Cahier des charges

INTRANET GSB

Entreprise :	*GSB Laboratoire*
Réalisateur :	*Ilyas MOHETNA*
Date du rapport :	*20/09/2023*

Table des matières

1) Historique du document :

Version	Date de mise à jour	Auteur	Parties modifiées	Description
V1.0	20/09/2023	Ilyas MOHETNA	Tout le document	La création du document
V1.1	01/10/2023	Ilyas MOHETNA	5) Extension Possible	L'ajout de suppression de frais dans le projet actuel
V1.2	15/10/2023	Ilyas MOHETNA	Les deux parties 6) 7)	L'ajout de la modélisation conceptuelle et la modélisation logique
V1.3	20/10/2023	Ilyas MOHETNA	8) Script de BDD	La création du script de base de données
V1.4	25/10/2023	Ilyas MOHETNA	8) Script de BDD	L'ajout d'indexation des attributs clés
V1.5	27/10/2023	Ilyas MOHETNA	9) A)	L'ajout de diagramme de contexte
V1.6	28/10/2023	Ilyas MOHETNA	9) B)	L'ajout des diagrammes de cas d'utilisations
V1.7	29/10/2023	Ilyas MOHETNA	9) C)	L'ajout des diagrammes de séquence des cas d'utilisations

2) Contexte du projet :

Née de la fusion de deux titans de l'industrie pharmaceutique, le laboratoire Galaxy Swiss Bourdin représente une force majeure dans le paysage pharmaceutique mondial. D'un côté, nous avons Galaxy, un géant américain reconnu pour son expertise dans le traitement des maladies virales, notamment le SIDA et les hépatites. De l'autre, Swiss Bourdin, un conglomérat européen résultant lui-même de la fusion de trois laboratoires, est renommé pour ses médicaments conventionnels.

En 2009, ces deux entités ont décidé de joindre leurs forces, donnant naissance à Galaxy Swiss Bourdin, un leader mondial du secteur pharmaceutique. Bien que l'entité européenne de Galaxy Swiss Bourdin ait choisi Paris comme siège administratif, la maison mère de cette multinationale se situe à Philadelphie, aux États-Unis.

3) Le domaine étudié :

a) Domaine de l'entreprise :

Le domaine d'étude se focalise sur le secteur dynamique et complexe de l'industrie pharmaceutique, en mettant un éclairage particulier sur le laboratoire Galaxy Swiss Bourdin (GSB). Originaire de la fusion entre Galaxy, un spécialiste reconnu dans le traitement des maladies virales, et Swiss Bourdin, un conglomérat travaillant sur des médicaments conventionnels, GSB représente une convergence unique de compétences et d'expertises.

Au-delà de sa genèse, l'étude s'intéresse aux interactions de GSB avec les praticiens, notamment à travers les visites médicales. Ces rencontres, essentielles pour la promotion des produits de GSB, sont souvent scrutées pour leur potentiel manque de transparence. L'analyse se penche également sur les procédures post-visites, comme la rédaction de rapports détaillés et la déclaration des frais associés par les employés.

Ainsi, ce domaine d'étude offre un aperçu approfondi des opérations, des défis et des opportunités auxquels GSB est confronté au sein du paysage pharmaceutique contemporain.

b) Domaine de développement :

Dans le cadre du développement d'une solution technique pour l'ensemble de problème informatique de GSB , une application web un ensemble d'outils sera utiliser pour le développement :

- Langage de programmation et framework :
  - PHP 8.0
  - Laravel 10
  - HTML / CSS
  - Javascript
  - JQuery
  - cURL
- Serveur et administrateur de base de donnée :
  - Maria DB 10.4.24
  - PHPMyAdmin 5.2.0
- Outil de versionning :
  - Github 3.10.0

- Serveur Web :
    - Développement :

Développement sera fait sous un ordinateur windows avec un serveur Apache le biais de XAMPP. Ainsi la mise en place de PHP et MariaDB.

    - Production :

Le déploiement de l'application web sera fait sous un serveur Linux dont un serveur web Apache avec les même caractéristique définis dans la cahier des charges (version des outils, langage installées …)

La sélection des outils pour le développement de la solution pour GSB s'est basée sur la combinaison de robustesse, d'efficacité et de flexibilité. Le choix de PHP 8.0 et du Framework Laravel 10 repose sur leur réputation d'être puissants pour le développement d'applications web tout en offrant une grande facilité d'utilisation et une large communauté de soutien. HTML/CSS et Javascript, combinés à JQuery, garantissent une interface utilisateur réactive et interactive. L'utilisation de cURL facilite la communication avec d'autres services via HTTP, ce qui est essentiel pour une application intégrée. En ce qui concerne la gestion des données, MariaDB 10.4.24 est reconnue pour sa rapidité et sa fiabilité, et PHPMyAdmin 5.2.0 offre une interface intuitive pour l'administration de la base de données. Le choix de Github 3.10.0 pour la gestion de version assure une collaboration fluide entre les développeurs et un suivi clair des modifications. Enfin, l'utilisation de XAMPP pour le développement local permet une simulation précise de l'environnement de production, tandis que le déploiement sur un serveur Linux garantit stabilité et performance pour l'application en production.

4) Le fonctionnement du système :

a) Les modules :

Pour résoudre l'ensemble de problèmes rencontré par le laboratoire GSB une solution web sera mise en place sous forme d'un intranet qui va regrouper l'ensemble d'application métier qui seront à destination des collaborateurs. Pour construire l'intranet 5 modules principales ont été mise en place pour de divers problèmes techniques rencontré :

- La gestion des visites
- La gestion des frais
- La réservation des salles
- Le covoiturage
- Le paramétrage et l'administration

b) L'accès à l'intranet :

L'accès à l'intranet sera à travers une combinaison **utilisateur** , **mot de passe** et destinée à chaque employeur dans l'entreprise notamment : Les visiteurs , Les comptables , Les délégués régionaux , Les responsables desecteurs.

c) Le périmètre du système :

- La gestion des visites :
    - La déclaration d'une visite
    - La modification d'une visite
    - La clôture d'une visite
    - La création d'un rapport de la visite
    - La notification du délégué lors de la création et la fin d'une visite
    - La génération automatique d'un fichier PDF ordre de mission
    - Page d'ensemble pour le suivi / la recherche avec filtres des anciennes/nouvelles visites.

- La gestion des frais :
    - Partie visiteur :
        - La création d'une note de frais (Hôtel, Train, Dépenses hors forfait …) en la liant à une visite.
        - La suppression d'une note de frais
        - Page d'ensemble pour le suivi / la recherche avec filtres de toutes notes de frais.
    - Partie comptable :
        - L'accès au tableau d'ensemble de notes de frais de tous les visiteurs
        - L'approbation, le refus d'une note de frais
        - L'accès à l'ensemble de détail d'une note de frais (montant, nature , quantité , date , justificatives)

- La réservation des salles :
    - Recherche rapide de disponibilité d'une salle
    - Dans le cas si la salle est disponible une possibilité de réservation sera proposé
    - Durant la réservation ; une liste de choix (Repas demandé, Projecteur, Accès à internet) sera proposé autres que les informations de bases (Date de réservation , Motif de réservation , Nombre total de personnes présentes)
    - La confirmation par mail après la réservation
    - Page d'ensemble pour le statut de réservation de toutes les salles
- Le covoiturage :
    - Le module sera accessible à tous collaborateur au sein du groupe GSB.
    - Possibilité de proposé des trajets avec le véhicule personnel de l'employer sous condition de la délivrance d'un permis de conduite valide et la carte grise et le pré ajout du véhicule soit avec le véhicule de service de l'entreprise.
    - La possibilité de définir le trajet et les arrêts desservis
    - La possibilité de supprimer une offre de covoiturage
    - La possibilité de recherche approfondi des offres de covoiturages des autres employés.
    - La possibilité d'accès au détail de réservation
    - L'accès au détail de l'offre et voir les personnes qui ont réservé le trajet.
    - Les notifications mail
    - L'intégration du module directement dans le cas de création d'une visite
- Le paramétrage et l'administration
    - La gestion du personnel :
        - L'ajout d'un employé en désignant sa fonction et d'où ses droits d'accès.
        - La suppression d'un employé
    - La gestion de frais :
        - La configuration des montants de forfaits
    - La réservation de salles :
        - L'ajout de bâtiment sous une région
        - L'ajout de salles sous un bâtiment

- Le covoiturage :
  - L'ajout de véhicule de service

- Configuration générale :
  - L'ajout de famille de médicaments
  - L'ajout de médicament sous une famille

5) Extensions possibles :

- Création d'un model statistiques (Visites par mois, Suivi de dépenses, Ratio des familles de médicaments …)
- Modification d'une note de frais sous conditions (elle ne soit pas vu par un comptable)
- IA : Détection automatique de détail d'une note de frais à travers un ticket justificative. Par exemple déposer un ticket de péage va engendrer la création automatiquement d'une note de frais avec le montant et la date inclus sur le ticket.

6) La modélisation conceptuelle :

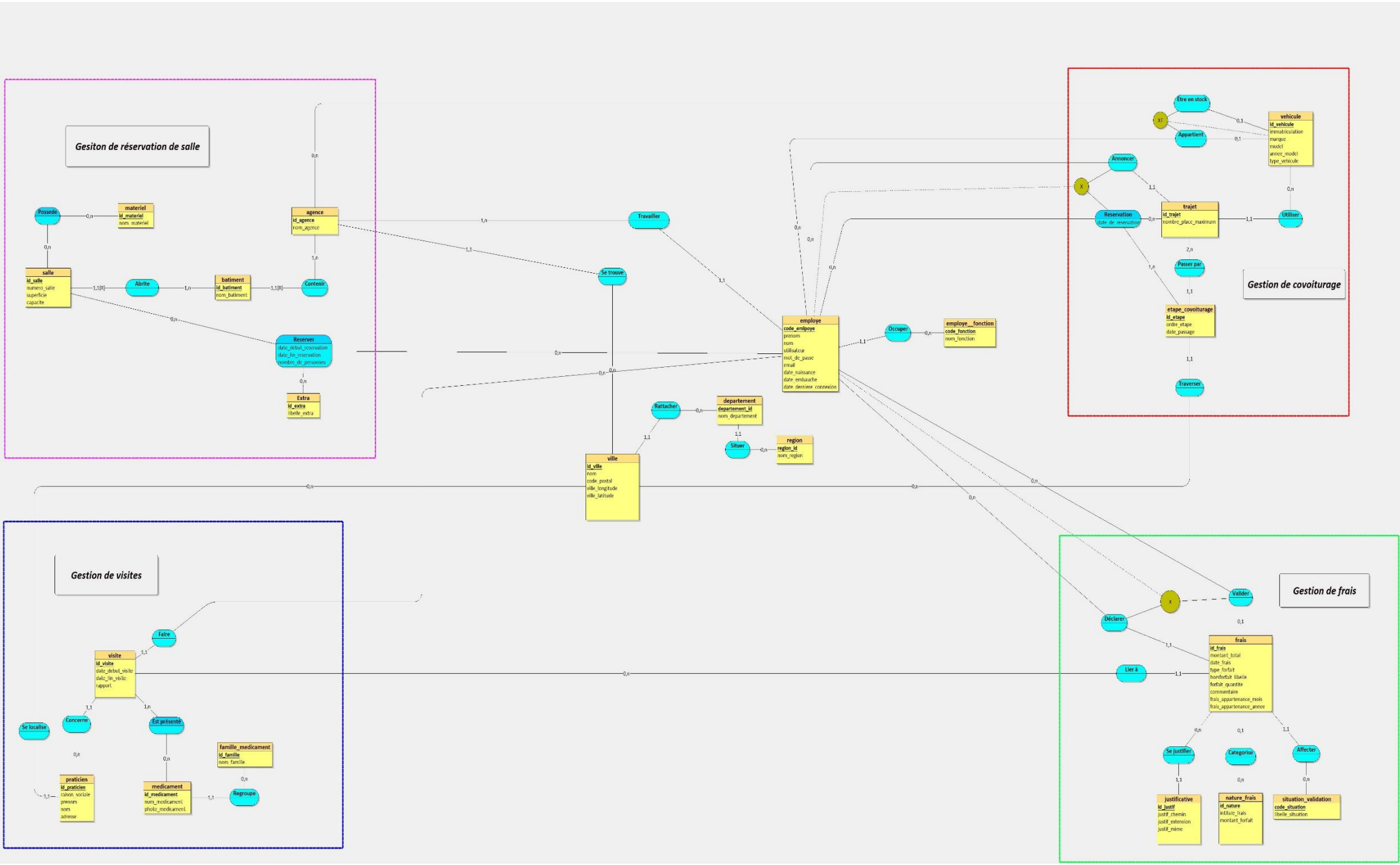
A) Modèle de dictionnaire :

Nom	Type	Nature	Longueur	Identifiant	Exemple de valeur
Employe__employe					
code_employe	E	N	20	OUI	1
prenom	E	A	30	-	GAUTIER
nom	E	A	30	-	DUPONT
utilisateur	E	A	50	-	GAUTIER.DUVALE
mot_de_passe	E	AN	255	-	$2y$10$ny2uulBo6crtt6it1BhQzuiP07lBswYfAOkbKOta8pOkaj/4gY3WC
email	E	AN	100	-	Gautier.duvale@domaine.com
date_naissance	E	DATE	10	-	10/06/1970
date_embauche	E	DATE	10	-	10/06/1995
dernière_connexion	E	DATETIME	19	-	2023-07-15 09:23:45
Employe__fonction					
code_fonction	E	N	20	OUI	1
nom_fonction	E	AN	50	-	Responsable régional
Parametrage__ville					
id_ville	E	N	20	OUI	1
nom	E	A	50	-	Belfort
code_postal	E	N	5	-	90000
ville_longitude	E	N	9	-	6.863849
ville_latitude	E	N	8	-	47.639674
Parametrage__departement					
id_departement	E	N	20	OUI	90
nom_departement	E	A	50	-	TERRITOIRE DE BELFORT
Parametrage__region					
id_region	E	N	10	OUI	1
nom_region	E	A	50	-	Bourgogne-Franche-Comté
Covoiturage__etape					
id_etape	E	N	20	OUI	1
ordre_etape	E	N	11	-	1
date_passage	E	DATETIME	19	-	2023-07-15 09:23:45
Covoiturage__reservation					
id_reservation	E	N	20	OUI	1
date_reservation	E	DATETIME	19	-	2023-07-15 09:23:45
Covoiturage__trajet					
id_trajet	E	N	20	OUI	1
nombre_place_maximum	E	N	11	-	5
Covoiturage__vehicule					
id_vehicule	E	N	20	OUI	1
immatriculation	E	AN	15	-	GT-934-BA
marque	E	AN	15	-	VOLKSWAGEN
model	E	AN	15	-	SCIROCCO
annee_model	E	N	4	-	2014
type_vehicule	E	A	7	-	ENUM deux valeurs possible ‘perso’ et ‘service’
Frais__frais					
id_frais	E	N	20	OUI	1
montant_total	Ca	N	16	-	534,23
date_frais	E	DATE	10	-	2023-07-15
type_forfait	E	A	10	-	ENUM deux valeurs possible ‘forfait’ et ‘horsforfait’
horsforfait_libelle	E	AN	50	-	Ticket d’avion N212
forfait_quantite	Ca	N	11	-	5
commentaire	E	AN	255	-	La somme présente est le pour une double nuitée d’hôtel
appartenance_mois	E	N	2	-	06
appartenance_annee	E	N	4	-	2023
Frais__nature					
id_nature	E	N	20	OUI	1
intitule_frais	E	AN	50	-	Forfait Repas + Nuit Hotel
montant_forfait	Ca	N	16	-	70.00
Frais__justificative					
id_justif	E	N	20	OUI	1
justif_chemin	E	AN	255	-	/storage/document/83JF438DEZKLSDFK234KL3K.pdf
justif_extension	E	AN	5	-	pdf
justif_mime	E	AN	20	-	application/pdf
Frais__situation_validation					

code_situation	E	N	20	OUI	1
libelle_situation	E	A	50	-	EN COURS DE TRAITEMENT
Salle__agence					
id_agence	E	N	20	OUI	1
nom_agence	E	AN	50	-	AGENCE LAB123 PHARMATECH
Salle__batiment					
id_batiment	E	N	20	OUI	1
nom_batiment	E	AN	50	-	BATIMENT PRIMAIRE 2
Salle__salle					
id_salle	E	N	20	OUI	1
nom_salle	E	AN	20	-	B204
Salle__extra					
id_extra	E	N	20	OUI	1
libelle_extra	E	AN	100	-	Jus d'orange
Salle__materiel_type					
id_materiel	N	N	20	OUI	2
nom_materiel	E	AN	50	-	CABLES RJ45 , PROJECTEUR …
Salle__posession_materiel					
id_salle	E	N	20	OUI	1
id_materiel	E	AN	20	OUI	2
Salle__reservation					
id_reservation	E	N	20	OUI	1
date_debut_reservation	E	DATETIME	19	-	2023-07-15 09:23:45
date_fin_reservation	E	DATETIME	19	-	2023-07-15 15:23:45
nombre_de_personnes	E	N	16	-	5
Visite__visite					
id_visite	E	N	20	OUI	1
date_debut_visite	E	DATETIME	19	-	2023-07-15 09:23:45
date_fin_visite	E	DATETIME	18	-	2023-07-15 15:23:45
rapport	E	AN	255	-	Un avis plutôt favorable émis par le praticien concernant …
Visite__praticien					
id_praticien	E	N	20	OUI	1
raison_social	E	AN	50	-	MICHEL BERNANRD, CABINET BERNARD
prenom	E	A	30	-	MICHEL
nom	E	A	30	-	BERNARD
adresse	Co	AN	100	-	26 RUE LA RENAISSANCE
Visite__medicament					
id_medicament	E	N	20	OUI	1
nom_medicament	E	AN	30	-	Acétaminophène
photo_medicament	E	AN	255	-	https://chemin_vers_photo_Acétaminophène.com
Visite__presentation_medicament					
id_visite	E	N	20	OUI	1
id_medicament	E	N	20	OUI	1
Visite__famille_medicament					
id_famille	E	N	20	OUI	2
nom_famille	E	AN	30	-	Analgésiques

B) Le modèle entité association :

a. La présentation du modèle entité association



b) Les explications sur les choix modélisation et cardinalités :

- **Employé (employé, fonction) :**

  Plutôt que de créer des tables séparées pour différents types d'employés (par exemple, "Visiteur" et "Comptable"), nous avons opté pour une seule entité "Employé". Cette approche permet de stocker l'ensemble des données sur tous les employés qui auront accès à l'intranet, évitant ainsi les redondances inutiles de données et garantissant l'intégrité des informations.

  Les cardinalités :

  - Un employé occupe une seule fonction
  - Un employé travaille dans une seule agence
  - Un employé peut avoir aucune ou plusieurs véhicules personnels
  - Un employé peut annoncer aucune ou plusieurs trajets de covoiturage
  - Un employé peut effectuer aucune ou plusieurs réservations de trajet
  - Un employé (visiteur) peut déclarer aucun ou plusieurs frais de mission
  - Un employé (comptable) peut valider aucune ou plusieurs notes de frais
  - Un employé (visiteur) peut faire aucune ou plusieurs visites
  - Un employé peut réserver aucune ou plusieurs salles dans tous les agences des différents départements ou régions.

- **Paramétrage (Ville, Département, Région) :**

  Le paramétrage pour la gestion des zones géographiques a été réalisé en tenant compte de la réalité, dans le but d'offrir un contrôle et une granularité optimaux. Notre choix s'est porté sur l'utilisation de villes en tant que clés étrangères, au lieu d'un champ texte libre, afin d'éviter les redondances de données, d'assurer la précision et d'éliminer les erreurs typographiques. Cette approche garantit également, à des fins statistiques, une présentation sans doublons des informations géographiques.

  Pour alimenter les données de ces trois entités, nous prévoyons de tirer parti de l'API fournie par l'État SIREN. Cette source de données nous permettra d'obtenir un ensemble complet d'informations, y compris la longitude et la latitude des villes. Ces coordonnées géographiques sont essentielles pour le développement de fonctionnalités de cartographie, offrant la possibilité de géolocaliser avec précision les villes sur une carte.

  Ce choix de modélisation garantit une base de données géographique fiable et précise, tout en facilitant l'intégration de données externes provenant de sources officielles, ce qui contribue à la qualité et à la cohérence des informations géographiques dans notre système.

  Les cardinalités :

  - Une ville est rattachée à un département
  - Un département est situé dans une région
  - Une région contient plusieurs départements
  - Un département contient plusieurs villes

- **Salle (Salle, Bâtiment, Agence, Réservation, Possession Matériel, Type Matériel, Extra)**

  Le choix des entités "Salle", "Bâtiment" et "Agence" a été délibéré pour refléter au mieux la structure d'une agence et pour simplifier la création future de salles sous de nouveaux bâtiments ou agences. Cette modélisation permet de suivre de manière organique la hiérarchie des locaux au sein de l'entreprise, en garantissant une cohérence dans la gestion des salles.

  L'entité "Matériel" a été introduite pour éviter les redondances de données. Elle permet de répertorier l'ensemble du matériel disponible, tout en offrant la possibilité de l'attribuer à différentes salles. Cela est réalisé grâce à l'association "Possède" (Possession Matériel), qui assure que chaque salle peut disposer de

plusieurs équipements, ce qui optimise la gestion des ressources.

Pour gérer les réservations de salles, l'association "Réserver" a été créée. Elle joue un rôle essentiel en stockant les valeurs qui identifient la date de début de la réservation, la date de fin, ainsi que le nombre de personnes prévues. Cette association est également porteuse d'informations essentielles pour le traitement des extras. Par exemple, dans l'entité "Extra," des éléments tels que "Café" ou "Jus d'orange" peuvent être définis en tant que formules générales. La réservation permet de faire des choix spécifiques parmi ces options pour s'assurer que la quantité nécessaire est disponible, par exemple, en fonction du nombre de participants prévus.

Cette approche de modélisation permet de gérer efficacement les ressources liées aux locaux et aux équipements, tout en offrant une flexibilité pour créer, réserver et gérer des salles de manière intuitive au sein de l'entreprise.

Les cardinalités :

- Une salle est abritée dans un seul bâtiment
- Une agence contient un ou plusieurs bâtiments
- Un bâtiment appartient à une seule agence
- Un bâtiment contient une ou plusieurs salles
- Une salle ne possède aucun ou plusieurs matériels
- Un matériel peut être posséder par aucune ou par plusieurs salles
- Une salle peut être réserver plusieurs fois (par un employé) « Sous conditions de différenciation de plage horaires »
- Les extras peuvent êtres réserver dans aucune ou dans plusieurs réservations.

- **Visite (Visite, Praticien, Médicament, Famille Médicament, Médicament présenté) :**

Dans le module "Visite", plusieurs choix de modélisation ont été délibérément effectués pour optimiser le développement et la gestion des données, ainsi que pour permettre une utilisation efficace des informations dans d'autres parties du système.

Entité "Praticien" Séparée : Le choix a été fait de créer une entité distincte pour les "Praticiens". Cette séparation vise à simplifier la gestion des informations sur les praticiens, en les distinguant clairement des employés. Cela permet également de faciliter le traitement des données liées aux visites médicales.

Entité "Visite" Centrale : Plutôt que d'utiliser une simple association entre les employés et les praticiens, une entité "Visite" a été créée. Cette approche centralise les informations relatives aux visites, offrant une plus grande flexibilité pour gérer les aspects spécifiques de chaque visite et simplifiant le suivi des données dans d'autres parties du système.

Entités "Médicament" et "Famille Médicament" pour une Meilleure Visibilité : Les entités distinctes "Médicament" et "Famille Médicament" ont été introduites pour offrir une visibilité plus claire lors de l'utilisation des données. Cela permet également de faciliter des analyses statistiques plus précises. Par exemple, il est possible d'étudier chaque branche de médicament de manière distincte pour obtenir des informations spécifiques.

Association "Médicament Présenté" : L'association "Médicament Présenté" a été créée pour permettre à un visiteur de présenter plusieurs médicaments lors de la même visite. Cette association facilite la gestion des informations sur les médicaments présentés lors des visites médicales et offre une souplesse pour les enregistrements complexes.

Les cardinalités :

- Une visite peut être faite par un employé (visiteur)
- Une visite n'engendre aucun ou plusieurs frais de mission
- Lors d'une visite on peut présenter un ou plusieurs médicaments

- Une visite concerne qu'un seul praticien
- Un médicament est présenté dans plusieurs visites
- Un médicament appartient à une seule famille de médicament
- Une famille de médicament ne comporte aucun ou comporte plusieurs médicaments
- Un praticien peut être visiter durant plusieurs visite (sous condition de plage horaire différentes)
- Un praticien se localise dans une ville

- **Frais (Frais, Justificative, Nature frais, Situation Validation) :**

  Le choix de passer par une table "Frais" commune a été fait pour éviter les doublons et centraliser les informations du même module. Cette approche gère à la fois les "Frais Forfait" et les "Frais Hors Forfait" en distinguant le champ "type_forfait" : les "Frais Forfait" auront des champs remplis tandis que les "Frais Hors Forfait" auront des champs nuls, et vice versa.

  Pour gérer la validation des frais par le comptable et les retards, deux champs, "frais_appartenance_mois" et "frais_appartenance_annee", ont été ajoutés. Ils servent de pivot pour indiquer la période à laquelle les frais sont attribués. Ces champs reflètent initialement les mêmes valeurs que la date des frais, mais si un retard survient dans la fourniture de justificatifs, le mois peut changer pour reporter le frais au mois suivant.

  Concernant les justificatifs, une entité distincte a été créée pour permettre à un frais d'avoir plusieurs justificatifs. Une table associée stocke l'ensemble des fichiers et est liée à travers l'identifiant du frais.

  L'entité "Nature Frais" contient des forfaits prédéfinis avec des montants fixés par l'administration. Chaque forfait a son montant fixe, qui est multiplié par le champ "frais.forfait_quantite" pour calculer le montant total dans la même table.

  Enfin, l'entité "Situation Validation" contient les différents statuts de validation attribués aux frais. Ces statuts évoluent tout au long du processus de traitement, par exemple, "Créé", "En cours de traitement", "Validé", "En cours de remboursement", "Annulé", etc. Cette structure permet au comptable et au visiteur d'avoir des pages de suivi pour leurs frais, offrant une gestion efficace des processus de validation et de remboursement.

  Les cardinalités :
  - Un frais est lié à une visite obligatoirement sans visite on ne peut pas déclarer de frais
  - Un frais est créé par un seul employé (visiteur)
  - Un employé (visiteur) peut créer un ou plusieurs frais
  - Un frais est validé par un employé (comptable)
  - Un justificative appartient à un seul frais
  - Un nature frais peut être la catégorie d'aucun ou de plusieurs frais
  - Une situation de validation peut être attribuer à plusieurs frais
  - Un frais peut avoir aucun ou plusieurs justificative (Forfait : Non obligatoire, Hors forfait : Obligatoire)
  - Un frais peut avoir 0 ou 1 nature frais (0 si le cas d'un frais hors forfait)
  - Un frais a une seule situation de validation

- **Covoiturage (Trajet, Véhicule, Etape Covoiturage, Réservation)**

  Annonces de Trajets et Étapes de Covoiturage : Le choix a été fait de diviser le processus de covoiturage en annonces de trajets et étapes de covoiturage distinctes. Les employés peuvent réserver des trajets en sélectionnant les étapes qui leur conviennent. Cette approche permet une flexibilité maximale pour les utilisateurs du système.

  Entité "Véhicule" Complète : L'entité "Véhicule" a été créée pour regrouper tous les types de véhicules, à la fois ceux utilisés à des fins professionnelles et personnels. La distinction est basée sur le champ "type_vehicule". Les employés peuvent effectuer une ou plusieurs réservations de véhicules dans différentes plages horaires, tout en garantissant que l'exclusion est respectée, ce qui signifie que la personne qui dépose

l'annonce ne peut pas réserver son propre trajet.

Ces choix de modélisation visent à rendre le module de covoiturage intuitif et flexible pour les utilisateurs, en facilitant la recherche, la réservation de trajets et la gestion des véhicules, tout en respectant les contraintes d'exclusion pour une utilisation équitable du système.

Les cardinalités :

- Un employé peut déposer 0 ou plusieurs trajets de covoiturage
- Un employé peut réserver 0 ou plusieurs trajets de covoiturage sous deux conditions (des plages horaires différentes ainsi qu'il ne soit pas la même personne qui a déposé le trajet)
- Un trajet utilise un seul véhicule obligatoirement
- Un véhicule peut être ou pas utiliser dans un trajet
- Un véhicule peut être en stock dans une agence (Le cas de véhicule de service)
- Un véhicule peut appartenir à un employé (Le cas de véhicule personnel)
- Un trajet contient la cardinalité particulière (2, N) pour bien préciser que le minimum d'un trajet est 2 étapes parce que le choix a été fait d'enregistrer les villes de départ et arrivé comme étape avec un ordre par exemple Belfort départ et Besançon arrivé on aura toujours au moins deux étapes covoiturage dans ce cas Belfort avec un ordre 1 et Besançon avec un ordre 2.
- Une agence peut avoir aucune ou plusieurs voitures de service
- Une étape de covoiturage est traverse obligatoirement une seule ville sur la table ville

7) La modélisation logique

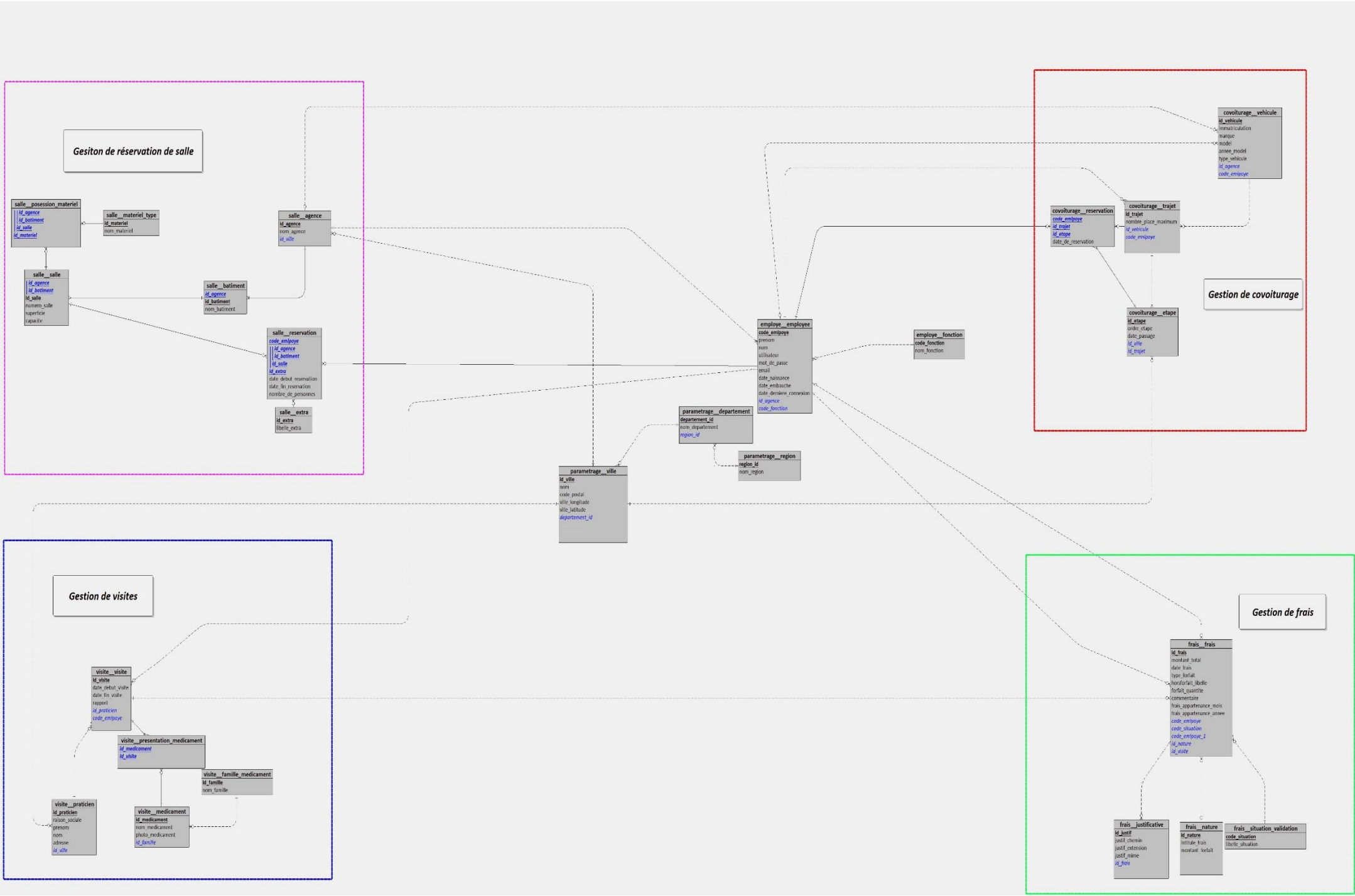
A) Le modèle logique des données relationnel :

Dans le cadre de la conception de notre base de données, On a décidé d'appliquer des normes de normalisation dès l'étape du modèle entité-association. Cette décision découle de ma volonté de garantir la cohérence, l'efficacité et la qualité de notre base de données, tout en facilitant le processus de développement ultérieur. La normalisation des données est une pratique éprouvée qui vise à éliminer les anomalies potentielles, à réduire la redondance des données et à rendre la base de données plus flexible. En conséquence, j'ai veillé à ce que chaque entité et ses attributs soient correctement structurés, en éliminant les données répétées et en établissant des relations claires entre les entités. Cette démarche vise à garantir que notre base de données répondra aux besoins de l'application de manière optimale, tant en termes de performance que d'intégrité des données. Plus de détails sur le processus de normalisation seront abordés dans les paragraphes suivants, permettant ainsi de mieux comprendre comment nous avons optimisé notre modèle logique de données pour atteindre ces objectifs.[Citez votre source ici.]

a. Le modèle textuel

- **parametrage__region = (region_id** , nom_region **);**
- **employe__fonction = (code_fonction** , nom_fonction **);**
- **frais__nature = (id_nature** , intitule_frais , montant_forfait **);**
- **visite__famille_medicament = (id_famille** , nom_famille **);**
- **frais__situation_validation = (code_situation** , libelle_situation **);**
- **salle__extra = (id_extra** , libelle_extra **);**
- **salle__materiel_type = (id_materiel** , nom_materiel **);**
- **parametrage__departement = (departement_id** , nom_departement *, #region_id*);
- **visite__medicament = (id_medicament** , nom_medicament , photo_medicament*, #id_famille*);
- **parametrage__ville = (id_ville** , nom , code_postal, ville_longitude, ville_latitude*, #departement_id*);
- **visite__praticien = (id_praticien** , raison_sociale , prenom , nom , adresse*, #id_ville*);
- **salle__agence = (id_agence** , nom_agence *, #id_ville*);
- **salle__batiment = (*#id_agence*, id_batiment** , nom_batiment **);**
- **employe__employee = (code_emlpoye** , prenom , nom , utilisateur , mot_de_passe , email , date_naissance , date_embauche, date_derniere_connexion *, #id_agence, #code_fonction*);
- **salle__salle = (id_salle** , numero_salle, superficie, capacite**);**
- **visite__visite = (id_visite** , date_debut_visite , date_fin_visite , rapport *, #id_praticien, #code_emlpoye*);
- **frais__frais = (id_frais** , montant_total, date_frais, type_forfait, horsforfait_libelle , forfait_quantite, commentaire , frais_appartenance_mois, frais_appartenance_annee*, #code_emlpoye, #code_situation, #code_emlpoye_1*, #id_nature*, #id_visite*);
- **covoiturage__vehicule = (id_vehicule** , immatriculation , marque , model , annee_model , type_vehicule *, #id_agence*, #code_emlpoye**);
- **frais__justificative = (id_justif** , justif_chemin , justif_extension , justif_mime *, #id_frais*);
- **covoiturage__trajet = (id_trajet** , nombre_place_maximum *, #id_vehicule, #code_emlpoye*);
- **covoiturage__etape = (id_etape** , ordre_etape , date_passage *, #id_ville, #id_trajet*);
- **salle__reservation = (*#code_emlpoye,#id_salle, #id_extra***, date_debut_reservation, date_fin_reservation , nombre_de_personnes **);**
- **visite__presentation_medicament = (*#id_medicament, #id_visite*);**
- **covoiturage__reservation = (*#code_emlpoye, #id_trajet, #id_etape***, date_de_reservation**);**
- **salle__posession_materiel = (*#id_salle, #id_materiel*);**

b. Le modèle schématisé



B) La vérification des types de données et l’ordre des colonnes des clés primaires composées

C) La normalisation

Grâce au modèle entité-association, l'ensemble de nos tables relationnelles répond à la première forme normale (1NF). Cette forme consiste à séparer les données composées de manière que toutes les données soient atomiques. Cela se reflète dans notre conception, car nous avons correctement séparé les noms et prénoms, ainsi que les adresses, villes et codes postaux de chaque employé et praticien. Nous n'avons pas regroupé l'ensemble des données sous un seul attribut. Par exemple, au lieu d'avoir un seul attribut pour le nom et le prénom, nous avons séparé ces informations en deux attributs distincts. De même, pour les adresses, villes et codes postaux, nous avons utilisé des attributs séparés au lieu de tout regrouper en un seul attribut.

En passant ainsi, nos tables respectent la deuxième forme normale (2NF), car chaque attribut non-clé est entièrement fonctionnel en fonction de la clé primaire. Cela signifie que nous n'utilisons dans aucune de nos tables un attribut non-clé pour identifier un enregistrement dans la table. Par exemple, dans la table des véhicules, nous n'utilisons pas l'année en plus de l'identifiant (id_vehicule) pour identifier un véhicule. L'identifiant du véhicule est indépendant de lui-même, et il n'est pas nécessaire de le combiner avec d'autres clés pour identifier un véhicule spécifique.

De plus, nos tables respectent la troisième forme normale (3NF), qui est mondialement reconnue comme étant exigeante et obligatoire. Elle stipule qu'aucun attribut non clé ne peut dépendre de manière transitoire d'un candidat clé. Cela se reflète dans l'ensemble de nos tables, où il n'y a aucune chance de production de répétition ou de redondance de données. Par exemple, notre conception garantit que la table "covoiturage__reservation" ne dépend pas directement des villes (via la table "parametrage__ville"). Si la table "covoiturage__reservation" dépendait directement des villes, cela créerait une dépendance transitive. Par exemple, si la ville d'un employé était stockée dans la table "parametrage__ville", et que la table "covoiturage__reservation" stockait la ville de l'employé, cela créerait une dépendance transitive de la table "covoiturage__reservation" à la table "parametrage__ville". Cependant, notre conception garantit que cela ne se produit pas, car nous utilisons une table "employe__employe" pour stocker des informations sur les employés et une table "parametrage__ville" pour stocker des informations sur les villes. Ainsi, la table "covoiturage__reservation" respecte la 3NF car elle relie uniquement les employés et les étapes de covoiturage, sans dépendance transitive directe à d'autres tables. Cette approche garantit une structure de base de données propre et bien normalisée.

8) Script de création de la base de données

```
DROP TABLE IF EXISTS `covoiturage__etape`;

CREATE TABLE `covoiturage__etape` (
  `id_etape` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `ordre_etape` int(11) NOT NULL,
  `date_passage` datetime NOT NULL,
  `id_trajet` bigint(20) unsigned NOT NULL,
  `id_ville` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_etape`),
  KEY `covoiturage__etape_id_etape_index` (`id_etape`),
  KEY `covoiturage__etape_ordre_etape_index`
(`ordre_etape`),
  KEY `covoiturage__etape_id_trajet_index` (`id_trajet`),
  KEY `covoiturage__etape_id_ville_index` (`id_ville`),
  CONSTRAINT `covoiturage__etape_id_trajet_foreign`
FOREIGN KEY (`id_trajet`) REFERENCES `covoiturage__trajet`
(`id_trajet`),
  CONSTRAINT `covoiturage__etape_id_ville_foreign` FOREIGN
KEY (`id_ville`) REFERENCES `parametrage__ville`
(`id_ville`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `covoiturage__etape` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `covoiturage__reservation`;

CREATE TABLE `covoiturage__reservation` (
  `id_reservation` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `date_de_reservation` datetime NOT NULL,
  `code_employe` bigint(20) unsigned NOT NULL,
  `id_trajet` bigint(20) unsigned NOT NULL,
  `id_etape` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_reservation`),
  KEY `covoiturage__reservation_id_reservation_index`
(`id_reservation`),
  KEY `covoiturage__reservation_code_employe_index`
(`code_employe`),
```

```
  KEY `covoiturage__reservation_id_trajet_index`
(`id_trajet`),
  KEY `covoiturage__reservation_id_etape_index`
(`id_etape`),
  CONSTRAINT
`covoiturage__reservation_code_employe_foreign` FOREIGN
KEY (`code_employe`) REFERENCES `employe__employe`
(`code_employe`),
  CONSTRAINT `covoiturage__reservation_id_etape_foreign`
FOREIGN KEY (`id_etape`) REFERENCES `covoiturage__etape`
(`id_etape`),
  CONSTRAINT `covoiturage__reservation_id_trajet_foreign`
FOREIGN KEY (`id_trajet`) REFERENCES `covoiturage__trajet`
(`id_trajet`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `covoiturage__reservation` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `covoiturage__trajet`;

CREATE TABLE `covoiturage__trajet` (
  `id_trajet` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nombre_place_maximum` int(11) NOT NULL,
  `id_vehicule` bigint(20) unsigned NOT NULL,
  `code_employe` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_trajet`),
  KEY `covoiturage__trajet_id_trajet_index` (`id_trajet`),
  KEY `covoiturage__trajet_id_vehicule_index`
(`id_vehicule`),
  KEY `covoiturage__trajet_code_employe_index`
(`code_employe`),
  CONSTRAINT `covoiturage__trajet_code_employe_foreign`
FOREIGN KEY (`code_employe`) REFERENCES `employe__employe`
(`code_employe`),
  CONSTRAINT `covoiturage__trajet_id_vehicule_foreign`
FOREIGN KEY (`id_vehicule`) REFERENCES
`covoiturage__vehicule` (`id_vehicule`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `covoiturage__trajet` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `covoiturage__vehicule`;

CREATE TABLE `covoiturage__vehicule` (
  `id_vehicule` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `immatriculation` varchar(15) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `marque` varchar(15) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `model` varchar(15) COLLATE utf8mb4_unicode_ci NOT NULL,
  `annee_model` year(4) NOT NULL,
  `type_vehicule` enum('perso','service') COLLATE
utf8mb4_unicode_ci NOT NULL,
  `id_agence` bigint(20) unsigned DEFAULT NULL,
  `code_employe` bigint(20) unsigned DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_vehicule`),
  KEY `covoiturage__vehicule_id_vehicule_index`
(`id_vehicule`),
  KEY `covoiturage__vehicule_id_agence_index`
(`id_agence`),
  KEY `covoiturage__vehicule_code_employe_index`
(`code_employe`),
  CONSTRAINT `covoiturage__vehicule_code_employe_foreign`
FOREIGN KEY (`code_employe`) REFERENCES `employe__employe`
(`code_employe`),
  CONSTRAINT `covoiturage__vehicule_id_agence_foreign`
FOREIGN KEY (`id_agence`) REFERENCES `salle__agence`
(`id_agence`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `covoiturage__vehicule` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `employe__employe`;

CREATE TABLE `employe__employe` (
  `code_employe` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `prenom` varchar(30) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `nom` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `utilisateur` varchar(50) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `mot_de_passe` varchar(255) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `email` varchar(100) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `date_naissance` date NOT NULL,
  `date_embauche` date NOT NULL,
  `derniere_connexion` datetime DEFAULT NULL,
  `id_agence` bigint(20) unsigned NOT NULL,
  `code_fonction` bigint(20) unsigned NOT NULL,
```

```
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`code_employe`),
  KEY `employe__employe_code_employe_index`
(`code_employe`),
  KEY `employe__employe_id_agence_index` (`id_agence`),
  KEY `employe__employe_code_fonction_index`
(`code_fonction`),
  CONSTRAINT `employe__employe_code_fonction_foreign`
FOREIGN KEY (`code_fonction`) REFERENCES
`employe__fonction` (`code_fonction`),
  CONSTRAINT `employe__employe_id_agence_foreign` FOREIGN
KEY (`id_agence`) REFERENCES `salle__agence` (`id_agence`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `employe__employe` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `employe__fonction`;

CREATE TABLE `employe__fonction` (
  `code_fonction` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_fonction` varchar(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`code_fonction`),
  KEY `employe__fonction_code_fonction_index`
(`code_fonction`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `employe__fonction` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `frais__frais`;

CREATE TABLE `frais__frais` (
  `id_frais` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `montant_total` decimal(16,2) NOT NULL,
  `date_frais` date NOT NULL,
  `type_forfait` enum('forfait','horforfait') COLLATE
utf8mb4_unicode_ci NOT NULL,
  `horsforfait_libelle` varchar(50) COLLATE
utf8mb4_unicode_ci DEFAULT NULL,
  `forfait_quantite` int(11) DEFAULT NULL,
  `commentaire` varchar(255) COLLATE utf8mb4_unicode_ci
DEFAULT NULL,
  `appartenance_mois` char(2) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `appartenance_annee` year(4) NOT NULL,
  `id_nature` bigint(20) unsigned DEFAULT NULL,
  `code_situation` bigint(20) unsigned NOT NULL,
  `code_employe_comptable` bigint(20) unsigned DEFAULT
NULL,
  `id_visite` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_frais`),
  KEY `frais__frais_id_frais_index` (`id_frais`),
  KEY `frais__frais_id_nature_index` (`id_nature`),
  KEY `frais__frais_code_situation_index`
(`code_situation`),
  KEY `frais__frais_code_employe_comptable_index`
(`code_employe_comptable`),
  KEY `frais__frais_id_visite_index` (`id_visite`),
  CONSTRAINT `frais__frais_code_employe_comptable_foreign`
FOREIGN KEY (`code_employe_comptable`) REFERENCES
`employe__employe` (`code_employe`),
  CONSTRAINT `frais__frais_code_situation_foreign` FOREIGN
KEY (`code_situation`) REFERENCES
`frais__situation_validation` (`code_situation`),
  CONSTRAINT `frais__frais_id_nature_foreign` FOREIGN KEY
(`id_nature`) REFERENCES `frais__nature` (`id_nature`),
  CONSTRAINT `frais__frais_id_visite_foreign` FOREIGN KEY
(`id_visite`) REFERENCES `visite__visite` (`id_visite`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `frais__frais` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `frais__justificative`;

CREATE TABLE `frais__justificative` (
  `id_justif` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `justif_chemin` varchar(255) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `justif_extension` varchar(5) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `justif_mime` varchar(20) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `id_frais` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_justif`),
  KEY `frais__justificative_id_justif_index`
(`id_justif`),
  KEY `frais__justificative_id_frais_index` (`id_frais`),
```

```
  CONSTRAINT `frais__justificative_id_frais_foreign`
FOREIGN KEY (`id_frais`) REFERENCES `frais__frais`
(`id_frais`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `frais__justificative` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `frais__nature`;

CREATE TABLE `frais__nature` (
  `id_nature` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `intitule_frais` varchar(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `montant_forfait` decimal(16,2) NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_nature`),
  KEY `frais__nature_id_nature_index` (`id_nature`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `frais__nature` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `frais__situation_validation`;

CREATE TABLE `frais__situation_validation` (
  `code_situation` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `libelle_situation` varchar(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`code_situation`),
  KEY `frais__situation_validation_code_situation_index`
(`code_situation`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `frais__situation_validation` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `parametrage__departement`;

CREATE TABLE `parametrage__departement` (
  `departement_id` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_departement` varchar(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `region_id` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`departement_id`),
  KEY `parametrage__departement_departement_id_index`
(`departement_id`),
  KEY `parametrage__departement_region_id_index`
(`region_id`),
  CONSTRAINT `parametrage__departement_region_id_foreign`
FOREIGN KEY (`region_id`) REFERENCES `parametrage__region`
(`region_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `parametrage__departement` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `parametrage__region`;

CREATE TABLE `parametrage__region` (
  `region_id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nom_region` varchar(50) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`region_id`),
  KEY `parametrage__region_region_id_index` (`region_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `parametrage__region` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `parametrage__ville`;

CREATE TABLE `parametrage__ville` (
  `id_ville` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nom` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `code_postal` char(5) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `ville_longitude` decimal(9,6) NOT NULL,
  `ville_latitude` decimal(8,6) NOT NULL,
  `departement_id` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_ville`),
  KEY `parametrage__ville_id_ville_index` (`id_ville`),
  KEY `parametrage__ville_nom_index` (`nom`),
```

```
  KEY `parametrage__ville_code_postal_index`
(`code_postal`),
  KEY `parametrage__ville_departement_id_index`
(`departement_id`),
  CONSTRAINT `parametrage__ville_departement_id_foreign`
FOREIGN KEY (`departement_id`) REFERENCES
`parametrage__departement` (`departement_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `parametrage__ville` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__agence`;

CREATE TABLE `salle__agence` (
  `id_agence` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nom_agence` varchar(50) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `id_ville` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_agence`),
  KEY `salle__agence_id_agence_index` (`id_agence`),
  KEY `salle__agence_id_ville_index` (`id_ville`),
  CONSTRAINT `salle__agence_id_ville_foreign` FOREIGN KEY
(`id_ville`) REFERENCES `parametrage__ville` (`id_ville`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__agence` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__batiment`;

CREATE TABLE `salle__batiment` (
  `id_batiment` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_batiment` varchar(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `id_agence` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_batiment`),
  KEY `salle__batiment_id_batiment_index` (`id_batiment`),
  KEY `salle__batiment_id_agence_index` (`id_agence`),
  CONSTRAINT `salle__batiment_id_agence_foreign` FOREIGN
KEY (`id_agence`) REFERENCES `salle__agence` (`id_agence`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

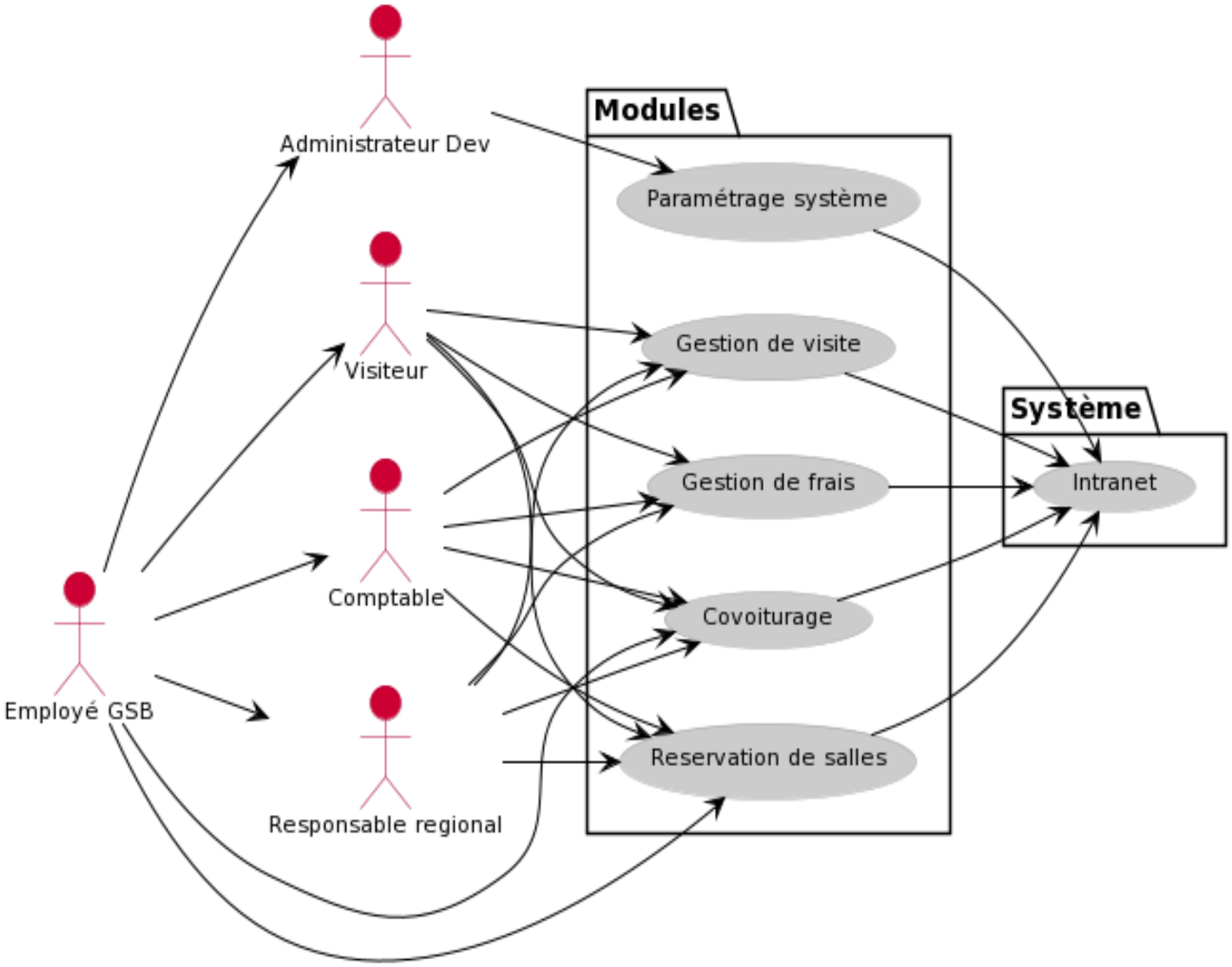
```
LOCK TABLES `salle__batiment` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__extra`;

CREATE TABLE `salle__extra` (
  `id_extra` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `libelle_extra` varchar(100) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_extra`),
  KEY `salle__extra_id_extra_index` (`id_extra`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__extra` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__materiel_type`;

CREATE TABLE `salle__materiel_type` (
  `id_materiel` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_materiel` varchar(255) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_materiel`),
  KEY `salle__materiel_type_id_materiel_index`
(`id_materiel`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__materiel_type` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__posession_materiel`;

CREATE TABLE `salle__posession_materiel` (
  `id_salle` bigint(20) unsigned NOT NULL,
  `id_materiel` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_salle`,`id_materiel`),
  KEY `salle__posession_materiel_id_salle_index`
(`id_salle`),
  KEY `salle__posession_materiel_id_materiel_index`
(`id_materiel`),
```

```
  CONSTRAINT
`salle__posession_materiel_id_materiel_foreign` FOREIGN
KEY (`id_materiel`) REFERENCES `salle__materiel_type`
(`id_materiel`),
  CONSTRAINT `salle__posession_materiel_id_salle_foreign`
FOREIGN KEY (`id_salle`) REFERENCES `salle__salle`
(`id_salle`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__posession_materiel` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__reservation`;

CREATE TABLE `salle__reservation` (
  `id_reservation` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `date_debut_reservation` datetime NOT NULL,
  `date_fin_reservation` datetime NOT NULL,
  `nombre_de_personnes` int(11) NOT NULL,
  `code_employe` bigint(20) unsigned NOT NULL,
  `id_salle` bigint(20) unsigned NOT NULL,
  `id_extra` bigint(20) unsigned DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_reservation`),
  KEY `salle__reservation_id_reservation_index`
(`id_reservation`),
  KEY `salle__reservation_code_employe_index`
(`code_employe`),
  KEY `salle__reservation_id_salle_index` (`id_salle`),
  KEY `salle__reservation_id_extra_index` (`id_extra`),
  CONSTRAINT `salle__reservation_code_employe_foreign`
FOREIGN KEY (`code_employe`) REFERENCES `employe__employe`
(`code_employe`),
  CONSTRAINT `salle__reservation_id_extra_foreign` FOREIGN
KEY (`id_extra`) REFERENCES `salle__extra` (`id_extra`),
  CONSTRAINT `salle__reservation_id_salle_foreign` FOREIGN
KEY (`id_salle`) REFERENCES `salle__salle` (`id_salle`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__reservation` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `salle__salle`;

CREATE TABLE `salle__salle` (
  `id_salle` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `nom_salle` varchar(20) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `id_batiment` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_salle`),
  KEY `salle__salle_id_salle_index` (`id_salle`),
  KEY `salle__salle_id_batiment_index` (`id_batiment`),
  CONSTRAINT `salle__salle_id_batiment_foreign` FOREIGN
KEY (`id_batiment`) REFERENCES `salle__batiment`
(`id_batiment`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `salle__salle` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `visite__famille_medicament`;

CREATE TABLE `visite__famille_medicament` (
  `id_famille` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_famille` varchar(30) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_famille`),
  KEY `visite__famille_medicament_id_famille_index`
(`id_famille`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `visite__famille_medicament` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `visite__medicament`;

CREATE TABLE `visite__medicament` (
  `id_medicament` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `nom_medicament` varchar(30) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `photo_medicament` varchar(255) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `id_famille` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_medicament`),
  KEY `visite__medicament_id_medicament_index`
(`id_medicament`),
```

```
  KEY `visite__medicament_id_famille_index`
(`id_famille`),
  CONSTRAINT `visite__medicament_id_famille_foreign`
FOREIGN KEY (`id_famille`) REFERENCES
`visite__famille_medicament` (`id_famille`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `visite__medicament` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `visite__praticien`;

CREATE TABLE `visite__praticien` (
  `id_praticien` bigint(20) unsigned NOT NULL
AUTO_INCREMENT,
  `raison_sociale` varchar(50) COLLATE utf8mb4_unicode_ci
NOT NULL,
  `prenom` varchar(30) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `nom` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `adresse` varchar(100) COLLATE utf8mb4_unicode_ci NOT
NULL,
  `id_ville` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_praticien`),
  KEY `visite__praticien_id_praticien_index`
(`id_praticien`),
  KEY `visite__praticien_id_ville_index` (`id_ville`),
  CONSTRAINT `visite__praticien_id_ville_foreign` FOREIGN
KEY (`id_ville`) REFERENCES `parametrage__ville`
(`id_ville`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `visite__praticien` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `visite__presentation_medicament`;

CREATE TABLE `visite__presentation_medicament` (
  `id_visite` bigint(20) unsigned NOT NULL,
  `id_medicament` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_visite`,`id_medicament`),
  KEY `visite__presentation_medicament_id_visite_index`
(`id_visite`),
  KEY
`visite__presentation_medicament_id_medicament_index`
(`id_medicament`),
  CONSTRAINT
`visite__presentation_medicament_id_medicament_foreign`
FOREIGN KEY (`id_medicament`) REFERENCES
`visite__medicament` (`id_medicament`),
  CONSTRAINT
`visite__presentation_medicament_id_visite_foreign`
FOREIGN KEY (`id_visite`) REFERENCES `visite__visite`
(`id_visite`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `visite__presentation_medicament` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `visite__visite`;

CREATE TABLE `visite__visite` (
  `id_visite` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `date_debut_visite` date NOT NULL,
  `date_fin_visite` date NOT NULL,
  `rapport` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `id_praticien` bigint(20) unsigned NOT NULL,
  `code_employe` bigint(20) unsigned NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id_visite`),
  KEY `visite__visite_id_visite_index` (`id_visite`),
  KEY `visite__visite_id_praticien_index`
(`id_praticien`),
  KEY `visite__visite_code_employe_index`
(`code_employe`),
  CONSTRAINT `visite__visite_code_employe_foreign` FOREIGN
KEY (`code_employe`) REFERENCES `employe__employe`
(`code_employe`),
  CONSTRAINT `visite__visite_id_praticien_foreign` FOREIGN
KEY (`id_praticien`) REFERENCES `visite__praticien`
(`id_praticien`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

LOCK TABLES `visite__visite` WRITE;

UNLOCK TABLES;
```

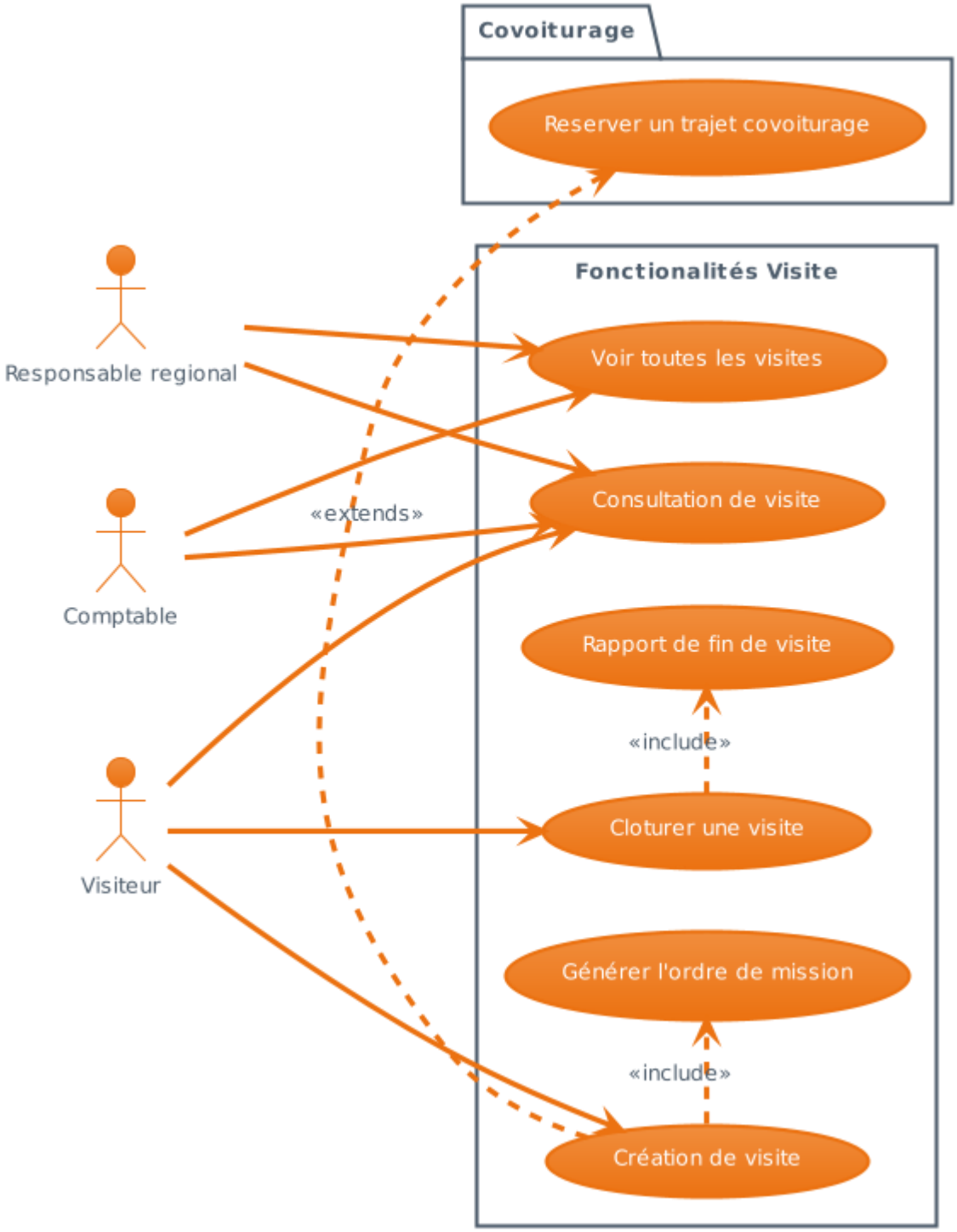
9) Modélisation UML

A) Diagramme de contexte :



B) Diagrammes des cas d'utilisations

a. Gestion de visite



- Objectif :
  L'objectif de ce cas d'utilisation est de mettre en évidence les fonctionnalités essentielles du système de gestion de visites, ainsi que de présenter les principaux acteurs qui auront soit un accès complet à ces fonctionnalités, soit un accès limité aux mêmes fonctionnalités.

- Conditions :
  L'accès à toutes les fonctionnalités sera sécurisé grâce à un système de connexion, couplé à un système de permissions étroitement associé à la configuration propre à chaque employé.

- Scénario :

  Le visiteur incarne le rôle d'un commercial au sein de l'entreprise GSB, ce qui le charge de présenter les médicaments aux professionnels de santé. Lorsqu'il souhaite rendre visite à un praticien, le visiteur a la possibilité de créer une visite. Il sélectionne le praticien qu'il souhaite visiter, les médicaments qu'il va présenter, ainsi que la date de départ. Ensuite, il est redirigé vers la page de covoiturage pour accepter ou refuser un trajet de covoiturage, si cette option est disponible grâce au module de covoiturage. Enfin, il a accès à un document d'ordre de mission qui lui précise toutes les autorisations et les conditions nécessaires pour mener à bien sa mission. De plus, son responsable direct est informé via une notification par e-mail, ce qui lui permet d'obtenir toutes les informations pertinentes sur la visite. Une fois la visite créée, le visiteur peut la consulter à tout moment et la clôturer une fois qu'elle est terminée, en indiquant un rapport de visite comme une étape obligatoire pour la clôture.

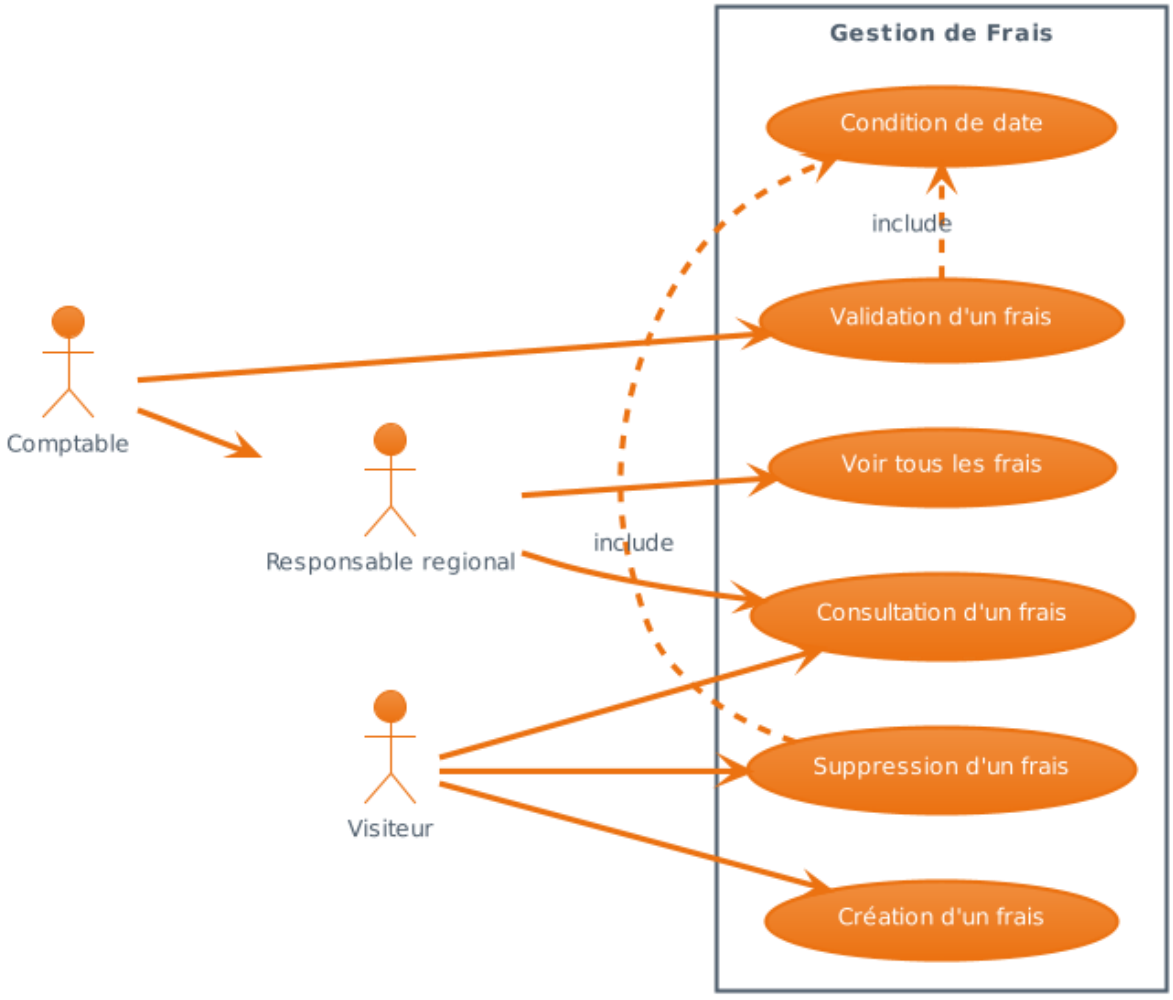
- Le responsable régional

  Le responsable régional a la capacité d'accéder à l'ensemble des visites effectuées par les visiteurs de son équipe. Cela lui permet d'être au fait de toutes les visites réalisées et de consulter les rapports de clôture.

- Comptable

  Le comptable a un accès complet à toutes les visites, car il est directement impliqué dans la gestion des frais. Il est responsable du traitement des déclarations de frais et doit être en mesure de les associer aux visites correspondantes pour une meilleure traçabilité.

b. Gestion de frais



- Objectif :
  L'objectif de ce cas d'utilisation est de mettre en lumière les fonctionnalités essentielles du système de gestion de frais, tout en présentant les principaux acteurs qui auront accès aux fonctionnalités spécifiées dans le diagramme UML de cas d'utilisation. Cette démarche vise à assurer une compréhension claire des capacités du système et de ses utilisateurs clés.

- Conditions :
  Pour garantir la sécurité de l'accès à l'ensemble des fonctionnalités, un système de connexion est mis en place. Ce système de connexion est associé à un système de permissions, étroitement lié à la configuration individuelle de chaque employé. Ainsi, seuls les employés autorisés peuvent accéder aux fonctionnalités en fonction de leurs rôles et de leurs responsabilités au sein de l'entreprise.

- Scénario :

    - Le visiteur

    Après avoir effectué une visite, le visiteur a la possibilité de déclarer tous les frais qu'il a engagés en vue d'une indemnisation, conformément aux forfaits préalablement établis par la direction. Dans le cas où des frais exceptionnels ne sont pas couverts par ces forfaits, le visiteur peut créer une déclaration de frais personnalisée. Cette déclaration permet au visiteur de spécifier le type de frais et de fournir toutes les informations et justificatifs requis. Une fois cette étape complétée, le visiteur est redirigé vers la section de consultation de ses frais, où il trouve un récapitulatif de toutes les informations saisies. Au sein de cette section, il a la possibilité de supprimer un frais, sous réserve de la vérification des conditions liées à la date, car il est interdit de supprimer un frais une fois qu'il est en cours de traitement par le comptable.

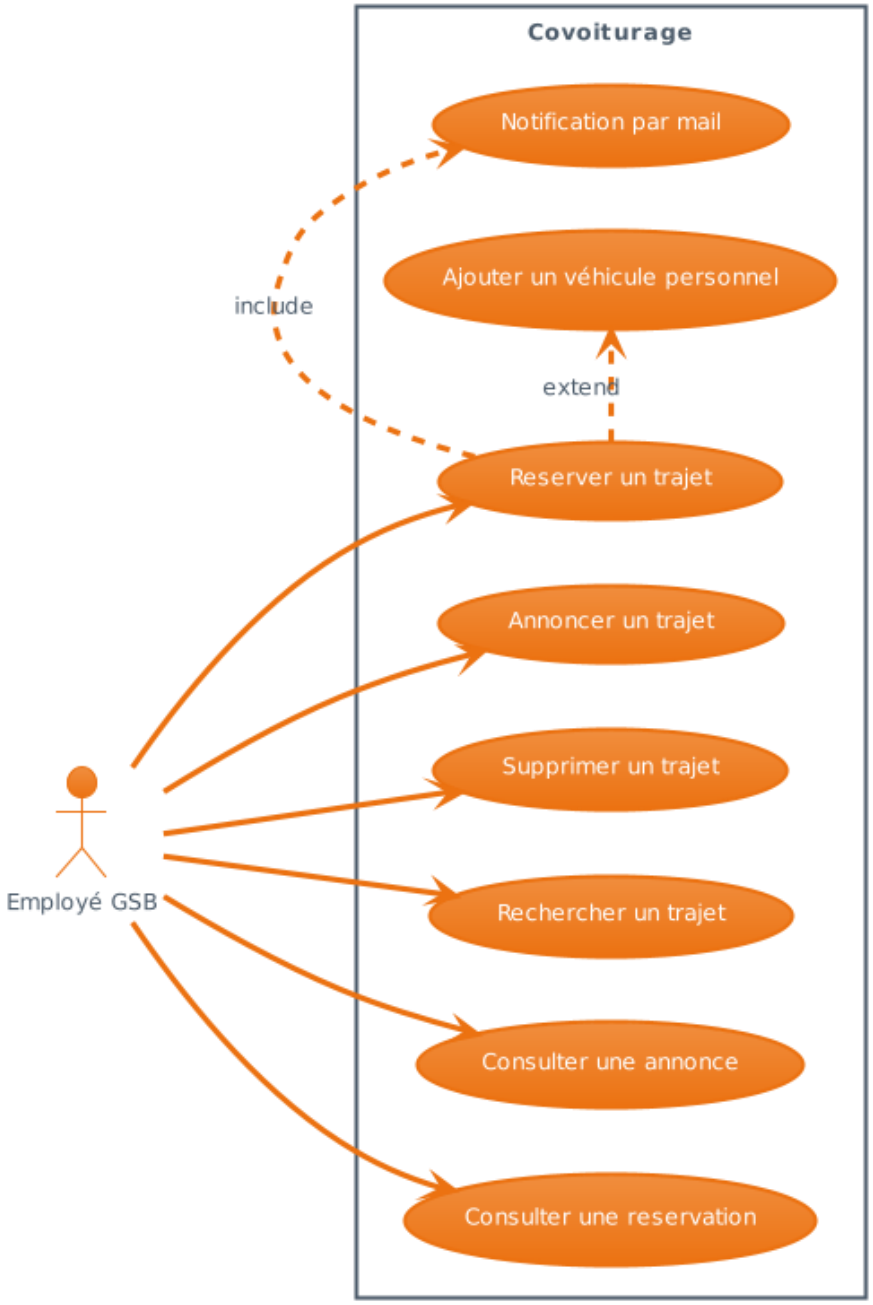
    - Le responsable régional

    Le responsable régional a un accès complet à l'ensemble des frais déclarés par les membres de son équipe. Cette visibilité lui permet d'organiser et de suivre efficacement les activités de son pôle, ainsi que de garantir que les procédures sont respectées.

    - Comptable

    Afin de procéder au remboursement des frais déclarés par les visiteurs, le comptable doit avoir une vue d'ensemble des notes de frais, y compris les détails fournis par le visiteur, ainsi que l'accès aux pièces justificatives associées. Cette approche lui permet de valider les frais en modifiant leur statut de validation. Il peut ainsi les marquer comme "remboursés," "annulés," ou "en cours de remboursement," assurant une gestion transparente et efficace des dépenses de l'entreprise.

c. Covoiturage



- Objectif :

  L'objectif de ce cas d'utilisation est de mettre en lumière les fonctionnalités essentielles du système de covoiturage interne de l'entreprise.

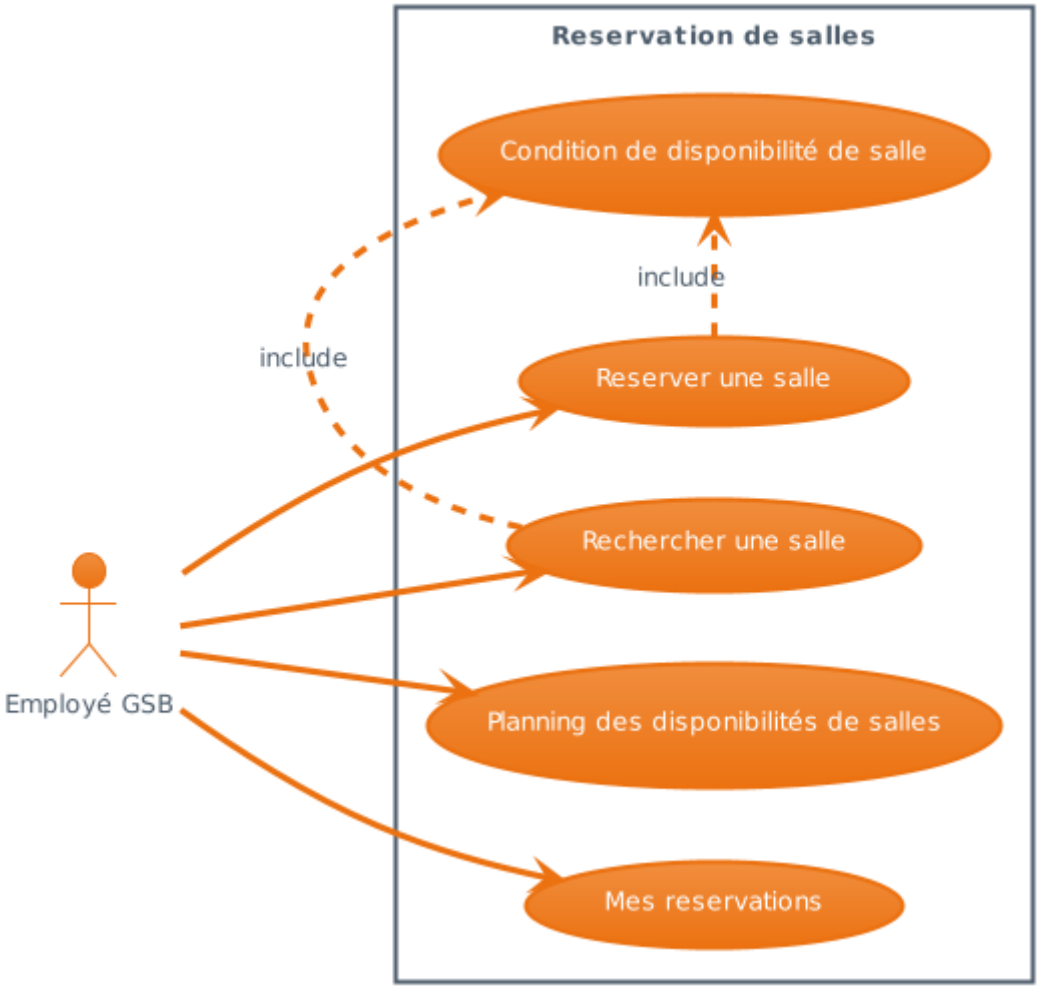
- Conditions :

  Ce module sera accessible à l'ensemble des employés de l'entreprise, offrant ainsi une solution interne visant à réduire les coûts liés aux déplacements professionnels. De plus, il contribue à la protection de l'environnement, en soutenant les initiatives environnementales et sociétales de l'entreprise pour préserver notre planète.

- Scénario :

  Chaque employé ayant accès à l'application de covoiturage peut proposer des trajets en covoiturage, que ce soit avec sa propre voiture ou celle de service de l'agence. Une fois une annonce de covoiturage publiée, tous les employés peuvent utiliser la plateforme pour rechercher des annonces correspondant à leurs créneaux horaires et les réserver. Lorsqu'un employé réserve un trajet, l'auteur de l'annonce reçoit une notification, et les détails de la réservation sont récapitulés dans la vue d'ensemble de l'annonce, permettant ainsi de suivre les réservations en cours.

d. Réservation de salles



- Objectif :

  L'objectif de ce cas d'utilisation est de mettre en évidence les fonctionnalités essentielles du système de réservation de salles, ainsi que de démontrer comment il peut contribuer à l'unification de la structure des agences dans toutes les régions. Cela permettra de mettre à disposition des salles dans l'ensemble des agences et facilitera le suivi et la diffusion des informations pour tous les employés de GSB.

- Conditions :

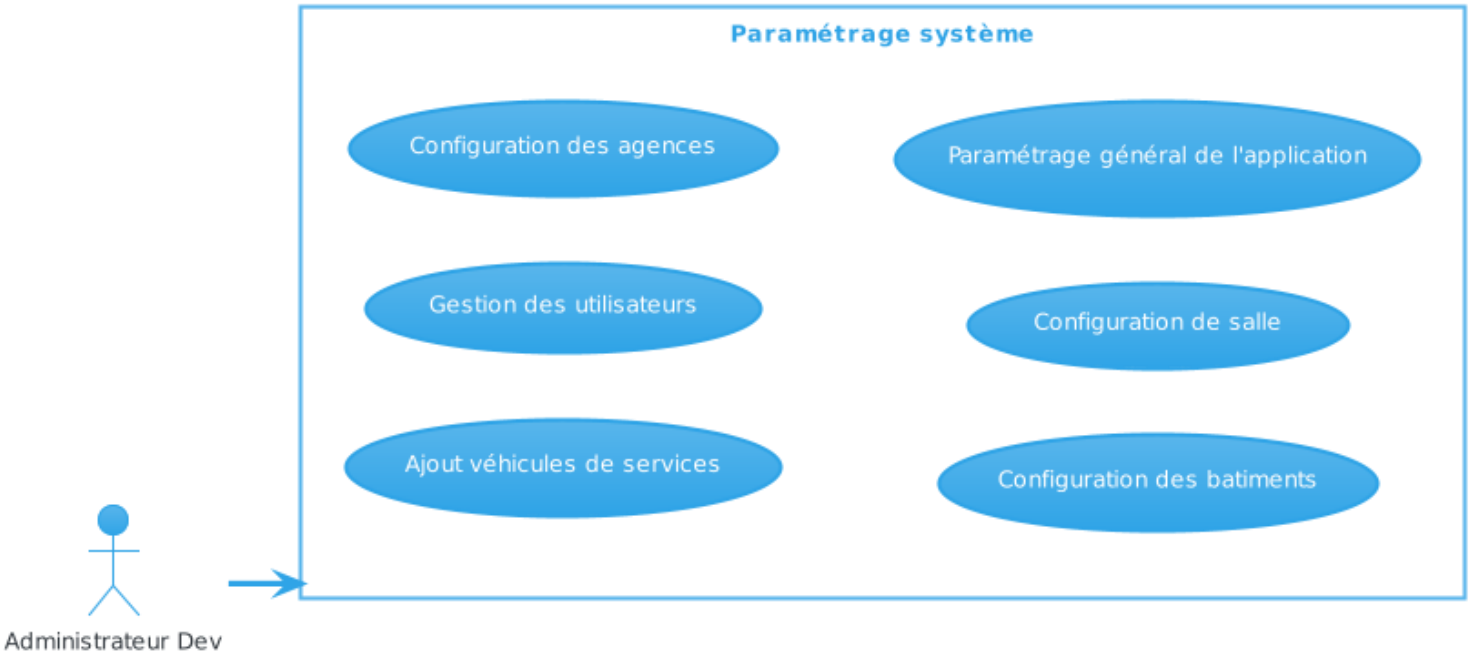
  L'accès à toutes les fonctionnalités sera sécurisé grâce à un système de connexion, couplé à un système de permissions étroitement associé à la configuration propre à chaque employé.

- Scénario :

  Le module de réservation de salle sera accessible à l'ensemble des employés, offrant la possibilité de réserver des salles au sein de multiples agences, tout en consolidant la disponibilité de chaque salle dans un bâtiment et au sein d'une agence. Un planning global unique sera mis à la disposition de tous les employés. Chaque individu ayant besoin d'une salle dans un bâtiment peut effectuer une recherche en spécifiant un créneau horaire, ce qui lui fournira une liste de toutes les salles disponibles. Chacune de ces salles est assortie de ses caractéristiques en termes de matériel, facilitant la recherche en utilisant des filtres pour affiner les critères. Les employés ont également la possibilité de changer d'agence ou de rechercher des salles dans d'autres régions pour vérifier leur disponibilité.

  Enfin, lors de la réservation, les employés peuvent indiquer tous les extras requis, tels que du jus d'orange, du café, ou des repas spécifiquement définis par la direction. Cette approche permet de personnaliser les réservations pour répondre aux besoins spécifiques de chaque employé.

e. Paramétrage système



- Objectif :

  L'objectif principal de ce cas d'utilisation est de mettre en lumière les fonctionnalités qui contribuent à la stabilité de l'intranet tout en enrichissant la base de données. Ces fonctionnalités seront alimentées par l'équipe de développement, en réponse aux demandes formulées par les responsables régionaux et la direction générale.

- Conditions :

  L'accès à ces fonctionnalités sera réservé exclusivement à l'équipe de la direction informatique. Ce groupe utilisera ces fonctionnalités pour mettre à jour et enrichir les données affichées sur l'intranet, garantissant ainsi l'intégrité et la pertinence des informations.

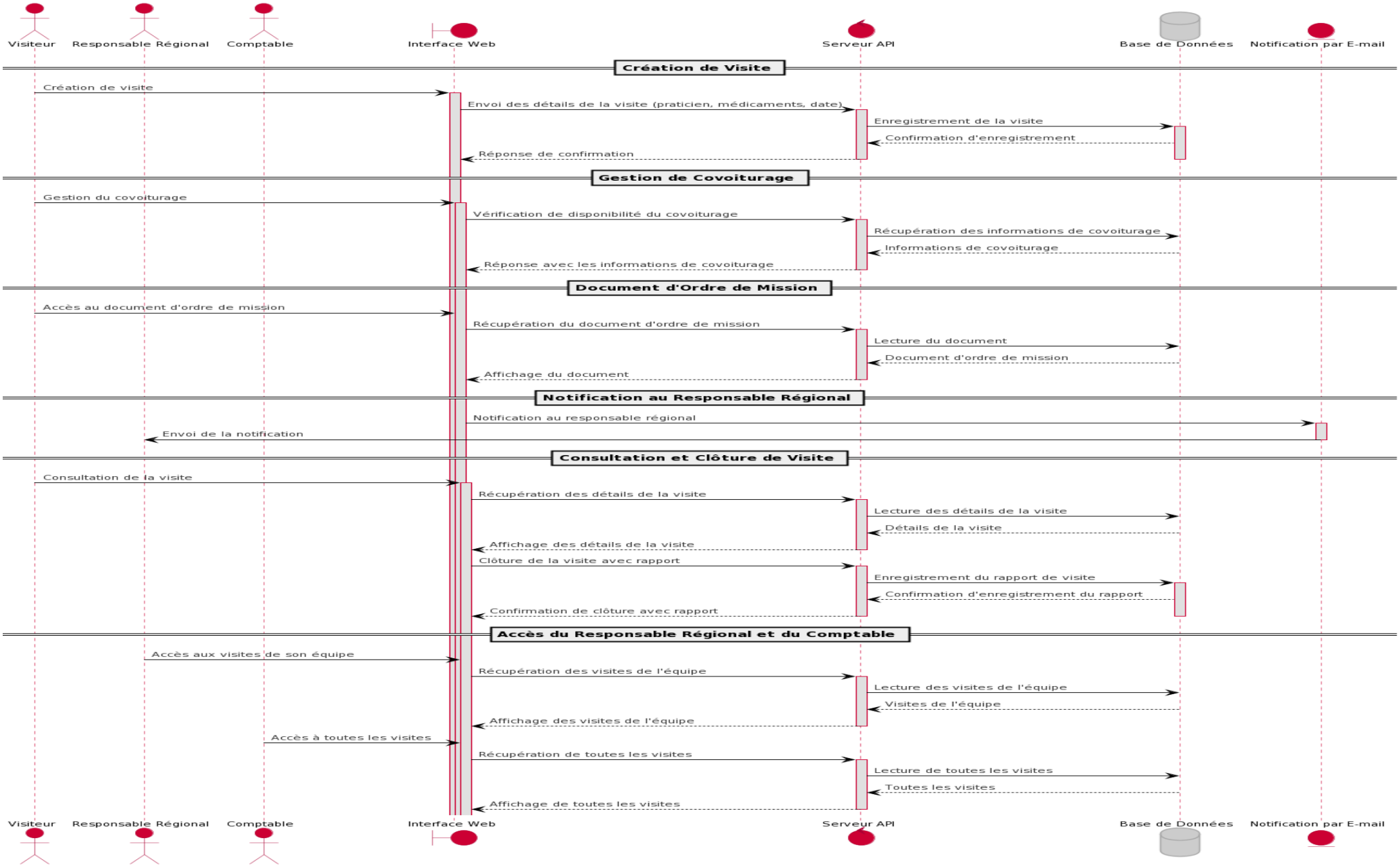
- Scénario :

  Afin de configurer correctement les informations préexistantes et la base de données de l'intranet, qui

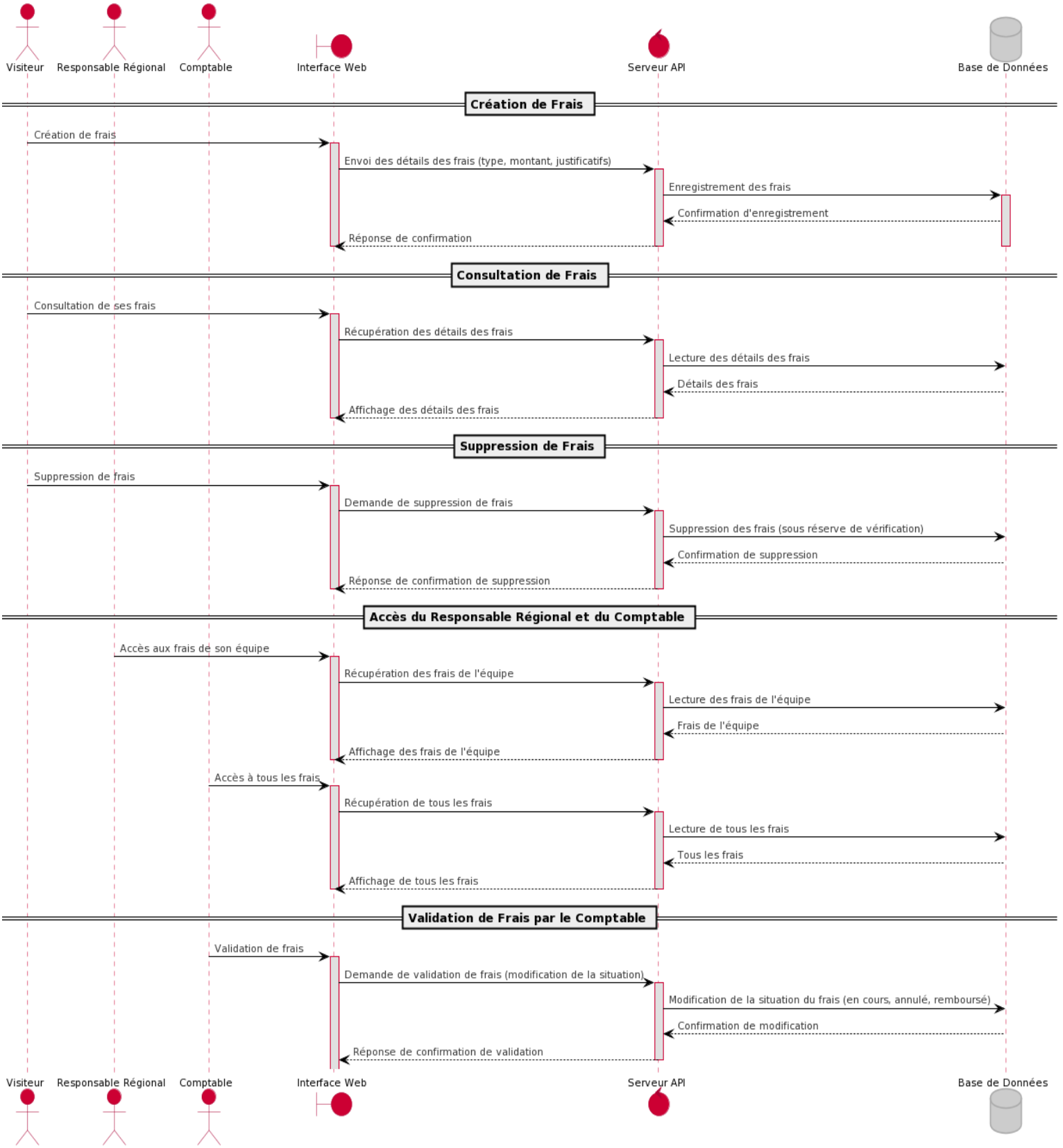
auront un impact sur l'ensemble des modules, un module de paramétrage a été mis en place. Il permet, par exemple, la modification de la structure des agences et des bâtiments en ajoutant de nouveaux bâtiments et salles, facilitant ainsi leur réservation. De plus, la gestion des utilisateurs, effectuée au niveau de la direction informatique, permet de configurer de nouveaux comptes d'utilisateurs, en accord avec les demandes des responsables. Les permissions nécessaires sont attribuées en fonction des rôles de chaque utilisateur. L'ajout de véhicules de service, destinés à être utilisés dans le module de covoiturage, est également réalisé au niveau du paramétrage. Enfin, une configuration générale du site est gérée via ce module pour assurer le bon fonctionnement de l'intranet dans son ensemble.

C) Diagramme de séquence des cas d'utilisations

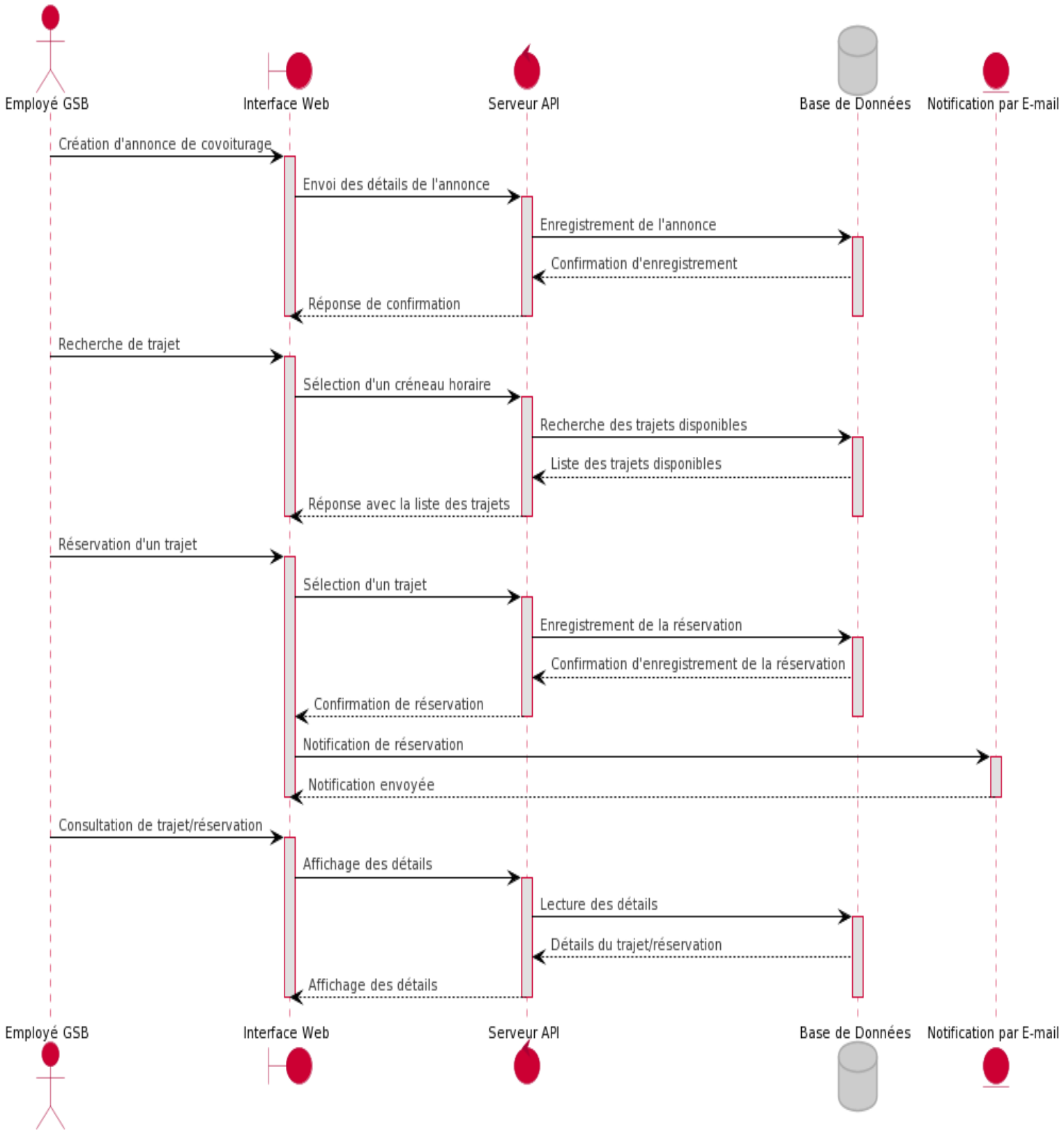
a. Gestion de visites



b. Gestion de frais



c. Covoiturage



d. Réservation de salles