UCLouvain

École polytechnique de Louvain

Développement d'un prototype d'application permettant de visualiser et structurer les données climatiques historiques et futures issues des modèles régionalisés pour des zones d'études variables au sein de la région des Grands Lacs (Afrique)

Auteurs : **Jimmy de la cruz Mallada**
Promoteur : **Marnik Vancloster**
Lecteurs : **Alice Alonso, Kim Mens**
Année académique 2020–2021
Master [60] en sciences informatiques

Résumé

Le but de ce mémoire est de proposer une technique d'accès au « Big Data » aussi simple et transparente que possible pour extraire et mettre en forme des données relatives aux changements climatiques dans certaines zones sous forte pression.

Il s'inscrit dans le cadre du programme KLIMSEC, en appui de la politique de coopération au développement belge. KLIMSEC est un programme de recherche interdisciplinaire et interuniversitaire qui cible les thèmes du changement climatique et de la sécurité humaine. Il vise le partage des connaissances relatives aux changements climatiques et leurs impacts sur le développement avec les parties prenantes que sont les organisations gouvernementales, les ONG et le secteur privé.

Plus précisément, l'objectif spécifique de ce mémoire est de développer un prototype d'application web permettant de récupérer des données climatiques d'une région spécifique à une très petite maille géographique (segments de 0°44' par 0°44') et de les retranscrire sous forme d'indicateurs prévisionnels de sécurité hydrique (indicateurs de sécheresse, de risques d'inondation, d'évènements climatiques extrêmes, …).

Par rapport aux autres fournisseurs de données climatiques existants, l'application web présentée a pour particularité d'être utilisable à deux échelles, c'est-à-dire, l'échelle locale et régionale.

Les données seront extraites du portail de l'ESGF « Earth System Grid Federation ».

Une fois extraites et classées, le prototype d'application web développé va les retranscrire sous forme de graphes prévisionnels liés à différents scénarios.

L'application est développée sous Django (infrastructure d'application gratuite, open source et bien documentée). Les applications web Django gèrent les données collectées via des objets Python appelés modèles. Les modèles définissent la structure des données stockées, leur type, leur taille maximale, leurs valeurs par défaut et d'autres caractéristiques jugées utiles dans le cadre de ce mémoire comme les options de listes pouvant être sélectionnées ou le texte d'aide pour la documentation.

Parmi l'ensemble des données climatiques disponibles, l'application traite des précipitations et des températures et permet de les visualiser sous trois paramètres : le modèle d'extrapolation choisi, la période de prévision (comprise entre 2070 et 2100) et le scénario RCP (« Representative Concentration Pathway ») retenu parmi les quatre scénarios de trajectoire du forçage radiatif jusqu'à l'horizon 2300. Le forçage radiatif est défini en climatologie comme la différence entre la puissance radiative reçue et la puissance radiative émise par un système climatique donné.

Ces scénarios ont été établis par le Groupe d'experts intergouvernemental sur l'évolution du climat (GIEC) pour son cinquième rapport, AR5 (« IPCC Fifth Assessment Report »).

Plus cette valeur est élevée, plus le système terre-atmosphère gagne en énergie et se réchauffe.

A travers ce document seront expliquées les technologies utilisées, la structure de l'application et l'interprétation des résultats obtenus.

Remerciements

Mes remerciements vont aux personnes ayant participé au bon déroulement de mon mémoire :

Mon promoteur, Marnik Vanclooster, ainsi qu'Alice Alonso pour m'avoir soutenu et aidé lors de mon mémoire.

L'équipe des professeurs de l'Université catholique de Louvain pour la transmission de leurs connaissances.

Aux étudiants, Wail Semlali et Manon Creplet pour l'entraide mutuelle entre étudiants.

Pour finir, ma famille et mes amis pour leur soutien durant ce travail.

Liste des images

Liste de tables

Liste des abréviations

AJAX	JavaScript et XML asynchrones
CCKP	Climate Change Knowledge Portal
CERGACS	Centre européen de recherche et de formation avancée en calcul scientifique
CMIP	Modèles couplés d'inter-comparaison
CNRS-IPSL	Centre national de la recherche scientifique
Cordex	Coordinated Regional Climate Downscaling Experiment
CRU	Climate Research Unit
DGD	Ministère des Affaires étrangères/ Coopération au Développement
ESGF	Earth System Grid Federation
HTTP	Protocole de transfert hypertexte
MCG	Modèle climatique global
MCR	Modèles climatiques régionaux
MVT	Model View Template
NASA	Administration nationale de l'aéronautique et de l'espace (National Aeronautics and Space Administration-USA)
NCAR	Centre national de recherche atmosphérique
NetCDF	Network Common Data Form
NOAA	Agence nationale d'observation océanique et atmosphérique (National Oceanic and Atmospheric Administration-USA)
NSF	Fondation Nationale pour la Science (National Science Foundation-USA).
OPeNDAP	Open-source Project for a Network Data Access Protocol
ORM	Mapping objet-relationnel
SGDB	Système de gestion de base de données
SQL	Langage de requête structurée (Structured Query Language)

Table des matières

1 Introduction

En novembre 2016, l'Accord de Paris suggérait à toutes les parties prenantes de renforcer leur coopération pour améliorer l'adaptation de la société aux enjeux des changements climatiques et réduire l'écart entre la science du climat et les politiques climatiques. Le changement climatique a été identifié comme un défi central par l'Agenda 2030 des Nations Unies pour le développement durable.

La mise en place des programmes et projets relatifs aux conventions politiques internationales nécessitent une bonne compréhension des changements climatiques en cours. Les données climatiques collectées depuis des décennies en constituent la base d'information indispensable. Toutefois, elles sont volumineuses et pas toujours disponibles sous une forme immédiatement exploitable. Il apparaît utile de traduire cette énorme quantité de données au travers d'outils visant à en extraire une information pertinente mise à disposition des personnes, institutions ou instances en charge de la lutte contre le changement climatique.

Analyser des programmes d'adaptation aux enjeux climatiques figure parmi les objectifs de KLIMSEC, programme de recherche interdisciplinaire et interuniversitaire financé par la DGD. Ce programme vise à générer des capacités pour permettre la transition nécessaire vers une société durable grâce à la recherche pour le développement. Il vise aussi le partage des connaissances avec les organisations gouvernementales, les ONG et le secteur privé. Il favorise dans ce cadre la diffusion d'informations et de connaissances techniques relatives aux changements climatiques.

Différentes organisations compilent des informations relatives aux changements climatiques observés et prédits au niveau de régions ou de pays, voire à l'échelle planétaire. Des portails existent pour diffuser ces informations à travers l'internet, mais celles-ci ne sont pas toujours sous le format requis par un utilisateur occasionnel. Plus particulièrement, une difficulté réside dans l'obtention d'informations relatives aux changements climatiques à l'échelle locale.
Des personnes impliquées dans des projets de développement locaux mais n'ayant que des notions limitées d'informatique devraient, selon la vision d'une diffusion de l'information

pertinente et utile, pouvoir accéder à ces informations dans une présentation compréhensible et sous un format exploitable.

Pour y contribuer, l'outil présenté dans ce mémoire vise à faciliter l'accès à l'information relative aux changements climatiques et risques hydrologiques des pays partenaires de la DGD, particulièrement sur le continent africain.

2 Contexte

2.1 Modèles climatiques

2.1.1 Modèles globaux et régionaux

Un modèle climatique global (MCG) est une modélisation mathématique spatio-temporelle du climat de la planète composé de cases, appelées mailles. Il simule les interactions entre l'atmosphère, l'océan et les surfaces continentales. Il fournit des représentations numériques saisonnalisées de la répartition géographique de paramètres, tels que vents, nuages, masses d'eau. Notons au passage la portée symbolique et philosophique de ce modèle qui tend à unifier les quatre éléments (l'eau, l'air, la terre et le feu) dans une relation causale.

Les modèles climatiques régionaux (MCR) sont pilotés par les MCG et appliqués sur une zone limitée. Ils peuvent fournir des informations à des échelles beaucoup plus petites soutenant une évaluation et une planification plus détaillées de l'impact du climat sur des régions du monde

La figure 1 ci-dessous, montre la mise en place d'un modèle de climat régional par un modèle de climat global.

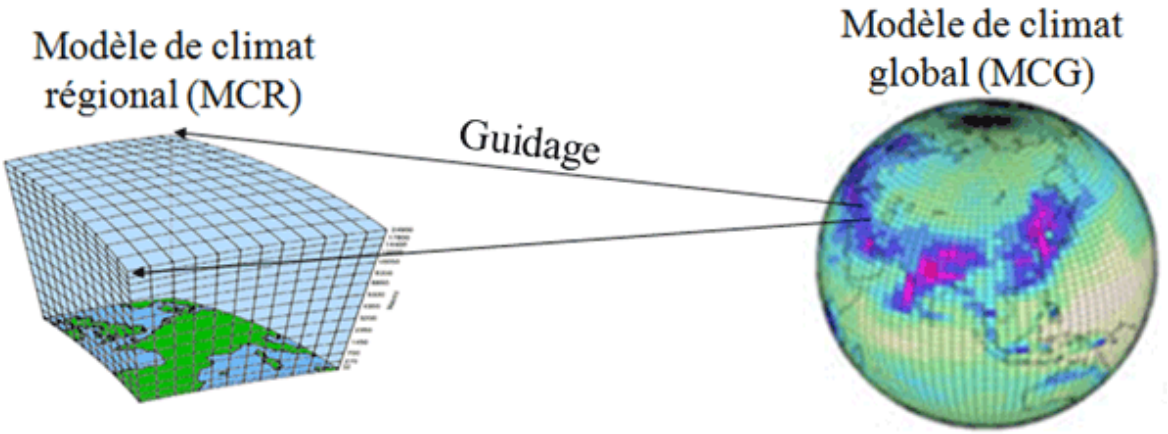

Figure 1: Interaction modèle régionaux et modèle globaux, prise sur le site : https://interstices.info/modeles-globaux-ou-regionaux-comment-zoomer-le-climat/

2.1.2 Scénario RCP

Le scenario RCP permet de modéliser le climat futur sur la base d'une hypothèse de travail concernant la quantité de gaz à effet de serre émise dans les années à venir (période 2000-2100), et donc selon un scénario de développement socio-économique.
Les quatre scénarios sont nommés d'après la gamme de forçage radiatif ainsi obtenue pour l'année 2100 : le scénario RCP2.6 correspond à un forçage de +2,6 W/m2, le scénario RCP4.5 à +4,5 W/m2 et de même pour les scénarios RCP6 et RCP8.5.

2.1.3 Cordex

Coordinated Regional Climate Downscaling Experiment (Cordex) a été lance en 2009. Ses objectifs sont de comprendre les phénomènes climatiques régionaux et locaux, d'évaluer et d'améliorer les modèles et les techniques de réduction d'échelle climatique, produire des ensembles de projections régionales à échelle réduite et de favoriser la communication et l'échange de connaissances avec les utilisateurs d'informations climatiques régionales.
Les projections climatiques de Cordex se trouvent sur le portail de l'Earth System Grid Federation (ESGF) qui contient la plupart des recherches mondiales sur le changement climatique. Il est reconnu comme la principale infrastructure pour la gestion et l'accès à de grands volumes de données distribuées pour la recherche sur le changement climatique.

2.2 Plateformes existantes

2.2.1 World Bank Climate Change Knowledge Portal

Site : https://climateknowledgeportal.worldbank.org/country/burundi/climate-data-projections

La plateforme web World Bank Climate Change Knowledge Portal (CCKP), réalise le même objectif que le prototype développé dans ce mémoire.

Cependant, les modèles utilisés pour la visualisation des données sont différents. Cette application utilise des modèles climatiques globaux (MCG avec une résolution spatiale approximative de 200 km), alors que dans ce mémoire les modèles utilisés sont des modèles régionalisés (MCR, résolution spatiale approximative de 12 à 50 km).

Le CCKP utilise les données issues des GCL CMIP5 (Modèle couplé d'inter-comparaison de projets) et NCAR (centre national de recherche atmosphérique) utilisées pour le traitement de données climatiques sur un espacement grille de résolution 1.0°x1.0°.

Le CCKP présente les projections climatiques futures extrapolées des moyennes des données historiques du CRU qui, lui, travaille sur un espacement de résolution de grille plus fin, de 0.5°x0.5°.

Points forts :

- Lisibilité de la plateforme
- Comparaison entre observations historiques et projections futures
- Possibilité de télécharger les différents graphes

Points faibles :

- Utilisation de modèles globaux utilisant un espacement de grille de 1.0x1.0 au niveau mondial par interpolation linéaire.
- Impossible de sélectionner une zone ou une région. On y retrouve une vue globale ou d'un point du pays sélectionné sans possibilité d'affichage sur la base de coordonnées géographiques. Par exemple sera affiché le Burundi, ainsi qu'un point de ce pays tel que « Bujumbura » mais il ne sera pas possible de faire une délimitation par quatre points d'une zone de la ville de Bujumbura.

Le programme développé pour ce mémoire vise à une plus grande précision en utilisant un espacement de grille de 0.44°x 0.44°. Ceci devrait permettre une analyse climatique plus

ciblée au niveau d'une région ainsi qu'une limitation spécifique plus fine dans une région par l'introduction de latitudes et de longitudes.

2.2.2 Climate impact portal

Site : https://climate4impact.eu/impactportal/general/index.jsp

La plateforme climate4impact est le résultat d'une collaboration au sein du projet européen. Ses objectifs sont d'améliorer l'utilisation des données de recherche, de soutenir d'autres portails climatiques et de favoriser la collaboration entre les groupes de modélisation pour accélérer le développement et l'utilisation de modèles du système climatique complexe de la Terre.

Cette plateforme propose des interfaces web pour rechercher, visualiser, analyser, traiter et télécharger des ensembles de données. Elle utilise des données provenant de l'ESGF.

Elle est soutenue par plusieurs organismes : Centre européen de recherche et de formation avancée en calcul scientifique (CERGACS), Centre national de la recherche scientifique (CNRS-IPSL), etc … et a reçu un financement du programme de recherche et d'innovation de l'union européenne Horizon 2020.

Points forts :

- Calculs de nombreux indicateurs climatiques
- Statistiques par pays
- Téléchargement des ensembles de données climatiques

Points faibles :

- Obligation de créer un compte ESGF pour le traitement des données
- Pas de projection climatique, à l'inverse du programme développé pour ce mémoire qui utilise des projections de données climatiques sur la période 2070-2100. Les fonctionnalités restent néanmoins similaires : visualiser et traiter.

2.3 Objectifs

L'objectif spécifique est de développer pour les régions du Sud (comprendre l'Afrique) où la coopération au développement belge est active, une application informatique permettant de :

- Collecter des données climatiques (précipitations, température, …) à haute résolution spatio-temporelle (maille d'observation fine et fréquence élevée) sur une longue durée pour certaines régions-cibles ;
- Transformer ces informations en indicateurs de risque hydrologique et climatique pertinents, calculés selon le scénario RCP retenu ;
- Visualiser ces données climatiques et ces indicateurs de risque ;
- Pouvoir exporter ces données (observations et résultats de calculs) vers des outils à vocation pédagogique ou de vulgarisation.

3 Méthodologie

Le contenu de ce chapitre va permettre de comprendre le prototype développé et le fonctionnement du projet.
La première étape est celle des choix technologiques, langage et logiciels de développement utilisés.
La deuxième étape est celle de l'explication de la plateforme ESGF, de ses données et des données historique provenant du CRU.
Suivront une visualisation de l'interaction entre le serveur et le navigateur du client à travers l'architecture du prototype.
Finalement, l'analyse de certains points d'implémentation pour comprendre les fonctionnalités utilisées sur le site web.

3.1 Choix technologiques

3.1.1 Langage informatique

Le choix du langage doit satisfaire à plusieurs conditions :

- L'application doit pouvoir être réutilisée et maintenue par différents développeurs, pour certains néophytes. Il est donc important de choisir un langage facile à apprendre et à comprendre.
- Il est nécessaire d'utiliser un langage ni trop récent (de fait insuffisamment éprouvé et encore potentiellement porteur d'erreurs, bugs ou incohérences) et ni trop ancien (obsolète, moins performant ou simplement plus maintenu).
- Le langage doit avoir un cadre de travail structuré et explicite (aussi appelé infrastructure logicielle ou Framework).

Le langage Python les respecte toutes. Python a été conçu par Guido van Rossum en 1991. Il bénéficie d'une bonne documentation et d'une communauté active. C'est un langage interprété ce qui signifie qu'il n'y a pas de compilation avant l'exécution. Il est utilisable sous différents systèmes d'exploitation (Linux, Mac, Windows).
Il permet au programmeur de voir rapidement les résultats d'un changement dans le code. Python est facile à apprendre et à utiliser ce qui permet la création des programmes rapidement et avec peu d'effort.
Sa syntaxe est conçue pour être lisible et directe.

Selon les sites « développez.com » *(https://www.developpez.com/actu/133538/)* et « bigdata.fr » *(https://www.lebigdata.fr/python-langage-definition)*, Python est le langage le plus utilisé dans le domaine scientifique.

3.1.2 Infrastructure logicielle (Framework web)

Un 'Framework' est littéralement un cadre de travail. Il est pris ici dans son acception d'infrastructure logicielle, servant à créer et à modéliser l'architecture d'une application. Il fournit une bibliothèque de composants logiciels structurels (aussi appelés «briques») qui créent les grandes lignes du logiciel. L'un de ses avantages est la réutilisation d'instructions déjà existantes et la formalisation simultanée et automatique d'une architecture nécessaire à la maintenance de l'application.
Les Frameworks peuvent être ventilés en deux types : « micros » et « full stack ».
Le premier type est adapté aux petits développements (une à deux pages de code).
Le second permet le développement d'applications plus volumineuses et complexes.
De par la taille du projet envisagé, un Framework de type « full stack » est nécessaire.
Il doit de plus être compatible avec le langage Python et plus particulièrement avec sa version la plus récente (version 3).
Les éléments pris en compte pour le choix du Framework se résument comme suit :

- Framework ouvert et évolutif, régulièrement mis à jour et enrichi ;
- Open source, accessible à une large communauté en cas de bug et en amélioration permanente ;
- Comportant une bonne documentation.

Compte tenu de ces éléments, trois Frameworks du web Python sont candidats : Pyramid, Web2py, et Django.

Les avantages et inconvénients de chaque Framework sont détaillés ci-dessous (Tableau.1).

Framework	Avantage	Inconvénient
Pyramid	• Flexible et personnalisable • Open-source et gratuit	• Complexe à prendre en main si besoin de personnaliser • Prend du temps pour la configuration
Web2py	• Evolutif • Puissant pour les traitements de données en base de données • Fonctionne avec de nombreux environnements • Communauté active	• Ne fonctionne pas en Python 3 • Assez complexe à comprendre
Django	• Fonctionnalités de base • Constante évolution • Beaucoup de plugins existants (logiciel appelé par un autre logiciel) • Interface administrateur pour gérer directement la base de données • Plus connu • Optimisé pour les projets de grande ampleur • Structure MVT (Model View Template) • Permet la création de modèle de base de données • Open-source et gratuit	• Pas de création d'application en temps réel • Impose un choix de fonctionnalités au développeur, telles que le langage de la base de données (SQL)

Tableau 1: Avantages et inconvénients des trois Frameworks retenus.

Pyramid

Pyramid est un framework récent (première version développée en 2011), bien documenté, riche et précis. Un de ses caractéristiques est qu'il admet tout type de structure de programmation. Cette absence de contrainte dans l'organisation logique de la programmation peut être vue comme un avantage mais risque de rendre compliquée sa maintenance ou sa reprise par un autre programmeur que son créateur.

Cette totale liberté rend Pyramid incompatible avec la première condition du choix du langage : « L'application doit pouvoir être réutilisée et maintenue par différents développeurs, pour certains néophytes. Il est donc important de choisir un langage facile à apprendre et à comprendre ». C'est la raison de son éviction.

Web2py

Ce Framework est conçu pour le développement d'applications sécurisées basées sur des bases de données. N'étant pas compatible avec la nouvelle version de Python (version 3), ce candidat ne peut être choisi.

Django

La première version fut développée en 2003, publié sous licence BSD en 2005. Il est utilisé par un certain nombre d'applications connues comme Instagram, Spotify ou le navigateur Internet Mozilla.

Django est un Framework organisé en Model/View/Template (MVT) structure d'architecture composée de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs. L'objectif principal est de dissocier le traitement, les données et leur présentation tout en définissant les interactions entre eux.

Django permet une gestion complète d'une base de données, point important pour ce projet. Parmi ses points forts, figurent la mise en place de fonctions de sécurité contre le Cross Site Scripting ou les injections SQL qui sont deux types d'attaques souvent utilisés par les hackeurs. De plus, sa grande communauté d'utilisateurs lui assure une maintenance ainsi qu'une documentation à jour. Il suit la philosophie « Battery-included », c'est-à-dire qu'il fournit dans son environnement natif la majorité des éléments nécessaires aux développements applicatifs.

Du fait du potentiel d'évolution du projet et du nombre de développeurs qui y participeraient, Django apparait comme le Framework le mieux adapté aux besoins.

Il faut néanmoins composer avec deux inconvénients importants et adapter l'utilisation de l'outil à ceux-ci :

Premièrement, Django ne reconnait pas AJAX (Asynchronous JavaScript and XML) nativement. AJAX est un outil permettant la mise à jour du contenu d'une page Web d'une manière rapide et sans chargement complet de celle-ci. Une adaptation particulière sous Django est nécessaire et expliquée dans la partie implémentation.
Deuxièmement, Django ne prend pas en charge les bases de données NoSQL. Sa documentation porte uniquement sur quelques bases de données SQL : PostgreSQL, MySQL (la plus courante dans les sites web), SQLite (pour de petites bases de données) et Oracle (base de données payante). Cette limitation impose donc le choix additionnel d'un SGDB, développé dans le paragraphe suivant.

3.1.3 Base de données utilisée par le serveur web

Le choix du Framework Django réduit les bases de données qui peuvent être utilisées aux seules bases SQL. Certaines d'entre elles ont été rejetées, Oracle car payante et SQLite car destinée aux petites bases de données. Restent PostgreSQL et MySQL. Leur implémentation dans Django est identique et elles supportent toutes deux l'insertion de fichiers de type JSON, point important du projet.
Selon le site DB-Engines Ranking (*https://db-engines.com/en/ranking*), MySQL est mieux classé que PostgreSQL (rang 2 sur 371, constant sur 12 mois avec un score de 1236 versus rang 4 sur 371, constant sur 12 mois avec un score de 559).

MySQL est donc retenu dans le cadre de ce mémoire

3.1.4 Outil de visualisation de carte utilisé

La cartographie est un point essentiel de cette application, à la fois pour localiser les données brutes et pour en visualiser les résultats. Il existe beaucoup d'outils de géopositionnement ou de géolocalisation, les plus connus étant Google Maps, Google Earth, Mapbox, Maplace, MapQuest et Leaflet.
Leur modèle économique est semblable : gratuité pour une utilisation limitée mais accès payant à partir d'un nombre donné de requêtes. Par exemple Google Maps « offre » $200 de budget gratuit aux développeurs web, ce qui équivaut à un mix mensuel de 28.000 requêtes/cartes, Mapbox est payant (5$ toutes les 1.000 requêtes) au-delà de 50.000 requêtes chargées (chargement de la carte sur l'application web).
Le choix pour la première version de ce prototype est l'outil Leaflet (dont la première version fut créée en 2011 par Vladimir Agafonkin). Les éléments à l'appui de ce choix sont, par ordre d'importance :

- Leaflet se trouve dans le top 10 des meilleures cartes géographiques en open source ;
- Leaflet permet l'intégration de cartes interactives ;
- Leaflet n'impose aucune restriction que ce soit en volume, en fréquence ou en zones géographiques couvertes ;
- Les mises à jour de Leaflet sont faites par la communauté des utilisateurs à partir des données satellitaires et d'organismes tels que le cadastre français, le bureau de recensement des Etats-Unis, ….
- Leaflet donne accès à un grand nombre de tutoriels et de documentations accessibles sur Internet.
- Leaflet est un des rares outils de cartographie encore gratuits à ce jour.

3.1.5 Outils de développement utilisés

GitHub est un service web d'hébergement et de développement de logiciels, utilisant le module de gestion de version Git. Pour le projet objet de ce mémoire, n'est utilisé que le service GitHub de copie du code-source sur un ordinateur distant. Ce service est gratuit pour un projet privé si sa taille ne dépasse pas 500 MBytes et s'il consomme moins de 2.000 minutes de temps d'exécution par mois pour la compilation et l'automatisation des tests. GitHub aura un coût pour les autres utilisateurs variant de :

- 4$ pour 3.000 minutes (50 heures de connexion) par mois et 2 GigaBytes de stockage
- 21$ pour 50.000 minutes (830 heures) par mois et 50 GigaBytes de stockage.

GitHub apporte une solution à deux des problèmes de l'environnement de développement :

- Il permet de garder une copie sur un ordinateur distant ;
- Il permet de partager le code avec les éventuels développeurs désirant participer à l'évolution de ce programme.

Panoply est une application multiplateforme (Windows, Mac et Linux). Elle est disponible en open source donc gratuite et en accès libre. Elle permet une lecture rapide ainsi que la visualisation des fichiers NetCDF. Un fichier NetCDF est une structure décrivant un ensemble de données représentée par des tableaux multi-dimensionnels.
Une contrainte de Panoply est d'imposer la version la plus récente de Java (supérieure ou égale à JAVA 9).

3.2. Projection climatique et source de données historiques

3.2.1 ESGF

La base de données décentralisée utilisée pour le traitement des données de la science du climat se nomme « Earth System Grid Federation » (ESGF), fondée à l'initiative du ministère de l'énergie des Etats-Unis. Les objectifs de l'ESGF sont de faciliter les progrès de la science du système terrestre et de favoriser la collaboration entre les différentes agences, standardiser et centraliser des données de modèle générés par différents centres de recherche à travers le monde, créer des infrastructures logicielles et des outils qui facilitent les avancées scientifiques.

Cette plateforme compile plus de 5 PetaBytes de données provenant de plus de 25 projets et de 70 modèles couplés d'inter-comparaison (CMIP). Elle sert à plus de 25.000 utilisateurs en fournissant plus de 700.000 ensembles de données aux laboratoires et universités dont sont extraits les protocoles permettant les évaluations périodiques effectuées par le GIEC. Elle est le fruit d'une collaboration internationale et interinstitutionnelle dirigée par le ministère de l'énergie des Etats-Unis et cofinancée par différentes institutions américaines et européennes dont la NASA, la NOAA ou la NSF.

Elle fonctionne selon un système « Peer To Peer » d'échange en réseau où chaque entité est à la fois client et serveur. Les nœuds de ce réseau sont répartis sur tous les continents pour permettre une fluidité dans le traitement des requêtes tout en optimisant la puissance de calcul. (Fig.2)

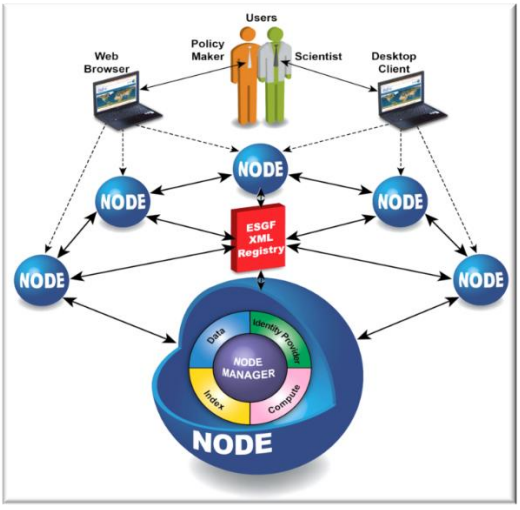

Figure 2: Structure de l'ESGF, prise sur le site : https://esgf.llnl.gov/mission.html , consulté le 16/05/2021

Cette organisation en réseau permet de fournir un accès permanent aux données, tout nœud non fonctionnel étant temporairement désactivé et ses ressources pilotées d'un autre point.

Actuellement, il existe 9 nœuds, 4 aux USA, 4 en Europe et 1 en Australie, comme indiqué en Figure.3.

Major ESGF Node Sites

Institution	Gateway URL	Version	Country	Project(s)	Contact
1 CEDA	esgf-index1.ceda.ac.uk	2.4.0	U.K.	CMIP5, CORDEX, Obs4MIPs, SPECS, ESA CCI, EUCLEIA, CLIPC	alan.iwi@stfc.ac.uk
2 DKRZ	esgf-data.dkrz.de	2.4.0	Germany	CMIP5, CORDEX, Obs4MIPs, ISI-MIP	berger@dkrz.de
3 ANU NCI	esgf.nci.org.au	2.4.0	Australia	CMIP5	ben.evans@anu.edu.au
4 NOAA GFDL	esgdata.gfdl.noaa.gov	2.4.0	U.S.	CMIP5, ncpp2013, Obs4MIPs	hans.vahlenkamp@noaa.gov
5 NASA GSFC	esgf.nccs.nasa.gov	2.4.0	U.S.	CMIP5, Obs4MIPs, Ana4MIPs, NEX-GDDP, NEX-DCP30, CREATE-IP	daniel.q.duffy@nasa.gov
6 IPSL	esgf-node.ipsl.upmc.fr	2.4.0	France	CMIP5, CORDEX, Obs4MIPs	sebastien.denvil@ipsl.jussieu.fr
7 NASA JPL	esgf-node.jpl.nasa.gov	2.4.0	U.S.	Obs4MIPs, GASS-YoTC, CMAC	luca.cinquini@jpl.nasa.gov
8 DOE LLNL	esgf-node.llnl.gov	2.4.0	U.S.	CMIP5, CMIP3, input4MIPS, ACME	sasha@llnl.gov
9 LiU	esg-dn1.nsc.liu.se	2.4.0	Sweden	CMIP5, CORDEX, SPECS, CLIPC	pchengi@nsc.liu.se

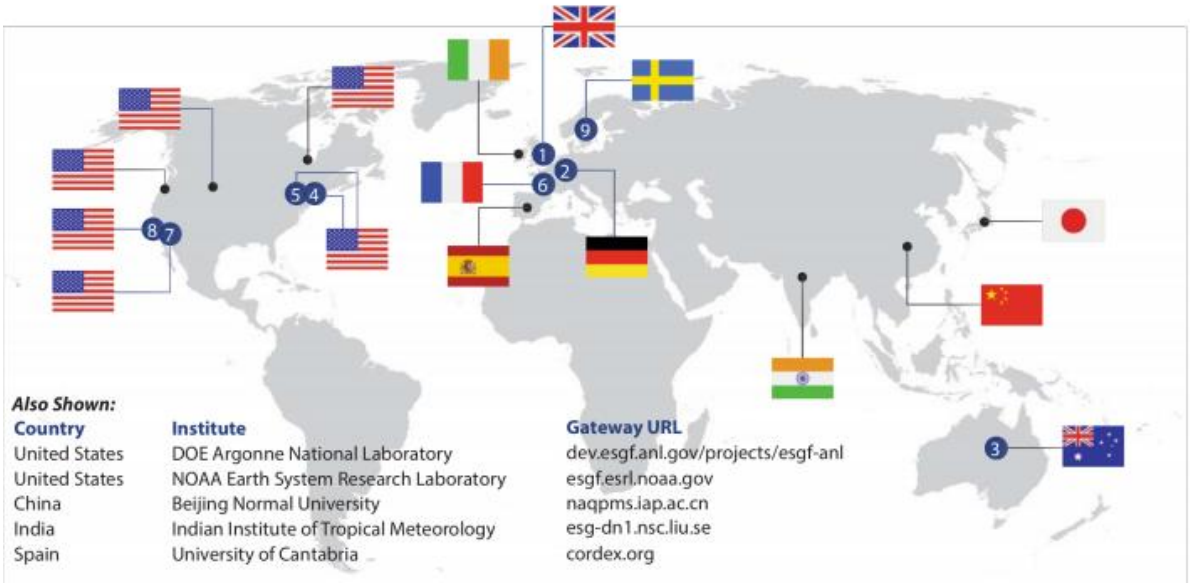

Figure 3: Principaux nœuds dans le monde, prise dans la brochure : https://esgf.llnl.gov/esgf-media/pdf/2017-ESGF-Brochure.pdf , consulté le 16/05/2021

3.2.2 Recherche et extraction des données

L'extraction des données climatologiques est un point de passage obligé. Grace à cette plateforme, la recherche et la récupération des données peuvent se faire de différentes manières :

- Via le site web de l'ESGF (Portail ESGF)
- Via Synda (outil de synchronisation et de téléchargement de données)
- Via le package Python

Les deux premières manières imposent la conservation des fichiers extraits sur un serveur externe dédié. L'application développée travaillant sur un grand volume de données, un espace mémoire considérable serait nécessaire sur un serveur de ce type.
Or le but n'est pas de dupliquer les données ESGF mais de les utiliser dans le cadre d'une procédure automatisée de connexion pour répondre à une requête utilisateur.

Dans ce contexte, une connexion instantanée définissant une recherche sous contraintes puis extrayant les données nécessaires au calcul sera privilégiée.
Une contrainte est un axe (série organisée de données brutes selon une relation d'ordre partiel ou total) sur lequel peuvent être appliqués des sélections, des tris, des analyses et des restitutions.

L'ensemble des contraintes sont :

- Le projet (Cordex)
- Le domaine (Afrique)
- L'ensemble (r1i1p1)
- La variable
- Le modèle climatique
- La fréquence de temps (heures, jours, mois,…) (le mois pour les précipitations et les jours pour les températures)
- Le scénario

Notons que toutes les contraintes listées ci-dessus seront identiques dans toutes les recherches ESGF du mémoire.
Sur cet ensemble seules trois contraintes sont retenues dans ce mémoire : le modèle climatique, la variable et le scénario RCP.

Les définitions qui suivent sont inspirées pour chacune d'entre elles des sites de futura-sciences (*https://www.futura-sciences.com/planete/definitions/climatologie-modele-climatique-12896/ , consulté le 16 mai 2021*) et Ouranos (*https://www.ouranos.ca/publication-scientifique/GuideScenarios2017_FR.pdf, consulté le 16 mai 2021*)

Le modèle climatique est une modélisation mathématique spatio-temporelle du climat dans une zone géographique donnée. Il simule les interactions entre l'atmosphère, l'océan et les surfaces continentales. Il fournit des représentations numériques saisonnalisées de la répartition géographique de paramètres, tels que vents, nuages, masses d'eau.

La variable est un facteur météorologique élémentaire (censé être orthogonal aux autres mais qui interagit avec eux) comme la température, les précipitations, la pression atmosphérique, le sens et l'intensité des vents ...
Le scenario RCP permet de modéliser le climat futur sur la base d'une hypothèse de travail concernant la quantité de gaz à effet de serre émise dans les années à venir (période 2000-2100).
Les contraintes permettent de définir la recherche et donc de qualifier son résultat.

Le domaine géographique et l'horizon temporel sont des données et non des contraintes à ce stade de développement de l'outil, objet de ce mémoire. Elles pourront être appliquées ultérieurement sur les résultats à la suite d'une recherche ESGF complémentaire.

Dans le cadre de ce mémoire, la consultation des données s'effectue via OPeNDAP (*Open-source Project for a Network Data Access Protocol*), qui est à la fois le nom du protocole et le nom de la société à but non lucratif qui l'a conçu.

OPeNDAP est largement utilisé dans le domaine des sciences de la Terre. Il permet d'accéder aux données stockées sur un serveur distant identifié par son adresse URL (*Uniform Resource Locator*), directement utilisée par Python.

La ressource URL est sollicitée dans un mode R/O (*Read Only*) car seule la consultation des données est utilisée dans le cadre de ce mémoire dont l'objectif est leur visualisation et non leur duplication ou leur transfert.

3.2.3 Données provenant de L'ESGF

L'ESGF fournit des fichiers contenant des valeurs climatiques estimées sur base de données historiques. Les données provenant du CRU sont des données historiques observées. Ces données sont utilisées alternativement en fonction de la requête.

Une part importante dans le développement du projet a été consacrée à l'analyse et à la compréhension des données obtenues lors de l'extraction des fichiers en NetCDF (Network Common Data Form) du site de l'ESGF. (Annexes 1)

Ce format de fichier permet de stocker des tableaux de données numériques sous forme de matrice multidimensionnelle.

Il se compose obligatoirement de dimensions (Fig. 4), de variables (Fig.5) et de « global attributes » (Fig.6), détaillés dans l'exemple réel présenté ci-dessous.

Dimensions

Les dimensions définissent la structure du fichier. Chaque dimension est composée d'un nom et d'une longueur.

```
netcdf file:pr_AFR-44_MPI-M-MPI-ESM-LR_rcp85_r1i1p1_CLMcom-CCLM4-8-17_v1_mon_203101-204012.nc {
  dimensions:
    time = UNLIMITED;   // (120 currently)
    rlat = 201;
    rlon = 194;
    bnds = 2;
    vertices = 4;
```

Figure 4: NetCDF dimension

L'organisation majeure du fichier NetCDF pris pour exemple (premier axe) est le temps. Seule la première dimension peut avoir une longueur UNLIMITED. La dimension « Time » peut donc être infinie, ce fichier pouvant compiler sans limite des données chronologiques. En commentaire figure sa taille actuelle, 120 occurrences mensuelles soit 10 ans.

Les axes mineurs sont latitude et longitude présentés dans des rectangles (forme géométrique à 4 sommets « vertices » donc sans altitude) avec 2 valeurs-limites (bnds), s'agissant de coordonnées après rotation de l'axe polaire : segment de latitude « Rlat » (latitude in rotated pole grid) et segment de longitude « Rlon » (longitude in rotated pole grid).

Variables

Les variables définissent la structure des tableaux multidimensionnels de données :

```
float lat(rlat=201, rlon=194);
  :standard_name = "latitude";
  :long_name = "latitude coordinate";
  :units = "degrees_north";
  :bounds = "lat_vertices";
  :_ChunkSizes = 201U, 194U; // uint

float lon(rlat=201, rlon=194);
  :standard_name = "longitude";
  :long_name = "longitude coordinate";
  :units = "degrees_east";
  :bounds = "lon_vertices";
  :_ChunkSizes = 201U, 194U; // uint

float lat_vertices(rlat=201, rlon=194, vertices=4);
  :units = "degrees_north";
  :_ChunkSizes = 201U, 194U, 4U; // uint

float lon_vertices(rlat=201, rlon=194, vertices=4);
  :units = "degrees_east";
  :_ChunkSizes = 201U, 194U, 4U; // uint
```

Figure 5: NetCDF Variables

Il s'agit ici de tableaux bi-dimensionnels stockant en abscisse la longitude et en ordonnées la latitude. Ces deux axes sont exprimés en degrés. Les chiffres accompagnant « rlat et rlon », représentent la taille des tableaux de données, respectivement (201x 0.44 degrés) et (194x 0.44 degrés).

Global attributes

Ces « Global attributes » sont rattachées, comme leur nom l'indique, au fichier lui-même et pas aux données qu'il contient.

```
// global attributes:
:institution = "Climate Limited-area Modelling Community (CLM-Community)";
:institute_id = "CLMcom";
:experiment_id = "rcp85";
:source = "CLMcom-CCLM4-8-17";
:model_id = "CLMcom-CCLM4-8-17";
:contact = "cordex-cclm@dkrz.de";
```

Figure 6: NetCDF Global attributes

Les « Global attributes » fournissent ici la source des données, le modèle sous lequel elles sont élaborées et les coordonnées de contact.

3.2.4 CRU

Le Climat Research Unit (CRU) est une institution créée dans le cadre de l'École des Sciences de l'Environnement de l'université de Liège en 1972 dont le but est l'étude des changements climatiques naturels. Son site web permet le partage de différentes données sous forme de fichiers « .dat ». Ce type de fichier peut contenir des données sous une forme texte ou binaire.

Exemple de fichier (Tableau.2) pouvant être récupéré via leur site : Historique des précipitations mensuelles exprimées en mm, couvrant la période de 1900 à 1909 pour la zone du sud de l'Afrique avec une résolution de 2.5 degrés de latitude (300 km) et 3.75 degrés de longitude (375 km),

2070-3500	1875	68SOUTH AFRICA	6	87%100% 62%	19001998	2070
1900	251	333	214	455	491	264	862	876	245	563	210	417	5181
1901	721	590	142	178	1254	250	744	195	741	247	569	51	5682
1902	160	610	371	334	686	966	839	1091	1455	817	178	123	7630
1903	294	72	248	380	670	1273	458	525	408	1241	618	97	6284
1904	103	408	149	913	510	845	491	1037	625	750	185	298	6314
1905	70	525	348	364	802	1742	422	631	549	506	108	77	6144
1906	43	45	212	605	657	554	354	571	360	689	164	1287	5541
1907	241	65	546	398	1191	363	278	289	515	450	147	353	4836
1908	224	92	149	1222	188	1288	801	602	335	570	304	187	5962
1909	174	122	564	172	448	328	481	1147	375	572	207	704	5294

Tableau 2: Extrait de l'historique des précipitations

A la première ligne on trouve :

Le 1 champ (7 caractères) : 2070, numéro de la parcelle.

Le 2 champ (5 caractères) : -3500, latitude du centre de la parcelle exprimée en décimale, soit -35° (0° Equateur/ -90° Pôle Sud).

Le 3 champ (5 caractères) : 1875, longitude du centre de la parcelle exprimée en décimale, soit 18.75° (0° Greenwich/180° Est).

Le 4 champ (5 caractères): 68, altitude moyenne en mètres.

Le 5 champ (15 caractères) : South Africa, nom du pays dominant dans la parcelle.

Le 6 champ (4 caractères): 6, nombre de stations d'observation dans la parcelle.

Le 7 champ (14 caractères) : 87%, 100%, 62%, informations sur la technique de quadrillage.

Le 8 champ (8 caractères) :1900-1998, valeurs extrêmes de la séquence chronologique

Le 9 champ (7 caractères) : 2070, rappel du champ 1

Le 10 champ (9 caractères) : zone réservée

La ligne 2 et les suivantes sont structurées ainsi :

La colonne 1 correspond à l'année ;

Les colonnes 2 à 13 correspondent aux précipitations mensuelles ;

La dernière colonne correspond à la somme annuelle.

Pour obtenir les précipitations en millimètres il faut diviser les valeurs mentionnées par 10.

3.3 Architecture

Dans ce chapitre seront expliqués le fonctionnement global de l'application développée et l'architecture utilisée par Django « Modèle Vue Template »

L'application est divisée en 4 parties (fig. 7)

- Utilisateurs (Administrateur et Utilisateur)
- Application
- Base de données
- Cache

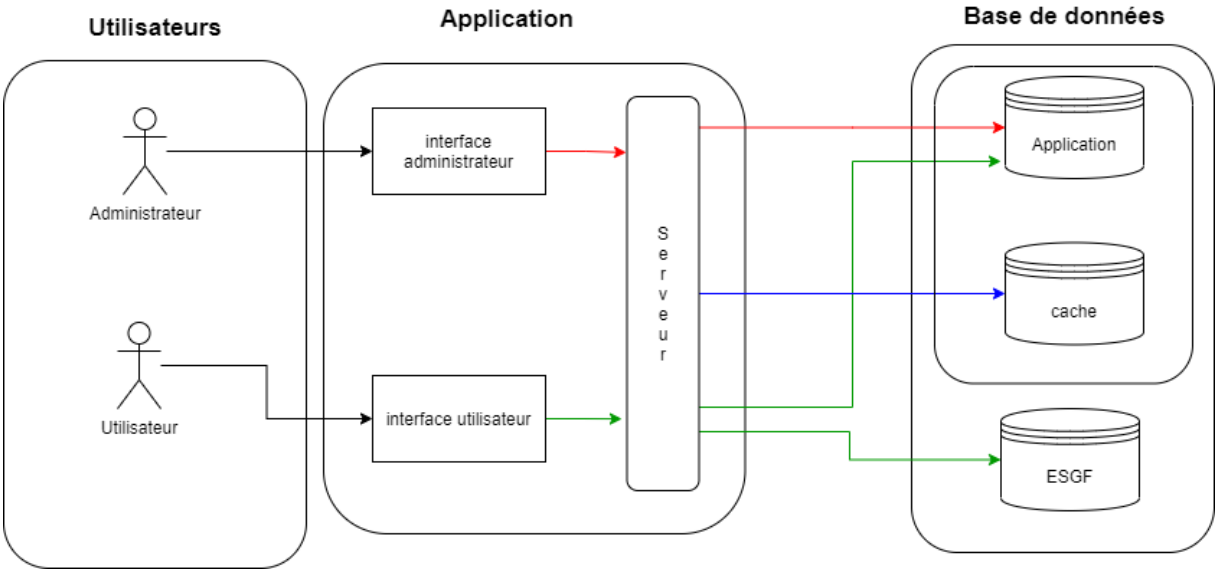

Figure 7: Architecture globale du site

Légende :

- Les flèches rouges signifient l'accès en lecture/écriture dans la base de données
- Les flèches vertes signifient l'accès uniquement en lecture dans les bases de données
- La flèche bleue signifie l'accès en lecture/écriture dans le cache

3.3.1 Utilisateurs

L'administrateur a accès à la base de données de l'application, qu'il pourra modifier en fonction des développements et des améliorations futures.

L'utilisateur pourra faire une sélection via l'interface utilisateur développée. Le site lui restituera un graphe de la sélection faite.

Au titre des améliorations futures, sera offerte à l'utilisateur la possibilité de contacter l'administrateur par courrier pour demander l'ajout de variables, modèles ou scénarios.

3.3.2 Application

L'application utilise MVT, architecture basée sur trois pôles : le Modèle, la Vue et le Template, qui respecte le principe de séparation des fonctions (Fig.8)

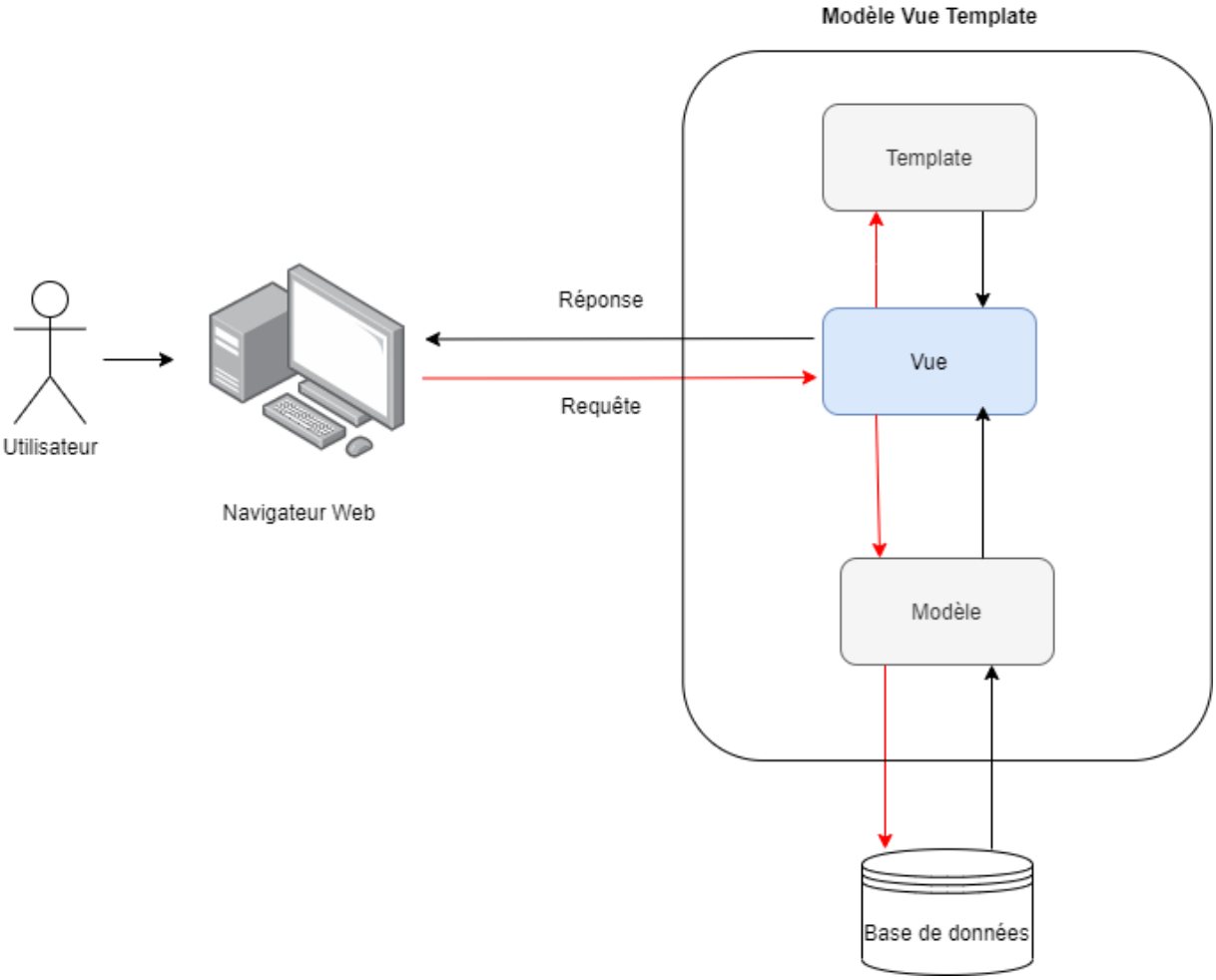

Figure 8: Diagramme "Model View Template"

Les flèches rouges figurent le cheminement de la requête et les noires son traitement.
La requête « http » (« Hypertext Transfer Protocol ») transite par la vue qui, après analyse, sollicite le Modèle pour obtenir les données puis fait appel au « template » pour formuler sa réponse, qui est renvoyée à l'utilisateur via le même protocole. Notons que cet échange ne nécessite pas d'être sécurisé.

3.3.2.1 La vue

La vue collecte les informations demandées lors de la formulation d'une requête et en envoie les résultats après exécution.
Si une requête nécessite une interaction avec la base de données, la vue appellera le modèle et récupèrera le résultat.

3.3.2.2 Le modèle

Le modèle interagit avec la base de données où il recherche les éléments correspondant à la requête.

La base de données est créée et organisée par des « Object Relational Mapping (ORM) » qui traduisent le résultat d'une requête SQL en objet python et inversement.

```python
class Variable(models.Model):
    name = models.CharField(max_length=100)
    full_name =  models.CharField(max_length=100)

    def __str__(self):
        return self.full_name
```

Figure 9: Structure d'un modèle en python

La Figure 9 crée un schéma ORM « class Variable ».

La fonction « Variable.objects.all( ) » permet la récupération de tous les éléments "variable" depuis la base de données.

3.3.2.3 Le Template

Le Template est un fichier HTML (Hypertext Makeup Language) qui prend en charge la mise en page et la représentation de pages web. Il est récupéré par la vue pour affichage à l'utilisateur. (fig.10)

```html
<div id="variable" class="select_block">
    <div>Variables</div>
    <select id="select_variable"  class="target">
    {% for variable in variables %}
        <option value={{variable.value}} selected> {{variable.full_name}}</option>
    {%endfor%}
    </select>
</div>
```

Figure 10: Partie de code HTML utilisant le contexte.

Cette image (Figure.10), montre la mise en page des éléments variable sur la page web. Ici, les différentes variables disponibles dans la base de données seront mises à disposition au client.

3.3.3 Base de données

La base de données a pour but d'enregistrer les différents modèles conçus par le développeur dans Django. (Figure 11.)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'djangodb',
        'USER': 'Admin',
        'PASSWORD':'password',
        'OPTIONS': {
         'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        },
    }
}
```

Figure 11: Création de la base de données en Django

Le STRICT_TRANS_TABLES est un mode permettant de gérer les messages d'erreur, lors d'un échec d'insertion.

Ci-dessous, un diagramme de base de données (Figure.12).

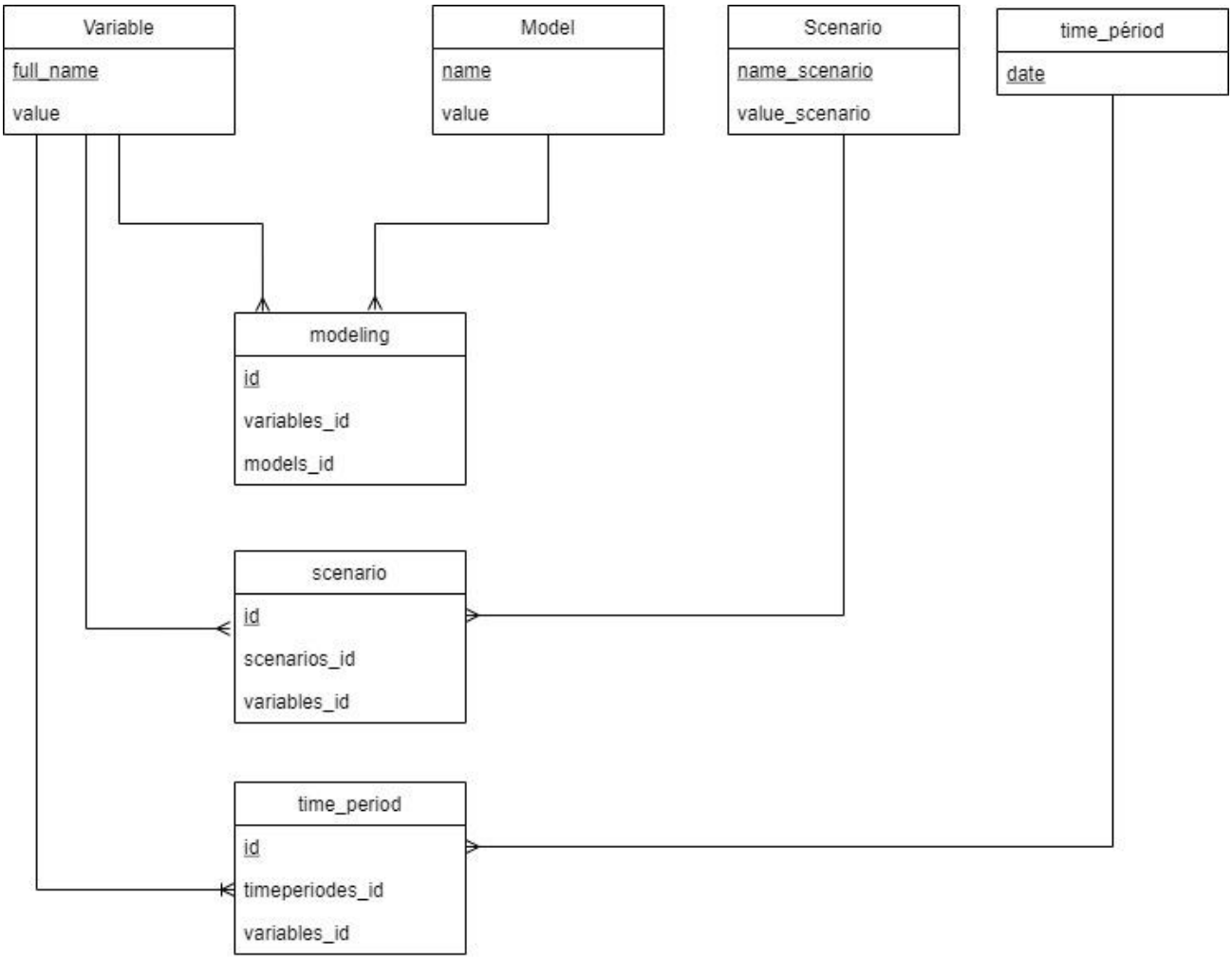

Figure 12: Diagramme de base de données

3.3.4 Caches

Un cache est une mémoire permettant la sauvegarde temporaire des données importées.
Deux caches sont nécessaires dans cette application (Fig.13), l'un pour conserver les URL « OpenDap » de la plateforme ESGF et l'autre pour conserver sous forme d'images les graphiques générés par le traitement, ce qui permet, pour une requête identique faite par plusieurs utilisateurs dans un laps de temps donné, de n'effectuer qu'une seule demande auprès de l'ESGF.

```
CACHES = {
    'default': {
        'Backend': 'django.core.cache.backends.db.DatabaseCache',
        'LOCATION': 'my_cache_table',
        'USER': 'Admin',
        'TIMEOUT': 60*60*24*2
    }
}
```

Figure 13: Création cache en Django

« Backend » est le type de cache utilisé.
« Location » indique le lieu où seront stockées les données.
« Timeout » fixe, en secondes, le temps de conservation des données dans le cache. 60*60*24*2 donne 172 800, soit 48 heures dans cet exemple.
Le cache a été positionné dans la base de données, ce qui permet de minimiser l'utilisation de la mémoire du serveur.

3.4 Implémentation

Dans ce chapitre, seront expliquées quelques parties d'implémentation pour réaliser le projet.

3.4.1 Contacter Django

3.4.1.1 Contacter Django via AJAX

Dans le cadre de ce projet, le choix s'est porté sur AJAX pour contacter Django.
AJAX « Asynchronous Javascript and Xml » est une technique qui permet de demander des informations au serveur et de les récupérer sans nécessiter un rafraîchissement de la page web, alors que ce même process, avec la fonction native de Django, aurait à chaque fois généré le rafraîchissement d'une nouvelle page, demandant plus de ressources.

```
function getPlot(url_search,send){
 $.ajax({
        type: 'GET',
        url: url_search,
        dataType: "json",
        contentType: 'application/json; charset=utf-8',
        data:send,
        success: function(data){
            document.getElementById("IMG_future").src=""+data["IMG_future"];
            $(".column").css({"display":"block"});
            showAlertSuccess();
          },
        error:function(data){
              showAlertError();
        }
       });
}
```

Figure 14: Exemple de requête Ajax

Voici les éléments importants pour comprendre l'exemple AJAX ci-dessus (Figure 14) :

- Url : permet d'identifier l'endroit où sera envoyée la requête.
- Type : type de la requête, « POST » ou « GET ».
    - GET permet de récupérer les informations du serveur
    - POST permet l'envoi et la sauvegarde d'informations
- Data : permet d'envoyer les informations utiles pour exécuter la requête par le serveur. Dans l'exemple ci-dessus, on envoie un fichier JSON contenant les coordonnées pour lesquelles on désire obtenir des informations.

Si la demande contient tous les éléments permettant au système de traiter la requête, la partie « success » véhiculera les différentes informations y répondant. (Fig 15)

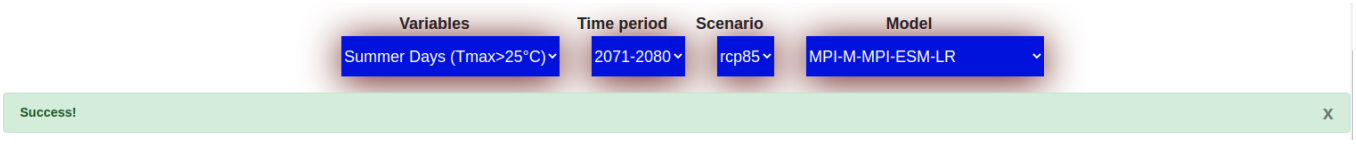

Figure 15 : Success

Si la demande contient des erreurs le système transmettra, via la partie « error », un message qui sera repris comme réponse à la requête du client. (Fig.16)

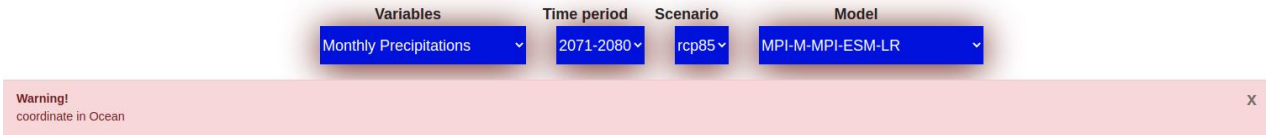

Figure 16: Erreur coordonnées

3.4.1.2 Traitement de la requête AJAX par Django

Lors de la réception de la requête AJAX, Django va suivre ce protocole d'exécution (fig.17) :

```
from django.urls import path
from . import views

urlpatterns = [
    path('Temperature/', views.get_Temperature_plot, name="map_info"),
    path('Rainfall/', views.get_Rainfall_plot, name="map_info"),
    path('TempData/', views.TempData, name="map_info"),
    path('HistoricalPrec/',views.historical_prec,name="historical_prec_info"),
    path('',views.index,name='index')
]
```

Figure 17: Contenu du fichier urls de Django

Première étape : identification des urls figurant dans la requête ;
Deuxième étape : chargement du module python et recherche des variables « urlpatterns » ;
Troisième étape, Django parcourt chaque chemin de l'url dans l'ordre d'apparition et transmet à la vue la première correspondance qu'il trouve pour répondre au client ;
Message d'erreur si aucune correspondance n'est trouvée.

3.4.2 Traitement d'une requête demandant les ressources de l'ESGF

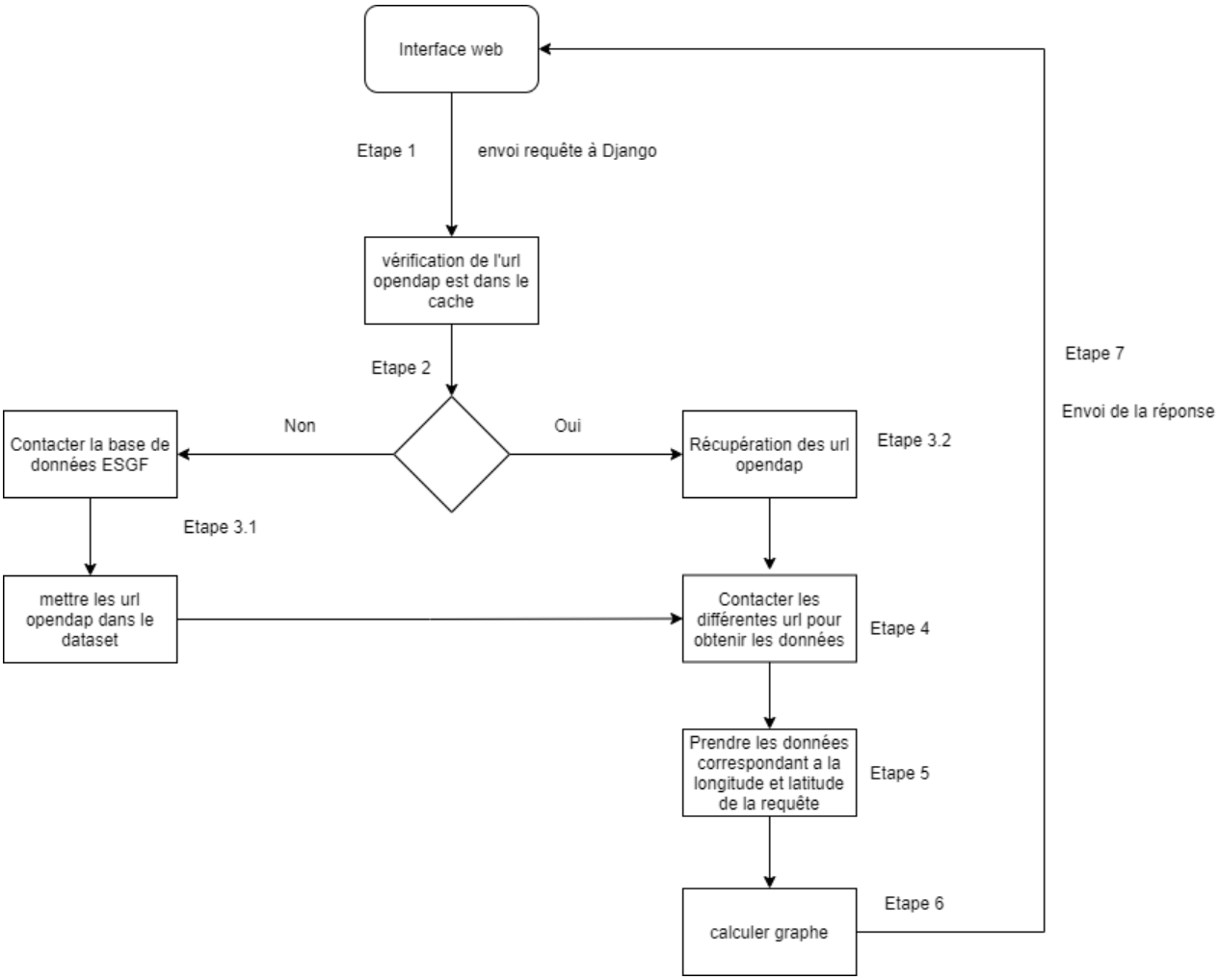

Figure 18: Diagramme de communication entre Django et ESGF.

Vous trouverez ci-dessous les différentes étapes du diagramme ci-dessus (Figure 18) :

Etape 1, le client envoie une requête à Django contenant toutes les informations nécessaires :

- Coordonnées géographiques
- Période de temps
- Modèle
- Scénario
- Variable

Etape 2, Django analyse les informations reçues et fait sa demande au cache ;

Si la demande est complète et si elle contient les urls Opendap exigées, passage à l'étape 3.2 où Django contacte le cache pour obtenir les urls correspondantes à la requête ;

Dans le cas contraire, exécution de l'étape 3.1 où Django contacte la base de données ESGF pour obtenir les urls correspondantes au scenario, période de temps (2070 à 2010) et requis. Ensuite Django va mettre en cache les url obtenues et passer à l'étape 4 ;
Etape 4, Django contacte les urls pour obtenir leurs données. La création d'une session est nécessaire car Opendap a besoin d'un certificat d'authentification de l'ESGF pour y avoir accès ;
Etape 5, Django va réduire l'ensemble des données reçues aux seules données de longitude et de latitude désirées.
L'étape 6, décrite ci-dessous, est la création du graphe et l'étape 7 est l'envoi de la réponse à l'interface utilisateur.

3.4.3 Création de graphe

La création d'un graphe impose le respect de certaines étapes qui sont :

1- Extraction des données
2- Changement d'unité et calcul
3- Création du graphe

3.4.3.1 Extraction des données

Le choix s'est porté sur le module « Numpy » pour parcourir, analyser et manipuler les données.
« Numpy » est un module permettant de manipuler des formules mathématiques sur un grand nombre de données à l'aide de tableaux, matrices et tableaux multidimensionnels.

```
def getPixelPointInFile(dataset,coord,variable_to_extract):
    prec = dataset.variables[variable_to_extract][:]
    # obtenir l'indice le plus proche de la valeur de la longitude et latitude donnée
    rlats = dataset.variables['rlat'][:]
    idxLat = bisect(rlats,float(coord[0]))
    rlongs= dataset.variables['rlon'][:]
    idxLong = bisect(rlongs,float(coord[1]))
    prec_data = prec[:,idxLat,idxLong]
    prec_data_unmask = prec_data.data
    return prec_data_unmask
```

Figure 19: Code pour obtenir la coordonnée la plus proche du point demandé

Cette fonction permet de récupérer, selon la résolution disponible, les données centrées sur la coordonnée géographique demandée (Fig.19).

Les paramètres de cette fonction sont :

- dataset : comprend les données du fichier Netcdf ;
- coord : renseigne les coordonnées géographiques de la requête ;
- variable_to_extract : le nom de la variable dans le fichier Netcdf. Par exemple, pour les précipitations, le nom est « prec ».

Dans la partie de code ci-dessus, les points importants sont :

- dataset.variables : permet de récupérer le tableau, la matrice ou le tableau multidimensionnel contenu dans le fichier NetCdf et manipulable par le module Numpy ;
- idxLat et idxLong : permet de donner les indices du tableau où se trouvent les coordonnées les plus proches des coordonnées de la requête. Pour ce faire, un algorithme de bissection a été utilisé ;
- Prec[:,idxLat,idxLong] : récupère les données de précipitation mensuelle pour ces coordonnées.

Le passage par ce processus est nécessaire car le fichier NetCDF contient un tableau de taille 201 x 0.44 degrés pour la latitude et de 194 x 0.44 degrés pour la longitude. Pour une coordonnée particulière, il donne la valeur de la coordonnée la plus proche de celle de la requête.

3.4.3.2 Changement d'unité spécifique

L'unité des précipitations dans le fichier NetCDF est le $kg/m^{-2}s^{-1}$. Le kg est une mesure de masse correspondant, pour de l'eau, à un volume de 0.001 m^3. En le rapportant à une surface de 1 m², on obtient une hauteur de 1mm. Il est donc pertinent de dire que l'unité une fois rationnalisée est le mm/seconde.

La transformation de kg/m²/s en mm/mois s'effectue selon la formule :

(Moyenne_prec_mois *86400* Nb_jour_mois)

Avec :

- Moyenne_prec_mois : la moyenne des valeurs de la précipitation pour le mois ;
- 86400 : nombre de secondes dans une journée (60*60*24) ;
- Nb_jour_mois : le nombre de jours dans un mois.

3.4.3.3 Création de graphe

Dans cette partie, le module « pyplot » est utilisé pour générer un graphe.

```
def simple_plot_graph(x,y,title,ylabel,xlabel,path,legend):
    plt.title(title)
    plt.ylabel(ylabel)
    plt.xlabel(xlabel)
    plt.plot(x, y, color='blue', linestyle='dashed', linewidth=3,
             marker='o', markerfacecolor='blue', markersize=12)
    plt.legend(legend)
    plt.savefig(path)
    plt.clf()
    return path
```

Figure 20: Fonction permettant de créer un graphe.

Cette fonction contient plusieurs paramètres (Figure 20) :

- x : les valeurs des données en abscisses ;
- y : les valeurs des données en ordonnées ;
- title : le titre du graphe ;
- ylabel : unité des données en abscisses ;
- xlabel : unité des données en ordonnées ;
- path : le chemin pour enregistrer le graphe généré sous forme d'image.

3.4.4 Carte colorée

Leaflet utilise les fichiers « Geojson »[1] pour cartographier sous forme de grille colorée. C'est un format d'encodage d'ensemble de données géospatiales qui utilise la norme Json « JavaScript Object Notation » comme format de données textuelles.

Un document Json comprend deux types de d'éléments structurels :

- Des ensembles de paires {« nom » : « valeur »}
- Des listes ordonnées de valeurs.

Chaque valeur peut être représentée par des tableaux, des nombres, une chaine de caractères, …

```
{"type": "FeatureCollection",
 "features": [
    {"type": "Feature",
     "properties": {"value": 43},
     "geometry": {
        "type": "Polygon",
        "coordinates":[
            [
                [19.799999237060547, -34.7599983215332],
                [19.799999237060547, -34.31999969482422],
                [20.239999771118164, -34.31999969482422],
                [20.239999771118164, -34.7599983215332],
                [19.799999237060547, -34.7599983215332]
            ]
        ]
     }
    }
 ]
}
```


Figure 21: Exemple fichier Geojson

Format d'un fichier Geojson (Figure 21) :

Les fichiers Geojson sont obligatoirement composés d'une clé « type » et d'une valeur « FeatureCollection » ainsi qu'une clé « features » qui a comme valeur un tableau d'objet.

Chaque objet de ce tableau contient comme clés le « type » et la « geometry ». La clé « properties » est un élément facultatif permettant de rajouter des éléments supplémentaires à l'objet, comme ci-dessus la valeur de la zone.

« geometry » contient un objet primitif de type polygone dont la clé coordonnée, la valeur de début et la valeur de fin doivent être identiques.

Pour obtenir plus de détail sur l'ensemble des objets primitif, suivre le lien (https://fr.wikipedia.org/wiki/GeoJSON).[1]

Toute la gestion des cartes graphiques est réalisée par le navigateur web en javascript.

Les fichiers Geojson sont assemblés par le serveur puis envoyés au navigateur en fonction de la requête du client.

Ces fichiers sont dits statiques, c'est-à-dire qu'ils ne sont pas créés au moment de la requête. Ceci implique que l'ajout d'un nouvel élément de requête exigera la création d'un nouveau fichier GeoJson, à ajouter en local.

[1] Pour plus de détail sur les fichiers Geojson : https://datatracker.ietf.org/doc/html/rfc7946, consulté le 20/06/2021

Le choix de fichiers statiques a été imposé par la contrainte du timeout. En effet, la création dynamique d'un fichier GeoJson pour tout le continent africain serait tellement longue que le serveur interromprait son exécution (Job aborted/ timeout), sans réponse coté client.

La recherche d'une solution au timeout pourrait faire l'objet d'un axe d'amélioration de ce projet.

4 Résultats

Dans ce chapitre, seront expliquées les différentes fonctionnalités de l'application développée, d'abord du point de vue de l'utilisateur (interface utilisateurs), puis du point de vue du gestionnaire (interface administrateur).

L'entièreté du code de l'application est disponible sur Github via :

https://github.com/JimmyDelacruz/These_climate.git

4.1 Solution

4.1.1 L'interface utilisateur

4.1.1.1 Vue d'ensemble

L'interface utilisateur se présente comme suit, figure.22

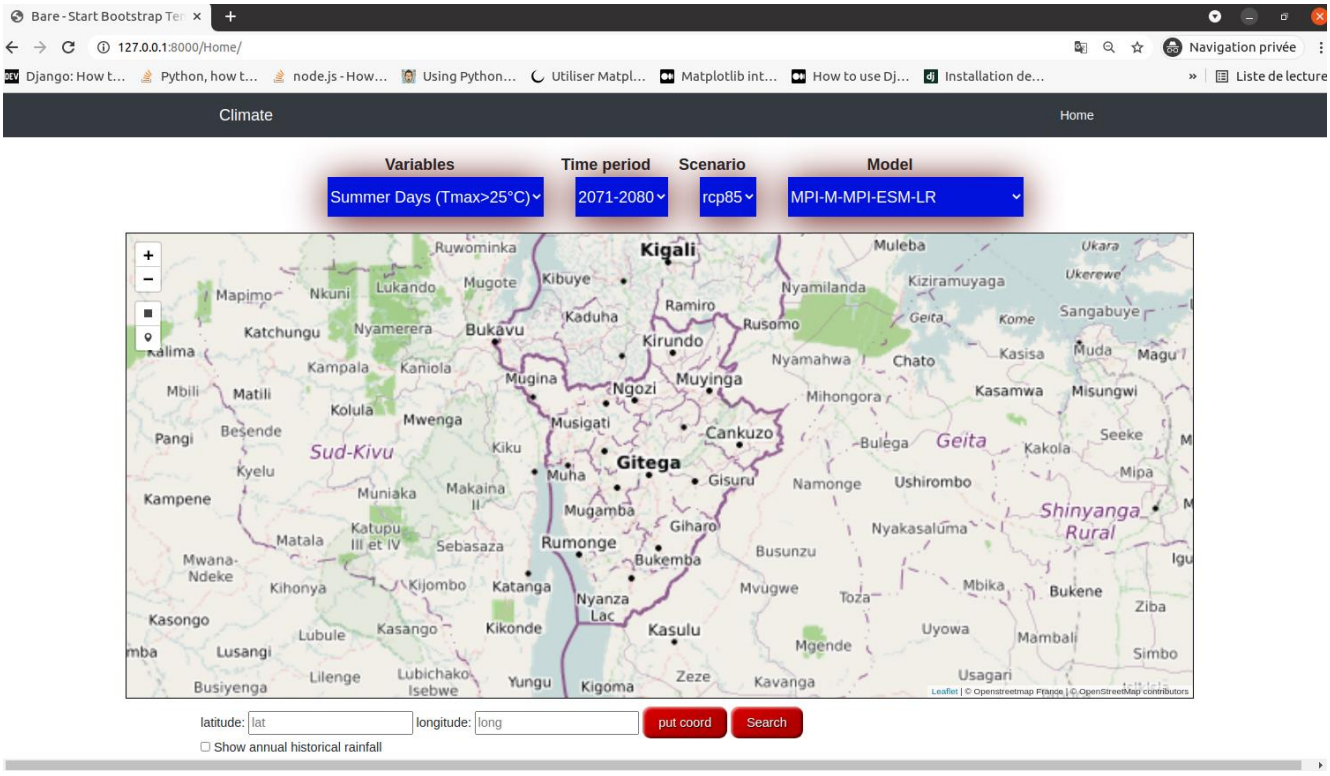

Figure 22: Page d'accueil

Ce projet donne la possibilité à l'utilisateur de choisir les variables qui l'intéressent parmi celles mises à disposition.

Une simulation climatique est la projection de l'évolution du climat et de ses interactions sur une variable sélectionnée, sur base :

- d'un scénario d'émission de gaz à effet de serre (RCP),
- d'un modèle,
- d'une période qui peut varier d'une à plusieurs années.

Au stade de développement actuel, l'utilisateur ne dispose que d'un scénario climatique, le RCP8.5 (le plus extrême et pessimiste : forçage radiatif pour l'année 2100 de +8,5 W/m2).

Il aura en revanche la possibilité de sélectionner une variable, une période de temps et un modèle sur la base des catégories mises à disposition.

Choix d'une variable parmi quatre possibles :

- Hot Days (Tmax > 35°C), le nombre de jours ayant atteint plus de 35 degrés Celsius ;
- Summer Days (Tmax > 25°C), le nombre de jours ayant atteint plus de 25 degrés Celsius ;
- Monthly Temperature, la moyenne des températures, en degré Celsius, pour une période donnée ;
- Monthly Precipitation, la moyenne des précipitations, en mm/jour, pour une période donnée.

Choix d'une période de temps parmi quatre possibles :

- 2071 à 2080
- 2081 à 2090
- 2090 à 2100
- 2071 à 2100

Choix d'un des trois modèles climatiques suivants :

- MPI-M-MPO-ESM-LR
- MOHC-HadGeM2-ES
- CNRM-CERFACS-CNRM-CM5

Ces modèles présentent tous une structure similaire (période de temps / scenario / variable).

D’autres pourront être intégrés par la suite, dès lors qu’ils respectent cette même structure, tels que :

- ECMWF-ERAINT
- IHEC-EC-EARTH

Le développement autorise un choix de modèle supplémentaire dit « Ensemble », qui permet de réaliser un graphe en agrégeant tous les modèles.

L’utilisateur doit indiquer les coordonnées géographiques de sa recherche en sélectionnant une position, montrée à la Figure 23 ou une zone à la Figure 24 via la saisie de la latitude et de la longitude (exprimée en décimales) ou via la sélection d’une des options à gauche.
En Annexe, quelques exemples de graphe provenant de différentes requêtes.

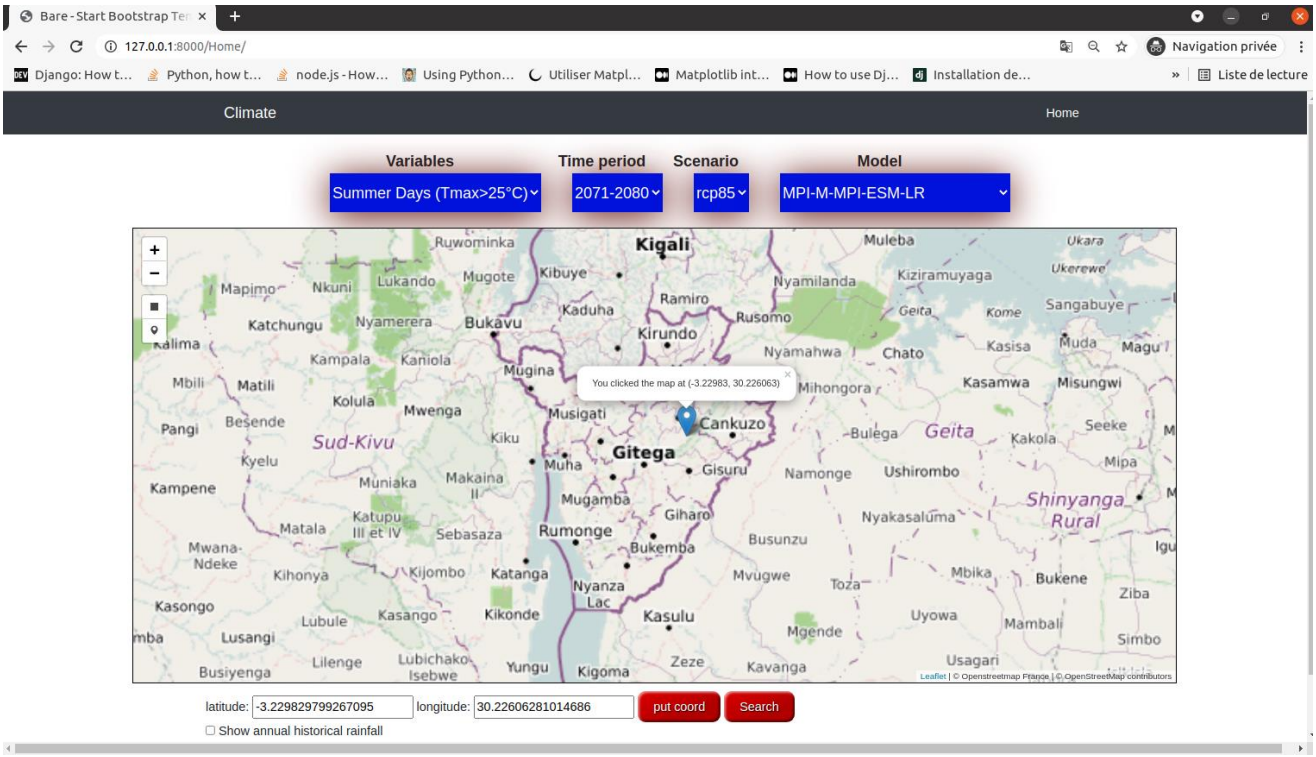

Figure 23: Sélection d’une coordonnée

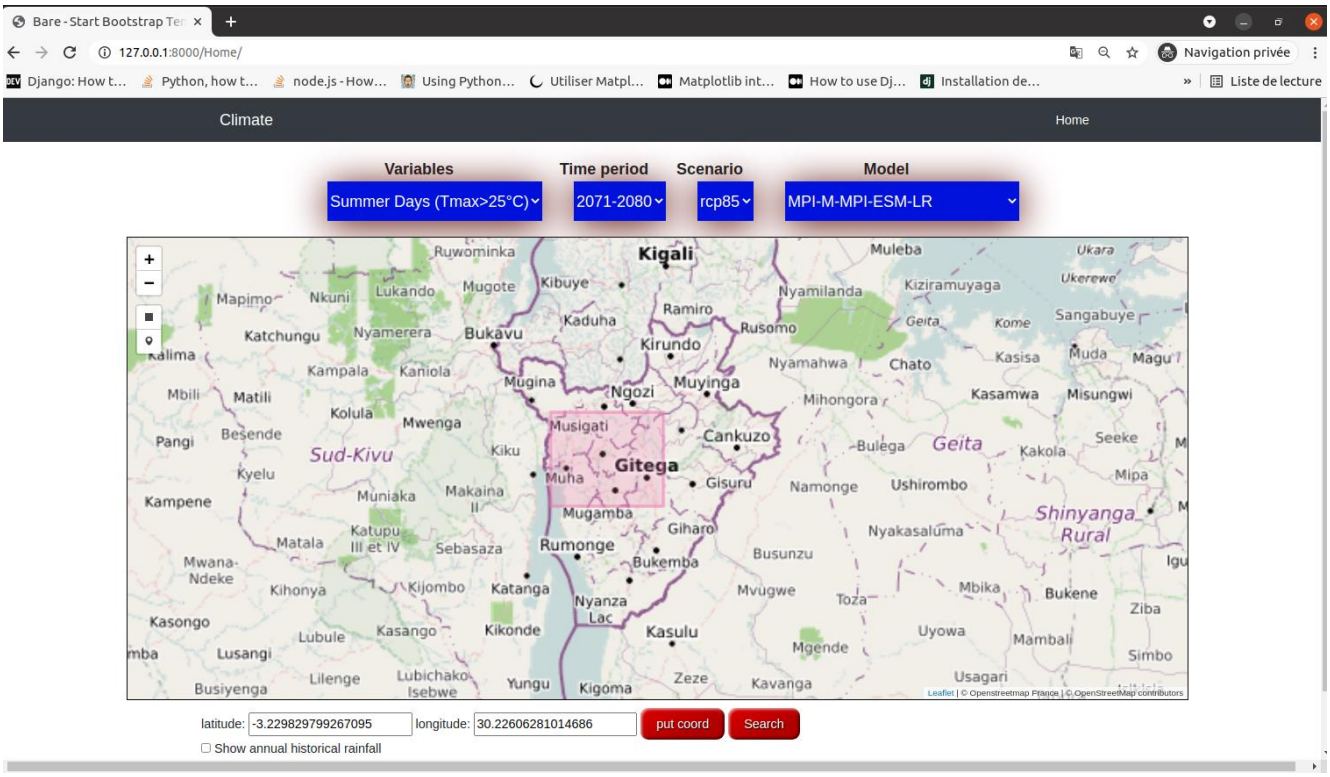

Figure 24:Sélection d'une zone

4.1.1.2 Carte colorée

L'utilisateur a la possibilité de consulter deux types de grilles, créées pour l'ensemble du continent africain.

La première est une grille historique développée sur base des informations exportées du site du CRU. Elle reprend la moyenne des précipitations annuelles pour la période allant de 1960 à 1990, exprimée en mm/jour. La visualisation de cette carte s'effectue en sélectionnant la case « show annual rainfall historical » (Fig.25)

La seconde est une grille climatique développée sur base des informations exportées du site de l'ESGF. Elle traite des températures et est accessible via l'interface de recherche en choisissant les variables liées aux températures qui sont « hot days » ou « summer days » suivant la sélection du « Time Period ». (Fig.26)

Ce choix pourra être visualisé sur la carte géographique du continent africain, avec une fonction « zoom ».

En pointant avec la souris sur la zone sélectionnée, l'utilisateur aura en lecture directe les précipitations historiques ou le nombre de jours ayant atteint la température maximale.

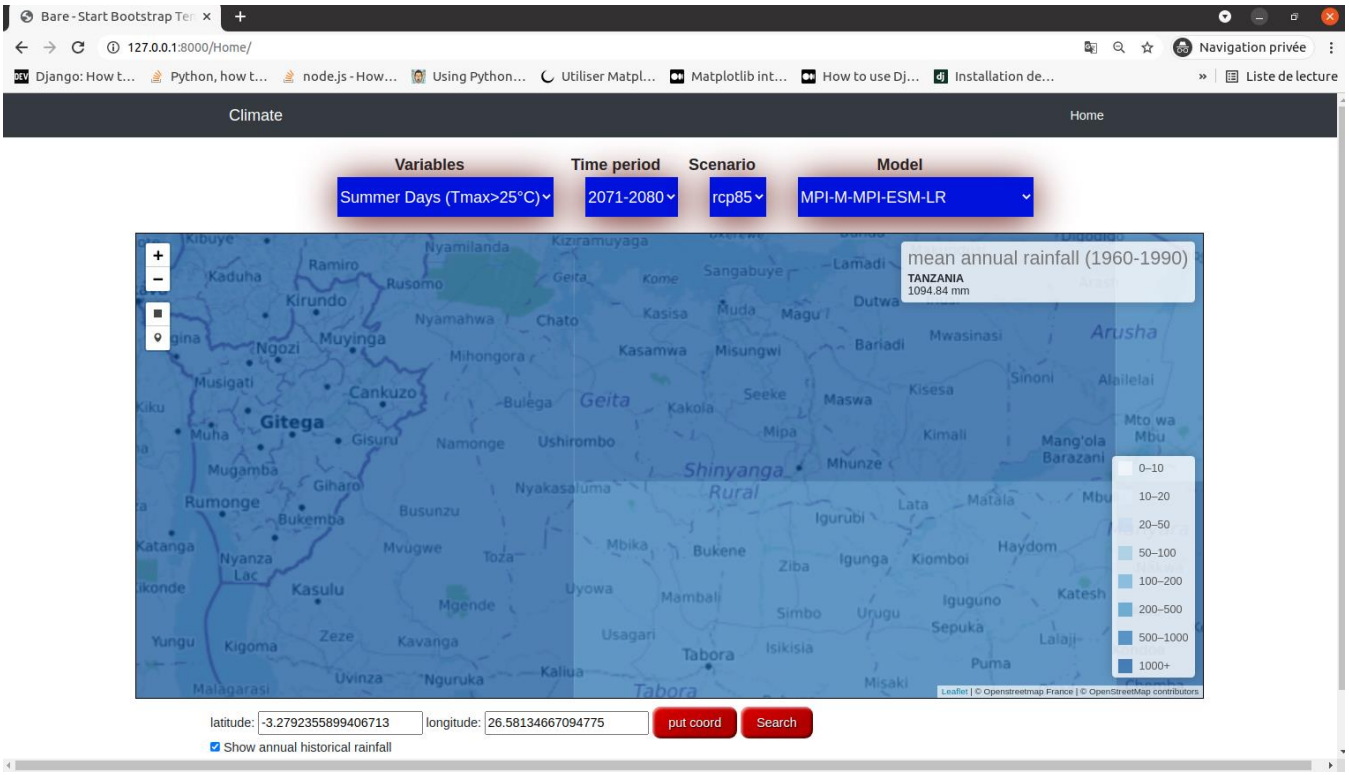

Figure 25: Précipitation historique

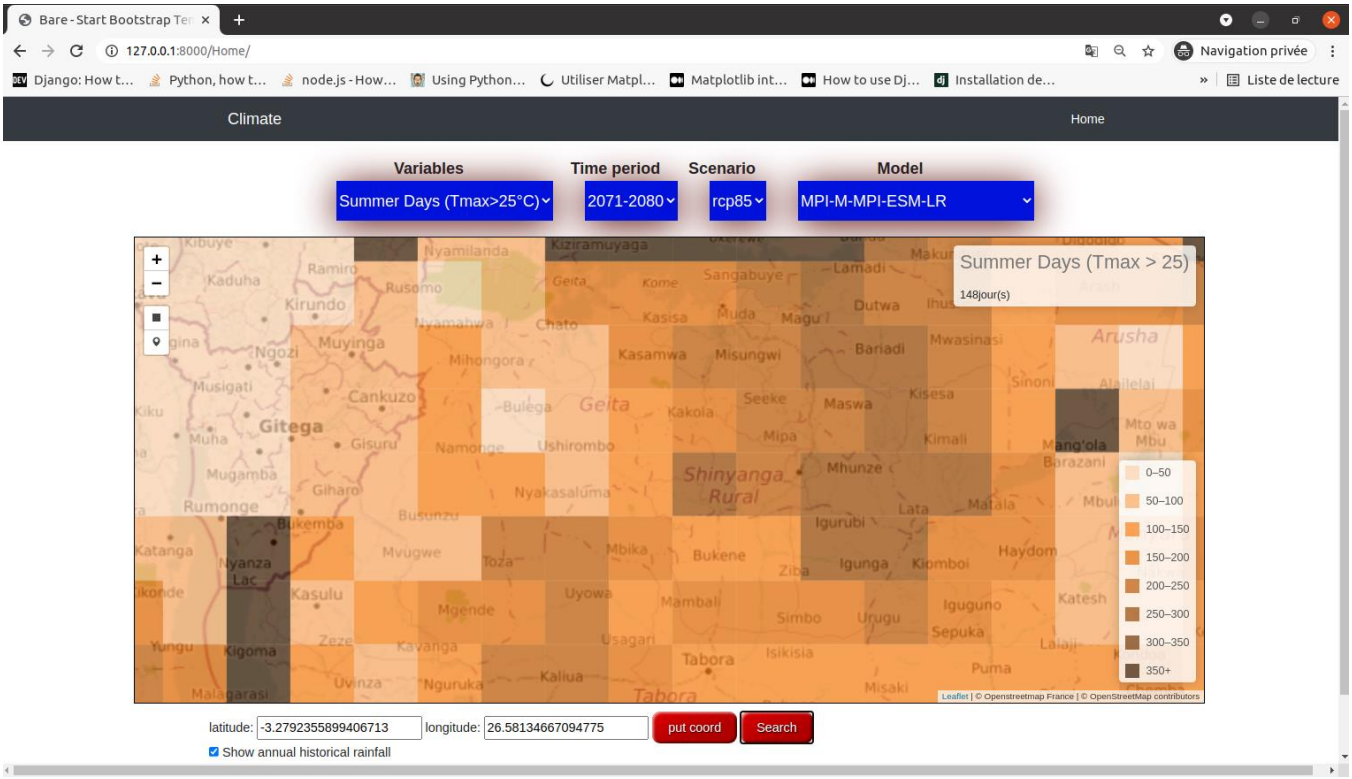

Figure 26:Summer Days

4.1.1.3 Chargement

Un spinner a été ajouté (voir ci-dessous, figure 27) afin d'indiquer à l'utilisateur que l'action demandée est en cours de réalisation, quand le temps d'attente devient trop long.

Figure 27: Spinner

L'étude de Harrison et al. (2010) montre qu'au-delà de 1 seconde, l'utilisateur a une sensation de perte de contrôle. L'ajout d'une barre de progression ou d'un spinner donne, en fixant l'attention, une impression de durée plus courte.

4.1.2 L'interface administrateur

L'administrateur peut apporter des modifications à la base de données existante telles que l'ajout ou la suppression de :

- Variables (°)
- Scénarios (*)
- Modèles (*)

  (°) Au niveau des variables, l'ajout d'un nouvel indicateur imposera une nouvelle implémentation au développeur, associée à une date de mise en service. La lourdeur de l'opération dépendra de l'indicateur et de sa complexité.

  (*) Pour ces deux points, s'ils existent déjà dans la base de données ESGF, leur ajout sera immédiatement disponible pour l'utilisateur, à coût nul tant que ESGF reste gratuit. A l'inverse, une suppression est envisageable si un modèle devient obsolète ou n'est plus exploitable.

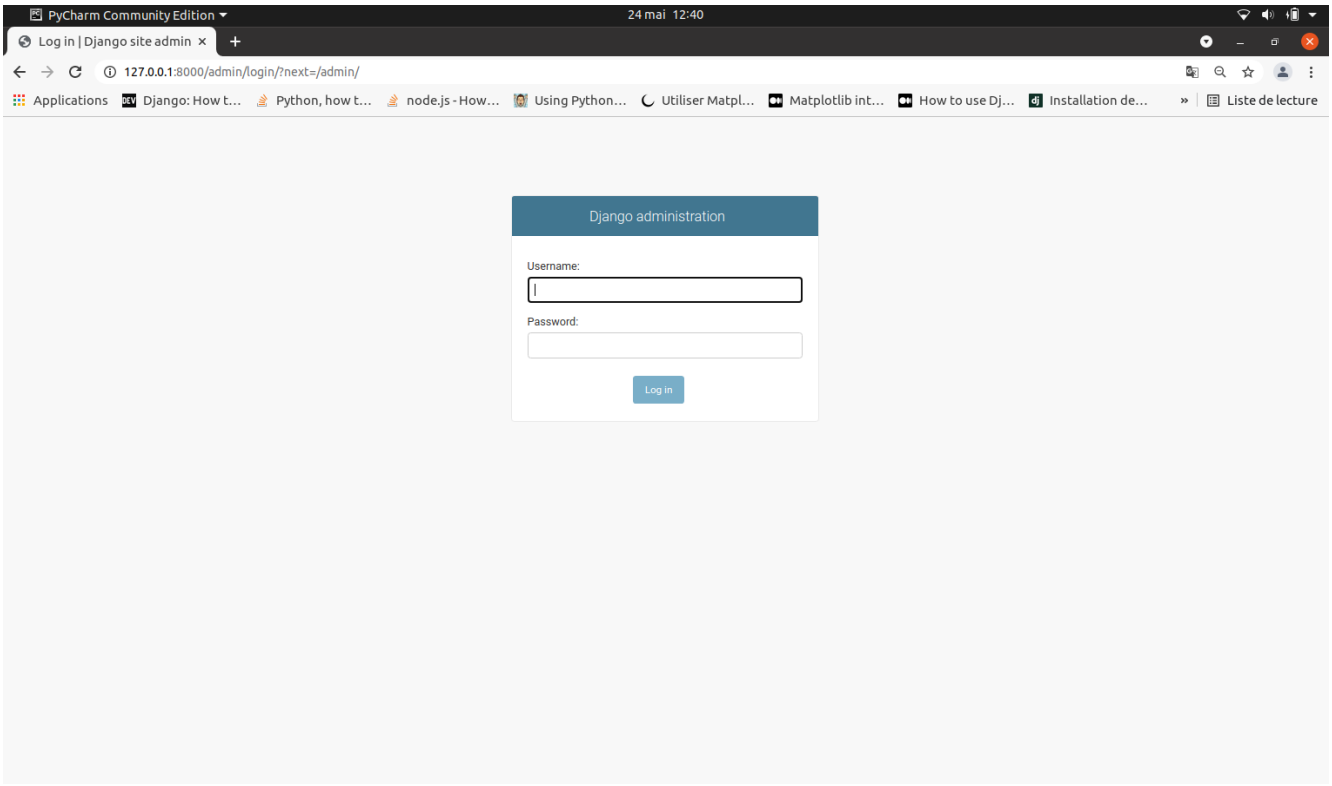

Figure 28:Page d'accueil administrateur

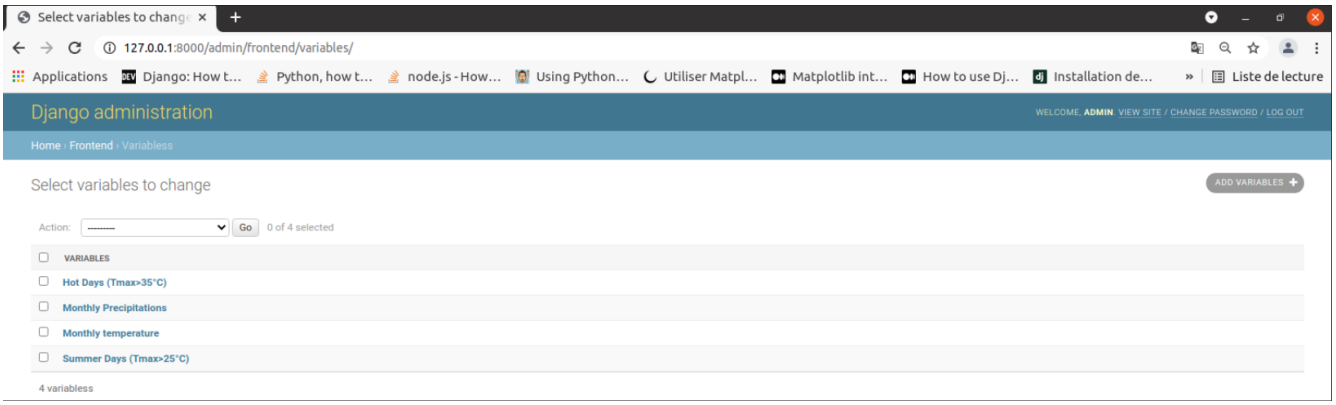

Figure 29: Ensemble des variables dans la base de données

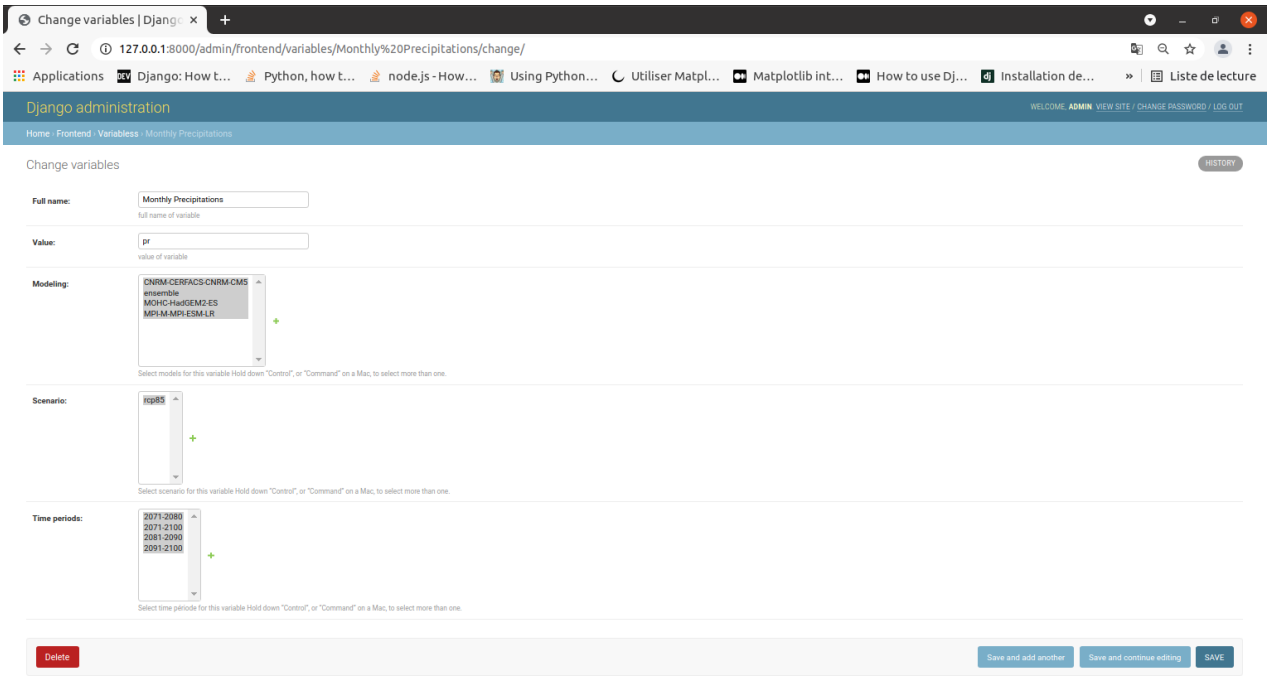

Figure 30: Ajout d'une nouvelle variable dans la base de données

4.2 Tests

Dans ce chapitre seront expliquées les méthodes de tests pour valider notre système.
Tous les tests se trouvent dans le dossier « *tests* » du projet.

4.2.1 Tests unitaires

Ce type de test permet de vérifier le bon fonctionnement d'une partie bien précise du code.
Les parties modèle et requête à la base de données ne sont pas testées, le système de modèle étant entièrement géré par Django.
Le plugin ESGF permettant la communication avec la base de données externe a déjà subi un certain nombre de tests, il n'est pas utile d'en faire davantage pour le projet.

```
def test_temperature_coord(self):
    files = glob.glob(FilesTemp + "/*")
    datasets = []
    for file in files:
        datasets.append(Dataset(file, 'r'))
    temp = decode_temperature.get_mean_temperature_coord(datasets,["-3","30"],"2071")
    result = [int(i) for i in temp]
    self.assertEqual(12, len(result))
    self.assertEqual([21, 22, 22, 21, 20, 20, 22, 24, 25, 23, 20, 20], result)
```

Figure 31: Exemple code test unitaire

Ce test, voir Figure 31, permet de valider le bon fonctionnement de cette méthode. Il compare le résultat obtenu par la fonction « *get_mean_temperature_coord* » et le résultat qu'il devrait obtenir. Si le résultat est identique, le test est validé, dans le cas contraire, une erreur s'est produite dans la méthode.

4.2.2 Tests fonctionnels

Il est primordial de tester la view (Vue), qui représente l'ensemble des exécutions du programme.

Django a une classe « Client » qui simule une requête d'un navigateur. Il permet donc de valider le bon fonctionnement de chaque view.

```
def test_view_Temperature_pixel_200(self):
    url = reverse('temperature')
    response = self.client.get(url, {"coord_1[]":['-3.2627672624439024', '30.04486029216889'],
    "variable":'ts',"model":'MPI-M-MPI-ESM-LR',"scenario":'rcp85',"time_period":"2071-2080"})
    self.assertEquals(response.status_code,200)
```

Figure 32: exemple code test fonctionnel

Ce test (Figure.32) permet de voir si la vue renvoie bien le code 200, qui valide le succès de la requête.

Parmi l'ensemble des codes http, voici ceux utilisés pour les tests :

(https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP).

- 400 : coordonnées dans l'océan
- 442 : coordonnées hors du continent Africain
- 500 : erreur du serveur.

4.2.3 Tests utilisateur

Les tests utilisateurs consistent à valider soi-même le comportement de l'application, c'est-à-dire que le développeur se met à la place du client et exécute différents scénarios pour voir si tous fonctionnent correctement.

Par exemple, un type de scénario est d'établir une recherche avec des coordonnées hors zone (le continent africain) et voir un message d'erreur apparaitre.

5 Discussion et perspectives d'améliorations

5.1 Discussion

Après réflexion, Django n'était pas nécessairement le candidat idéal pour réaliser cette application car il m'a demandé beaucoup de temps pour comprendre son fonctionnement, alors qu'un autre framework aurait pu être plus simple à appréhender.

Au lieu d'utiliser un Framework web existant, la création d'une API générale aurait pu être envisagée. « Une Application Programming Interface (API) permet de rendre disponible les données ou les fonctionnalités d'une application existante, afin que d'autres applications les utilisent » (https://www.agencedebord.com/api-definition-utilisation/ , *consulté le 13 aout 2021*) fournissant ainsi une flexibilité dans le développement des applications (Web, Android, iPhone, …)

Leaflet a été un candidat parfait pour ce prototype, étant gratuit et ayant beaucoup de tutoriels pour les interactions utilisateurs. Cependant, la création d'une grille colorée m'a demandé beaucoup de recherche sur internet pour découvrir et bien manipuler les fichiers Geojson utilisés par cette application.

L'utilisation du module « numpy » a été essentielle pour la manipulation des différents tableaux multi-dimensionnels présents dans ce mémoire. Simple d'usage, il facilite grandement le code à réaliser par le développeur.

5.2 Améliorations

Pour les développeurs qui prendront en charge la suite de ce projet, voici quelques points qui pourraient être ajoutés.

5.2.1 Calculer les SPEI

Il serait possible d'ajouter une nouvelle variable sur le site web permettant le calcul du Standardised Precipitation-Evapotranspiration Index (SPEI).

Le SPEI est conçu pour prendre en compte les précipitations et l'évapotranspiration potentielle dans la détermination de la sécheresse. Il capture l'impact principal de l'augmentation des températures sur la demande en eau.
La NOAA a créé un projet (https://github.com/monocongo/climate_indices) permettant de calculer différents indices du climat (SPI, SPEI, PET).
Sachant que le langage R permet le calcul du SPEI, il serait possible d'utiliser les fonctionnalités R en python via des modules.

5.2.2 Amélioration de la rapidité de création d'un Geojson.

Actuellement les fichiers Geojson sont enregistrés sur la mémoire locale du serveur car la création du fichier Geojson pour tout le continent africain demanderait trop de temps d'exécution, entraînant un timeout du serveur et l'abort de la tâche.
L'amélioration de l'algorithme de création permettra de générer ce fichier de manière dynamique et de l'enregistrer dans le cache une fois créé.
Cette amélioration permettrait de gagner de l'espace mémoire et donc un gain financier sur la location du serveur.

5.2.3 Contacter l'administrateur du site web par mail

Cet ajout permettrait aux utilisateurs de contacter l'administrateur, soit pour demander l'ajout d'indicateurs climatiques, soit pour proposer des améliorations, soit pour transmettre des bugs éventuels.

5.2.4 Réalisation de test sélénium

Les tests sélénium sont des tests permettant de réaliser des scénarios client en simulant un navigateur web et interagissant avec celui-ci comme un réel utilisateur de l'application. Ce type de test peut être un avantage pour le projet car cela éviterait aux développeurs de tester chaque fonctionnalité une à une et donc un gain de temps pour la suite du projet.

6. Conclusion

La finalité de ce mémoire était de développer un prototype d'application web permettant de récupérer des données climatiques d'une région spécifique et de les retranscrire sous forme d'indicateurs prévisionnels de sécurité hydrique, utilisable par quiconque souhaitant s'informer sur ces prévisions. Dans notre cas, la région spécifique est la région des Grands Lacs.

La solution informatique qui a été proposée pour ce mémoire est un prototype de site web créé avec le Framework Django. Il permet à l'utilisateur de rechercher sur base de critères : modèle, coordonnées géographiques, scénario RCP 8.5, etc. Suivant ces critères, les données sont récupérées sur la plateforme ESGF puis retranscrites sur le site web sous le format d'un graphe prévisionnel. Il est également possible d'afficher une carte colorée de l'Afrique.

L'ensemble des objectifs spécifiques cité dans ce document ont été atteints, à part l'exportation des ressources « données calculées et graphes » qui n'a pas pu être développée à temps.

Toutefois, le prototype d'application, originellement conçu pour s'appliquer à la région des grands lacs, a été étendu à l'ensemble du continent africain.

Annexes

```
netcdf file:pr_AFR-44_MPI-M-MPI-ESM-LR_rcp85_r1i1p1_CLMcom-CCLM4-8-17_v1_mon_203101-204012.nc {
  dimensions:
    time = UNLIMITED;   // (120 currently)
    rlat = 201;
    rlon = 194;
    bnds = 2;
    vertices = 4;
  variables:
    double time(time=120);
      :bounds = "time_bnds";
      :units = "days since 1949-12-01T00:00:00Z";
      :calendar = "proleptic_gregorian";
      :axis = "T";
      :long_name = "time";
      :standard_name = "time";
      :_ChunkSizes = 1U; // uint

    double time_bnds(time=120, bnds=2);
      :_ChunkSizes = 1U, 2U; // uint

    double rlat(rlat=201);
      :units = "degrees";
      :axis = "Y";
      :long_name = "latitude in rotated pole grid";
      :standard_name = "grid_latitude";
      :_ChunkSizes = 201U; // uint

    double rlon(rlon=194);
      :units = "degrees";
      :axis = "X";
      :long_name = "longitude in rotated pole grid";
      :standard_name = "grid_longitude";
      :_ChunkSizes = 194U; // uint

    int rotated_latitude_longitude;
      :grid_mapping_name = "rotated_latitude_longitude";
      :grid_north_pole_latitude = 90.0; // double
      :grid_north_pole_longitude = 180.0; // double
      :north_pole_grid_longitude = 0.0; // double

```

Figure 33: Fichier NetCDF partie 1

```
40
41      float lat(rlat=201, rlon=194);
42        :standard_name = "latitude";
43        :long_name = "latitude coordinate";
44        :units = "degrees_north";
45        :bounds = "lat_vertices";
46        :_ChunkSizes = 201U, 194U; // uint
47
48      float lon(rlat=201, rlon=194);
49        :standard_name = "longitude";
50        :long_name = "longitude coordinate";
51        :units = "degrees_east";
52        :bounds = "lon_vertices";
53        :_ChunkSizes = 201U, 194U; // uint
54
55      float lat_vertices(rlat=201, rlon=194, vertices=4);
56        :units = "degrees_north";
57        :_ChunkSizes = 201U, 194U, 4U; // uint
58
59      float lon_vertices(rlat=201, rlon=194, vertices=4);
60        :units = "degrees_east";
61        :_ChunkSizes = 201U, 194U, 4U; // uint
62
63      float pr(time=120, rlat=201, rlon=194);
64        :standard_name = "precipitation_flux";
65        :long_name = "Precipitation";
66        :comment = "at surface; includes both liquid and solid phases from all types of clouds
67        (both large-scale and convective)";
68        :units = "kg m-2 s-1";
69        :cell_methods = "time: mean";
70        :missing_value = 1.0E20f; // float
71        :_FillValue = 1.0E20f; // float
72        :associated_files = "gridspecFile: gridspec_atmos_fx_CLMcom-CCLM4-8-17_rcp85_r0i0p0.nc";
73        :grid_mapping = "rotated_latitude_longitude";
74        :coordinates = "lat lon";
75        :_ChunkSizes = 1U, 201U, 194U; // uint
```

Figure 34: Fichier NetCDF partie 2

```
77  // global attributes:
78  :institution = "Climate Limited-area Modelling Community (CLM-Community)";
79  :institute_id = "CLMcom";
80  :experiment_id = "rcp85";
81  :source = "CLMcom-CCLM4-8-17";
82  :model_id = "CLMcom-CCLM4-8-17";
83  :contact = "cordex-cclm@dkrz.de";
84  :history = "Processing for CORDEX archive at DKRZ (SVN revision 4244
85    http://svn-mad.zmaw.de/svn/mad/Model/IMDI/tags/cclm/cosmo_090213_4.8_clm17_cordex/util/running)
86    2014-03-14T06:41:13Z CMOR rewrote data to comply with CF standards and CORDEX requirements.";
87  :comment = "CORDEX Africa RCM CCLM 0.44 deg AFR-44";
88  :references = "http://www.clm-community.eu/";
89  :initialization_method = 1; // int
90  :physics_version = 1; // int
91  :tracking_id = "0493141f-def6-4377-bf92-329f2373cf23";
92  :title = "CLMcom-CCLM4-8-17 model output prepared for CORDEX RCP8.5";
93  :CORDEX_domain = "AFR-44";
94  :driving_experiment = "MPI-M-MPI-ESM-LR, rcp85, r1i1p1";
95  :driving_model_id = "MPI-M-MPI-ESM-LR";
96  :driving_model_ensemble_member = "r1i1p1";
97  :driving_experiment_name = "rcp85";
98  :rcm_version_id = "v1";
99  :product = "output";
100 :experiment = "RCP8.5";
101 :frequency = "mon";
102 :creation_date = "2014-03-14T06:41:13Z";
103 :Conventions = "CF-1.4";
104 :project_id = "CORDEX";
105 :table_id = "Table mon (Sept 2013) 1a8d24384e63c141a57dbedfd6710546";
106 :modeling_realm = "atmos";
107 :realization = 1; // int
108 :cmor_version = "2.9.1";
```

Figure 35: Fichier NetCDF partie 3

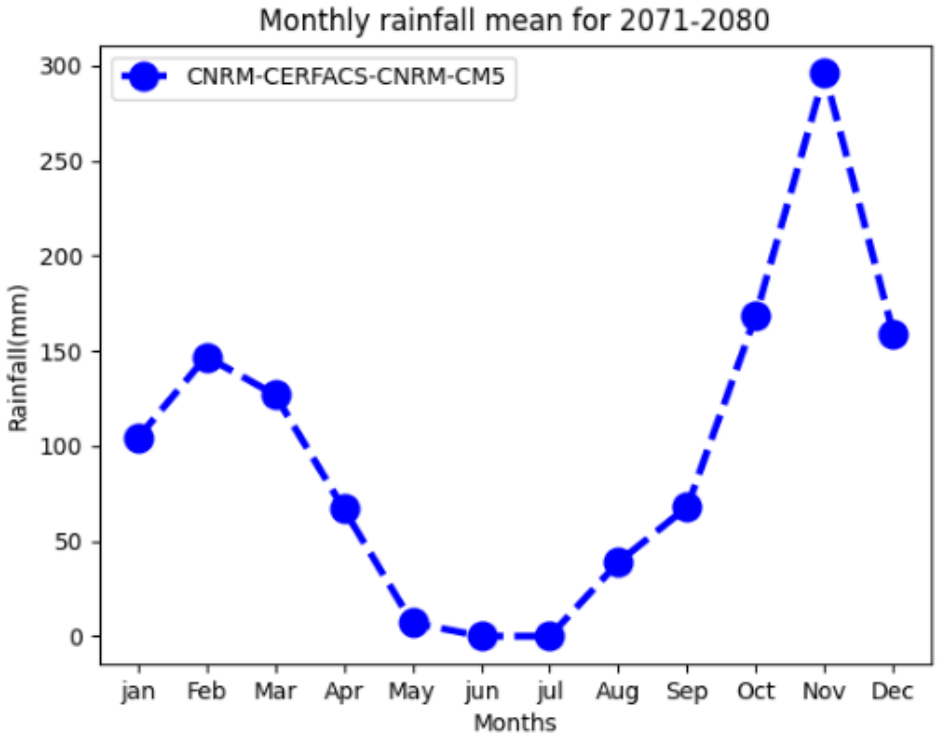

Figure 36: Moyenne des précipitations annuelles dans la région du Burundi,

Modèle : CNRM-CERFACS-CNRM-CM5

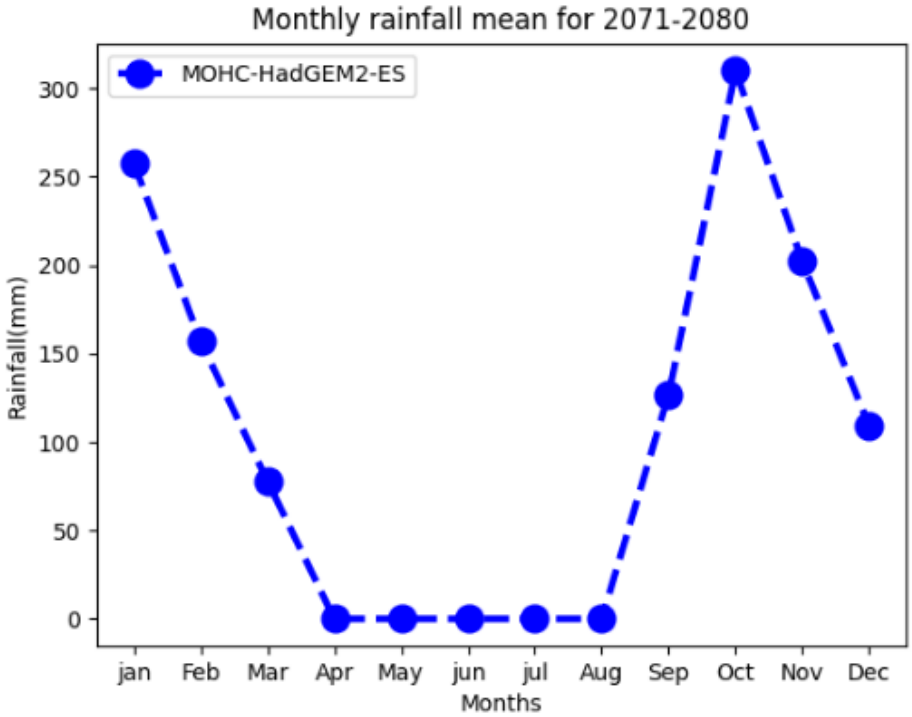

Figure 37:Moyenne des précipitations annuelles dans la région du Burundi,

Modèle : MOHC-HadGEM2-ES

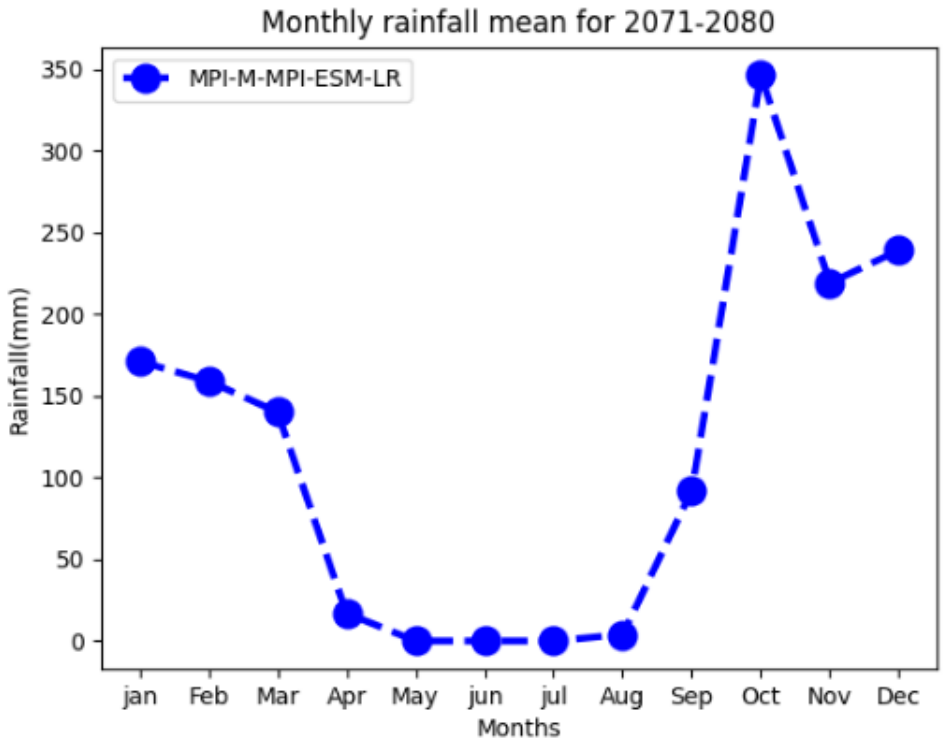

Figure 38: Moyenne des précipitations annuelles dans la région du Burundi, Modèle : MPI-M-MPI-ESM-LR

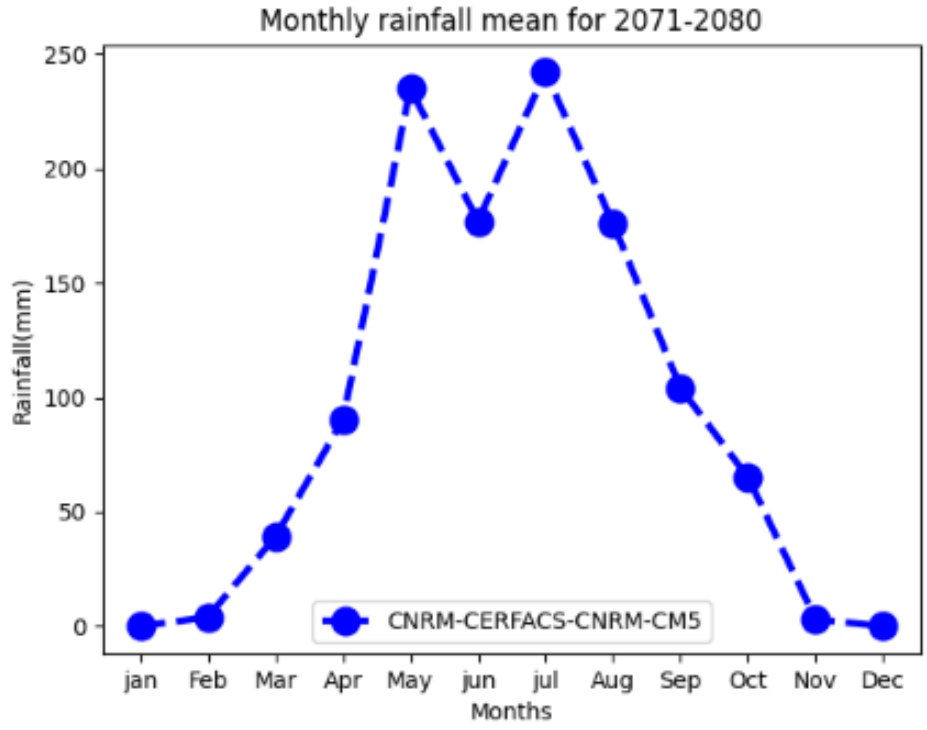

Figure 39:Moyenne des précipitations annuelles dans la région de l'Ethiopie, Modèle : CNRM-CERFACS-CNRM-CM5

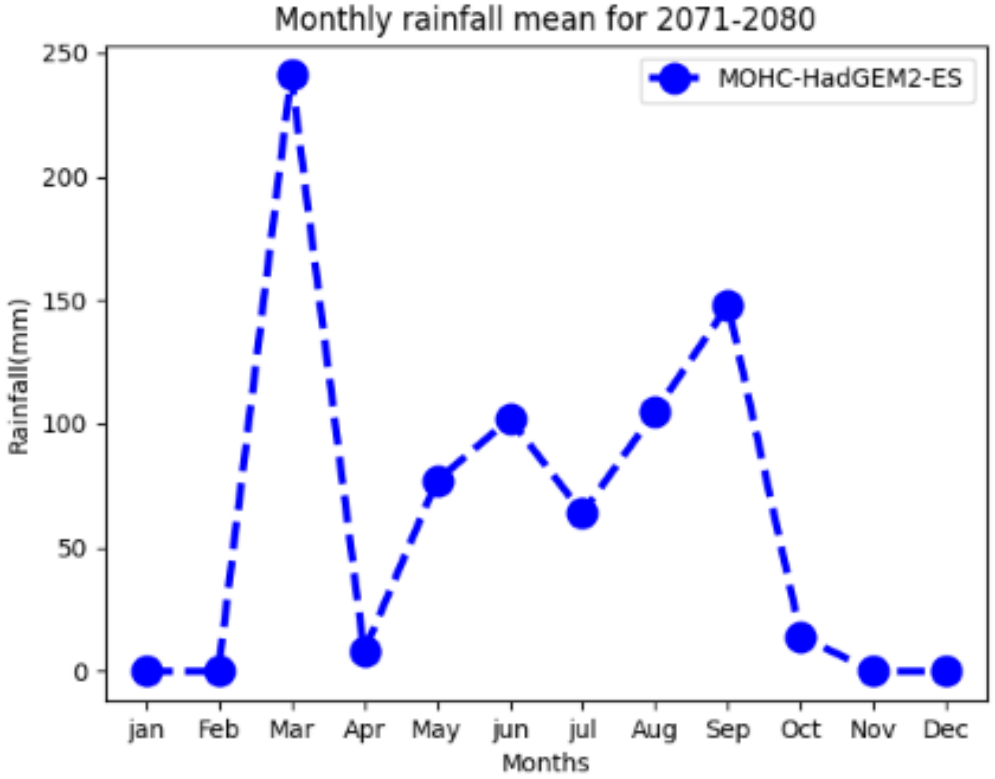

Figure 40: Moyenne des précipitations annuelles dans la région de l'Ethiopie, Modèle : MOHC-HadGEM2-ES

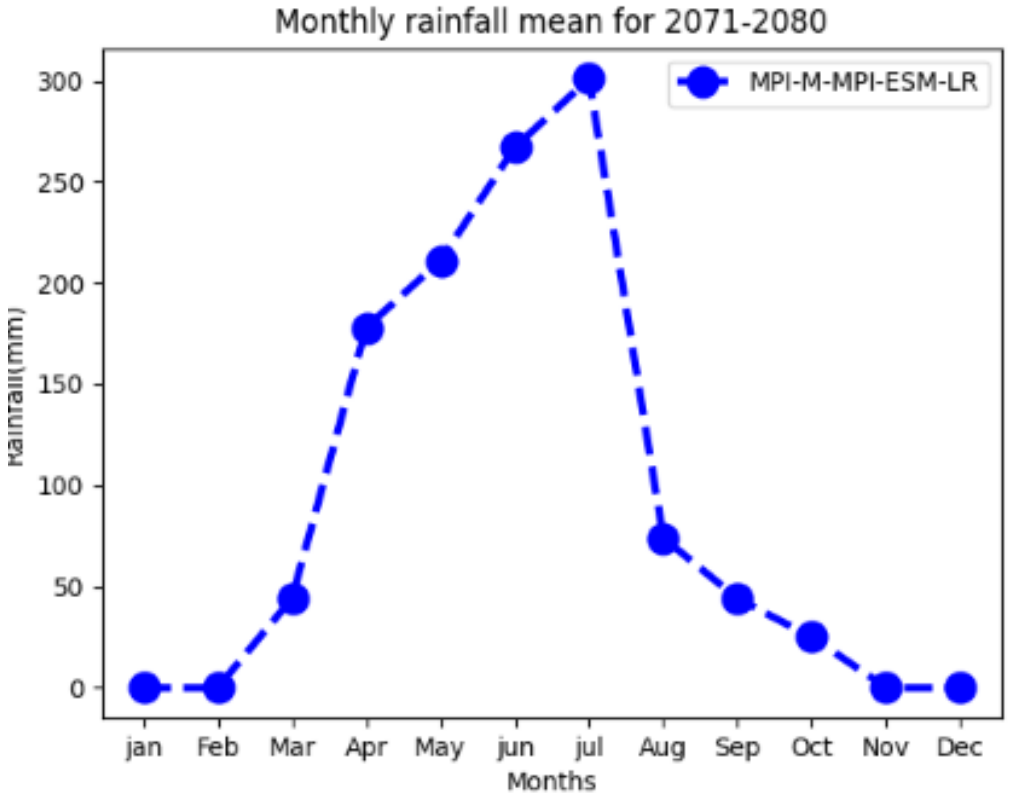

Figure 41: Moyenne des précipitation annuelles dans la région de l'Ethiopie, Modèle : MPI-M-MPI-ESM-LR

Références

(s.d.). Récupéré sur Leaflet: https://leafletjs.com/, consulté le 15/07/2021

About the SPEI. (s.d.). Récupéré sur Spei.csic: https://spei.csic.es/home.html

Addressing Climate Information Needs at the Regional Level: the CORDEX Framework. (s.d.). Récupéré sur https://public.wmo.int/en/bulletin/addressing-climate-information-needs-regional-level-cordex-framework

Brochure ESGF. (s.d.). Récupéré sur https://esgf.llnl.gov/esgf-media/pdf/2017-ESGF-Brochure.pdf

Chris Harrison, Z. Y. (s.d.). *Faster Progress Bars: Manipulating Perceived.* https://www.chrisharrison.net/projects/progressbars2/ProgressBarsHarrison.pdf.

De l'importance du temps dans l'UX. (s.d.). Récupéré sur https://www.altics.fr/de-limportance-du-temps-dans-lux/

données historique des précipitations. (s.d.). Récupéré sur Climat research unit: https://lr1.uea.ac.uk/cru/data

ESGF Node at DKRZ. (s.d.). Récupéré sur https://esgf-data.dkrz.de/projects/esgf-dkrz/

Hohenstein, J. (s.d.). *Shorter Wait Times: The Effects of Various Loading Screens on Perceived Performance.* https://www.researchgate.net/publication/302073992_Shorter_Wait_Times_The_Effects_of_Various_Loading_Screens_on_Perceived_Performance.

Hulme M, O. T. (s.d.). *Precipitation sensitivity to global warming: comparison of observations with HadCM2 simulations. Geophys. Res. Lett., 25, 3379-3382.*

Klimsec. (s.d.). Récupéré sur Kuleuven: https://ees.kuleuven.be/klimsec/

L'infrastructure de cache dans Django. (s.d.). Récupéré sur https://docs.djangoproject.com/fr/3.2/topics/cache/

L'accord de paris. (s.d.). Récupéré sur United Nations Climate Change: https://unfccc.int/fr/processus-et-reunions/l-accord-de-paris/l-accord-de-paris

M, H. (s.d.). *global land precipitation climatology for the evaluation of General Circulation Models. Climate Dynamics, 7, 57-72.*

Metadata for the change knowledge. (s.d.). Récupéré sur https://climateknowledgeportal.worldbank.org/themes/custom/wb_cckp/resources/data/CCKP_Metadata_Final_January2021.pdf

Modèle de circulation générale. (s.d.). Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Mod%C3%A8le_de_circulation_g%C3%A9n%C3%A9rale

Opendap. (s.d.). Récupéré sur https://www.opendap.org/

Ouranos. (s.d.). *Guide sur les scenarios climatique.* Récupéré sur https://www.ouranos.ca/wp-content/uploads/GuideScenarios2017_FR.pdf

Scenario RCP. (s.d.). Récupéré sur wikipedia: https://fr.wikipedia.org/wiki/Sc%C3%A9nario_RCP

WCRP Cordex. (s.d.). Récupéré sur Cordex: https://cordex.org/data-access/

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**