



UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

MÁSTER EN INGENIERÍA WEB

Proyecto Fin de Máster

Generación de catálogos en PDF para comercio electrónico

Autor

Jorge Cárdenas Caballero

Tutor

Eduardo García Pardo

Junio, 2017

Agradecimientos

Mis más sinceros agradecimientos a todos los que han hecho posible la realización de este proyecto, tanto profesores como compañeros. En especial a mi tutor Eduardo García Pardo y a mi pareja Claudia Lertora.

Este ha sido un gran año, espero que sigamos en contacto.

¡Gracias!



Resumen

En la actualidad el mundo del comercio electrónico está viviendo una edad de oro. Empresas como Amazon obtienen al año más de 100.000 millones en ventas *online* con alrededor de 300 millones de clientes en todo el mundo [22].

Uno de los problemas de las plataformas de comercio electrónico actuales radica en que un cliente necesita, en ocasiones, guardar información sobre un producto para su posterior consulta y, por lo general, la presentación que se obtiene al imprimir los productos mediante el navegador, por ejemplo en formato PDF, no es la más adecuada. En particular es habitual que el producto aparezca mezclado con elementos HTML de la página, que nada tienen que ver con el producto en sí. Un administrador de una plataforma de comercio electrónico, no suele contar con medios ni formación para poder modificar la presentación de sus productos a la hora de generar un documento, teniendo que recurrir a programadores especializados que realicen la maquetación de los mismos.

Otro problema relacionado con el anterior es que, pese a la gran inserción actual del comercio electrónico, muchos negocios *online* disponen también de tiendas físicas donde los clientes reclaman catálogos en papel. En este sentido, sería de utilidad poder generar un catálogo en PDF que incluya no solo un producto, sino el catálogo completo.

Como respuesta a este problema, se propone realizar una extensión para una plataforma de comercio electrónico conocida, con la que un administrador pueda crear plantillas de documentos PDF de forma rápida y sencilla, de modo que un usuario pueda guardar la información de los productos en PDF de acuerdo a las mismas.

En concreto, para la realización de este Trabajo Fin de Máster, se ha escogido la plataforma Drupal junto con su extensión de comercio electrónico Ubercart.

PALABRAS CLAVE

Comercio electrónico, PDF, Drupal, Catálogo, Producto



Abstract

Nowadays, the world of e-commerce is living its own golden age. Online companies are increasing their incomes every year. For instance, Amazon is reporting net incomes around 100.000 million dollar value in online sales, with 300 million customers around the globe.

One of the gaps of these online platforms is the inability to allow customers to save information of a product they are interested in, in order to subsequently consult it. Currently and generally speaking, when the details of a product are printed or saved in a document in PDF format, they are not usually rendered in the best way since, frequently, irrelevant HTML elements that have little to do with the product are displayed in between. This is normally due to the lack of easy ways for the administrator to modify how the appearance of the information will be stored, and thus, it results in a dependency in programmers in order to provide nice layouts.

As a side note, some online vendors are also able to reach out customers through physical stores, and a challenge they often face is the ability to generate an up-to-date PDF of their complete catalog in order to enhance and level up the shopping experience of all types of their customers.

As a way to overtake this drawback and enhance the experience of the users while shopping online, the objective of this project is to develop an independant module that can easily be plugged into an e-commerce platform and allow its administrators the possibility of designing templates for the rendering of their catalog. With this functionality, users may be able to easily save the details of a product and later be able to examine it. For the development of this project, Drupal platform and its extension for e-commerce Ubercart have been selected.

KEYWORDS

E-commerce, PDF, Drupal, Catalog, Product



Índice general

Agradecimientos	I
Resumen	III
Abstract	V
Lista de acrónimos	XIII
Objetivos	XV
Objetivo general	XV
Objetivos específicos	XV
1. Introducción	1
1.1. Motivación	1
1.2. Propuesta	3
1.2.1. Requisitos funcionales	3
1.2.2. Requisitos no funcionales	4
1.3. Tecnologías	5
1.3.1. Lenguajes	5
1.3.2. <i>Frameworks</i> y librerías	10
1.3.3. Sistemas	13
1.3.4. Herramientas de desarrollo	14
2. Metodologías de desarrollo	17
2.1. ¿Por qué usar metodologías?	17
2.2. Metodologías tradicionales vs ágiles	18
2.3. Metodología escogida	19
2.3.1. Scrum	20
2.3.2. Adaptación de Scrum a un Trabajo Fin de Máster (TFM)	22



3. Descripción informática	25
3.1. Aplicación de la metodología Scrum	25
3.1.1. Primera reunión 04/05/2017	25
3.1.2. Segunda reunión 25/05/2017	26
3.1.3. Tercera reunión 01/06/2017	27
3.1.4. Cuarta reunión 16/06/2017	27
3.1.5. Quinta reunión 23/06/2017	27
3.2. Desarrollo del proyecto	30
3.2.1. Primer <i>sprint</i>	30
3.2.2. Segundo <i>sprint</i>	37
3.2.3. Tercer <i>sprint</i>	43
3.2.4. Cuarto <i>sprint</i>	53
4. Conclusiones y trabajos futuros	61
4.1. Conclusiones	61
4.2. Trabajos futuros	62
Bibliografía	65
Anexo 1	67
4.3. Instalación manual de Drupal 7	67
4.4. Manual de instalación del módulo PDF-Warp	70

Índice de figuras

3.1. Arquitectura WAMP	31
3.2. Instalación de WAMP	31
3.3. Archivos de WAMP	32
3.4. Instalación de Drupal	34
3.5. Administración de Drupal	34
3.6. Formulario de añadir módulo a Drupal	35
3.7. Lista de módulos instalados en Drupal	36
3.8. Agregar nuevo tema a Drupal	36
3.9. Carpetas de módulo pdf_warp	38
3.10. Arquitectura de la aplicación	38
3.11. Exportación de vista a código PHP	40
3.12. Diseño de pantalla principal de administración	44
3.13. Diseño de creación de plantilla	45
3.14. Diseño de edición de plantilla	45
3.15. Página de administración principal	48
3.16. Página de creación de plantilla	49
3.17. Página de edición de plantilla	52
3.18. Diseño de edición de plantilla de catálogo	54
3.19. Diseño de edición de plantilla de catálogo	60
4.1. Web de descarga de Drupal	68
4.2. Directorio de la aplicación de Drupal	68
4.3. Creación de la base de datos	69
4.4. Wizard de Drupal	69
4.5. Lista de módulos y botón <i>Install new module</i>	71
4.6. Selección de modo de instalación de módulo	71
4.7. Botón <i>Save Configuration</i>	72



Índice de tablas

2.1. Comparativa procesos de desarrollo tradicionales vs ágiles	20
3.1. <i>Product Backlog</i> 04/05/2017	25
3.2. <i>Sprint</i> #1	26
3.3. <i>Product Backlog</i> 25/05/2017	26
3.4. <i>Sprint</i> #2	26
3.5. <i>Product Backlog</i> 01/06/2017	28
3.6. <i>Sprint</i> #3	29
3.7. <i>Sprint</i> #4	29



Lista de acrónimos

- **CSS:** *Cascading Style Sheets*
- **CMS:** *Content Management System*
- **DOM:** *Document Object Model*
- **FTP:** *File Transfer Protocol*
- **HTML:** *HyperText Markup Language*
- **HTTP:** *Hypertext Transfer Protocol*
- **JSON:** *Javascript Object Notation*
- **LAMP:** Linux, Apache, MySQL, PHP
- **SPA:** *Single-Page Applications*
- **SSH:** *Secure SHell*
- **TFM:** Trabajo Fin de Máster
- **W3C:** *World Wide Web Consortium*
- **WAMP:** Windows, Apache, MySQL, PHP



Objetivos

A continuación se presentan tanto el objetivo general como los objetivos específicos de este TFM.

Objetivo general

El objetivo general del proyecto consiste en el diseño e implementación de una solución que permita la generación de documentos PDF a partir de la información obtenida de los productos de una plataforma de comercio electrónico. Un administrador de un portal de comercio electrónico debe ser capaz de gestionar la disposición de los elementos que componen un producto (fotografía, descripción, título, precio, referencia, etc.) del PDF a generar de una forma ágil y sencilla. Todo este proceso será transparente para un usuario final, que únicamente con presionar un botón obtendrá la información de los productos en un documento PDF.

Objetivos específicos

Para alcanzar el objetivo general del proyecto es necesario cubrir los siguientes objetivos específicos:

- Elegir una plataforma base entre LAMP y WAMP para el desarrollo del TFM
- Instalar y aprender a utilizar el *Content Management System* (CMS) Drupal cogiendo soltura con sus elementos: vistas, tipos de contenido, módulos, etc.
- Coger soltura con el lenguaje de programación PHP.
- Utilizar Ubercart como extensión de comercio electrónico para la realización del TFM.
- Permitir que el administrador de la plataforma de comercio electrónico pueda gestionar distintas plantillas PDF.
- Permitir la edición ágil del contenido de una plantilla PDF en el lado del cliente.



- Elegir una tecnología para el lado del cliente que permita crear un interfaz de usuario ágil.
- Permitir que un usuario anónimo genere documentos PDF a partir de productos y catálogos de productos.
- Crear una solución exportable a otras instalaciones de Drupal mediante la creación de un módulo

Capítulo 1

Introducción

En este documento se recoge la memoria del TFM titulado “Generación de catálogos en PDF para comercio electrónico”. En este capítulo se recoge la motivación del proyecto, la propuesta, los requisitos y las tecnologías utilizadas en el desarrollo.

1.1. Motivación

El comercio electrónico tal y como lo conocemos hoy, consiste en la compra y venta de productos o servicios a través de medios electrónicos, como por ejemplo Internet. Con la llegada en masa de la red de redes a los hogares del mundo, se creó un nicho de mercado que en ese entonces estaba desaprovechado, como es el negocio de la venta *online*. Aunque anteriormente ya existían otros sistemas de comercio electrónico, basados en televisión, teléfono o incluso a través de Internet, no fue hasta la llegada de Amazon¹ o eBay² cuando el comercio *online* empezó a masificarse. A partir de este punto clave y, viendo el éxito que supone para un comercio satisfacer la demanda creciente de clientes de comercio electrónico, surgieron cada vez más plataformas que ofertaban productos y servicios por Internet, como Netflix³ (en sus inicios un servicio de alquiler de DVD con un sistema de reparto a domicilio), Alibaba.com⁴ (parecido a Amazon pero para el mercado chino) y Zappos⁵ (plataforma de venta de zapatos) entre otros. También surgieron alrededor del comercio electrónico, otros servicios útiles para usuarios y vendedores, como Paypal⁶ (plataforma de pago *online*) o AdWords⁷ de Google, para publicitar los productos y servicios a través de Internet, de manera dirigida al usuario con interés.

¹<https://www.amazon.es/>

²<http://www.ebay.com/>

³<https://www.netflix.com/es/>

⁴<https://www.alibaba.com/>

⁵<http://www.zappos.com/>

⁶<https://www.paypal.com/es/home>

⁷<https://adwords.google.com/home/>



Al principio sólo las grandes empresas podían permitirse aumentar su visibilidad *online* mediante el uso de caras soluciones personalizadas, sin embargo, poco a poco, fueron surgiendo plataformas mucho más asequibles para el pequeño comercio como pueden ser osCommerce⁸, Magento⁹, o PrestaShop¹⁰, o Shopify¹¹, además de soluciones gratuitas basadas en *frameworks* de *software* genéricos como Drupal, Joomla o Wordpress que poseen extensiones para comercio electrónico.

A partir de este punto, es raro ver una tienda que no tenga opción de venta *online*, actualmente la única diferencia es el tiempo que tarda un cliente en recibir el producto, siendo las grandes aquellas que se pueden permitir envíos en un único día.

Prácticamente la totalidad de las tiendas *online* se basan en los mismos principios: se presenta un catálogo de productos con imagen, descripción y precio y el usuario puede elegir tranquilamente y desde su hogar los elementos que va a adquirir. Actualmente, los sistemas de ranking y los comentarios de otros clientes pueden ayudar al usuario final a elegir un producto, siendo los de más valoración los que más posiblemente satisfarán sus necesidades.

Relacionado con el contexto anteriormente descrito, surge la necesidad que da lugar a este TFM. En ocasiones un usuario necesita guardar la información de un producto para su consulta posterior, o un comercial necesita ofrecer a un posible comprador un catálogo personalizado de productos. En estos casos existe una limitación muy grande a la hora de generar el documento con un formato atractivo a partir de la propia tienda online y, por lo general, los resultados suelen ser muy pobres y no muestran la información más relevante como al vendedor le gustaría verla, por ejemplo, el precio más grande, la foto en el centro, etc. Además, es común encontrarse que, al imprimir una página de un sitio web en formato PDF, se obtenga en el documento numerosos elementos HTML que nada tienen que ver con el producto deseado (cabeceras, menús, pies de página, etc.). Todos estos retoques se tienen que pensar previamente cuando se planifica el sistema de venta *online* y, por lo general, tienen que ser realizados por un experto en la materia que no todos los comercios poseen.

De manera resumida, la motivación principal de este TFM radica en la carencia de soluciones disponibles a la hora de elegir la disposición de elementos en un documento

⁸<https://www.oscommerce.com/>

⁹<https://magento.com/>

¹⁰<https://www.prestashop.com/es>

¹¹<https://www.shopify.es/>

PDF, orientado en este caso al mundo del comercio electrónico.

Existen muchas soluciones de generación de PDF, tanto a nivel de *back-end* como de *front-end*, con un alto nivel de personalización, pero son librerías que requieren conocimientos avanzados de programación y administración de sistemas para ser usadas, y estos conocimientos no suelen estar al alcance del propietario de una empresa que solamente quiere ofrecer sus productos en Internet.

También existen soluciones estándar, Chrome, por ejemplo, permite convertir una página web a PDF, pero está limitado por la calidad del trabajo que el maquetador web haya realizado con el archivo `print.css`, en el que se guardan las reglas de estilo a la hora de imprimir las diferentes páginas de la web. De nuevo, la modificación de este archivo requiere conocimientos avanzados de maquetación web y no suele ser fácil conseguir que una página se imprima como de la forma deseada, además de que habría que ocultar elementos sobrantes, como publicidad y menús.

1.2. Propuesta

A partir de lo elaborado en la introducción, motivación y objetivos, se propone por tanto, la realización de un módulo para una de las plataformas libres de comercio electrónico más utilizadas, como es Ubercart para Drupal, que permita a un administrador la gestión de plantillas para generación de PDF de productos del catálogo que el comercio tenga a la venta.

1.2.1. Requisitos funcionales

De manera resumida, el módulo deberá implementar la siguiente funcionalidad:

- Botón de generación de PDF: un usuario debe ser capaz de generar un documento PDF de un producto con tan sólo pulsar un botón.
- Administración de plantillas: un administrador puede acceder a una sección de administración de plantillas, protegida por privilegios especiales.
- Creación de plantillas: un administrador puede crear plantillas nuevas proporcionando un nombre de plantilla y un tipo.
- Borrado de plantillas: un administrador puede borrar plantillas antiguas para no sobrecargar la base de datos.



- Activado de plantillas: un administrador puede activar una plantilla para ser usada en la generación de PDF.
- Editado de plantillas: un administrador puede editar las plantillas entrando en una pantalla de edición en la que se muestran diferentes opciones.
- Mover elementos de una plantilla: un administrador en modo de edición de plantilla puede mover los elementos y posicionarlos dónde desee.
- Cambiar tamaño de elementos: un administrador en modo de edición de plantilla puede cambiar el tamaño de los elementos de la misma.
- Cambiar tamaño de fuente: un administrador en modo de edición de plantilla puede cambiar el tamaño de fuente de los elementos de la misma.
- Botón de guardar cambios de plantilla: un administrador en modo de edición de plantilla puede pulsar un botón para guardar los cambios de la misma.

1.2.2. Requisitos no funcionales

El módulo deberá cumplir los siguientes requisitos no funcionales

- Uso de tecnologías de libre distribución: las tecnologías usadas para el desarrollo del módulo deben ser de libre distribución.
- Módulo exportable: el módulo desarrollado debe poderse instalar en otras instalaciones de Drupal que cumplan los requisitos y dependencias.
- Stack tecnológico: uso de WAMP o LAMP.
- Uso de Drupal: Drupal tiene a su disposición varios módulos que expanden su funcionalidad para satisfacer las necesidades de los propietarios de tiendas on-line.
- Uso de Ubercart: dentro de las opciones de comercio electrónico disponibles para Drupal, debe usarse la extensión de Ubercart.
- JSON para comunicaciones entre *back-end* y *front-end*.
- Uso de Angularjs para el *front-end*.

Por lo tanto para facilitar la labor de propietarios y administradores en la presentación de sus productos a la hora de generar un documento PDF, se propone la creación de una aplicación que brinde la funcionalidad deseada para una plataforma de comercio electrónico.

1.3. Tecnologías

En este capítulo se enumerarán las tecnologías que se han utilizado para la realización de éste proyecto.

1.3.1. Lenguajes

Los lenguajes de programación usados para el desarrollo del proyecto aparecen explicados a continuación.

HTML

El lenguaje *Hypertext Markup Language* (HTML) sirve principalmente para la publicación y estructuración de documentos y especificación de hipervínculos. Junto a *Cascading Style Sheets* (CSS) y Javascript, forman la base de la programación web actual. Es un lenguaje basado en etiquetas y todo el contenido de un documento está almacenado entre una etiqueta de apertura y otra de cierre. La primera descripción pública de HTML fue un documento llamado “HTML Tags” publicado por Tim Berners-Lee a finales de 1991 y describe los primeros 18 elementos incluidos en el lenguaje [3]. Durante todo este tiempo se han ido publicando nuevas versiones de HTML en las que se añadían nuevas etiquetas y se eliminaban otras obsoletas. La versión actual es HTML5 que añade múltiples etiquetas avanzadas para controles multimedia, validaciones de formularios y maquetación de páginas web optimizadas para buscadores.

Para programar en HTML basta con usar un editor de texto cualquiera y un navegador web para ver la página web interpretada desde el código escrito. La extensión de un documento HTML es “.html” o “.htm”.

Ejemplo:

```
<!DOCTYPE HTML>

<!-- Muestra una página web con el título "Ejemplo"
y el texto "!Hola Mundo!" -->
<html>
  <head><title>Ejemplo</title></head>
  <body>
    <p>¡Hola, Mundo!</p>
  </body>
</html>
```



CSS

CSS es el acrónimo de *Cascade Style Sheets* el cuál es un lenguaje de estilo usado para describir la presentación de un lenguaje basado en etiquetas o marcas como HTML. Su diseño está enfocado principalmente a separar la presentación del contenido del documento, incluyendo aspectos como la estructura, colores y fuentes. Las especificaciones CSS son mantenidas por el *World Wide Web Consortium* (W3C). La primera propuesta de CSS la realizó Håkon Wium Lie en 1994 [12] y a pesar de que en el momento habían más propuestas de lenguajes de estilo, CSS se impuso y acabó siendo lanzado en 1996 por el W3C [4]. El lenguaje tuvo unos inicios complicados ya que los navegadores inicialmente ofrecían soporte muy limitado para CSS. No fue sino hasta la versión 5 de Internet Explorer que se dio soporte a la totalidad del lenguaje [2], y aun así muchos otros navegadores seguían ofreciendo compatibilidades limitadas. Actualmente la especificación CSS utilizada es la 3, aunque a partir de esta versión, el lenguaje se partió en módulos y se puede hacer uso de algunos de la especificación 4.

Al igual que con HTML, para programar en CSS basta con un programa editor de texto y un navegador web. Es un lenguaje ligado a HTML y puede aparecer o bien incrustado en el mismo documento “.html” o bien en su propio archivo “.css” que luego es importado al documento web.

Ejemplo:

```
body {  
    font-size: 12px;  
    //Cambio de tamaño de fuente a 12 en la etiqueta <body>  
}  
div {  
    color: #fff;  
    // Cambio de color de fuente en la etiqueta <div>  
}
```

Javascript

Javascript es un lenguaje de programación de alto nivel creado originalmente en el año 1995 para ser un lenguaje de *scripting* complementario a los navegadores web. Inicialmente se le dio el nombre de Mocha en su fase de desarrollo, aunque luego su nombre cambió a LiveScript para más tarde ser reemplazado por Javascript en su lanzamiento junto a Netscape Navigator 2.0. Se cree que se le dio este nombre para aprovechar la po-

pularidad de Java. Un año después fue adoptado por Microsoft bajo el nombre de JScript para evitar problemas legales de nomenclatura y más tarde fue estandarizado por ECMA Internacional lanzando la especificación de lenguaje ECMAScript [15, 1]. Las especificaciones de ECMAScript se han ido adaptando a los tiempos modernos, añadiendo nuevas funcionalidades y corrigiendo otras. Actualmente se trabaja mucho con la especificación ECMAScript 5, aunque la especificación 6 está ganando popularidad desde su salida en el año 2015 y sobre todo por la adopción de la misma por los desarrolladores de los *frameworks* y librerías más populares como Angularjs, Reactjs y Babel.

Javascript es un lenguaje dinámico, sin tipado e interpretado en tiempo de ejecución. Es un lenguaje con múltiples paradigmas: funcional, imperativo e incluso en las últimas especificaciones, integra orientación a objetos.

Para programar en Javascript, basta con usar un editor de texto y un navegador web, aunque para aprovechar todas sus funcionalidades, se requiere de un servidor web configurado adecuadamente, ya que, por ejemplo, el acceso a archivos de la máquina podría producir problemas. Es posible incrustar *scripts* de Javascript en un documento HTML o bien desarrollar en un documento aparte e incrustarlo posteriormente. La extensión de Javascript es “.js”.

Ejemplo:

```
// Muestra en la consola de desarrollo del navegador web el
// mensaje "¡Hola Mundo!"
function hello() {
    console.log("¡Hola Mundo!");
}
```

PHP

PHP es un lenguaje de scripting para *back-end* concebido principalmente para diseño web, pero que es utilizado también como lenguaje de propósito general. Fue creado en el año 1995 por Rasmus Lerdorf quien escribió unas aplicaciones de consola en C para mantener su página web personal [11]. Posteriormente las amplió para funcionar con WebForms y comunicarlas con bases de datos y lo llamó “*Personal Home Page/Forms Interpreter*” o PHP/FI. Inicialmente no fue concebido para ser un lenguaje de programación, pero Lerdorf fue paulatinamente añadiendo funcionalidades hasta que se formó un equipo de desarrollo, lanzando su primera versión oficial en 1997 [10].



Durante su tiempo de vida, el lenguaje ha recibido numerosas actualizaciones. Actualmente se encuentra en la séptima iteración de PHP, con la salida de la versión 7.2 programada para noviembre de 2017 [16].

Es un lenguaje orientado objetos pero que nos permite utilizar varios paradigmas de programación como el estructurado, imperativo, funcional y reflexivo. Como suele ocurrir con los lenguajes de *scripting*, PHP es un lenguaje interpretado, es decir, no requiere de compilación previa para su ejecución. Además es de tipado dinámico, por lo que las variables no están asociadas a un tipo de datos en el momento de su declaración, sino que se les asigna un tipo automáticamente en el momento de su uso.

Para empezar a programar en PHP se requiere un servidor web capacitado para su ejecución y además los paquetes de comandos de consola requeridos para el inicio de programas escritos en este lenguaje. Un servidor muy utilizado es Apache. Escribir programas en PHP es sencillo y la forma más rápida consiste en incrustar pequeños *scripts* en los documentos HTML de nuestra web que serán interpretados por el servidor y completamente invisibles para un usuario final. Los ficheros de lenguaje PHP tienen la extensión “.php”.

Ejemplo:

```
<!DOCTYPE HTML>
<!-- Crea una página con título Ejemplo -->
<html>
  <head>
    <title>Ejemplo</title>
  </head>
  <body>
    <?php
      // Muestra "!Hola, Mundo!" por la salida estándar
      echo "!Hola, Mundo!";
    ?>
  </body>
</html>
```

JSON

Javascript Object Notation (JSON) se trata de un estándar abierto de formato de archivo para el envío de información a través de pares clave-valor. Es muy utilizado hoy en

día para las comunicaciones entre cliente y servidor. La primera especificación de JSON vino de la mano de Douglas Crockford quien lo creó originariamente para ser un subconjunto de Javascript [7]. De hecho es actualmente muy utilizado junto a este lenguaje, sin embargo, JSON es independiente del lenguaje con el que sea usado. En el año 2013 se estableció un estándar por ECMA Internacional.

Los tipos de datos que pueden ser usados en un documento JSON son: Boolean, String, Number, Array, Object y null.

Ejemplo:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York"
  },
  "phoneNumbes": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "spouse": null
}
```

Este código representa un objeto con los datos personales de un individuo.

Nombre, apellido, edad, dirección compuesta por calle y ciudad, una lista de números de teléfono, en la que se incluye el número de su casa y de la oficina y si tiene cónyuge.



1.3.2. *Frameworks* y librerías

En esta sección se hace una recopilación de los *frameworks* y librerías de terceros usados durante el desarrollo del proyecto y sus características.

Drupal

Drupal es un sistema de gestión de contenidos (o CMS según sus siglas en inglés derivadas de *Content Management System*) de código abierto que es utilizado por un 2,3 % de sitios web del mundo [23]. Es un sistema libre, modular y multipropósito que permite publicar artículos, imágenes, archivos y que también ofrece la posibilidad de otros servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Es un sistema de carga dinámico, es decir, todos los recursos se almacenan en una base de datos y se cargan en tiempo real a la hora de generar una página. Este CMS está escrito utilizando el lenguaje PHP valiéndose de las librerías de Symfony, por lo que es un sistema multiplataforma soportado tanto por Windows como por sistemas Unix.

Originalmente fue escrito por Dries Buytaert como un foro de opinión. En el año 2001 se convirtió en un proyecto de código abierto y ganó popularidad en 2004 al utilizarse para crear el sitio *DeanSpace* para el candidato a las primarias por el Partido Demócrata de Estados Unidos Howard Dean [13].

Actualmente Drupal se encuentra en la versión 8, aunque aún brinda soporte a la versión 7 que cuenta aún con muchos adeptos. La mayoría de los módulos y temas de Drupal desarrollados por sus usuarios son compatibles con la versión 7.

Algunos ejemplos de sitios desarrollados con Drupal son las webs de Ubuntu, Intel o Economist, entre otros.

Angularjs

Angularjs es un *framework* para aplicaciones web de código abierto basado en Javascript especializado en *Single-Page Applications (SPA)*. Actualmente es mantenido por Google y una comunidad formada por desarrolladores individuales y empresas. Fue inicialmente programado en el año 2009 por Miko Hevery como un *software* base para una aplicación *online* de almacenamiento de datos en JSON [6], aunque debido al pobre acogimiento que tuvo la idea, el producto se desechó y se lanzó Angular como una librería de código abierto. Inicialmente todo el contenido de Angularjs iba en un único paquete pero

en las versiones más recientes, se ha modularizado para que cada desarrollador tenga una instalación personalizada de acuerdo a sus necesidades.

Actualmente se dispone de dos vertientes de Angularjs, la versión 1.x y la versión 2.x. Aunque compartan el nombre, la Angularjs 2.x no se trata de la siguiente iteración del *framework*, si no que se trata de un remodelado completo incompatible con la versión anterior, por lo que se tratan como si fueran dos librerías diferentes y ambas continúan recibiendo soporte.

El *framework* se basa en el modelo vista presentador con lo que proporciona ficheros de vista (HTML), un modelo de datos y un presentador (controlador). Cada vez que Angularjs detecta un cambio de estado, ejecuta un proceso que chequea a cuantos ámbitos (*scopes*) puede afectar dicho cambio de estado y busca todas las variables que hayan podido sufrir cambios para actualizar sus vistas. Este proceso de chequeo se llama ciclo de digest y constituye el núcleo del funcionamiento de Angularjs. También al cambiar un valor en una vista se actualiza la variable asociada en el controlador (*two-way data binding*). Angular está compuesto por módulos y directivas, y los desarrolladores pueden crear sus propios elementos y compartirlos con otros a través de repositorios públicos.

En este proyecto se usa la versión 1.6 del *framework*. Algunos sitios desarrollados con Angular son Wolfram Alpha, ABC News o NBC

JQuery

Es una librería de Javascript diseñada para simplificar la programación en el lado del cliente. Es de código abierto y es la librería más usada de Javascript [24]. Fue lanzado en el año 2006 por John Resig quien se inspiró en las librerías de cssQuery [18].

Principalmente contiene funciones para manipular el *Document Object Model* (DOM) que es un conjunto estándar de objetos para representar documentos HTML, y una interfaz estándar para acceder a ellos y manipularlos. También permite realizar animaciones, gestionar eventos y desarrollar aplicaciones basadas en Ajax. Sus principales características son las siguientes:

- Selección de elementos DOM
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath



- Eventos
- Manipulación de la hoja de estilos CSS
- Efectos y animaciones
- AJAX
- Soporte de extensiones

Ejemplo:

```
// Pinta por la consola del navegador la frase "¡Hola Mundo!"  
// al terminar de cargar la página  
$(document).ready(function() {  
    console.log('¡Hola Mundo!');  
});
```

JsPDF

JsPDF es una librería Javascript para la generación de PDF en el lado del cliente. Cuenta con un potente motor para dibujar elementos geométricos, incrustar imágenes y textos variando el tipo de letra y el tamaño. Su primera versión fue lanzada en el año 2012 y desde entonces sus creadores (Parallax) han ido incorporando funcionalidades y librerías para aumentar sus capacidades.

Dragabilly

Se trata de una librería creada para JQuery que permite utilizar las características de *drag and drop* en una página web. La ventaja que ofrece es que permite mover un elemento web a cualquier punto de un contenedor. Cuenta con un *wrapper* para Angularjs que permite añadirlo como una directiva para ser usado directamente sin necesidad de JQuery. En el proyecto es usado para mover los elementos de una plantilla PDF.

Html2Canvas

Html2Canvas es una librería Javascript para la conversión de código HTML al objeto Canvas de HTML5. Este objeto permite renderizar la página web con todos sus estilos y guardarla como si fuera una imagen. Al ser compatible con JsPDF permite la generación de un documento PDF en modo imagen tal y como se muestra en la web y sin los problemas derivados de los estilos css de impresión.

1.3.3. Sistemas

En esta sección se enumeran los sistemas principales del *back-end* sobre los que se monta la aplicación.

Apache HTTP Server

Llamado coloquialmente Apache, es un servidor web multiplataforma de código abierto. La Apache Software Foundation¹² es la encargada actualmente de dar soporte y programación del producto. Su desarrollo comenzó en el año 1995 y jugó un papel importante en el crecimiento de la World Wide Web sustituyendo rápidamente a NCSA HTTPd como servidor *Hypertext Transfer Protocol* (HTTP) dominante en el mercado. Actualmente es el servidor más utilizado del mundo siendo utilizado por un 46 % de los sitios web en activo [14].

Entre sus principales características se encuentra el soporte para lenguajes de programación como Perl, Python, Tcl y PHP, módulos de autenticación (mod_access, mod_auth, mod_digest y mod_auth_digest), soporte para SSL y TLS, hosting virtual, y autenticación por contraseña o firma digital entre otros.

MySQL

MySQL es un sistema gestor de base de datos relacionales de código abierto creado por la compañía sueca MYSQL AB en el año 1995 [17]. Durante este tiempo MySQL fue ganando popularidad y en el año 2008 fue comprado por Sun Microsystems para posteriormente pasar a manos de Oracle, actuales propietarios. Antes de la adquisición por parte de Oracle, uno de los creadores y fundadores de MYSQL AB, Monty Widenius, realizó un fork de la versión 5.5 de MySQL para crear MariaDB y mantener la esencia del producto [25].

Actualmente MySQL se encuentra en la versión 5.7 y se ofrece tanto la versión MySQL Community Server como Enterprise Server, siendo ésta última una licencia de pago. Aunque MySQL server ha evolucionado para poder albergar y tratar volúmenes altos de datos, se sigue utilizando principalmente para aplicaciones de pequeño a mediano tamaño.

Sus principales características son:

- Sistema de base de datos relacional

¹²<https://www.apache.org/>



- Arquitectura cliente-servidor
- Compatible con el estándar de SQL
- Sentencias SELECT anidadas
- Vistas
- Procedimientos almacenados y triggers
- Transacciones

1.3.4. Herramientas de desarrollo

Las herramientas de desarrollo facilitan las tareas de programación y gestión de recursos. A continuación se describen las principales herramientas usadas en el proyecto.

Brackets

Brackets es un editor de texto avanzado escrito en HTML, CSS y Javascript por Adobe Systems y que está orientado al desarrollo web. A diferencia de otros editores, contiene un pequeño servidor Nodejs para probar el código realizado en el momento. Es capaz de mostrar un correcto formato de múltiples lenguajes de programación como Javascript, PHP, Java, HTML y CSS entre otros. Es un programa de libre distribución y sus desarrolladores han publicado una API con la que se pueden desarrollar módulos para aumentar su funcionalidad. Su interfaz es sencilla, con una lista de archivos abiertos a la izquierda y espacio para escribir a la derecha ocupando el máximo espacio posible de la ventana. Sus principales características son:

- Edición rápida
- Acceso a archivos de forma rápida
- Previsualización en tiempo real
- JSLint
- Soporte para LESS
- Integración con Theseus
- Extensible

Git

Es un sistema de control de versiones cuya función principal es la de mantener un registro de cambios locales realizados en un directorio y coordinar el trabajo sobre esos archivos entre un grupo de gente. Se usa mayoritariamente como gestión de código a la hora de desarrollar *software* de manera distribuida. Fue creado por Linus Torvalds en el año 2005 para el desarrollo del *kernel* de Linux con la finalidad de coordinar el trabajo con otros desarrolladores [5]. A Torvalds no le convencían otros sistemas de control de versiones disponibles. Una de las características de este sistema es que cada directorio de Git en cada ordenador es un repositorio completo con histórico y rastreo de versiones independiente de cualquier servidor central. Esto hace posible que, en caso de perderse el repositorio remoto, este se pueda rehacer desde uno perteneciente a los desarrolladores. Sus principales características son:

- Soporte para desarrollo no lineal
- Desarrollo distribuido
- Compatibilidad con varios protocolos de comunicaciones, como HTTP, *File Transfer Protocol* (FTP) o *Secure SHell* (SSH)
- Manejo eficiente de proyectos grandes

Bower

Bower es una herramienta de gestión de dependencias utilizado principalmente para gestionar componentes HTML, CSS y Javascript y obtener las últimas versiones de los mismos con tan sólo usar un comando de consola. Para instalar Bower, se requiere tener instalado previamente Nodejs, npm y git. La forma más sencilla de instalar un nuevo paquete es buscar en Internet el nombre del paquete que deseemos y posteriormente ejecutar en una consola de comandos *bower install nombre_paquete*. También es posible tener un archivo llamado *bower.json* en el que se guardan todas las dependencias del proyecto (con nombre de paquete y versión), de esta forma al ejecutar *bower install*, se instalarán todos los paquetes del fichero, en la versión especificada. Es posible también utilizar sentencias condicionales en las versiones.



Capítulo 2

Metodologías de desarrollo

Una metodología de desarrollo *software*, es un *framework* utilizado para estructurar, planificar y controlar el proceso de desarrollo de *software*. Tanto escogiendo metodologías tradicionales o ligeras, de su correcta implantación en el proyecto dependerá mucho el éxito o fracaso del mismo.

2.1. ¿Por qué usar metodologías?

Hoy en día el número de empresas y proyectos que fallan en su ejecución por una mala gestión es muy alto, un gran número de proyectos son ejecutados sin usar una metodología clara. Esto es debido, en numerosas ocasiones, a que el cliente encuentra dificultad a la hora de ver el valor inmediato de los requerimientos del negocio, análisis de casos de uso y especificaciones de diseño que se requieren para generar un producto de calidad. En vez de eso, sienten que pueden simplemente decir lo que desean y que los desarrolladores serán capaces de dárselo sin hacer preguntas adicionales. Sin una metodología clara, los proyectos se suelen entregar tarde, con sobrecostos, y en muchos casos sin alcanzar las expectativas del cliente o del usuario final. Puede incluso llevar al fracaso del proyecto.

Algunas de las ventajas de implantar una metodología bien definida son las siguiente:

- Permite realizar mejores estimaciones.
- Entregar sistemas estables.
- Mantener al usuario informado.
- Preveer futuras tareas.
- Gestión del riesgo identificando posibles problemas.
- Mejor gestión de tiempo.

Cuando una metodología no se implementa correctamente, pueden surgir numerosos problemas. Por ejemplo, una pérdida de comunicación entre el cliente y el equipo de desarrollo lleva a desarrollar sistemas que no cumplen las expectativas del cliente. Además el mal uso de conceptos o procesos de una metodología de desarrollo conduce a menudo a desarrollar sistemas con numerosos defectos.

Por otro lado, la calidad de vida del equipo humano también se ve afectada por la falta de uso de metodologías. Muchos desarrolladores sufren de falta de moral y motivación a causa de los continuos cambios de alcance o parches debidos a la falta de definición en las tareas [21]. Por el contrario, usando los procesos de desarrollo adecuados, se puede gestionar debidamente el alcance de un proyecto y evitar muchos problemas comunes de desarrollo *software*. Siempre ocurrirán imprevistos, pero con una metodología adecuada, estos son de poco impacto y mínima ocurrencia.

No todas la metodologías de desarrollo funcionan con todos los proyectos. Elegir la adecuada también es uno de los retos que tienen que afrontar los desarrolladores. Algunas metodologías funcionan mejor con proyectos grandes y de larga duración, y otras son más adecuadas para proyectos medianos y pequeños con equipos de desarrollo más modestos. Sin embargo es un hecho que, aplicar una metodología de desarrollo a un proyecto, aumenta mucho las posibilidades de que termine con éxito.

2.2. Metodologías tradicionales vs ágiles

Dentro de las metodologías de desarrollo *software*, se puede encontrar una división entre metodologías tradicionales (o metodologías pesadas) y metodologías ágiles.

Las metodologías de desarrollo tradicionales se caracterizan por tener una serie secuencial de fases: requisitos, análisis, diseño, implementación, pruebas y despliegue. Primero, los requisitos del cliente son meticulosamente documentados. Después se analiza y diseña la arquitectura general del *software* y en la siguiente fase comienza la implementación, finalizando con el período de pruebas y el despliegue. La idea principal es tener una visión completa del proyecto antes de su inicio y realizar la implementación de acuerdo al esquema planificado.

Algunas características de este tipo de metodologías son:

- Desarrollo predictivo.

- Las decisiones se toman en las primeras fases del ciclo de vida y se especifican mediante modelos gráficos.
- Las decisiones se interpretan y materializan en la fase de desarrollo.
- Orientado a la documentación.
- Orientado al proceso.

Las metodologías ágiles son menos rigurosas a la hora de gestionar la documentación. La clave reside en que el desarrollo se realiza de manera incremental e iterativa, en la que las fases de diseño se ejecutan una y otra vez. Como resultado, el *software* deja de ser tan rígido para convertirse en una estructura más orgánica con mayor interactividad entre sus componentes. Se le da mayor importancia a la adaptabilidad y a las pruebas de integración. Algunas características destacadas de este tipo de metodologías son:

- Preparado para soportar cambios.
- Desarrollo iterativo.
- Pruebas de integración.
- Orientado al código.
- Orientado a las personas.

En la Tabla 2.1 se presenta una comparativa entre las metodologías tradicionales y ágiles.

2.3. Metodología escogida

Se ha optado por implementar metodologías ágiles en este proyecto ya que, debido al limitado tiempo para su desarrollo, se necesita la máxima flexibilidad posible. Por otro lado, las continuas reuniones entre cliente (tutor) y desarrollador (alumno) son beneficiosas a la hora de realizar revisiones y correcciones de pequeñas iteraciones del proyecto y poder aplicar una mejora continua. Al haber escogido esta metodología, no se requiere que todos los requisitos estén definidos al principio del proyecto, si no que se puede definir un conjunto básico de requisitos y en las revisiones añadir más o adaptarlos según se crea conveniente.

TRADICIONALES	ÁGILES
Proceso secuencial	Proceso iterativo e incremental. Fases en paralelo
Estructurado	Flexible. Adaptable. Mejora continua
Grandes y pocas entregas de <i>software</i>	Entregas pequeñas y frecuentes de <i>software</i> , que siempre aportan valor
Un gran proyecto	Muchos pequeños proyectos
Adecuado para situaciones donde los cambios no son frecuentes	Adecuado cuando el usuario no conoce al detalle todas las necesidades al principio, y, además, serán cambiantes
El usuario participa fundamentalmente en la fase inicial de toma de requisitos	El usuario está involucrado constantemente en el proyecto. Comunicación fluida
Buena definición de requisitos al principio	Los requisitos se van desgranando cuando se van a abordar
Plazos y costes son conocidos al principio	Ámbito y costes varían según evoluciona el proyecto
Proyecto bien documentado	La documentación no es prioritaria

Tabla 2.1: Comparativa procesos de desarrollo tradicionales vs ágiles

2.3.1. Scrum

Scrum es una metodología ágil, iterativa e incremental. Aunque fue creada por empresas dedicadas a productos tecnológicos, es posible adaptar la metodología a todo tipo de proyectos con requisitos inestables o cambiantes para otros negocios, como desarrollo de *hardware*, programas de *marketing* o iniciativas de equipos de ventas.

En Scrum participa un equipo de trabajo multidisciplinar y autoorganizado que debe actuar como una unidad para alcanzar un objetivo común manteniendo una colaboración activa tanto de forma presencial como *online*.

Uno de los puntos clave de Scrum es la asunción de que el cliente cambiará de opinión sobre lo que quiere o necesita y que se presentarán retos impredecibles para los cuales una planificación previa no son suficientes. De este modo, la metodología implica que el problema no puede entenderse en su totalidad o ser totalmente definido previamente, de manera que se enfoca en maximizar la capacidad del equipo de realizar entregas rápidas y la adaptabilidad a nuevos requisitos y cambios, tanto del cliente como el mercado.

Scrum es relativamente reciente. En 1986 Hirotaka Takeuchi e Ikujiro Nonaka presentaron el artículo “New New Product Development Game” para la Harvard Business Review centrándose en la innovación continua, incremental y en espiral [20]. Los autores describían una nueva aproximación al desarrollo comercial de productos, que llamaron la “aproximación rugby”, ya que todo el proceso es llevado a cabo por un equipo multifun-

cional a través de fases solapadas en dónde el equipo “intenta avanzar como una unidad pasándose el balón hacia atrás y hacia delante”.

En 1993 Ken Schwaber y Jeff Sutherland usaron, por primera vez, en sus empresas la aproximación propuesta por Hirotaka e Ikujiro, refiriéndose a ella como Scrum [19]. En 1995 Schwaber y Sutherland, presentaron su artículo “The SCRUM Development Process” en la conferencia OOPSLA (Texas), popularizando de este modo el término y abriendo nuevos horizontes a la expansión de la metodología [9]. Más tarde, en el año 2001, Schwaber trabajó con Mike Beedle para describir el método Scrum en el libro “Agile Software Development with Scrum” y en el año 2002 fundó junto con otros miembros, la Scrum Alliance, iniciando con ello las series de certificación en Scrum [8].

Las principales características de Scrum son las siguientes:

- Apropiaada para proyectos con requisitos inestables o cambiantes.
- Equipo de trabajo multidisciplinar y autoorganizado.
- Basado en un modelo de proceso iterativo y en los valores del manifiesto ágil.
- La velocidad para el desarrollo de un producto está marcada por el eslabón que tiene la menor velocidad en el equipo.
- Se destaca el trabajo en equipo, el aprendizaje constante y una estructura de desarrollo dinámico, flexible a los cambios durante el desarrollo.
- Los requisitos se expresan en forma de lista priorizada de características y capacidades del producto.
- La priorización se hace teniendo en cuenta el riesgo y el valor de negocio.
- Las características más importantes se abordan primero.
- Se realizan entregas iterativas del producto en ciclos cortos y fijos (*time boxed*).
- La planificación es adaptativa. Solo se compromete la próxima entrega.
- Cada entrega contiene un conjunto de características completas que permiten ponerse en explotación.

Para aplicar Scrum con eficiencia, el equipo de trabajo ideal se compone de entre 5 y 9 personas ya que se consigue un balance entre la dificultad de comunicación entre equipos mas grandes y los retrasos por imprevistos personales que puedan surgir en equipos más pequeños. En el equipo de trabajo se pueden encontrar los siguientes roles:



- *Product Owner*: es el jefe del producto y la persona que interactúa con el cliente para extraer los requisitos y sus prioridades y le representa en el equipo.
- *Scrum Master*: es el director del proyecto. Será el encargado de que se cumpla lo planificado en cada *sprint*.
- *Development Team*: programadores, diseñadores, personal de pruebas, etc.
- Roles externos: usuarios, clientes, especialistas, etc.

El equipo de trabajo lleva a cabo las diferentes actividades del proceso Scrum que le corresponden según su rol asignado. Este proceso se resume en 6 actividades:

- Creación de la pila de producto: También llamado *product backlog*. Es un listado priorizado de los requisitos del sistema (historias de usuario). Su responsable es el *Product Owner*.
- Creación del *sprint planning*: Se decide que historias de usuario serán implementadas en el próximo *sprint*, se estiman y se dividen en tareas si fuera necesario.
- Ejecución de las tareas de un *sprint*: Se realizan las tareas de diseño, ejecución y pruebas. Cada día se realizan reuniones de equipo (*daily scrum*) de no más de 15 minutos para poner en común las tareas realizadas por cada miembro del equipo.
- Reunión de inspección del producto: participa el equipo y los *stakeholders* (usuarios, clientes...), se entrega un producto potencialmente explotable y se valida. En caso de que algún requisito no funcione correctamente, vuelve al *product backlog*.
- Reunión de retrospectiva del proceso: Reunión de equipo dirigida por el Scrum máster para evaluar el proceso y proponer mejoras.
- Repetición de las iteraciones: Se repite todo el proceso hasta completar la implementación del producto.

2.3.2. Adaptación de Scrum a un TFM

Como se ha comentado en la sección anterior, Scrum está pensado para equipos multidisciplinares en los que suele haber varios integrantes, por lo tanto no es posible aplicar la metodología con todas sus características, sino que hay que realizar una adaptación de las mismas.

Se distribuyen los roles donde el tutor o tutores desempeñarán el rol de propietarios del producto, en conjunto el alumno y el tutor actuarán como *Scrum Master* y el alumno como miembro del equipo. En un TFM es el profesor el que suele conocer el dominio del problema y los alumnos los encargados de su desarrollo, de ahí la asignación de roles anteriormente propuesta. La labor del *Scrum Master* se hace manera conjunta ya que no debe recaer por completo en el alumno (que actúa principalmente como desarrollador) ni en el profesor (que actúa como cliente).

Además de la asignación de roles, se toman en cuenta las siguientes consideraciones:

- El tutor, al actuar como propietario, elabora la pila de producto.
- En la primera reunión se fijan las primeras tareas, se planifican y se fija una fecha para que el alumno las realice.
- En cada fecha de entrega de cada *Sprint*, el alumno realizará una pequeña exposición al tutor.
- La división de las historias en tareas, cuando ésta es única, es opcional.
- Ante la probale imposibilidad de llevar a cabo las *Daily Scrum*, se propone el uso de recursos compartidos para que el profesor pueda comprobar las tareas diarias llevadas a cabo. Estos recursos pueden ser: blog, repositorio, carpetas en la nube, etc.
- La pizarra de Scrum puede ser virtual. Es posible usar soluciones *online* de terceros.



Capítulo 3

Descripción informática

En este capítulo se realiza un análisis del proceso de desarrollo del proyecto, explicando la aplicación de la metodología, los hitos cumplidos en el desarrollo del proyecto e incluyendo toda la información relevante sobre el funcionamiento de la aplicación.

3.1. Aplicación de la metodología Scrum

En este proyecto se aplica la metodología Scrum adaptada a TFM como hemos explicado en la Sección 2.3.2. El tutor del proyecto toma el rol de propietario del producto y el alumno de desarrollador. El rol de *Scrum Master* se divide entre tutor y alumno.

3.1.1. Primera reunión 04/05/2017

En la primera reunión se trata todo lo referente al inicio del proyecto. El propietario del producto explica al desarrollador cual es el objetivo final del TFM: desarrollar un módulo para Drupal para la generación de PDF a partir de catálogos y productos de Ubercart. Se dan también los primeros detalles que debería contener la herramienta. Debido a que el desarrollador desconoce Drupal, se crea la primera tarea técnica de instalación, gestión y estudio de este *framework*. A partir de el *product backlog* del cuál se puede observar su estado tras la primera reunión en la Tabla 3.1, se establece el primer *sprint* (Tabla 3.2) y se da un plazo de 2 a 3 semanas para terminarlo.

ID	Alias	Enunciado	<i>Sprint</i>
HT01	Instalación de Entorno	Yo como desarrollador necesito montar un entorno de Drupal 7 para poder desarrollar la aplicación	#1

Tabla 3.1: *Product Backlog* 04/05/2017



ID	ID Tarea	Descripción	Esfuerzo
HT01	TT01.1	Instalar Drupal 7	8
HT01	TT01.2	Instalar base de datos	8
HT01	TT01.3	Instalar módulo	5
HT01	TT01.4	Instalar tema	5
HT01	TT01.5	Crear contenido	3
HT01	TT01.6	Crear tipos de contenido	3
HT01	TT01.7	Crear bloques	2
HT01	TT01.8	Crear vistas	2

Tabla 3.2: *Sprint* #1

3.1.2. Segunda reunión 25/05/2017

En la segunda reunión se muestra al propietario del producto las tareas realizadas con resultado satisfactorio. Al haber completado con éxito el primer *sprint* y haber demostrado el desarrollador soltura a la hora de utilizar el *framework* de Drupal, se comienza a hablar de la funcionalidad que debe cumplir el TFM. Se establecen nuevas historias de usuario para el *product backlog* estableciendo como prioritarias aquellas relacionadas con la generación de documentos PDF. Se actualiza el *product backlog* como se puede observar en la Tabla 3.3 y se establece el nuevo *sprint* (Tabla 3.4) con una duración de una semana.

ID	Alias	Enunciado	<i>Sprint</i>
HT01	Instalación de Entorno	Yo como desarrollador necesito montar un entorno de Drupal 7 para poder desarrollar la aplicación	#1
H01	Botón de generación de PDF de producto	Yo como usuario necesito ver un botón de generación de PDF para Productos para poder generar un documento PDF	#2
H02	Generación de PDF de producto	Yo como usuario debo ser capaz de generar PDF a partir de Productos para su posterior consulta offline	#2
HT02	Creación de vistas programáticamente	Yo como desarrollador debo crear las vistas JSON del módulo programáticamente para que el usuario no tenga que realizar trabajo extra en la fase de instalación	#2

Tabla 3.3: *Product Backlog* 25/05/2017

ID	ID Tarea	Descripción	Esfuerzo
H01	T01.1	Crear Hooks	2
H02	T02.1	Integrar librería pdf	5
H02	T02.2	Programar funciones de conversión a PDF según la librería	8
H02	T02.3	Añadir funcionalidad a botón H02	2
HT02	TT02.1	Añadir vistas por código en el fichero de instalación	5

Tabla 3.4: *Sprint* #2

3.1.3. Tercera reunión 01/06/2017

Tras la implementación del segundo *sprint* se lleva a cabo la revisión de las tareas realizadas. En este momento el propietario del producto informa que hay nuevas historias de usuario que implican cambios en la idea inicial del procedimiento. Además de la generación PDF, debe ser posible que un usuario administrador cree y gestione nuevas plantillas que serán aplicadas en el momento de la generación. Se pide también que además de la generación de PDF de productos independientes se implemente la generación de PDF de catálogos de productos. Al haber demasiadas historias de usuario en el *product backlog* como se puede ver en la Tabla 3.5 se decide dividir las tareas en dos *sprints* que serán ejecutados en sucesión. La definición del tercer *sprint* se puede ver en la Tabla 3.6 la duración del mismo será de 2 semanas.

3.1.4. Cuarta reunión 16/06/2017

Se lleva a cabo la cuarta reunión revisando las tareas realizadas en el tercer *sprint*. El resultado es satisfactorio y se planifica el siguiente conjunto de tareas a realizar a partir de las historias de usuario restantes en el *product backlog*. El estado del product backlog se mantiene de acuerdo a lo definido en la Tabla 3.5. Se define el *sprint* de la Tabla 3.7. La menor complejidad de estas tareas, ya que están muy relacionadas con las tareas del tercer *sprint* y la experiencia previa, permite que el tiempo asignado para estas tareas sea de una semana.

3.1.5. Quinta reunión 23/06/2017

Se muestran los resultados de la implementación de las tareas del cuarto *sprint*. El resultado es satisfactorio y se continúa con la elaboración de la documentación para el acTFM.



ID	Alias	Enunciado	<i>Sprint</i>
HT01	Instalación de Entorno	Yo como desarrollador necesito montar un entorno de Drupal 7 para poder desarrollar la aplicación	#1
H01	Botón de generación de PDF de producto	Yo como usuario necesito ver un botón de generación de PDF para Productos para poder generar un documento PDF	#2
H02	Generación de PDF de producto	Yo como usuario debo ser capaz de generar PDF a partir de Productos para su posterior consulta offline	#2
HT02	Creación de vistas programáticamente	Yo como desarrollador debo crear las vistas JSON del módulo programáticamente para que el usuario no tenga que realizar trabajo extra en la fase de instalación	#2
H03	Creación de plantillas de producto	Yo como administrador debo ser capaz de crear plantillas para la generación de PDFs de Productos para que el usuario genere un PDF atractivo	#3
H04	Asignación de plantillas de producto	Yo como administrador necesito poder asignar una plantilla de PDF a Productos para que se genere un PDF con los elementos posicionados de acorde a la plantilla	#3
H05	Mover elementos en plantillas de producto	Yo como administrador necesito poder mover elementos en una plantilla pdf de producto para posicionarlos donde crea conveniente	#3
H06	Borrado de plantillas de producto	Yo como administrador debo ser capaz de borrar plantillas PDF de producto para que no se acumulen plantillas sin uso	#3
H07	Acceso a sección de gestión de plantillas	Yo como administrador necesito un botón en la sección de administración para poder acceder a la sección de gestión de plantillas	#3
H08	Botón de generación de PDF de catálogo	Yo como usuario necesito ver un botón de generación de PDF para catálogos para poder generar un documento PDF	#4
H09	Generación de PDF de catálogo	Yo como usuario debo ser capaz de generar PDFs a partir de catálogos para su posterior consulta offline	#4
H10	Creación de plantillas de catálogo	Yo como administrador debo ser capaz de crear plantillas para la generación de PDFs de catálogos para que el usuario genere un PDF atractivo	#4
H11	Asignación de plantillas de catálogo	Yo como administrador necesito poder asignar una plantilla de PDF a catálogos para que se genere un PDF con los elementos posicionados de acorde a la plantilla	#4
H12	Mover elementos en plantillas de catálogo	Yo como administrador necesito poder mover elementos en una plantilla pdf de catálogo para posicionarlos donde crea conveniente	#4
H13	Borrado de plantillas de catálogo	Yo como administrador debo ser capaz de borrar plantillas PDF de catálogo para que no se acumulen plantillas sin uso	#4

Tabla 3.5: *Product Backlog* 01/06/2017

ID	ID Tarea	Descripción	Esfuerzo
H03	T03.1	Crear tablas de base de datos	8
H03	T03.2	Crear vistas de administración	8
H03	T03.3	Crear funciones en <i>back-end</i> de creación de plantillas	5
H04	T04.1	Crear funciones <i>back-end</i> para la asignación de plantillas	5
H04	T04.2	Crear vista para la asignación de plantillas	5
H05	T05.1	Integración de librerías <i>front-end</i>	8
H05	T05.2	Creación de vistas	3
H05	T05.3	Funciones de edición	8
H05	T05.4	Funciones de actualización de base de datos	5
H06	T06.1	Vistas para borrado	3
H06	T06.2	Funciones de borrado de base de datos	3
H07	T07.1	Creación de botón en <i>back-end</i>	5

Tabla 3.6: *Sprint #3*

ID	ID Tarea	Descripción	Esfuerzo
H08	T08.1	Crear Hooks	2
H09	T09.1	Programar funciones de conversión a PDF según la librería	5
H09	T09.2	Añadir funcionalidad a botón H02	2
H10	T10.1	Crear vistas de administración	5
H10	T10.2	Crear funciones en back-end de creación de plantillas	5
H11	T11.1	Crear funciones back-end para la asignación de plantillas	5
H11	T11.2	Crear vista para la asignación de plantillas	5
H12	T12.1	Creación de vistas	3
H12	T12.2	Funciones de edición	5
H12	T12.3	Funciones de actualización de base de datos	5
H13	T13.1	Vistas para borrado	3
H13	T13.2	Funciones de borrado de base de datos	3

Tabla 3.7: *Sprint #4*

3.2. Desarrollo del proyecto

En esta sección se expondrá el proceso de desarrollo del TFM. Según lo expuesto en la Sección 3.1, el desarrollo está dividido en cuatro *sprints* con sus diferentes tareas y cada una de ellas consta de diseño, desarrollo y pruebas. En este documento se explicarán en detalle las Historias y Tareas con mayor relevancia.

3.2.1. Primer *sprint*

Como se puede observar en la Tabla 3.2 de la sección anterior, el primer *sprint* tiene sólo una historia de usuario: “Instalación de Entorno”, que comprende ocho tareas básicas para desplegar y comprender el funcionamiento de un entorno Drupal.

Requisitos de Drupal

Para poder ejecutar una aplicación Drupal se tienen que cumplir los siguientes requisitos:

- Espacio en disco mínimo de 60MB.
- Servidor web con soporte PHP como Apache, Nginx ó Microsoft IIS.
- Base de datos MySQL, MariaDB, Percona Server, PostgreSQL o SQLite.
- PHP 5.2.5 mínimo

Se decide utilizar una infraestructura Windows, Apache, MySQL, PHP (WAMP) con lo que se cumplen todos los requisitos mínimos de Drupal. El esquema de WAMP se puede ver en la Figura 3.1. Realizar una instalación para WAMP es sencillo, simplemente hay que descargar un instalador de la página web de WAMP¹ y seguir los pasos del wizard de instalación como se puede observar en la Figura 3.2. Una vez completada la instalación, se arrancan los servicios de WAMP navegando a la carpeta en la que se han instalado los componentes y ejecutando wampmanager.exe como se ve en la Figura 3.3.

¹<http://www.wampserver.es/>

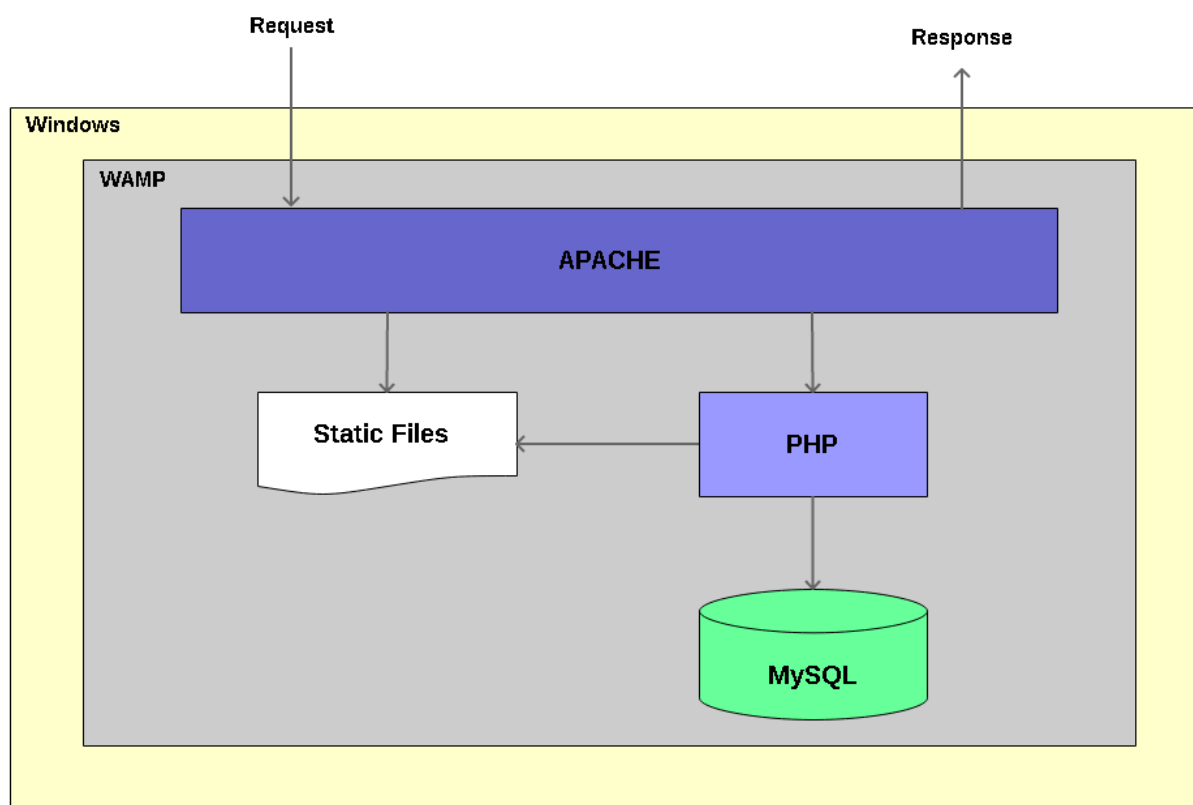


Figura 3.1: Arquitectura WAMP

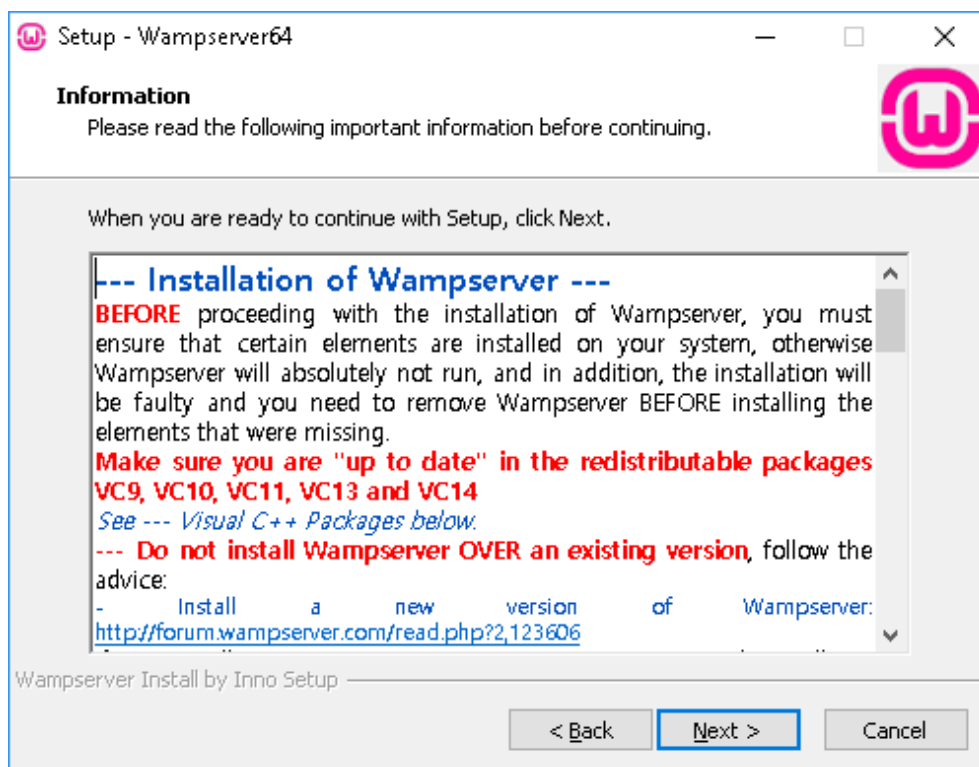


Figura 3.2: Instalación de WAMP



Nombre	Fecha de modifica...	Tipo	Tamaño
alias	10/05/2017 21:40	Carpeta de archivos	
apps	10/05/2017 21:41	Carpeta de archivos	
bin	10/05/2017 21:43	Carpeta de archivos	
cgi-bin	10/05/2017 21:40	Carpeta de archivos	
lang	10/05/2017 21:41	Carpeta de archivos	
logs	10/05/2017 22:01	Carpeta de archivos	
scripts	10/05/2017 21:41	Carpeta de archivos	
tmp	22/06/2017 21:59	Carpeta de archivos	
www	29/06/2017 18:42	Carpeta de archivos	
barimage.bmp	31/12/2010 8:39	Archivo BMP	5 KB
images_off.bmp	01/08/2016 16:21	Archivo BMP	27 KB
images_on.bmp	01/08/2016 16:22	Archivo BMP	27 KB
install-english.txt	16/10/2016 15:09	Archivo TXT	4 KB
license-english.txt	06/11/2015 10:00	Archivo TXT	8 KB
read_after_install-english.txt	08/03/2016 18:45	Archivo TXT	2 KB
unins000.dat	10/05/2017 21:47	Archivo DAT	950 KB
unins000.exe	10/05/2017 21:40	Aplicación	1.369 KB
uninstall_services.bat	10/05/2017 21:40	Archivo por lotes ...	1 KB
wampmanager.conf	10/05/2017 21:47	Archivo CONF	2 KB
wampmanager.exe	03/09/2008 15:46	Aplicación	1.205 KB
wampmanager.ini	20/06/2017 20:35	Opciones de confi...	497 KB
wampmanager.tpl	22/08/2016 10:42	Archivo TPL	24 KB

Figura 3.3: Archivos de WAMP

Instalación de Drupal

Hay varias formas para instalar Drupal y sus extensiones. Inicialmente se intentó usar la herramienta Composer con la cual poder crear un archivo composer.json con todos los parámetros de la instalación de Drupal, incluyendo módulos extra, y posteriormente ejecutar la instalación con un único comando de consola. Sin embargo, tras múltiples problemas relacionados con la plataforma Windows elegida, y, dado que el proceso de instalación de Drupal en sí no está incluido dentro del alcance de los propósitos del TFM, se decidió utilizar un proceso de instalación manual.

Para instalar Drupal manualmente, el proceso es el siguiente:

1. Descargar y extraer Drupal: El *framework* no requiere instalación propiamente dicha, tan sólo hace falta descargar el archivo comprimido de la web de Drupal² y extraer los ficheros comprimidos dentro de la carpeta de aplicaciones web en la instalación de WAMP.
2. Crear base de datos: En el servidor de MySQL incluido en WAMP, se crea la base de datos para la aplicación de Drupal.
3. Ejecución de scripts de Drupal: Finalmente se navega a la aplicación creada en el punto 1 (ejemplo: <http://localhost/drupal>) y se siguen los pasos para completar la instalación. Esto ejecutará los scripts necesarios para completar la instalación de Drupal como se puede ver en la Figura 3.4.

Expandiendo Drupal

Como CMS, Drupal contiene mecanismos para que un administrador pueda agregar contenido y funcionalidad extra. Esto se realiza desde los menús de administración en la aplicación web. Para acceder a esta sección, se navega a la aplicación web del servidor local (creado en la sección anterior) y se introducen los credenciales de un usuario con privilegios, creado en el paso 3 de la instalación manual de Drupal. En la Figura 3.5 se puede ver el menú de administración.

Las siguientes acciones sobre la instalación de Drupal se han llevado a cabo para el primer *sprint*:

²<https://www.drupal.org/project/drupal>

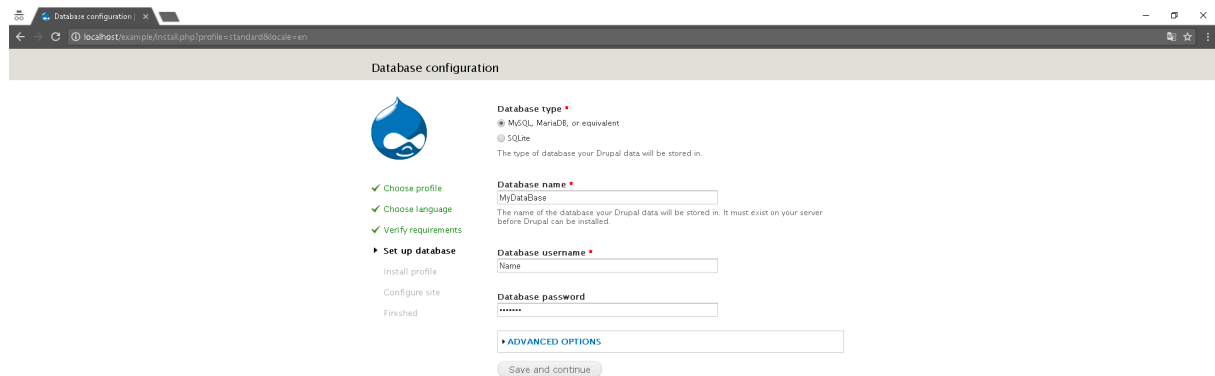


Figura 3.4: Instalación de Drupal

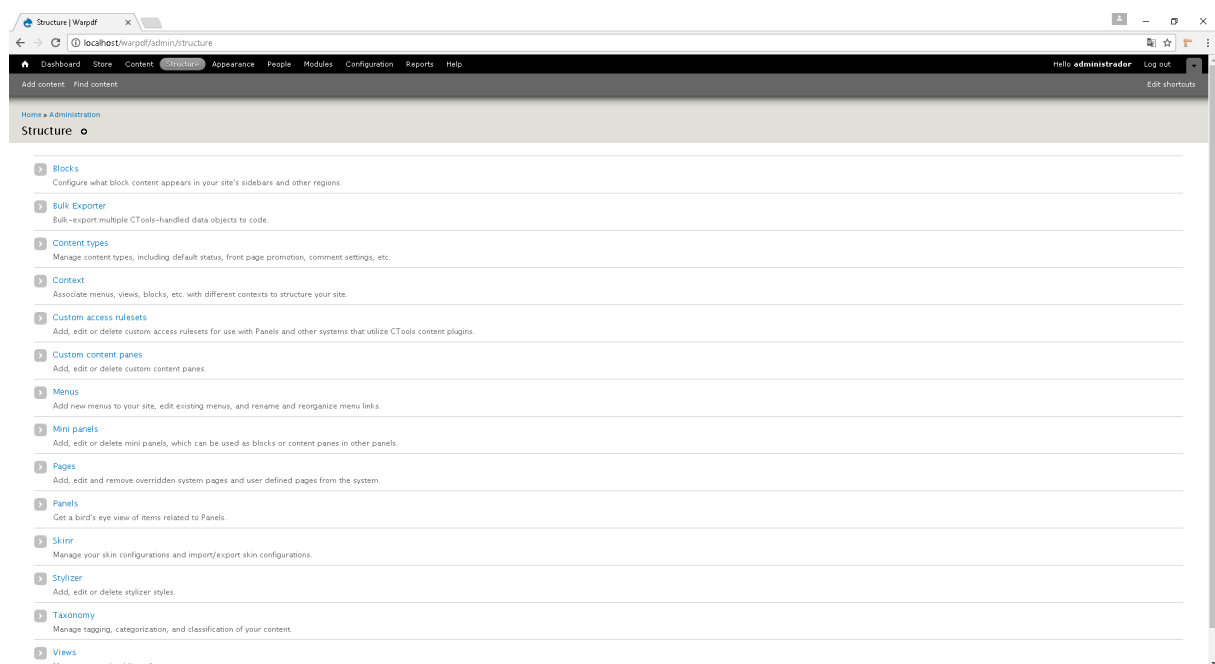


Figura 3.5: Administración de Drupal

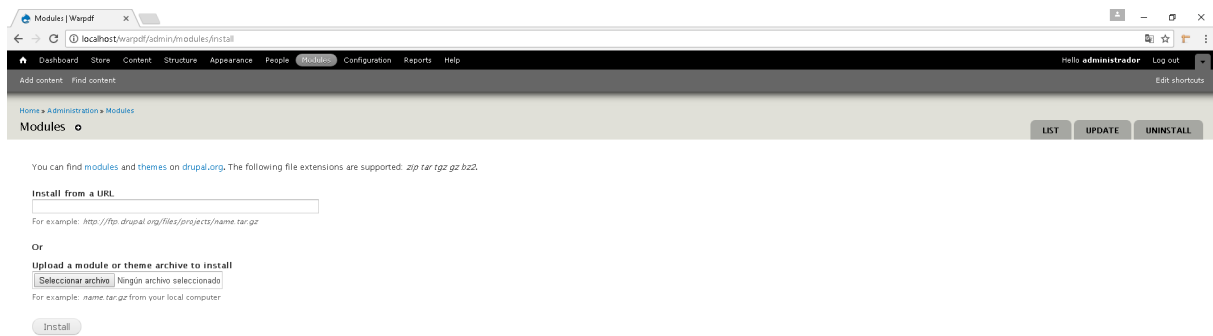


Figura 3.6: Formulario de añadir módulo a Drupal

- **Instalación de módulo:** Para instalar un módulo de Drupal manualmente, hay que realizar una búsqueda del mismo en su web³ y descargar su fichero comprimido. Una vez descargado, se añade el módulo a Drupal mediante la opción “Install new module” del menú “Modules” como se puede observar en la Figura 3.6. Una vez añadido, el módulo y sus dependencias, se activa desde la lista de módulos, como se muestra en la Figura 3.7. Para ganar tiempo en el futuro del desarrollo, se instala directamente el módulo de Ubercart y sus dependencias.
- **Instalación de tema:** Se instala el tema Acquia Prosper de manera manual. Para ello se realiza una búsqueda en la web de descarga de temas⁴ y se obtiene el fichero comprimido. Luego desde la sección “Appearance”, se elige la opción “Install new theme” y se elige el archivo descargado como se puede ver en la Figura 3.8. Por último se instalan sus dependencias y se activa desde el listado de temas de la sección “Appearance”.
- **Creación de contenido:** Se crean contenidos, tipos de contenido, vistas y bloques de prueba para familiarizarse con la diferente funcionalidad de Drupal y productos para el posterior desarrollo del proyecto.

³https://www.drupal.org/project/project_module

⁴https://www.drupal.org/project/project_theme

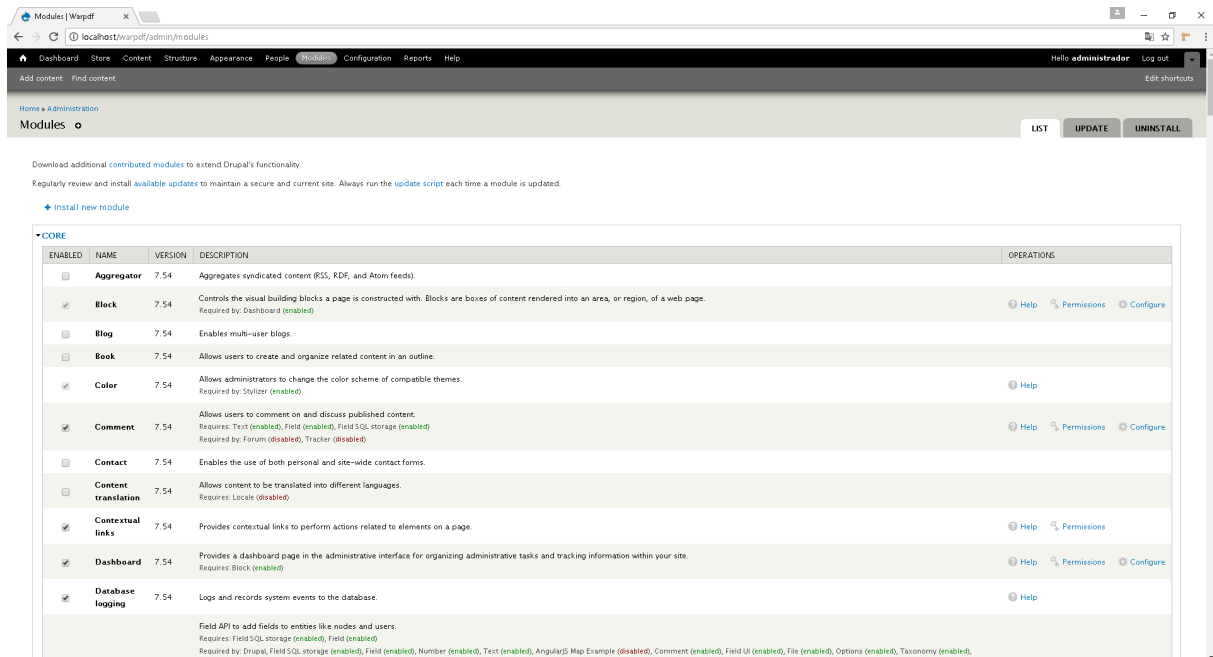


Figura 3.7: Lista de módulos instalados en Drupal

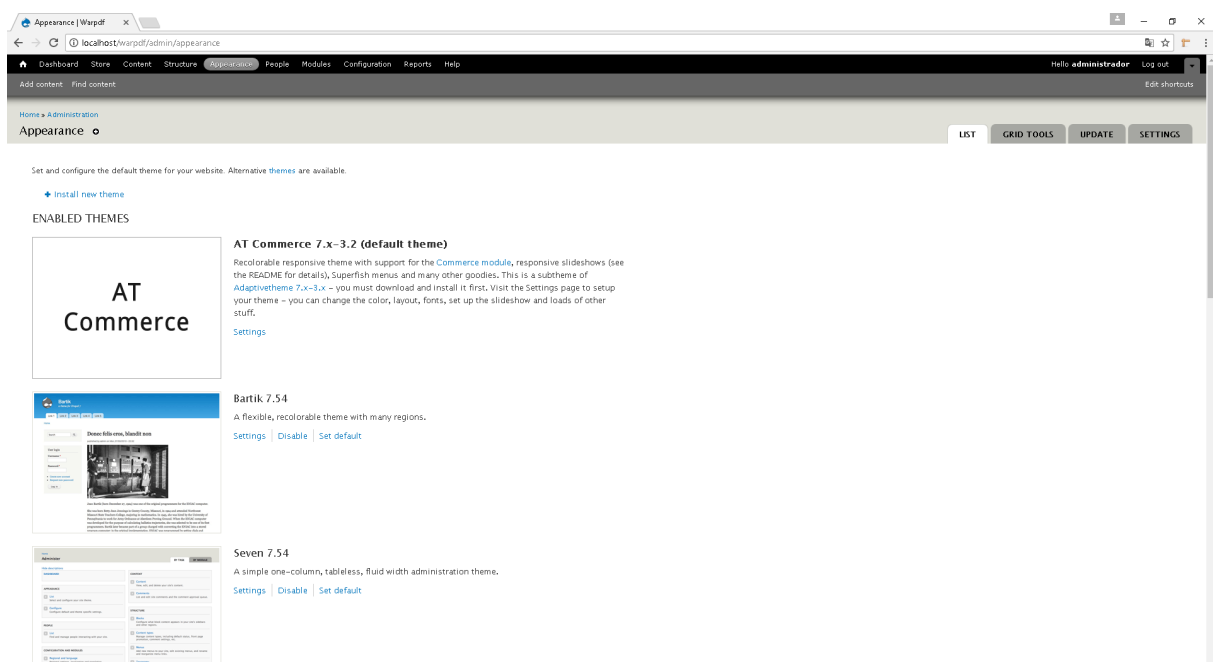


Figura 3.8: Agregar nuevo tema a Drupal

Creación de un módulo de Drupal

Para crear un nuevo módulo de Drupal 7, se navega a la carpeta “sites/all/modules/” de la instalación de Drupal, y se crea un nuevo directorio con el nombre de la nueva extensión que en este caso será “pdf_warp”. En este nuevo directorio se crea un fichero “pdf_warp.info” con el siguiente contenido:

```
name = PDF Warp
description = A module that provides Drupal 7 with PDF functionality for Ubercart
core = 7.x
package = PDF Warp
```

Esto indicará a Drupal las propiedades básicas del nuevo módulo para que lo reconozca como tal y se pueda activar desde el menú “Modules” de la administración.

Por último se crea en el directorio un fichero pdf_warp.module que tendrá el código PHP con la funcionalidad de la extensión.

3.2.2. Segundo *sprint*

Este *sprint* consta de tres historias de usuario que se dividen en cinco tareas, como está reflejado en el Tabla 3.4 de la sección anterior. A partir de este punto empieza el desarrollo de la funcionalidad del TFM.

Diseño

La arquitectura del módulo constará de un directorio base que contiene el archivo .info de descripción del módulo, el archivo .install con las instrucciones PHP a ejecutar durante la instalación, el archivo .module que contiene el código del módulo como la implementación de *hooks* de Drupal y los archivos .tpl con las plantillas de PHP. Dentro del mismo directorio y como se puede observar en la Figura 3.9 se incluirán una carpeta “css” para los estilos del módulo y una carpeta “js” para todo el contenido *front-end* dinámico.

La aplicación inicialmente seguirá el diseño de la Figura 3.10 con Drupal como fuente de datos, y comunicaciones JSON con el *front-end* para la generación de PDF.

Desarrollo

■ H01 “Botón de generación de PDF de producto”

Se decide abordar de principio la historia de usuario H01 “Botón de generación de PDF de producto” ya que presenta menor complejidad y es más necesaria para la

Nombre	Fecha de modifica...	Tipo	Tamaño
.git	28/06/2017 22:06	Carpeta de archivos	
css	21/06/2017 20:18	Carpeta de archivos	
js	22/06/2017 20:03	Carpeta de archivos	
pdf_warp.info	27/05/2017 21:30	Archivo INFO	1 KB
pdf_warp.install	13/06/2017 20:56	Archivo INSTALL	2 KB
pdf_warp.module	28/06/2017 19:41	Archivo MODULE	18 KB
pdf-warp-catalog-generation.tpl.php	22/06/2017 20:24	Archivo PHP	1 KB
pdf-warp-catalog-template.tpl.php	21/06/2017 21:58	Archivo PHP	2 KB
pdf-warp-generation.tpl.php	22/06/2017 20:01	Archivo PHP	9 KB
pdf-warp-template.tpl.php	21/06/2017 20:32	Archivo PHP	11 KB

Figura 3.9: Carpetas de módulo pdf_warp

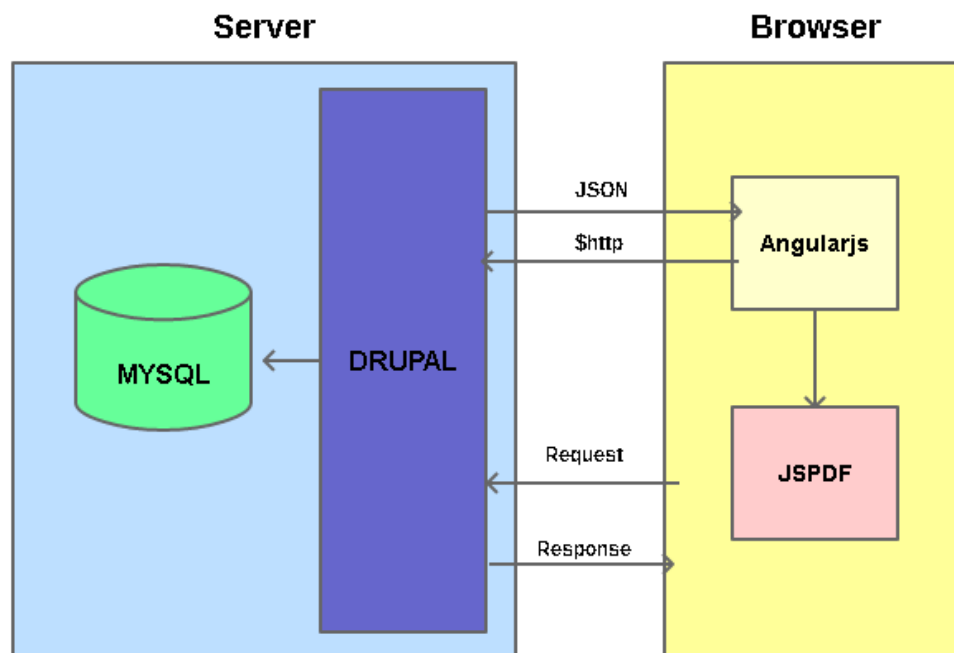


Figura 3.10: Arquitectura de la aplicación

interacción humano/máquina. Para mostrar un botón en los productos de Ubercart, es necesario implementar el *hook form_alter* en el archivo `.module`. Esta es la función a la que Drupal llama cada vez que pinta un formulario, de modo que se usará para comprobar si se está renderizando un formulario de producto de Ubercart y, de este modo, añadir el botón “Generar PDF” a cada producto. El código de este *hook* es el siguiente:

```
function pdf_warp_form_alter(&$form, $form_state, $form_id){
  if (strpos($form_id, 'uc_product_add_to_cart_form') !== false) {
    $form['to_pdf_button'] = array(
      '#type' => 'submit',
      '#value' => t('Generar PDF'),
      '#submit' => array('pdf_warp_goto_generation')
    );
  }
}
```

En dónde *\$form* contiene la información del formulario que se está renderizando, *\$form_state* contiene la información del estado de formulario y *\$form_id* contiene el id de formulario.

■ HT02 “Creación de vistas programáticamente”

Se necesita crear las vistas de producto y catálogo en JSON para que el código Javascript del cliente pueda trabajar con estos datos. Para ello se hace uso de los *hooks views_api* y *views_default_views*. Debido a que crear una vista programáticamente es complicado, lo más sencillo es crear la vista de forma manual y luego exportar el código generado a *views_default_views* como se observa en la Figura 3.11.

```
function pdf_warp_views_api() {
  // Definir path del módulo
}

function pdf_warp_views_default_views() {
  // Pegar código de la exportación
  $views[$view->name] = $view;
  return $views;
}
```

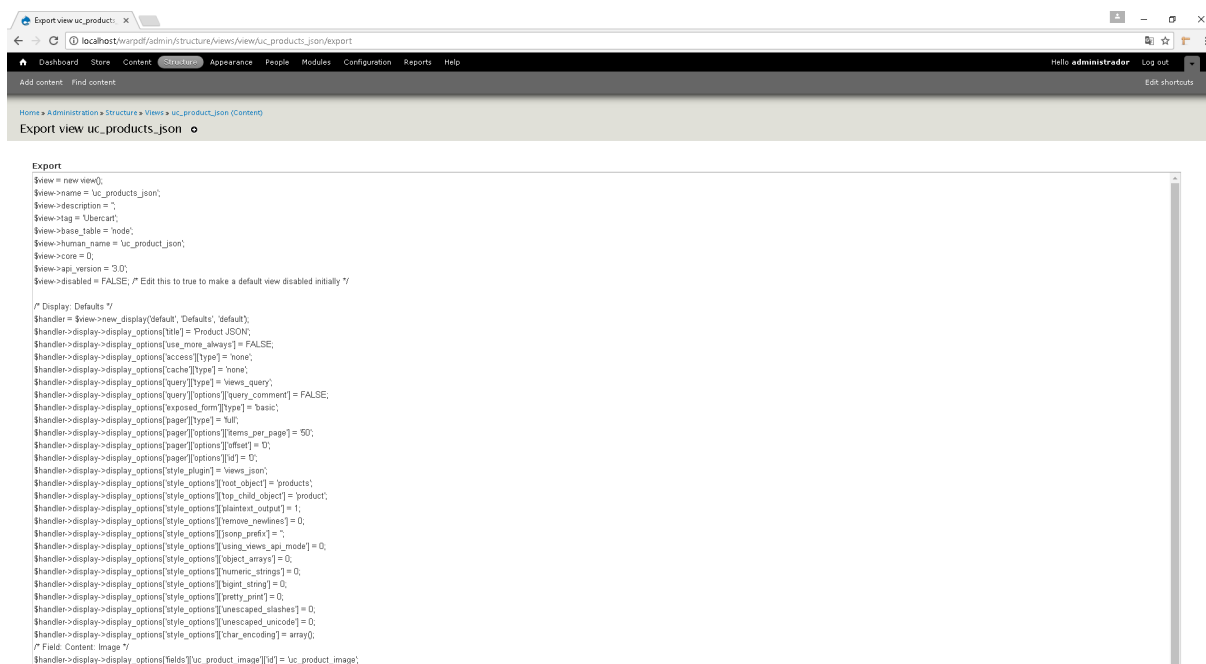



Figura 3.11: Exportación de vista a código PHP

■ H02 “Generación de PDF de producto”

Tras concluir la implementación de la historia H01 y HT02, se empieza con la H02 “Generación de PDF de producto”. En esta historia de usuario se necesita añadir las librerías de generación de PDF, implementar la funcionalidad del lado del cliente y agregar el lanzado de la generación al botón de la historia H01.

Para no sobrecargar las páginas de Drupal con librerías que no necesitan, se cargará el código Javascript localmente en las páginas en las que sea necesario. Con este fin, se crea una nueva página que realizará la generación de PDF.

El proceso de creación de una nueva página requiere implementar los *hooks menu* y *theme* en el archivo .module. Estas funciones son llamadas por Drupal cuando se va a generar el árbol de navegación y los temas.

En el *hook menu*, se añade un nuevo elemento al *array* \$items con su título, descripción, *callback*, argumentos de página y privilegios necesarios para acceder. La implementación es la siguiente:

```
function pdf_warp_menu() {  
    $items['pdf_warp/generate/product/%'] = array(  
        'title'           => 'PDF Warp generación',  
        'description'     => 'Generación de PDF',  
        'page callback'   => 'pdf_warp_generate_product_page',
```

```

    'page arguments' => array(3),
    'access arguments' => array('access content'),
  );
}

```

En este caso se establece que se está creando una página de título “PDF Warp generación”, con llamada función de .module “pdf_warp-generate-product_page” en el paso previo a su renderizado, con un argumento que viene dado por el índice 3 de la url y que se requiere el privilegio de acceso a contenido para su navegación.

Para el *hook theme* se devuelve un array de elementos que contienen un atributo *template* con el nombre del fichero dentro del directorio del módulo que contiene el código de dicha plantilla:

```

function pdf_warp_theme(){
  return array(
    'pdf_warp_generation' => array (
      'template' => 'pdf-warp-generation'
    )
  );
}

```

Teniendo los hooks, se programa la función “pdf_warp-generate-product_page”. Esta función tiene un argumento que será el contenido de “page arguments”. Aquí se realiza la inyección de las librerías de Javascript con el uso de drupal_add_js:

```

function pdf_warp_generate_product_page($id_item) {
  $path = drupal_get_path('module', 'pdf_warp');

  drupal_add_js($path.'/js/pdf_warp_generation.js');
  drupal_add_js($path.'/js/vendor/jspdf.min.js');
  drupal_add_js($path.'/js/vendor/html2pdf.js');

  return theme('pdf_warp_generation');
}

```



Una vez integradas las librerías de Javascript en Drupal se procede a implementar las funciones de generación PDF en el lado del cliente. Para ello se crea dentro de la carpeta “js” de pdf_warp un fichero pdf_warp_generation.js que ya estamos referenciando en la función anterior. En este fichero se hace uso de las librerías de JsPDF para generar un documento PDF del producto con id \$id_item. Para conseguir los datos de este producto en JSON, se hace uso de las vistas definidas previamente mediante una llamada ajax a:

```
base_url + '/store/catalog/' + id + '/json'
```

siendo *base_url* la url de base de la aplicación Drupal e *id* el id del producto. Con esta información en el Javascript se puede generar un pdf de manera programática haciendo uso de las funciones de la librería JsPDF:

```
var doc = new jsPDF('portrait','pt','a4');
var pdf_elements = vm.initTemplate.pdf_elements
doc.setFontSize(16 * 72 / 96);
doc.text(pdf_elements[key].x * 72 / 96 + 40 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 40 * 72 / 96,
        product[index].product.Title
);
doc.setFontSize(10 * 72 / 96);
doc.text(pdf_elements[key].x * 72 / 96 + 40 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 60 * 72 / 96,
        'SKU:' + product[index].product.SKU
);
doc.setFontSize(22 * 72 / 96);
doc.text(pdf_elements[key].x * 72 / 96 + 100 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 300 * 72 / 96,
        product[index].product.Price
);
doc.save('producto.pdf');
```

Con esta funcionalidad, al hacer click en el botón se generará un PDF del producto asociado a dicho botón.

Pruebas

Las pruebas realizadas son las siguientes:

- Se comprueba que el botón “Generar PDF” únicamente se genera para los productos.
- Se prueba el botón de “Generar PDF ” de cada producto para comprobar que se generan documentos PDF distintos en función del producto al que pertenecen.
- Se comprueba que las vistas en JSON devuelven el contenido correcto de un producto.
- Se comprueba que los *hooks* implementados funcionan correctamente.

3.2.3. Tercer *sprint*

El tercer *sprint* consta de cinco historias de usuario que se dividen en doce tareas, como está reflejado en la Tabla 3.6. Al ser más largo y complejo que el resto, ya que se han realizado cambios en la funcionalidad general, se cuenta con un período de dos semanas para su realización.

Diseño

Para este *sprint* se pide el uso de plantillas PDF que servirán para generar los documentos con los elementos en el sitio que decidamos. Para este fin se necesita diseñar una estructura de datos que contenga la plantilla. Contendrá los siguientes campos:

- Nombre: el nombre de la plantilla.
- Tipo: el tipo de plantilla, si de catálogo o de producto.
- Tipo de renderizado: imagen o texto.
- Activo: para saber si la plantilla está activa o no.
- Contenido: El contenido en JSON de la plantilla con sus elementos. Se trata de un objeto JSON que a su vez contiene dos objetos llamados `pdf_elements` y `edit_elements` que contienen una lista de elementos para la generación de PDF y para la edición de la plantilla respectivamente, ya que las posiciones de estos elementos en el modo edición son relativas.

Para almacenar la información de plantilla se diseña una nueva tabla de base de datos que contendrá las plantillas PDF. Esta tabla contará con los siguientes campos:



Appearance				PDF Warp
ID	Nombre	Tipo	Operaciones	ACTIVO
1	Plantilla 1	Catálogo	Editar Borrar Activar	SI
2	Plantilla 2	Producto	Editar Borrar Activar	SI

Figura 3.12: Diseño de pantalla principal de administración

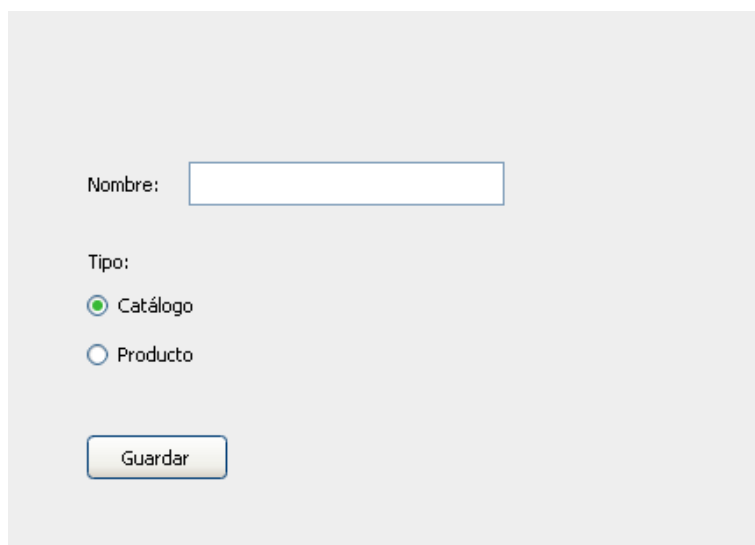
- id: identificador de plantilla de tipo UNSIGNED INT.
- name: nombre de la plantilla de tipo VARCHAR(255).
- type: el tipo de plantilla de tipo VARCHAR(32) con valor por defecto 'catalog' siendo el otro valor 'product'.
- render_mode: el modo de renderizado de tipo VARCHAR(32). Por defecto 'imagen'.
- active: para saber si la plantilla está activa o no de tipo TINYINT con 1 activo y 0 no activo.
- content: El contenido en JSON de tipo TEXT y tamaño medio.

Por otro lado se requiere una sección de administración para que un usuario con privilegios pueda gestionar plantillas. Se decide incluir una pestaña de administración dentro de la sección "Appearance". El diseño de las pantallas se puede ver en las Figuras 3.12, 3.13 y 3.14.

Desarrollo

■ H03 "Creación de plantillas de producto"

El primer paso en la creación de las plantillas es asegurarse de que es posible persistir los datos, de manera que se procede a crear la tabla de la base de datos. Esta tabla debe desplegarse durante la instalación del módulo para asegurar que sea exportable, y debe borrarse una vez se desinstale el módulo. Para ello se crea un nuevo archivo



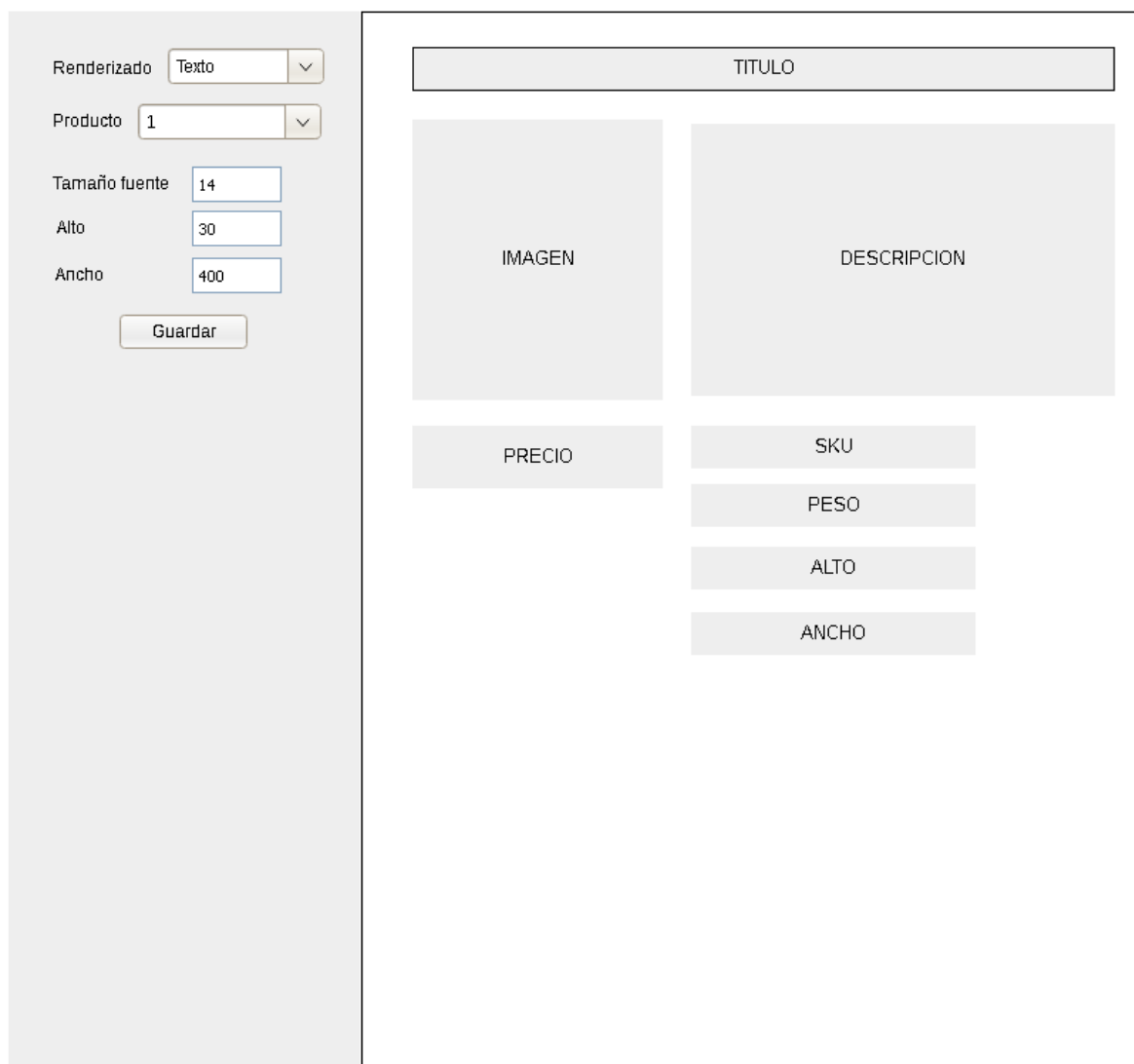
Nombre:

Tipo:

☒ Catálogo

☐ Producto

Figura 3.13: Diseño de creación de plantilla



Renderizado

Producto

Tamaño fuente

Alto

Ancho

TITULO

IMAGEN

DESCRIPCION

PRECIO

SKU

PESO

ALTO

ANCHO

Figura 3.14: Diseño de edición de plantilla



`pdf_warp.install` en la raíz del módulo. En este fichero se implementan los *hooks* *install*, *uninstall* y *schema*. *Install* y *uninstall* como su nombre indica instalaran y desinstalarán nuestro *schema* según se instale o elimine el módulo. *Schema* contiene la definición de la tabla de base de datos. La implementación se puede observar a continuación:

```
function pdf_warp_install() {
    drupal_install_schema('pdf_warp');
}

function pdf_warp_uninstall() {
    drupal_uninstall_schema('pdf_warp');
}

function pdf_warp_schema() {
    $schema['pdf_warp_templates'] = array(
        'description' => t('Tabla de plantillas del módulo PDF Warp.'),
        'fields' => array(
            'id' => array(
                'description' => 'Identificador de plantilla',
                'type' => 'serial',
                'unsigned' => TRUE,
                'not null' => TRUE),
            'name' => array(
                'description' => 'Nombre de la plantilla',
                //...RESTO DE CAMPOS
            ),
            'primary key' => array('id')
        );
    return $schema;
}
```

Para que un usuario con privilegios pueda crear plantillas, necesita una serie de páginas de administración, como por ejemplo una página para ver todas las plantillas, y otra para darlas de alta. Como hemos explicado en el *sprint* anterior, este proceso se realiza mediante el *hook menu* con la diferencia de que las páginas nuevas deberán estar contenidas en la ruta “admin/appearance/” y con privilegios “administer site configuration”. El tipo de la página principal “MENU_LOCAL_TASK”,

esto generará una nueva pestaña en “Appearance”. El tipo de la página para añadir plantilla será `MENU_LOCAL_ACTION` para que genere un botón de “Nueva plantilla”. Las nuevas páginas se definen con el siguiente código dentro del *hook menu* implementado previamente:

```
$items['admin/appearance/pdf_warp'] = array(
  'title'          => 'PDF Warp',
  'description'     => 'Configuración de plantillas',
  'page callback'   => 'pdf_warp_admin_page',
  'access arguments' => array('administer site configuration'),
  'type'           => MENU_LOCAL_TASK,
);

$items['admin/appearance/pdf_warp/add'] = array(
  'title' => t('Nueva plantilla'),
  'page callback' => 'drupal_get_form',
  'page arguments' => array('pdf_warp_form_add_template'),
  'access arguments' => array('administer site configuration'),
  'type' => MENU_LOCAL_ACTION,
);
```

Posteriormente se implementan las funciones de *callback*, *pdf_warp_admin_page* y *pdf_warp_form_add_template*, para generar el contenido de las páginas, además para el formulario de creación de plantilla se requiere implementar una nueva función *pdf_warp_form_add_template_submit* que contendrá el código para crear una nueva plantilla en base de datos. El aspecto que presentan se puede ver en las Figuras 3.15 y 3.16.

Una vez tenemos la plantilla creada, se modifica la generación de PDF de producto para que haga uso de ella en vez de una prefijada. Para ello tan sólo modificamos la función del back que renderiza la página de generación de PDF para que envíe el JSON de nuestra plantilla a la página de generación, en los *settings* de Drupal:



```
function pdf_warp_generate_product_page($id_item) {  
    ...  
    $sql = "SELECT * FROM {pdf_warp_templates} WHERE active=1  
    AND type='product'";  
    $settings['pdf_warp']['template_content'] =  
    db_query($sql)->fetchAll();  
    drupal_add_js($settings, 'setting');  
  
    return theme('pdf_warp_generation');  
}
```

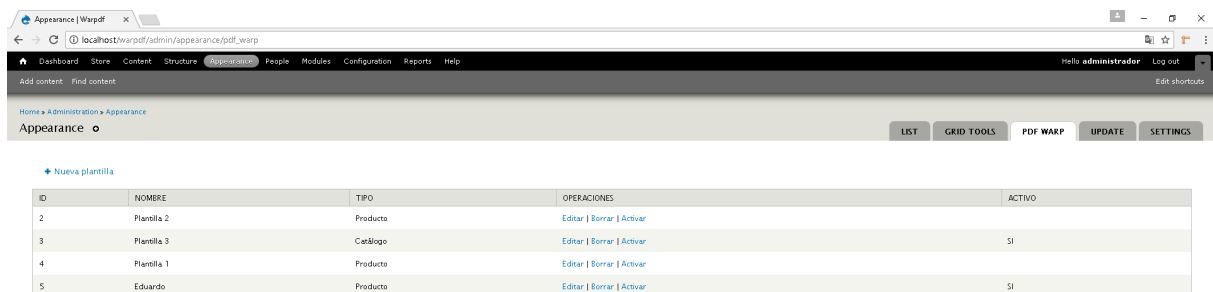


Figura 3.15: Página de administración principal

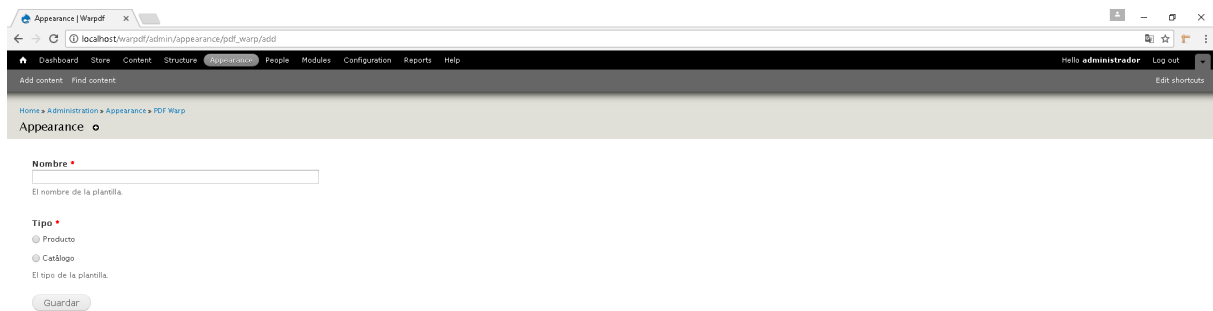


Figura 3.16: Página de creación de plantilla

■ H04 “Asignación de plantillas de producto”

Para la asignación o activación de plantillas el proceso es sencillo. Se crea una nueva página de formulario implementando el *hook menu* y su función de *callback* que se llamará `pdf_warp_template_enable_confirm`. El código que se añade al *hook* es el siguiente:

```
$items['admin/appearance/pdf_warp/enable/%'] = array(
  'title' => t('Activar plantilla'),
  'page callback' => 'drupal_get_form',
  'page arguments' => array('pdf_warp_template_enable_confirm', 4),
  'access arguments' => array('administer site configuration'),
  'access callback' => TRUE,
  'type' => MENU_CALLBACK,
);
```

A esta página se navegará desde un botón contenido en la tabla de plantillas de la pantalla principal de administración. Su funcionamiento será el de activar la plantilla seleccionada y desactivar el resto de plantillas. Para ello se hacen dos operaciones de *update* en base de datos, una que ponga a 0 el campo *active* de todas las plantillas del tipo seleccionado (catálogo o producto) y otra para marcar con 1 el campo *active* de la plantilla seleccionada.

■ H05 “Mover elementos en plantillas de producto”

Esta historia de usuario requiere realizar desarrollos tanto en el *back-end* como en el *front-end* y realizar una comunicación entre ellos. Debido a que la edición de plantillas requiere una interfaz de usuario dinámica e interactiva, se decide integrar AngularJS en el *front-end* para poder utilizar todos los desarrollos de la comunidad que necesitemos. Para ello una de las dependencias del módulo será la extensión AngularJS Bridge que añade cualquier versión del *framework* a Drupal.

Una vez realizada la instalación y activación de la dependencia, se procede a crear las páginas de edición. Se crean dos nuevas entradas mediante el *hook menu* una que contiene la funcionalidad de edición y otra un *endpoint* al que llamar cuando se quieran guardar los cambios. Estas páginas sólo podrán ser accedidas por un usuario con privilegios. Se definen con el siguiente código:

```
$items['admin/appearance/pdf_warp/edit/%'] = array(
  'title' => t('Editar plantilla'),
  'page callback' => 'pdf_warp_edit_template_page',
  'page arguments' => array(4),
  'access arguments' => array('administer site configuration'),
  'type' => MENU_CALLBACK,
);

$items['admin/appearance/pdf_warp/edit/%/save'] = array(
  'title' => t('Guardar plantilla'),
  'page callback' => 'pdf_warp_save_edit',
  'page arguments' => array(4),
  'access arguments' => array('administer site configuration'),
  'type' => MENU_CALLBACK,
);
```

En la función de *callback* de la página de edición, se tienen que cargar todas las librerías *front-end* necesarias y enviar el contenido de la plantilla desde base de datos. El código sería el siguiente:

```
function pdf_warp_edit_template_page($id_template) {
  drupal_add_library('angularjs', 'angularjs');

  $path = drupal_get_path('module', 'pdf_warp');
```

```

drupal_add_css($path.'/css/pdf_warp_template.css');
drupal_add_js($path.'/js/pdf_warp.js');
drupal_add_js($path.'/js/vendor/jspdf.min.js');
drupal_add_js($path.'/js/vendor/html2pdf.js');
drupal_add_js($path.'/js/vendor/bower_.../.../classie.js');
drupal_add_js(
    $path.'/js/vendor/bower_.../.../EventEmitter.js'
);
drupal_add_js(
    $path.'/js/vendor/bower_components/eventie/eventie.js'
);
drupal_add_js(
    $path.'/js/vendor/bower_.../.../get-style-property.js'
);
drupal_add_js(
    $path.'/js/vendor/bower_.../.../get-size.js'
);
drupal_add_js(
    $path.'/js/vendor/bower_.../.../draggabilly.js'
);
drupal_add_js(
    $path.'/js/vendor/bower_.../.../angular-draggabilly.js'
);

$sql = "SELECT * FROM {pdf_warp_templates} WHERE id=".$id_template;
$settings['pdf_warp']['template_content'] = db_query($sql)->fetchAll();
$sql = "SELECT nid FROM {uc_products}";
$settings['pdf_warp']['products'] = db_query($sql)->fetchAll();
drupal_add_js($settings, 'setting');

return theme('pdf_warp_template');
}

```

Como se puede observar, se incuyen librerías extra desde la carpeta *vendor* para poder usar la extensión de AngularJs Draggabilly que nos permite mover elementos mediante *drag and drop* al punto que queramos dentro de un elemento contenedor.

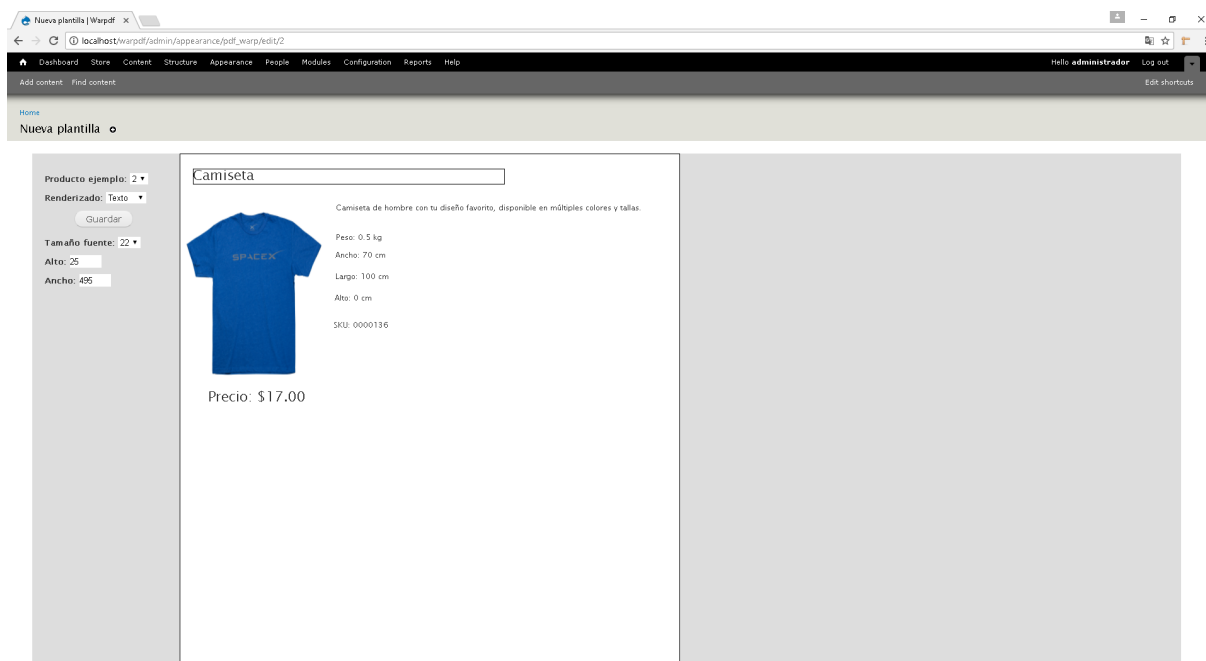


Figura 3.17: Página de edición de plantilla

La página de edición AngularJS se divide en dos ficheros, uno que contiene la lógica de la página llamado `pdf.warp.js` y otro con la el código HTML representando la vista de nombre `pdf-warp-template.tpl.php`. La página resultante se puede ver en la Figura 3.17.

■ H06 “Borrado de plantillas de producto”

El borrado de plantillas de producto también se realiza de manera sencilla. Se crea una nueva página de formulario implementando el *hook menu* y su función de *callback* que se llamará `pdf_warp_template_delete_confirm`. El código que se añade al *hook* es el siguiente:

```
$items['admin/appearance/pdf_warp/delete/%'] = array(
  'title' => t('Borrar plantilla'),
  'page callback' => 'drupal_get_form',
  'page arguments' => array('pdf_warp_template_delete_confirm', 4),
  'access arguments' => array('administer site configuration'),
  'access callback' => TRUE,
  'type' => MENU_CALLBACK,
);
```

A está página se navegará desde un botón contenido en la tabla de plantillas de

la pantalla principal de administración. Al pulsar este botón se navegará a una página de confirmación para borrar la plantilla seleccionada. Para ello se realiza una operación de borrado en base de datos por id de plantilla.

■ H07 “Acceso a sección de gestión de plantilla”

Esta historia de usuario se lleva a cabo de manera conjunta con la historia H03 “Creación de plantillas de producto” ya que al agregar la nueva página de administración de tipo *MENU_LOCAL_TASK* dentro de la navegación de un menú existente, obtenemos un botón de acceso, o, en este caso, una pestaña.

Pruebas

Las pruebas realizadas en este *sprint* son las siguientes:

- Creación de plantillas de producto.
- Borrado de plantillas de producto.
- Edición de plantillas de producto.
- Activado de plantillas de producto.
- Generación de PDF a partir de plantillas de producto.

3.2.4. Cuarto *sprint*

El último *sprint* consta de seis historias de usuario que se dividen en doce tareas, como está reflejado en la Tabla 3.7. Este *sprint* aún siendo similar en tamaño al anterior, tiene menor complejidad ya que podemos reutilizar código y experiencia previa para realizar un desarrollo más ágil, por ello el tiempo estimado es de una semana.

Diseño

Lo único nuevo en cuestión de diseño en este *sprint* es la pantalla de edición de plantillas de catálogos que podemos ver en la Figura 3.18.

Desarrollo

■ H08 “Botón de generación de PDF de catálogo”

Para la creación de un botón que genere documentos PDF a partir de catálogos, se requiere el uso del *hook page_alter* ya que al contrario que con productos, no se está asociando la generación a un único formulario de producto, sino a un catálogo



Productos por página

PRODUCTO 1

PRODUCTO 2

PRODUCTO 3

Figura 3.18: Diseño de edición de plantilla de catálogo

que se muestra en una página completa. El código de implementación del *hook* es el siguiente:

```
function pdf_warp_page_alter(&$page){
    if (strpos($page['content']['system_main']
['products']['#markup'], 'view-uc-catalog') != false){
        $url = explode("/", current_path());
        $page['content']['system_main']['products']['#markup'] =
        $page['content']['system_main']['products']['#markup'].
        '<a href="'. $GLOBALS['base_url'].
        '/pdf_warp/generate/catalog/'.
        $url[1].'";
    }
}
```

Con esta implementación aparecerá un botón en cada catálogo que navegue a la página `/pdf_warp/generate/catalog/id` en la que se desarrollará la generación del PDF de catálogo.

■ “H09 Generación de PDF de catálogo”

Al igual que en el segundo *sprint* se crea la página de generación de PDF a partir de plantillas de catálogo. En el *hook menu* se añade un elemento nuevo, el código es el siguiente:

```
$items['pdf_warp/generate/catalog/%'] = array(
    'title' => 'PDF Warp generación',
    'description' => 'Generación de PDF',
    'page callback' => 'pdf_warp_generate_catalog_page',
    'page arguments' => array(3),
    'access arguments' => array('access content'),
);
```

También es necesario crear nuevos temas, para utilizar las páginas *front-end* de catálogos:



```
function pdf_warp_theme(){
  return array(
    ...
    'pdf_warp_catalog_template' => array (
      'template' => 'pdf-warp-catalog-template'
    )
    ,
    'pdf_warp_catalog_generation' => array (
      'template' => 'pdf-warp-catalog-generation'
    )
  );
}
```

La función de *callback* de la página de generación, será prácticamente igual que la de producto, con la diferencia de que se cargan otros recursos, se envía una plantilla de tipo *catalog* y se devuelve otro tema distinto.

```
function pdf_warp_generate_catalog_page($id_item) {
  ...
  $path = drupal_get_path('module', 'pdf_warp');
  drupal_add_js($path.'/js/pdf_warp_catalog_generation.js');
  drupal_add_js($path.'/js/vendor/jspdf.min.js');
  drupal_add_js($path.'/js/vendor/html2pdf.js');
  ...
  $sql = "SELECT * FROM {pdf_warp_templates} WHERE active=1 AND type='catalog'";
  $settings['pdf_warp']['template_content'] = db_query($sql)->fetchAll();
  drupal_add_js($settings, 'setting');

  return theme('pdf_warp_catalog_generation');
}
```

Habiendo creado la página de generación con las librerías necesarias inyectadas, se puede implementar el código de generación del documento PDF de catálogo en el archivo `pdf_warp_catalog_generation.js`, el cual será similar al de productos con la diferencia de que se generan páginas diferentes dependiendo del número de elementos por página deseado:

```

...
var doc = new jsPDF('portrait', 'pt', 'a4');
var pdf_elements = vm.initTemplate.pdf_elements

angular.forEach(splitProducts, function(product, productIndex) {
  if (productIndex > 0){
    doc.addPage();
  }
  angular.forEach(Object.keys(pdf_elements), function(key, index) {
    if (product[index]) {
      doc.setFontSize(16 * 72 / 96);
      doc.text(pdf_elements[key].x * 72 / 96 + 40 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 40 * 72 / 96,
        product[index].product.Title);
      doc.setFontSize(10 * 72 / 96);
      doc.text(pdf_elements[key].x * 72 / 96 + 40 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 60 * 72 / 96,
        'SKU:' + product[index].product.SKU);
      doc.setFontSize(22 * 72 / 96);
      doc.text(pdf_elements[key].x * 72 / 96 + 100 * 72 / 96,
        pdf_elements[key].y * 72 / 96 + 300 * 72 / 96,
        product[index].product.Price);
    }
  });
});
...

```

Con esta implementación al hacer click en el botón creado en la historia de usuario anterior, se generará un documento PDF de catálogo a partir de su plantilla.

■ “H10 Creación de plantillas de catálogo”

Para la creación de plantillas de catálogo se añade un elemento en el formulario de creación de plantillas y una condición en la función de creación:

```

function pdf_warp_form_add_template_submit(&$form, $form_state) {
  ...
  'type' => $form_state['values']

```



```
['template_type'] == 0 ? 'product' : 'catalog',  
...  
}
```

■ H11 “Asignación de plantillas de catálogo”

Para la activación de plantillas de catálogo, se añade código a la función de activación de plantillas para que se diferencie entre una plantilla de catálogo y otra de producto. En concreto se realiza una consulta en la que se obtiene la información de plantilla para poder saber si estamos activando una plantilla de producto o catálogo. El código añadido es el siguiente:

```
function pdf_warp_template_enable_confirm_submit($form, &$form_state) {  
...  
$sql = "SELECT * FROM {pdf_warp_templates} WHERE id=".$id_template;  
$templateResult = db_query($sql);  
$templateType = 'product';  
  
foreach ($templateResult as $v) {  
    $templateType = $v->type;  
}  
...  
}
```

■ H12 “Mover elementos en plantillas de catálogo”

Esta historia de usuario, al igual que su homónima de productos, requiere realizar desarrollos tanto en el *back-end* como en el *front-end*. Se deben crear nuevas páginas de edición de plantillas de catálogo que también integren las librerías de AngularJS y Draggabilly, ya que la funcionalidad general será parecida. Las páginas que se añaden al *hook menu* son las siguientes:

```
$items['admin/appearance/pdf_warp/edit/catalog/%'] = array(  
  'title' => t('Editar plantilla de catálogo'),  
  'page callback' => 'pdf_warp_edit_catalog_template_page',  
  'page arguments' => array(5),  
  'access arguments' => array('administer site configuration'),  
  'type' => MENU_CALLBACK,  
);
```

```
$items['admin/appearance/pdf_warp/edit/catalog/%/save'] = array(
    'title' => t('Guardar plantilla'),
    'page callback' => 'pdf_warp_catalog_save_edit',
    'page arguments' => array(5),
    'access arguments' => array('administer site configuration'),
    'type' => MENU_CALLBACK,
);
```

Al igual que con los catálogos, en la función de *callback* de la página de edición, se tienen que cargar todas las librerías *front-end* necesarias y enviar el contenido de la plantilla desde base de datos. El código sería el siguiente:

```
function pdf_warp_edit_template_page($id_template) {
    // CARGA DE LIBRERÍAS

    $sql = "SELECT * FROM {pdf_warp_templates} WHERE id=".$id_template;
    $settings['pdf_warp']['template_content'] = db_query($sql)->fetchAll();
    drupal_add_js($settings, 'setting');

    return theme('pdf_warp_catalog_template');
}
```

La página de edición de plantillas de catálogo se divide en dos ficheros, uno que contiene la lógica de la página llamado pdf_warp_catalog.js y otro con la el código HTML representando la vista de nombre pdf-warp-catalog-template.tpl.php. La página resultante se puede ver en la Figura 3.19.

■ H13 “Borrado de plantillas de catálogo”

El borrado de plantillas de catálogo no requiere código extra, ya que el borrado inicial se realiza por ID de plantilla y no entra en conflicto con la implementación previa.

Pruebas

Las pruebas realizadas en este *sprint* son las siguientes:

- Creación de plantillas de catálogo.
- Borrado de plantillas de catálogo.



- Edición de plantillas de catálogo.
- Activado de plantillas de catálogo.
- Generación de PDF a partir de plantillas de catálogo.

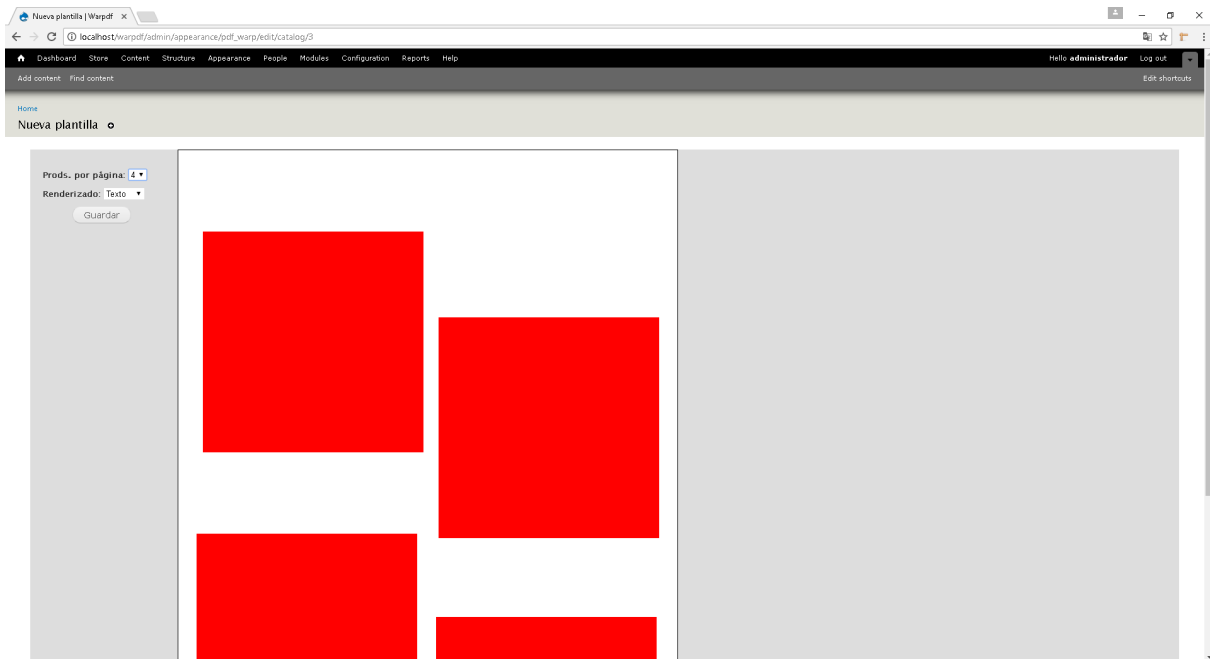


Figura 3.19: Diseño de edición de plantilla de catálogo

Capítulo 4

Conclusiones y trabajos futuros

En este capítulo se realiza un análisis del recorrido de este TFM y se recogen las conclusiones obtenidas y las posibilidades de ampliación y mejora del mismo.

4.1. Conclusiones

Durante la realización de este TFM se han integrado con éxito diversas tecnologías que trabajan en conjunto para lograr el objetivo principal del proyecto. En este caso concreto y, a modo de recordatorio, el objetivo era la generación de documentos PDF a partir de la información obtenida de los productos de una plataforma de comercio electrónico y las plantillas que haya creado un usuario administrador. Bajo el punto de vista de los objetivos, se puede decir que el proyecto ha resultado un éxito ya que se cumplen tanto el objetivo general, como los objetivos específicos y se ha conseguido realizar una extensión para Drupal completamente funcional y exportable, aun partiendo de un nivel de conocimientos bajo en PHP y Drupal.

Con respecto a la metodología de trabajo utilizada, no se han encontrado mayores dificultades a la hora de aplicar el proceso Scrum al ámbito de un TFM, en realidad no se requiere de la implementación de cambios mayores de la forma de trabajar en una colaboración tutor/alumno y se siente que es la forma de trabajar adecuada para un proyecto de estas características. El *product backlog* es una gran herramienta ya que, incluso con el limitado tiempo con el que se ha contado en este caso, permite organizar el trabajo de manera rápida y sencilla, eligiendo que tareas se desempeñarán para cada entrega.

Como se ha comentado, una dificultad encontrada ha sido el tiempo disponible. Habiendo dispuesto de más tiempo, se podría haber realizado una aplicación más sólida, con un diseño más atractivo y con más funcionalidades.



4.2. Trabajos futuros

Este proyecto podría ser continuado y extendido por otro alumno, tanto para un trabajo de una asignatura como para un proyecto propio, de modo que se proponen las siguientes mejoras:

- Elegir de forma sencilla que elementos aparecen en una plantilla, ocultando los que no se quieren mostrar.
- Generación de los elementos PDF genéricos. Actualmente funciona para los elementos básicos de un producto.
- Adición de atributos de Ubercart a los documentos PDF.
- Permitir modificar la plantilla de producto dentro de un catálogo.
- Redimensión de productos en catálogos.
- Menús de elementos de plantilla contextuales en vez de fijos.
- Redimensión dinámica arrastrando desde los bordes del elemento.
- Permitir la selección de tipo de fuente para el PDF.
- Permitir añadir la portada y contraportada desde otro documento PDF.
- Permitir añadir cabecera y pie a un PDF.
- Visor PDF integrado en Drupal.
- Diseño mejorado y atractivo.

Bibliografía

- [1] Marc Andersen. *TechVision: Innovators of the Net: Brendan Eich and JavaScript*. 2008. https://web.archive.org/web/20080208124612/http://wp.netscape.com/comprod/columns/techvision/innovators_be.html.
- [2] Bert Bros. *CSS software*. 1994-2017. <https://www.w3.org/Style/CSS/software.en.html#w26>.
- [3] World Wide Web Consortium. *First mention of HTML Tags on the www-talk mailing list*. 1991. <http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html>.
- [4] World Wide Web Consortium. *Cascading Style Sheets, level 1*. 1996. <https://www.w3.org/TR/1999/REC-CSS1-19990111>.
- [5] Git. *A Short History of Git*. 2014. <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>.
- [6] Misko Hever. *Hello World, angular is here*. 2009. <http://misko.hevery.com/2009/09/28/hello-world-angular-is-here/>.
- [7] Magnus Holm. *JSON: The JavaScript subset that isn't*. 2011. <http://timelessrepo.com/json-isnt-a-javascript-subset>.
- [8] Ken Schwabe Jeffrey Sutherland. *The Scrum Guide*. 2016. <http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf>.
- [9] Ken Schwaber Jeffrey Sutherland. *Business object design and implementation: OOPSLA '95 workshop proceedings*. 1995. p. 118. ISBN 3-540-76096-2.
- [10] Rasmus Lerdorf. *PHP, Behind the Mic*. 2003. <https://web.archive.org/web/20130728125152/http://itc.conversationsnetwork.org/shows/detail58.html>.



- [11] Rasmus Lerdorf. *PHP on Hormones*. 2007. 2009-06-22.http://web.archive.org/web/20130729204354id_/http://itc.conversationsnetwork.org/shows/detail3298.html.
- [12] Håkon Wium Lie. *Cascading HTML style sheets - a proposal*. 1994.
- [13] Benjamin Melançon. *The Definitive Guide to Drupal 7*. 2011. p. 823. ISBN 9781430231356.
- [14] Netcraft. *July 2016 Web Server Survey*. 2016. <https://web.archive.org/web/20160810054429/http://news.netcraft.com/archives/2016/07/19/july-2016-web-server-survey.html>.
- [15] Newswire. *Netscape and Sun announce JavaScript*. 1995. <https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html>.
- [16] PHP. *Preparation Tasks*. 2017. <https://wiki.php.net/todo/php72>.
- [17] Bob Remeika. *Five Questions With Michael Widenius ? Founder And Original Developer of MySQL*. 2009. <https://web.archive.org/web/20090313160628/http://www.opensourcereleasefeed.com/interview/show/five-questions-with-michael-widenius-founder-and-original-developer-of-mysql>.
- [18] John resig. *jQuery: past, present and future*. 2007. <https://www.slideshare.net/jeresig/history-of-jquery>.
- [19] Jeffrey Sutherland. *Agile Development: Lessons learned from the first Scrum*. 2004. <https://www.scrumalliance.org/resources/35>.
- [20] Ikujiro Takeuchi, Hirotaka; Nonaka. *New New Product Development Game*. 1986. <https://cb.hbsp.harvard.edu/cbmp/product/86116-PDF-ENG>.
- [21] Segue Technologies. *The Benefits of Adhering to Software Development Methodology Concepts*. 2015. <http://www.seguetech.com/benefits-adhering-software-development-methodology-concepts/>.
- [22] David Villafranca. *Amazon volvió en 2015 a los beneficios y superó los 100.000 millones en ventas*. 2016. <https://www.efe.com/efe/espana/economia/amazon-volvio-en-2015-a-los-beneficios-y-supero-100-000-millones-ventas/10003-2824141>.

- [23] W3Techs. *Usage of content management systems for websites*. 2017. https://w3techs.com/technologies/overview/content_management/all.
- [24] W3Techs. *Usage of JavaScript libraries for websites*. 2017. https://w3techs.com/technologies/overview/javascript_library/all.
- [25] Monty Widenius. *MariaDB versus MySQL ? Compatibility*. 2010. https://mariadb.com/about-us/wiki/MariaDB_versus_MySQL.



Anexo 1

4.3. Instalación manual de Drupal 7

El manual básico para la instalación no automática de Drupal es el siguiente:

1. Descargar el archivo comprimido de Drupal 7 de la web <https://www.drupal.org/project/drupal> como se puede ver en la Figura 4.1.
2. Extraer el contenido del archivo comprimido en la instalación de WAMP, concretamente en la carpeta <instalación_de_WAMP>/www/.
3. Renombrar directorio de la aplicación.(En el ejemplo warpdf).Figura4.2.
4. Crear la base de datos en el servidor de MySQL (Depende del gestor utilizado, en el ejemplo phpMy).Figura4.3.
5. Navegar a la aplicación de Drupal desde el navegador, en el ejemplo <http://localhost/warpdf>.
6. Seguir los pasos del instalador indicando que se utilice la base de datos del paso 3. Figura4.4.

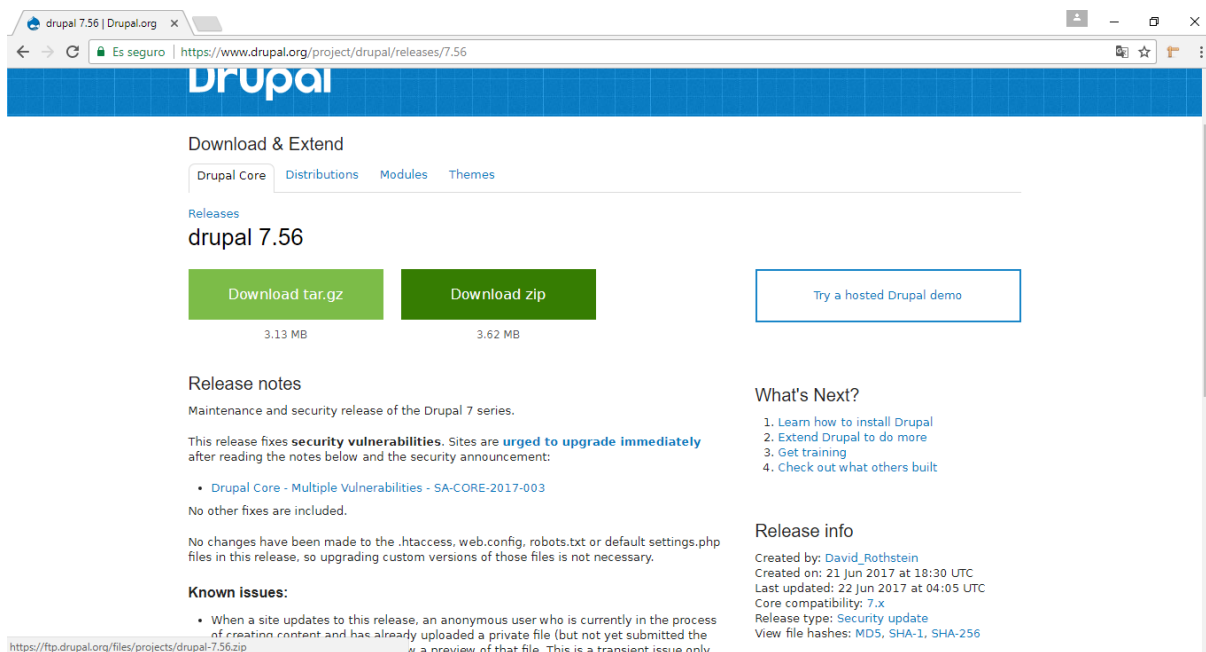


Figura 4.1: Web de descarga de Drupal

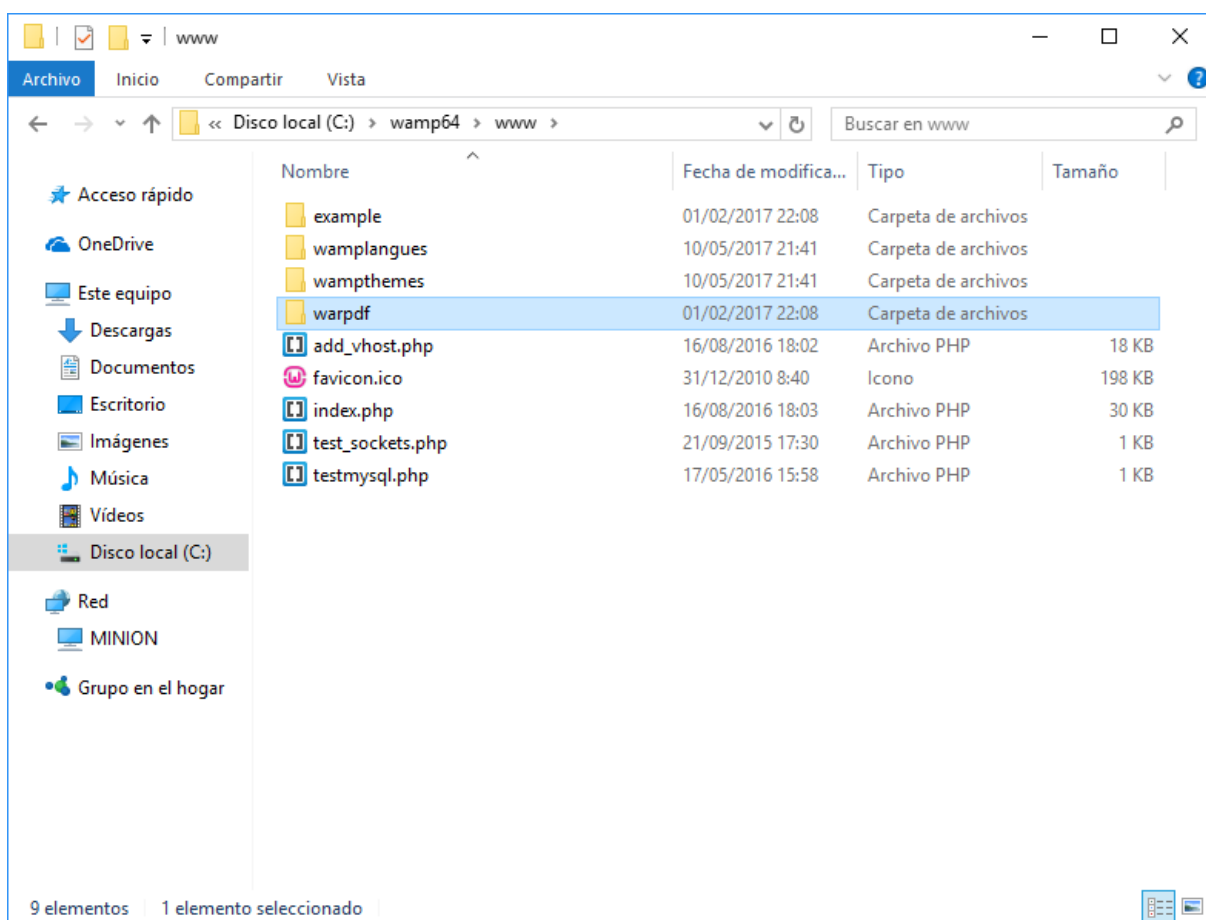


Figura 4.2: Directorio de la aplicación de Drupal

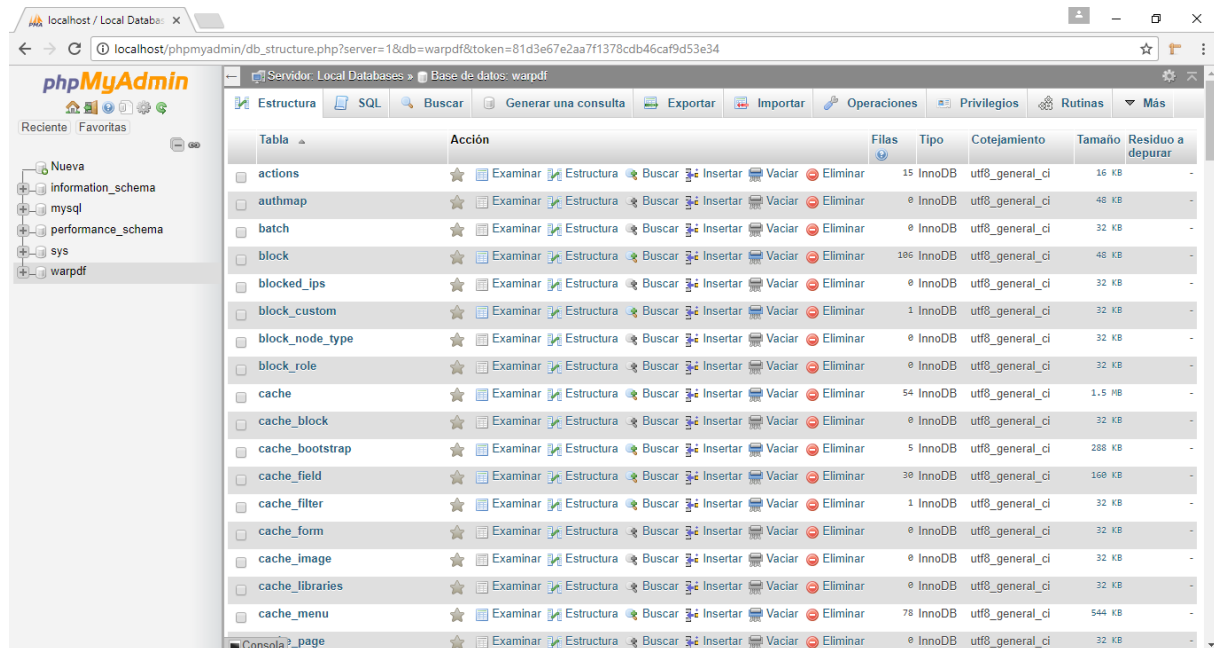


Figura 4.3: Creación de la base de datos

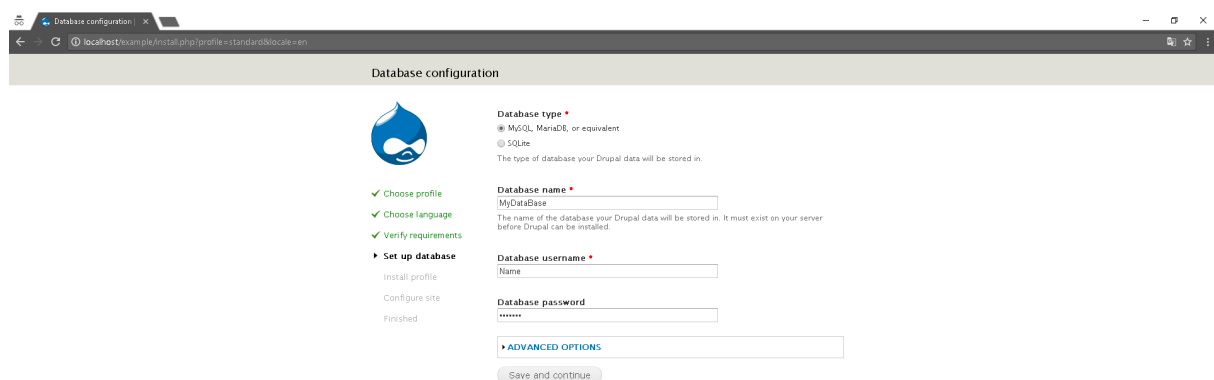


Figura 4.4: Wizard de Drupal

4.4. Manual de instalación del módulo PDF-Warp

Para instalar el módulo PDF-Warp primero se deben cumplir previamente los siguientes requisitos en la instalación de Drupal:

- Tener instalado y activado el módulo Ubercart con su módulo de productos y catálogos, además de sus dependencias.
- Tener instalado y activo el módulo Angularjs con la versión 1.6x, además de sus dependencias.
- Tener instalado y activo el módulo Views Datasource, en concreto su módulo `views_json`.

Cumpliendo estos requisitos, el manual de instalación es el siguiente:

1. Obtener el archivo comprimido con los componentes del módulo `pdf_warp`.
2. Navegar al menú “*Modules*” dentro de la instalación de Drupal.
3. Acceder a la opción “*Install new Module*”.(Figura 4.5)
4. Elegir la opción “*Upload a module or theme archive to install*” buscar el archivo del módulo `pdf_warp` y seleccionarlo.
5. Pulsar el botón “*Install*”.(Figura 4.6)
6. Activar el módulo desde la lista de módulos instalados seleccionando su *checkbox* y pulsando el botón “*Save Configuration*”.(Figura 4.7)

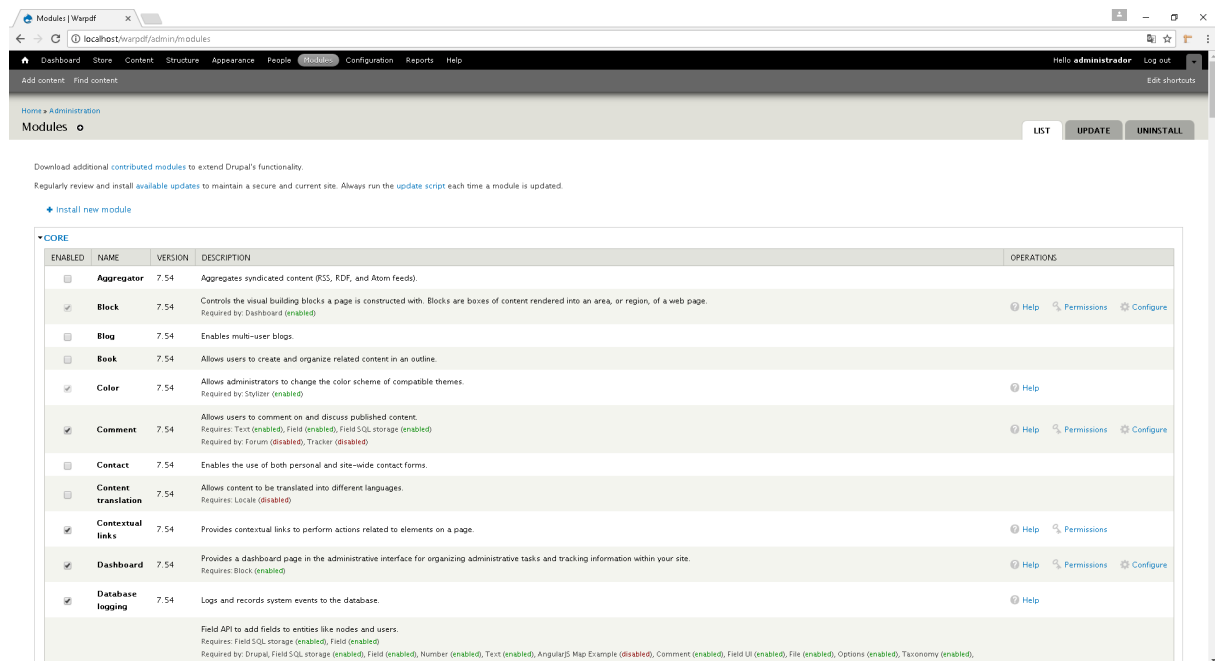


Figura 4.5: Lista de módulos y botón *Install new module*

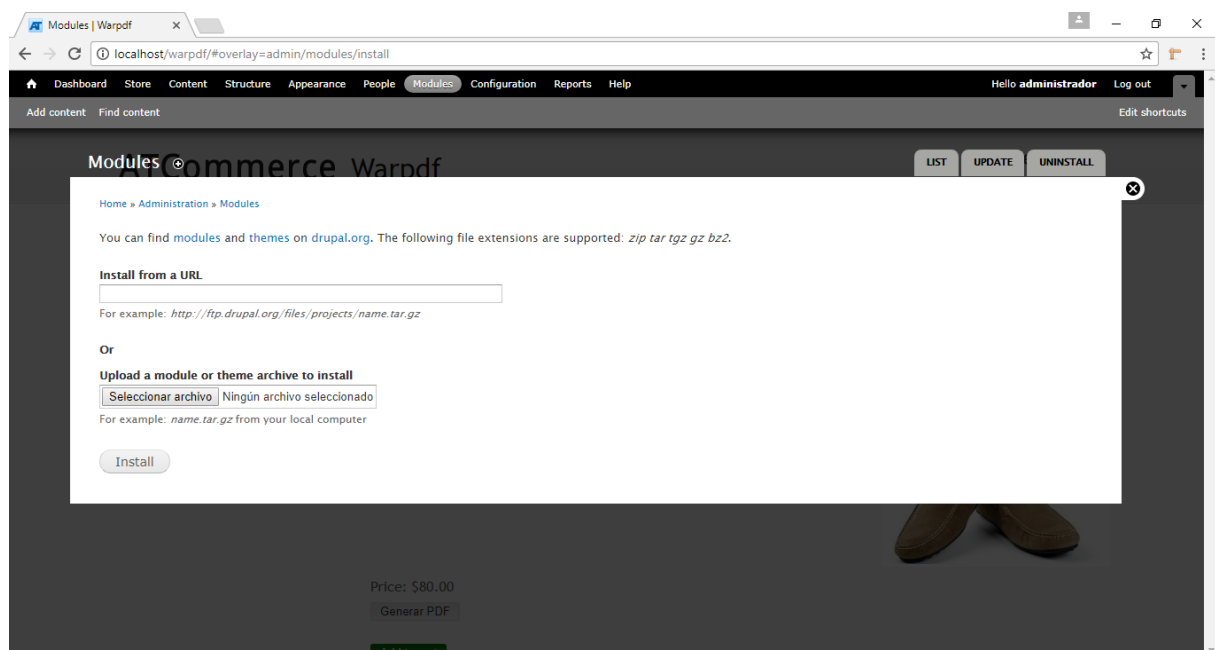


Figura 4.6: Selección de modo de instalación de módulo

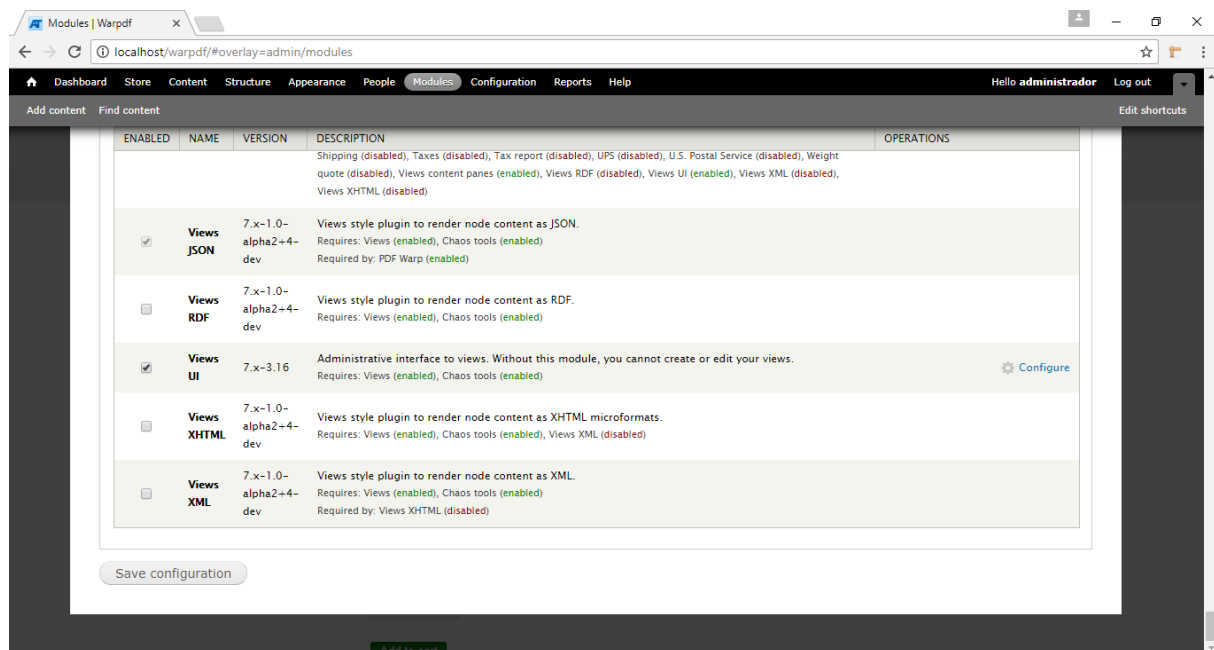


Figura 4.7: Botón *Save Configuration*