FARMING BY SAT

Partie informatique réalisée

Au cours du Semestre 1.

Partie 1 :

Rapport Complet fourni au Jury

PARTIE 1 : Recherche Satellites, Premier serveur et NDVI

A – Fonctionnement du programme Copernicus

La première étape de l'étude de faisabilité technique a consisté à choisir les outils les plus pertinents, par rapport aux attentes de la startup. Se présentait en effet le choix entre divers outils : imagerie satellite, photographies et mesures par survol en drone, ou encore effectuer des relevés sur le terrain, par exemple. La décision finale s'est arrêtée sur l'imagerie satellite, puisque cela permettra à la fois d'automatiser toutes les tâches, d'éviter les déplacements donc des frais supplémentaires, sur le terrain et de couvrir de grandes surfaces.

L'objectif était d'automatiser un maximum d'étapes lors de l'établissement des cartes, pour éviter à la startup de faire les tâches rébarbatives. Pour cela, les choix de conception ont été orientés vers l'élaboration d'outils capables de suivre un processus allant du téléchargement des données brutes du satellite vers la création de la carte qui sera envoyée à l'agriculteur.

Il n'y avait alors plus qu'à choisir le programme satellite parmi ceux qui existent. Les deux principaux concurrents sur ce marché sont les programmes américains LandSat et européen Copernicus. Les deux programmes fonctionnent à peu près sur le même principe, il existe des API pour accéder à leurs bases de données. La balance a finalement basculé vers Copernicus, car le principe des bandes de fréquences était plus simple à utiliser, que le programme est durable (jusque 2030 au moins), et qu'en plus, le programme est européen.

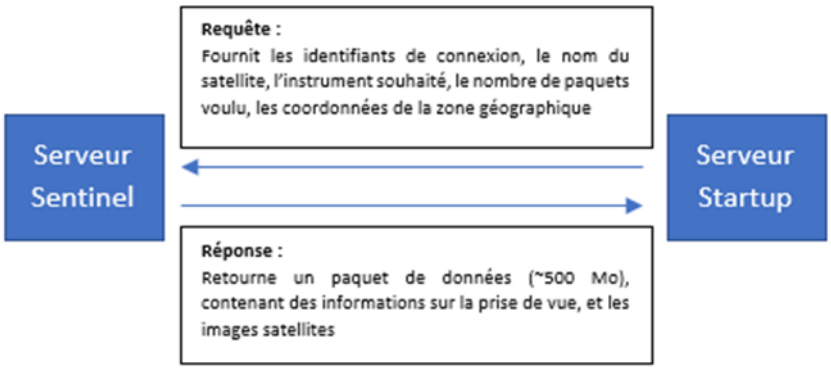
Le choix des satellites ayant été fixé à Copernicus, il a fallu chercher le moyen d'accéder à toutes ses données. Diverses possibilités étaient envisageables. Il existe une API Copernicus, accessible via un navigateur internet à l'adresse https://scihub.copernicus.eu/. Cette API propose une carte du monde, sur laquelle il faut dessiner la zone à étudier. Un menu déroulant contient un filtre pour sélectionner les données intéressantes. Il est possible de filtrer par date, par satellite, par instrument de mesure etc... Cette solution n'a pas été retenue, puisque pour atteindre une donnée en particulier, un grand nombre d'étapes doit être réalisé à la main.

Il existe également une copie des données de Copernicus sur un serveur Google, accessible à partir de Google Earth Engine. Cette méthode a été évitée également, étant donné que les données sont exploitables manuellement par navigateur internet.

La copie conservée par Google est également accessible sur Google Cloud Plateform, où les données sont accessibles par un fastidieux système d'arborescences, où chaque groupe d'image satellite est rangé selon ses coordonnées GPS

Finalement, en poussant les recherches sur les sites officiels du programme spatial Sentinel, un fichier de commande bash linux a été découvert. Celui-ci permet d'effectuer de nombreuses opérations de lecture et de tri dans les données des satellites Sentinel. Il a donc subi quelques adaptations pour notre utilisation : il a été simplifié et épuré au maximum, et le nombre d'arguments a été fortement réduit, pour n'arriver qu'aux coordonnées de la zone qui s'avère intéressante. Par exemple, les arguments concernant les identifiants de connexion au hub de Sentinel, les instruments qui sont intéressants, les satellites concernés, le nombre d'images désirées ont été intégrés au script pour ne pas avoir à les renseigner à chaque appel du script.

Le script bash permet donc de faire ceci :



Il a donc été possible de récupérer des données satellites depuis internet. Il suffit dès lors de renseigner les coordonnées GPS standard (avec latitude / longitude) pour avoir accès à de nombreuses données correspondant à un endroit donné (par exemple, le rayonnement infrarouge)

B – Création du premier serveur

Une fois l'existence du fichier Bash découverte, une répartition des tâches a été effectuée, certains ont travaillés sur la compréhension et la simplification dudit fichier, tandis que d'autres se sont attelés à savoir comment l'utiliser de manière automatisée. C'est ainsi qu'il a été constaté qu'il existait des modules en Node Js. Il s'agit d'une plateforme logicielle libre codée en JavaScript, permettant de mettre en place des serveurs web. Etant donné que la mise en place d'une application web avait été décidée, pour le côté multiplateforme, cette solution était parfaite.

Il a donc fallu se lancer dans l'apprentissage de Node JS, ne disposant de personne dans l'équipe qui ait déjà utilisé cette technologie. Pour ce faire, l'équipe dédiée à cette partie a suivi un tutoriel présent sur OpenClassroom. Ensuite, une communication en temps réelle était désirée pour le serveur. Il fut alors nécessaire de se pencher sur le système de socket, permettant d'établir ce genre de communication sur les serveurs, disposant également d'un module pour Node JS.

Cette version de site/serveur servait vraiment de base au projet, et à l'utilisation de bash et de Copernicus. Il y avait donc plusieurs boutons permettant d'indiquer au serveur de lancer telle ou telle fonctionnalité, afin de tester directement ces dernières, d'évaluer leurs robustesses, et surtout de ne pas avoir à relancer un téléchargement pour le test de fonctionnalités qui nécessitaient juste d'intervenir sur les dossiers/archives.

Les fonctionnalités présentes à ce moment étaient donc :

- Lancement du fichier bash de Copernicus, en lui passant latitudes et longitudes

- Extraction des images du fichier Zip ainsi téléchargé, en supprimant le reste, afin de répondre à la problématique posée par le poids des données de Copernicus.

- Suppression des fichiers de téléchargements de Copernicus, pour indiquer les noms de paquets téléchargés.

- Puis, lorsqu'il fut effectué, le calcul du NDVI à partir des images téléchargées.

Nous pouvions ainsi vérifier et garder une maitrise sur toutes les étapes du serveur. Cette partie technique fut assez complexe, étant donné que l'apprentissage et la découverte de Node JS était

en cours, et dans le même temps les modules utilisés qui s'avéraient pour certains assez complexes. Parmi, les modules utilisés à ce moment-là, il y avait :

- Http pour lancer le serveur.
- Fs et path pour la manipulation de fichiers et de dossiers dans le dossier de stockage du serveur.
- Socket io pour permettre les connexions et la communication en temps réel.
- Child_process pour lancer les fichiers bash et python depuis le serveur
- Csv-parse pour extraire les noms des archives téléchargées depuis un fichier csv créé par Copernicus.
- Adm-zip pour extraire uniquement les dossiers images des archives de Copernicus

Cette version de serveur était donc capable de télécharger les données nécessaires à la création d'un NDVI pour une zone donnée, de générer ce NDVI et de nettoyer les fichiers restants.

C – Calcul du NDVI par traitement d'images

Un souci se posait pour le traitement du NDVI. Les images téléchargées par Copernicus étaient au format JP2, un format qui paraissait à l'origine inexploitable. Plusieurs mesures ont été tentées pour contourner ce problème, notamment la conversion de ces images en jpg, via le python ou encore le langage Java, mais ces deux méthodes se sont soldées par des échecs. De plus, au cours des tentatives, il a été constaté que la conversion en JPG causerait une perte d'information. Des recherches ont donc été lancées afin de trouver un moyen d'exploiter tout de même les JP2.

Une piste a finalement fini par être exploitée, l'utilisation de la bibliothèque Rasterio pour Python. Cette bibliothèque était justement utilisée pour le traitement des images téléchargées via Copernicus. Cette découverte a pu permettre le lancement du codage du script Python nécessaire à la génération du NDVI.

La bibliothèque Rasterio permet en effet d'ouvrir un fichier JP2 et de lui appliquer diverses modifications. Le point intéressant de cette bibliothèque est qu'elle permet d'exploiter et de conserver les informations concernant les coordonnées UTM contenues dans le fichier image. Voici comment fonctionne ce système de coordonnées militaire :

Chaque endroit de la planète appartient à une zone de 6° de largeur. Il y en a 120 au total (60 hémisphère Nord et 60 hémisphère Sud). Par exemple, voici la zone 31N :

Rasterio permet de récupérer ces informations sur une image de type JP2. Celles-ci ont une résolution de 10m et représentent un carré de 100 km de côté. Chaque pixel de cette image peut donc facilement être mis en relation dans ce système de repérage planétaire.

Ce sont ces informations qui nous permettent de superposer plusieurs images satellites et de générer le NDVI. Il suffit alors de considérer les photographies infrarouges comme des matrices de pixels, où les 2 premières coordonnées représentent la position du pixel dans l'image, et où la 3ème dimension donne la réponse au rayonnement infrarouge. La formulation suivante, celle du NDVI, est ensuite appliquée à chacune des entrées de la matrice :

ndvi = ([Near InfraRed] - [RED]) / ([Near InfraRed]+[RED])

Le résultat de ces opérations est stocké dans une matrice suivant le modèle des deux premières. Chaque pixel se voit attribuer une valeur de NDVI comprise entre 0 et 1 (cette valeur pourra être changée d'échelle de façon proportionnelle).

Une fois que ce script python a été mis en place, il fallait l'appeler depuis le serveur, comme expliqué précédemment au moyen du module Child Process, en lui envoyant les chemins d'accès aux deux images de bandes B04 et B08. Le script python récupérait ces données comme il récupèrerait des arguments dans un exemple d'utilisation normal. Une image .tif était alors générée pour le NDVI, et stocké dans le dossier du serveur.

Il était donc possible d'obtenir les images NDVI automatiquement grâce au Python, Bash, et Node JS.

PARTIE 2 : Création du Site, découpage plus précis, nouveau serveur

A – Création du site

Une fois l'image NDVI obtenu, une proposition de réaliser un site Web qui permettrait à l'utilisateur de saisir ses informations personnelles, sur l'ensemble de ses parcelles, puis sur chaque parcelle précisément, ces informations seraient récupérées par un serveur puis stockées en interne, avec un référencement effectué par une base de données d'agriculteurs ayant utilisés le service. Il y aurait ensuite une interface de connexion, dans laquelle les agriculteurs pourraient se connecter pour récupérer les informations et conseils saisis par les experts de la start-up, et les dit experts pourraient se connecter pour accéder aux données saisies par les agriculteurs, et alors formuler leurs conseils. Il a été décidé, sous contrainte de temps, de ne réaliser que la première partie, pour déjà lancer une base de stockage et de réadapter le premier serveur à cette plateforme.

L'objectif était donc que l'utilisateur réponde à une suite de questions, afin d'obtenir le maximum d'informations sur ses parcelles, afin que le stockage des documents renseignés et du NDVI téléchargé soit fait, et que la récupération des données codées soit déjà prête. Il ne restera alors plus qu'à coder la partie administrateur et connexion, et de stocker les données textuelles au format désiré par l'équipe qui se chargera de cette partie.

En ce qui concerne le site internet, il a été décidé de rendre le site le plus simple et fonctionnel possible. En ce sens, l'utilisateur n'est invité qu'à renseigner les informations qui sont indispensables à sa prise en charge par la startup, comme ses coordonnées personnelles ou des informations simples sur ses parcelles. Les autres champs à remplir ne sont pas obligatoires. Et le renseignement de ces informations est la seule chose à faire pour recevoir les conseils agronomiques. En moins de 10 minutes, toutes les étapes sont passées, et l'agriculteur n'a plus qu'à attendre le mail qui va lui envoyer le conseil agronomique.

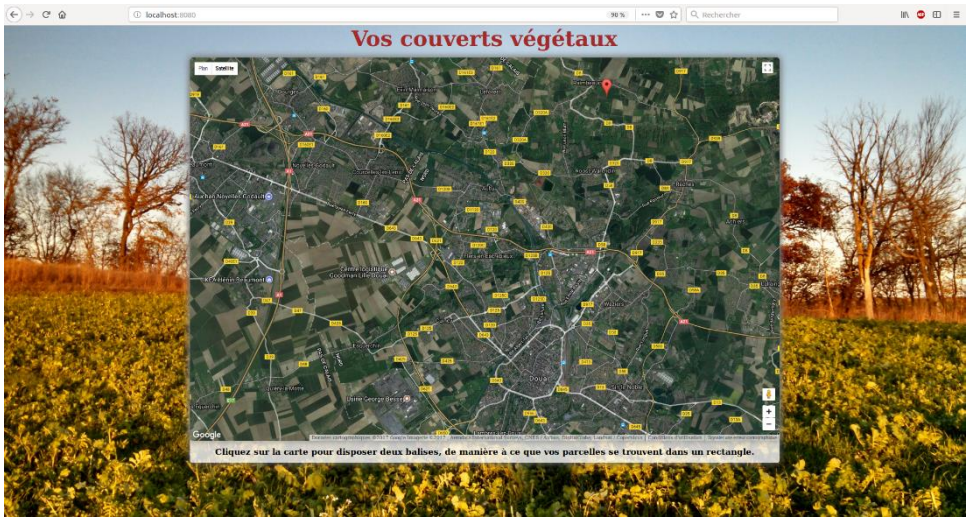
D'un point de vue visuel, le site a été travaillé pour que la simplicité d'utilisation saute aux yeux dès le premier coup d'œil. Le but étant de donner confiance au client dès le départ. Il va sans dire que le site a été pensé en responsive design, et qu'il est donc compatible sur toutes les plateformes web (PC, mobile, tablette).

Pour ce qui est de la partie applicative de la plateforme web, celle-ci a été réalisée entièrement en Javascript. L'enchainement des questions dans le site se faisant par groupe de questions, le Javascript servait essentiellement à vérifier et valider le cas échéant si un formulaire était validé, et donc d'afficher le suivant. Il envoie également une requête au serveur a certains moments clés du formulaire, notamment lorsque l'agriculteur à saisi les coordonnées de ses parcelles, afin de directement lancer le téléchargement d'un package Copernicus correspondant à ces coordonnées.

Dans un premier lieu, une vérification va être effectuée sur si l'utilisateur a correctement saisi ses informations personnelles, par exemple l'absence de chiffres dans son Nom/Prénom, ou encore par des expressions régulières (méthode informatique de validation de chaîne devant correspondre à un format précis) si son numéro de téléphone et son adresse e-mail étaient valides.

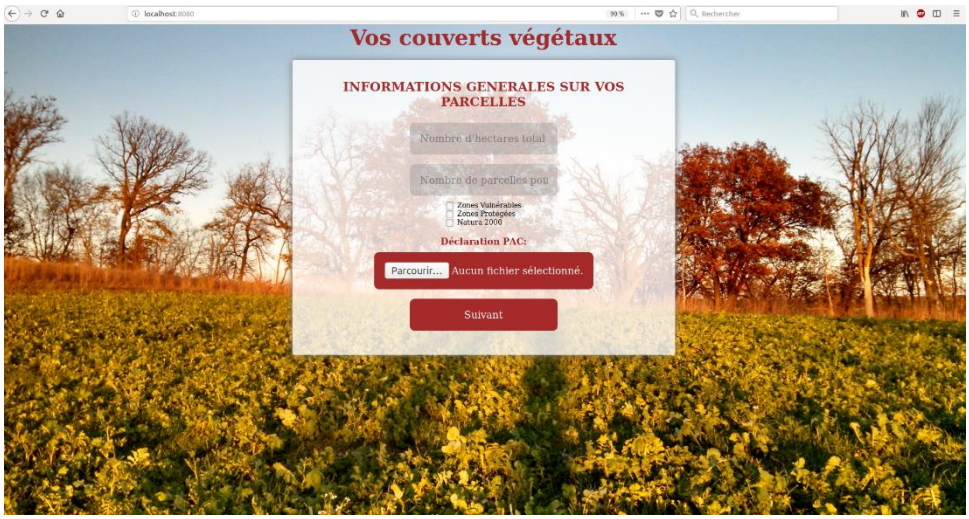
Formulaire de Données Personnelles, avec une image décrivant notre travail à rajouter.

Ensuite, la carte Google Map est affichée. A ce moment, le clic sur la carte est récupéré, afin de récupérer les latitudes et longitudes correspondantes, et d'afficher un marqueur à cet endroit. Au deuxième clic, les données sont de nouveau récupérées et stockées, un deuxième marqueur affiché, la carte reste affichée 1 seconde afin que l'utilisateur puisse visualiser la zone envoyée, puis l'on affiche la troisième partie du formulaire. Dans le même temps, un appel au serveur est effectué pour lui envoyer les informations déjà récupérées et lancer le téléchargement des données de Copernicus correspondantes.



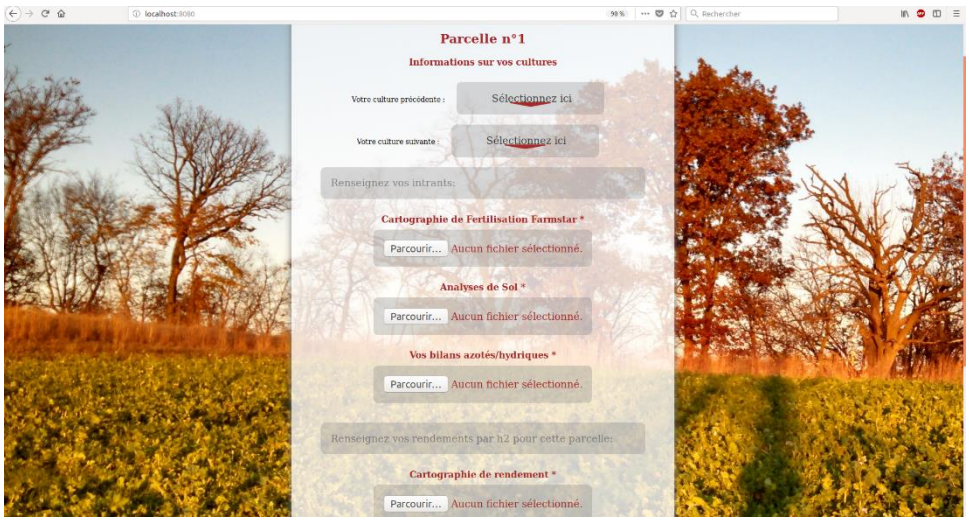
Placement des deux points dans lesquels sont compris les parcelles

Il est alors demandé à l'utilisateur sa surface totale, son nombre de parcelles et un bouton pour envoyer sa déclaration de PAC au serveur est affiché. Une vérification est effectuée pour vérifier que les deux premières données sont bien des nombres, puis un simple clic sur le bouton va afficher un gestionnaire de fichier pour que l'utilisateur indique sa déclaration PAC. Le fichier est alors uploadé sur le site, à la fin de l'upload un appel est lancé au serveur, stockera de son côté le fichier envoyé.

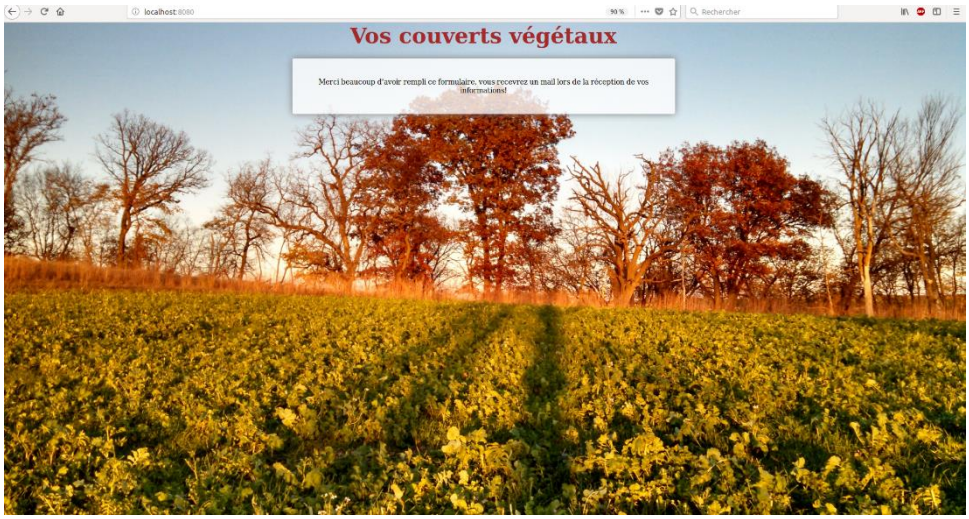


Données sur les terres en général

Enfin, la partie applicative va gérer l'affichage d'un formulaire qui se répète pour chaque parcelle de l'agriculteur, où diverses informations sur ladite parcelle lui seront demandées, ainsi que des documents (les documents étant optionnels, mais les renseigner permet une analyse plus précise pour cette parcelle). Le principe pour ces documents est le même que pour la déclaration PAC à l'étape précédente. Tant qu'il n'est pas à la dernière parcelle, un bouton "suivant" lui est affiché, puis un bouton "terminer" lorsqu'il atteint cette dernière. Un simple texte de remerciement est alors affiché à l'utilisateur.



Données par parcelle



Fin du Formulaire

B - Découpage plus précis

Il restait une dernière étape de traitement d'image à effectuer. En effet, une fois les images superposées pour obtenir le NDVI, par exemple, il faut encore ajuster l'image sur la surface agricole qui correspond. Il faut utiliser, à ce moment-là, les coordonnées qui ont été renseignées par l'agriculteur lors de l'étape avec la carte Google Map.

Après avoir convertis ces coordonnées sous le format UTM (découpage par zones), il est possible de repérer les coordonnées de l'agriculteur sur les différentes images. Le script élaboré permet donc de récupérer, à partir de la zone dessinée par l'agriculteur, l'ensemble des pixels contenus dans l'image. On stocke donc l'information de ces pixels dans une nouvelle image, et une modification des informations qui concernent le pixel en haut à gauche de l'image est effectuée pour conserver le repérage UTM sur cette nouvelle image.

Ce nouveau script a été mixé avec le précédent permettant d'obtenir le simple NDVI. Les deux images sont donc stockées dans le dossier de l'agriculteur, la complète contenant 100km², et la deuxième ne contenant que la partie découpée. A voir plus tard selon la politique et les désirs de la start-up, si l'agriculteur doit avoir accès au complet ou non.

C – Codage du nouveau serveur

Pour penser le nouveau serveur correspondant à la nouvelle plateforme web, il a fallu prendre en compte une première problématique majeure, comment télécharger du côté serveur les fichiers upload par le serveur, et également comment différencier et savoir dans quel dossier les stocker. Deux modules de Node JS ont donc été utilisés : Formidable, pour pouvoir télécharger cette donnée côté serveur, et Express, pour détecter quand un upload a eu lieu et déclencher un évènement serveur. Enfin, il y a une variable wichStep, initialement à 0, qui change une fois que la déclaration PAC est reçue, afin que le serveur sache que les documents suivants seront pour les parcelles. Le problème posé par cette méthode, cependant, est que dès qu'un fichier est upload, le serveur le récupère, ce qui fait qu'une erreur de la part de l'agriculteur est directement enregistrée.

Ensuite il a fallu ajouter la récupération de données passées par message au serveur dans des tableaux, et se servir de ces données. Notamment le trio Nom, Prénom et Numéro de téléphone, qui servent à générer un dossier dans le dossier de stockage, FarmingData, un dossier propre à l'agriculteur, donc le nom est tout simplement de la forme Nom_Prénom_Numéro, ce qui permet de les retrouver rapidement. Dès que les données personnelles et les coordonnées englobant les parcelles sont saisies par l'agriculteur, ces dernières sont envoyées, puis une vérification est effectuée, pour savoir si un dossier à ce nom existe déjà, sinon on le crée.

Dans ce dossier seront stockés la déclaration PAC de l'agriculteur, puis un dossier par parcelle sera créé au cours du formulaire, dossiers dans lesquels seront stockés les documents relatifs à telle ou telle parcelle. Ceci permet également une récupération plus précise et facilité des documents pour l'équipe qui s'occupera de l'interface Administrateur. Ensuite, les images pour les terres de l'agriculteurs de Copernicus sont extraites puis stockées dans ce dossier également, et le NDVI ainsi généré est également stocké dans ce dossier.

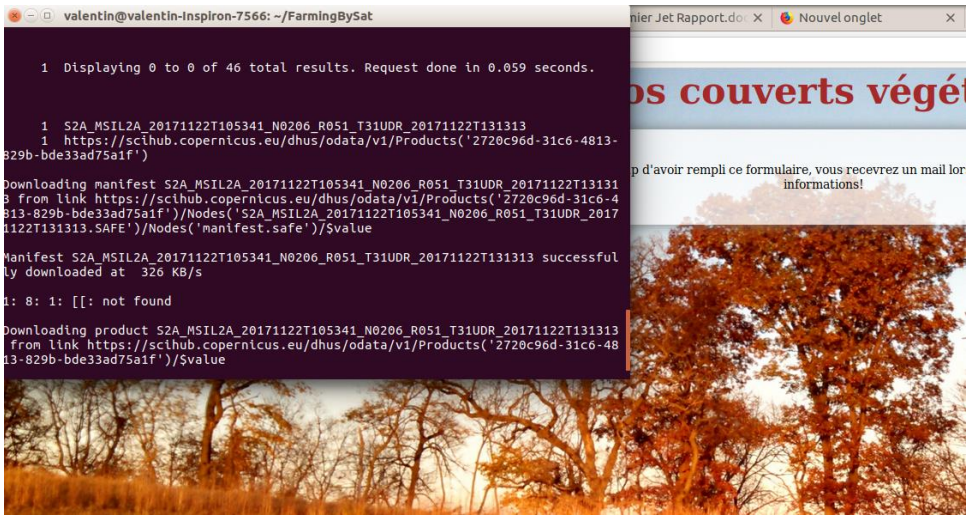
On notera le codage d'une base de données JSON (un format de donnée en javascript) qui n'est actuellement pas utilisée par le serveur en l'état, mais qui sera laissée dans le code GitHub, elle permet de stocker précisément les données utilisateurs à l'intérieur de ce JSON, qui peut ensuite être chargé via du javascript, ce qui permettra d'afficher notamment les données textuelles pour un utilisateur dans la partie interface administrateur.

La communication entre le client et le serveur est donc gérée, comme précédemment, par Socket Io, lorsqu'une étape nécessitant un appel au serveur est validée, la fonction javascript du bouton validant l'étape appelle une fonction permettant d'émettre un socket.emit, élément qui contient l'intitulé d'un message pour que le serveur sache le capter et quelle partie doit le traiter, et un message en lui-même, ici ce sont des tableaux. Le javascript permet de stocker plusieurs types de données différentes dans un seul et même tableau, ce qui permet d'envoyer facilement les données de l'utilisateur.

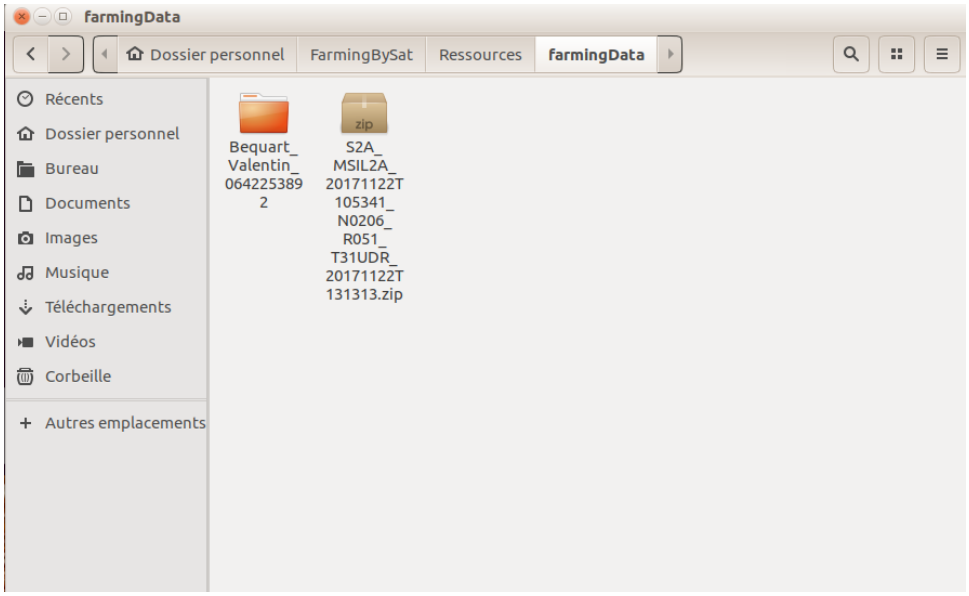
Pour la partie entière du téléchargement des données, avec l'extraction, la suppression et l'appel du script Python, là ou avant tout fonctionnait avec des boutons qui appelaient chaque partie, désormais le tout s'enchaine automatiquement. En effet, dès la réception des coordonnées, la requête de téléchargement est immédiatement lancée, ce qui est un choix de l'équipe, car l'agriculteur pourrait très bien finir par abandonner, mais s'il ne le fait pas, il recevra ses données plus rapidement. Le téléchargement est donc lancé, et dès qu'il se termine, un appel informant le client est lancé, qui va alors lancer la deuxième phase, l'extraction, qui elle-même préviendra le client, et ainsi de suite jusqu'à la dernière étape. Cette architecture est un choix de l'équipe, afin de toujours pouvoir mesurer côté serveur et côté client la fiabilité de ses mesures, mais pourrait tout simplement être changé en une suite de fonction s'appelant elles-mêmes dans le serveur uniquement, pour ne prévenir qu'à la toute fin.

Il faut cependant insister que toute cette architecture de serveur et ce fonctionnement a été pensé et codé par des personnes découvrant cette technologie, ne commençant qu'avec la théorie sur comment fonctionnait un serveur. Le tout est donc largement améliorable et comporte surement quelques failles en l'état actuel des choses, failles provoquées notamment par le manque de temps pour réaliser ce projet compte tenu des défis techniques imposés.

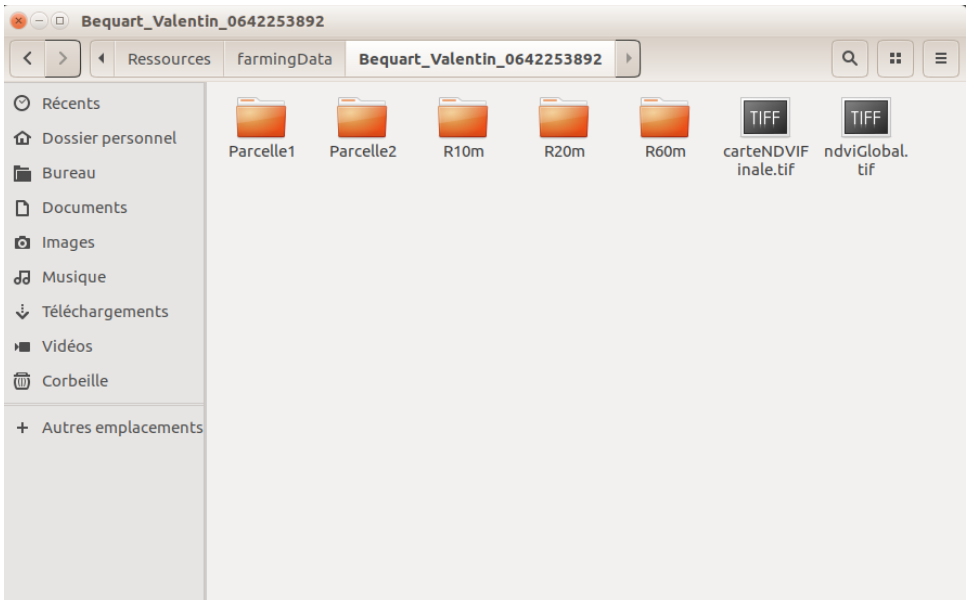
Pour le site internet, la création d'une base de données a permis de gérer les personnes enregistrées sur le site. Cette base de données a été codées en Json, qui est un dérivé du JavaScript, permettant la gestion de celles-ci. Il y a donc un attribut particulier à chaque instance. Une instance étant un inscrit, et un attribut peut être par exemple, son nom ou sa date de naissance. Le choix du Json pour coder la base de données est pensé de façon à ce que le codage soit simple. Dans le cas de la start-up, cela suffit amplement. Si par la suite, cette start-up venait à se concrétiser, il faudrait passer sur un langage plus professionnel, comme MongoDB.



Lancement du téléchargement du paquet nécessaire sur le serveur



Dossier de l'agriculteur créé, et le paquet en cours de téléchargement



Paquet téléchargé, dossiers des parcelles, cartes NDVI terminées, stocké dans le dossier Agriculteur

Partie 3 : Etude sur le carbone

Dans cette partie nous allons essayer de concevoir un script sous python, nous permettant d'obtenir une carte contenant les indices de carbone contenus dans les sols.

A-Détermination des variables utiles

Pour cela nous avons besoin en entrée des réponses spectrales du MSI pour les bandes 2, 3, 4, 8, 10 et 12 :

Tableau 1- Différentes bandes du MSI utilisées pour le calcul des indices

	Utilité	Longueur d'onde (nm)	Résolution (m)
Bande 2	Bleu	490	10
Bande 3	Vert	560	10
Bande 4	Rouge	665	10
Bande 8	NIR	842	10
Bande 10	Cirrus	1375	60
Bande 12	SWIR2	2190	20

Un problème se posait alors, quant à la taille des pixels, et donc la résolution des cartes. Nous avons donc choisi, tout comme le NDVI, que la résolution serait de 10 mètres, et les données proposées par Sentinel-2, nous proposes des résolutions normalisées à 10 mètres pour les bandes 10, et 12.

Afin d'obtenir le résultat attendu, nous avons besoin des variables prédictives suivantes : Greenness Index (GI), Wetness Index (WI), Brilliance Index (BI), Normalized Difference Vegetation Index (NDVI), Vegetation Temperature Condition Index (VTCI), Digital Elevation Model (DEM), Slope in percentage (S%) et le Compound Topographic Index (CTI).

B-Codage du calcul sous python

Commençons par la transformation de Tasselled Cap, qui va nous permettre de trouver BI, WI, GI.

Tableau 1- Coefficients des 3 composantes principales de la transformation de Tasselled Cap *(The IDB project, 2011-2017)*

	Bande 2	Bande 3	Bande 4	Bande 8	Bande 10	Bande 12
Brillance	0,3037	0,2793	0,4743	0,5585	0,5082	0,1863
Humidité	0,1509	0,1973	0,3279	0,3406	-0,7112	-0,4572
Couverture végétale	-0,2848	-0,2435	-0,5436	0,7243	0,0830	-0,1800

La formule de Tasselled Cap est la suivante

⇨ **Formule de la transformation de Tasselled Cap :** (MONDAL, et al., 2017)

$Tas.cap_i = (coeff_1 \times band_2) + (coeff_2 \times band_3) + (coeff_3 \times band_4) + (coeff_4 \times band_8) + (coeff_5 \times band_{10}) + (coeff_6 \times band_{12})$

Maintenant nous pouvons procéder à l'élaboration du script pour ces 3 indices.

Voici ci-dessous, celui de pour le Greenness Index.

Comme on peut le voir, le script est assez court, et seules les variables du tableau de Tasselled Cap changent d'un script à l'autre.

Tableau 1- Script Python pour création du GI

```python
import os,rasterio
import matplotlib.pyplot as plt
import numpy as np
import sys
from scipy import misc

b2 = sys.argv[1]
b3 = sys.argv[2]
b4 = sys.argv[3]
b8 = sys.argv[4]
b10 = sys.argv[5]
b12 = sys.argv[6]
name = "greenness_index"

outfile = r'farmingData/'+name+'.tif'

with rasterio.open(b2) as b2:
    B2 = b2.read()
with rasterio.open(b3) as b3:
    B3 = b3.read()
with rasterio.open(b4) as b4:
    B4 = b4.read()
with rasterio.open(b8) as b8:
    B8 = b8.read()
with rasterio.open(b10) as b10:
    B10 = b10.read()
with rasterio.open(b12) as b12:
    B12 = b12.read()

bi = ((0.1509*B2.astype(float))+(0.1973*B3.astype(float))+(0.3279*B4.astype(float))
      +(0.3406*B8.astype(float))+(-0.7112*B10.astype(float))+(-0.4572*B12.astype(float)))

profile = b4.meta
profile.update(driver = 'GTiff')
profile.update(dtype = rasterio.float32)

with rasterio.open(outfile,'w', **profile) as dst:
    dst.write(bi.astype(rasterio.float32))

monImage = misc.imread(name+".tif")
```

- Importation du module rasterio (le plus important).
- Déclaration des bandes indispensables au calcul des indices.
- Ouverture des images grâce à rasterio.
- Calcul des coefficients.
- Enregistrement de l'indice '-.tiff'

Nous avons donc en notre possession, les 3 indices BI, WI, et GI, ainsi que le NDVI.

C- Conclusion sur l'étude du Carbone

Malheureusement, les indices restants, c'est-à-dire VTCI, DEM, S%, et CTI nécessitent l'utilisation du satellite Sentinel-3, dont l'API est encore en construction. Ceci nous a donc empêché de pouvoir mener à bien l'élaboration du COS (Carbone Organique du Sol) dans les délais demandés. En effet, il aurait été très certainement possible de trouver des relations mathématiques permettant de les calculer, mais ceci aurait demandé un mois de plus.

De ce que nous avons pu constater, l'unique moyen est manuel via l'API Pre Opérationnelle de Sentinelle 3. Qui plus est, on ne télécharge pas des images, mais un ensemble de fichiers nécessaires à la reconstruction de celles-ci via un logiciel fourni par Copernicus. Le désir d'automatisation n'est donc pas respecté pour le moment.

Conclusion sur l'étude de Faisabilité :

Sur le plan technique, l'ensemble du projet représente un challenge certain. Il demande de mettre en œuvre beaucoup de connaissances, divers langages de programmation, et une rigueur de codage suffisante. En effet, selon les sondages effectués par l'équipe marketing, il y aurait une certaine quantité d'agriculteurs intéressés, il faudra donc un serveur robuste, qui sache tenir face aux connections, et surtout avec une capacité de stockage suffisante pour conserver les données, au vu du poids des données imposés par Copernicus.

Ce dernier représente en lui-même une grosse difficulté du projet, il s'est notamment avéré handicapant de par le téléchargement de ces paquets, et des paquets de Sentinel 3, nécessaires à l'élaboration de la carte du Carbone, qui ne peut pas se faire automatiquement via le serveur au vu du format interne de données. Le projet est donc réalisable techniquement, mais il faut prévoir le temps nécessaire à la mise en place d'un serveur robuste, le prix dû au coût de l'hébergement dudit serveur, et à l'emploi de développeurs pour achever le projet et surtout l'entretenir, car il nécessitera de l'entretien.

Enfin, pour la partie Carbone, il ne sera pour le moment pas possible de le faire de manière automatisée, un logiciel étant nécessaire pour reconstruire les images de Copernicus. Il faut donc prévoir quelqu'un pour s'en occuper, le temps que le service s'améliore.

Donc techniquement, ce projet est faisable, cependant son coût sera tout de même assez lourd, donc à voir s'il y a viabilité au vu du business plan.

Bibliographie :

https://scihub.copernicus.eu Par ESA, 2014

https://www.npmjs.com/ Par Isaac Z. Schlueter, dernière version le 04/10/2017

https://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js?status=published Par Mathieu Nebra le 11/08/2017

https://gis.stackexchange.com/questions/238789/bad-results-ndvi-in-rasterio-with-jp2-sentinel-2-bands

Partie 2 :

Ce qui a été effectué en bref.

Première partie :

- Réalisation d'un serveur Node JS basique capable d'utiliser le fichier bash de Copernicus
- Simplification et utilisation du dit fichier bash
- Stockage automatique des données téléchargées, extraction des images et suppression des fichiers restants par le serveur
- Script python combinant les images pour obtenir le NDVI et le stocker
- Exécution de ce script par le serveur

Deuxième partie :

- Conception d'un site de départ permettant d'exploiter les travaux sur le NDVI, en se mettant à la place de l'agriculteur.
- Réalisation du site et de ses différents formulaires.
- Modification du serveur pour récupérer et stocker directement les documents envoyés par l'agriculteur, en fonction de l'étape d'inscription du site.
- Découpage de la zone demandée par l'agriculteur dans le NDVI, à la base créé sur les images de 100km²
- Fusion des deux scripts Python, création et découpage
- Mise à niveau du serveur pour le nouveau script
- Scénario complet d'inscription pour obtenir le NDVI
- Début de codage d'une base de données pour stocker les informations textuelles (Non terminé)
- Début des recherches sur le Carbone.

Dernière partie :

- Analyse des variables nécessaires à l'élaboration du Carbone
- Tentative de connexion à Sentinel 3 de Copernicus, nécessaire pour certaines variables (Non Terminé)
- Recherches de solution alternatives.

Au final, pour ce qui est du Carbone, nous avons réussi à modifier le fichier bash afin de télécharger des données de Sentinel 3. Cependant les extensions de ces fichiers étaient inutilisables en l'état, selon nos dernières recherches, un logiciel serait mis à disposition par Copernicus pour reconstruire des images à partir de ces fichiers. Piste à explorer.

Possibilité de repartir sur ce qui a été fait sur le serveur pour le Sentinel 2 (NDVI) pour exécuter le bash pour Sentinel 3.

Lien pour les codes sources commentés, en cas de question ne pas hésiter à nous contacter :

https://github.com/Weyllan/FarmingBySat

valentin.bequart@isen.yncrea.fr

david.desmyttere@isen.yncrea.fr

antoine.bedhome@isen.yncrea.fr

Partie 3 :

Logiciels nécessaires

Pour exécuter ce projet, un environnement Linux est recommandé, Rasterio n'ayant pas fonctionné sur certains Windows.

Ensuite il faut installer python, et le module python rasterio pour pouvoir traiter les images en jp2, avec un pip install rasterio.

Pour exécuter le serveur, il sera nécessaire d'installer node JS. Tous les packages utilisés sont dans le dossier présent sur GitHub, en cas de problème, ne pas hésiter à nous contacter.

Il suffit alors, pour lancer le serveur, de taper dans une invite de commande : node app.js (bien veiller à se placer dans le dossier contenant app.js.

Enfin, une bonne connexion est à prévoir, les packets de Copernicus sont très lourds, 1.5 Go en moyenne, ce qui peut s'avérer laborieux pour les tests à effectuer.