|  |  |
| --- | --- |
| Mémoire Projet personnel  Clémence Vezin | Titre professionnel Concepteur Développeur d’applications Niveau II  Juin 2020 |

Table des matières

1. Présentation générale

* Abstract
* Présentation du projet
* Compétences utilisées

1. Analyse du projet

* Présentation de l’entreprise
* Analyse détaillée et Cahier des charges

1. Spécification fonctionnelle

* Use Case
* Diagramme d’activités
* Diagramme de séquence
* Diagramme de classe
* Maquettage

1. Conception d’une application

* MCD
* MLD

1. Arborescence
2. Spécification technique

* Outils utilisés
* Langages utilisés

1. Fonctionnalités

* Ecran de l’application
* Code associé

1. Conclusion
2. Annexes
3. Présentation générale

* Abstract

My project consists in the creation of an application for the stock’s management of medical supplies in ambulances and emergency vehicles.

The idea for this project came to me following a conversation with a friend and neighbour who is an ambulance driver in a private EMS company.

To deal with the inventories of their different vehicles and warehouses, they use a whiteboard, paper and pencils. So it takes time and it is not done as often as it should be. Mostly because they multitask by being vehicle drivers, first-aid responders, stock managers, liaison between the various emergency and rescue services... All these duties while listening to the patients.

Therefore I wanted to imagine a way to help them and facilitate this task, which is necessary for the good running of their interventions with patients.

That’s why I will try, on the one hand, to create a tool for the company from which the warehouse stock will be managed. There will be a link between the latter and the vehicles stock. On the other hand, the user friendly application will allow the staff to update information in real time.

It will also enable them to quickly make a list of the equipment they need to pick up before going back to the intervention, in order to have the essential stock required in each vehicle.

The main feature of the application will include the management of the stock (changing some medical supplies), checking and viewing the missing items as well as listing them effectively.

For now I’m quite satisfied with the application I have designed, however I still need to work on the software part which I will do during my spare time for the sake of practice.

* Présentation du projet

L’idée de ce projet m’est venue à la suite d’un échange avec un ami et voisin qui est ambulancier dans une société privée.

Actuellement pour faire leurs inventaires des différents véhicules et de leur entrepôt, ils utilisent un tableau blanc, papier et crayons. Cela prend donc du temps et ce n’est pas fait aussi souvent qu’il le faudrait.

J’ai donc voulu imaginer un moyen de les aider et de faciliter cette tâche nécessaire au bon déroulement de leurs interventions auprès des patients.

C’est dans cet esprit que je souhaite créer un outil pour l’entreprise sur lequel sera géré le stock entrepôt et une application qui permettra au personnel de mettre à jour les informations en temps réel. Le lien entre les deux se fera à travers la connexion sur un serveur et une base de données.

Cette application leur permettra de pouvoir faire une liste rapidement du matériel qu’ils doivent récupérer avant de repartir en intervention. Ce qui leur permettra d’avoir le stock minimum obligatoire dans chaque véhicule facilement.

Je vais vous présenter des éléments de l’application. Je travaillerais plus tard sur le logiciel afin de finaliser ce projet.

* Compétences utilisées

|  |  |
| --- | --- |
| Compétences utilisées | Sur le projet personnel |
| Maquetter une application | v |
| Développer des composants d’accès aux données | v |
| Développer la partie front-end d’une interface utilisateur web | v |
| Développer la partie back-end d’une interface utilisateur web | v |
| Concevoir une base de données | v |
| Mettre en place une base de données | v |
| Développer des composants dans le langage d’une base de données | v |
| Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement | v |
| Concevoir une application | v |
| Développer des composants métier | v |
| Construire une application organisée en couches | v |
| Développer une application mobile | v |

1. Analyse du projet

* Présentation de l’entreprise

Mon voisin travaille dans une entreprise de SAMU qui possède différents types de véhicule dans lesquels le stock de matériel médical doit être géré et réapprovisionné à chaque retour d’interventions avant de repartir.

Cette entreprise est située à Cazères sur Garonne et intervient auprès de patients dans la zone du Comminges qu’ils transportent sur les différents Hôpitaux de Haute-Garonne et chez différents médecins.

Il y a 8 véhicules : 4 Ambulances, 2 Taxis, 1 véhicule de secours léger (VSL), 1 véhicule pour le transport des personnes de mobilité réduite (TPMR).

L’entreprise est composée d’une quinzaine de collaborateurs qui alternent dans les différents véhicules.

Chaque collaborateur endosse les différents rôles nécessaires au bon déroulement des interventions.

Ainsi ils sont conducteurs de véhicule, intervenants de premier secours, gestionnaires de stock, liaison entre les différents services de secours et d’urgence tout en étant à l’écoute des patients.

* Analyse détaillée et cahier des charges

Objectifs :

* Création d’une application permettant la gestion du stock dans les véhicules
* Création d’une base de données avec tous les éléments nécessaires à la suite de l’analyse du cahier des charges
* Création d’un serveur pour la liaison entre la base de données et l’application
* Connexion des équipes par véhicule
* Gestion des stocks entrepôt et par véhicule (création, modification, suppression, consultation)
* Création de commandes de stock entrepôt (modification, suppression, consultation)
* Créer des alertes lorsque le stock minimum est atteint dans les véhicules et l’entrepôt
* Obtenir une liste des produits à récupérer à l’entrepôt après une intervention
* Gestion des interventions/prise en charge des patients

Besoins :

* Base de données
* Serveur
* Application de gestion pour les tablettes
* Liste des produits par véhicule
* Liste des employés
* Support à l’entrepôt avec le stock complet, la gestion d’inventaire et la génération des commandes
* Création d’un superviseur, d’un gestionnaire et des employés ayant des droits différents

1. Spécification fonctionnelle

* Use Case

Voici le Use Case global du projet avec tous les cas d’utilisation, tous les acteurs et les liaisons.

Nous verrons plus loin celui spécifique pour l’application. Cette première étape permet de visualiser l’interaction entre les acteurs et le système d’information.

Une image contenant texte

Description générée automatiquement

Dans ce Use Case nous pouvons voir qu’il y a 3 acteurs qui ont des droits et des actions différentes. La notion d’héritage est aussi présente, en effet le Gestionnaire hérite des droits et actions de l’employé et pourra donc lui aussi effectuer ces actions. De la même manière pour le superviseur qui hérite du gestionnaire.

* Diagramme d’activités

Pour les diagrammes d’activités, j’ai choisi plusieurs cas pour bien illustrer les différentes interactions et contraintes entre les acteurs et le système d’information.

Par exemple les actions concernant le véhicule qui devra être créer dans le système puis il pourra être modifié ou supprimé.

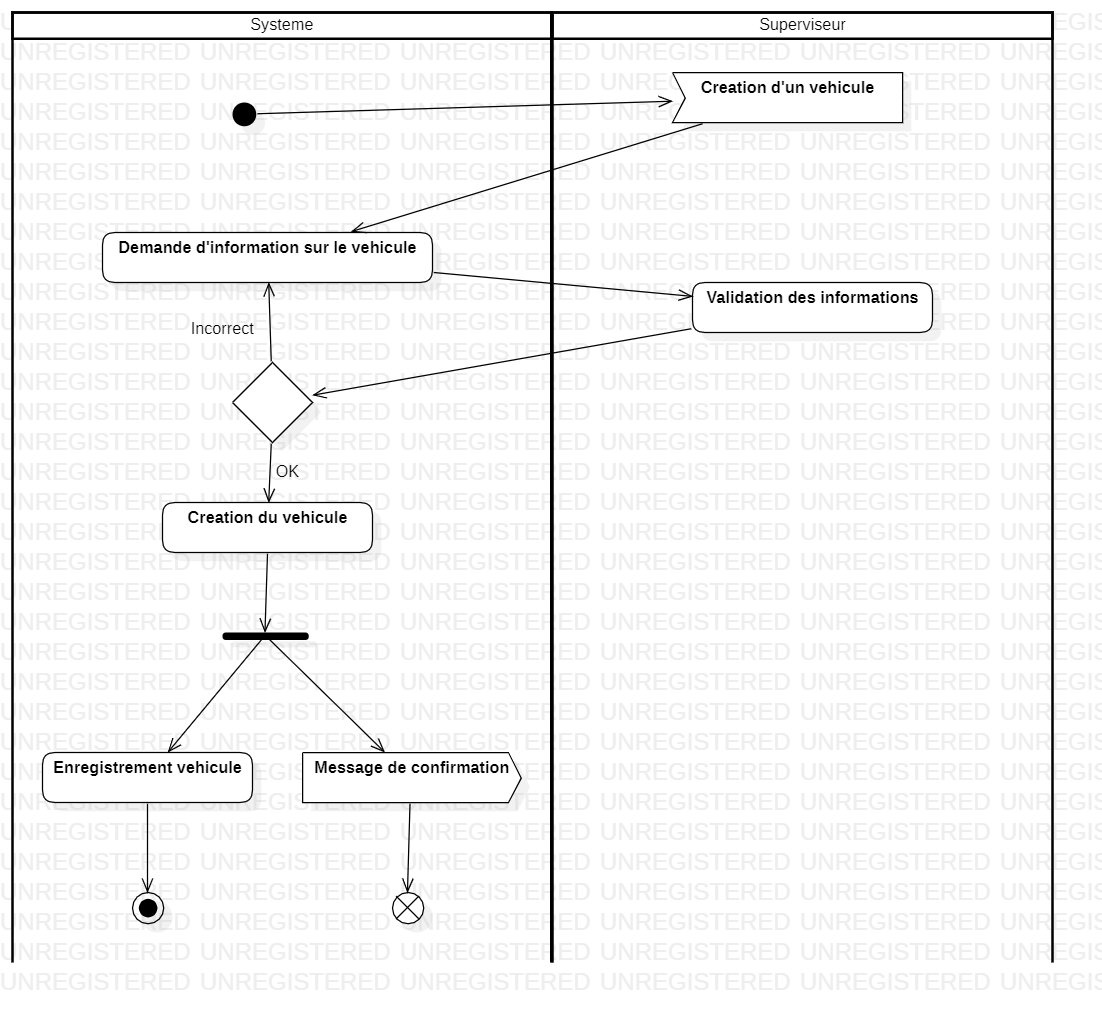
- Création du véhicule

Lors de la création du véhicule, lorsque le superviseur sélectionne la création il créé un évènement qui est réceptionné par le système. Cette action devient alors un Accept Signal.

L’action du superviseur de saisir et valider les informations du véhicule passe ensuite par le nœud de décision. C’est ici que le système vérifie les données transmises. Plusieurs cas de figure peuvent se présenter, soit la saisie est incorrecte (zone non renseignée ou texte erroné) et le système redemande à l’utilisateur de saisir à nouveau les données, soit la saisie est correcte et le système passe à l’action suivante.

Lorsque l’évènement de la création d’un véhicule sera terminé, je souhaite que l’utilisateur soit informé de la fin et validation de l’action par l’envoi d’un message. C’est un Send Signal.

En parallèle de l’envoi du message le système enregistrera le processus de la création du véhicule, pour cela on utilisera le Fork pour pouvoir lancer les 2 actions.



Etat initial

Flow final

Etat final

Send Signal

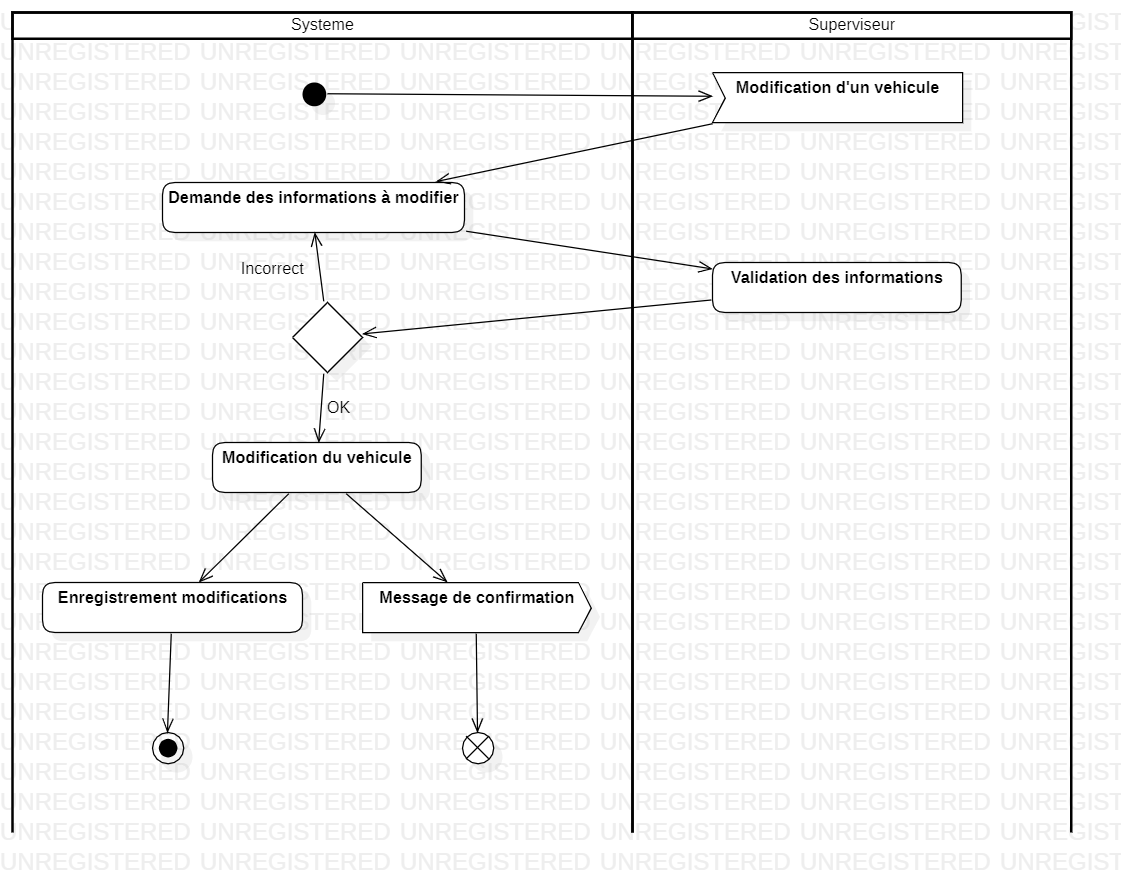
Nœud de décision

Fork

Accept Signal

- Modification du véhicule

Pour la modification du véhicule le diagramme d’activité est proche de celui de la création.



- Suppression du véhicule

Pour la suppression du véhicule il y a des actions supplémentaires au niveau du nœud de décision car le véhicule peut être lié à d’autres éléments. En effet si un employé ou une tablette sont assignés à un véhicule, le superviseur doit être informé de ce lien avant de supprimer le véhicule. Il pourra alors décider de supprimer ces liens avant de supprimer le véhicule.

Sans les messages d’informations pour que l’utilisateur agisse cela entrainerait des erreurs dans le système.

Une image contenant texte

Description générée automatiquement

* Diagramme de séquence

Les diagrammes de séquence reprennent les mêmes cas que les digrammes d’activités. Cela permet de voir les étapes qui seront nécessaires dans la réalisation du code.

Avec ce diagramme on se rapproche du code et de la façon dont les différents éléments vont interagir entre eux.

- Création du véhicule :

Une image contenant fenêtre

Description générée automatiquement

**(5) Scenario Alternatif**

Informations incorrectes

5.1 Informations saisies sur le véhicule non valide

5.2 Rollback

5.3 Retour (2) Demande informations du véhicule tant que c’est incorrect

Le scénario nominal considère que toutes les informations sont exactes

- Modification du véhicule

Une image contenant fenêtre

Description générée automatiquement

**(5) Scenario Alternatif**

Informations incorrectes

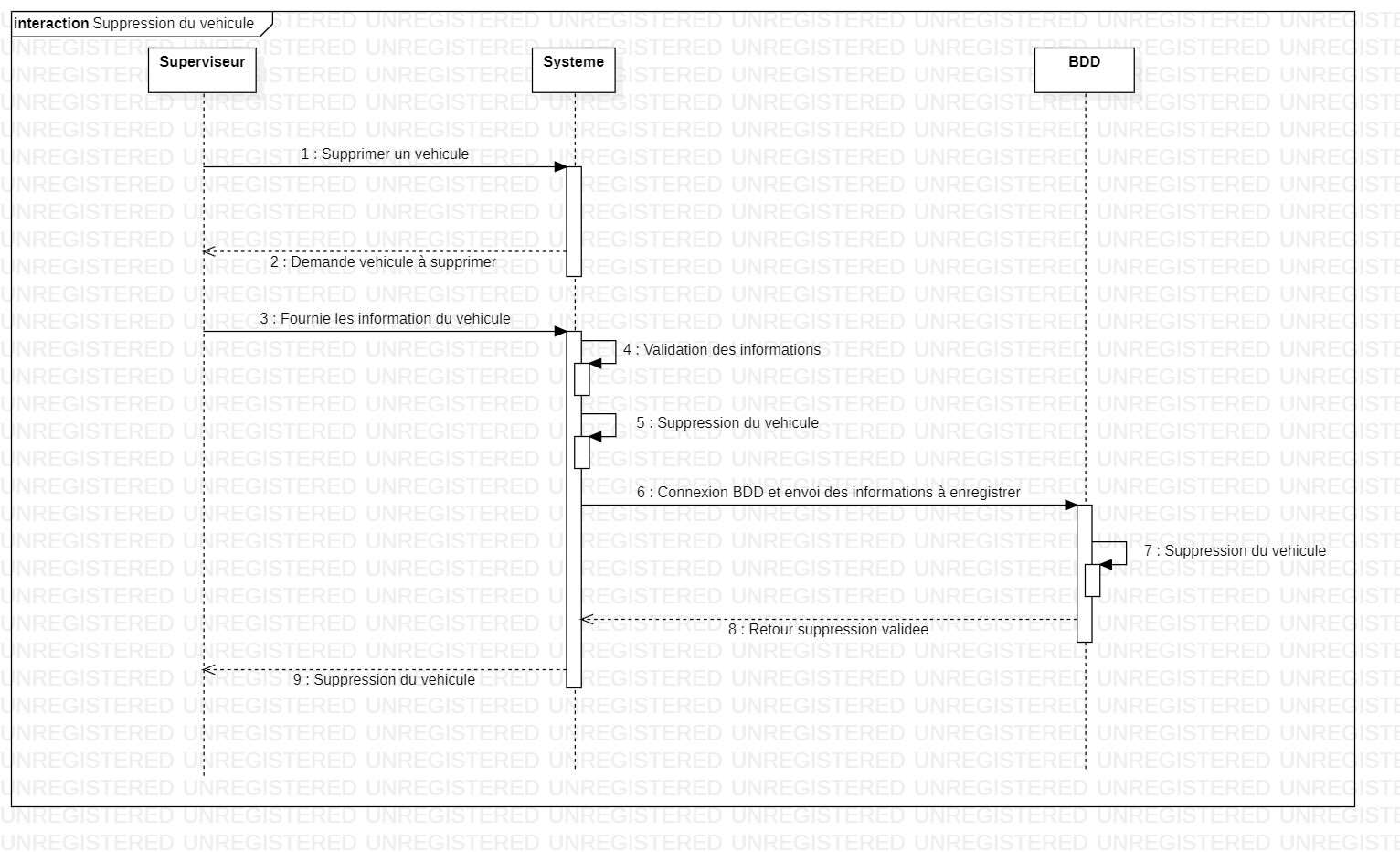
5.1 Informations saisies sur le véhicule non valide

5.2 Rollback

5.3 Retour (2) Demande informations du véhicule tant que c’est incorrect

Le scénario nominal considère que toutes les informations sont exactes

- Suppression du véhicule



**(5) Scenario Alternatif**

Le véhicule n’existe pas

5.1 Informations saisies sur le véhicule non valide

5.2 Rollback

5.3 Retour (2) Demande le véhicule à supprimer tant que c’est incorrect

Le scénario nominal considère que toutes les informations sont exactes

**(5) Scenario d’Erreur**

Le véhicule est assigné à un employé

5.1 Le véhicule est assigné à un employé

5.2 Rollback

Le scénario se termine

**(5) Scenario d’Erreur**

Le véhicule est assigné à une tablette

5.1 Le véhicule est assigné à une tablette

5.2 Rollback

Le scénario se termine

* Diagramme de classe

Le diagramme de classe permet de continuer l’analyse afin de définir les différentes classes qui seront nécessaires au système et les liaisons entre elles.

Les données nécessaires au système sont ici représentées par des classes qui représente un objet.

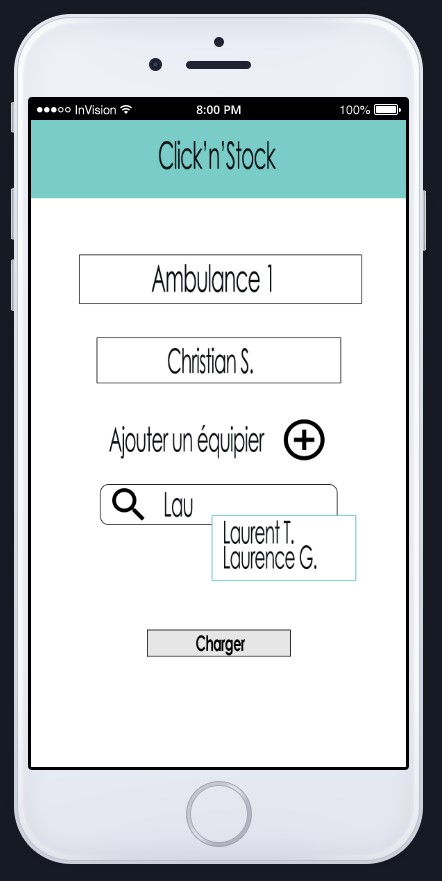
Pour réaliser ce diagramme j’ai fait la liste des données qu’il faudra stocker et les attributs de chaque classe qui permettent de caractériser ces objets.

Une image contenant texte, journal

Description générée automatiquement

* Maquettage de l’application Click’n’Stock

Une image contenant capture d’écran

Description générée automatiquement  Une image contenant capture d’écran

Description générée automatiquement

Une image contenant capture d’écran

Description générée automatiquement Une image contenant capture d’écran

Description générée automatiquement Une image contenant capture d’écran

Description générée automatiquement

1. Conception de l’application Click’n’Stock

* MCD

Pour le modèle conceptuel des données, j’ai repris l’ensemble des entités qui seront dans la base de données.

Les cardinalités sont ici à l’inverse de celles du diagramme de classe.

Une image contenant carte, capture d’écran

Description générée automatiquement

* MLD

La conversion en modèle logique des données nous permet de visualiser l’intégration des clés étrangères dans les tables. L’attribut « date\_affectation » est alors intégré dans la table Employé.

Pour la conversion il faut prendre en compte les associations, celle de type N et celle de type 1.

Dans le cas des associations de type N, il faut ajouter une table d’association qui intègrera les clés étrangères des tables qui la composent. Ici par exemple la table d’association Stocker a pour clé primaire les clés étrangères de Materiel\_medical et de Vehicule car il y a une association de type N entre les 2.

Dans le cas des associations de type 1, la table qui a le 1 absorbe la clé primaire de la table à laquelle elle est associée et elle devient une clé étrangère dans cette table. Par exemple la table Vehicule absorbe la clé de la table Tablette.

Une image contenant capture d’écran, carte

Description générée automatiquement

1. Arborescence

Voici l’arborescence de l’application avec les liens entre les différents écrans.

Une image contenant capture d’écran

Description générée automatiquement

1. Spécification technique

* Outils utilisés

Pour la réalisation de l’analyse du projet et sa conception j’ai utilisé différents outils :

* Star UML pour les diagrammes.
* Jmerise pour les MCD et MLD.
* Photoshop et InVision pour le maquettage.
* Balsamiq Mockup pour l’arborescence.
* Workbench pour la base de données.
* MySQL pour la partie serveur de la base de données.
* IntelliJ pour coder la partie back-end.
* Springboot et JPA pour la partie back-end.
* Android Studio pour la partie front-end.
* Langages utilisés

Voici les langages que j’ai pu utiliser pour la conception :

* SQL pour la base de données.
* Java et JEE pour la partie back-end.
* Kotlin pour la partie front-end.

1. Fonctionnalités

Pour cette partie nous allons voir de plus près l’application et le code.

Comme fonctionnalité de l‘application j’ai choisi de vous montrer l’affichage de l’écran du stock d’un véhicule et plus particulièrement le remplissage du RecyclerView. C’est un composant graphique qui permet d’afficher des données dans une liste déroulante.

Nous allons donc voir toutes les interactions entre les différents éléments qui permettent cet affichage.

Dans un premier temps voici l’analyse de l’application et les différents diagrammes relatifs à la fonctionnalité que je vais vous présenter.

* Use case

L’acteur ici est l’employé qui va utiliser l’application pour consulter , modifier ou supprimer le stock de matériel médical du véhicule.

Une image contenant texte

Description générée automatiquement

* Diagramme d’activités pour l’affichage du RecyclerView

Dans le diagramme d’activité les différentes actions du système et de l’utilisateur sont représentées.

Nous pouvons voir le processus d’événements qui permettra l’affichage des données dans le Recyclerview.



* Diagramme de séquence de l’affichage du RecyclerView

Voici le diagramme de séquence de l’affichage du RecyclerView de l’écran de modification du stock du clic de l’utilisateur à la récupération des données dans la base de données et le retour de ces informations jusqu’à l’affichage.

Nous verrons plus loin le code relatif à cette requête.

Une image contenant texte, journal

Description générée automatiquement

* Ecran de l’application

Une image contenant équipement électronique, ordinateur, téléphone mobile

Description générée automatiquement Une image contenant capture d’écran

Description générée automatiquement

RecyclerView

* Code associé

Nous allons ici voir le code des différentes étapes de l’affichage du RecyclerView. Entre le moment ou l’utilisateur clique sur le bouton et l’affichage de l’écran, il y a plusieurs échanges qui se passent entre le front-end (coté mobile) et le back-end (coté serveur et base de données).

Le front-end envoi une requête au back-end et après traitement obtient une réponse qu’il traite et affiche.

Une image contenant moniteur, écran, photo, assis

Description générée automatiquement

Base de données

Serveur

Smartphone ou tablette

**Front-End**

**Back-End**

Comme nous avons pu le voir dans le diagramme de séquence voici les différentes étapes :

* L’utilisateur clique sur le bouton « Charger », ce qui lance l’activité pour l’affichage du stock du véhicule.

Une image contenant capture d’écran

Description générée automatiquement

* Dans cette activité nous allons faire appel à 2 classes qui vont permettre d’afficher correctement le RecyclerView. L’Adapter et l’AsynTask.

L’Adapter est une classe qui va gérer, à travers différentes méthodes, l’affichage des données dans le RecyclerView.

L’AsynTask quant à elle va exécuter en tâche de fond des actions sans bloquer l’interface graphique.

Une image contenant capture d’écran

Description générée automatiquement

Une image contenant capture d’écran

Description générée automatiquement

* Dans la méthode doInBackground de l’AsynTask il y a un appel à une fonction web service pour communiquer avec le serveur.

Une image contenant capture d’écran

Description générée automatiquement

* Le web service fait alors appel à un utilitaire pour envoyer la requête au serveur grâce à la fonction sendGetOkhttpRequest en indiquant en paramètre l’URL à utiliser.

Une image contenant texte, capture d’écran

Description générée automatiquement

* La requête est transmise au serveur et exploitée dans la classe ServerAPI grâce à l’URL.

Une image contenant capture d’écran, oiseau

Description générée automatiquement

* Le ServerAPI fait ensuite appel à une méthode qui va transmettre la requête à la base de données pour récupérer les données.

Une image contenant capture d’écran

Description générée automatiquement

* Dans la base de données nous récupérons les données de la table Materiel\_medical.

Une image contenant capture d’écran

Description générée automatiquement

Cette table a été ajouté à la Base de Données à partir de l’analyse réalisée dans le MLD en utilisant le code SQL suivant :

CREATE TABLE Materiel\_medical(

id\_mat\_medic Int Auto\_increment NOT NULL ,

nom\_mat\_medic Varchar (100),

date\_expiration Date,

stock\_mat\_medic Int,

stock\_min\_mat\_medic Int

,CONSTRAINT Materiel\_medical\_PK PRIMARY KEY (id\_mat\_medic)

);

* Ces données sont alors renvoyées jusqu’à l’activité permettant l’affichage de la liste du stock du matériel médical du véhicule.

1. Conclusion

La réalisation de ce projet est beaucoup plus longue et complexe que je ne l’imaginais au départ. Cela m’a permis de comprendre et d’appréhender toute la dimension du métier de concepteur développeur.

C’est passionnant de voir comment d’une idée, d’un simple échange on peut réussir à créer tout un programme qui pourra faciliter le travail d’autres personnes.

J’ai conscience d’avoir encore du travail pour mener mon projet à son terme et pouvoir le faire tester à mon voisin et ses collègues. Cela ne m’empêche pas d’avoir des idées pour une possible évolution comme la prise en charge des données des patients et leurs transmissions aux centres de soins afin de gagner du temps.

Pour cette évolution il y a toute la partie confidentialité et sécurité des données qui sera très importante à gérer. Il me faudra acquérir de l’expérience et de nouvelles compétences pour y arriver car je ne suis qu’au début de cette nouvelle vie.

1. Annexes

* Code complet de la création de la base de données

CREATE database bddclicknstock;

use bddclicknstock;

CREATE TABLE Tablette(

id\_tablette Int Auto\_increment NOT NULL ,

nom\_tablette Varchar (100),

adresse\_mac\_tablette Varchar (50)

,CONSTRAINT Tablette\_PK PRIMARY KEY (id\_tablette)

);

CREATE TABLE Materiel\_medical(

id\_mat\_medic Int Auto\_increment NOT NULL ,

nom\_mat\_medic Varchar (100),

date\_expiration Date,

stock\_mat\_medic Int,

stock\_min\_mat\_medic Int

,CONSTRAINT Materiel\_medical\_PK PRIMARY KEY (id\_mat\_medic)

);

CREATE TABLE Type\_vehicule(

id\_type\_vehicule Int Auto\_increment NOT NULL ,

nom\_type\_vehicule Varchar (100)

,CONSTRAINT Type\_vehicule\_PK PRIMARY KEY (id\_type\_vehicule)

);

CREATE TABLE Vehicule(

id\_vehicule Int Auto\_increment NOT NULL ,

nom\_vehicule Varchar (100),

immatriculation\_vehicule Varchar (15),

id\_tablette Int ,

id\_type\_vehicule Int NOT NULL

,CONSTRAINT Vehicule\_PK PRIMARY KEY (id\_vehicule)

,CONSTRAINT Vehicule\_Tablette\_FK FOREIGN KEY (id\_tablette) REFERENCES Tablette(id\_tablette)

,CONSTRAINT Vehicule\_Type\_vehicule0\_FK FOREIGN KEY (id\_type\_vehicule) REFERENCES Type\_vehicule(id\_type\_vehicule)

);

CREATE TABLE Mouvement\_de\_stock(

id\_mvt\_stock Int Auto\_increment NOT NULL ,

date\_mvt\_stock Date,

intitule\_mvt\_stock Varchar (100),

entree\_stock Int,

sortie\_stock Int,

id\_mat\_medic Int,

id\_vehicule Int

,CONSTRAINT Mouvement\_de\_stock\_PK PRIMARY KEY (id\_mvt\_stock)

,CONSTRAINT Mouvement\_de\_stock\_Materiel\_medical\_FK FOREIGN KEY (id\_mat\_medic) REFERENCES Materiel\_medical(id\_mat\_medic)

,CONSTRAINT Mouvement\_de\_stock\_Vehicule0\_FK FOREIGN KEY (id\_vehicule) REFERENCES Vehicule(id\_vehicule)

);

CREATE TABLE Employe(

id\_employe Int Auto\_increment NOT NULL ,

nom\_employe Varchar (100),

prenom\_employe Varchar (100),

login Varchar (100),

mdp Varchar (100),

date\_affectation Date,

id\_vehicule Int

,CONSTRAINT Employe\_PK PRIMARY KEY (id\_employe)

,CONSTRAINT Employe\_Vehicule\_FK FOREIGN KEY (id\_vehicule) REFERENCES Vehicule(id\_vehicule)

);

CREATE TABLE stocker(

id\_mat\_medic Int NOT NULL ,

id\_vehicule Int

,CONSTRAINT stocker\_PK PRIMARY KEY (id\_mat\_medic,id\_vehicule)

,CONSTRAINT stocker\_Materiel\_medical\_FK FOREIGN KEY (id\_mat\_medic) REFERENCES Materiel\_medical(id\_mat\_medic)

,CONSTRAINT stocker\_Vehicule0\_FK FOREIGN KEY (id\_vehicule) REFERENCES Vehicule(id\_vehicule)

);

* Code complet de l’écran d’affichage du stock du véhicule

Une image contenant capture d’écran

Description générée automatiquement

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="20dp"  
 android:orientation="vertical"**>  
  
 <**TextView  
 android:id="@+id/tvStockVehicule"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="30dp"  
 android:gravity="center"  
 android:text="Stock du véhicule"  
 android:textSize="24sp"** />  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10dp"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/tvVehicule"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:text="Ambulance 1"  
 android:textSize="20sp"** />  
  
 <**LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:orientation="vertical"**>  
  
 <**TextView  
 android:id="@+id/tvEquipier1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Christian S"  
 android:textSize="20sp"** />  
  
 <**TextView  
 android:id="@+id/tvEquipier2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Laurent T."  
 android:textSize="20sp"** />  
 </**LinearLayout**>  
  
 </**LinearLayout**>  
  
 <**SearchView  
 android:id="@+id/svListeStock"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_marginRight="30dp"  
 android:background="@drawable/border"** />  
  
 <**androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rvListeStock"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginEnd="30dp"  
 android:layout\_marginBottom="15dp"  
 android:layout\_weight="1"  
 android:background="@drawable/bordebleu"  
 tools:listitem="@layout/row\_materiel\_medic"** />  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:orientation="horizontal"**>  
  
 <**Button  
 android:id="@+id/btModifier"  
 android:layout\_width="150dp"  
 android:layout\_height="wrap\_content"  
 android:onClick="chargerModifStock"  
 android:text="Modifier"** />  
  
 <**TextView  
 android:id="@+id/textView"  
 android:layout\_width="30dp"  
 android:layout\_height="wrap\_content"  
 android:text=" "  
 tools:visibility="invisible"** />  
  
 <**Button  
 android:id="@+id/btRecharger"  
 android:layout\_width="150dp"  
 android:layout\_height="wrap\_content"  
 android:onClick="chargerRechargementStock"  
 android:text="Recharger"** />  
 </**LinearLayout**>  
</**LinearLayout**>

Une image contenant capture d’écran

Description générée automatiquement

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:orientation="horizontal"**>  
  
 <**TextView  
 android:id="@+id/tvMaterielMedic"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:padding="10dp"  
 android:text="Masque Oxygène"  
 android:textSize="18sp"** />  
  
 <**TextView  
 android:id="@+id/tvStockMatMedic"  
 android:layout\_width="50dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginRight="20dp"  
 android:gravity="center"  
 android:padding="5dp"  
 android:text="5"  
 android:textSize="18sp"** />  
</**LinearLayout**>