Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Herramienta de gestión del conocimiento para programadores

Autor: Elisa Yelmo Alba

Tutor: Antonio Estepa Alonso

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Herramienta de gestión del conocimiento para programadores

Autor:
Elisa Yelmo Alba

Tutor:
Antonio Estepa Alonso
Profesor titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Proyecto Fin de Carrera: Herramienta de gestión del conocimiento para programadores

Autor: Elisa Yelmo Alba

Tutor: Antonio Estepa Alonso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi abuela

Agradecimientos

En primer lugar, agradecer a mis padres y a mi hermano todo su apoyo, nunca podré pagarles todo su esfuerzo para que pueda hoy escribir estas líneas.

A mi abuela, por preocuparse, alegrarse, escuchar a pesar de no entender nada, por confiar en mí, por contarme sus historias y llamarme (casi) cada domingo.

A las mejores personas que nunca podré conocer, a mis amigos, a mis compañeros, a todos los que pasaron por mi vida y de alguna forma pusieron su granito de arena. A Alberto por apoyarme y soportarme en mis malos momentos.

A todos los profesores que han hecho de los conocimientos adquiridos una pasión y, en especial, a mi tutor Antonio por orientarme en este proyecto.

Al grupo de Ayesa Advanced Technologies por su colaboración y, sobretodo, a Javi y Miguel Ángel por darme la oportunidad.

A todos, gracias.

Elisa Yelmo Alba

Sevilla, 2017

Resumen

En los últimos años la programación ha adquirido un papel crucial en la vida diaria, siendo así, uno de los pilares fundamentales de muchas empresas. Cuando un desarrollador genera código, está generando información y conocimiento que será utilizado en un futuro a corto o medio plazo por otros desarrolladores.

El objetivo de este proyecto es facilitar el proceso de búsqueda de información a los programadores de una empresa a partir de la integración de una Wiki (a nivel de código) en el Entorno de Desarrollo Integrado Eclipse. Para ello, se ha desarrollado una Plug-in en Java con el cual se pretende ofrecer un acceso rápido a una información de calidad.

Cada trabajador puede encontrar un botón en su IDE con el cuál acceder a la Wiki, una vez pulsado, se agilizará la búsqueda eligiendo una serie de opciones y, según la decisión tomada, aparecerá una plantilla del código correspondiente en el editor de texto.

Además, se han diseñado una serie de indicadores para asegurar que la información almacenada es de calidad.

A través del uso de este sistema se espera que los equipos de desarrollo de diferentes organizaciones mejoren su productividad y calidad de trabajo.

Abstract

In recent years, programming has acquired a main role in daily life, thus, it is one of the fundamental pillars of many companies. When a developer makes code, he's making information and knowledge which will be used in the future short and medium term by other developers.

The aim of this project is to facilitate the process of finding information to the developers in a company with an integration between a Wiki and the Integrated Development Environment Eclipse. For this, a Plug-in in Java has been developing with which is intended to offer a quick access to quality information.

Each employee will be able to find a button on his IDE with which access to the Wiki, if he pushes, he will have to choose between different options and, according the decision taken, a template with the correct code will appear on the text editor.

In addition, some indicators have been designed to ensure that the stored information is of quality.

Through the use of our system we can hope that differents development teams will improve their productivity and quality of work.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

IDE	Entorno de Desarrollo Integrado, Eclipse
Swing	Java Swing para interfaces gráficas
SWT	Java SWT (Standard Widget Toolkit) para interfaces gráficas
RCP	Rich Client Platform
BD	Base de Datos
GUI	Interfaz Gráfica de Usuario
JR	JasperReports
TI	Tecnología de la información

1 INTRODUCCIÓN

Todo comienzo tiene su encanto.

- Goethe -

El origen de este proyecto tiene lugar tras un periodo de prácticas en la empresa Ayesa Advanced Technologies[1], después de un mes de beca en el proyecto "Séneca y Pasen"[2] existía la sensación de que cada vez que se necesitaba desarrollar algo nuevo, encontrar la información necesaria para hacerlo era un auténtico reto. El problema era generalizado, no solo por la inexperiencia de los becarios en el proyecto sino también por la falta de una documentación clara, sencilla de manejar y con ejemplos.

Por lo que, tras explicar nuestras inquietudes a Javi y Miguel Ángel, los jefes, nació la idea de este trabajo. Pero para entender mejor la necesidad pongámonos en situación.

1.1. Situación actual

"Séneca y Pasen" es un proyecto de desarrollo web (basado en Java, JSP y HTML) que lleva vigente desde el año 1999 y sigue en crecimiento. Con estos datos, no es difícil imaginar la cantidad de ficheros de código que podemos encontrar y la cantidad de desarrolladores que estaban y se fueron.

En principio, la enormidad no tiene por qué ser caótica. El problema radica al no tener una documentación clara y, además, esto se agrava si el proyecto se basa en una serie de librerías propias.

En resumen, nos encontramos un proyecto donde:

- **No tenemos una documentación clara**, con ejemplos, donde sea fácil buscar.
- El código se basa en **librerías propias** previamente desarrolladas, por lo tanto, hay una forma concreta de hacer las cosas.
- Los programadores buscan la información revisando los códigos ya diseñados y decir que, tenemos casi **2GB de ficheros de código**.
- **Hay ficheros desfasados**. Por ejemplo, para incluir una serie de pestañas en una página Web, actualmente se utiliza una etiqueta HTML llamada <pestannas>. Sin embargo, si buscas entre ficheros lo más probable es que encuentres estas pestañas desarrolladas a partir de código JSP. Un código mucho

[1] Empresa Sevilla liderada por José Luís Manzanares, antigua Sadiel.
[2] Proyecto de Gestión Educativa en la provincia de Andalucía, Madrid, Navarra y Extremadura.

más complejo y amplio.

1.1.1. Conclusiones de la situación actual

En definitiva, los desarrolladores invierten mucho tiempo tratando de encontrar la información que necesitan para llevar a cabo sus tareas. Por lo que, a nivel empresarial supone un gran coste en tiempo.

La búsqueda en ficheros antiguos puede proporcionar información de mala calidad, además de ser una práctica muy poco eficiente.

Por otro lado, la falta de una documentación dificulta la comprensión de la funcionalidad del código.

1.2. Solución propuesta

Con esta situación y tras muchas reuniones, ya eran muchos los empleados que hablaban de hacer una Wiki. Una Wiki [1] es un sistema informático que permite a los usuarios crear, modificar, gestionar y almacenar contenido de una forma sencilla y rápida. El objetivo de esto era permitir a los propios desarrolladores explicar y aportar el contenido que ellos mismos considerasen importante, creando así un entorno completamente actualizado.

A partir de esto, nació una segunda idea ¿Por qué no integrar la propia Wiki en Eclipse y que los desarrolladores consigan plantillas de código desde el propio IDE[3]. Nadie sabía hasta qué punto era esto posible, pero sin duda era una idea muy atractiva y merecía la pena intentarlo.

1.2.1. Requisitos

Por un lado, la herramienta de integración que buscamos debe cumplir las siguientes especificaciones:

- Debe tener una interfaz gráfica con la que el usuario interactúe.
- Dentro del IDE el desarrollador tiene que tener acceso a todas las páginas con código de la Wiki y generará una plantilla de código cuando así se solicite.
- Una de las funcionalidades de la herramienta será la descarga de ejercicios. Cuando un nuevo becario se incorpora al equipo pasa sus primeras semanas haciendo 6 ejercicios de introducción para familiarizarse con el framework[4] y la aplicación. Actualmente, los ejercicios pasan de compañero en compañero, por lo que, queremos automatizar esta tarea.
- La herramienta tiene que estar completamente automatizada, sin necesidad de rellenar en un futuro campos de listas desplegables, opciones…
- La herramienta debe ser código abierto.

Por otro lado, la Wiki debe:

- Tener una estructura conocida
- Una página principal de buenas prácticas en la Wiki donde se expliquen las normas para el correcto uso de la misma y el correcto funcionamiento de la herramienta.

1.2.2. Tecnología utilizada

Para realizar este Proyecto se han utilizado distintos softwares y tecnologías. Este aparatado tiene como objetivo esquematizar todas y cada una de ellas para poner en situación al lector.

[3] Abreviatura en inglés de "Entorno de Desarrollo Integrado", Eclipse
[4] Entorno de trabajo para la organización y desarrollo de software

Tabla 1 – Softwares utilizados

Icono	Descripción
 Fig 1 – Logo MediaWiki	**MediaWiki** [2] **:** Software wiki de código libre escrito en PHP originalmente para su uso en Wikipedia. Actualmente es utilizado también por varios proyectos más de la Fundación Wikimedia sin ánimo de lucro y por muchos otros wikis, incluyendo este sitio web, el hogar de MediaWiki.
 Fig 2 – Logo XAMPP	**XAMPP** [3]**:** XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.
 Fig 3 – Logo phpMyAdmin	**PhpMyAdmin** [4]**:** PhpMyAdmin es una herramienta de software libre escrita en PHP , destinada a manejar la administración de MySQL a través de la Web. PhpMyAdmin soporta una amplia gama de operaciones en MySQL y MariaDB.
 Fig 4 – Logo Eclipse	**Eclipse** [5]**:** Eclipse proporciona IDEs y plataformas para casi todos los idiomas y arquitecturas. Famoso por sus IDE, C/C++, JavaScript y PHP construidos sobre plataformas extensibles para crear IDEs de escritorio, Web y la nube. Estas plataformas ofrecen la más amplia colección de herramientas complementarias disponibles para los desarrolladores de software.
 Fig 5 – Logo JasperReports	**JasperReport** [6]**:** La Biblioteca JasperReports es el motor de generación de informes de código abierto más popular del mundo. Está completamente escrito en Java y es capaz de utilizar datos procedentes de cualquier tipo de base de datos y producir documentos perfectos en píxeles que se pueden ver, imprimir o exportar en una variedad de formatos de documentos, incluyendo HTML, PDF, Excel, OpenOffice y Work.

Tabla 2 – Lenguajes de programación

Icono	Descripción
 Fig 6 – Logo Java SWT	**Java SWT** [5]**:** **SWT** (siglas en inglés de *Standard Widget Toolkit*) es un conjunto de componentes para construir interfaces gráficas en Java, (widgets) desarrollados por el proyecto Eclipse.

1.2.3. Límite

Puesto que es un proyecto de gran alcance. Este trabajo se presenta como una primera versión cumpliendo todos los requisitos antes expuestos.

1.2.4. Arquitectura

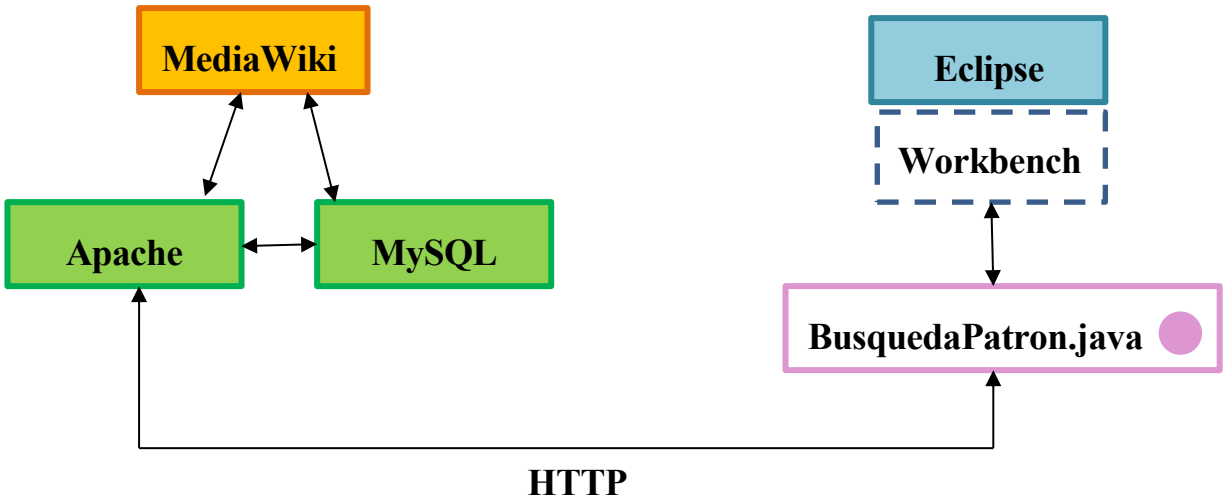


Fig 7 – Arquitectura de la herramienta

2 ALTERNATIVAS PARA EL DESARROLLO DE LA HERRAMIENTA

Si supiese lo que estoy haciendo, no lo llamaría investigación, ¿Verdad?

- Albert Einstein -

La primera parte de este proyecto se basa en un largo periodo de investigación. Primero, descubrir si era una idea posible y, segundo, descubrir cómo y qué herramientas tenía a mi disposición para llevarlo a cabo.

Tras muchas horas, cabezazos y tirar el proyecto a la basura en varias ocasiones, se encontró todo lo que hacía falta para dar vida a este trabajo y hacer una integración de la Wiki en Eclipse rápida y automatizada.

En los siguientes puntos se hará un estudio de mercado entre las distintas tecnologías que podemos encontrar para desarrollar este proyecto y el porqué de nuestra decisión.

2.1. Softwares de Wiki

Como se comentó en el capítulo anterior, una Wiki es un sistema informático que permite crear y modificar contenido a sus usuarios de una forma muy sencilla. Para funcionar, utiliza un software o motor de Wiki y actualmente en el mercado podemos encontrar gran variedad.

A continuación, se listan los 10 mejores softwares [7] :

- MediaWiki
- JSPWiki
- MoinMoin
- PmWiki
- TikiWiki
- Wacko Wiki
- DakoWiki
- XWiki
- VQWiki
- WikkaWiki

A pesar de las múltiples opciones estaba especialmente interesada en dos de ellos, **MoinMoin** y **MediaWiki.**

MoinMoin es el motor que usan Wikis como la de Python[5] o el equipo Ubuntu[6], mientras que MediaWiki es el software que da vida a Wikipedia[7]. Por lo que, ambos motores debían tener mucho que ofrecer.

2.1.1 MoinMoin VS MediaWiki

Tabla 3 – MoinMoin VS MediaWiki

MoinMoin	MediaWiki
Software basado en Python	Software basado en PHP
Código libre	Código libre
Disponible para: Unix, Windows, MacOS X	Disponible para: Unix, Windows, MacOS X
Fácil e intuitiva de manejar	Uso más complejo
Servidor Web: Apache+mod_wsgi	Apache
No utiliza base de datos, guarda la información en tablas del sistema de archivos.	Utiliza una base de datos (MySQL)
Pensado para Wikis pequeñas	Pensado para Wikis grandes
Cuenta con un editor completo, con múltiples opciones	Editor de texto muy básico (se recomienda intalar una extensión para su mejora)
Permisos de edición	Sin permisos de edición
-	Interfaz multilenguaje

Con esta idea general, la primera opción durante cerca de un mes fue MoinMoin. En principio, yo no quería una Wiki muy grande y MoinMoin ofrecía una interfaz muy intuitiva y sobretodo muy fácil de editar. Por lo que sería más sencillo para el usuario final añadir contenido. Además, en principio, parecía que la base de datos ofrecería más desventajas que ventajas. A día de hoy, todo el sistema se encuentra en mi ordenador personal, pero pensando en el futuro, todo se instalará en otro equipo por lo que migrar esa base de datos sería un proceso engorroso.

Sin embargo, finalmente se decidió cambiar de motor de Wiki y trabajar con MediaWiki por la base de datos que al principio no queríamos. Es cierto que este software está pensado para páginas con mucho contenido y tráfico, pero la base de datos nos ayudaría para automatizar la herramienta hasta el punto de que funcionase con cualquier Wiki que la compañía quisiese desarrollar. También nos aportaba información de gran valor para estudiar la rentabilidad de este proyecto.

2.2. Eclipse

Una vez que estaba claro cómo íbamos a desarrollar la Wiki pasamos al lado de Eclipse. Para este punto

[5] Lenguaje de programación
[6] Sistema Operativo basado en GNU/Linux
[7] Enciclopedia libre, políglota y editada colaborativamente.

utilizaríamos la última versión Eclipse Jee Neon ya que es el utilizado por los desarrolladores de Ayesa Advanced Technologies. Sin embargo ¿Cómo se podía hacer la integración?

En tercero de telemática tenemos una asignatura llamada "Ingeniería de Software", en esta asignatura trabajamos con eclipse para familiarizarnos con el control de versiones a través de GitHub[8], aplicación que está perfectamente integrada a través de un Plug-in, por lo que pensé que si desarrollaba mi propio Plug-in podría conseguir mi objetivo.

2.2.1. Plug-in

A través de un plugin podemos personalizar Eclipse según nuestras necesidades. Para llevarlo a cabo es necesario previamente hacer un estudio teórico y tener clara la arquitectura del mismo. Esto será explicado en el capítulo 3 "Eclipse: Arquitectura y extensión".

2.3. Java

Cuando se desarrolla un plugin en Eclipse, como veremos más adelante, se crea un proyecto Java. La idea principal era una aplicación con la que el usuario pudiese interactuar con el sistema, sin embargo, durante estos cuatro años de carrera nunca hemos visto el desarrollo de interfaces gráficas en ningún lenguaje de programación. Por lo que este punto de investigación lo dividiremos en dos categorías, por un lado, a estudiar las distintas opciones que tenemos en el mercado para desarrollar interfaces gráficas con Java, las ventajas de cada una de ellas y nuestra decisión final. Por otro lado, estudiaremos como controlar el propio IDE Eclipse.

2.3.1. Interfaz gráfica

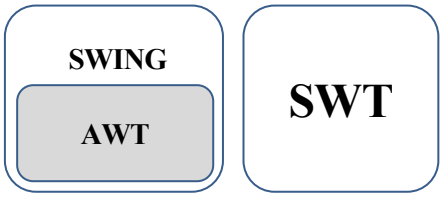


Fig 8 – Clasificación Java para Interfaces gráficas

2.3.1.1. Java Swing [8]

Java Swing puede considerarse la evolución de Java AWT[9], los cuales son un conjunto de librerías enfocadas a la construcción de interfaces gráficas. La mejora que propone Swing frente a AWT, además de en el aspecto visual, es una mejora de portabilidad y comportamiento entre los distintos sistemas operativos. Aquí, es importante destacar que Swing no se adaptaría al SO siendo una aplicación puramente Java.

[8] Plataforma que aloja proyectos a través del control de versiones de Git.
[9] AWT del inglés "Abstract Window Toolkit"

El siguiente ejemplo muestra la jerarquía de Swing con alguno de los principales componentes:

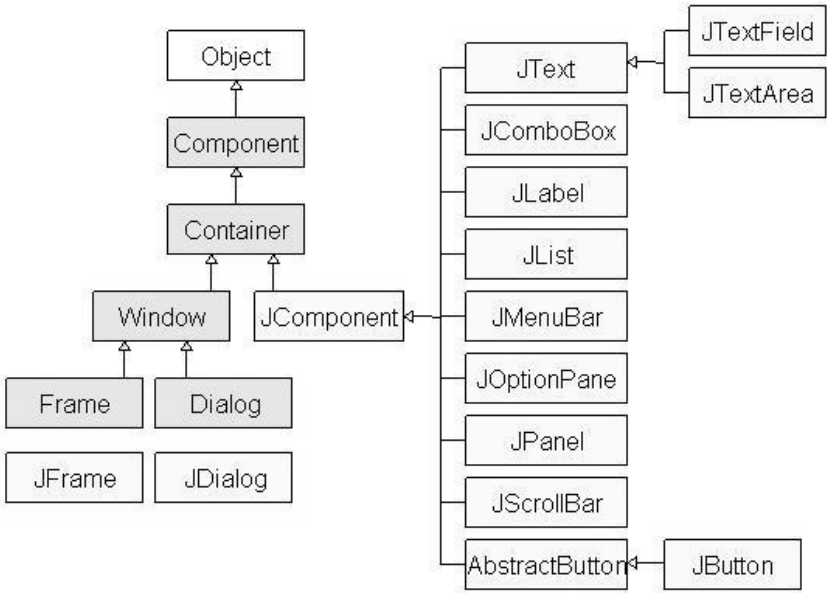


Fig 9 – Jerarquía Java Swing

Sin embargo, tras probar a hacer una ventana emergente muy simple en Eclipse me di cuenta de que la apertura de la misma era un proceso extremadamente lento, lo que, personalmente, no me gustó y me llevó a descubrir el siguiente punto, Java SWT.

2.3.1.2. Java SWT [9]

SWT viene de las siglas en inglés "Standard Widget Toolkit" y consiste en un conjunto de componentes para desarrollar interfaces en java en Eclipse. Estas librerías fueron desarrolladas por los propios desarrolladores del IDE reciclando las mejores cosas de Java AWT, a su vez, de Java Swing y añadiendo una nueva capa llamada JFace.

Así, Java SWT recupera la idea original de utilizar y basarse en componentes nativos de Java AWT que se había perdido con Swing, lo que, no solo hace una aplicación más rápida si no también hace que los programas se sientan como aplicaciones nativas adaptándose al sistema operativo, por ejemplo, una ventana emergente no es igual en Windows que en Linux.

Para conseguir esta adaptación al sistema operativo SWT utiliza los widgets del mismo. En la siguiente URL tenemos una lista completa: http://www.eclipse.org/swt/widgets/

Todas las librerías de SWT comienzan con el prefijo org.eclipse.swt.

2.3.1.2.1. JFace

La interfaz del workbench[10] en Eclipse depende de una capa intermedia llamada JFace que nos ayuda a simplificar la construcción de elementos mediante el uso de Java SWT.

Las clases que podemos encontrar son las siguientes:

- Viewers: encargados de buscar, filtrar y actualizar diferentes elementos.
- Actions, contributions: para programar acciones de usuario.
- Image, Font: imágenes y fuentes en la interfaz gráfica.
- Dialogs, Wizard: para interacciones avanzadas con el usuario.

[10] Workbench: Ventana completa de Eclipse, ver capítulo 3

En este caso todas las librerías comienzan por el prefijo org.eclipse.jface.

2.3.1.3. Decisión final

Para ver más claras las diferencias entre Java Swing y Java SWT vamos a hacer una tabla comparativa entre ambos:

Java Swing	Java SWT
Puede desarrollar aplicaciones gráficas avanzadas.	No puede desarrollar aplicaciones gráficas avanzadas.
Puede programarse en cualquier IDE.	Solo puede programarse en Eclipse
Hay mucha documentación	Hay muy poca documentación
Menor velocidad	Mayor velocidad

Tabla 4 – Java Swing vs Java SWT

La información sobre cómo programar con Java SWT es muy escasa, eclipse tiene su propia Wiki de documentación. Sin embargo, no cuenta con ejemplos ni tenemos una arquitectura clara de todos los objetos. Lo que dificulta el trabajo.

Por otro lado, tener una aplicación rápida era mi prioridad, por lo que, aunque la programación con Java Swing era más sencilla sin duda me decanté por utilizar SWT.

2.3.2. org.eclipse

La API[11] de eclipse cuenta con sus propias librerías para controlar el IDE, todas ellas comienzan con el prefijo org.eclipse y están basadas en Java.

Aunque hablemos de librerías, decir que, realmente son paquetes de plugins con el mismo nombre. A lo largo de este proyecto y especialmente en el capítulo de integración, se irán explicando las distintas librerías utilizadas importadas.

Todas las librerías pueden encontrarse en la Wiki de la API de Eclipse. [10]

[11] API del inglés Application Public Interface

3 ECLIPSE: ARQUITECTURA Y EXTENSIÓN

Los que se enamoran de la práctica sin la teoría son como los pilotos sin timón ni brújula, que nunca podrán saber dónde van

- Leonardo Da Vinci -

Nuestro objetivo es integrar una Wiki en Eclipse, de modo que a partir del propio IDE podemos acceder a la información almacenada en la Wiki y generar plantillas de código. En este capítulo queremos explicar cómo funciona Eclipse, cómo se desarrolla un plugin y la definición de algunos conceptos que necesitamos para dar vida a este proyecto.

3.1. Eclipse

El núcleo de la plataforma Eclipse es uno de los puntos más complejos de este trabajo, el primer concepto que tenemos que tener claro es que **Eclipse es un conjunto de plugins perfectamente integrados**. (org.eclipse.ui, org.eclipse.core…) Donde cada plugin cuenta con uno o varios puntos de extensión[12] que permiten a otros plugin interactuar con ellos, mejorar funcionalidades o desarrollarlas nuevas.

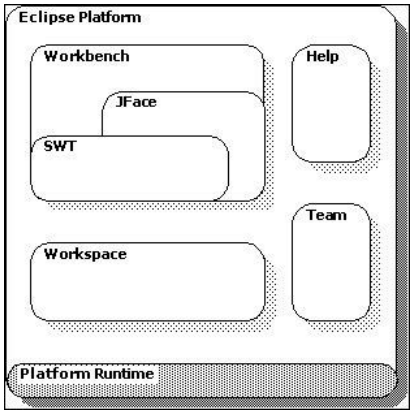
Gracias a esto podemos llevar a cabo nuestro trabajo, la idea es crear un plugin que, por un lado, añada una nueva funcionalidad en la barra de herramientas, es decir, un icono con el que acceder a la Wiki y, por otro lado, modifique el comportamiento del editor de texto.

Para encontrar los plugins que controlan las zonas que necesitamos para el desarrollo de este proyecto en primer lugar tenemos que entender cómo es la arquitectura modular de Eclipse. [10]

3.1.1. Estructura de Eclipse

[5] A partir de la versión 2.0 de Eclipse se decide implementar una plataforma estructurada en subsistemas implementados en uno o más plugins donde cada subsistema es creado por una especie de "motor de ejecución".

[12] Punto de extensión: "puertas" que abre un plugin hacia una funcionalidad propia y permiten a otro plugin extender o modificar esa funcionalidad.



Subsistemas:

- Workbench
- Workspace
- Help
- Team

"Motor de ejecución", platform runtime

Fig 10 – Plataforma Eclipse

3.1.1.1 Workbench

Ventana completa de la plataforma. En el workbench tenemos definidas todas las herramientas para viajar por Eclipse (menús, barra de herramientas…). Además, implementa un conjunto de puntos de extensión para añadir componentes como vistas, editores, acciones en el menú. [fig 9].

Por otra parte, cuenta con las bibliotecas gráficas de java **SWT** y java **JFace**.

3.1.1.2 Workspace

Subsistema que se encarga de la creación y el correcto uso de los proyectos, carpetas, ficheros que fueron creados por otras herramientas del workbench y se mantienen en un sistema de archivos.

3.1.2. Rich Client Platform

Cada subsistema de Eclipse está basado en aplicaciones RCP[13], por lo que Eclipse en su conjunto es una aplicación RCP.

RCP es una plataforma que nos permite crear aplicaciones Java con una UI[14] muy diferente a las tradicionalmente basadas en Java Swing. Esto se da gracias al uso de SWT. Concepto que encaja a la perfección con nuestro IDE e indica que la plataforma Eclipse se puede tener como base para crear aplicaciones ricas en funciones.

Por lo tanto, Eclipse es una aplicación Rich Client Platform que puede albergar otras aplicaciones RCP. En el siguiente punto veremos cómo es la arquitectura de este tipo de aplicaciones para entender cómo funcionará nuestra futura aplicación.

[13] Siglas en inglés Rich Client Platform
[14] Interfaz de Usuario

3.1.2.1. Arquitectura RCP

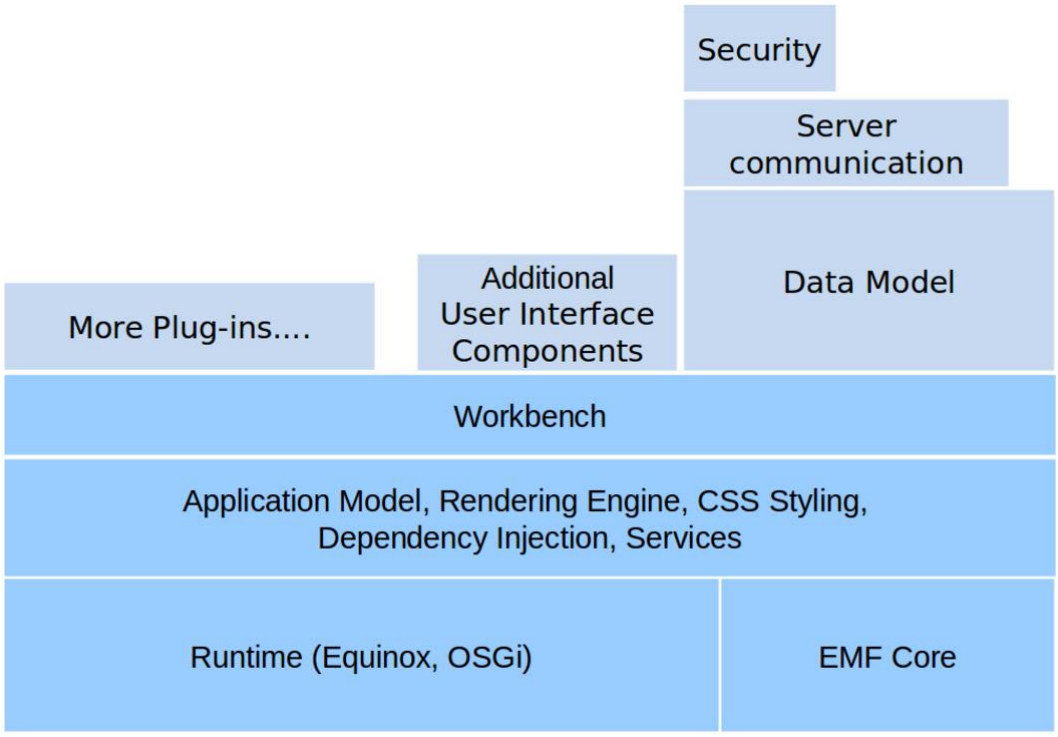


Fig 11 – Arquitectura de una aplicación RCP

Los puntos más importantes son: [9]

- **EMF Core:** Eclipse Modeling Framework proporciona una funcionalidad para crear un modelo de datos y utilizar este modelo de datos en el momento de ejecución.
- **Runtime:** Motor de ejecución de la aplicación.
    - **Equinox:** tiempo de ejecución
    - **OSGi:** especificación que describe un enfoque modular de las aplicaciones RCP.
- **App Model, Rendering Engine…:** Esta capa incluye todas las bibliotecas para crear la interfaz de usuario.
- **Workbench:** marco de la aplicación. "Ventana" que muestra todos los componentes gráficos de la herramienta.

3.1.3. Componentes del workbench en Eclipse

De entre todos los subsistemas el workbench es el que más nos interesa. Eclipse es un gran sistema formado por múltiples componentes definidos en el plugin **org.eclipse.ui**, plugin del que cuelga el workbench y, a su vez, todos sus componentes.

En la siguiente imagen mostramos gráficamente los componentes del workbench y pasaremos a explicar aquellos que más nos interesen. [5]

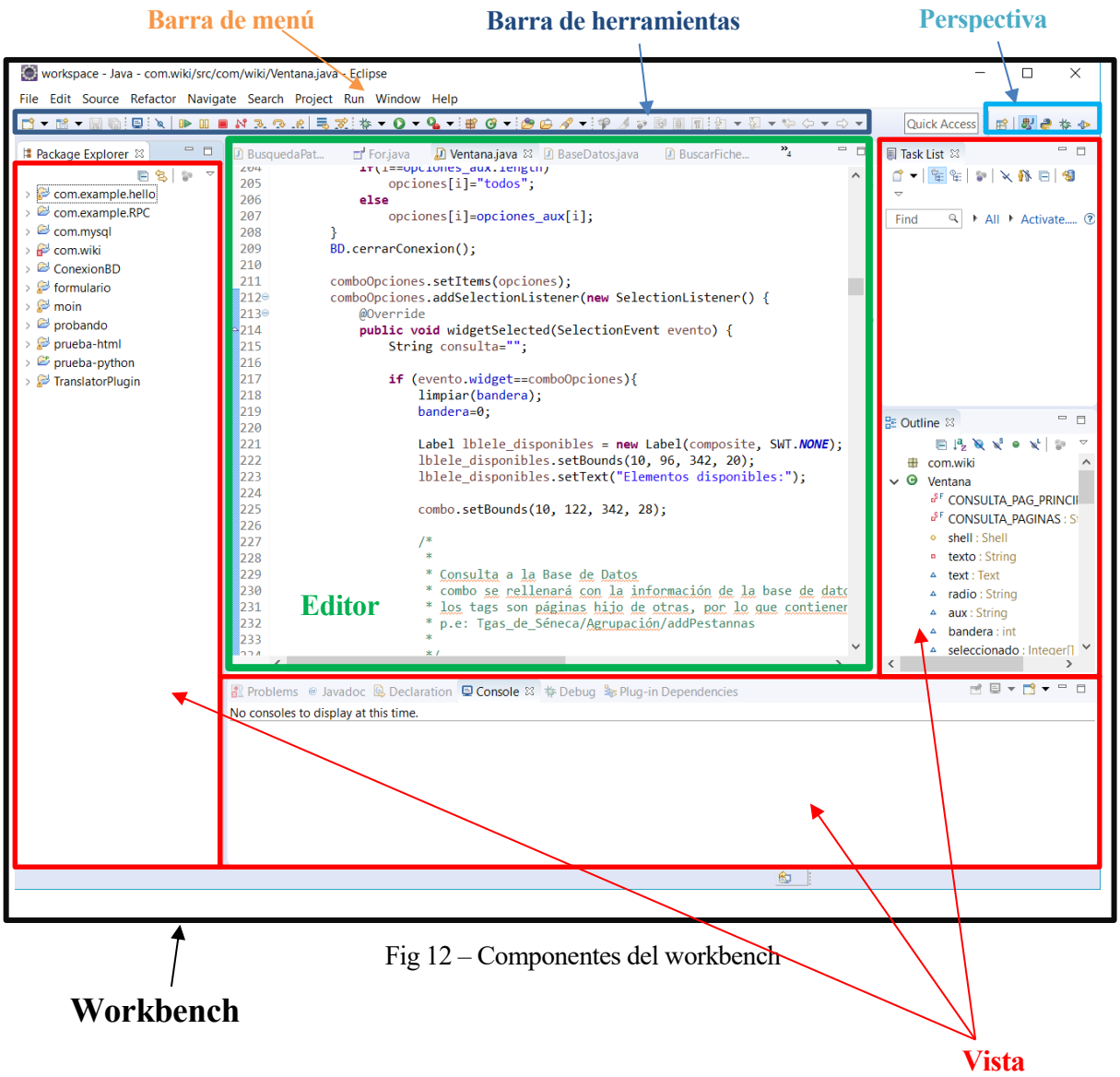


Fig 12 – Componentes del workbench

Vista: La vista son ventanas que nos muestran información de forma gráfica y en modo lectura.

Por supuesto, un desarrollador tiene libertad para modificar sus propias vistas a partir del plugin org.eclipse.ui.views.

Perspectiva: La perspectiva es una colección de vistas y editores controlada por el plugin org.eclipse.ui.perspectives.

3.1.3.1. Editor

El Editor es el módulo de Eclipse que más nos interesa, a grandes rasgos es muy parecido a una vista con la particularidad de que puede escribirse e ella.

Un Editor es un punto de extensión del plugin org.eclipse.ui.editors el cual, principalmente, implementa una clase llamada IEditorsPart, la cual nos da las funcionalidades necesarias para crear un editor de texto.

Sin embargo, la clave de nuestro proyecto se encuentra en la clase IDocument.

3.1.3.1.1. MVC

La clase IDocument podría entenderse como la entrada al editor activo en el momento de ejecución, es decir,

gracias a esta clase podríamos obtener el "documento" literal de nuestro editor de texto y escribir en él.

A su vez, este documento, junto con sus características de edición, es presentado por la clase SourceView, por ejemplo, esta clase se encargaría del autocompletado de las palabras, de marcar las palabras con el mismo nombre, sugerir opciones cuando pulsas CTRL+Espacio… y por último el estilo vendría marcado por la clase StyledText del paquete SWT.

Es fácil intuir que todos estos componentes están estructurados en un Modelo-Vista-Controlador (MVC) donde IDocument sería el modelo, SourceView el controlador y StyledText la vista.

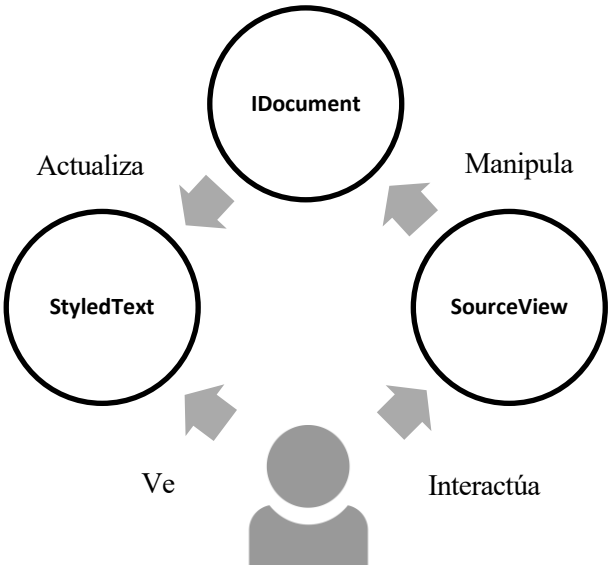


Fig 13 – Modelo-Vista-Controlador

3.1.4. Resumen

En definitiva, Eclipse tiene una arquitectura modular o de subsistemas donde cada módulo está formado por uno o más plugins que cuenta con uno o varios puntos de extensión que permiten a otros plugins modificar el comportamiento por defecto de los mismos o añadir nuevas funcionalidades.

Dentro de los módulos de Eclipse, el que más nos interesa es el Workbench ya que es el subsistema que cuenta con las barras de herramientas, menús, el editor de texto… y es definido por el plugin org.eclipse.ui

El editor de texto lo controlaremos con nuestra aplicación a partir de las clases IDocument, SourceView y StyledText del plugin org.eclipse.ui.editors.

3.2. Plugins

Los plugins en Eclipse están diseñados para acoplarse al entorno de trabajo o para acoplarse a otros plugins. Esto se logra mediante extensiones o puntos de extensión.

3.2.1. Extensiones

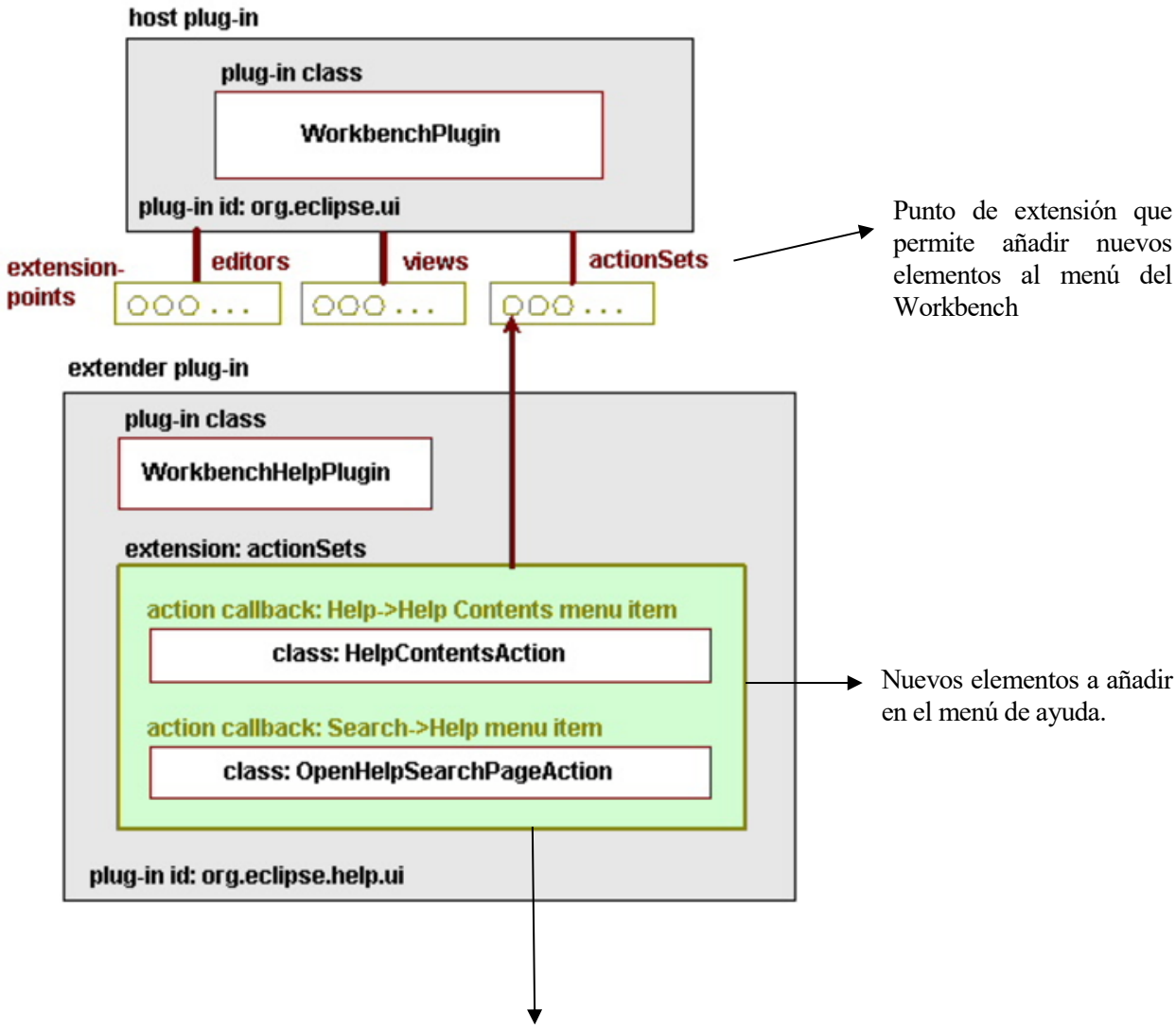
Las extensiones [11] son funcionalidades que los plugins nos ofrecen para que utilicemos en nuestro propio plugin. Como si importásemos una funcionalidad completa a nuestro plugin.

3.2.2. Puntos de extension

Un punto de extensión [11] es la funcionalidad que definimos para que otros puedan modificarla o mejorarla, pero sin llegar a importarla.

A continuación, ilustramos un ejemplo de extensión aportado por el grupo Eclipse [5] que será útil para nuestro proyecto.

En la Fig 11 podemos ver la relación entre los participantes de una extensión, en este caso la extensión del workbench de Eclipse, org.eclipse.ui, por el sistema de ayuda de Eclipse (subsistema help). Para ello, se pretende añadir nuevos elementos al menú de ayuda de Eclipse a partir de un punto de extensión conocido como actionSet. El complemento de extensión es la interfaz de usuario que ofrece Eclipse, org.eclipse.help.ui. que a partir de las clases HelpContentsAction y OpenHelpSearchPageAction añadirán una nueva funcionalidad en el menú de ayuda.



Extensión con la funcionalidad añadida en las clases HelpContentsAction y OpenHelpSearchPageAction

4. WIKI

El conocimiento si no se sabe explicar es peor que la ignorancia.

- Charles Bukowski -

Una de las partes de este proyecto es el desarrollo de una Wiki, un sistema informático que permite crear y modificar contenido a sus usuarios.

En concreto, para este proyecto se pretende crear una plataforma donde almacenar estructuras de código e información relativa al proyecto Séneca[15] y así mejorar el proceso de búsqueda de los programadores.

4.1. Instalación

4.1.1. MediaWiki

Como se comentó en otros capítulos, para implementar la Wiki se ha utilizado el software MediaWiki. Dicha instalación se encuentra explicada en el Anexo A.

4.2. Uso

MediaWiki es un software muy sencillo, en este punto explicaremos muy brevemente cómo se utiliza, edita y modifica el contenido.

Por otra parte, para velar por el correcto funcionamiento de la integración de la Wiki con Eclipse existen unas buenas prácticas de edición en la Wiki en el Anexo B.

4.2.1. Crear una página

Para crear una página nueva ponemos en el buscador el nombre que queremos que tenga.

[15] Séneca: proyecto de Ayesa Advanced Technologies para la gestión educativa de Andalucía.

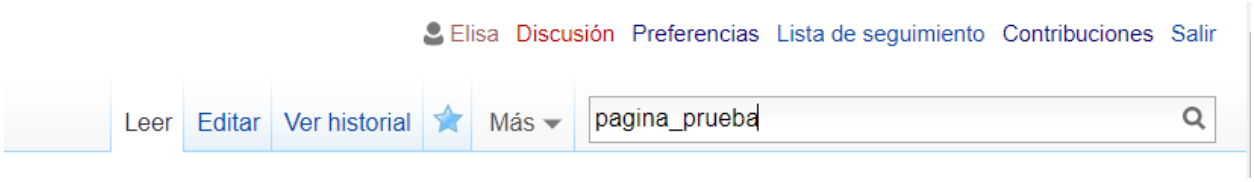


Fig 14 – Creación de una página nueva I

Cuando buscamos, nos aparecerá una página como la mostrada en la siguiente figura, donde nos ofrecerá crear

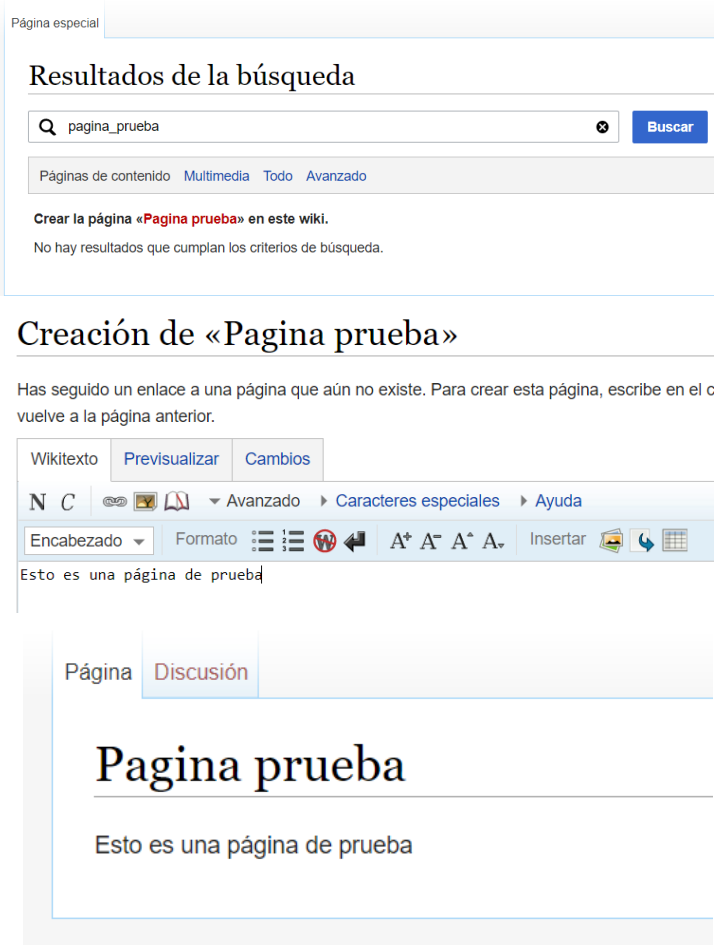


Fig 15 – Creación de una página nueva II

la página en caso de no existir y nos llevará directamente a un editor para escribir y publicar el nuevo contenido.

4.2.2. Subpáginas

Página_principal/subpágina/subsubpágina

[12] Las barras inclinadas (/) dentro de un nombre de página dividen la página entre padre y subpáginas, recursivamente. Por ejemplo, si tenemos una página padre llamada "Tags de séneca" y queremos crear una nueva página hijo (subpágina) llamada "Agrupación" tenemos que crear una nueva página con el nombre "Tags de Séneca/Agrupación". De la misma forma, si queremos crear una subsubpágina llamada "formulario" habría que nombrar a esta nueva página como "Tags de Séneca/Agrupación/formulario" y así sucesivamente.

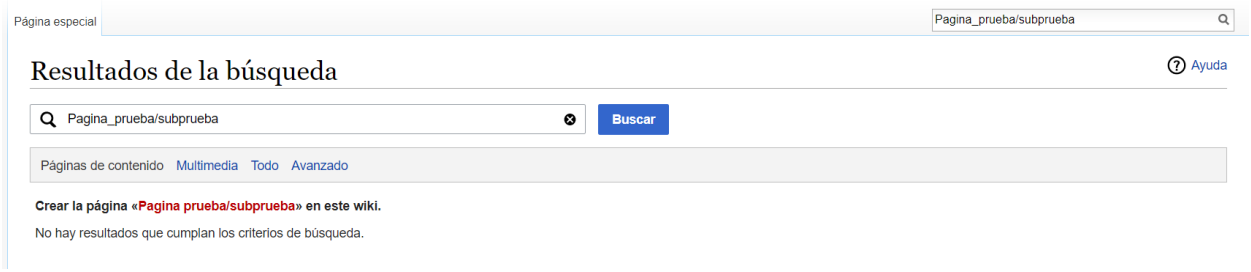


Fig 16 – Creación de una subpágina

4.2.3. Modificar Sidebar

Por defecto cuando iniciamos un nuevo espacio con MediaWiki tenemos un SideBar por defecto con el fin de ayudar al usuario en sus primeros pasos con la Wiki.

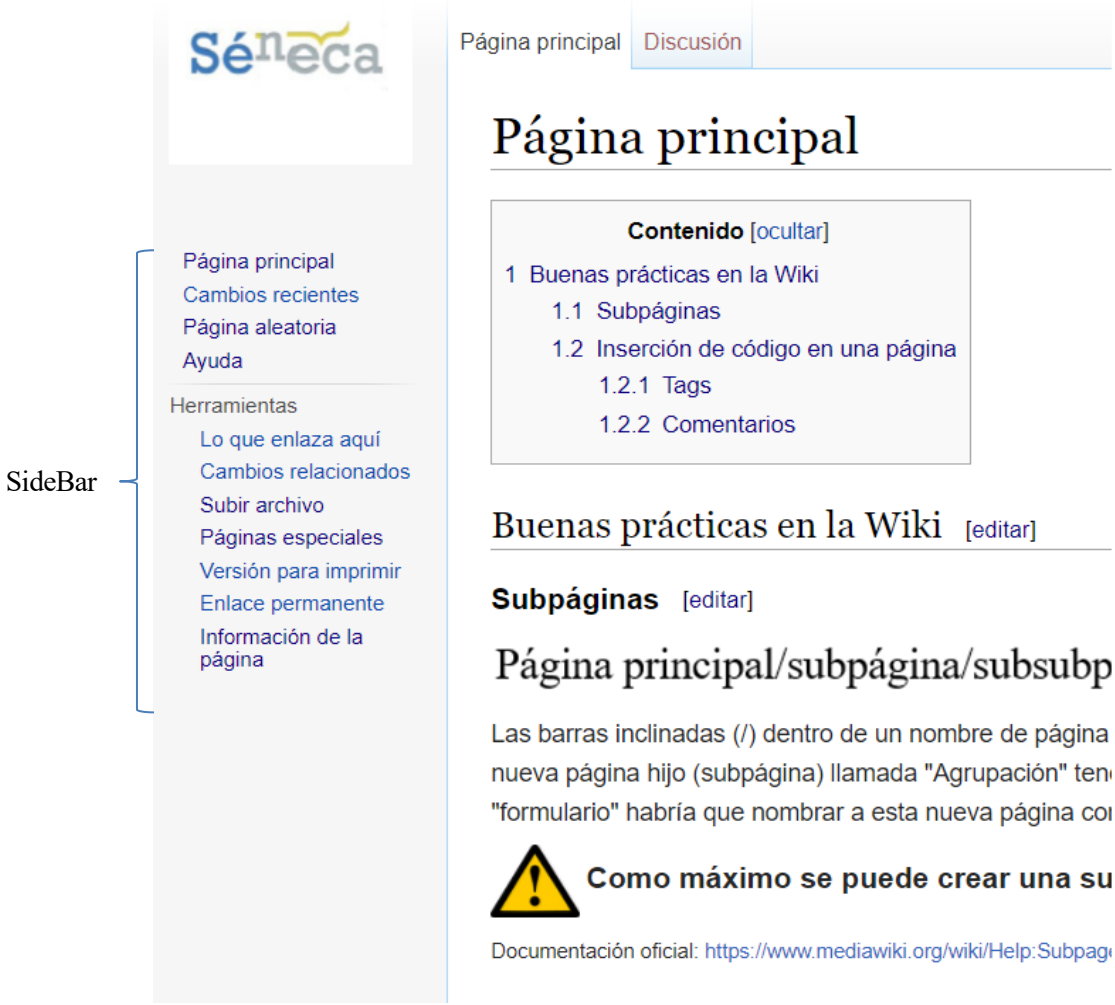


Fig 17 – SideBar de la Wiki

Para poder modificar esta barra lateral tenemos que acceder a la siguiente URL de la Wiki:

http://host_de_la_wiki /wiki/index.php?title=MediaWiki:Sidebar&action=edit

donde http://host_de_la_wiki /wiki/index.php es la dirección de la página principal de la Wiki. Una vez hemos accedido a la dirección anterior nos aparecerá una consola de edición donde podremos añadir y borrar elementos de la barra lateral.

4.3. Wikitexto

Toda Wiki tiene su propio lenguaje Wikitexto. Wikitexto [13] es un texto elaborado mediante lenguaje de marcado especial que fue diseñado para los distintos softwares de Wikis y utilizado para editar el contenido de la misma.

En el caso de MediaWiki podemos encontrar la siguiente sintaxis:

La sintaxis completa está en la siguiente referencia [14].

Formato	Significado
==Nivel 1==	Cabecera de primer nivel
===Nivel 2===	Cabecera de segundo nivel
====Nivel 3====	Cabecera de tercer nivel
''cursiva''	*cursiva*
'''negrita'''	**negrita**
***Entrada**	*Líneas en listas no numeradas
#Entrada	#Listas enumeradas
----	Línea horizontal
[[Título de nueva página]]	Enlace interno
[[Título | Otro texto para el título]]	Enlace interno con texto alternativo
http://www.ejemplo.com	Enlace externo (automáticamente se convierte en un enlace)
[http://www.ejemplo.com/ www.ejemplo.com]	Enlace externo con texto alternativo
[[Imagen:fichero.jpg]]	Inserta una imagen en la página
[[Imagen:fichero.jpg|thumb]]	Inserta un thumbnail en la página
{{Nombre}}	Integra dinámicamente el contenido de la página "Plantilla:Nombre"
{{subst:Nombre}}	Reemplaza el contenido de "Plantilla:Nombre" cuando almacena la página
{{:Nombre}}	Integra dinámicamente el contenido de la página "Nombre"
~~~	Crea una firma para el registro de usuario, esto es un enlace Wiki a una página personal del usuario
~~~~	Firma con la marca temporal

Fig 18 – Wikitexto

Afortunadamente, cuando creamos una página tenemos un editor que se encarga automáticamente de dar el formato correcto al texto que escribimos.

Cuando se lee la página en código HTML el código wikitexto aparece con etiquetas HTML.

Caracteres especiales [14]

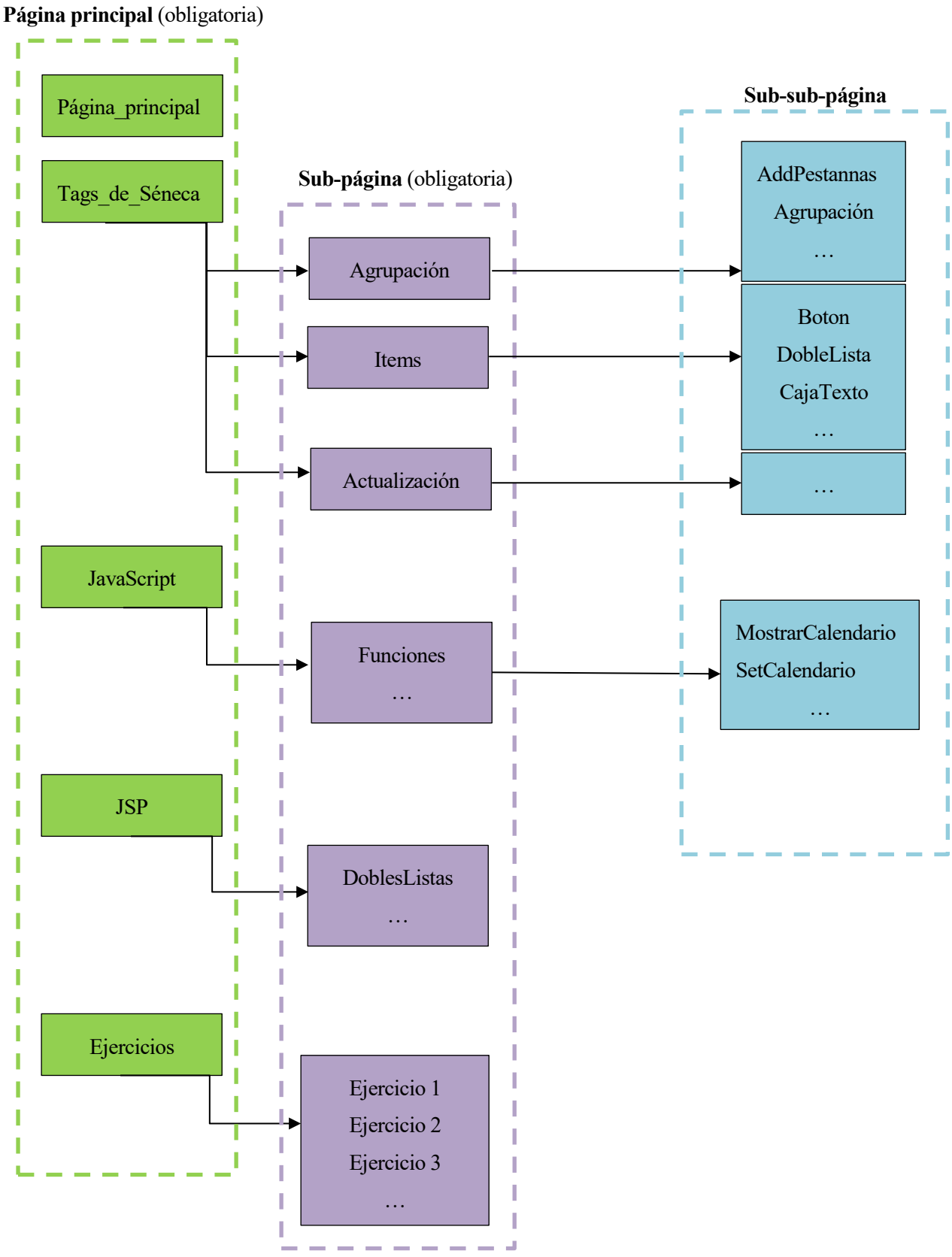
Hay que tener en cuenta una serie de caracteres que en wikitexto tienen un formato especial y al leerlos con html se mantiene, por lo que, a nivel de código habría que transformarlos para entender el texto.

Wikitexto	Significado
&amp	&
&ltj	<
&gtj	>
j	“

Tabla 5 - Wikitexto

4.4. Estructura de la Wiki

En este proyecto la Wiki tiene una estructura definida como veremos en el siguiente esquema:



5. GESTIÓN DE LA INFORMACIÓN

Toda información es importante si está conectada a otra.

- Umberto Eco -

En este capítulo se va a hacer un estudio a nivel estadístico del valor de la información de la Wiki. Con idea de que un futuro nuestras estadísticas ayuden a mejorar y estudiar la rentabilidad de la misma.

ITIL[16] es un marco de referencia que describe un conjunto de buenas prácticas y recomendaciones para la administración de servicios de la tecnología de la información. Dentro de esto marco podemos encontrar varios procesos, nosotros nos centraremos en la gestión del conocimiento.

5.1. Gestión del conocimiento

La gestión del conocimiento [15] de ITIL se encarga de recolectar, almacenar, analizar y compartir el conocimiento dentro de una organización.

ITIL define a dicho proceso como el único responsable de proporcionar conocimiento a todos y cada uno de los procesos de la gestión de servicios de TI[17].

Sistema de Gestión del Conocimiento del Servicio (SKMS)

El Sistema de Gestión del Conocimiento del Servicio (SKMS) es el repositorio central de los datos, información y conocimiento necesario para una organización. El objetivo es almacenar, analizar y presentar los datos al usuario. La gestión del conocimiento es responsable de mantener el servicio SKMS.

5.1.1. Etapas de la gestión del conocimiento

El conocimiento puede ser categorizado de acuerdo a la estructura datos-información-conocimiento-sabiduría (DIKW).

- Los datos representan hechos disponibles en la base de datos. Por sí mismos no tienen ningún valor. La gestión del conocimiento pretende capturar los datos, identificar los relevantes, mantener, archivar o eliminar cuando ya no sean necesarios.
- La información se genera cuando los datos son procesados. La gestión del conocimiento se encarga de gestionar el contenido de una manera que el usuario pueda consultar y analizarlo.

[16] ITIL: Information Technology Infrastructure Library
[17] TI: Tecnología de la información

- El conocimiento combina información con experiencia. El conocimiento puede utilizarse como base para tomar decisiones o llevar a cabo una acción,
- La sabiduría se crea a partir del conocimiento disponible, La gestión del conocimiento pone a disposición del usuario las herramientas necesarias para identificar esta asociación.

5.1.2. Fases de la gestión del conocimiento

Fig 19 – Fases Gestión del conocimiento

5.1.3. Beneficios de la gestión del conocimiento

Gracias a la gestión del conocimiento podemos:

- Disminuir la cantidad de tiempo de formación y búsqueda de los empleados de una organización. Dado que la información siempre estaría ubicada y actualizada en un repositorio.
- Reducir el número de errores de miembros del equipo.
- Responder a los problemas planteados en el trabajo de una forma más rápida y eficiente.

Así, nuestro proyecto gira en torno a este concepto, el cual, nos ayudará a mejorar la eficiencia mediante la reducción de la necesidad de redescubrir conocimiento. Para conseguir esto, por un lado, se ha creado una Wiki (SKMS) donde se almacena y presenta la información y, por otro lado, estos datos se analizan a partir de una serie de indicadores explicados en el siguiente punto.

Para la transferencia del conocimiento se define la integración de la Wiki en Eclipse.

5.2. Indicadores CSF y KPI

Los factores críticos de éxito (CSF[18]) y el indicar clave de rendimiento (KPI[19]) son herramientas muy útiles para medir el éxito de un negocio. En este caso, de nuestra Wiki.

[18] CSF: Critical success factors
[19] KPI: Key Perfomance Indicator

CSF es definida como las áreas o las razones que deciden el éxito de la organización, en este caso, de nuestra Wiki. Por ejemplo, en nuestro caso la Wiki tendrá éxito si tiene **muchas consultas**.

Por otro lado, **KPI** es definida como la herramienta para medir e indicar el éxito del CSF. Por ejemplo, para medir muchas consultas podemos ver los **accesos al servidor web**.

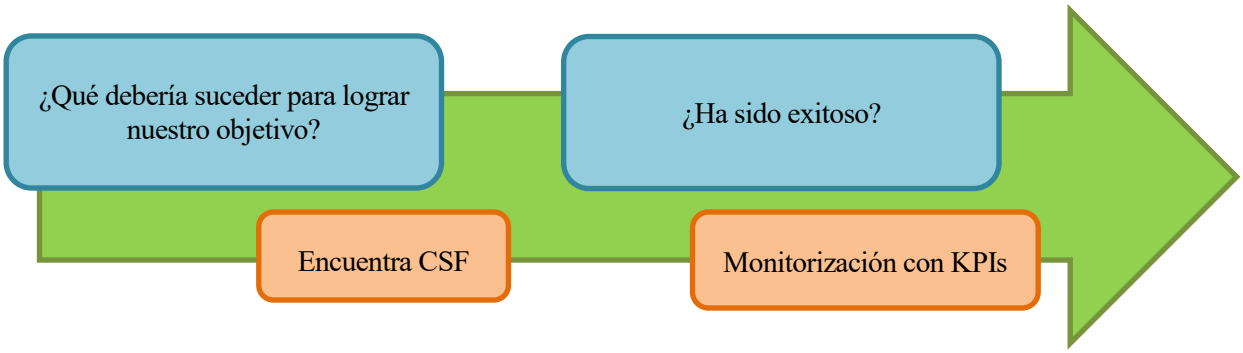


Fig 20 – CSF, KPI

5.2.1. CSF y KPI en la Wiki

CSF	KPI
Tener un gran número de consultas en las páginas de la Wiki	▪ IPs distintas que acceden a la Wiki ▪ Visitas a las páginas de la Wiki
Información de calidad	▪ Revisión de las páginas ▪ Última fecha de actualización ▪ Número de actualizaciones ▪ Plantillas descargadas
Fácil de usar	▪ Búsquedas en la Wiki

Tabla 6 – Indicadores de la Wiki

Para ver las medidas de las claves KPI se generan automáticamente una serie de informes con toda la información:

- Informe general
- Informe específico
- Top 3

5.1.2.1. Informe general

El informe general genera los resultados de todas las páginas de la Wiki con toda la información de la base de datos.

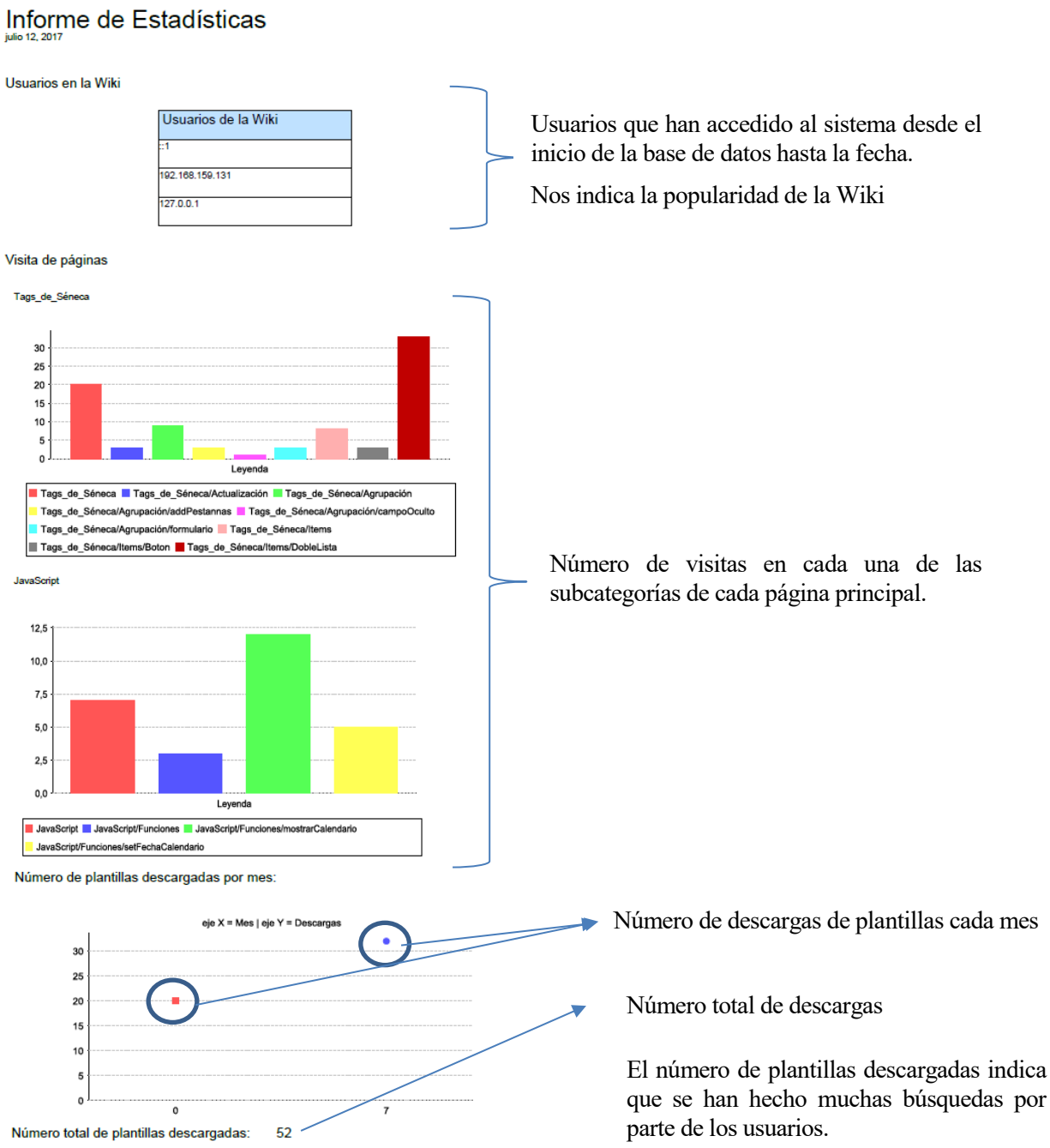
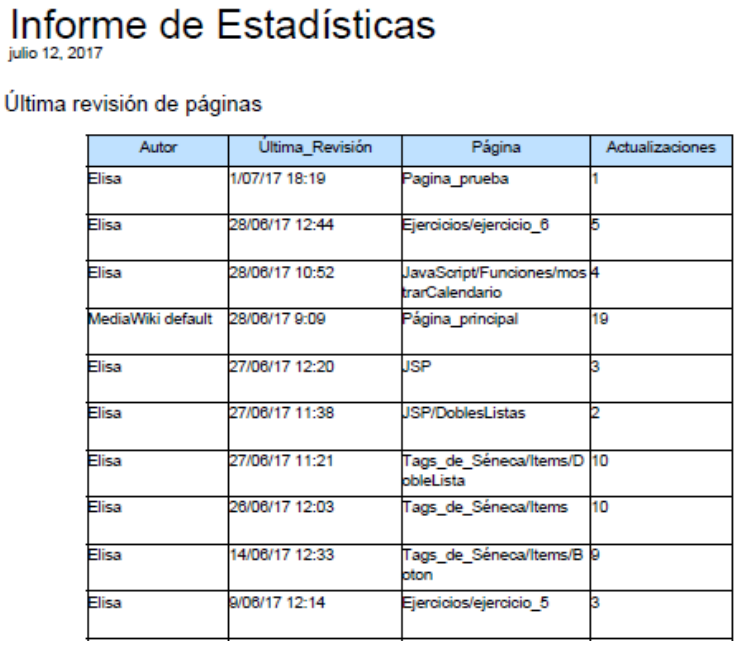


Fig 21 – Graficas informe general



Informe de Estadísticas
julio 12, 2017

Última revisión de páginas

Autor	Última_Revisión	Página	Actualizaciones
Elisa	1/07/17 18:19	Pagina_prueba	1
Elisa	28/06/17 12:44	Ejercicios/ejercicio_6	5
Elisa	28/06/17 10:52	JavaScript/Funciones/mostrarCalendario	4
MediaWiki default	28/06/17 9:09	Página_principal	19
Elisa	27/06/17 12:20	JSP	3
Elisa	27/06/17 11:38	JSP/DoblesListas	2
Elisa	27/06/17 11:21	Tags_de_Séneca/Items/DobleLista	10
Elisa	26/06/17 12:03	Tags_de_Séneca/Items	10
Elisa	14/06/17 12:33	Tags_de_Séneca/Items/Boton	9
Elisa	9/06/17 12:14	Ejercicios/ejercicio_5	3

Fig 22 – Revisión de páginas

En esta parte del informe de estadísticas se refleja la última revisión de cada página, quién llevó a cabo esa revisión y las actualizaciones totales que ha sufrido esa página desde el inicio de la base de datos hasta la fecha en la que se ha generado el informe.

Con esto, se pretenden sacar conclusiones con respecto a la calidad de la información. Por ejemplo, una página con una última revisión a la fecha actual y con muchas actualizaciones puede significar que la calidad de la información es mala y, por ello, sufre muchos cambios. O podría considerarse el caso contrario, es una página muy útil y se está mejorando lo máximo posible.

También nos ayuda a encontrar páginas que se han quedado abandonadas o que si tienen muchas visitas y la última revisión fue hace mucho tiempo significaría que tiene una información muy buena.

5.1.2.2. Informe específico

El informe específico nos permite definir las fechas de inicio y fin, la página principal y subpágina (en caso de tener) donde queremos definir las estadísticas.

Vamos a poner un ejemplo del resultado para Ejercicios (no tiene subpágina) desde el 1 de julio de 2017 hasta el 12 de julio de 2017.

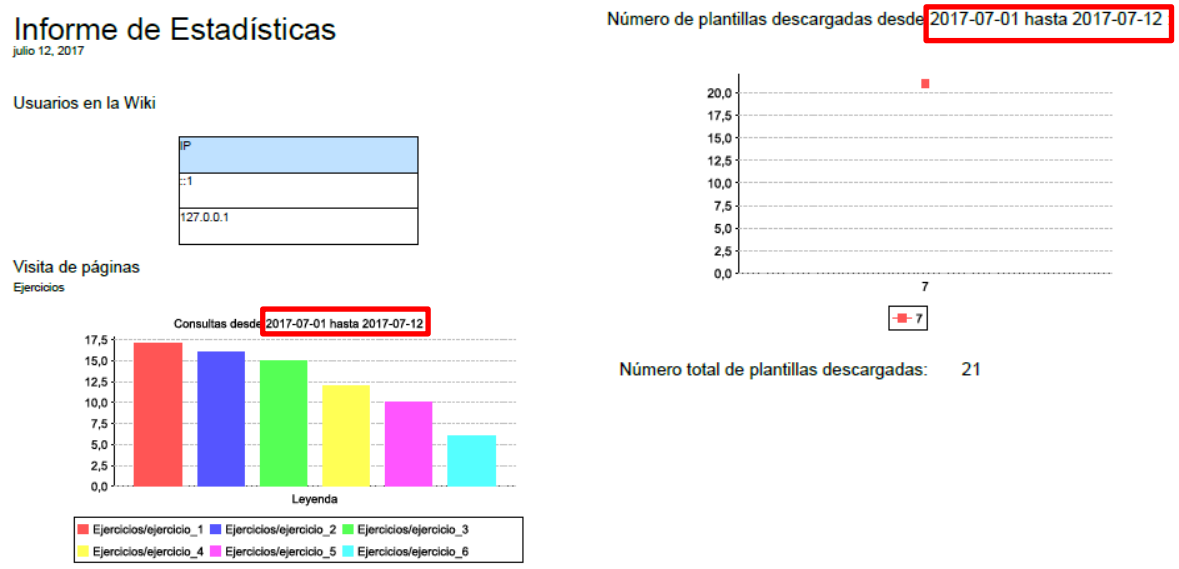
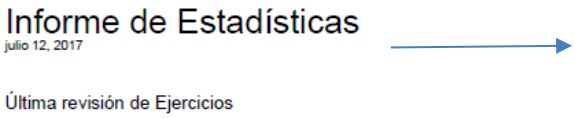


Fig 23 – Graficas informe específico

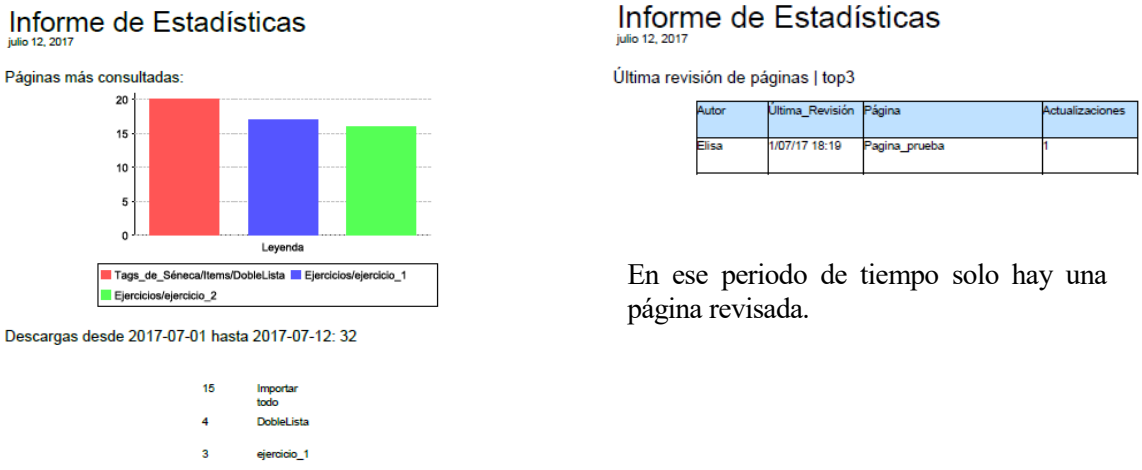


En el periodo de tiempo indicado no se ha hecho ninguna revisión a las páginas de "Ejercicios".

Fig 24 – Revisión informe específico

5.1.2.3. Top 3

El informe Top 3 nos muestra las 3 páginas más consultadas, las 3 plantillas más descargadas y las 3 últimas páginas revisadas de nuestra Wiki en un periodo de tiempo concreto. Lo que nos permite hacer un balance de cuáles son nuestras páginas más populares.



En ese periodo de tiempo solo hay una página revisada.

Fig 25 – Top 3

6. INTEGRACIÓN

La teoría es asesinada tarde o temprano por la experiencia.

-Albert Einstein -

En este capítulo vamos a explicar cómo se ha integrado nuestra Wiki en Eclipse. Para ello, explicaremos cómo se crea un nuevo proyecto de categoría plugin, cómo se ha programado nuestra aplicación, qué funcionalidades tiene…

6.1. Instalación

6.1.2. Eclipse

En primer lugar, para el desarrollo de este proyecto tenemos que instalar el software de Eclipse de la página oficial https://eclipse.org/ (de la versión Mars en adelante), para ello simplemente tenemos que descargar el archivo ejecutable compatible con nuestro sistema operativo.

6.1.3. Configuración de la Base de Datos

Finalmente, para utilizar la base de datos desde el plugin, tenemos que modificar la zona horaria de nuestra Base de Datos. Para el control de la misma el paquete XAMPP instala phpMyAdmin, un software para la administración de bases de datos MySQL o MariaDB. Para acceder a esta herramienta nos metemos en el navegador, en la URL http://127.0.0.1/phpmyadmin/ (en caso de que la BD[20] esté en local) nos aparecerá una página principal del software donde en la consola SQL pondremos lo siguientes comandos:

```
SET GLOBAL time_zone = 'Europe/Madrid';
SET time_zone = 'Europe/Madrid';
```

6.2. Editor Plugin

Una vez tenemos todo instalado, el primer paso para llevar a cabo nuestro proyecto es la creación de un nuevo

[20] BD: Base de Datos

"proyecto plugin" en Eclipse, para ello, vamos a File>New>Other… y seleccionamos la opción correspondiente.

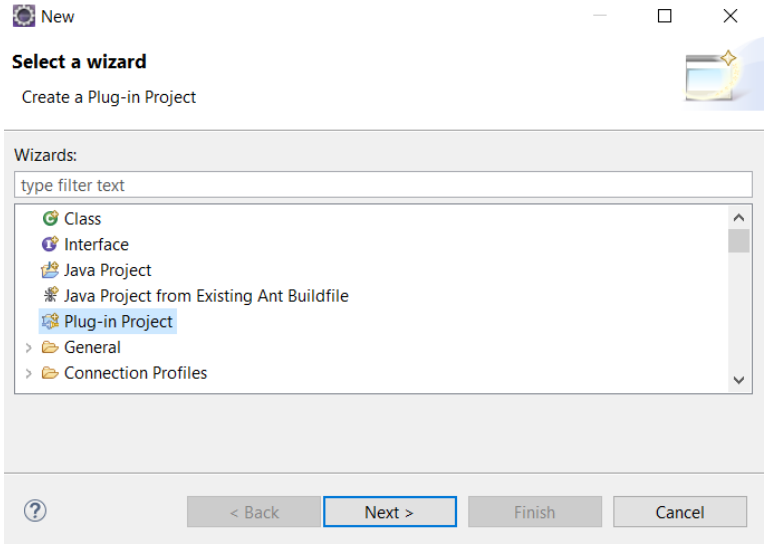


Fig 26 – Creación de un plugin

Ponemos un nombre al proyecto y nos aparecerá un editor como el siguiente:

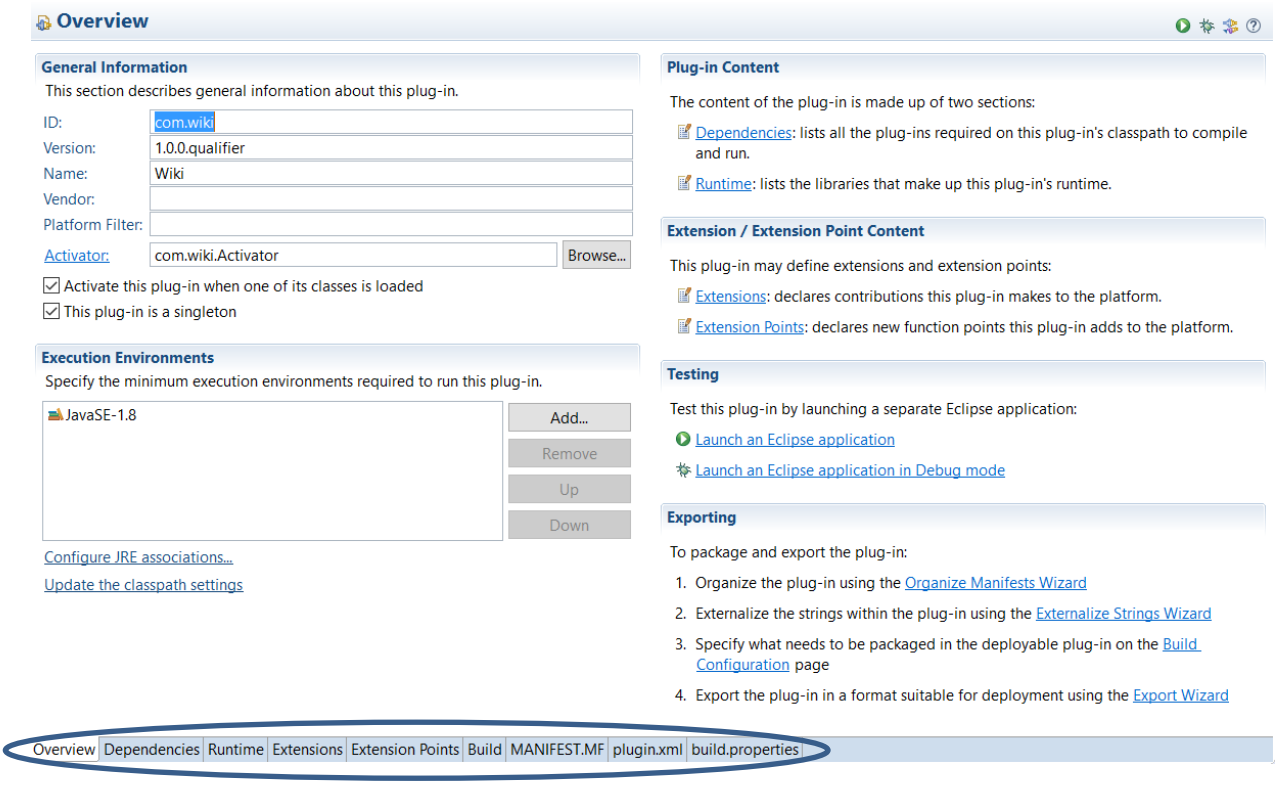


Fig 27 – Editor de un plugin

Abajo podemos ver una serie de pestañas que configuran nuestro plugin, serán explicadas a continuación [11]:

- Dependencias: En esta pestaña podemos indicar las dependencias de nuestro plugin con otro plugin. Sería como importar librerías.
- Runtime: Muestra los paquetes que el plugin hace visible a otros plugins, así, cuando otro cree dependencias con nuestro plugin puede utilizar las clases de nuestro proyecto.

- Extensiones: Extensiones tomadas de otros plugins. (El concepto de extensiones fue explicado en el punto 3.2.).
- Puntos de extensión: para ser utilizados por otros plugins (Concepto explicado en el punto 3.2.).
- Build: Contiene toda la información necesaria para construir, empaquetar y exportar el plugin.
- MANIFEST.MF: Archivo de metadatos que define datos relativos a la extensión y el paquete del plugin.
- Plugin.xml: Describe cómo nuestro plugin amplía la plataforma Eclipse, qué extensiones proporciona, qué puntos de extensión utiliza… Es un archivo escrito en XML y es analizado por Eclipse cuando este se inicia. Por ejemplo, si queremos añadir un nuevo botón a la barra de menú de Eclipse sería definido en este fichero.

6.3. Estructura del plugin

Antes de continuar, es importante tener clara la estructura que se ha dado al plugin de este proyecto. En nuestro caso, siguiendo los pasos anteriores hemos creado un proyecto llamado "com.wiki".

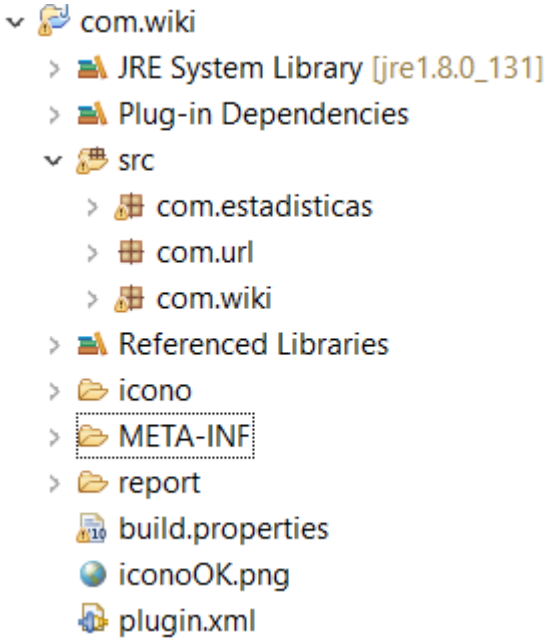


Fig 28 – Estructura del plugin "com.wiki"

El plugin "com.wiki" tendrá tres extensiones divididas en tres paquetes java distintos:

- **com.estadisticas:** Extensión que nos mostrará las estadísticas de la Wiki y utilidad del propio plugin. Con esta extensión podremos sacar conclusiones sobre la rentabilidad del proyecto.
- **com.url:** Extensión que nos lleva directamente a la página principal de la Wiki a través del navegador.
- **com.wiki:** Extensión que genera plantillas de código de forma automática a partir de las peticiones del usuario y la información almacenada en la Wiki.

La principal referencia para el desarrollo de código ha sido la web java2s [16]

6.4. Extensión com.wiki

La extensión com.wiki se añadirá tanto en la barra de herramientas como en la barra de menú. Su principal función es realizar búsquedas en la Wiki según la petición del usuario y generar la plantilla de código en el editor de texto activo de Eclipse.

Como podemos ver en la figura 20, dentro del paquete com.wiki podemos encontrar diferentes clases que dan forma a nuestra extensión.

En los siguientes puntos se va a explicar la funcionalidad de cada una de ellas y sus puntos más relevantes a nivel de código.

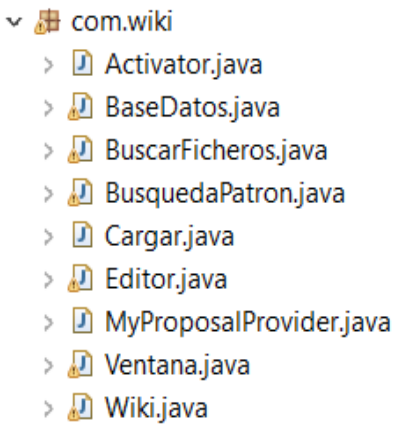


Fig 29 – Clases de la extensión "com.wiki"

6.4.1. Activator.java

Clase auto-generada al iniciar el proyecto plugin. [5] Gracias a esta clase el plugin se inicia o se para con los métodos start() o stop() respectivamente.

6.4.2. Wiki.java

La clase Wiki es la clase principal de esta extensión. Cuando pulsamos alguno de los botones de acceso de la Wiki en Eclipse, en primer lugar, se llamaría al método run() de esta clase.

Cuando añadimos una extensión al workbench de Eclipse, la clase principal de dicha extensión tiene que ser una implementación de la interfaz IWorkbenchWindowActionDelegate. Dicha interfaz [10] nos ayuda a inicializar la ventana o workbench de Eclipse y así poder hacer uso de ella. A su vez, dicha interfaz es una extensión de IActionDelegate quien a partir de métodos como run() nos ayuda a crear una acción para esa inicialización.

```
public class Wiki implements IWorkbenchWindowActionDelegate {
    IWorkbenchWindow activeWindow = null;

    // Se lleva a cabo la acción
    public void run(IAction proxyAction)  {
        try {
            //Ventana actual
            Shell ventanaActual = activeWindow.getShell();

            //Cuando pulsemos el botón saldrá una nueva ventana emergente
            //la instanciamos y abrimos
            Ventana window = new Ventana();
            window.open();
            String info = window.resultado();
            System.out.println(info);

            if (info!=null){
                //Hay que poner el texto en el editor de la ventanaActual
                //Tomamos el editor actual
                IEditorPart editor = Activator.getDefault().getWorkbench().getActiveWorkbenchWindow()
                        .getActivePage().getActiveEditor();
                if (editor==null){
                    MessageDialog.openInformation(ventanaActual, "Informacion", "No hay ningún fichero abierto");
                }
                else{

                    Editor clase_editor = new Editor(ventanaActual, editor, info);
                    clase_editor.open();
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

      }

    // IWorkbenchWindowActionDelegate method = tomamos la ventana/workbench activo
    public void init(IWorkbenchWindow window) {
        activeWindow = window;
    }
```

Fig 30 – Wiki.java

El método “init” se corresponde con el método de inicialización de la interfaz IWorkbenchWindowActionDelegate. En este método lo que se hace es inicializar la ventana activa de Eclipse y así convertirla en nuestro escenario para sacar ventanas emergentes, tomar el editor o cualquiera de los módulos de Eclipse.

Una vez que tenemos nuestra ventana hay que volcar acciones sobre la misma. Esto se lleva a cabo en el método run().

Cuando pulsamos el botón que llama a esta clase queremos que aparezca una ventana emergente. Esto lo conseguimos creando un objeto de la clase Ventana (Ventana.java) que veremos más adelante.

Para obtener el objeto del editor actual (línea importada de la API de Eclipse [10])

Se crea un objeto de clase Editor (Editor.java) que será quien escriba el texto resultante de la búsqueda en la Wiki.

```
// Se lleva a cabo la acción
public void run(IAction proxyAction)  {
    try {
        //Ventana actual
        Shell ventanaActual = activeWindow.getShell();

        //Cuando pulsemos el botón saldrá una nueva ventana emergente
        //la instanciamos y abrimos
        Ventana window = new Ventana();
        window.open();
        String info = window.resultado();
        System.out.println(info);

        if (info!=null){
            //Hay que poner el texto en el editor de la ventanaActual
            //Tenemos el editor actual
            IEditorPart editor = Activator.getDefault().getWorkbench().getActiveWorkbenchWindow()
                    .getActivePage().getActiveEditor();
            if (editor==null){
                MessageDialog.openInformation(ventanaActual, "Informacion", "No hay ningún fichero abierto");
            }
            else{

                Editor clase_editor = new Editor(ventanaActual, editor, info);
                clase_editor.open();
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

Fig 31 – Método run() de la clase Wiki

6.4.3. Ventana.java

Esta extensión cuenta con una interfaz gráfica como podemos ver en la siguiente figura. Nuestro objetivo es facilitar al programador el proceso de búsqueda lo máximo posible.

La clase encargada de ofrecer esta interfaz al usuario es Ventana, quien a partir de Java SWT creará una vista como la mostrada en la siguiente figura. Para el desarrollo de esta clase, principalmente, se ha usado la siguiente referencia [16].

Fig 32 – Interfaz gráfica Wiki en Eclipse

Como podemos ver, la interfaz gráfica de la wiki en Eclipse es una ventana emergente con diferentes pestañas. Cada pestaña se corresponde con una página principal tal y como se vio en el capítulo 4 "Estructura de la Wiki" y dentro de las mismas se ofrece un formulario con diferentes opciones para viajar por las subpáginas y que el usuario decida el elemento de código que quiere importar.

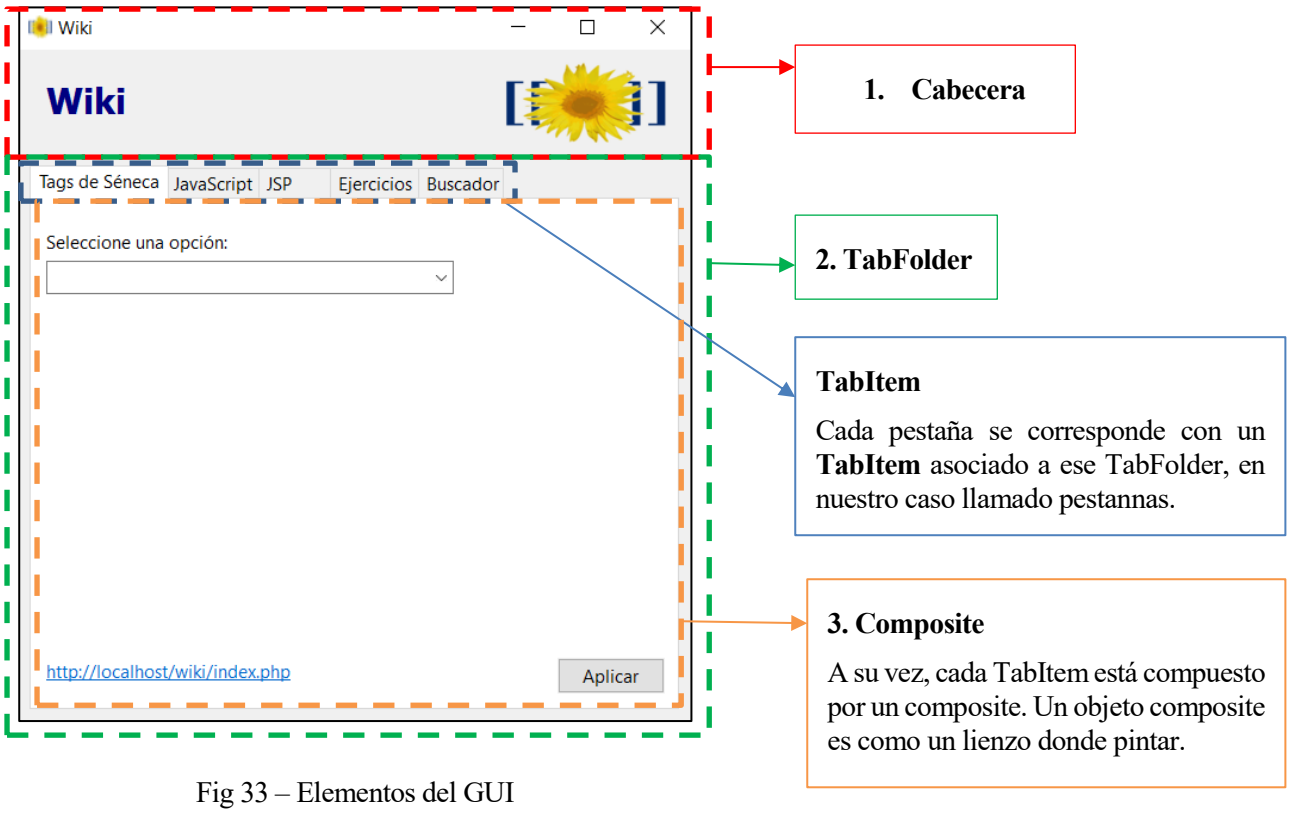


Fig 33 – Elementos del GUI

1. Cabecera

```
Composite composite = new Composite(shell, SWT.NONE);
composite.setBounds(0, 0, 557, 91);
```

Indica la posición del elemento en la ventana (shell)

```
// Titulo: Wiki
Label Wiki = new Label(composite, SWT.NONE);
Wiki.setBounds(24, 24, 95, 40);
Wiki.setForeground(Wiki.getDisplay().getSystemColor(SWT.COLOR_DARK_BLUE));
Wiki.setFont(new Font(Wiki.getDisplay(), new FontData("Tahoma", 18, SWT.BOLD)));
Wiki.setText("Wiki");

// Icono de la ventana
Label icono_shell = new Label(shell, SWT.NONE);
Image imagen_icono = new Image(icono_shell.getDisplay(),
"C:/Users/elisa/workspace/com.wiki/icono/icono.ico");
shell.setImage(imagen_icono);
```


```
// Icono
Label icono = new Label(composite, SWT.NONE);
// Cargamos la imagen de la cabecera
Image imagen = new Image(icono.getDisplay(),
"C:/Users/elisa/Desktop/¿TFG/cabecera2.ico");
icono.setImage(imagen);
icono.setBounds(407, 10, 140, 71);
```

2. TabFolder

```
TabFolder pestannas = new TabFolder(shell, SWT.NONE);
pestannas.setBounds(10, 97, 547, 457);

/* Pestañas de opciones */
TabItem pestanna_html = new TabItem(pestannas, SWT.NONE);
pestanna_html.setText("Tags de Séneca");
TabItem pestanna_js = new TabItem(pestannas, SWT.NONE);
pestanna_js.setText("JavaScript");
TabItem pestanna_JSP = new TabItem(pestannas, SWT.NONE);
pestanna_JSP.setText("JSP");
TabItem pestanna_Ejercicios = new TabItem(pestannas, SWT.NONE);
pestanna_Ejercicios.setText("Ejercicios");
TabItem pestanna_Buscador = new TabItem(pestannas, SWT.NONE);
pestanna_Buscador.setText("Buscador");
```

3. Composite

```
/* Composite para cada pestaña */
Composite compositeHTML = new Composite(pestannas, SWT.NONE);
pestanna_html.setControl(compositeHTML);
Composite compositeJavaScript = new Composite(pestannas, SWT.NONE);
pestanna_js.setControl(compositeJavaScript);
Composite compositeJSP = new Composite(pestannas, SWT.NONE);
pestanna_JSP.setControl(compositeJSP);
Composite compositeEjercicios = new Composite(pestannas, SWT.NONE);
pestanna_Ejercicios.setControl(compositeEjercicios);
Composite buscador = new Composite(pestannas, SWT.NONE);
pestanna_Buscador.setControl(buscador);
```

Para crear cada uno de los composite de cada una de las pestañas de la clase Ventana existen diferentes métodos cuyo aspecto, código y funcionalidad iremos viendo a continuación.

`public void crearComposite(TabItem pestanna, Composite composite, Shell shell) ->` Para las pestañas Tags_de_Séneca, JavaScript y JSP

`public void crearCompositeEjercicios(Composite compositeEjercicios, Shell shell) ->` Para la pestaña “Ejercicios”

`public void crearCompositeBuscador(Composite compositeBuscador, Shell shell) ->` Para la pestaña “Buscador”

6.4.3.1. Pestañas Tags_de_Séneca, JavaScript y JSP

En el caso del TabItem “Tags_de_Séneca”, “JavaScript” y “JSP” utilizamos el mismo método debido a que visualmente son iguales.

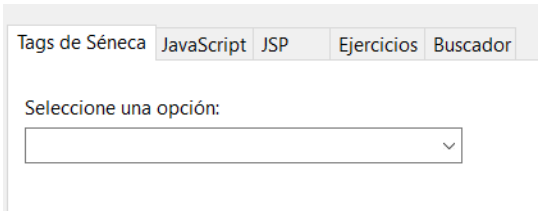


Fig 34 - Combo

Uno de los primeros elementos que podemos encontrar en este tipo de composite es un “Combo” o lista desplegable.

```
Label lblsel_opcion = new Label(composite, SWT.NONE);
lblsel_opcion.setBounds(10, 25, 342, 20);
lblsel_opcion.setText("Seleccione una opción:");

final Combo comboOpciones = new Combo(composite, SWT.NONE);
comboOpciones.setText("");
comboOpciones.setBounds(10, 51, 342, 28);
```

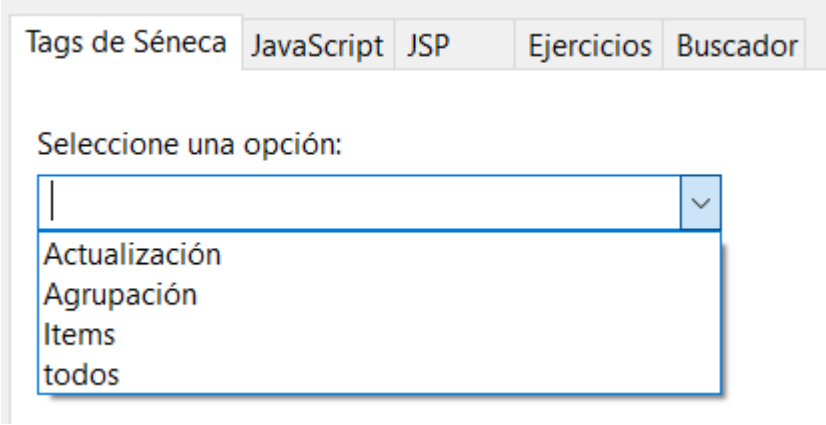


Fig 35 – Subpáginas dentro de un combo

Para alimentar los combos hacemos consultas a la base de datos de la Wiki, en concreto, para llenar las listas desplegables se lanzan consultas a la tabla "tlpage", lugar donde se almacenan todas las páginas de la Wiki, a través de la clase BaseDatos explicada más adelante.

Código:

```
BaseDatos BD = new BaseDatos();
```

```
consulta = "SELECT page_title FROM tlpage WHERE page_title LIKE '" +
pestanna.getText().replaceAll(" ", "_")
+ "/%' "+"AND page_title NOT LIKE '" +pestanna.getText().replaceAll(" ", "_")
+"/%/%'";
```

Busca las subpáginas para ese tipo de pestaña.

```
BD.abrirConexion();
opciones_aux = BD.consultarPaginas(consulta);
String[] opciones = new String[opciones_aux.length + 1];
for (int i = 0; i < opciones_aux.length + 1; i++) {
      if (i == opciones_aux.length)
            opciones[i] = "todos";
      else
            opciones[i] = opciones_aux[i];
}
BD.cerrarConexion();
comboOpciones.setItems(opciones);
```

Añade la opción "todos" a la lista desplegable.

6.4.3.1.1. Listener

Los listener, como su propio nombre indica, son objetos que escuchan hasta que ocurre un evento. Java SWT nos ofrece una serie de primitivas para distintas acciones, pulsar, modificar, pasar el ratón por encima…

Podemos encontrar una lista de eventos en la siguiente referencia [17]

En este caso, la idea es que en función de la subpágina elegida aparezca un nuevo combo cargado con la subsubpágina correspondiente. Para ello, aplicamos los listener, cuando un usuario elige una opción de "comboOpciones" saltará un listener que creará un nuevo combo y lo rellenará con otra consulta acorde a la elección del usuario.

Código:

```
comboOpciones.addSelectionListener(new SelectionListener() {

	public void widgetSelected(SelectionEvent evento) {
		String consulta = "";
		if (evento.widget == comboOpciones) {
			limpiar(bandera);
			bandera = 0;
			subCategoria = comboOpciones.getText();
			if (!pestanna.getText().equals("JSP")) {
				Label lblele_disponibles = new Label(composite, SWT.NONE);
				lblele_disponibles.setBounds(10, 96, 342, 20);
				lblele_disponibles.setText("Elementos disponibles:");
				combo.setBounds(10, 122, 342, 28);

				if (comboOpciones.getText().equals("todos"))
				   consulta = "select page_title from tlpage where"+
				              +"  page_title LIKE '
				              +pestanna.getText().replaceAll(" ", "_")
				               + "/%/%'";
				else
				  consulta = "select page_title from tlpage where"+
				             +" page_title LIKE '%/"
				              + comboOpciones.getText() + "/%'";
				BD.abrirConexion();
				String[] paginas = BD.consultarPaginas(consulta);
				BD.cerrarConexion();
				propuestas = paginas;
				combo.setItems(paginas);
```

Método que nos ayuda a limpiar los composite.

Nuevo combo

Rellena el nuevo combo con la información correspondiente

Por lo tanto, en este tipo de composite cuando el usuario elige una opción en el primer combo aparecerá un segundo combo con las opciones correspondientes a la primera.

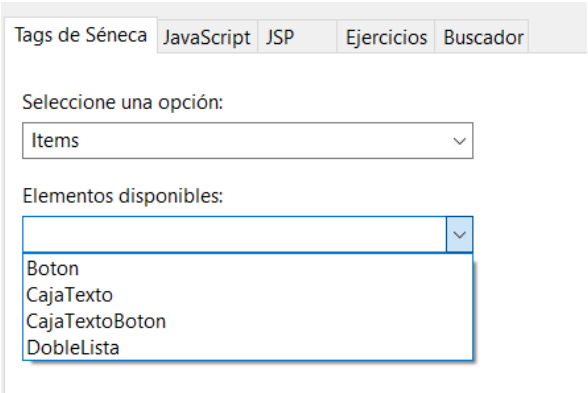


Fig 36 – Combo II crearComposite

6.4.3.1.2. Sugerencias: Proposal provider

En el futuro se espera una Wiki con un gran tamaño, por esta razón, en los realimentados con las páginas de la Wiki podría ser un auténtico reto encontrar lo que se busca. Para ello, se ha añadido la mejora de sugerir al usuario posibles opciones en función de lo que escriba en el combo, cuadro de texto, etc.

Normalmente en Eclipse para conseguir esto se escribe la siguiente línea de código:

```
new AutoCompleteField(textSugerencia, new TextContentAdapter(),propuestas);
```

Texto escrito en el combo

Objeto que hace el control de contenido entre textSugerencia y propuestas

Todas las opciones de ese combo.

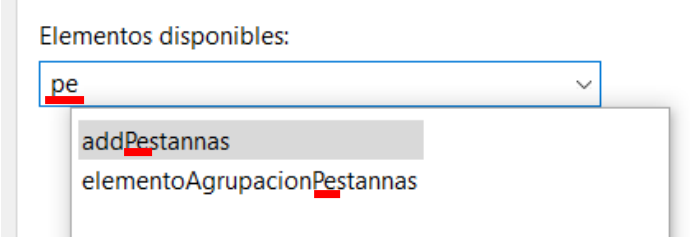


Fig 37 – Sugerencia en combos

Con la anterior línea únicamente sugiere entre las opciones que coincidan exactamente con el texto escrito por el usuario, es decir, para encontrar la página "addPestannas" el usuario tendría que escribir "addp…". Sin embargo, nuestro objetivo era sugerir las páginas que contenían el texto escrito por el usuario [Fig28] Para ello, se creó una nueva clase llamada MyProposalProvider, que buscará las palabras que contengan el texto del combo. Así, haciendo uso de la misma nos quedaría el siguiente código: [18]

```
MyProposalProvider provider = new MyProposalProvider(propuestas);

ContentProposalAdapter adapter = new ContentProposalAdapter(combo, new
ComboContentAdapter(),provider, null, null);

adapter.addContentProposalListener(new IContentProposalListener() {
       // Limpiamos todo lo que haya en text
       public void proposalAccepted(IContentProposal arg0) {
              String aux = combo.getText();
              for (int i = 0; i < propuestas.length; i++) {
                     int indice = aux.indexOf(propuestas[i]);
                     if (indice != -1) {
                            String quitar = aux.substring(0, indice);
                            aux = aux.replace(quitar, "");
                            combo.setText(aux);
                     }
              }
       }
});
```

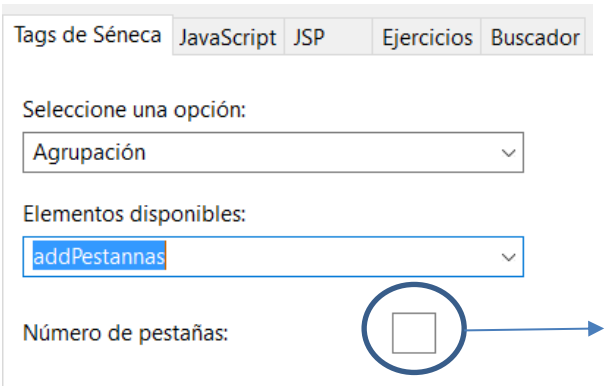
Añadimos el adaptador de sugerencias al combo.

Si alguna sugerencia es aceptada por el usuario se borra lo escrito en el combo y se pone la página sugerida.

6.4.3.1.3. Funcionalidades especiales

Principalmente, en la pestaña "Tags_de_Séneca" se han añadido algunas mejoras en ciertos elementos con idea de hacer mucho más útil la herramienta de la Wiki.

1. **addPestannas, CampoOculto**

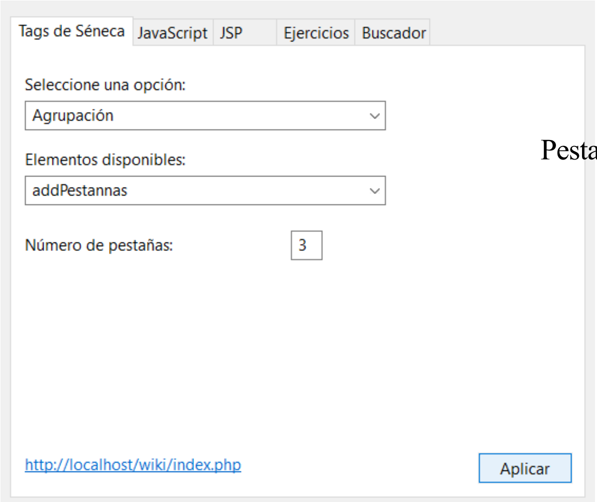


Estos elementos pertenecen a la ruta Tags_de_Séneca>Agrupación. Lo que se permite es que el usuario decida qué número de elementos desea. Por ejemplo, addPestannas es una etiqueta con la que se crean pestañas en una página web, si el usuario tiene que poner tres pestañas simplemente tendría que indicarlo en nuestra herramienta, la cual generará el código para tres pestañas.

```
text = new Text(composite, SWT.BORDER);
// text.setBounds(x, y, width, height);
text.setBounds(262, 174, 30, 28);
```

Fig 38 – Mejora Tags_de_Séneca>Agrupación

Resultado:



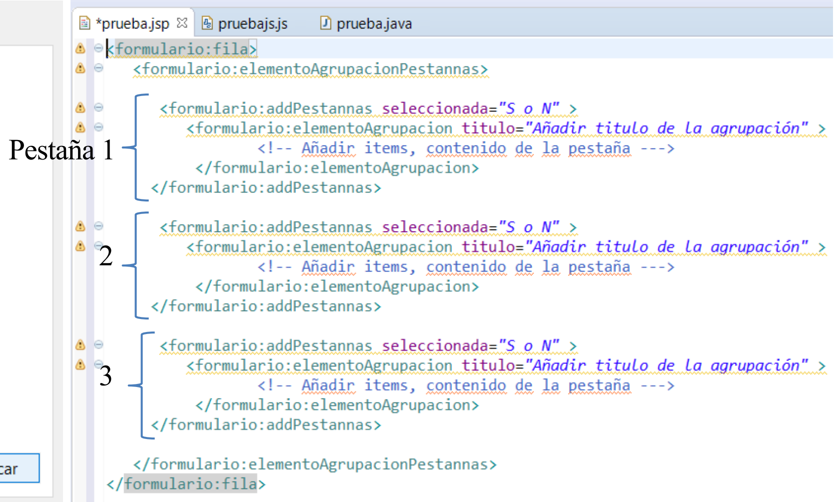


Fig 40 – Funcionalidad addPestannas

Fig 39 – Código addPestannas

Cómo conseguir esto lo veremos más adelante en BusquedaPatrón.java y Editor.java.

2. Tags_de_Séneca > Items

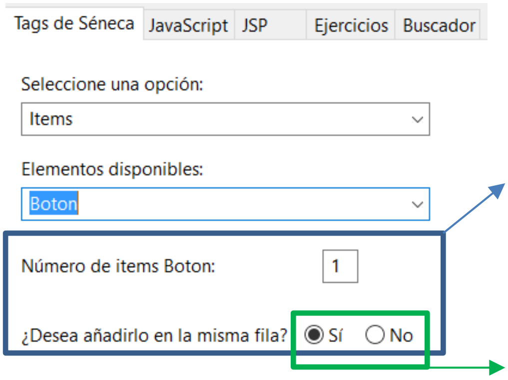


Fig 41- Funcionalidad Tags_de_Séneca>Items

Los ítems son elementos tales como botones, listas desplegables, tablas… Estos elementos pueden añadirse en una misma fila o en varias, con la aplicación se permite que el usuario decida el número de ítems que quiere y dónde los quiere.

Esto se añade dentro del listener del segundo combo.

```
btnNo = new Button(composite, SWT.RADIO);
btnNo.setBounds(298, 235, 111, 20);
btnNo.setText("No");
btnNo.addSelectionListener(new SelectionAdapter()
{
   public void widgetSelected(SelectionEvent e) {
      radio = "No";
      btnS.setSelection(false);
   }
});

btnS = new Button(composite, SWT.RADIO);
btnS.setBounds(247, 235, 45, 20);
btnS.setText("Sí");
btnS.setSelection(true);
btnS.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {
      radio = "Si";
    }
```

6.4.3.1.4. Botón aplicar

Cuando pulsamos el botón aplicar de alguna de estas pestañas es fácil intuir que salta un evento asociado a un listener.

Dicho listener lo que hace es definir la página que tiene que buscar en la Wiki para conseguir la plantilla de código, por ejemplo, si un usuario está en la pestaña Tags_de_Séneca > Agrupación > formulario la página que habría que buscar en la Wiki sería Tags_de_Séneca/Agrupación/formulario. Nuestro listener se encargaría de formar esa cadena con los datos del formulario.

```
Listener listener = new Listener() {
	public void handleEvent(Event evento) {
		if (evento.widget == boton) {
		   // Actualizamos estadísticas
		   nombre_pestanna = pestanna.getText();
		   elemento = combo.getText();
		   actualizar_estadisticas();

		   String subpagina = "";
		   String subsubpagina = "";
		   // Según lo elegido buscamos la información
		   BusquedaPatron buscar = new BusquedaPatron();
		  // Cuando el comboOpciones es todos ---> hay que transformar:
		   if (comboOpciones.getText().equals("todos")) {
		        subsubpagina = "/" + combo.getText();

		         opciones_aux = BD.consultarPaginas("SELECT page_title FROM
tlpage WHERE page_title LIKE '" + pestanna.getText().replaceAll(" ", "_") + "/%' " +
"AND page_title NOT LIKE '" + pestanna.getText().replaceAll(" ", "_") + "/%/%'");

		         for (int i = 0; i < opciones_aux.length; i++) {

		              String consulta_aux = "select page_title from tlpage where
page_title LIKE '%/" + opciones_aux[i] + "/%'";
		              String[] paginas_aux = BD.consultarPaginas(consulta_aux);

		              for (int p = 0; p < paginas_aux.length; p++) {
		                   if (paginas_aux[p].equals(combo.getText()))
		                        subpagina = opciones_aux[i];
		              }
		         }
		   } else {
		              subpagina = comboOpciones.getText();
		              if (!combo.getText().equals(""))
		                   subsubpagina = "/" + combo.getText();
		   }

		   String pagina = pestanna.getText().replaceAll(" ", "_") + "/" +
subpagina + subsubpagina;

		   int repeticiones = 0;
		   if (text != null)
		        repeticiones = Integer.parseInt(text.getText());
		   // radio: por defecto es Si
		   texto = buscar.buscar(pagina, pestanna.getText().replaceAll(" ",
"_"), repeticiones, radio);

		   shell.close();
		}
	}
};
```

Clase que se encarga de buscar las plantillas de código.

Código que define la cadena de la página a buscar **(explicación más abajo)**

Cierra la ventana emergente, el usuario ya ha terminado.

Para definir la página de la Wiki donde la clase BusquedaPatrón tiene que trabajar tenemos dos situaciones:

1. El usuario a rellenado todos los campos del formulario y simplemente tomamos el valor de los combos.
2. El usuario ha elegido en el primer combo la opción “todos” porque no sabía en qué subcategoría estaba el elemento que buscaba. En este caso (primer if del código), hay que hacer una consulta a la base de datos buscando la subpágina (subcategoría) correspondiente a la subsubpágina (elemento elegido).

Si recordamos la estructura de la Wiki: Página_principal/subpágina/subsubpágina

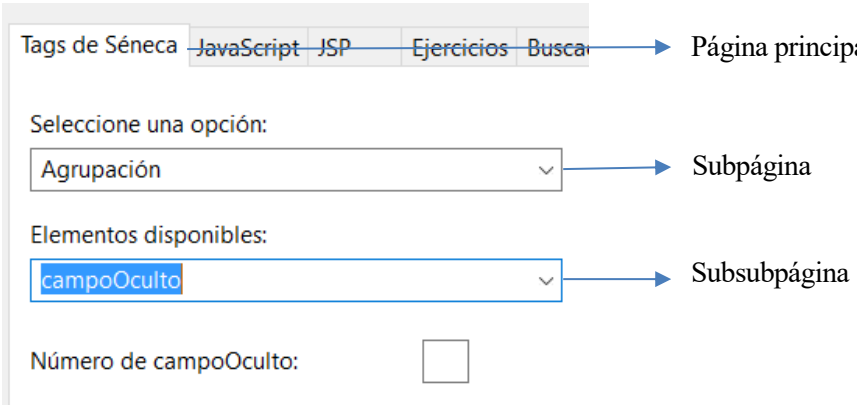


Fig 42 – Similitud aplicación - Wiki

6.4.3.2. Pestaña Ejercicios

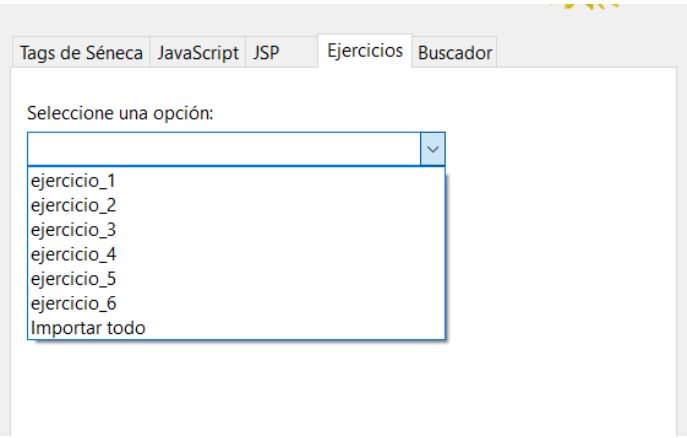


Fig 43 – Composite "Ejercicios" I

Cuando un nuevo empleado se incorpora a la plantilla, en primer lugar, tiene que realizar una serie de ejercicios para familiarizarse con el entorno. La aplicación permite descargar los ficheros correspondientes a estos ejercicios en nuestro equipo.

A priori, la pestaña "Ejercicios" funciona exactamente igual que el caso anterior. Un primer combo donde tenemos acceso a las subpáginas, en este caso, todos los ejercicios. La diferencia viene en el listener de la lista desplegable que, en este caso, actúa de la siguiente forma:

```
table = new Table(compositeEjercicios,
SWT.CHECK);

table.setBounds(95, 214, 351, 151);
```

```
TableItem tableitem = new TableItem(table,
SWT.NONE);

tableitem.setText(fich[i]);
```

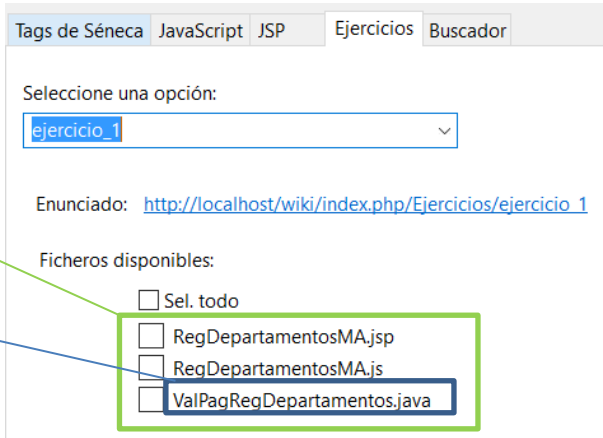


Fig 44 – Composite "Ejercicios" II

SWT.CHECK: Indica que cada elemento que se añada a la tabla tendrá un cuadro de "check".

Esta tabla se rellenará con los ficheros asociados a dicho ejercicio elegido en la lista desplegable, para encontrarlos se utiliza una nueva clase llamada "BusquedaFicheros" (BusquedaFicheros.java) quien se encargará de hacer las consultas oportunas en la base de datos.

Por otro lado, la propia tabla tendrá un listener para controlar sin un fichero se selecciona o no:

```
table.addSelectionListener(new SelectionAdapter() {
      public void widgetSelected(SelectionEvent arg0) {
            for (int i = 0; i < arrayItems.length; i++) {
                  if (arrayItems[i].getChecked())
                        seleccionado[i] = i;
            }
      }
});
```

ArrayItems es una tabla auxiliar de "TableItems", lo que se hace en este listener es comprobar cuál o cuáles elementos han sido seleccionados y guardar esa posición en la tabla para cuando se pulse el botón "Importar" saber que ficheros se requieren.

6.4.3.2.1. Botón Importar

Igual que antes, el botón tiene un listener. Primero, aparece una ventana para decidir en qué directorio desea el usuario guardar los ficheros. [16]

```
// Al pulsar el botón importar sale el browser para
// seleccionar directorio
final Shell shell_browser = new Shell();
DirectoryDialog dlg = new DirectoryDialog(shell_browser);
BuscarFicheros ficheros = new BuscarFicheros();

// Ruta donde se guardan los ficheros
String selectedDirectory = dlg.open();
```

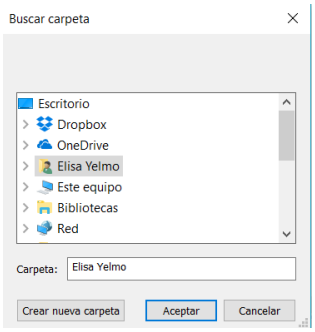


Fig 45 – Ventana browser

Una vez que la ruta donde guardar está decidida el siguiente paso es indicar a la clase BusquedaPatrón (BusquedaPatrón.java) qué fichero tiene que buscar y se escribe en un fichero en la ruta definida por el usuario y una jerarquía de carpetas definidas por el proyecto.

```
while (i < tamanno) { // Importar todo funciona de momento
      if (arrayItems[i].getChecked() || sel_todo == 1) {
            if (fich[i].contains("jsp")) {
                  actualizar_estadisticas();
                  // Estructura de carpetas donde se
                  // guardarán los ejercicios
                  File carpeta_jsp = new File( selectedDirectory + "/" + directorio
                                          + "/jsp/datosauxiliares");
                  carpeta_jsp.mkdirs();
                  fichero = new FileWriter(carpeta_jsp.getAbsolutePath() + "/" +
                                          fich[i]);
            } else if (fich[i].contains("js")) {
                  actualizar_estadisticas();
                  File carpeta_scripts = new File(selectedDirectory + "/" +
                                          directorio + "/scripts/datosauxiliares");
                  carpeta_scripts.mkdirs();
                  fichero = new FileWriter(carpeta_scripts.getAbsolutePath() + "/"
                                          + fich[i]);
            } else if (fich[i].contains("java")) {
                  actualizar_estadisticas();
                  File carpeta_java = new File(selectedDirectory + "/" + directorio
                                          + "/java/sadiel/cec/datosauxiliares");
                  carpeta_java.mkdirs();
                  fichero = new FileWriter(carpeta_java.getAbsolutePath() + "/" +
                                          fich[i]);
            }
```

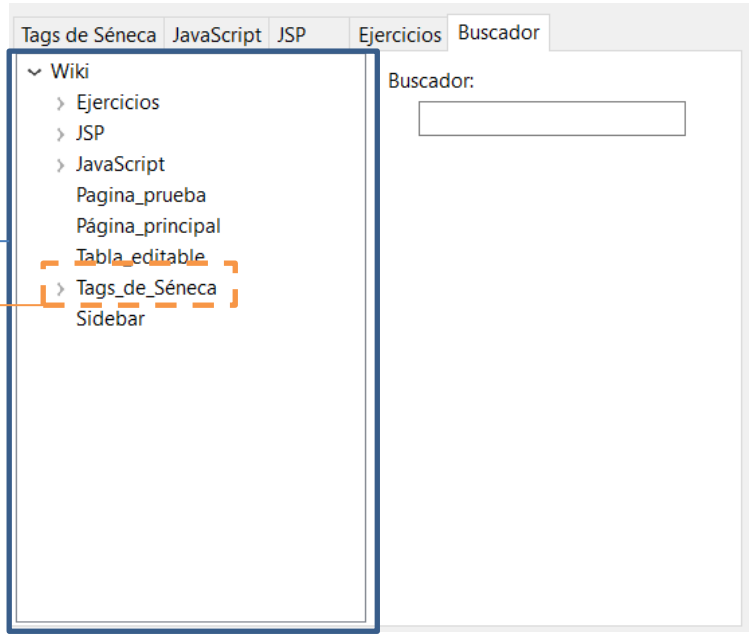
Ejercicio_1

Ejercicio_2 …

```
                    BufferedWriter buffer = new BufferedWriter(fichero);
                    BusquedaPatron buscar = new BusquedaPatron();
                    buffer.write(buscar.buscarEjercicios("Ejercicios", directorio,
                                                         fich[i]));
                    buffer.close();
                }
        i++;
}
```

6.4.3.3. Pestaña Buscador

La pestaña "Buscador" se diseñó con la idea de, por un lado, de ayudar al usuario a entender la estructura de la Wiki y, por otro lado, ayudarle a encontrar un elemento que no sabe en qué categoría se encuentra.




```
Tree tree = new Tree
(compositeBuscador, SWT.BORDER);
tree.setBounds(0, 0, 261, 424);

TreeItem itemWiki = new TreeItem
(tree, SWT.NULL);
itemWiki.setText("Wiki");
```

Fig 46 – Pestaña Buscador

Como siempre, el árbol tiene la misma estructura página principal/subpágina/subsubpágina que la Wiki y se a partir de consultas a la base de datos, por lo que el árbol, al igual que el resto de elementos de la ventana, se actualizaría automáticamente según vaya creciendo la Wiki.

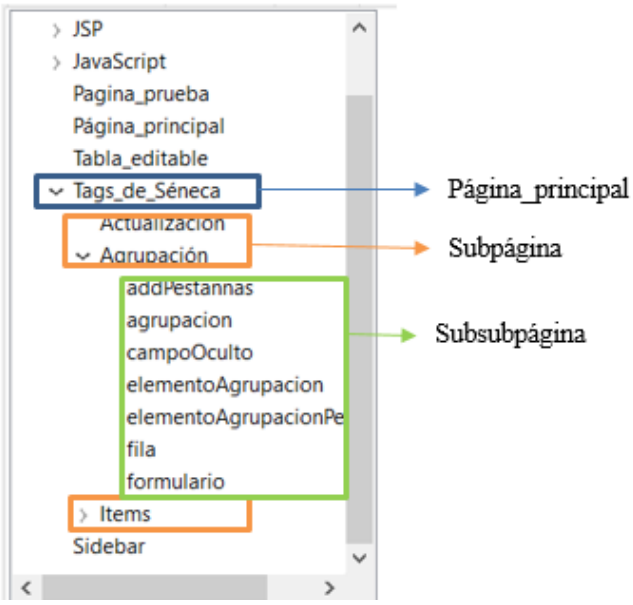


Fig 47 – Estructura árbol buscador

```
pagPrincipal = BD.consultarPaginas(CONSULTA_PAG_PRINCIPAL);
            for (int i = 0; i < pagPrincipal.length; i++) {
                  TreeItem itemPag_principal = new TreeItem(itemWiki, SWT.NULL);
                  itemPag_principal.setText(pagPrincipal[i]);

                  // Cada página principal tendrá hijos y esos hijos otros hijos...
                  String consulta_subpagina = "SELECT page_title from tlpage where
                                              page_title LIKE '" +
                                              pagPrincipal[i].replace(" ", "_") +
                                             "/%' and page_title NOT LIKE '%/%/%'";
                  String[] subPag = BD.consultarPaginas(consulta_subpagina);

                  for (int p = 0; p < subPag.length; p++) {
                        TreeItem itemPag_subpagina = null;
                        if (subPag.length != 0 || subPag[p] != null) {
                              itemPag_subpagina = new TreeItem(itemPag_principal,
                                                               SWT.NULL);
                              itemPag_subpagina.setText(subPag[p]);
                              propuestas[control] = subPag[p];
                        }
                   if (itemPag_subpagina != null) {
                        String consulta_subsubpagina = "SELECT page_title from + +
                                                       tlpage where page_title LIKE '"
                                                       + pagPrincipal[i] + "/" +
                                                        subPag[p] + "/%'";
                        String[] subsubPag =  BD.consultarPaginas
                                                          (consulta_subsubpagina);
                        if (subsubPag.length != 0) {
                            for (int k = 0; k < subsubPag.length; k++) {
                                  TreeItem itemPag_subsubpagina = new
                                             TreeItem(itemPag_subpagina, SWT.NULL);
                                   itemPag_subsubpagina.setText(subsubPag[k]);
                                   propuestas[control] = subsubPag[k];
                              }
                        }
                  }
            }
```

Buscamos e insertamos las páginas principales

Buscamos y asociamos las subpáginas de la página principal encontrada

Buscamos y asociamos las subsubpáginas de la subpágina encontrada

Por otra parte, para ayudar al usuario en la búsqueda de elementos se ha añadido un cuadro de texto donde, al igual que en los combos de las pestañas anteriores, se permite la sugerencia.

Para ello, el código sería exactamente igual excepto por la siguiente línea:

```
ContentProposalAdapter adapter = new ContentProposalAdapter(textSugerencia, new
TextContentAdapter(), provider, null, null);
```

Cuando el usuario busca el elemento el árbol se despliega y selecciona la rama correspondiente:

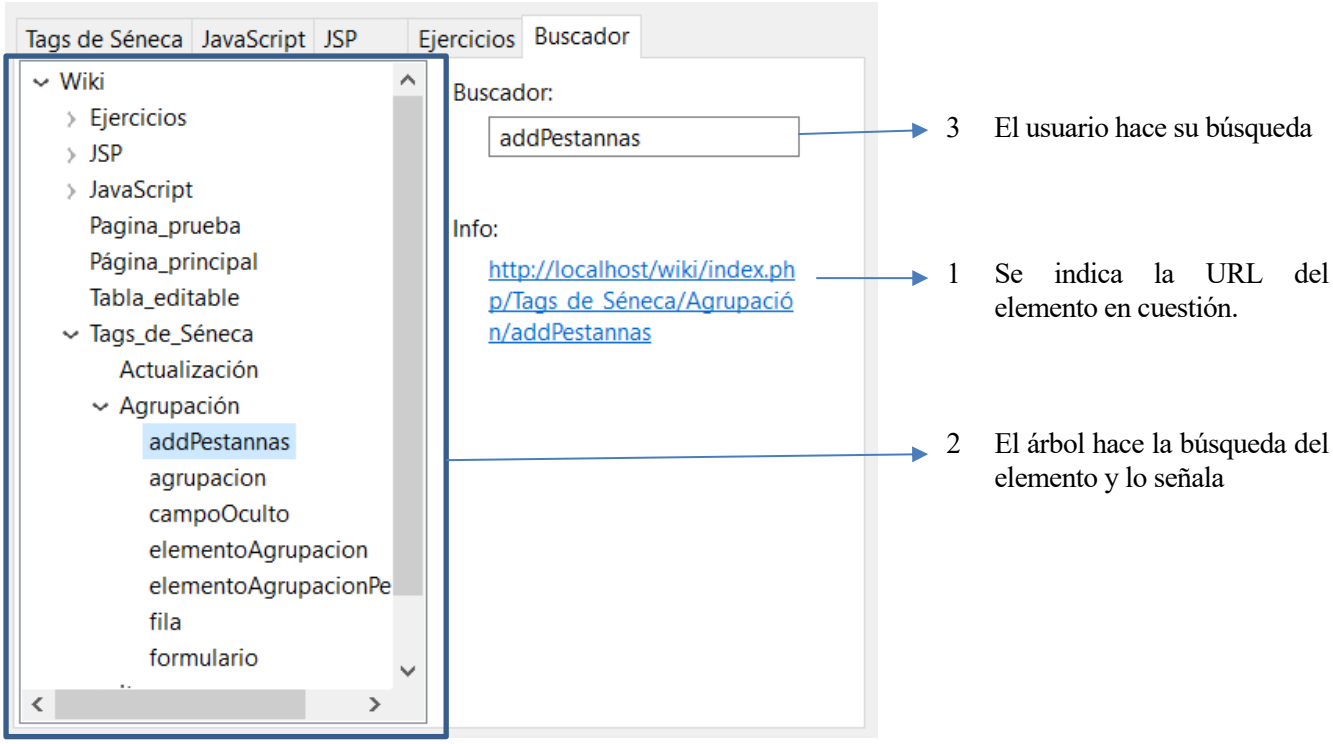


Fig 48 – Proceso de búsqueda

6.4.3.3.1. Búsqueda en el árbol

Para realizar el proceso de búsqueda del elemento en el árbol se ha creado un método llamado "BusquedaArbol" que recorre el árbol hasta que encuentra lo que busca.

Como no podemos saber el tamaño del árbol, se ha diseñado un método recursivo.

```
public void busquedaArbol(Tree tree, TreeItem item, String busqueda) {
		TreeItem aux = null;
		TreeItem[] hijos = item.getItems();
		for (int i = 0; i < hijos.length; i++) {
			if (busqueda.equals(hijos[i].getText())) {
				// Hemos encontrado coincidencia, lo seleccionamos
				tree.select(hijos[i]);
				// Expandimos la rama
				aux = hijos[i];
				while (aux != null) {
					aux = aux.getParentItem();
					if (aux != null)
						aux.setExpanded(true);
				}
				hijos[i].setExpanded(true);
			}
		}
		for (int i = 0; i < hijos.length; i++) {
			busquedaArbol(tree, hijos[i], busqueda);
		}
}
```

6.4.4. MyProposalProvider.java

Como se comentó brevemente, la clase MyProposalProvider ayuda a buscar palabras que contienen el texto añadido por el usuario en alguno de los elementos del formulario de Ventana.java.

En este punto se va a explicar cómo realiza el proceso de búsqueda esta clase.

```
public class MyProposalProvider implements IContentProposalProvider {

	String[] propuestas;
	public MyProposalProvider(String[] propuestas){
		this.propuestas=propuestas;
	}
	public IContentProposal[] getProposals(String contenido, int posicion) {
		List<IContentProposal> propuestas_validas = new
					ArrayList<IContentProposal>();
		for (String prop : propuestas){
			String prop_aux = prop.toLowerCase();
			contenido = contenido.toLowerCase();
			if (prop_aux.contains(contenido)){
				propuestas_validas.add(new ContentProposal(prop));
			}
		}
		return propuestas_validas.toArray(new
IContentProposal[propuestas_validas.size()]);
	}


}
```

El constructor recibe una lista de palabras entre las que buscar.

Lista donde guardar las propuestas que coinciden con el contenido.

Comprueba si el contenido buscado está en cada propuesta y guarda aquellas donde así sea.

* Se comprueba todo en minúsculas porque el método "contains" hace distinción entre masyúsculas y minúsculas.

6.4.5. BusquedaPatrón.java

Esta clase es el módulo de comunicación entre la Wiki y Eclipse. Se encarga de realizar peticiones HTTP a partir de la clase "URL" de Java y lleva a cabo la búsqueda y creación de plantillas que más tarde la clase Editor insertará en el editor de texto activo.

Esta clase, principalmente, tiene 2 métodos:

1. Buscar: Para las páginas "Tags_de_Séneca", "JSP", "JavaScript"… (si en un futuro se añadiesen más del mismo estilo no daría problemas).

```
public String buscar (String pagina, String opcion, int repeticion, String radio)
```

Nombre de la página donde buscar. Por ejemplo, JSP/DoblesListas

Nombre de la página principal

Esto se hace porque Tags_de_Séneca tiene un trato especial.

0 por defecto

Si/No

Null por defecto

2. BuscarEjercicios:

```
public String buscarEjercicios (String pag1, String pag, String fichero)
```

Nombre de la página donde buscar. Por ejemplo, Ejercicios/ejercicio_1

Fichero donde hay que buscar

Se lanza la petición HTTP a la URL y leemos la página html:

```
URL url = new URL("http://localhost/wiki/index.php/"+pagina);
URLConnection uc = url.openConnection();
uc.connect();
```

```
// Leemos la página html: uc.getInputStream()
BufferedReader in = new BufferedReader(new InputStreamReader(uc.getInputStream()));
```

En ambos métodos, cuando tenemos la respuesta html, se procede de la misma forma. Sabemos que las líneas de código de la Wiki siempre deben estar entre etiquetas <pre></pre>, tal y como quedó establecido en las buenas prácticas de la Wiki (Anexo B). En el caso de buscar ejercicios, tenemos que encontrar el código de un fichero concreto, por esta razón, se ha creado un método aparte.

Una vez tenemos el contenido entre esas etiquetas, si se trata de código HTML, habría que diferenciar entre cabecera, cuerpo y pie. La cabecera sería el texto desde el principio del contenido de las etiquetas <pre> hasta , el cuerpo el contenido entre y y, por último, el pie es el contenido entre y la etiqueta </pre>. Recordar que esto se hace así porque el cuerpo se puede repetir tantas veces como el usuario indique.

En el caso de "BusquedaEjercicios", pese a tener código HTML, no se permiten repeticiones, por lo que, no se aplica el procedimiento anterior.

Finalmente, tras limpiar el texto de etiquetas y símbolos de wikitexto es necesario pasar de la codificación UTF-8 (de la Wiki) a la ISO-… (de Eclipse) con la siguiente función: [7]

```
public static String codificar(byte[] UTF8texto){

        Charset utf8charset = Charset.forName("UTF-8");
        Charset iso88591charset = Charset.forName("ISO-8859-1");

        String string = new String ( UTF8texto, utf8charset );

        // "When I do a getbytes(encoding) and "
        byte[] iso88591bytes = string.getBytes(iso88591charset);

        // "then create a new string with the bytes in ISO-8859-1 encoding"
        String info = new String ( iso88591bytes, iso88591charset );

        // "I get a two different chars"

          return info;

                                                                    }
```

6.4.6. BusquedaFicheros.java

La clase BusquedaFicheros funciona exactamente igual que BusquedaPatrón con la diferencia de que busca las etiquetas <h4></h4>, entre las cuales se encuentra el nombre del fichero.

6.4.7. BaseDatos.java

La clase BaseDatos es la que se encarga de las tareas de consulta, inserción o actualización de la base de datos de la Wiki. Para ello, se utiliza el driver **JDBC**, una aplicación Java que permite la comunicación con bases de datos MySQL. [19]

En un plugin el término "importar librerías" se corresponde con "crear dependencias entre plugins". Con esto se quiere decir que para utilizar la librería JDBC de Java hemos tenido que crear un plugin del .jar descargado en java2s [20] y después añadirlo como dependencia de nuestro plugin tal y como se explica en el Anexo C.

6.4.7.1. Conexión con la base de datos

Para poder establecer la conexión con la base de datos se ha creado el método "abrirConexion" y "cerrarConexion". [21]

```
Connection conexion;
Statement s;
```

```
public void abrirConexion(){
try{
   Class.forName("com.mysql.jdbc.Driver");
   // Establecemos la conexión con la base de datos.
   conexion = DriverManager.getConnection ("jdbc:mysql://localhost/wiki","root", "");

   //Para poder viajar por el resultset
   s = conexion.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

} catch (Exception e){
      e.printStackTrace();
}

public void cerrarConexion(){
      try {
         conexion.close();
      } catch (SQLException e) {
            e.printStackTrace();
      }
}
```

6.4.7.2. Consulta a la base de datos

Para realizar consultas a la base de datos [21] se utiliza el método executeQuery de la clase Statement, lo cual nos devolverá un objeto tipo ResulSet:

```
ResultSet resultado_consulta = s.executeQuery (consulta);
while(resultado_consulta.next()){
      descargas = resultado_consulta.getInt(1);
}
```

Si el resultado ResultSet tiene más de una tupla se recorre con un bucle una a una y se obtiene la columna que más nos interesa.

Número de columna de la tupla

6.4.7.3. Inserción y actualización de la base de datos

Tanto para insertar como para actualizar información en una tabla de la base de datos se actúa de la misma forma. [21]

```
INSERT INTO tldescargas (Descargas, Fecha, Categoria,
subCategoria, Elemento) VALUES (?,?,?,?,?)
```

```
PreparedStatement inserccion = conexion.prepareStatement(consulta);
inserccion.setInt(1, descargas);
inserccion.setString(2, fecha);
```

Posición y valor de la "?" de la consulta

6.4.8. Editor.java

La clase Editor es aquella que se encarga de obtener el IDocument del editor activo pasado por la clase Wiki en el constructor. Si recordamos el punto 3.1.3.1. tenemos que entender el editor de texto como un documento en el cual podemos escribir.

```
ITextEditor textEditor = (ITextEditor) editor;
IDocumentProvider dp = textEditor.getDocumentProvider();
IDocument doc = dp.getDocument(editor.getEditorInput());
```

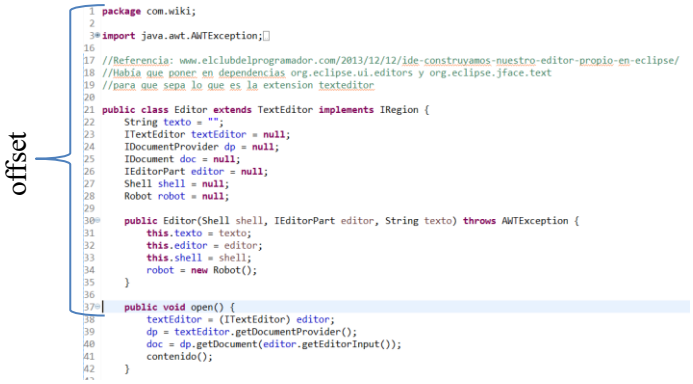
Obtención de la salida del editor de texto como un documento.

5.4.8.1. Escribir en el editor de texto

Nuestra aplicación siempre inserta las plantillas de código en el lugar donde se encuentre el cursor: [10]

```
ITextSelection selection =
(ITextSelection)textEditor.getSelectionProvider().getSelection();
```

Una vez tenemos esa posición, obtenemos el offset de la línea donde queremos escribir:




```
offset=selection.getOffset();
```

Y escribimos:

```
doc.replace(offset, 0, texto+"\n");
```

Fig 49 – offset editor de texto

6.5. Extensión com.estadisticas

La extensión com.estadisticas se añadirá a la barra de menú. Su principal función es generar informes para evaluar el estado y rendimiento de la Wiki. En este capítulo se va a explicar este punto desde el punto de vista del software, mientras que, en el siguiente capítulo se explicarán los informes desde el punto de vista de la gestión.

En el paquete com.estadisticas podemos encontrar las siguientes clases que, al igual que antes, se explicaran los aspectos más relevantes.

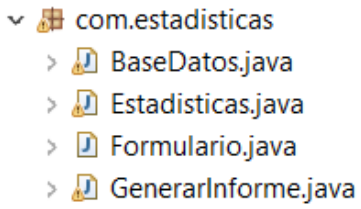


Fig 50 – Clases de la extensión com.estadisticas

6.5.1. Estadísticas.java

La clase Estadísticas funciona exactamente igual que la clase Wiki de la extensión anterior. En este caso, el método run instancia la clase Formulario.

```
public class Estadisticas implements IWorkbenchWindowActionDelegate {
	IWorkbenchWindow activeWindow = null;

	public void run(IAction arg0) {
		/********** Ventana emergente formulario **********/
		Formulario formulario = new Formulario();
		formulario.open();

	 }
}
```

6.5.1. Formulario.java

La clase Formulario genera la parte gráfica de esta extensión y tendrá el siguiente aspecto:

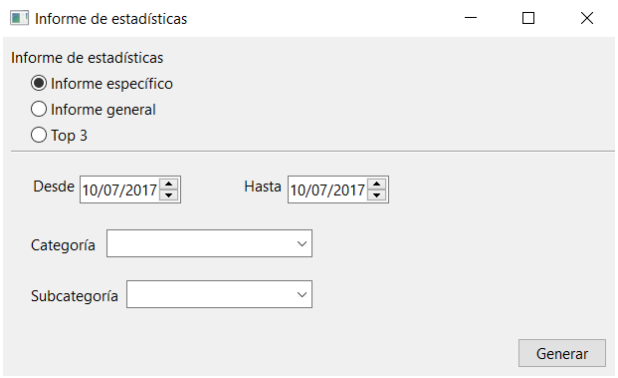


Fig 51 – GUI com.estadisticas

En este caso, el usuario puede elegir entre tres informes distintos:

- Informe específico: nos da las estadísticas en un determinado periodo de tiempo y en una categoría (página principal) o subcategoría (subpágina) determinadas.
- Informe general: da las estadísticas de todas las páginas en todo el periodo de tiempo registrado.
- Informe top 3: genera las estadísticas referentes a las 3 páginas más consultadas, actualizadas, descargadas…

Al igual que en Ventana.java de com.wiki los elementos han sido programados con Java SWT y funcionan de forma similar. En función de la opción que tome el usuario se habilitan unas opciones u otras, los combos se rellenan a partir de consultas a la base de datos…

Cuando el usuario pulsa el botón "Generar" salta un listener que crea un objeto GenerarInforme (GenerarInforme.java) y le pasa los parámetros necesarios para crear el informe seleccionado.

```
Listener listener = new Listener() {
	public void handleEvent(Event arg0) {
		// Al pulsar el botón generar...
		GenerarInforme generar = new GenerarInforme();
		// Parametros para generar el informe --> fecha desde y hasta
		String fech_desde = fecha_inicio.getYear() + "-" + String.format("%02d",
				fecha_inicio.getMonth()+1) + "-"+
				String.format("%02d", fecha_inicio.getDay());
		String fech_hasta = fecha_fin.getYear() + "-" + String.format("%02d", +
				fecha_fin.getMonth()+1) + "-"
			+ String.format("%02d", fecha_fin.getDay());
		String[] parametros = { fech_desde, fech_hasta, combo.getText(),  +
					combo_sub.getText() };
		// parametros=null; //No se necesitan parametros
		generar.generar(opcion, parametros);
		shell.close();
	}
};
```

6.5.2. GenerarInforme.java

La clase GenerarInforme tiene dos objetivos por un lado actualizar la información de la base de datos y, por otro lado, compilar y lanzar los informes.

6.5.2.1. Actualizar información de la Base de Datos

Las tablas que llevan el control de estadísticas son tldescargas, la cual se actualiza al pulsar el botón "Aplicar" o "Importar" de la extensión com.wiki, y tlconsultas, actualizada por esta clase a partir del fichero access.log del servidor Apache. Debido a que el fichero .log tiene un formato conocido, lo recorremos y vamos sacando y guardando la información que nos resulte interesante.

Las tablas e información cada una puede verse en las siguientes imágenes:

IP	Pagina	Padre	Consultas	Fecha 1
127.0.0.1	Tags_de_Séneca/Items/DobleLista	Tags_de_Séneca	1	2017-07-07 15:30:07
127.0.0.1	Tags_de_Séneca/Agrupación/addPestannas	Tags_de_Séneca	1	2017-07-07 15:29:43
127.0.0.1	Tags_de_Séneca/Items/DobleLista	Tags_de_Séneca	1	2017-07-07 15:02:49

Fig 53 - tlconsultas

Descargas	Fecha	Categoria	subCategoria	Elemento
3	2017-07-04	Tags de Séneca	Items	DobleLista
1	2017-07-04	JavaScript	Funciones	mostrarCalendario
1	2017-07-04	JSP	DoblesListas	

Fig 52 - tldescargas

6.5.2.2. Generar el informe

Para generar informes se usa un software libre llamado JasperReports [22], escrito completamente en Java y con una API a disposición del usuario, razón por la que fue la aplicación seleccionada para este proyecto.

En primer lugar, vamos a explicar cómo se generan los informes en JR[21].

6.5.2.2.1. JasperReports

Este software tiene un plugin en Eclipse que podemos obtener directamente de la tienda del mismo. Una vez descargado tendremos una perspectiva como la siguiente para poder generar nuestro informe, que no es más que un fichero XML, de forma gráfica.

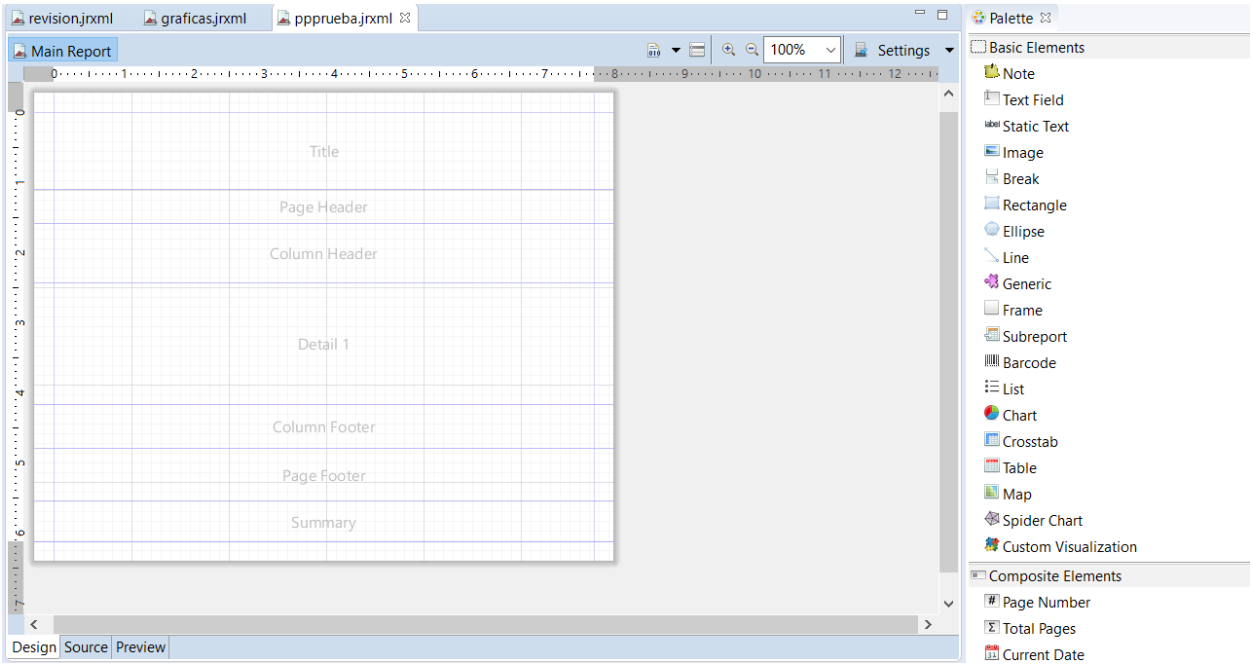


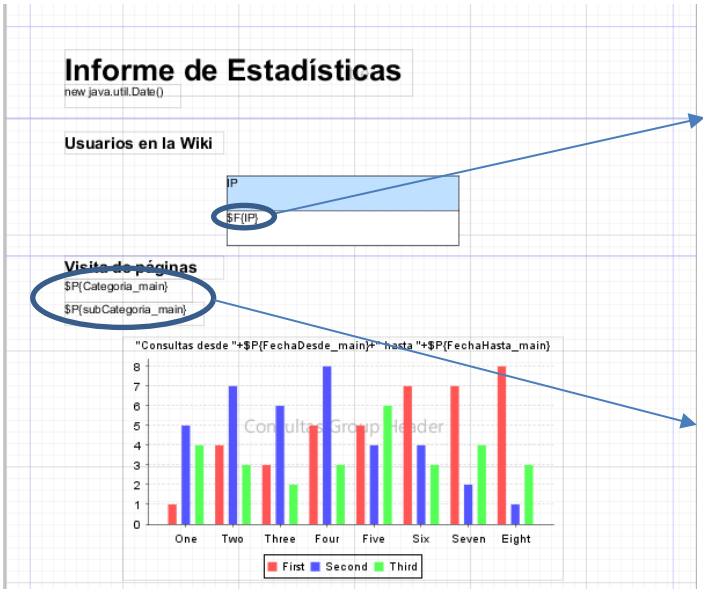
Fig 54 - JasperReports

Puesto que los informes son muy similares entre sí, cambia el tipo de consulta a la base de datos, únicamente se van a explicar los aspectos más relevantes del informe específico ya que es el más complejo.

En primer lugar, hay que entender qué es el dataset. En JasperReports un dataset es un complemento que define los parámetros y "fields" (explicados más abajo) del elemento al que complementa a partir de una consulta SQL (cada columna sería un "field"). El documento, por defecto, tiene un dataset principal y, por otro lado, cada gráfica, tabla, lista… del documento tiene su propio dataset.

Como se explicó anteriormente tenemos dos informes uno para las gráficas y otro con una tabla de actualizaciones y revisiones (tabla tlrevisiones).

[21] JasperReports



Título del informe

$F{IP} Campo **Field:** en JasperReports un campo field es una "variable" que coge su valor de una consulta SQL y genera una fila/valor por tupla recibida.

Los campos Field quedan definidos en el **dataset** de cada elemento.

En este caso, la tabla y la gráfica tienen datasets distintos porque hacen consultas distintas.

$P{Categoria_main}, $P{subCategoria_main}, **Parámetro**: Un parámetro en una "variable" que toma su valor de una fuente externa, en este caso del formulario de la extensión com.estadisticas.

Fig 55 – Informe I

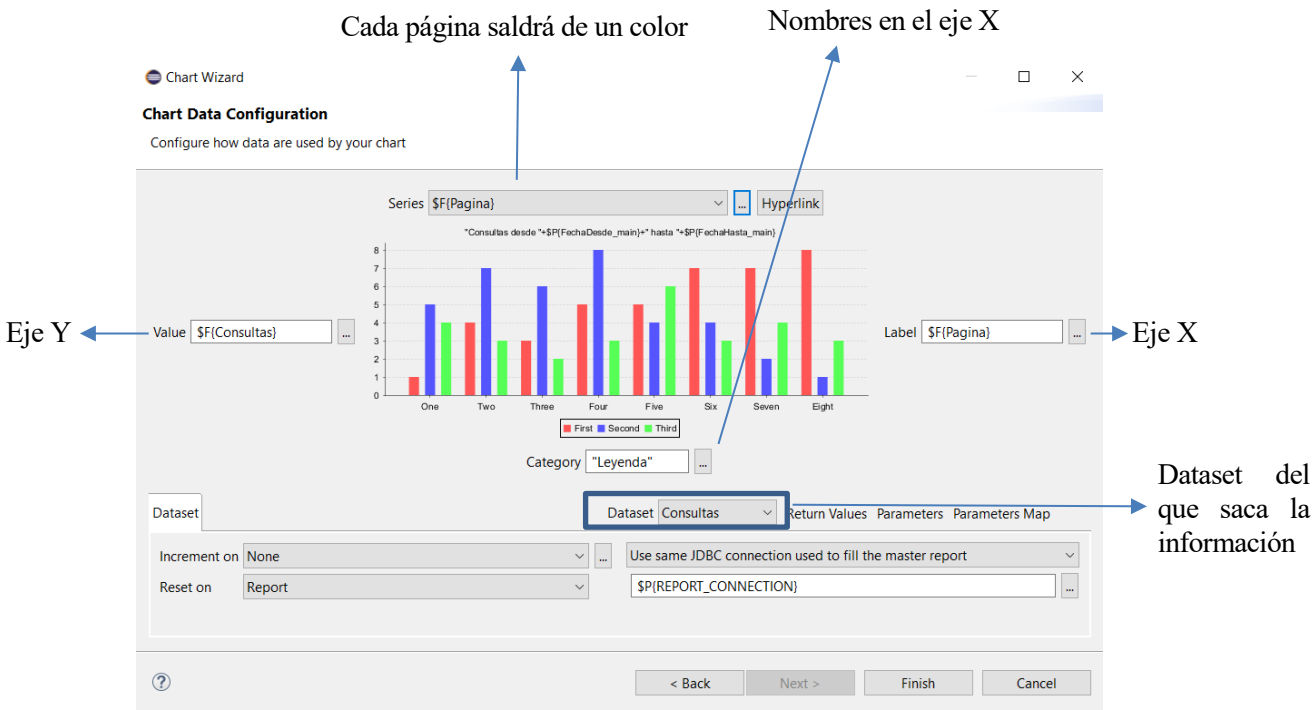
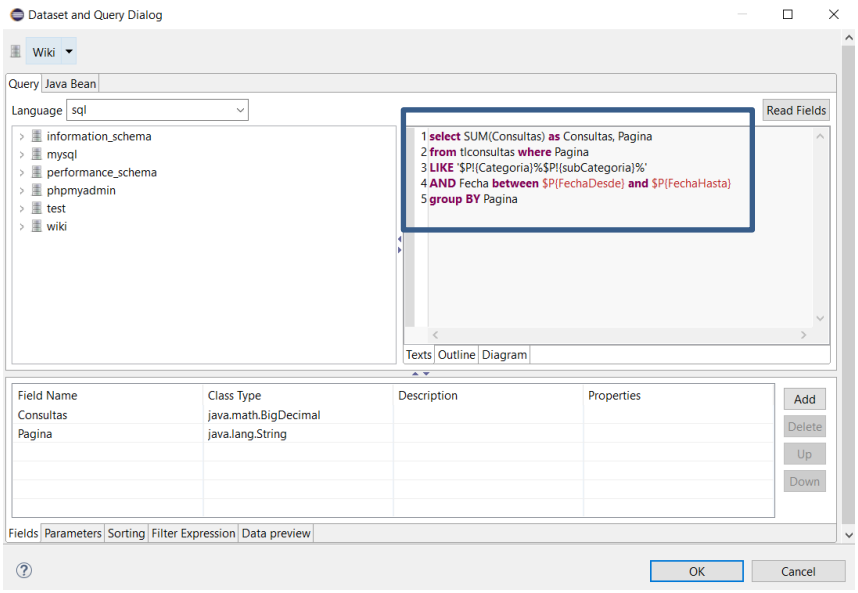


Fig 56 – Informe II



Como puede verse la consulta SQL depende de determinados parámetros. Los parámetros están asociados a su dataset, en este caso, a "Consultas".

Por otro lado, los parámetros se pasan al dataset principal cuando se compila el informe por lo que tenemos que relacionar los parámetros del dataset principal con los del dataset que hereda sus parámetros.

Para hacer esto lo más sencillo es añadir en el código XML las siguientes líneas: [23]

Fig 57 – Informe III, DataSet

Nombre del DataSet que se va a asociar al principal

```
<datasetRun subDataset="Consultas" uuid="2ce54d51-7b36-4611-b7aa-4741a16cae16">
      <datasetParameter name="FechaDesde">
      <datasetParameterExpression><![CDATA[$P{FechaDesde_main}]]>
                                                  </datasetParameterExpression>
      </datasetParameter>
      <datasetParameter name="FechaHasta">
      <datasetParameterExpression><![CDATA[$P{FechaHasta_main}]]>
                                                  </datasetParameterExpression>
      </datasetParameter>
      <connectionExpression><![CDATA[$P{REPORT_CONNECTION}]]></connectionExpression>
</datasetRun>
```

Parámetro del DataSet consultas

Parámetro del DataSet principal al que se asocia

6.5.2.2.2. JasperReports en Java

Cuando desarrollas un informe de este tipo se genera un fichero de extensión jrxml que cuando se compila crea un fichero de extensión jasper.

La API de este software permite, a partir de código Java, compilar los ficheros jrxml, pasarle los parámetros necesarios y abrirlos en formato pdf. Se haría de la siguiente forma: [24]

```
JasperReport report_graficas = JasperCompileManager
      .compileReport("C:/Users/elisa/workspace/com.wiki/report/especifico/todos_pagP
rin/graficas.jrxml");
JasperReport report_revision =  JasperCompileManager
.compileReport("C:/Users/elisa/workspace/com.wiki/report/especifico//todos_pagPrin/re
vision.jrxml");
```

Crea el .jasper del informe

```
int totalPlantillas = BD.consultar_plantillas("SELECT SUM(Descargas) FROM tldescargas
WHERE Fecha BETWEEN '"+parametros_formulario[0]+"' AND '"+parametros_formulario[1]+"'
AND Categoria like '"+parametros_formulario[2]+"' AND subCategoria like
'"+parametros_formulario[3]+"'");
```

```
HashMap<String, Object> parametros = new HashMap<String, Object>();
parametros.put("NumPlantillas", totalPlantillas);
parametros.put("FechaDesde_main", parametros_formulario[0]);
parametros.put("FechaHasta_main", parametros_formulario[1]);
parametros.put("Categoria_main", parametros_formulario[2]);
parametros.put("subCategoria_main", parametros_formulario[3]);

JasperPrint print_graficas = JasperFillManager.fillReport(report_graficas,
                                                          parametros, conexion);
JasperPrint print_revision = JasperFillManager.fillReport(report_revision,
                                                          parametros, conexion);

File informe = new File(directorio_seleccionado+"/informe_general.pdf");
File informe2 = new File(directorio_seleccionado+"/informe_general2.pdf");
OutputStream salida = new FileOutputStream(informe);
JasperExportManager.exportReportToPdfStream(print_graficas, salida);
OutputStream salida2 = new FileOutputStream(informe2);
JasperExportManager.exportReportToPdfStream(print_revision, salida2);
salida.close();
salida2.close();
Desktop.getDesktop().open(informe);
Desktop.getDesktop().open(informe2);
```

Pasa los parámetros al informe

Exporta el fichero .jasper como PDF

Abre automáticamente el PDF

6.6. Extensión com.url

El paquete com.url nos permite acceder a la página principal de la Wiki a través del navegador por defecto del equipo del usuario.

Esto se consigue con la siguiente línea: [25]

```
Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + direccion);
```

6.7. plugin.xml

Como se dijo anteriormente, este fichero describe cómo un plugin amplía la plataforma Eclipse. En nuestro caso, queremos incluir dos botones de acceso para la Wiki, uno en la barra de herramientas y otro en la barra de menú tal y como se muestra en la siguiente imagen.

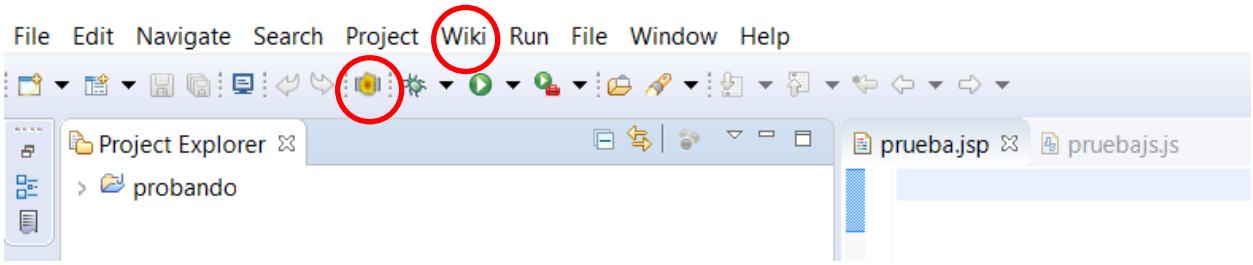
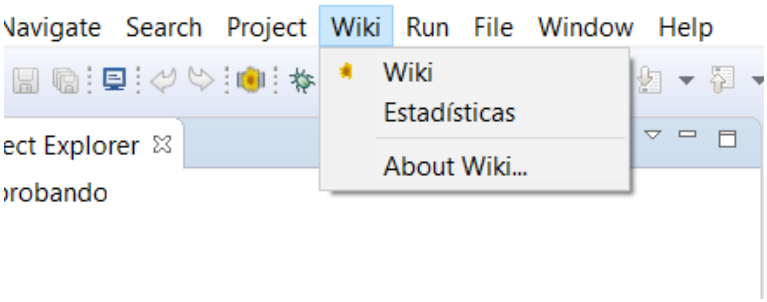


Fig 58 – Ubicación del acceso a la Wiki en Eclipse



Además de la extensión de acceso a Wiki en la barra de menú añadiremos la extensión de estadísticas y la extensión "About Wiki…" que nos lleva directamente a la página principal de nuestra Wiki.

Fig 59 – Barra de menú

Para conseguir esto, utilizaremos el punto de extensión "actionSets" del workbench de Eclipse (véase punto 3.2.) quedándonos el siguiente fichero:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.eclipse.ui.actionSets">

         <actionSet
            id="com.wiki.WikiActionSet"
            label="wiki"
          visible="true"
          description="Action set para el plug-in de la wiki">

          <menu
            id="com.wiki.WikiMenu"
            label="Wiki">
            <add name="estadisticas"/>
            <separator name="wiki"/>
            <separator name="url"/>
          </menu>

           <!-- label: Nombre que ve el usuario -->
           <!-- tooltip: Texto que aparece cuando pones el boton encima -->
           <!-- icon="/icono/cabans-hello-world.jpg"  icono, tamaño 16x16 -->
           <!-- menubarpath: Elementos a añadir en la barra de menú -->

            <action id="com.wiki.WikiAction"
            menubarPath="com.wiki.WikiMenu/wiki"
            label="Estadísticas"
            class="com.estadisticas.Estadisticas"/>

            <action id="com.wiki.WikiAction"
            toolbarPath="Normal"
            menubarPath="com.wiki.WikiMenu/wiki"
            label="Wiki"
            tooltip="Wiki"
            icon="/icono/icono.ico"
            class="com.wiki.Wiki"/>

            <action id="com.wiki.WikiAction"
            menubarPath="com.wiki.WikiMenu/url"
            label="About Wiki..."
            class="com.url.abrirWiki"/>

      </actionSet>
```

```
   </extension>
   <extension
         point="org.eclipse.ui.editorActions">
   </extension>
   <extension
         point="org.eclipse.ui.editors">
   </extension>
</plugin>
```

7. CONCLUSIÓN

El verdadero progreso es el que pone la tecnología al alcance de todos.

- Henry Ford -

Volviendo la vista atrás, observando los errores, los puntos donde el tiempo no se gestionó adecuadamente, las decisiones tomadas… me quedo con todo lo aprendido.

La idea de mejorar la gestión del tiempo y la información de una empresa me pareció tan atractiva que tenía que funcionar. Aunque nadie supiese cómo.

Hacer un plugin en Eclipse ha supuesto un gran reto por la falta de información que no me esperaba. Sin duda, estoy muy orgullosa del resultado y del conocimiento que ahora tengo de un IDE como Eclipse. Asimismo, tengo que resaltar la cantidad de conceptos que he aprendido y el desarrollo de interfaces gráficas. Área que durante la carrera no se ve.

Tras mucho tiempo de planteamiento, de intentos, de buscar una solución puedo decir que el trabajo de investigación y la toma de decisiones ha sido una de las mejores experiencias que he tenido en la carrera. Aplicar los conocimientos y la capacidad de resolución adquirida durante estos cuatro años.

Por otra parte, además de los conocimientos adquiridos, queda la satisfacción al terminar el proyecto y haber superado los momentos de desconfianza.

7.1. Mejoras

Sin duda es un trabajo con un alcance muy amplio y, por supuesto, se pueden añadir muchas mejoras.

Vista: Una de las últimas ideas que tuve fue que sería muchísimo mejor haber diseñado una vista en lugar de una ventana emergente a partir de un botón. El diseño y funcionamiento sería el mismo y únicamente cambiaría el punto de extensión donde se añaden las extensiones.

Mayor automatización: El proyecto ha sido automatizado en su mayor parte. Sin embargo, podría ser más automatizado añadiendo la página de la Wiki en el proceso de instalación.

REFERENCIAS

[1] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Wiki.

[2] MediaWiki, «MediaWiki,» 2001. [En línea]. Available: https://www.mediawiki.org/wiki/MediaWiki/es. [Último acceso: 07 05 2017].

[3] Apache, «apachefriends,» [En línea]. Available: https://www.apachefriends.org/es/index.html.

[4] phpmyadmin, «https://www.phpmyadmin.net/,» [En línea]. Available: https://www.phpmyadmin.net/.

[5] Eclipse, «https://eclipse.org/,» [En línea]. Available: https://eclipse.org/.

[6] JasperReport, «Community.jaspersoft.com,» [En línea]. Available: http://community.jaspersoft.com/project/jasperreports-library.

[7] Micaela, «http://www.elwebmaster.com,» [En línea]. Available: http://www.elwebmaster.com/articulos/los-10-mejores-sitios-para-crear-tu-propia-wiki.

[8] techopedia, «techopedia,» [En línea]. Available: https://www.techopedia.com/definition/26102/java-swing.

[9] Vogella, «vogella,» [En línea]. Available: http://www.vogella.com/tutorials/SWT/article.html#swt-overview.

[10] Eclipse, «API Eclipse,» [En línea]. Available: http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fapi%2Findex.html&overview-summary.html.

[11] E. Wiki, «wiki.eclipse,» [En línea]. Available: https://wiki.eclipse.org/FAQ_What_are_extensions_and_extension_points%3F.

[12] Mediawiki, «Mediawiki_subpages,» [En línea]. Available: https://www.mediawiki.org/wiki/Help:Subpages/es.

[13] mediawiki-wikitext, «mediawiki,» [En línea]. Available: https://www.mediawiki.org/wiki/Wikitext/es.

[14] meta.wikimedia, «meta.wikimedia,» [En línea]. Available: https://meta.wikimedia.org/wiki/Help:Wikitext_examples.

[15] wiki.en.it-processmaps, «wiki.en.it-processmaps,» [En línea]. Available: https://wiki.en.it-processmaps.com/index.php/Knowledge_Management.

[16] Anonimo, «java2s,» [En línea]. Available: http://www.java2s.com/Code/Java/SWT-JFace-

Eclipse/CatalogSWT-JFace-Eclipse.htm.

[17] informit, «Lista_event_listener,» [En línea]. Available: http://www.informit.com/articles/article.aspx?p=354574&seqNum=3.

[18] u. tkotisis, «Stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/13313410/is-eclipse-autocompletefield-only-suggesting-entries-starting-with-entered-text.

[19] mysql, «dev.mysql,» [En línea]. Available: https://dev.mysql.com/downloads/connector/j/.

[20] java2s/library, «java2s,» [En línea]. Available: http://www.java2s.com/Code/Jar/CatalogJar.htm.

[21] anonimo, «tutorialspoint,» [En línea]. Available: https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm.

[22] JasperReports, «Jaspersoft,» [En línea]. Available: http://community.jaspersoft.com/project/jasperreports-library.

[23] R. A. Díaz-Heredero, «adictosaltrabajo,» [En línea]. Available: http://www.adictosaltrabajo.com/tutoriales/jasper-table-component/.

[24] Anonimo, «tutorialspoint,» [En línea]. Available: http://www.tutorialspoint.com/jasper_reports/jasper_create_subreports.htm.

[25] A. Vilches, «albertovilches,» [En línea]. Available: albertovilches.com/abrir-una-url-en-un-navegador-desde-java.

[26] mediawiki, «mediawiki/wiki/instalación,» [En línea]. Available: https://www.mediawiki.org/wiki/Manual:Running_MediaWiki_on_Windows.

[27] Anonimo. [En línea]. Available: http://appwebivan.blogspot.com.es/2012/12/insertar-imagen-en-mediawiki.html.

[28] vogella-tutorials, «vogella,» [En línea]. Available: http://www.vogella.com/tutorials/EclipseJarToPlugin/article.html.

GLOSARIO

Orden alfabético

ANEXO A – INSTALACIÓN Y CONFIGURACIÓN DE MEDIAWIKI

1. Instalación y configuración por defecto

- **Instalación de requisitos del sistema:**

Tal y como podemos encontrar en la documentación oficial de Mediawiki [26] para el correcto funcionamiento de Mediawiki necesitamos:

1. Un Servidor Web, se recomienda el uso de Apache o ISS.
2. PHP versión 5.5.9 o superiores.
3. Una base de datos MySQL, MariaDB, PostgreSQL o SQLite.

Aunque no es una configuración muy compleja, la correcta instalación y configuración de Apache+PHP+MySQL puede ser muy engorrosa. Por ello, personalmente recomiendo instalar **XAMPP**[22] desde la página oficial https://www.apachefriends.org/index.html.

- **Descarga de MediaWiki**

Una vez que XAMPP ha sido instalado descargamos el archivo de MediaWiki [26] y descomprimimos en el directorio raíz del servidor web.

Una vez que ha sido descomprimido cambiamos el nombre del directorio por aquel que queremos que aparezca en la dirección URL.

En mi caso, le he dado al directorio el nombre wiki, por lo que mi URL sería http://ejemplo/wiki/index.php

Fig 60 – Directorio de descarga MediaWiki

[22] XAMPP: Distribución de Apache gratuita y fácil de instalar que contiene MariaDB, PHP, Perl, Apache.

- **Instalación de MediaWiki**

En primer lugar, accedemos al panel de XAMPP e inicializamos el servidor web Apache y la base de datos MySQL.

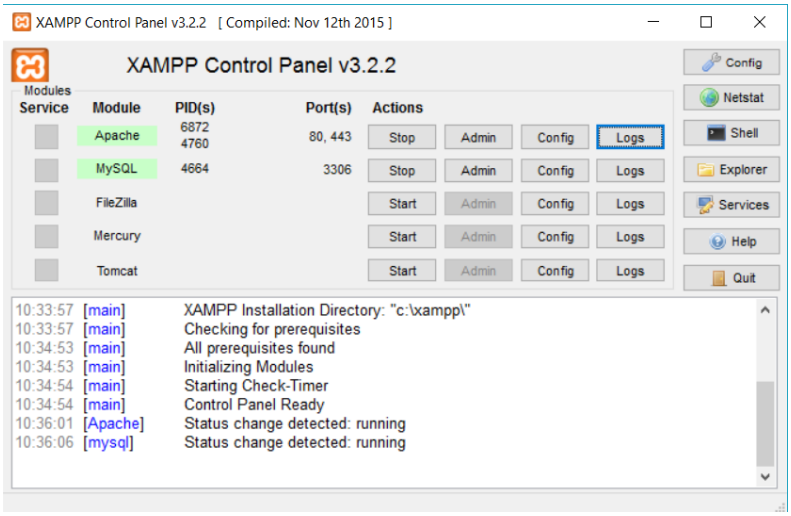


Fig 61 – Panel de control XAMPP

A continuación, nos dirigimos a nuestro navegador y accedemos a la URL http://localhost/wiki/ donde nos aparecerá la siguiente pantalla con todos los pasos para la instalación.

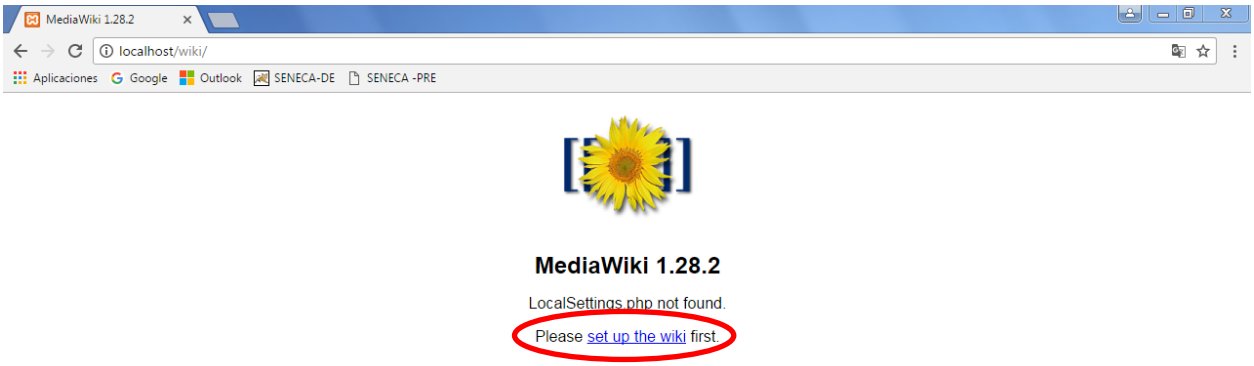


Fig 62 – Página principal de instalación MediaWiki

1. Configuración del idioma de nuestra Wiki

2. Configuración de la Base de Datos

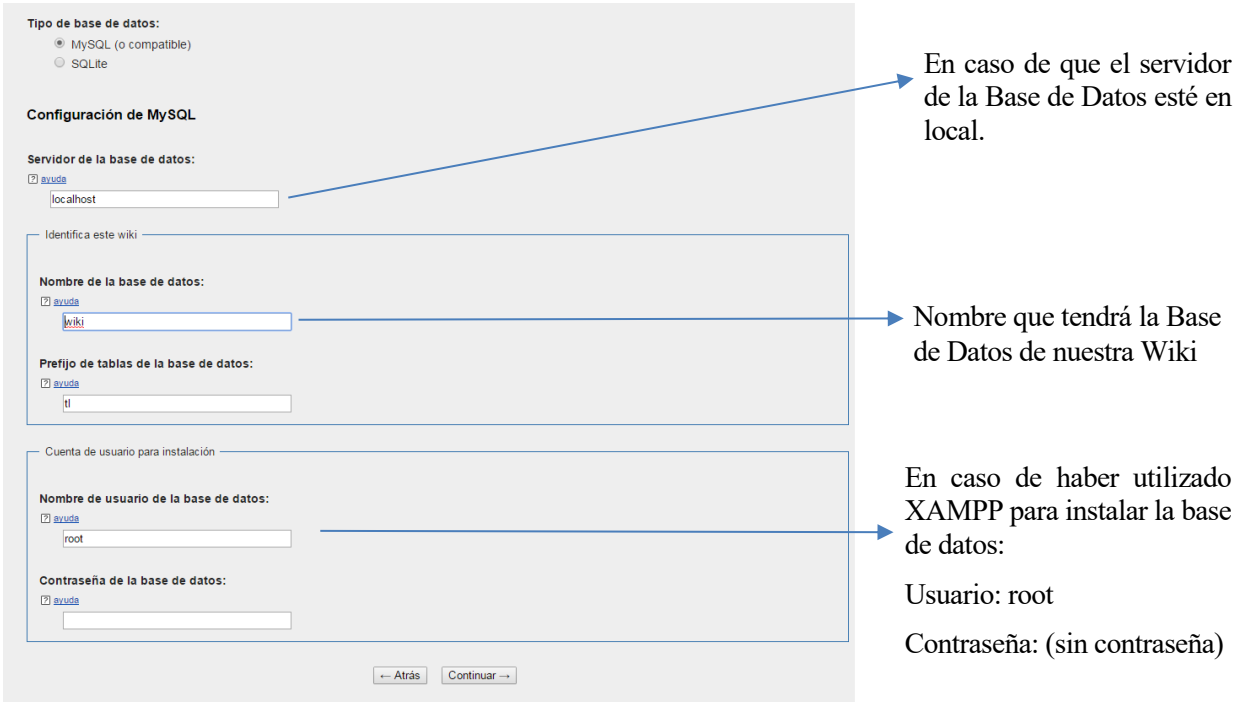


Fig 63 – Configuración de la Base de Datos MediWiki

3. Caracteres de la base de datos con codificación UTF-8

Fig 64 – Codificación de la Base de Datos MediaWiki

4. Configuración de la Wiki

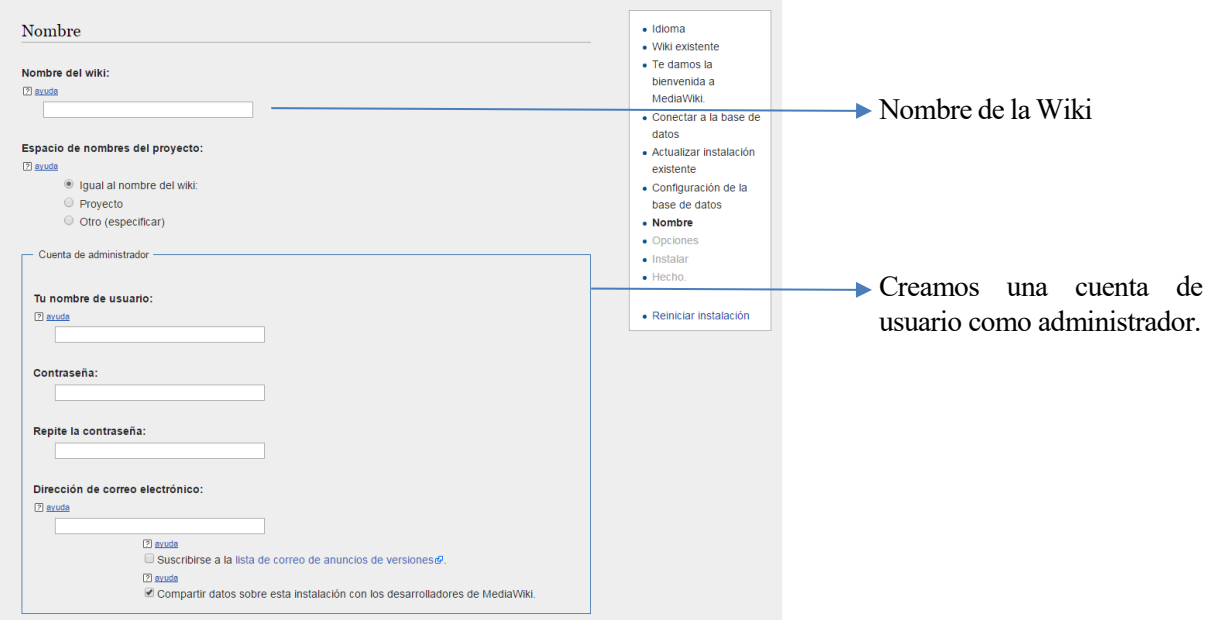


Fig 65 – Configuración final MediaWiki

5. Finalmente, MediaWiki estará instalada con las configuraciones por defecto.

En esta última ventana nos indica que descarguemos el fichero LocalSettings.php en el directorio de instalación de la nuestra wiki.

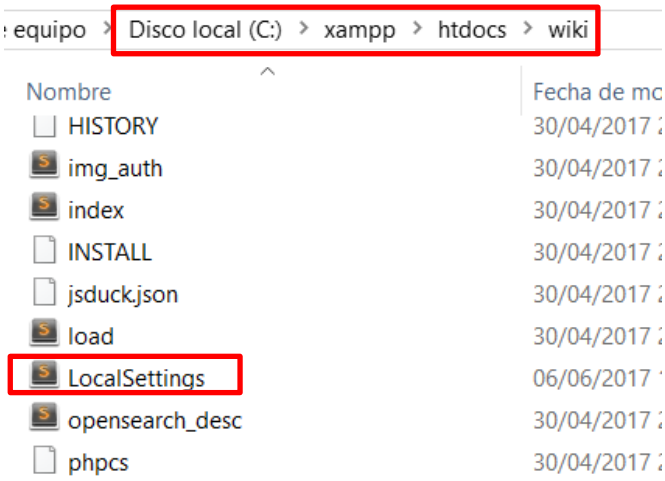


Fig 66 – Ruta destino fichero LocalSetting.php

2. Otras configuraciones (Imprescindibles)

Para un mejor funcionamiento de MediaWiki se recomienda añadir una serie de mejoras sobre el fichero LocalSettings.php

- **Subir imágenes a nuestra Wiki** [27]

Para subir y utilizar imágenes en la Wiki tenemos que habilitar la opción correspondiente. Para ello, en el fichero LocalSettings.php ponemos la siguiente variable a true:

```
$wgEnableUploads = true;
```

- **Subpáginas**

Para poder usar subpáginas tenemos que añadir las siguientes líneas al final del fichero LocalSettings.php.

```
# Enable subpages in the main namespace
$wgNamespacesWithSubpages[NS_MAIN] = true;
# Enable subpages in the template namespace
$wgNamespacesWithSubpages[NS_TEMPLATE] = true;
```

- **Editor Wiki**

El editor por defecto para escribir nuestro contenido en la wiki es muy pobre, por ello se recomienda instalar una extensión llamada "wikiEditor" con muchas más funcionalidades.

1. Nos descargamos la extensión de la página principal https://www.mediawiki.org/wiki/Special:ExtensionDistributor/WikiEditor dentro del directorio extensiones de nuestra wiki.

Fig 67 – Ruta del directorio de extensiones

2. Añadimos la siguiente línea en el fichero LocalSettings.php

```
WfLoadExtension('WikiEditor');
```

ANEXO B – BUENAS PRÁCTICAS EN LA WIKI

Para velar por el buen funcionamiento de la integración de la Wiki en Eclipse se deben cumplir y respetar una serie de normas tanto de edición como de estructura.

Estas buenas prácticas serán recogidas tanto en este documento como en la página principal de la Wiki.

1. Subpáginas

Página_principal/subpágina/subsubpágina

[12] Las barras inclinadas (/) dentro de un nombre de página dividen la página entre padre y subpáginas, recursivamente. Por ejemplo, si tenemos una página padre llamada "Tags de séneca" y queremos crear una nueva página hijo (subpágina) llamada "Agrupación" tenemos que crear una nueva página con el nombre "Tags de Séneca/Agrupación". De la misma forma, si queremos crear una subsubpágina llamada "formulario" habría que nombrar a esta nueva página como "Tags de Séneca/Agrupación/formulario" y así sucesivamente.

REGLA: Como máximo se puede crear una subsubpágina para no hacer jerarquías muy grandes (Ruta máxima: pag1/pag2/pag3).

2. Inserción de código Fuente

REGLA: En una página principal no puede escribirse código fuente

Cuando se escriba código en una página se escribirá entre etiquetas <pre> de la siguiente forma:

<pre>

Esto es un ejemplo de código.

</pre>

Y tendremos por resultado:

```
Esto es un ejemplo de código.
```

2.1. Excepción: Código HTML

En el caso de insertar código HTML, tendremos que hacerlo de la siguiente forma:

REGLA: Siempre hay que poner las etiquetas en el código HTML, aunque no haya elementos que se repitan.

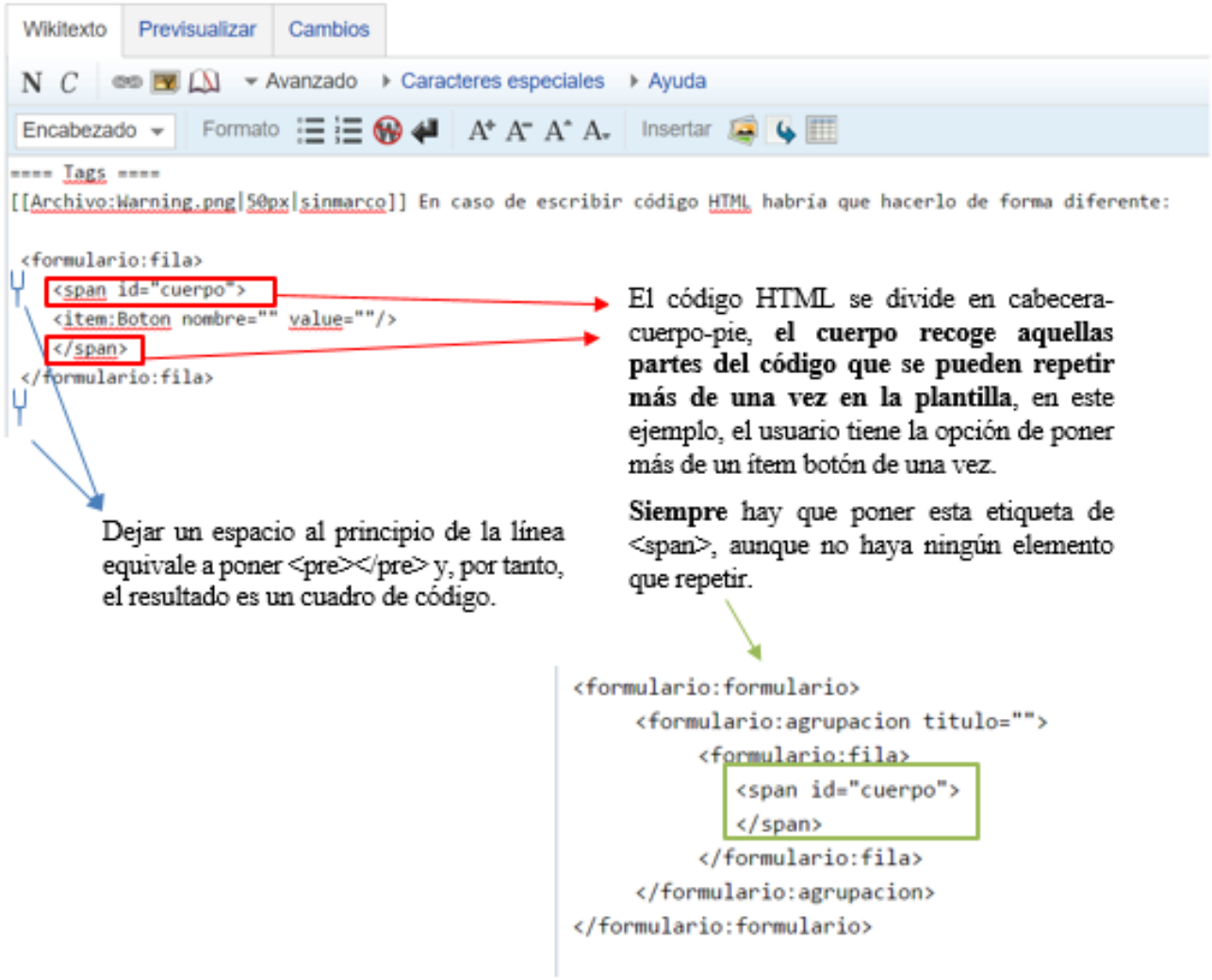


Fig 68 – Contenido HTML en la Wiki

El ejemplo anterior resulta en:

```
<formulario:fila>

   <item:Boton nombre="" value=""/>

</formulario:fila>
```

3. Comentarios en JSP o Java

En el caso de escribir comentarios propios de tecnologías JSP, Java... es imprescindible para el correcto funcionamiento de la Wiki en Eclipse que se abran y cierren con "/*" y "*/" respectivamente.

/* Esto sería un comentario */

ANEXO C – CREACIÓN DE UN PLUGIN A PARTIR DE UNA LIBRERÍA JAR

Referencia [28]

1. Creamos un nuevo proyecto en Eclipse siguiendo la ruta File>New>Project donde aparecerá la siguiente ventana y seleccionamos "Plug-in from Existing JAR Archives".

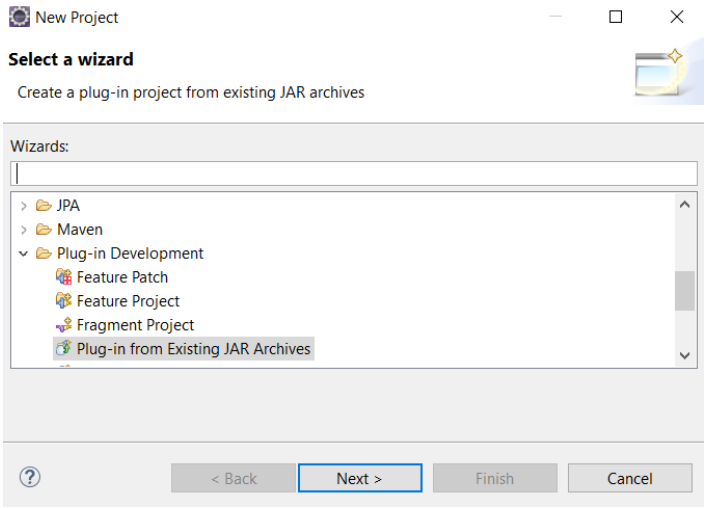


Fig 69 – Plug-in from Existing JAR Archives

2. Cuando damos a Next nos lleva a una pantalla donde añadir el .jar de la librería que queremos convertir en plugin.

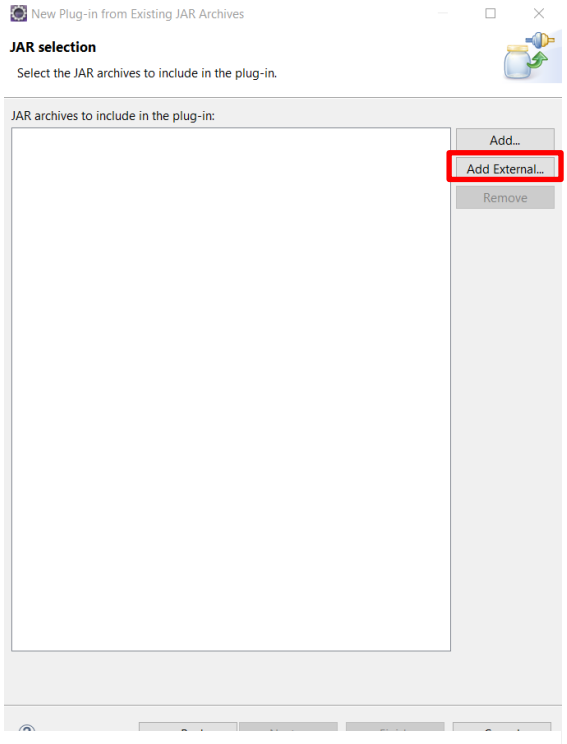
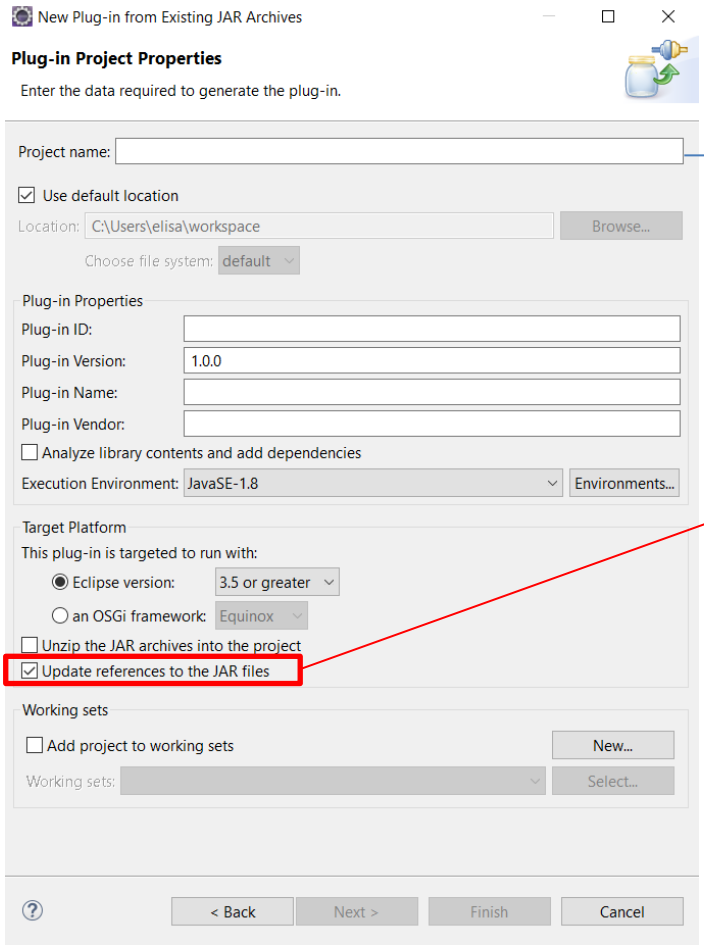


Fig 70 – Crear un plugin de un jar I

3. Configuramos las propiedades de nuestro proyecto:



Le damos el nombre que queramos, por ejemplo, com.jasper

Importante:

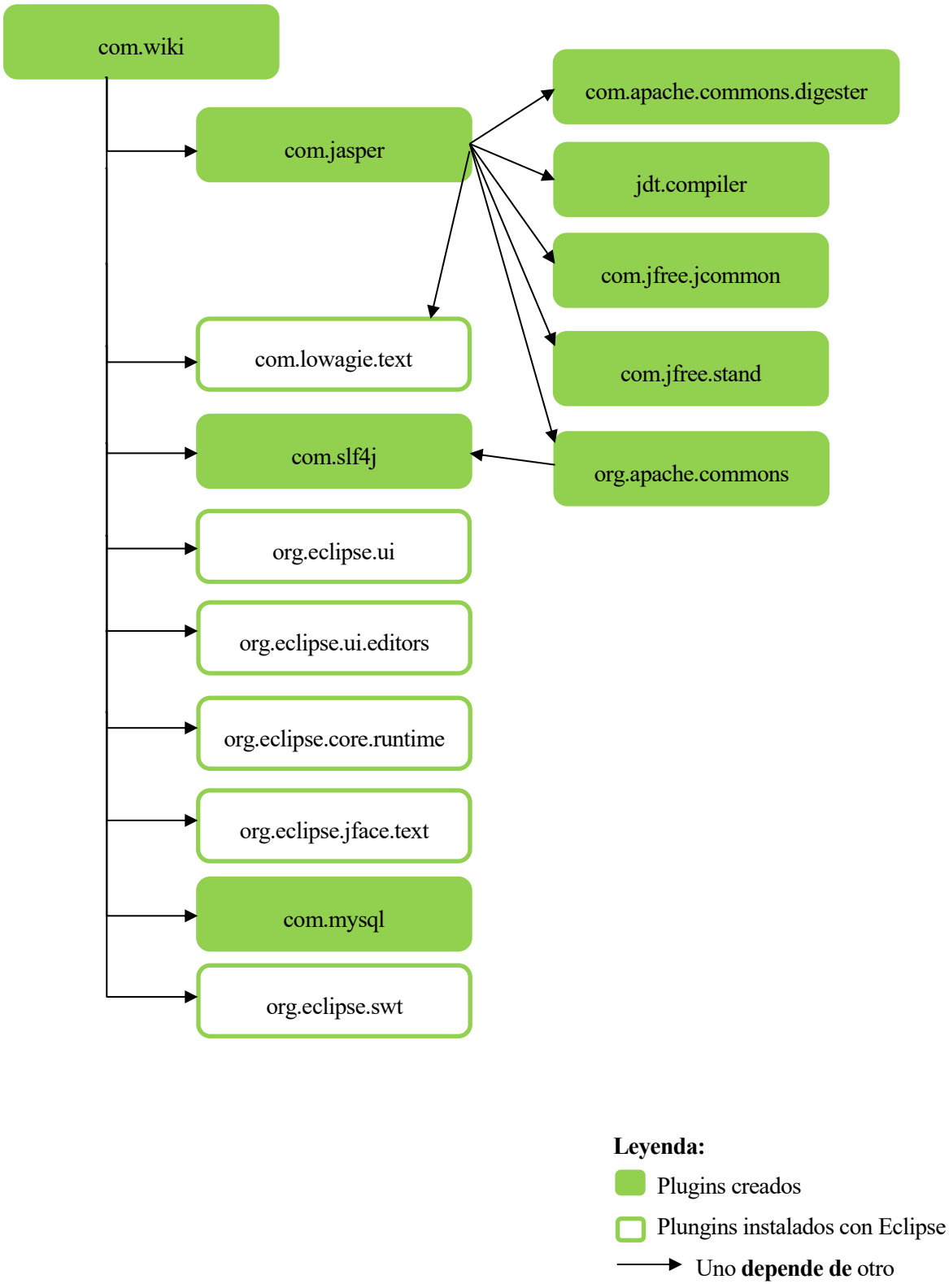
Si has añadido el .jar directamente hay que marcar esta opción.

En el caso de subir el .zip marcamos la primera opción.

Fig 71 - Crear un plugin de un jar II

4. Una vez terminados estos pasos, ya tendremos nuestro plugin creado y podremos crear dependencias entre plugins y, así, usar las diferentes clases y métodos de cada uno.

ANEXO D – RELACIÓN ENTRE PLUGINS



ANEXO E – DIAGRAMAS DE CLASES

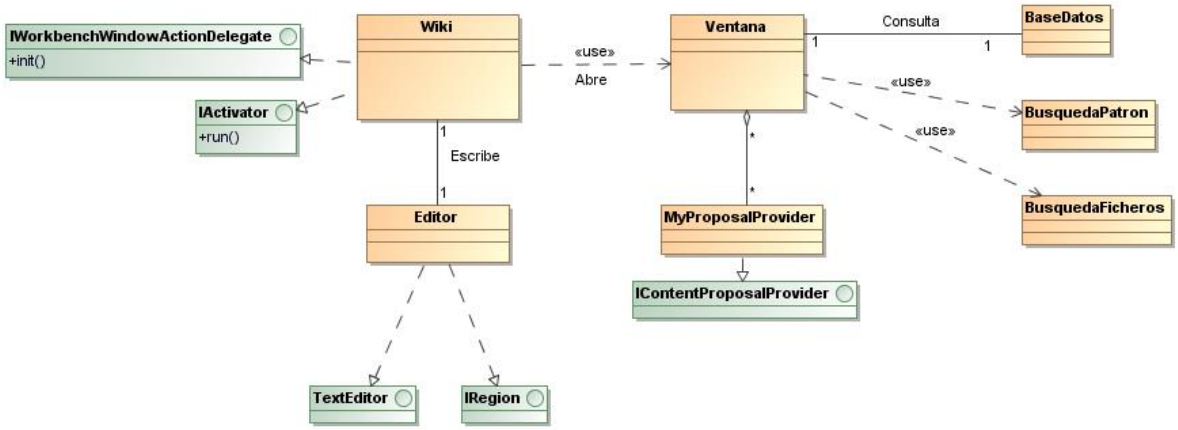


Fig 72 – Diagrama de clases Wiki

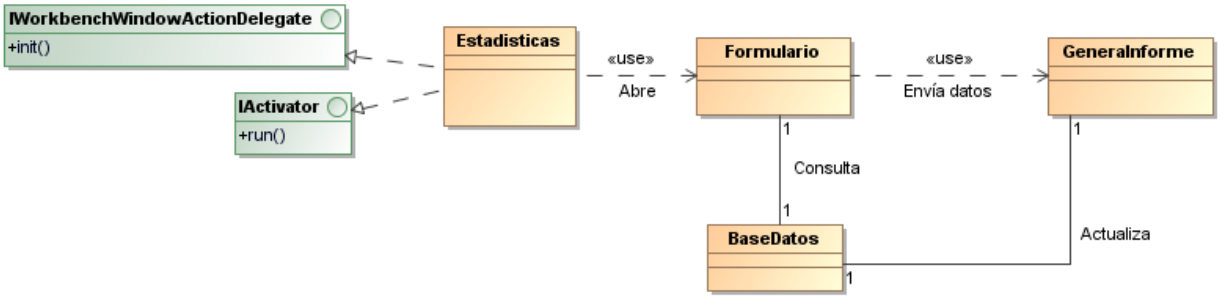


Fig 73 – Diagrama de clases Estadísticas

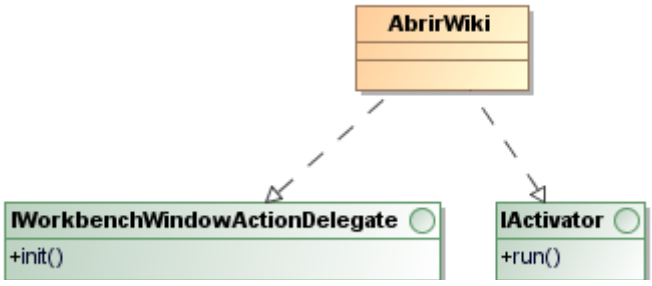


Fig 74 – Diagrama de clases AbrirWiki

ANEXO F – INSTALACIÓN DEL PLUGIN

1. Paquete de instalación

Debido a que nuestro plugin tiene muchas dependecias con otros plugin que a priori no se encuentran en el equipo creemos que lo major es exportar un proyecto "feature project" y hacer un fichero zip con el resultado. Para entenderlo mejor, vamos a explicar el paso a paso:

1. Creamos un nuevo proyecto "feature" File>New>Feature Project.

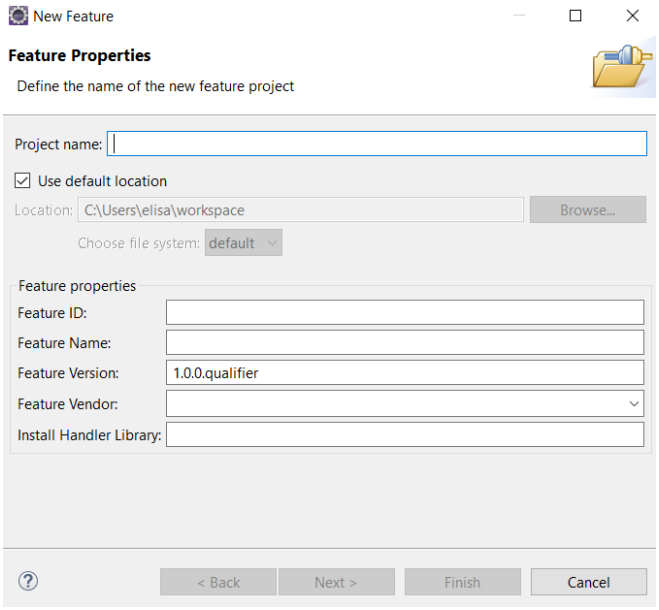


Fig 75 – Feature Project

2. Seleccionamos nuestro plugin "com.wiki" como inicial.

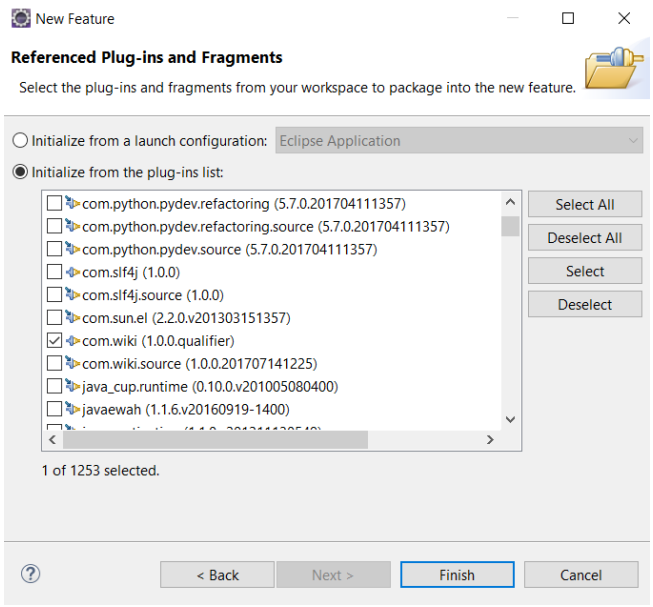


Fig 76 – Feature Project I

3. Seleccionamos todas las dependencias de nuestro plugin con el resto.

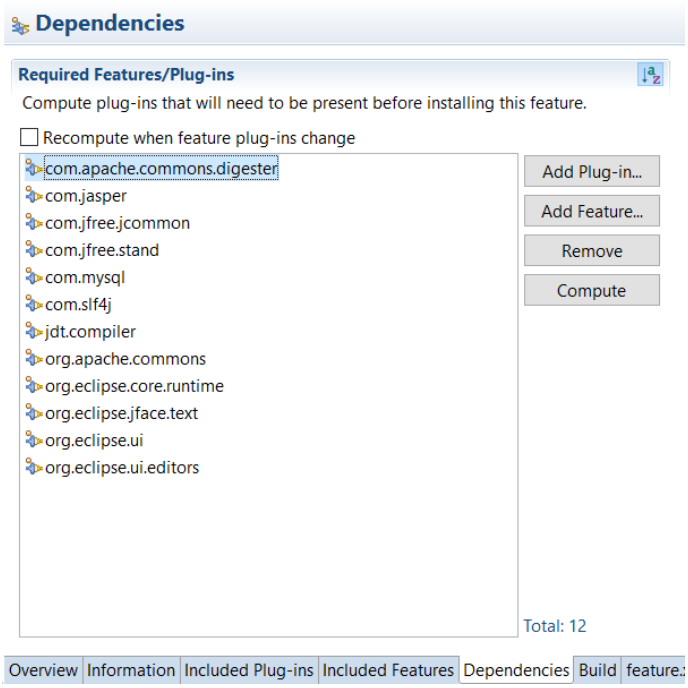


Fig 77 – Feature Project II

4. Seguidamente creamos un proyecto “Update Site Project” en la ruta File>New>Other>

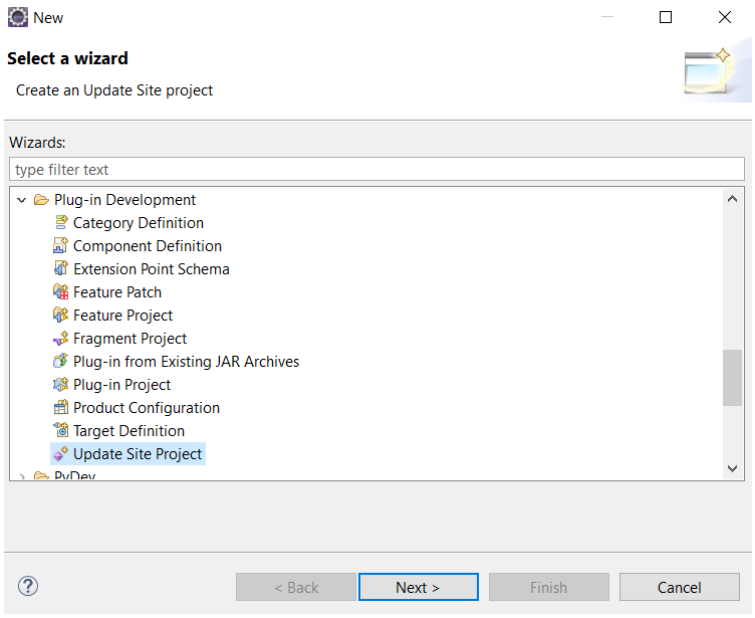
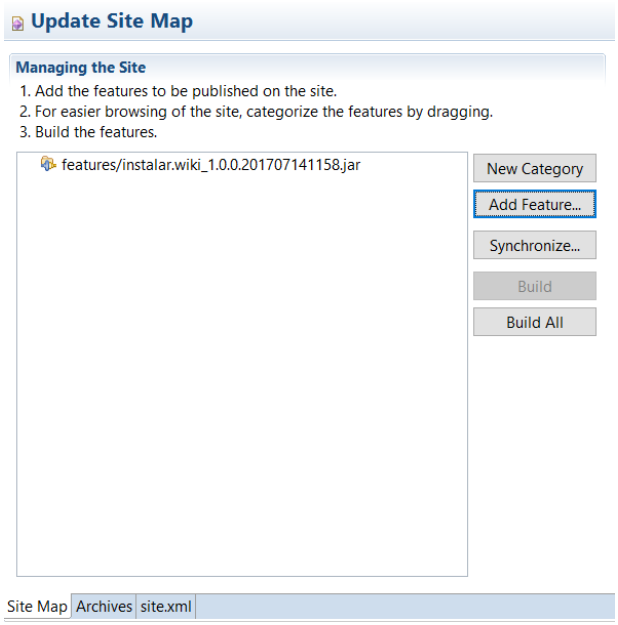


Fig 78 – Update Site Project I

5. Una vez creado y con un nombre nos aparecerá la pantalla de la siguiente figura. Una vez ahí, pulsamos “Add Feature…” y buscamos el proyecto creado anteriormente. [Fig 79]
6. Después pulsamos “Build All” y se crearán una sería de carpetas y ficheros de extensión jar [Fig 80]
7. Con los elementos de la figura 80 hacemos un paquete de extensión zip y tendremos nuestro paquete de instalación montado.



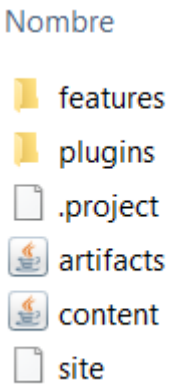
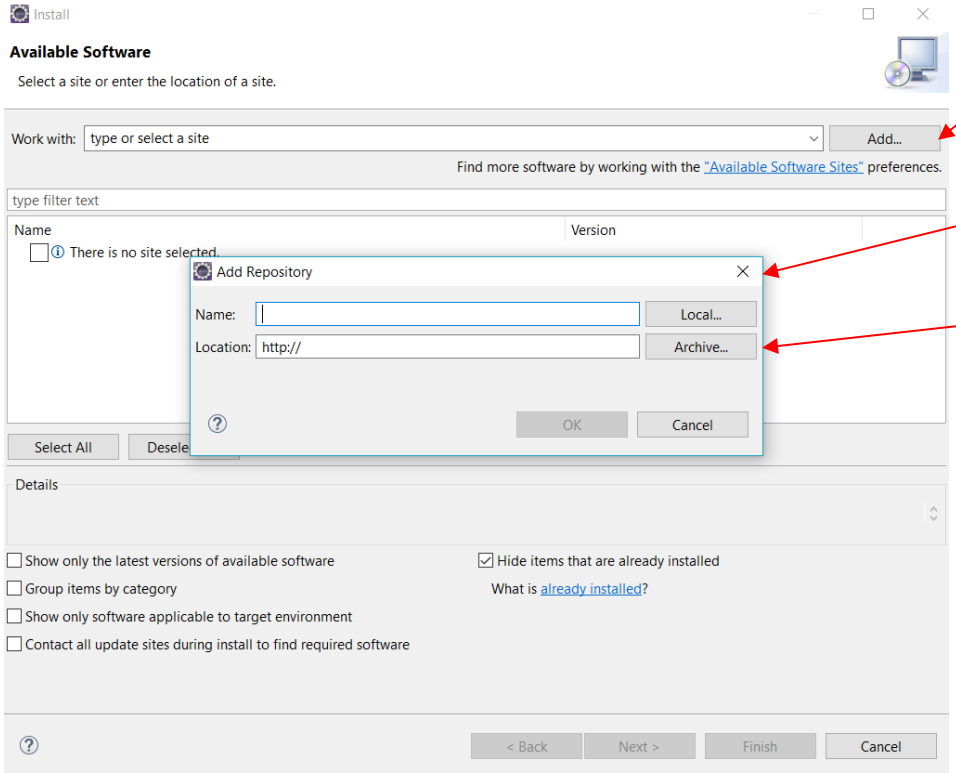


Fig 80 – Paquete Instalación

Fig 79 – Update Site Project II

2. Intalación

Para instalar un plugin lo más fácil es hacerlo a partir del instalador automático de Eclipse que se encuentra en la ruta Help>Install New Software. Nos aparecerá la siguiente imagen:



Pulsamos “Add” y nos aparecerá el siguiente cuadro de diálogo.

Pulsamos “Archive” y buscamos el zip creado en los pasos anteriores.

Tras pulsar “Next” el paquete será instalado automáticamente y solo quedaría reiniciar Eclipse.

Fig 81 – Asistente de instalación