UNIVERSIDAD
REY JUAN CARLOS

INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2017/2018

Trabajo Fin de Grado/Máster

SAKURA

ANGULAR 4 - ELASTICSEARCH - DASHBOARD INTERFACE

Autor : Ismael Slimane Zubillaga

Tutor : Dr. Jesús M. González-Barahona

Trabajo Fin de Grado/Máster

Título del Trabajo con Letras Capitales para Sustantivos y Adjetivos

Autor : Nombre del Alumno/a

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Contents

List of Figures

Chapter 1

Introducción

En este capítulo se introduce el proyeto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes[1].

Aconsejo a todo el mundo que mire y se inspire en memorias pasadas. Las mías están todas almacenadas en mi web del GSyC[2].

1.1 Descripción del Problema

1.2 Objetivo Principal y Estructura

En esta sección se debería introducir la esctura de la memoria. Así:

- En el primer capítulo se hace una intro al proyecto.
- En el capítulo **??** (ojo, otra referencia automática) se muestran los objetivos del proyecto.
- A continuación se presenta el estado del arte.
- ...

1.3 Disponibilidad del Software

[1] `http://www.tallerdeescritores.com/errores-de-escritura-frecuentes`
[2] `https://gsyc.urjc.es/~grex/pfcs/`

Chapter 2

State of Art

2.1 ElasticSearch

Elasticsearch is an open-source, *broadly-distributable*, *readily-scalable*, *enterprise-grade* search engine based on Lucene[1]. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

2.1.1 History

Compass was the predecessor to Elasticsearch and it was created by **Shay Banon** in 2004. Through the implementation of the third version of Compass, came the necessity of a *"scalable search solution"*. This necessity would have meant a lot of work rewriting big pieces of code, so Shay decided to build *"a solution built from the ground up to be distributed"* and used a common interface, *JSON* over *HTTP*, available for other programming languages, and not only *Java*.

The first verision of Elasticsearch was released on February 2010.

[1] `https://en.wikipedia.org/wiki/Apache_Lucene`

2.1.2 Basic Concepts

All the following concepts definitions are extracted from the Elasticsearch documentation web page[2].

- **Near Realtime (NRT)**: Elasticsearch is a near real time search platform. The practical meaning of this is that there is a little latency (about one second) from the moment you store a new document, to the moment it becomes available for searching.

- **Index**: An **index** is a set of documents that share some type of characteristics. For example, you can have an index for a shop product, another index for employee data and another for bill data. An index is identified by its name (in lowercase), and this name is used for several operations as: searching, deleting, updating, etc.

- **Shards & Replicas**: The index data can reach a large size exceeding the available physical memory. But this problem can be fixed by defining multiple **shards**. Each shard is a portion of data from the index data. The number of shards is defined when the index is created.

  But there is still another potential problem. It always exists the possibility to have a failure on the network system and to loose a shard/node, so it is advisable to have **replicas** of our data. A replica is a copy of the index shards.

- **Type**: We can see a **type** like an object class, with fields of different data-types.

  In Elasticsearch a type is defined by its *name* and its *mappings*. The *mappings* are a schema of the type, where it's defined the properties that our type has, and the data-type of each property, such as *integer, string, etc.*.

- **Document**: As we said about *types* being like classes, we could see a **document** like a record from a single class. Using the same example we used for *indexes*, you can have a document for a single product, another document for a single employee and yet another for a single bill.

[2] `https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html`

- **Node**: We can see a **node** like a single server, as a single unit that along with other nodes, make up a cluster. This node take part on cluster's indexing and searching tasks.

- **Cluster**: As we said in the previous definition, a **cluster** is made up with multiple nodes(servers). The cluster stores all the data and allows to search and index all this data across all nodes. A cluster is a collection of one or more nodes (servers) that together holds your entire data and provides federated indexing and search capabilities through all nodes.

2.2 Kibana

Kibana is an *Elasticsearch* open-source plug-in. It mainly works with Elasticsearch indexed data in order to represent it into different types of visualizations such as bar charts, scatter plot charts, pie charts, ...

Kibana, as an exploration tool, can be used to log and time series analytics, application monitoring, and operational intelligence use cases.

We can find another similar applications such as *Grafna*[3], *incubator-superset*[4] and *Tableau*[5].

2.2.1 Main Features

In the Figure 2.1, we can see the Kibana's *nav-bar*. Now we are going to speak about a couple of features.

- **Visualize**:

  Visualize allows you to represent data with a specific type of chart. The represented data will be chosen from the available index on Elasticsearch. In addition to the Elasticsearch index, we have to choose which type of *aggregations* are we going to use in order to extract our data. We can see an example on Figure 2.2.

- **Dashboard**:

[3] `https://grafana.com/`
[4] `https://github.com/apache/incubator-superset`
[5] `https://www.tableau.com/`

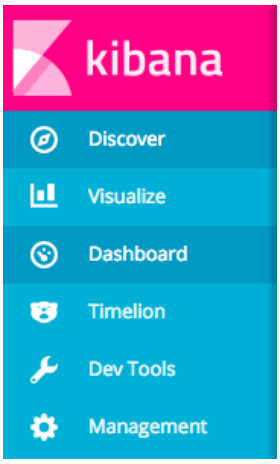

Figure 2.1: Kibana's nav-bar.

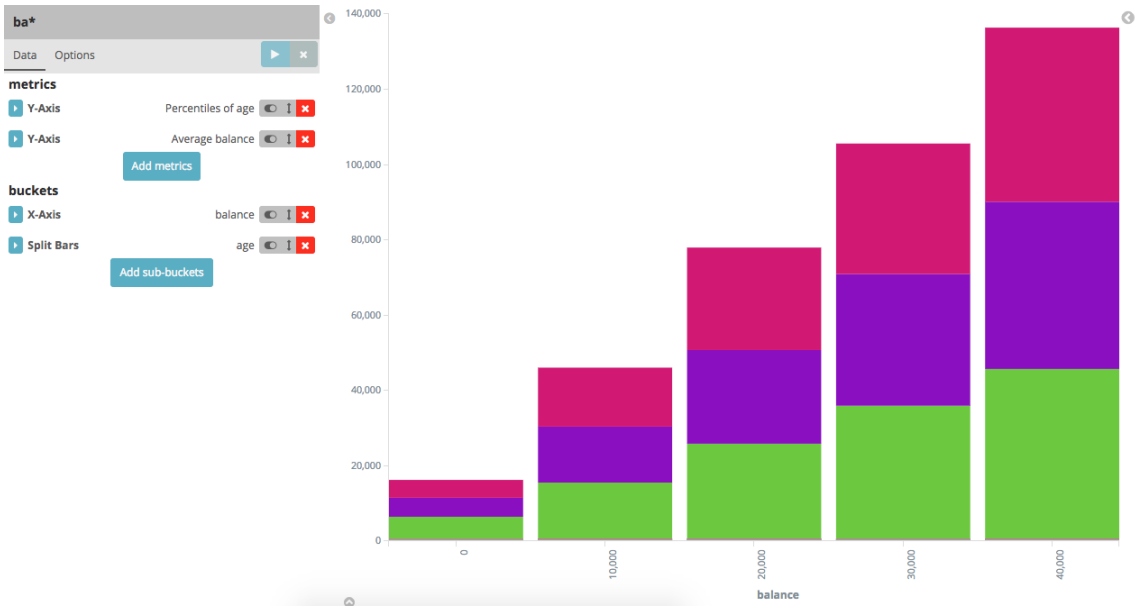

Figure 2.2: Visualization example.

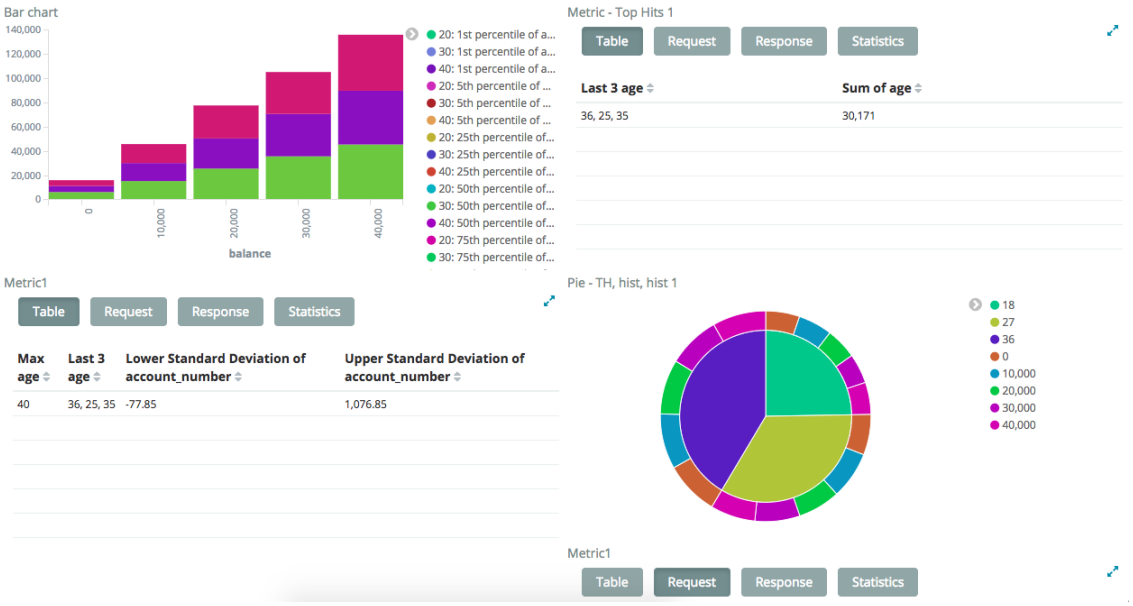

Figure 2.3: Dashboard example.

The Kibana's dashboard functionality allows us to display a collection of saved visualizations and organize them by dragging and dropping. We can see an example on Figure 2.3.

2.3 GitHub

"**GitHub** is a *Web-based* **Git version control repository hosting service**". Github is used mainly for programming code. In addition to Git functionalities as *deistributed version control* and *source code management*, Github provides more features such as collaboration features for *bug tracking* and for other porposes, task management, and *wikis* for documentation porposes.

For the current project, a repository was created on *Github* with the name of '*Angular-ElasticSearch-Dashboard_Interface*'[6].

2.4 Webpack

"**Webpack** is an *open-source JavaScript* module bundler". The main reason for using Webpack is to have a modular structure on your web application project. This allows you to have a cleaner

[6] `https://github.com/islimane/Angular-ElasticSearch-Dashboard_Interface`

code and make it more reusable and scalable. Webpack can be used from the command line or be configured with a config file named *webpack.config.js*.

2.5 Angular

Angular is an open source *web application platform* based on *Typescript* and developed by Google and by a community of individuals and corporations. Angular is commonly referred to as Angular 2.0 or later versions.

2.5.1 Versions

Before Angular was created, there was a previous version called *AngularJS* or Angular1.X. Angular was created as a complete rewrite of AngularJS.

- 2.0.0:

  This was the first version of Angular. Its announcing was made at the *ng-Europe* conference on September 2014. This version was build up from the ground and these were the main characteristics:

  - Introduced the components hierarchy.
  - Modules for core functionality, improving the speed of the Angular core.
  - Possibility of using *TypeScript* language. The use of this language is recommended by the Angular team.
  - Improved *dependency* injection.
  - Dynamic loading.
  - Reactive programming support using RxJS.

  And much more features. The final version was released on September 14, 2016.

- 4.0.0:

  This version was called *Angular 4* and was announced on 13 December 2016. Angular 3 was skipped due to some features already present on version 3.0.0 and to avoid confusion. This version introduced:

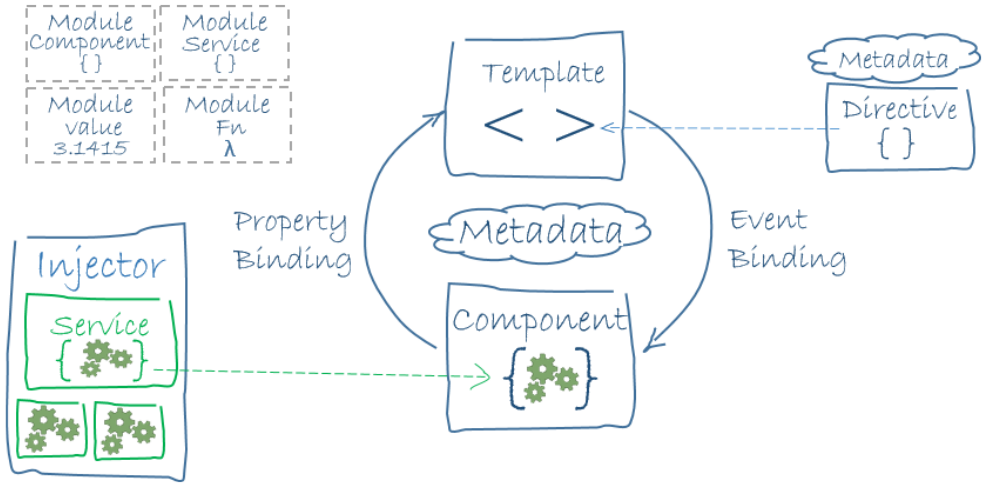

Figure 2.4: Architecture of an Angular application

- *Http* client.
- A new *Rout Live Cycle*.
- Conditionally disable animations.

- 5.0.0:

  This version was released on November 2017 and the main introduced features were:

  - Support for progressive Web apps.
  - A build optimizer and improvements related to *Material Design*.

2.6 TypeScript

"**TypeScript** is a free and open-source programming language developed and maintained by Microsoft. It is a strict syntactical *superset* of *JavaScript*, and adds optional static typing to the language".

2.6.1 History

Before it was made public, *TypeScript* gone through a two years of development process until it reached the 0.8 version. Then, was released on October 2012. Initially it had a lack of *IDE*

support, until 2013 when some IDE's began to have support for this language, such as *Eclipse* with a dedicated pug-in, and some text editors such as *Sublime*, *Atom*, etc.

The reason TypeScript was created came from the necessity of a front-end programming language that fulfill the task of the development of complex and large-scale front-end applications, since JavaScript has shortcomings on that sense.

TypeScript is based on the *EMACScript* approach, the reason why is because it is a standard and because it has support for *class-based programming*.

2.6.2 Language Features

- Type annotations and compile-time type checking:

  TypeScript provides an optional static typing, with annotations, that is checked at compile time. If this typing it is not used, then the Javascript's dynamic typing is used. Here is an example:

```
1 function planetMoons(planetName: string, moonsNum: number, spanish:
     boolean): any {
2   if(spanish){
3       console.log('El planeta ' + planetName + ' tiene ' + moonsNum +
           ' lunas');
4   }else{
5       console.log(planetName + '  planet has ' + moonsNum + ' moons');
6   }
7
8   return null;
9 }
```

  string, *boolean* and *number* are primitive types. For dynamic types it's used *any*.

- Type inference.
- Type erasure.
- Interfaces.
- Enumerated type.
- Mixin.

- Generic.
- Namespaces.
- Tuple.
- Await.
- Classes (backported from EMACScript 2015).
- Modules (backported from EMACScript 2015).

2.7 JavaScript

"**JavaScript**, often abbreviated as JS, is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production". Generally, the main purpose of Javascript is to make pages interactive and develop online programs such as online games. Javascript is supported by most of the web browsers, for that reason, the majority of the websites employ it. There are multiple engines for Javascript, but all are standardized with the *EMACScript* specification.

2.7.1 History

Javascript came with the necessity of dynamism on web browsers. The founder of Netscape Communications, Marc Andreessen, thought that HTML needed a "glue language" to assemble component like images or plug-ins into the web markup. In 1995, Netscape Communications began to use *Java* as a static programming language for Netscape Navigator, so the scripting language they had being searching for, had to be similar to Java and would complement it. Then, on May 1995, Brendan Eich wrote in 10 days this scripting language in order to compete against competing proposals. The first time this scripting language was called JavaScript was on December 1995, when the Netscape Navigator 2.0 beta 3 was released.

2.7.2 Syntax

These are a few examples to illustrate the JavaScript syntax:

- Variables:

```
var x; // defines the variable x and assigns to it the special value "
    undefined" (not to be confused with an undefined value)
var y = 2; // defines the variable y and assigns to it the value 2
var z = "Hello, World!"; // defines the variable z and assigns to it a
     string entitled "Hello, World!"
```

- Printing:

```
console.log("Hello World!");
```

- Functions:

```
function add(a, b){
    return a + b;
}
add(1, 2); // returns 3
```

2.8 Lodash

Lodash is a *JavaScript* library that provides extra functions for tasks that are often required on data structure managing, a lot of them, not implemented on the main JavaScript functionality.

For example:

- **_.difference(array, [values])**: Creates an array of unique array values not included in the other provided arrays using SameValueZero for equality comparisons.

```
_.difference([1, 2, 3], [4, 2]);
// returns [1, 3]
```

- **_.pluck(collection, path)**: Gets the property value of path from all elements in *collection*.

```
var users = [
  { 'user': 'barney', 'age': 36 },
  { 'user': 'fred',   'age': 40 }
```

```
];

_.pluck(users, 'user');
// returns ['barney', 'fred']
```

As we can see, Lodash functions are called through the global variable "_". All this functions can be found on the Lodash documentation[7] page. This JavaScript library saves a lot of time when managing data structures.

There are other similar libraries such as *UnderscoreJS*[8].

2.9 jQuery

"*jQuery* is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML". jQuery it's the most used JavaScript library. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.

2.9.1 History

jQuery was originally released in January 2006 at BarCamp NYC by John Resig and was influenced by Dean Edwards' earlier cssQuery library. It is currently maintained by a team of developers led by Timmy Willison.

2.9.2 Usage Examples

- HTML:

```
<p>Hello</p>
```

- JavaScript:

```
$("p").clone().add( "<span>Again</span>" ).appendTo(document.body);
```

[7] https://lodash.com/docs/3.10.1
[8] http://underscorejs.org/

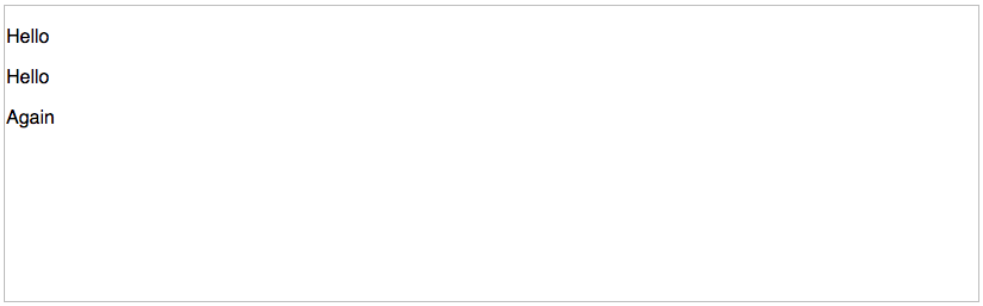

Figure 2.5: View result.

- Result:

  See the result on Figure 2.5

2.10 HTML5

"**HTML5** is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard".

The *World Wide Web Consortium* released HTML5 on October 2014, their goal was "to improve the language with support for the latest multimedia".

2.10.1 Features

These are some of the feature introduced in HTML5:

- **Markup**:

  HTML5 introduces some elements that are commonly used on modern websites, they usually are replacements of old HTML elements, between these new elements we can find: <*nav*>, <*footer*>, <*audio*>, <*video*>, ...

- **New APIs**:

  HTML5 introduces *application programming interfaces* or *APIs* in addition to the markup elements that we have seen, such as *Canvas*, *Timed Media Playback*, *Drag and Drop*, *MIME type*, *Web Messaging*, *Web storage*, ... In this project we have mainly used *Canvas* through the *ChartJS* library.

2.11 ChartJS

ChartJS is a JavaScript library for producing dynamic, interactive data visualizations in web browsers, this is achieved by using the *HTML5 canvas* element. With ChartJS you create different charts such as pie charts, bar charts, line charts, etc. You can find visual examples of each type on the ChartJS samples page[9].

2.11.1 Usage Examples

We have to instantiate the *Chart* class in order to create a chart, an then assign it to a *canvas* element on the DOM. The field *type* indicates which chart are we going to use, and the field *data* stores our chart info. Here is an example:

- HTML:

```
<canvas height='50' id='myChart' width='200'></canvas>
```

- JavaScript:

```
var ctx = document.getElementById("#myChart");
var myChart = new Chart(ctx, {
  type: 'bar',
  data: {
    labels: ["Red", "Blue", "Yellow"],
    datasets: [{
      label: '# of Votes',
      data: [12, 19, 3, 5, 2, 3],
      backgroundColor: [
        'rgba(255, 99, 132, 0.2)',
        'rgba(54, 162, 235, 0.2)',
        'rgba(255, 206, 86, 0.2)'
      ],
      borderColor: [
        'rgba(255,99,132,1)',
        'rgba(54, 162, 235, 1)',
        'rgba(255, 206, 86, 1)'
      ],
```

[9] http://www.chartjs.org/samples/latest/

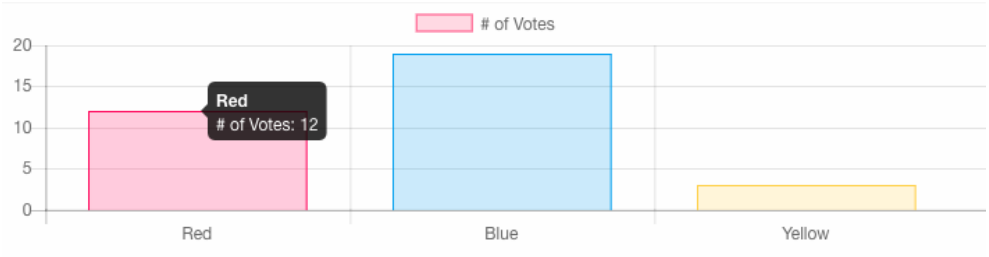

Figure 2.6: View result.

```
19        borderWidth: 1
20      }]
21    }
22 });
```

- Result:

  See the result on Figure 2.6

There are a lot of libraries that allows you to render charts or graphs on a web page[10] such as *C3.js*, *Plotly.js*, etc. In this project we used ChartJS because it adapted well to the project requirements.

2.12 SASS

"**Sass** is a scripting language that is interpreted or compiled into *Cascading Style Sheets* (CSS)". Sass uses two types of syntax. The first is original syntax, similar to *Haml*[11]. The second is *SCSS*, a syntaxt that uses blocks formatting like *CSS*

2.12.1 Features

- **Variables**:

  Sass allows defining variables using a dollar sign ($).

```
1 $primary-color: #3bbfce;
2
```

[10] https://blog.sicara.com/compare-best-javascript-chart-libraries-2017-89fbe8cb112d
[11] https://en.wikipedia.org/wiki/Haml

```
.content-navigation {
  border-color: $primary-color;
  color: darken($primary-color, 10%);
}
```

- **Block Nesting**:

  Sass allows nesting different style blocks, this a improves the code structure, makes it more readable and saves lines of code.

```
table.hl {
  margin: 2em 0;
  td.ln {
    text-align: right;
  }
}
```

- **Loops**:

  With Sass you can create loops of style blocks, this feature helps to save code when you have similar *id's* or *classes*. Her is an example:

```
$squareCount: 3
@for $i from 1 through $squareCount
  #square-#{$i}
   background-color: red
   width: 50px * $i
   height: 120px / $i
```

- There are more Sass features such as *arguments*, *selector inheritance*, ... All of these features can be found on the Sass documentation web page[12].

2.13 Flexbox

"The *CSS3 Flexible* Box, or **Flexbox**, is a layout mode intended to accommodate different screen sizes and different display devices". Flexbox is very useful when you have to organize

[12] http://sass-lang.com/documentation/file.SASS_REFERENCE.html

Figure 2.7: no-wrap example.

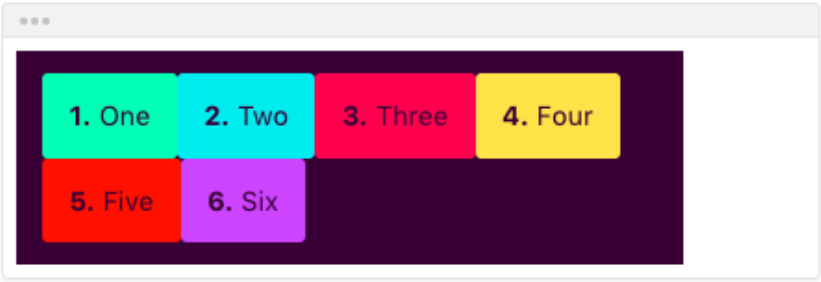

Figure 2.8: wrap example.

your application view, so for example, sticking a *footer* to the bottom of a page, designing a *navigation bar*, creating a custom grid, ... is much easier using Flexbox.

Related to the Flexbox concept, it allows to manage an element width and the height to improve its fitting in the available space on the display regardless of the device that is being used. This elements are shrunken to prevent overflow or expanded to fill remaining space.

2.13.1 Usage Examples

In order to use Flexbox styles, we have to apply the style "*display: flex*" in our container element. Let's see a few Flexbox usage examples.

- **flex-wrap**:

  Defines if flexbox items appear on a single line or on multiple lines within a flexbox container.

```
1 flex-wrap: nowrap;
2 flex-wrap: wrap;
```

- **justify-content**:

Figure 2.9: flex-end example.

Figure 2.10: center example.

Defines how flexbox items are aligned according to the main axis, within a flexbox container.

```
justify-content: flex-end;
justify-content: center;
```

Chapter 3

Desarrollo

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

3.1 Metodología SCRUM

3.2 Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a "vista de pájaro").

Por ejemplo, puedes verlo en la figura 3.1.

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

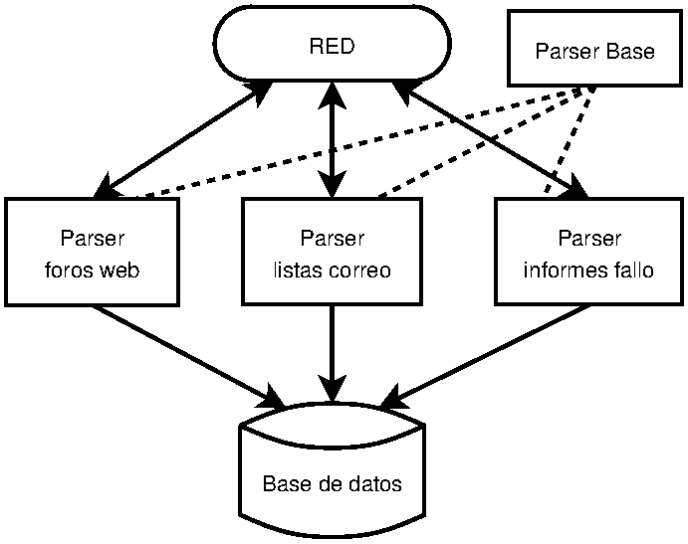

Figure 3.1: Estructura del parser básico

Chapter 4

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Chapter 5

Conclusiones

5.1 Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

5.2 Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

5.3 Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

5.4 Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Apéndice A

Manual de usuario

Bibliography