Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo de una heraamienta para la creación y despliegue de documentos

Development of a tool for creation and deployment of documents

Rudolf Cicko

La Laguna, 3 September 2017

D. **Casiano Rodríguez León**, con DNI 42020072S profesor Titular de Universidad adscrito al Departamento de informática de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Desarrollo de una herramienta para la creación y despliegue de documentos”

ha sido realizada bajo su dirección por D. **Rudolf Cicko**,

con N.I.E. X2060796-L.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 September 2017

Agradecimientos

Quiero agradecer principalmente a mi familia, a mi pareja, a mi tutor y a mis amigos más cercanos por apoyarme en todo momento durante la realización de este trabajo. Sin ellos esta aventura no sería tan divertida.

Licencia

Resumen

El objetivo de este trabajo ha sido el estudio de distintas herramientas para la creación de documentos y libros y el desarrollo de una herramienta que permite la creación y despliegue de documentos. También existe la posibilidad de crear documentos privados para restringir el contenido del mismo a un determinado grupo de personas a través de una autenticación Oauth. Finalmente se desarrolla una aplicación de escritorio multiplataforma que posee una interfaz elegante y cómoda para posibilitar el uso de la misma a todos los usuarios independiemente de sus conocimientos informáticos.

La herramienta está desarrollada en nodejs empleando un gran número de otros módulos como dependencias. Además, se ha construído una API sobre ésta para posibilitar a la aplicación de escritorio, la cual se ha desarrollado empleando las modernas tecnologías web del mercado, aprovechar todas las funcionalidades y con la intención de maximizar la usabilidad.

Palabras clave: react.js, electron, javascript, nodejs, gitbook

Abstract

The aim of this project has been the study of different tools for the creation of documents and books and the development of a tool that allows the creation and deployment of documents. There is also the possibility to create private documents to restrict the content of it to a certain group of people through Oauth authentication. Finally a multi-platform desktop application was developed that has an elegant and comfortable interface to enable the use of it to all users regardless of their computer skills.

The tool is developed in nodejs using a large number of other modules as dependencies. Also an API has been built on it to enable the desktop application, which has been developed using modern web technologies, take advantage of all the functionalities and with the intention of maximizing usability.

Keywords: react.js, electron, javascript, nodejs, gitbook

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Antecedentes

Cuando que apareció el ordenador personal muchos de los usuarios han pasado de usar el lápiz y papel a escribir todo en la computadora. Los ordenadores permitieron almacenar virtualmente textos, libros y otros documentos que antes sólo podías almacenar físicamente en los cajones de tu casa.

Con la aparición de internet mejoró aún más la cosa. Ahora los usuarios no sólo podían almacenar en su propio ordenador los documentos sino también podían compartirlo rápidamente con cualquier persona en cualquier parte del mundo en cuestión de segundos.

1.2 Estado actual del tema

En la actualidad existen varias herramientas que se pueden encontrar en internet con el propósito de crear documentos, apis, textos, libros, etc.. pero todas carecen de alguna funcionalidad que la haría muy útil. A continuación comentaré algunas de las herramientas encontradas por la red haciendo énfasis en las funcionalidades que ofrecen.

1.2.1 Book Creator

Book Creator es una aplicación disponible para la plataforma de android, IOS y Windows disponible en la PlayStore de Google.

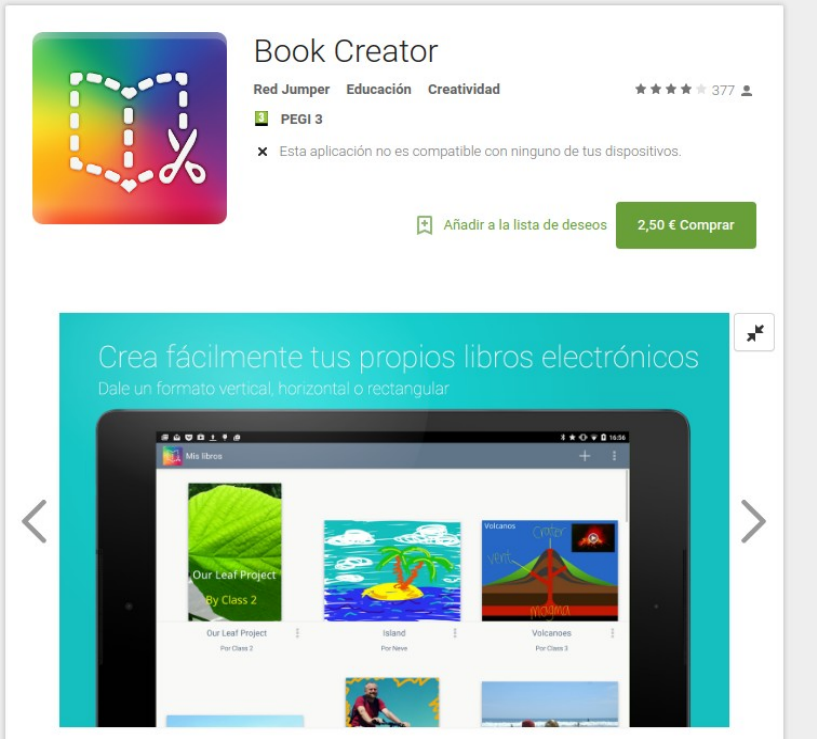

Figura 1.1: Book Creator

Es una herramienta orientada a profesores puesto que permite la inserción de contenido multimedia como videos, audios, fotos, etc.. El diseño del documento hay que hacerlo manualmente. Lo cual se vuelve lento y tedioso si el objetivo es hacer un documento profesional.

Book Creator está orientado principalmente a padres y profesores para crear documentos didactivos. También permite publicar los documentos creados pero solamente a Google Play Book. Si se desea publicar en alguna otra plataforma habrá que hacerse manualmente, lo cual requiere un determinado conocimiento informático.

1.2.2 iBooks Author

iBooks Author es otra herramienta que permite la creación de documentos mucho más profesional que la anterior. Sin embargo, solo está disponible para Apple y tampoco tiene la posibilidad de desplegarlo en el lugar que deseemos.

Figura 1.2: iBooks Author

1.3 Objetivos

El objetivo de este trabajo es desarrollar una herramienta que permita crear unos documentos profesionales de forma cómoda y además permitir al usuario subirlo a la plataforma que más le gusta.

La herramienta será multiplataforma para poder ser ejecutada en cualquier sistema operativo entre Windows, Mac y Linux. Podrá ser ejecutada por línea de comandos y también se desarrollará una aplicación de escritorio para que usuarios sin experiencia en la informática puedan aprovechar las mismas funcionalidades.

A la hora de crear el documento se podrá elegir el tipo de plantilla, que definirá la estructura y diseño del mismo. También se podrá elegir el lugar al que se desplegará la aplicación y además, especificar si el documento será privado o no, determinando el grupo de usuarios, que tendrán permiso de lectura en caso de pertenecer a la organización especificada.

1.4 Fases del desarrollo

Este trabajo está dividido en dos fases bien definidas:

- La primera fase del desarrollo consiste en la creación de la herramienta que tiene todas las funcionalidades necesarias para crear y desplegar el documento, además de especificar la autorización sobre el mismo. Esta herramienta se llama *Gitbook-setup*, para referirnos a ella a lo largo de esta memoria.

- Durante la segunda fase se desarrolla la aplicación de escritorio multiplataforma que aprovecha las funcionalidades de la herramienta gracias a la implementación de una API. Esta aplicación se llama *Modern-doc, para referirnos a ella a lo largo de esta memoria.*

Capítulo 2
Herramientas de desarrollo

Para el desarrollo de la herramienta se utilizará el lenguaje de programación *Javascript*, el cual es un lenguaje ligero e interpretado, orientado a objetos, más conocido como el lenguaje de script para páginas web. En este proyecto se usará este lenguaje en un entorno sin navegador, nodejs.

2.1 Node.js

NodeJS es el entorno sin navegador favorito de los programadores web. Permite crear un servidor de forma muy cómoda y sencilla orientado a la modularidad para que sea fácil de mantener y extender. Node se basa en módulos, que son, librerías de programación como las que se usan en Java, C++, etc.., lo cual permite una gran modularidad y comodidad a la hora de programar.

En caso de no tenerlo instalado, seguir las instrucciones en la página oficial de nodejs [1].

Para este proyecto hacen falta unas versiones mínimas para que funcione correctamente. En concreto la versión de node mínima es la 8.x.x y la de NPM (Node Package Manager) es 5.x.x.

Se necesitarán una gran cantidad de dependencias (que son paquetes node) para el desarrollo de *gitbook-setup*:

```
],
"dependencies": {
  "async": "^2.2.0",
  "bluebird": "^3.4.7",
  "cli-spinner": "^0.2.6",
  "contributor": "0.1.x",
  "ejs": "^2.5.6",
  "exec": "^0.2.1",
  "express": "^4.15.3",
  "fs-extra": "^2.0.0",
  "fs-promise": "^2.0.0",
  "gh-pages": "^0.12.0",
  "gitbook": "^3.2.2",
  "github": "^9.0.0",
  "gulp-gh-pages": "^0.5.4",
  "inquirer": "^3.0.1",
  "javascript-stringify": "^1.6.0",
  "jquery": "^3.2.1",
  "minimist": "^1.2.0",
  "mkdirp": "^0.5.1",
  "ncp": "^2.0.0",
  "node-github": "0.0.3",
  "octonode": "^0.7.9",
  "ora": "^1.3.0",
  "os": "^0.1.1",
  "passport": "^0.3.2",
  "path": "^0.12.7",
  "promise": "^7.1.1",
  "prompt": "^1.0.0",
  "scrape": "^0.2.3",
  "shell-task": "^1.0.0",
  "shelljs": "^0.7.7",
  "tacks": "^1.2.2"
},
```

Figura 2.1: Lista de dependencias del package.json

2.2 Electron

Electron [2] es un framework desarrollado en javascript que permite la creación de aplicaciones de escritorio multiplataforma empleando las tecnologías web desarrollado por GitHub, lo que garantiza revisiones constantes y es de código abierto.

Electron está basado en io.js y funciona bajo un subconjunto mínimo de librerías de Chromium, por eso da la impresión de abrir un navegador cuando empezamos a desarrollar una aplicación. Microsoft, Facebook, Slack y Docker utilizan esta plataforma.

Este framework se emplea en la segunda fase del desarrollo para crear la aplicación de escritorio que permitirá a los usuarios menos experimentados aprovechar todas las funcionalidades que ofrece nuestra herramienta *gitbook-setup*.

2.3 React.js

Para la interfaz de la aplicación se emplea la biblioteca de código abierto React.js que tiene el objetivo de desarrollar aplicaciones en una sola página.

React sólo maneja la interfaz de usuario en la aplicación; está construida únicamente para utilizar el patrón modelo-vista-controlador (MVC).

A continuación se puede ver un simple código en React.js que se encarga de mostrar una lista de la compra.

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}
```

Figura 2.2: Código ejemplo de React.js

Todo código html debe ser retornado por el método *render().* Se puede también construir distintas clases que hereden de Component. Muy útil si tenemos una página muy compleja y grande y queremos tenerlo todo modular.

Para más información visitar el tutorial oficial de React.js [9].

2.4 Gitbook

Gitbook [3] es una herramienta para crear documentos o documentación de proyectos o libros usando Markdown y Git/Github. Tiene la posibilidad de incluir plugins de todo tipo, desde ejercicios interactivos, hasta videos de Youtube.

Utilizando Markdown se pueden maquetar los documentos y ser exportados a distintos formatos como PDF, ebook o web (HTML, CSS y Javascript).

Gitbook está implementado empleando node.js, por lo tanto se puede instalar en nuestra máquina mediante el gestor de paquetes de node (NPM)

Capítulo 3
Implementación

En este capítulo se describen los aspectos más importantes de la implementación de nuestro proyecto. Tanto la parte de la herramienta como la parte de la aplicación de escritorio.

3.1 La herramienta gitbook-setup

La herramienta gitbook-setup permite la creación y despliegue de documentos, además de poder establecer una autorización sobre los mismos. A continuación se puede ver las principales partes de gitbook-setup que permiten al usuario aprovechar todas las funcionalidades.

A continuación se muestra la ayuda de gitbook-setup. Aunque la herramienta por línea de comandos require algo de conocimientos informáticos para su ejecución, se ha escrito una ayuda lo más clara posible para facilitar su uso:

```
[rudy@rudy bin]$ gitbook-setup help
Welcome to gitbook-setup. This program allow you to create book easily and also allows you to easily perform a deployment.
USAGE:
  $gitbook-setup [help] COMMAND

$gitbook-setup [help] create              -->   Create book. There are three options. [help] give you more detailed information about the command.
$gitbook-setup [help] install             -->   This command allows you to install all dependencies.
$gitbook-setup build                      -->   This command build your book. It work similarly as the command '$gitbook build'.
$gitbook-setup  help                      -->   Show general help.
$gitbook-setup  version | -v              -->   Show actual version of gitbook-setup.
```

Figura 3.1: Ayuda de Gitbook-setup

3.1.1 Creación de documentos

A la hora de crear un documento, hay que especificar previamente unos cuantos parámetros:

- Título *(obligatorio).*
- Tipo. *(por defecto book)*
- Autores. *(por defecto la variable de entorno $USER)*
- Lugares a donde se va a desplegar. *(por defecto ninguno)*
- Determinar si es privado o no. *(por defecto es público)*
- En caso de ser privado, indicar la organización de Github.
- Descripción. *(por defecto "No description about [Título]")*

Durante este proceso se comprueba si el tipo del documento y los lugares a los que se va a deplegar son válidos. En caso de no serlo, el documento no se crea.

Para crear el documento simplemente debemos ejecutar *gitbook-setup create* y nos pedirá que introduzcamos los parámetros nombrados anteriormente para la creación del mismo:

```
[rudy@rudy bin]$ gitbook-setup create
? Put the name of your book:  libro de rudolf
? What template do you want to use? Book
? Select where you would like to deploy your book: heroku
? Put some description of your book:  Descripción del libro
? Who will be the author\s (separated by commas)?:  rudy
? Will be private this document? (default: NO):  no
 heroku is correct module name
 Details about your book:
 {
        "name": "libro de rudolf",
        "template": "book",
        "deploys": [
                "heroku"
        ],
        "description": "Descripción del libro",
        "authors": [
                "rudy"
        ],
        "private": "no"
}
README.md file created.
 Now execute   $gitbook-setup install inside the libro de rudolf folder.
```

Figura 3.2: Creador de Gitbook-setup

El programa tiene tres posibilidades de crear documentos:

1. **Interactivamente**: como se ve en la figura 3.2 (justo encima). En esta forma el programa nos va preguntando secuencialmente por cada uno de los parámetros del documento y simplemente los vamos escribiendo. De forma similar se crea un proyecto node con el comando *npm init. Esta es la posibilidad más sencilla de crear documentos y recomendada a los usuarios menos experimentados con linux.*
2. **Por línea de comandos:** Esta forma es una de las más rápidas para la creación del documento. Simplemente se le pasan los argumentos al programa desde la línea de comandos y directamente te crea el documento. Si vemos la ayuda del creador con *gitbook-setup help creator,* se pueden ver distintos ejemplos.
3. **Por fichero de configuración:** Esta forma es también de las más rápidas y cómodas si queremos por ejemplo desde otro ordenador crear los ficheros de configuración para luego mandarlos a nuestro ordenador para que vayan creando los documentos. Es una manera muy cómoda una vez que ya tienes los ficheros de configuración listos.

3.1.1.2 Fichero de configuración

El fichero de configuración consiste en un objeto JSON en el cual se encuentran todos los parámetros especificados como pares de atributo-valor. El documento más simple tendría únicamente el atributo title puesto que es el obligatorio. Ya con ese fichero se podría crear un documento. Veamos un ejemplo de un fichero de título *“prueba” que es de tipo “api” y será desplegado en la plataforma* “heroku”:

Figura 3.3: Fichero de configuración de *gitbook-setup*

3.1.2 Instalación

Después de crear el documento sin errores, el usuario deberá instalar todas las dependencias y configurar el documento. Durante este proceso se realizan una serie de tareas:

1. Comprobar si es un proyecto gitbook-setup.
2. Crear un repositorio git.
3. Instalación de los módulos node de los que depende el documento.
4. Instalar los plugins de plantilla y despliegue del documento.
5. Rellenar gulpfile.js para automatización de tareas.

Una vez completada la instalación, el documento ya tiene el formato definido por el plugin de la plantilla y está listo

El gulpfile.js contiene las tareas que nos permitirán desplegar a las plataformas que se han seleccionado durante la creación. Puede que algunas de ellas requieran configuración previa como por ejemplo el establecimiento de un repositorio remoto.

Ahora podemos abrir con nuestro editor favorito los ficheros

que se encuentran en el directorio */docs* para empezar a escribir en nuestro documento.

Para instalar nuestro documento simplemente ejecutamos el comando *"gitbook-setup install"* y se encargará de instalar todas las dependencias y de configurar adecuadamente el proyecto para que funcione correctamente.

3.1.3 Despligue

Una vez instaladas todas las dependencias y tener configurado correctamente nuestro documento, podemos desplegar en internet a las plataformas que hemos elegido en su creación. Pero antes veremos cómo visualizar localmente el documento antes de publicarlo en la red.

Cada vez que queramos ver cómo nos está quedando el documento debemos ejecutar el comando ***gitbook serve*** desde la línea de comandos y se nos mostrará una url para previsualizar nuestro documento:

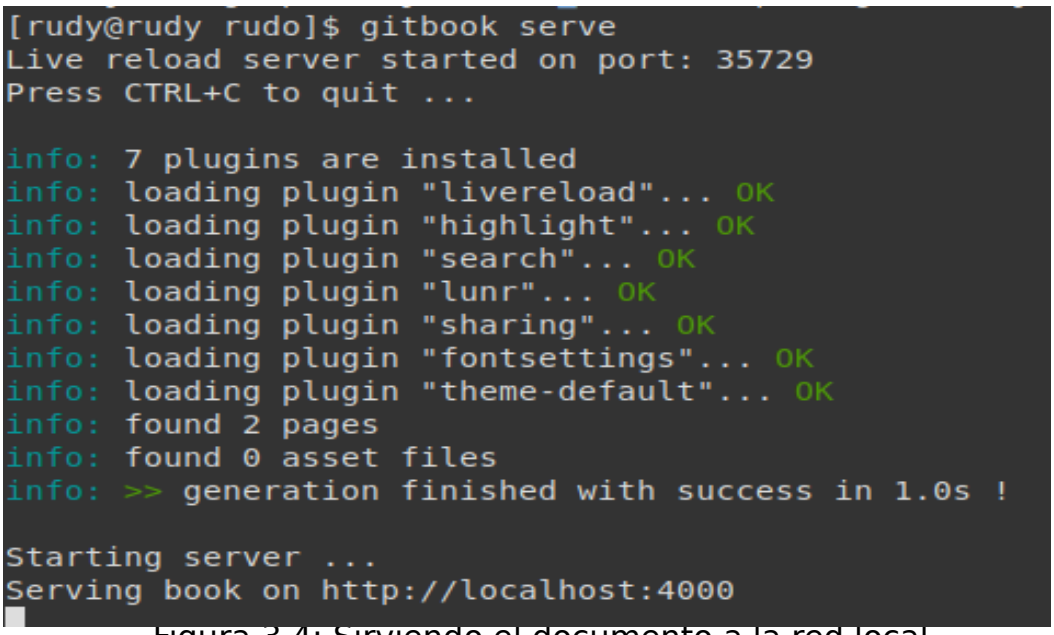

Figura 3.4: Sirviendo el documento a la red local

En el ejemplo de la imagen tendríamos que abrir el navegador poniendo en la url **http://localhost:4000.** Para quien no lo sepa, localhost quiere decir la propia máquina y el 4000 se refiere al puerto que está sirviendo el documento. Por lo tanto si visitamos esa dirección deberíamos ver algo así:

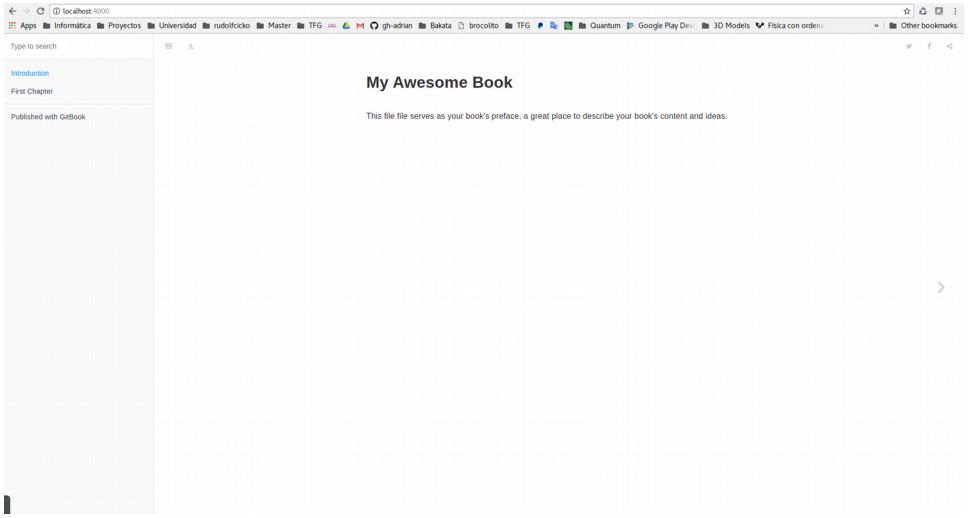

Figura 3.5: Visualización del documento en la red local.

Si nos gusta como ha quedado el libro debemos mirar las tareas que nos ofrece el fichero gulpfile.js por medio del comando ***gulp -T***. Este comando nos muestra una lista de las tareas que están disponibles y luego simplemente para ejecutar la tarea deseada.

Cada plugin de despliegue escribirá su tarea en el fichero gulpfile.js, por lo tanto cada tarea hace referencia a cada plataforma de despliegue. Asi que para desplegar a la plataforma que queramos, simplemente ejecutamos el comando ***gulp <nombre_tarea>***.

3.1.4 Los plugins de plantilla

A la hora de crear un documento, podemos elegir el tipo de documento que queramos. Podemos elegir si queremos un libro o una API, entre otras. Por ejemplo el tipo de documento API tendrá también incluídos varios plugins para reasaltar los códigos fuentes que pongamos en nuestro documento.

La plantilla es lo que define la estructura y diseño principal de nuestro documento. Para crear una plantilla propia podemos visitar el manual de Gitbook [5] para ello.

3.1.5 Los plugins de despliegue

Los plugins de despliegue son módulos node que poseen ficheros de configuración del documento para que esté listo para ser desplegado a la plataforma en cuestión.

Por ejemplo el plugin de despliegue de Heroku [4] se encarga de configurar el proyecto creando la aplicación en la plataforma Heroku, un servidor con node y en caso de ser un documento privado y configuración de la autenticación por medio de Oauth.

He definido una serie de reglas para que un plugin de despliegue pueda funcionar correctamente:

1. Todos los ficheros deben estar en una carpeta files.
2. El fichero *gulptask.js* contiene la tarea del plugin en cuestión y debe estar encapsulada [6] (Se puede ver el fichero gulptask.js del plugin de despliegue de Heroku en la figura 3.6). La tarea debería consistir en las instrucciones necesarias para desplegar el documento a la plataforma deseada.
3. Opcionalmente podemos tener un fichero *install.js* que posee el código que se desee para configurar todo el proyecto.

4. Todo el proyecto incluída la carpeta files debe ser un módulo npm con el nombre gitbook-setup-deploy-[nombre] donde [nombre] puede ir cualquier cosa que no exista. Para comprobar que no existe basta visitar el buscado de paquetes de node [7] buscando por "*gitbook-setup-deploy*-[nombre]" y comprobar que no existe.

```
(function () {
  var gulp = require('gulp');
  var exec = require('child_process').exec;
  var path = require('path');
  var configFile = require(path.join(process.cwd(), '.config.book.json'));;

  gulp.task('deploy-heroku', [], function() {
      exec('git add .', (err, out) => {
        if (err) console.log(err);
        console.log(out);
        exec('git commit -m \"Updating book for heroku at ' + new Date() + '\"', function (err, out) {
          console.log(out);
          exec('git push heroku master', function(err,out) {
            console.log(out);
            console.log("Now you can see your document at " + configFile['heroku_url']);
          })
        });
      });
  });
})(this);
```

Figura 3.6: gulptask.js del plugin de despliegue de Heroku.

Justo encima vemos el fichero gulptask.js del plugin de despliegue de Heroku. Como se puede ver, todo el código está encapsulado en una función. La gran ventaja de la encapsulación es que no pueden haber colisiones de variables (por tener el mismo nombre) puesto que cada tarea trabaja en su propio ámbito.

A continuación se puede ver la estructura de directorios del plugin de despliegue de Heroku para tener una idea más clara sobre los requisitos de un plugin de despliegue:

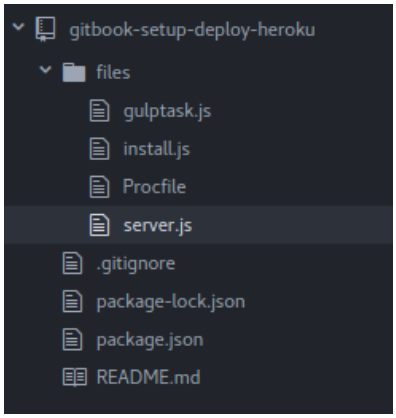

Figura 3.7: Estructura de directorios del plugin de Heroku.

En la carpeta *files* podemos meter todos los ficheros que deseemos y al instalar el proyecto gitbook-setup los tendremos en la raíz del documento para cualquier necesidad. Sobre todo hay que tener cuidado con los ficheros *gulptask.js* e *install.js puesto que* cada uno de ellos tienen un propósito determinado, el cual se ha descrito previamente.

En el caso del fichero *install.js* del plugin de Heroku [8] tiene unas serie de tareas bien determinadas para configurar el entorno para que esté listo para el despliegue:

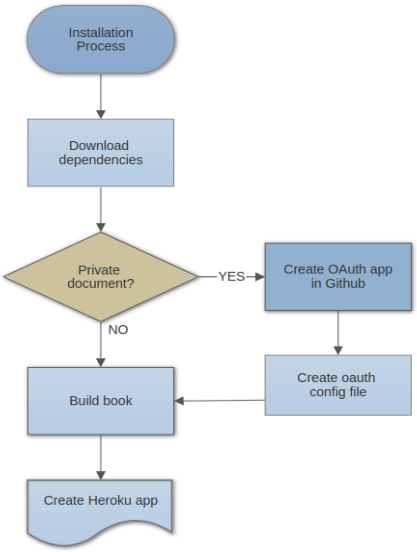

Figura 3.8: Diagrama de flujo de la instalación del plugin de Heroku.

3.1.6 La autorización

La herramienta nos permite también crear documentos privados, es decir que cualquier no podrá visualizar su contenido. Muy útil para por ejemplo, crear documentación de un determinado proyecto ya que los usuarios que tienen acceso a un documento privado son los miembros de una determinada organización en Github.

Para que un documento pueda ser privado y controlado por github, hay que crear una aplicación Oauth correspondiente. Para ello, a la hora de instalar el documento, en caso de ser privado, se pedirá al usuario introducir las credenciales de dicha aplicación. A continuación se ve la página de github que permite al usuario crear una aplicación Oauth:

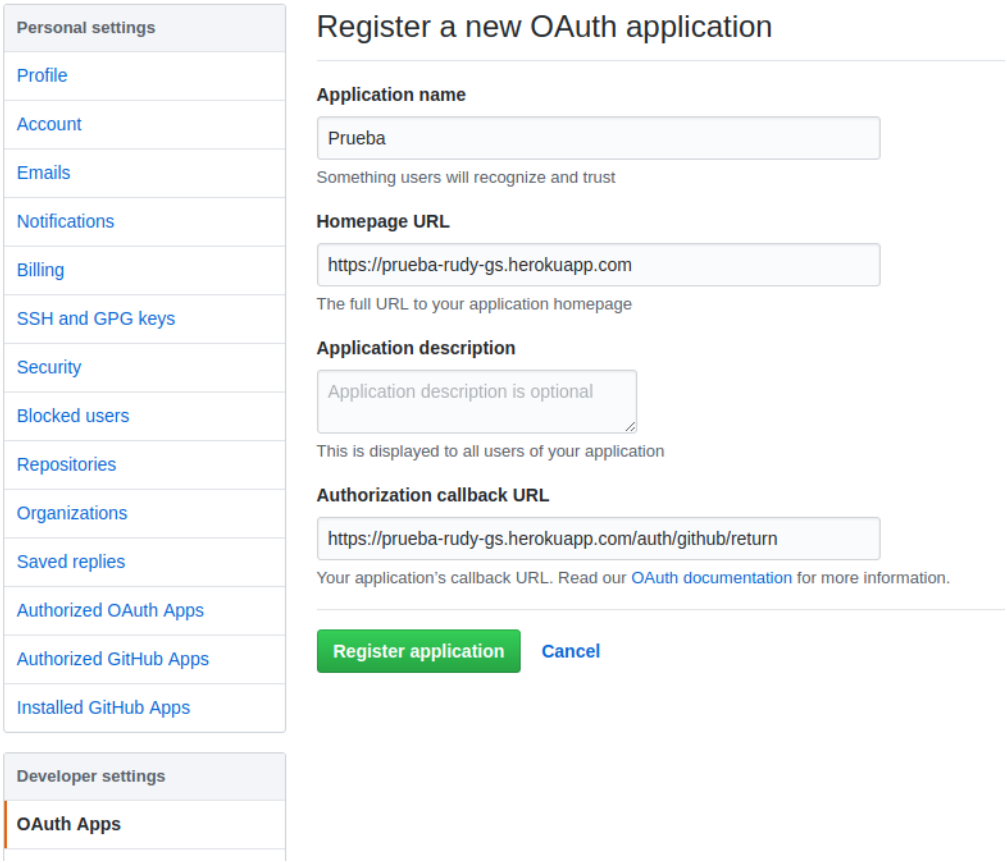

Figura 3.9: Página para crear una aplicación Oauth en Github

Debemos introducir un nombre para la aplicación, la url (en caso de tener el documento desplegado en varios servidores, deberemos crear una por cada uno de llos), una descripción opcional y la url de la callback. Esta url es llamada cuando la autenticación se ha realizado con éxito. Automáticamente el plugin de, por ejemplo Heroku, se encargará de mostrar el contenido del documento cuando se haga una petición a la callback url.

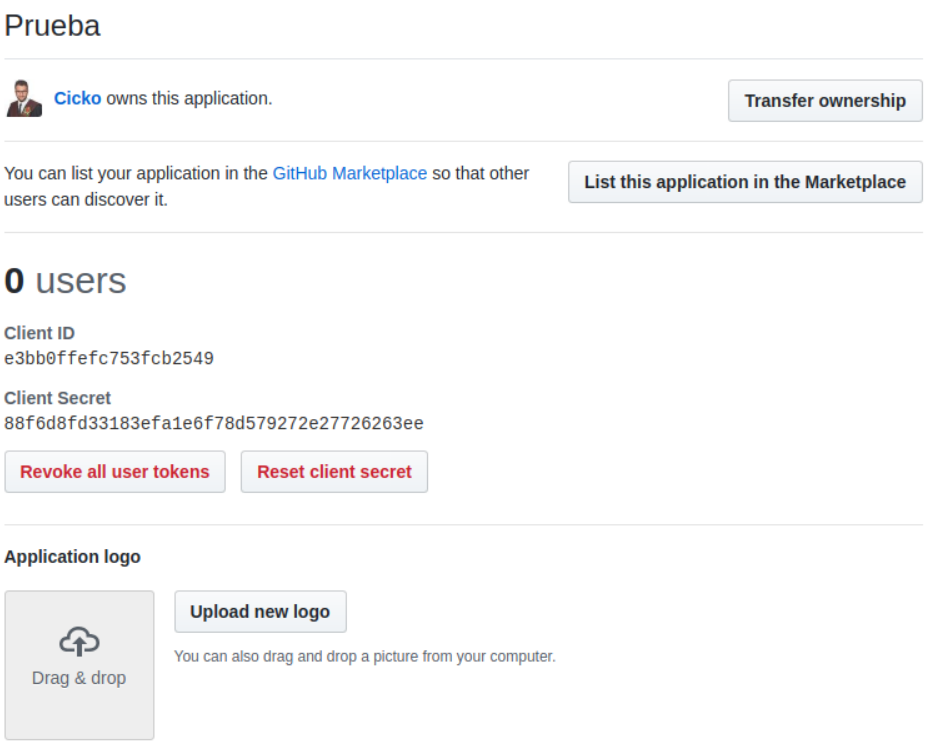

Figura 3.10: Página para crear una aplicación Oauth en Github

Una vez creada la aplicación podremos ver dos atributos muy importantes:

- Client ID: es un identificador público para aplicaciones. Debe ser único dentro del servidor, en este caso el servidor de Github.
- Client Secret: Este atributo es una clave secreta que sólo conoce la aplicación y el servidor de autorización. No debe conocerlo nadie más ya que se podría aprovechar para fines maliciosos.

3.1.7 La API

En esta sección comentaremos brevemente cómo se ha desarrollado la API de *gitbook-setup*. Gracias a la API nuestra herramienta puede ser utilizada desde cualquier otra aplicación de forma cómoda e intuitiva, posibilitando el uso de toda funcionalidad existente.

Para crear un documento usando la api simplemente llamaríamos al método create pasándole la configuración como parámetros y esperando a que nos retorne una callback con el posible error o mensaje:

```
var  gitbook_setup = require('gitbook-setup')

gitbook_setup.create({name: 'book', deploys: 'heroku'}, (err, msg) => {
  if (err) console.log(err);
  else console.log(msg);
});
```

Figura 3.11: Ejemplo de uso de la API de gitbook-setup

Del mismo modo se llaman a los demás métodos que ofrece la API. Para más detalles visitar la ayuda de la herramienta instalandola globalmente en nuestro sistema con *npm install -g gitbook-setup.* Luego simplemente ejecute *gitbook-setup -h* para ver la ayuda del programa.

3.2 La aplicación Modern Doc

Una vez descrita la herramienta desarrollada con node.js y entendido su funcionamiento, se puede usar la API en cualquier aplicación node.js deseada. Así ha surgido Modern Doc. Debido a que la herramienta en sí tiene una complejidad (aunque sea muy pequeña) de uso, los usuarios con ninguna experiencia en

informática no sabrán ni cómo abrir una línea de comandos. Por ello se ha desarrollado esta aplicación de escritorio multiplataforma para poder ser ejecutada fácilmente tanto en Windows, Linux y Mac.

3.2.1 La pantalla de creación.

A continuación se muestra la primera pantalla de la aplicación que se muestra al abrirla:

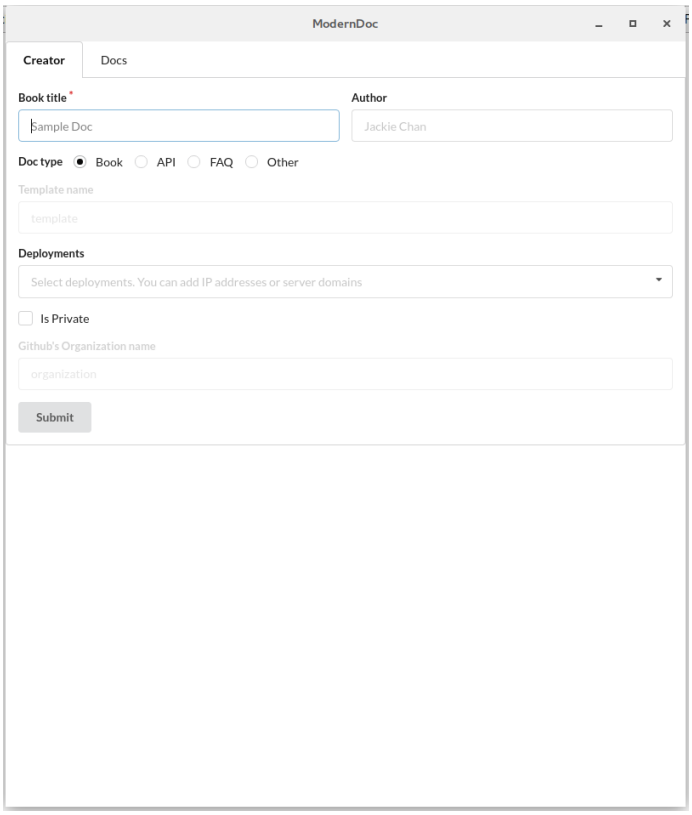

Figura 3.12: Pantalla principal de Modern Doc

Como se puede observar la aplicación tiene una interfaz simple y elegante. Ideal para que cualquier usuario no experimentado pueda usarla cómodamente.

3.2.2 Fichero *docs.json*

Una vez construido el documento con todos los campos necesarios, se creará, en caso de que sea el primer documento creado, un fichero dentro de una carpeta llamada *.modern-doc* en el *home* del usuario que contiene unas “copias de seguridad” de todos los documentos creados:

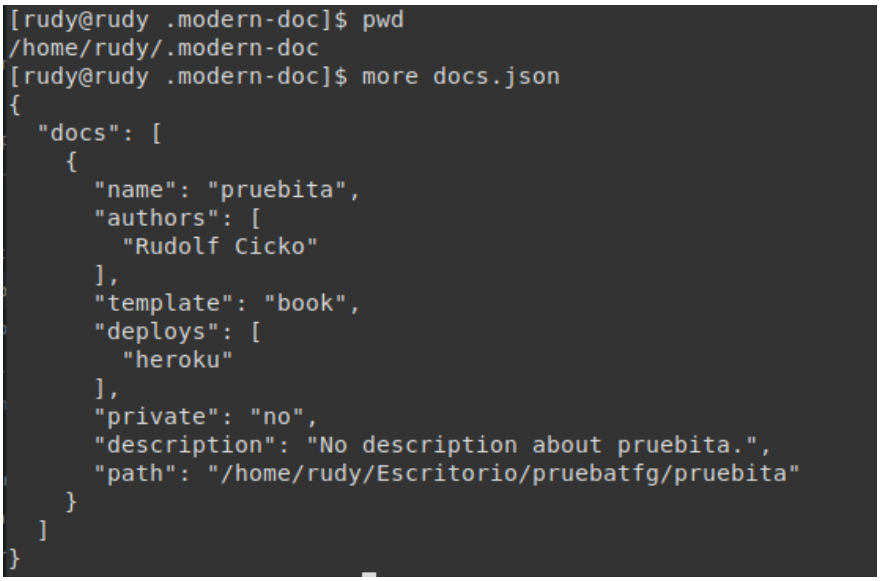

Figura 3.13: Fichero docs.json

Como se puede observar, después de crear el primer documento titulado *“pruebita”* se ha creado una carpeta en el *home* llamada *.modern-doc* que contiene un fichero JSON que contiene una lista de todos los documentos creados. La información que contiene este fichero es suficiente como para volver a recrear los documentos en caso de ser perdidos. También si se desea eliminar un documento debería hacerse a través de la aplicación puesto que si se elimina manualmente el fichero seguirá conteniendo dicho documento y provocará una inconsistencia.

3.2.3 *Pantalla Docs*

En la pestaña ‘*Docs*’ podemos ver una lista de todos los

documentos que ha creado el usuario en este ordenador. Como en el apartado anterior se ha visto un documento creado almacenado en el fichero *docs.json* ahora veremos cómo interpreta nuestra aplicación ese fichero para mostrar una lista de los mismos:

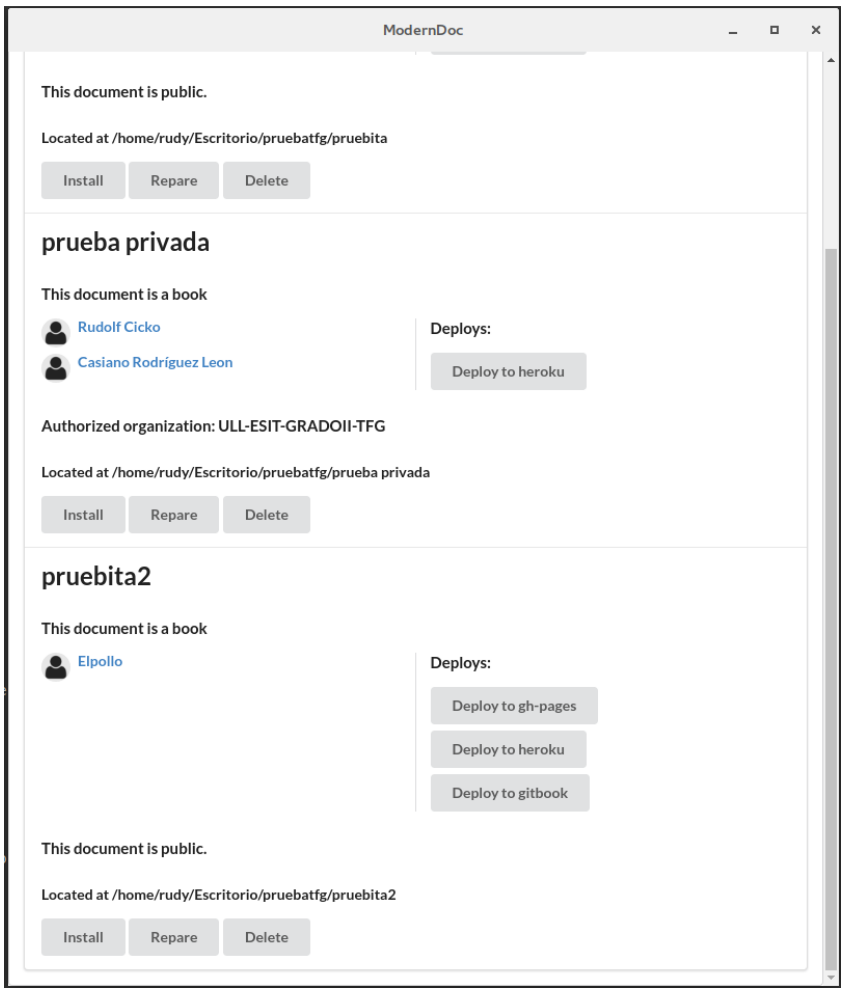

Figura 3.14: Pestaña ‘Docs’ de la aplicación

Como se puede observar, la pestaña ‘*docs*’ tiene una lista completa de todos los documentos que se han creado. Cada documento tiene distintos botones que permiten:

- Instalar: internamente hace exactamente lo mismo que el comando *gitboook-setup install* visto en el capítulo anterior.
- Reparar: en caso de que el documento esté corrupto, por ejemplo se ha borrado la carpeta pero la entrada sigue

existiendo en el fichero *docs.json*, se vuelve a crear la estructura del documento como si se volviera a crear.

- Borrar: si se desea borrar el documento este botón lo hace de la forma más correcta puesto que además de borrar la estructura de directorios del mismo también se encarga de borrar la entrada del fichero *docs.json*.

Por otro lado tenemos botones asociados a las distintas plataformas de despliegue para que de forma cómoda y rápida se pueda publicar.

También hay que tener en cuenta que los documentos pueden ser privados y de ser así, se indica la organización de Github que tiene permiso de lectura sobre el documento.

Para acabar de hablar sobre la aplicación, se puede destacar que prima su facilidad de uso y elegancia, permitiendo a una alta cantidad de usuarios puesto que se estima que un 2% de los usuarios en todo el mundo tienen el sistema operativo Linux instalado (asumiendo que éstos saben usar la terminal para ejecutar tareas). Por lo tanto la realización de esta aplicación incrementa teóricamente en un 98% la accesibilidad de la misma.

Capítulo 4
Problemas encontrados

En este capítulo se comentan los problemas más relevantes que se han encontrado durante el desarrollo de la herramienta y la aplicación de escritorio.

Al principio el mayor problema ha sido el asíncronismo de Javascript, el cual ha dificultado la realización del módulo puesto que se trabajan con múltiples ficheros (escribiendo y leyendo en ellos) y muchas veces un fichero depende de otro. Esto ha provocado buscar una solución para evitar una inconsistencia de los datos debido al dicho asincronismo.

Por suerte, Javascript es muy consciente de dicho comportamiento, por lo tanto ha creado las *Promesas [10].* Las promesas permiten coger bloques de código y encapsularlos en una promesa para tener un control sobre lo que ocurre después de dicho código, evitando un asincronismo. Veamos un pequeño ejemplo:

```
var fs = require('fs-extra')


fs.readFile('texto.txt', (err, out) => {
  if (err) return console.log(err)
  else {
    fs.writeFile('texto2.txt', out, err => {
      if (err) return console.log(err)
    })
  }
})

```

Figura 4.1: Ejemplo del hell de callbacks.

En el ejemplo de encima se puede ver un programa que lee del fichero *texto.txt* y escribe el contenido de éste en otro fichero *texto2.txt*. Como se puede ver, la lectura del código se vuelve un poco fea. Ahora que tenemos solo 2 callbacks anidados aún no es tan horrible pero imaginemos que tenemos 10 callbacks anidados. Eso hace que se vuelva el código difícil de mantener y de leer.

Ahora veamos el mismo programa pero que emplea promesas:

```
var fs = require('fs-extra')
var Promise = require('promise')

var readFile = Promise.denodeify(fs.readFile)
var writeFile = Promise.denodeify(fs.writeFile)


readFile('texto.txt', 'utf-8')
.then(out => writeFile('texto2.txt', out))
.catch(err => console.log(err));

```

Figura 4.2: Ejemplo de promesas

Capítulo 5
Conclusiones y líneas futuras

Este trabajo me ha permitido aprender mucho sobre las tecnologías web que se usan hoy en día en el mundo profesional. Además de aprender mucho he disfrutado mucho haciendo este trabajo puesto que constantemente tuvimos que luchar con las distintas dificultades que han ido apareciendo.

Como resultado final de este trabajo se ha desarrollado un módulo *nodejs* llamado *gitbook-setup* que permite la creación y despliegue de documentos. También se ha desarrollado plugins para despliegue a *Heroku* y a *Github Pages*. Además, el plugin de *Heroku* admite privatización del documento por medio de autenticación *Oauth*. Finalmente se ha desarrollado una aplicación de escritorio multiplataforma con una interfaz elegante y sencilla que permite al usuario menos experimentado aprovechar todas las funcionalidades que se ofrecen.

Este trabajo puede seguir creciendo mucho. La posibilidad de crear documentos que se visualizan por medio de las tecnologías web escribiendo su contenido en ficheros *Markdown [11]* podría incluso permitir la creación de una plantilla de gitbook que se encargue de crear páginas web empleando también Markdown. Lo cual proporcionaría una simplicidad y velocidad muy alta y efectiva a la hora de desarrollar páginas web. Gracias a que los documentos se basan en la herramienta *Gitbook,* se pueden añadir distintos plugins *[12]* para por ejemplo añadir videos, ejercicios interactivos, botones para compartir el contenido en redes sociales, etc..

Capítulo 6
Summary and Conclusions

This work has taught me a lot about the web technologies that are used today in the professional world. In addition to learning a lot I enjoyed doing this job very much, since we constantly had to struggle with the different difficulties that have been appearing.

As a final result of this work a node module called gitbook-setup has been developed that allows the creation and deployment of documents. Plugins have also been developed to deploy a Heroku and a Github Pages. In addition, the Heroku plugin supports document privatization through Oauth authentication. Finally, a multi-platform desktop application has been developed with an elegant and simple interface that allows the less experienced user to take advantage of all the functionalities that are offered.

Capítulo 7
Presupuesto

A continuación se describe el presupuesto en función de las tareas realizadas y el tiempo requerido.

7.1 Presupuesto

Tarea	Horas	Presupuesto
Investigación previa	10h	5€ / h
Desarrollo *gitbook-setup*	100h	10 € / h
Desarrollo de plugins	80h	10€ / h
Desarrollo *Modern-doc*	80h	10€ / h
Documentación	30h	7€ / h
TOTAL	**300h**	**2860€**

Tabla 7.1: Resumen del prespuesto

Referencias

[1] Descarga de nodejs

[2] Página oficial de Electron

[3] Página oficial de Gitbook

[4] Página oficial de Heroku

[5] Manual de Gitbook. Theming

[6] Encapsulación en Javascript

[7] Página oficial de los paquetes de Node

[8] Fichero install.js del plugin de despliegue de Heroku

[9] Tutorial de React.js

[10] Promesas

[11] Markdown

[12] Plugins de Gitbook