ENSG Géomatique

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

Université Gustave Eiffel

Ecole Nationale des Sciences Géographiques

Université Gustave Effeil

Master 2 Technologie des Systèmes d'Information,

Interface pour l'import de données météorologiques 4D au sein d'un environnement de co-visualisation

Cédric Périon, Félix Quinton
Thibault Petiet, Laura Wenclik

Mars - Avril 2021

☒ Non confidentiel ☐ Confidentiel IGN ☐ Confidentiel Industrie ☐ Jusqu'au ...

Ecole Nationale des Sciences Géographiques
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

Remerciements

Nous tenons tout d'abord à remercier notre commanditaire Jacques Gautier, membre de l'équipe GEOVIS du Laboratoire en Sciences et Technologies de l'Information Géographique (LASTIG). Sa disponibilité et ses conseils nous ont permis de mener à bien ce projet et d'avancer rapidement.

Nous remercions également Mathieu Brédif, pour son aide au cours du projet ; Louise G. et Florent G. pour la relecture de ce rapport..

Nous remercions l'École Nationale des Sciences Géographiques de nous avoir permis de mener à bien ce projet, notamment en nous offrant la possibilité de l'effectuer en présentiel.

Table des matières

Glossaire et sigles utiles

ENSG École Nationale des Sciences Géographiques.

TSI Technologies des Systèmes d'Information.

IGN Institut national de l'information géographique et forestière.

NetCDF Network Common Data Form, ce sont des fichiers contenant des valeurs de température pour plusieurs points et à différentes hauteurs. Ces fichiers sont transmis au LASTIG par Météo-France

LASTIG Laboratoire en Sciences et technologies de l'information géographique.

TEB Town Energy Balance.

Meso-NH non-hydrostatic mesoscale atmospheric model.

GLSL OpenGL Shading Language (GLSL) est un langage de programmation de shaders de haut-niveau dont la syntaxe est basée sur le langage C.

Git Git est un logiciel de gestion de versions décentralisé utilisé dans ce projet pour conserver un code cohérent.

Pousser du code Envoyer du code local sur un serveur de Git distant.

Commit Un commit est une commande git servant à enregistrer dans le dépôt les modifications apportées.

Sprint Un sprint informatique désigne le cycle de développement au cours duquel vont s'enchaîner un certain nombre de tâches pour, à terme, s'achever par la conception d'un produit final.

Introduction

L'urbanisation entraîne une densification de l'espace urbain qui, à terme, peut générer une augmentation de température dans les zones les plus denses. Ce phénomène nommé îlot de chaleur peut être néfaste pour les personnes vivant dans ces zones. Soit directement à cause de la chaleur créée, soit indirectement en engendrant une mauvaise dissipation des polluants par exemple. Différents facteurs sont responsables de la création d'îlots de chaleur comme la morphologie des bâtiments ou l'occupation des sols.

Un prototype d'environnement de visualisation 3D axé sur la visualisation spatiale et temporelle de données météorologiques et urbaines est donc en développement au LASTIG sous forme d'application web.

L'objectif principal de cette interface est de permettre la co-visualisation de données urbaines (le bâti) avec des données météo 3D (température de l'air) et de permettre une exploitation visuelle simultanée de ces deux composantes. Avant le début de notre projet l'interface permettait déjà de co-visualiser et d'explorer ces données mais seulement pour un moment donné.

L'objectif principal de notre projet a donc été de permettre la visualisation de la composante temporelle des données météo au sein de l'interface. Cet objectif a nécessité de remplir différentes sous-tâches. Puis, ayant atteint cet objectif principal, nous nous sommes consacrés à des objectifs secondaires.

Nous consacrerons donc la première partie de notre rapport à l'analyse de notre sujet en présentant ses objectifs ainsi que notre vision du projet. Puis nous nous consacrerons à la phase d'import et de sélection de données correspondant à différents instant au sein de l'application afin d'ajouter une composante temporelle à l'application. Ensuite nous aborderons la partie concernant la visualisation de la cette composante temporelle au sein de l'application. Après cela nous discuterons des bonus et tâches complémentaires sur lesquelles nous avons également travaillé en fin de projet. Enfin nous finirons par une partie sur la gestion et l'organisation de notre projet.

Chapitre 1 Analyse du projet

1.1 Présentation de l'application

L'application sur laquelle nous avons travaillé est une application web développée au LASTIG depuis environ un an. De nombreuses fonctionnalités y étaient donc déjà implémentées.

Dans cette application, on représente un modèle urbain 3D, où chaque bâtiment est représenté par un polyèdre assez abstrait, basé sur l'empreinte au sol du bâtiment et sa hauteur. En plus de cela, on représente également des données de température de l'air simulées, qui se présentent sous la forme d'une maille 3D irrégulière (chaque dalle de la maille ayant pour dimension moyenne 625m x 625m), et qui sont représentées projetées sur des géométries (nuages de points, plans horizontaux, plans verticaux basés sur le réseau routier) en utilisant une échelle de couleur.

Les données de température sont donc stockées sous forme de maille 3D. Dans un même fichier on va donc avoir pour chaque point (x,y) de la maille des informations de température associées à ce point. Ces températures sont au nombre de 37 et correspondent aux températures au niveau du point pour 37 hauteurs différentes. Ces 37 niveaux sont séparés en deux groupes, les niveaux TEB et les niveaux Meso-NH. Les 6 valeurs de températures les plus basses, correspondent aux niveaux dits TEB numérotés de 1 à 6 qui s'élèvent de 1 à 60 mètres. Les 31 valeurs restantes correspondent aux niveaux Meso-NH qui s'élèvent de 130 à 15 000 mètres. Il est important de noter que toutes les données stockées dans un même fichier on été recueillies au même moment. On possède donc un fichier par instant temporel donné.

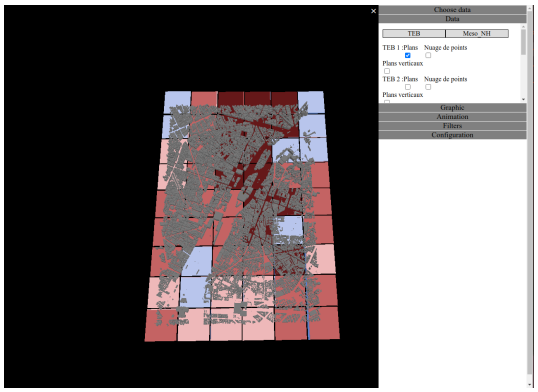

Figure 1.1 : Affichage de plan TEB 1 sous forme de quadrillage sur le centre de Paris

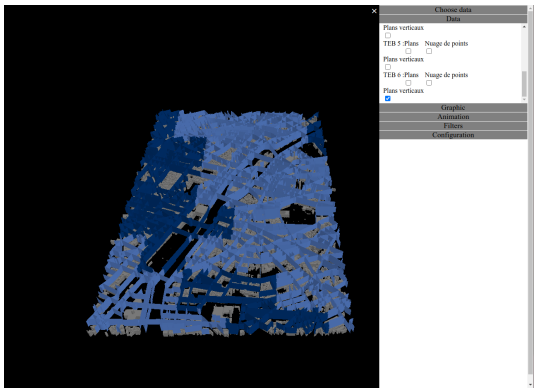

Figure 1.2 : Affichage de plan TEB 6 sous forme de plan verticaux sur le centre de Paris

Les figures 1.1 et 1.2 montrent les rendus obtenus suite à l'affichage de données sous forme de plans horizontaux et verticaux dans le centre de Paris en août 2003.

D'autres fonctionnalités ont également été développées mais nous ne prendrons pas le temps de les détailler ici, étant donné que saisir leurs fonctionnements n'est pas nécessaire à la compréhension de notre projet.

Vous trouverez en annexe A.1 du projet une maquette générale de l'application comme elle était initialement, permettant de mieux comprendre comment elle s'articule.

1.2 Analyse du sujet

1.2.1 Analyse du besoin

Afin de permettre la visualisation de de la composante temporelle des données utilisées dans l'application nous avons dû réaliser plusieurs tâches séparées en deux grandes parties que nous allons détailler ci-dessous.

Import et sélection de données correspondant à différents instants temporels

Les données correspondant à chaque moment temporel sont stockés sous le format NetCDF. Cependant les données chargés dans l'application au début du projet étaient au format CSV, un pré-traitement était donc nécessaire afin de pouvoir passer d'un format à l'autre. Pour pouvoir représenter plusieurs instants temporels dans l'application, il est donc nécessaire de s'affranchir de cette phase de pré-traitement des données et d'intégrer directement les données au format NetCDf au sein de l'application.

Les données pré-traitées correspondant à une sélection spatiale des données de base, permettant de travailler sur de plus petites emprises, il a été nécessaire d'intégrer à l'interface un moyen de sélectionner l'emprise des données à extraire du NetCDF afin que le navigateur ne soit pas surchargé et que l'utilisateur puisse ne travailler qu'avec la zone qui l'intéresse réellement.

Une fois l'import des données sous format NetCDF fonctionnel, il a fallu permettre d'intégrer des données correspondant à différents instants temporels, afin de visualiser l'évolution des températures au cours du temps, et donc pouvoir charger plusieurs fichiers NetCDF à la fois. Suite à cela, on a naturellement eut besoin de fournir à l'utilisateur un moyen de pouvoir choisir quelle date afficher dans la fenêtre de visualisation afin de lui permettre de naviguer entre les différents fichiers chargés.

Afin d'aider l'utilisateur dans son exploration de la composante temporelle des données, il faut lui proposer, en plus de la représentation 3D qui montre les données à un instant T, la possibilité de modifier cet instant T en affichant un autre fichier NetCDF.

Visualisation temporelle

La seconde partie de ce projet avait quant à elle pour but d'aider l'utilisateur dans son exploration de la composante temporelle des données. Il a donc fallu lui proposer, en plus de la représentation 3D qui montre les données à un instant t, une représentation temporelle présentant une vue d'ensemble des données sur toute leur étendue temporelle. Cette représentation temporelle de l'ensemble des données a pris la forme de deux graphiques 2D et 3D.

Représentation des valeurs de température interpolées

Un objectif bonus de ce projet et indépendant de l'objectif principal était de représenter les valeurs de température des plan verticaux de manière continue et non plus discrète comme c'était le cas jusqu'ici.

1.2.2 Cas d'utilisation

Nous allons ici détailler les différents comportements attendus de la part d'un utilisateur se servant des fonctionnalités que nous aurons implémentées.

Dans un premier temps, l'utilisateur va être invité à charger un ou plusieurs fichiers NetCDF. Suite à cela il lui sera proposé de sélectionner l'emprise géographique qui l'intéresse dans les fichiers chargés.

Si plusieurs fichiers ont été chargés, il pourra alors sélectionner le fichier qui l'intéresse afin de l'afficher à l'écran.

En parallèle une zone de l'écran sera dédiée à la représentation temporelle de tous les fichiers chargés. L'utilisateur pourra alors naviguer entre représentation temporelle 2D et 3D et pourra directement y sélectionner le fichier à afficher dans la zone principale de l'écran.

Si l'utilisateur souhaite afficher les plan verticaux correspondants aux températures, il doit pouvoir choisir entre une représentation de la température discrète ou continue.

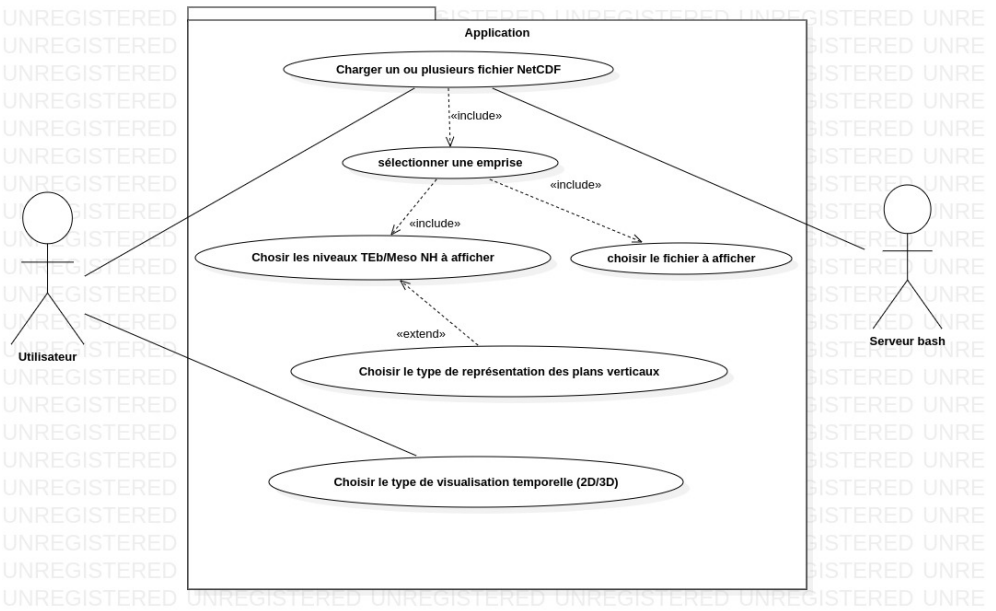

FIGURE 1.3 : Diagramme de cas d'utilisation

1.2.3 Diagramme d'activité

Par la suite, nous avons détaillé le fonctionnement de l'application dans la limitation de notre projet et des objectifs demandés. Après l'analyse des objectifs nous avons pu déterminer le fonctionnement suivant.

Dans un premier temps, il est possible de choisir simultanément un ou plusieurs fichiers NetCDF correspondant à des dates différentes ainsi que les fichiers des routes et des bâtis. Les données NetCDF sont par la suite chargées dans le prototype.

A partir de l'emprise initiale récupérée dans les fichiers, l'utilisateur peut choisir de réduire ou non l'emprise affichée à l'écran. Les données NetCDF, bâtis et routes sont ainsi filtrées selon la nouvelle emprise choisie par l'utilisateur.

Une fois les données filtrés, les bâtiments vont être affichés dans le prototype. Il est ensuite possible d'afficher les données météorologiques TEB et Meso-NH (plan, nuage de point et plan verticaux). Par défaut, les données affichées sont celles du fichier stockant les informations les plus anciennes (chronologiquement). Il est possible de changer de fichier affiché à l'écran de différentes manières. La première manière est de sélectionner le fichier voulu à l'aide d'un menu déroulant contenant la liste des dates de chaque fichier chargé.

En parallèle de l'affichage spatiale, les graphiques temporelles 2D et 3D vont se dessiner. L'utilisateur a le choix d'afficher l'un ou l'autre. De plus, les graphiques sont dynamiques, l'utilisateur peut changer de date directement en cliquant sur une donnée sur l'un des diagrammes.

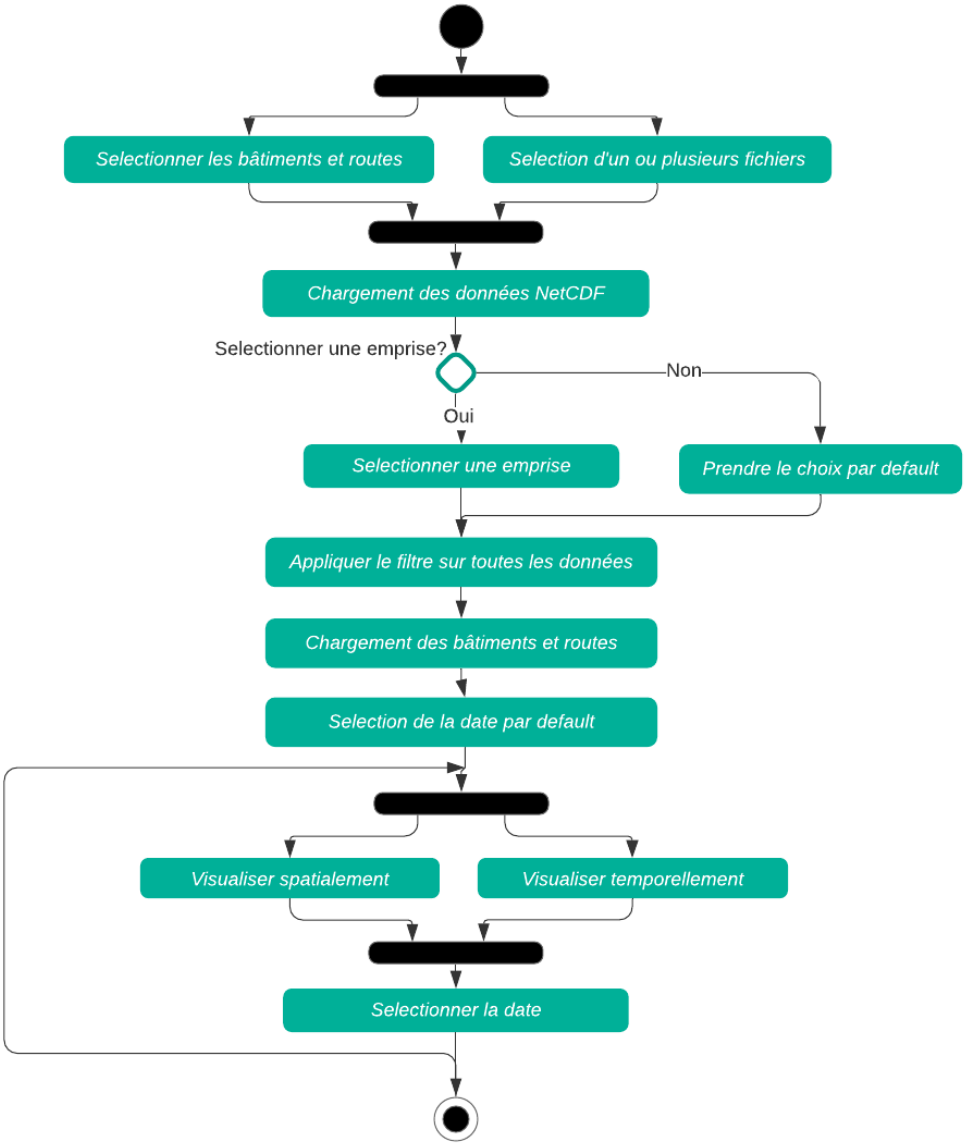

FIGURE 1.4 : Diagramme d'activité

1.3 Les maquettes de l'application

Dans ce genre de projets, les échanges entre l'équipe et le commanditaire peuvent ne pas être clairs. Les deux parties peuvent donc ne pas avoir la même vision du projet et le résultat risque donc d'être différent de ce qui est attendu par le commanditaire.

Il est donc intéressant de présenter plusieurs maquettes au commanditaire qui vont permettre de confronter les visions de l'application des deux parties. Faire ce travail permet de s'assurer que le résultat sera conforme aux attentes du client afin de ne pas avoir à retravailler les rendus.

Nous avons ainsi pu proposer plusieurs maquettes à notre client afin de répondre au mieux à ses besoins.

La sélection de l'emprise

Dans le cadre du projet, le client souhaitait pouvoir choisir l'emprise des données à afficher. Nous avons donc réfléchi à deux solutions différentes.

La première méthode consistait à afficher l'emprise des fichiers dans une carte Leaflet en permettant à l'utilisateur de faire varier cette emprise.

La seconde méthode consistait à afficher le bâti dans la partie principale de l'application puis à n'en sélectionner qu'une partie à partir de curseurs situés dans le menu.

Après discussion avec notre commanditaire, la carte nous a semblé être la meilleure solution. En effet, elle permet de sélectionner de façon précise les données. Elle permet également de visualiser le contexte spatiale dans lequel se trouve nos données. La seconde méthode est plus complexe à mettre en oeuvre, plus coûteuse en ressources et nécessite d'avoir chargé un fichier de bâti au préalable.

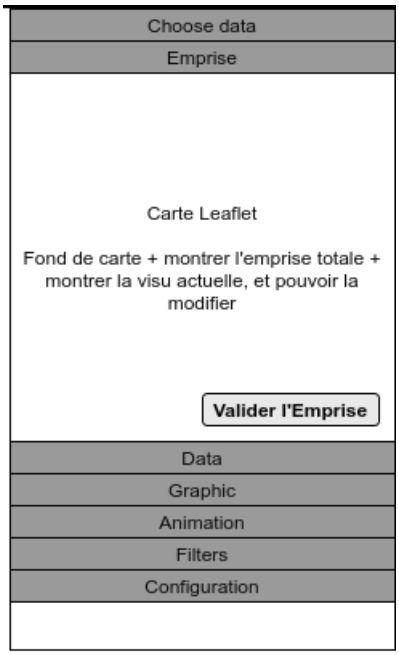

FIGURE 1.5 : Proposition de sélection d'emprise à partir d'une carte leaflet.

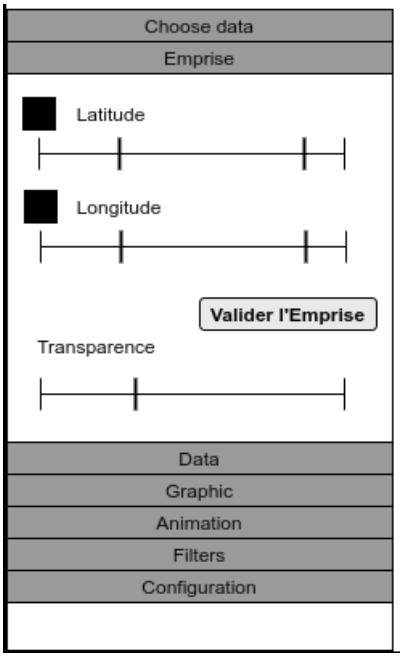

FIGURE 1.6 : Proposition de sélection d'emprise à partir du bâti.

La visualisation temporelle

L'un des besoins du client était d'ajouter une composante de visualisation temporelle au prototype. Pour cela, il a été nécessaire de déterminer un emplacement pour les visualisations temporelles 2D et 3D.

Dans un premier temps nous avons proposé deux emplacements à notre client.

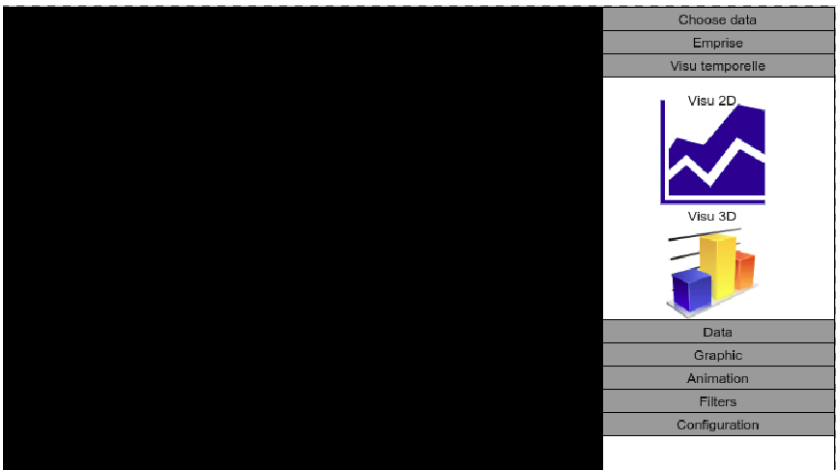

FIGURE 1.7 : Proposition pour la vue temporelle 1

La première version consistait à afficher directement les diagramme dans une partie dédié à cela du menu.

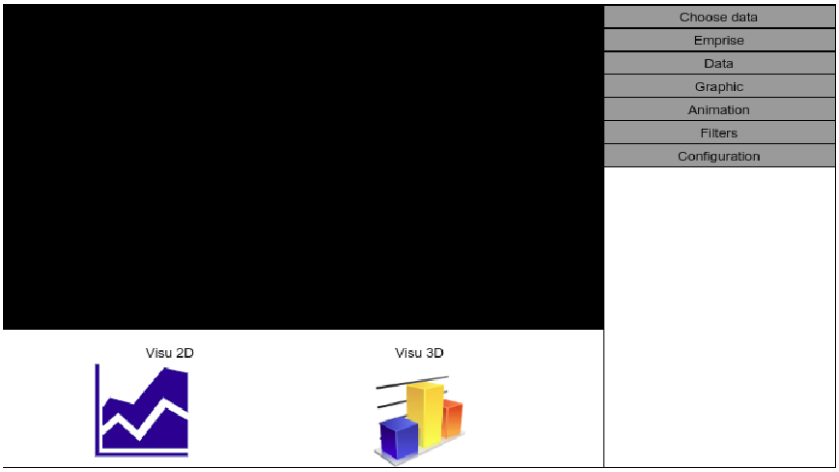

FIGURE 1.8 : Proposition pour la vue temporelle 2

Dans la seconde version, il s'agissait de créer une section contenant les deux graphiques situés sous la partie de visualisation spatiale des données. Cette section empiétant sur la partie principale de l'application, il était donc nécessaire que sa taille ne soit pas fixe afin que l'utilisateur puisse la faire coulisser de haut en bas.

Après échanges avec le client, nous avons choisi de garder la seconde solution car elle offre à l'utilisateur une meilleure expérience de visualisation. Cependant, il a été décidé de ne pas afficher simultanément les deux diagrammes et de choisir lequel afficher à l'aide d'un bouton afin de ne pas noyer l'utilisateur sous un flot d'informations comme on peut le voir dans la version finale (3.1 et 3.4).

Chapitre 2 Import et sélection de données correspondant à différents instants temporels

Dans cette partie, nous décrirons l'import et la sélection de données correspondant à différents instants temporels. Afin de résoudre l'objectif principal du projet, nous avons procédé par étape. Dans un premier temps, nous avons permis l'import direct d'un fichier NetCDF dans l'application. Puis nous avons ajouté une sélection de l'emprise spatiale à représenter. Enfin, nous avons permis le chargement de plusieurs fichiers, pour en extraire la composante temporelle et ainsi permettre la sélection du moment à représenter.

2.1 Import d'un fichier NetCDF

2.1.1 Contexte

Lorsque nous avons pris l'application en main, elle n'était faite que pour pouvoir importer des fichiers CSV contenant les données météorologiques étudiées. Si le format CSV était utilisé jusqu'ici c'est grâce à sa simplicité d'utilisation, il suffisait d'avoir un fichier contenant pour chaque point : sa position en latitude/longitude, la hauteur de chacun de ses 6 niveaux TEB avec leurs températures associées ainsi que la température de chaque niveau Meso-NH et l'altitude au sol de la cellule permettant de calculer la hauteur des niveaux Meso-NH. Cependant le format CSV n'est pas le format utilisé par METEO FRANCE, une intervention humaine était donc jusqu'ici nécessaire pour pouvoir convertir les données de base en fichiers CSV. Cette manipulation pouvant vite devenir gênante et chronophage, il était important de pouvoir se passer de cette conversion de format et d'importer directement des fichiers au format de base des données, le format NetCDF.

NetCDF signifie Network Common Data Form, c'est un format dit "auto-documenté", autrement dit, les fichiers NetCDF sont séparés en deux parties l'en-tête (header) et le corps (body). L'en-tête décrit le disposition du fichiers, le type de données représentés, leurs unités et leurs positions dans le corps. Le corps lui va stocker les données. Les données du corps étant stockées sous forme de tableaux, l'intérêt de ce format est de pouvoir accéder rapidement à un tableau du corps à partir des données de l'en-tête.

Voici par exemple un fichier NetCDF ouvert avec le logiciel Panoply permettant de visualiser l'en-tête d'un fichier NetCDF. Comme on peut le voir sur cette image, de nombreux champs sont présent dans l'en-tête, il est donc important de ne garder que les champs qui vont être utilisés par l'utilisateur afin de ne pas charger de données inutiles. Ici les champs qui nous intéresseront seront ceux de latitude/longitude, ceux de température/hauteur des niveaux TEB/Meso-NH et ceux de date.

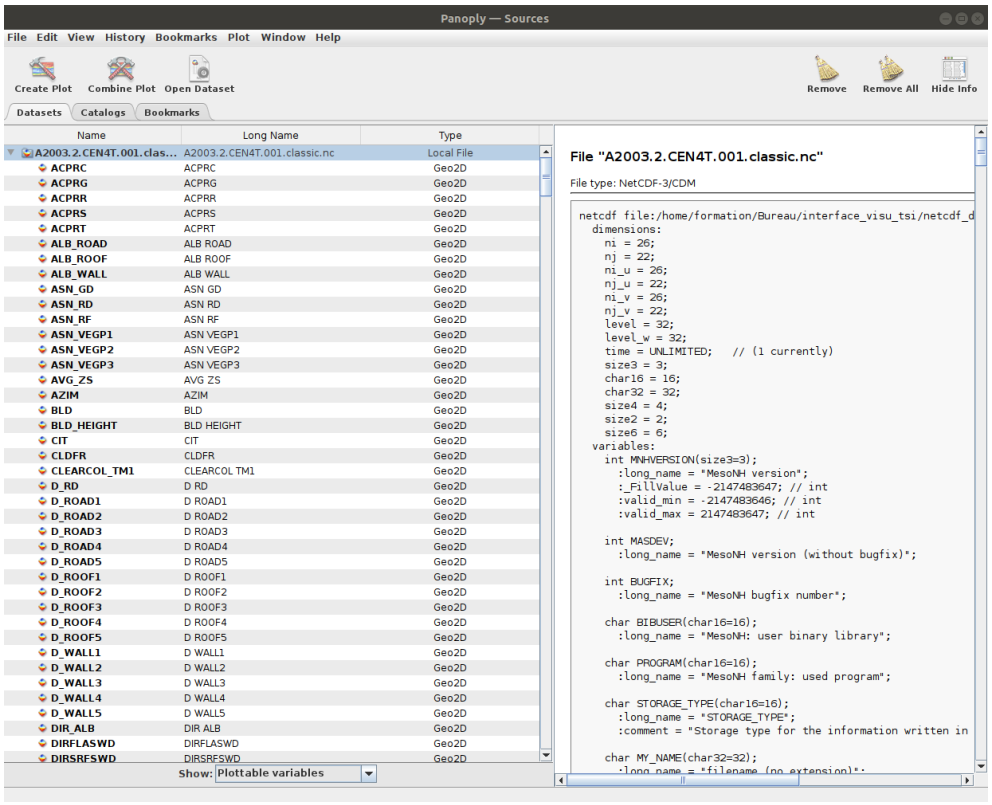

FIGURE 2.1 : Exemple d'en-tête de fichier NetCDF ouvert avec Panoply

2.1.2 Implémentation

Si le format NetCDF semble être très pratique pour stocker des données, il ne l'est pas pour en extraire. C'est pourquoi une tâche aussi anodine que changer le format des fichiers en entrée peut vite devenir assez complexe. Le format NetCDF n'est pas un format que l'on rencontre souvent, et il n'existe donc pas forcément de librairies permettant d'interagir de façon simple avec des fichiers NetCDF. En ce qui concerne le JavaScript la librairie NetCDFJs permet d'ouvrir des fichiers NetCDF en version "classic", cependant les données que nous utilisons dans l'application sont en version NetCDF 4. Changer de format est donc nécessaire afin de pouvoir utiliser cette librairie. La librairie NetCDFJS étant très rudimentaire elle ne permet pas de faire cette conversion, nous avons donc dû mettre en place un serveur exécutant un script bash qui va permettre de changer de version de fichiers NetCDF. Suite à cela on va ensuite pouvoir extraire les données des fichiers NetCDF à l'aide de la librairie NetCDFJS.

2.2 La composante spatiale des données

2.2.1 Problématique de la sélection spatiale

Comme nous l'avons dis précédemment, notre commanditaire avait émis, dans les principales tâches à effectuer, la création d'une interface permettant de sélectionner graphiquement, au sein du fichier NetCDF, les données à visualiser selon une emprise 3D.

Avec le chargement d'un fichier NetCDF directement depuis l'application, une problématique apparaît donc sur l'emprise des données que l'on veut regarder : les fichiers contiennent des données sur tout Paris et les départements limitrophes. L'utilisateur n'a pas nécessairement besoin de toutes ces données. Il peut vouloir se concentrer sur un ou plusieurs arrondissements ou communes. De plus, l'ajout d'une sélection spatiale permet de libérer la mémoire en supprimant toutes les données

en dehors de celle-ci ce qui améliore grandement le confort de l'utilisation en fluidifiant toutes les actions.

La version du prototype qui nous a été fournie au début du projet incluait une sélection des niveaux TEB et Meso-NH à représenter. Il s'agit de la partie que l'on peut le voir sur la figure 2.2.

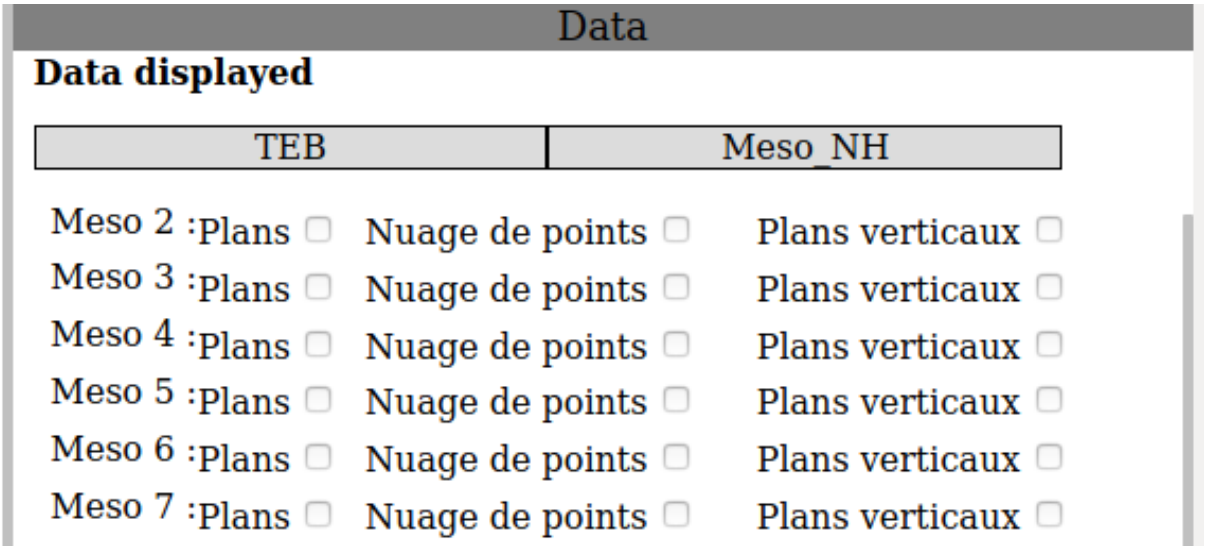

FIGURE 2.2 : Sélection des niveaux Meso NH

Ainsi, la sélection de l'emprise en hauteur étant déjà faite, charge à nous de créer une interface pour la sélection en deux dimensions.

2.2.2 Solution retenue

Comme vous avez pu le voir dans les maquettes de la partie 1.3, nous avons retenu la première solution : une carte.

Ainsi, nous avons choisi d'utiliser la librairie Leaflet pour créer et interagir avec la carte. Cette librairie permet d'afficher une carte, d'en ajouter des points ou des marqueurs. Ainsi, comme vous pouvez le voir sur l'image 2.3, nous avons décidé de représenter deux informations sur la carte. Chaque point du NetCDF par un cercle bleu qui devient rouge lorsque il sort de l'emprise. Deux marqueurs au coin sud-ouest et nord-est peuvent être déplacer, afin de sélectionner l'emprise voulue.

2.2.3 Filtrage des données

Suite à cette sélection, les données peuvent facilement être filtrées. On garde les points appartenant à l'emprise. La difficulté principale de cette partie fut la sélection de la totalité des points d'une ligne ou d'une colonne. En effet, sur une même ligne, les points n'ont pas exactement la même latitude. Il arrive qu'un point sorte de l'emprise alors qu'il doit être affiché. Comme c'est le cas de la figure 2.4. Certains points sont en bleu et d'autre en rouge, dû au fait que les latitudes ne sont pas exactement les mêmes. Dans ce cas, nous avons décidé de prendre tous les points d'une même ligne (ou d'une même colonne) si au moins un point est sélectionné.

Pour le filtrage des bâtiments et des routes, l'emprise qui est prise en compte n'est pas exactement celle choisie par l'utilisateur. En effet, le découpage s'effectue à la limite des dalles des points extérieurs afin que les bâtiments et les routes s'arrêtent à la frontière des dalles de données, pour un rendu plus juste et plus esthétique.

Pour le fichier de bâtiment, nous avons convenu que tout bâtiment était dans l'emprise si son barycentre faisait partie de l'emprise.

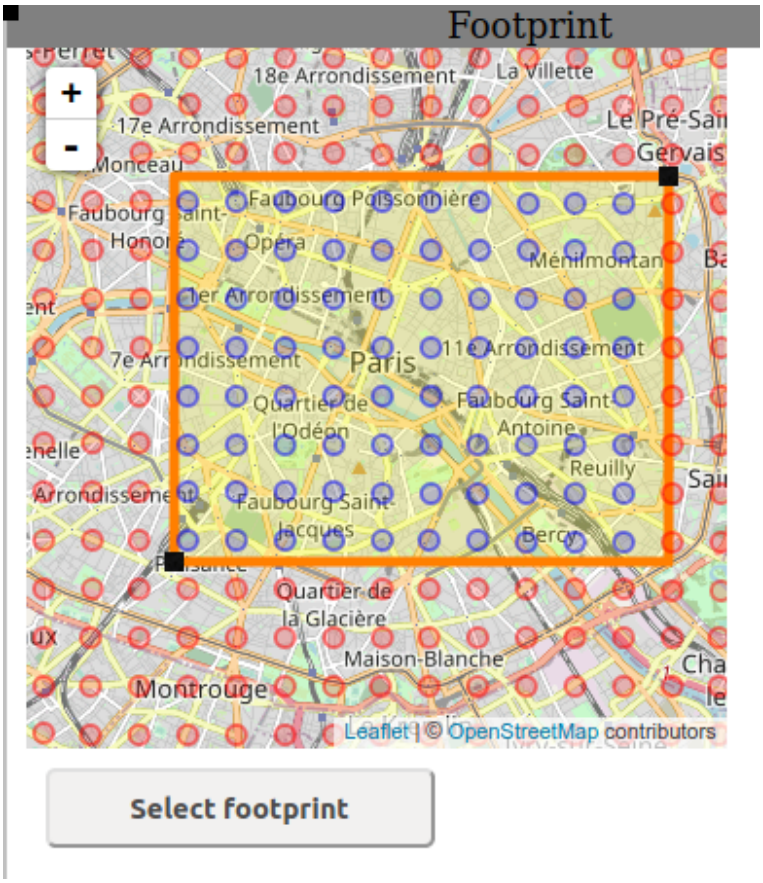

FIGURE 2.3 : Interface pour la sélection de l'emprise

FIGURE 2.4 : Exemple d'un problème de sélection

Pour le fichier des routes, il s'agit d'une liste de points appartenant à un tronçon. Ainsi, on garde simplement les points faisant partie de l'emprise.

2.2.4 Reprojection des données

Les données fournies dans les CSV étaient en Lambert 93 et le prototype gère des données en mètres. Or les données du NetCDF sont en WGS 84 c'est à dire dans un système de projection en latitude, longitude. Ainsi, après avoir chargé et filtré les données, il faut effectuer une reprojection des données. Pour ce faire, nous utilisons la librairie proj4js.

2.3 Chargement de fichiers multiples

Si l'un des buts de cette application est de permettre une visualisation spatiale des données, un autre est de permettre une visualisation temporelle des données. Pour cela il est donc nécessaire de pouvoir charger plusieurs fichiers NetCDF d'un coup car chaque fichier correspond à un instant, ou à une date précise. Charger plusieurs fichiers nécessite donc de faire plusieurs appels au serveur et de restructurer l'application afin qu'elle soit capable de gérer la présence de plusieurs fichiers. Pouvoir

importer plusieurs fichiers contenant des données localisées sur une même zone va alors permettre à l'utilisateur de pouvoir visualiser l'évolution temporelle de la température en milieu urbain.

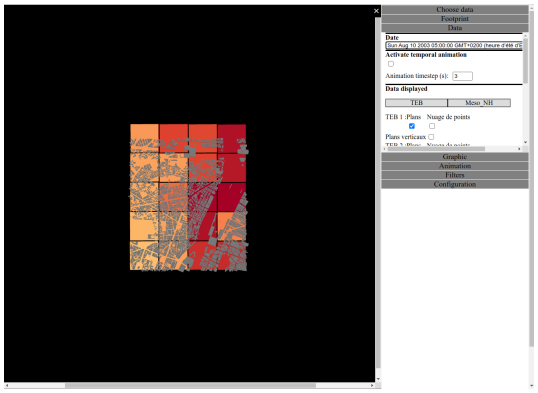

FIGURE 2.5 : Température dans Paris centre le 10/08/2003 à 5h

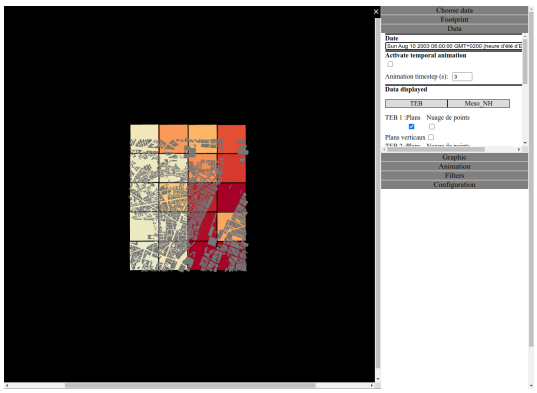

FIGURE 2.6 : Température dans Paris centre le 10/08/2003 à 8h

On peut donc observer ici l'évolution de la température dans le centre de Paris (pour le niveau TEB 1) le 10 août 2003 pendant la matinée.

Les autres fonctionnalités que nous avons implémentées tout au long de ce projet vont d'ailleurs également aller dans ce sens. La représentation temporelle des données étant un enjeux majeur pour les météorologues.

2.4 Piste d'amélioration du serveur

Actuellement le serveur utilise un script bash qui renvoie tous les fichiers NetCDF convertis au côté client qui va ensuite extraire les données qui l'intéresse à l'aide de la librairie NetCDFJS puis qui va redécouper chaque fichier en fonction de l'emprise sélectionnée.

Une façon plus optimale de faire les choses pourrait être par exemple d'utiliser la librairie Python NetCDF4 offrant plus de fonctionnalités que NetCDFJS. On peut alors imaginer l'enchaînement d'étapes suivante :

1. Demande des fichiers NetCDF au serveur.
2. Extraction des champs utilisés (températures, hauteurs...).
3. Envoi des données d'un fichier au client.
4. Sélection de la nouvelle emprise par l'utilisateur.
5. Envoi des coordonnées de la nouvelle emprise au serveur.
6. Sélection de la nouvelle emprise sur tous les fichiers.
7. Envoi des données de chaque fichier au client.

Chapitre 3
Visualisation temporelle

Afin de remplir l'objectif principal du projet, nous devions ajouter d'autres visualisations représentant cette fois l'évolution de la température dans le temps, avec en entrée un fichier NetCDF par instant temporel. Ainsi il a fallu concevoir et développer des outils de visualisation qui intègrent la dimension temporelle des données.

Nous avons proposé deux solutions. Premièrement,un graphe en 2 dimensions qui représente l'évolution de la température en fonction du temps avec des courbes de différentes couleurs pour représenter les différents niveaux TEB. En parallèle nous avons proposé un diagramme en 3 dimensions qui représente la température en Z et avec la couleur, en fonction du niveau TEB et de la date.

3.1 Visualisation temporelle 2D

Afin d' intégrer la dimension temporelle, il nous a été demandé de mettre en place un diagramme 2D. Le but de cette visualisation est de pouvoir afficher, pour chacun des fichiers chargés, les températures moyennes de chaque niveau TEB et la moyenne de tous les niveaux TEB.

Le principal objectif de cette visualisation est de pouvoir trouver les dates les plus intéressantes, afin de les afficher dans la visualisation spatiale 3D. Ainsi, le graphique devrait être dynamique. L'utilisateur devait pouvoir y sélectionner un point et changer directement de fichier affiché dans la visualisation spatiale.

3.1.1 Technologies utilisées

Afin de dessiner le graphique la librairie Chart.js a été utilisé.

La librairie Chart.js est une bibliothèque Java Script open source gratuite pour la visualisation de données sous forme graphique. Cette bibliothèque permettant une prise en main rapide d'outils de visualisation 2D fut très adaptée a nos besoins. Nous avons pu ainsi grâce à cette bibliothèque créer un diagramme 2D dynamique très facilement, les événements de souris y étant déjà implémentés.

3.1.2 Résultat

La figure 3.1 présente ainsi la visualisation temporelle 2D pour 8 moments différents s'étalant sur la journée du 8 août 2003.

Le diagramme représente les températures des différents niveaux TEB en fonction des différentes dates ainsi qu'une moyenne de tous les niveaux TEB. Il est aussi possible de choisir quels niveaux l'utilisateur souhaite visualiser dans le diagramme. Cela s'effectue en cliquant sur la légende de la courbe.

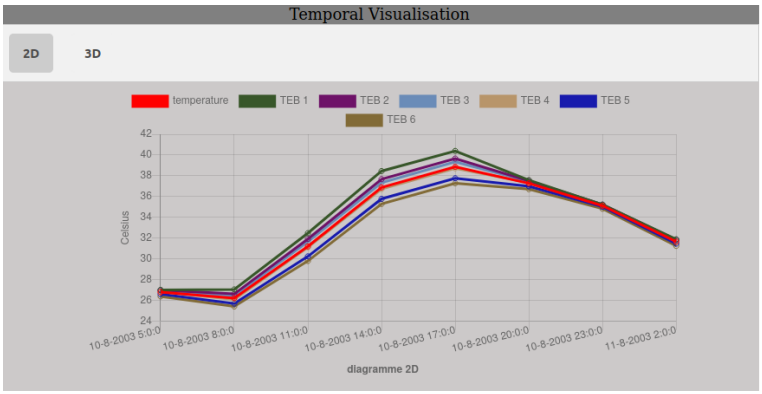

FIGURE 3.1 : Affichage final de la visualisation temporelle 2D

3.2 Visualisation temporelle 3D

3.2.1 Présentation du diagramme

Pour la visualisation 3D nous avions 3 variables à représenter :

- Niveau TEB
- Instant
- Température

Nous nous sommes inspirés de visualisations temporelles 3D qui existaient déjà [2]. La solution que nous avons dégagée est une représentation en histogramme en trois dimension avec les paramètres suivants :

- Sur l'axe rouge, la coordonnée x du pied de la barre représente le $x^{ième}$ niveau TEB
- Sur l'axe bleue, la coordonnée y du pied de la barre représente la $y^{ième}$ date des données en entré
- Sur l'axe vert, la hauteur de la barre au dessus de son pied représente la température moyenne d'un niveau TEB donné à une date donnée
- La couleur de la barre représente également la température comme dans la représentation spatiale des données.

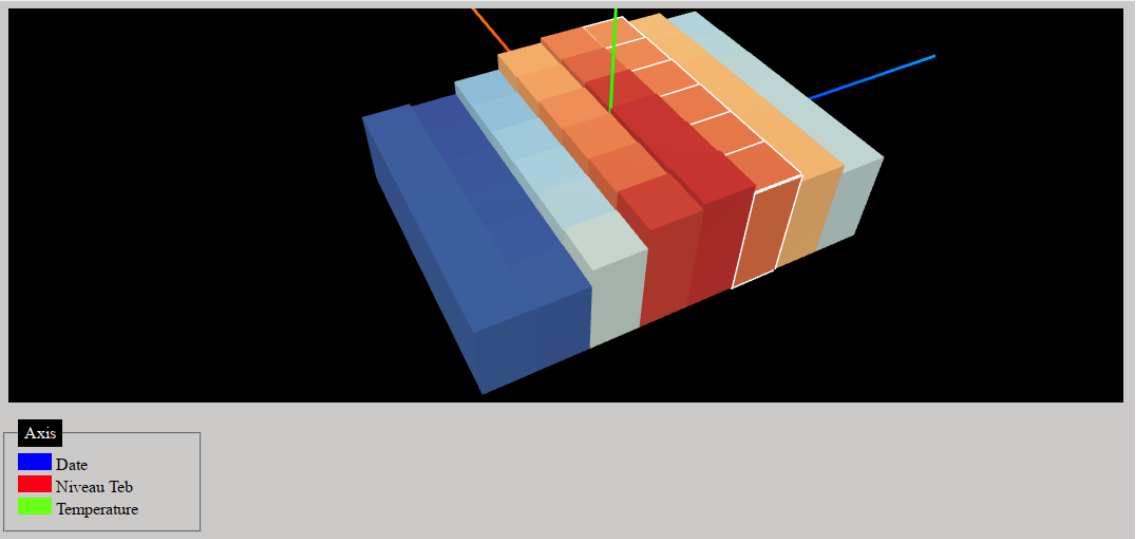

FIGURE 3.2 : Affichage final de la visualisation temporelle 3D

Nous avons donc agrégé les données sur les dimensions longitude et latitude en faisant une moyenne des températures sur toutes les dalles à un niveau fixé.

Il faut noter que la coordonnée x du pied de la barre représente le niveau TEB. Or la hauteur de celui-ci dépend des coordonnées planimétriques des points O. Ainsi la coordonnée x du pied de barre dans notre histogramme ne représente pas strictement une hauteur au-dessus du sol précise mais plutôt l'indice d'un niveau TEB.

Même si la couleur est un paramètre que nous pouvions utiliser pour représenter une autre dimension de notre donnée, il s'avère en pratique difficile de trouver une représentation de la donnée lisible en utilisant la couleur pour représenter une autre dimension. Cela est entre autre du à des problèmes d'occlusion 3D.

Pour rendre cette visualisation utile à la sélection des dates nous avons fait le choix de permettre à l'utilisateur de pouvoir explorer le diagramme et sélectionner une date en utilisant la souris. La date des données sous le curseur est affichée, les autres barres de la même date sont mises en évidence. En cliquant sur le diagramme, l'utilisateur peut sélectionner une date c'est à dire un fichier NetCDF. Cette sélection met automatiquement à jour la date visualisée dans la visualisation spatiale et inversement, la sélection d'une date différente dans l'onglet data met également à jour la date sélectionnée dans la visualisation temporelle 3D.

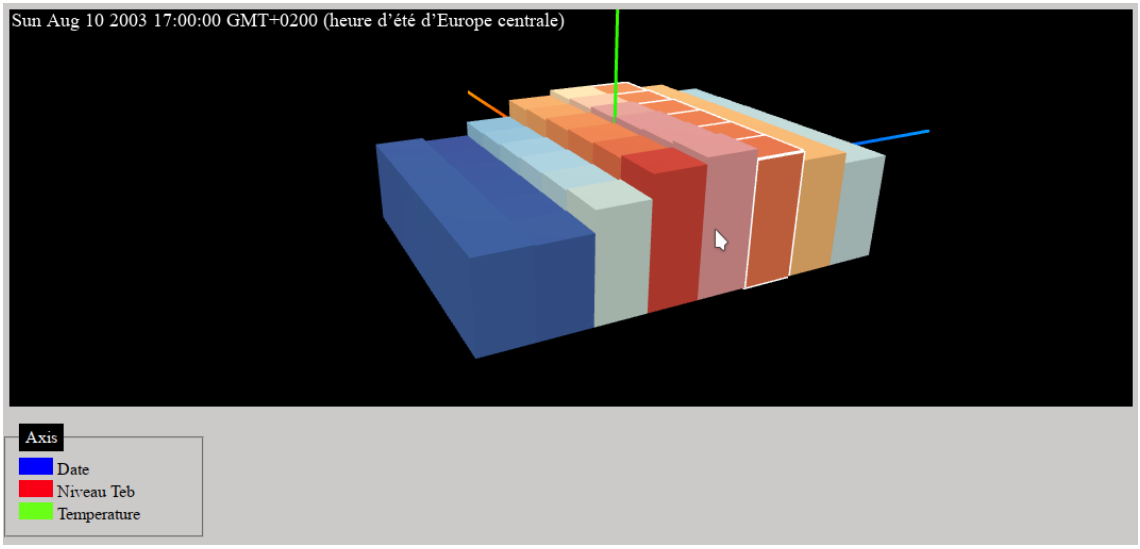

FIGURE 3.3 : Sélection de la date dans la visualisation temporelle 3D

3.2.2 Implémentation

La création d'une autre visualisation 3D implique la reconstruction d'une autre scène 3D sur la page. Nous avons logiquement utilisé THREE.js pour effectuer cela. Pour réaliser cette partie du code de l'application nous avons choisi d'utiliser une approche plus orientée objet afin de faciliter sa maintenance et les développements futurs.

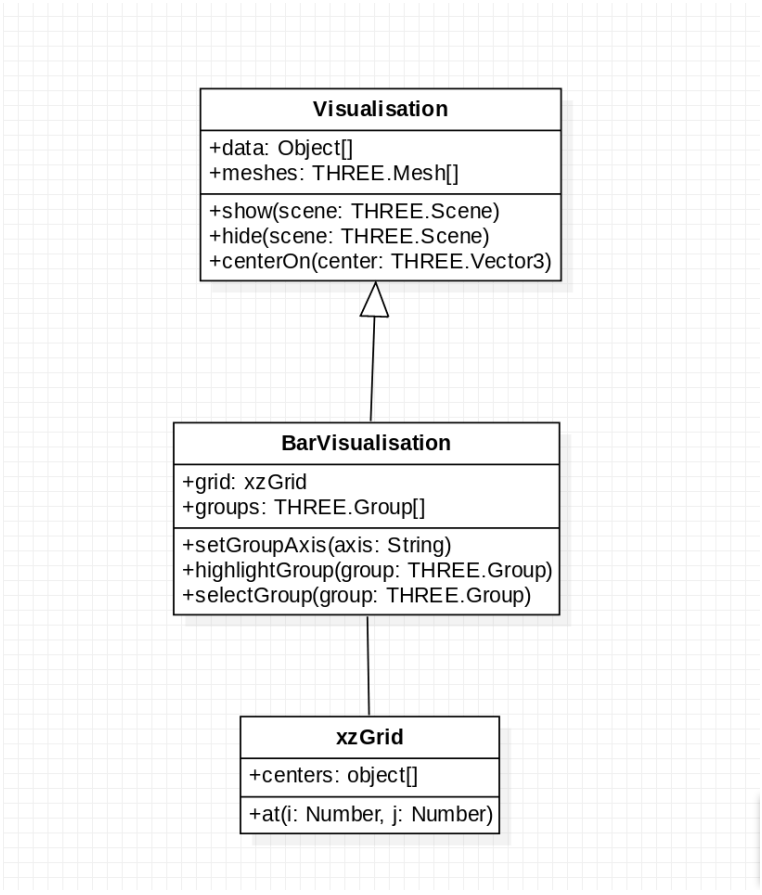

FIGURE 3.4 : Diagramme de Classe

Classe Visualisation

Cette classe est la classe mère. C'est une classe abstraite qui représente une visualisation quelconque. Elle porte les méthodes de base qui lui permettent d'afficher, cacher et centrer la visualisation. Elle porte également des attributs qui contiennent les données de la visualisation.

Classe BarVisualisation

Cette classe hérite de la classe visualisation. Sa spécificité réside principalement dans son constructeur : Il prend en entrée un tableau contenant les données, les attributs à afficher, une grille et une configuration. Ce constructeur instancie les meshs de l'histogramme à partir des données d'entrée et les autres paramètres. Elle porte des méthodes spéciales comme setGroupAxis qui permet de grouper les géométries des barres qui sont sur le même axe. Cette méthode nous permet de regrouper toutes les barres qui correspondent à la même date (ensemble elles représentent un fichier NetCDF). Les méthodes highlightGroup et selectGroup sont les deux méthodes qui permettent respectivement de mettre en surbrillance les barres d'une même date et sélectionner un date à afficher dans la visualisation spatiale.

Classe xzGrid

Cette classe permet de faciliter le placement des pieds de barres d'un histogramme. Elle permet de construire une grille régulièrement espacée en x et z au-dessus de laquelle on va construire les barres de notre histogramme.

Chapitre 4
Tâches complémentaires

Dans cette partie, nous décrirons les taches complémentaires que nous avons pu effectuer au cours de ce projet. Dans un premier temps, nous parlerons de celles en lien avec notre objectif principal : l'animation temporelle et les scènes 3D synchronisées. Puis nous parlerons de la représentation de températures interpolées.

4.1 L'animation temporelle et les scènes 3D synchronisées

4.1.1 L'animation temporelle

Un outil permettant de changer de fichier NetCDF affiché à l'écran à intervalle de temps régulier a également été implémenté ce qui permet de créer une "animation" de l'évolution temporelle de la température sur la zone étudiée.

4.1.2 Les scènes 3D synchronisées

Une dernière demande du commanditaire est arrivée en fin de projet : il s'agit de pouvoir visualiser deux dates au même moment. Cela permettrait à l'utilisateur de regarder exactement comment évolue la température à des lieux précis. Et donc de pouvoir détecter les îlots de chaleur.

Ainsi, ces fenêtres doivent permettre d'afficher 2 dates différentes mais avec les mêmes données (bâtis, plan horizontaux et verticaux ...). De plus, le déplacement ou le zoom sur l'une des deux fenêtres de visualisation doit pouvoir déplacer l'autre, afin d'observer toujours les mêmes lieux.

Nous avons commencé à implémenter cette partie mais mettre en place cette solution signifie le changement une grande partie du fonctionnement de l'application. En effet, on se retrouve alors avec deux dates à gérer dans 2 visualisations différentes.

4.2 Représentation des températures interpolées

4.2.1 Problématique des plans verticaux

Après avoir répondu aux attentes du commanditaire, nous avons développé plusieurs autres parties, plus ou moins conséquentes. Une grosse modification a concerné la façon d'afficher les plans verticaux. C'est à dire l'évolution de la température sur une route.

Dans le prototype de départ, une route est de la même température que le bloc dans lequel elle se situe. On obtient alors la visualisation de la figure 4.1.

4.2.2 Résolution via le calcul de la température

Cependant dans la réalité la température n'est pas constante sur tout un bloc et elle devrait donc évoluer de façon continue et non discrète. Afin de se rapprocher de la réalité pour le calcul de la température, on calcule pour chaque point de la route une valeur de température moyenne par rapport à tous les blocs adjacents pondérée par la distance.

Ce calcul s'effectue en GLSL et non en JavaScript. C'est à dire qu'il doit être effectué au moment de l'affichage du tronçon.

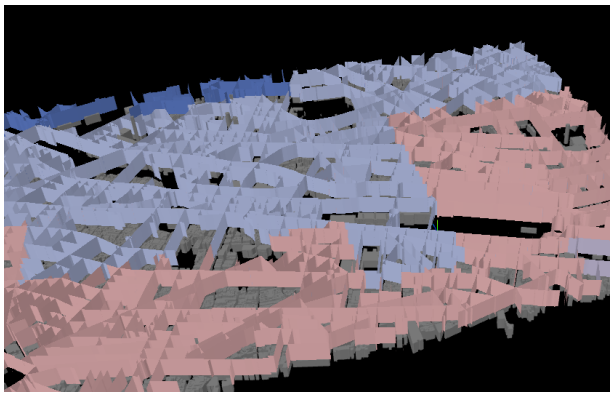

FIGURE 4.1 : Plan verticaux : 1 couleurs = 1 bloc

Pour ce faire, dans un premier temps, on récupère les points les plus proches : le bloc dans lequel on se trouve et les blocs les plus proches sur le même niveau comme on peut le voir sur le schéma 4.2.

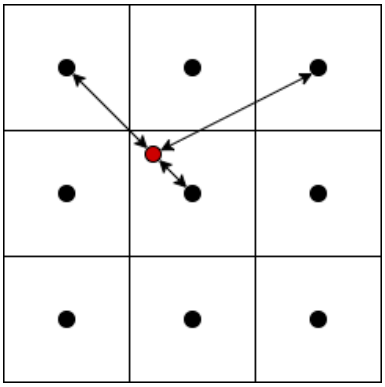

FIGURE 4.2 : Schéma explicatif des blocs les plus proches

On voit sur le schéma les 9 blocs les plus proches du point rouge. On récupère aussi les 9 blocs au dessus et en dessous. On se retrouve donc avec 27 points. Ce nombre peut varier suivant si on se situe sur une extrémité de l'emprise 3D.

On peut ensuite calculer la moyenne pondérée via la formule :

$$T_p = \frac{\sum \frac{T}{D}}{\sum \frac{1}{D}}$$

Où la somme se fait sur chaque bloc ; T la température du bloc ; D la distance entre le centre du bloc et la route ; Tp la température pondérée par la distance. Ainsi, on obtient le résultat suivant :

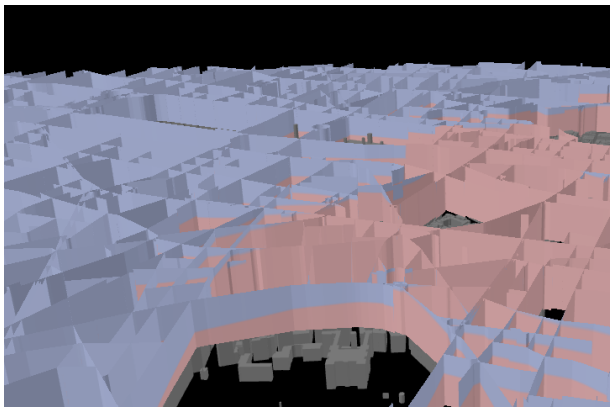

FIGURE 4.3 : Plan verticaux : 1 couleurs = 27 blocs

4.2.3 Passage à une échelle de couleur continue

Afin de trouver la couleur d'un plan, vertical comme horizontal, l'application se base sur une échelle de couleur discrète. Il s'agit de la figure 4.4. Ainsi, chaque couleur représente un intervalle de température. Entre 2 plans, la couleur peut donc être la même pour 2 valeurs de température différentes. Cela explique pourquoi les variations de couleurs de la figure 4.3 ne sont pas lisses.

FIGURE 4.4 : Échelle de couleur discrète

FIGURE 4.5 : Échelle de couleur continue

Nous l'avons donc remplacée par l'échelle de couleur continue de la figure 4.5. Ainsi, chaque température sera représentée par une couleur différente. On obtient dans le cas des plans verticaux un résultat plus lisse, que l'on peut voir sur la figure 4.6.

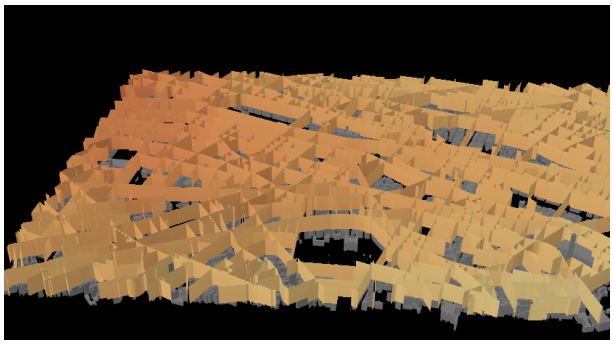

FIGURE 4.6 : Plan verticaux : échelle de couleurs continue

4.2.4 Affichage des plans horizontaux

Comme nous l'avons vu dans la partie 4.2.3, nous avons modifié l'échelle de couleur. Cette modification est aussi valable pour l'affichage des plans horizontaux. On peut observer sur les figures suivantes deux affichages différents pour la même donnée : le niveau TEB 1 le 10 août 2003 à 8h.

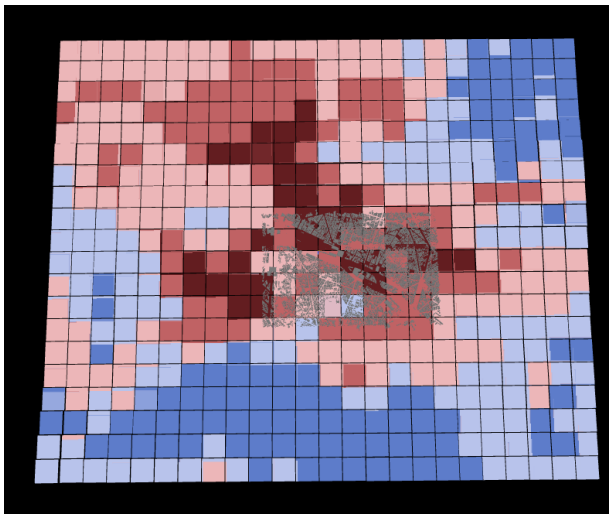

FIGURE 4.7 : Échelle de couleur discrète

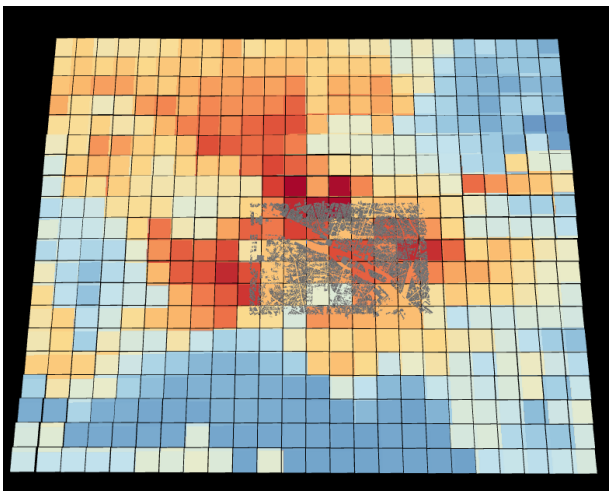

FIGURE 4.8 : Échelle de couleur continue

Nous pourrions imaginer, comme pour les plans verticaux un affichage plus lisse de la donnée. Cela signifierait, non seulement effectuer un calcul de la température de manière interpolée mais aussi modifier la façon dont sont affichées les plans horizontaux.

Chapitre 5
Gestion de projet

Dans un projet informatique de cette ampleur, il est important d'avoir une gestion du projet optimisée.

5.1 Notre organisation

5.1.1 Méthode de travail

Dans le cadre d'un projet de cette ampleur, il est impératif de trouver une organisation de travail efficace.

Nous avons décidé d'utiliser la méthode scrum pour ce projet. Nous avons reparti les différents rôles entre nous. Thibault Petiet étant de scrum master, Laura Wenclik product owner et enfin Félix Quinton et Cédric Périon furent développeurs. Cependant, Thibault et Laura ont eu la double casquette de développeur.

Par la suite, nous avons choisi de travailler à l'aide de GitHub comme gestionnaire de version étant donnée que le prototype initial était sur un dépôt GitHub. Nous avons décidé d'organiser nos sprints principalement en une ou deux semaines. En fin de sprint, nous avons organisé une review afin de regarder le travail effectué au cours du sprint et de vérifier nos objectifs. De plus, à chaque fin de sprint, nous avons effectué une rétrospective afin de voir ce qui s'est bien ou moins bien passé durant le sprint et voir comment l'améliorer.

A l'inverse en début de sprint, nous faisions le backlog en listant les objectifs et les tâches à effectuer ainsi que le planning du sprint.

De plus chaque jour, nous organisons un daily afin de faire un point sur notre avancement et les problèmes rencontrés. Une fois par semaine nous organisons une réunion avec notre client afin de lui présenter l'avancement de projet et de vérifier que le travail effectué correspondait à ses besoins. De plus, ces réunions permettaient de spécifier certains de ces besoins afin de pouvoir avancer dans le prototype et d'être sur de ne pas partir dans le mauvais sens. Et enfin, nous avons mis en place un environnement Teams afin de pouvoir échanger plus facilement entre nous et avec notre client hors des réunions et de garder une trace des discussions.

5.1.2 Backlog

Pour chaque sprint nous avons découpé le travail à faire en User Story (US), c'est à dire en ticket qui indique le travail en précisant la priorité de celui-ci dans le sprint ainsi que la difficulté en story points (1, 2, 3, 5, 8...). Il est aussi inscrit dans ces US les dépendances entre les différentes tâches. Nous avions ainsi listé les différentes tâches à accomplir dans différentes parties (NetCDF, application, visualisation spatiale, visualisation temporelle ...) comme on peut le voir sur la figure 5.1. Ces US ont été rentrées dans un fichier appelé backlog. En fin de sprint, le Scrum Master et le Product Owner nettoyaient le backlog en rentrant la date de fin par l'anotation du "f" de fini et en le complétant des nouvelles US. Les tâches non finies sont par la suite transposées dans le sprint d'après en priorité 1.

id	US	User Point	Priorité	Sprint	dépendance	Fini le
0	**NETCDF**	0	0	0	0	
1	Comprendre le format des données NetCDF	5	f	1		17-mars
3	Etat de l'art sur l'intégration des données NetCDF	3	f	1		12-mars
4	Verification de la version du format	2	f	1		17-mars
5	Changement de version de format	5	f	1		17-mars
7	Etat de l'art de jsfive	3	f	1		17-mars
	Importation des données selon le pas de temps	21	f			
9	Import direct des fichiers NetCDF au sein du Prototype	13	f	1	2	19-mars
22	Integration du serveur de conversion dans le code (au mc	5	f	1	5	19-mars
11	Lancer la méthode CreateDataTexture avec les bons argu	8	f	1	21	19-mars
55	Filtrer les données general_config.0, U, V selon l'emprise	5	f	2	depend de l'empris	23-mars
56	Calculer les Temps min et max	2	f	2	depend du filtrage	23-mars
58	Déterminer ni et nj	3	f	2	depend du filtrage	23-mars
89	Resoudre le probleme de ni et nj	3	f	2		24-mars
0	**APPLICATION**	0	0	0	0	
2	Création d'un bouton de chargement de donnée NetCDF	3	f	1		16-mars
13	Pouvoir importer plusieurs fichiers NetCDF	13	f	2		
	Affichage graphique des données temporelles (etat de l'a	21	4			
21	Integration de la récup dans le code actuel	5	f	1	8	
8	Récuperation des données Meso NH et TEB dans un code	8	f	1	1	17-mars
10	Comprendre le code	5	f	1		16-mars
24	Création de la carte leaflet	2	f	1		16-mars
25	Création d'un cursueur sur la carte leaflet par rapport à l'	3	f	1	24	17-mars
26	Extraire les coordonnées de la carte leaflet	1	f	1	25	17-mars
27	Verfier que les coordonnées soient dans la bonne project	1	f	1	26	18-mars
32	Limiter le changement d'emprise à l'emprise max	2	f	1	24	18-mars
29	Afficher le nuage de points sur la carte	2	f	1	24	17-mars
61	Gérer l'affichage des panneaux	2	f	2		23-mars
63	Créer un select de date dans la partie Data	1	f	2		23-mars
64	Ajouter les dates au select (extraction et creation)	2	f	2	depend de l'extract	23-mars
69	Modifier l'interface pour accepter plusieurs fichiers	1	f	2		25-mars
70	Modifier l'application pour faire une succession de requê	5	f	2		23-mars
71	Appel multiple à la fonction Load_Data pour plusieurs fic	5	f	2	70	23-mars
79	Changer la couleur des points n'appartenant pas à l'empr	3	f	2		23-mars
95	Utilisation des fichiers routes et batis	2	2	2		
112	Documentation et comprehension des shaders	5	f	4		13-avr
114	Réfléchir aux choix du calcul de l'interpolation	2	f	3		
116	Faire le calcul de la temperature du plan vertical, à partir	2	f	4	114	13-avr
118	Intégrer le calcul dans l'affichage	3	f	4	116	12-avr
120	Permettre le choix entre interpolé ou non	3	f	4		12-avr
122	Ajouter un "Chargement en cours" lors du chargement de	1	f	3		30-mars
125	Débugage des nuages de point : pourquoi tout est de la n	8	f	3		06-avr

FIGURE 5.1 : Une partie du backlog du projet

5.1.3 Dashboard

Le dashboard est un outil permettant de suivre, au cours d'un sprint, l'avancement de chaque US. Pour ce faire, nous avons réparti notre dashboard en 6 colonnes :

- **To do** : Les US qui n'ont pas encore été commencées.
- **Waiting** : Les US qui sont en attente.
- **In Progress** : Les US en cours de développement ou de réalisation.
- **Test** : Les US qui sont en cours de test par un autre membre de l'équipe.
- **Done** : Les US du sprint actuel qui sont terminées, testées et validées.
- **Close** : Les US des sprints précédents.

Cette organisation nous a permis de suivre le niveau d'avancement de l'équipe au cours du sprint, le travail restant du sprint mais aussi de voir sur quelle US travaille chaque membre.

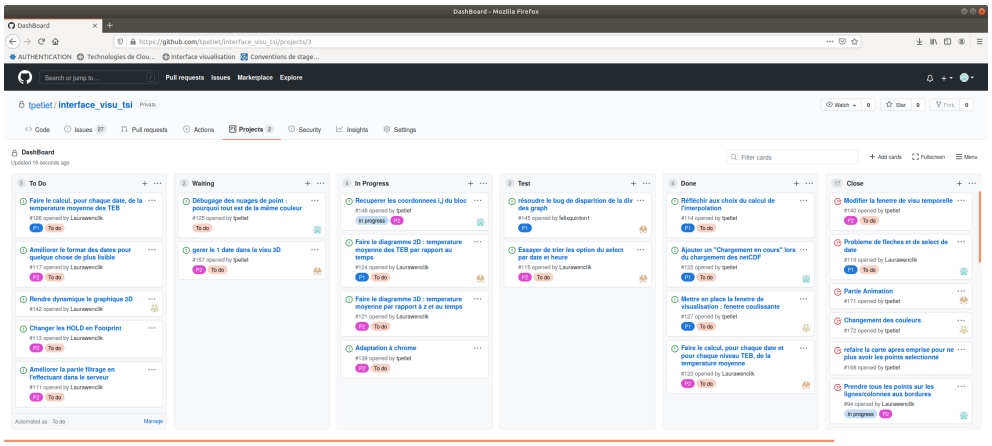

FIGURE 5.2 : Dashboard au cours du sprint 3

5.2 Les risques et contraintes

5.2.1 Matrice de risques

Au cours d'un projet, il existe de nombreux risques et contraintes qu'il est nécessaire d'identifier au préalable.

Ainsi, au début du projet, nous avons tenté de dresser une liste la plus exhaustive possible des risques auxquels nous allions éventuellement être confronté et avons pu les regrouper en quatre grandes catégories : lié à l'humain, au matériel, au client et à la gestion de projet.

Humain

La partie humaine correspond aux risques qui proviennent de l'équipe ou qui nous impacte directement.

- **1 : Dégradation de nos relations** : Il est nécessaire de savoir se dire les choses, de faire des points régulièrement et des rétrospectives afin de dire ce qu'il va ou ne va pas.
- **2 : Manque de compétences** : Il est possible de faire un état de l'art , de se former sur certaines compétences. Il faut aussi ne pas hésiter à s'informer auprès du reste de l'équipe.
- **3 : Confinement** : En cas de confinement, il est nécessaire de se connecter régulièrement sur Teams afin de continuer à communiquer et de pouvoir travailler correctement à distance.
- **4 : Absence, maladie, problème de transport** : Il faut prévenir ses camarades.
- **5 : Manque de motivation** : L'entraide est importante pour ne pas perdre sa motivation.
- **6 : Problème de communication** : Il est important d'optimiser le transfert des informations les plus importantes, ne pas hésiter à demander de l'aide et être rigoureux dans la mise à jour du tableau de bord.

Matériel

Les risques matériels proviennent principalement de nos machines ou de notre environnement.

- **7 : Vol de matériel** : Il faut bien fermer les salles informatiques.
- **8 : Perte de donnée** : Utiliser git de façon optimale en poussant son code afin qu'il soit dupliqué fréquemment, privilégier les échanges sur Teams afin de garder une trace.

- **9 : Code difficile à maintenir** : Il faut être rigoureux, commenter et établir des normes dans la façon de coder.
- **10 : Casse matériel**

Client

Les risques Client proviennent de notre relation avec notre client.

- **11 : Indisponibilité** : Les autres clients sont aussi là pour répondre aux questions.
- **12 : Dégradation des relations entre nous et le client** : Il faut bien communiquer avec lui.
- **13 : Changement d'avis du client** : Il est important d'échanger régulièrement afin de savoir rapidement si des changements sont nécessaires.
- **14 : Incompréhension entre nous et le client** : Il faut reformuler sa demande, communiquer, faire des points réguliers et présenter nos travaux.

Gestion

- **15 : Non respect des dates limites, fin de sprint** : Se baser sur les sprints précédents pour pouvoir estimer correctement les capacités de l'équipe.
- **16 : Gestion de Git laborieuse** : Il faut faire des commits fréquemment, faire relire et vérifier ses changements auprès de quelqu'un d'autre avant de fusionner les branches.

Nous avons par la suite classé ces risques par gravité et probabilité comme indiqué dans la figure 5.3. Les risques dont la couleur est rouge sont ceux qui sont les plus préoccupants dû à leur probabilité élevé de survenir et leur gravité. C'est à propos de ces risques qu'il faut être le plus vigilant et trouver un moyen de les surmonter s'ils surviennent.

Gravité du risque	Catastrophique	1,7	10	2,15	9
	Grave	8	6	14	
	Majeur		5,12	11	
	Mineur		16	3,4,13	
		Improbable	Peu probable	Probable	Très probable
		Probabilité			

FIGURE 5.3 : Matrice de risque

5.2.2 Les problèmes rencontrés pendant le projet

Travail à distance Durant le projet trois membres de l'équipe sur quatre ont été déclaré *cas contact* et se sont retrouvées en télétravail pendant plus d'une semaine. Cela n'a pas posé de problèmes en terme de communication et d'accès aux données. En effet, suite aux multiples confinements, notre équipe s'est acclimatée aux multiples contraintes qu'impose le télétravail. Cependant les machines personnelles de certains membres de l'équipe n'étaient pas suffisamment performantes pour faire fonctionner l'application correctement, principalement en ce qui concerne l'affichage de géométries 3D. Ces personnes ont donc été contraintes de travailler sur des thématiques ne nécessitant pas l'utilisation de ce genre de technologies.

5.2.3 Les contraintes

Dans le cadre de ce projet, nous avons eu plusieurs contraintes temporelles et matérielles.

Les contraintes temporelles s'illustrent dans les dates limites du projet. En effet, une première version du rapport est à rendre pour le 20 avril et le code ainsi que la documentation sont à rendre pour le 23 avril. A cela s'ajoute la soutenance qui aura lieu le vendredi 23 avril.

En ce qui concerne les contraintes matérielles, il y a dans un premier temps les rendus attendus par le commanditaire et notre responsable de filière qui sont : le code, un README, une documentation utilisateur et un rapport. Étant donné que nous ne démarrons pas le projet de zéro, nous devons nous adapter aux technologies et à l'architecture initialement présents au sein de l'application.

5.3 Notre planning

Conformément à la méthode agile Scrum, nous avons découpé notre projet en sprint d'une à deux semaines. Ces sprints nous ont permis de créer un planning organisé de notre projet.

5.3.1 Sprint 1

Le sprint 1 a duré une semaine et demi et consistait dans un premier temps à comprendre et documenter le code, analyser le sujet afin de déterminer les différents besoins du client. Dans un second temps, créer une première version de sélection d'emprise sur les données de base. Et enfin, comprendre le format des données NetCDF et arriver à charger un fichier NetCDF au lieu d'un fichier CSV.

5.3.2 Sprint 2

Le sprint 2 était lui d'une semaine. Les objectifs de celui-ci étaient d'intégrer la sélection de l'emprise aux données NetCDF ainsi que permettre l'import de plusieurs fichiers NetCDF à différents instants temporels.

5.3.3 Sprint 3

Le sprint 3 était sur deux semaines à cause du week-end prolongé de Pâques. L'objectif principal de ce sprint était d'obtenir une visualisation temporelle 2D et 3D dans l'application. Nous avons donc créé deux graphiques 2D et 3D ce qui a fourni une nouvelle façon de visualiser les données. Le second objectif du sprint était de commencer le bonus "principal" du projet : la représentation continue des températures en fonction des axes routiers. Cette partie étant codée en GLSL et non en JavaScript une longue analyse a du être effectuée par certains membres de l'équipe pour pouvoir anticiper les changements à apporter à l'application.

5.3.4 Sprint 4

Le sprint 4 a été le dernier sprint et a duré deux semaines. Les objectifs étaient de finir la visualisation temporelle 3D, terminer la fonctionnalité de représentation continue de températures en fonction des axes routiers, corriger les derniers bugs, et implémenter une ébauche de double visualisation. Ce sprint a aussi eu pour objectif de créer les rendus finaux (rapport, documentation utilisateur, présentation).

5.3.5 Le résultat de nos sprints

A la fin de chaque sprint, nous avons créé des Burndown chart. C'est à dire des diagrammes représentant notre avancée au cours du sprint. On y représente la courbe idéale du nombre story points effectué par jour ainsi que la courbe réelle de ceux-ci. Cet outil nous permet d'observer notre

rythme de travail, au sein même du sprint mais aussi entre chaque sprint. Ainsi, pour le sprint 3 en figure 5.4, nous avons d'abord été plus lent que prévu puis le 4ème jour, nous avons réussi à inverser la tendance.

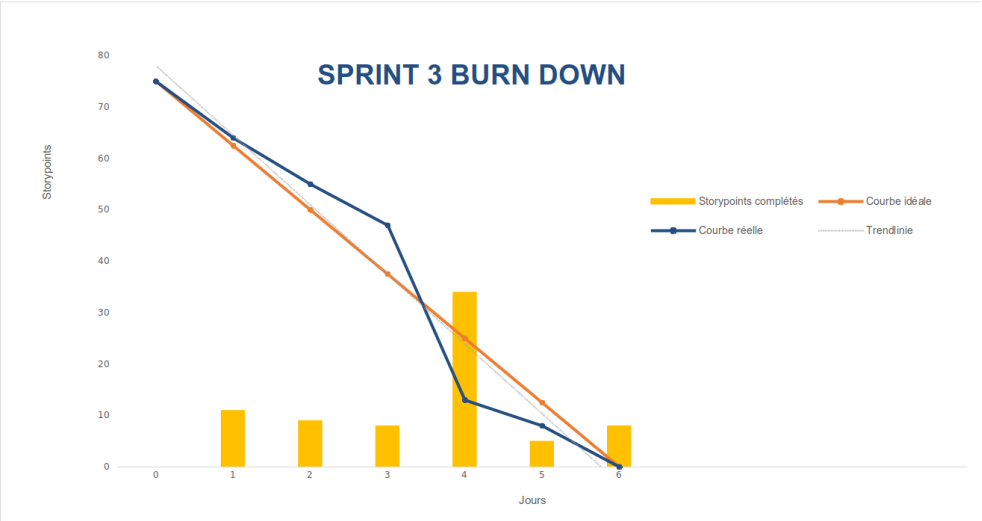

FIGURE 5.4 : Burndown chart du sprint 3

En comparant aux autres sprints en annexe B, on peut voir que le nombre de story points effectué pour chaque sprint fluctue entre 70 et 80. En prenant ces données, on trace la courbe de vélocité en figure 5.5.

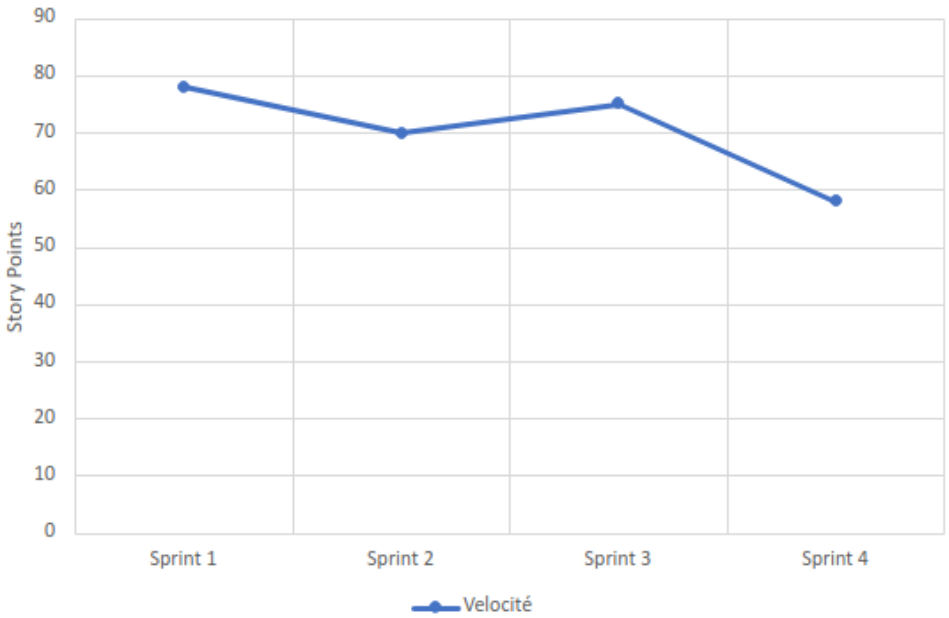

FIGURE 5.5 : Vélocité de l'équipe

Conclusion

L'objectif principal de ce projet était de permettre la visualisation de la composante temporelle des données au sein d'une interface de co-visualisation de données urbaines et météo. Pour répondre à cette problématique, nous avons permis à l'utilisateur de travailler avec le format NetCDF, afin de nous affranchir d'une phase de pré-traitement. L'utilisation de ce format a nécessité d'offrir à l'utilisateur un outil de sélection spatiale permettant de limiter la charge de données affichées à l'écran et donc de pouvoir travailler avec plusieurs fichiers à la fois. Si la sélection d'emprise a permis d'orienter l'utilisateur de façon spatiale, l'implémentation des graphiques de visualisation temporelle permet d'orienter l'utilisateur temporellement.

Une fois l'objectif principal du projet atteint nous avons pu mettre en place une nouvelle façon de représenter les données, en le faisant de manière continue, ce qui est plus proche de la réalité mais également plus coûteux en ressources.

A un niveau plus personnel ce projet nous a permis de découvrir de nouvelles technologies et d'approfondir nos compétences en développement web ainsi que dans certaines libraires spécialisées comme THREE.js. Nous avons également pu développer les compétences en méthodes agiles que nous avons acquises durant notre scolarité, et ce malgré la crise sanitaire actuelle.

Nous avons aussi eu l'occasion de rassembler les différentes compétences que nous avions acquises dans nos options respectives (C et G). En effet, Félix et Cédric qui viennent de l'option C ont apporté une composante technique technique au projet. Alors que Thibault et Laura qui proviennent de l'option G ont apporter un certain recul sur l'ensemble du projet pour permettre de mener à bien les rôles de Scrum Master et de Product Owner.

Nous ressortons donc satisfait du travail que nous avons pu fournir tout au long de ce projet ainsi que du résultat auquel nous sommes arrivés.

Bibliographie

[1] *Chart.js*. https://www.chartjs.org//. Page consultée le 6 avril 2021.

[2] *exemple3D*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.685&rep=rep1&type=pdf. Page consultée le 15 mars 2021.

[3] *GLSL*. https://fr.wikipedia.org/wiki/OpenGL_Shading_Language. Page consultée le 6 avril 2021.

[4] *Leaflet.js*. https://leafletjs.com/. Page consultée le 22 mars 2021.

[5] *netcdfjs*. https://cheminfo.github.io/netcdfjs/. Page consultée le 16 mars 2021.

[6] *Proj.js*. http://proj4js.org/. Page consultée le 23 mars 2021.

Table des figures

Annexes

Maquette générale de l'application

Annexe A

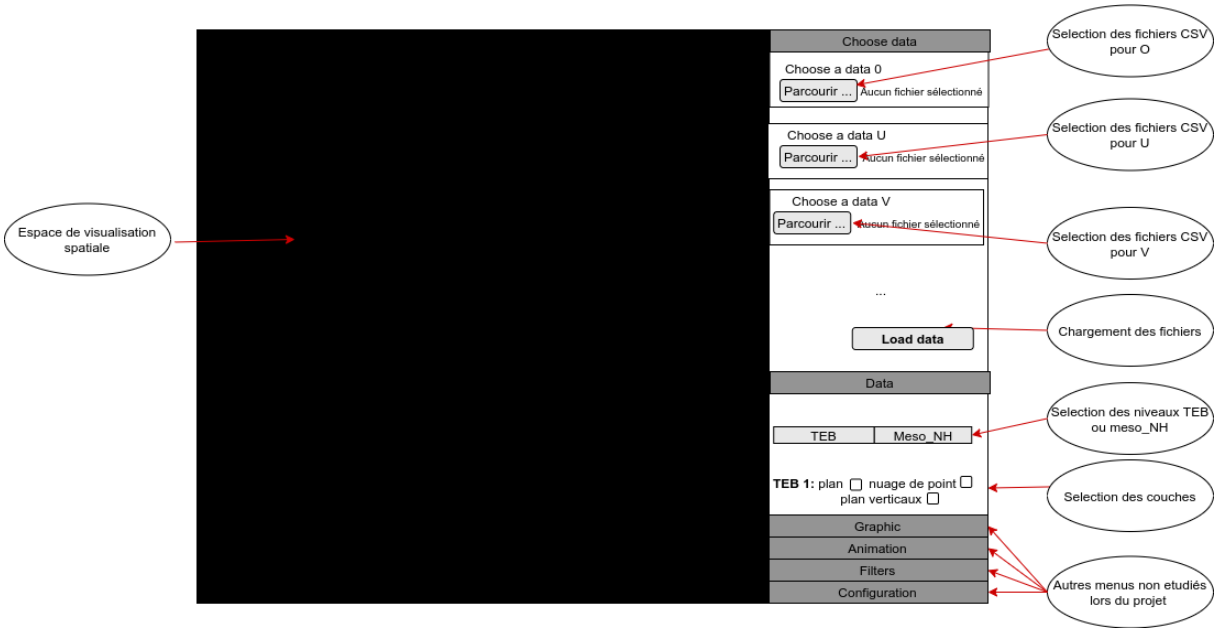

Figure A.1 : Maquette générale de l'application initiale

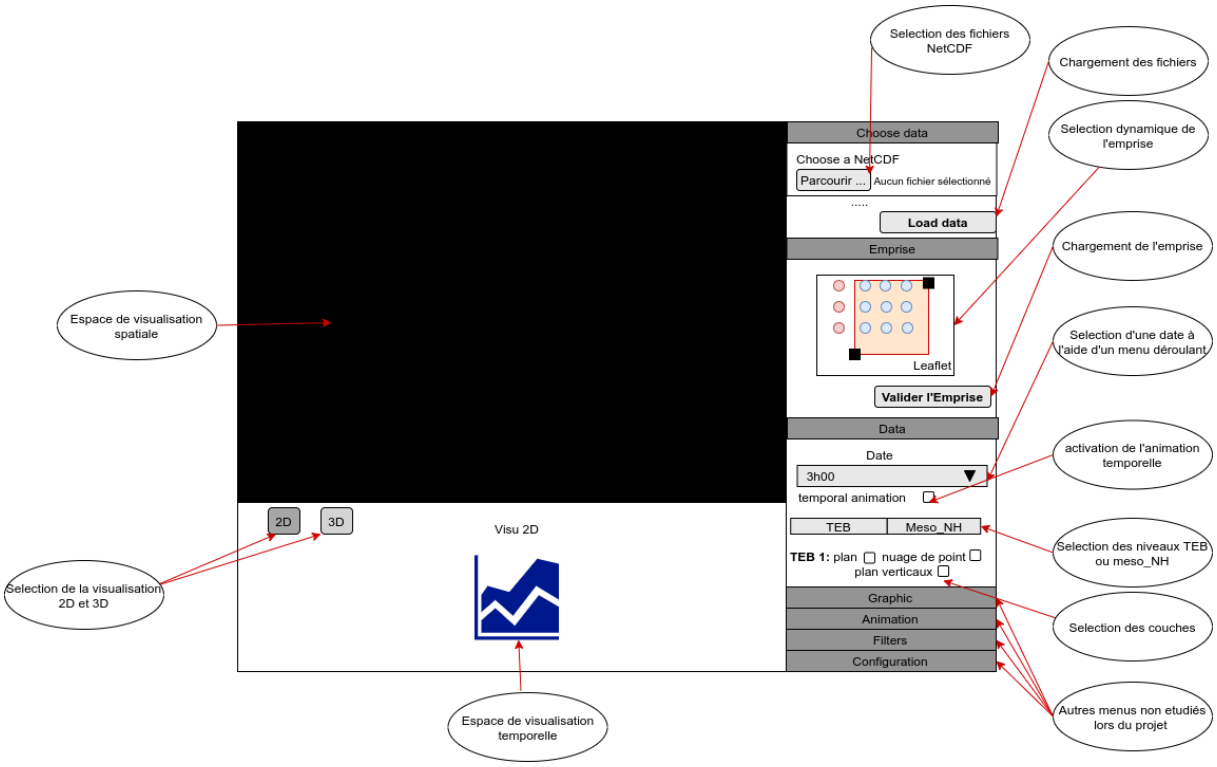

FIGURE A.2 : Maquette générale de l'application

Burndown Chart

Annexe B

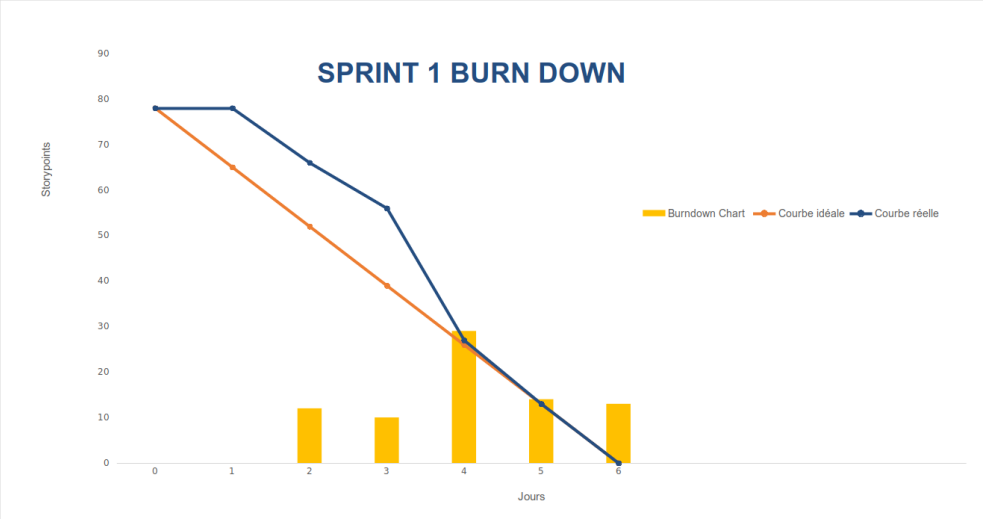

Figure B.1 : Burndown chart du sprint 1

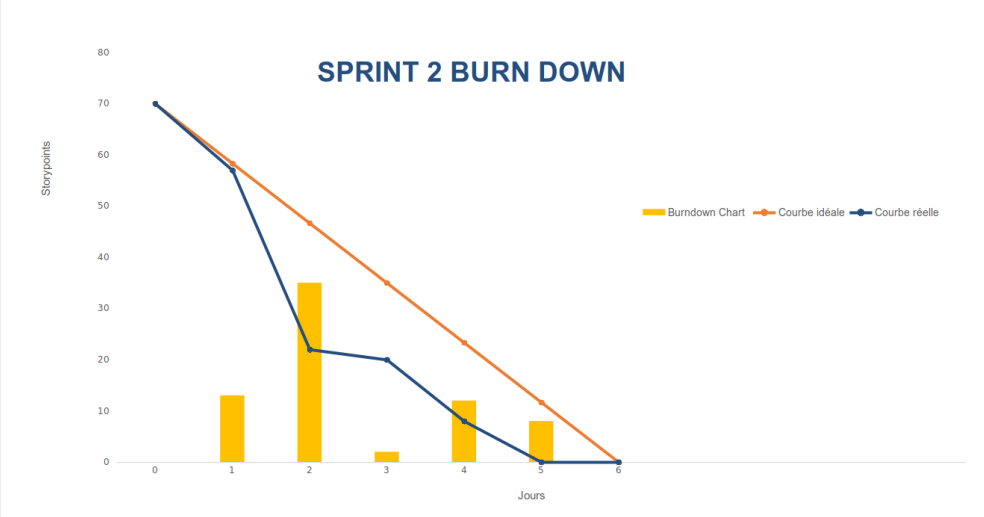

Figure B.2 : Burndown chart du sprint 2

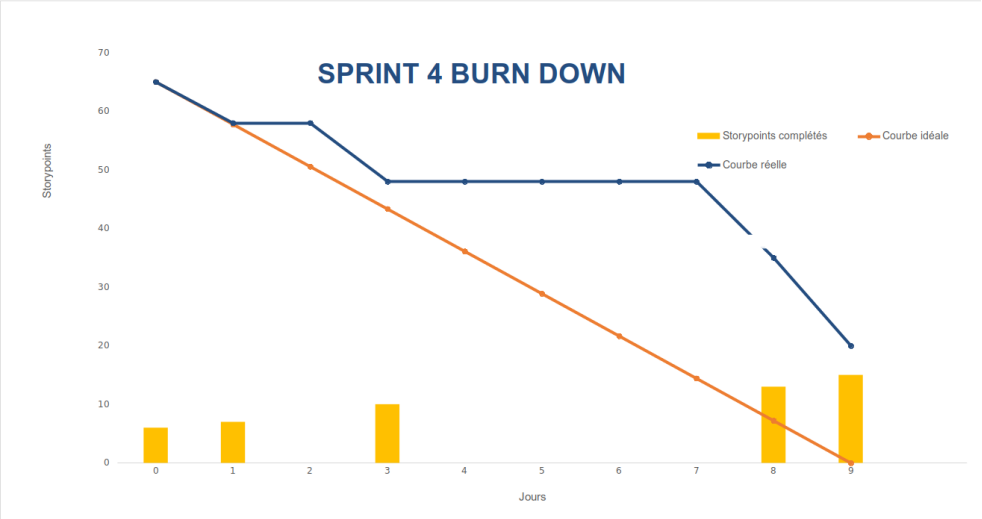

FIGURE B.3 : Burndown chart du sprint 4

Note : Le nombre de story points pour le dernier sprint n'a pas fini à 0 car la partie double visualisation décrite dans le paragraphe 4.1.2 n'a pas été aboutie.