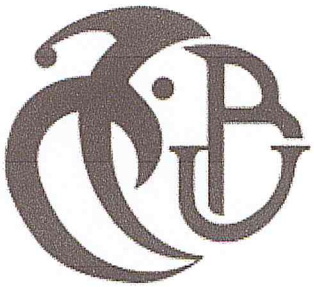
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saad DAHLEB de Blida
Faculté des sciences
Département d'informatique



Mémoire
MASTER ACADEMIQUE
Domaine : Mathématiques et d'Informatique
Filière : Informatique
Spécialité: Ingénierie Logiciel
THEME

Suivi du contrôle et la procédure au temps réel et au temps différé du produit métrologique conforme.

Rapport présenter par : Azizi Abdenacer
Redjel Abdelouahed

Soutenu devant le jury composé de :

Président	Mme Cherfa Imane	
Examinateur	Mr baouya Abdelhakim	
Promoteur	Mme Chikhi Imane	
Co-promoteur	Mr. Sofiane Chaya	Ofiice National de la Meterologie

Promotion : 2017-2018

Résumé

ملخص

تدرس الأرصاد الجوية ظواهر الغلاف الجوي ، لا سيما الطقس والمطر والغيوم ودرجة الحرارة ... فهي تتخلل حياتنا اليومية ، وتغذي جزءًا جيدًا من محادثاتنا وتضمن أنشطتنا. في هذه الرسالة ، نقدم مشروع نهاية دراستنا في مجال الأرصاد الجوية .وهدفها هو تصميم وتطوير نظام للرصد والتحكم في الوقت الحقيقي في منتجات الأرصاد الجوية.

يشمل نظامنا إدارة البيانات في الوقت الفعلي والحساب السريع لإحصاءات منتجات الأرصاد الجوية. وهو يتألف من تطبيق مخبأ وتطبيقيين: موقع ويب وتطبيق سطح مكتب ، مما يتيح إمكانية استغلال البيانات والنتائج الناتجة عن خدمة ذاكرة التخزين المؤقت.

الكلمات المفتاحية: *الأرصاد الجوية ، منتجات الأرصاد الجوية ، الوقت الفعلي ، خدمة الكاش ، الإحصائيات*

Résumé

La Météorologie étudie les phénomènes de l'atmosphère, en particulier le temps qu'il fait, la pluie, les nuages, la température…Ces derniers imprègnent notre vie courante, alimentent une bonne partie de nos conversations et conditionnent nos activités. Dans ce mémoire, nous présentons notre projet de fin d'étude dans le domaine de la Météorologie.

Il a pour objectif de concevoir et développer un système permettant le suivi et le contrôle en temps réel de produits météorologiques.

Notre système permet notamment une gestion en temps réel des données et assure un calcul rapide des statistiques relatives aux produits météorologiques. Il est composé d'une application cache et de deux applications : un site web et une application desktop, permettant d'exploiter les données et résultats générés par le service cache.

Mots-clés : Météorologie, Produit météorologique, Temps réel, Service cache, Statistiques.

Abstract

Meteorology studies the phenomena of the atmosphere, especially the weather, the rain, the clouds, the temperature ... They make our life easy to live.

We have presented our project in the field of Meteorology, its goal is to design and develop a system which is allowing real-time monitoring and control of meteorological products.

Our system includes real-time data management and fast calculation of statistics relating to meteorological products.

It consists of a cached application and two applications: a website and a desktop application, making it possible to exploit the data and results generated by the cache service.

Keywords: Meteorology, Meteorological product, Real time, Cache service, Statistics

Remerciement

Nous remercions tout d'abord notre dieu (Allah) pour la patience
qui nous offert pour finaliser ce travail dans des bonnes conditions.
Au terme de ce travail, nous tenons à exprimer nos vifs remerciements et nos profondes reconnaissances à tous particulièrement notre encadreur ***Mr. CHAYA SOFIANE,*** *d'avoir accepté de diriger ce travail.*

Nous remercions chaleureusement Madame . ***CHIKHI IMANE*** *qui nous*
aide durant notre travail et par sa patience et ses Précieux conseils.
Nous voudrions à exprimer ma profonde reconnaissance de gratitude à Mr. ***KHALED BELHOUT,*** *qui m'a accompagné de près durant tout ce travail,*
Nous remercions vivement les membres du jury qui ont Bien accepter du juger notre travail.
Nous remercions tous les enseignants du département d'informatique, aussi notre
promotion 2017-2018 Master génie logiciel.
A tous ceux qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements et leurs amitiés, à l'édification de ce projet.
A toutes ces personnes, merci du fond du coeur.

DÉDICACES

C'est avec un grand plaisir que nous dédions ce modeste travail,
fruit de notre études en exprimant notre profonde gratitude à
tous nos proches particulièrement :
À nos précieux parents pour leur amour, affection et compréhension
À nos sœurs !
À nos frères !
À tous nos amis

DEDICACES

Je dédie ce modeste travail :

A mes très chers parents qui ont tout fait pour que je réussie dans ma vie et mes études

A mon chère frère Zakaria

A ma chère sœur Rayen

A ma grande mère, à tous mes oncles et mes tantes, cousines et cousins

A toute ma famille sans exception

A mon binôme et meilleur ami Azizi Abdenacer

A tous mes amies en particulier Abdeljalil, Salim, Zakaria, Abdelhakim, Sofien …

A tous ceux qui me sont chères

Abdelouahed

Table des matières

Table des matières :

Résumé

Remercîment

Dédicace

Liste figures

Liste tableaux

Liste des acronyme

Introduction générale 1

CHAPITRE I : Présentation de l'organisation d'accueil

1.Introduction 3

2.Historique de l'Office Nationale de la Météorologie 3

3.Schéma d'organisation 3

3.1 Le Directeur général 3

3.2 Les structures fonctionnelles 3

3.3 Les structures fonctionnelles Administratives 4

3.4 Organigramme générale 4

3.5 Les unités Opérationnelles à vocation Nationale 4

S3.7 station météorologique ..... ;;;;;;;;.....5

3.7.1 Codage des stations 5

3.7.2 Horaires des stations.....6

4. Les missions principales de l'ONM.....6

5. Le Centre National Des Télécommunications Météorologiques CNTM.....7

5.1 Organigramme du CNTM 7

5.2 Les missions du CNTM 8

6. Conclusion 8

CHAPITRE II : Etude de l'existant et Analyse des besoins

1.Introduction 9

2.Les observations (produits) météorologiques 9

2.1. Le but des observations météorologiques 9

2.2. Types des observations météorologiques 10

2.2.1 Observations SPECI (SP) 10

2.2.2. Observations synoptiques (SI|SM) 10

2.2.2.1. Le code Synoptique 10

2.2.2.2. Horaires des observations Synoptiques 11
2.2.2.3. Conditions de validation 11
2.2.3. Observations METAR (SA) 11
2.2.3.1. Le code METAR 11
2.2.3.2. Date/heure de l'observation METAR.....11
2.2.3.3 Conditions de validation 12
2.2.4. Observations Altitude (US) 12
2.2.4.1. Conditions de validation des observations Altitude 12
3. Les statistiques.....12
3.1. SMQ (Système de Management de la Qualité) statistique.....12
3.2 Production Statistique 13
4. Système de commutation MESSRIR-COMM 14
4.1 Principes du mode «Pilote/Secours» 14
4.2 Les journaux (historique des événements) 15
4.2.1 Les messages d'alerte traités.....15
4.2.2 Exemplaire d'un fichier trace 16
5.Etudes de l'existant et analyse des besoins 16
6.Approche proposée.....17
6.1. L'accès au disque 17
6.2. Le calcul redondant des statistiques.....18
6.3 L'Absence d'un système de surveillance des services.....18
Conclusion 19
CHAPITRE III : Analyse et conception du système
1.Introduction 20
2.Les principes S.O.L.I.D 20
2.1. Single responsibility principle 20
2.2. Open / Close principle 21
2.3. Liskov Substitution Principle 21
2.4. Interface segregation principle 22
2.5. Dependency Inversion Principle 22
3.Conception MVC 22
3.1. Principe : 22
4.Présentation des diagrammes UML utilisés 23
4.1. Diagramme de cas d'utilisation (use case) 24

4.1.1 Liste des acteurs 24
4.1.2 Diagramme cas utilisation de service cache 25
4.1.2.1 Diagramme cas utilisation générale 25
4.1.2.2 Diagramme cas utilisation détaillé du « service cache » 25
4.1.3 Diagramme cas d'utilisation « accéder au cache » 26
4.1.4 Diagramme cas d'utilisation « présentation des données cache »: 27
4.1.5 Diagramme cas utilisation « affichage des statistique » 27
4.1.6 Diagramme cas utilisation « surveillance des services » 28
4.1.7 Diagramme cas utilisation « gestion des comptes » 28
4.2 Diagramme de séquence 28
4.2.1 Diagramme de séquence « service cache » 29
4.2.1.1 Diagramme de séquence « statistique des produits » 29
4.2.1.2 Diagramme de séquence « surveillance des services » 30
4.2.2 Diagramme de séquence « se connecter » 30
4.2.3 Diagramme de séquence d'un client d'affichage 31
4.2.4 Diagramme de séquence « gestion des comptes » 31
4.2.4.1 Diagramme de séquence « ajouter un utilisateur » 31
4.2.4.2 Diagramme de séquence « supprimer un utilisateur » 32
4.2.4.3 Diagramme de séquence « modifier le nom d'un utilisateur » 32
4.2.4.4 Diagramme de séquence « modifier le mot de passe d'un utilisateur » 33
4.3 Diagramme de classe 33
4.3.1- Service cache 33
4.4 Diagramme de package 34
4.4.1 Diagramme de package service cache 35
5.Conclusion 35
CHAPITRE IV : Réalisation du système
1.Introduction 36
2.Etude technique 36
2.1 Environnement de réalisation 36
2.1.1 Matériels de base 36
2.1.3 Choix des langages de développement et de BDD RAM 37
3. Outils de développement 40
3.1 IntelliJ IDEA 40
3.3 JUNIT 40

3.3 PHPStorm 40
3.4 StarUML 41
3.5 Scene Builder 41
3.6 Bootstrap 41
4.Mise en œuvre de notre système 41
4.1 La réalisation de l'application Service Cache 42
4.1.1 La méthode TDD 42
4.1.2 Fonctionnement du TDD 42
4.1.3 TDD tests unitaires 43
4.1.4 ATDD (Acceptance TDD) 43
4.1.5 Pourquoi utiliser le TDD ? 44
4.1.6 Exemple d'une classe test réalisée avec la méthode TDD 44
4.2 Calculs des statistiques météorologiques 46
4.2.1 Traitement des produits 46
4.2.2 Calculer les statistiques 49
4.2.3 Stocker les produits traités sur REDIS 49
4.3 Surveillance des services 50
4.3.1 Récupérer les nouveaux messages d'alerte 50
4.3.2 Structurer les messages 50
4.3.3 Filtrage des données structurées 52
4.3.4 Capturer les états des serveurs et les stocker sur Redis 53
4.3.5 Réalisation d'un ordonnanceur (Scheduler) 53
4.3.6 La classe Main 54
4.4 La réalisation des clients d'affichage 55
4.4.1 La réalisation du site web 55
4.4.1.1 Le répertoire d'application 55
4.4.1.2 Présentation de quelques interfaces de l'application web 56
4.4.2 La réalisation de l'application d'affichage Desktop 59
4.4.2.1 Le répertoire d'application 59
4.4.2.2 Présentation de quelques interfaces de l'application client d'affichage desktop 60
5.Conclusion 61
Conclusion générale 62
Bibliographie 63

Liste tableaux

Liste tableaux :

Tableau I.1: les observations synop, metar prévu de chaque horaires.....6

Tableau II.1 Les heures de transmission des messages synoptique.....11

Tableau II.2 : Validation des observations SYNOP.....12

Tableau II.3 : Les ranges des observations METAR et SYNOP.....13

Tableau II.4 : Message d'alertes du mode pilote/secours.....15

Tableau III.1 Les principes SOLID 20

Tableau III.2 List des acteurs.....24

Tableau III.3 Signification des symboles.....33

Tableau III.4 Dictionnaire des champs.....34

Tableau IV.1 : Matériel de base.....36

Liste des Acronymes

Liste des Acronymes

ONM L'office nationale de la météorologie

EPIC Entreprise public industriel et commercial

DDP La Direction du Développement et de la Planification

DCE La Direction de la Coordination de l'Exploitation

DRH La Direction des Ressources Humaines

DFC La Direction des Finances et de la Comptabilité

CNTM Centre National de Télécommunications Météorologiques

CNPM Centre National des Prévisions Météorologiques

CCN Centre Climatologique National

ART région Afrique de télécommunication

SP Observations SPECI

SI synoptiques principaux

SM synoptiques intermédiaires

SA Observations METAR

US Observations Altitude

SMQ Système de Management de la Qualité

SMT Surface-mount technology

SADIS Access Device Information Service

OACI Organisation de l'Aviation Civile Internationale

OMM Organisation Météorologique Mondiale

RAM Random-Access Memory

SSD Solid-state drive

HDD Hard Disk Drive

BDD Base de Données

UML Unified Modeling Language

MVC Model view controller

CSS Cascading Style Sheet

IDE integrated development environment

JDK Java Development Kit

HTML Hypertext Markup Language

FTP File Transfer Protocol

GNU Gnu's Not Unix

IBM International Business Machine

TDD (Test Driven Development)

BDD (Behavior Driven Development)

RGR (Red Green Refactor)

Introduction générale

INTRODUCTION GENERALE

Actuellement, l'amélioration remarquable de la qualité des prévisions météorologiques est un des grands succès de la science de l'environnement. Cela est dû notamment au progrès des systèmes numériques de prévision et aux observations de plus en plus nombreuses et variées de l'état de l'atmosphère et des milieux connexes (océan, sols, végétation…), particulièrement les observations des satellites d'observation de la Terre. Le développement rapide des supercalculateurs a été une des clés de ce succès.

Le travail présenté dans ce mémoire s'inscrit dans ce contexte. Il a été effectué au niveau de l'ONM (Office Nationale de la Météorologie) notamment dans le CNTM (Centre National de Télécommunication Météorologique).

L'ONM fait autorité pour les questions relatives au temps, au climat et aux ressources en eau. Elle diffuse des informations scientifiques et des prévisions sur l'atmosphère terrestre, son interaction avec les océans, le climat et la répartition des ressources en eau. L'ONM encourage l'échange rapide et libre de données météorologiques, la normalisation des observations, ainsi que la publication uniforme des observations et des statistiques.

Notre projet a pour objectif de concevoir et développer un système permettant le suivi et le contrôle en temps réel de produits (observations) météorologiques. Notre système permet notamment une gestion en temps réel des données et assure un calcul rapide des statistiques relatives aux produits météorologiques. Il est composé d'une application cache et de deux applications (un site web et une application desktop) permettant d'exploiter les données et résultats générés par le service cache.

Pour la mise en œuvre de notre système, nous nous somme basés sur les principes orientés objet S.O.L.I.D et adopter un processus de développement dirigé par les tests : la méthode DDT. Nous avons utilisé également pour la conception de notre système le langage de modélisation UML.

Ce mémoire est organisé en 4 chapitres :

Le premier chapitre introduit l'organisme d'accueil à savoir l'ONM. Nous présentons les principales missions de l'établissement, son historique et son organisation. Nous mettons par la suite l'accent sur le service CNTM dans lequel notre projet a été mené.

Dans le chapitre 2, nous présentons l'étude de l'existant et l'analyse des besoins effectuées au niveau du CNTM pour aborder les problèmes visés par notre projet et les solutions proposées.

Dans le chapitre 3, nous présentons l'étude conceptuelle de notre système et les principes S.O.L.I.D.

Le dernier chapitre est consacré à la mise en œuvre de notre système. Nous décrivons les différents constituants de ce dernier, la démarche de développement adoptée, les outils et langages utilisés et une analyse des résultats obtenus.

Enfin, le mémoire se termine par une conclusion générale qui présente un bilan du travail réalisé dans ce mémoire pour améliorer et compléter le travail réalisé.

1. Introduction

Dans ce chapitre, nous introduisons l'organisme d'accueil à savoir l'Office Nationale de la Météorologie : ONM. Nous présentons les principales missions de l'établissement, son historique et son organisation. Nous mettons par la suite l'accent sur le service CNTM de l'ONM dans lequel notre projet a été mené. Nous décrivons son organigramme et ses principales missions.

2. Historique de l'Office Nationale de la Météorologie

L'office nationale de la météorologie (ONM) est passé depuis 1998 du statut d'EPA (entreprise public administratif) au statut d'EPIC (Entreprise public industriel et commercial).

De ce fait, pour la bonne mise en œuvre des programmes répondant à ses missions et a ses objectifs, ainsi que pour la bonne gestion et stimulation du personnel, l'Office est tenu de se donner ou mettre à jour tous les outils nécessaires à la conduite efficace d'un Etablissement de type EPIC .

Parallèlement au projet de nomenclature des postes de travail spécifique à l'ONM, il a été jugé important de procéder au réaménagement de l'Organigramme qui tiendrait mieux compte des réalités actuelles de l'Office et de son nouveau statut d'EPIC, et ce après avoir évalué l'organisation qui a prévalu depuis 1988 à ce jour.

Ce réaménagement doit être considéré comme une première étape, l'organigramme est évolutif et devra être adapté périodiquement en fonction du développent de l'ONM.

3. Schéma d'organisation

3.1 Le Directeur général :

Au Directeur Générale sont directement rattachées les fonctions suivantes :

- Directeur d'Audi
- Relation extérieures et Communication
- Bureau d'Ordre Général /Secrétariat
- Chargés de synthèses

3.2 Les structures fonctionnelles

On distingue deux directions techniques fonctionnelles chargées de la stratégie, la Planification et la coordination opérationnelles:

- La Direction du Développement et de la Planification (DDP)
- La Direction de la Coordination de l'Exploitation (DCE)

3.3 Les structures fonctionnelles Administratives :

- La Direction des Ressources Humaines (DRH)
- La Direction des Finances et de la Comptabilité(DFC)

3.4 Organigramme générale

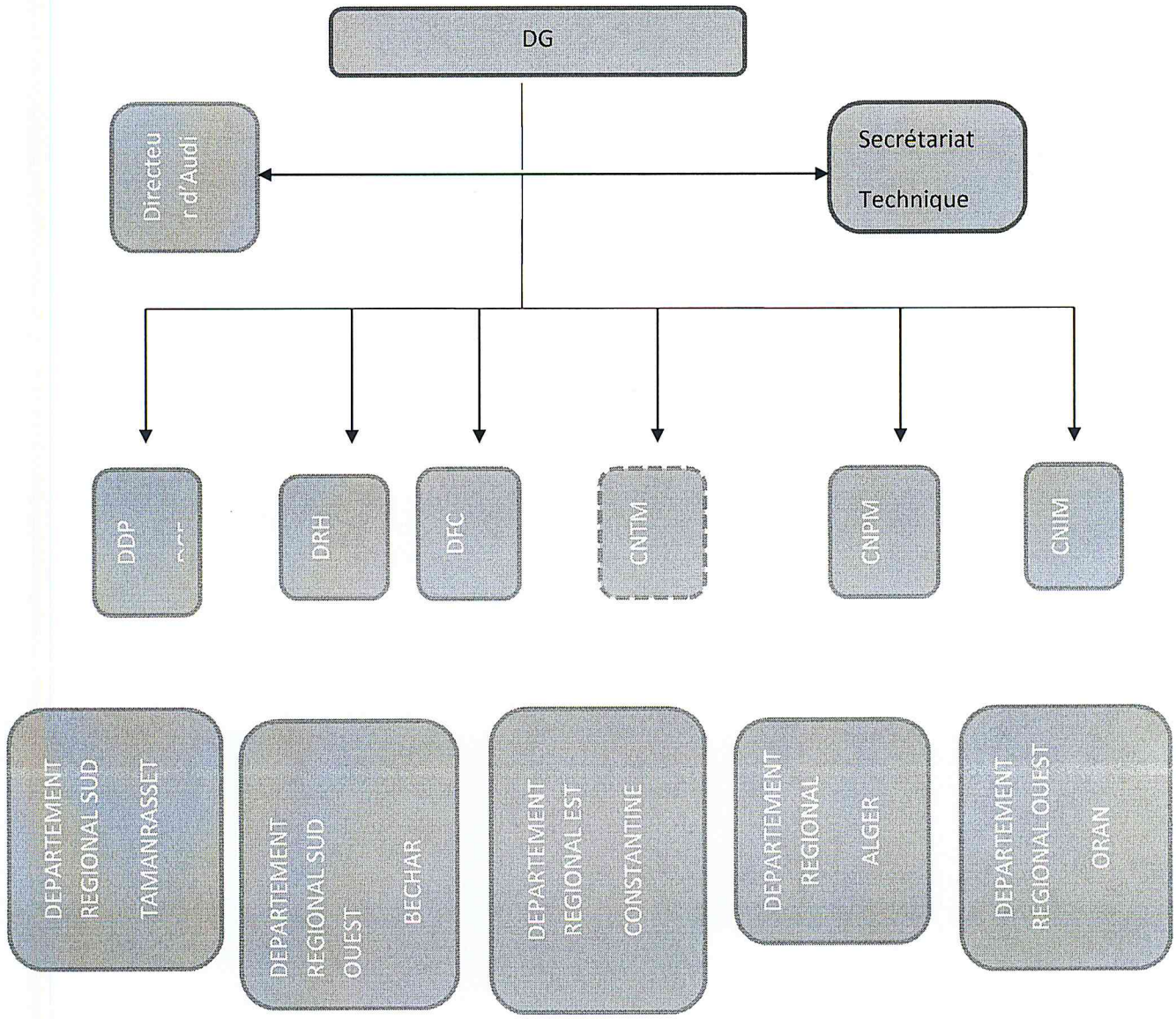


Figure I.1 Organigramme générale de l'Office Nationale de la Météorologie (ONM)

3.5 Les unités Opérationnelles à vocation Nationale

- Centre National de Télécommunications Météorologiques (CNTM)
- Centre National des Prévisions Météorologiques (CNPM)
- Centre Climatologique National (CCN)
- Centre National des Installations et de la Maintenance (CNIM)

Au niveau des unités opérationnelles à vocation régionale, on distingue six Départements Régionaux :

- Département Régional OUEST (ORAN)
- Département Régional EST (CONSTANTINE)
- Département Régional Centre (ALGER)
- Département Régional SUD EST (OURGLA)
- Département Régional SUD OUEST (BECHAR)
- Département Régional SUD (TAMANRASSET)

Chaque région contient plusieurs stations.

3.7 Station Météorologique

Une station météorologique est un ensemble de capteurs qui enregistrent et fournissent des mesures physiques et des paramètres météorologiques liés aux variations du climat, ces capteurs étant placés dans un boîtier, abri météorologique qui réalise l'équilibre thermique du thermomètre avec l'air et le protège du rayonnement solaire. Les variables à mesurer sont la température, la pression, la vitesse et direction du vent, l'hygrométrie, le point de rosée, la pluviométrie, la hauteur et le type des nuages, le type et l'intensité des précipitations ainsi que la visibilité. Les stations peuvent comporter des capteurs pour toutes ou une partie seulement de ces informations, selon leur type : agro-météorologique, d'aéroport, météo routière, climatologique, etc. [1]

3.7.1 Codage des stations

- **OMM** : C'est un code de six chiffres attribué à chaque aérodrome ou station dans le monde (60390)
- **OACI** : C'est un code de quatre lettres attribué à chaque aérodrome ou station dans le monde

Le code OACI de chaque aérodrome est attribué selon un préfixe dépendant de sa localisation géographique. Généralement, la première lettre désigne une région du monde, la seconde le pays et les deux dernières l'aérodrome ou la station. Exemple : DAMM, DAAG, DAUH,….

3.7.2 Horaires des stations

Les horaires de stations sont des catégories qui contiennent des horaires dans lesquels les stations peuvent être nécessaires d'envoyer des observations météorologiques, L'Algérie compte actuellement six catégories dans lesquelles les stations fonctionnent, et chaque station a sa propre classe, ces horaires sont celles que l'on peut connaître le nombre d'observations envoyées quotidiennement pour chaque type.

Horaires	SYNOP	METAR
0000/2300	8	24
0000/4800	8	48
0100/2300	8	16
0600/1800	5	13
0700/1700	5	8
0812/1500	3	5

Tableau I.1: les observations synop, metar prévu de chaque horaires

4. Les missions principales de l'ONM

L'Office National de la météorologie fonctionne selon les normes ayant un caractère opérationnel prononcé compte tenu de sa mission de veille de l'atmosphère et du temps. Il est chargé par l'état des missions suivantes :

- Il assure la veille météorologique nationale
- Il est chargé de l'installation, de l'exploitation, de la gestion et du développement des infrastructures météorologiques de base au niveau national
- Il assure la fourniture d'une assistance météorologique diversifiée aux différents secteurs d'activités nationales. En matière de sécurité, des bulletins d'alerte sont diffusés pour la protection des biens et des personnes.
- Il assure la promotion des actions internationales en coordination avec la Tutelle, compte tenu du fait que la Météorologie n'a pas de frontière, nécessitant la coopération internationale pour garantir les échanges de données Météorologiques.
-

5. Le Centre National Des Télécommunications Météorologiques CNTM

5.1 Organigramme du CNTM

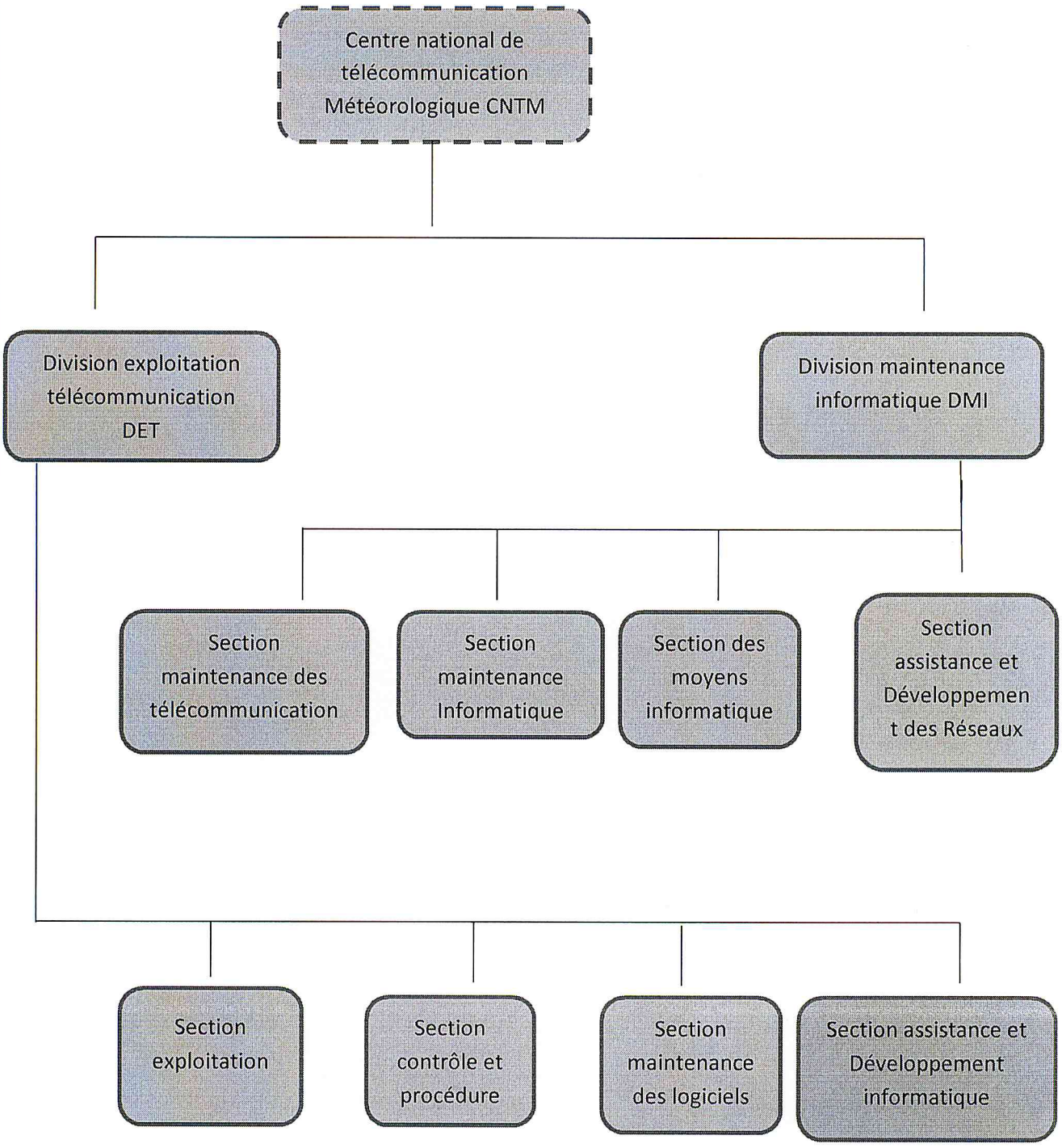


Figure I.2. Organigramme du centre national télécommunication météorologique CNTM

Il comprend deux divisions :

a. La Division de l'Exploitation des Télécommunications qui comporte quatre sections :

- La Section Exploitation Télécommunication
- La Section de l'Assistance et du Développement Informatique

- La Section Contrôle et Procédures
- La Section Maintenances des Logiciels

Par ailleurs, cette divisions possède des équipes d'exploitation qui se relayent 24Heures sur 24 dirigées chacune par un Chef d'équipe. La composition des équipes est fixée par Décision du Directeur Général.

b. La Division de maintenance des systèmes et de l'assistance Informatique qui comporte quatre sections :

- La Section de la Maintenance Informatique
- La Section de la Maintenance des Télécommunications
- La Section Gestion des moyens Informatiques
- La Section de l'Assistance et du Développement des Réseaux

5.2 Les missions du CNTM

Le centre national des télécommunications météorologiques CNTM est chargé d'assurer :

- Le C.N.T.M collecte en temps réel (80) stations d'observation en surface du réseau national d'observation météorologique professionnel. Chaque stations transmise des observations métrologique.
- La gestion et le fonctionnement des systèmes informatiques.
- Le CNTM assure les fonctions du centre régional de télécommunications(CRT)

6. Conclusion

Dans ce chapitre, nous avons présenté l'organisme dans lequel s'inscrit notre travail à savoir l'Office Nationale de la Météorologie. Nous avons décrit ses principales missions et défini ses services en particulier le service CNTM concerné par notre projet.

Dans le chapitre qui suit, nous présentons l'étude de l'existant et l'analyse des besoins que nous avons effectués au niveau du CNTM, les problèmes que nous avons constatés par la suite et les solutions proposées.

Chapitre II :

Etude de l'existant et Analyse des besoins

1. Introduction

L'ONM fait autorité pour les questions relatives au temps, au climat et aux ressources en eau. Elle diffuse des informations scientifiques et des prévisions sur l'atmosphère terrestre, son interaction avec les océans, le climat et la répartition des ressources en eau. L'ONM encourage l'échange rapide et libre de données météorologiques, la normalisation des observations, ainsi que la publication uniforme des observations et des statistiques. Cet échange de données est effectué en utilisant le système de commutation de messages météorologiques ouvert à tous les nouveaux moyens de télécommunication.

Ainsi, dans ce deuxième chapitre, avant de décrire ce système de commutation, nous présentons le concept d'observation météorologique (définition, types et buts). Par la suite, nous présentons l'étude de l'existant et l'analyse des besoins que nous avant effectués au niveau du CNTM pour aborder ensuite les problèmes constatés et les solutions proposées.

2. Les observations (produits) météorologiques

L'observation est le point de départ de toute prévision météorologique. Elle est donc naturellement au cœur des activités de recherche de Météo d'Algérie visant à améliorer la qualité des prévisions.

Certaines stations d'observation météorologiques sont choisies pour effectuer des observations météorologiques et fournir des messages sous une forme destinée d'abord à répondre aux besoins du personnel navigant de l'aviation et à d'autres usagers. Ces messages sont appelés observations horaires ou des produits météorologique. Les observateurs assignés aux observations horaires doivent observer continuellement le temps et signaler immédiatement toute variation importante. On doit continuellement maintenir en tout temps le programme d'observation prévu de façon qu'aucune discontinuité ne survienne dans les relevés.

Toutes les stations doivent respecter ces horaires d'observation, à moins d'avoir reçu une permission spéciale du sous-ministre adjoint de s'en écarter.

2.1. Le but des observations météorologiques

Le but des observations météorologiques est de recueillir des renseignements détaillés sur le temps et le climat afin de répondre aux besoins de divers usagers. Certains de ces usagers demandent des renseignements à la minute près; d'autres s'intéressent aux données climatologiques quotidiennes, mensuelles ou à long terme. Par exemple, un pilote voudra les

conditions actuelles et prévues; une entreprise de chauffage demandera les données de degrés-jours. L'agriculteur s'intéressera aux renseignements sur la température, l'insolation et les précipitations. D'autre part, lorsqu'on décide de construire un aéroport dans un endroit donné ou de déterminer les normes de stress auxquelles un édifice devrait se conformer pour résister à l'accumulation de neige sur le toit ou à la force du vent, on tiendra compte de données météorologiques recueillies sur une longue période. Par conséquent, les observations météorologiques et les bulletins soigneusement préparés ont une valeur tant immédiate qu'à long terme. [2]

2.2. Types des observations météorologiques

2.2.1 Observations SPECI (SP)

On doit faire une observation SPECI sans délai pour signaler les changements du temps entre les heures prescrites de transmission. Une observation SPECI doit comprendre les éléments suivants :

- État du ciel
- Visibilité
- Conditions atmosphériques et obstacle à la vue
- Pression au niveau de la mer
- Température
- Point de rosée
- Vent
- Calage de l'altimètre
- Nuages
- Remarques (au besoin)
- Portée visuelle de piste (RVR) (si disponible).[3]

2.2.2. Observations synoptiques (SI|SM)

2.2.2.1. Le code Synoptique

Le code météorologique international FM 12-IX SYNOP est utilisé pour transmettre les observations synoptiques de surface des stations terrestres, dotées de personnel ou automatiques. Ce code est appelé FM 13-IX SHIP lorsqu'il sert à transmettre des observations similaires de stations maritimes dotées de personnel ou automatiques. Le code synoptique élémentaire comprend six sections numérotées de 0 à 5. Chaque section est composée de groupes de code à 5 chiffres. La plupart des groupes des sections 0 à 5 commencent par un indicateur numérique et ces indicateurs sont numérotés successivement à l'intérieur de chaque

section. Les indicateurs numériques identifient un groupe particulier contenant toujours les mêmes éléments atmosphériques. De ce fait, l'omission qu'elle soit accidentelle ou volontaire, d'un groupe quelconque n'affectera pas l'identification des autres groupes. De toute façon, le code permet l'omission d'un groupe dont les éléments atmosphériques sont absents ou ne peuvent être observés. Cela assure une souplesse de code suffisante aux stations dotées de personnel et automatiques. [4]

2.2.2.2. Horaires des observations Synoptiques

Les horaires sont liés au type du message Synoptique :

Type du message	Les heures de transmission (UTC)
synoptiques principaux (SM)	0000, 0600, 1200 et 1800
synoptiques intermédiaires(SI)	03 00, 09 00,1500 et 2100

Tableau II.1 Les heures de transmission des messages synoptique

Dans tous les cas, le baromètre doit être lu sur l'heure. L'observation, l'enregistrement et le codage de tous les éléments, sauf la pression et la tendance barométrique devraient être effectués dans les 10 minutes qui précèdent l'heure. Toutes les stations doivent se conformer à cet horaire d'observation, à moins d'obtenir du sous-ministre adjoint la permission expresse de s'en écarter.

2.2.2.3. Conditions de validation

Pour être considérée comme une observation valide : L'heure reçue d'observation ne dépasse pas 170 minutes après et 10 minutes avant l'heure prévue, sinon elle est considérée comme une observation avancée ou retardée.

2.2.3. Observations METAR (SA)

2.2.3.1. Le code METAR

METAR est le nom du code météorologique international pour un message d'observation météorologique régulière pour l'aviation. Normalement, les observations METAR sont enregistrées et diffusées à l'heure pile pour des stations et chaque demi-heure pour d'autres.[5]

2.2.3.2. Date/heure de l'observation METAR

La date et l'heure de l'observation doivent être incluses dans tous les messages. La date et l'heure d'observation à l'heure pile sont utilisées pour tous les messages METAR.

2.2.3.3 Conditions de validation

Pour considère comme observation valide :

Condition / horaire	00 :48 (chaque demi-heure)	autre
Avant l'heure prévue	10 minutes	20 minutes
Après l'heure prévue	10 minutes	50 minutes

Tableau II.2 : Validation des observations SYNOP

Les autres sont considérées comme des observations avancées ou retardées.

2.2.4. Observations Altitude (US)

En météorologie, la distance verticale d'un point de donné de l'atmosphère au niveau moyen de la mer n'est pas mesurée par son altitude, mais par un nombre qui en est généralement très voisin et qui s'appelle l'altitude géo potentielle de ce point. Il en résulte en particulier que les lignes isohypses destinées à décrire le "relief" de surfaces isobares données (850 hPa, 500 hPa, etc.) ne tracent pas exactement des courbes d'altitude constante, comme le font les lignes de niveau des cartes topographiques, mais des lignes d'altitude géo potentielle constante.

2.2.4.1. Conditions de validation des observations Altitude

Nous considérons l'observation comme indésirable quand l'heure prévue de l'observation appartient pas à l'horaire de stations.

3. Les statistiques

Pour chacune des statistiques suivantes, nous distinguons des statistiques quotidiennes, mensuelles, trimestrielles et annuelles :

3.1. SMQ (Système de Management de la Qualité) statistique

C'est des statistiques qui nous permettent de contrôler l'heure d'envoi des stations des observations météorologique. Le temps est divisé en périodes spécifiques, à travers lesquelles nous pouvons connaître le nombre et le pourcentage d'observations envoyées par chaque station dans chaque période. Plus la période est plus proche de l'heure prévue plus les

prévisions météorologiques étaient plus proches du réalisme, plus les statistiques sont précises.

Pour les observations de type METAR et SYNOP, on calcule l'intervalle entre l'heure de transmission et l'heure de transmission attendue, puis on convertit en entier.
Diff = (entier) (heure reçu - heure prévu).

Range	Période (SYNOP)	Période(METAR)
AV	Diff < -10	-10 < Diff
A0	-10 <= Diff < +5	-10 <= Diff < +3
A1	+5 <= Diff < +10	+3 <= Diff < +5
A2	+10 <= Diff < +15	+5 <= Diff < +7
A3	+15 <= Diff < +20	+7 <= Diff < +9
A4	+20 <= Diff < +170	+9 <= Diff < +15
A5	/	+15 <= Diff < +20
A6	/	+20 <= Diff < +170
AR	+170 <= Diff	+170 <= Diff

Tableau II.3 : Les ranges des observations METAR et SYNOP

3.2 Production Statistique

C'est une statistique par laquelle on peut connaître le nombre et le pourcentage d'observations météorologiques envoyées par chaque station chaque jour, mois, trimestre et année, compte tenu du nombre d'observations que chaque station doit envoyer, et on peut aussi connaître le nombre d'observations envoyées avant ou après la durée prévue de la transmission. La durée prévue de la transmission varie en fonction de la station envoyée et du type d'observation. Les observations avancées et retardées sont incluses dans la période de transmission attendue dans le domaine des notes manquantes, telles que les observations non envoyées.

4. Système de commutation MESSRIR-COMM

MESSIR-COMM est un autocommutateur de messages du SMT et aussi un système de télécommunications qui reçoit et distribue tous types de données et produits.

- Messages texte (SYNOP, TEMP, METAR…)
- Produits binaires comme les GRIB, BUFR, T4-DFAX
- Images satellite et images Radar
- Fichiers au format OMM Non-standard (.doc, .bmp etc…)

Il reçoit les données des sources suivantes :

- SADIS
- MSG
- ALGER

Il transmet ces données aux différents abonnés.

Les serveurs MESSIR fonctionnent avec deux serveurs en mode pilote et secours:

- Un est **Pilote** et est en charge des opérations.
- Un est **Secours** et est prêt à prendre automatiquement les opérations en cas de dysfonctionnement du **Pilote**.

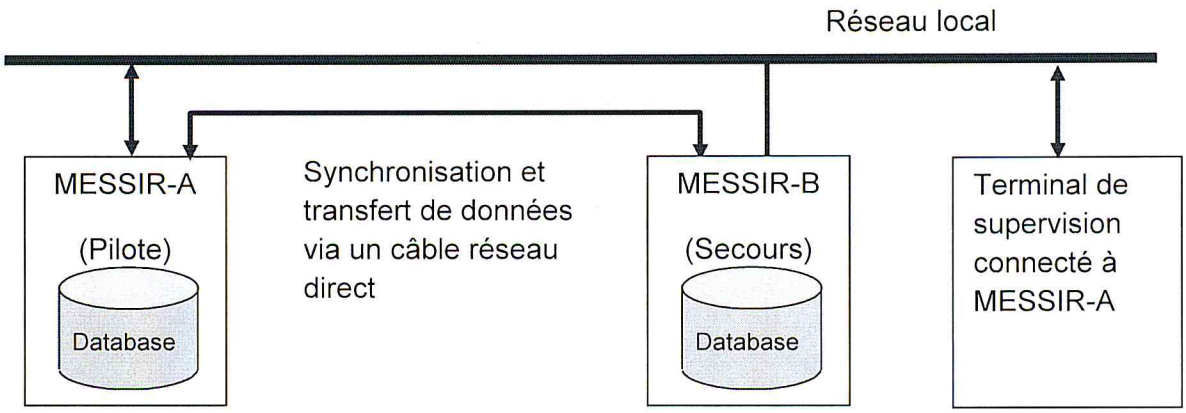
Les deux serveurs peuvent fonctionner indifféremment en mode **Pilote** ou en mode **Secours**.[6]

4.1 Principes du mode «Pilote/Secours»

Le mode pilote/secours consiste à avoir deux serveurs, un serveur « pilote » faisant le travail réel de commutation de messages et de stockage en base, et un autre serveur « secours » recevant en temps réel du serveur pilote toutes les données nécessaires pour être à jour. Il y a une communication et un contrôle permanent entre les serveurs pilote et secours via un câble réseau direct. La machine secours est mise à jour pour mettre à jour sa propre base de données.

Le serveur de secours a donc à un instant donné le même état que le serveur pilote. De plus, il vérifie périodiquement l'état du serveur pilote afin de devenir pilote en cas de défaut du pilote.

Pour distinguer les deux machines, l'une est appelée serveur A et l'autre serveur B.



FigureII.1 Model pilot/secoure

4.2 Les journaux (historique des événements)

Chacun des serveurs MESSIR-A ou MESSIR-B stockent dans des fichiers Txt une trace de leur activité et en particulier des erreurs rencontrées. Par exemple, l'état des serveurs : pilot ou secoure, la communication avec les stations et la réception des données et autres problèmes de fonctionnements du système.

4.2.1 Les messages d'alerte traités

Les fichiers de trace contiennent de nombreux messages. Dans notre projet, nous traitons les messages d'alerte suivants :

a. Pour mode pilote/secours

Etat	Exemple du message d'alertes
Pilot	« Dec 31 21:14:11 [Supervision.cpp; 1330] Application started - pilot»
Secours	« Dec 18 11:14:28 [Supervision.cpp; 1044] Application started standby, Dec 01»
En panne	« Dec 18 11:12:07 [Supervision.cpp; 236] Application stopping » « Dec 17 08:20:01 [Supervision.cpp; 1079] Pilot not responding » « Dec 17 08:20:01 [Supervision.cpp; 1079] Pilot not responding»

Tableau II.4 : Message d'alertes du mode pilote/secours

b. Pour les problèmes de communication avec les stations

S'il y a un problème de connexion avec une station, nous devrions le savoir préalablement afin de recevoir les observations en temps réel, par exemple :

« Dec 31 18:36:59 [Supervision.cpp; 645] Impossible d'appeler CTR_RX_OBS 192.168.0.41 (Transfer failed) »

« Dec 31 19:36:23 [Supervision.cpp; 645] Impossible d'appeler CMFA-2Mb-SOC reghaia::5555 (Connexion refusée) »

« Jan 06 22:55:29 [Supervision.cpp; 645] Impossible d'appeler test-controle 10.144.1.49::5555 (Connexion refusée) »

4.2.2 Exemplaire d'un fichier trace

Il s'agit de fichiers .TXT qui n'ont pas des structures :

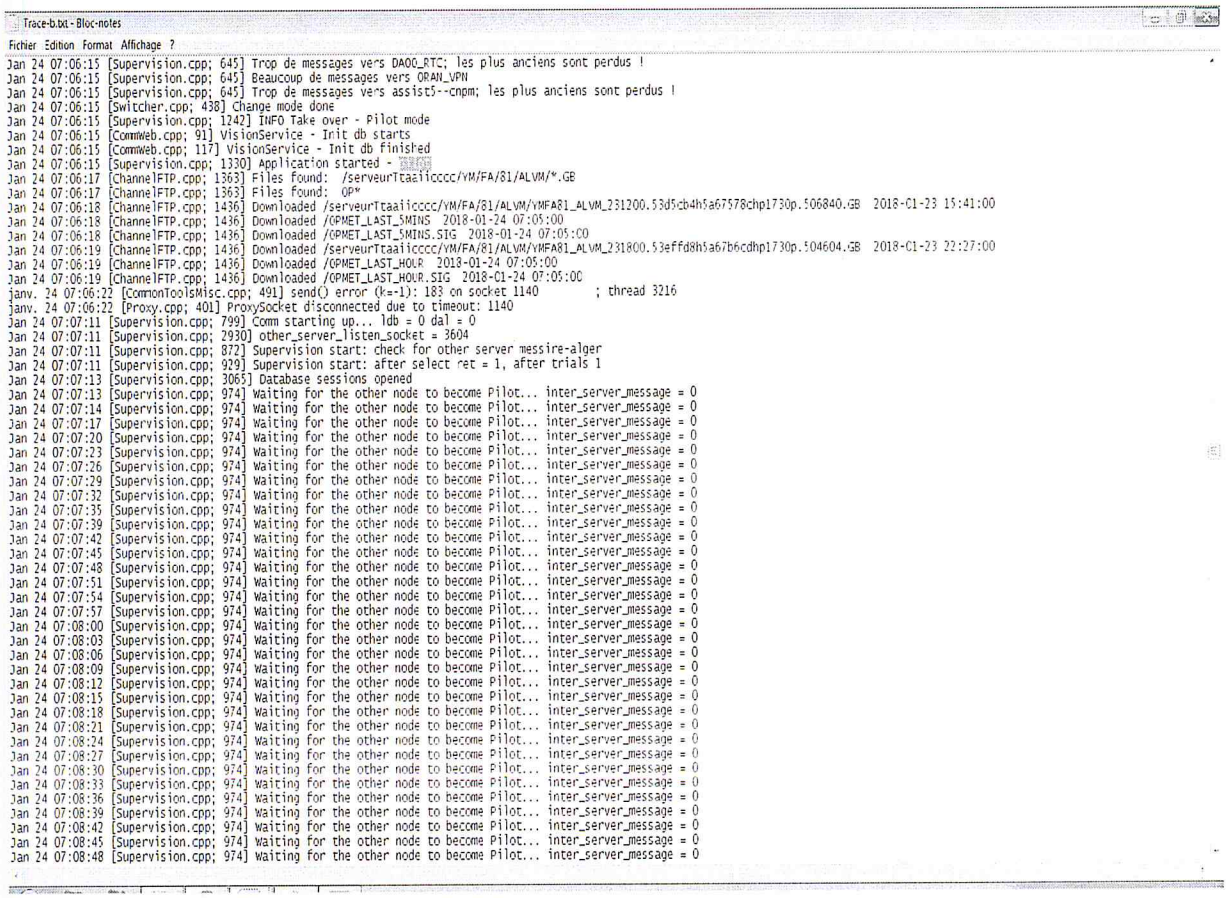


Figure II. Exemple d'un fichier trace

5. Etudes de l'existant et analyse des besoins

L'objectif ultime de la CNTM est :

- L'amélioration et le contrôle de la production des stations météorologiques pour obtenir des prévisions météorologiques plus précises: plus les observations sont traitées en temps réel plus on a de meilleures prévisions.

- L'attente mensuellement d'un taux de diffusion de 90 % des METAR au plus tard H+5 suivant les exigences de l'OACI.
- L'attente mensuellement d'un taux de diffusion de 90 % des SYNOP au plus tard H+10.

Pour cela, des solutions informatiques ont été réalisées afin de répondre aux besoins suivants :

- La réalisation du contrôle en temps réel et en temps différé.
- L'élaboration des statistiques par des tableaux et des graphes.
- L'optimisation du cycle des données.

Cependant, après une étude de la situation actuelle au niveau de la CNTM, nous avons constaté les problèmes suivants :

- Etant donné que les systèmes météorologiques et les conditions climatiques ne connaissent pas de frontières, il est indispensable que les renseignements météorologiques circulent librement par le monde, la base de données PostgreSQL du système MISSIR reçoit chaque jour plus de 5G de données météorologique nationales.
- Le temps d'exécution d'une requête qui retourne les données d'un mois dépasse 10 minutes, ainsi, ce temps est encore plus important dans le cas de données d'un an ou plus !
- Les agents du service approvisionnement de CNTM utilisent un outil réalisé par un agent spécialiste en informatique. Cet outil est en fait un site web réalisé avec HTML 4. Le site fonctionne mais il est très lent ; Il faut accéder à chaque fois au disque et calculer les statistique depuis le début ainsi y'a un calcul redondant des statistiques.
- Absence d'un système de surveillance des services (analyses des journaux et évaluations des performances) et de solutions de tolérance aux pannes.

6. Approche proposée

Après l'étude de l'existant et une analyse des besoins, nous proposons de remédier aux différents problèmes soulevés comme suit :

6.1. L'accès au disque

Pour l'optimisation des accès aux données dans le disque, nous proposons l'optimisation périodique de fichiers via l'exécution de taches de maintenances de BDD (analyses et réindexassions). Ces taches sont réalisées automatiquement par le serveur BDD. Par

exemple : sous PostgreSQL, l'ajout de la directive auto Vacuum aux tables concernées par notre travail. Cette solution va réduire le temps d'accès aux données.

Par ailleurs, nous proposons de se baser sur les serveurs de disque stockage hiérarchique avec l'utilisation de plusieurs technologie de stockage combinées (RAM,SSD,HDD). Ceci va réduire considérablement le temps d'accès aux données. Cependant, ceci est très couteux et nécessite une étude de planification long terme.

6.2. Le calcul redondant des statistiques

Afin d'éviter un calcul redondant des statistiques, nous proposons de créer un service cache qui met à jour périodiquement les statistiques et stocke les résultats sur une BDD sur une mémoire vive (RAM) par exemple REDIS.

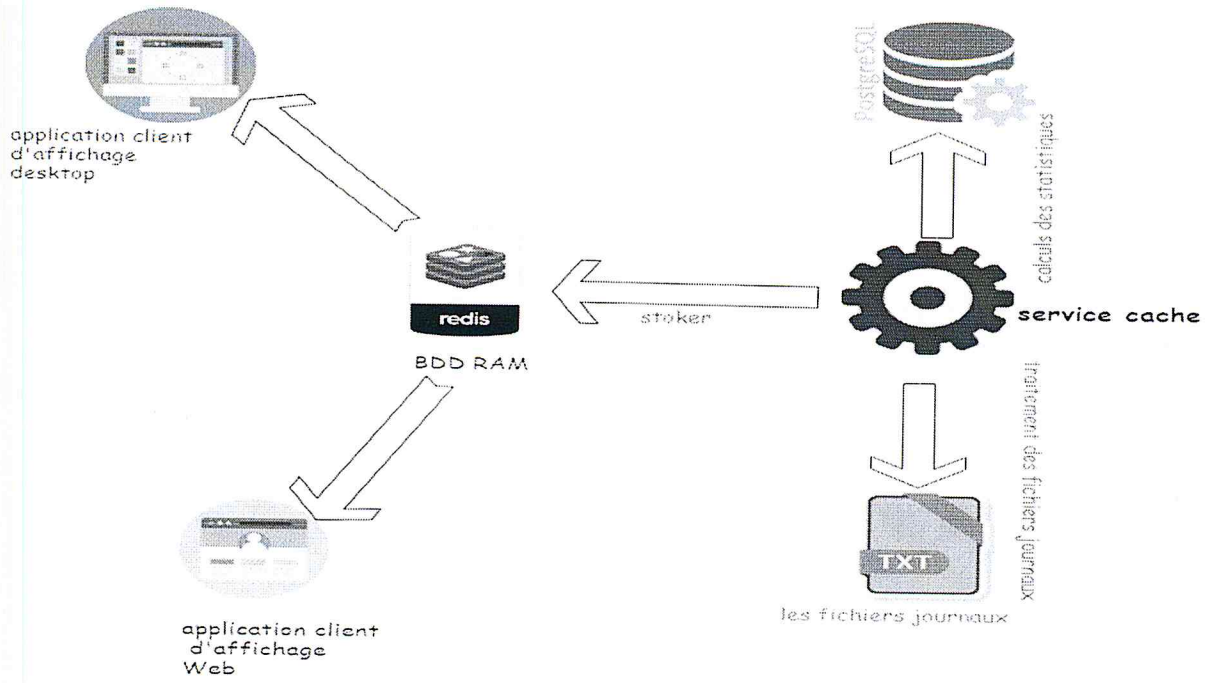
Ce service permet d'effectuer une tâche périodiquement en background sans qu'il n'y ait une interface utilisateur. Ce type de services sont très utiles si l'on souhaite notamment lancer une application au démarrage de l'ordinateur, sans pour autant qu'un utilisateur ne soit connecté.

L'accès au contenu du cache est fourni via deux applications client utilisées par deux services différents :

- **Un site Web :** Permet le contrôle en temps réel de la réception des produits météorologique et afficher les détails du contrôle du système (alertes).
- **Une application desktop :** Utilisée par le service Operations Coordination Manager (DCE). C'est le seul service qui peut accéder à cette application. Elle contient des détails sur le système de gestion de la qualité et les statistiques de production pour contrôler la production trimestrielle et annuelle des stations.

6.3 L'Absence d'un système de surveillance des services

Nous proposons d'utiliser le même service cache afin d'analyser les journaux et de créer des statistique sur les événements importants, capturer l'état des serveurs (pilot ou secoure, hors service) et stocker les résultats en mémoire cache.



FigureII.3 : Aperçu de la solution proposée.

7. Conclusion

Dans ce chapitre, nous avons présenté les différents types d'observation météorologique sur lesquels porte notre travail ainsi que les statistiques utilisées par le service CNTM. Nous avons aussi décrit le système de commutation MESSIR-COMM. Par la suite, nous avons abordé les problèmes constatés au niveau du CNTM. Enfin, nous avons présenté nos solutions proposées. Le chapitre suivant présente la conception de nos solutions.

Chapitre III :

Analyse et conception du système

1. Introduction

Ce chapitre est dédié à la conception de notre solution. Pour cela, nous avons utilisé le formalisme UML, qui s'impose aujourd'hui comme le langage de modélisation objet standardisé pour la conception des logiciels. Il a été pensé pour permettre de modéliser les activités de l'entreprise, et employés dans les projets logiciels, Ainsi il offre une flexibilité marquante.

Par ailleurs, il existe également de grands principes permettant de guider la création d'une architecture organisée, cohérente et d'améliorer la qualité des différents processus du cycle de développement logiciel. Dans notre cas, nous avons utilisé les principes S.O.L.I.D et le modèle de conception MVC, que nous décrivons dans ce chapitre.

2. Les principes S.O.L.I.D

SOLID est un acronyme représentant 5 principes de bases pour la programmation orientée objet.

L'acronyme	Le nom	Le synonyme
S	Single responsibility principle	Une classe = une et une seule responsabilité
O	Open/closed principle	Modifier le comportement d'un module sans modifier le code source de ce dernier !
L	Liskov Substitution Principle	Le contrat défini par la classe de base doit être respecté par les classes dérivées
I	Interface segregation principle	Petites interfaces, adaptées au besoin
D	Dependency Inversion Principle	Le commandant n'a pas besoin de tout savoir faire

Tableau III.1 Les principes SOLID [7]

2.1. Single responsibility principle

Cela signifie:

- Une classe = une et une seule responsabilité
- Une seule responsabilité = Servir un seul acteur.
- Mais aussi : une méthode = une seule fonctionnalité
- Un package = un seul ensemble cohérent de fonctionnalité.

C'est le principe numéro UN !

Bon nombre de design patterns servent à respecter ce principe. Il a pour objectif :

- Limiter rigidité et fragilité
- Aider à la localité

Une des principales difficultés du développement d'un logiciel complexe est la gestion et le contrôle des dépendances. Le respect du Single responsabilité principale est essentiel pour concevoir des composants/packages/classes sans dépendances excessives et sans dépendances cachées. [8]

2.2. Open / Close principle

Une classe ou un module est soit

- Open : Le module ouvert pour extensibilité (changement du comportement est possible)
- Close : Source code est fermé pour modification.

En théorie, on peut passer un module que de open vers close mais pas vice-versa. L'extension fonctionnelle / réutilisation ne veut pas forcément dire héritage. Au quotidien, la plupart des modules sont fermes :

- parce qu'on ne PEUT pas les modifier
- parce qu'on ne VEUT pas les modifier : pour ne pas les complexifier inutilement, ne pas leur donner une nouvelle responsabilité, ...

2.3. Liskov Substitution Principle

« Si une propriété P est vraie pour une instance x d'un type T, alors cette propriété P doit rester vraie pour tout instance y d'un sous-type de T »

Par conséquent :

- Le contrat défini par la classe de base (pour chacune de ses méthodes) doit être respecté par les classes dérivées.
- L'appelant n'a pas à connaitre le type exact de la classe qu'il manipule : n'importe quelle classe dérivée peut être substituée à la classe qu'il utilise.
- C'est le principe de base du polymorphisme : si on substitue une classe par une autre dérivée de la même hiérarchie, le comportement est différent mais conforme au contrat. [9]

2.4. Interface segregation principle

Un client doit avoir des interfaces avec uniquement ce dont il a besoin. Ceci :

- Incite à ne pas faire "entracte interface" sans réfléchir.
- Incite à avoir des interfaces petites.

2.5. Dependency Inversion Principle

Le principe de base qui sous-tend l'inversion de dépendance : "Depend upon Abstractions. Do not depend upon concretions."[10]

Sur , deux aspects a bien comprendre :

.• Les modules de haut niveau ne doivent pas dépendre de modules de bas niveau. Les deux devraient dépendre de abstractions

.• Les abstractions ne devraient pas dépendre de détails. Les détails doivent dépendre des abstractions.

3. Conception MVC

MVC est un design pattern (modèle de conception) de conception d'interface utilisateur permettant de découpler :

- Le modèle logique métier et accès aux données.
- Des vues : interfaces utilisateur (présentation des données et interface de saisie pour l'utilisateur).

Des modifications de l'un n'auront ainsi, idéalement, aucune conséquence sur l'autre ce qui facilitera grandement la maintenance. Ce design pattern a tendance à multiplier le nombre de classes à définir et semble alourdir la conception d'application mais le découplage ainsi obtenu assure une maintenance facilitée.

3.1. Principe :

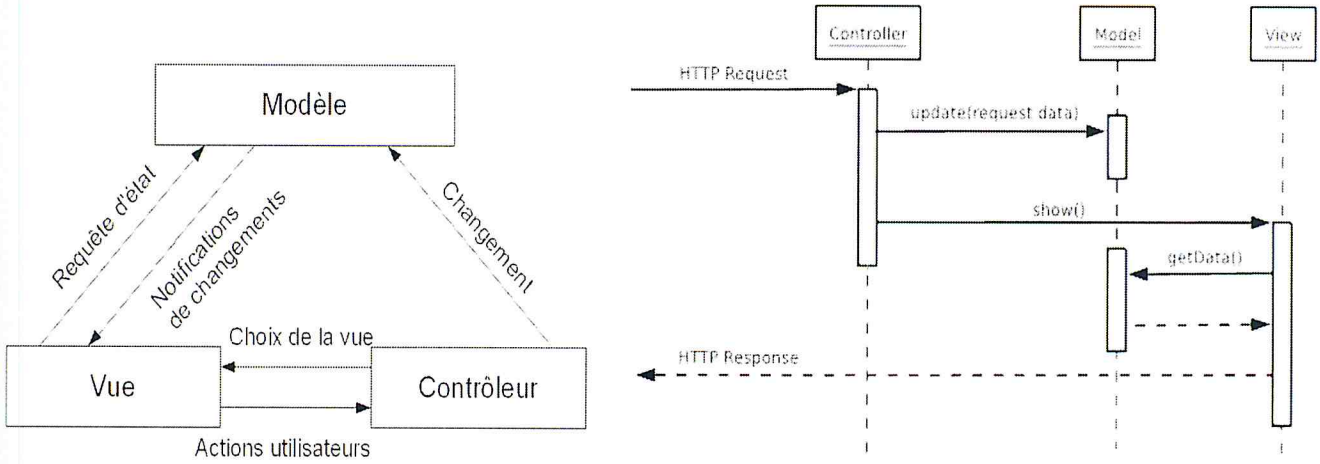
a. **Modèle** : gère les données et reprend la logique métier (le modèle lui-même peut être décomposé en plusieurs couches mais cette décomposition n'intervient pas au niveau de MVC). Le modèle ne prend en compte aucun élément de présentation.

b. **Vue** : elle affiche les données, provenant exclusivement du modèle, pour l'utilisateur et/ou reçoit ses actions. Aucun traitement, autre que la gestion de présentation, n'y est réalisé.

c. **Contrôleur**: son rôle est de traiter les événements en provenance de l'interface utilisateur et les transmet au modèle pour le faire évoluer ou à la vue pour

modifier son aspect visuel (pas de modification des données affichées mais des modifications de présentation (couleur de fond, affichage ou non de la légende d'un graphique…). Le contrôleur connaît la (les) vu(es) qu'il contrôle ainsi que le modèle.

Le contrôleur pourra appeler des méthodes du modèle pour réagir à des événements (demande d'ajout d'un client par exemple), il pourra faire modifier à la vue son aspect visuel. et instancier de nouvelles vues (demande d'affichage de telle ou telle info). Pour faire cela, le contrôleur sera à l'écoute d'événements survenant sur les vues. La vue observera le modèle qui l'avertira du fait qu'une modification est survenue. Dans ce cas, la vue interrogera le modèle pour obtenir son nouvel état. [12]



FigureIII.1 Interactions entre les différentes couches du modèle MVC

4. Présentation des diagrammes UML utilisés

UML permet de présenter et de manipuler les concepts objet, et de faire une démarche d'analyse qui permet de concevoir une solution de manière itérative grâce aux diagrammes, et d'exprimer visuellement une solution objet. Il se caractérise comme un langage de modélisation graphique et textuel qui est une étape importante du cycle de développement des systèmes utilisé ainsi pour visualiser, comprendre et définir des besoins, spécifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. Ces modèles doivent être proche de la réalité.[11]

4.1. Diagramme de cas d'utilisation (use case)

Le Diagramme de cas d'utilisation est le premier diagramme du modèle UML utilisé pour la modélisation des besoins des utilisateurs.

Les cas d'utilisations décrivent le comportement du système étudié du point de vue de l'utilisateur, et décrivent les possibilités d'interactions fonctionnelles entre le système et les acteurs Ils permettent de définir les limites et les relations entre le système et son environnement. Il est destiné à structurer les besoins des utilisateurs et les objectifs par rapport au système. C'est donc l'image d'une fonctionnalité en réponse à la simulation d'un acteur externe. [12]

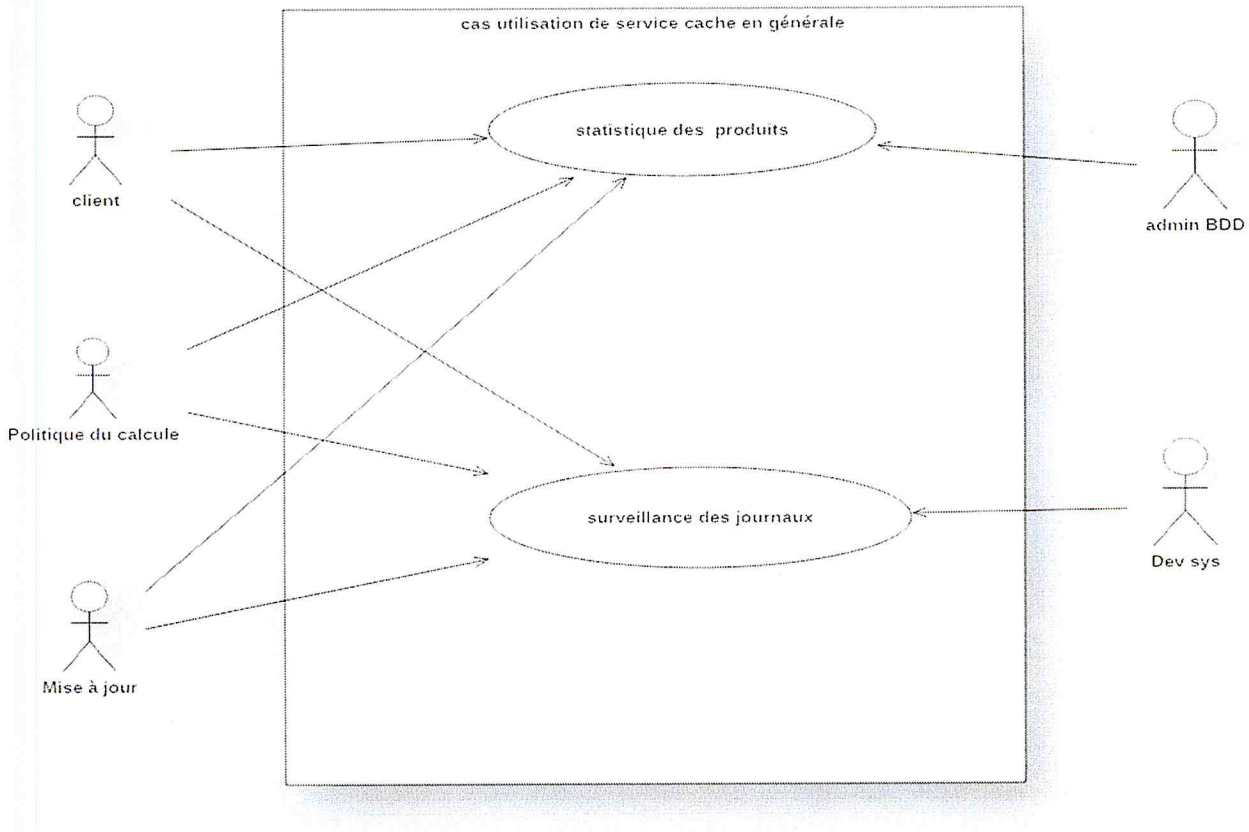
4.1.1 Liste des acteurs :

Acteur	Description et Rôle
Client d'affichage	Les applications d'affichage qui utilisent le service cache.
Politique du calcul	Cet acteur est le responsable de la politique utilisée pour calculer les statistiques.
Ordonnanceur	Processus qui gère les ordonnancements (temps de déclenchement des calculs).
Administrateur BDD	Responsable de maintenance des schémas des bases de données
Développeur du système	Le développeur du système MESSIR et le seul qui a le droit de modifier le format des fichiers journaux.
Opérateur	Tous les acteurs humains (administrateur ou utilisateur) ont le droit d'accéder à l'application d'affichage.
Administrateur	Administrateur d'application d'affichage
Utilisateur	Utilisateur d'application d'affichage

Tableau III.2 List des acteurs

4.1.2 Diagramme cas utilisation de service cache

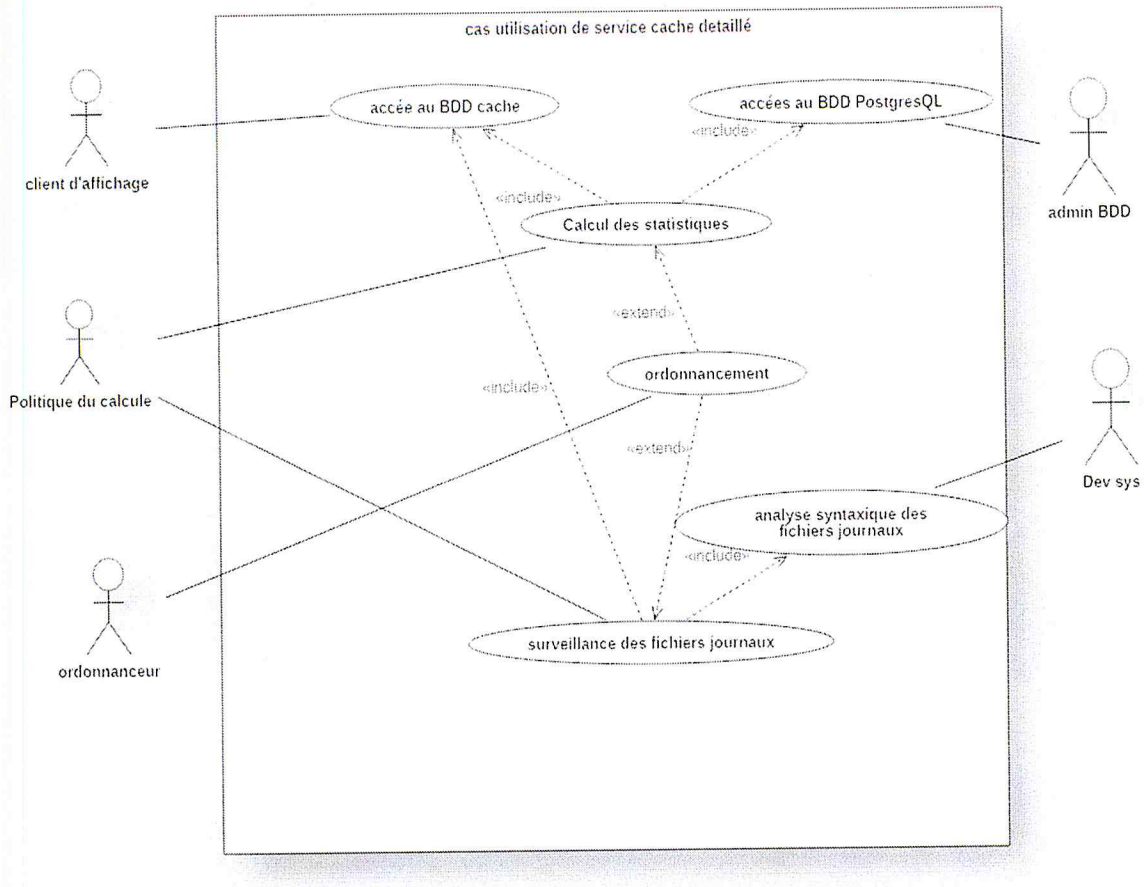
4.1.2.1 Diagramme cas utilisation générale



FigureIII.2 Diagramme cas utilisation de l'application service cache en générale

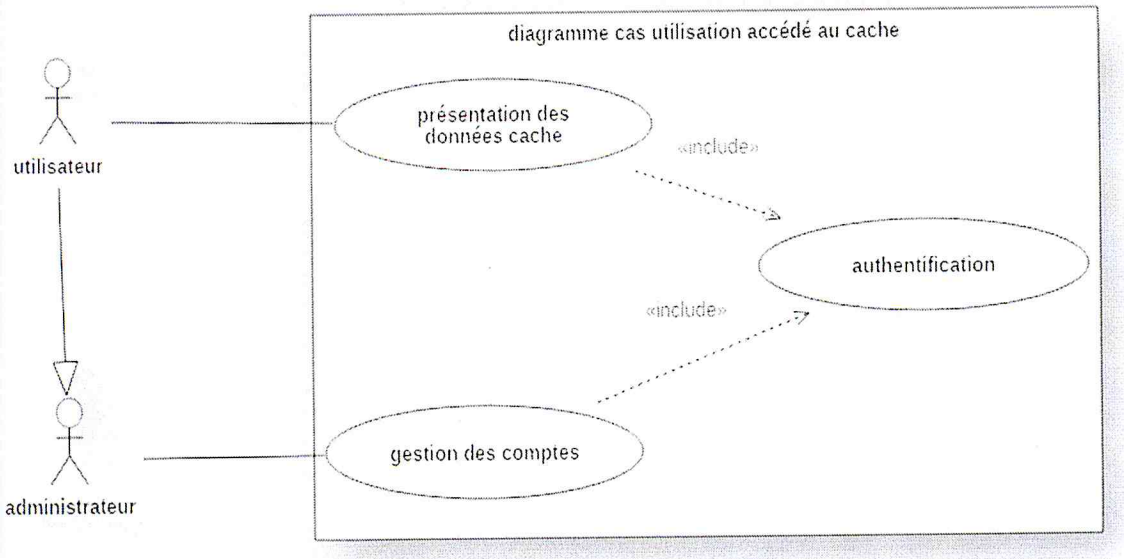
4.1.2.2 Diagramme cas utilisation détaillé du « service cache »

Nous appliquons le premier principe du SOLID pour obtenir un diagramme détaillé, chaque module doit servir un seul acteur par ce que chaque acteur demande des changements dans les modules. Si un module servira plus d'un acteur, leurs demandes de changements peuvent induire à des conflits.



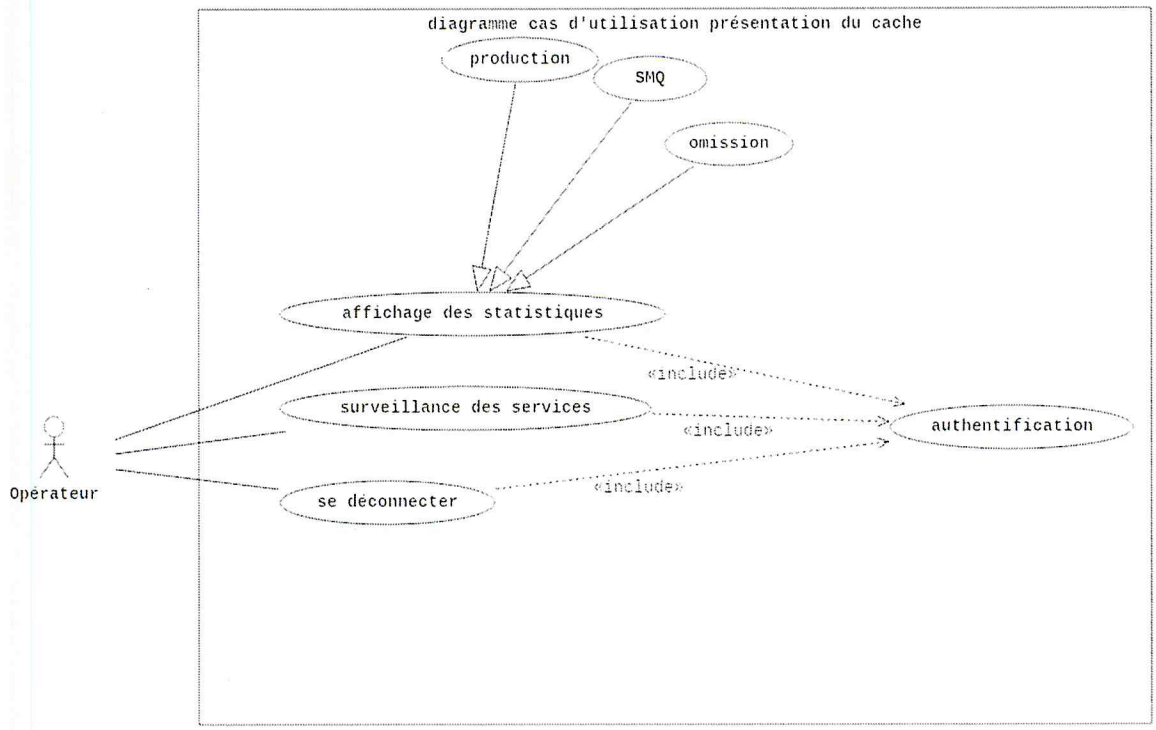
FigureIII.3 cas utilisation détaillé de l'application service cache

4.1.3 Diagramme cas d'utilisation « accéder au cache »:



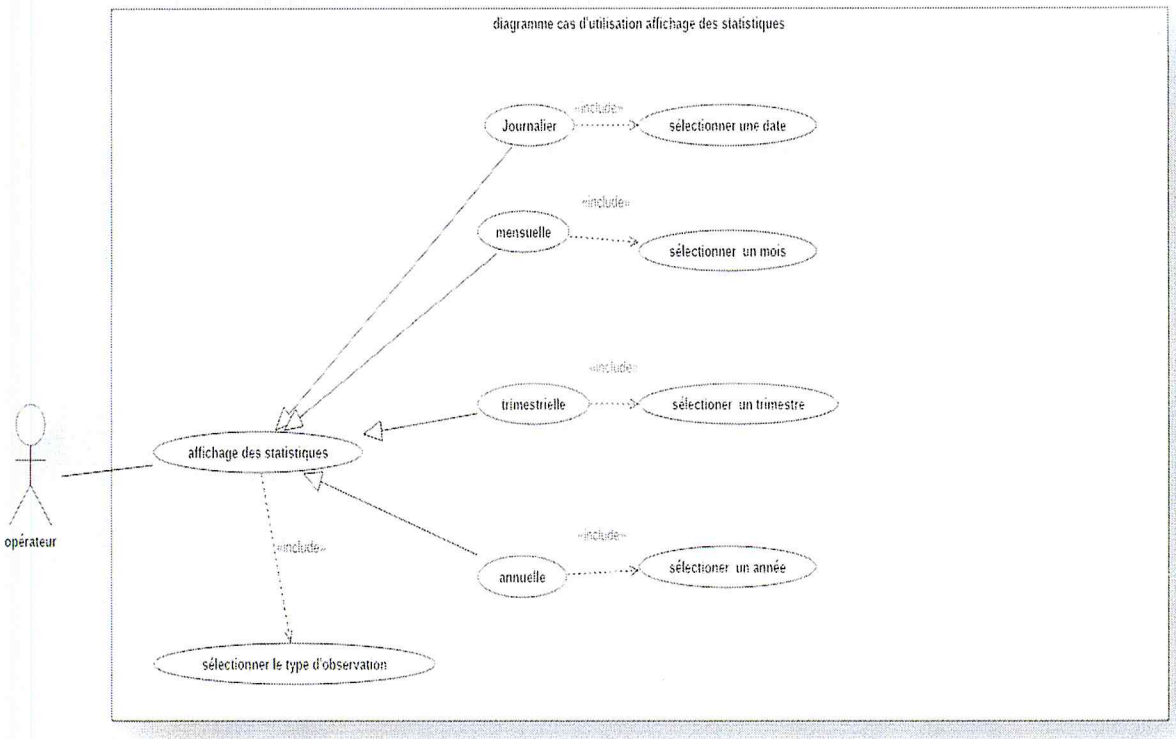
FigureIII.4 diagramme cas utilisation accédé au cache

4.1.4 Diagramme cas d'utilisation « présentation des données cache »:



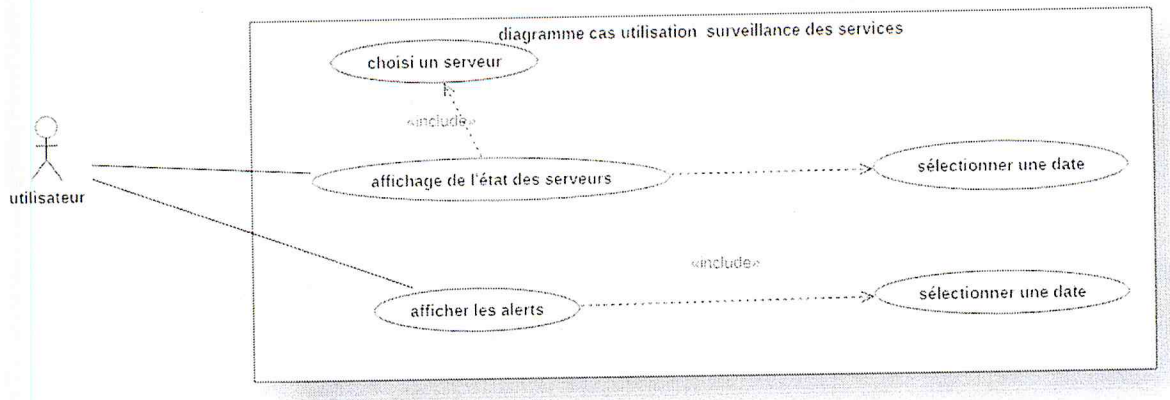
FigureIII.5 Diagramme cas d'utilisation « présentation des données cache »

4.1.5 Diagramme cas utilisation « affichage des statistique »



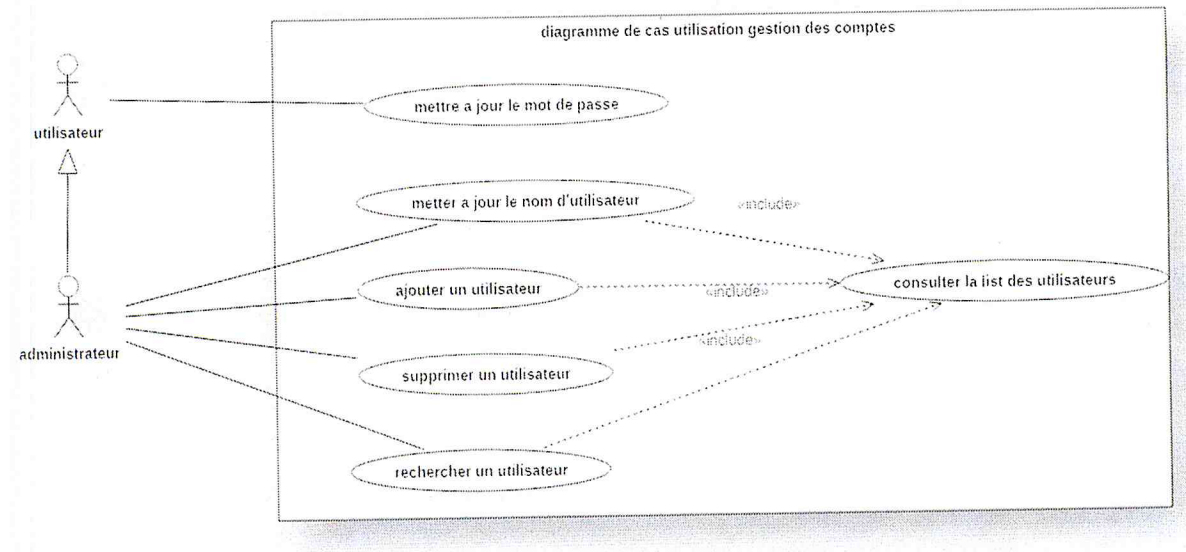
FigureIII.6 diagramme cas utilisation affichage des statistiques

4.1.6 Diagramme cas utilisation « surveillance des services »



FigureIII.7 diagramme cas utilisation « surveillance des services »

4.1.7 Diagramme cas utilisation « gestion des comptes »



FigureIII.8 Diagramme cas utilisation « gestion des comptes »

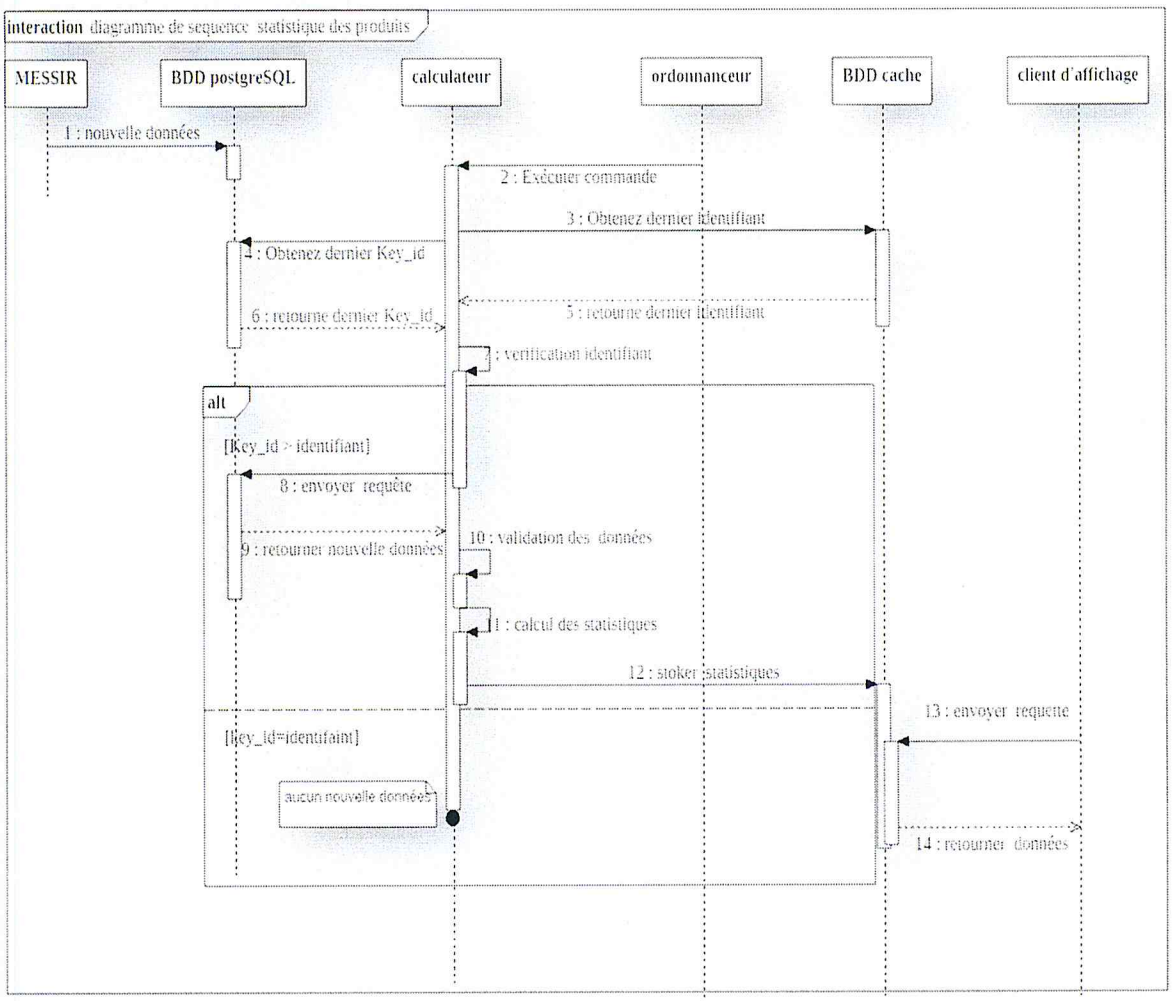
4.2 Diagramme de séquence

Le diagramme de séquence suit le diagramme de cas d'utilisation car il le complète. Il permet de décrire les scénarios (déroulement des traitements entre les éléments du système et les acteurs) de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. En particulier, il montre aussi les objets qui participent à l'interaction par leur « ligne de vie » et les messages qu'ils échangent présentés en séquence dans le temps. Voici quelques notions de base du diagramme :

- **Scénario :** Une liste d'actions qui décrivent une interaction entre un acteur et le système.
- **Interaction :** Un comportement qui comprend un ensemble de messages échangés par un ensemble d'objets dans un certain contexte pour accomplir une certaine tâche.
- **Message :** Un message représente une communication unidirectionnelle entre objets qui transporte de l'information avec l'intention de déclencher une réaction chez le récepteur. [13]

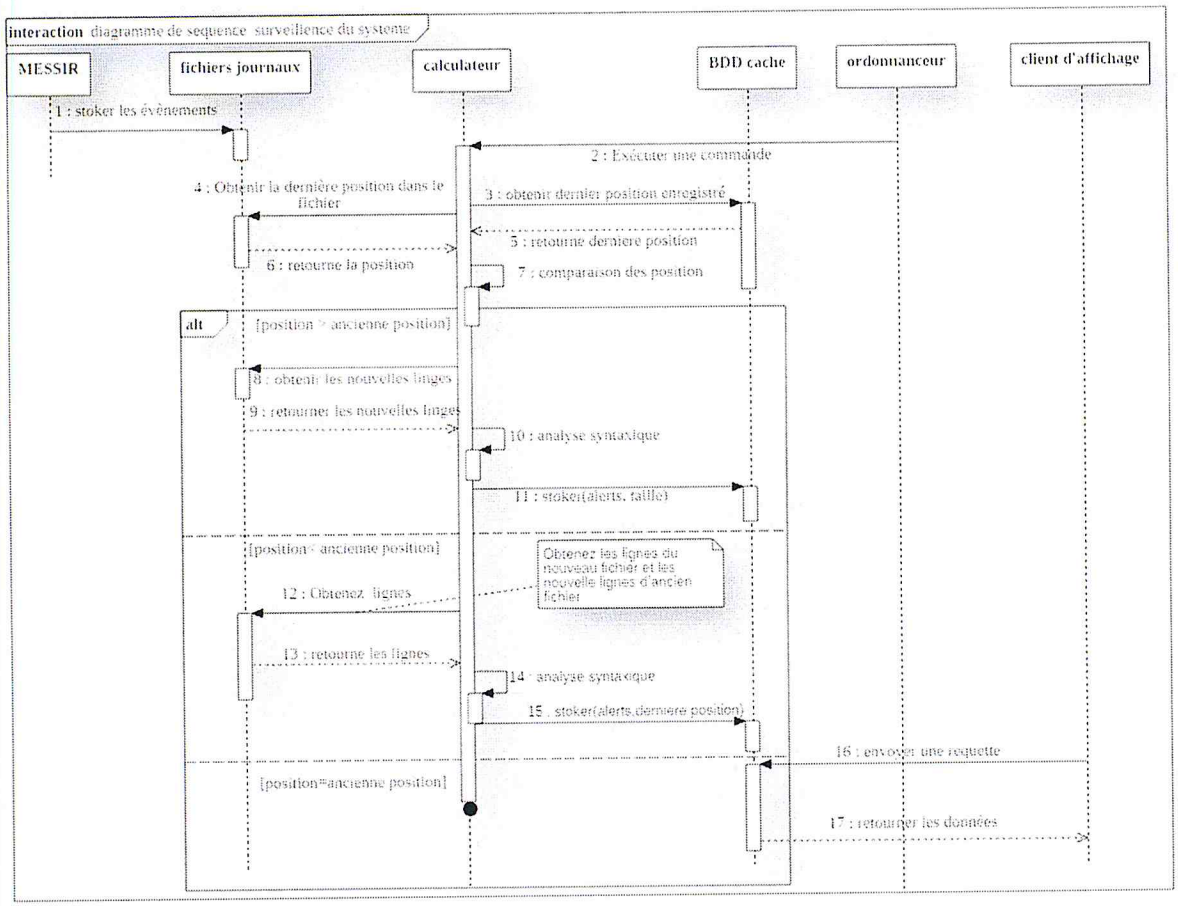
4.2.1 Diagramme de séquence « service cache »

4.2.1.1 Diagramme de séquence « statistique des produits »



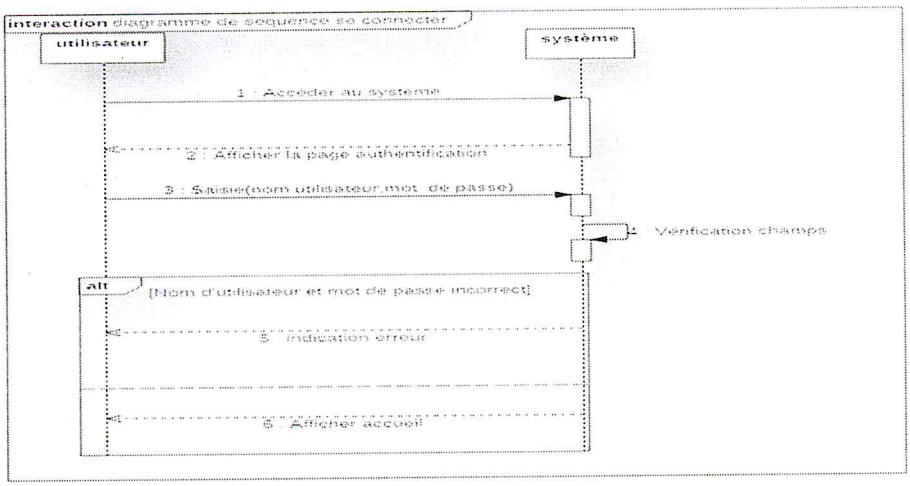
FigureIII.9 Diagramme de séquence « statistique des produits »

4.2.1.2 Diagramme de séquence « surveillance des services »



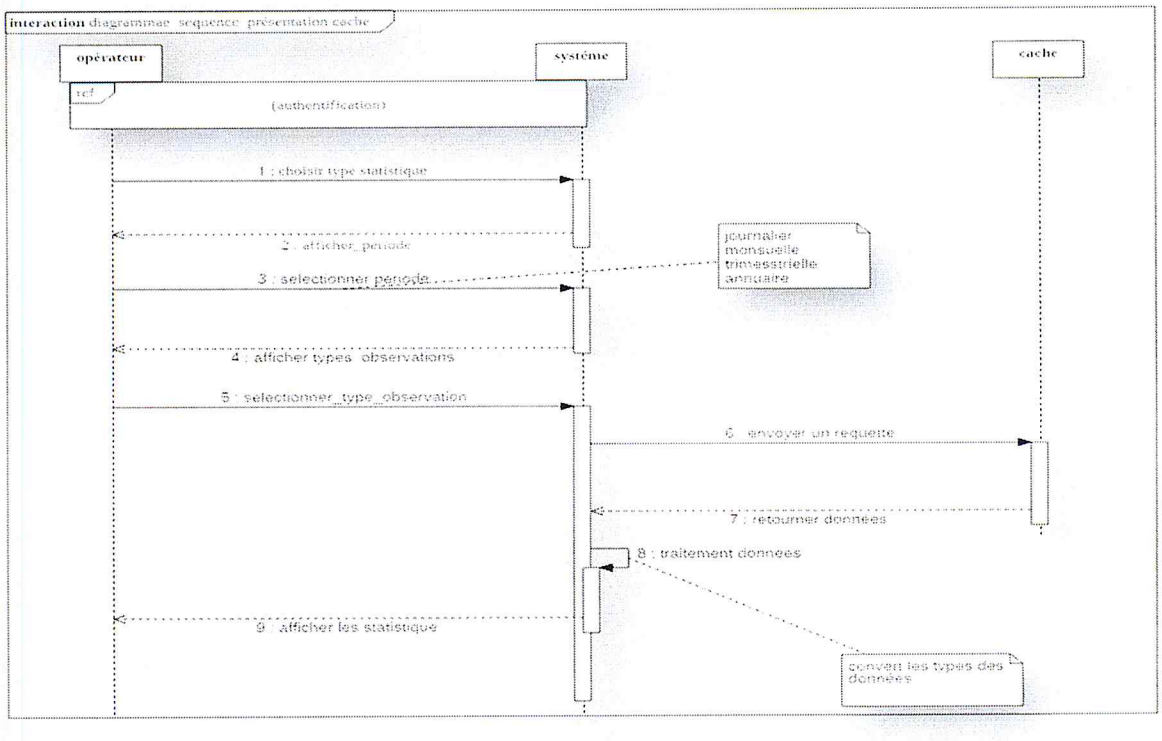
FigureIII.10 Diagramme de séquence « surveillance des services »

4.2.2 Diagramme de séquence « se connecter »



FigureIII.11 Diagramme de séquence « authentification »

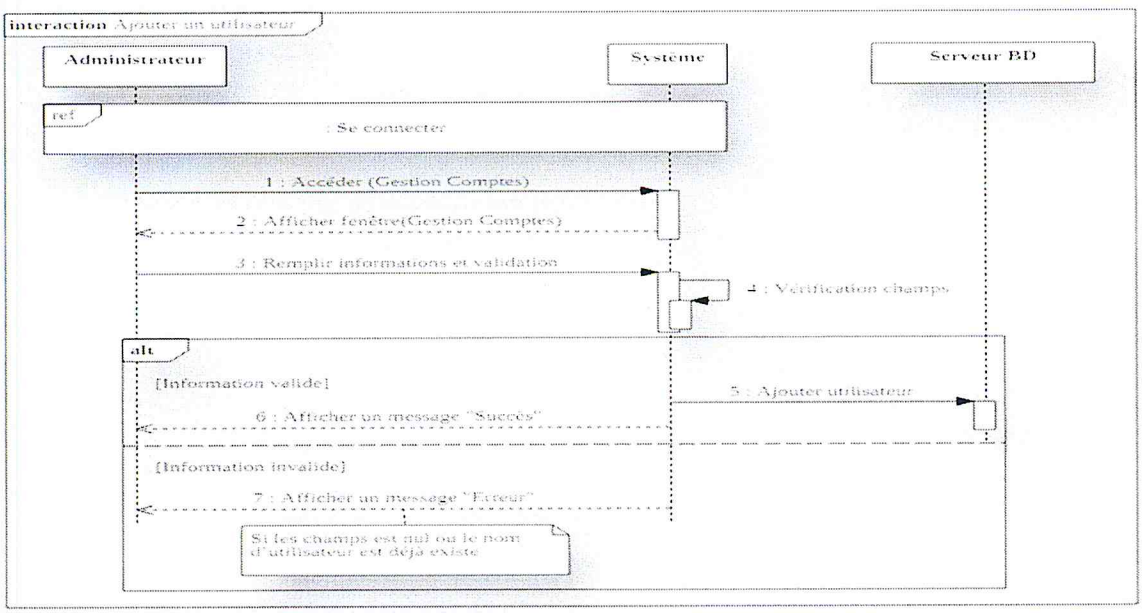
4.2.3 Diagramme de séquence d'un client d'affichage



FigureIII.12 Diagramme de séquence présentation cache

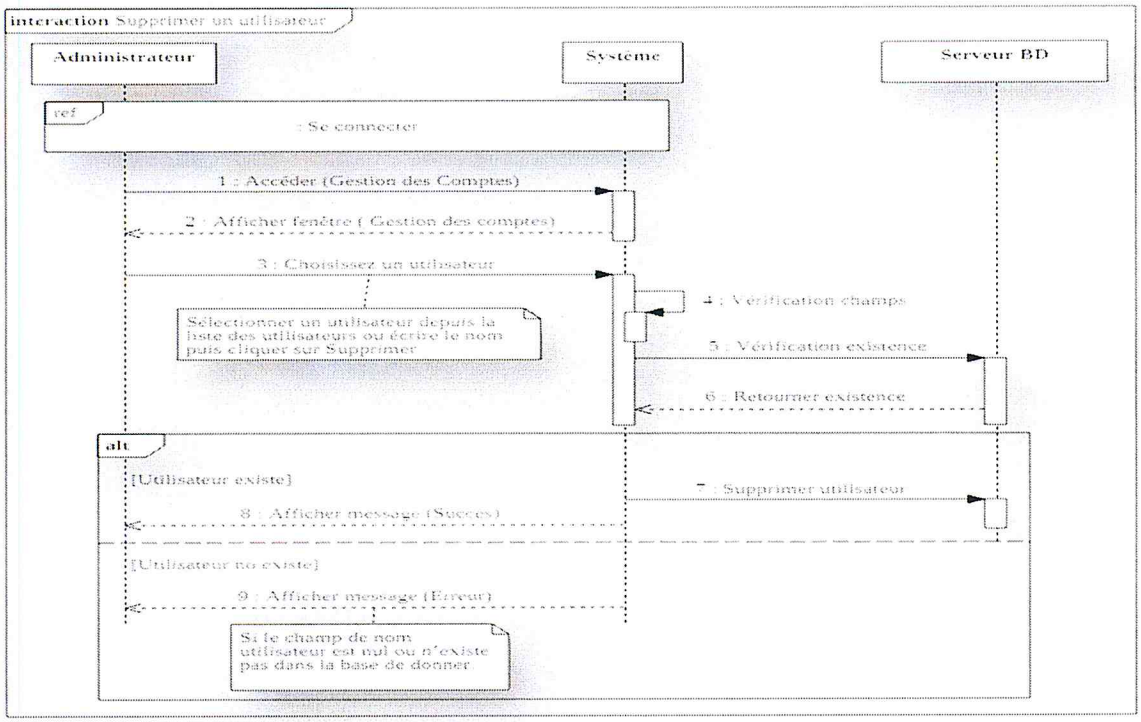
4.2.4 Diagramme de séquence « gestion des comptes »

4.2.4.1 Diagramme de séquence « ajouter un utilisateur »



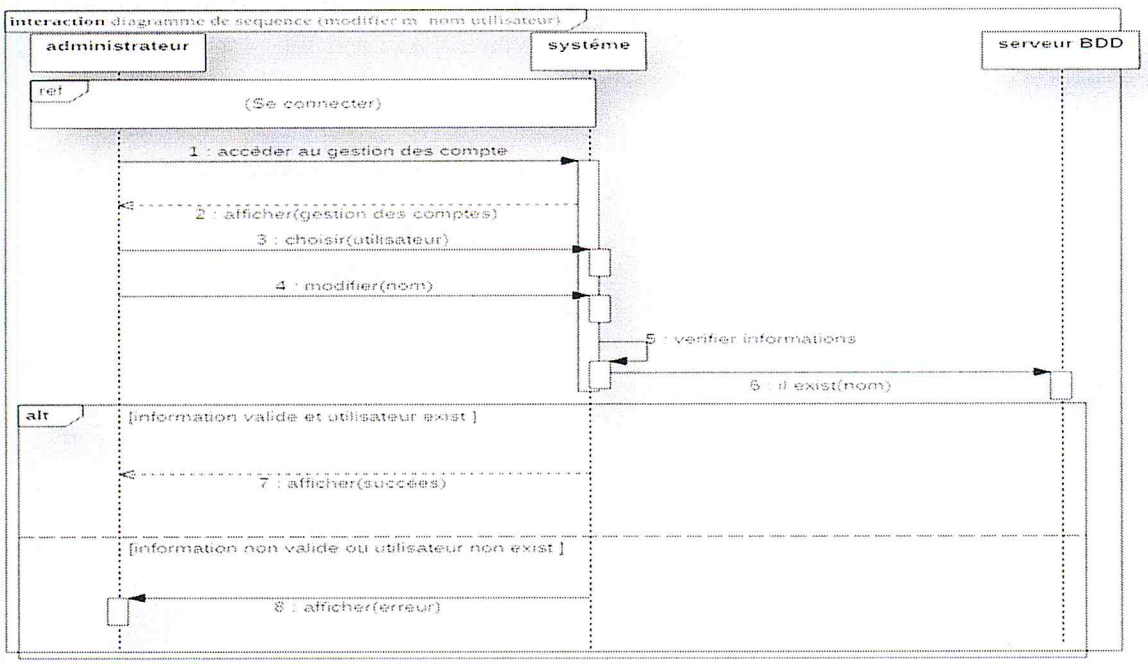
FigureIII.13 Diagramme de séquence ajouter un utilisateur

4.2.4.2 Diagramme de séquence « supprimer un utilisateur »



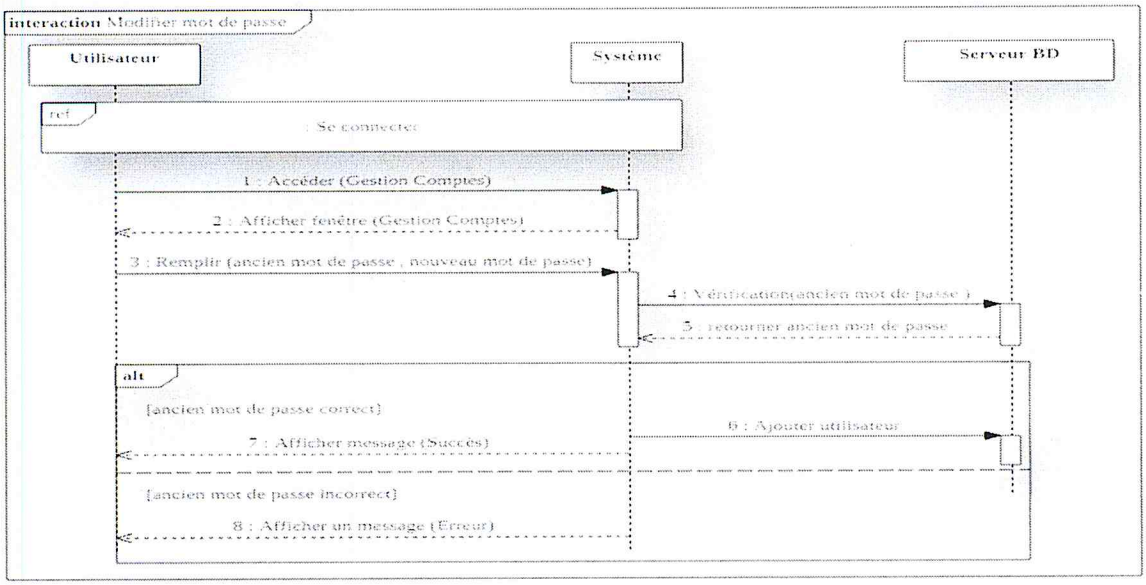
FigureIII.14 diagramme de séquence supprimer un utilisateur

4.2.4.3 Diagramme de séquence « modifier le nom d'un utilisateur »



FigureIII.15 Diagramme de séquence « modifier le nom d'un utilisateur »

4.2.4.4 Diagramme de séquence « modifier le mot de passe d'un utilisateur »



FigureIII.16 Diagramme de séquence « modifier le mot de passe d'un utilisateur »

4.3 Diagramme de classe

Le diagramme de classe constitue un élément très important de la modélisation. Il permet de définir quelles seront les composantes du système final. Il représente les classes intervenant dans le système. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

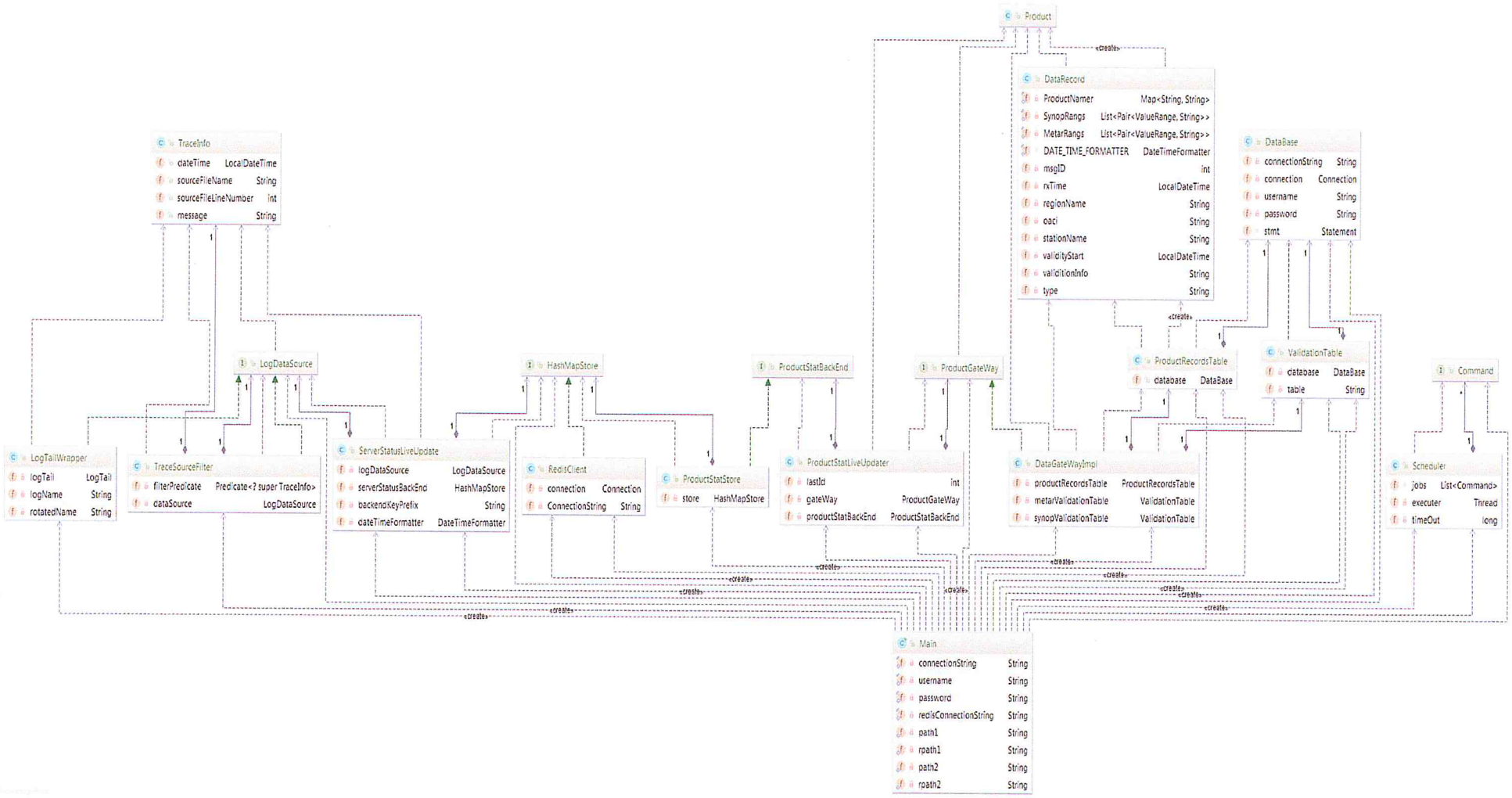
4.3.1- Service cache

Nous devons faire le diagramme de classe qui permet de vérifier la correspondance aux principes SOLID :

- **Table des symboles :**

Symbole	Signification
	classe.
	interface.
	Méthode
	Champs
	implémenté un interface.
create	création d'un instance d'un classe
	Utilisé

Tableau III.3 Signification des symboles



FigureIII.17 Diagramme de classe

- **Dictionnaire des champs :**

champ	Type	Description
msgId	Integer	Nombre d'observation
rxTime	LocalDateTime	La date reçue d'observation
RegionName	String	La région de station qui a envoyé l'observation
Oaci	String	Code station
stationName	String	Nom de station
validityStar	LocalDateTime	l'heure prévue d'observation
validitionInfo	string	Horaire de station
Type	String	Type de message (SI,SM ,SA,…..)
dateTime	LocalDateTime	Date d'alerte
SourceFileName	String	le nom du fichier source code du système MESSIR
sourceFileLineNumber	Integer	Le Nombre de la linge sur le fichier source code du système MESSIR
Message	String	Message d'alerte

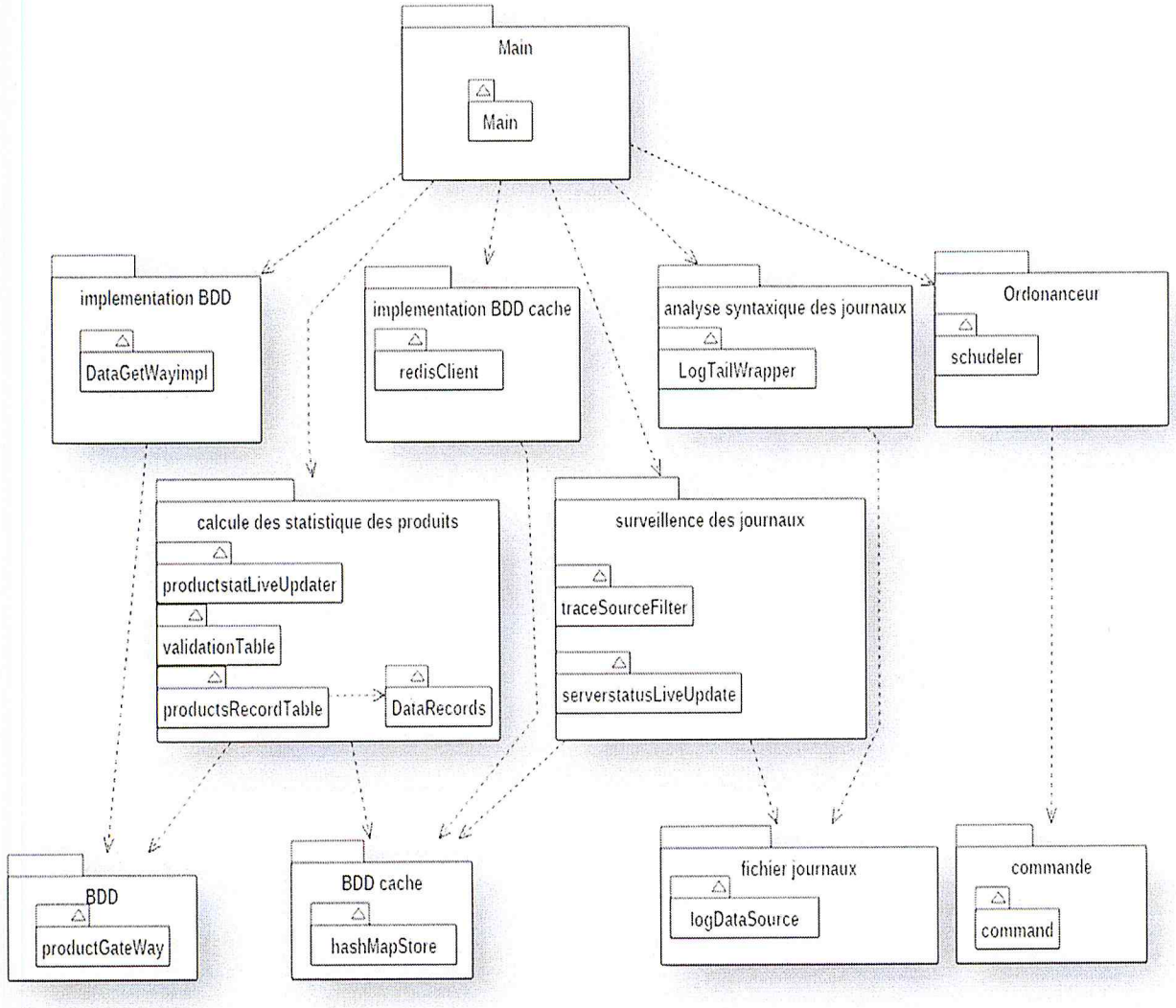
Tableau III.4 Dictionnaire des champs

4.4 Diagramme de package

Un package en UML (ou *paquetage* en français) sert à grouper des éléments en un ensemble cohérent, et à fournir un espace de noms pour ces éléments. Un package peut contenir la plupart des éléments UML : classes, objets, cas d'utilisations, composants, etc. Il peut également contenir d'autres packages, selon une organisation hiérarchique. L'intérêt des packages est de permettre de structurer les diagrammes et de donner une vision globale plus claire. [14]

4.4.1 Diagramme de package service cache :

Un package a un seul ensemble cohérent de fonctionnalités.



FigureIII.17 Diagramme de package service cache

5. Conclusion

Dans ce chapitre, nous avons présenté la conception de notre système pour les calculs statistiques et la surveillance du système. Pour cela, nous nous sommes basés sur les principes SOLID et un certain nombre de diagrammes UML. Le chapitre suivant est dédié à la réalisation de notre solution et les aspects techniques de notre système.

Chapitre IV :

Réalisation du système

1. Introduction

Après avoir élaboré la conception de notre système, nous abordons dans ce chapitre le dernier volet de ce mémoire, qui a pour objectif d'exposer la phase de réalisation. Cette dernière est considérée comme étant la concrétisation finale de toute la méthode de conception. Après avoir présenter l'environnement technique (outils et langages utilisés) et le processus de développement adopté, nous présentons la réalisation des différentes partie de notre système à savoir le service cache et les deux applications d'affichage (application web et application desktop). Pour ces deux dernières, nous présentons un certain nombre d'interfaces pour illustrer le fonctionnement de quelques fonctionnalités du système.

2. Etude technique

L'étude technique est une phase d'adaptation de conception à l'architecture technique. Elle a pour objectif de décrire au plan fonctionnel la solution à réaliser d'une manière détaillée ainsi que la description des traitements.

2.1 Environnement de réalisation

Pour la réalisation de notre système, nous avons eu recours à plusieurs moyens matériels et logiciels que nous décrivons dans ce qui suit.

2.1.1 Matériels de base

Le développement de l'application est réalisé via deux ordinateurs portables ayant les caractéristiques suivantes :

Caractéristique	DELL LATTITUDE E6440	SONY VAIO VPCEC4L1E
Marque	DELL	SONY
Processeur	Intel® core™ i5-4200M CPU	Intel® core™ i3 CPU
RAM	12 GO	4 GO
Système d'exploitation	Windows 7 64 bit	Windows 8.1 Entreprise 64 bit

Tableau IV.1 : Matériel de base

2.1.3 Choix des langages de développement et de BDD RAM

1- Java :

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par Sun Microsystems. Il permet une programmation orientée-objet (à l'instar de Smalltalk et, dans une moindre mesure, C++), modulaire (langage ADA) et reprend une syntaxe très proche de celle du langage C. Outre son orientation objet, le langage Java a l'avantage d'être modulaire (on peut écrire des portions de code génériques, c.-à-d. utilisables par plusieurs applications), rigoureux (la plupart des erreurs se produisent à la compilation et non à l'exécution) et portable (un même programme compilé peut s'exécuter sur différents environnements).[15]

2- PHP :

PHP (officiellement, ce sigle est un acronyme récursif pour PHP Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML. [16]

3- Java FX :

Java FX est une technologie créée par Sun Microsystems qui appartient désormais à Oracle, à la suite du rachat de Sun Microsystems par Oracle le 20 avril 2009. Avec l'apparition de Java 8 en mars 2014, Java FX devient la bibliothèque de création d'interface graphique officielle du langage Java, pour toutes les sortes d'application (applications mobiles, applications sur poste de travail, applications Web), le développement de son prédécesseur Swing étant abandonné (sauf pour les corrections de bogues). [17]

Java FX est désormais une pure API Java (le langage de script spécifique qui a été un temps associé à Java FX est maintenant abandonné). Java FX contient des outils très divers, notamment pour les médias audio et vidéo, le graphisme 2D et 3D, la programmation Web, la programmation multi-fils etc.

4- JavaScript :

Le JavaScript est un langage informatique utilisé dans le développement des pages web. Ce langage a la particularité de s'activer sur le poste client, Autrement dit, c'est votre ordinateur qui va recevoir le code et qui devra l'exécuter. C'est en opposition à d'autres langages qui sont activé côté serveur. L'exécution du code est effectuée via un navigateur internet tel que Firefox ou Internet Explorer. [18]

5- CSS :

CSS est l'acronyme de Cascading Style Sheets, c'est un langage de feuille de style utilisé pour décrire la mise en forme d'un document écrit avec un langage de balisage. Il permet aux concepteurs de contrôler l'apparence et la disposition de leurs pages web.[19]

6- JSON :

JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple. Créé par Douglas Crockford entre 2002 et 2005
Un document JSON a pour fonction de représenter de l'information accompagnée d'étiquettes permettant d'en interpréter les divers éléments, sans aucune restriction sur le nombre de celles-ci. Un document JSON ne comprend que deux types d'éléments structurels :

- des ensembles de paires « nom » (alias « clé ») / « valeur » ;
- des listes ordonnées de valeurs.

Ces mêmes éléments représentent trois types de données :

- des objets ;
- des tableaux ;
- des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null. [20]

7- FXML :

FXML n'a pas de schéma, mais il a une structure prédéfinie de base. Ce que vous pouvez exprimer dans FXML, et comment cela s'applique à la construction d'un graphe de scène, dépend de l'API des objets que vous construisez. Parce que FXML est directement lié à Java, vous pouvez utiliser la documentation de l'API pour comprendre quels éléments et attributs sont autorisés. [32] En général, la plupart des classes JavaFX peuvent être utilisées en tant qu'éléments, et la plupart des propriétés Bean peuvent être utilisées en tant qu'attributs.

8- REDIS :

Redis est un serveur de structure de données avec un ensemble de données en mémoire pour la vitesse. C'est ce qu'on appelle une donnée serveur de structure et pas simplement un magasin de valeurs clés, car Redis implémente des structures de données autoriser les clés à contenir des chaînes de sécurité binaires, des hachages, des ensembles et des ensembles triés, ainsi en tant que listes. Cette combinaison de flexibilité et de rapidité fait de Redis l'outil idéal pour applications. Redis a débuté début 2009 en tant que magasin de valeur clé développé par

Salvatore Sanfilippo. Aujourd'hui, Redis est utilisé par les entreprises, grandes et petites, qui effectuent des tâches à la fois petites et grandes.

Redis est un excellent choix pour maintenir des données en mémoire pour un accès en temps réel très rapide. C'est une forme de remplacement d'un cache tel que mémoire cache pour offrir une plus grande richesse fonctionnelle et une manipulation de structures de données plus riches. [21]

9- PostgreSQL :

PostgreSQL (prononcé "post-gress-Q-L") est un système de gestion de base de données relationnelle open source (SGBD) développé par une équipe mondiale de bénévoles. PostgreSQL n'est contrôlé par aucune société ou autre entité privée et le code source est disponible gratuitement.

PostgreSQL prend en charge les transactions, les sous-sélections, les déclencheurs, les vues, l'intégrité référentielle des clés étrangères et le verrouillage sophistiqué. Il fonctionne sur de nombreuses plateformes, y compris Linux, la plupart des versions d'UNIX, Mac OS X, Solaris, Tru64 et Windows. Il supporte le texte, les images, les sons et la vidéo, et inclut des interfaces de programmation pour C / C ++, Java, Perl, Python, Ruby, Tcl et ODBC (Open Database Connectivity).[22]

10- MYSQL :

MySQL(My Structured Query Langage-Langage de requêtes structuré) est un système de gestion de base de données relationnelle open source (SGBDR) basé sur SQL (Structured Query Language). MySQL fonctionne sur presque toutes les plateformes, y compris Linux, UNIX et Windows. Bien qu'il puisse être utilisé dans un large éventail d'applications, MySQL est le plus souvent associé aux applications Web et à la publication en ligne et est un composant important d'une pile d'entreprise open source appelée LAMP. LAMP est une plate-forme de développement Web qui utilise Linux comme système d'exploitation, Apache comme serveur Web, MySQL comme système de gestion de base de données relationnelle et PHP comme langage de script orienté objet. (Parfois, Perl ou Python est utilisé à la place de PHP.) [23]

3. Outils de développement

3.1 IntelliJ IDEA :

IntelliJ IDEA est un environnement de programmation spécial ou un environnement de développement intégré (IDE) largement destiné à Java. Cet environnement est utilisé notamment pour le développement de programmes. Il est développé par une compagnie appelée JetBrains, qui s'appelait officiellement IntelliJ. Il est disponible en deux éditions: l'édition communautaire sous licence Apache 2.0 et une édition commerciale connue sous le nom d'édition ultime. Les deux peuvent être utilisés pour créer un logiciel qui peut être vendu. Ce qui rend IntelliJ IDEA si différent de ses homologues, c'est sa facilité d'utilisation, sa flexibilité et son design solide. [24]

3.2 JDK

Le Java Développement Kit (JDK) est un environnement de développement logiciel utilisé pour le développement d'applications et d'applets Java. Il inclut Java Runtime Environment (JRE), un interpréteur / loader (java), un compilateur (javac), un archiver (jar), un générateur de documentation (javadoc) et d'autres outils nécessaires au développement Java. [25]

3.3 JUNIT

JUnit est un Framework de test unitaire pour le langage de programmation Java. JUnit a joué un rôle important dans le développement du développement piloté par les tests, et fait partie d'une famille de Framework de tests unitaires connus sous le nom de xUnit, qui est né avec JUnit.

3.3 PHPStorm

PhpStorm est un éditeur pour PHP, HTML et JavaScript, édité par JetBrains. Il permet d'éditer du code PHP 5.3, 5.4, 5.5, 5.6 et 7.0. Il possède :

- Une coloration syntaxique ;
- Affichage des erreurs à la volée ;
- Auto-complétion intelligente du code ;
- Ré usinage du code.

Il intègre :

- L'envoi des fichiers via FTP ;
- Un gestionnaire de version.

Il permet aussi de visualiser l'architecture de bases de données de différentes sources (MySQL, SQLite, Redis, ...). [26]

3.4 StarUML

C'est un logiciel de modélisation UML open source. StarUML gère la plupart des diagrammes spécifiés dans la norme UML 2.0.

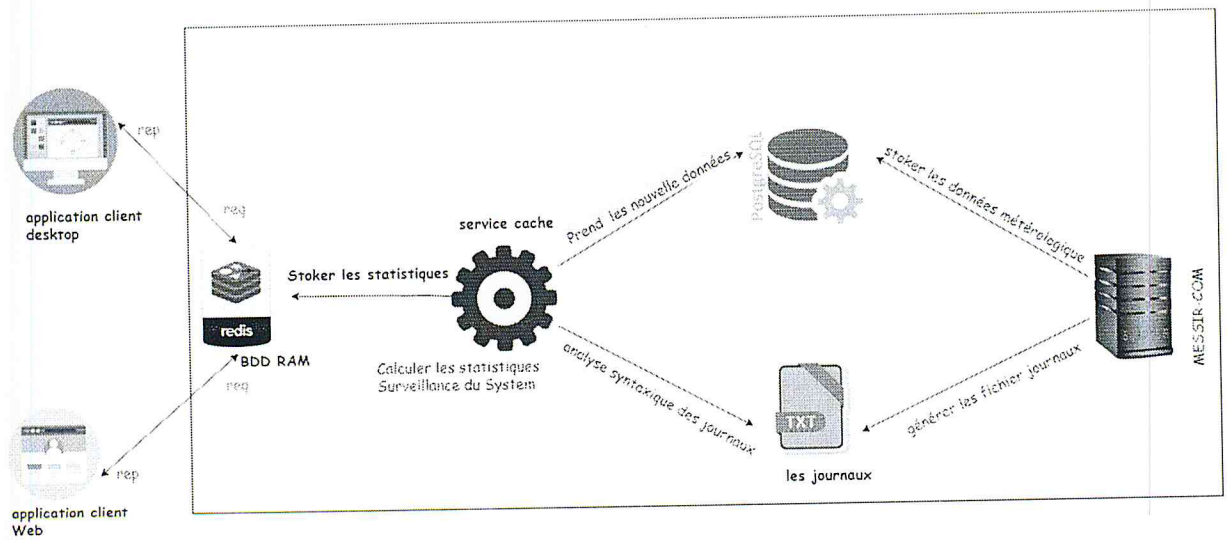
3.5 Scene Builder

JavaFX Scene Builder est un outil de présentation visuelle qui permet aux utilisateurs de concevoir rapidement des interfaces utilisateur d'application JavaFX, sans codage. Les utilisateurs peuvent glisser et déposer des composants de l'interface utilisateur dans une zone de travail, modifier leurs propriétés, appliquer des feuilles de style et le code FXML de la mise en page qu'ils créent est automatiquement généré en arrière-plan. Le résultat est un fichier FXML qui peut ensuite être combiné avec un projet Java en liant l'interface utilisateur à la logique de l'application. [27]

3.6 Bootstrap

Bootstrap est un Framework CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript.[33] Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation… Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées.

4. Mise en œuvre de notre système



FigureIV.1 Architecture du système réalisé.

4.1 La réalisation de l'application Service Cache

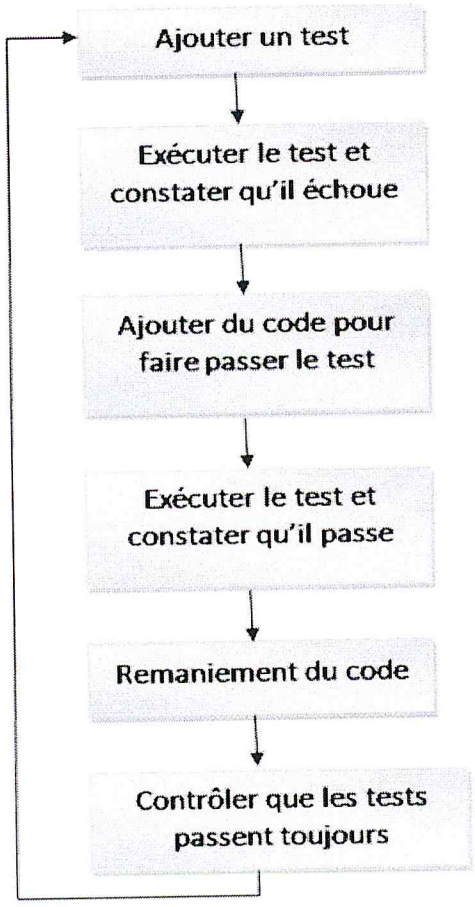
Pour la réalisation de l'application Service Cache, nous nous sommes basés sur les principes SOLID présentés dans le chapitre précédent et la méthode TDD que nous présentons dans ce qui suit :

4.1.1 La méthode TDD

Le TDD (Test Driven Development) est comme son nom l'indique, une méthode de développement. Elle consiste à écrire les tests avant d'écrire le code. Le but est donc pour l'équipe projet de développer un logiciel construit par la validation de tests plutôt qu'un logiciel construit par le suivi des spécifications. Le TDD est donc une méthode de travail différente du BDD (Behavior Driven Development) où le développeur développe en fonction des spécifications fonctionnelles puis écrit les cas de tests unitaires, les autres niveaux de tests étant exécutés ultérieurement.

4.1.2 Fonctionnement du TDD

Le TDD fonctionne par cycle



FigureIV.2 Le cycle de fonctionnement TDD[30]

- On écrit d'abord un test.
- On l'exécute puis on valide que le test est en échec car la fonctionnalité n'a pas été implémentée. Si le cas de test est en succès, alors, s'il est mal été écrit il doit être remanié, sinon la fonctionnalité est déjà implémentée et aucun développement n'est nécessaire.
- On développe la fonctionnalité jusqu'à ce que le test soit en succès.
- On « nettoie » le code, pour garder un code propre, de bonne qualité et facilement maintenable.
- On ré-exécute les tests (celui du cycle et les anciens tests) et s'assure que l'ensemble des tests est encore en succès.
- On écrit un nouveau test en échec.

Le TDD repose sur l'écriture des tests à mettre en succès. On s'aperçoit également que c'est une méthode incrémentale demandant de nombreuses exécutions. L'automatisation des tests doit donc être envisagée en TDD (comme en SCRUM).

On parle souvent de RGR (Red Green Refactor) lorsque l'on parle du TDD. Red correspond à un cas en échec, Green à un cas en succès et Refactor au nettoyage du code pour améliorer la qualité. Cela est un bon résumé de la méthode TDD.

4.1.3 TDD tests unitaires

Lorsque l'on parle de TDD on parle généralement de TDD pour les tests unitaires. Avec la méthode du BDD les développeurs écrivent leur code puis les tests unitaires. Dans le cas des TDD c'est le contraire, il écrit les tests avant puis doit écrire son code afin de valider ces tests. Les TDD sur les tests unitaires sont assez faciles à implémenter, en effet, ces tests étant écrit (mais aussi généralement exécutés et maintenus) par les développeurs, seuls ces derniers doivent s'habituer à cette nouvelle méthode.

4.1.4 ATDD (Acceptance TDD)

Le principe ici est d'appliquer la méthode du TDD aux tests d'acceptance (aussi appelés tests métiers). Le cycle reste le même par contre les métiers fonctionnels (MOA, recette…) et techniques (développeurs…) doivent travailler sur ces tests. Cela oblige donc une adaptation au TDD par un plus grand nombre de personnes et encourage également fortement la communication entre ces équipes.

4.1.5 Pourquoi utiliser le TDD ?

Comme pour chaque méthode, le TDD a ses points forts et ses points faibles. Le TDD vise à assurer une bonne qualité du produit de par la certitude que les cas de tests existent et sont exécutés mais aussi par le « nettoyage » récurent du code. Cette méthode incrémentale s'adapte parfaitement aux autres méthodes de travail incrémentales (comme le SCRUM) et aux projets à fort potentiel d'automatisation.

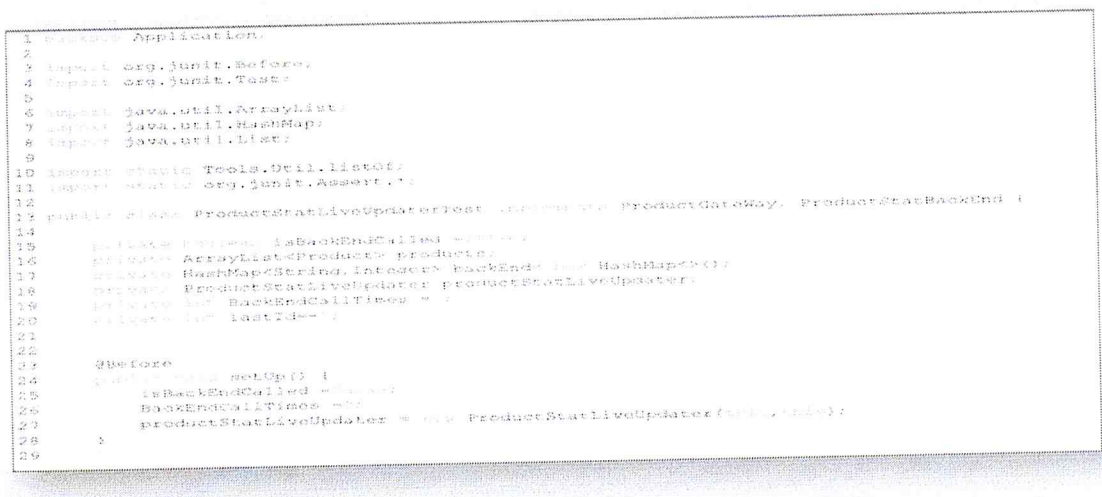
Le TDD est généralement bien perçu par les développeurs, ces derniers voyant par la validation d'un test la valorisation (et un accomplissement) de leur travail alors qu'en BDD, l'écriture du cas de tests arrivant après le développement, elle est souvent perçue comme une corvée.

Le TDD (pour les ATDD) encourage la communication et donc permet d'éviter de mauvaises interprétations.

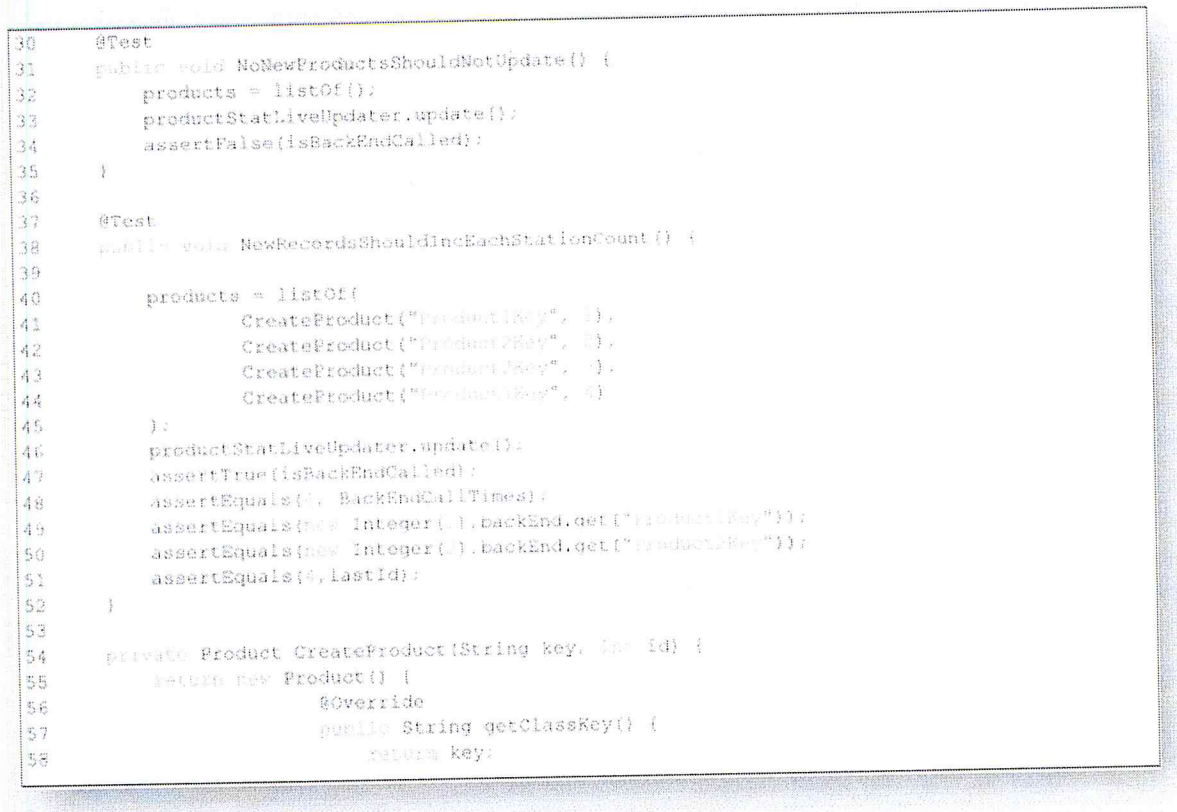
Enfin, effectuer régulièrement les cas de test permet de détecter plus tôt les bugs, ils sont donc moins cher à corriger.

4.1.6 Exemple d'une classe test réalisée avec la méthode TDD

- La classe ProductStatLiveUpdaterTest est le test de la classe ProductStatLiveUpdater.
- Nous utilisons le pattern « Self shunt [31]», que la classe-test implémente les interfaces utilises par la classe ProductStatLiveUpdater.
- Nous utilisons des données statiques pour éviter l'accès à la base de données à chaque exécution du test.

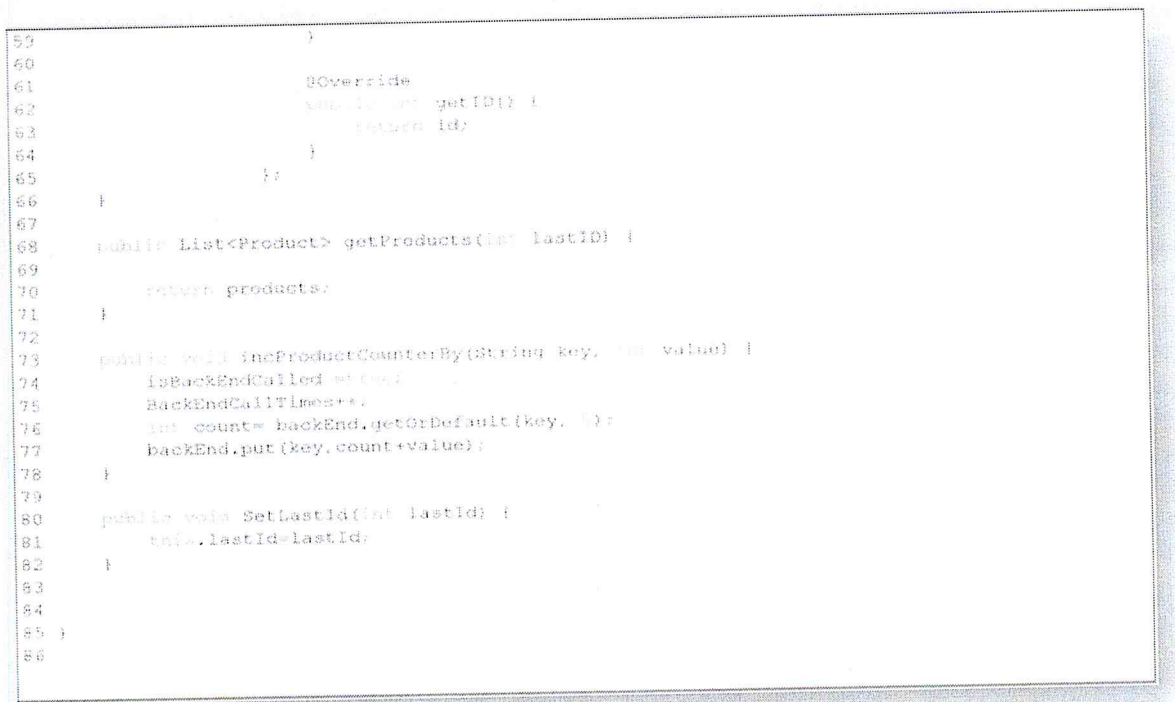


FigureIV.3 Exemple d'une classe Test partie1



```
30      @Test
31      public void NoNewProductsShouldNotUpdate() {
32          products = listOf();
33          productStatLiveUpdater.update();
34          assertFalse(isBackEndCalled);
35      }
36
37      @Test
38      public void NewRecordsShouldIncEachStationCount() {
39
40          products = listOf(
41                  CreateProduct("Product1Key", 1),
42                  CreateProduct("Product2Key", 3),
43                  CreateProduct("Product3Key", 2),
44                  CreateProduct("Product4Key", 4)
45          );
46          productStatLiveUpdater.update();
47          assertTrue(isBackEndCalled);
48          assertEquals(4, BackEndCallTimes);
49          assertEquals(new Integer(1),backEnd.get("Product1Key"));
50          assertEquals(new Integer(3),backEnd.get("Product2Key"));
51          assertEquals(4,lastId);
52      }
53
54      private Product CreateProduct(String key, int id) {
55          return new Product() {
56                      @Override
57                      public String getClassKey() {
58                          return key;
```

FigureIV.4 Exemple d'une classe test partie 2



```
59                      }
60
61                      @Override
62                      public int getID() {
63                          return id;
64                      }
65                  };
66      }
67
68      public List<Product> getProducts(int lastID) {
69
70          return products;
71      }
72
73      public void incProductCounterBy(String key, int value) {
74          isBackEndCalled = true;
75          BackEndCallTimes++;
76          int count= backEnd.getOrDefault(key, 0);
77          backEnd.put(key,count+value);
78      }
79
80      public void SetLastId(int lastId) {
81          this.lastId=lastId;
82      }
83
84
85  }
86
```

FigureIV.5 Exemple d'une classe test partie3

4.2 Calculs des statistiques météorologiques

4.2.1 Traitement des produits

Avant de calculer les statistiques, nous devons :

a. Récupérer les nouvelles données stockées dans la base de données PostgreSQL :

Nous créons une structure de données DataRecord :

```
DataRecord(int msgID, LocalDateTime rxTime, String regionName, String oaci, String stationName,
                      LocalDateTime validityStart, String validitionInfo, String type) {
    this.msgID = msgID;
    this.rxTime = rxTime;
    this.regionName = regionName;
    this.oaci = oaci;
    this.stationName = stationName;
    this.validityStart = validityStart;
    this.validitionInfo = validitionInfo;
    this.type = type;
}
```

FigureIV.6 Structure de données DataRecord

La classe ProductRecordsTable renvoie les nouvelles observations (liste des DataRecord) :

```
public ProductRecordsTable(DataBase database) { this.database = database; }

private DataRecord getDataRecord(ResultSet rs) throws SQLException {
    return new DataRecord(
            rs.getInt("msg_id"),
            rs.getTimestamp("rx_time").toLocalDateTime(),
            rs.getString("region"),
            rs.getString("oaci"),
            rs.getString("name"),
            rs.getTimestamp("validity_start").toLocalDateTime(),
            rs.getString("horaires"),
            rs.getString("type")
    );
}

private List<DataRecord> getDataRecords(String query) throws SQLException {
    ArrayList<DataRecord> result = new ArrayList<>();
    ResultSet rs = database.Query(query);
    while (rs.next()) {
        DataRecord dataRecord = getDataRecord(rs);
        result.add(dataRecord);
    }
    return result;
}

List<DataRecord> getProductRecords(int lastID) {

    String query ="SELECT * FROM public.e_nat3 WHERE msg_id > "+ Integer.toString(lastID) +" LIMIT 1000;";

    ArrayList<DataRecord> result =new ArrayList<>();
    try {
        result.addAll(getDataRecords(query));
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result;
}
```

FigureIV.7 la classe ProductRecordsTable

b. Vérifier Le type du produit :

Nous utilisons une table de hachage :

```
    private static final Map<String, String> ProductNamer = new HashMap<String, String>(){
        {
            put( k: "SI",  v: "Synop");
            put( k: "SM",  v: "Synop");
            put( k: "SA",  v: "Metar");
            put( k: "SP",  v: "Speci");
            put( k: "US",  v: "Altit");
        }
    };
    private String getProductType() {return ProductNamer.getOrDefault(type, v: "Other") ;}
```

FigureIV.8 Vérification le type d'observation

c. Vérifier La validation du produit :

Si le produit ne respecte pas les règles de validation énumérées ci-dessus dans le deuxième chapitre, indiquez qu'il est indésirable.

```
public class ValidationTable {
    private DataBase database;
    private String table;

    public ValidationTable(DataBase database, String table) {
        this.database = database;
        this.table = table;
    }

    public Map<String, List<LocalTime>> getContent() {
        String query ="SELECT * FROM "+table+" ;";

        Map<String,List<LocalTime>> result =new HashMap<>();
        try {

            ResultSet rs = database.Query(query);
            while (rs.next()) {
                String key =rs.getString( i: "horaires");
                LocalTime value = rs.getTime( s: "message").toLocalTime();
                if (result.containsKey(key))
                    result.get(key).add(value);
                else
                {
                    List<LocalTime> lvalue =new ArrayList<>();
                    lvalue.add(value);
                    result.put(key,lvalue);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return result;
    }
}
ValidationTable  getContent()
```

FigureIV.9 La classe ValidationTable

La méthode qui vérifie la validité :

```
@    private boolean chkValidity(Map<String, List<LocalTime>> validationMap) {
        return validationMap.get(validitionInfo).contains(validityStart.toLocalTime());
     }

     private boolean chkValidity(Map<String, List<LocalTime>> synopValidation, Map<String, List<LocalTime>> metarValidation) {
        return getProductType().equals("Synop")? chkValidity(synopValidation): chkValidity(metarValidation);
     }
```

FigureIV.10 ChekValidity Méthode

d. Catégoriser chaque produit valide dans son range :

```
    private static final List<Pair<ValueRange,String>> SynopRangs = Collections.unmodifiableList(Arrays.asList(
            new Pair<>(ValueRange.of(Long.MIN_VALUE, -11),"AV"),
            new Pair<>(ValueRange.of( -10, +4 ),"A0"),
            new Pair<>(ValueRange.of( +5, +9  ),"A1"),
            new Pair<>(ValueRange.of( +10, +14),"A2"),
            new Pair<>(ValueRange.of( +15, 19 ),"A3"),
            new Pair<>(ValueRange.of( +20, 169),"A4"),
            new Pair<>(ValueRange.of( 170,Long.MAX_VALUE),"AR")
    ));
    private static final List<Pair<ValueRange,String>> MetarRangs = Collections.unmodifiableList(Arrays.asList(
            new Pair<>(ValueRange.of(Long.MIN_VALUE, -11),"AV"),
            new Pair<>(ValueRange.of( -10, +2 ),"A0"),
            new Pair<>(ValueRange.of( +3 , +4 ),"A1"),
            new Pair<>(ValueRange.of( +5 , +6 ),"A2"),
            new Pair<>(ValueRange.of( +7 , +8 ),"A3"),
            new Pair<>(ValueRange.of( +9 , +14),"A4"),
            new Pair<>(ValueRange.of( +15, +19),"A5"),
            new Pair<>(ValueRange.of( +20, +49),"A6"),
            new Pair<>(ValueRange.of( +50,Long.MAX_VALUE),"AR")
    ));
@   private static String getSubType(long diff, List<Pair<ValueRange, String>> ranges) {
        String subType = "UNK";
        for (Pair<ValueRange,String> R: ranges) {
            if(R.getFirst().isValidValue(diff))
                subType=R.getSecond();
        }
        return subType;
    }
```

FigureIV.11 Catégoriser les produit

4.2.2 Calculer les statistiques :

ProductStatLiveUpdater est l'algorithme qui calcule les statistiques :

```
package Application;

import java.util.List;

public class ProductStatLiveUpdater {
    private  int lastId;
    private ProductGateWay gateWay;
    private ProductStatBackEnd productStatBackEnd;


    public ProductStatLiveUpdater(ProductGateWay gateWay, ProductStatBackEnd productStatBackEnd, int lastId ) {

        this.gateWay = gateWay;
        this.productStatBackEnd = productStatBackEnd;
        this.lastId=lastId;
    }

    public ProductStatLiveUpdater(ProductGateWay gateWay, ProductStatBackEnd productStatBackEnd) {
        this.gateWay = gateWay;
        this.productStatBackEnd = productStatBackEnd;
        this.lastId=-1;
    }

    public void update() {
        List<Product> products = gateWay.getProducts(lastId);
        for (Product product : products) {
                productStatBackEnd.incProductCounterBy(product.getClassKey(),
            lastId=product.getID();
        }
        productStatBackEnd.SetLastId(lastId);
    }

}
```

FigureIV.12 L'algorithme ProductStatLiveUpdater

4.2.3 Stocker les produits traités sur REDIS :

a. La Forme de la clé :

Exemple : "Synop:05032018:DAUA:A0"

SYNOP : le type d'observation.

05032018 : la date reçu d'observation.

DAUA : la station

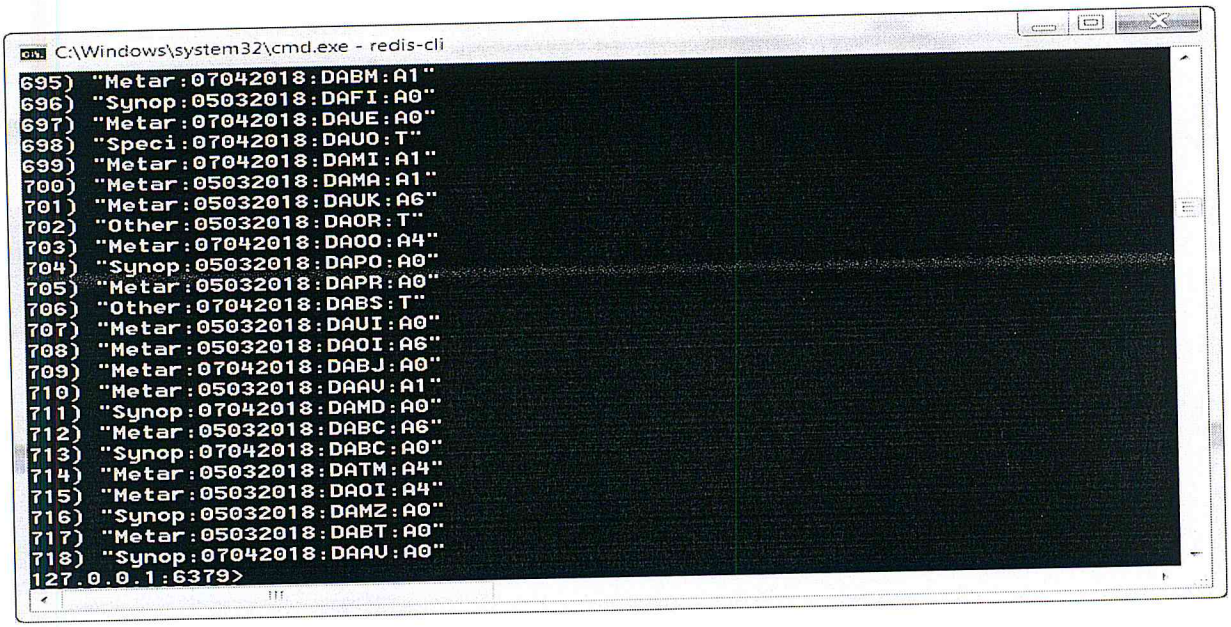
A0 : range

b. Formulation de clé :

```
private String getClassKeyFrom(Map<String, List<LocalTime>> synopValidation, Map<String, List<LocalTime>> metarValidation) {
    String type= getProductType();
    String subType= getProductSubType(synopValidation,metarValidation);
    return type+":"+ validityStart.format(DATE_TIME_FORMATTER)+":"+ oaci+":"+subType;
```

FigureIV.13 Formulation de clé

c. Présentation des clés sous Redis :



FigureIV.14 Fenêtre CMD Redis

4.3 Surveillance des services

4.3.1 Récupérer les nouveaux messages d'alerte


```
package TraceFile;

import ...

public class LogTailWrapper implements LogDataSource {
    private LogTail logTail;
    private String logName;
    private String rotatedName;

    public LogTailWrapper(String logName, String rotatedName) {

        this.logName = logName;
        this.rotatedName = rotatedName;
        logTail = new LogTail();
    }

    public List<TraceInfo> getNext() throws IOException {

        return logTail.getNext(new FileInputStream(logName),new FileInputStream(rotatedName));
    }
}
```

FigureIV.15 Classe logTailWrapper

4.3.2 Structurer les messages

a. Créer une expression régulière :

Les expressions régulières constituent un système très puissant et très rapide pour faire des recherches dans des chaînes de caractères (des phrases, par exemple). C'est une sorte de

fonctionnalité Rechercher / Remplacer très poussée, dont vous ne pourrez plus vous passer une fois que vous saurez vous en servir. Les expressions régulières vont nous permettre d'effectuer des recherches et des remplacements poussés dans des textes.

```
package TraceFile;

import ...

public class LineTokenizer {
    private String input;
    private final String pattern = "^([A-Z][a-z]{2})\\s([0-3][0-9])\\s([0-2][0-9]\\:[0-5][0-9]\\:[0-5][0-9])\\s\\[([\\w]+\\.[\\w]+)\\;\\s([0-9]+)\\]\\s(.*)";

    public LineTokenizer(String input) { this.input=input; }

    public ArrayList<String> getTokens() {
        validateInput();
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(input);

        ArrayList<String> tokens =new ArrayList<>();
        if (!m.find())
            return tokens;
        for (int i = 0; i < m.groupCount(); i++) {
            tokens.add(m.group( i+1));
        }
        return tokens;
    }

    private void validateInput() {
        if(input==null || Objects.equals(input, "")
            throw new InvalidInputException();
    }
}
```

FigureIV.16 La classe LineTokenizer

b. Créer une structure de données TraceInfo :

Exemple d'une ligne :

« Mar 05 06:55:08 [Supervision.cpp; 645] Impossible d'appeler VISION14-FTP 10.16.16.101 (No answer) »

Après la structuration :

Mar 05 06:55:08 : la date d'alerte .

Supervision.cpp : le nom du fichier source.

645 : le nombre de la linge sur le fichier .

Impossible d'appeler VISION14-FTP 10.16.16.101 (No answer) : message d'alerte.

La structure TraceInfo :

```
package Application;

import java.time.LocalDateTime;

public class TraceInfo {
    public LocalDateTime dateTime;
    public String sourceFileName;
    public int sourceFileLineNumber;
    public  String message;

    public TraceInfo(LocalDateTime dateTime, String sourceFileName, int sourceFileLineNumber, String message) {
        this.dateTime = dateTime;
        this.sourceFileName = sourceFileName;
        this.sourceFileLineNumber = sourceFileLineNumber;
        this.message = message;
    }

    public TraceInfo(TraceInfo org) {
        this.dateTime = org.dateTime;
        this.sourceFileName = org.sourceFileName;
        this.sourceFileLineNumber = org.sourceFileLineNumber;
        this.message = org.message;
    }
}
```

FigureIV.17 La classe TraceInfo

4.3.3 Filtrage des données structurées

```
package TraceFile;

import ...

public class TraceSourceFilter implements LogDataSource {
    private Predicate<? super TraceInfo> filterPredicate;
    private LogDataSource dataSource;

    public TraceSourceFilter(Predicate<? super TraceInfo> filterPredicate, LogDataSource dataSource) {
        this.filterPredicate = filterPredicate;
        this.dataSource = dataSource;
    }


    public List<TraceInfo> getNext() throws IOException {
        return dataSource.getNext().stream()
                .filter(filterPredicate)
                .map(TraceInfo::new)
                .collect(Collectors.toCollection(ArrayList::new));

    }
}
```

FigureIV.18 La classe TraceSourceFilter

4.3.4 Capturer les états des serveurs et les stocker sur Redis

```
package Application;

import ...

public class ServerStatusLiveUpdate {
    private LogDataSource logDataSource;
    private HashMapStore serverStatusBackEnd;
    private String backendKeyPrefix;
    private DateTimeFormatter dateTimeFormatter=DateTimeFormatter.ofPattern("HH:mm:dd:MM:yyyy");

    public ServerStatusLiveUpdate(LogDataSource logDataSource, HashMapStore serverStatusBackEnd, String backendKeyPrefix) {
        this.logDataSource = logDataSource;
        this.serverStatusBackEnd = serverStatusBackEnd;
        this.backendKeyPrefix = backendKeyPrefix;
    }
    public ServerStatusLiveUpdate update(){

        try {
            List<TraceInfo> logData = logDataSource.getNext();
            for (TraceInfo traceInfo: logData ) {
                String key = backendKeyPrefix +traceInfo.dateTime.format(dateTimeFormatter);
                if(traceInfo.message.matches( s: "Pilot not responding|Application Stopped"))
                    serverStatusBackEnd.setValue(key,0);
                else if(traceInfo.message.matches( s: "Application started - pilot"))
                    serverStatusBackEnd.setValue(key,200);
                else if(traceInfo.message.matches( s: "Application started - standby"))
                    serverStatusBackEnd.setValue(key,100);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

FigureIV.19 La classe ServerStatusLiveUpdate

4.3.5 Réalisation d'un ordonnanceur (Scheduler)

```
package Application;

import ...

public class Scheduler {

    List<Command> jobs = new ArrayList<>();
    private Thread executer;
    private long timeOut;

    public Scheduler(long timeOut) {

        this.timeOut = timeOut;
    }
    public void registerJob(Command cmd) { jobs.add(cmd); }
    public void start() {
        if (executer != null)
            return;

        Runnable runnable = () -> {
            boolean run=true;
            while (run){
            for (Command cmd : jobs) {
                cmd.execute();
            }
            try {
                Thread.sleep(timeOut);
            } catch (InterruptedException e) {
                run=false;
            }
        }
        };
        executer = new Thread(runnable);
        executer.start();
    }
```

FigureIV.20 La classe Scheduler part1

```
    public void stop(){
        if(executer == null)
            return;
        if (executer.isAlive())
        try {
            executer.interrupt();
            executer.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

FigureIV.21 La classe Scheduler part 2

4.3.6 La classe Main

a. La configuration

```
public class Main {
    private static final String connectionString="jdbc:postgresql://localhost:5432/odb";
    private static final String username="corm_admin";
    private static final String password="123456789";
    private static final String redisConnectionString="jdbc:redis://localhost";

    private static String path1="mfile.txt";
    private static String rpath1="mfile2.txt";
    private static String path2="mfile1.txt";
    private static String rpath2="mfile3.txt";
```

FigureIV.22 La classe Main (la partie de la configuration)

b. Le déclenchement des calculs

```
    public static void main(String[] args){

        DataBase dataBase = null;
        try {
            dataBase = new DataBase(connectionString,username,password);
            dataBase.Connect();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        DataGateWayImpl dataGateWay = getDataGateWay(dataBase);

        RedisClient redisClient = new RedisClient(redisConnectionString);
        LogInfoBuffer buffer1 = createLogInfoBuffer(path1, rpath1);
        LogInfoBuffer buffer2 = createLogInfoBuffer(path2, rpath2);
        Scheduler scheduler = new Scheduler( timeOut: 4000);
        scheduler.registerJob(createProductUpdater(dataGateWay, redisClient));

        scheduler.registerJob(buffer1::update);
        scheduler.registerJob(buffer2::update);
        scheduler.registerJob(createServerUpdater(redisClient, serverName: "Server2k",buffer1));
        scheduler.registerJob(createServerUpdater(redisClient, serverName: "Server3G",buffer2));
        scheduler.start();
        try {
            Thread.sleep( millis: 25000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }finally {
            scheduler.stop();
            dataBase.disConnect();
```

FigureIV.23 La classe Main (déclenchement des calculs)

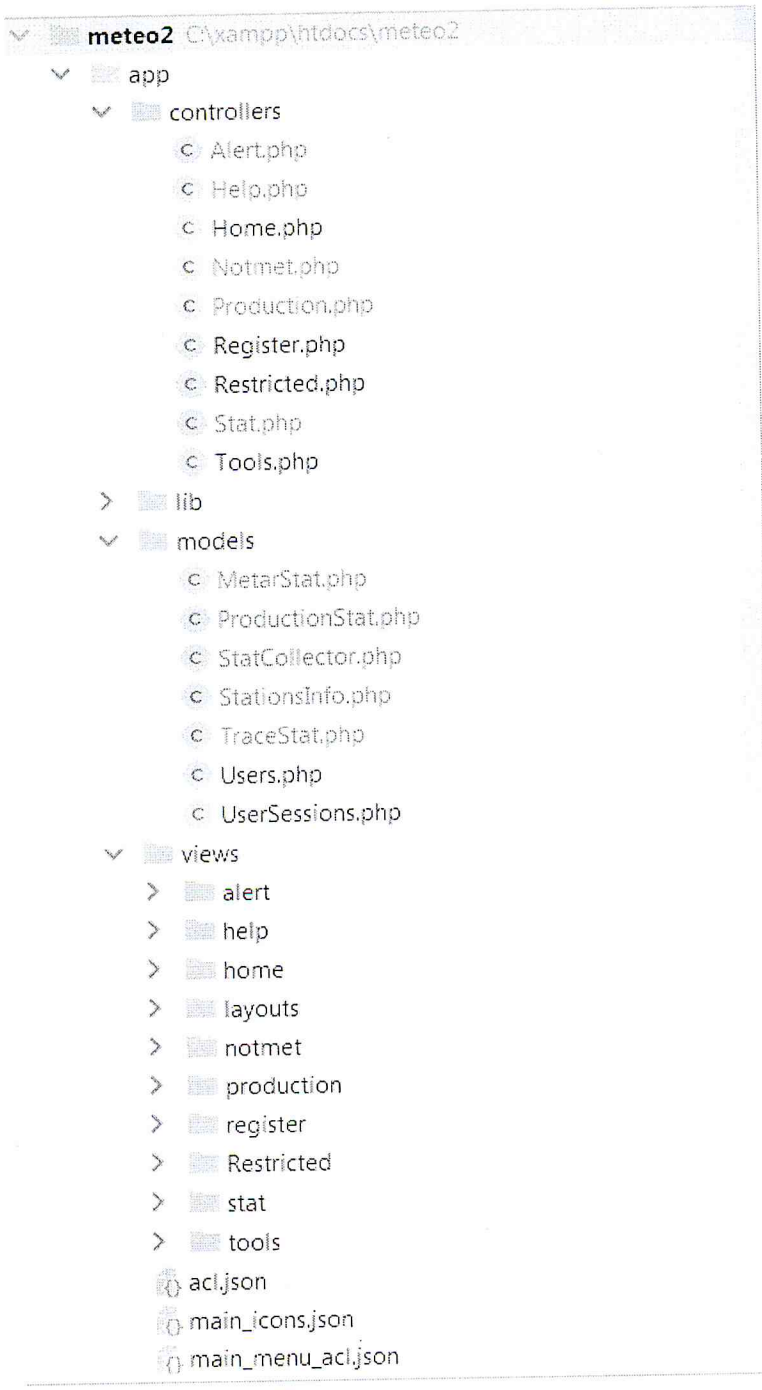
4.4 La réalisation des clients d'affichage

Les deux applications d'affichage sont réalisées avec les principes du pattern MVC.

4.4.1 La réalisation du site web

4.4.1.1 Le répertoire d'application

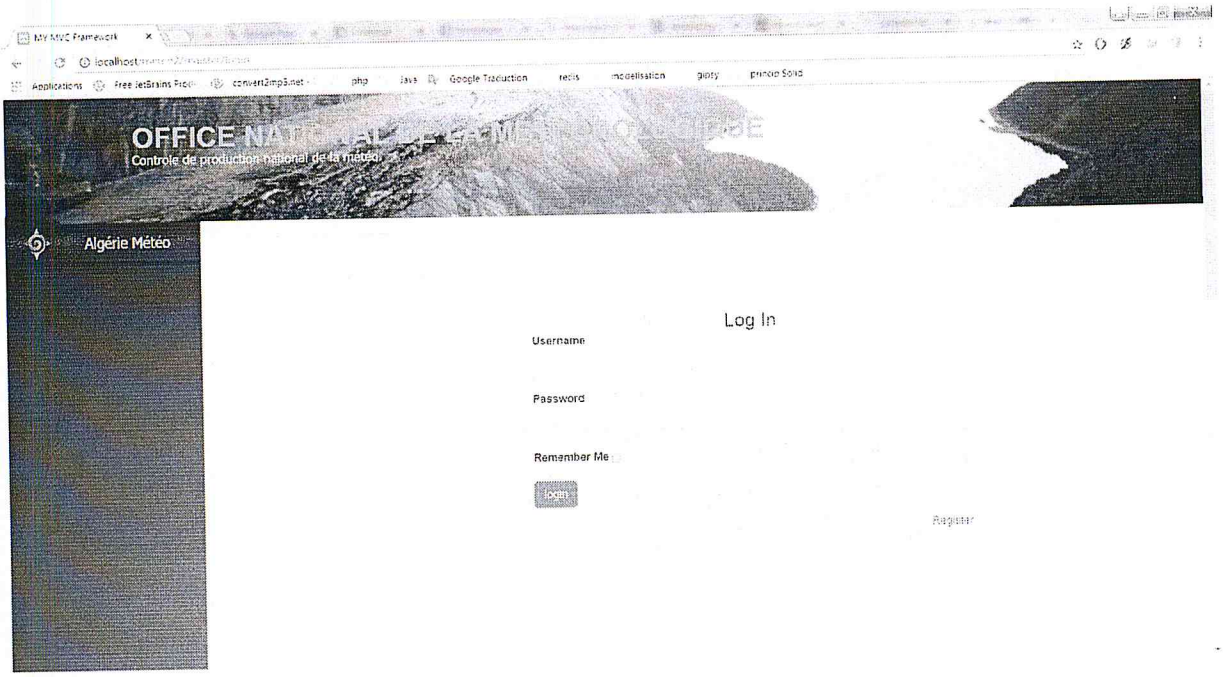
Arborescence des répertoires respectant le modèle MVC :



FigureIV.24 répertoires d'application client d'affichage WEB

4.4.1.2 Présentation de quelques interfaces de l'application web

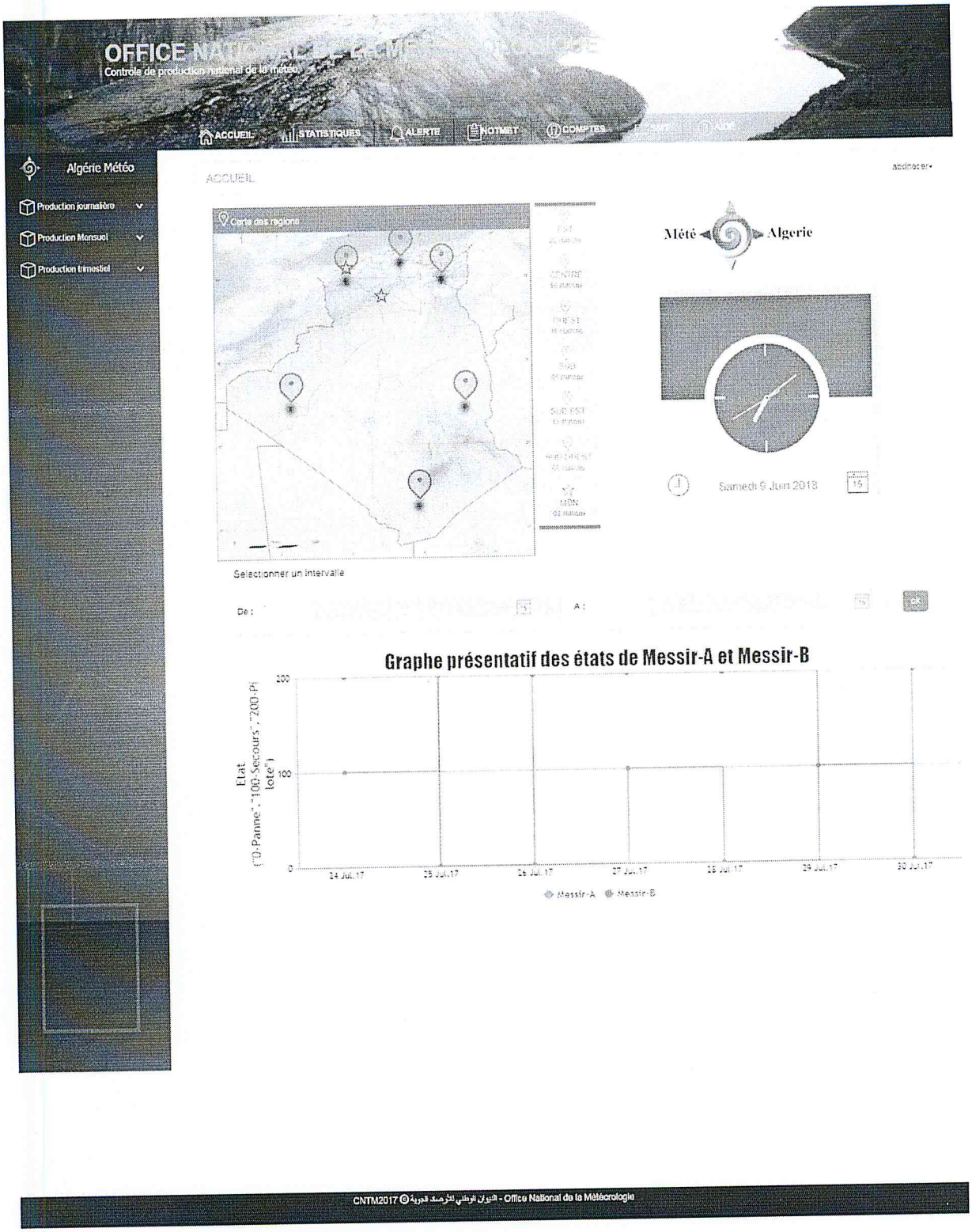
a. La page login :



FigureIV.25 La page login

b. la page d'accueil :

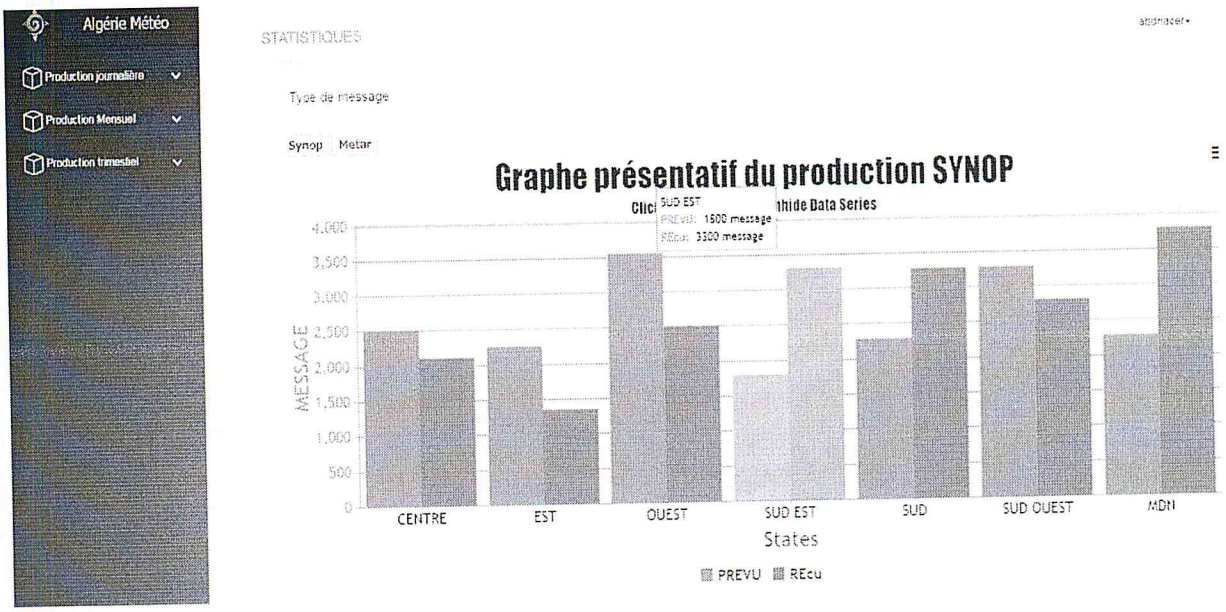
C'est la première page téléchargée et visualisée par l'internaute. Elle permet de naviguer ouvertement sur le site et découvrir les différentes fonctionnalités offertes par ce dernier tel que les problèmes des stations et les rapports journaliers et mensuels. En bas de cette page il y a une représentation graphique. Elle représente l'états des MESSIR A et B le pilote et le secours a l'état 0 le MESSIR a une panne et (100 le secours) et (200 le pilote).



FigureIV.26 La page d'accueil

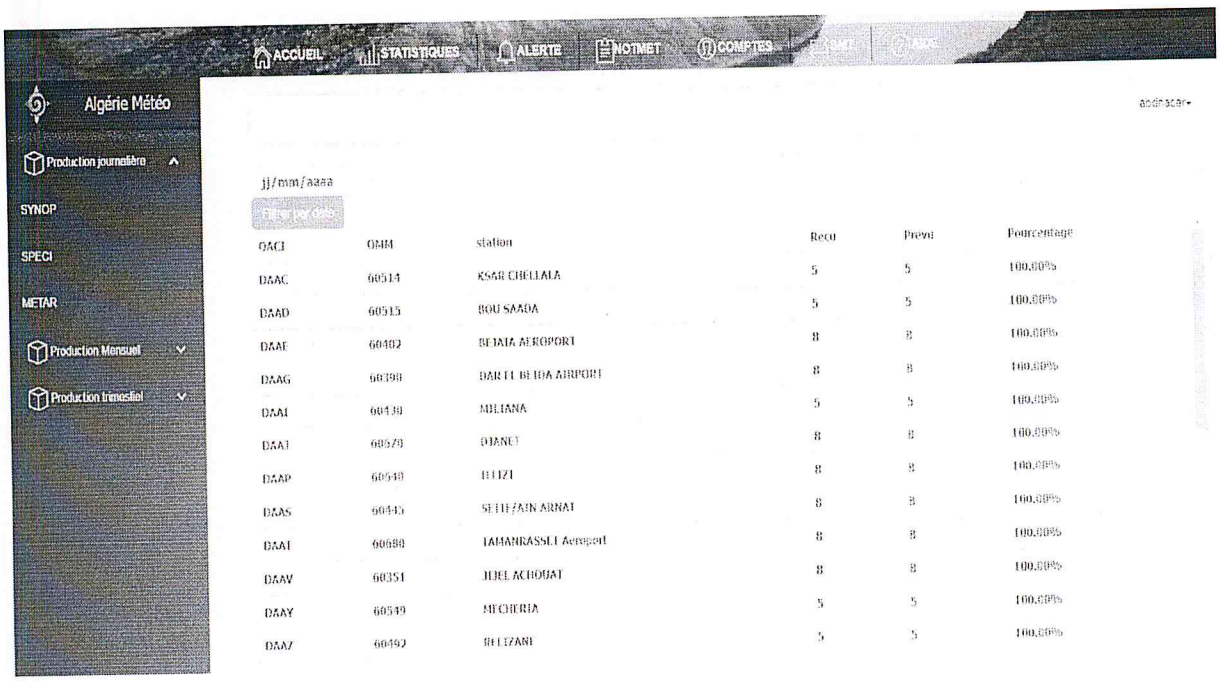
c. La page statistique :

La page statistique nous permet de voir tous les statistiques de productions de SYNOP et METAR des sept régions en une représentation graphique le bleu représente les produits reçus et le rouge les produits prévus :



FigureIV.27 La page statistiques

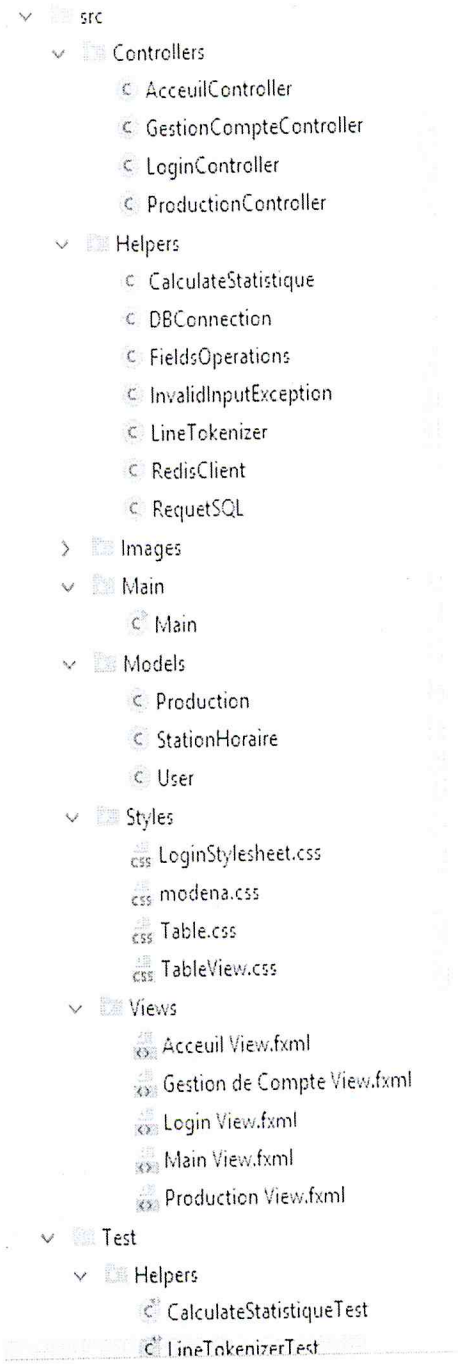
d. La page production :



FigureIV.28 production journalière Synop

4.4.2 La réalisation de l'application d'affichage Desktop :

4.4.2.1 Le répertoire d'application :



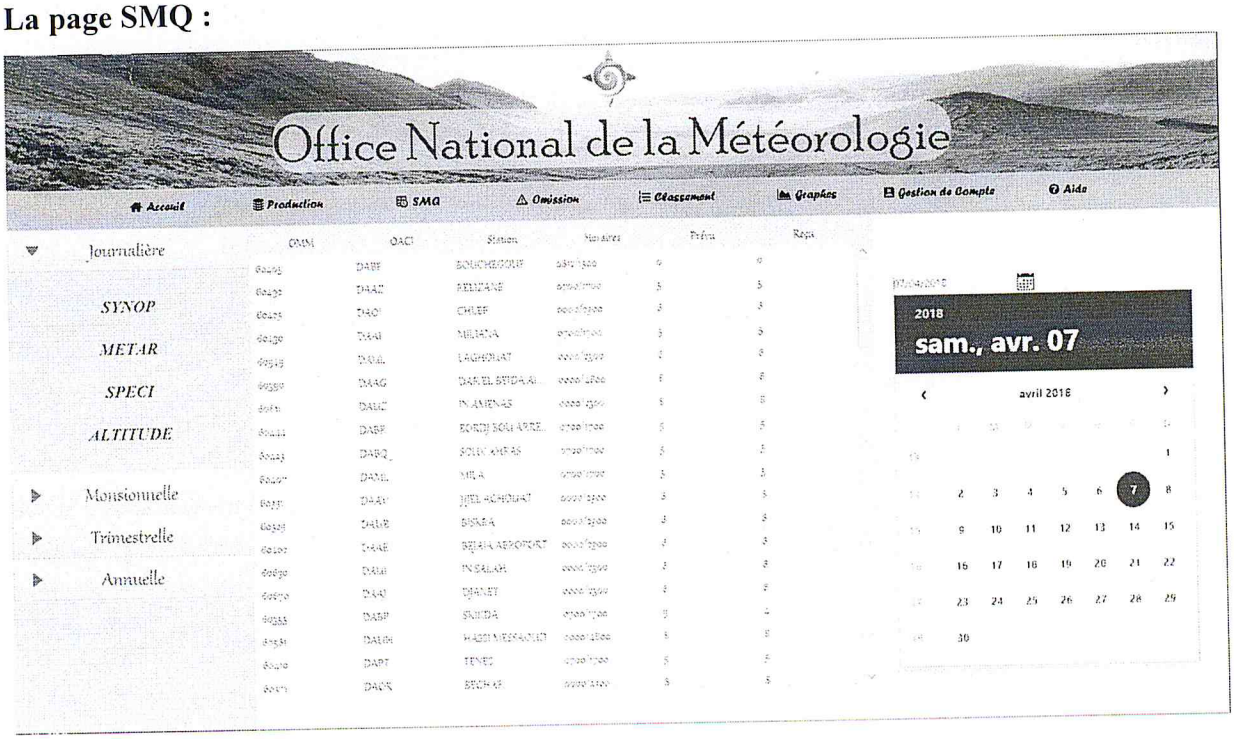
FigureIV.29 répertoire d'application client d'affichage desktop

4.4.2.2 Présentation de quelques interfaces de l'application client d'affichage desktop :

a. La page Login :

FigureIV.30 la page login d'application client d'affichage desktop

b. La page SMQ :



FigureIV.31 la page SMQ

5. Conclusion

La partie réalisation détermine une idée plus claire sur les taches qui sont réalisées dans ce projet via la présentation de l'environnement et le processus de développement. Nous avons également exposé les différentes parties de notre réalisation à savoir le service cache et les deux applications d'affichage via une présentation d'une partie du code source et quelques interfaces.

Conclusion générale

CONCLUSION GENERALE

Dans ce mémoire, nous avons présenté notre projet de fin d'étude qui s'inscrit dans le domaine de la météorologie au niveau l'Office National de la Météorologie. Ce dernier diffuse des informations scientifiques et des prévisions sur l'atmosphère terrestre, encourage l'échange rapide et libre de données météorologiques, la normalisation des observations, ainsi que la publication uniforme des observations et des statistiques. Plus précisément, nous avons travaillé au niveau du service CNTM.

Notre projet a pour objectif de concevoir et développer un système informatique permettant le suivi et le contrôle en temps réel de produits (observations) météorologiques et un calcul rapide des statistiques relatives à ces derniers.

Dans un premier temps, nous avons effectué une étude du domaine d'application (les différents types d'observation météorologique et les statistiques utilisés) suivie d'une étude de l'existant au niveau du CNTM pour établir une analyse des besoins par la suite. En effet, nous avons énuméré un certain nombre de problèmes auxquels, nous avons proposé nos solutions.

Pour la conception et le développement de notre système, nous nous sommes basés sur un certain nombre de standards et techniques notamment les principes SOLID, le formalisme UML et la méthode DTT.

Le système mis en œuvre comprend une application service cache et deux applications dédiées à l'affichage : un site web et une application desktop.

Bibliographie

Bibliographiques

[1] https://fr.wikipedia.org/wiki/Station_m%C3%A9t%C3%A9orologique

[2] MANOBS « Manuel d'observations météorologiques de surface » Septième édition, Modification 19, avril 2015 p17

[3] MANOBS « Manuel d'observations météorologiques de surface » Septième édition, Modification 19, avril 2015 p 186

[4] manuel des codes internationaux VOLUME I.1 partie A alphanumérique édition 2011 mis a jours en 2012 p21

[5] MANOBS « Manuel d'observations météorologiques de surface » Septième édition, Modification 19, avril 2015 p383

[6] manuelle d'utilisation MESSIR-COM

[7] Remi Forax, Philippe Finkel Programmation Orientée Objet -Design Pattern p27

[8] (Robert C. Martin Series) Martin, R.C.-Clean Architecture_ A Craftsman's Guide to Software Structure and Design-Pearson Education (2017) III.DESIGN PRINCIPLES

[10] POO-DP 2015-2016 Programmation Orientee Objet - Design Pattern 30 / 53

la page wikipedia du principe d'inversion de dependance

[9] Dans le livre Agile Software Development, Pinciples, Patterns and Practices, Robert C. Martin a condensé, en 2002

[11] Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System p3

[12] (Robert C. Martin Series) Martin, R.C.-Clean Architecture_ A Craftsman's Guide to Software Structure and Design-Pearson Education (2017) P25 .

[12] Modélisations UML diagrammes structurels, Génie électrique et informatique

[13] (Robert C. Martin Series) Martin, R.C.-Clean Architecture_ A Craftsman's Guide to Software Structure and Design-Pearson Education (2017) P26 .

[14] (Robert C. Martin Series) Martin, R.C.-Clean Architecture_ A Craftsman's Guide to Software Structure and Design-Pearson Education (2017) P37 .

[15] http://www.kernix.com/developpement/26-developpement-java

[16] http://php.net/manual/fr/intro-whatis.php

[17] https://fr.wikipedia.org/wiki/JavaFX

[18] http://glossaire.infowebmaster.fr/javascript/

[19] http://membres.multimania.fr/ahmedfarag/documents/ArticlesFrancais/XML.pdf

[20] https://fr.wikipedia.org/wiki/JavaFX

[21] Tiago Macedo, Fred Oliveira-Redis Cookbook -O'Reilly Media (2011)

[22] https://whatis.techtarget.com/definition/PostgreSQL

[23]http://pages-web.com/docs/phpmyadmin/Documentation.fr.html

[24]https://www.techopedia.com/definition/7755/intellij-idea

[25]https://www.techopedia.com/definition/5594/java-development-kit-jdk

[26]https://fr.wikipedia.org/wiki/PhpStorm

[27]http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

[28]https://www.supinfo.com/articles/single/493-decouverte-framework-css-boostrap

[29]https://www.techopedia.com/definition/5594/java-development-kit-jdk

[30] http://igm.univ-mlv.fr/~dr/XPOSE2009/TDD/pagesHTML/PresentationTDD.html

[31] Article: The 'Self'-Shunt Unit Testing Pattern, by Michael Feathers

[32] https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html

[33] https://getbootstrap.com/docs/4.0/getting-started/introductio