1. ***: CADRE THEORIQUE***
2. ***PRESENTATION DU SUJET***
3. ***PROBLEMATIQUE***

La maitrise de l’information est primordiale pour tout secteur ainsi que celui l’élevage. Avec le logiciel de gestion de troupeau, vous pourrez gérer facilement le suivi administratif et technique de votre élevage grâce à son ergonomie moderne et intuitive. Cette plateforme a pour solution de piloter votre élevage, le planning pour identifier rapidement les vaches à surveiller (vêlage, prête à la reproduction), un suivi sanitaire simplifié avec des plans de traitements et alertes sanitaires (traitement du jour, rémanences, rappels de vaccins) ainsi que le suivi du potentiel de production de vos vaches. L’embouche bovine se justifie par la nécessité de lever un certain nombre de contraintes afin de booster la production et la productivité de la sous filière bovine.

Pour pallier aux problèmes que rencontre la ferme, n’est-il possible de suivre l’état de santé du bovin ?

Un plan de production sera-t-il nécessaire pour une bonne gestion de ferme ?

Est-il évident d’établir le compte de résultats du cycle de production avec plusieurs hypothèses de prix de vente ?

1. ***EXPLICATION DU SUJET***

Le sujet est intitulé «Etude et mise en place d’une plateforme de gestion et suivi de ferme». Un système de gestion et suivi de ferme est défini dans ce contexte comme la gestion de bovins ainsi que leur suivi et y compris la mise en vente des produits de ces derniers.

Afin d’offrir des fonctionnalités aux utilisateurs, il doit disposer d’une interface d’utilisation permettant son exploitation. L’interface sera accessible via un service web.

Comme susmentionné dans la problématique, le programme devra aussi permettre d’une part la gestion de la reproduction pour atteindre l’objectif d’une naissance par an chez les bovins et d’autre part de la maitrise de la santé des troupeaux par un contrôle des pathologies endémiques et/ou émergentes pour le système extensif.

1. ***OBJECTIF DU TRAVAIL***

#### **Objectif principal**

L’objectif final de ce mémoire est d’offrir à toute ferme ou à toute personne utilisant le moyen pour faire des commandes de produits de la ferme, un outil de sa gestion également. L’interface Web devra leur permettre de visualiser et faire des achats, à partir d’un réseau distant, en temps réel, l’état de santé des animaux, ainsi que le chiffre d’affaire du terrain. L’interface se fera simple et intuitive pour permettre un accès rapide à l’information.

#### **Objectif spécifique**

Pour atteindre l’objectif principal nous nous sommes assigné les objectifs intermédiaires suivants :

* Suivre l’évolution du bovin ainsi que son état de santé
* Mettre en valeur les produits de l’élevage
* Faire des commandes en ligne
* Obtenir des rapports des activités sur le terrain
* Avoir les chiffres d’affaire

Dans l’étude de notre document les questions financières seront abordées, et nous parlerons aussi de la simplicité des infrastructures proposées afin de minimiser les couts d’implémentation.

1. ***DELIMITATION DU CHAMP D’ETUDE***

Nous ne prétendons pas étudier l'univers entier étant un travail en informatique, il nous est difficile de nous délimiter sur le plan temporaire. Cependant dans le cadre de cette étude nous avons délimitez notre travail en deux points :

1. Délimitation spatiale

Notre champ d'investigation est la gestion de bovin et l’achat en ligne des produits de la ferme afin d’assurer la bonne gestion de cette dernière.

1. Délimitation du point de vue matérielle (Contrainte)

Cette plateforme qui est un outil de gestion de notre ferme, d’achat en ligne ainsi que de communication disponible 24h/24, 7j/7, ouverte. Notre application sera accessible pour tout le monde puisqu’elle sera déployée en ligne.

1. ***: CADRE METHODOLOGIQUE***
2. ***METHODE ET TECHNIQUE UTILISEES***
   1. ***METHODE***

La méthode est une démarche intellectuelle qui permet à l'auteur d'atteindre ses objectifs. Dans le cadre de notre étude, nous nous appuierons sur la conduite de projets informatiques en procédant à la modélisation basée sur le Processus Unifié (UP) qui est un processus du Langage de Modélisation Unifié (Unified Modeling Language) UML en sigle.

* 1. ***TECHNIQUES***

La technique est l'outil pratique de la méthode ou encore elle est l'instrument qui permet à la méthode de collecter les informations nécessaires. En ce qui concerne les techniques utilisées, nous avons utilisé les techniques suivantes pour la collection des informations :

* + 1. **Technique d’observation directe**

Partant de cette technique, nous avons récolté quelques informations en discrétion, dans certaines fermes et avions fait majoritairement le même constat : difficulté à gérer les bovins et les vendre facilement.

* + 1. **Technique d’interview**
    2. **Technique des questionnaires**

Ces personnes ont pu répondre au questionnaire élaboré et leurs réponses nous ont permis de recueillir des informations riches et spontanées, plus brutes et plus fraiches nécessaires à notre travail. Avec cette technique, nous avons utilisé les questionnaires ouverts et les questionnaires fermés.

1. **DIFFICULTES RENCONTREES**

**DEUXIEME PARTIE :**

**ANALYSE FONCTIONNELLE ET ETUDE CONCEPTUELLE**

**CHAPITRE 1 : *CONCEPTS ET METHODES***

1. ***LES CONCEPTS DE LA PROGRAMMATION ORIENTE OBJET***

La **programmation orientée objet (POO)**, ou **programmation par objet**, est un [paradigme](https://fr.wikipedia.org/wiki/Paradigme_(programmation)) de [programmation informatique](https://fr.wikipedia.org/wiki/Programmation_informatique) élaboré par les Norvégiens [Ole-Johan Dahl](https://fr.wikipedia.org/wiki/Ole-Johan_Dahl) et [Kristen Nygaard](https://fr.wikipedia.org/wiki/Kristen_Nygaard) au début des années 1960 et poursuivi par les travaux de l'Américain [Alan Kay](https://fr.wikipedia.org/wiki/Alan_Kay) dans les années 1970. Il consiste en la définition et l'interaction de briques logicielles appelées [*objets*](https://fr.wikipedia.org/wiki/Objet_(informatique)) ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ; l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes. Dès lors, l'étape de modélisation revêt une importance majeure et nécessaire pour la POO. C'est elle qui permet de transcrire les éléments du réel sous forme virtuelle.

Orthogonalement à la programmation par [objet](https://fr.wikipedia.org/wiki/Objet_(informatique)), afin de faciliter le processus d'élaboration d'un programme, existent des méthodologies de développement logiciel objet, dont la plus connue est le [processus unifié](https://fr.wikipedia.org/wiki/Processus_unifi%C3%A9) (« *Unified Software Development Process* » en anglais), qui utilisent des langages de modélisation tels que le [*Unified Modeling Language* (UML)](https://fr.wikipedia.org/wiki/UML_(informatique)).

Même s'il est possible de concevoir par objets une application informatique sans utiliser des outils logiciels dédiés, ces derniers facilitent de beaucoup la conception, la maintenance, et la productivité. On en distingue plusieurs sortes :

* les langages de programmation ([Java](https://fr.wikipedia.org/wiki/Java_(langage)), [C#](https://fr.wikipedia.org/wiki/C_Sharp), [VB.NET](https://fr.wikipedia.org/wiki/Visual_Basic), [Python](https://fr.wikipedia.org/wiki/Python_(langage)), [Ruby](https://fr.wikipedia.org/wiki/Ruby), [C++](https://fr.wikipedia.org/wiki/C%2B%2B),  [PHP](https://fr.wikipedia.org/wiki/PHP));
* les outils de modélisation qui permettent de concevoir sous forme de schémas semi-formels la structure d'un programme (UMLDraw, Rhapsody, DBDesigner…)
* les [ateliers de génie logiciel](https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel) ou [AGL](https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel) ([Visual Studio](https://fr.wikipedia.org/wiki/Visual_Studio), [NetBeans](https://fr.wikipedia.org/wiki/NetBeans) ou [Eclipse](https://fr.wikipedia.org/wiki/Eclipse_(projet))  etc)
  1. ***LES CONCEPTS DE BASE DU MODELE OBJET***

L’approche orientée objet considère le logiciel comme une collection d’objets, leurs traitements dissociés et associés définis par des propriétés. Une propriété est attribut dont la valeur peut intervenir sur l’aspect comportemental de l’objet. La fonctionnalité du logiciel émerge alors que l’interaction entre les différents objets qui le constituent. L’une des particularités de cette approche est qu’elle rapproche les données et les traitements associés au sein d’un unique objet. La modélisation orientée objet est basée essentiellement sur trois concepts dont :

* + - * L’encapsulation
      * L’héritage
      * Le polymorphisme

La programmation Orienté Objet (POO) est un style de programmation structurant le programme en un ensemble d’objets ayant des caractéristiques, des comportements et pouvant interagir. Les acteurs du problème sont identifiés et transformés en classes. Une classe est le moule à partir duquel les objets seront générés. Les caractéristiques et les comportements sont nommés respectivement ‘attributs’ et ‘méthodes’.

* 1. **MODELISATION ET LANGAGE DE DEVELOPPEMENT OBJET**

Les modèles à objets, encore appelés modèles orientés objets ou simplement modèles objet, sont issus des réseaux sémantiques et des langages de programmation orientés objets. Ils regroupent les concepts essentiels pour modéliser de manière progressive des objets complexes encapsulés par des opérations de manipulation associées. Ils visent à permettre la réutilisation de structures et d’opérations pour construire des entités plus complexes.

Les modèles de données à objets ont été créés pour modéliser directement les entités du monde réel avec un comportement et un état. Le concept essentiel est bien sûr celui d’objet. Il n’est pas simple à définir car composite, c’est-à-dire intégrant plusieurs aspects. Dans un modèle objet, toute entité du monde réel est un objet, et réciproquement, tout objet représente une entité du monde réel.

1. **CHOIX DE LA METHODE D’ANALYSE ET DE CONCEPTION**

En [ingénierie](https://fr.wikipedia.org/wiki/Ing%C3%A9nierie), une **méthode d'analyse et de conception** est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

La modélisation est préconception. Elle permet en général de modéliser ou présenter, l’organisation de la conception, des processus, des relations entre objets.

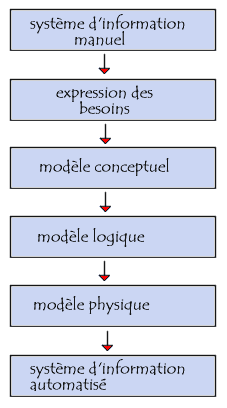
* 1. **PRESENTATION**
* **MERISE :**

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

La méthode MERISE date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'Industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information.  Les deux principales sociétés ayant mis au point cette méthode sont le CTI (Centre Technique d'Informatique) chargé de gérer le projet, et le CETE (Centre d'Etudes Techniques de l'Equipement) implanté à Aix-en-Provence. La conception du système d'information se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. Il s'agit donc de valider une à une chacune des étapes en prenant en compte les résultats de la phase précédente. D'autre part, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitements afin de vérifier que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données superflues.

Cette succession d'étapes est appelée *cycle d'abstraction pour la conception des systèmes d’information :*



L'expression des besoins est une étape consistant à définir ce que l'on attend du système d'information automatisé, il faut pour cela :

* faire l'inventaire des éléments nécessaires au système d'information
* délimiter le système en s'informant auprès des futurs utilisateurs

Cela va permettre de créer le **MCC** (Modèle conceptuel de la communication) qui définit les flux d'informations à prendre en compte. L'étape suivante consiste à mettre au point le **MCD** (Modèle conceptuel des données) et le **MCT** (Modèle conceptuel des traitements) décrivant les règles et les contraintes à prendre en compte. Le modèle organisationnel consiste à définir le **MOT** (Modèle organisationnel des traitements) décrivant les contraintes dues à l'environnement (organisationnel, spatial et temporel).

* **UML :**

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling Language* (**UML**), est un [langage](https://fr.wikipedia.org/wiki/Langage) de modélisation graphique à base de [pictogrammes](https://fr.wikipedia.org/wiki/Pictogramme) conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en [développement logiciel](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_logiciel) et en [conception orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet).

L'UML est le résultat de la fusion de précédents langages de modélisation objet : [Booch](https://fr.wikipedia.org/wiki/M%C3%A9thode_Booch), [OMT](https://fr.wikipedia.org/wiki/Object_Modeling_Technique), [OOSE](https://fr.wikipedia.org/wiki/OOSE). Principalement issu des travaux de [Grady Booch](https://fr.wikipedia.org/wiki/Grady_Booch), [James Rumbaugh](https://fr.wikipedia.org/wiki/James_Rumbaugh) et [Ivar Jacobson](https://fr.wikipedia.org/wiki/Ivar_Jacobson), UML est à présent un standard adopté par l'[Object Management Group](https://fr.wikipedia.org/wiki/Object_Management_Group) (OMG). UML 1.0 a été normalisé en janvier 1997; UML 2.0 a été adopté par l'OMG en juillet 2005[1](https://fr.wikipedia.org/wiki/UML_(informatique)#cite_note-1).La dernière version de la spécification validée par l'OMG est UML 2.5.1 (2017).

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

* Activité d'un objet/logiciel
* Acteurs
* Processus
* Schéma de base de données
* Composants logiciels
* Réutilisation de composants

Grâce aux outils de modélisation UML, il est également possible de générer automatiquement tout ou partie du code d'une application logicielle, par exemple en langage [Java](https://fr.wikipedia.org/wiki/Java_(langage)), à partir des divers documents réalisés. La version actuelle, UML 2.5, propose 14 types de diagrammes dont 7 structurels et 7 comportementaux. A titre de comparaison, UML 1.3 comportait 25 types de diagrammes.

UML n'étant pas une méthode, l'utilisation des diagrammes est laissée à l'appréciation de chacun. Le [diagramme de classes](https://fr.wikipedia.org/wiki/Diagramme_de_classes) est généralement considéré comme l'élément central d'UML. Des méthodes, telles que le [processus unifié](https://fr.wikipedia.org/wiki/Processus_unifi%C3%A9) proposé par les créateurs originels de UML, utilisent plus systématiquement l'ensemble des diagrammes et axent l'analyse sur les [cas d'utilisation](https://fr.wikipedia.org/wiki/Cas_d%27utilisation) (« use case ») pour développer par itérations successives un modèle d'analyse, un modèle de conception, et d'autres modèles. D'autres approches se contentent de modéliser seulement partiellement un système, par exemple certaines parties critiques qui sont difficiles à déduire du code.

UML se décompose en plusieurs parties :

* Les *vues* : ce sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet.
* Les *diagrammes* : ce sont des ensembles d'éléments graphiques. Ils décrivent le contenu des vues, qui sont des notions abstraites. Ils peuvent faire partie de plusieurs vues.
* Les *modèles d'élément* : ce sont les éléments graphiques des diagrammes.
  1. **ETUDE COMPARATIVE DES APPROCHES MERISE ET UML**

Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions. Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires... Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes.

Dans UML les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction...

La méthode Merise ressemble à la méthode UML pour la phase de modélisation de la base de données. La différence principale est que Merise est une méthode d'analyse, et UML un langage de modélisation de données .UML ne propose pas de cycle précis : les organisations sont libres de choisir le cycle qui leur convient.

UML fonctionne sur un principe d’itérations qui ne s’oppose pas aux phases définies dans

MERISE. MERISE découpe plus au travers de ses phases l’analyse métier et l’architecture logicielle. Dans UML, l’architecture logicielle a une place prépondérante et est intégrée très en amont dans l’élaboration du système d’information.

Dans UML, l’avancement du projet est mesuré par le nombre de cas d’utilisation, de classes... réellement implantées et non par la documentation produite ce qui est le cas dans

Merise. Les itérations servent en outre à répartir l’intégration et les tests tout au long du processus d’élaboration du système d’information.

* 1. **CHOIX ET JUSTIFICATION**

Merise et UML ont des caractéristiques voisines au niveau de la modélisation des bases de données mais également des points de divergence.

En effet, la méthode MERISE nécessite une démarche par étape qui favorise la qualité de chaque modèle avec ses différents niveaux de validations. Alors que le langage UML n’impose pas de méthode de travail particulière.

MERISE présente l'intérêt d'avoir des modèles logiques moins détaillés facilement compréhensibles. Tandis qu'UML conçu pour s'adapter à n'importe quel langage de programmation orientée objet (POO), présente plusieurs modèles (diagrammes) dont leurs compréhensions nécessitent une grande attention.

MERISE est moins préférable. Malgré sa clarté, il manque une précision du fait qu'elle est éloignée du langage donc difficile à implémenter alors qu'UML intègre les éléments communs des différents langages, sa volonté est d'être fidèle à la réalisation finale. Elle est beaucoup plus complète avec ses différents diagrammes.

Pour en finir avec l'exploitabilité, MERISE est une méthode plus généraliste. Elle donne une vue globale de la solution sans autant entrer dans les petits détails. Contrairement à UML qui est conçu pour l'implémentation objet avec ses différents détails et sa portabilité (s'adapte à n'importe quelle plateforme) elle est donc plus exploitable.

L'une ou l'autre présente des avantages et des inconvénients. Il est réservé au concepteur de choisir la méthode la mieux adaptée pour son cas. Si on cherche la précision et l'exploitabilité UML devance MERISE. Tandis que, si c'est la clarté et l'accessibilité qui sont en question MERISE est préférable.

Dans mon cas, on va gérer des données plus complexes, donc il est préférable d’utiliser UML, vu qu’elle rend la modélisation plus simple à implémenter.

1. **FORMALISME DES DIFFERENTS DIAGRAMMES**
   1. **DIAGRAMME DE CAS D’UTILISATION**

Les **diagrammes de cas d'utilisation** (**DCU**) sont des [diagrammes UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language#Les_diagrammes) utilisés pour donner une vision globale du comportement fonctionnel d'un système [logiciel](https://fr.wikipedia.org/wiki/Logiciel). Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les [cas d'utilisation](https://fr.wikipedia.org/wiki/Cas_d%27utilisation) sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).

[UML](https://fr.wikipedia.org/wiki/Unified_Modeling_Language) définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas de standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique peut seulement donner une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les **diagrammes de cas d'utilisation** sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation.

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système ! Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les *use case* (cas d'utilisation) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur et un cas d'utilisation sont mis en relation par une association représentée par une ligne.

Le plus souvent, le diagramme des cas est établi par la [maîtrise d'ouvrage](https://fr.wikipedia.org/wiki/Ma%C3%AEtrise_d%27ouvrage) (MOA) d'un projet lors de la rédaction du cahier des charges afin de transmettre les besoins des utilisateurs et les fonctionnalités attendues associées à la [maîtrise d'œuvre](https://fr.wikipedia.org/wiki/Ma%C3%AEtrise_d%27%C5%93uvre) (MOE).

* 1. **DIAGRAMME DE SEQUENCES**

Un diagramme de séquence est un type de diagramme d'interaction, car il décrit comment et dans quel ordre plusieurs objets fonctionnent ensemble. Ces diagrammes sont utilisés à la fois par les développeurs logiciels et les managers d'entreprises pour analyser les besoins d'un nouveau système ou documenter un processus existant. Les diagrammes de séquence sont parfois appelés diagrammes d'événements ou scénarios d'événements.

Les diagrammes de séquence peuvent constituer des références utiles pour les entreprises et d'autres organisations. Essayez de dessiner un diagramme de séquence pour :

* Représenter les détails d'un cas d'utilisation UML
* Modéliser le déroulement logique d'une procédure, fonction ou opération complexe
* Voir comment les objets et les composants interagissent entre eux pour effectuer un processus.
* Schématiser et comprendre le fonctionnement détaillé d'un scénario existant ou à venir.
  1. **DIAGRAMME DE CLASSES**

Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles, car ils décrivent clairement la structure d’un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets. Le langage UML a été créé sous forme de modèle standardisé pour décrire une approche de la programmation orientée objet. Comme les classes sont les composantes des objets, les diagrammes de classes sont les composantes de l'UML. Les divers éléments d'un diagramme de classes peuvent représenter les classes qui seront effectivement programmées, les principaux objets ou les interactions entre classes et objets.

La forme de la classe à proprement parler se compose d'un rectangle à trois lignes. La ligne supérieure contient le nom de la classe, celle du milieu affiche les attributs de la classe et la ligne inférieure exprime les méthodes ou les opérations que la classe est susceptible d'utiliser. Les classes et sous-classes sont regroupées pour illustrer la relation statique entre chaque objet. Les diagrammes de classes présentent de nombreux avantages pour n'importe quel type d'organisation. Vous pouvez les utiliser pour :

* Illustrer des modèles de données pour des systèmes d’information, quel que soit leur degré de complexité.
* Mieux comprendre l’aperçu général des schémas d’une application.
* Exprimer visuellement les besoins d'un système et diffuser cette information dans toute l'entreprise.
* Créer des schémas détaillés qui mettent l'accent sur le code spécifique qui doit être programmé et mis en œuvre dans la structure décrite.
* Fournir une description indépendante de l'implémentation des types utilisés dans un système, qui sont ensuite transmis entre ses composants.
  1. **DIAGRAMME DE DEPLOIEMENT**

Dans le contexte du langage de modélisation unifié (UML), un diagramme de déploiement fait partie de la catégorie des diagrammes structurels, car il décrit un aspect du système même. Dans le cas présent, le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. On appelle artefact l'information qui est générée par le logiciel.

Les diagrammes de déploiement sont constitués de plusieurs formes UML. Les boîtes en trois dimensions, appelées nœuds, représentent les composants du système, qu'ils soient logiciels ou matériels. Les lignes entre les nœuds indiquent les relations et les petites formes à l'intérieur des boîtes représentent les artefacts logiciels qui sont déployés. Ils sont utiles dans plusieurs domaines. Vous pouvez les utiliser pour :

* Montrer quels éléments logiciels sont déployés par quels éléments matériels.
* Illustrer le traitement d'exécution du point de vue matériel
* Visualiser la topologie du système matériel
  1. **CHAPITRE II : CONCEPTION ET DEPLOIEMENT**
  2. **ANALYSE DU SYSTEME**
  3. **LES BESOINS DES UTILISATEURS**

Au sortir de notre collecte d’informations au sein de la structure, nous avons recensé les problèmes suivants, rencontrés par chacun des acteurs des fermes :

* On note tout d’abord les problèmes de santé des bovins de la ferme, ce qui pose un problème à la ferme et même la mort des animaux.
* On a eu à rencontrer de problèmes de la mauvaise gestion de la ferme. Les animaux ne sont pas bien suivis et retarde l’évolution de cette dernière.
* Les produits de l’élevage de la ferme se vendent difficilement. On n’a pas de la clientèle.

Ainsi, nous avons résumé les besoins suivants:

* Instaurer un système performant d’archivage des données de suivi des animaux sur l’exploitation, ce qui facilitera le travail rétrospectif.
* Informer à temps le vétérinaire de toutes anomalies remarquées dans la ferme.
* Organiser les ateliers de formation des bouviers dans le but de leur montrer l’importance et l’impact de leur travail dans la chaîne de production laitière.