Document de spécifications techniques - Application Web CYDRE

Géosciences Rennes/CNRS

30 septembre 2024

Résumé

Le projet CYDRE (Cycle hydrologique, Disponibilité de la Ressource et Évolution) vise à développer une application Web permettant de manipuler un modèle de prévision saisonnière des ressources en eau en Bretagne, afin de répondre aux défis posés par le changement climatique. L'application doit améliorer l'interface graphique et développer de nouvelles fonctionnalités pour répondre aux besoins variés des utilisateurs.

1. Problématique

Les ressources en eau en Bretagne sont soumises à des pressions croissantes dues au changement climatique. Les acteurs de l'eau ont besoin de prévisions saisonnières fiables pour mieux gérer les barrages d'approvisionnement en eau potable et anticiper les situations de sécheresse. Le projet CYDRE doit développer un outil accessible, capable de fournir ces prévisions à divers utilisateurs, y compris des scientifiques et des gestionnaires d'eau.

2. Besoins et Contraintes

2.1 Besoins Fonctionnels

1. Amélioration de l'interface graphique :
   - Redéveloppement de l'interface en Angular.
   - Optimisation de l'ergonomie et du design des résultats.
   - Intégration de graphiques interactifs (plotly.js et leaflet.js).
2. Développement de nouvelles fonctionnalités :
   - Personnalisation de l'affichage en fonction du profil utilisateur.
   - Intégration de la capacité à lire et traiter des fichiers CSV dans le backend.
3. Préparation à la maintenance :
   - Ajout facile de nouveaux modèles.
   - Documentation exhaustive pour la maintenance.

2.2 Contraintes Techniques

- Utilisation de technologies web modernes : Angular pour le front-end, Flask pour le back-end.
- La solution doit être compatible avec les navigateurs web courants.
- La gestion des utilisateurs doit permettre des sessions actives et sécurisées.

3. Environnement Technique

3.1 Matériel

Le serveur est géré par l'Université de Rennes et utilise une machine virtuelle sous Linux avec 8 Go de RAM.

3.2 Logiciel

- IDE : Visual Studio Code pour le développement en Python, TypeScript, HTML, SCSS.
- Gestion des environnements Python : Anaconda 3.
- Gestion du code source : GitLab pour le versionnage et l'intégration continue.

3.3 Langages et Technologies

- **Front-End :** Angular avec TypeScript, HTML, SCSS.
- **Back-End :** Python avec Flask pour l'API REST, Pandas et GeoPandas pour la gestion des données.
- **Base de données :** MySQL pour le stockage des données utilisateurs et l'historique des simulations.

4. Définitions des utilisateurs

Dans cette section, nous allons définir les différents utilisateurs de l'application. Il existe plusieurs types d'utilisateurs, car les besoins varient selon leur rôle. Les acteurs de l'eau, par exemple, nécessitent un affichage simple et une information directe pour faciliter la prise de décision rapide. En revanche, les scientifiques ont besoin d'un accès à une multitude de paramètres sur les algorithmes de calcul pour effectuer des analyses approfondies. Dans l'application, cela se manifeste par une distinction de l'affichage, où chaque utilisateur trouve une interface personnalisée en fonction de ses besoins. Cette personnalisation est gérée par la base de données MySQL, qui stocke le statut de l'utilisateur.

4.1 Visiteur

Le profil "visiteur" est un utilisateur non connecté. Ce profil offre un accès limité à l'application, permettant uniquement d'interagir avec l'interface de base. Cette interface de base est visible sans connexion. Contrairement aux utilisateurs authentifiés, les visiteurs ne peuvent pas accéder aux fonctionnalités avancées.

4.2 Acteur de l'eau

Le profil "acteur de l'eau" en plus des fonctionnalités déjà accessibles au profil "visiteur", dispose d'un compte utilisateur qui lui permet d'enregistrer, de supprimer et de consulter l'historique de ses simulations.

4.3 Scientifique

Le profil "scientifique" bénéficie de fonctionnalités avancées telles que l'accès à des paramètres supplémentaires spécifiques aux algorithmes de prévisions (méthode de calcul, optimisation, etc.).

4.4 Administrateur

Le profil "administrateur" est chargé de gérer les utilisateurs, incluant la création et la suppression des comptes, ainsi que de l'intègre des nouvelles fonctionnalités et la vérification de leur bon fonctionnement.

5. État Initial de l'Application

Initialement, il y avait plusieurs projets déjà en place qui ont servi de base pour l'application finale. Il y avait un prototype Python utilisant la bibliothèque Dash et un autre prototype d'application Angular permettant de modifier les conditions de simulation développé par Brian Gentile. Il existait plusieurs autres travaux issus de projets d'étudiants en 5ème année d'Ingénieur à l'INSA de Rennes, chacun se concentrant sur divers sujets tels que l'ergonomie de l'interface de saisie des conditions, l'affichage des résultats varié (cartes, graphiques), etc.

Comme illustré dans la figure 1, l'interface Dash réunit tous les éléments sur une seule page. Ce prototype intègre déjà plusieurs fonctionnalités important du projet : une zone de sélection dédiée à la station hydrologique souhaitée, des cartes interactives permettant de visualiser les bassins versants correspondant à la station choisie, ainsi que des graphiques fournissant des détails supplémentaires sur la station sélectionnée. De plus, un bouton d'options est également présent, permettant à l'utilisateur de basculer vers l'onglet "simulateur Cydre".

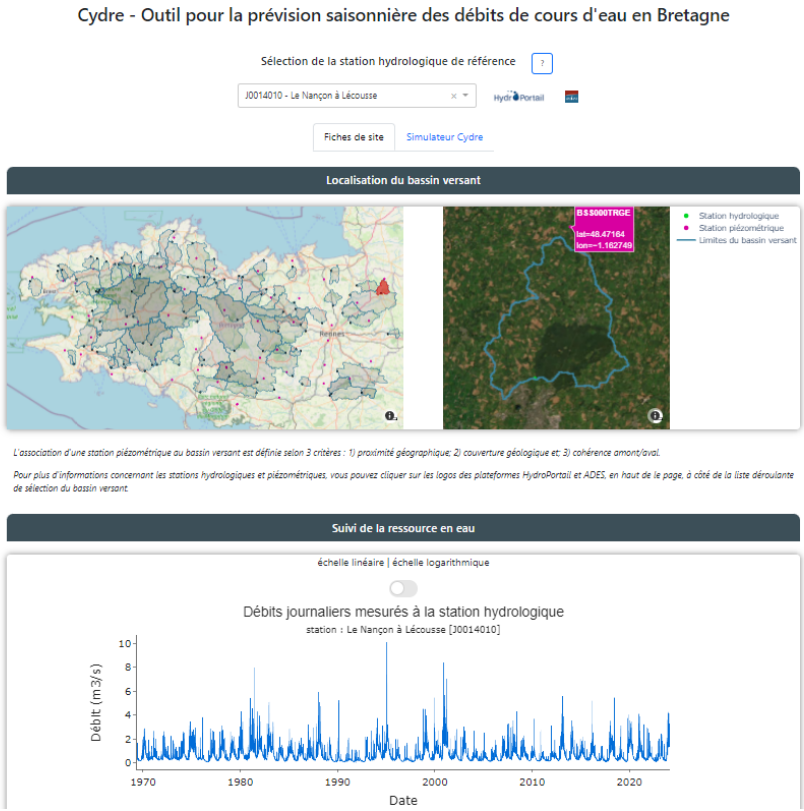

Figure 1: Prototype de l'interface Dash

La version de Brian Gentilé est concentrée sur l'exécution du modèle hydrologique. Le Front-end de cette version est visuellement simple, comme le montre la figure 2, car il est principalement utilisé pour saisir les conditions de la prévision saisonnière à réaliser (le site géographique concerné, les paramètres du modèle hydrologique et la durée de la prévision).Le Back-End quant à lui exécute le modèle hydrologique avec les paramètres du Front-End.

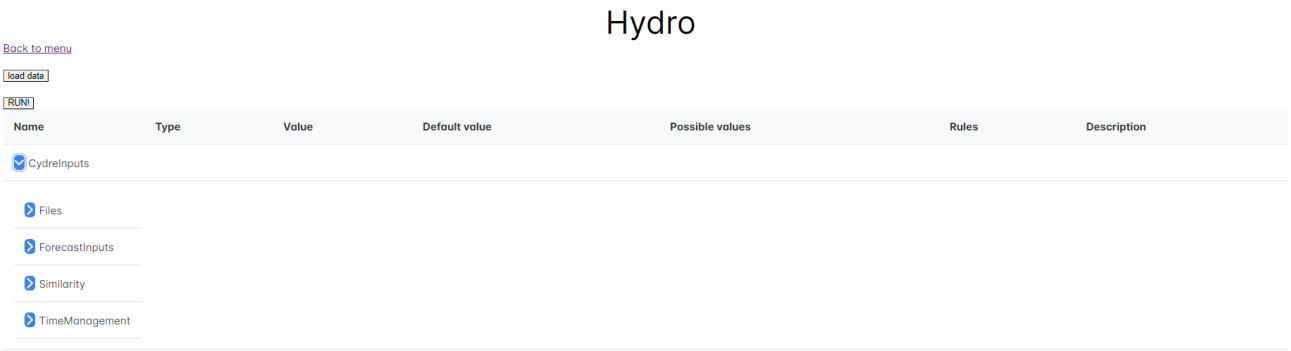

Figure 2: Front-End de l'application développé par Brian Gentile

Les détails des différents projets réalisés par les élèves de l'INSA ne seront pas abordés ici. Toutefois, le visuel présent dans la figure 3 est le Front-End du projet qui a servi de base. Pour explorer d'autres projets des étudiants de l'INSA, des visuels supplémentaires sont disponibles en annexe.

Figure 3: Front-End projet Angular du groupe LeHoang

Au début, nous avons envisagé de créer une application entièrement en Python avec Dash en raison de la simplicité de maintenance et de la relative diversité de l'interface visuel. Un visuel de la version améliorée de l'interface Dash est disponible en annexe. Cependant, nous avons finalement opté pour une solution intégrant Flask pour le backend et Angular pour le frontend. Cette décision s'est basée sur la nécessité de gérer efficacement les multiples utilisateurs avec des sessions actives et de fournir une meilleure performance. Les éléments de comparaison qui ont contribué à ce choix sont détaillés dans la TABLE 1.

Points de comparaison	Application tout en Python (Dash)	Application en Python (Backend) + et Angular (Frontend)
Complexité Technologique	Utilisation exclusive de Python pour le frontend et le backend avec Dash.	Intégration de Flask pour le backend et Angular (TypeScript/HTML/CSS) pour le frontend.
Possibilité d'Utilisation par Plusieurs Utilisateurs	Limité par Dash, qui n'est pas conçu pour gérer simultanément plusieurs utilisateurs avec des sessions actives	Gestion efficace des sessions et de l'authentification multi-utilisateur grâce à Angular
Expérience Utilisateur	Interface utilisateur interactive mais potentiellement moins dynamique	Interface utilisateur riche, dynamique et interactive grâce à Angular
Maintenance par des Non-Informaticiens	Facilement compréhensible par ceux qui qui connaissent Python	Requiers une compréhension de Angular (frontend) et Flask (backend) pour effectuer des maintenances
Performance et Optimisation	Bien adapté pour des applications de taille moyenne	Nécessite une optimisation pour gérer la communication frontend et backend.

Table 1: Tableau comparatif pour le choix entre une application entièrement en Python et une application combinant Python et Angular

6. Les cas d'utilisation

L'application doit répondre à des besoins fonctionnels, tel que pouvoir gérer les comptes, visualiser les informations des stations, pouvoir utiliser le simulateur, visualiser les résultats du simulateur et gérer l'historique des résultats de ces simulations. Les diagrammes de cas d'utilisation vont servir à modéliser les interactions entre les acteurs et le système. Pour une meilleure lecture des diagrammes de cas d'utilisation, il convient de détailler les relations standards présentes :

- Association : indique que l'acteur participe avec ce cas d'utilisation.(Une ligne simple)

- Inclus (Include) : Le cas d'utilisation inclus est exécuté à chaque fois.(Une flèche en pointillés avec "include")
- Étendu (Extend) : Le cas d'utilisation d'extension n'est exécuté que si certaines conditions sont remplies.(Une flèche en pointillés avec "extend")

6.1 Cas d'utilisation générale

Le diagramme de cas d'utilisation général nous donne une vision générale sur les fonctionnalités du système, la figure 4 montre les quatre acteurs principaux avec les cas de chacun d'une manière générale. Elle montre la hiérarchie entre les différents acteurs et l'importance de l'authentification pour avoir accès à des cas d'utilisation.

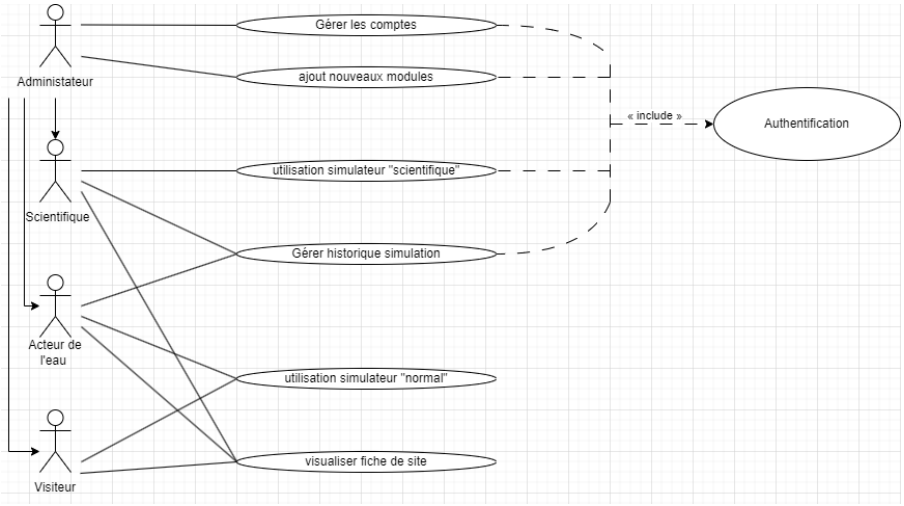

Figure 4: Diagramme des cas d'utilisation générale du projet

6.2 Cas d'utilisation " authentification "

Le cas d'utilisation "Authentification"(voir Figure 5) implique trois acteurs principaux : l'Administrateur, le Scientifique et l'Acteur de l'eau. L'objectif est de vérifier l'identité de l'utilisateur pour sécuriser l'accès au système. Pour initier ce processus, l'utilisateur doit déjà posséder un compte valide et se trouver sur l'onglet de connexion du système. Lorsqu'il saisit son nom d'utilisateur et son mot de passe, le système vérifie la validité de ces informations en consultant la base de données. Si les données sont correctes, l'utilisateur est dirigé vers son espace personnel, où il peut accéder aux fonctionnalités spécifiques à son rôle. En revanche, si les informations sont incorrectes, un message d'erreur apparaît, invitant l'utilisateur à répéter le processus d'authentification.

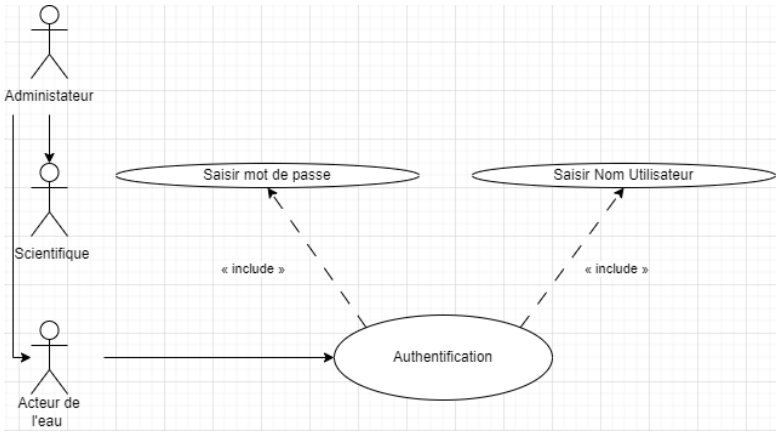

Figure 5: Diagramme d'utilisation "authentification"

6.3 Cas d'utilisation "gérer les comptes"

Le cas d'utilisation "gérer les comptes" (voir Figure 6) permet à l'Administrateur de gérer les comptes d'utilisateur en ayant la possibilité de les lister, créer, modifier et supprimer. Pour accéder à ces fonctionnalités, l'Administrateur doit être connecté et se trouver sur l'onglet de création des comptes. Une fois connecté, l'Administrateur peut afficher la liste des comptes, créer de nouveaux comptes, modifier ceux qui sont déjà présents, ou les supprimer si nécessaire. Toutes les modifications effectuées sont enregistrées dans la base de données. Si des données incorrectes sont saisies lors de ces opérations, un message d'erreur s'affiche, demandant une correction.

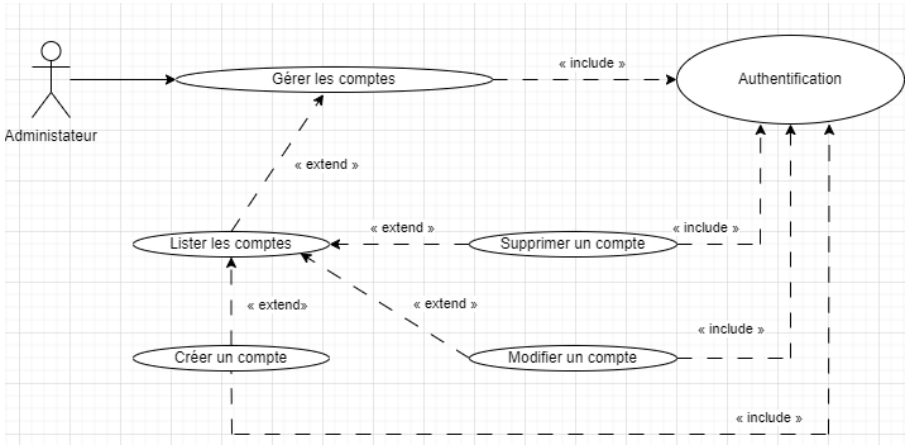

Figure 6: Diagramme d'utilisation "gérer les comptes"

6.4 Cas d'utilisation "visualiser fiche de site"

Le cas d'utilisation "Visualiser fiche de site" (voir Figure 7) permet aux utilisateurs, qu'ils soient Visiteurs, Acteurs de l'eau ou Scientifiques, de visualiser la fiche d'un site spécifique et de récupérer les données associées. Pour cela, les sites et les années de données souhaitées doivent être disponibles dans la base de données, et l'utilisateur doit être sur l'onglet correspondant. L'utilisateur sélectionne le site désiré soit à partir d'une liste, soit sur une carte, puis choisit les années pour lesquelles il souhaite visualiser des graphiques. Il peut ensuite télécharger les données associées aux graphiques. Si le site ou les années sélectionnées ne sont pas disponibles, ils ne seront pas affichés dans la liste.

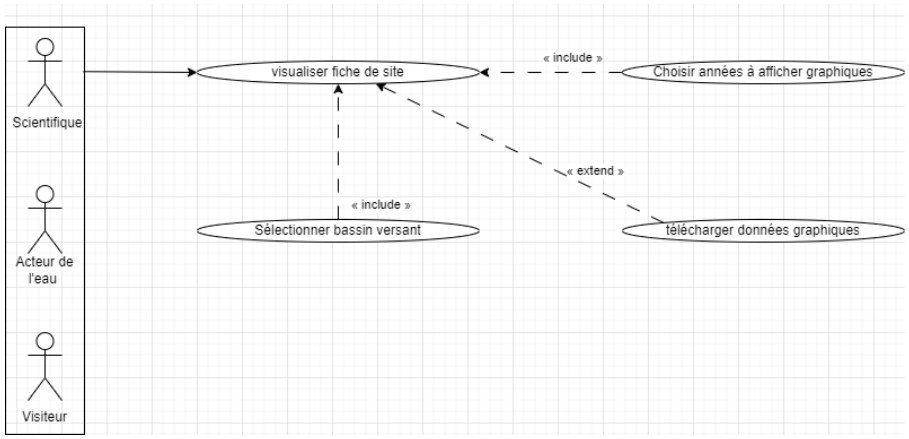

Figure 7: Diagramme d'utilisation "visualiser fiche de site"

6.5 Cas d'utilisation " utilisation simulateur 'normal'"

Le cas d'utilisation "Utilisation simulateur 'normal'"(voir Figure 8) permet aux utilisateurs, tels que les Visiteurs et les Acteurs de l'eau, de lancer une simulation, de visualiser les résultats, et de récupérer les données correspondantes. L'utilisateur doit se trouver sur l'onglet simulateur. L'utilisateur sélectionne le site, choisit une date de début pour la simulation, puis lance la simulation. Une fois les résultats générés, ils sont affichés à l'utilisateur, qui peut alors les télécharger.

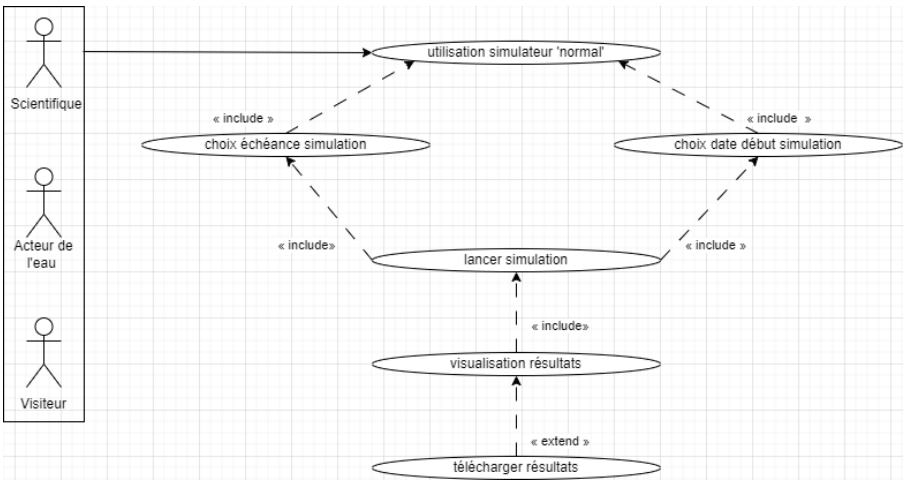

Figure 8: Diagramme d'utilisation "utilisation simulateur 'normal'"

6.6 Cas d'utilisation " utilisation simulateur 'scientifique'"

Le cas d'utilisation "Utilisation simulateur 'scientifique'"(voir Figure 9) permet aux utilisateurs ayant le rôle de Scientifique de personnaliser les simulations en modifiant des paramètres plus détaillé. Pour utiliser cette fonctionnalité, l'utilisateur doit être connecté, et se trouver sur l'onglet simulateur. Les préconditions remplies, l'utilisateur sélectionne le mode scientifique, où il peut modifier les paramètres plus détaillés. Après l'exécution de la simulation, les résultats sont affichés, et les données correspondantes peuvent être téléchargées. Si le site sélectionné n'existe pas, un message d'erreur est affiché pour informer l'utilisateur.

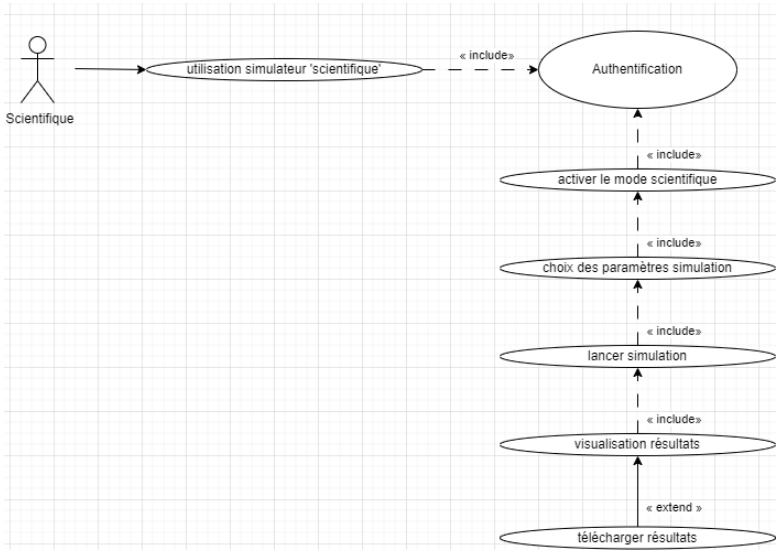

Figure 9: Diagramme d'utilisation "utilisation simulateur 'scientifique'"

7. Architecture globale de l'application web

L'application comme illustré dans la figure 11 est structurée autour de deux éléments principaux : le Back -End et le Front -End. Le Back -End exploite deux types de sources de données remplissant des fonctions spécifiques : des fichiers et une base de données MySQL.

Les fichiers XML(Extensible Markup Language) sont traités à l'aide de la librairie lxml. Dans l'application, ils sont utilisés pour gérer les paramètres d'entrée du simulateur. La Figure 10 illustre un exemple de fichier XML contenant le groupe de paramètres 'UserConfig'. Ce groupe inclut les paramètres user_watershed_id' et 'user_horizon', qui sont utilisés pour obtenir l'identifiant de la station et la durée de la simulation saisie par l'utilisateur dans l'application.

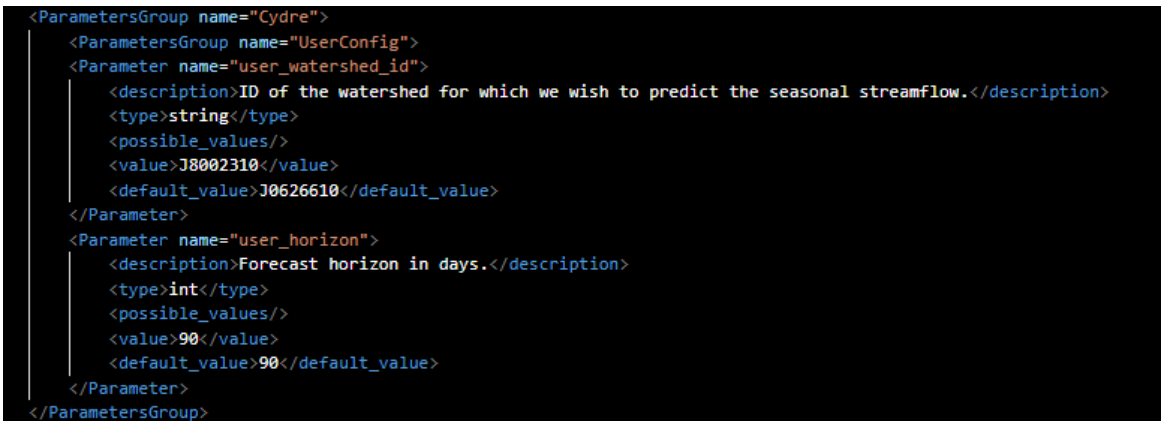

Figure 10: Vue des paramètres XML modifiés par l'utilisateur dans l'application

Les fichiers CSV(Comma-separated values), quant à eux, contiennent des séries temporelles, incluant des données climatiques(évapotranspiration, précipitations), les débits des rivières, la profondeur des nappes phréatiques, ainsi que des informations détaillées sur les stations de mesure des débits et les stations piézométriques. Ces données sont utilisées par le simulateur pour générer des prévisions de débits. Ensuite, elles sont également employées dans les fiches de site afin de permettre la visualisation sous forme de graphiques et de cartes géographiques. Leur gestion repose sur la librairie pandas.

Les fichiers SHP (shapefiles) sont un format utilisé en Systèmes d'Informations Géographiques (SIG) pour représenter des points, des lignes et des polygones. Ces fichiers gèrent et stockent des informations spatialisées, telles que les coordonnées des stations de mesure et pour coordonnées des points des périmètres des bassins versants liés aux stations. Les informations géospatiales contenues sont gérées via Géopandas (une extension géospatiale de pandas).

Enfin, la base de données MySQL stocke les informations relatives aux comptes utilisateurs ainsi que l'historique des résultats des simulations de ces derniers. Pour gérer l'interaction en la base de données et API Flask, ORM (Object -Relational Mapping) SQLAlchemy est utilisé.

L'API REST, exposée par Flask, permet la communication entre le frontend et le backend. Les composants du frontend envoient des requêtes HTTP au backend, qui traite ces requêtes (par exemple, récupéré des données, les mettre à jour, etc.) et renvoie des réponses sous forme de données JSON, lesquelles sont ensuite utilisées par le frontend pour mettre à jour l'interface utilisateur.

Le frontend est développé en Angular et comporte plusieurs services clés, bien que tous ne soient pas représentés dans la figure 11. Les services les plus importants y figurent, notamment le service JSON, qui gère le formatage des données reçues du backend, et le service Auth, qui s'occupe de l'authentification des utilisateurs. Le service Shared des variables globales, tandis que le service Users gère les profils et les préférences des utilisateurs.

Concernant les composants, bien que les détails ne soient pas entièrement fournis dans la figure 11, voici les composants :

- FicheSite.component : Ce composant génère la page correspondant à une fiche site spécifique.
- SimulateurCydre.component : Ce composant est responsable de la génération de la page du simulateur.
- SimulateurRésultat.component : Ce composant génère et affiche les résultats de la simulation, en les intégrant dans le composant SimulateurCydre.

Enfin, l'AppComponent est le composant principal qui orchestre l'ensemble des autres composants de l'application. Il sert de point d'entrée à l'utilisateur.

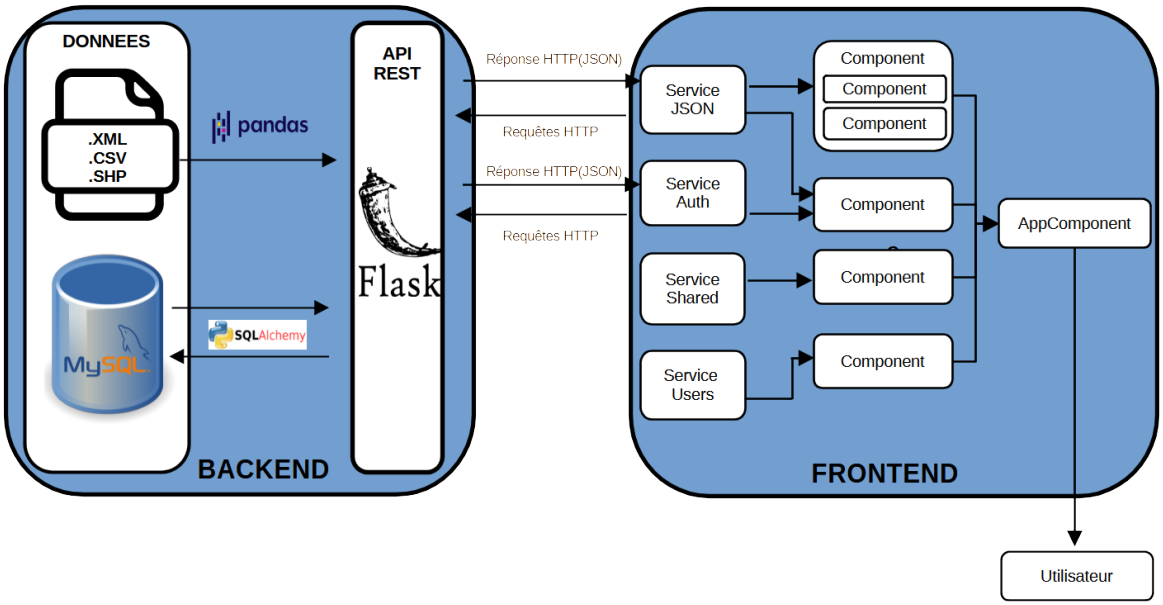

Figure 11: Architecture globale de l'application web