivement et du nombre de plats servis. Le magasinier doit enregistrer l'inventaire de consommation pour chaque repas, c'est-à-dire dans la journée de leur consommation (au quotidien sauf les jours de week-end et férié).Ce suivi de la consommation doit être nécessairement accompagné d'une mise à jour de l'état du stock au quotidien.

- En finalité, le magasinier doit calculer le prix moyen par repas selon les aliments utilisés réellement pour assurer le suivi des coûts de consommation. .
- Dans le cas des dommages de produits, le magasinier établit l'estimation des pertes en termes des produits, leurs quantités, et les couts correspondants.

1.5. Tâches de l'économiste :

L'économe de l'établissement est l'expert-comptable du réfectoire. Il se charge du suivi du budget accordé au réfectoire de l'établissement. Il assure le paiement des fournisseurs conventionnés après s'être assuré de l'intégrité financière du dossier. Pour cela, il compare les bons de livraisons et factures établies par les fournisseurs aux bons de commandes et aux réceptions du réfectoire. L'économe s'assure que ces données doivent respecter les exigences formelles prévues par le cahier de charge en terme des codes, désignations, quantités et montants des aliments en question.

1.6. Tâches du Directeur de l'établissement :

Le directeur certifie le réfectoire. Il préside le conseil de gestion du réfectoire et est membre du conseil juridique pour les restaurants scolaires municipaux. Le directeur est premier responsable de la signature des bons de commandes. Ce pendant, il peut déléguer cette responsabilité au magasinier.

1.7. Tâches du cuisinier :

Le cuisinier doit identifier les aliments nécessaires à la préparation de chaque repas. Il est chargé d'en informer le magasinier via une fiche se sortie des articles en stock. Cette fiche doit renseigner le nom et signature du cuisinier, la date de demande de sortie des aliments ainsi que leur code, désignation et quantité.

Chapitre 2

Présentation d'UML et du processus unifié (UP)

2.1 Introduction

Dans ce deuxième chapitre, nous allons présenter le langage de modélisation unifié UML et le processus unifié UP que nous adoptons pour l'analyse et la conception d'un système pour gérer le réfectoire d'un établissement scolaire.

2.2 Langage de modélisation

2.2.1 Qu'est ce que un langage de modélisation ?

Un langage de modélisation est un langage artificiel utilisé pour représenter l'information, dans une structure qui est définie par un ensemble cohérent de règles. Ces règles sont utilisées pour interpréter de la signification des composants dans la structure.

2.2.2 Forme des langages de modélisation

Un langage de modélisation peut être graphique ou textuel.

Les langages de modélisation graphiques utilisent des techniques de diagrammes. Ces diagrammes sont composés de symboles associés à des noms et qui représentent les concepts utilisés dans la modélisation. Ces symboles sont connectés par des lignes qui représentent les relations entre concepts. Diverses autres annotations graphiques permettent de représenter les contraintes. EXPRESS-G est un exemple de langage de modélisation graphique.

Les langages de modélisation textuels utilisent typiquement des mots-clés standardisés accompagnés de paramètres pour rendre l'expression interprétable par les ordinateurs, EXPRESS (ISO 10303-11) est un exemple de langage de modélisation textuel.

2.2.3 Domaines Fonctionnels

Les langages de modélisation, qu'ils soient textuels ou graphiques, sont utilisés dans des disciplines variées :

- Gestion de l'information.
- Modélisation de processus d'affaires (business processus modeling).

- Génie logiciel est une science de génie industriel qui étudie les méthodes de travail et les bonnes pratiques des ingénieurs qui développent des logiciels.
- Ingénierie des systèmes : est une approche scientifique interdisciplinaire de formation récente, dont le but est de formaliser et d'appréhender la conception de systèmes complexes avec succès.

2.3 Exemples de langage de modélisation

- EXPRESS -G et EXPRESS (ISO 10303-11) sont un standard international de langage de modélisation des données dont le champ d'application est général.
- SYSML (SYStems Modeling Langage) est un langage de modélisation spécifique au domaine de l'ingénierie système.
- BPMN (Business Process Modeling Notation) est une norme industrielle de plus en plus importante pour la représentation graphique des processus d'affaires.
- ESL (Energy Systems language) a été conçu comme un outil conceptuel pour cartographier les flux d'énergie, des matériaux et de l'argent dans un système complexe. Il s'agit d'un langage de modélisation graphique, tout comme le langage de modélisation unifié (UML) en génie logiciel ou SysML en ingénierie des systèmes.
- UML (Unified Modeling Language) est une notation graphique conçue pour représenter, spécifier, construire et documenter les systèmes logiciels.

2.3.1 langage EXPRESS (ISO 10303 -11)

Express est un langage informatique servant à spécifier les données formellement. Il a fait l'objet d'une normalisation (ISO 10303-11).

Express permet de définir une représentation non ambiguë des données, interprétable pour un système informatique ce qui permet de créer directement et automatiquement un grand nombre d'éléments à partir d'un modèle Express.

En fait, Express est un langage de modélisation qui se base sur une approche objet comme UML et dont le but est seulement de spécifier une base de données et non pas de modéliser un système. D'autre part, Express propose également une réponse pour savoir où séparer bases de données objet et application objet :

- les contraintes d'intégrité, logiques et fonctionnelles, font partie de la base de données
- toutes les autres méthodes font partie de l'application et ne sont pas représentable en Express

Un modèle Express peut être écrit sous forme graphique ou sous forme textuelle. La forme graphique dite « Express-G » facilite la communication entre différents intervenant, toutefois un schéma « Express-G » ne permet pas de reprendre tous les éléments du modèle textuelle.

Un exemple de modèle en langage Express :

```
SCHEMA établissement;

ENTITY PERSONNE
ABSTRACT SUPERTYPE OF (ONEOF (ETUDIANT, SALARIE));
END_ENTITY;

ENTITY ETUDIANT
SUBTYPE OF (PERSONNE);
END_ENTITY;

ENTITY SALARIE
SUBTYPE OF (PERSONNE);
END_ENTITY;

END_SCHEMA;
```

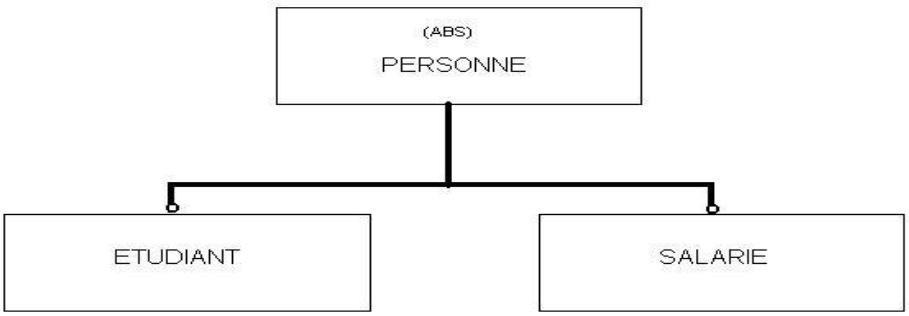


Figure 1 : représentation de établissement par diagramme EXPRESS

2.3.2 Langage UML (Unified Modeling Language)

Le langage UML est un langage de modélisation graphique initialement conçu pour représenter, spécifier, concevoir et documenter les artefacts de systèmes logiciels. Adopté par l'Object Management Group (OMG) en tant que standard, il est devenu une référence incontournable dans le domaine du génie logiciel. Sa richesse et sa puissance d'expression le rendent également éligible pour la modélisation de concepts et de processus « métier » (« business modeling ») et pour l'ingénierie de systèmes non logiciels. UML résulte de l'unification de techniques ayant fait leurs preuves pour l'analyse et la conception de grands logiciels et de systèmes complexes.

Dans la suite, nous nous intéressons d'avantage au langage de modélisation **UML.**

2.4 UML (Unified Modeling Language):

2.4.1 Présentation d'UML

UML, ou « langage de modélisation unifié » est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

UML est l'accomplissement de la fusion de langages de modélisation précédents objet : Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard défini par l'Object Management Group (OMG). La dernière version diffusée par l'OMG est UML 2.4.1 depuis août 2011.

2.4.2 Historique

UML est né en octobre 1994 chez Rational Software Corporation à l'initiative de G. Booch et de J. Rumbaugh. UML 1.1 a été standardisé par l'OMG (Object Management Group) le 17 novembre 1997 suite à la demande émanant de la collaboration de plusieurs entreprises (Hewlett-Packard, IBM, i-Logix, ICON Computing, Intelli Corp, MCI System house, Microsoft, ObjecTime, Oracle, Platinum Technology, Ptech, Rational Software Corporation, Reich Technologies, Softeam, Sterling Software, Taskon et Unisys).

2.4.3 Mise en œuvre d'une démarche a l'aide d'UML : les vues

UML propose différents modèles pour représenter les différents points de vue de la modélisation. il s'agit de :

• **la vue logique** (intégrité de conception) : perspective abstraite de la solution (classes, relations, machines états-transitions, etc.).

• **la vue des composants** (intégrité de gestion du code) : perspective physique de l'organisation du code (modules, composants, concepts du langage ou de l'environnement d'implémentation).

• **la vue des processus** (intégrité d'exécution) : perspective sur les activités concurrentes et parallèles (tâches et processus).

• **la vue de déploiement** (intégrité de performance) : répartition du système (logiciel) à travers un réseau.

• **la vue des cas d'utilisation** (intégrité de conception), qui guide et justifie les autres moyen rigoureux et systématique pour guider la modélisation (cas d'utilisation, scénarios, collaborations d'objets, machines à états).

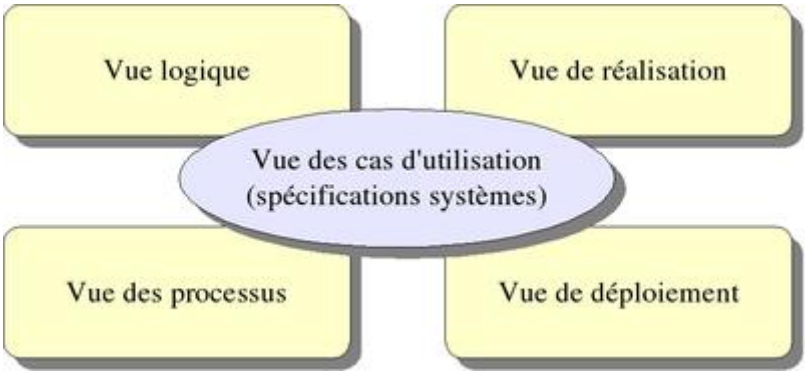


Figure 2 : Vues d'UML

2.4.4 Diagrammes d'UML

UML propose aujourd'hui 13 diagrammes t dépendant hiérarchiquement et se complètent, de façon à Permettre la modélisation d'un projet tout au long de son cycle de vie. Ils sont classés en 3 types :

a-Diagrammes structurels

b- Diagrammes comportementaux

c- Diagrammes d'interaction

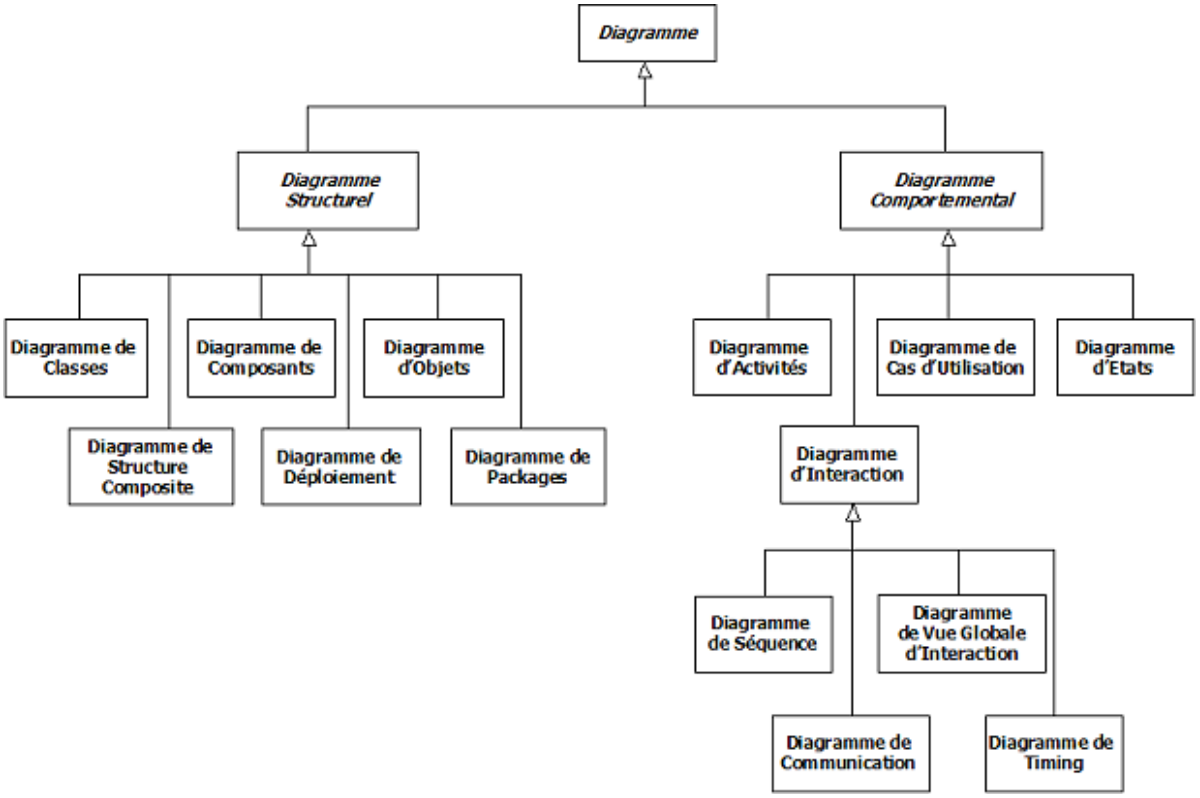


Figure 3 : la hiérarchie des diagrammes UML 2.0 sous forme d'un diagramme de classes

a. Diagrammes structurels

Ils sont également dits statiques. Il s'agit de 6 diagrammes dont les principales caractéristiques sont :

- **Un diagramme de classes** montre la structure statique des classes qui composent le système. Vous pouvez utiliser un diagramme de classes pour identifier le type des objets qui vont composer votre système, et définir les modalités selon lesquelles ils seront associés. Pour plus d'informations,
- **Un diagramme de structure** composite permet de définir plus en détails la structure interne de vos classes et les modalités selon lesquelles elles seront associées. Vous pouvez utiliser un diagramme de structure composite en particulier pour modéliser des formes de composition qui seraient particulièrement fastidieuses à modéliser dans un diagramme de classes. Pour plus d'informations,
- **Un diagramme d'objets** est similaire à un diagramme de classes, à un détail près qu'il montre des instances particulières des classes. Vous utilisez un diagramme d'objets pour représenter des relations entre les instances des classes.
- **Un diagramme de packages** montre la structure des packages qui constituent votre application, et les relations entre elles. Pour plus d'informations le diagramme de composants décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants .
- **diagramme de déploiement** est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un diagramme de déploiement sont principalement les nœuds, les composants, les associations et les artefacts. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.

b. Diagrammes comportementaux

- **diagramme de cas d'utilisation** est un diagramme UML qui fournit une représentation graphique des exigences de votre système, et vous aide à identifier la façon dont les utilisateurs interagissent avec ce dernier. Avec un diagramme de cas d'utilisation, vous disposez d'un aperçu instantané des fonctionnalités du système. Vous pouvez par la suite

ajouter plus de détails dans le diagramme si vous le souhaitez afin d'éclaircir certains points relatifs au comportement du système.

- **diagramme d'états-transitions** qui représente un cas d'utilisation particulier ou une fonctionnalité du système en utilisant les états par lesquels passe un classificateur et les transitions entre ces états.
- **diagramme d'activités** qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'actions ou d'activités effectuées et de transitions déclenchées par l'accomplissement de ces actions. Il permet également de représenter des branches conditionnelles.

c. Diagrammes d'interaction ou dynamique

- **Un diagramme de séquence** qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'interactions entre des objets, tout en se focalisant sur l'ordre chronolo**g**ique des messages échangés. Pour plus d'informations,
- **Un diagramme de communication** qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'interactions entre des objets, tout en se focalisant sur la structur**e** de l'objet. Pour plus d'informations
- **Un diagramme d'interactions** qui fournit une représentation de haut niveau des interactions se produisant dans votre système. Pour plus d'informations
- **Diagramme de temps** permet de décrire les variations d'une donnée au cours du temps.

2.4.5 Les éléments de modélisation d'UML

Nous abordons dans ce qui suit les éléments de modélisation d'UML avec plus de détails.

a. Les éléments structurels

- La classe

C'est un type d'éléments de la modélisation objet qui partagent les mêmes propriétés (attributs, méthodes, relations, sémantique). Si les propriétés définies par la classe sont immuables et ne dépendent pas des éléments, on parle d'attributs ou de variables de classe

et de méthodes de classes. Inversement si leur contenu dépend de l'élément on parle d'attribut ou de variable d'instances et de méthodes. L'accès aux propriétés d'une classe par une autre est contrôlé. On dit que les propriétés d'une classe ont une visibilité.

UML représente les classes par un rectangle divisé en 3 parties comme indiquée sur la figure suivante et dont l'entête porte le nom de la classe.

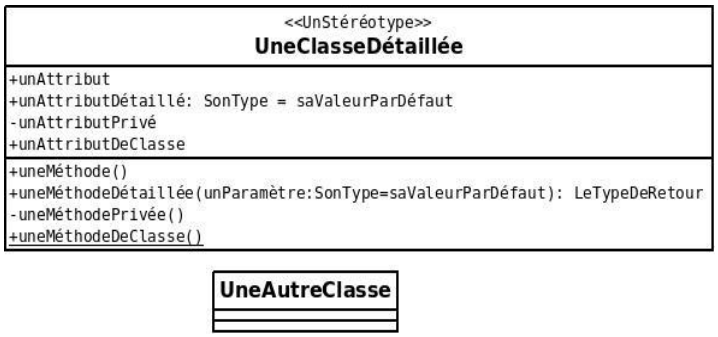


Figure 4 : Différentes représentations des classes en UML.

- **L'interface**

C'est un ensemble de méthodes qui définissent le comportement attendu d'une classe en laissant aux classes issues de cette interface de choisir l'implémentation.

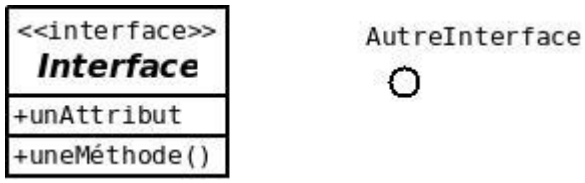


Figure 5 : Eventuelles représentations des interfaces

- **L'objet**

C'est un élément concret actif ou passif appartenant à une classe. Si la classe de l'objet est connue, on parle alors d'instance de cette classe. L'objet possède toutes les propriétés de sa classe et définie les valeurs des attributs d'instances.

Figure 6 : Représentations des objets

- **L'acteur**

C'est une classe externe au système agissant ou réagissant au système. Ainsi tous les intervenants sur le système sont des acteurs.

Figure 7 : Représentation des acteurs

- **Autres éléments structurels**

Figure 8 : Représentation des autres éléments structurels

b. Les éléments comportementaux

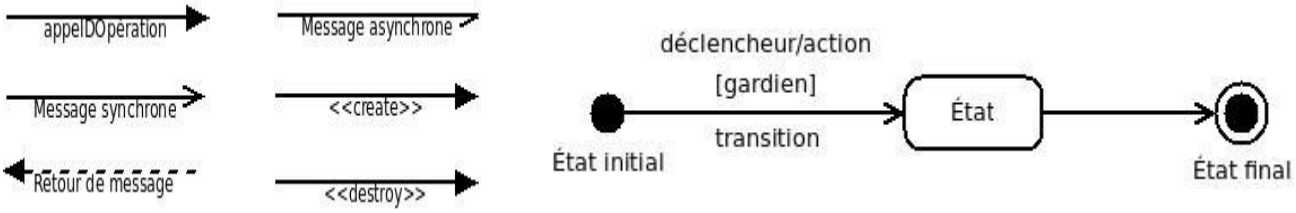


Figure 9 : Représentation des éléments comportementaux

c. Les relations

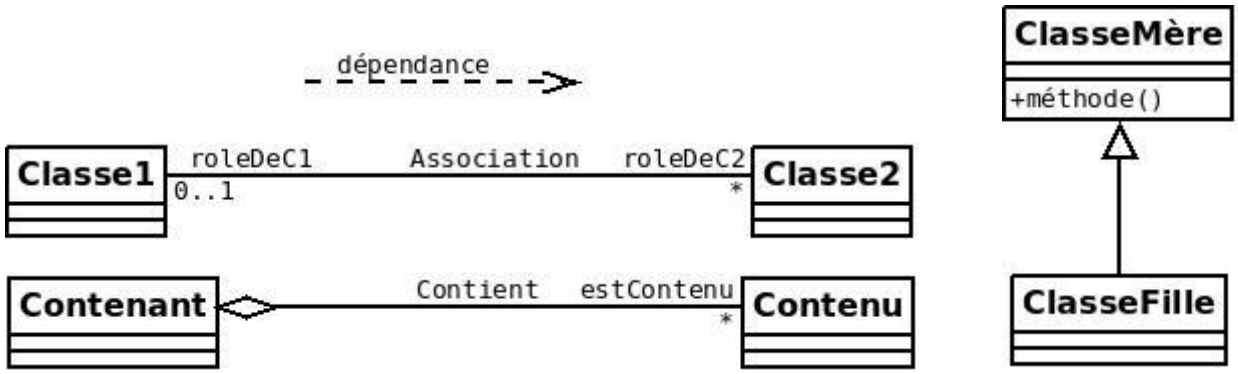


Figure **10** : Représentation des éléments modélisation de type relation

2.5 Le processus unifié UP

2.5.1 Définition/Présentation

Le processus unifié est un processus construit sur UML (Unified Modeling Language). en d'autres termes, il détermine les meilleures pratiques du développement objet Suivies pour la réalisation d'un système, à savoir ;

- Développer le logiciel de manière itérative
- Développements centrés sur l'architecture (et orientés sur les risques majeurs) avec le principe de la réutilisation des composants
- Focus sur la qualité (tester tôt et souvent, niveau unitaire, intégration, système) avec les principes de la validation et de l'intégration continue
- Modélisation graphique (UML, digrammes, grilles …) avec le principe de l'utilisation du visuel pour communiquer
- Gestion des exigences (organisation, traçabilité, évolution des exigences)
- Contrôle du changement (anticipation, gestion des demandes) avec le principe du réajustement (feedback, adaptation)

L'application de ces principes ne peut assurer la réussite du projet à 100 % mais elle y contribuera fortement.

2.5.2 Les quatre 'P' du processus unifié

Le processus de développement a les quatre P sont :

- Les **personnes** : architectes, développeurs, testeurs, utilisateurs, clients est autres intervenant.
- Le **projet** : élément d'organisation à travers lequel est géré le développement du logiciel.
- Le **produit** : ensemble d'artefacts créés au cours du cycle de vie du projet, tels que les modèles, les codes sources, les exécutables et la documentation.
- Le **processus** : fournit une définition de l'ensemble des activités requises pour transformer en produit les besoins exprimés par les utilisateurs.
- Les **outils** : environnements logiciels permettant d'automatiser les activités définies par le processus.

2.5.3 PU piloté par les cas d'utilisation :

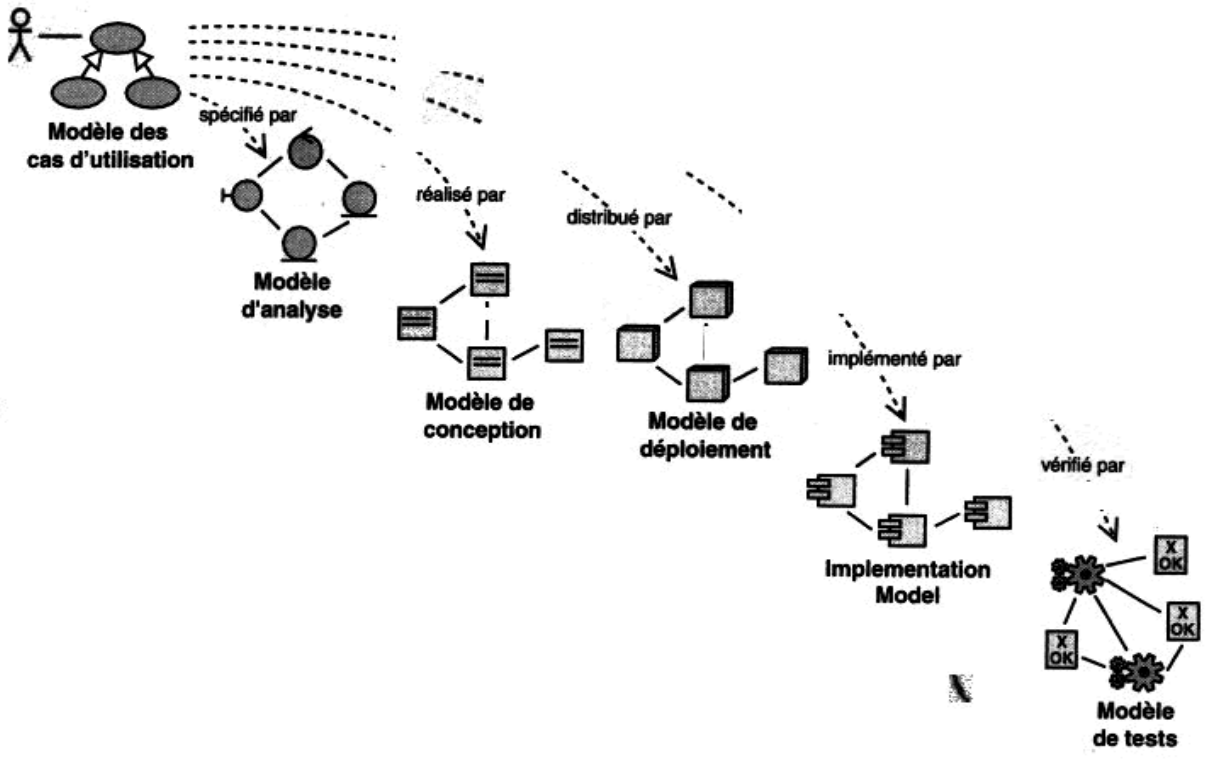


Figure **11** : UP est piloté par les cas d'utilisation

L'objectif d'un système logiciel est de rendre service à ses utilisateurs. Pour réussir sa mise au point, il importe par conséquent, de bien comprendre les objectifs et les besoins de ses futurs utilisateurs du système.

Un cas d'utilisation est une fonctionnalité du système produisant un résultat satisfaisant pour l'utilisateur. Les cas d'utilisation saisissent les besoins fonctionnels et leur ensemble forme le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.

2.5.4 PU centré sur l'architecture d'utilisation

Le rôle de l'architecture logicielle est comparable à celui que joue l'architecte dans la construction d'un bâtiment. Le bâtiment est envisagé de différents points de vue: structure, services, conduite de chauffage, plomberie, etc. Ce regard multiple dessine une image complète du bâtiment avant le début de la construction. De la même façon, l'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit.

2.5.5 PU itératif et incrémental

Le développement d'un produit logiciel destiné à la commercialisation est une vaste opération qui peut s'étendre sur plusieurs mois, voire sur une année ou plus. Il n'est pas inutile de découper le travail en plusieurs parties qui sont autant de mini-projets (Concept systémique de système et sous-systèmes).Chacun d'eux représente une itération qui donne lieu à un incrément. Les itérations désignent des étapes de l'enchaînement d'activités, tandis que les incréments correspondent à des stades de développement du produit.

La figure suivante détaille les différentes phases d'élaboration d'un système selon le PU.

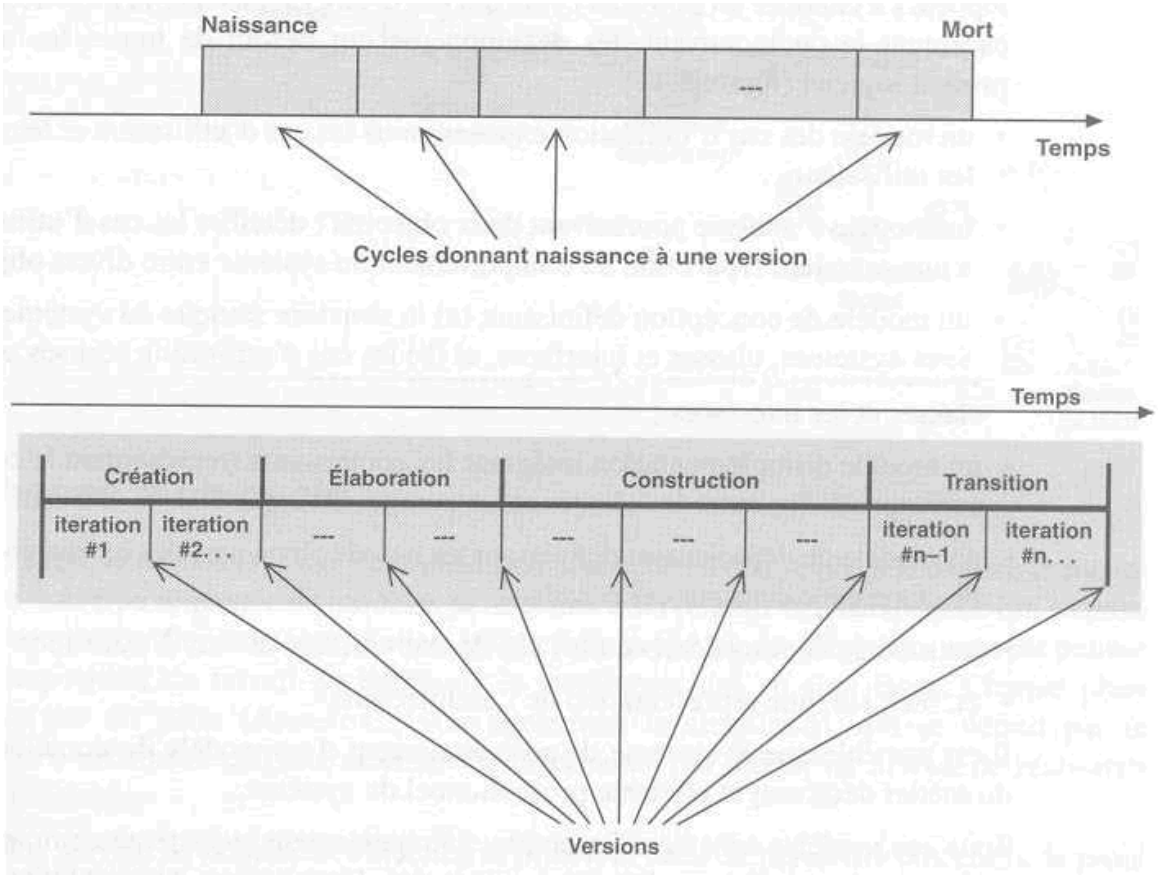


Figure 12 : La vie du Processus unifié

Ce procédé itératif présente les avantages suivants :

- permet de limiter les coûts : un risque ou problème est confiné à une itération
- permet de limiter les risques de retard : identification des risques et résolution des problèmes dès les premiers stades de développement
- se traduit par une accélération du rythme de développement : grâce à des objectifs clairs à court terme, il facilite l'adaptation à l'évolution des besoins

Chapitre 3

Langage de programmation (Delphi)

3.1 INTRODUCTION

Dans ce troisième chapitre, nous présentons d'abord les caractéristiques de Visual Basic et Delphi. Ensuite, nous comparons Delphi avec d'autres environnements de programmation. Nous expliquons en finalité, les principes de fonctionnement de Delphi.

3.2 Langage de programmation

Un langage de programmation est un code de communication, permettant à un être humain de dialoguer avec une machine en lui soumettant des instructions et en analysant les données matérielles fournies par le système, généralement un ordinateur. Le langage permet à la personne qui rédige un programme, de faire abstraction de certains mécanismes internes.

L'activité de rédaction du code source d'un programme est nommée programmation. Elle consiste en la mise en œuvre de techniques d'écriture et de résolution d'algorithmes informatiques, lesquelles sont fondées sur les mathématiques.

Les langages de programmation permettent de définir les ensembles d'instructions effectuées par l'ordinateur lors de l'exécution d'un programme. Il existe des milliers de langages de programmation, la plupart d'entre eux étant réservés à des domaines spécialisés. Ils font l'objet de recherches constantes dans les universités et dans l'industrie.

3.2.1 Classement du langage de programmation :

Les langages de programmation peuvent être classés de nombreuses manières :

- généraliste/spécialisé.
- haut niveau/bas niveau.
- interprété/compilé.
- avec/sans gestion de mémoire automatisée.
- avec/sans système de gestion d'exceptions.
- à typage fort/typage faible.

- à typage statique/typage dynamique.
- à syntaxe fixe/extensible.
- non objet/orienté objet/purement objet.
- impératif/fonctionnel/déclaratif.
- fonctionnel pur/impur.

3.3 Visual Basic (VB)

Créé par Microsoft pour son modèle de programmation COM(component Object model) ,Visual Basic est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO(data Access Object), ADO(activeX data Objects) et RDO, ainsi que la création de contrôles ou objets ActiveX. Les langages de script tels que Visual Basic for Applications et VB Script sont syntaxiquement proches de Visual Basic, mais s'utilisent et se comportent de façon sensiblement différente.

Un programme en VB peut être développé en utilisant les composants fournis avec Visual Basic lui-même. Les programmes écrits en Visual Basic peuvent aussi utiliser l'API Windows, ceci nécessitant la déclaration dans le programme des fonctions externes.

Visual Basic est un des langages les plus utilisés pour l'écriture d'applications commerciales. Il est également été très utilisé dans le monde de l'ingénierie et de la recherche appliquée en raison de sa capacité à permettre des développements très rapides et très efficients permettant ainsi aux scientifiques de se consacrer d'avantage à l'algorithmique et moins aux aspects formels du codage.

3.3.1 Historique :

Bill Gates y était particulièrement attaché, probablement parce que son premier succès avait été un programme écrit en Basic pour l'altair, premier ordinateur grand public. Depuis les bouleversements introduits dans ce langage en 1998 par Microsoft, ce segment d'utilisateurs chevronnés mais non spécifiquement programmeur a du se ré-orienter vers des plateformes tels que MATLAB, sans retrouver toute l'efficacité et la souplesse de VB6. Le défaut

souvent reproché à VB est justement sa facilité de mise en œuvre : un débutant VB pourra rapidement faire un programme opérationnel mais souvent tellement mal fait (sans analyse, structures ni règles, sans même la moindre expérience en programmation parfois...) qu'il sera difficilement maintenable par la suite. De même, VB étant utilisable à la fois en mode interprété et en mode compilé, l'analyse du comportement des variables au sein d'un algorithme complexe est considérablement facilité et permet des cycles de développement de quelques heures seulement, là où du code écrit (par exemple) en C++ requerrait des semaines de travail.

Dans une étude conduite en 2005, 62 pour cent des développeurs déclaraient utiliser l'une ou l'autre forme de Visual Basic. Actuellement, les langages les plus utilisés dans le domaine commercial sont Visual Basic, C++, C#, Java.

La dernière mise à jour de Visual Basic est la version 6.0, sortie en 1998. Le support étendu Microsoft a pris fin en 2008. À partir de la version 7, le Visual Basic subit des changements substantiels le rapprochant de la plate-forme « dot Net », et qui amènent Microsoft à le commercialiser sous le nom de Visual Basic .NET.

3.3.2 Fonctionnalités du langage

Visual permet de créer des applications graphiques de façon simple, mais également de créer des applications véritablement complexes. Programmer en VB est un mélange de plusieurs tâches, comme disposer visuellement les composants et contrôles sur les formulaires, définir les propriétés et les actions associées à ces composants, et enfin ajouter du code pour ajouter des fonctionnalités. Comme les attributs et les actions reçoivent des valeurs par défaut, il est possible de créer un programme simple sans que le programmeur ait à écrire de nombreuses lignes de code.

3.3.3 Caractéristiques de Visual Basic

Visual Basic possède quelques caractéristiques inhabituelles :

- Rétrocompatibilité avec les (anciennes) versions du BASIC de Microsoft (QBASIC/Quick BASIC) permettant le portage de vieux programmes.

- Possibilité d'utiliser à la fois des méthodes procédurale à l'ancienne (via des branchements avec des labels. Ex.: <label>: GOTO <label> ; et des sous-procédures du type GOSUB <label> ... RETURN), et à la fois des techniques plus modernes comme la programmation orienté objet (avec des modules de classe, ...) le rendant ainsi très versatile.
- Optionalité d'un grand nombre de déclarations (typage, référencement, portées, ...) ainsi qu'une syntaxe extrêmement souple (espaces facultatifs).
- Les opérateurs bit à bit et les opérateurs logiques sont les mêmes. Ce n'est en revanche pas le cas dans tous les langages dérivés de C (tels que Java et Perl) qui disposent d'opérateurs différenciés pour les opérations logiques et les opérations bit à bit. Ceci est également une caractéristique traditionnelle du langage BASIC.
- Forte intégration avec le système d'exploitation Windows ainsi qu'avec le modèle COM.

3.4 Delphi

3.4.1 Présentation

Delphi est un environnement de programmation permettant de développer des applications pour Windows. Il incarne la suite logique de la famille Turbo Pascal avec ses nombreuses versions. Delphi est un outil moderne, qui fait appel à une conception visuelle des applications, à la programmation objet. De plus, il prend en charge le maintien automatique d'une partie du code source.

3.4.2 Les caractéristiques de Delphi

- Programmation objet.
- Outils visuels bidirectionnels.
- Compilateur produisant du code natif.
- Traitement complet des exceptions.
- Possibilité de créer des exécutables et des DLL.
- Bibliothèque de composants extensible.
- Débogueur graphique intégré.
- Support de toutes les API de Windows: OLE2, DDE, VBX, OCX, ...

3.4.3 Delphi et les autres

La plupart des applications Windows sont encore écrites en C ou C++ pour des raisons essentiellement historiques. Microsoft pousse bien entendu le C++ du fait que l'environnement Windows lui-même conçu sous forme d'objets.

A la naissance de Windows, Microsoft donnait également la possibilité d'écrire des applications Windows en Pascal, mais probablement aucune application réelle et commerciale n'a jamais été écrite dans ce langage. Le premier changement majeur dans le développement sous Windows fut l'apparition de Visual Basic:

- Facilitée de programmation qui incitent les débutants à produire des codes imparfaits.
- Pénalisé par un passé lourd de catastrophes dues à la moitié de son nom : Basic
- Décevant par ses performances.

Visual Basic a ainsi détourné quelques programmeurs C, mais surtout permis à d'autres programmeurs d'entrer en scène.

Alors pourquoi Delphi? La question a un sens car, à première vue, en comparant Delphi et Visual Basic, on penserait plutôt à du plagia. En réalité il n'en est rien. Par sa conception, son environnement de développement et ses performances, Delphi fait de plus en plus d'adeptes.

C'est sa courte histoire qui tend à le prouver, au travers de quelques constatations:

- De l'avis quasi unanime Delphi surpasse largement Visual Basic, même et surtout si on compare sa dernière version 32 bits et le nouveau Visual Basic 6.

- SUN, créateur de Java et Hot Java, a fait appel à Borland pour créer des composants Delphi permettant de construire des applications Java sur Internet.

Et comparé au langage C ? Mis à part des questions de langage (donc de syntaxe) est-il vraiment plus performant qu'un outil comme Delphi ? Les compilateurs Turbo C++, Borland C++ produits par Borland ont toujours été parmi les plus performants du marché. Or il se trouve que Borland a utilisé le même noyau de compilateur pour son Delphi et pour son C++. Par exemple, le code généré est aussi optimisé pour l'un que pour l'autre. Partant de là, les performances doivent probablement être comparables.

Il y a bien d'autres environnements de développement à part les trois cités ci-dessus, mais leur faible diffusion les marginalise.

3.4.4 Principes de développement Delphi

Delphi fait évidemment partie de la famille de la programmation à interface graphique, comme ne peuvent que l'être les langages de développement modernes sous Windows. On ne peut plus se permettre d'attendre des années avant de découvrir (ou d'apprendre par cœur) que l'objet "barre de défilement verticale" possède telle ou telle propriété.

Les propriétés, entre autres, des objets doivent être immédiatement et toujours visibles au programmeur. Pour construire l'interface d'une application, ce dernier place des objets sur une fiche "fenêtre" et les personnalise en modifiant éventuellement leurs propriétés et/ou en leur attachant des instructions liées à des événements donnés.

De par le nombre de composant fournis avec Delphi, les possibilités en termes d'interface graphique sont très grandes. D'autres parts, les contrôles ActiveX, des composants actifs utilisables dans une application, permettent d'avoir accès à des fonctions avancées :

- Accès à des bases de données.
- Accès à des fonctionnalités réseau.
- Accès à des fonctions d'entrée-sortie.

3.4.5 Avantages du langage Delphi :

- Delphi offre une façon de développer des applications sous un environnement très riche qui est Windows.
- Delphi apporte une grande souplesse aux développeurs ;
- Delphi permet de réaliser des applications Microsoft Windows très efficace avec un minimum de codage manuel.
- Delphi offre un compilateur optimisé qui donnera une application rapide sans qu'il soit nécessaire de fournir plus d'efforts pour optimiser le programme.
- Delphi offre une bibliothèque de composants visuels.
- Delphi possède un utilitaire (Report) qui permet de créer la forme désirée de la feuille d'impression.
- Delphi permet de développer des applications client/serveur.
- Delphi permet de simplifier le développement des applications Windows car il inclut plusieurs experts et d'autres outils spécialisés accélérant le développement.
- Delphi génère un vrai fichier exécutable (.EXE), aucun autre fichier n'est nécessaire pour son exécution donc on obtient une application plus facile à distribuer et à maintenir**.**

3.4.6 Delphi et Windows

Delphi permet de créer et de manipuler tout objet de Windows. Tous les objets et une grande partie de l'API de Windows sont encapsulés dans des composants Delphi. Les messages Windows sont redirigés par Delphi vers les objets auxquels ils sont destinés. Dans tous les cas, il est toujours possible de faire appel directement à l'API de Windows.

3.4.7 L'interface de Delphi

La figure ci-dessous représente l'interface typique de Delphi. Elle est composée de

• la barre de menus (en haut),

• la barre d'icônes (à gauche sous la barre de menus),

• la palette de composants (à droite sous la barre de menus),

• le concepteur de fiche (au centre),

• l'éditeur de code (au centre sous le concepteur de fiche),

• l'inspecteur d'objets (à gauche).

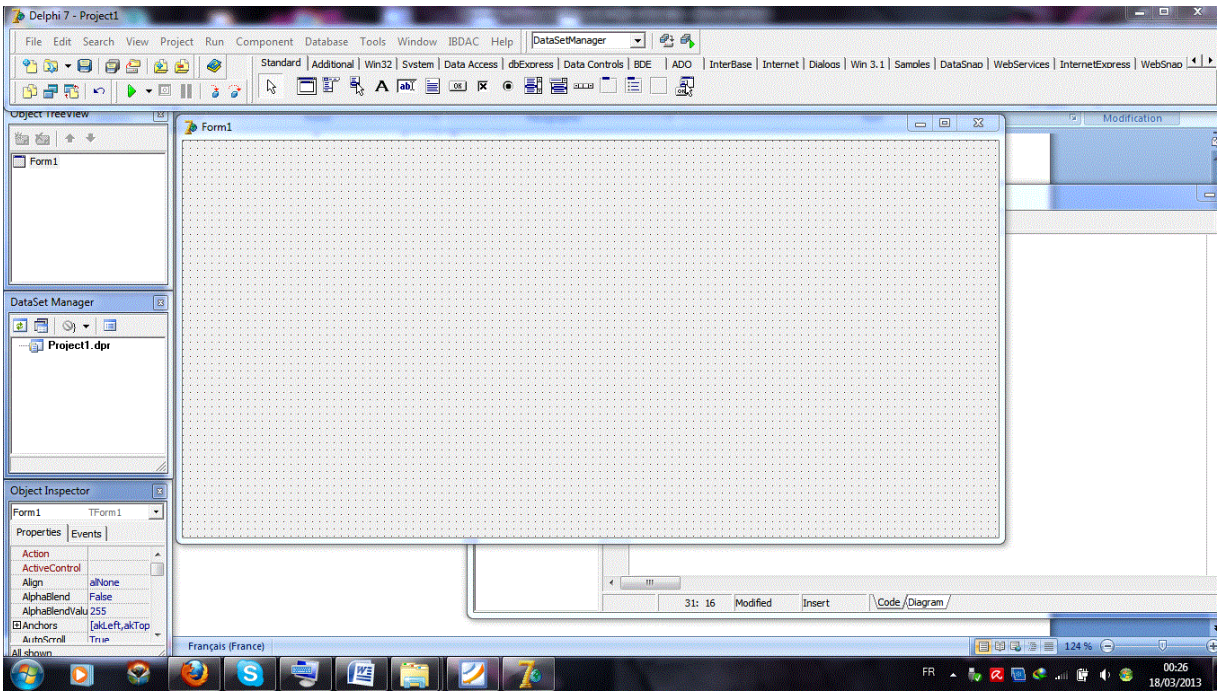


Figure 13: L'interface de Delphi

Chapitre 4

ETUDE DE CAS

4.1 Introduction

L'objectif de cette étape est de déterminer de façon détaillée et précise ce que le système devra faire, afin de répondre aux objectifs établis lors de l'étude de l'existant, tout en respectant les contraintes établies préalablement.

4.2 Identification des cas d'utilisations

Le tableau suivant englobe les différents CAS D'UTILISATION de ce système :

N°	Cas d'utilisation	Acteur
1	Authentification	Magasinier
2	Etablir bon commande	Magasinier
3	Réception des aliments	Magasinier
4	Consommation des aliments	Magasinier
5	Renseigner des articles	Magasinier
6	Gestion fournisseur	Magasinier
7	Gestion des familles	Magasinier
8	Etablir menu des repas	Magasinier
9	Renseigner les informations administratives	Magasinier

Tableau 4.1 : Les cas d'utilisations.

4.3 Description des cas d'utilisation

Nous allons maintenant donner une description de chaque cas d'utilisation.

4.3.1 Cas d'utilisation «authentification»

Titre :	Authentification
Finalité :	Ce cas autorise le magasinier le système par A accéder à tout les cas d'utilisation du système
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Connexion avec le Système est opérationnelle
Post-conditions :	Utilisateur authentifié Session utilisateur ouverte
Scenarios Nominal :	1- Le magasinier lance l'application. 2- Le système affiche le formulaire d'authentification. 3- Le magasinier saisis son login et son mot de passe. 4- Le système vérifie leur validité. 5- Le système lance le menu principal de l'application.
Scenarios alternatifs :	4. **Le mot de passe et/ou login incorrecte** 4.1. le système indique à l'utilisateur que Login ou mot de passe incorrecte (3 fois.) a- si 1ere et 2eme fois, le système ré- affiche le formulaire d'authentification. - le scenario nominal reprend au point 3 b- Si 3eme fois, auto-fermeture du système - le système quitte l'application **4. formulaire non valide** 4.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 3

4.3.2 Cas d'utilisation « Etablir bon commande »

Titre :	Etablir bon commande
Finalité :	Ce cas permet au magasinier d'établir bon des commandes
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié
Post-conditions :	Le bon des commandes imprimé
Scenarios Nominal :	1- le magasinier choisis établir bon commande. 2- Le système affiche un formulaire. 3- Le magasinier saisi les informations (date du bon, fournisseur, code produit, quantité) des aliments à commander. 4- Le système vérifié les informations du formulaire. 5- Le système affiche un message de succès ou échec. 6- Le système sauvegarde les informations de la commande. 7- le système demande à l'utilisateur s'il veut une impression du bon de commande. 8-le magasinier répond par oui 9- le système imprimer le bon de commande et valide l'état remis au fournisseur 10. le système revient à l'interface principale.
Scenarios alternatifs :	**4. formulaire non valide** 4.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 3 **8. imprimer le bon de commande** 8.1 le magasinier répond par non 8.2 le scénario reprend au point 10

4.3.3 Cas d'utilisation « Réception des aliments »

Titre :	Réception des aliments
Finalité :	Ce cas permet d'enregistrer les aliments reçus
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié.
Post-conditions :	- La nouvelle réception des aliments a été sauvegardée. Les articles reçus identique en code et quantité aux articles du bon de commande correspondant.
Scenarios Nominal :	1- le magasinier choisis l'ajout de nouvelle réception dans le menu du système. 2- Le système affiche au magasinier une interface de saisie formulaire. 3- Le magasinier saisi les informations des aliments reçus (le numéro du bon de commande, date et numéro de réception, numéro du bon de livraison, la quantité, etc). 4- Le système vérifie les informations du formulaire. 5- Le système génère un message de succès de l'opération 6- Le système sauvegarde les informations de la réception.
Scenarios alternatifs :	**4. formulaire non valide** 4.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 3

4.3.4 Cas d'utilisation «Consommation des aliments» :

Titre :	Consommation des aliments
Finalité :	Ce cas permet satisfaire les demandes de réfectoire.
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié. Le cuisinier demandés aliment.
Post-conditions :	Le système indique que l'aliment sortie avec succès.
Scenarios Nominal :	1- Le magasinier choisi d'ajouter une nouvelle sortie du stock. 2- Le système lui demande d'entre les nouvelles informations. 3- Le magasinier entre les nouvelles informations (désignation article, quantité sortie, date, …etc.). 4- Le système contrôle les informations saisies. 5- Le système indique que les nouvelles informations sont enregistrées.
Scenarios alternatifs :	**4. formulaire non valide** 4.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 3 4.2. Le système indique à l'utilisateur que quantité de sortie plus de Quantité de stock. 4.3. Le système indique à l'utilisateur que l'aliment en stock alerte

4.3.5 Cas d'utilisation « renseigner des articles » :

4.3.5.1 Cas Ajouter article :

Titre :	ajouter un article
Finalité :	Ce cas permet d'ajouter un article
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié.
Post-conditions :	Le système indique que l'ajout d'article termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi renseigner des articles dans le menu d'application. 2- Le système affiche le menu de renseigner des articles 3- Le magasinier choisi l'ajout d'un nouvel article dans le menu de renseigner des articles 4- Le système affiche le formulaire pour saisir les informations de nouveau article. 5- Le magasinier remplit ce formulaire (Désignation …etc.) 6- Le système contrôle les informations. 7- Le système lui indique que le nouvel article est enregistré.
Scenarios alternatifs :	**6. formulaire non valide** 6.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 4 6.2. Le système indique à l'utilisateur que l'article déjà dans la base de données.

4.3.5.2 Cas Modifier article :

Titre :	Modifier un article
Finalité :	Ce cas permet de modifier un article
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - L'article existe
Post-conditions :	Le système indique que la modification d'article termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi renseigner des articles dans le menu d'application. 2- Le système affiche le menu de renseigner des articles 3- Le magasinier choisi la modification d'un article dans le menu de renseigner des articles. 4- Le système affiché la recherche de l'article. 5- Le magasinier entre le nom de l'article qui veut recherche. 6- Si l'article existe le système affiché les informations de l'article. 7- Le magasinier modifier les informations d'article. 8- Le système contrôle les informations. 9- Le système lui indique que l'article est modifié.
Scenarios alternatifs :	**5. Si l'article n'existe pas** 5.1. le système affiche message 'l'article n'existe pas' - le scenario nominal reprend au point 4 **8. formulaire non valide** 8.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 7

4.3.5.3 Cas Supprimer article :

Titre :	Supprimer article
Finalité :	Ce cas permet de Supprimer article
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - L'article existe
Post-conditions :	Le système indique que la suppression d'article termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi renseigner des articles dans le menu d'application. 2- Le système affiche le menu de renseigner des articles 3- Le magasinier choisi la suppression d'un article dans le menu de renseigner des articles 4- Le système affiche la liste de tous les articles existants. 5- Le magasinier choisis un article pour supprimer. 6- Le système informe le magasinier s'il veut vraiment supprimer. 7- Le magasinier valide la suppression de l'article.
Scenarios alternatifs :	----------

4.3.6 Cas d'utilisation « Gestion fournisseur » :

4.3.6.1 Cas Ajouter fournisseur :

Titre :	ajouter un fournisseur
Finalité :	Ce cas permet d'ajouter un fournisseur
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié.
Post-conditions :	Le système indique que l'ajout de fournisseur termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion fournisseur dans le menu d'application. 2- Le système affiche le menu de gestion fournisseur 3- Le magasinier choisi l'ajout d'un nouveau fournisseur dans le menu gestion fournisseur. 4- Le système affiche le formulaire pour saisir les informations de nouveau fournisseur. 5- Le magasinier remplit ce formulaire (Nom, Prénom, adresse …etc.) 6- Le système contrôle les informations. 7- Le système lui indique que le nouveau fournisseur est enregistré.
Scenarios alternatifs :	**6. formulaire non valide** 6.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 5 6.2. Le système indique à l'utilisateur que le fournisseur déjà dans la base de données.

4.3.6.2 Cas Modifier fournisseur :

Titre :	Modifier fournisseur
Finalité :	Ce cas permet de modifier un fournisseur
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - Le fournisseur existe
Post-conditions :	Le système indique que la modification de fournisseur termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion fournisseur dans le menu d'application. 2- Le système affiche le menu de gestion fournisseur 3- Le magasinier choisi la modification d'un fournisseur dans le menu de gestion fournisseur 4- Le système affiché la recherche de fournisseur. 5- le magasinier entre le nom de fournisseur qui veut recherche. 6- Si le fournisseur existe le système affiché les informations de fournisseur. 7- Le magasinier modifier les informations de fournisseur. 8- Le système contrôle les informations. 9- Le système lui indique que fournisseur est modifié.
Scenarios alternatifs :	**5. Si le fournisseur n'existe pas** 5.1. le système affiche message 'le fournisseur n'existe pas' - le scenario nominal reprend au point 4 **8. formulaire non valide** 8.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 7

4.3.6.3 Cas Supprimer fournisseur :

Titre :	Supprimer fournisseur
Finalité :	Ce cas permet de Supprimer fournisseur
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - Le fournisseur existe
Post-conditions :	Le système indique que la suppression de fournisseur termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion fournisseur dans le menu d'application. 2- Le système affiche le menu de gestion fournisseur 3- Le magasinier choisi la suppression d'un fournisseur dans le menu de gestion fournisseur 4- Le système affiche la liste de tous les fournisseurs existants. 5- Le magasinier choisis un fournisseur pour supprimer. 6- Le système informe le magasinier s'il veut vraiment supprimer. 7- Le magasinier valide la suppression de fournisseur.
Scenarios alternatifs :	----------

4.3.7 Cas d'utilisation « Gestion des familles » :

4.3.7.1 Cas Ajouter famille :

Titre :	ajouter une famille
Finalité :	Ce cas permet d'ajouter une famille
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	Le magasinier s'est authentifié.
Post-conditions :	Le système indique que l'ajout de famille termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion des familles dans le menu d'application. 2- Le système affiche le menu de gestion des familles 3- Le magasinier choisi l'ajout d'une nouvelle famille dans le menu gestion des familles. 4- Le système affiche le formulaire pour saisir les informations de nouveau famille. 5- Le magasinier remplit ce formulaire (Désignation …etc.) 6- Le système contrôle les informations. 7- Le système lui indique que la nouvelle famille est enregistrée.
Scenarios alternatifs :	**6. formulaire non valide** 6.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 5 6.2. Le système indique à l'utilisateur que la famille déjà dans la base de données.

4.3.7.2 Cas Modifier famille :

Titre :	Modifier famille
Finalité :	Ce cas permet de modifier une famille
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - La famille existe
Post-conditions :	Le système indique que la modification de famille termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion des familles dans le menu d'application. 2- Le système affiche le menu de gestion des familles 3- Le magasinier choisi la modification d'une famille dans le menu de gestion des familles 4- Le système affiché la recherche de famille. 5- le magasinier entre le nom de famille qui doit recherche. 6- Si la famille existe le système affiché les informations de la famille. 7- Le magasinier modifier les informations de famille. 8- Le système contrôle les informations. 9- Le système lui indique que famille est modifié.
Scenarios alternatifs :	**5. Si la famille n'existe pas** 5.1. le système affiche message 'la famille n'existe pas' - le scenario nominal reprend au point 4 **8. formulaire non valide** 8.1. Le système indique à l'utilisateur que le formulaire n'est pas correctement remplis. - le scenario nominal reprend au point 7

4.3.7.3 Cas Supprimer famille :

Titre :	Supprimer une famille
Finalité :	Ce cas permet de Supprimer une famille
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - La famille existe
Post-conditions :	Le système indique que la suppression de famille termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi gestion des familles dans le menu d'application. 2- Le système affiche le menu de gestion des familles 3- Le magasinier choisi la suppression d'une famille dans le menu de gestion des familles 4- Le système affiche la liste de toutes les familles existantes. 5- Le magasinier choisis une famille pour supprimer. 6- Le système informe le magasinier s'il veut vraiment supprimer. 7- Le magasinier valide la suppression de famille.
Scenarios alternatifs :	----------

4.3.8 Cas d'utilisation « Etablir menu des repas »:

4.3.8.1 Cas Etablir une liste des repas :

Titre :	Etablir une liste des repas
Finalité :	cas permet d'établir une liste des repas
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié - Les composants de repas existent (Les articles)
Post-conditions :	Le système indique que l'ajout de liste des repas termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi Etablir menu des repas dans le menu d'application. 2- Le système affiche le formulaire pour ajouter ou modifier ou supprimer les repas 3- Le magasinier faire les opérations (ajouter, modifier ou supprimer) dans le menu des repas de cette semaine. 4- Le magasinier lancée l'impression de la liste des repas de cette semaine.
Scenarios alternatifs :	----------

4.3.9 Cas d'utilisation « Renseigner les informations administratives :

Titre :	Ajouter les informations administratives
Finalité :	cas permet d'ajouter les informations administratives
Acteurs Principaux :	Le magasinier
Acteurs Secondaire :	Aucun
Pré-condition :	- Le magasinier s'est authentifié
Post-conditions :	Le système indique que l'ajout des informations administratives termine avec succès.
Scenarios Nominal :	1- Le magasinier choisi Renseigner les informations administratives dans le menu d'application. 2- Le système affiche le formulaire pour saisir les informations d'administratives. 3- Le magasinier remplit ce formulaire (année …etc.) 4- Le système contrôle les informations. 5- Le système lui indique que les informations d'administratives est enregistrée.
Scenarios alternatifs :	----------

Diagramme de cas d'utilisateur

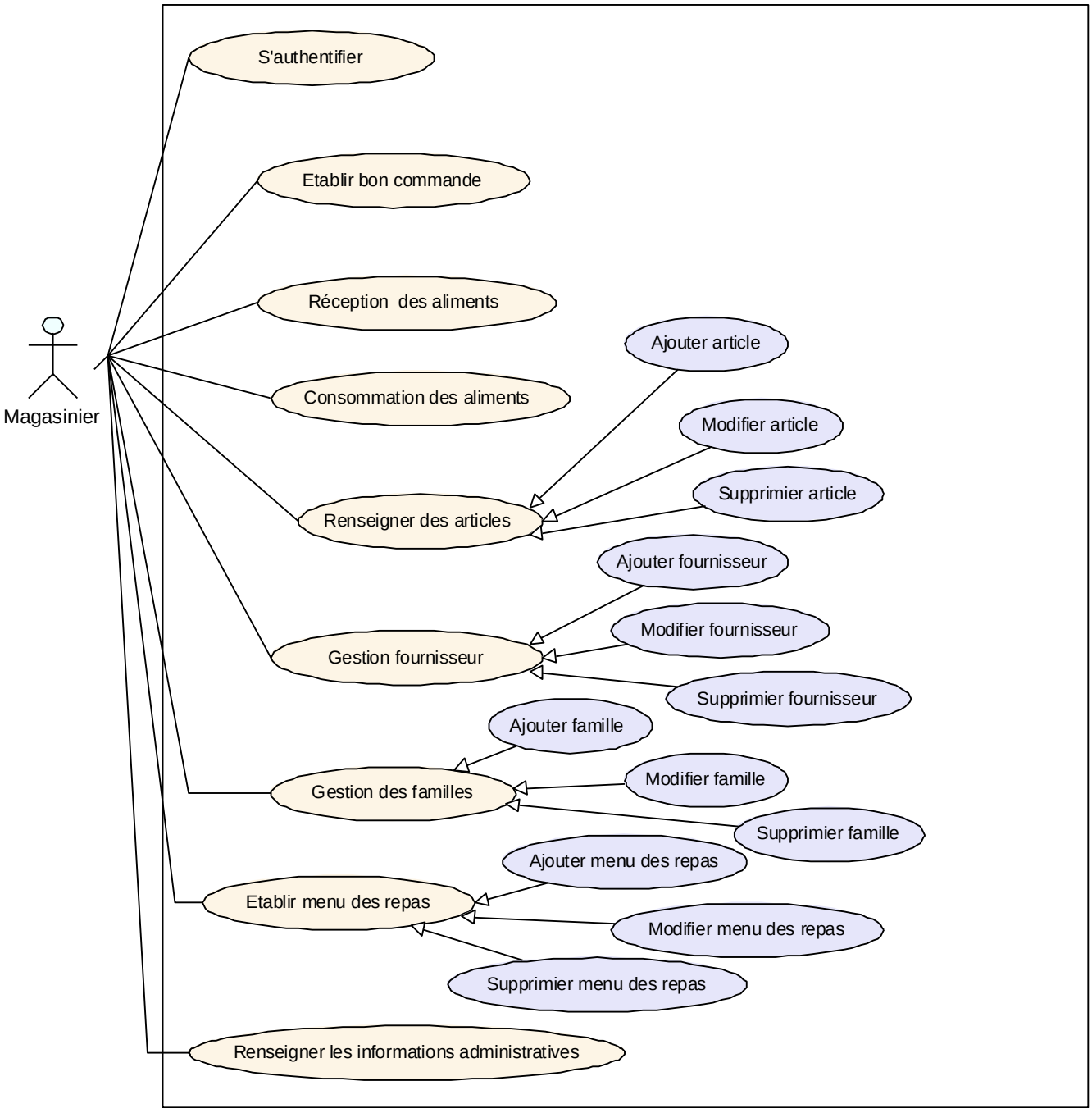


Figure14 : Diagramme des cas d'utilisation

Diagramme de séquence système

Cas S'authentifier

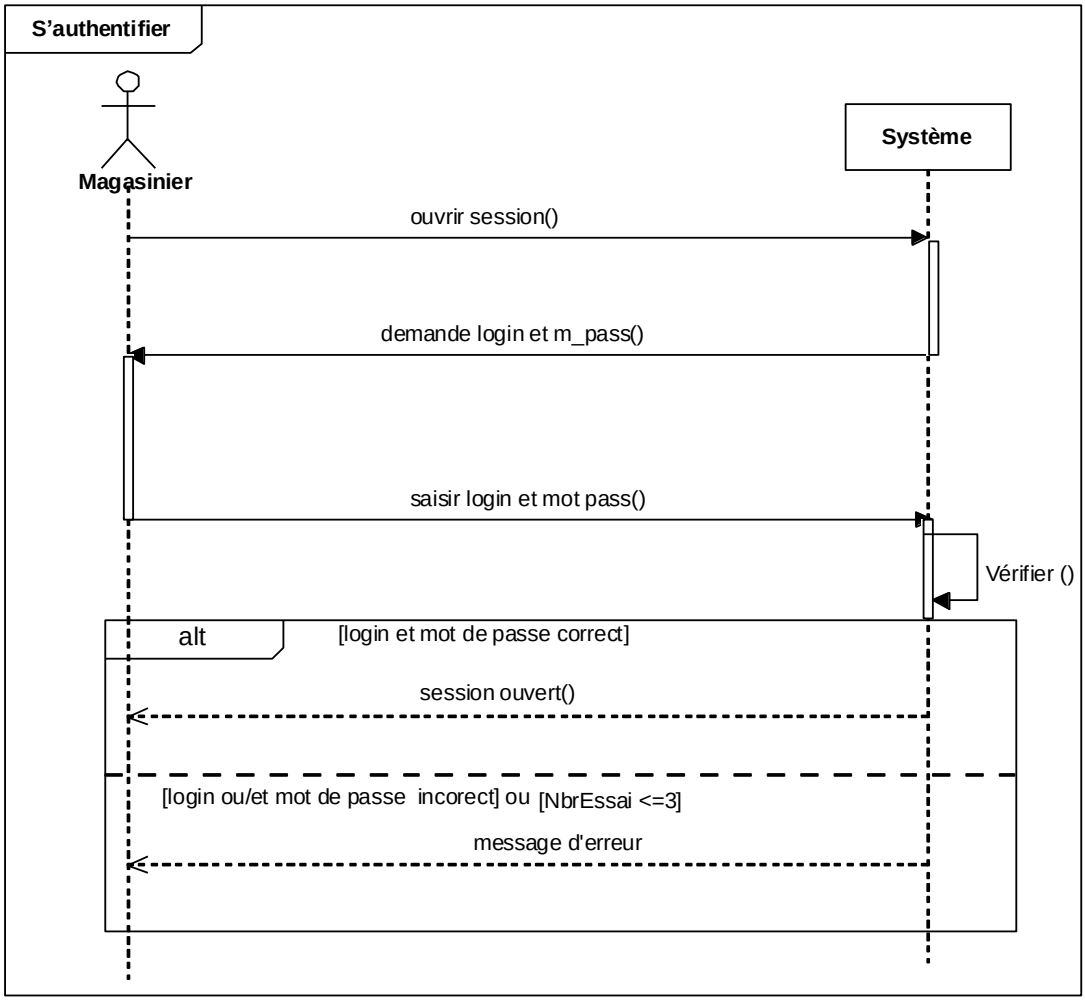


Figure15 : Diagramme de séquence système pour le cas d'utilisation Authentification

Cas Etablir bon commande

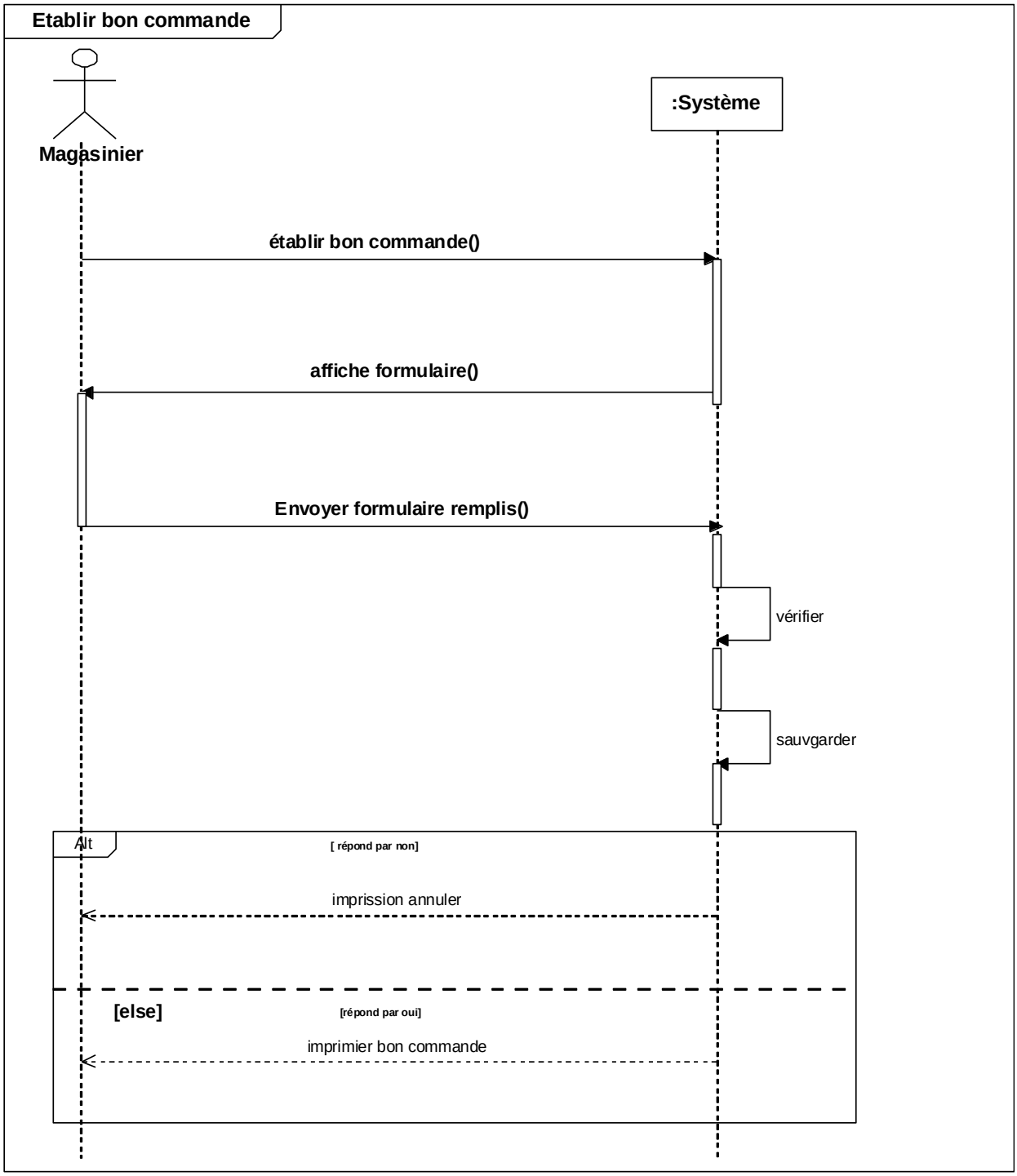


Figure16 : Diagramme de séquence système du cas d'utilisation Etablir bon commande

Réception des aliments

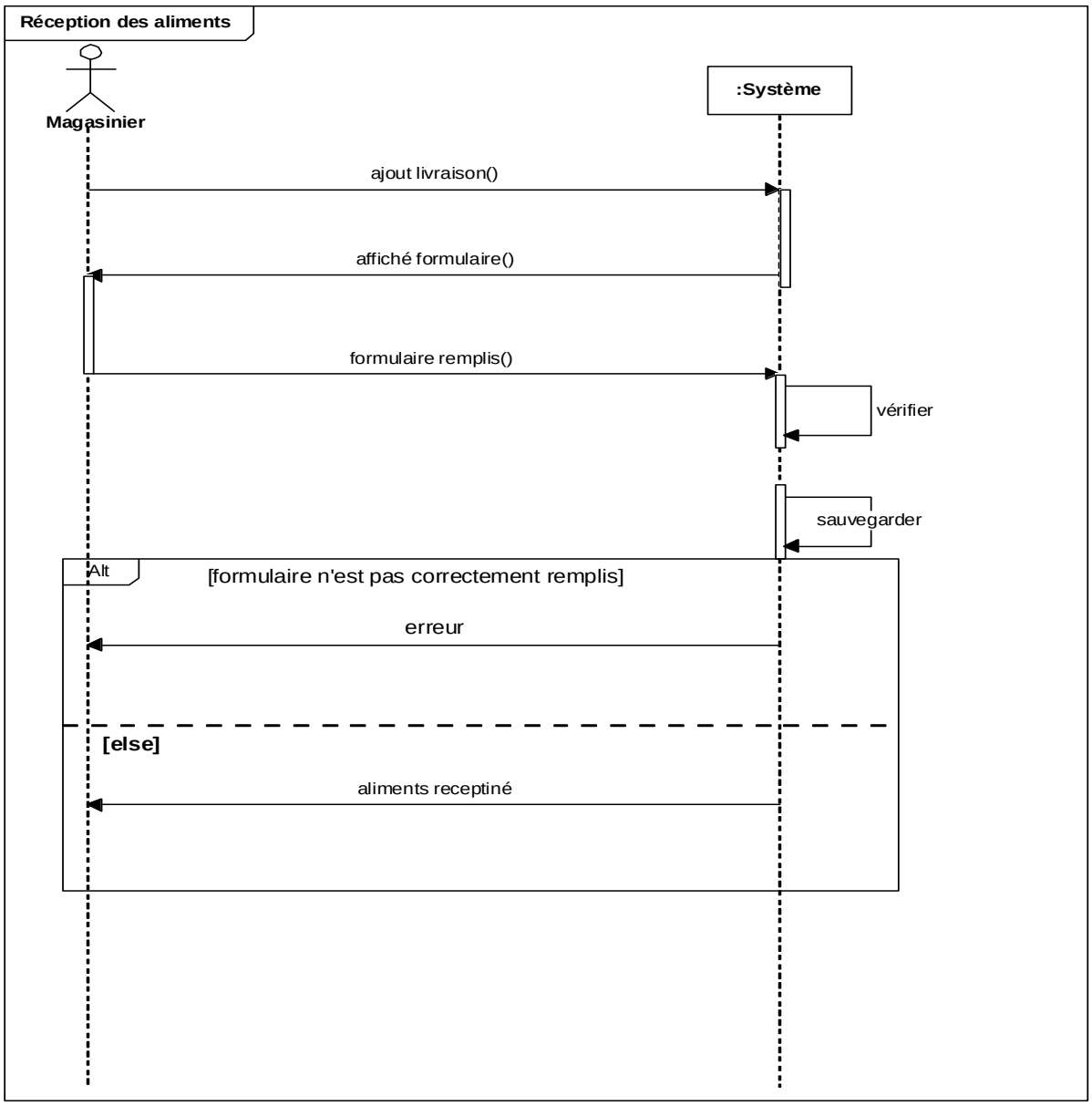


Figure17 : Diagramme de séquence système du cas d'utilisation Réception des articles.

Consommation des articles

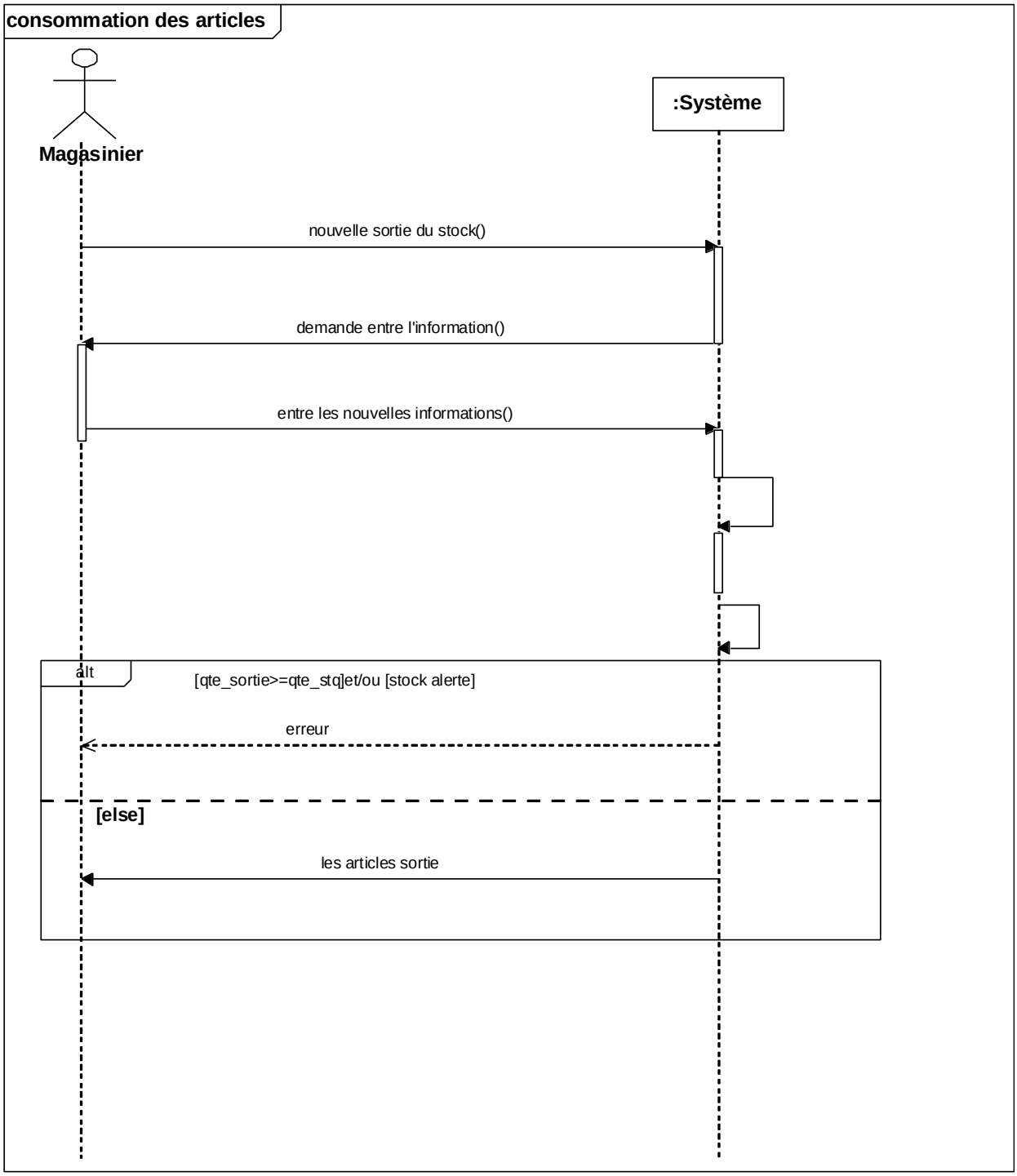


Figure18 : Diagramme de séquence système du cas d'utilisation consommation des aliments.

Renseigner des articles

Ajouter article

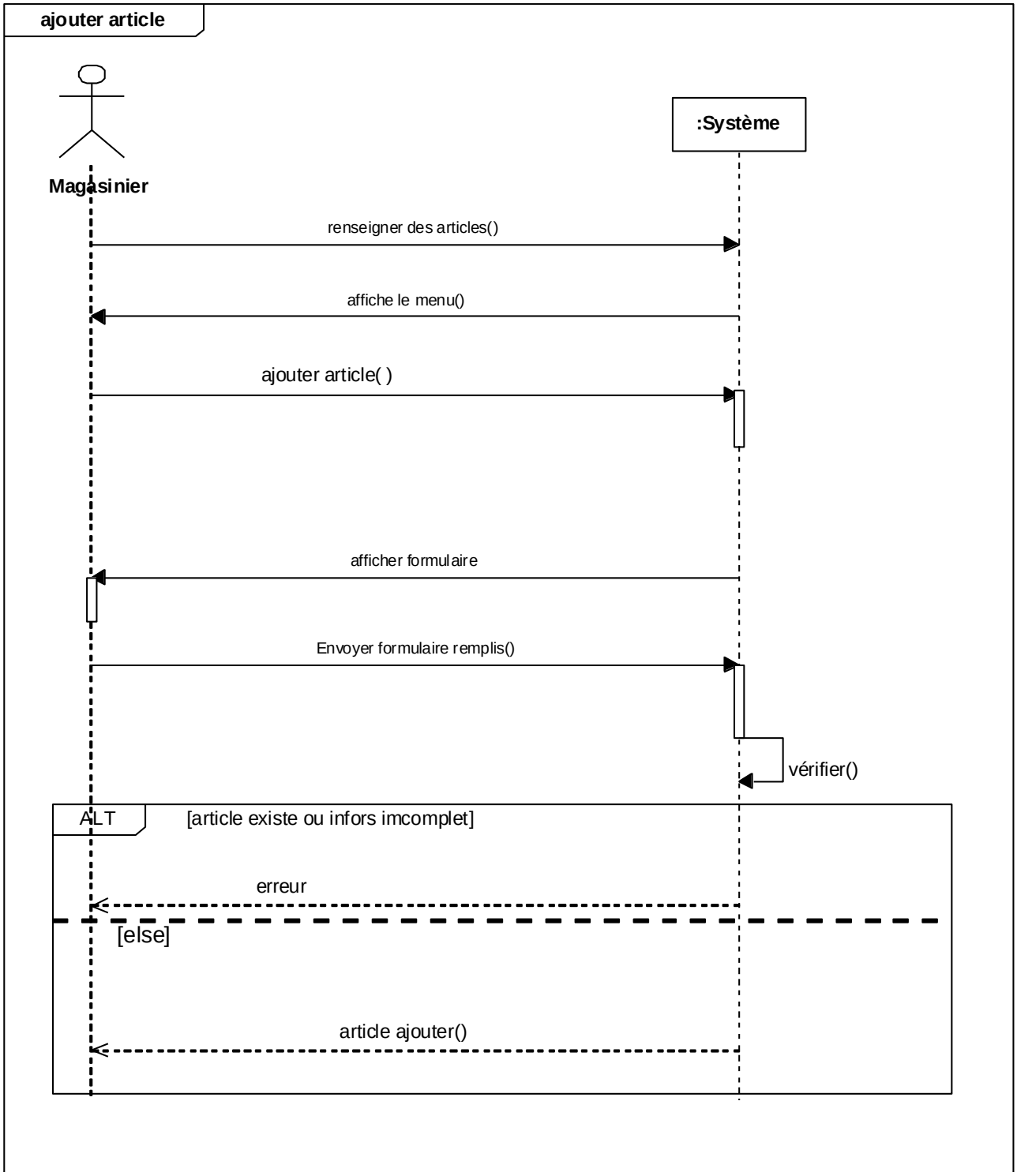


Figure19 : Diagramme de séquence système du cas d'utilisation renseigner des articles (ajouter articles).

Modifier article

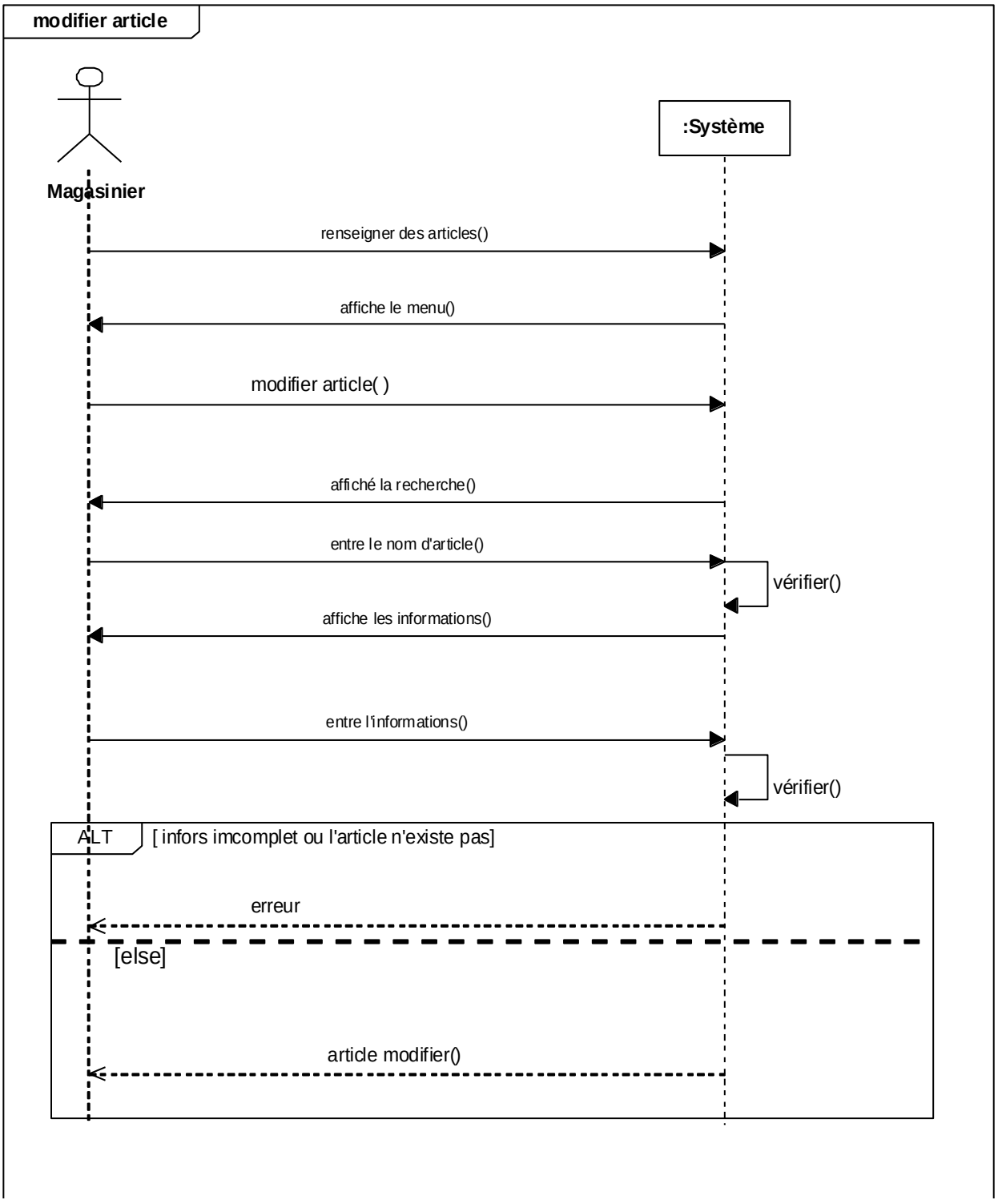


Figure20 : Diagramme de séquence système du cas d'utilisation renseigner des articles (modifier articles).

Supprimer article

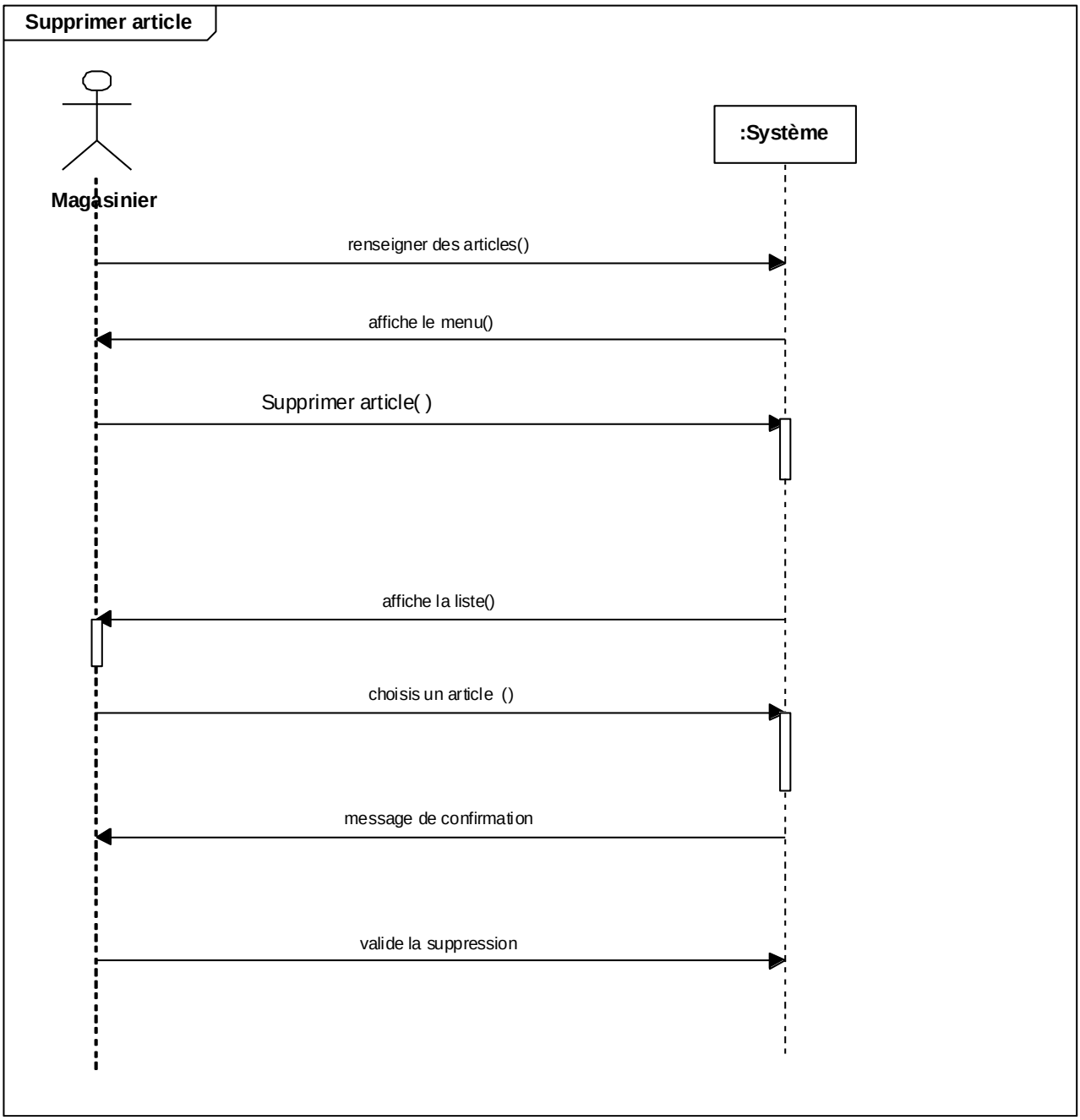


Figure21 : Diagramme de séquence système du cas d'utilisation renseigner des articles (supprimer articles).

Gestion fournisseur

Ajouter fournisseur

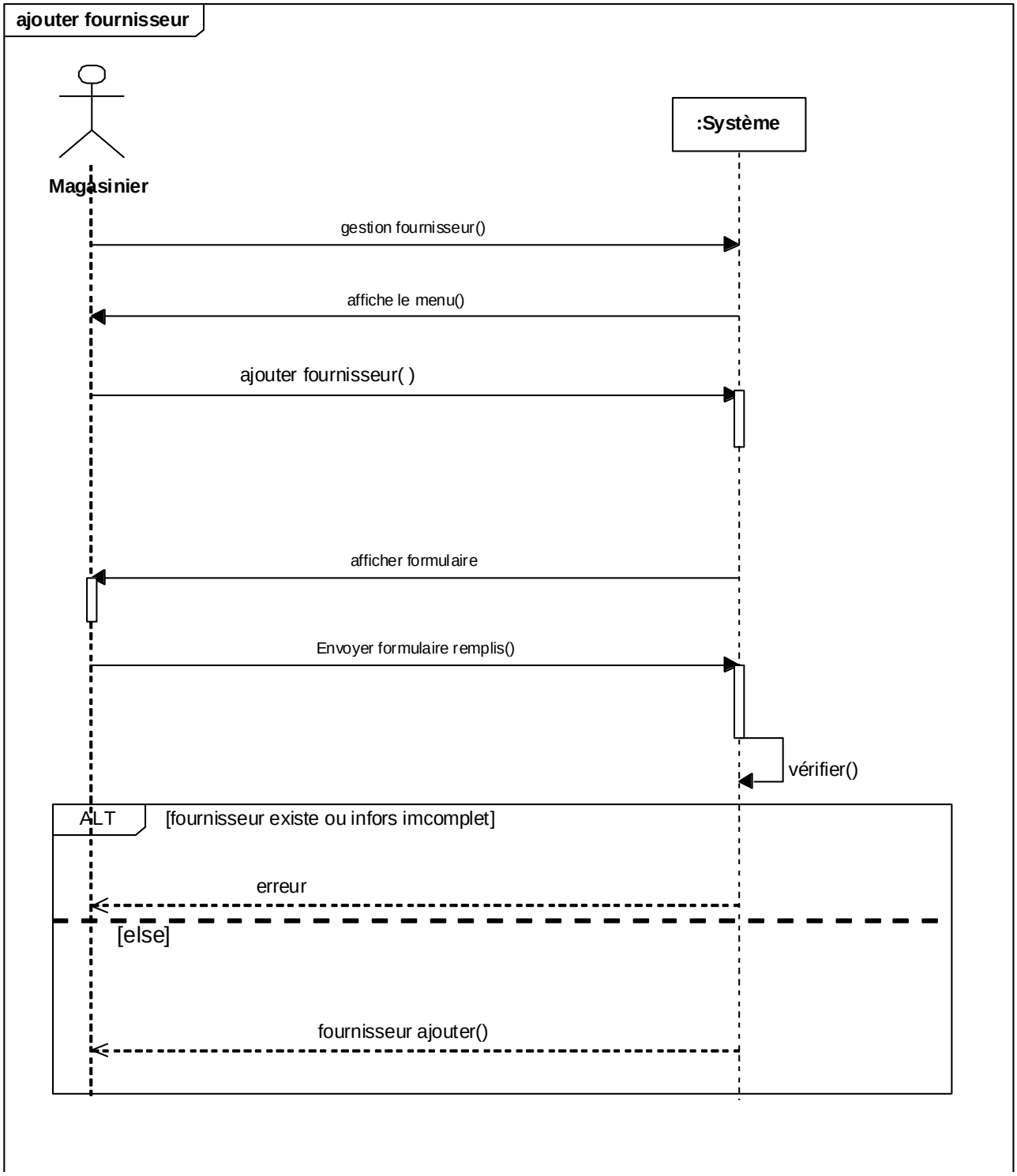


Figure22 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs (ajouter fournisseur).

Modifier fournisseur

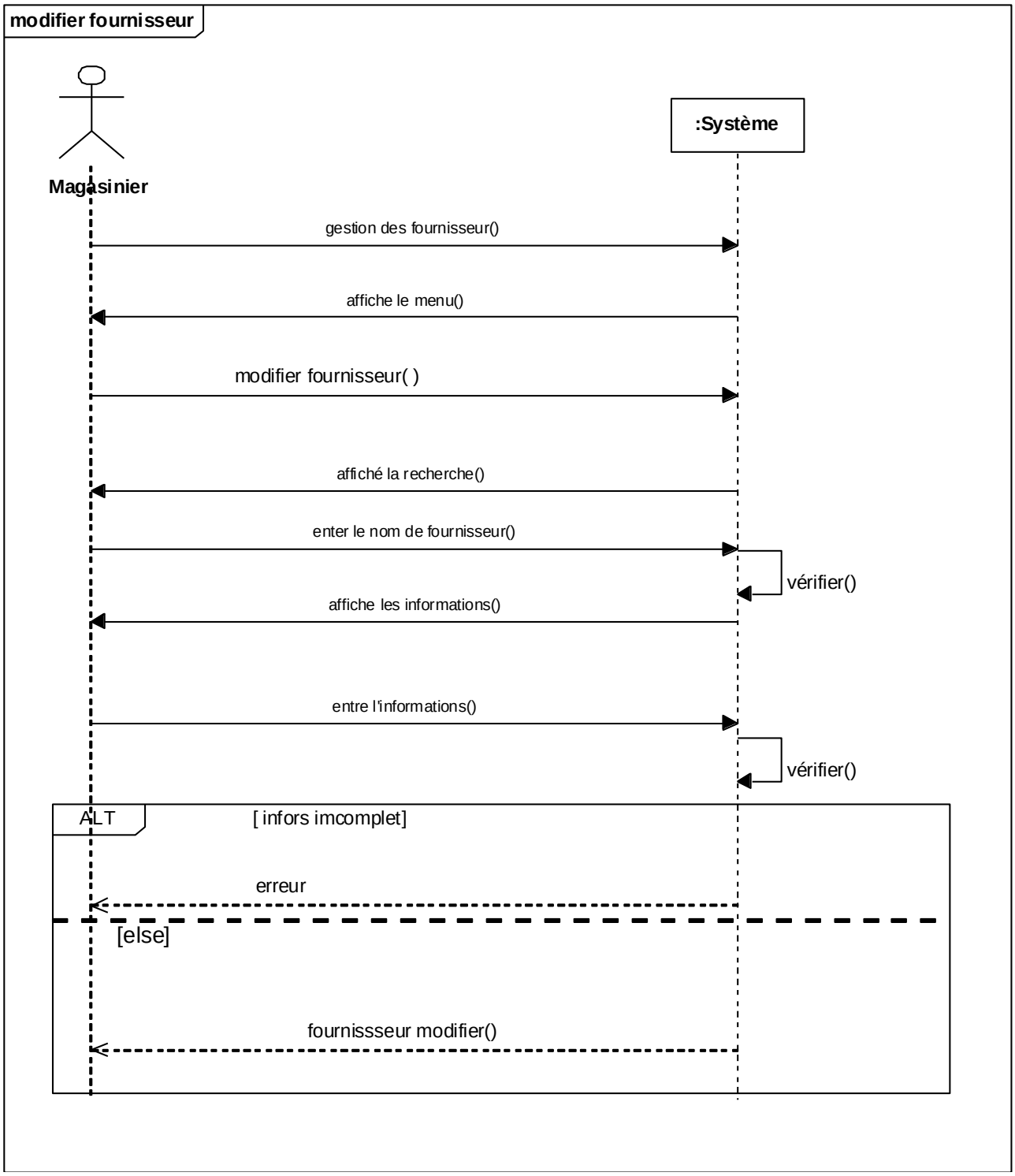


Figure23 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs (modifier fournisseur).

Supprimer fournisseur

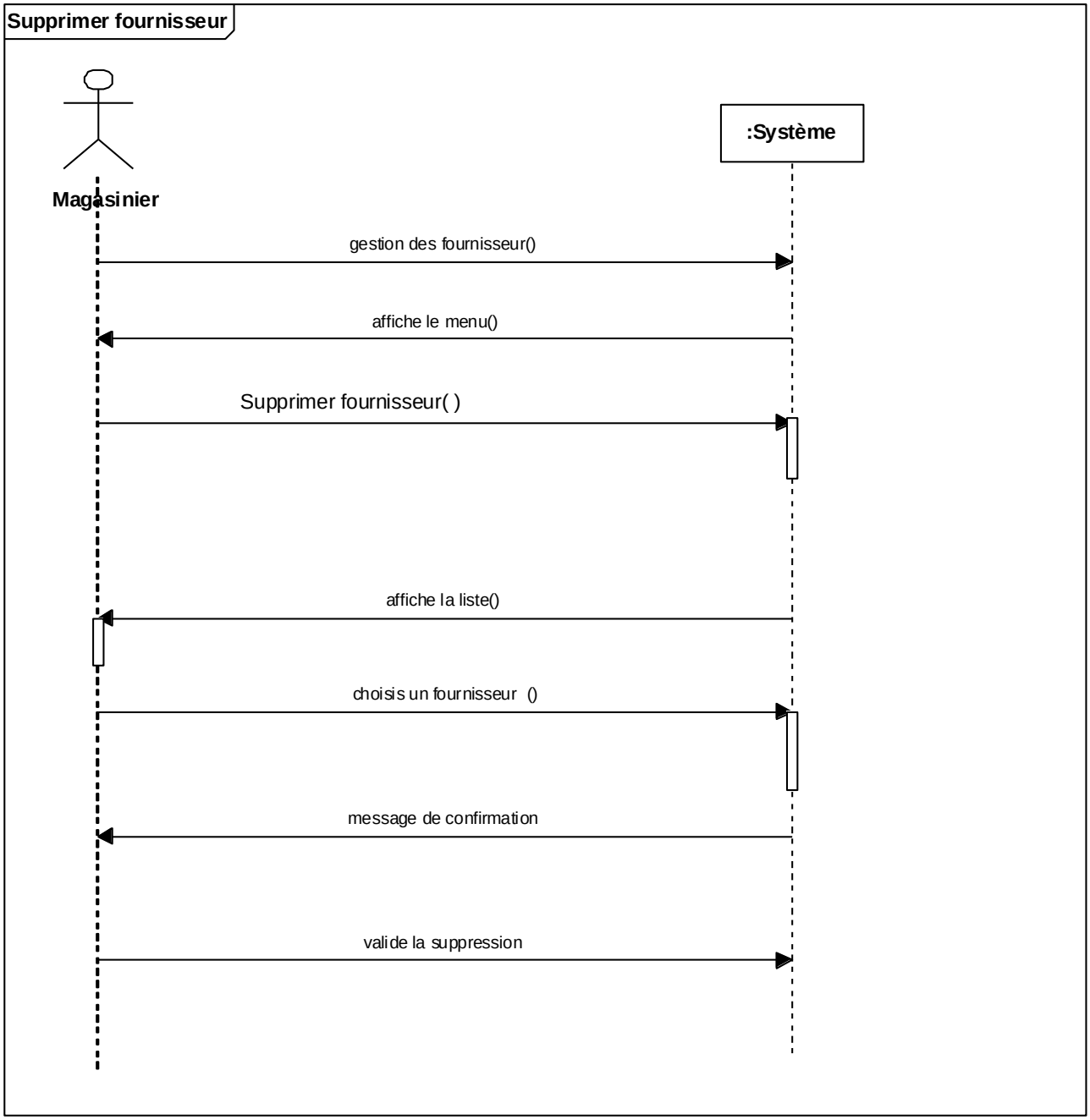


Figure24 : Diagramme de séquence système du cas d'utilisation gestion des fournisseurs (supprimer fournisseur).

Gestion des familles

Cas Ajouter famille

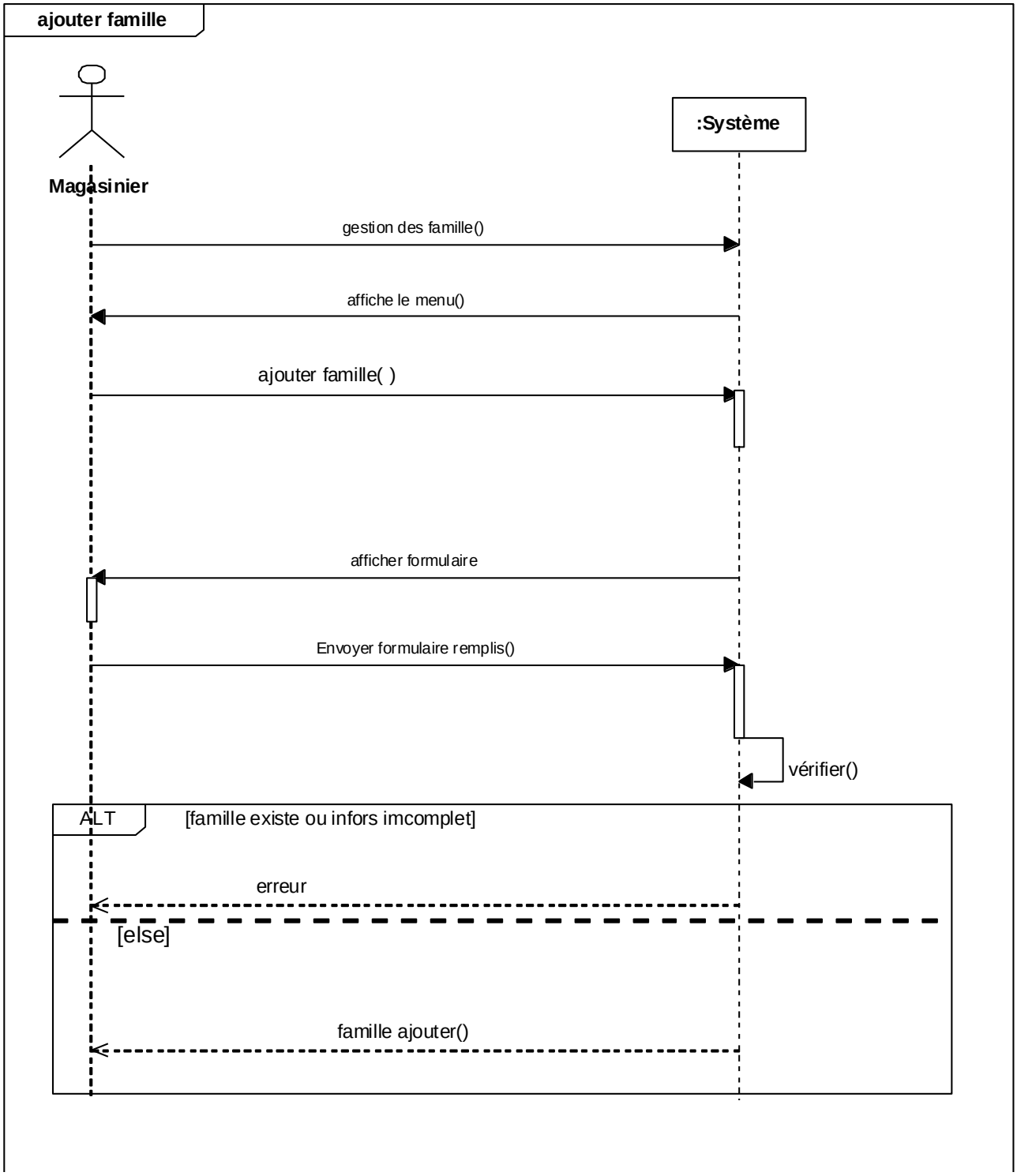


Figure25 : Diagramme de séquence système du cas d'utilisation gestion des familles (ajouter famille).

Cas Modifier famille

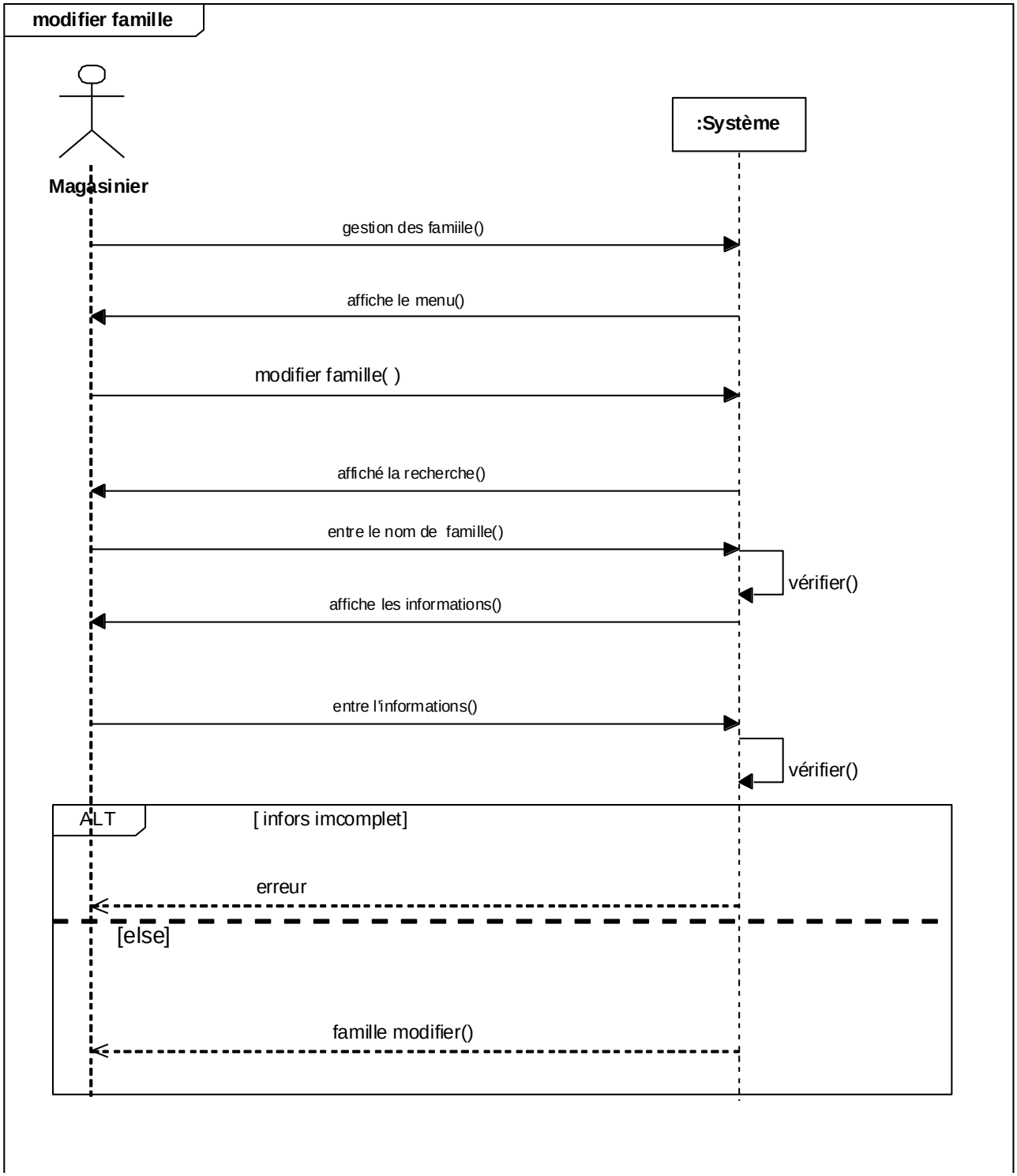


Figure26 : Diagramme de séquence système du cas d'utilisation gestion des familles (modifier famille).

Cas Supprimer famille

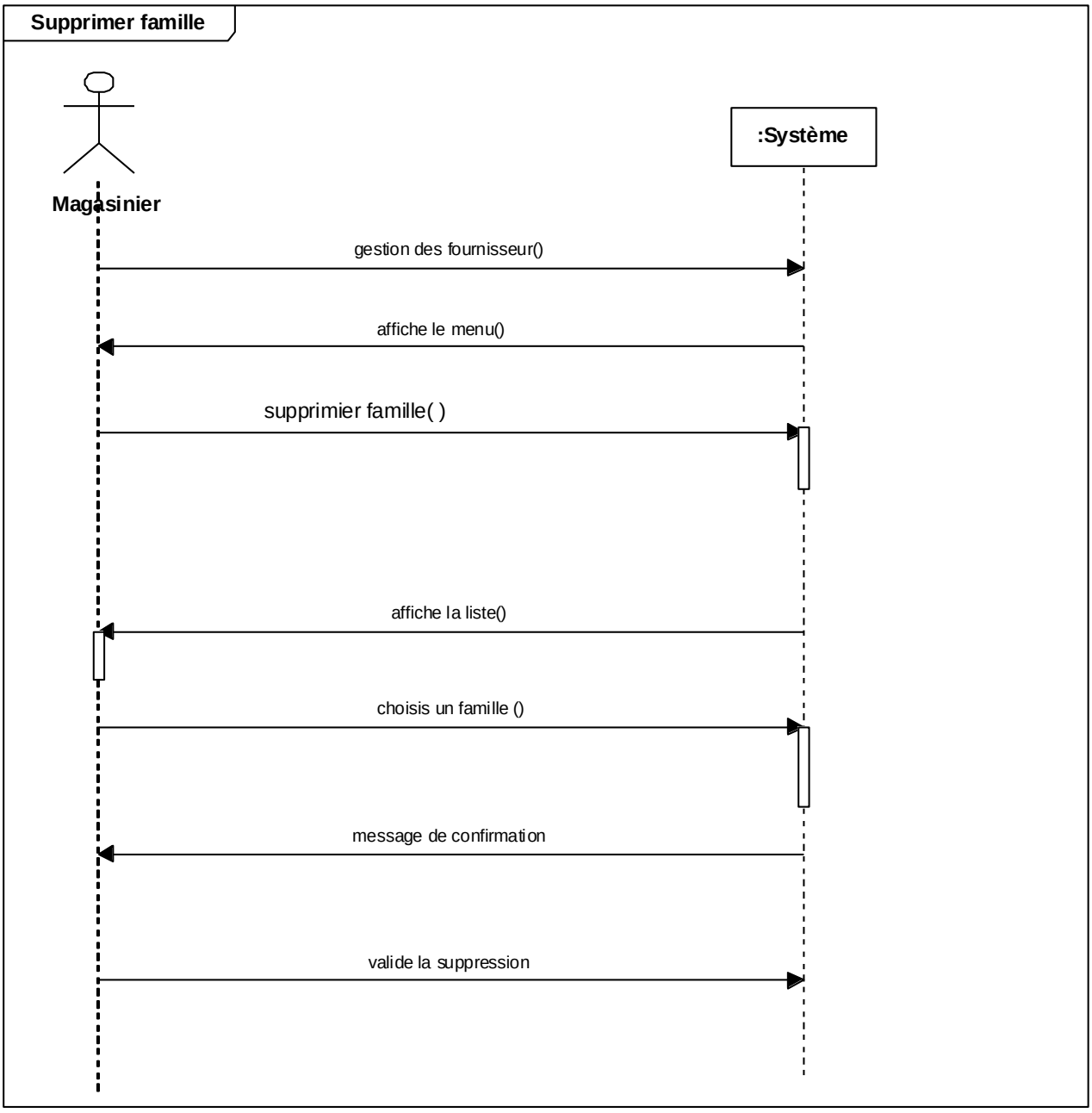


Figure27 : Diagramme de séquence système du cas d'utilisation gestion des familles (supprimer famille).

Etablir menu des repas

Etablir une liste des repas

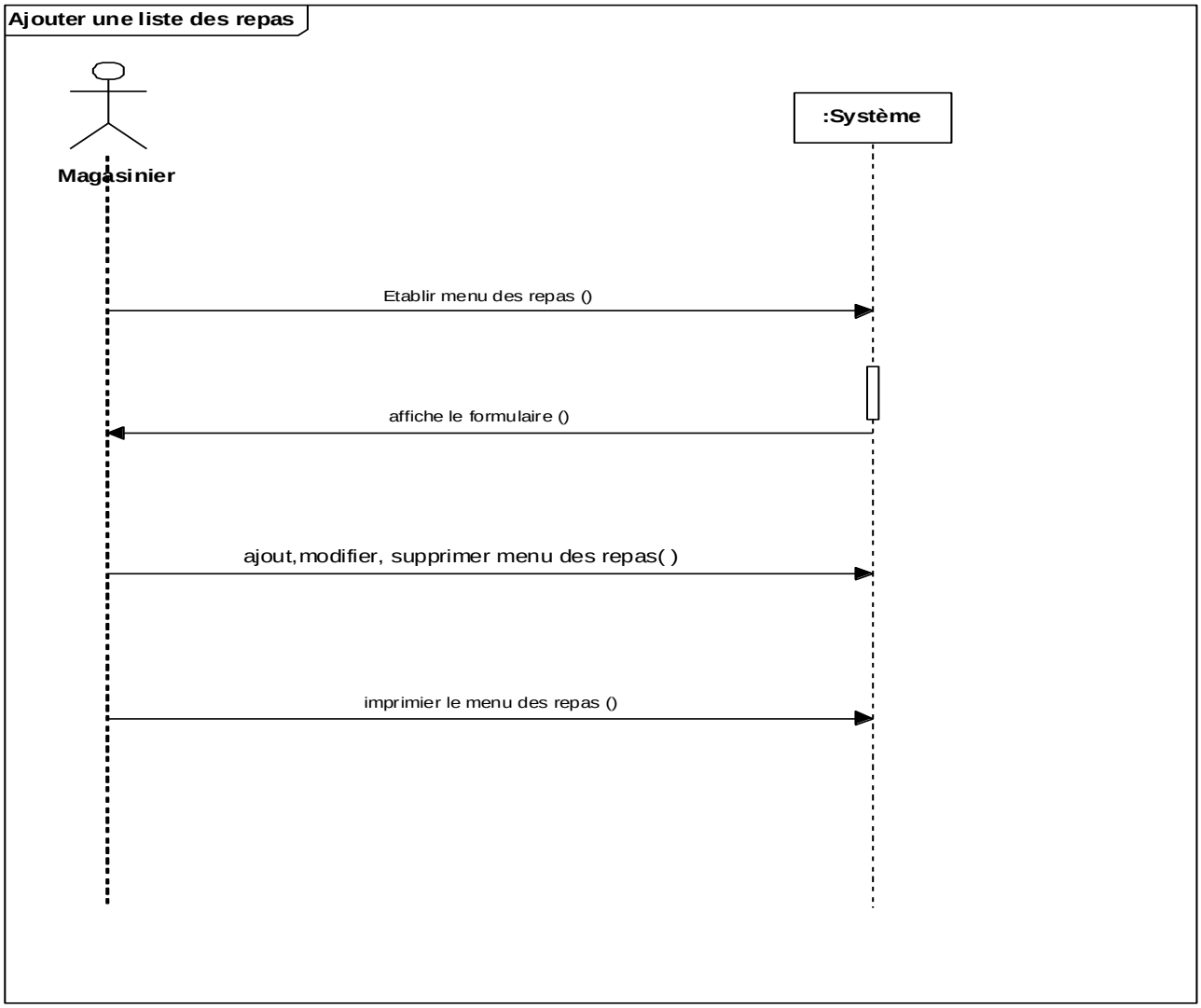


Figure28 : Diagramme de séquence système du cas d'utilisation établir menu des repas

Diagrammes d'activités de navigation

Cas s'authentifier

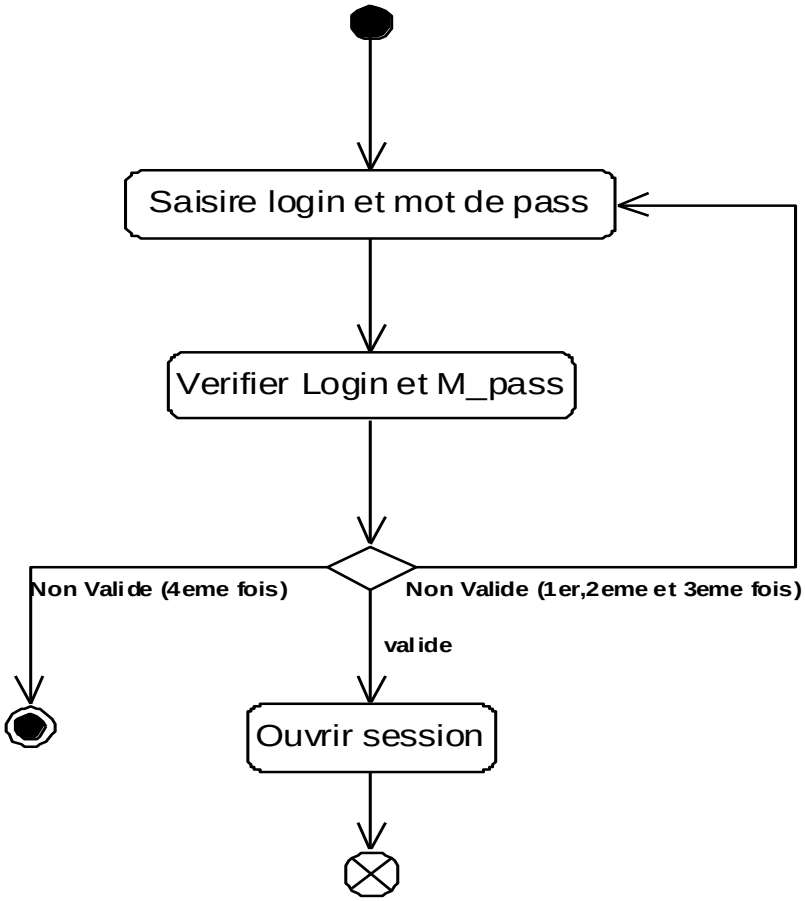


Figure29 : Diagrammes d'activités de navigation de cas « s'authentifier »

Renseigner des articles

Ajouter article

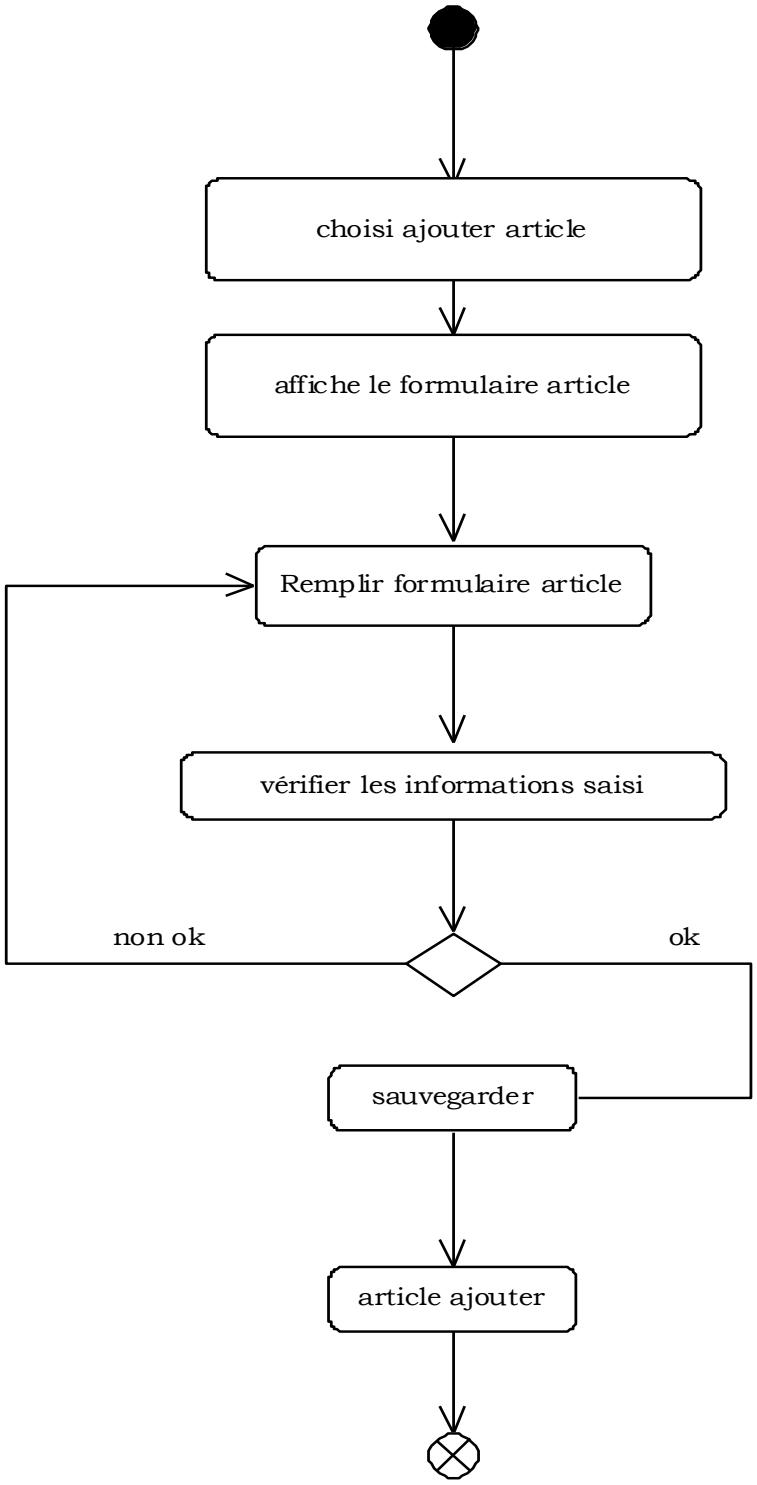


Figure30 : Diagrammes d'activités de navigation : Renseigner des articles (ajouter article)

Cas modifier article

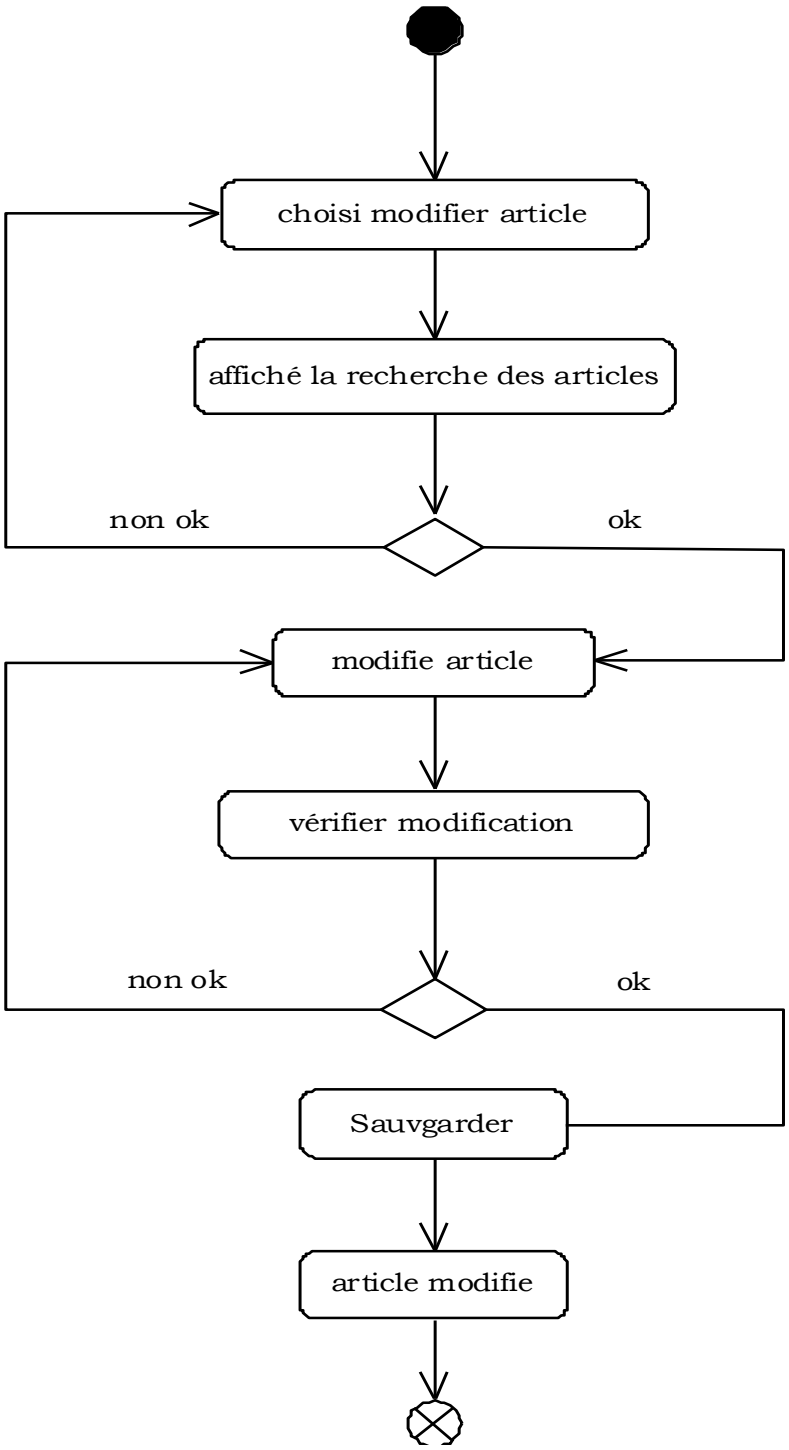


Figure31: Diagrammes d'activités de navigation : Renseigner des articles (modifier article)

Cas supprimer article

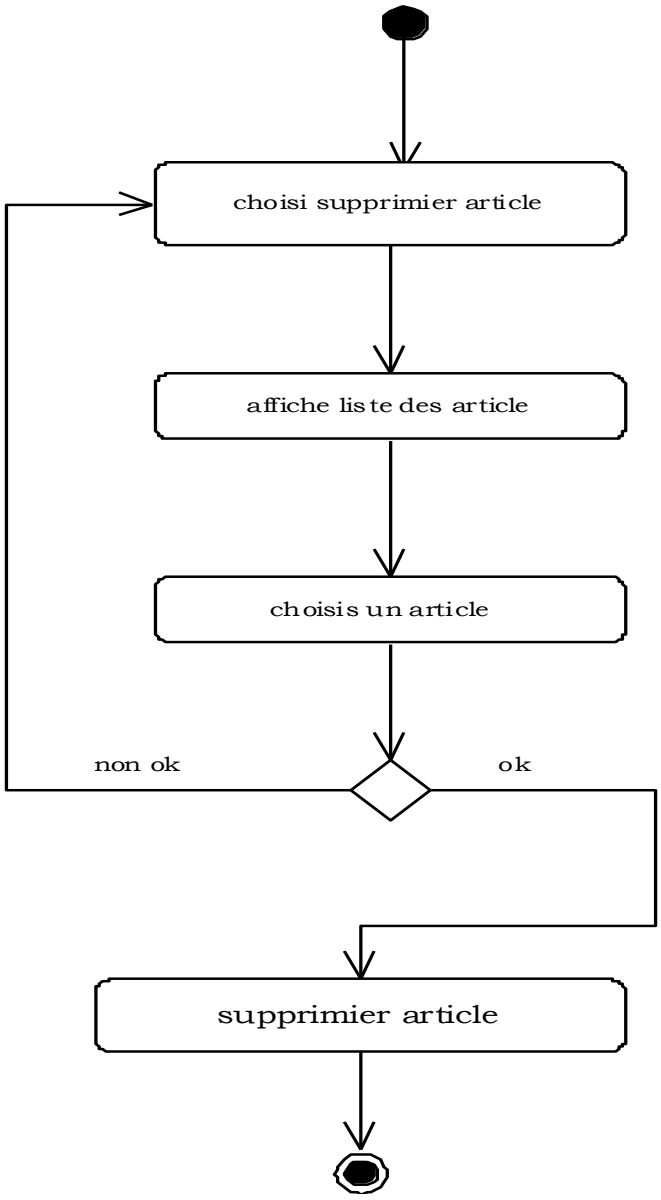


Figure32 : Diagrammes d'activités de navigation : Renseigner des articles (supprimer article)

Gestion de fournisseur

Cas ajouter fournisseur

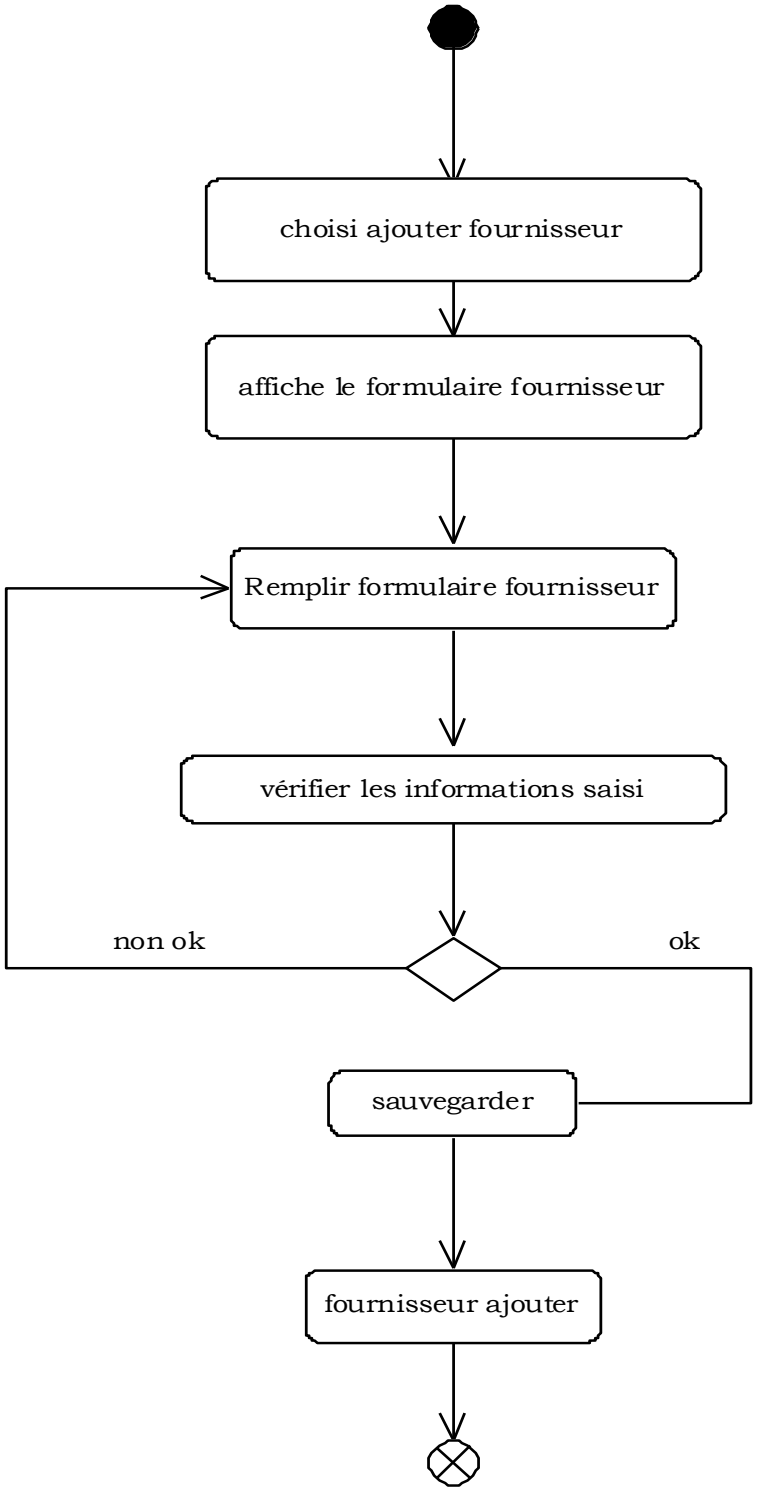


Figure33 : Diagrammes d'activités de navigation : gestion de fournisseur (ajouter fournisseur)

Cas modifier fournisseur

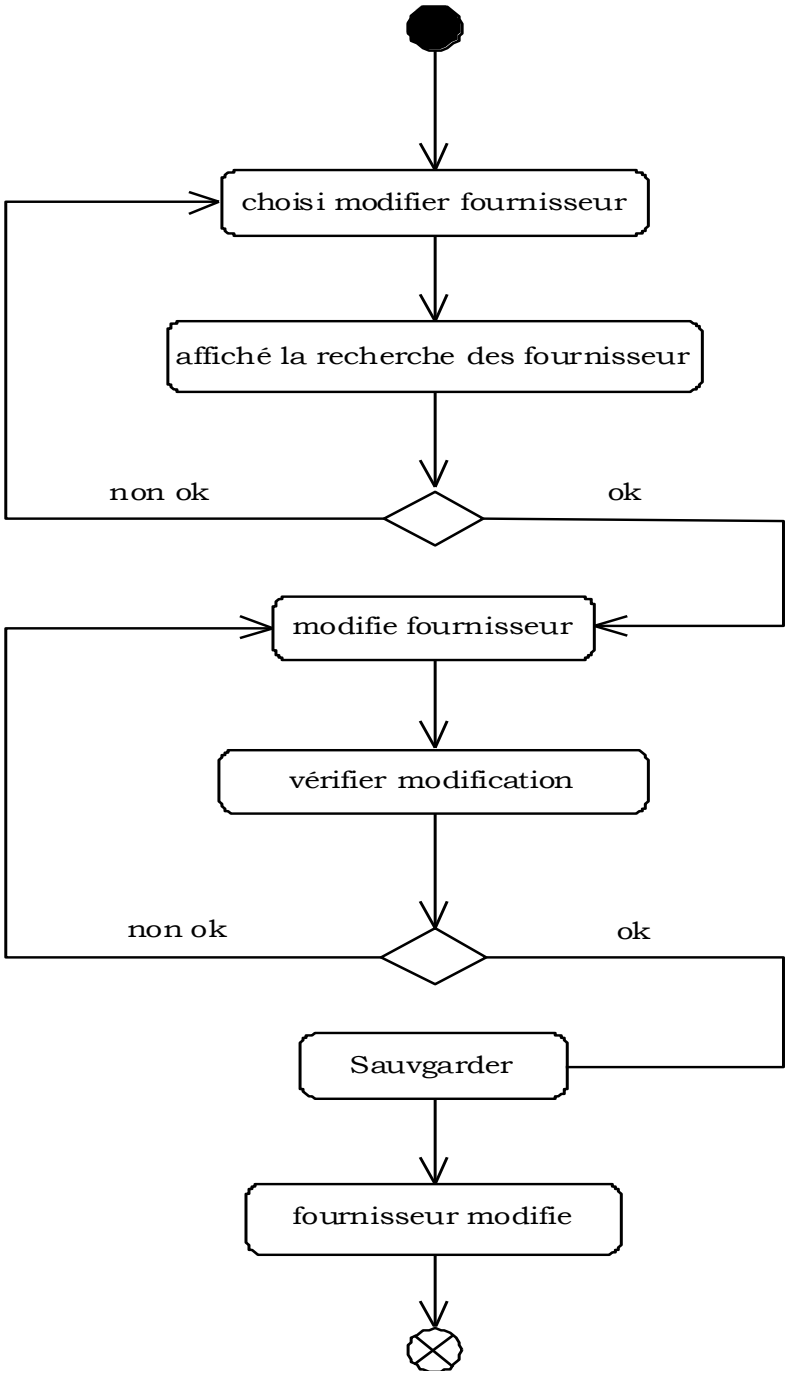


Figure34 : Diagrammes d'activités de navigation : gestion de fournisseur (modifier fournisseur)

Cas supprimer fournisseur

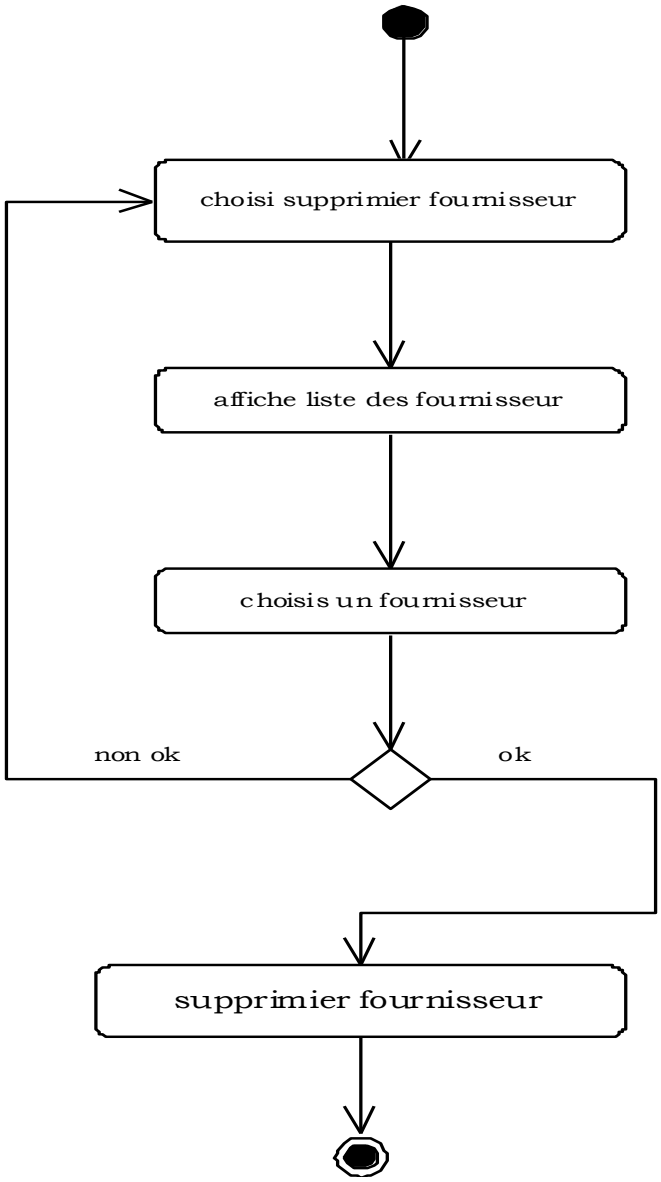


Figure35 : Diagrammes d'activités de navigation :gestion de fournisseur (supprimer fournisseur)

Diagrammes d'interactions

Cas s'authentifier

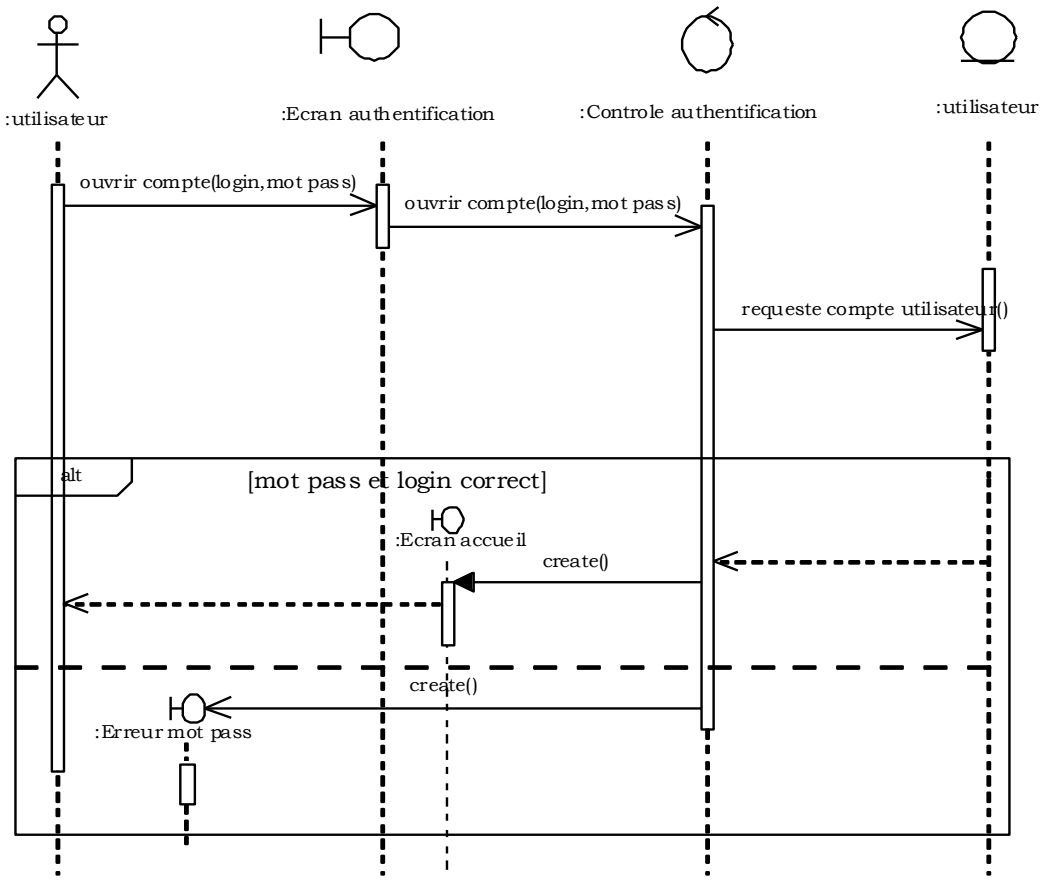


Figure36 : Diagrammes d'interactions cas d'utilisation s'authentifier

Diagramme des classes participantes

Cas s'authentifier

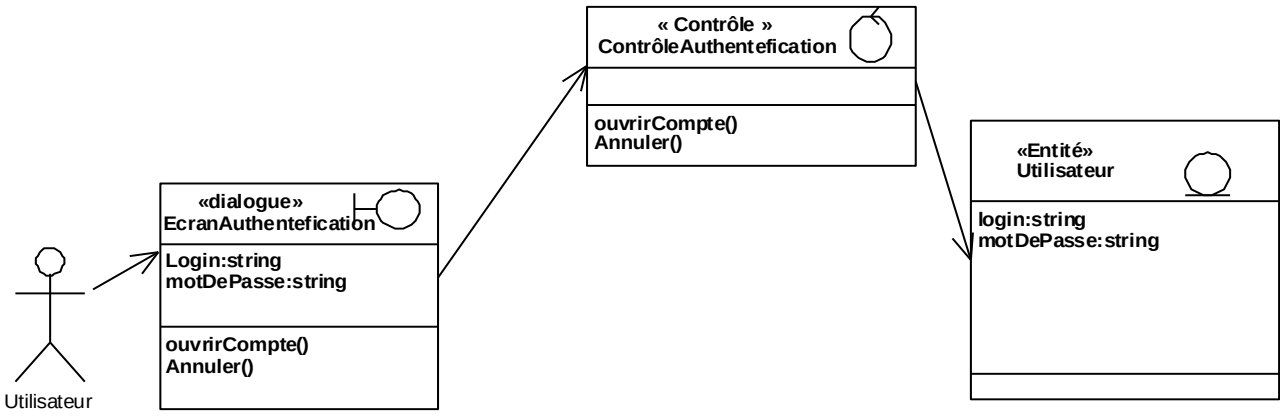


Figure37 : Diagramme de classes participantes cas d'utilisation s'authentifier

Renseigner des articles

Cas d'ajout

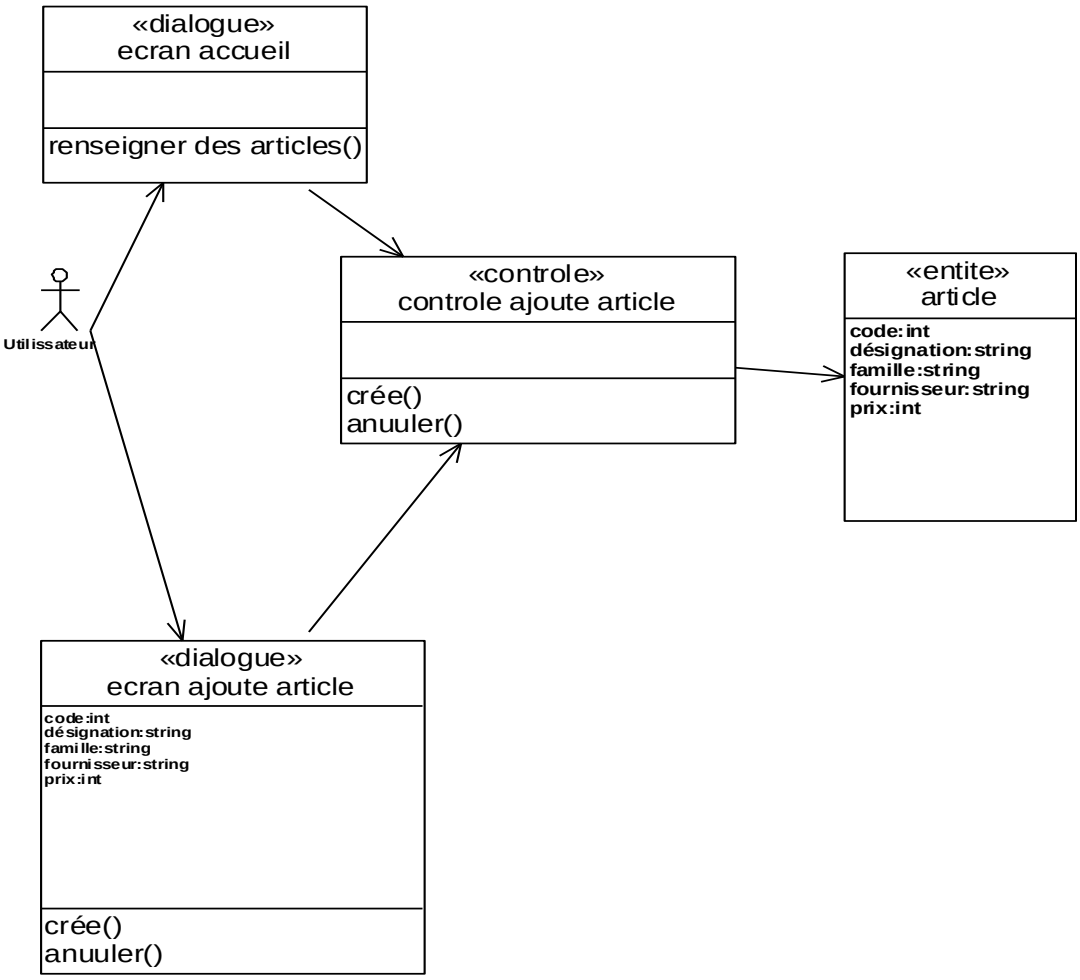


Figure38 : Diagramme de classes participantes : Renseigner des articles (ajouter article)

Cas modification

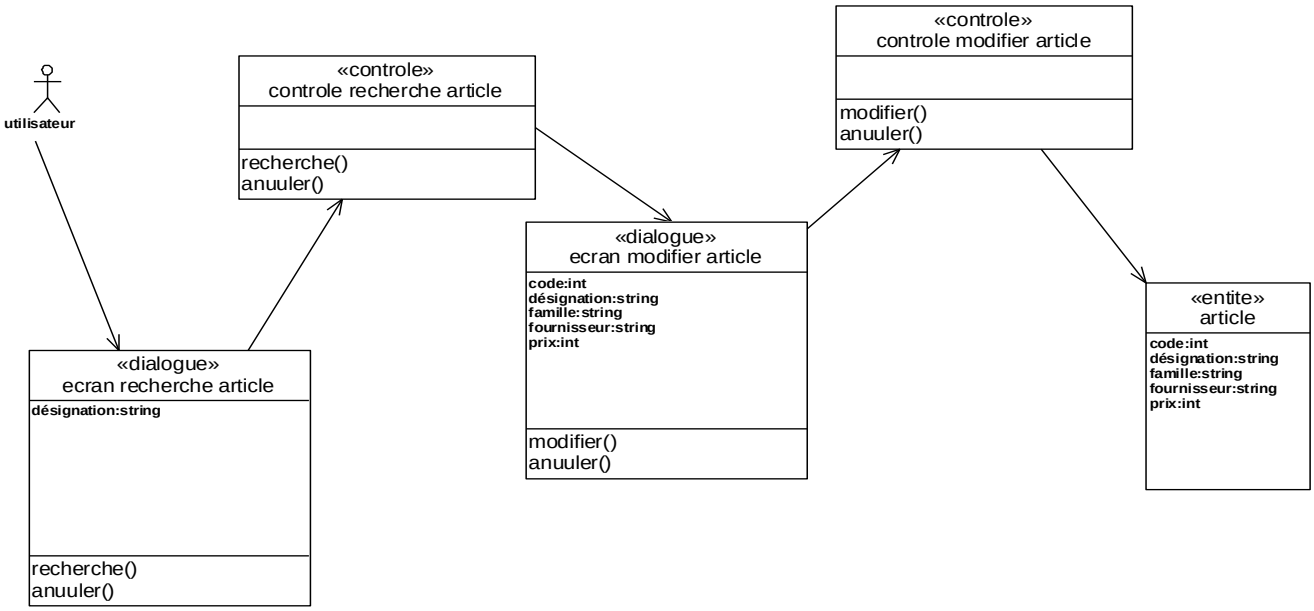


Figure39 : Diagramme de classes participantes : Renseigner des articles (modifier article)

Cas de suppression

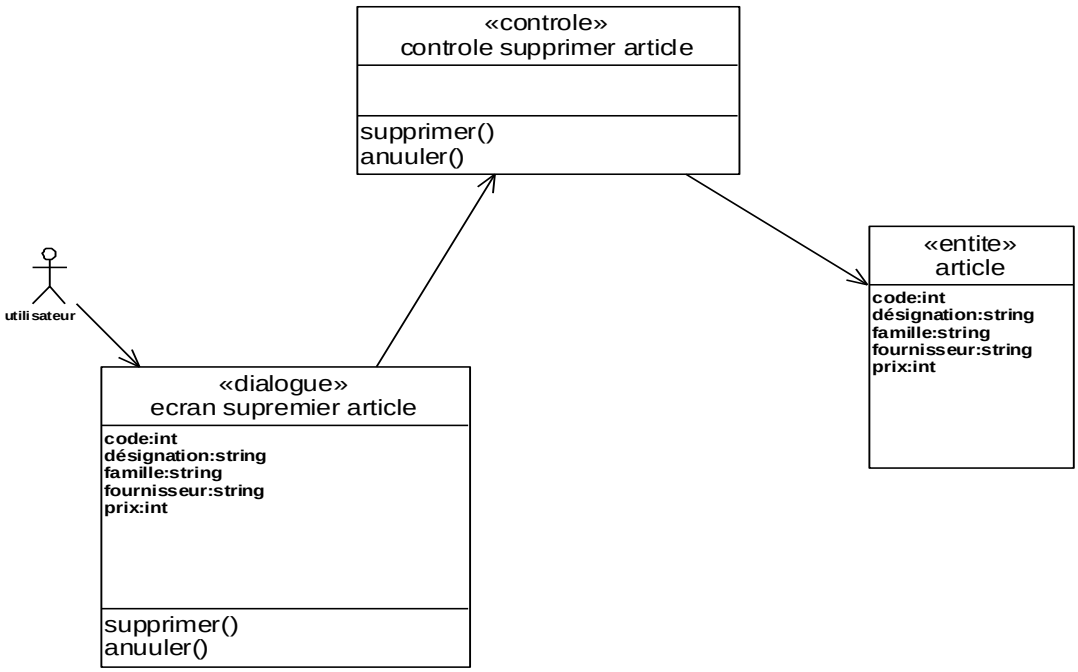


Figure40 : Diagramme de classes participantes Renseigner des articles (supprimer article)

Diagramme de séquence

Renseigner article

Cas d'ajout

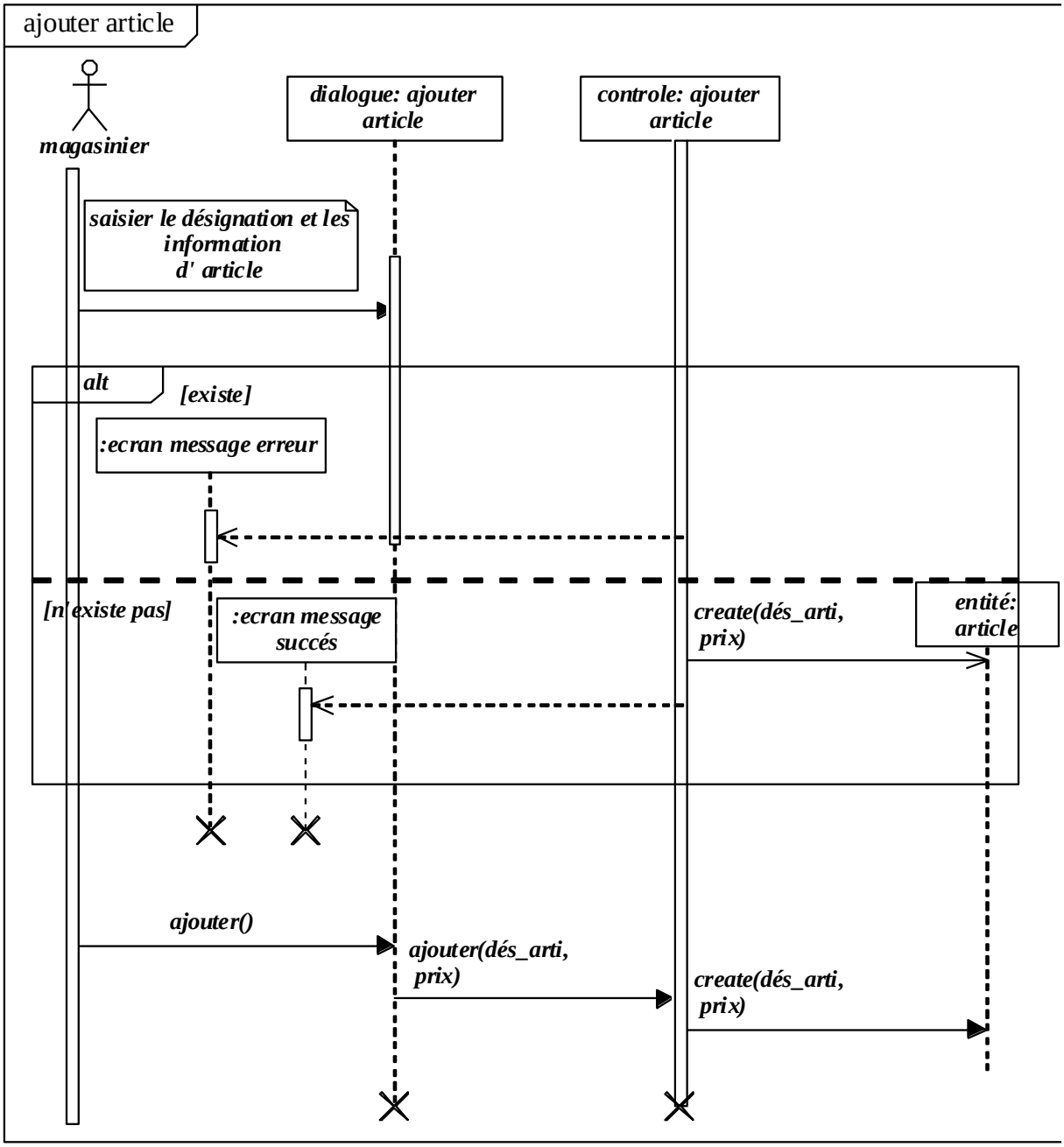


Figure41 : Diagramme de séquence : Renseigner des articles (ajouter article).

Cas de modification

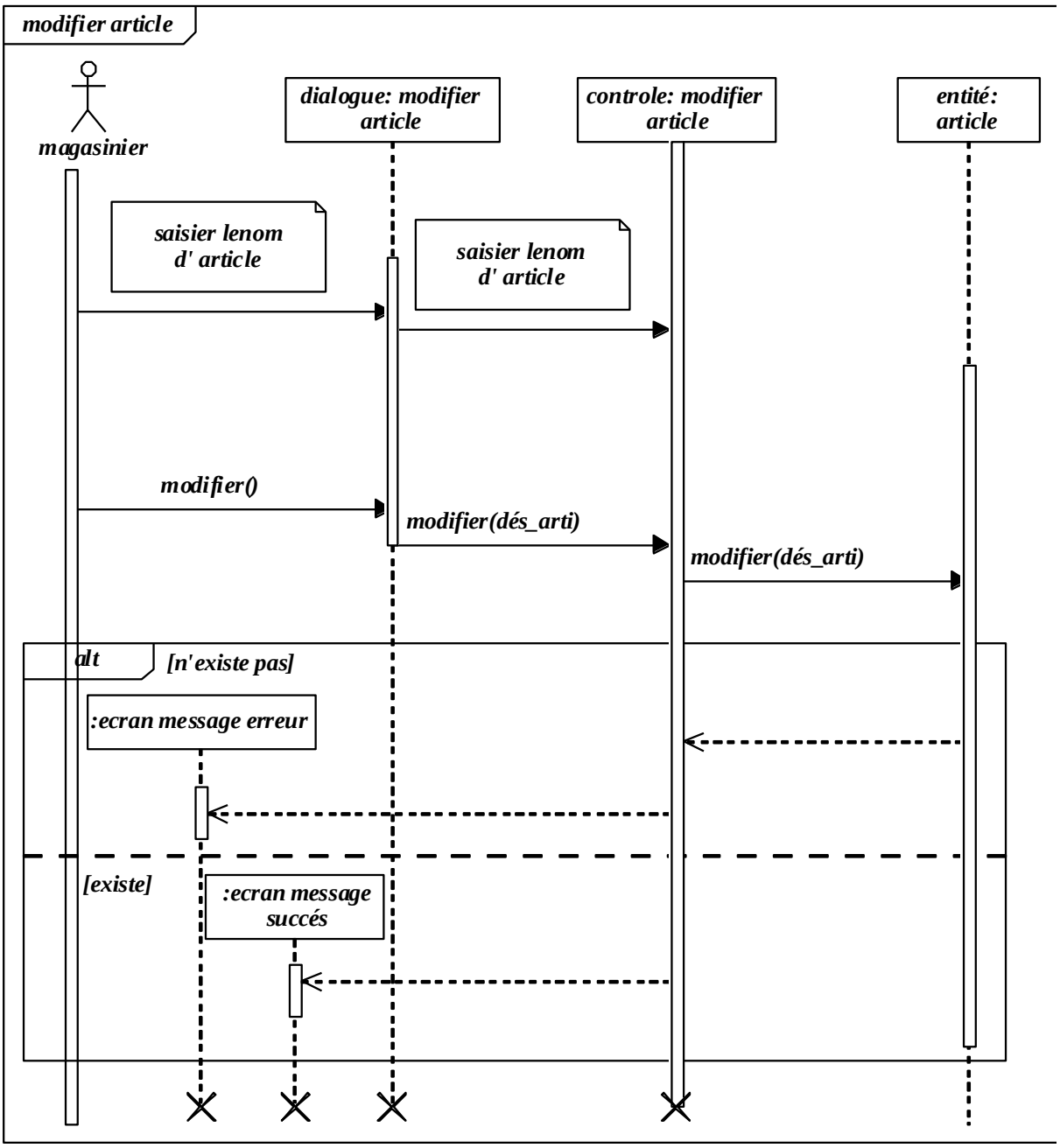


Figure42 : Diagramme de séquence : Renseigner des articles (Modifier article).

Diagramme de classes

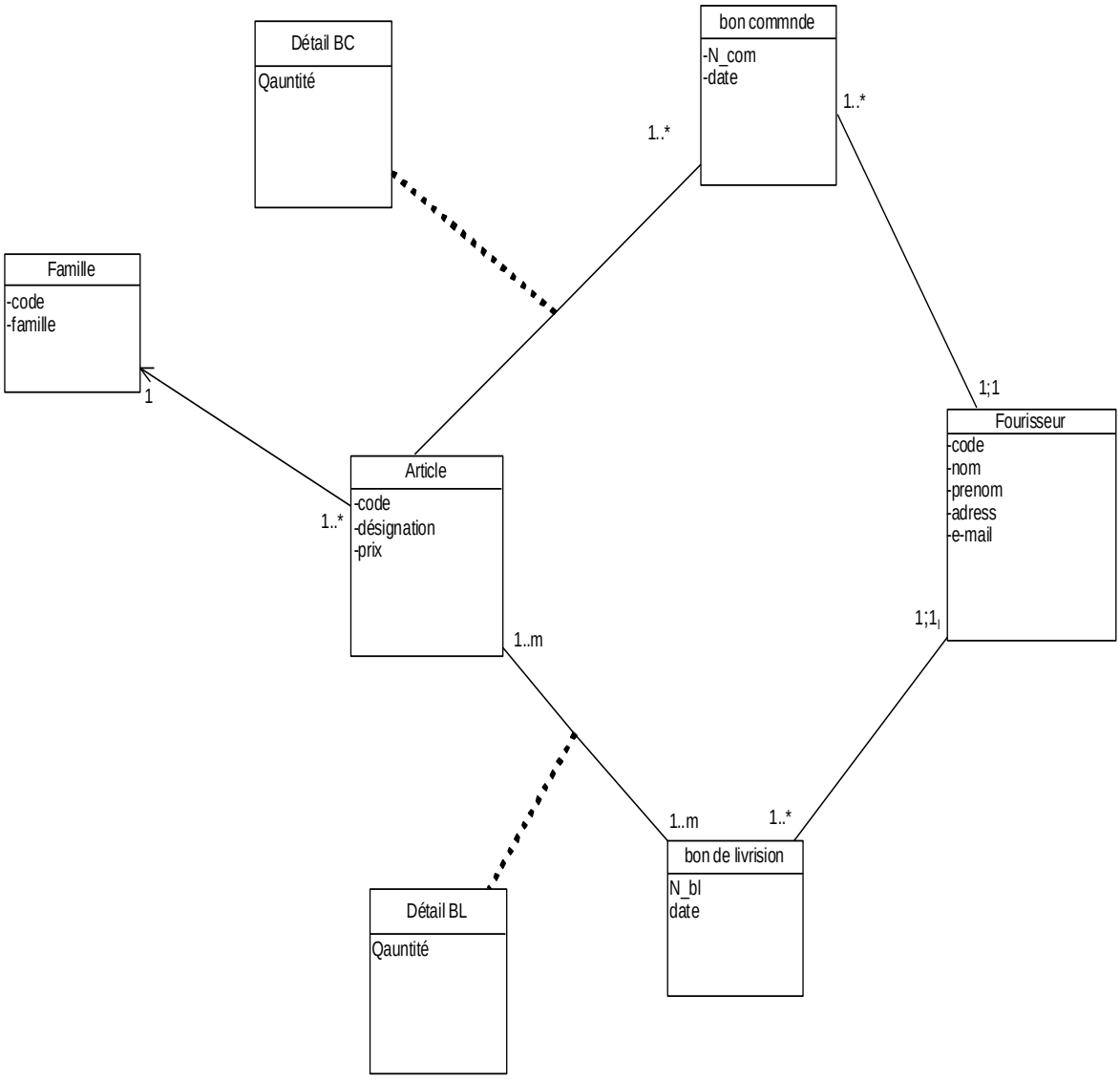


Figure43 : Diagramme de classes.

Chapitre5

IMPLEMENTATION

5.1 Introduction

Chaque application doit être accompagnée ou illustrée par un dossier technique, c'est-à-dire passer des spécifications détaillées et des contraintes techniques à la définition d'une solution en termes de traitement de données (production d'un logiciel), et qui constitue un guide manuel pour n'importe quel utilisateur. Dans ce dernier chapitre, nous mettons l'accent sur quelques caractéristiques de notre application et des ressources utilisées pour le développement. L'environnement de développement opté est DELPHI 7avec access2010 comme SGBD.

5.2 Modèle relationnel

Le modèle relationnel a été inventé en 1970 et a fait l'objet de très nombreuses recherches qui ont débouché sur la réalisation et la commercialisation des SGBD relationnels. C'est le modèle le plus utilisé par les SGBD actuellement disponible sur le marché.

5.2.1 Définition

Le modèle relationnel est un modèle de données simple fondé sur la théorie mathématique bien connue des relations. Cette théorie se construit à partir de la théorie des ensembles.il y a trois notions importantes pour introduire les bases de données relationnelles.

- Domaine : le domaine est un ensemble de valeur caractérisé par un nom.
- Relation : est un sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom.
- Attribut : est une colonne d'une relation caractérisée par un nom.

5.2.2 Le passage du diagramme de classe au modèle relationnel

Afin de pouvoir implémenter une base de données, il faut pouvoir traduire le modèle conceptuel en modèle logique. Cela signifie qu'il faut pouvoir convertir un modèle UML en modèle relationnel. Le passage du modèle conceptuel au modèle relationnel est systématique en appliquant les règles de transformation suivantes :

- Transformation des classes :

Pour chaque classe C non abstraite, on crée une relation R dont le schéma est celui de la classe C. la clé primaire de R est une des clés de C.

- Transformation des associations :

  ➢ **associations 1..*** :

  Pour chaque association binaire A de type 1..* entre les classes S et T (représentée par les relations RS et RT respectivement) on inclut dans la définition de RT comme clé étrangère la clé de RS

  ➢ **Associations M..*et associations de degré supérieur à2** :

  Pour chaque association binaire A de type M..* ou Pour chaque association A de degré supérieur à2, on crée une nouvelle relation RA pour représenter A. on met dans RA comme clé étrangère, les clés de toutes les relations correspondant aux classes participant à A et dont la concaténation formera sa clé.

  ➢ **associations 1..1 :**

  Pour chaque association binaire A de type 1..1 entre les classes S et T (représentée par les relations RS et RT respectivement) RS définit par clé primaire qui est également définie comme clé étrangère vers RT.

  ➢ **Agrégation**

  L'association de type agrégation se traite de la même façon.

5.2.3 Représentation de notre base de données

Famille : sert stocké les information relatives à une famille

Champ	type
Code	Int(11)
Famille	texte

Article : sert stocké les information relatives à un article

Champ	type
Code	Int(11)
Désignation	texte
Famille	texte
Fournisseur	texte
Prix	Int(11)

Fournisseur : sert stocké les information relative a un fournisseur

Champ	type
Code	Int(11)
Nom	texte
Prenom	texte
Adress	texte
Tel	Int(11)
e-mail	texte

Commande article: sert à stocker les informations relatives à une commande d'articles

Champ	type
N_com	Int(11)
Date	Date
Fournisseur	text
Désignation	text
Prix	Int(11)
Quantité	Int(11)

Réception article: sert à stocker les informations relatives à une réception article

Champ	type
Nbl	Int(11)
Date	date
Code	Int(11)
Désignation	texte
Quantité	Int(11)
Prix	Int(11)

Consommation article: sert à stocker les informations relatives à une consommation article

Champ	Type
Nbs	Int(11)
Date	Date
Code	Int(11)
Désignation	Texte
Quantité	Int(11)

5.3 La base de données ACCESS 2010

Avec Microsoft Office Access 2010, les informaticiens peuvent, en un clin d'œil et en toute simplicité, contrôler les informations dont ils disposent et créer des états à partir de celles-ci grâce à l'interface utilisateur Microsoft Office Fluent et aux fonctionnalités de conception interactives qui n'exigent aucune connaissance approfondie en matière de base de données.

MS Access est un logiciel utilisant des fichiers au format Access (extension de fichier mdb pour Microsoft Data Base (extension *.accdb depuis la version 2010)). Il est compatible avec les requêtes SQL (sous certaines restrictions) et dispose d'une interface graphique pour saisir les requêtes (QBE - Query By Exemple - Requête par l'exemple). Il permet aussi de configurer, avec des assistants ou librement, des formulaires et sous-formulaires de saisie, des états imprimables (avec regroupements de données selon divers critères et des totalisations, sous-totalisations, conditionnelles ou non), des pages html liées aux données d'une base, des macros et des modules VBA.

Comme beaucoup de systèmes de gestion de base de données relationnelle, ses données peuvent être utilisées dans des programmes écrits dans divers langages.

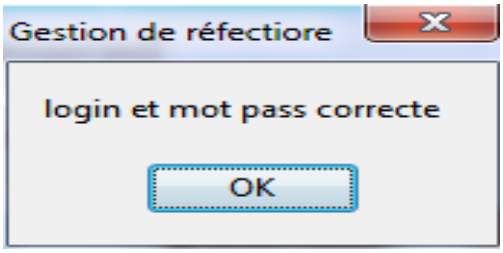
Les langages couramment utilisés avec Access sont le Visual Basic for Application (VBA) et les langages qui disposent de modules d'accès aux données pour les fichiers .mdb : Delphi de Borland, Visual Basic, C++ sous Visual Studio de Microsoft.

Les bases de données produites par Access restent accessibles à tous les langages de programmation qui permettent une connexion à une base ODBC.

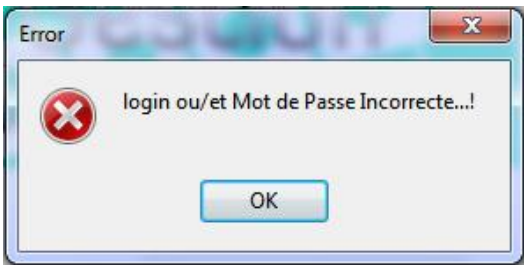
5.4 Les interfaces de l'application

5.4.1 Interface authentification : L'interface d'authentification permet au magasinier d'accéder à sa session par son login et son mot de passe.

- Si le mot de passe ou login erroné alors on affiche la boite de dialogue suivante



- Si le mot de passe et login correcte alors on affiche la boite de dialogue suivante

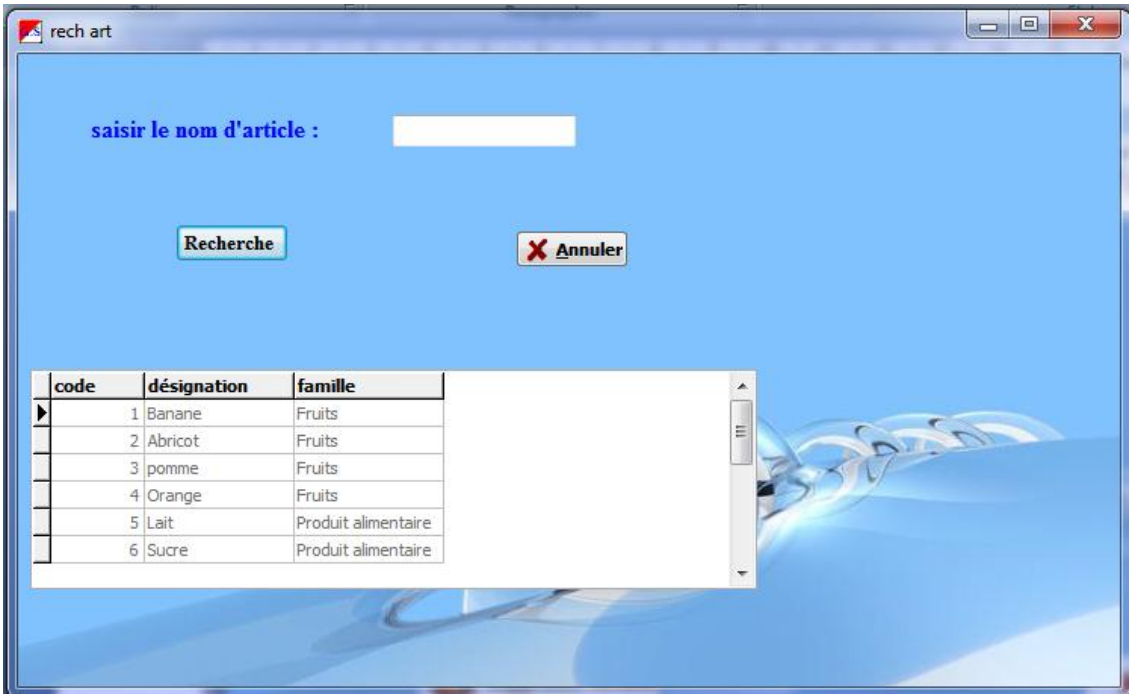


5.4.2 Interface menu principale : affichée directement après l'authentification. L'interface du menu principal permet au magasinier de choisir un cas.

5.4.3 Interface Renseigner des articles :

5.4.4 Interface Ajouter Article :

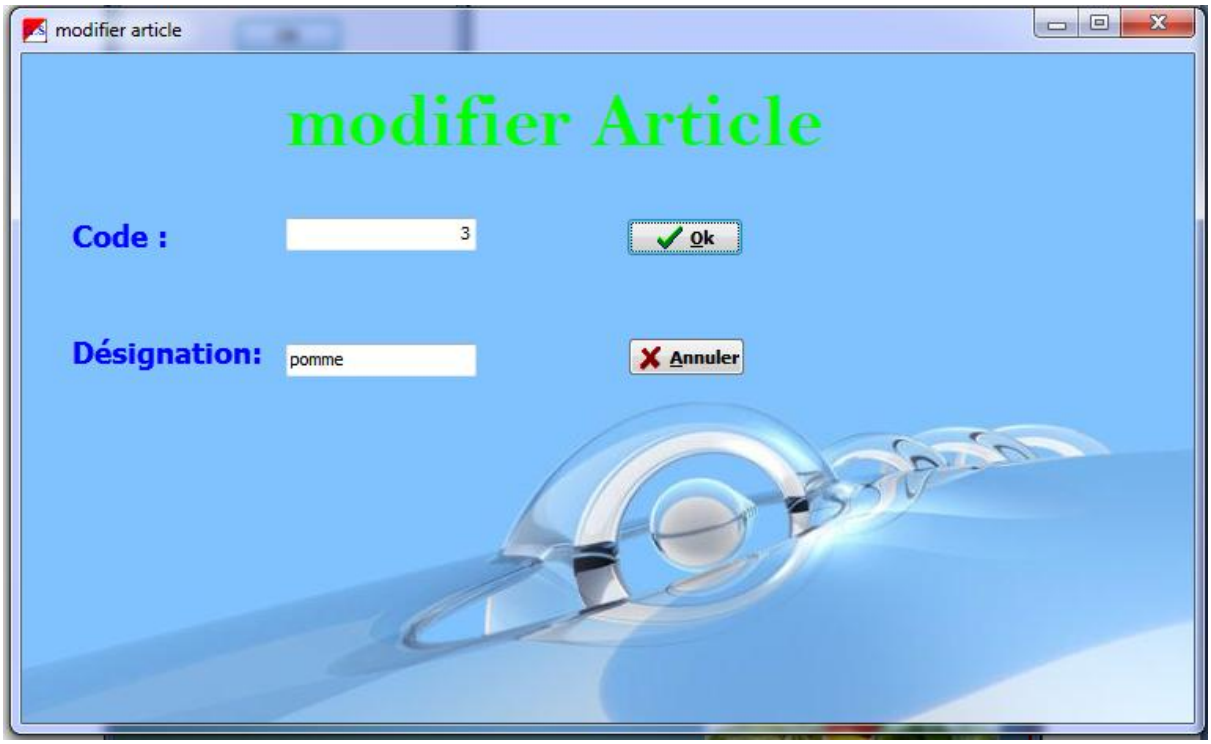
5.4.5 Interface de recherche pour modifier article :



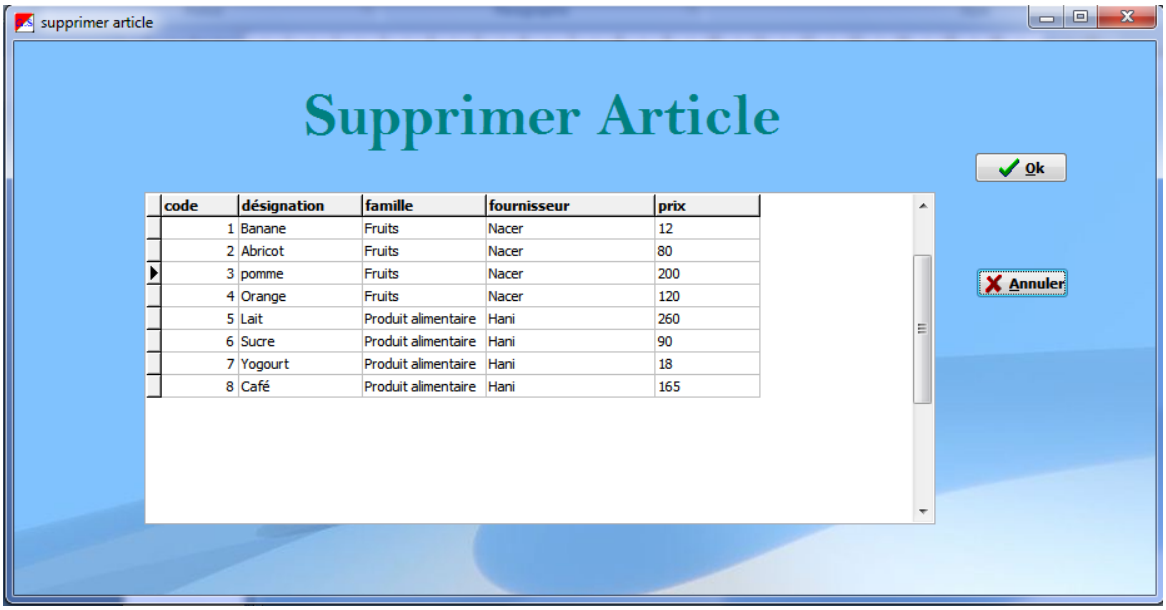
Si l'article n'existe pas dans la base de données, le système affiche la boite de dialogue suivante

Si l'article existe dans la base de données, le système affiche :

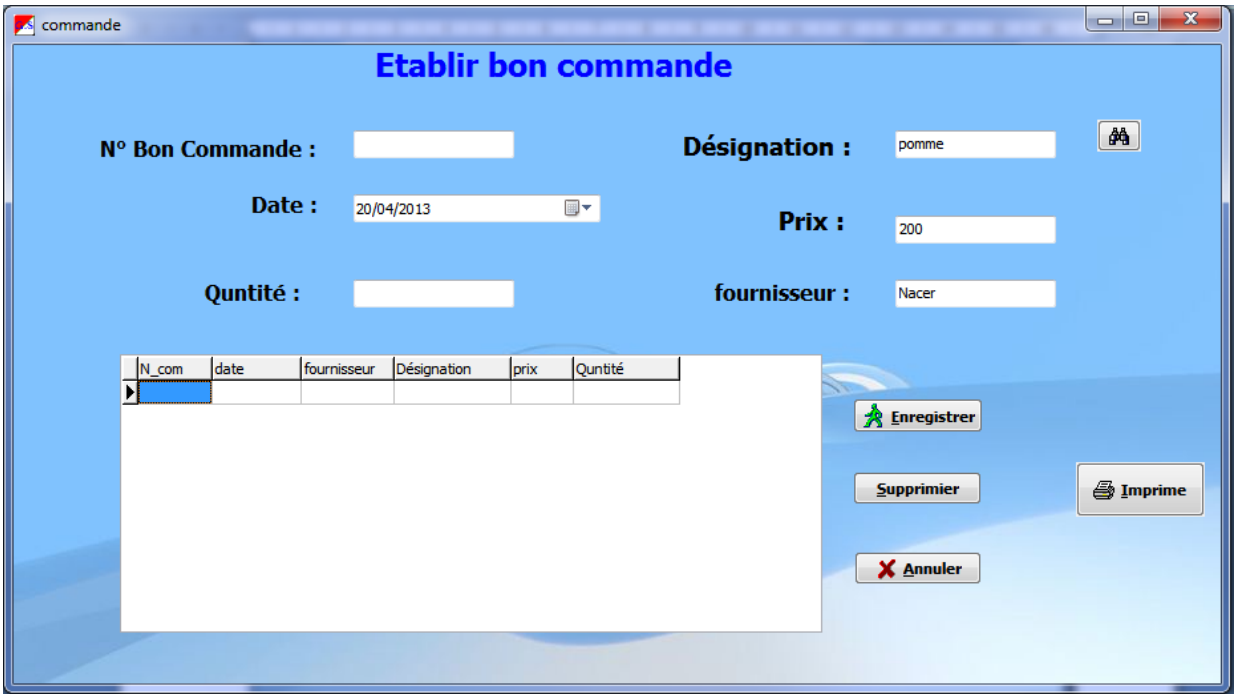
5.4.6 Interface : modifier article :



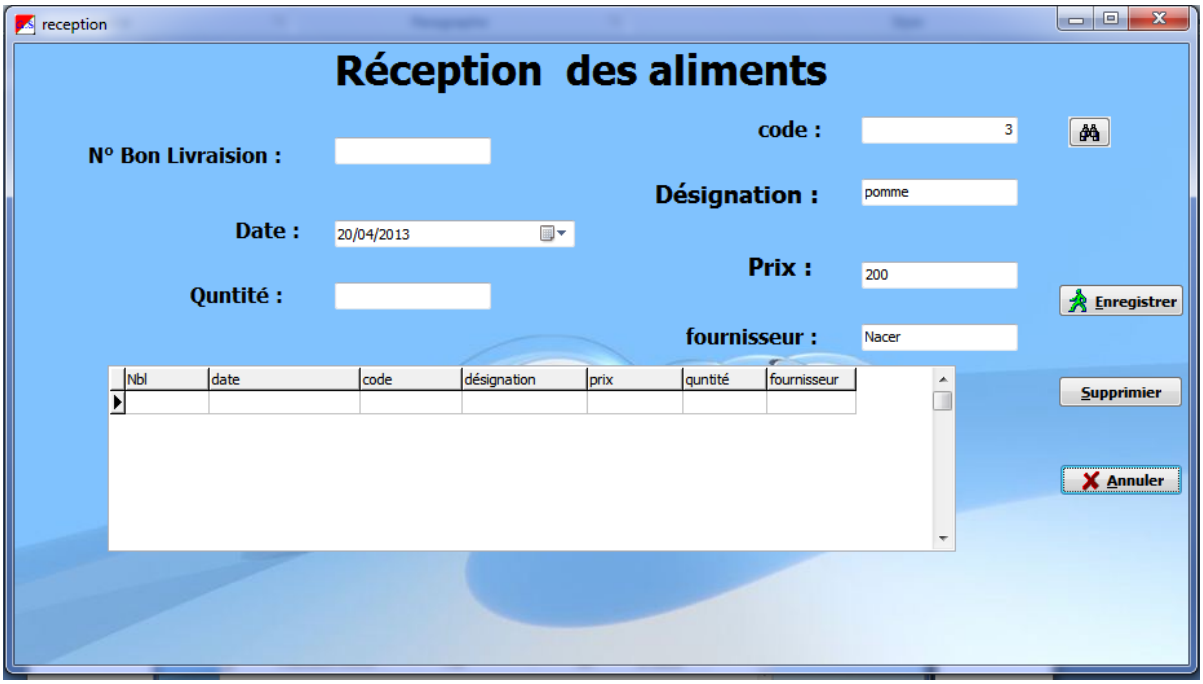
5.6.7 Interface : supprimer article :



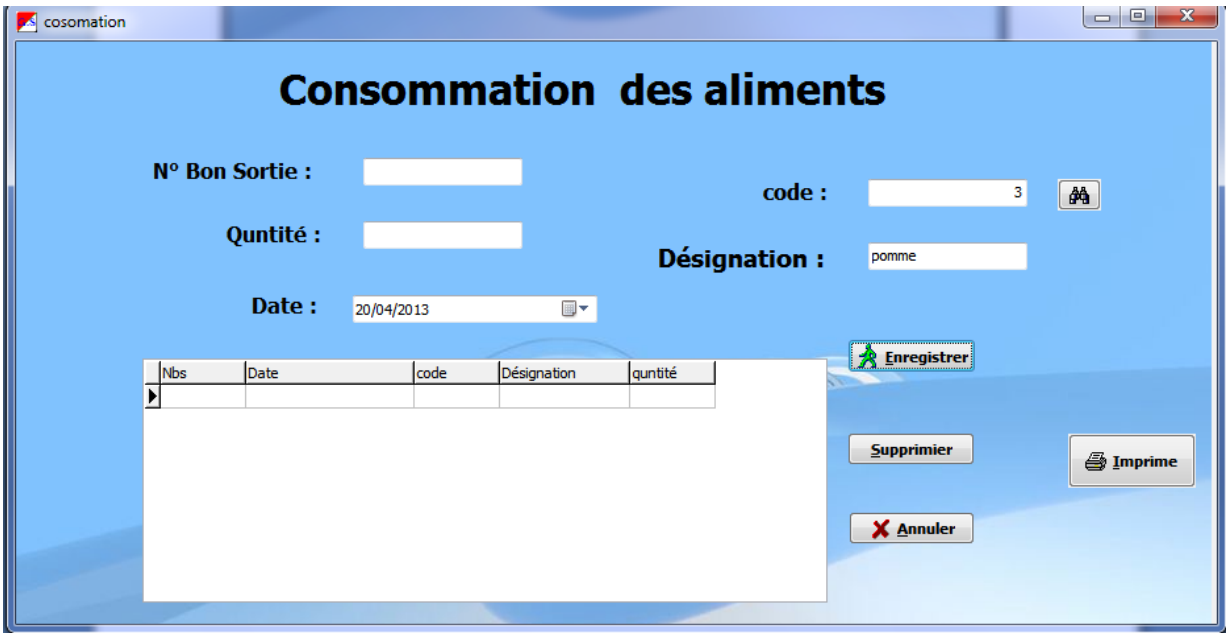
5.4.8 Interface établir bon commande :



5.4.9 Interface réception des aliments :



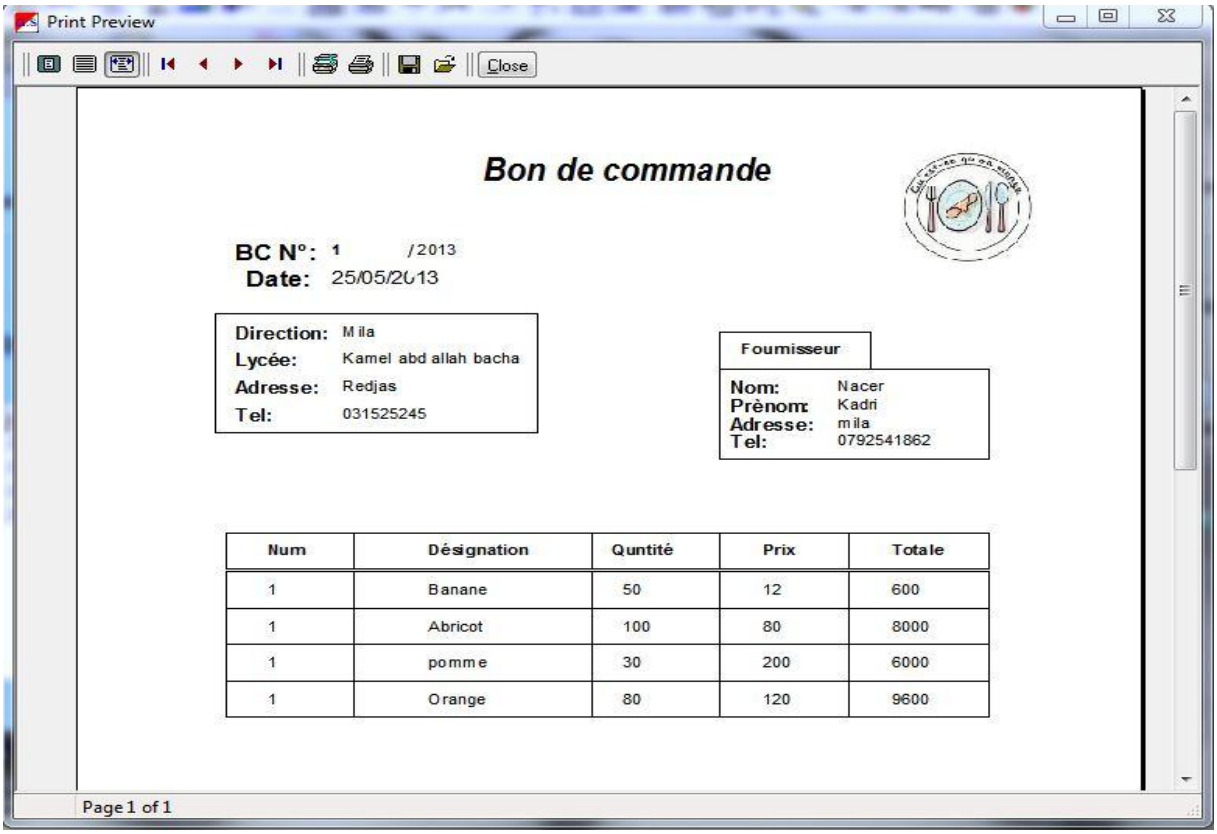
5.4.10 Interface consommation des aliments :



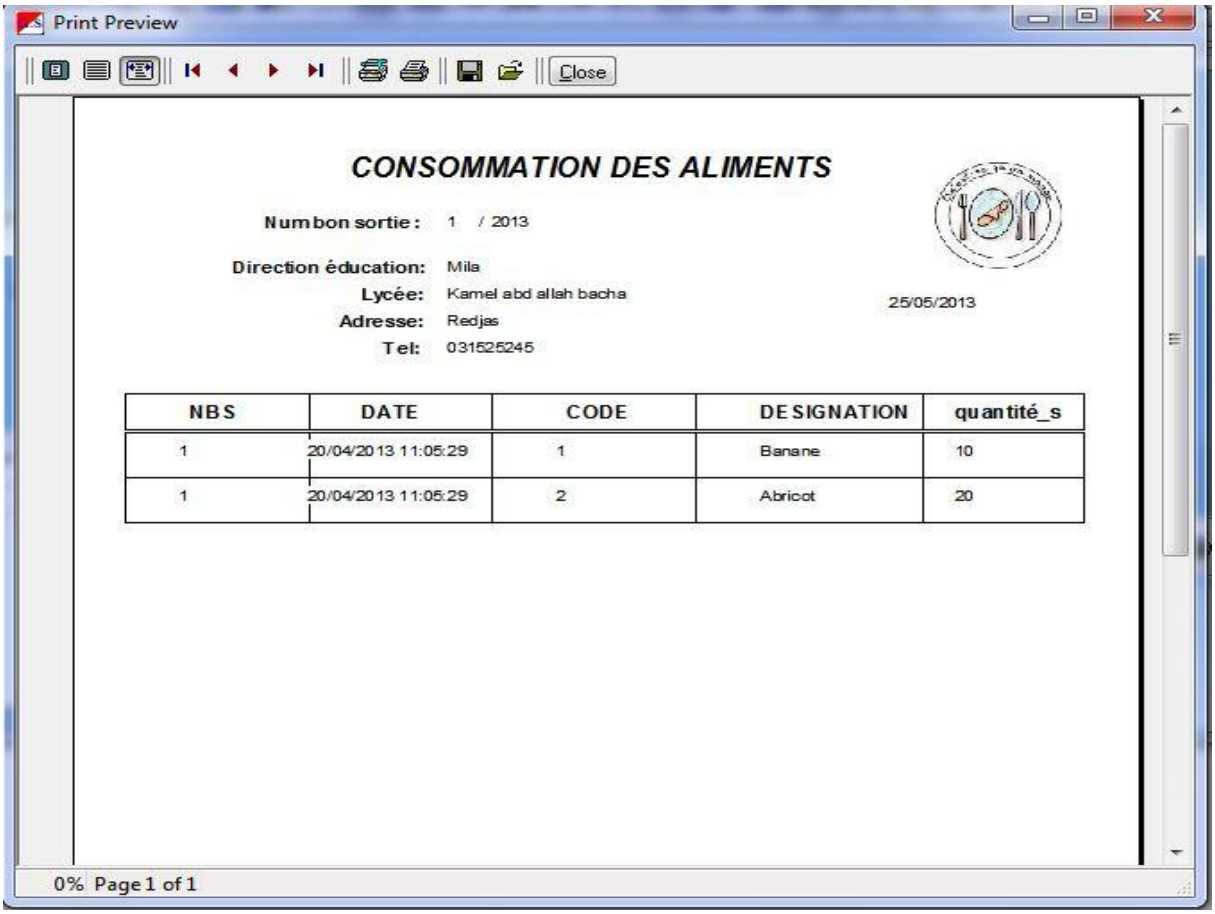
5.4.11 Interface établir menu des repas :

5.4.12 Interface informations administratives :

5.4.13 Interface imprimer bon de commande :



5.4.14 Interface imprimer consommation des aliments :



Conclusion générale

Notre projet s'articule autour de la conception et la réalisation d'un système d'information pour la gestion de réfectoire.

Pour mieux comprendre le fonctionnement du système, on a entamé sa modélisation avant sa réalisation. Nous avant appliquer les règles de base permettant d'avoir une application performante. En effet, nous avons utilisé UML comme un langage de modélisation. Pour l'implémentation, nous avons porté notre choix respectivement sur le langage Delphi et le système de gestion de base de données Access 2010.

Nous avons pu produire une application qui peut offrir à ce stade les principaux services d'un système de gestion de réfectoire. L'application n'est pas encore à sa phase finale, elle reste ouverte à toute évolution. Néanmoins, elle assiste le magasinier dans la gestion de son réfectoire.

La période passée au développement de notre application, nous a été d'un apport considérable. En effet, c'est une expérience qui nous a permis d'enrichir nos connaissances dans de domaines très variés comme: L'Orienté Objet, l'UML, le langage Delphi, le SGBD Access ...etc. Elle nous à permis aussi de découvrir un domaine en pleine évolution qui est le cœur de toutes activités dans le réfectoire.

BIBLIOGRAPHIE

Livres

- UML 2. Laurent AUDIBERT
- UML 2. Par la pratique .Pascal ROQUES
- Le champion programmation Delphi 7.Mc Belaid
- Formation multimédia a la programmation Delphi . Mc Belaid

Sites web

- www. Wikipédia.com
- www.Delphifr.com
- www.Vbfrance.com
- www.codes-sources.com