

TRABALLO FIN DE GRAO
GRAO EN ENXEÑERÍA INFORMÁTICA
MENCIÓN EN ENXEÑERÍA DO SOFTWARE

Aplicación web para la minería de ofertas de productos a partir de canales de Telegram

Estudiante: Álvaro Balirac Seijas
Director/a/es/as: Javier Parapar López
Paula López Otero
Daniel Valcarce Silva

A Coruña, 3 de septiembre de 2019.

A mi mujer Sandra por apoyarme y siempre creer en mí.

Agradecimientos

A mis directores Daniel, Paula y Javier, por el excelente trato que me han dado durante todo el proceso y los magníficos consejos y toda la ayuda sin la que este proyecto no podría haber salido adelante.

A mi mujer y a mi familia por el apoyo incondicional que me han dado durante todos estos años.

A mis compañeros de la facultad y amigos por ayudar a que este trayecto fuera mucho más fácil y llevadero.

Y a toda la gente que siempre me ha apoyado, gracias.

Resumen

El proyecto consiste en el diseño y desarrollo de una aplicación web para la visualización de ofertas de productos de tiendas online como Amazon o Gearbest. Las ofertas son obtenidas mediante minería de datos realizada a canales en la plataforma de mensajería instantánea Telegram. El proyecto está dividido siguiendo una arquitectura basada en servicios, comenzando por un servicio que consiste en un bot de Telegram capaz de minar datos de canales en los que se comparten ofertas de productos y son almacenados en una base de datos a través de una API.

A medida que el bot va extrayendo información de los canales, otro servicio realiza análisis de texto sobre los datos recabados, extrayendo información útil de las distintas ofertas, como la URL, el precio en oferta, el nombre del producto o la tienda. Estas ofertas minadas y clasificadas son almacenadas en la base de datos a través de la API y posteriormente usadas en otro servicio consistente en una aplicación web para su búsqueda y visualización.

El resultado es una aplicación web que, de forma automática, muestra casi en tiempo real información sobre ofertas de productos en las principales tiendas online del mundo, extraídas y clasificadas a partir de multitud de canales de Telegram.

Abstract

This project consists in the design and development of a web application for visualizing product offers from online stores such as Amazon or Gearbest. We obtain offers from Telegram (an instant messaging platform) channels through data mining. The project is divided following a service-based architecture, starting with a service consisting of a Telegram bot capable of mining data from channels in which product offers are shared and stored in a database through an API.

As the bot extracts information from the channels, another service performs text analysis on the collected data, extracting useful information from the different offers, such as the URL, the offer price, product name, or store. These mined and classified offers are stored in a database through an API and then used in another service consisting of a web application for search and viewing.

The result is a web application that automatically shows, almost in real time, information about product offers in the main online stores of the world, extracted and classified from a multitude of Telegram channels.

Palabras clave:

- SPA
- API
- REST
- Angular
- Laravel
- Web
- PHP
- Javascript
- Elasticsearch
- SCRUM
- MySQL
- Minería de datos
- Web Scraping
- Análisis de texto
- Goutte
- Lumen
- cURL
- NPL

Keywords:

- SPA
- API
- REST
- Angular
- Laravel
- Web
- PHP
- Javascript
- Elasticsearch
- SCRUM
- MySQL
- Data Mining
- Web Scraping
- Text Analysis
- Goutte
- Lumen
- cURL
- NPL

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	2
1.3	Visión global del sistema	3
1.4	Estructura de la memoria	3
1.5	Plan de trabajo	4
2	Conceptos	5
2.1	Introducción	5
2.2	Minería de texto	6
2.3	Análisis de texto	7
2.4	Web scraping	8
2.5	Web crawling	9
3	Tecnología	11
3.1	Extracción de datos	11
3.1.1	MadelineProto	11
3.1.2	Goutte	12
3.2	Persistencia de datos	12
3.2.1	MySQL	12
3.2.2	Elasticsearch	13
3.3	Acceso a datos	14
3.3.1	Laravel Lumen	14
3.3.2	PHP libcurl	14
3.4	Análisis de datos	14
3.4.1	PHP NlpTools	15
3.5	Visualización de datos	15
3.5.1	Angular	15

3.6	Herramientas de desarrollo	18
3.6.1	PhpStorm	18
3.6.2	Git	19
3.6.3	Taiga	19
3.6.4	Npm	19
3.6.5	Composer	20
3.6.6	Vagrant	20
3.6.7	Oracle VM Virtual Box	21
3.6.8	Travis-CI	21
3.6.9	Overleaf	21
3.6.10	Postman	22
4	Proceso de ingeniería	23
4.1	Metodología	23
4.1.1	Scrum	23
4.1.2	Aplicación de Scrum	26
4.2	Gestión del proyecto	29
4.2.1	Estimación	29
4.2.2	Planificación	30
4.2.3	Recursos	30
4.2.4	Costes	31
4.2.5	Gestión de riesgos	32
5	Desarrollo	35
5.1	Análisis de Requisitos	35
5.1.1	Requisitos funcionales	35
5.1.2	Requisitos no funcionales	35
5.2	Arquitectura propuesta	36
5.3	API REST	37
5.3.1	Descripción y diseño de funcionalidades	37
5.4	Bot de Telegram	41
5.5	Elasticsearch	42
5.5.1	Búsqueda de ofertas	42
5.5.2	Categorización de ofertas	43
5.6	Text Miner	44
5.6.1	Análisis de precios	45
5.6.2	Análisis de cupones de descuento	46
5.6.3	Análisis de información del producto	47

5.6.4	Web Crawler	47
5.7	Aplicación web	48
5.7.1	Registro y autenticación de usuarios	49
5.7.2	Listado de ofertas	50
5.7.3	Detalles de oferta y producto	50
5.7.4	Búsqueda de ofertas	51
5.7.5	Filtrado de ofertas por categoría	51
5.8	Ciclo de desarrollo	52
5.8.1	Sprint 0	52
5.8.2	Sprint 1	54
5.8.3	Sprint 2	54
5.8.4	Sprint 3	55
5.8.5	Sprint 4	56
5.8.6	Sprint 5	57
5.8.7	Sprint 6	59
5.8.8	Sprint 7	59
5.8.9	Sprint 8	60
5.8.10	Sprint 9	61
5.9	Despliegue	61
5.9.1	Amazon Web Services	62
5.9.2	Google Cloud	63
5.9.3	Entrega y despliegue continuo	64
6	Conclusiones	67
6.1	Trabajo futuro	67
A	Glosario de acrónimos	71
B	Glosario de términos	73
	Bibliografía	75

Índice de figuras

1.1	Ejemplo de un canal de Telegram.	2
1.2	Arquitectura del sistema.	3
2.1	Fases de la minería de texto.	6
2.2	Diferencias entre web scraper y web crawler.	9
3.1	Visualizador Kibana para Elasticsearch.	13
3.2	Diagrama de la ejecución de una aplicación SSR.	17
3.3	Vista del editor PhpStorm.	18
3.4	Ejemplo de uso de la aplicación Taiga.	20
3.5	Editor de Overleaf.	22
4.1	Roles de Scrum	25
4.2	Ciclo de vida de Scrum.	26
4.3	Primera <i>release</i> de la API del proyecto en Github.	29
4.4	Diagrama de Gantt de la planificación temporal.	31
5.1	Esquema general de la arquitectura del sistema.	38
5.2	Router de la API.	39
5.3	Ejemplo de llamada a la API para registrar un usuario.	40
5.4	Migración de la tabla <code>users</code>	41
5.5	Diagrama modelo de datos de la API.	42
5.6	Bot de Telegram recopilando ofertas.	43
5.7	Modelo de datos de ofertas en Elasticsearch.	45
5.8	Vista en Kibana del corpus de productos almacenados en Elasticsearch.	48
5.9	Registro y autenticación de usuarios en la aplicación.	50
5.10	Diagrama autenticación con JWT.	51
5.11	Listado de ofertas en la aplicación web.	52

5.12	Detalles de una oferta y su producto.	53
5.13	Búsqueda de ofertas en la aplicación	54
5.14	Filtrado de ofertas de electrónica.	55
5.15	Gráfica de burn-down del proyecto.	56
5.16	Gráfica de burn-down del sprint 0.	56
5.17	Gráfica de burn-down del sprint 1.	57
5.18	Gráfica de burn-down del sprint 2.	57
5.19	Gráfica de burn-down del sprint 3.	58
5.20	Gráfica de burn-down del sprint 4.	58
5.21	Gráfica de burn-down del sprint 5.	59
5.22	Gráfica de burn-down del sprint 6.	59
5.23	Gráfica de burn-down del sprint 7.	60
5.24	Gráfica de burn-down del sprint 8.	60
5.25	Gráfica de burn-down del sprint 9.	61
5.26	Diagrama de despliegue de la aplicación.	62
5.27	Configuración fichero app.yml.	64
5.28	Dashboard de travis-CI.	65

Índice de tablas

4.1	Costes de servidores.	32
4.2	Riesgos del proyecto.	33
5.1	Requisitos funcionales: Product Backlog.	36
5.2	Requisitos no funcionales.	37

Introducción

ACTUALMENTE el consumo de productos a través del comercio electrónico se ha disparado. Han surgido tiendas y empresas que se han posicionado como líderes del mercado como Amazon, Gearbest o Aliexpress. La cantidad de productos a un *click* del ratón disponibles para los usuarios es inmensa, contándose por miles o millones de productos. Tal cantidad de información puede ser abrumadora y llegar a confundir a los usuarios, siendo muy difícil asegurarse de que se está haciendo una buena compra.

En este contexto, han empezado a surgir grupos y canales de usuarios en distintos medios, como páginas web, foros, grupos de WhatsApp o canales de Telegram en los que se comparte información de ofertas que se van encontrando en las distintas tiendas para, de forma colaborativa, ayudar unos usuarios a los otros a conseguir comprar productos con el mejor precio posible. Entre ellos destacan a nivel nacional webs como *Chollometro*¹, que tiene más de un millón de usuarios registrados y factura anualmente más de 800.00 € [1] o *forocupon*² y multitud de canales de Telegram como *[Canal] Chollos*, *Canal Ninja* o *ZONACHOLLOS* (ver Figura 1.1).

1.1 Motivación

A pesar de la multitud de medios a través de los que se difunden ofertas sobre productos de tiendas online, no hay un medio óptimo a través del cual compartir estas ofertas. En las páginas web y foros se puede, de manera sencilla, estructurar la información y permitir una búsqueda eficiente de los productos y ofertas, pero normalmente no son medios demasiado ágiles a la hora de compartir la información con los usuarios.

Por ello, uno de los principales métodos escogidos por miles de personas es el uso de canales en el programa Telegram Messenger que permite, mediante notificaciones *push*, recibir

¹<https://www.chollometro.com/>

²<http://www.forocupon.com>

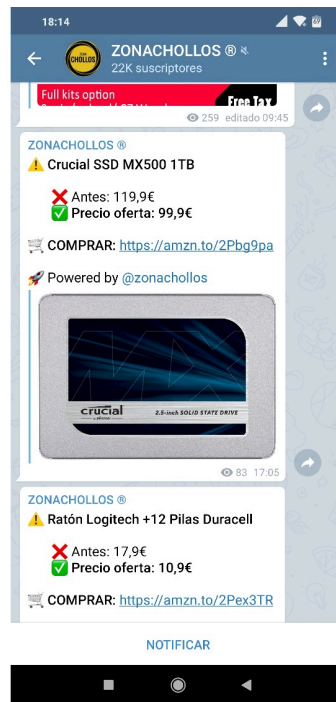


Figura 1.1: Ejemplo de un canal de Telegram.

al instante multitud de ofertas de productos cómodamente en el móvil. El problema de este método es que el medio para mostrar las ofertas no está diseñado expresamente para dicho fin, por lo que la visualización de los datos y búsqueda de ofertas es bastante deficiente.

1.2 Objetivos

El principal objetivo de este proyecto es aunar las fortalezas de los distintos medios para la compartición de ofertas de productos. Extraer de forma automática la ingente cantidad de ofertas compartidas en los canales de Telegram, analizarlas y clasificarlas permitiendo, en un medio más afín como es una aplicación web, la búsqueda y visualización de los datos estructurados de forma útil y coherente. De esta manera el usuario podrá acceder a una gran cantidad de ofertas de forma rápida e intuitiva y seleccionar aquellas en las que esté interesado:

- Recopilación de ofertas de canales de Telegram mediante la creación de un bot capaz de extraer, de forma continua y automática, información de numerosos canales.
- Análisis de las ofertas mediante un servicio implementado usando técnicas de análisis de texto y heurísticas.
- Clasificación de las ofertas en distintas categorías haciendo uso de la tecnología ofrecida

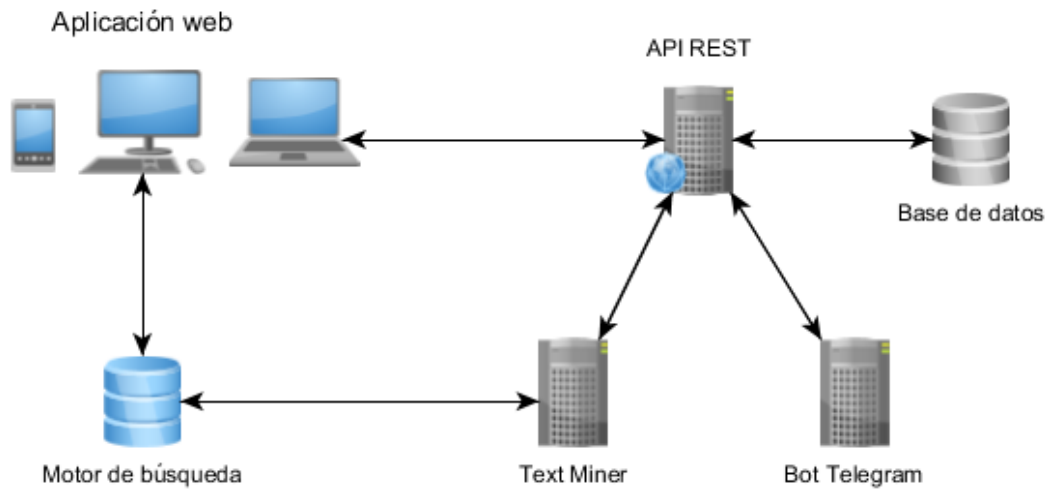


Figura 1.2: Arquitectura del sistema.

por Elasticsearch.

- Creación de una aplicación web para la búsqueda y visualización de las ofertas.

1.3 Visión global del sistema

El proceso de extracción de las ofertas será realizado con la creación de un bot de Telegram hecho en PHP con la librería MadelineProto, capaz de conectarse a distintos canales dentro de dicha aplicación y de recopilar información compartida por los usuarios. Una vez extraídos los datos, se clasificarán mediante técnicas de análisis de texto con Tex Miner, un script en PHP para la obtención de información relevante sobre las ofertas. Así mismo, dichas ofertas serán almacenadas en una base de datos MySQL y un índice de Elasticsearch que serán accedidos a través de una API, que permitirá la visualización y búsqueda de las mismas en una aplicación web.

En la Figura 1.2 puede observarse un esquema básico de la arquitectura del sistema.

1.4 Estructura de la memoria

La memoria del presente proyecto está estructurada del siguiente modo:

- **Introducción.** Explica el contexto en el que se enmarca el proyecto, introduce la problemática a tratar y detalla el alcance y objetivos del mismo desde un punto de vista global. También muestra la estructura de la memoria y el plan de trabajo seguido.

- **Conceptos.** Introducción a la minería de texto y descripción de las distintas técnicas empleadas.
- **Tecnología.** Describe y justifica las principales tecnologías empleadas para desarrollar el proyecto atendiendo a los requisitos del mismo.
- **Proceso de ingeniería.** Detalla el proceso de ingeniería: la metodología, la planificación y la gestión del proyecto.
- **Desarrollo.** Realiza una descripción detallada del análisis, diseño, implementación, pruebas y despliegue del sistema.
- **Conclusiones y trabajo futuro.** Proporciona una evaluación global del producto obtenido así futuras líneas de trabajo que se podrían explotar alrededor del proyecto.
- **Apéndices.** Está compuesto por las siguientes secciones complementarias:
 - **Glosario.** Define los términos y acrónimos técnicos empleados en la memoria del proyecto.
 - **Bibliografía.** Recoge la documentación bibliográfica sobre la que se apoya el proyecto.

1.5 Plan de trabajo

Conforme a la metodología empleada en el proyecto se ha dividido el plan de trabajo en varias etapas:

- Estudio de campo de las aplicaciones y medios más populares empleados por los usuarios para descubrir y compartir ofertas, así como de las principales tiendas online del mundo, comprobando sus funcionalidades y sus capacidades y limitaciones.
- Especificación de los objetivos, así como el análisis de los requisitos funcionales y no funcionales necesarios para la creación del proyecto.
- Análisis y estudio de las distintas tecnologías y herramientas disponibles para la realización de los servicios y componentes que forman el proyecto.
- Diseño de la arquitectura global del sistema de acuerdo a los objetivos y requisitos establecidos anteriormente.
- Diseño e implementación de los distintos servicios que componen el proyecto.
- Elaboración de la memoria del proyecto y toda la documentación necesaria.

Conceptos

EN esta sección se definirán los distintos conceptos sobre análisis y minería de texto que han sido necesarios abordar a lo largo del proyecto para conseguir definir los requisitos del sistema y llevar a cabo las distintas funcionalidades.

2.1 Introducción

Hasta hace poco los investigadores informáticos y especialistas en sistemas de la información se concentraban en la búsqueda de conocimiento a partir de fuentes de datos estructurados, bases de datos numéricas o almacenes de datos [2]. A este proceso se le denomina *data mining* o minería de datos. Sin embargo, el texto es el tipo de información más usado por las personas [3], por lo que gran cantidad de los datos disponibles actualmente son capturados a partir de fuentes de texto no estructurado como páginas web o aplicaciones para móviles u otros dispositivos digitales. El hallazgo de conocimiento a partir de fuentes de datos que contienen texto no estructurado es denominado *text mining* o minería de texto. A menudo la minería de texto es considerada como una rama dentro de la minería de datos.

Dado que para este proyecto los requisitos establecen la necesidad de obtener información principalmente de fuentes en las que los datos se encuentran en forma de texto, ya sea de forma no estructurada como en los mensajes de los canales de Telegram o de manera estructurada o semi-estructurada como en las páginas web, se ha intentado abordar los principales conceptos necesarios para crear las funcionalidades del sistema. Por un lado las técnicas o procesos dentro de la minería de datos de extracción y recopilación de datos como puede ser el Web Scraping, minería de textos o el Web Crawling. Por otro lado, una vez recopilados los datos, el análisis de los mismos mediante técnicas de análisis de texto como *tokenizar* o clasificar el texto.

Minería de texto

La minería de texto conlleva una serie de actividades necesarias para obtener la información de manera eficiente. Estas actividades son:



Figura 2.1: Fases de la minería de texto.

2.2 Minería de texto

La minería de texto es el proceso de exploración y análisis de grandes cantidades de datos encontrados en textos no estructurados, haciendo uso de software capaz de identificar conceptos, temas, patrones o palabras clave. Está relacionado con el análisis de texto. En este caso se refiere al uso de técnicas de minería de texto para clasificar grandes cantidades de datos y algoritmos que puedan analizar conjuntos masivos de datos no estructurados. La minería y el análisis de texto nos ayudan a encontrar información valiosa en fuentes de datos basadas en texto como pueden ser: publicaciones en redes sociales, correos electrónicos, páginas web o documentos de texto. La minería de textos es similar a la minería de datos, pero enfocándose más en el texto en lugar de formas de datos más estructuradas. Sin embargo, uno de los primeros pasos en el proceso de minería de datos es la organización y estructuración de datos para que puedan ser sometidos a análisis, tanto cualitativos como cuantitativos. En este proceso está implicado el uso de tecnologías de procesamiento de lenguaje natural (NLP) que aplica principios de lingüística computacional para analizar e interpretar un conjunto de datos.

Como se puede observar en la Figura 2.1 en las aplicaciones de minería de texto el trabajo del analista o del científico de datos, consiste en realizar las siguientes actividades:

- **Extracción:** recolectar datos de distintos recursos, como páginas web, correos electrónicos, mensajes de usuarios o ficheros de texto. Dependiendo de la aplicación, este proceso puede ser completamente automático o ser guiado por analista.

- **Pre-procesamiento:** identificación de contenido o extracción de características representativas. Por un lado está la limpieza del texto o *text clean-up* en la que las partes no necesarias o la información no requerida como los anuncios en una página web son descartados. Por otro lado está el proceso de *tokenización* mediante el cual el texto es dividido en entidades con significado como palabras o frases dependiendo de la presencia de espacios o signos de puntuación. Una vez realizadas ambas técnicas se puede proceder a obtener conjuntos de datos cuantitativos como la frecuencia de palabras en el texto, tipos de palabra o información sintáctica. Estos elementos podrán ser usados en análisis posteriores.
- **Indexación:** creación de índices con los términos, su localización en el texto y número de apariciones. Esto permitirá un acceso rápido a la estructura de los datos procesados.
- **Proceso de minado:** llegados a este punto, el texto ha sido pre-procesado adecuadamente y es posible *minarlo*. Para ello, se pueden aplicar distintas técnicas de exploración para extraer nuevo conocimiento. Por ejemplo, podría identificarse la ocurrencia de diferentes términos, enlazarlos con un diccionario para su desambiguación e identificar las relaciones entre ellos.
- **Análisis:** el proceso de minado produce resultados sin tratar. Dichos resultados deben ser visualizados y evaluados de manera que puedan ser interpretados conforme a las cuestiones que el científico de datos quiera investigar.

2.3 Análisis de texto

El análisis de texto es un término muy amplio que abarca múltiples procesos mediante los cuales el texto y los documentos con lenguaje natural pueden ser modificados de modo que puedan ser organizados y descritos [4]. Su objetivo es analizar el lenguaje en lugar de los datos convenientemente normalizados en un almacén de datos. Por ejemplo, antes de que el software pueda realizar el reconocimiento de patrones, debe extraer el significado lingüístico de los datos del texto. En otras palabras, se debe leer el texto antes de que pueda ser analizado. A menudo el análisis de texto es considerado como una parte del proceso de minería de texto o incluso como un sinónimo de minería de texto, siendo descrito como un abanico de técnicas para la búsqueda y extracción de información útil de colecciones de documentos a través de la identificación y exploración de patrones en datos no estructurados como libros, páginas web, correos electrónicos o descripciones de productos [5].

Algunas de las técnicas más básicas utilizadas para el análisis de texto son:

- **Frecuencia de palabras:** crear listas de palabras y analizar su frecuencia en el texto.

- **Colocación:** analizar las palabras que aparecen comúnmente unas cerca de las otras.
- **Concordancia:** analizar las instancias o contextos en los que aparecen una palabra o un conjunto de palabras.
- **N-gramas:** el estudio de los n-gramas. Un n-grama es una subsecuencia de n elementos de una secuencia dada como por ejemplo fonemas, sílabas, letras o palabras.
- **Reconocimiento de entidades nombradas:** es una tarea de extracción de información que busca localizar y clasificar en categorías predefinidas, como personas, organizaciones, lugares, expresiones de tiempo y cantidades, las entidades nombradas encontradas en un texto.

Además de las anteriores, es común la utilización de técnicas más complejas como la clasificación de documentos, la comparación de corpus lingüísticos o la detección de *clústeres*.

2.4 Web scraping

A veces es necesario extraer información de páginas web que están hechas para lectores humanos y no agentes de software. A este proceso se le denomina *web scraping* [6]. Aunque no todos los datos que se obtienen de dicho medio son textos, ya que también hay imágenes, vídeos u otros medios de información, sí forman la mayor parte de los datos disponibles. Los distintos métodos para realizar web scraping han ido evolucionando junto la *World Wide Web*. Algunos de los más importantes son:

- **Scraping manual:** es una opción válida cuando el tamaño de los datos es mínimo, cuando extraer los datos no requiere tareas repetitivas, cuando el proceso de automatizado llevaría más tiempo que la extracción de datos en sí misma o cuando haya medidas de seguridad implementadas que no permitan el uso de métodos automatizados.
- **HTML Parsing:** normalmente las páginas web no muestran los datos en formatos amigables como los ficheros *.csv* o *.json*. Las páginas HTML son generadas por el servidor como respuesta a las peticiones de los usuarios. El análisis de la estructura HTML de una web puede mostrar elementos estáticos y repetitivos de los que se pueda extraer datos relevantes mediante el uso de un lenguaje de programación o herramienta de web *scraping*.
- **DOM Parsing:** el *parseo* del *Document Object Model* (DOM) es una evolución del *HTML Parsing* basado en el desarrollo de los lenguajes y navegadores que lo han implementado. El DOM es usado de manera muy común en las hojas de estilos (CSS) y en

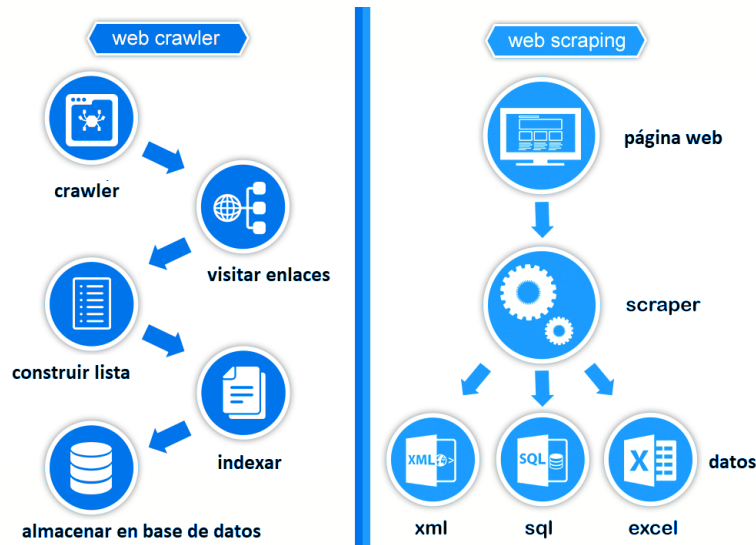


Figura 2.2: Diferencias entre web scraper y web crawler.

JavaScript. La integración del DOM revela nuevas posibilidades para gestionar determinadas partes de una página web y poder extraer datos de forma más rápida y sencilla.

- **XPath:** como su nombre sugiere, es usado en ficheros XML. Requiere una estructura de página web más precisa que el DOM y tiene las mismas posibilidades a la hora de gestionar los segmentos que la conforman.
- **APIs:** mientras que los métodos anteriores están diseñados para extraer información de interfaces para humanos, una *Application Programm-ing Interface*(API) está diseñada para comunicarse con una aplicación. Por ello, las APIs son usadas a menudo para describir directorios con enlaces relevantes a contenido web. Algunas páginas web proveen a sus usuarios de interfaces de este tipo para que la extracción de información automatizada sea más sencilla.

2.5 Web crawling

Un *web crawler* (también conocido como robot web o araña) es un programa usado para descargar páginas web. Dada una lista de semillas *Uniform Resource Locators* (URLs), el *crawler* extrae una URL de la lista, descarga la página correspondiente, extrae todas las URLs que contiene y añade todas las que sean desconocidas a la lista. Aunque el algoritmo de *web crawling* es conceptualmente sencillo, diseñar un *crawler* altamente eficiente comparable a los usados por los grandes motores de búsqueda es una tarea compleja [7]. Un *web crawler* puede usarse en conjunción con un *web scraper* para extraer datos de una página web, pero en realidad

son dos actividades diferenciadas (ver Figura 2.2). Existen varias técnicas de *crawling*, las más usadas son [8]:

- **Crawler de propósito general:** recolecta tantas páginas como puede a partir de URLs y sus enlaces. De esta manera el *crawler* es capaz de extraer un gran número de páginas de distintas localizaciones. Este tipo de *crawler* puede llegar a ralentizar la velocidad y ancho de banda de una web dado que está accediendo a todas sus páginas.
- **Crawler enfocado:** está diseñado para recolectar documentos solo de un tema en específico, por lo que puede reducir considerablemente el tráfico de red y las descargas realizadas en el proceso. Su objetivo es buscar selectivamente páginas que cumplen una serie de requisitos.
- **Crawler distribuido:** este tipo de *crawler* se usan múltiples procesos para descargar páginas de una web.

Capítulo 3

Tecnología

LA selección de la tecnología adecuada es una condición necesaria, aunque no suficiente, para llevar a cabo un proyecto con éxito. En este capítulo se detallarán las elecciones tecnológicas y se justificarán debidamente. Este capítulo contiene una sección para explicar las tecnologías de cada uno de los componentes. Adicionalmente, se incluye un apartado donde se describen las herramientas utilizadas para dar soporte a la gestión y desarrollo del proyecto. Con estas tecnologías se ha llevado a cabo la construcción de los distintos componentes del sistema siguiendo los pasos que comentaremos en el próximo capítulo.

3.1 Extracción de datos

En esta sección se describen las distintas tecnologías usadas en la capa lógica de extracción de datos.

3.1.1 MadelineProto

MadelineProto¹ es una librería escrita en el lenguaje de programación PHP a través de la cual es posible interactuar con la aplicación de mensajería Telegram Messenger.

Actualmente Telegram permite el uso de dos tipos distintos de API. Por un lado está la *Bot API* que permite crear bots con capacidad de conectarse al sistema de Telegram [9]. Los bots de Telegram son cuentas especiales que no necesitan estar ligadas a un número de teléfono para ser creados.

Por otro lado está la *Telegram API* que permite la creación de clientes de Telegram totalmente funcionales similares a las versiones para móviles oficiales o a la aplicación web.

La gran ventaja de MadelineProto es que permite crear bots en Telegram con la API para clientes en vez de con la API de bots, lo cual permite al desarrollador superar ciertas limitaciones impuestas a los bots como por ejemplo la incapacidad de unirse a canales de Telegram

¹<https://docs.madelineproto.xyz>

de los cuales no es administrador. Por lo tanto, gracias a esta librería es posible crear un bot capaz de unirse a multitud de canales de Telegram y leer los mensajes ahí compartidos.

3.1.2 Goutte

Goutte² es una librería en PHP de *screen scraping* y *web crawling*. Provee una API para extraer datos de páginas web de forma sencilla, permitiendo acceder a los distintos nodos del árbol DOM. A su vez, Goutte utiliza internamente varios componentes del framework PHP Symfony: BrowserKit, CssSelector y DomCrawler, así como del cliente HTTP Guzzle.

3.2 Persistencia de datos

En esta sección se describen las tecnologías utilizadas en la capa lógica de persistencia de datos.

3.2.1 MySQL

MySQL³ es un sistema de gestión de bases de datos relacional. Es usado en 9 de cada 10 de las mayores páginas web del mundo [10]. Está diseñado y optimizado para aplicaciones web. Algunas de sus principales características son:

- Uso de procesos multi-hilo mediante hilos del kernel.
- Sistemas de almacenamiento transaccionales y no transaccionales.
- Licencia dual GPL o para uso comercial.
- Tablas en disco B-tree MyISAM con compresión de índice.
- APIs para acceder a base de datos en multitud de lenguajes, entre ellos PHP.
- Tablas hash temporales en memoria.

El uso de una base de datos relacional como MySQL permite una mejor y más rápida integración con frameworks de PHP como Laravel con respecto a otras bases de datos relacionales como PostgreSQL o bases de datos NoSQL.

²<https://github.com/FriendsOfPHP/Goutte>

³<https://dev.mysql.com>

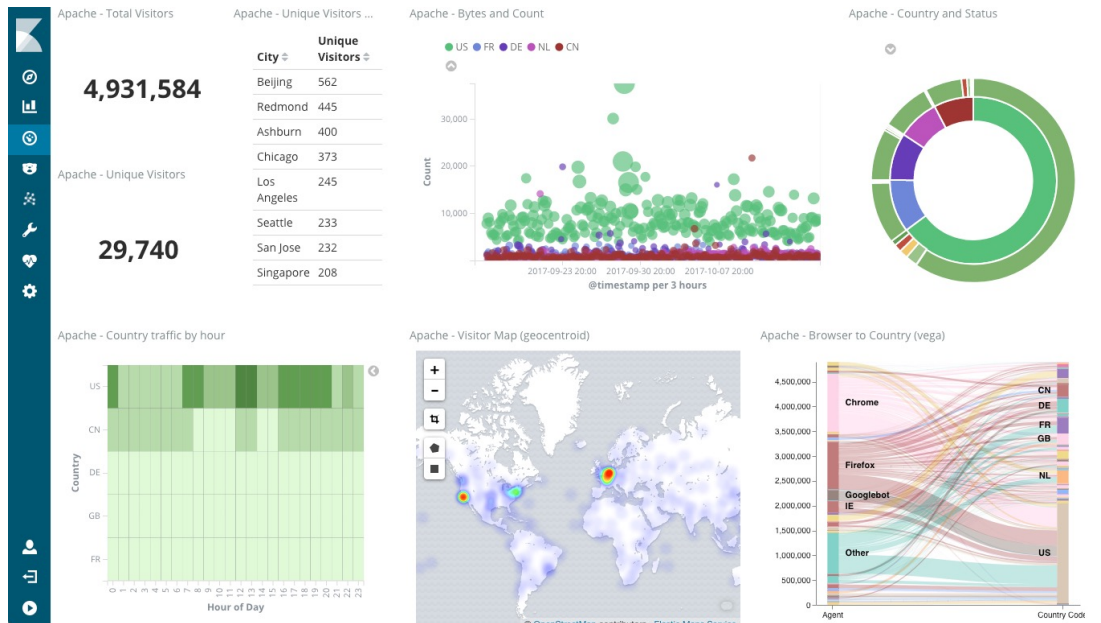


Figura 3.1: Visualizador Kibana para Elasticsearch.

3.2.2 Elasticsearch

Elasticsearch⁴ es un motor de búsqueda y análisis basado en Apache Lucene, distribuido y con capacidad de tenencia múltiple, a través de una API REST y documentos JSON. Sus principales características son:

- **Rapidez:** hace uso de índices invertidos distribuidos. Es capaz de encontrar coincidencias en búsquedas de texto completo con gran velocidad incluso en conjuntos de datos muy grandes.
- **Facilidad:** ofrece una interfaz muy sencilla a través de una API HTTP.
- **Integración:** Elasticsearch provee de integración con Kibana para la visualización de datos (ver Figura 3.1) y de Logstash para la carga y transformación de datos.
- **Orientado a documentos:** almacena entidades complejas como documentos JSON estructurados e indexa todos los campos de forma predeterminada.

La principal ventaja de Elasticsearch es la capacidad de realizar búsquedas de forma muy sencilla utilizando filtros, funciones y consultas personalizadas, dotando al desarrollador de una capacidad inmensa para la obtención de información relevante a partir de una colección de documentos.

⁴<https://www.elastic.co/guide/en/elasticsearch/reference/master/index.html>

3.3 Acceso a datos

En esta sección se describen las tecnologías utilizadas en la capa lógica de acceso de datos.

3.3.1 Laravel Lumen

Lumen⁵ es un micro-framework PHP basado en Laravel y optimizado para el desarrollo de APIs. Permite utilizar todas las funcionalidades de Laravel, como el ORM Eloquent, fachadas, cacheo, colas, validación, rutas, o middlewares sin apenas configuración.

Está pensado para proyectos y componentes que se pueden beneficiar de las características y funcionalidades de Laravel sacrificando las posibilidades de configuración y flexibilidad en aras de la rapidez y la eficiencia.

jwt-auth

El paquete `jwt-auth`⁶ para la autenticación con JSON Web Tokens en Laravel y Lumen permite la generación de JSON WEB TOKENS en la API de forma sencilla. Además permite usar un middleware para la comprobación de la autenticación en las rutas de la API y validar las peticiones comprobando el token recibido. El uso de JWT permite crear un mecanismo de autenticación sin estado (en inglés, *stateless*), sin estado, ya que la sesión del usuario nunca se guarda en el backend de la aplicación, por lo que es muy útil para la creación de APIs RESTful. El token generado es único y normalmente tiene una caducidad, siendo almacenado en el cliente y enviado a la API en cada petición que requiera ser autenticada.

3.3.2 PHP libcurl

PHP soporta `libcurl`⁷, una biblioteca que permite conectarse y comunicarse con diferentes tipos de servidores y diferentes tipos de protocolos. Actualmente, `libcurl` admite los protocolos `http`, `https`, `ftp`, `gopher`, `telnet`, `dict`, `file` y `ldap`. La biblioteca `libcurl` también admite certificados `HTTPS`, `HTTP`, `POST`, `HTTP PUT`, subidas mediante `FTP` (también se puede hacer con la extensión `FTP` de PHP), subidas basadas en formularios `HTTP`, proxies, cookies, y autenticación usuario+contraseña [11].

3.4 Análisis de datos

En esta sección se describen las tecnologías utilizadas en la capa lógica de análisis de datos.

⁵<https://lumen.laravel.com/docs/5.8>

⁶<https://jwt-auth.readthedocs.io/en/develop/>

⁷<http://php.net/manual/es/book.curl.php>

3.4.1 PHP NlpTools

NlpTools⁸ es una biblioteca para el procesamiento de lenguaje natural escrita en PHP. Entre sus funcionalidades están la capacidad para realizar clasificación de texto, *clustering*, *tokenizing*, *stemming* etc.

3.5 Visualización de datos

En esta sección se describen las tecnologías utilizadas en la capa lógica de visualización de datos.

3.5.1 Angular

Angular⁹ es un framework de desarrollo para el lenguaje de programación Javascript y creado por la compañía Google. Está diseñado para la creación de aplicaciones web SPA, que destacan por mostrar en una sola página todo el contenido de la web sin necesidad de recargar el navegador. Uno de los principales objetivos de este tipo de aplicaciones es mejorar la experiencia de usuario así como disminuir los tiempos de respuesta y latencia en la carga de los distintos componentes de la aplicación.

Este tipo de aplicación intenta emular el comportamiento de las aplicaciones nativas de cualquier sistema operativo aunado a las ventajas del entorno de desarrollo de las páginas web.

El lenguaje principal de este framework es TypeScript, un conjunto de JavaScript y ECMAScript, que resume y facilita el desarrollo, aunque es posible usar Angular sin TypeScript, su utilización está enormemente recomendada ya no sólo por las ventajas que aporta sino porque el código y los ejemplos de la documentación oficial están escritos usando dicho lenguaje.

Las principales características de Angular son:

- Uso de componentes: Los bloques básicos para construir las aplicaciones en Angular son los *NgModules* que proveen un contexto para la creación de componentes. Dichos componentes definen las vistas y usan los servicios que hacen funcionar la aplicación y los engloban en unidades lógicas.
- Two-way data binding: Angular soporta two-way data binding, eso quiere decir que cualquier cambio en el árbol DOM, como por ejemplo por una elección del usuario, es reflejada también en los datos del modelo y, al revés, un cambio en los datos del modelo es automáticamente propagado a la interfaz.

⁸<http://php-nlp-tools.com>

⁹<https://angular.io/docs>

- Inyección de dependencias: es un patrón de diseño muy importante que permite a una clase obtener las dependencias de recursos externos en vez de crearlas ella misma.
- Rutas: permite mapear URLs a vistas en vez de páginas, apoyándose en la filosofía SPA sin abandonar las bondades de las páginas web convencionales.

Una de las principales ventajas del uso de Angular sobre otros frameworks en el mercado como React, es su capacidad para brindar desde el núcleo del framework la mayor parte de funcionalidades necesarias para abordar este proyecto sin tener que usar demasiadas bibliotecas de terceros. Por otro lado, a la hora de implementar determinadas funcionalidades, Angular es mucho más estricto que React, siguiendo por defecto algunos patrones de diseño como la inyección de dependencias, por lo que es más adecuado para el uso por principiantes. Además tiene el respaldo de una comunidad mayor que otros frameworks competidores como Vue.js.

Angular Universal

Angular Universal¹⁰ es un componente que permite renderizar una aplicación en el servidor. Comenzó como un componente independiente, y ahora forma parte del núcleo del *framework*.

Una aplicación normal en Angular se ejecuta en el navegador, *renderizando* páginas en el DOM como respuesta a las acciones llevadas a cabo por los usuarios. Angular Universal genera páginas estáticas en el servidor a través de un proceso llamado *server side rendering* (SSR), mediante el cual el navegador recibe una versión estática de la aplicación, ya *precompilada*, que permite la ejecución de la aplicación de forma más rápida mientras es interpretada y servida la versión de cliente (ver Figura 3.2).

Las técnicas de SSR tienen la ventaja de facilitar la indexación de contenido para los *web crawlers*, así como aumentar la velocidad de carga inicial de las aplicaciones web SPA.

Angular Material

Angular Material¹¹ es un conjunto de componentes creados a partir de las especificaciones de Material Design de Google para la interfaz de las aplicaciones hechas con Angular.

Esta *suite* de componentes permite construir aplicaciones web de forma sencilla teniendo a tu disposición los principales elementos necesarios en la mayoría de páginas como son los controles de formularios, botones, *pop-ups* o menús de navegación entre otros, todos ellos ya configurados y *testeados* para la mayoría de navegadores disponibles.

¹⁰<https://angular.io/guide/universal>

¹¹<https://material.angular.io/>

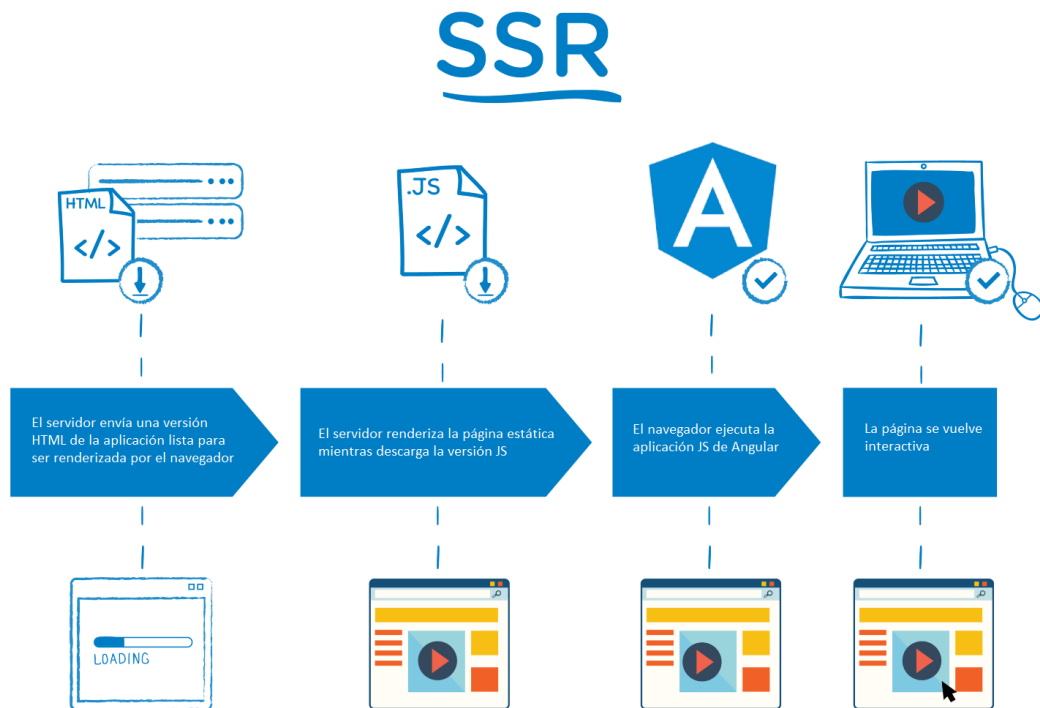


Figura 3.2: Diagrama de la ejecución de una aplicación SSR.

Angular flex-layout

Angular Flex Layout¹² provee una sofisticada API para el uso de Flexbox CSS y media queries. El motor de Flex Layout automatiza el proceso de aplicar las propiedades de Flexbox CSS a los distintos elementos de la vista del navegador de forma apropiada. Esta automatización soluciona muchos de los problemas y complejidades que se producen a la hora de implementar CSS de manera manual.

Además, uno de sus grandes beneficios es el motor de *responsividad*, que permite a los desarrolladores especificar los distintos *layouts*, tamaños, y configuraciones para cada resolución y tamaño de pantalla.

Angular-jwt

Angular-jwt¹³ es una biblioteca de Angular que permite interceptar las peticiones HTTP de una aplicación y añadirles de forma automática JSON Web Tokens.

¹²<https://github.com/angular/flex-layout>

¹³<https://github.com/auth0/angular2-jwt>

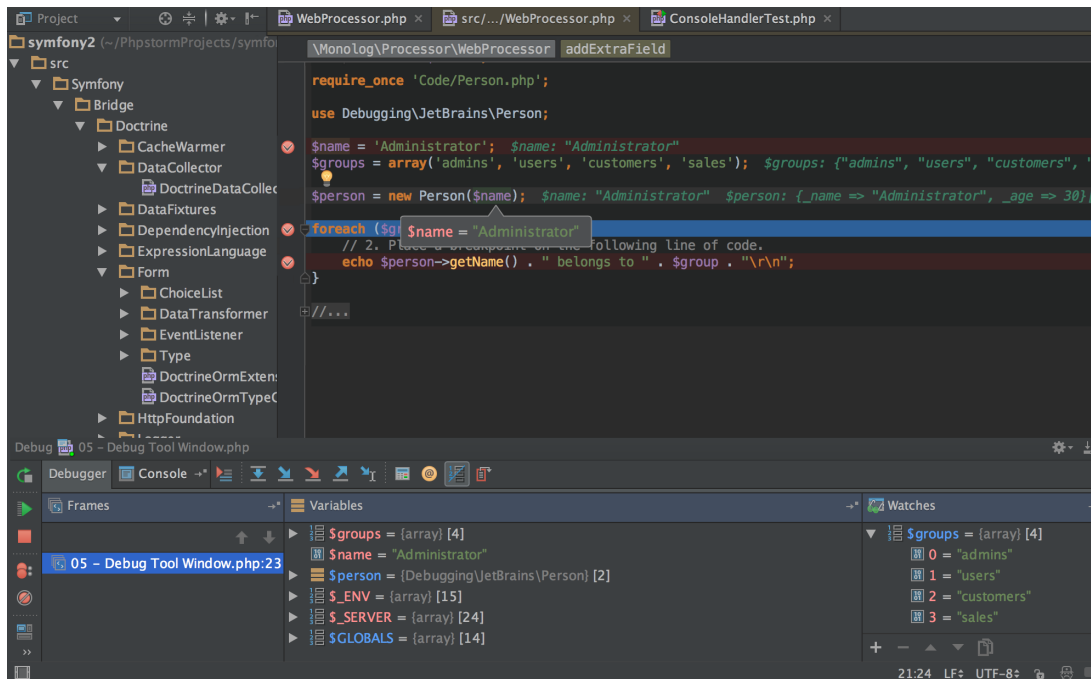


Figura 3.3: Vista del editor PhpStorm.

Sass

Sass¹⁴ es un preprocesador de CSS que permite el uso de elementos dinámicos como la posibilidad de utilizar variables, encadenar selectores, importación de ficheros parciales, *mixins* para agrupar selectores de CSS o operadores matemáticos.

3.6 Herramientas de desarrollo

En esta sección se describen las herramientas de desarrollo empleadas en la creación de este proyecto.

3.6.1 PhpStorm

PhpStorm¹⁵ es un IDE comercial y multi-plataforma para PHP desarrollado por JetBrains (ver Figura 3.3).

Las características principales de PHPStorm son:

- Asistencia de código inteligente: realiza inspecciones para verificar que el código es correcto y analizar el proyecto en su conjunto.

¹⁴<https://sass-lang.com>

¹⁵<https://www.jetbrains.com/phpstorm>

- Navegación de código inteligente: el IDE entiende a dónde queremos ir y nos lleva allí instantáneamente.
- Refactorización rápida y segura: podemos *refactorizar* nuestro código de manera segura para renombrar, mover, eliminar, extraer un método, etc.
- Fácil *debugging* y *testing*.
- Control de versiones con Git, SVN, Mercurial, Bases de Datos y SQL con soporte para Vagrant, Docker y Composer entre otros.

3.6.2 Git

Git¹⁶ es un software de control de versiones creado por Linus Torvalds. Su propósito es llevar registro de los cambios en archivos de software y coordinar el trabajo que varias personas puedan realizar sobre archivos compartidos. Es superior a otras herramientas de SCM como Subversion, CVS o Perforce gracias a características como un sencillo manejo de ramas locales, el uso de áreas de *staging* o la posibilidad de utilizar múltiples flujos de desarrollo.

3.6.3 Taiga

Taiga¹⁷ es una plataforma de gestión de proyectos para desarrollo ágil que permite controlar el progreso de un proyecto (ver Figura 3.4). Provee plantillas de Kanban o Scrum así como interfaces de conexión con repositorios de control de versiones como Github¹⁸ o Bitbucket¹⁹. Consta de una interfaz de usuario muy sencilla y fácil de usar que permite gestionar un proyecto de forma rápida y eficiente.

3.6.4 Npm

Npm²⁰ es la herramienta por defecto para la gestión de paquetes de node.js. Se ejecuta desde la línea de comandos y maneja las dependencias para una aplicación. Sus principales características son:

- Descarga de dependencias de repositorios y la posibilidad de instalarlas en distintos proyectos.
- Versionado: es posible instalar o actualizar dependencias para cualquier versión de un paquete de software en concreto.

¹⁶<https://git-scm.com>

¹⁷<https://taiga.io>

¹⁸<https://github.com>

¹⁹<https://bitbucket.org/product>

²⁰<https://www.npmjs.com>

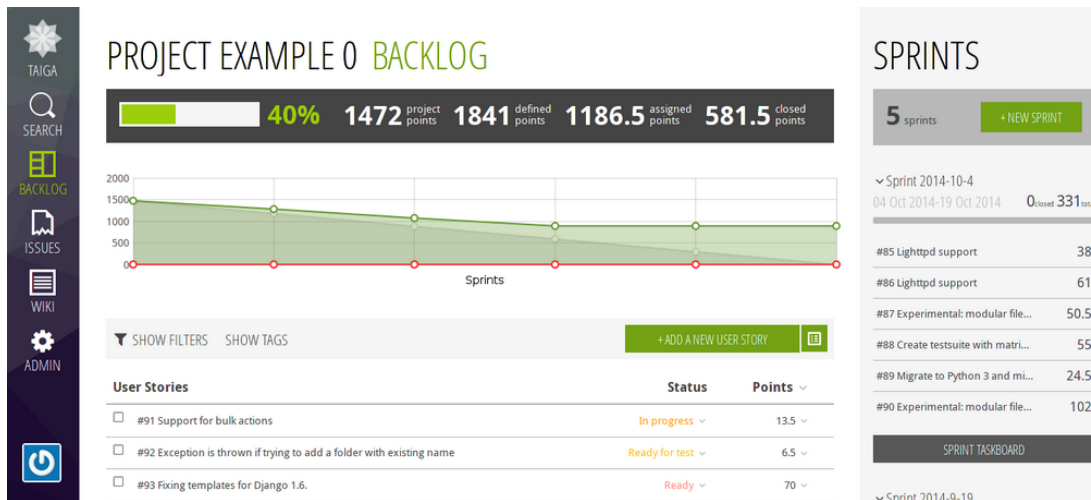


Figura 3.4: Ejemplo de uso de la aplicación Taiga.

- Tareas: los ficheros package.json permiten especificar tareas de línea de comandos.

3.6.5 Composer

Composer²¹ es el equivalente a Npm para el lenguaje de programación PHP. Es un gestor de paquetes a nivel de aplicación que provee un formato estándar para el manejo de dependencias de un software PHP así como las bibliotecas requeridas por el mismo. Funciona desde línea de comandos y permite instalar aplicaciones PHP que estén disponibles en Packagist²², que es su principal repositorio. Además proporciona capacidades de carga automática para bibliotecas haciendo más sencillo el uso de código de terceros.

3.6.6 Vagrant

Vagrant²³ es una herramienta para la creación y configuración de entornos de desarrollo virtualizados. Permite tener un flujo de trabajo sencillo independientemente del rol del usuario: desarrollador, operador o diseñador. Hace uso de un fichero de configuración declarativa que permite describir todos los requisitos de software, paquetes, configuración del sistema operativo o usuarios. Su objetivo es intentar emular entornos de producción ofreciendo el mismo sistema operativo y configuraciones a la vez que permitiendo a los usuarios hacer uso de usos editores, IDEs o navegadores favoritos. Además Vagrant se puede integrar con herramientas de control de configuración com Chef, Puppet, Ansible o Salt, de modo que sea posible

²¹<https://getcomposer.org>

²²<https://packagist.com/>

²³<https://www.vagrantup.com>

usar los mismos scrips para configurar Vagrant y los distintos entornos de producción. Está creado para funcionar tanto en Mac como en Linux o Windows.

3.6.7 Oracle VM Virtual Box

Oracle VM VirtualBox²⁴ es un software de virtualización para arquitecturas x86/amd64. Algunas de las principales características de VirtuakBox son:

- Portabilidad: VirtualBox funciona en un gran número de sistemas operativos de 32 y 64 bits.
- No requiere virtualización por hardware. A diferencia de otras soluciones de virtualización se puede usar VirtualBox en hardware antiguo sin perder funcionalidades como Intel VT-x o AMD-V.
- Guest Additions: permite el uso de paquetes de software instalables en los sistemas virtualizados para mejorar el rendimiento y proveer integración adicional y comunicaciones con el sistema anfitrión. De forma automática permite el ajuste de resoluciones de vídeo, aceleración de gráficos 3D o el uso de carpetas compartidas.
- Generación múltiple de imágenes de la maquina virtual. Permite ir atrás en el tiempo y revertir los cambios en la máquina ofreciendo un árbol de imágenes de su estado.
- Acceso remoto a la máquina virtual de gran rendimiento. Soporta el protocolo Remote Desktop Protocol (RDP) de Windows.

3.6.8 Travis-CI

Travis-CI²⁵ es un servicio de integración continua en la nube, usado para construir y probar proyectos de software alojados en GitHub. Permite dar soporte al proceso de desarrollo mediante la construcción y prueba automática de los cambios en el código del proyecto brindando una respuesta inmediata del éxito de los mismos. Además puede automatizar otras partes del proceso de desarrollo como el manejo de despliegues y notificaciones.

3.6.9 Overleaf

Overleaf²⁶ es una herramienta de escritura colaborativa en línea. Permite hacer todo el proceso de escritura, edición y publicación de documentos científicos mucho más rápido y sencillo. Facilita un editor de uso sencillo de L^AT_EX (ver Figura 3.5) así como la compilación en tiempo real del documento y su visualización.

²⁴<https://www.virtualbox.org>

²⁵<https://travis-ci.com>

²⁶<https://overleaf.com>

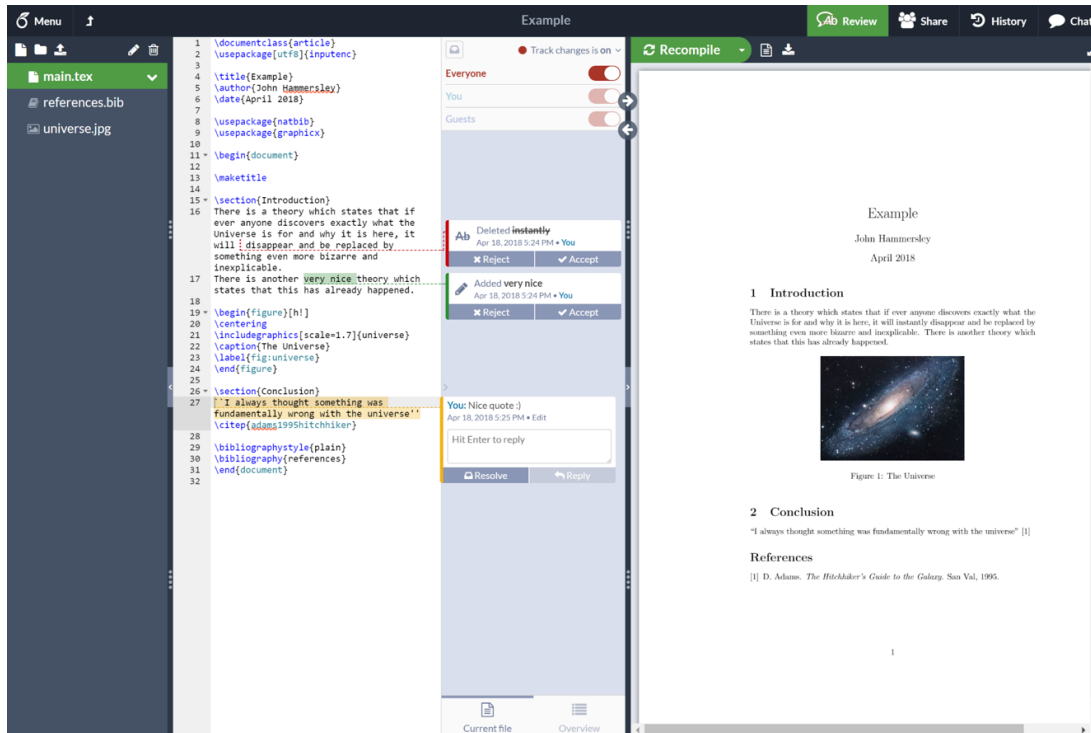


Figura 3.5: Editor de Overleaf.

3.6.10 Postman

Postman²⁷ es una herramienta que permite realizar peticiones HTTP a cualquier API. Es muy útil a la hora de programar y hacer pruebas, puesto que ofrece la posibilidad de comprobar el correcto funcionamiento de los desarrollos. Permite usar un conjunto de funcionalidades que nos ayudarán a organizar las peticiones en colecciones, hacer y automatizar pruebas, mantener equipos sincronizados y crear Mocks de APIs.

²⁷<https://www.getpostman.com>

Proceso de ingeniería

EN este capítulo se justifica y se describe la metodología de desarrollo sobre la que se apoya el proceso de ingeniería. En el caso del presente proyecto se ha optado por emplear una aproximación ágil de desarrollo con Scrum. Se estudiarán también el resto de aspectos relacionados con el proceso de ingeniería como son la organización, la estimación de costes y duración, la gestión de proyecto, la planificación y la gestión de riesgos.

4.1 Metodología

Las metodologías de desarrollo en el ámbito de la ingeniería del software establecen una guía para la estructuración, planificación y control integral del proceso de desarrollo llevado a cabo en los sistemas de información. El uso de una metodología de desarrollo en un proyecto permite obtener ciertas ventajas como la mejora de la productividad y la calidad del producto final.

Dadas las circunstancias especiales en la creación de este proyecto se ha optado por el uso de una versión *personalizada* de Scrum, ya que ni los recursos disponibles, ni el esfuerzo, tiempo o coste destinados al mismo, son los típicos para los que fue pensado en un primer momento Scrum.

4.1.1 Scrum

Scrum¹ es una metodología ágil que es fácilmente adaptable y permite el desarrollo del producto mediante iteraciones incrementales (sprints)². En cada Sprint se realiza análisis, diseño, implementación y pruebas, pudiendo introducir cambios para los siguientes sprints y produciendo entregas parciales o regulares del producto final.

¹<https://www.scrum.org>

²Iteración de Scrum

Al inicio de cada Sprint se deciden las tareas a realizar en dicho ciclo (Sprint Backlog) en función de los resultados del ciclo anterior y del conjunto de tareas pendientes del proyecto, definidas en el Product Backlog.

Roles

Dentro del marco de trabajo de Scrum se definen tres roles [12]. Cada uno de estos roles tiene definidos un conjunto de responsabilidades que deberán ser llevadas a cabo mediante la interacción de todos los implicados para la óptima consecución final del producto.

- **Scrum Team:** todo el trabajo entregado al cliente es desarrollado por un equipo dedicado de Scrum. Se trata de un conjunto de individuos que trabajan juntos para entregar los incrementos de producto solicitados.
- **Scrum Master:** es el encargado de garantizar que todo el proceso de desarrollo del producto se lleva a cabo siguiendo las directrices de Scrum. Interactúa tanto con el Product Owner como con el Scrum Team para asegurar una comunicación correcta entre ambas partes.
- **Scrum Product Owner:** suele ser el cliente, aunque a veces es un representante directo de los usuarios del producto. Es quien decide las funcionalidades y características del producto.

Artefactos

Scrum propone una serie de herramientas y artefactos pensados para la optimización de la organización del proyecto facilitando la transparencia de todo el proceso:

- **Product Backlog:** es una lista o agenda única, formada por todo lo que es necesario hacer en un proyecto. Recoge todos los requisitos y necesidades que el cliente solicita de manera dinámica, pudiendo crecer y evolucionar a lo largo del tiempo. El Product Owner es el responsable de la misma, manteniéndola actualizada y realizando la priorización de tareas a lo largo del proyecto.
- **Sprint Backlog:** es el conjunto de tareas que se van a realizar a lo largo de un determinado Sprint. Estas tareas son seleccionadas durante el Sprint Planning y normalmente se les asigna una duración en base al esfuerzo requerido para realizarlas o a su complejidad, no pudiendo exceder nunca la duración del Sprint. Cada tarea debería poder realizarse en un único Sprint y son asignadas directamente por los miembros del equipo de desarrollo.

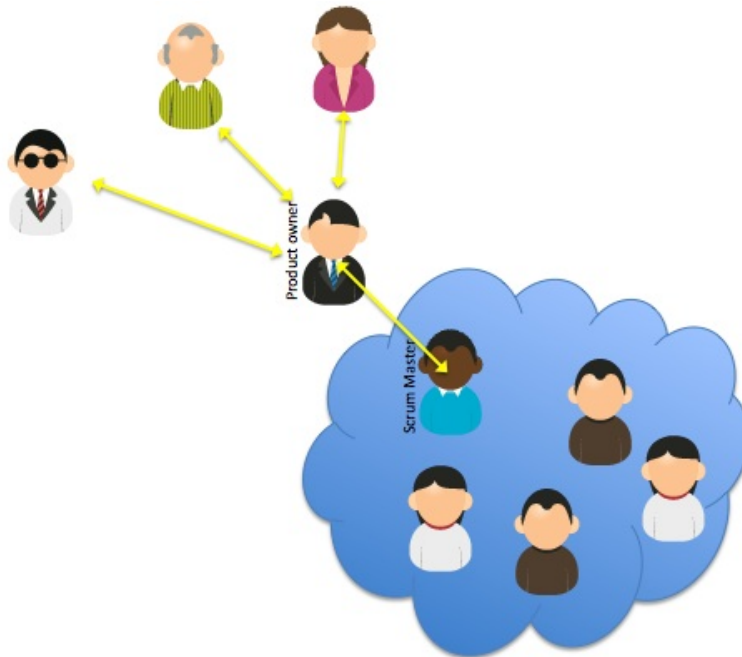


Figura 4.1: Roles de Scrum

- **Burn-Down Chart:** es una gráfica que muestra el trabajo pendiente al comienzo de cada Sprint. Es muy útil para visualizar la velocidad a la que se completan los distintos objetivos del proyecto. Idealmente la línea de la gráfica irá descendiendo hasta llegar a 0 indicando así la finalización del proyecto.
- **Entregable:** es una suma de todas las tareas del Product Backlog completadas durante el Sprint, más los incrementos de todos los sprints anteriores. Al final de cada Sprint el entregable debe ser un producto funcional, independientemente de si el Product Owner decide usarlo o no.

Reuniones

La comunicación es un pilar clave en la consecución exitosa de cualquier proyecto. Por ello, Scrum establece una serie de reuniones periódicas necesarias en distintos puntos del ciclo de vida:

- **Sprint Planning Meeting:** el equipo lleva a cabo un reunión al comienzo de cada Sprint para definir la Agenda del Sprint, es decir, el conjunto de tareas planificadas para ser realizadas en ese Sprint. Estas tareas se extraen del Product Backlog.
- **Daily Scrum Meeting:** reunión diaria de unos 15 minutos en la que participan el equipo de desarrollo y el Scrum Master para determinar el estado del proyecto, identificando

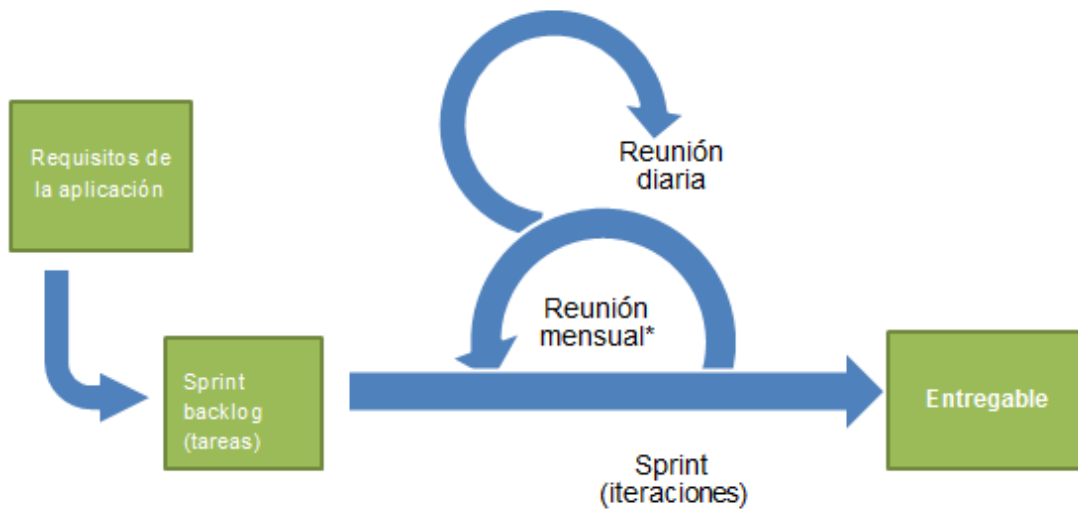


Figura 4.2: Ciclo de vida de Scrum.

los problemas, necesidades o impedimentos puedan surgir.

En esta reunión cada miembro del equipo debe responder a las siguientes preguntas:

- ¿Qué hiciste ayer?
- ¿Qué vas a hacer hoy?
- ¿Hay obstáculos en tu camino?

- **Sprint Review:** al final de cada Sprint se deben realizar dos reuniones. Una de ellas es el Sprint Review, en la que se muestran las tareas realizadas a lo largo del Sprint, normalmente suele hacerse mediante una demo en la que, de forma más visual, pueden presentarse a los distintos *stakeholders* los avances del proyecto. Esta reunión no debe tener una duración mayor a las cuatro horas para un sprint de un mes.
- **Sprint Retrospective:** es la otra reunión que se realiza al finalizar el sprint además del Sprint Review. En la misma, se hace una retrospectiva del sprint intentando identificar tanto los aciertos como los fallos cometidos, y todos los miembros del equipo deben comunicar sus impresiones. Tiene como objetivo la mejora continua del proceso.

4.1.2 Aplicación de Scrum

En esta sección se aborda la manera en la que se han adaptado las recomendaciones de Scrum a las particularidades de este proyecto.

Justificación

La elección de la metodología Scrum a la hora de realizar este proyecto se debe principalmente a las siguientes características:

- Planificación inicial mediante la creación del Product Backlog y las historias de usuario. De este modo se consigue tener una visión global de las necesidades del proyecto desde el principio de forma más ordenada y estructurada.
- Desarrollo basado en iteraciones, que permite gran flexibilidad a la hora de dividir las distintas fases de desarrollo y abordar cada característica del proyecto por separado. Además permite enfocarse en metas más pequeñas abordando los problemas de forma más contenida, aumentando la productividad y mejorando la calidad final.
- Reuniones al final de cada Sprint para determinar el estado del proyecto en intervalos predefinidos.

Adaptación al proyecto

A la hora de adaptar la metodología Scrum a este proyecto se han tenido en cuenta las necesidades especiales del mismo. Dado que es un proyecto realizado por un equipo de desarrollo de una sola persona con una capacidad de esfuerzo variable a lo largo del tiempo, ha habido que modificar la metodología de Scrum para adaptarla a dichas necesidades.

Para ayudar a realizar el proyecto cumpliendo con la metodología Scrum se ha hecho uso de la herramienta Taiga tal y como se ha descrito en la sección 3.6.3.

Roles

A continuación se describe cómo han sido cubiertos los roles de Scrum en base a las necesidades del proyecto:

- **Scrum Team:** al tratarse de un Trabajo de Fin de Grado el equipo de desarrollo está conformado por un solo miembro, el alumno, encargado de todo el análisis y desarrollo del producto.
- **Scrum Product Owner:** este rol es llevado de forma compartida entre los directores del proyecto y el propio alumno, ya que juntos decidieron las necesidades y características del proyecto.
- **Scrum Master:** la tarea principal de este rol es la de establecer un puente de comunicación entre el equipo de desarrollo y el cliente. Dado que ambos roles están en mayor o menor medida llevados a cabo por el alumno, el objetivo de este rol queda un tanto

difuminado para el caso de este proyecto y es ejercido tanto por el alumno como por los directores a la hora de asegurar el cumplimiento de los objetivos marcados por la metodología.

Artefactos

En cuanto a la generación de los distintos artefactos de Scrum para el proyecto, se han identificado los siguientes:

- **Product Backlog:** a partir de las historias de usuario se ha generado un Product Backlog en la herramienta Taiga con las distintas tareas necesarias para la finalización del proyecto.
- **Sprint Backlog:** en cada Sprint llevado a cabo durante la realización del proyecto se ha definido un backlog con las tareas asignadas en Taiga.
- **Burn-Down Chart:** se ha generado con Taiga que muestra el avance del proyecto según las historias de usuario realizadas en cada Sprint.
- **Entregables:** al final de cada Sprint se han creado *releases* en Github (ver Figura 4.3) con los avances en cada servicio que determinan una versión del producto en esa etapa. Además, al finalizar el proyecto, se ha generado un CD con todo el código producido en el desarrollo así como la memoria final y los diagramas e imágenes que la componen.

Reuniones

A lo largo de la creación del proyecto se han llevado a cabo reuniones intentando seguir el proceso de Scrum desde un punto de vista pragmático, dadas las circunstancias especiales de los distintos *stakeholders*.

La mayoría de reuniones se han *simulado* mediante la comunicación por correo electrónico entre los directores y el alumno, si bien en algunos casos como el Sprint Planning Meeting de varios sprints se han realizado en persona para poder determinar de una forma más eficiente las necesidades del futuro Sprint.

Al comienzo y final de cada Sprint el alumno envió un correo electrónico a sus directores indicando cómo se había desarrollado el mismo, describiendo el grado de finalización de cada tarea programada así como las tareas planificadas para el siguiente Sprint.

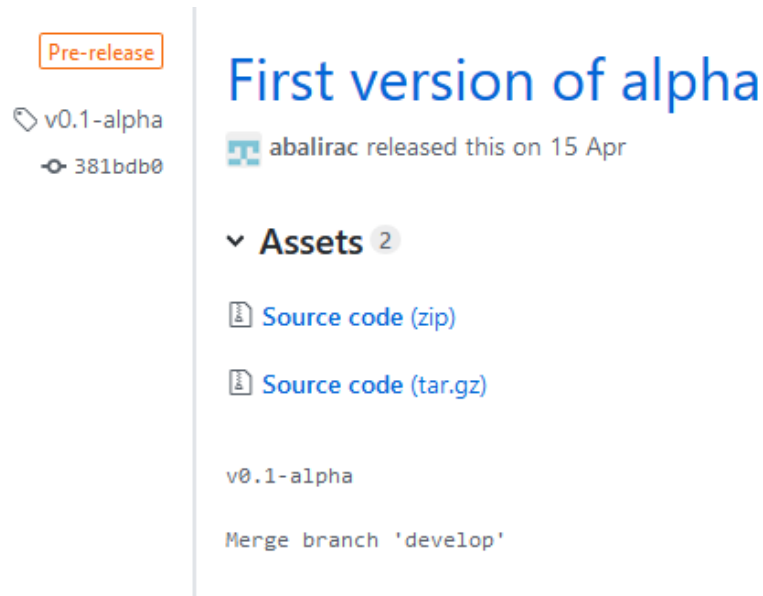


Figura 4.3: Primera *release* de la API del proyecto en Github.

4.2 Gestión del proyecto

En esta sección se expone cómo ha sido gestionado el proyecto a lo largo de su ciclo de vida, intentando llevarlo a cabo de forma exitosa, cumpliendo los objetivos de tiempo, coste, esfuerzo y calidad marcados.

Gracias a una buena gestión del proyecto podemos realizar un seguimiento adecuado de su evolución en cualquier punto del desarrollo, de forma que sea posible optimizar los recursos al máximo. Además, se ha realizado un análisis del coste, estimación del tiempo, asignación de recursos y la gestión de riesgos en el proyecto.

4.2.1 Estimación

A la hora de estimar las distintas historias de usuario se ha tenido en cuenta el esfuerzo necesario para realizarlas, no siendo necesariamente equivalente a las horas-hombre [13] que suelen ser usadas en métodos más tradicionales. Para ello se han empleado **Story Points**, que son unidades basadas en el esfuerzo y tienen en cuenta la complejidad, incertidumbres y capacidades para llevar a cabo una tarea, entre otros factores.

En este proyecto se ha estimado el tamaño de cada Sprint en **29 Story Points**, que en condiciones normales debería tener una duración de dos semanas. Sin embargo, a lo largo del proyecto, la capacidad de esfuerzo del equipo de desarrollo ha sido variable, por lo que se ha

intentado mantener la regularidad en el esfuerzo en vez de en la duración, y por tanto, unos sprints han durado más que otros.

La estimación total del proyecto fue de 10 sprints de 29 Story Points cada uno con un total de **290 Story Points**, y una duración aproximada de 5 meses.

4.2.2 Planificación

A continuación se muestra la planificación hecha para el proyecto enumerando las historias de usuario previstas en cada Sprint:

- Sprint 0: Análisis de requisitos, creación arquetipo inicial de la aplicación.
- Sprint 1: Extraer ofertas de Telegram.
- Sprint 2: Mostrar información detallada de una oferta, listar últimas ofertas disponibles.
- Sprint 3: Registro y autenticación de usuarios.
- Sprint 4: Extraer información relevante de las ofertas de los canales de Telegram.
- Sprint 5: Hacer despliegue de servicios en la nube, búsqueda de ofertas.
- Sprint 6: Hacer web scraping a las tiendas relacionadas con las ofertas.
- Sprint 7: Creación del Web Crawler.
- Sprint 8: Añadir paginación a lista de ofertas, detectar duplicados en ofertas.
- Sprint 9: Borrador de la memoria del proyecto, clasificar ofertas por categoría.

Como recomienda la metodología Scrum, la planificación del proyecto se fue realizando de forma independiente en cada iteración, al comienzo de cada Sprint.

Idealmente la duración de todos los sprints debería ser la misma, y así se hizo al comienzo del proyecto. Los primeros sprints tienen todos una duración similar de 14 días, pero posteriormente, debido a circunstancias externas al propio proyecto (motivos laborales del equipo de desarrollo), el esfuerzo requerido para cada Sprint tuvo que dilatarse en el tiempo, e incluso hacer paradas entre sprints al no haber recursos personales disponibles para la realización del proyecto (ver Figura 4.4).

4.2.3 Recursos

En cuanto a los recursos destinados a la realización del proyecto cabe destacar el componente humano:

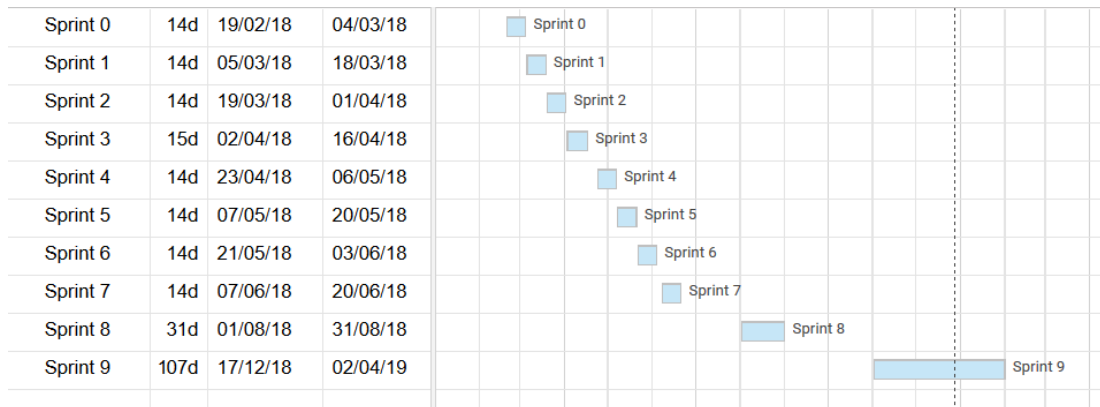


Figura 4.4: Diagrama de Gantt de la planificación temporal.

- El equipo de desarrollo: Álvaro Balirac Seijas.
- Los directores: Javier Parapar López, Paula López Otero y Daniel Valcarce Silva.

Por otro lado, los recursos materiales empleados en el proyecto son propiedad del alumno, como el ordenador portátil, o proporcionados por los directores, como la herramienta Taiga. También se ha hecho uso de herramientas de software libre como Elasticsearch o software con licencia de estudiantes como PhpStorm o GitHub. Para los recursos de despliegue en servidores se ha hecho uso de servicios como la capa Amazon Web Services Free Tier³ o los servicios en la nube Google Cloud⁴.

4.2.4 Costes

Para calcular los costes del proyecto se ha usado la siguiente fórmula:

$$(StoryPointsTotales/velocity)*horasPorSprint*costePorHora+costesNoLaborales$$

Se ha determinado un salario medio por hora de 25€, y el tamaño del Sprint en dos semanas, así como el esfuerzo diario normal en una empresa de ocho horas por cinco días a la semana. Además se han tenido en cuenta los costes no laborables como las licencias del software empleado.

Para el despliegue en servidores se ha hecho uso de servicios gratuitos como la capa Amazon Web Services Free Tier o los créditos gratuitos de Google Cloud, pero su coste real supondría un total 102,64€ al mes y dos meses y medio de uso, ya que los servidores en la nube no serán necesarios hasta la segunda mitad del desarrollo del proyecto cuando los servicios empiecen a estar operativos (ver Tabla 4.1).

³<https://aws.amazon.com>

⁴<https://cloud.google.com>

Nombre	Servicios	Precio
Instancia de Amazon RDS	Base de datos relacional	9,27€ / mes
Instancia de Amazon EC2	API REST, Text Miner, Bot Telegram	5,73€ / mes
Instancia de Amazon Elasticsearch	Motor de búsqueda, Text Miner	24,48€ / mes
Instancia flexible de Google Cloud	Aplicación web	62,86€ / mes

Tabla 4.1: Costes de servidores.

- **StoryPointsTotales** = 290.
- **Velocity** = 29 Story Points por Sprint.
- **horasPorSprint** = 8 horas * 5 días * 2 semanas = 80 horas por Sprint.
- **costePorHora** = 25€/hora.
- **costesNoLaborales** = 1599€ (portátil) + 199€ (Licencia PhpStorm) + 102,64€ (instancias Amazon AWS y Google Cloud) * 2,5 meses = 2054,60€.

El coste total estimado del proyecto es $(290 / 29) * 80 * 25 + 2054,60 = \mathbf{22.054,60€}$.

4.2.5 Gestión de riesgos

En todos los proyectos de ingeniería un factor muy importante a tener en cuenta es la correcta gestión de los riesgos asociados. Por ello, se intenta identificar, clasificar y tratar la exposición a los riesgos que puedan darse, intentando minimizar las posibles consecuencias de su aparición. En base a la exposición que pueda tener el proyecto a los mismos, se intentarán prevenir, realizar un seguimiento o asimilarlos (ver Tabla 4.2).

Identificación y análisis

El primer paso a la hora de tratar los riesgos asociados al proyecto es identificarlos. Para ello, se ha creado una tabla en la que se identifica cada riesgo y se clasifica según su probabilidad de aparición, impacto en el proyecto en caso de que ocurra y la exposición del proyecto al riesgo⁵:

Prevención

Una vez se han identificado y clasificado los riesgos según su exposición, aquellos con un riesgo de exposición alta deben ser tratados desde el principio del proyecto con especial cuidado, dado que una exposición alta a los mismos podría conllevar el fracaso del proyecto.

⁵La exposición al riesgo es la probabilidad de que suceda un determinado evento, multiplicado por la pérdida que genere.

#ID	Descripción	Probabilidad	Impacto	Exposición
R1	Mala planificación	Media	Medio	Media
R2	Arquitectura incorrecta	Baja	Alto	Media
R3	Diseño equivocado	Baja	Medio	Baja
R4	Cambio de requisitos	Baja	Medio	Baja
R5	Falta de tecnología	Baja	Medio	Baja
R6	Escasez de recursos	Alta	Medio	Alta
R7	Mala implementación	Baja	Alto	Media
R8	Cambio origen de datos	Media	Alto	Alta

Tabla 4.2: Riesgos del proyecto.

Por ello, estos riesgos deben ser prevenidos creando planes de contingencia para intentar mitigarlos en la medida de lo posible:

- **R6 - Escasez de recursos:** Dadas las condiciones especiales del alumno a la hora de destinar recursos personales para la realización del proyecto (motivos laborales), uno de los mayores riesgos para la consecución del mismo es la mitigación de la escasez de recursos. Por ello, se ha intentado planificar los recursos de modo que puedan destinarse los suficientes a este proyecto.
- **R8 - Cambio origen de datos:** Un gran riesgo a la hora de la consecución de este proyecto es la posibilidad de cambios en el origen de los datos. En caso de que la plataforma Telegram cambie sus requisitos para acceder a los canales o dichos canales dejen de estar disponibles generarían un riesgo difícilmente asumible para este proyecto. Por ello, se ha estudiado la posibilidad de extraer las ofertas de páginas web o buscarlas directamente en las tiendas online. Aunque estos métodos no serían tan sencillos ni óptimos como la solución planteada en este proyecto, sí mitigarían en gran parte el riesgo de no poder ser llevado a cabo, a cambio de una gran modificación en los requisitos del sistema y de la planificación del mismo.

Seguimiento

A los riesgos de exposición media se les debe realizar un seguimiento para evitar que puedan degenerar en riesgos de exposición alta, comprometiendo así el proyecto:

- **R1 - Mala planificación:** Debido a que la experiencia del alumno a la hora de planificar proyectos no es muy amplia, aumenta la probabilidad de se lleve a cabo una mala planificación, generando desviaciones según la planificación inicial.

Al hacer uso de una metodología como Scrum, es posible realizar un seguimiento del proyecto de forma más organizada y sencilla, al tener todo dividido en Sprints y tareas

de modo que puedan realizarse los ajustes necesarios para evitar cualquier desviación.

- **R2 - Arquitectura incorrecta:** Aunque la arquitectura debe ser escogida al comienzo del proyecto de forma cuidadosa y la probabilidad de la elección no haya sido buena es baja, el impacto que tendría en el proyecto sería muy alto, por lo que se ha realizado un seguimiento a lo largo del desarrollo para comprobar que se ha escogido la opción correcta.
- **R7 - Mala implementación:** Mediante el seguimiento continuo e integración continua realizados a lo largo del desarrollo del proyecto se ha garantizado la implementación correcta de todas las funcionalidades.

Asimilación

Aquellos riesgos cuya exposición sea leve no requerirán realizar un plan de contingencia ni un seguimiento a lo largo del proyecto, dado que llevar a cabo dichas acciones podría resultar en un mayor coste que el beneficio adquirido de su implementación:

- **R3 - Diseño equivocado:** Al usar Scrum podemos asimilar errores en el diseño planificando Sprint a Sprint las tareas relacionadas con ajustes en el proyecto derivados de un diseño equivocado.
- **R4 - Cambio de requisitos:** Dado que el proyecto ha sido principalmente diseñado y planificado por el alumno, la probabilidad de que se exijan cambios en el proyecto es muy baja, pudiendo ser los mismos asimilados y reajustados en los distintos Sprints del desarrollo.
- **R5 - Falta de tecnología:** La elección tecnológica ha sido llevada a cabo al inicio del proyecto teniendo en cuenta las necesidades del mismo, escogiendo una solución que permita realizar todas las funcionalidades de una manera óptima, aprovechando los conocimientos del alumno de las mismas así como de las posibilidades de cara al futuro que dichas tecnologías permiten. Por lo tanto, un cambio en las tecnologías usadas es poco probable y su impacto medio ya que se podrían realizar cambios en el proyecto debido a su arquitectura basada en servicios que permite una mayor flexibilidad a la hora de realizar modificaciones en las tecnologías usadas.

Desarrollo

EN este capítulo se detalla el proceso de desarrollo del proyecto. En primer lugar, se analizan los requisitos del proyecto para definir la arquitectura general del sistema. A continuación se describen el análisis, el diseño y la implementación de los componentes. Por último, se muestran datos ofrecidos por las herramientas de soporte al desarrollo.

5.1 Análisis de Requisitos

La captura de requisitos se ha realizado mediante el uso de historias de usuario tal y como se recomienda en la metodología Scrum.

Al inicio del proyecto los distintos Stakeholders del proyecto (alumno y directores) llevaron a cabo una reunión para determinar los requisitos del proyecto así como crear una primera versión del Product Backlog con las tareas generadas a partir de las historias de usuario recabadas.

5.1.1 Requisitos funcionales

En la Tabla 5.1 se muestran los requisitos funcionales definidos mediante historias de usuario, así como los puntos de historia que determinan su complejidad.

5.1.2 Requisitos no funcionales

Por otro lado, no debemos olvidarnos de los requisitos no funcionales. Estos son aquellos que especifican criterios a cumplir por el sistema (en lugar de funciones específicas como indican los requisitos funcionales). Estos dan lugar a decisiones de diseño en la arquitectura y sus componentes así como en las elecciones tecnológicas detalladas en la Sección 3. En la Tabla 5.2 se recogen los requisitos no funcionales deducidos tras el proceso de ingeniería de requisitos.

#ID	Nombre	Story Points
#52	Análisis de requisitos	15
#1	Creación arquetipo inicial de la aplicación	14
#10	Extraer ofertas de Telegram	29
#3	Mostrar información detallada de una oferta	13
#2	Listar últimas ofertas disponibles	14
#5	Registro y autenticación de usuarios	29
#18	Extraer información relevante de las ofertas de los canales de Telegram	28
#45	Hacer despliegue de servicios en la nube	17
#12	Hacer web scraping a las tiendas relacionadas con las ofertas	29
#6	Creación del Web Crawler	28
#8	Búsqueda de ofertas	13
#19	Detectar duplicados en ofertas	15
#40	Añadir paginación a lista de ofertas	15
#9	Clasificar ofertas por categoría	18
#53	Borrador memoria del proyecto	13

Tabla 5.1: Requisitos funcionales: Product Backlog.

5.2 Arquitectura propuesta

En base a los requisitos enunciados en la sección anterior se propone una **arquitectura orientada a servicios (*service-oriented architecture* - SOA)**, en la que el sistema está dividido en varios componentes que proveen de servicios a otras partes del sistema a través de protocolos de comunicación (en este caso HTTP).

Un servicio es una unidad discreta de funcionalidad que puede ser accedida de forma remota y actualizada de forma independiente.

Hasta hace poco la forma más común de implementar un sistema basado en una arquitectura orientada a servicios era mediante la creación de servicios basados en SOAP y construidos sobre una arquitectura estilo RCP [14], pero en los últimos años el uso de APIs REST como interfaces de conexión entre las distintas partes del sistema ha ganado mucha popularidad.

En SOAP se emplea el intercambio de datos XML, las operaciones son definidas como puertos WSDL y la dirección es única, a lo que hay que añadir que numerosas instancias del proceso comparten la misma operación. En REST, sin embargo, las operaciones se definen en los propios mensajes y hay una dirección única para cada instancia del proceso. Los componentes no se acoplan del mismo modo en ambos casos; mientras que en SOAP están fuertemente acoplados, en una API REST esa unión es débil [15]. Por ello se ha decidido usar una API REST para conectar los distintos servicios en este proyecto.

El resultado del proceso de ingeniería seguido ha dado lugar al esquema general de la arquitectura del sistema que se ilustra en la Figura 5.1, donde se pueden apreciar las caracte-

#ID	Nombre	Descripción
RNF-01	Fiabilidad	El sistema deberá reducir al mínimo la tasa de fallos, devolviendo siempre que sea posible los mismos resultados.
RNF-02	Seguridad	El sistema deberá ser seguro frente a amenazas externas. Los datos sensibles como información de los usuarios deben ser cifrados.
RNF-03	Usabilidad	El sistema deberá ser fácil de usar tanto por los usuarios como por los administradores.
RNF-04	Robustez	El sistema deberá ser capaz de reaccionar apropiadamente ante situaciones excepcionales, como picos en los niveles de carga.
RNF-05	Disponibilidad	El sistema deberá tener una alta disponibilidad. Los usuarios deberán poder acceder a los servicios la mayor parte del tiempo.
RNF-06	Rendimiento	El sistema deberá responder de forma rápida a las peticiones del usuario reduciendo al máximo posible los tiempos de respuesta y velocidad general.

Tabla 5.2: Requisitos no funcionales.

rísticas arquitectónicas de cada uno de los componentes de la plataforma.

5.3 API REST

Este servicio es el encargado de realizar las comunicaciones entre la base de datos y el resto de servicios. Provee de una interfaz estándar basada en las APIs RESTful para poder establecer las distintas operaciones de lectura y escritura de datos.

La API ha sido construida usando el framework PHP Laravel Lumen (ver Sección 3.3.1).

Como se puede observar en la Figura 5.2 la creación de interfaces REST con Lumen es muy sencilla, basta con unas pocas líneas de código para establecer la base de las llamadas a la API que estarán disponibles para el resto del sistema.

Además, con el uso del ORM imbuido en Lumen llamado Eloquent, se puede mapear cada llamada a la API a una consulta a la BD para cada operación CRUD necesaria.

5.3.1 Descripción y diseño de funcionalidades

Alta y autenticación de usuarios

La API permite, mediante peticiones HTTP, la **creación de nuevos usuarios** para la aplicación como se puede ver en la Figura 5.3.

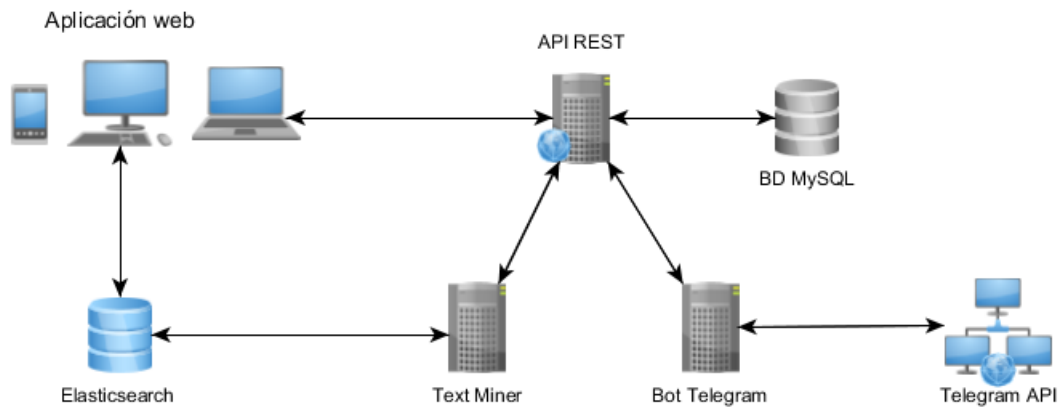


Figura 5.1: Esquema general de la arquitectura del sistema.

En primer lugar se ha creado una migración, que es un esquema que permite el control de versiones de la base de datos así como su definición, tal y como se muestra en la Figura 5.4

A partir de la llamada POST realizada a la API, desde el router de la aplicación se enruta la llamada al controlador:

```
$router->post('auth/register', 'AuthController@register');
```

Desde el controlador se invoca la función *register* que, dados los parámetros enviados a través de la petición HTTP, se hace primero una búsqueda en la base de datos por correo electrónico y en caso de no haber un usuario con esos datos se invoca a la clase *User* que generará automáticamente un nuevo usuario en la base de datos empleando el ORM.

Si la creación del usuario ha sido exitosa se devolverá en la petición la información del usuario y un código HTTP 200. En caso de ya existir un usuario con la misma dirección de correo electrónico se devuelve un código de error HTTP 400 con el mensaje *user-already-exists*, en cualquier otro caso se retorna un 401 *Unauthorized*.

Para la **autenticación de usuarios** se ha hecho uso de un sistema de Json Web Tokens. Una de las razones principales para el uso de JWT en vez de sesiones o cookies es que al usar JWT se puede asegurar uno de los principios fundamentales de las APIs Restful, que sea "stateless", ya que los JWT se mandan en cada petición y no requieren que se almacene información de la sesión del usuario en el servidor.

Este sistema se ha implementado mediante el uso de una librería para Laravel llamada *jwt-auth* que permite la creación y comprobación de tokens en el backend y un *middleware* de autenticación para crear restricciones de acceso a las distintas rutas de la API según el nivel de acceso del usuario.

Para la aplicación se han creado tres niveles de usuario distintos:

```

$router->get('/', function () use ($router) {
    return $router->app->version();
});

$router->group(['prefix' => 'api'], function () use ($router) {

    $router->get('offers', ['uses' => 'OfferController@showAllOffers']);
    $router->get('offers/{id}', ['uses' => 'OfferController@showOneOffer']);

    $router->post('auth/login', 'AuthController@login');
    $router->post('auth/register', 'AuthController@register');
});

$router->group(['middleware' => ['auth:api', 'admin'], 'prefix' => 'api'], function () use ($router) {

    $router->post('offers', ['uses' => 'OfferController@create']);
    $router->delete('offers/{id}', ['uses' => 'OfferController@delete']);
    $router->put('offers/{id}', ['uses' => 'OfferController@update']);

    $router->post('raw-offers', 'RawOfferController@create');
    $router->get('raw-offers', 'RawOfferController@showAllRawOffers');
    $router->delete('raw-offers/{id}', ['uses' => 'RawOfferController@delete']);
});

$router->group(['middleware' => ['auth:api'], 'prefix' => 'api'], function () use ($router) {

    $router->get('me', 'AuthController@getAuthenticatedUser');

```

Figura 5.2: Router de la API.

- **Guest:** nivel de acceso predeterminado. Los usuarios con este nivel pueden acceder a las funcionalidades de registro y autenticación, así como a las de visualización de ofertas.
- **User:** nivel de acceso de usuarios registrados. Permite el acceso a su propia información de usuario.
- **Admin:** nivel de administrador. Con este nivel se pueden añadir ofertas, modificarlas o borrarlas.

Estos niveles son especificados en el *router* de la API como se muestra en la figura 5.2.

Ver ofertas

A través de peticiones HTTP GET cualquier usuario de la aplicación puede acceder a un listado de ofertas así como a los detalles de una oferta en concreto.

De igual forma que en el alta de usuarios las peticiones son enrutadas a través del router de la API al controlador *OfferController.php*:

```

$router->get('offers', ['uses' => 'OfferController@showAllOffers']);
$router->get('offers/{id}', ['uses' => 'OfferController@showOneOffer']);

```

En la respuesta de la petición se devuelve el valor de las ofertas y un código HTTP 200.

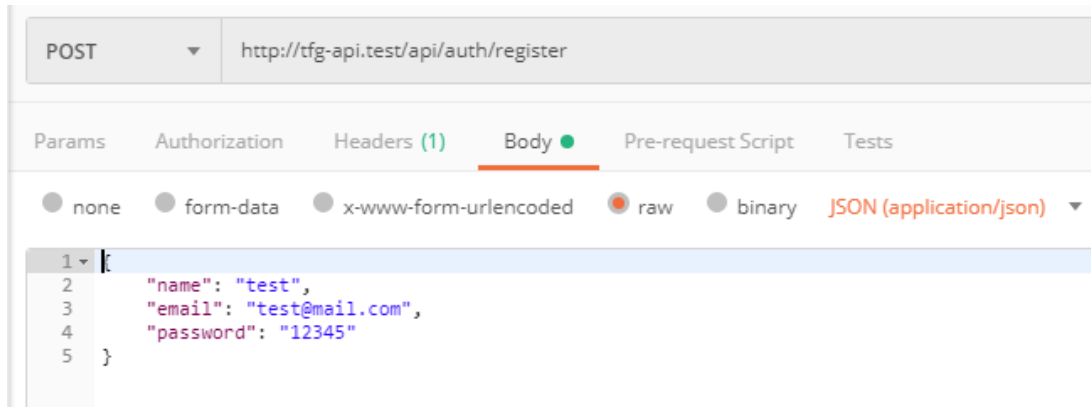


Figura 5.3: Ejemplo de llamada a la API para registrar un usuario.

Creación, actualización y borrado de ofertas

La API también ofrece métodos para las funcionalidades de creación, actualización y borrado de ofertas a través de peticiones POST, DELETE y PUT:

```
$router->post('offers', ['uses' => 'OfferController@create']);
$router->delete('offers/{id}', ['uses' => 'OfferController@delete']);
$router->put('offers/{id}', ['uses' => 'OfferController@update']);
```

Estas peticiones son gestionadas por el *middleware* de autenticación encargado de comprobar que las peticiones son hechas por usuarios autenticados y con el rol de *Admin*, devolviendo el resultado adecuado para cada petición.

Creación, actualización y borrado de ofertas del bot

Aparte de las ofertas almacenadas para ser visualizadas en el *frontend*, también se almacenan y gestionan a través de la API las ofertas extraídas directamente de los canales de Telegram. Estas ofertas son accedidas desde el servicio del *Text Miner* para analizar y extraer la información relevante de ellas.

```
$router->post('raw-offers', 'RawOfferController@create');
$router->get('raw-offers', 'RawOfferController@showAllRawOffers');
$router->delete('raw-offers/{id}', 'RawOfferController@delete');
```

En este caso también se exponen peticiones HTTP gestionadas a través del *middleware* de autenticación para evitar que usuarios no autorizados puedan acceder a esta información.

Modelo de datos

Para la API se ha utilizado un modelo clásico de datos basado en MySQL. Aunque al comienzo del proyecto se barajó la posibilidad de usar bases de datos NoSQL para la API, dadas

```
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->string( column: 'name');
            $table->string( column: 'email')->unique();
            $table->string( column: 'password');
            $table->string( column: 'role');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Figura 5.4: Migración de la tabla users.

las características del proyecto así como de las tecnologías disponibles, se decidió descartar esta opción para favorecer el uso de tecnologías con menores dependencias y más probadas.

Dado que el sistema va estar formado por varios servicios, el modelo de datos de la API ha sido simplificado todo lo posible para favorecer la flexibilidad del sistema así como la independencia de los servicios en la medida de lo posible, como se ve en la Figura 5.5.

5.4 Bot de Telegram

Este servicio se encarga de extraer las ofertas de los canales de Telegram haciendo uso de la API para clientes que la propia plataforma provee. Está basado en un proceso que se ejecuta continuamente escuchando los mensajes enviados en la plataforma de Telegram a los distintos canales a los que está suscrito el cliente configurado. Las ofertas leídas de los distintos canales son enviadas a la base de datos MySQL del sistema, a través de la API REST implementada para dicho proceso. Para crear este servicio se ha hecho uso de la biblioteca para PHP MadelineProto encargada de interactuar con la API de Telegram para la creación del Bot. Aunque Telegram provee dos APIs distintas, una para Bots y otra para clientes, para este proyecto ha sido necesario usar la interfaz que provee la biblioteca MadelineProto para crear un Bot a partir de la API de clientes, denominado userBot, dado que la API para Bots de Telegram no

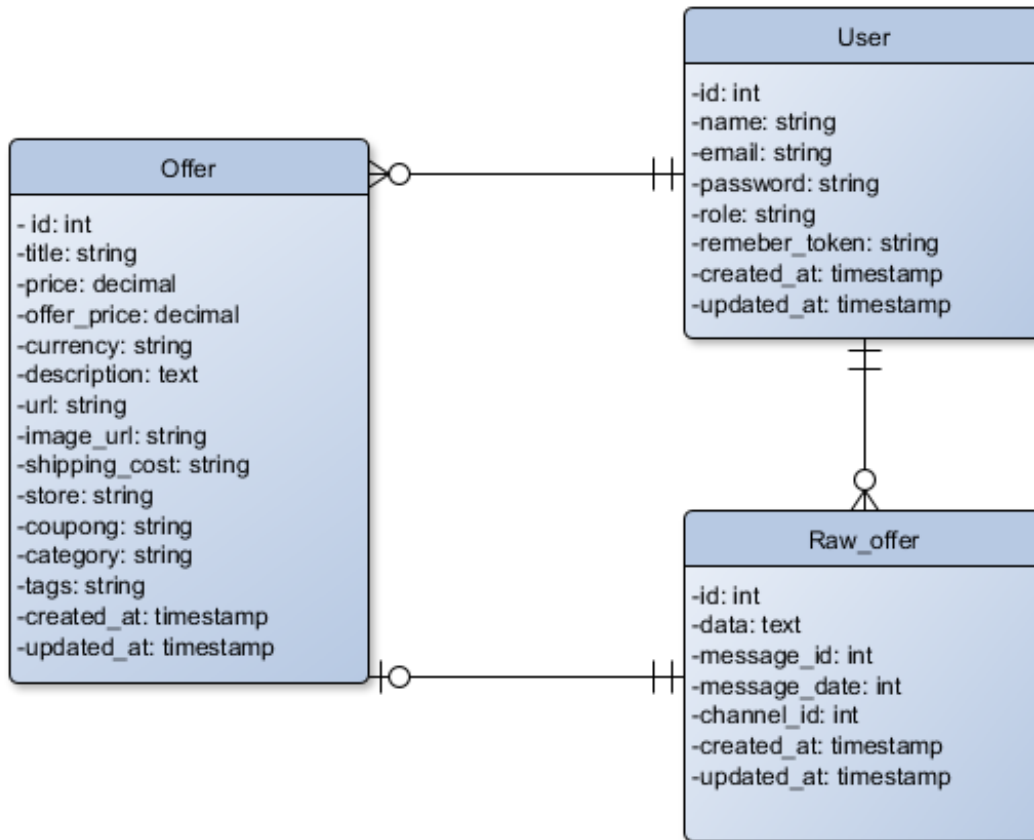


Figura 5.5: Diagrama modelo de datos de la API.

permite la lectura de información de canales a los Bots que no son administradores de dicho canal.

5.5 Elasticsearch

Este servicio con su propia API, que hace de motor de búsqueda para la aplicación así como de sistema para analizar y categorizar las distintas ofertas.

5.5.1 Búsqueda de ofertas

Elasticsearch provee un lenguaje *full Query DSL* (Lenguaje de dominio específico) basado en JSON para definir las consultas. Gracias a esto, se ha hecho uso de la consulta `search?q=` para realizar búsquedas de ofertas en el frontend de la aplicación. Pasando como parámetro el título de la oferta o producto que se quiera buscar, se pueden obtener de forma rápida y sencilla las ofertas más relevantes haciendo uso del potente motor proporcionado por Elasticsearch.

```

UpdateHandler, 328989699: Got channel too long update, getting difference...
UpdateHandler, 328989699: Handling an update of type updateEditChannelMessage...
UpdateHandler, 328989699: Applying pts, current pts: 283785, new pts: 283785, channel id: 1007686014
UpdateHandler, 328989699: Saving an update of type updateEditChannelMessage...
-----Searching for new offers-----
CallHandler, 328989699: Didn't serialize in a while, doing that now...
API, 328989699: Serializing MadelineProto...
Serialization, 328989699: Waiting for exclusive lock of serialization lockfile...
Serialization, 328989699: Lock acquired, serializing
*«UE Roll 2 - Altavoz Bluetooth ultraportátil #Amazon
*Antes: 33,99€ |Ahora: 28,09€ (-17%)
*Envío GRATIS con Amazon Prime
  «Link: chx.to/9xr
  «Precio mínimo
  «Impermeable, resistente a golpes
  «Incluye flotador para la piscina
  «-----
  «*¡Entra en t.me/ClubCHOLLOX y descubre todos nuestros canales!
  «-----Offer sent-----
  «MIFA SoundBox Altavoz Portátil Bluetooth 30W Todo en Aluminio Subgrave Potente, 4000mAh de Litio Recargable, Tecnología .
    x Antes: 99,99 EUR
    o Reacondicionado: Sin stock
    v Precio oferta: 24.9 EUR ■
  « CUPÓN: MIFA2019
  « COMPRAR: https://amzn.to/2IQS5mY
  « Powered by @zonachollos
  «-----Offer sent-----
  «Xiaomi Mi Mix 2S 6GB Versión Global desde 261,95€ en Aliexpress»
  «-----
  «Precios mínimos históricos para el Mi Mix 2S Versión Global con Rom Global (Multilenguaje) y diferentes capacidades de almacenamiento»

```

Figura 5.6: Bot de Telegram recopilando ofertas.

5.5.2 Categorización de ofertas

Además de ser usado para realizar búsquedas, ElasticSearch también puede ser empleado para la clasificación de texto [16]. En el caso de este proyecto es empleado para, a partir de un conjunto de datos de ofertas clasificadas en distintas categorías, extrapolar la categoría de otras ofertas usando la información disponible de las mismas como son su título o descripción. De este modo podemos usar un conjunto homogéneo de categorías para clasificar las ofertas de distintas tiendas online que de otro modo estarían categorizadas de distinta forma según los criterios de cada tienda.

La forma común de realizar clasificación de texto sin Elasticsearch consistiría en obtener el texto usando código personalizado, *parsear* el documento a mano o usando algún tipo de biblioteca como Tika library¹ y utilizar una biblioteca tradicional o API como NLTK², OpenNLP³, Stanford NLP⁴ o cualquier otra herramienta desarrollada para investigación. A menudo los formatos de datos son propietarios y las herramientas necesitan ser compiladas y ejecutadas en línea de comandos.

Por otro lado, con ElasticSearch y sus analizadores de diferentes idiomas, solo se necesita configurar los *mappings* e indexar los datos. El pre-procesado sucede automáticamente en tiempo de indexación.

¹<https://tika.apache.org/>

²<https://www.nltk.org/>

³<https://opennlp.apache.org/>

⁴<https://nlp.stanford.edu/software/>

Consulta *More Like This*

Para implementar la clasificación de ofertas en distintas categorías se ha hecho uso de la consulta que provee Elasticsearch llamada *More Like This* (MLT). Esta consulta encuentra documentos que son parecidos a un conjunto de documentos de entrada. Para conseguir esto, MLT selecciona de forma automática un conjunto representativo de términos de los documentos introducidos, genera una consulta usando estos términos, y la ejecuta devolviendo los resultados convenientes. El usuario controla la entrada de documentos, cómo deberían ser seleccionados los términos y cómo debe ser creada la consulta.

Para realizar la clasificación fue necesario crear un gran corpus de datos a partir de las ofertas extraídas de una de las tiendas. En este caso se usó la clasificación de tiendas utilizada por Amazon.es para categorizar sus productos en venta. Este conjunto de datos fue creado mediante el desarrollo de un web crawler tal y como es explicado en la Sección 5.6.4.

Una vez se obtiene un conjunto de datos suficientemente grande, se realiza la consulta MLT, obteniendo un subconjunto de productos de los más parecidos al producto de entrada ordenados según un *score* establecido por Elasticsearch. Aquella categoría con mayor representación dentro de ese subconjunto de resultados es seleccionada como la categoría de la nueva oferta y almacenada en la base de datos.

Modelo de datos

Las ofertas además de ser almacenadas en la base de datos MySQL, se guardan en una base de datos NoSQL basada en documentos, haciendo uso de la tecnología de Elasticsearch, con la finalidad de tener un motor de búsqueda robusto y fiable para realizar todas las búsquedas necesarias en la aplicación. Ver Figura 5.7.

5.6 Text Miner

Este servicio es el encargado de leer las ofertas sin analizar almacenadas en la base de datos y extraer información relevante de ellas, guardando de forma estructurada aquellas que son válidas y descartando las ofertas que no se ajustan a los parámetros establecidos. Está construido utilizando el lenguaje de programación PHP. Debido a limitaciones en la capacidad para extraer datos de algunas tiendas online, para este proyecto se ha limitado la extracción de ofertas a las tiendas online de Amazon y Gearbest, que en conjunto representan alrededor del 80-90% del total de las ofertas extraídas de Telegram.

El *Text Miner* almacena las ofertas en la base de datos para su posterior uso en las búsquedas con Elasticsearch o para ser mostradas en la aplicación web. Es un proceso que se ejecuta continuamente esperando a recibir ofertas no procesadas, almacenadas en la base de datos por el Bot de Telegram. Una vez obtiene un conjunto de ofertas sin tratar, descompone cada

```
1 {
2   "offers": {
3     "properties": {
4       "title": {
5         "type": "text",
6         "analyzer": "spanish"
7       },
8       "price": {"type": "float"},
9       "offer_price": {"type": "float"},
10      "currency": {"type": "text"},
11      "description": {
12        "type": "text",
13        "analyzer": "spanish"
14      },
15      "url": {"type": "text"},
16      "image_url": {"type": "text"},
17      "shipping_cost": {"type": "text"},
18      "store": {"type": "text"},
19      "coupon": {"type": "text"},
20      "category": {
21        "type": "text",
22        "analyzer": "spanish",
23        "fields": {
24          "raw": {
25            "type": "keyword"
26          }
27        }
28      },
29      "tags": {
30        "type": "text",
31        "analyzer": "spanish"
32      },
33      "created_at": {"type": "text"},
34      "updated_at": {"type": "text"}
35    }
36  }
37 }
```

Figura 5.7: Modelo de datos de ofertas en Elasticsearch.

una de ellas *tokenizando* el texto almacenado de cada una con PHP NlpTools (Sección 3.4.1), para permitir un mejor análisis posterior de sus datos. Se usa un `WhitespaceTokenizer` que divide el texto en palabras. El proceso de análisis de texto se realiza mediante el uso de heurísticas para la mejora de la detección del contenido relevante de las ofertas.

5.6.1 Análisis de precios

Una vez *tokenizado* el texto se comprueba cada token para ver si contiene un precio, analizando sus caracteres para detectar símbolos de monedas como \$, €, EUR o USD. Tras haber detectado uno de estos símbolos se comprueba si los tokens cercanos contienen alguna palabra clave como: *oferta*, *ahora*, *flash*, *final*, *solo*, *sólo*, *desde*, o *precio*. Cuando se encuentra una palabra de las anteriores cerca de un token con un precio, existe una alta probabilidad de que

ese precio sea el de la oferta, descartando así cualquier otro precio sin relevancia para este proyecto incluido en el texto. En este caso el rango de cercanía al token de precio establecido es uno.

5.6.2 Análisis de cupones de descuento

Muchas ofertas se ofrecen mediante un código de descuento o cupón que deberá ser introducido en la tienda a la hora de realizar la compra. Por ello, se han establecido una serie de comprobaciones heurísticas para identificar de forma rápida los códigos de descuento. Se hace una búsqueda en el texto para encontrar tokens que contengan las palabras clave: *cupon*;, *cupón*;, *cupon*, *cupón*, o *descuento*. Una vez encontrado el token, se comprueba que el siguiente token no contenga las palabras clave: *descuento*, *oferta* o *flash* ya que éstas normalmente indican la presencia de un descuento directo, sin cupón. A continuación se comprueba el siguiente token para descartar tokens con precios y se obtiene el potencial código de cupón.

Como los códigos de descuento suelen tener unas características únicas, como un tamaño largo de caracteres, uso de mayúsculas, o ausencia de caracteres no alfanuméricos, se ha creado una función para determinar entre todos los potenciales códigos de descuento de una oferta cuál es el código con mejor puntuación y, por lo tanto, escogido como cupón de descuento para la oferta:

```
function couponCodeScore($coupon) {

    $score = 0;

    //código en mayúsculas
    if (mb_strtoupper($coupon, 'utf-8') == $coupon)
        $score++; else $score = $score - 0.5;

    //tamaño menor de 3 caracteres
    if (strlen($coupon) <= 3) $score = $score - 1.5;
    //tamaño mayor o igual a 6 caracteres
    if (strlen($coupon) >= 6) $score = $score + 0.8;
    //tamaño mayor que 8 caracteres
    if (strlen($coupon) > 8) $score = $score + 0.4;

    //código alfanumérico
    if(preg_match('/^[A-Z0-9]{4,64}$/i', $coupon))
        $score++; else $score = $score - 0.5;

    return $score;

}
```

5.6.3 Análisis de información del producto

Obtenidos ya el precio de la oferta y el posible cupón de descuento se procede a determinar las restantes características de la oferta, ligadas a la información del producto y la tienda, como son el título, descripción, imagen, enlace, gastos de envío, categoría o etiquetas. Se recorre el texto tokenizado para extraer los enlaces que contiene, se selecciona aquellos que son de las tiendas permitidas (Amazon y Gearbest) y se extrae información de la oferta. Para ello se hace uso de la biblioteca para PHP Goutte (3.1.2) que permite hacer *Web Scraping* a una URL determinada, seleccionando la información contenida en determinados elementos del DOM de la web.

Durante este proceso si la tienda corresponde a Amazon se almacena su categoría y en caso de ser de Gearbest se hace una consulta MLT a Elasticsearch para categorizar la oferta según la estructura de Amazon, tal y como se explica en la Sección 5.5.2).

5.6.4 Web Crawler

El *web crawler*, también llamado “araña” (del inglés *spider*) es un script implementado dentro del servicio Text Miner para recorrer la página web amazon.es y recopilar información de multitud de productos para ser usado en conjunción con Elasticsearch para la categorización de ofertas (ver Sección 5.5.2). De este modo se ha conseguido crear un conjunto de más de 10.000 productos clasificados dentro de más de 20 categorías distintas y almacenado en ElasticSearch para su posterior uso (Figura 5.8). Ha sido implementado haciendo uso también de la biblioteca para PHP Goutte.

El *script* comienza analizando el árbol DOM de la URL semilla, en este caso <http://amazon.es>, que es la página índice de la tienda de Amazon para España, en busca de enlaces relativos para acotar el recorrido del crawler a la propia web de Amazon. Estos enlaces también son filtrados descartando aquellos que no contienen un código ASIN (código único de producto) en su ruta o parámetros, asegurando que sólo se accede a partes de la web que contienen productos. Estos enlaces son almacenados en Elasticsearch con un parámetro *visited* a falso por lo que de forma recursiva, el *script* irá accediendo a los enlaces almacenados recuperando más enlaces y, en caso de encontrar un nuevo producto, almacenando su información y marcando la URL como visitada. Las URL que contienen un código ASIN son comprobadas para no ser recorridas en caso de haber sido visitadas o ya haber en el conjunto de productos almacenado uno con el mismo código.

De esta forma se ha podido crear un conjunto de datos de productos de Amazon en cuestión de horas, balanceando el número de productos por categoría introduciendo un valor aleatorio a la hora de acceder al siguiente enlace.

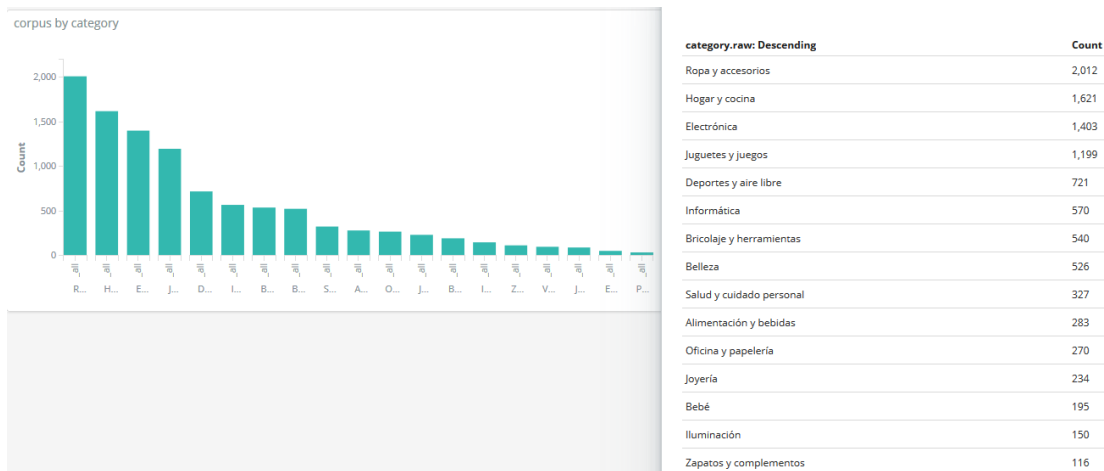


Figura 5.8: Vista en Kibana del corpus de productos almacenados en Elasticsearch.

5.7 Aplicación web

Este servicio es el encargado de la visualización de datos. Se trata de una aplicación web capaz de mostrar las ofertas en multitud de dispositivos compatibles con navegación web. Además, permite realizar búsquedas conectándose a la API del servicio de Elasticsearch del sistema. Ha sido desarrollado haciendo uso del framework Javascript Angular.

La aplicación está dividida en componentes siguiendo las directrices del manual de referencia de Angular [17]. Cada componente divide lógicamente la aplicación por funcionalidades permitiendo una mayor modularidad y desacoplamiento de sus distintas partes. Los componentes están estructurados en 4 ficheros diferentes que sirven el código siguiendo una división basada en responsabilidades:

- `$nombre.component.html`: fichero con el código HTML, encargado de la estructura de la vista del componente.
- `$nombre.component.scss`: fichero con el código Sass que genera los estilos CSS aplicados al HTML.
- `$nombre.component.spec.ts`: fichero Typescript con el código encargado de las pruebas del componente.
- `$nombre.component.ts`: fichero Typescript que implementa el controlador encargado de manejar la lógica del componente.

La comunicación entre el fichero `.ts` que contiene el controlador, y el fichero `.html` que implementa la lógica de la vista es llevada a cabo de forma automática por Angular mediante

two-way databinding, que permite la actualización de datos en dos direcciones, cambios en la vista (introducción de datos por el usuario por ejemplo) pueden producir cambios en el controlador, y viceversa, los cambios producidos en el controlador (por ejemplo una llamada a la API con nuevos datos) serán inyectados de forma automática en la vista, *renderizando* de nuevo el componente cuando sea necesario.

La propia aplicación es a su vez el componente principal en el que se incluyen los distintos sub-componentes que implementan las funcionalidades. Asimismo la comunicación de los componentes y de la API se realiza a través de servicios inyectables que encapsulan la lógica de las comunicaciones entre capas de abstracción.

Además mediante el uso de un *router*, así como de la implementación de Angular Universal (ver sección 3.5.1) se ha conseguido emular las características principales de una web tradicional multi-página en una aplicación SPA, aunando los beneficios de ambos métodos y reduciendo sus desventajas. Así podemos inicializar la aplicación en distintos estados de manera sencilla gracias a la implementación de rutas que siguen una estructura similar a las URLs clásicas sin la necesidad de recargar la página entera como se hacía anteriormente. Por otro lado, la implementación de Angular Universal permite pre-renderizar partes de la aplicación de forma estática, para que sean servidas de forma rápida a los usuarios mientras la aplicación dinámica es cargada y preparada por el navegador. De esta forma aumenta la velocidad de carga y se soluciona el problema de indexación en buscadores que suele ser común en las aplicaciones SPA, ya que a diferencia del contenido servido mediante peticiones AJAX, las versiones pre-renderizadas de la aplicación sí son visibles para los *Bots* que *indexan* la Web[18].

5.7.1 Registro y autenticación de usuarios

La aplicación web permite a los usuarios crear una cuenta y registrarse, tal y como puede verse en la Figura 5.9. Para llevar a cabo esta funcionalidad se decidió implementar un sistema de Json Web Tokens que permite realizar autenticación con la API de forma totalmente *stateless*, es decir, que el *backend* no necesita mantener un registro de los tokens, sino que es el *frontend* el encargado de almacenarlo y enviarlo en cada petición, para que a través de la API simplemente se valide el token y por lo tanto, se autorice la petición (Ver figura 5.10).

Este componente permite al usuario registrarse mediante un formulario introduciendo un nombre, dirección de correo electrónico y una contraseña. Además los usuarios una vez conectados, podrán autenticarse en la aplicación mediante otro formulario en el que deberán introducir su dirección de correo electrónico y contraseña. Una vez autenticado, el usuario es redirigido al listado de ofertas teniendo acceso a las funcionalidades asignadas a su rol.

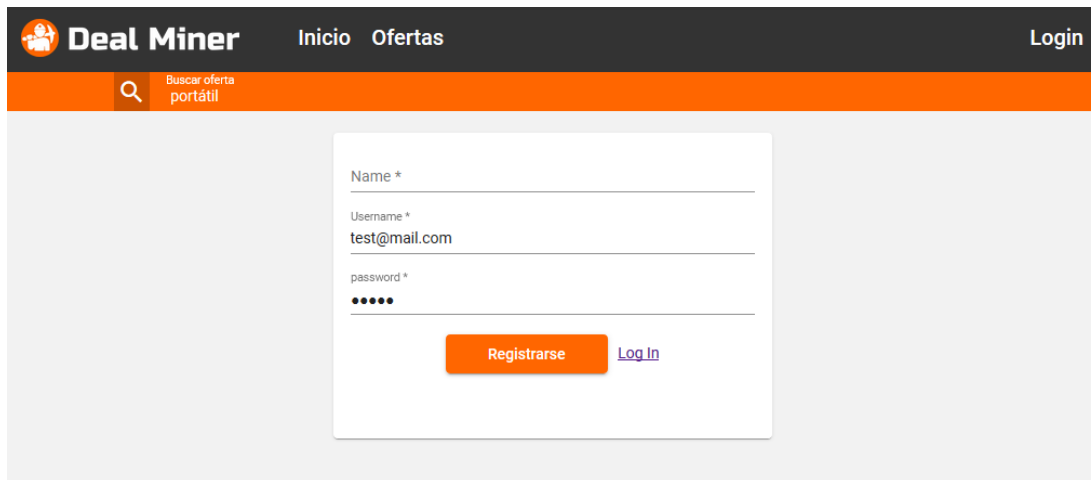
The image shows a screenshot of the Deal Miner web application. At the top, there is a dark grey header with the Deal Miner logo on the left, the text 'Inicio Ofertas' in the center, and 'Login' on the right. Below the header is an orange navigation bar containing a search icon and the text 'Buscar oferta portátil'. The main content area is light grey and features a white registration and login form. The form has three input fields: 'Name *', 'Username *' (with 'test@mail.com' entered), and 'password *' (with five dots). Below the fields are two buttons: an orange 'Registrarse' button and a blue 'Log In' link.

Figura 5.9: Registro y autenticación de usuarios en la aplicación.

5.7.2 Listado de ofertas

El componente de listado de ofertas (ver Figura 5.11) permite a los usuarios visualizar de forma cómoda las últimas ofertas disponibles. Para cada oferta se muestran datos básicos como una imagen, nombre del producto, precio o tienda. Haciendo click en una oferta se accede a la vista de detalles.

Para mejorar la visualización de datos en este componente se han desarrollado varias soluciones. Por un lado la implementación de un *sticky header* que permite acceder a la cabecera de la aplicación desde cualquier punto. Esta cabecera contiene los principales elementos de navegación de la aplicación como es el acceso a los datos de usuario (conexión y autenticación), las opciones de menú para navegar entre las distintas categorías de ofertas, la barra de búsqueda, o el logo de la aplicación. Además la cabecera reduce su altura automáticamente cuando el usuario hace *scroll down* en la página por debajo de un umbral establecido de 500 píxeles, escondiendo la barra de búsqueda mejorando así la visualización de las ofertas. En caso de que el usuario haga *scroll up*, el *sticky header* volverá a su tamaño original. Por otro lado se ha implementado una solución de paginación en cliente para mejorar la carga y visualización del listado de ofertas. Además de lo anterior, las ofertas son mostradas en modo *masonry*, forma de muro de ladrillos, para optimizar el espacio disponible y el *look and feel* de la aplicación.

5.7.3 Detalles de oferta y producto

Los detalles de una oferta y su producto (ver Figura 5.12) son mostrados en un componente independiente que permite visualizar la imagen del producto en un tamaño más adecuado así como información extra como puede ser su descripción, coste de envío o código de descuento.

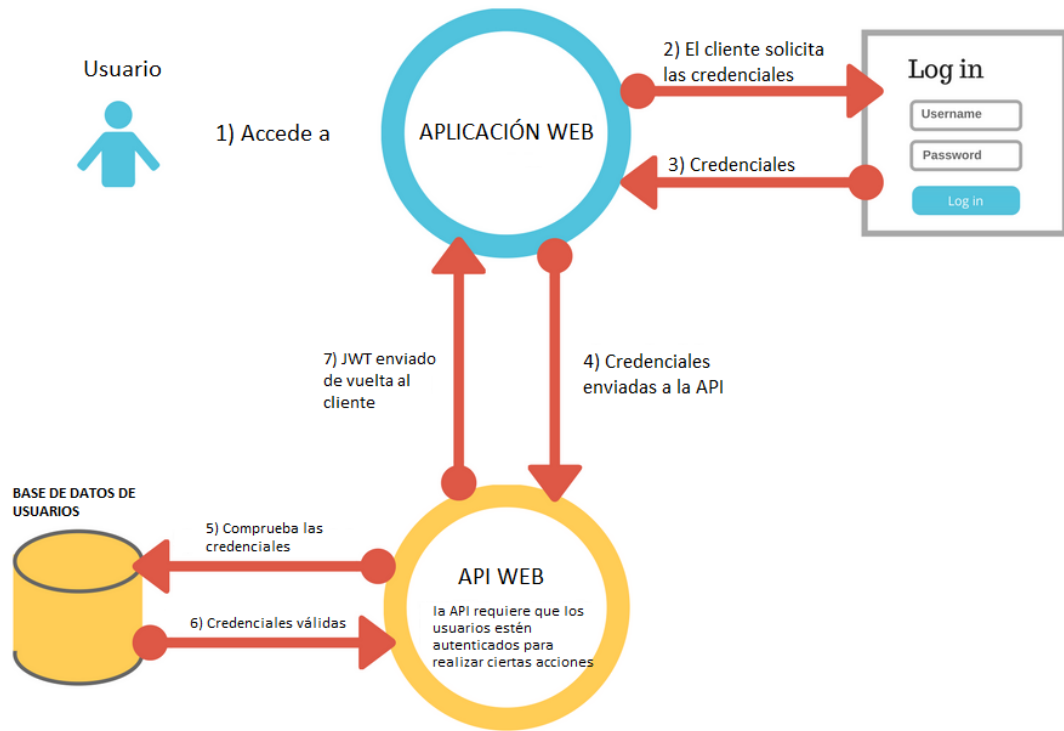


Figura 5.10: Diagrama autenticación con JWT.

Cada oferta es accedida a través de una URL única servida por el router proporcionado por Angular a partir del id de la oferta en la base de datos. De este modo, el funcionamiento de la aplicación es similar al de una web tradicional, permitiendo el acceso a las ofertas directamente desde cualquier enlace externo o buscador de enlaces.

5.7.4 Búsqueda de ofertas

El componente de búsqueda (ver Figura 5.13) forma parte de la cabecera de la aplicación y permite a los usuarios realizar búsquedas de productos y ofertas. Ha sido implementado haciendo uso de la directiva `matAutocomplete` proporcionada por la biblioteca Angular Material y que, en conjunción con el uso de observables, permite realizar búsqueda con auto-completado, mejorando la experiencia de usuario.

5.7.5 Filtrado de ofertas por categoría

Las ofertas pueden ser filtradas y mostradas según la categoría a la que pertenezcan (ver Figura 5.14) de forma que sea más sencillo encontrar la oferta adecuada. Cada oferta está incluida en una única categoría que es mostrada tanto en el listado de ofertas como en el

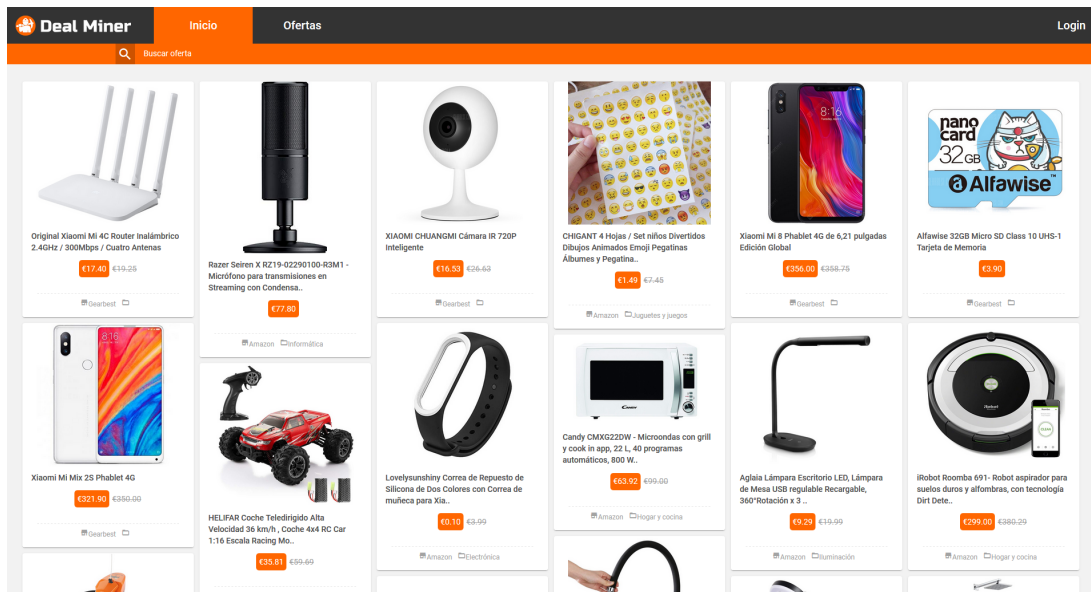


Figura 5.11: Listado de ofertas en la aplicación web.

detalle de ofertas siendo posible hacer click en la categoría para acceder al filtrado y encontrar ofertas de la misma categoría. Las ofertas de todas las tiendas son categorizadas siguiendo el esquema de la tienda Amazon para de esta forma homogeneizar la clasificación como se ha explicado en el apartado 5.5.2.

5.8 Ciclo de desarrollo

En este apartado se describe el ciclo de desarrollo, detallando las historias de usuario y las tareas que se han realizado en cada sprint, así como el ritmo al que han sido completadas (ver Figura 5.15).

5.8.1 Sprint 0

Este fue el sprint inicial, comprendido desde el 19 de febrero de 2018 al 4 de marzo de 2018, en el que se realizó la toma de requisitos para el proyecto y se llevó a cabo la creación del arquetipo inicial de la aplicación web, la API y el Bot de Telegram (ver Figura 5.16).

#52 Análisis de requisitos

En esta historia de usuario se completaron 15 *story points* y en ella se realizó la toma inicial de los requisitos llevada a cabo mediante la creación de historias de usuario tal y como es sugerido en la metodología de desarrollo Scrum. Se realizó una primera reunión entre los distintos stakeholders del proyecto para determinar las funcionalidades y características básicas

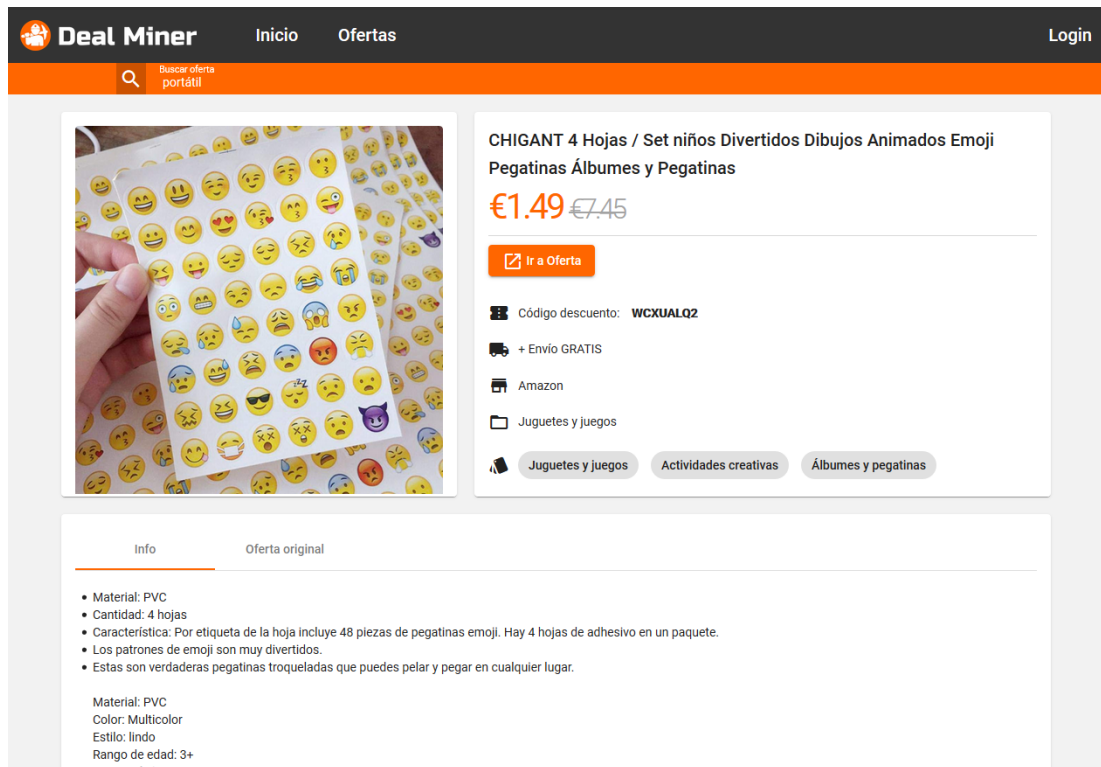


Figura 5.12: Detalles de una oferta y su producto.

del proyecto que permitirían el comienzo del desarrollo, creando así el product backlog en la aplicación Taiga.

#1 Creación arquetipo inicial de la aplicación

Estimada en 14 story points, esta historia de usuario permitió al alumno una primera toma de contacto con las distintas tecnologías escogidas para la creación de los servicios del sistema. Se instaló el entorno de desarrollo mediante el uso de las herramientas vagrant, virtualBox para la instalación del contenedor Homestead que provee el framework Laravel, y que permite la creación en el entorno local de una máquina virtual ya preparada con las configuraciones necesarias para trabajar con Laravel y Elasticsearch.

Además se inició el arquetipo de la aplicación web con el framework Angular estableciendo las dependencias de las principales bibliotecas que serían usadas para la creación del frontend como son Angular Universal, Flex o Angular Material.

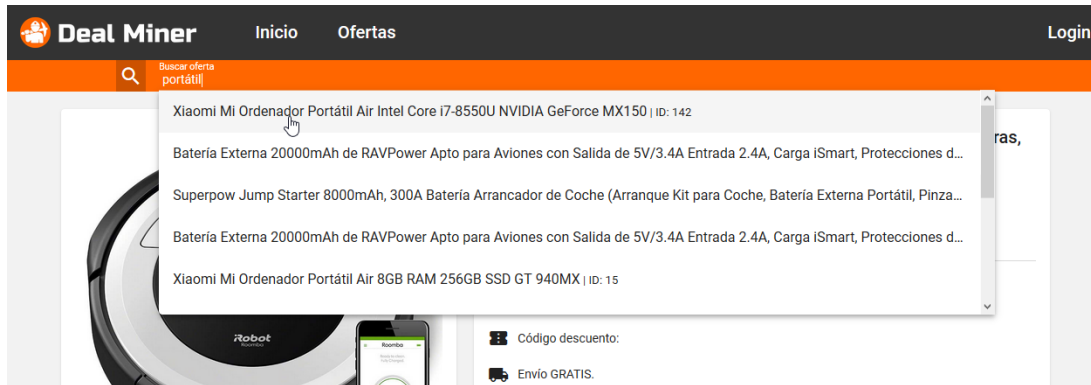


Figura 5.13: Búsqueda de ofertas en la aplicación

5.8.2 Sprint 1

En este sprint se desarrolló el Bot de Telegram, servicio encargado de minar las ofertas de los distintos canales, desde el 5 de marzo de 2018 al 18 de marzo de 2018 (ver Figura 5.17).

#10 Extraer ofertas de Telegram

Con una estimación de 29 *story points*, en esta historia de usuario se creó el servicio encargado de minar las ofertas de los canales de Telegram mediante un Bot creado con la biblioteca de PHP MadelineProto. Se creó un nuevo repositorio de Git en GitHub, se añadió la biblioteca al servidor de desarrollo, y se creó el userBot conectándolo con la *API* de Telegram. De esta manera se consiguió escuchar los canales suscritos por el usuario para su posterior tratamiento.

5.8.3 Sprint 2

El Sprint 2 fue llevado a cabo desde el 19 de marzo de 2018 hasta el 1 de abril de 2018. En él se comenzaron a desarrollar las primeras funcionalidades de la aplicación web con ofertas *mockeadas* (ver Figura 5.18).

#3 Mostrar información detallada de una oferta

Con esta historia de usuario se terminaron 13 *Story Points* y su ejecución se basó en la generación de la vista de detalles en el *frontend* así como de la creación de la petición en la *API Rest*, haciendo la llamada a la misma desde el *service* de *Offer* en la aplicación de Angular.

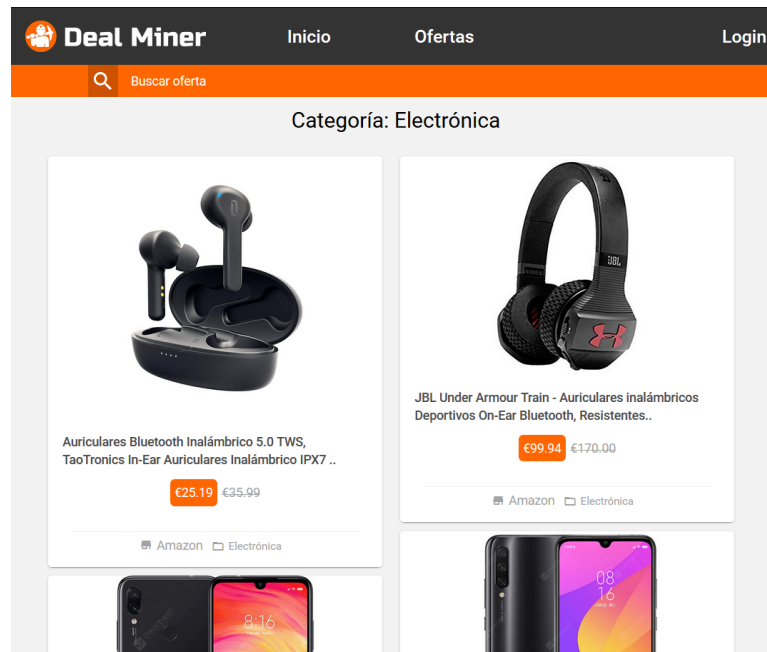


Figura 5.14: Filtrado de ofertas de electrónica.

#2 Listar últimas ofertas disponibles

Como en la anterior historia de usuario, aquí fue necesario crear las distintas partes que componen la funcionalidad para mostrar el listado de ofertas: creación de la vista, *service* y petición de la API. Además se creó el *Sticky Header* que permite un mejor aprovechamiento del espacio vertical al visualizar las ofertas. También se añadió la configuración para hacer que la aplicación fuese SSR, es decir, *renderizada* desde el servidor, y en conjunción con la creación de rutas se dotó a la aplicación *single page* de un comportamiento más cercano a las páginas web tradicionales. Con esta historia de usuario se completaron 14 Story Points.

5.8.4 Sprint 3

Durante este Sprint, que fue realizado desde el 2 de abril al 23 de abril, se llevó a cabo la funcionalidad para permitir el registro de nuevos usuarios y su autenticación en la aplicación web (ver Figura 5.19).

#5 Registro y autenticación de usuarios

Se ha implementado una biblioteca para la generación de *Json Web Tokens* en la API cada vez que recibe un usuario y contraseña válidos. Los *JWT* se mandan en cada petición y no requieren que se almacene información de la sesión del usuario en el servidor. Esta biblioteca

DEALMINER BACKLOG

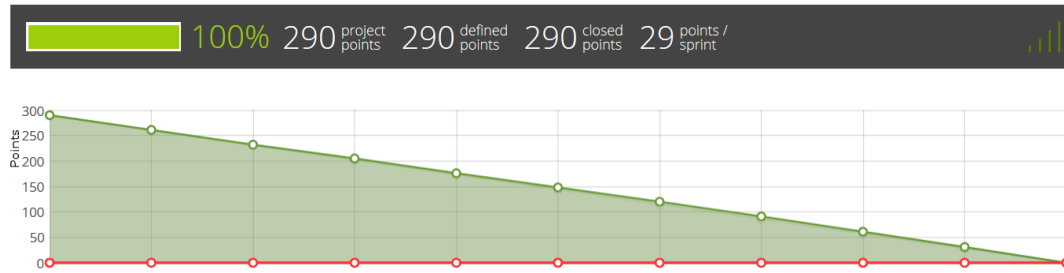


Figura 5.15: Gráfica de burn-down del proyecto.

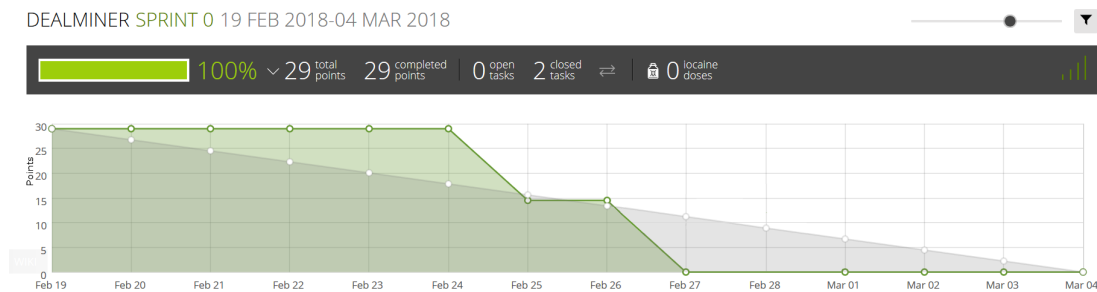


Figura 5.16: Gráfica de burn-down del sprint 0.

además genera un *middleware* que se puede usar en las rutas requeridas de la API para comprobar la autenticación de las peticiones a la API, comprobando y validando el token recibido. Por otro lado se ha implementado un sistema sencillo de roles de usuario y se ha creado un *middleware* de seguridad para comprobar la autenticación basada en roles y controlar si la petición es realizada por un usuario normal o un *admin*.

En cuanto al *frontend*, se ha creado un pequeño inyector implementado desde el service de autenticación. También se han implementado las *route guards* de Angular para comprobar el acceso a las distintas vistas del frontend según los permisos y roles de usuario (usuario no autenticado, usuario autenticado y usuario administrador), así se comprueba tanto en el frontend como en el backend. Por último se crearon las vistas de login y registro de usuarios. En esta historias de usuario se completaron 29 Story Points.

5.8.5 Sprint 4

El Sprint completado entre el 24 de abril de 2018 y el 6 de mayo de 2018 fue destinado en su totalidad al desarrollo del Text Miner (ver Figura 5.20).

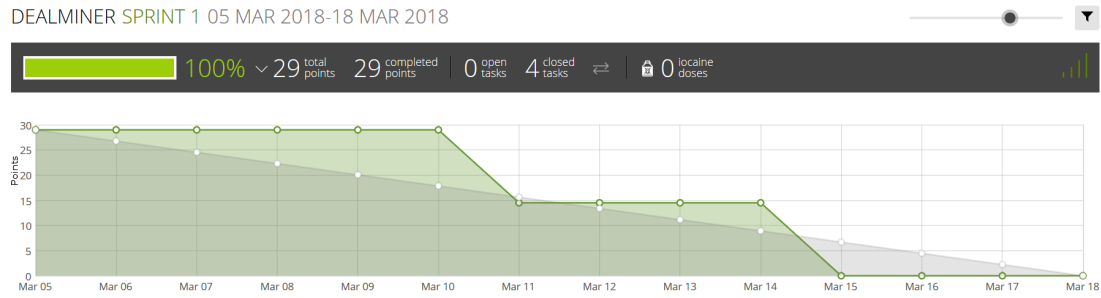


Figura 5.17: Gráfica de burn-down del sprint 1.

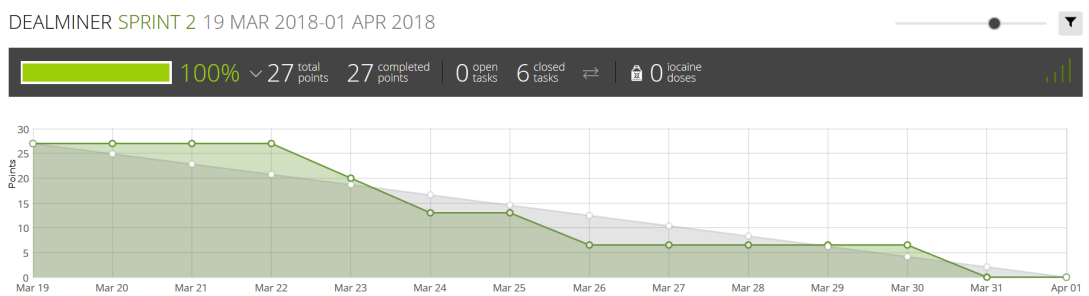


Figura 5.18: Gráfica de burn-down del sprint 2.

#18 Extraer información relevante de las ofertas de los canales de Telegram

Con 28 Story Points completados, se desarrolló la base del servicio Text Miner para extraer la información necesaria a partir del texto de las ofertas en los canales de Telegram, para así poder almacenarlas en la base de datos. Se extraen las URLs del cuerpo de la oferta mediante *regex*, y además, se ha creado una función para recorrer el camino que siga el enlace para extraer la URL final, ya que la mayoría de enlaces vienen en las ofertas con URL recortadas como por ejemplo `chx.t0/5` i.e. También se ha creado otra función para extraer el precio del cuerpo de la oferta usando `WhitespaceTokenizer` y luego utilizando una expresión regular para detectar los tokens que contienen un símbolo de € o EUR. Se ha tenido en cuenta que a veces aparece un espacio entre el precio y el símbolo de € así que se analizan los tokens anterior y posterior para comprobar si son números.

5.8.6 Sprint 5

Realizado del 07 mayo de 2018 al 20 de mayo de 2018, este sprint fue destinado al despliegue de los servicios en la nube para emular un entorno de producción real (ver Figura 5.21).

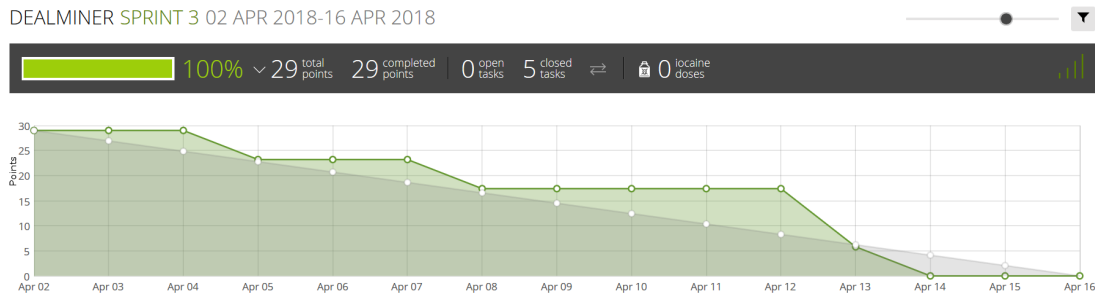


Figura 5.19: Gráfica de burn-down del sprint 3.

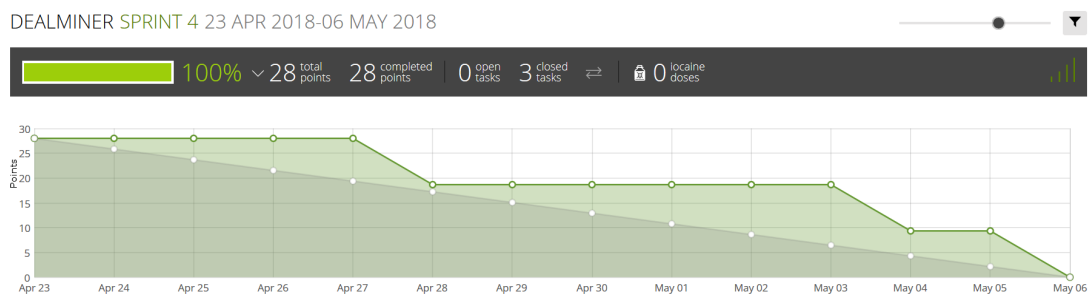


Figura 5.20: Gráfica de burn-down del sprint 4.

#45 Hacer despliegue de servicios en la nube

Se ha usado la infraestructura de Amazon AWS para el *backend* montando la API en una instancia de Amazon EC2 y la base de datos en otra instancia de Amazon RDS usando Amazon Beanstalk. Además se ha configurado un balanceador de carga para poder usar HTTPS y poder añadir más instancias en caso de que fuera necesario. El despliegue se hace de forma automática a través de Travis-CI cuando se realiza un *push* a una rama del proyecto en GitHub.

Para el frontend, la aplicación en Angular se ha subido a una instancia de Google cloud usando App Engine. Es una instancia flexible con Node.js que se puede aumentar de número y recursos de forma automática. Al igual que la API, el despliegue se hace de forma automática a través de travis-CI. Se han completado 17 Story Points.

#8 Búsqueda de ofertas

Historia de usuario finalizada en 13 Story Points, durante la cual se ha desarrollado y configurado la búsqueda de ofertas. Por un lado se ha configurado una instancia de Elasticsearch para almacenar las ofertas minadas, y por otro se ha implementado un componente y servicio en Angular para realizar las búsquedas y su visualización.

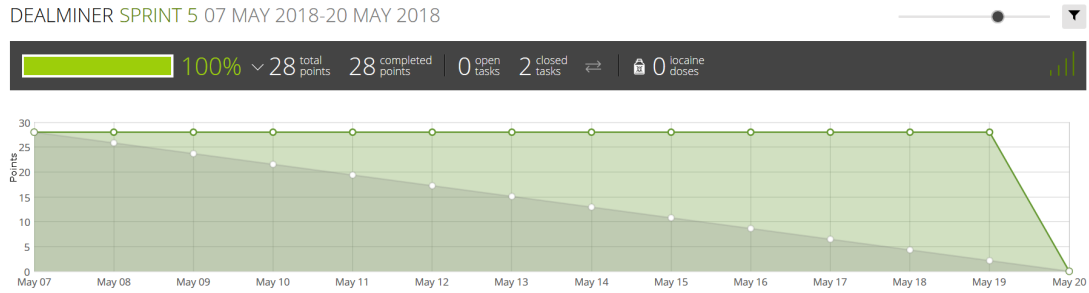


Figura 5.21: Gráfica de burn-down del sprint 5.

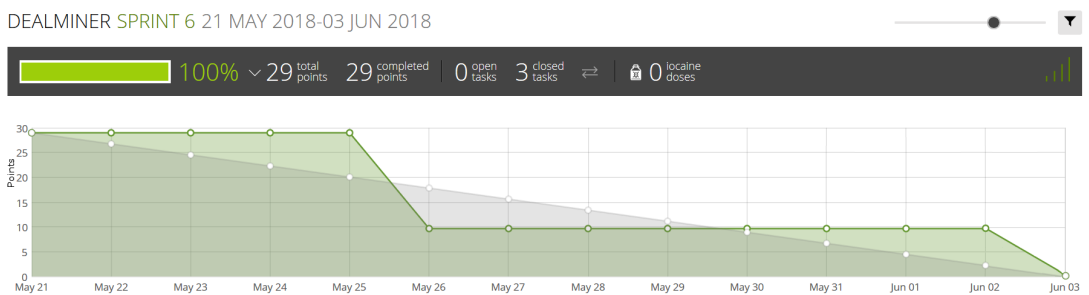


Figura 5.22: Gráfica de burn-down del sprint 6.

5.8.7 Sprint 6

Sprint finalizado del 21 de mayo al 3 de junio de 2018 en el que se ha avanzado el servicio Text Miner para obtener la información necesaria de las ofertas (ver Figura 5.22).

#12 Hacer web scraping a las tiendas relacionadas con las ofertas

Con esta historia de usuario, estimada en 29 Story Points, se ha realizado el scraping a las webs de Amazon y Gearbest. Se ha intentado hacerlo con Aliexpress también pero implementa medidas anti-bots por lo que ha sido descartada. Se obtiene la información como el título del producto, el precio, coste de envío, descripción o url de imagen avanzando así el desarrollo del servicio Text Miner.

5.8.8 Sprint 7

Durante este sprint llevado a cabo del 18 junio al 1 de julio de 2018 se ha completado la creación del Web Crawler destinado a obtener un gran conjunto de productos de la web de Amazon (ver Figura 5.23).

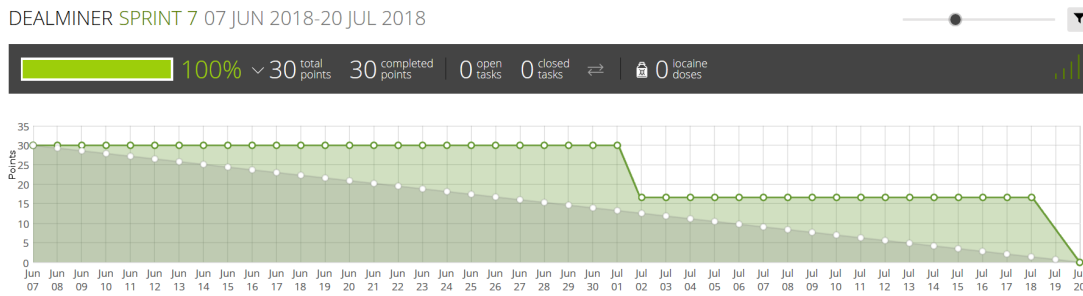


Figura 5.23: Gráfica de burn-down del sprint 7.

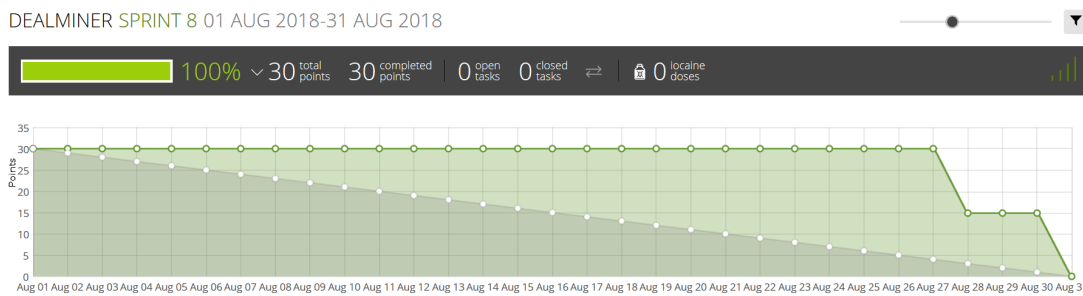


Figura 5.24: Gráfica de burn-down del sprint 8.

#6 Creación del Web Crawler

Esta historia de usuario ha llevado 28 Story Points, y en ella se ha implementado por completo el Web Crawler y se ha creado un enorme conjunto de datos de productos de Amazon de más de 10.000 elementos para posteriormente ser empleado en la clasificación por categorías de las ofertas que no son de Amazon.

5.8.9 Sprint 8

Realizado entre el 01 de agosto de 2018 y el 31 de agosto de 2018. Se han implementado las funcionalidades para detectar ofertas repetidas así como la paginación de ofertas en el índice de la aplicación web (ver Figura 5.24).

#19 Detectar duplicados en ofertas

Completando 15 Story Points, se ha implementado una función en el Text Miner capaz de detectar las ofertas duplicadas en distintos canales, teniendo en cuenta la diferencia de tiendas, días y precios de oferta.

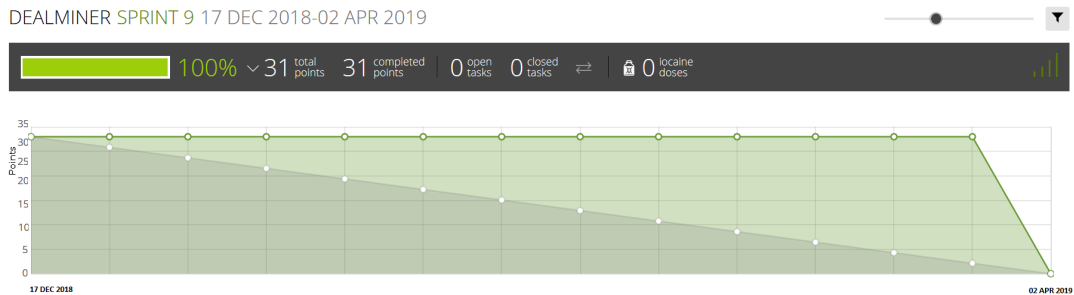


Figura 5.25: Gráfica de burn-down del sprint 9.

#40 Añadir paginación a lista de ofertas

También con una duración de 15 Story Points, se ha añadido un nuevo componente al índice de la aplicación web para paginar los resultados y mejorar la visualización de datos.

5.8.10 Sprint 9

Último sprint completado entre el 17 de diciembre y el 2 de abril de 2019 en el que se ha completado el servicio del Text Miner para clasificar las ofertas y se ha creado la memoria final del proyecto. Este sprint se ha alargado especialmente en el tiempo debido a las dificultades del alumno para destinar horas a la finalización del proyecto (ver Figura 5.25).

#9 Clasificar ofertas por categoría

Historia de usuario de 18 Story Points destinada a la implementación de la función encargada de clasificar por categorías las ofertas de tiendas que no son Amazon e incluirlas en su estructura de categorías. Se ha realizado usando Elasticsearch y su motor de búsqueda para realizar clasificación de texto. Además se ha añadido en la API y en la aplicación web un filtro por categoría para poder acceder a las ofertas clasificadas y agrupadas.

#53 Borrador memoria del proyecto

Con 13 Story Points, la historia de usuario final estuvo destinada a la creación del borrador de la memoria final del proyecto realizada en \LaTeX en la plataforma Overleaf⁵.

5.9 Despliegue

En esta sección se detalla el proceso de despliegue de la aplicación en un entorno de producción (ver Figura 5.26).

⁵<https://overleaf.com>

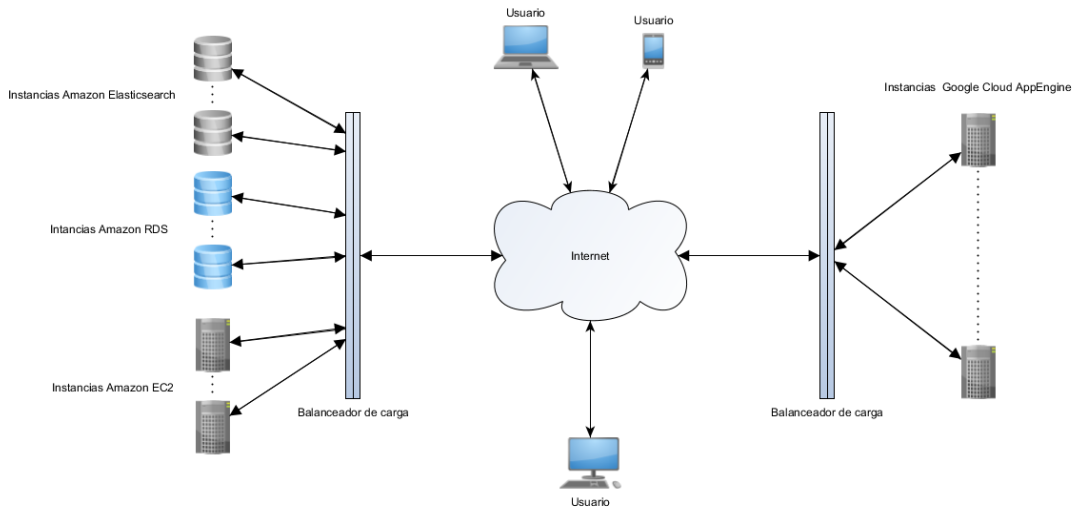


Figura 5.26: Diagrama de despliegue de la aplicación.

Dada la naturaleza del sistema creado siguiendo una arquitectura basada en servicios se ha planificado un despliegue en producción intentando separar cada servicio en distintas instancias aumentando así el desacoplamiento de los mismos. Se ha decidido hacer uso de dos de las plataformas de servicios en la nube más importantes del mundo como son Amazon Web Services⁶ y Google Cloud⁷. De esta forma se han podido aprovechar las ofertas iniciales y créditos de prueba de ambas plataformas para optimizar el gasto en infraestructura para el proyecto y comprobar las diferencias entre ambas.

5.9.1 Amazon Web Services

La nube de Amazon ofrece casi todas las funciones de la industria de la computación. Sus servicios en la nube permiten acceder fácilmente a la máxima potencia de cómputo, el almacenamiento de datos o cualquier otra funcionalidad necesaria para los desarrolladores de aplicaciones. Por ejemplo ofrecen herramientas para desarrolladores, herramientas de administración, servicios móviles y servicios de aplicaciones. La base de datos MySQL, el motor de búsqueda de Elasticsearch y la API implementada en Laravel han sido puestos en producción usando instancias de AWS.

Amazon RDS

Con Amazon Relational Database Service (Amazon RDS), es sencillo configurar, utilizar y escalar una base de datos relacional en la nube. El servicio suministra capacidad rentable y

⁶<https://aws.amazon.com>

⁷<https://cloud.google.com>

escalable al mismo tiempo que automatiza las arduas tareas administrativas, como el aprovisionamiento de hardware, la configuración de bases de datos, la implementación de parches y la creación de copias de seguridad. Por ello, se ha escogido para este proyecto el uso de una instancia *db.t2.micro* para la realización de las pruebas necesarias, siendo posible aumentar el número de instancias disponibles de forma rápida y automática según las necesidades de recursos que el servicio pudiera necesitar. Este tipo de instancias tienen un coste de 9.27€ al mes, siendo la clase más baja en recursos disponibles que ofrece AWS. Tiene una sola CPU y 1 GB de memoria RAM y el motor de bases de datos MySQL.

Amazon EC2

Para la API Rest de Laravel se ha hecho uso de una instancia *t2.micro* de EC2. Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable. Está diseñado para simplificar el uso de la informática en la nube a escala web para los desarrolladores. De esta forma, se ha implementado un balanceador de carga que permite además de filtrar las peticiones HTTPS redirigir de forma automática el tráfico a las distintas instancias configuradas. Las instancias T2 son instancias de rendimiento ampliable que proporcionan un nivel base de rendimiento de la CPU con la posibilidad de ampliarse por encima del nivel básico. Tienen un procesador escalable Intel de hasta 3,3 GHz y 1 GB de RAM, un sistema operativo con Linux y un rendimiento de red declarado por Amazon de bajo a moderado. Su precio es de 5,73 € al mes.

Amazon ES

Amazon Elasticsearch Service es un servicio completamente administrado que facilita la implementación, seguridad y operación de Elasticsearch a escala sin tiempo de inactividad. El servicio ofrece las API de Elasticsearch de código abierto, el complemento Kibana administrado e integraciones con Logstash y otros servicios de AWS, lo que le permite incorporar datos de forma segura de cualquier fuente y buscarlos, analizarlos y visualizarlos en tiempo real. Para la implementación del servicio de búsqueda del proyecto se ha hecho uso de una instancia *t2.small.elasticsearch*. Este tipo de instancia posee una única CPU virtual, 2 GB de memoria RAM y 10GB de almacenamiento. Su coste es de 24,48€ mensuales.

5.9.2 Google Cloud

Google Cloud Platform ofrece una gran variedad de servicios para desarrolladores como App Engine. Esta aplicación permite que un desarrollador cree aplicaciones sin tener que lidiar con configuraciones tediosas del servidor. Es una solución totalmente gestionada para desarrollar aplicaciones de forma rápida. Además, puede alcanzar un alto nivel de almacena-


```
runtime: nodejs
env: flex

manual_scaling:
  instances: 1
resources:
  cpu: 1
  memory_gb: 0.5
  disk_size_gb: 10
```

Figura 5.27: Configuración fichero app.yml.

miento, redes y bases de datos. Para este proyecto se ha elegido Google Cloud y su sistema App Engine para la puesta en producción de la aplicación web de Angular.

En la Figura 5.27 puede observarse como se ha configurado el despliegue en App Engine a partir de un fichero app.yml introducido en la carpeta raíz del repositorio. De esta manera es sencillo configurar el entorno y la tecnología, así como los recursos destinados a la aplicación, pudiendo especificar el número de instancias y el escalado que pueda realizar el balanceador de carga. Este tipo de instancias flexibles tienen un coste aproximado de 33,12€ por CPU al mes, 4,54€ al mes por GB de memoria usado y 25,2€ al mes por GB de memoria de disco usada. Se ha escogido este tipo de instancia flexible en vez de instancias estáticas más baratas porque en el momento del desarrollo de este proyecto no era posible usar instancias estáticas con node.js en Google Cloud.

5.9.3 Entrega y despliegue continuo

La entrega continua es una extensión de la integración continua [19] que permite a los desarrolladores introducir nuevos cambios en las aplicaciones de una manera sostenible, conveniente y rápida. El despliegue continuo va un paso más allá que la entrega continua. Mediante esta práctica, los cambios que pasan por todas las fases de tu canal de producción se publican para tus clientes. No hay intervención humana y solo una prueba fallida evitará implementar un nuevo cambio en producción.

Gracias al uso de las plataformas de Google y Amazon en conjunción con el sistema de versionado Git y la herramienta de integración continua travis-CI se ha podido implementar en este proyecto un sistema de despliegue continuo para los principales servicios como son la aplicación web en Angular o la API Rest de Laravel. De este modo es posible automatizar todo el proceso de puesta en producción de estos servicios desde que se realiza un envío al repositorio Github. Fue necesario conectar el repositorio con la aplicación travis-CI (ver Figura 5.28) mediante la configuración de un fichero *travis.yml* en el que se especifican las distintas propiedades necesarias para realizar el *build* y el despliegue del servicio. Este fichero

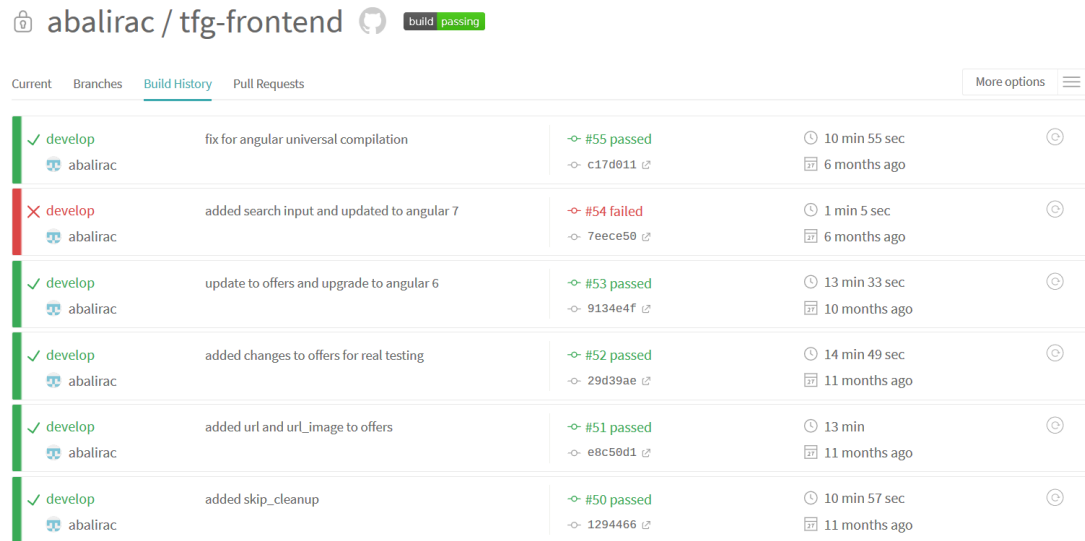


Figura 5.28: Dashboard de travis-CI.

debe subirse en la carpeta raíz del repositorio y en él se configuran parámetros como los *scripts* que deben ser ejecutados para la construcción del software, lenguaje y tecnologías usados, o configuración de la plataforma de despliegue. Una vez configurado correctamente, la aplicación travis-CI se encarga de realizar todo el trabajo de forma automática, realizando el *build*, *test* y despliegue del servicio en caso de que todo se haya ejecutado correctamente.

Conclusiones

EL proyecto ha sido una gran oportunidad para el alumno a la hora de demostrar todas las habilidades y conocimientos adquiridos en la titulación. Por un lado los conocimientos básicos de programación, algoritmos, bases de datos, gestión de proyectos o diseño de software aprendidos en la titulación en las asignaturas obligatorias han permitido establecer una base adecuada para el desarrollo y puesta a punto de los distintos servicios. Por otro lado, gracias a los conocimientos adquiridos durante la mención de *Enxeñería do Software* de arquitectura de software, herramientas y metodologías de desarrollo se ha podido realizar un proyecto con una arquitectura bien planificada y haciendo uso de las herramientas y tecnologías más avanzadas y adecuadas para las necesidades del mismo.

Gracias a esto se han podido cumplir los objetivos principales del proyecto, creando un sistema con una arquitectura basada en servicios capaz de mostrar a usuarios finales ofertas de productos de tiendas en línea de forma automática. Gracias a la realización del proyecto se ha podido ver la dificultad de llevar a cabo de forma práctica un sistema de gran complejidad y se ha conseguido aumentar los conocimientos sobre nuevas tecnologías como Angular o Laravel, así como aprender sobre nuevas disciplinas como la minería de datos, *web scraping* o el análisis de texto. Además se ha profundizado el conocimiento de cómo funcionan las nuevas plataformas en la nube como Amazon Web Services o Google Cloud y la infraestructura necesaria para la interconexión de distintos servicios.

6.1 Trabajo futuro

Aunque se han cumplido los principales objetivos a la hora de implementar el proyecto, se han identificado posibles líneas de trabajo futuro a la hora de mejorar el sistema y añadir nuevas funcionalidades:

- **Votar ofertas:** como una funcionalidad extra se podría permitir a los usuarios registrados votar positiva o negativamente las ofertas, pudiendo así ordenarlas y clasificarlas

según la valía para los propios usuarios.

- **Comentar ofertas:** también crearía un valor añadido la posibilidad de permitir a los usuarios realizar comentarios de las distintas ofertas.
- **Subscripción a productos:** una funcionalidad interesante sería la posibilidad de permitir a los usuarios suscribirse a los distintos productos que aparecen en las ofertas de la aplicación dando la opción de recibir notificaciones en distintos medios como el correo electrónico o la propia web cuando haya nuevas ofertas de dicho producto.
- **Añadir más tiendas:** aunque con las webs de Amazon y Gearbest se ha cubierto un alto porcentaje de las ofertas de los canales de Telegram, sería muy útil añadir en un futuro la capacidad de minar otras tiendas como Aliexpress, Bangood etc.
- **Log in desde redes sociales:** aunque ya se permite registrarse con un usuario y contraseña, sería interesante aprovechar el sistema de JWT para permitir el log in desde cuenta de redes sociales como Facebook, Instagram o Youtube.
- **Pruebas automáticas:** añadir pruebas unitarias y de integración de los distintos componentes en todos los servicios del sistema. Realizar pruebas de aceptación en la aplicación web para emular el comportamiento del sistema con la interacción de usuarios reales. Además sería conveniente añadir pruebas de sistemas para poder comprobar el correcto funcionamiento de la interconexión de los distintos servicios.

Apéndice

Glosario de acrónimos

API Application Programming Interface: conjunto de funciones o procedimientos englobados en una biblioteca de software a modo de capa de abstracción, para ser usados por otras aplicaciones.

DOM Document Object Model: es una interfaz que permite representar documentos HTML o XML mediante un estándar de objetos.

HTTP Hypertext Transfer Protocol: es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

NoSQL Not Only SQL: clase de sistemas de gestión de bases de datos que difieren del modelo relacional y cuyo principal objetivo es proporcionar escalabilidad horizontal.

ORM Object-Relational Mapping: el mapeo objeto-relacional es una herramienta de programación que establece una correspondencia entre clases y objetos de un lenguaje de programación orientado a objetos con tablas y filas de una base de datos relacional siguiendo el patrón arquitectónico *active record*.

JSON JavaScript Object Notation: es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

REST Representational State Transfer: es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

JWT Json Web Tokens: es un estándar abierto basado en JSON para la creación de tokens de acceso que permiten la propagación de identidad y privilegios.

SPA *Single-page Application*: es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio.

CRUD *Create, Read, Update and Delete*: se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

SSR *Server Side Rendering*: técnica que se basa en la posibilidad de poder renderizar el HTML de los componentes de una aplicación web en cadenas de texto en el servidor en vez de hacerlo en el cliente.

JS *JavaScript*: es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase para páginas web.

CSS *Cascading Style Sheets*: es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

HTML *HyperText Markup Language*: es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet.

PHP *PHP: Hypertext Preprocessor*: es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

IDE *Integrated Development Environment*: es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

SOAP *Simple Object Access Protocol*: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

DSL *Domain Specific Language*: es un lenguaje de programación o especificación dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.

URL *Uniform Resource Locator*: es un identificador de recursos uniforme cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. Están formados por una secuencia de caracteres de acuerdo a un formato modular y estándar que designa recursos en una red como, por ejemplo, Internet.

AJAX *Asynchronous JavaScript And XML*: es una técnica de desarrollo web para crear aplicaciones interactivas.

Glosario de términos

Layout Cuadrícula imaginaria que divide en espacios o campos la página que se diseña para facilitar la distribución de elementos como textos ó gráficos en la misma.

Framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Javascript Lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Typescript Lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.

ECMAScript Especificación de lenguaje de programación publicada por ECMA International.

Flexbox CSS Modelo unidimensional de layout, y método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.

Debugging Proceso para identificar y corregir errores de programación.

Testing Investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad de un producto.

Kanban Método para gestionar el trabajo intelectual, con énfasis en la entrega justo a tiempo, mientras no se sobrecarguen los miembros del equipo.

node.js Entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript.

Bibliografía

- [1] G. Cid, “Los ‘reyes del chollo’ en españa: estos tres veinteañeros triunfan cazando ofertas online [online],” *elconfidencial.com*, 01 2019, disponible en: https://www.elconfidencial.com/tecnologia/2019-01-11/chollometro-reyesdelchollo-pepper-jovenes-espanoles_1750270/. [Accedido: 31-ago-2019].
- [2] J. Kroeze, M. Mathee, and T. Bothma, “Differentiating data-and text-mining terminology,” *researchgate.net*, pp. 93–101, 01 2003.
- [3] S. M. ChengXiang Zhai, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Morgan & Claypool, 06 2016.
- [4] Duke University Libraries, “Introduction to text analysis: About text analysis [online],” disponible en: <https://guides.library.duke.edu/c.php?g=289707&p=1930854>. [Accedido: 20-mar-2019].
- [5] P. E. Maarten Truyens, *Computer Law & Security Review*. Elsevier, 04 2014.
- [6] M. Watson, *Using Web Scraping to Create Semantic Relations*. Springer US, 01 2009.
- [7] A. Najork, Marc y Heydon, *High-Performance Web Crawling*. Boston, MA: Springer US, 2002, pp. 25–45.
- [8] M. A. Kausar, V. S. Dhaka, and S. K. Singh, “Web crawler: A review,” 2013.
- [9] Telegram, “Telegram apis [online],” disponible en: <https://core.telegram.org>. [Accedido: 01-jun-2019].
- [10] Oracle, “Top 10 reasons to choose mysql for web - based applications [online],” disponible en: <http://www.oracle.com/us/products/mysql/mysql-wp-top10-webbased-apps-461054.pdf>. [Accedido: 16-sep-2018].

- [11] The PHP Group, “Introducción a curl [online],” disponible en: <http://php.net/manual/es/intro.curl.php>. [Accedido: 20-oct-2018].
- [12] International Scrum Institute, “Scrum roles - the scrum team [online],” disponible en: https://www.scrum-institute.org/Scrum_Roles_The_Scrum_Team.php. [Accedido: 03-nov-2018].
- [13] Vikas Hazrati, “Do story points relate to complexity or time? [online],” disponible en: <https://www.infoq.com/news/2010/07/story-points-complexity-effort>. [Accedido: 06-nov-2018].
- [14] Bruce Sun, “Arquitectura multinivel para la construcción de servicios web restful [online],” disponible en: <https://www.ibm.com/developerworks/ssa/library/wa-aj-multitier/index.html>. [Accedido: 03-nov-2018].
- [15] Chakray, “¿cuáles son las ventajas de una api rest? [online],” disponible en: <https://www.chakray.com/cuales-son-las-ventajas-de-una-api-rest>. [Accedido: 06-ago-2018].
- [16] Saskia Vola, “Text classification made easy with elasticsearch [online],” disponible en: <https://www.elastic.co/blog/text-classification-made-easy-with-elasticsearch>. [Accedido: 20-dic-2018].
- [17] angular.io, “Master/detail components [online],” disponible en: <https://angular.io/tutorial/toh-pt3>. [Accedido: 21-dic-2018].
- [18] Chris Baker, “Single page applications (spa) and the seo problem [online],” disponible en: <https://adkgroup.com/blog/single-page-applications-spa-and-seo-problem>. [Accedido: 21-dic-2018].
- [19] Sten Pittet, “Comparación de integración continua, entrega continua e implementación continua [online],” disponible en: <https://es.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>. [Accedido: 16-abr-2019].