Rapport de Stage
DUT Informatique

Conception et développement d'une solution de visualisation de séries temporelles

Du 01/04/2019 au 21/06/2019 à OSU OREME

Réalisé par :
Solène ISSARTEL

Sous la direction de :
Olivier LOBRY
Juliette FABRE

Année Universitaire 2018-2019

Remerciements

Je souhaite, avant tout, remercier mes tuteurs de stage, Olivier Lobry et Juliette Fabre, pour m'avoir permis de travailler sur ce projet ainsi que pour m'avoir accompagnée tout au long de sa réalisation. Je les remercie de la confiance qu'ils m'ont apportée ainsi que des conseils qu'ils m'ont donnés durant le stage.

Enfin, je tiens également à remercier Alexia Vaillé et les personnes des bureaux attenants pour leur accueil chaleureux.

Sommaire

Glossaire

Architecture MVC : Le MVC (Modèle-vue-contrôleur) est une architecture logicielle destinée aux interfaces et applications web. Il est utilisé afin d'organiser le code et de faciliter l'implémentation.

Bibliothèque ou Librairie : En informatique, une bibliothèque ou librairie logicielle est un ensemble de fonctions regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire. Les fonctions sont regroupées de par leur action et leur appartenance à un même domaine (graphique, tris, autocompletion, etc).

Drag and drop : Le drag and drop (ou glisser-déposer en français) est une méthode du domaine informatique qui permet de déplacer un objet d'un endroit à un autre en utilisant la souris.

Dygraphs : Dygraphs est une librairie* graphique JavaScript permettant de représenter des données sous forme de courbes.

Fichier CSV : Un fichier CSV, qui signifie *Comma-Separated Values,* désigne un format de fichier de type tableur. Dans notre rapport, les fichiers CSV regroupent les mesures réalisées par les scientifiques et servent à tracer les graphiques.

Framework : Ensemble de composants logiciels permettant de créer l'architecture et les grandes lignes d'un logiciel. Se distingue d'une simple librairie* par son caractère générique, très peu spécialisé.

Open source : La désignation open source, ou « code source ouvert », s'applique aux logiciels dont la licence respecte des critères précis, comprenant les possibilités de libre redistribution, d'accès au code source et de création de travaux dérivés.

RENATER : Le REseau NAtional de Télécommunications pour la technologie, l'Enseignement et la Recherche est un regroupement d'organisations nationales telles que le CNRS, le CIRAD ou bien le ministère de l'Education nationale.

Série temporelle : Une série temporelle est une suite de valeurs numériques qui représentent l'évolution d'une variable au cours du temps. Dans notre rapport, l'utilisation de cette expression désignera les données relatives à une variable tracée sur un graphique.

Services et Tâches d'Observation (SO et TO) : Un Service d'Observation est une équipe composée de scientifiques qui étudient une ou plusieurs Tâches d'Observation. Par exemple, le SO ECOlogie des POPulations étudie plusieurs TO : les comportements des mésanges, des mouettes, etc.

Unité Mixte de Service : une unité mixte de service est une entité administrative créée par la signature d'un contrat d'association entre un établissement d'enseignement supérieur (le plus souvent une université) et un organisme de recherche tel que le CNRS.

Table des figures

Introduction

Réchauffement climatique, montée des eaux, pollution environnementale, évolution des populations animales et végétales… Petit à petit, chacun prend conscience des menaces climatiques et des changements naturels radicaux qui pèsent sur notre avenir et qui risquent d'impacter notre environnement. Depuis de nombreuses années, les scientifiques récoltent des données d'observation afin d'étudier les comportements liés à l'environnement. Pour ce faire, ils réalisent des observations (automatiques ou manuelles) de manière régulière à certains points stratégiques. A l'Observatoire OREME de Montpellier, les scientifiques réalisent ainsi des observations en écologie, hydrosciences, géosciences, etc.

Ces informations constituent une grande masse de données qu'il faut pouvoir stocker au fil des années afin qu'elles deviennent exploitables et qu'elles puissent donner une vision sur l'évolution des milieux étudiés. Ces données doivent ensuite être mises à disposition des scientifiques et du grand public.

Pour répondre à ces besoins, l'OREME dispose d'un portail web permettant l'accès aux données réalisées dans la région méditerranéenne et partout dans le monde. Ce site permet la consultation de données d'observation propres à différents Services d'Observation* (SO). Pour chacun de ces services, plusieurs fonctionnalités sont proposées : visualisation des données cartographiques, informations relatives aux données récoltées par les scientifiques, export des données et visualisation des séries temporelles* sous forme de graphiques interactifs. C'est sur cet aspect du site que s'inscrit le projet proposé. En effet, l'objectif est de développer une solution permettant aux utilisateurs de réaliser un affichage "à la carte" des graphiques.

Afin de mieux comprendre l'organisme et la mission du projet, une présentation de l'OREME sera faite dans un premier temps, avant de passer à l'analyse de l'existant du projet et à la solution proposée.

Ensuite seront exposés, dans le rapport technique, les choix de conception pris pour l'interface web ainsi que le résultat obtenu. Dans la troisième partie, un rapide manuel d'utilisation de la solution implémentée sera présenté. Pour finir, le document comprendra le rapport d'activité et les méthodes de travail utilisées lors de la réalisation de ce projet.

Présentation de l'organisme

L'OSU OREME

L'Observatoire de REcherche Méditerranéen de l'Environnement (OREME) est un Observatoire des Sciences de l'Univers (OSU) qui a été créé en 2007 et qui est aujourd'hui dirigé par Eric SERVAT. Il se consacre aux activités scientifiques concernant les risques naturels et s'intéresse à l'impact des changements globaux ainsi que ceux liés à l'être humain sur l'espace méditerranéen et dans le monde.

L'Observatoire est une Unité Mixte de Service* basée à l'Université de Montpellier. Il est sous tutelle de quatre organismes :

UM L'Université de Montpellier
CNRS Le Centre National de Recherche Scientifique
IRD L'Institut de Recherche pour le Développement
IRSTEA L'Institut National de Recherche en Sciences et Technologies pour l'Environnement et l'Agriculture

Huit laboratoires de la communauté scientifique des sciences de l'univers et de l'environnement constituent le cœur de l'observatoire. Ce sont tous des Unités Mixte de Recherche (UMR). On y trouve :

GM Géosciences Montpellier
HSM Hydrosciences Montpellier
CEFE Le Centre d'Ecologie Fonctionnelle et Evolutive
G-EAU Gestion de l'Eau, Acteurs, Usages
ISEM Institut de Sciences de l'Evolution de Montpellier
LUPM Laboratoire Univers et Particules de Montpellier
MARBEC Biodiversité marine et ses usages
TETIS Territoires Environnements, Télédétections et Information Spatiale

Les missions de l'OREME sont :

- Soutenir l'activité ou le développement d'observation systématique en science de l'univers et de l'environnement ;
- Soutenir la construction de bases de données environnementales ouvertes, partagées, référencées au niveau international ;
- Encourager la mutualisation des moyens analytiques (observation, expérimentation, modélisation) et des savoir-faire ;
- Constituer le relais local des réseaux nationaux d'observations, et un acteur fort des actions tournées vers la Méditerranée en environnement.

L'OREME est structuré en différents Services d'Observation et plusieurs plateformes (*cf organigramme Annexe A*), dont le service Système d'Information.

Durant le stage, je serai intégrée à ce service composé de deux ingénieur·es. Leurs principales missions sont de :

- Pérenniser les données de l'Observatoire : stockage, sauvegarde, archivage, structuration et description ;
- Valoriser les données : diffusion, catalogage, référencement ;
- Améliorer la qualité des données : outils de saisie, d'aide à la saisie, validation, surveillance etc. ;
- Permettre le croisement des données avec des bases de données nationales et internationales, et avec des outils et applications métier.

1 Analyse

1.1 Analyse de l'existant

1.1.1 Analyse du portail des données en ligne de l'OREME

Le portail de données de l'OREME est un site web qui diffuse les données relatives aux activités scientifiques réalisées par l'organisme. Il possède de nombreuses fonctionnalités permettant de trouver des informations liées à différents Services d'Observation*.

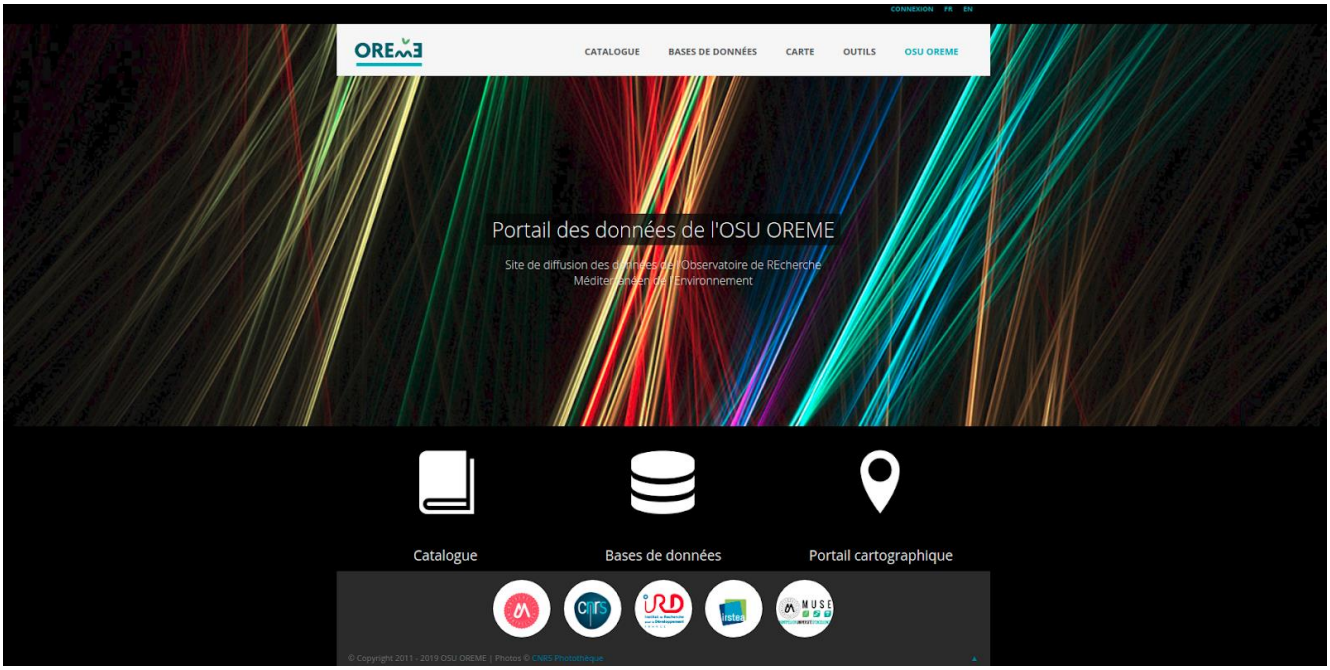

Figure 1 : Page d'accueil du portail des données OREME

Les technologies utilisées dans le site existant sont :

- CodeIgniter qui est un framework open source* utilisant le langage PHP. Il suit le motif de conception MVC* (Model View Controller) ;
- Le langage SQL pour interagir avec la base de données PostgreSQL et accéder aux informations de celle-ci ;
- Bootstrap qui est un framework* se focalisant sur le graphisme, l'animation et l'interaction avec la page dans le navigateur. Il est basé sur les langages HTML, CSS et Javascript ;
- Le langage Javascript pour apporter de l'interaction aux pages créées. Les bibliothèques* jQuery et jQuery UI qui permettent de faciliter l'utilisation de Javascript et qui possèdent différentes fonctionnalités intéressantes ;
- Les langages HTML5 et CSS3 pour la conception de notre site internet et de son aspect visuel.

1.1.2 La base de données PostgreSQL

Une base de données PostgreSQL stocke toutes les données scientifiques. La structure de la base est organisée de façon à ce qu'un schéma corresponde à un Service d'Observation (SO)* ou une Tâche d'Observation (TO). Un schéma est composé de différents éléments SQL tels que des tables, des vues ou des triggers. Chaque table est composée de différents attributs, aussi appelés colonnes. On retrouve dans chaque schéma, des tables correspondantes à différentes informations concernant les stations de mesure, les instruments ou encore les séries temporelles dont est composé le SO ou la TO, et des tables contenant les données elles-mêmes. Les informations et données diffusées sur le portail web sont issues de ces différents schémas et tables.

1.1.3 Module de visualisation des séries temporelles

Le portail propose pour les différents SO et TO des pages de graphiques permettant la visualisation interactive des séries temporelles. Ces pages sont basées sur un modèle générique.

Modèle de représentation des pages de graphiques (graph_tree)

A chaque Service ou Tâche d'Observation correspondent un ou plusieurs *thèmes* de graphiques, représentés par des pages web. Chaque thème contient des *groupes* qui sont représentés par des onglets. Chaque groupe contient un certain nombre de *graphiques*. Chaque graphique contient un certain nombre de *variables*. Enfin, les variables sont parfois composées de plusieurs sous-groupes de données.

Par exemple, une variable concernant des mesures de pluviométrie peut contenir différents sous-groupes de données selon la localisation de la mesure. La notion de *discriminant de variable* permet de distinguer ces groupes. On peut ainsi trouver grâce aux discriminants les données relatives à une seule station de mesure.

A titre d'exemple, le Service d'Observation ECOPOP étudie les communautés animales telles que les mésanges, les mouettes ou encore les babouins et comporte une TO "mésanges". Sur la page de graphiques de cette TO (*cf Figure 2*), chaque onglet / groupe représente une station, et sur le premier groupe, celui de la station "La Rouvière" on trouve deux graphiques. Chacun d'entre eux possède une ou plusieurs variables, comme on peut le voir pour le premier graphique.

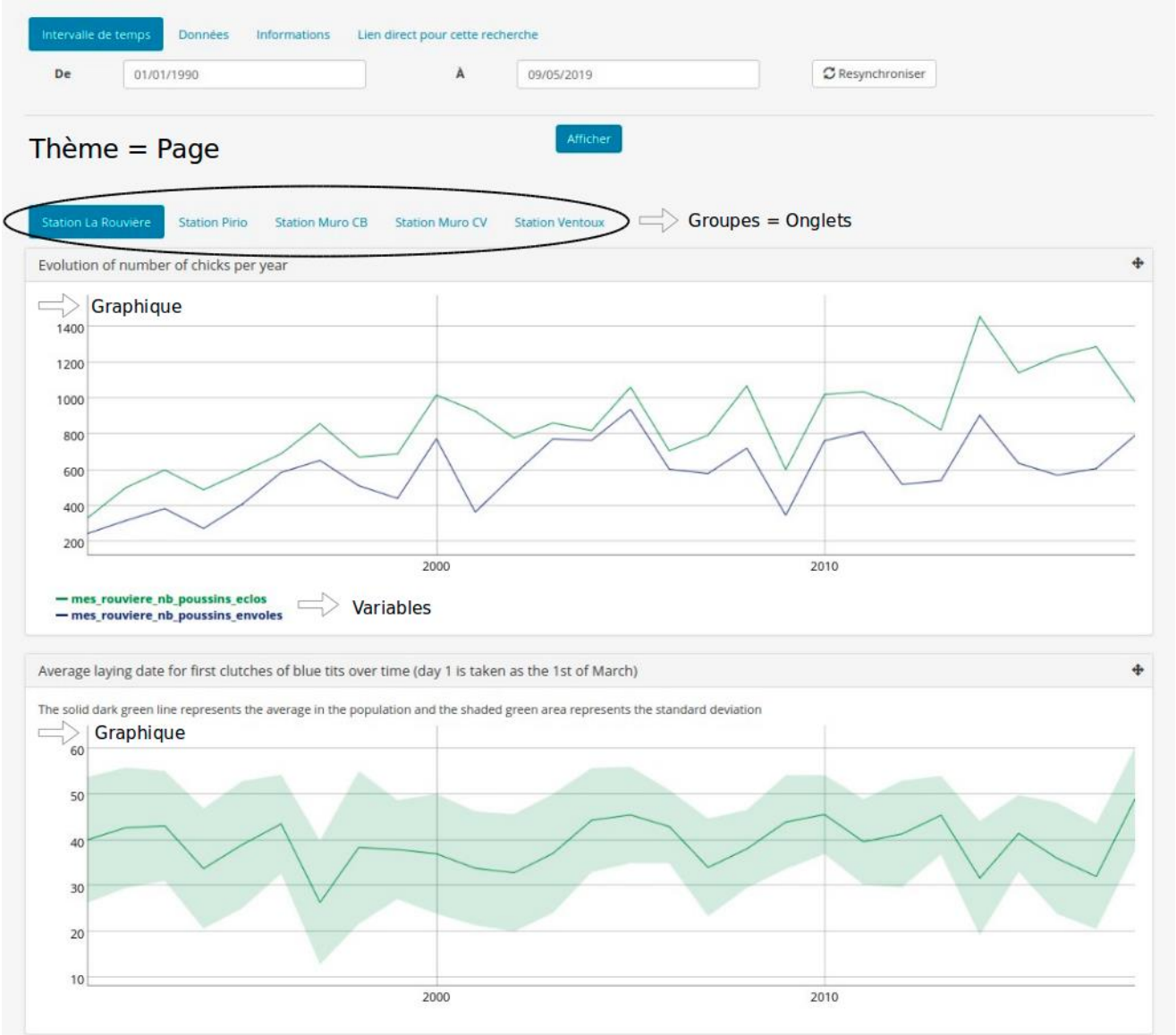

Figure 2 : Représentation de chaque partie sur la page du site

Implémentation en base de données

Le modèle est implémenté dans la base de données relationnelle de l'observatoire (*cf Figure 3*). Cinq tables décrivent ainsi les différents thèmes, groupes, graphiques, variables et discriminants. Ce sont dans ces tables que sont récupérées, via des fonctions génériques, les informations définissant la structure de chaque page de graphiques du site.

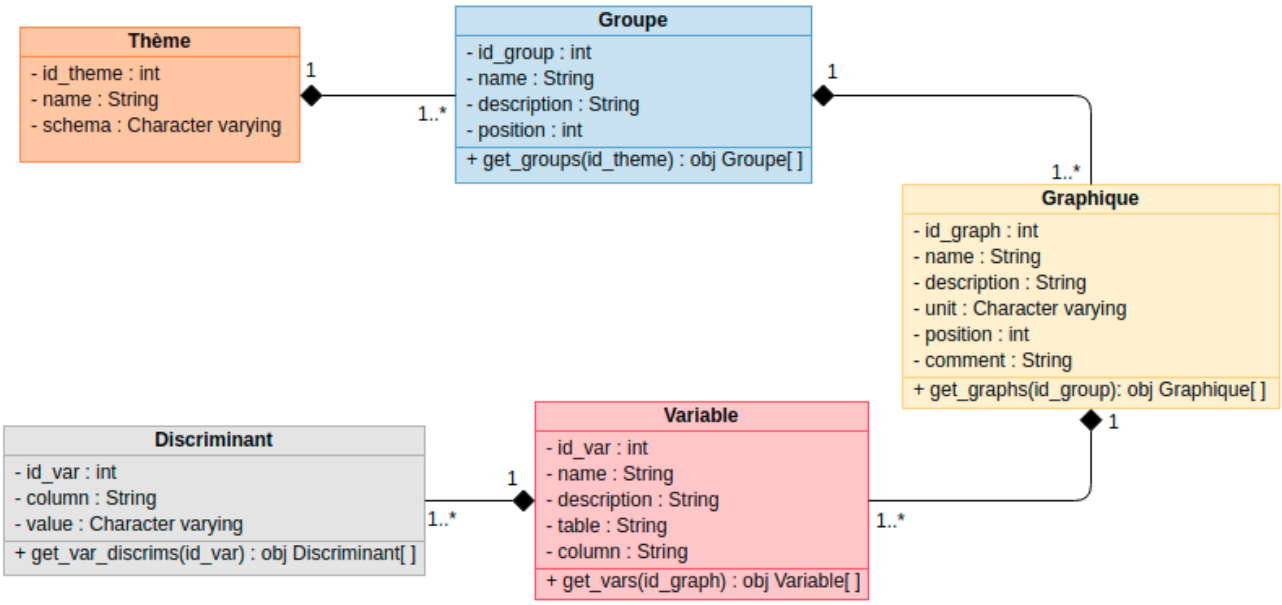

Figure 3 : Diagramme de classe du "graph_tree"

Chacune des variables - ou séries temporelles - tracées est répertoriée dans la base de données. Le champ *schéma* de la table *thème* ainsi que les champs *table* et *column* de la table *variable* permettent de trouver les mesures correspondantes aux graphiques à tracer.

Implémentation en PHP

La communication entre la base de données et les pages de graphiques du site se fait à travers le modèle *graph_model* implémenté en PHP. Ce modèle est composé de différentes fonctions qui ont pour objectif d'extraire les informations concernant la structure des pages de graphiques. Ce modèle permet, entre autres, de créer le graph_tree permettant de structurer la page grâce à un tableau associatif qui permet de représenter l'arborescence (thème, groupe, graphique, etc.) (*cf Annexe B*). Il permet aussi de générer les fichiers CSV* contenant les mesures scientifiques réalisées pour chaque graphique contenu dans la page. Ce sont ces fichiers qui seront utilisés pour tracer les graphiques.

Bibliothèque Dygraphs

Afin de tracer chaque graphique interactif, la bibliothèque Dygraphs* est utilisée. C'est une bibliothèque graphique JavaScript permettant de représenter des données sous forme de courbes. Les graphiques sont tracés à partir des données extraites au format CSV*.

Cette librairie permet de rajouter à un site web des graphiques interactifs sur lesquels le visiteur a la possibilité d'effectuer un certain nombre d'actions telles que zoomer, dézoomer et se déplacer (ou des actions personnalisables qui peuvent être rajoutées lors de l'implémentation).

Par exemple, le graphique de pluviométrie de la station de la Blaquererie (*cf Figure 4*), disponible sur le site, permet aux utilisateurs de zoomer sur des données jusqu'à l'échelle journalière.

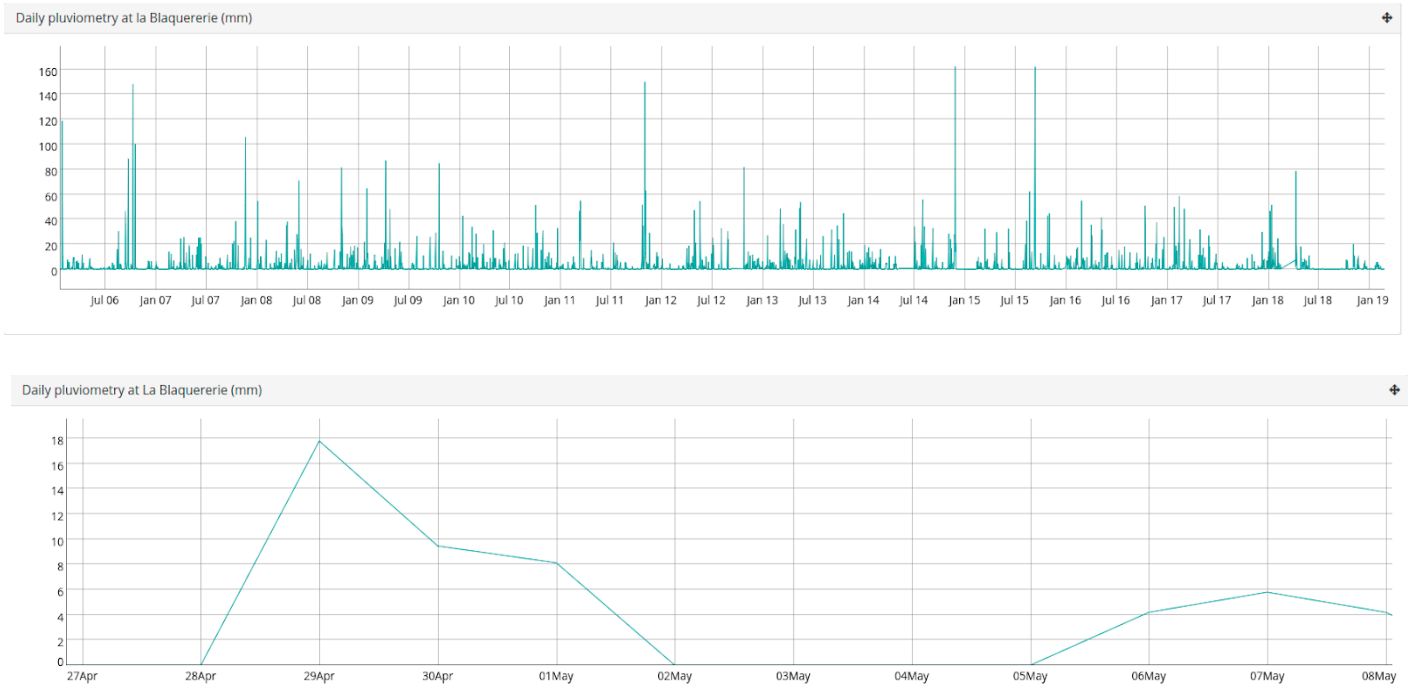

Figure 4 : Graphiques interactifs de pluviométrie

Communications

Pour avoir une meilleure idée des communications entre chaque partie du code, le diagramme de séquence de l'affichage des graphiques (*cf Figure 5*) a été réalisé afin d'aider à la compréhension du code et de son déroulement. Il a permis d'analyser chaque partie composant le MVC des pages de graphiques existantes.

Lors de la consultation d'une page de graphiques disponible sur le site, l'interface se charge en appelant tout d'abord la méthode *index* du contrôleur *graphs* prenant en paramètre le type de données que la page va afficher. Le plus souvent cette variable correspond à un thème. Par la suite, le contrôleur va demander au modèle, nommé *graph_model*, les informations relatives au thème comme les dates de début et de fin des observations.

Une fois toutes les informations récupérées, le contrôleur va de nouveau faire appel au modèle en lui demandant de créer le graph_tree correspondant au thème. Le modèle s'occupe aussi de créer les fichiers CSV afin de les communiquer à la librairie Dygraphs qui se chargera de tracer les graphiques.

L'affichage de la page se fait à travers une vue nommée *graphs*. Elle permet de structurer la page en différentes parties.

En effet, chacune des pages de graphique est composée de différentes parties générées par des vues qui leur sont spécifiques. On retrouve :

- La partie formulaire permettant de modifier l'intervalle de temps ou bien d'avoir accès au lien direct vers la page, générée à partir de la vue *forms* ;
- La partie permettant de structurer et de tracer les graphiques, générée à partir de la vue *tabs ;*
- La partie d'export des données permettant de télécharger les fichiers CSV liés aux graphiques tracés, générée à partir de la vue *data-export*.

Pour gérer l'affichage des graphiques, ce sont donc les vues *graphs* et *tabs* qui sont appelées.

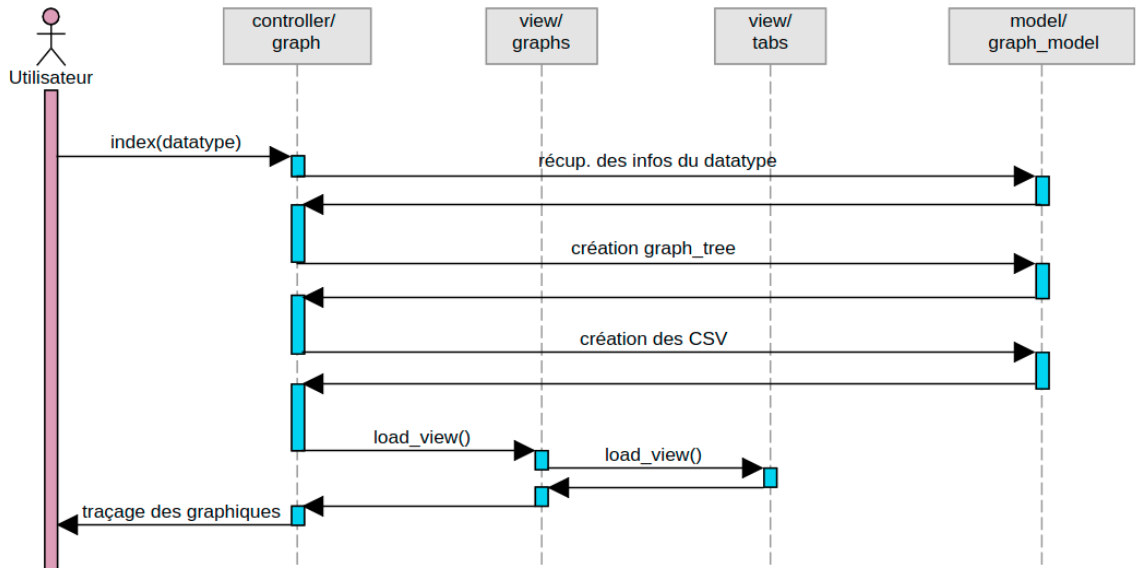

Figure 5 : Diagramme de séquence du traçage des graphiques

1.1.6 Authentification et droits associés

L'utilisateur à la possibilité de s'authentifier via la page de connexion, à travers la fédération d'identité proposée par RENATER*. La librairie* CIzACL du framework CodeIgniter permet ensuite de gérer les informations de connexion et les droits associés à chaque utilisateur.

Une fois connecté, le visiteur peut donc avoir accès à des fonctionnalités et des informations liées à ses droits.

1.2 Cahier des charges

1.2.1 Contexte et définition du problème

Le module de visualisation des graphiques connaît des limites qui sont liées à son organisation et sa structure. En effet, la structure des données est :

- *figée*, car les utilisateurs ne peuvent pas modifier l'organisation et le contenu des pages de graphiques ;
- *restrictive*, car les séries temporelles ne sont pas forcément toutes proposées à la visualisation sur le site ;
- *orientée*, car les administrateurs proposent une manière de visualiser les données (organisation en thèmes, onglets, etc.) qui ne correspond pas forcément aux besoins des utilisateurs.

De plus, cette organisation ne leur permet pas d'avoir accès à toutes les données ni de croiser des séries issues de thèmes et de groupes différents. Enfin, les administrateurs ne peuvent pas facilement administrer les pages de graphiques. En effet, actuellement, la gestion des pages de graphiques se fait via une interface CRUD (Create Read Update Delete) permettant de modifier, d'ajouter ou de supprimer les informations relatives à la structure des pages de graphiques. Cependant, ce moyen ne se fait qu'à partir de données textuelles et ne possède pas d'aspect visuel, ne facilitant pas l'administration des pages.

1.2.2 Objectifs du projet

Le projet a pour objectif de permettre la personnalisation du choix des graphiques pour tous les utilisateurs, qu'ils soient authentifiés ou non. En effet, chaque utilisateur aura la possibilité de créer une page éphémère qui sera consultable pendant la durée de vie d'une session. Il aura tout de même la possibilité de sauvegarder sa page après authentification afin que celle-ci soit disponible lors de ses futures visites.

Lors de la personnalisation, l'utilisateur aura différentes possibilités qui lui permettront de personnaliser sa page, par exemple il pourra choisir un certain nombre de séries temporelles à visualiser et à ordonner.

Le but est que l'interface soit simple mais assez complète afin que l'utilisateur n'ait pas beaucoup de manipulation à effectuer. L'interface devra respecter la charte graphique en vigueur afin que les pages générées soient intégrées le mieux possible. Pour cela, l'interface devra être implémentée en suivant les normes du code déjà présentes afin d'avoir un résultat propre et compréhensible par tous les développeurs.

Enfin, et si cela est possible, l'interface devra permettre une certaine généricité afin que l'interface des graphiques à la carte soit disponible au sein de chaque Service d'Observation* et de chaque schéma de la base de données. De plus, les utilisateurs pourront également visualiser des séries temporelles issues de différents SO ou schéma.

Par ailleurs, l'interface pourra être utilisée par les administrateurs pour créer les pages de graphiques prédéfinies.

1.2.3 Public visé

L'interface Web devra être disponible et utilisable par tous les utilisateurs du site, c'est-à-dire par les scientifiques, le grand public et les administrateurs du site.

1.2.4 Analyse des besoins fonctionnels des utilisateurs

Pour pouvoir répondre aux besoins des utilisateurs, la réalisation d'une analyse détaillée m'a permis de visualiser chaque action et besoin de chaque acteur qui intervient, de près ou de loin, dans l'utilisation de l'interface.

La réalisation du diagramme d'utilisation rassemble les actions que chaque acteur réalise (*cf Figure 6*). Le but n'étant pas de détailler chacune d'entre elles, mais d'avoir une vue d'ensemble de l'utilisation de l'interface à implémenter.

On peut donc voir que les trois acteurs principaux sont les utilisateurs non connectés, les utilisateurs connectés et les administrateurs.

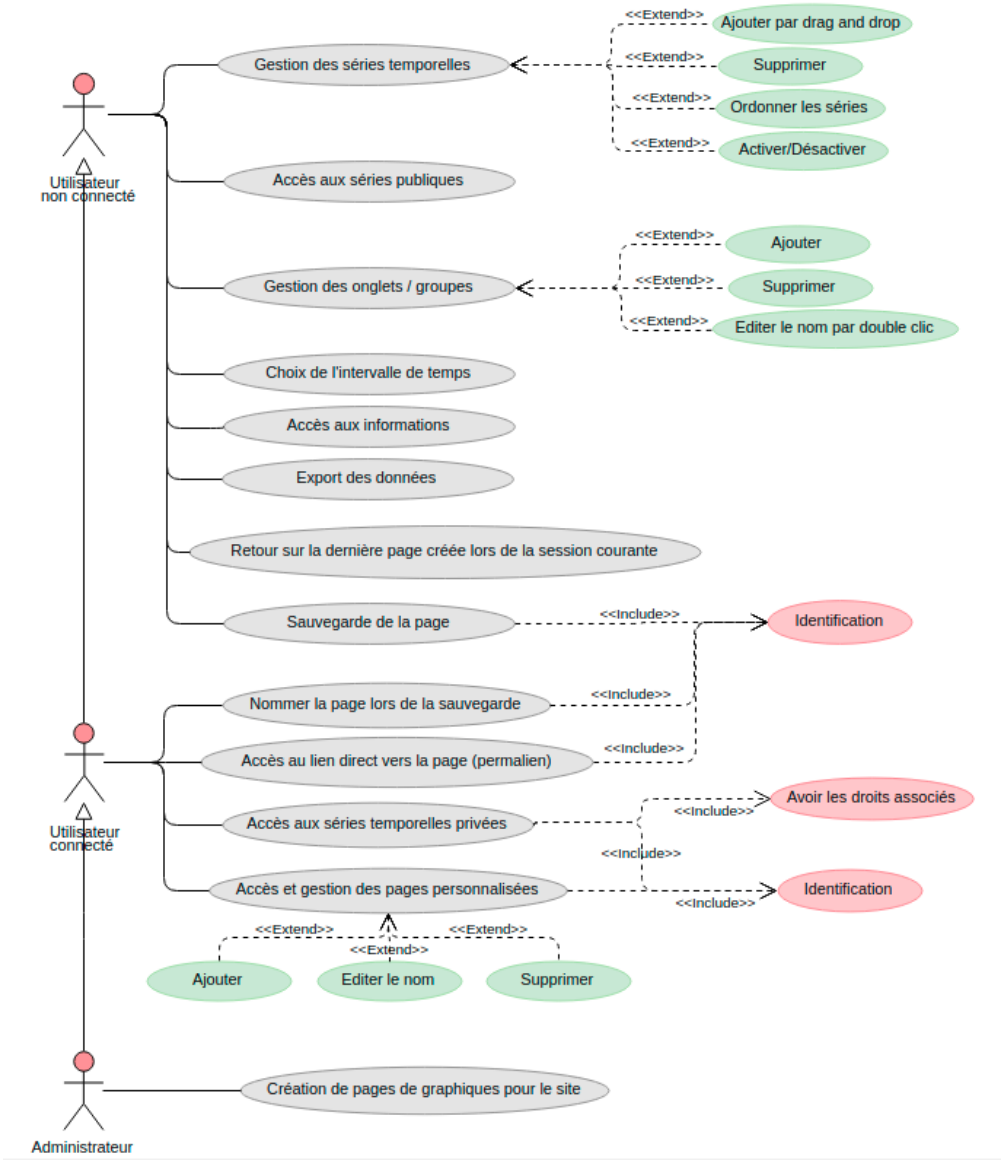

Figure 6 : Diagramme de cas d'utilisation

Pour répondre aux besoins, nous avons donc choisi d'implémenter les fonctionnalités suivantes :

Tous les utilisateurs pourront :

- Avoir accès aux graphiques des séries temporelles publiques du site ;
- Gérer les graphiques des séries temporelles afin de personnaliser le contenu de la page :
  - Ajouter des graphiques de séries par "drag and drop"* ;
  - Supprimer des séries ajoutées sur la page via une icône 'supprimer' ;
  - Ordonner les séries ;
  - Choisir d'activer ou non chaque graphique de la page via une icône 'checkbox' ;
  - Revenir sur la dernière page créée lors de la session courante.
- Gérer les onglets de la page :
  - Ajouter des onglets via une icône '+'
  - Supprimer un onglet via une icône 'supprimer' (sauf pour l'onglet 1 qui n'est pas supprimable) ;
  - Éditer le nom d'un onglet en double-cliquant sur celui-ci ;
- Choisir l'intervalle de temps des graphiques ;
- Accéder aux informations des graphiques concernant les formats utilisés ou l'utilisation des outils mis à disposition ;
- Exporter les données des graphiques sous format CSV* lorsque les droits sont suffisants ;
- Sauvegarder la page personnalisée via un bouton 'sauvegarder' qui redirige vers la page de connexion.

Les utilisateurs connectés pourront de plus :

- Sauvegarder directement la page ;
- Choisir un nom pour la page qu'ils viennent de créer ;
- Avoir accès à toutes leurs pages personnalisées.

Les administrateurs pourront de plus :

- Utiliser cette page de personnalisation pour créer de nouvelles pages de graphiques prédéfinies ;
- Gérer les pages de graphiques prédéfinies déjà créées.

1.2.4 Analyse des besoins non fonctionnels

La portabilité du site est un critère important dans la réalisation du projet. Il faut donc faire en sorte que l'interface soit disponible sur les différents navigateurs suivants : Firefox, Chrome et Safari.

2 Rapport technique

Le projet qui m'a été proposé doit être intégré au site existant c'est pourquoi la charte graphique ainsi que la logique du code doivent être respectés.

Pour ce qui est de l'aspect technique, les technologies utilisées sont les mêmes que celles du site existant. Ce choix a été fait afin de faciliter l'intégration de l'interface ainsi que pour garder la forme actuelle du code.

2.1 Conception

2.1.1 Maquette de la page de personnalisation

Dans l'objectif de concevoir le visuel et l'organisation des éléments dans la page de personnalisation, la création de maquettes a permis de représenter chaque partie qui compose la page.

Dans la maquette finale (*cf Figure 7*), on retrouve différents éléments :

- Un panneau latéral composé d'un champ textuel permettant aux utilisateurs d'entrer des mots-clés pour filtrer les séries temporelles. Les séries sont affichées sous forme de liste sous ce champ ;
- Une section de personnalisation composée d'onglets et de divisions permettant d'avoir accès aux séries temporelles déposées. Les éléments représentant les graphiques sont structurés en deux parties, l'une pour le nom de la série et son unité ainsi que trois icônes et l'autre partie pour le graphique. Les icônes permettent d'activer / désactiver, supprimer ou ordonner les graphiques ;
- Un bouton permettant de soumettre les modifications réalisées par les utilisateurs.

L'utilisateur a aussi la possibilité d'ouvrir et fermer le panneau latéral afin de gérer la largeur de la section de personnalisation et donc d'agrandir les graphiques. De plus, il peut gérer le nombre d'onglets et les renommer.

Figure 7 : Maquette de la page de personnalisation

2.1.2 Analyse des actions de chaque acteur

Les différents acteurs réalisent plusieurs actions au cours de l'utilisation de l'interface. Pour permettre de rendre chaque action plus claire et précise, j'ai établi un diagramme d'activité permettant de lier chaque action à chaque entité à laquelle l'interface fait appel.

Le diagramme d'activité réalisé détaille de façon plus développée la partie principale du projet qui est la gestion des séries temporelles et des graphiques dans le page (*cf Figure 8*). Il permet de pousser plus loin l'analyse du diagramme de cas d'utilisation présenté plus haut (*cf Figure 6*). En effet, chaque action est détaillée afin de montrer toutes les possibilités offertes à l'utilisateur.

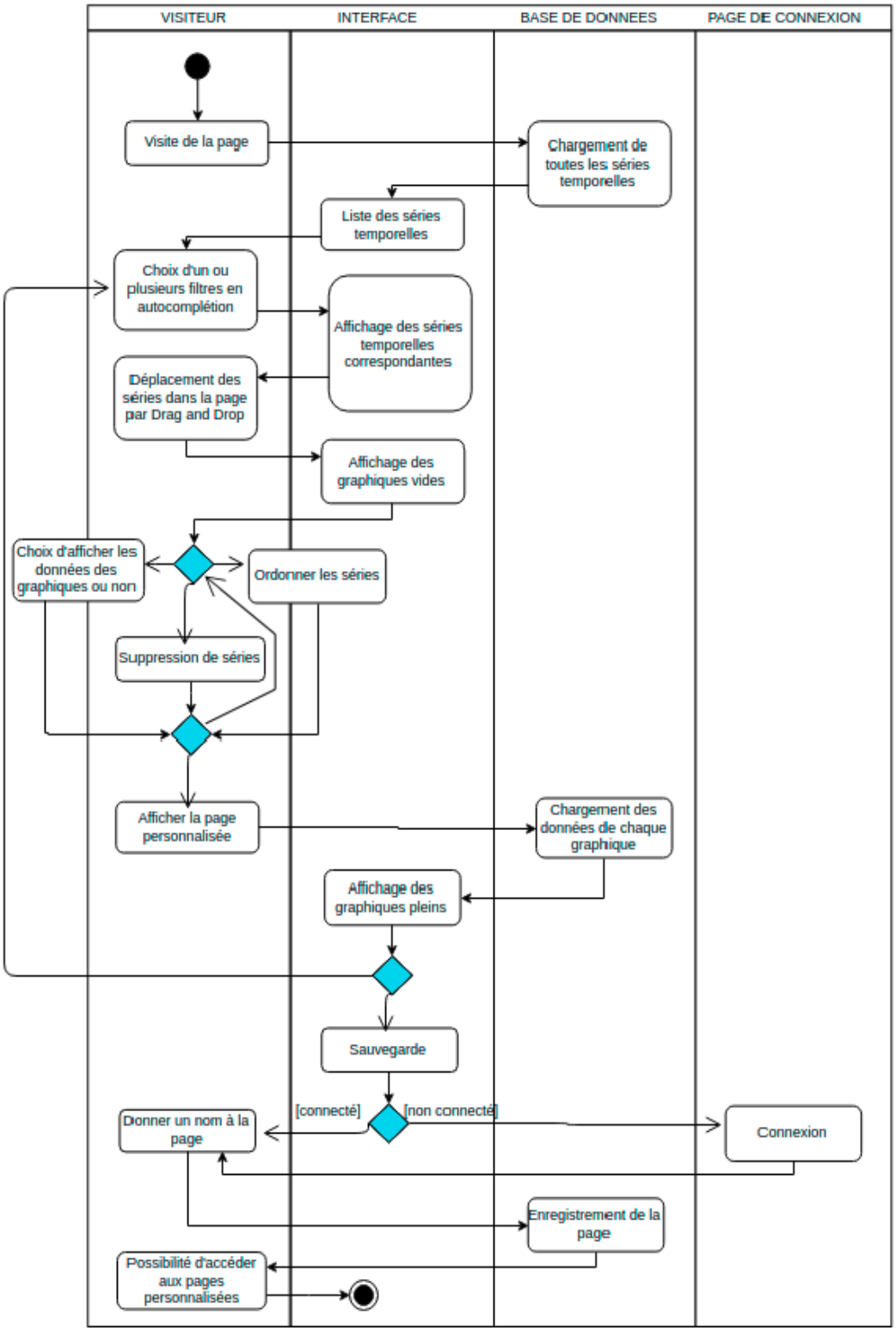

Figure 8 : Diagramme d'activité de la gestion des séries temporelles

Lors de la première visite, la structure de la page est vide. On trouve uniquement le panneau latéral, servant à saisir des mots-clés et filtrer les séries temporelles.

Après le chargement de cette page, la liste des séries temporelles visualisables est disponible à l'utilisateur. Cependant, cette liste étant trop longue, le visiteur doit commencer par choisir un ou plusieurs critères afin de filtrer des séries correspondant à sa recherche dans le panneau prévu à cet effet.

Pour cela, le visiteur doit taper et sélectionner des mots-clés correspondants à sa recherche (comme la localisation des données, la catégorie, le paramètre mesuré, etc.) dans l'espace prévu afin d'afficher la liste des séries correspondante aux filtres sélectionnés. Par la suite, le visiteur peut gérer les graphiques des séries temporelles : les ajouter, les supprimer, les ordonner ou encore les activer / désactiver afin d'afficher, ou non, les données d'un graphique.

C'est ensuite lorsqu'il confirme sa personnalisation que les graphiques vont être tracés. L'utilisateur a ensuite deux choix, soit sauvegarder la page qu'il vient de réaliser, soit continuer de modifier la page.

Si l'utilisateur choisit de sauvegarder et qu'il n'est pas connecté, il va être automatiquement redirigé vers la page de connexion. Une fois connecté, il pourra donner un nom à la page et avoir accès à toutes ses pages personnalisées.

.

2.1.2 Analyse des communications

Après avoir analysé le code existant en rapport avec l'interface à implémenter, l'analyse des communications entre chaque partie du code a été très importante pour visualiser la logique du code et permettre d'approfondir le concept.

En effet, le choix de la création des diagrammes de séquence (*cf Figure 9 et Figure 10*) permet de structurer le code et de réaliser un chemin de communication clair entre chaque partie du MVC.

En ce qui concerne la structure de l'interface à implémenter, le MVC est composé d'un modèle, d'un contrôleur et de deux vues principales. Cette structure permet une communication organisée grâce aux différents éléments qui la compose. En effet, le modèle permet de charger les données nécessaires directement via la base de données, les vues ont pour rôle d'afficher les différentes parties de la page (panneau latéral et affichage des graphiques) ainsi que le contrôleur permettant la liaison entre les deux.

Affichage initial de la page

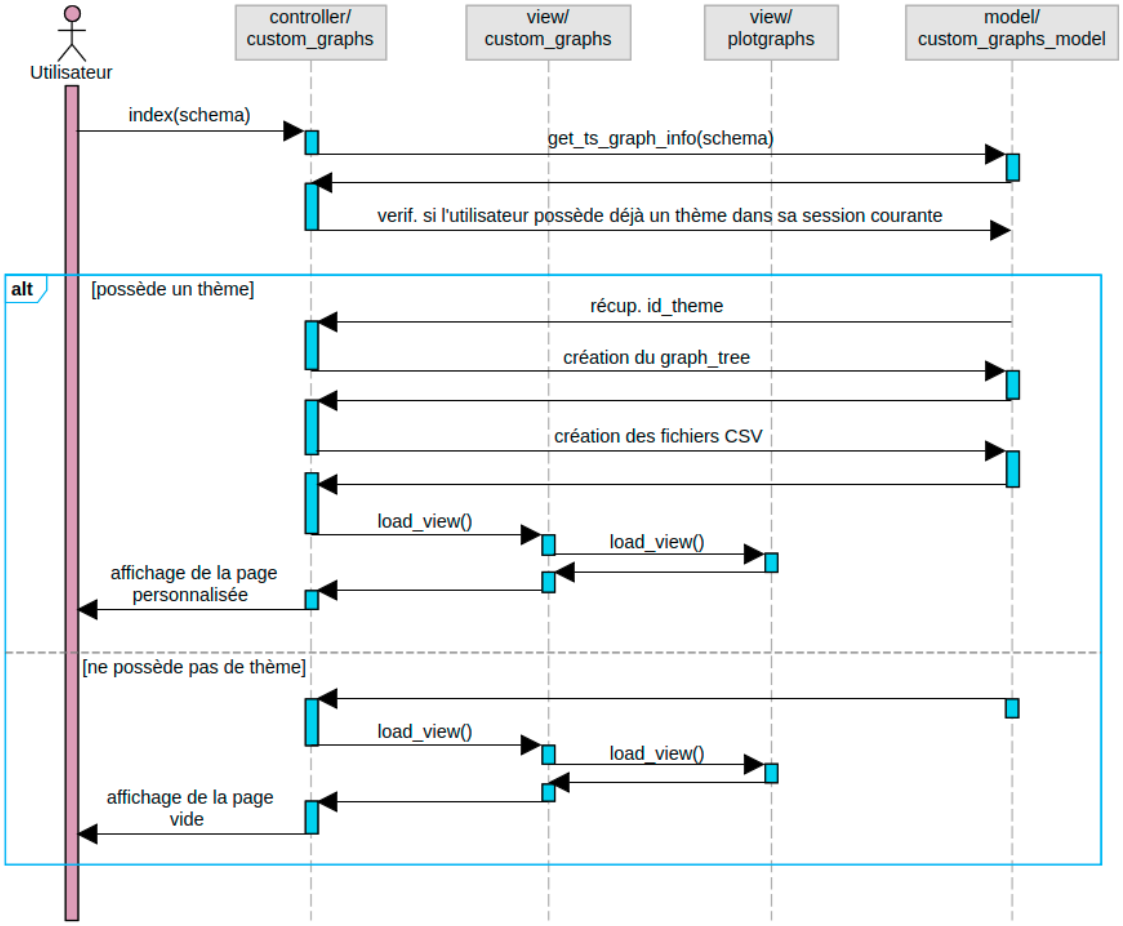

Figure 9 : Diagramme de séquence de l'affichage de l'interface

On remarque que le déroulement des actions de l'affichage de la page et des graphiques est très ressemblant à celui de l'existant. Cependant, on retrouve des différences qui permettent d'adapter le code à la personnalisation des pages.

Pour mettre en place chaque partie, l'interface va tout d'abord faire appel au contrôleur *custom_graphs* via la fonction *index*, prenant en paramètre un schéma dans la base de données correspondant à un Service ou Tâche d'Observation.

Le contrôleur va alors charger toutes les série temporelles en appelant le modèle *graph_model* qui va les récupérer dans la base de données grâce à la fonction *get_ts_graph_info*.

Il y a ensuite deux possibilités :

- Soit l'utilisateur ne possède pas de thème dans sa session courante, c'est à dire que c'est sa première consultation de la page, ce qui entraîne un affichage de la structure de la page vide ;
- Soit l'utilisateur possède un thème dans sa session courante mais ne l'a pas sauvegardé, et on affiche ce thème.

Ce système est pratique car si l'utilisateur n'est pas connecté, il a la possibilité d'avoir accès à sa page le temps de durée de sa session (car le thème est lié à l'identifiant de cette session). Il n'est donc pas obligé de sauvegarder sa page dans ses préférences pour y avoir accès.

Si l'utilisateur est connecté et qu'il souhaite accéder à l'une de ses pages personnalisées et sauvegardées c'est un autre système, assez similaire, qui est mis en place. Lorsqu'il décide d'accéder à son thème, l'identifiant de ce thème est mis en paramètre de la fonction *index*, permettant d'afficher la page préalablement sauvegardée et de tracer les graphiques qui y avait été insérés.

Par la suite le contrôleur va se charger de l'affichage de la page en appelant la vue *custom_graph* mettant en place les séries temporelles dans un panneau latéral. Cette vue charge elle-même, dans une division HTML, la vue *plotgraphs* qui permet d'afficher les différents onglets et graphiques qui composent la page.

Personnalisation de la page

Par la suite, pour ce qui est du déroulement des actions, on choisit un utilisateur non connecté, afin d'analyser les actions à réaliser jusqu'à l'enregistrement d'un thème.

Il commence la personnalisation de sa page en filtrant les séries temporelles disponibles en entrant des mots-clés dans l'espace prévu à cet effet. Il peut alors ajouter des séries et des onglets afin de personnaliser la page comme il le souhaite. Quand le visiteur décide d'afficher la page et clique sur le bouton *Afficher*, il va lancer le processus de création d'un nouveau thème. Pour cela, la fonction *"create_graphs"* du contrôleur est appelée via un appel Ajax. Elle va ensuite communiquer avec le modèle qui va permettre d'insérer toutes les informations de la page dans la base de données.

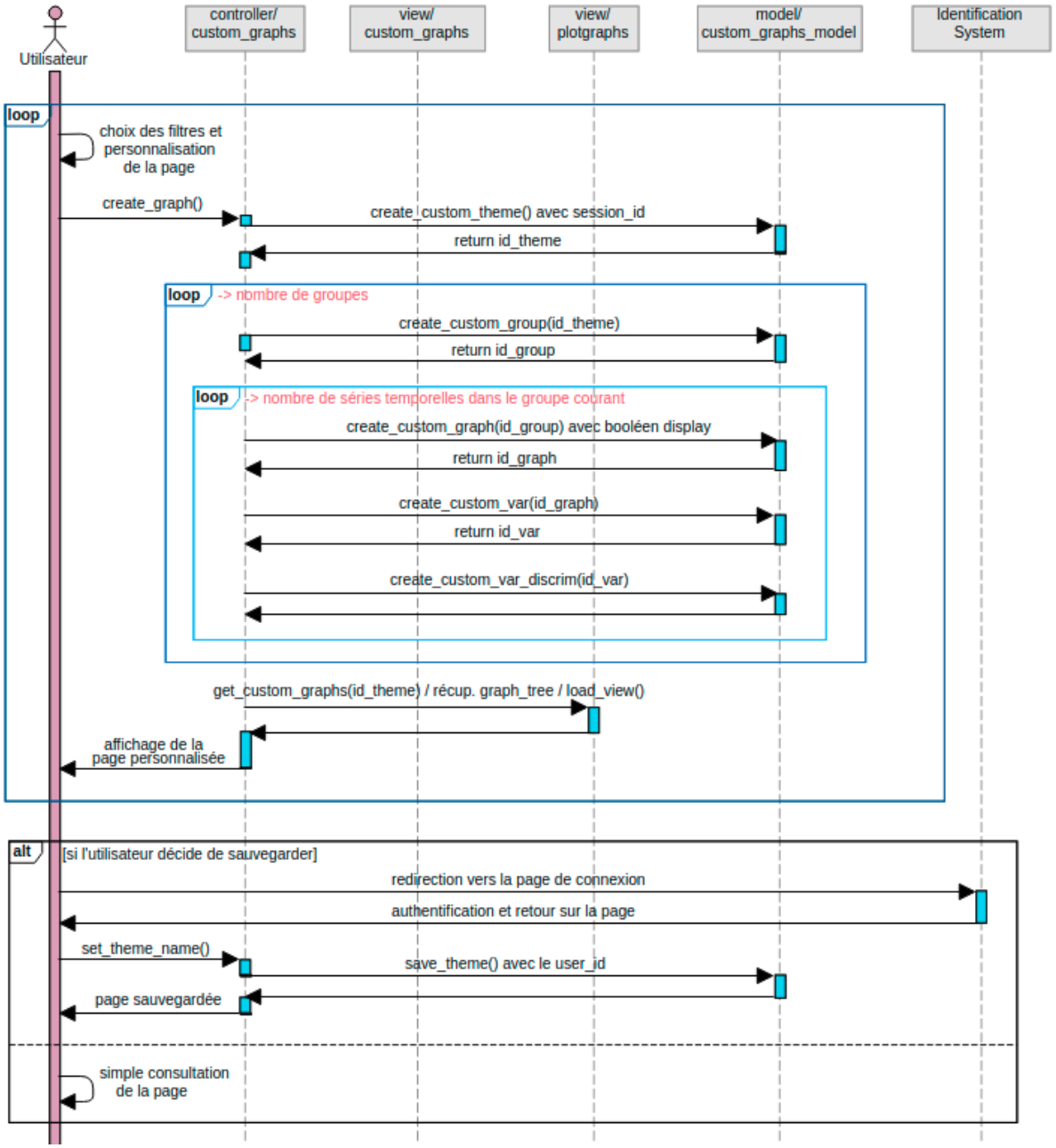

Figure 10 : Diagramme de séquence de la création et de la sauvegarde de la page d'un utilisateur non connecté

Par la suite, pour ce qui est du déroulement des actions, on choisit un utilisateur non connecté. Il commence la personnalisation de sa page en filtrant les séries temporelles disponibles en entrant des mots-clés dans l'espace prévu à cet effet. Il peut alors ajouter des séries et des onglets afin de personnaliser la page comme il le souhaite.

Une fois cette étape terminée et que le visiteur décide d'afficher la page, il va lancer le processus de création d'un nouveau thème. Pour cela, la fonction *"create_graphs"* du contrôleur est appelée. Elle va ensuite communiquer avec le modèle qui va permettre d'insérer toutes les informations de la page dans la base de données.

Pour commencer, c'est le thème qui est créé grâce à la fonction "*create_custom_theme"* avec différentes informations telles que le schéma de base de données associé et l'identifiant de session afin de lier la session de l'utilisateur et le thème.

Ensuite, ce sont les groupes, correspondant aux onglets, qui sont créés, pour cela, le contrôleur appelle la fonction "*create_custom_group*" (qui prend en paramètre l'identifiant du nouveau thème précédemment créé) autant de fois que la page possède de groupes.

Cette technique est de même utilisée pour la création des graphiques grâce à une boucle sur le nombre de séries sélectionnées. Ces séries correspondent à des graphiques qui vont être créés via la fonction "*create_custom_graphs*". La fonction prend en paramètre l'identifiant du groupe dans lequel il se trouve. Pour chacun des graphiques insérés, l'information sur son activation ou désactivation est passée via le booléen "*display*" permettant de tracer ou non le graphique.

Enfin, à la fin de la boucle, les informations sur la variable du graphique et le discriminant sont insérés dans la base de données via les fonctions respectives "*create_custom_var*" et "*create_custom_var_discrim*".

Une fois l'arborescence entièrement créée, le contrôleur va se charger de créer le graph_tree grâce à la fonction "*get_graph_tree_from_id*" du modèle *graph_model* et de faire appel à la vue *plotgraphs,* lui envoyant toutes les informations requises, et lui permettant de tracer tous les graphiques. Cette vue est ensuite insérée dans la page de l'utilisateur qui a maintenant accès aux graphiques qu'il a choisis.

L'utilisateur a deux choix, soit il continue de personnaliser sa page ou de la consulter, soit il décide de la sauvegarder pour pouvoir y accéder lors de ses prochaines connexions. S'il décide de sauvegarder, étant non connecté, il est redirigé vers la page d'authentification. Il doit ensuite donner un nom à la page ce qui entraîne la sauvegarde définitive de celle-ci avec son identifiant de compte.

Tous ces diagrammes réalisés lors de la conception ont permis de mieux visualiser le projet, que ce soit sur son concept ou sur ses fonctionnalités et de pouvoir être plus efficace lors de la réalisation de celui-ci.

2.2 Réalisation

2.2.1 Méthodologie utilisée

Pour la réalisation du projet, il a été décidé de commencer à implémenter l'interface pour la gestion des graphiques d'un Service d'Observation spécifique, nommé SNOKARST. L'objectif étant de rendre l'interface de moins en moins dépendante de ce SO, afin qu'elle devienne complètement générique et puisse être compatible avec tous les SO ou TO.

La stratégie a été d'adapter le code générique de la création des graphiques (*cf Partie Analyse de l'existant*), afin d'implémenter les fonctionnalités propres au projet de personnalisation de graphiques. Pour cela, différentes parties ont été reprises et adaptées telles que le modèle pour la création du graph_tree, la vue générant le code permettant de tracer les différents graphiques, ainsi que le contrôleur qui permet de faire le lien entre la vue et le modèle.

Par ailleurs, il a fallu à plusieurs reprises apporter des modifications sur les tables existantes dans la base de données dans le but d'apporter des informations supplémentaires.

2.2.2 Affichage des graphiques

Les informations utilisées dans la création des graphiques de l'interface sont récupérées dans la vue *timeserie_graph* de la base de données (*cf Figure 11*). Cette vue liste l'ensemble des séries temporelles pouvant être visualisées. Elle fournit les informations utilisées dans la structuration des graphiques (*thème, groupe, graphique etc.*).

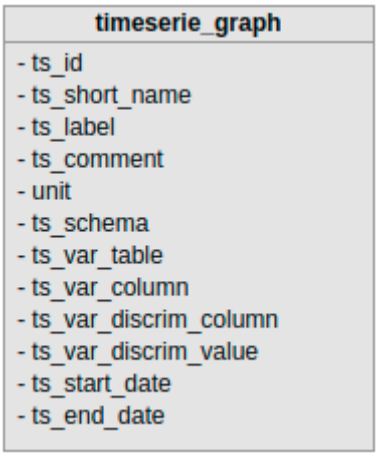

Figure 11 : Champs composant la vue timeserie_graph

Cette vue générique pourra être chargée si besoin dans chacun des schémas de la base de données.

Le tableau ci-après montre les correspondances entre les différents champs présents dans le modèle et dans la vue *timeserie_graph.*

On remarque que le nom du thème est un nom par défaut lorsque l'utilisateur n'est pas connecté. Ce nom est "temp" signifiant temporaire. Il sera modifié lorsque l'utilisateur sauvegardera sa page.

Par ailleurs, les champs nécessaires à la création des groupes sont récupérés via les choix que l'utilisateur a fait dans la personnalisation de sa page et plus précisément de ses onglets (leur nom, leur position etc.).

Enfin, on retrouve les mêmes champs *name* et *description* pour les graphiques et les variables. En effet, dans l'interface un graphique représente une seule série temporelle et donc une seule variable.

Par ailleurs, les champs nécessaires à la création des groupes sont récupérés via les choix que l'utilisateur a fait dans la personnalisation de sa page et plus précisément de ses onglets (leur nom, leur position etc.).

De plus, on retrouve les mêmes champs *name* et *description* pour les graphiques et les variables, en effet, dans l'interface un graphique représente une série temporelle et donc une seule variable.

	Champs du modèle	Champs de la vue
Thème	name schema	*"temp" pour temporaire* ts_schema
Groupe	name description position	*ces champs sont choisis par l'utilisateur*
Graphique	name description unit comment	ts_short_name ts_label unit ts_comment
Variable	name descritpion table column	ts_short_name ts_label ts_var_table ts_var_column
Discriminant	column value	ts_var_discrim_column ts_var_discrim_value

Tableau de correspondance des champs du modèle et ceux de la vue timeserie_graph pour la création des graphiques

2.2.3 Implémentation du système de filtres

Récupération des mots-clés dans la base de données

Afin de permettre une recherche de séries temporelles simple et rapide, un système de filtre par mot-clé a été mis en place. Pour cela, il a fallu adapter la base de données afin de définir une liste de mots-clés pour chaque série temporelle.

J'ai ainsi créé la vue "*keywords_view*" (*cf Figure 12*). Cette vue regroupe des informations présentes dans cinq tables différentes. Pour chaque série temporelle on retrouve : la catégorie et le nom du paramètre (tables *parameter* et *category)*, l'observatoire qui a effectué les mesures (table *organisation)*, le bassin et la station de mesure (tables *station* et *watershed*) permettant de situer géographiquement les observations faites. Le résultat de la vue se compose de deux colonnes : la première est l'identifiant de la série temporelle (*ts_id*) et la deuxième est le mot-clé qui la caractérise (*keyword*). On retrouve donc pour chaque série cinq lignes correspondant à chacun des filtres.

Cette vue générique pourra être créée si besoin dans chacun des schémas de la base de données.

```
CREATE OR REPLACE VIEW snokarst_new.keywords_view AS
 SELECT kw.ts_id,
    kw.keyword
   FROM ( SELECT ts.ts_id,
            (o.org_name::text || ' (obs)'::text)::character varying AS keyword
           FROM snokarst_new.timeserie ts
             JOIN snokarst_new.station s ON ts.station_id = s.station_id
             JOIN snokarst_new.organisation o ON s.obs_id::text = o.org_id::text
        UNION
         SELECT ts.ts_id,
            s.station_name AS keyword
           FROM snokarst_new.timeserie ts
             JOIN snokarst_new.station s ON ts.station_id = s.station_id
        UNION
         SELECT ts.ts_id,
            p.par_name_en AS keyword
           FROM snokarst_new.timeserie ts
             JOIN snokarst_new.parameter p ON ts.par_id = p.par_id
        UNION
         SELECT ts.ts_id,
            c.cat_name AS keyword
           FROM snokarst_new.timeserie ts
             JOIN snokarst_new.parameter p ON ts.par_id = p.par_id
             JOIN snokarst_new.category c ON p.cat_id = c.cat_id
        UNION
         SELECT ts.ts_id,
            (w.watershed_name::text || ' (bassin)'::text)::character varying AS keyword
           FROM snokarst_new.timeserie ts
             JOIN snokarst_new.station s ON ts.station_id = s.station_id
             JOIN snokarst_new.watershed w ON s.watershed_id::text = w.watershed_id::text) kw
  ORDER BY kw.ts_id;
```

Figure 12 : Code de la vue pour les mots-clés des séries temporelles

Insertion et choix des mots-clés en HTML

La liste des mots-clés se construit grâce à la fonction PHP "*get_ts_graph_info*" (*cf Figure 13*). Cette fonction récupère la liste des séries temporelles disponibles, avec, pour chacune d'entre elles, les mots-clés associés, concaténés par un point-virgule. C'est sous cette forme que sont insérés les mots-clés dans un attribut "keywords" de chaque élément HTML correspondant à une série temporelle :

<li class="ts_list ui-draggable" *keywords="keyword1;keyword2;keyword3;keyword4;keyword5"* ts_id="identifiant de la série">Nom de la série temporelle (unité)

```
/**
 * Returns all ts with values
 * @return mixed[]
 */
public function get_ts_graph_info($schema)
{
  $sql = "SELECT timeserie_graphs.ts_id, timeserie_graphs.ts_label,
  timeserie_graphs.unit, string_agg(keywords_view.keyword, ';') AS keywords
  FROM $schema.timeserie_graphs
  JOIN $schema.keywords_view ON timeserie_graphs.ts_id = keywords_view.ts_id
  GROUP BY timeserie_graphs.ts_id, timeserie_graphs.ts_label, timeserie_graphs.unit
  ORDER BY ts_label";

  $query = $this->db->query($sql);
  return $query->result_array();
}
```

Figure 13 : Fonction pour trouver toutes les informations des séries temporelles

Sur la page de personnalisation, un élément de type 'input' permet à l'utilisateur de saisir une liste de mots-clés. Le système est basé sur une méthode de saisie textuelle par auto-complétion tirée de l'extension nommée '*jQuery UI Autocomplete MultiSelect Plugin*' de la librairie jQuery UI. Cette extension est basée sur la fonction '*autocomplete*' de cette même librairie, mais modifie l'événement 'select' de la fonction initiale afin que, lors de la sélection d'un mot-clé, celui-ci se mette sous la forme d'étiquette.

Ainsi, lorsque l'utilisateur saisit une lettre, la liste des mots-clés correspondants lui est proposée. A la sélection d'un mot-clé, un élément HTML se crée, composé du mot choisi ainsi que d'une croix afin que l'utilisateur ait la possibilité de supprimer le filtre.

Filtrage des séries temporelles

Une fois les filtres choisis, on cherche à récupérer la liste des séries temporelles (et plus particulièrement leur identifiant) pour lesquelles les mots-clés présents dans leur attribut '*keywords*' en HTML correspondent à la sélection des mots-clés faite par l'utilisateur.

Pour cela, j'ai implémenté la fonction JavaScript "*availableTs*" afin de récupérer cette liste de séries sous forme de tableau (*cf Annexe C*). Dans un premier temps, tous les mots-clés sélectionnés par l'utilisateur sont récupérés. Ensuite, une chaîne de caractères est créée à partir du tableau afin de définir un sélecteur jquery avec une intersection de conditions sous la forme :

$('[keyword="keyword1"][keyword*="keyword2"] ... ')*

L'opérateur * signifie que l'attribut contient la chaîne de caractères mis en paramètre et il permet ici de trouver tous les éléments dont l'attribut *keyword* contient chacun des filtres choisis par l'utilisateur.

Une fois le résultat trouvé, les séries correspondantes sont affichées dans le panneau latéral. Chacune d'entre elles possède un style CSS comportant des informations sur leur position, leur couleur etc., mais c'est grâce à l'attribut "display", qui permet d'afficher ou de cacher un élément, qu'est géré l'affichage ou non des séries temporelles dans le panneau.

2.2.4 La gestion des séries temporelles

Une fois la sélection des filtres effectuée, l'utilisateur a la possibilité de sélectionner et d'ajouter les séries temporelles dans la structure de la page de graphiques. Pour cela, il doit utiliser les méthodes "*drag and drop*" de la librairie jQuery UI. Cette méthode est implémentée en jQuery grâce aux fonctions nommées *draggable* et *droppable*. Les éléments HTML correspondant à chaque série temporelle sont *draggable*, c'est-à-dire déplaçables, et les divisions contenues dans chaque onglet sont *droppable*, c'est-à-dire que l'on peut déposer des éléments à l'intérieur.

Par la suite et au fur et à mesure de l'expérimentation de l'interface, il a été décidé que l'ajout des séries temporelles se ferait aussi au double-clic sur l'élément de la série ciblée, ce qui facilite la personnalisation lorsque la liste des séries est trop longue et que la structure de la page de graphiques n'est plus visible.

Lorsque l'utilisateur dépose ou double-clique sur une série, un élément HTML est créé comportant plusieurs divisions HTML et icônes. On trouve le nom et l'unité de la série temporelle déplacée, l'icône du choix d'affichage du graphique, l'icône de suppression et l'icône de déplacement de la série afin de l'ordonner avec les autres séries. Cependant, la division principale ne contient qu'un message car aucune donnée n'a encore été chargée. J'ai fait en sorte que la structure de cet élément soit semblable à celle des graphiques existant dans le site.

Pour choisir de tracer le graphique ou bien de le désactiver, l'utilisateur doit cliquer sur l'icône ressemblant à une case à cocher. La gestion de l'affichage se fait via un booléen (attribut *display*) inséré dans les attributs HTML de chaque graphique de série temporelle. Par défaut, la valeur est à *true*, c'est-à-dire que toute série déposée dans la partie de droite est tracée. Cependant, si l'utilisateur choisit de ne pas tracer les données des graphiques, alors le booléen passe à false lorsque l'utilisateur clique sur l'icône, ce qui aura pour effet visuel de modifier l'icône "cochée" par l'icône "non cochée". Cette information sera récupérée lors de la validation par l'utilisateur et de l'envoi du formulaire au contrôleur (*cf "La création des pages temporaires"*).

Pour supprimer une série temporelle déposée, l'utilisateur doit cliquer sur l'icône croix. Cette action est gérée via un événement Javascript 'onclick' sur cette icône, entraînant l'appel de la fonction correspondante, nommée '*deleteGraph*' (*cf Annexe C*).

Il y alors deux possibilités :

- Soit les mots-clés précédemment saisis correspondent à la série temporelle à supprimer, dans ce cas la série se remet à sa place initiale dans le panneau latéral. Pour cela, c'est son style CSS qui revient comme initialement implémenté (*cf Annexe C*) ;
- Soit la série supprimée ne correspond pas aux mots-clés saisis, dans ce cas la série n'est pas affichée dans le panneau latéral.

Pour ordonner les séries entre elles, l'utilisateur doit maintenir cliquée la section la plus haute de l'élément contenant les graphiques (celle qui contient le nom de la série) ou bien maintenir l'icône de déplacement. Ce classement est rendu possible et automatiquement géré par la fonctionnalité '*sortable*' de la librairie jQuery UI.

2.2.5 La gestion des onglets

L'interface offre la possibilité aux utilisateurs de créer plusieurs onglets/groupes de graphiques. La gestion des onglets ressemble aux méthodes utilisées pour les onglets d'un navigateur web. On retrouve des fonctionnalités similaires telles que l'ajout et la suppression mais aussi des différences comme l'édition du nom de l'onglet. Chacune des actions est déclenchée à partir d'événements ('onclick', 'ondblclick'), gérés en Javascript.

L'icône "+" permet de créer un nouvel onglet. Pour cela, il suffit de cliquer sur celle-ci, ce qui va appeler la fonction *addTab()*. Cette fonction va créer plusieurs éléments. Dans une première partie l'onglet (*cf Annexe C*), puis la section qui va contenir les séries temporelles par la suite. C'est à cet instant que tous les événements liés à la gestion des onglets sont ajoutés (événement "onclick", "ondblclick", etc.).

Pour renommer le groupe, il faut double-cliquer sur l'onglet souhaité. Cette action va insérer un élément de type 'input' permettant à l'utilisateur de rentrer un nouveau nom. Pour valider, il suffit d'appuyer sur la touche 'Entrée' ou de cliquer ailleurs dans la page ce qui va remplacer l'input par l'onglet avec le nouveau nom donné (*cf Annexe C*).

La croix permet quant à elle de supprimer un onglet et de replacer les séries contenues dans le groupe dans le panneau latéral grâce à la fonction *deleteTab*() faisant appel à la fonction *deleteGraph*() (*cf Annexe C*) si l'onglet contient des graphiques.

2.2.6 La création des pages/thèmes temporaires

Une fois que l'utilisateur a fini de modifier la page, pour pouvoir tracer les graphiques qu'il a choisis, il doit cliquer sur le bouton 'Afficher'. Cette action entraîne l'envoi via un appel AJAX d'un certain nombre de données.

On retrouve :

- Le nom du schéma ;
- Un tableau JavaScript contenant les noms des onglets / groupes ;
- Un tableau deux dimensions comprenant pour chaque onglet / groupe, les identifiants des séries temporelles (nommé *ts_id*) dont il est composé ;
- Les dates 'from' et 'to' indiquant les dates de début et de fin choisis par l'utilisateur (facultatif)
- Un tableau de tous les identifiants des séries temporelles composant la page ;
- Un tableau regroupant les booléens indiquant si la série doit être tracée ou non (booléen 'display').

Pour ce qui est de l'information sur les positions des groupes et des graphiques dans la page, elle est donnée grâce à l'insertion dans l'ordre d'apparition dans le tableau à deux dimensions qui est cité plus haut.

Ces informations sont ensuite passées à la méthode *create_graphs* du contrôleur *custom_graphs*. Cette fonction se charge de créer en base de données (*cf la partie 2.1.2 Analyse des communications*) le thème personnalisé que l'utilisateur vient de définir ainsi que de charger la vue permettant l'affichage de la page de graphiques.

Pour lier le thème à la session j'ai modifié la table *graph_themes* dans la base de données en ajoutant une nouvelle colonne nommée *session_id* contenant l'identifiant de la session associée, et une contrainte de clé étrangère sur l'attribut *session_id* de la table *cizacl_session* avec ON DELETE CASCADE. Ainsi, lorsque la session de l'utilisateur se termine et que la session est supprimée de la table, tous les thèmes liés à celle-ci sont automatiquement supprimés. Cela garantit que les informations créées en base pour les thèmes temporaires créés dans le cadre des sessions seront automatiquement détruits à la fin des sessions (lorsque le framework supprimera en base l'identifiant des sessions).

C'est lors de la création d'un thème, dans la fonction "*create_custom_theme*" que l'identifiant de session courante est récupéré grâce à la fonction CodeIgniter nommée "*user_data(session_id)*" puis inséré comme attribut du thème. De même, lors de la création de chaque graphique, l'information concernant son activation ou sa désactivation (avec le booléen *show*) est récupérée et insérée dans la base de donnée (via la fonction *create_custom_graph* du modèle), dans l'attribut *display* de la table *graph_graphs.*

2.2.7 La sauvegarde de la page personnalisée

Afin de sauvegarder sa page personnalisée et y avoir accès lors de ses futures visites, l'utilisateur doit cliquer sur le bouton 'Sauvegarder la page' présent en haut de l'écran. Ce bouton n'apparaît qu'après la première personnalisation de l'utilisateur, c'est-à-dire après la création de son premier thème.

La sauvegarde est gérée par un événement *'onclick'* sur le bouton qui va se charger de la gestion des différents cas, grâce à un appel ajax vers la fonction *set_theme_name()* du contrôleur. La fonction va tester si l'utilisateur est connecté en vérifiant s'il possède un identifiant (nommé *user_id*).

Si l'utilisateur n'est pas connecté, il est redirigé vers la page de connexion et devra alors cliquer de nouveau sur le bouton *sauvegarder* une fois connecté.

Si l'utilisateur est connecté, une fenêtre s'ouvre, composée de différents 'input' qui vont offrir différentes possibilités à l'utilisateur :

- Soit l'utilisateur choisit de créer un nouveau thème, et doit alors donner un nom à celui-ci avant de l'enregistrer ;
- Soit l'utilisateur possède déjà des pages personnalisées, dans ce cas il peut décider d'écraser un thème existant.

Cette fonctionnalité est en cours d'implémentation, c'est pourquoi seule la première possibilité peut être effectuée pour l'instant.

3 Résultats

3.1 Tests

Durant la phase de développement de l'interface, de nombreuses phases de tests ont été réalisées. Aucun test n'a été automatisé pendant la durée du projet, c'est pourquoi chaque fonctionnalité du site a été testée manuellement lors de son implémentation.

Avant l'implémentation et l'intégration de chacune des fonctionnalités sur le site de production, une phase de réflexion était menée afin de trouver un maximum de failles et de cas pouvant bloquer ou bien faire dysfonctionner l'interface. C'est de cette manière, en se mettant dans la peau d'un utilisateur, que des tests fonctionnels et d'intégration ont été réalisés.

De plus, des tests unitaires ont été effectués sur chaque fonction, permettant de vérifier leur fonctionnement et de trouver tous les cas qui déclencheraient des erreurs.

Cependant, tous les tests n'ont pas encore pu être réalisés. En effet, les tests fonctionnels et d'affichage sur les différents navigateurs Firefox et Opéra sont en perspective de réalisation tandis qu'ils ont été réalisés sur le navigateur Chrome.

Pour ce qui est de la gestion des bugs rencontrés, les outils de développement des navigateurs m'ont permis, grâce à l'utilisation des "breakpoints" (ou points d'arrêt en français), de pouvoir suivre, ligne par ligne, l'évolution du code.

Ces phases de tests et de débuggage ont été très importantes dans le déroulement du développement car elles ont permis de réaliser des tests tout au long de l'implémentation de l'interface, permettant d'enlever une charge de travail lors des tests finaux.

Par ailleurs, un rapide retour utilisateur a été fait lors d'une visite de deux chercheurs, futurs utilisateurs de l'interface. Cette expérience a permis de réellement voir l'objectif qu'ont les scientifiques en utilisant l'outil développé mais aussi de tester l'interface dans de nouvelles conditions et d'avoir un retour sur la pertinence des éléments et des fonctionnalités implémentées.

3.2 Manuel d'utilisation

Dans cette partie, un guide des actions réalisables va être détaillé, permettant aux utilisateurs de l'interface de savoir comment utiliser les outils mis à leur disposition.

3.2.1 Utilisation des filtres

Pour commencer, la première action à réaliser pour personnaliser une page est de saisir un mot-clé dans l'espace prévu à cet effet dans le panneau latéral qui se trouve à gauche de la page. Il n'est pas obligatoire de rentrer un mot en entier, une lettre suffit à proposer une liste de différents mots-clés contenant cette lettre (*cf Figure 14*).

Figure 14 : Système de saisie d'un filtre

Une fois que le mot est choisi dans la liste, une étiquette s'affiche et une liste de séries temporelles correspondant au(x) mot(s)-clé(s) apparaît sous l'espace de saisie des filtres (*cf Figure 15*).

L'étiquette contient le nom du filtre ainsi qu'une croix. Lorsqu'un mot est sélectionné, la liste des mots-clés disponibles est filtrée afin de pouvoir affiner la recherche des séries temporelles.

Pour supprimer un filtre, il suffit de cliquer sur la croix de l'étiquette correspondante. La liste des mots-clés disponibles s'adapte alors à la sélection faite.

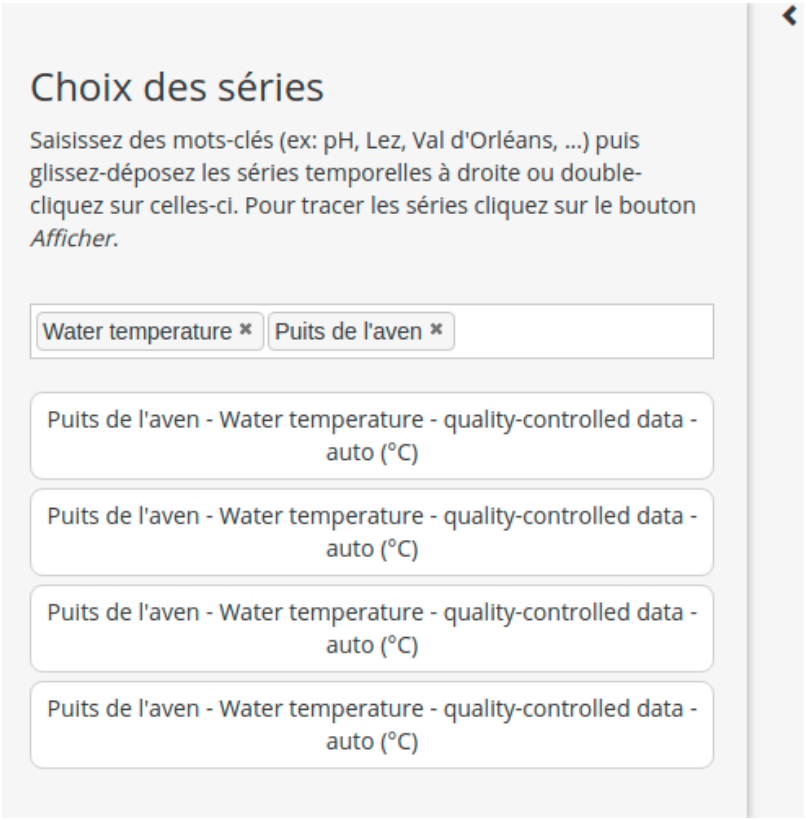

Figure 15 : Système de filtre sur la page

La page offre aussi la possibilité de fermer et de ré-ouvrir le panneau latéral grâce à l'icône présente en haut à droite de celui-ci (visible sur les figures précédentes). Cette fonctionnalité permet de modifier la largeur de la page de personnalisation afin d'avoir un meilleur confort visuel lorsque les graphiques sont tracés.

3.2.2 Gestion des séries temporelles

Une fois les filtres choisis, il suffit d'ajouter les séries temporelles voulues dans la page de personnalisation. Pour cela, il y a deux possibilités :

- Utiliser le "*drag and drop*"* pour déposer les séries directement dans la partie personnalisation ;
- Utiliser le double-clique directement sur les séries.

Lorsqu'une série temporelle est déposée, un élément apparaît composé du nom et de l'unité de celle-ci, ainsi que de trois icônes (*cf Figure 16*).

L'icône de gauche permet de choisir si le graphique doit être tracé (activé par défaut) ou non (désactivé). La prochaine icône permet de supprimer l'élément et par extension le graphique associé lorsque celui-ci est tracé. Pour finir, la dernière icône permet d'ordonner les graphiques des séries temporelles entre elles lorsqu'elles sont présentes dans la page de personnalisation qui est représentée en pointillé gris.

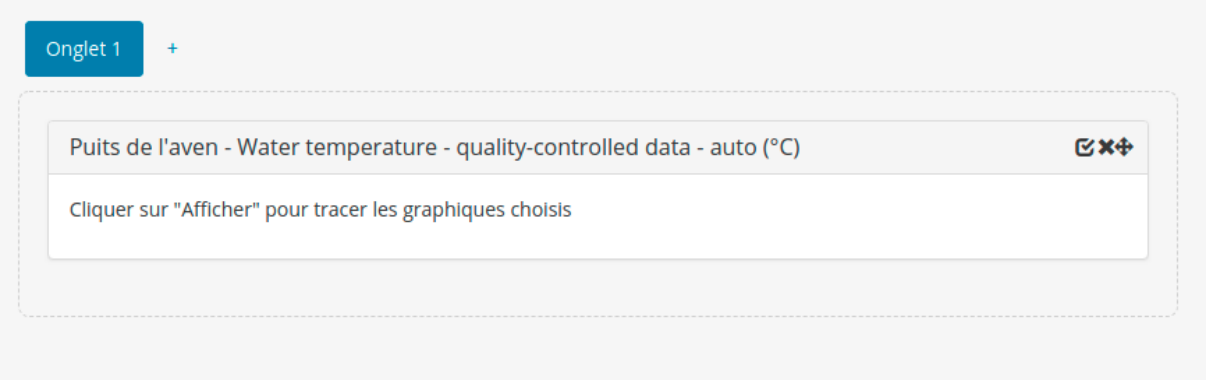

Figure 16 : Élément contenant un graphique non tracé

Une fois les graphiques tracés, le mécanisme reste le même que précédemment énoncé. En effet, on retrouve les mêmes éléments ce qui permet une gestion similaire (*cf Figure 17*).

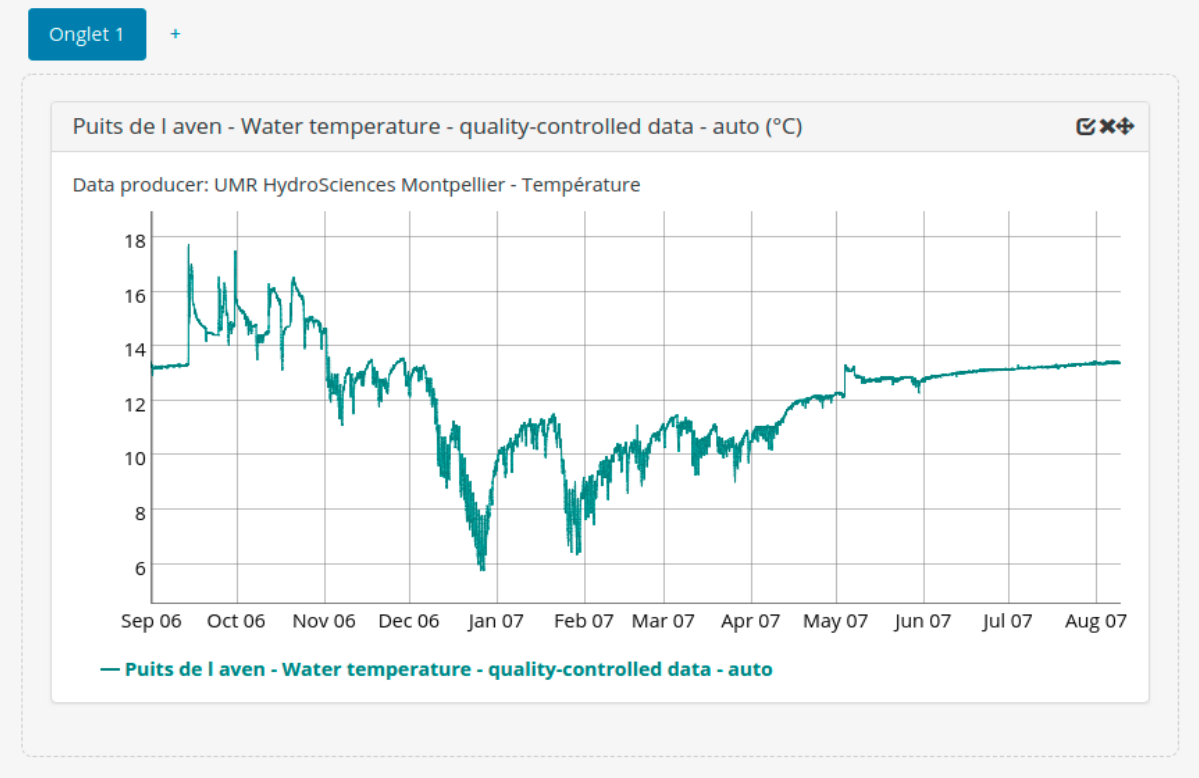

Figure 17 : Élément contenant un graphique tracé

Ce mécanisme reste inchangé si l'on décide de ne pas tracer / activer un graphique de série temporelle (*cf Figure 18*). La seule différence est que l'icône de gauche est différente car le graphique est désactivé. Pour le réactiver il suffit de cliquer sur cette icône.

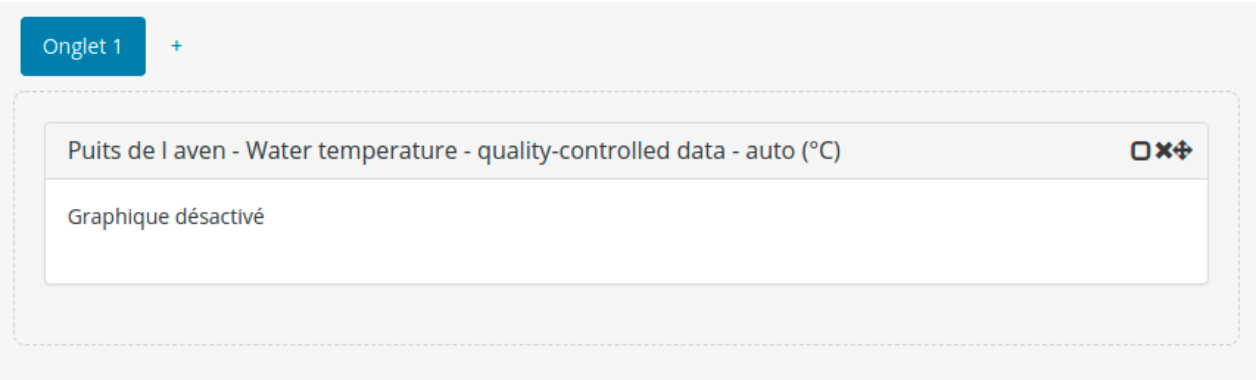

Figure 18 : Élément contenant un graphique non activé

Lorsqu'une série est supprimée, elle n'est pas forcément réinsérée dans le panneau latéral. En effet, pour cela il faut que les mots-clés sélectionnés correspondent à ceux de la série. Si ce n'est pas le cas, alors la série n'est pas visible dans le panneau, il suffit alors de sélectionner les filtres qui lui correspondent pour la retrouver.

3.2.3 Gestion des onglets

Lors du premier chargement de la page, elle n'est composée que d'un seul onglet, nommé 'Onglet 1' (visible sur les figures précédentes).

Cependant, il est possible d'ajouter des onglets en cliquant sur l'icône '+', permettant de créer un nouvel onglet ainsi que sa division associée (*cf Figure 19*).

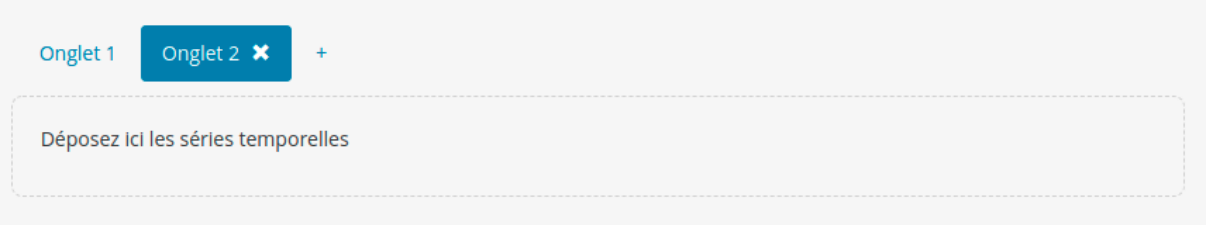

Figure 19 : Création d'un nouvel onglet

Sur ce nouvel onglet on retrouve une icône permettant de supprimer l'onglet ainsi que sa division, même lorsque celle-ci possède des graphiques. En effet, la fermeture d'un onglet entraîne automatiquement la suppression des graphiques qu'il contient.

Chaque onglet peut être renommé : il suffit pour cela de double-cliquer sur l'icône afin de faire apparaître un élément qui permet de rentrer un nouveau nom (*cf Figure 20*). Pour le valider, il faut appuyer sur la touche 'Entrée' du clavier ou cliquer ailleurs sur la page. Cette action peut être répétée autant de fois que souhaitée.

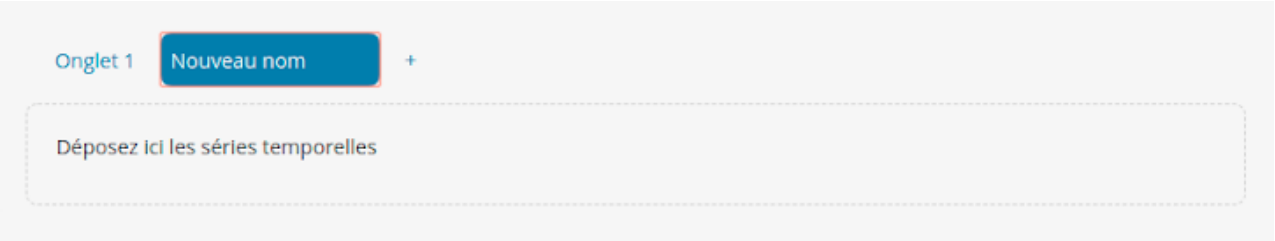

Figure 20 : Édition du nom d'un onglet

3.2.3 Création de la page personnalisée

Pour tracer les graphiques une fois la personnalisation de la page terminée, il convient de cliquer sur le bouton 'Afficher'. Cette action va déclencher la création de la page, ce qui peut prendre un certain temps en fonction du nombre de graphiques à tracer.

Après la création de la page personnalisée, il est toujours possible de continuer à la personnaliser, en ajoutant / supprimant des graphiques et des onglets. Afin de voir le résultat des modifications apportées, il suffit de cliquer sur le bouton 'Mettre à jour' (*cf Figure 19*).

Il est important de cliquer sur ce bouton à chaque modification sur la page et avant de vouloir sauvegarder celle-ci.

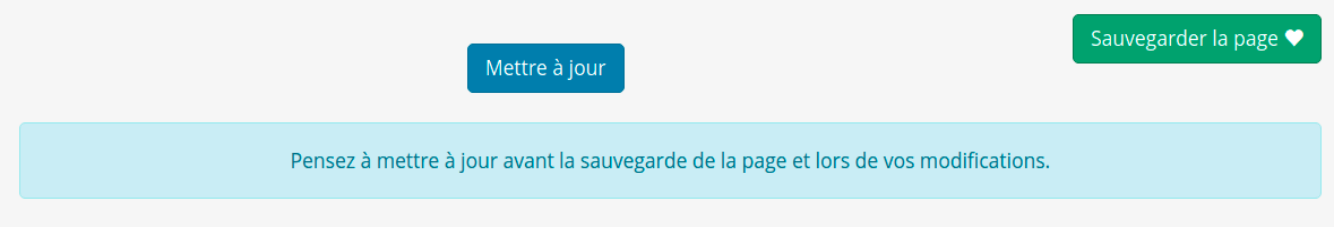

Figure 21 : Mise à jour et sauvegarde de la page

Par ailleurs, l'interface propose des fonctionnalités en commun avec les pages de graphiques existantes sur le site. On retrouve la possibilité de changer l'intervalle de temps grâce aux champs de dates présents en haut de l'écran. Il suffit de cliquer sur le champ pour ouvrir un calendrier, permettant de choisir une date précise. De plus, pour certains graphiques, on retrouve la possibilité d'exporter les données d'observation au format CSV*. Les champs d'export des données se trouvent en bas de l'écran et il suffit de cliquer sur le bouton 'Télécharger' pour charger les fichiers contenant les mesures.

Il est important de noter que lors de la création d'une page, celle-ci est temporaire et se supprime automatiquement au bout de quelques heures. C'est ce thème temporaire qui est affiché lors de chaque chargement de l'interface de personnalisation.

Cependant, il est possible de la sauvegarder afin de ne pas perdre les modifications apportées.

3.2.4 Sauvegarde de la page personnalisée

Lorsque la personnalisation est terminée, l'interface offre la possibilité de sauvegarder la page afin qu'elle soit accessible lors des prochaines connexions grâce au bouton 'Sauvegarder la page' (*cf Figure 21*).

Tout d'abord, si l'utilisateur n'est pas connecté, il va être redirigé vers la page de connexion afin qu'il puisse se connecter et pouvoir sauvegarder sa page.

Une fois connecté, lors du clic sur le bouton 'Sauvegarder la page', une fenêtre apparaît afin d'offrir plusieurs possibilités (*cf Figure 22).*

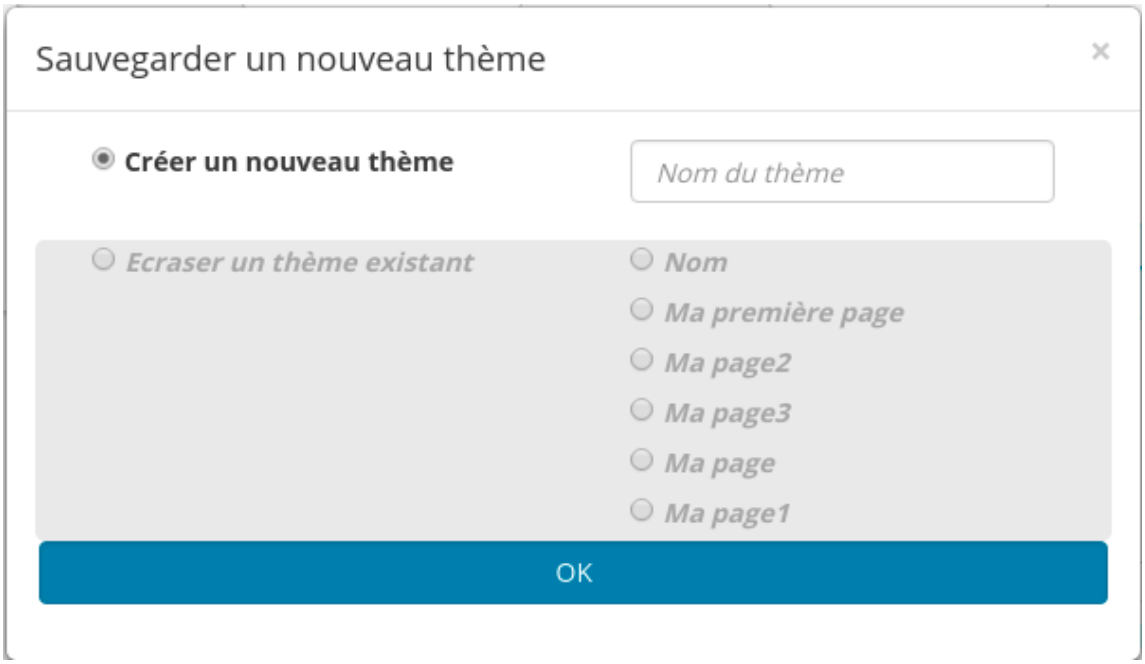

Figure 22 : Création d'un nouveau thème

Soit l'utilisateur ne possède actuellement aucune page personnalisée, auquel cas il ne peut que créer un nouveau thème en lui donnant un nom et cliquer sur le bouton 'OK' pour valider.

Soit l'utilisateur possède des pages personnalisées, dans ce cas il a le choix entre créer un nouveau thème ou écraser un thème existant. S'il choisit cette possibilité il suffit de choisir quel thème écraser et cliquer sur 'OK' pour confirmer.

4 Rapport d'activités

4.1 Planification et répartition

Au début du stage, n'ayant jamais travaillé sur de framework et de librairie, aucun plan de travail précis n'a été établi ; mais étant seule sur le projet, j'ai eu la liberté de m'organiser au fur et à mesure de l'avancement du travail.

De plus, plusieurs fonctionnalités ont été ajoutées au cours du stage, ce qui rendait difficile la conception d'une organisation prévisionnelle. Il m'était compliqué de prévoir le temps qu'allaient me prendre la conception et le développement de chaque fonctionnalité. J'ai donc choisi une planification journalière.

Chaque fin de journée, une 'todo list' été créée, récapitulant les fonctionnalités à implémenter, objectifs à réaliser et problèmes à résoudre pour les jours suivants. Tout au long du projet, j'ai pris des notes sur mes avancées me permettant de savoir où en était la progression du projet, notant ce qui avait été réalisé et ce qui restait à faire.

Régulièrement, des réunions étaient organisées afin que mes tuteurs voient mon avancée sur le projet. Ces réunions permettaient de faire un point sur ce qui avait été réalisé et ce qu'il fallait revoir, ainsi que de planifier les prochaines fonctionnalités à implémenter.

4.2 Logiciels utilisés durant le projet

4.2.1 L'environnement Apache2

Apache2 est un serveur HTTP (dans le domaine du web) permettant de travailler sur un serveur différent de celui de production du site. Il est donc utilisé comme serveur de développement : c'est là où les nouvelles fonctionnalités sont codées avant d'être envoyées sur le site en ligne. Il respecte le protocole "client-serveur" permettant des communications entre le serveur web et le navigateur.

4.2.2 Apache Subversion (SVN)

Apache Subversion est un logiciel de gestion de versions (versioning) permettant de stocker et de fusionner les différentes modifications effectuées sur le site. Son but est de gérer les différentes versions du site (qu'on appelle branches) afin de pouvoir effectuer des mises à jour ou bien de pouvoir revenir à une version inférieure du site.

Cet outil permet donc de suivre l'évolution du site, de son commencement à son état actuel avec l'objectif de conserver la chronologie de toutes les modifications.

De plus, il peut créer une copie de développement du site, pendant que le vrai reste fonctionnel en ligne. On peut donc se permettre de tester, de modifier, et d'ajouter du code, sans que le site de production ne soit impacté par le développement

4.2.3 PostgreSQL et PgAdmin III

PostgreSQL est un système de gestion de base de données relationnelle et objet. Il permet de définir, manipuler, récupérer et gérer les données stockées. Il extrait des informations de la base de données sur demande de l'utilisateur, en fonction des requêtes effectuées.

PgAdmin est un outil graphique multi-plateforme de gestion de bases de données PostgreSQL. Il permet d'administrer plusieurs bases et d'aider l'utilisateur dans l'écriture des requêtes SQL.

4.2.4 Logiciels de traitement de texte

Pour le développement de l'interface j'ai utilisé des éditeurs de texte adaptés tels que Sublime Text et Atom qui m'ont aidé à implémenter chaque fonctionnalité.

4.3 Difficultés rencontrées

Tout au long du projet, j'ai rencontré différents problèmes portant sur plusieurs domaines. Tout d'abord, les premières difficultés rencontrées ont été de s'adapter aux technologies utilisées dans le cadre du projet. J'ai découvert de nouveaux outils de programmation comme le framework CodeIgniter, la gestion de base de données en PostgreSQL et les librairies JavaScript jQuery et Dygraphs. En effet, dans le cadre de l'IUT nous n'avons jamais utilisé de framework et la programmation en JavaScript se fait sans librairie. De plus, l'apprentissage de la gestion des bases de données se fait en MySQL. C'est pourquoi il a fallu un temps d'adaptation et de découverte de ces technologies au début du stage. Pour cela, j'ai consulté de nombreux sites de documentation et j'ai beaucoup analysé le site existant utilisant chacune de ses technologies afin de comprendre leur utilisation et de pouvoir me les approprier afin de commencer le projet.

Par ailleurs, j'ai rencontré des problèmes liés à l'intégration de l'interface au site existant. En effet, plusieurs des fonctionnalités devaient être reprises et adaptées au nouveau système entraînant des bugs.

Conclusion

Le but de ce projet était d'offrir la possibilité à chaque utilisateur de définir et de sauvegarder une configuration de graphiques "à la carte" sur le site de données de l'OSU OREME. L'interface à implémenter devait être intégrée au site existant, entraînant des contraintes sur les technologies à utiliser et sur la charte graphique à respecter. L'objectif était de permettre une personnalisation complète des graphiques à afficher pour chaque utilisateur, qu'il soit scientifique ou bien simple visiteur, mais aussi de permettre une meilleure maintenance pour les administrateurs. Pour cela, j'ai conçu et développé un outil permettant de choisir parmi des centaines de séries temporelles liées à un Service d'Observation, et qui permet de tracer chacun des graphiques, grâce à la librairie Dygraphs.

Les fonctionnalités principales du projet ont été implémentées, mais des perspectives d'améliorations et d'ajout de fonctionnalités sont envisagées. Par exemple, lors de la sauvegarde d'un thème, l'utilisateur pourra écraser un de ses thèmes existants. Il aura aussi accès à ses différents thèmes enregistrés via une page de préférences. De plus, chaque graphique pourra contenir plusieurs variables (au lieu d'une seule pour l'instant). L'utilisateur pourra de même déplacer des graphiques d'un onglet à un autre, tout en pouvant personnaliser des paramètres d'affichage des graphiques (minimum et maximum sur l'axe des ordonnées). Les administrateurs pourront se servir de l'interface pour créer de nouvelles pages de graphiques pour le site. Par ailleurs, plusieurs tests et améliorations du code devront être effectués avant la mise en production de l'interface sur le site.

Ce projet fut une expérience complète dans le domaine professionnel mais aussi sur le plan personnel. En effet, j'ai pu bénéficier d'une vraie expérience en programmation Web, côté serveur et côté client mais aussi du côté des bases de données, en utilisant des technologies qui m'étaient jusque-là encore inconnues. Durant ce stage, j'ai beaucoup appris sur le développement Php et Javascript, tout en mettant en application des connaissances et compétences apprises lors de ma formation. J'ai aussi beaucoup appris sur le travail en autonomie avec la prise de décision et la résolution des problèmes qui ont été rencontrés.

Par ailleurs, j'ai apprécié travailler sur un projet qui va vraiment être utilisé par plusieurs personnes.

Bibliographie

[1] OREME [En ligne]. Disponible sur: https://oreme.org/ [Consulté le: 02-avr-2019].

[2] Portail des données OREME [En ligne]. Disponible sur: https://data.oreme.org/ [Consulté le: 02-avr-2019].

[3] Dygraphs [En ligne]. Disponible sur: http://dygraphs.com/ [Consulté le: 03-avr-2019].

[4] jQuery API [En ligne]. Disponible sur: https://api.jquery.com/ [Consulté le: 05-avr-2019].

[5] CodeIgniter Documentation [En ligne]. Disponible sur: https://www.codeigniter.com/docs [Consulté le: 05-avr-2019].

[6] MDN Javascript [En ligne]. Disponible sur: https://developer.mozilla.org/fr/docs/Web/JavaScript [Consulté le: 08-avr-2019]

[7] MDN HTML [En ligne]. Disponible sur: https://developer.mozilla.org/fr/docs/Web/HTML [Consulté le: 11-avr-2019]

[8] PHP Documentation [En ligne]. Disponible sur: https://www.php.net/docs.php [Consulté le: 15-avr-2019]

[9] PostgreSQL Documentation [En ligne]. Disponible sur: https://www.postgresql.org/docs [Consulté le: 15-avr-2019]

[10] jQuery UI API [En ligne]. Disponible sur: https://api.jqueryui.com/ [Consulté le: 25-avr-2019]

[11] Bootstrap [En ligne]. Disponible sur: https://getbootstrap.com/docs/3.3/ [Consulté le: 25-avr-2019]

[12] Créateur de maquettes Pencil Project [En ligne] http://pencil.evolus.vn/Features.html [Consulté le: 29-avr-2019]

[13] Dynamisez vos sites web avec JavaScript ! [En ligne]. Disponible sur: https://openclassrooms.com/fr/courses/1916641-dynamisez-vos-sites-web-avec-javascript/1922434-le-drag-drop [Consulté le: 06-mai-2019]

[14] jQuery Autocomplete Multiselect Demo [En ligne]. Disponible sur: https://www.jqueryscript.net/demo/jQuery-jQuery-UI-Plugin-For-Simle-Tokenized-Autocomplete-Autocomplete-Multiselect/ [Consulté le: 20-mai-2019]

[15] SVN Documentations [En ligne]. Disponible sur: https://subversion.apache.org/docs/ [Consulté le: 22-mai-2019]

Annexe A

Organigramme

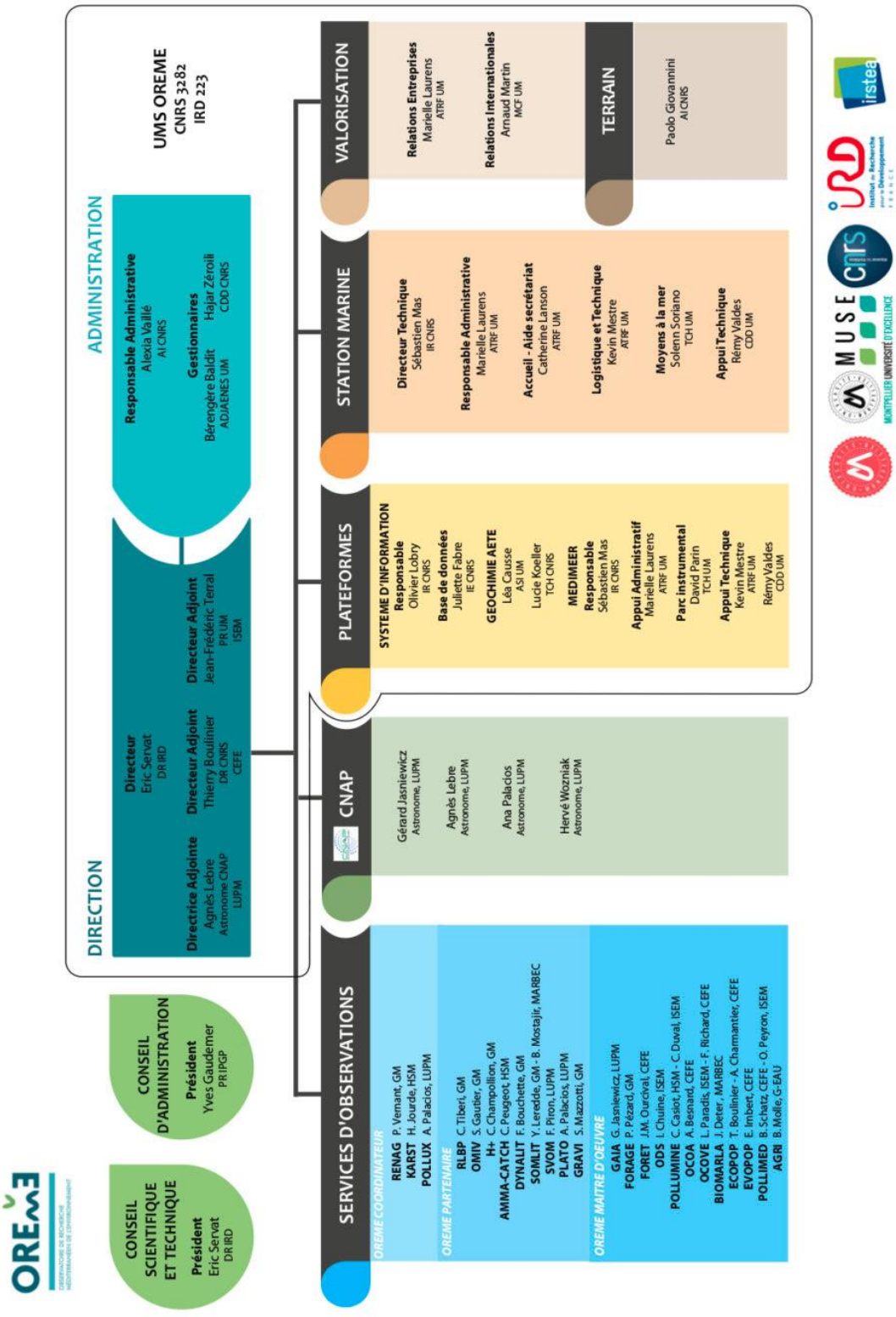

Figure A.1 : Organigramme de l'OSU OREME

Annexe B

Fonctions PHP de l'existant

```php
/**
 * Returns graph tree for given informations theme
 * @param mixed[] $theme
 * @return mixed[]
 */
public function get_graph_tree($theme) {
  //récupération de(s) groupe(s) composant le thème
    $groups = $this->graph_model->get_groups($theme->id_theme);

  //création du tableau graph_tree conservant toutes les informations relatives au thème donné
    $graph_tree = array("info" => $theme, "db_conn" => $theme->db_conn, "groups" => array(), "avail_graph_ids" => array());

    foreach($groups AS $group)
    {
      //récupération des informations sur le groupe courant
        $graph_tree["groups"][$group->name] = array("info" => $group, "graphs" => array(), "enabled" => FALSE);
      //récupération de(s) graph(s) composant le groupe courant
        $graphs = $this->graph_model->get_graphs($group->id_group);

        foreach($graphs AS $graph)
        {
          //récupération des informations sur le graphique courant
            $graph_id = "graph-" . $group->name ."-". $graph->name;
            $graph_tree["groups"][$group->name]["graphs"][$graph->name] =
            array("info" => $graph, "vars" => array(), "schema" => $theme->schema, "id" => $graph_id, "enabled" => FALSE,
             "min" => $graph->min, "max" => $graph->max, "legend" => $graph->legend);
          //récupération de(s) variable(s) composant le graph courant
            $vars = $this->graph_model->get_vars($graph->id_graph);
            $graph_tree['avail_graph_ids'][] = $graph_id;

            foreach ($vars AS $var)
            {
              //récupération de(s) variable(s) discriminante(s) composant la variable courante
                $discrims =  $this->graph_model->get_var_discrims($var->id_var);
                $var->discrims =  $discrims;
              //récupération des informations sur la variable courante
                $graph_tree["groups"][$group->name]["graphs"][$graph->name]["vars"][$var->name] = array("info" => $var,
                 "discrims" => $discrims);
            }
            $graph_tree["groups"][$group->name]["graphs"][$graph->name]["var_tree"] = $this->graph_model->get_var_tree($vars);
        }
    }
    return $graph_tree;
}
```

Figure A.2 : Fonction PHP existante de création du graph_tree

Annexe C

Fonctions JavaScript de l'interface

```
287   /**
288    * Trouver toutes les séries temporelles correspondantes aux mots clé
289    */
290  function availableTs()
291  {
292    //tout les mots clé choisis par l'utilisateur
293    var allKeywords=new Array();
294    $('.ui-autocomplete-multiselect-item').each(function(){
295      allKeywords.push($(this).text());
296    })
297
298    //création d'un sélecteur en récupérant les mots clé un par un
299    var selector="";
300    allKeywords.forEach(function(keyword){
301      //concaténation du sélécteur du mot clé courant
302        selector=selector+'[keywords*=\"'+keyword+'\"]';
303    })
304
305    //création d'un tableau contenant tous les id des séries
306    //correspondantes aux mots clé
307    var results=new Array();
308    $(selector).each(function(){
309      results.push($(this).attr('ts_id'));
310    })
311
312    return results;
313  }
```

Annexe A.3 : Fonction JavaScript du système de filtres

```
503  /**
504   *Remove a graph and put it back in the list if it matches with current keywords
505   *param : target graph id
506   */
507  function deleteGraph(id)
508  {
509    //on supprime la section contenant le graphique ciblé
510    $("div.panel-default[ts_id="+id+"]")[0].remove();
511
512   //on replace la série dans la sidebar en réinitialisant son style css
513    $(".ts_list[ts_id="+id+"]").removeAttr('style');
514    $(".ts_list[ts_id="+id+"]").css({
515      'border': '1px solid lightgrey',
516      'text-align': 'center',
517      'padding': '3px',
518      'margin-bottom': '3px',
519      'border-radius': '8px',
520      'position': 'relative',
521    })
522
523   //si l'utilisateur a choisi des mots clé
524    if(!$('.ui-autocomplete-multiselect-item').text()==""){
525      //on vérifie si la série ciblée correspond aux mots clé courants
526      var results=availableTs();
527      //ne pas l'afficher dans la sidebar si elle ne correspond pas
528      if(!results.includes(""+id+""))
529      {
530        $(".ts_list[ts_id="+id+"]").css("display","none");
531      }
532    }
533  }
```

Annexe A.4 : Fonction JavaScript de suppression des graphiques

```
/**
 * Add a new tab with its content
 */
function addTab()
{
    cptGroup+=1;
    nbNewTab+=1;

//--------------------CREATION OF THE NEW TAB-----------------------------//

  //création de l'élement qui va contenir l'onglet
    var newTab=document.createElement('li');
    //on le rend actif c'est à dire qu'il est sélectionné
    //et que son contenu est affiché
    $('li.active').removeClass('active');
    newTab.className='active tab-'+nbNewTab;

  //création du nouvel onglet
    var newA=document.createElement('a');
    newA.id=nbNewTab;
    newA.setAttribute('role','tab');
//on met un événement au double-clic pour pouvoir modifier le nom de l'onglet
    newA.setAttribute('ondblclick','editGroupName('+newA.id+')');
    newA.setAttribute('data-toggle','tab');
    newA.href='#tabs-'+nbNewTab;
    newA.innerHTML='Onglet '+nbNewTab;

//on créer l'icône pour supprimer l'onglet
    var newImg=document.createElement('i');
    newImg.className="glyphicon glyphicon-remove";
  //on met un événement au clic pour pouvoir supprimer l'onglet
    newImg.setAttribute('onclick','deleteTab('+newA.id+')');
    newImg.style="margin-left : 8px;"

//on positionne les élements au bon endroit dans la page
    newTab.appendChild(newA);
    newA.appendChild(newImg);

//on insére l'onglet à la derniere position (juste avant l'icone '+')
    var matches = document.querySelectorAll("ul#tabList > li");
    $(newTab).insertBefore(matches[cptGroup-1]);
```

Annexe A.5 : Extrait de la fonction JavaScript d'ajout d'un onglet

```
/**
 * Creation of input to edit group name / tab name
 * param : target tab id
 */
function editGroupName(id)
{
  //on récupère l'élément qui contient l'onglet à éditer
    var parentLi=document.getElementsByClassName('tab-'+id)[0];

  //création de l'input
    var newInput=document.createElement('input');
    newInput.id="newGroupName";
    newInput.setAttribute('type','text');
    newInput.setAttribute('placeholder','Entrer un nom');
    newInput.style="width: 150px; border: 0;border-radius: 8px;color: #ffffff;"
    background-color: #337ab7;height: 38px;padding-left: 7px;";

 //on cache l'onglet ciblé
    $('a#'+id).css('display','none');
  //on insére l'input à la place
    parentLi.appendChild(newInput);

//événements pour valider le nouveau nom de groupe
/* When user push 'enter' */
    $(newInput).keydown(function(e) {
        if (e.keyCode == 13) {
            setGroupName(id);
        }
    });
/* When user click elsewhere */
    $(newInput).focusout(function() {
      setGroupName(id);
    });
}

/**
 * Change old tab/group name by new one, typing in input
 * param : target tab id
 */
function setGroupName(id)
{
  //on récupère le nom entré dans l'input
    var newName=$('#newGroupName').val();
  //on supprime l'input
    $('#newGroupName').remove();
  //on affiche l'onglet
    $('a#'+id).css('display','block');
  //si l'onglet ciblé n'est pas l'onglet 1, on ajoute le nouveau nom
  //et l'icone croix pour le supprimer
    if(id!=1){
      $('a#'+id).html(newName+"<i class=\"glyphicon glyphicon-remove\" onclick=
      \"deleteTab("+id+")\" title=\"<?php echo lang('close');?>\" data-toggle="
      \"tooltip\" data-placement=\"right\" style=\"margin-left: 8px;\"></i>");
    }
    //sinon on ajoute seulement le nouveau nom
    else{
      $('a#'+id).html(newName);
    }
}
```

Annexe A.6 : Fonctions JavaScript d'édition du nom d'onglet

Résumé

Le projet mentionné dans ce rapport est une interface destinée à être utilisée par tous les utilisateurs du site de données de l'OREME, qu'ils soient scientifiques ou simples visiteurs. Cette interface a été conçue pour offrir la possibilité à chacun d'entre eux de définir et de sauvegarder une configuration de graphiques "à la carte" sur le site. Pour cela, l'utilisation de filtres et le choix des séries temporelles permettent à chacun de personnaliser sa page de graphiques et d'y avoir accès lors de chaque connexion, en la sauvegardant dans ses préférences.

Mots-clés : interface web, graphiques, séries temporelles, filtration par mots-clés, personnalisation, sauvegarde.

Abstract

The project mentioned in this report is an interface intended to be used by all users of the OREME data site, both scientists and visitors. This interface was designed to provide each of them with the ability to define and save a customizable chart configuration on the site. To do this, the use of filters and the choice of time series allow users to customize their graphic page and have access to it during each connection, by saving it in their preferences.

Keywords : web interface, charts, time series, filtration by keywords, customization, backup.