Trabajo Fin de Grado

24 de junio de 2015

Generación dinámica de parrillas de productos en Magento en base al análisis de datos de fuentes abiertas.

Autor:
Urko Lopez de Abetxuko Ruiz de Mendarozketa

Director:
José Miguel Blanco Arbe

Universidad del País Vasco
Euskal Herriko Unibertsitatea

No cabe ninguna duda de que no podría estar aquí, escribiendo estas últimas líneas de la memoria, sin la ayuda de muchas personas que han pasado al lado mío durante mis, casi, dos patitos de edad.

Sin duda alguna, tengo que empezar agradeciendo a mis "aitas", Miren y Patxi, que sin ellos no podría ser la persona que soy. Asimismo, a mi hermana, Oihane, que aunque no hemos podido tener la relación que me hubiese gustado, sé que siempre estás en los malos y buenos momentos.

A todos esos amigos y amigas del instituto, que de alguna manera, habéis marcado mi etapa escolar: Diego, Eneko, Aitzol, Amane, Dorleta, Amaia... Pero con especial hincapié en la persona más despreocupada, pero uno de los mejores amigos, que he conocido: Josu.

A los de la universidad, que de alguna hemos compartido clases y viajes: Ibai, Mikel, Sergio, Anartz, Adrian, Asier, Amaia, Erik, Markel, Dani x2, etc.

No me puedo olvidar de mis cuatros años en la residencia RESA de Donostia. He podido conocer a gente con motes como Piko, Piña, La Loca, Houston, Magis, Aja... y a gente con nombres de verdad: Endika, Rubén, Esti, Adrián Pedrosa, Lucia... y muchos más que he conocido a través de la resi (Iratxe). Un saludo a Ainhoa y Amaia, responsables de la residencia, que habéis estado ahí para todo lo que he necesitado, profesional y personalmente. Otro saludo a todas las empleadas y a Dani, único chico. Todos vosotros y vosotras habéis marcado mis años en Donostia. Sin vuestra presencia, los proyectos que he/hemos llevado a cabo no hubiesen podido ser lo que son ahora.

Está claro que no podemos saber que nos depara el futuro, y si seguiremos al lado de la persona que estamos amando. Y menos en mí. Muchas gracias por haber compartido estos, casi, tres últimos años a mi lado, Saioa. No han sido los años perfectos, pero ¿quién quiere algo perfecto? No tendría emoción... Has marcado un antes y un después en mi vida, no solo como novia, sino como amiga también. Gracias.

No me puedo olvidar ni de la universidad ni de algunas y algunos de sus profesores, pero para que la gente no piense que estoy haciendo "pelota", sólo nombrare a mi director del TFG, que sin él, este proyecto no lo hubiese podido terminar. También has marcado mi vida, sobre todo porque has sabido entenderme cuando lo he necesitado. Pocos profesores hay como tú. Gracias.

No me voy a poder olvidar fácilmente de mi primera experiencia en el mundo laboral. Además, he tenido la suerte de estar en On4U, que no es una empresa "corriente". He estado muy a gusto, muchas gracias a todas

las personas que forman la plantilla de On4U.

Algunos de vosotros os sorprenderéis al leer vuestros nombres aquí, pero tened en cuenta que cada momento de nuestras vidas es único, y que no se va a volver a repetir. En estas líneas faltan algunos nombres, que por falta de espacio no he podido añadir.

Ya me habían dicho que la vida no era fácil, pero nadie me había avisado que en algunos momentos fuera a ser tan dura. Aquí se acaba la mejor etapa de mi vida, pero por primera vez, con unas pocas ganas de seguir luchando por hacer que mi vida mejore.

Gracias una vez más.

Urko.

PD: De vez en cuando, hay que dejar de pensar en los demás, y empezar a pensar en ti mismo, aunque sea sólo unos segundos.

Resumen

En este Trabajo de Fin de Grado desarrollado en la empresa On4U, se ha implementado un módulo para Magento, cuya función principal es la generación dinámica de parrillas de productos en base al análisis del tiempo meteorológico, teniendo en cuenta la localización del cliente. Además, el módulo guarda automáticamente las compras efectuadas, junto con la información externa, para un posible análisis posterior que relacione los hábitos de compra con el tiempo meteorológico. Aunque se haya centrado en este caso de uso, se ha desarrollado con un enfoque modular, de tal manera que fuese fácil de integrar en el módulo el uso de otra fuente abierta de información. Para poder realizar el proyecto, se ha tenido que profundizar en varios conceptos relacionados con la plataforma de eCommerce Magento, entre ellos, el patrón Modelo-Vista-Controlador y el ciclo de vida de una petición.

Laburpena

On4U enpresan egindako Gradu Amaierako Lan honetan Magentorentzako modulo bat praktikan ipini da, zeinaren betebehar nagusia, eguraldiaren analisia oinarri bezala harturik, produktuen parrilla dinamikoak sortzea izan da, bezeroaren kokapena kontuan izanik. Honetaz gain, moduloak, erosketari buruzko informazioaz batera, kanpoko informazioa automatikoki gordetzen du, eguraldia erosketa-ohiturekin lotzen duen ondorengo analisi posible batean erabiltzeko. Erabilpen kasu honetan zentratu arren, modulua ikuspuntu modular batetik garatu da; horrela, moduluan beste informazio iturri bat erabiltzea erraza izan zedin. Egitazmo hau aurrera eramateko, eCommerce Magento-rekin lotutako hainbat kontzeptutan sakondu behar izan da; hauen artean Eredu-Bista-Kontrolatzaile patroian eta eskaera baten bizitza zikloan.

Abstract

A module for Magento has been implemented in this end-of-degree project, which has been carried out at the company On4U. The main function of this module is the dynamic generation of product grids based on the analysis of the weather forecast, taking the client's location into account. Furthermore the module automatically saves the purchases made, together with the outside information for a subsequent analysis which links shopping habits with the weather forecast. Although this is its main use, it has been developed with a modular approach so that it is easy to integrate the use of another open data source in the module. In order to carry out this project, it has been necessary to deepen into general concepts related to the eCommerce Magento platform, such as the pattern Model-View-Controller and the life cycle of a request.

Índice general

Índice de figuras

Índice de cuadros

Índice de comandos y extractos de código

Capítulo 1

Introducción

El proyecto que se presenta en esta memoria, se ha realizado durante el transcurso del segundo cuatrimestre del curso 2014-2015, del Grado de Ingeniería del Software. Sus orígenes se remontan al mes de enero de 2015. Tras una entrevista facilitada por el director académico con dos de los fundadores de la empresa On4U, se decidió hacer el Trabajo de Fin de Grado (TFG) en ella. El tema escogido fue la generación dinámica de parrillas de productos en base al análisis de datos de fuentes abiertas, usando la plataforma de eCommerce Magento.

Los objetivos del proyecto que se plantearon inicialmente se enmarcaron en la intersección de los campos del eCommerce y del aprovechamiento de fuentes abiertas de datos, con la intención de proyectarlo en el futuro a una integración de técnicas provenientes del área del *Big Data*. Tras más de cinco meses de trabajo, se ha logrado un sistema que, a modo de prueba de concepto, da soporte a la idea anteriormente mencionada. Concretamente, se ha implementado en Magento una particularización de esa idea: la generación dinámica de una parrilla de productos en base al tiempo meteorológico de la localización del visitante.

En el ámbito de la tecnología a desplegar en el proyecto, la apuesta estuvo influida por el software utilizado en On4U. Por un lado, como se ha mencionado anteriormente, la implementación del módulo se basó en la suite de eCommerce Magento, en su versión de libre uso. La suite está programada en el lenguaje de programación PHP. Por otro lado, el software para el control de versiones del código del módulo recayó en la herramienta Git, también de libre uso. Así mismo, se ha tenido muy en cuenta a la hora del desarrollo, la arquitectura avanzada que utilizaron los desarrolladores de la plataforma de eCommerce, entendiéndola y poniéndola en práctica en la implementación del módulo.

El proyecto se ha elaborado en las instalaciones que la empresa dispone en Vitoria-Gasteiz. Con tal fin, On4U puso a disposición un puesto de trabajo con un ordenador portátil. En cuanto al servicio online para el uso de Git en el proyecto, la opción escogida fue BitBucket, ya que permite el uso de repositorios privados gratuitos. Por otro lado, el resultado del desarrollo ha quedado implantado en un servidor virtual de la empresa francesa OVH, especializada en el mundo del hosting.

En esta memoria se presenta el trabajo confeccionado por Urko Lopez de Abetxuko Ruiz de Mendarozketa, autor del proyecto, y los resultados obtenidos. El proyecto se ha realizado bajo la dirección académica del doctor José Miguel Blanco Arbe y con la colaboración de la plantilla de On4U. A tal fin, en primer lugar se describen en profundidad los **antecedentes**. Seguidamente, se detallan los **objetivos** y el alcance del proyecto, incluyendo las exclusiones de éste, junto a unos casos de uso descritos en términos de historias de usuario. A continuación, se explica la arquitectura de Magento, haciendo hincapié en el patrón **Modelo-Vista-Controlador** utilizado en su desarrollo. En cuanto a la **prueba de concepto**, se describe la solución propuesta, su implementación y unas pruebas del funcionamiento. Posteriormente, se detalla el funcionamiento de la herramienta **Git**, junto con su uso en este proyecto. El siguiente capítulo, relata la **gestión** propia del proyecto en las áreas principales, es decir, el alcance, el tiempo y los costes. El proyecto concluye con unas **valoraciones personales** como extracto de la experiencia que ha generado su desarrollo dentro de la empresa On4U. Para finalizar, junto a las valoraciones personales se acompañan unas **propuestas de mejora**, que debido al alcance y tiempo limitado del proyecto no se han podido incluir en él.

Así mismo, se adjunta un glosario que incluye la definición de algunos términos y acrónimos que se pueden encontrar a lo largo del presente documento. En el caso de que el lector o lectora no entendiese a la perfección alguna palabra, ya sea por su desconocimiento en el ámbito específico que trata o por tener un significado diferente en el contexto del proyecto, es conveniente acudir a este capítulo de la memoria.

Como suele ser habitual en este tipo de documentos, también se incluye la bibliografía utilizada en el transcurso de su elaboración. Hay que mencionar, que dado el carácter del proyecto, gran parte de la bibliografía son páginas webs visitadas durante su realización, y que estaban online en la fecha indicada en cada una de ellas.

Por último, y para concluir la memoria, se incluyen varios apéndices, que a pesar de su condición, pueden ser de interés para el lector o lectora. Entre ellos:

- La instalación del entorno necesario en Ubuntu para que realice las funciones de un servidor web.
- La instalación de la plataforma de eCommerce utilizada en este proyecto: Magento.

Capítulo 2

Antecedentes

En la última década, la tecnología que rodea el mundo de Internet, ha ido avanzando a una velocidad vertiginosa, llegando a posibilitar infinidad de nuevos usos y funciones que no hace mucho tiempo eran totalmente inimaginables [8]. Esto ha llevado, por una parte, a que muchos sectores se hayan tenido que amoldar a la nueva era tecnológica; y por otra parte, a que los propios consumidores y consumidoras hayan cambiado sus hábitos. Uno de los ejemplos más importantes, ha sido el del mundo del comercio. La transformación sufrida por este sector, ha llevado a que hoy en día sea habitual hacer compras online. Con el tiempo, esas compras no se han limitado a productos no perecederos [15], sino que actualmente podemos encontrar supermercados, fruterías, farmacias, etc. online. Pero dicha transformación, no sólo se ha centrado en la aparición de nuevas tiendas online de productos o servicios no ofertados anteriormente, sino que también la propia tecnología de la web de la tienda ha ido avanzando.

Al acceder a algunos comercios de Internet, si se está identificado o simplemente se ha estado buscando anteriormente productos en ellos, en muchas ocasiones se muestran en portada artículos sugeridos en base a la huella que se ha ido dejando. En los últimos años, y en gran parte gracias a la infraestructura que ha ido desarrollando Google, mediante la publicidad y la llegada de otros servicios a muchas webs, la huella de un sitio web particular ha desaparecido dando paso a una única huella globalizada por persona. Se podría decir, que los productos que se muestran en la parrilla de algunas tiendas online, se escogen automáticamente dependiendo de la huella de cada persona.

Al mismo tiempo, han ido apareciendo en Internet innumerables fuentes abiertas de información, ligadas con el mundo del *Big Data* [9]. Todo esto, ha llevado a la idea de utilizar parte de esa información en el eCommerce, y de esta forma, una vez procesados esos datos, poder mostrar parrillas de

productos mucho más personalizadas. Es decir, no sólo tener en cuenta el "historial" personal de cada *internauta* a la hora de establecer el algoritmo de filtrado, sino también otros factores externos, como la calidad del aire, la existencia de enfermedades contagiosas, la proximidad de un festival dirigido a jóvenes, el tiempo meteorológico actual etc. Además, una vez llevada a producción esta idea y cada vez que se efectúe una compra, se podrá almacenar los datos relacionados con el "historial" y con las fuentes externas. De esta forma, se almacenaría mucha información, que dependiendo del volumen de ventas de la tienda donde se haya implementado, podría llegar a considerarse *Big Data*. Esos datos, podrían ser una nueva fuente de información mucho más personalizada.

Este proyecto, que consiste en la idea de generación dinámica de parrillas de productos en base al análisis de datos de fuentes abiertas versada en el párrafo anterior, ha sido realizado en la empresa On4U, tras un intercambio de emails y una entrevista. La decisión de escoger este proyecto, y no otro, se tomó por dos motivos:

- Ser un tema actual y estar relacionado con el mundo del *Big Data*, lo que ha conllevado la necesidad de explorar nuevos ámbitos.
- Ser un proyecto en empresa, lo cual ha aportado nuevas experiencias que de otra manera, dado el carácter académico de la universidad, ésta no hubiese podido contribuir de la misma manera. Además, como se explicará en este mismo capítulo, On4U es una empresa poco usual, destacando por ser innovadora en el mundo de la informática.

A continuación, se le ofrece al lector o lectora la posibilidad de conocer la empresa On4U y las dos tecnologías predominantes en el proyecto: Magento, como plataforma de eCommerce, y Git, como control de versiones. Por último, se da una breve explicación sobre las fuentes abiertas y su historia.

2.1. On4U

Figura 2.1: Logo de On4U

On4U es una empresa fundada en Vitoria-Gasteiz, relativamente nueva, que fue emprendida por tres personas:

1. Mikel Ruiz (Responsable de desarrollo)
2. Olmo Gonzalez (Responsable de seguridad y sistemas)
3. Angel Villa (Responsable de estrategia y marketing)

Actualmente, está compuesta por 11 personas (incluyendo los tres fundadores), de las cuales dos son mujeres. La ampliación de plantilla está siendo progresiva, lo cual es un indicador del buen funcionamiento tanto del sector, como de la empresa.

Como se puede comprobar en la página web (`on4u.es`) de la empresa, ella misma se define como una agencia especializada en eCommerce, aunque también realizan otro tipo de trabajos: desarrollos a medida, desarrollos móviles, formación avanzada, etc. Así mismo, han realizado trabajos para muchas empresas, tanto grandes como pequeñas, por ejemplo, Adolfo Domínguez y SM. El portfolio de la compañía se puede encontrar en la siguiente dirección web: `http://www.on4u.es/portfolio.html`.

El gran éxito que ha tenido On4U como empresa, como se puede comprobar en las siguientes líneas, ha sido gracias a la forma de trabajar y a la tecnología que emplea para su día a día.

2.1.1. Filosofía y tecnología de On4U

On4U se caracteriza por ser una empresa innovadora, ya que utiliza las últimas tecnologías que se pueden encontrar en el mercado. Además, le da importancia a que esas tecnologías sean soluciones con licencias abiertas; lo cual no sólo beneficia a la empresa reduciendo costes económicos, sino que también al cliente, ya que, por un lado, no tiene que pagar por los derechos de uso (licencia), y por otro lado, al no ser software cerrado, permite que muchas más empresas puedan hacer, mejorar y solucionar errores del proyecto (competencia).

Esta filosofía también se extiende a los entornos de desarrollo y producción de la plantilla de On4U, siendo la mayoría del software utilizado de libre uso. El sistema operativo predominante en los PCs y servidores es Ubuntu, basado en GNU/Linux. Para el desarrollo de tiendas online utilizan la plataforma de eCommerce Magento Community Edition (Open Source), la cual también se ha utilizado en este proyecto. Para poder ejecutarlo, utilizan Apache (Open Source), como servidor web y MySQL (licencia GNU GPL), como base de datos. En cuanto al desarrollo para dispositivos móviles, emplean Apache Cordova, un framework reciente de código abierto que utilizando JavaScript, HTML5 y CSS3, permite crear aplicaciones para diferentes sistemas operativos portátiles (Android, iOS, Windows Phone,

etc.). Cuando el proyecto web no está dirigido al eCommerce, o necesitan un framework adicional, se basan en Symfony (licencia MIT), diseñado para optimizar el desarrollo de las aplicaciones web utilizando el patrón Modelo-Vista-Controlador. Así mismo, para la documentación y creación de informes utilizan LibreOffice, paquete de ofimática libre y de código abierto. Por último, para el control de versiones y desarrollo de software en equipo, utilizan Git, licenciado bajo GNU GPL v2. Software, que también se ha utilizado en este proyecto.

2.2. Magento

Figura 2.2: Logo de Magento

Magento CE es un gestor de contenido web desarrollado con Software Libre destinado al eCommerce. Provee a las tiendas de mayor flexibilidad y control para sus plataformas [11].

Cuenta con una intuitiva interfaz de administración con potentes herramientas de marketing, SEO y gestión de catálogo para permitir crear tiendas online adaptadas a las necesidades de cada negocio.

Magento, fue creado en 2001 por una empresa americana llamada Varien con el objetivo de proporcionar una alternativa real a las principales plataformas de eCommerce propietarias, las cuales estaban dominando el mercado.

En 2011, Magento fue adquirida por eBay Inc. para formar parte de la unidad de negocio de X.Commerce. A día de hoy Magento cuenta con una plantilla con más de 375 personas, la plataforma está traducida a más de 60 idiomas, cuenta con una comunidad de más 80.000 miembros, tiene cientos de partners y hay más de 110.000 comercios funcionando con ella en todo el mundo. Además, cuenta con el mayor MarketPlace de eCommerce del mundo, Magento Connect.

2.2.1. Magento vs Prestashop

En la actualidad, dos de las plataformas más potentes y extendidas destinadas al eCommerce son Prestashop y Magento. Cada una tiene sus aportaciones positivas y negativas, pero cuando los responsables de On4U se tuvieron que decantar por una de ellas, no tuvieron dudas: Magento [12].

Técnicamente hablando, gracias a su arquitectura extensible, permite integrarse con otras herramientas (ERPs, dispositivos móviles etc.), lo cual, según el punto de vista de On4U, iba a ser un factor muy importante. Eso sí, hay que tener en cuenta que la curva de aprendizaje puede que sea mayor en comparación a la de Prestashop. Otros motivos de la elección [11], fueron los siguientes:

1. Es Software Libre.
2. Es la plataforma de eCommerce líder del mercado y además la que tiene un mayor crecimiento.
3. Es la única plataforma multistore.
4. Tiene la mayor comunidad de usuarios y usuarias.
5. Detrás de Magento está eBay (referente mundial en materia de eCommerce).
6. Cuenta con la mayor red de partners.
7. Tiene una API de Web services para integrar con cualquier software.
8. Dispone de un avanzado sistema de extensiones y de un market específico para distribuirlas de manera transparente.
9. Es una plataforma pensada para crecer.
10. Es un pilar esencial del X.Commerce (el futuro del eCommerce).

2.3. Git

Git, diseñado por Linus Torvalds, es un software destinado al control de versiones, pensado en la rapidez y eficiencia, tanto para proyectos pequeños como grandes.

Permite tener múltiples ramas locales (copias del código), siendo estas totalmente independientes entre sí. La creación o supresión de una rama, o la fusión de ésta con otra, se hace en cuestión de segundos, lo que permite un desarrollo fluido. Esto, por ejemplo, posibilita tener una rama de producción

y otra u otras de desarrollo, pudiendo fusionarlas en segundos si se quiere llevar el desarrollo a producción.

Así mismo, dispone de diferentes flujos de trabajo, adaptándose a las necesidades de cada situación. Por ejemplo, permite que todas y todos los desarrolladores sean iguales, o que una o uno de ellos tenga que aprobar las modificaciones del resto. Permite disponer de copias de seguridad y tener un control sobre que parte del código ha realizado o modificado cada persona.

Hoy en día, hay muchas empresas que utilizan Git como control de versiones, pero también hay muchos proyectos de código abierto alojados en plataformas que permiten el uso de Git, por ejemplo, el núcleo de Linux está alojado en GitHub (`https://github.com/torvalds/linux`).

2.4. Fuentes abiertas de información

Gracias al avance de las tecnologías, y en gran parte al avance que ha sufrido Internet en los últimos años, han ido apareciendo muchas páginas webs dónde se ofrecen datos de forma abierta, es decir, sin ningún tipo de restricción de derechos de autor, de patentes o de otro tipo de licencias. Además, estos sitios webs ofrecen mucha variedad de información, algunos de ellos ya procesada y otros sin procesar. En cuanto a la procedencia de esos datos, se pueden encontrar instituciones públicas o empresas privadas que ofrecen ciertos datos de forma libre. Uno de los ejemplos que se ha utilizado en este proyecto, es el del tiempo meteorológico, que lo ofrecen tanto instituciones públicas, como empresas privadas [10]. Pero hay muchos más ejemplos, como la calidad del aire que muestra la página web del Gobierno Vasco (`http://www.ingurumena.ejgv.euskadi.eus/calidad-aire-en-euskadi-2015/r49-3614/es/`).

Por último, hay que mencionar que estas fuentes de información se integran en cada vez más servicios o productos, dando un valor añadido. En los últimos meses, y con la aparición y uso del concepto *Big Data*, el uso de las fuentes abiertas está sufriendo un avance muy pronunciado.

Capítulo 3

Objetivos del proyecto

Como se ha mencionado en el capítulo anterior, el proyecto se ha centrado en la creación dinámica de parrillas de productos en base al análisis de datos procedentes de fuentes abiertas. Para ello se ha creado un módulo utilizando Magento como plataforma de eCommerce, y se ha llevado el control de versiones del código mediante el software Git.

A la hora de establecer la arquitectura del módulo, ésta se vio supeditada a la propia arquitectura de Magento. Como se indica en el siguiente capítulo, Magento se basa en el patrón Modelo-Vista-Controlador (MVC), lo cual ha llevado a la necesidad de entender la implementación particular del patrón que hace Magento, basándose en los conceptos que se habían estudiado en diferentes asignaturas del Grado.

Este capítulo explica los objetivos del proyecto. Para ello, se detalla su alcance, se describe la idea principal del módulo de Magento mediante historias de usuarios, y por último, se establecen las exclusiones del mismo.

3.1. Alcance del proyecto

El alcance del proyecto se divide en dos apartados.

Por un lado, como se ha comentado anteriormente, se ha tenido que profundizar en el patrón Modelo-Vista-Controlador (MVC), para poder llegar a entender la implementación, poco usual, que hacen los desarrolladores de Magento. Por ello, se ha tenido que enlazar los conceptos dados en las asignaturas de la especialidad de Ingeniería del Software del Grado con los conceptos propios de Magento, llegando a entender el motivo de la implementación especial que hicieron sus desarrolladores. Al mismo tiempo, esto ha llevado a la necesidad de entender otros conceptos de la plataforma de eCommerce, como la arquitectura de módulos, o la estructura de las URLs.

Por otro lado, se ha desarrollado un módulo de Magento como prueba de concepto. El desarrollo se ha hecho modular, de tal manera que las características del módulo que se presentan a continuación, pueden variar de forma sencilla:

- La "home" habitual de Magento se mantendrá y se mostrarán los productos más nuevos, aunque desde el panel de administración de la tienda, se podrá redireccionar automáticamente desde la "home" a la página con la parrilla dinámica.

- Si entramos a la ruta /ParrillaDinamica/, se mostrará la parrilla dinámica en base al tiempo meteorológico. Para ello, el módulo obtendrá la IP del visitante, la localizará y preguntará el tiempo en ese punto.

- En caso de que el cliente compre un producto desde la parrilla dinámica, el módulo guardará automáticamente ciertos datos: fecha de compra, producto que ha comprado, localización del cliente y el tiempo meteorológico.

- Cuando se añada o se edite un producto desde el panel de administración, se podrá agregar el artículo a uno de los grupos existentes: sol, lluvia, nieve, nuboso y helada.

- Asimismo, en la configuración del módulo se podrá poner en modo manual la obtención del tiempo, de la localización y de la dirección IP. Debido a esto, se podrá configurar de tal manera que genere automáticamente la parrilla de productos en base a la meteorología que se registra en unas coordenadas indicadas de forma manual, sin tener en cuenta la dirección IP y la localización del cliente.

Para obtener la localización de la dirección IP del cliente, y posteriormente, el tiempo que hace en ese punto, se ha delegado en dos servicios web destinados a esos propósitos:

- Para localizar la IP, se usará la API de la siguiente web: `http://ipinfo.io`.

- Para obtener el tiempo meteorológico en base a unas coordenadas, se usará la API de la siguiente web: `http://openweathermap.org/`

Por último, al ser un proyecto en empresa y haber sido desarrollado dentro de las instalaciones de On4U, se estableció un requerimiento:

3.1.1. Requerimiento

Como se ha expresado en los antecedentes, en On4U utilizan el software de control de versiones Git, para poder trabajar en equipo de una manera ordenada y tener un control del código exhaustivo. Desde la empresa, se estableció el requisito de utilizar Git como herramienta de control y de esta manera, familiarizarse con el uso que le dan en On4U.

3.2. Historias de usuario

A continuación se describen tres historias de usuario como modo concreto de representar el proyecto. Estas historias, aunque sean relatos ficticios, se basan en sugerencias y peticiones reales de algunos clientes de On4U. Las historias, están ordenadas de menos características solicitadas por parte del cliente, a más, siendo la tercera la más completa.

3.2.1. En base a la proximidad de un festival

Mendidenda, es una tienda de deporte online que hace envíos a Euskadi. Entre muchas otras cosas, vende tiendas de campaña "low-cost" y otros productos que van destinados a jóvenes que acuden a festivales.

La empresa que ha montado su tienda online les ha ofrecido la posibilidad de implementar la siguiente prestación: obtener mediante un servicio "OpenData" los días que se celebran festivales de ocio destinados a jóvenes, y 15 días antes de cada uno de los festejos, generar automáticamente la parrilla de productos de la "home" con artículos específicos para esos certámenes.

Mendidenda, ha valorado la idea positivamente, ya que podría incrementar las ventas de ciertos productos en determinas fechas.

3.2.2. En base al estado de la calidad del aire

Tu Farmacia Online, S.A. es una farmacia online que sólo opera en Granada capital y que vende productos que no requieren receta médica. Entre la variedad de artículos que ofrece la web, están los que pueden ser necesarios en momentos de mucha polución. Además, ofrece el servicio "express", que consiste en enviarlo a una dirección concreta de la ciudad en menos de 1 hora.

Una de las empleadas de la farmacia ha aportado una nueva idea: obtener la información de la calidad del aire que ofrece el ayuntamiento de Granada, y usarla para determinar que artículos van a aparecer en cada momento en portada. También, guardará el estado del aire junto con qué artículo se ha demandado en cada momento, para determinar que productos se venden

más dependiendo del grado de polución.

De esta manera, se podrá lograr en un momento de contaminación alta que el cliente vea en portada el producto que pueda necesitar comprar, y además se podrá ir mejorando la generación automática de parrillas paulatinamente.

3.2.3. En base al tiempo meteorológico

El bazar chino Ying-Yang-Yung, es una de las primeras tiendas de este sector que ha saltado a la venta online. Para ello, ha encargado a un equipo de programadores y programadoras que le hiciesen su tienda online, basándose en la plataforma de e-commerce Magento. Además, ha vislumbrado una nueva oportunidad de negocio que hasta ahora no se conocía: potenciar ciertos productos ante otros, dependiendo del tiempo meteorológico.

Para ello, les ha pedido al equipo informático, que programen un módulo de Magento donde obtenga el tiempo de la localización desde el lugar que se conecta el cliente, y a partir de ahí, muestre unos u otros productos.

Pero la solicitud del cliente no se acaba aquí, el dueño del bazar también quiere que se vayan almacenando los siguientes datos:

- Fecha de compra
- Producto que ha comprado
- Localización del cliente
- Tiempo meteorológico en su localización

En un futuro quiere utilizar estos datos para mejorar sus ventas, y ofrecer un servicio que esté relacionado con todo ello.

3.3. Exclusiones del proyecto

Del alcance del proyecto, a quedado excluido los puntos siguientes:

1. El procesamiento y obtención de los datos almacenados una vez que el cliente haya comprado algún producto desde la parrilla dinámica.
2. No se ha profundizado en el análisis del rendimiento, disponibilidad o seguridad de los diferentes servicios externos que se han utilizado en el proyecto.

3. Queda excluido el análisis de carácter legal que resultaría necesario para llevar a producción el módulo: el uso de la dirección IP en distintos servicios externos para obtener el tiempo meteorológico; y el almacenamiento de distintos datos de la compra y del cliente para su posterior procesamiento.

Capítulo 4

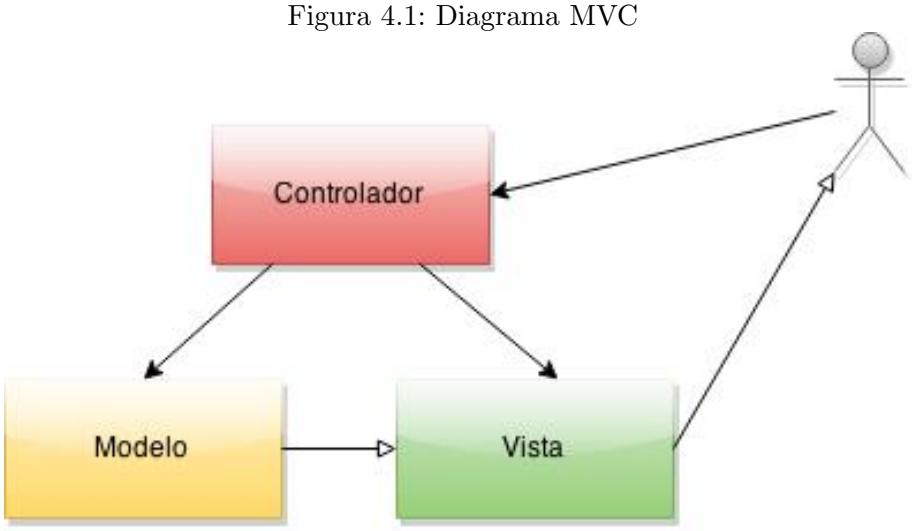
Magento y el patrón MVC

En aplicaciones de gran escala, como el caso de Magento, es necesario tener un patrón de diseño, el cual ayude a tener el código del software lo más ordenado y organizado posible. En el caso de la plataforma de eCommerce Magento, el patrón de diseño que decidieron utilizar sus desarrolladores fue el patrón MVC [18].

El patrón Modelo-Vista-Controlador (en inglés, Model-View-Controller, MVC) es un patrón de diseño de software donde la idea básica es la separación de conceptos y la reutilización del código. Para ello, separa los datos (modelo), la lógica del negocio (controlador) y la interfaz de usuario (vista).

A continuación, se muestra un diagrama del funcionamiento del patrón MVC y posteriormente se explica el ciclo de vida:

Figura 4.1: Diagrama MVC



1. El usuario o usuaria realiza una solicitud al sitio web.

2. El controlador recibe la acción requerida por el cliente.

3. El controlador, si es necesario, solicita al modelo la obtención o actualización de algún dato.

4. El controlador delega en la vista la tarea de construir la interfaz web que se le va a entregar al usuario, pasándole, si es necesario, los datos obtenidos en el paso anterior.

5. La vista construye la respuesta haciendo uso de los datos que ha recibido. En algunas ocasiones y dependiendo de la implementación, puede llegar a hacer peticiones al modelo en busca de la obtención de más datos.

6. Una vez creada la vista, ésta será devuelta al usuario o a la usuaria, aunque dependiendo de la implementación del patrón, ésta puede ser entregada vía el controlador.

En este capítulo se explica la arquitectura de algunos de los elementos más importantes de Magento que tienen que ver con el patrón MVC, a continuación se explica detalladamente el ciclo de vida de una petición en Magento con la ayuda de un diagrama y se cierra el capítulo dando una visión global de la arquitectura de Magento.

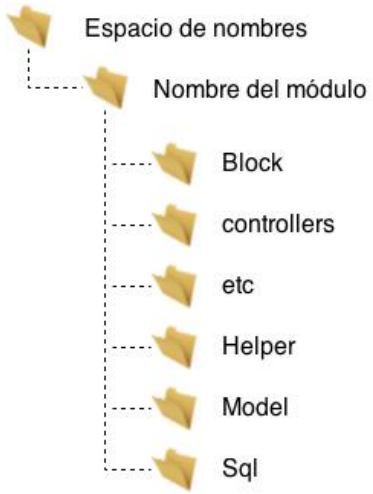
4.1. Arquitectura de los elementos de Magento

Para poder entender correctamente la arquitectura y la implementación del patrón que utiliza Magento, primero hay que comprender otros elementos de la plataforma de eCommerce. En esta sección, se podrá encontrar la estructura que sigue un módulo de Magento y el esqueleto de las URLs. Por último, se profundizará en el funcionamiento del patrón según la implementación que le ha dado Magento.

4.1.1. Estructura de un módulo

Al programar un nuevo módulo de Magento, la estructura típica de éste, es la siguiente:

Figura 4.2: Estructura genérica de un módulo [3]



Como se puede apreciar en la figura 4.2 (página 39), la primera carpeta es el espacio de nombres (namespace) y dentro de ella se encuentran los diferentes módulos separados en carpetas. El nombre de la carpeta corresponde con el nombre del módulo. Cada módulo, está subdivido en diferentes carpetas, basándose en el patrón MVC [3] [7]:

- **Block**: se podría decir que los bloques se corresponden con la vista del patrón MVC (figura 4.1, página 37). Los bloques unen los modelos con las plantillas del tema escogido, es decir, los bloques obtienen los datos de la base de datos y se los entregan a las plantillas.
- **controllers**: dentro de esta carpeta se encuentran los diferentes controladores del módulo, que cada uno de ellos se corresponde con el controlador del patrón MVC. Representan todas las acciones de la lógica del negocio implementada y delegan tareas a otras partes del sistema.
- **etc**: Magento utiliza archivos XML para configurar el comportamiento del módulo. Como mínimo tiene que tener el archivo config.xml, en el cual se declaran los modelos, helpers, bloques, rutas etc. Asimismo, puede contener el archivo system.xml y/o el archivo adminhtml.xml.
- **Helper**: en esta carpeta se encuentran los métodos que pueden ser útiles de diferentes maneras y que se utilizan en todo el sistema (modelos, controladores, plantillas y bloques).
- **Model**: como su nombre indica, es la parte del modelo del patrón MVC. Hay que tener en cuenta que la implementación que hace Magento del modelo no es la habitual:

La mayoría de los modelos de Magento pueden clasificarse de la siguiente manera:

1. ActiveRecord (modelo: un objeto, una tabla)
2. Entidad-atributo-modelo (Entity Attribute Value, EAV)

Cada modelo también obtiene un modelo de colección. Las colecciones son objetos PHP que se utilizan para mantener una serie de instancias individuales de modelos de Magento. El equipo de Magento implementó las interfaces *IteratorAggregate* y *Countable* de la librería estándar de PHP de tal manera que permite que cada tipo de modelo tenga su propio tipo de colección. Si no se está familiarizado con la biblioteca estándar de PHP, hay que pensar en los modelos de colecciones como arrays que también tienen métodos "adjuntos".

Los modelos de Magento no contienen ningún código para la conexión con la base de datos. En cambio, cada modelo utiliza dos clases "modelResource" (una lee, otra escribe), que se usan para comunicarse con el servidor de la base de datos (a través de los objetos-adaptadores de lectura y escritura). [19]

- **sql**: Maneja las tablas de la base de datos nuevas que serán utilizadas por el módulo y procesa todas las actualizaciones que se puedan ir implementando.

4.1.2. La estructura de las URLs de Magento

Se toma como ejemplo la siguiente URL, para explicar la estructura de ésta. [22]

http://www.Tienda.com/HolaMundo/mensajes/saludar/nombre/Pepito

1. **http://www.Tienda.com**: Dirección web (nombre del servidor) donde está alojado Magento.
2. **HolaMundo**: La primera parte de la URL se llama "Front name". Mediante HolaMundo, se le está indicando a Magento donde tiene que buscar el controlador. Para ello, tendrá que corresponderse con el nombre del módulo (no confundir con el nombre del paquete), que en este caso será HolaMundo.
3. **mensajes**: La segunda parte es el nombre del controlador ("Controller name"). Para ello, se tiene que corresponder con algún controlador que esté dentro de la carpeta "controllers" del módulo indicado en la primera parte de la URL. En el caso del ejemplo, el controlador

será mensajesController.php que estará dentro de la carpeta HolaMundo/controllers.

4. **saludar**: La tercera parte de la URL, y muchas veces la última, indica la acción del controlador ("Action name"). Para ello, tendrá que existir una función llamada "saludarAction" dentro del controlador que se ha indicado anteriormente.

5. **nombre/Pepito**: A partir de la acción del controlador, se consideran variables con el formato "key/value". En el ejemplo, la variable será nombre, y el valor Pepito.

Nota: Dependiendo de la configuración del servidor web Apache (o equivalente) puede ser necesario poner en la URL el nombre del archivo PHP. Esto no modifica la estructura de la URL, simplemente hay que añadirle "index.php", antes del "Front Name", quedando de la siguiente manera:

http://www.Tienda.com/index.php/HolaMundo/mensajes/saludar/nombre/Pepito

4.1.3. La implementación poco usual del patrón MVC

Como se ha mencionado anteriormente, la implementación del modelo-vista-controlador en Magento difiere un poco del patrón MVC clásico (figura 4.1, página 37) [18]. Hay que recordar dos cosas:

1. Los modelos de Magento no contienen ningún código para la conexión con la base de datos. Utilizan dos clases separadas.

2. El controlador no pasa datos a la vista, sino que los encargados de ello, como se explica a continuación, son los bloques.

Para dar una correcta explicación del funcionamiento del ciclo de una petición, usaremos la figura 4.3 (página 43). Los números de la siguiente lista, corresponden con los de la figura:

1. Cuando le llega una petición a Magento, éste mediante la configuración determina a que módulo le corresponde, y posteriormente a que controlador y que acción del controlador se ejecuta. Inicia la lógica del negocio, y delega las tareas.

2. Los métodos Action del controlador manipulan, si es necesario, los modelos correspondientes. También pueden utilizar los Helpers para tareas que no requieran la base de datos (como, por ejemplo, obtener información de algún servicio web). Principal diferencia si se compara con el modelo clásico de MVC.

3. Los métodos Action del controlador inicializan los layout y empiezan con la renderización.

4. Si es necesario, las plantillas de la vista obtienen datos mediante los bloques, ya que no reciben ningún tipo de dato desde el controlador. Los bloques son los encargados de leer los datos de los modelos.

5. Todos los bloques se renderizan en la página HTML y se entrega ésta.

Al igual que en el patrón clásico de MVC, los modelos (aunque no directamente, como se ha explicado antes) son los únicos que obtienen o modifican la información de la base de datos. Los helpers, son simplemente funciones que facilitan ciertas tareas.

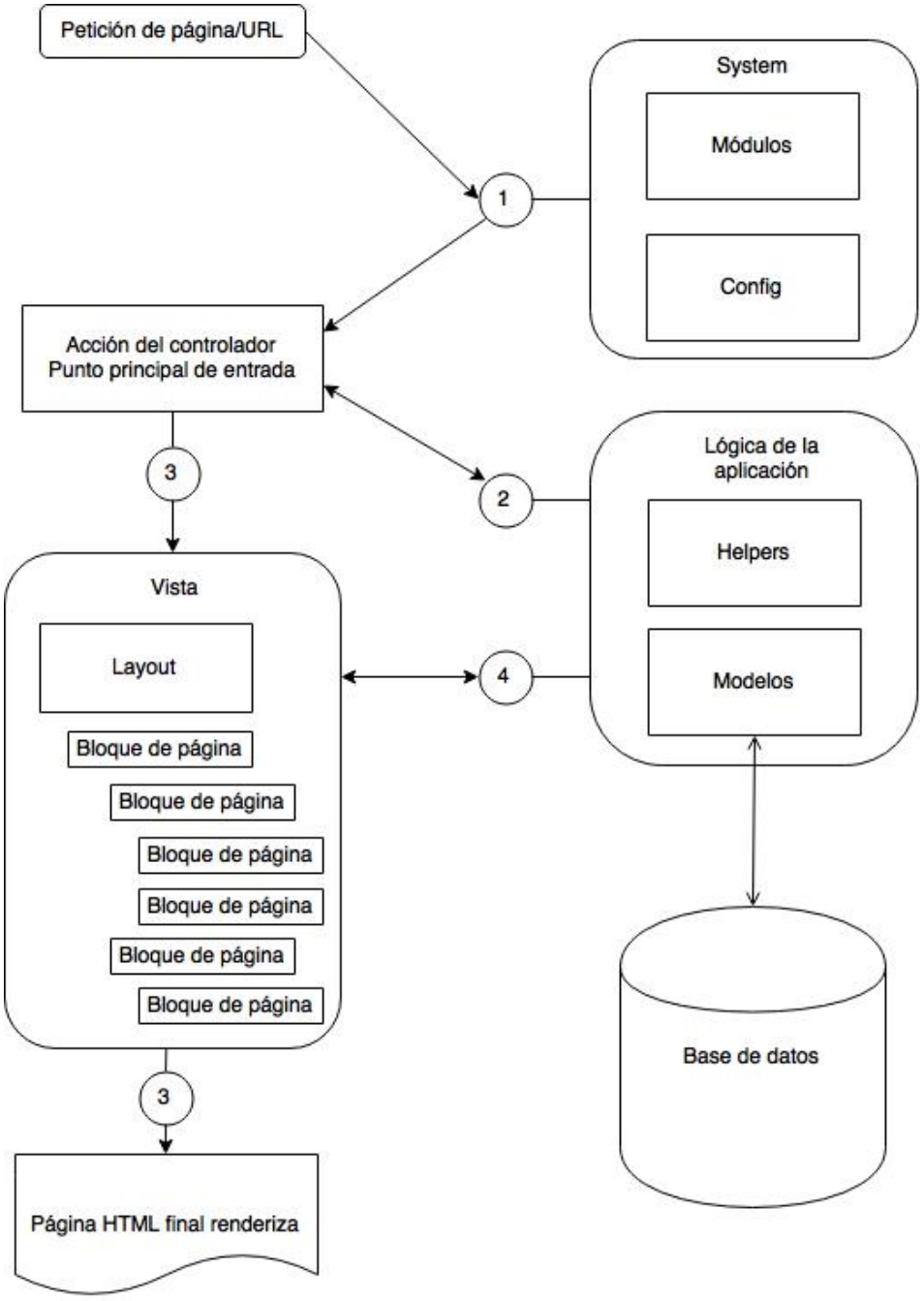


Figura 4.3: Diagrama MVC de Magento [6]

4.2. El ciclo de vida de las peticiones

En esta sección se profundiza en el ciclo de vida de una petición[18]. Para ello, se coge como referencia la figura 4.4, página 46).

Al igual que se ha hecho en la sección anterior, los números de la siguiente lista se corresponden con los números de la figura.

1. Cuando se hace una petición a la tienda, lo primero que hace el servidor es ejecutar el archivo index.php. Éste, instancia la aplicación de Magento.

2. La aplicación, genera los objetos "Response" y "Request", y los guarda en la clase estática "Mage". Los objetos, se referencian a lo largo del ciclo de vida de la petición.

3. La aplicación instancia el controlador de la fachada ("Front Controller").

4. Durante la instanciación, el controlador de la fachada ("Front Controller") comprueba la configuración global de global/web/routers en busca de cualquier ruta, y las inicializa, guardándolas como una propiedad interna.

5. Durante la ejecución del controlador de la fachada ("Front Controller"), éste busca una ruta que coincida con la de la petición entre todas las rutas. Cuando se encuentra una coincidencia, se instancia una acción del controlador y se llama a su método. Para encontrar el controlador y la acción del controlador, Magento se basa en la estructura de la URL, como se explica en la sección 4.1.2 de la página 40.

6. El método de la acción del controlador escogido en el punto anterior, manipula, si es necesario, los modelos.

7. Cada acción del controlador es responsable de cargar el diseño (layout) y de renderizarlo.

8. Cada petición tendrá un número de "handles". El diseño (layout) combinado global buscará estos "handles", y sus innerXML se combinan para crear un fichero XML de diseño para la petición particular.

9. Según el patrón MVC, la petición se encuentra en la Vista. En este momento se renderiza la página HTML que se muestra como consecuencia de la petición. Para ello hay que tener en cuenta lo siguiente:

- En las plantillas (template) phtml, el objeto $this referencia al objeto del bloque. Las plantillas, llaman a los métodos de $this para obtener datos (que no modificarlos).
- Como se ha explicado anteriormente, los bloques son los encargados de obtener los datos; ya que el controlador no pasa datos ni a las plantillas ni a los bloques.

Nota: Las plantillas (templates) de diseño de Magento utilizan la extension phtml [17].

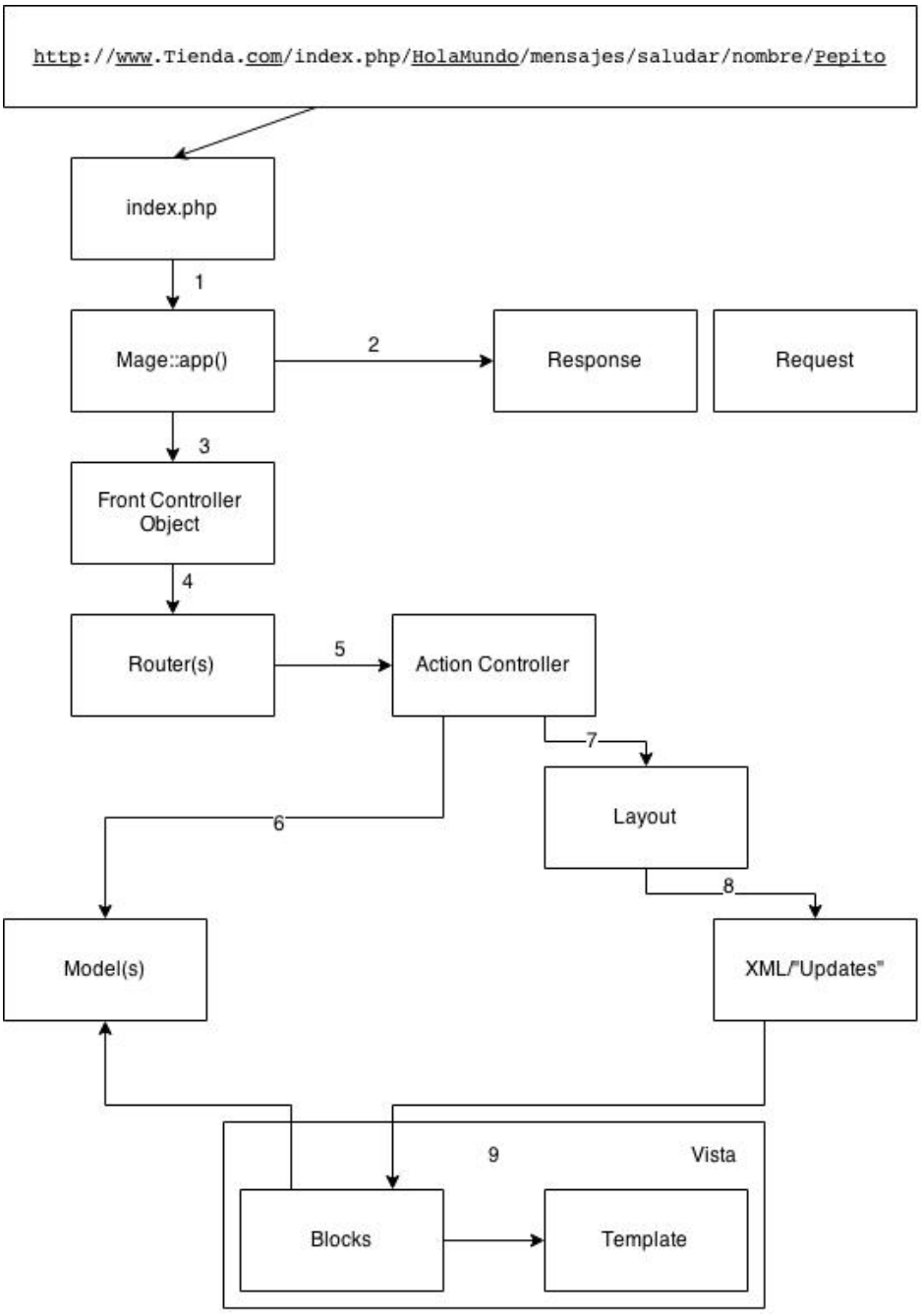


Figura 4.4: Diagrama MVC extenso de Magento [20]

4.3. Conclusiones: El patrón MVC y su visión global en Magento

Como se ha podido comprobar en las secciones anteriores de este capítulo, Magento se ha basado en el patrón MVC para poder organizar de forma adecuada su arquitectura. Los desarrolladores no se limitaron a organizar de esta manera el núcleo de la plataforma, sino que también llevaron esa forma de organizar los elementos a todos los aspectos, empezando por el ciclo de vida de una petición, siguiendo por la estructura que siguen las URLs de las peticiones, para terminar con la configuración de carpetas que siguen sus módulos. En definitiva, las estructuras de los diferentes elementos de Magento, tienen una relación estrecha con el patrón de diseño MVC.

Por último, es necesario señalar, que los conceptos vistos en las diferentes asignaturas de la especialidad Ingeniería del Software tienen una estrecha relación con las diferentes arquitecturas de los elementos de Magento, aunque los desarrolladores de éste lo lleven a un nivel mucho más avanzado.

Capítulo 5

Desarrollo en implementación de la prueba de concepto

Se ha implementado como prueba de concepto, una de las historias de usuario de la sección 3.2.3 de la página 34, exactamente la tercera historia, que consiste en la idea de generar dinámicamente las parrillas de productos en base al tiempo meteorológico.

El caso de uso, consiste en obtener el tiempo meteorológico del lugar donde se conecta el cliente, y dependiendo del resultado, mostrar unos u otros artículos en la parrilla de productos. Además, cuando el cliente añada al carro un artículo, la información más relevante se guarda en la base de datos, para su posterior análisis y procesado.

En las próximas páginas, se describe la solución propuesta a la historia de usuario indicada anteriormente, los conceptos más importantes de la implementación de éste en Magento y para terminar, se detallan unas pruebas de funcionamiento.

5.1. Solución propuesta

Se ha creado un módulo de Magento, donde se ha implementado el caso de uso mencionado anteriormente. El módulo, que se le llamó "ParrillaDinamica", sólo tiene un controlador con dos acciones. En la tabla 5.1 de la página 50, se pueden encontrar las rutas para ejecutar dichas acciones.

Las tres primeras rutas de la tabla, a efectos prácticos, son las mismas, ya que ejecutan el mismo controlador y la misma acción. Esto sucede cuando la URL de la petición no está completa y falta algún último campo de los

que se han explicado en la sección 4.1.2 de la página 40. Por ejemplo, cuando falta la parte de la acción de la URL, Magento interpreta que es la acción "index"; o cuando falta el controlador y la acción, Magento interpreta que ambos son "index" [16].

Ruta	**Controlador**	**Acción**
/ParrillaDinamica/	indexController	indexAction
/ParrillaDinamica/index/		indexAction
/ParrillaDinamica/index/index		indexAction
/ParrillaDinamica/index/anadir/id/:id		anadirAction

Cuadro 5.1: Rutas del módulo

A continuación, se pormenoriza la finalidad y el funcionamiento de las dos acciones del controlador "index".

5.1.1. Acción por defecto (indexAction)

Cuando se ejecuta indexAction, se muestra una parrilla de productos generada dinámicamente. Esta parrilla se genera en base al tiempo meteorológico de la localización del cliente.

El funcionamiento se explica con la ayuda de la imagen 5.1 de la página 51. Como en otros casos, los números de la siguiente lista, se corresponden con los de la figura.

1. Una persona hace una solicitud a la tienda. La URL de la petición será la siguiente: tienda.com/ParrillaDinamica/index/index. Esta solicitud llega al servidor, que mediante Apache y Magento, ejecuta el módulo.

2. Magento le pregunta a Apache cual es la IP del cliente y éste se la devuelve.

3. Magento, mediante el servicio externo de localización IP, obtiene las coordenadas de dicha IP.

4. Una vez que se han obtenido las coordenadas, se hace una petición al servicio externo del tiempo meteorológico para que le devuelva la situación meteorológica actual en las coordenadas mencionadas.

5. Magento devuelve al cliente una página web con una parrilla de productos generada dinámicamente, en base al tiempo meteorológico. El botón de añadir al carro de la parrilla, no enlazará a la acción por defecto de añadir un producto, sino a la acción "anadirAction" del módulo.

Hay que mencionar, que todos estos pasos ocurren cuando la configuración del módulo está en modo automático. Cuando, por ejemplo, se establecen a mano las coordenadas, el paso dos y tres se saltan, ya que las coordenadas a usar han sido introducidas manualmente. También podría darse el caso que se haya configurado de tal manera de que el tiempo siempre sea sol. En este último caso, sólo se ejecutarán los pasos uno y cinco.

5.1.2. Acción de añadir un producto (anadirAction)

Una vez cargada la parrilla dinámica, si el usuario acciona el botón de "añadir al carro", como se ha indicado anteriormente, la acción del botón no es la habitual. Al pulsar el botón, la ruta cargada es /ParrillaDinamica/index/anadir/id/:id, siendo :id el ID del producto que se quiere añadir.

Según se está cargando la ruta, se guarda en la base de datos la siguiente información (en una tabla de la base de datos dedicada a ello):

- Fecha y hora de la acción
- ID del producto añadido
- La localización (coordenadas) que se han obtenido.
- El tiempo meteorológico de la localización.

Al efectuar la escritura de la información en la base de datos, los siguientes pasos serán los habituales: se añadirá al carro el artículo, y se redireccionará a la página donde se muestran los productos añadidos al carro.

Hay que señalar que, todo este proceso pasa desapercibido para el cliente.

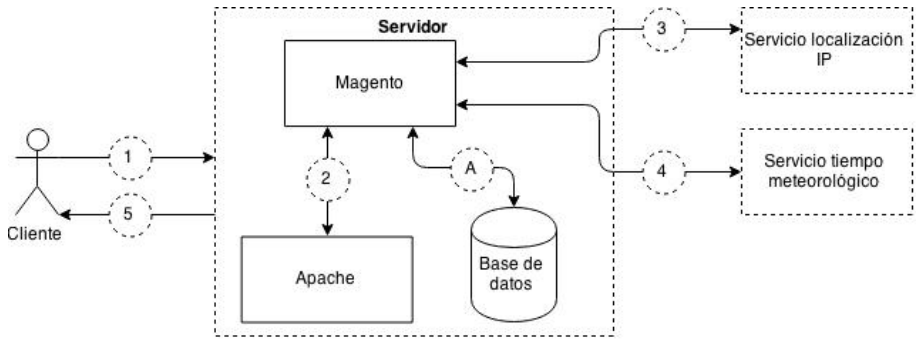


Figura 5.1: Diagrama de la solución propuesta del caso de uso (indexAction)

5.2. Implementación de la solución

En las siguientes páginas se explican algunos de los pasos mas relevantes de la implementación, entre ellos los que utilizan un servicio externo para obtener algún dato:

5.2.1. Obtener la dirección IP del visitante

La dirección IP que se tiene que obtener, es la IP real del visitante. Para ello, hay que revisar que el o la visitante no esté detrás de algún proxy o algo del estilo.

En el proyecto, se ha implementado de la siguiente manera:

```php
<?php
   public function obtenerIP(){
      if (!empty($_SERVER['HTTP_CLIENT_IP'])){
         //Verificar la ip compartida de internet
         $ip = $_SERVER['HTTP_CLIENT_IP'];
      } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])){
         //Verificar si la ip fue provista por un proxy
         $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
      } else {
         $ip = $_SERVER['REMOTE_ADDR'];
      }
      return $ip;
   }
```

Extracto de código/comando: 5.1: Obtener la IP del visitante

5.2.2. Obtener la geolocalización de una dirección IP

Se parte de que se necesita geolocalizar una dirección IP. Para ello, se ha utilizado un servicio externo que facilita esta tarea, ya que las direcciones IP no guardan una relación directa con una región geográfica, sino que se van asignando aleatoriamente según las necesidades de cada proveedor. Además, hay que tener en cuenta que la localización geográfica de una dirección IP no tiene porque ser exacta, es más, muchas veces no será del todo correcta.

`ipinfo.io` ofrece una API JSON que ofrece este servicio. La versión gratuita permite 1000 peticiones diarias, pero existe la opción de comprar más peticiones (`http://ipinfo.io/pricing`).

La URL de la solicitud a la API, es algo del estilo: `http://ipinfo.io/8.8.8.8/json`. En este caso, el servicio devuelve una respuesta como la siguiente:

JSON

```
{
   "ip": "8.8.8.8",
   "hostname": "google-public-dns-a.google.com",
   "city": "Mountain View",
   "region": "California",
   "country": "US",
   "loc": "37.3860,-122.0838",
   "org": "AS15169 Google Inc.",
   "postal": "94040"
}
```

Se puede observar, que el campo "loc" devuelve las coordenadas de la dirección IP que se había pasado en la petición. En PHP, se ha implementado una función que pasándole como parámetro la dirección IP, devuelve un array con las coordenadas, tal que así:

```
<?php
   public function obtenerCoordenadasIP($IP){
      $ipinfo = json_decode(
         file_get_contents("http://ipinfo.io/{$IP}/json"));
      $coordenadas = explode(",", $ipinfo->loc);
      return $coordenadas;
   }
```

Extracto de código/comando: 5.2: Obtener las coordenadas de una IP

5.2.3. Obtener el tiempo meteorológico de unas coordenadas

En este proyecto, se necesita obtener el tiempo meteorológico partiendo de unas coordenadas concretas, ya sea geo-localizando la dirección IP o estableciendo manualmente las coordenadas. Cuando se habla del tiempo meteorológico, se refiere a obtener un conjunto de datos que sirvan para clasificar el punto de coordenadas en ese instante entre unos grupos preestablecidos.

Para ello, se ha utilizado la API de OpenWeatherMap, que es un servicio online que provee una API sobre el tiempo meteorológico gratuita y de pago. Este servicio, ofrece información sobre la meteorología del momento,

así como la de los próximos días, datos históricos etc. En este caso, sólo se ha usado el servicio del tiempo actual. Se puede obtener, en el siguiente enlace, toda la información: `http://openweathermap.org/current`

Una de las ventajas de este servicio, es que pasándole unas coordenadas o el nombre (o ID) de la ciudad, se puede obtener información de distintas maneras: JSON, HTML y XML . De esta forma, por ejemplo, partiendo de que las coordenadas de Barcelona son las que se indican a continuación:

- Latitud: 41.381685
- Longitud: 2.1728317

Llamando a la siguiente dirección `http://api.openweathermap.org/data/2.5/weather?lat=41.3816854&lon=2.1728317` se obtiene un JSON, como el que se muestra a continuación:

JSON

```
{
   "coord":{"lon":2.17,"lat":41.38},
   "sys":
      {
         "message":0.0106,"country":"ES","sunrise":1430714657,
         "sunset":1430765513
      },
   "weather":
      [{
         "id":801,"main":"Clouds",
         "description":"few clouds","icon":"02d"
      }],
   "base":"stations",
   "main":
      {
         "temp":291.739,"temp_min":291.739,"temp_max":291.739,
         "pressure":1013.55,"sea_level":1023.88,
         "grnd_level":1013.55,"humidity":82
      },
   "wind":{"speed":4.9,"deg":62.5033},
   "clouds":{"all":24},"dt":1430727456,
   "id":6690837,"name":"Forum","cod":200
}
```

La definición de que es cada campo y para qué sirve, se puede encontrar en la siguiente URL: `http://openweathermap.org/weather-conditions`.

Para este proyecto en concreto sólo se ha necesitado el "id" del apartado "weather" Mediante este número y utilizando la información facilitada en la documentación (`http://openweathermap.org/weather-conditions`), se ha organizado, en grupos, de la siguiente manera:

- Sol: 800, 801, 904, 951-954
- Nuboso: 7xx, 802, 803, 804
- Lluvia: 2xx, 3xx, 500-504, 520-522, 531, 960, 961
- Helada: 903
- Nieve: 511, 6xx, 906

Cuando se ha llevado este proceso a PHP/Magento, se puede observar que la obtención del JSON se ha realizado de una forma muy simple:

```php
<?php
   public function obtenerTiempo($latitud, $longitud){
        $tiempo = json_decode(
           file_get_contents(
              "http://api.openweathermap.org/data/2.5/weather
              ?lat={$latitud}&lon={$longitud}&lang=es"
           )
        );
      $idCondicionTiempo = $tiempo->weather[0]->id;

      /*Tratar la variable idCondicionTiempo para devolver
      Sol,   Nuboso, etc. dependiendo del numero
      que se haya obtenido.*/
   }
```

Extracto de código/comando: 5.3: Obtener el tiempo de unas coordenadas

5.2.4. Crear una tabla en la BD y escribir en ella

Como se ha comentado anteriormente, cuando el cliente añade/compra un producto desde la parrilla dinámica, se guarda cierta información para su posterior procesamiento. Dicha información es la siguiente:

- ID del producto
- Hora de la compra
- Latitud y longitud (coordenadas)

- El tiempo meteorológico en ese momento

Para ello, se ha creado una tabla nueva en la base de datos [1] [21]. A la tabla se le ha llama "Parrilladinamica_bigdata", ya que guarda cierta relación con el concepto en sí de *Big Data*. La finalidad de esta sección es explicar los pasos más relevantes, no ser la de un manual exhaustivo.

En el primer paso se ha configurado el archivo config.xml del módulo, añadiendo las siguientes líneas, que le indican a Magento la existencia de una nueva tabla y sus características:

```
<config>
   <global>
      <models>
         <Parrilladinamica>
            <class>Tfg_Parrilladinamica_Model</class>
            <resourceModel>
               Parrilladinamica_resource
            </resourceModel>
         </Parrilladinamica>
         <Parrilladinamica_resource>
            <class>Tfg_Parrilladinamica_Model_Resource</class>
            <entities>
               <bigdata>
                  <table>Parrilladinamica_bigdata</table>
               </bigdata>
            </entities>
         </Parrilladinamica_resource>
      </models>
   </config>
</global>
```

Extracto de código/comando: 5.4: Configurar una nueva tabla en la base de datos (config.xml)

A continuación, se ha creado un archivo dentro de la carpeta de recursos de la carpeta sql del módulo, con la siguiente información:

```php
<?php
$installer = $this;
$installer->startSetup();

$table = $installer->getConnection()
    ->newTable(
        $installer->getTable('Parrilladinamica/bigdata'))
    ->addColumn('id', Varien_Db_Ddl_Table::TYPE_INTEGER,
        null, array(
            'identity' => true,
            'unsigned' => true,
            'nullable' => false,
            'primary' => true,
        ), 'id')
    ->addColumn('product_id', Varien_Db_Ddl_Table::TYPE_INTEGER,
        null, array(
            'unsigned' => true,
            'nullable' => false,
        ), 'Product id')
    ->addColumn('addedtocart_at',
        Varien_Db_Ddl_Table::TYPE_TIMESTAMP, null, array(
            'nullable' => false,
        ), 'AddedToCart at')
    ->addColumn('latitud', Varien_Db_Ddl_Table::TYPE_TEXT,
        64, array(
            'nullable' => false,
        ), 'Latitud')
    ->addColumn('longitud', Varien_Db_Ddl_Table::TYPE_TEXT,
        64, array(
            'nullable' => false,
        ), 'Longitud')
    ->addColumn('tiempo', Varien_Db_Ddl_Table::TYPE_TEXT,
        64, array(
            'nullable' => false,
        ), 'Tiempo')
    ->setComment('Parrilladinamica bigdata');

$installer->getConnection()->createTable($table);

$installer->endSetup;
```

Extracto de código/comando: 5.5: Crear una tabla en la base de datos (Magento)

En este archivo es donde se ha especificado las columnas de la tabla, indicándole las características de cada una. Posteriormente, se han creado los modelos:

Estos modelos son muy simples:

- Model/Bigdata.php

```php
<?php

class Tfg_Parrilladinamica_Model_Bigdata
    extends Mage_Core_Model_Abstract {

    protected function _construct(){
        $this->_init('Parrilladinamica/bigdata');
    }
}
```

Extracto de código/comando: 5.6: Modelo de la tabla Bigdata

- Model/Resource/Bigdata.php

```php
<?php

class Tfg_Parrilladinamica_Model_Resource_Bigdata
    extends Mage_Core_Model_Resource_Db_Abstract {

    protected function _construct(){
        $this->_init('Parrilladinamica/bigdata', 'id');
    }
}
```

Extracto de código/comando: 5.7: Modelo del recurso de la tabla Bigdata

- Model/Resource/Bigdata/Collection.php

```php
<?php

class Tfg_Parrilladinamica_Model_Resource_Bigdata_Collection
   extends Mage_Core_Model_Resource_Db_Collection_Abstract {

   protected function _construct(){
      $this->_init('Parrilladinamica/bigdata');
   }
}
```

Extracto de código/comando: 5.8: Modelo colección de BigData

Gracias a la arquitectura de Magento, la creación de una tabla nueva en la base de datos y en sus modelos correspondientes, es muy sencilla. En consecuencia, la escritura en la tabla también lo es:

```php
<?php

public function anadirFilaATabla(...){
   $bigdata = Mage::getModel('Parrilladinamica/bigdata');

   $bigdata->setProduct_id($_product->getId());
   $bigdata->setAddedtocart_at(time());
   $bigdata->setLatitud($coordenadas[0]);
   $bigdata->setLongitud($coordenadas[1]);
   $bigdata->setTiempo($tiempo);

   $bigdata->save();

}
```

Extracto de código/comando: 5.9: Añadir una fila a una tabla

5.3. Pruebas de funcionamiento

En esta sección se muestran tres pruebas, que comprueban el correcto funcionamiento de la implementación de la prueba de concepto. Para ello, se han realizado tres ensayos en una tienda de Magento que se ha montado en un servidor de pruebas para este fin. Los ensayos han sido los siguientes:

- Entrar a la "home" de la tienda y comprobar que se muestran productos varios, sin ningún tipo de clasificación.

- Con la configuración del módulo en automático, comprobar que entrando en la ruta /ParrillaDinamica/ se muestran productos relacionados con el tiempo meteorológico actual.
- Con la configuración del tiempo del módulo en sol (manual), comprobar que entrando en la ruta /ParrillaDinamica/ se muestran productos relacionados con el sol. Además se añadirá un producto al carro para comprobar que se registra la acción en la base de datos.

5.3.1. Primera prueba

Para realizar esta prueba, se ha entrado desde un navegador a la tienda del servidor. Nada más entrar ("home"), el resultado ha sido el siguiente:

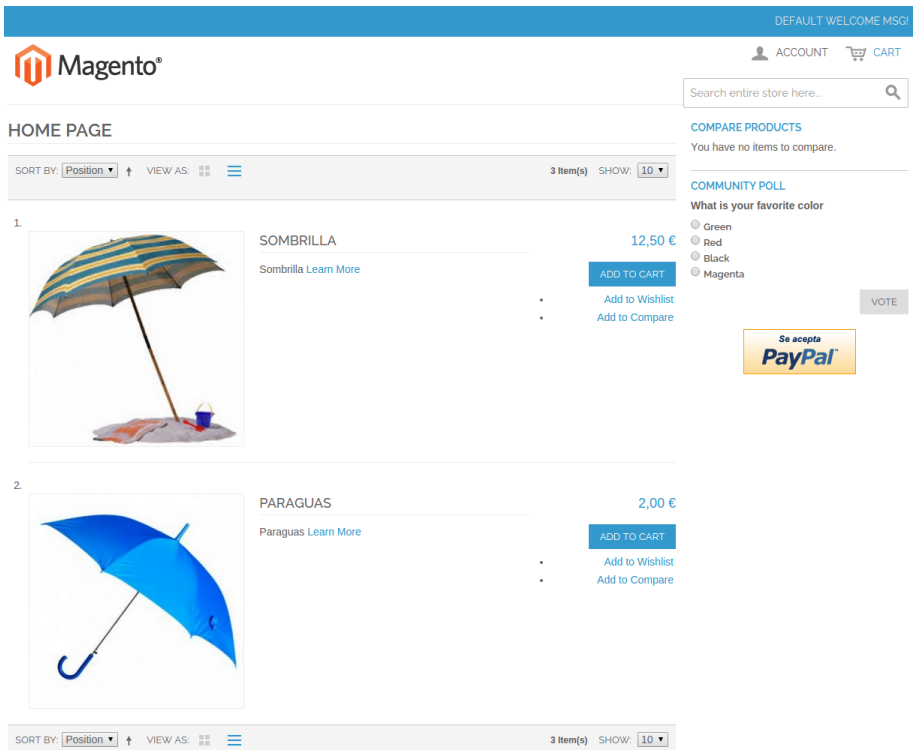


Figura 5.2: Primera prueba. Resultado.

Como se puede comprobar en la imagen, hay dos productos, uno de ellos relacionado con el sol y el otro con la lluvia. Queda comprobado que los productos no se muestran bajo una clasificación del tiempo meteorológico.

5.3.2. Segunda prueba

La segunda prueba ha consistido en entrar mediante el navegador a la ruta del módulo y comprobar que la IP detectada es la correcta. A continua-

ción, que la localización de la IP es adecuada, y por último, que el tiempo que ha indicado se corresponde con el de ese momento, en esa localización. La configuración del módulo debía estar en automática, como se aprecia en la siguiente imagen:

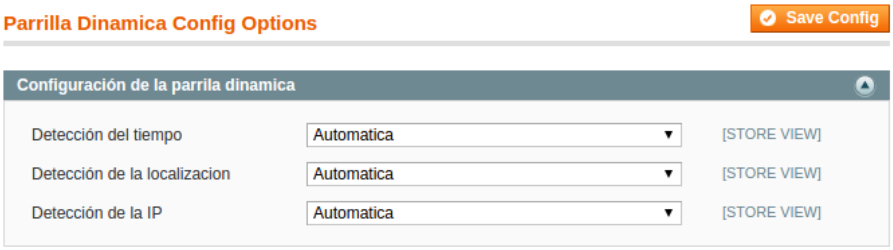


Figura 5.3: Segunda prueba. Configuración módulo.

El resultado de haber entrado en la dirección del módulo (/ParrillaDinamica/) ha sido el siguiente:

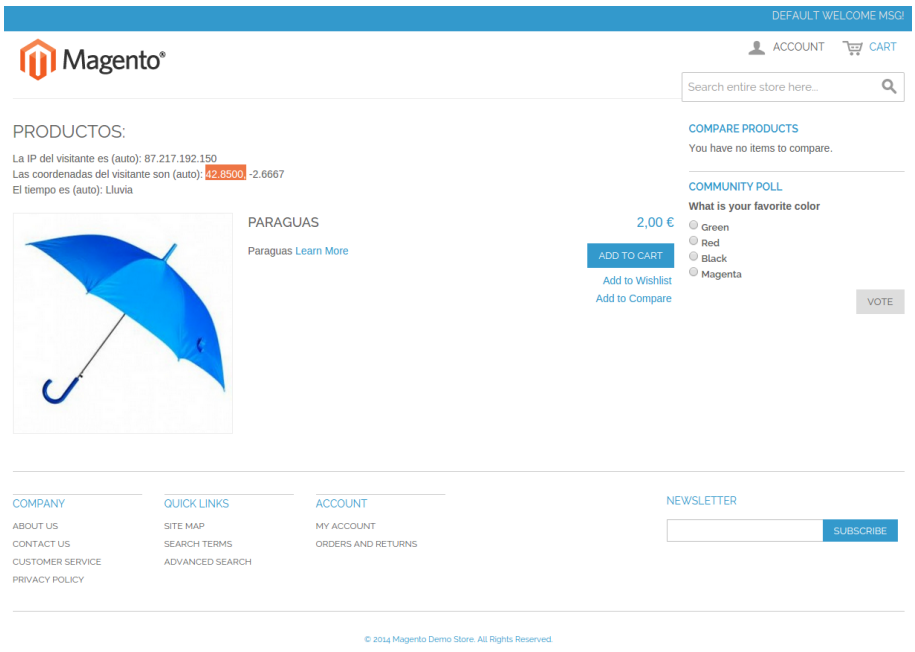


Figura 5.4: Segunda prueba. Resultado.

Tal y como se puede apreciar en la figura, la IP detectada automáticamente fue la “87.217.192.150”, que se corresponde con la IP pública de donde se estaban realizando las pruebas. Las coordenadas, detectadas fueron las “42.8500, -2.6667”, que están localizadas en Vitoria-Gasteiz, ciudad desde donde se realizaron las pruebas. El tiempo meteorológico detectado, era “llu-

via", que también se correspondía. Por ende, la prueba de funcionamiento ha sido correcta.

5.3.3. Tercera prueba

La tercera, y última prueba, ha consistido en entrar en la configuración del módulo y establecer el tiempo manualmente a "sol", como se puede apreciar en la siguiente imagen:

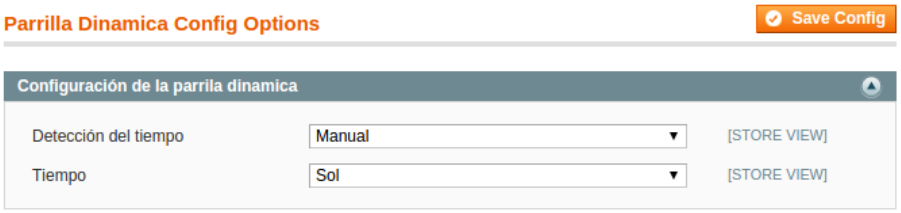


Figura 5.5: Tercera prueba. Configuración módulo.

A continuación, comprobar entrando al módulo desde el navegador que solo se muestran productos relacionados con el "sol":

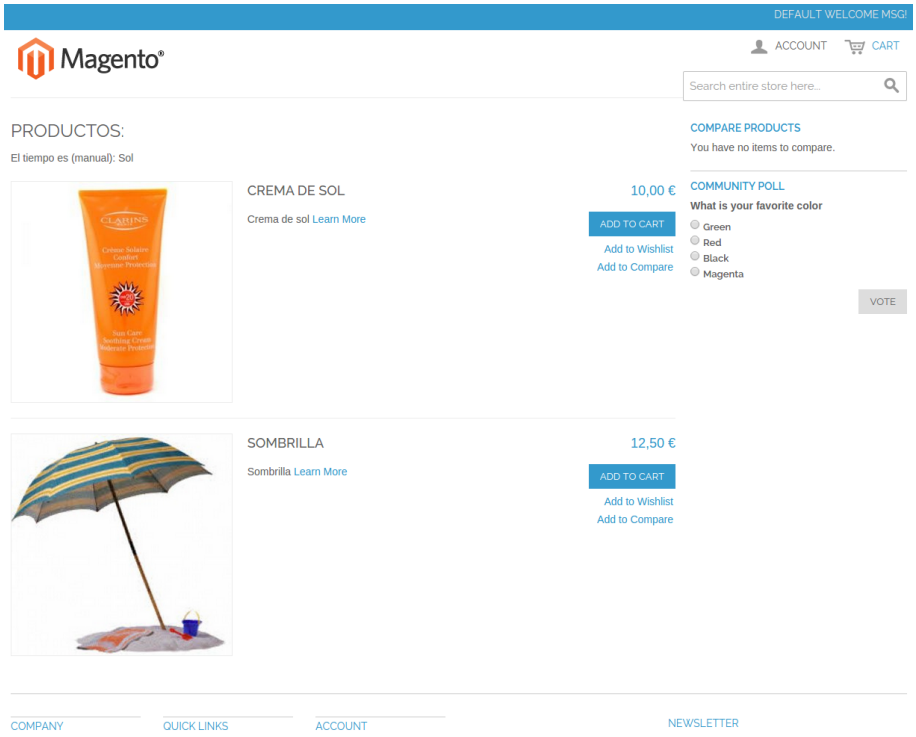


Figura 5.6: Tercera prueba. Resultado.

Por último, se ha añadido la sombrilla a la cesta de la compra y se ha comprobado que se ha escrito una fila en la tabla de la base de datos destinada a este fin.

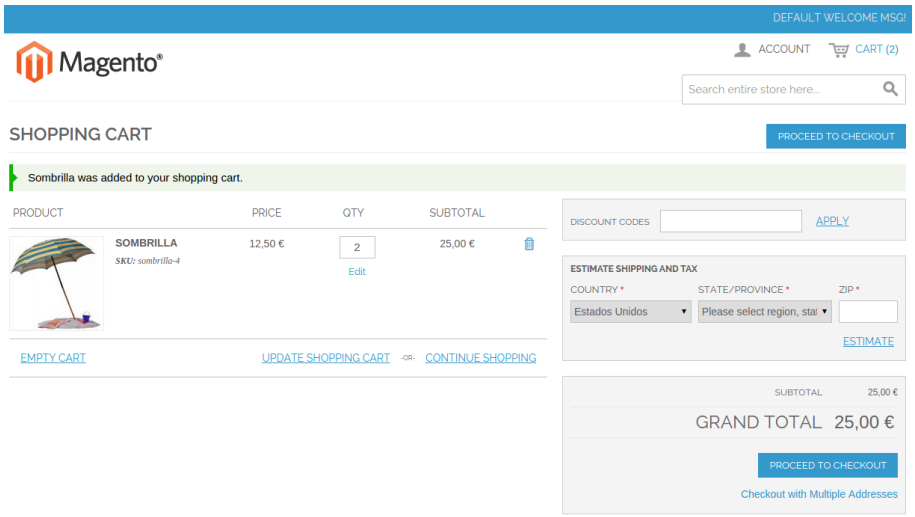


Figura 5.7: Tercera prueba. Añadir al carro la sombrilla.

Figura 5.8: Tercera prueba. Fila afectada BD.

Capítulo 6

Control de versiones: Git

Hoy en día, casi todos los proyectos de software son realizados por varias personas a la vez, y en muchas ocasiones, esas personas no se encuentran en el mismo sitio trabajando. Esto, no sólo ocurre en el ámbito empresarial, también sucede, por ejemplo, en el software libre desarrollado por *internautas*. Además, cuando el proyecto toma una envergadura significativa, es preciso tener un control del código, sobre todo cuando existen diferentes versiones de éste (desarrollo, producción, pruebas, etc.). A lo largo de la última década, se han ido desarrollando diferentes productos de software, llamados "control de versiones", para solventar los problemas anteriores. Entre todos ellos, destaca Git, gracias a su facilidad de uso y su gran popularidad, tanto en el sector empresarial, como en el sector del desarrollo libre.

Como se ha mencionado en los objetivos del proyecto, en On4U también se utiliza Git como control de versiones, lo que condujo a poner como requisito el uso de esta herramienta, y de esta forma, familiarizarse con este software.

En este capítulo, se explican las características básicas y su funcionamiento. Por último, se detalla el uso que se le ha dado en la implementación de la prueba de concepto.

6.1. Características básicas de Git

A continuación, se explica la instalación de esta herramienta en Ubuntu y los comandos básicos para empezar a utilizar Git [2]. Aunque la instalación sólo se describa para el sistema operativo Ubuntu, existen versiones compatibles con Windows [5] y Mac OS [23]. Eso sí, los comandos serán los mismos en todos los sistemas operativos.

6.1.1. Instalación de Git en Ubuntu

Para instalar Git en Ubuntu, sólo hay que ejecutar el siguiente comando desde la línea de comandos, ya que el repositorio que contiene Git, ya está incluido por defecto en este sistema operativo. Al completar la instalación, la herramienta ya estará lista para su funcionamiento.

```
1 sudo apt-get install Git
```

Extracto de código/comando: 6.10: Comando instalación Git

6.1.2. Primeros pasos con Git

Para poder explicar mejor el funcionamiento de Git, se supondrá que existe una carpeta llamada "MagentoProject", la cual contiene un proyecto de Magento. El código fuente de este proyecto, es al que se le quiere aplicar un control de versiones.

El primer paso, es inicializar un repositorio local dentro de la carpeta "MagentoProject". Para ello, hay que abrir una nueva línea de comandos y cambiar la ruta de ésta, a la de la carpeta del proyecto. Después, se ejecuta el siguiente comando para inicializar el repositorio:

```
1 git init
```

Extracto de código/comando: 6.11: Comando inicialización Git

De esta forma, se habrá creado un repositorio local dentro del proyecto.

En las siguientes páginas se explican los comandos más básicos de Git, suponiendo que sólo se está trabajando con una rama (copias del código fuente para diferentes usos: desarrollo, producción, etc.). Los comandos se tienen que ejecutar en la carpeta del repositorio que se ha inicializado. En este caso, dentro de la carpeta "MagentoProject".

6.1.2.1. Añadir un archivo al repositorio

Por defecto, un repositorio de Git, aunque esté dentro de la carpeta del proyecto, no lleva seguimiento de ningún archivo. Por esto, hay que indicarle a Git cuáles son los archivos del proyecto de los que tiene que llevar control. Para ello, existe el siguiente comando que añade un archivo específico o una carpeta al repositorio:

```
1 git add <ruta del archivo o carpeta>
```

Extracto de código/comando: 6.12: Comando add Git

6.1.2.2. Confirmar los cambios realizados

Cada vez que se realiza una modificación en algún archivo, Git no guarda esas modificaciones hasta que se le indique lo contrario. Por ello, hay que añadir de nuevo el o los archivos modificados al repositorio mediante el comando indicado en la sub-sección 6.1.2.1, de la página 66. Por último, hay que ejecutar el comando "commit" con el fin de guardar una versión con los cambios realizados.

```
1 git commit
```

Extracto de código/comando: 6.13: Comando commit Git

Los dos pasos indicados anteriormente, se pueden simplificar con un solo comando. Hay que tener en cuenta, que de este modo se guardarán todos los archivos que se han añadido anteriormente al repositorio y que en algún momento se hayan modificado.

```
1 git commit -a
```

Extracto de código/comando: 6.14: Comando commit Git

En los dos casos, al ejecutar el comando "commit", pedirá que se escriba un mensaje indicando los cambios en la versión.

6.1.2.3. Revertir los cambios guardados

En el caso de que se quiera revertir los cambios realizados en el último "commit" o anteriores, simplemente hay que ejecutar el siguiente comando:

```
1 git revert <commit>
```

Extracto de código/comando: 6.15: Comando revert Git

Siendo <commit> el número único de la versión a la que se quiere volver. Para saber ése número, hay que ejecutar el siguiente comando, el cual mostrará la siguiente información de cada "commit":

- Número único del "commit"
- Autor
- Fecha
- Mensaje

```
1 git log
```

Extracto de código/comando: 6.16: Comando log Git

6.1.2.4. Estado del repositorio

El comando que se especifica a continuación, muestra el estado actual del repositorio, indicando los archivos que no llevan seguimiento y los que no se han confirmado.

```
1 git status
```

Extracto de código/comando: 6.17: Comando status Git

6.1.2.5. Añadir y listar los repositorios remotos

Una vez inicializado el repositorio localmente, se puede añadir uno (o más) repositorios remotos para poder tenerlo "sincronizado" en otro PC/-servidor/servicio. En las siguientes líneas, se explica la forma de hacerlo con servicios (plataformas) online ofertados gratuitamente.

Para ello, lo primero que hay que hacer es inicializar un nuevo repositorio en la plataforma que se haya escogido (GitHub , BitBucket etc.). Dicha plataforma permitirá obtener una URL (o algo equivalente) con el fin de poder acceder localmente al repositorio. Ahora, hay que añadir el repositorio remoto mediante el siguiente comando, siendo <remoteRepositoryURL> la URL indicada anteriormente:

```
1 git remote add origin <remoteRepositoryURL>
```

Extracto de código/comando: 6.18: Comando añadir repositorio remoto Git

Si se quiere ver los repositorios remotos que existen, se ejecutará el siguiente comando:

```
1 git remote -v
```

Extracto de código/comando: 6.19: Comando ver repositorios remotos Git

6.1.2.6. Interactuando con el repositorio remoto

En las siguientes dos páginas, se explican los comandos básicos para interactuar con un repositorio remoto [4].

Push: Este comando sirve para sincronizar los cambios guardados en el repositorio local, con el repositorio remoto.

```
1 git push <NombreRepositorioREMOTO> <NombreRAMA>
```

Extracto de código/comando: 6.20: Comando push Git

Fetch: Este comando obtiene todos los cambios del repositorio remoto y los descarga al repositorio local. Todos estos cambios no se combinan con el repositorio local, sino que se guardan en la rama de rastreo remoto.

```
1 git fetch <NombreRepositorioREMOTO>
```

Extracto de código/comando: 6.21: Comando fetch Git

Merge: Al ejecutarlo, combina los cambios locales con los cambios del repositorio remoto. Previamente, se habrá tenido que ejecutar el comando fetch.

```
1 git merge <NombreRepositorioREMOTO>/<NombreRAMA>
```

Extracto de código/comando: 6.22: Comando merge Git

Pull: Este comando combina los dos anteriores (fecth y merge), es decir, descarga los cambios que hay en el repositorio remoto y los combina con los locales.

```
1 git pull <NombreRepositorioREMOTO> <NombreRAMA>
```

Extracto de código/comando: 6.23: Comando pull Git

Clone: En el caso de que queramos clonar un repositorio remoto en nuestro ordenador o servidor, existe un comando de Git que aglutina varios pasos, evitando tener que ejecutar varios comandos:

```
1 git clone remoteRepositoryURL
```

Extracto de código/comando: 6.24: Comando clone Git

El resultado de la ejecución de dicho comando es el siguiente:

- Crea un nueva carpeta, llamada como el repositorio, donde se encuentra actualmente la línea de comandos.
- Inicializa el repositorio local Git.
- Añade el repositorio a la lista de repositorios remotos con el nombre "origin".
- Todos los archivos del repositorio remoto se descargan al local.
- La rama por defecto se desprotege.

6.2. Uso de Git en el desarrollo de software en el proyecto

En esta sección se explican qué pasos se han dado para el uso de Git en el entorno de desarrollo, a continuación, los motivos de la elección de BitBucket como plataforma online de Git en el proyecto, y por último, el uso de Git en el servidor de prueba.

6.2.1. Git en el entorno de desarrollo

A la hora de desarrollar el módulo de Magento de la prueba de concepto explicada en el capítulo anterior, se han generado varias versiones de código. Todas estas versiones han sido controladas mediante el software Git.

Para poder explicar el uso de Git en el proyecto, hay que mencionar que se han utilizado dos ordenadores para el desarrollo del módulo. El principal, ha sido el que ha proporcionado On4U, para su uso dentro de las instalaciones, y el otro, ha sido un portátil, que se ha utilizado en realización de tareas fuera de la empresa. Debido a esto, se ha necesitado un repositorio remoto para poder tener los dos ordenadores sincronizados. Éste, ha estado alojado en BitBucket, como se versa en la siguiente sección.

Los pasos seguidos para montar la infraestructura fueron los siguientes:

- Ordenador On4U

    1. Crear un nuevo proyecto de Magento.
    2. Hacer alguna modificación.
    3. Inicializar el repositorio local dentro de la carpeta del proyecto en el ordenador, mediante el comando "init".
    4. Añadir, mediante el comando "add", los archivos que se necesiten agregar al repositorio.
    5. Guardar los cambios (comando "commit").
    6. Crear un nuevo repositorio en BitBucket.
    7. Añadir ("Git remote add") el repositorio de BitBucket como repositorio remoto en el local.
    8. Sincronizar los cambios del repositorio local con el remoto, mediante el comando "push".

- Portátil

    1. Crear un nuevo proyecto de Magento.
    2. Clonar el repositorio de BitBucket mediante el comando "Git clone".

La mecánica de trabajo seguida ha sido muy simple. Cuando se empezaba a trabajar en un ordenador, se ejecutaba siempre el comando "pull", para descargar las modificaciones (si las hubiese). Si se hacían progresos, y se querían guardar, los archivos modificados se añadían al seguimiento del repositorio ("Git add") y se guardaban, haciendo un "commit". Posteriormente, se sincronizaba con el repositorio remoto, mediante el comando "push". De esta forma, los cambios siempre estaban sincronizados con el repositorio remoto, y se descargaban localmente al empezar a trabajar.

Por último, hay que mencionar que en el repositorio sólo se guardaban los archivos del módulo, y no todo el proyecto (genéricos para todos los proyectos). Para que fuese más fácil la tarea de añadir únicamente los archivos necesarios al seguimiento del repositorio, se creó el archivo .gitignore. Éste sirve para indicarle a Git de que archivos no tiene que hacer seguimiento.

6.2.2. Elección del soporte de Git en la nube (servicio escogido)

Hoy en día GitHub es la plataforma más extendida para alojar proyectos Git. Su versión gratuita permite tener repositorios ilimitados públicos, es decir, que cualquier persona puede verlos y contribuir de algún modo. Pero en muchos proyectos tanto de empresa, como de de otro tipo, son necesarios

repositorios privados. Debido a esto, GitHub tiene versión de pago, la cual está enfocada a empresas.

Aun así, hay muchas otras plataformas que, aunque no tengan la popularidad de GitHub , efectúan las mismas funciones (o incluso más) y además permiten tener gratuitamente repositorios privados. Este es el caso de BitBucket, la plataforma escogida en este proyecto.

6.2.3. Git en el servidor

Para poder comprobar el funcionamiento de la prueba de concepto, se contrató un servidor virtual "en la nube". En proyectos sin sistemas de control, la subida de los cambios al servidor se suele hacer a mano, pero en este caso se ha realizado con la ayuda de Git.

El primer paso, fue clonar el repositorio de BitBucket en el servidor. Para ello se utilizó el comando "clone" de Git. Por último, cada vez que se quería descargar los cambios realizados en el repositorio, se ejecutaba el comando "pull". Como se puede suponer, el proceso era muy sencillo y rápido.

Capítulo 7

Gestión del proyecto

El proyecto no ha tenido una planificación inicial detallada, sino que su desarrollo se ha ido basando en una *metaplanificación* que se abordó en la primera semana y que se ha ido modificando durante su desarrollo.

Son varios los motivos que llevaron a no elaborar una planificación preliminar. Las tecnologías a utilizar dentro del proyecto no eran las habituales y muchas de ellas eran totalmente desconocidas para la persona implicada en el proyecto. El ejemplo más claro fue la propia plataforma de eCommerce utilizada. Su funcionamiento y arquitectura son tan avanzadas, que el simple hecho de entenderlas para poder implementar la solución de la prueba de concepto, llevó a emplear mucho tiempo del proyecto. Otro ejemplo, podría ser el control de versiones de Git, que, aunque aparentemente es fácil de usar, es una herramienta tan compleja que en su aprendizaje se invirtió más tiempo del esperado. Además, al contemplar en el proyecto una prueba de concepto, la planificación detallada no es la más adecuada.

La *metaplanificación* comentada anteriormente, ha consistido en ir definiendo objetivos y ampliando progresivamente el alcance del proyecto, según se iban desarrollando las metas anteriores. Cada vez que se definía un objetivo, se le asignaba una fecha límite, aunque esa fecha pudiese variar dependiendo del transcurso de su desarrollo.

Este tipo de filosofía, es muy parecida a la de *Lean Startup* [14] [24], que sin tener que planificar el proyecto de una manera inicial, se basa en una serie de principios que sirven a modo de *metaplanificación*.

Durante el transcurso de la realización del proyecto, se ha ido escribiendo un diario, el cual se describe a continuación. Junto a la explicación del diario, se puede encontrar la gestión del alcance, del tiempo y de los costes.

7.1. Diario

Según se iba desarrollado el proyecto, se iba escribiendo un diario, en el cual se detallaba día a día las tareas realizadas, junto con el coste de tiempo que habían supuesto.

También se han ido adjuntando enlaces a páginas web, así como, los títulos de libros que se utilizaban. Asimismo, muchas de las tareas se han explicado de forma similar a la de un manual.

Se ha usado un sistema de "tags", con el objetivo de clasificar y posteriormente, poder encontrar de forma más sencilla las diferentes tareas.

El motivo de escribir el diario no ha sido otro, que llevar el control de las tareas realizadas cada día, para tenerlas guardadas y que fuese una herramienta útil a la hora de confeccionar este documento.

7.2. Gestión del alcance

Tal y como se ha mencionado en las secciones anteriores, el alcance del proyecto ha ido ampliándose a medida que se iban logrando los objetivos marcados. Dicho alcance, ha quedado divido en tres fases.

La primera, se centró en la familiarización con las diferentes tecnologías usadas en su confección. Esto, abarco la familiarización con el software de control de versiones Git y con la plataforma Magento, tanto con la arquitectura utilizada en ésta por los desarrolladores, como con el patrón de diseño Modelo-Vista-Controlador. Esta fase, que se le dedico aproximadamente tres meses, consumió más o menos el 50 % de los recursos disponibles.

Las siguientes dos fases, que se pueden categorizar como *evolutivas- experimentales*, se han desarrollado con los recursos restantes. Es por ello, que la segunda fase abarcó dos tercios de los recursos sobrantes, centrándose en la prueba de concepto de los diferentes casos de uso que se han descrito en el capítulo de objetivos del proyecto, en términos de historias de usuario.

Por último, la tercera fase ha concluido con la prueba de concepto, terminando con los recursos disponibles en el proyecto.

7.3. Gestión del tiempo

Seguidamente, se detallan los hitos más importantes que ha tenido el proyecto:

- **Comienzo del proyecto:** 11 de enero de 2015
- **Definición del título del proyecto:** 22 de enero de 2015
- **Inscripción en GAUR:** 29 de enero de 2015
- **Incorporación a On4U:** 16 de febrero de 2015
- **Finalización de la implementación de la prueba de concepto:** 27 de mayo de 2015
- **Finalización de la primera versión completa de la memoria del proyecto:** 20 de junio de 2015
- **Deposito de la versión final de la memoria:** 24 de junio de 2015

7.4. Gestión de costes

El coste económico que ha supuesto el proyecto, es totalmente despreciable en comparación con el tiempo que ha supuesto el volumen de trabajo del proyecto. Debido a esto, se excluyen de esta memoria los costes de carácter económico.

En la tabla 7.1 de la página 75, se puede apreciar el coste de tiempo que ha supuesto, dividido en paquetes de trabajo. Hay que tener en cuenta, que gran parte del tiempo dedicado al proyecto ha sido académico.

Paquete de trabajo	**Actividad/Tarea**	**Dedicación** (h)
Tecnología a desarrollar	Magento y MVC	135
	Git	30
Prueba de concepto		85
Gest. del proyecto	Alcance, tiempo y costes	15
	Diario	40
	Reuniones y otras tareas en On4U	20
Trabajo académico	Memoria	60
	Reunión con el director académico	25
	Defensa	10

Gest. = gestión

Cuadro 7.1: Tabla de dedicación

Capítulo 8

Conclusiones

En este proyecto, se ha alcanzado el objetivo de implementar un módulo de Magento como prueba de concepto de la idea de generación dinámica de parrillas en base al análisis de datos de fuentes abiertas.

El hecho de haber realizado la prueba de concepto mediante la plataforma de eCommerce Magento, ha posibilitado profundizar y poner en práctica muchos conceptos estudiados durante los cuatro años del Grado de Ingeniería Informática. Entre ellos, destaca el concepto del patrón Modelo-Vista-Controlador, no sólo porque los desarrolladores de Magento le hayan dado mucha importancia, sino porque su implementación en la propia plataforma es mucho más avanzada que la implementación del patrón clásico. De este modo, se ha tenido que analizar el funcionamiento de la plataforma para entender de manera correcta su funcionamiento; es decir, la implementación, poco usual, del patrón llevó a la necesidad de documentarse sobre otros conceptos, como el ciclo de vida de una petición, que a su vez, condujo a entender la importancia del patrón MVC en toda la arquitectura de Magento.

Dicho esto, ha quedado patente la importancia de usar patrones de diseño en la programación de software, no sólo para tener el código de los distintos archivos fuente organizados, sino que también para simplificar y agilizar la tarea de programar. Además, el simple hecho de utilizar los patrones, ayuda a otros programadores a entender el código escrito por otras personas.

No hay que olvidar, la importancia que ha tenido en el proyecto el uso de fuentes abiertas de información a la hora de realizar la prueba de concepto. En los últimos años, el número de webs destinadas a ofrecer datos de diferente índole de manera libre se ha incrementado exponencialmente, en muchos casos potenciado por la popularización del uso del *Big Data*. Esto

lleva a pensar que en un futuro, no muy lejano, se usarán "masivamente" estos servicios para facilitar las distintas tareas del día a día.

Por otra parte, el requisito inicial propuesto por On4U, de utilizar la herramienta Git como software de control de versiones, ha posibilitado adquirir conocimientos ligados con el modo de trabajar actual en el ámbito de la programación de software. No sólo se ha entendido el modo de funcionar de Git, sino que también se ha llevado a la práctica con el control de versiones de la prueba de concepto. Obviando la necesidad de haber tenido que documentarse dada la casi inexperiencia en este tipo de herramientas al comenzar el proyecto, su uso ha sido muy positivo. Ha permitido, con unos pocos comandos, tener una copia de seguridad de las distintas versiones del código en la "nube", tener los dos ordenadores de trabajo utilizados durante la implementación del módulo siempre sincronizados y, lo que desde un punto de vista personal es lo más sorprendente e importante, poder descargar en el servidor los cambios realizados en cuestión de segundos.

Además, el haber realizado el proyecto dentro de las instalaciones de On4U, ha permitido conocer el ámbito laboral, que de otra manera, no se hubiese conocido dentro de la universidad. También ha aportado nuevas experiencias, nuevas formas de organizarse y nuevos métodos de trabajo. Asimismo, el hecho de que sea una empresa con características innovadoras y poco tradicionales, ha llevado a ver de distinta manera, a como se percibía, el mundo laboral, de forma que en caso de emprender y montar una empresa, se tendría muy en cuenta las experiencias vividas, no sólo académicas y laborales, sino también, personales. Sirva como ejemplo, el que los empleados y empleadas no tengan un horario fijado por los responsables de On4U y el poder trabajar en días puntuales desde fuera de la oficina, lo que permite tener un clima de trabajo mucho más tranquilo y de confianza. Hay que puntualizar que esto no conduce al descontrol, ya que el trabajo y las horas trabajadas diariamente, se controlan desde una intranet propia que también permite, entre otras cosas, comunicarse entre los empleados y empleadas.

Hay que resaltar que la ejecución de este proyecto no hubiera sido posible sin la adquisición académica obtenida durante los cuatro años del Grado. Esta adquisición de conocimientos, que en algunas ocasiones se percibe como demasiado teórica o académica, en el momento de ponerla en práctica, se convierte en la herramienta que posibilita la utilización de nuevas tecnologías y el aprendizaje de ellas.

Para finalizar, este proyecto no ha sido solamente un factor de enriquecimiento académico y laboral, sino que también ha permitido concluir los cuatro años de Ingeniería Informática cursados en la facultad de Informática de Donostia, de una forma práctico-experimental.

Capítulo 9

Propuestas de mejora

El proyecto, dado su carácter académico y de tiempo limitado, ha tenido que llegar a su final, y en el cierre de éste se han tenido que dejar varias propuestas de mejora en el tintero. Estas propuestas han ido surgiendo en el propio desarrollo o con aportaciones externas realizadas al mostrar el proyecto a otras personas. A continuación, se mencionan algunas de las propuestas de mejora más importantes.

La implementación de la prueba de concepto realizada en el módulo se ha hecho modular, de tal manera que cambiando pocas líneas del código se podría modificar el funcionamiento para que realizase otro caso de uso diferente. Precisamente en esto se centra la primera propuesta de mejora, planteando que las modificaciones del comportamiento del módulo las pudiesen hacer los usuarios/administradores desde el panel de administración de Magento. De esta forma, un usuario podría cambiar (añadiendo, modificando o eliminando) los grupos de productos creados para organizarlos, o cambiar la fuente de datos de donde se obtiene la información que posteriormente se procesa, utilizando otra que ofrezca el mismo servicio (la meteorología) u otro totalmente diferente.

La siguiente propuesta consiste en que la generación dinámica de parrilla de productos se pudiese extender a cualquier página de la plataforma de Magento, y no sólo como aparece en este proyecto, en la "home" o en la páginas del módulo. De esta forma, los productos de las distintas categorías que se muestran en las páginas de Magento se podrían ordenar teniendo en cuenta, por ejemplo, la relevancia de los artículos y otro factor externo.

Otra opción podría ser que el propio usuario pudiese determinar ciertas premisas, para que el algoritmo de procesamiento de información las tuviese en cuenta. Por ejemplo, un usuario se va a ir de viaje a Bélgica, pero el en el momento de entrar a la tienda está en Barcelona. La organización de

los productos de la tienda se mostraran en base a la información de Barcelona, pero él cliente, podría determinar que se mostrasen en base a Bélgica.

Para finalizar, se cierra este capítulo con la propuesta más ambiciosa de todas. Cuando el cliente añade un producto a la cesta de la compra, el módulo registra y guarda en la base de datos la información referente a ese evento y a la información meteorológica obtenida (o algún otro factor externo). Todos esos datos se guardan, pero no se procesan. Dependiendo del volumen de ventas de la tienda online donde se haya implementado, se podría estar hablando de miles o millones de filas en la base de datos, lo suficiente para considerarlo *Big Data*. Por esto mismo, una de las ideas finales del proyecto consiste en procesar toda esa información con herramientas destinadas a ese fin, y conseguir unas conclusiones que emparejen las ventas de ciertos productos con la información externa obtenida. De esta forma, se podría ir afinando el algoritmo de análisis de los datos externos con el objetivo de lograr una experiencia de usuario mejor.

Glosario

Adolfo Domínguez Adolfo Domínguez S.A. es una empresa centrada en la moda, cuyo fundador fue Adolfo Domínguez. 27

Android Sistema operativo basado en el núcleo Linux, diseñado principalmente para dispositivos móviles con pantalla táctil. 27

Apache Servidor web de código abierto. 27, 41, 50, 93, 95

Apache Cordova Plataforma para la construcción de aplicaciones para móvil utilizando HTML, CSS y JavaScript . 27

API Application Programming Interface. 29, 32, 52, 53

Big Data Acumulación masiva de datos, para su posterior análisis y procesamiento, con el fin de identificar patrones recurrentes dentro de esos datos. 21, 25, 26, 30, 56, 77, 80

BitBucket Servicio de alojamiento basado en web, para proyectos que utilizan el sistema de control de versiones. 22, 68, 70–72

CE *Community Edition*. 28

CSS3 Hojas de estilo en cascada. 27

DNS Sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. 92

eBay Es un sitio web destinado a la subasta de productos a través de Internet. 28, 29

eCommerce Comercio electrónico. 21, 23, 25–29, 31, 37, 38, 73

ERP Sistema de Planificación de Recursos Empresariales. 29

framework Estructura conceptual y tecnológica de soporte definido, de software concretos, que puede servir de base para la organización y desarrollo de software. 27, 28

fuentes abiertas Páginas webs dónde se ofrecen datos de forma abierta, es decir, sin ningún tipo de restricción de derechos de autor, de patentes o de otro tipo de licencias. 21, 25, 26, 30, 31, 77

Git Software de control de versiones diseñado por Linus Torvalds. 21, 22, 26, 28–31, 33, 65–67, 70–75, 78

GitHub Plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. 30, 68, 71, 72

GNU GPL La licencia más ampliamente usada en el mundo del software, garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software. 27

GNU GPL v2 Versión 2 de la licencia más ampliamente usada en el mundo del software, garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software. 28

GNU/Linux Términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux con el sistema GNU. 27

Google Empresa multinacional estadounidense especializada en productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías. 25

handle Tipo particular de punteros "inteligentes". 44

home Inicio. 32, 33, 59, 60, 79

HTML Lenguaje de marcado para la elaboración de páginas web. 42, 44, 54

HTML5 Quinta revisión importante del lenguaje básico de la World Wide Web, HTML. 27

internauta Neologismo, utilizado normalmente para describir a los usuarios habituales de Internet. 26

iOS Sistema operativo móvil de la multinacional Apple Inc. 27

IP Número que identifica a cada dispositivo dentro de una red con protocolo IP. 32, 35, 50, 52, 53, 60, 61, 91–93

JavaScript Lenguaje de programación interpretado. 27

JSON Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. 52, 54, 55

LibreOffice Paquete de oficina libre y de código abierto desarrollado por The Document Foundation. 28

Linus Torvalds Es un ingeniero de software finlandés, conocido por iniciar y mantener el desarrollo del kernel Linux, basándose en el sistema operativo libre Minix creado por Andrew S. Tanenbaum y en algunas herramientas, varias utilidades y los compiladores desarrollados por el proyecto GNU . 29

Linux Kernel (núcleo) del sistema del sistema operativo "GNU/Linux". 30

low-cost Servicio ofrecido a un precio inferior al de la competencia. 33

Mac OS Nombre del sistema operativo creado por Apple para su línea de computadoras Macintosh. 65

Magento Es un gestor de contenido web desarrollado con Software Libre destinado al eCommerce. 21–23, 26–29, 31, 32, 34, 37–41, 43–47, 49, 50, 55, 56, 59, 66, 70, 71, 74, 75, 77, 79, 93, 97, 98

Magento Connect Mercado de aplicaciones de comercio electrónico. 28

market Tienda. 29

MarketPlace Es un sitio que permite a vendedores y compradores relacionarse para efectuar una transacción comercial. 28

MIT Esta licencia permite reutilizar el software así licenciado tanto para ser software libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original. También permite licenciar dichos cambios con licencia BSD, GPL, u otra cualquiera que sea compatible.. 28

multistore Multi-tienda. 29

MVC Modelo Vista Controlador, un patrón de diseño en Ingeniería de Software. 31, 37–39, 41–44, 46, 47, 75, 77

MySQL Sistema de gestión de bases de datos relacional, multihilo y multiusuario. 27, 93–95, 97

namespace Conjunto de nombres en el cual todos los nombres son únicos. 39

On4U On4U Global Services es una agencia especializada en eCommerce. `on4u.es`. 21, 22, 26, 27, 29, 32, 33, 65, 70, 71, 75, 78, 89

Open Source Expresión con la que se conoce al software o hardware distribuido y desarrollado libremente. 27, 97

OpenWeatherMap Servicio online que ofrece una API de datos meteorológicos, incluidos los datos del tiempo actual, previsiones y datos históricos a los desarrolladores de servicios web y aplicaciones móviles . 53

OVH Proveedor de alojamiento web francés. 22, 92

partner Socio de una empresa, que se dedica a distribuir un Producto ó Servicio. 28, 29

PHP Lenguaje de programación enfocado a la web. 21, 40, 41, 53, 55, 94, 95

PHPMyAdmin Herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web. 93, 95, 97

phtml Extensión utilizada por el lenguaje de programación PHP para plantillas HTML. 45

portfolio Se refiere a una recopilación de documentos que pueden mostrar diferentes aspectos globales o parciales de una persona (personales, académicos, profesionales...) o de una organización. 27

Prestashop Es un gestor de contenidos cms libre, de código abierto. 29

registro A Enlaza un dominio con una dirección IP . 92

SEO *Search Engine Optimization*, posicionamiento en buscadores. 28

servicio web Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. 41

SM (Grupo SM) es una editorial española. 27

Software Libre Es el software que puede ser ejecutado, copiado, distribuido, estudiado, modificado y mejorado libremente, aunque no tiene porque ser gratuito. 28, 29

ssh Al igual que telnet, protocolo de red que nos permite viajar a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella. 93

Symfony Completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. 28

telnet Al igual que shh, protocolo de red que nos permite viajar a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella. 93

TFG Trabajo Fin de Grado. 21, 92

Ubuntu Es un sistema operativo basado en GNU/Linux que se distribuye como software libre. 23, 27, 65, 66, 93, 96

URL Localizador de recursos uniforme, formados por una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que designa recursos en una red. 40, 41, 44, 47, 49, 50, 53, 54, 68, 95, 98

VPS Virtual Private Server, Servidor Virtual Privado. 91, 92

Web services Servicios Web. 29

Webmin Herramienta de configuración de sistemas accesible vía web para sistemas Unix, como GNU/Linux y OpenSolaris. 93, 96

Windows Nombre de una familia de distribuciones de software para PC, smartphone, servidores y sistemas empotrados, desarrollados y vendidos por Microsoft. 65

Windows Phone Sistema operativo móvil desarrollado por Microsoft. 27

X.Commerce Plataforma dirigida a los desarrolladores creada por eBay, PayPal y el gestor de contenidos OpenSource Magento. 28, 29

XML Lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos de forma legible. 39, 44, 54

Algunos de los términos anteriores han sido definidos gracias a la enciclopedia libre Wikipidea.

Bibliografía

[1] Branko Ajzele. *Magento – Install, install upgrade, data and data upgrade scripts*. http://inchoo.net/magento/magento-install-install-upgrade-data-and-data-upgrade-scripts/. [Online; accedido el 30-05-2015] (vid. pág. 56).

[2] Scott Chacon y Ben Straub. *Pro Git book. Apress, 2014. ISBN: 978-1-484-20077-3*. http://git-scm.com/book/es/v2. [Online; accedido el 30-05-2015] (vid. pág. 65).

[3] belvg blog. *Magento Module Structure (Magento Certified Developer Exam)*. http://blog.belvg.com/magento-certification-module-structure.html. [Online; accedido el 30-05-2015] (vid. pág. 39).

[4] Roger Dudler. *Git - la guía sencilla*. http://rogerdudler.github.io/git-guide/index.es.html. [Online; accedido el 30-05-2015] (vid. pág. 69).

[5] github.io. *Git for Windows*. http://msysgit.github.io/. [Online; accedido el 30-05-2015] (vid. pág. 65).

[6] magentocommerce.com. *Magento MVC*. http://www.magentocommerce.com/images/uploads/Magento_MVC.pdf. [Online; accedido el 30-05-2015] (vid. pág. 43).

[7] Wiki magentocommerce.com. *Magento folder structure*. http://www.magentocommerce.com/wiki/2_-_magento_concepts_and_architecture/magento-folder-structure. [Online; accedido el 30-05-2015] (vid. pág. 39).

[8] marketingdirecto. *La gran evolución de internet desde su creación en 1969*. http://www.marketingdirecto.com/actualidad/infografias/la-gran-evolucion-de-internet-desde-su-creacion-en-1969/. [Online; accedido el 30-05-2015] (vid. pág. 25).

[9] Marcos Merino. *¿Qué es la inteligencia de fuentes abiertas?* http://www.ticbeat.com/bigdata/que-es-la-inteligencia-de-fuentes-abiertas/. [Online; accedido el 30-05-2015] (vid. pág. 25).

[10] Energía y Turismo Ministerios: Hacienda y Administraciones Públicas // Industria. *Reutiliza la información pública*. `http://www.datos.gob.es/`. [Online; accedido el 30-05-2015] (vid. pág. 30).

[11] On4U. *Expertos en Magento*. `http://www.on4u.es/magento.html`. [Online; accedido el 30-05-2015] (vid. págs. 28, 29).

[12] On4U. *Magento vs Prestashop*. `http://www.on4u.es/blog/2012/02/15/magento-vs-prestashop/`. [Online; accedido el 30-05-2015] (vid. pág. 29).

[13] Eric Ries. *Magento 1.8 Development Cookbook. Packt Publishing , febrero 2014. ISBN: 978-1-782-16332-9*. `https://www.packtpub.com/web-development/magento-18-development-cookbook`. [Online; accedido el 30-05-2015] (vid. pág. 89).

[14] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses (ingles). Crown Business, septiembre 2011. ISBN: 978-0-307-88789-4*. `http://theleanstartup.com/`. [Online; accedido el 30-05-2015] (vid. pág. 73).

[15] Samuel Rodríguez. *60 años de comercio electrónico; La evolución de las compras online*. `http://ecommerce-news.es/servicios/metodos-de-pago/60-anos-de-comercio-electronico-la-evolucion-de-las-compras-online-11327.html`. [Online; accedido el 30-05-2015] (vid. pág. 25).

[16] Alan Storm. *In Depth Magento Dispatch: Standard Router*. `http://alanstorm.com/magento_dispatch_standard_router`. [Online; accedido el 30-05-2015] (vid. pág. 50).

[17] Alan Storm. *Layouts, Blocks and Templates*. `http://alanstorm.com/layouts_blocks_and_templates`. [Online; accedido el 30-05-2015] (vid. pág. 45).

[18] Alan Storm. *Magento Controller Dispatch and Hello World*. `http://alanstorm.com/magento_controller_hello_world`. [Online; accedido el 30-05-2015] (vid. págs. 37, 41, 44).

[19] Alan Storm. *Magento Models and ORM Basics*. `http://alanstorm.com/magento_models_orm`. [Online; accedido el 30-05-2015] (vid. pág. 40).

[20] Alan Storm. *Magento MVC Flow*. `http://alanstorm.com/2009/img/magento-book/magento-mvc.png`. [Online; accedido el 30-05-2015] (vid. pág. 46).

[21] Alan Storm. *Magento Setup Resources*. `http://alanstorm.com/magento_setup_resources`. [Online; accedido el 30-05-2015] (vid. pág. 56).

[22] Rajat Jain - techsharepoint.com. *Module Configuration and URL Structure*. `http://techsharepoint.com/module-configuration-and-url-structure/`. [Online; accedido el 30-05-2015] (vid. pág. 40).

[23] Timcharper. *Git for Mac OS X - git-osx-installer*. `http://sourceforge.net/projects/git-osx-installer/`. [Online; accedido el 30-05-2015] (vid. pág. 65).

[24] Wikipedia. *Lean Starup*. `http://es.wikipedia.org/wiki/Lean_startup`. [Online; accedido el 30-05-2015] (vid. pág. 73).

Gran parte del proyecto se ha basado en el libro Magento 1.8 Development Cookbook [13], facilitado por On4U.

Apéndice A

Elección del servidor y del dominio

Un servidor web en una máquina local puede ser suficiente para determinados proyectos. Pero utilizar la máquina local, suele conllevar el hecho de que desde fuera de la red local no se pueda acceder a él.

Por este motivo, es interesante tener un servidor web en la "nube", sencillo, accesible desde cualquier punto. Así mismo, con la dirección IP puede llegar a ser suficiente si el proyecto no se va a hacer público, aunque en determinadas ocasiones puede ser útil contratar un dominio y enlazarlo al servidor.

A.1. Elección del servidor

Tradicionalmente, siempre han existido dos tipos de servidores:

- Servidor dedicado: un servidor con un hardware concreto del que dispones del 100 % de los recursos
- Servidor virtual (o VPS): el servidor se comparte entre varios clientes, y a cada cliente se le asignan ciertos recursos.

Hoy en día, a estos dos modelos tradicionales de servidores, hay que añadirle un tercero: servicios de computación en la nube. El ejemplo más importante de hoy en día es Amazon Web Services (AWS). Este tipo de servicios funcionan de diferente manera a los anteriormente citados: cuando se contrata un recurso, éste no pertenece a una sola máquina sino que puede estar distribuido entre varias, comunicadas entre ellas. Esto conlleva que los recursos se pueden ir asignando dinámicamente y con un límite holgado. De esta forma, si en un momento concreto necesitamos más RAM o más rapidez de cálculo, se irán asignando automáticamente más recursos. Además,

se puede contratar por horas, días, meses o años, con lo que se ajusta mucho mejor a todo tipo de proyectos.

En el caso de este TFG, y teniendo en cuenta que no se va a llevar a producción, se ha decidido contratar un servidor VPS por varios motivos:

- Un VPS sencillo es suficiente para la carga de trabajo que se le va a hacer soportar.
- En caso de necesitar más recursos se podría escalar.
- Económicamente hablando, al mes no supone mucho dinero, ya que partimos de que se pueden contratar desde los 2€/mes.

Este proyecto ha estado alojado en uno de los servidores virtuales (`http://www.ovh.es/vps/`) que ofrece OVH, concretamente en el más básico (`http://www.ovh.es/vps/vps-classic.xml`).

A.2. Enlazar dominio con IP

Muchas veces al contratar un dominio y un servidor, estos dos no se contratan con el mismo proveedor. Esto conlleva que hay que enlazar ese dominio con la dirección IP del servidor contratado.

Para ello, hay que añadir un nuevo registro A (o también llamado de dirección) a la configuración del dominio. En ese registro A, habrá que añadir la dirección IP del servidor.

Una vez realizado estos cambios, habrá que esperar, ya que la propagación entre los servidores DNS puede llevar varias horas.

Apéndice B

Instalación del entorno necesario en Ubuntu

Un entorno básico para que funcione correctamente Magento en Ubuntu sería el siguiente: Apache + PHP5 + MySQL. A esto añadiremos PHPMyAdmin para poder tener una interfaz gráfica de la base de datos (MySQL) y Webmin, que lo usaremos para poder configurar el servidor.

B.1. Apache + PHP5 + MySQL

Lo primero que hay que hacer es conectarse mediante telnet o ssh al servidor. En caso de estar en la maquina local, bastaría con abrir una línea de comandos. Se recomienda autenticarse como root, para tener todos los derechos.

Seguidamente hay que instalar Apache Para ello, sólo habrá que ejecutar los siguientes comandos:

```
1 sudo apt-get update
2 sudo apt-get install apache2
```

Una vez ejecutados estos dos comandos, si entramos por el navegador a la dirección IP del servidor, tendría que mostrarse una página web en blanco con el siguiente texto: “It works! This is the default ...”.

A continuación, para instalar MySQL se ejecuta el siguiente comando:

```
1 sudo apt-get install mysql-server libapache2-mod-auth-mysql
2    php5-mysql
```

Durante la instalación, el asistente nos preguntará por la contraseña de root. Una vez instalado MySQL, hay que ejecutar dos comandos para inicializar MySQL y dejar la base de datos preparada para producción.

```
1 sudo mysql_install_db
2 sudo /usr/bin/mysql_secure_installation
```

Cuando se ejecute el segundo comando, éste hará varias preguntas:

- Enter current password for root (enter for none): como su nombre indica, la contraseña del usuario root (se ha introducido anteriormente).
- Change the root password?: No.
- Remove anonymous users?: Yes.
- Disallow root login remotely?: Yes.
- Remove test database and access to it? Yes.
- Reload privilege tables now? Yes.

Por último, hay que instalar PHP. Para ello, habrá que ejecutar el siguiente comando:

```
1 sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
2   php5-curl
```

Para comprobar que se ha instalado correctamente, hay que añadir un nuevo archivo:

```
1 sudo nano /var/www/info.php
```

Con el siguiente código:

PHPInfo

```
1 <?php
2 phpinfo();
```

Ahora si accedemos a ese archivo desde el navegador (IP/info.php), se mostrará una página con toda la información sobre PHP.

Una vez comprobado que funciona, habrá que borrar el archivo con el siguiente comando:

```
1 sudo rm /var/www/info.php
```

B.2. PHPMyAdmin

PHPMyAdmin es una herramienta para administrar las bases de datos MySQL a través del navegador. Está escrita en PHP.

Su instalación es muy sencilla: el primer paso es ejecutar el siguiente comando:

```
1 sudo apt-get install phpmyadmin apache2-utils
```

El asistente hará varias preguntas:

- Servidor web que desea reconfigurar automáticamente: apache2
- ¿Desea configurar la base de datos para phpmyadmin con «dbconfig-common»?: Sí
- Contraseña del usuario de administración de la base de datos: contraseña de usuario root de MySQL
- Contraseña de aplicación MySQL para phpmyadmin: contraseña con la que habrá que identificarse en PHPMyAdmin

Una vez instalado, habrá que modificar la configuración de Apache. Para ello, se ejecutará el siguiente comando:

```
1 sudo nano /etc/apache2/apache2.conf
```

Al final del archivo que se ha abierto, hay que añadir lo siguiente:

```
1 #Include PHPMyAdmin Configuration
2 Include /etc/phpmyadmin/apache.conf
```

Se reinicia el servicio de Apache:

```
1 sudo service apache2 restart
```

Ahora, si accedemos mediante un navegador a la siguiente URL IP/phpmyadmin/, se tendría que ver la página de identificación de PHPMyAdmin.

B.3. Webmin

Dado que en Ubuntu no está añadido el repositorio de Webmin, habrá que añadirlo a mano. Para ello, modificaremos el siguiente archivo:

```
sudo nano /etc/apt/sources.list
```

Al final del archivo, se añade lo siguiente:

```
##### Webmin repo
deb
   http://download.webmin.com/download/repository sarge contrib
deb
   http://webmin.mirror.somersettechsolutions.co.uk/repository
   sarge contrib
```

Ahora se añadirá la clave del repositorio al APT, para que el servidor sea de confianza:

```
wget -q http://www.webmin.com/jcameron-key.asc -O-
   | sudo apt-key add -
```

A continuación, se ejecutarán los siguientes comandos:

```
sudo apt-get update
sudo apt-get install webmin
```

Por último, se accederá a Webmin mediante la siguiente dirección en el navegador: https://IP:10000/. El usuario y la contraseña serán los de Ubuntu (root).

Apéndice C

Instalación de Magento

Lo primero que hay que hacer es descargarse el fichero con la versión Open Source (Magento Community Edition) desde la página web de Magento (`https://www.magentocommerce.com/products/downloads/Magento/`). Cuando se descargue, habrá que descomprimirlo en la carpeta pública del servidor web.

Una vez descomprimido, hay que entrar desde el navegador. Se mostrará un asistente que nos guiara durante la instalación. Antes de seguir con el asistente, habrá que crear una nueva base de datos (por ejemplo, mediante la herramienta PHPMyAdmin), ya que éste no la creará automáticamente. Las preguntas serán las siguientes (si alguna pregunta no aparece, hay que dejarla como viene por defecto):

- I agree to the above terms and conditions: Yes.
- Locale: Español (España)
- Time Zone: Romance Standard Time (Europe/Madrid).
- Default Currency: euro.
- Database Type: MySQL
- Host: localhost
- Database Name: Magento
- User name: usuario de la base de datos
- User password: la contraseña del usuario
- First Name: Nombre
- Last Name: Apellido

- Email: email
- Username: el nombre del usuario de administración de Magento.
- Password: la contraseña del usuario de administración de Magento.

Algunos de los pasos del asistente pueden llevar varios minutos, dependiendo de la capacidad del servidor. Una vez que el asistente haya terminado, se habrá instalado una versión limpia de Magento. Para acceder al panel de administración tendremos que introducir en el navegador la siguiente URL: IP/index.php/admin

Si en algún momento de la instalación, el asistente diera algún problema de permisos, habría que ejecutar los siguientes comandos dentro de la carpeta de Magento:

```
1 chown -R www-data .
2 find . -type d -exec chmod 700 {} \;
3 find . -type f -exec chmod 600 {} \;
```