

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN



Desarrollo de un editor de visualizaciones de propiedades de historiales de wikis

Autor: Leonardo Testa

Tutor: Eugenio Scalise

junio, 2017

Índice general

Índice de figuras	2
1. Introducción	3
2. Tecnologías para la visualización de datos en web	4
2.1. SVG vs Canvas	5
2.2. Bibliotecas basadas en Canvas	5
2.3. Bibliotecas basadas en SVG	6
2.4. D3 (Data-Driven Documents)	7
2.5. Evaluación de Bibliotecas	8
3. Tecnologías para el desarrollo web	10
3.1. Arquitectura	10
3.1.1. Angular	11
3.1.2. jQuery	13
3.2. Diseño	14
3.2.1. Angular Material	14
3.2.2. Bootstrap	15
3.3. Utilidades	16
3.3.1. Lodash	16
3.3.2. Moment	17
4. Propuesta de Trabajo Especial de Grado	19
4.1. Motivación e identificación del problema	19
4.2. Objetivos del trabajo	19
4.2.1. Objetivo General	19
4.2.2. Objetivos Específicos	19
4.3. Estrategia de solución y método de desarrollo ágil a utilizar	20
4.4. Trabajos similares, diferencias y ventajas de la solución a desarrollar	21
4.5. Herramientas	25
4.6. Requerimientos	25
4.7. Prototipo de interfaz	26
4.8. Planificación de las actividades	29
Bibliografía	30

Índice de figuras

2.1. Comparación de bibliotecas de visualización	9
3.1. Ciclo de vida de una aplicación SPA	11
4.1. Ejemplo de pizarra de Github	20
4.2. Información general proporcionada por la herramienta WikiHistory sobre el artículo 'Chocolate'.	21
4.3. Gráfica de tipo torta de WikiHistory, que refleja el porcentaje de ediciones anónimas, menores y otras, sobre el artículo 'Chocolate'.	22
4.4. Línea de tiempo, que proyecta visualmente las ediciones totales, menores y anónimas del artículo 'Chocolate', que pueden ser filtradas por año, mes o semana.	22
4.5. Visualización tipo barra sobre el artículo 'Chocolate', en donde el eje 'y' representa el número de visitas, y el eje 'x' representa la fecha.	23
4.6. Uso de Chromogram donde el color morado representa al artículo de US Ships (en donde el título del artículo empieza con 'USS')	23
4.7. Explicación del mecanismo de visualización de History Flow	24
4.8. En la página 'Chocolate' se observa un patrón en zigzag que demuestra una guerra de ediciones	24
4.9. Demostración de History Graph	25
4.10. Interfaz para iniciar sesión	26
4.11. Interfaz de la vista principal	27
4.12. Interfaz de la vista detallada de un wiki	27
4.13. Interfaz de la vista de creación de una visualización	28

Capítulo 1

Introducción

Un Wiki es un sitio web que puede ser modificado por cualquier persona, la finalidad de esto es que la misma comunidad aporte una mejora. Mayormente los wikis son de carácter informativo, un claro ejemplo es Wikipedia, que es la enciclopedia online más popular del mundo basado en wiki. Al ser un sitio que acepte colaboración se ve necesario a llevar un historial de los cambios realizados por los usuarios, logrando un control en las ediciones de estos. De una edición se pueden sacar algunas propiedades importantes como: el autor de la edición, que se representa con una dirección IP cuando es anónimo y con un nombre de usuario en caso contrario, la cantidad de texto editado y la fecha y hora en que realizó la edición. Hoy en día existen tantas personas colaborando en estos sitios que hacen que el historial de ediciones se haga muy extenso y difícil de comprender, la abundancia de datos provoca complejidad en su búsqueda e interpretación, lo que da lugar a la necesidad de un mecanismo que permite facilitar la comprensión y asimilación de la información, llamado visualización de datos.

La visualización de datos logra convertir un conjunto de datos complejo en una información clara y lo hace como su nombre indica, a través de elementos visuales, es decir, gráficas que juegan variedades de colores, figuras y texto. Es importante destacar que la visualización de datos necesita un estudio previo para la preparación y análisis de los datos.

Debido a lo expuesto anteriormente, en este trabajo se propone la implementación de un editor de visualizaciones de propiedades de historiales de wikis.

El presente trabajo se encuentra dividido en 3 capítulos, donde el primero habla sobre la visualización de datos y herramientas que explotan esta área para ayudarnos en el desarrollo. El segundo capítulo abarca las tecnologías web que contemplaremos para el desarrollo de la aplicación, como la arquitectura y diseño. Por último, en el cuarto capítulo se propone todo lo necesario para el desarrollo adecuado de la aplicación web.

Capítulo 2

Tecnologías para la visualización de datos en web

La visualización de datos es un área que tiene tiempo siendo explorada, donde el objetivo es lograr transmitir una información pero visualmente, es decir, mediante gráficas haciendo uso de colores, figuras, orientación, escalas, entre otros elementos. Es importante tomar en cuenta la visualización de datos en casos donde la información original a transmitir puede ser muy extensa y/o compleja, por lo tanto podemos hacerla más legible y comprensiva hacia una audiencia en específico. Posiblemente antes de representar la información visualmente esta tiene que ser transformada a otro tipo de dato dependiendo del caso. Es importante saber que existen dos paradigmas a la hora de realizar visualización de datos, explicativo y exploratorio ¹.

- **Exploratorio:** cuando la audiencia/usuario está familiarizado con los datos y hace uso de la visualización para encontrar o formar algún patrón en particular.
- **Explicativo:** cuando se quiere transmitir una información específica ya explorada hacia una audiencia que no tiene conocimiento alguno de los datos.

Para este trabajo especial de grado tenemos como meta una herramienta que ofrezca la posibilidad de construir gráficas en base a propiedades correspondientes a un historial de artículos de wikis, por lo tanto es indiscutible la necesidad de hacer uso de la visualización de datos. Cabe destacar que el historial de un artículo de wiki puede ser lo suficientemente extenso, por lo tanto representarlo de una forma visual, es decir, mediante algún tipo de gráfica, puede transmitir mejor una información que a simple vista no es obvia.

Construir una gráfica representativa puede llegar a tomarnos mucho tiempo y

¹Exploratorio y explicativo <http://www.storytellingwithdata.com/blog/2014/04/exploratory-vs-explanatory-analysis>

posiblemente ser compleja. La idea es aprovechar la existencia de avanzados proyectos que nos ayudan en la implementación de la visualización para facilitar el trabajo y de la misma forma, tener un mejor resultado. Existen muchas bibliotecas donde la mayoría tiene la misma finalidad, por lo que se realizó una búsqueda y evaluación, para luego quedar con las que mejores se adapten a las necesidades.

Para comenzar, hay que saber que las visualizaciones para este caso se construirán en el ámbito web, haciendo uso de HTML [1] y JavaScript [2]. De las bibliotecas que nos apoyaremos, algunas construyen la gráfica utilizando el componente Canvas [3] de HTML5 y otras utilizando el estándar SVG [4] (Scalable Vector Graphics).

2.1. SVG vs Canvas [5]

El elemento Canvas es literalmente, un lienzo donde se va a pintar la gráfica, el proceso de construcción es más complejo y manual ya que la manera para dibujar consiste en pintar pixel por pixel, por lo que el cambio de resoluciones afecta lo dibujado. El elemento SVG (Scalable Vector Graphics) representa un vector escalable, donde cada elemento de la gráfica, ya sea una caja, círculo, texto, o imagen representa un sub-elemento del SVG general. Adicionalmente SVG permite el manejo de eventos y los cambios de resoluciones no afectan la visualización, esto hace que sea más fácil trabajar y personalizar los elementos de la gráfica a dibujar.

Como usaremos bibliotecas para la creación de las gráficas no debemos preocuparnos por la dificultad que toma la construcción con Canvas comparado con SVG, ya que la biblioteca lo hará por nosotros. El punto importante es que Canvas tiene un rendimiento mayor que el SVG cuando la gráfica maneja muchos objetos, debido a que cada objeto de la gráfica es un elemento que impacta en el DOM [6] del HTML y esto hace que más memoria RAM sea consumida.

2.2. Bibliotecas basadas en Canvas

A continuación se presentará una lista de bibliotecas que construyen las gráficas haciendo uso de Canvas, acompañada de una lista de las gráficas principales que incluyen:

- **Processing.js²**: Es un lenguaje de programación visual. Al ser un lenguaje de programación visual queda claro que su objetivo es general y

²<http://processingjs.org/>

no únicamente gráficas, por lo que nos permite elaborar desde animaciones hasta juegos. Basta con aprender sus definiciones propias para poder usarlo.

- **Chartjs**³: Biblioteca con aporte de 5 tipo de gráficas: gráfica de línea, barra, área polar, circular (dona), dispersión. En particular, las gráficas de esta biblioteca presentan un buen diseño adaptativo a distintos tipos de pantalla que pueden ser personalizado, incluyendo animaciones. Adicionalmente, podemos extender las funcionalidades descargando y configurando *plugins* elaborados por otras personas.
- **Echarts**⁴: Posee una gran variedad de gráficas personalizables y dando la posibilidad de habilitar animaciones. Ofrece soporte para la mayoría de los navegadores web y buena usabilidad para los dispositivos móviles, tanto rendimiento como adaptabilidad a la pantalla. La biblioteca con todas las gráficas y componentes incluidos ocupa alrededor de 500KB, pero es posible solo descargar algunos tipos de gráficas para reducir el tamaño de la biblioteca. Entre las gráficas disponibles se tienen: gráfica de línea, barra, área, mapa, circular (dona), dispersión, velas, grafos, boxplot, paralela, embudo y *themeriver* (variación temática sobre el tiempo).

2.3. Bibliotecas basadas en SVG

A continuación se presentará una lista de bibliotecas que construyen las gráficas haciendo uso de SVG, acompañada de una lista de las gráficas principales que incluyen:

- **Raphael.js**⁵: Es una pequeña biblioteca *cross-browser*, es decir, soportada por la mayoría de los navegadores web, con la capacidad de ofrecernos herramientas para elaborar cualquier visualización con vectores. No está orientada solo a la elaboración de gráficas, por lo que podemos crear cualquier visualización, ya sea juegos, alguna especie de arte o animación.
- **Google Chart**⁶: Biblioteca elaborada por Google, donde ofrecen aproximadamente 29 tipos de gráficas, animadas, con un estilo minimalista y soporte para muchos navegadores, incluyendo versiones viejas. Además, nos permite configurar y personalizar las gráficas a nuestro gusto. La biblioteca solo ocupa 70 KB. Entre los tipos de gráficas disponibles destacan: gráfica de línea, barra, área, mapa, circular (dona), dispersión, intervalos, boxplot, velas, treemap y línea en el tiempo.
- **Plotly.js**⁷: Es una biblioteca que está basada (construida) con ayuda de

³<http://www.chartjs.org/>

⁴<https://ecomfe.github.io/echarts-doc/public/en/index.html>

⁵<http://dmitrybaranovskiy.github.io/raphael/>

⁶<https://developers.google.com/chart/>

⁷<https://github.com/plotly/plotly.js/>

la biblioteca D3 y stackgl⁸. Tiene 20 tipos de gráficas, incluyendo en 3D (tres dimensiones) con un diseño agradable y con una cómoda caja de herramientas flotante para interactuar con la gráfica. Entre los tipos gráficas disponibles se tienen: gráfica de línea, barra, área, circular (dona), mapa, dispersión, boxplot, velas, treemap e histogramas (2D y 3D).

2.4. D3 (Data-Driven Documents)

D3 (Data-Driven Documents)⁹ es considerada una de las bibliotecas más potentes para la manipulación de datos, con varios años en desarrollo y con una comunidad bastante activa. Con esta biblioteca se puede llegar a construir casi cualquier tipo de gráfica deseable en SVG o HTML Canvas, gracias a que tenemos un control total sobre la construcción y diseño de la gráfica. También se dispone de *plugins* y gran cantidad de ejemplos aportados por otras personas que pueden ayudarnos a facilitar la programación de la gráfica. La biblioteca tiene un tamaño base de 230 KB aproximadamente en su versión actual (v4.7.3). Para hacer posible la construcción de una gráfica, D3 incluye los siguientes elementos claves:

- **Selecciones:** modificar elementos de manera imperativa, siendo menos tediosa a la tradicional.

```
d3.selectAll("p").style("color", "white");
```

Podemos modificar atributos o estilos, registrar eventos, agregar, eliminar nodos logrando cambiar HTML o texto contenido.

- **Propiedades dinámicas:** los estilos, atributos y otras propiedades pueden ser especificadas como funciones de datos, es decir no siempre reciben constantes.

```
d3.selectAll("p").style("color", function() {  
    return "hsl(" + Math.random() * 360 + ",100%,50%)";  
});
```

- **Entrar y salir:** facilita agregar y/o remover elementos en un grupo de datos.

```
var p = d3.select("body")  
    .selectAll("p")  
    .data([4, 8, 15, 16, 23, 42])  
    .text(function(d) { return d; });  
  
// Entrar  
p.enter().append("p")  
    .text(function(d) { return d; });  
  
// Salir  
p.exit().remove();
```

⁸Es un ecosistema para WebGL <http://stack.gl/>

⁹<https://d3js.org/>

En casos donde se busca optimizar esta sección es útil debido a que podemos establecer una navegación en la visualización y solo mostrar un grupo de elementos donde el resto se elimina, de tal forma que se vayan agregando y eliminando elementos a medida que se realicen acciones sobre la gráfica.

- **Transiciones:** existen controles para las animaciones, como la duración y tiempo de aplazo.

```
d3.selectAll("circle").transition()
  .duration(750)
  .delay(function(d, i) { return i * 10; })
  .attr("r", function(d) { return Math.sqrt(d * scale); });
```

La biblioteca D3 no introduce una nueva representación visual como hace Raphael.js y Processing.js (bibliotecas mencionadas anteriormente), debido a que se trabaja directamente con estándares web (HTML, CSS [7], SVG), por ejemplo, podemos crear una gráfica en SVG y luego darle estilo con un archivo externo CSS. Sin embargo, estas tres (3) bibliotecas por el hecho de darnos una libertad total al construir visualizaciones requieren un tiempo considerado para aprender y poder usarlas debidamente.

2.5. Evaluación de Bibliotecas

Es importante saber que para nuestro caso todas las bibliotecas mencionadas anteriormente son las que mejor se adaptan según las necesidades, aún así se realizaron ciertas evaluaciones para decidir cuál utilizar. Para la evaluación se consideró lo siguiente:

- **Variedad de gráficas**

Para nuestro trabajo es necesario varios tipos de gráficas, ya que se realizarán visualizaciones con cantidad y tipo de datos distintos, por lo tanto es indispensable que existan diferentes tipos de gráficas para cada situación a visualizar en particular. Se revisó todos los tipos de gráficas que podían ofrecer cada una de las bibliotecas. Si es necesario un diseño o tipo de gráfica bastante particular probablemente para ese caso la mejor opción sería D3, Raphael.js o Processing.js.

- **Rendimiento**

Definimos rendimiento como el comportamiento que toma la biblioteca al construir una gráfica considerablemente pesada (con inmensa cantidad de datos). El problema que podemos presentar al visualizar grandes cantidades de datos es que la vista donde está la gráfica se perciba con cierta lentitud y en el peor de los casos el navegador (browser) se detenga, es decir deje de trabajar. Afortunadamente, hay algunas bibliotecas que optimizan las gráficas y aplican un estilo de paginación, donde no dibujan todos los puntos a la vez, sino a medida que nos profundizamos en la gráfica.

Se sometieron algunas de las bibliotecas a la creación de una gráfica desde 10.000 hasta 1.000.000 de datos. En la **Figura 2.1** podemos apreciar una aproximación del resultado basado en la cantidad de objetos que pudieron soportar, en donde el eje 'y' corresponde al número de objetos, donde M representa el millón. Cabe destacar que D3, Processing y Raphael no fueron considerados ya que su rendimiento es totalmente relativo a como se programe la gráfica.

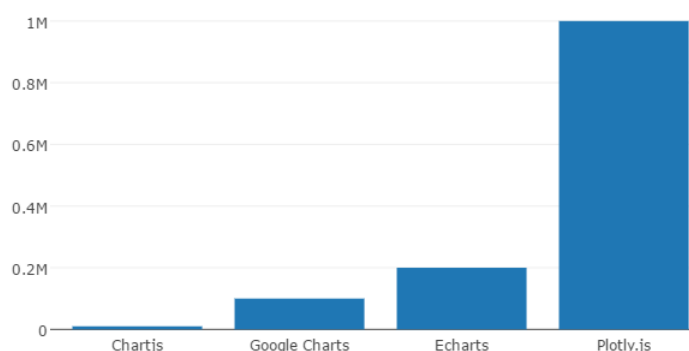


Figura 2.1: Comparación de bibliotecas de visualización

Al finalizar la evaluación, quedamos con tres (3) candidatos: D3, Echarts y Plotly. Donde cada uno predomina en distintos escenarios necesarios para nuestro trabajo. Adicionalmente, **D3** al ser una biblioteca que nos ofrece la mayor libertad para elaborar gráficas a nuestro gusto dará aquel soporte en gráficas sumamente específicas, también viene acompañada de muchos ejemplos elaborados por la comunidad que pueden servir de apoyo. **Plotly** a pesar de construir las visualizaciones en SVG logra un excelente rendimiento, aún mejor que el resto de las bibliotecas, además su diseño es agradable y nos ofrece una caja de herramientas para interactuar con la gráfica. Como último candidato a **Echarts**, por su gran variedad de ejemplos y los posibles tipos de gráficas que nos puede ofrecer, agregando que, dan soporte con gráficas que son usables en dispositivos móviles logrando además un buen rendimiento para aquellos sistemas de bajos recursos.

Capítulo 3

Tecnologías para el desarrollo web

Brindar una buena experiencia de usuario en una aplicación puede llegar a ser un gran desafío donde existen varios elementos que afectan directamente, como el **diseño**, ya que es importante jugar un buen estilo, de carácter agradable, y con animaciones amigables. El **rendimiento**, donde la aplicación debe desempeñar una buena fluidez de respuesta y **correctitud**, que se contemple una interfaz que tenga sentido con la funcionalidad de la misma.

Cuando entramos en el área web, nuestro caso una aplicación web, llegan algunos desafíos adicionales como aplicar soporte de diseño para variedad de densidades de pantallas y versiones de navegadores web, esto se debe a que nuestra aplicación estará expuesta en la web, donde puede ser accedida desde tabletas, móviles, computadores de escritorio, televisores, y cualquier dispositivo que tenga conexión a internet y un navegador. También el peso de la aplicación, incluyendo códigos, fuentes e imágenes impacta considerablemente al tiempo de espera para conexiones lentas a internet.

En este capítulo presentaremos herramientas que nos facilitarán el trabajo para lidiar las problemáticas mencionadas segmentando la aplicación por: arquitectura, diseño y utilidades extras a emplear. Cabe acotar que para el manejo de estas herramientas es necesario tener conocimiento sobre JavaScript, TypeScript¹⁰ (subconjunto de JavaScript con un sistema de tipos más robusto), CSS y HTML.

3.1. Arquitectura

La aplicación adoptará una arquitectura llamada *Single-page Application* [8] (SPA) que significa aplicación de una página, donde se busca englobar toda la aplicación en una vista, logrando cargar la base de la aplicación completa y luego dinámicamente mediante JavaScript hacer la transición de las vistas. La

¹⁰TypeScript <https://www.typescriptlang.org/>

gran ventaja que podemos sacar de SPA es la fluidez de la aplicación entre cambios de vistas, ya que todos estos elementos ya fueron cargados previamente. Por contraparte, la carga inicial suele ser pesada por traerse elementos de más, pero de igual forma esta puede optimizarse para traer solo los recursos necesarios.

Mayormente nuestra aplicación tendrá comunicación casi constante con un API que nos ofrecerá cantidades de datos para nosotros transmitir visualmente, por lo tanto manejaremos peticiones HTTP¹¹ (Hypertext Transfer Protocol) con AJAX [9] (Asynchronous JavaScript And XML), cabe destacar que al ser asíncrono evitamos que afecte el flujo principal de la aplicación.

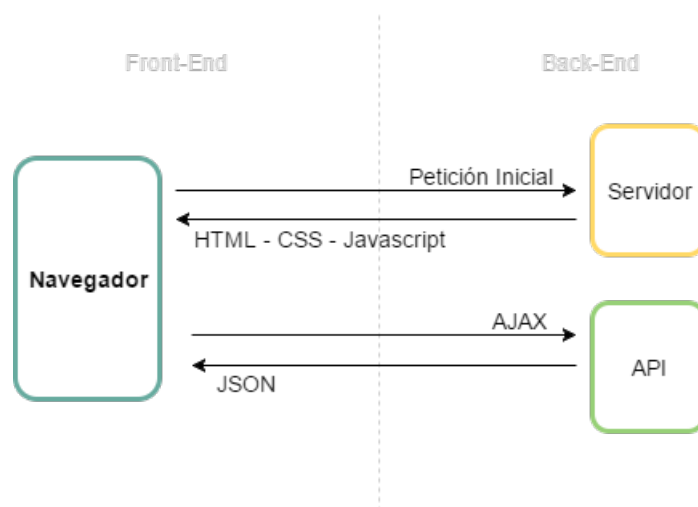


Figura 3.1: Ciclo de vida de una aplicación SPA

El ciclo de una aplicación SPA como se muestra en la **Figura 3.1** requiere por lo general una primera petición para cargar las vistas, funciones y diseño, el resto de las peticiones se dirigen al API a través de AJAX para pedir los datos a visualizar.

A continuación presentaremos dos (2) bibliotecas que nos permiten definir la arquitectura de la aplicación: Angular y jQuery.

3.1.1. Angular

Angular¹² es un framework muy popular creado por Google destinado a construir aplicaciones web, donde se maneja la lógica de la aplicación con JavaScript. Para la versión 2 de Angular, que es la que se contempla para este proyecto, se logra una estructura modular en la aplicación, donde cada elemento es considerado un componente que se construye de manera aislada, por lo tanto no se ve afectado por el resto de los mismos. Claramente, el enfoque de Angular sigue

¹¹HTTP: es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

¹²Angular 2 <https://angular.io/>

la arquitectura SPA, donde logra optimizar y resolver posibles problemas con los estados de transiciones, carga inicial, caching y mejor manejo de peticiones HTTP.

Es importante saber que Angular es una biblioteca con un catálogo inmenso de funcionalidades soportado por la mayoría de los navegadores, donde ofrece desde mecanismos de seguridad para evitar *Cross-site scripting*¹³ (XSS) hasta un paquete completo de animaciones.

A continuación, se muestra un ejemplo de código representando un componente en Angular:

```
// app.component
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: '<h1>Hola Mundo</h1>'
})
export class AppComponent { name = 'Angular'; }
```

Podemos observar que para declarar un componente es necesario importar primero el paquete de Angular y luego llamar al decorador **@Component()**, en donde es pasado un objeto identificado por un **selector**, que corresponde a un alias del componente y el **template** que puede ser una ruta de un archivo HTML o este caso una pieza de código HTML.

```
// app.module
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent }  from './app.component';

@NgModule({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:   [ AppComponent ]
})
export class AppModule { }
```

Es indispensable al menos un módulo en Angular por lo tanto en el código anterior estamos definiendo un módulo con el decorador **@NgModule()**, lo importante de aquí es que en dicho módulo declaramos el componente anteriormente creado.

Otras características que destacan a Angular son:

■ Directivas

Es una manera de lograr que las vistas sean dinámicas, es decir, los componentes HTML pueden tener atributos especiales de Angular para poder modificar elementos del DOM.

```
<li *ngFor="let item of arrayItem"></li>
<app-detail *ngIf="selectedItem"></app-detail>
```

¹³Cross-site scripting https://es.wikipedia.org/wiki/Cross-site_scripting

La directiva ***ngFor** nos permite iterar en un ciclo, originando un `ngFor` por cada elemento de `arrayItem` y ***ngIf** nos permite mostrar o no elementos mediante una condición.

■ Servicios

Pueden representar cualquier función, valor o características que la aplicación necesite. Mayormente los componentes son consumidores de estos servicios, por ejemplo: una configuración de la aplicación, un servicio que hace peticiones HTTP a un API, un servicio encargado de actualizar una barra de navegación, un servicio para mostrar mensajes estilo Logs, etc.

```
// logger.service.ts
export class Logger {
  log(msg: any) { console.log(msg); }
  error(msg: any) { console.error(msg); }
  warn(msg: any) { console.warn(msg); }
}
```

El código de arriba representa un ejemplo de un servicio de Logs, en donde nos ofrece una clase con tres (3) variedades de avisos a consola.

3.1.2. jQuery

jQuery¹⁴ es una biblioteca ligera y sencilla, que lleva bastante tiempo siendo usada. Su finalidad es ofrecer soluciones a funciones complejas de manera sencilla principalmente para la manipulación de elementos HTML y eventos, animaciones y AJAX. Considero jQuery como una opción manual al construir una aplicación SPA, ya que directamente no ofrece una solución completa, pero existen plugins y/o guías para lograr dicha arquitectura.

■ Manipulación de elementos

```
$( "#button.green" ).html( "Next" );
```

■ Manejo de Eventos

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event )
{
  hiddenBox.show();
});
```

■ Petición HTTP Asíncrona con AJAX

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $( "#weather-temp" ).html( "<strong>" + result + "</strong>"
    + " degrees" );
  }
});
```

¹⁴jQuery <https://jquery.com/>

3.2. Diseño

Es importante que nuestra aplicación presente un buen diseño, donde se simplifique la finalidad de la aplicación, además de ser capaz de adaptarse a distintos tamaños de pantallas. No hay un guion o estilo definido para el diseño, la idea es aplicar colores que combinen, componentes sencillos como: botones, selectores, barra de menú, diálogos, iconos, *inputs* (entradas de información), texto y componentes no tan sencillos como las gráficas para visualizar los datos. Todos estos componentes harán posible una interfaz que sea capaz de representar un editor de visualizaciones, en donde recibiremos apoyo de bibliotecas que nos ofrecen estos componentes bien formados y listos para usar.

3.2.1. Angular Material

Angular Material¹⁵ es una biblioteca que ofrece componentes siguiendo un diseño llamado Material (Material Design)¹⁶ creado por Google. Para poder integrar esta biblioteca es requerido Angular 2, por lo tanto, esta opción se ve atada a usar dicho framework. Lo que destaca de Angular Material es la variedad de componentes, que además son totalmente adaptativos gracias a una biblioteca incluida llamada *Flex Layout*¹⁷ que nos ofrece propiedades sencillas de usar para corresponder el tamaño de los componentes en distintos escenarios. Adicionalmente, es soportado por la mayoría de los navegadores modernos.

Angular Material al estar atado al framework nos da la posibilidad de exportar sólo los componentes que necesitemos y no todos los de la biblioteca, además nos permite añadir soporte de gestos a los componentes como *toggle* o *slider*.

Entre tantos componentes, se mostrará ejemplos de algunos a continuación:

■ Botones

```
<button md-raised-button>Raised button</button>
<button md-fab><md-icon>check</md-icon></button>
```

En donde el atributo **md-raised-button** representa un botón con elevación, **md-fab** es un botón mayormente flotante con forma circular. Podemos observar que existe otro elemento **md-icon** que nos permite colocar un ícono a través de una fuente especial llamada *Material Design Icons*¹⁸, obteniéndola con el siguiente código desde nuestro archivo HTML.

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

■ Spinner de Progreso

```
<md-progress-spinner
  class="example-margin"
  [attr.color]="color"
  [mode]="mode"
```

¹⁵Angular Material <https://material.angular.io/>

¹⁶Material Design <https://material.io/>

¹⁷Flex Layout <https://github.com/angular/flex-layout>

¹⁸Material Design Icons <https://material.io/icons/>

```
[value]="value">
</md-progress-spinner>
```

El *spinner*, que hace referencia a un elemento circular con acción de progreso, se puede lograr usando **md-progress-spinner** en donde puede tener como atributo un color, un modo para representar si girará finitamente o infinitamente y un valor que representa el porcentaje de progreso actual.

■ Barra de Herramientas

```
<md-toolbar>My App</md-toolbar>
```

Con **md-toolbar** logramos crear una barra en donde podremos colocar texto e iconos con acciones, por ejemplo, la barra principal superior de la aplicación.

3.2.2. Bootstrap

Bootstrap¹⁹ ofrece un gran catálogo de componentes, desde distintos tipos de botones, formularios, barras de menú, hasta diálogos. Recientemente lanzaron la versión 4, en donde se reescribió la mayoría del proyecto corrigiendo bastantes errores, sin embargo esta versión se encuentra en fase alfa, esto quiere decir que no es estable. Bootstrap es una biblioteca elaborada por Twitter, donde es posible la construcción rápida de una interfaz tanto para escritorio como para móvil, ofreciendo gran soporte para la mayoría de los navegadores y también posibilidad de adaptarse a distintas dimensiones de pantallas. Para que Bootstrap funcione es necesario la biblioteca jQuery que anteriormente mencionamos.

Bootstrap tienen su propio sistema *grid* para escalar los componentes según cambie el tamaño de la pantalla, en donde se distinguen tres medidas: **lg** (largo), **md** (mediano), **sm** (pequeño) y **xs** (extra pequeño). Estas medidas pueden aplicarse a columnas o filas, representadas como **col** y **row**. Por cada fila puede haber doce (12) columnas, si este es superado el elemento faltante irá posicionado abajo.

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

Entre los componentes considerados a usar:

■ Botones

```
<button type="button" class="btn btn-default">Default</button>
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-danger">Danger</button>
```

¹⁹Bootstrap <http://getbootstrap.com/>

Bootstrap nos proporciona un abanico de botones con estilos definidos, en donde cada uno juega un color de fondo distinto.

■ Tipografía

```
<span class="glyphicon glyphicon-star" aria-hidden="true"></span> Star
```

Es posible incluir íconos gracias a que Bootstrap trae su propia tipografía.

■ Barra de Progreso

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="
    60" aria-valuemin="0" aria-valuemax="100" style="width:
    60%;">
    <span class="sr-only">60% Complete</span>
  </div>
</div>
```

■ Barra de Herramientas

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        
        <h1>Title</h1>
      </a>
    </div>
  </div>
</nav>
```

Es posible posicionar una barra superior de manera fija, con cualquier elemento adentro de ella, pero para ellos debes cumplir con los 3 primeros elementos padres: el primer **nav** y los dos siguientes **div**.

3.3. Utilidades

Existen distintas bibliotecas de propósito más específico cuyo uso se ha vuelto cotidiano, donde pueden solucionar tareas complejas o engorrosas de implementar en JavaScript, específicamente para nuestro proyecto existe la necesidad de preparar ciertos conjuntos de datos complejos para poder construir una gráfica, al igual que manejar varios formatos de fechas.

3.3.1. Lodash²⁰

La manipulación de ciertos objetos, arreglos y strings complejos pueden contemplar soluciones un tanto ilegible, por lo tanto la biblioteca Lodash nos otorga utilidades que resuelven dichos problemas de una manera simple. Adicionalmente, cuenta con funciones matemáticas y soluciones para manejar valores null,

²⁰Lodash <https://lodash.com/>

NaN y undefined. Todas sus funcionalidades son soportadas por la mayoría de los exploradores.

A continuación, se mostrarán unos ejemplos de las ventajas que puede traer lodash a nivel de implementación:

- **Verificar variable (no sea null ni undefined)**

```
// no-lodash
if(a !== null && a !== undefined) {...}

// lodash
if(!_.isNull(a)) {...}
```

- **Ciclos**

```
// no-lodash
for(var i = 0; i < 5; i++) {
  ...
}

// lodash
_.times(5, function(){
  ...
});
```

- **Números aleatorios**

```
// no-lodash
Math.floor(Math.random() * (max - min + 1)) + min;

// lodash
_.random(min, max);
});
```

- **Seleccionar elementos de un arreglo**

```
// no-lodash
function pick(arr) {
  var _this = this;
  var obj = {};
  arr.forEach(function(key){
    obj[key] = _this[key];
  });

  return obj;
};
objA.pick(['x', 'y'])

// lodash
_.pick(objA, ['x', 'y']);
});
```

3.3.2. Moment²¹

La manipulación de fecha y tiempo en JavaScript suele ser confusa y consume muchas líneas de código, es probable que en nuestra aplicación tengamos que

²¹Momentjs <https://momentjs.com/>

recibir fechas con un formato específico y de la misma forma representar este visualmente de otra. Moment es una biblioteca que tiene como finalidad analizar sintácticamente, validar, manipular y mostrar fechas/horas (tiempo) de la forma más simple posible. Adicionalmente, esta biblioteca ofrece extensiones para poder realizar ciertas funcionalidades más específicas.

A continuación mostraremos ejemplos de cosas interesantes que podemos lograr con dicha biblioteca:

- **Obtener la fecha/hora de hoy con formato**

```
// date
moment().format("DD/MM/YYYY");
// time
moment().format("HH:mm:ss");
```

- **Agregar/Quitar tiempo**

```
// add
moment().add(7, 'days');
moment().add(1, 'week')
// subtract
moment().subtract(7, 'days');
moment().subtract(1, 'week')
```

- **Obtener unidad de tiempo específica**

```
// hours
moment().hours();
// month
moment().month();
```

Capítulo 4

Propuesta de Trabajo Especial de Grado

4.1. Motivación e identificación del problema

Un artículo wiki puede llegar a tener miles de ediciones a través del tiempo, en donde cada edición tiene la fecha y hora, el nombre de usuario o dirección de IP, el tamaño de la página en bytes, la diferencia de tamaño de página entre la edición y el estado anterior, una pequeño texto resumiendo la edición, entre otros atributos. Sin embargo, a simple vista todos estos datos pueden ser difíciles de leer y comprender por un humano, planteado esto surge la iniciativa de proyectar los datos a través de gráficas, lo que sería visualización de datos, en donde es posible ofrecer una información más clara y simple, además de poder explorar y hacer comparaciones entre los datos y el tiempo de manera visual para poder obtener una información más interesante al respecto.

4.2. Objetivos del trabajo

4.2.1. Objetivo General

Desarrollar una aplicación web que permita construir y editar visualizaciones de propiedades de historiales de wikis.

4.2.2. Objetivos Específicos

- Diseñar visualizaciones generales basadas en la información de los historiales de artículos de wikis provista por el API de Wikimetrics 2.0.
- Definir los requerimientos de la aplicación.

- Implementar una interfaz SPA adaptativa que ofrezca las funcionalidades requeridas por un *watcher* de un wiki.
- Utilizar un método ágil para el desarrollo de la aplicación.
- Realizar el despliegue y puesta en producción.

4.3. Estrategia de solución y método de desarrollo ágil a utilizar

Los objetivos específicos serán desglosados de manera técnica en pequeñas tareas ordenadas por prioridad. El control de estas asignaciones se manejará mediante la plataforma GitHub (aplicación web para alojar repositorios Git) en donde cada tarea será un *issue* a resolver. Github nos permite proyectar estos *issues* en una pizarra, con el fin de visualizar el estado de cada asignación, los estados definidos son: Por hacer, En progreso, Terminado. Puede verse un ejemplo en la **Figura 4.1**

- **Por hacer:** representa aquellas tareas especificadas, que por el momentos son *issues* sin resolver.
- **En progreso:** representa las tareas que están siendo desarrolladas, cabe destacar que cada asignación se resuelve en una rama distinta del repositorio git haciendo uso del *pull request* para llevar un mayor control del desarrollo de la asignación.
- **Terminado:** representa aquellas tareas culminadas. Cuando se considera que una tarea está lista, esta tiene que ser mezclada a la rama principal del repositorio, llamada *master*, luego cerrar el *pull request* y el *issue* asociado.

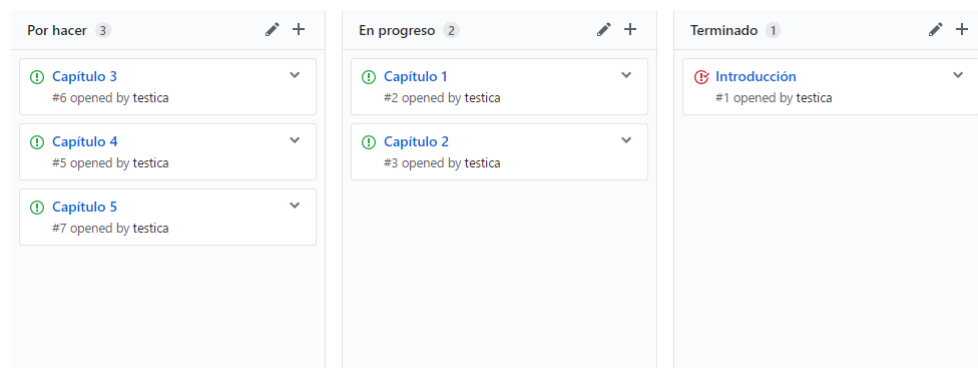


Figura 4.1: Ejemplo de pizarra de Github

La pizarra de Github manifiesta una metodología Kanban [10], en donde visualizar el flujo de trabajo y hacerlo visible es la base para comprender cómo avanza el trabajo. Sin comprender el flujo de trabajo, realizar los cambios adecuados es más difícil. Una forma común de visualizar el flujo de trabajo es el

uso de columnas. Las columnas representan los diferentes estados o pasos en el flujo de trabajo. Kanban logra un desarrollo evolutivo e incremental, donde las soluciones de las asignaciones puede que no sean las mejores comenzando, pero a medida que se itera se va perfeccionando.

4.4. Trabajos similares, diferencias y ventajas de la solución a desarrollar

Wikimedia Tools Labs²² es una página que ofrece un ambiente apropiado para los desarrolladores de la comunidad, en donde podemos encontrar una lista de herramientas elaboradas por personas que trabajan sobre wikis ayudando a facilitar determinadas tareas. Las siguientes herramientas son similares:

- **WikiHistory**²³: Esta herramienta web, nos ofrece para cada wiki, una información general, en donde podemos saber cuántos usuarios editan por día, al mes o al año, la primera y última edición, entre otras cosa como se puede observar en la **Figura 4.2**. En la misma sección de información general es proyectada una gráfica de porcentajes sobre el tipo de edición, véase la **Figura 4.3**. También nos ofrece una gráfica de Línea de Tiempo, vea **Figura 4.4**, en donde podemos saber la cantidad de ediciones filtradas por año, mes o semana. Por último brinda una tabla de los usuarios que han editados, tomando en cuenta la cantidad de ediciones y la fecha de la primera y última edición.

GENERAL INFORMATION	
Edits total	6396
Minor edits	1780 (27.8%)
Anonymous edits	2068 (32.3%)
Different users	3132
Anonymous users	1439 (45.9%)
Edits per user	2.04
First edit	November 13 2001 19:08 (15 years and 208 days ago) User: Paul Dye
Last edit	June 02 2017 20:37 (3 days ago) User: GreenC bot
Edits per day	1.13
Edits per month	33.76
Edits per year	411.08
Current size	82,109 Byte

Figura 4.2: Información general proporcionada por la herramienta WikiHistory sobre el artículo 'Chocolate'.

²²Wikimedia Tool Labs <https://tools.wmflabs.org/>

²³WikiHistory <https://tools.wmflabs.org/xtools/wikihistory/>

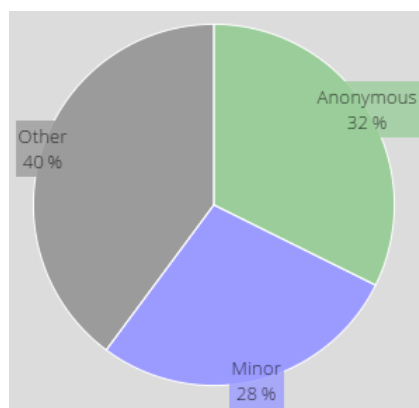


Figura 4.3: Gráfica de tipo torta de WikiHistory, que refleja el porcentaje de ediciones anónimas, menores y otras, sobre el artículo 'Chocolate'.

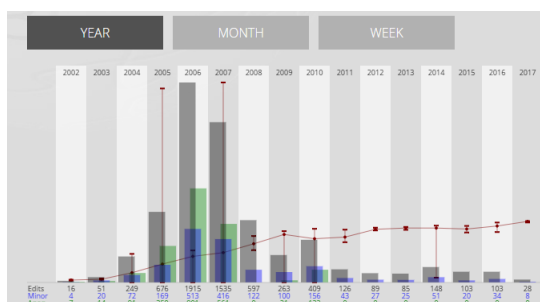


Figura 4.4: Línea de tiempo, que proyecta visualmente las ediciones totales, menores y anónimas del artículo 'Chocolate', que pueden ser filtradas por año, mes o semana.

Esta página ofrece datos importantes pero de manera estática, es decir no podemos realizar otras consultas adicionales y solo ofrece dos (2) gráficas, nuestro trabajo explotará más el área de visualización de datos, dejando cierta libertad al usuario para que explore. Por otro lado, cuando el wiki tiene muchas ediciones de usuarios distintos la página tarda en cargar y se vuelve lenta debido a que el DOM tiene muchos elementos, la idea que nos llevamos a nuestro trabajo es mostrar la información por segmentos, ofreciendo paginación o filtros.

- **PageViews**²⁴: Es una herramienta para visualizar las vistas que ha recibido una wiki a través del tiempo, vea **Figura 4.5**, lo interesante de esta herramienta es que podemos editar el intervalo de tiempo, además destaca la posibilidad de sobreponer las líneas de tiempo para poder comparar las visitas de varias wikis con el propósito de encontrar datos interesantes. Cabe acotar que nuestro trabajo no se especializará en visualizar las vistas de un wiki, pero esta herramienta logra exprimir el poder de la visualización de datos y eso es lo que queremos lograr pero con las propiedades de las ediciones de un wiki.

²⁴PageViews <https://tools.wmflabs.org/pageviews/>

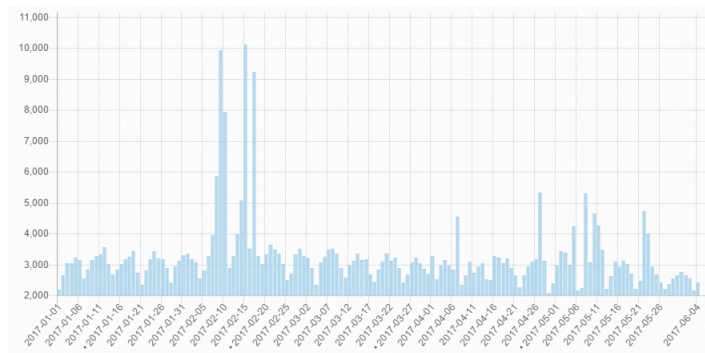


Figura 4.5: Visualización tipo barra sobre el artículo 'Chocolate', en donde el eje 'y' representa el número de visitas, y el eje 'x' representa la fecha.

- **Chromograms** [11]: Es una técnica de visualización con el fin de encontrar patrones en las numerosas ediciones de un artículo de Wikipedia a través de colores. Chromogram corresponde un texto con un color, en donde las tres primeras letras de una palabra definen el color: la primera letra representa el matiz, la segunda la saturación y la tercera el brillo. Cada edición representa un rectángulo con un color posicionado de izquierda a derecha y de arriba hacia abajo. Podemos observar un ejemplo en la **Figura 4.6**

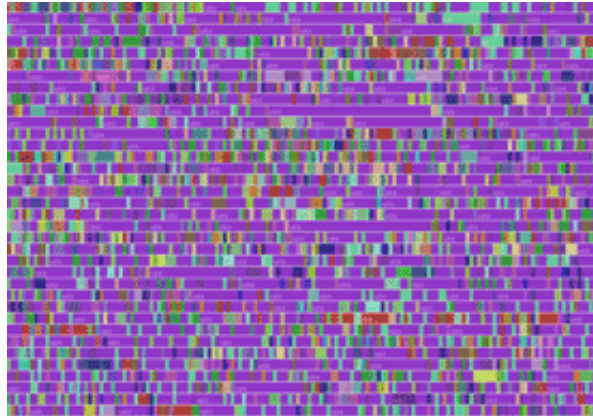


Figura 4.6: Uso de Chromogram donde el color morado representa al artículo de US Ships (en donde el título del artículo empieza con 'USS')

- **History Flow** [12]: Es una técnica de visualización que refleja el comportamiento de las ediciones en el tiempo según el usuario que la realizó. Como indica la **Figura 4.7**, cada edición es representada mediante una barra vertical en donde cada porción de color en la barra representa el usuario, cabe destacar que cada usuario tiene un color específico. Esta visualización nos reporta el crecimiento en el tiempo del artículo wiki, y gracias a los colores podemos observar patrones e identificar comportamientos sospechosos como lo muestra la **Figura 4.8**.

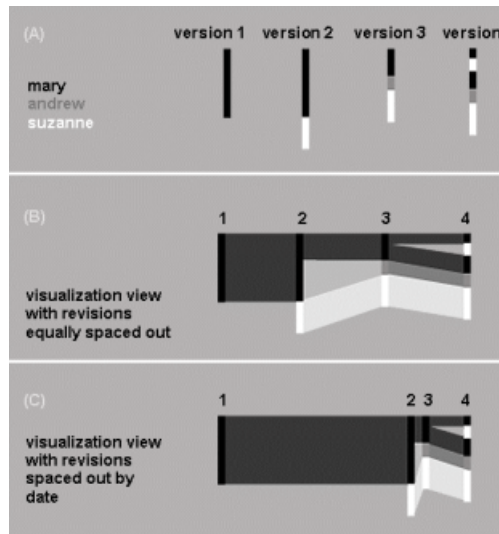


Figura 4.7: Explicación del mecanismo de visualización de History Flow

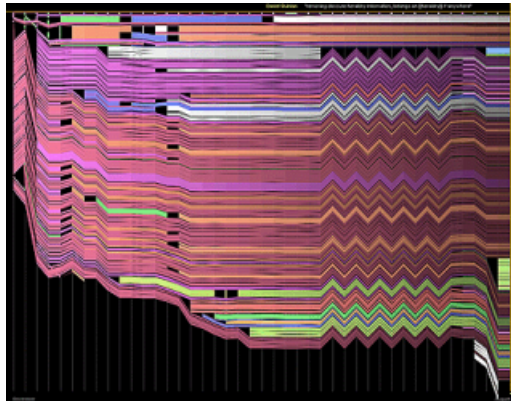


Figura 4.8: En la página 'Chocolate' se observa un patrón en zigzag que demuestra una guerra de ediciones

- History Graph** [13]: Es una adaptación de la técnica de visualización *History Flow*, donde cada barra de la visualización corresponde a una versión, el ancho de cada barra corresponde a la distancia con respecto al cambio anterior, el alto de la barra representa el tamaño en bytes de la versión y el color representa el usuario que hizo el cambio como se aprecia en la **Figura 4.9**. Esta visualización es muy útil para proyectar la evolución en el tiempo de un artículo y observar la frecuencia de cambios, los usuarios que participan y el crecimiento/decrecimiento del texto editado.



Figura 4.9: Demostración de History Graph

4.5. Herramientas

Tecnologías necesarias para el ambiente de desarrollo

- Editor de texto inteligente (Visual Studio Code, Atom, Sublime Text, Notepad++, etc.)
- Git, para manejar el versionamiento del código.
- Navegador de Internet (Google Chrome o Firefox).
- Alguna base de datos (preferiblemente MongoDB) para almacenar configuración de usuarios.
- Servidor Web para levantar la aplicación (Node, Apache, etc.)

Tecnologías web a utilizar

- Bibliotecas de visualización: para las visualizaciones genéricas vamos a usar las bibliotecas **Echarts** y/o **Plotly**, para visualizaciones no comunes y que tengan una finalidad específica que puedan surgir usaremos **D3**.
- Arquitectura: construiremos la aplicación con ayuda del framework **Angular 2**, que tiene un mayor enfoque SPA y logra una buena estructura en el código.
- Diseño: con ayuda de la biblioteca **Angular Material** desplegaremos el diseño.
- Biblioteca **Lodash** y **Moment** para facilitar ciertas tareas como el formato de fechas y manipulación de objetos y arreglos.

4.6. Requerimientos

- Una herramienta capaz de visualizar gráficas sobre propiedades de edición de un wiki.
- Los wikis a visualizar son extraídas del watchlist del usuario autenticado a través de Wikipedia (usando API de MediaWiki).

- Permitir editar las visualizaciones (alternar tipo de gráfica, cambiar los tipos de datos a visualizar).
- Los datos a representar en las visualizaciones son proporcionados a través del API Wikimetrics 2.0.
- La herramienta tiene que ser una aplicación web.
- La aplicación tiene que contar con una vista principal que incluya gráfica e información general de cada uno de los wiki del watchlist.
- Permitir al usuario crear una visualización nueva y guardarla de forma permanente, para este caso es necesario el uso de una base de datos (preferiblemente MongoDB)
- La aplicación tiene que estar alojada en un servidor, para poder ser accedida de manera remota.

4.7. Prototipo de interfaz

Como vista inicial, si no estamos autenticados, es el inicio de sesión, en donde hay que suministrar el usuario y contraseña de Wikipedia, vea **Figura 4.10**.

The image shows a web browser window with the title "Navegador". Inside the window, on the left, is the Wikipedia logo (a globe with letters) and the text "WIKIPEDIA The Free Encyclopedia" and "Inicia sesión con tu usuario de Wikipedia". On the right, there is a login form titled "Iniciar sesión". The form contains two input fields: "Usuario:" with the text "xyz" and "Contraseña:" with asterisks. Below the fields is a blue button labeled "Iniciar". At the bottom of the form is a link that says "¿No tienes cuenta?".

Figura 4.10: Interfaz para iniciar sesión

La vista principal, consta de un listado de los wiki del watchlist del usuario que inició sesión, con una visualización y un resumen general, vea **Figura 4.11**.

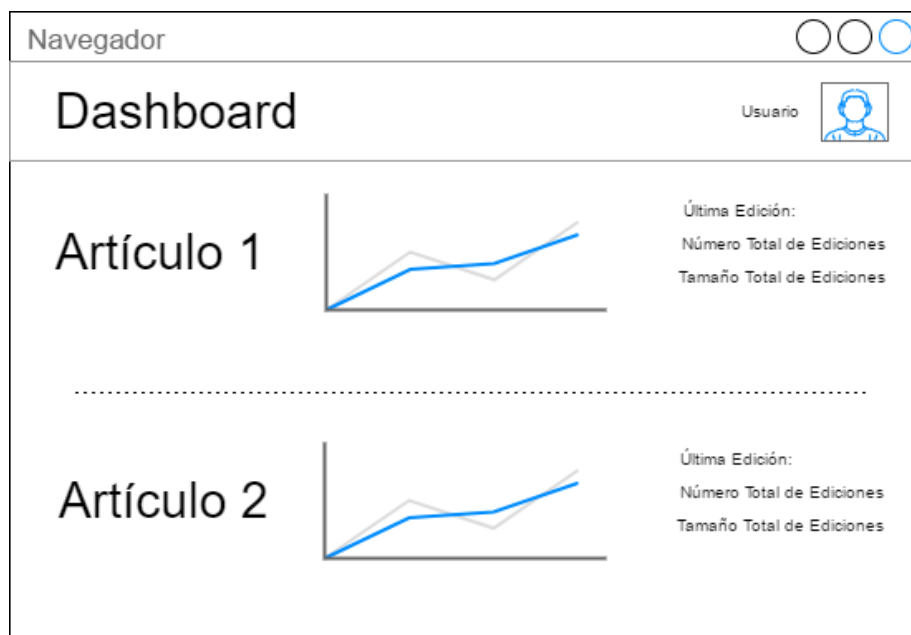


Figura 4.11: Interfaz de la vista principal

Al seleccionar uno de los wiki de la lista, accederemos a una vista detallada del mismo, en donde se ofrecerá información más específica y una lista de visualizaciones generales, adicionalmente un botón para poder crear una, vea **Figura 4.12**.



Figura 4.12: Interfaz de la vista detallada de un wiki

En la vista de creación de una visualización, es necesario suministrar un nombre y los ejes a comparar, de igual forma seleccionar el tipo de gráfica, vea **Figura 4.13**.

Navegador

Artículo1

Nombre visualización

Ediciones ▼

Tiempo ▼

Guardar

Tipo de gráfica ▼

The chart displays a bubble plot with approximately 7 data points. There are 4 blue bubbles and 3 grey bubbles. The bubbles vary in size, with the largest blue bubble located in the lower right quadrant of the chart area.

Figura 4.13: Interfaz de la vista de creación de una visualización

4.8. Planificación de las actividades

Actividad	Tiempo estimado
Preparar el entorno de desarrollo	1/2 semana
Estudiar API Wikimetrics 2.0 (Back-end)	1/2 semana
Estudiar API MediaWiki	1/2 semana
Integración del API Wikimetrics 2.0 (Back-end)	1/2 semana
Integración del API MediaWiki	1/2 semana
Elaborar estructura base de la aplicación web	1/4 semana
Definir que datos se van a almacenar del usuario e integrar base de datos	1 semana
Implementar vista de iniciar sesión haciendo uso del API MediaWiki	1/2 semana
Definir y diseñar gráficas generales en base a los datos que proporciona el API de Wikimetrics 2.0	2 semanas
Estudiar documentación de bibliotecas de visualización	1 semana
Implementar vista de Dashboard	1 semana
Implementar vista de Detalle de wiki, incluyendo gráficas generales	2 semana
Diseñar e implementar creador de gráfica en base a los datos que proporciona el API de Wikimetrics 2.0	1 semana
Añadir estilo apropiado a la aplicación	1/2 semana
Soportar adaptación a distintas pantallas	1/2 semana
Añadir animaciones a la aplicación	1/2 semana
Elaborar despliegue de la aplicación	1/2 semana

Bibliografía

- [1] Mozilla Developer Network. *Introduction to HTML*. URL: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction> (vid. pág. 5).
- [2] Mozilla Developer Network. *JavaScript*. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (vid. pág. 5).
- [3] Mozilla Developer Network. *Canvas API*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API (vid. pág. 5).
- [4] SVG Working Group. *Scalable Vector Graphics (SVG)*. World Wide Web Consortium. 2010. URL: <http://www.w3.org/Graphics/SVG/> (vid. pág. 5).
- [5] Mihai Sucan. *SVG or Canvas, choosing between the two*. Opera Software. 2010. URL: <https://dev.opera.com/articles/svg-or-canvas-choose/> (vid. pág. 5).
- [6] Texcel Research Jonathan Robie. *What is the Document Object Model*. W3. URL: <https://www.w3.org/TR/WD-DOM/introduction.html> (vid. pág. 5).
- [7] Mozilla Developer Network. *CSS*. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (vid. pág. 8).
- [8] the free encyclopedia Wikipedia. *Single-page application*. URL: https://en.wikipedia.org/wiki/Single-page_application (vid. pág. 10).
- [9] Jesse James Garrett. “Ajax: A New Approach to Web Applications”. En: (2005) (vid. pág. 11).
- [10] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Con pról.de Donald G Reinertsen. 2010 (vid. pág. 20).
- [11] Martin Wattenberg, Fernanda B Viégas y Katherine Hollenbach. “Visualizing activity on wikipedia with chromograms”. En: (2007) (vid. pág. 23).
- [12] Fernanda B Viégas, Martin Wattenberg y Kushal Dave. “Studying cooperation and conflict between authors with history flow visualizations”. En: (2004) (vid. pág. 23).
- [13] Eugenio Scalise, Nancy Zambrano y Jean-Marie Favre. “Visualización de Propiedades del Historial de los Artículos de un Manejador de Contenidos

Basado en Wiki Aplicando Ingeniería Dirigida por Modelos”. En: (2008)
(vid. pág. 24).