
Universidad de las Ciencias Informáticas

FACULTAD 2

# HERRAMIENTA DE CONSUMO Y REUTILIZACIÓN DE CONTENIDO WEB

**Autor:**

Luis Manuel Gonçalves Torres

**Tutora:**

Ing. Yenlys Guerra Dávila

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

La Habana

18 de junio de 2015

# Declaración de autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año ____.

| | |
|---|---|
| Luis Manuel Gonçalves Torres | Ing. Yenlys Guerra Dávila |
| **(Autor)** | **(Tutora)** |

# Agradecimientos

*A mis padres, por apoyarme en todo momento, por todo su amor, cariño, comprensión, y por darme mucho más de lo que pueden, los amo.*

*A mi hermanito del alma, que siempre ha creído en mí.*

*A mis tías queridas por todo su apoyo a lo largo de mi vida, las quiero mucho.*

*A toda mi familia en general, que significa mucho para mí a pesar de la distancia.*

*A mi novia Yisel por todo su amor, paciencia y apoyo incondicional en todo este tiempo.*

*A mis amigos y compañeros de grupo, por su amistad, solidaridad y apoyo.*

*A mi tutora por acogerme, aconsejarme y conducirme por el camino correcto durante este trabajo.*

*A todos los que de una forma u otra han hecho posible este logro.*

# Resumen

El avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha permitido ampliar las posibilidades de progreso de la sociedad y la adquisición de conocimientos de una forma rápida y sencilla a través de la web, la cual ha experimentado un crecimiento exponencial en los últimos años. Este alto volumen de información dificulta el consumo de la misma a los usuarios y no permite seguir constantemente las publicaciones realizadas por las diversas fuentes de interés, lo cual provoca que los usuarios deban invertir mucho tiempo y esfuerzo para encontrar información relevante y actualizada. En este nuevo escenario el uso de Interfaces de Programación de Aplicaciones (APIs) permitirá mejorar los procesos de consumo y reutilización de contenido web.

En el presente trabajo se desarrolla una herramienta que permite informatizar el proceso de consumo y reutilización de contenido web a través de la técnica de web scraping. En la propuesta se utiliza como metodología de desarrollo de software XP, como tecnologías del lado del servidor el lenguaje de programación Python en conjunto con el framework Django y el plugin Django Rest Framework, además del gestor de tareas Celery, para la creación de una API REST accesible desde cualquier tipo de entorno. Por otro lado, como tecnologías del lado del cliente se utilizó HTML, CSS, JavaScript, jQuery y el framework AngularJS para crear una aplicación cliente que interactúa con la API en el servidor y ofrece sus funcionalidades de una forma amigable a los usuarios.

**Palabras claves:** consumo, contenido web, reutilización, web scraping

# Índice de Contenidos

# Índice de Figuras

# Índice de Tablas

# Introducción

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha permitido ampliar las posibilidades de progreso de la sociedad y la adquisición de conocimientos de una forma rápida y sencilla. Como parte de esas tecnologías se encuentra Internet, siendo la World Wide Web (WWW o la web) el servicio que más éxito ha tenido. Gracias a la web, millones de personas tienen acceso fácil e inmediato a una cantidad extensa y diversa de información en línea, que además crece cada año a un ritmo acelerado [18].

Cuba, a pesar de no contar con una infraestructura de primer nivel a lo largo y ancho del país, posibilita el acceso a Internet a diferentes organizaciones y empresas, dentro de las cuales se encuentran las universidades del país. Una de estas instituciones es la Universidad de las Ciencias Informáticas (UCI), un centro docente-productivo que desarrolla aplicaciones y servicios informáticos orientados a diversos sectores de la economía y los servicios, dentro y fuera de Cuba. En cada uno de los centros productivos de la UCI el consumo de información en Internet es elevado y tiene un impacto directo en la calidad de los productos, ya que es el medio por excelencia para consultar documentación, realizar trabajos investigativos, resolver dudas en foros de discusión, seguir las últimas noticias publicadas, entre otras muchas tareas.

Sin embargo, existen varios factores que dificultan el consumo de información a los usuarios, debido fundamentalmente al crecimiento exponencial de la web en los últimos años. Entre estos factores se encuentran el volumen elevado de fuentes de información a consultar, la diseminación de estas fuentes en la web, así como la dispersión de la información de interés dentro de las fuentes. Estos factores provocan que los usuarios deban invertir mucho tiempo y esfuerzo para encontrar información relevante y actualizada, incidiendo negativamente en el proceso de desarrollo de software. Además, resulta muy difícil para los usuarios seguir constantemente las publicaciones realizadas por las diversas fuentes de interés, sobre todo si la frecuencia de las mismas es alta.

Otro factor que no permite un aprovechamiento adecuado de la información en la web, es la ausencia de Interfaces de Programación de Aplicaciones (APIs) para acceder a los datos en bruto publicados. Esto impide la integración y reutilización de la información en aplicaciones

web híbridas, así como la manipulación automática de los datos en la lógica de negocio de otras aplicaciones. De esta forma, se desaprovecha una gran variedad de información (muchas veces de calidad, completa y fiable), que es potencialmente reutilizable, como por ejemplo, información social, económica, geográfica, estadística, meteorológica, turística, empresarial y educacional.

De la problemática descrita anteriormente se deriva el siguiente **problema a resolver**: ¿Cómo informatizar el proceso de consumo y reutilización de contenido web en la UCI?

El problema a resolver planteado, se enmarca en el **objeto de estudio**: El proceso de consumo y reutilización de contenido web.

Teniendo como **objetivo general**: Desarrollar una aplicación web que permita informatizar el proceso de consumo y reutilización de contenido web en la UCI.

Definiendo como **campo de acción**: El proceso de consumo y reutilización de contenido web de forma informatizada en la UCI.

Para dar cumplimiento al objetivo establecido se definen las siguientes **tareas de investigación**:

1. Análisis de las herramientas y tecnologías existentes que permiten el consumo y reutilización de contenido web, estableciendo similitudes con la investigación en curso.
2. Análisis de los procesos que intervienen en el consumo y reutilización de contenido web, para tener un mejor dominio del problema a resolver.
3. Estudio de los elementos correspondientes a la planificación del software con el objetivo de llevar a cabo un desarrollo organizado.
4. Análisis de los elementos correspondientes al diseño del software para guiar la implementación del mismo.
5. Estudio de las herramientas y tecnologías seleccionadas para la implementación del sistema para lograr una aplicación funcional como solución al problema de la investigación.
6. Estudio de los diferentes tipos de pruebas para verificar el correcto funcionamiento del sistema desarrollado.

Definiéndose como **idea a defender** que: El desarrollo de una aplicación web que permita informatizar el proceso de consumo y reutilización de contenido web permitirá un mejor aprovechamiento del mismo en la Universidad de las Ciencias Informáticas.

La realización del presente trabajo se sustenta en los siguientes métodos de investigación:

**Métodos teóricos:**

- Analítico-Sintético: Permite realizar el estudio teórico de la investigación, facilitando el análisis de documentos y la extracción de los elementos más importantes acerca del funcionamiento de las aplicaciones que permiten el consumo y reutilización de contenido web en la actualidad.
- Histórico-Lógico: Este método permite estudiar todo lo relacionado con los sistemas de consumo y reutilización de contenido web, para así obtener un conocimiento histórico de su desarrollo y comportamiento tanto a nivel internacional como nacional.

**Métodos empíricos:**

- Medición: El sistema obtenido como resultado de la investigación será medido mediante las pruebas unitarias y de aceptación garantizando el correcto funcionamiento del mismo.

En aras de estructurar el proceso de desarrollo del presente trabajo de una manera coherente y organizada, se dividió el mismo en un total de 4 capítulos:

Capítulo 1. Fundamentación Teórica: Se abordan los elementos teóricos más importantes que sirven de base para la realización de toda la investigación; también se proporciona una descripción de las herramientas y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

Capítulo 2. Propuesta del Sistema: Se brinda una explicación de la solución propuesta, enfatizando las características del sistema a desarrollar, haciendo un análisis de los flujos de trabajo involucrados en el campo de acción y definiendo el objeto de automatización de la propuesta.

Capítulo 3. Exploración, Planificación y Diseño: Se detallan los artefactos generados durante las fases de exploración, planificación y diseño del proyecto.

Capítulo 4. Implementación y Pruebas: Se expone todo lo relacionado a los procesos de implementación y pruebas del sistema.

CAPÍTULO 1

# Fundamentos teóricos necesarios para la implementación del sistema de consumo y reutilización de contenido web

## 1.1. Introducción

En el presente capítulo se realiza un análisis de los sistemas fundamentales que permiten realizar el consumo y reutilización de contenido web, identificando características interesantes y limitaciones de los mismos. Además, se describen las tecnologías seleccionadas para el desarrollo del sistema, tanto del lado del cliente como del lado del servidor, así como las herramientas de desarrollo utilizadas.

## 1.2. Consumo y reutilización de contenido web. Estado del arte

Cada día en Internet se publican millones de noticias, se suben millones de imágenes, videos, cursos, opiniones, etc. Saber encontrar aquella información relevante (fiable y de calidad) en ese caos informativo es una tarea cada vez más compleja e importante [29].

Con el objetivo de facilitar el consumo de información actualizada a los usuarios han surgido diferentes tecnologías y herramientas. Algunas de las opciones más usadas son los canales Really Simple Syndication (RSS) y los boletines electrónicos, así como las técnicas de web scraping. Todos estos elementos serán analizados a continuación.

### 1.2.1. Canales RSS

RSS es un formato de archivo, basado en XML, que sirve para recoger contenidos publicados en páginas web. Constituye un estándar creado para distribuir contenidos, usualmente las novedades de los sitios web, por un canal distinto de la propia página web. Gracias a RSS el usuario visitante de una página web puede suscribirse a sus novedades y recibirlas en el instante de ser publicadas, sin necesidad de acceder a la página web donde se han publicado [4].

Las ventajas que ofrecen los sistemas RSS son muchas, de las cuales se destacan las siguientes [4, 33]:

- Supone un importante ahorro en el tiempo de navegación y búsqueda de información, ya que en el lector RSS, el usuario tendrá un resumen de los últimos artículos publicados.
- El RSS está libre de contenido no deseado (spam).
- La suscripción o cancelación a las fuentes RSS es rápida y sencilla, ya que es el propio usuario quien realiza estas acciones.
- Es ampliamente usado en internet, por ejemplo en blogs, portales, diarios online, sitios corporativos, sitios especializados, catálogos y novedades de sitios de comercio electrónico.
- Se puede integrar el contenido de los RSS dentro de otros sitios web, lo cual permite reutilizar la información publicada de manera sencilla.

Esta tecnología también posee varias desventajas o limitantes [33]:

- Recibir los canales RSS crea mayor tráfico y demanda en el servidor, por lo que puede hacer que la conexión a Internet se vuelva algo más lenta.
- No es posible aplicar filtros avanzados a la información que se desea recibir.
- No se pueden agrupar actualizaciones desde sitios web diferentes en un único canal RSS.
- El nivel de reutilización que brinda es muy básico, ya que la información se ofrece en un archivo de texto.
- Es posible que algunas páginas web de interés no publiquen sus noticias a través de un canal o fuente RSS.

Aunque la tecnología RSS está siendo ampliamente utilizada por los sitios web en internet, la misma no ha tenido una gran aceptación por parte de los usuarios. Según un informe de la empresa Yahoo, donde se muestran algunas estadísticas de uso, se estima que solo un 12 % de los usuarios de Internet conoce lo que es RSS, mientras que solo el 4 % de los usuarios lo usan [21].

Se puede concluir que los canales RSS poseen varias limitaciones que no permiten un consumo y reutilización adecuados de la información publicada en la web, aunque constituyen una de las mejores opciones actualmente.

### 1.2.2. Boletines electrónicos

Un boletín electrónico es un correo electrónico enviado de forma periódica en texto plano o en código HTML, y que contiene información, promoción comercial o publicidad de la página web que la emite. El boletín se recibe en el buzón de correo del destinatario, quien previamente se tiene que haber subscrito voluntariamente a dicho servicio en la página web que lo emite o presta el servicio [24].

Los boletines electrónicos han alcanzado un alto grado de popularidad, debido fundamentalmente al amplio uso del correo electrónico a nivel mundial. En 2012 se estima que existían 2.200 millones de usuarios de correo electrónico en el mundo, generando un tráfico total diario de 144.000 millones de mensajes. Además se estima que el 68.8 % de todo el tráfico generado eran correos no deseados (spam). Estas estadísticas denotan un uso masivo del correo electrónico, así como una amplia práctica del envío de correos no deseados, lo cual es un factor en contra de las subscripciones a boletines electrónicos [28].

De forma general, los boletines electrónicos presentan algunas desventajas y limitaciones, las cuales se señalan a continuación:

- Al tener que proporcionar una dirección de correo electrónico para suscribirse se corre el riesgo de recibir correos no deseados.
- No es posible filtrar ni agrupar la información que se desea recibir.
- No se puede reutilizar adecuadamente la información recibida a través de boletines electrónicos.
- No todos los sitios web implementan el envío de boletines electrónicos para difundir sus contenidos.

Luego de las características expuestas anteriormente, se puede concluir que los sistemas de boletines electrónicos no representan la herramienta más idónea para consumir contenido web

desde cualquier fuente de interés, además de que no permiten una adecuada reutilización de la información recibida.

### 1.2.3. Web scraping

Se denomina web scraping (raspado web) al conjunto de técnicas usadas para extraer automáticamente alguna información de una página web en lugar de copiarla manualmente. El web scraping es usado regularmente para buscar cierto tipo de información, extraerla, e integrarla en una nueva página web [37].

Los web scrapers (aplicaciones basadas en web scraping) son especialmente útiles cuando se tienen necesidades específicas de búsqueda de información, de ejecución de acciones y toma de decisiones usando la información encontrada, o cuando un sitio web despliega información relevante, pero no proporciona una API para que otras aplicaciones realicen consultas directas sobre dicha información. Estos sistemas ofrecen automatización y cierto nivel de inteligencia para la obtención y procesamiento de los datos. Podría decirse que permiten acceder a los datos camuflados en la capa de presentación, constituyendo una forma de adaptar el contenido web publicado a necesidades personales y capitalizar información que está aparentemente oculta [30].

Por lo expuesto anteriormente, la técnica de web scraping debe constituir una parte esencial de la solución del problema de la investigación en curso. Sin embargo, por sí sola no constituye una solución, ya que presenta algunas particularidades que impiden su uso directo:

- Se debe programar manualmente el acceso a cada una de las páginas de donde se desea obtener la información, así como la lógica de procesamiento y/o reutilización de la misma.
- El proceso de parsear el HTML para obtener la información de interés puede ser una tarea complicada si la estructura es deficiente.
- Se debe tratar de no afectar el desempeño del sitio que se analiza, evitando ataques de denegación de servicio no deseados.

Por estas razones, se analizaron otras aplicaciones que utilizan web scraping para obtener la información publicada, que permitieran emplear las potencialidades del web scraping sin necesidad de programar manualmente todo el proceso.

### 1.2.4. Casos de estudio

Actualmente existen una gran cantidad de aplicaciones para la extracción de contenido web, las cuales varían en cuanto a costo y funcionalidades [36].

Por esta razón se hace necesario llevar a cabo un estudio sobre las aplicaciones de consumo y reutilización de contenido web, con el propósito de evaluar las fortalezas y debilidades de cada una en el contexto de la presente investigación.

Las analizadas a continuación son las más completas y disponibles de manera gratuita [35], según una recopilación realizada por el sitio scraping.pro, que desde el año 2012 analiza, prueba y divulga información asociada a este tipo de aplicaciones.

#### 1.2.4.1. Import.io

Import.io es un scraper online gratuito que permite recolectar diferentes tipos de datos y organizar la información obtenida en tablas. Posee una interfaz moderna y facilidades de uso tanto para usuarios novatos como para expertos, con un sistema de ayuda que guía todo el proceso de scraping.

Cuenta una amplia gama de funcionalidades útiles para el consumo y reutilización de la información, siendo una de las herramientas gratuitas más completas y novedosas, ya que fue fundada en 2012 y lanzada al mercado en 2013 [26].

El objetivo principal de esta aplicación es facilitar el proceso de extracción, almacenamiento y reutilización de datos, ofreciendo un sinfín de características impresionantes en este sentido. Sin embargo, no brinda opciones avanzadas para el consumo de información, teniendo como principales limitantes:

- No posibilita la generación de canales RSS y boletines electrónicos personalizados a partir del contenido recolectado.
- Carece de opciones para aplicar filtros avanzados al contenido.

Además de las limitantes en cuanto a funcionalidades, es necesario señalar que se trata de una aplicación aún en fase de desarrollo, que puede presentar errores o comportamientos inesperados ocacionalmente [2].

#### 1.2.4.2. Kimono

Kimono es un servicio de web scraping en línea que permite crear APIs a partir de sitios web. Fue lanzado en enero de 2014, y aún se encuentra en desarrollo. Es un proyecto muy prometedor, que permite recuperar la información recolectada en varios tipos de formatos (JSON, CSV y RSS), así como recuperar versiones anteriores de los datos, enviar alertas sencillas por correo electrónico, y una gran cantidad de opciones para configurar y gestionar las APIs creadas [36].

Es una herramienta bastante completa, presentando algunas limitaciones, que son señaladas a continuación:

- No posibilita la generación de boletines electrónicos personalizados a partir del contenido recolectado.
- Carece de opciones para aplicar filtros avanzados al contenido.
- Dependencia de conectividad a Internet para obtener los datos.
- Su versión profesional tiene un costo de hasta $168.

Además, al igual que Import.io, es una herramienta muy nueva, con muchas promesas, pero aún inestable, sobre todo la versión gratuita. Sin embargo, esta versión es bastante funcional y puede llegar a ser mucho más rápida que su homóloga Import.io [36].

Luego del proceso de análisis de las aplicaciones anteriores, se concluye que las mismas no presentan todas las funcionalidades demandadas en la presente investigación, además de no contar con un nivel de madurez suficiente, ya que ambas se encuentran en fase de desarrollo. Sin embargo, el presente estudio ha permitido constatar la ebullición que presenta el mercado de este tipo de aplicaciones en la actualidad, debido fundamentalmente a las crecientes necesidades de las organizaciones y empresas de mejorar los procesos de consumo y reutilización de contenido web. Además, se seleccionaron algunas funcionalidades de las aplicaciones analizadas que no deben faltar en la solución propuesta, como son el desarrollo de una API independiente de la aplicación, la selección del contenido con un click, el almacenamiento del contenido a través de versiones, entre otras.

## 1.3. Metodología de desarrollo de software

Actualmente existen diferentes metodologías de desarrollo de software, debido a que todos los proyectos de software no son iguales y por tanto necesitan un enfoque de desarrollo acorde a sus características particulares. Los proyectos difieren en cuanto al tamaño, composición y prioridades. Las personas de un proyecto poseen además diferentes niveles de experiencia, principios y habilidades. Todos estos aspectos deben ser considerados para seleccionar correctamente una metodología de desarrollo de software [8].

Para la realización del sistema propuesto en este trabajo se decidió utilizar la metodología de desarrollo Extreme Programming (Programación Extrema, XP por sus siglas en inglés). XP es una metodología basada en valores de simplicidad, comunicación, retroalimentación y coraje. Funciona mediante la unión de todo el equipo y la aplicación de prácticas bien definidas, con suficiente retroalimentación para permitirle al equipo ver donde están y adaptar sus prácticas a cada situación única [6].

### 1.3.1. Fundamentación de la metodología de desarrollo seleccionada

Se decidió utilizar XP debido a que se adapta en gran medida al tipo de proyecto a desarrollar, las condiciones de trabajo, y las prácticas utilizadas. A continuación se describen varias de las razones que motivaron la selección de esta metodología.

- El proyecto es pequeño. XP está concebida para ser utilizada dentro de proyectos pequeños.
- Los requisitos del sistema pueden cambiar. El sistema debe cambiar y ampliar sus funcionalidades de forma que sea capaz de adaptarse a cada nueva situación. Uno de los principios básicos de XP es que el cambio frecuente de los requerimientos es algo normal en el proceso de desarrollo. Esta metodología se adapta perfectamente a los proyectos cuyos requerimientos cambian a menudo.
- El cliente forma parte del equipo de desarrollo. Mediante la aplicación de XP se puede lograr una retroalimentación mayor y lograr un producto que satisfaga sus necesidades.

- El riesgo de desarrollo es elevado debido al corto tiempo de entrega planteado y a los continuos cambios de requerimientos. XP está diseñada para mitigar los riesgos en proyectos con estas características.
- Algunas prácticas de XP como las entregas pequeñas, el diseño simple y el Desarrollo Guiado por Pruebas (TDD, del inglés Test Driven Development) son parte de la filosofía de desarrollo del equipo.

## 1.4. Tecnologías y herramientas de desarrollo

Existen numerosas tecnologías y herramientas relacionadas con la programación de aplicaciones web, tanto del lado del servidor como del cliente. La gran cantidad de recursos disponibles, y esta característica dual de las aplicaciones web (que poseen un conjunto de funcionalidades independientes del lado cliente y del lado servidor), dificulta enormemente el análisis y diseño de este tipo de aplicaciones [19].

A continuación se describen brevemente las tecnologías y herramientas seleccionadas (tanto del lado del servidor como del cliente) para el desarrollo del presente trabajo.

### 1.4.1. Tecnologías del lado del servidor

#### 1.4.1.1. Python 2.7

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata además de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos [17].

Es además un lenguaje de programación de código abierto muy popular para el desarrollo de aplicaciones independientes y de scripting en una gran cantidad de dominios de problemas. Es gratis, portable, potente, y relativamente fácil de usar. Python es empleado tanto en proyectos grandes como pequeños, mejorando la productividad de los desarrolladores y la calidad del software [22].

Python ha sido seleccionado como lenguaje de programación por las características enunciadas anteriormente, así como por la gran cantidad de bibliotecas disponibles que pueden ser útiles en el presente trabajo.

#### 1.4.1.2. Django 1.7

Django es un framework para desarrollo web escrito en Python, que permite construir y mantener aplicaciones web de alta calidad con el mínimo esfuerzo posible. Django permite a los desarrolladores concentrarse en el código de la aplicación propiamente dicha, manejando tareas repetitivas y complejas propias de la programación web. De esta forma, provee un alto nivel de abstracción para patrones comunes en el desarrollo web, incrementando la calidad de las soluciones, aumentando la productividad y disminuyendo los errores en el código [20].

Django provee una arquitectura similar a la muy conocida Modelo-Vista-Controlador (MVC), con los mismos beneficios, pero con un marcado énfasis en la productividad, llamada Modelo-Plantilla-Vista (MTV). Consecuentemente, provee un conjunto de herramientas asociadas a la arquitectura, como un sistema de Mapeo Relacional de Objetos (ORM), un motor de plantillas, entre otras utilidades [1].

#### 1.4.1.3. Django Rest Framework 3.1

REST (REpresentational State Transfer), es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. REST permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP, por lo que es mucho más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC. Por lo tanto, REST es el tipo de arquitectura más natural y estándar para crear APIs para servicios orientados a Internet [25].

Debido a las características mencionadas anteriormente, se decide emplear una herramienta que facilite el desarrollo de una API REST que permita exponer los datos almacenados en el servidor. Se escoge Django Rest Framework como solución, debido a diversas razones que serán expuestas a continuación.

Django Rest Framework es una aplicación Django que permite construir proyectos software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar (Views, Resources,

Serializers, etc.), una API navegable y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP, como por ejemplo: POST y GET. Cabe resaltar además que los mentores de este proyecto hacen un uso intensivo de las vistas genéricas, con el objetivo de aprovechar las ventajas de la programación orientada a objetos [27].

#### 1.4.1.4. BeautifulSoup 4.3

BeautifulSoup es una biblioteca escrita en Python diseñada para aplicaciones de web scraping. Algunas de sus características más interesantes se exponen a continuación [31]:

- Provee métodos robustos para navegación, búsqueda y modificación del árbol DOM de cualquier documento HTML, así como una completa caja de herramientas para extraer exactamente la información que se desea.
- BeautifulSoup realiza automáticamente las tareas de codificación y decodificación necesarias en el proceso de web scraping.
- Se puede integrar fácilmente con los parsers HTML para Python más populares, como lxml y html5lib, permitiendo probar diferentes estrategias en cuanto a velocidad y flexibilidad.

Sus características hacen que sea una biblioteca de obligada inclusión en el presente trabajo, ya que facilita muchas de las tareas de desarrollo.

#### 1.4.1.5. Celery 3.1.17

Celery es una aplicación que nos permite crear tareas de trabajo asíncronas gestionadas por un gestor de colas que está basado en el envío de mensajes de manera distribuida. Se focaliza en operaciones en tiempo real pero también soporta la calendarización de tareas, es decir, que puede ejecutar tareas en un momento determinado o de manera periódica. Las unidades de ejecución, llamadas tareas, se ejecutan de manera concurrente en uno o más nodos de trabajo. Estas tareas pueden ejecutarse tanto asíncrona como síncronamente, siendo el sistema en general altamente configurable [3].

#### 1.4.1.6. PostgreSQL 9.2

PostgreSQL es uno de los Sistemas de Gestión de Bases de Datos (SGBD) más avanzados a nivel mundial. Sus características técnicas lo hacen uno de los más potentes y robustos del

mercado. Su desarrollo comenzo hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios [11].

Además, se integra muy bien con el lenguaje de programación seleccionado, y es una de las bases de datos mejor soportadas por el ORM de Django.

#### 1.4.1.7. Servidores web

**Gunicorn**

Gunicorn, también conocido como Green Unicorn (Unicornio Verde), es un servidor WSGI HTTP para Python y sistemas unix. Se trata de un fork portado del proyecto Ruby's Unicorn. Es ampliamente compatible con varios frameworks web, entre ellos Django. Gunicorn permite administrar las peticiones simultáneas que se generan en una aplicación web y definir de forma sencilla diversos parámetros de configuración .

Su función en el sistema es servir la aplicación escrita en Python, es decir, el contenido dinámico generado a partir de interpretar el código.

**Nginx**

Nginx es un servidor web y proxy inverso gratuito, de código abierto y de alto rendimiento, además de ser un proxy para protocolos de correo electrónico (IMAP/POP3). Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada. Es multiplataforma, por lo que corre en sistemas tipo Unix (GNU/Linux, BSD, Solaris, Mac OS X, etc.) y Windows. Este servidor es conocido por su estabilidad, su gran conjunto de características, una configuración sencilla y por consumir pocos recursos [34].

En el presente trabajo, Nginx se utilizará como proxy inverso, que servirá los archivos estáticos de la aplicación, de manera que se logre un buen balance de carga, y por tanto un buen rendimiento.

### 1.4.2. Tecnologías del lado del cliente

#### 1.4.2.1. HTML5

HTML, más que el lenguaje de marcado predominante usado para estructurar el contenido web, es realmente una tecnología web que interrelaciona varios estándares. En esta última revisión del lenguaje (HTML5) se incluyen varias características verdaderamente útiles, como son nuevos elementos semánticos, mejoras en los formularios, soporte nativo para video y audio, un lienzo (canvas) para el dibujo a través de JavaScript, soporte para almacenamiento local y geolocalización [23].

#### 1.4.2.2. CSS3

Cascading Style Sheets (CSS) no es más que un lenguaje de estilo que describe cómo el contenido HTML es presentado al usuario. CSS3 es la última versión de la especificación CSS, incluyendo nuevas características para ayudar a los desarrolladores a resolver numerosos problemas sin la necesidad de un marcado no semántico, complejos scripts o imágenes extra. Estas nuevas características incluyen soporte para selectores adicionales, sombras, esquinas redondeadas, múltiples fondos, animaciones, transparencias, entre otras [16].

#### 1.4.2.3. JavaScript

JavaScript es un potente lenguaje de scripting basado en objetos. El código JavaScript puede ser embebido directamente en un documento HTML, e interpretado por un navegador web en el lado del cliente (su uso más extendido). Cuando se combina con el Document Object Model (DOM) definido por un navegador web, JavaScript permite crear o manipular dinámicamente el contenido HTML [13].

#### 1.4.2.4. jQuery 1.11.1

jQuery es una de las bibliotecas JavaScript más populares y ampliamente usadas. Su objetivo fundamental es simplificar algunas tareas comunes, como buscar elementos en el DOM, manipular esos elementos, editar atributos HTML y propiedades CSS, definir manejadores de eventos, entre otras. Cuenta además con utilidades Ajax para realizar peticiones HTML dinámicamente, así como funciones de propósito general para trabajar con objetos y colecciones [14].

#### 1.4.2.5. AngularJS 1.3.15

AngularJS es un proyecto de código abierto, escrito en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones SPA (Single Page Applications), y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso con programación del lado del cliente. Angular además promueve y usa patrones de diseño de software, como el conocido MVC, aunque en una variante muy extendida en el mundo de Javascript. Básicamente estos patrones marcan la separación del código de los programas dependiendo de su responsabilidad. Eso permite repartir la lógica de la aplicación por capas, lo que resulta muy adecuado para aplicaciones de negocio y para las aplicaciones SPA, sobre todo si las mismas cuentan con una API REST en el servidor [5].

### 1.4.3. Herramientas de desarrollo

Durante la etapa de implementación del sistema propuesto en este trabajo se utilizó un gran número de herramientas para facilitar esta actividad.

Como sistema de control de versiones se utilizó Git, un sistema distribuido de gran popularidad actualmente, sin el cual es muy difícil llevar a cabo tareas como la refactorización del código o prácticas como TDD.

Se utilizó Sass como preprocesador CSS, en conjunto con los frameworks Compass y Susy, herramientas que permitieron escribir código CSS altamente funcional, bien organizado y fácil de mantener.

Como Entorno de Desarrollo Integrado (IDE) se utilizó PyCharm, ya que el mismo tiene un excelente soporte para Python y Django, además de facilitar los procesos de búsqueda de errores, ejecución de tareas a través de consola, inicio y detención de servidores de desarrollo, utilidades para el sistema de control de versiones, entre muchas otras.

## 1.5. Conclusiones del capítulo

Con el desarrollo del marco teórico se logró organizar y guiar el trabajo, profundizando en el estudio de algunos conceptos fundamentales para el desarrollo de una aplicación web

que permita informatizar el proceso de consumo y reutilización de contenido web en la UCI. Además se realizó un estudio de las aplicaciones similares existentes a nivel internacional, lo que posibilitó la selección de las principales características de las tecnologías, herramientas, lenguajes y técnicas de desarrollo propuestas para la solución del problema actual.

CAPÍTULO 2

# Propuesta de solución del sistema de consumo y reutilización de contenido web

## 2.1. Introducción

En el presente capítulo se realiza un análisis de las características del sistema a desarrollar, partiendo de la problemática que da origen al mismo. Además se realiza un análisis del proceso de consumo y reutilización de contenido web en la UCI, definiendo las tareas vinculadas a este proceso que necesitan ser automatizadas para lograr un aprovechamiento adecuado de la información publicada.

## 2.2. Análisis del proceso de consumo y reutilización de contenido web en la UCI

En la actualidad el proceso de consumo y reutilización de contenido web en la UCI se realiza de forma manual mayoritariamente. Esto significa que, a través de una navegador web, cada usuario ejecuta normalmente un procedimiento como el siguiente para mantenerse actualizado:

1. Acceder a un sitio web de interés.
2. Verificar si hay nuevas publicaciones desde su última visita.
3. Seleccionar la información de su interés.
4. Leer y/o resguardar la información para su uso posterior.

Este proceso se repite en correspondencia con el número de sitios web de interés de cada usuario, así como la cantidad de veces que cada usuario visita un mismo sitio en busca de actualizaciones. Esto conlleva a que dicho proceso sea largo y tedioso, afectando directamente la productividad de los desarrolladores y por tanto su rendimiento en los proyectos. Si además se tiene en cuenta que muchos usuarios tienen sitios de interés en común, se puede apreciar que se incurre en un gasto innecesario de ancho de banda y de cuota de navegación.

De las herramientas y tecnologías más usadas para realizar este proceso se encuentran los boletines electrónicos, y en menor medida los canales RSS. Estas tecnologías presentan algunas limitaciones que fueron analizadas en el capítulo anterior.

Otro factor que afecta el aprovechamiento eficiente de la información es la forma en que la misma es almacenada. Las herramientas más usadas para este fin son el guardado de páginas web que ofrecen los navegadores de forma nativa, así como el uso de plugins más avanzados, como el ScrapBook de Firefox. Esto provoca que la información esté desorganizada, se almacene información innecesaria y se dificulten los procesos de búsqueda y procesamiento de la misma.

## 2.3. Objeto de automatización

Todo lo expuesto anteriormente indica que existen varios procesos que deben ser automatizados, puesto que su ejecución de forma manual resulta tediosa, además de consumir una valiosa porción del tiempo de los desarrolladores. La correcta realización de estos procesos es de vital importancia para lograr un aprovechamiento adecuado de la información publicada en la web. A continuación se realiza una descripción de los procesos que necesitan ser automatizados con el sistema propuesto.

- **Acceso a páginas web.** Se debe garantizar el acceso automático a cualquier página web accesible desde un navegador, tanto de la Intranet UCI como de sitios nacionales e internacionales.
- **Selección de información.** Para definir solo la información de interés, con posibilidades de aplicar filtros y seleccionar datos en un formato adecuado.
- **Almacenamiento de información.** Se debe posibilitar el resguardo de la información seleccionada de forma organizada, facilitando posteriores procesos de búsqueda, recuperación y transformación de la misma.
- **Distribución de información.** Se debe permitir la distribución de la información por las vías más usadas, facilitando la creación de boletines electrónicos y canales RSS personalizados, así como el uso de dichas formas de distribución por parte de los usuarios interesados.
- **Reutilización de información.** Se debe garantizar el acceso a los datos en bruto (previamente seleccionados), a través una API accesible desde cualquier entorno.

## 2.4. Características de la propuesta del sistema

Se propone la implementación de una aplicación web que provea las funcionalidades necesarias para informatizar todo el proceso de consumo y reutilización de contenido web. La misma estará programada fundamentalmente en Python y Django, posibilitando el acceso a los usuarios a través de un sistema de autenticación basado en tokens.

El sistema debe permitir gestionar la lista de páginas de interés de cada usuario, además de las secciones específicas a seguir por cada página. Cada página tendrá una URL asociada para acceder a la misma, mientras que las secciones de interés podrán seleccionarse a través de selectores CSS, permitiendo además filtrar la información a través de palabras clave o expresiones regulares. A partir del contenido de interés seleccionado, se debe permitir la gestión de las vías de distribución de la información, como boletines electrónicos y canales RSS personalizados con los datos en un formato adecuado. La aplicación debe permitir además la socialización de todos los datos y las vías de distribución generadas por los usuarios, de forma tal que un usuario pueda subscribirse a un boletín creado por otro usuario, así como clonar los canales RSS y boletines públicos que necesite.

Cada solicitud de acceso a las páginas web debe ser tratada como una tarea independiente, estableciendo una cola de tareas, donde las mismas se ejecutan cada un intervalo de tiempo configurable. Esto permitirá que las tareas asociadas al proceso de consumo y reutilización de la información no interfieran en la experiencia de usuario de la aplicación web.

## 2.5. Conclusiones del capítulo

En el presente capítulo se realizó un análisis sobre el proceso de consumo y reutilización de contenido web en la UCI, el cual permitió definir los procesos que deben ser automatizados para dar solución al problema planteado en esta investigación. Además se analizó de forma detallada el funcionamiento del sistema propuesto como solución, haciendo énfasis en las características y funcionalidades que debe poseer el mismo para lograr el objetivo de la presente investigación.

CAPÍTULO 3

# Exploración, Planificación y Diseño del sistema de consumo y reutilización de contenido web

## 3.1. Introducción

En el presente capítulo se presentan los resultados obtenidos en las fases de Exploración, Planificación y Diseño correspondientes a la metodología de desarrollo seleccionada (XP) para la implementación del sistema que se propone, donde se exponen los artefactos generados durante el transcurso de las mismas.

## 3.2. Exploración

La metodología de desarrollo Extreme Programming (XP) comienza con la fase de exploración. Durante esta se realiza el proceso de identificación de las Historias de Usuarios (HU), así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto [6].

### 3.2.1. Historias de Usuario

Las HU son la forma en que se especifican en XP los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas [6]. Durante la fase de exploración se identificaron seis historias de usuario:

- Sistema de autenticación
- Gestionar páginas
- Gestionar selectores
- Gestionar filtros
- Gestionar canales RSS
- Gestionar boletines

Las HU definidas se clasificaron atendiendo a diferentes criterios, facilitando así su organización y la correcta planificación del proyecto. Los parámetros definidos en cada una de las HU se detallan a continuación.

- Nombre: Para identificar la HU.
- Prioridad:
  - Alta: Constituyen funcionalidades principales del sistema, o forman parte esencial de la arquitectura del mismo.
  - Media: Son funcionalidades importantes y de gran valor para el usuario pero que no impiden poner el proyecto en marcha si no se tienen.
  - Baja: Son funcionalidades que sería deseable tener y podrían incluirse en caso de que hubiese recursos para ello.
- Riesgo en desarrollo
  - Alto: En caso de tener algún error de implementación, pueden afectar la disponibilidad del sistema.
  - Medio: En caso de presentar errores retrasan la entrega de la versión.
  - Bajo: En caso de presentar errores, estos pueden ser tratados con facilidad y no afectan el desarrollo del proyecto.
- Iteración asignada: Número de la iteración a la que ha sido asignada.
- Puntos estimados: Tiempo estimado de desarrollo para completar la HU. Un punto equivale a una semana ideal de programación (40 horas).
- Descripción: Es donde se define la funcionalidad que se quiere desarrollar.
- Observaciones: Detalles a tener en cuenta para desarrollar la HU correctamente.

Las HU del presente proyecto fueron elaboradas siguiendo el formato propuesto por Mike Cohn en su libro Historias de Usuario Aplicadas (User Stories Applied) [9], quedando definidas de la siguiente forma:

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Sistema de autenticación | | |
| **Prioridad:** | Alta | **Riesgo en desarrollo:** | Medio |
| **Iteración asignada:** | 1 | **Puntos estimados:** | 1 |
| **Descripción:** | Un usuario puede acceder al sistema con su nombre de usuario y contraseña del dominio UCI. | | |
| **Observaciones:** | Un usuario puede acceder al sistema, siempre y cuando el servidor LDAP de la UCI esté disponible, o ya se haya autenticado anteriormente. | | |

TABLA 3.1: Historia de Usuario - Sistema de autenticación.

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Gestionar páginas | | |
| **Prioridad:** | Alta | **Riesgo en desarrollo:** | Alto |
| **Iteración asignada:** | 1 | **Puntos estimados:** | 2 |
| **Descripción:** | Un usuario puede gestionar (crear, editar, mostrar, eliminar, recargar contenido, iniciar y detener monitoreo) las páginas web de su interés. | | |
| **Observaciones:** | La recuperación del contenido web asociado a las páginas debe realizarse de forma concurrente durante el monitoreo. En el contenido recuperado deben incluirse los estilos CSS para una correcta visualización del mismo, así como eliminar el código JavaScript. | | |

TABLA 3.2: Historia de Usuario - Gestionar páginas.

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Gestionar selectores | | |
| **Prioridad:** | Media | **Riesgo en desarrollo:** | Alto |
| **Iteración asignada:** | 2 | **Puntos estimados:** | 2 |
| **Descripción:** | Un usuario puede gestionar (crear, editar, mostrar y eliminar) selectores, para definir el contenido de su interés en cada página. | | |
| **Observaciones:** | Un selector debe estar asociado a una página, y puede tener varios filtros para refinar el contenido seleccionado. Se debe permitir la selección del contenido de forma sencilla a usuarios con pocos conocimientos sobre selectores CSS. | | |

TABLA 3.3: Historia de Usuario - Gestionar selectores.

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Gestionar filtros | | |
| **Prioridad:** | Media | **Riesgo en desarrollo:** | Medio |
| **Iteración asignada:** | 2 | **Puntos estimados:** | 1 |
| **Descripción:** | Un usuario puede gestionar (crear, editar y eliminar) filtros aplicables a los selectores, para refinar el contenido seleccionado. | | |
| **Observaciones:** | - | | |

TABLA 3.4: Historia de Usuario - Gestionar filtros.

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Gestionar canales RSS | | |
| **Prioridad:** | Baja | **Riesgo en desarrollo:** | Medio |
| **Iteración asignada:** | 3 | **Puntos estimados:** | 1 |
| **Descripción:** | Un usuario puede gestionar (crear, editar, mostrar, eliminar, clonar y desmarcar como clonado) canales RSS como vía de consumo del contenido seleccionado. | | |
| **Observaciones:** | Cada uno de los componentes (título, descripción y vínculo) asociados a los elementos de un canal RSS deben ser definidos a través de selectores. | | |

TABLA 3.5: Historia de Usuario - Gestionar canales RSS.

| Historia de Usuario | | | |
|---|---|---|---|
| **Nombre:** | Gestionar boletines | | |
| **Prioridad:** | Baja | **Riesgo en desarrollo:** | Alto |
| **Iteración asignada:** | 3 | **Puntos estimados:** | 2 |
| **Descripción:** | Un usuario puede gestionar (crear, editar, mostrar, eliminar, enviar, subscribir, desubscribir, clonar y desmarcar como clonado) boletines como vía de consumo del contenido seleccionado. | | |
| **Observaciones:** | El cuerpo del boletín puede contener selectores simples o ciclos. En el primer caso se sustituye el selector por la última versión del contenido seleccionado, mientras que en los ciclos se sustituye cada selector por todas las últimas versiones no enviadas anteriormente. | | |

TABLA 3.6: Historia de Usuario - Gestionar boletines.

## 3.3. Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizado como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario, que incluye: las pruebas unitarias, la integración y refactorización del código, y la preparación y ejecución de las pruebas de aceptación [6].

### 3.3.1. Estimación de esfuerzo por Historias de Usuario

Para el desarrollo del sistema propuesto en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los resultados que se muestran en la siguiente tabla.

| Historia de Usuario | Puntos estimados |
|---|---|
| Sistema de autenticación | 1 |
| Gestionar páginas | 2 |
| Gestionar selectores | 2 |
| Gestionar filtros | 1 |
| Gestionar canales RSS | 1 |
| Gestionar boletines | 2 |

TABLA 3.7: Estimación de esfuerzo por Historias de Usuario.

### 3.3.2. Plan de iteraciones

Una se vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa de implementación del proyecto [6]. En base a lo antes mencionado se decide realizar esta en tres iteraciones, las cuales se detallan a continuación.

#### 3.3.2.1. Iteración 1

Esta iteración tiene como objetivo la implementación de las historias de usuario de mayor prioridad. Durante el transcurso de la misma se creará la base de la arquitectura del sistema con una funcionalidad mínima. Al final de esta se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el equipo de desarrollo.

#### 3.3.2.2. Iteración 2

El objetivo de esta iteración es la implementación de las historias de usuario de prioridad media. Al finalizar la misma se contará con una versión funcional del sistema, donde se podrán probar las funcionalidades relacionadas con la selección y refinamiento del contenido web de interés. Esta versión igualmente se mostrará al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

#### 3.3.2.3. Iteración 3

Durante el transcurso de esta iteración se implementan las historias de usuario de baja prioridad. Al finalizar la misma se contará con la versión 1.0 del producto final, adicionando a las funcionalidades anteriores todo lo concerniente al sistema de distribución de la información. Como resultado de esta, el sistema estará listo para su despliegue.

#### 3.3.2.4. Duración de las iteraciones

Como parte del ciclo de vida de un proyecto que utiliza XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con que se cuenta. Este plan se encarga de mostrar las historias de usuario que serán abordadas en cada una de las iteraciones, así como la duración estimada de estas últimas y el orden en que se implementarán las HU.

| Iteración | Historias de Usuario a implementar | Duración total de la iteración |
|---|---|---|
| Iteración 1 | Sistema de autenticación | 3 semanas |
| | Gestionar páginas | |
| Iteración 2 | Gestionar selectores | 3 semanas |
| | Gestionar filtros | |
| Iteración 3 | Gestionar canales RSS | 3 semanas |
| | Gestionar boletines | |

TABLA 3.8: Plan de duración de las iteraciones.

### 3.3.3. Plan de entregas

A continuación se presenta el plan de entregas elaborado para la fase de implementación, que comienza el lunes 16 de febrero de 2015. Como producto del mismo se harán entregas incrementales del sistema al finalizar cada iteración, en la fecha aproximada que se indica en la siguiente tabla.

| Iteración | Fecha de entrega |
|---|---|
| Iteración 1 | 6/03/2015 |
| Iteración 2 | 27/03/2015 |
| Iteración 3 | 17/04/2015 |

TABLA 3.9: Plan de entregas.

## 3.4. Diseño

Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra deben ser evitados en todo momento. El diseño adecuado para el software es aquel que supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos. Se debe tener en cuenta que un diseño complejo siempre tarda más en desarrollarse que uno simple, y que siempre es más fácil añadir complejidad a un diseño simple que quitarla de uno complejo.

### 3.4.1. Patrón arquitectónico

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionan un conjunto de subsistemas predefinidos, especificando sus responsabilidades, reglas y directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. Los patrones arquitectónicos heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos reutilizables específicamente para la arquitectura general de los sistemas de información [12].

#### 3.4.1.1. Modelo Plantilla Vista (MTV)

Django, el framework de desarrollo seleccionado del lado del servidor, respeta el paradigma conocido como Modelo Plantilla Vista (MTV), el cual está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador (MVC), con ciertas diferencias. Como resultado, lo que se llamaría *controlador* en un verdadero framework MVC se llama en Django *vista*, y lo que se llamaría *vista* se llama *plantilla*. Para un mejor entendimiento se presenta el siguiente diagrama [10].

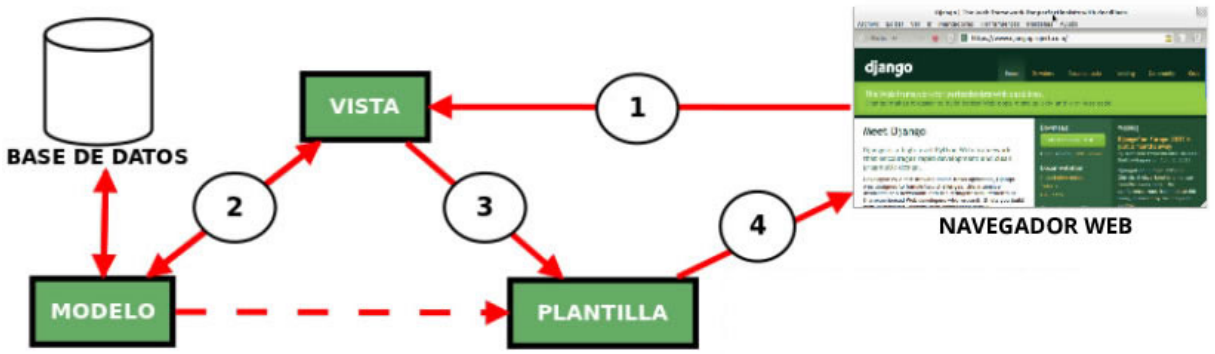


FIGURA 3.1: Diagrama del patrón Modelo Plantilla Vista (MTV)

El patrón MTV al igual que el MVC permite separar los datos, la lógica de la aplicación y la presentación gráfica con la finalidad de conseguir un código limpio y un fácil mantenimiento ya que las modificaciones de una parte de la aplicación no afectan a las demás.

### 3.4.2. Patrones de diseño

Los patrones de diseño son una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del software [15]. Para el diseño de la aplicación se hizo uso de los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como de los patrones Gang of Four ó Pandilla de los Cuatro (GoF).

Los patrones GRASP utilizados fueron:

- Alta cohesión: Se aplica en la mayoría las clases del diseño, ya que en cada una solo se implementan las funcionalidades que le corresponden.
- Bajo acoplamiento: Ya que cada clase se comunica con un número relativamente pequeño de clases.
- Creador: Las clases que tienen la responsabilidad de crear objetos contienen toda la infromación necesaria para construir los mismos.
- Experto: Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

Además se utilizaron como patrones GoF:

- Front Controller (Controlador Frontal): Django posee una implementación de Controlador Frontal que despacha las peticiones hacia métodos o clases, que en la práctica son páginas controladoras. Antes del despacho, la petición es procesada por varios filtros (middlewares).
- Decorator (Decorador): Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades.

### 3.4.3. Tarjetas CRC

En la fase de Diseño, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación Lenguaje Unificado de Modelado (UML). En su lugar se usan otras técnicas como las tarjetas Clase Responsabilidad Colaboración (CRC) como una extensión informal a UML. La técnica de las tarjetas CRC se puede usar para guiar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades [38].

A continuación se exponen las tarjetas CRC obtenidas durante la etapa de Diseño del presente trabajo.

| **Tarjeta CRC #1** | |
|---|---|
| **Clase:** Views | |
| **Responsabilidad** | **Colaboración** |
| ▪ Exponer las APIs correspondientes al sistema de autenticación y los modelos del sistema.<br>▪ Gestionar adecuadamente las peticiones GET, POST, PUT, DELETE y OPTIONS para cada una de las APIs generadas. | ▪ Models<br>▪ Serializers<br>▪ UCILDAPBackend<br>▪ RSSFeed<br>▪ Visor |

TABLA 3.10: Tarjeta CRC para la clase Views.

| **Tarjeta CRC #2** | |
|---|---|
| **Clase:** Models | |
| **Responsabilidad** | **Colaboración** |
| ▪ Definir el modelo de datos a través del ORM de Django.<br>▪ Guardar y recuperar desde la Base de Datos la información asociada a cada uno de los modelos del sistema. | ▪ django.db.models |

TABLA 3.11: Tarjeta CRC para la clase Models.

| **Tarjeta CRC #3** | |
|---|---|
| **Clase:** Serializers | |
| **Responsabilidad** | **Colaboración** |
| ▪ Definir la forma en que serán serializados los datos de cada uno de los modelos del sistema, atendiendo al formato, los campos a serializar y las relaciones entre ellos. | ▪ Models |

TABLA 3.12: Tarjeta CRC para la clase Serializers.

| **Tarjeta CRC #4** | |
|---|---|
| **Clase:** UCILDAPBackend | |
| **Responsabilidad** | **Colaboración** |
| ▪ Realizar la autenticación a través del servicio LDAP de la UCI y gestionar la información correspondiente al usuario autenticado. | ▪ Models |

TABLA 3.13: Tarjeta CRC para la clase UCILDAPBackend.

| **Tarjeta CRC #5** | |
|---|---|
| **Clase:** RSSFeed | |
| **Responsabilidad** | **Colaboración** |
| ▪ Generar canales RSS personalizados a partir del contenido web almacenado. | ▪ Models |

TABLA 3.14: Tarjeta CRC para la clase RSSFeed.

| **Tarjeta CRC #6** | |
|---|---|
| **Clase:** Visor | |
| **Responsabilidad** | **Colaboración** |
| ▪ Iniciar y detener la recuperación de contenido web por cada página definida.<br>▪ Parsear, seleccionar, filtrar, formatear y almacenar el contenido recuperado. | ▪ Models |

TABLA 3.15: Tarjeta CRC para la clase Visor.

## 3.5. Conclusiones del capítulo

Con la realización de este capítulo se especificaron los resultados obtenidos durante las etapas de Exploración, Planificación y Diseño al aplicar la metodología de desarrollo XP. Se identificaron seis HU; de ellas dos fueron catalogadas de prioridad alta, dos de prioridad media y dos de prioridad baja. Se elaboraron los artefactos correspondientes a las etapas de Exploración, Planificación y Diseño, definiéndose además las funcionalidades que tendrá el sistema así como la planificación del esfuerzo dedicado a la realización de cada una de estas en el orden en que se les dará cumplimiento según las necesidades del cliente. Además permitió definir una fecha estimada en la que se le entregará una primera versión del producto al cliente. Se definieron los patrones arquitectónicos y de diseño usados con el objetivo de lograr una mayor organización en los elementos que conforman la aplicación, así como las tarjetas CRC para una mejor comprensión de la misma.

CAPÍTULO 4

# Implementación y Pruebas del sistema de consumo y reutilización de contenido web

## 4.1. Introducción

La metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para retroalimentar a los desarrolladores con la opinión de este. En el presente capítulo se describen los estándares de codificación utilizados para obtener un código limpio y legible. Se detallan además las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario, así como las pruebas unitarias y de aceptación efectuadas sobre el proyecto.

## 4.2. Estándares de codificación

Según Guido van Rossum (científico de la computación, conocido por ser el autor del lenguaje de programación Python), el código es leído muchas más veces de lo que es escrito. Por tanto se hace necesario definir pautas para lograr una mejor legibilidad del código y hacerlo consistente.

Para lograr este objetivo se utilizó la Guía de estilo para el código Python (PEP 8) [32]. Esta guía posee una gran cantidad de convenciones para escribir código legible, dentro de las cuales se destacan:

- Usar cuatro espacios por indentación.
- Nunca mezclar tabulaciones y espacios.
- Limitar todas las líneas a un máximo de caracteres (120 en este proyecto).
- Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las definiciones de métodos dentro de una clase son separadas por una línea en blanco.
- Codificación UTF-8 en todos los módulos.

- Las importaciones deben estar en líneas separadas.
- Evitar usar espacios en blanco innecesarios.
- Utilizar el estilo CamelCase para nombrar clases, y el lower_case_with_underscores para funciones y métodos.

## 4.3. Desarrollo por iteraciones

Durante el transcurso de las iteraciones se realiza la implementación de las historias de usuario seleccionadas para cada una de estas. Al inicio de las mismas, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Como parte de este plan, se descomponen las HU en tareas de ingeniería. Estas tareas son para el uso de los programadores, pueden escribirse utilizando un lenguaje técnico y no necesariamente deben ser entendibles para el cliente [6].

Ajustándose a la planificación realizada, se llevaron a cabo tres iteraciones de desarrollo sobre el sistema, obteniéndose al finalizar un producto listo para su despliegue. A continuación se detallan cada una de las iteraciones.

### 4.3.1. Pruebas

A las pruebas se le confiere un valor esencial dentro del desarrollo de aplicaciones ya que un fallo en estas puede representar costos elevados. Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. Para la validación de un software de manera general se utilizan principalmente dos métodos de prueba: caja blanca y caja negra. Las pruebas de caja blanca basan su funcionamiento en evaluar la información sobre cómo el software ha sido diseñado y codificado. El método caja negra actúa sobre la validación de los requisitos funcionales y se aplican a la interfaz del software para examinar algún aspecto funcional de un sistema, por lo que también se les denominan pruebas funcionales o de aceptación [7].

Durante el desarrollo de cada una de las iteraciones del presente proyecto se realizaron pruebas de caja blanca (pruebas unitarias) y de caja negra (pruebas de aceptación). Las mismas fueron

evaluadas automáticamente en cada una de las iteraciones, obteniéndose un resultado final que se muestra en las próximas secciones.

## 4.4. Iteración 1

Durante esta iteración se abordaron las historias de usuario de mayor prioridad y se construyó la base de la arquitectura del sistema con el fin de obtener un producto con las funcionalidades críticas para ser mostrado al cliente y obtener una rápida retroalimentación de este.

### 4.4.1. Tareas de Ingeniería

#### 4.4.1.1. HU - Sistema de autenticación

| **Tarea de Ingeniería** | | | |
|---|---|---|---|
| **Nombre:** | API de autenticación | | |
| **HU:** | Sistema de autenticación | **Puntos estimados:** | 0.3 |
| **Fecha de inicio:** | 16/02/2015 | **Fecha de fin:** | 17/02/2015 |
| **Descripción:** | Se debe proveer una API de autenticación basada en tokens, donde las credenciales a recibir son el nombre de usuario y contraseña del dominio UCI. Los datos del usuario autenticado deben guardarse en la base de datos local para permitir el acceso sin conexión con el servidor LDAP. | | |

TABLA 4.1: Tarea de Ingeniería - API de autenticación.

| **Tarea de Ingeniería** | | | |
|---|---|---|---|
| **Nombre:** | Autenticar usuario | | |
| **HU:** | Sistema de autenticación | **Puntos estimados:** | 0.4 |
| **Fecha de inicio:** | 17/02/2015 | **Fecha de fin:** | 19/02/2015 |
| **Descripción:** | Se debe mostrar un formulario de autenticación para que el usuario entre sus credenciales (nombre de usuario y contraseña). Los datos deberán ser válidos para autenticarse, mostrando un mensaje de error en caso contrario. No se permitirá el acceso a las funcionalidades del sistema hasta que el usuario no se haya autenticado. | | |

TABLA 4.2: Tarea de Ingeniería - Autenticar usuario.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Cerrar sesión | | |
| **HU:** | Sistema de autenticación | **Puntos estimados:** | 0.3 |
| **Fecha de inicio:** | 19/02/2015 | **Fecha de fin:** | 20/02/2015 |
| **Descripción:** | Se debe permitir al usuario cerrar la sesión desde cualquier página, redirigiéndolo inmediatamente hacia la página de acceso. | | |

TABLA 4.3: Tarea de Ingeniería - Cerrar sesión.

#### 4.4.1.2. HU - Gestionar páginas

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | API para las páginas | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.4 |
| **Fecha de inicio:** | 23/02/2015 | **Fecha de fin:** | 24/02/2015 |
| **Descripción:** | Se debe proveer una API para gestionar las páginas creadas por los usuarios, garantizando un correcto funcionamiento de los procesos de acceso, recuperación y monitoreo de la información. | | |

TABLA 4.4: Tarea de Ingeniería - API para las páginas.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Listar páginas | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 25/02/2015 | **Fecha de fin:** | 25/02/2015 |
| **Descripción:** | Un usuario puede ver una tabla con las páginas que tiene guardadas y un resumen de cada una (nombre, url y disponibilidad), permitiendo gestionar (editar, mostrar, eliminar, actualizar contenido, iniciar y detener monitoreo) cada una de ellas. | | |

TABLA 4.5: Tarea de Ingeniería - Listar páginas.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Añadir página | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 26/02/2015 | **Fecha de fin:** | 26/02/2015 |
| **Descripción:** | Un usuario puede añadir una página llenando el formulario correspondiente con los datos requeridos (nombre, url, tiempo de espera entre comprobaciones, necesidad de utilizar el servidor proxy de la UCI). | | |

TABLA 4.6: Tarea de Ingeniería - Añadir página.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Editar página | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 27/02/2015 | **Fecha de fin:** | 27/02/2015 |
| **Descripción:** | Un usuario puede editar una página de su listado, permitiendo cambiar los datos de la misma y actualizando el contenido asociado. | | |

TABLA 4.7: Tarea de Ingeniería - Editar página.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Mostrar página | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 2/03/2015 | **Fecha de fin:** | 2/03/2015 |
| **Descripción:** | Un usuario puede ver el contenido asociado a una página de su listado. Se deben cargar los archivos externos para una correcta visualización del contenido, así como desabilitar JavaScript para una mayor seguridad. | | |

TABLA 4.8: Tarea de Ingeniería - Mostrar página.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Eliminar página | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 3/03/2015 | **Fecha de fin:** | 3/03/2015 |
| **Descripción:** | Un usuario puede eliminar una página de su listado. Los selectores asociados a la página serán elimininados también. Esta acción debe confirmarse debido a que es irreversible. | | |

TABLA 4.9: Tarea de Ingeniería - Eliminar página.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Actualizar contenido | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 4/03/2015 | **Fecha de fin:** | 4/03/2015 |
| **Descripción:** | Un usuario puede actualizar el contenido de una página de su listado. | | |

TABLA 4.10: Tarea de Ingeniería - Actualizar contenido.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Iniciar monitoreo | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 5/03/2015 | **Fecha de fin:** | 5/03/2015 |
| **Descripción:** | Un usuario puede iniciar el proceso de monitoreo para una o varias páginas de su listado. Mientras el monitoreo esté activo, se debe actualizar repetidamente (con un tiempo de espera definido) el contenido asociado a la página, además de actualizar también el contenido que extraen los selectores correspondientes a la misma. | | |

TABLA 4.11: Tarea de Ingeniería - Iniciar monitoreo.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Detener monitoreo | | |
| **HU:** | Gestionar páginas | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 6/03/2015 | **Fecha de fin:** | 6/03/2015 |
| **Descripción:** | Un usuario puede detener el proceso de monitoreo para una o varias páginas de su listado. | | |

TABLA 4.12: Tarea de Ingeniería - Detener monitoreo.

### 4.4.2. Pruebas

#### 4.4.2.1. Pruebas Unitarias

En esta primera iteración se realizaron 10 pruebas unitarias en el lado del servidor y 37 pruebas unitarias del lado del cliente. A continuación se muestran estos resultados de manera gráfica.

```
Creating test database for alias 'default'...
..........
----------------------------------------------
Ran 10 tests in 1.803s

OK
Destroying test database for alias 'default'...
```

FIGURA 4.1: Pruebas Unitarias en el lado del servidor - Iteración 1

```
Running "karma:unit" (karma) task
INFO [karma]: Karma v0.12.31 server started at http://localhost:8080/
INFO [launcher]: Starting browser PhantomJS
INFO [PhantomJS 1.9.8 (Linux)]: Connected on socket m7AAibBA4gNEoynznIQf with id 75139824
PhantomJS 1.9.8 (Linux): Executed 37 of 37 SUCCESS (0.47 secs / 0.713 secs)
```

FIGURA 4.2: Pruebas Unitarias en el lado del cliente - Iteración 1

#### 4.4.2.2. Pruebas de Aceptación

Para relizar las pruebas de aceptación correctamente se elaboraron casos de pruebas para cada una de ellas. Seguidamente se describe el caso de prueba correspondiente a la HU

Sistema de autenticación. Los casos de prueba de las demás HU pueden consultarse en el Anexo A.

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Nombre:** | Sistema de autenticación - Datos válidos |
| **Historia de Usuario:** | Sistema de autenticación |
| **Descripción:** | Se prueba el acceso al sistema con credenciales correctas. Comprobando los elementos del escenario correspondiente. |
| **Precondiciones:** | Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Acceder al sistema con credenciales correctas.<br>▪ Cerrar sesión. |

TABLA 4.13: Caso de Prueba de Aceptación - Sistema de autenticación - Datos válidos.

| **Caso de Prueba de Aceptación** | |
|---|---|
| **Nombre:** | Sistema de autenticación - Datos inválidos |
| **Historia de Usuario:** | Sistema de autenticación |
| **Descripción:** | Se intenta acceder al sistema y sus funcionalidades con credenciales incorrectas. En cada caso se comprueban los mensajes de error, la redirección al sistema de autenticación y los elementos del escenario correspondiente. |
| **Precondiciones:** | - |
| **Pasos de ejecución:** | ▪ Intentar acceder a una página restringida sin autenticarse.<br>▪ Intentar acceder con credenciales incorrectas. |

TABLA 4.14: Caso de Prueba de Aceptación - Sistema de autenticación - Datos inválidos.

En esta iteración se realizaron 2 pruebas de aceptación, una por cada HU a implementar. A continuación se muestran los resultados arrojados durante la ejecución de ambas pruebas en conjunto.

```
Finished in 51.231 seconds
2 tests, 15 assertions, 0 failures
```

FIGURA 4.3: Pruebas de Aceptación - Iteración 1

## 4.5. Iteración 2

En esta segunda iteración se desarrollaron las HU de prioridad media, donde se implementaron las funcionalidades relacionadas con la selección y refinamiento del contenido web de interés.

### 4.5.1. Tareas de Ingeniería

#### 4.5.1.1. HU - Gestionar selectores

| **Tarea de Ingeniería** | | | |
|---|---|---|---|
| **Nombre:** | API para los selectores | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.4 |
| **Fecha de inicio:** | 9/03/2015 | **Fecha de fin:** | 10/03/2015 |
| **Descripción:** | Se debe proveer una API para gestionar los selectores de contenido, garantizando un correcto funcionamiento de los procesos de selección de la información. | | |

TABLA 4.15: Tarea de Ingeniería - API para los selectores.

| **Tarea de Ingeniería** | | | |
|---|---|---|---|
| **Nombre:** | Listar selectores | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 11/03/2015 | **Fecha de fin:** | 11/03/2015 |
| **Descripción:** | Un usuario puede ver una tabla con los selectores que tiene guardados y un resumen de cada uno (nombre, selector CSS y tipo de contenido), permitiendo gestionar (editar, mostrar contenido y eliminar) cada uno de ellos. | | |

TABLA 4.16: Tarea de Ingeniería - Listar selectores.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Añadir selector | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.4 |
| **Fecha de inicio:** | 12/03/2015 | **Fecha de fin:** | 13/03/2015 |
| **Descripción:** | Un usuario puede añadir un selector llenando el formulario correspondiente con los datos requeridos (página, nombre, selector, tipo de contenido, filtros, condición de guardado y palabras claves). | | |

TABLA 4.17: Tarea de Ingeniería - Añadir selector.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Editar selector | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.3 |
| **Fecha de inicio:** | 16/03/2015 | **Fecha de fin:** | 17/03/2015 |
| **Descripción:** | Un usuario puede editar un selector de su listado, permitiendo cambiar los datos del mismo y actualizando el contenido asociado. | | |

TABLA 4.18: Tarea de Ingeniería - Editar selector.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Mostrar contenido del selector | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.4 |
| **Fecha de inicio:** | 17/03/2015 | **Fecha de fin:** | 19/03/2015 |
| **Descripción:** | Un usuario puede ver el contenido asociado a un selector de su listado, el que debe ser mostrado por versiones y en orden cronológico descendente. | | |

TABLA 4.19: Tarea de Ingeniería - Mostrar contenido del selector.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Eliminar selector | | |
| **HU:** | Gestionar selectores | **Puntos estimados:** | 0.3 |
| **Fecha de inicio:** | 19/03/2015 | **Fecha de fin:** | 20/03/2015 |
| **Descripción:** | Un usuario puede eliminar un selector de su listado. Esta acción debe confirmarse debido a que es irreversible. | | |

TABLA 4.20: Tarea de Ingeniería - Eliminar selector.

#### 4.5.1.2. HU - Gestionar filtros

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | API para los filtros | | |
| **HU:** | Gestionar filtros | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 23/03/2015 | **Fecha de fin:** | 23/03/2015 |
| **Descripción:** | Se debe proveer una API para gestionar los filtros de contenido, garantizando un correcto funcionamiento de los procesos de refinamiento de la información. | | |

TABLA 4.21: Tarea de Ingeniería - API para los filtros.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Listar filtros | | |
| **HU:** | Gestionar filtros | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 24/03/2015 | **Fecha de fin:** | 24/03/2015 |
| **Descripción:** | Un usuario puede ver una tabla con los filtros que tiene guardados y un resumen de cada uno (nombre, expresión, tipo y salida), permitiendo gestionar (editar y eliminar) cada uno de ellos. | | |

TABLA 4.22: Tarea de Ingeniería - Listar filtros.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Añadir filtro | | |
| **HU:** | Gestionar filtros | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 25/03/2015 | **Fecha de fin:** | 25/03/2015 |
| **Descripción:** | Un usuario puede añadir un filtro llenando el formulario correspondiente con los datos requeridos (nombre, tipo, expresión y salida). | | |

TABLA 4.23: Tarea de Ingeniería - Añadir filtro.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Editar filtro | | |
| **HU:** | Gestionar filtros | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 26/03/2015 | **Fecha de fin:** | 26/03/2015 |
| **Descripción:** | Un usuario puede editar un filtro de su listado, permitiendo cambiar los datos del mismo. | | |

TABLA 4.24: Tarea de Ingeniería - Editar filtro.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Eliminar filtro | | |
| **HU:** | Gestionar filtros | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 27/03/2015 | **Fecha de fin:** | 27/03/2015 |
| **Descripción:** | Un usuario puede eliminar un filtro de su listado. Esta acción debe confirmarse debido a que es irreversible. | | |

TABLA 4.25: Tarea de Ingeniería - Eliminar filtro.

### 4.5.2. Pruebas

#### 4.5.2.1. Pruebas Unitarias

Al concluir esta segunda iteración el total de pruebas unitarias en el lado del servidor ascendió a 22, mientras que las pruebas unitarias del lado del cliente fueron 74. A continuación se muestran

estos resultados de manera gráfica.

```
Creating test database for alias 'default'...
......................
------------------------------------------------
Ran 22 tests in 3.889s

OK
Destroying test database for alias 'default'...
```

FIGURA 4.4: Pruebas Unitarias en el lado del servidor - Iteración 2

```
Running "karma:unit" (karma) task
WARN [karma]: Port 8080 in use
INFO [karma]: Karma v0.12.31 server started at http://localhost:8081/
INFO [launcher]: Starting browser PhantomJS
INFO [PhantomJS 1.9.8 (Linux)]: Connected on socket sJGiQVaH9W_BrWYYvggE with id 31144709
PhantomJS 1.9.8 (Linux): Executed 74 of 74 SUCCESS (1.055 secs / 1.238 secs)
```

FIGURA 4.5: Pruebas Unitarias en el lado del cliente - Iteración 2

#### 4.5.2.2. Pruebas de Aceptación

En esta iteración se realizaron 2 pruebas de aceptación (una por cada HU a implementar), sumando un total de 4. A continuación se muestran los resultados arrojados durante la ejecución de estas pruebas en conjunto.

```
Finished in 113.738 seconds
4 tests, 31 assertions, 0 failures
```

FIGURA 4.6: Pruebas de Aceptación - Iteración 2

## 4.6. Iteración 3

En la tercera iteración (final) se desarrollaron las HU de prioridad baja, donde se implementaron las funcionalidades relacionadas con la distribución del contenido web seleccionado.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | API para los canales RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 30/03/2015 | **Fecha de fin:** | 30/03/2015 |
| **Descripción:** | Se debe proveer una API para gestionar los canales RSS, garantizando un correcto funcionamiento del proceso de distribución de la información a través de esta vía. | | |

TABLA 4.26: Tarea de Ingeniería - API para los canales RSS.

### 4.6.1. Tareas de Ingeniería

#### 4.6.1.1. HU - Gestionar canales RSS

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Listar canales RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 31/03/2015 | **Fecha de fin:** | 31/03/2015 |
| **Descripción:** | Un usuario puede ver una tabla con los canales RSS que tiene guardados y un resumen de cada uno (título, descripción y visibilidad), permitiendo gestionar (editar, mostrar y eliminar) cada uno de ellos. Se deben mostrar además los canales RSS públicos, permitiendo mostrarlos, clonarlos y desmarcarlos como clonados. | | |

TABLA 4.27: Tarea de Ingeniería - Listar canales RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Añadir canal RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 31/03/2015 | **Fecha de fin:** | 31/03/2015 |
| **Descripción:** | Un usuario puede añadir un canal RSS llenando el formulario correspondiente con los datos requeridos (título, descripción, vínculo, visibilidad y elementos del canal). | | |

TABLA 4.28: Tarea de Ingeniería - Añadir canal RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Mostrar canal RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 1/04/2015 | **Fecha de fin:** | 1/04/2015 |
| **Descripción:** | Un usuario puede mostrar un canal RSS contruído a partir de los datos del mismo, permitiendo realizar todas las acciones de un canal RSS común. | | |

TABLA 4.29: Tarea de Ingeniería - Mostrar canal RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Editar canal RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 1/04/2015 | **Fecha de fin:** | 1/04/2015 |
| **Descripción:** | Un usuario puede editar un canal RSS de su listado, permitiendo cambiar los datos del mismo. | | |

TABLA 4.30: Tarea de Ingeniería - Editar canal RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Eliminar canal RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 2/04/2015 | **Fecha de fin:** | 2/04/2015 |
| **Descripción:** | Un usuario puede eliminar un canal RSS de su listado. Esta acción debe confirmarse debido a que es irreversible. | | |

TABLA 4.31: Tarea de Ingeniería - Eliminar canal RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Clonar canal RSS | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 2/04/2015 | **Fecha de fin:** | 3/04/2015 |
| **Descripción:** | Un usuario puede clonar un canal RSS público para modificarlo de acuerdo a sus necesidades. Esta acción debe confirmarse debido a que se clonarán además las páginas, selectores y filtros asociados al canal RSS. | | |

TABLA 4.32: Tarea de Ingeniería - Clonar canal RSS.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Desmarcar canal RSS como clonado | | |
| **HU:** | Gestionar canales RSS | **Puntos estimados:** | 0.1 |
| **Fecha de inicio:** | 3/04/2015 | **Fecha de fin:** | 3/04/2015 |
| **Descripción:** | Un usuario puede desmarcar un canal RSS público como clonado, permitiéndole volver a clonarlo. | | |

TABLA 4.33: Tarea de Ingeniería - Desmarcar canal RSS como clonado.

#### 4.6.1.2. HU - Gestionar boletines

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | API para los boletines | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 6/04/2015 | **Fecha de fin:** | 6/04/2015 |
| **Descripción:** | Se debe proveer una API para gestionar los boletines, garantizando un correcto funcionamiento del proceso de distribución de la información a través de esta vía. | | |

TABLA 4.34: Tarea de Ingeniería - API para los boletines.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Listar boletines | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 7/04/2015 | **Fecha de fin:** | 7/04/2015 |
| **Descripción:** | Un usuario puede ver una tabla con los boletines que tiene guardados y un resumen de cada uno (nombre, asunto y visibilidad), permitiendo gestionar (editar, enviar y eliminar) cada uno de ellos. Se deben mostrar además los boletines públicos, permitiendo mostrarlos, enviarlos, subscribirse, cancelar subscripción, clonarlos y desmarcarlos como clonados. | | |

TABLA 4.35: Tarea de Ingeniería - Listar boletines.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Añadir boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 8/04/2015 | **Fecha de fin:** | 8/04/2015 |
| **Descripción:** | Un usuario puede añadir un boletín llenando el formulario correspondiente con los datos requeridos (nombre, visibilidad, asunto y cuerpo). El cuerpo del boletín puede contener selectores simples, que deben ser sustituidos por la última versión del mismo, o ciclos con selectores dentro, los cuales serán sustituidos por las últimas versiones no enviadas. | | |

TABLA 4.36: Tarea de Ingeniería - Añadir boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Mostrar boletín público | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 9/04/2015 | **Fecha de fin:** | 9/04/2015 |
| **Descripción:** | Un usuario puede mostrar un boletín público para ver con detalles los datos del mismo, aunque no puede editar dichos datos. | | |

TABLA 4.37: Tarea de Ingeniería - Mostrar boletín público.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Editar boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 10/04/2015 | **Fecha de fin:** | 10/04/2015 |
| **Descripción:** | Un usuario puede editar un boletín de su listado, permitiendo cambiar los datos del mismo. | | |

TABLA 4.38: Tarea de Ingeniería - Editar boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Eliminar boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 13/04/2015 | **Fecha de fin:** | 13/04/2015 |
| **Descripción:** | Un usuario puede eliminar un boletín de su listado. Esta acción debe confirmarse debido a que es irreversible. | | |

TABLA 4.39: Tarea de Ingeniería - Eliminar boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Subscribirse a un boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 14/04/2015 | **Fecha de fin:** | 14/04/2015 |
| **Descripción:** | Un usuario puede subscribirse a un boletín público para recibir la información del mismo, exactamente como lo hace el usuario que crea el boletín. | | |

TABLA 4.40: Tarea de Ingeniería - Subscribirse a un boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Cancelar subscripción a un boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 15/04/2015 | **Fecha de fin:** | 15/04/2015 |
| **Descripción:** | Un usuario puede cancelar su subscripción a un boletín público para dejar de recibir la información del mismo. | | |

TABLA 4.41: Tarea de Ingeniería - Cancelar subscripción a un boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Clonar boletín | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 16/04/2015 | **Fecha de fin:** | 16/04/2015 |
| **Descripción:** | Un usuario puede clonar un boletín público para modificarlo de acuerdo a sus necesidades. Esta acción debe confirmarse debido a que se clonarán además las páginas, selectores y filtros asociados al boletín. | | |

TABLA 4.42: Tarea de Ingeniería - Clonar boletín.

| Tarea de Ingeniería | | | |
|---|---|---|---|
| **Nombre:** | Desmarcar boletín como clonado | | |
| **HU:** | Gestionar boletines | **Puntos estimados:** | 0.2 |
| **Fecha de inicio:** | 17/04/2015 | **Fecha de fin:** | 17/04/2015 |
| **Descripción:** | Un usuario puede desmarcar un boletín público como clonado, permitiéndole volver a clonarlo. | | |

TABLA 4.43: Tarea de Ingeniería - Desmarcar boletín como clonado.

### 4.6.2. Pruebas

#### 4.6.2.1. Pruebas Unitarias

Con esta iteración se termina la aplicación, quedando lista para su despliegue. Al concluir la misma se tenían un total de 44 pruebas unitarias en el lado del servidor, así como 122 pruebas unitarias del lado del cliente. A continuación se muestran estos resultados de manera gráfica.

```
Creating test database for alias 'default'...
.............................................
----------------------------------------------
Ran 44 tests in 8.289s

OK
Destroying test database for alias 'default'...
```

FIGURA 4.7: Pruebas Unitarias en el lado del servidor - Iteración 3

```
Running "karma:unit" (karma) task
WARN [karma]: Port 8080 in use
INFO [karma]: Karma v0.12.31 server started at http://localhost:8081/
INFO [launcher]: Starting browser PhantomJS
INFO [PhantomJS 1.9.8 (Linux)]: Connected on socket PHIBOY1N0PU44BdqwP0C with id 59221254
PhantomJS 1.9.8 (Linux): Executed 122 of 122 SUCCESS (1.791 secs / 1.978 secs)
```

FIGURA 4.8: Pruebas Unitarias en el lado del cliente - Iteración 3

#### 4.6.2.2. Pruebas de Aceptación

En esta iteración se realizaron 2 pruebas de aceptación más (una por cada HU a implementar), sumando un total de 6. A continuación se muestran los resultados arrojados durante la ejecución de todas estas pruebas en conjunto.

```
Finished in 149.004 seconds
6 tests, 41 assertions, 0 failures
```

FIGURA 4.9: Pruebas de Aceptación - Iteración 3

## 4.7. Conclusiones del capítulo

En este capítulo se llevaron a cabo las fases de implementación y pruebas planteadas por la metodología XP. Se realizaron las tareas de ingeniería que dieron solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema y se realizaron las pruebas unitarias y de aceptación con resultados satisfactorios, cuyo objetivo fue contribuir a elevar la calidad final del producto, validando cada una de las funcionalidades para evitar que software con defectos llegue al cliente.

# Conclusiones

El desarrollo de este trabajo permitió definir las principales tecnologías y herramientas que facilitan a los usuarios el consumo y reutilización de información actualizada, lo cual garantizó una base metodológica sobre los elementos que deben caracterizar el proceso de consumo y reutilización de contenido web. Se obtuvo un modelo guía para la implementación del sistema mediante la generación de los artefactos correspondientes a los flujos de trabajo propuestos por la metodología XP y se ha obtenido una herramienta que permite informatizar el proceso de consumo y reutilización de contenido web, posibilitando que se cumpla satisfactoriamente el objetivo trazado para este trabajo de diploma, dando solución a la problemática planteada inicialmente por el cliente.

# Recomendaciones

Se recomienda continuar con el desarrollo de la aplicación propuesta, agregando nuevas funcionalidades, como por ejemplo:

- Permitir la recuperación de información que esté protegida por un sistema de autenticación.
- Consumir información desde canales RSS u otros tipos de archivos de forma adecuada.

Se propone además desplegar la aplicación en la Universidad de las Ciencias Informáticas para comprobar su utilidad en un entorno real, así como validar su posible implementación para las demás organizaciones del país.

# Acrónimos

**TIC** Tecnologías de la Información y las Comunicaciones . . . . . 1

**UCI** Universidad de las Ciencias Informáticas . . . . . 1

**API** Interfaz de Programación de Aplicaciones . . . . . 1

**RSS** Really Simple Syndication . . . . . 4

**HU** Historias de Usuarios . . . . . 21

**XP** Extreme Programming . . . . . 21

**GRASP** Patrones Generales de Software para Asignar Responsabilidades . . . . . 29

**GoF** Gang of Four ó Pandilla de los Cuatro . . . . . 29

**UML** Lenguaje Unificado de Modelado . . . . . 30

**CRC** Clase Responsabilidad Colaboración . . . . . 30

# Referencias bibliográficas


ALCHIN, Marty. 2009. *Pro Django Second Edition*. 2009.

ALEXANDER, Chris. 2014. Slow responsiveness on APIs and website. *Import.io*. 2014. URL: ⟨http://blog.import.io/status-blog/slow-responsiveness-on-apis-and-website⟩.

ALOY, Antoni. 2011. Introducción a Celery. *APSL Blog*. 2011. URL: ⟨http://blog.apsl.net/weblog/2011/01/14/introduccion-a-celery/⟩.

ÁLVAREZ, Miguel Angel. 2008. Qué es RSS. *DesarrolloWeb.com* [online]. 2008, [visitado 2015-01-11]. dirección: ⟨http://www.desarrolloweb.com/articulos/que-es-rss.html⟩.

BASALO, Alberto y ÁLVAREZ, Miguel Angel. 2014. Qué es AngularJS. *DesarrolloWeb.com*. 2014. URL: ⟨http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html⟩.

BECK, Kent. 2000. *Extreme Programming Explained. Embrace Change*. 2000.

BERTOLINO, Antonia. 2007. Software testing research: Achievements, challenges, dreams. En. *2007 Future of Software Engineering*. 2007, págs. 85-103.

COCKBURN, Alistair. 2000. Selecting a project's methodology. *IEEE software*. 2000, vol. 17, n.º 4, págs. 64-71.

COHN, Mike. 2004. *User stories applied: For agile software development*. 2004.

CONDORI AYALA, José Luis. 2012. Phython-DjangoFramework de desarrollo web para perfeccionistas. Basado en el Modelo MTV. *Revista de Información Tecnología y Sociedad*. 2012, págs. 36.

DOUGLAS, Korry y DOUGLAS, Susan. 2003. *PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases*. 2003.

DS, Ingenio. 2013. Patrones Arquitectónicos. *IngenioDS*. 2013. URL: ⟨https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/⟩.

FLANAGAN, David. 2006. *JavaScript: the definitive guide*. 2006.

FLANAGAN, David. 2010. *jQuery Pocket Reference*. 2010.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph y VLISSIDES, John. 1994. *Design patterns: elements of reusable object-oriented software*. 1994.

GOLDSTEIN, Alexis; WEYL, Estelle y LAZARIS, Louis. 2011. *HTML5 & CSS3 for the Real World*. 2011.

GONZÁLEZ, Raúl. 2006. Python para todos. *Creative Commons*. 2006, vol. 2.

GROUP, Miniwatts Marketing. 2013. World Internet Users and Population Stats. *Internet World Stats*. 2013. URL: ⟨http://www.internetworldstats.com/stats.htm⟩.

GUERRERO, Luis A. 2000. Modelando Interfaces para Aplicaciones Web. *Departamento de Ciencias de la Computación. Universidad de Chile.* 2000.

HOLOVATY, Adrian y KAPLAN-MOSS, Jacob. 2009. The Definitive Guide to Django. *Estados Unidos: Editorial Apress*. 2009, págs. 41.

HORTELANO, Juan Luis. 2011. Estadisticas sobre el uso de RSS. [online]. 2011, [visitado 2015-01-11]. dirección: ⟨http://www.tecnorantes.com/2005/10/11/estadisticas-sobre-el-uso-de-rss/⟩.

LUTZ, Mark. 2013. *Learning python*. 2013.

MACDONALD, Matthew. 2013. *HTML5: The Missing Manual, 2nd Edition*. 2013.

MARÍA, Eva. 2010. Boletín electrónico. *Consumoteca - Información y consejo a los consumidores en España* [online]. 2010, [visitado 2015-01-11]. dirección: ⟨http://www.consumoteca.com/telecomunicaciones/internet/boletin-electronico/⟩.

MARQUÉS, Asier. 2013. Conceptos sobre APIs REST. 2013. URL: ⟨http://asiermarques.com/2013/conceptos-sobre-apis-rest/⟩.

MARSHAL, Sarah. 2013. Data scraping tool for non-coding journalists launches Media news. [online]. 2013, [visitado 2015-01-12]. dirección: ⟨https://www.journalism.co.uk/news/data-scraping-tool-for-non-coding-journalists-launches/s2/a554002/⟩.

PAUTASSO, Cesare. 2014. RESTful web services: principles, patterns, emerging technologies. En. *Web Services Foundations*. 2014, págs. 31-51.

PINGDOM, Royal. 2013. Internet 2012 in numbers. [online]. 2013, [visitado 2015-01-12]. dirección: ⟨http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/⟩.

POLIVALENCIA. 2014. ¿Cómo sobrevivir al exceso de información en la red? *Polivalencia.com* [online]. 2014, [visitado 2015-01-11]. dirección: ⟨http://polivalencia.com/not/1207/_como_sobrevivir_al_exceso_de_informacion_en_la_red_/⟩.

RAMÍREZ, Elizabeth. 2013. Web Scraping: excavando en la red. *Hackers & Developers Magazine*. 2013, n.º 3.

RICHARDSON, Leonard. 2014. Beautiful Soup: We called him Tortoise because he taught us. 2014. URL: ⟨http://www.crummy.com/software/BeautifulSoup/⟩.

ROSSUM, Guido van; WARSAW, Barry y COGHLAN, Nick. 2001. PEP 8 Style guide for python code. 2001. URL: ⟨http://www.python.org/dev/peps/pep-0008⟩.

RSS. 2014. Ventajas Sistema RSS. [online]. 2014, [visitado 2015-01-11]. dirección: ⟨http://www.rss.nom.es/ventajas-sistema-rss/⟩.

SABHARWAL, Navin y WADHWA, Manak. 2014. *Automation through Chef Opscode*. 2014.

SCRAPING.PRO. 2014. Software for Web Scraping. *Web Scraping*. 2014. URL: ⟨http://scraping.pro/software-for-web-scraping/⟩.

SHILOV, Michael. 2014. Software for Web Scraping. *Web Scraping* [online]. 2014, [visitado 2015-01-12]. dirección: ⟨http://scraping.pro/software-for-web-scraping/⟩.

VARGIU, Eloisa y URRU, Mirko. 2012. Exploiting web scraping in a collaborative filtering-based approach to web advertising. *Artificial Intelligence Research* [online]. 2012, vol. 2, n.º 1 [visitado 2015-01-12]. dirección: ⟨http://www.sciedu.ca/journal/index.php/air/article/view/1390⟩. ISSN 1927-6982, 1927-6974. ISSN 1927-6982, 1927-6974.

WALLACE, Doug y RAGGETT, Isobel. 2003. *Extreme Programming for Web Projects*. 2003.

# Anexo A. Pruebas de Aceptación

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar páginas - Datos válidos. |
| **Historia de Usuario:** | Gestionar páginas. |
| **Descripción:** | Se prueba que se realice de forma correcta la gestión de páginas, haciendo énfasis en las operaciones listar, añadir, editar, eliminar, mostrar, actualizar contenido, iniciar y detener monitoreo. En cada caso se comprueban los mensajes de notificación y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ■ Acceder al listado de páginas.<br>■ Añadir una nueva página.<br>■ Editar la página creada.<br>■ Mostrar la página creada.<br>■ Actualizar contenido de la página creada.<br>■ Iniciar monitoreo de la página creada.<br>■ Detener monitoreo de la página creada.<br>■ Eliminar la página creada. |

TABLA 4.44: Caso de Prueba de Aceptación - Gestionar páginas - Datos válidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar páginas - Datos inválidos. |
| **Historia de Usuario:** | Gestionar páginas. |
| **Descripción:** | Se prueba que se validen de forma correcta los datos correspondientes a las páginas, haciendo énfasis en las operaciones añadir y editar. En cada caso se comprueban los mensajes de error y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Intentar añadir una nueva página con datos incorrectos.<br>▪ Añadir una nueva página.<br>▪ Intentar editar la página creada con datos incorrectos.<br>▪ Eliminar la página creada. |

TABLA 4.45: Caso de Prueba de Aceptación - Gestionar páginas - Datos inválidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar selectores - Datos válidos. |
| **Historia de Usuario:** | Gestionar selectores. |
| **Descripción:** | Se prueba que se realice de forma correcta la gestión de selectores, haciendo énfasis en las operaciones listar, añadir, editar, mostrar contenido y eliminar. En cada caso se comprueban los mensajes de notificación y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. Se debe crear una página y un filtro. |
| **Pasos de ejecución:** | ▪ Acceder al listado de selectores.<br>▪ Añadir un nuevo selector.<br>▪ Editar el selector creado.<br>▪ Mostrar contenido del selector creado.<br>▪ Eliminar el selector creado. |

TABLA 4.46: Caso de Prueba de Aceptación - Gestionar selectores - Datos válidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar selectores - Datos inválidos. |
| **Historia de Usuario:** | Gestionar selectores. |
| **Descripción:** | Se prueba que se validen de forma correcta los datos correspondientes a los selectores, haciendo énfasis en las operaciones añadir y editar. En cada caso se comprueban los mensajes de error y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. Se debe crear una página. |
| **Pasos de ejecución:** | ▪ Intentar añadir un nuevo selector con datos incorrectos.<br>▪ Añadir un nuevo selector.<br>▪ Intentar editar el selector creado con datos incorrectos.<br>▪ Eliminar el selector creado. |

TABLA 4.47: Caso de Prueba de Aceptación - Gestionar selectores - Datos inválidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar filtros - Datos válidos. |
| **Historia de Usuario:** | Gestionar filtros. |
| **Descripción:** | Se prueba que se realice de forma correcta la gestión de filtros, haciendo énfasis en las operaciones listar, añadir, editar y eliminar. En cada caso se comprueban los mensajes de notificación y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Acceder al listado de filtros.<br>▪ Añadir un nuevo filtro.<br>▪ Editar el filtro creado.<br>▪ Eliminar el filtro creado. |

TABLA 4.48: Caso de Prueba de Aceptación - Gestionar filtros - Datos válidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar filtros - Datos inválidos. |
| **Historia de Usuario:** | Gestionar filtros. |
| **Descripción:** | Se prueba que se validen de forma correcta los datos correspondientes a los filtros, haciendo énfasis en las operaciones añadir y editar. En cada caso se comprueban los mensajes de error y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Intentar añadir un nuevo filtro con datos incorrectos.<br>▪ Añadir un nuevo filtro.<br>▪ Intentar editar el filtro creado con datos incorrectos.<br>▪ Eliminar el filtro creado. |

TABLA 4.49: Caso de Prueba de Aceptación - Gestionar filtros - Datos inválidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar canales RSS - Datos válidos. |
| **Historia de Usuario:** | Gestionar canales RSS. |
| **Descripción:** | Se prueba que se realice de forma correcta la gestión de canales RSS, haciendo énfasis en las operaciones listar, añadir, editar y eliminar. En cada caso se comprueban los mensajes de notificación y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Acceder al listado de canales RSS.<br>▪ Añadir un nuevo canal RSS.<br>▪ Editar el canal RSS creado.<br>▪ Eliminar el canal RSS creado. |

TABLA 4.50: Caso de Prueba de Aceptación - Gestionar canales RSS - Datos válidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar canales RSS - Datos inválidos. |
| **Historia de Usuario:** | Gestionar canales RSS. |
| **Descripción:** | Se prueba que se validen de forma correcta los datos correspondientes a los canales RSS, haciendo énfasis en las operaciones añadir y editar. En cada caso se comprueban los mensajes de error y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Intentar añadir un nuevo canal RSS con datos incorrectos.<br>▪ Añadir un nuevo canal RSS.<br>▪ Intentar editar el canal RSS creado con datos incorrectos.<br>▪ Eliminar el canal RSS creado. |

TABLA 4.51: Caso de Prueba de Aceptación - Gestionar canales RSS - Datos inválidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar boletines - Datos válidos. |
| **Historia de Usuario:** | Gestionar boletines. |
| **Descripción:** | Se prueba que se realice de forma correcta la gestión de boletines, haciendo énfasis en las operaciones listar, añadir, editar y eliminar. En cada caso se comprueban los mensajes de notificación y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Acceder al listado de boletines.<br>▪ Añadir un nuevo boletín.<br>▪ Editar el boletín creado.<br>▪ Eliminar el boletín creado. |

TABLA 4.52: Caso de Prueba de Aceptación - Gestionar boletines - Datos válidos.

| Caso de Prueba de Aceptación | |
|---|---|
| **Nombre:** | Gestionar boletines - Datos inválidos. |
| **Historia de Usuario:** | Gestionar boletines. |
| **Descripción:** | Se prueba que se validen de forma correcta los datos correspondientes a los boletines, haciendo énfasis en las operaciones añadir y editar. En cada caso se comprueban los mensajes de error y los elementos del escenario correspondiente. |
| **Precondiciones:** | Un usuario debe estar autenticado. Se deben proveer credenciales correctas. |
| **Pasos de ejecución:** | ▪ Intentar añadir un nuevo boletín con datos incorrectos.<br>▪ Añadir un nuevo boletín.<br>▪ Intentar editar el boletín creado con datos incorrectos.<br>▪ Eliminar el boletín creado. |

TABLA 4.53: Caso de Prueba de Aceptación - Gestionar boletines - Datos inválidos.