



ÉCOLE NATIONALE
DES SCIENCES
GÉOGRAPHIQUES

Ecole Nationale des
Sciences Géographiques



Université Gustave Eiffel

Projet ANR Oracles

Visualisation de scénarios de submersion côtière

Equipe :

BARAN ILONA
FRÉDOC LÉA
RICO QUINTERO FERNANDO
WILLIAMS JACQUELINE

Commanditaires :
CHRISTOPHE SIDONIE
GAUTIER JACQUES



TSI 2022

Mars - Avril 2023

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

Table des matières

Glossaire	4
1 Définition du projet	5
1.1 Contexte	5
1.2 Simulation et scénario	5
1.3 Objectifs et besoins des utilisateurs	5
2 Gestion de projet	7
2.1 Dates limites	7
2.2 Organisation et méthode Agile	7
2.3 Diagramme de Gantt	7
2.4 Risques	8
3 Analyse Fonctionnelle	9
3.1 Données et cas d'études	9
3.1.1 Données de la zone de Gâvres	9
3.1.2 Données de la zone d'Arcachon	9
3.1.3 Prétraitement des données	10
3.2 Maquettes et data visualisation	10
3.2.1 Maquettes du site	10
3.2.2 Styles de visualisations de données	11
3.3 Diagramme de classes	14
3.4 Diagramme d'activités	16
4 Analyse Technique	17
4.1 Environnement de Développement	17
4.1.1 VCS	17
4.1.2 IDE	17
4.1.3 CI/CD	17
4.2 VTThree	17
4.3 Architecture	18
4.4 Langages et Technologies	19
4.4.1 VueJS	19
4.4.2 ThreeJS	19
4.4.3 iTowns	19
4.4.4 Stockage des données	20
4.4.5 Python et Flask	20
4.4.6 Docker	20
5 Présentation de l'application	21
5.1 Fonctionnalités et solutions réalisées	21
5.1.1 Sélection des scénarios	21
5.1.2 Visualisation 2D/3D	21

5.1.3	Changement fond de plan	22
5.1.4	Affichage hauteur d'eau sur la carte	22
5.1.5	Affichage et masque des bâtiments	23
5.1.6	Visualisation des données	24
5.2	Solutions envisagées	24
5.3	Limites	25

Conclusion	26
-------------------	-----------

Glossaire

API Application Programming Interface

ANR Oracles ..

BRGM Bureau de Recherches Géologiques et Minières

CI/CD Continuous Integration/Continuous Delivery

CNRS Centre National de la Recherche Scientifique

DocJS Générateur de documentation JavaScript

ENSG Ecole Nationale des Sciences Géographiques

GeoJSON GeoJSON est un format d'encodage de données géographiques dans le format d'échange de données JSON.

GeoTIFF GeoTIFF est un format de fichier d'image géoréférencée, basé sur le format de fichier TIFF (Tagged Image File Format)

Git Git est un logiciel de gestion de version de code source, souvent utilisé pour le développement de logiciels.

IDE Integrated Development Environment

IGN Institut National de l'Information Géographique et Forestière

itowns iTowns est une bibliothèque logicielle open source développée par l'IGN

JSON JavaScript Object Notation

LASTIG Laboratoire des Sciences et Technologies de l'Information Géographique

MNS Modèle Numérique de Surface

Orthophoto Une orthophoto est une image aérienne ou satellite d'un terrain qui a été corrigée pour prendre en compte les distorsions dues à la perspective de la prise de vue.

PrimeVue PrimeVue est une bibliothèque de composants d'interface utilisateur pour Vue.js, qui propose une collection de composants préfabriqués pour la construction d'applications web réactives.

ThreeJS Three.js est une bibliothèque JavaScript open-source qui permet de créer des graphiques 3D interactifs et animés dans un navigateur web

TSI Technologies des Systèmes d'Informations

UI User Interface

UMR Unité mixte de recherche

VTThree Vector Tiles Three

VCS Version Control System

VueJS VueJS est un framework JavaScript open-source pour la construction d'interfaces utilisateur interactives et réactives.

WMS Web Mapping Service

.dat ".dat" est une extension de fichier qui est souvent utilisée pour indiquer un fichier de données.

.mat Le format de fichier .mat est un format de fichier binaire utilisé par MATLAB

DÉFINITION DU PROJET

1.1 Contexte

Le projet ORACLE de l'Agence Nationale de la Recherche (<https://anr.fr/Projet-ANR-21-CE04-0012>) a pour objectif de développer des méthodes de prévision spatialisée du phénomène de submersion côtière à partir de données sur les conditions météo-océaniques, issues du système de prévision d'ensemble de Météo-France. Ces prévisions d'ensemble consistent à rendre compte des incertitudes présentes dans les prévisions météorologiques. Les résultats du projet ORACLES pourraient aider les experts et les gestionnaires de crise à mieux comprendre les risques de submersion marine et à prendre des mesures préventives pour minimiser les impacts sur les populations et les infrastructures.

Notre équipe projet est constituée de quatre élèves de la filière 'Technologie des Systèmes d'Informations' de l'École Nationale des Sciences Géographiques - Géomatique. Nous avons été invités à contribuer au projet Oracles en proposant un prototype de Visualisation 2D/3D de scénarios de submersion côtière sur les sites du Bassin d'Arcachon en Gironde et la ville de Gâvres dans le Morbihan.

1.2 Simulation et scénario

Les visualisations que nous cherchons à réaliser se basent sur des simulations de scénarios de submersion côtière, rattachés à différentes conditions météo-océaniques. Un scénario correspond à un évènement historique passé.

1.3 Objectifs et besoins des utilisateurs

Le projet ORACLE doit répondre à des problématiques en terme de visualisation, notamment :

- co-visualiser conditions météo-océanique en entrée, scénarios de submersion en sortie ;
- co-visualiser différents scénarios (car plusieurs scénarios dans le cadre de prévisions d'ensemble).

Pour notre projet, nous avons pour objectif de principal de construire un prototype de visualisation pour l'exploration de différents styles et approches d'interaction dans la suite de l'ANR Oracles. Ce prototype servira de socle pour de futurs travaux dans le cadre du projet ANR Oracles.

Les objectifs de notre projet comprennent donc de :

- Proposer et développer un premier prototype de visualisation 2D/3D, permettant d'intégrer les données d'entrée/et de sortie de modèles de simulation, et de la visualisation ;
- Permettre l'intégration et la visualisation de plusieurs scénarios ;
- Utiliser ThreeJS comme moteur de rendu 3D.

Nous avons réalisé un diagramme des cas d'utilisation (figure 2.1) de notre application pour détailler les deux premiers objectifs. L'utilisateur doit pouvoir facilement naviguer du site de Gâvres à celui d'Arcachon. Il doit également pouvoir visualiser la scène en 2D ou 3D et afficher des données (fond de carte, bâtiments, hauteur d'eau minimale et maximale). Enfin, il doit pouvoir sélectionner un ou plusieurs scénarios, et co-

visualiser hauteurs d'eau issues des modèles de simulation et conditions météo-océaniques associées sur une représentation cartographique et différents graphiques.

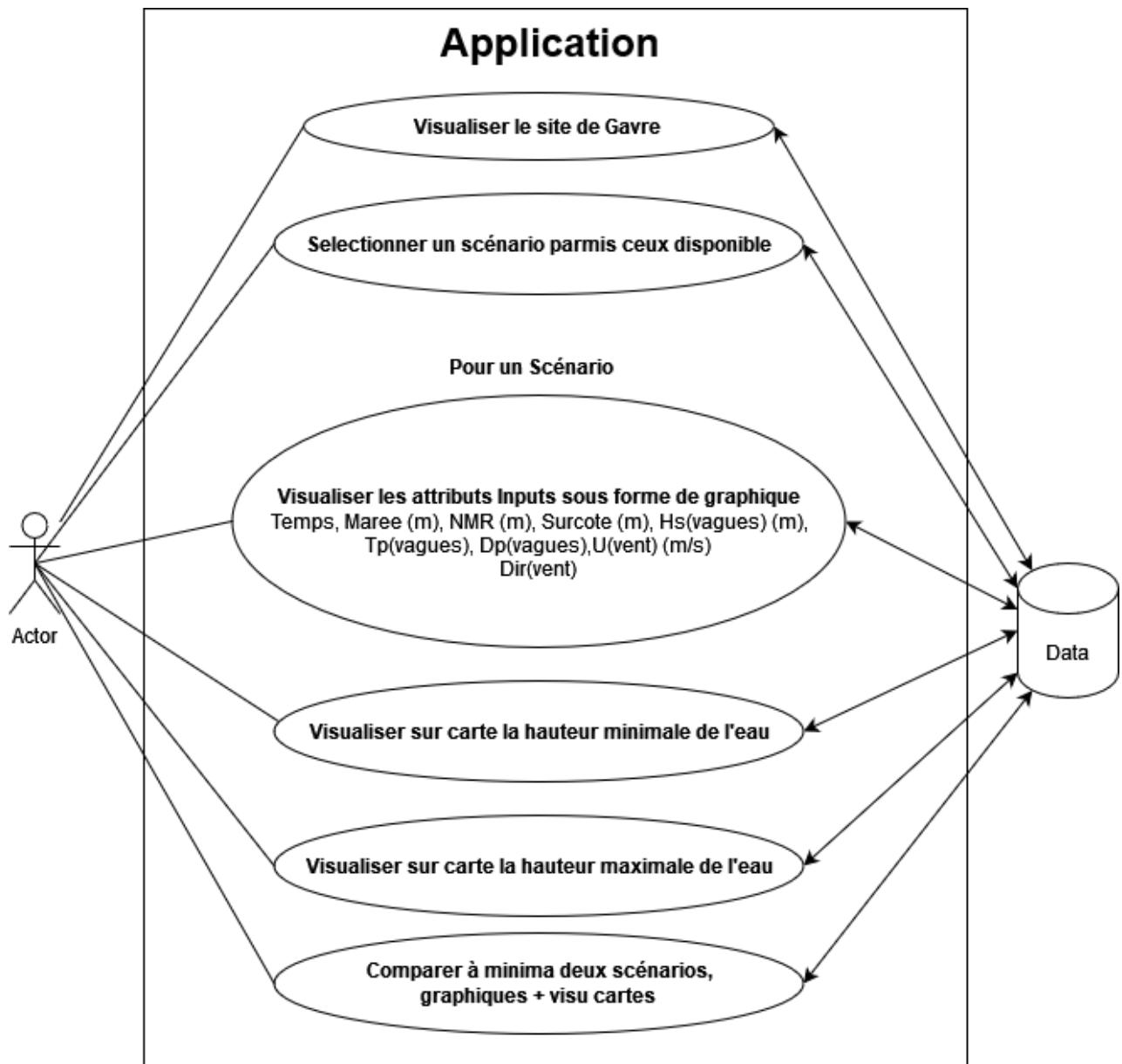


FIGURE 1.1 – Diagramme des cas d'utilisation

GESTION DE PROJET

2.1 Dates limites

Le projet se tiendra sur une durée de six semaines. Les phases d'analyses et de développement du projet seront comprises entre les dates du 17 mars et du 23 avril. La dernière semaine, du 24 avril au 27 avril, sera consacrée à la réalisation des livrables, de la documentation et de la préparation à la soutenance. La soutenance de ce projet se tiendra le 28 avril.

2.2 Organisation et méthode Agile

La méthode Scrum est une méthodologie agile. Nous avons utiliser la méthode Scrum car nous avons besoin de réaliser une application fonctionnelle en peu de temps et cette méthode est adaptée pour la gestion de développement logiciel et que celui-ci doit être réalisé rapidement.

Cette méthode fonctionne par des cycles itératifs appelés « Release ». Chaque Release est constituée de "sprints". Nos sprints duraient une semaine et a chaque fin de sprint nous publions un release. Les sprints sont des périodes au cours desquelles l'équipe de développement exécute un ensemble de tâches appelées les « stories ». Les stories étaient définies en début de projet par le product Owner assisté par l'équipe. A chaque fin de sprint, nous définissions le travail fait et celui qui sera réalisé la semaine suivante. Conformément à la méthode Agile Scrum, nous participions aussi à des cérémonies quotidiennes animées par le Scrum Master.

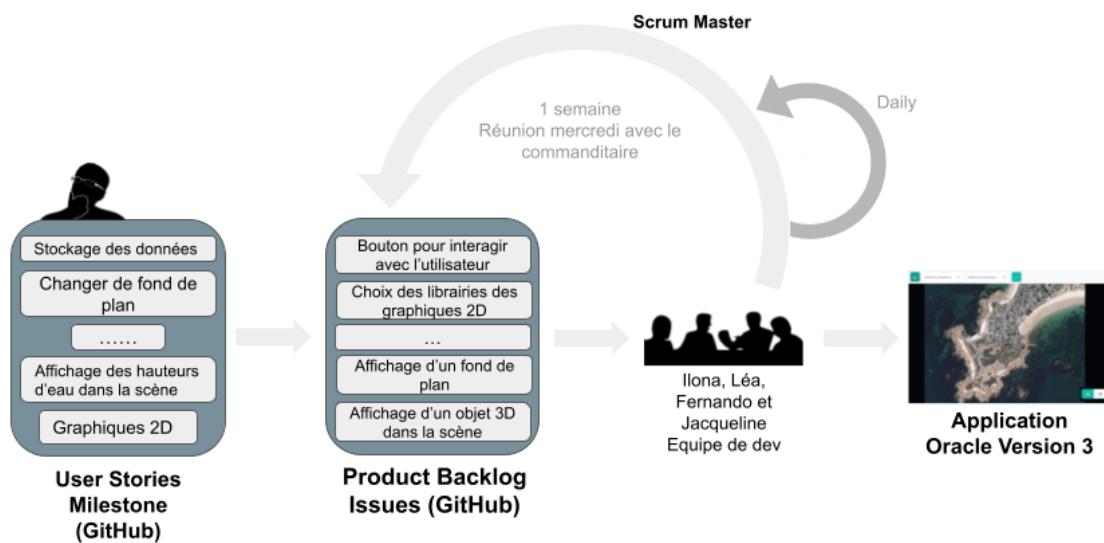


FIGURE 2.1 – Schéma de principe Méthode Agile Scrum : Version 3 de l'application

2.3 Diagramme de Gantt

Travaillant en méthode Agile Scrum, nous n'avons pas réalisé de diagramme prévisionnel lors de la phase d'analyse. Le diagramme suivant (2.3) présente le diagramme de Gantt du projet.

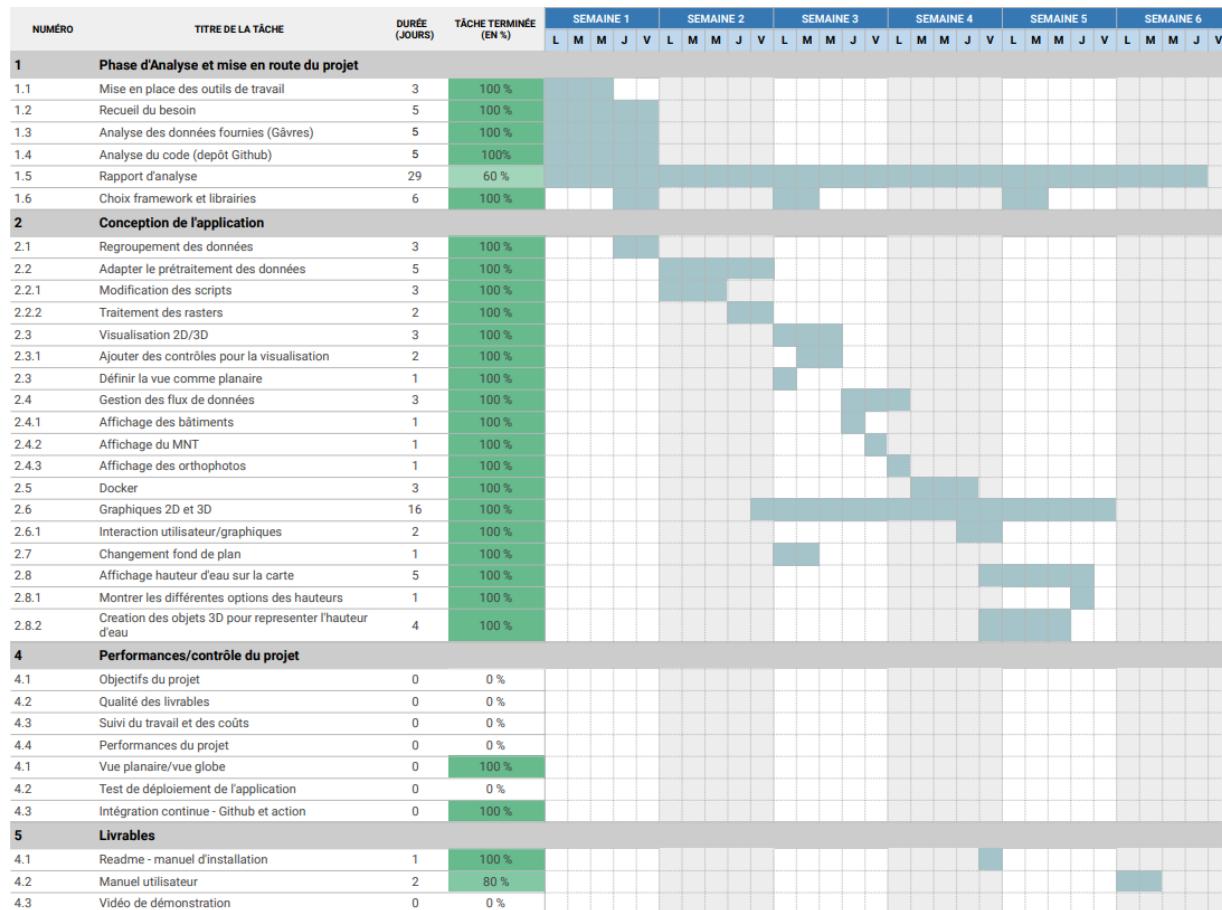


FIGURE 2.2 – Diagramme de Gantt

2.4 Risques

Connaître et anticiper les risques permet de minimiser les impacts qu'ils peuvent avoir sur le déroulement du projet.

Nous nous sommes demandés quels pouvaient être les différents risques encourus lors de ce projet lors de la phase d'analyse. Nous avons complété ce tableau jusqu'à la troisième semaine.

NUMÉRO	TITRE DE LA TÂCHE	IMPACT	PROBABILITÉ	CRITICITÉ	CONSÉQUENCES
1					
1.1	Utilisation de nouvelles technologies	Majeur	Très probable	Modéré	Allongement du temps des tâches concernées
1.2	Ressources matérielles limitantes	Grave	Peu Probable	Modéré	Tronquage des données
1.3	Accès aux données	Majeur	Peu Probable	Très critique	Rien n'est affiché sur la page
1.4	Publication des données sensibles	Grave	Probable	Très critique	Des informations sensibles peuvent être divulguées
1.5	Erreurs dans le prétraitement	Grave	Peu Probable	Très critique	Les données affichées peuvent être erronées
1.6	Bas performance de l'outil	Majeur	Peu Probable	Modéré	Temps d'attente plus long pour l'utilisation
1.7	Scalabilité de l'application	Mineure	Peu Probable	Critique	L'application ne supportera pas une grande quantité d'informations.

FIGURE 2.3 – Tableau de risques

ANALYSE FONCTIONNELLE

Dans cette partie, nous décrirons les grandes fonctionnalités identifiées pour répondre aux besoins utilisateur. Cette partie détaille la logique dont notre logiciel a besoin pour fonctionner et pas forcement la technologie qui a été utilisé pour le faire. Ces modélisations et diagrammes ont évolué au cours de l'implémentation des solutions proposées du projet. Par la suite, nous avons tenu au courant nos commanditaires des modifications apportées et de nos choix.

3.1 Données et cas d'études

3.1.1 Données de la zone de Gâvres

Pour réaliser notre prototype, notre commanditaire nous a fourni des données pour le site de Gâvres. Toutes ces données sont confidentielles et n'ont pas pu être partagées sur Git ou mises dans une base de données.

Parmi ces données se trouve d'une part des données spatialisées et non temporelles. Elles représentent les données de sortie des simulations de submersion et traitent des hauteurs d'eau sur la zone. D'une autre part nous avons des données non spatialisées mais temporelles. Elles représentent les données d'entrée des simulation et traitent de la direction et vitesse du vent, de la direction et hauteur des vagues, de la hauteur de la marée ou encore de la surcote.

- un dossier Zone_extract qui contient un geotiff (Zone_Extract_Flood_v3.tif) définissant la zone d'intérêt (Lambert 93) ;
- un fichier .xlsx utilisé pour le suivi des simulations afin d'identifier quels scénarios correspondent à des évènements historiques ;
- un dossier contenant les données d'entrées et sorties pour chaque scénario S_"i" suivant la structure suivante :
 - S_"i"/IN_OUT/INPUT contient :
 - X.dat : fichier pour retrouver à quel évènement correspond le scénario ;
 - X.mat et Xt.dat : séries temporelles des données X (sur 6h, centré sur la pleine-mer).
 - S_"i"/IN_OUT/OUTPUT : contient :
 - les fichiers hmax_land.mat (hauteur d'eau maximal à terre) et hfin_land.mat (hauteur d'eau finale à terre) . Ces fichiers sont composés d'une matrice obtenue de la manière suivante :
 - extraction des hauteurs d'eau maximales en chaque pixel au cours des 6h de simulation. On obtient une matrice de la zone d'intérêt avec des valeurs en mer ;
 - un filtre avec le fichier Zone_Extract_Flood_v3.tif : les pixels en dehors de la zone d'intérêt à terre prennent une valeur de 0.
- un fichier .nc qui contient les longitudes et latitudes (Coordonnées en Lambert93) correspondant aux matrices résultats hmax.

3.1.2 Données de la zone d'Arcachon

Les données d'Arcachon n'étaient pas encore disponibles au démarrage de notre projet. Après réunion avec notre commanditaire au bout de quelques semaines, nous avons décidé de nous concentrer uniquement

sur Gâvres. Le temps d'appropriation, de traitements et de choix de visualisaiton des données seraient trop court et n'auraient pas permis d'exploiter au maximum les données de Gâvres.

Les données d'Arcachon sont dans des formats différents (format NetCDF dont les traitements peuvent parfois être complexes) avec des mesures différentes. Ces mesures sont, de plus, spatialisées et temporelles contrairement aux données d'entrée de Gâvres.

3.1.3 Prétraitement des données

Pour pouvoir exploiter les données plus facilement, nous avons choisi de modifier le format des données. Cette étape de prétraitement a été réalisée une fois en début de projet et nous a permis d'exploiter les données de Gâvres converties tout au long du projet.

- Les données d'entrées ont été converties en format .dat à .json. En effet, nous utilisons VueJS et le JSON est reconnu nativement par JavaScript, il est donc simple d'extraire les informations souhaitées. De plus, le format .json est simple à mettre en œuvre. Ces données d'entrée nous serviront pour la réalisation de graphiques 2D et 3D ainsi que des statistiques afin de représenter les données.
- Les données de sorties ont été converties en format mat à .tif pour pouvoir utiliser la bibliothèque JavaScript Geotiff. Ces données serviront à importer un objet 3D représentant les hauteurs d'inondation sur la zone d'études.

3.2 Maquettes et data visualisation

Nous avons réalisé différentes maquettes pour avoir un aperçu du site, des fonctionnalités ainsi que des approches pour visualiser les données d'entrée et de sorties des scénarios.

3.2.1 Maquettes du site

Tout d'abord, nous avons réalisé une maquette du site. Nous avons prévu d'avoir un header permettant à l'utilisateur de choisir le site d'étude (Gâvres ou Arcachon) et les informations sur le site. Une validation dans ce header permettra l'apparition d'un panel contenant les différents graphiques sélectionnés.

Il y aura aussi un footer qui donnera les informations sur la carte.

Viens ensuite la partie centrale composée de la carte. La carte contient un bouton permettant de passer de la vue 2D à 3D.

Le dernier composant de l'interface est une boîte à outils. Il permettra le changement de fond de carte, l'affichage ou non de bâtiments sur la carte, un re-centrage de la vue et enfin l'affichage des données spatialisées directement sur la carte.

Voici un modèle de conception UI avec les widgets d'interaction fermés :

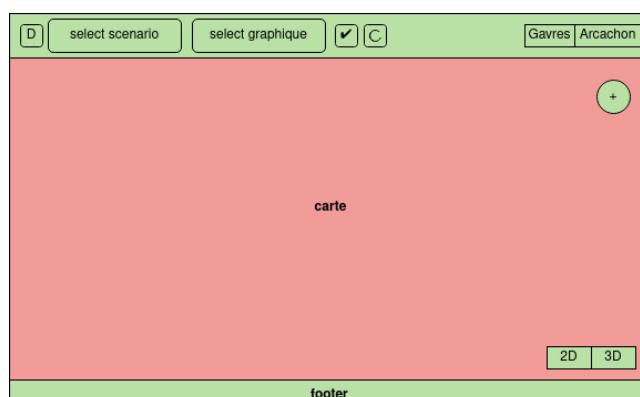


FIGURE 3.1 – Maquette du site avec widgets pliés

Voici un modèle de conception UI avec les widgets d'interaction ouvert :

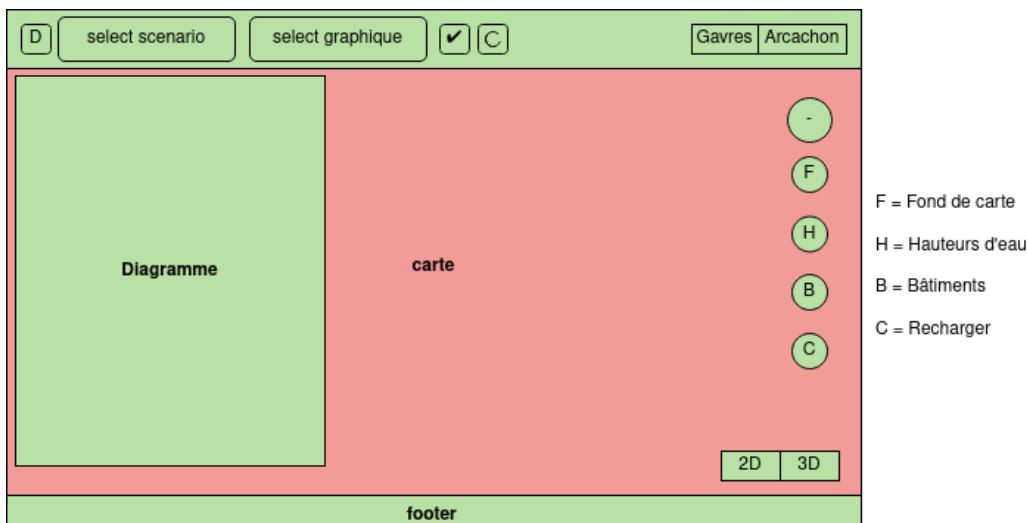


FIGURE 3.2 – Maquette du site avec widgets dépliés

3.2.2 Styles de visualisations de données

Avant de réaliser les maquettes des datavisualisations, nous avons regardé les visualisations possibles avec les différentes librairies graphiques compatibles avec vueJS3 telles que Vue-ChartJS, Vue-HighCharts, D3, ApexCharts ou encore JSCharting. Nous avons également regardé certains sites proposant des datavisualisations sur le vent ou sur les vagues comme des sites pour le surf.

Avec cette étude préalable, nous avons retenu certains graphiques que nous allons vous présenter ci-dessous.

Le premier diagramme (figure 3.3) présente la hauteur des vagues en fonction du temps. Ce type de diagramme sera également réalisé pour :

- la surcote en fonction du temps ;
- la vitesse du vent en fonction du temps.

Il est possible d'afficher une ou plusieurs simulations.

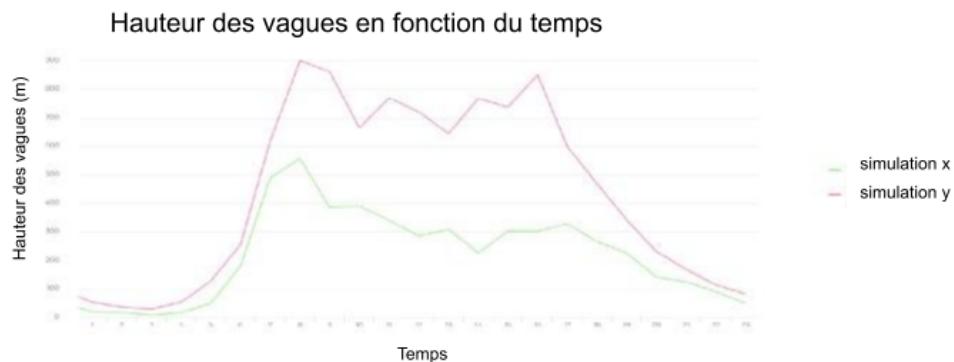


FIGURE 3.3 – Graphique hauteur des vagues en fonction du temps

Le diagramme suivant (figure 3.4) présente la distribution de la vitesse du vent pour une simulation comme nous avons en données d'entrée la direction du vent ainsi que la vitesse du vent. Les données présentées dans le diagramme ci-contre ne sont pas issues des données fournies.

Nous avons vu que les données pour une simulation sont très proches en terme :

- de vitesse de vent (écart d'environ 3 m/s entre la valeur minimale et maximale) ;
- de direction du vent (écart d'environ 10° entre la valeur minimale et maximale).

Distribution de la vitesse du vent à Gâvres pour la simulation X

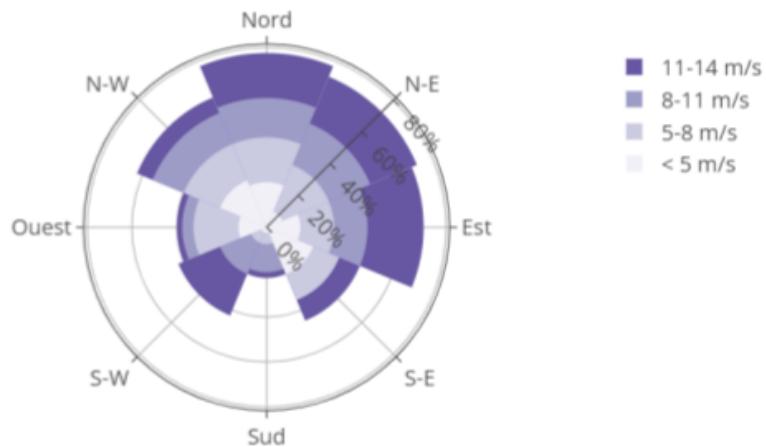


FIGURE 3.4 – Graphique distribution de la vitesse du vent

Le diagramme suivant (image 3.5) présente une carte montrant les zones de Gâvres selon la hauteur des vagues maximales. Pour ce faire, il faut ré-échantillonner les données dont on dispose selon un carroyage. Si cette carte est retenue, nous pourrons étudier plus en détail la structure à appliquer. Il est possible de prendre en compte une ou plusieurs simulations. Dans le cas de plusieurs simulations, nous pensons simplement faire une moyenne des valeurs des données de sorties car ces données sont bien spatialisées.

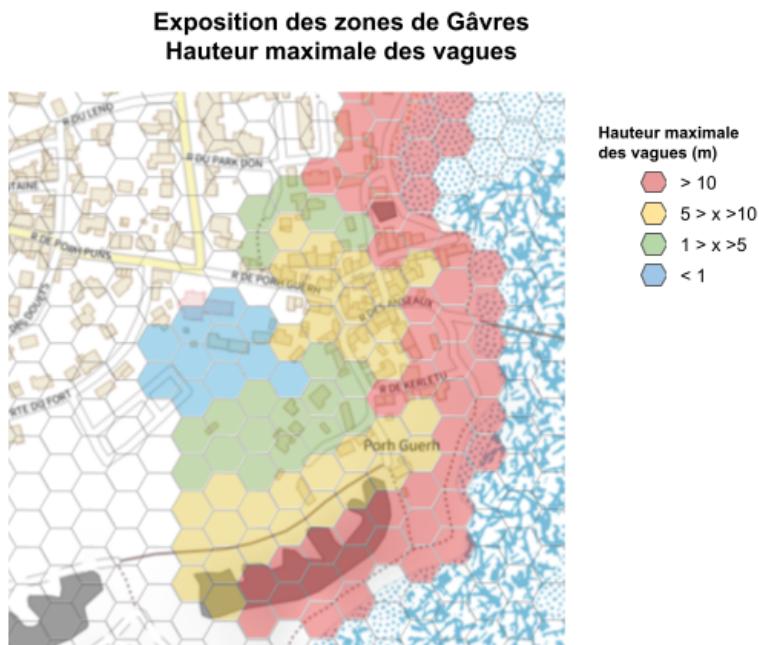


FIGURE 3.5 – Graphique distribution des hauteurs maximales des vagues

L'histogramme empilé suivant (image 3.6) présente la répartition en pourcentage de la vitesse du vent. Dans l'exemple, il y a quatre intervalles, ils sont également choisi de manière arbitraire et pourront évoluer au fur et à mesure de l'implémentation du graphique. La répartition des valeurs de la légende pourra donc être revue, nous voulions juste présenter le principe du graphique. Il est possible de prendre en compte une ou plusieurs simulations.

Ce type de diagramme sera également réalisé pour :

- la direction du vent ;
- la direction des vagues ;
- la hauteur des vagues.

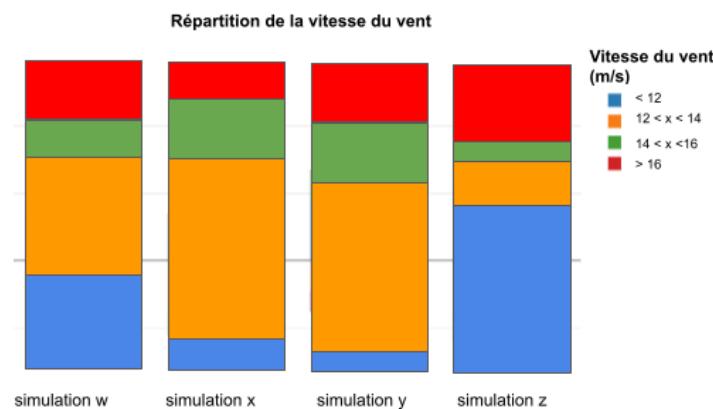


FIGURE 3.6 – Graphique en histogramme empilé

Le diagramme suivant (image 3.7) présente l'évolution de la hauteur des vagues en fonction de l'évolution temporelle. La variable "date" ne sera pas pris en compte dans le cas de notre étude. Il est possible de prendre en compte une ou plusieurs simulations.

Ce type de diagramme sera également réalisé pour :

- la direction du vent ;
- la direction des vagues ;
- la vitesse du vent.

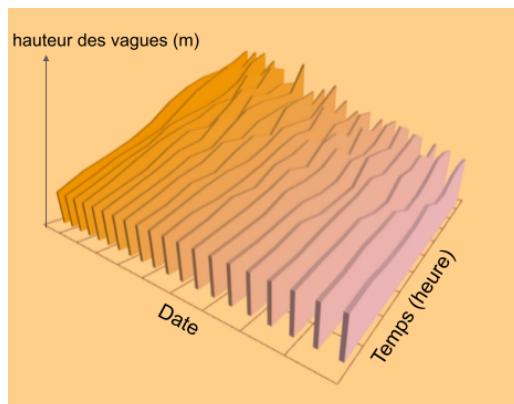


FIGURE 3.7 – Graphique hauteur des vagues 3D

La visualisation suivante (figure 3.8) présente l'évolution de la hauteur (maximal ou finale) des vagues selon le critère spatial. Elle utilisera les données de sortie.

Il est possible de prendre en compte une ou plusieurs simulations. Dans le cas de plusieurs, nous pensons proposer à l'utilisateur de visualiser la hauteur maximale de l'ensemble des scénarios, la hauteur la plus basse de l'ensemble des scénarios ou alors la hauteur moyenne de l'ensemble des scénarios sélectionnés.

Cette représentation sera visible directement sur la carte tel un MNT.

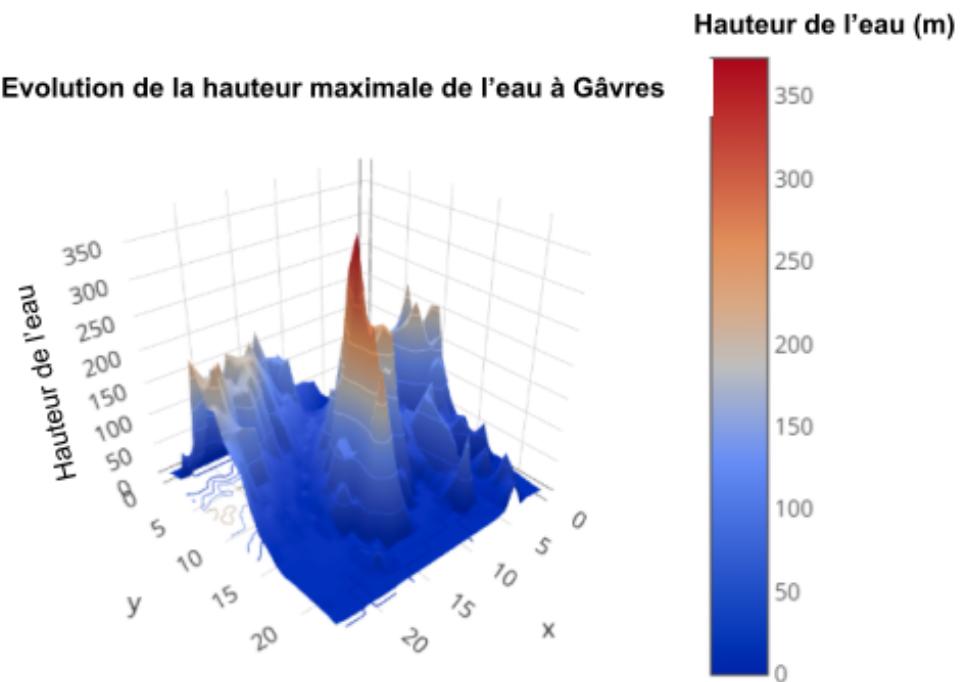


FIGURE 3.8 – Graphique évolution hauteur d'eau maximale 3D

3.3 Diagramme de classes

À travers le diagramme de classes ci-dessous, vous pouvez voir ce que sont exactement les visualisations de notre site. Nous avons des données et des groupes de données à montrer, à faire voir via des outils.

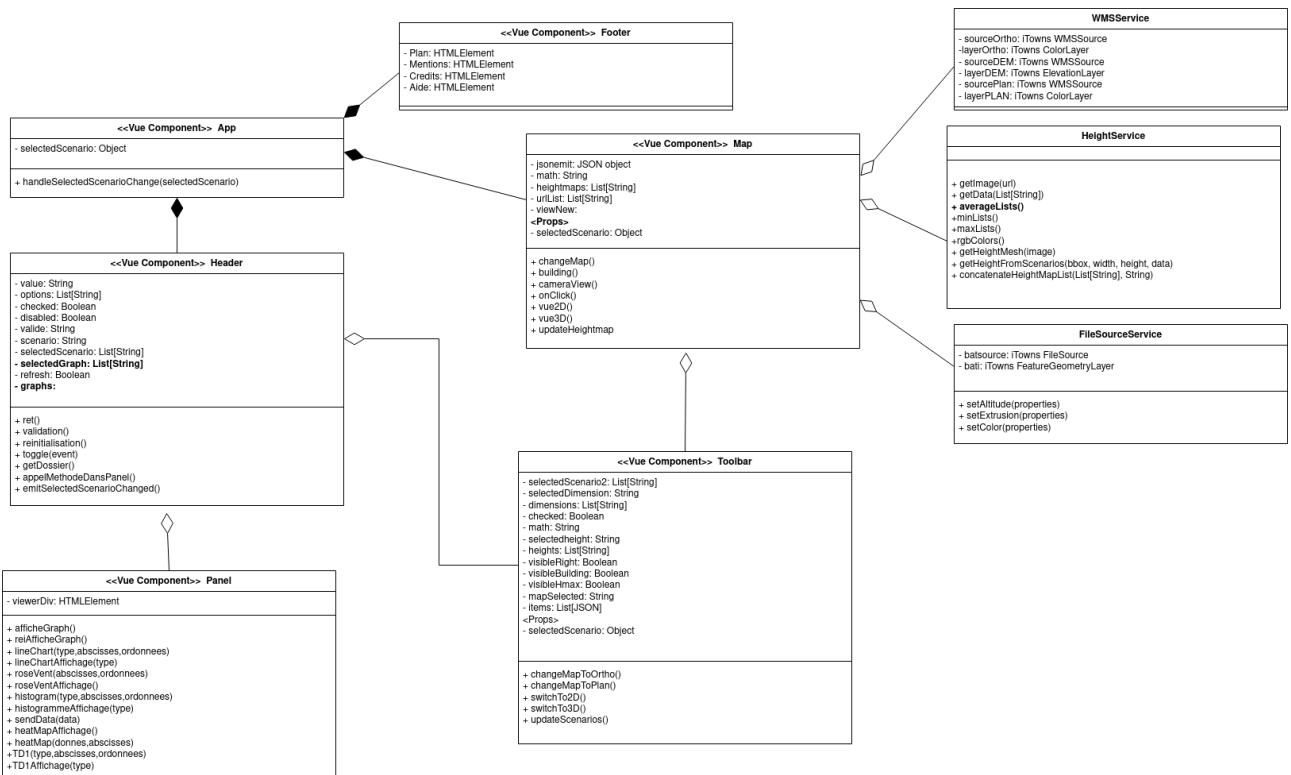


FIGURE 3.9 – Diagramme de classes

3.4 Diagramme d'activités

Pour accomplir certaines tâches, l'ordre des événements peut être important. Pour clarifier les tâches à accomplir et l'ordre des événements pour chacune d'entre elles, nous présenterons ci-dessous deux diagrammes d'activités cruciaux au fonctionnement de notre logiciel. Nous présenterons ci-dessous la séquence d'actions pour l'affichage d'un diagramme représentant les données d'entrée (diagramme 3.10) et le chargement d'un objet 3D représentant les hauteurs d'eau (diagramme 3.11).

Pour le diagramme 3.10, ApexCharts et HighCharts sont les deux librairies utilisées pour réaliser les différents types de graphiques. ApexCharts propose une option pour les télécharger directement dans certains formats (png ou jpg notamment), ce que ne permet pas HighCharts.

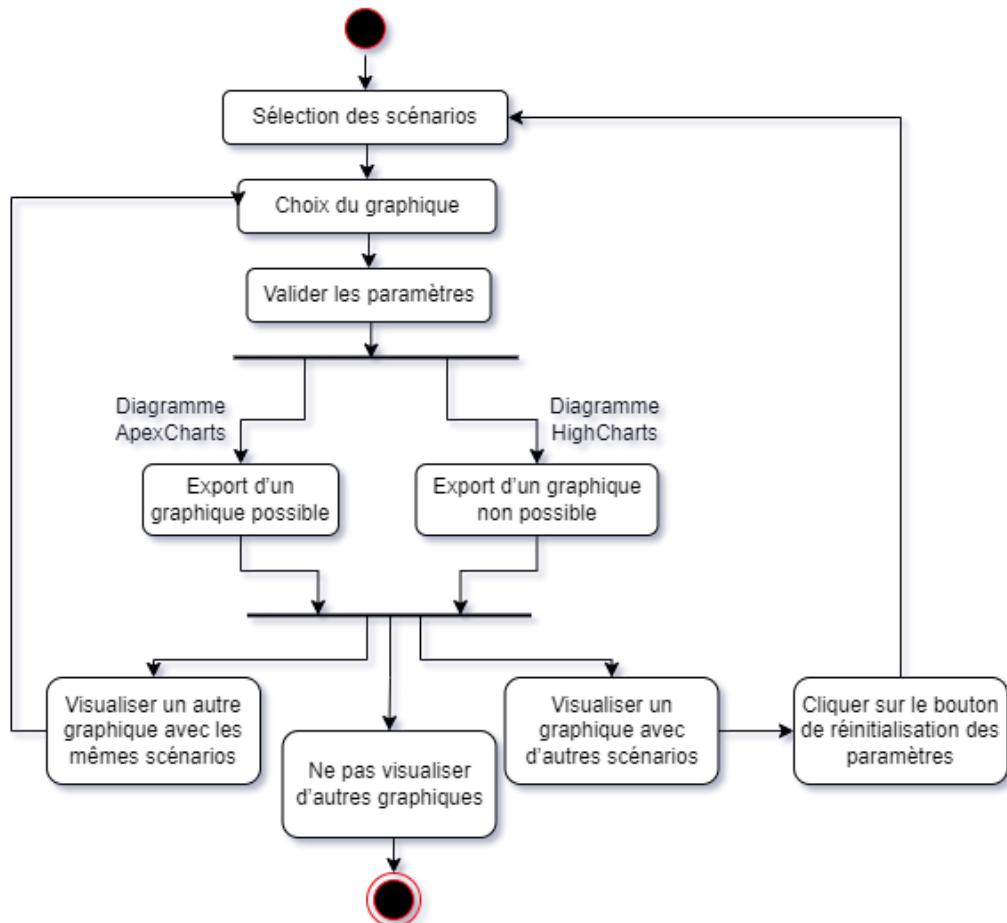


FIGURE 3.10 – Diagramme d'activité : visualisation des graphiques

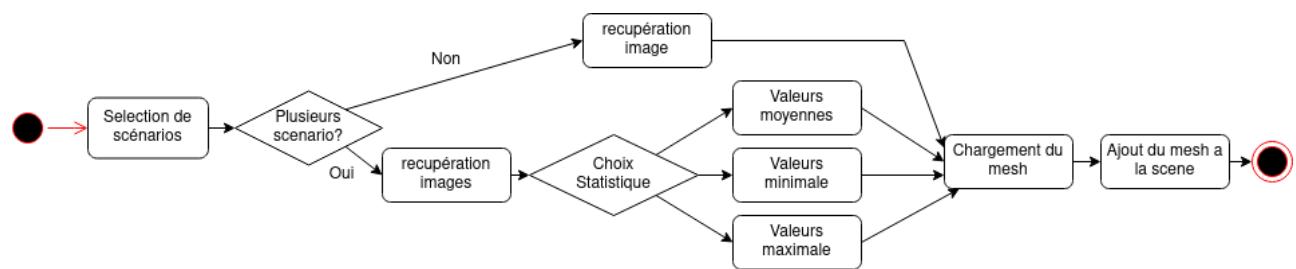


FIGURE 3.11 – Diagramme d'activité : chargement des hauteurs d'eau

ANALYSE TECHNIQUE

Dans ce chapitre consacré à l'analyse technique, nous aborderons tous les choix techniques que nous avons fait et les raisons qui les ont motivés. Nous commencerons rapidement par vous présenter notre environnement de développement. Ensuite, nous présenterons la fondation de notre projet, VTThree, un projet de développement LASTIG que nous devions utiliser comme base pour notre prototype et auquel nous devions ajouter des fonctionnalités. Dans la partie consacrée à VTThree, nous passerons en revue les principaux changements qui ont dû être apportés à VTThree pour qu'on puisse répondre aux besoins des utilisateurs. Ensuite, nous passerons en détails sur tous les langages et technologies que nous avons choisi d'utiliser et pourquoi.

4.1 Environnement de Développement

4.1.1 VCS

En tant que gestionnaire de contrôle de version, l'équipe utilise Github. Github nous permet également de définir et d'assigner des tâches à réaliser et de voir la progression de nos sprints.

4.1.2 IDE

En tant qu'environnement de développement intégré, l'équipe utilise VSCode. Ce choix s'est fait naturellement, car nous avons l'habitude de l'utiliser. Il est facile d'intégrer Git dans VSCode grâce aux extensions proposées, tout comme JSDoc pour la documentation.

4.1.3 CI/CD

Le CI/CD est un processus d'automatisation pour la construction, les tests et le déploiement de logiciels. Dans le sein de ce projet, il a été implémenté avec l'utilisation de Github Actions, une fonctionnalité de Github qui permet de vérifier automatiquement la construction du code à chaque modification. Cela assure une qualité et une cohérence du code, ainsi qu'une livraison continue du logiciel.

4.2 VTThree

VTThree, alias Vector Tiles Three, est un prototype logiciel conçu pour la visualisation données cartographiques Vecteur Tuilés 3D et le chargement d'Objet 3D ThreeJS (<https://github.com/umrlastig/vtthree>). VTThree contient des fonctionnalités de bases que nous étions encouragés de reprendre dans la réalisation de notre prototype appliquée à la visualisation de données de submersion côtière confidentielles. Nos instructions étaient de continuer le développement à partir du code existant de VTThree.

Cependant, nous devions répondre à ces demandes clientes supplémentaires :

- Permettre le chargement de différents formats que vecteur tuilé.
- Ajouter un passage clair de la vue 2D à la vue 3D ; c'est-à-dire, d'afficher un scène en 2D/3D géospatiale avec une transition entre ces deux dimensions.
- Ajouter des composants au logiciel comme une fenêtre statistique et une barre d'outils
- Protéger l'accès aux données qui doivent rester confidentielles.

- Assurer la scalabilité du code afin de faciliter l'ajout de nouvelles fonctionnalités, l'import de différent format de données ou la visualisation d'autres cas d'Études que Gâvres ou Arcachon dans le futur.

Suite à la prise en main de VTThree, il s'est avéré que reprendre ce travail existant serait limitant pour nos objectifs fonctionnels.

- Dans VTThree tout le concept de scène 3D est rattaché au format vecteur tuilé. Donc intégrer différents types de données dans VTThree aurait demandé une compréhension détaillé et une déconstruction considérable du code existant.
- VTThree a été développé en JavaScript sans utilisation de framework front. L'utilisation d'un framework front facilite l'utilisation de routes et de composants HTML. Les routes permettraient facilement le passage entre plusieurs cas d'études comme Gâvres ou Arcachon. Les composants faciliteraient la réutilisation de la fenêtre statistique ou la boîte à outils. Un framework front a aussi le potentiel d'améliorer le dynamisme et la performance global du site.
- De plus, VTThree a été conçu dans une architecture 1-tiers. C'est-à-dire que le stockage des données et la visualisation se faisait dans la même brique logicielle. Cette solution n'est pas adaptée à la protection de données confidentielles.

Compte tenu de tous ces facteurs, voici les solutions que nous avons convenues avec notre commanditaire :

- Intégrer iTowns qui une bibliothèque logiciel 2D/3D permettant le chargement de différents types de données géographiques ainsi que le chargement d'objet 3D ThreeJS.
- Migrer sur VueJS afin de permettre l'utilisation de routes, de composants, d'améliorer la performance et facilement de mettre en place un environnement de tests unitaire.
- Passer sur une architecture Multi-tiers pour séparer le logiciel, les traitements et les données.

4.3 Architecture

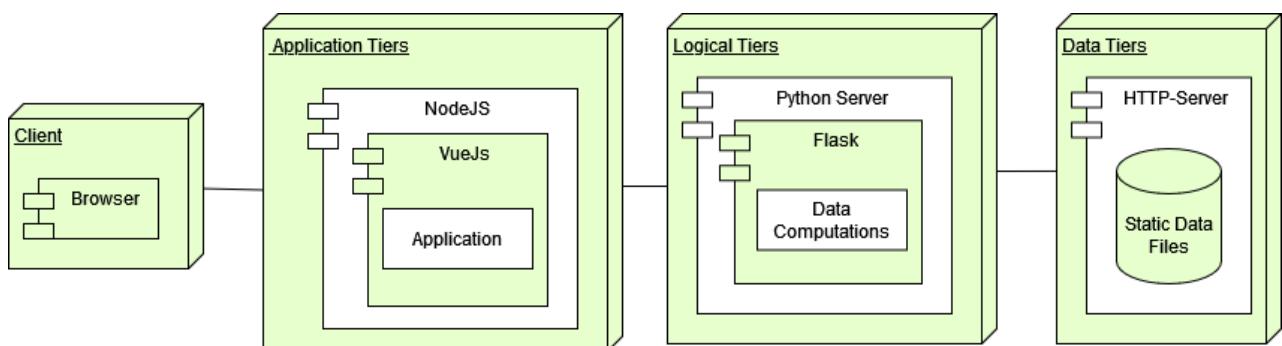


FIGURE 4.1 – Diagramme d'architecture

Cette partie est consacrée à l'architecture de notre prototype. Notre prototype est divisée en plusieurs Tiers. Le tiers d'application contient notre front-end du projet. L'interface utilisateur auquel l'utilisateur pourra faire appel de son navigateur. Beaucoup de traitement de données sont fait directement de ce tiers, notamment tout ce qui est visualisation de carte et d'objet 3D. Toutes les données sont des fichiers statiques rendus accessibles grâce à notre serveur HTTP. Pour certains graphiques, il est nécessaire d'appliquer un prétraitement python aux données avant de les envoyer à notre tiers d'application. Lorsque nos données ont besoin d'un prétraitement, il passe par notre serveur python. Plus tard dans ce chapitre, nous allons vous expliquer comment nous avons conteneurisé ces briques applicatives en utilisant docker afin de pouvoir lancer tous les composants grâce à une seule commande. Il aurait été possible et facile de déployer notre

application en ligne. Cependant, dû à la confidentialité, ça a été la demande de commanditaire que nous réservons l'accès au logiciel uniquement au local.

4.4 Langages et Technologies

4.4.1 VueJS

VueJS présente plusieurs avantages pour notre projet : tout d'abord, c'est un framework front end très léger. VueJS est idéal pour les petits projets de faible complexité. VueJS permet facilement la création de SPA, Single Page Application, mais aussi facilite le routage en différentes URL. Dans le cadre de notre projet, la réutilisation de composants était une priorité, car nous voulions fournir des fonctionnalités comme la fenêtre statistique ou la barre à outil qui pourront ensuite être réutilisées sur plusieurs sites de recherche comme Gâvres et Arcachon. Nous avons choisi d'utiliser Vue 3, car la syntaxe est facile à comprendre et propose des meilleures performances comparé à Vue 2. Vue est un framework très populaire qui possède une grande panoplie de librairies compatibles. Notamment, nous avons créé tous les widgets permettant l'interaction entre l'utilisateur et l'application à partir de la librairie PrimeVue.

Une priorité dans notre projet était de fournir une agréable manière de visualiser les données, par exemple la vitesse et direction du vent, sous forme de graphiques. Pour les graphiques, nous avons tout d'abord réalisé des maquettes et nous avons ensuite regardé les différentes librairies possibles. À notre mauvaise surprise, certaines n'étaient pas compatibles avec VueJS et nous n'avons pas trouvé une seule librairie avec tous les graphiques (2D et 3D) que nous souhaitions. Nous avons alors utilisé deux librairies : ApexCharts pour les graphiques de base et HighCharts pour les graphiques un peu plus complexes comme la rose des vents ou les graphiques en 3D.

4.4.2 ThreeJS

Toutes les fonctionnalités cartographiques et 3D de notre projet ont été rendus plus facile à développer grâce à la bibliothèque logicielle Opensource ThreeJS. ThreeJS est une bibliothèque JavaScript permettant de créer et d'afficher des scènes 3D sur le web. ThreeJS possède beaucoup de fonctionnalités intégrées pour gérer et faciliter l'utilisation des contrôles clavier et souris, des caméras et des objets 3D ainsi que leurs géométries dans une scène 3D. Cependant, ThreeJS n'est pas une bibliothèque cartographique. ThreeJS n'a pas de notion intégrée de projection cartographique, d'échelle cartographique ou de gestion de couche. C'est pour ça que l'IGN a créé iTowns.

4.4.3 iTowns

iTowns est une bibliothèque de cartographie 2D/3D JavaScript Opensource basée sur ThreeJS. À travers iTowns, nous avons accès à toutes fonctionnalités de ThreeJS en plus de fonctionnalités cartographiques. iTowns permet de créer une fenêtre cartographique qui est destinée à héberger des données spatialisées. iTowns facilite l'import de données géospatiales issues de fichiers ou de flux issus de services web. Dans le cadre de notre projet, nous avons uniquement utilisé des flux WMS provenant des services web de l'IGN, mais il serait également très facile d'importer différents types de flux comme les flux WMTS, WFS de n'importe quelle autre source. iTowns est aussi très puissant pour visualiser les modèles numériques d'élévation et dispose de fonctionnalités avancées telles que la visualisation de couches vectorielles et de données multi-résolution.

iTowns nous a donc aussi permise d'utiliser les fonctionnalités de bases de ThreeJS. Dans notre projet, nous chargeons les hauteurs d'eau en tant qu'objet 3D dans la scène. Grâce à ThreeJS, nous avons pu assigner un gradient de couleur personnalisé sur l'objet 3D en fonction de la hauteur d'eau de la donnée qu'on affiche parmi d'autres avantages.

4.4.4 Stockage des données

Due à la confidentialité et aux formats nos standardisées des données de scénarios d'inondation côtières, il a été jugé non pertinent par le commanditaire et notre équipe de les mettre dans une base de données. Nous sommes aussi arrivés à la conclusion qu'il était plus sage de stocker les fichiers de données statiques en dehors de projet en lui-même pour permettre le partage de l'application sans forcément partager les données. De plus, ce sont des données très lourdes. Du coup, un serveur HTTP appart a été mise en place ce qui permet d'appeler les données dans notre application et de gérer les CORS.

Si vous avez accès aux données, vous aurez accès à un dossier 'data' qui contient les données retravaillées sous format JSON et TIF. Les instructions sont disponibles dans le README du projet.

4.4.5 Python et Flask

Pour réaliser des statistiques, des traitements sur les données et pour gérer les requêtes HTTP, nous avons choisi d'utiliser Flask. Flask est un micro-framework développé en Python qui est très léger, car il ne propose pas énormément de fonctionnalités. Cependant, pour notre utilisation, Flask convient.

Initialement, nous voulions faire un ré-échantillonnage des données selon un carroyage Gâvres pour visualiser les zones les plus exposées au phénomène de submersion. Cette visualisation n'a pas été réalisée suite au temps consacré pour les autres graphiques avec les librairies de VueJS.

Flask n'est donc pas réellement exploité, mais nous avons choisi de la garder, car l'application et Flask communiquent bien ensemble. Il est donc possible, pour des futurs travaux, de s'appuyer sur Flask et ArcPy pour réaliser d'autres traitements en couplant les données spatiales de Gâvres à d'autres données comme la population de Gâvres ou les types de bâtiments.

4.4.6 Docker

La mise en place de Docker s'est rapidement imposée pour le déploiement de notre application. En effet, sans Docker, nous devons démarrer trois serveurs (Node pour l'application, un serveur HTTP pour l'accès aux données et Flask pour le traitement des données). C'est un processus qui peut être assez pénible, d'autant plus qu'il faut faire attention aux ports. Pour ce faire, chaque tiers du logiciel a son propre Dockerfile qui génère une image du service en question. Ensuite, le Docker-compose permet d'orchestrer le service multi-conteneurs, de lancer et d'arrêter l'ensemble de l'application d'une manière cohérente et reproductible, en gérant les dépendances entre les différents services et en configurant les variables d'environnement pour chaque conteneur. Toutes les informations d'utilisation peuvent être retrouvées dans le README.

PRÉSENTATION DE L'APPLICATION

5.1 Fonctionnalités et solutions réalisées

5.1.1 Sélection des scénarios

Cette fonctionnalité nous permet de choisir des scénarios à partir d'une liste d'options disponibles dans un dossier local, accessible via la création d'un serveur interne. En choisissant les scénarios, nous pouvons ensuite accéder aux différentes fonctionnalités présentées ci-dessous dans cette section, qui utilise les données contenues dans ces scénarios soit pour le traitement, soit pour l'affichage.

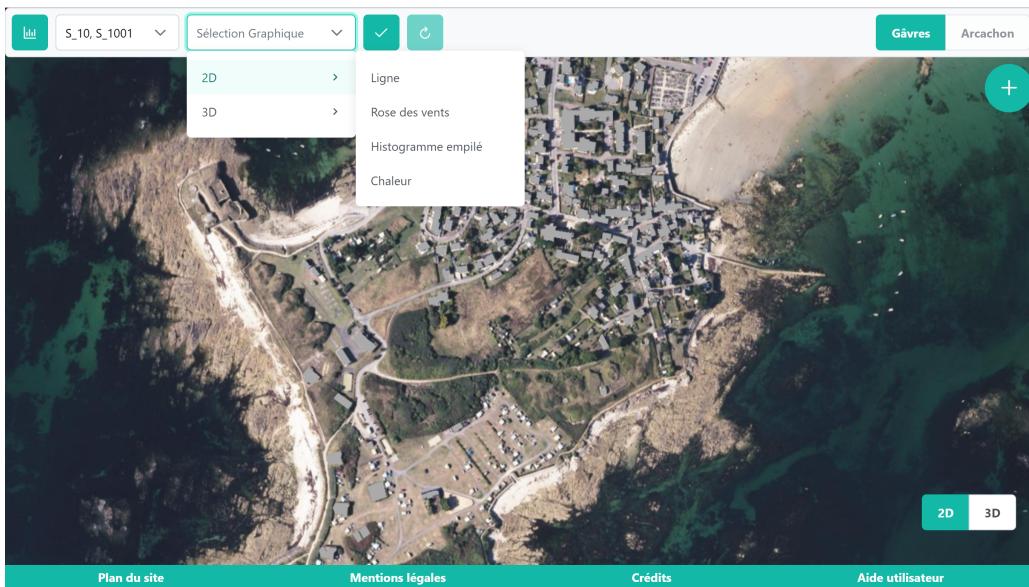


FIGURE 5.1 – Sélection de scénarios et d'un type de graphique

5.1.2 Visualisation 2D/3D

Cette fonction nous permet de choisir la dimension dans laquelle nous voulons voir notre plan. Au démarrage de l'application, nous sommes par défaut sur une vue 2D ; les commandes de navigation 3D sont verrouillées. L'utilisateur ne peut se déplacer qu'uniquement sur le plan horizontal. En sélectionnant la vue 3D, nous disposons des commandes de navigation 3D et de la possibilité de faire pivoter la vue en utilisant la touche 'Ctrl' du clavier ainsi que la souris pour modifier l'angle de vision.

Sur la figure ci-dessous (figure 5.2), nous sommes en 3D et l'affichage des bâtiments est activée. Si l'utilisateur repasse en 2D, les bâtiments seront toujours affichés.

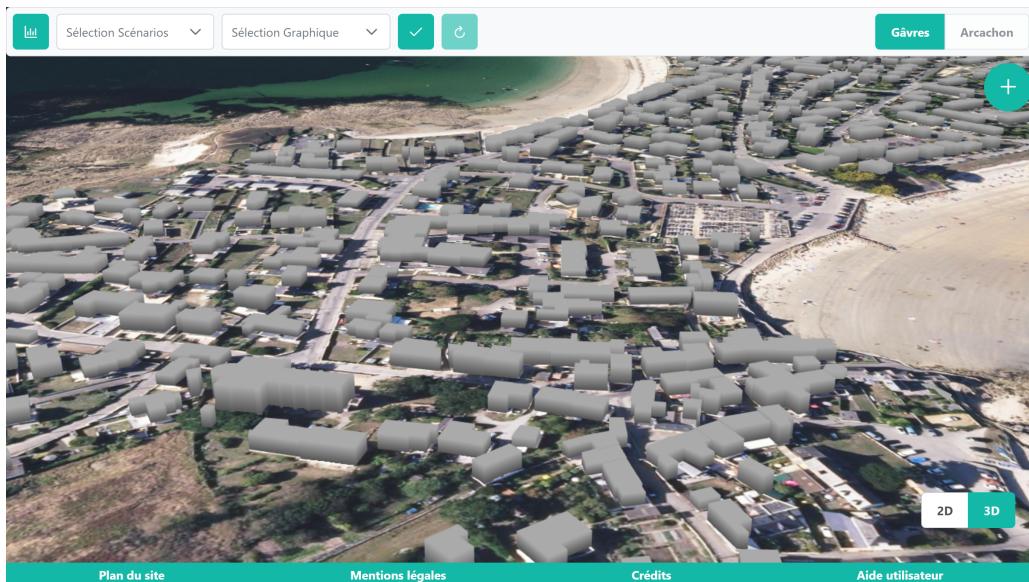


FIGURE 5.2 – Passage de 2D à 3D

5.1.3 Changement fond de plan

La fonction de modification de fond de plan nous permet de modifier la carte de base de notre vue. Le fond de carte par défaut est la vue satellite générée à partir d'un protocole WMS (Web Map Service) d'un flux d'images de l'IGN, la deuxième option est le Plan IGN, un service spécialisé pour l'affichage sur écran et contenant des images à différents niveaux de zoom. Les deux plans nécessitent une connexion internet pour être chargés.

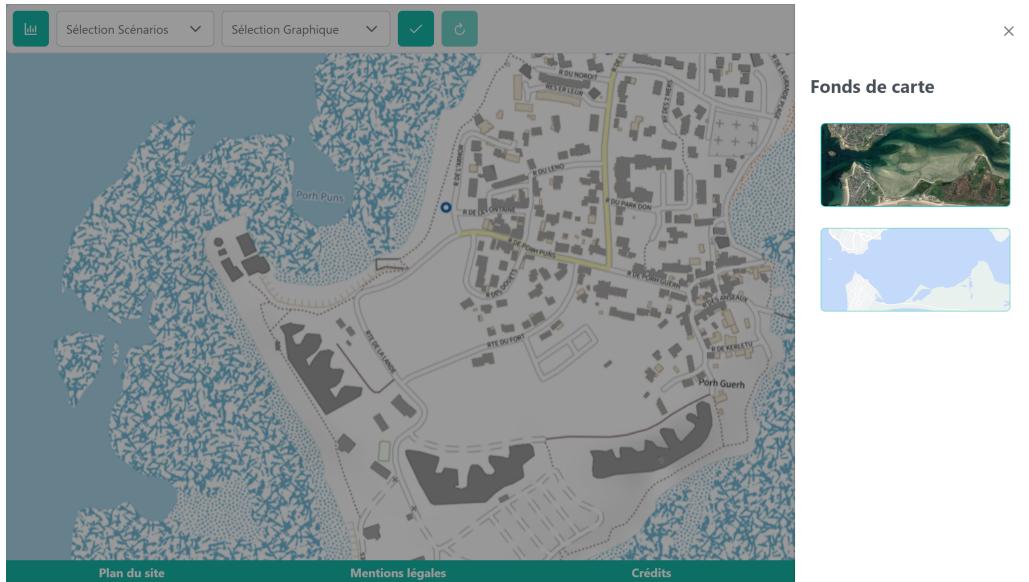


FIGURE 5.3 – Changer le fond de carte

5.1.4 Affichage hauteur d'eau sur la carte

En ce qui concerne la représentation des hauteurs d'eau, les maillages sont générés en tant qu'objets 3D qui sont superposés aux couches existantes (orthophotos et élévations). Il y a plusieurs façons d'afficher les hauteurs d'eau, la première est de choisir un scénario et de définir si nous voulons afficher sa hauteur maximale (hmax) ou la hauteur finale (hfin) du scénario. La deuxième option nous permet de choisir plus d'un scénario, ce qui nous donne accès à des options supplémentaires, en plus de pouvoir choisir les

hauteurs maximales ou finales des scénarios, nous avons la possibilité d'afficher la moyenne, le minimum ou le maximum des hauteurs de l'ensemble des images que nous avons sélectionnées. Les mailles sont générées avec une gamme de couleurs bleues qui dépendent de la hauteur de l'eau par rapport à la hauteur de la surface.

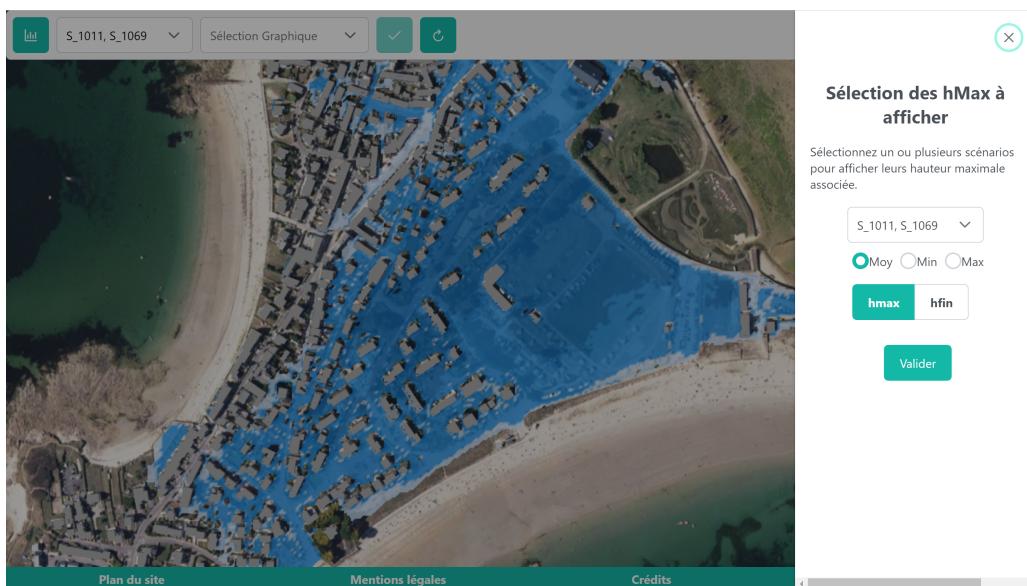


FIGURE 5.4 – Sélection des scénarios à afficher

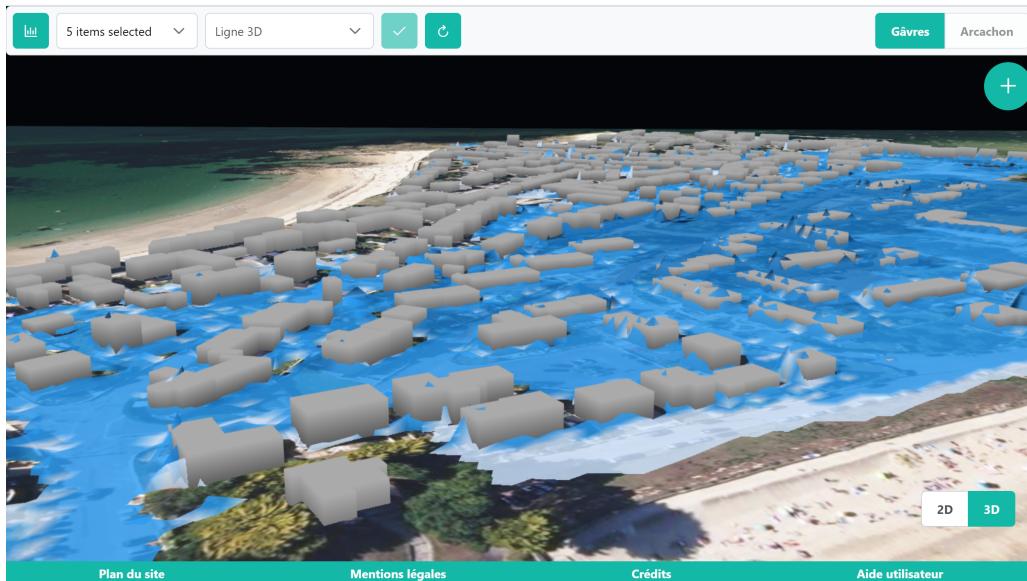


FIGURE 5.5 – Couche affichée en 3D

5.1.5 Affichage et masque des bâtiments

Cette fonction permet d'afficher ou de masquer une couche de bâtiments représentée en 3D dans notre vue. L'importation de ces bâtiments se fait à partir d'un fichier GeoJson qui est ensuite dessiné et extrudé comme un objet de géométrie 3D de iTowns.

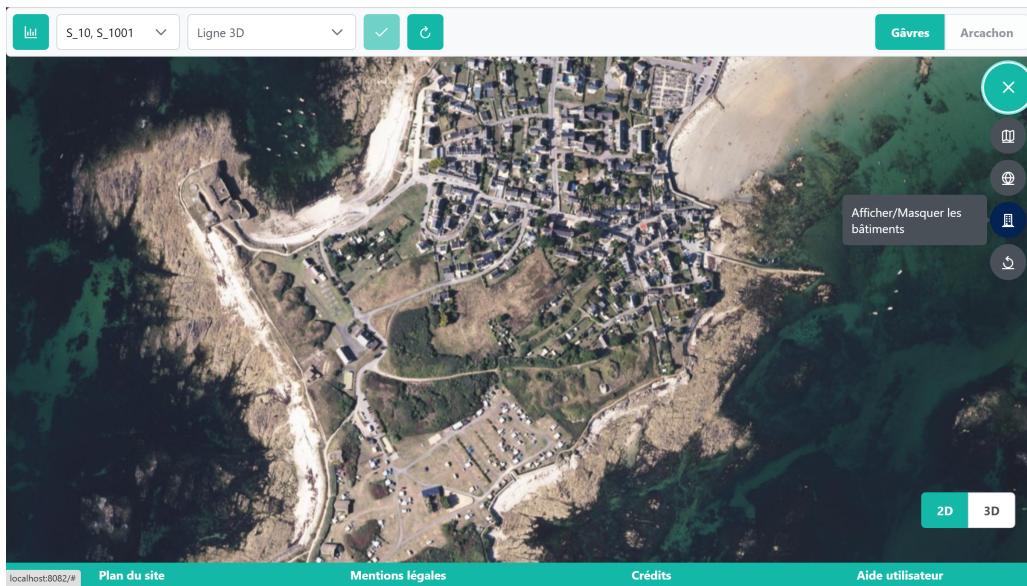


FIGURE 5.6 – Fonction d'affichage des bâtiments

5.1.6 Visualisation des données

Différentes manières de visualiser les informations contenues dans les fichiers .dat fournis par le commanditaire ont été développées. L'interface permet de visualiser les données via des graphiques de type rose des vents, linéaire, linéaire 3D, carte de chaleur et histogramme empilé. Les variables représentées sont : hauteur de la marée, hauteur des vagues, direction du vent, vitesse du vent et surcote.

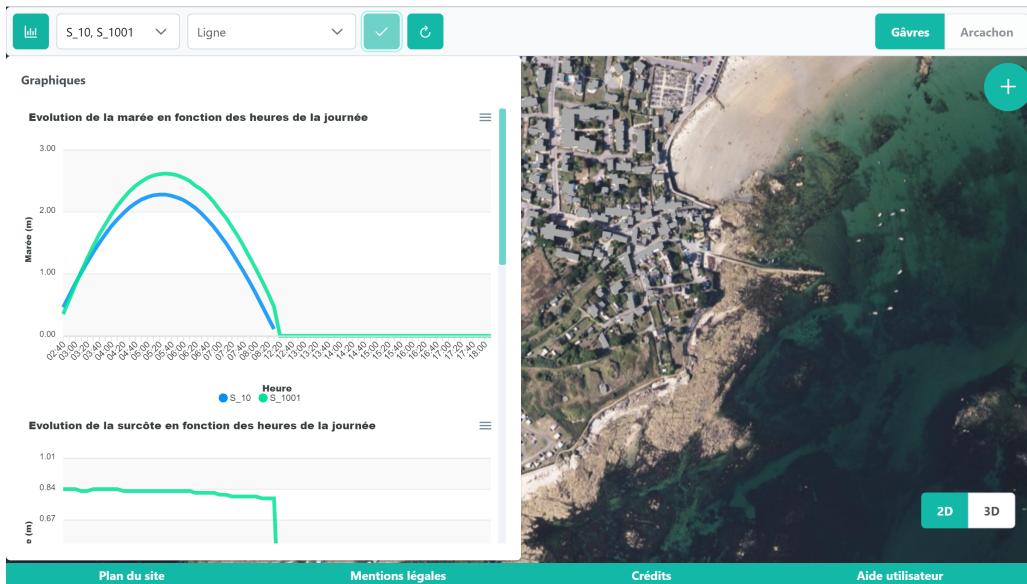


FIGURE 5.7 – Affichage de graphiques

5.2 Solutions envisagées

Voici les ajouts que nous avons identifiés mais que nous n'avons pas implementés, faute de données ou de temps.

La résolution des géotiffs créés lors du pré-traitement dépend directement de la résolution du MNS utilisés pour la modélisation des scénarios. Si une visualisation plus détaillée des hauteurs d'eau est souhaitée,

un MNS de plus haute résolution peut être utilisé.

5.3 Limites

Au cours de la phase d'analyse trois types de limites ont été identifiées :

- Une première limitation due aux formats des données ;
- Une seconde due au code du travail déjà effectué sur ce projet ;
- Une troisième trouvée au moment de la mise en place de l'environnement de test.

Les formats dans lesquels les fichiers étaient fournis présentaient certaines limites, l'une étant qu'ils ne pouvaient être ouverts que par un logiciel spécifique pour lequel une licence était nécessaire, et l'autre étant la manière dont les informations pouvaient être extraites. Les outils utilisés ne permettaient pas d'introduire des données dans ces formats car ils n'étaient pas très réactifs et un prétraitement était nécessaire, changeant le format des fichiers en un format intermédiaire, dupliquant les informations existantes.

Nous aurions également pu repartir du code qui nous a été fourni sur le dépôt Github de VTThree. Cependant nous avons choisi de ne pas l'utiliser et de partir de zéro et d'utiliser VueJS. Cela nous permet au final de gagner du temps car nous n'avons pas besoin de consacrer beaucoup de temps à la compréhension du code existant. Ce code existant est non documenté, pas à jour sur certaines fonctionnalités et ne nous a pas semblé modulaire pour notre projet.

En ce qui concerne les tests, il convient de noter que différentes méthodes de test ont été mises en place en utilisant divers frameworks. Nous avons tenté d'utiliser Mocha puis Jest avec les mêmes résultats. Toutefois, ces tests ont été confrontés à des erreurs lors de l'import d'iTowns. Les tests passaient en revue tout le contenu du module iTowns importé et détectaient des erreurs sur des fonctions iTowns que l'on n'utilisait nulle part dans notre code. Les tests sont mis en place pour éviter des régressions de code et de s'assurer reste fonctionnel tout au long du développement pour gagner du temps. Cependant, nous avons consacré énormément de temps à la mise en place de tests unitaire en début de projet. De plus, les erreurs que les tests détectaient n'étaient pas issus de nos développements. Par conséquent, nous avons mis les tests de côté. Pour futur développement sur ce projet, il serait judicieux de trouver une solution pour créer un environnement de tests unitaires adaptés aux spécificités de la bibliothèque iTowns.

Conclusion

Ce projet de sept semaines nous a permis de réaliser un prototype de visualisation de scénarios de submersion marine et s'appuie notamment sur des données de formats différents de deux zones : Gâvres et Arcachon. Les objectifs sont, d'une part, de proposer des visualisations 2D et 3D des données d'entrées et de sorties de scénarios de submersions simulés. D'autre part, ce projet permet de proposer une un prototype qui pourra être réutilisé dans le cadre du projet ANR Oracles.

La réalisation de ces objectifs s'est tout d'abord déroulée avec une phase d'analyse du code existant et des données disponibles et ensuite la prise en main de différentes technologies et langages utilisés au cours du projet. Des maquettes sont présentées comme support pour la réalisation technique des outils d'analyse visuelle. Une seconde phase permet de mettre en place l'application qui intègre notamment les différentes visualisations ainsi que la gestion de la 2D ou de la 3D. Cette seconde phase concerne également la phase de traitement des données en amont.

Ces travaux permettent d'obtenir une application fonctionnelle pour la zone de Gâvres incluant principalement la visualisation de la scène en 2D ou 3D, la visualisation des données d'entrée d'un ou de plusieurs scénarios à travers des graphiques et la visualisation des données de sorties directement sur la carte ; Arcachon n'étant finalement pas inclus dans l'application par manque de certaines données.

Table des figures

1.1	Diagramme des cas d'utilisation	6
2.1	Schéma de principe Méthode Agile Scrum : Version 3 de l'application	7
2.2	Diagramme de Gantt	8
2.3	Tableau de risques	8
3.1	Maquette du site avec widgets pliés	10
3.2	Maquette du site avec widgets dépliés	11
3.3	Graphique hauteur des vagues en fonction du temps	11
3.4	Graphique distribution de la vitesse du vent	12
3.5	Graphique distribution des hauteurs maximales des vagues	12
3.6	Graphique en histogramme empilé	13
3.7	Graphique hauteur des vagues 3D	13
3.8	Graphique évolution hauteur d'eau maximale 3D	14
3.9	Diagramme de classes	15
3.10	Diagramme d'activité : visualisation des graphiques	16
3.11	Diagramme d'activité : chargement des hauteurs d'eau	16
4.1	Diagramme d'architecture	18
5.1	Sélection de scénarios et d'un type de graphique	21
5.2	Passage de 2D à 3D	22
5.3	Changer le fond de carte	22
5.4	Sélection des scénarios à afficher	23
5.5	Couche affichée en 3D	23
5.6	Fonction d'affichage des bâtiments	24
5.7	Affichage de graphiques	24