Dubail Noah 15/09/2023

Gestion des frais GSB



**Lycée Condorcet BTS SIO – SLAM Bloc 2**

# **Tables des matières**

[**Tables des matières** 2](#_Toc155289269)

[**Historique du document** 3](#_Toc155289270)

[**Description du sujet de projet** 4](#_Toc155289271)

[**Le contexte du projet** 4](#_Toc155289272)

[**Le domaine de développement** 4](#_Toc155289273)

[**Le fonctionnement de l’application web** 4](#_Toc155289274)

[**Le périmètre du système** 5](#_Toc155289275)

[**Les extensions possibles** 5](#_Toc155289276)

[**La modélisation conceptuelle** 6](#_Toc155289277)

[**Le Modèle de dictionnaire** 6](#_Toc155289278)

[**Le Modèle entité association** 8](#_Toc155289279)

[**La modélisation logique** 11](#_Toc155289280)

[**Script de création de la base de données** 13](#_Toc155289281)

[**Modélisation UML** 15](#_Toc155289282)

# **Historique du document**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date de mise à jour | Auteur | Modification essentielles |
| 1.0 | 22/09/2023 | Noah Dubail | Création du document |
| 1.1 | 29/01/2023 | Noah Dubail | Mise à jour version E2 |
| 1.1.1 | 04/01/2024 | Noah Dubail | Mise à jour des informations du cahier des charges | Mise à jour des diagrammes UML. |

# **Description du sujet de projet**

## **Le contexte du projet**

GSb (Galaxy Swiss Bourdin) est née de la fusion des deux sociétés pharmaceutiques : le géant américain Galaxy et le conglomérat européen Swiss Bourdin. Cette fusion a pour but l’optimisation de l’activité du groupe, en réalisant des économies d’échelle dans la production et la distribution des médicaments, tout en prenant le meilleur des deux laboratoires sur les produits concurrents.

Mon objectif est de concevoir un prototype de logiciel de gestion des frais de déplacement pour aider l'entreprise à économiser et simplifier cette tâche, particulièrement complexe au sein de grandes entreprises. En effet les déplacements et actions de terrain menées par les visiteurs engendrent des frais qui doivent être pris en charge par la comptabilité. GSB dépense chaque année plus de 25 millions d’euro dans les frais de déplacement.

## **Le domaine de développement**

Ce projet sera développé avec le framework laravel sur l’IDE Visual Studio Code pour l’application web, et sera couplé à une base de données MySQL avec laquelle il pourra interagir. Une historisation des fichiers sera effectuée grâce à git.

Lien GitHub : <https://github.com/99Noah99/Projet_GSB>

## **Le fonctionnement de l’application web**

Le fonctionnement de l’application sera simple : une fois leur déplacement effectué, les utilisateurs vont pouvoir créer la « mission » correspondant à leur déplacement et ajouter les différentes dépenses qu’ils ont effectué durant cette mission afin d’être remboursé.

De leur côté les comptables pourront consulter les missions des utilisateurs et leur frais correspondant, ils pourront donc choisir de valider demande de remboursement de la missions et ses frais ou non.

**Les différents frais :**

Il existe différents types de frais forfaitisé :

* Le repas du midi sera géré par un forfait à 20€
* Le relais étape (nuit + repas) sera géré par un forfait à 120€
* Une nuit dans un hôtel sera gérée par un forfait à 90€
* Les kilomètres seront remboursés à hauteur de 0.20€ par kilomètres.
* Les billets d’avions seront entièrement remboursés
* Les billets SNCF seront entièrement remboursés

**Etat des demandes :**

Une fois que l’utilisateur à créer sa mission, il pourra consulter l’évolution de sa demande, voici les différents statuts :

* En attente de déclaration (l’utilisateur a créé la mission mais n’a pas fini d’ajouter ses frais, le comptable ne voit pas la mission sur son tableau de bord)
* En attende de validation (l’utilisateur à déclarer sa mission et attend la validation ou le refus d’un comptable)
* Validée (le comptable a validé la demande, l’utilisateur à le statut Validée sur sa liste des missions)
* Refusée (le comptable a refusé la demande par manque d’information, ou de justificatif… l’utilisateur à le statut Refusée sur sa liste des missions)

**Ordre des actions :**

Premièrement l’utilisateur se connecte via son identifiant et son mot de passe, il peut ensuite crée sa mission. Il arrive ensuite sur la mission qu’il a créé et peut ajouter des frais liés à cette mission. Pour chaque frais une pièce justificative sera demandée. Une fois tous les frais ajoutés il peut déclarer la mission.

Le comptable peut à son tour se connecter via un identifiant et un mot de passe et consulter le(s) la liste de tous les utilisateurs, il clique sur un utilisateur et peut visionner toutes les missions que celui si à déclarer. Il peut maintenant valider ou refuser.

Si il refuse, l’utilisateur pourra voir dans la liste de ses missions que le statut de sa mission a changé en « Refusée », si il accepte le statut changera en « Validée ».

**Les différents comptes :**

L’application aura donc 2 interfaces, une interface pour les utilisateurs qui peuvent ajouter leur missions, frais et pièces justificatives, et une interface pour les comptables qui vont pouvoir consulter la liste de tous les utilisateurs et leur mission.

Les visiteurs pourront donc :

* Ajouter des missions
* Ajouter les frais qu’ils ont dépenser durant cette mission et leur justificatif
* Déclarer leur mission une fois tous les frais ajoutés

Les comptables pourront :

* Visionner la liste de tous les utilisateurs
* Visionner les missions d’un utilisateur
* Visionner les frais d’une mission et télécharger les pièces justificatives.
* Valider ou Refusée la demande de remboursement d’une mission

## **Le périmètre du système**

Une application qui prend un charge une interface pour les visiteurs, et une interface pour les comptables, une interface admin permettant d’ajouter des nouveaux utilisateurs ou comptable dans l’application.

Une base de données MySQL qui communiquera avec l’application web afin de stocker les factures, les demandes de remboursement, les comptes utilisateurs et comptables…

## **Les extensions possibles**

Ajouter une date de visite ce qui permettrait de laisser un délai de 10 jours après la date de visite pour faire la demande de remboursement.

A la fin du mois, toute les demandes de remboursement non clôturés seront automatiquement clôturées par l’application.

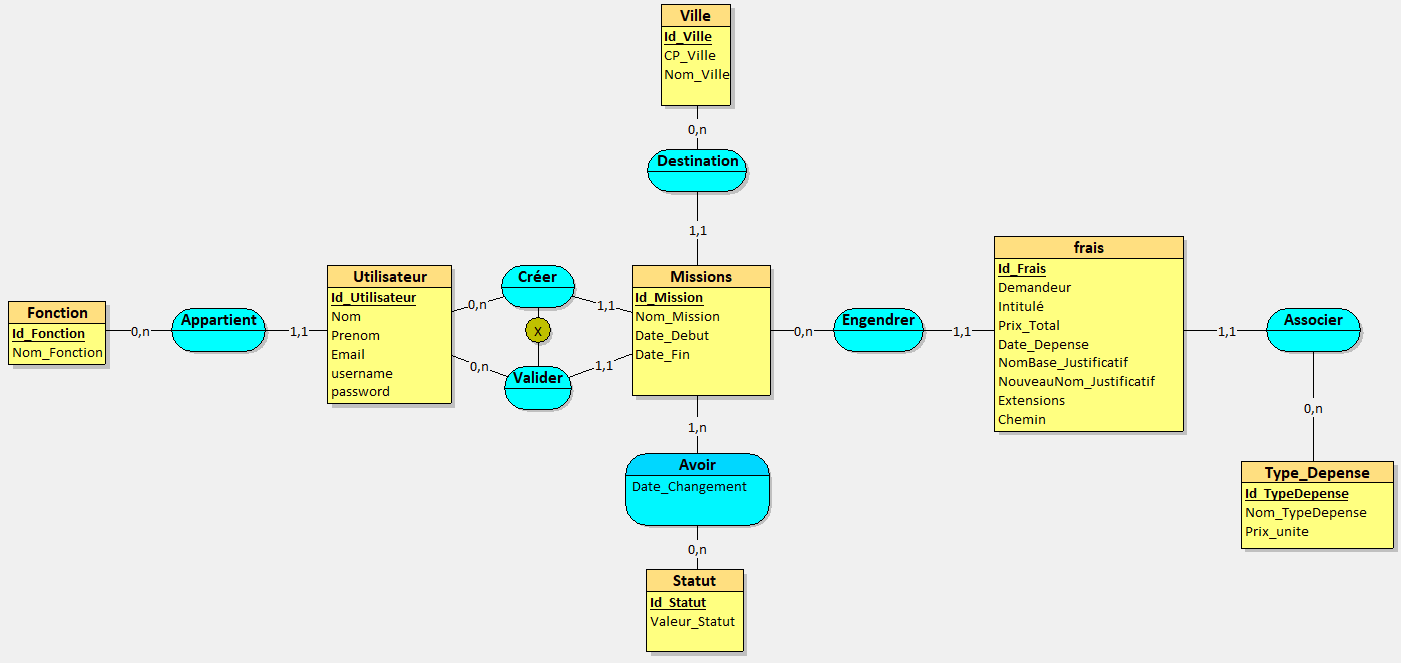
# **La modélisation conceptuelle**

## **Le Modèle de dictionnaire**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nom Conceptuel** | **Nom Logique** | **Type (E,Ca,Co)** | **Nature** | **Longueur** | **Identifiant** | **Exemple de valeur** |
| Identifiant de l’Utilisateur | Id\_Utilisateur | E | N | 4 | Oui | 251 |
| Nom de l’Utilisateur | Nom | E | A | 30 | Non | Walter |
| Prénom de l’Utilisateur | Prenom | E | A | 30 | Non | Thibaut |
| Email de l’Utilisateur | Email | E | AN | 100 | Non | [Walter.Thibaut@gmail.com](mailto:Walter.Thibaut@gmail.com) |
| Identifiant de connexion de l’Utilisateur | username | E | AN | 20 | Non | WThibaut8 |
| Mot de passe de l’Utilisateur | password | E | AN | 255 | Non | TW9$\*\* |
|  |  |  |  |  |  |  |
| Identifiant de la Ville | Id\_Ville | E | N | 20 | Oui | 42 |
| Code Postal de la ville | CP\_Ville | E | N | 5 | Non | 9000 |
| Nom de la Ville | Nom\_Ville | E | A | 40 | Non | Toulouse |
|  |  |  |  |  |  |  |
| Identifiant de la Mission | Id\_Mission | E | N | 5 | Oui | 23 |
| Nom de la Mission | Nom\_Mission | E | A | 250 | Non | Déplacement à Belfort |
| Date du début de la Mission | Date\_Debut | E | AN | 10 | Non | 02/11/2023 |
| Date de fin de la Mission | Date\_fin | E | AN | 10 | Non | 07/11/2023 |
|  |  |  |  |  |  |  |
| Date du changement du statut | DateChangement | E | AN | 10 | Non | 05/11/2023 |
|  |  |  |  |  |  |  |
| Identifiant du statut | Id Statut | E | N | 5 | Oui | 367 |
| Valeur du statut | Valeur\_Statut | E | A | 100 | Non | Refusé |
|  |  |  |  |  |  |  |
| Identifiant du frais | Id\_Frais | E | N | 5 | Oui | 545 |
| Demandeur du Remboursement | Demandeur | E | A | 30 | Non | Patrick |
| Intitulé de la demande de Remboursement | Intitulé | E | A | 200 | Non | Train concernant mon déplacement à Belfort |
| Prix de la demande de remboursement | Prix\_Total | E | N | 5 | Non | 250.00 |
| Date de la dépense | Date\_Depense | E | AN | 10 | Non | 20/10/2023 |
| Nom par défaut du justificatif | NomBase\_Justificatif | E | A | 150 | Non | JustificatifTransport.pdf |
| Nouveau nom du justificatif | NouveauNom\_Justificatif | E | AN | 150 | Non | PdfTransport250.pdf |
| Type d’extension | Extension | E | A | 10 | Non | .pdf |
| Chemins d’accès au Justificatif | Chemin | E | AN | 300 | Non | C:\Users\Utilisateur\Documents\BTS SIO\2ème année\SLAM |
|  |  |  |  |  |  |  |
| Identifiant du type de dépense | Id\_TypeDepense | E | N | 4 | Oui | 24 |
| Nom du type de dépense | Nom\_TypeDepense | E | A | 50 | Non | Repas midi |
| Prix de la dépense | Prix\_unite | E | N | 6 | Non | 20.00 |
|  |  |  |  |  |  |  |
| Identifiant de la fonction de l’utilisateur | Id\_Fonction | E | N | 4 | Oui | 3 |
| Nom de la fonction de l’utilisateur | Nom\_Fonction | E | A | 50 | Non | Comptable |

## **Le Modèle entité association**

Voici la modélisation du modèle entité association :



**Explications sur le choix de la modélisation :**

Entité Utilisateur :

Un utilisateur est identifié par un identifiant unique (ici id\_Utilisateur) et possède les attributs suivants : Nom, Prenom, Email, username (identifiant de connexion), password.

Un utilisateur a une fonction : utilisateur, comptable, admin.

Un utilisateur ayant la fonction utilisateur peut créer 0 ou plusieurs missions.

Un utilisateur (fonction comptable) peut valider 0 ou plusieurs missions. Un utilisateur ne peut pas créer ET valider une mission il peut faire sois l’un sois l’autre, en effet les utilisateurs ayant la fonction utilisateur crée les missions, et les utilisateurs ayant la fonction comptable valide ou refuse les missions.

Une mission est créée ou valider que par un seul utilisateur.

Nous avons relié la table Utilisateur à la table Mission par deux associations, cependant une contrainte d'exclusivité a été ajouté sur ses associations afin qu’il soit impossible pour un utilisateur à la fois de crée et de valider une mission.

Entité Missions :

Une mission est identifiée par un identifiant unique (ici id\_Missions) et possède les attributs suivants : Nom\_Mission, Date\_Debut, Date\_Fin.

Le nom de la mission permet de savoir à quoi correspond la mission, et les dates de début et de fin permettent de déterminer la durée de la mission.

Une mission a lieu dans une seule ville.

De plus, une mission peut avoir un ou plusieurs statuts, car celui-ci va évoluer au fil du temps (validé, refusé…).

Pour finir, une mission entraîne 0 ou plusieurs dépenses (frais).

Lors de la mission, l’utilisateur va dépenser de l'argent pour des repas, des trajets, des nuits d’hôtel, etc. Toutes ces dépenses sont enregistrées dans la table "frais" associée à la mission. Les frais sont calculés et classés par type de dépense. En effet, la table "Type\_Depense" permet de catégoriser les frais, par exemple, trajet, repas… Dans la table "Type\_Depense", on retrouve également le prix unitaire de chaque catégorie. La catégorie du frais est définie par la propriété "Nom\_TypeDepense".

Entité Ville :

Une ville est identifiée par un identifiant unique (ici id\_Ville) et possède les attributs suivants : CP\_Ville, Nom\_Ville, Rue.

Le CP\_Ville correspond au code postal, le Nom\_Ville au nom de la ville.

Dans une ville peut avoir lieu 0 ou plusieurs missions.

La table Ville permet d’avoir plus d’information concernant le lieu de la mission.

Entité Statut :

Un statut est identifié par un identifiant unique (ici id\_Statut) et possède l’attribut suivant : Valeur\_Statut.

La Valeur\_Statut va contenir le statut d’une mission, si elle est validée par le comptable, refusée…

Le statut est affecté à 0 ou plusieurs missions.

L’association reliant mission et statut possède l’attribut DateChangement qui nous permet de connaitre la date à laquelle le statut a changé.

Nous avons ajouté la table Statut afin de suivre les différentes étapes de la mission, de plus dans l’association reliant mission et statut se trouve la propriété DateChangement permettant de savoir à quel moment le statut a été modifié.

Entité Frais :

Un frais correspond à une dépense effectuée pendant une mission. Il est identifié par un identifiant unique (ici id\_Remboursement) et possède les attributs suivants : Demandeur, Intitulé, Prix\_Total, Date\_Depense, NomBase\_Justificatif, NouveauNom\_Justificatif, Extensions, Chemin. Le Demandeur est la personne faisant la demande de remboursement pour la dépense effectuée pendant la mission. L’Intitulé permet d’expliquer la nature de cette dépense. Le Prix\_Total correspond au montant que le demandeur demande à se faire rembourser. Par exemple, s'il a pris trois repas de midi, le Prix\_Total correspond à la somme de ces trois repas. Le NomBase\_Justificatif est le nom du fichier joint fourni par le demandeur en tant que justificatif.

Le NouveauNom\_Justificatif correspond au nom de la pièce jointe une fois renommer aléatoirement dans l’application et stocker dans celle-ci.

L’extension correspond à l’extension de la pièce jointe et le chemin correspond au chemin permettant d’accéder au justificatif.

Un frais appartient à une seule mission, et est associé à un seul type de dépense.

En effet un frais peu sois être un repas par exemple, ou une demande de remboursement de billet de train. Si on doit faire rembourser un repas et un billet de train il faudra créer deux frais.

Entité Type\_Depense :

Un type de dépense est identifié par un identifiant unique (ici id\_TypeDepense) et possède les attributs suivants :

Nom\_TypeDepense et Prix\_Unite.

Le nom du type de la dépense correspondra au type de dépense, par exemple : repas midi, nuit dans un hôtel, billet de train…

Le Prix\_Unite correspond au prix à l’unité d’une dépense, par exemple le prix d’une nuit dans un hôtel, d’un kilomètre, d’un repas de midi…

Un type de dépense est rattachée à 0 ou plusieurs frais.

**Succession des actions :**

Dans ce modèle entité-association on retrouve donc un utilisateur qui va partir en missions dans une ville. Cette mission va avoir un statut qui va évolué au court du temps en fonction des actions menée par l’utilisateur ou le comptable.

Lors de cette mission l’utilisateur va engendrer différents frais, dont il pourra demander le remboursement en contrepartie d’un justificatif. Ces frais sont rattachés à un type de dépense qui permet de les identifiés et de définir à quelle catégorie de dépense ils appartiennent.

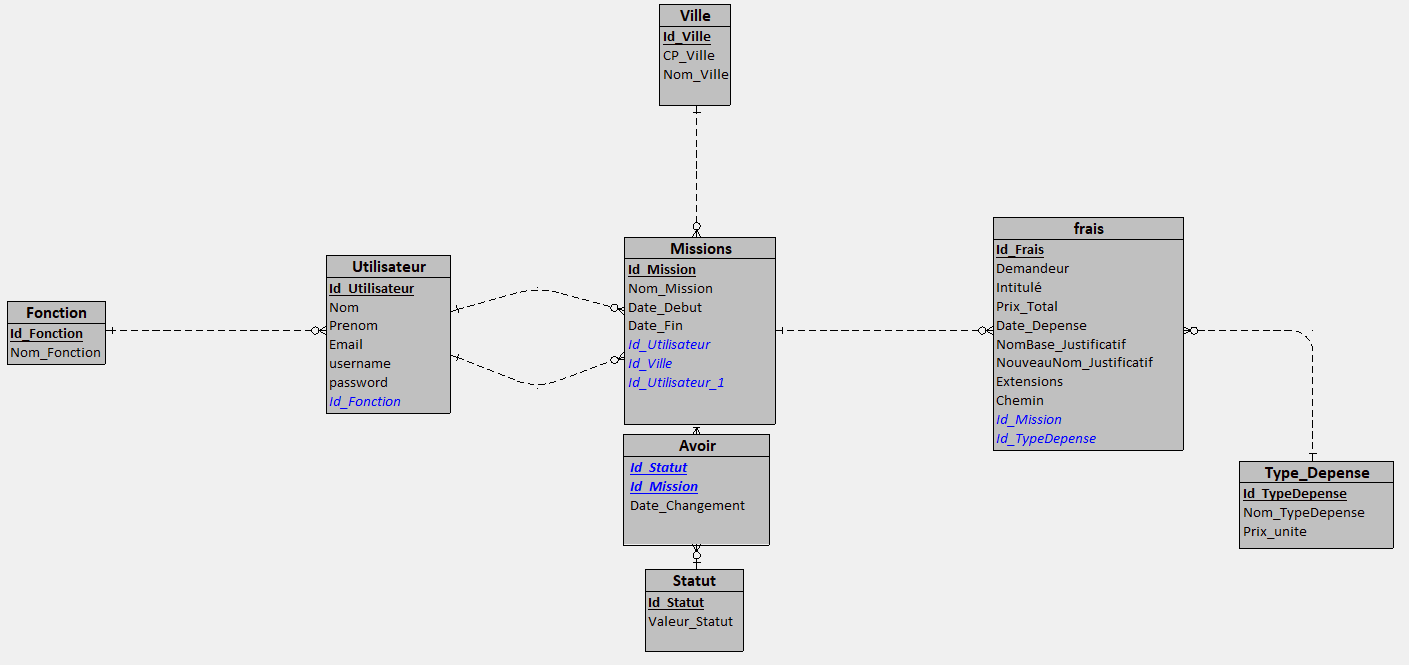
**Les cardinalités :**

* Un utilisateur peut une seule fonction
* Une fonction est rattaché à 0 ou plusieurs utilisateur
* Une mission est créée ou validée par 1 seul utilisateur
* Un utilisateur peut créer une ou plusieurs missions
* Un utilisateur peut valider une ou plusieurs missions
* L’utilisateur qui crée et qui valide la mission ne peut pas être le même grâce à la contrainte d’exclusivité
* Une mission à pour destination une seule ville.
* Une ville peut accueillir 0 ou plusieurs missions.
* Une mission à un ou plusieurs statuts.
* Un statut est rattaché à 0 ou plusieurs missions.
* Une mission engendre 0 ou plusieurs frais.
* Un frais appartient à une seule mission.
* Un frais est associé à un seul type de dépense.
* Un type de dépense est associé à 0 ou plusieurs frais.

## 

## **La modélisation logique**

**Voici le modèle logique des données relationnel schématiser :**



**Voici le modèle logique des données relationnel contextuelle :**

**Type\_Depense = (Id\_TypeDepense *COUNTER***, Nom\_TypeDepense *VARCHAR(50)*, Prix\_unite *DECIMAL(4,2)***);**

**Statut = (Id\_Statut *COUNTER***, Valeur\_Statut *VARCHAR(100)***);**

**Ville = (Id\_Ville *COUNTER***, CP\_Ville *INT*, Nom\_Ville *VARCHAR(100)***);**

**Fonction = (Id\_Fonction *VARCHAR(50)***, Nom\_Fonction *VARCHAR(50)***);**

**Utilisateur = (Id\_Utilisateur *COUNTER***, Nom *VARCHAR(30)*, Prenom *VARCHAR(30)*, Email *VARCHAR(100)*, username *VARCHAR(20)*, password *VARCHAR(255), #Id\_Fonction***);**

**Missions = (Id\_Mission *COUNTER***, Nom\_Mission *VARCHAR(250)*, Date\_Debut *DATE*, Date\_Fin *DATE, #Id\_Utilisateur, #Id\_Ville, #Id\_Utilisateur\_Comptable***);**

**frais = (Id\_Remboursement *COUNTER***, Demandeur *VARCHAR(30)*, Intitulé *VARCHAR(200)*, Prix\_Total *DECIMAL(15,2)*, Date\_Depense *DATE*, NomBase\_Justificatif *VARCHAR(150)*, NouveauNom\_Justificatif *VARCHAR(150)*, Extensions *VARCHAR(10)*, Chemin *VARCHAR(300), #Id\_Mission, #Id\_TypeDepense***);**

**Avoir = (*#Id\_Statut, #Id\_Mission***, Date\_Changement *DATE***);**

**Justification du niveau de normalisation du modèle relationnel :**

Mon modèle relationnel respecte le premier niveau de normalisation (1FN), en effet tous ses attributs possèdent des valeurs atomiques, par exemple dans la table Utilisateur, le nom et prénom sont enregistrés dans deux colonnes différentes. De plus les valeurs des données d’une colonne sont identiques, par exemple dans la table Type\_Depense, la propriété Prix\_unite est toujours en euro.

Mon modèle respecte le deuxième niveau de normalisation (2FN), en effet les propriétés de chaque table sont identifiées par la clé primaire de la table en question. Par exemple chaque utilisateur de la table Utilisateur est identifié par la clé primaire Id\_Utilisateur. Il en est de même pour la ville ou chaque ville est identifié par la clé primaire Id\_Ville.

Les propriétés des tables Missions et Ville sont séparées, car sinon les propriétés auraient été identifié par le nom de la ville et celui de la mission ce qui aurai fait un clé primaire composite, et cela ne respecte pas le deuxième niveau de normalisation. Il n’y a donc pas de dépendance partielle.

Mon modèle relationnel respecte le troisième niveau de normalisation (3FN), en effet aucune propriété hors clé primaire ne dépend d’une autre propriété hors clé primaire. Il n’y a donc pas de dépendance transitive.

Mon modèle relationnel respecte le niveau de normalisation BOYCE CODD, en effet aucune propriété ne détermine la clé primaire, et toutes les propriétés sont dépendantes de la clé primaire. Cela est valable pour chacune des tables de mon modèle de données relationnel.

## 

## **Script de création de la base de données**

CREATE TABLE Type\_Depense(

Id\_TypeDepense COUNTER,

Nom\_TypeDepense VARCHAR(50),

Prix\_unite DECIMAL(4,2),

PRIMARY KEY(Id\_TypeDepense)

);

CREATE TABLE Statut(

Id\_Statut COUNTER,

Valeur\_Statut VARCHAR(100),

PRIMARY KEY(Id\_Statut)

);

CREATE TABLE Ville(

Id\_Ville COUNTER,

CP\_Ville INT,

Nom\_Ville VARCHAR(100),

PRIMARY KEY(Id\_Ville)

);

CREATE TABLE Fonction(

Id\_Fonction VARCHAR(50),

Nom\_Fonction VARCHAR(50),

PRIMARY KEY(Id\_Fonction)

);

CREATE TABLE Utilisateur(

Id\_Utilisateur COUNTER,

Nom VARCHAR(30),

Prenom VARCHAR(30),

Email VARCHAR(100),

username VARCHAR(20),

password VARCHAR(255),

Id\_Fonction VARCHAR(50) NOT NULL,

PRIMARY KEY(Id\_Utilisateur),

FOREIGN KEY(Id\_Fonction) REFERENCES Fonction(Id\_Fonction)

);

CREATE TABLE Missions(

Id\_Mission COUNTER,

Nom\_Mission VARCHAR(250),

Date\_Debut DATE,

Date\_Fin DATE,

Id\_Utilisateur INT NOT NULL,

Id\_Ville INT NOT NULL,

Id\_Utilisateur\_1 INT NOT NULL,

PRIMARY KEY(Id\_Mission),

FOREIGN KEY(Id\_Utilisateur) REFERENCES Utilisateur(Id\_Utilisateur),

FOREIGN KEY(Id\_Ville) REFERENCES Ville(Id\_Ville),

FOREIGN KEY(Id\_Utilisateur\_comptable) REFERENCES Utilisateur(Id\_Utilisateur)

);

CREATE TABLE frais(

Id\_Frais COUNTER,

Demandeur VARCHAR(30),

Intitulé VARCHAR(200),

Prix\_Total DECIMAL(15,2),

Date\_Depense DATE,

NomBase\_Justificatif VARCHAR(150),

NouveauNom\_Justificatif VARCHAR(150),

Extensions VARCHAR(10),

Chemin VARCHAR(300),

Id\_Mission INT NOT NULL,

Id\_TypeDepense INT NOT NULL,

PRIMARY KEY(Id\_Frais),

FOREIGN KEY(Id\_Mission) REFERENCES Missions(Id\_Mission),

FOREIGN KEY(Id\_TypeDepense) REFERENCES Type\_Depense(Id\_TypeDepense)

);

CREATE TABLE Avoir(

Id\_Statut INT,

Id\_Mission INT,

Date\_Changement DATE,

PRIMARY KEY(Id\_Statut, Id\_Mission),

FOREIGN KEY(Id\_Statut) REFERENCES Statut(Id\_Statut),

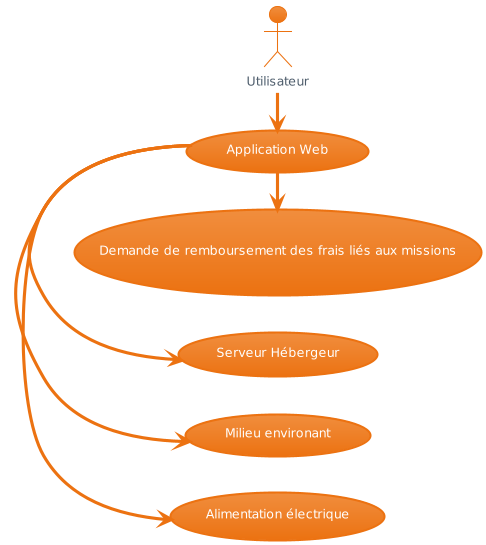
FOREIGN KEY(Id\_Mission) REFERENCES Missions(Id\_Mission)

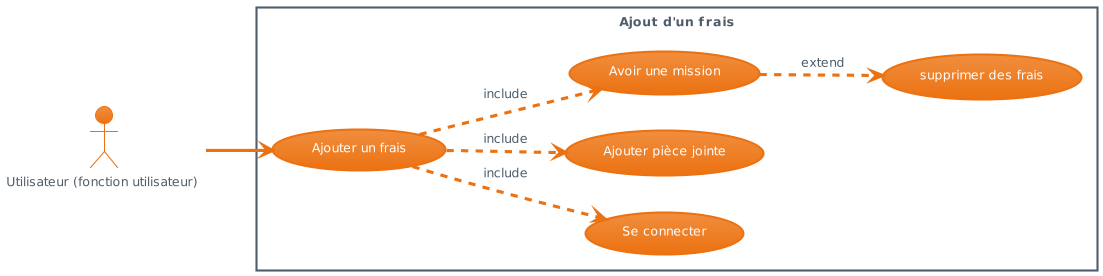
);

## 

## **Modélisation UML**

**Diagramme de contexte :**



**Diagramme cas d’utilisation d’Ajout d’un frais :**

Objectif :

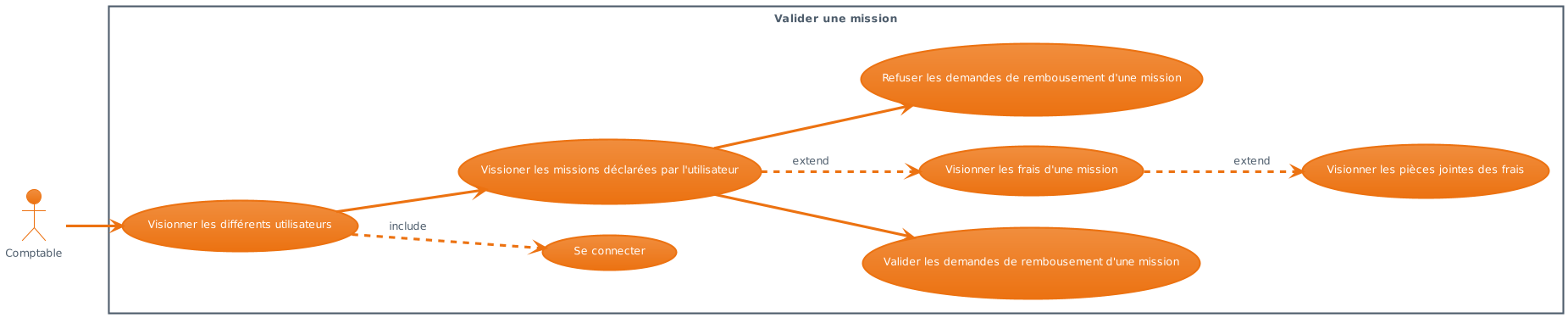
Le but de ce diagramme d’utilisation est de permettre de visualiser comment un utilisateur peut via le site internet, ajouter les frais qu’il a effectué durant la mission afin de demander le remboursement de celle-ci.

Prérequis :

Afin de soumettre une demande de remboursement, l'utilisateur doit disposer d'une connexion Internet pour accéder au site web et se connecter en utilisant son identifiant et son mot de passe. Cette démarche contribue à garantir la sécurité de l'accès au site. De plus, pour inclure des frais, il est nécessaire que l'utilisateur ait préalablement ajouté la mission pour laquelle il souhaite enregistrer ces frais.

Scénario :

Lorsque le visiteur accède au site web, la première étape est de se connecter en utilisant son identifiant et son mot de passe. Sans cette démarche, il ne sera pas possible d'effectuer l'ajout d'un frais sur le site. En premier lieu, il doit créer une mission à laquelle il souhaite associer le frais. Une fois cette étape accomplie, il peut cliquer sur la mission pour y accéder. À ce stade, il a la possibilité d'ajouter un frais à la mission en fournissant obligatoirement une pièce justificative de la dépense.

**Diagramme cas d’utilisation de validation d’une mission :**

Objectif :

Le diagramme d'utilisation a pour objectif de fournir une représentation visuelle du processus par lequel un comptable peut observer les divers utilisateurs et leurs missions. Il illustre également la manière dont le comptable peut approuver ou rejeter les demandes de remboursement associées à ces missions.

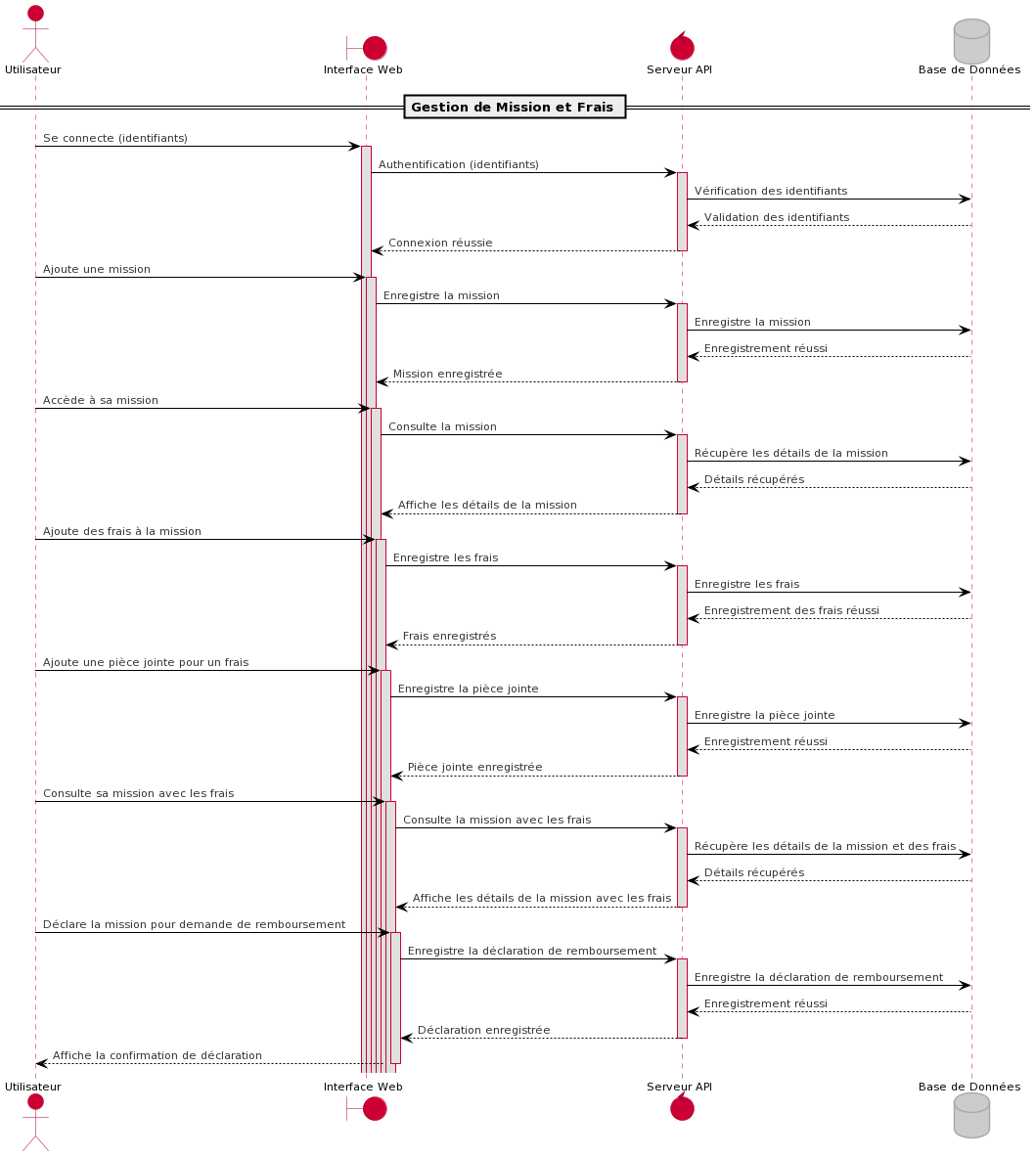
Prérequis :

Afin que le comptable puisse effectuer ces différentes actions il faut qu’il dispose d’une connexion internet et qu’il se connecte via son identifiant et son mot de passe. Cette démarche contribue à garantir la sécurité de l'accès au site.

Scénario :

Quand le comptable arrive sur le site web il devra dans un premier temps se connecter via son identifiant et son mot de passe. Sans cette étape, l'accès au site et à ses fonctionnalités est impossible. Une fois connecté, il peut explorer la liste des utilisateurs et choisir un utilisateur spécifique. En sélectionnant cet utilisateur, le comptable peut examiner les missions qu'il a déclarées pour solliciter un remboursement. En accédant à une mission particulière, il a la possibilité de consulter les frais associés, télécharger les pièces justificatives et prendre la décision de valider ou de refuser la demande de remboursement associée à cette mission.

**Diagramme de séquence de création mission jusqu’à déclaration :**



**Diagramme de séquence de validation d’une demande de remboursement par un comptable :**

