

# Projet de fin d'études

---

**R&D démonstrateur web** : le développement web full-stack d'un flux de travail boutique-logistique-entrepôt et de système IoT

Encadré par : Bruno Le Fellic, Vincent Ricordel

Rédigé par : Zijun PAN

POLYTECH NANTES - ETN5  
Mars - Août 2020

# Sommaire

---

Sommaire

Abstract

1. Introduction

1.1. Contexte

1.2. Objectives

1.3. À propos de SpikeeLabs

2. Technologies

2.1. Développement

2.1.1. Angular

2.1.2. Vue

2.1.3. Swagger

2.1.4. Flask

2.1.5. MongoDB

2.1.6. SQL Server

2.1.7. Cassandra

2.1.8. RabbitMQ

2.1.9. Kafka

2.1.10. Docker

2.2. Outils de coopération

2.2.1. GitLab

2.2.2. Teams

2.2.3. Trello

2.2.4. One Drive

2.2.5. Gantt

3. Projet ETL

3.1. Organisation

3.2. Génération des données fictives

3.3. Modifications des APIs Python

3.4. Angular pour l'Interface Homme-Machine

3.5. Test Unitaire

3.6. Déploiement en Docker

4. Projet IoT

4.1. Organisation

4.2. Normalisation

4.3. Consolidation

4.4. Kafka

4.5. Dashboard

4.6. Déploiement en Docker

5. Conclusion

# Abstract

---

Le premier thème de ce stage de fin d'études est sur le développement démonstrateur d'un flux de travail boutique-logistique-entrepôt. Mon travail comprend le développement de pages, la modification d'API et le déploiement de pages Web.

Dans la partie back, on utilise MongoDB, une base de données distribuée, universelle et basée sur des documents, pour stocker les données fictives générées par un script Python. L'API de ce projet est construite avec Python Flask et Swagger, qui nous permet de récupérer les données persistantes.

Dans la partie front, on utilise Angular, le framework front-end de Google, pour construire un site web structuré et évolutif. Typescript(un sur-ensemble de Javascript), CSS et HTML sont à la base de la construction d'un composant Angular. Angular Material, une bibliothèque de composants UI, est utilisée pour améliorer l'interface utilisateur.

Le logiciel Git est utilisé pour la synchronisation de projet avec le dépôt Git distant. Le front et le back sont connectés avec la structure JSON. Docker et docker-compose sont utilisés pour le déploiement de nos logiciels front et back à une machine Linux.

# 1. Introduction

---

Ce rapport explique mon stage de six mois chez SpikeeLabs à Rennes. SpikeeLabs est une entreprise qui propose à ses clients les concepts et les réalisations des systèmes de service d'information. J'ai participé au stage dans le département BillingLabs de l'entreprise, le but étant de réaliser la fonction et l'affichage de l'ensemble du processus du système de service, y compris API, Analytics et IHM.

## 1.1. Contexte

Mes projets de stage se compose de deux parties: le but des deux projets est de migrer le projet original de l'entreprise vers la nouvelle scène pour la réalisation, afin que notre fonction d'origine puisse être itérée, mise à jour et étendue dans la nouvelle scène.

Le premier projet est un système de gestion des télécommunications basé sur l'entreprise. Nous espérons migrer le système vers le scénario entrepôt-logistique-magasin pour construire un système de gestion ETL entrepôt-logistique-magasin.

Le deuxième projet est basé sur le système d'architecture distribuée de l'entreprise. Nous espérons faire migrer un système de télécom billing vers le scénario de piles de recharge de véhicules électriques. Grâce à la collecte d'informations du système distribué, nous collectons la file d'attente des messages du projet et effectuons certaines analyses big data sur les données.

Le temps consacré à chaque projet représente la moitié de l'ensemble du cycle de stage.

## 1.2. Objectives

Le but de mon stage est de fournir un soutien aux entreprises dans le processus de migration de nouveaux projets, et de conduire la recherche et le développement de certains projets. Dans ce processus, je connais les différentes technologies utilisées dans l'entreprise et les opérations quotidiennes de l'entreprise.

## 1.3. À propos de SpikeeLabs

Spikeelabs est immatriculée en 2016. C'est une entreprise ESN basée à Rennes mais sert pour les client autour de la France. Depuis sa création, SpikeeLabs connaît une forte croissance tant en termes de chiffre d'affaires.



Le logo de SpikeeLabs

Comme une entreprise ESN typique, Spikeelabs fournit les services sur 4 activités:

- Conseil: Analyse les existences ainsi que les besoins. Donner la conception, Créer le cahier des charges pour le produit du client
- Réalisation: Développement du architecture et projet par rapport aux besoins du client. migration pour la base de données, amélioration et test de fonctionnements.
- Intégration: Appliquer le produit de Spikeelabs au projet existé
- Support: Maintenance, évaluation et adaption pour un projet du client.



Les clients de SpikeeLabs

Aujourd'hui le chiffre d'affaire est arrivé à 4 million euro et il y a plus de 50 salariés qui travaillent dans le bureau à Rennes, à Paris et à Nantes.



Cartes des 3 sites SpikeeLabs en France

Plus d'informations peuvent être trouvées sur ce lien: <https://www.spikeelabs.fr/>.

## 2. Technologies

---

### 2.1. Développement

Dans nos projets, nous utilisons des technologies auxiliaires pour faciliter notre développement.

#### 2.1.1. Angular

Afin de simplifier à la fois le développement et les tests de SPA (application d'une seule page), Angular peut nous fournir des architectures MVC (model-view-controller) et MVVM (model-view-viewmodel) côté client.

Le front-end de notre projet télécom original a été développé en utilisant Angular.



Logo d'Angular

Plus d'informations sur Angular peuvent être trouvées ici : <https://angular.io/>.

#### 2.1.2. Vue

Vue est un framework évolutif pour construire des interfaces utilisateur. Il s'agit d'un framework MVVM léger qui fournit une liaison de données efficace et un système de composants flexible via une API simple.

Afin de réaliser un développement rapide, nous utilisons le framework Vue léger et simple pour développer le Dashboard du projet IoT.



Logo de Vue.js

Plus d'informations sur Angular peuvent être trouvées ici : <https://vuejs.org/>.

#### 2.1.3. Swagger

Swagger est une suite d'outils de développement d'API à la fois puissants et faciles à utiliser pour les équipes et les individus, permettant le développement sur l'ensemble du cycle de vie de l'API, de la conception et de la documentation aux tests et au déploiement.

Afin de faciliter la réalisation des fonctionnalités de l'API, nous utilisons le Swagger Codegen pour la génération de code et Swagger UI pour la visualisation.



Logo de Swagger

Plus d'informations sur Swagger peuvent être trouvées ici : <https://swagger.io/>.

#### 2.1.4. Flask

Flask est un framework d'application web WSGI (Web Server Gateway Interface) léger. Il est conçu pour rendre la mise en route rapide et facile, avec la possibilité de s'adapter à des applications complexes.

Afin de réaliser un hébergement de services Web back-end, nous utilisons Flask pour nous aider à créer le serveur.

Plus d'informations sur Flask peuvent être trouvées ici : <https://palletsprojects.com/p/flask/> .

### 2.1.5. MongoDB

Afin de rendre l'intégration des données plus facile et plus rapide, nous utilisons MongoDB comme base de données pour développer des documents de type JSON avec des schémas dynamiques.



Logo de MongoDB

Plus d'informations sur MongoDB peuvent être trouvées ici : <https://www.mongodb.com/> .

### 2.1.6. SQL Server

SQL Server est un système de gestion de base de données (SGBD) en langage SQL incorporant entre autres un SGBDR (SGBD relationnel) développé et commercialisé par la société Microsoft.

Nous utilisons une base de données SQL Server pour stocker les informations relatives au compte.



Logo de SQL Server

Plus d'informations sur SQL Server : <https://www.microsoft.com/sql-server>

### 2.1.7. Cassandra

Cassandra est un SGBD de type NoSQL conçu pour gérer des quantités massives de données sur un grand nombre de serveurs. C'est une solution de stockage de données structurée distribuée populaire.

Dans notre projet IoT, nous l'utilisons pour stocker les données EDR (Event Detail Record).



Logo de Cassandra

Plus d'informations sur Cassandra peuvent être trouvées ici : <https://cassandra.apache.org/>

### 2.1.8. RabbitMQ

RabbitMQ est un logiciel d'agent de messages open source qui implémente le protocole Advanced Message Queuing (AMQP). Afin de permettre au projet de faire face à un volume plus élevé de demandes, nous avons utilisé RabbitMQ pour traiter les messages de demande.





Logo de RabbitMQ

Plus d'informations sur RabbitMQ peuvent être trouvées ici : <https://www.rabbitmq.com/>

### **2.1.9. Kafka**

Kafka peut fournir un mécanisme de file d'attente de messages pour améliorer la fiabilité des données de log.

Pour transmettre les données de log au réseau distribué Kafka afin de garantir la fiabilité, nous utilisons lensio/fast-data-dev, une configuration Kafka à part entière.



Logo de Kafka

Plus d'informations sur Kafka : <https://kafka.apache.org/>

### **2.1.10. Docker**

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs

Afin d'implémenter le déploiement de microservices pour ces deux projets, nous avons utilisé Docker dans le serveur pour déployer les microservices dans le conteneur Docker.



Logo de Docker

Plus d'informations sur Docker peuvent être trouvées ici : <https://www.docker.com/>

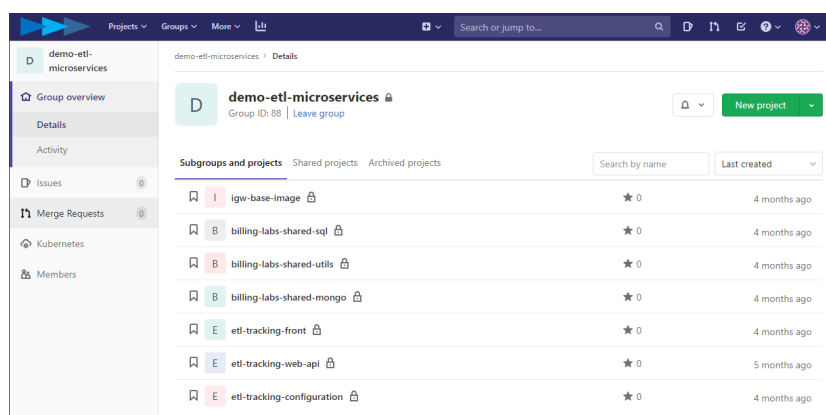
## 2.2. Outils de coopération

Bien que le projet de stage soit un projet personnel, je suis également entré en contact avec de nombreux outils de projets collaboratifs dans l'entreprise, tels que GitLab, Teams, Trello et One Drive.

### 2.2.1. GitLab

GitLab est un service d'hébergement de référentiel Git basé sur le Web. Il offre toutes les fonctionnalités de contrôle des révisions distribuées et de gestion du code source (SCM) de Git ainsi que l'ajout de ses propres fonctionnalités. Contrairement à Git, qui est strictement un outil de ligne de commande, GitLab fournit une interface graphique Web.

Il fournit également un contrôle d'accès et plusieurs fonctionnalités de collaboration comme le suivi des bogues, les demandes de fonctionnalités, la gestion des tâches et les wikis pour chaque projet.



Les projets ETL dans le Git de SpikeeLabs

Contrairement à GitHub, GitLab nous permet de créer gratuitement un entrepôt privé et de le déployer sur notre propre serveur. De plus, il intègre également des fonctions telles que CI (Continuous Integration).

### 2.2.2. Teams

Microsoft Teams est une application de collaboration qui permet à l'équipe de rester organisée et d'avoir des conversations au même endroit.

- Équipes: rechercher des canaux pour en faire partie ou réserver notre propre canal. Dans les canaux, nous pouvons organiser une réunion immédiate, discuter et partager des fichiers.
- Réunions: afficher toutes les réunions prévues de la journée ou de la semaine. Ou bien, planifier une réunion. Ce calendrier se synchronise avec le calendrier Outlook.
- Appels: dans certains cas, si l'organisation a installé cette fonctionnalité, nous pouvons appeler qui nous voulons à partir de Teams, même si ces personnes ne l'utilisent pas.
- Activité: consulter tous les messages non lus, @mentions, réponses, etc.

### 2.2.3. Trello

Trello est un outil de gestion de projet gratuit en ligne. Inspiré par la méthode Kanban de Toyota, Trello repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches. Les cartes sont assignables à des utilisateurs et sont mobiles d'une planche à l'autre, traduisant leur avancement.

Trello permet ainsi de travailler sur des projets collaboratifs depuis n'importe où dans le monde, tout en étant notifié sur les différents apports réalisés par les membres.

#### **2.2.4. One Drive**

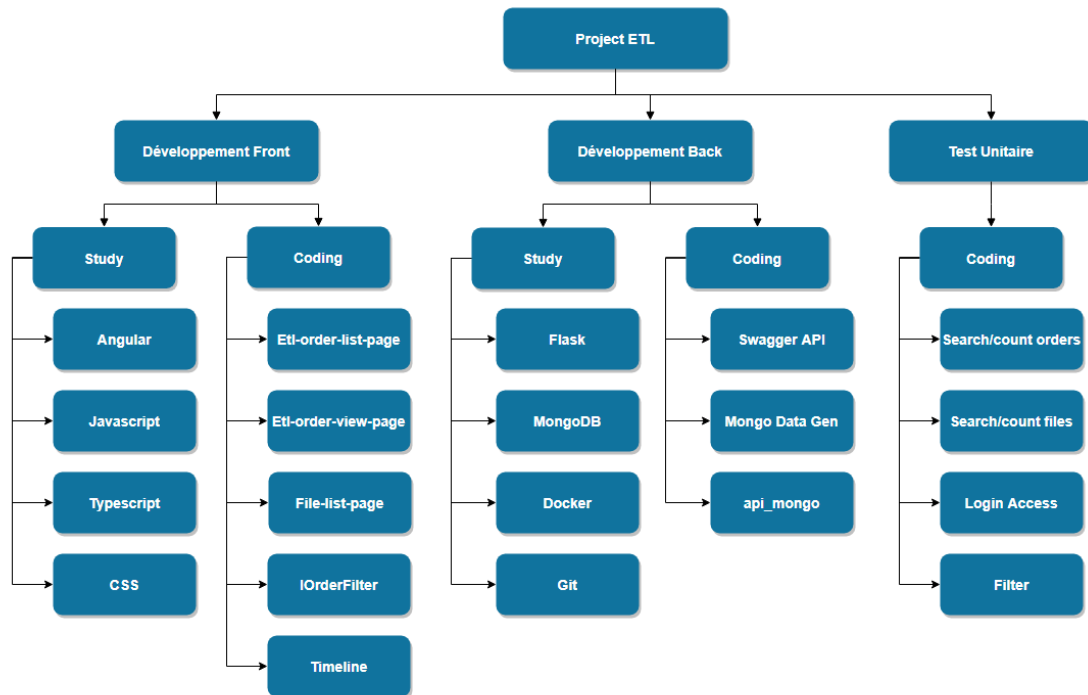
OneDrive est un service d'hébergement de fichiers qui permet aux utilisateurs de synchroniser des fichiers et d'y accéder ultérieurement à partir d'un navigateur Web ou d'un appareil mobile. Les utilisateurs peuvent partager des fichiers publiquement ou avec leurs contacts.

#### **2.2.5. Gantt**

## 3. Projet ETL

### 3.1. Organisation

Nous divisons le projet ETL en les parties suivantes pour organiser l'exécution des tâches, afin de faciliter la compréhension et la gestion de l'ensemble de notre projet.

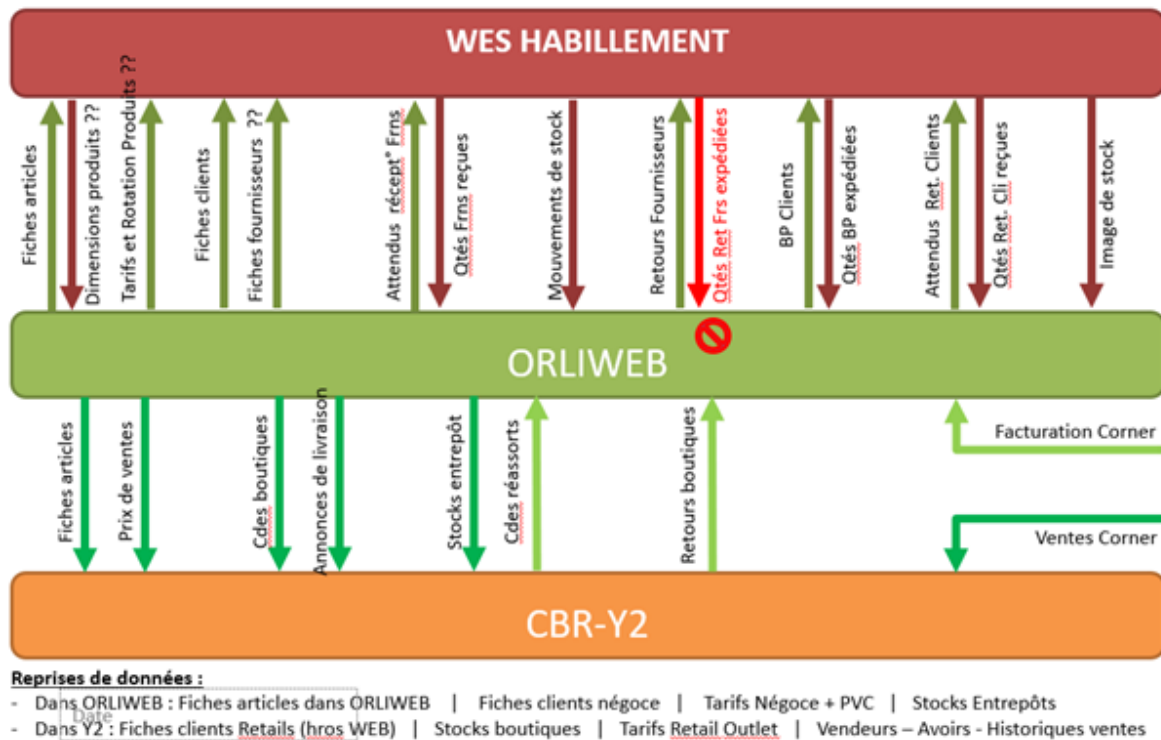


Les tâches détaillées de projet ETL

Nous divisons le développement de chaque étape en deux étapes d'apprentissage et de codage, et subdivisons les tâches spécifiques de cette étape.

## 3.2. Génération des données fictives

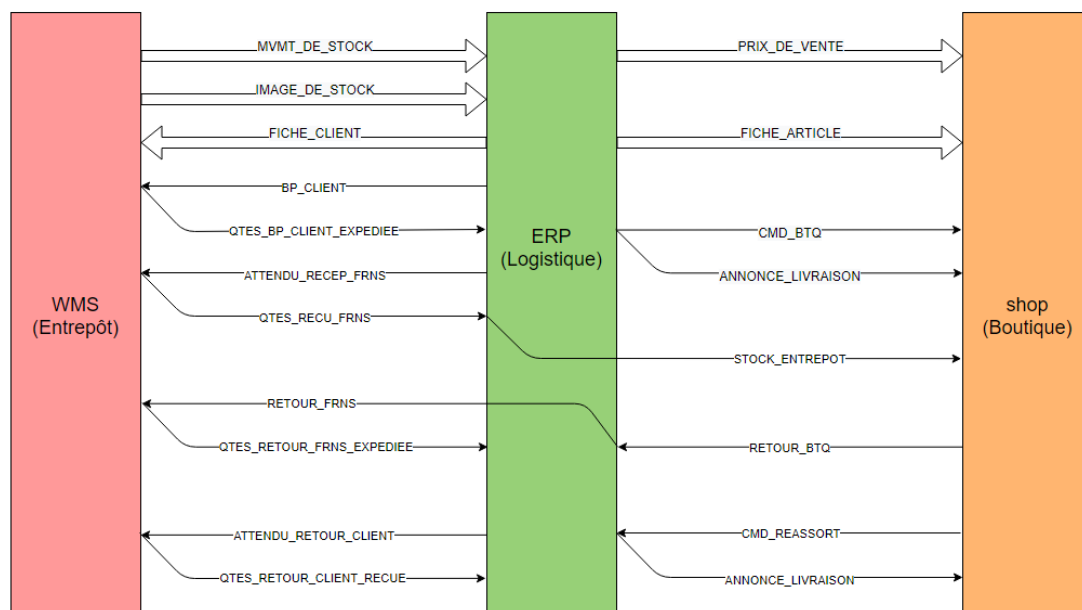
Voici un organigramme de notre projet de gestion logistique. Il s'agit notamment de magasin (CBR-Y2), de réseau logistique (ORLIWEB) et d'entrepôt (WES HABILLEMENT).



L'organigramme du projet ETL

Nous pouvons voir qu'il existe de nombreux workflows ci-dessus, ce sont les données qui seront générées dans le travail réel. Mais comme notre projet est un projet de recherche, nous utiliserons plutôt des données virtuelles.

Selon cet organigramme, nous pouvons obtenir la relation entre différents flux de transmission. Nous résumons l'organigramme et obtenons le flux de commandes suivant.



Les flux dans le système ETL

Nous pouvons diviser ces flux en quatre catégories, à savoir `Catalogue`, `Commande`, `Retours` et `Synchro`.

```
export enum FileFluxEnumCatalog {
  FICHE_ARTICLE = 'FICHE_ARTICLE',
  FICHE_CLIENT = 'FICHE_CLIENT',
  PRIX_DE_VENTE = 'PRIX_DE_VENTE'
}

export enum FileFluxEnumCommande {
  CMD_BTQ = 'CMD_BTQ',
  ANNONCE_LIVRAISON = 'ANNONCE_LIVRAISON',
  ATTENDU_RECEP_FRNS = 'ATTENDU_RECEP_FRNS',
  QTES_RECU_FRNS = 'QTES_RECU_FRNS',
  STOCK_ENTREPOT = 'STOCK_ENTREPOT',
  CMD_REASSORT = 'CMD_REASSORT'
}

export enum FileFluxEnumReturns {
  RETOUR_BTQ = 'RETOUR_BTQ',
  RETOUR_FRNS = 'RETOUR_FRNS',
  QTES_RETOUR_FRNS_EXPEDIEE = 'QTES_RETOUR_FRNS_EXPEDIEE',
  BP_CLIENT = 'BP_CLIENT',
  QTES_BP_CLIENT_EXPEDIEE = 'QTES_BP_CLIENT_EXPEDIEE',
  ATTENDU_RETOUR_CLIENT = 'ATTENDU_RETOUR_CLIENT',
  QTES_RETOUR_CLIENT_ATTENDU = 'QTES_RETOUR_CLIENT_ATTENDU'
}

export enum FileFluxEnumSynchro {
  MMT_DE_STOCK = 'MMT_DE_STOCK',
  IMAGE_DE_STOCK = 'IMAGE_DE_STOCK'
}
```

### Les quatre catégories de flux dans l'Angular

Dans cette partie, nous avons terminé l'opération de génération de données de simulation. Nous avons spécifiquement utilisé Python comme langage pour générer des scripts et nous nous sommes connectés à la base de données MongoDB pour implémenter l'insertion de documents. De plus, nous utilisons également JSON comme type de données pour réaliser une communication sans état entre les API.

Key	Value
▼ (1) ObjectId("5eec7c883a9723ff993d7c4c")	{ 16 fields }
_id	ObjectId("5eec7c883a9723ff993d7c4c")
status	success
protocol	Retail
reference	1126fad1-e090-ff4c-63e5-cc09ab4a15c
is_last_update	true
file_type	CMD_REASSORT
file_name	shop_ERP_CMD_REASSORT_2019-07-02_2.csv
created_at	2019-07-02T02:52:26
destination	ERP
updated_at	2019-07-04T15:54:53
source	shop
file_line	14
w_id	ce70cb96-8fc4-963e-9406-255d7a2a4ea
> error	{ 3 fields }
protocol_version	1.0
> json_content	{ 14 fields }

### Un exemple d'une commande générée dans MongoDB

### 3.3. Modifications des APIs Python

Le serveur principal de notre projet utilise le framework Flask et Swagger pour construire ce service. Notre ancien projet est un système typique de gestion de l'information logistique des télécommunications, donc notre travail consiste à modifier l'ancien projet pour l'adapter à notre nouveau type de données et système de type de service.

URI	Method	Result
/file	GET	Retrieve file list
	POST	Create/upload a file
/order	GET	Retrieve order list
/orders/{reference}/{opérateur}	GET	Retrieve order view page for a given reference and opérateur
/metrics/orders/{protocol}	GET	Retrieve all orders for a given protocol
/flux	GET	Retrieve order with the label of protocol

#### L'API du système d'origine

Nous avons défini un nouveau type de données pour notre projet et défini une nouvelle API basée sur l'API d'origine. Il comprend les opérations suivantes.

URI	Method	Result
/etl_file	GET	Retrieve etl file list
	POST	Create/upload etl file property
/etl_order	GET	Retrieve etl order list
/etl_order/{item_reference}	GET	Retrieve etl order view page with a given item_reference
/etl_history/{item_reference}	GET	Retrieve etl order history usage with a given item_reference

#### La conception d'API pour ETL

Sur la base du routeur défini ci-dessus, nous pourrions construire le fichier `swagger.yaml` pour définir la documentation de l'API. Ensuite, nous utilisons swagger-codegen officiellement fourni par swagger pour générer automatiquement des classes d'API et des contrôleurs. Nous pouvons donc voir l'API suivante sur la page Web.

#### **etl\_file**

GET	/etl_file
POST	/etl_file

#### **etl\_order\_list**

GET	/etl_order
GET	/etl_order/{item_reference}
GET	/order_history/{item_reference}



#### Swagger UI pour visualiser l'API

### 3.4. Angular pour l'Interface Homme-Machine

Nous avons utilisé Angular comme framework frontal. Angular est un framework côté client open source de Google, basé sur TypeScript.

Nous utilisons Angular pour faire beaucoup de conception fonctionnelle. Par exemple, nous pouvons ajouter une zone de recherche pour filtrer les résultats EAN, ou nous pouvons ajouter des informations supplémentaires sur le timeline.

Ici en ci-dessous nous présente les filtres qu'on a créés.

Q Filtres : Référence de la commande Référence de la boutique Code-barres EAN Date de la commande  Date d'intégration 

Nom du fichier Source Destination Famille de Flux  Version de protocole 

Les filtres pour filtrer les commandes

La page détail pour une commande particulière.

#### Détails de la commande

07/03/2019 23:23:42

Nom du fichier	Source
ERP_WMS_ATTENDU_RECEP_FRNS_19960207_6.csv	ERP
Destination	Date de dernière mise à jour
WMS	29/05/2020 18:28:15



Le timeline est affiché dans la page détail

De plus, nous avons également marqué des commandes érronnée.



19/10/2018 00:32:31



FICHE\_ARTICLE *ERR\_STG*

Erreur de stockage du ERP

› Voir la totalité des données

Une commande erronée

## 3.5. Test Unitaire

Les tests unitaires sont utilisés pour détecter si le projet peut fonctionner normalement. C'est une pratique très importante pour le développement TDD (développement piloté par les tests).

Dans notre projet, nous testons l'API et le front-end. Parce que notre backend utilise le langage Python, nous avons choisi la bibliothèque unittest, une bibliothèque très couramment utilisée pour écrire des tests unitaires en Python, pour nous aider à implémenter le processus de test unitaire.

```
53
54 # test count_files without param
55 def test_count_files_none(self):
56     results = None
57     try:
58         param_dict = {}
59         logger.info('test count_files without param')
60
61         results = self.count_documents(
62             db=self.db,
63             collection=self.file_collection,
64             param_dict=param_dict
65         )
66     except Exception as e:
67         logger.debug('Exception: %s', e)
68     # we have 3 file documents for test in total
69     self.assertEqual(results, 3)
70
71 # test count_files with param
72 def test_count_files_param(self):
73     results = None
74     try:
75         param_dict = {
76             'item_reference': 'IAU4002',
77             'file_name': 'shop_ERP_RETOUT_BTQ_2018-12-20_5.csv'
78         }
79         logger.info('test count_files with param: %s', param_dict)
80
81         results = self.count_documents(
82             db=self.db,
83             collection=self.file_collection,
84             param_dict=param_dict
85         )
86     except Exception as e:
87         logger.debug('Exception: %s', e)
88
89     # we have only one result that satisfies such param_dict
90     self.assertEqual(results, 1)
91
92 # test search_files without param
93 def test_search_files_none(self): ...
94
95 # test search_files with param
96 def test_search_files_param(self): ...
97
98 # test count_orders without param
99 def test_count_orders_none(self): ...
100
101 # test count_orders with param
102 def test_count_orders_param(self): ...
103
104 # test search_orders without param
105 def test_search_orders_none(self): ...
106
107 # test search_orders with param
108 def test_search_orders_param(self): ...
```

### Tests unitaires pour count\_files

Ce qui suit est le résultat de notre test unitaire, ce qui signifie que notre projet a réussi le test unitaire.

```
2020-07-09 16:40:15,112 - INFO - __main__ - test count_files without param
2020-07-09 16:40:15,112 - DEBUG - __main__ - count_documents [db=igw, collection=test_etl_demo_file, param_dict={}]
2020-07-09 16:40:15,156 - DEBUG - __main__ - result:3
2020-07-09 16:40:15,158 - INFO - __main__ - test count_files with param: {'item_reference': 'IAU4002', 'file_name': 'shop_ERP_RETOUT_BTQ_2018-12-20_5.csv'}
2020-07-09 16:40:15,158 - DEBUG - __main__ - count_documents [db=igw, collection=test_etl_demo_file, param_dict={'item_reference': 'IAU4002', 'file_name': 'shop_ERP_RETOUT_BTQ_2018-12-20_5.csv'}]
2020-07-09 16:40:15,161 - DEBUG - __main__ - result:1
2020-07-09 16:40:15,162 - INFO - __main__ - test count_orders without param
2020-07-09 16:40:15,162 - DEBUG - __main__ - count_documents [db=igw, collection=test_etl_demo_order, param_dict={}]
2020-07-09 16:40:15,165 - DEBUG - __main__ - result:2
2020-07-09 16:40:15,166 - INFO - __main__ - test count_orders with param: {'error_step_status': 'failed', 'json_content.shop_reference': 'SHOP_457'}
2020-07-09 16:40:15,166 - DEBUG - __main__ - count_documents [db=igw, collection=test_etl_demo_order, param_dict={'error_step_status': 'failed', 'json_content.shop_reference': 'SHOP_457'}]
2020-07-09 16:40:15,169 - DEBUG - __main__ - result:1
2020-07-09 16:40:15,170 - INFO - __main__ - test search_files without param
2020-07-09 16:40:15,171 - DEBUG - __main__ - get_documents [db=igw, collection=test_etl_demo_file, param_dict={}, limit=0, offset=0, sorts_list=None, projection=None]
2020-07-09 16:40:15,175 - INFO - __main__ - test search_files with param: {'item_reference': 'IAU4002', 'file_name': 'shop_ERP_RETOUT_BTQ_2018-12-20_5.csv'}
2020-07-09 16:40:15,175 - DEBUG - __main__ - get_documents [db=igw, collection=test_etl_demo_file, param_dict={'item_reference': 'IAU4002', 'file_name': 'shop_ERP_RETOUT_BTQ_2018-12-20_5.csv'}, limit=0, offset=0, sorts_list=None, projection=None]
2020-07-09 16:40:15,179 - INFO - __main__ - test search_orders without param
2020-07-09 16:40:15,179 - DEBUG - __main__ - get_documents [db=igw, collection=test_etl_demo_order, param_dict={}, limit=0, offset=0, sorts_list=None, projection=None]
2020-07-09 16:40:15,185 - INFO - __main__ - test search_orders with param: {'updated_at': {'$gt': '2019-01-01T22:22:22'}, 'json_content.item_reference': {'$regex': 'IAK'}}
2020-07-09 16:40:15,185 - DEBUG - __main__ - get_documents [db=igw, collection=test_etl_demo_order, param_dict={'updated_at': {'$gt': '2019-01-01T22:22:22'}, 'json_content.item_reference': {'$regex': 'IAK'}}, limit=0, offset=0, sorts_list=None, projection=None]
.....
Ran 8 tests in 0.078s

OK
```

### L'affichage dans la console des tests unitaires

## 3.6. Déploiement en Docker

Après avoir terminé la conception front-end et back-end, notre projet peut s'exécuter normalement sur le serveur. Mais si nous voulons déployer le projet afin que le projet puisse toujours fournir des services et être plus facilement accessible, nous devons utiliser Docker.

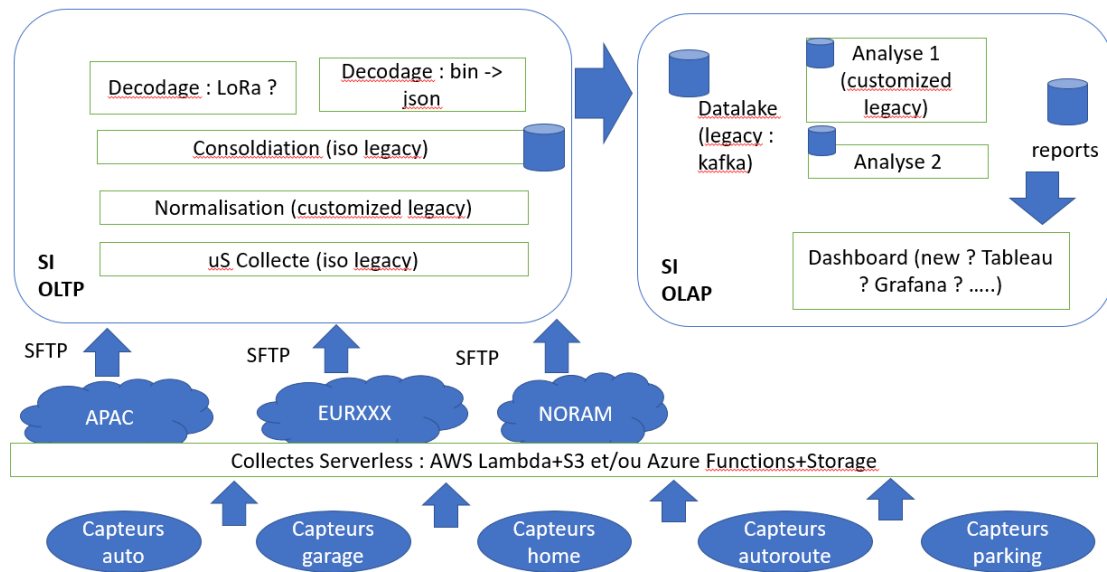
```
start@dev-zpa:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
259daac7f592	etl-tracking-front:develop	"nginx -g 'daemon of..."	7 days ago	Up 7 days	80/tcp, 0.0.0.0:8010->8000/tcp	etl-tracking-front
7a4a78156978	etl-tracking-web-api:develop	"bash /spikeelabs/run..."	7 days ago	Up 7 days	0.0.0.0:8011->8000/tcp	etl-tracking-web-api
f3d86b709cd5	mongo:latest	"docker-entrypoint.s..."	3 months ago	Up 3 months	0.0.0.0:27017->27017/tcp	keen_ganguly

Les services sont stockées dans des conteneurs docker

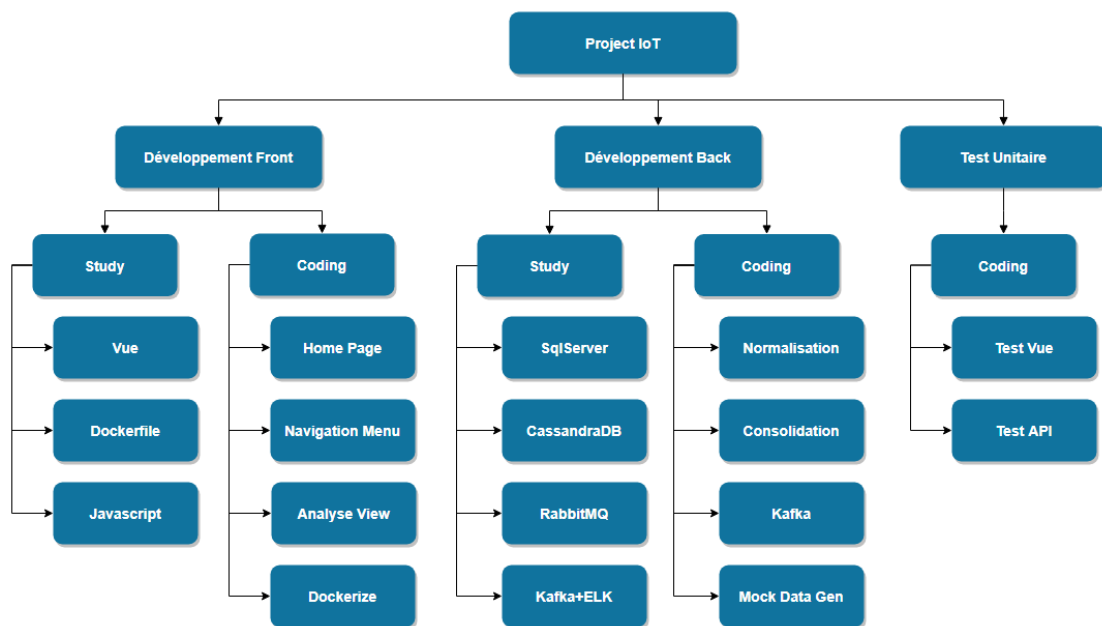
## 4. Projet IoT

### 4.1. Organisation



Le processus global du projet

Tout comme ce que nous avons fait dans le projet ETL, nous pouvons diviser le projet IoT en quelques petites tâches.



Les tâches détaillées de projet IOT

## 4.2. Normalisation

À l'étape Normalisation, ce que nous devons faire est de traiter les données brutes collectées, afin que nous puissions obtenir les données de trois formats de fichiers différents (JSON, CSV et XML), et les normaliser pour obtenir un format unifié. Les données sont transférées à l'étape suivante.

Properties Data Diagram

Cassandra CQL - system

ed:json Enter a SQL expression to filter results (use Ctrl+Space)

	arrivaltime	departuretime	area	betweenstreet1	betweenstreet2	in	sign	streetmarker			
1	01/31/2017 06:30:00 PM	01/31/2017 06:48:31 PM	Docklands	BATMANS HILL DRIVE	VILLAGE STREET	1,111	False	4	1,457	13271S	
2	05/26/2017 12:00:00 AM	05/26/2017 12:01:10 AM	Docklands	FISHPLATE LANE	COLLINS STREET	70	False	2	79	13466E	
3	01/16/2017 01:34:40 PM	01/16/2017 01:43:35 PM	Docklands	DOCKLANDS DRIVE	CARAVEL LANE	535	False	2	1/4P M-F 7:30-18:30	1,227	13912E
4	12/19/2017 08:32:26 AM	12/19/2017 08:33:38 AM	Jolimont	GISBORNE STREET	MORRISON PLACE	72	False	4	16	12263S	
5	08/17/2017 06:39:46 AM	08/17/2017 06:48:34 AM	East Melbourne	GREY STREET	GIPPS STREET	528	False	5	511	12049W	
6	12/12/2017 09:14:27 AM	12/12/2017 09:20:46 AM	East Melbourne	GREY STREET	GIPPS STREET	379	False	5	2P MTR M-F 7:30-18:30	511	12049W
7	02/03/2017 10:07:58 PM	02/03/2017 10:10:11 PM	Titbits	LITTLE LONSDALE STREET	LONSDALE STREET	133	False	1	1,171	C1210	
8	03/01/2017 11:17:41 AM	03/01/2017 11:23:38 AM	Southbank	STURT STREET	DODDS STREET	357	False	4	1P TKT A M-F 7:30-18:30	547	9437S
9	04/18/2017 12:53:33 PM	04/18/2017 01:08:45 PM	East Melbourne	GIPPS STREET	HOTHAM STREET	912	False	5	2P MTR M-F 7:30-18:30	511	12001W
10	06/02/2017 07:30:00 AM	06/02/2017 06:30:00 PM	West Melbourne	VICTORIA STREET	RODEN STREET	39,600	True	5	1P A RPE M-F 7:30-18:30	839	7327W
11	06/04/2017 03:26:41 PM	06/04/2017 06:00:39 PM	Family	BOURKE STREET	LITTLE COLLINS STREET	9,238	True	5	2P SUN 7:30-18:30	200	1381W
12	09/06/2017 06:20:52 PM	09/06/2017 06:24:39 PM	Family	BOURKE STREET	LITTLE COLLINS STREET	227	False	5	1P MTR M-SAT 7:30-18:30	200	1381W
13	12/16/2017 07:32:46 AM	12/16/2017 07:34:09 AM	West Melbourne	BATMAN STREET	JEFFCOTT STREET	83	False	5	1/2P MTR M-SAT 7:30-19:30	1,285	1607W
14	08/15/2017 02:14:18 PM	08/15/2017 02:32:50 PM	Mint	WILLIAM STREET	QUEEN STREET	1,112	False	3	1/2P TKT A M-SAT 7:30-19:30	5	6006N
15	10/01/2017 10:06:09 PM	10/01/2017 10:15:48 PM	Mint	WILLIAM STREET	QUEEN STREET	579	False	3		5	6006N
16	08/03/2017 12:00:00 AM	08/03/2017 07:30:00 AM	Docklands	COLLINS STREET	BOURKE STREET	27,000	False	5		753	13303W
17	12/21/2017 12:53:27 AM	12/21/2017 01:02:04 AM	Docklands	COLLINS STREET	BOURKE STREET	517	False	5		753	13303W
18	06/13/2017 12:55:28 PM	06/13/2017 01:18:45 PM	Jolimont	JOLIMONT ROAD	CLARENDON STREET	1,397	False	3	2P TKT A M-F 7:30-18:30	1,413	12540N
19	01/25/2017 04:17:24 PM	01/25/2017 04:19:50 PM	Tavistock	WILLIAM STREET	QUEEN STREET	146	False	3	LZ 15M M-F 7:30-19:30	669	1806N
20	06/14/2017 01:29:27 PM	06/14/2017 01:31:03 PM	Docklands	CAPTAIN WALK	ENTERPRIZE WAY	96	False	3	LZ 15M M-F 7:30-18:30	123	13232N
21	02/21/2017 06:37:05 PM	02/21/2017 06:41:56 PM	Docklands	COLLINS STREET	BOURKE STREET	291	False	5		998	13101W
22	06/15/2017 06:38:26 PM	06/15/2017 06:47:06 PM	Docklands	HARBOUR ESPLANADE	WURUNDJERI WAY	520	False	3	1P M-SUN 7:30-23:00	123	13210N
23	01/04/2017 04:02:18 PM	01/04/2017 04:21:38 PM	Hyatt	COLLINS STREET	FLINDERS LANE	1,160	False	2	1P MTR M-SAT 7:30-18:30	647	342E
24	10/24/2017 01:51:44 PM	10/24/2017 01:53:55 PM	Jolimont	GISBORNE STREET	LANSLOWNE STREET	131	False	4	1P TKT A M-SAT 7:30-18:30	178	12383S
25	09/19/2017 09:57:30 AM	09/19/2017 10:39:59 AM	Drummond	LYGON STREET	DRUMMOND STREET	2,549	False	1	2P TKT A M-F 7:30-18:30	1,102	C9060
26	03/11/2017 07:39:10 PM	03/11/2017 07:43:08 PM	Magistrates	LITTLE LONSDALE STREET	LONSDALE STREET	238	False	5	2P MTR SAT 7:30-20:30	1,285	1571W
27	12/12/2017 07:30:00 AM	12/12/2017 07:51:00 AM	Hardware	LONSDALE STREET	LITTLE BOURKE STREET	1,260	False	1	1P MTR M-SAT 7:30-19:30	1,171	C1176
28	03/30/2017 05:41:33 PM	03/30/2017 06:30:00 PM	RACV	WILLIAM STREET	QUEEN STREET	2,907	False	3	1P MTR M-SAT 7:30-18:30	911	2232N
29	07/22/2017 12:30:00 PM	07/22/2017 02:10:30 PM	Chuanzhang	CHETWIND STREET	HOWARD STREET	10,170	False	1		1,314	C7190

Les données normalisées sont stockées dans Cassandra

## 4.3. Consolidation

mapper tous les données normalisées avec le serveur sql qui s'occupe des utilisateurs

edr

edr

normalized\_edrs

normalization\_runs

consolidated\_edrs

Properties

Data

Diagram

consolidated\_edrs

Enter a SQL expression to filter results (use Ctrl+Space)

	run_id	billing_period	subsc_id	lot_id	unique_id	attr		logo_id
						key	value	
1	20200727174748767_140080187544672	09/2016	23	data_edr_csv.csv_20	uid_17864_2017062116414200	acc0_id	35	35
2	20200727174748701_140080187544672	08/2015	19	data_edr_csv.csv_18	uid_20506_2017101907300000	acc0_id	27	27

Les données consolidées sont stockées dans Cassandra

## 4.4. Kafka

écouteur pour capturer les données

The screenshot displays the Landoop interface for managing Kafka topics. On the left, a list of 10 topics is shown, including 'backblaze\_smart', 'coyote-test-avro', 'coyote-test-binary', 'coyote-test-json', 'logs\_broker', 'nyc\_yellow\_taxi\_trip\_data', 'telecom\_italia\_data', 'sea\_vessel\_position\_reports', 'telecom\_italia\_grid', and 'applog'. The 'applog' topic is selected, showing 3 partitions and 1 replication. On the right, the 'applog' topic details are displayed, including the 'PARTITIONS' tab with 3 partitions and the 'RAW DATA' tab showing a list of log messages. The messages are in JSON format, containing fields like 'timestamp', 'level', 'component\_name', 'inst\_id', 'name', 'message', 'partition', and 'offset'.

**10 TOPICS** System Topics

Search topics:

- backblaze\_smart  
2 Partitions • 1 Replication | 25 configs
- coyote-test-avro  
1 Partitions • 1 Replication | 25 configs
- coyote-test-binary  
1 Partitions • 1 Replication | 25 configs
- coyote-test-json  
1 Partitions • 1 Replication | 25 configs
- logs\_broker  
1 Partitions • 1 Replication | 25 configs
- nyc\_yellow\_taxi\_trip\_data  
5 Partitions • 1 Replication | 25 configs
- telecom\_italia\_data  
4 Partitions • 1 Replication | 25 configs
- sea\_vessel\_position\_reports  
3 Partitions • 1 Replication | 25 configs
- telecom\_italia\_grid  
1 Partitions • 1 Replication | 25 configs
- applog**  
3 Partitions • 1 Replication | 25 configs

Powered by **Landoop**

**applog**

**DATA** **PARTITIONS 3** **CONFIGURATION 25**

Total Messages Fetched: 188. Data type: binary

filter All partitions

TOPIC TABLE RAW DATA

```
1- [
2- {
3-   "topic": "applog",
4-   "key": "Be",
5-   "value": "{\"@timestamp\": \"2020-07-31T14:00:43.797060Z\", \"levelName\": \"DEBUG\", \"component_name\": \"\\\"consolidation\\\"\", \"inst_id\": \"0\", \"name\": \"\\\"spikeealabs.common.base.rootdir\\\"\", \"message\": \"\\\"prefix_with_rootdir relative_path: //config/config.json\\\"\", \"partition\": 0, \"offset\": 0}",
6-   "partition": 0,
7-   "offset": 0
8- },
9- {
10-   "topic": "applog",
11-   "key": "Be",
12-   "value": "{\"@timestamp\": \"2020-07-31T14:00:43.798254Z\", \"levelName\": \"DEBUG\", \"component_name\": \"\\\"consolidation\\\"\", \"inst_id\": \"0\", \"name\": \"\\\"spikeealabs.common.base.rootdir\\\"\", \"message\": \"\\\"check path /home/start/config/config.json\\\"\", \"partition\": 0, \"offset\": 1}",
13-   "partition": 0,
14-   "offset": 1
15- },
16- {
17-   "topic": "applog",
18-   "key": "Be",
19-   "value": "{\"@timestamp\": \"2020-07-31T14:00:43.812342Z\", \"levelName\": \"ERROR\", \"component_name\": \"\\\"consolidation\\\"\", \"inst_id\": \"0\", \"name\": \"\\\"spikeealabs.common.base.appconf\\\"\", \"message\": \"\\\"Error when trying to load config file\\\"\", \"partition\": 0, \"offset\": 2}",
20-   "partition": 0,
21-   "offset": 2
22- },
23- {
24-   "topic": "applog",
25-   "key": "Be",
26-   "value": "{\"@timestamp\": \"2020-07-31T14:00:43.812593Z\", \"levelName\": \"DEBUG\", \"component_name\": \"\\\"consolidation\\\"\", \"inst_id\": \"0\", \"name\": \"\\\"_main_\\\"\", \"message\": \"\\\"Fatal error\\\"\", \"partition\": 0, \"offset\": 3}",
27-   "partition": 0,
28-   "offset": 3
29- }
30- ]
```

## 4.5. Dashboard

afficher un API pour faciliter les opérations

request handled correctly with status code 200

Home

Analyse

Community Detection

Data Exploration

Fraud Detection

Setting

Data Persist (post)

Type

persist

Time period

02/08/2020

-

29/08/2020

Persist

Reset

Communities Detection (get)

Type

please choose a type

Time period

Start date

-

End date

Get Data

Rest

L'interface Homme-Machine de projet IoT pour faire des demandes vers serveur



## 4.6. Déploiement en Docker

Déployer le projet en docker

## 5. Conclusion

---

Grâce à ce projet, j'ai maîtrisé de nombreuses technologies et outils de développement front-end et back-end avec des scénarios d'application pratiques, tels que Git et Docker. Et ce projet de recherche a été fait par moi-même, et cela m'a aussi permis d'apprendre à gérer le projet et à être responsable de mon propre projet.

Tous ces éléments ont apporté un soutien conceptuel plus riche à la gestion de mon projet de fin d'études mené en 5ème année d'Électronique et Technologies Numériques.