République Algérienne Démocratique et populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira-Bejaia

Faculté des Sciences Exactes

Département d'Informatique


# Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master Professionnel en Informatique

Option : Génie Logiciel

Thème

---

# Conception et réalisation d'une application web pour la gestion du système management intégré de BMT (Déclaration des accidents/incidents et non-conformités)

---

**Réalisé par :**
Yasmine IKHLEF
Hanane NASRI

**Encadré par :**
M. Khaled ALLEM Univ. Bejaia.
M. Moussa BOUMERZOUG BMT

**Evalué par :**
M. Achour ACHROUFENE Univ. Bejaia.
M. Nabil DJEBARI Univ. Bejaia.

Année universitaire 2019/2020

# Dédicaces

Je dédie ce modeste travail

À ma chère mère pour son soutien, ses encouragements et sa patience,

À ma sœur Nadia et mon beau-frère Mustapha pour leur aide, contribution et présence,

À mes frères pour leur bienveillance,

À mes amis et toutes les personnes qui ont participé de près ou de loin à la réussite de mon travail.

Yasmine Ikhlef

# Remerciements

On commence par remercier les respectables examinateurs pour nous avoir accordé de leur temps précieux pour juger et évaluer notre travail.

On tient à remercier aussi notre encadreur M. **Khaled ALLEM** pour l'attention qu'il a apporté à notre projet et ses précieux conseils.

On adresse aussi toute notre gratitude, et nos sincères remerciements à Mr **Mustapha HAMMAS** pour son engagement sans relâche et sa grande disponibilité pour nous avoir aidées à réaliser ce projet.

On adresse aussi notre profond respect à Mr **Moussa BOUMERZOUG** pour son bon encadrement toute la durée de stage dans l'entreprise BMT.

# Table des matières

# Liste des figures

# Liste des tableaux



# Acronymes

**2TUP** Two Tracks Unified Process

**API** : Application Programming Interface

**BMT** : Bejaia Mediterranean terminal

**DRH** : Direction des Ressources Humaines

**EPB :** Entreprise Portuaire de Bejaia

**HSE :** Hygiène Sécurité Environnement

**HTTP** : HyperText Transfer Protocol

**ORM** : Object Relational Mapping

**PORTEK** : Portek Systems and Equipement

**RUP** : Rational Unified Process

**UML :** Unified Modeling Languge

**UP** : Unified Process

**XML** : Extensible Markup Language

**SMI** : Système Management Intégré

# *Introduction Générale*

# Introduction générale

Le monde de l'informatique a connu une évolution foudroyante en seulement quelques années et ne cesse de progresser et de s'améliorer. Les applications et logiciels informatiques sont devenus très importants et indispensables à la vie quotidienne de tout individu. Ce sont des outils qui facilitent la communication et le traitement de données d'une manière rapide et efficace. C'est la raison pour laquelle les entreprises investissent fortement dans des applications et logiciels, elles cherchent à se procurer les meilleurs outils qui vont leur permettre un maximum de coordination dans la gestion de leurs entreprises.

L'entreprise Béjaia Mediterranean Terminal (BMT) dispose d'une direction des ressources humaines (DRH) dans laquelle on retrouve le service hygiène et sécurité environnement (HSE), qui est chargé de la gestion du Système Management Intégré (SMI) de l'entreprise. Le système de management de sécurité facilite la gestion des risques santé et sécurité au travail associé aux activités de l'entreprise. Le service HSE reçoit régulièrement les dossiers des déclarations d'incidents, accidents ou non-conformités sollicitant une étude et un traitement, et face à ces demandes de plus en plus grandissantes des employés, les responsables du service HSE rencontrent d'énormes difficultés à savoir :

- Une mauvaise organisation des dossiers qui entraine des recherches longues et fastidieuses, parfois la perte de ceux-ci ;
- Difficulté à traiter les déclarations mal informées de certains employés ;
- Difficulté à informer l'employé de l'état d'avancement de la procédure, ou de venir compléter les pièces manquantes.

Comment permettre aux responsables de ce service d'avoir une meilleure organisation des dossiers ? D'éviter les lenteurs de traitement des dossiers par certains services ? De traiter facilement les déclarations ? D'informer les employés ?

Lors de notre stage qui s'est déroulé au sein de l'entreprise BMT, il nous a été demandé de réaliser une application web pour la gestion du Système Management Intégré de BMT (déclaration des accidents/incidents et non-conformités) permettant de réduire la lourdeur des traitements et faciliter aux responsables, la gestion des tâches qui sont très nécessaires pour l'identification des risques au sein de l'entreprise. L'objectif majeur de ce travail est de découvrir les causes d'un accident et de corriger la situation afin de prévenir d'autres accidents et recommander des mesures correctives pour les contrôler à la source et aussi reconnaitre les non-conformités et les traiter. Pour ce faire, on a choisi d'utiliser le langage de programmation C#, la technologie ASP.NET MVC, et le Système de Gestion de Base de Données Microsoft SQL server.

Dans ce rapport trois principaux chapitres permettent d'englober le travail à réaliser :

- Le **premier chapitre** présente le contexte général du projet, en décrivant l'entreprise d'accueil, les problèmes rencontrés par cette dernière, la solution proposée pour pallier aux problèmes survenus, le processus de développement et la modélisation.

- Le **deuxième chapitre** concerne les spécifications des besoins, l'analyse fonctionnelle du projet, et la conception de la solution finale afin de garantir la fiabilité et l'efficacité de la phase de réalisation.

- Le **troisième chapitre** se concentre sur la description des outils techniques adaptés, y compris le langage de programmation, les technologies utilisées, et l'environnement de développement. Ainsi que la description de l'application en se basant sur la présentation de ses interfaces graphiques.

Enfin, comme pour tout travail, on termine le mémoire par une conclusion générale afin de synthétiser les objectifs atteints, les acquis sur les plans théorique et surtout pratique, ce qui reste à faire et quelques perspectives en vue.

# *Chapitre 1 :*
# *Contexte général du projet*

*Chapitre* I

# *Contexte général du projet*

## 1 Introduction

L'étude de l'existant est une étape initiale de la réalisation d'une application ou du logiciel informatique, elle est aussi une étape importante et décisive. C'est grâce à cette étude que nous arriverons à voir de plus près les spécificités de l'organisme et recenser ses besoins, ses problèmes et anomalies.

Notre étude se porte sur le service Hygiène et Sécurité de la Direction des Ressources Humaines de l'entreprise portuaire BMT qui est un service qui participe à la définition de la politique de qualité et assure la sécurité de l'entreprise (personnel, matériel, conditions de travail, respect de l'environnement) et prévient les risques professionnels.

Ce chapitre sera donc réservé pour la présentation de l'organisme d'accueil, l'étude de l'existant, proposition des solutions aux problèmes soulevés et la méthodologie de conception.

## 2 Présentation de BMT

### 2.1 Création

Dans son plan de développement 2004-2006, l'entreprise portuaire de Béjaia avait inscrit à l'ordre du jour le besoin d'établir un partenariat pour la conception, le financement, l'exploitation d'un terminal à conteneurs au port de Béjaia.

Dès lors, L'EPB (Entreprise Portuaire de Béjaia) s'est lancée dans la tâche d'identifier les partenaires potentiels et a arrêté son choix sur le groupe PORTEK qui est une société Singapourienne spécialisée dans le domaine de la gestion des terminaux à conteneurs. Elle est présente dans plusieurs ports dans le monde. Le projet a été présenté au conseil de la participation de l'état (CPE) en février 2004. Le CPE a donné son accord au projet en mai 2004. Donc Béjaia Mediterranean Terminal (BMT) a vu le jour avec la jointe venture de l'EPB à 51% et PORTEK à 49%.

BMT est créée comme une société par action. C'est une entreprise prestataire de service, spécialisée dans le fonctionnement, l'exploitation, et la gestion du terminal à conteneur. Pour atteindre son objectif, elle s'est dotée d'un personnel compétent particulièrement formé dans les opérations de gestion du terminal. Elle dispose d'équipement d'exploitation les plus perfectionnés de qualité, d'efficacité et de fiabilité en des temps record et à des coûts compétitifs. BMT offre ses prestations sur la base de 24h/7j [W1].


**Figure 1** Jointe venture de l'EPB et PORTEK

## 2.2 La structure de l'entreprise

### 2.2.1 Direction Générale (DG)

C'est la pièce maitresse de l'entreprise. A sa tête, le directeur générale qui la gère, il a le pouvoir de décider, administrer, assigner des directives pour les différentes structures et faire le lien entre les directions de l'entreprise.

### 2.2.2 Direction des Ressources Humaines (DRH)

- **Service personnel :** Mettre en œuvre des systèmes de gestion intégrés de l'entreprise et qui traduisent une adéquation entre les impératifs économiques et les attentes du personnel.
- **Service des moyens généraux :** Chargé des achats et de la gestion des stocks de l'entreprise.
- **Service HSE hygiène et sécurité environnement :** Participe à la définition de la politique de qualité et assure la sécurité de l'entreprise (personnel, matériel, conditions de travail, respect de l'environnement) et prévient les risques professionnels.

### 2.2.3 Direction des Finances et Comptabilité

La mission de la Direction des Finances et Comptabilité est de veiller à la gestion financière et la comptabilité de l'entreprise.

### 2.2.4 Direction Marketing

La Direction Marketing est restructurée récemment après la jonction des trois départements (Commercial + Marketing + Informatique).

### 2.2.5 Direction des Opérations

La Direction des Opérations assure la planification, les opérations de manutentions, le suivi des opérations d'acconage, et la logistique de l'entreprise.

## 2.3 Direction Technique

Assure la maintenance, l'entretient, l'installation, test, mise à jour, préparation et dépannage des différentes machines et la formation technique pour le personnel interne et externe.

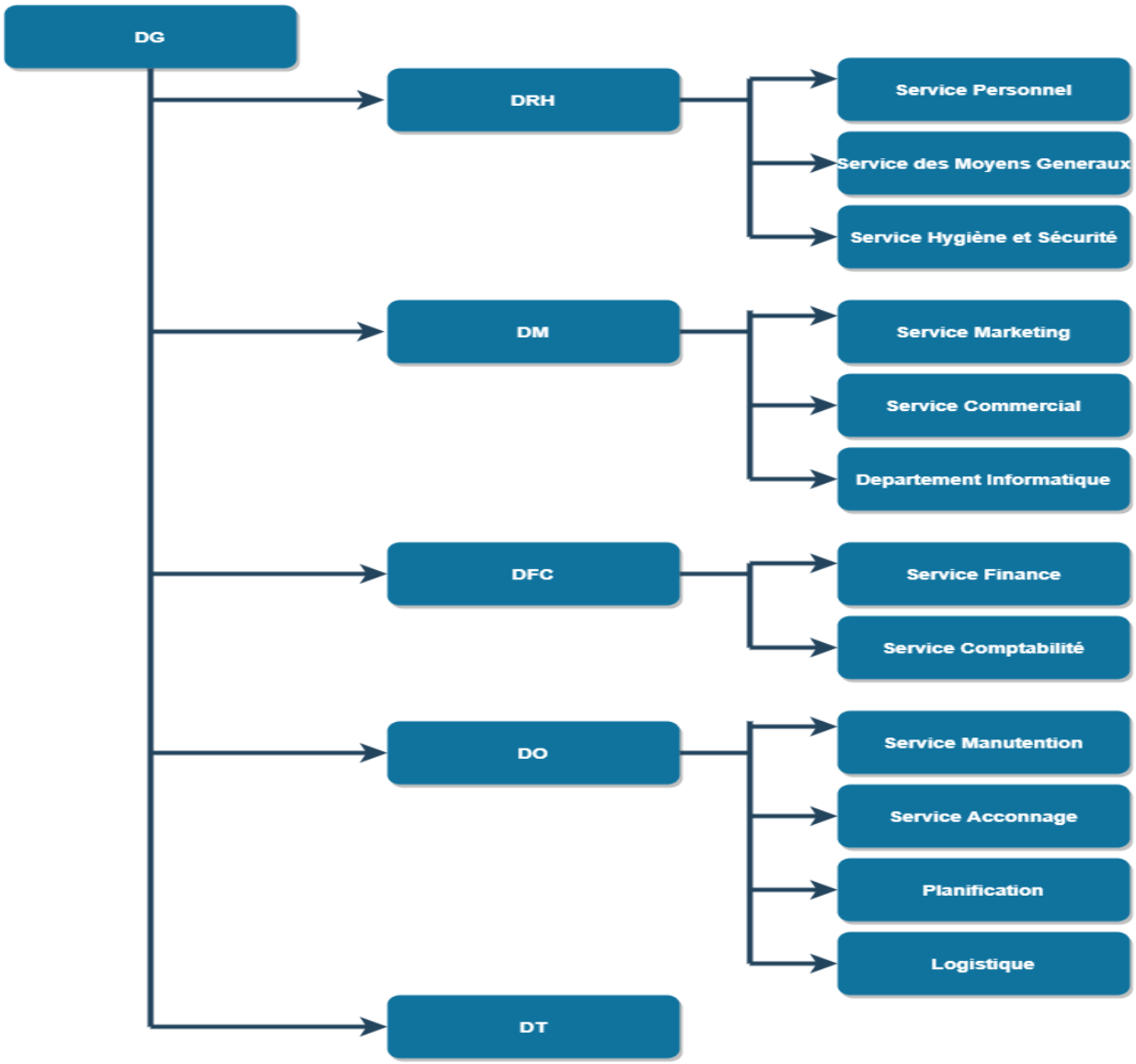


**Figure 2** Structure de l'entreprise BMT

# 3 Présentation du projet

## 3.1 Description de l'existant

Chez BMT, et à forte probabilité chez beaucoup d'autres entreprises avec un nombre considérable d'équipements et d'employés, la déclaration des accidents, incidents du travail et la détection des non-conformités, leurs enquêtes et analyses sont des tâches très nécessaires pour l'identification des risques. L'objectif majeur de ce travail est de découvrir les causes d'un accident et de corriger la situation afin de prévenir d'autres accidents et recommander des mesures correctives pour les contrôler à la source et aussi reconnaitre les non-conformités et les traiter.

Actuellement la déclaration des accidents, incidents et non-conformités se fait d'une manière très basique voir manuelle, la paperasse passe d'une direction à une autre, d'un service à un autre pour enfin arriver au service Hygiène et Sécurité Environnement de la Direction des Ressources Humaines qui est concerné par le traitement.

Définitions :

- **Incident :** Une suite d'évènements non contrôlés pouvant causer des blessures à des personnes, dégâts aux biens, l'environnement ou l'image de marque de l'entreprise.
- **Accident :** Un incident imprévu qui a causé des blessures aux personnes, des dégâts aux biens ou pertes de production y compris des incendies.
- **Non-conformité :** Non satisfaction d'une exigence.
- **Action corrective :** Action visant à éliminer la cause d'une non-conformité ou d'une autre situation indésirable détectée.

## 3.2 Critique de l'existant

- **Tache pénible**
  La procédure que BMT suit pour la gestion des déclarations est très pénible. En fait, ce sont des tâches fastidieuses, et ennuyeuses avec beaucoup de paperasse. Encore un autre inconvénient, certains employés ne déclarent pas par manque de compétences linguistiques.

- **Temps important consacré et efficacité**
  La gestion manuelle des déclarations à l'aide des fichiers Excel et Word prend beaucoup de temps, le processus devient très long et diminue l'efficacité de cette tâche.

- **Manque de souplesse durant la gestion**
  Les employés doivent se déplacer entre les départements pour effectuer la déclaration mais aussi pour se renseigner sur l'état d'avancement de la procédure, ce qui empêche la souplesse au niveau de la gestion.

- **Pertes de données**
  Absence de traçabilité et de bases de données ce qui engendre la perte des déclarations.

## 3.3 Solution proposée

Pour remédier aux problèmes cités auparavant, nous proposons la mise en place d'une application web qui automatise la gestion du système management intégré de la BMT pour la déclaration des accidents, incidents et non-conformités. Cette application permettra de :

- Limiter les tâches de la création et l'affectation des comptes et des rôles à une seule personne (Administrateur).
- Structurer la procédure des déclarations.
- Aider les employées à tout déclarer.
- Gérer les différentes déclarations, les consulter et suivre leurs états.
- Gérer les données en utilisant une base de données qui assure leur intégrité.
- Traiter les déclarations d'une manière fluide et structurée et notifier le déclarant à chaque étape du traitement.
- Un tableau de bord pour visualiser toutes les statistiques.
- Afficher des notifications une fois la déclaration traitée par l'agent HSE et une fois envoyée aux assurances par l'agent juridique.

# 4 Processus de développement

Le développement d'un logiciel informatique nécessite le suivi d'un processus de développement qui définit l'ensemble des étapes à suivre au cours du développement d'une application informatique et puisqu'il n'existe pas un seul processus de développement, ni de processus standard ça reste donc une question de choix. De notre côté, nous avons choisi le processus unifié (UP : Unified Process en anglais).

## 4.1 Processus Unifié

### 4.1.1 Définition

Le processus unifié est un processus de développement logiciel orienté objet, centré sur l'architecture, guidé par des cas d'utilisation et orienté vers la diminution des risques.
C'est une méthode générique, itérative et incrémentale. Le processus unifié permet d'affecter des tâches et des responsabilités au sein d'une organisation de développement logiciel. C'est aussi une approche à adapter pour des petits projets [W2].

### 4.1.2 Caractéristiques du processus unifié

Parmi les caractéristiques essentielles du processus unifié on peut citer :

- ❖ Le processus unifié est guidé par les « patrons » de conception (Design Patterns),
- ❖ Le processus unifié est basé sur le langage UML (ensemble d'outils et de diagrammes),
- ❖ Le processus unifié est piloté par les cas d'utilisation,
- ❖ Centré sur l'architecture,
- ❖ Itératif et incrémental.

La figure suivante représente les caractéristiques essentielles du processus unifié :

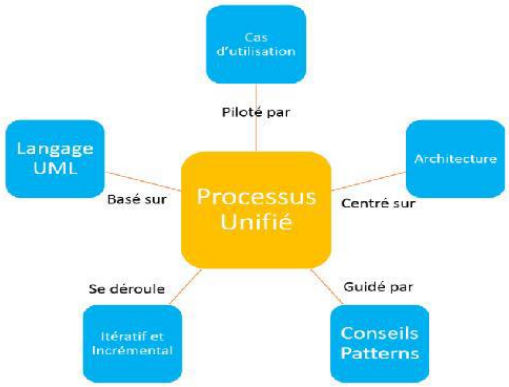


**Figure 3** Caractéristiques du processus unifié

## 4.2 Adaptation du processus unifié RUP

Il existe plusieurs processus de développement qui implémente l'UP dont le plus intéressant le RUP (Rational Unified Process).

### 4.2.1 Définition

Le concept de Rational Unified Process (RUP) provenait de la Rational Software Corporation, une division d'IBM (International Business Machines Corporation).

RUP est l'une des plus célèbres implémentations de la méthode UP, elle fournit les meilleures lignes directrices pratiquées, des modèles et des illustrations de tous les aspects du programme de développement.

### 4.2.2 Les phases du cycle de vie de la méthodologie RUP

La méthodologie de développement RUP répète un certain nombre de fois une série de cycles.

Tout cycle se conclut par la livraison d'une version du produit aux clients et est divisée en quatre phases : création (pré-étude), élaboration, construction et transition, chacune d'entre elles se subdivisent à son tour en itérations. Chaque cycle se traduit par une nouvelle version du

système. Ce produit se compose d'un corps de code source reparti sur plusieurs composants pouvant être compilés et exécutés et s'accompagne de manuels et de produits associés.
La figure suivante représente le cycle de vie d'un processus unifié rationnel [W3].

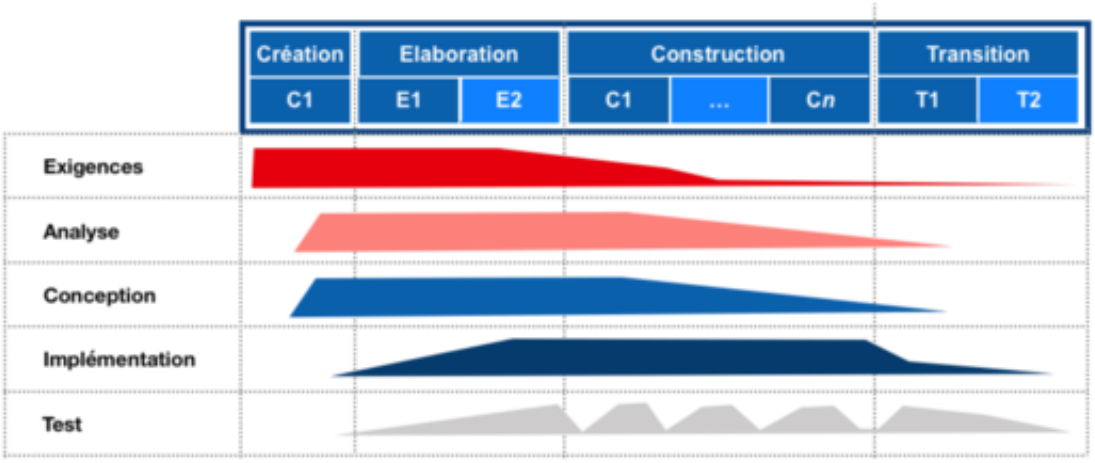


**Figure 4** Le cycle de vie d'un processus unifié rationnel

Les phases de la méthodologie RUP sont :

- Création : Le résultat de cette phase est la détermination des objectifs à atteindre dans notre future application en partant de l'existant.
- Elaboration : Il s'agit de détailler les spécifications des fonctions ainsi que la structure des données, des contrôles et les interfaces.
- Construction : Il s'agit de réaliser l'implémentation des programmes et effectuer les tests unitaires.
- Transition : Il s'agit de tester mon application web et corriger les erreurs découvertes [W4].

### 4.2.3 Justification du choix du processus unifié

Nous avons adopté le processus de développement RUP parce qu'il est beaucoup plus complet que les autres processus, ses itérations sont courtes et varient entre trois et six semaines. En outre, il permet de limiter les coûts, en termes de risques, aux strictes dépenses liées à une itération. Ce processus se focalise sur la qualité du produit vue que les tests sont invoqués d'une façon itérative.

# 5 Modélisation par UML

Le grand intérêt de la modélisation est de définir un ensemble de techniques de conception afin de réduire la complexité du système étudié et d'organiser l'évolution du projet tout en représentant les étapes de la réalisation de ce dernier [1].

Dans ce projet on a adopté une approche orienté objet basée sur un outil de modélisation qui est UML (Unified Modeling Language). En fait, UML est un standard pour l'analyse et la modélisation des applications informatiques construites à l'aide d'objets (logiciels, applications web, applications desktop…), c'est un consortium d'entreprises qui a été fondé pour construire des techniques pour faciliter l'interopérabilité, et plus spécifiquement, l'interopérabilité des systèmes orientés objet. UML est issu de l'unification de nombreux langages de modélisation graphique orientée objet. Il unifie à la fois les notations et les concepts orientés objets [2].

## 5.1 Caractéristiques de l'UML

- ✓ UML est un support d'analyse performant.
- ✓ Facilite la compréhension des représentations abstraites complexes.
- ✓ Il englobe les modèles des classes, des états, et d'interaction.
- ✓ Ses notations de modélisation limitent les ambiguïtés et les incompréhensions [3].

# 6 Conclusion

Dans ce chapitre, nous avons présenté le cadre général du travail tout en décrivant l'organisme d'accueil, le contexte et les objectifs du projet suivi d'une étude de l'existant dans laquelle nous avons dégagé les défaillances du système actuel et enfin, nous avons abordé notre solution et la méthodologie de conception. Cette étude nous permet de commencer plus clairement la phase d'analyse fonctionnelle et conception de l'application.

# *Chapitre 2 :*

# *Analyse des besoins et conception de l'application*

*Chapitre* 2

# *Analyse des besoins et conception de l'application*

## 1 Introduction

Au cours du chapitre précédent nous avons choisi le processus unifié UP comme processus du développement et l'UML comme langage de modélisation nous allons à présent consacrer ce chapitre pour la spécification fonctionnelle en identifiant les différents acteurs du système, recenser les besoins fonctionnels et non fonctionnels, l'analyse et la conception de l'application avec quelques diagrammes de séquence et finir avec un diagramme de classes.

## 2 Spécifications fonctionnelles

### 2.1 Spécification des besoins

La capture des besoins est une phase fondamentale dans la réalisation du projet. C'est l'étape qui nous permettrait de définir toutes les interactions entre le système et son environnement. Ainsi nous identifierons qui va interagir avec le système et quelles fonctionnalités le système va fournir aux acteurs.

#### 2.1.1 Identification des acteurs

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages éventuellement porteurs de données.

Les acteurs qui participent à notre système sont :

- Administrateur : le responsable de l'administration de l'application.
- User (déclarant).
- Agent HSE.
- Agent juridique.
- Lecteur de rapport.

### 2.1.2 Capture des besoins fonctionnels

Cette section concerne l'étude du domaine de l'application. Il s'agit de déterminer les frontières, le rôle, les ressources disponibles et requises, les contraintes d'utilisation et de performance, etc. Cette étape a pour but d'éviter de développer un logiciel non adéquat.
La spécification fonctionnelle se traduit par un diagramme de cas d'utilisation qui donne une vue extérieure et globale du système et définit les liens entre les utilisateurs et toutes les fonctionnalités que propose ce système. Mais avant de passer au diagramme de cas d'utilisation, il faut tout d'abord identifier ces cas d'utilisation.

#### 2.1.2.1 Identification des cas d'utilisation

Un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système. Il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera [W5]. En se basant sur les renseignements recueillis auprès des employés du service HSE (hygiène et sécurité environnement) de la Direction des Ressources Humaines (DRH), nous avons établi une liste des tâches effectuées par ces derniers. À partir de cette liste (qui est citée précédemment), nous allons déduire les différentes fonctionnalités du futur système. De ce fait nous avons distingué les cas d'utilisation suivants pour chaque acteur :

**Acteur 1 : déclarant** (appelé aussi **agent de saisie)**

- ✓ Gérer les déclarations des employés (à savoir les déclarations d'accidents, incidents et non-conformités, l'ajout et la modification de ces dernières).

**Acteur 2 : agent HSE**

- ✓ Traiter tous types de déclarations ;
- ✓ Etablir tous types de décisions et réponses à savoir la validation des déclarations, les actions correctives, les actions recommandées...etc.

**Acteur 3 : agent juridique**

- ✓ Consultation des déclarations validées ;
- ✓ L'envoi des déclarations traitées et validées aux assurances.

**Acteur 4 : lecteur de rapport**

- ✓ Consultation des tableaux de bord pour observer les statistiques.

L'exécution de tous les cas d'utilisation inclut le cas d'utilisation « s'authentifier », par exemple si l'agent HSE veut établir une décision il doit tout d'abord s'authentifier.

## 2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de formaliser les besoins et de modéliser les services offerts par le système. C'est donc une vue du système dans son environnement extérieur. Il modélise à la fois des activités (fonctionnalités) et des communications (interactions) pour les entités concernées (acteurs) [W5].

Le diagramme exposé dans la figure suivante décrit l'ensemble des cas d'utilisation de notre projet :

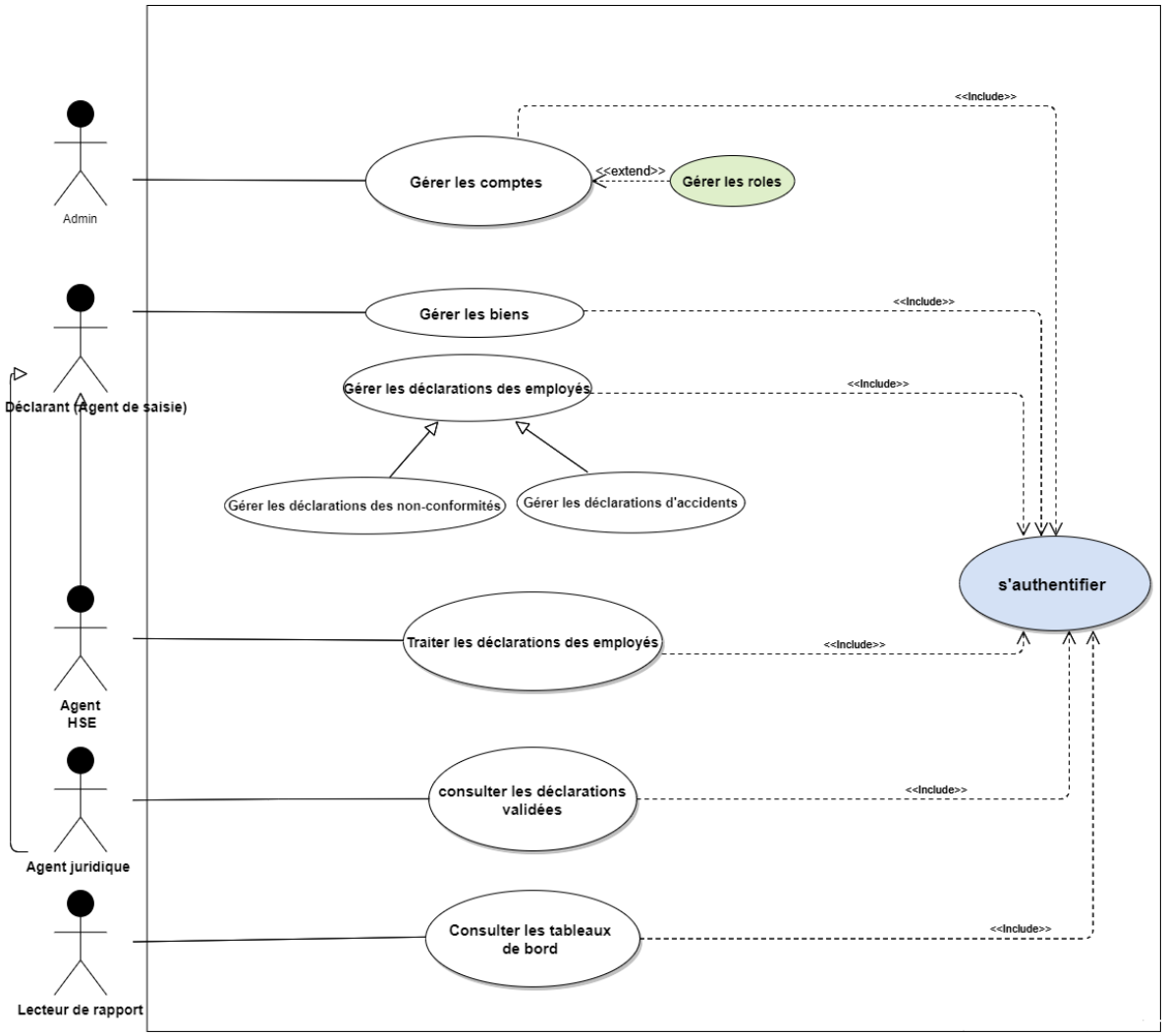


**Figure 5** Diagramme de cas d'utilisation global

### 2.2.1 Description textuelle des principaux cas d'utilisation

✓ **Cas d'utilisation « Gérer les comptes »**

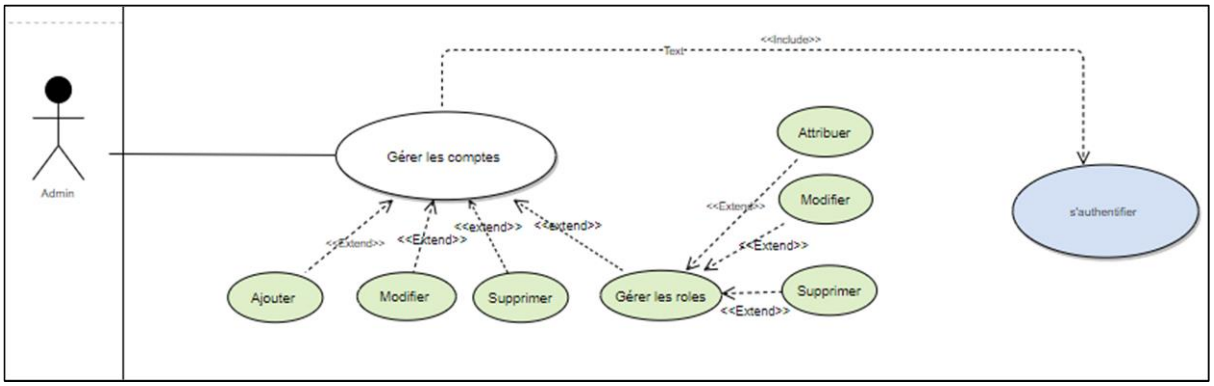


**Figure 6** Cas d'utilisation « Gérer les comptes »

| | |
|---|---|
| Cas d'utilisation | Gérer les comptes |
| Acteur | Administrateur |
| Objectif | La gestion des comptes utilisateurs et leurs rôles.<br>Régler les problèmes d'authentification. |
| Condition | S'authentifier en tant qu'administrateur. |
| Scénario nominal | Aprés l'authentification de l'administrateur, ce dernier s'occupe de l'ajout, modification, suppression des comptes et la gestion des rôles. |
| Exception | L'administrateur ne pourra pas créer un compte pour un utilisateur qui n'est pas un employé de BMT. |

**Tableau 1** Description textuelle du cas d'utilisation "gérer les comptes"

✓ **Cas d'utilisation « Gérer les déclarations »**

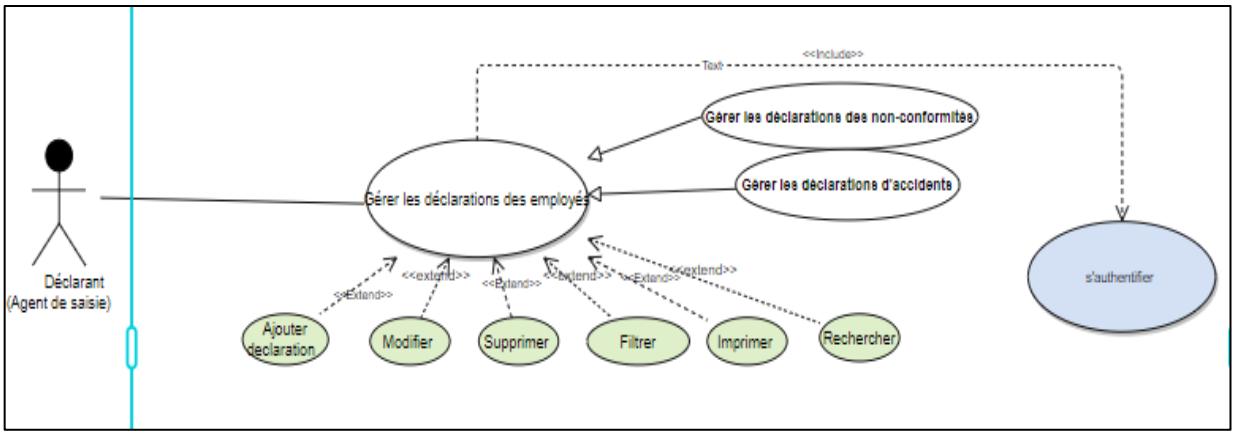


**Figure 7** Cas d'utilisation « Gérer les déclarations »

| Cas d'utilisation | Gérer les déclarations |
|---|---|
| Acteur | Déclarant |
| Objectif | Accéder à son espace et pouvoir déclarer un accident/incident et non-conformité, tout en ayant la possibilité de modifier, supprimer, filtrer, imprimer et rechercher. |
| Condition | S'authentifier. |
| Scénario nominal | 1. Le déclarant dans l'espace des declarations choisit d'ajouter une declaration d'accident par exemple.<br>2. Le système affiche un formulaire de saisie.<br>3. Le declarant remplie le formulaire.<br>4. Le système vérifie les informations.<br>5. Le système ajoute la déclaration à la base de données.<br>6. Le système affiche la liste des declarations. |
| Exception | Pas d'exceptions. |

**Tableau 2** Description textuelle du cas d'utilisation « Gérer les déclarations »

- ✓ **Cas d'utilisation « Traiter les déclarations des employés »**

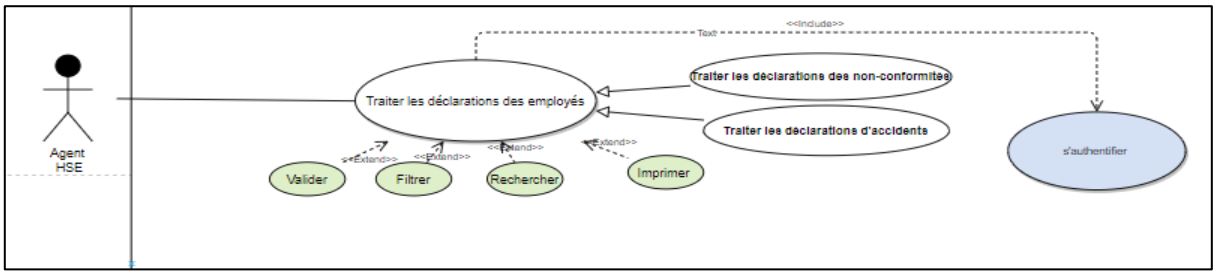


**Figure 8** Cas d'utilisation « Traiter les déclarations des employés »

| Cas d'utilisation | Traiter les déclarations des employés |
|---|---|
| Acteur | Agent HSE |
| Objectif | Traiter les déclarations d'accidents, si ces dernières sont validées l'agent HSE les envoie à l'agent juridique pour que ce dernier les envoie à son tour aux assurances, dans le cas des non-conformités, établir des actions correctives. |
| Condition | S'authentifier. |
| Scénario nominal | En accédant à son compte, l'agent HSE accède à au tableau des déclarations.<br>1. L'agent HSE peut effectuer une recherche d'une declaration precise.<br>2. L'agent HSE vérifie les informations de la declaration et l'étudie.<br>3. L'agent HSE choisit de la valider ou la refuser.<br>4. Si validée, le système notifie le declarant et l'agent juridique.<br>5. Le système ajoute la declaration traitée au tableau des declarations traitées.<br>6. L'instance de cas d'utilisation se termine. |

**Tableau 3** Description textuelle du cas d'utilisation "Traiter les déclarations des employés"

✓ **Cas d'utilisation « Consulter les déclarations validées »**

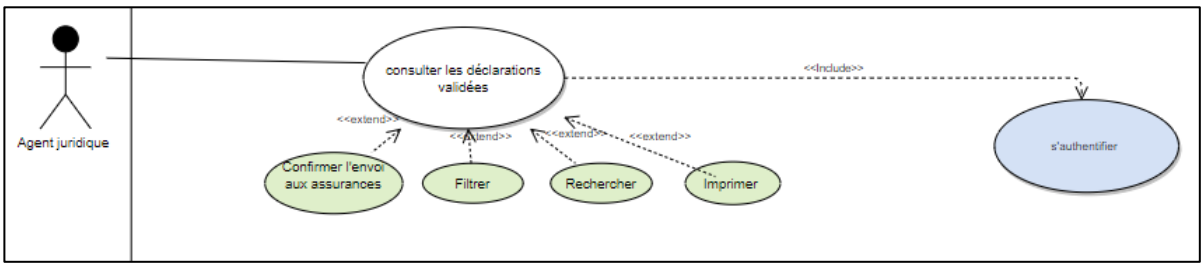


**Figure 9** Cas d'utilisation "Consulter les déclarations"

| | |
|---|---|
| Cas d'utilisation | Consulter les déclarations validées |
| Acteur | Agent juridique |
| Objectif | Recevoir les déclaration validées pour les envoyer au service des assurances. |
| Condition | S'authentifier. |
| Scénario nominal | 1. L'agent juridique appuie sur la declaration qu'il veut consulter.<br>2. Une page correspondante aux details et informations de la declaration sélectionnée s'affiche.<br>3.L'agent juridique l'imprime et l'envoie aux assurances.<br>4. Le système mit à jour le tableau des declarations traitées.<br>5. L'instance de cas d'utilisation se termine. |
| Exception | Pas d'exception. |

**Tableau 4 D**escription textuelle d'utilisation "Consulter les déclarations validées"

✓ **Cas d'utilisation « Consulter les tableaux de bord »**

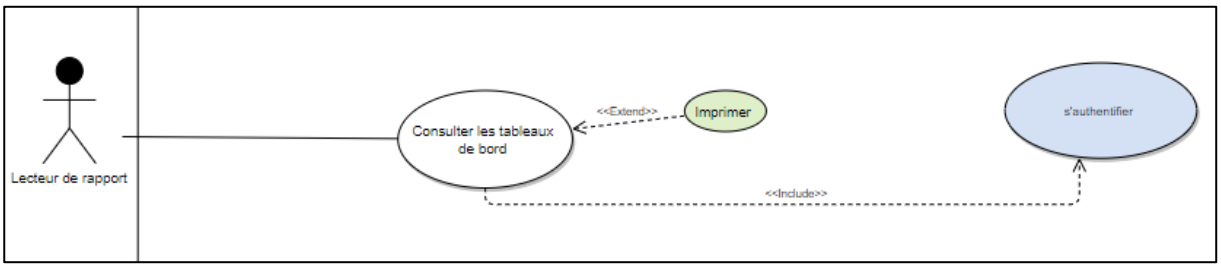


**Figure 10** Cas d'utilisation "Consulter les tableaux de bord"

| | |
|---|---|
| Cas d'utilisation | Consulter les tableaux de bord |
| Acteur | Lecteur des rapport. |
| Objectif | Avoir une vue sur les tableaux de bord pour pouvoir analyser les statistiques des différentes activités. |
| Condition | S'authentifier. |
| Scénario nominal | Un lecteur de rapport consulte Le tableau de bord pour avoir un aperçu sur les différentes progressions et statistiques, comme le nombre d'accidents et de non-conformités par mois ou par années, et les accidents par type. |

**Tableau 5** Description textuelle du cas d'utilisation "Consulter les tableaux de bord"

## 2.3 Recueil des besoins non fonctionnels

- ✓ **Sécurité :**

La gestion des comptes des utilisateurs est effectuée par l'administrateur.
Ensuite, chaque utilisateur peut s'authentifier en utilisant une adresse mail et un mot de passe pour pouvoir accéder à ses données et aux différentes fonctionnalités.

- ✓ **Ergonomie :**

**L'organisation visuelle et la cohérence** :

Pour avoir une interface conviviale et simple à utiliser par les différents utilisateurs de l'application, cette dernière aura comme caractéristiques :

- L'application est développée dans l'approche « responsive design », donc elle est adaptée aux différentes tailles d'écrans, ce qui permet aux utilisateurs de profiter de toutes ses fonctionnalités quel que soit l'appareil utilisé.
- Pour faciliter la navigation dans les différentes rubriques proposées par l'application on a adapté un menu clair et simple en utilisant des icônes identificatrices.

- ✓ **Portabilité :**

L'application doit fonctionner sur n'importe quel type de machine et de système d' exploitation et accessible à travers le plus grand nombre des navigateurs connus et largement utilisés (Google Chrome, Firefox, Microsoft Edge, Safari ...)

- ✓ **Gestion des erreurs :**

Pour chaque formulaire dans l'application, un ensemble de contraintes a été défini pour assurer la bonne saisie de tous les champs, et donc respecter la validation côté client, ainsi que la validation côté-serveur.

# 3 Analyse

La phase analyse est une étape qui consiste à analyser et étudier les besoins du client et de l'utilisateur, c'est une sorte de reformulation des besoins du client sous une forme exploitable en conception. Elle permet de déterminer de « quoi » et de « quoi faire » d'une application.

## 3.1 Diagrammes de séquence système

Le diagramme de séquence est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.

### 3.1.1 Authentification

Le diagramme suivant a été référencé à partir de plusieurs autres diagrammes. Ce cas d'utilisation « s'authentifier » apparaît dans le cas où le système donne l'accès à l'utilisateur, autrement dit, il apparaît dans le cas où l'identifiant et le mot de passe de l'utilisateur sont validés et celui-ci se connecte correctement à l'application.

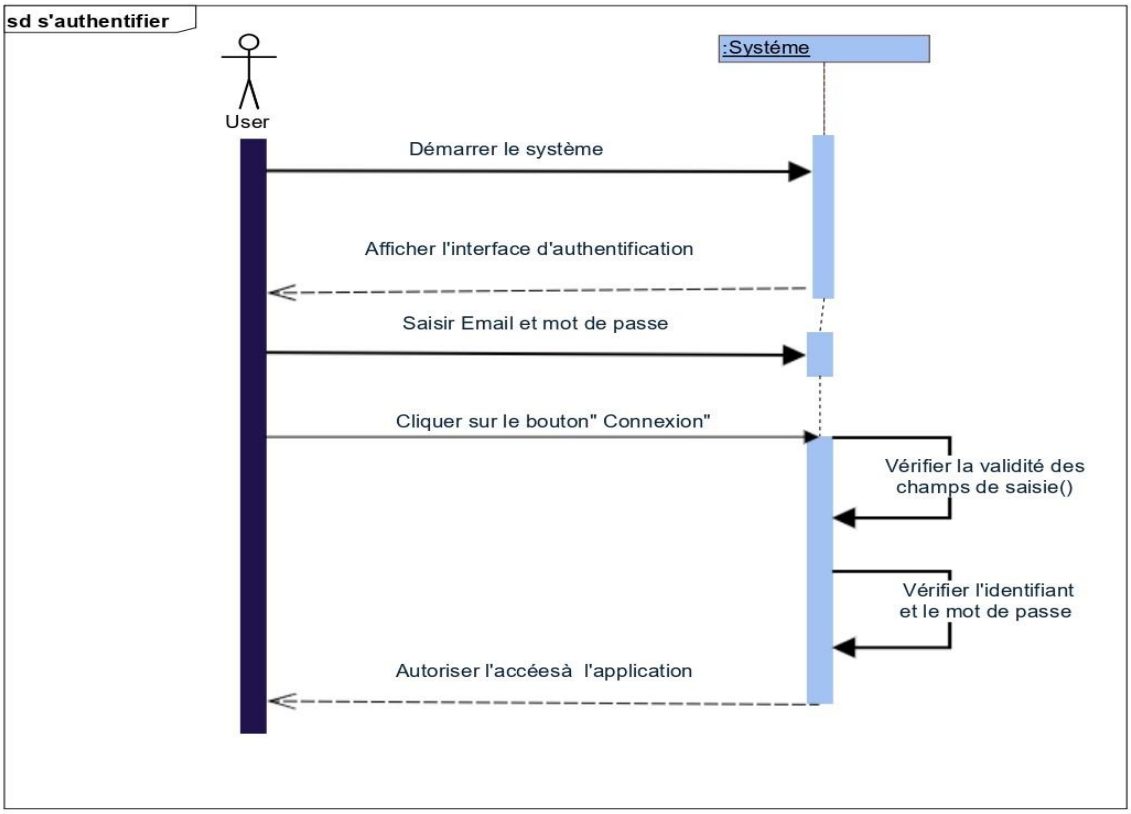


**Figure 11** Diagramme de séquence système du cas d'utilisation "s'authentifier"

### 3.1.2 Valider les déclarations

L'agent HSE a pour rôle de traiter et valider les déclarations établies par le déclarant. La figure ci-après interprète le scenario qui se déroule entre le système et l'agent HSE lorsque ce dernier valide une déclaration.

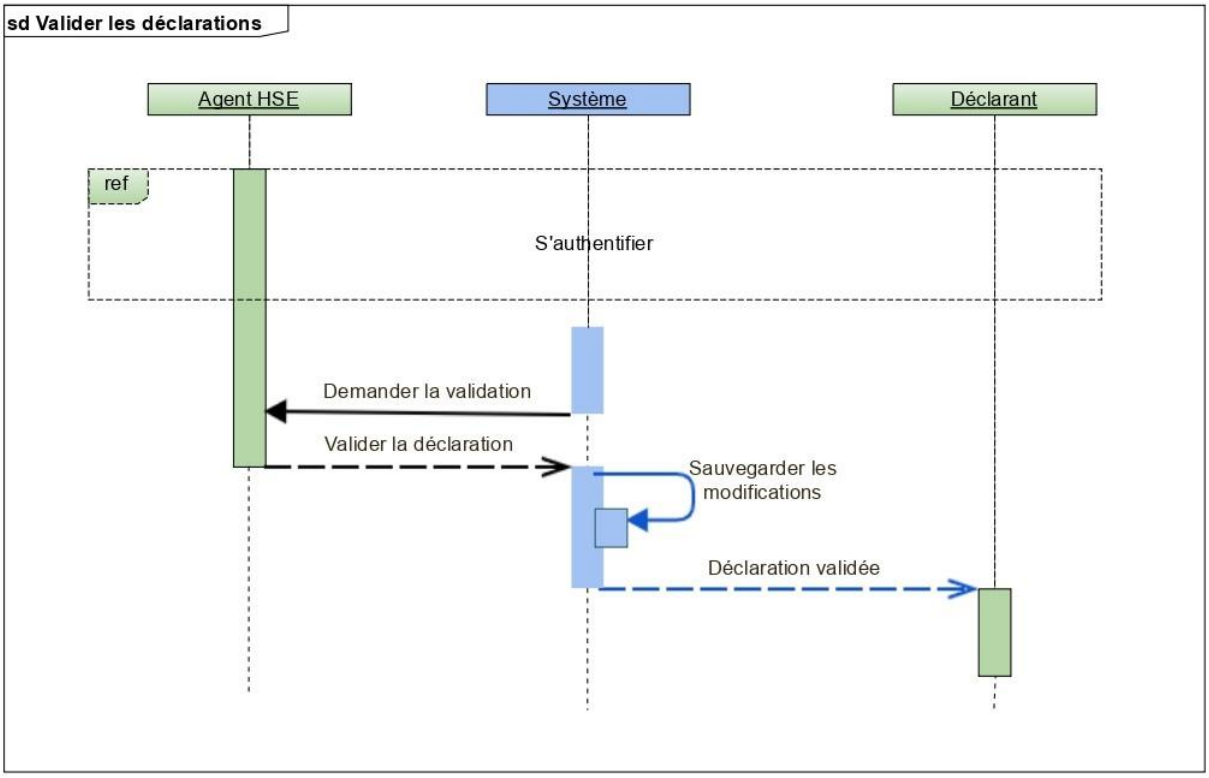


**Figure 12** Diagramme de séquence système du cas d'utilisation "Valider les déclarations"

# 4 Conception

Une fois les besoins du futur système sont recueillis et analysés, nous passons désormais à la phase de conception. Durant cette phase nous allons réaliser quelques diagrammes de séquence détaillés, et nous terminerons par un diagramme de classe de conception. Cette phase nous permettra de définir l'architecture et le comportement de l'application.

## 4.1 Les diagrammes de séquences détaillés

Les diagrammes ci-dessous sont une représentation graphique détaillée des interactions selon leur ordre chronologique. Dans ces diagrammes nous avons noté : IU : interface utilisateur, V : Vue, C : Controller, M : Modèle et BD pour Base de Données.

### 4.1.1 Authentification détaillée

Chaque membre du système dispose d'un email et d'un mot de passe qu'il doit saisir dans le formulaire d'authentification afin d'accéder à son espace.

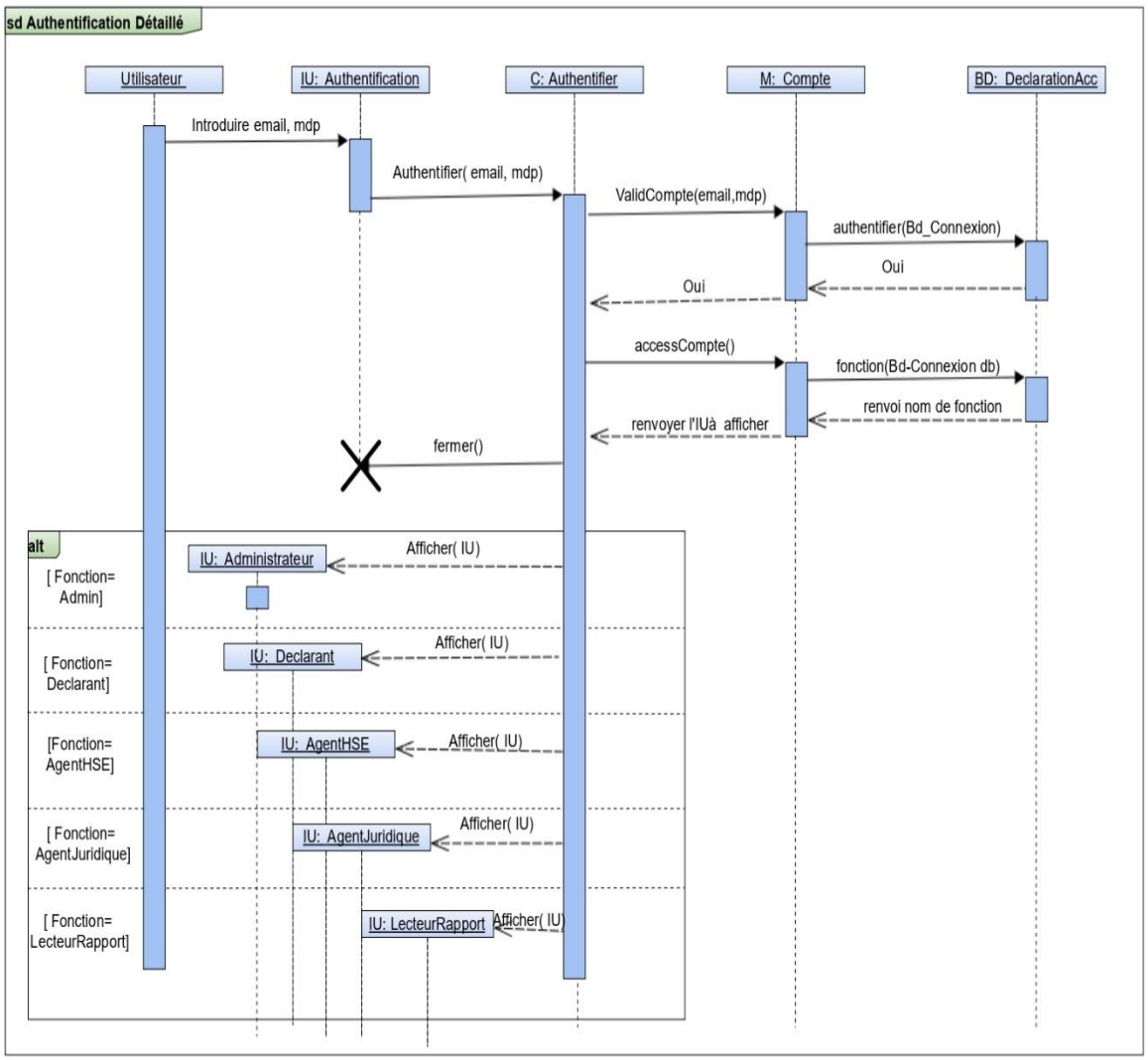


**Figure 13** Diagramme de séquence détaillé du cas d'utilisation "s'authentifier"

### 4.1.2 Ajouter une déclaration

Le digramme exposé dans la figure 14 ci-dessous, décrit les scénarios possibles lors d'une opération d'ajout d'une déclaration de non-conformité.

- Le déclarant clique sur le bouton créer une nouvelle déclaration et il remplit les champs nécessaires.
- Le système vérifie la validité des champs.
- Si tous les champs sont corrects, le système prend en charge les informations introduites et les enregistre dans la base de données.
- La déclaration créée s'affiche sur la liste des déclarations de non-conformité.

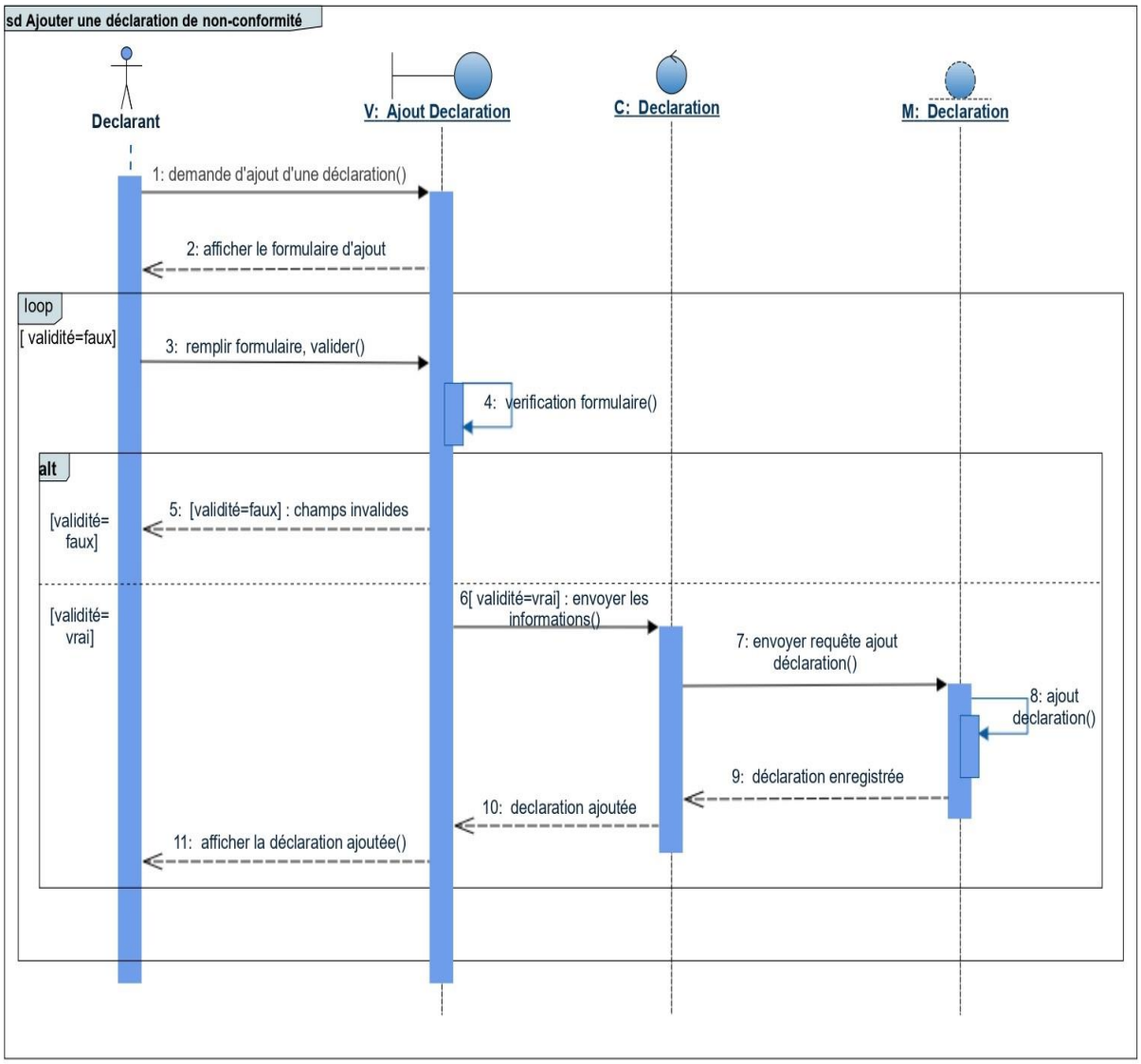


**Figure 14** Diagramme de séquence détaillé du cas d'utilisation "Ajouter une déclaration"

### 4.1.3 Modifier une déclaration

Le diagramme ci-dessous illustre les détails de l'opération de modification d'une station par l'administrateur de l'application.

- Le déclarant sélectionne une déclaration d'accident pour la modifier depuis la liste des déclarations d'accidents.
- Une page s'affiche et contient le formulaire à modifier.
- Le déclarant modifie les champs nécessaires.
- Si tous les champs sont corrects, le système prend en charge les informations introduites et les enregistre dans la base de données.

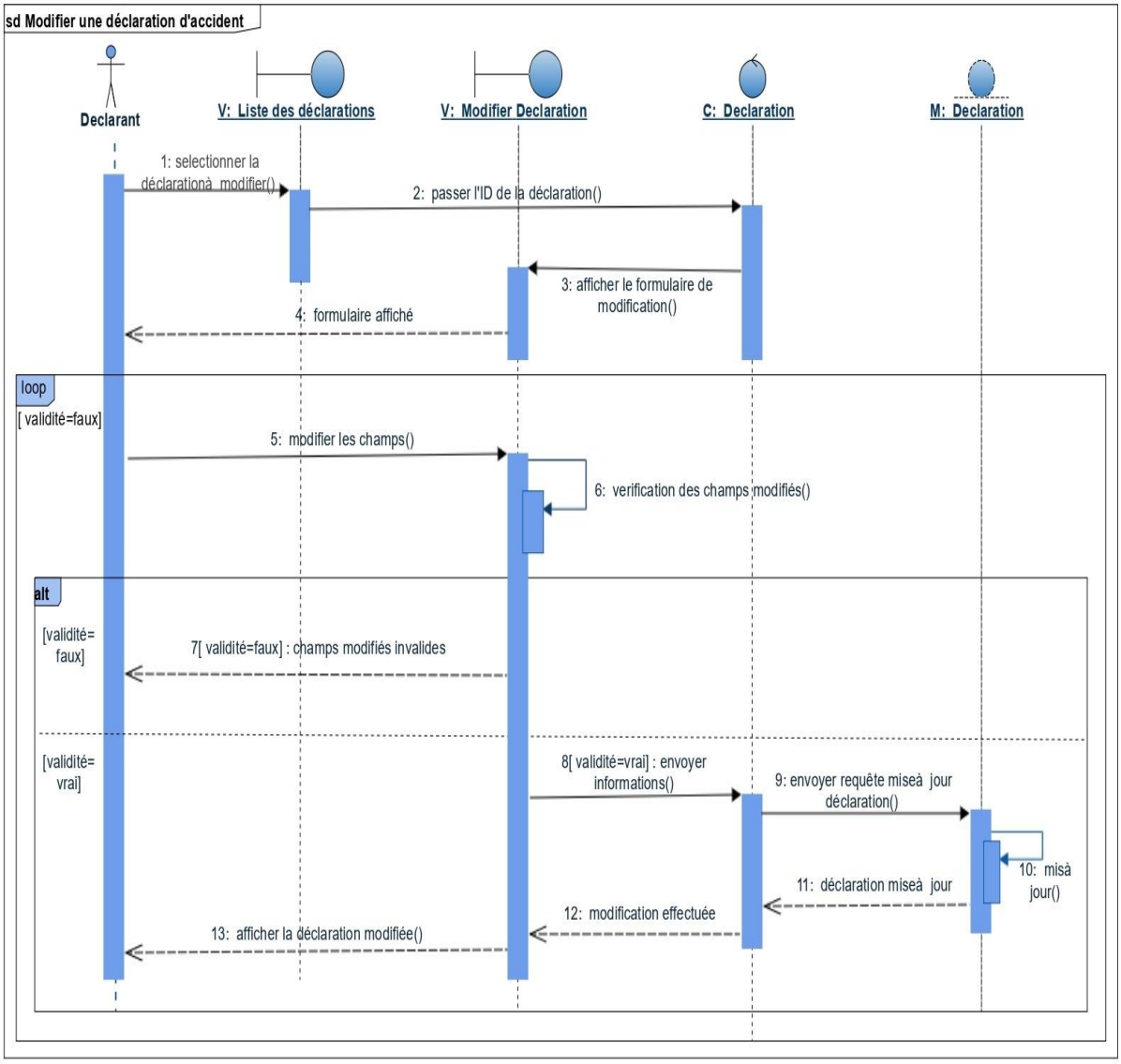


**Figure 15** Diagramme de séquence détaillé du cas d'utilisation « Modifier une déclaration d'accident »

## 4.2 Diagramme de classes

C'est une représentation utilisée dans la phase de conception d'une application pour présenter les classes entités et les différentes relations entre celles-ci.

Une classe est constituée d'un ensemble de fonctions (méthodes) et de données (attributs).

### 4.2.1 Dictionnaire de données

Le dictionnaire des données est un document qui regroupe toutes les données que vous aurez à conserver dans votre base (et qui figureront donc dans le diagramme de classe). Pour chaque donnée, il indique :

- le **code mnémonique :** il s'agit d'un libellé désignant une donnée.
- la **désignation :** il s'agit d'une mention décrivant ce à quoi la donnée correspond (par exemple «Action corrective») ;
- le **type de donnée :A** alphabétique**, N** numérique, **AN** alphanumérique, date et booléen.
- la **taille :** elle s'exprime en nombre de caractères ou de chiffres. Dans le cas d'une date au format JJ-MM-AAAA, on compte également le nombre de caractères, soit 10 caractères. Pour ce qui est du type booléen, nul besoin de préciser la taille.
- et parfois des **remarques** ou **observations** complémentaires (par exemple si une donnée est strictement supérieure à 0, etc.) [W7].

| **Classe** | **Code mnémonique** | **Désignation** | **Type** | **Taille** | **Remarque** |
|---|---|---|---|---|---|
| Direction | idDirection | Identifiant numérique d'une direction | N | | |
| | nomDirection | Nom de la direction | AN | 200 | |
| | departement | Département concerné | AN | 200 | |
| | service | Service du département | AN | 200 | |
| Declaration | idDeclaration | Identifiant numérique d'une déclaration | N | | |
| | description | Description d'une déclaration | AN | 1500 | |
| | date | Date de la déclaration | Date | 10 | Format JJ-MM-AAAA |
| Utilisateur | idUser | Identifiant numérique d'un utilisateur | N | | |
| | nom | Nom d'un utilisateur | A | 100 | |
| | prenom | Prénom d'un utilisateur | A | 100 | |

| | | | | | |
|---|---|---|---|---|---|
| | dateNaiss | Date de naissance d'un utilisateur | Date | 10 | Format JJ-MM-AAAA |
| | adresse | Adresse de l'utilisateur | AN | 150 | |
| | email | Adresse e-mail d'un utilisateur | AN | 100 | |
| | tel | Numéro de téléphone portable d'un utilisateur | AN | 15 | |
| | fonction | Fonction de l'utilisateur (l'employé) | AN | 100 | |
| | processus | Processus d'une non-conformité | AN | 100 | |
| | numFnc | Numéro FNC d'une non-conformité | AN | 30 | |
| | cause | Cause identifiée par le groupe d'analyse | AN | 100 | |
| | idSource | Identifiant numérique d'une source | N | | |
| | libelle | Libelle de la source | A | 100 | |
| Derogation | idDerogation | Identifiant numérique d'une dérogation | N | | |
| | demandeur | Nom et prénom du demandeur de la dérogation | A | 100 | |
| | decision | Décision concernant la dérogation | AN | 200 | |
| | description | Description de la dérogation | AN | 1000 | |
| | motif | Motif de la dérogation | AN | 200 | |
| ActionCor | idActionCor | Identifiant numérique d'une action corrective | N | | |
| | responsable | Nom et prénom du responsable de l'action corrective | A | 100 | |
| | delai | Délai du traitement | N | 3 | S'exprime en nombre de jours |
| | efficacite | Efficacité de l'action corrective | Bool | | |
| | commentaire | Commentaire sur l'action corrective | AN | 1000 | |
| Accident | type | Type d'accident | A | 200 | |
| | meteo | Météo | AN | 200 | |

| | | | | | |
|---|---|---|---|---|---|
| | enquete | Enquêter sur l'accident ou non | Bool | | |
| | detailDommage | Détails sur les dommages causés | AN | 1000 | |
| PertePotentielle | idPerte | Identifiant numérique d'une perte | N | | |
| ActionRecommandee | idActionRecom | Identifiant numérique d'une action recommandée | N | | |
| | dateLimite | Date limite du traitement de l'action recommandée | Date | 10 | Format JJ-MM-AAAA |
| | fonctionRecom | Fonction du recommandeur | AN | 100 | |
| Vehicule | chauffeur | Nom et prénom du chauffeur | A | 100 | |
| | immatriculation | Identifiant alphanumérique d'un véhicule | AN | 100 | |
| | pcn | Numéro PCN d'un véhicule | AN | 100 | |
| | dateDelivre | Date de délivrance | Date | 10 | Format JJ-MM-AAAA |
| Bien | idBien | Identifiant numérique d'un bien | N | | |
| | proprietaire | Nom et prénom du propriétaire | A | 100 | |
| | type | Type de bien | AN | 100 | |
| GroupeAnalyse | idGrp | Identifiant numérique d'un groupe d'analyse des non-conformités | N | | |
| | nom | Nom du groupe d'analyse | AN | 100 | |
| | fonction | Fonction du groupe | AN | 100 | |

**Tableau 6** Dictionnaire de données

Ci-dessous le diagramme de classes de notre application :

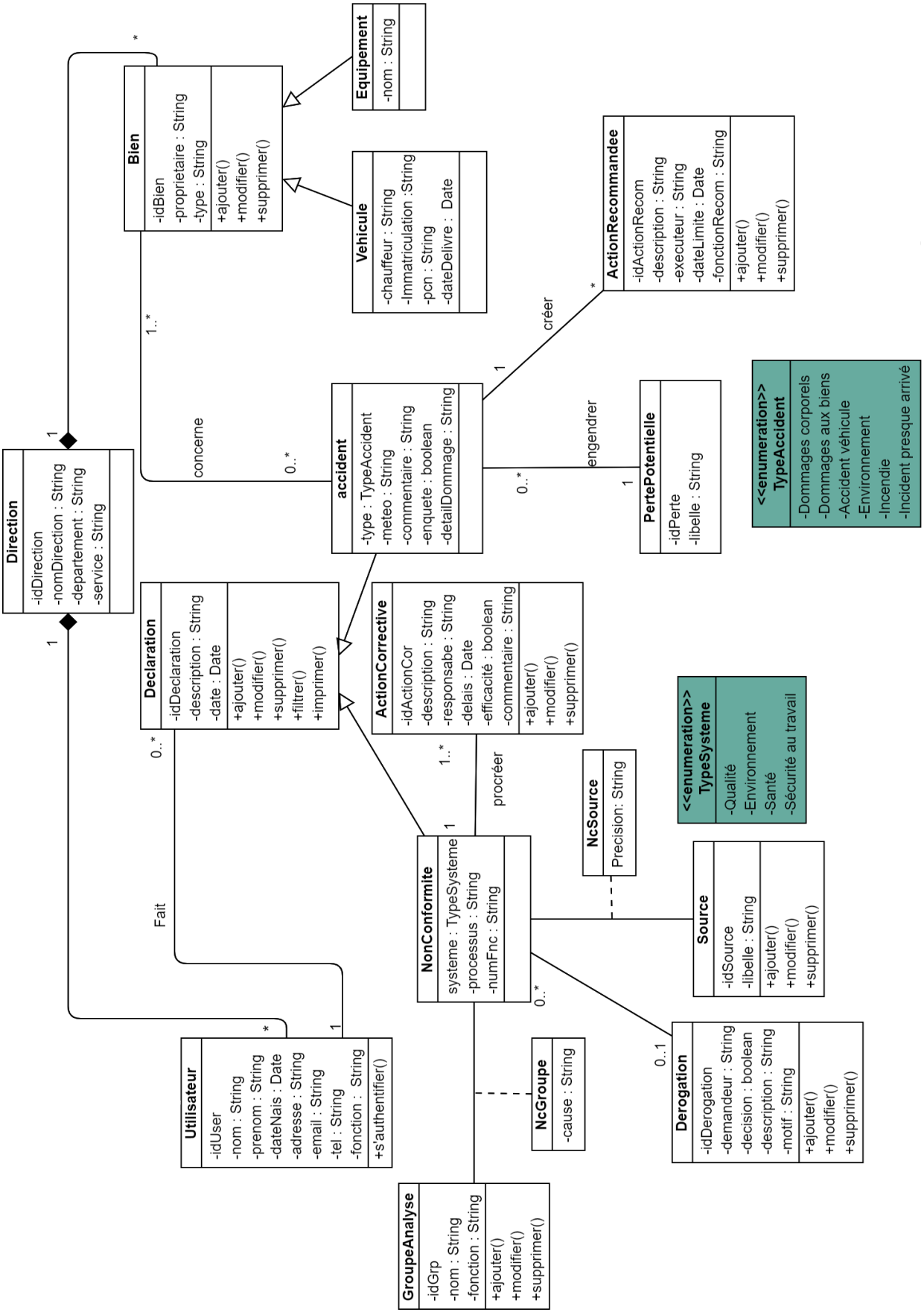


**Figure 16** Diagramme de classes

## 4.3 Modélisation logique des données

La modélisation logique des données est une représentation des données, issues de la modélisation conceptuelle puis des données.

Dans ce qui suit, nous présentons les différentes règles de passages d'un diagramme de classe vers un modèle relationnel.

### 4.3.1 Règles de passage d'un diagramme de classe vers un modèle relationnel

Dans notre projet, nous avons adapté les règles suivantes pour faire le passage du diagramme de classe vers le modèle relationnel.

Chaque classe sera représentée par une table dont les colonnes sont les attributs de cette classe.

- Les associations de types 1 : N, N : 1, 1 :1 se traduisent par la conversion des deux classes constituant cette association en deux tables dont une va contenir une clé étrangère pour référencer à l'autre table.

- Les associations de type N : M entre deux classes sont représentées par une nouvelle table qui prend pour clé primaire la concaténation des clés primaires des deux classes.

- Une association de classe entre deux classes est représentée par une nouvelle table qui prend pour clé primaire la concaténation des clés primaires des deux classes.

- L'agrégation sera traitée comme une association tout en tenant compte des cardinalités

- La composition sera traitée comme une association sauf qu'on doit ajouter une clause ONDELETE CASCADE.

Pour la généralisation (héritage) il existe trois stratégies de génération :

- Père et fils : La classe mère et la classe fille seront représentées par deux tables distinctes chacune.

- Père seulement : Seulement la classe mère sera représentée par une table portant ses attributs et les attributs de sa classe fille comme champs dans la table.

- Fils seulement : Seulement la classe fille sera représentée par une table portant ses attributs ainsi que les attributs hérités de la classe mère comme champs dans la table[4].

On a utilisé la stratégie **père et fils**.

### 4.3.2 Le schéma relationnel

**Declaration** (idDeclaration, date, description, #IdUser)

**Accident** (# idDeclaration, type, meteo, commentaire, enquete, detailDommage, #idPerte)

**ActionRecommandee** (idActionRecom, description, executeur, dateLimite, fonctionRecom, #idDeclaration)

**NonConformite** (#idDeclaration, systeme, processus, numFnc, #idDerogation)

**Utilisateur** ( idUser, nom, prenom, dateNaiss, adresse, email, tel, fonction, #id_direction)

**ActionCorrective** (idActionCor, description, responsable, delais, efficacité, commentaire, #idDeclaration)

**Direction** (idDirection, nomDirection, departement, service)

**Bien** (idBien, type, proprietaire, #idDirection)

**Véhicule** (#idBien, immatriculation, pcn, chauffeur, dateDelivre)

**Equipement** (#idBien, nom),

**Concerne** (#id_bien#id_declaration),

**Derogation** (idDerogation, description, demandeur, decision, motif)

**Source** (idSource, libelle)

**NcSource** (#idSource #idDeclaration, precision)

**PertePotentielle** (idPerte, libelle)

**GroupeAnalyse** (idGrp, nom, fonction)

**NcGroupe** (#idGrp#idDeclaration, cause)

## 5 Conclusion

Tout au long de ce chapitre, nous avons présenté les diagrammes UML que nous avons utilisés lors de l'analyse et la conception de notre application afin de garantir la fiabilité et l'efficacité de la phase de réalisation.

Le chapitre suivant sera consacré à la réalisation de l'application en implémentant la conception et en présentant les différentes interfaces.

# *Chapitre 3 :*
# *Réalisation*

*Chapitre* 3

# *Réalisation*

## 1 Introduction

Après avoir terminé la conception détaillée de notre application, nous traitons dans le présent chapitre les détails liés à la réalisation de l'application.

Pour cela, nous exposons tout d'abord le choix de l'environnement de travail adopté afin de réussir la réalisation de l'application, et nous décrivons les étapes d'implémentation suivies de quelques interfaces de l'exécution de l'application pour illustrer quelques fonctionnalités de notre système.

## 2 Environnement de réalisation

**Microsoft Visual Studio :**

Microsoft Visual Studio est un environnement de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles en se basant sur divers langages de programmation ; Visual Basic, C++ et C# qui ai le plus populaire dans ce sens du développement des applications [W8].

**Microsoft SQL Server :**

Microsoft SQL Server est un système de gestion de base de données (abrégé en SGBD) incorporant entre autres un SGBDR (SGBD relationnel) utilisant un langage informatique normalisé servant à exploiter des bases de données relationnelles, développé et commercialisé par la société Microsoft. Microsoft Visual Studio possède une version allégée et intégrée de ce SGBD, qui permet de manipuler les données en local [W9].

## 3 Langages et technologies utilisés

### 3.1 Langage de programmation

C'est le langage de la nouvelle version de Visual Studio .NET le plus utilisé, un langage dérivé du C++, qui reprend des caractéristiques du langage Java. C# peut être utilisé pour créer des applications Windows et Web. Le langage C# ajoute au C++ les techniques de construction des programmes sur base des composants prêts à l'emploi avec propriétés et événements.

Il a les caractéristiques suivantes :

- Purement orienté objet.
- Types précisément conformes à l'architecture .NET et vérifications de type plus élaborées.
- Gestion automatique de la mémoire (ramasses miettes) [W10].

## 3.2 La technologie ASP .NET MVC

ASP.NET MVC est un sous-Framework du grand Framework .NET, plus récent et a fait son apparition en 2009. Cette façon de penser permet la réalisation des applications web qui s'appuie sur le célèbre patron de conception MVC (Model-View-Controller), qui existe depuis 1979, et qui est la référence conceptuelle de développement chez beaucoup de développeurs.

**MVC en ASP .NET**

MVC est une architecture de conception qui est adapté par beaucoup de développeurs. Il assure la séparation du ''Model'' qui représente les données concernées par le traitement qui vont être stockés dans une base de données, dans des fichiers XML, ou bien qui s'obtiennent à l'aide d'un web service, et le ''Controller'' qui s'occupe de la récupération et le traitement de ces données pour les transférer enfin à la ''View'' qui permet à son tour de présenter ces données au client que ce soit un client Web ou un client Mobile [5].

**Description de l'architecture ASP.NET MVC :**

1. Le client envoie une requête HTTP de type GET ou POST.
2. Toutes les requêtes HTTP sont traitées par un Contrôleur Frontal (MvcHandler) fourni par Asp.net. Pour chaque action de l'URL, le Contrôleur frontal devrait exécuter une opération associée à cette action. Cette opération est implémentée dans une classe appelée « Controller » qui représente un sous-contrôleur.
3. Le sous contrôleur exécute le traitement associé à l'action en faisant appel à la couche métier et récupère le résultat.
4. Le sous contrôleur stocke le résultat dans le modèle fourni par Asp.Net MVC.
5. Le sous contrôleur retourne le nom de la vue et le modèle au contrôleur frontal MvcHandler.
6. Le contrôleur frontal fait appel à la vue et lui transmet le modèle.
7. La vue est soumise à un moteur de Template qui récupère les résultats à partir du modèle et génère un rendu HTML qui est retourné au contrôleur frontal.
8. MvcHandler renvoie la réponse HTTP au client. Cette réponse http contient le code HTML générée par la vue [W11].

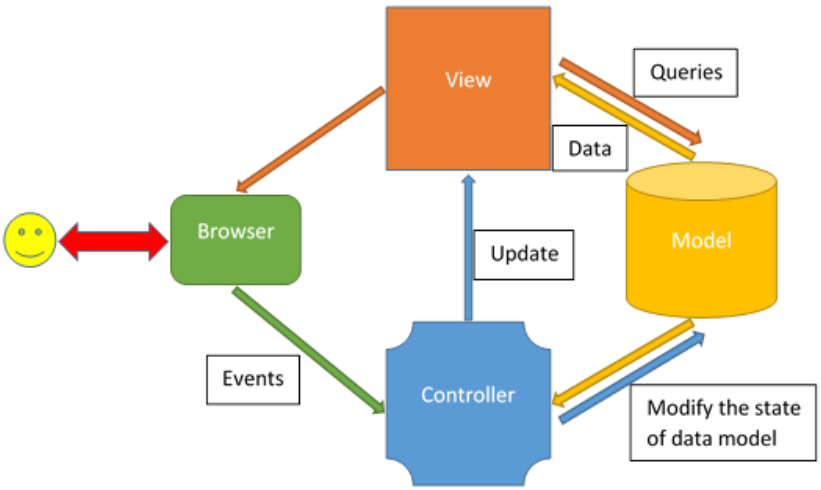


**Figure 17** Description de l'architecture ASP.NET MVC

**Répartition de l'architecture**

On a organisé notre architecture en trois couches (3-tiers) :

- **La couche DAO** (Data Access Object) : c'est la couche de données responsable de l'accès à la base de donnes avec un ORM implémenté avec EntityFramework.
- **la couche métier** : C'est généralement la couche la plus stable de l'architecture. Elle ne change pas si on change l'interface utilisateur ou la façon d'accéder aux données nécessaires au fonctionnement de l'application. Dans cette couche on a implémenté le patron Repository qui est utilisé par la couche interface utilisateur pour la logique métier et l'accès aux données.
- **La couche interface utilisateur** : permet à l'utilisateur de piloter l'application et d'en recevoir des informations. Les couches métiers et dao sont utilisées via des interfaces. Ainsi la couche métier ne connaît de la couche dao que son ou ses interfaces et ne connaît pas les classes les implémentant. C'est ce qui assure l'indépendance des couches entre elles : changer l'implémentation de la couche dao n'a aucun impact sur la couche métier tant qu'on ne touche pas à la définition de l'interface de la couche dao. Il en est de même entre les couches interface utilisateur et métier.

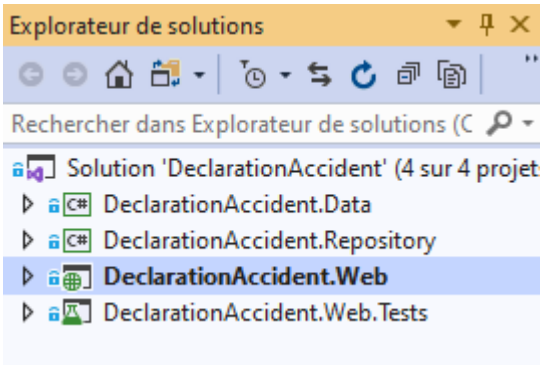


***Figure 18*** *Architecture de l'application*

## 3.3 ORM EntityFramework et Identity EF

Dans notre projet on a adopté le Framework d'authentification ''Identity EF'' et l'ORM (Object Relational Mapping) EntityFramework qui est un système qui permet de faire les correspondances entre les schémas de la base de données et les classes entités, et qui permet donc la génération du code à partir du model relationnel (les tables relationnels de la BD) [W12].

## 3.4 Bootstrap pour CSS3 et HTML5

Bootstrap est un framework développé par Twitter. Proposé en open source, ce framework utilise les langages HTML, CSS et JavaScript. Il est conçu pour développer des sites avec un design responsive, qui s'adapte à tout type d'écran. Il fournit des outils avec des styles déjà en place pour des typographies, des boutons, des interfaces de navigation et bien d'autres encore. On appelle ce type de framework un "Front-End Framework" [W13].

## 3.5 JQuery

JQuery est un Framework JavaScript sous licence libre qui permet de faciliter des fonctionnalités communes de JavaScript. L'utilisation de cette bibliothèque permet de gagner du temps de développement en exploitant des fonctions JavaScript prédéfinie [W14].

## 3.6 KnockoutJS

KnockoutJS est une bibliothèque de fichiers en Javascript qui permet de créer des interfaces utilisateurs d'auto-mise à jour des objets Javascript. Ses atouts :

- Bibliothèque JavaScript très petite et légère, "open source" qui fonctionne avec tous les Frameworks web.
- KnockoutJS est entièrement documenté. Le site officiel a une documentation complète, y compris documentation de l'API, des exemples et des didacticiels interactifs.
- Fonctionne sur tous les navigateurs courants [W15].

# 4 Présentation et description des interfaces

Nous présentons dans cette partie quelques interfaces de notre application.

- **Authentification :**

Au lancement de l'application, soit par l'administrateur, ou quelconque employé quel que soit son rôle, la première interface qui s'ouvre c'est celle de l'authentification. L'utilisateur insère ses informations, il clique sur « Se connecter », le système vérifie que ses informations existent dans la base de données et sont correctes, et lui autorise la connexion, sinon, le système l'informe que la tentative de connexion a échoué.

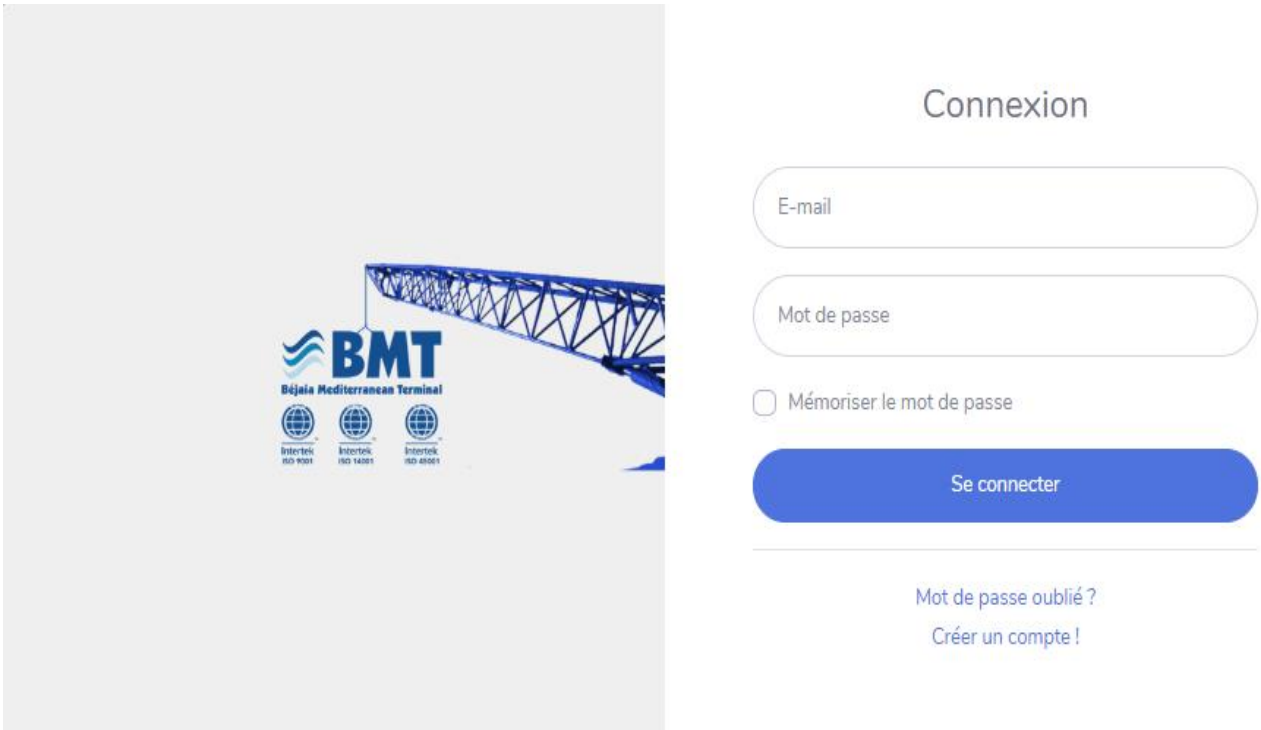


**Figure 19** Interface "Auhentification"

**Liste des accidents :**

Cette interface représente le panneau qui permet de consulter toutes les déclarations d'accidents, les déclarations en cours de traitement et celles déjà traitées. Il permet également de filtrer les accidents selon divers critères. On peut aussi afficher les détails de chaque déclaration, modifier, supprimer ou imprimer.

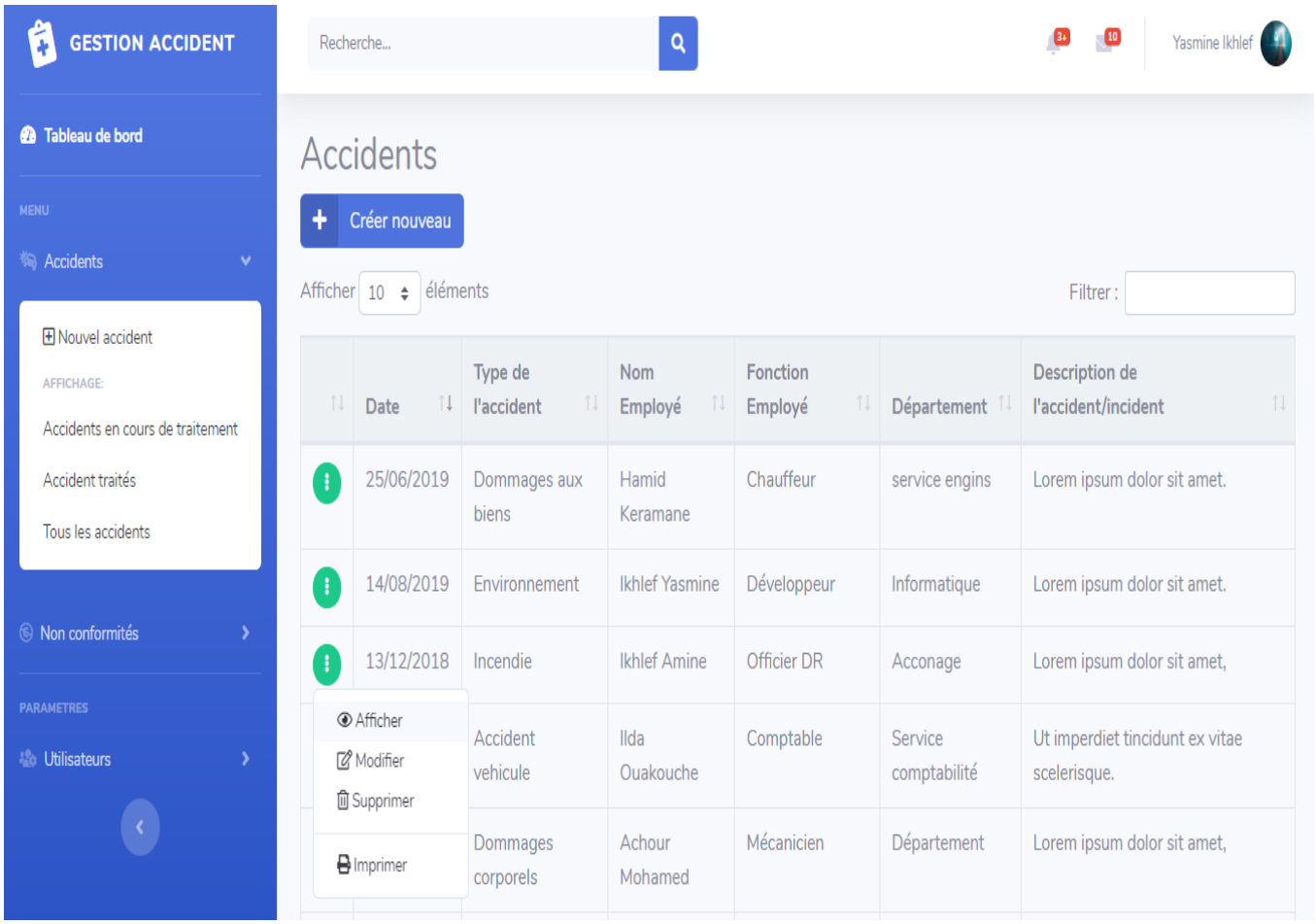


**Figure 20** Interface "Liste des accidents"

**Modifier un accident :**

Cette interface représente le formulaire de modification d'une déclaration d'accident et sa mise à jour.

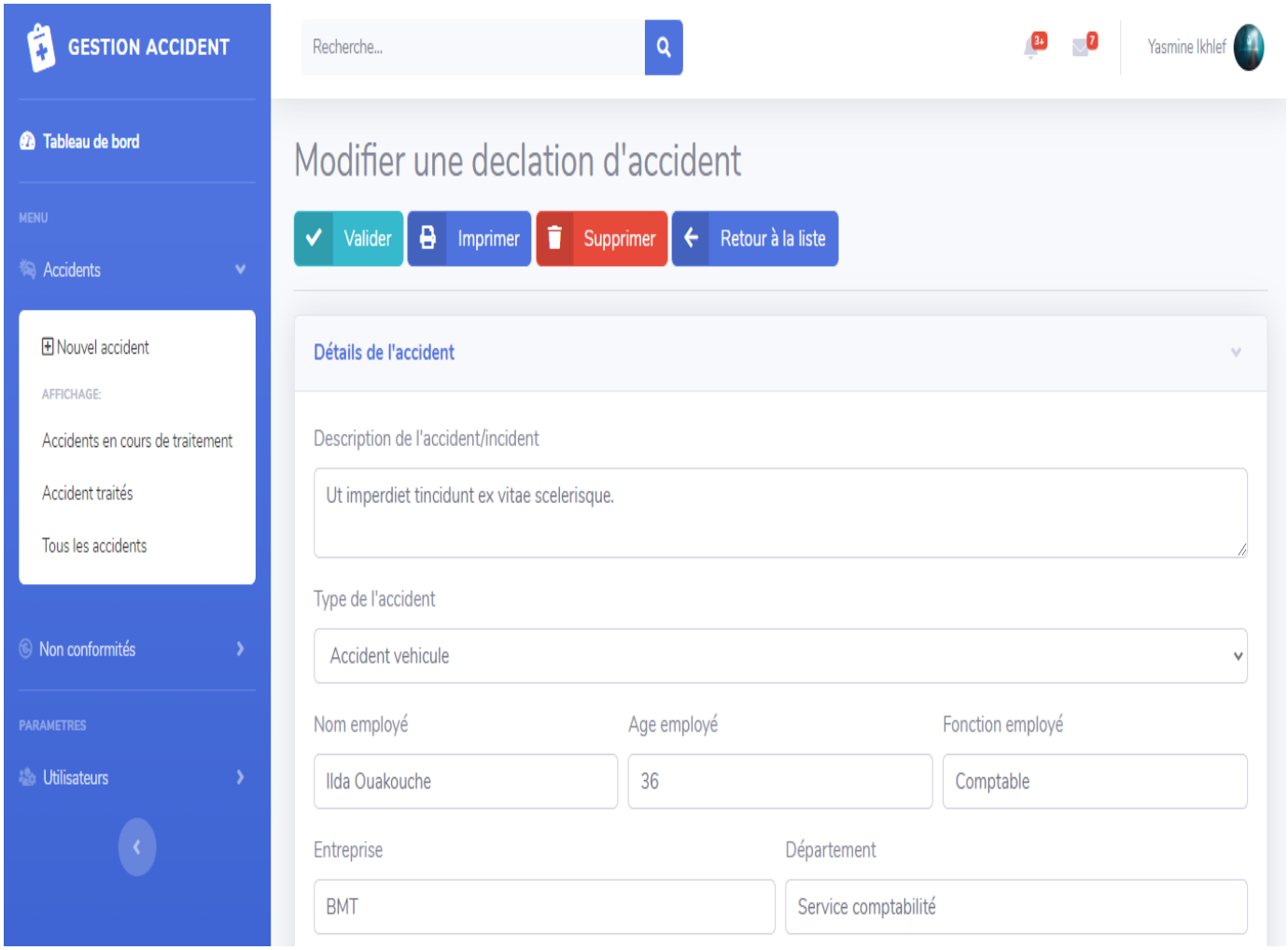


**Figure 21** Interface "Modifier un accident"

**Création d'une déclaration de non-conformité :**

Cette interface permet au déclarant de créer une nouvelle déclaration d'accident. Il remplit le formulaire affiché avec les informations demandées dans chaque section.

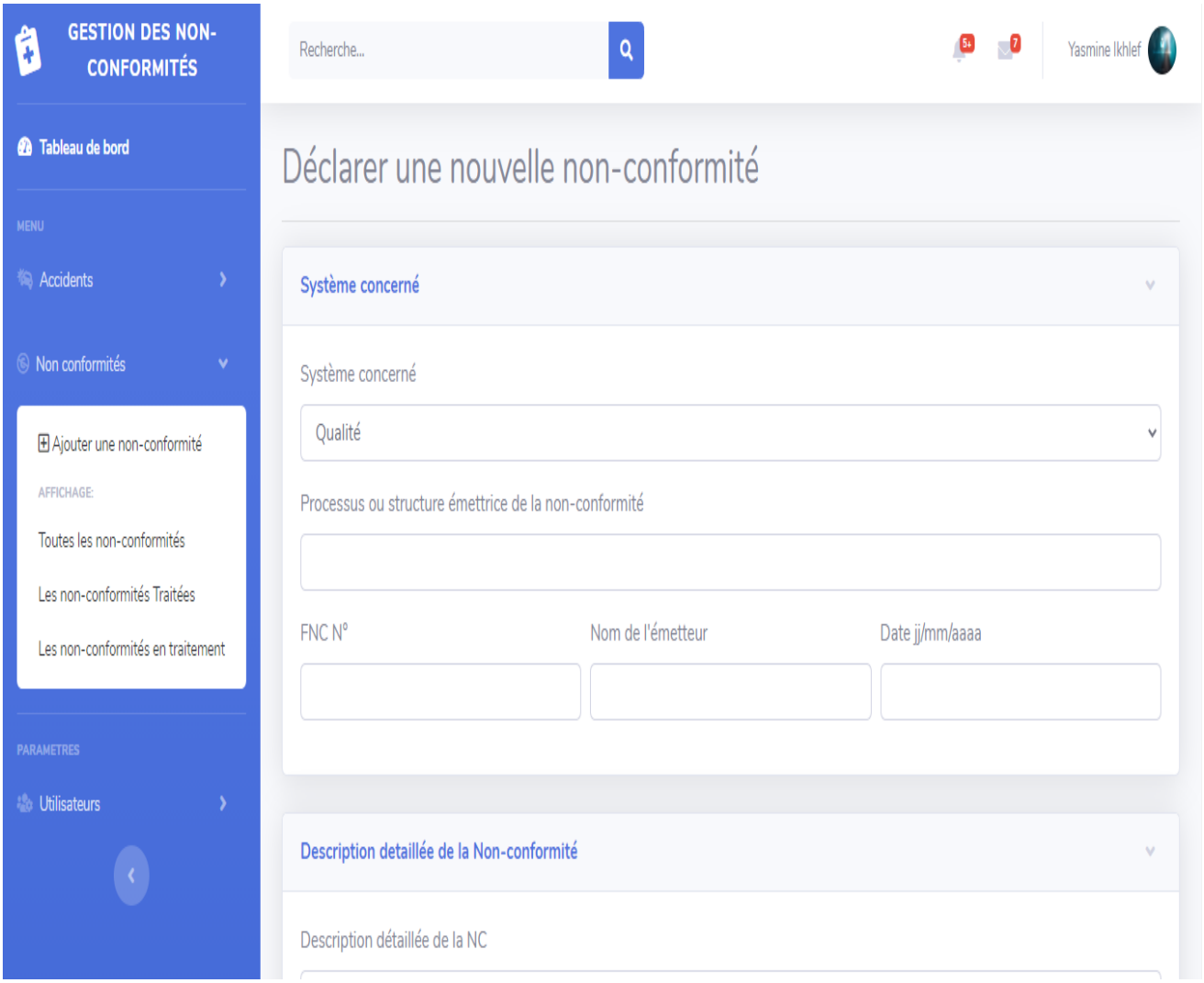


**Figure 22** Interface « Déclarer une nouvelle non-conformité »

**Détails d'une non-conformité :**

L'utilisateur (le déclarant, l'agent HSE, l'agent juridique) peut consulter les détails d'une non-conformité, et une fois la déclaration de cette dernière est traitée, les actions correctives seront affichées. Avec les possibilités de modifier, envoyer, imprimer, supprimer et retourner à la liste des non-conformités.

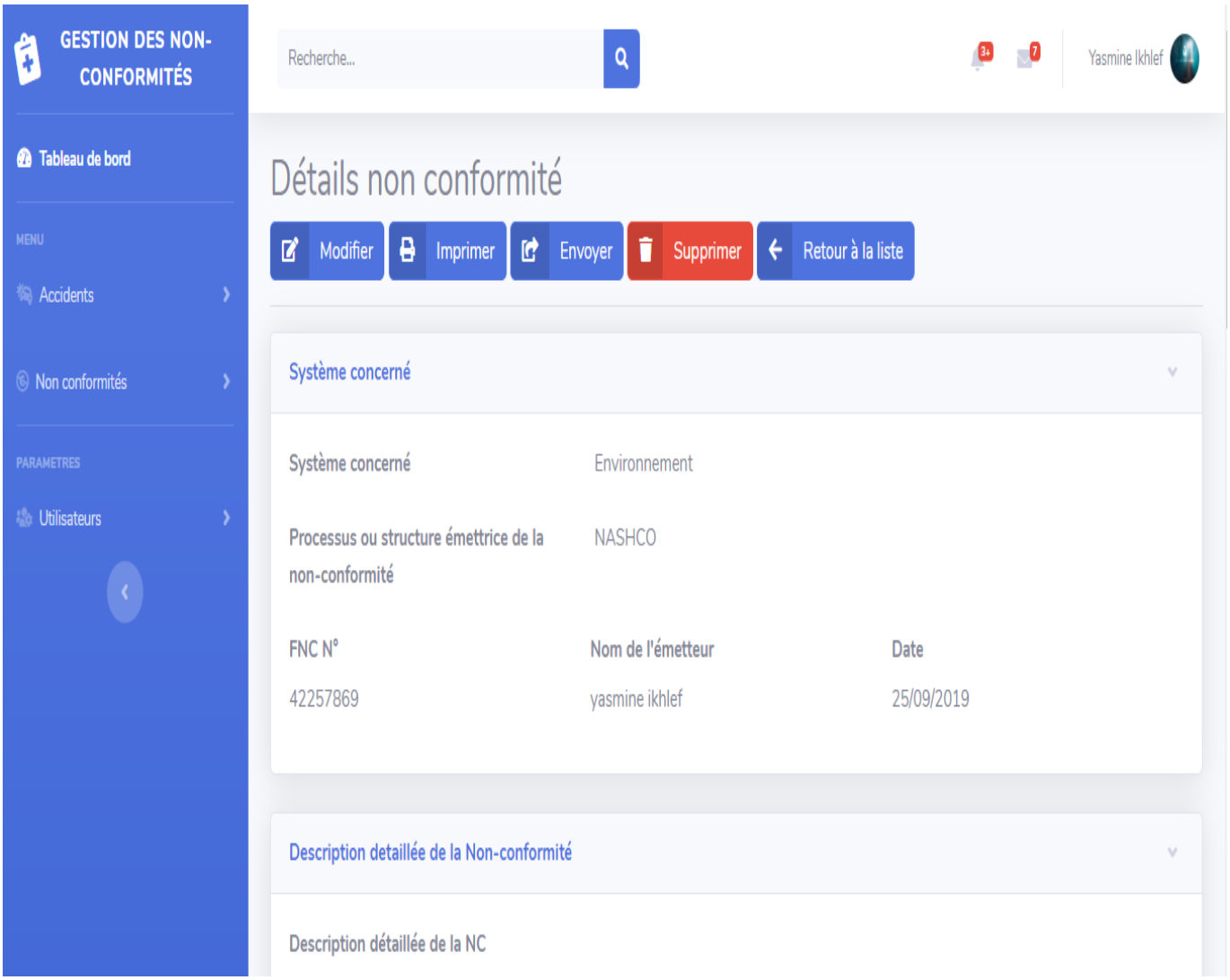


**Figure 23** Interface « Détails d'une non-conformité»

**Tableau de bord :**

Le tableau de bord est conçu pour avoir un aperçu sur les différentes progressions et statistiques, comme le nombre d'accidents et de non-conformités par mois ou par années, et les accidents par type.

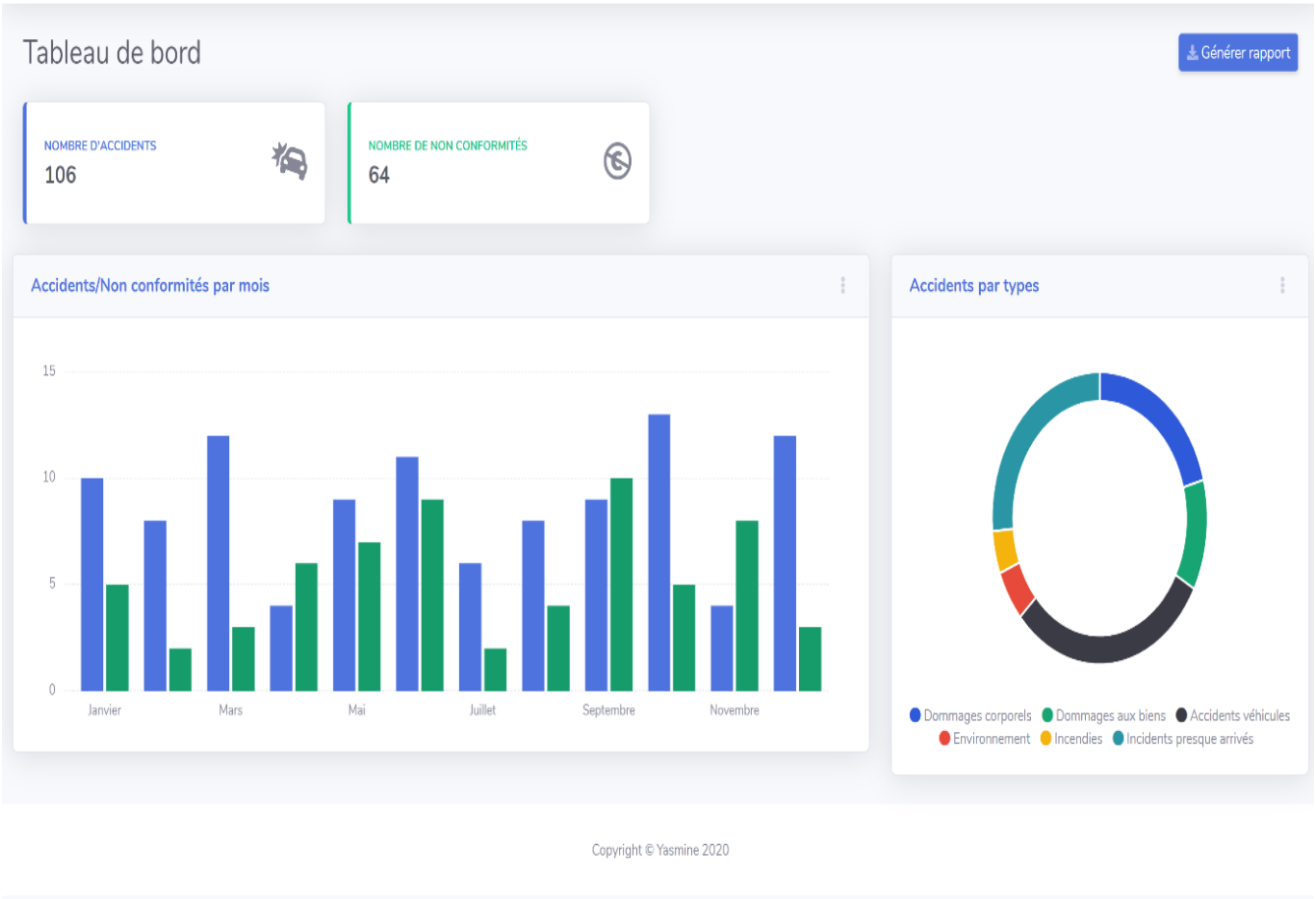


**Figure 24** Interface « Tableau de bord »

## 5 Conclusion

Ce dernier chapitre nous a permis de présenter l'environnement de travail et les technologies utilisées pour réaliser notre application ainsi que quelques interfaces graphiques.

# *Conclusion Générale*

# Conclusion générale

Dans ce projet, nous avons réalisé une application web pour la gestion du système management intégré de BMT (déclaration des accidents/incidents et non-conformités), une application flexible et facile à manipuler. Les objectifs que nous nous sommes fixés ont donc été, en grande partie, atteints.

En outre, ce projet nous a permis d'apprendre beaucoup de choses importantes dans un vaste domaine qui englobe différentes technologies : le C#, ASP.NET MVC, Identity EntityFramework, etc. Ainsi qu'en termes de modélisation, conception et de programmation, et cela, pour pouvoir concrétiser notre bagage théorique et pratique et nous familiariser avec de nouveaux environnements de développement.

Notre travail a été donc basé, en premier lieu, sur l'étude du contexte du projet et l'analyse des cas d'utilisation des différents acteurs en interaction avec le système. En deuxième lieu, nous avons conçu et développé les interfaces servant à réaliser leurs tâches appropriées en utilisant plusieurs technologies.

Cependant, vu la taille importante du projet et la limitation du temps, plusieurs améliorations peuvent être envisagées. En perspective, on pourrait notamment implémenter des fonctionnalités telles que l'intégration du tableau d'analyse dans SharePoint Online en utilisant l'outil de modélisation Power BI.

# Bibliographie

**[1]** P.ROQUES. UML 2 Modéliser une application Web. EYROLLES, 4ème édition, 2007.

**[2]** B. CHARROUX, A. Osmani, Y. Thierry-Mieg, UML 2, Pratique de la modélisation, PEARSON Education, 2ème édition, 2008.

**[3]** F. VALLÉE & P.ROQUES. UML 2 en action de l'analyse des besoins à la conception. EYROLLES, 4ème édition, 2007.

**[4]** S. Crozat. Passage UML Relationnel : classes et associations (Contributions : Benjamin Lussier, Antoine Vincent). Compiègne : UTC. Publications UTC-COSTECH, Cours, 2009,46p.

**[5]** M. YOUSSFI. Dot Net : Programmation Orientée Objet C#. Casablanca : ENSET, Université Hassan II Casablanca, Cours, Publié en 25/04/2014, 122p.

# Webographie

**[W1]** BMT- Bejaia Mediterrannean Terminal, https://bejaiamed.com/ (Consulté le 22/03/2020)

**[W2]** Up : Unified Process, https://sabricole.developpez.com/uml/tutoriel/unifiedProcess/ (Consulté le 02/04/2020)

**[W3]** RUP, https://www.nutcache.com/fr/blog/methode-rup/ (Consulté le 26/04/2020)

**[W4]** RUP, https://www.techterms.com/definition/rup (Consulté le 29/04/2020)

**[W5]** UML, https://laurent-audibert.developpez.com/Cours-UML/ (Consulté le 08/08/2020)

**[W6]** Identityserver4 https://identityserver4.readthedocs.io/ (Consulté le 14/03/2020)

**[W7]** Base de données relationnelles, https://ineumann.developpez.com/tutoriels/merise/initiation-merise/ (Consulté le 13/04/2020)

**[W8]** Visual Studio, https://visualstudio.microsoft.com/fr/ (Consulté le 22/07/2020)

**[W9]** ORM EntityFramework, http://dictionnaire.sensagent.leparisien.fr/ (Consulté le 19/05/2020)

**[W10]** https://openclassrooms.com/fr/courses/1526901-apprenez-a- developper-en-c (Consulté le 01/06/2020)

**[W11]** Cours ASP.NET, https://www.clicours.com/cours-asp-net-mvc-premiers-pas/ (Consulté le 05/06/2020)

**[W12]** ASP.NET MVC, https://docs.microsoft.com/en-us/aspnet/mvc/ (Consulté le 07/06/2020)

**[W13]** Introduction Bootstrap, https://getbootstrap.com/docs/4.4/getting-started/introduction/ (Consulté le 22/08/2020)

**[W14]** jQuery,https://www.w3schools.com/jquery/jquery_get_started.asp (Consulté le 11/09/2020)

**[W15]** Getting Started With Knockout.js, https://www.c-sharpcorner.com/article/getting-started-with-knockout-js/ , (Consulté le 19/09/2020)

*Annexe*

# Extraits du code de l'application

Dans ce qui suit nous allons donner quelques extraits du code de notre application :

## Accident Repository Pattern

Le patron Repository est utilisé par la couche interface utilisateur pour la logique métier et l'accès aux données.

...nAccident.Repository\Repositories\AccidentRepository.cs 1

```
 1  using DeclarationAccident.Data;
 2  using System;
 3  using System.Collections.Generic;
 4  using System.Data.Entity.Migrations;
 5  using System.Linq;
 6  using System.Text;
 7  using System.Threading.Tasks;
 8
 9  namespace DeclarationAccident.Repository.Repositories
10  {
11      public class AccidentRepository : IAccidentRepository
12      {
13          AccidentDB accidentDB = new AccidentDB();
14          public Accident CreateAccident(Accident acc)
15          {
16              var newAcc = accidentDB.Accidents.Add(acc);
17              accidentDB.SaveChanges();
18
19              return newAcc;
20          }
21
22          public void DeleteAccidentById(int accidentID)
23          {
24
25              var accident = accidentDB.Accidents.SingleOrDefault(a => a.Id ==
                  accidentID);
26              if (accident != null)
27              {
28                  //TODO: Tester le comportement si l'accident n'a pas d'actions
                      recommandées
29                  accidentDB.ActionRecommandees.RemoveRange
                      (accident.ActionRecommandees);
30                  accidentDB.Vehicules.RemoveRange(accident.Vehicules);
31                  accidentDB.Accidents.Remove(accident);
32                  accidentDB.SaveChanges();
33              }
34          }
35
36          public Accident GetAccidentById(int id)
37          {
38              return accidentDB.Accidents.SingleOrDefault(a => a.Id == id);
39          }
40
41          public List<Accident> GetAllAccidents()
42          {
43              return accidentDB.Accidents.ToList();
44          }
45
46
47      }
48  }
49
```

## Vue Index des non-conformités

C’est la vue qui affiche la liste des non-conformités.

...cidentMVC\DeclarationAccident.Web\Views\NC\Index.cshtml 1

```
1  @model IEnumerable<DeclarationAccident.Web.Models.NC>
2
3  @{
4      ViewBag.Title = "Index";
5      Layout = "~/Views/Shared/_sbadmin.cshtml";
6  }
7
8  <h2>Non conformités</h2>
9
10 <p>
11     <a href="/NC/Create" class="btn btn-primary btn-icon-split">
12         <span class="icon">
13             <i class="fas fa-plus"></i>
14         </span>
15         <span class="text">Créer nouveau</span>
16     </a>
17 </p>
18 <div class="table-responsive">
19     <table class="table table-bordered" id="listAccidents" width="100%"
         cellspacing="0">
20         <thead>
21             <tr>
22
23                 <th></th>
24                 <th>@Html.DisplayNameFor(model => model.Date)</th>
25                 <th>@Html.DisplayNameFor(model => model.TypeSystemeConrne)</
                     th>
26                 <th>@Html.DisplayNameFor(model => model.Description)</th>
27                 <th>@Html.DisplayNameFor(model => model.ProcessusEmetteur)</
                     th>
28                 <th>@Html.DisplayNameFor(model => model.ProcessusConcerne)</
                     th>
29                 <th>@Html.DisplayNameFor(model => model.ActionCorrective)</th>
30                 <th>@Html.DisplayNameFor(model => model.ResponsableAc)</th>
31             </tr>
32         </thead>
33         <tfoot>
34             <tr>
35                 <th></th>
36                 <th>@Html.DisplayNameFor(model => model.Date)</th>
37                 <th>@Html.DisplayNameFor(model => model.TypeSystemeConrne)</
                     th>
38                 <th>@Html.DisplayNameFor(model => model.Description)</th>
39                 <th>@Html.DisplayNameFor(model => model.ProcessusEmetteur)</
                     th>
40                 <th>@Html.DisplayNameFor(model => model.ProcessusConcerne)</
                     th>
41                 <th>@Html.DisplayNameFor(model => model.ActionCorrective)</th>
42                 <th>@Html.DisplayNameFor(model => model.ResponsableAc)</th>
43             </tr>
44         </tfoot>
45         <tbody>
46
47             @foreach (var item in Model)
48             {
49             <tr>
```

# Méthode Post:Create du contrôleur Non-conformité

```
121         }
122
123         // POST: NC/Create
124         [HttpPost]
125         public ActionResult Create(FormCollection collection)
126         {
127             try
128             {
129                 DeclarationAccident.Data.NC newNc = new Data.NC();
130                 DateTime dateNc = DateTime.MinValue;
131                 DateTime.TryParse(collection.Get("Date"), out dateNc);
132                 newNc.Date = dateNc;
133                 newNc.TypeSystemeID = int.Parse(collection.Get
                      ("TypeSystemeID") ?? "0");
134                 newNc.ProcessusEmetteur = collection.Get("ProcessusEmetteur");
135                 newNc.NumFNC = collection.Get("NumFNC");
136                 newNc.NomEmetteur = collection.Get("NomEmetteur");
137                 newNc.Description = collection.Get("Description");
138                 newNc.ProcessusConcerne = collection.Get("ProcessusConcerne");
139                 newNc.Nom = collection.Get("Nom");
140                 newNc.Fonction = collection.Get("Fonction");
141                 newNc.ActionCorrective = collection.Get("ActionCorrective");
142                 newNc.ResponsableAc = collection.Get("ResponsableAc");
143                 newNc.Delais = int.Parse(collection.Get("Delais") ?? "0");
144                 newNc.Efficacite = bool.Parse(collection.Get("Efficacite") ??
                      "False");
145                 newNc.Commentaire = collection.Get("Description");
146                 bool parse = DateTime.TryParse(collection.Get
                      ("DateCommentaire"), out dateNc);
147                 if (parse) { newNc.DateCommentaire = dateNc; };
148
149                 var derogation = new Data.Derogation();
150                 derogation.Description = collection.Get
                      ("Derogation.Description");
151                 derogation.NomDemandeur = collection.Get
                      ("Derogation.NomDemandeur");
152                 derogation.FonctionDemandeur = collection.Get
                      ("Derogation.FonctionDemandeur");
153                 derogation.Decision = bool.Parse( collection.Get
                      ("Derogation.Decision") ?? "False");
154                 derogation.NomDecideur = collection.Get
                      ("Derogation.NomDecideur");
155                 derogation.FonctionDecideur = collection.Get
                      ("Derogation.FonctionDecideur");
156                 derogation.Motif = collection.Get("Derogation.Motif");
157                 newNc.Derogation = derogation;
```

```
...MVC\DeclarationAccident.Web\Controllers\NCController.cs                4
158
159                 var json = collection.Get("causesIdentifiees");
160                 var recJson =
                      Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>
                      (json);
161                 newNc.Causes = DataHelper.CausesFromJson(recJson);
162
163                 var grpeJson =
                      Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>
                      (collection.Get("groupeAnalyse"));
164                 newNc.GroupeAnalyses = DataHelper.GroupesFromJson(grpeJson);
165
166                 repository.CreateNC(newNc);
167                 return RedirectToAction("Index");
168             }
169             catch (Exception ex)
170             {
171                 return View();
172             }
173         }
174
175         // GET: NC/Edit/5
176         public ActionResult Edit(int id)
177         {
```

# Vue « Liste des déclarations d'accidents »

...DeclarationAccident.Web\Views\Accident\Accidents.cshtml 1

```
@model IEnumerable<DeclarationAccident.Web.Models.Accident>

@{
    ViewBag.Title = "Accidents";
    Layout = "~/Views/Shared/_sbadmin.cshtml";
}


<h2>Accidents</h2>

<p>
    @*@Html.ActionLink("Create New", "Create")*@

    <a href="/accident/Create" class="btn btn-primary btn-icon-split">
        <span class="icon">
            <i class="fas fa-plus"></i>
        </span>
        <span class="text">Créer nouveau</span>
    </a>
</p>


<div class="table-responsive">
    <table class="table table-bordered" id="listAccidents" width="100%"
      cellspacing="0">
        <thead>
            <tr>
                <th></th>
                <th> @Html.DisplayNameFor(model => model.Date)</th>
                <th>@Html.DisplayNameFor(model => model.TypeAccident)</th>
                <th>@Html.DisplayNameFor(model => model.Nom)</th>
                <th>@Html.DisplayNameFor(model => model.Fonction)</th>
                <th>@Html.DisplayNameFor(model => model.Departement)</th>
                <th>@Html.DisplayNameFor(model => model.Description)</th>
            </tr>
        </thead>
        <tfoot>
            <tr>
                <th></th>
                <th> @Html.DisplayNameFor(model => model.Date)</th>
                <th>@Html.DisplayNameFor(model => model.TypeAccident)</th>
                <th>@Html.DisplayNameFor(model => model.Nom)</th>
                <th>@Html.DisplayNameFor(model => model.Fonction)</th>
                <th>@Html.DisplayNameFor(model => model.Departement)</th>
                <th>@Html.DisplayNameFor(model => model.Description)</th>
            </tr>
        </tfoot>
        <tbody>
            @foreach (var item in Model)
            {
            <tr>
                <td>
                    <div>
                        <button type="button" class="btn btn-success btn-circle
                     btn-sm" data-toggle="dropdown" aria-haspopup="true" aria-
                    expanded="false">
```

# Résumé

Le Système Management Intégré s'inscrit dans une démarche qualité et sa vocation est l'amélioration continue de la performance globale des entreprises. Sa gestion demeure une activité très complexe due principalement à la difficulté d'organisation et les lenteurs de traitement des tâches. Notre travail consiste à concevoir et à implémenter une application web qui permet la gestion du SMI (gestion des déclarations d'accidents/incidents et non-conformités) du service Hygiène et Sécurité Environnement de l'entreprise BMT. L'objectif est de permettre aux utilisateurs une gestion régulière et automatique des tâches qui leur sont confiées ainsi qu'une facilité et une rapidité de traitement de l'information. Pour ce faire, nous avons opté pour UP et UML pour la conception et ASP.NET MVC pour la réalisation.



**Mots-clés :** SMI, UP, UML, C#, ASP .NET MVC, ORM EntityFramework et Identity EF, Bootstrap, KnockoutJS.



# Abstract

The Integrated Management System is part of quality approach and its vocation is the continuous improvement of the overall performance of companies. Its management remains a very complex activity due mainly to the difficulty of organization and the slowness of processing tasks. Our job is therefore to design and implement a Web Application allowing the management of accident, incident and non-conformances declarations of the integrated management system of the Health and Safety Environment department of the BMT company, in order to emphasize quality, the environment and occupational safety. As sequence, ensure the good process of these projects. To do this, we opted for UML and UP for the design and ASP.NET MVC technology for the realization.



**Keywords** : SMI, UP, UML, C#, ASP .NET MVC, ORM EntityFramework & Identity EF, Bootstrap, KnockoutJS.