

RAPPORT DE STAGE

THEME DU STAGE : Développement de nouvelles fonctionnalités de l'application **ManageChart**.

<https://www-iuem.univ-brest.fr/wapps/managechart/>



M1 TIIL 2016/2017

Table des matières

Remerciements	4
Introduction.....	5
1. Présentation du laboratoire Letg-Brest Géomer.....	6
2. L'IUEM	6
3. Sujet du stage	6
4. Projet ROZA	6
1. Contexte scientifique et objectifs.....	6
2. Méthodes	6
5. Analyse des spécifications fonctionnelles	7
1. L'application ManageChart	7
2. Analyse du cahier des charges	7
6. Présentation de l'application ManageChart (Aspect fonctionnel).....	8
1. L'existant	8
2. Utilisateur et droits	8
3. Base de données	11
4. Requêtes.....	12
5. Graphiques	13
7. Architecture de l'application ManageChart (Modélisation fonctionnel)	16
1. Architecture de l'application sous le modèle MVC.....	16
2. Architecture général de l'application ManageChart modèle et package	20
3. Dictionnaire des données.....	21
4. Architecture de la base de donnée	23
1. Contraintes d'intégrité	24
2. Dépendances fonctionnelles	24
8. Présentation du Framework Symfony 2 MVC (Model, Views, Controller).....	25
1. Architecture du Framework	25
9. Partie conception	26
1. Diagramme de cas d'utilisation	26
2. Diagramme de séquence.....	27
3. Diagramme de composants.....	28
10. Partie développement (Aspect technique)	29
1. Bogue	30

	3
2. Quelques modifications :.....	34
3. Fonctionnalités.....	34
11. Modélisation techniques.....	40
1. Choix et outils techniques.....	40
2. Framework et librairies.....	41
3. Mise à jour.....	41
4. Tests.....	41
5. Dépôt GIT.....	42
6. Poursuite du stage en Juillet et Aout.....	42
Conclusion.....	44
Bibliographie.....	44

Remerciements

Je tiens à remercier M. Mathias **Rouan** par son accueil chaleureux au sein du laboratoire Letg à BREST, sa bonne humeur et son intérêt pour les stagiaires au sein du laboratoire ainsi que M. Jean.-Phillip. **Babau** par son suivi des différentes phases du projet.

D'une façon plus générale, je remercie l'ensemble des personnes au laboratoire Letg pour l'intérêt qu'ils m'ont porté tout au long de mon stage ainsi que leurs aides et précisions.

Introduction

Ce rapport présente le travail que j'ai effectué lors de mon stage au sein du laboratoire Letg à L' IUEM, qui se déroule **du 3 avril au 31 aout 2017**, Pendant la première période du stage je me suis familiarisé avec le Framework Symfony2 basé en PHP pour comprendre l'application existante.

Ce stage d'une durée de 5 mois va me permettre d'avoir des connaissances en programmation dans le développement de nouvelles fonctionnalités d'une application existante.

ManageChart est une application de visualisation des données selon des formes graphiques, ce projet rentre dans le cadre du projet **ROZA**.

Cette application s'est avérée très intéressante et très enrichissante pour mon expérience professionnelle puisque c'est un projet basé sur un Framework parmi les meilleurs utilisés et c'est le langage clé pour mon alternance en septembre 2017.

Le but de ce rapport n'est pas de faire uniquement une présentation exhaustive de tous les aspects techniques que j'ai pu apprendre ou approfondir mais aussi de manière systématique et claire de faire un tour d'horizon sur les aspects scientifiques des zones ateliers.

Je vous expose dans ce rapport en premier lieu une présentation du laboratoire Letg, le sujet du stage ainsi que le projet Roza.

Ensuite, les différents aspects de l'application existante et de mon travail durant ces mois et pour finir une conclusion qui résume les apports de ce stage.

1. Présentation du laboratoire Letg-Brest Géomer

Letg Brest Géomer est un des 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

2. L'IUEM

L'IUEM est l'institut de recherche dédié à l'océan et au littoral dont les objectifs sont les suivants :

- + Accroître la connaissance du monde marin
- + Etudier et observer les interactions de ce monde marin avec l'atmosphère et les espaces continentaux
- + Former des chercheurs et cadres dans ces domaines

3. Sujet du stage

Développer des nouvelles fonctionnalités de l'application ManageChart et mise en place d'un prototype de plateforme.

L'application s'inscrit dans les évolutions de l'Infrastructure de Données Spatiales (IDS) Indigéo qui 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 géo spatialisées.

ManageChart permettra la création de graphiques paramétrables à partir de sources de données différentes. Ces graphiques seront visualisables dans une application externe telle qu'Indigéo qui pourra l'afficher dans une iframe avec son URL.

4. Projet ROZA

1. Contexte scientifique et objectifs

ROZA est un projet fédératif pluriannuel. Il vise, via la mise en place d'une infrastructure numérique nationale, à favoriser la réutilisation et la méta-analyse des données et échantillons issus d'archives naturelles étudiées au sein des Zone Ateliers, au-delà de la communauté des paléo sciences.

ROZA a pour ambition de constituer une banque d'échantillons et de données géo-référencées (métadonnées de collecte), chrono référencés (datation) et transposables (réutilisation des données acquises sur de nouveaux prélèvements) sans équivalent de par le monde.

Les moyens ont été affectés à du réseautage et de l'analyse pour homogénéiser les séquences, ainsi qu'à la constitution des prémices d'un parc de modules d'acquisition de données de terrain (tablettes, imprimantes labo ou mobiles).

Les attendus sont une augmentation notable de la réutilisation des données issues des Zone ateliers et l'émergence de projets inédits de méta analyse.

Loin d'être un « trou noir » de donnée, isolé et donc peu visible, Roza s'appuie sur des infrastructures existantes comme la Cyber-carothèque nationale (C2FN) et anticipe la mise en place d'une infrastructure de données géographiques des Zones Ateliers.

ROZA pourra par ailleurs être étendu à l'échelle européenne via l'interopérabilité vers le système d'information ILTER DEIMS.

2. Méthodes

ROZA 1 (2015) avait permis à chaque ZA d'organiser la collecte des métadonnées sur les échantillons physiques existants et de sélectionner les séquences sédimentaires de référence. Le projet s'était achevé par une réunion visant à sélectionner collectivement les données obligatoires et optionnelles à conserver.

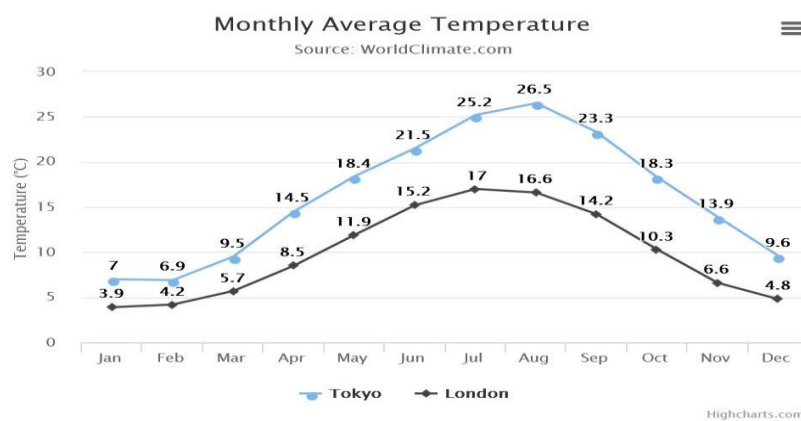
ROZA 2 (2016) a poursuivi le travail d'uniformisation des séries de données issues de carottages à partir de réunions en groupes de travail. Un portail web de catalogage, visualisation (représentations graphiques) et mise à disposition (téléchargement) des données issues des carottes de référence a été prototypé sur l'IDG. Des analyses du signal géochimique (XRF core scanner) sont en cours actuellement sur les carottes de référence afin de finaliser l'uniformisation des jeux de données entre les différentes ZA.

5. Analyse des spécifications fonctionnelles

1. L'application ManageChart

L'application « ManageChart » est une application web permettant une représentation de données scientifiques sous la forme de graphique.

Exemple de graphiques possible:



SimpleChart With data labels

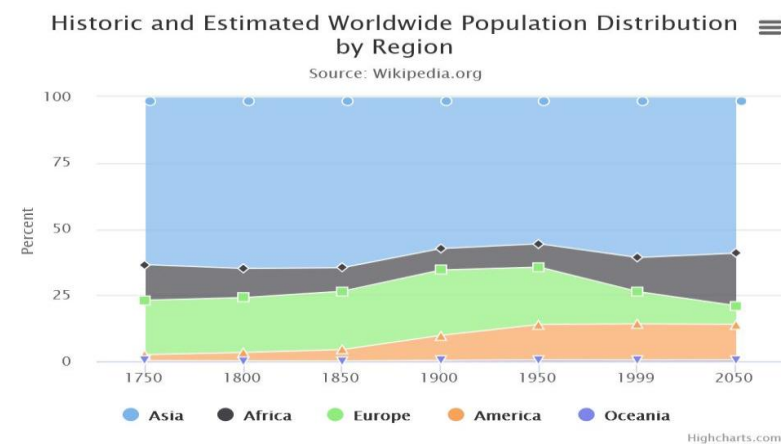


Chart with type area

Au début du stage, l'application est déjà construite et opérationnelle. Elle permet de réaliser des graphiques à partir de bases de données.

➔ Les différents types de graphique disponible sont :

- Graphique simple
- Graphique temporelle
- Graphique temporelle multi axes
- Graphique dynamique
- Graphique polaire

2. Analyse du cahier des charges

L'objectif de ce stage est de permettre d'enrichir l'application ManageChart en explorant d'autres modes de représentations graphique utilisables dans le cadre de ce projet.

Après discussion entre les différents intervenants, il est convenu d'intégrer à l'application plusieurs nouvelles représentations graphiques selon le besoin du projet ROZA

- + Mise à jour des librairies et dépendances,
- + Correction de bogues,
- + Amélioration des interfaces,
- + Ajout de nouvelles fonctionnalités,
- + Ajout de nouveaux types de graphes.

6. Présentation de l'application ManageChart (Aspect fonctionnel)

1. L'existant

2. Utilisateur et droits

La gestion des utilisateurs est effectuée par le bundle « FOSUserBundle » qui intègre tout un système de gestion d'utilisateur. Après une configuration minimale, il peut être directement utilisable. Cependant, il est tout à fait possible de surcharger les formulaires, les vues, les traductions...

Ce bundle permet l'inscription de nouveaux utilisateurs (avec la possibilité de générer un email de confirmation), l'édition et la suppression d'utilisateurs ainsi que le changement de mot de passe. Ici, le formulaire d'enregistrement a été volontairement attribué aux administrateurs, ce qui signifie qu'un nouvel utilisateur doit forcément demander à un administrateur de lui créer un compte.

La gestion des droits est définie dans Symfony2 dans un fichier décrivant tous les paramètres de sécurité de l'application. C'est donc dans ce fichier que l'on définit les rôles et où l'on définit les droits d'accès pour chaque rôle. Dans l'application « ManageChart » sont définis les rôles : « administrateur » et « scientifique ». Une route est ensuite associée à chaque rôle.

Actions	id	Nom	Rôle
	12	christophe	ROLE_SCIENTIFIC
	11	tbouinot	ROLE_ADMIN
	10	scientifique	ROLE_SCIENTIFIC
	9	scientificplus	ROLE_SCIENTIFIC_PLUS
	0	admin	ROLE_ADMIN

Figure 1 : L'interface de gestion d'utilisateur

L'interface permet la gestion des utilisateurs qui peuvent accéder à l'application ManageChart selon un rôle spécifique à chaque utilisateur pouvant lui donner le droit ou non de consulter différents onglets de l'application.

Les différents rôles sont : Role_scientifique, Role_Admin, Role_scientifique_plus.

Cette application est constituée d'une barre de menu comportant les principales fonctionnalités.

Cette barre de menu est différente selon le profil d'utilisateur :





















- Session Scientifique => Ajout du menu graphique
- Session Scientifique plus => Ajout du menu requête
- Session Administrateur => Ajout du menu base de données et utilisateur

Session invité : L'utilisateur « invité » peut visualiser la liste des graphiques et récupérer les URL.

id	Nom	Type	URL pour l'iframe
196	groupe travail	Classique	http://localhost/ManageChar
173	test2	Classique	http://localhost/ManageChar
175	test4	Classique	http://localhost/ManageChar
176	test5	Classique	http://localhost/ManageChar
177	test12	Temporel	http://localhost/ManageChar
178	test6	Classique	http://localhost/ManageChar
179	test7	Classique	http://localhost/ManageChar
172	test1	Classique	http://localhost/ManageChar
206	testExportsv	Classique	http://localhost/ManageChar
209	testurl	Classique	http://localhost/ManageChar
211	testparamimage	Classique	http://localhost/ManageChar
212	testmultiaxeY	Temporel	http://localhost/ManageChar
171	test11	Temporel	http://localhost/ManageChar
219	testmultiaxeY2	Temporel	http://localhost/ManageChar
220	testmultiaxeY3	Temporel	http://localhost/ManageChar

Figure 2 : Interface des listes de graphes avec leurs URL

Session Administrateur : L'utilisateur avec le rôle « administrateur » a les droits de lecture, de création, d'édition et de suppression sur les graphiques mais également sur les connexions à des bases de données, sur les requêtes et sur les utilisateurs. Il peut éditer son profil utilisateur.

ManageChart						Bases de données	Requêtes SQL	Graphiques	Utilisateurs	admin
Graphiques						Nouveau graphique				
Nb d'entrée						10	Recherche			
Actions	id	Nom	Type	URL						
 	73	test Nouveau graphique	Classique	http://localhost/ManageChart/web/app_dev.php						
 	72	test flag 1	Classique	http://localhost/ManageChart/web/app_dev.php						
 	69	test bien	Classique	http://localhost/ManageChart/web/app_dev.php						
 	68	test 1axe	Classique	http://localhost/ManageChart/web/app_dev.php						
 	67	test bug cc	Classique	http://localhost/ManageChart/web/app_dev.php						
 	66	test 1serie	Classique	http://localhost/ManageChart/web/app_dev.php						
 	65	sdf	Classique	http://localhost/ManageChart/web/app_dev.php						
 	64	test record 2 axes	Classique	http://localhost/ManageChart/web/app_dev.php						
 	63	test new	Classique	http://localhost/ManageChart/web/app_dev.php						
 	62	ss	Classique	http://localhost/ManageChart/web/app_dev.php						

Page 14 sur 18
ManageChart 2014-2016 LETG-Brest Géomer

Figure 3 : L'interface de gestion graphique (Administrateur)

Ajout d'un utilisateur

Dans l'existant, l'application permet d'ajouter un nouvel utilisateur avec son mot de passe et lui affecter un rôle selon son métier et son besoin.

ManageChart Bases de données Requêtes SQL Graphiques Utilisateurs admin

Enregistrer un nouvel utilisateur

Nom d'utilisateur

Adresse e-mail

Mot de passe

Répéter le mot de passe

Rôles
Admin
Scientific+
Scientific

Enregistrer

ManageChart 2014-2016 LETG-Brest Géomer

Figure 4 : L'interface pour un nouvel utilisateur

Authentification

L'authentification va permettre à l'utilisateur de s'authentifier pour accéder au contenu de l'application.

ManageChart

localhost/ManageChart/web/app_dev.php/tr/login

ManageChart Connexion

Connexion

Nom d'utilisateur

Mot de passe

Se souvenir de moi

Connexion

[Mot de passe oublié ?](#)

ManageChart 2014-2016 LETG-Brest Géomer

Figure 5 : L'interface pour l'authentification

Après être authentifié, un affichage des menus selon le rôle de chaque utilisateur, l'administrateur à la possibilité d'accéder à tout les onglets de l'application.

3. Base de données

Cette interface est une gestion des bases récupérant des données permettant la génération des graphes paramétrés selon des requêtes précises.

The screenshot shows the 'Bases de données' (Databases) management interface. At the top, there is a navigation bar with 'ManageChart', 'Bases de données', 'Requêtes SQL', 'Graphiques', and 'Utilisateurs'. A user dropdown menu shows 'admin'. Below the navigation bar, the title 'Bases de données' is displayed next to a 'Nouveau' button. A search bar labeled 'Recherche' is present. A table lists the database connections with columns for Actions, id, Nom de connexion, Nom Bdd, Description, Type, and Date. The table contains six entries. At the bottom, there is a pagination control showing 'Page 1 sur 1' and a footer 'ManageChart 2014-2016 LETG-Brest Géomer'.

Actions	id	Nom de connexion	Nom Bdd	Description	Type	Date
	6	clapot postgres9.3	adviclim_gama	clapot postgres9.3	PostgreSQL	2016/11/22
	5	clapot postgres8.4	SEVE	base de données simulations SEVE	PostgreSQL	2016/11/15
	4	Geosu	geosu	geosu postgresql sur 7uem68:5433	PostgreSQL	2016/10/17
	3	managechart sur mysql	managechart	svghzgurformz	MySQL	2016/07/08
	2	Somlit	somlit	Base Somlit IUEM	PostgreSQL	2016/07/01
	1	Base de test testData	testdata	Base de données comportant des tables de test	PostgreSQL	2016/06/15

Figure 6 : L'interface de gestion des bases de données

Ce formulaire concerne la création et l'édition d'une connexion à une base de données selon son type accessible par un « administrateur ».

The screenshot shows the 'Nouvelle connexion' (New connection) form. It features a navigation bar at the top with 'ManageChart', 'Bases de données', 'Requêtes SQL', 'Graphiques', and 'Utilisateurs'. The user dropdown menu shows 'admin'. The title 'Nouvelle connexion' is centered. Below the title, there are several input fields: 'Nom de connexion', 'Nom de Bdd', 'Type' (a dropdown menu currently showing 'PostgreSQL'), 'Description', 'Hôte', 'Port', 'Identifiant', and 'Mot de passe'. At the bottom of the form, there are two buttons: 'Test de connexion' and 'Enregistrer'. A footer 'ManageChart 2014-2016 LETG-Brest Géomer' is visible at the bottom.

Figure 7 : L'interface pour une nouvelle connexion

4. Requêtes

Cette interface est une gestion des requêtes paramétrés et alimentés par des bases de données précédemment ajouté afin de récupérer les données voulus pour la génération des graphes.

Actions	id	Nom	Base de données	Date
	17	TraonBihan - Statistiques PLU (Test Donut)	Geosu - geosu - PostgreSQL	2017/05/23
	16	zabri - données météo horaires	Geosu - geosu - PostgreSQL	2017/05/23
	14	test1	Base de test testData - testdata - PostgreSQL	2017/04/27
	13	test logarithmique	Geosu - geosu - PostgreSQL	2017/04/11
	12	stades flag	clapot postgres9.3 - adviclim_gama - PostgreSQL	2016/11/25
	10	météo tinytag tasmin 2-6	clapot postgres9.3 - adviclim_gama - PostgreSQL	2016/11/22
	9	indice bioclim	clapot postgres9.3 - adviclim_gama - PostgreSQL	2016/11/15
	8	actions flag	clapot postgres9.3 - adviclim_gama - PostgreSQL	2016/11/15
	7	somlit light	Somlit - somlit - PostgreSQL	2016/10/20
	6	test guillemet	Geosu - geosu - PostgreSQL	2016/10/17

Page 1 sur 2
ManageChart 2014-2016 LETG-Brest Géomer

Figure 8 : L'interface pour Requêtes SQL

Ce formulaire concerne la création et l'édition d'une requête. Il est indispensable de sélectionner une connexion à une base de données. Il est également possible d'ajouter dynamiquement la clause « WHERE » de la requête.

ManageChart Bases de données Requêtes SQL Graphiques Utilisateurs admin

Nouvelle requête

Nom

BDD

Requête

Veiller à respecter le format suivant :

- 1er champ : x
- 2nd champ : y
- 3eme champ : nom du parametre y
- 4eme champ : unite du parametre en y (optionnel)

Toutes les valeurs d'un même champ d'une série données doivent être de même type (numérique ou chaîne de caractère).

La valeur de date doit être un **UNIX TIMESTAMP**

- PostgreSQL : EXTRACT(EPOCH FROM '2007-11-30 10:30:19')
- MySQL : UNIX_TIMESTAMP('2007-11-30 10:30:19')

Attention ! Les graphiques attendent des millisecondes. Il est nécessaire d'ajouter un facteur 1000.
[JavaScript Date class](#)

ManageChart 2014-2016 LETG-Brest Géomer

Figure 9 : L'interface pour une nouvelle Requête

5. Graphiques

La première fenêtre sert à lister les différents graphes ajoutés.

La deuxième fenêtre permet l'ajout d'un graphe selon le choix du type.

The screenshot shows the ManageChart application interface. At the top, there is a navigation bar with "ManageChart", "Requêtes SQL", and "Graphiques". A user profile "said-souilem@live.fr" is visible in the top right. Below the navigation bar, the title "Graphiques" is displayed next to a "Nouveau graphique" button. A search bar labeled "Recherche" is present. Below the search bar, there is a table listing various charts. The table has columns for "Actions", "id", "Nom", "Type", and "URL". Each row contains a blue pencil icon for editing and a red trash icon for deleting. The table lists 10 entries, including "geobs - accessibilité des données 2016-2017", "geobs - volume de métadonnées 2016-2017", "test météo", and others. At the bottom of the table, there is a pagination control showing "Page 1 sur 7" and a set of navigation buttons (1, 2, 3, 4, 5, 6, 7).

Actions	id	Nom	Type	URL
[Pencil] [Trash]	226	geobs - accessibilité des données 2016-2017	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	225	geobs - volume de métadonnées 2016-2017	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	224	test météo	Temporel	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	213	geobs-test-jade-1004-R6	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	212	traonBihan - surface vocation	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	211	Superposition des profils	Dynamique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	203	geobs-test-jade-1004-R4	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	202	geobs-test-jade-1004-R5	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	201	geobs_test_jade_GT&PERIM	Classique	http://www-ueem.univ-brest.fr/vapps/
[Pencil] [Trash]	199	test_jade_1703	Classique	http://www-ueem.univ-brest.fr/vapps/

Page 1 sur 7
ManageChart 2014-2016 LETG-Brest Géomer

The screenshot shows a dialog box titled "Sélectionner un type de graphique". It displays five different chart types with their respective preview images and labels:

- Graphique simple**: A line chart showing monthly average temperature with multiple data series.
- Graphique temporel**: A line chart showing a time series of data.
- Graphique temporel multi-axes**: A line chart with multiple y-axes showing time series data.
- Graphique dynamique**: A line chart showing a dynamic time series.
- Graphique Polar/Spiderweb**: A polar chart showing data points connected by lines.

Lors du choix du Type de graphe, un formulaire est mis en place pour afficher les caractéristiques du graphe en fonction des paramètres à lui attribuer pour sa création et son enregistrement.

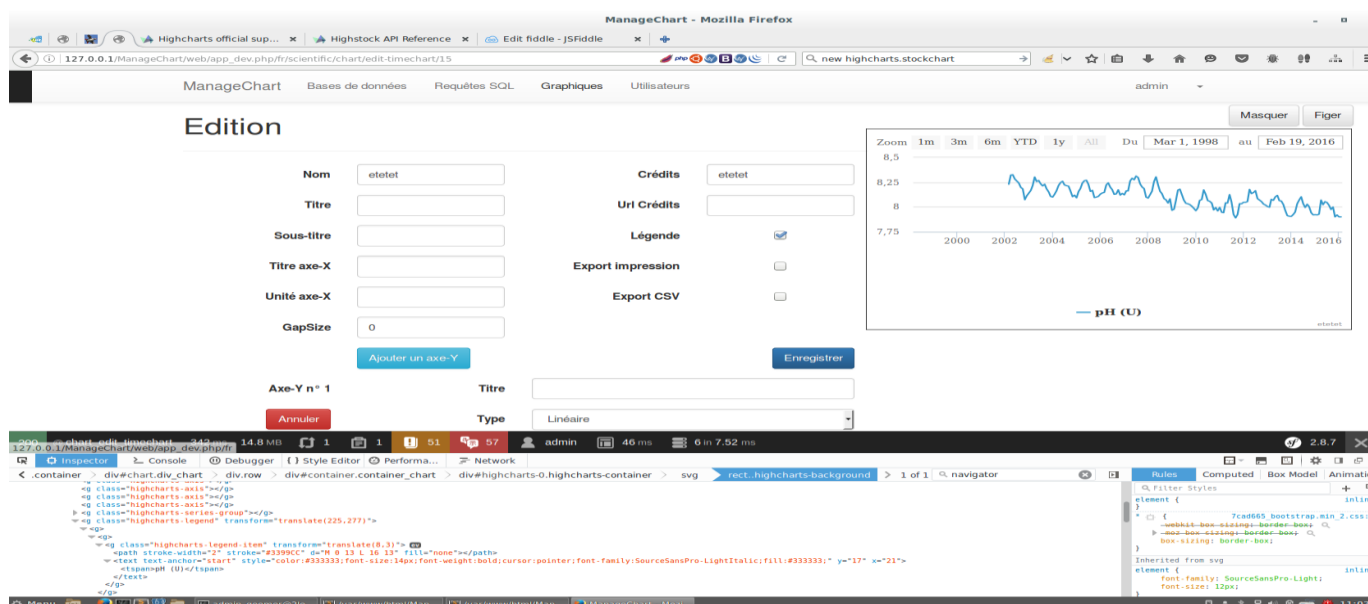


Figure10 : Interface pour l'édition d'un graphique

Ce formulaire concerne la création et l'édition d'un graphique. On définit ici les caractéristiques et options générale du graphique. Il est ensuite possible, d'ajouter un axe Y au graphique en cliquant sur le bouton « **Ajouter un axe Y** » faisant ainsi apparaître un formulaire imbriqué. (Voir ci-dessous).

Ce formulaire est accessible à un « scientifique ».

Figure 11 : Formulaire imbriqué de l'ajout d'un axe y

Ce formulaire imbriqué permet l'ajout d'un axe Y au graphique. Il est possible d'ajouter autant d'axe Y que souhaité. Ce formulaire imbriqué est différent dans le cas d'un graphique temporel. En effet, celui-ci ne peut avoir qu'un seul axe Y et son type est déjà connu. De ce fait, seul le titre de l'axe Y est éditable. Dans tous les cas, le bouton « **Ajouter une série** » apparaît, qui au clic, affiche un second formulaire permettant l'ajout d'une série de données à notre axe Y, Voir ci-dessous

Série n° 1 Annuler

Titre

Requête

Type

Marqueur

Unité

Paramètre

Couleur

Style de ligne

Figure 12 : Formulaire imbriqué de l'ajout d'une série

Lors de la création ou de la modification d'un graphique, celui-ci doit pouvoir être visualisé et rafraîchi à chaque modification de paramètre. Si une requête SQL avec paramètres est choisie, des champs de saisie doivent être ajoutés au formulaire afin de renseigner les paramètres de sélection de l'objet.

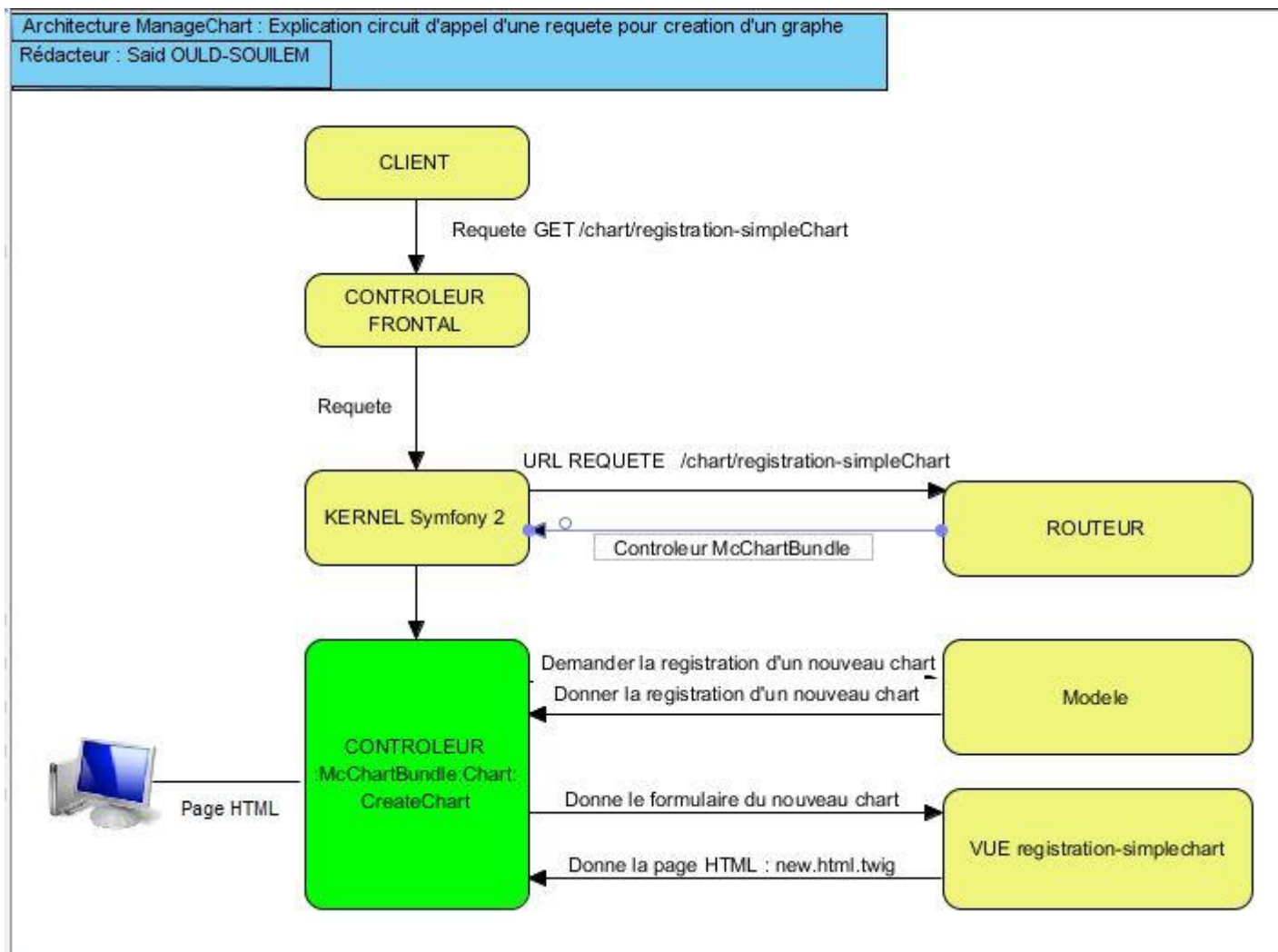
Cela permettra de visualiser son jeu de données dans le graphique et le temps de sa création. Par la suite ces paramètres devront être renseignés au travers de l'URL qui appellera ce graphique. Ces mêmes champs pourront être rajoutés si besoin dans l'interface de visualisation du graphique

Ici on retrouve une capture d'écran pour vous montrer l'appelle des requêtes pour la génération du graphe:

The screenshot shows a web interface for creating a chart. On the left, there is a form for 'Série n° 1' with fields for 'Titre', 'Requête', 'Type', 'Unité', 'Paramètre', 'Couleur', and 'Style de ligne'. A red box highlights the 'Titre' (toto), 'Requête' (Requête de test SEI), and 'Paramètre' (toto@1) fields. To the right, a line chart displays a single data series named 'toto (1)' with values ranging from approximately 100k to 175k over time (x-axis from 23 to 29). The chart is titled 'toto (1)' and has a y-axis labeled 'TotalObservation'.

7. Architecture de l'application ManageChart (Modélisation fonctionnel)

1. Architecture de l'application sous le modèle MVC



Le visiteur demande la page /chart/registration-simpleChart

Le contrôleur frontal reçoit la requête, charge le Kernel et la lui transmet

Le Kernel demande au Routeur quel contrôleur exécuter pour l'URL /chart/registration-simpleChart.

Ce Routeur est un composant Symfony2 qui fait la correspondance entre URL et contrôleurs.

Le Routeur fait donc son travail, et dit au Kernel qu'il faut exécuter le contrôleur **McCartBundle: Chart: CreateChart**

Le Kernel exécute donc ce contrôleur. Le contrôleur demande au modèle Chart la registration d'un nouveau graphe, puis la donne à la vue **registration-simpleChart** pour qu'elle construise la page HTML et la lui retourne. Une fois cela fini, le contrôleur envoie au visiteur la page HTML complète.

L'arborescence de Symfony se présente comme ceci sur l'application ManageChart, il y a 4 répertoires importants :

App/ → Ce répertoire contient tout les informations concernant la configuration de l'application, les fichiers logs, le cache, et le fichier routing.yml.

Src/ → Ce répertoire contient toutes les informations concernant le code source, une organisation des bundles séparés

Listes des bundles → BddBundle , ChartBundle , DataListBundle , DataSourceBundle , FormBuilder , EncryptBundle , UserBundle

Vendor/ → Ce répertoire contient toutes les bibliothèques externes à notre application.

Web/ → Ce répertoire contient tous les fichiers destinés à nos visiteurs, Image, Fichier CSS, JavaScript, Bibliothèque Highcharts, Highstock, on y trouve aussi le contrôleur frontal App.php,

Le contrôleur frontal c'est le point d'entrée de l'application, c'est le fichier par lequel passent tout les pages comme le principe d'index.php , ce contrôleur se situe dans le répertoire web il s'agit de **App.php** ou **App_dev.php**.

+ La différence entre le fichier App.php et App_dev.php ?

L'objectif de symfony c'est de faciliter le développement et le temps d'exécution d'une requête et propose un contrôleur frontal pour les visiteurs « App.php » et un contrôleur frontal lorsqu'on développe « App_dev.php ». Il s'agit de deux environnements de travail.

L'objectif est de répondre au mieux suivant la personne qui visite l'application :

Un développeur a besoin d'informations sur la page afin de l'aider à développer. En cas d'erreur, il veut tous les détails pour pouvoir déboguer facilement. Il n'a pas besoin de rapidité.

Un visiteur normal n'a pas besoin d'informations particulières sur la page. En cas d'erreur, l'origine de celle-ci ne l'intéresse pas du tout, il veut juste retourner d'où il vient. Par contre, il veut que le site soit le plus rapide possible à charger.

L'environnement de développement, appelé « dev », accessible en utilisant le contrôleur frontal app_dev.php. C'est l'environnement que l'on utilisera *toujours* pour développer.

L'environnement de production, appelé « prod », accessible en utilisant le contrôleur frontal app.php.

+ Explication du fonctionnement et du rôle de chaque bundles du répertoire Src/ :

ChartBundle → Gère la création et la modification des types de graphes (Dynamicchart, Multiaxistimechart, Polarchart, Simplechart, Timechart).

DataListBundle → Gère les requêtes SQL qui seront exécutées pour récupérer les données des bases de données, il est réservé aux administrateurs.

DataSourceBundle → Contient les informations sur (hote, port, nom de la base, login, password) pour se connecter aux bases de données distantes

EncryptBundle → qui crypte et décrypte des chaînes de caractères avec la méthode du OU-exclusif. Cette méthode a été choisie pour la simplicité d'implémentation. Il est cependant envisagé de migrer sur la méthode AES256 avec un bundle existant. EncryptBundle est utilisé pour crypter les identifiants et mots de passe des comptes des bases de données d'où sont issues les données des graphiques. Ces identifiants et mots de passe sont stockés en base de données et ont donc besoin d'être cryptés. Ils aussi besoin d'être décryptés pour que l'application puisse se connecter aux bases de données en question. Ce bundle ne possède qu'un seul contrôleur qui crypte ou décrypte la chaîne de caractères passée en paramètre en fonction d'une chaîne connue de lui seul.

BddBundle → ce bundle gère les connexions aux bases de données des graphiques, et les opérations de requête et de récupération des résultats. Php a pour chaque base de données ses propres fonctions et l'objectif de BddBundle est d'offrir une couche d'abstraction au type de base de données. Bien entendu cela est déjà le cas avec l'extension PDO de PHP. Cependant l'avantage de créer notre propre bundle est d'avoir plus la main sur les fonctions utilisées et donc les performances.

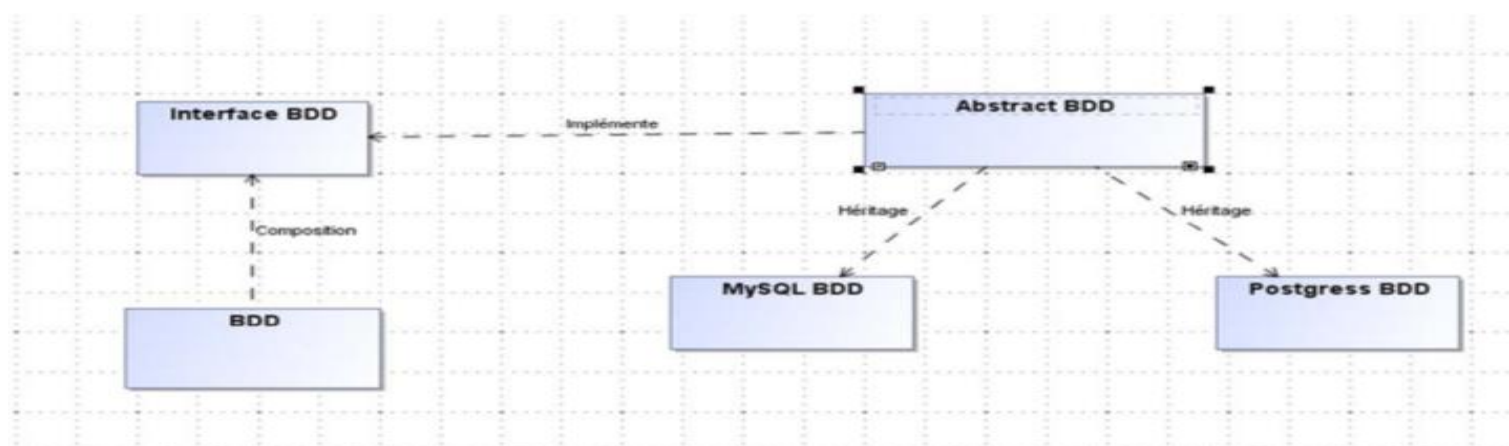


Schéma BDD Bundle Rapport de stage Maxime COLIN

La structure d'un bundle :

Controller : contient tous les contrôleurs

DependencyInjection : contient des informations sur les bundles (chargement automatique de la configuration).

Entity : contient les modèles. ///////////////Commande

Form : contient les éventuels formulaires. // // // Commande

Config : contient les fichiers de configuration du bundle.

Public : contient les fichiers publics du bundle : fichiers CSS et JavaScript, images, etc.

Views : contient les vues du bundle, les templates twig.

La commande pour générer un nouveau bundle est :

```
admin_geomer@2letg163:/var/www/html/ManageChart$ php app/console generate:bundle
```

Dans notre application, il existe des libraires enregistré dans le dossier vendor qui permettent d'intégrer des bundles externes à notre application comme :

Doctrine→ Est un ORM (couche d'abstraction à la base de données) pour PHP. Il fournit la persistance transparente des objets PHP, et c'est l'interface qui permet de faire le mapping entre nos objets et les éléments de la base de données.

Twig→ Est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le Framework Symfony.

SwiftMailer→ S'intègre dans n'importe quelle application Web écrite en PHP 5, offrant une approche orientée objet flexible et élégante pour envoyer des courriels avec une multitude de fonctionnalités contrôle de flux complexe, échappement automatique, héritage des templates, Langage extensible.

Composer→ Composer est un outil pour gérer les dépendances en PHP. Les dépendances, dans un projet, ce sont toutes les bibliothèques dont notre projet dépend pour fonctionner. Par exemple, notre projet utilise la bibliothèque SwiftMailer qu'on vient d'expliquer précédemment pour envoyer des e-mails, il « dépend » donc de SwiftMailer. Autrement dit, SwiftMailer est une dépendance dans votre projet.

Composer a donc pour objectif de nous aider à gérer toutes les dépendances. En effet, il y a plusieurs problématiques lorsqu'on utilise des bibliothèques externes :

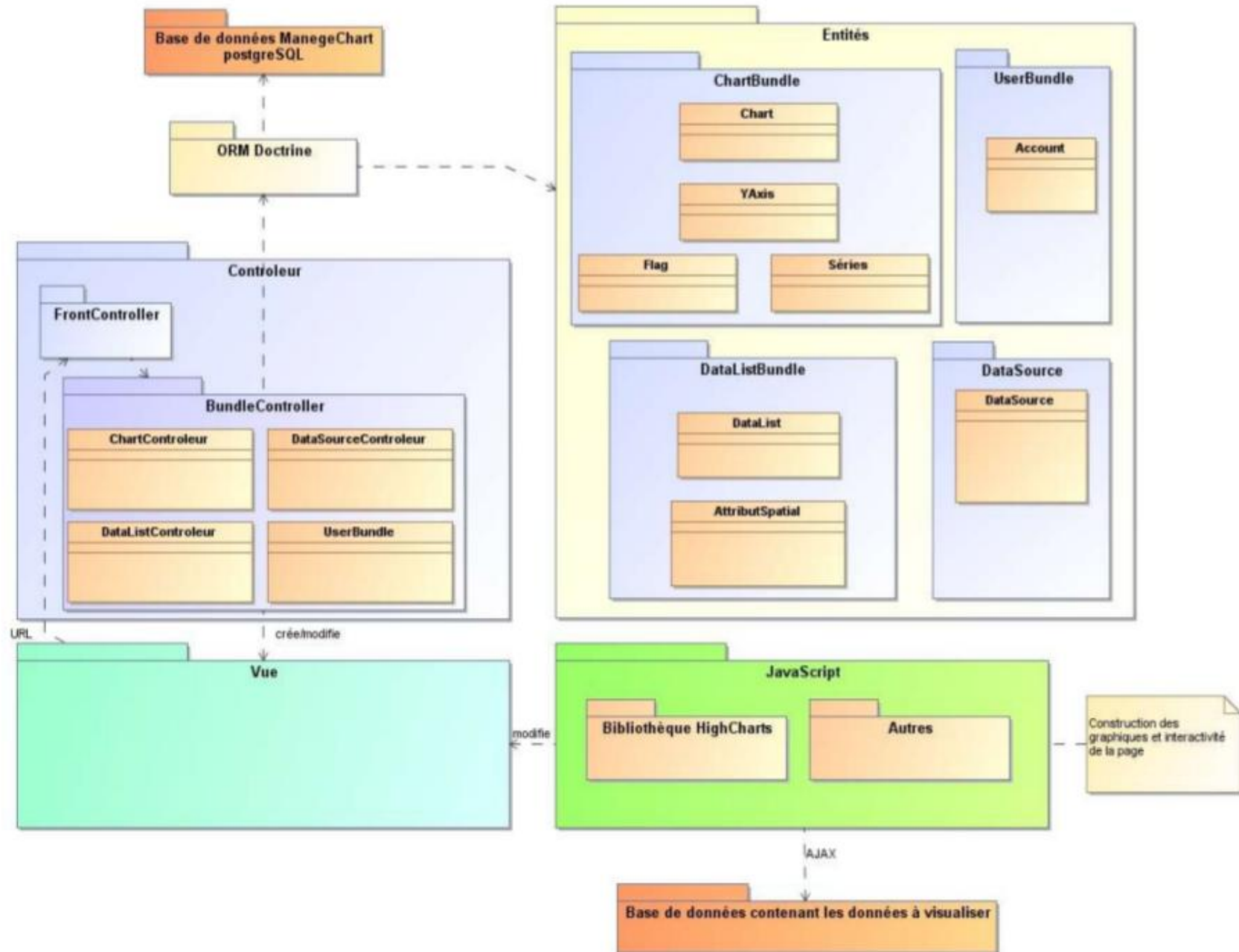
Ces bibliothèques sont mises à jour. Il vous faut donc les mettre à jour une à une pour vous assurer de corriger les bogues de chacune d'entre elles.

Ces bibliothèques peuvent elles-mêmes dépendre d'autres bibliothèques. En effet, si une de vos bibliothèques dépend d'autres bibliothèques, cela vous oblige à gérer l'ensemble de ces dépendances (installation, mises à jour, etc.).

Ces bibliothèques ont chacune leur paramètres d'autoload, et vous devez gérer leur autoload pour chacune d'entre elles.

Pour faire court, le fonctionnement de cette outil est pareille que 'APT' le gestionnaire de paquet de la distribution Linux. Cependant, c'est très récent, il n'existait aucun outil de ce genre pour PHP avant. La forte communauté qui s'est construite autour de Symfony2 a fait naître le besoin d'un tel outil, et l'a ensuite développé.

2. Architecture général de l'application ManageChart modèle et package



Architecture général de ManageChart

Schéma Rapport de stage Maxime COLIN

3. Dictionnaire des données

Dans l'existant, le dictionnaire des données reprend, pour chaque entité, la liste de ses attributs.

Les entités sont générées en base de données par Doctrine, qui est l'ORM par défaut utilisé par Symfony2, grâce à la commande PHP : « doctrine:schema:update ».

Entité account

id	auto_increment
username	varchar (255)
username_canonical	varchar (255)
email	varchar (255)
email_canonical	varchar (255)
enabled	bool
salt	varchar (255)
password	varchar (255)
last_login	timestamp (22)
locked	bool
expired	bool
expires_at	timestamp (22)
confirmation_token	varchar (255)
password_requested_at	timestamp (22)
roles	text
credentials_expired	bool
credentials_expire_at	timestamp (22)

Entité chart

id	auto_increment
namechart	varchar (255)
subtitlechart	varchar (255)
legendchart	bool
creditschart	text
xaxis_title	varchar (255)
xaxis_unit	varchar (255)
xaxis_type	varchar (255)
exportprintchart	bool
exportcsvchart	bool
typechart	text
gapsizchart	int
titlechart	varchar (255)
urlcreditschart	text
invertedchart	bool

Entité datasource

id	auto_increment
namebdd	varchar (255)
descriptionbdd	text
hostbdd	varchar (255)
portbdd	int
loginbdd	varchar (255)
passwordbdd	varchar (255)
typebdd	int
typestrbdd	varchar (255)
datebdd	timestamp (22)

Entité attributspatial

id	auto_increment
nameattribut	varchar (255)
valueattribut	varchar (255)
typeattribut	varchar (255)
keywordattribut	varchar (255)

Entité series

id	auto_increment
titleserie	varchar (255)
unitserie	varchar (255)
typeserie	varchar (255)
colorserie	varchar (255)
markerserie	bool
dashstyleserie	varchar (255)
parameterdatalist	varchar (255)

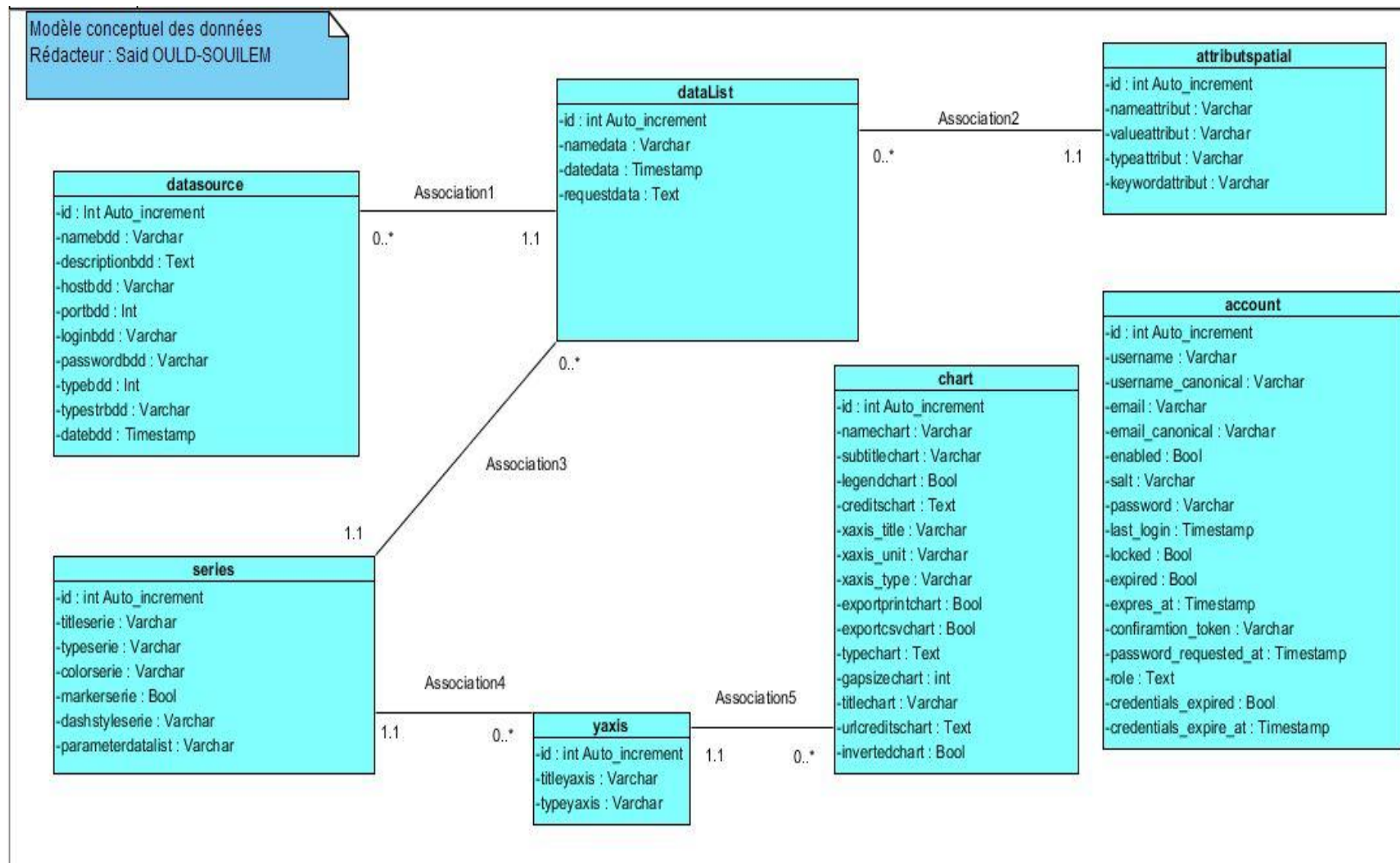
Entité datalist

id	auto_increment
namedata	varchar (255)
datedata	timestamp (22)
requestdata	text

Entité yaxis

id	auto_increment
titleyaxis	varchar (255)
typeyaxis	varchar (255)

4. Architecture de la base de donnée



L'entité « account » servant à la gestion des utilisateurs, étant géré par FOSUserBundle, n'a aucune interaction avec le reste de l'application.

dataList 0,* → association2 ← 1,1 attributspacial

datasource 0,* → association1 ← 1,1 dataList

yaxis 0,* → association4 ← 1,1 series

dataList 0,* → association3 ← 1,1 series

chart 0,* → association5 ← 1,1 yaxis

1. Contraintes d'intégrité

Les contraintes d'intégrité sont représentées par les restrictions établies aux données. Elles sont principalement définies par les cardinalités 0,1 et 1,1.

Dans notre cas, les règles de gestion montrent que :

- Un « attributspacial » ne peut être associé qu'à une seule « datalist »,
- Une « datalist » ne peut être associé qu'à une seule « datasource »,
- Une « series » ne peut être associé qu'à un seul « yaxis » ainsi qu'à une seule « datalist »,
- Un « yaxis » ne peut être associé qu'à un seul « chart ».

Les contraintes d'intégrité sont également définies les contraintes « UNIQUE » et « NOT NULL » dont :

- « UNIQUE » ne représente que les champs « id » qui sont auto-incrémenté, - « NOT NULL » s'applique à tous les champs qui ne peuvent être vide. Ici, seuls les champs « subtitlechart », « xaxis_title », « xaxis_unit » de l'entité « chart » ainsi que le champ « unitserie » de l'entité « series » peuvent être vide. Tous les autres attributs sont donc concernés par cette contrainte.

2. Dépendances fonctionnelles

Les dépendances fonctionnelles permettent de déterminer pour chaque entité, un attribut source. Pour chaque valeur de cet attribut source, on identifie les attributs déterminés.

Entité account : {id} { username, username_canonical, email, email_canonical, enabled, salt, password, last_login, locked, expired, expires_at, confirmation_token, password_requested_at, roles, credentials_expired, credentials_expire_at }

Entité chart : {id} { namechart, subtitlechart, legendchart, creditschart, xaxis_title, xaxis_unit, xaxis_type, exportprintchart, exportcsvchart, typechart, gapsizechart, titlechart, urlcreditschart, invertedchart }

Entité datasource : {id} { namebdd, descriptionbdd, hostbdd, portbdd, loginbdd, passwordbdd, typebdd, typestrbdd, datebdd }

Entité series : {id} { titleserie, unitserie, typeserie, colorserie, markerserie, dashstyleserie, parameterdatalist }

Entité attributspatial : {id} { nameattribut, valueattribut, typeattribut, keywordattribut }

Entité datalist : {id} { namedata, datedata, requestdata }

Entité yaxis : {id} { titleyaxis, typeyaxis }

8. Présentation du Framework Symfony 2 MVC (Model, Views, Controller)

1. Architecture du Framework



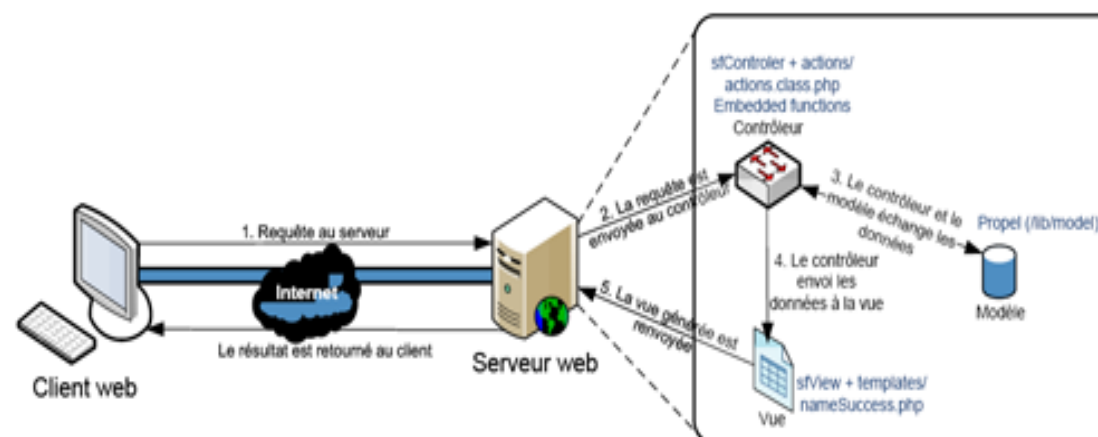
Symfony est entièrement basé sur le design pattern MVC (Modèle-Vue-Contrôleur). Utilisé dans d'autres technologies (notamment Java, C #), il n'a été largement diffusé dans le monde PHP qu'avec des Framework tels que symfony.

Le MVC est un pattern architectural qui sépare les données ça veut dire le modèle, l'interface homme-machine (la vue) et la logique de contrôle (le contrôleur).

Ce modèle de conception impose donc une séparation en 3 couches :

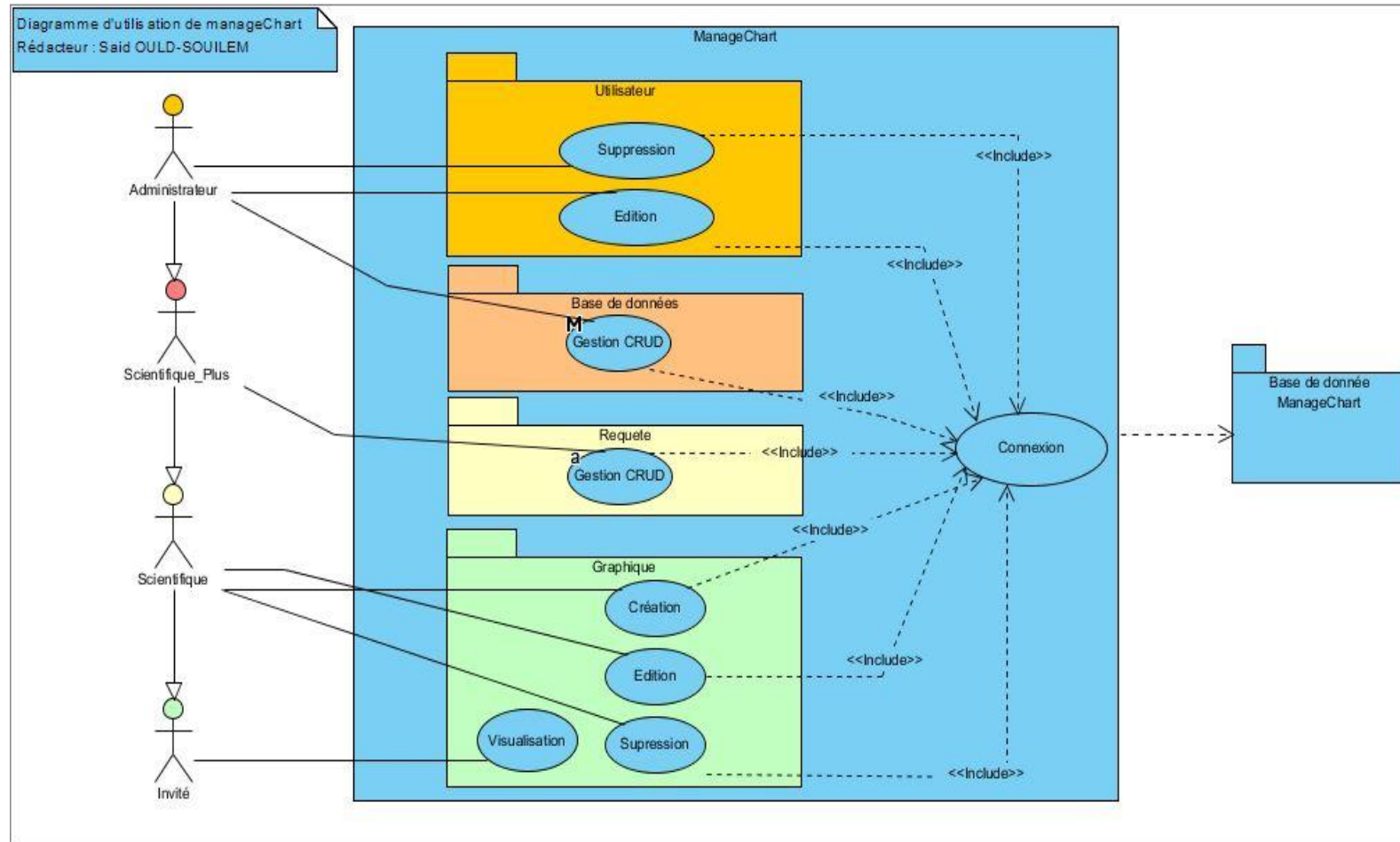
- **Le modèle** : Il représente les données de l'application. Il définit aussi l'interaction avec la base de données et le traitement de ces données.
- **La vue** : Elle représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle.
- **Le contrôleur** : Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues. Symfony implémente donc trois couches répondant à ce pattern.

Le schéma ci-dessous illustre le mécanisme du MVC dans un projet symfony lorsque l'utilisateur navigue sur le site.



9. Partie conception

1. Diagramme de cas d'utilisation



En Fonction du Rapport de stage Maxime COLIN

Il existe 4 types d'utilisateurs différents :

L'invité : Il peut visualiser les graphiques existant sur l'application.

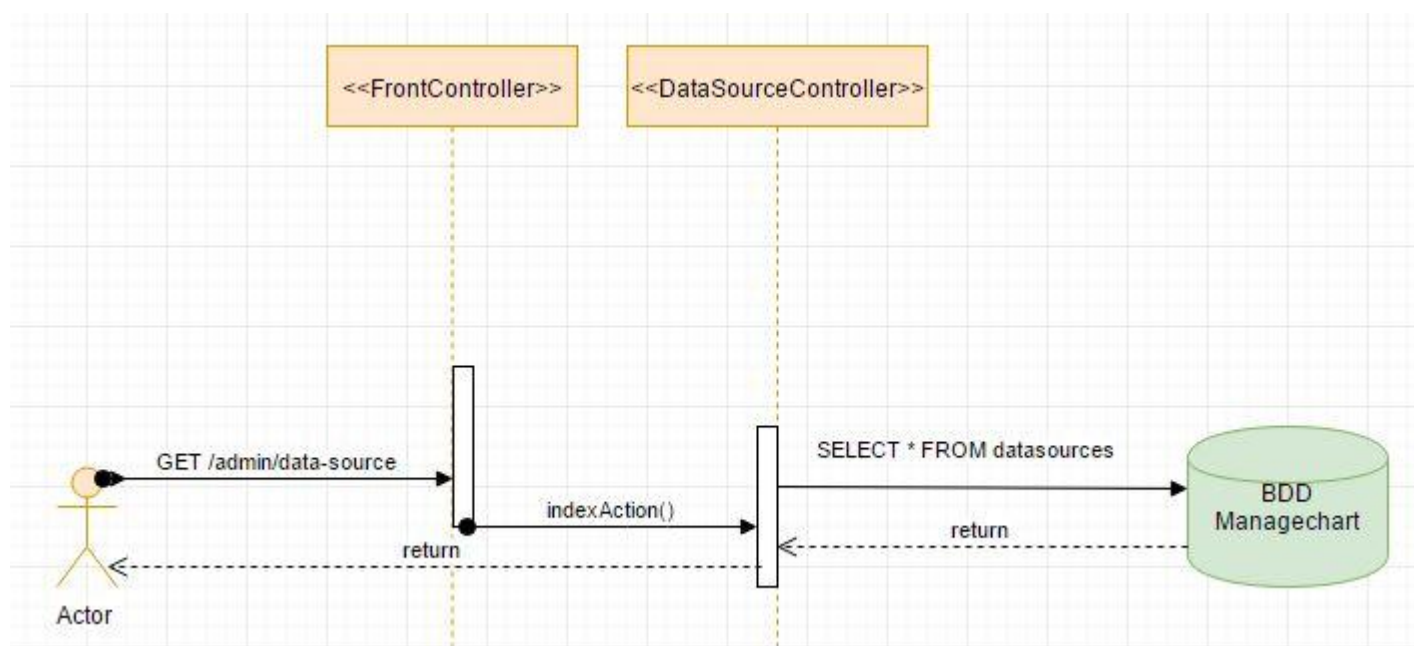
Le scientifique : Il peut créer, éditer ou supprimer des graphiques

Le scientifique plus : Il a la possibilité de créer, éditer, supprimer ses propres requêtes, ça nécessite des connaissances en base de données.

L'administrateur : Il gère les bases de données ainsi que les utilisateurs, il peut supprimer d'autres utilisateurs mais ne peut éditer que lui-même.

2. Diagramme de séquence

Exemple d'affichage des DataSources existantes



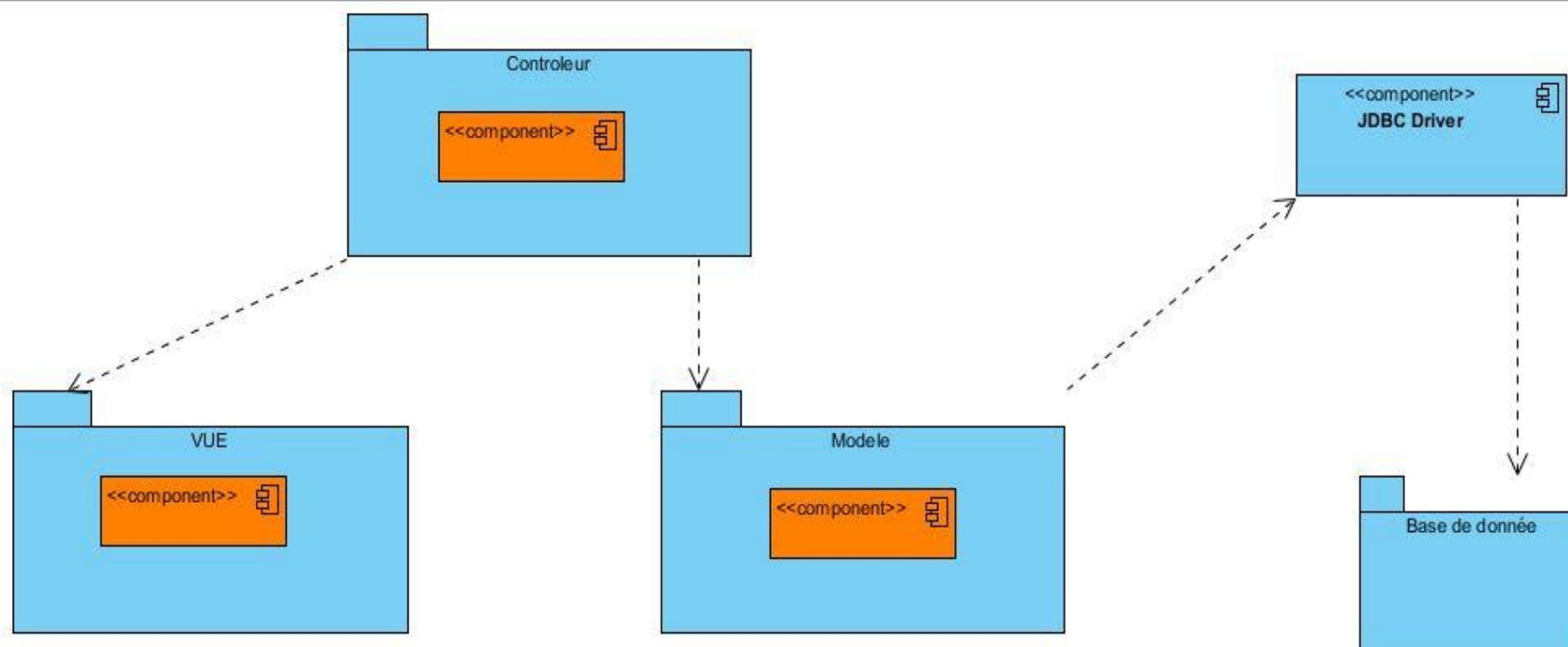
Le client effectue une requête pour afficher la liste des connexions aux bases de données, Comme on a vu précédemment, le Front Controller lit la requête et va chercher le bon Controller avec la bonne méthode,

La méthode indexAction() effectue une requête SQL sur la base de données de ManageChart pour récupérer la liste des connexions aux bases de données existantes et renvoie directement la réponse au client.

Le même exemple fonctionne aussi pour afficher ces listes mais avec des méthodes différentes pour chaque Controller :

- Des graphiques
- Des requêtes SQL
- Des utilisateurs

3. Diagramme de composants



En fonction du Rapport de stage Maxime COLIN

Le contrôleur désigne le répertoire : /SRC

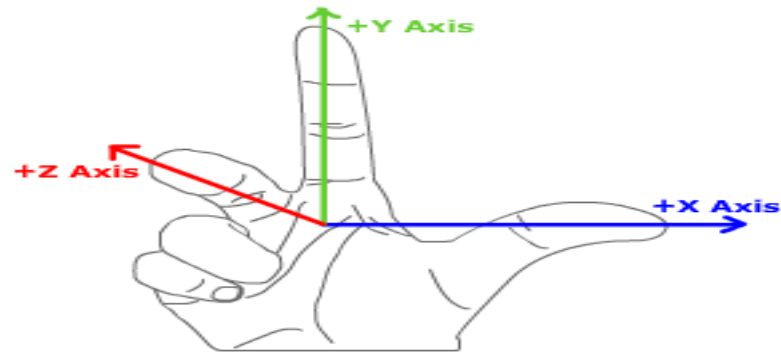
Le modèle désigne le répertoire : /APP

La vue désigne le répertoire : /WEB

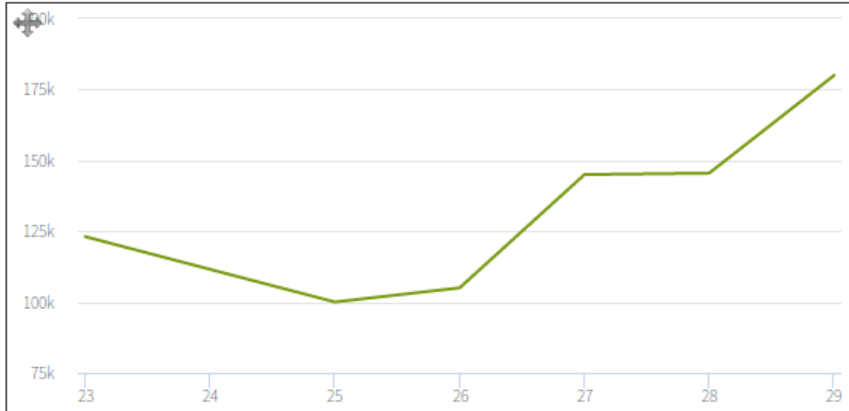
10. Partie développement (Aspect technique)

Avant de commencer à parler des Bogues réalisés, une explication des types d'axes rencontrés lors du développement,

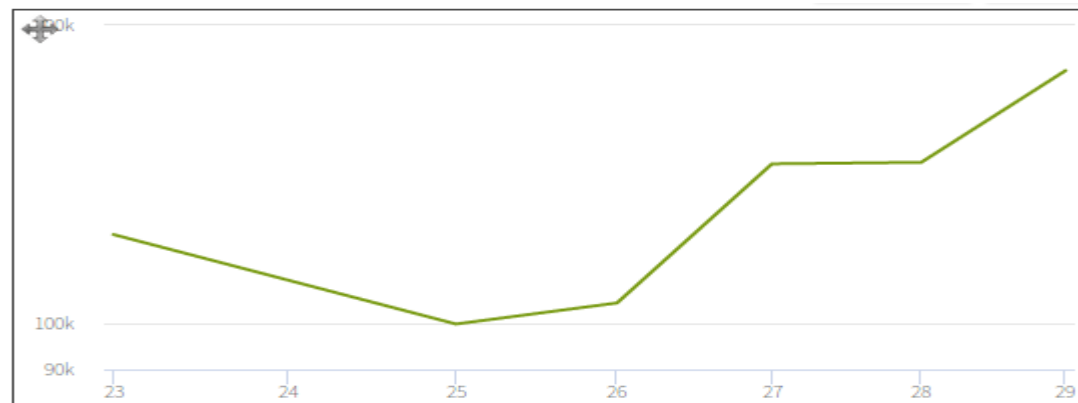
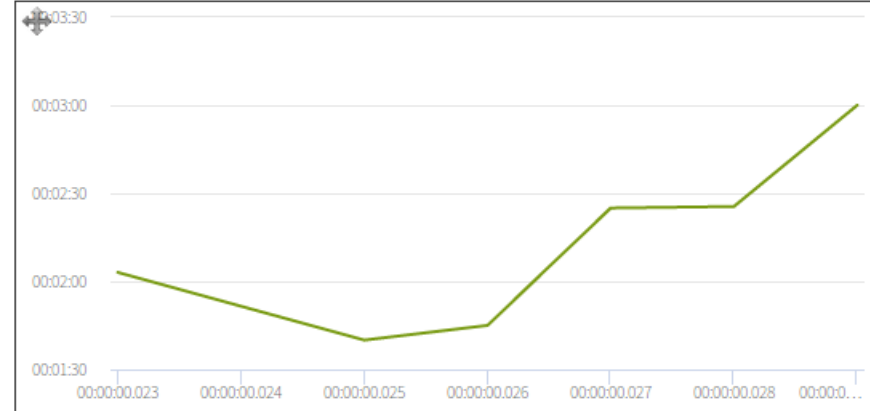
Pour créer un graphe, il est nécessaire de comprendre le fonctionnement des axes pour la génération des données



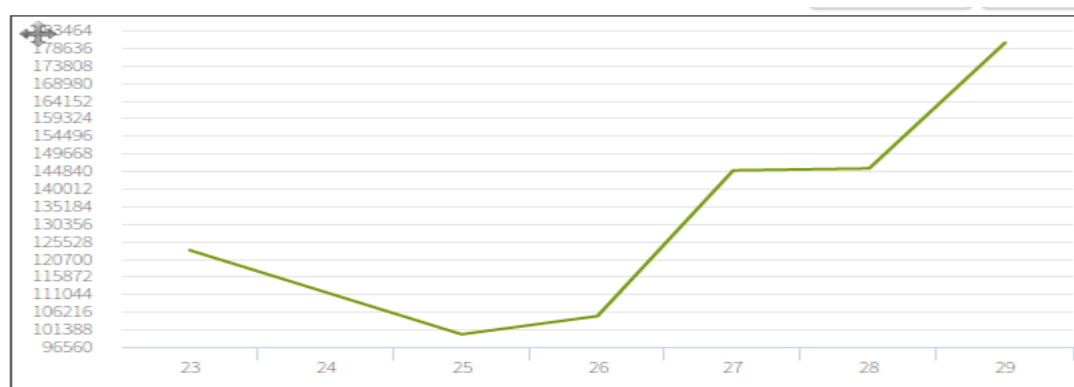
Linéaires



Temporel



Dans le cas de logarithmique, l'axe x et y n'accepte pas des valeurs négatives puisque l'échelle logarithmique est de '0.1' à '100' donc toutes valeurs négatives ne sera pas accepté.



Catégorie : ce type d'axe se base sur des données qui sont stockés dans la base et qui sont afficher dans chaque axe du graphe (pour mon exemple)

1. Bogues

Le développement à commencé par la correction de différents bogues

Bogue 1

Le premier problème rencontré c'est le cas du type logarithmique, Lors de la création d'un graphe de type simple, une contrainte doit être mise en cas de valeurs négatives et une erreur doit être lancé, le bug se situait pour l'axe y la contrainte n'apparaissait pas.

Exemple :

```
type: {% if (chart.typeChart == 'simplechart' or chart.typeChart == 'dynamicchart' or chart.typeChart == 'polarchart') %}
```

```
{{ chart.xAxisType | e('js') }}{% else %}null{% endif %} //
```

Cette ligne est extraite du fichier showIframe.html.twig, elle permet de récupérer le type d'axe x en fonction du type de graphe (Dans notre cas – Graphe simple).

⇒ La portion de code permettant le changement du type d'axe x et y lors de son enregistrement :

```
/* Creation du chart */
var chart = new Highcharts.Chart(options);
var axisy = $('#container').highcharts().yAxis[(idYAxis - 1)]; // Recuperation de l'axisy en fonction de l'idYaxis
chart.yAxis[0].remove();
```

```
/*ON CHANGE */
```

```
$(typeXAxis).change(function() {
updateTypeAxis(this,axis);
});
$(typeYAxis).change(function(){
updateTypeAxis(this,axisy);
});
```

⇒ La deuxième erreur concerne la requête qu'on devait préciser pour la récupération des valeurs et l'événement change()

```
var indexY = 1;
```

```
{% for yAxis in list_yAxis | reverse %}
```

```
$( 'input#mc_chartbundle_chart_list_yAxis_' + indexY + '_titleYAxis').val('{{ yAxis.titleYAxis | e('js') }}').change();
```

```
$( 'select#mc_chartbundle_chart_list_yAxis_' + indexY + '_typeYAxis').getTypeAxis('{{ yAxis.typeYAxis | e('js') }}').change();
```

```
indexY++;
```

```
{% endfor %}
```

Cette boucle en JavaScript va permettre lors de l'édition du formulaire de parcourir la liste des yAxis et se placer en fonction de l'input pour récupérer le titleYaxis et lancer l'événement change () et le selector pour récupérer le typeYaxis et lancer l'événement change () .

Bogue2

+Correction dans l'affichage du graphe dans le Frame lors de l'enregistrement d'un graphe souhaité.

```
$(typeYAxis).on('change', function() {
console.log('%cModification du type de l'axe-Y pour : %c' + $(this).val(), 'color: #90C3D4', 'color: #8085E8');
idYAxis = $(yAxis).find('label').first().html().match(/(.*)\d+\/)[2];
chart.yAxis[(idYAxis - 1)].update({
type: $(this).val()
});
```

Cette fonction va modifier le type d'axe y en fonction du type à lui attribuer lors de la saisie, la récupération va se faire par le billet de l'idYaxis pour reconnaître le type d'axe y.

Bogue3

Dans l'existant, les dates s'affichaient mais pas en ordre, l'application se basait sur une date courante en anglais défini par un timestamp pour l'ajout soit d'une base de donnée soit d'une requête. En installant une librairie twig "DATATABLE SORT" qui permet de classer les données.

Les dates s'afficheront par ordre chronologique.

ManageChart
Bases de données
Requêtes SQL
Graphiques
Utilisateurs
admin ▾

Bases de données

Nouveau

Nb d'entrée

Actions	id	Nom de connexion	Nom Bdd	Description	Type	Date
	6	clapot postgres9.3	adviclim_gama	clapot postgres9.3	PostgreSQL	2016/11/22
	5	clapot postgres8.4	SEVE	base de données simulations SEVE	PostgreSQL	2016/11/15
	4	Geosu	geosu	geosu postgresql sur 7iuem68:5433	PostgreSQL	2016/10/17
	3	managechart sur mysql	managechart	svghzgurfomz	MySQL	2016/07/08
	2	Somlit	somlit	Base Somlit IUEM	PostgreSQL	2016/07/01
	1	Base de test testData	testdata	Base de données comportant des tables de test	PostgreSQL	2016/06/15

Page 1 sur 1

«
<
1
>
»

ManageChart 2014-2016 LETG-Brest Géomer

La portion de code permettant le classement des dates par ordre chronologique :

```
<td>{{ 'datasources.table.titlecolumn.dateBDD'|trans() }}</td>
```

```
<td>{{ dataSource.dateBDD|date('Y/m/d') }}</td>
```

Changement du format de la date de timestamp en se basant sur le tri des années et des mois pour une meilleure visualisation lors de la création.

Stagiaire : Saïd Ould-souilem | Formateur : M. Jean-Philippe Babau | Tuteur : M. Mathias Rouan | Juin 2017

Bogue4

Dans l'existant, le choix des couleurs se fait par une liste déroulante, le problème rencontré c'est lors de l'enregistrement d'un graphique circulaire, les couleurs qui sont choisis et insérés sont pas toujours les mêmes affiché dans l'iframe.

Dans ce cas, et d'après le code source, l'ancien stagiaire employait une classe en php qui permet de déclarer une variable qui va récupérer une liste de couleurs et les insérer dans le selector :

Exemple :

```
<?php
namespace Mc\ChartBundle\AvailableChoice;
/**
 * Classe listant les couleurs disponibles pour les series
 */
class AvailableColorSerie {
    public static $colorSerie = array(
        '#3399CC' => '#3399CC',
        '#65BFE4' => '#65BFE4',
        '#006699' => '#006699',
        '#1A7BB8' => '#1A7BB8',
        '#D4E6F2' => '#D4E6F2',
        '#007D83' => '#007D83',
        '#2AA8B0' => '#2AA8B0',
        '#A0D8DC' => '#A0D8DC',
        '#228527' => '#228527',
        '#3BA737' => '#3BA737',
        '#BAE0B9' => '#BAE0B9',
        '#799B13' => '#799B13',
        '#9CC11B' => '#9CC11B',
        '#CBDE88' => '#CBDE88',
        '#E0B100' => '#E0B100',*
    );
}
```

Cependant, le problème n'est pas ici mais lors de la création du graphique circulaire , on pourrait dire que le graphique ne pourrait pas avoir une couleur pour chaque tranche , donc la solution que j'ai mis en place c'est l'installation d'une extension en twig qui permet de prendre une liste de couleur et la mélanger pour avoir un graphe circulaire c'est l'extension "Twig_Extension_Array" par la fonction shuffle.

Cette implémentation n'est pas vraiment idéale dans notre cas puisqu'il serait mieux d'avoir un système qui présente un tirage de couleur en fonction du nombre de tranche du graphique circulaire mais cela nous convient pour le moment mais au cours du développement il serait impératif de trouver une solution adapté au graphique circulaire.

```
1 var listColors = [
2     {% for key,color in listColorSerie|shuffle %}
3         '{{ color|e('js') }}',
4     {% endfor %}
5 ];
```

2. Quelques modifications :

+ Correction problème "AJOUT D'axis " crash du fichier JS

Symfony fait l'interaction entre le php et le js, En déportant les couches MVC du côté client, on obtient donc une API comme Modèle (la plupart du temps une API REST), un Contrôleur et une Vue écrits en JS.

Ainsi, une fois la première page chargée, les transferts avec le serveur sont moins coûteux (quelques kilo-octets de données JSON, contre quelques centaines pour une page html générée côté serveur).

En effet, on ne rafraîchit que la partie désirée de la page, et on évite ainsi d'avoir à recharger toute la page (événements JavaScript, librairies highcharts ...).

Mais dans notre cas, à chaque modification du code source, et lors du lancement de l'application, le fichier AJOUT D'axis ne se chargeait pas, puisque le fichier n'avait pas tout les droits d'écriture ou de lecture donc un crash du fichier est probable.

+ Mise à jour des fichiers js par les commandes symfony suivantes :

+Mettre à jour les liens symboliques de l'application :

```
Php app/console assets :install web/ --symlink
```

+ Importation des ressources selon l'environnement de développement ou de production choisi :

```
Php app/console assetic :dump --env= prod/dev
```

+ Correction erreur type 'Time-Chart'== Récupération du type d'axe y pour un graphe enregistré lors de l'édition.

+Correction erreur type 'Multi-axis'== Récupération du type d'axe y pour un graphe enregistré lors de l'édition = Linéaire logarithmique, Temporel, catégorie.

+ Correction erreur type 'Dynamic-Chart'== Récupération du type d'axe y pour un graphe enregistré lors de l'édition = Linéaire, logarithmique, Temporel, catégorie.

3. Fonctionnalités

Ajout d'un nouveau type de graphe circulaire

Dans l'existant, le graphique simple se composait d'une fonctionnalité permettant de changer le type de série en pie = diagramme circulaire appelé diagramme en secteurs.

La fonctionnalité se basait sur des séries mais apres discussion avec mon maitre de stage une solution est possible en mettant le graphique circulaire comme un autre type séparé du graphe simple

On représente le type de graphe circulaire :

ManageChart Bases de données Requetes SQL Graphiques Utilisateurs admin

Graphiques

Nb d'entrée 10

Actions

id	
371	
370	
369	
368	
367	

Nouveau graphique

Recherche

localhost/ManageChart/web/app_dev.pt

localhost/ManageChart/web/app_dev.pt

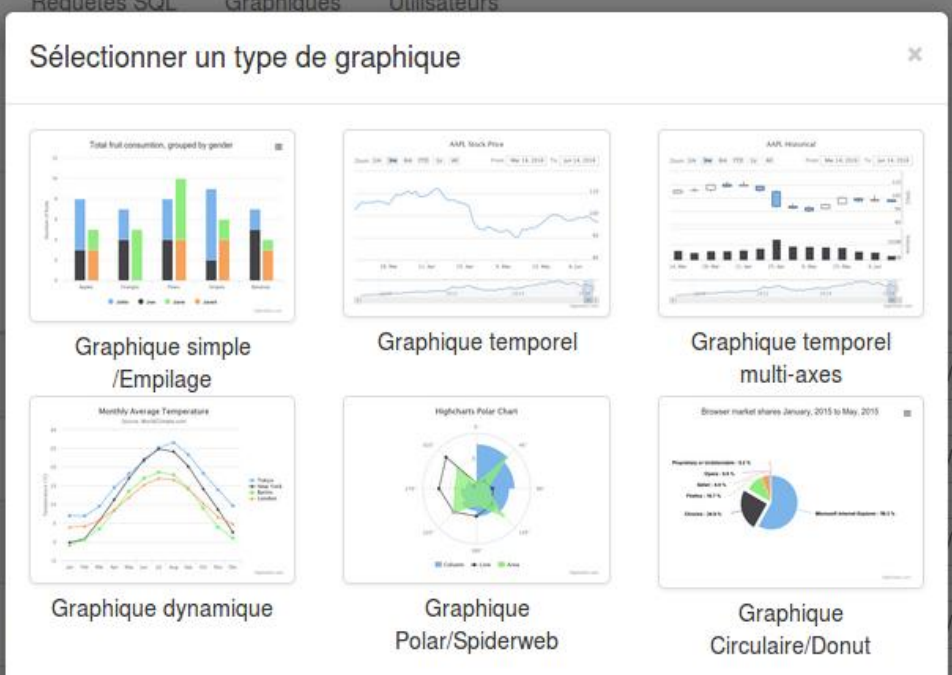
localhost/ManageChart/web/app_dev.pt

localhost/ManageChart/web/app_dev.pt

localhost/ManageChart/web/app_dev.pt

localhost/ManageChart/web/app_dev.pt

Sélectionner un type de graphique



Graphique simple /Empilage

Graphique temporel

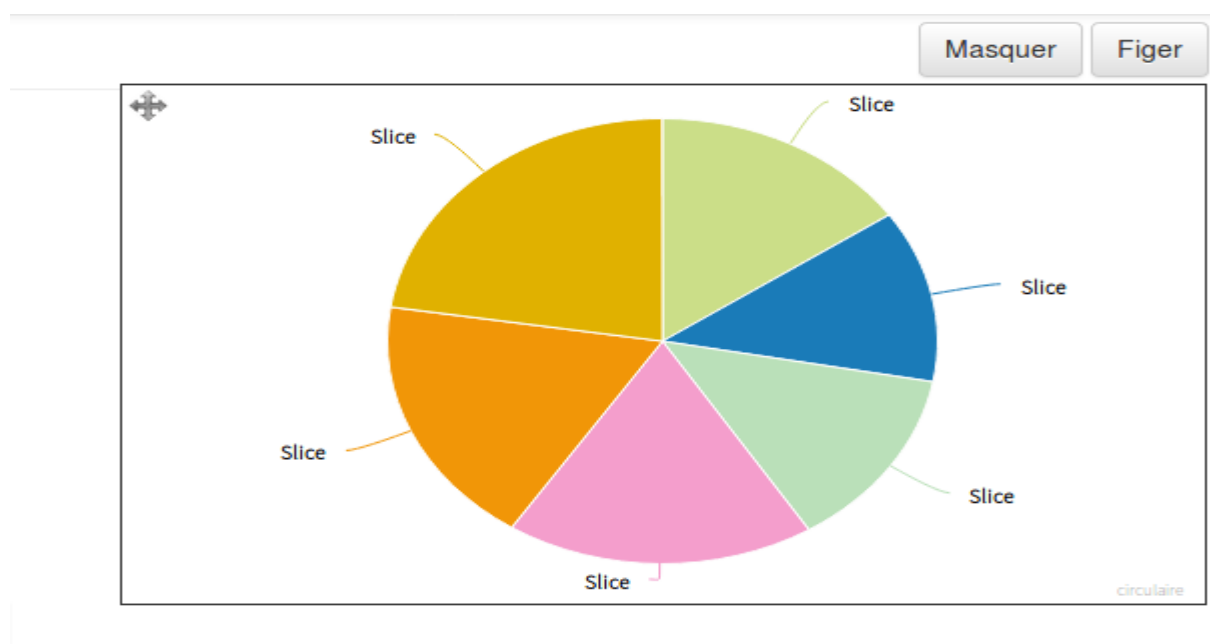
Graphique temporel multi-axes

Graphique dynamique

Graphique Polar/Spiderweb

Graphique Circulaire/Donut

+ Graphique circulaire :

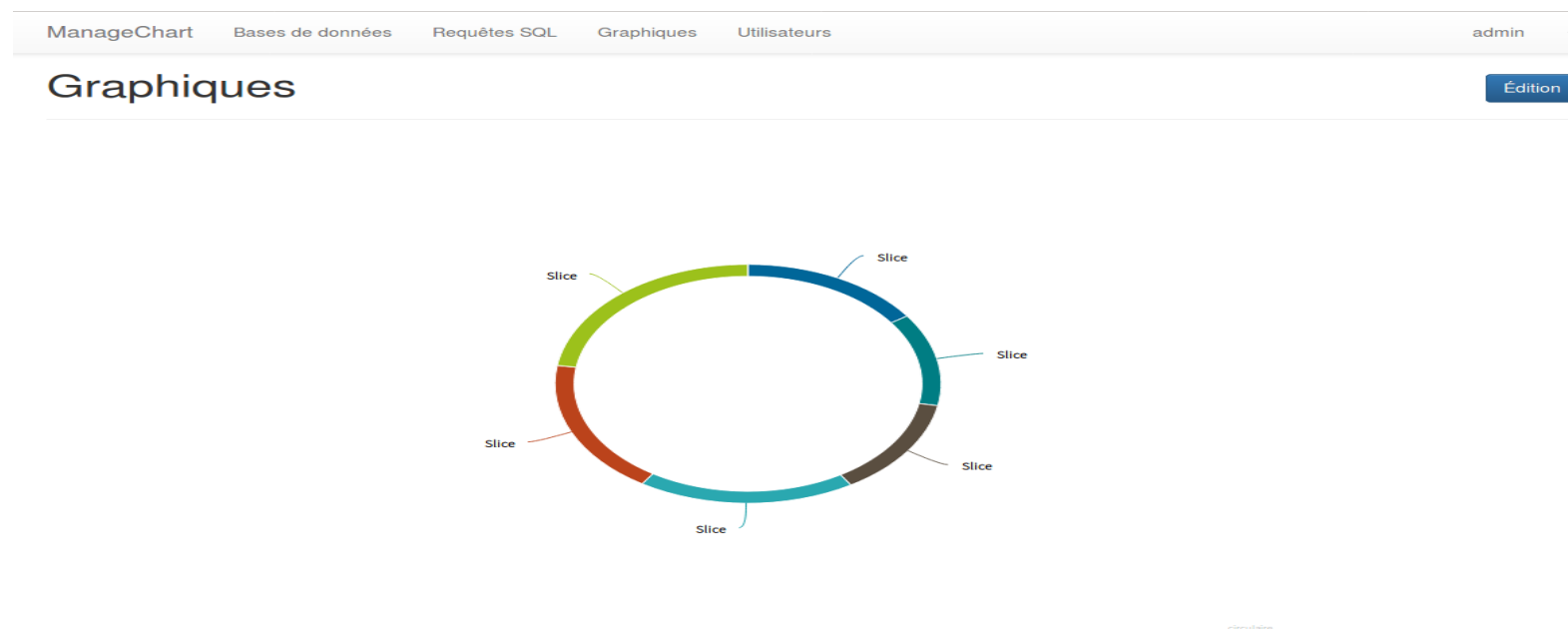


Série n° 1 Annuler

Titre	<input type="text" value="toto"/>	Unité	<input type="text" value="1"/>
Requête	<input type="text" value="Requête de test SE"/>	Paramètre	<input type="text" value="toto@1"/>
Type	<input type="text" value="Circulaire"/>	Couleur	<input type="text" value="#3399CC"/>
Marqueur	<input type="checkbox"/>	Style de ligne	<input type="text" value="Ligne"/>
Taille	<input type="text" value="50%"/>	Taille intérieure	<input type="text" value="90%"/>

Dans le cas de notre graphe, le graphique circulaire se base sur une série de donnée, j'ai ajouté deux paramètres, size et innersize pour modifier le graphique circulaire et changer son type pour un graphique en anneau appelé DonutChart

Avec une taille de 50% et une taille intérieure de 90 %, le graphe circulaire aura cet affichage :



NB : Cette fonctionnalité est opérationnelle pour l'instant, il suffit de penser à rendre quelques champs de la saisie non visible. (on pourrait penser à des modifications dans la base de données).

Démarche suivie :

- + Le graphique circulaire se base sur des séries de données,
 - + Création des fichiers d'ajout et modification dans le cas d'un graphe de type 'Pie'
 - + Dans l'API highcharts ils existent des fonctionnalités qu'on pourrait paramétrer pour une section définit,
- Quand je parle de section ça peut concerner les options du graphe.

Exemple :

```
var options = {
  chart: {
    renderTo: 'container',
    zoomType: 'x',
    events: {
      load: function() {
        this.credits.element.onclick = function() {
          if ($('#urlcreditsChart').val() != "") {
            window.open(
              $('#urlcreditsChart').val(),
              '_blank'
            );
          }
        }
      }
    }
  },
};
```

+En ajoutant la ligne dans l'option du chart :

```
Options.chart.pie = true ;
{% if chart.typeChart == 'piechart' %} type : 'pie' , {% else %} {% endif %}
```

+En ajoutant les deux lignes dans la section series [] pour récupérer les champs size et innersize :

```
size : '{{serie.size | e('js')}}',
innersize : '{{serie.innersize | e('js')}}'
```

+Pour le changement des tailles en fonction du graphe lors de l'édition:

```
Var size = $('#select#mc_chartbundle_series_size') ; // Récupération du size par la requête
Var innersize = $('#select#mc_chartbundle_series_innersize') // Récupération du innersize par la requête
$('#select#mc_chartbundle_series_size').val('{{serie.size | e('js')}}').change(); // l'événement change sur le size
$('#select#mc_chartbundle_series_innersize').val('{{serie.innersize | e('js')}}').change(); // l'événement change sur le innersize
```

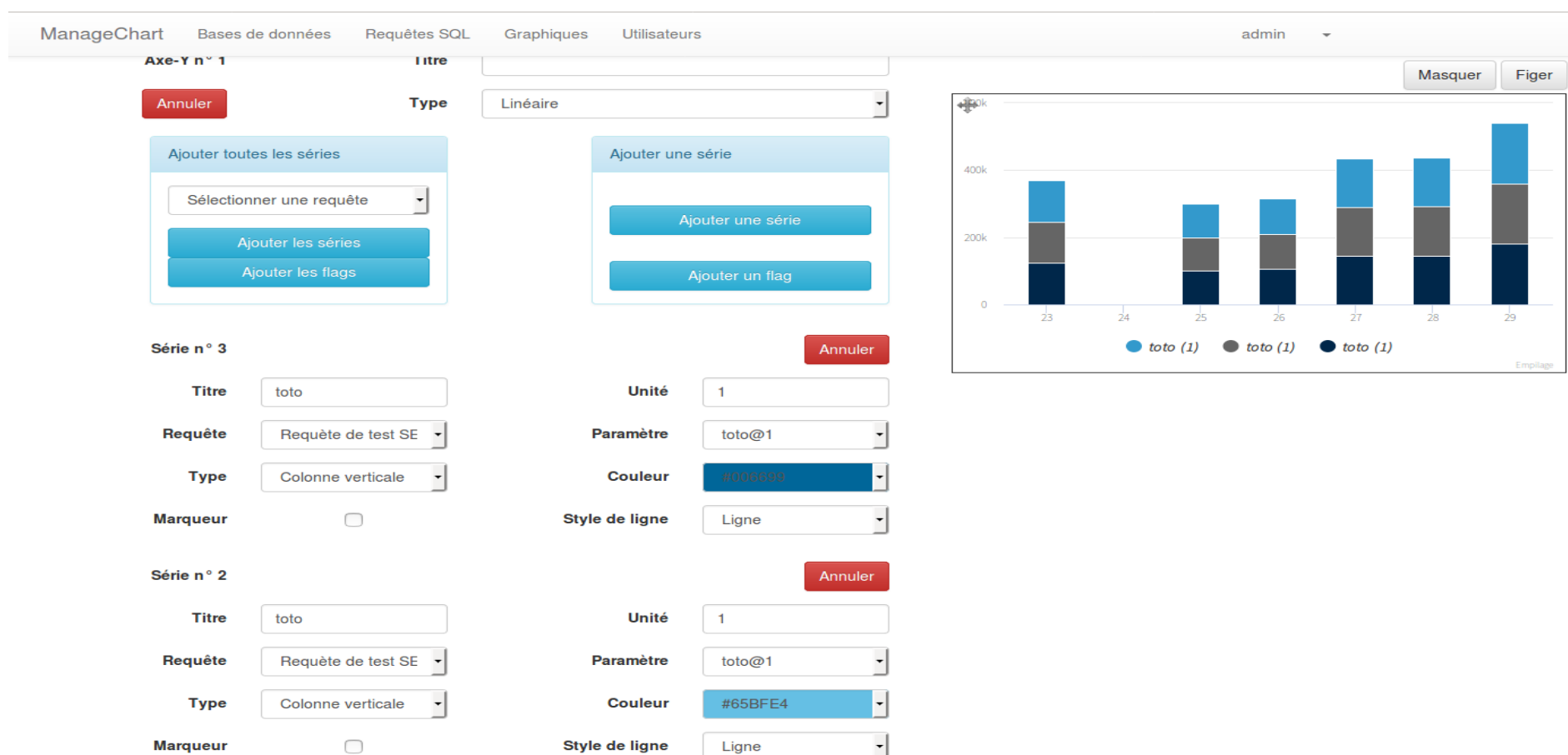
Ajout d'une nouvelle fonctionnalité d'empilage pour un graphique simple

L'ajout d'une nouvelle fonctionnalité Stacking = empilage permettant d'empiler le graphe en deux modes différents

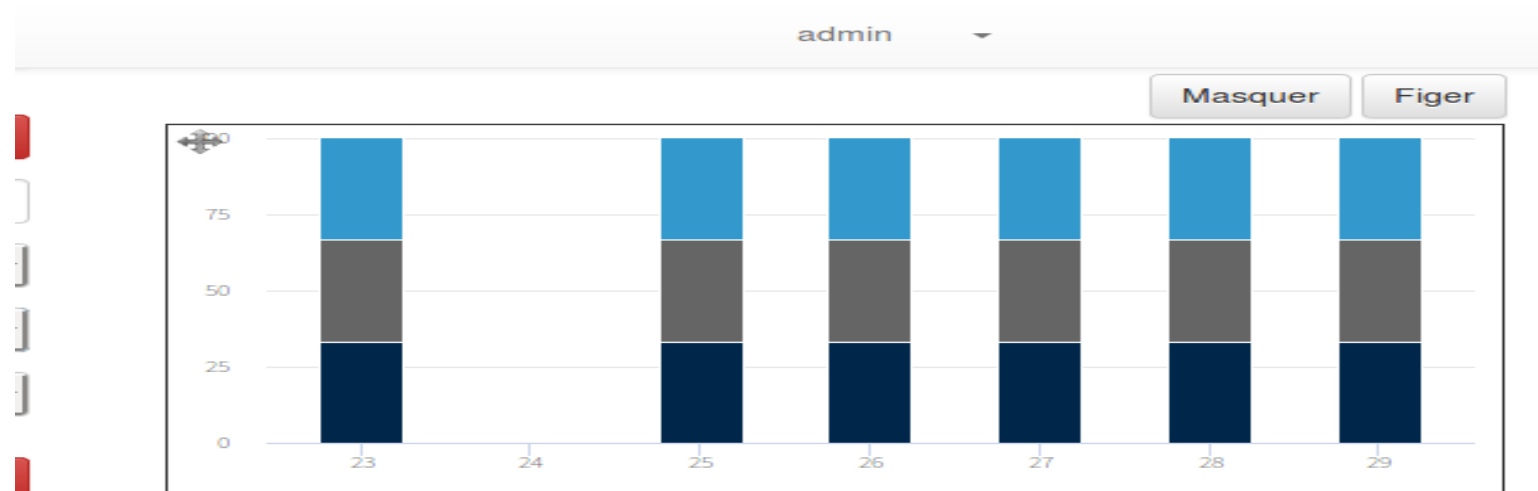
Figure décrivant l'ajout de la fonctionnalité Empilage,

Le choix d'empilage se fait en fonction des séries ajoutées soit concernant le type courbe, Aire, Aire en courbe, colonne verticale, colonne horizontale.

Mode normal :



Mode percent : (pourcentage)



Démarche suivie :

+ La classe permettant de réunir les deux modes dans un tableau array :

```
Public static $typeStack = array(
    normal => formchart.normal',
    percent => formchart.percent'
);
```

+ La classe chartType.php permet d'ajouter le champ empilage dans le formulaire et insérer les deux modes à partir du tableau array :

```
($entity->getTypeChart() == 'simplechart') {
    $builder
        ->add('typestacked', 'choice', array(
            'label' => 'formChart.typestacked',
            'choices' => AvailableTypeSeries::$typesStack,
            'placeholder' => 'choisir une option',
            'Required' => false
        ));
```

+ Dans le fichier showIframe.html.twig lors de la définition du graphe, l'empilage doit être effectué pour des séries de données donc c'est dans la section series

```
Series : [
    {% if (chart.typeChart == 'simplechart') %}
        Stacking : '{{chart.typestacked|e('js')}}',
    {% endif %}
    ..... ]
```

11. Modélisation techniques

1. Choix et outils techniques

Serveur	Apache 2.2
Base de données	PostgreSQL 9.3 , MYSQL
Modélisation	Visual paradigm, Drawlo
Langage de développement	PHP 5.5.9 JavaScript HTML 5 CSS 3 TWIG
Librairies	JQuery Highchart, Highstock
Framework	Symfony 2.8 Bootstrap
IDE	Sublime texte 2
Gestionnaire de version	GIT
Automatisation des taches	MS Project
Débogage	xDebug Firebug Web Debug Toolbar de Symfony

Visual Paradigm :



Pour la modélisation, le logiciel Visual Paradigm fournit tous les outils nécessaires grâce à son large choix de diagrammes. Il est comme son nom le laisse supposer, un logiciel permettant aux programmeurs de mettre en place des diagrammes UML. Disposant d'un outil créant des rapports personnalisables aux formats PDF, Word ou HTML afin de les partager et les publier sur Internet, cette application est compatible avec de nombreuses applications, standards et environnements. Ainsi, vous pourrez générer notamment des diagrammes de séquences ou de cas d'utilisation et ainsi produire du code source dans de nombreux langages comme le Java ou encore le C++, ou bien faire l'inverse, générer des diagrammes à partir de code déjà existant.

Symfony 2.8, HTML5, CSS3, JQuery, PHP 5, Bootstrap :



+PHP est un langage de programmation qui se connecte à une base de données, et produit des pages web dynamiques via des requêtes sur un serveur http.

+jQuery et le Framework Bootstrap : pour sa grille adaptable à la différente taille d'écran, ainsi que pour ses nombreuses classes mettant rapidement en forme les éléments.

2. Framework et librairies

J'ai choisi ce Framework parce qu'il est très efficace dans les développements PHP, respectant le design pattern MVC et augmentant nettement le gain de temps pour le développeur et les performances de l'application. Le projet est aussi très bien organisé ce qui en facilite la maintenabilité.

J'emploi également Sublime-Texte parce que c'est parmi les IDE les plus utilisés en PHP.

Le but de l'application ManageChart étant de fournir des représentations graphiques, la librairie Highcharts, permettant la création de graphiques interactifs, a été sélectionnée car elle offre une grande variété de graphiques et d'options d'exports.



Se décline en trois sous-librairies :

Highcharts permettant la génération de graphiques.

Highstock permettant la génération de graphiques spécifiquement temporel.

Exporting permettant l'ajout d'options d'export au format image et PDF, ainsi que l'impression papier.

Export-csv permettant l'ajout d'options d'export au format CSV, XLS, ainsi que la visualisation dans le navigateur. Celle-ci dépend de la précédente

3. Mise à jour

Les librairies et Framework utilisé ont été mis à jour vers les dernières versions à ce jour, les plug-ins jQuery suivant on été ajouté :

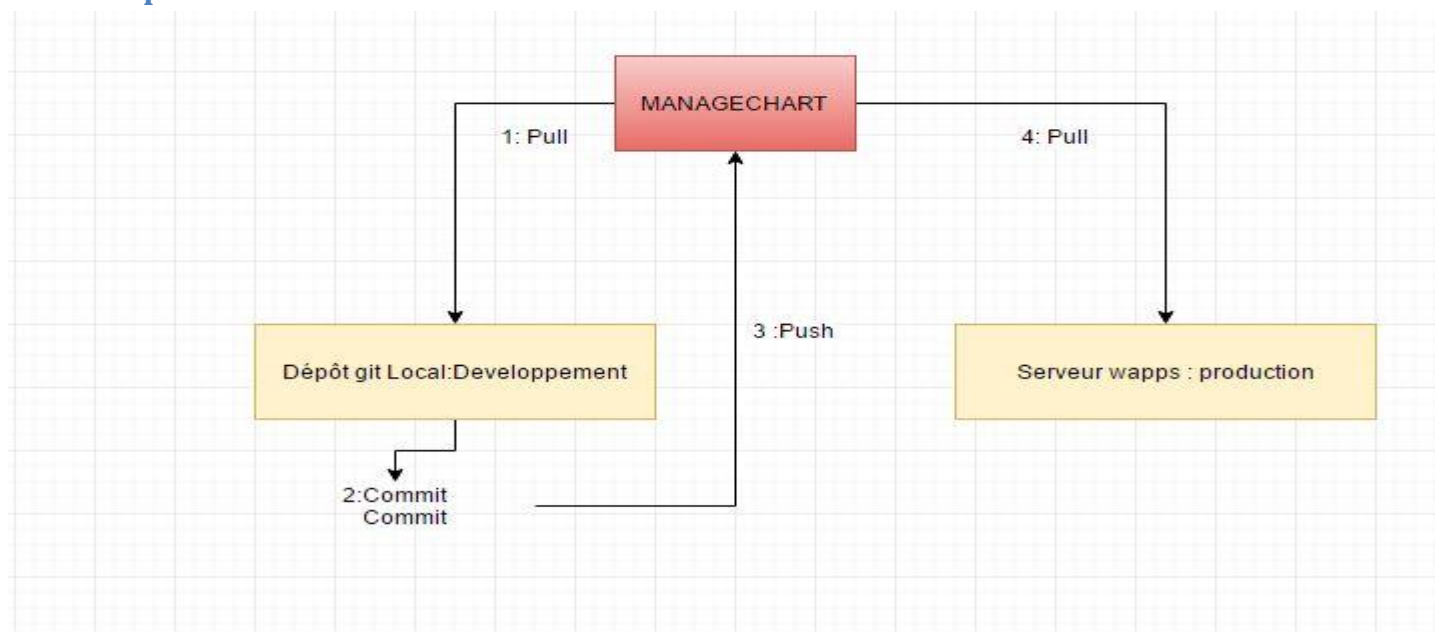
- +jquery-tablesorter permettant le tri paramétrable de tableaux.
- +Installation de l'extension DataTable Sort pour le tri des dates chronologiquement pour des données enregistrées.
- +Toastr permettant l'ajout et le paramétrage de notification lors d'erreur
- + Mise à jour à jour des librairies highcharts et highstock en passant de la version 4.2.5 à la version 5.0.11

4. Tests

Les tests ont été réalisés à l'aide de :

- Firebug
- Web debug toolbar de symfony
- L'extension du navigateur Google chrome Xdebug helper

5. Dépôt GIT

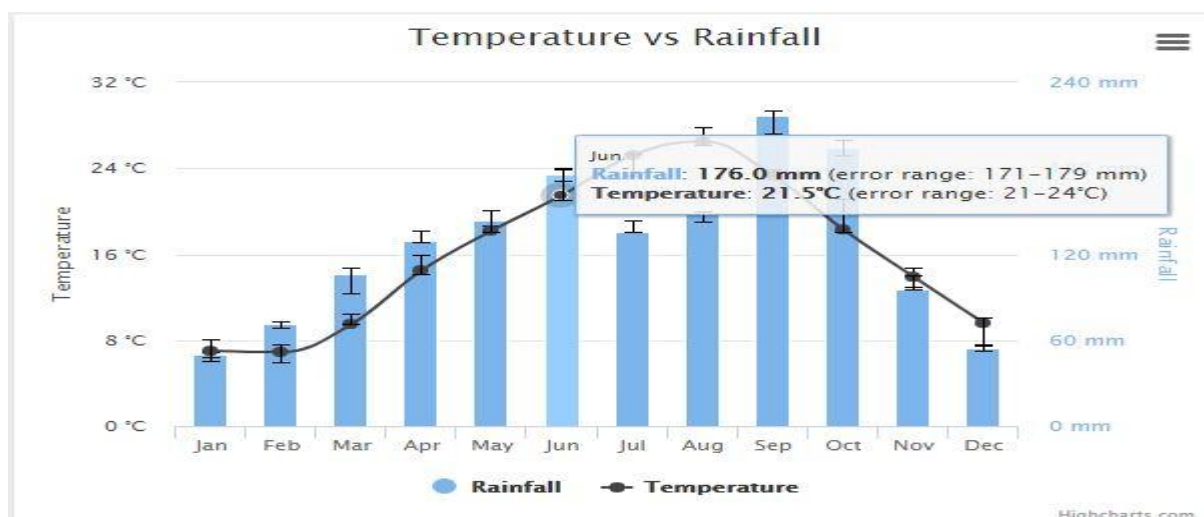


Concernant le cas du dépôt de git, jusqu'à présent je me contente de travailler que dans le dépôt local et lors d'une modification ou l'ajout d'une fonctionnalité un commit est effectué pour enregistrer l'état du code, c'est performant puisqu'on peut revenir facilement à une modification déjà enregistré, j'ai également participé à des déploiements sur le serveur de production avec M. Rouan.

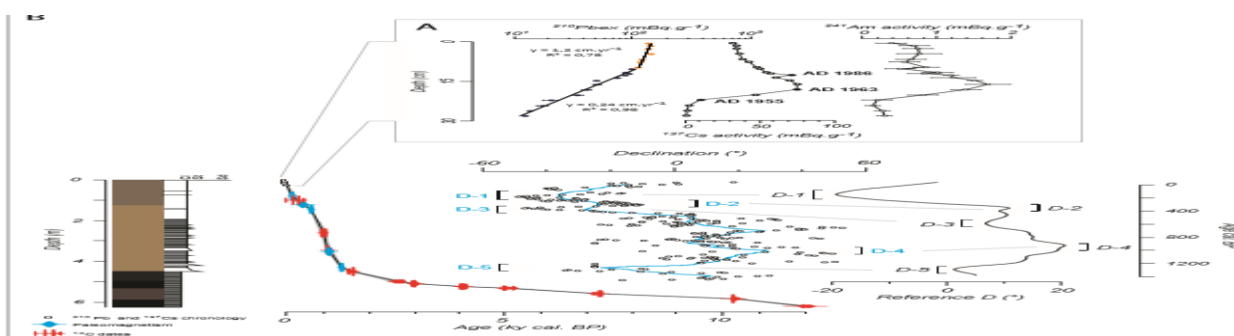
6. Poursuite du stage en Juillet et Aout

Le projet Roza nécessite la mise en place d'autres fonctionnalités comme par exemple :

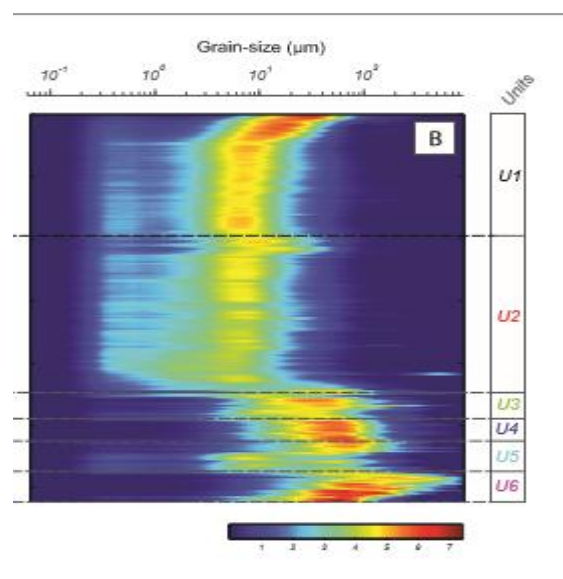
+ Le graphe de type ErrorBar



+ Le graphe Multi axe x (axe déporté)

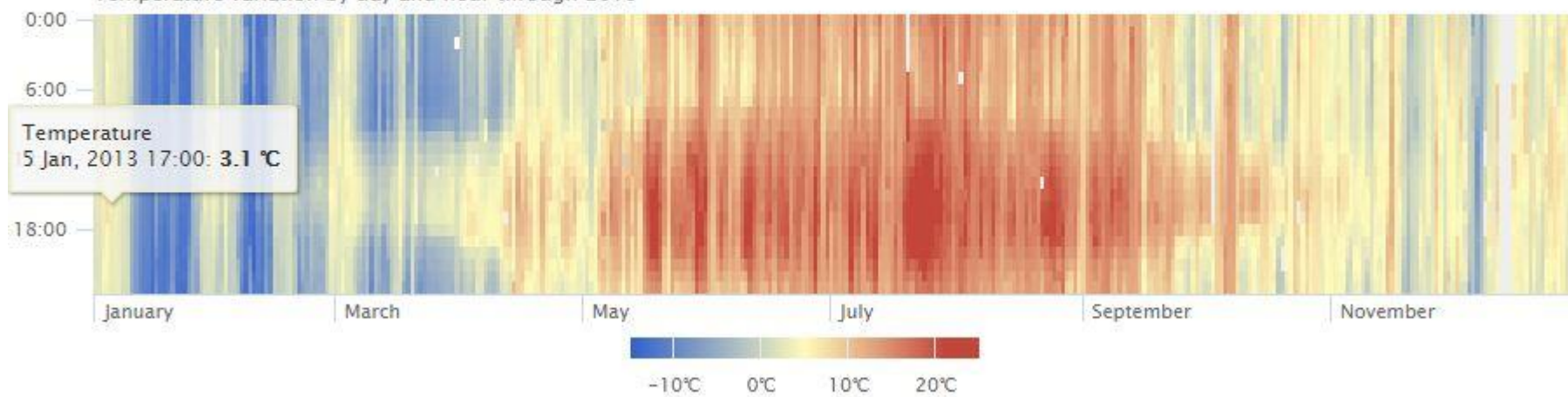


+ Le graphe HeatMap



Highcharts heat map

Temperature variation by day and hour through 2013



Highcharts.com

Conclusion

L'application « ManageChart » étant un projet existant et un stage à la hauteur de mes attentes m'a apporté des compétences complémentaires en symphony 2 ainsi que les librairies JS, une bonne manipulation des outils de débogage ainsi que le logiciel de gestion de version GIT .

Le code déjà écrit étant aussi enrichissant que ce que je lui ai apporté ainsi que la documentation sur l'ensemble de l'application m'a permis de me repérer facilement à travers les lignes de code et comprendre différents choix effectués par des anciens stagiaires , puisque le travail de développeur consiste aussi en la collaboration .

J'ai débuté mon stage en me familiarisant avec le Framework Symfony, la compréhension du rôle de la librairie highcharts et son interaction avec le code source PHP.

J'ai poursuivi par une phase de développement en deux temps :

- +La correction de différents bogues de l'application
- + L'implémentation de fonctionnalités.

Bibliographie

→ Liens

<http://c-maneu.developpez.com/tutorial/web/php/symfony/intro/>

<https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2/installer-un-bundle-grace-a-composer>

[https://fr.wikipedia.org/wiki/Doctrine_\(ORM\)](https://fr.wikipedia.org/wiki/Doctrine_(ORM))

<https://www.draw.io>

<https://www.highcharts.com>

<http://symfony.com>

<http://letg.cnrs.fr>

→ Partage :

<https://tucuxi.univ-brest.fr/projects/managechart/wiki>

→ Documents :

Stage/CARTAHU / Manuel d'administration

Rapport de stage Maxime Collin et Thomas BOUINOT