CAZALIS Pauline – VON TOLKACZ Karol – JORET Eddy

Ville connectée

Pour une amélioration de la qualité de vie dans la ville de Pau

Rapport projet tutoré



Remerciement

Nous remercions Fabien LAMAS, Nicolas DUBOIS, Mikaël LEFEVRE, Guillaume FAUVET, Thomas ANSELMI, Lamia AMRANI et Lucas PAUZIES, encadrant de ce projet, pour leur temps et leur aide.

Nous tenons aussi à remercier toutes les personnes ayant aidée lors de la rédaction de ce rapport de projet tutoré.

Table des matières

[Introduction 3](#_Toc71317194)

[I) Présentation du projet 4](#_Toc71317195)

[1-1 – Contexte 4](#_Toc71317196)

[1-2 – Problématique 4](#_Toc71317197)

[1-3 – Description du projet 5](#_Toc71317198)

[II) Analyse du projet 6](#_Toc71317199)

[2-1- Cahier des charges 6](#_Toc71317200)

[2-2 – Analyse des données 7](#_Toc71317201)

[III) Présentation du matériel 9](#_Toc71317202)

[3-1 – Arduino 9](#_Toc71317203)

[3-2 – Capteurs 10](#_Toc71317204)

[3-3 – Logiciels et environnement de travail 11](#_Toc71317205)

[IV) Déroulement du projet 13](#_Toc71317206)

[4-1 – Planning via les diagrammes de Gantt 13](#_Toc71317207)

[4-2 – Répartition des tâches 13](#_Toc71317208)

[V) Présentation du travail réalisé 14](#_Toc71317209)

[5-1 – Gestion des poubelles 14](#_Toc71317210)

[5-2 – Gestion de l’éclairage public 17](#_Toc71317211)

[VI) Difficultés rencontrées 20](#_Toc71317212)

[6-1 – Retard sur le planning initial 20](#_Toc71317213)

[6-2 – Difficultés techniques 20](#_Toc71317214)

[6-3 – Changements apportés au projet 20](#_Toc71317215)

[Conclusion 22](#_Toc71317216)

[Annexes 23](#_Toc71317217)

# Introduction

Durant notre second semestre de Master Technologie de l’Internet à l’université de Pau et des Pays de l’Adour, nous avons été amenés à réaliser un projet tutoré dans le cadre de notre module « Gestion et réalisation de projet ».

Ce projet devait être réalisé en groupe. Les membres de notre groupe sont :

* Pauline CAZALIS
* Eddy JORET
* Karol VON TOLKACZ

Il a été proposé et encadré par Fabien LAMAS, Nicolas DUBOIS, Mikaël LEFEVRE, Guillaume FAUVET et avec l'aide de Thomas ANSELMI, Lamia AMRANI et Lucas PAUZIES, des ingénieurs logiciels de Capgemini (entreprise de services du numérique français basé à Pau).

Ce projet a pour but de partir d’une solution déjà existante d’agence connectée et d’étendre ce système à la ville de Pau pour améliorer sa qualité de vie ainsi que sa dépense énergétique.

Nous avons été deux groupes à travailler autour de ce projet. Le premier a concentré son travail sur l’urbanisme de la ville. Notre groupe, lui, s’est orienté sur l’environnement.

Nous avons donc commencé la mise en place de notre solution en respectant la problématique suivante : Quels changements pouvons-nous apporter pour améliorer la qualité de vie de Pau ?

Afin de structurer notre rapport écrit, nous allons dans un premier temps faire une présentation globale de notre projet, nous ferons ensuite la description du matériel utilisé et enfin nous présenterons de façon détaillé le travail réalisé et les difficultés rencontrées.

# Présentation du projet

## 1-1 – Contexte

Chaque année en France, un habitant produit en moyenne plus de 500kg de déchets. Malheureusement, ce nombre continu d’augmenter au fil des années.

A cause de la crise sanitaire, la vente de repas à emporter a considérablement augmenté.

De ce fait, les déchets s’entassent dans les poubelles de la ville, obligeant les passants à essayer de trouver une place pour jeter leurs déchets. Certains n’hésitent pas à les poser à côté des poubelles ce qui a pour effet de gâcher le paysage de la ville.

A l’heure actuelle aucune solution n’est mise en place pour remédier à ce problème.

Concernant l’éclairage public de la ville de Pau, en 2009 la consommation annuelle de l’éclairage public était de l’ordre de 2,9 mégawatts[[1]](#footnote-1). Cela correspond à une production de CO (monoxyde de carbone) de 1518 tonnes.

En 2017, la ville a décidé de réduire sa consommation électrique en changeant les ampoules actuelles par des LED.

Il est donc primordial de pouvoir gérer ces deux cas en temps réel afin de pouvoir améliorer la qualité de vie des habitants tout en réduisant les coûts engendrés par la ville.

## 1-2 – Problématique

Ce projet, répartis entre les deux groupes, avait pour objectif la mise en place d’une « Ville Connectée » ou « Ville intelligente » c’est-à-dire une ville où l’utilisation de capteurs permet la récupération de données. Son objectif est l’amélioration de la qualité des services urbains et la réduction de leurs couts.

Ainsi notre groupe, avait deux tâches principales à réaliser :

* Gestion des déchets : détecter si une poubelle est remplie ou non
* Gestion de l’éclairage public : allumer ou éteindre des lampadaires en fonction de la luminosité extérieure

## 1-3 – Description du projet

Nous avons repris le projet proposé l’année dernière. Il consistait en la création d’une agence, équipé d’une solution de surveillance de données, se voulant être une prémisse à l’élaboration d’une solution à l’échelle d’une ville.

Celle-ci nous a permis de construire la base de notre projet tout en apportant des changements et des améliorations afin de pouvoir répondre à notre problématique.

Ainsi, pour pouvoir extraire et interpréter les données voulus, nous avions à disposition une carte *Arduino* ainsi que différents capteurs et logiciels, servant à récupérer, traiter et stocker ces données.

# Analyse du projet

## 2-1- Cahier des charges

Contexte :

Ce projet est un POC (proof of concept) ayant pour objectif futur de pouvoir répondre à des opportunités de Ville Connectée avec la ville de Pau ou des alentours.

Objectifs :

Les objectifs de celui-ci sont :

* Récupérer les données des différents capteurs
* Mettre en place une architecture pour analyser les données récupérées
* Afficher les données analysées sur une page web

Contraintes :

* Contraintes techniques :

Tous les mardis, nous avions une réunion en ligne avec nos encadrants. Celle-ci nous permettait de discuter du projet afin de parler de nos avancées mais aussi de nos problèmes rencontrés. A l’issu de ces réunions un planning pour la semaine suivante était mis en place.

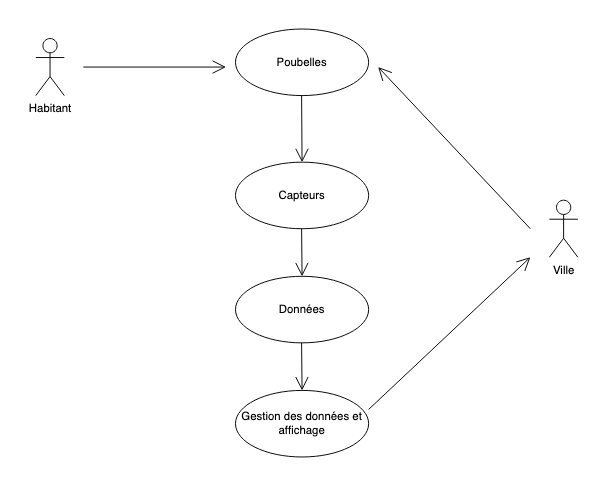
A cause de la crise sanitaire nous n’avons pas pu travailler ou nous voir en personne. Nous avons donc travaillé chacun de notre côté avec nos propres ordinateurs.

* Contraintes de temps :

Nous devions finir le projet et le rapport pour le 12 mai, ainsi que préparer une soutenance orale pour le 25 Mai.

## 2-2 – Analyse des données

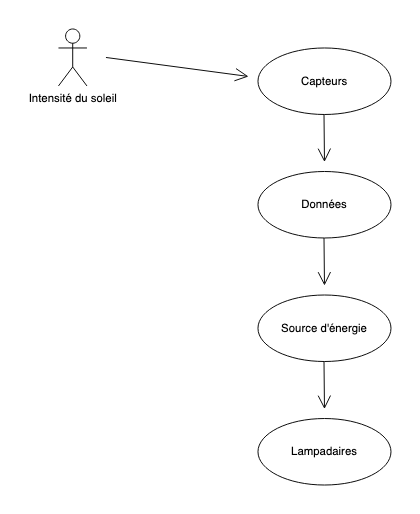
Dans cette partie nous allons présenter nos deux tâches principales à l’aide de diagrammes de cas d’utilisation.



Nous avons deux acteurs principaux :

* Les habitants de la ville qui se contentent de placer leurs déchets dans les poubelles
* La ville, qui gère les poubelles de la ville et regarde les données affichées.

Chaque poubelle possède un capteur de pression attitré, qui récupère la pression des déchets contenues à l’intérieur. Celui-ci envoie les données qui seront ainsi affichés et cela permettra à la ville d’agir en conséquence.



Dans ce diagramme nos capteurs récupèrent l’intensité lumineuse du soleil suivant le passage jour/nuit ou inversement. Cet évènement fournira une certaine donnée qui agira sur les lampadaires en les éteignant ou en les allumant.

# Présentation du matériel

Dans le but de répondre au mieux aux besoins techniques de notre projet, l’utilisation de plusieurs outils a été nécessaire. Ceux-ci sont présentés dans cette partie.

## 3-1 – Arduino

Nous disposons d’une carte *Arduino NodeMCU ESP8266* de la marque *Velleman*, avec son câble d’alimentation USB/Micro-USB.

C’est une carte avec circuit intégré à un microcontrôleur avec connexion Wi-Fi, composé de 10 broches GPIO (General Purpose Input/Output), que l’on utilisera avec le langage C++ avec l’IDE Arduino.

Chaque GPIO peut être PWM (Pulse Width Modulation), I2C (Inter-Integrated Circuit) ou encore 1-Wire (adressage et utilisation de plusieurs capteurs à communication série sur une seule et même broche numérique).

Elle est aussi composée de deux boutons poussoirs rst (reset, redémarrage de la carte) et flash (permet de lancer une séquence de flasharge de la mémoire) et d’une antenne PCP qui permet d’améliorer la portée.



Arduino NodeMCU ESP8266 Velleman

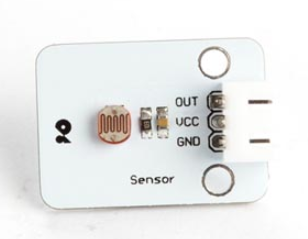
Cette carte nous permet de recevoir des informations fournis par un ou plusieurs capteurs et de les transmettre vers un logiciel externe.

L’*Arduino ESP8266* étant équipée d’un module wifi, cela nous permettra d’envoyer les données reçus (sous un format type -> Précision du format) via une connexion wifi, bien plus pratique qu’une connexion filière à l’échelle d’une ville.

Nous disposons aussi de 21 câbles pour la connexion de l’*Arduino* aux capteurs.

## 3-2 – Capteurs

3 capteurs nous ont été fournis pour obtenir les données voulus :

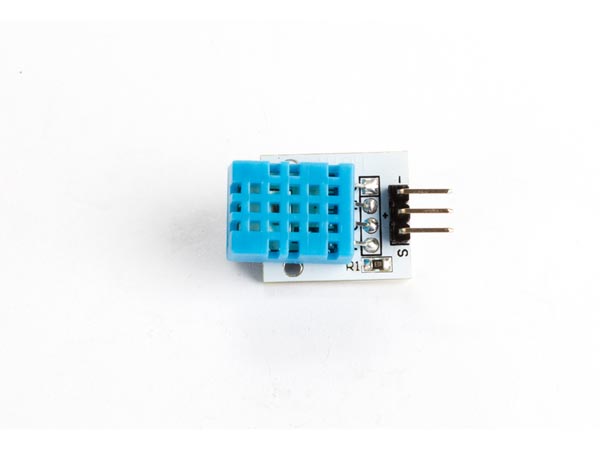
* 1x capteur photosensible qui utilise une photorésistance pour produire une tension de sortie.

Capteur photosensible Velleman

Une photorésistance est une résistance dont la valeur change en fonction de la lumière qu’elle reçoit.

Grâce à ce capteur, nous allons pouvoir récupérer la luminosité actuelle de la ville et donc gérer l’éclairage public.

* 2x capteurs de température (NTC) et d’humidité (DHT11).



Capteur de température et d'humidité digital DHT11 Velleman

Il va nous permettre de récupérer la température et l’humidité présente dans l’air de la ville.

## 3-3 – Logiciels et environnement de travail

Afin de pouvoir exploiter et afficher nos résultats, nous avons utilisé plusieurs logiciels :

Arduino IDE : c’est un logiciel de programmation pour la carte *Arduino*. Il va nous permettre de créer le code pour faire fonctionner la carte avec les différents capteurs.



Broker MQTT Mosquitto : c’est un protocole de messagerie de type publish-subscribe. Il va permettre à notre *Arduino* d’envoyer et/ou de recevoir des messages en se connectant à un « topic » pour éviter que les messages ne se mélanges pas.



Node-RED : c’est un outil de programmation, basé sur un navigateur, qui permet de développer des objets connectés à l’aide de flux. Il va nous permettre de simuler les données reçues pour la gestion des poubelles par un capteur fictif.



MongoDB Atlas : c’est une base de données distribuée. Elle nous permet de stocker les données reçues par les différents capteurs pour pouvoir ensuite avoir un historique de celles-ci.

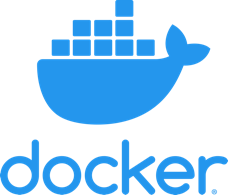


Express JS : C’est un framework (permet de simplifier le travail des developpeurs en offrant une architecture « prête à l’emploi ») qui fournit les fonctionnalités d’applications web et mobile fondamentales. Il est basé sur *Node Js* (environnement d’exécution qui permet d’utiliser JavaScript côté serveur).

Il va nous permettre de recevoir les données des capteurs et les traiter pour pouvoir les envoyer à la fois sur notre page web et sur la base de données pour l’historique.



Angular : C’est un framework, écrit en JavaScript, qui permet la création d’applications web. C’est grâce à lui que nous allons pouvoir afficher nos données pour qu’elles soient visibles de l’utilisateur.



Docker : C’est une plateforme logicielle qui permet de concevoir, tester et déployer des applications rapidement grâce à des conteneurs logiciels. Les conteneurs rassemblent tous les éléments nécessaires aux logiciels (bibliothèques, outils systèmes…) afin de pouvoir les exécuter dans n’importe quel environnement.

Ainsi, nous avons pu regrouper notre Broker MQTT, *Node-RED* ainsi que *Express* pour qu’ils puissent s’exécuter en même temps et correctement.

Pour pouvoir organiser notre projet, nous avons utilisé deux logiciels :



Github : c’est un service web d’hébergement et de gestion de développement de logiciel. Il nous a permis de stocker nos différents fichiers en ligne afin que tous les membres du groupe puissent y avoir accès.



Trello : c’est un outil de gestion de projet en ligne. Nos encadrants ont pu lister les différentes tâches que nous avions à faire pour la réalisation de ce projet.

Nous avons pu, grâce à cet outil, nous organiser pour la réalisation de ce projet.

# Déroulement du projet

## 4-1 – Planning via les diagrammes de Gantt

Pour réussir notre projet tutoré, il nous fallait absolument passer par une planification efficace. Celle-ci a été mise en place via un planning prévisionnel.

Le planning prévisionnel est un outil de travail qui nous permet de visualiser la manière dont le projet va s’organiser dans le temps et les différentes étapes à suivre pour atteindre les objectifs. Il permet de construire et planifier les étapes nécessaires à la réalisation du projet et de les visualiser dans le temps (Annexe 1).

Pour comparer avec ce qui aura été réellement fait durant le projet, nous avons fait un planning réel avec l’ajout de nouvelles tâches dont nous n’avons pas pensé lors de la création du planning prévisionnel (Annexe 2).

Pour les modéliser, nous avons utilisé le logiciel *GanttProject*, qui nous permet de créer des diagrammes de Gantt.

## 4-2 – Répartition des tâches

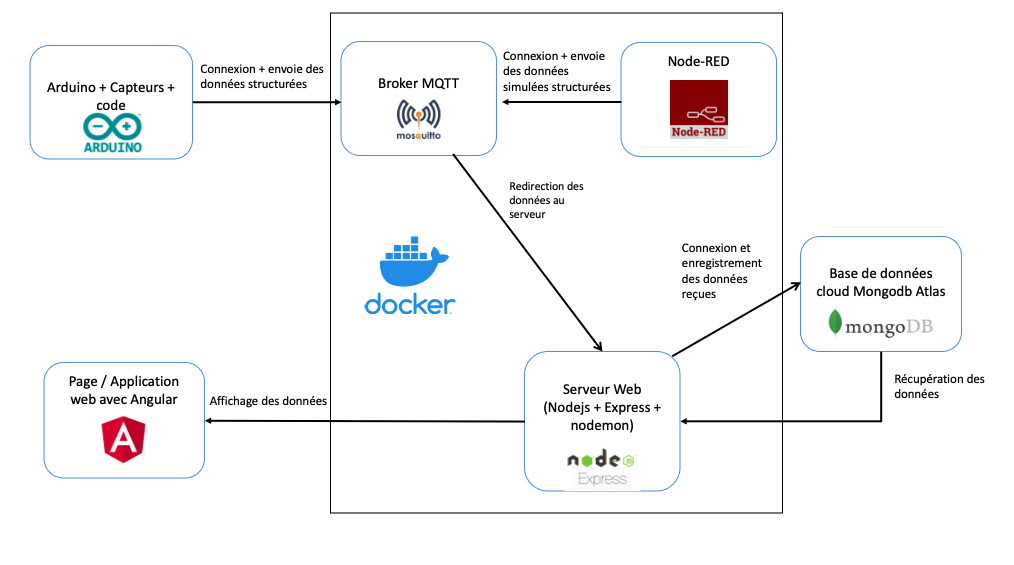
Après avoir établi ce planning, nous avons réparti ces différentes tâches entre nous trois.

Elles ont été partagées en fonction des compétences de chacun :

* Réflexion sur le sujet et proposition de solutions (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)
* Écriture du code pour les différents capteurs (Eddy JORET)
* Connexion MQTT de l’*Arduino* à *Node-RED* (Eddy JORET)
* Simulation des données sur *Node-RED* (Eddy JORET)
* Création du conteneur *Docker* (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)
* Création du serveur *Express* (*Node Js*) (Pauline CAZALIS)
* Récupération des données de MQTT au serveur (Pauline CAZALIS, Eddy JORET)
* Envoie des données du serveur à la base de données *MongoDB* (Pauline CAZALIS, Eddy JORET)
* Récupération des données du serveur vers *Angular* (via socket) (Pauline CAZALIS)
* Création du dashboard sur *Angular* (Pauline CAZALIS)
* Rédaction du rapport (Pauline CAZALIS, Karol VON TOLKACZ, Eddy JORET)

# Présentation du travail réalisé

Le schéma ci-dessous représente le parcours des données allant du capteur jusqu’à l’enregistrement et leur affichage :



## 5-1 – Gestion des poubelles

Nous devions trouver une solution simple et efficace à cette demande. Nous avons donc décidé de placer un capteur de pression directement à l’intérieur de la poubelle.

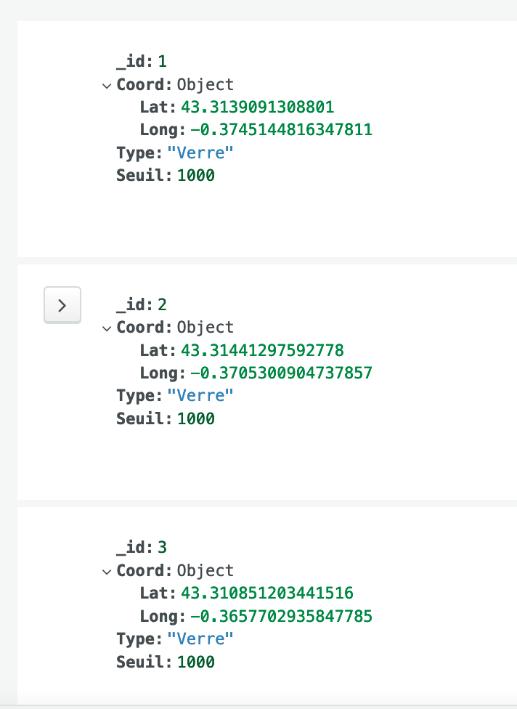
Ne possédant pas le capteur de pression pour l’*Arduino*, nous avons donc dû le simuler.

🡪 Création de la base de données

La ville possède 3 types de poubelles : verre, déchets communs et recyclables.

Les données qui nous sont intéresse sont les coordonnées de la poubelle, son type, son seuil, sa pression ainsi qu’un moyen d’identification. Pour cela nous avons créé deux types de collections :

* poubelle\_id : contient les données globales des poubelles de Pau. Elles possèdent un id, des coordonnées, un type et un seuil.



Exemple du contenu de poubelle\_id

* poubelle\_hist : contient les données de temps réel des poubelles de Pau et est relié à la première collection grâce à la clé Poubelle\_id.



Exemple du contenu de poubelle\_hist

🡪 Création du code de simulation

Pour pouvoir récupérer les données du capteur, nous avons utilisé deux moyens différents :

Le premier avec un véritable capteur et un *Arduino* et le second avec une simulation sur *Node-RED* via différentes fonctions codées en JavaScript.

Pour pouvoir utiliser le capteur et l’*Arduino*, nous avons utilisé *Arduino IDE*.

Nous avons dû connecter l’*Arduino* à une connexion internet via WiFi par le module correspondant de l’*ESP8266* pour pouvoir accéder au broker MQTT et s’y connecter. Ainsi nous avons pu y émettre les données reçues du capteur avec leur date de réception qui ont été au préalablement changé dans un format JSON.

Pour simuler sur *Node-RED*, nous avons créé un flux spécifique aux données liés à la gestion des poubelles. Dans ce flux, nous avons utilisé des variables liées au flux en lui-même pour y avoir accès depuis n’importe quelle fonction.

Nous utilisons des nodes « inject » qui permet de produire un message spécifique au bout d’un temps donné et de répéter cet envoi périodiquement. Grâce à celles-ci, nous pouvons produire une seule fois les données initiales par rapport aux poubelles via deux fonctions différentes :

* Une va produire les données de localisation (Coord), de type (Type) et de seuil (Seuil) pour un identificateur (\_id) de poubelle.
* Une va produire la pression initiale avec son heure de création en y associant l’id de la poubelle.

Ces données crées seront envoyées ensuite au topic correspondant : poubelle\_id ou poubelle\_hist.

Une fois cela fait, nous utilisons d’autres de ces nodes pour créer une boucle nous servant à mettre à jour la pression de chaque poubelle et émettre cette nouvelle valeur sur le topic voulu sous un format JSON défini lors de la création de la pression initiale.

🡪 Création du serveur

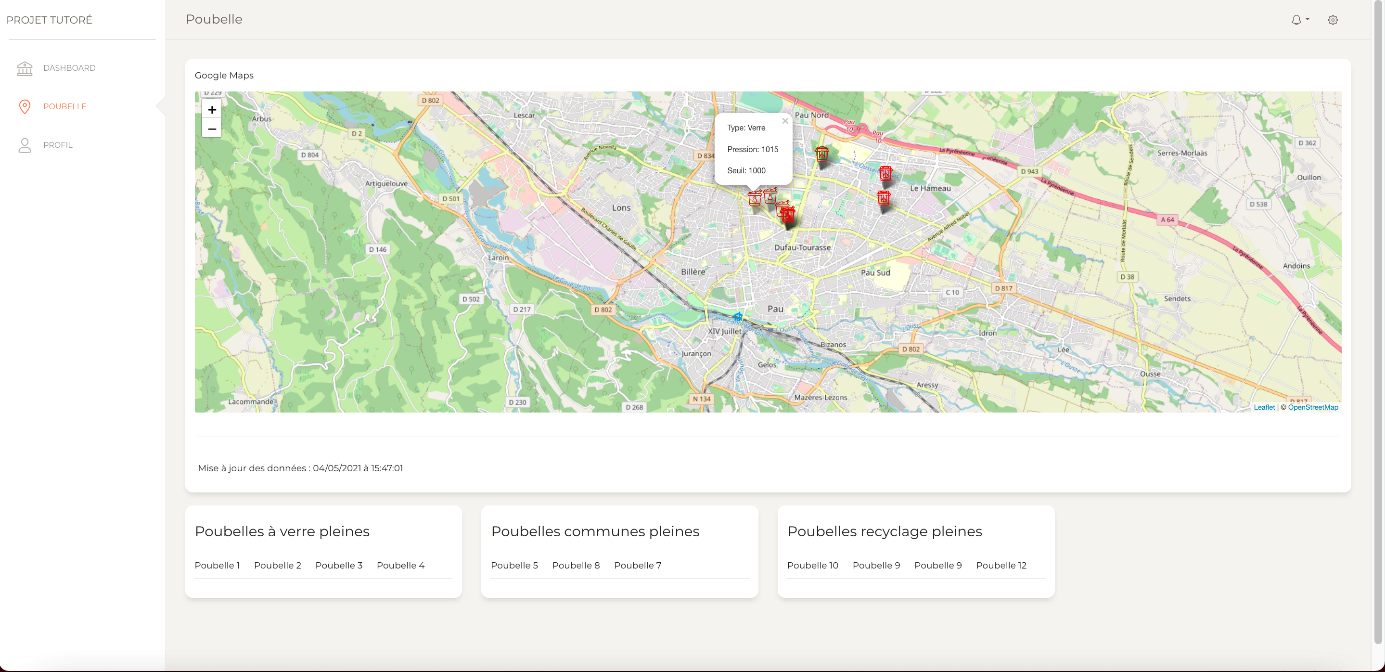
Notre serveur nous permet de récupérer nos données dans un message via le broker MQTT qui est reçu sous format texte. Le message, étant écrit au format JSON, il ne nous reste plus qu’à le transformer en ce même format pour les sauvegarder dans la base de données.

Le serveur pourra aussi récupérer les données de la base de données et seront envoyées sur notre page web *Angular* via une socket.

Une socket est un protocole réseau permettant la création d’un canal de communication à double sens entre un serveur et un navigateur.

🡪 Création de la page web

Après avoir reçu les données depuis la socket, *Angular* va stocker ces données au bout d’un temps défini pour pouvoir les afficher. Nous avons décidé d’afficher le type, le seuil et la pression de chaque poubelle de façon séquentielle sur une carte (crée par la bibliothèque *Leaflet*). L’utilisateur pourra donc ainsi surveiller en temps réel l’évolution de celle-ci sans être distrait par des données pas importante.



Notre "dashboard" indiquant l'état en temps réel des poubelles de la ville de Pau

## 5-2 – Gestion de l’éclairage public

Notre deuxième tâche a été de proposer une solution pour réduire la consommation électrique et le coût concernant l’éclairage public de la ville de Pau.

Nous avons opté pour la solution la plus simple : le capteur photo sensible va détecter s’il fait jour ou nuit de manière à activer ou non le lampadaire.

🡪 Création de la base de données

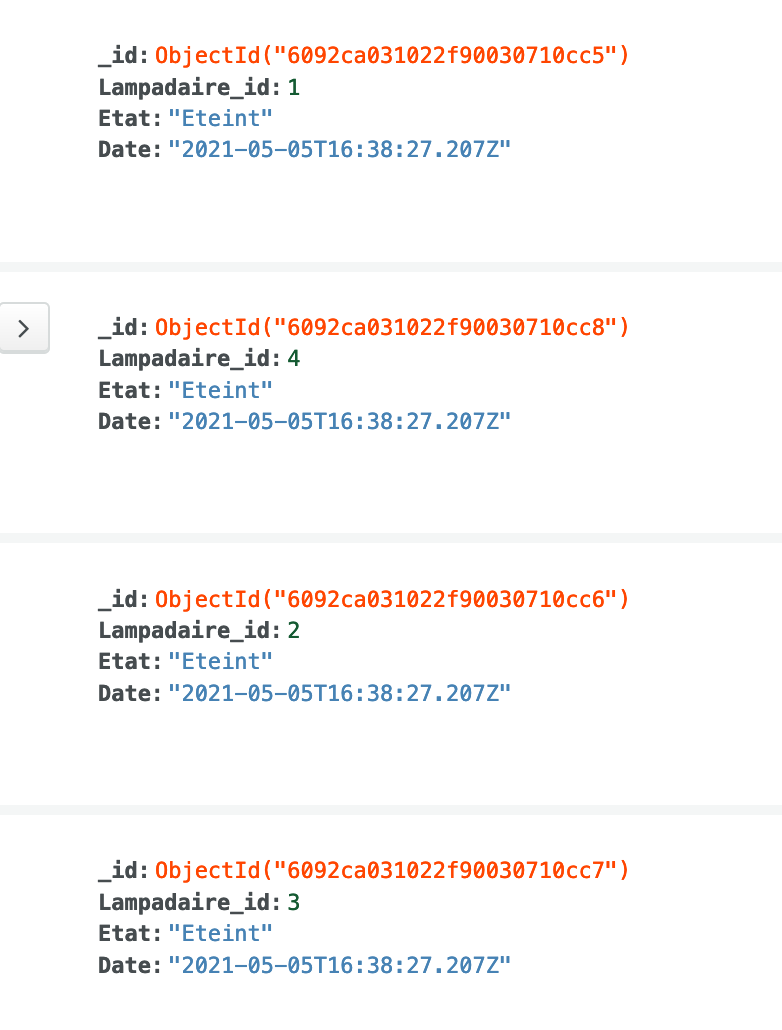
Notre base de données comporte deux collections :

* lampadaire\_id : contient l’identificateur et les coordonnées du lampadaire.



Exemple du contenu de lampadaire\_id

* lampadaire\_hist : contient les données lors du changement jour/nuit avec l’heure de changement et est relié à la première collection grâce à la clé Lampadaire\_id.



Exemple du contenu de lampadaire\_hist

🡪 Création du code

La conception du code sera la même que la simulation des données des poubelles.

Cependant, ici, il ne faut pas envoyer les données au MQTT sur une période de temps défini mais uniquement lors du changement jour/nuit.

Lors de la création des données initiales, nous n’envoyons que l’identificateur du lampadaire ainsi que ses coordonnées sur le topic lampadaire\_id et pour les données historiques nous envoyons l’état initial éteins ainsi que l’id du lampadaire avec l’heure de la création de ce message sur le topic lampadaire\_hist.

Une fois cela fait, nous devons périodiquement simuler le temps qui a été passé pour atteindre les changements jour/nuit et envoyer le message correspondant sur le topic voulu.

Pour cela, nous avons défini des variables supplémentaires pour nous aider à simuler le temps passé durant la journée et la nuit ainsi que pour savoir à quel moment se trouve le changement. Cela permet d’éviter d’envoyer sur le MQTT l’heure et l’état voulu de chaque lampadaire alors qu’il a déjà été envoyé.

# Difficultés rencontrées

## 6-1 – Retard sur le planning initial

Le plus gros problème que nous avons eu avec le planning initial est la gestion des poubelles.

Il y a eu aussi un problème de communication entre les logiciels.

A cause des différents soucis techniques et de changement, cela nous a pris deux semaines de plus que ce que nous avions initialement prévu.

## 6-2 – Difficultés techniques

La première difficulté technique que nous avons rencontrée a été la connexion MQTT sur l’*Arduino*. Sans passer par *Docker*, nous n’avions pas accès au MQTT par une adresse IP mais uniquement par le nom « localhost », or avec l’*Arduino* nous n’avons pas pu nous connecter à ce protocole via ce nom.

Le second problème a été de revoir la base de données qui était initialement composé d’une seule collection. De ce fait la base de données n’était pas assez générique.

Le dernier problème a été de transmettre les données simulées par *Node-RED* au framework *Express* de *NodeJs* sans passer par *MongoDB Atlas*. Pour cela nous avons essayé avec des requêtes HTTP et WebSockets. Malheureusement nous n’avons pas pu réussir à faire fonctionner les requêtes HTTP et nous avons laissé tomber l’idée des WebSockets car ayant utilisé une socket de type « socket.io » sur le serveur, nous ne pouvons pas utiliser une de type WebSocket pour se connecter à celle-ci.

## 6-3 – Changements apportés au projet

Le problème de la connexion MQTT sur *Arduino* a été résolu une fois que nous avons défini un conteneur MQTT accessible par le nom défini dans le fichier dockerfile.

Pour pallier à la base de données qui n’était pas assez générique, nous avons utilisé deux collections et par conséquent nous avons dû changer les codes de simulations pour que les données correspondent aux données voulus de la base de données.

Pour réussir à envoyer les données de *Node-RED* au framework *Express* de *NodeJs*, nous avons décidé d’utiliser le protocole MQTT qui est aussi utilisé par les *Arduino* pour transmettre leurs données. De ce fait, nous avons supprimé le flux servant à insérer directement les données simulées aux collections de la base de données.

Nous avons décidé de créer un conteneur *Docker* global composé en plusieurs conteneur : *Node-RED*, MQTT et *NodeJs* avec le module nodemon (pour relancer le serveur automatiquement lors d’une modification). Le fait de les mettre dans un seul et même conteneur, cela permet de les lier et de les faire connaître l’un à l’autre pour établir des connexions de communications entre ceux-ci.

# Conclusion

Ce projet nous as permis de découvrir plusieurs solutions possibles d’utilisation de l’IOT pour améliorer des techniques à créer ou déjà existantes par rapport à l’écologie.

Pour mettre en œuvre ce projet, nous avons dû passer par des phases de réflexions, conceptions et créations suivant différents points de vue :

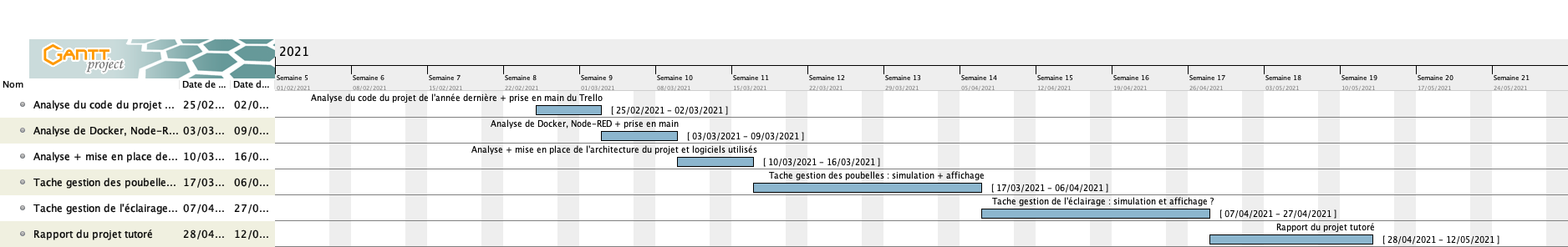
* Travail d’équipe : l’ambiance et la répartition des tâches se sont installés sans problème
* Communication : une bonne communication dans le groupe est nécessaire pour savoir ce que chaque membre a besoin pour exploiter et faire ressortir un meilleur résultat de sa partie. Cela permet une meilleure cohésion et adaptabilité entre les parties.
* Délai : cela nous a permis de voir qu’il est difficile de respecter un planning fait à l’avance à cause de toutes sortes de problèmes, techniques ou humains (différence de choix, …) par exemple.
* Nouvelles technologies : pour parvenir à faire ressortir notre application, nous avons appris à connaitre et à utiliser de nouvelles technologies comme *NodeJs* et *Angular*.

Afin d’optimiser l’application nous avons eu quelques idées possibles :

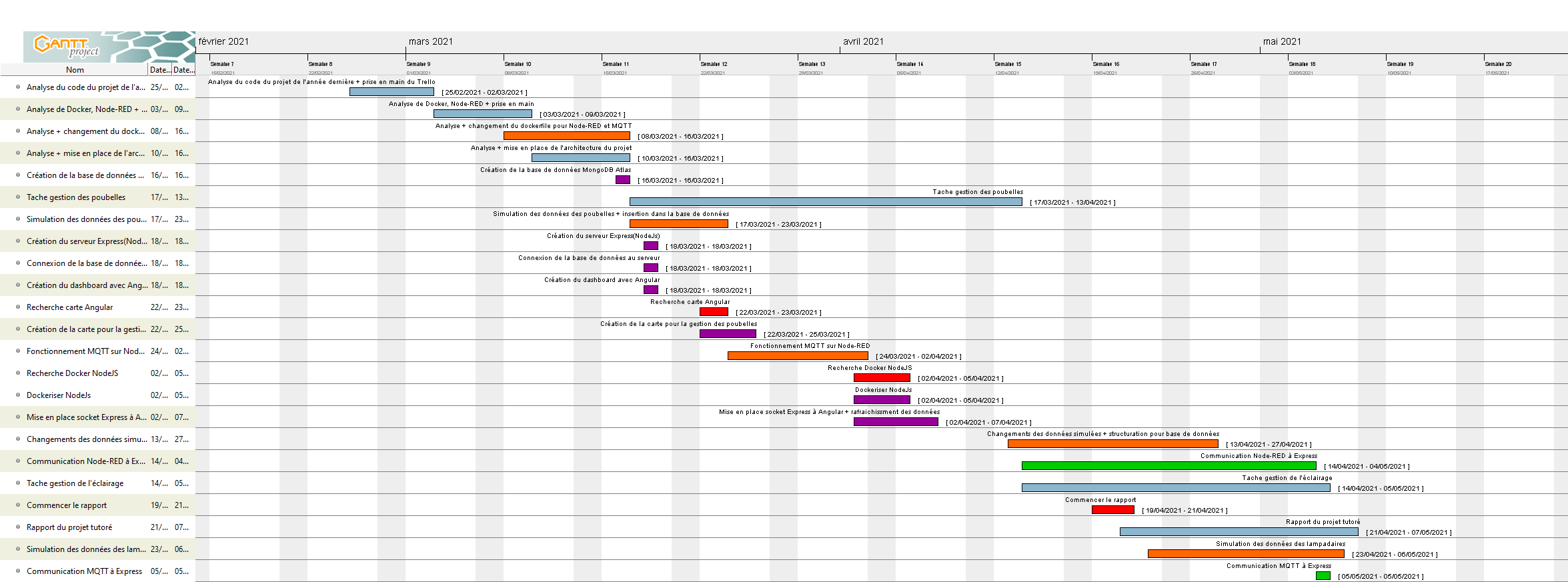
* Gestion des poubelles : faire une liste des poubelles triées par type et chacune sera cliquable pour s’afficher sur la map pour mieux s’y retrouver lors d’une concentration trop dense de poubelle.
* Gestion de l’éclairage : réduire encore la consommation d’électricité en optimisant l’éclairage avec la présence de capteurs de mouvement qui fera connaître aux lampadaires situés plus loin que quelqu’un arrive (passage de l’éclairage en mode normal en mode réduit et vice versa)

# Annexes

Annexe 1 : Planning prévisionnel



Annexe 2 : Planning réel



Légende :

Couleurs des taches :

* Bleu : groupe
* Vert : Eddy JORET et Pauline CAZALIS
* Orange : Eddy JORET
* Violet : Pauline CAZALIS
* Rouge : Karol von Tolkacz

1. Source : <https://www.larepubliquedespyrenees.fr/2010/11/13/un-meilleur-eclairage-pour-une-consommation-moindre,165788.php> [↑](#footnote-ref-1)