# Résumé

Lors de notre projet de dernière année à l’ISIMA, nous avons effectué un projet en collaboration avec le **LIMOS**. Il consistait en la création d’une **application mobile** pour le **SICTOM des Couzes**.

Celle-ci devait permettre d’enregistrer les tournées effectuées lors de la collecte des déchets. L’utilisateur s’identifie sur l’application, renseigne les paramètres de la tournée et indique, à l’aide d’un bouton, quand commence et termine cette-dernière. Il peut également alerter les autres utilisateurs en cas de problème et ils recevront sa position actuelle. Cette application a été codée en Android et fait appel à **l’API** de **Google Maps**. Les données sont stockées la **base de données NoSQL** de Google, le **Datastore**.

Nous avons également créé une **application web** permettant à un administrateur de consulter ces différentes tournées en temps réel ainsi que différents graphiques permettant de traiter les données enregistrées en base. Pour cela, nous avons utilisé les technologies **JavaScript**, **JSON** et **Google Chart**.

# Abstract

During our grad school in ISIMA, we worked on a project in collaboration with the LIMOS. We have to create a mobile application for SICTOM of Couzes.

This application allows to save the tours of SICTOM’s trucks during their waste collection. The user connects, indicates the parameters of the tour (what is the truck used? Who is the driver? …). Then, he has to press a button to record the start of the tour and another button allows to finish it. In case of trouble, he can alert the other users who will receive its current location. This application is in Android coding. We used the API of Google Maps and our data are stored in the NoSQL database of Google which is called Datastore.

We have also created a web application which allows an administrator to check all the tours in real time. He can also see different charts which present diverse information and trends from our data. We used JavaScript, JSON and Google Chart technologies.

The objective is to present our work to the employees of SICTOM to show them how our application can be useful and try to convince them to use it.

# Introduction

Dans le cadre de notre dernière année à l’Institut Supérieur d’Informatique, de Modélisation et de leurs Applications (ISIMA), nous avons réalisé un projet tuteuré de deux cents heures. Celui-ci a été encadré par Raksmey Phan.

Ce projet été conçu pour le SICTOM ??? … Il consistait à créer une application mobile permettant aux employés du SICTOM d’enregistrer leurs parcours en temps réel afin de faire ressortir différentes statistiques et permettre à l’utilisateur de signaler, via cette application, s’il a un problème pour que tous les employé soient alertés et puissent intervenir sur la localisation qui leur est envoyée.

* LIMOS (Laboratoire d’Informatique, de Modélisation et d’Optimisation des Systèmes)

Pour réaliser ce projet, nous avons utilisés différents outils qui seront présentés par la suite dans ce rapport. Nous avons réalisé cette application Android à l’aide de l’environnement de développement Android Studio. Nous avons créé et utilisé une base de données NoSQL avec le DataStore de Google.

Nous commencerons par présenter le LIMOS ainsi que le SICTOM de … pour lequel nous avons travaillé. Nous présenterons ensuite les différentes fonctionnalités de notre application puis nous expliquerons la manière et les moyens avec lesquels nous avons développé ces dernières. Nous finirons par présenter le résultat obtenu à la fin de ce projet.

# Présentation du LIMOS et du SICTOM

Ce projet a été réalisé pour le LIMOS.

## Le LIMOS

Le LIMOS est une unité du Centre national de la recherche scientifique français rattaché aux universités de Clermont-Ferrand (*Université d’Auvergne-Clermont 1 et Université Blaise Pascal – Clermont 2*) ainsi qu’à l’Ecole des Mines de Saint-Etienne depuis 2012. Il a été créé en 1995 et se composait alors de 23 enseignants chercheurs. Aujourd’hui, on compte, entre autres, 63 enseignants-chercheurs, 2 chercheurs CNRS (*Centre National de la Recherche Scientifique*) et 70 doctorants.

* Voir rapport



## SICTOM des Couzes

Le SICTOM des Couzes couvre 46 communes du Puy-de-Dôme ce qui représente 26 037 habitants. Il assure la collecte des déchets des ménages et le stockage des déchets résiduels.

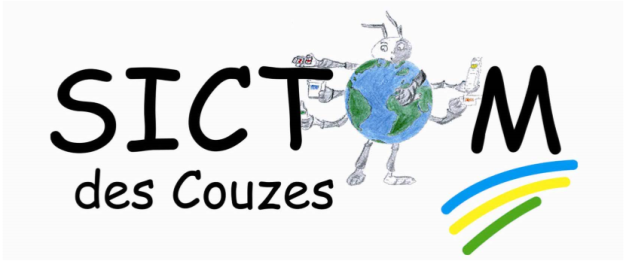


Il dispose de deux déchèteries qui sont situées à Besse et Montaigut-le-Blanc. Le SICTOM dispose de 6 camions. Il existe deux types de collecte :

* **La collecte des ordures ménagères :** Celle-ci a lieu une fois par semaine, en porte à porte pour les bacs individuels et de manière variable pour les bacs de regroupement.
* **La collecte des déchets recyclables :** Elle peut se faire au porte à porte pour les bacs individuels (*suivant un calendrier prévu à cet effet*) ou en point d’apport volontaire.

Il dispose d’une Installation de Stockage des Déchets Non Dangereux (*ISDND*) à St-Diéry, ce qui permet de traiter localement les déchets produits sur le territoire du SICTOM.

Le SICTOM a mis en place un système de prévention visant à sensibiliser le public au tri des déchets et trouver des solutions pour éviter de produire trop de déchets (*ex : action « Stop pub »*)



# Définition du sujet

## Les objectifs

La première étape consistait à identifier les différentes fonctionnalités que nous devions apporter à l’application que nous devions concevoir. Comme indiquait précédemment, celle-ci a été conçue pour les employés du SICTOM de X. Celle-ci n’avait pas pour but de répondre à une demande particulière de leur part mais de leur proposer une solution permettant de les aider dans leur travail au quotidien. Nous avons alors, à l’aide de notre tuteur Raksmey Phan, définit les caractéristiques de notre application. Les différentes fonctionnalités retenues étaient :

* **Authentification de l’utilisateur**

Afin de pouvoir utiliser l’application, l’utilisateur doit nécessaire s’est identifié précédemment via l’application. Ainsi, nous pouvons savoir directement qui est l’employé concerné par les données que nous allons collecter.

* **Géolocalisation**

Une Google Map permet d’indiquer, à tout moment, à l’utilisateur, son emplacement courant, à condition bien sûr qu’il dispose d’une connexion internet.

* **Sauvegarde du parcours**

Les trajets effectués par les différents utilisateurs sont sauvegardés dans notre base de données. Deux possibilités existent : l’utilisateur souhaite se mettre en mode « connecté » (*Les données sont enregistrées en temps réel*) ou bien l’historique de ses positions sera sauvegardé sur le téléphone et sauvegardé en base à la prochaine connexion.

* **Alerte en cas de problème**

Un bouton d’alerte est disponible, permettant à l’utilisateur de prévenir ses collègues en cas de problème. Ainsi, ces-derniers seront prévenus et recevront la position exacte de l’utilisateur, afin de pouvoir intervenir.

* **Visualisation des parcours de l’ensemble des camions**

On peut accéder, via un service web, aux différents historiques de parcours par camion. Une page web présentera les différents trajets par camion afin de visualiser, par exemple, ceux qui sont le plus utilisés.

* **Analyse et traitement des données**

On utilise les données récoltée afin d’en faire ressortir plusieurs tendances (à définir). Des schémas sont proposés afin de présenter les différentes statistiques basées sur les données.

Le diagramme de cas d’utilisation présenté en figure X regroupe les différentes fonctionnalités rattachées à notre application.

* Ici le diagramme de cas d’utilisation
* Donner un nom à notre appli

Une fois les objectifs définis, nous devions déterminer les outils avec lesquels nous allions réaliser ce travail.

## Les outils utilisés

Nous avons utilisés divers outils que nous ne connaissions pas forcément. Dans un premier temps, nous nous sommes concentrés sur la base de données.

### Base de données NoSQL

Il existe différentes catégories de bases de données. Celle que l’on retrouve le plus souvent est la **base de données relationnelle**. Il s’agit du modèle classique se composant de tables reliées entre elles.

Ici, nous avons choisi d’utiliser une **base de données NoSQL** pour différentes raisons. Nous étions libres de choisir le type de bases de données que nous souhaitions. N’ayant pas beaucoup de connaissances dans cet autre type de bases de données nous avons trouvé cela intéressant de pouvoir approfondir nos compétences sur le sujet. De plus, nous avion pu découvrir au cours de nos TP de J2EE le Datastore de Google que nous présenterons ci-après. Utilisant déjà la Map de Google pour la géolocalisation, nous avons voulu utiliser aussi la base de données de Google. De plus, celle-ci permet de déployer facilement nos APIs sur le Cloud de Google.

Une base de données NoSQL est un SGBD (*Système de gestion de base de données*) adapté aux gros volumes de données. L’objectif est d’accéder rapidement aux données, et ce malgré leur quantité considérable. Il n’est plus question de table mais de tableaux associatifs unidimensionnels pouvant contenir des millions de lignes.

Il existe différents type de bases de données NoSQL mais nous ne nous intéresserons qu’au système clé-valeur car le DataStore de Google est de ce type-là.

### Datastore de Google

Le Datastore de Google permet de stocker nos données sur Google App Engine (*Plateforme permettant d’héberger nos application et bases de données sur les serveurs de Google*). Comme énoncé précédemment, le Datastore est de type **clé-valeur**. Le principe est similaire aux tables de hachage. Chaque donnée est représentée par un couple clé / valeur. Pour accéder à celle-ci, il suffit de connaître sa clé. Afin d’organiser et structurer ces données, chaque couple clé-valeur est rattaché à une **entité**. La figure X représente un exemple d’entité.



Figure 1 : Exemple d'entité du Datastore

Ici, l’entité est *Livre.* Elle contient trois paires clé-valeur avec pour clés : *Titre, Prix, Description.* A chaque fois que nous voulons créer un livre, nous devons créer une entité *Livre* et donner une valeur à chacun de ses paramètres. Il est possible ensuite d’accéder et de modifier les données avec différentes opérations :

* **Get**: Récupérer une entité à partir de sa clé
* **Put**: Ajouter une nouvelle entité
* **Delete**: Supprimer une entité à partir de sa clé
* **Query**: Effectuer une requête plus complexe en affinant sa recherche à l’aide de critères

Un autre outil que nous avons utilisé pour cette application est l’API Google Maps nous permettant d’afficher la position courante de l’utilisateur.

### L’API de Google Maps

Google met à notre disposition une API permettant d’accéder aux fonctionnalités de sa Maps. Nous avons utilisé deux versions de celle-ci : l’API JavaScript pour notre application web et l’API android pour notre application mobile.

Détailler davantage

## Organisation du projet

Le diagramme de Gantt réel de notre projet est présenté en figure X.

* Diagramme
* **Test API Google Maps**

Nous avons commencé par tester l’API de Google Maps sur une simple page Web afin de nous familiariser avec celle-ci et de comprendre quelles en étaient les principales fonctionnalités.

* **Gestion BDD**

Nous avons ensuite travaillé sur notre base de données. Nous avons défini les tables qui nous étaient nécessaires et mis en place les méthodes d’accès à celle-ci en Java.

* **Connexion application – BDD**

C’est une des parties du développement les conséquentes. Il s’agit de créer une application mobile utilisant l’API Google Maps et envoyant les données à la BDD.

* **Développement des interfaces**

Dans le même temps, l’un développait le back-end\*, l’autre la partie front\*. Celle-ci consistait à créer les différentes vues de l’application en gérant les événements provoqués par l’utilisateur.

…

# Développement de l’application mobile

Avant de commencer à nous intéresser à la partie Android du développement, nous devions réfléchir à la structure de notre base de données.

## Gestion de la base de données

### Organisation de la base de données

Pour réaliser ce projet, nous avions besoin de trois entités différentes :

* **Coordonnée**

Nous sauvegardons les coordonnées des points composant le parcours des camions. Pour cela, chaque coordonnée se compose d’une **latitude** et d’une **longitude**. On enregistre aussi sa **date** d’enregistrement.

* **Utilisateur**

Il est nécessaire d’enregistrer tous nos utilisateurs pour qu’ils puissent se connecter à l’application. Chaque utilisateur possède un **identifiant** et un **mot de passe** ainsi qu’un **nom**et un **prénom.**

* **Camion**

Chaque camion est enregistré dans la base car un utilisateur ne conduira pas forcément le même camion à chaque fois. Au début de sa tournée, il lui est demandé de renseigner le camion avec lequel l’opération doit être enregistrée. Un camion possède un **nom,** le **poids courant** qu’il transporte et sa capacité **maximale.**

* **Tournée**

Les tournées possède un **numéro** sont rattachées à **trois utilisateurs** : un chauffeur et deux rippers (*personnes chargées du ramassage des poubelles*) et un **camion**. On sauvegarde également la **date début** et de la **date fin** de tournée.

* Schéma récapitulatif

### Sauvegarde des données en base

* Explication du code Java
* Création de la clé pour API Google Map
* Méthodes java d’accès aux données
* Utilisation « endpoint » pour accéder au web service
* Le reste tu sais mieux que moi
* Peut-être faire une troisième partie axée plus programmation et rester axé sur la BDD pour cette partie 2

## Développement des interfaces

Notre code se compose de 4 activités :

* **Page d’authentification**

Lorsque l’utilisateur démarre l’application il doit se connecter avec son identifiant et son mot de passe. Ces derniers lui seront transmis directement, ce n’est donc pas à lui de s’inscrire pour utiliser l’application.

* Screenshot

Note : est-ce qu’on met des bouts de codes à chaque fois (ce qui se passe pour la connexion, quand on appuie sur le bouton alerte) ou est ce qu’on fait une partie à part.

* **Page de paramétrage de la tournée**

Avant de pouvoir démarrer sa tournée, il est nécessaire de renseigner certaines informations telles que : le camion utilisé, le chauffeur du camion, les rippers et le numéro de la tournée. Une fois que l’utilisateur clique sur le menu *Démarrer* on enregistre la date de début de la tournée et on affiche l’interface suivante.

* Screenshot
* **Page affichant la position courante**

Une fois la tournée lancée, l’utilisateur peut voir sa position actuelle sur une Google Maps avec pour indication complémentaire, les coordonnées de celle-ci. Sur cette même interface se trouve deux boutons :

* Le bouton *Fin* permettant d’indiquer que la tournée est finie.
* Le bouton *Alerte* permettant de prévenir les autres camions qu’un problème est survenu. Ils recevront une notification permettant d’indiquer les coordonnées du camion en difficulté.
* Screenshot
* **Page affichant le chemin parcouru**

La dernière interface disponible permet à l’utilisateur de consulter le trajet qu’il a parcouru depuis le début de la tournée. Il peut, s’il le souhaite réinitialiser la carte (réinitialise début tournée ou pas ?)

* Scréeenshot

# Développement de l’application web d’administration

Comme indiqué précédemment, nous avons développé une application web permettant de traiter les données enregistrées en base. Pour cela, nous avons utilisé l’API JavaScript de Google Maps.

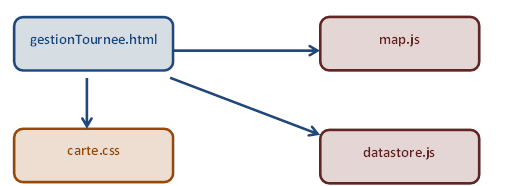
## Utilisation des APIs

### Organisation du code

La partie Web est une partie moins conséquente que la partie android. Il s’agit d’une seule page HTML, tous les traitements sont traités en JavaScript :

* Appel à l’API de Google
* Traitement des évènements de l’utilisateur
* Appel à l’API d’accès à la BDD avec traitement de la réponse JSON reçue

L’organisation de notre code peut être représentée comme suivant, en figure X.



La structuration de la page est définie d ans le fichier *gestionTournee.html.* La présentation est gérée par notre feuille de style *carte.css.* Le fichier *map.js* gère gestion de la carte (*affichage, ajout des tournées,…*). Enfin, *datastore.js* contient l’ensemble des appels à notre API Google.

### API web de Google Maps

Afin de bénéficier des différents services de Google Maps (*carte avec géolocalisation*) nous avons dû faire appel à cette API.

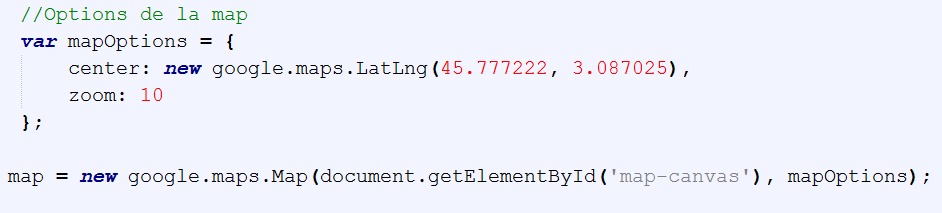
Pour cela, tout comme pour l’API Android, il est nécessaire de se créer une clé.

*src="https://maps.googleapis.com/maps/api/js?key=****API\_KEY****">*

Une fois l’inclusion du script ajouté, il nous est possible d’accéder à l’ensemble des fonctionnalités de la carte de Google. Il est nécessaire de paramétrer l’affichage de notre carte :

* On définit le centre de notre carte (*Pour nous, le centre sera donc Clermont-Ferrand*)
* On détermine le niveau de zoom que l’on souhaite appliquer à notre carte

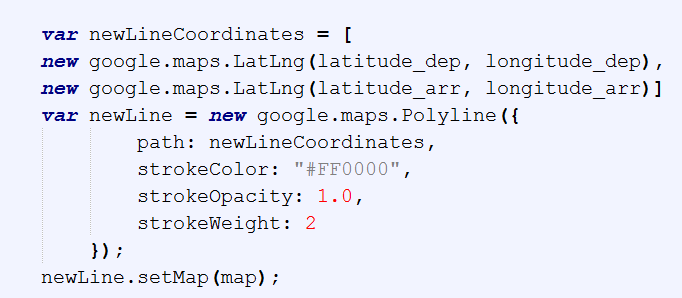
Par exemple, si nous voulons obtenir une carte centrée sur Clermont-Ferrand, le code adapté est :



La figure X permet d’initialiser notre carte. On crée notre objet *Map* en indiquant, d’une part, l’emplacement où insérer la carte dans notre page HTML, puis ensuite la liste des options à appliquer. Le résultat est présenté en figure X.



Le but, ensuite, est de pouvoir afficher la tournée qu’a effectuée le camion, en fonction des coordonnées sauvegardées en base. Pour cela, nous avons utilisé la méthode *Polyline* comme présenté en figure X.



On stocke les coordonnées de départ et d’arrivée de la ligne que l’on souhaite tracer, dans un tableau. Ce tableau est ensuite passé en paramètre de la méthode *Polyline* ainsi que le style d’apparence de notre trait. Nous répétons cette opération pour chaque couple de coordonnées composant la tournée.

L’API de Google n’a pas été la seule API dont nous avions besoin. Il nous a fallu faire appel à l’API que nous avions déployée sur le Cloud de Google.

### Appel de l’API d’accès aux données

Pour pouvoir utiliser notre API dans notre fichier JavaScript, voici les différentes étapes à suivre.

1 – Inclure dans notre fichier le script [*https://apis.google.com/js/client.js?onload=init*](https://apis.google.com/js/client.js?onload=init)*"*

2 – Implémenter la méthode *init* pour paramétrer notre appel :

On renseigne l’URL de notre API, puis on fait appel à *gapi.client.load* qui permet de charger l’API

3 – Appeler les méthodes qui nous intéressent :

Par exemple, si nous voulons récupérer la liste de tous les camions, nous devons écrire :

*gapi.client.sictomApi.getAllCamions().execute(function(resp){}) ;*

4 – Traiter le retour au format JSON

Si nous gardons notre exemple, l’appel à notre requête va nous retourner une liste JSON comme suit (valeurs de tests) :

200 OK

- Show headers -

{

"items": [

{

"id": "5093108584808448",

"nom": "Camion 2",

"poidsCourant": 0,

"poidsMax": 80,

"kind": "sictomApi#resourcesItem"

},

{

"id": "5656058538229760",

"nom": "Camion 1",

"poidsCourant": 40,

"poidsMax": 100,

"kind": "sictomApi#resourcesItem"

},

{

"id": "5762637883244544",

"nom": "Camion 3",

"poidsCourant": 10,

"poidsMax": 120.5,

"kind": "sictomApi#resourcesItem"

}

],

"kind": "sictomApi#resources",

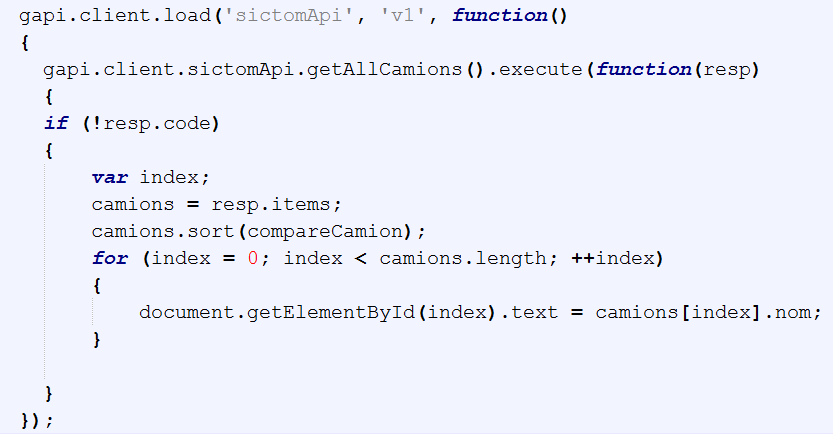
"etag": "\"vYcKJ78Wy\_0AVxb6KZe4OfzOAXo/j-gj8caktnRRxWZArzu4W-p1B6o\""

}

Le code HTML 200 nous indique que l’appel s’est bien passé. La réponse se compose d’une liste nommée *items* qui se compose de 3 camions : *Camion 2*, *Camion 1, Camion 3.* Chacun possède un identifiant, son poids courant, sa capacité maximale et le type de l’élément.

Nous avons codé, en JavaScript, diverses méthodes permettant de trier nos listes selon les critères qui nous intéressaient.

Voici la manière dont nous avons converti ceci en objet JavaScript :



Ici, nous récupérons notre liste *items* que nous parcourons après l’avoir triée. Pour chaque élément nous ajoutons le nom du camion dans notre fichier HTML afin de remplir notre menu dynamiquement.

### Présentation du résultat

Lorsqu’il se connecte, l’administrateur clique sur le camion qui l’intéresse. S’il est actuellement en tournée, celle-ci sera affichée sur la map, sinon on affichera la dernière en date. Toutes les informations relatives à la tournée sont affichées dans la partie droite de l’écran. L’utilisateur peut alors choisir de voir d’autre tournées, plus anciennes, effectuées par le camion. Pour cela, il doit sélectionner la date qui l’intéresse, et les tournées disponibles seront proposées. La figure X montre l’ensemble de ces fonctionnalités.

//Screenshot

## Traitement des informations récoltées

### Présentation de Google Chart

* Les statistiques qui nous intéressaient
* Google Chart
* Présentation résultat