# Complemento Web para la gestión de bibliografías en Google Docs

Andrea Martín Arias


Trabajo de fin de grado del Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Director: Enrique Martín Martín
Codirector: Adrián Riesco Rodríguez

# Índice

# Resumen

Es común que durante la elaboración de memorias y otros documentos de carácter técnico, se quiera crear una sección bibliográfica con el fin de referenciar a libros, tesis o artículos que se hayan mencionado en el transcurso del documento.Sin embargo, en ocasiones la elaboración de una bibliografía se complica pues su estructura es más compleja de lo que parece y requiere del conocimiento de ciertos aspectos importantes.

Por esta razón, este proyecto tratará de solucionar dicho problema facilitando ese complejo proceso de elaboración. Para ello, nos apoyaremos en Google Docs para elaborar documentos de cualquier tipo en los que añadiremos ciertas funcionalidades adicionales a través de un lenguaje de programación específico: Google Apps Script. De esta forma, el programa simulará la semántica y sintaxis de LaTeX para crear de manera automática y en cuestión de segundos una bibliografía con la información de todos aquellos documentos que hayan sido referenciados durante el documento. Para proporcionar la información de los diferentes documentos que se podrán referenciar utilizaremos una herramienta especial denominada BibTeX, que nos permitirá dar formato a listas de referencias.

Es por ello, que LaTeX y BibTeX serán dos de los grandes pilares a lo largo de esta memoria, pues este proyecto trata de reflejar la esencia de los mismos a través de Google Apps Script: un lenguaje de programación similar a Javascript que otorga la posibilidad de mecanizar tareas en servicios de Google creando aplicaciones web amigables con el usuario.

Durante la interacción con Google Docs nos sumergiremos en el amplio mundo de los *Complementos,* que nos facilitarán y ayudarán en la elaboración de documentos personales a través de funcionalidades adicionales muy diversas.

Desde un comienzo, este trabajo se ha ido subiendo a una plataforma de gestión de versiones ampliamente conocida en el mundo informático: GitHub. De esta manera, se ha podido disponer de un mayor control de las distintas versiones que se han ido produciendo a lo largo de las semanas, disponiendo en todo momento de copias de seguridad para cualquier imprevisto. Para acceder ella, se puede utilizar el siguiente enlace https://github.com/andreamartin96/TFG

**Palabras clave:**
*Google Apps Script, Google Docs, Addons, LaTeX, BibTeX, Bibliografía* y *Referencias.*

# Abstract

When writing a report, or any kind of technical document, it is common to include a reference section in order to quote every source used in the course of the document. However, these bibliographic sections can turn out to be a burden, as their structure is actually more complex than it seems and a prior knowledge of certain aspects is required.

Thus, our project will attempt to solve this problem, transforming the process of creating a reference section into a simple task. To achieve our goal, we will need Google Docs to help us create documents in which we will add some extra functionality via a specific programming language: Google Apps Script. Our program will emulate the LaTeX syntax and semantics in order to automatically generate a bibliographic section within a few seconds, when given a document with reference marks in it. For this purpose, we will use a tool called BibTeX, which will allow us to provide information of the various documents that can be referenced, and to format reference lists as we intend.

The aforementioned tools, both LaTeX and BibTeX, will serve as the fundamental pillars for this project, as we try to reflect their essence through the Google Apps Script language. This language resembles Javascript, and it provides the tools needed to manage and automate various Google-services tasks, and to develop user-friendly web applications.

As we get to know Google Docs, we will dive into the world of Google Add-ons, which will help us elaborate documents and interact with them, providing us with a broader spectrum of additional functionality.

Right from the start, this project has been managed through a free, widely known version control system: GitHub. Thanks to this service we have been able to possess a greater control over every version we would push as we developed our Add-on, which is a key aspect when it comes to keeping a backup version in case anything goes wrong. Our open-source project is available in the following GitHub repository: https://github.com/andreamartin96/TFG

**Keywords:**
*Google Apps Script, Google Docs, Add-ons, LaTeX, BibTeX, Bibliography and References.*

# 1. Introducción

Durante la elaboración de documentos, es frecuente que exista una sección de referencias en la que se incluye un breve resumen de todos aquellos documentos (libros, artículos, tesis…) que se hayan nombrado en alguna parte del escrito indicando toda la información de carácter importante de los mismos, como el autor o el título. Sin embargo, la realización de las bibliografías no siempre resulta sencilla, pues a medida que crece el número de páginas que conforman el documento, la complejidad en la construcción de esta sección también lo hace.

Es por ello, que este proyecto trata de simplificar su elaboración, reduciéndolo a un simple clic. Así, el usuario no tendrá que focalizar sus esfuerzos en confeccionar la bibliografía, permitiéndole centrarse en otros aspectos más importantes del documento.

Uno de los principios que constituyen este proyecto es un lenguaje de programación basado en Javascript que toma las bases del mismo para agregar nuevas clases que permitirán manipular los documentos de forma sencilla y eficaz: *Google Apps Script*. [1] No obstante, este trabajo se basará especialmente en la sintaxis que conforma a LaTeX que, junto a BibTeX, constituirán dos de los grandes pilares.

Para poder utilizar Google Apps Script, nos centraremos en el editor de documentos que el propio Google proporciona: *Google Docs*. De esta manera, tendremos a nuestra disposición un amplio mundo de posibilidades en lo que respecta a la edición de documentos, incluso la posibilidad de acceder con unas simples líneas de código a Google Drive, un banco de archivos donde el usuario puede almacenar hasta casi *15 GB* en documentos de cualquier tipo desde un primer momento.

Utilizando Google Apps Script, añadiremos nuevas funcionalidades que nos permitirán realizar tareas con un coste de tiempo mínimo, construyendo así un nuevo “Complemento de Google”. Actualmente, existen una larga lista de complementos ya creados que están a disposición de cualquier usuario.

Los complementos son creados con el fin de proporcionar una tarea que el editor de Google Docs no aporta en un inicio. De esta forma, a través de una interfaz de usuario amigable que nuestro complemento facilita, el usuario podrá seleccionar uno de entre todos sus documentos almacenados en Google Drive, siempre y cuando se trate de un fichero BibTeX. Así, con un simple clic de ratón, todas las referencias existentes en el documento serán identificadas por la aplicación para juntarlas y reconstruirlas en una nueva sección de referencias que se creará en el lugar indicado.

## 1.1. Objetivos

El principal objetivo de este proyecto es facilitar la creación de bibliografías en documentos de Google Docs. Para ello será necesario adentrarse en el mundo de LaTeX y BibTeX aplicando los conocimiento aprendidos durante el grado de HTML5 y Javascript para familiarizarse con un nuevo lenguaje: Google App Script. Podemos resumir los objetivos secundarios en los siguiente puntos:

- Aprender y manejar Google App Script para crear un complemento que lleve a cabo las funcionalidades deseadas.
- Conocer la sintaxis de LaTeX profundizando en su mecanismo y así simular su funcionamiento.
- Familiarizarse con BibTeX para identificar los distintos tipos de entradas y estilos que soporta.

## 1.2. Plan de trabajo

En este punto, se llevará a cabo una recopilación de todas las pautas que se propusieron durante la planificación de este proyecto de Fin de Grado detallando las tareas que se llevaron a cabo desde el comienzo hasta el final de desarrollo.

En un primer momento, el proyecto estaba pensado para dos personas, por ello, con motivo de lograr un correcto reparto de tareas, se convocó una primera reunión donde se organizaron las actividades que se debían realizar. Durante la reunión también se llevó a cabo un análisis en profundidad del proyecto para extraer aquellos puntos que presentaban mayor importancia:

- Conocer Google Apps Script.
- Aprender las funcionalidades y objetivos de LaTeX para comprender su semántica y sintaxis.
- Profundizar en las bases de BibTeX para descubrir cómo funcionan los archivos de este tipo.

Una vez conocidas las tareas que se debían abarcar en un inicio, propusimos un reparto de actividades equitativo para cada integrante del grupo. Con este reparto, cada integrante cubría ciertas tareas en lugar de abarcar todas las actividades reduciendo el tiempo de este proceso preliminar de investigación. Una vez completadas las tareas asignadas, se procedería a una puesta en común entre los integrantes del grupo explicando los aspectos importantes que se habían extraído de la investigación.

De esta manera, las primeras semanas se centraron en adentrarse en el nuevo lenguaje de programación desconocido hasta el momento. Para ello, se llevaron a cabo varios tutoriales que la propia web de Google Apps Script proporciona en los

que te enseñan de forma sencilla cómo crear paso a paso nuevos complementos (*Addons*) para tus documentos tal y como se muestra en la figura 1. Para la futura elaboración del proyecto se tomó como base uno de los tutoriales que se realizó, un simple traductor de texto que convertía el texto que habías seleccionado a la lengua escogida [2].

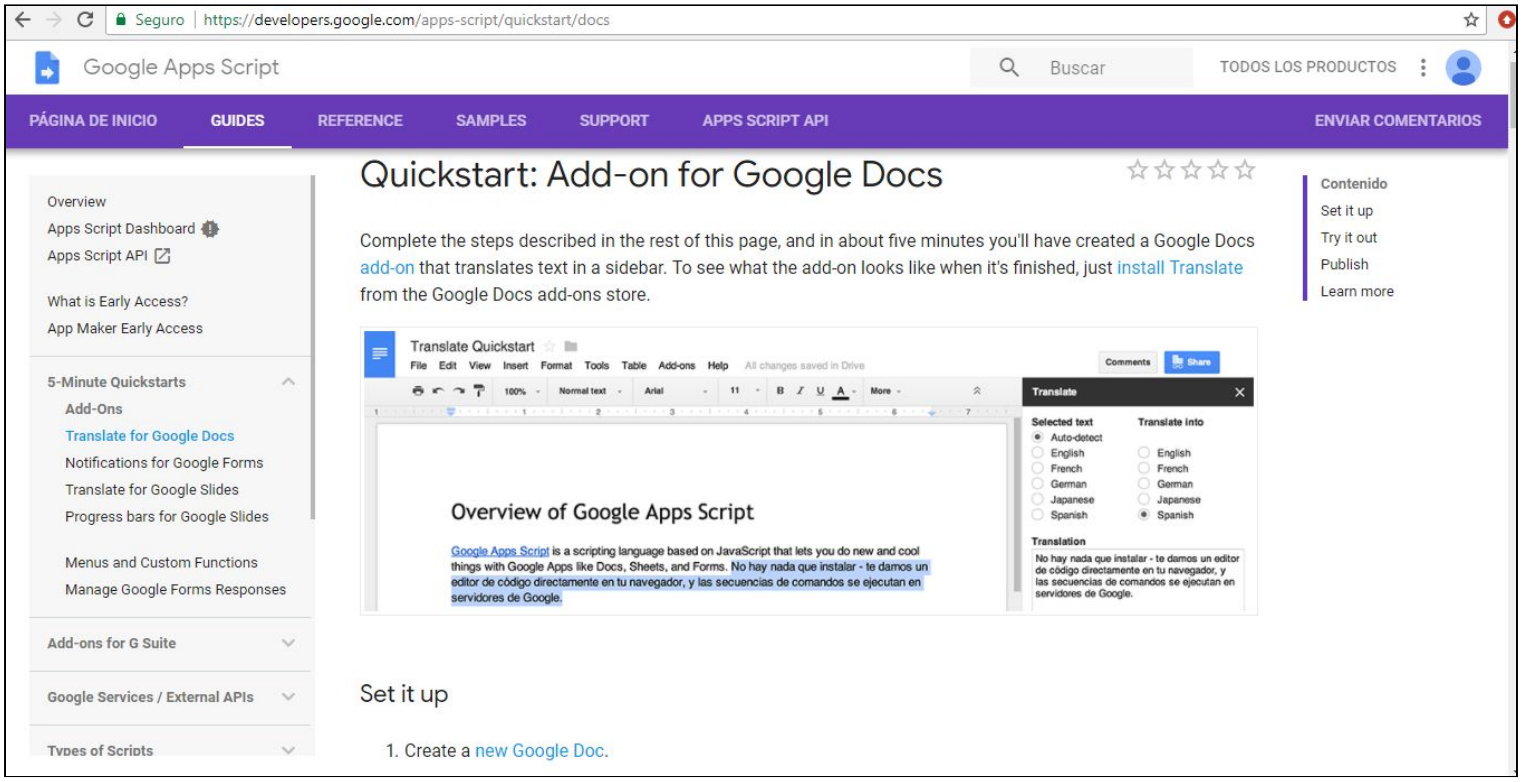



***Figura 1**: Página web oficial de Google Apps Script donde enseñan tutoriales sencillos sobre sus Addons.*

Estos tutoriales te enseñan cómo con unas pocas líneas de código es posible crear nuevos complementos que añaden funcionalidades extra a tus Documentos de Google. Sin embargo, esto es solo el principio pues Google Apps Script proporciona una amplia gama de nuevas clases que te permiten realizar todo tipo de interacción con los documentos. De esta manera, para profundizar más en el lenguaje se utilizó la documentación que los desarrolladores de Google Apps Script proporcionan a través de su página oficial [3] como se muestra en la figura 2, donde te explican las diferentes clases que el lenguaje soporta y funciones auxiliares que servirán de gran utilidad.

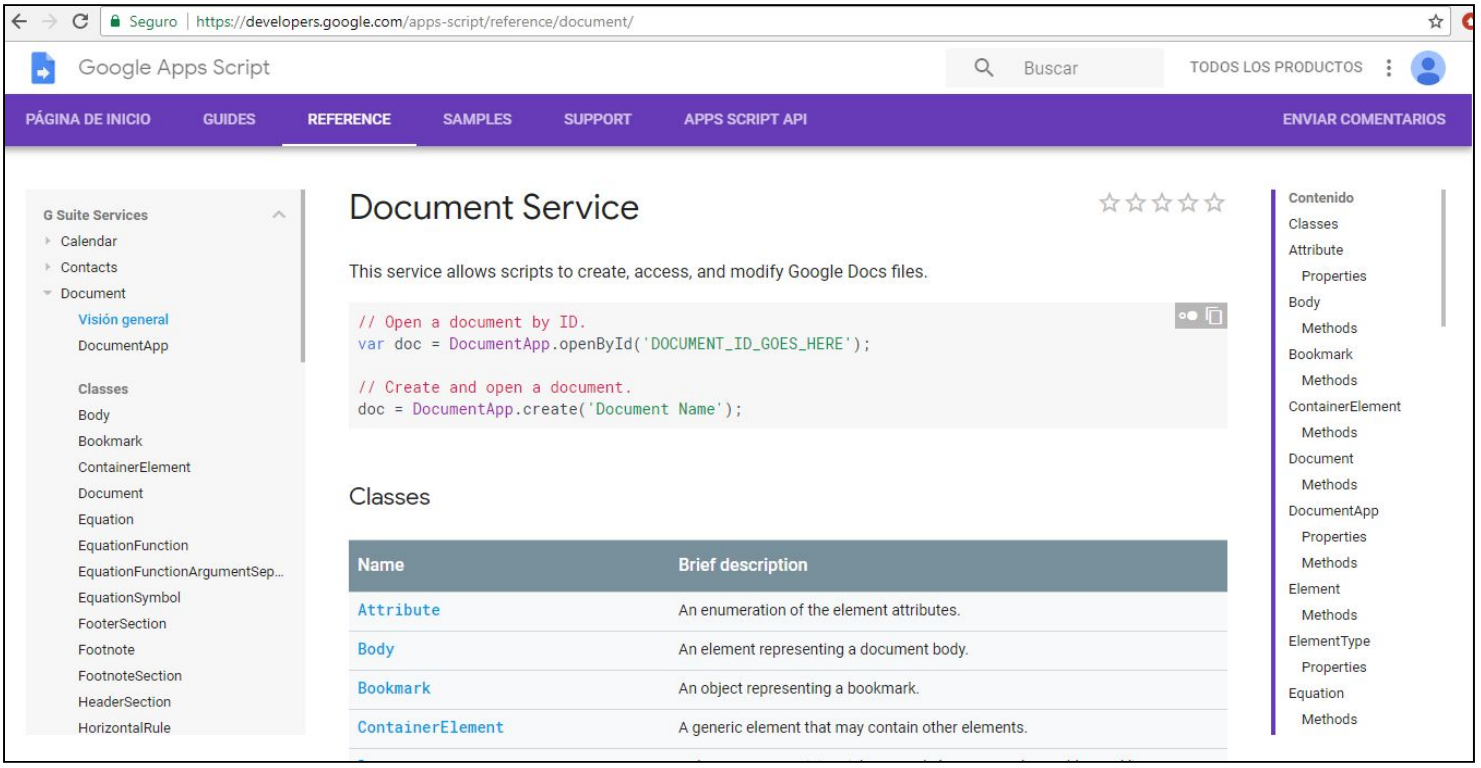



***Figura 2**: Página web oficial de Google Apps Script con toda su documentación.*

Durante las siguientes semanas, se realizó un análisis exhaustivo de los dos principales pilares en este proyecto: BibTex y LaTeX. En lo que respecta a BibTeX, se estudió en profundidad la página oficial de *Alexander Feder*: http://www.bibtex.org. Durante el aprendizaje del mismo, también tuvo importancia el libro [4] acerca del formato de los autores, el cual está centrado única y exclusivamente en el estudio del mismo.

El estudio de LaTeX dirigió el interés y el esfuerzo en saber cómo este sistema de editor de textos maneja internamente las citas bibliográficas y cómo las transforma para crear la nueva sección bibliográfica.

Durante la investigación de este campo, se encontró una versión beta en un repositorio de GitHub [5] que se tomó como base para la futura elaboración de este Trabajo de Fin de Grado, pues aportaba métodos y procesos de gran utilidad. Inicialmente, esta versión beta que se encontró, únicamente leia un fichero Bibtex para devolver su contenido como una lista de objetos. Es por eso que posteriormente se procedió a una fase de depuración de esta versión preliminar, con el objetivo de identificar y corregir los errores presentes en ella añadiendo nuevas funcionalidades que creíamos interesantes. Además, durante este proceso se trató de comprender el funcionamiento de la misma aplicando ingeniería inversa. De esta forma, se profundizó en todos sus componentes recaudando información del funcionamiento de sus métodos, desde funciones auxiliares hasta la propia clase que realizaba todo el procedimiento principal.

Una vez conocidos todos los aspectos importantes que se iban a utilizar, se procedió a elaborar el complemento aplicando los estudios aprendidos durante la investigación preliminar.

A través de GitHub (figura 3) se mantuvo durante todo el desarrollo del proyecto una gestión de versiones, impidiendo así la pérdida de cualquier dato, manteniendo el código almacenando de manera incremental a medida que se añadían nuevas implementaciones y pudiendo recuperar cualquier antigua versión en el momento que se desee.

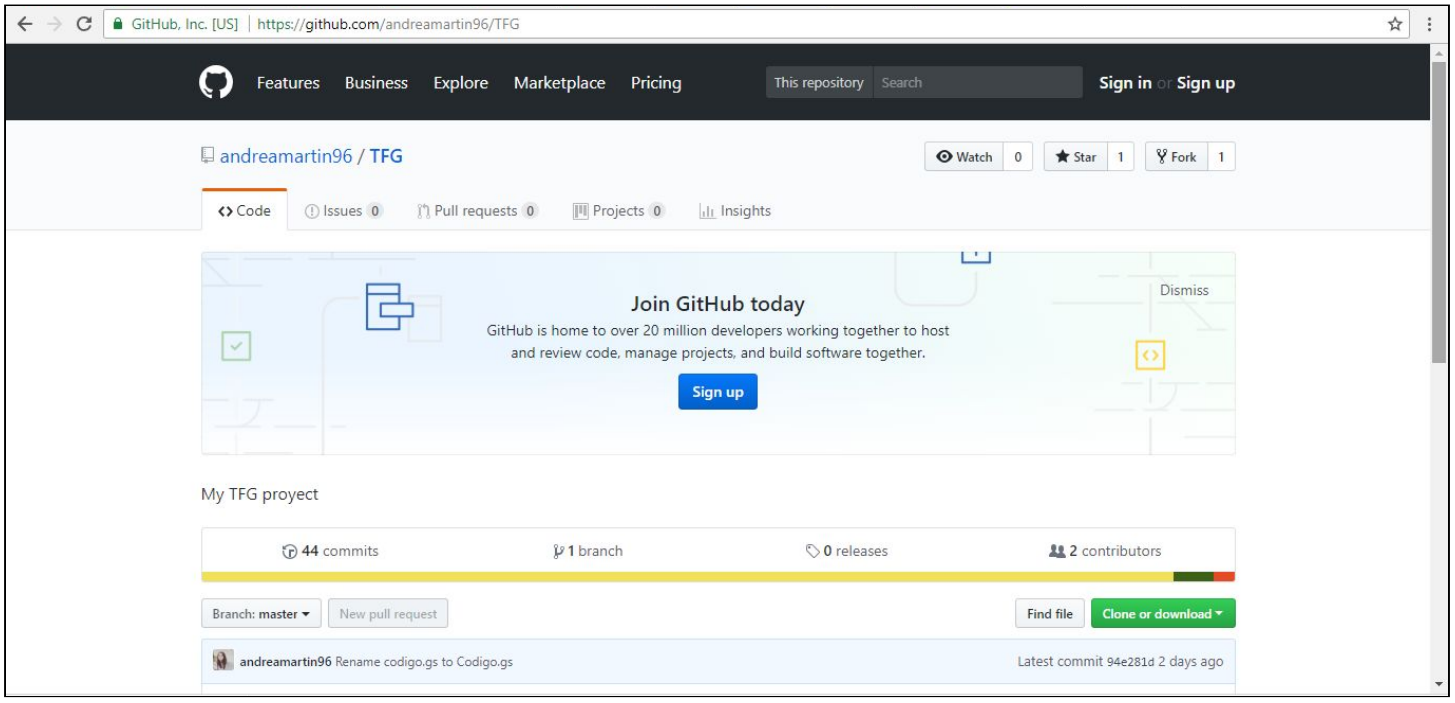



***Figura 3**: GitHub del proyecto: [https://github.com/andreamartin96/TFG](https://github.com/andreamartin96/TFG).*

En los que respecta al reparto de trabajo, inicialmente se realizó un reparto equitativo de las tareas donde queda persona se ocupaba de unas actividades que posteriormente se ponían en común. Sin embargo, uno de los integrantes del grupo de trabajo abandonó el proyecto y todas las tareas pasaron a mano de la otra persona que formaba el grupo. De igual manera, el escrito de esta memoria también forma parte de esas tareas asignadas al integrante restante del grupo.

## 1.3. Explicación de la Memoria

A lo largo de esta memoria explicaremos los conceptos básicos que serán utilizados en el desarrollo de este proyecto. Es por ello que, antes de proceder con dicho desarrollo, destacaremos tres grandes pilares en la sección de Preliminares (sección 3):

- Google y sus diferentes formas de elaborar nuestros documentos.
- El sistema de editor de textos LaTeX.
- BibTeX, una herramienta para dar formato habitualmente utilizada con LaTeX.

Una vez explicado estos conceptos necesarios, en la sección 4 nos adentraremos en los métodos y funciones de nuestro nuevo componente que, simulando las funcionalidades de LaTeX, es capaz de elaborar una sección de referencias a partir de un documento de texto con marcas y un fichero BibTeX. Aquí explicaremos e interpretaremos todas las funciones del código haciendo hincapié en aquellas que tengan más importancia.

Durante la sección 5 se detallarán los problemas surgidos a lo largo del desarrollo del complemento. En ella, se detallarán algunos de los mayores obstáculos que se tuvieron tanto en el proceso de investigación como implementación.

A lo largo de la sección 6 llevaremos a cabo un breve explicación de cómo poder utilizar el complemento desarrollado en este proyecto. Para ello, se detallará una guía con los pasos a seguir incluyendo imágenes de los diferentes procedimientos para lograr una mejor descripción.

Finalmente, en la sección 7 nos dedicaremos a mencionar cuáles serían los mejores puntos a mejorar en nuestro proyecto, destacando los posibles trabajos futuros que podrían hacer a este complemento aún mejor y más eficiente.

# 2. Introduction

When writing a document, a reference section is frequently included. Every source of information (books, articles, thesis…) quoted or mentioned in the document must be reflected in a brief summary that contains its most relevant information, such as the title, the author, or the date of issue. Despite the simplicity of the concept, writing a reference section is no easy task: as the document's volume increases, more and more bibliographic sources need to be included, and what seems an effortless process turns out to be a painful burden.

Our project intends to shorten the time spent on this task, reducing the process to a simple click. Thus, users will avoid focusing their efforts on the confection of a reference section, and will be able to target other meaningful aspects of their document.

The core of this project is a JavaScript cloud scripting language that provides the tools needed to manage and automate document creation in a simple, efficient way: Google Apps Script [1]. With that being said, this particular task will be centred around the LaTeX syntax which, in addition to BibTeX, will be the cornerstone of our project.

In order to use Google Apps Script, we will focus on the document editor that Google provides: Google Docs. This way, a wide range of possibilities will be at our disposal. The platform, commonly used for simple text editing purposes, is actually capable of so much more. For instance, with a few lines of code we can access a Google Drive file storage, where any user can store up to 15GB worth of documents from the very first moment.

Using Google Apps Script, we will try to add new functionality, aiming to accomplish certain tasks in the minimum time required and, therefore, developing a new Google Add-on.

Currently, there is a long list of Google Add-ons available for any user to try. These extensions are built by third-party developers in order to perform tasks that are not supported natively by the Google Docs editor.

With our Add-on, users will be able to choose a BibTeX document from their Google Drive storage and, with a simple click of the mouse, generate a reference section for it. All references included in the document will be automatically detected, identified, and summarised in a new section located wherever the user specifies.

## 2.1. Objectives

The main goal for this project is to simplify the creation of a reference section in a Google Docs document. To accomplish this goal, we will need to dive into the world of LaTeX and BibTeX, applying the concepts we have learned about HTML5 and Javascript in order to familiarize ourselves with a new language: Google Apps Script.

There are also three secondary goals:

- Learning how to handle Google Apps Script to create an Add-on that meets the desired functionality.
- Getting to know the LaTeX syntax, and understanding its in-depth procedure in order to simulate it.
- Becoming familiar with BibTeX to identify the different styles and formats it supports.

## 2.2. Work plan

The following paragraphs include a compilation of patterns and guidelines proposed during the project planification, and a detailed description of all tasks accomplished throughout the development of the project.

Initially, the project was designed for groups of two people and therefore, in order to achieve an adequate task distribution, a first meeting was arranged. In this meeting, activities were divided and organised, and an in-depth analysis of the project was made. The three main highlighted items in this study were:

- Getting to know Google Apps Script.
- Learning about LaTeX's functionality, to understand its syntax and semantics.
- Delving into the BibTeX foundation to find out how these files work.

Once all these tasks were taken into consideration, we proceeded to divide the activities, trying to reach a fair distribution. Each member would cover a different area so the preliminary research stage could be shortened. After that, each member went on to share their results, explaining the key aspects they had found in the course of their research.

Thus, the first weeks were dedicated to becoming comfortable with this new programming language. Several tutorials were run in the Google Apps Script website to learn how to build new Add-ons step by step, as shown in Figure 4. One of these tutorials, which consisted of a simple text-translation tool [2], would serve as the basis for our project.

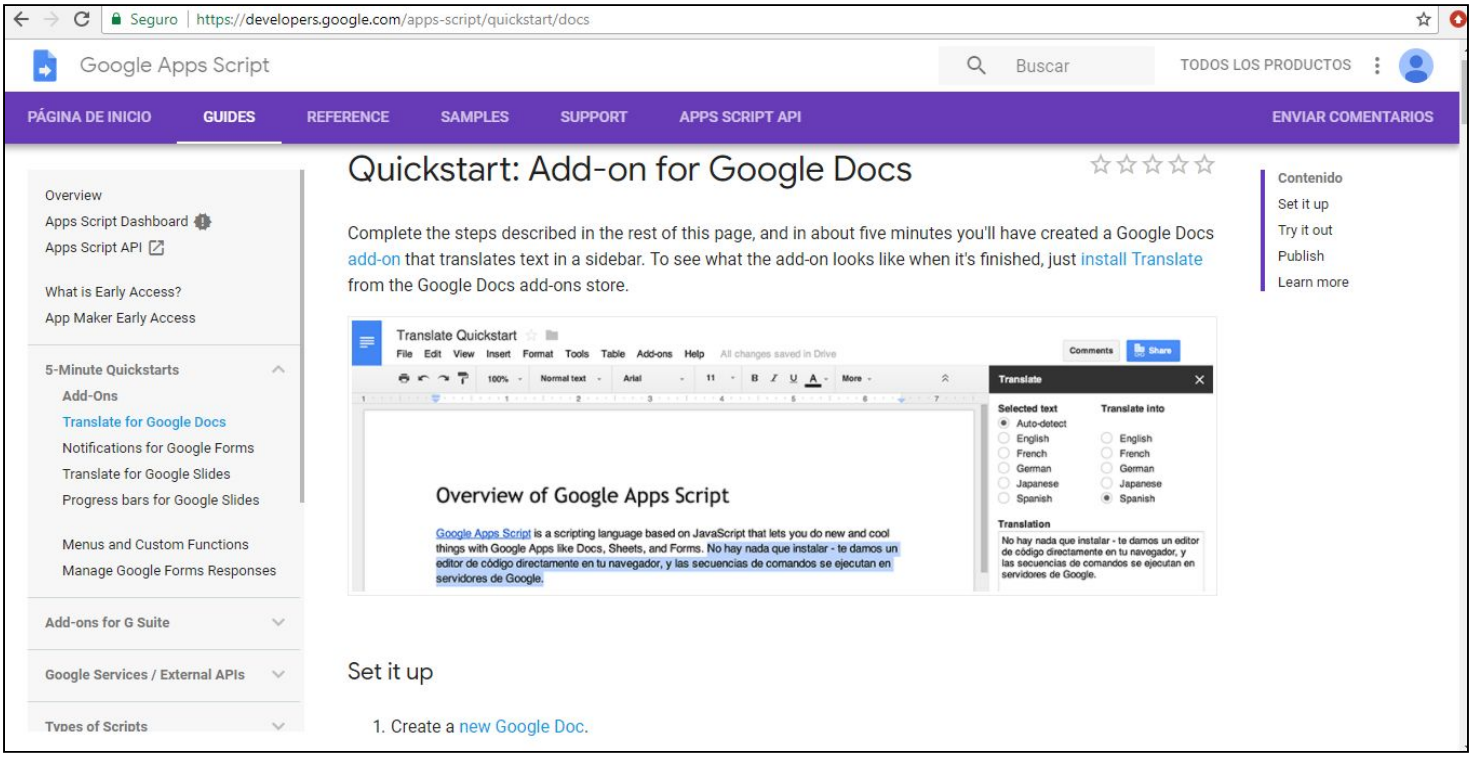



***Figure 4**: Google Apps Script official website has simple tutorials about their Add-ons.*

These guides show us that, with very few lines of code, it is possible to create new Add-ons to add extra functionality to our Google Documents. However, this is only the tip of the iceberg, compared with the astounding potential these extensions have, as Google Apps Script provides a wide range of classes for us to interact in many different ways with our documents. In our case, we used the documentation that the Google Apps Script developers provide in the official website [3], as shown in Figure 5. There, the various classes and auxiliary functions this language supports are explained in detail.

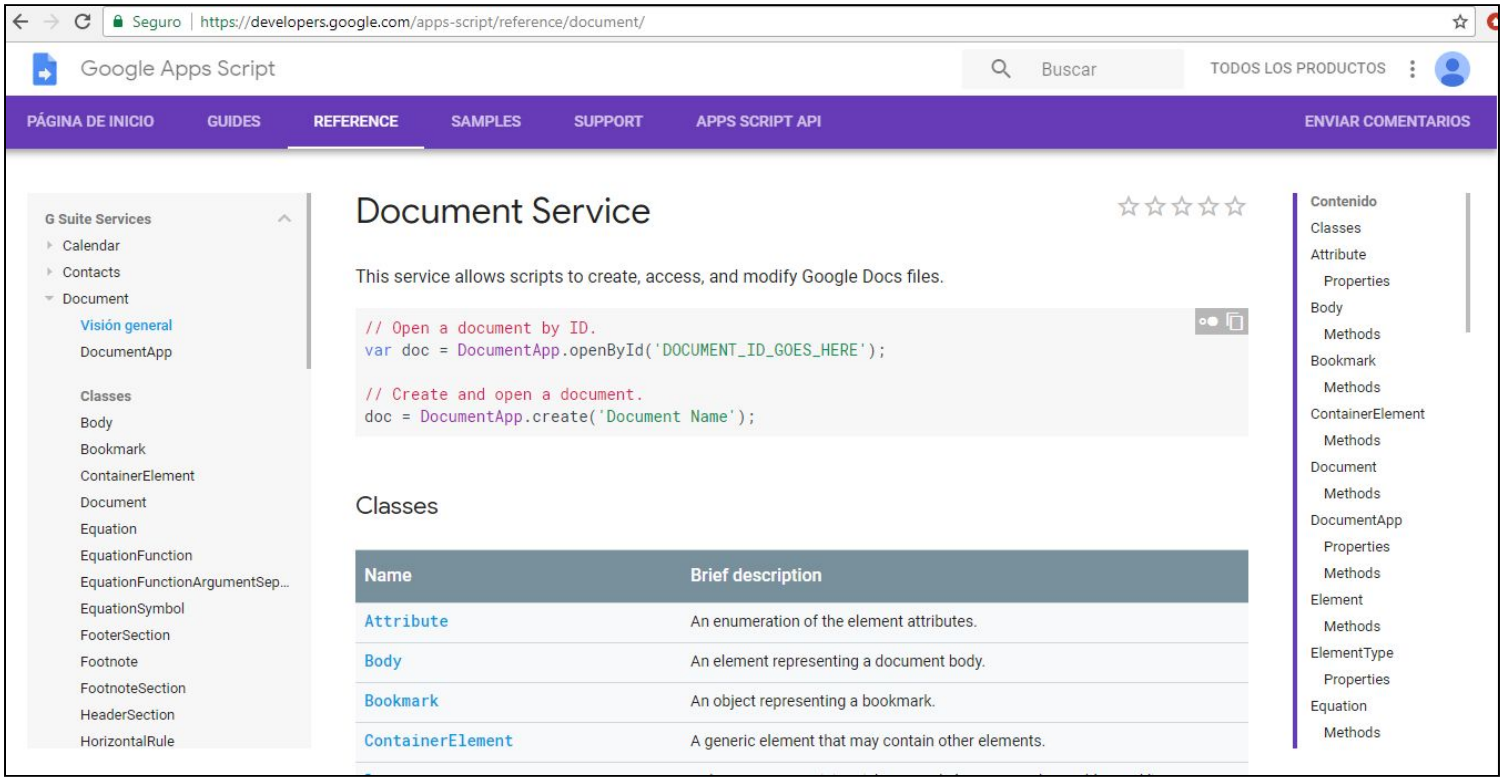



***Figure 5**: Google Apps Script official website, and the developer documentation.*

During the following weeks, a thorough analysis was made, concerning the two most important pillars of this project: BibTeX and LaTeX. Regarding BibTeX, we extensively studied Alexander Feder's official website (http://www.bibtex.org). Throughout the learning process, the book [4] about the authors' format —which is exclusively focused on this topic— was also very significant.

The study of LaTeX sparked a greater interest and effort towards learning how this text-editing system internally handles bibliographic quotes, and how it transforms them to create the new reference section.

While researching this particular field, a beta version was found in a GitHub repository [5]. This beta version would serve as the foundation for our project, as it provided very useful methods and procedures. Initially, this beta could only read a BibTeX file and return its content as an object list. Later on, a debugging stage took place with the purpose of spotting and correcting the errors found in this preliminary version, and including new capabilities that we considered interesting. Furthermore, in the course of this process, we attempted to understand how the application worked, via reverse engineering. This way we delved into each component, gathering information about its methods, from auxiliary functions to the class that executed the main procedure.

Once all the relevant components that were going to be used were completely understood, we proceeded to elaborate the Add-on applying the concepts studied during the previous research stage.

The project was managed through GitHub (Figure 6), as we needed a version control system all along the development of the Add-on to avoid data loss and to maintain the code organized and stored as new functionality was implemented. This also meant we could revert changes and go back to any previous version if necessary.

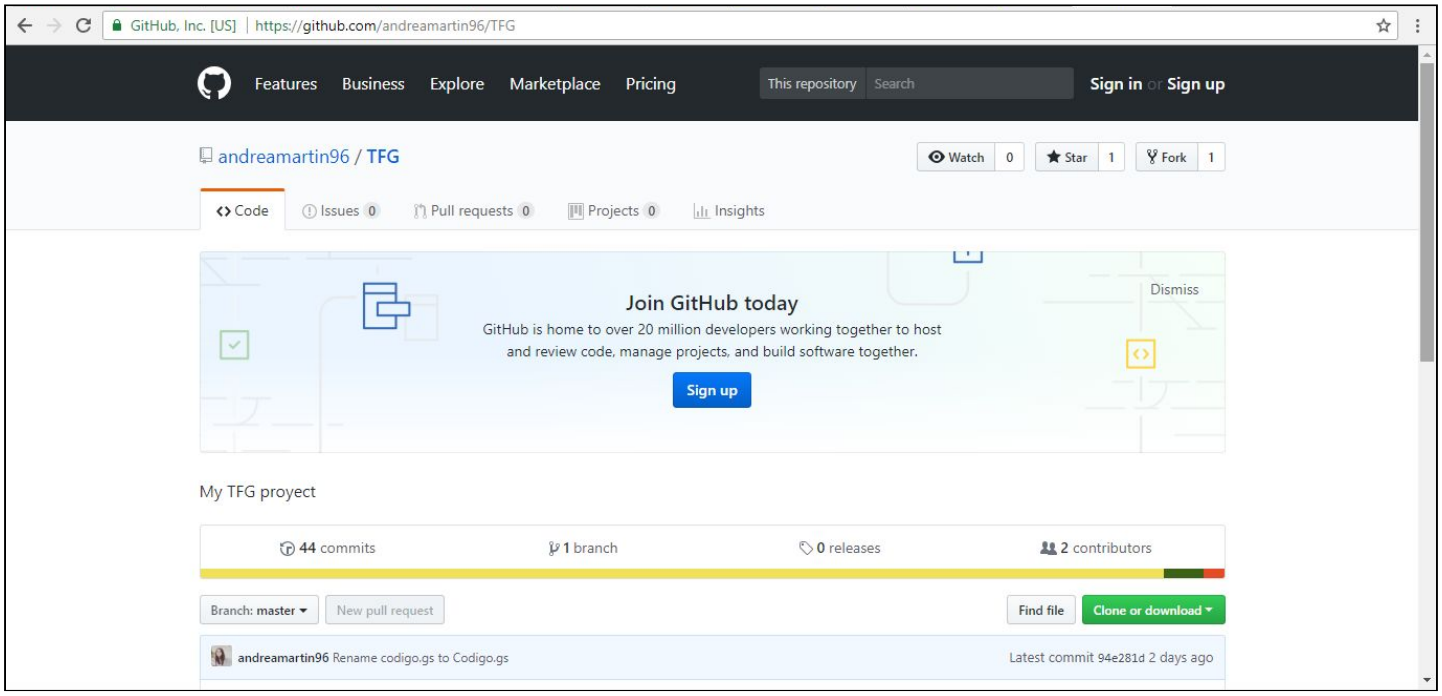



***Figure 6:*** *GitHub repository of the project: [https://github.com/andreamartin96/TFG](https://github.com/andreamartin96/TFG)*

With regards to task distribution, initially each member was given a fair amount of the total number of activities so, after that, each member would share their results with the rest of the group. However, one of the members in our group left the project, so the tasks that had been designated to him had to be reassigned. In fact, this particular document had to be referred to another member of the group when the aforementioned member abandoned the project.

## 2.3. Explanation of memory

In the following paragraphs, we will attempt to explain the basic concepts that were used during the development of our project. But before getting into detail, it is necessary to point out the three vital pillars in the Preliminary section (Section 3). These pillars are:

- Google, and the services it provides for us to elaborate our documents.
- The LaTeX text editing tool.
- BibTeX, a text formatting tool that is commonly used along with LaTeX.

Once these basic concepts have been clarified, in the fourth section we will dive into the methods and functions of our new component which, simulating LaTeX's functionality, is capable of generating a reference section when given a document with reference marks and a BibTeX file. In this section, we will explain every function used in the code, emphasizing the most relevant ones.

During the fifth section, a list of difficulties encountered while developing the Add-on will be included. In this list, we will highlight the main obstacles we found in the process, and we will go into detail about the different issues that arose during both the research and implementation stages.

As of the sixth section will contain a brief explanation on how to use the Add-on we have developed. It will include a step-by-step guide that will illustrate with images the different procedures, in order to achieve a better, more detailed description.

Finally, the seventh section, we will go on to mention some of our project's shortcomings, and how they could be addressed in future work in order to improve this Add-on and make it more efficient.

# 3. Preliminares

A lo largo de esta sección nos adentraremos en los conceptos básicos que se deben conocer para entender con claridad todos los términos que se usarán a lo largo de esta memoria.

## 3.1. Google Drive

Google Drive proporciona una amplia gama de ficheros de todos los tipos, desde Hojas de Cálculo y Formularios, hasta Documentos y Presentaciones de Google. Es por ello, que en este proyecto nos centraremos en Documentos de Google a los que se les

puede aplicar diversos complementos (*Addons*) para dar más juego a nuestros documentos, utilizando para ello Google Apps Script y HTML5.

### 3.1.1. Documentos de Google (*Google Docs*)

Como ya hemos mencionado, Google proporciona distintas formas de elaborar escritos de todos los tipos. Los Documentos de Google se caracterizan porque puedes colaborar con personas de todo el mundo para elaborar escritos utilizando herramientas inteligentes de edición, añadiendo imágenes, enlaces o incluso dibujos. Los cambios en estos documentos se irán guardando de manera automática manteniendo un historial de versiones donde se indican los cambios realizados y el autor que los realizó pues varias personas pueden editar un mismo documento al mismo tiempo desde cualquier parte, incluso a través del móvil.

A través del menú superior puedes añadir complementos que te servirán de ayuda a la hora de elaborar tu documentos, desde correctores de la acentuación hasta traductores de documentos.

### 3.1.2. Complementos (*Addons*)

Los Addons son complementos que sirven como recurso adicional para extender los documentos, permitiendo traducir textos, añadir notificaciones o incluir otras muchas funcionalidades adicionales.

Están basados en un *script* que se ejecuta de manera transparente al usuario. Este *script* se ayuda de un documento HTML que proporciona la interfaz de usuario con la que el cliente interactúa. No obstante, también hay otras formas de proporcionar al usuario estas funcionalidades del script, como por ejemplo un menú que se añade a lista superior de elementos desplegables que hay por defecto.

Para la publicación de los Addons se puede seguir los pasos que los desarrolladores de Google Apps Script proporcionan a través de una guía detallada [6], no obstante, por motivos de complejidad y tiempo, se decidió no subir este complemento. Una vez que el usuario ha subido a la nube el nuevo complemento que ha desarrollado, cualquier persona desde cualquier parte del mundo puede acceder a dicho complemento descargándolo a través del buscador que se encuentra en la sección superior de tu documento: “*Complementos*” -- “*Obtener complementos…*”.

### 3.1.3. Google App Script

Google Apps Script es un lenguaje de programación que permite elaborar scripts capaces de acceder a Documentos de Google a través de métodos proporcionados por los diferentes elementos que conforman el árbol de clases. No solo permite acceder a ellos, sino que también concede la posibilidad de crear nuevos documentos que se guardarán de forma inmediata a través del servicio de Google Drive.

Gracias a este lenguaje es posible elaborar las diferentes funciones que componen a los complementos. A través de Google Apps Script, se crean los Scripts anteriormente mencionados que incluirán los métodos encargados de interactuar con los documentos de Google. Estos Scripts se relacionan generalmente con archivos HTML que proporcionan la interfaz de usuario con la que interactúa el cliente.

Los desarrolladores de Google Apps Script proporcionan una guía [3] extensa en la que proponen muchos ejemplos de uso a través de tutoriales con los pasos a seguir para su elaboración. Esta guía también ofrece un análisis detallado de todas las funciones y clase que el lenguaje soporta.

En el diagrama que se muestra a continuación, se pueden observar todas las clases que están a disposición del usuario durante el desarrollo de un complemento, mostrando el árbol clases y las relaciones que presentan entre ellas:

- Document
  - Body
    - ListItem
      - Equation
        - EquationFunction
          - EquationFunction...
          - EquationFunctionArgumentSeparator
          - EquationSymbol
          - Text
        - EquationSymbol
        - Text

        - Footnote
        - HorizontalRule
        - InlineDrawing
        - InlineImage
        - PageBreak
        - Text
    - Paragraph
        - Equation
            - EquationFunction
                - EquationFunction...
                - EquationFunctionArgumentSeparator
                - EquationSymbol
                - Text
            - EquationSymbol
            - Text
        - Footnote
        - HorizontalRule
        - InlineDrawing
        - InlineImage
        - PageBreak
        - Text
    - Table
        - TableRow
            - TableCell
                - Paragraph...
                - ListItem...
                - Table...
    - TableOfContents
        - Paragraph...
        - ListItem...
        - Table...

Dada la enorme extensión del árbol de clases, a continuación nombraremos aquellas clases que más hemos utilizado en el desarrollo de nuestro documento: `Paragraph`, `Text`, `Table`, `Body` y `Document`.

## Paragraph

Esta clase representa la estructura básica que compone los documentos: *los párrafos*. Está compuesta por clases más pequeñas como *Text* y dispone de una amplia gama de funciones que permiten desde añadir nuevas cadenas de texto al párrafo hasta aplicar cierto estilos y formatos para darles forma y color.

## Text

Es una de las clases que se encuentra más abajo en la jerarquía de clases que componen el árbol. Representa cadenas de texto básicas a las cuales se les pueden aplicar los estilos que comúnmente utilizamos cuando desarrollamos un documento de texto: texto en **negrita**, *itálica*, subrayado… También proporciona la posibilidad de comprobar si dispone de unos estilos en concreto o incluso añadir más cadenas de texto a la frase. Esta clase se puede añadir a otras muchas clases a través de funciones que la misma proporciona como por ejemplo a tablas o a párrafos.

## Table

Esta clase concede la posibilidad de elaborar tablas formadas por filas y columnas de celdas, a partir de otras clases más pequeñas que la componen: **`TableRow`** y a su vez **`TableCell`**. Cuando se añade una nueva fila (`TableRow`) se le puede añadir el número de celdas (`TableCell`) deseado que representará el número de columnas que tiene la fila. De esta forma se pueden crear tablas de todas las formas y tamaños a partir de estructuras más pequeñas.

## Body

El cuerpo de un documento (`Body`) es la estructura principal que simboliza el esqueleto que lo forma. A partir del cuerpo se pueden añadir al documento todas las clases que anteriormente hemos mencionado. Esta clase representa la base de una estructura sobre la que se elevan el resto de clases que engloban toda la información. Se podría decir que el Body es el elemento maestro que contiene el resto de piezas salvo tres: **`HeaderSection`**, **`FooterSection`** y **`Footnotes`**.

## Document

La clase `Document` es la raíz del árbol de clases de Google Apps Script, siendo el elemento base del que se extienden el resto de clases. Esta clase proporciona métodos que permiten abrir documentos de Google Drive para acceder a ellos y modificarlos o incluso la posibilidad de crear nuevo documentos que serán almacenados en nuestro Drive. También proporciona funciones auxiliares que permiten enviar correos electrónicos, añadir `BookMarks` o incluso añadir a un usuario como editor del documento.

En lo que respecta a Google Apps Script, uno de los principales conceptos a tener en cuenta, y que lo distingue de otros muchos lenguajes de programación, es que todas estas clases que el propio servicio provee se trabaja con ellas una vez insertadas en el documento. Es decir, no existe la posibilidad de crear un objeto de la clase, añadir los elementos deseados y posteriormente insertarlo en el documento en el lugar deseado. Esto hace que haya que cambiar la mentalidad a la hora de programar. Para explicarlo mejor a continuación se mostrará un ejemplo intuitivo del mismo:

Imaginemos que queremos crear una clase `Paragraph` para crear un párrafo al final de nuestro documento, se haría de la siguiente manera:

1. Obtenemos el cuerpo de nuestro documento:

```
var body = DocumentApp.getActiveDocument().getBody();
```

2. Añadimos el nuevo párrafo al cuerpo del documento con texto: “Some text ”:

```
var myParagraph = body.appendParagraph("Some text ");
```

3. Añadimos más texto a nuestro párrafo devolviendo el nuevo texto insertado:

```
var newText = myParagraph.appendText("and more text.");
```

4. Creamos un nuevo estilo:

```
var style = {};

style[DocumentApp.Attribute.BOLD] = true;
```

5. Y se lo aplicamos al texto añadido:

```
newText.setAttributes(style);
```

De esta manera, el párrafo añadido al final de nuestro documento tendrá el siguiente contenido:

“Some text **and more text.**”

## 3.2. LaTeX

LaTeX es un editor de texto profesional, que está basado en marcas. Este permite la composición tipográfica de alta calidad, permitiendo la elaboración y publicación de documentos de carácter científico. LaTeX es un software libre que utiliza un lenguaje de marcas para construir el esqueleto de documentos de distintos

tipos (libros, artículos…). A través de ello LaTeX proporciona un lenguaje de alto nivel que se apoya en TeX para proporcionar a los escritores y académicos una forma más sencilla de elaborar documentos.

Aunque inicialmente estaba destinado al mundo científico e informático, LaTeX fue utilizado de forma incremental por académicos y estudiantes que querían añadir expresiones complejas del mundo matemático, o palabras en lenguajes que no utilizan nuestro mismo alfabeto como el chino o sánscrito. A través de documentos LaTeX, el programa de TeX elabora los documentos científicos a partir de macros o marcas que se definen en los documentos LaTeX originales.

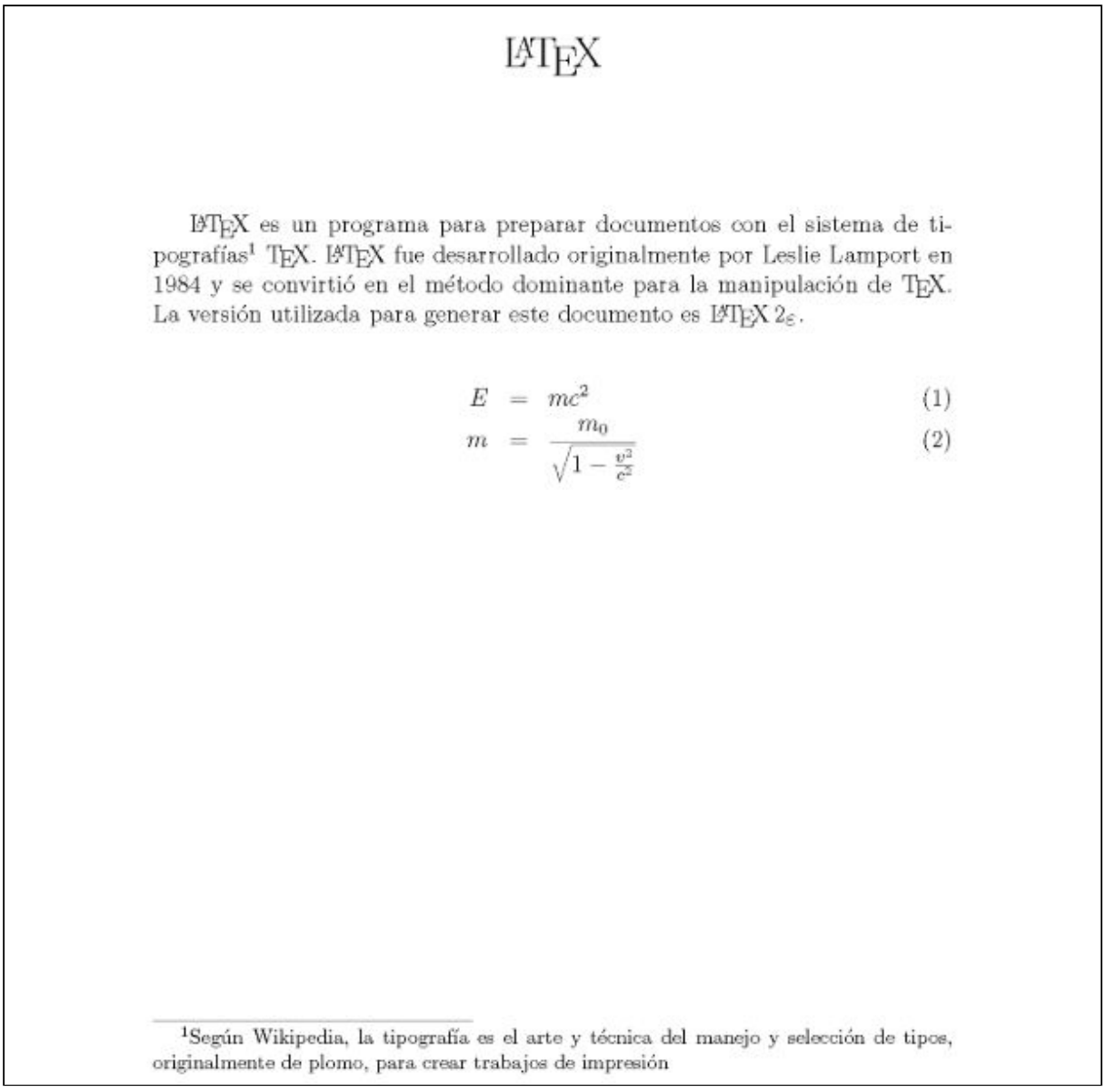
La mayoría de las marcas que LaTeX utiliza vienen definidas por el identificador **** seguido de ciertas palabras concretas que determinan qué proceso se debe realizar con cada una de ellas. Un posible ejemplo de las marcas que podríamos encontrar en un documento de LaTeX tal y como se muestra en el siguiente ejemplo extraído de la Wikipedia [7]:

```
\documentclass[12pt]{article}
\usepackage[spanish]{babel}
\usepackage{amsmath}
\title{\LaTeX}
\date{}
% Este es un comentario, no será mostrado en el documento
final.
\begin{document}
\maketitle

\LaTeX{} es un programa para preparar documentos con el
sistema de
tipograf\'ias\footnote{%nota al pie de página
```

```
Seg\'un Wikipedia, la
tipograf\'ia es el arte y t\'ecnica del manejo y selecci\'on de tipos,
originalmente de plomo, para crear trabajos de impresi\'on }
%fin nota al pie de página
\TeX{}. \LaTeX{} fue desarrollado originalmente por Leslie Lamport en 1984 y se convirti\'o en el m\'etodo dominante para la  manipulaci\'on
de \TeX. La versi\'on utilizada para generar  este documento es \LaTeXe.
\newline
% El siguiente código muestra la calidad de la tipografía de LaTeX
\begin{align}
E &= mc^2                                    \\
m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}
\end{align}
\end{document}
```

A partir de estas marcas, LaTeX interpreta la forma que tienen que tener los documentos permitiendo desde definir el tipo de documento, indicar acentuaciones o incluir notas a pie de página, hasta elaborar notaciones matemáticas complejas. De este modo, el resultado que LaTeX generaría a partir del ejemplo anterior, se vería tal y como se muestra en la figura 7.



LaTeX

LaTeX es un programa para preparar documentos con el sistema de tipografías[1] TeX. LaTeX fue desarrollado originalmente por Leslie Lamport en 1984 y se convirtió en el método dominante para la manipulación de TeX. La versión utilizada para generar este documento es LaTeX $2_\varepsilon$.

$$E = mc^2 \quad (1)$$

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \quad (2)$$

[1]Según Wikipedia, la tipografía es el arte y técnica del manejo y selección de tipos, originalmente de plomo, para crear trabajos de impresión

***Figura 7**: Ejemplo de un documento generado por LaTeX.*

Existen muchos tipos de marcas que LaTex maneja, sin embargo, para ofrecer al usuario un uso más sencillo, en nuestros documentos solo serán necesario dos tipos de etiquetas:

- **Las marcas de citas**: Se definen con el identificador `\cite` seguido entre llaves por la clave única que debe existir en el archivo BibTeX. Estas marcas permiten localizar el documento que se quiere referenciar dentro de los archivos BibTeX.

- **La etiqueta bibliográfica**: Permite determinar el lugar en el que se quiere crear la sección de referencias. Se define con el identificador propio de marcas bibliográficas `\bibliography`.

De esta forma, en este proyecto trataremos de simular esta funcionalidad basada en marcas que TeX transforma a partir de los documentos LaTeX tal y como se muestra en la figura 8 que se muestra a continuación.

```
Hay muchas variaciones de los pasajes de Lorem Ipsum disponibles, pero la mayoría sufrió alteraciones en alguna manera, ya sea porque se le agregó humor, o palabras aleatorias que no parecen\cite{Shapiro83} ni un poco creíbles. Si vas a utilizar un pasaje de Lorem Ipsum, necesitás estar seguro de que \cite{hola,Verdejo99,testingBook07} no hay nada vergonzante escondido en el medio del texto. Todos los generadores de Lorem Ipsum que se encuentran en Internet tienden a repetir trozos predefinidos cuando sea necesario, haciendo\cite{testingBook07} a éste el único generador verdadero (válido) en la Internet. Usa un diccionario de más de 200 palabras provenientes del latín, combinadas con estructuras muy útiles de sentencias, para generar texto de Lorem Ipsum que parezca razonable. \cite{GoguenMalcolm96} Este Lorem Ipsum generado siempre estará libre de repeticiones, humor agregado o palabras no características del lenguaje, etc.\cite{testingMc}


\bibliography
```

***Figura 8**: Ejemplo de un documento con marcas de citas bibliográficas.*

Inicialmente tendremos una serie de marcas específicas que nuestro programa interpretará para construir la sección bibliográfica utilizando la información proporcionada a través de un archivo BibTeX.

La nueva bibliografía contará con la información al detalle de todos los documentos referenciados que según el estilo bibliográfico escogido, presentarán un orden y formato específicos (figura 9).

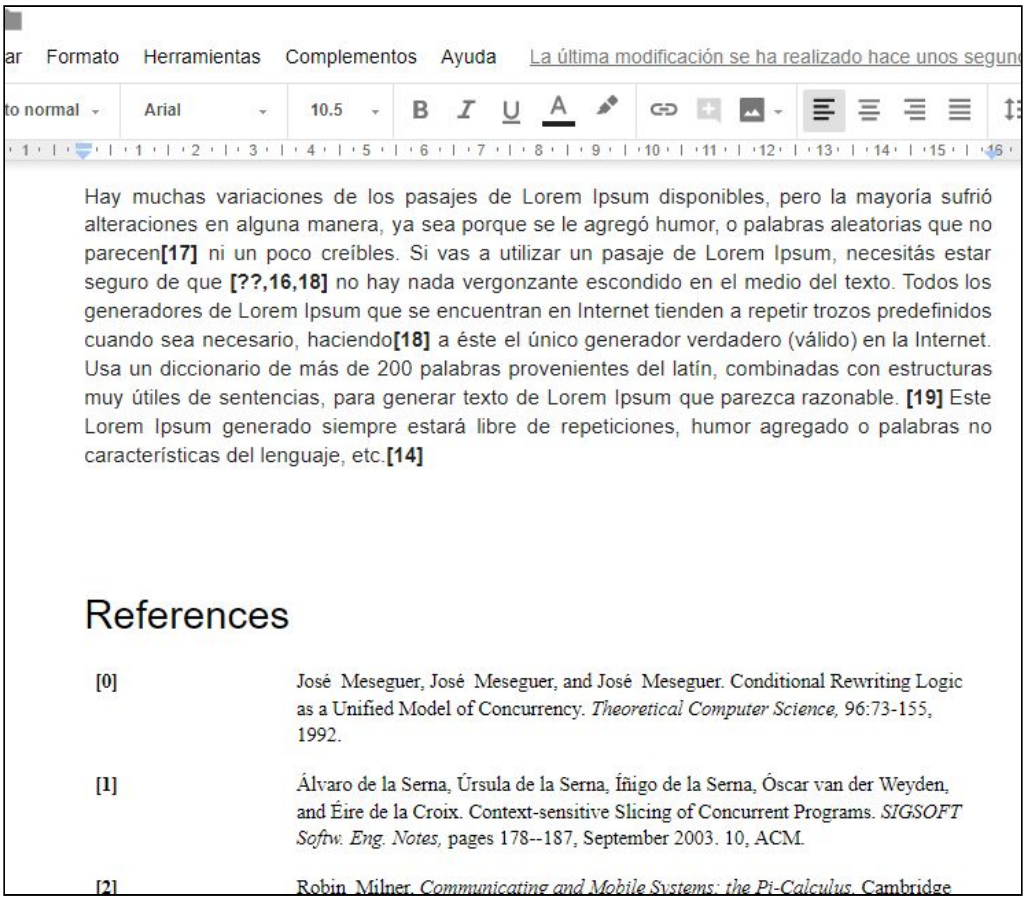



***Figura 9**: Cómo se vería el documento después de elaborar la bibliografía y sustituir las marcas.*

## 3.3. BibTeX

Es una herramienta que usa un formato de archivo especial para dar estilo a citas bibliográficas en documentos. BibTeX permite crear un fichero de texto a partir de una lista de ítems utilizados habitualmente en bibliografías (conferencias, artículos, tesis, libros, manuales…) independientemente del estilo. Esta lista de elementos bibliográficos representan las diferentes entradas que podemos encontrar en un archivo de este tipo.

LaTeX, a través de la etiqueta o marca `\cite`, permite hacer referencia a una entrada BibTeX que contiene toda la información que se debe añadir en la bibliografía para ese documento referenciado. De esta manera, si esta marca específica aparece en un documento LaTeX (seguida de un identificador que coincidirá con el ID de la entrada BibTeX que contiene la información bibliográfica del documento), BibTeX será capaz de añadir esta referencia en la bibliografía según el estilo que se le haya indicado en un fichero de estilo.

Dentro de esta sección identificamos dos apartados a destacar: las entradas y los estilos.

### 3.3.1. Entradas

A lo largo de un fichero BibTeX existen diferentes tipos de entradas que se pueden utilizar según el documento que se quiera referenciar. Dado el creciente número de entradas, en este trabajo nos hemos centrado en una lista más reducida de elementos bibliográficos: *article, book, inproceedings, proceedings, mastersthesis, phdthesis, techreports* y *misc.*

- **Article**: Artículos de un periódico o revista.
- **Book**: Libros con una editorial explícita.
- **Inproceedings**: Artículos de unas actas de una conferencia.
- **Proceedings**: Actas de conferencias.
- **Mastersthesis**: Una tesis de máster.
- **Phdthesis**: Una tesis doctoral.
- **Techreports**: Un informe publicado por una escuela u otra institución, generalmente numerado dentro de una serie.
- **Misc**: Este tipo de entrada se utiliza cuando el documento que quieres referenciar no encaja en ninguno de los anteriores tipos

Todos estos tipos de entradas se caracterizan por una serie de campos obligatorios y opcionales que contienen toda la información de los elementos anteriormente mencionados, como artículos de un periódico, reportes de una institución o colegio o tesis de un doctorado o maestría. Para los tipos de entradas que trataremos en este trabajo es importante conocer qué campos son obligatorios y opcionales:

| | |
|---|---|
| article | *Campos requeridos: author, title, journal, year.*<br>*Campos opcionales: volume, number, pages, month, publisher, address, note.* |
| book | *Campos requeridos: author or editor, title, publisher, year.*<br>*Campos opcionales: volume or number, series, address, edition, month, note.* |
| inproceedings | *Campos requeridos: author, title, booktitle, year.*<br>*Campos opcionales: editor, volume or number, series, pages, address, month, organization, publisher, note.* |
| mastersthesis | *Campos requeridos: author, title, school, year.*<br>*Campos opcionales: type, address, month, note.* |
| misc | *Campos requeridos: none.*<br>*Campos opcionales: author, title, howpublished, month, year, note.* |
| phdthesis | *Campos requeridos: author, title, school, year.*<br>*Campos opcionales: type, address, month, note.* |
| proceedings | *Campos requeridos: title, year.*<br>*Campos opcionales: editor, volume or number, series, address, publisher, note, month, organization.* |
| techreport | *Campos requeridos: author, title, institution, year.*<br>*Campos opcionales: type, number, address, month, note.* |

De esta forma, una posible entrada que nos podríamos encontrar en un archivo BibTeX sería de la forma en la que se muestra en el ejemplo a continuación, donde el texto resaltado en azul representaría el identificador de la entrada, el color rojo es el tipo de entrada y el texto que está en color negro identifica los diferentes campos que una entrada puede tener, tanto obligatorios como opcionales:

```
@article{Krinke:2003:CSC,
author = {Enrique Martin-Martin and Adrian Riesco and Giacomo della Porta},
 title = {Context-sensitive Slicing of Concurrent Programs},
 journal = {SIGSOFT Softw. Eng. Notes},
 volume = {28},
 number = {5},
 month = sep,
 year = {2003},
 pages = {178--187}
}
```

### 3.3.2. Estilos

Existe una larga lista de estilos que pueden aplicarse para mostrar la información bibliográfica según se desee. En este proyecto nos hemos centrado únicamente en el estilo *unsrt*, un formato sencillo que muestra la información en el orden en el que aparecen las marcas `\cite` en el documento que se está escribiendo, a diferencia de otros como *alpha* que elaboran la sección de referencias ordenando la lista en función del campo *author*.

Este estilo tampoco manipula el campo *author* de la manera en la que lo hacen otros estilos, como por ejemplo *abbrv* que abrevian el nombre de todos los autores del documentos referenciado; simplemente muestra los campos en la forma en la que aparecen en la entrada del fichero BibTeX.

Cuando se aplica el estilo, cierto campos presentan cambios en el formato. En la mayoría de los casos, el campo *title* se muestra resaltado en itálica, sin embargo, si se trata de un artículo, el campo que se resaltaría en itálica sería *journal* en lugar de *title*.

- Para el ejemplo de entrada de la sección 3.3.1., el resultado de aplicarle el estilo *unsrt* y *alpha* sería:

  Enrique Martin-Martin, Adrian Riesco, and Giacomo della Porta. Context-sensitive Slicing of Concurrent Programs. *SIGSOFT Softw. Eng. Notes,* 28(5):178--187, September 2003.

- Mientras que para el estilo *abbrv* se vería de la siguiente forma:

  E. Martin-Martin, A. Riesco, and G. della Porta. Context-sensitive Slicing of Concurrent Programs. *SIGSOFT Softw. Eng. Notes,* 28(5):178--187, September 2003.

El estilo también influye en el formato de notación que aparece en la parte izquierda en una bibliografía. Para la mayoría de estilos, se utiliza la notación numérica **[1]**, **[2]**, etc. No obstante, en algunos casos esta notación se elabora de una forma especial que identifica a ese estilo concreto. Por ejemplo, para el caso del estilo alpha, la notación se elabora juntando parte del apellido del autor y el año, de manera que se vería de la siguiente forma **[Mar03],** y en caso de haber más de un autor, toma las iniciales de los apellidos de los autores **[EAG03]**.

A continuación se mostrará un ejemplo de cómo se mostraría la sección de referencias según el estilo escogido.

- Estilo *unsrt*:


**[0]** Andrea Martín Arias. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science,* 96:73-155, 1992.

**[1]** Enrique Martin-Martin, Adrian Riesco, and Giacomo della Porta. Context-sensitive Slicing of Concurrent Programs. *SIGSOFT Softw. Eng. Notes,* 28(5):178--187, September 2003.

**[2]** Robin Milner, and Jorge García Martín. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, September 1999.


- Estilo *abbrv*:


**[0]** A. Martín Arias. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science,* 96:73-155, 1992.

**[1]** E. Martin-Martin, A. Riesco, and G. della Porta. Context-sensitive Slicing of Concurrent Programs. *SIGSOFT Softw. Eng. Notes,* 28(5):178--187, September 2003.

**[2]** R. Milner, and J. García Martín. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, September 1999.


- Estilo *alpha*:


**[And92]** Andrea Martín Arias. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science,* 96:73-155, 1992.

**[RJ99]** Robin Milner, and Jorge García Martín. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, September 1999.

**[EAG03]** Enrique Martin-Martin, Adrian Riesco, and Giacomo della Porta. Context-sensitive Slicing of Concurrent Programs. *SIGSOFT Softw. Eng. Notes,* 28(5):178--187, September 2003.

# 4. Implementación

A lo largo de esta sección nos centraremos en los elementos y procesos que conforman nuestro programa, destacando las funciones principales y auxiliares utilizadas.

Por lo general, los complementos requieren dos tipos de archivos que funcionan en sincronía para lograr una ejecución con éxito: `Sidebar.html` y Código.gs. El archivo HTML proporciona la interfaz de usuario y las llamadas a funciones de Google Apps Script que se encuentran almacenadas en el fichero Código.gs, el script principal que contiene los métodos y funciones primordiales del esqueleto de nuestro complemento.

Para comprender mejor la implementación de nuestro complemento se ha elaborado un esquema en la figura 10 con la mayoría de las funciones que están presentes durante la ejecución del programa. En esta figura, las funciones marcadas con color naranja constituyen a procesos Javascript del fichero Sidebar.html, mientras que los métodos de color azul representan procesos del Código.gs.

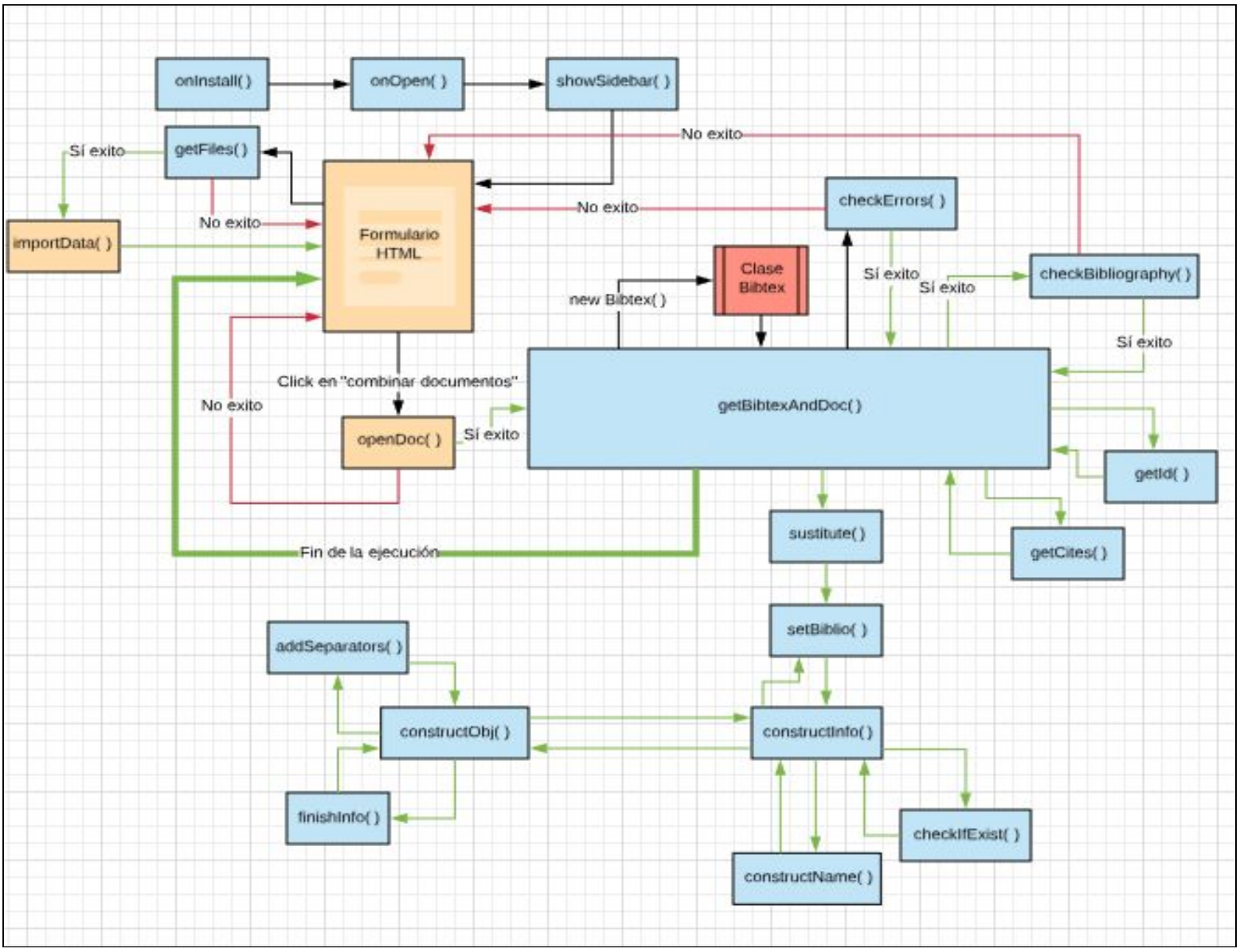



***Figura 10**: Esquema del árbol de funciones del complemento desarrollado.*

Como se puede observar en el esquema, el número de funciones es considerablemente elevado, es por ello que a lo largo de esta sección solo se detallarán únicamente aquellos métodos que tienen una funcionalidad principal durante el proceso de ejecución de nuestro complemento.

La ejecución comienza con la función `onInstall()` situada en la esquina superior izquierda de la figura 10 y finaliza, en caso de que haya éxito, con la función `getBibtexAndDoc()` situada en el centro de la misma figura.

## 4.1. onInstall( )

Es la función base de la que se compone todo Complemento de Google que, a través de la función auxiliar `onOpen()`, lanza la interfaz de usuario (HTML) que el cliente observa cuando este inicia el complemento.

Esta interfaz de usuario le proporcionará al cliente la posibilidad de seleccionar un archivo BibTeX de entre todos sus documentos de Google Drive y la opción de seleccionar un tipo de estilo bibliográfico (por defecto *unsrt*) que se aplicará a la sección de referencias. Internamente, la aplicación creará un desplegable con todos los documentos que el usuario tiene almacenados en su Google Drive, a través de dos funciones llamadas `getFiles()` e `importData()`.

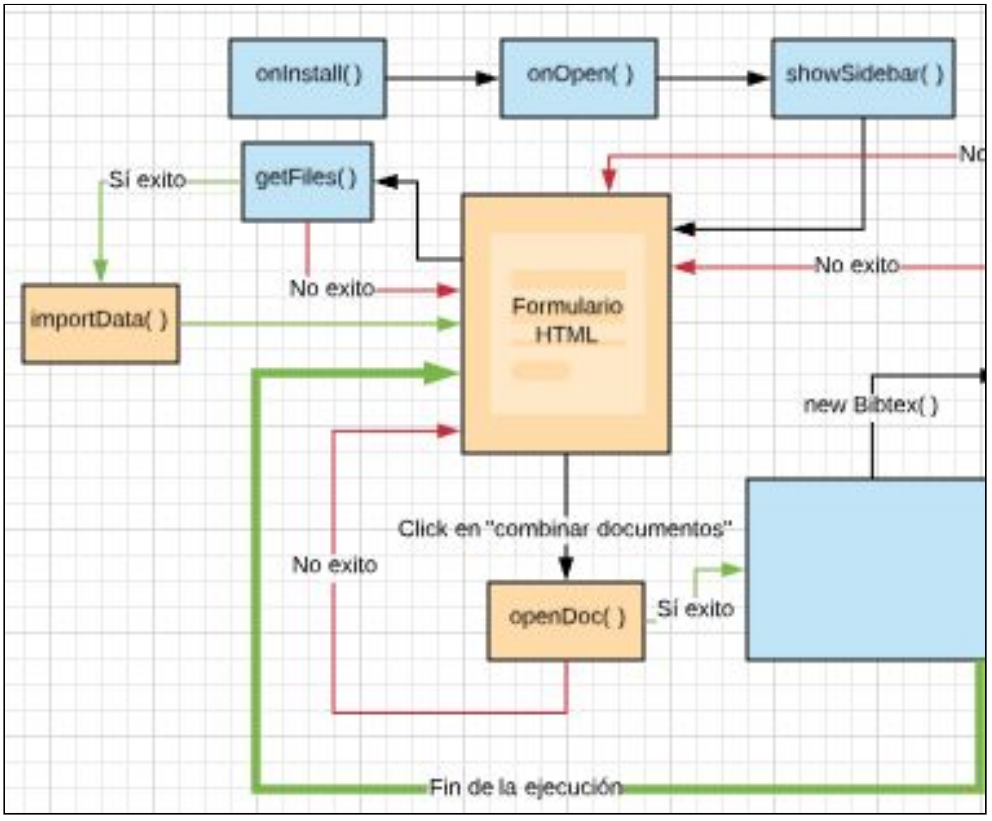



***Figura 11**: Esquema de las funciones que se centran en elaborar la interfaz de usuario.*

Cuando el usuario pulsa el botón de "**combinar documentos**", habiendo seleccionado un archivo BibTeX previamente, la ejecución de nuestro complemento se iniciará comenzando por la función `getBibtexAndDoc()`. Si el usuario ha seleccionado un archivo que no sea BibTeX, el complemento le notificará indicandole que a ocurrido un error, tal y como se muestra en la figura 25 de la sección 7.

## 4.2. getBibtexAndDoc( )

A partir de esta función continuará la ejecución la cual dependerá de varios factores importantes para lograr que acabe con éxito.

`getBibtexAndDoc()` es el método base de la que parten el resto, además de tener asignada las tareas principales en la ejecución. Una de esas tareas es la de elaborar el objeto que contendrá toda la información proporcionada por el archivo BibTeX. Este objeto se crea a partir de una nueva clase llamada `Bibtex` que se elaboró en gran medida gracias al código encontrado en un repositorio de Github [5] durante la investigación de LaTeX. Consta de una serie de funciones auxiliares elaboradas con el fin de factorizar el código, además de incluir una larga lista de propiedades que se van configurando a medida que avanza la ejecución. A partir de la clase `Bibtex` se construye el elemento final que contendrá todas las entradas del fichero BibTeX proporcionado en el formulario.

En un inicio, la clase BibTeX únicamente tenía en cuenta el tipo de entrada bibliográfica *article*, es decir, si en el fichero .bib existía alguna entrada de otro tipo (*book*, *techreport…*), la trataba como si fuese un artículo. Además, solo identificaba cuatro campos dentro de las entradas del archivo (*author*, *year*, *journal* y *title*) mostrándolos separados por comas y con un espacio en blanco en caso de que no existiesen. No obstante, el procesado del archivo BibTeX lo realizaba en su mayoría de forma correcta identificando el inicio y final de una entrada, almacenando cada campo como propiedad de un objeto y añadiendo como propiedades finales el tipo y el identificador de la entrada. Además, en lo que respecta al campo *author,* realizaba un procesado especial formando un array con todos los autores donde cada posición es un objeto con una propiedad por cada elemento del nombre del autor (*first* para nombre, *last* apellidos, *von* para una partícula y *junior* para un sufijo). Sin embargo, el manejo y tratado de este campo no era del todo correcto pues no tenía en consideración todos los casos posibles y no detectaba adecuadamente las posibles acentuaciones que los nombres de autor podrían tener. El formato del campo *author* no resulta del todo sencillo pues existe una amplia gama de posibles formas de escribirlo, es por ello que se utilizó el libro online ***Names in BibTEX and MlBibTEX*** [4] para el estudio de este formato.

No obstante, tras un largo proceso de entender cómo funciona el código obtenido del repositorio GitHub, se trató de corregir los errores presentes y orientarlo para que tratase los diferentes tipos de entradas junto a sus campos.

Una vez que el objeto con toda la información del fichero BibTeX se ha creado, se comprueba si existe alguna entrada que le falte algún campo obligatorio. Para saber cuales son los campos obligatorio de cada tipo de entrada se utilizó la documentación proporcionada por la página web Bib-it [8].

En caso de éxito, a través de varios procesos, el programa identifica todas las marcas de citas presentes en el documento para finalizar con la llamada a la función `sustitute( )`, encargada de crear y elaborar la bibliografía.

## 4.3. sustitute( )

Su finalidad principal es la de crear la nueva sección de referencias a partir de la información extraída del archivo BibTeX y de la lista de citas que fueron identificadas en el documento. En un primer lugar, la función filtra todas aquellas citas que no estén presentes en el archivo BibTeX, con el fin de añadir a la bibliografía un nuevo apartado informativo que avise al usuario que existen claves que no se han encontrado, sustituyendolas por [??].

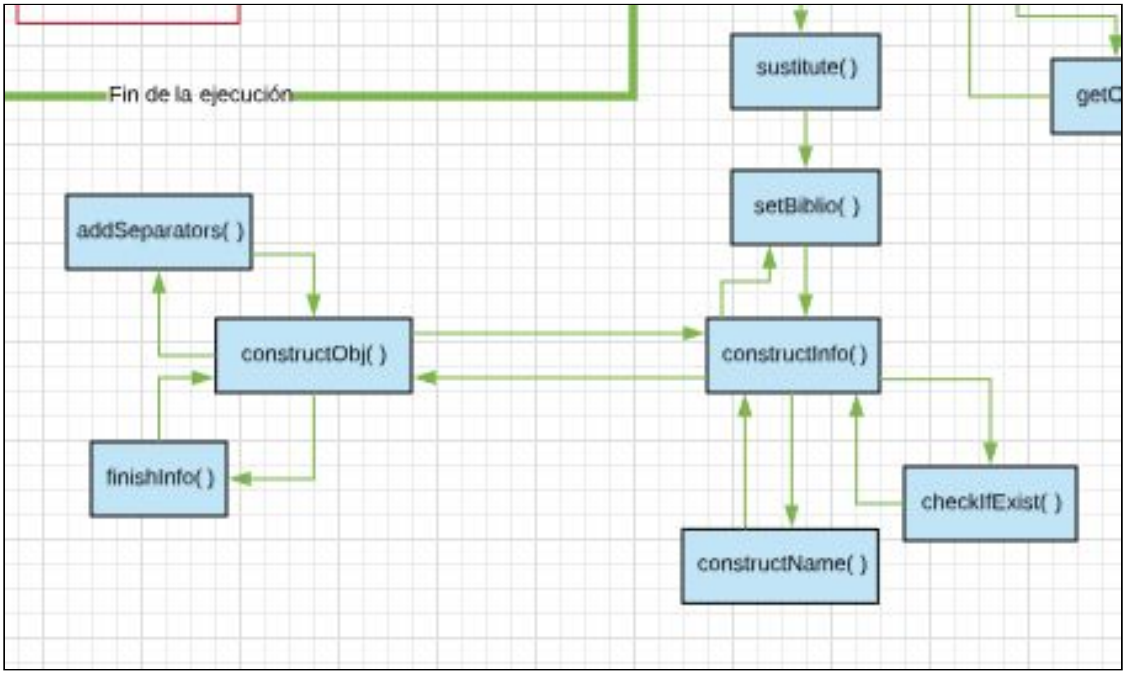



***Figura 12**: Esquema de las funciones que se centran en elaborar la nueva sección de referencias.*

A continuación, la función comienza elaborando la sección bibliográfica identificando el lugar donde se debe colocar a través de la etiqueta `\bibliography`, delegando esta tarea a una subfunción llamada `setBiblio()`. En caso de no existir esta marca en alguna parte del documento, la aplicación devolverá un mensaje de error informando al usuario de lo ocurrido. Para localizar la marca o etiqueta, se hace uso de un método que la clase `Body` proporciona, llamado a `findText()`. Esta función recibe por parámetro la cadena de texto que se desea buscar y devuelve como resultado un elemento de la clase `RangeElement`, que indica la posición del texto buscado.

Una vez conocido el lugar en el que se debe comenzar a añadir la sección de referencias, se añade un nuevo párrafo (clase Paragraph) con el título de la sección. Con el fin de que la función no contenga un elevado número de líneas de código, la tarea de completar la bibliografía es encomendada a la función `constructInfo()`.

Antes de comenzar con el cuerpo de la bibliografía, esta nueva función estructura una tabla (clase `Table`) que contendrá dos columnas principales: la notación de referencia y la información correspondiente del documento referenciado. En función del estilo seleccionado, se utilizará un tipo de notación y se aplicará el formato específico a la información del documento. El estilo *unsrt* se caracteriza por utilizar una notación numérica tal y como se ha mencionado en la sección 3.3.2. En caso de que una misma cita aparezca varias veces en el documento se utiliza el mismo identificador numérico evitando que la misma entrada esté repetida en la sección de referencias. En la segunda columna, los campos de cada entrada se colocarán siguiendo un orden específico, y aplicandoles el formato que caracteriza al estilo *unsrt*. Para el caso del estilo *unsrt*, se utiliza el mismo orden en el que aparecen las marcas de citas en el documento.

La información se construye a partir de un objeto al que se le van añadiendo los diferentes campos como una nueva propiedad del mismo, añadiendoles los elementos de separación. Finalmente cada propiedad del objeto pasa un proceso de filtrado para identificar posibles acentuaciones o símbolos que deban ser eliminados.

A partir del objeto anterior, `constructInfo()` elabora un párrafo utilizando la clase `Paragraph` que servirá de base para formar las celdas de la tabla.

# 5. Problemas surgidos

Durante el desarrollo e implementación del complemento han surgido ciertos problemas que se han tenido que solventar para continuar con su desarrollo.

La elaboración de tutoriales ha supuesto un avance en el estudio de la estructura y utilización de los complementos, sin embargo, estos tutoriales no aportan conocimientos adicionales del funcionamiento interno de los mismos. Por ejemplo, el tutorial que se utilizó como base para la futura elaboración permitió conocer qué archivos son necesarios para el funcionamiento del complemento. No obstante, en ninguna parte del tutorial se explican las clases y funciones utilizadas, ni por qué se utilizan ciertos métodos y no otros. Esta falta de documentación, unida a la ausencia de ejemplos de uso en la mayoría de métodos de las clases de Google Apps Script, ha supuesto un retraso considerable restando tiempo a su desarrollo, pues aprender un nuevo lenguajes sin ejemplos no siempre resulta sencillo.

Mientras se estaba elaborando la sección de referencias, se realizó una investigación de los diferentes estilos bibliográficos para conocer cómo influyen estos en el formato y orden. Sin embargo, no se encontró una documentación como tal, en la que se detalla cómo se debería elaborar la bibliografía según el estilo y el tipo de entrada. De ese modo, la elaboración del estilo *unsrt* se tuvo que hacer en base a algunos ejemplos encontrados y al sentido común, pues no fue posible encontrar todos los casos posibles en el que el estilo y el tipo de entrada influyen en el formato de los campos.

Por ejemplo, imaginemos que tenemos una entrada del tipo *article*, con los campos opcionales *volume, number* y *pages*. Tras aplicar el estilo *unsrt*, estos campos se mostrarían de la siguiente forma: 19(3):215-216 (donde el 19 es el campo *volume*, el 3 el campo *number* y 215-216 el campo *pages*). Pero, ¿cómo se mostrarían estos campos si faltase uno de ellos puesto que son opcionales? La respuesta a esta pregunta no viene documentada en ninguna parte, lo que lleva a seguir al sentido común para solucionarlo.

La mayor parte de las personas que utilizan LaTeX, hacen uso de él sin conocer la forma en la que este editor de texto transforma la información, pues LaTeX realiza este mecanismo de manera transparente al cliente. Es por eso que el problema de todo esto proviene de la falta de una documentación completa en la que se expliquen los diferentes conceptos que LaTeX utiliza para elaborar una bibliografía:

- Los diferentes tipos de estilos.
- Los distintos tipos de entradas.
- Cómo influyen en el formato y orden el tipo de estilo y entrada.
- Cómo se vería una bibliografía con diferentes tipos de entradas para cada estilo, proporcionando para ello una lista de ejemplos.

# 6. Tutorial de instalación y uso

Habitualmente los complementos que normalmente se utilizan, están disponibles en la amplia lista de complementos de Internet. Los propios complementos que un desarrollador elabora, se pueden incluir a esta lista siguiendo una serie de pasos que los desarrolladores de Google Apps Script proporcionan a través de un tutorial disponible en su página oficial [6]. No obstante, este proceso no siempre resulta sencillo. Además, durante este procedimiento, el complemento debe pasar, entre otras cosas, una fase de validación que en ocasiones puede llevar un largo tiempo. Es por todo ello, que se decidió no llevar a cabo dicho proceso de publicación para nuestro complemento, lo que supone que se debe instalar de manera manual para poder utilizarlo.

De esta manera, en esta sección presentaremos una breve guía con los pasos a seguir para su instalación y uso.

1. En el documento que desees usar el complemento, deberás seleccionar el menú de "*Herramientas*" que se encuentra en la parte superior. Se abrirá un desplegable con varias opciones, a continuación selecciona "*Editor de secuencias de comandos*" (figura 13).

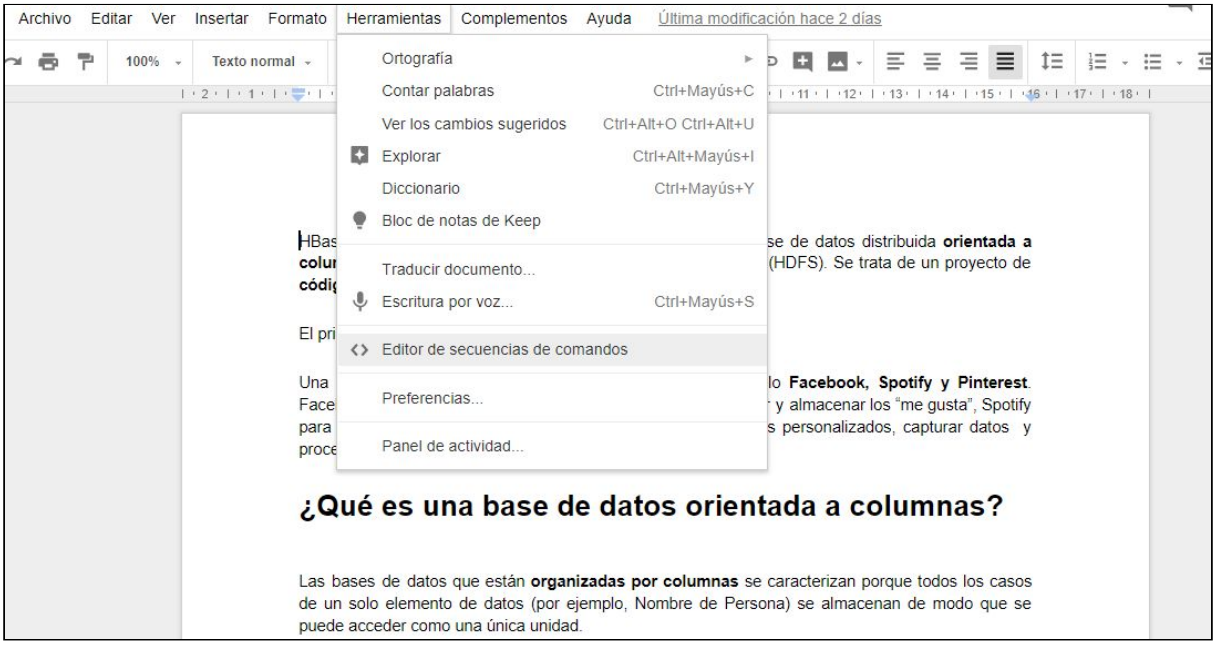



***Figura 13**: Configuración del complemento.*

2. A continuación se abrirá una nueva ventana tal y como se muestra en la figura 14. Esta ventana muestra un primer vistazo al archivo `Código.gs`. Como ya hemos mencionado anteriormente, este fichero será el *script* encargado de las funciones principales. Para incluir la interfaz de usuario debemos añadir un archivo HTML. Para ello, seleccionaremos en el menú "*Archivo*" situado en la izquierda superior de la ventana. A continuación escogeremos la opción "*Nuevo*" y posteriormente "*Archivo HTML*" (figura 15).

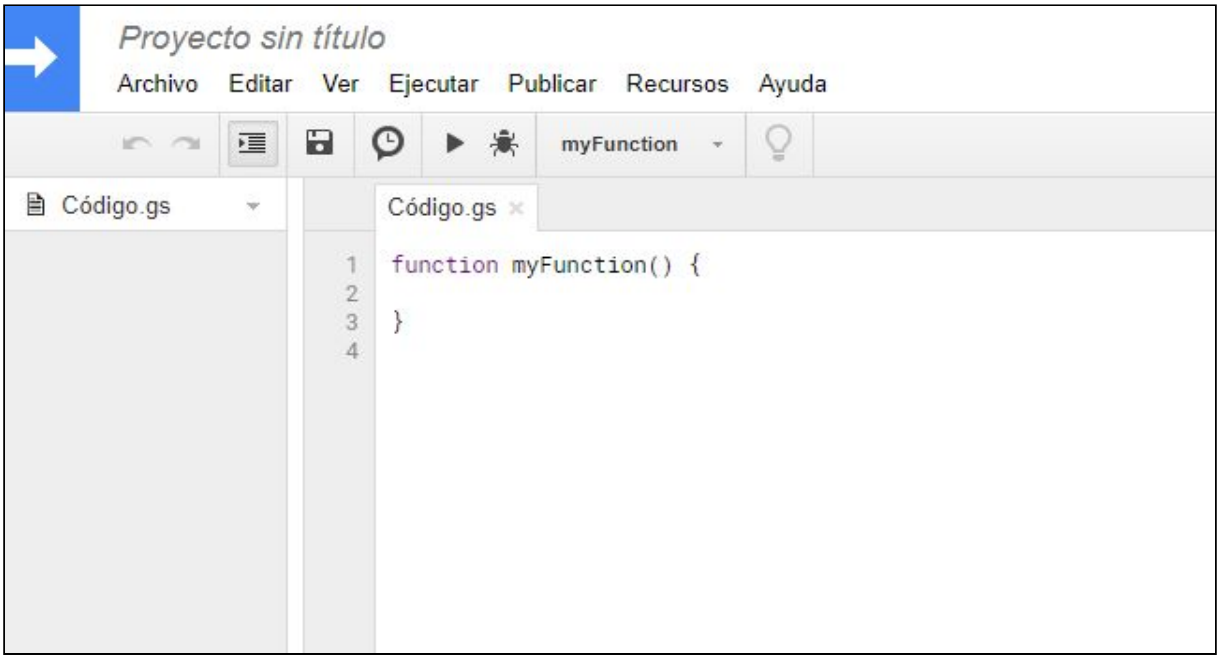



***Figura 14**: Vista de la configuración de un complemento.*

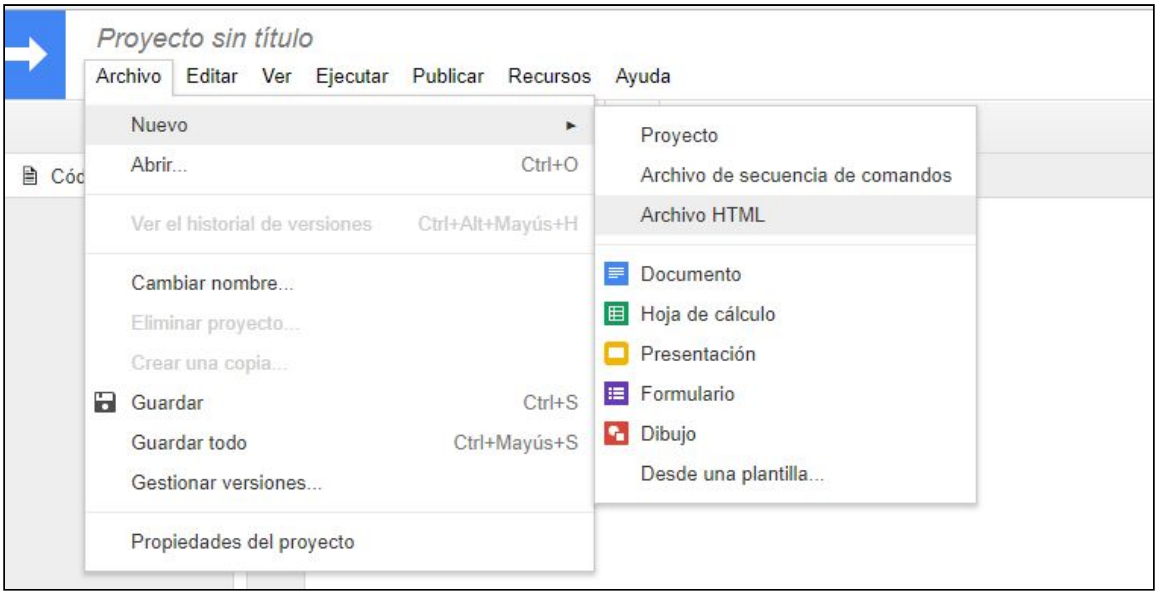



***Figura 15**: Creación de un archivo HTML.*

3. Tras esto, nos aparecerá un mensaje indicándonos que debemos asignarle un nombre a nuestro nuevo archivo HTML. Es importante que el nombre de este archivo sea exactamente "*Sidebar*", incluyendo la mayúscula (figura 16).

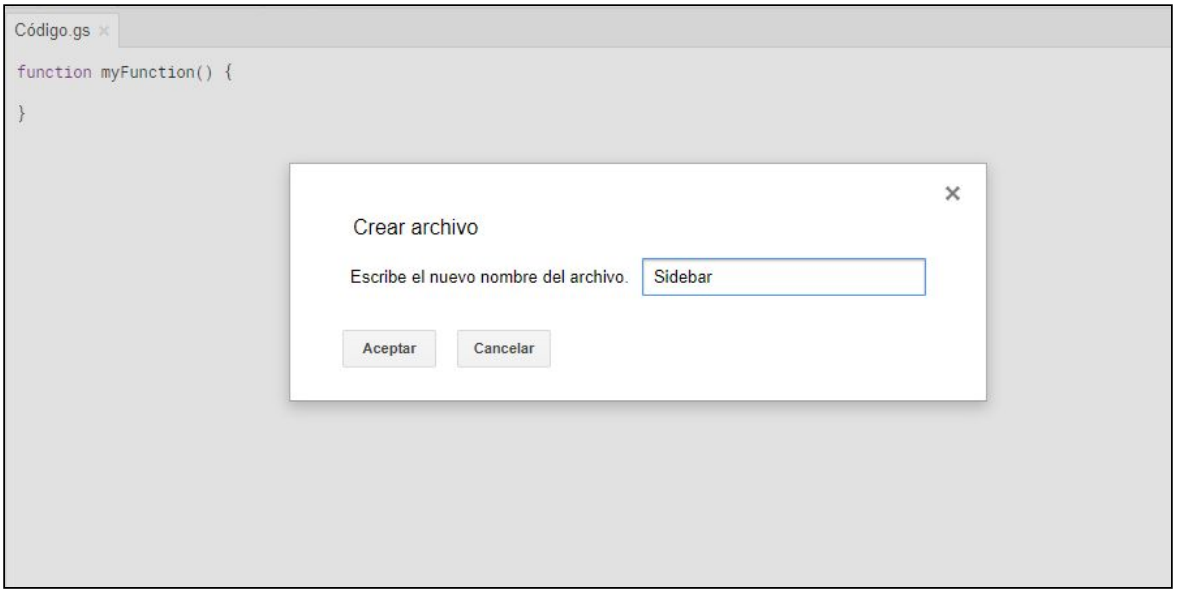



***Figura 16**: Creación de un archivo HTML.*

4. Como resultado veremos que nuestro nuevo archivo `Sidebar.html` se ha añadido en la sección de la izquierda junto al fichero Codigo.gs como se ve en la figura 17. Deberemos eliminar el contenido que se crea por defecto en los dos archivos, dejándolos en blanco. Ahora, deberemos copiar el contenido de los archivos con el mismo nombre que se encuentran en el Github del proyecto (https://github.com/andreamartin96/TFG). Para extraer el código de cada archivo, accedemos a cada uno de ellos y seleccionamos el botón Raw de la derecha como se muestra en las figuras 18, 19 y 20. Una vez hemos copiado los códigos, salvaremos el proyecto de la misma forma que se muestra en la figura 21.

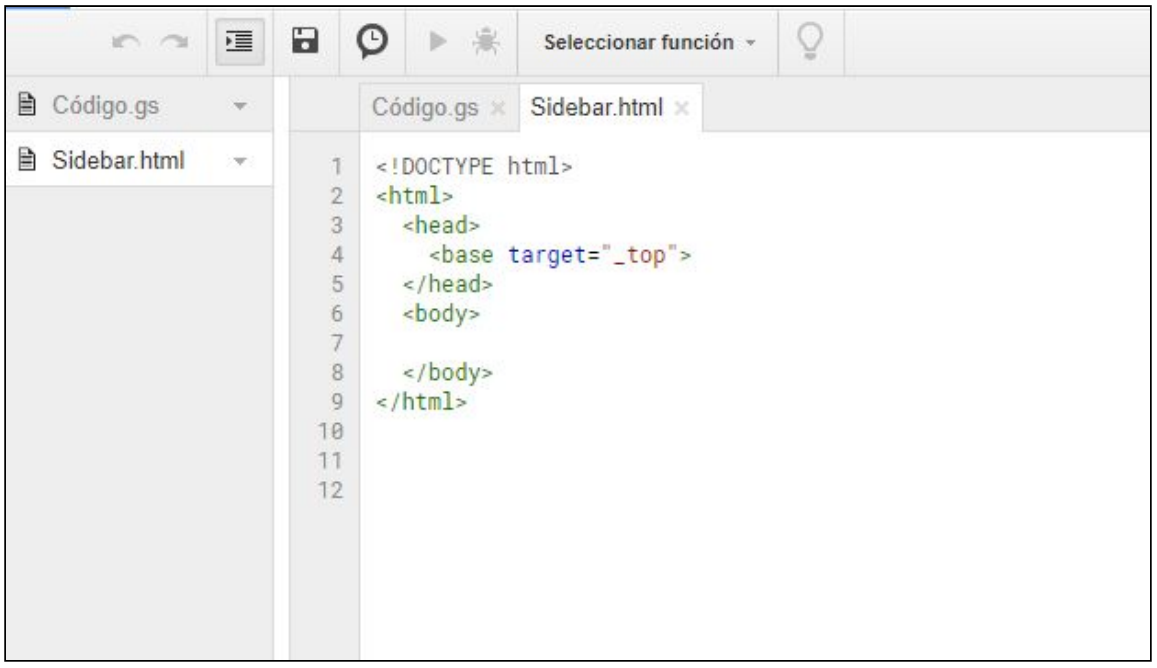



***Figura 17**: Configuración del complemento*

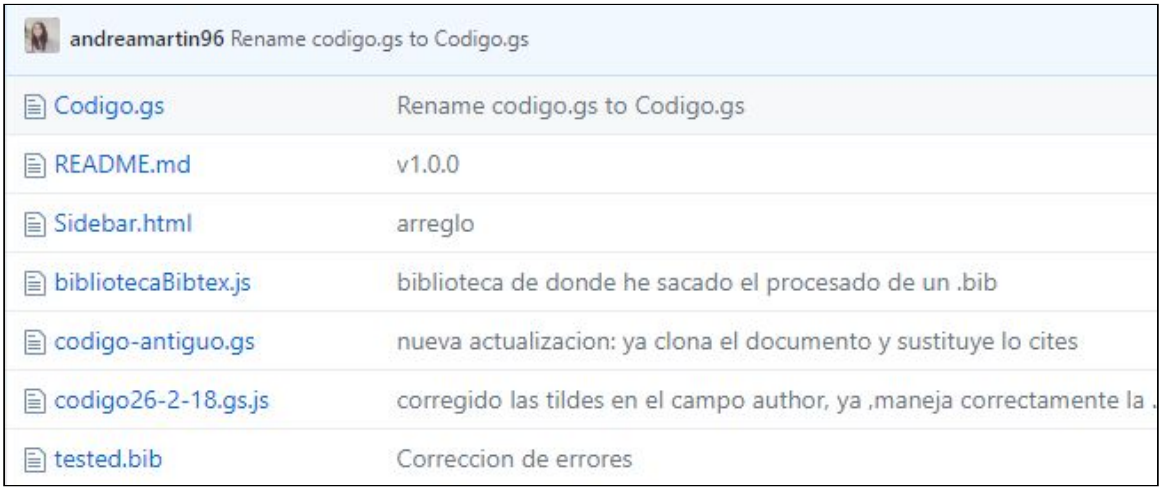



***Figura 18**: GitHub del proyecto.*

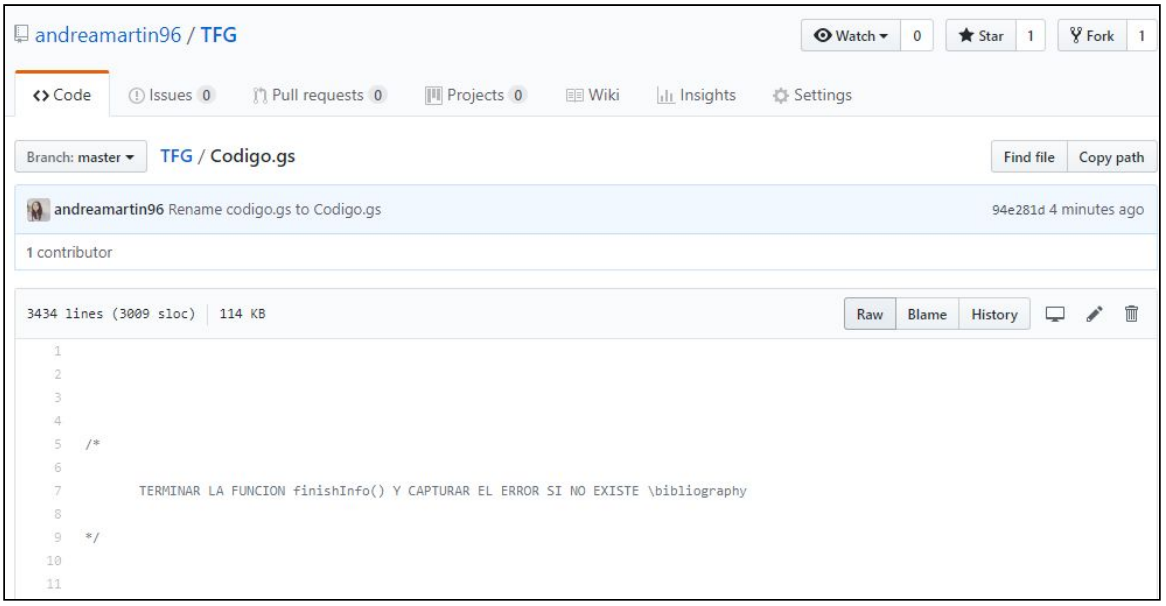



***Figura 19**: Archivo Codigo.gs del proyecto disponible en Github.*

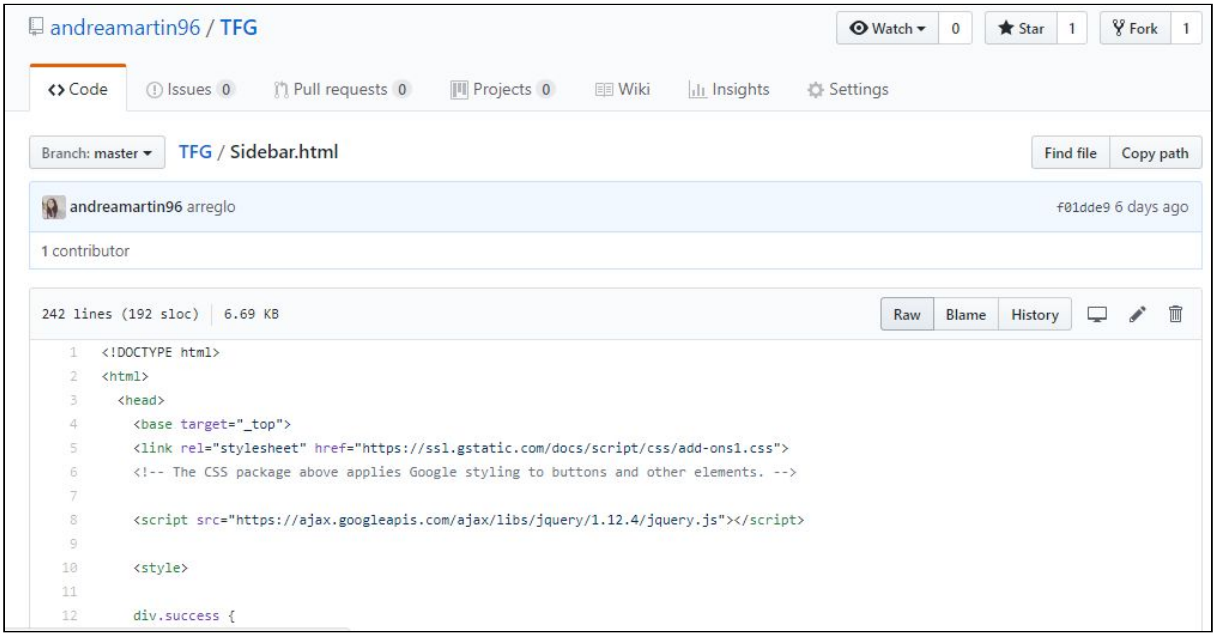



***Figura 20**: Archivo Sidebar.html del proyecto disponible en Github.*

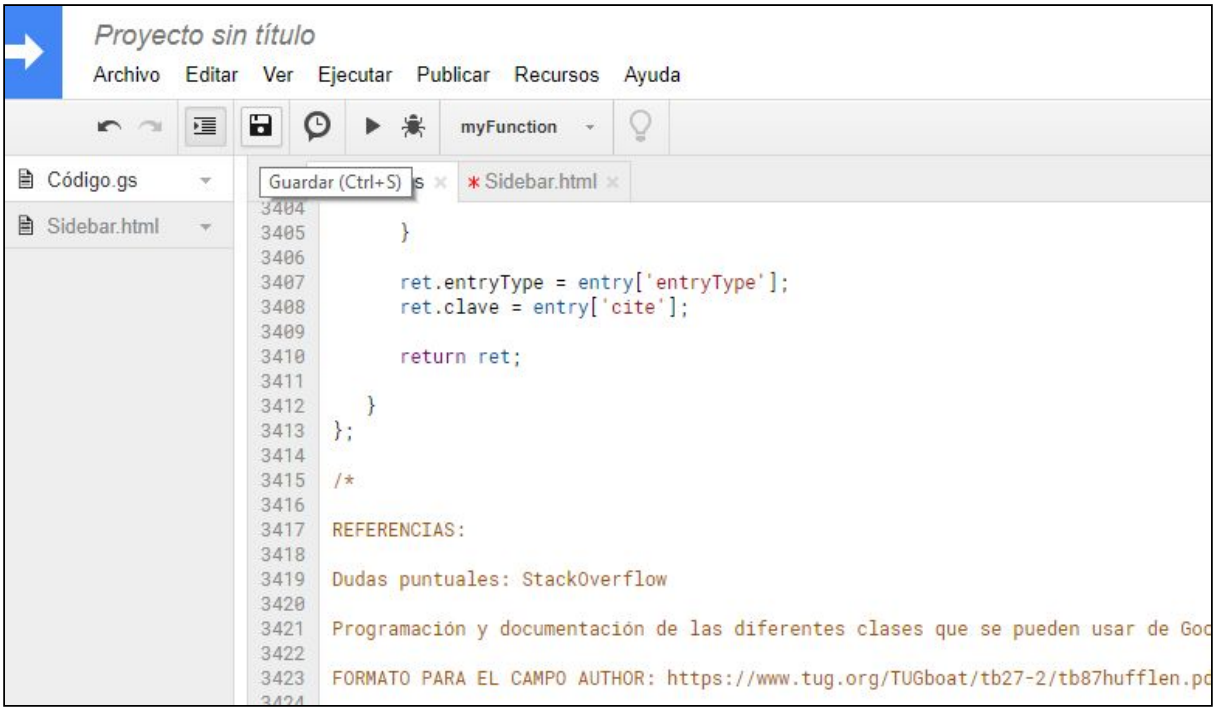



***Figura 21**: Guardar cambios en un complemento.*

5. Cuando lo guardemos, se nos mostrará una ventana en la que se nos pregunta el nombre que se le asignará al proyecto (figuras 22). Podemos llamarlo como queramos. Una vez guardado, el complemento estará listo para usarse, pero antes deberemos compilar los cambios utilizando “Ctrl F5”.

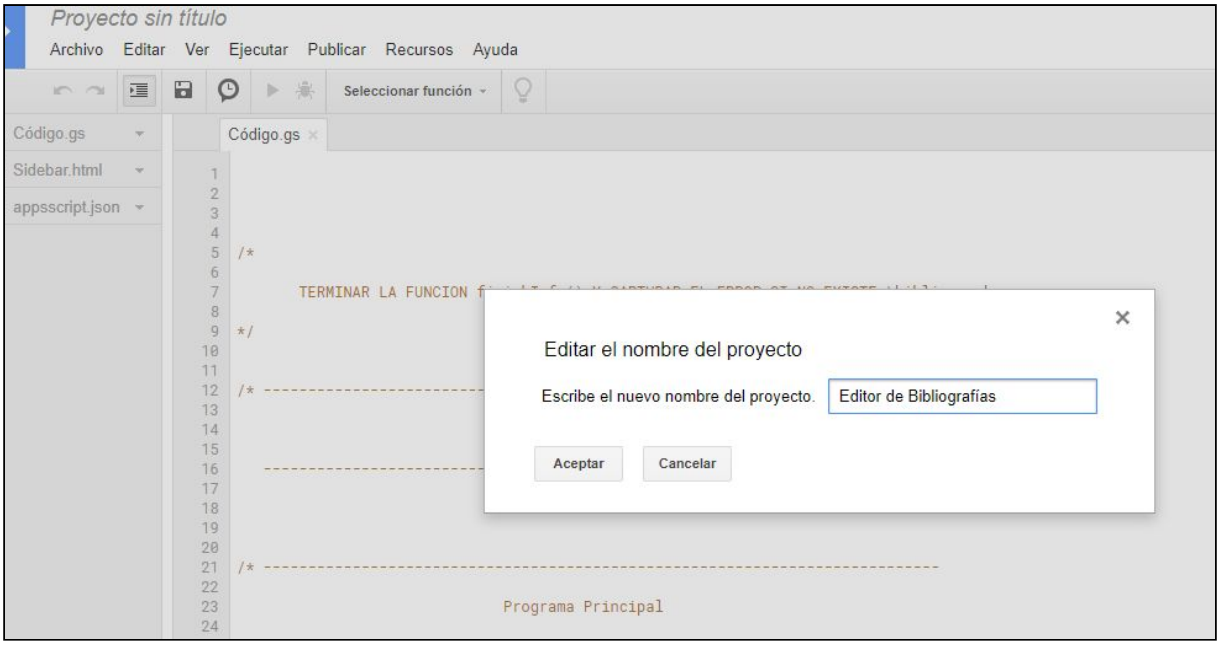



***Figura 22**: Renombrar el proyecto de un complemento.*

6. Para utilizarlo, lo primero de todo es escribir las marcas de citas de aquellos documentos que queramos referenciar, añadiendo la etiqueta `\bibliography` en el lugar donde queremos que se cree la nueva sección. Una vez que el documento está completo y disponemos del fichero BibTeX con las entradas correspondientes, cargaremos el complemento seleccionando el menú superior "Complementos" y posteriormente escogiendo el proyecto cuyo nombre fue asignado en el paso anterior (figura 23).

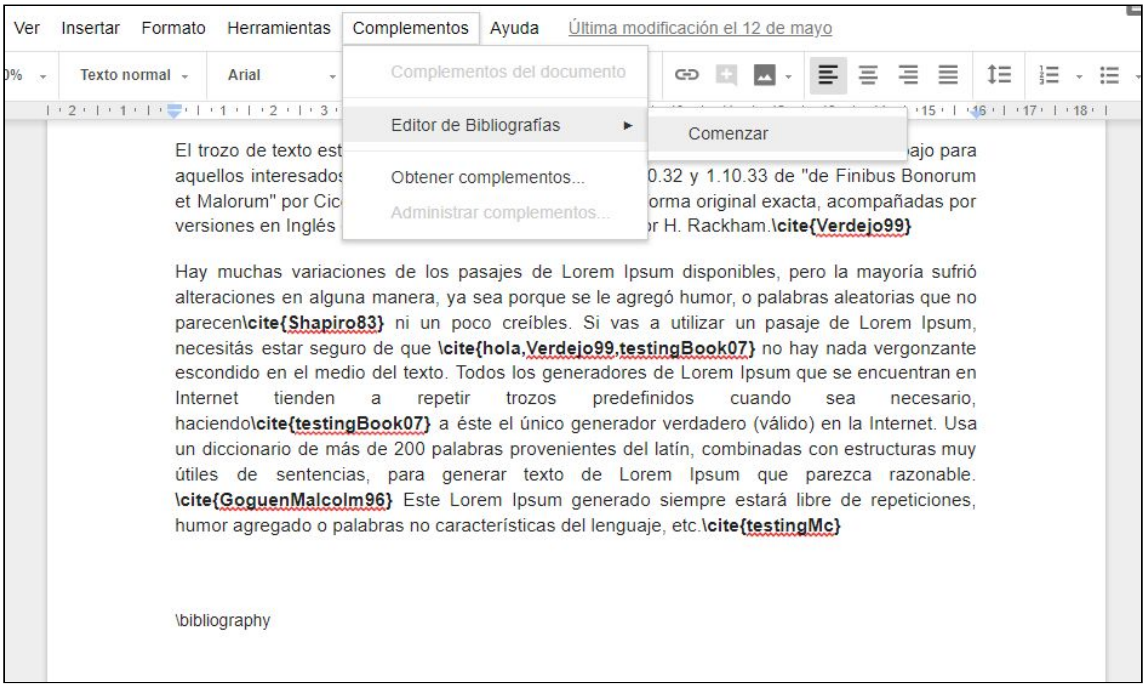



***Figura 23**: Abrir el nuevo complemento creado.*

7. A continuación nos aparecerá una nueva ventana en la parte derecha como se muestra en la figura 24. Deberemos seleccionar el archivo BibTeX donde se nos indique. Cuando todo esté listo, seleccionaremos el botón inferior de "Combinar documentos". Si todo va bien[1], en el lado inferior de esta ventana, aparecerá un mensaje en verde que nos informará de que se ha creado un nuevo documento en nuestro Drive con nombre "New Document". En este nuevo documento se habrá creado la sección de referencias y las citas bibliográficas habrán sido sustituidas.

[1] Para que el resultado ses un éxito, debe existir en el documento la marca `\bibliography`. Además, en el archivo BibTeX seleccionado todas las entradas deben tener todos los campos obligatorios correspondientes según el tipo de entrada.

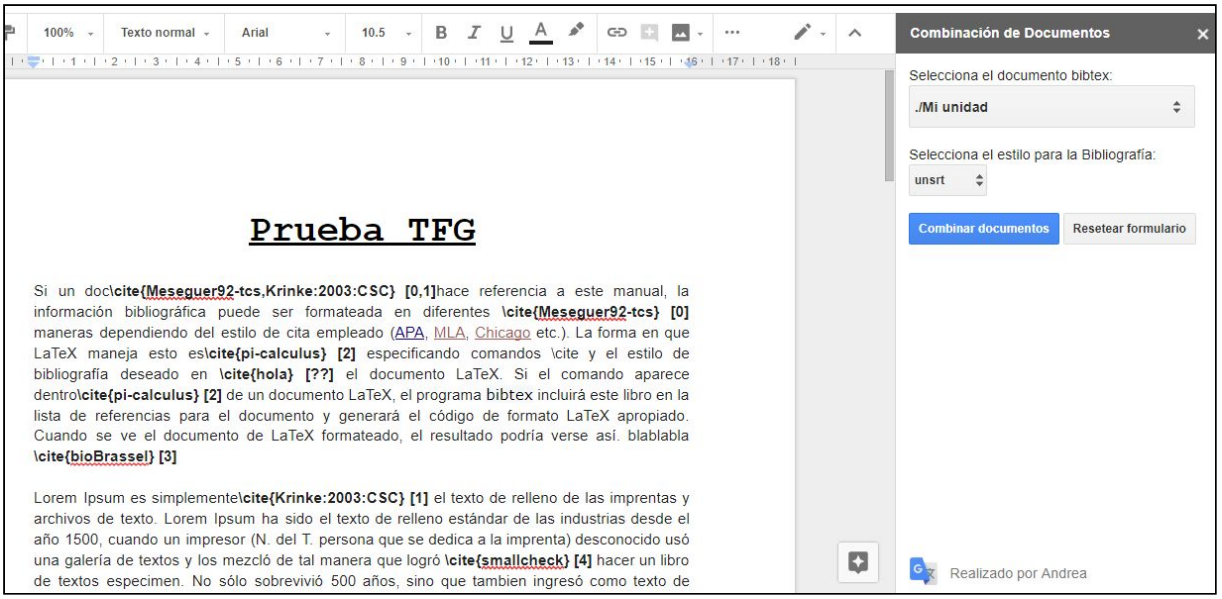



***Figura 24**: Documento de Google con el complemento activo.*

# 7. Conclusiones y trabajo futuro

## 7.1. Conclusiones

Con el desarrollo de este proyecto se ha elaborado un nuevo complemento para los documentos de Google, el cual otorga la posibilidad de crear bibliografías de manera sencilla y rápida a través de una interfaz de usuario funcional elaborada por medio de Google Apps Script.

De esta forma, hemos logrado obtener conocimientos en materias y campos que se desconocían en un primer momento. El hecho de aprender un nuevo lenguaje ha supuesto un reto, pues los lenguajes programación que hemos asimilado durante la carrera han sido impartidos en todo momento por profesores expertos en esto campos, sin embargo, para aprender Google Apps Script hemos tenido que hacerlo por nuestra propia cuenta, ayudándonos de tutoriales y guías para lograr entenderlo.

Además, también nos ha ayudado a descubrir cómo de compleja resulta la sintaxis y semántica de LaTeX. El funcionamiento de este editor no siempre resulta sencillo de comprender, es por ello que a través de nuestro proyecto, los aspectos comunes con LaTeX son manejados de manera transparente al usuario, evitando cualquier tipo de dificultad en el uso del complemento, pues conociendo los dos tipos de marcas que se deben utilizar, se puede lograr un uso rápido del mismo.

## 7.2. Trabajo futuro

Existen formas en la que este complemento se puede mejorar para lograr una ejecución más eficiente o proporcionar al usuario una gama más amplia de posibilidades. Estableceremos una lista en la que destacaremos las formas de mejorar el programa y cómo se debería planear su desarrollo.

De cara al cliente, una forma de extender el complemento desarrollado sería añadiendo nuevos estilos con los que elaborar su bibliografía. Existe una larga lista de estilos bibliográficos habitualmente utilizados: *abbrv, alpha, acm, apalike, plain*, etc. La mejor forma de implementar estos estilos es hacerlo de forma de similar a como se ha implementado el estilo *unsrt*, cambiando el formato de los campos y la forma en la que se estructura la notación de referencias.

De igual manera, también es posible perfeccionar los tipos de entradas que el programa identifica. El complemento está diseñado para identificar los tipos de entradas que se mencionan en la sección 3.3.1., no obstante, existen otro tipos de entradas que el complemento no identifica ignorándolas por completo (*inbook, booklet, incollection, manual* y *unpublished*). La forma de añadir estos nuevos tipos es incluyendo nuevos

casos en el método `google()` de la clase Bibtex, incluyendo aquellos campos obligatorios en la función que se añadió (el método `checkRequieredFields()` de la clase Bibtex), la cual comprueba que existan todos ellos devolviendo error en caso de que falte algún campo. Tampoco hay que olvidarse de añadir los posible campos opcionales que puede incluir cada entrada. Por otra parte, también será necesario añadir el caso base para cada entrada, en el *switch* de la función `constructObj()`, a partir del cual se construirá el objeto final con toda la información ordenada y con el formato deseado.En lo que respecta al código, sería posible factorizar el código, reutilizando la estructura de algunos de los procesos para elaborar los nuevos casos. De esta forma, se evitará escribir varias veces ciertas líneas de código que presenten una funcionalidad semejante.

No siempre la ejecución resulta un éxito, si se dan ciertos factores se pueden producir errores que pueden llevar a un final inadecuado. Es por ello, que el complemento desarrollado en este proyecto, contempla la posibilidad de que ocurran ciertos errores durante la ejecución: *no se ha seleccionado un fichero BibTeX (figura 25), falta algún campo obligatorio en alguna entrada (figura 26), no existe la marca* `\bibliography` *en el documento (figura 26), etc*.

El principal problema cuando se informa al usuario de que falta algún campo obligatorio es que no se indica qué entrada del archivo BibTeX ha causado dicho error. Como trabajo futuro, una posible idea para solucionarlo sería elaborar una lista que contenga todas las entradas a las que les falta algún campo obligatorio. Esta lista sería devuelta al archivo HTML tras finalizar la ejecución del *script*, para elaborar el mensaje de error incluyendo la lista de entradas completa. De esta forma, el usuario podría reconocer inmediatamente lo ocurrido durante la ejecución, conociendo todos los errores presentes en el documento BibTeX.

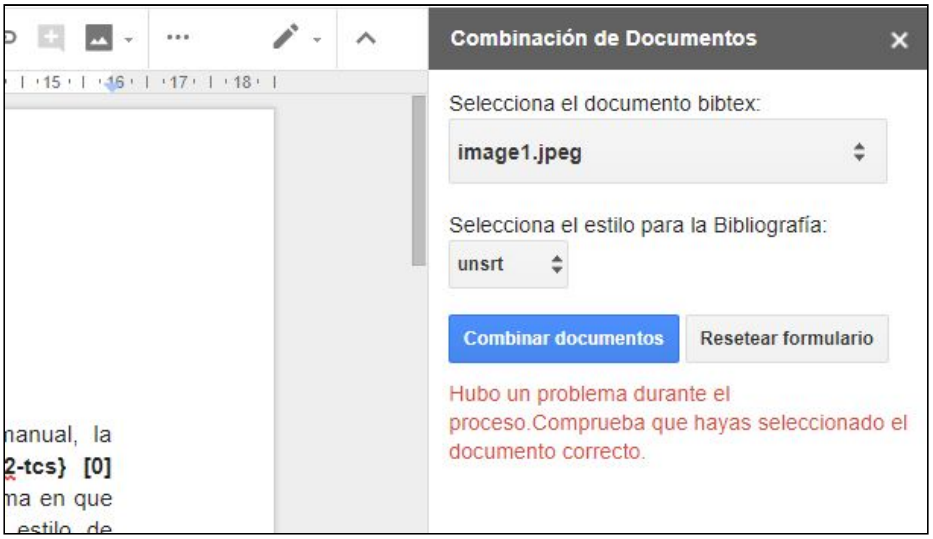



***Figura 25**: Ejemplo de error.*

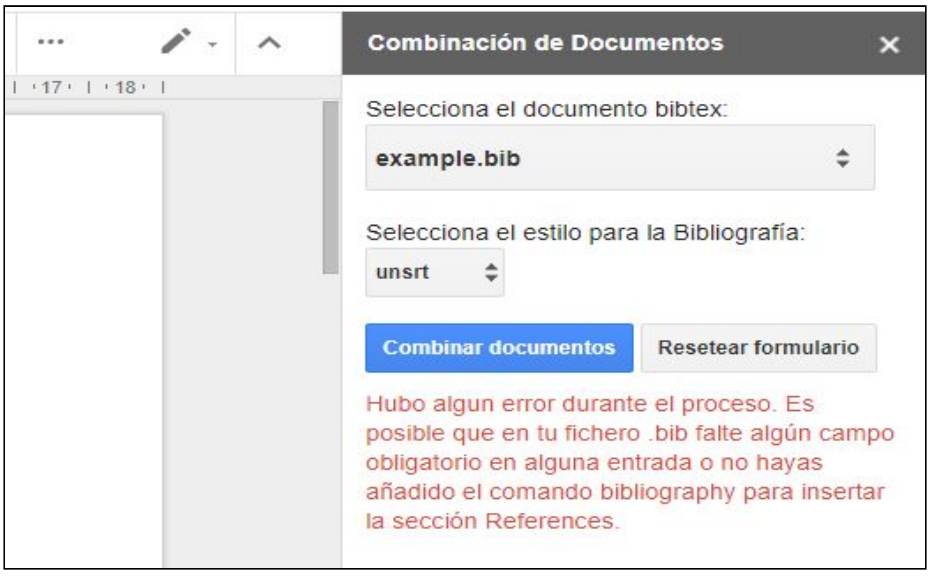



***Figura 26**: Ejemplo de error.*

# 8. Conclusions and Future Work

## 8.1 Conclusions

The development of this project has led to the creation of a new Add-on for the Google Documents service that is able to generate reference sections automatically and in an efficient way, through an appealing user interface built with the tools that Google Apps Script provides.

Thus, the process has gone through the acquisition of knowledge and skills in a field that was initially unknown for us. Learning to use this new programming language has proven to be a challenge. We are accustomed to be taught by expert professors but in this case, their guidance was not available, and we had to go through the learning curve by ourselves, by the means of online tutorials and forum guides.

Moreover, we have discovered the complexity that the LaTeX syntax and semantics entail. The procedures of the LaTeX editor are not always easy to understand, and that is the reason why we have opted to simplify the user interaction. Our Add-on does not require any skill from the users. To use it, they will only need to know the two reference marks that are available:

## 8.2 Future work

There are some ways our Add-on could be improved to achieve a more efficient execution or to provide the user with a wider range of capabilities. In this section, we will go on to mention some of our project's shortcomings, and how they could be addressed in future work.

When it comes to user options, it is true that our Add-on could include more styles to create the reference section. Currently, there is a long list of bibliographic styles that are used: *abbrv, alpha, acm, apalike, plain,* etc. The best way to implement these styles would be to do it in a similar way as the *unstr* style has been implemented, changing the format of the different fields and the way the reference notation is structured.

Similarly, it would be possible to polish the inputs the program can identify. As of now, our Add-on is designed to recognize the types of inputs that are mentioned in section 3.3.1. Nevertheless, there are other kinds of inputs that the Add-on will not identify, and that will actually be completely ignored (*inbook, booklet, incollection, manual*, and *unpublished*). One way we could include these new types would be to address new cases in the "google()" method of the BibTeX class, adding the necessary

fields in the function that was included (in the checkRequiredFields() method of the BibTeX class), which checks for every field and returns an error in case any one of them is missing. As a side note, it should not be forgotten that the optional fields that the input might include have to be added as well. Besides that, it is also necessary to include a base case for every input, in the *switch* that can be found in the constructObj() function, from which the final object will be built. This object will have all of its information ordered and will be formatted as desired. The code will be able to be factorized, reusing the structure of some processes to build the new cases, in order to avoid writing certain lines of code many times if they share similar functionality.

Sometimes, the execution might exit without success due to errors produced by certain factors. This is the reason why the Add-on that we have developed addresses the possibility that some errors can occur, such as “A BibTeX file was not selected” (Figure 27), “There are missing fields in an input” (Figure 28), “There is no `\bibliography` mark in the selected document” (Figure 28), etc.

The main issue is that, when the user is informed that there is a missing field, our Add-on does not show which BibTeX file input has caused the error. As a future improvement, this problem could be solved by elaborating a list that would contain every input that lacks a required field. This list would be returned to the HTML file after the script is run, to generate an error message that would include the complete input list. Thus, users would be able to understand what happened during the execution, and to identify every error found in the BibTeX document.

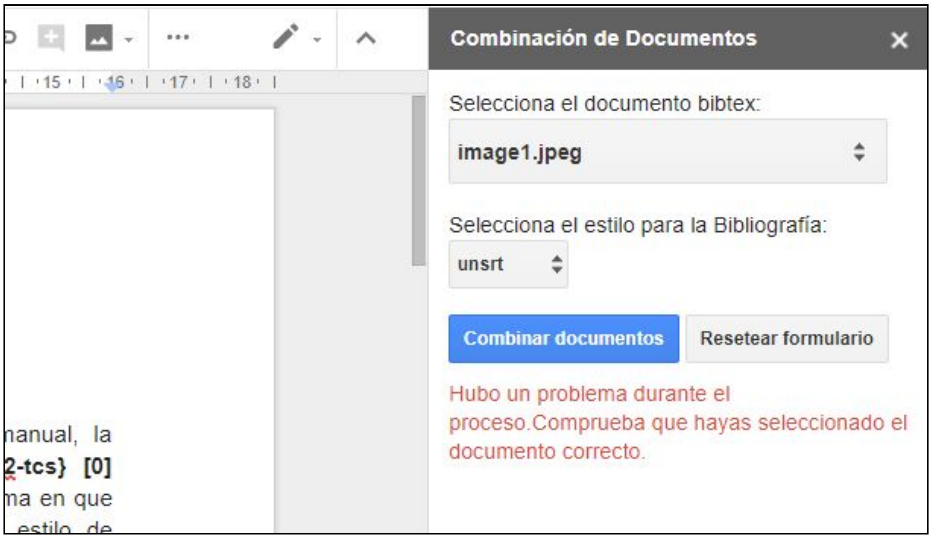



***Figure 27**: Example of an error.*

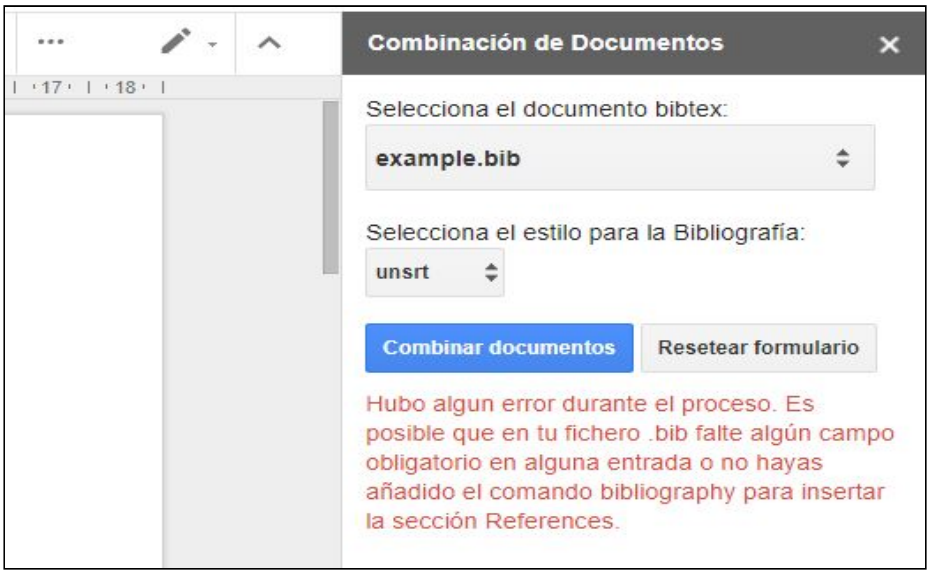



***Figure 28**: Example of an error.*

# 9. Bibliografía


**[1]** *Google Apps Script.* https://developers.google.com/apps-script/.

**[2]** *Quickstart: Add-on for Google Docs.* https://developers.google.com/apps-script/quickstart/docs.

**[3]** *Google Apps Script References.* https://developers.google.com/apps-script/reference/document/.

**[4]** Jean-Michel Hufflen. *Names in BibTEX and MlBibTEX.* LIFC (FRE CNRS 2661), University of Franche-Comté 16, route de Gray 25030 Besançon Cedex France, 2006.

**[5]** *Github de la versión beta.* https://github.com/select/google-docs-bibtex-cite.

**[6]** *Publicando un complemento.* https://developers.google.com/apps-script/add-ons/publish.

**[7]** *LaTeX.* https://es.wikipedia.org/wiki/LaTeX.

**[8]** *Bib-it.* http://bib-it.sourceforge.net/help/fieldsAndEntryTypes.php.