TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN INFORMÁTICA Y ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS

Visualización de datos de los Objetivos de Desarrollo Sostenible

Autor

Santiago Carbó García

Director

Carlos Ureña Almagro

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Noviembre de 2022

Visualización de datos de los Objetivos de Desarrollo Sostenible

Realizado por Santiago Carbó García

Palabras clave: ODS, API Canvas, JavaScript, WebGL, HTML5, CSS

Resumen

La representación de los datos es una rama de tamaño considerable dentro del desarrollo de sistema gráficos. Asimismo, el Desarrollo Sostenible es un tópico de suma relevancia en la época contemporánea y, por ende, la representación de sus datos. Es por ello por lo que este documento tratará de analizar y explicar el funcionamiento de una aplicación que tiene el objetivo de mostrar dichos datos utilizando especificación de suma utilidad como es WebGL y, dentro de ella, la herramienta API Canvas. La aplicación consiste en un programa JavaScript que utiliza dicha herramienta para la representación de gráficos de dos dimensiones. Para ello, los datos se recogen de un fichero obtenido de la base de datos del Instituto Nacional de Estadística, con un formato en particular. Después son procesados y almacenados para su posterior visualización. Asimismo, cada gráfica dispone de unos parámetros de estilo que pueden ser configurados *a posteriori*. Se han desarrollado tres tipos de gráficos para la visualización de dichos datos: *gráficos de barras*, *gráficos de líneas* y *gráficos circulares*, los cuales permiten tres formas muy intuitivas de reproducir la información. Para la elaboración de los ejes de coordenadas en los dos primeros tipos de gráficos se tendrán en cuenta algunas consideraciones. Primero, el cálculo del número de líneas horizontales necesarias para representar los valores referencia del eje vertical y, segundo, el número de valores mostrados en la gráfica teniendo en cuenta que puede no haber espacio para visualizarlos todos. Es por ello por lo que se usan algunos parámetros como el valor máximo de la serie o el número total de valores.

Data visualization of the Sustainable Development Goals

Written by Santiago Carbó García

Keywords: ODS, API Canvas, JavaScript, WebGL, HTML5, CSS

Abstract

Data representation is a branch of considerable size within the development of graphic systems. Likewise, Sustainable Development is a topic of great relevance in contemporary times and, threfore, the representation of its data. That is why this document will try to analyze and explain the functioning of an application that has the goal of displaying such data using a very useful specification such as WebGL and, within it, the Canvas API tool. The application consists of a JavaScript program that uses this tool in order to represent two-dimensional charts. Fort he purpose of doing this, the data are collected from a file, obtained from the database of the Spanish National Institute of Statistcs, with a particular format. Then the data are processed and stored for later viewing. Likewise, each chart has style parameters that can be configured. Three types of charts have been developed for the visualization of such data: bar charts, line charts and pie charts, which represent three intuitive ways of reproducing information. Fort he elaboration of the coordinate axes in the first two types of charts, some considerations will be taken into account. First, the calculation of the number of horizontal lines necessary to represent the reference values of the vertical axis and, secondly, the number of values shown in the chart, taking into account that there may not be space to display them all. That is why some parameters are used, such as the máximum value of the serie or the total number of values.

Yo, **Santiago Carbó García**, alumno de la titulación *Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas* de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77561011C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Santiago Carbó García

Granada, a 1 de noviembre de 2022

D. Carlos Ureña Almagro, Profesor del Área de [] de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Visualización de datos de los Objetivos de Desarrollo Sostenible***, ha sido realizado bajo su supervisión por **Santiago Carbó García**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 1 de noviembre de 2022.

El director: Dr. Carlos Ureña Almagro

Carlos Ureña Almagro

Agradecimientos

A mi madre y a mi hermana por apoyarme desde que nací, a mis abuelos maternos por cuidarme desde que era pequeño, a mi abuelo Pepe por venir siempre a visitarme desde tan lejos y a María por cuidar de mi todos los días.

Índice general

Índice de figuras

1. Introducción

1.1. Motivación

El desarrollo sostenible es de gran relevancia hoy en día, afectando, entre otras cosas, al derecho y a la informática. Es importante tener en cuenta el consumo energético a la hora de la construcción de sistemas y redes, un consumo que deberemos ir reduciendo con el transcurso de los años para adaptarnos, no solo a la legislación, sino a las necesidades energéticas.

Estas gráficas también deben estar respaldadas por fuentes fiables de información, ya que gran parte de esta no proviene de fuentes que garanticen que la información es veraz. Es por ello por lo que se deben utilizar siempre datos de fuentes fiables (del Instituto Nacional de Estadística o Naciones Unidas, por ejemplo). Su influencia y su impacto son lo suficientemente importantes como para preocuparse por la fiabilidad de los datos. Es por todo ello que un desglose de estos datos y su respectiva representación por el software pertinente es de suma importancia. La estructura de los gráficos, los colores utilizados para ello, el espaciado, etc., serán buenos indicadores de una correcta representación.

Los datos por sí solos carecen de significado. Estos deben ser visualizados e interpretados correctamente mediante el software necesario. Este software no solo debe interpretar y visualizar datos, sino que debe hacerlo correctamente, es decir, debe moldearse a las necesidades cambiantes del usuario, de ahí la importancia de elaborar interfaces correctas en términos de usabilidad y accesibilidad. Por ende, una parte esencial de la visualización de los datos es la comunicación, ya que una imagen tiene el potencial de comunicar de una forma más efectiva que la palabra. Actualmente, debido a la rapidez con la que se desarrolla información y la capacidad de acceso por parte del usuario a este, el consumo de información se ha disparado de manera exponencial. Debido a ello, estamos expuestos diariamente a grandes cantidades de datos, los cuales tienen formatos y estilos diferentes para su representación, lo cual hace aún más difícil su identificación.

Finalmente, aclarar la importancia de las hojas de cálculo para la obtención de información de una forma clara y seguro. Con el objetivo de obtener datos para la representación de gráficos estas son de gran ayuda para almacenarlos. Muchos bancos de información utilizan hojas de cálculos para exportar información con respecto a gráficos, hecho por el cual me he guiado para el desarrollo de la aplicación. Permiten exportar dichas hojas de cálculos de distintas maneras para su procesamiento (celdas separadas por punto y coma, por tabuladores, etc.). Para interpretar esta información, los gráficos de barras, líneas y circulares conforman una de las forman más simples e intuitivas para adquirir esta información.

1.2. Objetivos

Al inicio de todo proyecto es fundamental establecer una serie de objetivos generales y específicos. Es importante poner el enfoque en los aspectos más importantes y cuantificar la relevancia de cada idea. Unos objetivos fiables nos permitirán, en el futuro, el cumplimiento de estos y estimar si son necesarios algunos cambios. Un error muy común en todos los proyectos es, inicialmente, intentar poner en práctica un montón de ideas que no tienen por qué estar sincronizadas y coordinadas. Es posible que esta práctica en el futuro genere problemas a la hora de integrar todas estas ideas, de modo que los objetivos nos recuerdan qué es lo que debemos hacer y qué es lo que no debemos hacer.

1.2.1. Objetivos generales

A modo de síntesis, una definición correcta de los objetivos de este proyecto sería "desarrollar una aplicación interactiva que permite visualizar a los usuarios gráficos que representan información relacionada con los Objetivos del Desarrollo Sostenible y sus diferentes indicadores". No obstante, a modo de desglose, sería posible identificar algunos objetivos generales:

- *Desarrollo de una aplicación web intuitiva e interactiva*. La aplicación realizada contiene una interfaz de usuario intuitiva que permitirá que el usuario pueda utilizarla con facilidad.
- *Puesta en práctica de una especificación de renderización de gráficos en el explorador web*. Uso de esta especificación para representación de datos mediante gráficas en el explorador.
- *Visualización de gráficos en dos dimensiones*. Uno de los fines fundamentales del proyecto consiste en el visualizado de gráficos de dos dimensiones. Estos podrán ser gráficos de barras, de líneas o circulares
- *Procesamiento de datos almacenados en una hoja de cálculo*. Los datos utilizados para la representación de gráficos serán previamente obtenidos y procesados de una hoja de cálculo.
- *Visualización de información auxiliar*. Información acerca del uso de la aplicación o la procedencia de los ficheros utilizados. También será posible acceder a información relacionada con el proyecto.

1.2.2. Objetivos específicos

Una vez explicitados los objetivos generales es posible obtener una visión general del alcance del proyecto. Esto nos permitirá tener una visión clara de la magnitud de este. No obstante, los objetivos específicos nos permiten desmenuzar los anteriores en unidades más pequeñas y fáciles de alcanzar, con el objetivo de trazar metas a corto plazo.

Los objetivos específicos del proyecto serían los siguientes:

- *Elaboración y visualización de cada uno los elementos de las gráficas*. Cada elemento de las gráficas deberá ser elaborado por el programa, incluyendo los ejes de coordenadas en los gráficos de barras y líneas y el texto que representa valores y etiquetas.
- *Elaboración de un algoritmo que permita optimizar la visualización de elementos en el gráfico*. Es posible que se deban representar una gran cantidad de valores, de forma que es necesario optimizar la aparición de estos en el gráfico.
- *Desarrollo de botones para agregar y eliminar series de datos y gráficas*. Se podrá añadir y eliminar series de datos y gráficas mediante sus botones pertinentes.
- *Opción que permita la carga de ficheros mediante "drag and drop"*. Se podrán cargar ficheros por dos vías: o bien seleccionándolo en el explorador de archivos por la vía clásica o arrastrando el fichero al cuadro.
- *Desarrollo de opciones de estilo para cada gráfica*. Cada gráfica dispondrá de una serie opciones para poder cambiar parámetros de estilos tales como el grosor de línea o el gradiente.
- *Desarrollo de un efecto de tres dimensiones para los gráficos de barras y circulares*. Una de las opciones permitidas será activar el efecto de tres dimensiones, que permitirá darle profundidad a las barras y al gráfico circular.
- *Representación de las secciones mediante colores fáciles de combinar*. Los colores que representarán las barras en el gráfico de barras y las secciones en los gráficos circulares deberán ser colores pastel que combinen fácilmente con la gráfica.
- *Visualización de los criterios utilizados en la serie de datos*. Será posible conocer aquellos criterios utilizados para la elaboración de las series de datos.

2. Planificación y Presupuesto

2.1. Metodología escogida

Para el desarrollo del proyecto la metodología ágil escogida ha sido Scrum. Este permite un desarrollo iterativo e incremental que encaja con el perfil de este proyecto. Esta metodología funciona esencialmente con iteraciones. Al principio del proyecto se deben concretar aquellas funcionalidades que se desean para este y la prioridad de cada una. Posteriormente, al principio de cada iteración se debe decidir incrementar o desarrollar una funcionalidad que sea integrable para, al final de la iteración, evaluar si el proceso ha sido elaborado correctamente o si es necesario realizar cambios.

Se utilizará GitHub como plataforma para almacenar y actualizar todo lo relativo al proyecto. Además, las plataformas que garantizan control de versiones son de gran utilidad junto con metodologías ágiles como Scrum. Aunque el Scrum existen varios roles, esto se verá simplificado debido a que el proyecto va a ser desarrollado en su totalidad por una persona. Tampoco se utilizarán para este proyecto artefactos como el *Product Backlog* o el *Sprint Backlog*.

2.2. Descripción de la metodología escogida

Para ello se han definido las siguientes características:

- Los Sprints tendrán una duración de 30 días. Cada Sprint contendrá dos iteraciones, es decir, una iteración estará formada.
- La duración total del proyecto será de 5 meses. Se han asignado dos iteraciones por Sprint, es decir, por mes, de forma que son un total de 10 iteraciones.
- La primera iteración se utilizará para adquirir todos los conocimientos relativos a la elaboración del proyecto, mientras que la última iteración será utilizada para realizar una revisión global del proyecto, pulir contratiempos del diseño y la implementación y evaluar el cumplimiento de los objetivos del proyecto.
- En las ocho iteraciones restantes se irán desarrollando de manera incremental las funcionalidades del proyecto. Los apartados de diseño e implementación se elaborarán de manera coordinada, con el objetivo de no realizar grandes cambios en caso de equivocación.
- Al final de cada iteración se evaluará si ha sido posible elaborar el incremento funcional planificado para poder realizar los cambios pertinentes.
- Debido a que el desarrollo es elaborado por una persona no serán necesarias algunas herramientas de cooperación de la metodología. No obstante, cada día se hará una revisión de aquello desarrollado.

2.3. Presupuesto

Aunque el proyecto no vaya a ser realmente desarrollado, se expone, a continuación, un presupuesto estimado del mismo.

Gastos de equipamiento

El equipamiento se estima que estará conformado por un único ordenador. Se ha supuesto un coeficiente de utilización del 5%, es decir, el porcentaje de dedicación al proyecto por parte del equipamiento. A continuación, se muestra la información desglosada:

- Asus Zenbook 13 OLED (UX325).
    - Precio estimado de adquisición: 850 €.
    - Vida útil estimada: 5 años. Su valor residual estimado es nulo.
    - Amortización anual: 850/5 = 170 €/año. Se ha supuesto amortización lineal.
    - Valor actual del ordenador (se ha supuesto que ha pasado un año desde su compra): 850 – 170 = 680 €.
    - Gasto correspondiente del equipamiento al proyecto: 680 * 0,05 = 34 €.
- El gasto total estimado en equipamiento es de 34 €.

Gastos de personal

El personal se estima que se encontrará constituido por un trabajador. Se ha supuesto que el personal le dedica 5 meses a la elaboración del proyecto y que el sueldo del trabajador es 1.000 €.

- Gasto correspondiente del personal: 1.000 * 5 = 5.000 €.
- El gasto total estimado en personal es de 5.000 €.

El gasto expuesto previamente se encuentra en términos brutos. Si suponemos unas retenciones del 19% estas serían de 5.000 * 0,19 = 950. Por otro lado, si suponemos una cotización a la Seguridad Social del 30%, esta sería de 5.000 * 0,30 = 1.500 €. Estas se encuentran incluidas en el gasto.

Presupuesto total

El gasto total resultaría de la suma del gasto de equipamiento y del gasto de personal, es decir, 34 + 5.000 = 5.034 €. Por ende, el presupuesto total sería de 5.044 €.

3. Especificación de requerimientos

3.1. Requisitos funcionales

A continuación, se presenta un listado de requisitos funcionales de la aplicación:

RF.1	**Inicio de la aplicación web**
Descripción	Posibilidad de ejecuta la aplicación web
Datos de entrada	Ninguno
Datos de salida	Ninguno

RF.2	**Cierre de la aplicación web**
Descripción	Posibilidad de terminar la aplicación web
Datos de entrada	Ninguno
Datos de salida	Ninguno

RF.3	**Visualización de gráficos de barras**
Descripción	Los datos importados del archivo .csv podrán visualizarse en un gráfico de barras
Datos de entrada	Datos procesados obtenidos del archivo .csv
Datos de salida	Gráfico de barras que representa los datos pertinentes

RF.4	**Visualización de gráficos de líneas**
Descripción	Los datos importados del archivo .csv podrán visualizarse en un gráfico de líneas
Datos de entrada	Datos procesados obtenidos del archivo .csv
Datos de salida	Gráfico de líneas que representa los datos pertinentes

RF.5	**Visualización de gráficos circulares**
Descripción	Los datos importados del archivo .csv podrán visualizarse en un gráfico circular
Datos de entrada	Datos procesados obtenidos del archivo .csv
Datos de salida	Gráfico circular que representa los datos pertinentes

RF.6	**Conocer el número de variables utilizadas para cada gráfico**
Descripción	
Datos de entrada	Datos procesados obtenidos del archivo .csv
Datos de salida	Número indicando el número de variables

RF.7	**Insertar archivos ".csv", con el objetivo de visualizar dichos datos**
Descripción	Se podrán utilizar datos organizados en archivos .csv, con el fin de utilizarlos para las gráficas
Datos de entrada	Archivo .csv
Datos de salida	Datos procesados en su respectiva estructura de datos

RF.8	**Conocer información relacionada con el proyecto en un apartado de información**
Descripción	Apartado que permite conocer información relacionada con el proyecto
Datos de entrada	Ninguno
Datos de salida	Ninguno

3.2. Requisitos no funcionales

RNF.1	**La aplicación se ejecutará en un navegador web actual en cualquier sistema operativo, mediante la visualización de una página web con los controles y gráficas**
Descripción	El desarrollo de la aplicación web se realizará utilizando las herramientas web pertinentes con el fin de la visualización de gráficas

RNF.2	**La aplicación tendrá fácil navegabilidad**
Descripción	La aplicación será usable en términos de navegabilidad. Se podrá acceder a todo con pocos *clicks*.

RNF.3	**La aplicación se encontrará disponible en inglés**
Descripción	El lenguaje del que dispondrá la aplicación será el inglés

3.3. Bocetos iniciales

Los bocetos son una forma sencilla e intuitiva para un primer contacto con la aplicación. A continuación, se muestran los bocetos de la aplicación:

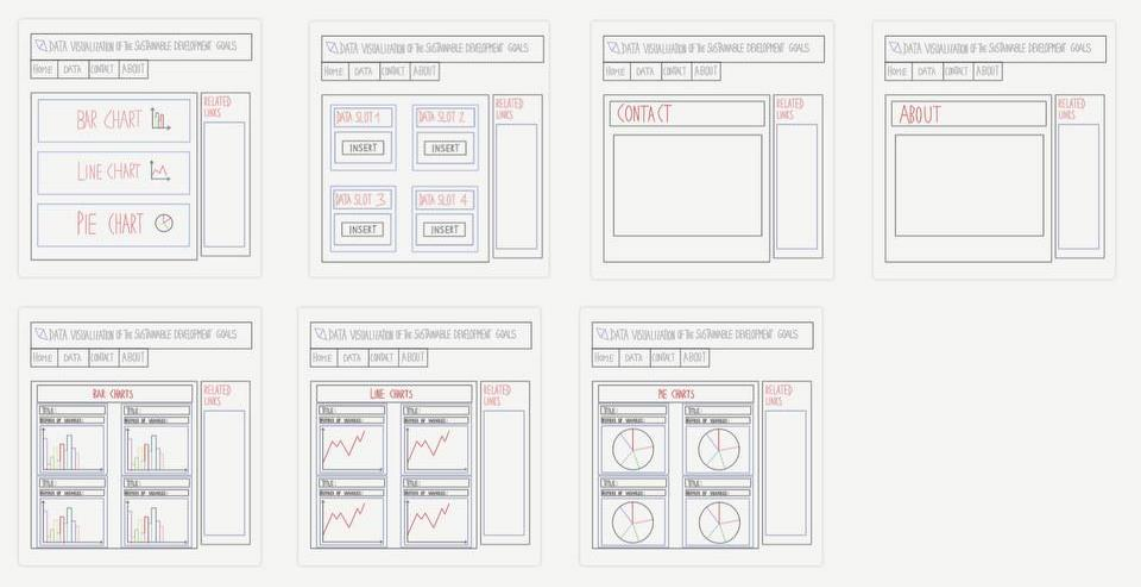

Fig X. Ejemplo

Con el objetivo de simplificar el boceto, se han sintetizado algunos conceptos de la versión final de la aplicación. Un ejemplo importante es el hecho de que se han dibujado únicamente cuatro gráficas. No obstante, la aplicación web será un sistema en el que será posible visualizar gráficas sin que exista un límite preestablecido en estas.

La aplicación web dispondrá de una pestaña "*Home*", la cuál será la pestaña de inicio. En ella se podrá elegir el tipo de gráfico que se desea: gráfico de barras, de líneas o circular. Una vez se pinche en algunas de estas opciones, se redirigirá a la pestaña pertinente donde se mostrarán los gráficos con los datos establecidos (en caso de no haber introducido ningún dato, existirán unos datos preestablecidos).

También se dispone de una pestaña "*Data*", la cual está desarrollada con la intención de indicar cuáles son los datos que queremos para cada una de las gráficas. Como se ha indicado previamente, no existe un límite preestablecido de gráficas, de forma que una vez se inserte un archivo de datos aparecerá una ranura que permitirá introducir otro archivo de datos, y así sucesivamente.

A modo de desglose:

1. **Página principal**. Se accede pinchando en la pestaña "*Home*", aunque es la "*Landing Page*" de la aplicación. Boceto:

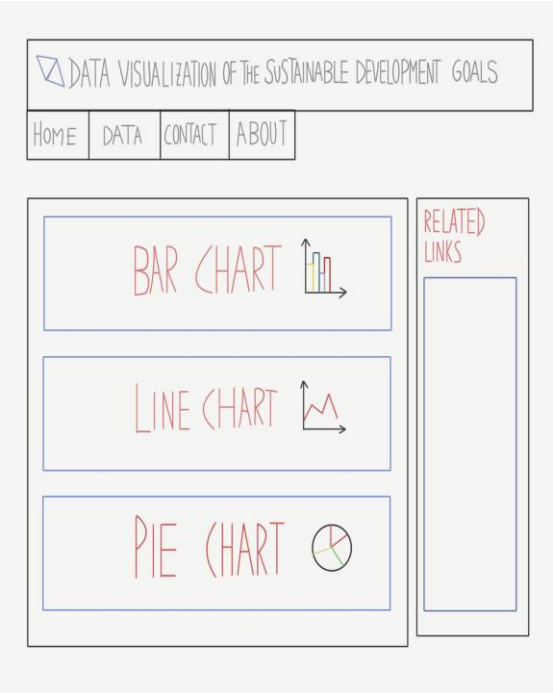

Fig X. Ejemplo

2. **Pestaña de selección de los datos**. Se accede pinchando en la pestaña "*Data*". En ella se podrán seleccionar los datos que queremos ver representados en las gráficas. Se puedan seleccionar hasta cuatro archivos de datos. Boceto:

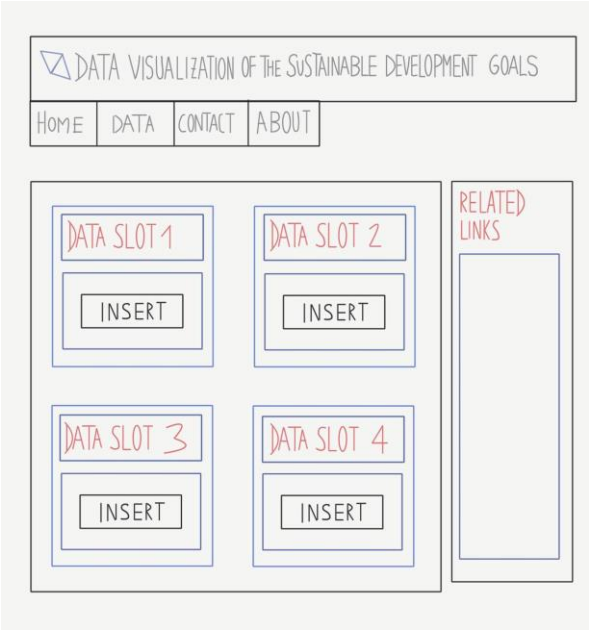

Fig X. Ejemplo

3. **Contacto e Información relacionada**. Se puede acceder pinchando en las pestañas "*Contact*" y "*About*", respectivamente. Bocetos:

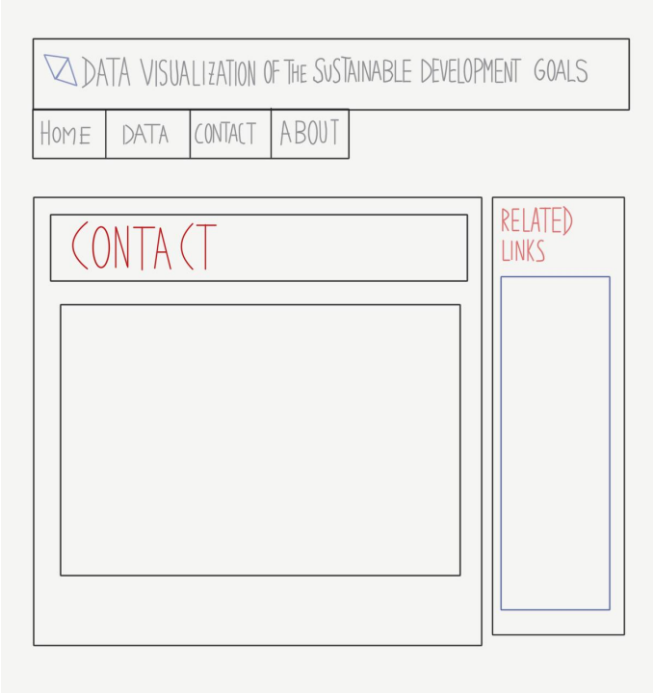

Fig X. Ejemplo

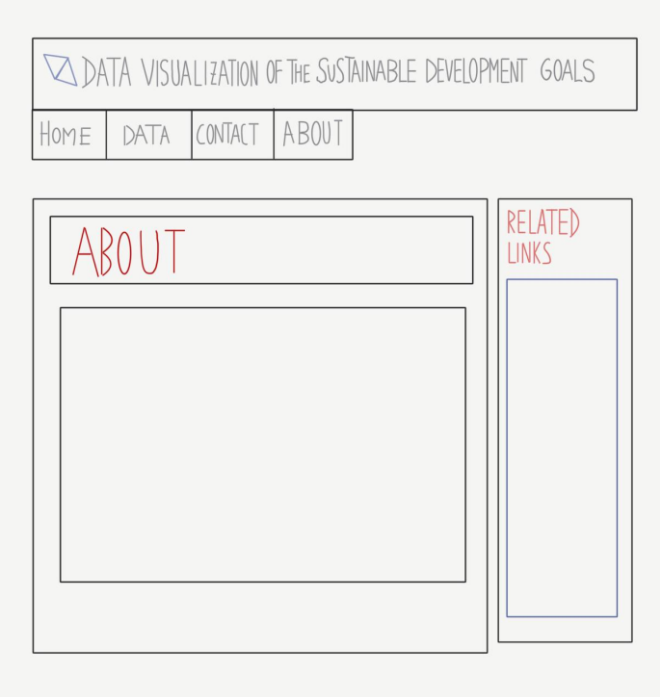

Fig X. Ejemplo

4. **Visualización de gráficas de barras, líneas y circulares**. Se puede acceder pinchando dentro de la pestaña “Home” en los apartados “Bar Chart”, “Line Chart” y “Pie Chart”, respectivamente. Bocetos:

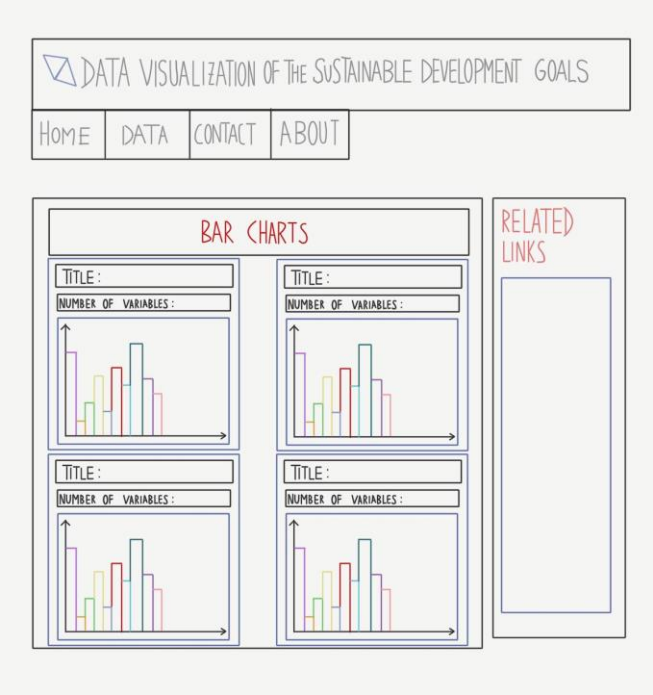

Fig X. Ejemplo

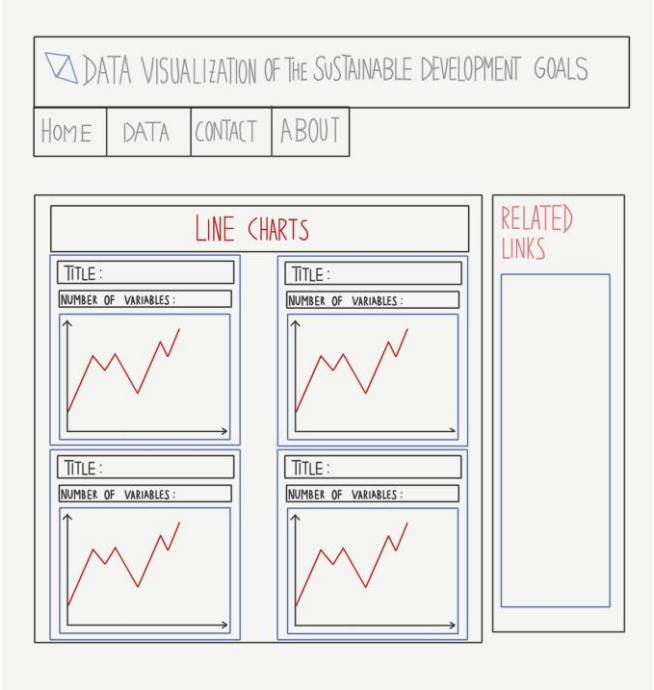

Fig X. Ejemplo

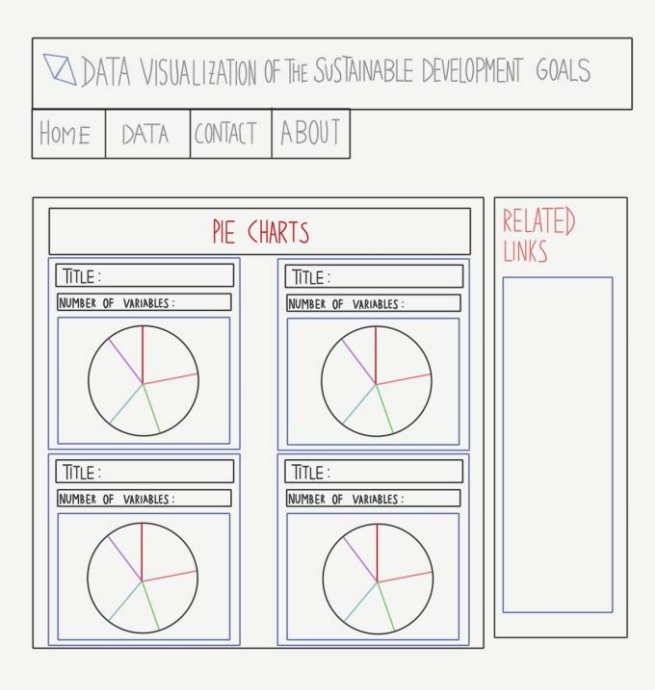

Fig X. Ejemplo

4. Diseño

4.1. Formato de los ficheros de datos

Los gráficos que, posteriormente, serán visualizados, obtienen la información de un fichero de datos que contiene las variables, los valores y el periodo de esta. Este debe tener un formato en particular:

- Debe ser un fichero ".csv" separado por punto y coma.
- El periodo debe encontrarse en la penúltima columna y los valores deben encontrarse en la última columna. El formato debe ser como indica la imagen:

	A	B	C	D	E	F
1	Variable1	Valor1	PERIODO	VALOR		
2	Nombre	ODS - 1.5.4, 1	2021	68		
3	Nombre	ODS - 1.5.4, 1	2020	42		
4	Nombre	ODS - 1.5.4, 1	2019	40		
5	Nombre	ODS - 1.5.4, 1	2018	37		
6	Nombre	ODS - 1.5.4, 1	2017	32		
7	Nombre	ODS - 1.5.4, 1	2016	30		
8	Nombre	ODS - 1.5.4, 1	2015	28		
9						
10						
11						
12						

Fig X. Ejemplo de formato correcto de fichero de datos

- No se deben escribir datos en la primera fila, ya que esta únicamente contiene los títulos y no será procesada.

Los datos serán procesados previamente a la visualización. Es necesario obtener, en primer lugar, el periodo de la serie de datos y los valores de esta. También es necesario obtener las variables o criterios de selección utilizados.

Aunque la aplicación puede utilizar cualquier fichero de datos que tenga este formato, los ficheros utilizados serán obtenidos del Instituto Nacional de Estadística, en particular, series de datos del cumplimiento de los Objetivos de Desarrollo Sostenible. El periodo deberán ser números enteros, mientras que los valores podrás ser o números reales. No obstante, todo ello será procesado correctamente.

4.2. Diagrama de clases

El diagrama de clases es el siguiente:

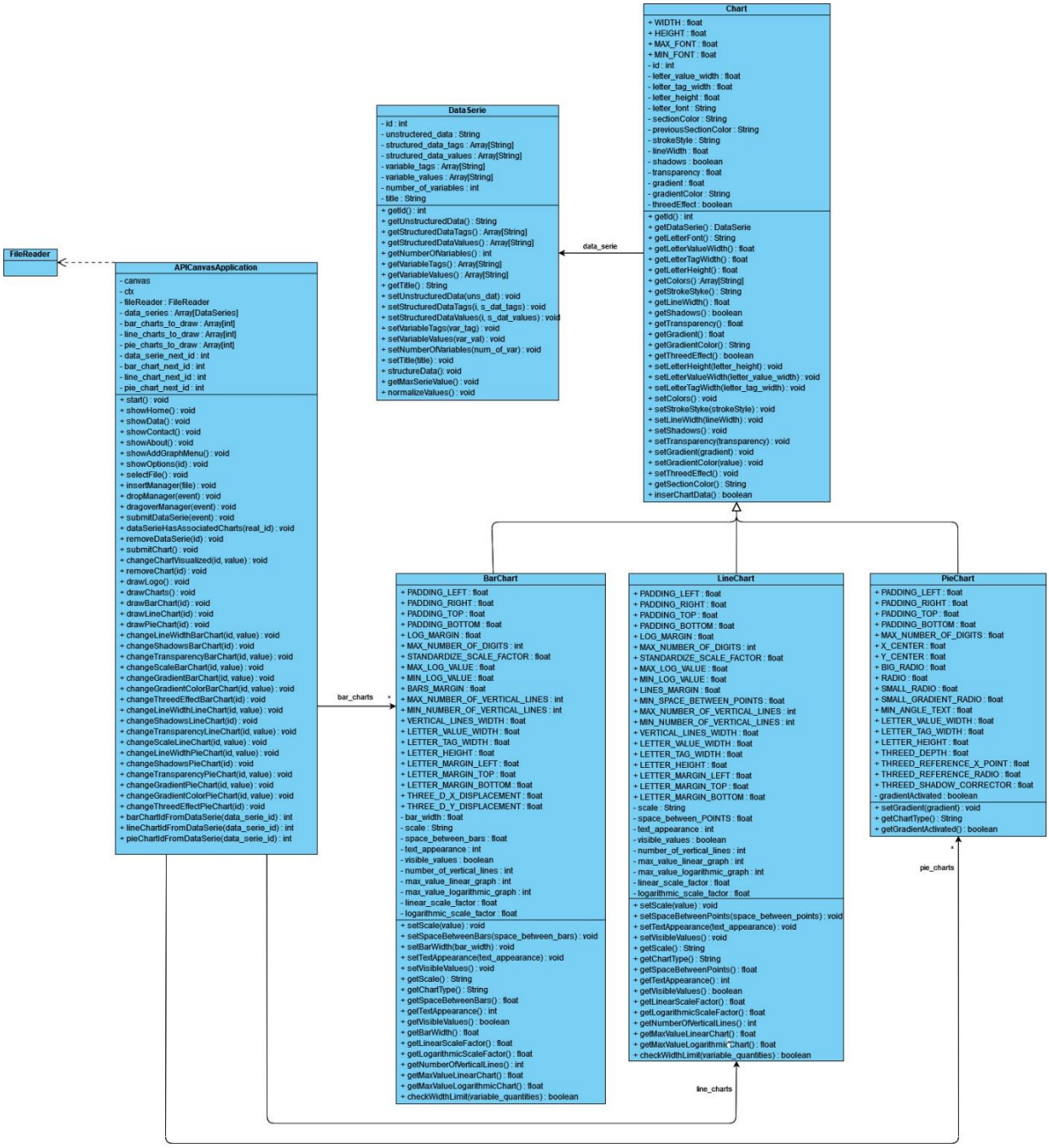

Fig X. Ejemplo

4.3. Diagramas de interacción

Trazado de gráficos de barras

El diagrama de interacción para el trazado de gráficas de barras es el siguiente:

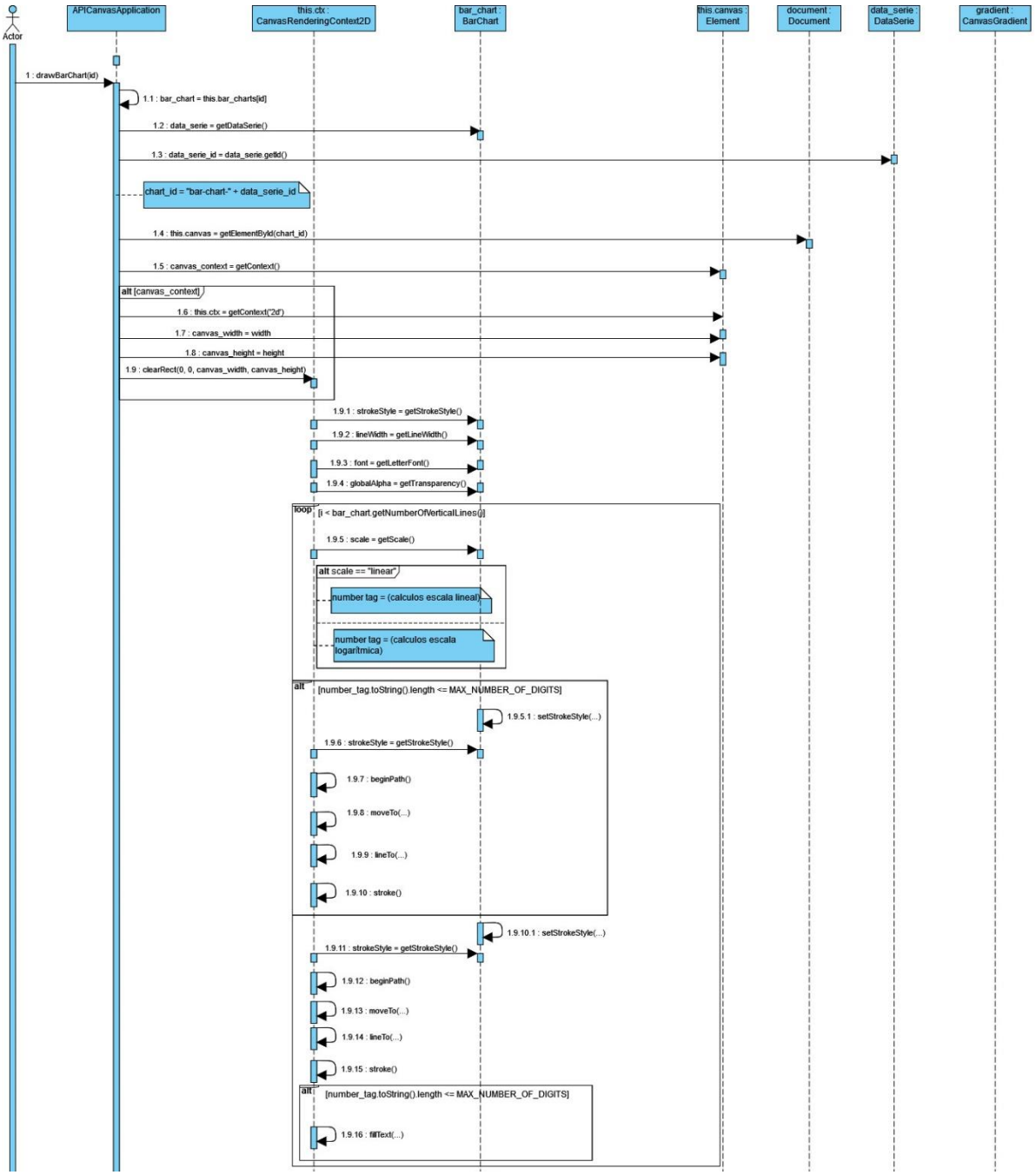

Fig X. Ejemplo

Continúa a continuación…

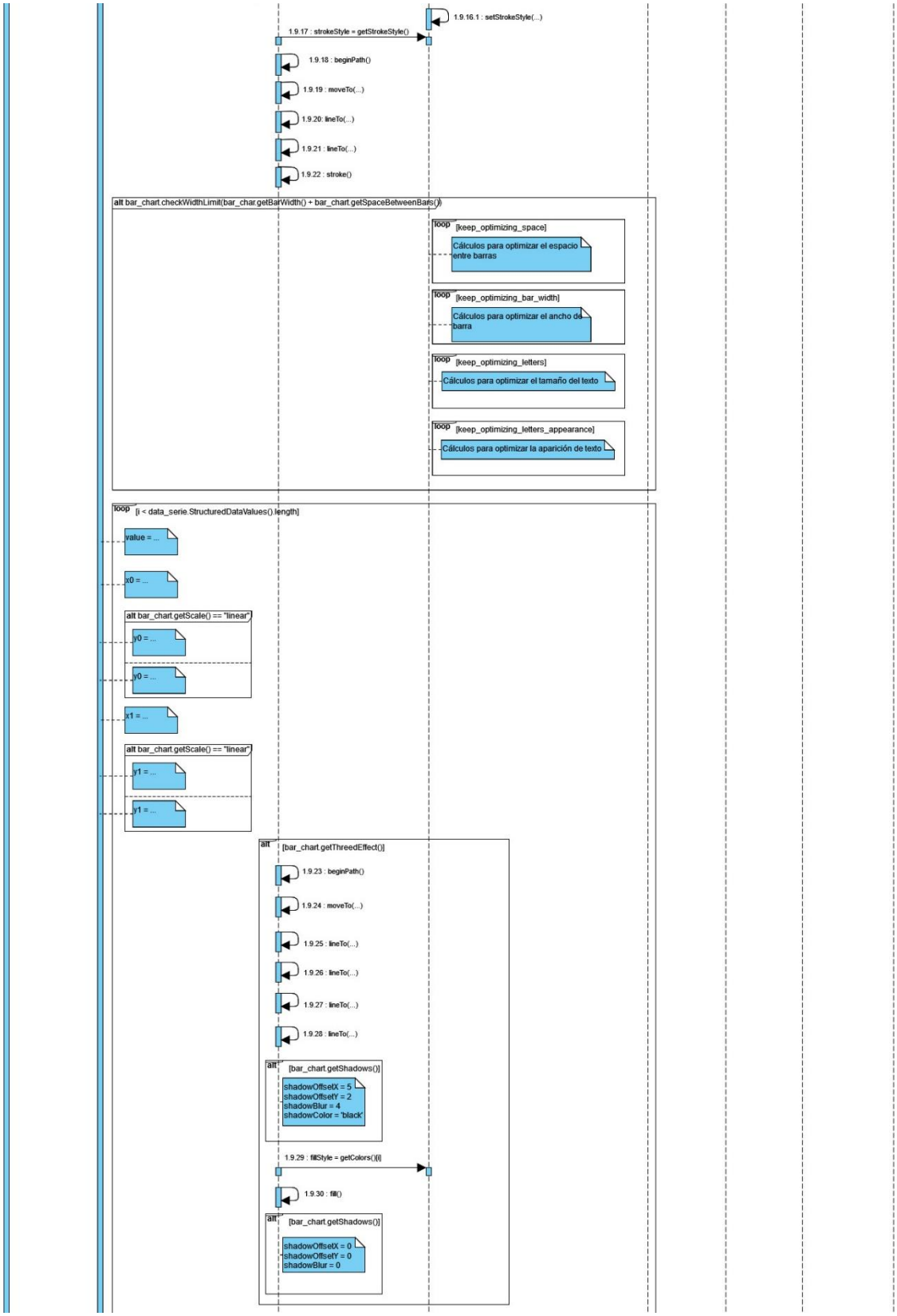

Fig X. Ejemplo

Continúa a continuación…

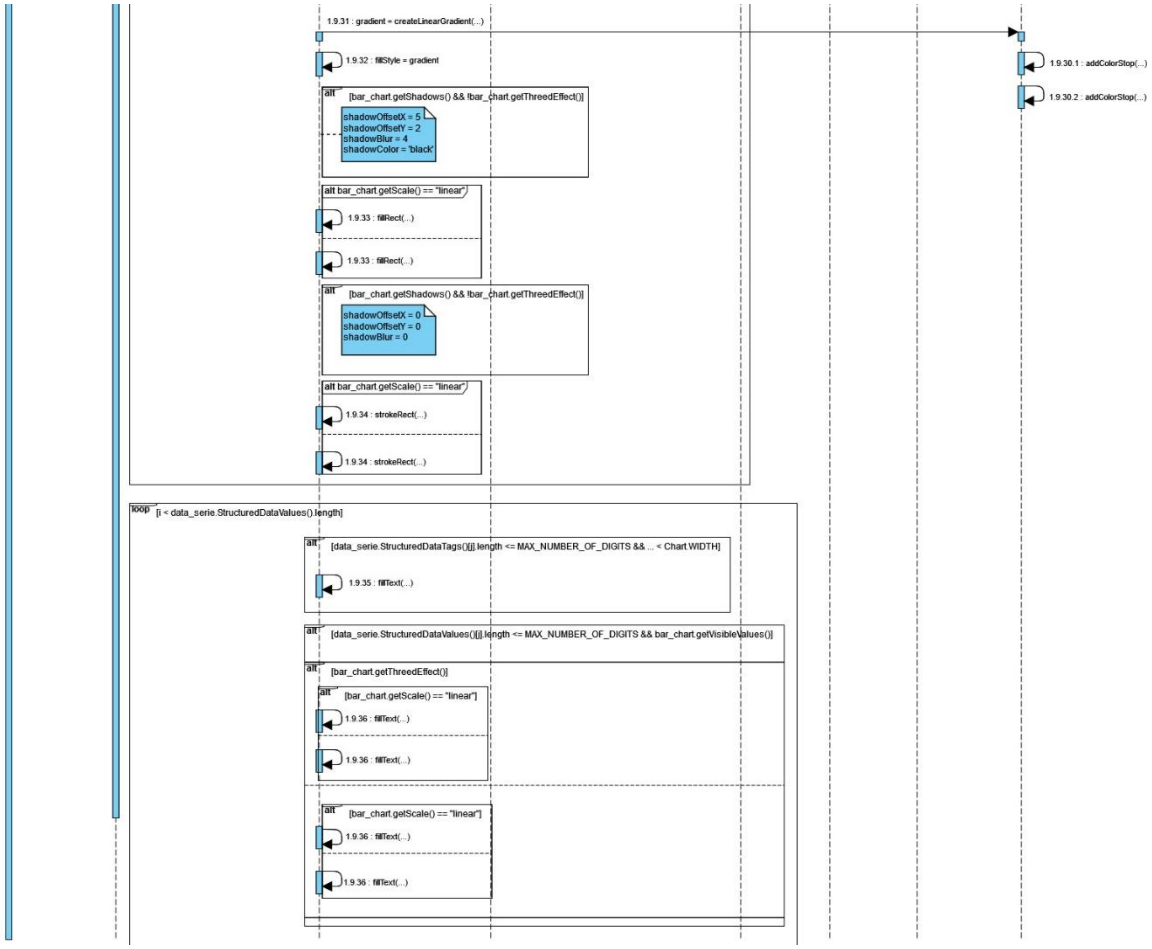

Fig X. Ejemplo

Dicho proceso puede dividirse en distintas partes:

- *Trazado de los ejes de coordenadas*. Ambos ejes tienen unas medidas estáticas, de forma que no dependen del número de valores o del valor máximo para su trazado. En este proceso también se incluye el trazado de las líneas horizontales de referencia y sus respectivos valores y etiquetas.
- *Trazado de rectángulos*. Representan a cada uno de los elementos de la serie y sus respectivos valores. En función del valor máximo de la serie se hará una escala para poder cubrir el espacio requerido. En caso de que haya demasiados valores para el ancho especificado, aparecerán solo la mitad de estos y así sucesivamente.
- *Escala utilizada*. Para el cálculo de los rectángulos, es importante tener en cuenta la escala utilizada. Estos varían en función de si la escala es linear o logarítmica.
- *Añadido de atributos de estilo*. Cada atributo afectará, o bien, a elementos específicos, o bien, el conjunto total del trazado. Las sombras y el efecto de tres dimensiones deben afectar solo a los rectángulos, mientras que la opacidad o el grosor de línea afectarán a todo el conjunto. Los atributos de estilo utilizados son grosor de línea, gradiente, color del gradiente, sombras, opacidad y efecto de tres dimensiones.

Trazado de gráficos de líneas

El diagrama de interacción para el trazado de gráficas de líneas es el siguiente:

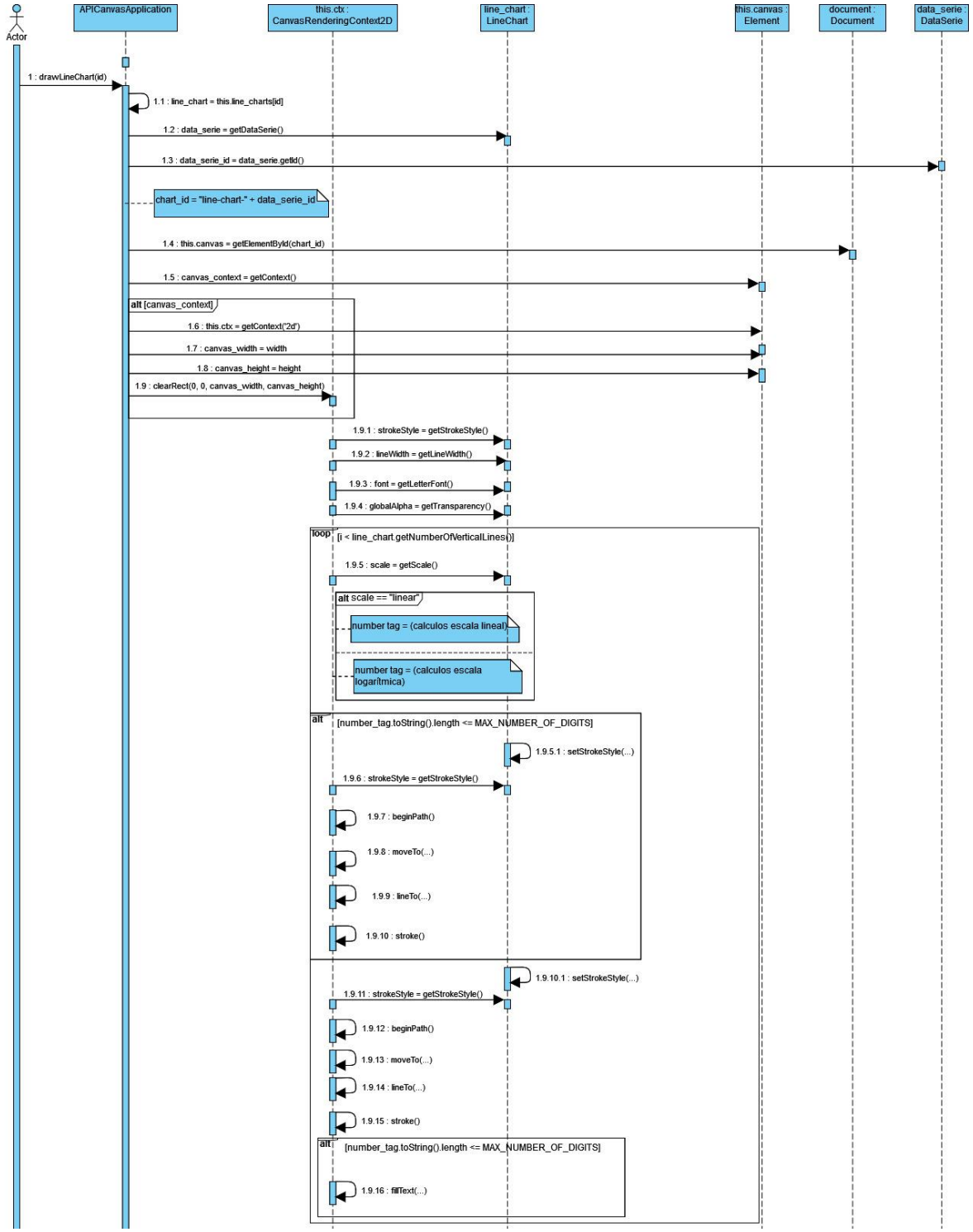

Fig X. Ejemplo

Continúa a continuación…

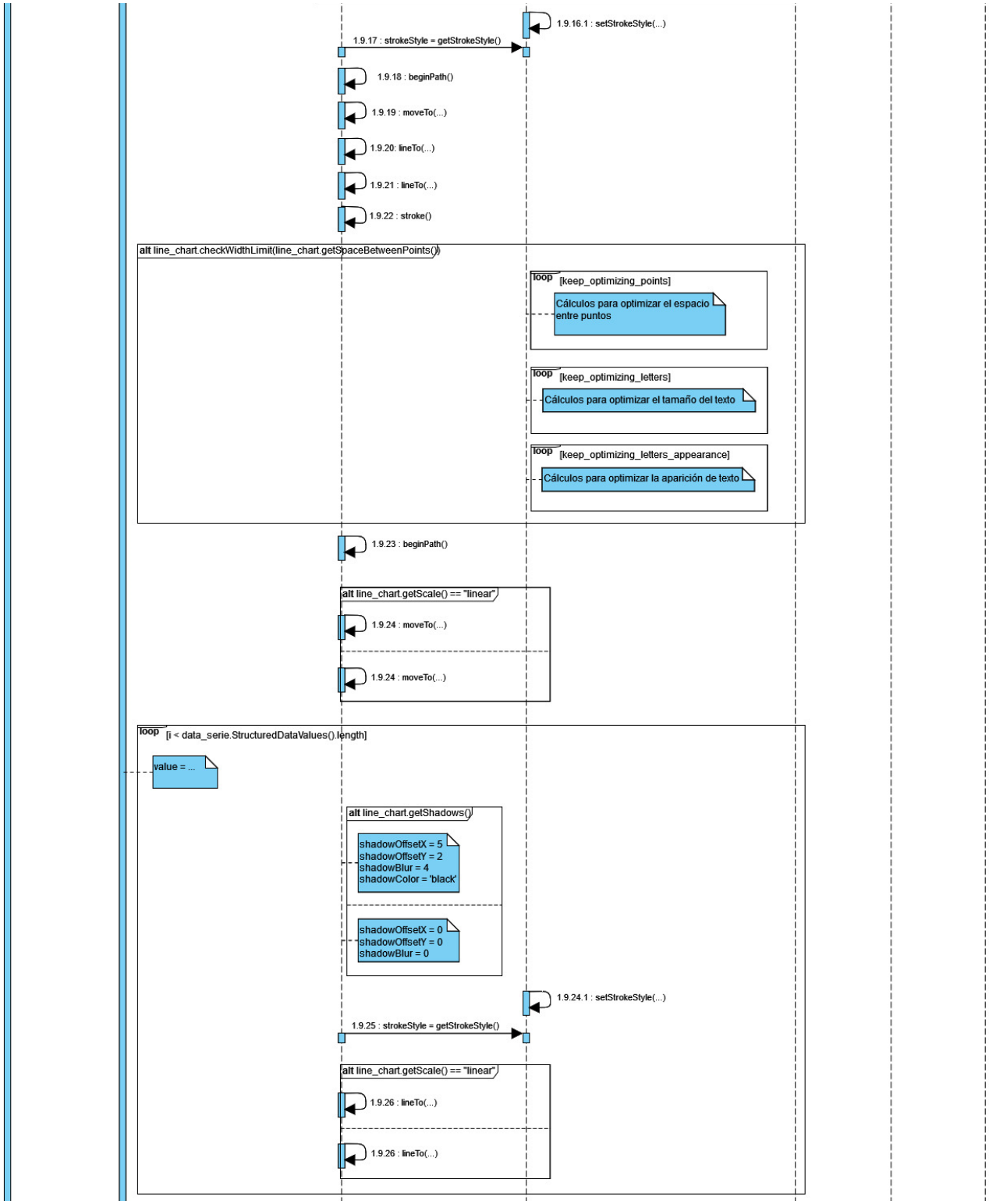

Fig X. Ejemplo

Continúa a continuación…

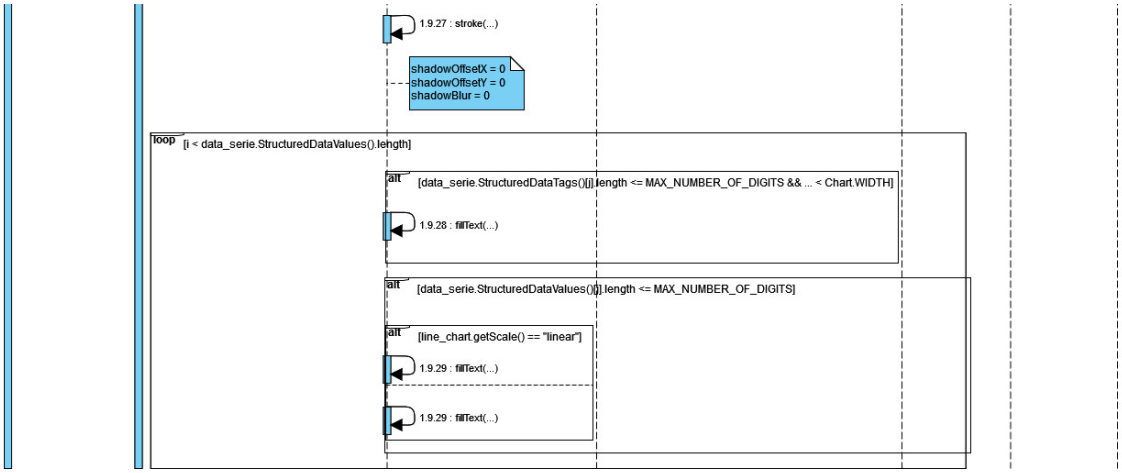

Fig X. Ejemplo

Dicho proceso puede dividirse en distintas partes:

- *Trazado de los ejes de coordenadas*. Al igual que el gráfico de barras, es necesario trazas los ejes de coordenadas, con las mismas características y propiedad que el anterior gráfico.
- *Trazado de líneas*. Hacen alusión a los valores de la serie. También es necesario escalar el eje vertical utilizando el valor máximo de la serie y comprobar si se ha llegado al ancho máximo mediante el número total de valores.
- *Escala utilizada*. Para calcular la ordenada de cada punto de la línea principal es fundamental tener en cuenta si la escala utilizada es linear o logarítmica, ya que los cálculos varían en función de esto.
- *Añadido de atributos de estilo*. Los atributos de estilo utilizados son grosor de línea, sombras y opacidad. Posee menos atributos que el gráfico de barras debido al área reducida de la línea.

Trazado de gráficos circulares

El diagrama de interacción para el trazado de gráficas circulares es el siguiente:

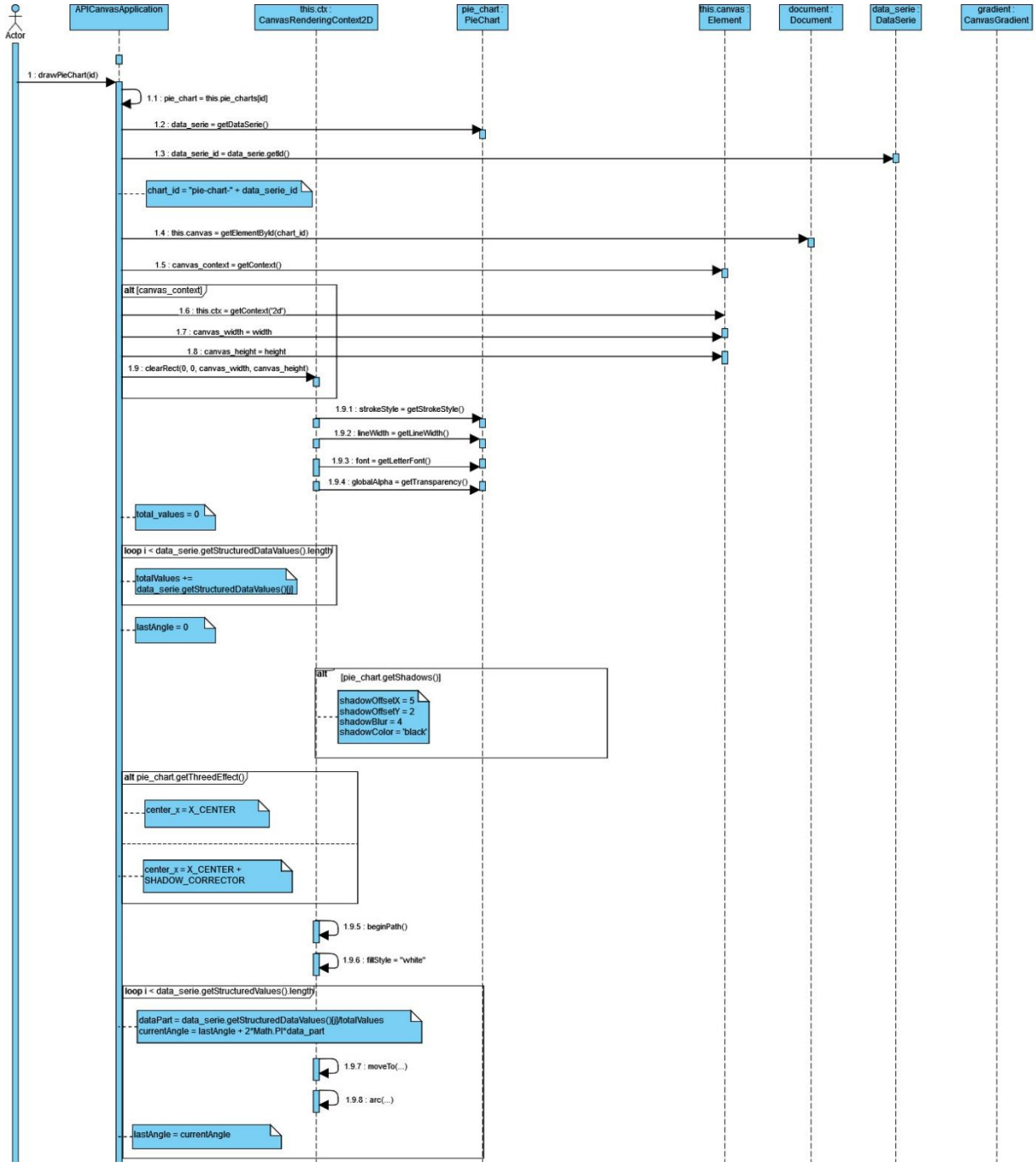

Fig X. Ejemplo

Continúa a continuación…

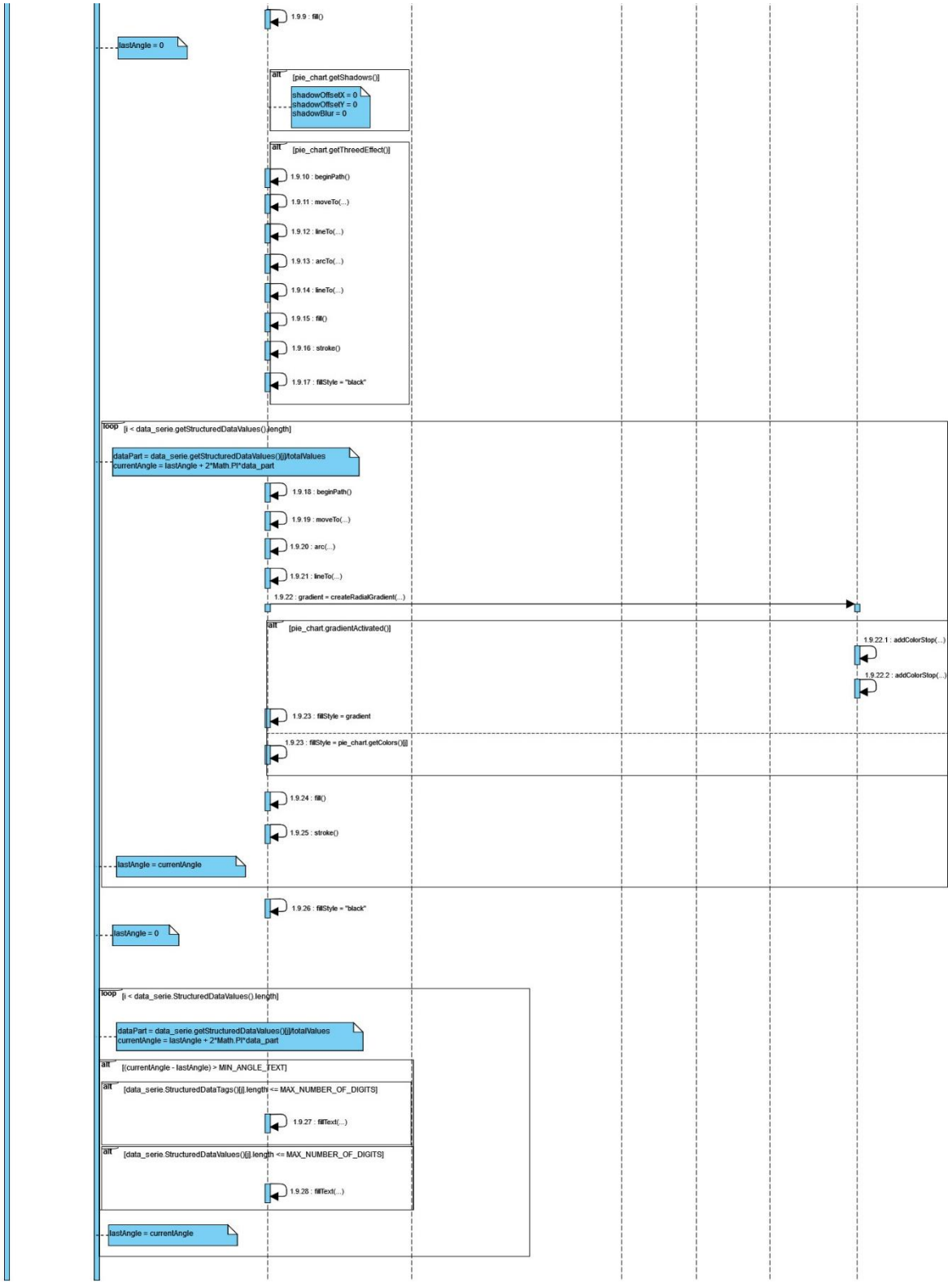

Fig X. Ejemplo

El proceso del trazado del gráfico circular difiere de los dos anteriores gráficos. No obstante, será explicado posteriormente en el apartado de “Implementación”.

4.4. Bocetos de las interfaces de usuario

Los bocetos iniciales permitían una primera aproximación a la apariencia de la interfaz de usuario. No obstante, a lo largo del proyecto, su apariencia se ha ido modificando para, finalmente, alcanzar una versión final de esta. Los siguientes bocetos mostrarán las pestañas de navegación de la aplicación, así como los menús desplegables para seleccionar datos u otros elementos de la aplicación.

Pestaña “Gráficas”

Esta pestaña es la página principal de la aplicación, la primera que se encuentra el usuario al abrir la misma. También se puede acceder a la misma pinchando en la pestaña “Gráficas”. En esta pestaña se podrán visualizar las gráficas que el usuario haya creado. Para añadir una gráfica es necesario pinchar en el botón “Añadir gráfica”. Esto nos permitirá visualizar un menú que nos pedirá elegir una serie de datos y un tipo de gráfica. En caso de que no se haya seleccionado alguna de las dos opciones aparecerá un mensaje en pantalla para indicarlo. Lo mismo sucederá si se intenta crear una gráfica ya existente. Además, cada gráfico dispone de una serie de opciones que modifican parámetros de estilo de este. Para acceder a las opciones, es necesario pinchar en el botón “Mostrar opciones”. La siguiente imagen muestra la página según se inicia:

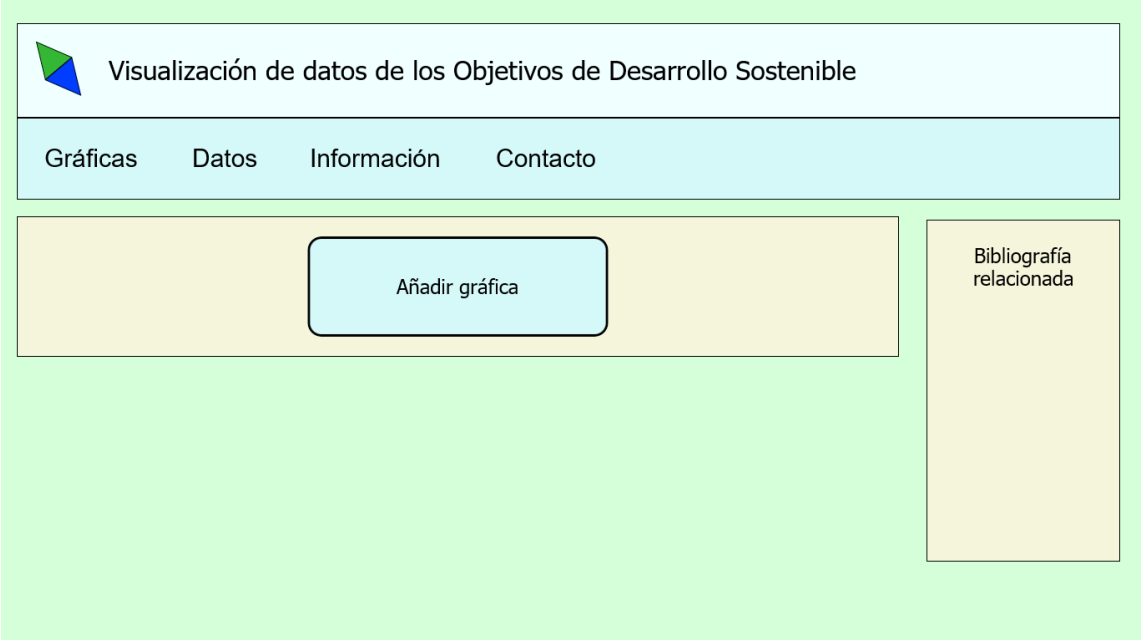

Fig X. Ejemplo

El siguiente boceto muestra el menú de selección de la gráfica:

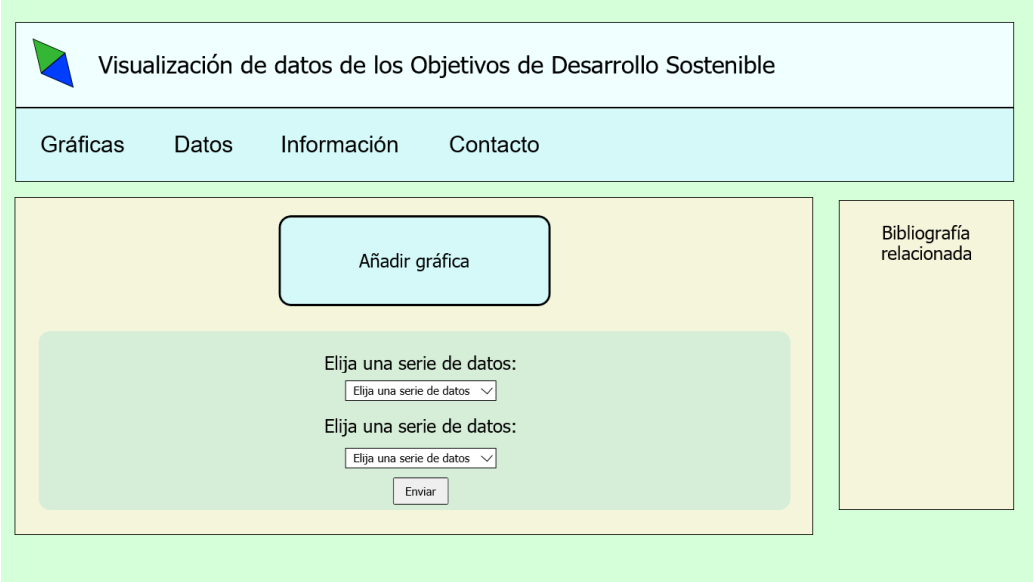

Fig X. Ejemplo

Así se muestra la pestaña una vez insertada una gráfica:

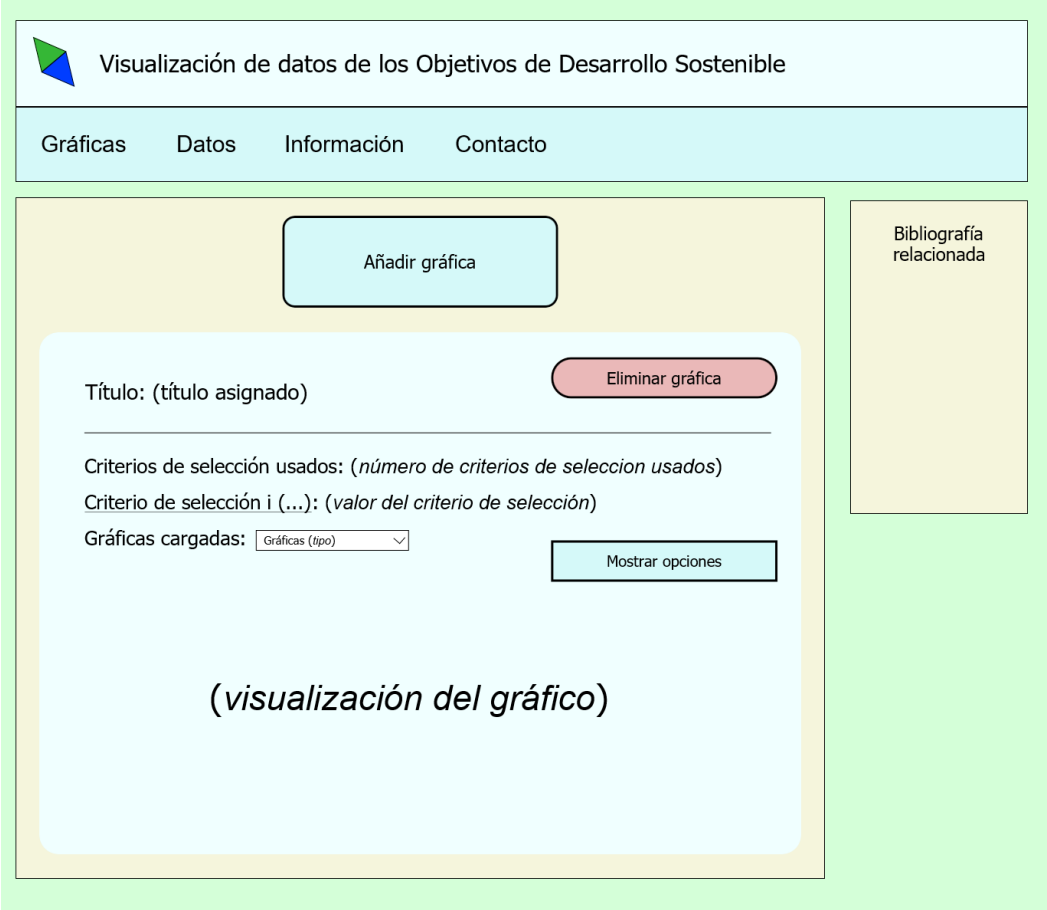

Fig X. Ejemplo

Finalmente, así es como se muestra la pestaña una vez desplegado el menú de opciones de alguna gráfica:

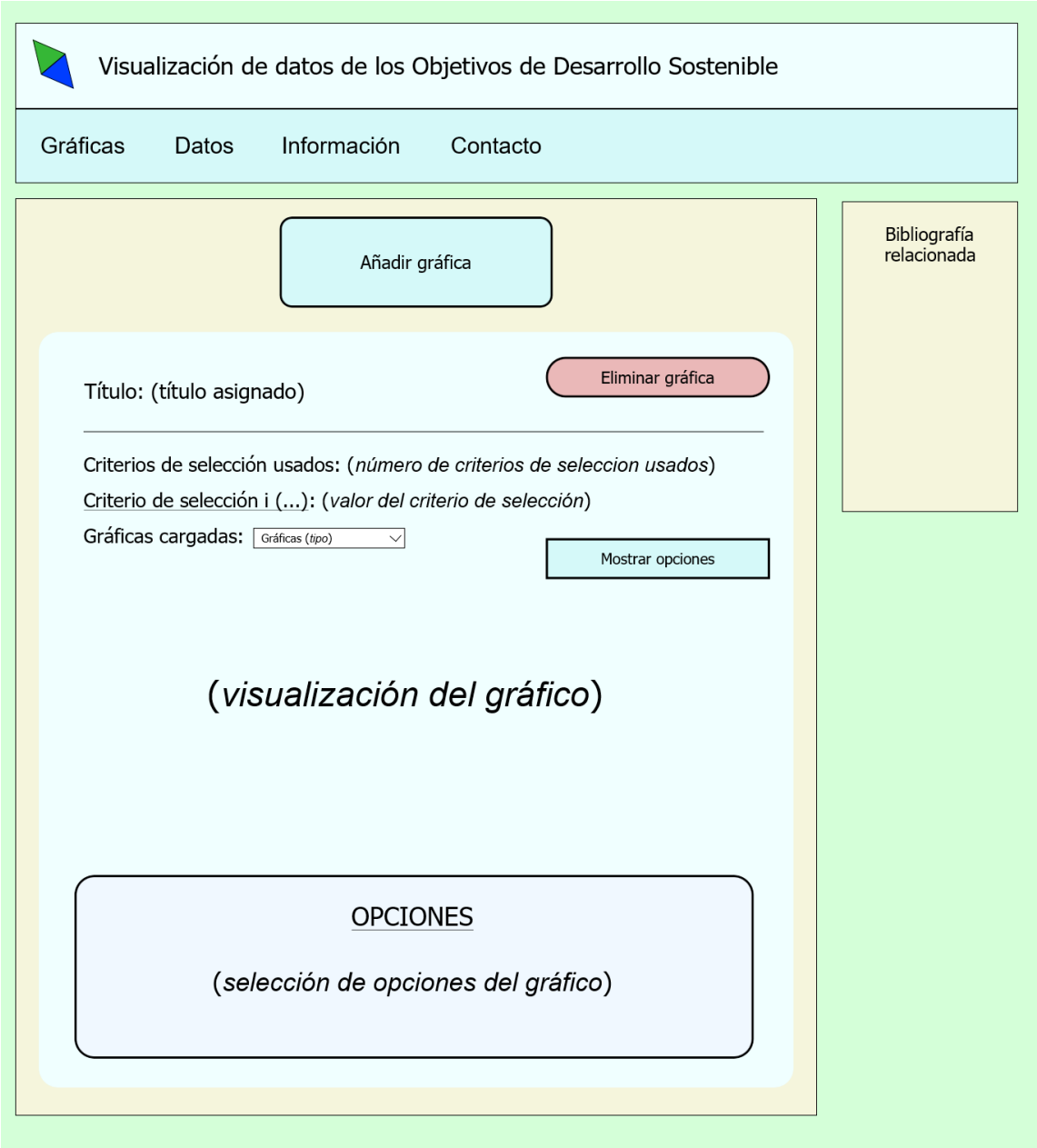

Fig X. Ejemplo

Pestaña “Datos”

Para acceder es necesario pinchar en la pestaña “Datos”. Esta se utiliza para subir las series de datos que conformarán la fuente de datos de la gráfica. En primer lugar, se debe pinchar en el texto “Seleccione un archivo .csv”. Esto abrirá el explorador de archivos para seleccionar un fichero. Es necesario que sea en formato “.csv” (más especificaciones del formato del archivo se describirán en la pestaña “Información”). Una vez seleccionado el archivo, será necesario proporcionar un título para la serie de datos y pulsar “Enviar”. Una vez hecho esto, aparecerá un nuevo cuadro con toda la información de esta. Para eliminar la serie de datos se debe pulsar en el botón “Eliminar serie de datos”. A continuación, se muestra un boceto de la sección nada más pinchar en la pestaña “Datos”:

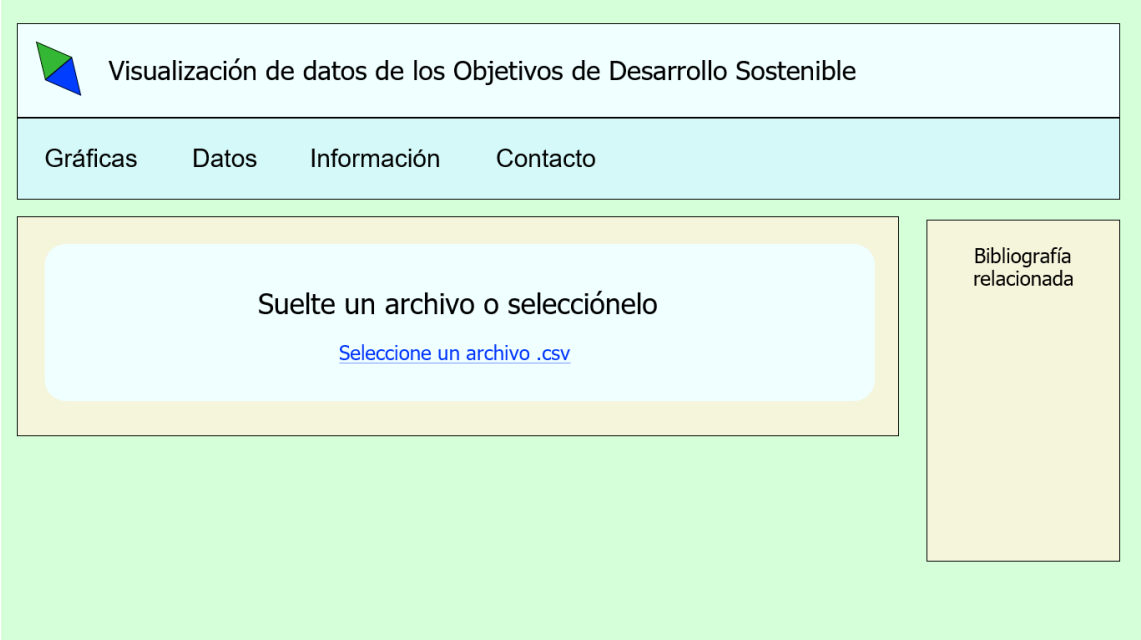

Fig X. Ejemplo

A continuación, se muestra la pestaña una vez desplegada la selección del título:

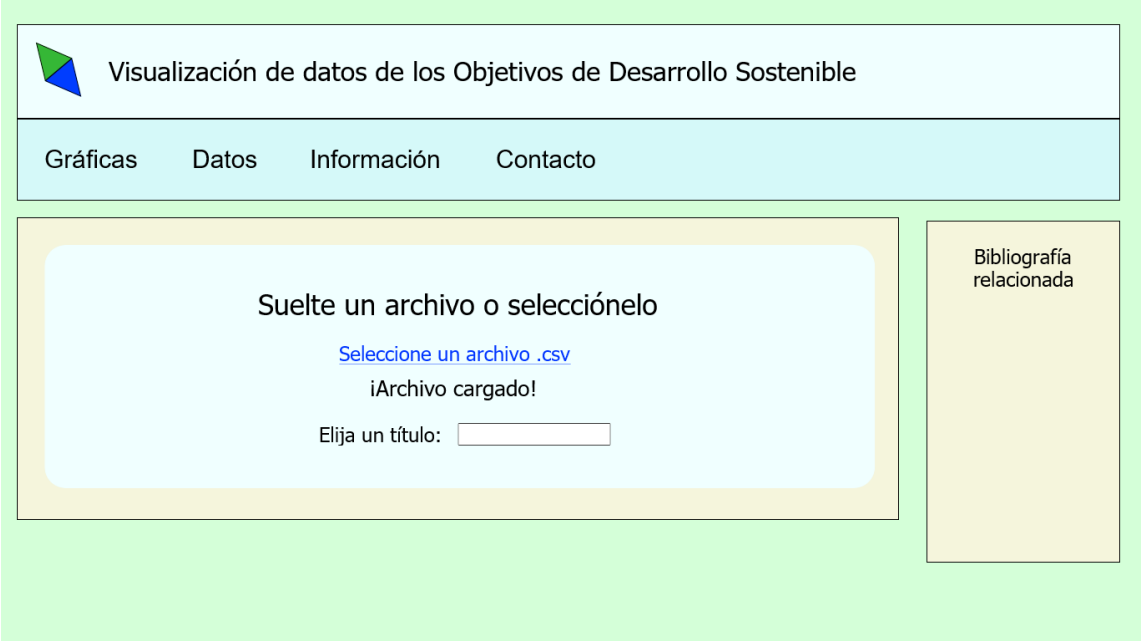

Fig X. Ejemplo

Finalmente, así se visualiza la pestaña una vez creada una serie de datos:

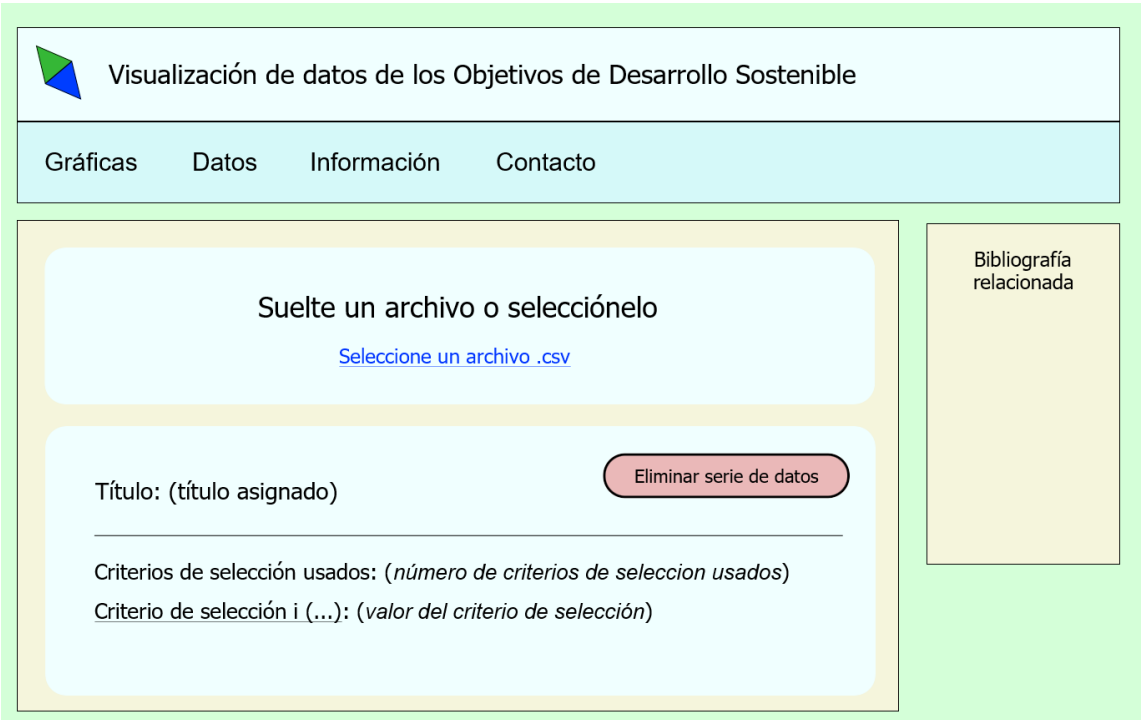

Fig X. Ejemplo

Pestaña “Información”

Esta pestaña muestra aquella información necesaria para el usuario como, por ejemplo, el formato de los ficheros de las series de datos. El boceto se muestra a continuación:

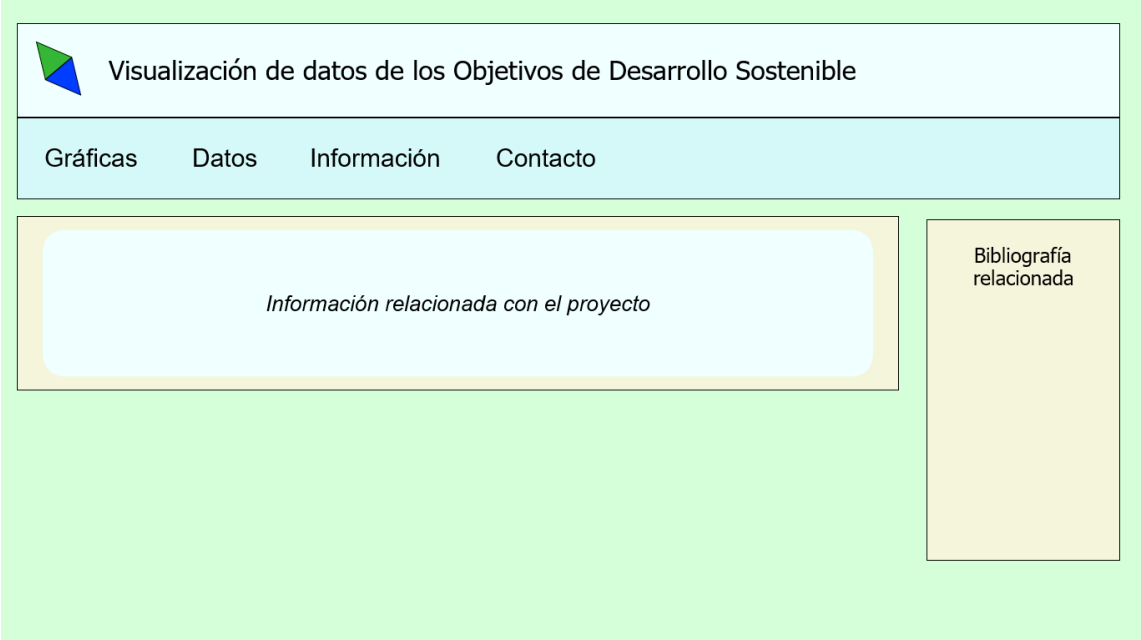

Fig X. Ejemplo

Pestaña “Contacto”

Finalmente, la pestaña de contacto muestra información relativa al proyecto y al autor y al tutor de este. El boceto es el siguiente:

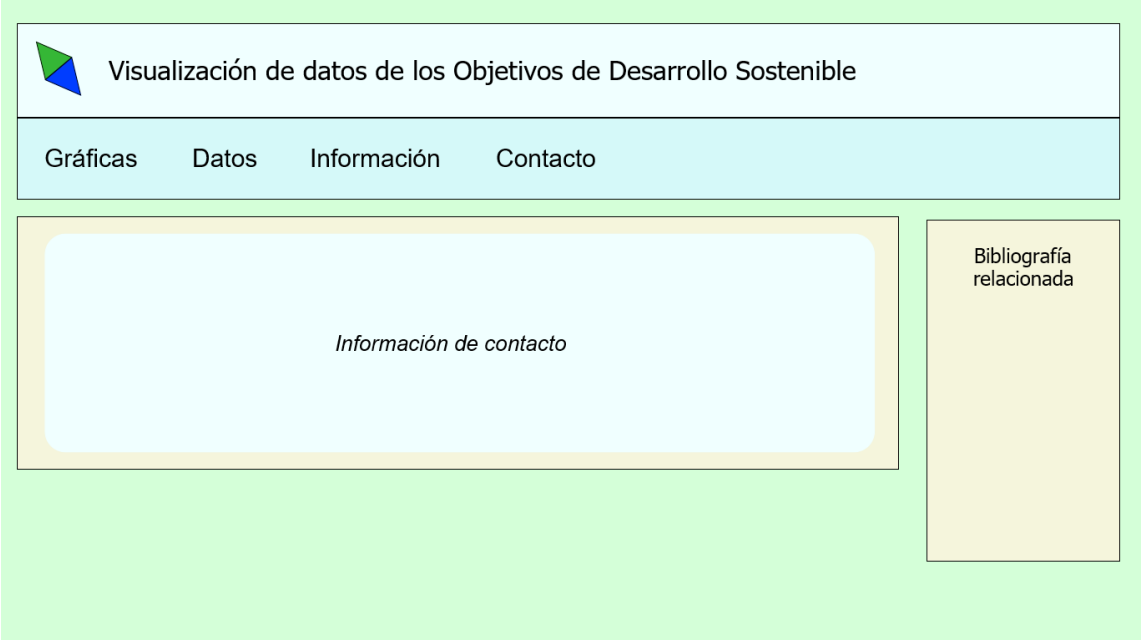

Fig X. Ejemplo

4.5. Estructuras de datos fundamentales

Aquellas estructuras de datos necesarias para el desarrollo de la aplicación son:

- *Colección de gráficos de barras*. Está compuesto por todos los gráficos de barras preparados para su visualización. Cada uno tiene asociado una serie de datos.
- *Colección de gráficos de líneas*. Formado por todos aquellos gráficos de líneas listos para ser visualizados. Todos ellos tienen una serie de datos asociada.
- *Colección de gráficos circulares*. Constituido por todos los gráficos circulares habilitados para su visualización. Cada gráfico individual se encuentra asociado con una serie de datos.
- *Colección de identificadores de gráficos de barras a trazar*. Colección compuesta por los identificadores de aquellos gráficos de barras que están siendo visualizadas. No todos gráficos disponibles asociados a una serie de datos son visualizados a la vez, es el usuario el que elige cuáles quiere visualizar.
- *Colección de identificadores de gráficos de líneas a trazar*. Estructura formada por los identificadores de aquellos gráficos de líneas que pueden ser visualizados. Por la misma razón que se ha comentado anteriormente, no todos los gráficos son visualizados a la vez, de forma que es necesario distinguir cuáles sí van a ser visualizados.
- *Colección de identificadores de gráficos circulares a trazar*. Se encuentra constituido por los identificadores de los gráficos circulares que son susceptibles de ser visualizados. Al igual que lo explicado previamente, es importante entender que no todos los gráficos van a poder ser visualizados al mismo tiempo.
- *Colección de series de datos*. Constituido por aquellas series de datos cargadas por el usuario. Los identificadores de las series de datos son utilizados para agrupar las gráficas de tres en tres (un gráfico de barras, de líneas y circular). Esto es debido a que, en términos de usabilidad, es más intuitivo tener los gráficos de una serie de datos agrupadas en el mismo sitio.
- *Estructura de acceso a la información de ficheros*. Es necesaria para poder recoger la información de la hoja de cálculo y procesarla posteriormente.
-

4.6. Desarrollo de algoritmos no triviales

Algoritmo de la escala lineal y la escala logarítmica para el eje vertical

Previamente, es necesario conocer los valores con los que estamos tratando:

- *Valores de la serie*. Son aquellos valores de la serie de datos.
- *Valores en píxeles*. Es el tamaño utilizado para el trazado de elementos. Es necesario traducir los valores de la serie a estos para su trazado.

A continuación, se muestra un ejemplo para identificar ambos valores:

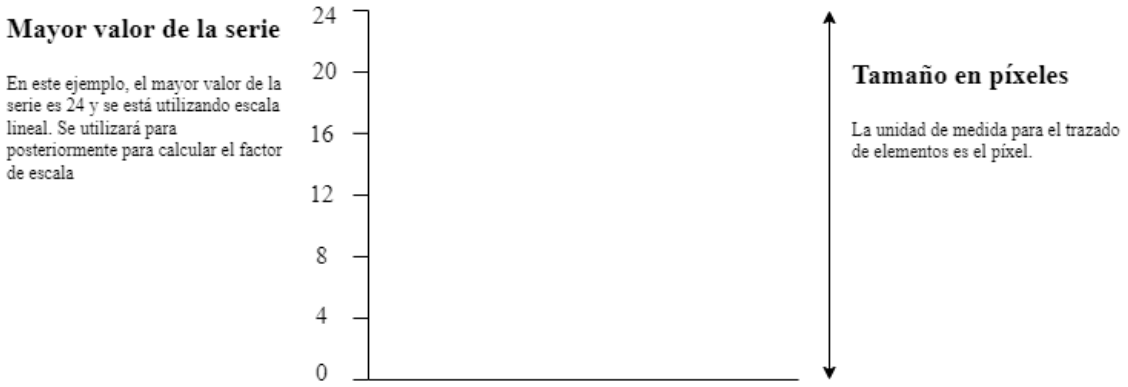

Fig X. Ejemplo para identificar los distintos valores para el cálculo de la escala.

Las escalas desarrolladas son las siguientes:

- ***Escala lineal***. Esta escala se caracteriza por el hecho de que todos sus valores tienen el mismo intervalo de separación. Para el cálculo de esta, son necesarias las siguientes etapas:
    - *Cálculo del máximo valor en píxeles de la gráfica*. Si traducimos directamente el mayor valor de la serie a píxeles y este asumimos que es el mayor valor de la gráfica, el mayor valor de la serie siempre se encontrará tocando el techo. Para una mejor visualización, utilizamos un máximo valor para la gráfica que se encuentre ligeramente por encima, utilizando un redondeo hacia arriba. Además, forzamos a que sea múltiplo del número de marcas para que los valores referencia sean enteros. Para ello, si divide entre el número de marcas previamente al redondeo y, después de este, se multiplica por el anterior.
    - *Cálculo del factor de escala lineal*. Es el factor que utilizaremos para traducir valores de la gráfica a píxeles. En este caso, el mínimo valor en píxeles será 0 y el mínimo valor escogido también será 0. El factor de escala sería el siguiente:

$$Factor\ de\ escala\ lineal = \frac{(Máximo\ valor\ en\ píxeles) - (Mínimo\ valor\ en\ píxeles)}{(Máximo\ valor\ de\ la\ serie) - (Mínimo\ valor\ escogido)}$$

- *Traducción de los valores de la serie a píxeles*. Este proceso se realiza en correspondencia con la siguiente ecuación ("*Pxs*" hace referencia a "*Valor en píxeles*", mientras que "*FL*" hace referencia a "*Factor de escala lineal*"):

$$Pxs = ((Mínimo\ valor\ en\ píxeles) + FL * (Valor\ de\ la\ serie) - (Mínimo\ valor\ escogido))$$

- ***Escala logarítmica***. En esta, la distancia entre cada valor no es la misma y, al contrario, se utiliza el logaritmo para calcularla. Los logaritmos utilizados son logaritmos neperianos. Son necesarias las siguientes etapas:
  - *Cálculo del máximo valor de la gráfica*. El cálculo de esta también hace utilizando el mayor valor de la serie y redondeándolo hacia arriba. También lo dividimos por el número de marcas antes del redondeo y lo multiplicamos después del redondeo, para que este sea mayor y haya cierta distancia con el mayor valor de la serie.
  - *Cálculo del factor de escala logarítmico*. Además, los logaritmos utilizados serán logaritmos neperianos. En este caso, el mínimo valor en píxeles será 1 y el mínimo valor escogido será 0. El factor de escala sería el siguiente:

$$Factor\ de\ escala\ logarítmica = \frac{\ln(Máximo\ valor\ en\ píxeles) - \ln(Mínimo\ valor\ en\ píxeles)}{(Máximo\ valor\ de\ la\ serie) - (Mínimo\ valor\ escogido)}$$

  - *Traducción de los valores de la serie a píxeles*. Este proceso se realiza en correspondencia con la siguiente ecuación ("*Pxs*" hace referencia a "*Valor en píxeles*", mientras que "*FLOG*" hace referencia a "*Factor de escala logarítmica*"):

$$Pxs = e^{(\ln(Mínimo\ valor\ en\ píxeles) + FLOG * (Valor\ de\ la\ serie) - (Mínimo\ valor\ escogido))}$$

Es importante anotar que no existe $\log(0)$, de modo que es necesario las oportunas comprobaciones para que no ocurra.

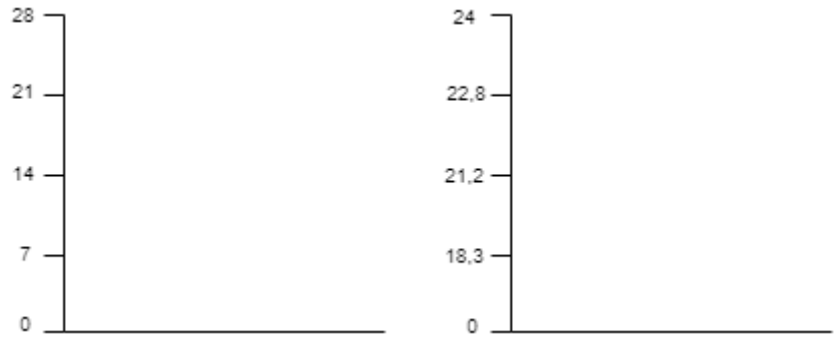

Fig X. Ejemplo de escala linear a la izquierda y de escala logarítmica a la derecha.

Algoritmo de optimización de elementos visualizados

El espacio utilizado para el mostrado de elementos gráficos es reducido y limitado, de forma que un algoritmo con el objetivo de perfeccionar esta funcionalidad era fundamental. El algoritmo tiene dos variantes en función de su uso:

- *Algoritmo para optimizar el espacio del gráfico de barras*. El algoritmo recoge aquellas modificaciones necesarias para optimizar el espacio de dicho gráfico. Consta de un proceso con tres partes diferenciadas:
    - **Primera etapa**. Se reducirá el espacio entre cada barra y el ancho de esta todo lo que sea necesario. En caso de que, al final de una iteración, se compruebe que es posible mostrar todos los elementos, el algoritmo salta a la siguiente etapa. De lo contrario, el algoritmo continúa.
    - **Segunda etapa**. Se reducirá el tamaño del texto utilizado para indicar los valores y las etiquetas. En cada iteración se comprueba si la optimización a finalizado o no.
    - **Tercera etapa**. Tras reducir el ancho de barra, el espacio entre estas y/o el tamaño del texto, es posible que los textos se solapen entre sí. Si este hecho ocurre, se muestran solo parte del texto, de forma que no se solape y pueda visualizarse correctamente.
- *Algoritmo para optimizar el espacio del gráfico de líneas*. Dicho algoritmo especifica aquellas modificaciones que tienen como objetivo optimizar el espacio en un gráfico de líneas. El proceso dispone de tres etapas:
    - **Primera etapa**. Progresivamente, se irá reduciendo el espacio entre cada punto todo lo que sea necesario. En caso de que no sea necesario continuar con la optimización, se concluye esta etapa y se procede a la siguiente.
    - **Segunda etapa**. Esta esta disminuye el tamaño de los textos pertenecientes a los valores y las etiquetas todo lo que sea necesario. Mientras sea necesario, el algoritmo reduce el texto en cada iteración.
    - **Tercera etapa**. La última etapa comprueba si los textos de los valores o de las etiquetas se solapan. De ser así, se mostrarán solo algunos de ellos.

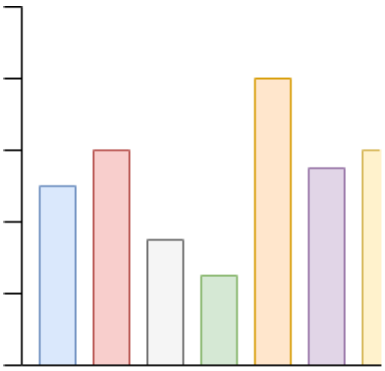

Fig X. Ejemplo de gráfico de barras donde no se ven todos los elementos.

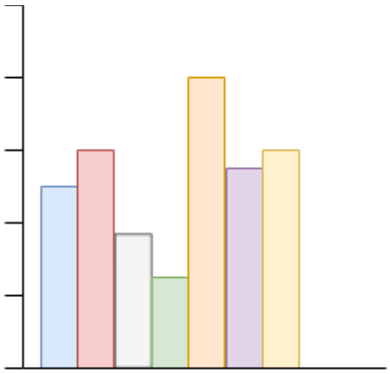

Fig X. Ejemplo de gráfico de barras donde sí se ven todos los elementos.

4.7. Códigos de colores utilizados

Los colores han sido elegidos en términos de usabilidad. A continuación, se muestran los códigos hexadecimales de los colores escogidos:

- Código #000000. Representa el color negro. Es utilizado, esencialmente, para contornos de figuras, ejes de coordenadas y texto.

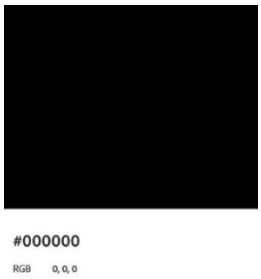

Fig X. Ejemplo de código de color #000000.

- Código #FFFFFF. Representa el color blanco. Es usado, fundamentalmente, para representar el fondo.

Fig X. Ejemplo de código de color #FFFFFF.

- Código #F0FFFF. Es utilizado para la cabecera de la página web, cuadros de texto y el efecto de cambio de color al colocar el puntero sobre las distintas pestañas.

Fig X. Ejemplo de código de color #F0FFFF.

- Código #F0F8FF. Es utilizado para el cuadro de opciones de la gráfica.

Fig X. Ejemplo de código de color #F0F8FF.

- Código #D5F9F9. Es utilizado para distintos cuadros de información y botones en la página web.

Fig X. Ejemplo de código de color #D5F9F9.

- Código #D4FFD8. Este color representa el cuerpo de la página web.

Fig X. Ejemplo de código de color #D4FFD8.

- Código #F5F5DC. También utilizado para cuadros dentro de la página web.

Fig X. Ejemplo de código de color #D5F9F9.

- Código #D6EDD8. Usado para el menú de selección de la gráfica.

Fig X. Ejemplo de código de color #D6EDD8.

- Código #EAB8B8. Este color es utilizado para aquellos botones que eliminan series de datos y gráficas.

Fig X. Ejemplo de código de color #EAB8B8.

- Código #E5E4E2. Usado para representar las líneas referencia en los gráficos de barras y de líneas.

Fig X. Ejemplo de código de color #E5E4E2.

- Código #4D4D4D. Utilizado para el trazado de la línea principal del gráfico de líneas.

Fig X. Ejemplo de código de color #4D4D4D.

Los colores de las secciones del gráfico de barras y del gráfico circular han sido representados con colores *HSL* ("*hue, saturation, lightness*"). Esto ha sido así debido a que, para representar las secciones, es deseable utilizar colores pastel. De esta forma podemos utilizar un coeficiente bajo de "*lightness*" para generar colores aleatorios pastel.

5. Implementación

5.1. Descripción de las clases

Clase 'CanvasApiApplication'

Esta clase encapsula todas las funcionalidades principales de la aplicación, tales como la creación o borrado de series de datos y gráficas. Sus atributos de instancia son los siguientes:

- Atributo *canvas*. Atributo utilizado para almacenar un elemento *canvas*. Este es un elemento HTML utilizado principalmente para el trazado de gráficos. Almacena un objeto de la clase *Element*.
- Atributo *ctx*. Almacena el contexto de renderizado 2D para el trazado de elementos en *canvas*. Es adquirido mediante el método *getContext('2d')* de este atributo. Almacena un objeto de la clase *CanvasRenderingContext2D*.
- Atributo *fileReader*. Atributo utilizado para almacenar un objeto de la clase *FileReader*. Este es utilizado para la lectura de ficheros.
- Atributo *data_series*. Vector utilizado para almacenar las series de datos creadas. Los elementos son objetos de la clase *DataSerie*.
- Atributo *data_serie_next_id*. Atributo utilizado para almacenar el identificador de la siguiente serie de datos una vez sea creada. El atributo es un entero.
- Atributos *bar_charts*, *line_charts* y *pie_charts*. Vectores utilizados para almacenar los gráficos de barras, de líneas y circulares, respectivamente, creados. Los elementos son objetos de las clases *BarChart*, *LineChart* y *PieChart*, respectivamente.
- Atributos *bar_charts_to_draw*, *line_charts_to_draw* y *pie_charts_to_draw*. Vector utilizado para almacenar los identificadores de los gráficos de barras que van a ser visualizados. Los elementos son números enteros.
- Atributos *bar_chart_next_id*, *line_chart_next_id* y *pie_chart_next_id*. Atributos utilizados para almacenar los identificadores de los siguientes gráficos de barras, líneas y circulares, respectivamente, una vez sean creados. Los atributos son enteros.

Los métodos, esencialmente, se pueden clasificar en los siguientes tipos:

- Métodos relacionados con la ocultación y la aparición de elementos. Estos se encargan de mostrar u ocultar elementos de la aplicación en función de las acciones del usuario. Ejemplos de estos métodos son *showHome()* o *showData()*.
- Métodos relacionados con la carga de ficheros. Estos tienen la función de la obtención de ficheros, ya sea seleccionándolos directamente o mediante un *drag and drop*. Ejemplos de estos métodos son *insertManager(file)* o *dropManager(event)*.
- Métodos relacionados con la creación o borrado de gráficos y series de datos. Estos desempeñan la función de insertar gráficos y series de datos, así como borrar las mismas, en función de las peticiones del usuario. Ejemplos de estos métodos son *submitChart()* o *removeDataSerie(id)*.
- Métodos relacionados con el trazado de gráficos. Estos desempeñan la función principal de la aplicación. Para cada gráfico se utiliza un *canvas* y un *ctx* (contexto) en particular.

El contexto permite llamar a aquellos métodos necesarios para el trazado (*moveTo(), lineTo(), fillRect(), arc(),* etc.) y a los parámetros de estilo para modificar atributos tales como la fuente, el tamaño de esta, el grosor de línea, etc.

Clase 'DataSerie'

Clase encargada de encapsular las funcionalidades de una serie de datos. Sus atributos de instancia pueden clasificarse en dos tipos:

- Atributos no estructurados. Este es el caso del atributo *unstructured_data*. Almacena la toda la información en un *String* para su posterior estructuramiento. Contiene la información según es generada por *fileReader*.
- Atributos estructurados. Son aquellos atributos que representan la información una vez estructurada. Estos son:
    - Atributos *structured_data_tags* y *structured_data_values*. La primera contiene aquellas etiquetas que conformaran el eje horizontal. Debido a que las series de datos suelen tener la propiedad de estar ligadas a periodos de tiempo, de forma que normalmente representarán años.
    - Atributos *variable_tags* y *variable_values*. Contiene, en primer lugar, el nombre de los criterios de selección utilizados y, en segundo lugar, el valor de estos últimos.
    - Atributo *number_of_variables*. Contiene el número de criterios de selección utilizados en la serie de datos.
- Título. Representa el rótulo de la serie de datos.

Los métodos de esta clase son esencialmente métodos para mostrar o modificar atributos. El método *structureData()* será explicado posteriormente.

Clase 'Chart'

Esta clase tiene encomendado representar una gráfica. Es la clase padre de *Barchart*, *LineChart* y *PieChart*. Posee dos tipos de atributos estáticos:

- Atributos que representan el tamaño del *canvas*. Estos son *WIDTH*, que representa el ancho de este, y *HEIGHT*, que representa la altura de este.
- Atributos de estilo de la fuente. Representan el mínimo tamaño de fuente permitido (*MIN_FONT*) y el máximo (*MAX_FONT*).

Los atributos de instancia son los siguientes:

- Atributo *id*. Representa el identificador del gráfico. Los vectores que almacenan estos identificadores los utilizan para saber que gráfico se visualiza en cada momento.
- Atributo *data_serie*. Representa la serie de datos asociada a dicha gráfica. Los identificadores de las series de datos son utilizados para agrupar los tres tipos de gráficas en función de este para su posterior visualización.

- Atributos de estilo. Estos son *letter_value_width* (representa la anchura del texto que hace referencia a los valores de la serie), *letter_tag_width* (indica la anchura del texto que hace referencia a las etiquetas del eje horizontal de la serie), *letter_height* (representa la altura del texto), *letter_font* (representa la fuente utilizada para el texto), *colors* (vector que almacena los colores de cada una de las barras o secciones de los gráficos), *sectionColor* (indica el color de la barra o sección trazada en un instante determinado), *previousSectionColor* (representa el color de la barra o sección previa a la trazada en un instante determinado), *strokeStyle* (indica el color utilizado para la línea), *lineWidth* (indica el grosor de línea utilizado), *lineCap* (representa el revestimiento de línea utilizado), *shadows* (indica si las sombras están activadas), *transparency* (representa el nivel de trasparencia actual), *gradient* (representa el nivel de gradiente actual), *gradientColor* (indica el color del gradiente utizado) y *threedEffect* (indica si está activado el efecto de tres dimensiones).

Sus métodos son, en su mayoría, métodos para modificar o mostrar atributos. El método *insertChartData()* será explicado posteriormente.

Clase 'BarChart'

Clase que hereda de la clase *Chart* para representar un gráfico de barras. Posee numerosos atributos estáticos para representar parámetros de estilo:

- Atributos *PADDING_LEFT*, *PADDING_RIGHT*, *PADDING_TOP* y *PADDING_BOTTOM*. Representa la distancia entre un lado del «canvas» y el lado correspondiente donde inicia la gráfica.

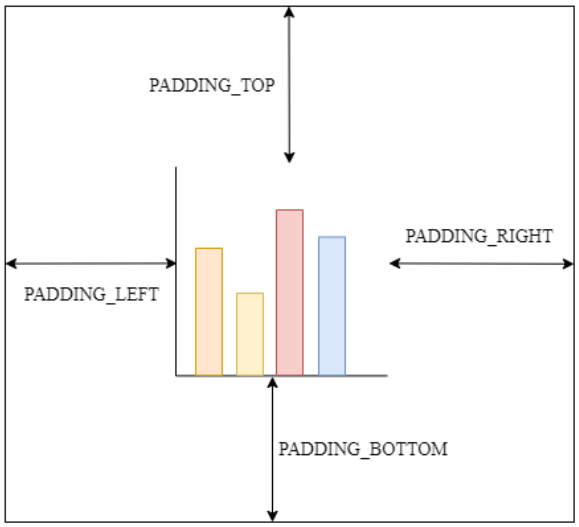

Fig X. Ejemplo representativo de cada uno de los distintos tipos de PADDING para *BarChart*.

- Atributo *LOG_MARGIN*. Representa, para la escala logarítmica, el logaritmo del menor valor en píxeles alcanzado.
- Atributo *MAX_NUMBER_OF_DIGITS*. Indica la mayor longitud que puede tener el texto que aparece en la gráfica en términos de caracteres de *String*. El punto decimal también cuenta dentro de este cálculo.

- Atributo *STANDARDIZE_SCALE_FACTOR*. Es utilizado en la escala linear para que exista una cierta distancia entre el máximo valor de la gráfica y el máximo valor de la serie. De lo contrario la sección que representa el mayor valor de la serie estaría siempre pegado al techo.
- Atributos *MAX_LOG_VALUE* y *MIN_LOG_VALUE*. Son utilizado en el cálculo del número de marcas del eje vertical. El primero representa el logaritmo en base 10 a partir del cual se trazan el máximo número de marcas, mientras que el segundo representa el logaritmo en base 10 para el cual se trazan el mínimo número de marcas.
- Atributo *BARS_MARGIN*. Indica la separación entre el eje vertical y la primera barra trazada.
- Atributos *MAX_NUMBER_OF_VERTICAL_LINES* y *MIN_NUMBER_OF_VERTICAL_LINES*. Representan el máximo y el mínimo posible de marcas trazadas en el eje vertical.
- Atributo *VERTICAL_LINES_WIDTH*. Representa el ancho de cada una de las líneas verticales.
- Atributos *LETTER_VALUE_WIDTH*, *LETTER_TAG_WIDTH* y *LETTER_HEIGHT*. Los dos primeros representan el ancho inicial estimado, en píxeles, del texto de los valores y de las etiquetas, respectivamente. El tercero representa la altura inicial, en píxeles, de cualquier texto del gráfico.
- Atributos *LETTERS_MARGIN_LEFT*, *LETTERS_MARGIN_TOP* y *LETTERS_MARGIN_BOTTOM*. El primero representa el margen izquierdo inicial con respecto al eje vertical que posee el texto de valores y etiquetas. El segundo representa el margen que posee el texto de etiquetas con respecto al eje horizontal. El tercero indica el margen que posee el texto de valores con respecto a la barra a la que acompaña.
- Atributos *THREE_D_X_DISPLACEMENT* y *THREE_D_Y_DISPLACEMENT*. Indican el desplazamiento horizontal y vertical que debe hacer texto una vez se active el efecto de tres dimensiones.

Sus atributos de instancia, además de los heredados de *Chart*, son los siguientes:

- Atributo *bar_width*. Representa el ancho de cada barra.
- Atributo *scale*. Representa el tipo de escala utilizada actual. Puede ser linear o logarítmica.
- Atributo *space_between_bars*. Indica el espacio que existe entre cada barra.
- Atributo *text_appearance*. Representa el número de apariciones del texto de valores y de etiquetas.
- Atributo *visible_values*. Representa si el texto de valores será visible o no.
- Atributo *number_of_vertical_lines*. Indica el número de marcas horizontales que tendrá el eje vertical.
- Atributo *max_value_linear_graph*. Expresa el máximo valor de la gráfica en la escala lineal.

- Atributo *max_value_logarithmic_graph*. Representa el máximo valor de la gráfica en la escala logarítmica.
- Atributo *linear_scale_factor*. Indica el factor de escala lineal.
- Atributo *logarithmic_scale_factor*. Expresa el factor de escala logarítmica.

Los métodos de esta clase son, en su mayoría, métodos para obtener y modificar atributos. Por otro lado, el método *checkWidthLimit(variable_quantities)* es utilizado para indicar si los elementos de dentro del gráfico han alcanzado el máximo ancho disponible o no.

Clase 'LineChart'

Clase hija de la clase *Chart*, utilizada para representar un gráfico de líneas. La amplia mayoría de sus atributos y métodos son análogos a los existentes en *BarChart*.

Clase 'PieChart'

Esta clase es utilizada para representar un gráfico circular. Hereda de la clase *Chart*. Comparte algunos atributos estáticos con *BarChart* y *LineChart* aunque, sin embargo, se implementan otros nuevos. Estos son:

- Atributos *X_CENTER* y *Y_CENTER*. Representan las coordenadas del centro del gráfico circular.
- Atributos *BIG_RADIO*, *RADIO*, *SMALL_RADIO* y *SMALL_GRADIENT_RADIO*. El primero indica el radio que va a tener el circulo imagino que conforma las etiquetas. El segundo indica el radio del gráfico circular. El tercero representa el radio que va a tener el circulo imagino que constituyen los valores. El último indica el radio máximo del gradiente.
- Atributo *MIN_ANGLE_TEXT*. Indica el ángulo a partir del cual no se traza el texto debido a su poca visibilidad.
- Atributos *THREED_DEPTH*, *THREED_REFERENCE_X_POINT*, *THREED_ REFERENCE_RADIO* y *THREED_SHADOW_CORRECTOR*. El primero indica la profundidad que va a tener el falso cilindro del efecto de tres dimensiones. El segundo indica la abscisa del punto de referencia para el trazado de la curva utilizada para crear la segunda base del falso cilindro del efecto de tres dimensiones. El tercero indica el radio del círculo que conforma la segunda base del falso cilindro del efecto de tres dimensiones. El cuarto y último expresa el desplazamiento en el eje horizontal que se debe hacer para situar la sombra una vez se active el efecto de tres dimensiones.

Además de los atributos de instancia heredados contiene un atributo de instancia (*gradientActivated*), debido a que el método *createRadialGradient()* crea un siempre un gradiente una vez creado, de forma que es necesario indicarle cuando mostrarlo y cuando no.

Los métodos son, principalmente, métodos para obtener y modificar atributos.

5.2. Descripción de los métodos no triviales

Algunos métodos tienen una complejidad más alta que otros. Esta subsección explicará con detalle el funcionamiento de cada uno de estos métodos.

Métodos de la clase *APICanvasApplication*

La amplia mayoría de los métodos no triviales se encuentran en esta clase.

Método 'submitDataSerie(event)'

Este método es el encargado de crear una serie de datos. Realiza las siguientes acciones:

- Obtiene el título especificado por el usuario y comprueba si dicho título ya existe. En caso de que exista dicho título, aparece un mensaje de error indicando que dicho título ya existe y finaliza la ejecución. En caso de que no exista, se continúa con la ejecución.
- A continuación, se obtiene el fichero a través del lector de ficheros. En caso de que sea el primer fichero, utilizamos el método *insertAdjacentHTML()* para añadir una una estructura * * que contendrá los cuadros de información de los ficheros.
- Creamos un nuevo objeto DataSerie, le asignamos su id, la información no estructurada y el título y llamamos al método *structureData()* para que estructure la información. Añadimos el objeto al vector de series de datos.
- Creamos en HTML la estructura del cuadro de información del fichero y la añadimos con *insertAdjacentHTML()*. Finalmente, en el menú de selección de la gráfica, añadimos la opción de elegir el fichero creado.

Método 'removeDataSerie(id)'

Método que se encarga de eliminar una serie de datos. Realiza las acciones:

- Obtenemos el identificador de la serie de datos a partir del parámetro del método y extraemos la serie de datos. Comprobamos si la serie de datos tiene gráficas asociadas. En caso de que tenga gráficas asociadas, aparece un mensaje de error que indica que no se pueden eliminar series de datos con gráficas asociadas y finalizamos la ejecución. En caso contrario, se continúa con la ejecución.
- Buscamos la serie de datos correspondiente, la eliminamos del vector de series de datos, eliminamos el cuadro de información de esta y eliminamos su opción correspondiente en el menú de selección de la gráfica.

Método ‘submitChart()’

El método se encarga de crear una nueva gráfica. Para ello, realiza lo siguiente:

- Obtenemos el título de la serie de datos relacionada y el tipo de gráfica seleccionada. En caso de que no se seleccione alguna de las dos opciones, aparecerá un mensaje de error diciendo que hace falta seleccionarla. En caso contrario, continúa la ejecución.
- Buscamos y obtenemos el objeto la serie de datos asociada a la gráfica. Comprobamos si ya existe la gráfica examinando si existe en la selección de opciones del cuadro correspondiente. Si ya existe la gráfica, mostramos un mensaje de error. En caso contrario, continúa la ejecución.
- En función del tipo de gráfica seleccionada, creamos un *BarChart*, un *LineChart* o un *PieChart*, le asignamos su identificador y su serie de datos y llamamos al método de asignación de los colores para las secciones en caso del *BarChart* y el *PieChart*. Si es la primera gráfica relacionada con esa serie de datos, la insertamos en el vector de gráficas a trazar.
- Llamamos al método para dibujar las gráficas.

Método ‘removeChart(id)’

El método es el encargado de eliminar una gráfica. Para ello, realiza las siguientes acciones:

- Obtenemos el identificador de la gráfica y su tipo.
- Eliminamos dicha gráfica de su vector y vector de gráficas a trazar correspondiente.
- En caso de que existan más gráficas relacionadas con la serie de datos, se elimina únicamente la opción del menú para cambiar de gráfica. En caso de que solo exista dicha gráfica, borramos todo el cuadro de información.
- Llamamos al método de trazado de gráficas.

Método ‘changeChartVisualized(id, value)’

Este método es el encargado de seleccionar cuáles son las gráficas que deben ser visualizadas y cuáles no, en función de las indicaciones del usuario. Para ello, este realiza lo siguiente:

- Obtenemos el identificador de la serie de datos relacionada con la gráfica. También obtenemos los vectores donde se almacenan las gráficas seleccionadas y las gráficas a dibujar y los vectores de las otras dos gráficas que no van a ser visualizadas. También obtenemos todos los elementos HTML de las tres gráficas, diferenciando entre la gráfica que se va a dibujar y las que no, como antes.
- Insertamos la gráfica que va a ser dibujada en su correspondiente vector de gráficas a trazar y eliminamos, si existen, las otras gráficas de sus respectivos vectores de gráficas a dibujar.
- Ocultamos aquellos elementos HTML de las gráficas que no van a ser dibujadas y mostramos los elementos HTML de la gráfica que sí va a ser dibujada.
- Finalmente, llamamos al método que se encarga de trazar las gráficas.

Método ‘drawBarChart(id)’

Método encargado de realizar el trazado de todas las gráficas de barras. Para ello, realiza lo siguiente:

- Obtenemos la gráfica de barras que va a ser trazada, su serie de datos asociada y el *canvas* donde se va a dibujar. Obtenemos el contexto a partir del método getContext(‘2d’) y deja despejado el rectángulo que conforma el *canvas*.
- Utilizamos el contexto para asignar los parámetros de estilo iniciales, tales como el grosor de línea, el alfa o nivel de transparencia, el color de línea o la fuente.
- A continuación, se proceden a dibujar las marcas de referencia horizontales. En función de si la escala es lineal o logarítmica existirá un máximo valor para la gráfica u otro. Utilizamos el contexto para trazar dos tipos de líneas: las marcas horizontales de color negro que aparecen a la izquierda del eje vertical y las marcas horizontales de color grisáceo que aparecen en la gráfica. En caso de que los valores de referencia no superen el número máximo de caracteres serán mostrados. Esto también se aplica a las marcas horizontales de color negro.
- Utilizamos el contexto para trazar los ejes vertical y horizontal, de color negro.
- Realizamos las operaciones oportunas relacionadas con la optimización de la aparición de elementos del gráfico. En primer lugar, comprobamos si el ancho máximo permitido ha sido superado. Si no ha sido superado, continuamos con la ejecución. En caso contrario, procedemos con la optimización:
    - Reducimos la distancia entre cada barra y comprobamos en cada iteración si es necesario continuar con esta optimización. Una vez finalice pasamos a la siguiente etapa.
    - Reducimos el tamaño de la letra de los valores y etiquetas y calculamos el nuevo ancho de barra en función de ello. Comprobamos, también, en cada iteración si es necesaria más optimización y en caso contrario continuamos.
    - Calculamos el número de apariciones del texto de valores y etiquetas necesarias en función de si estas se solapan o no entre sí.
- Procedemos al trazado de los rectángulos que conformarán las barras:
    - Calculamos las abscisas y ordenadas del inicio y del final de rectángulo en función de si la escala es linear o logarítmica. En caso de que el efecto de tres dimensiones esté activado, deberemos trazar cinco líneas y rellenar el espacio con el mismo color para dar la sensación de dicho efecto. También deberemos situar las sombras para el caso en el que el efecto esté activado y para el caso en el que no.
    - Creamos un gradiente para que luego pueda ser modificado por el usuario, aunque inicialmente aparecerá solo el color original.
    - Trazamos y rellenamos los rectángulos en función de si la escala es linear o logarítmica.
- Finalmente, situamos el texto de valores y etiquetas (en caso de que se cumplan las condiciones para que sean visibles) en función de si la escala es linear o logarítmica y en función de si está activado el efecto de tres dimensiones.

Método ‘drawLineChart(id)’

Método encargado de realizar el trazado de las gráficas de líneas. Realiza las siguientes acciones:

- Obtenemos el identificador del gráfico de líneas y su serie de datos asociada, así como el *canvas* donde va a ser dibujada. También obtenemos el contexto a partir del método *getContext(‘2d’)* y despejamos el rectángulo que conformará el *canvas*.
- A través del contexto damos valores iniciales a los parámetros de estilo, tales como el color de rellenado, el color del trazado, el grosor de línea, la fuente o el alfa o nivel de transparencia.
- Procedemos al trazado de las marcas de referencia horizontales. Se trazarán marcas de referencia horizontales de color negro a la izquierda del eje horizontal para acompañar a los valores de referencia y marcas de referencia horizontales de color grisáceo para que sea más fácil identificar dichos valores en la gráfica. También aparecerán valores de referencia (si su número de caracteres es menor que el máximo permitido, al igual que las marcas de referencia de color negro) al lado de las marcas de referencia de color negro.
- Trazamos los ejes de coordenadas (de color negro) vertical y horizontal.
- Realizamos las operaciones oportunas a la optimización de elementos:
    - Reducimos la distancia entre cada punto en cada iteración y comprobamos si es necesario reducirlo más. Una vez finalizado, continuamos con el siguiente paso.
    - Reducimos el tamaño de la letra de los valores y etiquetas todo lo que sea posible (hasta un mínimo especificado para poder ser visualizado correctamente).
    - Evaluamos el número de apariciones necesarias del texto de valores y etiquetas para que no se solapen entre sí.
- Realizamos el trazado de la línea principal, teniendo en cuenta si la escala es linear u horizontal. Para esta línea se ha utilizado un color gris oscuro para que no sea el mismo de los ejes de coordenadas y se pueda ver por encima de las marcas de referencia. Es también necesario realizar las oportunas operaciones de sombras si estas están activadas.
- Finalmente, situamos el texto de valores y etiquetas (si se cumplen las condiciones para su visibilidad) en función de la escala.

Método ‘drawPieChart(id)’

Método encargado de realizar el trazado de los gráficos circulares. Realiza lo siguiente:

- Obtenemos el identificador de la gráfica circular, su serie de datos asociada y el *canvas* donde va a ser dibujada. También se obtiene el contexto con el método *getContext(‘2d’)* y despejamos el rectángulo que conforma el *canvas*.
- Utilizamos el contexto para establecer los parámetros iniciales de estilo, tales como el grosor de línea, la fuente, el color de trazado o el alfa o nivel de transparencia.
- Sumamos todos los valores de la serie para poder calcular, posteriormente, las distintas fracciones de esta.
- Creamos un círculo de color blanco que situará justo detrás del gráfico circular. Esto lo hacemos con el objetivo de que la sombra se vea como un todo, ya que si le ponemos una sombra a cada sección esta se verá fragmentada. Este círculo tendrá como centro el mismo que el gráfico o se situará un poco más a la derecha en función de si está activado el efecto de tres dimensiones o no. Es necesario asignar las operaciones de sombras oportunas en función de si estas están activadas o no. En caso de que esté activado el efecto de tres dimensiones, realizamos dos líneas y un arco con el objetivo de dar sensación de profundidad.
- Realizamos el trazado de las secciones del gráfico circular. Su ángulo dependerá de la fracción del total del valor en particular. Asignamos un gradiente a cada sección del gráfico para poder ser visualizado en el gráfico circular entero.
- Finalmente, situamos los textos de valores y textos (si estos cumplen las condiciones para ser visibles como, por ejemplo, que el ángulo sea mayor que un mínimo establecido).

Métodos de la clase *DataSerie*

A continuación, se presentan aquellos métodos de la clase DataSerie cuya estructura no es trivial.

Método ‘structureData()’

Método que realiza labores relacionadas con la transformación de la información no estructurada a estructurada. Realiza lo siguiente:

- Se transforma un *String* que contiene toda la información no procesada a un vector de *String*, utilizando como elemento de separación el salto de línea. Eliminamos la primera posición del vector ya que únicamente contiene los títulos.
- Establecemos el número de variables de la serie de datos. Las distintas casillas del fichero, al traducirlas al vector, están separadas por punto y coma, de modo que este valor se puede calcular creando otro subvector, cuyo elemento de separación sea el punto y coma, y calculando la mitad menos uno de una casilla de este último. Debido a

que dos casillas pertenecen al periodo y al valor, el resto de las casillas pertenecen a las variables. No obstante, cada dos casillas es una variable, ya que una pertenece a la "etiqueta" de la variable y otra al "valor" de esta. De ahí que sea la mitad menos uno de la longitud.

- Asignamos los valores de las etiquetas y los valores de las variables. Las etiquetas se encuentran en los elementos cuyo índice es par (ya que el primer índice es el cero) y los valores se encuentran en los elementos cuyo índice es impar. Esto se debe hacer sin contar los dos últimos elementos, ya que pertenecen al periodo y al valor.

- Establecemos, en cada iteración, el periodo correspondiente y se lo asignamos a su respectivo atributo (para ello antes obtenemos toda la fila de un periodo concreto al separarlo por punto y coma). También obtenemos el valor de dicho periodo, eliminamos los separadores de millares y transformamos la coma decimal a punto decimal. Tras ello, asignamos el valor a su respectiva variable.

- Finalmente, asignamos los vectores auxiliares de etiquetas y valores estructurados a sus respectivos atributos, pero en orden contrario, ya que la serie de datos se encontraba en dicho orden. Por último, llamamos al método para normalizar el tipo de dato de los valores.

Método 'normalizeValues()'

Método que se encarga de que todos los valores de la serie de datos tengan el mismo tipo de dato (número entero o real). Sigue los siguientes pasos:

- Comprobamos si los valores son enteros o no. Para ello, utilizamos la siguiente expresión regular: /^-?\d+$/. Esta expresión indica que el número debe comenzar como se indica (^) y debe finalizar como se indica ($), puede o no ser negativo (-?) y que debe puede tener uno o más dígitos (\d+).

- En caso de que todos los números sean enteros, se almacenaran en su respectivo atributo como enteros. En caso de que haya uno o más números reales, se almacenarán como reales.

Métodos de la clase *Chart*

Por último, se presentan aquellos métodos no triviales relacionados con la clase *Chart*.

Método 'insertChartData()'

Método encargado de insertar los elementos HTML relacionados con la creación de un gráfico. Realiza lo comentado a continuación:

- Comprobamos si ya existen más gráficos relacionados con la serie de datos del gráfico. En caso de que existan más gráficos relacionados, insertamos una nueva opción al menú para cambiar de gráfica y añadimos otros elementos HTML (como el botón de eliminado, el botón de las opciones o el *canvas*) pero estos permanecerán ocultos, ya que esta gráfica no es la que está siendo visualizada en este instante. En caso de que no haya gráficas relacionadas, deberemos insertar, además de los elementos HTML mencionados anteriormente, todos los elementos relacionados con el cuadro de información de la gráfica, ya que es la primera vez que se introduce (estos no deben permanecer ocultos).

- Tras lo anterior, se insertarán todos los elementos HTML relacionadas con las opciones (los tipos opciones dependerán del tipo de gráfico).

- Finalmente, establecemos el ancho y alto del *canvas* y devolvemos *true* en caso de que sea la primera gráfica creada asociada con dicha serie de datos o *false* si ya existe alguna.

5.3. Pruebas del software e informes de ejecución

6. Conclusiones y vías futuras

Bibliografía

[1] bucephalus.org. (s. f.). *The HTML5 Canvas Handbook*. https://bucephalus.org/text/CanvasHandbook/CanvasHandbook.html

[2] *API Canvas - Referencia de la API Web | MDN*. https://developer.mozilla.org/es/docs/Web/API/Canvas_API

[3] *HTML Canvas Deep Dive*. (s. f.). https://joshondesign.com/p/books/canvasdeepdive/title.html

[4] Hughes, J., Dam, V. A., McGuire, M., Sklar, D., Foley, J., Feiner, S. & Akeley, K. (2013). *Computer Graphics: Principles and Practice* (3rd Revised ed.). Addison-Wesley Professional.

[5] Schwaber, K. (2004). *Agile Project Management with Scrum* (1.). Microsoft Press.

[6] *Adobe Color*. (s. f.). https://color.adobe.com