Gestion du Projet de Développement Informatique

Rapport final d'analyse Informatique

Cycle ING2 - PDI 25

17 mai 2023

Equipe Projet :
ABDUL KUTHOOS Hannick
CACHARD Benoit
CANTAL Bryan
DE LACOUR Malo

Commanditaires :
AUBERT Julie
BRENAC Charlotte
GUIBAL Nicolas
PARISI Lucas

Table des matières

Table des figures

Contexte du projet

Depuis plusieurs années, les stations de ski entrent dans ce que l'on peut appeler le cercle vicieux de la production de neige de culture. En effet, le réchauffement climatique engendre une augmentation de la production de la neige de culture sur les domaines skiables (+20% en 5 ans).
Cette production nécessite une forte consommation d'eau (1m^3 d'eau pour produire 2m^3 de neige artificielle) et d'énergie, ce qui accroît et donc aggrave le réchauffement climatique. Le problème soulevé par cette tendance est ainsi la surproduction de la neige artificielle.

La problématique à se poser est donc : Comment optimiser la production de neige de culture au sein des stations de ski ?

L'entreprise **ELDA TECHNOLOGY** s'est donnée pour mission de répondre à ce problème. Elle est constituée de 4 employés qui sont nos commanditaires :

Nom	**Charlotte BRENAC**	**Julie AUBERT**	**Nicolas GUIBAL**	**Lucas PARISI**
Domaine	Communication		Technique	
Rôle	Directrice des opérations	CEO - Responsable commercial	Responsable pôle R&D	Responsable pôle produit

Figure 1 : Présentation de l'équipe de ELDA TECHNOLOGY

Nos 4 commanditaires ont donc proposé un projet visant à **concevoir une interface web** permettant la visualisation de statistiques sur un domaine skiable ainsi **qu'une visualisation 2D et 3D de la hauteur de neige sur les pistes du domaine**.
ELDA TECHNOLOGY utilise les techniques de l'acquisition Lidar par drone pour mesurer les hauteurs de neige sur le domaine skiable, en comparant l'élévation du terrain avec un MNT de base.

Ce projet contribuera au développement de l'entreprise ELDA TECHNOLOGY car elle acquerra l'expertise suffisante pour éviter aux gérants des domaines skiables la surproduction de neige de culture. C'est un projet qui apporte donc une solution économique (moins de dépenses pour la production de neige) et écologique (moins de consommations énergivores).

Notre équipe projet est constituée de 4 étudiants en cycle ING2 : Hannick ABDUL-KUTHOOS, Benoit CACHARD, Bryan CANTAL et Malo DE LACOUR.

I) Objectifs du projet - Reformulation du besoin

1.1) Les objectifs de l'étude

L'objectif de notre projet a été de développer une **application Web contenant une interface graphique SIG cartographiant (en 2D et 3D) le niveau de neige sur l'ensemble des pistes d'un domaine skiable**.
Cette application a également pour but d'informer l'utilisateur de d'autres paramètres de suivi du domaine tels que :

- la position des points d'intérêt sur le domaine skiable (enneigeurs, sondes météo)
- les données historiques et les prévisions météorologiques sur le domaine skiable
- les données historiques de la production des enneigeurs

Cette interface Web doit respecter les performances imposées par notre cahier des charges : elle doit être **ergonomique et simple d'utilisation.**

Le premier objectif du projet a été de **définir les fonctionnalités de notre application**. Certaines avaient été proposées par nos commanditaires : possibilité d'obtenir la hauteur de neige à un endroit donné à partir d'un simple clic, ou encore visualiser les statistiques associées à la hauteur de neige (volume, moyenne) sur un périmètre tracé par l'utilisateur. Pour ce faire, il a été nécessaire de définir le besoin. *Quels sont les paramètres en entrée nécessaires pour implémenter ces fonctionnalités ? Comment afficher l'information sur le site Web de sorte qu'elle soit le plus accessible possible par l'utilisateur ?*

Ensuite, nous avons dû déterminer comment réaliser de telles visualisations. Cela s'est manifesté par un **travail de recherche visant à définir l'architecture serveur** la plus optimale pour mener à bien ce projet.
Nous avons dû réaliser une étude comparative des fournisseurs de fonds de carte afin de pouvoir sélectionner le plus adapté (Leaflet, MapBox, Cesium …). En outre, il était nécessaire d'organiser la structure d'une base de données SQL capable de stocker l'ensemble des données nécessaires à l'utilisateur.

Le but est d'aboutir au fonctionnement suivant :
L'utilisateur souhaite visualiser une donnée qu'il possède en local (raster des hauteurs de neige, csv de données météo …) sur l'interface Web.
Tout d'abord, si ce n'est pas déjà fait, il devra importer le fichier correspondant dans la base de données via une plateforme de dépôt de fichiers sur l'interface Web.
Ensuite, l'utilisateur effectue une requête de visualisation de cette donnée via un formulaire sur l'interface Web. Cette requête ira chercher le fichier dans la base de données, puis va l'exporter vers l'interface Web afin d'afficher le résultat.

1.2) Les contraintes

La première contrainte a été le temps : le délai de 10 semaines a été très serré pour réaliser ce projet.
De plus, la solution proposée devait obligatoirement être une **solution web**. Le développement du site Web s'est essentiellement basé sur un MVP complet Figma de la plateforme fourni par les commanditaires en début de projet. Ces maquettes contiennent les différents assets nécessaires à l'implémentation du site web (icônes, logo, images, couleurs ...) et ont défini une **charte graphique à respecter** pour correspondre au site de communication de ELDA TECHNOLOGY.

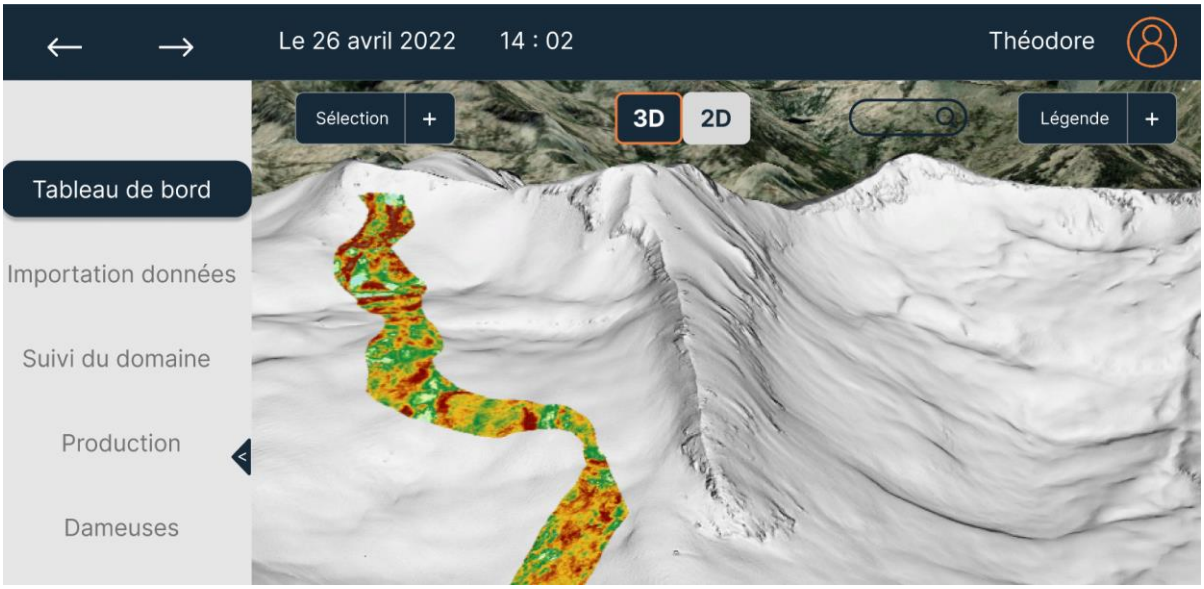

Figure 2 : Maquette Figma du tableau de bord de l'interface Web

Cette solution web doit avoir **une grande ergonomie et être intuitive pour l'utilisateur** pour lui permettre de naviguer simplement et efficacement dans les données. Enfin, la **représentation colorimétrique des hauteurs de neige** nous est également imposée (échelle de couleurs vert-rouge sur la maquette) : il s'agit d'une norme à laquelle doivent répondre les commanditaires eux-mêmes.

De plus, pour des raisons budgétaires, il a été convenu avec nos commanditaires que cette solution devra être développée avec des logiciels Open Source.

1.3) Le recueil du besoin - Les acteurs

Le but est de satisfaire les besoins du **client**, qui dans le cadre de notre projet est le **gestionnaire du domaine skiable de Serre Chevalier, et à long terme celui de n'importe quel domaine skiable**. Le client a besoin des informations suivantes :

- une visualisation 2D et 3D des hauteurs de neige sur les pistes du domaine skiable pour une date donnée, et représentées selon une norme colorimétrique *(dégradé de couleurs précis)*.
- Les positions des points d'intérêt de son domaine : emplacement des enneigeurs, des pistes et des stations météo
- La possibilité de visualiser l'évolution temporelle des données météorologiques et de production des enneigeurs afin d'éventuellement anticiper la production de neige de culture.

Pour répondre à ces besoins, il est nécessaire que notre **commanditaire ELDA TECHNOLOGY** soit en mesure de diffuser une plateforme offrant les services suivants :

- Capacité à produire des modèles de visualisation 2D et 3D correspondant aux attentes du client
- Mise en place d'une Architecture Serveur comportant une base de données regroupant toutes les informations réclamées par le client.

L'objectif de notre **équipe projet** a donc été de répondre aux 2 besoins du commanditaire, le tout dans une interface Web accessible par le client.
Pour cela, nous avons eu besoin des données de base suivantes :

- Délimitation géographique du domaine skiable avec position des points d'intérêt au format geojson
- Tableurs csv des données météorologiques et de production des enneigeurs
- Le plus important : fichiers rasters des hauteurs de neige au format geotiff

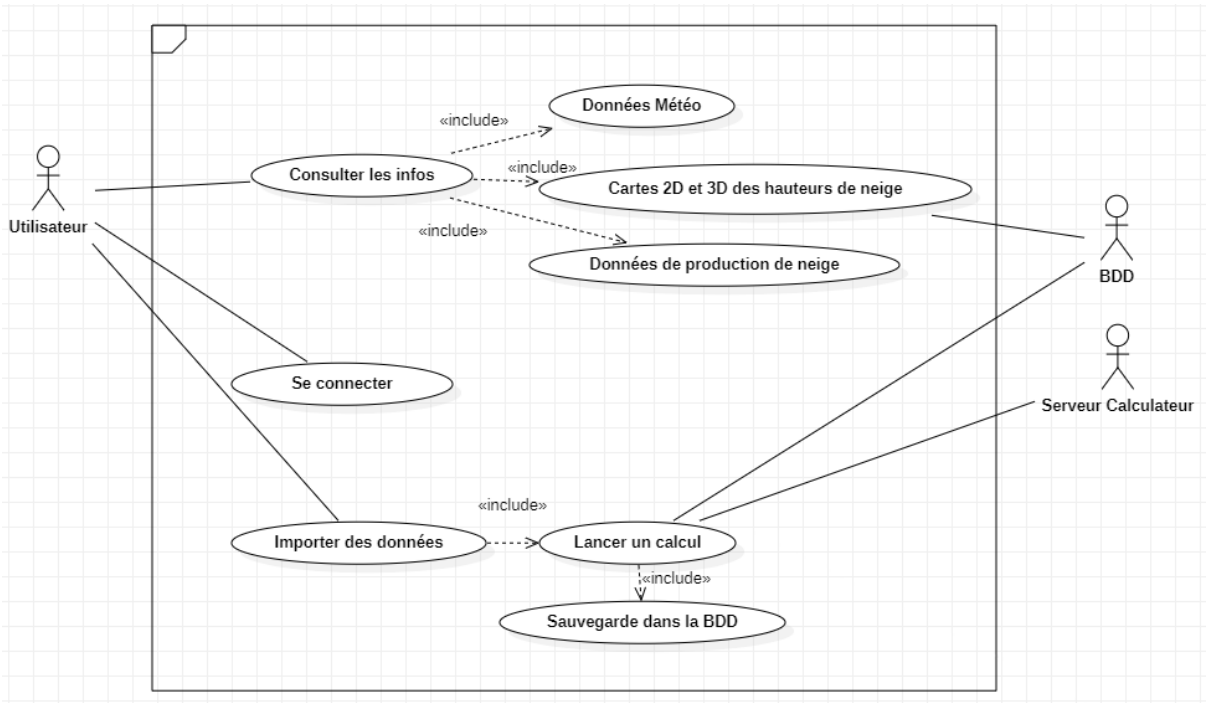

Figure 3 : Diagramme de cas d'utilisation illustrant l'analyse du besoin du point de vue du client

II) Analyse fonctionnelle et présentation des solutions implémentées

2.1) Fonctionnement de l'interface graphique côté client

Fonctionnalité prioritaire : le tableau de bord

Lorsque le client se connecte, le site est censé le rediriger vers un tableau de bord avec un MNT "vierge" (sans hauteurs de neige) du domaine skiable dont il est le gestionnaire.
Ce MNT a été chargé via la bibliothèque Javascript Open Source **Cesium.js** .
Cette bibliothèque a été choisie car il s'agit d'une solution Open Source proposant des cartes interactives en 2D et en 3D, sous le nom de **Cesium Viewer**, sur laquelle nous pouvons ajouter des données géospatiales (raster des hauteurs de neige, plan des pistes, positions des points d'intérêt).

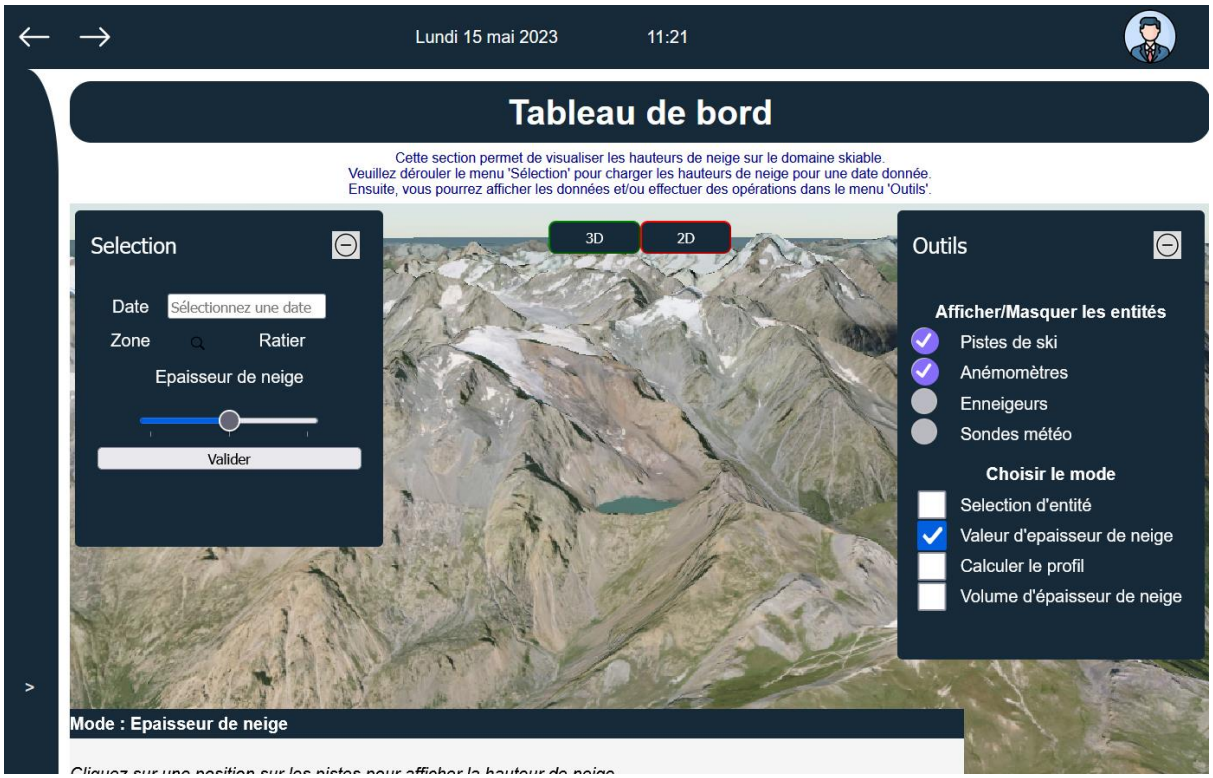

Figure 4 : Capture d'écran du tableau de bord du site web - visuel final

L’interface propose au client 2 menus déroulants :

- A gauche, le menu “Sélection” demande à l’utilisateur à quelle date il souhaite ses hauteurs de neige, et va donc charger le raster correspondant depuis la base de données pour une visualisation 2D et 3D de ces hauteurs sur les pistes du domaine. Ce raster est donc géoréférencé puis “plaqué” sur le MNT du Cesium Viewer, tout en respectant la symbolique colorimétrique de notre projet.

- A droite, le menu “Outils” permet de choisir entre les fonctionnalités suivantes :
    - Lorsque l’utilisateur clique sur une localisation quelconque d’une piste de ski, l'interface graphique devra être capable d’y afficher le niveau de neige correspondant.
    - L’utilisateur doit pouvoir tracer une ligne sur la carte, et l’interface affiche une coupe des niveaux de neige le long de la ligne.
    - L’utilisateur doit pouvoir tracer un polygone sur la carte, et l’interface affichera le volume de neige dans ce polygone, ce qui peut donner une idée de la quantité de neige à produire (ou pas) dans cette zone.

Autres fonctionnalités :

- Une plateforme d’importation et visualisation de données historiques météo (température, humidité, vitesse et direction du vent) sur un graphique.
- Une plateforme de suivi du domaine : prévisions météo, position des sondes météo, historique et prévisions des hauteurs de neige moyennes sur le domaine
- Une plateforme d’importation et visualisation de données historiques de production de neige : position des enneigeurs sur une carte et rapports de production sur un graphique.

Pour les données historiques météo et de production, l’utilisateur est censé spécifier une période ainsi que l’identifiant de la sonde météo/enneigeur. Ces paramètres aboutiront automatiquement à la création d’un graphique à partir de données importées depuis la BDD.

En somme, l’utilisateur dispose de tous les outils permettant d’optimiser sa production de neige sur son domaine skiable.
Voici un diagramme d’activités résumant les interactions entre l’utilisateur et les fonctionnalités de l’application Web :

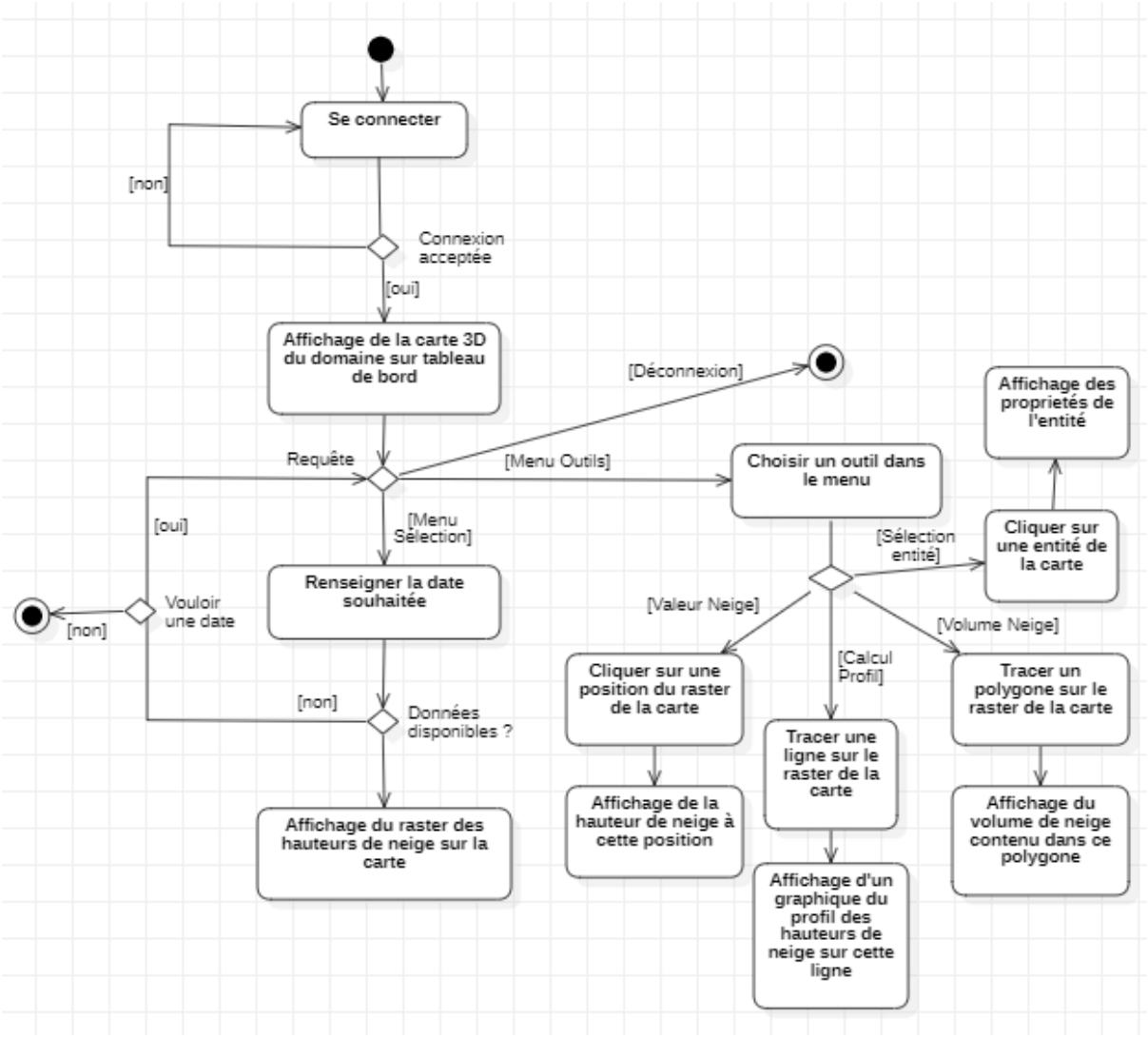

Figure 5 : Diagramme d'activité illustrant les différentes interactions entre l'utilisateur et l'interface web sur le tableau de bord

2.2) Fonctionnement du côté serveur

Nous avons choisi de répondre à cette question en implémentant un diagramme de classes:

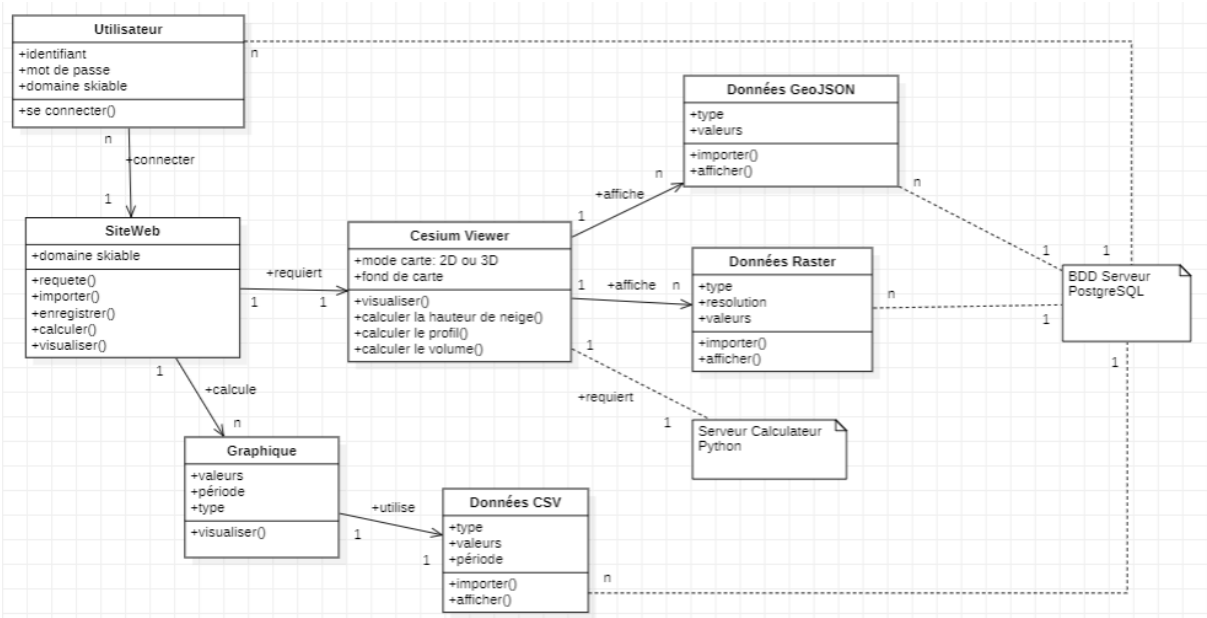

Figure 6 : Diagramme de classes des données utilisées par l'interface Web

Tout d'abord, pour que le client ait accès à la visualisation de son domaine skiable sur le site Web, il faut enregistrer son compte dans une base de données utilisateur.
Ensuite une fois sur l'interface web, le client peut :

- Importer sur le site Web des données raster (geotiff) correspondant aux données de hauteurs de neige à une certaine date, ou geojson pour les infrastructures du domaine, ou csv pour les données météo ou de production des enneigeurs. Une fois importées, elles sont enregistrées dans la base de données.
- Effectuer une requête d'affichage des pistes et des points d'intérêt ou des hauteurs de neige à une certaine date sur le Cesium Viewer. Ces données, au format geotiff (raster) ou geojson ont été récupérées sur la BDD, car elles ont déjà été importées par le client, comme expliqué au premier point.
- Effectuer une requête de calcul puis d'affichage d'un graphique de données météo ou de production des enneigeurs.

Les requêtes récupèrent les données sur la BDD (au format geotiff, geojson ou csv), car elles ont déjà été importées par le client, comme expliqué au premier point.

III) Etude technique : Choix des logiciels et langages - Architecture

3.1) Historique recherches de solutions serveur

Le choix entre une solution propriétaire et une solution Open Source :

Dans le cadre de l'étude technique visant à choisir les logiciels et les langages pour l'architecture du projet, plusieurs recherches et comparaisons ont été effectuées. Une des solutions envisagées est l'utilisation des logiciels ESRI, notamment ArcGIS, pour mettre en place une solution web capable d'afficher des données raster en 3D et répondre aux exigences du cahier des charges d'ELDA TECHNOLOGY. Ce choix implique plusieurs étapes clés, telles que la configuration d'un serveur ArcGIS, la publication de services web pour l'affichage des données raster en 3D, l'utilisation des API ArcGIS pour intégrer des fonctionnalités de cartographie et d'analyse, ainsi que l'utilisation de bibliothèques 3D comme Three.js pour créer des scènes interactives. Cependant, un inconvénient majeur de cette solution est son coût potentiellement élevé en termes de licences, qui doivent être renouvelées chaque année. Les coûts sont variables en fonction de plusieurs facteurs, rendant difficile une estimation précise, ce qui peut constituer un défi pour des startups comme ELDA TECHNOLOGY. Malgré cela, la solution ESRI offre des services très efficaces, notamment une large gamme d'API utilisables avec différentes plateformes ArcGIS. Ces API permettent d'utiliser plusieurs langages de programmation, offrent des fonctionnalités avancées de cartographie, d'analyse spatiale et de création d'applications mobiles et web. Elles sont constamment mises à jour, offrant ainsi un accès aux dernières fonctionnalités et technologies SIG.
En conclusion, la solution ESRI offre des logiciels et des API performants pour la visualisation en 3D de données raster, avec des fonctionnalités avancées de cartographie, d'analyse spatiale et une flexibilité d'intégration dans divers environnements de développement.

La possibilité d'une solution open source a également été envisagée, proposant une architecture serveur utilisant des logiciels et des outils open data, ne nécessitant aucune licence. Cette solution offre des services équivalents à ceux de la solution ESRI, mais avec la différence majeure que les outils de traitement des données doivent être développés en interne plutôt que d'être disponibles clé en main. Les principaux outils de cette architecture comprennent QGIS, un logiciel SIG open source pour la construction, la visualisation et l'analyse de données géospatiales, ainsi que PostGIS, une extension pour PostgreSQL permettant le stockage et la manipulation de données géospatiales. PostgreSQL est utilisé comme système de gestion de base de données, tandis que PGAdmin est utilisé comme interface graphique pour gérer les bases de données via des requêtes SQL. GeoServer est utilisé comme serveur d'application pour stocker et publier les données géospatiales sur l'interface web. De plus, l'extension Qgis2threejs permet de générer des visualisations 3D interactives à partir des données géospatiales en 2D, offrant des fonctionnalités telles que

l'affichage en 3D, la personnalisation des styles, la génération d'ombres, la navigation interactive, l'ajout d'animations, d'étiquettes et de légendes, ainsi que l'exportation et le partage des scènes 3D.
En résumé, cette solution open source utilise de multiples outils pour fournir une puissante plateforme de visualisation et de traitement des données géospatiales en 3D. Cette solution présente l'avantage d'être 100 % open source et donc gratuite mais semble plus complexe à mettre en place.

Après avoir examiné les rapports techniques sur les deux solutions, ELDA TECHNOLOGY a pris la décision d'opter pour la solution Open Source en raison des coûts élevés et imprévisibles de la solution ESRI, ainsi que de la plus grande adaptabilité de la solution Open Source. Cependant, ce choix a un impact important sur le projet, nécessitant un volume de travail plus important pour développer les outils de traitement des données SIG et mettre en place l'architecture serveur en utilisant GeoServer, malgré une expérience limitée dans l'utilisation de cet outil. L'architecture de la solution Open Source validée dans un premier temps par nos commanditaires est présentée ci-dessous :

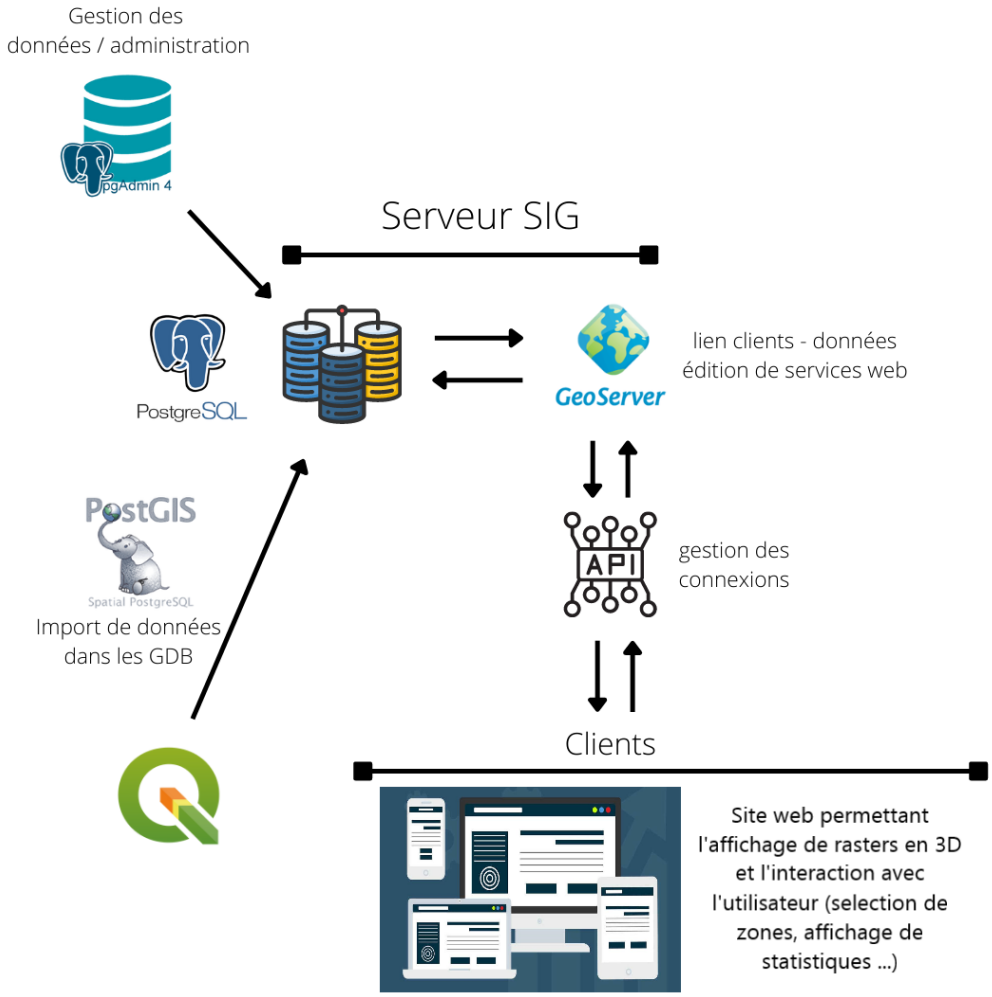

Figure 7 : Architecture open-source initialement envisagée

Choix stratégiques et outils / logiciels utilisés au cours du développement :

A la suite du CoTech de lancement, des retours encourageant sur nos premiers pas dans le projet ont été donnés. De plus, certains détails techniques ont été affinés. Le choix de la solution Open Source ayant été fait par les commanditaires, les retours du CoTech de lancement ont été en grande partie focalisés sur cette dernière.

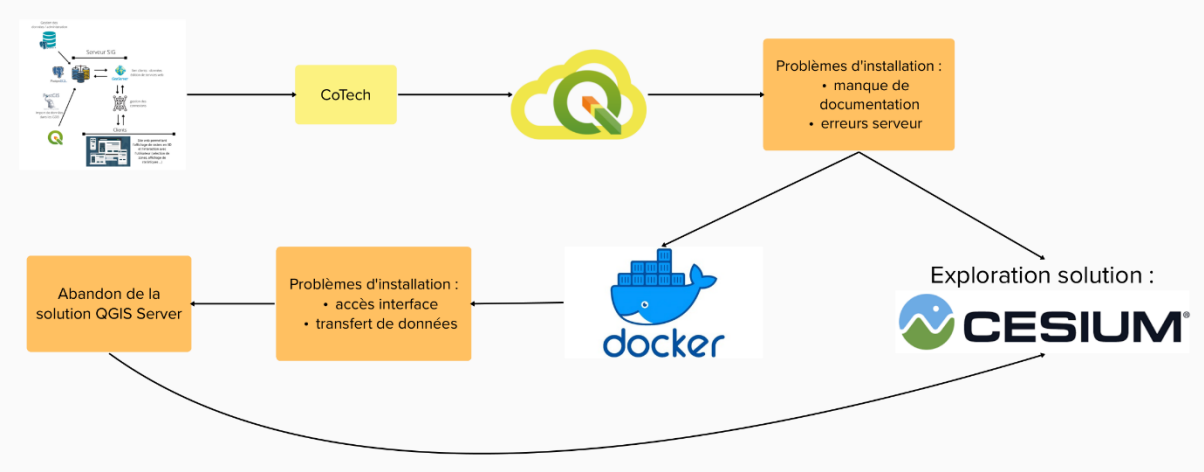

Figure 8 : problèmes rencontrés et choix effectués dans le développement de la solution permettant l'affichage des données traitées sur l'interface web

La planification ainsi que l'architecture globale proposée a été validée. Cependant, un problème a été soulevé. En effet, l'architecture initialement présentée (voir figure ci-dessus) proposait une utilisation de QGIS pour initialiser les traitements des données géographiques et automatiser leur application en lien avec le site web développé en parallèle par notre équipe. Or, les retours du CoTech nous ont permis de comprendre que QGIS ne permettrait pas de remplir cette fonction, l'utilisation de QGIS Server a alors été proposée.

QGIS Server est un serveur web géospatial puissant et polyvalent qui permet la distribution et la publication de données et de services géospatiaux. Basé sur la plateforme Quantum GIS (QGIS), un système d'information géographique (SIG) open-source, QGIS Server s'intègre parfaitement à l'infrastructure géospatiale existante, offrant une solution évolutive et efficace pour accéder et manipuler des données géospatiales via le web. En tant que serveur WMS (Web Map Service) et WFS (Web Feature Service), QGIS Server permet aux utilisateurs de générer et de fournir des images de cartes dynamiques et des données vectorielles à des clients, facilitant ainsi la création d'applications cartographiques interactives et l'analyse spatiale. QGIS Server se distingue notamment par sa capacité à traiter et à afficher des jeux de données raster, ce qui le rend particulièrement utile pour appliquer des chaînes de traitement géospatiales complexes et afficher des informations basées sur des rasters. Son extensibilité et sa conformité aux normes ouvertes renforcent également son attrait, en permettant l'interopérabilité et la compatibilité avec une large gamme de logiciels et d'applications géospatiales. Avec QGIS Server, les organisations peuvent exploiter son ensemble riche de fonctionnalités pour créer des services web géospatiaux robustes, permettant aux utilisateurs d'accéder et d'explorer les données spatiales de manière transparente et efficace.

Sur le papier, QGIS Server semble donc être l'outil approprié pour répondre à nos besoins, qui consistent globalement à stocker sous forme de Web Service une chaîne de traitement QGIS sous forme d'un « modeleur graphique » afin de pouvoir appliquer cette chaîne de traitement de façon automatisée à des données en lien avec le site web.

Cependant, nous avons rencontré de nombreux problèmes lors de l'installation de QGIS Server sur une des machines de l'école à notre disposition. A titre d'exemples, certains de ces problèmes sont détaillés et expliqués ci-dessous :

- Pour des raisons que nous ignorons, la documentation en lien avec l'installation et le fonctionnement général de QGIS Server est très peu fournie. En particulier dans le cas d'une installation sous Windows. Environ 1 à 2 jours complets de recherches ont été nécessaires rien que pour trouver une plateforme de téléchargement ainsi qu'un guide fiable et relativement récent pour l'installation de QGIS Server (qui comporte un nombre d'étapes très important)
- Une fois le guide d'installation suivi et QGIS Server installé sur la machine, il est nécessaire d'initialiser les serveurs apaches via une plateforme telle que XAMP. Ces serveurs permettent de faire tourner QGIS Server par la suite. Là encore, des problèmes ont été rencontrés, menant à des erreurs chroniques lors des tentatives de lancement des serveurs. Il nous était donc impossible d'accéder à l'interface de QGIS Server via un navigateur.

En voyant que de nombreuses difficultés ont été rencontrées lors de la tentative d'installation de QGIS Server, nous avons décidé de commencer à travailler en parallèle sur une solution annexe potentielle. Nous avons donc séparé notre équipe travaillant sur l'aspect serveur du projet (composée de 2 personnes) en 2. Une personne continuant à essayer d'installer QGIS Server par d'autres moyens qui seront décrit par la suite. Une autre commençant à étudier les possibilités offertes par la bibliothèque javascript « CesiumJS » qui est une solution annexe découverte en cours de projet. Cette dernière permet d'afficher les différents jeux de données et de les manipuler (tracés de coupes, points, polygones …) en agissant directement du côté client.

Une dernière tentative d'installation de QGIS Server a donc été effectuée, par l'intermédiaire du service Docker. Docker est une plateforme de virtualisation légère qui permet l'isolation et la gestion efficace des applications au sein de conteneurs logiciels. Fonctionnant sur la base de la technologie de virtualisation de système d'exploitation (OS-level virtualization), Docker offre une approche modulaire pour le déploiement et la gestion des applications. Les conteneurs Docker encapsulent les applications, ainsi que leurs dépendances et configurations, permettant ainsi de les exécuter de manière cohérente et fiable sur différents environnements, qu'il s'agisse de machines locales, de serveurs distants ou de services cloud. L'utilisation de Docker repose sur la notion de "conteneurisation", où chaque conteneur est isolé et partage les ressources du système hôte de manière efficace. Cette approche permet une utilisation optimale des ressources et une portabilité accrue des applications. Parmi les fonctionnalités principales de Docker, on trouve la gestion des images, qui permet de créer, partager et distribuer des conteneurs préconfigurés, ainsi que le déploiement facile et reproductible d'applications. De plus, Docker offre des mécanismes pour le réseautage entre conteneurs, la gestion des volumes de stockage et l'orchestration des conteneurs à grande échelle grâce à des outils complémentaires tels que Docker

Compose et Kubernetes. En résumé, Docker constitue un outil essentiel dans le domaine de la virtualisation et de la gestion des applications, offrant une approche efficace, flexible et portable pour le déploiement d'applications dans des environnements variés.

Nous avons donc trouvé une image préconfigurée de QGIS Server dans la bibliothèque de conteneurs proposée par Docker. Cette image a été créée par l'entreprise « Camptocamp ». Cette deuxième tentative d'installation a dans un premier temps permis de régler les problèmes rencontrés lors de l'installation dite classique. Cependant, nous avons rencontré d'autres problèmes, détaillés ci-dessous, dans la suite de l'installation :

- Impossibilité d'accéder à l'interface de QGIS Server ainsi qu'à la page nous permettant normalement d'uploader un fichier de type modeleur graphique sur le serveur. Une erreur réseau semble être la cause de ce problème, mais nous avons malheureusement été dans l'impossibilité de la régler.
- L'utilisation de Docker générant une « machine virtuelle » tournant sous le système d'exploitation Linux au sein de notre machine, il est nécessaire d'établir une connexion entre les deux machines pour transférer des données et espérer pouvoir les uploader sur notre serveur. Cette connexion doit s'établir via des lignes de commandes en créant un fichier sur la machine et en créant une image miroir de ce fichier sur la machine virtuelle via l'établissement d'une connexion entre les deux machines par l'intermédiaire de ports spécifiques. L'ensemble de ces étapes ont été traitées avec succès mais pour des raisons que nous ignorons, tout accès aux données nous était impossible via l'interface de QGIS Server.

Ainsi, à la suite de tous les problèmes rencontrés, nous avons décidé d'abandonner l'idée de l'utilisation de QGIS Server afin de nous pencher sur l'utilisation de la solution annexe, CesiumJS. Bien qu'un grand volume horaire de travail passé sur les différentes tentatives d'installation de QGIS Server ne porte pas ses fruits, le choix de la séparation de l'équipe pour travailler sur deux solutions potentielles en parallèle nous a permis de ne pas repartir de zéro et de gagner un temps précieux, la date de rendu du projet arrivant à grands pas.

CesiumJS est une bibliothèque JavaScript open-source spécialisée dans la visualisation de données géospatiales en 3D sur le Web. Son fonctionnement repose sur la puissance du WebGL, une technologie permettant le rendu graphique accéléré par le processeur graphique dans les navigateurs web modernes. CesiumJS offre une panoplie de fonctionnalités permettant de créer des applications géospatiales immersives et interactives. Parmi ses fonctionnalités principales, on trouve la capacité de charger et de visualiser des données géospatiales à partir de diverses sources telles que des fichiers géospatiaux, des services web ou des flux de données en temps réel. CesiumJS prend également en charge l'affichage de rasters, permettant ainsi l'affichage et l'analyse de données sous forme d'images matricielles. De plus, cette bibliothèque offre des outils avancés pour l'interrogation spatiale, la mesure de distances et d'aires, ainsi que la création d'animations et de visualisations 3D. CesiumJS se distingue également par sa compatibilité multiplateforme, étant capable de s'exécuter sur divers navigateurs web et dispositifs mobiles. Grâce à son architecture modulaire et sa documentation complète, CesiumJS facilite le développement d'applications géospatiales personnalisées et la création de chaînes de traitement complexes pour l'analyse et la visualisation de données géospatiales. En résumé, CesiumJS est un outil incontournable pour les développeurs et les scientifiques

travaillant avec des données géospatiales, offrant des fonctionnalités avancées pour la visualisation immersive en 3D, l'analyse spatiale et la manipulation de rasters.

Le choix de CesiumJS est pertinent dans le cadre de notre projet en tant qu'alternative à l'aspect SIG adopté dans un premier temps. En effet, ce dernier permet de se passer du traitement des différents MNT ce qui nous a permis de résoudre un problème majeur qui est la manipulation de MNT de plusieurs dizaines de gigas sur une plateforme web. Par ailleurs, CesiumJS supporte l'import de données nativement et contient une interface qui est intuitive pour l'utilisateur. Et le dernier critère qui nous a orienté vers cette architecture est la documentation de la librairie qui est complète et mise à jour avec une communauté active et un forum qui recense la plupart des erreurs rencontrées à l'image de Stackoverflow.

3.2) Architecture serveur finale

Voici un schéma illustrant l'architecture serveur finale de notre projet, cette figure est expliquée dans les paragraphes suivants :

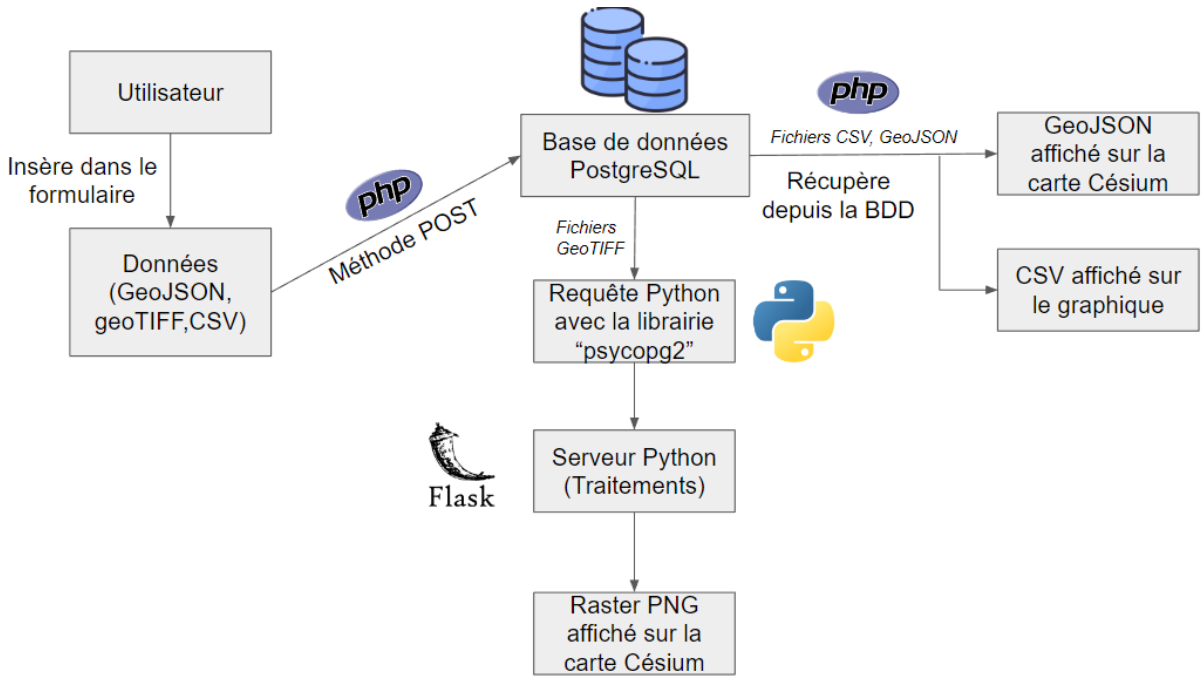

Figure 9 : Architecture finale du serveur du projet

Utilisation de PHP et de PostgreSQL pour la gestion d'une base de données

Comme expliqué précédemment, notre architecture serveur manipule trois types de données : des GeoTIFF et des GeoJSON et des CSV, qui sont les formats attendus par la plateforme web de la part de l'utilisateur.
Ces données seront stockées dans une base de données PostgreSQL.

PostgreSQL est un système de gestion de bases de données relationnelles open-source reconnu pour sa fiabilité et sa puissance. Il offre une grande flexibilité pour le stockage et la gestion des données, en prenant en charge des types de données diversifiés. Parmi les extensions les plus populaires de PostgreSQL, on trouve PostGIS, une extension géospatiale qui étend les fonctionnalités du système en permettant le stockage, la gestion et l'analyse de données géographiques. PostGIS est particulièrement utile pour stocker des données de type raster et GeoJSON, permettant ainsi la manipulation et l'affichage efficace de ces données sur des sites web. Son fonctionnement repose sur des fonctions et des index spatiaux avancés, permettant des requêtes géospatiales performantes telles que l'interrogation par la localisation, l'intersection, la distance et bien d'autres encore. Grâce à la combinaison de PostgreSQL et de PostGIS, les utilisateurs bénéficient d'une solution complète et robuste pour la gestion et l'analyse de données géospatiales, ainsi que pour l'affichage de ces données sur des sites web, ce qui en fait un outil indispensable pour les applications géospatiales professionnelles et scientifiques.

Afin de travailler en local et de pouvoir tester fréquemment l'avancée du développement du site web, nous avons utilisé l'environnement WAMPServer, qui présente l'avantage de très bien fonctionner (après configuration) avec les bases de données PostgreSQL.

Pourquoi PostgreSQL et pas MySQL ?

En plus de toutes ces fonctionnalités utiles dans le cadre de notre projet, nous avons choisi de travailler avec PostgreSQL car c'est un outil qui est beaucoup utilisé dans le milieu professionnel, notamment dans les entreprises ou organisations qui prônent l'utilisation de logiciels open-source. Nous avons donc jugé qu'il était intéressant pour nous d'apprendre à utiliser cet outil dans le cadre du développement d'un projet à une échelle relativement grande.
De plus, MySQL est beaucoup moins adapté que PostgreSQL pour le traitement de données géographiques de type GeoTIFF et GeoJSON.

Interactions avec le côté client

Dans un premier temps le PHP récupère le fichier depuis le formulaire client en utilisant la méthode POST et le poste dans la BDD PostgreSQL dans une colonne de type "jsonb" pour les GeoJSON et "bytea" pour les GeoTIFF, le tout dans des tables SQL propres aux différents types de données.
On récupère ensuite avec des requêtes SQL avec plusieurs fichiers PHP (un pour chaque type de données) pour finalement "fetch" ces derniers avec JavaScript afin de les récupérer côté client.

La gestion du serveur calculateur : comment calculer des hauteurs de neige sur un raster ?

Les GeoJSON (données relatives aux enneigeurs, pistes de ski, anémomètres et sondes météo) peuvent être utilisées directement dans Cesium grâce à son support natif. En revanche, ce n'est pas le cas pour les couches d'épaisseur de neige GeoTIFF.

Nous avons essayé d'utiliser la librairie "geotiff.js", mais les coordonnées renvoyées n'étaient pas assez précises pour être exploitables, de plus, le GeoTIFF devait être en local.

L'alternative du serveur Python :
Nous avons donc adopté une solution qui consiste à charger le GeoTIFF du côté serveur à l'aide de **FLASK, la solution serveur de Python**.

Pourquoi FLASK ?

FLASK est une solution serveur légère et flexible pour Python qui permet de facilement créer des applications web. Elle est particulièrement bien adaptée pour utiliser des librairies Python du côté serveur. FLASK offre une grande flexibilité dans la configuration de l'application et la gestion des routes, ce qui facilite l'intégration de librairies tierces.

Cette flexibilité sur l'utilisation des différentes librairies Python est utile pour le traitement des images, en effet Python est souvent utilisé dans ce cadre-là.

Quels traitements sont réalisés par FLASK ?

Parmi ces traitements, le plus important est celui qui va **extraire la donnée d'épaisseur de neige présente dans notre GeoTIFF** en exploitant les coordonnées cliquées sur la carte Cesium.
Python récupère également la bounding box du GeoTIFF pour créer un rectangle sur Cesium, sur lequel nous étalons l'image PNG obtenue pour l'afficher sur la carte.
FLASK permet aussi d'assurer les fonctionnalités du site telles que le calcul du profil pour l'obtention du volume de neige sur la surface d'un polygone dessiné sur la carte.

Pour cela, on utilise les coordonnées des points cliqués par l'utilisateur sur la carte Cesium qui sont envoyées vers le serveur Python. Ensuite, le serveur va chercher la donnée d'épaisseur de neige dans le GeoTIFF situé dans la base de données.
Cesium utilise le système de coordonnée WGS84 (EPSG : 4326) et les données des commanditaires utilisent le CC 45 (EPSG : 3945). On utilise donc la libraire "proj" du côté JavaScript et Python selon la tâche à effectuer.

Enfin, FLASK assure l'application de la palette de couleur fournie par nos commanditaires qui nécessite la **transformation du GeoTIFF en PNG**, qui est mieux supporté par Cesium.

Remarque : Une alternative serait d'utiliser le service Cesium ION, qui possède une RESTful API permettant d'afficher des GeoTIFF directement sous forme de tuiles. Cependant, cette solution est payante et repose sur un service externe, ce qui peut poser des problèmes de coûts et de dépendance.

3.3) Pistes d’amélioration

Amélioration de la Sécurité :

La sécurité est un aspect critique de toute application Web, et il est important de s'assurer que les données stockées dans la base de données sont protégées contre les attaques malveillantes. Une piste d'amélioration consiste à renforcer la sécurité du serveur en utilisant des techniques de cryptage pour stocker les informations sensibles, telles que les identifiants et les mots de passe des utilisateurs ou les données des stations de ski (les GeoTIFF et les GeoJSON).

Il est par ailleurs important de s’assurer que l’utilisateur ne puisse pas accéder aux données des autres stations de ski. On peut également mettre en place un pare-feu pour limiter l'accès aux ressources du serveur uniquement aux adresses IP autorisées. De plus, l'utilisation d'un protocole de sécurité comme HTTPS peut améliorer la confidentialité et l'intégrité des communications entre le client et le serveur.

Il est à noter que l’encryptage des données des stations qui sont des données géospatiales peut induire des erreurs sur celles-ci lors de la récupération du côté client. En effet s’il n’y a pas moyen de décrypter la donnée sans l’endommager, le processus n’est peut-être pas pertinent pour ces données-là.

Optimisation des performances :

La performance est un autre aspect important de l'architecture serveur, car elle peut affecter l'expérience utilisateur et la satisfaction des clients. Comme on peut le voir sur le schéma il y a une partie sur le traitement des données GeoTIFF du côté serveur à l’aide de Python et de son environnement FLASK. Ce traitement est conséquent mais est nécessaire pour le bon fonctionnement de l’affichage de la plaquette de couleur sur notre GeoTIFF ainsi que d’autres traitements en fonction des outils désirés par l’utilisateur sur le site. Une piste pour l’amélioration de cette étape serait d’améliorer l’algorithme en général et l’optimiser pour de la donnée qui va être de plus en plus importante en termes de poids et d’espace.

Ensuite, on peut utiliser des outils tels que le cache côté serveur pour stocker temporairement les données fréquemment utilisées et accélérer les temps de réponse. L'utilisation de techniques de mise en cache et de compression de fichiers peut également réduire le temps de chargement de la page, améliorant ainsi l'expérience utilisateur.

Amélioration de l'évolutivité :

L’évolutivité peut permettre à l'application de s'adapter à des volumes de données plus importants et à des charges de trafic plus élevées. Une piste d'amélioration consiste à adopter une architecture distribuée pour permettre la répartition de la charge de travail sur plusieurs serveurs. Cela peut être réalisé en utilisant des solutions de conteneurisation

telle que Docker, ou en utilisant une plateforme de « cloud computing » telle que AWS, Azure ou Google Cloud Platform.

En utilisant une architecture distribuée, on peut également améliorer la tolérance aux pannes en permettant aux nœuds individuels de prendre en charge la charge de travail si l'un d'entre eux tombe en panne.

IV) Réalisation et suivi de projet

4.1) Retour sur le calendrier prévisionnel : diagramme de GANTT

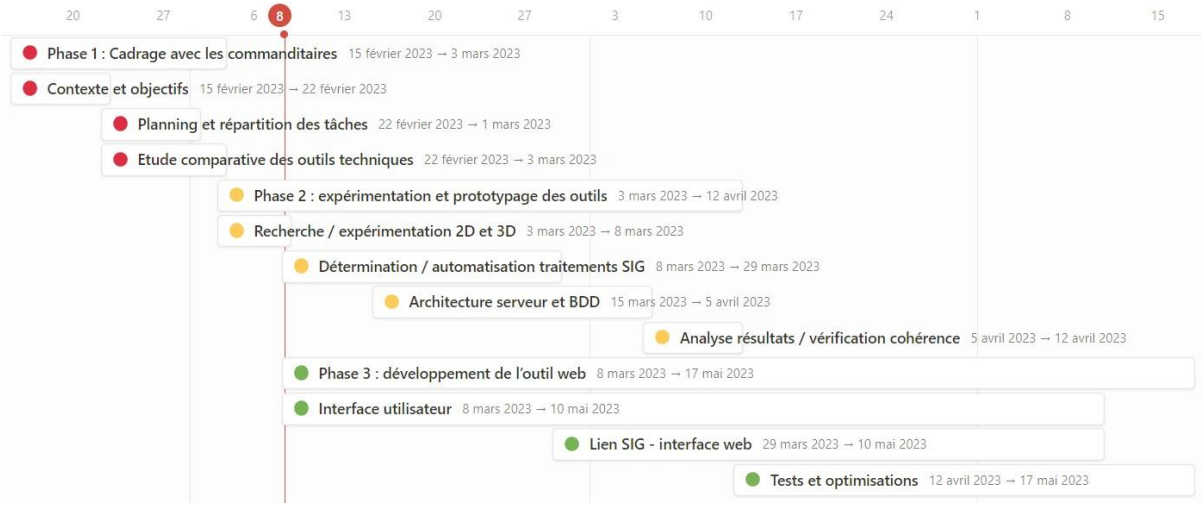

Figure 10 : Diagramme de GANTT - Représentation temporelle des tâches du projet

Le document ci-dessus est le diagramme de Gantt de notre projet.
La phase 1 de définition des besoins et de **cadrage du projet** fut rapidement achevée grâce à une bonne et fréquente communication avec nos commanditaires durant les premières séances.

La phase 2 correspond à la **réalisation des méthodes de visualisation 2D et 3D, le développement des méthodes de calcul des hauteurs de neige**, ainsi qu'à la **sauvegarde des données dans une BDD** via la construction d'une **architecture serveur**.
La phase 3 est portée sur le **développement de l'interface web** : d'une part l'interface utilisateur, puis l'ajout des calculs implémentés dans l'architecture serveur de la phase 2 à l'interface Web, et enfin des tests et des optimisations.

Les phases 2 et 3 du projet ont été effectuées de façon **quasi parallèle**.

Notons que la phase 2 a été très prolongée notamment à cause de nombreuses complications à la suite de nos choix des différents outils à utiliser. La piste de l'outil QGIS Server, qui n'a pas abouti comme expliqué dans la partie précédente, nous a fait perdre plusieurs semaines. De plus, l'imbrication des réalisations de la phase 2 (côté serveur) dans l'interface Web (côté client) a pris plus de temps que prévu à cause de nombreuses complications (problème de volume de données, d'encodage entre PHP et Javascript …).

Chacune des deux dernières phases se termine par une étape de test ou d'analyse qui a permis de vérifier que le travail effectué dans la phase correspondante remplit l'intégralité du cahier des charges de nos commanditaires.

Concernant les outils informatiques utilisés par notre équipe projet, nous avons utilisé premièrement un repository GitHub pour avoir un suivi au niveau du code et avoir un moyen de communiquer l'avancement du code avec les commanditaires. Ensuite nous avions un Google drive où nous communiquions au sein du groupe au sujet des différentes étapes du projet (CoPIL et CoTech par exemple) et où nous déposions de la documentation.
Enfin, nous avions un OneDrive partagé avec les commanditaires qui contient les différentes données nécessaires à la réalisation du projet (rasters, GeoTIFF, csv).

4.2) Retour sur la répartition des tâches

Notre équipe projet a été également répartie entre les deux phases avec deux responsables de l'architecture serveur travaillant sur la phase 2, ainsi que deux développeurs de l'interface web (phase 3).

<table>
<tr><th>Nom</th><th>Benoit
CACHARD</th><th>Hannick
ABDUL
KUTHOOS</th><th>Bryan
CANTAL</th><th>Malo
DE LACOUR</th></tr>
<tr><td>Rôle</td><td>Chef de projet

Gestion du coté serveur</td><td>Développeur

Gestion du coté serveur</td><td>Développeur

Programmeur Web : UX Designer</td><td>Développeur

Programmeur Web : UI Designer</td></tr>
<tr><td rowspan="2">Tâches effectuées</td><td rowspan="2">Responsable de la recherche de méthodes de visualisation et de la gestion de la BDD</td><td rowspan="2">Responsable du développement des méthodes de calcul de hauteur de neige</td><td colspan="2">Responsables du développement de l’interface Web

Sections implémentées :</td></tr>
<tr><td>Tableau de bord
Importation des données
Production des enneigeurs</td><td>Menu déroulant
Suivi du domaine</td></tr>
<tr><td>Compétences</td><td>Gestion d’équipe
PostgreSQL</td><td>PHP
Python
Javascript</td><td>Javascript
HTML/CSS
PHP</td><td>Javascript
HTML/CSS</td></tr>
</table>

Figure 11 : Tableau détaillant la répartition des tâches

Cette répartition a été conçue de sorte à permettre une grande adaptabilité en cas de problème de volume de travail des différentes tâches.
Dans le cas des complications de réalisation des tâches par l’un des membres de l’équipe projet, cette répartition a temporairement été modifiée afin de faire avancer le projet de façon optimale.
Voici des exemples de modifications temporaires des tâches qui ont eu lieu :

- Assistance occasionnelle des développeurs client par le côté serveur notamment lors de complications d’importation des méthodes de calcul et des fichiers depuis la BDD ou le serveur Python (problèmes d’encodage, de format …)
- Les développeurs client ont pu constater un temps de calcul trop long lors de requêtes vers le côté serveur. Cela a donc mené à une assistance des développeurs serveur par le côté client pour optimiser la gestion de la BDD ou les méthodes de calcul des hauteurs de neige.

En cas de problèmes dont aucun des membres n’avait la solution, nous avons parfois pris l’initiative d’en informer nos commanditaires ou de contacter un professeur pour une aide supplémentaire.

4.3) Analyse stratégique : les risques rencontrés

Dans le cadre de la gestion de notre projet, nous avions initialement établi un tableau des risques afin de déterminer, et donc d'anticiper, les différents problèmes pouvant avoir lieu durant notre projet.

Voici les risques principaux qui se sont produits durant notre projet :

Nature du risque arrivé	Probabilité	Conséquence	Solution trouvée	Evolution
Complications éventuelles suite à nos choix d'outils (QGIS Server)	**Forte**	Ralentissement du projet, éventuellement changement d'outil	S'orienter entièrement sur une des solutions à notre disposition : Cesium.js	**Moyenne**
Bug au milieu de la chaîne de traitement lors des méthodes de calcul	**Moyenne**	Interruption du traitement : pas de données	Subdiviser la chaîne de traitements. Exemple : création d'un serveur Python	**Moyenne**
Incohérences, erreurs de calcul	**Moyenne**	Mauvaise interprétation des graphes, des rasters etc..	Calcul d'incertitude et utilisation d'outils statistiques pour détecter et déterminer la source de l'erreur	**Faible**
Problème de gestion de la base de données (format des données)	**Forte**	Impossibilité d'effectuer l'affichage des données sur l'interface web	Recherche de méthodes d'encodage en binaire adaptées afin d'inclure les fichiers bases de données	**Faible**

Figure 12 : Tableau détaillant les principaux risques apparus au cours du développement du projet

4.4) Analyse stratégique : les menaces

La réalisation de ce projet a été soumise à quelques menaces :

- La mauvaise compréhension de l'architecture du site web en termes de liaison entre le front end et le back end. Il a donc été nécessaire d'assurer une communication importante entre les développeurs client et serveur.
- La gestion d'un grand volume de données, étant donné le fait que l'on ait rarement manipulé des données aussi conséquentes au niveau d'un site web.
- La focalisation sur le détail et non l'essentiel du projet, qui est un risque courant dans les projets de développement.

Conclusion

Ce rapport final d'analyse informatique a permis de présenter le contexte et les objectifs du projet développement en partenariat avec ELDA Technology. Mais également de présenter les choix stratégiques et techniques déjà réalisés, et notamment les choix des outils et logiciels. Les langages de programmation utilisés durant la réalisation de ce projet ont également été présentés et leurs choix justifiés.

Ce rapport présente également les différents outils utilisés pour la gestion de projet en lui-même aussi bien en termes de gestion d'équipe, du temps ou de réalisation des différents choix stratégiques.

Pour répondre à son besoin de créer une plateforme web permettant l'affichage 3D et 2D de leurs données, l'équipe de ELDA TECHNOLOGY a décidé de se tourner vers une solution entièrement Open Source.
De plus, le fait de réaliser en 10 semaines un site Web entier avec une architecture serveur (PHP, Python, BDD) et d'assurer leur relation a été un défi important pour notre équipe de développeurs, notamment en termes d'utilisation de certains outils.
Malgré cela, notre équipe projet est satisfaite car elle est parvenue à implémenter toutes les fonctionnalités demandées par nos commanditaires, tout en travaillant en coopération de façon précise et efficace dans une ambiance de travail sereine.

Annexes

Glossaire :

Neige artificielle/neige de culture : Neige dont la formation est provoquée volontairement par l'homme au moyen d'un canon à neige. La neige artificielle est utilisée dans un nombre croissant de stations de ski.

MNT : Un modèle numérique de terrain (MNT) est une représentation de la topographie (altimétrie et/ou bathymétrie) d'une zone terrestre (ou d'une planète tellurique) sous une forme adaptée à son utilisation par un calculateur numérique (ordinateur).

Enneigeurs : Un canon à neige ou enneigeur est un dispositif permettant de fabriquer de la neige mécaniquement à partir d'eau et d'air, le tout à basses températures (0 °C et moins). Le principe est de projeter un mélange d'air comprimé et d'eau par temps suffisamment froid.

Modeleur graphique : Outil présent sous le logiciel SIG QGIS qui permet l'automatisation du traitement successif de différentes tâches réalisées sur les différentes couches manipulées.

Dameuse : Une dameuse est un véhicule sur chenilles spécialement conçu pour améliorer la qualité de la neige pour la pratique des principaux sports de glisse d'hiver, tels que le ski ou le snowboard. Elle est utilisée dans les domaines skiables sur les pistes de ski ou de motoneiges.

Liens utiles :

Installation de QGIS Server :

- https://docs.qgis.org/3.22/en/docs/server_manual/getting_started.html#installation-on-windows
- https://www.sigterritoires.fr/index.php/installer-qgis-server-sous-windows-10/

QGIS Server – image docker Camptocamp :

- https://hub.docker.com/r/camptocamp/qgis-server

Documentation – fonctionnement CesiumJS :

- https://cesium.com/learn/cesiumjs/ref-doc/

Aide développement site web – bibliothèque CSS :

- https://uiverse.io/all