

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

**DESARROLLO DE UN SISTEMA DE RECOGIDA Y
CATEGORIZACIÓN DE DATOS DE RECETAS**

GONZALO GALLASTEGUI PEÑALBA

Tutor: ALEJANDRO BELLOGÍN KOUKI

Ponente: PABLO CASTELLS AZPILICUETA

MAYO 2015

RESUMEN

La popularidad del área de la gastronomía en internet se ha visto impulsada, en parte, gracias al auge de las redes sociales y las nuevas tecnologías aplicadas a la web. Este mundo virtual se ha convertido en un medio ideal para los amantes de la gastronomía tanto a nivel profesional o aficionado, para difundir e intercambiar recetas, compartir consejos y descubrir nuevos métodos culinarios.

Concretamente, los blogs y aplicaciones web relacionadas con el sector de la cocina se han convertido en las herramientas más utilizadas para el intercambio de información, y como consecuencia de la gran cantidad de datos que fluyen a través de estas aplicaciones, la gestión de la información para los usuarios es cada vez más complicada, especialmente en el área de la búsqueda y recomendación de datos.

Este Trabajo de Fin de Grado describe y presenta una aplicación de búsqueda de contenido gastronómico, donde la información utilizada se ha obtenido usando un sistema de extracción de datos desde una web especializada, haciendo uso de tecnología basada en búsquedas de varios tipos, los cuales se explican en el presente documento.

PALABRAS CLAVE

Receta, Buscador, *Web Scraper*, Lucene, extracción de datos, filtro, ingrediente, nutriente.

ABSTRACT

The popularity in the area of gastronomy has been boosted, in part, thanks to the rise of social networks and the new technologies applied to the web. This virtual world has become in the ideal way for cuisine lovers and for professional or amateurs to broadcast and exchange recipes, share tips, and discover new cooking methods.

Specifically, blogs and web applications related to the cuisine area have become in the most important tools for dynamic data exchange, and as a consequence of the large amount of information that flows through these applications, the management of this information, for customers, is increasingly difficult, especially in the area of data mining.

This document describes and presents a gastronomy search application, where the information was obtained using a data mining system from specialized webs, using several types of searches, which are explained below.

KEYWORDS

Recipe, Searcher, *Web Scraper*, Lucene, Data mining, filter, ingredient, nutrient.

ÍNDICE DE CONTENIDOS

1.	Introducción.....	1
1.1.	Motivación.....	1
1.2.	Objetivos	2
1.3.	Ciclo de vida.....	3
1.4.	Estructura del documento	4
2.	Estado del arte.....	7
2.1.	Estudio del contexto actual del sector	7
2.1.1.	Portal web AllRecipes	8
2.1.2.	Portal web SimplyRecipes.....	9
2.1.3.	Portal web Wikicocina	10
2.1.4.	Conclusiones del estudio	10
2.2.	Estudio de alternativas tecnológicas.....	11
2.2.1.	Extracción de la información	11
2.2.2.	Tratamiento de datos e indexación	13
2.2.3.	Motor de base de datos.....	15
2.2.4.	Interacción con el usuario.....	17
2.3.	Tecnologías a utilizar	19
3.	Análisis de requisitos.....	21
3.1.	Requisitos funcionales.....	21
3.1.1.	Requisitos funcionales: proyecto <i>Web Scraper</i>	22
3.1.2.	Requisitos funcionales: proyecto Buscador	24
3.2.	Requisitos no funcionales	25
3.2.1.	Requisitos no funcionales: proyecto <i>Web Scraper</i>	26
3.2.2.	Requisitos no funcionales: proyecto Buscador	26
4.	Diseño del software.....	29
4.1.	Diseño general	29
4.1.1.	Diseño modular del proyecto <i>Web Scraper</i>	31
4.1.2.	Diseño modular del proyecto Buscador	32
4.2.	Diseño de la estructura de datos	33
4.2.1.	Tabla Recipe.....	34
4.2.2.	Tabla Direction	35
4.2.3.	Tabla Review	35
4.2.4.	Tabla Ingredient.....	36
4.2.5.	Tabla Nutrient.....	36
4.2.6.	Tabla Recipe_Nutrition.....	36

5.	Implementación.....	37
5.1.	Implementación del proyecto <i>Web Scraper</i>	37
5.1.1.	Explicación de la implementación modular.....	37
5.1.2.	Explicación de los procesos y algoritmos utilizados.....	39
5.2.	Implementación del proyecto Buscador.....	40
5.2.1.	Explicación de la implementación modular.....	41
5.2.2.	Explicación de los procesos y algoritmos implementados.....	45
6.	Pruebas.....	49
6.1.	Pruebas sobre el proyecto <i>Web Scraper</i>	50
6.1.1.	Pruebas unitarias: Proyecto <i>Web Scraper</i>	50
6.1.2.	Pruebas de integración: Proyecto <i>Web Scraper</i>	51
6.1.3.	Pruebas de sistema: Proyecto <i>Web Scraper</i>	52
6.1.4.	Pruebas de validación: Proyecto <i>Web Scraper</i>	52
6.1.5.	Pruebas de aceptación: Proyecto <i>Web Scraper</i>	52
6.2.	Pruebas sobre el proyecto Buscador.....	53
6.2.1.	Pruebas unitarias: Proyecto Buscador.....	53
6.2.2.	Pruebas de integración: Proyecto Buscador.....	54
6.2.3.	Pruebas de sistema: Proyecto Buscador.....	54
6.2.4.	Pruebas de validación: Proyecto Buscador.....	54
6.2.5.	Pruebas de aceptación: Proyecto Buscador.....	55
7.	Resultados.....	57
7.1.	Almacenamiento de la información.....	57
7.1.1.	Objeto de base de datos.....	57
7.1.2.	Índice creado a partir de la base de datos.....	58
7.2.	Aplicación de interacción con el usuario.....	59
8.	Conclusiones y trabajo futuro.....	61
9.	Referencias.....	63
A.	Anexo: Tipos de búsquedas implementadas.....	A
B.	Anexo: Casos de éxito.....	C
C.	Anexo: Resultados gráficos de la encuesta.....	I

ÍNDICE DE FIGURAS Y TABLAS

ILUSTRACIÓN 1: MODELO DE CICLO DE VIDA.....	4
ILUSTRACIÓN 2: SITIO WEB ALLRECIPES	8
ILUSTRACIÓN 3: SITIO WEB SIMPLYRECIPES	9
ILUSTRACIÓN 4: SITIO WEB WIKICOCINA	10
ILUSTRACIÓN 5: ESTUDIO DE LAS CUALIDADES DE LAS PÁGINAS	10
ILUSTRACIÓN 6: GRADO DE UTILIZACIÓN DE LOS MOTORES DE BASE DE DATOS.....	15
ILUSTRACIÓN 7: ESQUEMA DE INTEGRACIÓN DE AMBOS SISTEMAS.....	30
ILUSTRACIÓN 8: DISEÑO MODULAR DE LA APLICACIÓN <i>WEB SCRAPER</i>	31
ILUSTRACIÓN 9: DISEÑO MODULAR DE LA APLICACIÓN BUSCADOR	32
ILUSTRACIÓN 10: IMPLEMENTACIÓN MODULAR DEL PROYECTO <i>WEB SCRAPER</i>	37
ILUSTRACIÓN 11: CLASES DE LAS ENTIDADES COMUNES DEL PROYECTO.....	38
ILUSTRACIÓN 12: PROCESO DE EXTRACCIÓN DE LA INFORMACIÓN.....	39
ILUSTRACIÓN 13: ALGORITMO DE EXTRACCIÓN DE INFORMACIÓN.....	40
ILUSTRACIÓN 14: PATRÓN DE IMPLEMENTACIÓN DEL PROYECTO BUSCADOR.....	41
ILUSTRACIÓN 15: PATRÓN DE IMPLEMENTACIÓN DEL MODELO.....	42
ILUSTRACIÓN 16: ACCESO A LA INFORMACIÓN.....	45
ILUSTRACIÓN 17: PROCESO DE CREACIÓN Y ACTUALIZACIÓN DEL ÍNDICE.....	46
ILUSTRACIÓN 18: TOTAL DE RECETAS OBTENIDAS.....	57
ILUSTRACIÓN 19: CONTENIDO DEL ÍNDICE.....	58
ILUSTRACIÓN 20: RESULTADOS DEL FILTRO DE BÚSQUEDA.....	59
ILUSTRACIÓN 21: RESULTADOS DE LA ENCUESTA.....	60
ILUSTRACIÓN 22: BÚSQUEDA AVANZADA POR INGREDIENTES.....	C
ILUSTRACIÓN 23: RESULTADOS DE LA BÚSQUEDA POR INGREDIENTES.....	C
ILUSTRACIÓN 24: VISTA AVANZADA DE LA RECETA.....	D
ILUSTRACIÓN 25: RESULTADOS DE <i>ALLRECIPES</i>	D
ILUSTRACIÓN 26: BÚSQUEDA FILTRADA.....	E
ILUSTRACIÓN 27: RESULTADOS DE LA BÚSQUEDA FILTRADA.....	E
ILUSTRACIÓN 28: RESULTADOS DE LA BÚSQUEDA FILTRADA POR SIMPLYRECIPES.....	F
ILUSTRACIÓN 29: BÚSQUEDA RÁPIDA.....	G
ILUSTRACIÓN 30: RESULTADOS DE BÚSQUEDA RÁPIDA.....	G
ILUSTRACIÓN 31: RESULTADOS PARA LA PREGUNTA I.....	I
ILUSTRACIÓN 32: RESULTADOS PARA LA PREGUNTA II.....	J
ILUSTRACIÓN 33: RESULTADOS PARA LA PREGUNTA III.....	J
ILUSTRACIÓN 34: RESULTADOS PARA LA PREGUNTA IV.....	K
ILUSTRACIÓN 35: RESULTADOS PARA LA PREGUNTA V.....	K

GLOSARIO

API: Las siglas en inglés son “*Application Programming Interface*” y ofrece un conjunto de rutinas para ser incluido en la lógica interna de otro software.

Bot: Programa informático que tiene la finalidad de imitar el comportamiento humano con acciones como poner comentarios en una página, editar información, navegar por el sitio web de una página, etc.

Business intelligence: Conjunto de estrategias dedicadas a la creación de conocimiento de una cierta área, a través del análisis de datos existentes.

Big data: Área de la informática dedicada a la acumulación a gran escala de datos para identificar patrones dentro de dichos datos.

DOM: Las siglas en inglés son “*Document Object Model*” y es una *API* que proporciona los elementos necesarios para representar documentos *HTML* y *XML*.

Feedback: Capacidad de un emisor para recoger reacciones de los receptores, e interpretar su mensaje, de acuerdo con lo recogido.

Google Insight: Herramienta creada por Google Labs que muestran los términos de consulta más populares por los usuarios que utilizan la tecnología Google.

HTTP: Las siglas en inglés son “*Hypertext Transfer Protocol*” y es un protocolo mediante el cual se transfiere información entre servidores y clientes.**HTML:** Las siglas en inglés son “*Hypertext Markup Language*” y es el lenguaje por etiquetas utilizado para el desarrollo de páginas web.

JVM: Las siglas en inglés son “*Java Virtual Machine*” y es una máquina virtual ejecutable en una plataforma específica capaz de interpretar expresiones generadas por un compilador basado en el lenguaje Java.

jQuery: Biblioteca que simplifica la interacción con los elementos del *DOM* y con las páginas *HTML*.

MVC: Las siglas en inglés son “*Model View Controller*” y es un patrón de arquitectura del software muy utilizado para el software destinado a la interacción con el usuario.

PetaByte: Unidad de almacenamiento de la información cuyo símbolo es PB y equivale a 10^{15} bytes.

Recomendador: Sistema inteligente que proporciona a los usuarios un conjunto de sugerencias sobre un conjunto de elementos. Trabaja en función de los gustos de cada usuario para ofrecer dichas sugerencias.

RAM: Las siglas en inglés son “*Random Access Memory*” y es la memoria principal del ordenador sobre la que se pueden realizar operaciones de lectura y escritura.

Rollback: Operación relacionada con el mundo de la informática que se caracteriza por devolver el software a una versión anterior.

TCP/IP: Las siglas en inglés son “*Transmission Control Protocol/Internet Protocol*” y es un protocolo que permite a dos anfitriones establecer una conexión e intercambiar datos.

URL: Las siglas en inglés son “*Uniform Resource Locator*” y es una dirección formada por una secuencia de caracteres, hace referencia a un recurso variable en el tiempo.

Web Scraping: Técnica utilizada por algunas aplicaciones de software para extraer información de páginas web.

WAR: Las siglas en inglés son “*Web Application aRchive*” y es un archivo utilizado para distribuir un conjunto de elementos que juntos forman una aplicación web.

XML: Las siglas en inglés son “*eXtensible Markup Language*” y es el lenguaje estándar por etiquetas utilizado para el intercambio de información estructurada entre sistemas.

1. INTRODUCCIÓN

En esta sección se da a conocer la motivación que permite orientar este proyecto en la dirección adecuada para lograr una serie de objetivos propuestos. También se da a conocer el ciclo de vida diseñado y la estructura del documento dividida por apartados.

1.1. MOTIVACIÓN

Desde hace ya algunos años y gracias a la innovación de internet y sus servicios, la tecnología web se posiciona como referente de búsqueda de información frente a los métodos convencionales, como prensa, radio o televisión. Esta información abarca un gran número diferente de áreas, donde cada vez son más los servicios que ponen a disposición de los usuarios dicha información.

A medida que han ido creciendo las posibilidades de explotación comercial en internet, han sido muchas empresas las que han decidido especializarse en las diferentes áreas que ofrece este nuevo mundo virtual, y una de las más importantes a día de hoy es la recopilación, estructuración y clasificación de la información contenida en internet. Un buen ejemplo de este sector son los diferentes buscadores de información que existen como consecuencia de ello. Entre los más conocidos están: *Google*, donde casi un 90% de los usuarios de internet lo utilizan [1], *Yahoo!*, el competidor directo de *Google*, o *Bing*, buscador oficial de la empresa *Microsoft*.

Una de las áreas que se ha puesto de moda en internet en estos últimos años es el sector de la cocina, ya que según detalla la herramienta *Google Insight* [2], el porcentaje de búsquedas relacionadas con dicha área se han visto aumentadas en estos últimos años. Como consecuencia de ello, el número de proyectos en internet que ofrecen fundamentalmente información gastronómica han incrementado gracias a esta nueva tendencia.

Aunque la mayoría de los portales de gastronomía ofrecen buscadores integrados en sus propias páginas, es muy habitual que dichos buscadores estén implementados con búsquedas literales, convirtiéndolos en buscadores poco óptimos y con un amplio margen de mejora. En el Anexo B “*Casos de éxito*” se muestra un ejemplo de esta problemática.

Esta situación abre la posibilidad de realizar un proyecto basado en la creación de un buscador que ofrezca técnicas de búsqueda más avanzadas [2]: booleana, *wildcard*, por rangos, etc. En el Anexo A “*Tipos de búsquedas implementadas*”, se detallan los métodos implementados.

1.2. OBJETIVOS

El proyecto posee una serie de objetivos que se exponen a continuación y definen el itinerario a seguir:

Extracción de los datos a utilizar: Para la consecución del proyecto son necesarios datos relacionados con el sector gastronómico sobre los que probar el sistema, por lo que es vital desarrollar un sistema de extracción de la información que ofrecen los sitios web de cocina, y que permita insertar la información en una base de datos relacional.

Almacenamiento de la información y aplicación de búsqueda: Tal y como se ha comentado en la motivación del proyecto, la mayoría de los buscadores ofrecen la funcionalidad de búsqueda literal. El reto, por tanto, es ofrecer un producto con un sistema de búsqueda más potente que este. Para ello es imprescindible la creación de un sistema de búsquedas que contenga dentro de la lógica de la aplicación dos tipos de motores de almacenamiento, que hacen referencia a las técnicas de búsqueda mencionadas en el anterior apartado del presente documento.

Estudio posterior del producto desarrollado: Otro de los retos del proyecto es verificar si los sistemas implementados mejoran las aplicaciones de búsqueda ‘convencionales’ (por literal). Para ello es necesario analizar las valoraciones de los usuarios sobre las herramientas desarrolladas.

Posteriores fases del proyecto: Este proyecto al trabajar sobre el mundo de internet, tiene el riesgo de que las herramientas implementadas queden desfasadas debido a la gran competencia existente. Por ello, es vital el estudio de posibles mejoras que se pueden abordar en fases futuras.

1.3. CICLO DE VIDA

El propósito de este apartado es detallar las distintas fases por las que el proyecto debe pasar para garantizar que el producto final ha sido desarrollado con una metodología verificada y estándar. A continuación se exponen las fases del proyecto:

Análisis de requisitos: Consiste en la educación de los requisitos del usuario. Para esta fase se debe seguir una serie de pasos, que permiten establecer las funcionalidades del proyecto y valorar la dificultad del software a implementar. Dichos pasos son:

- I. Obtener los requisitos a través de entrevistas con el cliente.
- II. Realizar un análisis de dichos requisitos y transformarlos en condiciones apropiadas para ser tratadas en la fase de diseño del proyecto.
- III. Analizar con el cliente que los requisitos tratados sean válidos para la necesidad que inicialmente propició el inicio del proyecto.

Diseño: Esta fase permite describir cómo el sistema va a satisfacer los requisitos validados de la fase anterior, poniendo énfasis en la estructuración modular de la aplicación.

Implementación: La fase de implementación comprende el intervalo de tiempo desde el inicio de la codificación hasta el comienzo de las pruebas de validación sobre el sistema desarrollado. Por tanto, las pruebas unitarias y de integración se realizan durante esta fase, ya que el coste de localización y corrección de errores es menor que en fases posteriores.

Pruebas: Durante esta fase se realizan las pruebas de sistema, validación y aceptación que permiten verificar que la aplicación desarrollada es consistente con los requerimientos iniciales del cliente.

Mantenimiento: En esta fase se contemplan los cuatro tipos de mantenimiento existentes: correctivo, adaptativo, perfectivo y preventivo.

A continuación se expone y argumenta el ciclo de vida escogido para el proyecto:

Evolutivo

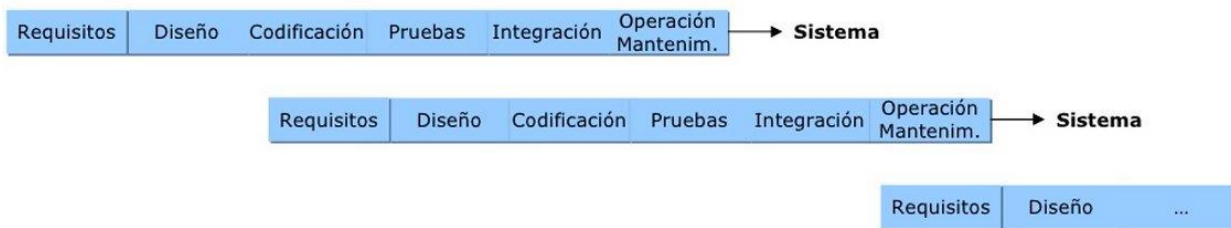


Ilustración 1: Modelo de ciclo de vida

Se ha escogido el **modelo evolutivo** como patrón a seguir, ya que facilita el desarrollo de nuevas funcionalidades sobre los sistemas para fases posteriores, y permite al equipo desarrollador llevar mejoras continuas del producto para hacer frente a la fuerte competencia del sector. Otra de las circunstancias por las que este ciclo de vida resulta el más apropiado es a causa del desconocimiento inicial de todas las necesidades operativas del sistema, ya que estas irán surgiendo como peticiones de mejora del cliente para adaptar la aplicación a sus gustos y necesidades futuras.

1.4. ESTRUCTURA DEL DOCUMENTO

Todas las fases por las que el proyecto ha pasado están descritas en el presente documento, donde cada punto está subdividido en los correspondientes apartados que recogen los detalles y las características más importantes que lo componen.

El **capítulo 2** de este documento, hace referencia al *estado del arte*, y en él se ofrece un estudio previo de investigación, junto con las tecnologías y herramientas tenidas en cuenta durante la realización del proyecto.

El *análisis de requisitos* se describe en el **capítulo 3**, y recoge las funcionalidades tanto del proyecto *Web Scraper*, que de aquí en adelante será como se denomine al proyecto de extracción de la información, como del buscador de recetas. Para ambos sistemas se detallan, por tanto, las capacidades que debe satisfacer cada uno, así como las restricciones en sus funcionalidades.

En el **capítulo 4** se exponen los *diseños* de ambos sistemas, especificando la estructura modular de las aplicaciones y tipos de datos, junto con los diagramas diseñados.

Las especificaciones más técnicas de la fase de *implementación*, para ambas aplicaciones, se describen en el **capítulo 5**.

Las fases de *pruebas y resultados* se exponen en los **capítulos 6 y 7** respectivamente. En estas fases se describe la batería de pruebas diseñada para la verificación de las aplicaciones desarrolladas y el estudio posterior realizado de dichas pruebas mostrando las comparativas y diagramas obtenidos.

Por último, en el **capítulo 8** se describen las *conclusiones y trabajo futuro*.

2. ESTADO DEL ARTE

En esta sección se expone un análisis del dominio de la aplicación y las alternativas tecnológicas estudiadas, argumentando las que finalmente se han escogido para su uso en el proyecto.

2.1. ESTUDIO DEL CONTEXTO ACTUAL DEL SECTOR

En este proyecto se va a trabajar con una página de cocina como base para la obtención previa de la información que necesita la aplicación de búsqueda que se quiere desarrollar. Por ello, se ha realizado un estudio previo de las páginas de cocina más importantes de este sector: *Allrecipes.com*, *SimplyRecipes.com* y *Wikicocina.com*, teniendo en cuenta los siguientes factores:

- Distribución de la información: Se evalúa la disposición de los elementos del *DOM*, ya que las páginas mejor estructuradas son más fáciles de tratar.
- Calidad de la información: Se evalúa si la información que ofrece la página está revisada y corregida frente a errores.
- Cantidad de la información: Cuanta más información exista, mayor será el abanico de servicios que se puedan ofrecer.
- Número de usuarios: Es uno de los factores más importantes, ya que las páginas más ‘vivas’ ofrecen un mayor número de servicios.

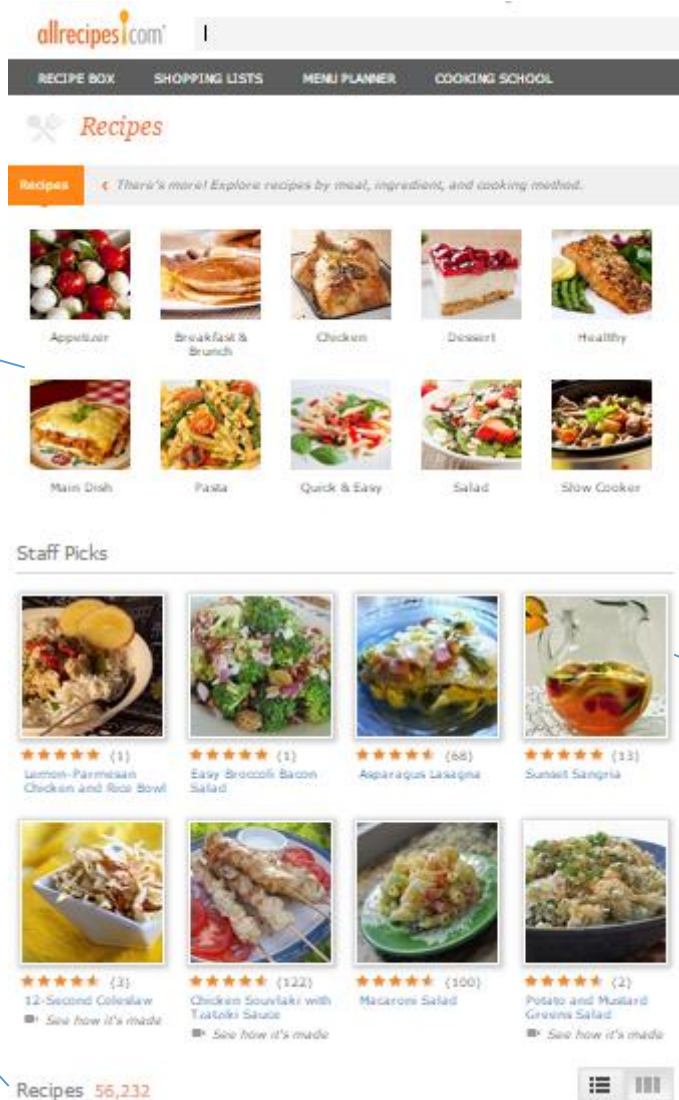
Aunque se trabaje con la información de un sitio web, todos los futuros diseños e implementaciones están pensados para que el sistema sea capaz de extraer la información de cualquier página web. Esta decisión evita futuros problemas en fases posteriores si se quiere ampliar las búsquedas a otras páginas web que no se han contemplado durante el ciclo de vida de este proyecto.

Desarrollo de un sistema de recogida y categorización de datos de recetas

2.1.1. PORTAL WEB ALLRECIPES

El aspecto de esta web durante su análisis es el siguiente:

Las categorías de las recetas establecen el patrón de distribución de la información.



El método de búsqueda literal es la que predomina en esta página.

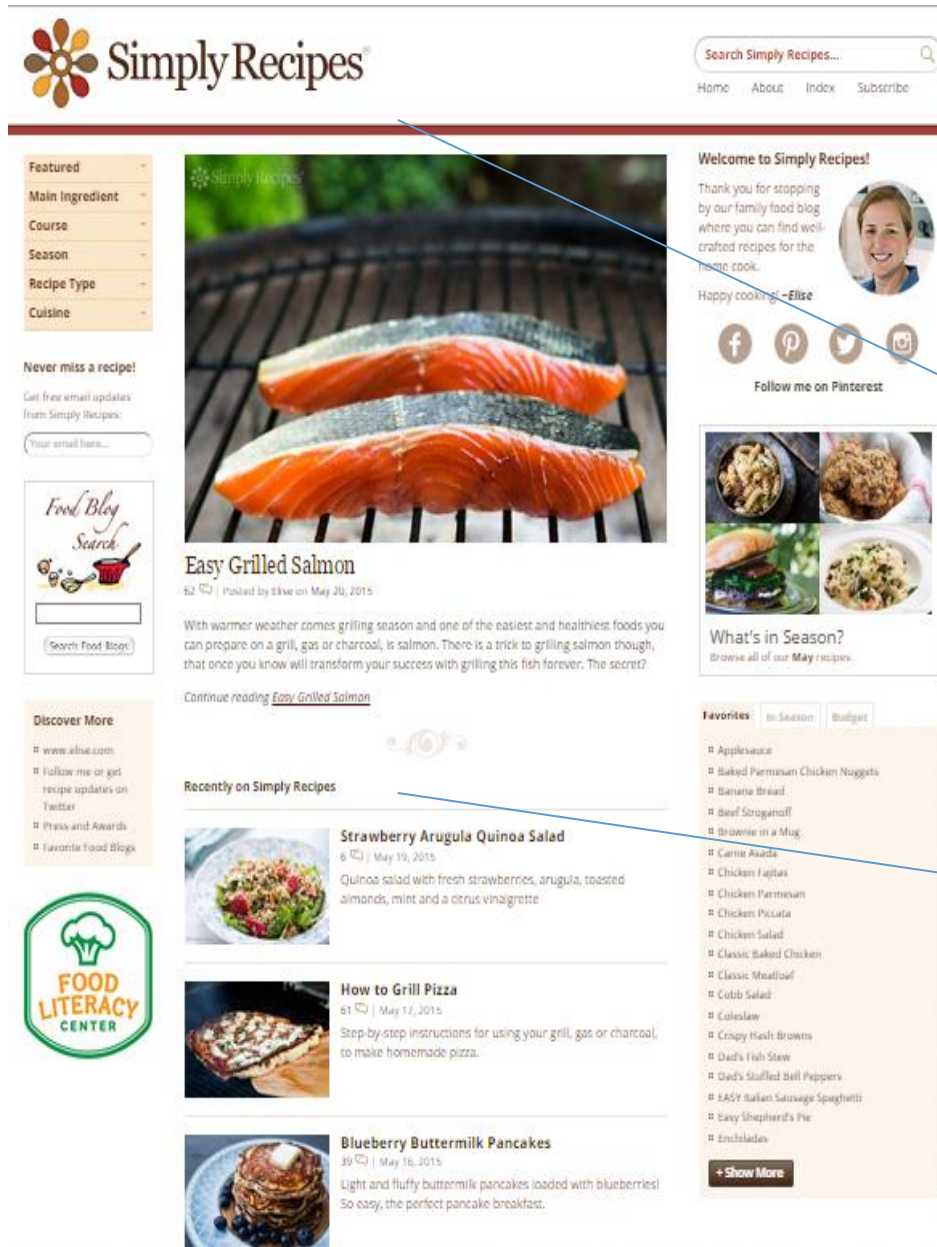
Ofrece una gran cantidad de información.

El sitio web es muy vivo y con gran número de valoraciones de usuarios.

Ilustración 2: Sitio web Allrecipes

2.1.2. PORTAL WEB SIMPLYRECIPES

El análisis visual de esta web es el siguiente:



El método de búsqueda literal es la que predomina en esta página.

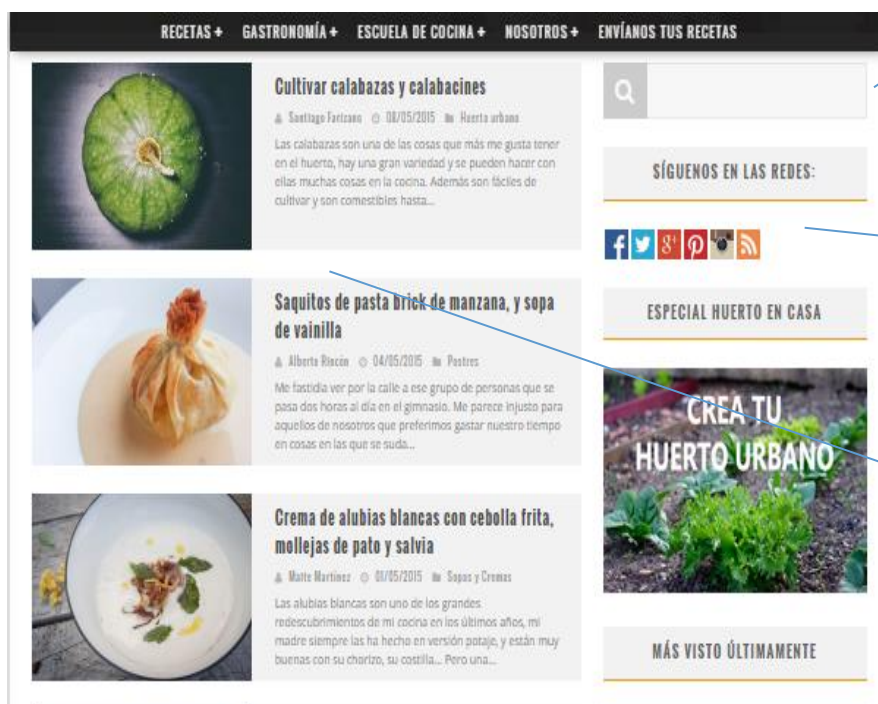
La distribución de la página no permite visualizar el número total de recetas.

No ofrece valoraciones ni puntuaciones de los usuarios a simple vista.

Ilustración 3: Sitio web Simplyrecipes

2.1.3. PORTAL WEB WIKICOCINA

A continuación se muestra el estudio para este portal:



Como en los casos anteriores predomina la búsqueda literal.

Esta web tampoco proporciona las valoraciones de los usuarios

La distribución de la información hace complicada su extracción.

Ilustración 4: Sitio web Wikicocina

2.1.4. CONCLUSIONES DEL ESTUDIO

La siguiente tabla muestra las conclusiones del estudio realizado:

Nombre	Distribución	Calidad	Cantidad	Nº de usuarios
Allrecipes.com	Buena	Buena	Alta	Alta
Simplyrecipes.com	Aceptable	Aceptable	Media	Alta
Wikicocina.com	Mala	Aceptable	Media	Baja

Ilustración 5: Estudio de las cualidades de las páginas

Como se puede observar, *Allrecipes* se diferencia por ser el sitio web que ofrece la mejor distribución, calidad y cantidad de contenido, por lo que ha sido la opción escogida ya que además de las cualidades detalladas en la tabla superior, se caracteriza por ofrecer para la mayoría de las recetas todas las opiniones y *feedback* de los usuarios.

2.2. ESTUDIO DE ALTERNATIVAS TECNOLÓGICAS

En este apartado se expone un análisis previo de las distintas alternativas técnicamente viables para el proyecto, y una justificación de la solución propuesta.

Para optimizar la calidad del producto final que se desea alcanzar, se ha dividido el estudio de las tecnologías en secciones que hacen referencia a los objetivos que se quieren conseguir.

2.2.1. EXTRACCIÓN DE LA INFORMACIÓN

La técnica más empleada para la extracción de información de una página web a un formato estructurado (por ejemplo, a una base de datos como es este caso), es la denominada ‘*web scraping*’.

A continuación se exponen las librerías más utilizadas a día de hoy, para la implementación de esta técnica.

2.2.1.1. LIBRERÍA JSOUP

Es la librería [3] más utilizada para la técnica anteriormente explicada y se caracteriza por los siguientes puntos:

- Codificación similar a *JQuery* en la selección de los elementos del *DOM*, por lo que la hace muy fácil de entender si ya se ha trabajado con *JQuery* anteriormente.
- Permite manipular los elementos, sus atributos y el texto del *HTML*.
- Posibilidad de utilización del método *UserAgent* para las conexiones con el servidor. Esto permite acceder al contenido emulando a un usuario y evitando problemas de denegación de contenido por *bot*.
- Amplia documentación de la librería en internet debido a su amplio uso, lo que hace que sea más fácil la fase de implementación de este punto en el proyecto.
- Permite estructurar el código de forma sencilla gracias a los paquetes de la librería.
- Optimiza el código de recogida de datos, ya que se necesitan menos líneas de código para la implementación de un motor de extracción de información frente

a otras *APIs*. Como ejemplo de eficiencia, solo es necesario una línea para obtener el contenido *HTML* de una *URL*.

- Como librería de Java, se puede integrar con cualquier lenguaje *JVM*, por lo que no limita la elección del lenguaje a utilizar del proyecto en general, en este sentido.

2.2.1.2. LIBRERÍA XPATH

Esta librería [4] está diseñada para el tratamiento de documentos *XML*, y como se verá es potente y de fácil utilización. Los beneficios de su utilización son:

- Su diseño está pensado para ser utilizado en diversos contextos: extracción de información, *big data*, automatización de pruebas, etc.
- Librería diseñada para ser reutilizable, lo que facilita la navegación por los sitios web.
- Amplio manejo de las búsquedas, lo que la hace una librería potente.
- Es una librería extensible, por lo que permite la optimización de código y desarrollo de nuevos componentes sin perder la eficiencia.

2.2.1.3. LIBRERÍA HTMLUNIT

Otra librería [5] para la manipulación de alto nivel de sitios web es *HtmlUnit*. A continuación se listan las particularidades de esta librería:

- Librería potente que permite simular el comportamiento de los navegadores, incluyendo los aspectos de bajo nivel de los protocolos *TCP/IP* y *HTTP*.
- Permite la extracción de información de forma estructurada directamente de la página web.
- Librería basada en *XPath*, lo que puede indicar no ser muy efectiva para las páginas web más modernas que hagan uso de información no estructurada.
- Permite la navegación a través de hipertexto y obtener otras páginas web.
- Su uso más común es para la automatización de pruebas de páginas web.
- Al no poseer una interfaz gráfica, el paquete que contiene la API es ligero.

2.2.2. TRATAMIENTO DE DATOS E INDEXACIÓN

La función del tratamiento de datos es filtrar, fragmentar y estructurar la información extraída de las páginas web, de tal forma que se pueda pasar de forma eficiente de información no tratada obtenida de la web, a información estructurada y lista para ser consultada.

La mejora al acceso de la información es un punto clave ya que una información estructurada, clara y relevante, proporciona una base importante en el producto final del proyecto.

A continuación se exponen las diferentes librerías estudiadas para su uso en la implementación del proyecto.

2.2.2.1. LIBRERÍA LUCENE

Esta *API* [6] proporciona los objetos y funciones necesarias para la implementación de un motor de búsqueda de texto completo. Esto incluye desde el diseño y creación de los documentos que van a formar el índice, hasta funciones de indexación y búsqueda de dichos documentos a partir de una consulta. También permite diseñar los analizadores de texto que tratan la información antes de incluirla en los documentos que conforman el índice. A continuación se detallan sus características más importantes:

- Versátil: Es una *API* multiplataforma que tiene versiones para los lenguajes: Java, c#, c++, Perl, Python, Ruby, PHP y Delphi.
- El acceso a los datos se realiza de manera eficiente, ya que los requerimientos de memoria *RAM* son de pocos *megabytes*; también es rápida, ya que se pueden realizar búsquedas de millones de registros en pocos milisegundos.
- Flexibilidad en los tipos de búsqueda, entre las más conocidas están la búsqueda proximal, lógica o por rangos.
- Proporciona gran cantidad de documentación fácil de seguir.
- Permite de forma simultánea la actualización y búsqueda en el índice sin perder la consistencia.
- Resultados relevantes y ranking: el motor de búsqueda devuelve el listado de los resultados con un valor asociado de relevancia, estando situados los más relevantes en las primeras posiciones.

2.2.2.2. LIBRERÍA SPHINX

Sphinx [7] es un servidor de búsquedas de contenidos, implementado en el lenguaje c++. Este sistema es de código abierto al igual que en el caso anterior. Las características de esta librería son:

- Permite configurar los pesos de los campos en los que se realiza la búsqueda, de manera que unos tengan más relevancia que otros.
- La indexación de información se realiza de manera eficiente y rápida.
- No permite actualización parcial del índice, por lo que para la realización de este proceso se requieren métodos más complejos (índices auxiliares con los cambios).
- Es un servidor versátil, ya que está soportado en múltiples sistemas operativos: Linux, Windows, Mac OS, Solaris, etc.
- Búsqueda distribuida: Las búsquedas pueden ser distribuidas en varias máquinas, permitiendo una alta disponibilidad en el acceso a datos.
- Fácil integración: Sphinx está distribuido en tres *APIs* diferentes (SphinxAPI, SphinxSE, y SphinxQL), de esta manera, el sistema está soportado en múltiples lenguajes de programación como: Java, PHP, Python, Perl, C, entre los más comunes.

2.2.2.3. LIBRERÍA SOLR

Se trata de un motor de búsqueda [8] basado en la API del proyecto Lucene, aunque también se le conoce por tener algunas características únicas de este sistema. A continuación se destacan las particularidades del motor:

- Solr es una aplicación web (*WAR*) que puede ser desplegada en cualquier contenedor de aplicaciones (como por ejemplo en Apache Tomcat).
- Facilidad de uso: Permite ser configurado y utilizado directamente por los usuarios.
- Excelente soporte: Gracias a la comunidad de Solr, se tiene al alcance información muy ventajosa y útil.
- Versátil: Debido a que Solr está basado en el proyecto Lucene, el índice de Lucene y el índice de Solr son lo mismo, por lo que es viable el uso del índice de Solr por Lucene y viceversa.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- Funcionalidad extra: Proporciona múltiples tipos de campos como boolean, string, date, etc. También ofrece la funcionalidad “*more like this*”, que permite la obtención de documentos similares.

2.2.3. MOTOR DE BASE DE DATOS

En este punto se detallan las tecnologías tenidas en cuenta para el módulo que permite contener en un modelo relacional toda la información extraída de los sitios web de cocina.

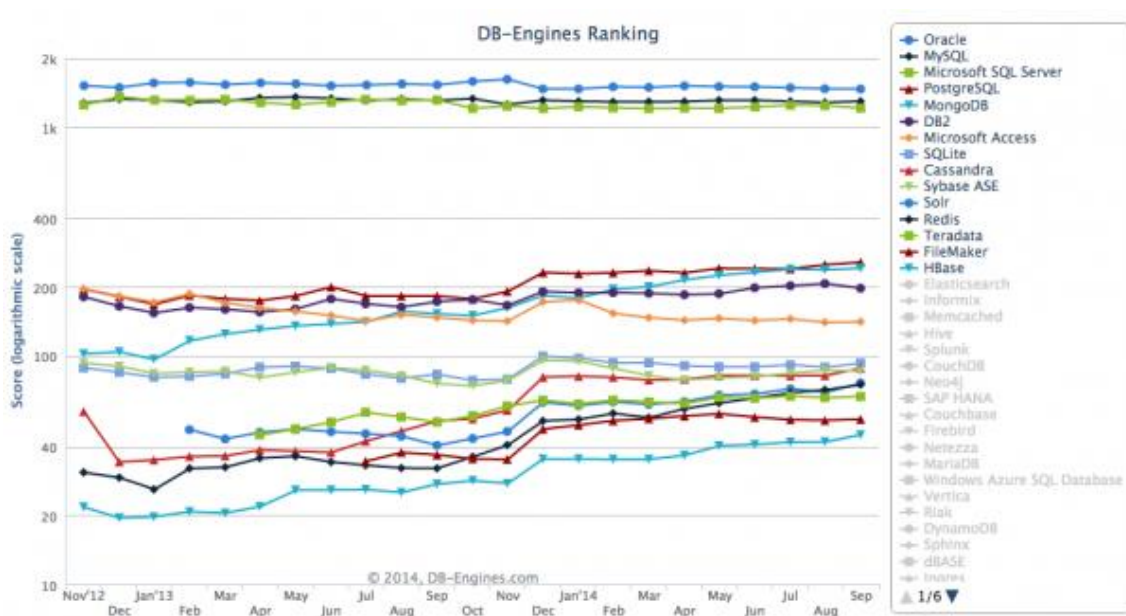


Ilustración 6: Grado de utilización de los motores de base de datos

En la imagen superior se puede ver un gráfico que representa el grado de utilización de los diferentes motores de bases de datos disponibles en el mercado. Se tendrá en cuenta para la elección posterior de la tecnología de este área.

2.2.3.1. MICROSOFT SQL SERVER

Microsoft SQL Server [9], es el motor de base de datos desarrollado por la empresa Microsoft y basado en el modelo relacional. Se caracteriza por los siguientes puntos:

- Es el motor más ampliamente utilizado en el sector comercial.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- Limitado al SO de Windows. Esta característica es una ventaja para aquellos sistemas que utilizan prioritariamente productos de Microsoft.
- Seguridad: SQL server permite integrarse con la autenticación de Windows, lo que proporciona un sistema seguro para el manejo de la información.
- Escalabilidad: SQL server es un motor capaz de gestionar *petabytes* de datos sin apenas perder la calidad de los servicios que ofrece.
- Es uno de los productos ‘estrella’ de Microsoft, por lo que viene con un buen soporte y una gran cantidad de información útil y bien explicada.
- Al no ser de código abierto, es un producto muy caro (su precio supera los 20,000 dólares).

2.2.3.2. ORACLE DATABASE

Oracle 11g [10] es un sistema de gestión de bases de datos relacionales desarrollado por Oracle Corporation y ofrece las siguientes ventajas:

- Es la opción más popular para los proyectos de carácter comercial más importantes, y la más antigua del mercado.
- Está disponible en diferentes versiones: Enterprise, Standard, Standard Edition One, Express.
- Multiplataforma: mayor flexibilidad en cuanto a despliegue en sistemas operativos que su homólogo de Microsoft.
- Completo: Oracle contiene todas las funcionalidades que la definen como un motor serio, el lenguaje que utiliza (PL/SQL) permite implementar diseños con *triggers* y procedimientos almacenados, con una integridad referencial potente.
- Eficiencia: Es posible segmentar las bases de datos para la mejora de la eficiencia, de la replicación y algunas versiones permiten la administración de bases de datos distribuidas.
- Al igual que el caso anterior, es un producto caro.

2.2.3.3. SQLITE

SQLite [11] es un sistema de base de datos relacional de dominio público creado por D. Richard Hipp. A continuación se destacan sus características:

- Gracias a su diseño, SQLite no es un proceso diferente del programa con el que se comunica, sino que permite integrarse directamente con la lógica interna del proyecto, permitiendo de esta manera tener un mayor control de la comunicación entre base de datos/aplicación.
- Reduce el tiempo de respuesta de la recuperación de la información gracias a sus subrutinas y funciones, las cuales son más eficientes que la comunicación entre procesos.
- El conjunto de bases de datos se guarda en forma de fichero binario en la localización del programa cliente.
- Multiplataforma: SQLite permite su uso en diversos sistemas operativos, y su biblioteca está disponible en muchos lenguajes de programación, siendo los más utilizados: C/C++, Java, Python, PHP y C#.
- Motor potente: Su API incluye funcionalidades como: transacciones de bases de datos atómicas, consistencia de base de datos, aislamiento, durabilidad y *triggers*.
- Permite concurrencia de lectura: Varios hilos o procesos pueden acceder y consultar sobre una misma base de datos sin problemas.

2.2.4. INTERACCIÓN CON EL USUARIO

Se define este punto como el sistema de interacción directa con el usuario, es decir, el módulo principal que se encarga de manejar el resto de funcionalidades del proyecto.

Se han tenido en cuenta los siguientes frameworks para su implementación.

2.2.4.1. JAVA SWING

Esta API [12] basada en el lenguaje Java y en la arquitectura *MVC*, proporciona una librería para la implementación de interfaces gráficas. A continuación se detallan las características de esta plataforma:

- Extensible: Java Swing permite diseñar implementaciones personalizadas basadas en las que provee la librería por defecto.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- Interfaz ligera: Utiliza los controladores de las *APIs* de Java 2D para la visualización de sí mismo, en vez de utilizar los controladores del sistema nativo en el que se despliega.
- El comportamiento de la librería entre plataformas es similar, por lo que reduce el esfuerzo del mantenimiento de la aplicación.
- Dispone de un gran número de componentes: Árboles de búsqueda, tablas, barras de desplazamiento, herramientas de ayuda, etc.
- Gracias a la librería “*look and feel*”, se puede modificar la apariencia de los componentes de Java Swing. (Por ejemplo, los componentes se pueden adaptar a la apariencia de la plataforma en la que es desplegada la solución).
- Proporciona un método de depuración que reduce el tiempo de resolución de incidencias.

2.2.4.2. WINDOWS PRESENTATION FOUNDATION (WPF)

WPF [13] es un *framework* de Microsoft que permite el desarrollo de interfaces gráficas en sistemas Windows. Las características de esta tecnología son:

- WPF está basado en características de aplicaciones de escritorio de Windows y aplicaciones web.
- Es un *framework* muy potente que permite diseñar aplicaciones basadas en la interacción con componentes de animación, audio, video, documentos y gráficos en 3D.
- Separa la interfaz gráfica de la lógica de negocio interna mediante los lenguajes de programación de *.NET*, ofreciendo de esta manera una arquitectura *MVC* para el diseño de aplicaciones.
- Accesibilidad: WPF permite la integración con “*Microsoft UI Automation*” que posibilita el diseño de interfaces más accesibles.

2.3. TECNOLOGÍAS A UTILIZAR

En el apartado de extracción de información, se ha tenido en cuenta las futuras ampliaciones que puede experimentar este sistema, tales como la posibilidad de obtención de datos a través de varias webs, la inclusión del método *Big Data*, etc.

Entre las opciones estudiadas, hay una de ellas que destaca respecto a las demás, la *API* de JSoup. Como se ha comentado antes, esta librería, permite incluir la funcionalidad de *UserAgent* en las conexiones con la página, lo que evitaría problemas de exclusión de información de algunas páginas. Además esta librería tiene un comportamiento más óptimo frente a las otras opciones para páginas más modernas gracias a los selectores de CSS y JQuery que contiene.

Para la administración de la base de datos se ha elegido SQLite por ser un motor de código abierto que permite un mayor manejo e inclusión en la lógica de la aplicación.

Respecto al tratamiento de la información, se descarta la opción de Sphinx, a causa de la imposibilidad de actualización parcial del índice, ya que supone tener que crear de nuevo el índice completo en caso de que se produzca algún cambio, y con ello un consumo excesivo de recursos.

Por tanto, con el sistema Sphinx descartado para el tratamiento de la información del proyecto, queda por decidir entre dos opciones: Lucene y Solr. En principio, ambas opciones son muy parecidas y convenientes para la aplicación que se presenta. Sin embargo, Lucene facilita la inclusión del motor de búsqueda dentro de la lógica del sistema, de manera que se tiene un control total de las funcionalidades que aporta Lucene. Por otro lado, con Solr se necesita mantener como un sistema separado. Por lo tanto, la opción de Lucene es la óptima para este caso.

Por último, queda elegir el *framework* a utilizar para la implementación del sistema de interacción con el usuario. En este caso, Java Swing es la opción elegida ya que con WPF, la aplicación se limitaría a sistemas con SO Windows, y se pretende conseguir una aplicación multiplataforma y de software libre.

3. ANÁLISIS DE REQUISITOS

En esta sección se recogen todos los requerimientos que debe cumplir el proyecto de categorización y búsqueda de recetas, por lo que se va a analizar, documentar y verificar los requisitos del software, de manera que el proyecto quede bien definido para las fases posteriores.

Antes de describir los requerimientos de cada aplicación, se ofrece una breve descripción de la estructuración de cada sistema (en la sección 4 del presente documento se analiza de forma detallada el diseño de las aplicaciones):

- La aplicación *Web Scraper* se encarga de la extracción de la información a partir de una página web seleccionada. Este proyecto está formado por los siguientes subsistemas:
 - El módulo de navegación se encarga del rastreo de páginas de todo el sitio web.
 - El módulo de extracción de datos se encarga de la recopilación de la información.
 - El módulo de acceso a datos se encarga de la administración del fichero de base de datos que permite contener la información extraída.
- La aplicación buscador se encarga de la interacción con el usuario y es la encargada de mostrar los resultados de las búsquedas de recetas, está estructurado siguiendo un patrón *MVC*.

3.1. REQUISITOS FUNCIONALES

Los requisitos funcionales establecen el comportamiento del sistema, y describen las transformaciones que la aplicación realiza sobre las entradas para producir salidas.

En las siguientes secciones, se describen los requisitos funcionales para los sistemas definidos. En ambos casos, los requisitos se exponen en un formato de casos de uso, que contiene los siguientes campos:

- Nombre/Identificador: Este campo permite tener etiquetados todos los requerimientos, de manera que en fases posteriores sea fácil consultarlos y clasificarlos.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- Descripción: El campo descripción permite a la empresa desarrolladora tener una idea clara del requerimiento que se desea cubrir.
- Pre-condiciones: Este campo describe las situaciones previas que deben darse dentro del entorno de la aplicación.
- Post-condiciones: Las post-condiciones definen los efectos producidos en el sistema tras producirse el evento.

3.1.1. REQUISITOS FUNCIONALES: PROYECTO WEB SCRAPER

Los requisitos funcionales para esta aplicación son los siguientes:

Nombre identificador	RFA01 Integración con sitios web de cocina.
Descripción	El sistema debe tener la capacidad de comunicación con la página web de recetas, para posteriormente realizar la extracción de las mismas.
Pre-condiciones	El entorno donde se ejecuta la aplicación debe tener conexión a internet.
Post-condiciones	La aplicación informa en el log de que la conexión ha sido establecida con éxito.

Nombre identificador	RFA02 Navegación por la web de cocina.
Descripción	El sistema debe tener la capacidad de navegar por el sitio web de recetas de forma automática.
Pre-condiciones	La integración previa con la página web se estableció con éxito.
Post-condiciones	La aplicación devuelve todas las <i>URLs</i> de categorías de recetas encontradas, para comenzar con su exploración.

Nombre identificador	RFA03 Extracción de las direcciones.
Descripción	El sistema debe tener la capacidad de extracción de todas las direcciones de una misma categoría.
Pre-condiciones	La navegación por la web devolvió las categorías encontradas.
Post-condiciones	La aplicación devuelve un listado de las direcciones encontradas.

Desarrollo de un sistema de recogida y categorización de datos de recetas

Nombre identificador	RFA04 Extracción de la información.
Descripción	El sistema debe tener la capacidad de extracción de todos los datos relacionados con una receta: información básica, ingredientes, pasos, nutrientes, etc.
Pre-condiciones	La integración con la web de cocina se estableció correctamente y se posee el enlace a la página de la receta.
Post-condiciones	La aplicación crea un objeto para insertar toda la información extraída.

Nombre identificador	RFA05 Tratamiento de la información.
Descripción	El sistema debe poder tratar todos los campos extraídos de tal forma que se desechen todos los datos no válidos.
Pre-condiciones	La aplicación ha extraído con éxito los datos de una receta.
Post-condiciones	No procede.

Nombre identificador	RFA06 Integración con la librería de base de datos.
Descripción	El sistema debe integrarse con la librería SQLite de manera que sea capaz de crear el esquema de base de datos a utilizar.
Pre-condiciones	No procede.
Post-condiciones	La aplicación crea el objeto de base de datos que contendrá toda la información extraída de la página web, únicamente en caso de que no exista.

Nombre identificador	RFA07 Almacenamiento de la información.
Descripción	El sistema debe ser capaz de almacenar en la base de datos la información tratada.
Pre-condiciones	La aplicación ha extraído y tratado con éxito los datos de una receta de la web, además de haber creado e inicializado la base de datos.
Post-condiciones	El sistema se prepara para llevar a cabo la misma acción para todas las direcciones de las recetas.

3.1.2. REQUISITOS FUNCIONALES: PROYECTO BUSCADOR

Los requisitos funcionales para esta aplicación son los siguientes:

Nombre identificador	RFB01 Creación y almacenamiento del índice.
Descripción	La aplicación debe ser capaz de crear un índice haciendo uso de la <i>API</i> de Lucene, almacenando toda la información de las recetas en los documentos indexados.
Pre-condiciones	La aplicación debe tener acceso al objeto de base de datos creado previamente por el sistema <i>Web Scraper</i> .
Post-condiciones	La aplicación informa en el log que el índice basado en Lucene está listo para su uso.

Nombre identificador	RFB02 Carga del índice.
Descripción	La aplicación debe ser capaz de cargar el índice creado previamente.
Pre-condiciones	El índice basado en Lucene ha sido creado de forma correcta con anterioridad.
Post-condiciones	La aplicación informa en el log que el índice ha sido cargado de forma correcta.

Nombre identificador	RFB03 Consultar al índice.
Descripción	La aplicación debe tener la capacidad de lanzar consultas contra el índice cargado, y este a su vez debe ser capaz de devolver un listado ordenado por puntuación de los documentos recomendados de la búsqueda.
Pre-condiciones	El sistema ha cargado el índice previamente.
Post-condiciones	El sistema muestra por pantalla los resultados obtenidos.

Desarrollo de un sistema de recogida y categorización de datos de recetas

Nombre identificador	RFB04 Consultar a la base de datos.
Descripción	La aplicación debe tener dualidad de posibilidades de obtención de la información. Es decir, también tiene que ser capaz de lanzar consultas por base de datos aplicando una serie de filtros.
Pre-condiciones	El buscador debe tener acceso a la base de datos.
Post-condiciones	El sistema muestra por pantalla los resultados obtenidos.

Nombre identificador	RFB05 Mostrar los resultados obtenidos.
Descripción	La aplicación debe ser capaz de mostrar en un listado previo, las recetas encontradas para el filtro aplicado.
Pre-condiciones	El usuario previamente ha aplicado un filtro.
Post-condiciones	El sistema crea dinámicamente los accesos que permiten ver con detalle cada resultado.

Nombre identificador	RFB06 Acceso detallado a la información.
Descripción	El sistema debe ser capaz de mostrar un informe detallado para cualquier receta mostrada en el listado previo.
Pre-condiciones	La aplicación previamente ha mostrado al usuario un listado de recetas como resultado de una búsqueda del cliente.
Post-condiciones	El sistema crea los accesos necesarios que permiten la navegación a distintos informes detallados.

3.2. REQUISITOS NO FUNCIONALES

Los requisitos no funcionales definen las propiedades y cualidades que el producto debe tener. Estos no son partes de la razón básica del producto, pero sí son necesarios para hacer funcionar el producto de la manera deseada.

Al igual que en el caso anterior, se detallan los requisitos no funcionales para ambos sistemas, estructurándolos en tablas.

Desarrollo de un sistema de recogida y categorización de datos de recetas

3.2.1. REQUISITOS NO FUNCIONALES: PROYECTO WEB SCRAPER

Los requisitos no funcionales para este sistema son los siguientes:

Nombre identificador	RNFA01 Estabilidad del sistema.
Descripción	El sistema debe ser lo suficientemente robusto como para estar funcionando durante grandes intervalos de tiempo sin disminuir la calidad del servicio.
Prioridad	Alta.

Nombre identificador	RNFA02 Rendimiento del sistema.
Descripción	La aplicación debe ser capaz de procesar y almacenar los datos de cada receta de forma rápida.
Prioridad	Alta.

Nombre identificador	RNFA03 Escalabilidad del sistema.
Descripción	El sistema debe estar diseñado de tal forma que esté preparado para soportar evoluciones en fases posteriores.
Prioridad	Alta.

3.2.2. REQUISITOS NO FUNCIONALES: PROYECTO BUSCADOR

Los requisitos no funcionales para este sistema son los siguientes:

Nombre identificador	RNFB01 Rendimiento del sistema.
Descripción	El sistema debe ser capaz de procesar la información obtenida del índice o la base de datos de manera rápida.
Prioridad	Alta.

Nombre identificador	RNFB02 Estabilidad del sistema.
Descripción	El sistema debe ser capaz de soportar el procesamiento de una gran cantidad de información.
Prioridad	Alta.

Desarrollo de un sistema de recogida y categorización de datos de recetas

Nombre identificador	RNFB03 Usabilidad del sistema.
Descripción	El sistema debe tener una interfaz amigable de tal forma que su complejidad de uso sea mínima.
Prioridad	Alta.

Nombre identificador	RNFB04 Portabilidad del sistema.
Descripción	El sistema debe ser capaz de funcionar en los entornos de Windows, Mac y Linux.
Prioridad	Alta.

Nombre identificador	RNFB05 Escalabilidad del sistema.
Descripción	La aplicación debe ser capaz de soportar las posibles mejoras incluidas en futuras fases del proyecto.
Prioridad	Alta.

4. DISEÑO DEL SOFTWARE

El diseño es el proceso de definición de las especificaciones que permite a los desarrolladores tener una serie de referencias o disciplinas que hace que la fase de desarrollo del producto no solo sea más coherente, sino también más formal.

Por lo tanto, en esta sección se establecen conceptos como estructuras de datos, arquitectura general del software y representaciones de interfaz.

4.1. DISEÑO GENERAL

En el diseño de proyecto se ha tenido en cuenta la estructuración de los requisitos del apartado anterior, por lo que se tendrán dos proyectos separados: el primero de ellos se encargará de, para una página seleccionada, obtener la información de todas las recetas, procesarlas e introducirlas en una base de datos relacional. El segundo proyecto, tendrá la tarea de permitir búsquedas sobre base de datos e índice, atendiendo a las peticiones de los usuarios, y devolviendo los resultados oportunos.

Desarrollo de un sistema de recogida y categorización de datos de recetas

Para tener una visión general de todo el proyecto, se ofrece a continuación, una figura que muestra la integración de ambos sistemas, los cuales se explicarán con detalle más adelante:

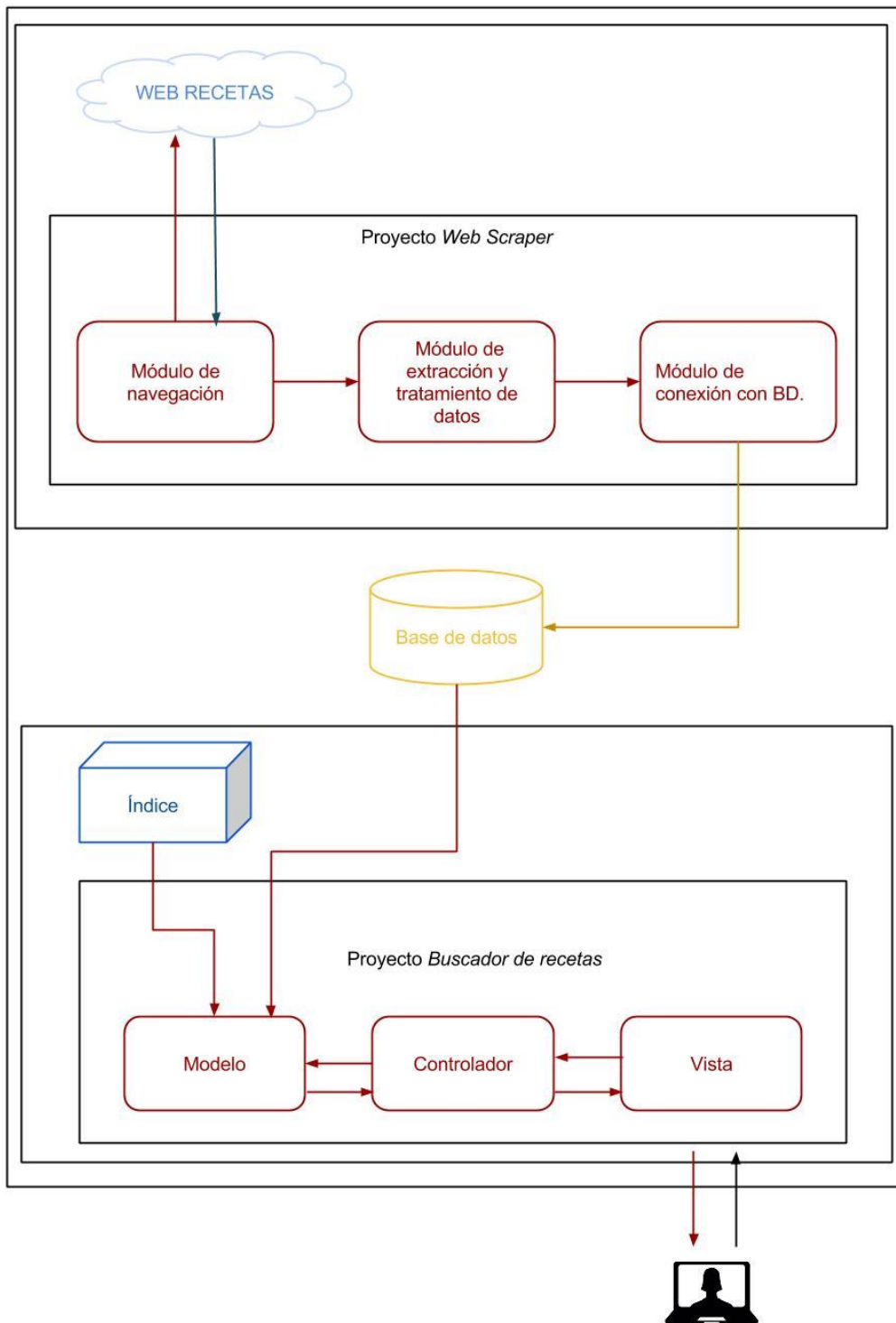


Ilustración 7: Esquema de integración de ambos sistemas.

Desarrollo de un sistema de recogida y categorización de datos de recetas

El patrón que se va a seguir durante todo este apartado es detallar una visión modular de los diseños de ambos sistemas, de tal forma que posteriormente el proyecto en su conjunto sea de fácil implementación y permita en posteriores fases realizar posibles ampliaciones de forma sencilla.

4.1.1. DISEÑO MODULAR DEL PROYECTO *WEB SCRAPER*

Este sistema se ha dividido en 3 módulos: módulo de extracción de datos y navegación, módulo de tratamiento de datos y módulo de conexión con la base de datos, tal y como se puede ver a continuación:

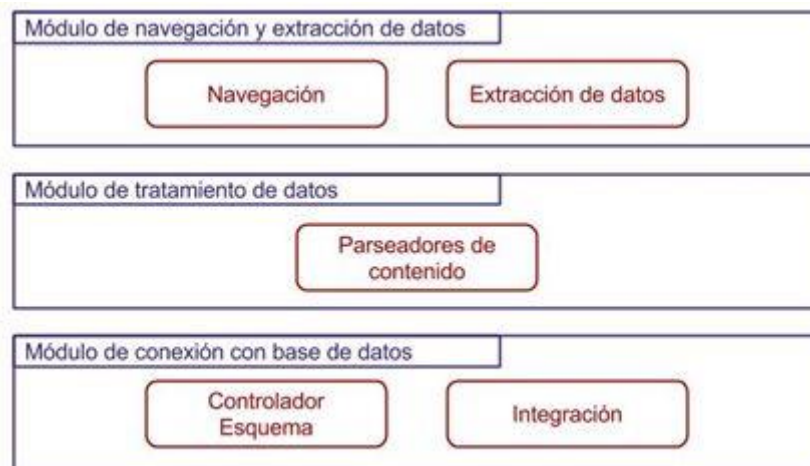


Ilustración 8: Diseño modular de la aplicación *Web Scraper*

4.1.1.1. MÓDULO DE EXTRACCIÓN DE DATOS Y NAVEGACIÓN

Este módulo abarca dos funciones principales: navegación y extracción de datos.

En la navegación se incluyen controles que permite al módulo encontrar los enlaces apropiados que facilitan el avance por el listado interno de las recetas de la página. El objetivo que se pretende conseguir es la automatización del proceso de extracción de recetas.

La extracción de datos se ha colocado como función a parte de la navegación, porque con ello se logra abstraer los posibles problemas que puedan surgir durante el proceso. Es decir, si la extracción de una receta falla, el sistema de navegación no se ve afectado por ello.

Desarrollo de un sistema de recogida y categorización de datos de recetas

4.1.1.2. MÓDULO DE TRATAMIENTO DE DATOS

Este módulo permite desechar los datos erróneos o inservibles extraídos por el módulo anteriormente explicado. Con ello se pretende optimizar este proceso, ya que puede ser un elemento clave del sistema para aquellas páginas webs, donde la información que ofrecen no está bien organizada o presenta errores.

4.1.1.3. MÓDULO DE CONEXIÓN A LA BASE DE DATOS

En este módulo se definen dos funciones principales: creación de la base de datos e inserción de la información.

Con la inclusión de esta función, se consigue tener un control del diseño de la base de datos desde el comienzo del proyecto. De esta manera, se hace más fácil las posibles mejoras de este punto en fases posteriores, ya que se puede consultar la evolución de esta funcionalidad y añade la posibilidad de hacer *rollback* en caso de que la mejora dé paso a un sistema inestable.

La función de inserción de información permite integrar al sistema con la base de datos, de tal forma que hace posible actualizar dicha base de datos con la información tratada por la aplicación.

4.1.2. DISEÑO MODULAR DEL PROYECTO BUSCADOR

Este sistema está dividido en tres módulos claramente diferenciados: módulo de presentación, módulo de lógica de negocio y módulo controlador, tal y como se muestra en la siguiente imagen:

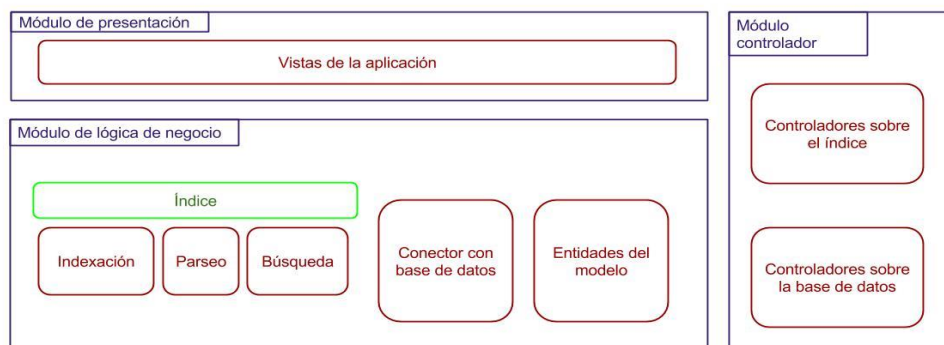


Ilustración 9: Diseño modular de la aplicación Buscador

4.1.2.1. MÓDULO DE PRESENTACIÓN

Este módulo comprende la interfaz que permite a los usuarios interactuar con la aplicación, y realizar todas las posibles acciones.

Gracias a este patrón de diseño, se puede transformar la aplicación a otros formatos optimizando el tiempo de implementación, ya sea bien como sitio web/aplicación web, o sacar diferentes versiones sin modificar la lógica de negocio, para otros lenguajes.

4.1.2.2. MÓDULO DE LÓGICA DE NEGOCIO

En la imagen se puede observar que este módulo está dividido en tres partes: índice, conector y entidades.

La función índice permite crear un índice basado en Lucene con toda la información tratada gracias a los algoritmos contenidos en el sub-módulo de parseo. También habilita la realización de búsquedas haciendo uso de los diferentes algoritmos diseñados.

El conector permite realizar búsquedas sobre la base de datos. Gracias a las funciones de índice y conector, el buscador ofrece la posibilidad de operar con cualquiera de los dos motores de obtención de información.

Por último, las entidades del modelo, permiten a la aplicación trabajar con información estándar y son las encargadas de contener los datos extraídos del motor.

4.1.2.3. MÓDULO CONTROLADOR

Para lograr que la aplicación trabaje sobre los dos motores de obtención de información, es necesario diseñar a los controladores con dos funciones principales: controlador sobre el índice y sobre la base de datos.

De esta manera, es capaz de transmitir al módulo de presentación la información para cualquiera de las dos formas y de garantizar cierta independencia de los mismos.

4.2. DISEÑO DE LA ESTRUCTURA DE DATOS

Como se ha mencionado en el apartado de las tecnologías escogidas, la base de datos está basada en SQLite y en ningún caso esta se verá afectada a cambios en el modelo estructural por la interacción con el usuario.

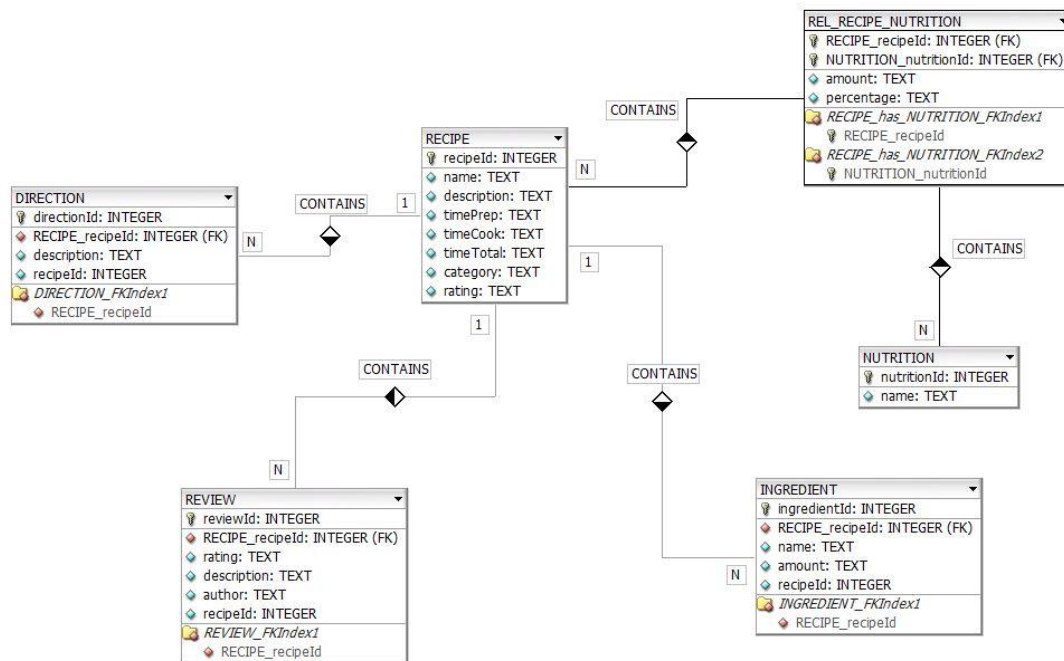
Desarrollo de un sistema de recogida y categorización de datos de recetas

Por tanto, la base de datos debe ser capaz de almacenar grandes cantidades de información sin perder la consistencia relacional entre tablas en el proceso de inserción de los datos extraídos de la web.

Las tablas del esquema contendrán información relevante sobre:

- Información importante de las recetas, como: nombre, descripción, tiempos de preparación, etc.
- Información adicional sobre las recetas como: ingredientes, pasos, valor nutricional.
- Información asociada con las opiniones de los usuarios sobre las recetas.

Para el modelo de negocio al que se enfoca el proyecto, se propone el siguiente modelo de base de datos:



4.2.1. TABLA RECIPe

La tabla *Recipe* es la principal y más importante del modelo. Está orientada a guardar toda la información útil de la receta para cualquier página web orientada a recetas. Se puede observar que, a excepción del identificador único de la receta, todos los campos de esta tabla son de tipo *text*, ya que es el campo menos restrictivo a la hora de insertar información. Por otro lado, esta tabla, sirve de precursora para la posterior creación, en

Desarrollo de un sistema de recogida y categorización de datos de recetas

una segunda fase del proyecto, de otras aplicaciones como buscadores o recomendadores.

A continuación se ofrece un listado explicativo de los distintos campos que forman la tabla:

recipeId: El campo `recipeId` permite identificar a cada una de las recetas y es la clave primaria de esta tabla.

name: Este campo hace referencia al nombre de la receta.

description: Es un campo de tipo texto que contiene la descripción detallada de la receta.

timePrep: Este campo permite guardar el tiempo de preparación previo a la realización de la receta.

timeCook: Este campo contiene los tiempos de realización de la receta comprendidos entre el primer paso y el último.

timeTotal: Este campo hace referencia el tiempo total que se tardaría en tener lista una receta.

category: Este campo permite la catalogación de las recetas en todas las categorías extraídas de la página web de cocina.

rating: Este campo almacena la valoración general que los usuarios han dado a la receta.

4.2.2. TABLA DIRECTION

Normalmente una receta tendrá más de un paso para su elaboración, por ello se ha creado la tabla *Direction*, que almacena la descripción del paso (**description**), junto con un identificador único de dicho paso (**directionId**).

Con esta tabla y las explicadas más abajo, la información está correctamente distribuida y no se recarga innecesariamente de atributos la clase principal *Recipe*.

4.2.3. TABLA REVIEW

La información se puede extraer de páginas web que están muy vivas en cuanto a usuarios se refiere, por ello se ha creado una tabla que recoge las opiniones de los usuarios para cada receta. Esto permite para futuras ampliaciones del proyecto, la adaptación de funcionalidades orientadas a la creación de un recomendador.

Desarrollo de un sistema de recogida y categorización de datos de recetas

Los campos de esta tabla son los siguientes:

author: Este campo permite saber qué usuario es el que ha escrito el comentario de valoración.

rating: Este campo almacena el valor de la nota de valoración que le ha dado el usuario a la receta.

description: Este campo ofrece la descripción de la valoración del usuario.

4.2.4. TABLA INGREDIENT

La tabla *Ingredient* permite almacenar todos los ingredientes de la receta, guardando para cada uno de ellos el nombre (**name**) y la cantidad (**amount**). Se ha pensado como relación 1:n con la tabla *Recipe*, ya que favorece el proceso de extracción e inclusión de la información de la web a la base de datos.

4.2.5. TABLA NUTRIENT

En algunas páginas web de cocina, las recetas llevan asociadas unos valores nutricionales muy útiles. La tabla *Nutrient* está pensada como tabla maestra para los valores nutricionales de la web, por lo que únicamente se almacenan los nombres de dichos valores (**name**), guardándose en otra tabla la cantidad de estos para cada receta.

4.2.6. TABLA RECIPE_NUTRITION

La forma en la que se indica la cantidad de los valores nutricionales para una receta es a través de la relación de la tabla *Recipe* con la tabla *Nutrition*.

La relación *Recipe_Nutrition* está pensada para optimizar las consultas de búsqueda de los valores nutricionales de las recetas. Ya que en este caso, lo que varía para cada receta son las cantidades (**amount**) y porcentajes (**percentage**) de los valores nutricionales.

5. IMPLEMENTACIÓN

En esta sección se ofrece un análisis de las características implementadas para los sistemas implementados en este trabajo y que se han descrito en el capítulo anterior.

5.1. IMPLEMENTACIÓN DEL PROYECTO *WEB SCRAPER*

La implementación de este sistema se ha llevado a cabo por separado en tres módulos diferentes. La elección de este método de implementación se debe principalmente a que el proyecto puede verse sometido a fuertes ampliaciones, por lo que resulta necesario abstraer cada uno de los módulos que forman la aplicación.

5.1.1. EXPLICACIÓN DE LA IMPLEMENTACIÓN MODULAR

En la imagen inferior se puede ver la estructura organizativa de paquetes que se ha seguido durante esta fase:

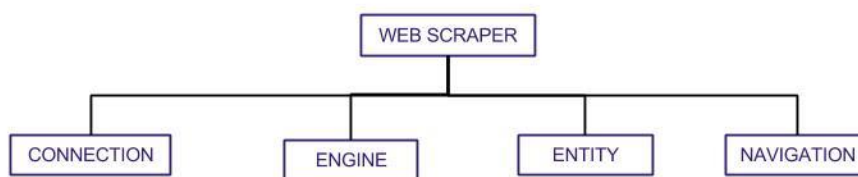


Ilustración 10: Implementación modular del proyecto *Web Scraper*

5.1.1.1. PAQUETE CONNECTION

Implementa la lógica que permite crear y administrar el objeto de base de datos donde se va a introducir toda la información extraída, y está compuesto de los siguientes componentes:

- La interfaz *Connector* permite definir un patrón de implementación para este módulo. De esta forma, se habilita una vía para posibles integraciones con otros motores de bases de datos en futuras ampliaciones.
- La clase *SQLiteConnection* hace uso de la interfaz explicada en el punto superior para facilitar los métodos necesarios que permitan administrar el objeto de base de datos.

5.1.1.2. PAQUETE ENGINE

Este paquete implementa la funcionalidad de extracción de datos del sitio web de recetas. Los componentes que lo forman son los siguientes:

- *ScraperEngine* es una interfaz que hace la función de modelo para los diferentes tipos de motores de extracción que se pueden implementar. Permite por tanto, la extracción de información de cualquier página web orientada al mundo de la cocina.
- *AllRecipesEngine* es la clase implementada que ofrece los métodos necesarios para la recopilación de toda la información de la web *Allrecipes.com*.

5.1.1.3. PAQUETE ENTITY

Las clases de este paquete ofrecen los objetos y métodos necesarios para contener, de forma estructurada, toda la información extraída de la web. Este módulo es común para ambos proyectos, ya que de esta manera se evitan problemas de compatibilidad de datos entre ambos sistemas.

Como se puede ver en la siguiente imagen, existe una clase por cada entidad de la base de datos expuesta en el apartado anterior.



Ilustración 11: Clases de las entidades comunes del proyecto

5.1.1.4. PAQUETE NAVIGATION

El paquete *navigation* ofrece una serie de componentes que permiten la navegación por el sitio web del que se está llevando a cabo la extracción de datos.

- La interfaz *ScraperNavigation* representa el patrón de implementación de este subsistema.
- La clase *AllRecipesNavigator* permite la navegación por el sitio web de recetas *AllRecipes.com*.

5.1.2. EXPLICACIÓN DE LOS PROCESOS Y ALGORITMOS UTILIZADOS

En esta sección se detalla el procedimiento de extracción de la información haciendo uso del siguiente sistema de caja blanca:

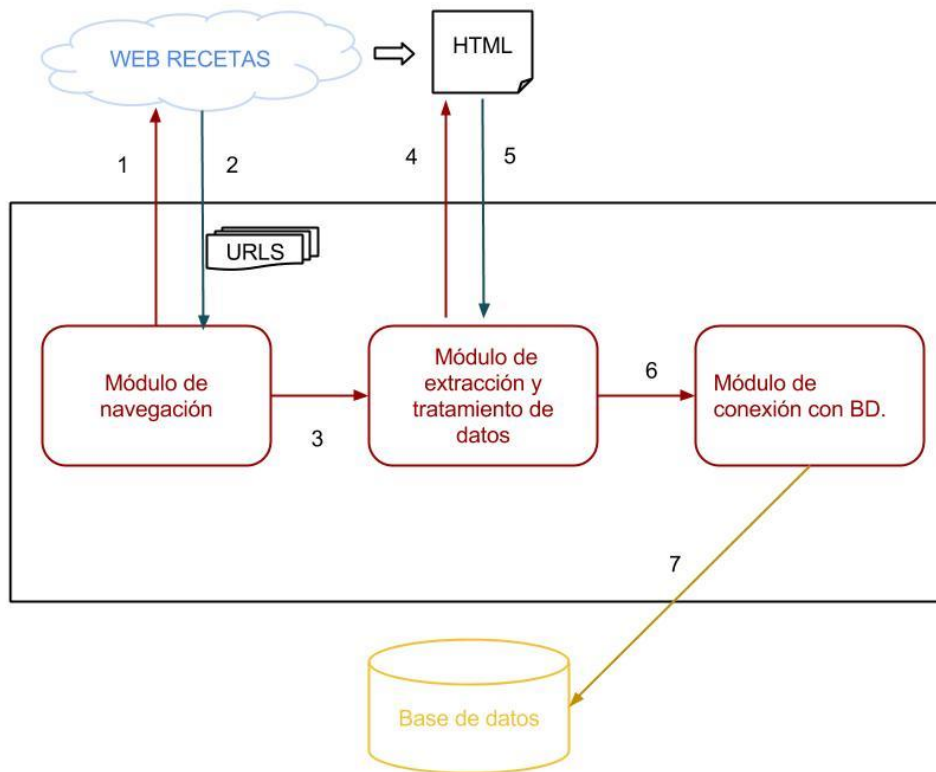


Ilustración 12: Proceso de extracción de la información.

En la imagen superior, se puede apreciar la organización modular del proceso de extracción de información. A continuación se ofrece una explicación de los puntos señalados en el gráfico:

- 1) En primera instancia, el módulo de navegación solicita en serie todas las direcciones de categorías, haciendo uso del método *Jsoup.get*.
- 2) Una vez obtenidas las categorías, realiza un recorrido del contenido *HTML* de cada una de ellas, para obtener todas las direcciones de las recetas de la página.
- 3) Cada dirección encontrada se va transfiriendo al módulo de extracción de datos, que será el encargado de obtener toda la información relacionada de dicha receta.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- 4) El módulo de extracción utiliza los métodos selectores *JQuery* de la *API* de *JSoup* que permiten sacar cada uno de los campos necesarios de la receta.
- 5) La información obtenida se trata de manera que pueda ser insertada en la base de datos.
- 6) La información tratada es transferida al módulo de navegación haciendo uso de los objetos del módulo entidad.
- 7) Por último, se establecen las conexiones necesarias para la persistencia de la información extraída de la web y posteriormente tratada.

A continuación se deja en forma de pseudocódigo el procedimiento de obtención de información explicado en esta sección. Para ayudar al lector, se han marcado en rojo los pasos explicados en el párrafo anterior.

```
1  INICIO PROGRAMA
2
3  getCategoryUrls 1
4
5  PARA CADA url HACER
6
7      page = obtainCategoryHtml(url) 1
8
9  HASTA LLEGAR AL FINAL HACER
10
11      getRecipeUrlsFromPage(page) 2
12
13      PARA CADA url ENCONTRADA HACER 3
14
15          r = obtainRecipe(url) 4,5
16          insertRecipe(r) 6,7
17
18      FIN PARA
19
20      page = getNextPage(page) 2
21
22  FIN HASTA
23
24  FIN PARA
25
26  FIN PROGRAMA
```

Ilustración 13: Algoritmo de extracción de información.

5.2. IMPLEMENTACIÓN DEL PROYECTO BUSCADOR

Este sistema se ha implementado siguiendo un patrón *MVC* haciendo uso del *framework* de Java Swing.

Desarrollo de un sistema de recogida y categorización de datos de recetas

En la siguiente imagen se observa la organización de módulos del patrón implementado:

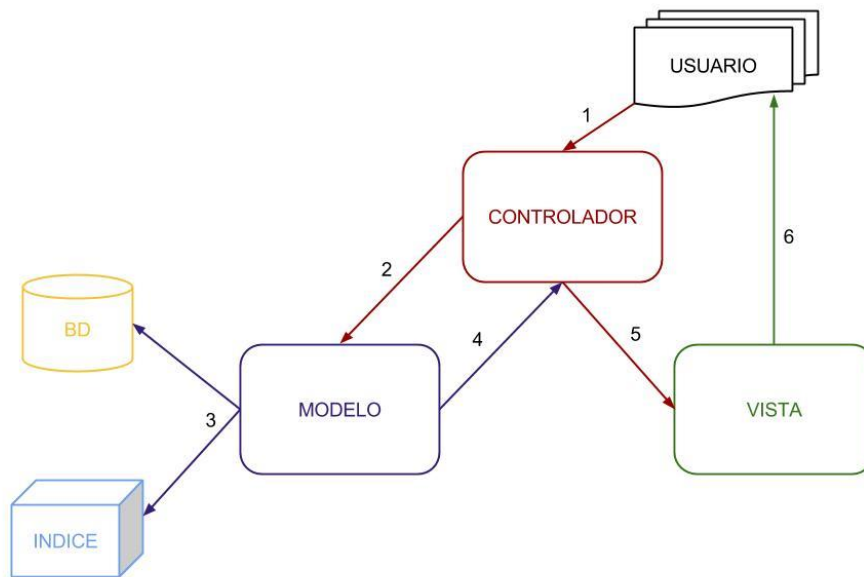


Ilustración 14: Patrón de implementación del proyecto Buscador.

Los pasos del proceso de interacción del usuario con la aplicación son los siguientes:

- 1) El usuario realiza una petición de búsqueda a la aplicación, y esta llega en primera instancia al controlador del sistema.
- 2) El controlador transfiere la petición al modelo y se queda a la espera de recibir los resultados.
- 3) El modelo trata en primer lugar la petición para después transferirla a alguno de los dos sistemas de obtención de información disponibles (La elección sobre a qué sistema consultar depende de la configuración interna del buscador).
- 4) Una vez obtenidos los datos del sistema, el modelo los trata para que puedan ser devueltos al controlador.
- 5) El controlador modifica el contenido de la vista, insertando en ella la información solicitada, dejándola preparada para la posterior consulta por el usuario.
- 6) Finalmente la información se hace visible por medio de la vista, quedando el sistema a la espera de una nueva solicitud por parte del usuario.

5.2.1. EXPLICACIÓN DE LA IMPLEMENTACIÓN MODULAR

Para la implementación de este sistema, se han repartido las tareas de cada módulo de la siguiente manera:

Desarrollo de un sistema de recogida y categorización de datos de recetas

- El modelo contiene las clases que facilitan el acceso a datos para las dos formas posibles, y la lógica de negocio, que son todas las clases necesarias para el manejo de las recetas, ingredientes, pasos, etc.
- La vista alberga todo el código que permite visualizar por pantalla la información, por lo que está poblado por objetos relacionados con la librería que proporciona Java Swing: *JFrame*, *JTable*, *JButton*, *JTextField*, *JLabel*, etc.
- El módulo del controlador contiene una clase controladora para cada vista, y estas hacen de elemento de unión entre el modelo y la vista.

5.2.1.1. MODELO

Para la arquitectura en la que se está basando la implementación, el modelo se encarga de representar la lógica de negocio.

La organización de los paquetes y clases de este módulo es muy importante, ya que es necesario conseguir que la aplicación sea lo más escalable posible, de forma que se facilite las futuras incorporaciones que pueden llevarse a cabo. A continuación se muestra la estructura organizativa del modelo de la aplicación:

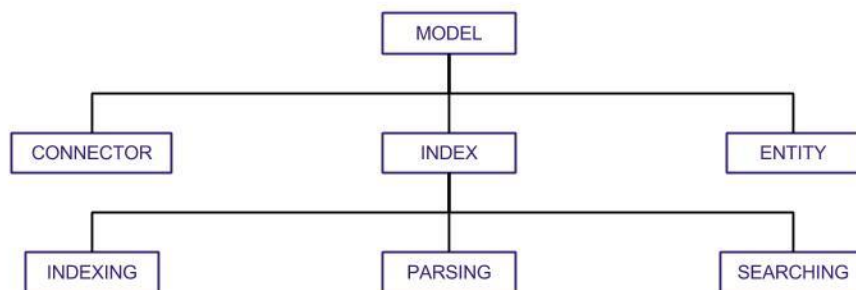


Ilustración 15: Patrón de implementación del modelo.

5.2.1.1.1. PAQUETE CONNECTOR

El paquete *connector* implementa las clases que permiten extraer información de la base de datos por lo que permite las búsquedas por consultas. Al contrario que en el caso del sistema anterior, el buscador no permite llevar a cabo actualizaciones de ningún tipo, ya que únicamente se utiliza para la lectura y recuperación de información.

5.2.1.1.2. PAQUETE INDEX

El paquete *index* implementa toda la lógica asociada al índice que utiliza la aplicación para la recuperación de información. Sigue el diseño visible en la imagen superior ya

que permite en fases posteriores del proyecto la inclusión de un recomendador en la lógica de negocio de forma fácil y sencilla.

A continuación se explica las funcionalidades implementadas en los sub-módulos que lo componen:

- El paquete *indexing* contiene la lógica asociada a la creación de un índice a partir de una base de datos. Consta de tres elementos:
 - La interfaz *Index* permite tener un modelo base para la implementación de indexadores.
 - La clase *LuceneIndexer* que implementa la interfaz *Index* es el indexador implementado para este proyecto, hace uso de la API de *Lucene* para la creación de un índice.
 - La clase *Posting* permite obtener la frecuencia de aparición de un determinado término en los documentos.
- El paquete *parsing* contiene la lógica asociada al tratamiento de la información antes de ser indexada en el índice. Consta de tres elementos:
 - La interfaz *TextParser* permite tener un modelo base para la implementación de parseadores.
 - La clase *HtmlSimpleParser* facilita el tratamiento de la información de un documento *html* de una página determinada.
 - La clase *StopWordParser* es la clase utilizada en el proyecto para la eliminación de *StopWords*.
- El paquete *searching* contiene la lógica asociada a la búsqueda en un índice creado previamente. Consta de los siguientes elementos:
 - La interfaz *Searcher* permite tener un modelo base para la implementación de buscadores para el proyecto.
 - La clase *LuceneSearcher* que implementa la interfaz *Searcher* es el buscador implementado para las búsquedas en el índice.

5.2.1.1.3. PAQUETE ENTITY

El paquete *entity* facilita el manejo de la información a través de los diferentes módulos de la aplicación, y tal y como se ha explicado en la sección anterior, es común para los dos sistemas de este proyecto.

5.2.1.2. VISTA

La vista hace uso de la información proporcionada por el modelo para representar la aplicación.

En este módulo se han implementado varias vistas, una para cada pantalla de la aplicación, las cuales se exponen a continuación junto con una breve explicación:

- *DefaultView*: Forma la pantalla inicial de la aplicación, y permite navegar a los diferentes métodos de búsqueda que ofrece la aplicación.
- *BasicSearchView*: Muestra la visualización del buscador sencillo de la aplicación.
- *AdvancedSearchView*: Esta vista ofrece la visualización del buscador avanzado de la aplicación.
- *PredefinedSearchView*: Esta vista ofrece la visualización del buscador predefinido de la aplicación.
- *ResultsView*: La vista *ResultsView* muestra los resultados devueltos por la aplicación, se muestra tras efectuar una búsqueda en cualquiera de los dos métodos disponibles.
- *RecipeView*: Gracias a esta vista, se puede efectuar una visualización detallada de las recetas.

5.2.1.3. CONTROLADOR

El controlador actúa como módulo intermediario entre el modelo y las vistas. Este módulo es el que se encarga de decidir qué vista se muestra al usuario en cada momento, junto con los datos apropiados.

Para este proyecto se han implementado los siguientes controladores:

- La interfaz *IController* se ha implementado con la idea de generar un patrón inicial que heredan todas las vistas que forman la aplicación.
- El controlador *DefaultController* se encarga de administrar la vista de la pantalla inicial, y su función principal es controlar la navegación a las vistas búsqueda.
- *BasicSearchController* es la clase implementada con el objetivo de controlar las posibles acciones que puede realizar el usuario en la vista de búsqueda básica.
- El controlador *AdvancedSearchController* se encarga de gestionar la pantalla de búsquedas avanzadas de la aplicación.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- El controlador *PredefinedSearchController* se encarga de gestionar la pantalla de búsquedas predefinidas de la aplicación.
- La función principal del controlador *ResultsController* es la de insertar los resultados de la búsqueda efectuada en la vista *ResultsView* y controlar los botones que permiten la navegación a otras pantallas.
- El controlador *RecipeController* administra la vista que permite visualizar en detalle una receta determinada, por lo que se encarga de insertar la información apropiada en dicha vista y controlar las acciones que permiten la navegación a otras pantallas.
- La clase *Preferences* contiene la información importante de la aplicación, como: la localización del índice, la ruta al archivo binario que forma la base de datos, el sistema a utilizar para la consulta de los datos, etc.

5.2.2. EXPLICACIÓN DE LOS PROCESOS Y ALGORITMOS IMPLEMENTADOS

Para este sistema, se ha tenido en cuenta las dos funcionalidades principales que abarca: acceso a la información mediante dos formas posibles, por índice y base de datos, y actualización del índice.

5.2.2.1. ACCESO A LA INFORMACIÓN

Para la explicación de este proceso, se ha esbozado un sistema de caja blanca en el que se pueden ver los módulos que intervienen en este procedimiento.

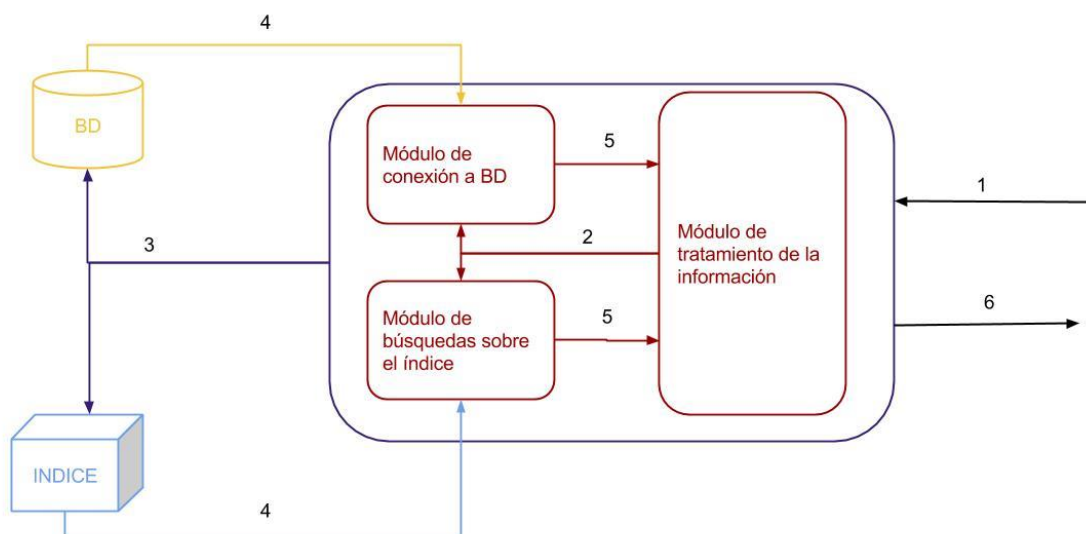


Ilustración 16: Acceso a la información.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- 1) La petición llega del controlador y es transferida al módulo de tratamiento de la información.
- 2) Si la disposición interna del buscador, en relación al tipo de motor utilizado, está configurada para extraerse del índice, la petición tratada se pasa al módulo de búsquedas sobre el índice, en caso contrario se pasa al módulo de conexión con la base de datos.
- 3) Ya sea uno u otro, se realiza una búsqueda de contenido sobre los datos almacenados.
- 4) La información devuelta es transferida al módulo correspondiente.
- 5) Dicha información es procesada por el módulo de tratamiento de manera que pueda ser devuelta al controlador.
- 6) Finalmente los resultados son devueltos al controlador.

5.2.2.2. CREACIÓN YACTUALIZACIÓN DEL ÍNDICE

En la siguiente imagen se pueden ver los sub-módulos del modelo que intervienen en este proceso:

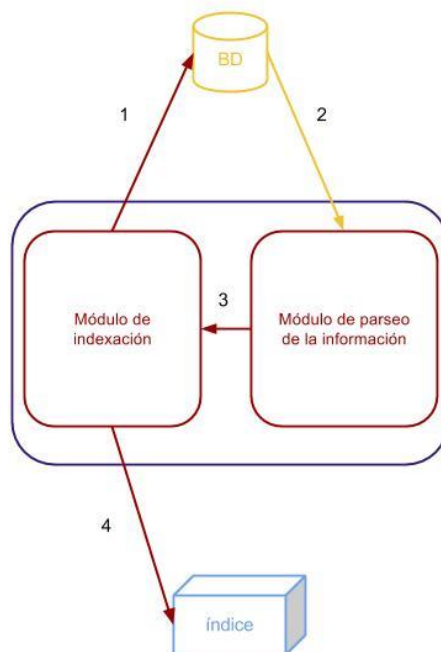


Ilustración 17: Proceso de creación y actualización del índice.

- 1) En primer lugar, el módulo de indexación solicita toda la información de cada una de las recetas a la base de datos por medio de la librería de SQLite.

- 2) Los datos son obtenidos y transferidos al módulo de parseo de la información, que elimina en primer lugar los caracteres especiales, como acentos, comas puntos, etc.
- 3) Una vez eliminados los caracteres especiales, se quitan los *stopwords* de la información extraída de la base de datos y se transfiere dicha información al módulo de indexación.
- 4) Por último, el módulo de indexación crea un documento de Lucene añadiendo cada dato en un campo, replicando de esta manera el diseño relacional de la base de datos. Una vez configurado el documento, este se añade al índice.

6. PRUEBAS

En esta sección se exponen las pruebas realizadas para las dos aplicaciones que forman el proyecto, por lo que la presentación de estas pruebas se ofrece de forma separada para cada sistema.

A continuación se da una explicación, en orden, de las pruebas efectuadas:

- I. Pruebas unitarias o de componente: Estas pruebas se han realizado durante el desarrollo de cada módulo. Se han diseñado de tal forma que permiten verificar que cada componente está implementado de forma robusta. Por lo que además de comprobar las funcionalidades de cada módulo se verifica que las clases soportan el ingreso de datos erróneos.
- II. Pruebas de integración: Al igual que el caso anterior, estas pruebas son realizadas durante la fase de implementación. Permiten verificar que el intercambio de información entre elementos y módulos se efectúa correctamente.
- III. Pruebas de sistema: Las pruebas de sistema verifican que la aplicación funciona de forma correcta en los entornos, hardware y software donde es integrada. Permite descubrir al equipo de trabajo ciertas limitaciones de software de la aplicación implementada.
- IV. Pruebas de validación: Este tipo de pruebas son efectuadas tras la implementación del sistema, y verifican que la funcionalidad total de la aplicación cumple con la especificación de requisitos funcionales y no funcionales definidos durante la fase de análisis.
- V. Pruebas de aceptación: Las pruebas de aceptación comprueban que el producto final que se monta en el entorno de preproducción se ajusta a los requerimientos del usuario. Cuando estas pruebas son ejecutadas directamente en la infraestructura del cliente, se denominan pruebas *Beta*, mientras que las pruebas que son efectuadas en ambientes proporcionados por la empresa desarrolladora, se denominan pruebas *Alpha*.

6.1. PRUEBAS SOBRE EL PROYECTO WEB SCRAPER

En esta sección se detallan las pruebas efectuadas sobre el sistema de recogida de datos. Se ha seguido una filosofía ascendente en relación a su ejecución, comenzando con las pruebas unitarias y finalizando con las de aceptación y validación.

6.1.1. PRUEBAS UNITARIAS: PROYECTO WEB SCRAPER

A continuación se describen las pruebas más relevantes, divididas por cada módulo de la aplicación.

- 1) Pruebas sobre la clase encargada de la navegación por las páginas del sitio web de recetas.
 - a) Se comprueba que la obtención de las direcciones de las categorías a través del documento JSoup se efectúa de forma correcta y sin problemas.
 - b) Se valida que el módulo es capaz de avanzar sobre el listado de páginas que muestran las recetas, además de acceder a la siguiente página de dicho listado sin problemas.
 - c) El módulo es capaz de obtener las direcciones de las recetas del *HTML* de una página contenida en el documento de la librería de JSoup.
 - d) Para todas las funcionalidades anteriores, se valida que el módulo es capaz de tratar información errónea o nula.

- 2) Pruebas sobre la clase encargada de la extracción y tratamiento de los datos del sitio web de recetas.
 - a) Se verifica que el módulo es capaz de extraer e insertar en un documento de JSoup el código *HTML* de la página a partir de una *URL*.
 - b) El módulo es capaz de obtener los datos básicos de una receta, a partir del código *HTML* de la página de dicha receta.
 - c) Se comprueba que es capaz de obtener todos los campos básicos de cada ingrediente asociado a la receta.
 - d) El módulo de extracción de datos recoge toda la información asociada a cada uno de los pasos de una receta.
 - e) Se valida que es capaz de extraer todos los datos relacionados con los comentarios de los usuarios sobre una receta.

- f) Se obtienen de forma correcta todos los valores nutricionales asociados a una receta.
 - g) Para todas las funcionalidades anteriores, se verifica que el módulo es capaz de tratar con información errónea o nula.
- 3) Pruebas sobre la clase encargada de la conexión a la base de datos.
- a) El esquema de base de datos se crea de forma correcta, generando un archivo binario de datos en la ruta especificada.
 - b) La conexión y desconexión a la base de datos se efectúa correctamente.
 - c) Se inserta correctamente un registro asociado a la receta con toda la información que esta contiene: nombre, descripción, tiempos, valoraciones, etc.
 - d) El módulo es capaz de insertar los ingredientes relacionados con una receta en la base de datos.
 - e) Se comprueba que los pasos para la realización de una receta se insertan de manera correcta en la base de datos.
 - f) Las opiniones de las recetas son insertadas sin problemas en la base de datos.
 - g) Se verifica que los valores nutricionales de las recetas son insertadas correctamente en la base de datos.
 - h) Para todas las funcionalidades anteriores, se valida que el módulo es capaz de tratar información errónea o nula.

6.1.2. PRUEBAS DE INTEGRACIÓN: PROYECTO WEB SCRAPER

A continuación se exponen las pruebas más significativas efectuadas en esta fase:

- 1) Se extrae el *HTML* de la página principal del sitio web de recetas y se transfiere de forma correcta el documento al módulo de navegación.
- 2) El módulo de navegación procesa de forma correcta el documento con el contenido *HTML* y extrae de este todas las *URLs* de las categorías. Las *URLs* son transferidas sin problemas al módulo de extracción de datos.
- 3) Se comprueba que la extracción de datos y la transferencia de estos entre el módulo de extracción y el módulo de conexión con la base de datos se efectúa de forma correcta.

Desarrollo de un sistema de recogida y categorización de datos de recetas

- 4) El módulo de navegación es capaz de trabajar integrado con el módulo de extracción de información y el módulo de conexión a la base de datos, de tal forma que es capaz de extraer todas las recetas de la página.
- 5) Se verifica que la información que se persiste en la base de datos está preparada para su uso por el sistema de búsquedas de este proyecto.

6.1.3. PRUEBAS DE SISTEMA: PROYECTO *WEB SCRAPER*

Las pruebas de esta fase se han efectuado durante la implementación de la aplicación y probado en los siguientes entornos:

- 1) Se ha verificado que la aplicación funciona correctamente sobre el SO basado en Ubuntu Linux Mint.
- 2) Se ha comprobado que la aplicación funciona sin problemas sobre el SO Mac OS X Yosemite.
- 3) Se ha validado que la aplicación funciona sobre el SO Windows 8.

6.1.4. PRUEBAS DE VALIDACIÓN: PROYECTO *WEB SCRAPER*

Se han estructurado las pruebas en dos apartados, que se exponen a continuación:

- 1) Prueba de creación del esquema relacional de la base de datos.
- 2) Prueba de inserción de datos extraídos de la web, en el esquema relacional creado en el primer apartado.

6.1.5. PRUEBAS DE ACEPTACIÓN: PROYECTO *WEB SCRAPER*

Para la verificación de las pruebas de esta fase, se ha seguido un sistema de caja negra, de tal manera que, tras la ejecución de la aplicación Web Scraper, se valide si efectivamente se ha logrado integrar todos los datos extraídos de la web en una base de datos relacional.

Estas pruebas se describen con detalle en la sección siguiente de resultados obtenidos.

6.2. PRUEBAS SOBRE EL PROYECTO BUSCADOR

Al igual que en el caso anterior, las fases de las pruebas se van a exponer de forma creciente comenzando con las pruebas unitarias, para continuar describiendo las pruebas de integración y sistema, y finalizar detallando las pruebas de validación y aceptación.

6.2.1. PRUEBAS UNITARIAS: PROYECTO BUSCADOR

A continuación se muestran las pruebas más trascendentes de cada módulo:

- 1) Pruebas sobre el módulo encargado de representar el modelo de la aplicación.
 - a) Se valida que los métodos que permiten la obtención de la información contenida en la base de datos la facilitan sin problemas, como también son capaces de tratar con datos erróneos o nulos.
 - b) El módulo es capaz de crear un índice de Lucene e insertar en este una receta extraída de la base de datos, junto con todas sus posibles relaciones. También se comprueba que es capaz de manejar información errónea sin que el índice se vea afectado.
 - c) Se comprueba que el módulo es capaz de realizar una búsqueda en el índice y devolver un listado de los resultados obtenidos. También se verifica que es capaz de tratar con búsquedas que no devuelven ningún resultado.
 - d) El módulo es capaz de trabajar la información de forma que elimina los *stopwords* de una frase.

- 2) Pruebas sobre el módulo encargado de representar al controlador de la aplicación.
 - a) Se verifica que los escuchadores controlan correctamente las posibles acciones posibles.
 - b) Los métodos que manejan la información a visualizar, la tratan correctamente y son capaces de trabajar con datos erróneos o nulos.
 - c) El módulo es capaz de ocultar y mostrar las ventanas apropiadas en cada caso de tal forma que la navegación de la aplicación sea la correcta.

- 3) Pruebas sobre el módulo encargado de representar a la vista de la aplicación.
 - a) Cada una de las vistas pintan los objetos contenidos en ellas.
 - b) Las vistas son capaces de controlar los posibles objetos erróneos sin comprometer a la visualización de los datos.

6.2.2. PRUEBAS DE INTEGRACIÓN: PROYECTO BUSCADOR

A continuación se ofrecen las pruebas más representativas de esta fase:

- 1) El controlador es capaz de comunicarse con la vista para mostrar la pantalla de inicio en el arranque de la aplicación. También es capaz de tratar todas las posibles acciones que le llegan de la vista, de manera que dependiendo de la operación realizada por el usuario, el controlador muestre la vista correspondiente.
- 2) Dependiendo del tipo de motor de búsqueda configurado en las preferencias, SQLite o Lucene, los datos de búsqueda introducidos por el usuario son transferidos sin problemas por medio del controlador al modelo.
- 3) El controlador es capaz de transferir a la vista correspondiente los resultados generados por el modelo, y esta a su vez ofrece dichos datos al usuario.
- 4) El controlador es capaz de manejar el resto de módulos de la aplicación de manera que el proceso de búsqueda de recetas pueda realizarse tantas veces como el usuario desee.

6.2.3. PRUEBAS DE SISTEMA: PROYECTO BUSCADOR

Al igual que en el proyecto *Web Scraper*, el buscador se ha probado en los siguientes entornos:

- 1) Se ha verificado que la aplicación funciona correctamente sobre el SO basado en Ubuntu Linux Mint.
- 2) Se ha comprobado que la aplicación funciona sin problemas sobre el SO Mac OS X Yosemite.
- 3) Se ha validado que la aplicación funciona sobre el SO Windows 8.

6.2.4. PRUEBAS DE VALIDACIÓN: PROYECTO BUSCADOR

En esta fase de las pruebas, se ha verificado que la aplicación implementada efectivamente cumple con los requisitos funcionales que se aprobaron en el análisis del proyecto. Para ello se ha comprobado que el sistema cumple con los siguientes puntos:

- 1) Es capaz de crear un índice de Lucene a partir de la base de datos creada en una ejecución previa del sistema *Web Scraper*.

- 2) Es capaz de realizar búsquedas sobre el índice de Lucene y devolver los resultados recomendados en orden descendente.
- 3) El sistema también es capaz de realizar búsquedas sobre la base de datos y devolver los resultados para su posterior visualización.
- 4) La aplicación permite visualizar un listado previo de todas las recetas recomendadas, así como un informe detallado de cada una de las recetas.

6.2.5. PRUEBAS DE ACEPTACIÓN: PROYECTO BUSCADOR

El objetivo de esta fase es verificar que el proyecto de búsqueda de recetas funciona de forma correcta y realmente se ajusta a los requerimientos del cliente. Para ello se han diseñado dos entornos de pruebas que permitirá posteriormente realizar un estudio del grado de aceptación del producto.

- 1) Pruebas llevadas a cabo por el cliente en un entorno controlado de desarrollo: El objetivo de estas primeras pruebas de preproducción son la primera aceptación del producto antes de permitir el uso de la aplicación en el entorno de trabajo del cliente y sin observadores del grupo desarrollador.
- 2) Pruebas en el entorno de trabajo del cliente con un grupo de usuarios: Estas pruebas consisten en mostrar el uso de la aplicación para los usuarios y evaluar la opinión del grupo sobre la herramienta a través de una encuesta.

Las pruebas de ambos entornos se han llevado a cabo de manera favorable y sin problemas. El estudio del grado de aceptación de la herramienta por parte de los usuarios para el segundo entorno se describe en la siguiente sección del presente documento, mostrando conclusiones y gráficas de los resultados obtenidos.

7. RESULTADOS

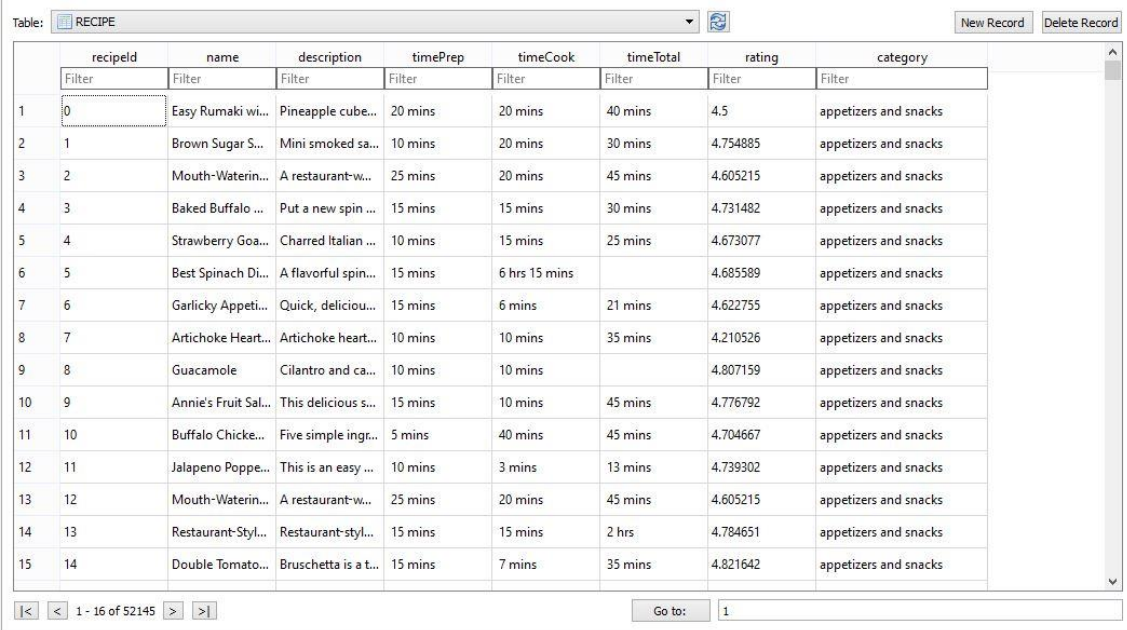
En esta sección se exponen los resultados obtenidos de las pruebas de aceptación para las dos aplicaciones detalladas en la sección anterior.

7.1. ALMACENAMIENTO DE LA INFORMACIÓN

En esta sección se muestran los resultados de los dos motores utilizados durante este proyecto para el almacenamiento y la obtención de la información.

7.1.1. OBJETO DE BASE DE DATOS

Para verificar que la aplicación *Web Scraper* ha extraído e insertado con éxito todas las recetas en el modelo relacional, se ha utilizado la aplicación SQLiteBrowser [14] como elemento de verificación de la estructura y de los datos del objeto resultante:



	recipeld	name	description	timePrep	timeCook	timeTotal	rating	category
1	0	Easy Rumaki wi...	Pineapple cube...	20 mins	20 mins	40 mins	4.5	appetizers and snacks
2	1	Brown Sugar S...	Mini smoked sa...	10 mins	20 mins	30 mins	4.754885	appetizers and snacks
3	2	Mouth-Waterin...	A restaurant-w...	25 mins	20 mins	45 mins	4.605215	appetizers and snacks
4	3	Baked Buffalo ...	Put a new spin ...	15 mins	15 mins	30 mins	4.731482	appetizers and snacks
5	4	Strawberry Goa...	Charred Italian ...	10 mins	15 mins	25 mins	4.673077	appetizers and snacks
6	5	Best Spinach Di...	A flavorful spin...	15 mins	6 hrs 15 mins		4.685589	appetizers and snacks
7	6	Garlicky Appeti...	Quick, deliciou...	15 mins	6 mins	21 mins	4.622755	appetizers and snacks
8	7	Artichoke Heart...	Artichoke heart...	10 mins	10 mins	35 mins	4.210526	appetizers and snacks
9	8	Guacamole	Cilantro and ca...	10 mins	10 mins		4.807159	appetizers and snacks
10	9	Annie's Fruit Sal...	This delicious s...	15 mins	10 mins	45 mins	4.776792	appetizers and snacks
11	10	Buffalo Chicke...	Five simple ingr...	5 mins	40 mins	45 mins	4.704667	appetizers and snacks
12	11	Jalapeno Poppe...	This is an easy ...	10 mins	3 mins	13 mins	4.739302	appetizers and snacks
13	12	Mouth-Waterin...	A restaurant-w...	25 mins	20 mins	45 mins	4.605215	appetizers and snacks
14	13	Restaurant-Styl...	Restaurant-styl...	15 mins	15 mins	2 hrs	4.784651	appetizers and snacks
15	14	Double Tomato...	Bruschetta is a t...	15 mins	7 mins	35 mins	4.821642	appetizers and snacks

Ilustración 18: Total de recetas obtenidas.

Como se puede observar en la imagen superior, el número total de recetas obtenidas por la aplicación ha sido de 52.145, por lo que se puede afirmar que se ha completado satisfactoriamente la obtención previa de la información que utiliza el proyecto buscador.

7.1.2. INDICE CREADO A PARTIR DE LA BASE DE DATOS

Para verificar que el índice creado a partir de la base de datos es completo y coherente, se ha utilizado el proyecto Luke [15] que permite visualizar el contenido del índice. Principalmente, se han tenido en cuenta dos puntos:

- Se han indexado todas las recetas contenidas en la base de datos, es decir, se ha comprobado que en el índice existen 52.145 documentos.
- Los términos de los documentos aparecen tratados, es decir, se han eliminado todos los *stop-words* que aparecen en el contenido del documento.

La imagen que se presenta a continuación muestra el resultado de estas pruebas:

Field	Idf	Norm	Value
Rating	Idfp--SV-Ni08-----	1.0	4.754985
description	Idfp--SV-Ni08-----	0.25	Mini smoked sausages disappear extra quickly blanketed crisp bacon topped brown-sugar glaze.
direction	Idfp--SV-Ni08-----	0.15625	4;Preheat oven 350 degrees F (175 degrees C).
direction	Idfp--SV-Ni08-----	0.15625	5;Cut bacon thirds wrap strip little sausage. Place wrapped sausages wooden skewers, skewer. /
direction	Idfp--SV-Ni08-----	0.15625	6;Bake bacon crisp brown sugar melted.
ingredient	Idfp--SV-Ni08-----	0.25	bacon;1 pound
ingredient	Idfp--SV-Ni08-----	0.25	little smokie sausages;1 (16 ounce) package
ingredient	Idfp--SV-Ni08-----	0.25	brown sugar, taste;1 cup
name	Idfp--SV-Ni08-----	0.5	Brown Sugar Smokies
nutrient	Idfp--SV-Ni08-----	0.1875	Calories;356 kcal;18%
nutrient	Idfp--SV-Ni08-----	0.1875	Carbohydrates;18.9 g;6%
nutrient	Idfp--SV-Ni08-----	0.1875	Cholesterol;49 mg;16%
nutrient	Idfp--SV-Ni08-----	0.1875	Fat;27.2 g;42%
nutrient	Idfp--SV-Ni08-----	0.1875	Fiber;0 g;0%
nutrient	Idfp--SV-Ni08-----	0.1875	Protein;9 g;18%
nutrient	Idfp--SV-Ni08-----	0.1875	Sodium;696 mg;28%

Ilustración 19: Contenido del índice

En la imagen superior se puede comprobar que el rango de los identificadores de los documentos se sitúa entre [0, 52.144] por lo que se puede confirmar que están todos los documentos indexados.

Se ha verificado también que la información contenida en los términos de cada documento ha sido tratada debidamente por el módulo de tratamiento de datos, tal y como se puede observar en los campos para el ejemplo de la imagen.

7.2. APLICACIÓN DE INTERACCIÓN CON EL USUARIO

Las pruebas para este apartado consisten en seleccionar un filtro de búsqueda y aplicarlo sobre el buscador desarrollado para encontrar las recetas asociadas a dicho filtro.

El filtro escogido ha sido filtrar por: “pollo (*chicken*) que no contenga cebolla (*onion*) y que se pueda preparar en menos de treinta minutos”. Por último, se ha realizado una encuesta a un grupo de usuarios con las siguientes preguntas:

- I. ¿Con que frecuencia accede a contenidos informativos en internet de carácter alimenticio? {"A diario", "cada tres días", "cada semana", "cada mes", "nunca"}
- II. ¿Los resultados devueltos satisfacen la receta que andaba buscando? {"Si", "No"}
- III. ¿Le parece interesante la posibilidad de realizar filtros más avanzados?{"Si", "No"}
- IV. ¿Echa de menos algún filtro de búsqueda que no aparezca?{"Si", "No", "No lo sé"}
- V. ¿Cree que mejora esta aplicación de búsqueda al medio que usted utiliza para encontrar recetas?{"Si", "No"}

Estos han sido los diez primeros resultados devueltos por el buscador tras la ejecución del filtro definido:

We found these recipes for you:				
Recipe id	Recipe name	Total time	Rating	Explore
128	Buffalo Chicken Wings I	25 mins	4.651351	visualize
541	Sweet Chili Thai Sauce	20 mins	4.520408	visualize
795	Vietnamese Salad Rolls	25 mins	4.548387	visualize
849	Buffalo Chicken Strips II	20 mins	4.192983	visualize
1291	Shrimp Summer Rolls ...	25 mins	3.612903	visualize
1321	Buffalo Sauce	15 mins	4.4375	visualize
1375	Jalapeno Chicken Dip	10 mins	4.321429	visualize
1601	Hot Chicken Dip	25 mins	4.136364	visualize
1752	Easy Lemon Pepper Ch...	20 mins	4.789474	visualize
1762	Orange Glazed Chicken...	25 mins	4.277778	visualize

Ilustración 20: Resultados del filtro de búsqueda.

Desarrollo de un sistema de recogida y categorización de datos de recetas

A continuación se exponen los resultados obtenidos en forma de tabla tras la realización de la encuesta. Los resultados en formato gráfico están expuestos en el anexo C del presente documento.

Id usuario	Pr I	Pr II	Pr III	Pr IV	Pr V
U1	Cada semana	Si	Si	Si	Si
U2	Cada mes	Si	Si	Si	Si
U3	Cada semana	No	Si	No lo sé	No
U4	Cada tres días	Si	Si	Si	Si
U5	Cada mes	No	Si	No lo sé	-
U6	Nunca	Si	Si	No lo sé	-
U7	Cada mes	Si	Si	No lo sé	Si

Ilustración 21: Resultados de la encuesta.

8. CONCLUSIONES Y TRABAJO FUTURO

Uno de los principales objetivos durante la realización de este proyecto ha sido el desarrollo de una aplicación de búsqueda diseñada con tecnologías novedosas en el sector, sobre un gran conjunto de datos extraídos de un sitio web de cocina.

El proyecto no ha surgido como una petición directa debido a una necesidad de un cliente en concreto, por lo que ha sido necesario un estudio previo de las tecnologías ya existentes para poder esclarecer los objetivos del proyecto y los requisitos del sistema a desarrollar. La restricción de no conocer de manera concreta las necesidades del cliente, ha propiciado al grupo desarrollador a seguir un patrón de diseño e implementación lo más modular posible, ya que de esta manera es posible una mejora continua de los sistemas implementados a medida que se dan a conocer las preferencias de los usuarios, sin perder la calidad del servicio de búsqueda que ofrece.

Finalmente las aplicaciones implementadas han sido dos: el sistema *Web Scraper*, que permite extraer la información de la web e insertarla en una base de datos, y el sistema Buscador, que interacciona con el usuario mostrando los resultados de las búsquedas sobre el índice y la base de datos.

Tras el desarrollo del proyecto se puede destacar:

- Se ha aprendido a gestionar el ciclo de vida del software y a estandarizar las pautas de cada fase.
- Se ha aprendido a diseñar e implementar nuevas técnicas y métodos tecnológicamente novedosos en aplicaciones relacionadas con el área del *Big Data* y *Business Intelligence*.
- Se ha seguido una metodología de implementación modular en ambos proyectos de manera que queden claramente diferenciados.
- El patrón utilizado de las pruebas realizadas para los dos sistemas ha sido de dentro hacia fuera, es decir, se ha comenzado por los módulos unitarios para concluir con las pruebas de aceptación de los sistemas completos.
- Se ha llevado a cabo un estudio de satisfacción del producto desarrollado por medio de una encuesta efectuada a un grupo de usuarios.

En relación a las líneas futuras de trabajo, serían viables las siguientes mejoras y adaptaciones:

- Adaptar una aplicación web que permita la integración del proyecto dentro de otras aplicaciones.
- Integrar dentro de la lógica de negocio un sistema de recomendación basado en las preferencias extraídas en este proyecto, personalizado para cada usuario.
- Adaptar el sistema *Web Scraper* de manera que sea posible la extracción de datos de varias páginas web de forma paralela.
- Desarrollar un sistema de personalización de filtros de manera que cada usuario disponga de sus búsquedas más utilizadas.
- Incorporar en la aplicación web, la opción de comentar gustos y preferencias en las redes sociales.

Finalmente se ofrece el repositorio donde se encuentran implementados los dos sistemas de este proyecto:

(sistema *WebScraper*), https://github.com/gallastegui/1415_004_SITI_RecipeScraper
https://github.com/gallastegui/1415_004_SITI_RecipeSearcher.git (sistema Buscador).

9. REFERENCIAS

- [1] BLOG INFORMÁTICA, «bloginformatica,» 22 Febrero 2013. [En línea]. Available: <http://www.bloginformatica.com.es/marketing-online/ranking-de-los-buscadores-en-enero-de-2013/>. [Último acceso: 23 Mayo 2015].
- [2] Google Inc., «Tendencias sector cocina,» [En línea]. Available: <https://www.google.es/trends/explore#q=chef>. [Último acceso: Mayo 2015].
- [3] IBM, «IBM,» Agosto 2009. [En línea]. Available: <http://www.ibm.com/developerworks/library/os-apache-lucenesearch/>. [Último acceso: 23 Mayo 2015].
- [4] jsoup HTML parse, «jsoup,» [En línea]. Available: <http://www.jsoup.org/>. [Último acceso: Mayo 2015].
- [5] Oracle, «docs.oracle,» [En línea]. Available: <http://docs.oracle.com/javase/7/docs/api/javax/xml/xpath/package-summary.html>. [Último acceso: Mayo 2015].
- [6] Gargoyle Software Inc, «Htmlunit,» [En línea]. Available: <http://htmlunit.sourceforge.net/>. [Último acceso: Mayo 2015].
- [7] Apache Lucene, «Apache Lucene,» [En línea]. Available: <https://lucene.apache.org/core/>. [Último acceso: Mayo 2015].
- [8] Sphinx Technologies Inc., «Sphinx,» [En línea]. Available: <http://sphinxsearch.com/>. [Último acceso: Mayo 2015].
- [9] Apache Solr, «Apache Solr,» [En línea]. Available: <http://lucene.apache.org/solr/>. [Último acceso: Mayo 2015].
- [10] Microsoft, «Microsoft SQL Server,» [En línea]. Available: <http://www.microsoft.com/es-es/server-cloud/products/sql-server/>. [Último acceso: Mayo 2015].
- [11] Oracle, «Oracle 11g,» [En línea]. Available: <http://www.oracle.com/lad/solutions/midsize/oracle-products/database/index.html>. [Último acceso: Mayo 2015].
- [12] SQLite, «SQLite,» [En línea]. Available: <https://www.sqlite.org/>. [Último acceso: Mayo 2015].
- [13] «Java Swing,» [En línea]. Available: <http://docs.oracle.com/javase/tutorial/uiswing/>. [Último acceso: Mayo 2015].
- [14] Microsoft, «WPF,» [En línea]. Available: <https://msdn.microsoft.com/es-es/library/ms754130%28v=vs.110%29.aspx>. [Último acceso: Mayo 2015].

- [15] «DB browser fro SQLite,» [En línea]. Available: <http://sqlitebrowser.org/>. [Último acceso: Mayo 2015].
- [16] Google Project Hosting, «Luke - Lucene index toolbox,» [En línea]. Available: <https://code.google.com/p/luke/>. [Último acceso: Mayo 2015].
- [17] Lucene Apache, «Query Lucene,» [En línea]. Available: https://lucene.apache.org/core/4_1_0/core/org/apache/lucene/search/Query.html. [Último acceso: Mayo 2015].
- [18] Lucene Apache, «MultiFieldQueryParser Lucene,» [En línea]. Available: https://lucene.apache.org/core/4_1_0/queryparser/org/apache/lucene/queryparser/classic/MultiFieldQueryParser.html. [Último acceso: Mayo 2015].
- [19] Apache Lucene, «BooleanQuery Lucene,» [En línea]. Available: http://lucene.apache.org/core/3_0_3/api/core/org/apache/lucene/search/BooleanQuery.html. [Último acceso: Mayo 2015].
- [20] Lucene Apache, «Numeric Range Query Lucene,» [En línea]. Available: https://lucene.apache.org/core/4_2_1/core/org/apache/lucene/search/NumericRangeQuery.html. [Último acceso: Mayo 2015].
- [21] Apache Lucene, «Wildcard Query,» [En línea]. Available: https://lucene.apache.org/core/4_1_0/core/org/apache/lucene/search/WildcardQuery.html. [Último acceso: Mayo 2015].

A. ANEXO: TIPOS DE BÚSQUEDAS IMPLEMENTADAS

Las búsquedas implementadas están divididas en dos grupos diferenciados: búsquedas en el índice de Lucene y búsquedas sobre la base de datos, donde en cada uno de estos dos grupos a su vez se encuentran categorizadas en rápidas, avanzadas y predefinidas.

El objetivo de este anexo es detallar las búsquedas implementadas sobre el índice de Lucene ya que son las que han supuesto un reto para este proyecto y las que ofrecen una funcionalidad distinta respecto a los proyectos estudiados disponibles hoy en día.

Para el **buscador rápido** se han aplicado tres métodos de búsqueda:

- Búsqueda de cada palabra de la consulta: Se ha tenido en cuenta cada palabra “relevante” por separado (ver apartado de *StopWords*). De esta manera el buscador devolverá todos aquellos documentos con al menos una palabra de la búsqueda en los campos seleccionados. Se ha utilizado la clase *Query* [16] para la implementación de este punto.
- Búsqueda del texto de la consulta: Para que durante la búsqueda las recetas que cumplan tener el filtro completo posean más valoración que aquellas que solo tienen un término de todo el filtro, se ha incluido como uno de los métodos de búsqueda. Se ha utilizado la clase *PhraseQuery* de Lucene para la implementación de este punto.
- Búsqueda por todos los campos del documento: Para realizar la búsqueda en todos los campos del documento que hace referencia a la receta, se ha utilizado la clase *MultiFieldQueryParser* [17].

Para el **buscador avanzado** se han aplicado varios métodos sobre una consulta ‘padre’ basada en la clase *BooleanQuery* [18], ya que permite aplicarse como contenedor del resto de tipos que a continuación se exponen:

- Para el filtro de texto de la página se han utilizado las mismas técnicas descritas en el buscador rápido.

- Para el filtro de ingredientes incluidos y excluidos del plato se ha utilizado una lista (*ArrayList*) de *Queries* que permiten incluir la configuración de los ingredientes en la consulta booleana padre.
- Para el filtro de tiempo se han creado tres *arrays* de forma que en función del rango que se escoja, se aplica el *array* concreto sobre la búsqueda booleana padre.
- Para el filtro de valoración se ha utilizado una búsqueda por rangos haciendo uso de la clase *NumericRangeQuery* [19].
- Para el filtro de categorías se ha configurado un objeto *PhraseQuery* para que aparezca obligatoriamente (*must*) en la consulta padre.

Para el **buscador predefinido** se han utilizado técnicas relacionadas con búsquedas de tipo *PhraseQuery* y *WildcardQuery* [20].

B. ANEXO: CASOS DE ÉXITO

A continuación se muestran un conjunto de casos de éxito que son contemplados por la aplicación desarrollada, pero que sin embargo para las páginas web gastronómicas estudiadas no devuelven los resultados esperados.

BÚSQUEDA POR INGREDIENTES

La búsqueda de recetas basada en ingredientes, permite al usuario aplicar filtros muy concretos. Por ejemplo, se ha realizado una búsqueda con los ingredientes típicos que lleva una paella:

Advanced searcher

Search by title:

Add ingredient that appears on the plate

Ingredient name	Total amount	
olive oil	4 tablespoons	+
onion	1	-
garlic	2 cloves	
chicken	2	
wine		
salt	1 pinch	
rice	1	
tomatoes	2	

Add ingredient that does not appear on the plate

Ingredient name	
-----------------	--

Time: Stars:

Difficulty: Category:

[Back](#) [Search](#)

Ilustración 22: Búsqueda avanzada por ingredientes.

Los cinco primeros resultados devueltos por la aplicación implementada han sido los siguientes:

We found these recipes for you:

Recipe id	Recipe name	Total time	Rating	Explore
6762	Carol's Arroz Con Pollo	45 mins	4.2982...	visualize
7227	Paella I	1 hr 30 mi...	4.4388...	visualize
9943	Jambalaya with Fresh Fruit	1 hr 10 mi...	4.5	visualize
10158	Everything But the Kitchen Sink ...	1 hr 40 mi...	2.6666...	visualize
10709	Anise Wine Chicken	5 hrs	4.5	visualize

Ilustración 23: Resultados de la búsqueda por ingredientes.

Como se puede observar en el listado superior, el buscador ha devuelto en segundo lugar la receta que se andaba buscando:

Desarrollo de un sistema de recogida y categorización de datos de recetas

Paella I (rating:4.438849)

Description:
A very traditional paella, garnished with chorizo, chicken, peas, squid, mussels, and shrimp. Chorizo is a sausage spiced with garlic and chili powder, remove casing before cooking. A paella pan is recommended.

Directions:
1. Heat olive oil in paella pan over medium heat. Add in onion, garlic and pepper, cook and stir for a few minutes. Add chorizo sausage, diced chicken, and rice; cook for 2 to 3 minutes. Stir in 3 1/2 cups stock, wine, thyme leaves, and saffron. Season with salt and pepper. Bring to the boil, and simmer for 15 minutes; stir occasionally.
2. Taste the rice, and check to see if it is cooked. If the rice is uncooked, stir in 1/2 cup more stock. Continue cooking, stirring occasionally. Stir in additional stock if necessary, up to 2 cups additional stock, 5 cups total. Cook until rice is done.

chicken

Ingredients:

Name	Amount
olive oil	4 tablespoons
onion, chopped	1
garlic, minced	2 cloves
red bell pepper, chopped	1
chorizo sausage, cut into pieces	4 ounces
skinless, boneless chicken breast halves - cut i...	2
uncooked Arborio rice	1 (12 ounce) package
chicken broth	5 cups
white wine	1/2 cup
fresh thyme	1 sprig
saffron	1 pinch
salt to taste	1 pinch
ground black pepper to taste	1 pinch
squid, cleaned and cut into 1 inch pieces	2
tomatoes, seeded and chopped	2
frozen green peas	1/2 cup
shrimp, peeled and deveined	12 large

Nutrition:

Name	Amount	Percentage
Calories	503 kcal	25%
Carbohydrates	55.5 g	18%
Cholesterol	98 mg	33%
Fat	17.6 g	27%
Fiber	3.3 g	13%
Protein	26.4 g	53%
Sodium	340 mg	14%

[Back](#)



Ilustración 24: Vista avanzada de la receta.


Sin embargo, aplicando el mismo filtro sobre la página *Allrecipes.com*, los resultados devueltos no han sido los esperados.



olive oil onion garlic chicken wine salt rice tomatoes


"olive oil onion garlic chicken wine salt rice tomatoes"
1,028 recipes

Relevance Rating Popularity

 **Olive Oil Pressure-Cooked Whole Roasted Chicken**
Rating ★★★★★
Reviews 7
"Pressure-cooked whole chicken using plenty of olive oil is placed under the broiler for a few minutes to produce a crispy skin and a very flavorful chicken." — by emilyv 

 **Olive Chicken II**
Rating ★★★★★
Reviews 122
"Chicken breasts simmered with wine, chicken broth, olives, tomatoes and a smattering of herbs and spices. Serve with saffron rice, if desired." — by LEG52

 **Slow Cooker Year-Round**
Our new Slow Cooker newsletter give you top-rated recipes and tips you can use right now. [Sign Up Now!](#) 

 **Summer Italian Marinade**
Rating ★★★★★
Reviews 5
"This refreshing Italian-style marinade works great for grilled chicken!" — by Mat & Kerry


 **Chickpeas in Tomato Sauce With Feta and Wine**
Rating ★★★★★
Reviews 81
"Chickpeas (garbanzo beans) and tomatoes are simmered in a delicious white wine sauce with feta cheese." — by JEN

Ilustración 25: Resultados de *Allrecipes*.

BÚSQUEDA POR FILTROS

La aplicación permite aplicar filtros de tiempo, categoría, dificultad y valoración de usuarios.

Advanced searcher

Search by title:
pizza

Add ingredient that appears on the plate

Ingredient name	Total amount	
mozzarella		+
tomatoes		-

Add ingredient that does not appear on the plate

Ingredient name	
onion	+

Time: between 30-60 mins | Stars: | Difficulty: | Category: vegetarian

Back Search

Ilustración 26: Búsqueda filtrada.

Los resultados del filtro aplicado han sido:


We found these recipes for you:

Recipe id	Recipe name	Total time	Rating	Explore
51723	Summer Olive Pizza	30 mins	4	visualize
51938	Pizza-Style Portabello Mushrooms	45 mins	5	visualize


Ilustración 27: Resultados de la búsqueda filtrada.

Desarrollo de un sistema de recogida y categorización de datos de recetas


El mismo filtro aplicado sobre la página *SimplyRecipes* ha devuelto los siguientes resultados:




About 836 results




How to Slice an Onion
Step by step instructions on how to safely slice onions both lengthwise, and crosswise.



How to Chop an Onion
How to chop an onion, safely, easily, and with minimum tears! Video plus step-by-step photos.



How to Caramelize Onions
How to slowly caramelize onions to bring out deep, rich, sweet flavor as the natural sugars in the onions caramelize. Video included.



How to Grill Pizza
Step-by-step instructions for using your grill, gas or charcoal, to make homemade pizza.

Ilustración 28: Resultados de la búsqueda filtrada por simplyrecipes.

BÚSQUEDA GENERAL

La aplicación ofrece también la funcionalidad estándar de búsqueda rápida por literal que poseen todos los buscadores estudiados.

Permite aplicar el filtro sobre un *input* y filtrar de forma adicional por la categoría que se desea buscar. En el caso de no seleccionar ninguno, la aplicación automáticamente busca sobre todas las categorías existentes.

Basic searcher

Search by title:

Category:

main dish

[Back](#) [Search](#)

Ilustración 29: 'Búsqueda rápida.

Y los resultados han sido:

We found these recipes for you:

Recipe id	Recipe name	Total time	Rating	Explore
66	Hot Pizza Dip	15 mins	4.631732	visualize
222	Jet Swirl Pizza A...	35 mins	4.700422	visualize
259	Mexican Pizza I	50 mins	4.450495	visualize
278	Grilled Pizza Wr...	30 mins	4.60733	visualize
456	White Pizza Dip	35 mins	4.286956	visualize
493	Mozzarella Puffs	20 mins	4.075472	visualize
541	Sweet Chili Thai...	20 mins	4.520408	visualize
566	Pizza Dip	35 mins	4.606383	visualize
625	Cheesy Pizza Dip	15 mins	4.188235	visualize
722	Pepperoni Dip I	30 mins	4.6	visualize
858	Pizza Rolls	40 mins	4.428571	visualize
925	Vegetable Pizza ...	45 mins	4.686275	visualize
964	Cheesy Italian P...	30 mins	4.291667	visualize
969	Greek Salad Dip		4.541667	visualize
971	Pizza Stix	15 mins	4.375	visualize
982	Vegetable Pizza II	1 hr	4.702127	visualize
1041	Baked Cheese ...	43 mins	3.604651	visualize
1081	Tomato Bacon ...	40 mins	4.5	visualize
1110	Dianne's Crab	35 mins	4.125	visualize
1191	Mexican Pizza II	25 mins	4.457143	visualize
1371	Tomato Basil S...	35 mins	4.642857	visualize
1412	Italian Nachos ...	25 mins	4.62963	visualize
1417	Apple Cheese P...	40 mins	3.481482	visualize
1500	Pizza Moons	30 mins	4.291667	visualize
1508	Roasted Potato ...	45 mins	4.375	visualize
1679	Pizza Balls	50 mins	3.65	visualize
1843	Toaster Oven Pi...	15 mins	4.294117	visualize
1849	Pineapple Jalap...	30 mins	4.352941	visualize
1984	Baked Potato Pi...	1 hr 50 mins	4.928571	visualize
2046	Veggie Pizza	25 mins	4.357143	visualize
2057	No Bake Pizza A...		4.461538	visualize
2119	BBQ Chicken To...	15 mins	4.692307	visualize
2132	Homemade Spi...	45 mins	4.666667	visualize
2180	Microwave Crac...	7 mins	4.166667	visualize
2300	RITZ White Pizz...	45 mins	4.2	visualize
2313	Grilled Zucchini ...	40 mins	4.6	visualize
2316	Fiesta Corn Tort...	30 mins	4.9	visualize
2412	Onion Strips	30 mins	4.444445	visualize
2416	Chunky Pizza Din		3.888889	visualize

[Go Back](#)

Ilustración 30: Resultados de búsqueda rápida.

C. ANEXO: RESULTADOS GRÁFICOS DE LA ENCUESTA

A continuación, se muestran los gráficos del estudio de la encuesta realizada en la fase de pruebas de aceptación para la aplicación Buscador.

- I. ¿Con qué frecuencia accede a contenidos informativos en internet de carácter alimenticio? {"A diario", "cada tres días", "cada semana", "cada mes", "nunca"}

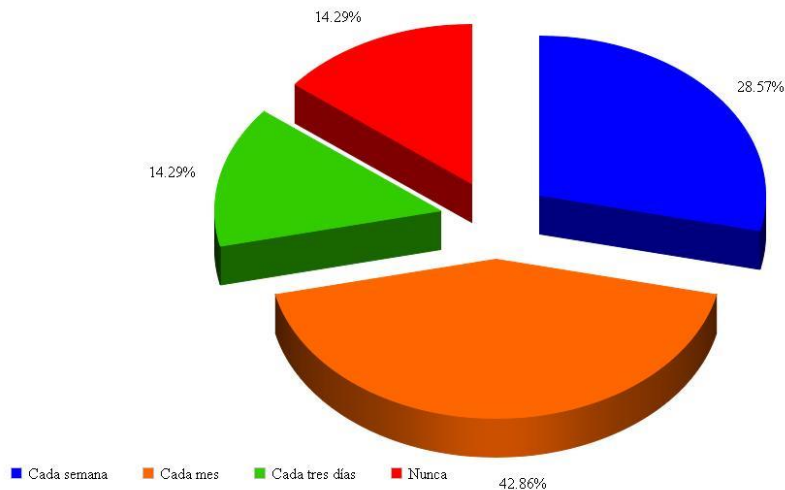


Ilustración 31: Resultados para la pregunta I.

II. ¿Los resultados devueltos satisfacen la receta que andaba buscando? {"Sí", "No"}

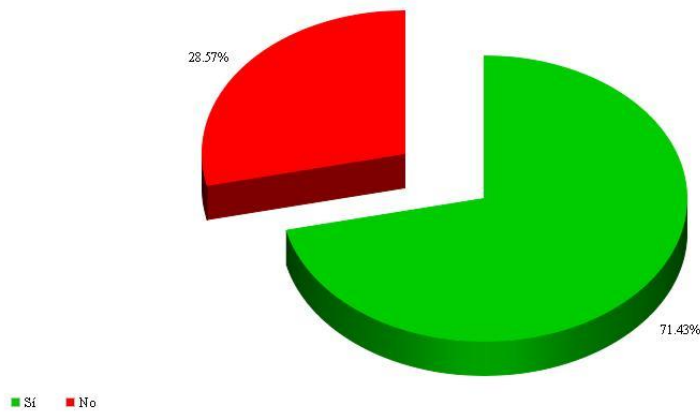


Ilustración 32: Resultados para la pregunta II.

III. ¿Le parece interesante la posibilidad de realizar filtros más avanzados? {"Sí", "No"}

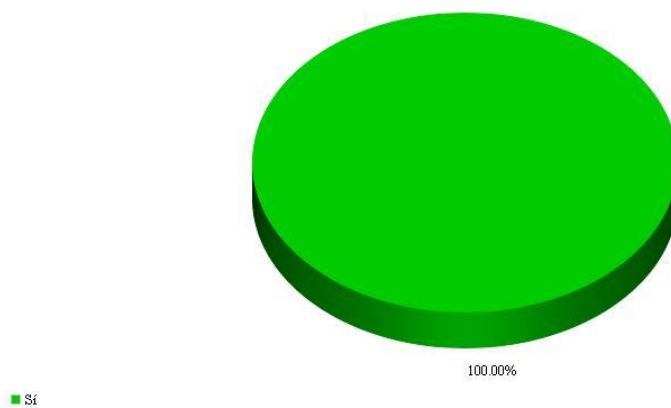


Ilustración 33: Resultados para la pregunta III.

IV. ¿Echa de menos algún filtro de búsqueda que no aparezca? {"Si", "No", "No lo sé"}

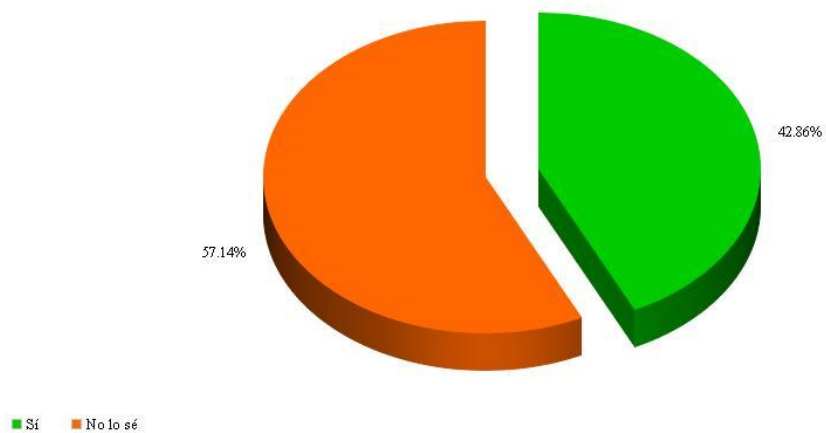


Ilustración 34: Resultados para la pregunta IV.

V. ¿Cree que mejora esta aplicación de búsqueda al medio que usted utiliza para encontrar recetas? {"Si", "No"}

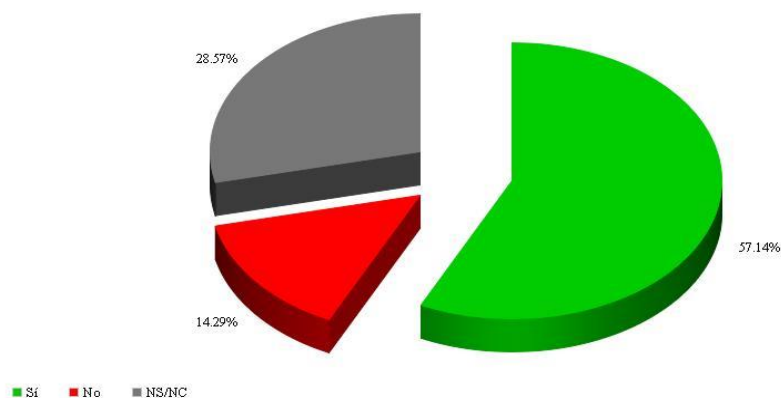


Ilustración 35: Resultados para la pregunta V.