# **ANALYSE ET CONCEPTION**

## **ANALYSE PREALABLE**

### **Analyse de l’existant**

#### **Organisation actuelle**

La figure x montre l’organigramme de la direction des Systèmes Informatiques montrant la branche d’étude.

Figure Organigramme de la DSI montrant la branche d’etude

#### **Mode de traitement des données**

Les informations traitées sont obtenues à partir d’un fichier log qui est créé sur chaque site à partir d’un script planifié à une heure précise tous les jours. Le responsable se connecte par accès à distance sur chaque site.

Il envoie par la suite, un résumé sous un fichier Excel par email à l’administrateur Système Agences et Usines chargé de l’audit qui en analysant les informations inscrites décide des actions à faire. Comme exemple allouer une nouvelle taille pour les tablespaces qui ont dépassés le seuil maximal pour les tablespaces ou de prévoir l’approvisionnement de nouvelles espaces de stockages pour les serveurs…

#### **Les applications existantes**

Les applications existantes sont pour le moment le script chargé de recueillir les informations depuis chaque site et de les écrire dans un fichier log et l’application Microsoft Office Excel pour la présentation des informations aux administrateurs Système Agences et Usines.

#### **Inventaire des moyens matériels et logiciels**

Matériels utilisés par les responsables

### **Critique de l’existant**

L’organisation du Groupe STAR comprend en tout 25 sites dans tout Madagascar dont font partie : le siège ainsi que **18 agences** et **6 usines**. Or, on doit les administrateurs Système Agences et Usines doivent être au courant de l’état de l’espaces de stockage de chaque serveur et de l’état de tablespaces ajouté aux objets rattachées (tables, index).

Ceci dit, se connecter par accès à distance à chaque site prendrait beaucoup de temps et augmenterait les tâches du chargé de contrôle et d’envoie des fichiers logs. Ces actions de contrôle et d’envoie doivent par ailleurs fait à la fin de la journée. Ce qui implique des tâches hors de l’heure de travail pour les personnes concernées.

Naviguer dans les fichiers logs implique aussi beaucoup de précisions pour ne pas perdre les informations car le contrôle est fait visuellement.

### **Conception avant-projet**

#### **Proposition de solutions**

Face à ces contraintes, on a alors proposé les solutions suivantes :

1. Utiliser des logiciels spécifiques pour l’audit des tablespaces comme : **« Manage Engine Application Manager d’ORACLE ».**
2. Acheter un logiciel sur mesure auprès d’une SSII.
3. Automatiser les taches directement pour l’alimentation vers la base de donnée à l’aide d’un script et permettre de visualiser ces informations.

Choix 1 – Fanazavana kely anle étapes Mila reformulena

#### **Avantages et inconvénients de chaque solution**

1. Utiliser des logiciels spécifiques pour l’audit des tablespaces comme : **« Manage Engine Application Manager d’ORACLE ».**

* **Avantages :**
* Logiciel stable
* Vue globale et précis des informations des tablespaces
* **Inconvénients :**
* Nécessite connexion à chaque site et exécution de script SQL spécifique
* Ouverture de plusieurs fenêtres pour chaque site
* Manque d’informations comme les détails des disques

1. Acheter un logiciel sur mesure auprès d’une SSII.

* **Avantages :**
* Bénéfice des mises à jours
* Outil très robuste et opérationnel
* Gain de temps considérable
* Formation assurée par des professionnels
* **Inconvénients :**
* Coût du logiciel considérable
* Coût de formation considérable

1. Automatiser les taches directement pour l’alimentation vers la base de donnée à l’aide d’un script et permettre de visualiser ces informations.

* **Avantages :**
* Produit adapté aux besoins de l’utilisateur
* Vue globale de chaque site
* Logiciel pour le bénéfice de la Star sans dépense financière
* Code source modifiable
* **Inconvénients :**
* Cout de temps considérable lors de la réalisation jusqu’à la mise en place
* Processus incrémental et itératif
* Difficulté de maintenir les mises à jours

#### **Choix de la solution et justification**

On a opté pour la dernière solution car la conception et la réalisation d’une application sur mesure répond pleinement aux besoins des futurs l’utilisateur. Pour ce faire, on a choisi de développer une application d’alimentation de données et d’un autre destinés à les afficher à partir d’un interface Web, nous avions aussi besoins d’une méthode de conception, d’un langage de programmation et d’un outil de développement. Dans le cadre de ce stage, cette dernière solution nous correspond parfaitement.

#### **Méthodologie**

##### **Méthode utilisé**

Pour modéliser d’une manière claire et précise la structure et le comportement de notre système

Indépendamment de tout langage de programmation nous nous allons adopter la démarche 2TUP et le langage de modélisation UML.

Justification any anaty boky bandy L3

##### **Système de Gestion de Base de Données**

En informatique, un système de gestion de base de données (SGBD) est logiciel système destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations. [13].

Les plus utilisées sont les systèmes de gestion de base de données relationnelle client-serveur. Le serveur est une application installée sur l’ordinateur où sont stockées les données, il attend des requêtes des clients, accède à la base de données pour exécuter la requête du client et lui fournit une réponse. Le client est un programme installé sur l’ordinateur qui se connecte par intermédiaire d’un réseau au serveur pour effectuer une requête.

Les plus connus sont:

* **MySQL** : libre et gratuit, c'est probablement le SGBD le plus connu.
* **PostgreSQL** : libre et gratuit comme MySQL, avec plus de fonctionnalités mais un peu moins connu ;
* **SQLite:** libre et gratuit, très léger mais très limité en fonctionnalités ;
* **Oracle** : utilisé par les très grosses entreprises ; sans aucun doute un des SGBD les plus complets, mais il n'est pas libre et on le paie le plus à un prix très coûteuse ;
* **Microsoft SQL Server** : le SGBD de Microsoft.

**Choix et justification**

Dans ce projet, on a choisi d’utiliser le SGBD ORACLE Database 11 G pour le stockage des données parce que c’est le SGBD utilisé par le Groupe STAR mais aussi ORACLE Database fait l’objet de ce mémoire.

[Oracle Database](https://fr.wikipedia.org/wiki/Oracle_Database) est un SGBD relationnel le plus employés au monde. Il fût créé en 1979 et son langage de requête est nommé [PL/SQL](https://fr.wikipedia.org/wiki/fr:PL/SQL)  basé sur les paradigmes de programmation [procédurale](https://fr.wikipedia.org/wiki/Programmation_proc%C3%A9durale) et [structurée](https://fr.wikipedia.org/wiki/Programmation_structur%C3%A9e). Il permet de combiner des requêtes [SQL](https://fr.wikipedia.org/wiki/Structured_Query_Language) et des instructions procédurales (boucles, conditions...), dans le but de créer des traitements complexes destinés à être stockés sur le serveur de base de données (objets serveur), comme des [procédures stockées](https://fr.wikipedia.org/wiki/Proc%C3%A9dure_stock%C3%A9e) ou des [déclencheurs](https://fr.wikipedia.org/wiki/D%C3%A9clencheur). [Oracle]

Tout programme PL/SQL doit se présenter sous forme de blocs. Voici la forme générale d'un bloc.

<<label>> *-- Optionnel*

**DECLARE** *-- Section optionnelle*

déclarations

**BEGIN** *-- Section obligatoire, doit contenir au moins une instruction exécutable*

implémentation

**EXCEPTION** *-- Section optionnelle*

gestion des exceptions

**END**;

Oracle est disponible sur [Windows](https://fr.wikibooks.org/wiki/Windows) et [Linux](https://fr.wikibooks.org/wiki/Linux), en plusieurs versions :

Les numéros de version d'Oracle ont introduit une certaine confusion chez de nombreuses personnes. La nomenclature de version a changé au cours du temps, mais se base en général sur la convention suivante : *version X.Y.Z.a* où X.Y désigne la version majeure du produit : 8.1 pour 8i, 9.2 pour 9i Release 2, etc. Ceci n'est plus vrai en 10, où l'on a une version majeure (10g), déclinée en deux « *releases* ». Au sein d'une même version majeure (8i, 9i ou 10g), les fonctionnalités générales du [RDBMS](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_relationnelle) sont censées être les mêmes.

Le chiffre Z désigne quant à lui le niveau de correctif générique appliqué. Oracle délivre en général 3 ou 4 mises à jour (patch) pour une version majeure, mais cela reste très variable.

Enfin, le chiffre a désigné un niveau de correctif spécifique à un système d'exploitation, ou une plateforme.

Depuis la version 7, les versions majeures du [SGBD](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es) Oracle ont été les suivantes :

* Oracle 7 : 7.1, 7.2, 7.3
* Oracle 8 : 8.0.3, 8.0.4, 8.0.5, 8.0.6
* Oracle 8i : 8.1.5 (Release 1), 8.1.6 (Release 2), 8.1.7 (Release 3)
* Oracle 9i : 9.0.1 (Release 1), 9.2.0 (Release 2)
* Oracle 10g : 10.1 (Release 1), 10.2 (Release 2)
* Oracle 11g : 11.1 (Release 1), 11.2 (Release 2)
* Oracle 12c : 12.1 (Release 1), 12.2 (Release 2)

À partir de 2018, un rythme annuel de nouvelles versions est prévu : le numéro de version rappellera l'année, la prochaine version sera donc la 18 (au lieu de la 12.2.0.2 initialement prévue), la suivante la 19... [Oracle]

ORACLE Database gère le stockage de ses données de façon intelligente dans ce qu’on appelle « tablespaces » ou espaces logiques qui contiennent les objets tels que les tables, les index, les vues.

Une tablespace est composé d’au moins un DataFile ou fichier qui contient physiquement les données qui est présent sur le serveur à l’endroit stipulé lors de sa création. Chaque DataFile est composé de segments d’au moins un Extent ou page lui-même composé d’au moins 3 blocs, l’élément le plus petit de la Base de donnée.

##### **Langage de programmation**

La phase de developpement de la méthode

On ne pourrait pas passer à la phase de développement de l’application sans un langage de programmation.

Pour la première partie, c’est-à-dire : la phase d’alimentation, on a utilisé le WLangage spécifique à WinDev.

Pour la seconde partie, on a utilisé NodeJs pour collecter les informations et les passer aux applications clients.

Le tableau x montre une comparaison entre langage de programmation côté serveur.

|  |  |  |
| --- | --- | --- |
| **Langage** |  |  |
| **Apprentissage** | Lourd  Manque de souplesse dans les syntaxes | Souple  Approche évènementielle performante |
| **Gestion des libraires** | Nécessite une gestion des dépendances et paramétrage des serveurs d’applications | Simplicité d’intégration des librairies et d’exploitation |
| **Framework** | Difficultés liées à l’utilisation de ces Framework | Architecture logicielle ultra-modulaire |
| **Mode d’exécution et performance** | Baisse de régime voire crash du serveur lors de montées en charge | Bonnes performances, résiste à la montée en charge |
| **Maintenance et évolutivité** | Peu de nouveautés à attendre | Plateforme très dynamique, beaucoup de nouvelles fonctionnalités à venir |
| **Sécurité** | Prêt à utiliser | Possibilité de personnaliser entièrement sa stratégie sécuritaire par une gestion plus bas niveau  Nécessite la connaissance sur le sujet |

**Choix et justification**

Le choix d’un langage de programmation repose en effet sur les besoins de l’utilisateur car chaque langage a ses spécificités. En effet, Java EE est une plateforme très complète, fiable et éprouvée, qui offre des facilités de développement, mais peu de souplesse. Ses performances sont bonnes, à nuancer en cas de montée de charge. On notera également la complexité des paramétrages et l’âge de la technologie, qui laisse espérer peu de nouveautés à venir.

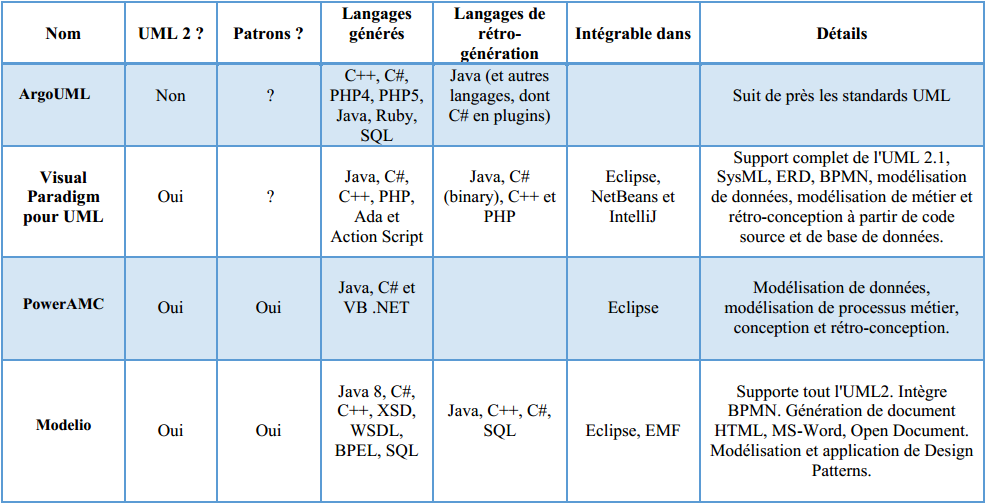
Mais de son côté, Nodejs est une plateforme innovante, dynamique, souple et performante, qui permet une architecture logicielle modulaire. Le développement événementiel côté serveur répond parfaitement aux besoins exprimés. Nodejs est aussi plus adapté à la gestion des requêtes multiples et possède aussi une implémentation stable d’ORACLE Database. La création d’API est d’autant plus simplifiée pour permettre un développement rapide de l’application.

##### **Outil de modélisation**

Pour bien implémenter la phase de modélisation, il nous est utile durant le cours de notre projet d’utiliser un outil de modélisation pour permettre la création des diagrammes UML et des modèles qui en sont à l’origine.

Le tableau x présente la comparaison entre quelques outils de modélisation.

Tableau x: Comparaison de quelques outils de modélisation



**Choix et justification**

Nous avons opté pour Visual Paradigm durant notre projet car celui-ci permet de générer du code dans un langage de programmation déterminé afin de faciliter le développement. Il propose également la création d’autres types de diagrammes, comme celui qui permet la modélisation des bases de données pouvant, lui aussi, générer des canevas d’applications basé sur des Framework et Pattern mais en plus, générer du code SQL qu’il peut ensuite déployer automatiquement dans différents environnements.

## **ANALYSE CONCEPTUELLE**

### **Présentation de la méthode utilisée**

Processus de développement :

Les processus unifiés :

Un processus unifié est un processus construit sur UML (Unified Modeling Language). Plusexactement ce sont les meilleures pratiques du développement objet

suivies pour la réalisation d’un

système.Un processus unifié se distingue par les caractéristiques suivantes (ROQUES, 2008) :



Itératif : Le logiciel nécessite une compréhension progressive du problème à travers desraffinements successifs et développer une solution effective de façon  incrémentale par desitérations multiples.



Piloté par les risques : les causes majeures d’échec d’un projet logiciel doivent être écartées

en priorité.



Centré sur l

’architecture : le choix de l’architecture logicielle est

effectué lors des premièresphases de développement du logiciel. La conception des composants du système est baséesur ce choix.



Conduit par les cas d’utilisation : le processus est orienté par les

besoins utilisateurs

présentés par des cas d’utilisation.

 Dans la communauté objet et nouvelles technologies ils existent plusieurs processus unifiés envogue comme eXtreme Programming (XP) et Rational Unified Process (RUP). Dans notre étudeon a choisi

de travailler avec le processus 2TUP parce qu’il ci

ble des projets de toute taille, et il apu faire une large place dans le domaine de la technologie et les risques des projets.

Le processus 2TUP :

Le processus 2TUP (Two Track Unified Process) est un processus unifié. Il gère la complexitétechnologique en donnant  part à la technologie dans son processus de développement

(Franck,2004)

.Le 2TUP propose un cycle de développement qui sépare les aspects techniques des aspectsfonctionnels et propose une étude parallèle des deux branches : fonctionnelle (étude de

l’application) et la technique (étude de l’implémentation). Illustré

dans(

Figure 2

), le processus

2TUP s’articule autour de trois

branches :



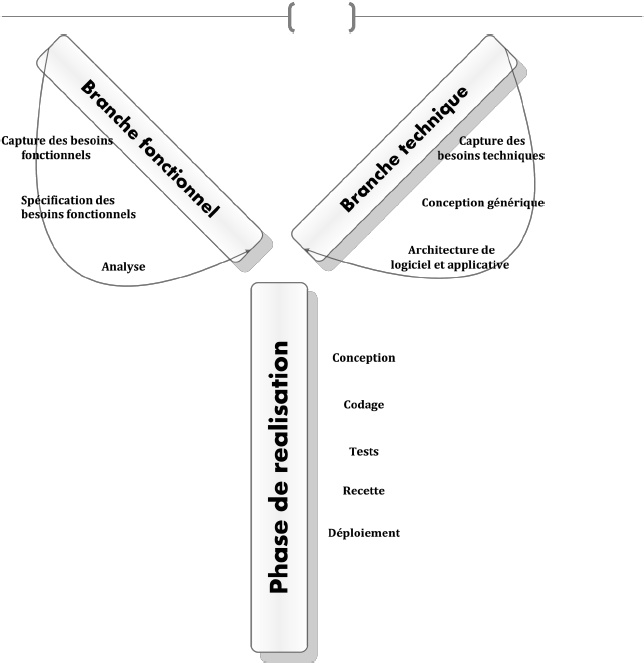
Une branche technique



Une branche fonctionnelle



Une branche de conception et réalisation. La figure suivante détaille les étapes de développement des  trois branches du processus 2TUP.

****

1.2.1 Branche fonctionnelle

: Les principales étapes de la branche fonctionnelle se présentent comme suit :

L’étape capture des besoins fonctionnels

: Cette phase a pour objectif de définir :

o

La frontière fonctionnelle entre le système et son environnement.

o

Les activités  attendues des différents utilisateurs  par rapport au système.

L’étape d’analyse

:

consiste à étudier précisément les spécifications fonctionnelles de manièreà obtenir une idée de ce que va réaliser le système en terme de métier.

Branche technique

Les principales étapes de la branche technique se présentent comme suit :

L’étape capture des besoins techniques

: Cette étape recense toutes les contraintes sur leschoix de technologies pour la conception du système. Les outils et le matériel sélectionnés ainsi

que la prise en compte des contraintes d’intégration avec l’existant (pré requis d’architecture

technique).

L’étape conception générique

: Définit les composants nécessaires à la construction de

l’architecture technique. Cette conception est complètement indépendante des aspects

fonctionnels. Elle permet  de générer le modèle de conception technique qui définit lesFrameworks.

Phase conception - réalisation

Les principales étapes de cette branche se présentent comme suit :

L’étape conception préliminaire

: Cette étape permet de produire le modèle de conceptionsystème. Ce dernier organise le système en composants, délivrant les services techniques et fonctionnels, Ce qui induit le regroupement des informations des branches technique et fonctionnelle.

L’étap

e conception détaillée

:

permet d’étudier comment réaliser chaque composant.

le résultat

fournit l’image prête { fabriquer du système complet.

L’étape

de codage

: permet d

’

effectuer la production des composants et les tests des unités de codeau fur et à mesure de leur réalisation.

L’étape de recette

: consiste à valider les fonctionnalités du système développé.

### **Dictionnaire des données**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** | **Signification** | **Type** | **Longueur** | **Règle de calcul** | **Commentaires** |
| **DIVISION** | Code du Site | AN | 2 |  |  |
| **DIVISION\_TYPE** | Type du Site | A | 6 |  |  |
| **DIVISION\_NAME** | Nom du site | AN | 64 |  |  |
| **DATE\_AUDIT\_JOURNALIER** | Date de l’audit | DD-MM-YYYY HH24 :MI :SS | 8 |  |  |
| **CACHE\_HIT\_RATIO** | Valeur de la cache hit ratio | N |  |  | Obtenu à partir d’une requête SQL |
| **ID\_OBJET\_DETAIL** | Identifiant de l’objet détail de l’audit journalier | N | 10 expo 127 |  |  |
| **TYPE\_OBJET** | Type de l’objet de la base de données | A | 12 | **-** Tablespace  - Index  - Tables |  |
| **NOM\_OBJET** | Nom de l’objet de la base de données | A | 64 |  |  |
| **TAILLE\_OBJET** | Taille de l’objet de la base de données | N |  | **R1** | Obtenu à partir d’une requête SQL |
| **NB\_EXTENT** | Nombre d’Extent | N |  |  | Obtenu à partir d’une requête SQL |
| **TAILLE\_OBJET\_UTILISE** | Espace utilise par l’objet de la BD | N |  | **R1** | Obtenu à partir d’une requête SQL |
| **TAILLE\_OBJET\_LIBRE** | Espace libre par l’objet de la BD |  |  | **R1** |  |
| **SERVEUR** | Catégorie du serveur | AN | 2 |  |  |
| **NOM\_DISQUE** | Nom de la disque | A | 3 |  |  |
| **ESPACE\_DISQUE\_UTILISE** | Espace utilisé pour un disque | N |  | **R2** | Obtenu à partir d’une requête SQL |
| **ESPACE\_DISQUE\_TOTAL** | Espace total du disque | N |  | **R2** | Obtenu à partir d’une requête SQL |
| **ESPACE\_DISQUE\_LIBRE** | Espace libre du disque |  |  | **R2** |  |
| **TABLESPACES\_AVG\_FREE** | Moyenne de tous les espaces libres des tablespaces | N |  |  | Obtenu à partir d’une requête SQL |
| **TABLESPACES\_AVG\_USED** | Moyenne de tous les espaces utilisées des tablespaces | N |  |  | Obtenu à partir d’une requête SQL |
| **TABLESPACES\_SUM\_TOTAL\_KO** | Somme du total en Ko des tablespaces | N |  | **R3** | Obtenu à partir d’une requête SQL |
| **TABLESPACES\_SUM\_LIBRE\_KO** | Somme du total en Ko libre des tablespaces | N |  | **R4** |  |
| **TABLESPACES\_SUM\_USED\_KO** | Somme du total en Ko utilise des tablespaces | N |  | **R5** | Obtenu à partir d’une requête SQL |
| **TAILLE\_TABLE\_TOTAL\_KO** | Somme du total en Ko utilisée par les tables | N |  | **R6** | Obtenu à partir d’une requête SQL |
| **TAILLE\_INDEX\_TOTAL\_KO** | Somme du Taille total des index en Ko | N |  | **R7** | Obtenu à partir d’une requête SQL |

### **Règles de gestion**

**R1 :**

TAILLE\_OBJET = TAILLE\_OBJET\_UTILISE + TAILLE\_OBJET\_LIBRE

TAILLE\_OBJET\_UTILISE = TAILLE\_OBJET - TAILLE\_OBJET\_LIBRE

TAILLE\_OBJET\_LIBRE = TAILLE\_OBJET\_UTILISE - TAILLE\_OBJET\_UTILISE

**R2 :**

ESPACE\_ DISQUE \_TOTAL = ESPACE\_ DISQUE \_UTILISE + ESPACE\_ DISQUE \_LIBRE

ESPACE\_ DISQUE \_UTILISE = ESPACE\_ DISQUE \_TOTAL - ESPACE\_ DISQUE \_LIBRE

ESPACE\_ DISQUE \_LIBRE = ESPACE\_ DISQUE \_TOTAL - ESPACE\_ DISQUE \_UTILISE

**R3 :**

TABLESPACES\_SUM\_TOTAL\_KO = ∑ TAILLE\_OBJET

Où l’objet est de type « tablespace »

**R4 :**

TABLESPACES\_SUM\_LIBRE\_KO = ∑ TAILLE\_OBJET\_LIBRE

Où l’objet est de type « tablespace ».

**R5 :**

TABLESPACES\_SUM\_USED\_KO = ∑ TAILLE\_OBJET\_UTILISE

Où l’objet est de type « Tablespace ».

**R6 :**

TAILLE\_TABLE\_TOTAL\_KO = ∑ TAILLE\_OBJET\_UTILISE

Où l’objet est de type « Table ».

**R7 :**

TAILLE\_INDEX\_TOTAL\_KO = ∑ TAILLE\_OBJET\_UTILISE

Où l’objet est de type « Index ».

### **Représentation et spécification des besoins**

Le diagramme de cas d’utilisation est le diagramme principal du modèle UML. Il décrit la succession des opérations réalisé par un acteur (personne qui assure l’exécution d’une activité). Par conséquent, il assure la relation entre l’utilisateur et les objets que le système met en œuvre.

Le diagramme des cas d’utilisation permet de structurer les besoins des utilisateurs et les objectifs correspondants d’un système. Leur but est justement d’éviter de tomber dans la dérive d’une approche fonctionnelle, où l’on liste une litanie de fonctions que le système doit réaliser.

Un cas d’utilisation est donc une représentation d’un ensemble de séquence d’actions qui sont réalisées par le système et produisent un résultat observable intéressant pour un acteur particulier.

Pour représenter un diagramme de cas d’utilisation, on a besoin les éléments de bases suivant :

* **Acteur** : entité externe (utilisateur humain, dispositif matériel ou autre) qui agit sur le système étudié. Il peut consulter et/ou modifier directement l’état du système, en émettant et/ou en recevant des messages éventuellement porteurs de données.

**Formalisme :**

La figure 04 représente le formalisme d’un acteur

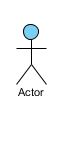


Figure 2: Formalisme d'un acteur dans UML

* **Cas d’utilisation** : suite d’interactions entre un acteur et le système. Il doit définir exhaustivement les exigences fonctionnelles du système où chaque cas d’utilisation correspond à une fonction métier du système, selon le point de vue d’un de ses acteurs.

**Formalisme :**

La figure 5 représente le formalisme d’un cas d’utilisation



Figure 3: Formalisme d'un cas d'utilisation dans UML

UML définit trois types de relations standardisées entre cas d’utilisation :

* **Relation d’inclusion** : Une relation représentée par un prototype « include » permet d’enrichir un cas d’utilisation (cas de base) par un autre cas d’utilisation (cas inclus). Le cas inclus est ajouté obligatoirement au cas de base. On utilise fréquemment cette relation pour éviter de décrire plusieurs fois le même enchainement, en factorisant le comportement commun dans un cas d’utilisation.

**Formalisme :**

La figure 6 représente le formalisme d’un cas d’utilisation.

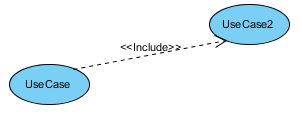


Figure 4: Formalisme d'une relation d'inclusion

* **Relation d’extension** : une relation représentée par prototype « extends » permet d’enrichir un cas d’utilisation par un autre, cependant, cet enrichissement est optionnel. Le cas de base peut fonctionner tout seul, mais il peut également compléter par un autre. On utilise principalement cette relation pour séparer le comportement optionnel du comportement obligatoire.

**Formalisme :**

La figure 7 représente le formalisme d’une relation d’extension.

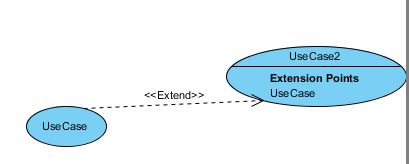


Figure 5: Formalisme d'une relation d'extension

* **Relation de généralisation/spécification** : Une relation d’héritage. La relation de généralisation entre deux entités exprime le fait que l’entité spécialisée est un cas particulier de l’entité général. L’entité spécialisée peut réaliser tout ce que l’entité général peut réaliser.

**Formalisme :**

La figure 8 représente le formalisme d’une relation de généralisation/spécialisation.

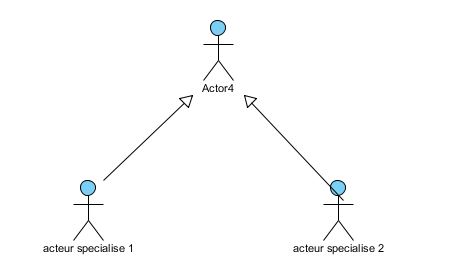


Figure 6: Formalisme d’une relation de généralisation/spécialisation

La figure suivante montre le diagramme de cas d’utilisation du nouveau système.

#### **Priorisation des cas d’utilisation**

La priorisation des cas d’utilisation permet de décrire chronologiquement les opérations.

Le tableau ci-dessous montre la liste des cas d’utilisations par leur niveau de priorité.

|  |  |
| --- | --- |
|  | **Cas d’utilisation** |
|  |  |

* **Description textuelle de quelques cas d’utilisation :**

Le tableau présente la description textuelle de chaque cas d’utilisation.

|  |  |
| --- | --- |
|  | -**Acteur :** L'administrateur  -**Description :** Ajouter un nouvel utilisateur du système  -**Démarrage**: L’agent demande la page d’ajout d’un utilisateur.  -**Précondition** :   1. L'administrateur est authentifié.   **-Scénario nominal :**   1. L’administrateur demande le formulaire d’ajout d’un nouvel utilisateur. 2. Le système retourne le formulaire d’ajout d’un nouvel utilisateur. 3. L’administrateur remplit le formulaire d’ajout. 4. L’administrateur valide et envoie le formulaire d’ajout. 5. Le système vérifie les données envoyées. 6. Le système retourne le résultat de l’ajout.   -**Scénario d’exception :**  5 .a. Les données envoyées sont invalides.  -**Fin :** Le système affiche la confirmation de l’ajout effectué.  -**Post-conditions** : Le système informe l’administrateur si  l’enregistrement a été effectué ou a échoué. |

#### **Diagramme de séquences système pour chaque cas d’utilisation**

Le diagramme de séquences permet de représenter la succession chronologique des opérations et des collaborations entre objets selon un point de vue temporel. Il peut servir à illustrer un cas d'utilisation. On y met l'accent sur la chronologie des envois de messages qui est déterminé par sa position sur l'axe vertical du diagramme, le temps s'écoule "de haut en bas" de cet axe. La disposition des objets sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme. Le diagramme de séquences est la vue dynamique les plus importantes d'UML.

* **La ligne de vie :** Une ligne de vie se représente par un rectangle, auquel est accroché une ligne verticale pointillée, qui symbolise la participation à une interaction objet ou acteur.

- **Le message :** Le message définit la communication particulière entre des lignes de vie. L’invocation d’une opération est le type de message le plus utilisé en programmation objet. L’invocation peut être asynchrone ou synchrone.

* **Messages asynchrones :** Les messages asynchrones sont des signaux. Ils n’attendent pas de réponse et ne bloquent pas l’émetteur c'est-à-dire ils n’interrompent pas l’exécution de l’expéditeur. Quand ils arriveront, ils seront traités par le destinataire.

La figure 10 montre le formalisme d’un message asynchrone.

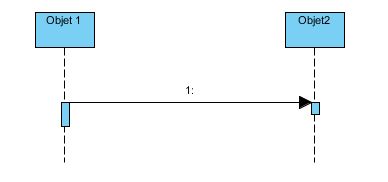


Figure 7: Formalisme d’un message asynchrone

* + **Messages synchrones :** Durant un message synchrone, l’émetteur reste bloquer jusqu’à la prise en compte du message par le destinataire. Un message synchrone se représente par une flèche en traits pleins, partant de la ligne de l'expéditeur et allant vers celle du destinataire. Ce message peut être suivi d’une réponse qui se représente par une flèche en pointillé.

La figure 11 montre le formalisme d’un message synchrone.

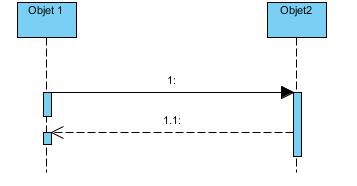


Figure 8: Formalisme d’un message synchrone

* Diagramme de séquence système « visualiser les audits »

La figure x affiche le diagramme de séquence système du cas d’utilisation

* Diagramme de séquence système « visualiser les audits »

La figure x affiche le diagramme de séquence système du cas d’utilisation

* Diagramme de séquence système « visualiser les audits »

La figure x affiche le diagramme de séquence système du cas d’utilisation

### **Spécification des besoins techniques**

Les besoins du maître d’ouvrage étant modélisés de façon générale, des spécifications techniques nous ont été aussi données :

* Une application intégrée à l’intranet disponible à partir du navigateur web et depuis plusieurs postes.
* Une vision globale des situations les plus récentes des tablespaces et des disques des chaque site Star.
* Une interface fluide et rapide.
* La possibilité de filtrer par date l’audit des tablespaces et disques.
* La possibilité de choisir le type des sites à afficher.
* La possibilité de voir les détails de l’audit pour un site.
* Un rafraîchissement périodique des informations.

### **Modèle du domaine**

Un modèle du domaine est une visualisation des concepts d’un domaine du monde réel. L’élaboration du modèle des classes du domaine permet d’opérer une transition vers une véritable modélisation objet. L’analyse du domaine est une étape totalement séparée de l’analyse des besoins. Elle peut être menée avant, en parallèle ou après cette dernière. La phase d’analyse du domaine permet d’élaborer la première version du diagramme de classe appelée modèle du domaine. Ce modèle doit définir les classes qui modélisent les entités ou concepts présents dans le domaine de l’application. Il s’agit donc de produire un modèle des objets du monde réel dans un domaine donné. Ces entités ou concepts sont identifiés directement à partir de la connaissance du domaine ou par des entretiens avec des experts du domaine. [4]

Pour établir le diagramme, on suit des étapes :

* Identifier les entités ou concepts du domaine
* Identifier et ajouter les associations et les attributs
* Organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant l’héritage
* Le cas échéant, structurer les classes en paquetage selon les principes de cohérence et d’indépendance.

La figure x représente le modèle du domaine.

## **CONCEPTION DETAILLEE**



### **Architecture du système**

L’architecture de notre application est basé sur l’architecture client/serveur dont le fonctionnement

### **Diagramme de séquence de conception pour chaque cas d’utilisation**

Le diagramme de séquence de conception est la représentation graphique des interactions entre les objets manipulés par le système selon un ordre chronologique.

* Diagramme de séquence de conception « visualiser les audits »

La figure x affiche le diagramme de séquence de conception du cas d’utilisation « x »

* Diagramme de séquence de conception « visualiser les audits »

La figure x affiche le diagramme de séquence de conception du cas d’utilisation « x »

* Diagramme de séquence de conception « visualiser les audits »

La figure x affiche le diagramme de séquence de conception du cas d’utilisation « x »

### **Diagramme de classe de conception pour chaque cas d’utilisation**

La modélisation des concepts ou des domaines permet d'identifier les objets importants dans une application. Ce processus nous permettra d'identifier les futurs problèmes et de mieux comprendre le fonctionnement de l'application. Ces concepts sont représentés dans le diagramme de classes. Le diagramme de classes est la clé de la conception orientée objet. Ce diagramme représente la structure du code à développer. Certaines applications UML permettent même d'exporter du code à partir de diagrammes de classes. Cela permet d'unifier le travail de plusieurs programmeurs au sein d'une même équipe, en plus de sauver du temps.

Le diagramme de classes se base sur les concepts suivants :

* **Classe :** description formelle d’un ensemble d’objets ayant une sémantique, des propriétés et un comportement commun.
* **Association :** relation sémantique entre deux ou plusieurs classes.
* **Propriété (attribut) :** élément permettant de décrire une classe ou une association.
* Diagramme de classe de conception « visualiser les audits »

La figure x affiche le diagramme de classe de conception du cas d’utilisation « x »

* Diagramme de classe de conception « visualiser les audits »

La figure x affiche le diagramme de classe de conception du cas d’utilisation « x »

* Diagramme de classe de conception « visualiser les audits »

La figure x affiche le diagramme de classe de conception du cas d’utilisation « x »

### **Diagramme de classe de conception global**

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu’un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes. Un diagramme de classes n'est donc pas adapté (sauf cas particulier) pour détailler, décomposer, ou illustrer la réalisation d'un cas d'utilisation particulier. Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

La figure x représente le diagramme de classe de conception global.

### **Diagramme de paquetages**

Un paquetage permet de regrouper sous une même appellation un ensemble d'élément de modélisation UML tels que:

* Des classes, des composants, des nœuds, des collaborations, des cas d’utilisation, ...
* Des diagrammes de classes, de collaboration, de séquence, de cas d’utilisation, ...

Paquets alimentations et paquets consommations.

La figure x représente le diagramme de paquetages.

### **Diagramme de déploiement**

Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un nœud, le diagramme de déploiement précise comment les composants sont répartis sur les nœuds et quelles sont les connexions entre les composants ou les nœuds. Les diagrammes de déploiement existent sous deux formes : spécification et instance. [3].

Selon déploiement dans bloc note.

La figure x représente le diagramme de paquetages.