Rapport de stage

Projet Managechart

Stage LETG Brest Geomer
du 23/10/17 au 15/12/17

Alexandre SANQUER
Maitre de stage : Mathias ROUAN
Tuteur : Gilbert DELPECH

Remerciements

Je tiens tout d'abord à présenter mes remerciements à Mathias ROUAN pour l'aide qu'il m'a apporté, sa disponibilité et le suivi de mon travail. Je le remercie pour le soutien ainsi que pour les réponses à mes questions.

Je remercie également Cyril, avec qui j'ai partagé le bureau durant la durée du stage. De plus, je tiens à remercier l'ensemble de l'équipe LETG Brest Geomer pour leur accueil, l'ambiance au sein de leur bâtiment et l'intérêt apporté à mon projet.

Enfin, je tiens à remercier Gilbert DELPECH et Benoist DUBUC, ainsi que l'ensemble de l'ensemble des personnes associé à la formation DL de l'établissement AFPA de Brest. Egalement, je souhaite remercier Jean-Michel de la POP de Brest pour m'avoir dirigé vers la formation DL.

Merci encore à vous tous.

TABLE DE MATIÈRES

ABSTRACT

In order to validate my training as a software developer at the AFPA in Brest, I worked as an intern from October 23th to December 15th 2017 under the supervision of Mathias ROUAN, Research Engineer at CNRS.

This internship took place within the LETG laboratory (Littoral - Environnement – Télédétection - Géomatique) / Brest Geomer, specialized in the coastal environment of temperate and tropical ecosystems.

The subject of the project was to continue the development of an existing web application, ManageChart, allowing the users to create and export graphs for their researches. Developed in PHP, it also uses a Javascript library called 'Highcharts'.

Part of ROZA, a scientific project, ManageChart requires to allow the scientific community to use more graphs than what is currently available on the web application. For instance, in order to show the result of measurements of cores of sedimentary rocks, the application would require to use new graphic types, such as heatmaps.

The main objective of the intership is to develop new features for ManageChart by including new types of graphics for the users to represent their studies with. In addition of that, bug fixes will also be required amongst existing types of graphics.

Using Highcharts, heatmap and error bar graphics will be implemented to the application. In addition to that, specific forms will be added to it, using Symphony2 and Doctrine, framework used to provide entities for databases, which allows the user to generate an appropriate database, hosted with PostgreSQL, and the dedicated forms.

The project will require to use the existing code and respect the structure of it. In addition to that, it will require to leave, at least, a way to trace my modifications within the files, in order to be able to determinate where to resume to work, if unfinished.

I. INTRODUCTION

Afin de valider ma formation en tant que développeur logiciel, effectué au centre AFPA de Brest, j'ai effectué un stage du 23 octobre au 15 décembre 2017 sous la supervision de Mathias ROUAN, ingénieur-chercheur au CNRS.
Il a été effectué au sein du laboratoire LETG (Littoral - Environnement - Télédétection - Géomatique) / Brest Geomer, spécialisé dans les environnements côtiers et les écosystèmes tropicaux.

Le but du projet était de continuer le développement d'une application web existante et fonctionelle, ManageChart, permettant aux utilisateurs de créer et d'exporter des graphiques pour leurs recherches. Développé en PHP, l'application utilise une librairie Javascript 'Highcharts'.

Prenant part au sein de ROZA, un projet scientifique, ManageChart nécessite de permettre à la communauté scientifique d'utiliser de nouveaux types de graphiques. Par exemple, afin de montrer les résultats de mesure des carottes sédimentaires, l'application aurait besoin d'utiliser de nouveaux types de graphiques, tel que les "Heatmaps" (carte de chaleur).

Le but principal du stage est de développer ces nouvelles fonctionnalités au sein de ManageChart. En plus de ça, il est nécessaire de fixer des bugs et ajuster des paramètres au sein de l'application.

Utilisant Highcharts, des graphiques Heatmap et barres d'erreur seront implémenté dans l'application web. Associé à ceci, il sera nécessaire d'inclure de nouveaux formulaires, utilisant Symphony2 et Doctrine, frameworks pour fournir des entités pour les bases de données, permettant à l'utilisateur de générer les éléments appropriés, hébergé avec PostgreSQL.

Le code devra respecter la structure actuelle du projet. En plus de ceci, il sera nécessaire de commenter et laisser une trace concernant les modifications au sein de projet, afin de faciliter la reprise du code, si non complet.

1) Présentation du LETG Brest Géomer

LETG-Brest Géomer est un des cinq laboratoires de l'UMR LETG (Littoral, Environnement, Télédétection, Géomatique). La partie brestoise est pluridisciplinaire (géographie humaine, géographie physique, géomatique), et ses axes de recherche se déclinent en sept grands thèmes :

- dynamiques de l'occupation des sols
- dynamiques géomorphologiques des littoraux
- risques côtiers
- fréquentation et usages (espaces littoraux et insulaires)
- gestion intégrée des zones côtières
- modélisation des activités humaines
- géomatique.

Figure 1 : Visualiseur d'Indigéo zoomé sur le technopôle et la plage de Sainte Anne, avec la station Somlit

De plus, dans le cadre de l'OSU-IUEM, LETG Brest - Géomer alimente aussi une base de données sur l'évolution du trait de côte, grâce à un suivi morpho sédimentaire des plages du Finistère.

2) Mission du stage

Ce stage s'inscrit dans le cadre du projet « Rétro-Observatoire des Zones Ateliers » - ROZA qui est un projet du Réseau National des Zones Ateliers visant à bancariser à l'échelle de la France, les données et échantillons issus de carottes sédimentaires lacustres de référence afin de favoriser leur réutilisation à une échelle plus vaste.

Une des tâches de ce projet consiste à mettre en oeuvre une plateforme de catalogage et de diffusion de ces données de carottes sédimentaires. Elle permettra ainsi le porter à connaissance des données et disposera de fonctionnalités de visualisation et de téléchargement.

Cette plateforme s'appuie sur des outils développés dans le cadre de l'Infrastructure de Données Géographiques indigeo et parmi lesquels on trouve geoCMS (un outil générique de visualisation et d'interrogation de données géoréférencées) et manageChart : un outil de visualisation de graphiques basé sur la librairie highcharts.js et permettant de réaliser des requêtes paramétrées sur des bases de données.

Cette IDS est destinée aux scientifiques désirant faire de la recherche et de l'observation sur l'environnement Ouest de la France. Elle se présente sous la forme d'un visualiseur cartographique, d'un catalogue de métadonnées et d'un serveur de données géospatialisées. Indigéo respecte la directive INSPIRE de la Direction générale de l'environnement de la Commission européenne.

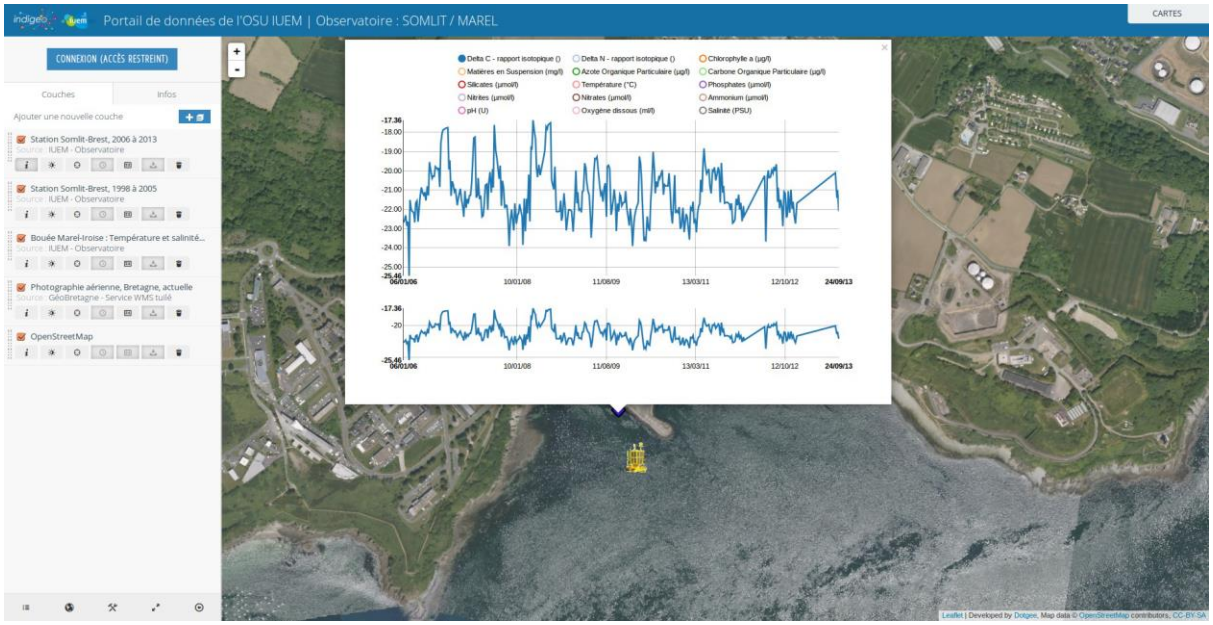

Figure 2 : affichage d'un graphique dans le visualiseur de Indigéo

L'objectif du stage est double : dans un premier temps, il s'agit d'ajuster l'application web ManageChart, qui permet de gérer les différents graphiques qui seront ensuite exporter sous Indigeo, avec différent fix de bugs. Par la suite, il s'agit de poursuivre les développements de ManageChart en ajoutant de nouveaux modes de représentation graphique permettant de répondre aux besoins de visualisation des données de carottes sédimentaires.

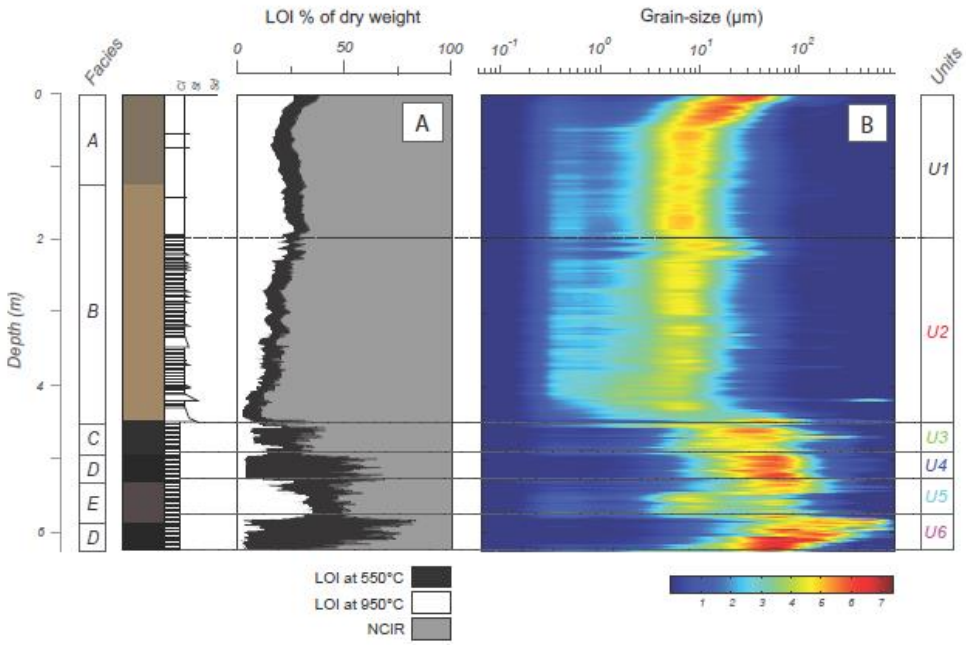

Figure 3 : Exemple de représentation des données de carottes sédimentaires généré sous MatLab du projet ROZA

En plus de ces graphes, l'ajout des types de série à barre d'erreur permettrait de combler les besoins en terme de visualisation de ces carottes.

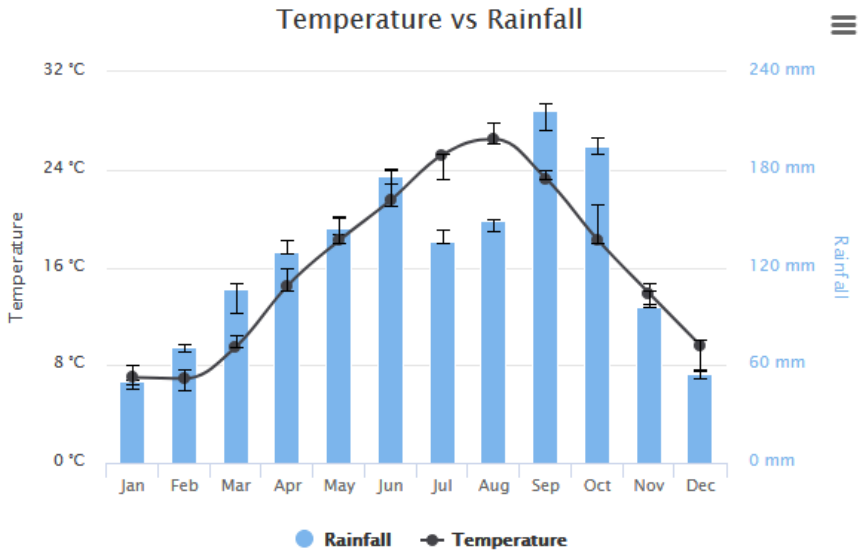

Figure 4 : Exemple de graphique d'erreur

II. CONTEXTE DU STAGE : MANAGECHART

Les graphiques disponibles sur l'application Indigeo sont en réalité des graphiques visualisés à travers une iframe depuis l'application ManageChart, une application développé en PHP, Javascript et utilisant notamment plusieurs librairies & framework supplémentaires :

- **Symfony2 & Doctrine**

Présent au sein de l'application ManageChart, Symfony2 est un framework MVC (Modèle Vue Contrôleur) permettant la gestion d'entité, qui, associé à Doctrine, permet d'assurer la cohérence avec la base de donnée, hébergée sous PostgreSQL. Utilisant Symfony2, il est également possible de générer des templates pour les pages & formulaires.

- **jQuery & Boostrap**

En ajoutant jQuery, permettant de faciliter la création des scripts côté clients pour l'affichage des pages, et Bootstrap pour l'affichage CSS, il permet de réduire le temps de développement.

- **Highcharts**

Utilisé au sein de ManageChart, Highcharts est une librairie Javascript permettant de générer des graphiques, qui sont ensuite paramétrer et insérer dans des fenêtres qui seront ensuite visualiser au sein de l'application Indigeo.
Associé aux formulaires templates, ceci permet de générer un large spectre de graphiques. Il permet également d'insérer plusieurs séries ou d'insérer différent axes sur le même graphique.
Actuellement sont présents :

- Graphiques simples
- Graphiques multiaxes
- Graphiques temporels
- Graphiques dynamiques
- Graphiques circulaires

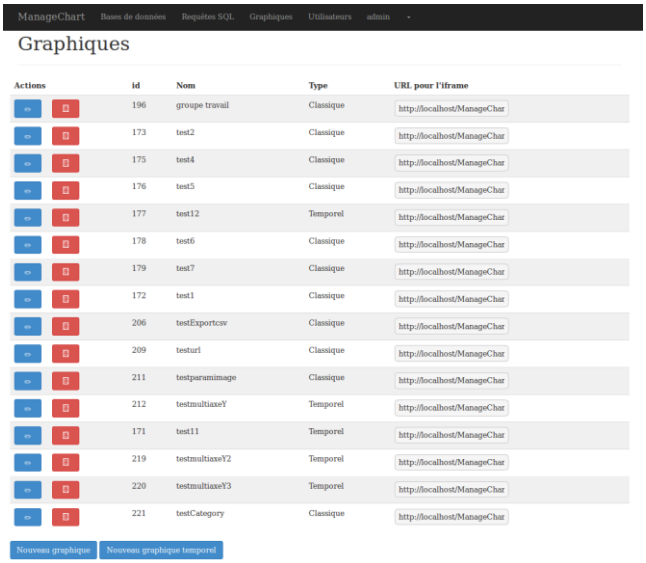

Figure 5 : Page d'accueil de l'application web ManageChart

1) Fonctionnalités

Afin de décrire le travail effectué, il est nécessaire de décrire les fonctionnalités de base de ManageChart. Certaines fonctions, comme celles associées aux utilisateurs, seront omises car elles ne sont pas pertinentes dans les développements effectués.

Tout d'abord, afin de créer un graphique, il est nécessaire de créer une connexion avec une base de donnée, hébergée sous PostgresSQL ou MySQL. Par ailleurs, il est nécessaire de faire la distinction entre les données en base pour ManageChart et les données scientifiques qui peuvent être sauvegardé ailleurs.

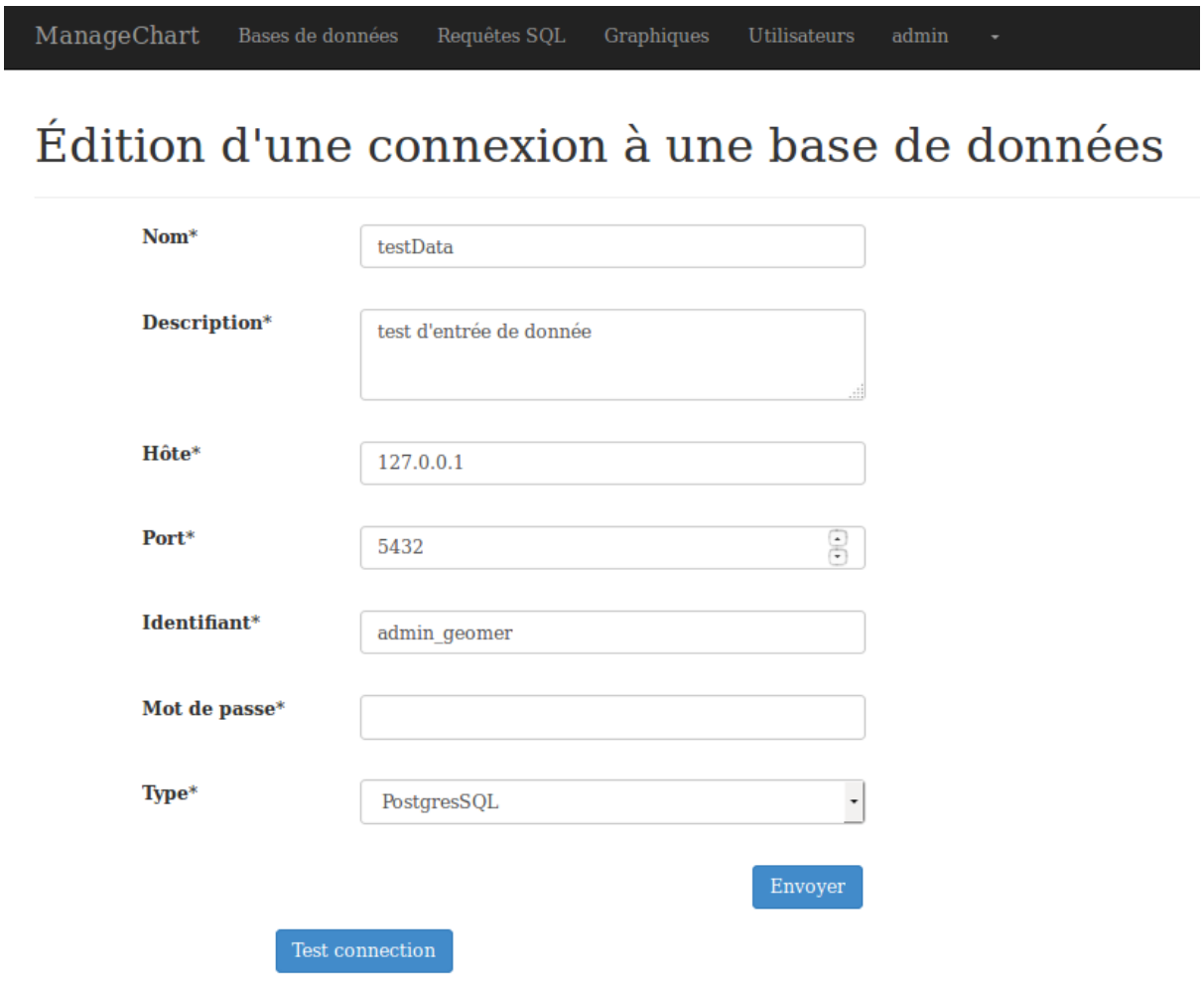

Figure 6 : Création d'une base de donnée

Ensuite, il est nécessaire de créer une requête SQL. Ces requêtes permettent d'extraire les données qui seront ensuite utilisé par ManageChart pour la création des graphiques. Les données nécessitent d'être formatées d'une certaine façon pour l'insertion des données.

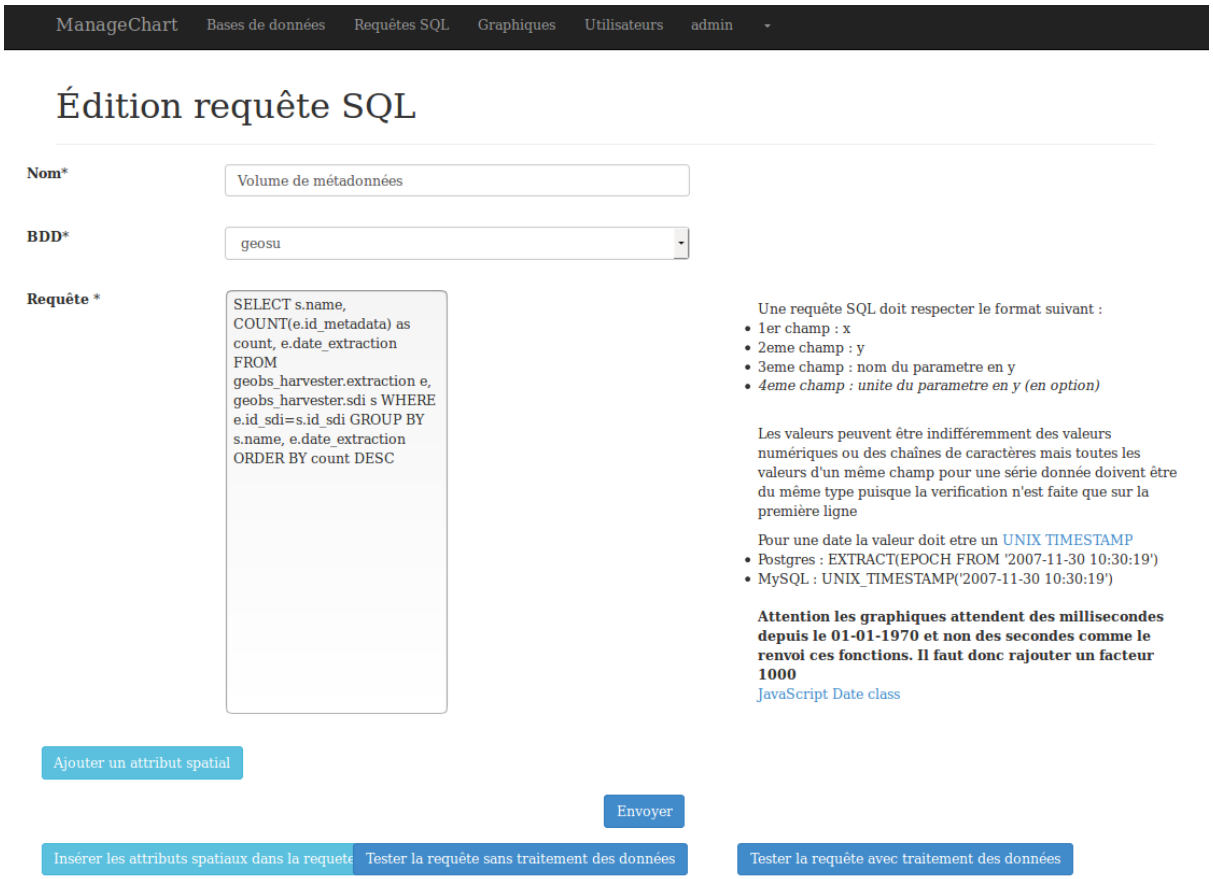

Figure 7 : Création d'une requête SQL

Enfin, la création d'un graphique se fait par un formulaire, par lequel une preview du graphique est disponible pour l'utilisateur. Les caractéristiques principales du graphique sont définies sur cette page.

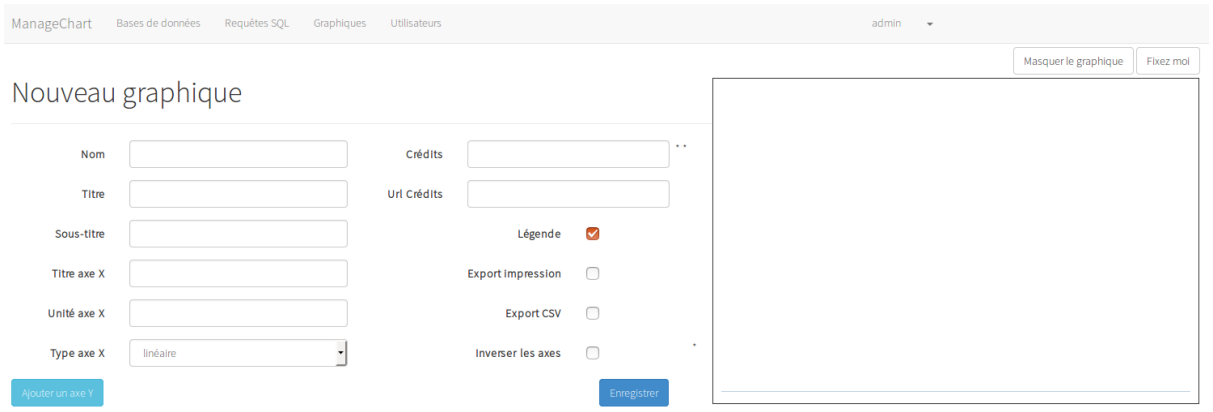

Figure 8 : Formulaire de création d'un graphique

Pour un graphique, il est possible d'afficher, pour un même axe X, plusieurs axes Y. Il est possible d'éditer le titre de l'axe qui sera affiché sur le graphique.

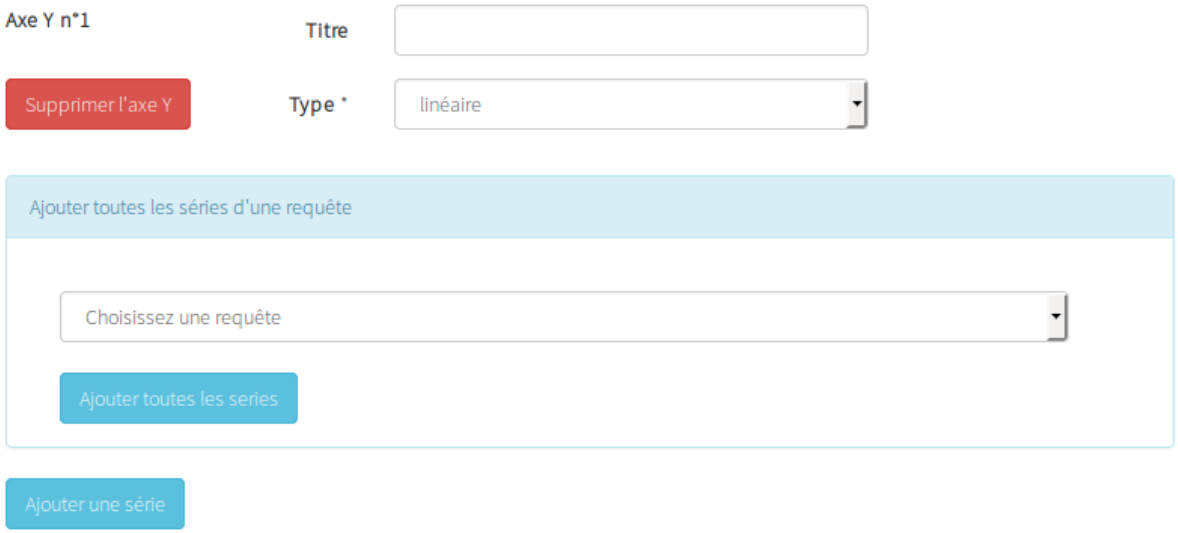

Figure 9 : Ajout d'un axe Y

Enfin, affecté à un axe Y, il est possible d'ajouter plusieurs séries, auquel seront affectées les requêtes SQL pour l'insertion des données.

Figure 10 : Ajout d'une série

L'ensemble de ce formulaire est généré par Symfony par le biais de templates. Selon le type de graphique ajoutés, certains champs sont ajoutés ou retirés, préparant partiellement le formulaire et les données insérées au sein de la base de données.

2) Analyse de l'existant

De façon général, vu que le stage requiert de respecter la base de donnée existante, quelques modifications seront effectué au niveau de la base de donnée.

Le dictionnaire de données est composé de la façon suivante :

- chart représente le graphique général.
- yAxis représente les axes verticaux du graphique chart
- series représente les séries de données
- data_list représente les requêtes
- data_source représente les bases de données externes où sont situées les données à visualiser
- attribut_spatial représente un paramètre de la requête pouvant être injecté via l'URL du graphique et notamment utilisé pour filtrer les données sur leur localisation géographique
- account représente les utilisateurs
- flag représente les séries de données sous forme de balise

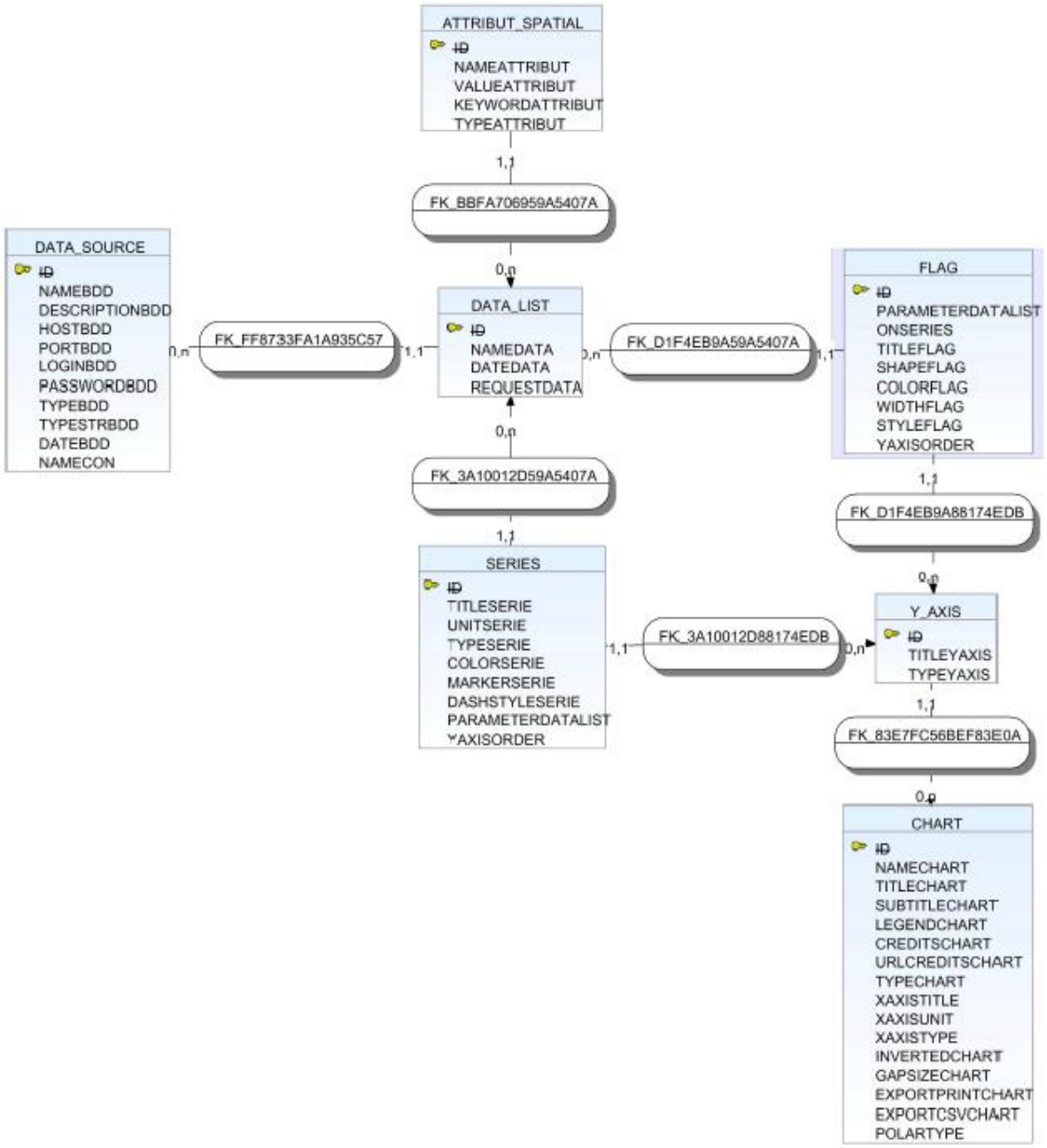

Figure 11 : Modèle conceptuel des données

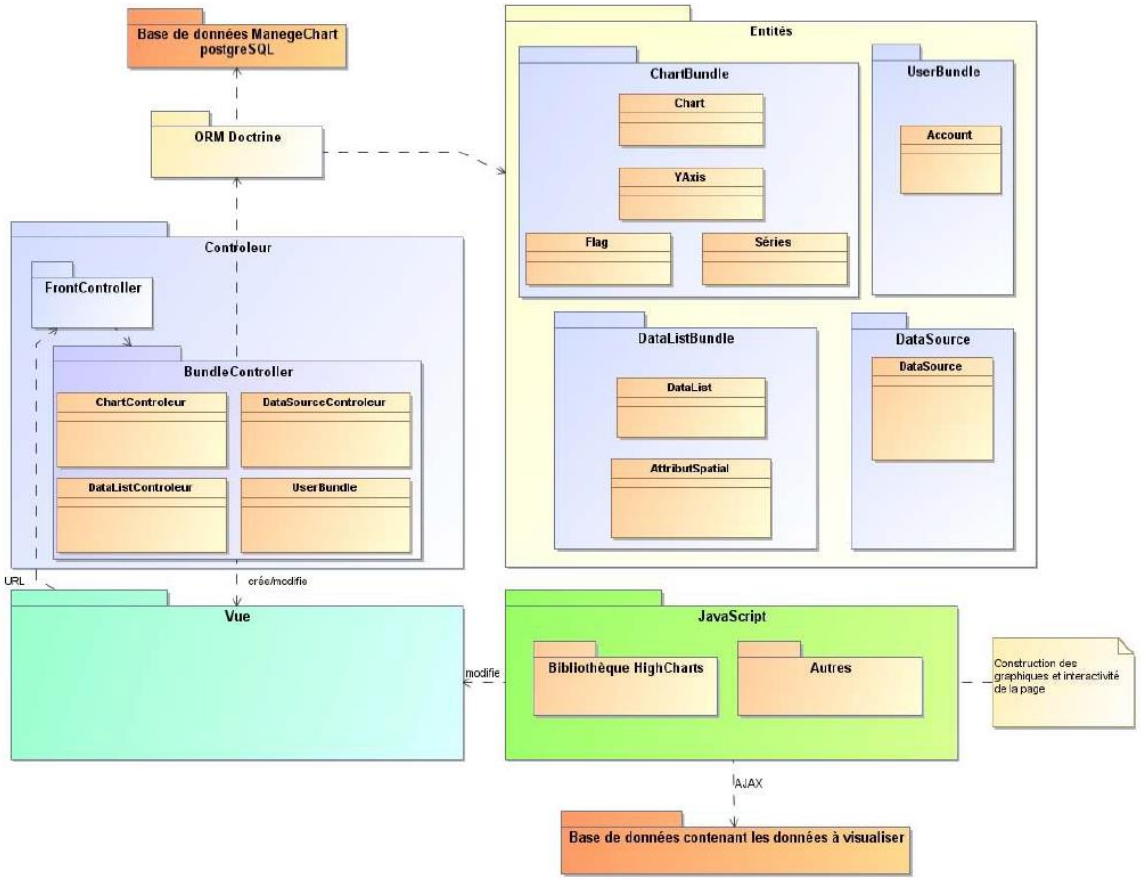

Figure 12 : Architecture générale de ManageChart

3) Tâches à réalisé

Tout d'abord, afin de faciliter la prise en main du logiciel, il sera nécessaire de corriger certains bugs présents dans l'application web et d'ajouter des fonctionnalités mineures

- Affichage des labels obligatoires
- Sauvegarde et affichage des données au sein des formulaires
- Erreur lors de l'ajout des données lors d'un nombre important de données
- Sauvegarde et génération de couleur pour les graphiques circulaires
- Inversion des axes X & Y

Ensuite, l'ajout des nouveaux types de graphiques associé au projet ROZA représentera l'ensemble du stage.

- **Graphiques de type "Heatmap" (carte de chaleur)**
  Ces graphiques permettent d'afficher des valeurs sur un axe X & Y et d'associer, à chaque point, une valeur qui est représenté par une couleur.

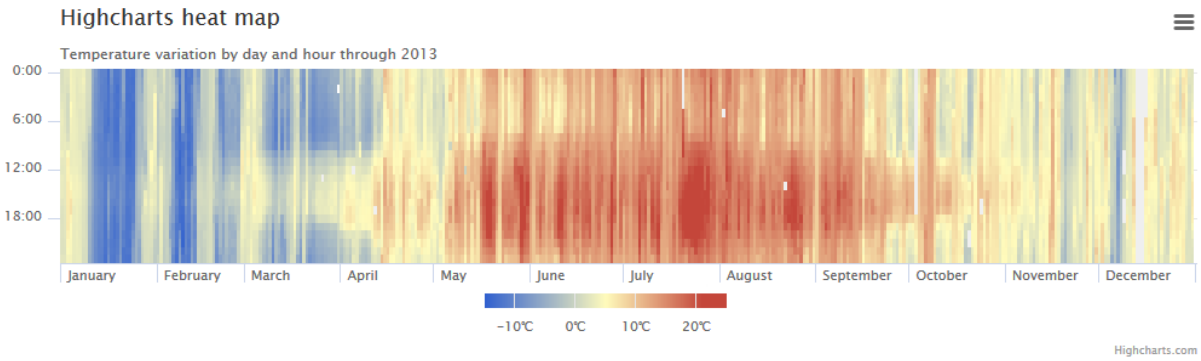

Figure 13 : Exemple de graphique "Heatmap" présent sur le site Highcharts

- **Graphiques à barre d'erreur**
  Ces graphiques, associé à une autre série, permettent d'afficher les échelles d'erreur pour un point donné.

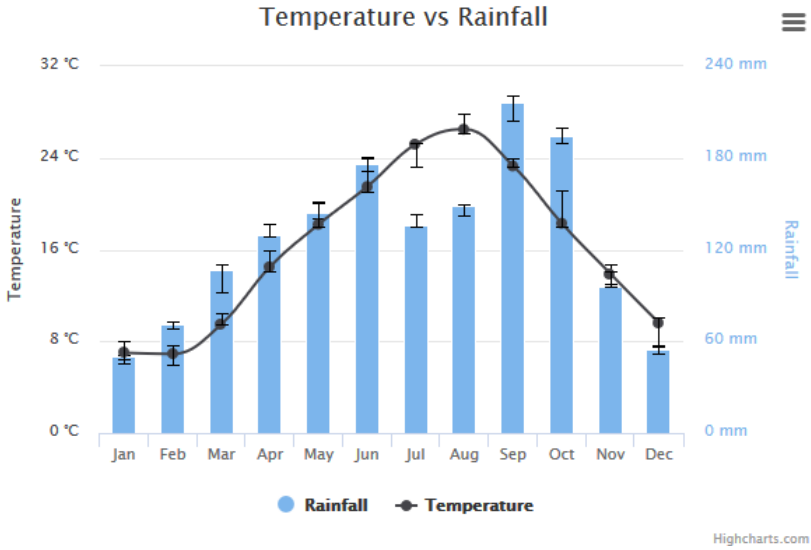

Figure 14 : Exemple de graphiques à barre d'erreur présent sur le site Highcharts

Planification du projet

		Nom	Durée	Début	Fin	Prédécesseurs
1		Découverte du projet	5 jours	23/10/17 07:00	27/10/17 17:00	
2		Correction des bugs	9 jours	30/10/17 09:00	10/11/17 09:00	1
3		Développement de Heatmap	19 jours	13/11/17 09:00	08/12/17 09:00	2
4		Développement de Piechart	14 jours	20/11/17 09:00	08/12/17 09:00	2
5		Inversion des axes	4 jours	04/12/17 09:00	08/12/17 09:00	2
6		Finalisation du projet	5 jours	08/12/17 09:00	15/12/17 09:00	3;4;5

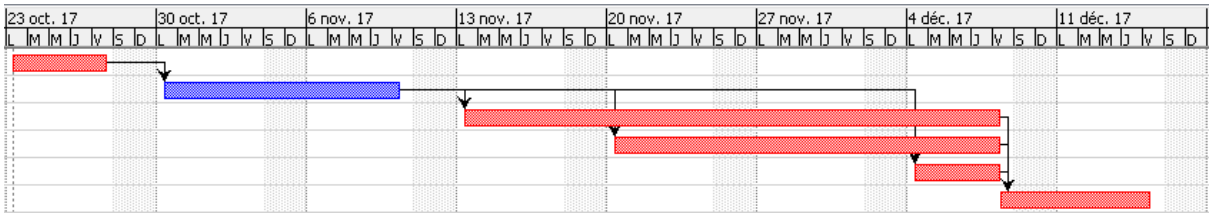

Figure 15 & 16 : Planification du projet

III. RÉALISATION DU PROJET

1) Correction des bugs

- **Ajout des champs obligatoires**

Le problème présent au sein de l'application est que les champs obligatoires n'étaient pas identifiables pour l'utilisateur, mis à part lors de la tentative d'enregistrement du graphique.

Généré par Symfony, Doctrine & le template des formulaires, une modification dans le fichier .css pour les champs "required" d'une astérisque rouge permet de visualiser les champs obligatoires.

```
.required::after {
    content: "*";
    color: red;
}
```

- **Sauvegarde et affichage des données au sein des formulaires**

Des précédents développements ont rajouté des paramètres au sein des formulaires. Toutefois, bien qu'ils soient bien enregistrés en base, ces valeurs n'étaient pas chargé lors de l'édition d'un graphique. Ceci pose problème pour la pré-visualisation du graphique et l'enregistrement des valeurs. Cela est gouverné par les scripts Javascript qui chargent les données au sein de l'application.

La modification de ces scripts Javascript pour inclure l'insertion des données au sein du formulaire a été effectué.

Voir Annexe 1

```
$('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS + '_innersize').val('{{ serie.innersize|e('js') }}').change();
$('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS + '_colorSerie').val('{{ serie.colorSerie|e('js') }}').change();
```

- **Erreur lors de l'ajout d'un nombre important de données**

Lors de l'ajout d'un nombre important de données au sein d'un graphique, le script de création du graphique rencontrait un problème et interrompait de manière inopiné. Après investigation, ce problème provenait d'un des composants de Highcharts appelé Boost. Ce composant permet, lorsqu'un seuil est dépassé, de simplifier l'affichage des données au sein du graphique.
Ce module requiert, lors de l'activation du module Boost, d'insérer les données sous une forme spécifique : plutôt d'accepter des données sous forme JSON, les valeurs devaient être insérées sous la forme d'un tableau.

Deux solutions se proposaient alors : la première est de désactiver ce module, ce qui résulterait à des créations de graphiques plus longues. Toutefois, il s'est révélé que ce problème provenait partiellement du module, il n'avait pas été mis à jour depuis l'ajout de nouvelles fonctionnalités au sein de Highcharts : mettre à jour le module Boost permet alors de régler la situation.

- **Sauvegarde & génération de couleur pour les graphiques circulaires**

Plutôt que d'utiliser une série de couleur prédéfini pour les couleurs d'un graphique circulaire, de type camembert, les couleurs étaient actuellement généré de façon aléatoire, parmi une liste de couleur choisi.
Souhaitant avoir une couleur fixe, la solution retenu est de générer une gamme de couleur à partir du nombre d'éléments en utilisant un algorithme qui, à partir d'une liste prédéfini, génère la liste de couleur utilisé.

Ceci fait appel à une conversion des couleurs en valeur hexadécimal en valeur RGB (Red Green Blue), d'ajouter une valeur RGB à cette couleur et de convertir à nouveau en couleur en valeur hexadécimal.

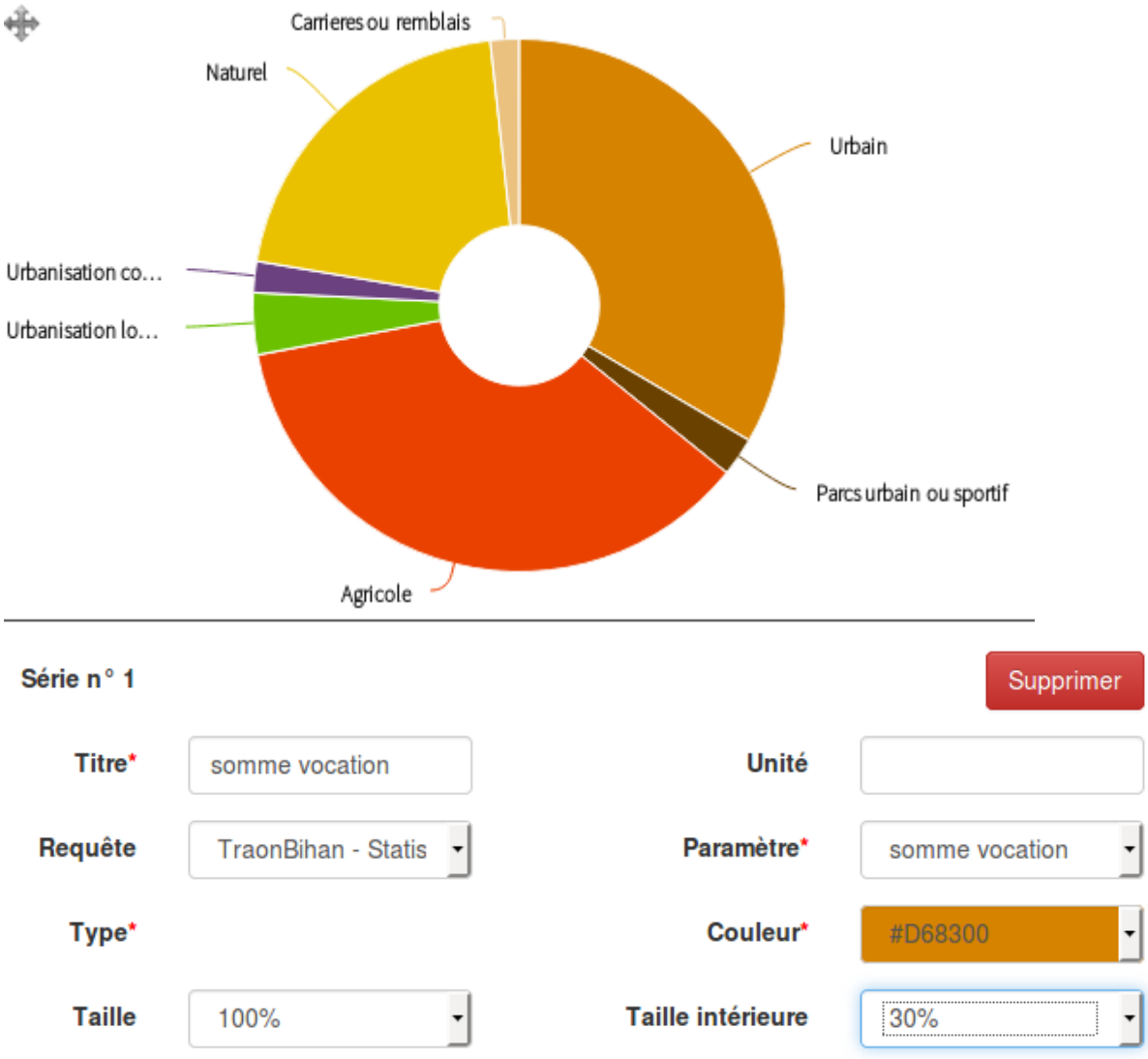

Figure 17 & 18 : Graphique circulaire

Voir Annexe 2 & 3

2) Position des axes Y

En ajoutant plusieurs axes Y, il est nécessaire de positionner chaque axe Y sur le graphique. Ceci se fait par l'ajout de deux paramètres aux axes Y : Top & Height. Ces valeurs permettent de modifier la taille du graphique et la position de l'axe en pourcentage.

Figure 19 : Ajout des paramètres Top & Height aux axes Y

Combiné à cela, on souhaite, pour l'affichage des graphiques d'une manière verticale plutôt que horizontale, d'ajuster ces mêmes valeurs d'une façon différente.

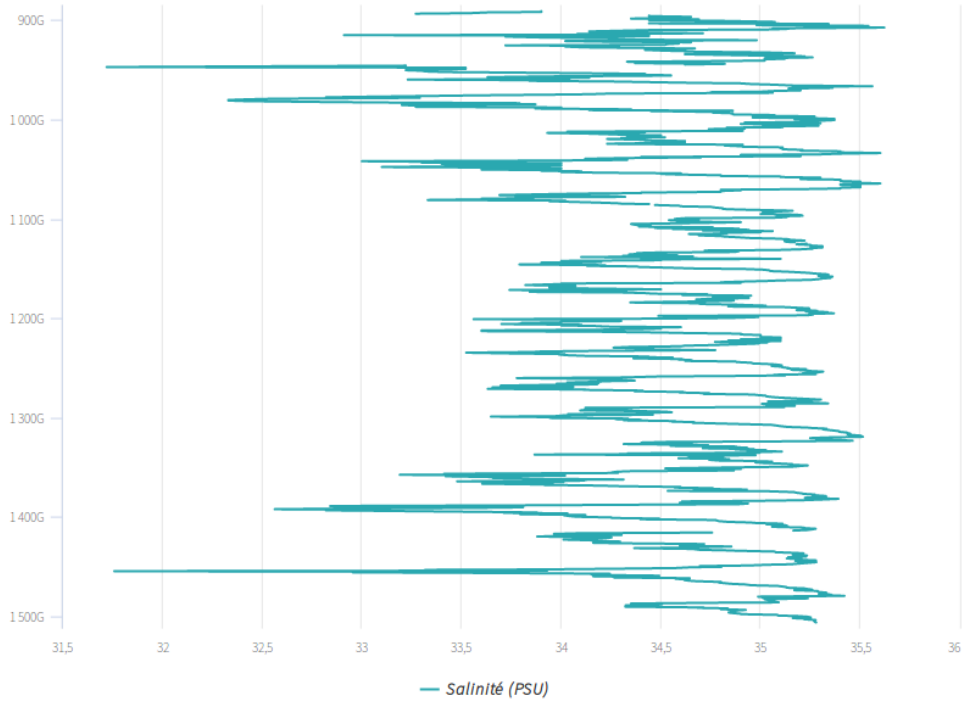

Figure 20 : Inversion des axes

Ceci, toutefois, nécessite d'ajuster la BDD par la mise à jour de la base de donnée par Doctrine et d'exécuter un script pour modifier les anciennes données inséré dans la base de données.

Voir Annexe 4

3) Ajout des graphiques "Heatmap"

Les graphiques "Heatmap" (carte thermique en français) sont des graphiques permettant, pour une valeur X & Y, d'avoir une valeur supplémentaire qui sera représenté par une couleur. Ceci, toutefois, nécessite d'adapter le code, puisque l'insertion des données est différente des graphiques habituels.

- **Ajout des champs spécifiques**

Lors de la création du graphique, il est nécessaire de rajouter des champs spécifiques, lié uniquement aux graphiques de type "Heatmap" : en effet, un graphique de ce type possède trois axes, un axe X, un axe Y et un axe Value, appelé colorAxis dans le cas de Highcharts. Il est alors nécessaire d'ajouter et de configurer cet axe au sein de la création du graphique. Par ailleurs, il est nécessaire d'affecter un spectre de couleur à ce graphique : la Jet Colormap est celle utilisé par les scientifiques de LETG pour la création de leur graphique sur Mathlab.
Voir Annexe 5

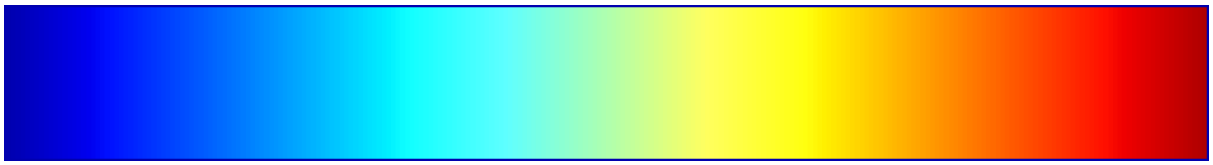

Figure 21 : Colormap JET

- **Formater les données**

Un autre problème se présente alors : dans le cas actuel, lors des requêtes SQL qui servent de base pour insérer les données, seul un champ X & Y était récupéré. Afin de ne pas impacter le code de façon sérieuse, la deuxième valeur de la requête, qui est habituellement la valeur Y, est alors remplacé par un tableau de valeurs, qui contient la valeur Y & la valeur du point.

Ceci, ensuite, est transformé en tableau JSON de la façon approprié pour un heatmap : en effet, plutôt d'avoir un tableau JSON construit de la façon suivante...

```
{"x": valeurX, "y": valeurY}
```

...les tableaux de type Heatmap utilise le format suivant…

```
{"x": valeurX, "y": valeurY, "value": valeurValue}
```

Cela se fait, lors de l'ajout des données, selon le type du graphique.
Voir Annexe 6

- **Type d'axe logarithmique**

En revanche, une limite de Highcharts s'est alors présentée. Lorsqu'on définit le type d'axe X étant logarithmique, les points sont correctement représentés mais ne sont pas assez nombreux pour peupler l'ensemble du graphique, laissant des espaces vides.

A l'opposé de Matlab sur lequel il est possible d'effectuer des calculs d'aires pour remplir les espaces blancs, Highcharts, de son côté, ne permet que d'afficher des points, sans modification préalable.

Ceci pose problème parce que, à moins que la mesure effectuée et insérée en base de données possède assez de points pour peupler l'ensemble du graphique, il restera de l'espace vide au sein du graphique.

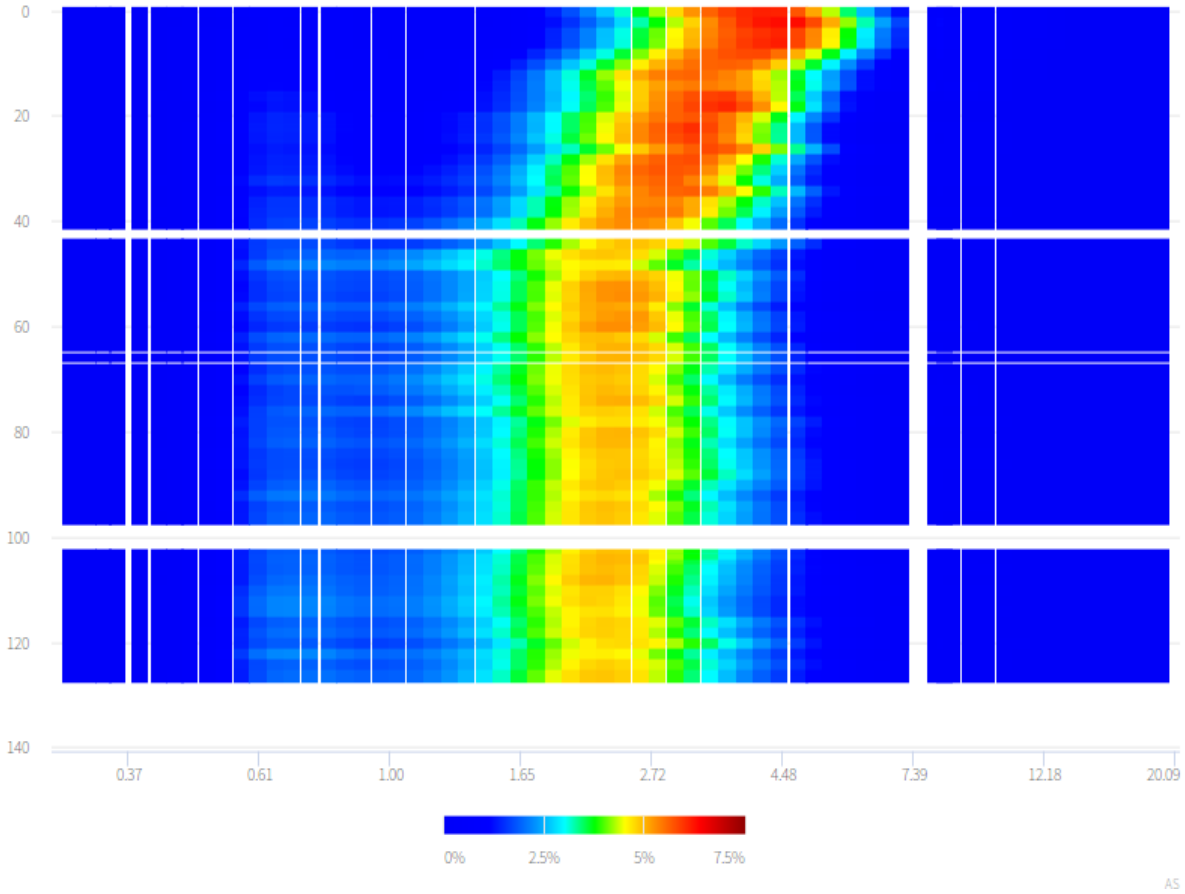

Figure 22 : Heatmap après les modifications

Cela peut être corrigé en procédant de trois façons :

- Premièrement, il est possible, par la requête SQL, de récupérer plus de points. En effet, à l'état actuel, un calcul est déjà effectué pour récupérer la valeur médiane d'une mesure.
  Lors d'une mesure d'une profondeur, par exemple, c'est la distance entre une valeur A & B qui possède une mesure : en doublant par exemple, ce nombre de calcul, il est possible d'augmenter la population du graphique.

- Dans un second temps, il est nécessaire, plutôt de choisir un type d'axe X logarithmique, de conserver un axe X linéaire mais, en revanche, d'affecter pour les valeurs X des valeurs logarithmiques.

  En effet, en procédant de cette manière, les valeurs sont proprement inséré dans le tableau, en réduisant considérablement le nombre de champs vides. Il suffit ensuite de faire la manipulation inverse, soit afficher la valeur exponentielle d'une valeur logarithmique pour refléter les valeurs correctes sur l'axe et l'infobulle.

```
function updateChart(chart){
    var limitValues = getLimitValues(chart.yAxis[0].series[indexSerieChart-1].data);
    defaultColumnSize = (limitValues['maxXAxis'] - limitValues['minXAxis'])/(limitValues['uniqueXAxis']);

    // ASFLAG : Penser à modifier les champs des noms de labels X, Y & Value pour affecter selon la requête/les champs
    chart.update({
        xAxis: {
                        labels: {
                        formatter: function(){
                                return '' + Math.exp(this.value).toFixed(2);
                        }
                }
                },
        yAxis: {
                        reversed: true
        },
        tooltip: {
                pointFormatter: function(){
                        return 'X : ' + Math.exp(this.x) + '<br>' +
                                'Y : ' + this.y + '<br>' +
                                'Value : ' + this.value;
                }
        }
        },false);

    chart.update({
         colorAxis: {
                min: limitValues['minValue'],
                max: limitValues['maxValue']
        }
        },false);

        chart.update({
         series:{
                colsize: defaultColumnSize
        }
        });

        // ASFLAG : Assigne la valeur sur l'Input associé. Si la valeur colsize est défini en base, elle remplacera cette
valeur.
        $('input#mc_chartbundle_chart_list_yAxis_1_series_1_colsize').val(defaultColumnSize).change();
}
```

- Enfin, il est possible d'ajuster la largeur des points, afin d'affiner la représentation des mesures.

  A l'état actuel, ce calcul est fait de façon soit automatique, en récupérant la moyenne des valeurs divisé par le nombre d'éléments unique, soit de façon manuel, par l'ajout d'un champ supplémentaire nommé "colsize" au sein de la série.
  De nouveau, cette modification de la base de données est mise à jour par Doctrine et l'exécution d'un script simple pour adapter la base de données.

Cette solution, bien que provisoire, comble largement les besoins de l'application. En effet, après discussion avec les parties intéressées, il est préférable d'avoir un graphique de ce genre de vide plutôt que les graphiques dont les vides sont comblés par Matlab : l'absence de valeur est une donnée importante plutôt qu'une valeur approximative.

4) Ajout des graphiques à barres d'erreur

A l'instar des graphiques "Heatmap", il est nécessaire de formater la donnée de façon spécifique. En effet, plutôt que d'utiliser une valeur X et Y, ces graphiques utilisent une valeur X, une valeur Low et une valeur High, représentant l'échelle d'erreur affecté à une mesure.

De ce fait, les modifications de codes appliqués par l'ajout du Heatmap s'applique à celui-ci également, à l'exception que les données, plutôt d'avoir un tableau JSON construit de la façon suivante...

```
{"x": valeurX, "y": valeurY}
```

...les barres d'erreur utilisent le format suivant…

```
{"x": valeurX, "low": valeurLow, "high": valeurHigh}
```

Voir Annexe 6

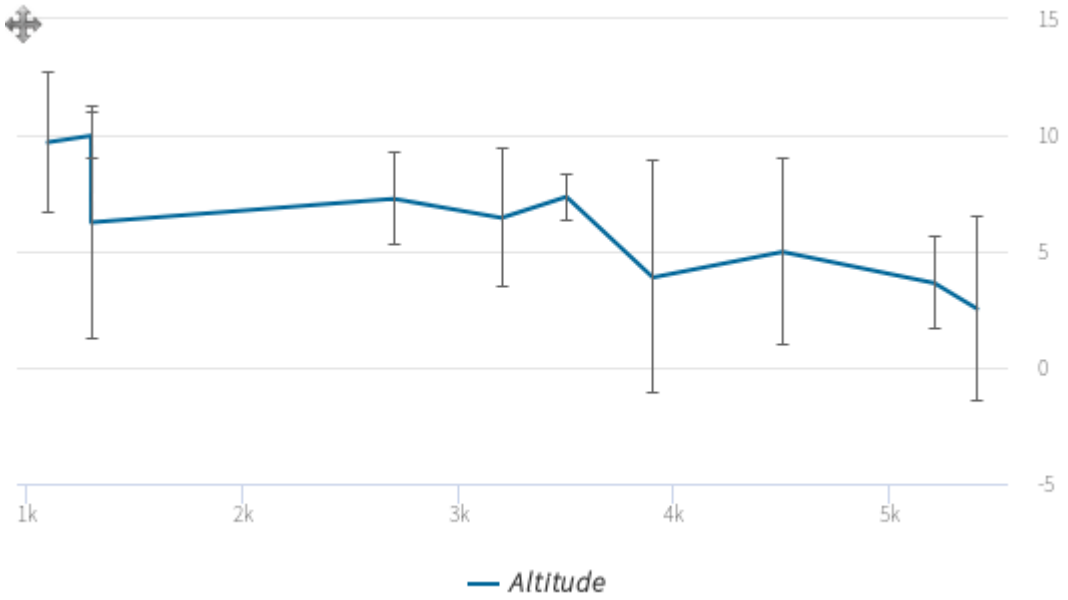

Figure 23 : Exemple de barre d'erreur, affecté comme seconde série à un axe Y

IV. AXES D'AMÉLIORATIONS

A) Au niveau du travail personnel

- **Génération des couleurs pour les graphiques circulaires**

A l'état actuel, l'algorithme des couleurs utilisent un tableau utilisant les extrèmes des couleurs des valeurs RGB. De ce fait, bien qu'on sélectionne une couleur différente avec lequel mélanger ces couleurs, on obtient, au final, des graphiques avec des tons de couleurs assez similaires.

Présent dans le code, développé mais non implémenté, une autre méthode serait plutôt envisageable, utilisant le choix de palette préféré par les scientifiques et parcourant ce tableau en sélectionnant les couleurs, selon le nombre d'éléments présent dans le graphique.

De façon idéal, il serait de même préférable de pouvoir choisir la couleur de chaque champ. Cela, en revanche, impliquerait plus de développement et une refonte partielle du code.

- **Graphique "Heatmap"**

Actuellement, la solution présente au sein du projet réponds spécifiquement aux besoins du projet ROZA, c'est-à-dire, un tableau utilisant des données logarithmiques.
La majorité des graphiques utiliseront ce modèle, mais, dans le futur, ce choix sera obsolète et rentrera en conflit avec les exigences si le graphique concerne des données générales.

La solution proposé est fonctionnelle, possède des limites mais réponds aux besoins immédiats. Des développements seront à prévoir pour une utilisation plus générale.
Des champs supplémentaires dans le formulaire serait adapté.

- **Coquilles au niveau de la présentation**

Certains éléments, bien que valide, nécessite une refonte graphique pour harmoniser l'ensemble de l'application. La modification des templates seraient au cœur de ces modifications.

B) Au niveau de l'application en général

- **Attribuer aux requêtes SQL des type de séries compatibles**

Lors de la sélection d'une requête SQL, le type des données sélectionné n'est pas vérifié. De ce fait, il est possible de sélectionner une série de données incompatibles avec le type du graphique. Ce cas est particulièrement présent dans les nouveaux types de graphiques ajoutés (heatmap et errorbar).

CONCLUSION

Ce stage, de façon générale, est plus que positif : les objectifs sont atteints, des ouvertures sur l'amélioration de l'application ont été identifiés et il m'a permis, avant tout, de renforcer mes compétences professionnelles acquis dans le passé et durant la formation.

Egalement, ces tâches m'ont permis de cerner quelques défauts au niveau de mon travail : bien que je sache formuler, présenter un problème et le cerner, il m'arrive de prendre du temps à voir comment corriger le problème. Toutefois, l'étude et la connaissance du code en général permet d'accélérer et de proposer des solutions aux autres problèmes qui apparaissent au fur et à mesure du projet.
Par ailleurs, mon travail sur le stage a révélé un second défaut dans ma manière de procédé : il se peut que, lancer dans le développement, j'anticipe un peu trop le besoin, augmentant le temps de développement.

Du fait que le développement était axé sur le web, cela m'a permis de consolider mes acquis dans ce domaine, un domaine d'expertise assez recherché sur le marché de l'emploi.

LEXIQUE

Terminologie	Définition
Laboratoire LETG	Laboratoire Littoral - Environnement - Télédétection - Géomatique
Projet ROZA	Rétro-Observatoire des Zones Ateliers
geoCMS	Outil de visualisation cartographique des données
ManageChart	Gestionnaire des graphiques
IDS	Infrastructure de données spatiales
PHP	Langage de programmation web
HTML	Langage de programmation web
Javascript	Langage de programmation web
Symfony	Framework MVC écrit en PHP
MVC	Modèle – Vue - Contrôleur
Doctrine	Module associé à Symfony pour gérer les entités en base de données
PostgreSQL	Base de données
Bootstrap	Librairie Javascript pour générer l'affichage
jQuery	Librairie Javascript pour faciliter l'écriture des scripts
Highcharts	Librairie Javascript pour la création des graphiques
Heatmap	Carte de chaleur
Twig/Template	Modèle prédéfini et paramétrable de pages
Couleur RGB	Red-Green-Blue : Couleur décomposé selon ce spectre
Couleur HEX	Hexadécimal : Couleur codé sous valeur hexadécimal (#000000 à #FFFFFF)
Mathlab	Logiciel scientifique pour réaliser des calculs scientifiques

ANNEXE

Annexe 1 : src/Mc/ChartBundle/Resources/Piechart/edit.html.twig

```
{# src/Mc/ChartBundle/Resources/views/Chart/Polarchart/edit.html.twig #}

{# template affichant le formulaire d'edition d'un chart polar et spiderweb #}

{% extends "McChartBundle::layout.html.twig" %}

{% block body %}
        {% if is_granted('ROLE_SCIENTIFIC') %}
                {% include 'McChartBundle:Form:formChart.html.twig' with {'title' : 'chart.editHighchart'|trans(), 'typeChart':
'piechart'} %}
        {% endif %}
{% endblock %}

{% block javascripts %}
        {{ parent() }}

        {% javascripts
                'js/Chart/color.js'
    'js/Chart/yaxis.js'
    'js/Chart/seriepie.js'
    'js/Chart/highcharts.js'
    'js/Chart/highcharts-more.js'
    'js/Chart/boost.js'
    'js/Chart/piechart.js'
  filter='?yui_js' %}

  <script type="text/javascript" src="{{ asset_url }}"></script>
  {% endjavascripts %}

        {% include 'McChartBundle:Js:iframe.js.twig' with {'typeChart': 'piechart'} %}

        <script type="text/javascript">
                {% block document_ready %}
                        $('#submitbtn').click(function (e) {
                            if (e.target) {
                                $("form")[0].submit();
                            }
                        });
                {% endblock %}

                {% include 'McChartBundle:Js:createChart.js.twig' %}

                load = true;

                var typeXAxis = $('select#mc_chartbundle_chart_xAxisType');
                options.xAxis.type = getTypeAxis($(typeXAxis).val());
                options.chart.pie = true;

                /* Creation du chart */
                var chart = new Highcharts.Chart(options);
                chart.yAxis[0].remove();

                $(typeXAxis).change(function() {
                        var axis = $('#container').highcharts().xAxis[0];
                        updateTypeAxis(this, axis);
                });

                //$(pieType).change(function() {
                //      var axis = $('#container').highcharts().chart.pie;
                //      updateTypeAxis(this, axis);
                //});


                $(creditsDisplay).on('click', function(e) {
                        if (e.target) {
                    $("form").submit(function (ev) {
                                                ev.preventDefault();
                                        });
                    }
                });
```

```
                $('#container').css('width', (chart.chartWidth + 3));

                /* Ajout de la fonctionnalité drag'n drop au graphique */
                $('#container').append('<img id="drag" src="{{ asset("img/drag\'n drop.png") }}" />');
                $("#container").draggable({ handle: "#drag" });

                /* Pour chaque axes-Y et ses séries, on recupère les valeurs en base et on appel l'évènement change
pour mettre à jours le graphique */
                $(document).ready(function() {

                var pieType = $('select#mc_chartbundle_chart_pieType');
                        if ($(pieType).val() == 'pie') {
                                chart.series[0].update({ type : 'pie' })
                                chart.yAxis[0].update({ type: 'pie' });
                        } else {
                                chart.yAxis[0].update({ gridLineInterpolation: 'polygon' });
                        }


                        {% for yAxis in list_yAxis %}
                        var indexS = 1;
                                {% for serie in yAxis.series|reverse %}
                                        var containerSerie = $('div[id^=container_series_]');
                                        addOneSerie($(containerSerie).children(), containerSerie, 1);

                                        $.ajaxSetup({async: false});
                                                $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS
+ '_dataList').val('{{ serie.dataList.id }}').change();
                                                $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS
+ '_parameterDataList').val('{{ serie.parameterDataList }}').change();
                                        $.ajaxSetup({async: true});

                                        $('input#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_titleSerie').val('{{ serie.titleSerie|e('js') }}').change();
                                        $('input#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_unitSerie').val('{{ serie.unitSerie|e('js') }}').change();
                                        $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_typeSerie').val('{{ serie.typeSerie|e('js') }}').change();
                                        $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_size').val('{{ serie.size|e('js') }}').change();
                                        $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_innersize').val('{{ serie.innersize|e('js') }}').change();
                                        $('select#mc_chartbundle_chart_list_yAxis_1_series_' + indexS +
'_colorSerie').val('{{ serie.colorSerie|e('js') }}').change();
                                        indexS++;
                                {% endfor %}
                        {% endfor %}

                        {% for yAxis in list_yAxis|reverse %}
                                $('input#mc_chartbundle_chart_list_yAxis_1_titleYAxis').val('{{ yAxis.titleYAxis|e('js')
}}').change();
                                $('select#mc_chartbundle_chart_list_yAxis_1_typeYAxis').val('{{
yAxis.typeYAxis|e('js') }}').change();
                        {% endfor %}
                });

                {% include 'McChartBundle:Js:docReady.js.twig' %}

                if ($('input#mc_chartbundle_chart_invertedChart').is(':checked')) {
                        chart.inverted = true;
                        chart.xAxis[0].update({}, false);
                   chart.yAxis[0].update({}, false);
                        chart.redraw();
                } else {
                         chart.inverted = false;
                         chart.xAxis[0].update({}, true);
                         chart.yAxis[0].update({}, true);
                         chart.Series[0].update({}, true);
                        chart.redraw();
                }

                load = false;
        </script>
{% endblock %}
```

Annexe 2 : web/js/Chart/seriepie.js

```
/* Fonction permettant l'ajout d'une série */
function addOneSerie(divserie, containerserie, i) {
	/* Mise à jours et ajout du prototype */
	// console.log("container : " + containerserie);
	var divflag=containerserie.children('div.flag'); // Ne pas oublier les flags
	var yaxisOrder=containerserie.children().length + 1;
	console.log('yaxisOrder serie : ' + yaxisOrder);

	var series = $($('#prototype_series').attr('data-prototype')
		.replace(/__nameyaxis__/g, i)
		.replace(/__name__/g, yaxisOrder)
	);
	// console.log("series : " + series[0]);
	containerserie.prepend(series);

	var dataList = $(containerserie).find('select[id^=mc_chartbundle_chart_list_yAxis_' + i + '_series_][id$=_dataList]');
	var idSerie = yaxisOrder;
	var idYAxis = i;
	// var idSerie = $(dataList).parent().html().match(/<select(.*)_(\d+)_dataList/)[2];
	// var idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
	console.log('%cIdentifiant de l\'axe - n°serie : %c' + idYAxis +'-'+ idSerie, 'color: #43A832', 'color: #8085E8');

	colorSeries(idSerie,idYAxis);
	addAjaxQueryOnDataList(dataList, idSerie, idYAxis);

	var parameterDataList = $(dataList).parents('.form-group').find('select[id$=' + idSerie + '_parameterDataList]');
	var titleSerie = $(dataList).parents('.form-group').prev().find('input[id$=' + idSerie + '_titleSerie]');
	var unitSerie = $(dataList).parents('.form-group').prev().find('input[id$=' + idSerie + '_unitSerie]');
	var typeSerie = $(dataList).parents('.form-group').next().find('select[id$=' + idSerie + '_typeSerie]');
	var colorSerie = $(dataList).parents('.form-group').next().find('select[id$=' + idSerie + '_colorSerie]');
	var orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' + idSerie +
'_yaxisOrder').val(yaxisOrder);
	var invertedChart = $('input#mc_chartbundle_chart_invertedChart');
	var typestacked = $('select#mc_chartbundle_chart_typestacked');
	var size = $(containerserie).find('select#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' + idSerie +
'_size');
	var innersize = $(containerserie).find('select#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' + idSerie +
'_innersize');

	var defaultColor = $(colorSerie).val();
	var mixedColors = getColorsMixed(defaultColor);

	/* Ajout de la série au graphique */
	var chart = $('#container').highcharts();
		chart.addSeries({
			id: idSerie,
			yAxis: 'yAxis' + i,
			color: defaultColor,
			colors: mixedColors,
			type: 'pie',
			name: $(titleSerie).val(),
			stacking: $(typestacked).val(),
			data: [],
			size: $(size).val(),
			innerSize: $(innersize).val()
		});

	/* Mise à jours du graphique en fonction des valeurs des champs */
	$(parameterDataList).on('change', function() {
		console.log('%cModification du parametre de la serie pour : %c' + $(parameterDataList).val(), 'color:
#43A832', 'color: #8085E8');
		idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
		idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
		orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
		setSerie(orderSeries, $(dataList).val(), $(parameterDataList).val(),idYAxisChart); //Ajout du numéro de
Yaxis
		updateTitleAndUnitWithParameter($(parameterDataList));
	});

	$(titleSerie).on('change', function() {
		console.log('%cModification du titre de la serie pour : %c' + $(titleSerie).val(), 'color: #43A832', 'color:
```

```
#8085E8');
                        idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
                        idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                        orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
                        chart.series[(orderSeries -1)].update({
                                name: produceTitleWithUnit($(titleSerie).val(), $(unitSerie).val())
                        });
        });

        $(unitSerie).on('change', function() {
                console.log('%cModification de l\'unite de la serie pour : %c' + $(unitSerie).val(), 'color: #43A832', 'color:
#8085E8');
                idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
                idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
                chart.series[(orderSeries -1)].update({
                        name: produceTitleWithUnit($(titleSerie).val(), $(unitSerie).val())
                });
        });


        //TEST POUR LE SCATTER ET PIE
        if ($(typeSerie).val() == 'scatter') {
                // $(stacked).prop('checked', false).change();
                $(invertedChart).prop('checked', false).change();
        } else if ($(typeSerie).val() == 'bar') {
                $(invertedChart).prop('checked', true).change();
        }

        $(size).on('change', function() {
                console.log('%cModification de la taille size pour : %c' + $(this).val(), 'color: #43A832', 'color: #8085E8');
                idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
                idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
                //chart.yAxis[(idYAxisChart - 1)].series[(orderSeries -1)].update({
                chart.series[(orderSeries -1)].update({
                        size: $(this).val()
                });
        });

        $(innersize).on('change', function() {
                console.log('%cModification de la taille inner pour : %c' + $(this).val(), 'color: #43A832', 'color: #8085E8');
                idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
                idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
                chart.series[(orderSeries -1)].update({
                        innerSize: $(this).val()
                });
        });

        $(typestacked).on('change', function() {
                console.log('%cModification dempilage selon le type : %c' + $(this).val(), 'color: #43A832', 'color:
#8085E8');
                idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
                idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();
                chart.series[(orderSeries -1)].update({
                        stacking: $(this).val()
                });
        });

        $(colorSerie).on('change', function() {
                console.log('%cModification de la couleur de la serie pour : %c' + $(colorSerie).val(), 'color: #43A832',
'color: #8085E8');
                idYAxis = $(dataList).parent().html().match(/<select(.*)_(\d+)_series/)[2];
```

```
                        idYAxisChart = $('input#mc_chartbundle_chart_list_yAxis_' + idYAxis +
'_titleYAxis').parents().children('label').html().match(/(Axe-Y n° )(\d+)/)[2];
                        console.log('%cseriecolor : %c' + idYAxisChart + '-' + orderSeries, 'color: #43A832', 'color: #8085E8');
                        orderSeries = $(containerserie).find('input#mc_chartbundle_chart_list_yAxis_' + idYAxis + '_series_' +
yaxisOrder + '_yaxisOrder').val();

                        chart.series[(orderSeries -1)].update({
                                color: $(this).val(),
                                colors: getColorsMixed($(this).val())
                        });
        });

        /*      Si ce n'est pas la première série ajouté, on implémente la nouvelle de la même requête avec le paramètre
suivant */
        var prevDatalist = $('select[id^=mc_chartbundle_chart_list_yAxis_][id$=_series_' + (idSerie -1) + '_dataList]').val();
        var prevParam = $('select[id^=mc_chartbundle_chart_list_yAxis_][id$=_series_' + (idSerie -1) +
'_parameterDataList]').val();

        if (typeof prevDatalist !== 'undefined' && prevDatalist != '') {
                console.log('%cDatalist de la serie precedente : %c' + prevDatalist, 'color: #43A832', 'color: #8085E8');
                console.log('%cParametre de la serie precedente : %c' + prevParam, 'color: #43A832', 'color: #8085E8');
                $('select[id^=mc_chartbundle_chart_list_yAxis_][id$=_series_' + idSerie +
'_dataList]').val(prevDatalist).change();
        }
}
```

Annexe 3 : web/js/color.js

```
/* Couleurs primaires */
var colorMix = [
    '#000000'    // Black
    ,'#FF0000'    // Red
    ,'#00FF00'    // Green
    ,'#0000FF'    // Blue
    ,'#FFFF00'    // Yellow
    ,'#FFFFFF'    // White
    ,'#8800FF'    // Purple
    ,'#FF8800'    // Orange
    ,'#FF00FF'    // Pink
];

/* Liste de couleurs sélectionnés
    Valeurs identiques à ./managechart/src/Mc/ChartBundle/AvailableChoice/AvailableColorSerie.php */
var colorAvailable = [
    '#3399CC' // Blue
    ,'#65BFE4'
    ,'#006699'
    ,'#1A7BB8'
    ,'#D4E6F2'
    ,'#007D83' // Green
    ,'#2AA8B0'
    ,'#A0D8DC'
    ,'#228527'
    ,'#3BA737'
    ,'#BAE0B9'
    ,'#799B13'
    ,'#9CC11B'
    ,'#CBDE88'
    ,'#E0B100' // Orange
    ,'#FCCA12'
    ,'#FEE58A'
    ,'#D68300'
    ,'#F19607'
    ,'#F9D397'
    ,'#BB431B' // Red
    ,'#E65321'
    ,'#F4B39D'
    ,'#B82222'
    ,'#E53336'
    ,'#F19395' // Pink
    ,'#B80065'
    ,'#E1037C'
    ,'#F49ECC'
    ,'#6E1E68' // Purple
    ,'#9E5298'
    ,'#D9BCD7'
    ,'#5A4E40' // Brown
    ,'#968A7A'
    ,'#DBD6D1' // Light Gray
    ,'#00274A' // Dark Blue
    ,'#656565'     // Dark Gray
];

/* Convertit une chaîne de caractère en couleur */
function componentToHex(c) {
            var hex = c.toString(16);
            return hex.length == 1 ? "0" + hex : hex;
}

/* Convertit une chaîne de caractère RGB en couleur Hex */
function rgbToHex(r, g, b) {
            return "#" + componentToHex(r) + componentToHex(g) + componentToHex(b);
}

/* Fonction REGEX convertisant une couleur rgb(r,g,b) en tableau */
function parseRGB(color){
    var matchColors = /^rgb\((0|255|25[0-4]|2[0-4]\d|1\d\d|0?\d?\d),(0|255|25[0-4]|2[0-4]\d|1\d\d|0?\d?\d),(0|255|25[0-4]|2[0-
4]\d|1\d\d|0?\d?\d)\)$/
    var match = matchColors.exec(color);
    if( match != null){
            return [match[1],match[2],match[3]];
```

```
    }else
            return null;
}

/* Fonction permettant de mixer une couleur de base avec une couleur prédéfini */
function colorMixing(baseColor, mixColor){
    var rgbBaseColor = parseRGB(baseColor);
    var rgbMixColor = parseRGB(mixColor);

    var red = Math.floor((parseInt(rgbBaseColor[0])+parseInt(rgbMixColor[0]))/2);
    var green =  Math.floor((parseInt(rgbBaseColor[1])+parseInt(rgbMixColor[1]))/2);
    var blue =  Math.floor((parseInt(rgbBaseColor[2])+parseInt(rgbMixColor[2]))/2);
    var hexColor = rgbToHex(red,green,blue);

    return hexColor;
}

/*  Fonction créant un tableau de couleur HEX, résultat du mix entre une couleur de base et de couleurs prédéfini */
function getColorsMixed(baseColor){
    var graphColors = [];
    graphColors.push(baseColor);

    for(var mix = 0; mix < colorMix.length;mix++){
            var hexColor = colorMixing(Highcharts.Color(baseColor).get(), Highcharts.Color(colorMix[mix]).get());
            graphColors.push(hexColor);
    }

    return graphColors;
}

/* Fonction créant un tableau de couleur HEX, résultat d'un algorithme de sélection entre la couleur de base, le nombre
d'entité de la série et le tableau de couleurs à disposition */
function getColorsShuffled(baseColor, numberItems){
    var graphColors = [];
    graphColors.push(baseColor);

    if((numberItems != null) && (numberItems > 0)){
            var startingIndex = colorAvailable.indexOf(baseColor);
            var currentIndex = startingIndex;
            var step = 1;

            if(numberItems < colorAvailable.length)
                    step = Math.floor(colorAvailable.length / numberItems);

            for(var shuffle = 1; shuffle < numberItems; shuffle++){
                    currentIndex = (currentIndex + step)%((colorAvailable.length) - 1);
                    graphColors.push(colorAvailable[currentIndex]);
            }
    }
    return graphColors;
}
```

Annexe 4 : src/Mc/ChartBundle/Resources/view/Chart/showiframe.html.twig

```
…
// ASFLAG : Fix inversion des axes
{% if chart.invertedChart == false %}
		top: '{{ yAxis.top|e('js') }}',
		height: '{{ yAxis.height|e('js') }}'
{% else %}
		left: '{{ yAxis.top|e('js') }}',
		width: '{{ yAxis.height|e('js') }}'
{% endif %}
…
```

Annexe 5 : web/js/Chart/heatmapscript.js

```
var colorJet = [
		[0, '#0000f8'],
		[0.15, '#0000ff'],
		[0.3, '#008fff'],
		[0.4, '#00ffff'],
		[0.5, '#00ff00'],
		[0.6, '#ffff00'],
		[0.7, '#ff8f00'],
		[0.85, '#ff0000'],
		[1, '#8f0000']
];
function getLimitValues(data){
		var limitValues = {
		'minXAxis': null,
		'maxXAxis': null,
		'minYAxis': null,
		'maxYAxis': null,
		'minValue': null,
		'maxValue': null,
		'uniqueXAxis': null
		};
   limitValues.length = 7;
   var unique = [];
   var firstValue = true;
   data.forEach(function(element){
		var xElement = element['x'];
			var yElement = element['y'];
			var valueElement = element['value'];
			if(firstValue){
				firstValue = false;
				limitValues['minXAxis'] = xElement;
				limitValues['maxXAxis'] = xElement;
			limitValues['minYAxis'] = yElement;
			limitValues['maxYAxis'] = yElement;
			limitValues['minValue'] = valueElement;
			limitValues['maxValue'] = valueElement;
			unique = [xElement];
			}else{
			if(limitValues['minXAxis'] > xElement)
				limitValues['minXAxis'] = xElement;
			else if(limitValues['maxXAxis'] < xElement)
				limitValues['maxXAxis'] = xElement;
			if(!unique.includes(xElement))
				unique.push(xElement);

			if(limitValues['minYAxis'] > yElement)
				limitValues['minYAxis'] = yElement;
			else if(limitValues['maxYAxis'] < yElement)
				limitValues['maxYAxis'] = yElement;

			if(limitValues['minValue'] > valueElement)
				limitValues['minValue'] = valueElement;
			else if(limitValues['maxValue'] < valueElement)
				limitValues['maxValue'] = valueElement;
			}
		});
		limitValues['uniqueXAxis'] = unique.length;

		return limitValues;}
```

Annexe 6 : src/Mc/ChartBundle/Resources/views/Js/iframe.js.twig

```
…
		data.forEach( function(element, index) {
		var elem=element[0];

		if(typeof element[1] === 'string'){
		var y=element[1].split(",");
		y=y[0].replace("{","");
		var value=element[1].split(",");
		value=value[1].replace("}","");

		if (element[1]==''){element[1]='0'};

		// ASFLAG : Format des données selon son typeSerie
		if(typeSerie == 'heatmap'){
		if (index == data.length-1) {
		data_update+= "{"+
				'"x":'+ element[0]+','+
				'"y":'+ y +','+
				'"value":'+ value +
				'}';
		}
		else {
		data_update+= "{"+
				'"x":'+ element[0]+','+
				'"y":'+ y +','+
				'"value":'+ value +
				'},';
		}
		}else if(typeSerie == 'errorbar'){
		if (index == data.length-1) {
		data_update+= "{"+
				'"x":'+ element[0]+','+
				'"low":'+ y +','+
				'"high":'+ value +
				'}';
		}
		else {
		data_update+= "{"+
				'"x":'+ element[0]+','+
				'"low":'+ y +','+
				'"high":'+ value +
				'},';
		}
		}
		}});
		data_update+=']';
…
```