SOMMAIRE

INTRODUCTION GÉNÉRALE

INTRODUCTION GENERAL

Durant ces trente dernières années , les avancées technologiques du traitement de l'information ont eu des conséquences capitales sur le rôle de l'outil informatique. Si les premières applications ont permis d'automatiser les activités opérationnelles des organisations (gestion de production, gestion commerciale et financière, ressources humaines…), aujourd'hui les systèmes d'information prennent en charge des niveaux de gestion de plus en plus stratégiques.

Les ERP (Enterprise Resource Planning) ou PGI (Progiciels de Gestion Intégrés), ont connu leur essor en profitant de l'évolution nécessaire des systèmes d'information. En effet, il était séduisant de remplacer tous les logiciels de gestion de l'entreprise par un intégré offrant « l'état de l'art » plutôt que d'engager des corrections des programmes existants plus ou moins anciens.

Notre Projet de Fin d'Etudes porte sur L'OpenERP, un PGI open-source extrêmement modulaire. Notre travail consiste à réaliser une « gestion de la flotte automobile » que nous avons baptisé «DZ Fleet ».

Ce sujet nous été proposé en collaboration avec l'entreprise SOGESI. C'est une société située à Tlemcen en Algérie spécialisée dans les services informatiques et d'ingénierie de proximité, la gestion des applicatifs et des infrastructures ainsi que le conseil en technologies. Elle conçoit, développe, teste, maintient et optimise les systèmes d'information des entreprises et des organismes publics. En Algérie, le savoir-faire de SOGESI multi spécialiste est unique avec son profil de multi-spécialiste.

L'idée de notre projet est de commencer par effectuer d'abord une analyse du besoin, afin de bien souligner les différentes fonctionnalités requises, ensuite à chercher parmi ces fonctionnalités celles qui sont déjà préposées par OpenERP. Ceci, afin d'adapter celles qui existent au contexte de l'organisation algérienne et surtout d'entamer le développement des fonctionnalités restantes qui n'existaient pas encore.

Notre mémoire est subdivisé en trois chapitres. Le premier présente en générale les notions de base concernant les ERP et la gestion de la flotte automobile. Le second s'altère autour de l'analyse et la conception de DZ Fleet projet. Le dernier explique l'étape du développement de DZ Fleet. En fin, ce mémoire s'achève par une conclusion sur l'apport du travail réalisé et des perspectives futurs.

CHAPITRE I

LES ERP ET LA GESTION DE LA FLOTTE AUTOMOBILE

I. Introduction

L'informatique occupe évidemment une grande place dans le domaine de transport et en particulier, la gestion de la flotte automobiles. En effet, cette gestion est une tâche capitale qui présente un nombre important de sous tâches réalisés manuellement. Elle consiste généralement à répartir les véhicules entre les différents chauffeurs pour leurs missions, l'entretien des véhicules la réparation des automobiles et la gestion du personnel du parc automobile, la gestion des documents, à coordonner également les déplacements des chauffeurs pour leurs missions à l'intérieur du pays, etc. Donc le gestionnaire de parc automobile joue un rôle important dans le fonctionnement des parcs automobiles des entreprises et des sociétés privés et étatiques. En effet, ces établissements cherchent toujours à assurer une bonne gestion de leurs parcs automobile en rendant cette pénible tâche informatisée.

Néanmoins, pendant ces dernières années l'ERP (Enterprise Resource Planning) est imposé comme un moyen d'optimisation du processus de gestion. C'est ce que nous présenterons avec détails dans le présent chapitre.

II. ERP (Enterprise Resource Planning)

Un PGI progiciel de gestion intégré (en anglais Enterprise Resource Planning ou ERP) est un « logiciel qui permet de gérer l'ensemble des processus d'une entreprise en intégrant l'ensemble des fonctions de cette dernière comme la gestion des ressources humaines, la gestion comptable et financière, l'aide à la décision, mais aussi la vente, la distribution, l'approvisionnement, le commerce électronique »(Définition du grand dictionnaire terminologique de l'Office québécois de la langue française (OLF)). [1]

Le principe fondateur d'un ERP est de construire une application paie, (comptabilité, gestion de stocks) de manière modulaire tout en partageant une base de données unifiée. Cela crée une différence importante avec la situation préexistante car les différentes fonctions de l'entreprise étaient gérées par une multitude d'applications dédiées souvent hétérogènes. Ainsi, les Achats, la Comptabilité, la Gestion des Stocks, les Ressources Humaines, la Gestion Commerciale,... sont maintenant totalement interconnectés. Avec l'arrivée de l'ERP, les données sont désormais standardisées et partagées entre les différents modules, ce qui élimine les saisies multiples et évite l'ambiguïté des données

multiples de même nature (ex : «Clermont-Fd », « Clermont Ferrand », « Clermont-Ferrand », ...). [1]

Ceci permet un accroissement considérable de la fiabilité des informations puisque la source des données est unique, d'où une réduction des délais et des coûts de traitements.

L'autre principe qui caractérise un ERP est l'usage systématique de ce qu'on appelle un moteur de Workflow, et qui permet, lorsqu'une donnée est entrée dans le système d'information, de la propager dans tous les modules du système qui en ont besoin, selon une programmation prédéfinie.

Ainsi, on peut parler d'ERP lorsqu'on est en présence d'un système d'information composé de plusieurs applications partageant une seule et même base de données, par le biais d'un système automatisé prédéfini éventuellement paramétrable (un moteur de workflow).

Plus qu'un simple logiciel, un ERP est un véritable projet demandant une intégration totale d'un outil logiciel au sein d'une organisation et d'une structure spécifique, et donc des coûts importants d'ingénierie. [1]

D'autre part, sa mise en place dans l'entreprise entraîne des modifications importantes des habitudes de travail d'une grande partie des employés.

II.1 Les avantages et les inconvénients

a. Avantages

Comparés à des applications sur mesure, les ERP = PGI présentent plusieurs avantages [1] :

– optimisation des processus de gestion (flux économiques et financiers) ;
– cohérence et homogénéité des informations ;
– intégrité et unicité du système d'information ;
– partage du même système d'information facilitant la communication interne et externe ;
– globalisation de la formation (même logique, même ergonomie) ;
– maîtrise des coûts et des délais de mise en œuvre et de déploiement.

Il est important de remarquer que la mise en place d'un ERP dans une entreprise est souvent le déclencheur d'une réorganisation et rationalisation de l'ensemble des tâches et processus de l'entreprise.

b. Inconvénients

Les ERP = PGI ne sont cependant pas exempts d'inconvénients [1] :

– coût élevé ;

– périmètre fonctionnel souvent plus large que les besoins de l'organisation ou de l'entreprise (le progiciel est parfois sous-utilisé) ;

– lourdeur et rigidité de mise en œuvre ;

– difficultés d'appropriation par le personnel de l'entreprise ;

– nécessité d'une bonne connaissance des processus de l'entreprise (par exemple, une commande d'achat et une commande de vente nécessitent deux processus différents : il est important de savoir pourquoi, de savoir décrire les points communs et les différences entre ces deux processus de façon à bien les paramétrer) ;

– nécessité d'adapter parfois certains processus de l'organisation ou de l'entreprise au progiciel ;

– nécessité d'une maintenance continue.

II.2 ERP libres

Le secteur des ERP a depuis quelques années déjà subi un petit bouleversement : l'arrivée de logiciels libres (OFBiz Tiny ERP, ERP5, Compiere, ODOO (ancien open ERP) ...) sur des terres où règnent en maîtres les logiciels propriétaires : SAP, BAAN, Oracle, PostgreSQL…… [1]

Le premier avantage des ERP Libre sur leurs alter-ego propriétaires est bien sûr l'absence de coût de licence ; coût qui peut souvent apparaître comme prohibitif pour les PME.

Un autre atout important est la possibilité d'adapter et de faire évoluer soi-même le progiciel sans dépendre du bon vouloir de la société éditrice. En outre, le logiciel libre mobilise souvent des communautés, qui le font évoluer au gré des nouveaux besoins, et qui peuvent répondre rapidement à des demandes précises. [1]

De plus comme tout logiciel libre, les ERP libre donne la garantie de travailler sur des standards ouverts et donc interopérables, avantages stratégiques pour beaucoup d'entreprises. Et comme notre projet se base sur l'ERP libre ODOO on va le détaillé ultérieurement.

II.3 ODOO

ODOO est un progiciel de gestion intégrée publié sous la licence AGPL à partir de la version 6. Toutes ses fonctionnalités sont développées en Python alors que les interfaces sont développées en XML. Il possède deux types de client : un client lourd GTK qui doit être installé dans chaque poste utilisateur et un client Web qui utilise un navigateur pour se connecter au serveur ODOO. [Open, 2012]

a. L'architecture technique d'ODOO

L'architecture d'ODOO se base sur l'architecture client/serveur comme présenté sur la figure I.1

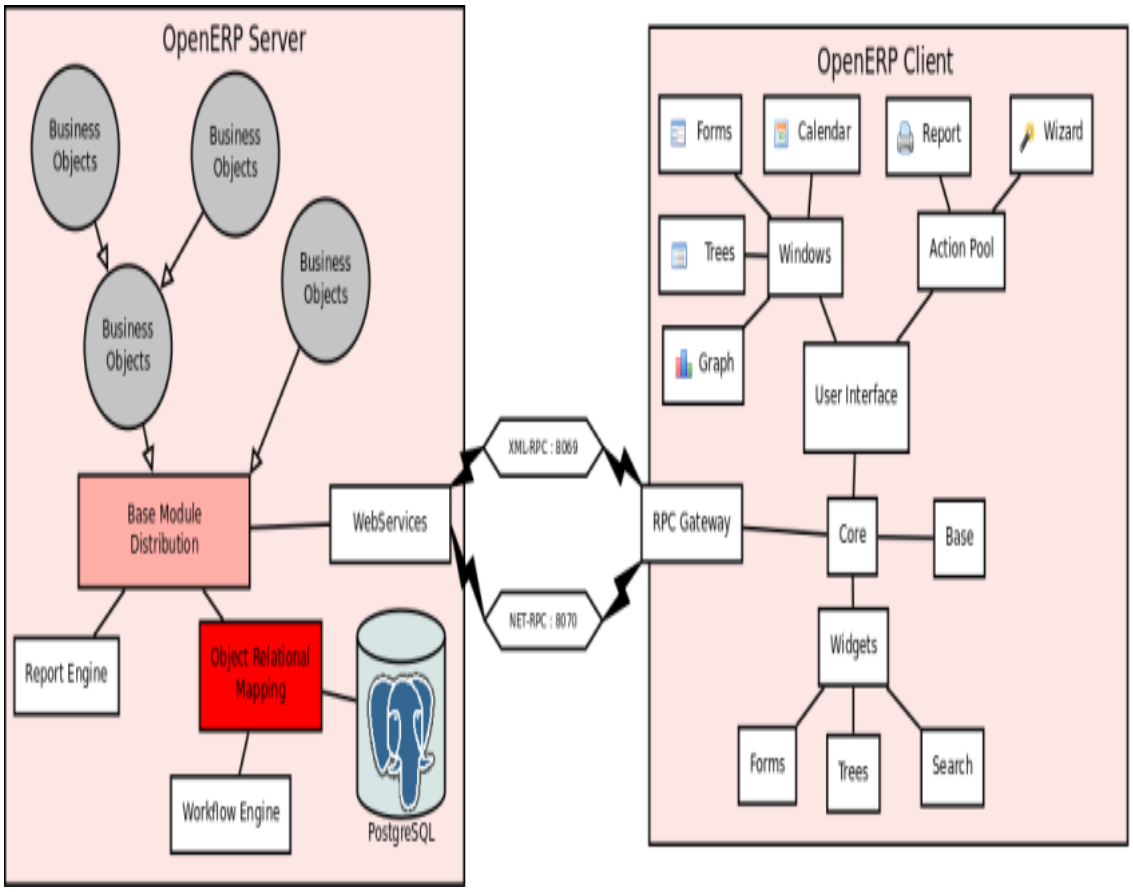


Figure I.1 : l'architecture technique d'ODOO [Open, 2012]

Le client ODOO peut visualiser des vues sous format de liste, de graphe ou bien d'un calendrier. Comme, il peut effectuer des actions à savoir l'impression et consultation des rapports. La communication entre le client et le serveur est assurée par le protocole XML/RPC.

La logique ODOO se centralise du côté serveur car le rôle du client se limite à effectuer des requêtes et à recevoir des données envoyé par le serveur. En effet, le serveur ODOO suit l'approche orienté objet en utilisant l'architecture Modèle-Vue-Contrôleur. Les modèles sont représentés par des objets de la base de données PostgreSQL, les vues, quant à elles, sont des interfaces développées en XML et les contrôleurs sont des fichiers développés en Python qui décrivent les fonctionnalités métiers. [Open, 2012]

b. Les composantes d'un serveur ODOO

Le serveur ODOO est constitué de plusieurs composantes principales comme montre la figure I.2

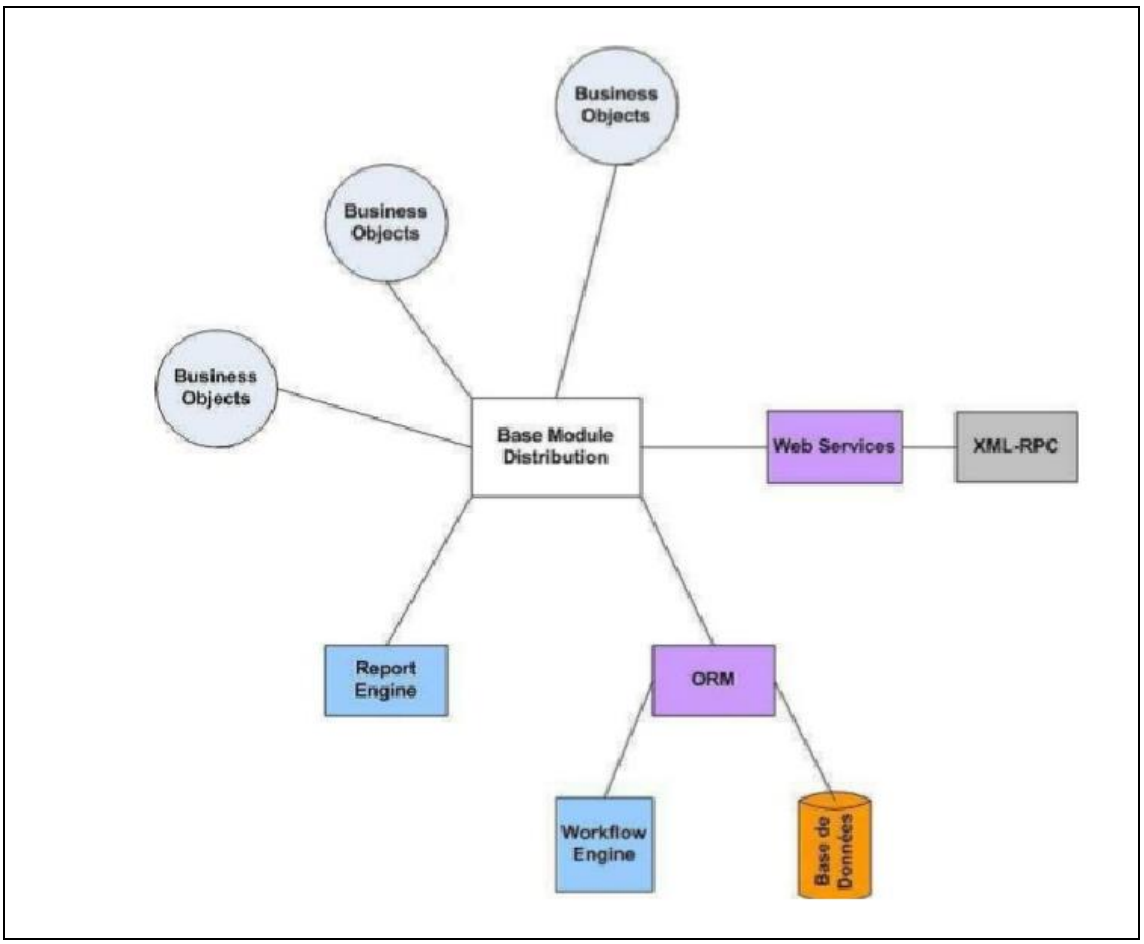


Figure I.2 : les composantes d'un serveur ODOO [Open, 2012]

Chaque composante possède un rôle bien défini :

- **Base Module Distribution :** ce composant assure la communication distribuée au sein du serveur OpenERP. En effet, une fois une requête arrive au serveur, ce mécanisme fait appel aux différents objets reliés à cette requête et rassemble toutes les informations afin de construire la réponse.
- **Mapping Objet/Relationnel ou ORM :** constitue le pont entre les objets métier et la base de données relationnelle. En effet, il crée une base de données objet qui peut être utilisée dans un langage de programmation orienté objet sans faire appel à la base de données relationnelle. Cette couche permet d'assurer la consistance et la sécurité (gestion de droits par utilisateur et par groupe) des données.
- **Web Services :** permet la connexion du serveur OpenERP avec un client web via un navigateur web, ce qui n'est pas nécessaire pour un client GTK.
- **Moteur de workflow :** ce moteur assure le suivi des flux métiers et l'enchainement des tâches au sein d'un processus métier.
- **Moteur de reporting :** prend en charge la génération des rapports dans OpenERP en faisant appel à des outils comme Open Report qui génère des rapports en PDF. Φ Business Object : représente les objets métiers d'OpenERP construites par l'ORM, ils peuvent être représentés par différentes vues.
- **Serveur de base de données PostgreSQL :** présenté par la suite, il assure le stockage et la persistance des données d'OpenERP. [Open, 2012]

c. Architecture modulaire d'ODOO

Un module ODOO est la définition, dans le «Framework» ODOO, d'une gestion informatisée d'un domaine. Cette architecture n'est pas propre à open ERP. Elle est en fait partagée par tous les ERP. Il s'agit de la faculté de construire des applications informatiques de manière modulaire (modules indépendants entre eux) tout en partageant une base de données unique. Ceci apporte une importance significative puisque les données sont maintenant standardisées et partagées. Ce qui élimine les saisies multiples et évite l'ambiguïté des données de même nature. [2]

L'architecture modulaire d'ODOO lui permet de couvrir plusieurs domaines illustrés dans la figure I.3.

Figure I.3 : Architecture modulaire de ODOO [2]

d. Modèle Vue Contrôleur (MVC)

Dans open ERP, on peut appliquer cette sémantique de Model View Controller avec :

- **Le serveur de données PostgreSQL : (les modèles)**

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO) qui vise à stocker et à partager les données. Il offre, en plus des fonctionnalités d'un SGBDR, la possibilité de stockage persistant des objets. En effet, PostgreSQL est doté des fonctions nécessaires pour la logique orientée objets comme la notion d'héritage, d'encapsulation et de polymorphisme. [Postgre, 2012]

Pour la partie administration, PgAdmin III est la plateforme de PostgreSQL la plus populaire et la plus riche en fonctions. PgAdmin III est conçu pour répondre aux besoins de tous les utilisateurs depuis l'écriture de simples requêtes SQL jusqu'au développement de bases de données complexes. Elle permet l'administration de bases de données (création de tables, d'index, des utilisateurs, gestion des groupes, des droits, des schémas, des sauvegardes et des restaurations). Il offre en plus, un éditeur de SQL et il permet la visualisation les dépendances entre les différents objets. [Wiki, 2012]

- **Python (le contrôleur)**

ODOO a été développé en Python et utilise son propre Framework : le Framework OpenObject. Il a été développé de façon modulaire. Autrement dit, il est possible d'ajouter ou de retirer des modules pour changer le fonctionnement du programme : ajout, modification et suppression des fonctionnalités. La personnalisation d'une instance d'ODOO peut se faire de deux manières [3] :

- ✓ Avec l'interface d'OpenERP, qui est plus de la configuration que du développement. Les modifications resteront sur la base de données, et le déploiement du module sur une autre instance nécessitera l'exportation dudit base de données.
- ✓ En créant des modules pythons dédiés, qui est de la programmation pure, basée sur l'héritage d'objets. Le module ainsi crée sera un dossier contenant plusieurs fichiers pythons et XML. Pour le déploiement du module sur une autre instance, il suffirait de copier ce dossier dans le répertoire correspondant.

D'où l'utilisation du langage de programmation python. C'est un langage multi-paradigme qui favorise la programmation impérative structurée, et orientée objet. C'est un langage beaucoup apprécié par les pédagogues car il propose une syntaxe simple à utiliser, qui permet une initiation aisée des concepts de base de la programmation. [3]

- **XML : (les vues)**

XML (eXtensible Markup Language), aussi appelé langage à balisage étendu, est un langage simple et puissant de description et d'échange de documents structurés. Il est souvent surnommé « Langage HTML amélioré » puisque celui-ci permet de définir de nouvelles balises, contrairement à HTML, qui a des balises figés. Sa plus grande force réside dans le fait qu'il soit capable de décrire n'importe quel domaine de données grâce à son extensibilité. [3]

Comme python, XML fait partie des indispensables dans la réalisation d'un module ODOO. En effet, les vues par lesquelles sont représentés les différents objets sont écrites en XML. Nous y trouvons la description détaillée de l'affichage des arbres, des formulaires, des graphes, du calendrier, etc. Toutefois, son utilisation ne s'arrête pas là. Nous avons recours à XML lors de la création des menus, des actions, des profils utilisateur, des workflows, des séquences, etc. [3]

III. Quelques projets existants sur la gestion de la flotte automobile

III.1 OpenERP - Fleet vehicle (version standard)

Avec ce module [4], ODOO aide à gérer tous véhicules, les contrats associés à ceux véhicule ainsi que les services, le journal de carburant les entrées, les coûts et de nombreuses autres fonctionnalités nécessaires à la gestion de la flotte des véhicules comme :

* Ajouter des véhicules à la flotte.

* Gérer les contrats pour les véhicules.

* Rappel lorsqu'un contrat atteint sa date d'expiration.

*Ajouter les services, l'entrée du journal de carburant, les valeurs du compteur kilométrique pour tous les véhicules.

* Afficher tous les coûts associés à un véhicule ou à un type de service.

* Graphique d'analyse des coûts.

La figure I.4 présente l'interface du module

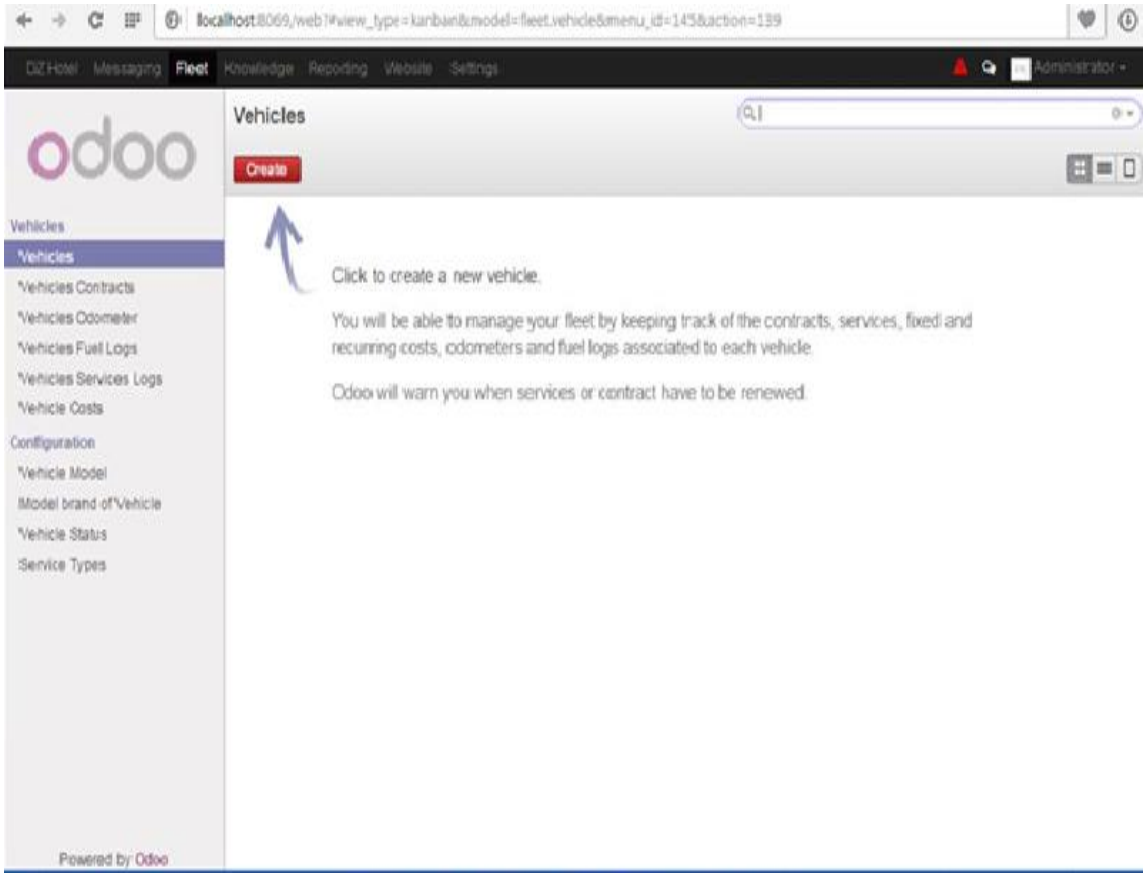


Figure I.4 : l'interface du module de la gestion de parc véhicule (version Standard) [4]

III.2 OpenERP - Gestion de parc Véhicules (version tunisienne)

Ce module est créé par le développeur tunisien Taieb kristou sous ODOO [5]. Il a le même fonctionnement que le module sous la version algérienne, il gère de plus les ordres de missions et le suivi des interventions sur les accompagnateurs.

Présente les mêmes fonctionnalités de fleet vehicle mais de plus, il permet de :

- Créer un ordre de mission.
- Afficher les interventions des accompagnateurs.
- Afficher les détails de déplacement pour chaque mission.

La figure I.5 présente l'interface principale du module

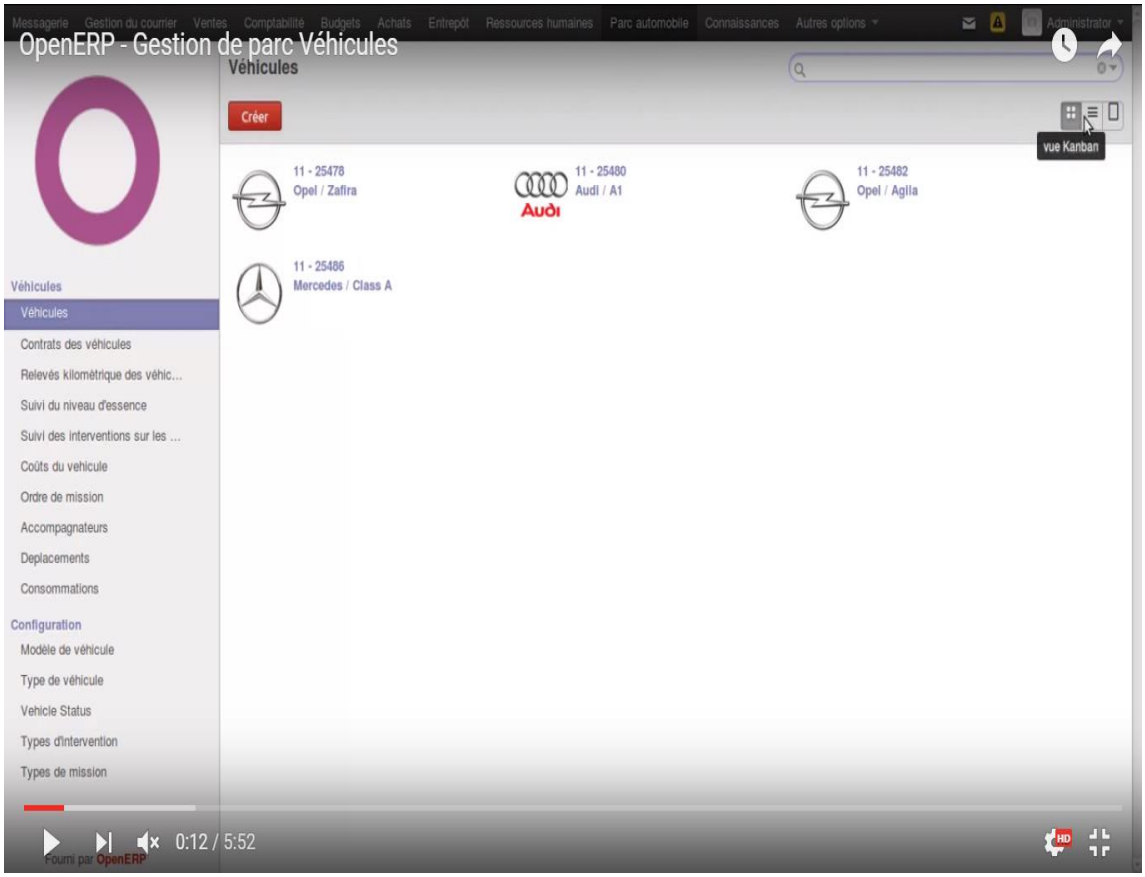


Figure I.5 : l'interface du module de la gestion de parc véhicule (version tunisienne) [5]

III.3 DIGIPARC - gestion de parc automobile

C'est un autre progiciel de gestion intégré [6] permet au dirigeant d'entreprise de gérer la gestion des véhicules, la gestion des utilisateurs et de nombreuses fonctionnalités nécessaire à la gestion de la flotte automobile comme :

- Afficher un Tableau de bord pour visualiser le parc en un coup d'œil.
- Des indicateurs révélateurs de l'évolution du parc
- Des alertes pour veiller surla flotte
- Tout savoir sur chaque véhicule

La figure I.6 ci-dessous présente l'interface du module DIGIPARC

Figure I.6 : interface du module DIGIPARC [6]

IV. Synthèse

Après l'étude des projets de gestion de la flotte automobile, cités précédemment, nous avons pu dégager les points forts et les faiblesses de chacun d'eux que nous résumons dans le tableau N°1 ci-dessous. Ceci, dans le but d'essayer de reprendre ces points forts et d'éviter les points faibles lors de la réalisation de notre application DZ Fleet.

Les modules existants	Les points forts	Les points faibles
Fleet vehicle (version standard)	✓ l'ajout des véhicules à la flotte. ✓ l'Ajout des services, l'entrée du journal de carburant, les valeurs du compteur kilométrique pour tous les véhicules.	o Absence de la notion des missions, réservation, prévision et bon de carburant
Gestion de parc Véhicules (version tunisienne)	✓ Créer un ordre de mission. ✓ Afficher les détails de déplacement pour chaque mission.	o Absence de la notion de réservation et prévention et de bon de carburant
DIGIPARC gestion de parc automobile	✓ Des alertes pour veiller sur la flotte	o Absence de la notion des missions, réservation, prévision et bon de carburant

Tableau N°1 : les points forts et faiblesse des projets existants

V. Conclusion

Tout au long de ce chapitre, nous nous sommes attachés à définir les principaux éléments alimentant notre travail. A savoir : les systèmes d'information, les ERP et l'ERP libre ODOO ainsi qu'une étude de synthèse sur quelques projets existants de gestion de la flotte automobile dans les ERP.

Nous pouvons maintenant, poursuivre notre travail en passant à l'étape de conception de notre application DZ Fleet.

CHAPITRE II

LA CONCEPTION DU DZ FLEET

I. Introduction

Nous abordons dans ce chapitre une analyse du besoin, une brève présentation du langage de conception UML qu'on a utilisé pour notre conception. Apres nous présentons les diagrammes de conception adoptés lors de la phase d'élaboration.

II. Analyse des besoins

A partir du cahier de charge transmit par SOGESI, nous avons dégagés les principales fonctionnalités de DZ Fleet qui sont : La gestion des véhicules, la gestion des employées, la gestion des missions, la gestion de maintenance des véhicules, la gestion des bons de carburant, la gestion de document de bord des véhicules, la gestion d'inventaire des véhicules, la gestion de réservation des véhicules, la gestion de prévision des interventions.

Seul l'administrateur (gestionnaire du parc) peut accéder à l'application par un nom d'utilisateur et mot de passe.

De plus DZ Fleet permet à l'administrateur de :

- Visualiser le parc automobile
- D'imprimer les ordres de mission

Apres avoir définit les principales fonctionnalités qui interviennent dans la gestion du parc automobiles, il est nécessaire d'établir une analyse détaillée pour chaque fonction afin de mieux cerner les informations qu'elle manipule. Ceci pour préparer la conception.

II.1 Gestion des véhicules

La solution proposée doit permettre d'identifier l'ensemble des véhicules existants et de gérer les informations générales et techniques correspondantes comme : l'immatriculation, le modèle, le numéro de châssis, dernier relevé Kilométrage, date d'acquisition, nombre de places, nombre de portes, la couleur, l'état, transmission, type de Carburant, nombre de chevaux, puissance

Seulement l'utilisateur qui a le droit de la consultation complète de ce module, il peut :

- Mettre à jour un véhicule (supprimer, ajouter, modifier)
- Afficher la liste des véhicules
- Chercher des véhicules avec plusieurs critères de recherche

II.2 Gestion des missions

L'application est censée de manipuler les informations suivante :

- Numéro de la mission
- Véhicule disponible
- Chauffeur disponible
- Accompagnateurs disponibles
- Objectif de la mission
- Nombre de bon de carburant
- Date et heure de début et de fin
- Kilométrage de départ et de retour
- Lieu de départ et d'arrivé
- Heures et kilométrages parcourus durant la mission

Dans cette gestion l'utilisateur peut :

- ✓ Mettre à jour une mission
- ✓ Afficher la liste des missions
- ✓ Chercher les missions avec selon leur état si elle est ouverte, en cours ou bien terminer

II.3 Gestion des bons de carburant

Cette gestion porte sur :

- la quantité des bons de carburant
- le prix unitaire

Dans ce cas l'utilisateur peut saisir la quantité des bons de carburant et le prix unitaire.

II.4 Gestion des employés

N'importe quelle entreprise doit avoir l'affiliation de ses employés qui contient :le nom, prénom, poste occupé, l'adresse électronique, le type de pièce d'identité, le numéro de pièce d'identité, le numéro de téléphone, genre, l'état civil, la date de naissance, lieu de naissance, nationalité , la rue, la ville, le nom du pays, le code postal et si l'employé est un chauffeur il doit avoir un numéro de permis de conduire, la catégorie de permis, la date d'expiration du permis.

Dans cette gestion l'utilisateur peut :

- ✓ Mettre à jour les informations des employés
- ✓ Afficher la liste des employés

- ✓ Chercher des employés selon des plusieurs critères

II.5 Gestion de la maintenance

Chaque véhicule doit avoir un entretien, il se fait en saisissant les champs : Nom du véhicule, kilométrage, numéro de facture, date d'entretien, type d'intervention, cout d'intervention.

DZ Flette doit permettre à l'administrateur de gérer les interventions des véhicules et le cout d'intervention.

II.6 Gestion des documents de bord des véhicules

DZ Fleet permet l'administrateur de gérer les documents de bord de chaque véhicule c'est-à-dire de faire un suivi de type de document pour n'importe quel véhicule existant dans l'entreprise en saisissant les informations suivantes :

- Nom de véhicule
- Type de document
- Date d'expiration
- Cout de type de document

II.7 Gestion d'inventaire des véhicules

Cette gestion permet d'afficher l'état des interventions de chaque véhicule, en premier les informations suivantes :

- Nom du Véhicule
- Date d'intervention
- Description de l'intervention
- Et surtout le cout d'intervention et le total

II.8 Gestion de réservation de véhicule

DZ Fleet permet de réserver un véhicule, avec son chauffeur pour une date précise en remplissant les champs : nom du véhicule, nom du chauffeur, nom d'accompagnateur, date début et de fin de la mission.

II.9 Gestion de prévision d'interventions

En termes de prévision DZ Fleet offre un mécanisme d'alerte pour avertir l'administrateur qu'un véhicule a besoin d'une maintenance. Ceci en indiquant : le nom du véhicule, la date de prévision d'intervention et la description de l'intervention.

III. Conception de DZ Fleet

Pour réaliser cette conception, nous avons choisi d'utiliser l'UML [réf] qui est un langage de modélisation qui permet de présenter, d'une manière graphique, les divers aspects d'une solution. Il définit plusieurs diagrammes afin d'assurer une bonne modélisation. Parmi les diagrammes de modélisation nous utilisons le diagramme de cas d'utilisation, diagramme de séquence et diagramme de classe.

III.1 Diagramme de cas d'utilisation

C'est un diagramme qui permet de représenter le comportement de ces composants, du point de vue de ses utilisateurs (les acteurs). [7]

a. Diagramme de cas d'utilisation relatif au gestionnaire du parc

Une fois sue l'administrateur s'authentifie, il aura l'autorisation de gérer les missions, les véhicules, la maintenance des véhicules etc. De plus il peut effectuer des recherches avec différents critères. Le digramme de la figure II.1 illustre cette idée.

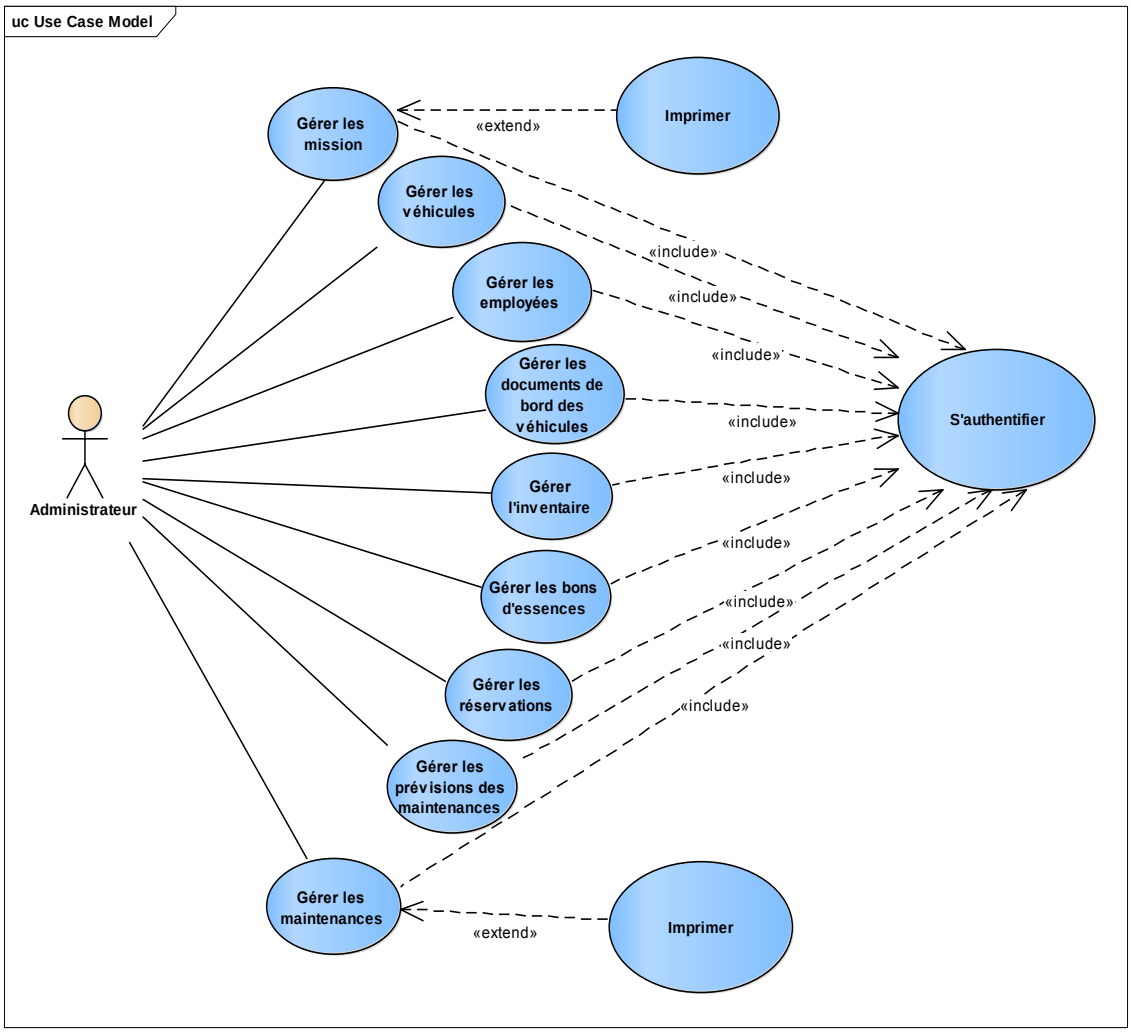


Figure II.1 : diagramme de cas d'utilisation relatif au gestionnaire du parc

b. Description du cas d'utilisation : Administration du parc :

Acteur : gestionnaire du parc

Description : ce cas d'utilisation montre ce que l'administrateur peut effectuer comme tache.

Préconditions :l'authentification de l'administrateur.

Poste-conditions : les fonctionnalités proposées par DZ Fleet deviennent accessibles

Quelque Cas possibles :

- Mettre à jour les missions.
- Mettre à jour les véhicules et les employés
- Gérer la maintenance des véhicules
- Gérer la réservation de véhicule avec son chauffeur
- Gérer les documents de bords d'un véhicule
- Gérer les bons de carburant
- Imprimer un ordre de mission
- Imprimer la maintenance des véhicules

III.2 Diagramme de sequence

C'est une présentation graphique des interactions entre les acteurs et le système.[7]

a. Diagramme de séquence pour un Scenario de création de mission

Le digramme, exposé sur la figureII.2, décrit les scénarios possibles lors d'une opération de création d'une mission. Pour créer une nouvelle mission il faut remplir la liste des champs.

Le système Fleet_mission va vérifier tous les ressources humaines ou matériel pour déterminer celles qui sont disponible ou pas et vérifier de plus l'existence des champs vides. Le résultat retenu est soit le sucée de l'opération de création soit l'apparition d'un message indiquant l'obligation de saisie des champs obligatoires.

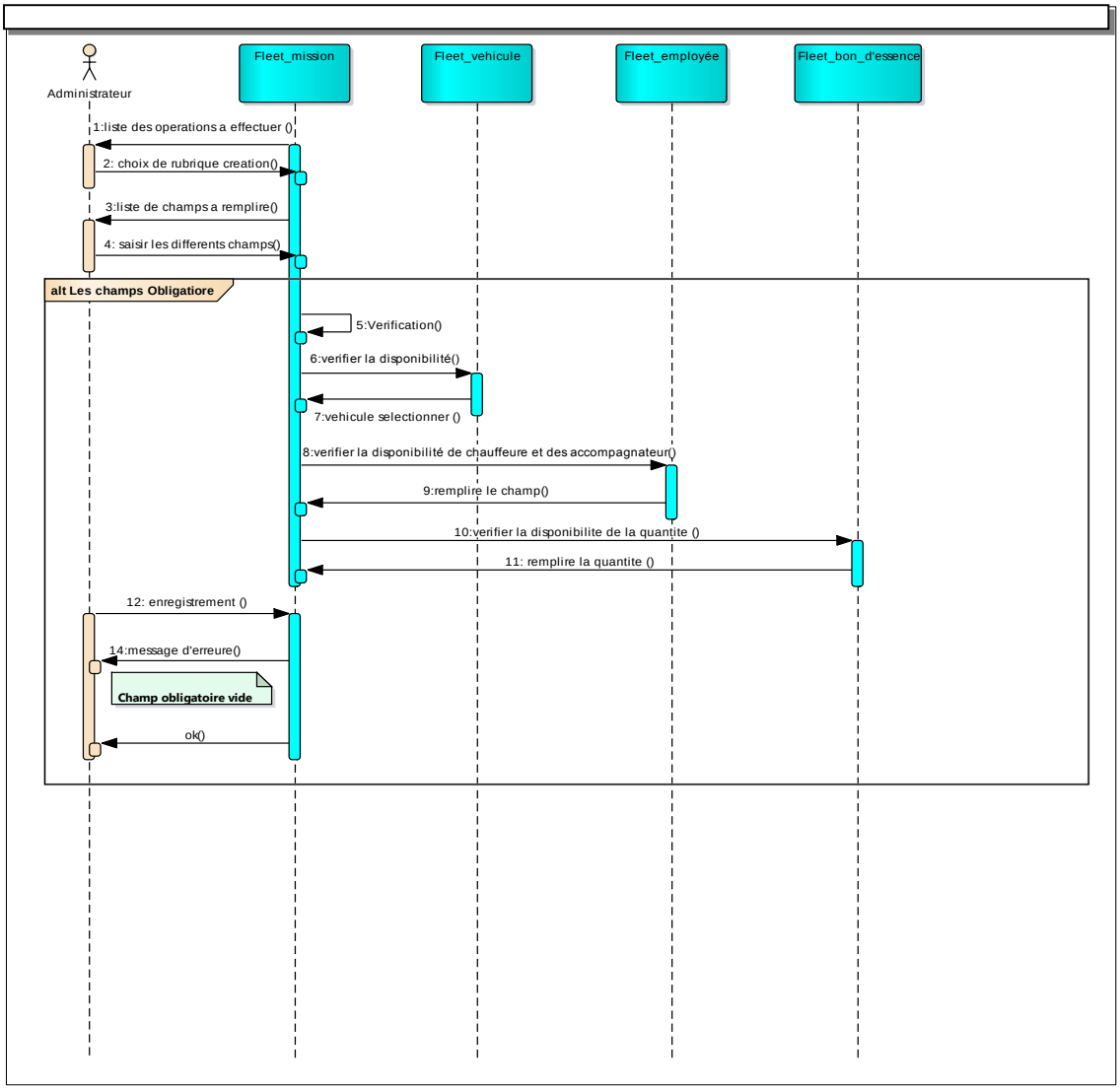


Figure II.2 : diagramme de séquence scénario de création d'une mission

b. Diagramme de séquence Scenario modification d'une mission

La figure II.3, décrit les scénarios possibles lors d'une opération de modification d'une Mission. En effet, si l'administrateur choisit la rubrique de la modification des informations d'une mission. Le système à son tour cherche les anciennes informations et affiche une interface contenant des champs remplis avec ces derniers. L'administrateur met à jour ces informations puis il valide. Le système Fleet_mission vérifie la modification de l'administrateur. Le résultat retenu est soit la validation de l'opération ou bien l'apparition d'un message indiquant l'obligation de saisie des champs obligatoires ou bien pas de ressources disponibles.

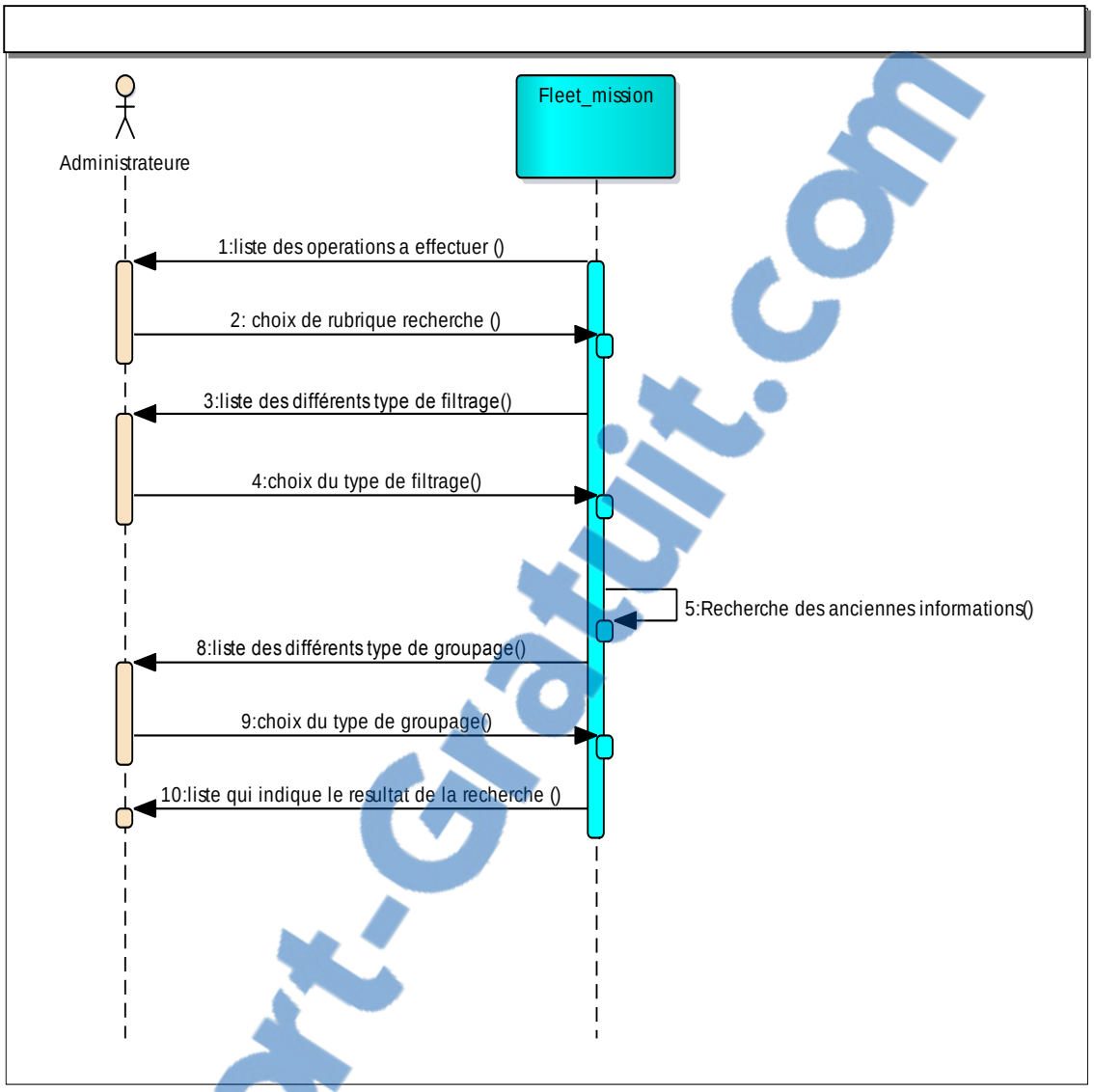


Figure II.3 : diagramme de séquence scénario de modification d'une mission

c. Diagramme de séquence Scenario *de recherche d'une mission*

La figure II.4, décrit les scénarios possibles lors d'une opération de recherche d'une mission, En effet si l'administrateur est dans sa session et il choisit la rubrique de recherche dans l'objet mission. Le systèmeFleet_mission à son tour affiche des critères de recherche par filtrage. Ensuite le groupage est fait de la même manière que le filtrage, c'est-à-dire l'administrateur fait son choix selon les informations qui lui intéresse puis il valide. A ce niveau, l'application va solliciter le système de la gestion de la base de données afin d'obtenir un résultat.

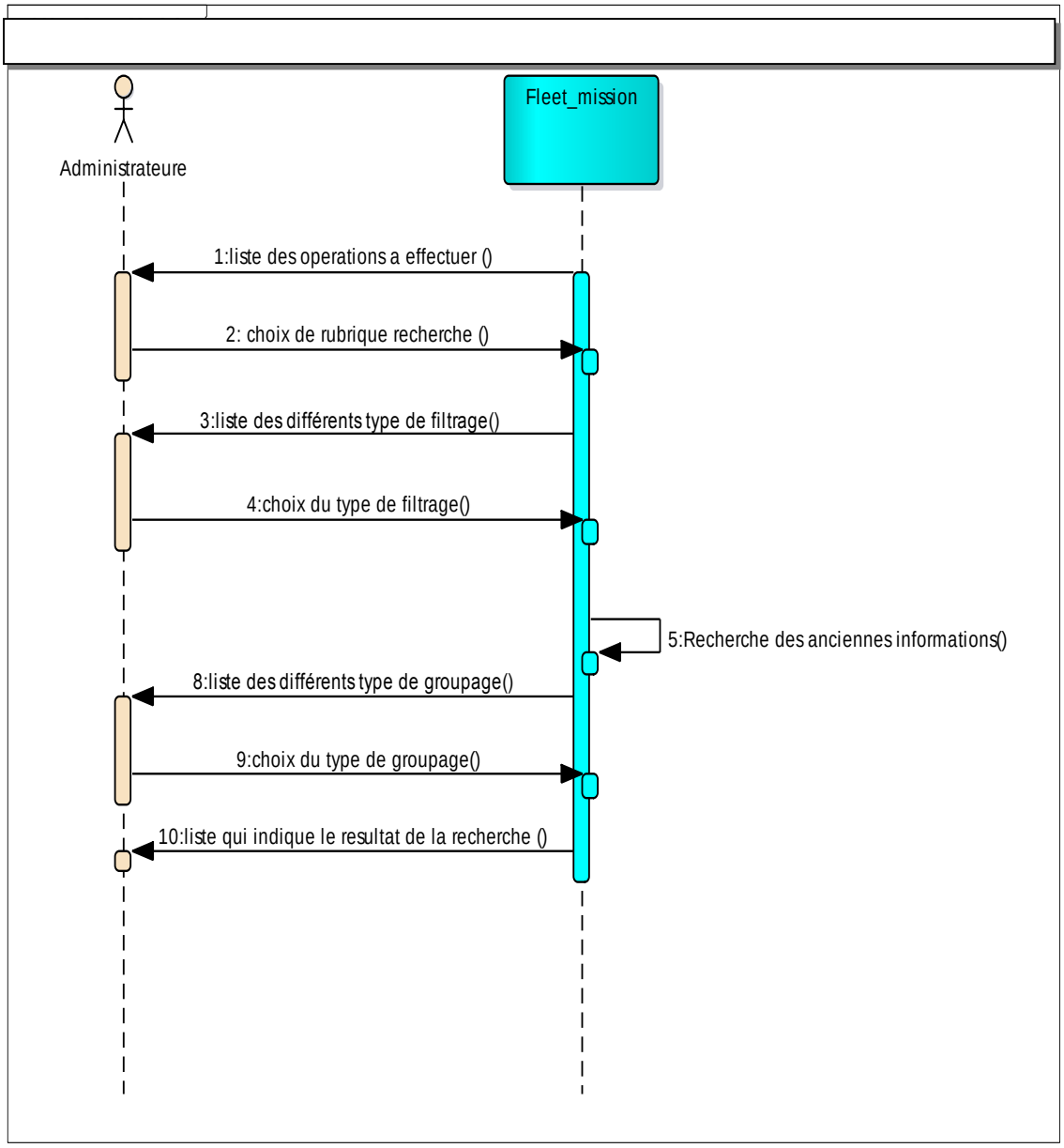


Figure II.4 : diagramme de séquence scénario de recherche d'une mission

d. Diagramme de séquence scenario consultation de la table inventaire

Le digramme, exposé sur la figureII.5, décrit le scénario de l'inventaire .En effet si l'administrateur choisit de consulter la table inventaire le système Fleet_invetaire va insérer dans cette dernière le cout de chaque dépense pour chaque véhicule. L'affichage de la table sera au choix de l'administrateur c'est à dire par groupage.

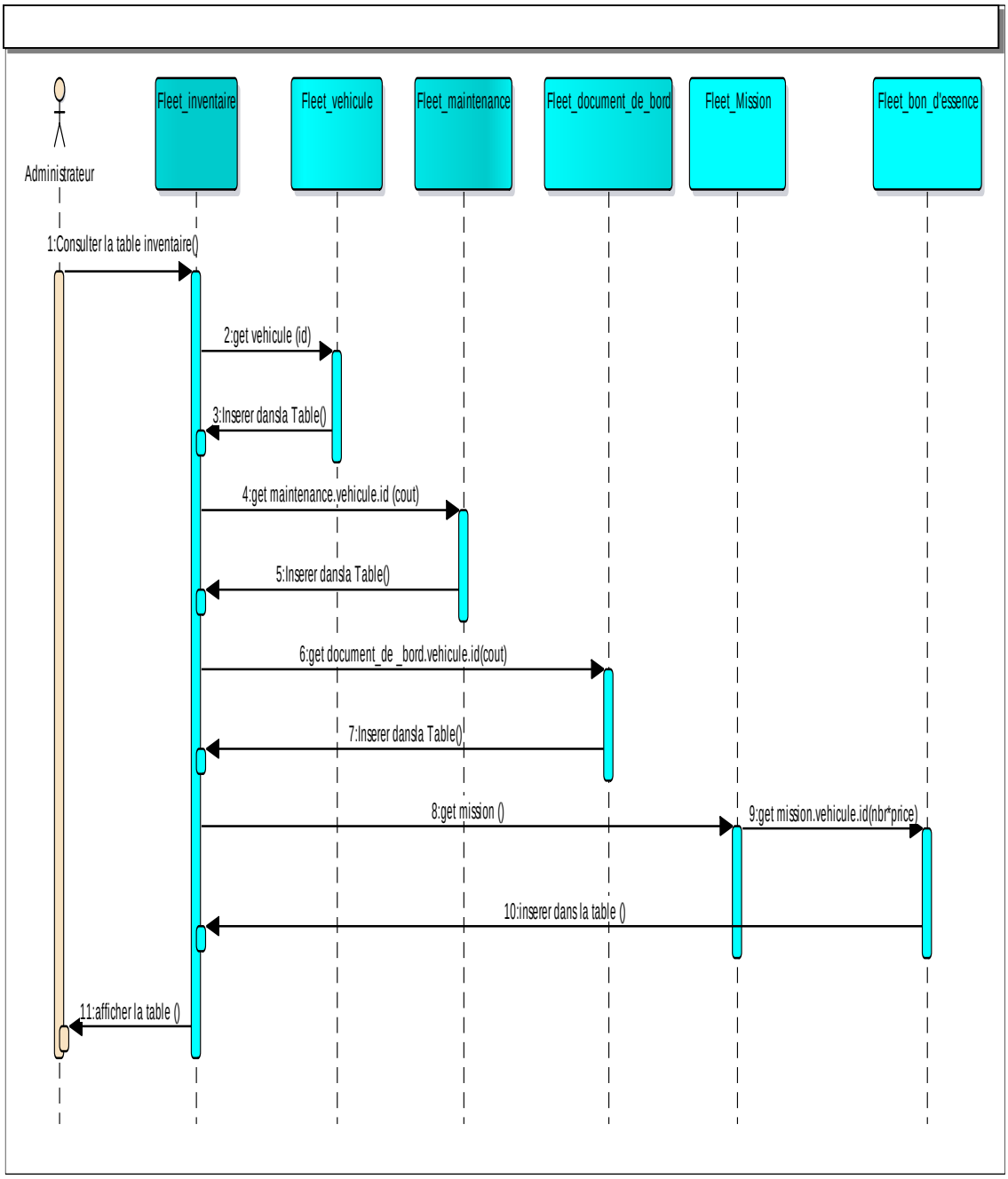


Figure II.5: diagramme de séquence scénario consultation de la table inventaire

e. Diagramme de séquence scenario création d'une maintenance

Le digramme, exposé sur la figure II.6, décrit les scénarios possibles lors d'une opération de création d'une maintenance. En effet, si l'administrateur choisit d'effectuer une maintenance a un véhicule. Le système Fleet_maintenanceà son tour affichera une interface contenant des champs à remplir. L'administrateur commence par choisir le véhicule, ensuite le type de maintenance. Le résultat retenu est soit le sucée de l'opération de création soit l'apparition d'un message indiquant l'obligation de saisir des champs obligatoires.

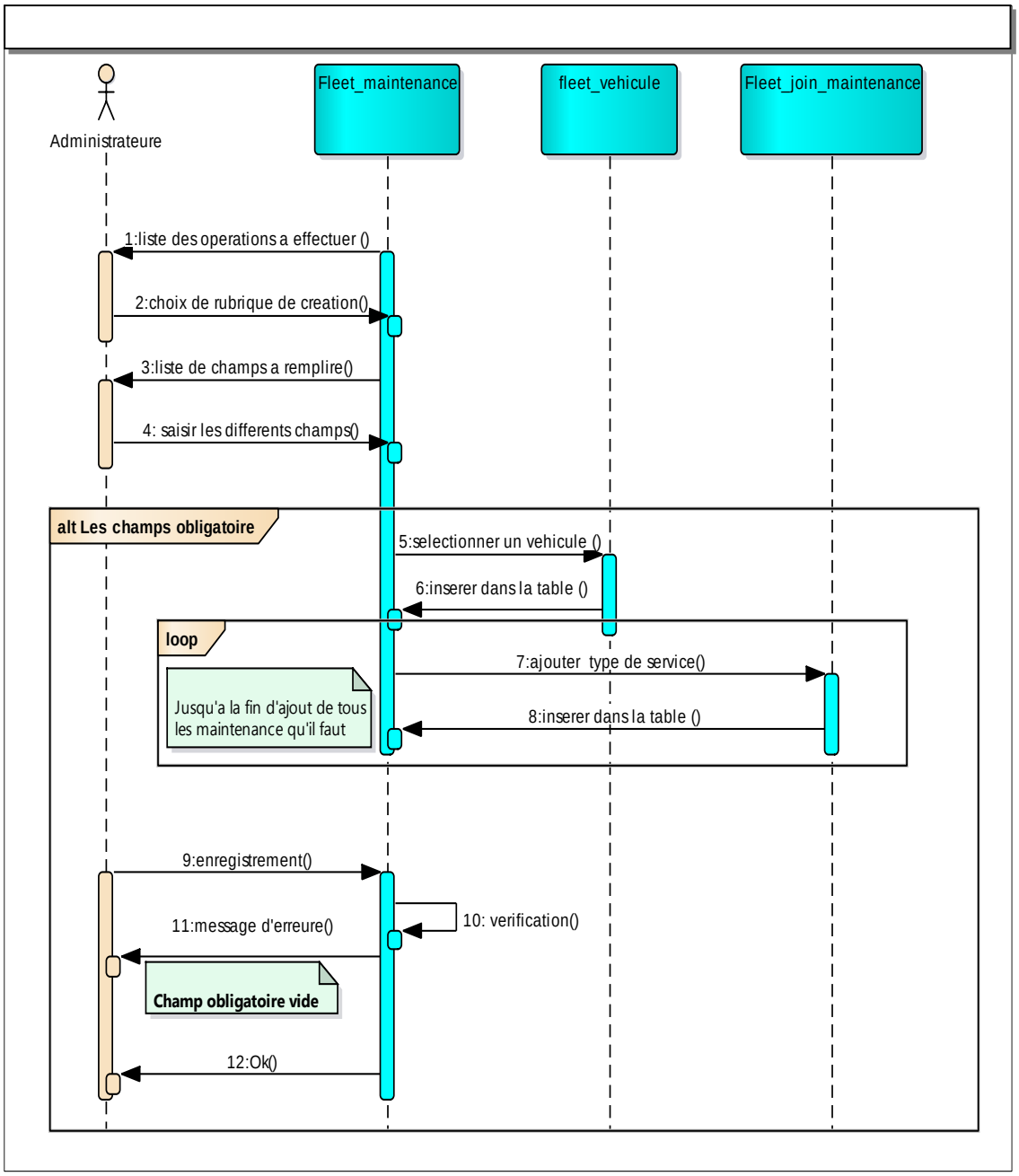


Figure II.6: diagramme de séquence scénario création d'une maintenance

f. Digramme de séquence scenario prévision d'une maintenance

La figure II.7, montre les scénarios de prévision d'une maintenance d'un véhicule si l'administrateur choisit de clôturer une mission. Le système Fleet_prévision à son tour vérifie le kilométrage de retour du véhicule utilisé, dans cette mission. S'il dépasse le kilométrage prévu d'une maintenance donc il sera automatiquement insérer dans la table prévision avec la description de la maintenance à effectuer pour se véhicule et en fin la mission est clôturée.

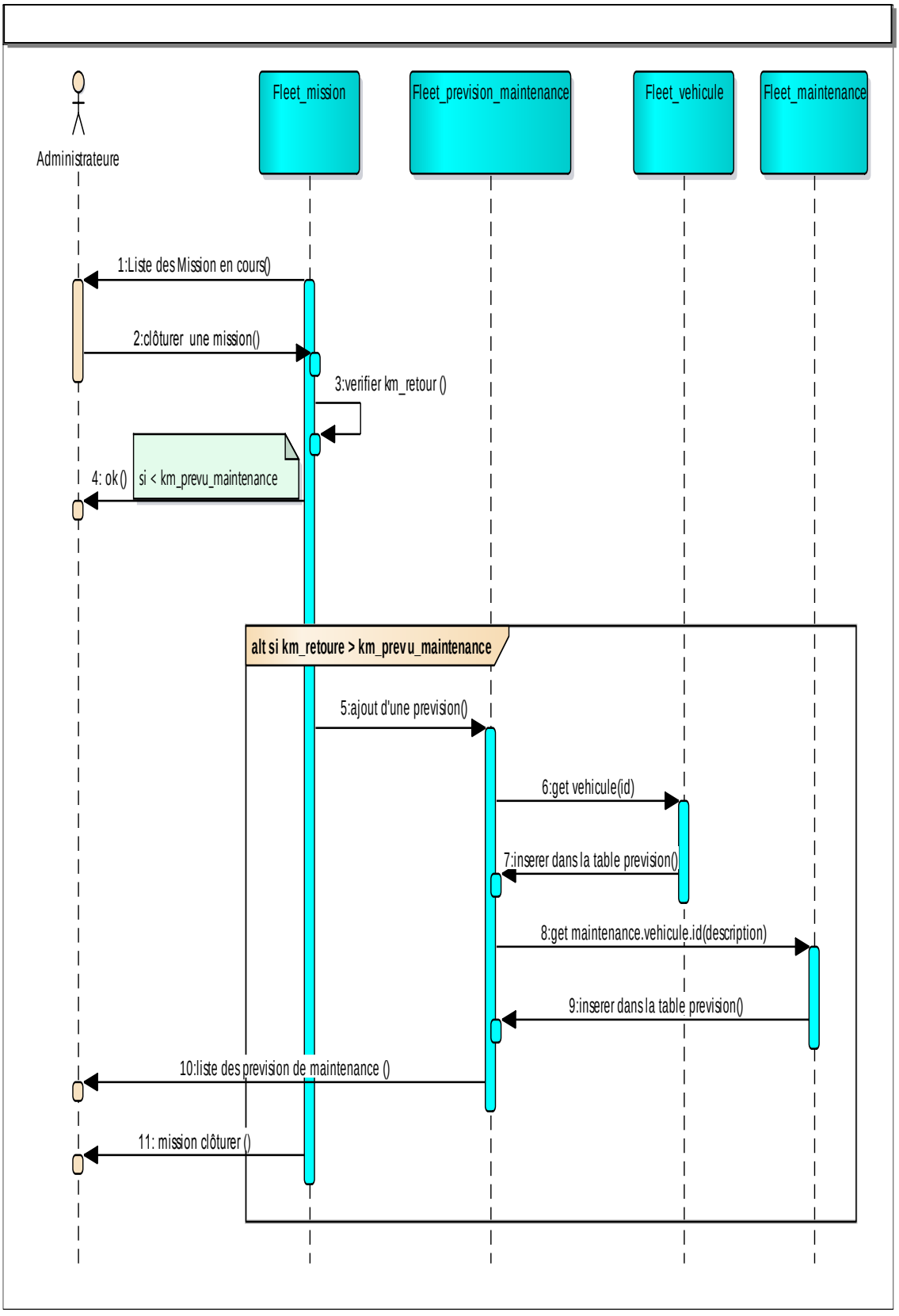


Figure II.7: diagramme de séquence scénario prévision d'une maintenance

III.3 Diagramme de classe

Contrairement au premier diagramme, le diagramme de classes montre la structure interne de système, en fournissant une représentation de ces objets.[7]

La figure II.8est constituée de plusieurs classes dont chacune représente un objet dans DZ Fleet et répond à un besoin bien précis. Nous définissons quelques classes :

Fleet_employée : cette classe regroupe les informations d'un employé qui peut être un chauffeur ou bien un accompagnateur (nom, prénom..). Un employé peut être affecté à une mission. Pour préciser le pays de l'employé nous avons utilisés une association a la classefleet_res_countryqui contient tous les états (pays) déjà définit par ODOO.**Fleet_vehicule** : Cette classe hérite de la classe véhicule (ODOO)

Fleet_maintenance : Cette classe représente les différents types de maintenance pour un véhicule. Elle est reliée par une jointure avec la classefleet_join_maintenance qui englobe tous les différents services fournit par ODOO.

Fleet_Resevation : Cette classe permet de regrouper la réservation d'un véhicule et son chauffeur avec une date de début et date de fin précise.

Fleet_document_de_bord : Cette class regroupe les différents documents de bord d'un véhicule (Assurance, Contrôle technique, vignette), le suivi par la date d'effet et la date d'expiration ET le cout de chaque type de document

Fleet_Inventaire : Cette classe regroupe toutes les différentes dépense des véhicules comme le cout de chaque maintenance ou bien la consommation des bons d'essence et de plus le cout de chaque document de bord .

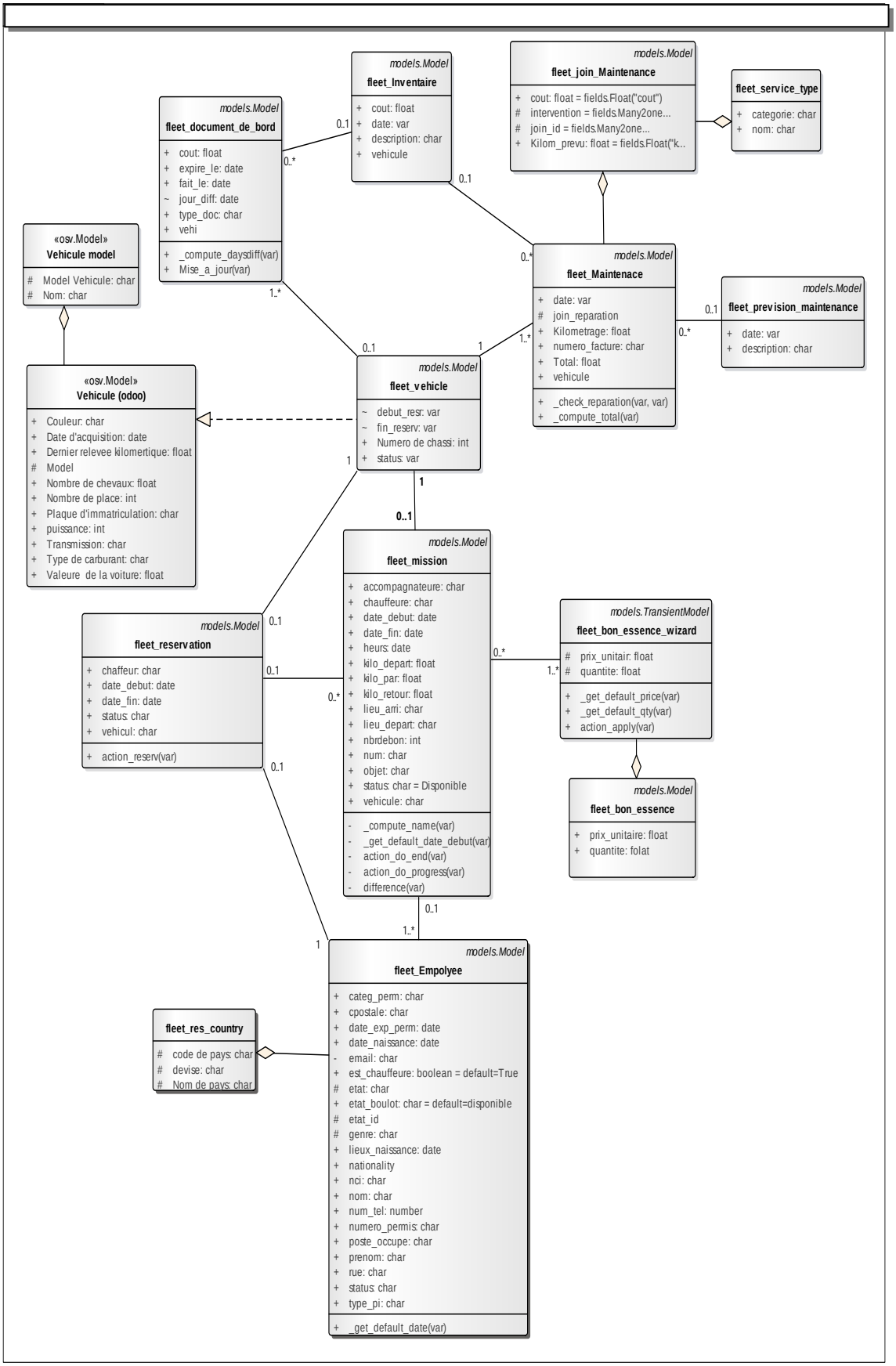


Figure II.8: Le diagramme de classe

IV. Conclusion

Dans ce chapitre on a présenté l'analyse et la conception du DZ Fleet. La conception a été faite par UML spécialement le diagramme de cas d'utilisation, les diagrammes de séquences et le diagramme de classe. Dans le chapitre suivant, nous abordons l'implémentation du DZ Fleet.

CHAPITRE III

LA RÉALISATION DU DZ FLEET

I. Introduction

Après avoir terminé la spécification et la conception du DZ Fleet, nous traitons dans ce chapitre les détails liés à l'implémentation de l'application. Pour cela nous allons tout d'abord exposer les choix technique que nous avons adoptés afin de réussir la réalisation de DZ Fleet. Enfin nous allons citer les étapes d'implémentation suivies de quelques imprimes d'écrans de l'exécution de certains modules de l'application pour illustrer quelques fonctionnalités de notre système.

II. Choix technique

Comme nous l'avons déjà cité notre travail se base sur l'open ERP libre ODOO (Voir chapitre 1). Ceci exige l'utilisation de certaines choix techniques tel que : PostgreSQL pour la base de donnée et deux langages de programmation XML et Python.

II.1 PostgreSQL

PostgreSQL est le système de gestion de base de données relationnelle et objet (SGBDRO) qui vise à stocker et à partager les données. (voir chapitre 1)

II.2 XML

Sous ODOO les vues sont générées à partir de fichiers XML .elle définissent comment l'utilisateur affiche les objets. Il peut y avoir plusieurs vues pour le même objet.

Nous citons quelques vues utilisés :

- Form : permet d'afficher les champs dans une page formulaire
- Tree : permet d'afficher les champs dans un tableau
- Search : permet de créer des filtres de recherche
- Actions : permet d'exécuter une action
- Menus items : permet de faire apparaitre le menu dans l'onglet des menu

Nous proposons un extrait de la vue form correspondante à l'objet réservation comme l'illustré sur la figure III.2

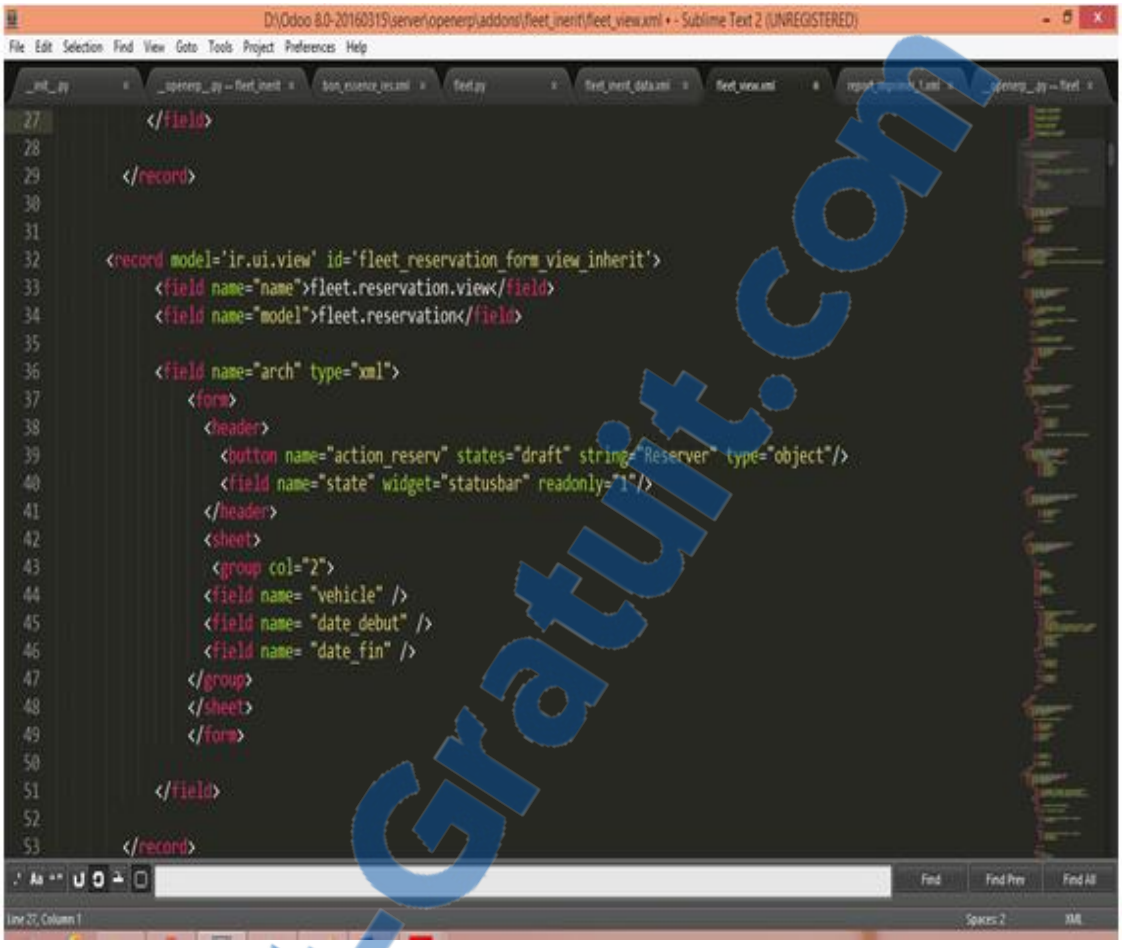
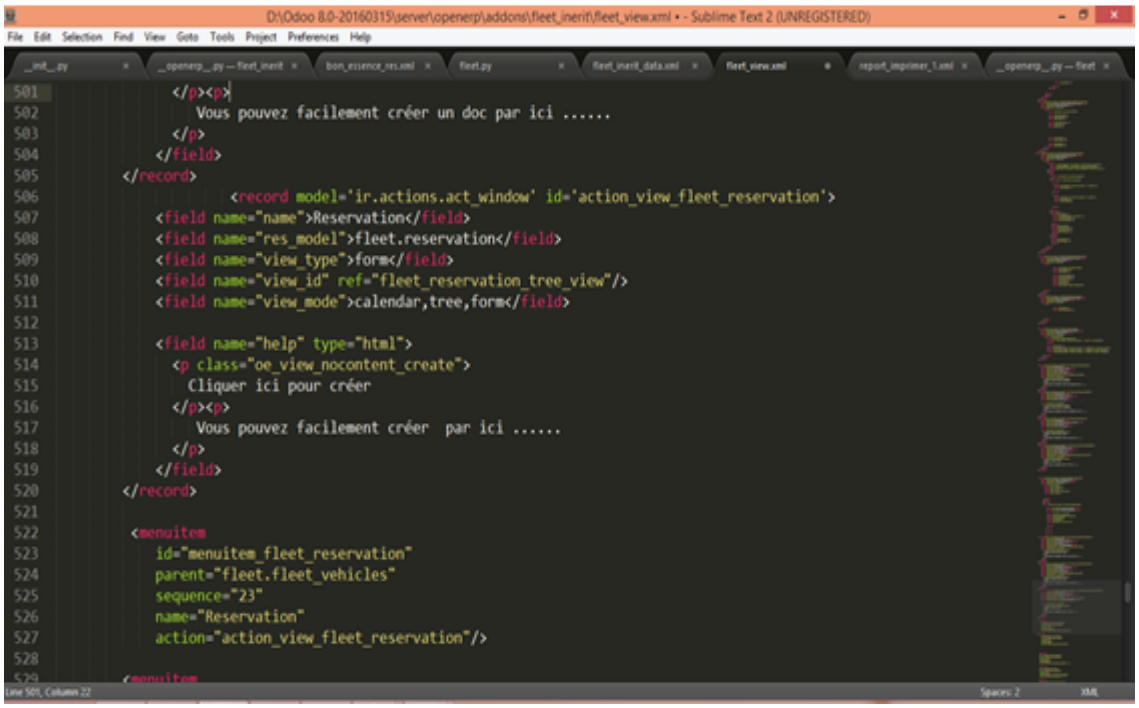


Figure III.1 : extrait de la vue form de l'objet réservation

La première ligne comporte l'identifiant de la vue et le modèle utilisé (id, model). Comme c'est une « Vue », le modèle utilisé sera toujours ir.ui.view (la vue sera enregistrée dans la table ir_ui_view d'ODOO).suivi de trois champs obligatoire, le champ (name=name) représente le nom de la vue, le champ (name=model) c'est le nom de la table utilisée et le champ (name =arch) C'est à l'intérieur de cette balise qu'on va mettre la vue proprement dite.

La figure III.2 présente la partie de déclaration des actions window et du menu item. C'est tout un petit paragraphe de codes qu'on ajoute après la partie précédente de la vue.



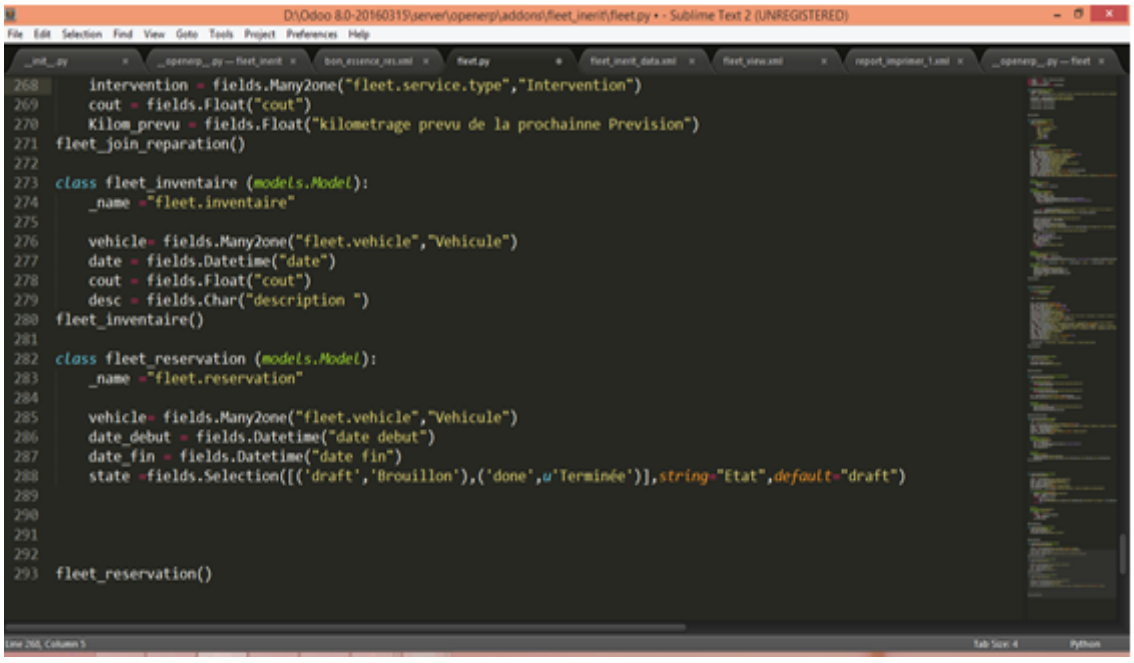
```
            </p><p>
              Vous pouvez facilement créer un doc par ici ......
            </p>
          </field>
        </record>
                <record model='ir.actions.act_window' id='action_view_fleet_reservation'>
          <field name="name">Reservation</field>
          <field name="res_model">fleet.reservation</field>
          <field name="view_type">form</field>
          <field name="view_id" ref="fleet_reservation_tree_view"/>
          <field name="view_mode">calendar,tree,form</field>

          <field name="help" type="html">
            <p class="oe_view_nocontent_create">
              Cliquer ici pour créer
            </p><p>
              Vous pouvez facilement créer  par ici ......
            </p>
          </field>
        </record>

         <menuitem
            id="menuitem_fleet_reservation"
            parent="fleet.fleet_vehicles"
            sequence="23"
            name="Reservation"
            action="action_view_fleet_reservation"/>
```

Figure III.2 : la vue action et menuitem de l'objet réservation

II.3 Python

C'est le langage orienté objet de programmation utilisé dans la logique ODOO pour la création d'objets.

La figure III.3 présente un extrait du code python de l'objet réservation



```
    intervention = fields.Many2one("fleet.service.type","Intervention")
    cout = fields.Float("cout")
    Kilom_prevu = fields.Float("kilometrage prevu de la prochainne Prevision")
fleet_join_reparation()

class fleet_inventaire (models.Model):
    _name ="fleet.inventaire"

    vehicle= fields.Many2one("fleet.vehicle","Vehicule")
    date = fields.Datetime("date")
    cout = fields.Float("cout")
    desc = fields.Char("description ")
fleet_inventaire()

class fleet_reservation (models.Model):
    _name ="fleet.reservation"

    vehicle= fields.Many2one("fleet.vehicle","Vehicule")
    date_debut = fields.Datetime("date debut")
    date_fin = fields.Datetime("date fin")
    state =fields.Selection([('draft','Brouillon'),('done',u'Terminée')],string="Etat",default="draft")



fleet_reservation()
```

Figure III.3 : extrait du code python de l'objet réservation

Cet extrait contient la déclaration de la classe class fleet_reservation (models.Model) qui représente la déclaration d'objet, ensuite _name qui est le nom de la table dans la base de donnée et les lignes qui suivent sont des champs que l'on va créer dans la table. !ù

III. Présentation de l'application DZ Fleet

Bien évidemment comme toutes les applications ODOO, nous avons d'abord se connecter puis accéder aux paramètres.

La figure III.4 présente l'interface de l'authentification

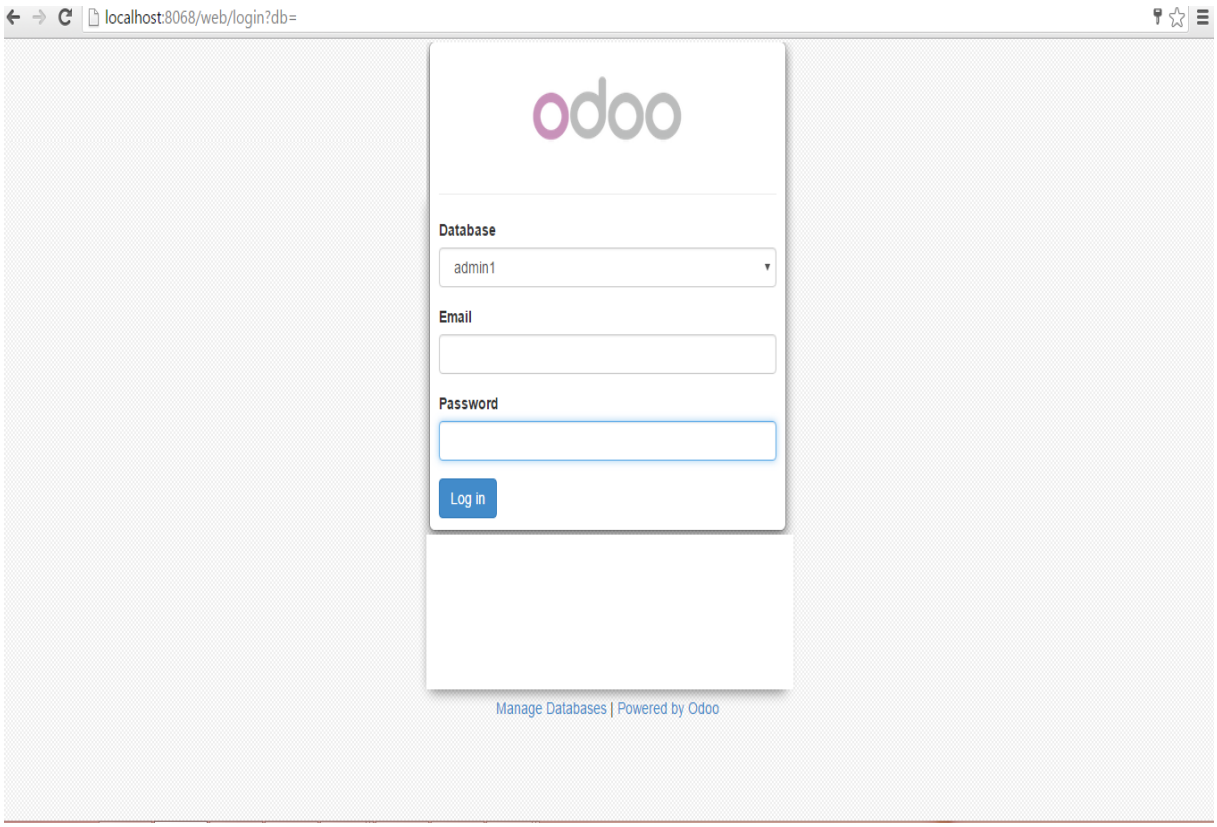


Figure III.4 : interface d'authentification

Une fois connecté, il faut préciser les paramètres, puis les modules. Nous lançons une mise à jour de la liste des modules. Ceci afin d'accéder à DZ Fleet.

La figure III.5 présente l'interface d'accueil ODOO –client web

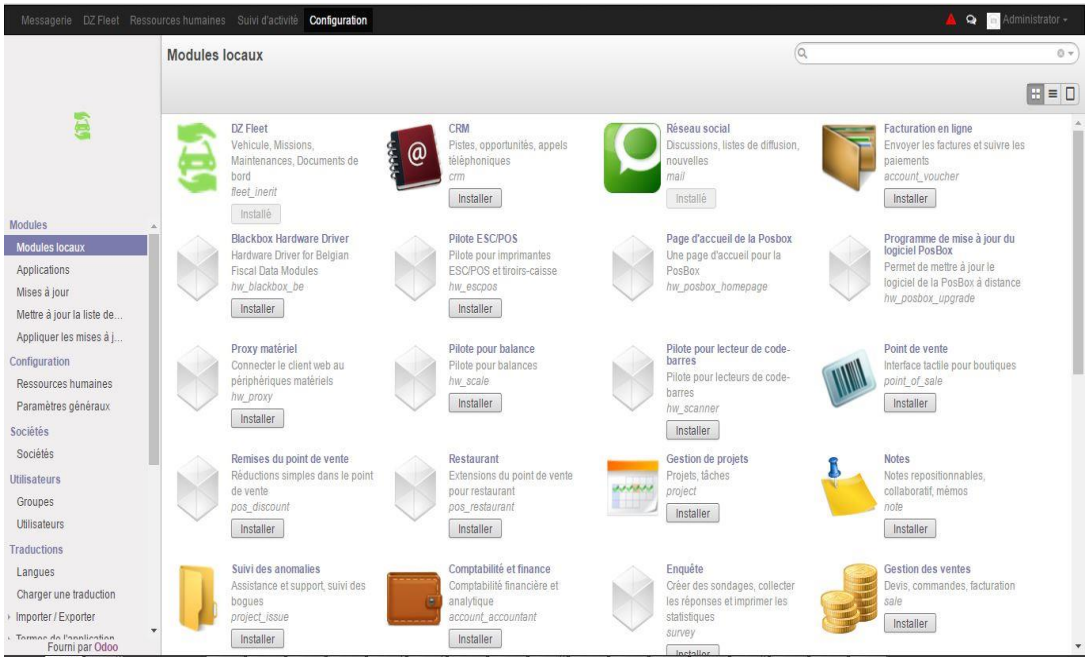


Figure III.5 : L'interface accueil ODOO - client web

Apres avoir accéder à DZ Fleet, l'interface sur la figure III.6 s'affiche. Elle contient le module installé et ses gestions.

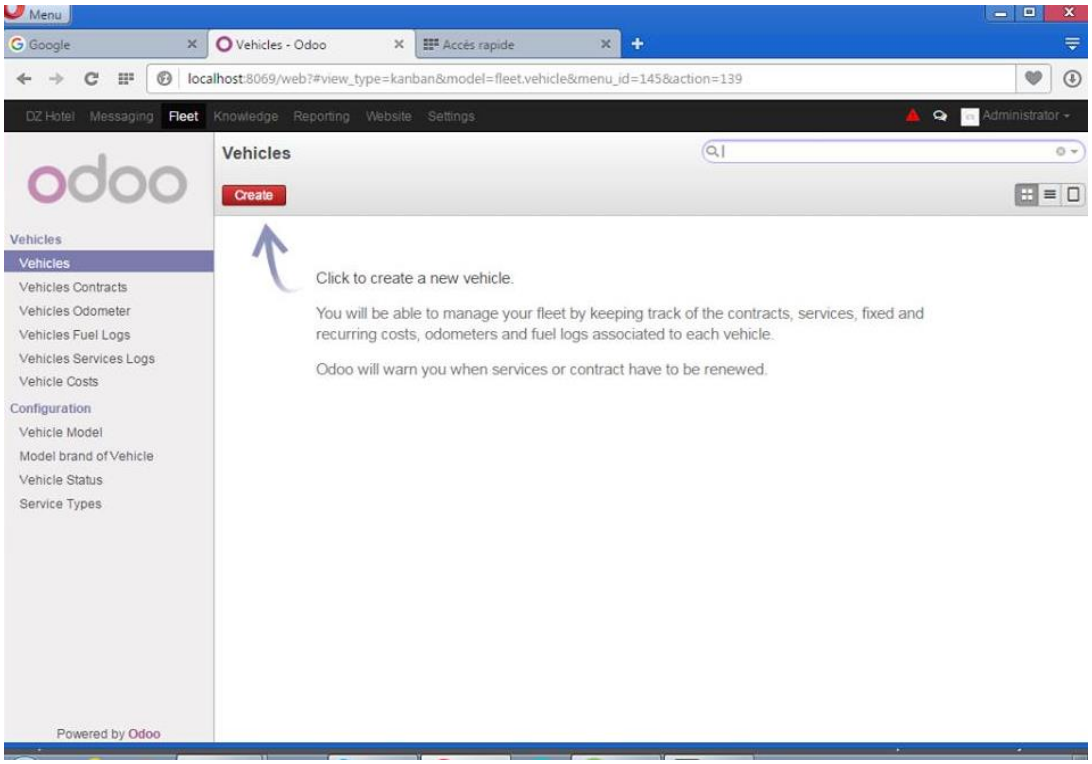


Figure III.6 : fonctionnalité de gestion des véhicules

III.1 Interface de gestion des véhicules

Dans le menu véhicule nous pouvons ajouter des véhicules, consulter la liste des véhicules, ainsi que les détails de chaque véhicule et la barre de la recherche personnalisée par un ensemble des critères. Le mode d'affichage peut être sous format liste, formulaire ou kanban.

a. La vue formulaire « form »

Dans cette interface l'utilisateur peut créer, modifier véhicule en saisissant les champs.il doit respecter les champs obligatoire qui sont en bleu comme le montre la figure III.7

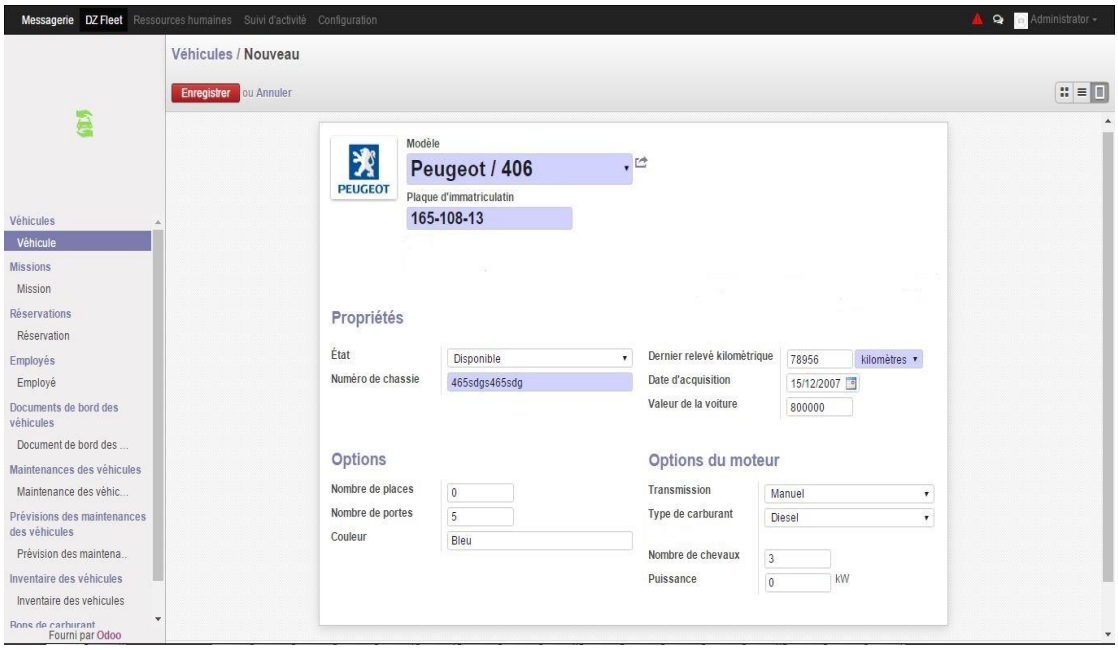


Figure III.7 : la vue formulaire de la gestion des véhicules

b. La vue liste « tree »

Elle permet de lister tous les véhicules enregistrer dans la base de données et leurs détails, aussi de faire une recherche des véhicule avec plusieurs critère de recherche en fonction des véhicules disponibles, en mission ou bien en pannes, ainsi que le regroupement de véhicule selon leur modèle comme illustré sur la figure III.8

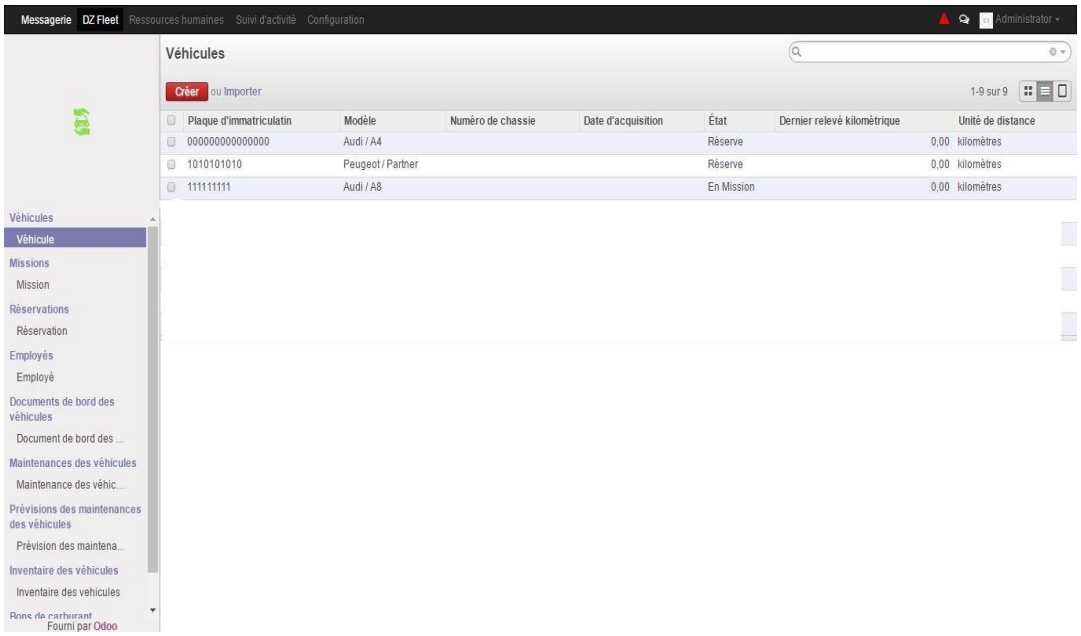


Figure III.8 : la vue liste de la gestion des véhicules

c. La vue kanban

Cette vue permet d'afficher seulement les détails les plus importants des objets de véhicule à l'accueil voir la figure III.9.

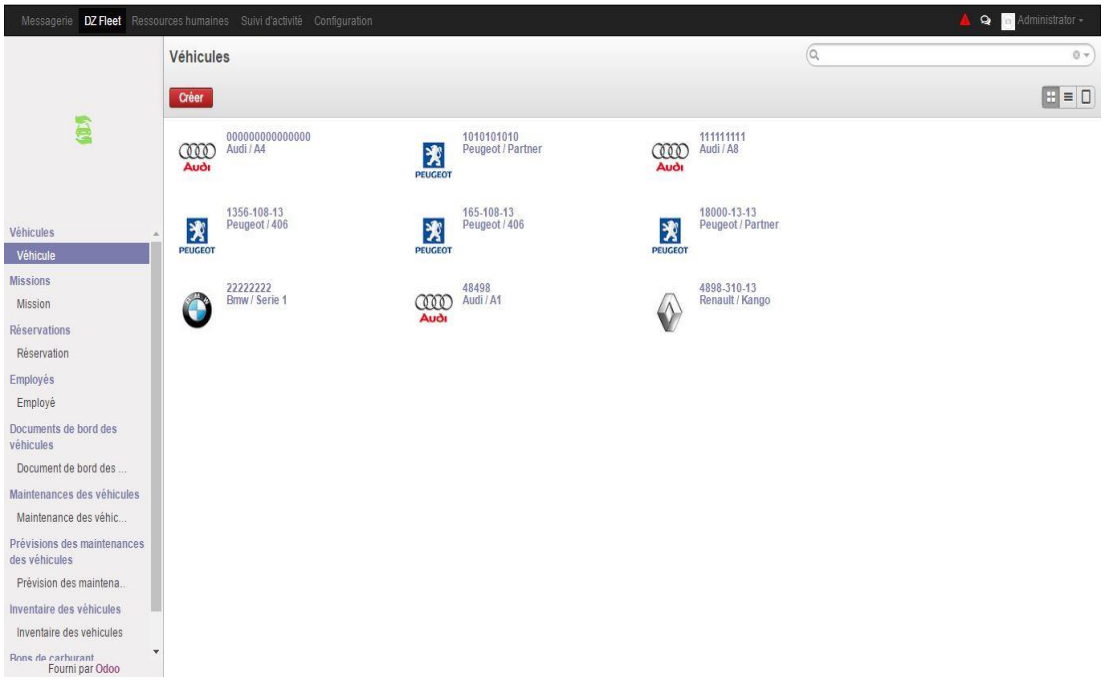


Figure III.9 : la vue kanban de la gestion des véhicules

III.2 Interface de gestion des missions

La gestion des missions dans DZ Fleet se fera à travers sa création et sa validation. La vue formulaire décrit les informations générales d'un ordre de mission.

- **Le numéro de la mission :** il se fait automatiquement et s'incrémente à chaque création.
- **Véhicule** : une liste des véhicules disponibles est proposée.
- **Chauffeur :** une liste des chauffeurs disponibles est proposée.
- **Accompagnateur :** une liste d'accompagnateurs disponibles est proposée.
- **Bon de carburant :** indique le nombre de bons qui ne doit pas dépasser le nombre de bon dans le stock (gestion de bons d'essence). Sinon un message d'alerte s'affiche indique l'insuffisance de quantité.
- **La date de début** : elle est créer automatiquement par rapport à la date actuelle.
- **Le kilométrage d'arriver** : ce champs sera rempli lorsque la mission est terminée afin qu'il soit affecté automatiquement au dernier relevé de kilométrage dans la base de la gestion des véhicules pour le même véhicule choisi.

De plus il y a deux champs calculables qui comptent la durée et le kilométrage parcourues durant la mission et pour la consommation des bons de carburants sera calculé et mis dans la table de gestion d'inventaire des véhicules.

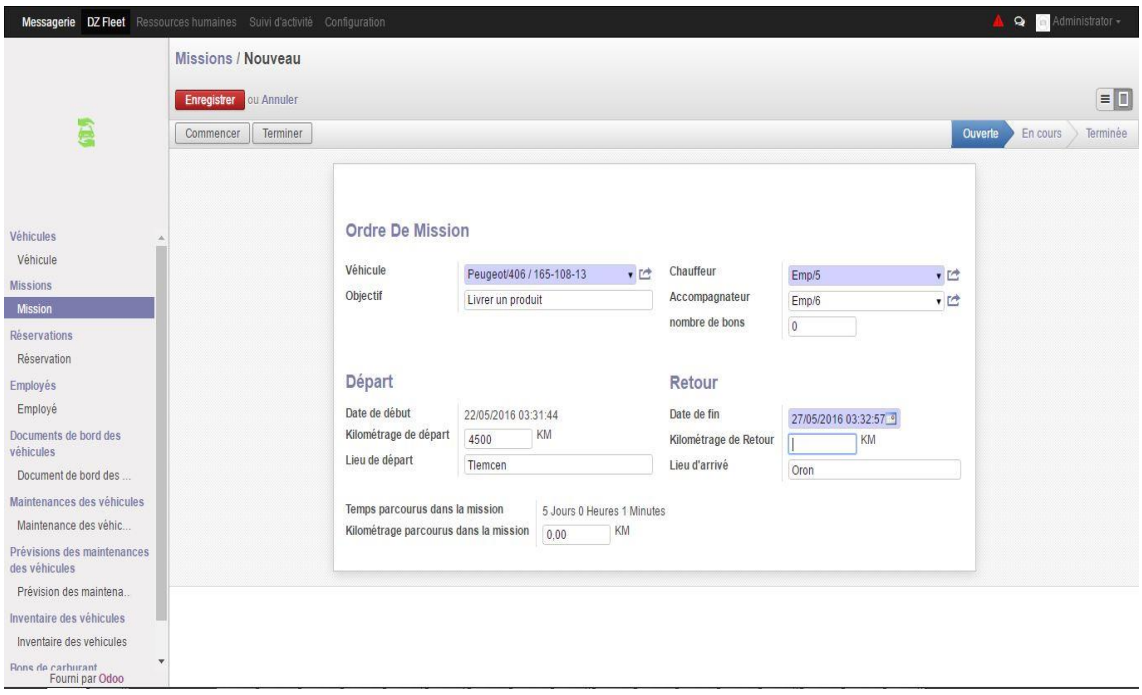


Figure III.10 : la vue formulaire de la gestion des missions

III.3 Interface de gestion des réservations de véhicule

La figure III.11 illustre la vue calendrier «calendar ».A travers cette vue l'administrateur peut effectuer une réservation d'un véhicule et son chauffeur pour une mission avec une date de début et date de fin et si le véhicule ou le chauffeur est déjà réserver dans la même date un message d'alerte apparaitra.

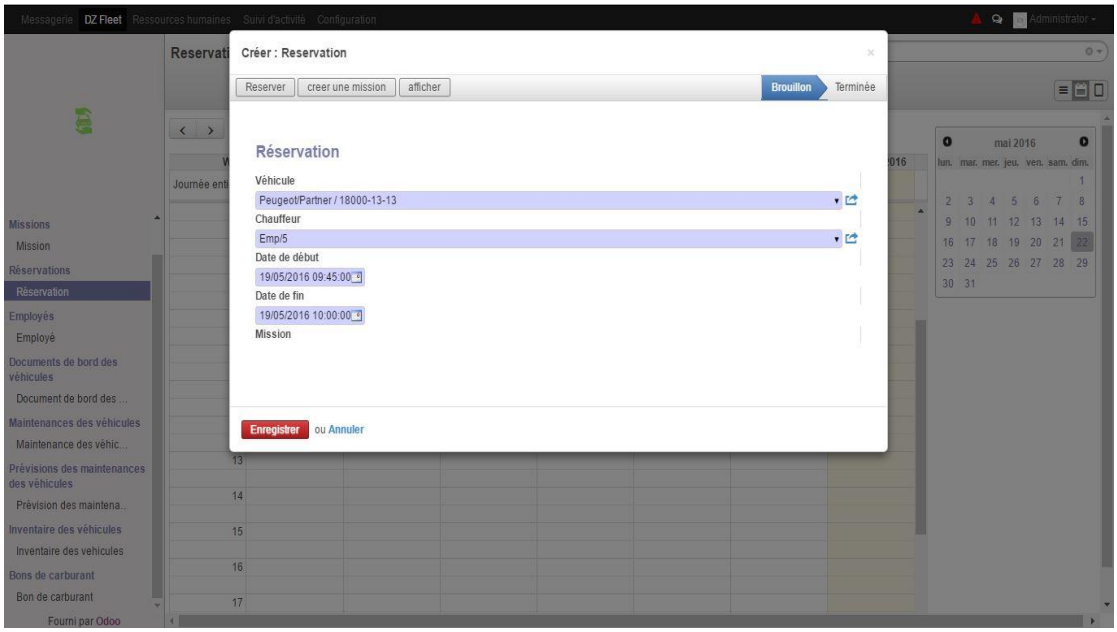


Figure III.11 : la vue calendrier de la gestion des réservations

III.4 Interface de gestion de maintenance des véhicules

La figure III.12 illustre la vue formulaire de la gestion de maintenance des véhicules. cette vue permet à l'administrateur de créer un ensemble d'interventions pour un véhicule avec son nom, son kilométrage, numéro de la facture, la date et le type d'intervention avec son cout et le cout total et le kilométrage prévu pour la prochaine intervention.

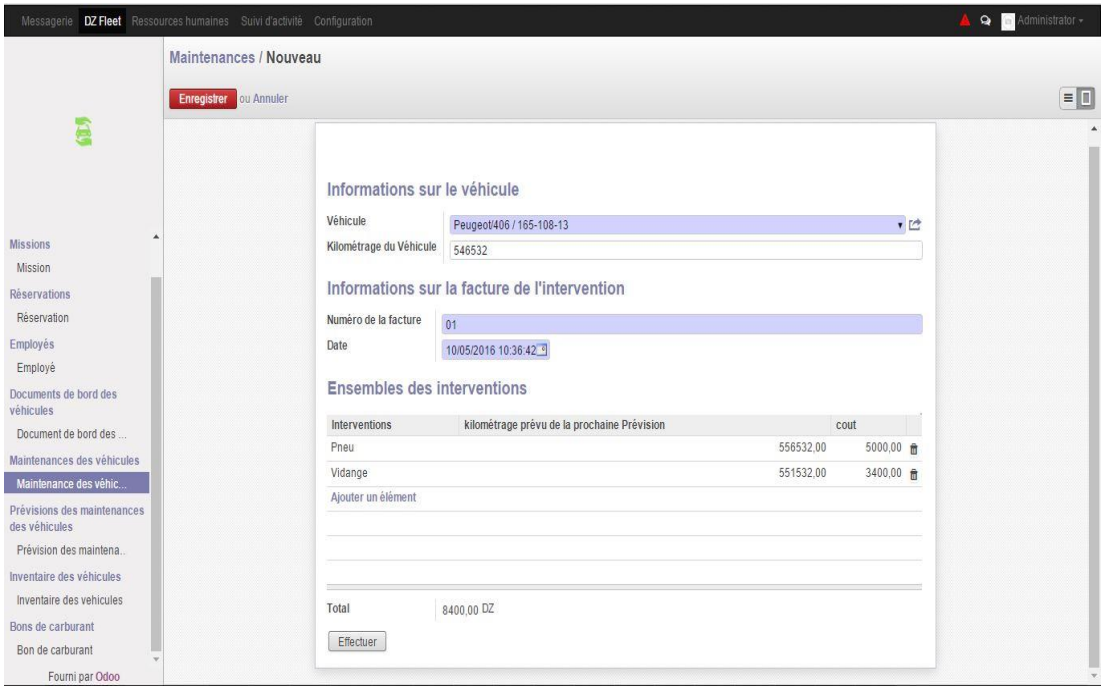


Figure III.12 : la vue formulaire de la gestion de maintenance

III.5 Interface de gestion de documents de bords des véhicules

Cette interface qui est représenté sur La figure III.13 permet d'indiquer les champs : nom du véhicule et son document de bord, sa date d'effet et d'expiration et le cout de document

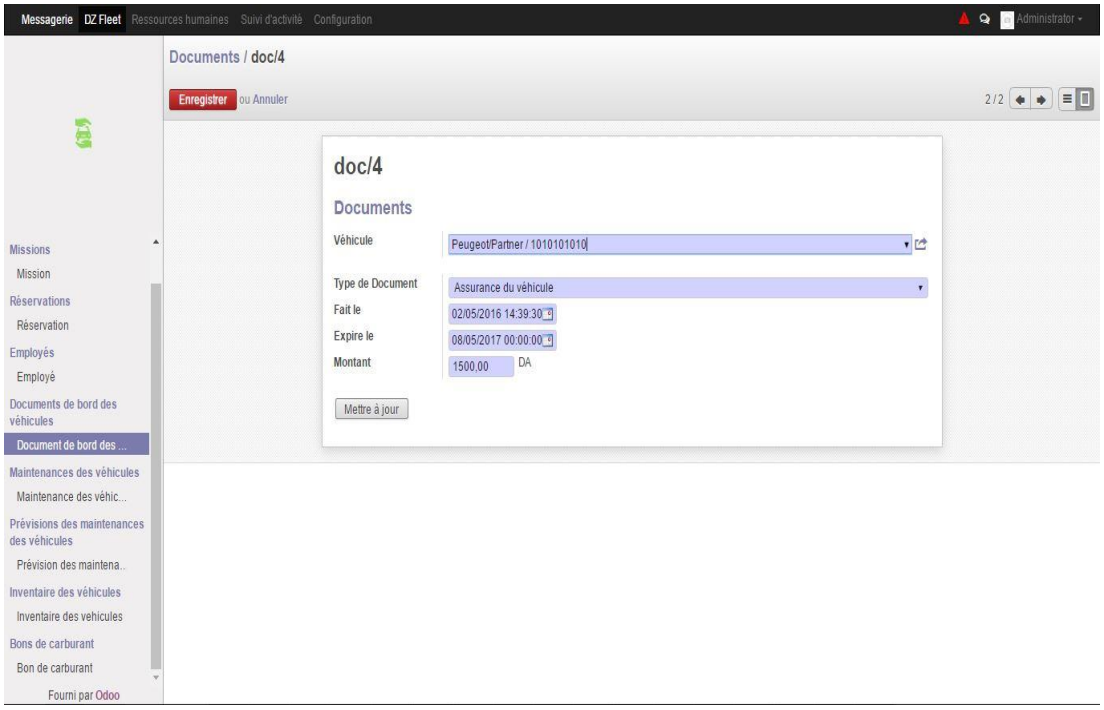


Figure III.13 : la vue formulaire de la gestion des documents de bord

Et pour la prévision de délais des documents l'administrateur doit consulter l'interface liste .S'il y a un véhicule que la date d'expiration de son document est proche. La ligne de ce dernier va être colorée en rouge pour avertir l'administrateur. C'est ce qui montre la figure III.14

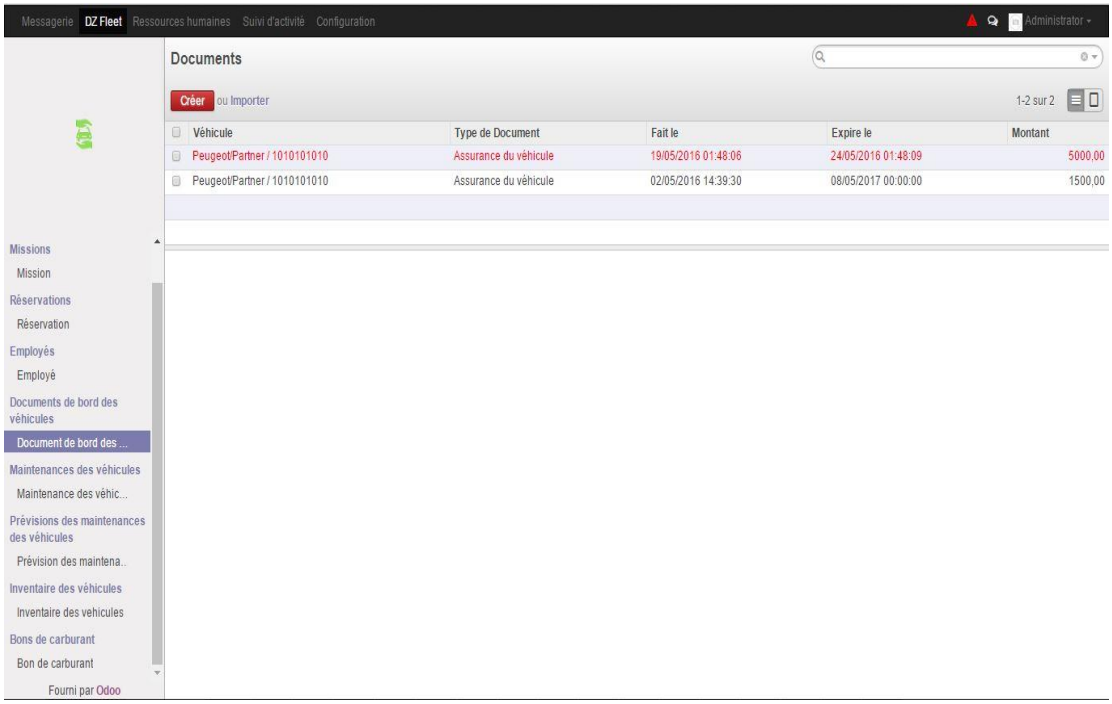


Figure III.14 : la vue liste de la gestion des documents de bord

III.6 Interface de gestion des bons de carburant

La figure III.1 illustre la vue Wizard. A travers cette vue l'administrateur rempli le champs de stock de bons de carburant en lui affectant un prix unitaire.

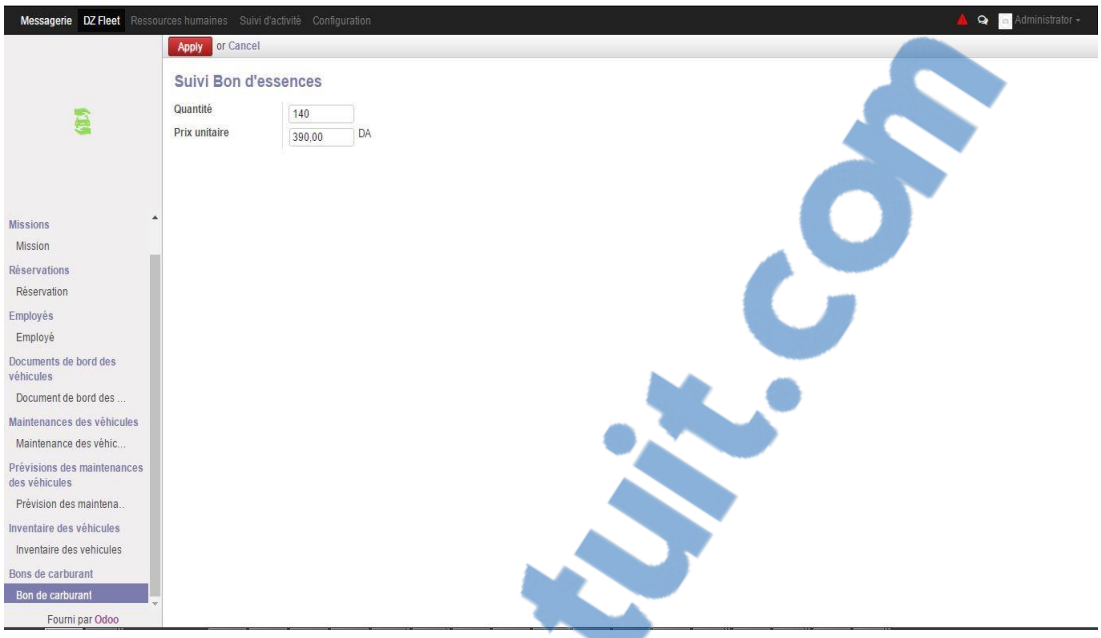


Figure III.15 : la vue Wizard de des bons de carburant

III.7 Interface de gestion d'inventaire des véhicules

La figure III.16 illustre la vue graphique qui représente les différents véhicules avec leur cout total de dépense (document de bords, maintenance, bon de carburant…). Elle peut être représentée en barre ou bien en secteur selon le choix de l'administrateur.

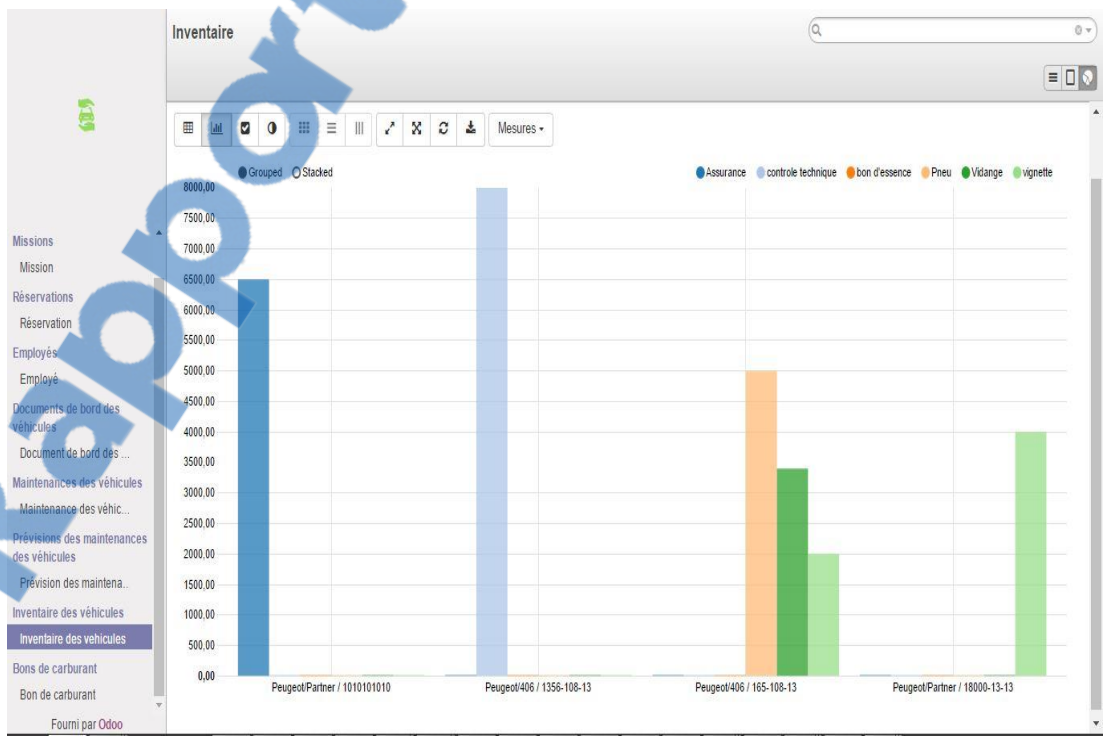


Figure III.16 : la vue graphique de la gestion d'inventaire des véhicules

IV. Conclusion

Dans ce chapitre nous avons, en premier lieu, présenté les différents outils et langages utilisés pour implémenter DZ Fleet. Par la suite, nous avons présenté en détails quelques interfaces de DZ Fleet.

CONCLUSION GENERALE

Le stage qui a été effectué au sein de l'entreprise SOGESI et qui nous a permis de réaliser notre projet de fin d'études a été sous plusieurs aspects riche d'enseignements. Nous avons commencé dès le début par comprendre le contexte général de notre application et identifier les différentes exigences de notre futur système.

Notre travail s'est basé sur le développement d'une application sous l'ERP libre ODOO, ce qui nous a amené à nous familiariser avec un nouvel environnement de développement auquel nous n'avons pas été formés. Par ailleurs, il nous a permis d'exploiter ce que nous avons acquis durant nos études universitaires.

Dans le cadre de notre projet nous avons conçu et développé une solution qui assure les différentes gestions d'un parc automobile sous la Platform ODOO. A savoir la gestion des missions, la gestion des véhicules, la gestion des employés, la gestion de bon de carburant, la gestion des document de bords des véhicules, la gestion d'inventaire des véhicules, la gestion de réservation de véhicule, la gestion de maintenance des véhicules et la gestion de prévision d'intervention.

Le travail que nous avons réalisé nous a procuré beaucoup de satisfactions car les principaux objectifs tracés ont pu être atteints. Nous sommes convaincus que l'informatisation du parc automobile va permettre une meilleure gestion des données, et va résoudre plusieurs problèmes de gestion manuelle responsable d'une circulation d'informations très lente et parfois incohérente.

En termes de perspectives, nous envisageons d'améliorer notre application sur plusieurs aspects comme par exemple le Mapping. Ainsi nous pourrons tracer le trajet d'une mission et avoir toujours un œil sur la position géographique du véhicule. De plus, nous pourrons étendre notre application sur tous les types de véhicules existants pour satisfaire le besoin de différentes entreprises.

I. Liste des tableaux et figures

Bibliographie

[1] Simon BAUDRY, Développement sur l'ERP OfbizNéogia, rapport de stage, université de tours, 2005-2006.
http://labs.libre-entreprise.org/frs/download.php/499/RapportStage_SimonBaudry.pdf

[2] Ait mlouk Addi, Conception et réalisation d'une application de gestion intégrée au sein de la société Eone Group basée sur OpenERP, thèse de Master, université de Marrakech, 2013
http://fr.slideshare.net/cherkamed/rapport-38424059

[3] JOUDAR Youssef et TAZARNI, OpenERP a la poste Maroc, université de Maroc, 2013-2014
http://fr.slideshare.net/nextma/openerp-la-postemaroc

[4] le site officiel de ODOO, 2016
https://www.odoo.com/fr_FR/page/download

[5] Taieb kristou, OpenERP - Gestion de parc Véhicules, tutoriel, 2014
https://youtu.be/mEXSyYlqMOw

[6] le site officiel de l'ERP DIGIPARC, 2016
http://digiparc.com/fr-fr/Accueil

[7] laurent-piechocki, cours UML, tutoriel, 2007
http://laurent-piechocki.developpez.com/uml/tutoriel/lp/cours/

[Open, 2012] le site officiel d'OpenERP
http://www.oepnerp.com

[Wiki, 2012] informations sur PgAdmin
https://fr.wikipedia.org/wiki/PostgreSQL#pgAdmin

[Postgre, 2012] site officiel du PostgreSQL
http://www.postgresql.org/

Résumé

Jour après jour, la gestion de la flotte automobile ne cesse de gagner en importance au sein de toutes les entreprises, quelles que soient leurs tailles.

A l'ère des systèmes d'information intégrés, la possession d'un progiciel est un atout majeur pour l'entreprise. Le présent travail vient pour répondre au besoin de la création d'un module, que nous avons nommé DZ Fleet, pour la gestion de la flotte automobile à l'aide du PGI (Progiciel de Gestion Intégré) open source ODOO. DZ Fleet traite et facilite la gestion des véhicules, la gestion des missions, la gestion des employés ainsi que la gestion d'inventaire des véhicules…etc.

Mots clé : ERP (Enterprise Ressource Planning), PGI (Progiciel de Gestion Intégrée), ODOO, gestion de la flotte automobile.

Abstract

Day after day, the fleet management is gaining in importance in all businesses, whatever their size.

In the era of integrated information systems, possession of software is a major asset for the company. This work has to meet the need of creating a module, which we named DZ Fleet, for the management of the vehicle fleet with the ERP (Enterprise Resource Planning) open source Odoo. DZ Fleet trafficking and facilitates the management of vehicles, mission management, employee management and inventory management vehicle ... etc.

Keywords:ERP (Enterprise Resource Planning),PGI (Progiciel de Gestion Intégrée), Odoo, fleet management.

ملخص

في الآونة الأخيرة شهد تطور المؤسسات بشتى أنواعها تقدما ملحوظا مما انجر عنه تطوير مختلف فروعها والتي يعتبر تسيير حضيرة السيارات جزءا منها.

بمساعدة الأنظمة الإعلامية الحديثة فإن حيازة كل مؤسسة على برنامج في الإعلام الآلي خاص بتسيير هذا الفرع يعتبر ميزة خاصة بها، في هذا النطاق جاء مشروع تخرجنا لإنشاء وحدة برمجية بنظام برنامج أودو المفتوح خاصة بتسيير حضيرة السيارات للمؤسسة والذي أطلقنا عليه اسم د يزاد فليت.

هذه الوحدة البرمجية تعالج وتسهل تسيير الحضيرة بما فيها تسيير السيارات والمهمات و عمال الحضيرة و عملية جرد السيارات.

الكلمات المفتاحية: البرنامج المفتوح لتخطيط موارد المؤسسات – أودو – ديزاد فليت – تسيير حضيرة السيارات.