# Introduction

Twelec est une application web réalisée dans le cadre du projet informatique du 4ème semestre du cursus ingénieur à l’École Nationale Supérieure de Cognitique de Bordeaux.

2017 est une année majeure dans la vie des français puisqu’ils se préparent à élire le(la) 9ème président(e) de la Vème république française. Plusieurs outils sont mis à disposition des citoyens français pour connaître la tendance de leurs compatriotes comme par exemple les sondages. Seulement à l’heure actuelle aucuns outils ne permet de connaître la réputation de chacun des candidats sur internet.



logo de l’élection présidentielle française

Il peut donc paraître utile de mettre à disposition un nouvel qui analyse seulement les réseaux sociaux et en ressort une tendance sur le pays, par région, par département, par ville, en mesurant également une évolution temporelle.

Les données doivent donc être géo-localisées et horodatées. Pour afficher ces informations, la plupart des applications similaires utilisent des cartes interactives qui sont très intuitives et facile d’utilisation.

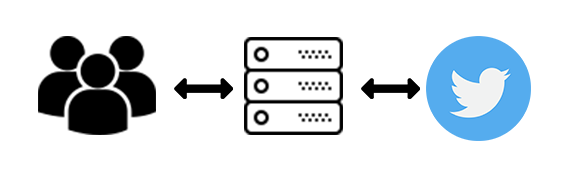
Le but de Twelec est donc de montrer la réputation des hommes politiques sur les réseaux sociaux dans le cadre des élections présidentielles française de 2017. Il a été réalisé par maxime Da Silva en collaboration avec l’entreprise Quorum spécialisée dans l’analyse de données pour la politique. Twelec a donc pour vocation d’afficher une carte interactive de la notoriété des 11 hommes et femmes politiques « qualifié » pour le premier tour de la présidentielle française.

# Objectifs du projet

Le but de Twelec est donc de mieux comprendre l’ « e-réputation » des hommes politiques à travers une analyse des réseaux sociaux. Une carte interactive devra être présentée avec des statistiques pour la mettre en valeur. Plusieurs niveaux d’accès doivent être présents (la région, le département et la ville), et l’utilisateur doit pouvoir naviguer simplement entre les différents niveaux d’accès.

L’application doit être en mesure de récupérer les données sur les réseaux sociaux en temps réel et les ajouter à une base de données interne.

Un serveur doit ensuite mettre à disposition ces informations pour qu’une application externe puisse y accéder. Il également être réalisé une application côté client qui récupère les données du serveur et les affiche sous la forme de carte interactive et statistiques.



L’application doit pouvoir afficher une comparaison entre les 11 candidats, mais aussi une carte de chaleur de la répartition des mentions de chaque candidat.

# Gestion de projet

## Organisation générale

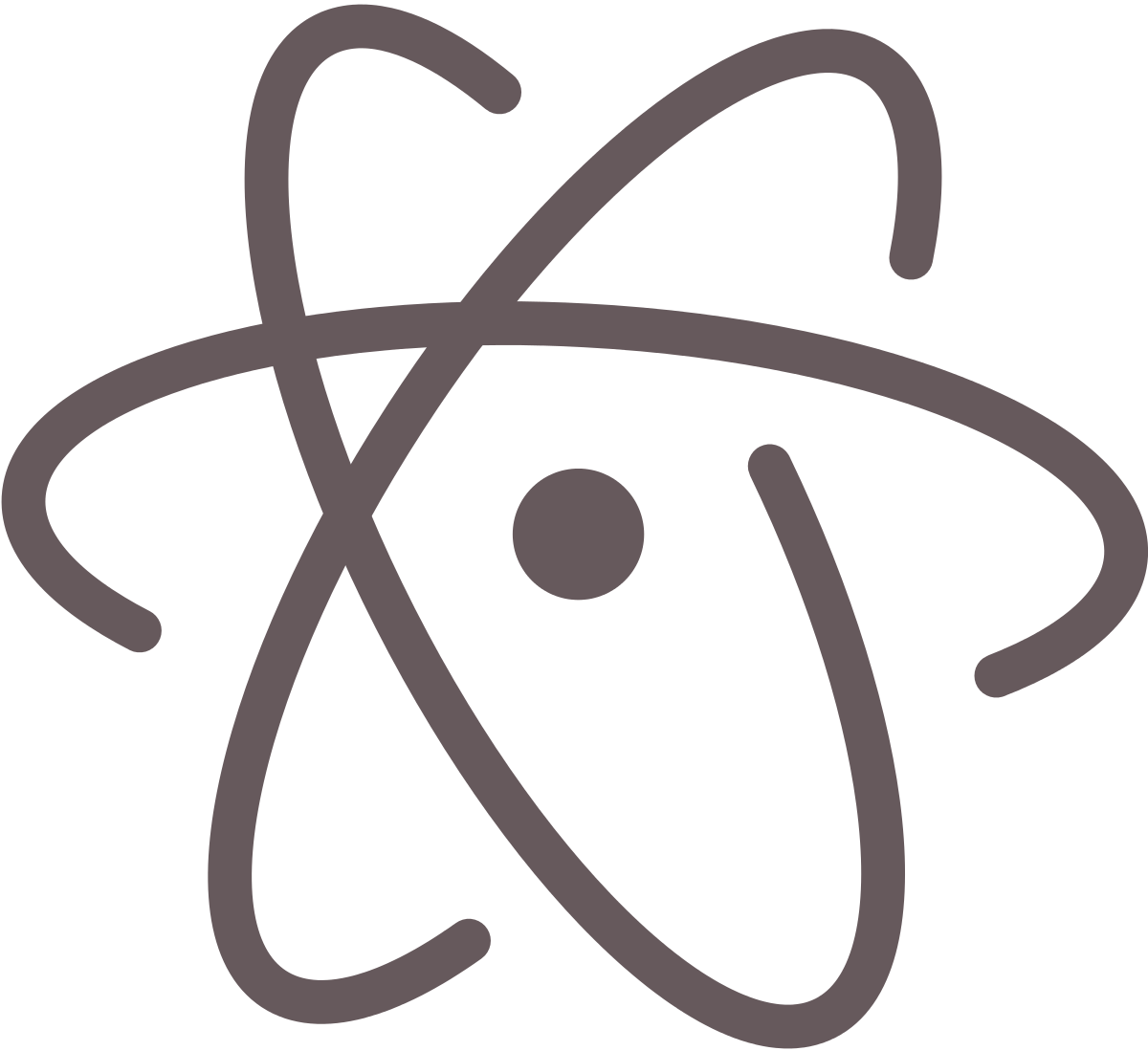
Ce projet a été démarré le 23 janvier 2017, j’ai commencé par ce qui me paraissait l’essentiel dans mon application, c’est-à-dire la récupération des données sur les réseaux sociaux.

Cette opération a été beaucoup plus longue que le temps que je lui avais consacré dans mon gantt rendu au début du projet puisque je ne pensais que la récupération serait aussi laborieuse. (cf. la partie Présentation du projet).

J’ai donc commencé la seconde partie, qui avait pour but de mettre en place le serveur qui présenterait les données aux clients, en retard. Elle m’a également posé beaucoup de soucis. C’est pourquoi tous les éléments que je voulais implanter dans mon projet ne sont pas forcément présents.

## Matériel et outil utilisé

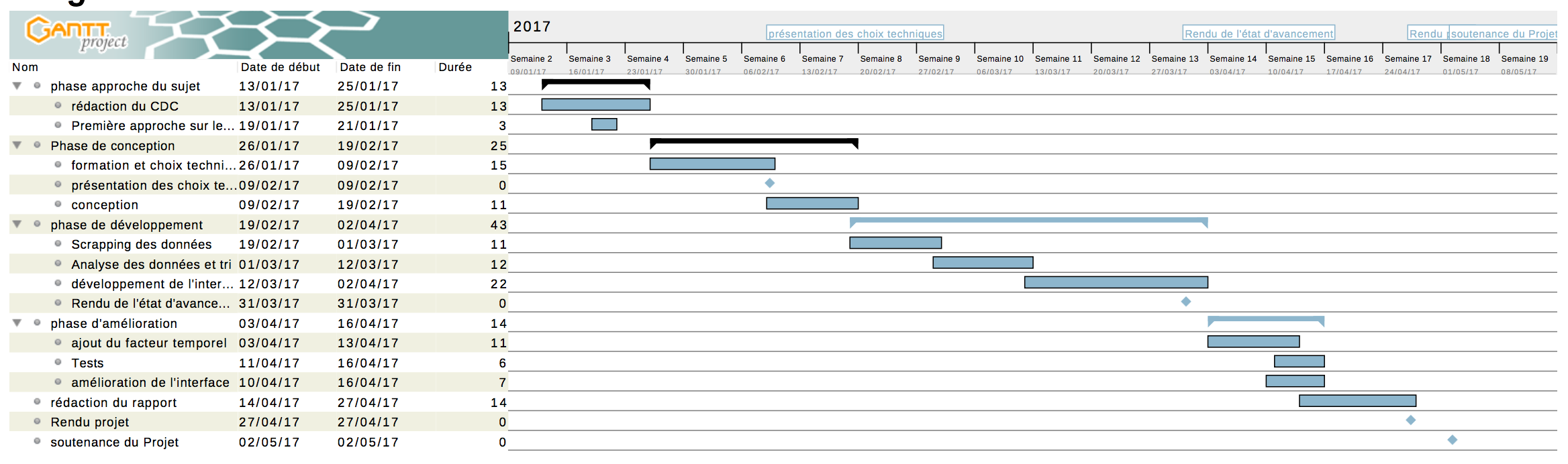
La plupart de mon code a été écrit en javascript, et en python. J’ai donc utilisé l’IDE « Atom » qui permet de gérer ces deux langages sous peine d’installer les packages adéquats.



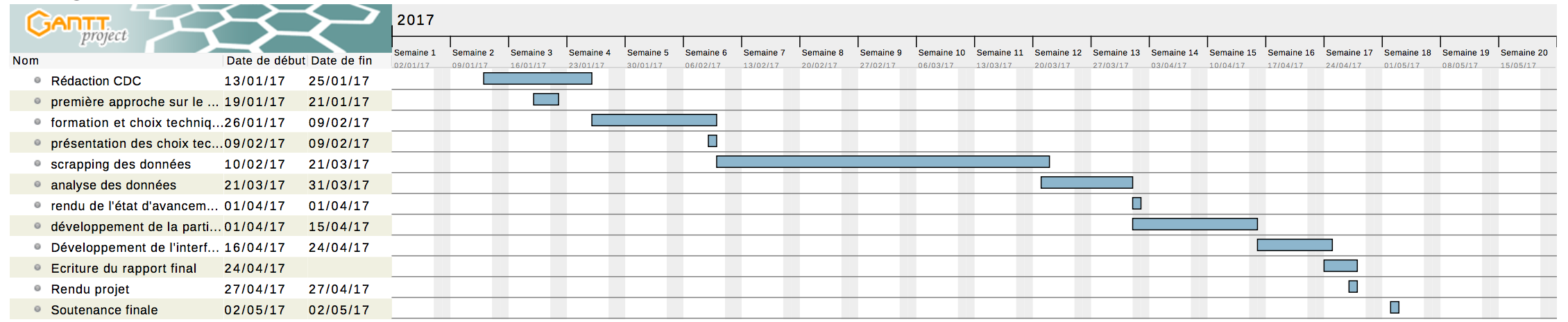
J’ai également utilisé plusieurs outils pour la production et la gestion des versions de mon code comme Git pour la gestion de version sur la plateforme Git (le projet est d’ailleurs accessible ici: <https://github.com/maximedasilva/SocialNetwork_Reputation> ). J’ai aussi utilisé la plateforme Heroku pour la mise en production de mon serveur.

## Planning

Voici le planning que j’avais prévu au début du projet :



Et voici le planning qui a été mien durant ce projet :



# Présentation du projet

## Présentation technique

Twelec est une application basée sur Node.JS qui permet de créer des applications en JavaScript. Le serveur est donc écrit en node.JS le Scraper permettant de récupérer les informations en temps réel sur les réseaux est quant à lui écrit en Python.



La partie client est écrite en HtML/CSS /Javascript. C’est donc une simple page qui nécessite seulement une connexion à internet pour être lancée.



Le scraper utilise « Tweepy » qui permet d’accéder à l’API twitter vgrâce à Python. Tweepy me permet donc de me connecter à Twitter. Mon module Tweepy est configuré en mode streaming pour récupérer les tweets en temps réel. Il est également configuré pour ne chercher que sur le territoire français.



Cette option à deux effets :

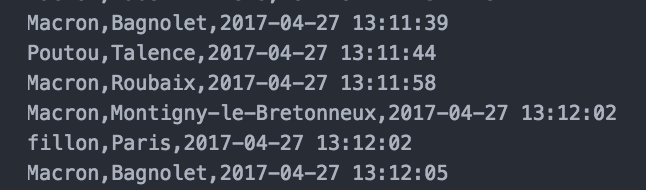
* Évidemment les tweets provenant d’autres pays sont ignorés
* Mais les tweets qui ne possèdent pas de géolocalisation sont ignorés également, puisque pour que les tweets soient geo-localisés il faut activer une option sur son compte twitter.

Une fois qu’un tweet est récupéré, le script python écrit sur un fichier CSV les 3 informations qui me sont utiles : le candidat mentionné (il y aura autant de ligne que de candidats mentionnés dans un tweet), la ville d’où provient le tweet et l’horodatage du tweet.



(Ne voyez pas d’affirmation politique dans ce tweet, c’est un exemple)

Ce tweet une fois analysé et récupéré par notre module Python donne dans notre fichier CSV :



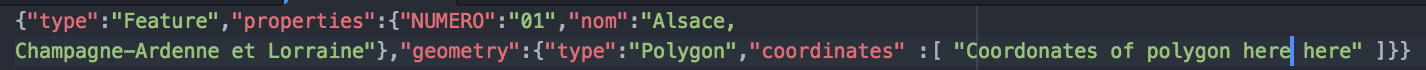
Il y a un décalage de 2heures entre le tweet et le résultat sur le fichier CSV car l’API se base sur l’heure GMT.

Une fois toutes ces données collectées on veut maintenant qu’elles soient accessibles aux clients de notre application.

Pour créer la carte interactive utiles à l’affichage des données j’ai utilisé une carte faite par mapBox couplé à l’API de Leaflet pour l’affichage des données.



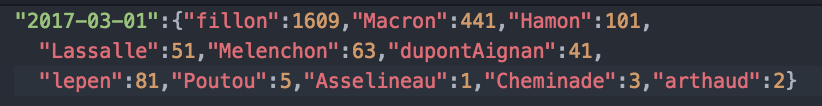
Leaflet utilise le format GeoJSON pour afficher ses données. Le format GeoJSON est un format courant pour l’affichage de données géolocalisées. Il se présente sous cette forme :



Dans l’exemple juste au-dessus, on peut voir dans la partie properties tous les éléments qui ne sont pas utiles à la consctruction du polygone mais qui sont utiles pour l’interprétation de celui-ci. Par exemple on sait que ce polygone est la région numéro 01 de France, Alsace Champagne-Ardenne et Lorraine, aujourd’hui appelée « Grand Est ».

Il suffit donc de rajouter les informations pertinentes dans cette partie properties, comme par exemple le nombre de tweets par candidat pour la région ici. Dans Twelec, toutes les régions départements et villes possèdent un fichier GeoJSON semblable à celui-ci.

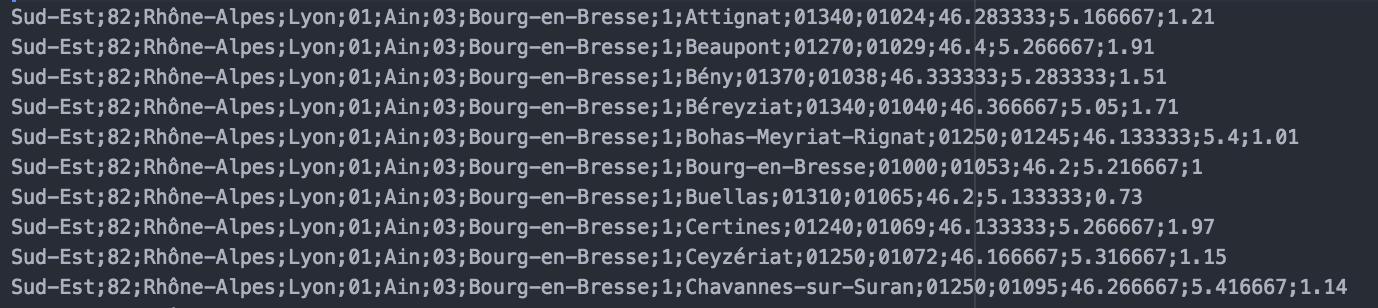
On veut que le client ait accès à une évolution temporelle des tweets pour chaque candidat. On remplit donc un fichier JSON avec pour chaque jour, le nombre de tweet pour chaque candidat :



Il faut donc remplir ces fichiers GeoJSON avec les informations présentes sur le fichier CSV. Pour cela le serveur prend régulièrement tous les tweets présents dans le fichier CSV et qui n’ont pas été ajoutés aux fichiers GeoJSON, il compare le nom de la ville enregistré au fichier contenant toutes les villes françaises (fourni par Data.gouv.fr).

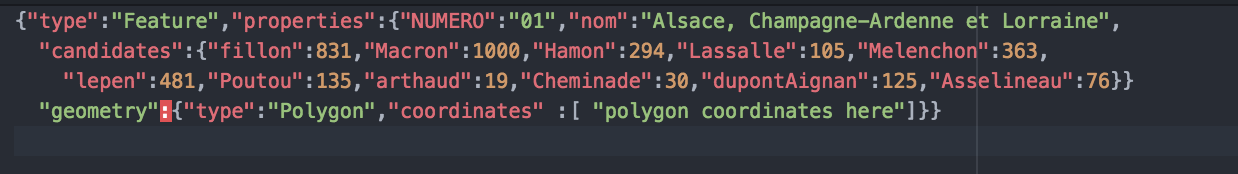


Grâce à ce fichier, on récupère le code Insee de la commune, ce code va nous permettre de l’associer avec son polygone dans les fichiers geojson (puisque le code insee est présent dans tous les fichiers geojson des ville, le deux premier chiffres représentent le code du département et le code département permet d’accéder à la région.



Extrait du fichier CSV communes fourni par data.gouv.fr

On ajoute donc +1 dans les fichier GeoJSON ville, département et région correspondants pour le candidat.



Ci-dessus on peut voir le GeoJSON complet (sans les coordonnées) pour la région grand Est. On peut voir la parties candidates dans properties qui permet d’accéder au nombre de tweets par candidats dans la région sélectionnée.

Les fichiers sont donc remplis et utilisables mais ne sont pour l’instant pas accessibles depuis une application extérieure. IL a donc été mis en place des routes dans le serveur pour récupérer les fichiers JSON :

* /region
* /departement/idRegion
* /city/idDepartement
* /date

La route /date et faite pour récupérer l’évolution temporelle des tweets.

Pour les fichiers département et ville on ne peut pas tout afficher au même moment. Il est donc demandé l’ID de l’entité « au-dessus ».

Par exemple :

* /city/33 renverra les fichiers geoJSON de toutes les villes en gironde.
* /departement/02 renverra les fichiers geoJSON de tous les départements en Nouvelle Aquitaine. Car la nouvelle aquitaine a un ID=02.

Pour finir le client peut accéder à l’application, et récupérer les fichier geoJSON.

## Problèmes rencontrés

## Rendu de l’application

## Tests

## Limites du projet

# Comment utiliser Twelec ?

# Évolutions possibles

# Conclusion