**Partie analyse**

# ***Basic use :***

# ***Line chart***

# ***Code de la configuration***

ArrayList<String> months = **new** ArrayList<String>();  
months.add(**"January"**);  
months.add(**"February"**);  
months.add(**"March"**);  
months.add(**"April"**);  
months.add(**"May"**);  
months.add(**"June"**);  
months.add(**"July"**);  
  
LineChartConfig lineChartConfig = **new** LineChartConfig();  
lineChartConfig.data()  
 .labelsAsList(months)  
 .addDataset(**new** LineDataset().label(**"First data set"**).fill(**false**))  
 .addDataset(**new** LineDataset().label(**"Second data set"**).fill(**false**))  
 .addDataset(**new** LineDataset().label(**"Hidden data set"**).hidden(**true**))  
 .and()   
 .options()  
 .responsive(**true**)  
 .title()  
 .display(**true**)  
 .text(**"Charts js real time line"**)  
 .fontColor(**"#f5b041"**)  
 .position(Position.***TOP***)  
 .and()  
 .hover()  
 .mode(Hover.Mode.***DATASET***)  
 .and()  
 .scales()  
 .add(Axis.***X***, **new** CategoryScale()  
 .display(**true**)  
 .scaleLabel()  
 .display(**true**)  
 .labelString(**"Month"**)  
 .and()  
 .position(Position.***TOP***))  
 .add(Axis.***Y***, **new** LinearScale()  
 .display(**true**)  
 .scaleLabel()  
 .display(**true**)  
 .labelString(**"Value"**)  
 .and()  
 .ticks()  
 .suggestedMax(250)  
 .suggestedMin(-100)  
 .and()  
 .position(Position.***RIGHT***))  
 .and()  
 .done();

### ***JSON***

{

"type": "line",

"data": {

"labels": [

"January",

"February",

"March",

"April",

"May",

"June",

"July"

],

"datasets": [

{

"data": [

52,

74,

100,

89,

72,

76,

46

],

"label": "First data set",

"fill": false,

"backgroundColor": "#F7464A",

"borderColor": "#F7464A"

},

{

"data": [

58,

26,

91,

67,

85,

13,

85

],

"label": "Second data set",

"fill": false,

"backgroundColor": "#46BFBD",

"borderColor": "#46BFBD"

},

{

"data": [

91,

32,

93,

35,

17,

2,

91

],

"label": "Hidden data set",

"hidden": true,

"backgroundColor": "#FDB45C",

"borderColor": "#FDB45C"

}

]

},

"options": {

"responsive": true,

"title": {

"display": true,

"position": "top",

"text": "Charts js real time line",

"fontColor": "#f5b041"

},

"hover": {

"mode": "dataset"

},

"scales": {

"xAxes": [

{

"type": "category",

"display": true,

"position": "top",

"scaleLabel": {

"display": true,

"labelString": "Month"

}

}

],

"yAxes": [

{

"type": "linear",

"display": true,

"position": "right",

"scaleLabel": {

"display": true,

"labelString": "Value"

},

"ticks": {

"suggestedMin": -100,

"suggestedMax": 250

}

}

]

}

}

}

## ***Pie chart***

### ***Code de la configuration***

DonutChartConfig config = **new** DonutChartConfig();  
config.data()  
 .labels(**"Red"**, **"Green"**, **"Yellow"**, **"Grey"**, **"Dark Grey"**)  
 .addDataset(**new** PieDataset().label(**"Dataset 1"**))  
 .and();  
config.options()  
 .rotation(Math.***PI***)  
 .circumference(Math.***PI***)  
 .responsive(**true**)  
 .title()  
 .display(**true**)  
 .text(**"Charts.js gauge donut chart"**)  
 .fontColor(**"#cb4335"**)  
 .and()  
 .animation()  
 .animateScale(**false**)  
 .animateRotate(**true**)  
 .and()  
 .done();  
  
String[] colors = **new** String[] {**"#F7464A"**, **"#46BFBD"**, **"#FDB45C"**, **"#949FB1"**, **"#4D5360"**};  
  
List<String> labels = config.data().getLabels();  
**for**(Dataset<? , ?> ds : config.data().getDatasets()){  
 PieDataset lds = (PieDataset) ds;  
 lds.backgroundColor(colors);  
 List<Double> data = **new** ArrayList<>();  
 **for**(**int** i = 0; i < labels.size(); i++){  
 data.add((**double**) (Math.*round*(Math.*random*() \* 100)));  
 }  
 lds.dataAsList(data);  
}  
  
**chart** = **new** ChartJs(config);  
**chart**.setJsLoggingEnabled(**true**);  
**chart**.addClickListener((a, b) -> {  
 PieDataset dataset = (PieDataset) config.data().getDatasets().get(a);  
 Notification.*show*(**"Click on "**+b+**" value : "**+dataset.getData().get(b));  
});

{

"type": "doughnut",

"data": {

"labels": [

"Red",

"Green",

"Yellow",

"Grey",

"Dark Grey"

],

"datasets": [

{

"data": [

15,

6,

93,

1,

20

],

"label": "Dataset 1",

"backgroundColor": [

"#F7464A",

"#46BFBD",

"#FDB45C",

"#949FB1",

"#4D5360"

]

}

]

},

"options": {

"responsive": true,

"title": {

"display": true,

"text": "Charts.js gauge donut chart",

"fontColor": "#cb4335"

},

"animation": {

"animateRotate": true,

"animateScale": false

},

"rotation": 3.1415926535898,

"circumference": 3.1415926535898

}

}

## ***Classes***

La librairie Chart-Js propose plusieurs classes permettant de créer les différentes configurations des Charts, par exemple : LineChartConfig pour les graphiques linéaires. Toutes les classes de configuration implémentent l’interface ChartConfig. Cette *interface* hérite de l’interface JsonBuilder qui possède la méthode *buildJson()* qui nous permettra par la suite de créer les données JSON associées aux différents charts.

## ***Manière d’implémentation***

Proposer dans le configurator un formulaire où il sera possible, pour un client, de configurer son graphique.

Une fois les charts créés nous les mettrons dans le configurator[[1]](#footnote-1). Mon point de vue serait de créer un formulaire qui varie en fonction du type de chart, en effet, tous les charts ne possèdent pas la même configuration. Ce formulaire proposerait de choisir un type, le nom de l’axe des Y , le nom de l’axe des x, une source de données, …

## ***Différents type de Charts proposés par Chart-Js***

* 1. Line
  2. Bar
  3. Pie & doughnut charts
  4. Bubble chart
  5. Radar chart

## ***Comment l’enregistrer dans la database ?***

Une des manières de stocker les charts dans la database serait de créer une table pour un Chart . Celui-ci serai caractérisé par :

* Un type (line, bar, pie, …)
* Une collection de données
* Une collection d’options ?
* Un axe X (facultatif pour les PIE)
* Un axe Y (facultatif pour les PIE)
* Un label pour l’axe X (facultatif pour les PIE)
* Un label pour l’axe Y (facultatif pour les PIE)
* Un minimum (facultatif pour les PIE)
* Un max (facultatif pour les PIE)
* Un titre
* Un identifiant artificiel (auto-généré)

## ***Profil utilisateur***

2 profils utilisateur peuvent accéder au configurator : le client et le consultant, L’authentification est donc requise.

### ***Besoin de l’utilisateur***

L’utilisateur veut pouvoir : créer des graphiques en fonction de données, les paramétrer en fonction de son type (Pie, line, bar, …) et les afficher sur un Dashboard.

Certains critères sont spécifiques à un type de graphique. Ainsi l’utilisateur pourra définir un style propre à son graphique : animation, couleurs, titre, nomination de l’axe X et Y (Bar, line), et autre.

## ***Répondre aux besoins***

Un des moyens de répondre aux besoins de l’utilisateur serait de créer un formulaire avec un canevas. Dans un premier temps, le formulaire reprendrait toutes les caractéristiques communes à chaque graphique (titre, couleurs, source des données, …) et dans un second temps, faire un formulaire dynamique, les champs s’activeraient en fonction du type de graphique sélectionné. (Plus rouage + giplet)

Une fois le graphique créé, on doit l’enregistrer dans la base de données. L’utilisateur ne devra pas recommencer ses graphiques à chaque connexion, il pourra les récupérer et les modifier à sa guise. Un moyen de faire serait de demander au configurator de généré le json associés aux exigences et pouvoir l’afficher sur la page html via Javascript. Il faudrait peut-être parser le json pour pouvoir l’enregistrer en tant que modèle dans la base de données.

## ***Données communes intéressantes***

* + 1. Les configurations
       1. Chaque « chart » possède sa propre configuration (classe en java), par exemple pour créer un Dounught chart il faut instancier la classe DonutChartConfig, pour le « Line chart » LineChartConfig, etc.
    2. Les données
       1. Chaque graphique est basé sur une source de données, on commence par définir les éléments qui seront présent sur l’axe des X, ensuite il faut y ajouter un dataset propre à chaque type de graphique (new PieDataSet() pour un Pie chart par exemple). Les dataset sont les données du chart.
    3. Les options
       1. Chaque chart peut être ou non customisable en fonction des options choisies. On peut choisir le titre, la position du titre, lui ajouter des styles, des animations, etc.

1. ***Modèle relationnel***

Possibilité d’enregistrer JSON dans database oracle dans un champ, faire des requêtes facilement à l’intérieur du JSON

1. Outil permettant à une personne tiers de configurer, créer différents Widget sans en connaitre l’implémentation, via formulaire, input … [↑](#footnote-ref-1)