

2020 - 2021



# Rapport projet annuel



**Nom - Prenom :** BENDAVID Natane, WADE Cheikh Abdourahmane

**Promotion :** 4 IABD 2

**Matière :** Projet Annuel

**Intervenant :** WAJNBERG Charles David

## Table des matières

Introduction .....	3
Contexte du projet .....	3
Projet similaire .....	3
Objectifs .....	4
Description fonctionnelle .....	5
Pipeline de déploiement des données et du modèle de ML .....	5
Pipeline de déploiement pour la génération de la data visualisation .....	9
Pipeline pour l'api de prédiction immobilière .....	11
Application web pour estimer un bien immobilier .....	13
Architecture technique .....	17
Architecture du code .....	18
Données mises en œuvre .....	19
Algorithmes et outils utilisés .....	20
Gestion de projet .....	21
Résultats obtenus .....	22
Sources d'informations pour réaliser le projet .....	23

## Introduction

Dans ce rapport de projet annuel, nous allons mettre en évidence toute l'élaboration et le déroulement de la réalisation de ce projet. Nous expliquerons la partie technique et fonctionnelle mise en place pour ce projet. Nous mettrons en évidence les choix réalisés pour aboutir à notre résultat final

### Contexte du projet

Nous commençons par un rappel de l'objectif du projet Annuel. Ce projet a pour but de mettre en application les savoirs issus des cours suivis à l'ESGI.

Le but du projet annuel est de réaliser un projet qui respecte les différents critères :

- L'utilisation d'une plateforme et d'une architecture dans le Cloud
- L'automatisation de tous les traitements réalisés
- Le déploiement de notre application
- La réalisation de l'objectif via le savoir issus des cours de l'ESGI
- La mise en place d'une partie Intelligence artificielle

Le sujet du projet sur lequel nous avons décidé de travailler concerne l'immobilier.

Lors de nos premières réunions en groupe, nous avons évoqué nos différents sujets qui pourraient nous intéresser. C'est alors que le sujet de l'immobilier nous a interpellé car dans notre vie privée nous étions tous les deux dans une période de recherche d'appartement. Le sujet de l'immobilier a été tout de suite une idée validée par l'ensemble de l'équipe.

### Projet similaire

Une fois le sujet validé par l'ensemble de l'équipe, nous avons décidé de s'informer sur le monde de l'immobilier ainsi que les sites existants sur le marché.

Le projet de référence que nous avons vu et utilisé durant nos recherches d'appartement dans le cadre de notre vie privée est le site internet [meilleurs agents](#).

meilleurs agents

Menu

Ex : "10 rue du Château", "Paris 15", "69002"...


CARTE

Prix immobilier > Ile-de-France > Paris > Paris 12e > Picpus > Rue du Faubourg Saint-Antoine > N°242-b

Prix au m<sup>2</sup> Loyer au m<sup>2</sup>

**242 B rue du Faubourg Saint-Antoine, 75012 Paris**

Estimations de prix MeilleursAgents au 1 juillet 2021. [Comprendre nos prix](#)

 Prix m<sup>2</sup> moyen  
**10 152 €**  
de 8 466 € à 11 848 €

APPARTEMENT Indice de confiance ●●●●●

Estimez votre bien en fonction de ses caractéristiques

[Estimer un bien en ligne](#)

Ou obtenez les prix de vente des biens à proximité

[Obtenir les prix de vente](#)

Sur ce site web nous pouvons renseigner une adresse postale du bien à estimer. Le site meilleurs agents nous renvoie alors une estimation moyenne du prix au mètre carré de l'adresse donnée, ainsi qu'une fourchette de prix.

Nous nous sommes alors donnés comme objectif de réaliser une application du même type.

## Objectifs

L'objectif de notre projet est de réaliser une Api et une application avec de l'intelligence artificielle capable d'estimer un bien immobilier pour une adresse et surface données. Le but est que notre application et Api nous donne une estimation la plus précise du bien donné.

Comme dit au-dessus, ce projet a deux objectifs :

Le premier est de créer une Api permettant de renvoyer plusieurs informations sur un bien immobilier (prix estimé, prix du voisinage et distance entre les biens estimés). Cette api permettra aux développeurs voulant intégrer sur le site internet ou application immobilière un estimateur d'un bien immobilier.

Le second est la création d'une application web permettant aux utilisateurs lambda, agents immobiliers ou même particuliers ou professionnels voulant estimer, vendre leurs biens immobiliers et connaître une estimation de leurs prix.

Comme décrit un peu plus haut notre objectif pour ce projet est de réaliser cet estimateur de bien immobilier pour :

- des particuliers ou professionnels voulant estimer leurs biens immobiliers.
- des particuliers ou professionnels voulant vendre leurs biens immobiliers
- des particuliers ou professionnels voulant acheter un bien immobilier
- des particuliers ou professionnels voulant investir dans un bien immobilier
- des développeurs voulant mettre en place un système d'estimation de bien immobilier sur leurs sites web

## Description fonctionnelle

Pour la partie fonctionnelle de ce projet d'estimateur immobilier nous avons 3 grandes parties fonctionnelles :

1. Une première partie récupération, nettoyage, traitement et préparation des données avec la création de modèle de Machine Learning de régression et de clustering ainsi que le stockage de l'ensemble dans le cloud. Toute cette partie est automatisée
2. Une seconde partie qui permet de générer toute la partie data visualisation de l'ensemble des données mises à jour. Ce traitement est aussi automatisé.
3. Une dernière partie qui est l'API qui permet de prédire le prix d'un bien immobilier par rapport à son adresse postale et sa surface.

Dans chaque partie nous détaillerons précisément toutes les démarches réalisées pour aboutir à chaque fonctionnement et nous schématiserons le résultat.

## Pipeline de déploiement des données et du modèle de ML

Dans cette pipeline automatisée via une tâche CRON, nous avons réalisé différentes étapes.

La première partie de cette pipeline est la récupération des données. Pour établir notre modèle de Machine Learning plus tard nous avons besoin d'une grande quantité de données pour avoir un modèle performant.

Nous avons alors utilisé les données proposées par Etalab sur data-gouv.fr : **Demandes de valeurs foncières géolocalisées** (<https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres-geolocalisees/>). Sur ce dataset, **Etalab** nous fournit un historique de données sur les 5 dernières années.

# Index of /geo-dvf/latest/csv/

---

<a href="#">../</a>	
<a href="#">2016/</a>	01-Jun-2021 15:32
<a href="#">2017/</a>	01-Jun-2021 15:32
<a href="#">2018/</a>	01-Jun-2021 15:30
<a href="#">2019/</a>	01-Jun-2021 15:31
<a href="#">2020/</a>	01-Jun-2021 15:30

Comme sur l'image ci dessus, nous avons à disposition un file système avec un dossier par années (sur les 5 dernières années) qui contient un fichier compressé csv « **full.csv.gz** » qui comprend toutes les données pour chaque département. Pour notre pipeline, nous téléchargeons les 5 fichiers csv des 5 dernières années courantes. Nous décompressons toutes les données et l'utilisons dans un seul et même dataset.

Une fois les données mise à disposition, nous téléchargeons un second dataset de data-gouv qui comprend le numéro de région correspondant au numéro de département (<https://www.data.gouv.fr/fr/datasets/regions-departements-villes-et-villages-de-france-et-doutre-mer/>). Ce dataset est stocké puis récupéré depuis le bucket S3 créer pour le projet immobilier. Une fois le dataset chargé, on le fusionne avec le dataset immobilier par rapport au code département donné pour obtenir le code région correspondant

Une fois toutes les données correctement chargées nous nettoyons les données mise a disposition pour ne pas avoir de mauvaises données, des données incomplètes ou des données qui pourraient altérer la création du modèle ML. Sur les 40 variables du dataset nous gardons uniquement celle-ci :

*'id\_mutation', 'date\_mutation', 'nature\_mutation', 'valeur\_fonciere', 'code\_postal', 'code\_commune', 'code\_region', 'code\_departement', 'code\_type\_local', 'type\_local', 'surface\_reelle\_bati', 'nombre\_pieces\_principales', 'latitude', 'longitude'*

Maintenant-nous filtrons les données de manière à ne garder que :

- Les données correspondantes à une vente ou vente en l'état futur d'achèvement
- Les valeurs foncières non vide et différentes de 0
- Les surface du biens non vide et différent de 0
- La longitude et la latitude non vide
- Un nombre de lots égales a 0
- Un code départements non vide

Une fois les données nettoyées et filtrées, nous procédons à quelques traitements sur certaines variables. Nous créons aussi la variable *prix\_au\_metre\_carre* qui nous servira plus tard dans notre modèle. Pour terminer sur le traitement des données, nous regroupons toutes nos données des différentes années dans un même dataframe pour l'utiliser ensuite dans nos modèles.

Une fois toute la partie traitement terminée, nous commençons la partie ML. Mais avant cela, pour reproduire l'estimation d'un agent immobilier nous avons besoin d'une autre donnée qui n'est pas présente dans le dataset. Pour estimer correctement un bien, nous ne pouvons pas uniquement nous fixer sur le prix au mètre carré d'un bien, mais nous avons besoin du prix au mètre carré des autres biens aux alentours du bien estimé. Pour cela nous allons alors récupérer les 10 biens les plus proches et récupérer la moyenne du prix au mètre carré de l'ensemble de ces biens. Pour déterminer cette variable et récupérer les biens les plus proches, nous devons déterminer la distance entre le bien estimé et les autres biens de la région et récupérer les 10 biens les plus proches. C'est à partir des données latitude/longitude que nous pouvons calculer cette distance. Nous mettons alors en place un modèle de partitionnement spatial appelé le modèle *BallTree* ou modèle de *voisinage* disponible dans la librairie *sklearn*.

Malheureusement, il est impossible d'utiliser ce modèle directement sur l'ensemble des données car le traitement serait trop long. C'est pour cela que nous découpons notre dataset par régions et créons alors un modèle de partitionnement par régions.

Ce modèle nous permettra ensuite d'estimer pour chaque bien le prix au mètre carré du voisinage.

Pour réaliser ce modèle de partitionnement :

- On sépare les données par numéro de région
- On crée ensuite le modèle avec les données latitude/longitude
- On enregistre et charge les modèles de chaque région dans le bucket S3
- On utilise ensuite le modèle pour récupérer la distance moyenne entre les 10 biens voisins les plus proches ainsi que leurs indices. La distance obtenue est en haversine. Pour la convertir en Km on doit multiplier par la surface de la terre soit 6371 km)
- Avec ces données on récupère pour chaque bien ses 10 voisins les plus proches et on calcule la moyenne du prix au mètre carré de ces 10 biens. On appelle cette variable *prix moyen du quartier*.
- On sauvegarde chaque dataset par région et les charge sur le bucket S3

Une fois toutes ces étapes réalisées, on obtient bien alors la variable qui nous intéresse qui est le prix moyen du quartier que nous utiliserons comme variable pour notre modèle de ML

Pour terminer cette pipeline, nous devons maintenant entraîner notre modèle de Machine Learning. Pour cela nous avons réalisé différents tests avec différents modèles :

- Un MLP avec la lib Keras de Tensorflow
- Un GridSearch avec la lib XGboost
- Un RandomForestRegressor avec la lib scikit-learn

Après ces différents tests réalisés avec ces modèles, nous avons obtenus de meilleur résultat avec un RandomForestRegressor.

Nous avons obtenu avec ce modèle 7% d'erreur. Les inputs (variables) que nous utilisons pour entraîner ces modèles sont :

*'latitude', 'longitude', 'surface\_reelle\_bati', 'prix\_metre\_carre', 'code\_region',  
'prix\_moyen\_quartier'*

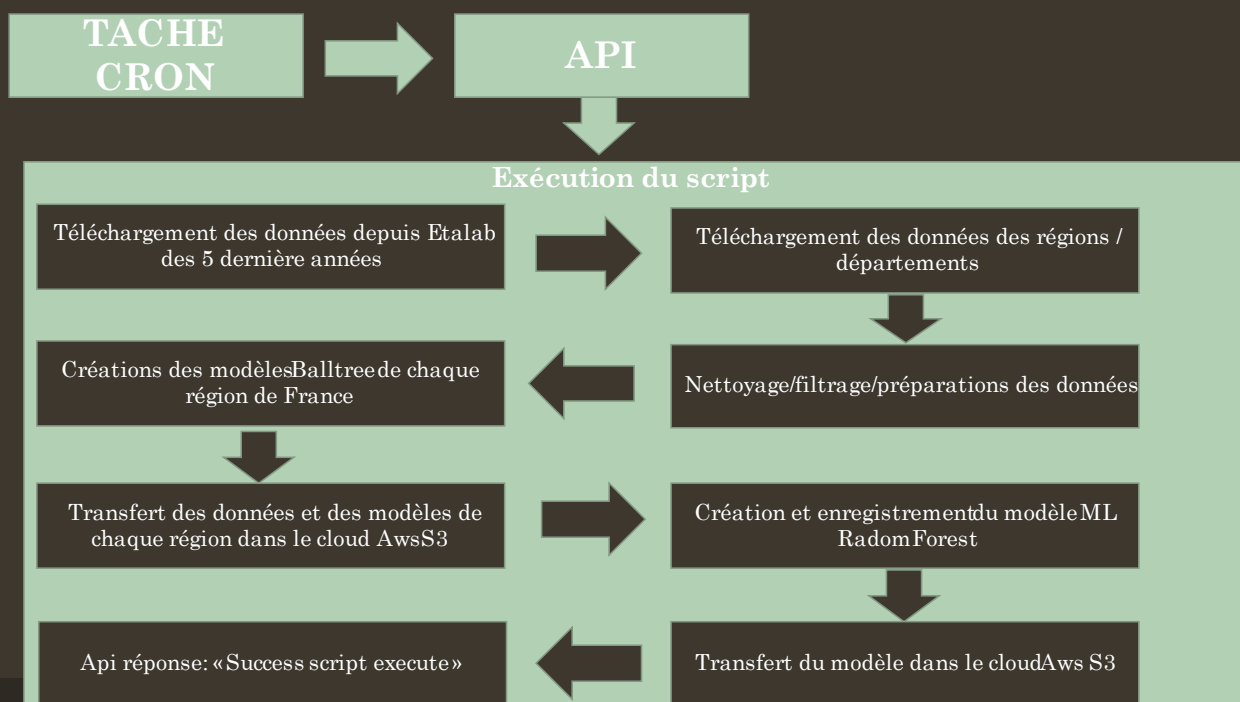
Une fois le modèle entraîné nous le sauvegardons via la lib pickle (disponible directement sur sklearn) et le chargeons sur le bucket S3

Tous ces traitements sont exécutés via un call API. Pour cela nous utilisons la librairie python Flask pour mettre en place cette API.

Pour automatiser alors tous les traitements, nous lançons une tâche CRON tous les 6 mois pour récupérer les nouvelles données mises à jour par Etalab, effectuer les traitements sur les données, les mettre à jour sur le bucket S3 et réentraîner le modèle de ML.

Pour schématiser tous les traitements réalisés et expliqués ci-dessus, voici un schéma qui résume les grandes étapes de réalisation de ce workflow :

## Pipeline de déploiement – Les données





## Pipeline de déploiement pour la génération de la data visualisation

Pour la réalisation de cette pipeline, nous avons aussi décidé d'automatiser tous les traitements réalisés via une tâche planifiée (CRON).

De la même manière que le premier pipeline, nous allons récupérer les données sur le site data-gouv mise à disposition par Etalab. Nous récupérerons les 5 dernières années pour réaliser nos graphiques de visualisations.

Une fois les données des valeurs foncières téléchargées, nous procédons au traitement et filtrage des données. Pour cela nous ne gardons uniquement que les variables qui nous intéressent :

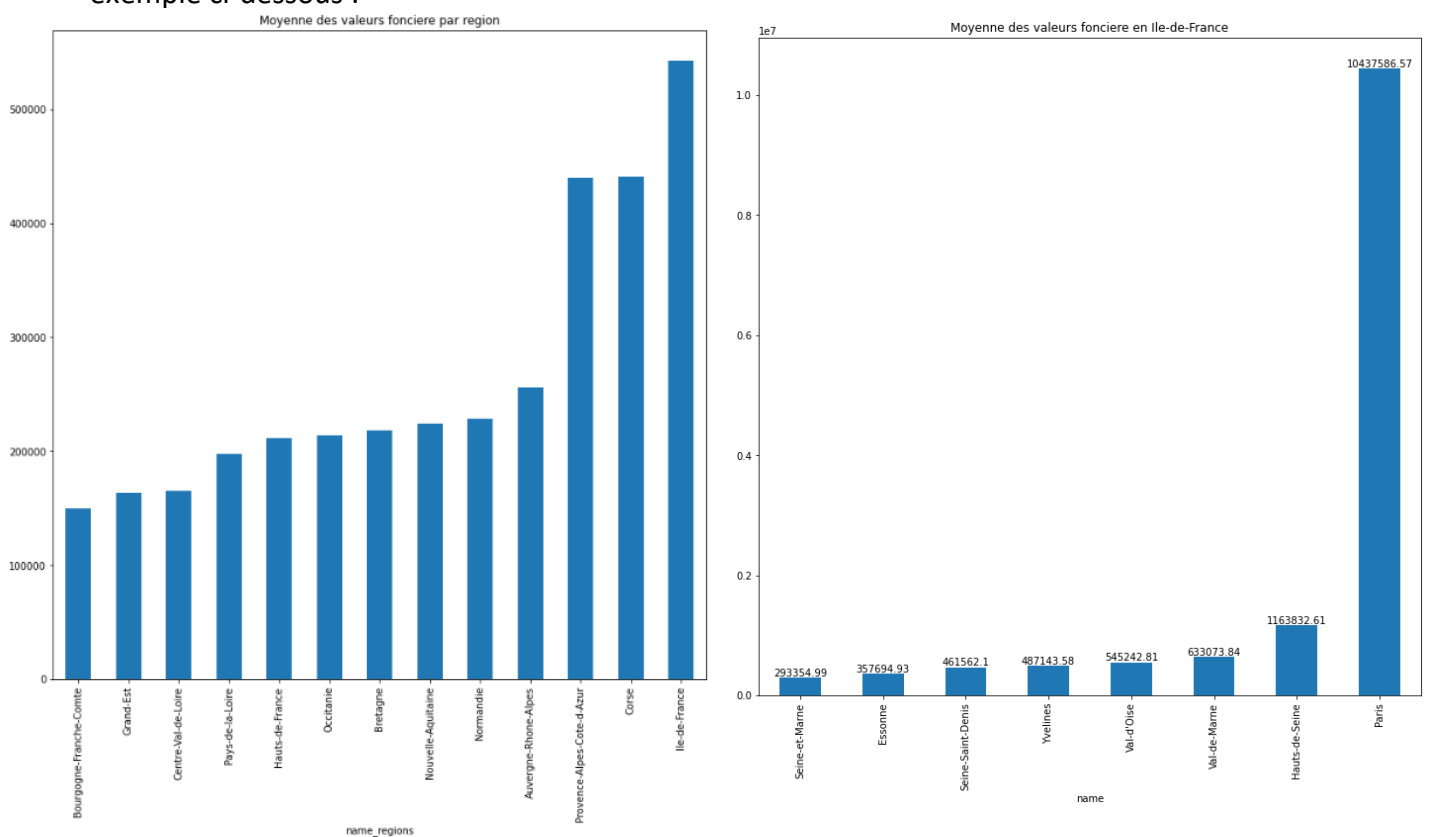
*'id\_mutation', 'date\_mutation', 'nature\_mutation', 'valeur\_fonciere', 'code\_postal', 'code\_commune', 'code\_region', 'code\_departement', 'name', 'code\_type\_local', 'type\_local', 'surface\_reelle\_bati', 'nombre\_pieces\_principales'*

Une fois les variables récupérées nous procédons à un nettoyage de celles-ci ainsi que quelques transformations. Nous calculons la variable `prix_metre_carre` par rapport à la valeur foncière et la surface. Nous filtrons bien sûr les données en ne récupérant que les maisons et les appartements.

Une fois nos données correctement préparées nous nous penchons alors sur la restitution de nos données. Pour cela nous avons décidé de générer nos graphiques sous différents angles.

1. Des données regroupées par rapport aux régions
2. Des données regroupées par rapport aux départements d'une région

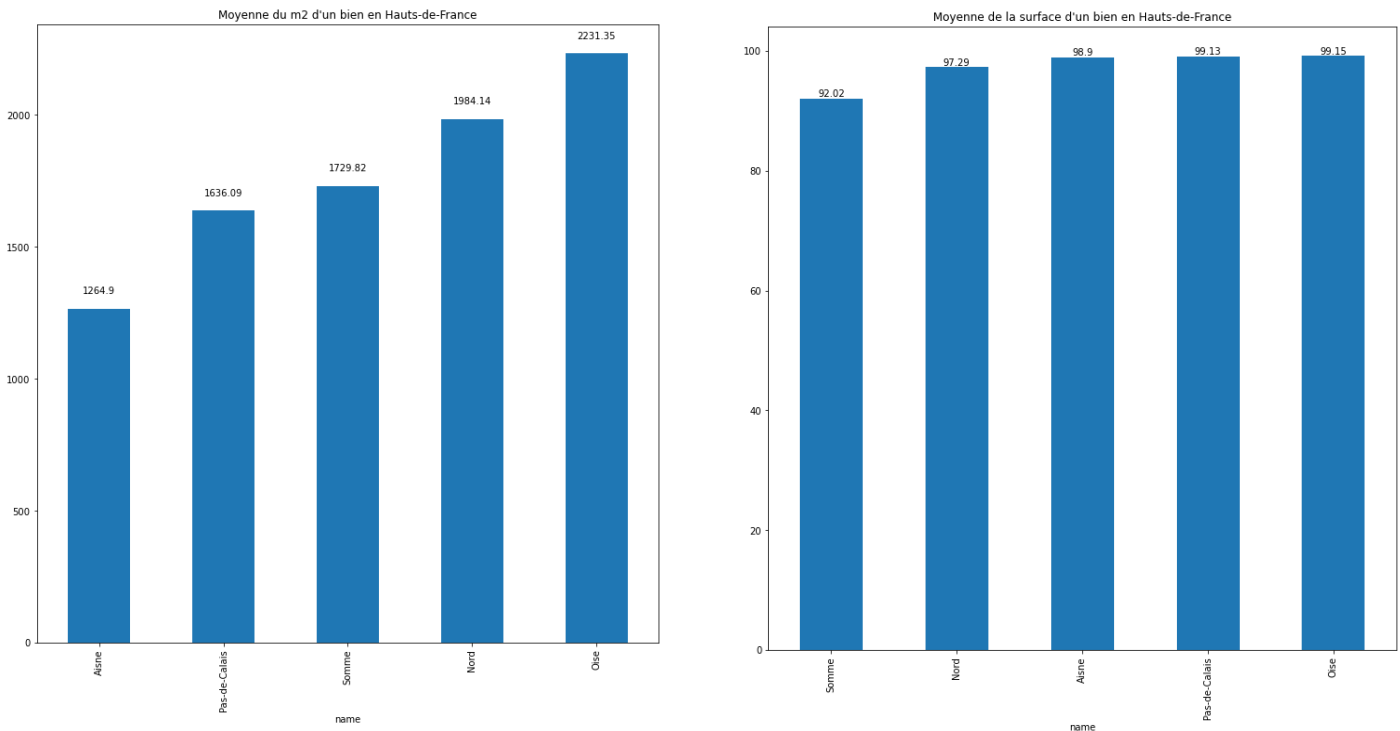
Sur ces deux axes différents nous avons décidé de réaliser une restitution sur les valeurs foncières par régions et plusieurs par rapport aux départements d'une région. Voici un exemple ci-dessous :



Sur la restitution de gauche nous avons la moyenne des valeurs foncières répartie par régions et sur la droite la moyenne des valeurs foncières d'une seule région (ici l'île de France) de l'ensemble des départements de cette région.

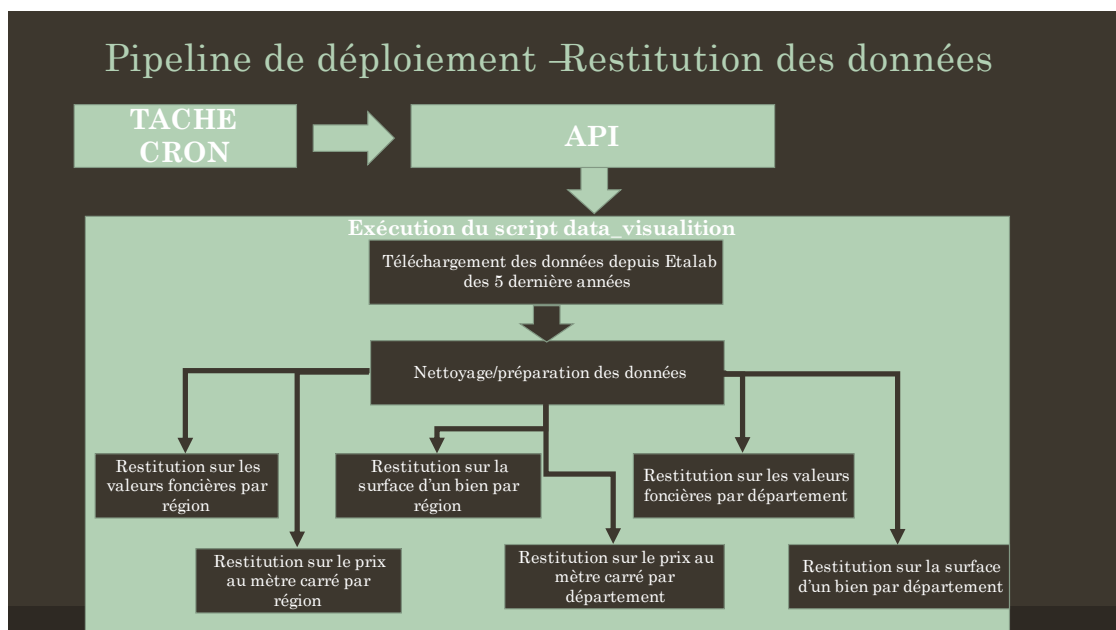
Nous avons aussi réalisé une restitution sur la moyenne de la surface d'un bien, et une restitution sur la moyenne du prix au m<sup>2</sup> d'un bien. Ces deux restitutions sont aussi réalisées sur les régions et les départements comme l'exemple ci-dessus.

Nous pouvons voir ici un exemple des graphiques générés par départements (région Haute France) pour les deux restitutions :



Pour terminer, tous ces traitements sont exécutables via une Api GET à l'aide de la lib Flask qui est appelée et automatisée via une tâche Cron.

Voici un schéma récapitulatif de l'ensemble des traitements réalisés pour cette pipeline :



## Pipeline pour l'api de prédiction immobilière

La pipeline que nous avons réalisé ici permet aux développeurs d'intégrer sur leur application notre Estimateur immobilier via L'api et nous servira pour l'intégrer à notre application web d'estimation immobilière.

Premièrement, nous avons réalisé cette Api via la lib flask. Pour faire appel à cette api nous devons faire une requête POST en passant dans le body plusieurs informations sous le format d'un fichier JSON.

Les inputs sont les suivants :

- Latitude de type String ou number
- Longitude de type String ou number
- Le code du département concerné de type String ou number
- La surface du bien de type String ou number

Pour récupérer les informations **Latitude** et **longitude**, nous utilisons l'API mise a disposition par le gouvernement pour retrouver les informations d'une adresse donnée. L'api utilisé est geo.api.gouv (<https://geo.api.gouv.fr/adresse>). Nous utilisons cette api en faisant un call api a cette url ci : <https://api-adresse.data.gouv.fr/search/?q=monadresse>. Cette api nous renvoie plusieurs informations sur l'adresse dont les coordonnées (latitude, longitude) ainsi que le **numéro de département** associé à l'adresse.

Avec ces différentes informations récupérées, nous pouvons alors faire appel à notre API en lui passant dans un script JSON les différents éléments ainsi que la surface au mètre carré du bien estimé. Voici un exemple :

```
{
  "latitude": "44.84749",
  "longitude": "-0.586692",
  "code_departement": "33",
  "surface_reelle_bati": "150"
}
```

Une fois les différents éléments réunis nous passons alors au traitement.

Lorsque les informations arrivent à l'api, premièrement nous parons le body json de la requête pour vérifier si tous les éléments sont bien renseignés dans le JSON. Une fois les informations récupérées nous récupérons le code de la région correspondante au numéro de département donnée.

A partir de ce code régions, nous allons télécharger le model de voisinage (Balltrre) depuis notre bucket S3 qui correspond au modèle de la région concernée. Une fois le modèle téléchargé, nous récupérons aussi l'ensemble des données de la région stockées sur s3. Une fois les deux fichiers téléchargés, nous utilisons le modèle pour récupérer les 10 biens les plus proches ainsi que la distance moyenne de ces biens. Une fois l'indice des données récupéré à partir du modèle, nous récupérons les données via le fichier de données

téléchargé juste avant. Une fois les 10 biens récupérés nous calculons alors le prix moyen du quartier pour notre bien immobilier.

A partir de cette étape, nous avons tous les éléments qui nous permettent d'utiliser notre IA via notre modèle de ML, pour récupérer le prix au mètre carré du bien estimé.

Tout d'abord, pour réaliser cela, nous téléchargeons notre modèle depuis S3 si celui-ci n'est pas présent sur notre serveur. Une fois le modèle récupéré, nous le chargeons et l'utilisons pour la prédiction. Pour cela nous lui donnons en entrée du modèle :

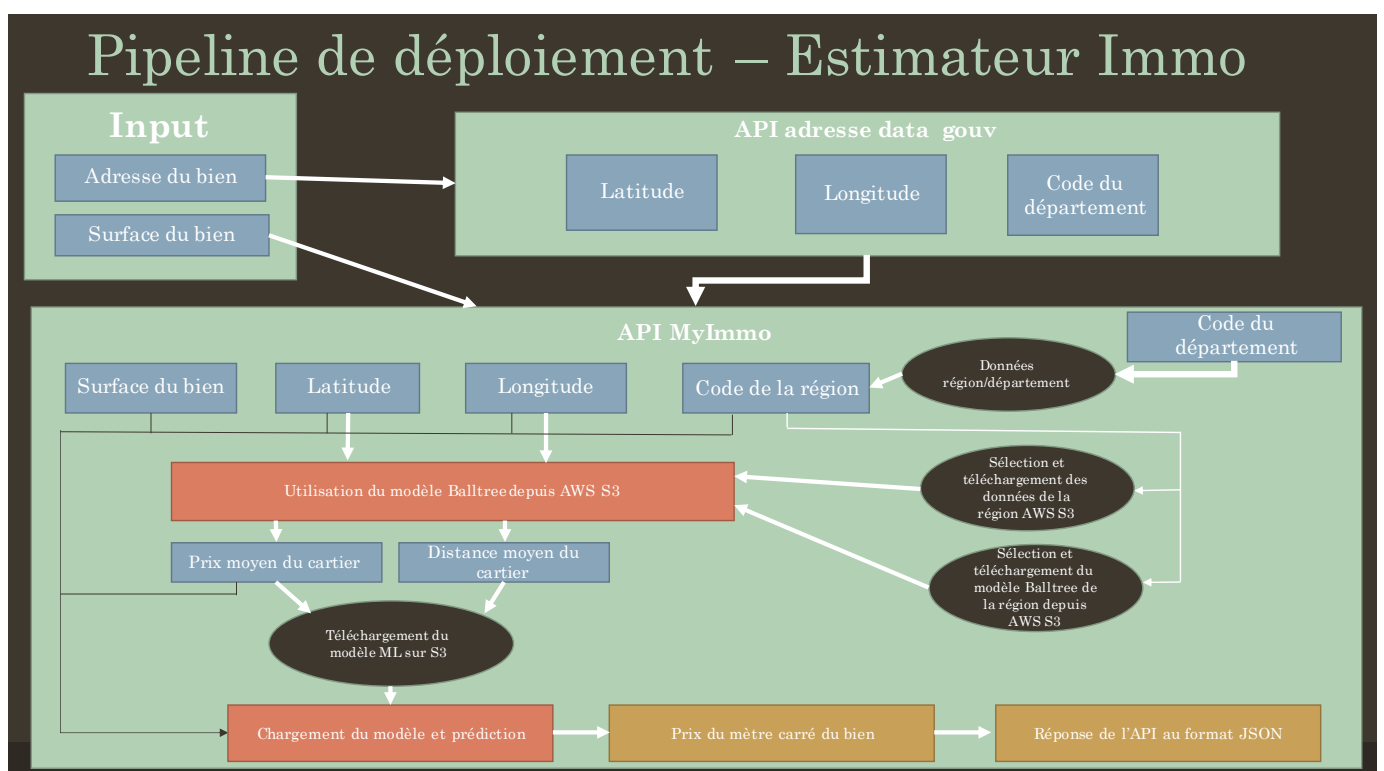
- Latitude
- Longitude
- Surface
- Code région
- Prix moyen du quartier

Une fois ces données envoyées, le modèle nous renvoie le prix au mètre carré estimé pour ce bien.

Pour terminer nous renvoyons à l'appel de cette api un fichier JSON avec :

- La prédiction (prix au mètre carré du bien)
- Le prix moyen du quartier
- La distance moyenne des 10 biens les plus proches
- Latitude
- Longitude
- Surface
- Code région

Voici un schéma récapitulatif des traitements réalisés lors du call API de notre estimateur immobilier :



## Application web pour estimer un bien immobilier

Pour terminer l'ensemble de nos fonctionnalités, nous avons réalisé une application web qui permet aux utilisateurs tels que des agents immobiliers ou des personnes voulant estimer un bien immobilier via notre IA Estimateur immobilier.

Dans cette interface web, nous avons réalisé plusieurs pages :

- Une page qui permet aux utilisateurs de prédire le prix aux mètres carré de son bien immobilier via notre IA depuis l'API
- Une page avec la restitution de nos données
- Une page pour les développeurs qui voudraient utiliser notre API

Concernant notre première page principale qui est la page pour estimer son bien immobilier,

The screenshot shows a web form titled "Estimer votre bien immobilier avec MyImmo !". It features a dark header with the "MyImmo" logo and navigation links for "HOME", "DATA", and "API". The form contains three input fields: "Adresse de votre bien" with a location pin icon, a larger "Selectionner une adresse" dropdown menu, and "Surface financiere m²". A blue button labeled "ESTIMER MON BIEN - €" is positioned below the inputs.

Nous avons un premier champ qui permet à l'utilisateur de renseigner une adresse postale. Dès que l'utilisateur rentre un caractère, le bloc en dessous se met à jour avec les données renvoyées par l'api des adresses du gouvernement (geo.api.gouv.fr). Une fois l'adresse renseignée, l'utilisateur choisit une adresse parmi les adresses proposées dans le bloc en dessous. Pour terminer, l'utilisateur renseigne alors la surface de son bien. Il peut alors cliquer sur le bouton pour estimer son bien.


Voici un exemple ci-dessous :

This screenshot shows the same form as above but with example data. The "Adresse de votre bien" field contains "242 rue du faubourg saint antoine". The "Selectionner une adresse" dropdown is open, displaying a list of suggestions: "242 Rue du Faubourg Saint-Antoine 75012 Paris" (highlighted), "Rue du Faubourg Saint-Antoine 75011 Paris", "Rue du Faubourg Saint-Antoine 51000 Châlons-en-Champagne", "Rue du Faubourg Saint-Antoine 84120 Pertuis", and "Rue du Faubourg Saint-Antoine 41400 Saint-Georges-sur-Cher". The "Surface financiere m²" field contains "150". The "ESTIMER MON BIEN - €" button remains at the bottom.

Une fois que l'utilisateur a cliqué sur le bouton, une pop-up s'affiche avec les différentes informations :

- Dans un premier bloque nous avons les informations sur l'adresse du bien, les coordonnées d'où se situe le bien ainsi que la surface du bien
- Dans le second bloc nous avons les informations renvoyées par notre API d'estimation :
  - o Le code de la région de l'adresse concerné
  - o Le prix moyen du quartier par rapport au voisinage
  - o La distance moyenne des biens à proximités
  - o La valeur estimer par notre IA du prix au mètre carré du bien.
  - o La valeur totale du bien

Voici la pop-up décrite ci-dessus :



**Mon bien immobilier**

**Adresse:** 242 Rue du Faubourg Saint-Antoine 750  
12 Paris

**Coordonnées:** long: 2.3896, lat: 48.8492

**Surface du bien:** 150 m<sup>2</sup>

---

**Estimation**

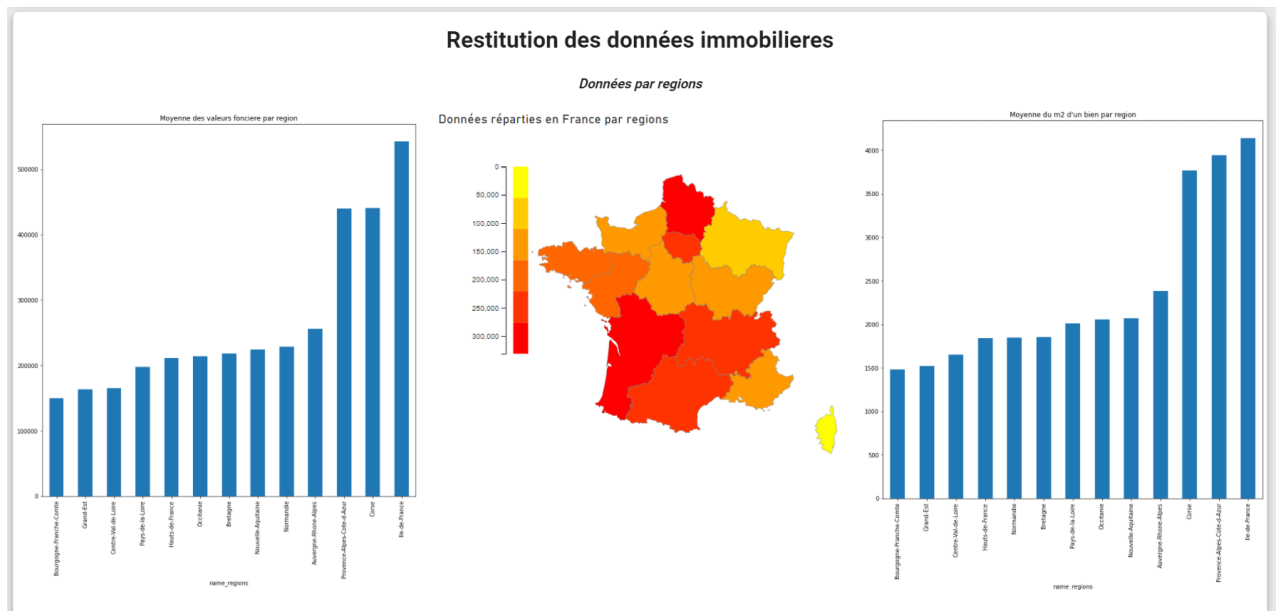
Code region : 11	Prix moyen du quartier : 13050.7	Distance moyenne des biens à proximités : 24.99 km
------------------	-------------------------------------	--

€ Valeur estimer : 8079.71€ /m<sup>2</sup> soit 1211956.5€ le bien

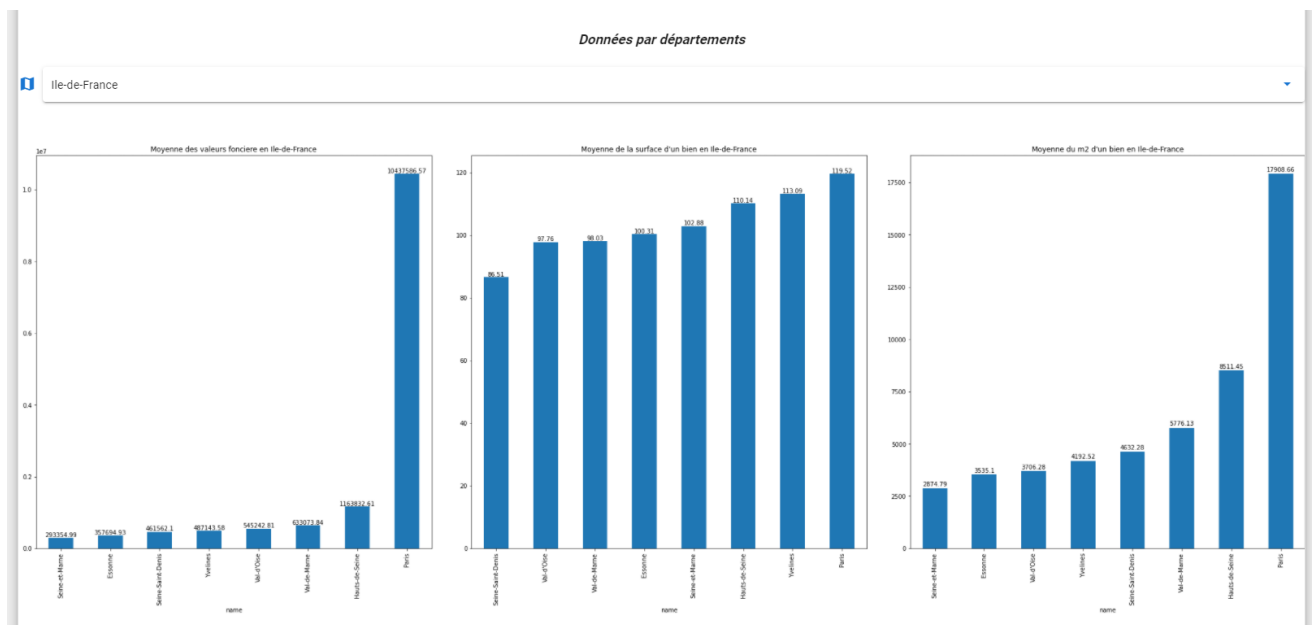
[FERMER](#)

Ensuite l'utilisateur peut se diriger vers la page data en cliquant sur le bandeau en haut à gauche qu'il le redirige vers quelques métriques utiles à la réalisation de ce projet.

Dans cette page nous avons une première partie avec la restitution des données par régions. Nous avons aussi une carte avec le nombre de données réparties en France par région comme ceci :

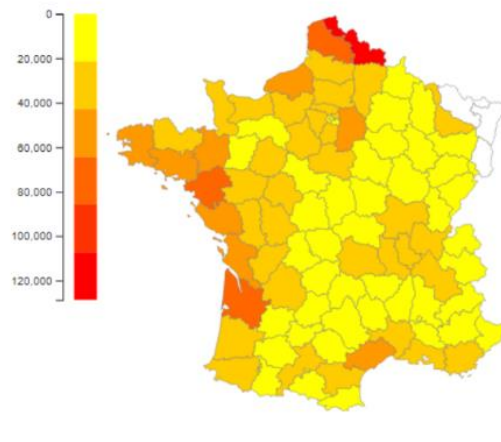


Dans la seconde partie de cette page, nous avons cette fois les données mais réparties par département. L'utilisateur peut ici choisir la région via une liste déroulante, pour visualiser les données de l'ensemble des départements de la région qui l'intéressent.



Pour terminer, nous avons ensuite la carte de la France avec le nombre de données mais cette fois ci, répartie par département comme ci-dessous.

Données réparties en France par départements



Pour terminer, la dernière page présente sur l'application web est l'onglet API. Cet onglet permet aux développeurs d'utiliser l'api via les informations fournies. Il donne une description des méthodes http qui doivent être utilisées, dans ce cas-là une requête http POST avec un body JSON. Les données nécessaires dans le JSON sont aussi présentes.

Pour finir l'utilisateur peut cliquer sur le bouton « Fichier API Postman » qui lui téléchargera le fichier à importer sur l'application Postman qui est une application pour tester les api pour les développeurs et d'intégrer directement l'api sur cette application.

## API développeur

POST

<http://13.37.61.224:5000/predict>

JSON

"latitude": Int

"longitude": Int

"code\_departement": Int

"surface\_reelle\_bati": Int

FICHIER API POSTMAN



## Architecture technique

Pour ce projet nous avons une contrainte technique qui était d'utiliser obligatoirement une plateforme cloud. C'est pour cela que nous nous sommes redirigés vers la plateforme cloud de AWS.

Concernant l'architecture technique que nous avons mis en place pour ce projet, nous avons utilisé différents composants :

- Une machine EC2 x-large sur AWS
- Une interface api gateway pour la machine EC2
- Un bucket S3 AWS
- La plateforme Heroku

Tous ces différents composants sont interconnectés entre eux.

Dans la machine EC2, nous avons stocké notre serveur web qui nous permet d'exécuter notre api ainsi que faire entraîner nos modèles de machine Learning et de clustering, transformer les données et préparer toute la data visualisation.

Nous avons aussi connecté le service api Gateway, pour nous permettre d'avoir une ip fixe pour notre machine EC2 pour les appels api. Via cette api nous pouvons faire appel au 3 routes différentes :

- GET « execute script » pour nettoyer, transformer et préparer les données ainsi que d'entraîner les données
- GET « data\_visualition » pour générer toutes les images de data visualisation
- POST pour utiliser notre modèle de machine learning et prédire le prix au mètre carré du bien donné.

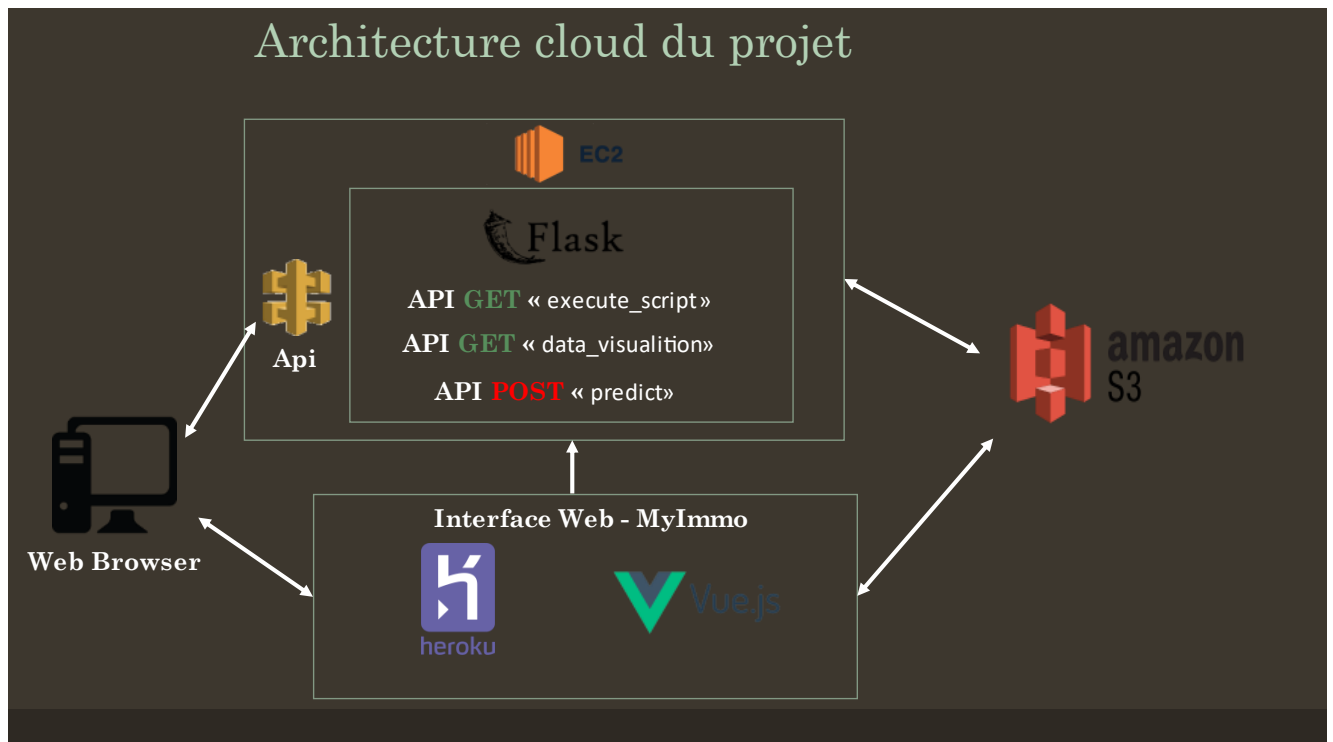
Toutes les URL sont en communication avec le Bucket S3 mise en place pour stocker :

- Les données transformées brutes
- Les données transformées et préparées par région
- Les modèles de voisinage pré-entraîné par région
- Toutes les images de data visualisations générées
- Le modèle de machine Learning pré-entraîné
- Le fichier des départements et régions de France

Enfin nous avons déployé notre application web via la plateforme cloud Heroku.

L'application communique avec notre api stockée sur la machine EC2 disponible via l'API Gateway et communique avec le bucket S3 pour récupérer les images de data visualisation.

Pour résumer, voici l'architecture technique complète de notre projet d'estimation Immobilière dans le cloud :



## Architecture du code

Concernant l'architecture mise en place pour le code, nous avons développé notre API via la librairie Flask. Pour notre serveur web nous avons alors réalisé une architecture très simple :

- Un fichier *app.py* qui est le point d'entrée de notre serveur web. C'est à partir de ce fichier que nous définissons les différentes routes de notre API.
- Un fichier *config.py* qui contient les différentes informations pour configurer notre bucket S3 ainsi que les noms des différents fichiers stockés
- Un fichier *data\_vis.py* où est écrit tout le code pour générer les images pour la restitution des données
- Un fichier *data\_prep.py* qui contient toute la partie préparation nettoyage et transformation des données ainsi que la réalisation de nos modèles de voisinage.
- Un fichier *ml.py* qui contient la partie réalisation de notre modèle de machine learning RandomForest
- Un fichier *predict\_ml* pour charger notre modèle ML et prédire le prix au mètre carré d'un bien donné
- Une architecture avec les dossiers :
  - o *Data* pour les données brutes des 5 dernières années

- Data-region pour les données transformées et préparées par régions (région de France)
- Data-vis pour les images des restitutions de données
- Model\_balltree pour les modèles de voisinage de chaque région de France

Concernant notre application web, nous avons développé le site via le Framework Vuejs qui permet de créer des applications web en Javascript. L'application fait des appels API vers les différents services de l'architecture cloud mise en place sur AWS pour notre projet.

## Données mises en œuvre

Les données que nous avons traitées pour ce projet sont des données provenant de Etalab. Etalab est un département de la direction interministérielle du numérique (DINUM), dont les missions et l'organisation sont fixées par le décret du 30 Octobre 2019. Il coordonne la conception et la mise en œuvre de la stratégie de l'État dans le domaine de la donnée. Il développe et anime la plateforme d'open data data.gouv.fr destinée à rassembler et à mettre à disposition librement l'ensemble des informations publiques de l'État, de ses établissements publics et, si elles le souhaitent, des collectivités territoriales et des personnes de droit public ou de droit privé chargées d'une mission de service public.

Le jeu de données que nous avons utilisé est dérivé du jeu de données Demandes de valeurs foncières diffusé par la DGFIP. Les données fournies par Etalab rejettent plusieurs informations en plus :

- La géolocalisation des biens via les coordonnées longitude et latitude.
- Le code postal du bien
- La date de la mutation du bien
- Etc....

Ces données sont fournies sous un format CSV compressé au format .gz

Les données fournies contiennent 40 colonnes. Les données sont variées avec des données :

- Géographique
- Temporelle
- Quantitative
- Qualitatives nominales

Etalab nous fournit un lien qui nous mène vers un File system où sont situés les différentes données. Ils sont regroupés dans un premier niveau par année (il n'y a que les 5 dernières années). Une fois dans le dossier de l'année, nous pouvons récupérer le fichier avec toutes les données de l'année directement ou Etalab a divisé les données dans des dossiers communes (1 fichier par communes) et département (un fichier par département).

Pour notre cas à nous, nous avons pris directement le fichier full.csv.gz qui contient toutes les données de l'année choisie.

Après transformations sur les données nous nous sommes intéressés plus particulièrement à 17 variables dont 13 variables brutes et 4 variables précalculées ou définies via un modèle :

**13 variables données par le dataset :** *id\_mutation, date\_mutation, nature\_mutation, code\_postal, code\_commune, code\_region, code\_departement, type\_local, latitude, longitude, valeur\_fonciere, surface\_reelle\_bati, nombre\_pieces\_principales*

**4 variables précalculées :** *prix\_metre\_carre, distance\_moyenne, index\_voisins, prix\_moyen\_cartier*

## Algorithmes et outils utilisés

Les algorithmes et outils utilisés durant ce projet sont nombreux :

Nous avons développé l'ensemble des traitements et les algorithmes de machine Learning en python.

Pour la partie traitement de données nous avons utilisé :

- Python
- Pandas
- matplotlib

Pour la partie Machine Learning et clustering :

- Tensorflow (testé mais non utilisé)
- XGboost (testé mais non utilisé)
- Scikit learn pour utiliser :
  - Le modèle Random foreste Regressor pour entrainer nos modèles à prédire le prix d'un bien au mètre carré
  - Le modèle neighbors.BallTree pour trouver les 10 biens les plus proches d'un bien donné

Pour la partie Web :

- Javascript
- Lib Vuejs

Pour la partie machine learning, comme décrite ci-dessus, nous avons utilisé différentes bibliothèques pour tester différents modèles et performances. Dans un premier temps nous avons voulu tester un simple MLP via la lib Tensorflow. Malheureusement, le modèle n'est pas assez performant et nous donne un taux d'erreur supérieur à 60%. Nous nous sommes ensuite redirigés vers la lib xgboost en utilisant un modèle grid search et nous arrivons déjà à de meilleures performances équivalentes à 20% d'erreur. Nous avons terminé nos tests avec un dernier modèle qui est le random Forest avec la lib Sklearn et nous sommes arrivés à 15 % d'erreur. Après plusieurs entraînements et différents paramétrages de notre modèle,

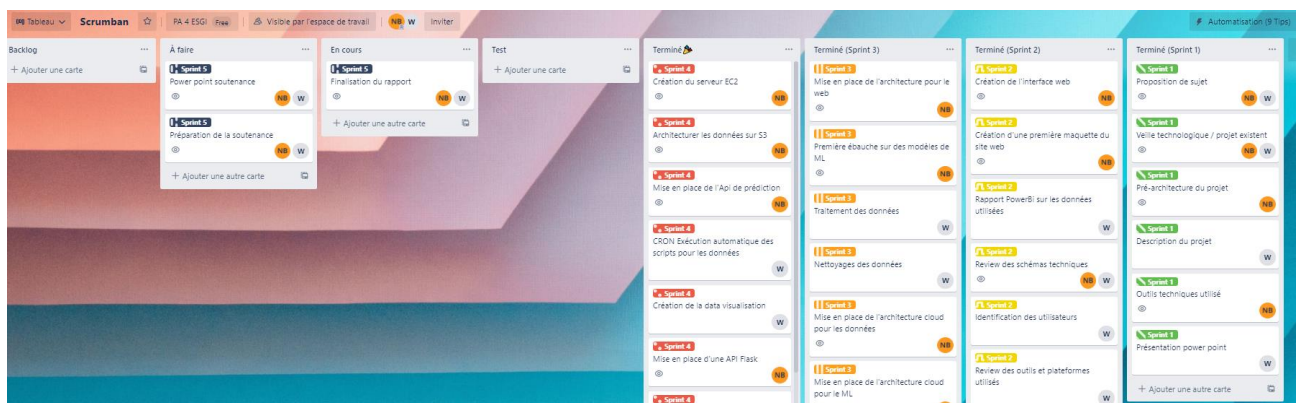
nous avons réussi à obtenir un modèle avec 7% d'erreur. Nous avons alors décidé d'utiliser le modèle RandomForest pour notre modèle de prédiction.

## Gestion de projet

Pour la partie gestion de projet, il a été très simple de nous répartir les tâches vues que nous n'étions que deux sur ce projet. Nous avons décidé de mettre en place un trello pour étiqueter toutes les tâches que nous devons réaliser puis nous les répartir.

Pour cela, nous avons opté pour la méthodologie de travail ScrumBan qui est un mélange entre la méthode SCRUM répartie en sprint et Kanban la méthode par étapes de réalisation. Nous nous sommes alors donné un Sprint par rendu. Nous avons alors 5 sprints qui correspondent aux 5 rendus intermédiaires du projet.

Voici le tableau que nous avons mis en place pour la gestion de projet :



Chaque tâche a été étiquetée et répartie entre nous deux. Dès que quelqu'un avait terminé une tâche, il mettait à jour le tableau.

Pour résumer les grands points de la répartition des taches :

### Natane :

- Machine Learning
  - o Entrainement, prédiction, performance
- Modèle voisinage Balltree
- Application web
- L'architecture cloud

### Abdourahmane :

- Donnée
  - o Traitement
  - o Nettoyage
  - o Préparation
- Stockage et envoie de données dans le cloud

- Reporting des données
- Mise en service de S3

## Résultats obtenus

Pour terminer ce rapport, le bilan général de notre projet est plutôt concluant. Nous avons réussi à réaliser tous les objectifs que nous nous sommes fixés lors du commencement du projet, qui sont :

- Mettre en place un projet complet sur une plateforme Cloud
- Réaliser et produire un modèle plutôt performant
- Mise en place de l'application dans le cloud
- Communications avec nos différents composants et application dans le cloud
- Réaliser une architecture complète et prête à être mise en production

Nous avons réussi à établir notre IA de prédiction du prix d'un bien immobilier avec un bon score : 7% d'erreur. Nous avons aussi comparé nos résultats avec le site [meilleurs agents](#). Notre résultat était dans la tranche de prix que le site donnait. Nous sommes plutôt satisfaits du modèle que nous avons mis en place.

Concernant les améliorations possibles pour ce projet, nous aurions bien aimé mettre en place notre modèle dans une architecture cloud moins coûteuse que celle actuelle. Pour l'instant la machine que nous utilisons a besoin de beaucoup de ressources pour pouvoir effectuer tous les traitements ainsi que charger et prédire un résultat via le modèle. Pour ce projet étudiant et pour un souci de coût, nous sommes obligés de couper la machine pour ne pas être trop facturé.

Concernant une amélioration possible pour les données il aurait été intéressant de mélanger des données prise sur les sites de vente de maison comme leboncoin ou seloger pour traiter encore plus de données et être peut-être plus précis sur les régions où nous avons moins de données.

## Sources d'informations pour réaliser le projet

<https://www.economie.gouv.fr/cedef/estimer-prix-immobilier>

<https://www.fnaim.fr/3306-estimation-bien-immobilier-que-faut-il-prendre-en-compte.htm>

<https://www.meilleursagents.com/prix-immobilier/>

<https://app.dvf.etalab.gouv.fr/>

<https://www.economie.gouv.fr/particuliers/prix-immobilier-estimation-demande-valeur-fonciere>

<https://geo.api.gouv.fr/adresse>

<https://www.data.gouv.fr/fr/datasets/demandes-de-valeurs-foncieres-geolocalisees/>

<https://www.data.gouv.fr/fr/datasets/5c4ae55a634f4117716d5656/>