Grado en Ingeniería Telemática

Escuela Politécnica Superior de Linares

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

APLICACIÓN PARA EL PROCESADO Y SUBIDA AUTOMÁTICOS DE TEST DE EVALUACIÓN A ILIAS. TESTUJA.

Alumno: Antonio Osuna Melgarejo

Tutor: Prof. D. Juan Carlos Cuevas Martínez
Depto.: Ingeniería de Telecomunicación

Febrero, 2020

Resumen

En el presente Trabajo Fin de Grado se muestra una revisión de las capacidades de interacción con ILIAS (*Integrated Learning, Information and Cooperation System*) así como una aplicación, denominada TestUJA, para la creación de bancos de preguntas y las preguntas en sí en ILIAS. El motivo de emplear ILIAS se debe a que es el gestor de aprendizaje online empleado por la Universidad de Jaén. TestUJA es una aplicación de escritorio desarrollada en Java que, a través del empleo de los servicios web de ILIAS y de comunicaciones HTTP, permite, tanto la creación de bancos de preguntas, como la creación de preguntas tipo test en un banco de preguntas en dicha plataforma. Todo esto lo realiza a través de la interpretación de ficheros de texto con un formato sencillo, especialmente diseñado en el presente Trabajo fin de Grado, que facilita la edición y subida de cuestiones a los bancos de preguntas. Esta aplicación evita el uso de la interfaz propia de ILIAS, la cual implica la selección de elementos en varias páginas, así como completar una gran diversidad de campos en formularios, los cuales, en ciertos casos, no están pensados para su revisión.

Abstract

This Final Degree Project presents a revision of the interaction skills with ILIAS (Integrated Learning, Information and Cooperation System) as well as an application, TestUJA, that allows the creation of question banks and test questions in ILIAS. The reason for using ILIAS is because the University of Jaén employs ILIAS as online learning manager. TestUJA is a desktop application developed in Java that, through the use of ILIAS web services and HTTP communications, allows both the creation of question banks and the creation of test questions in a question bank in that platform. TestUJA employs the interpretation of text documents with a simple format, specially designed in this Final Degree Project, which facilitates the editing and uploading of test questions to question banks. This application avoids that users have to use the ILIAS interface, which implies the selection of elements on several pages, as well as completing a lot of fields in a form.

INDICE

Resumen 2
Abstract 3
1 Memoria 8
1.1 Introducción 8
1.1.1 Objetivos 8
1.1.2 Antecedentes 9
1.1.3 Estado del arte 9
1.2 Descripción del proyecto 11
1.3 Estudios y Desarrollo 12
1.3.1 Estudios 12
1.3.2 Desarrollos necesarios. 30
1.4 Resultado 45
1.4.1 Proceso correcto. 45
1.4.2 Proceso con errores. 45
1.5 Formato de las preguntas tipo test. TESTUJAFORMAT. 47
1.5.1 Pregunta Respuesta Única 48
1.5.2 Pregunta Respuesta Múltiple 50
1.5.3 Pregunta Seleccionar Error (es) 52
1.5.4 Pregunta Rellenar hueco(s) 53
1.5.5 Pregunta Respuesta Corta 54
1.5.6 Ejemplo de fichero TXT enviado por los alumnos. 55
1.6 Conclusiones y líneas de futuro. 56
1.6.1 Conclusiones 56
1.6.2 Líneas de futuro. 59
1.7 Bibliografía 59
2 Anexos 61
Manual de usuario 61
Introducción 61
Inicio 61
Login 62
Menú principal 63
Buscar bancos de preguntas 63
Crear bancos de preguntas 68
Validación de documentos de texto 72
Subir preguntas 73

Ayuda HTML 75
Manual de mantenimiento 77
3 Planos 78
3.1 Plano inicio de sesión 79
3.2 Plano menú principal. 80
3.3 Plano buscar banco de preguntas. 81
3.4 Plano buscar banco de preguntas según Ref_ID 82
3.5 Plano crear banco de preguntas. 83
3.6 Plano crear banco de preguntas según Ref_ID. 84
3.7 Plano comprobar documento. 85
3.8 Plano subir preguntas. 86
4 Pliego de condiciones 87
4.1 Elementos software y licencias. 87
4.2 Requisitos del usuario 87
5 Mediciones 88
5.1 Estudios preliminares de tecnologías y especificación de requisitos. 88
5.2 Diseño preliminar 88
5.3 Implementación del servicio 88
5.4 Diseño definitivo 88
5.5 Pruebas y corrección de errores 88
5.6 Elaboración de la documentación 88
6 Presupuesto 89

Índice de imágenes.

Ilustración 1. Ejemplo interfaz Respondus. Pregunta tipo Respuesta Única. 10
Ilustración 2. Esquema de funcionamiento de TestUJA. 12
Ilustración 3. Ejemplo de pregunta tipo test formada con QTI. 14
Ilustración 4. Resultado de la pregunta tipo test creada en un LMS. 15
Ilustración 5. Estructura WSDL según versiones. 16
Ilustración 6. Información mostrada en el WSDL sobre el servicio "login". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php. 18
Ilustración 7. Información mostrada en el WSDL sobre el servicio "getUserIdBySid". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php. 20
Ilustración 8. Información mostrada en el WSDL sobre el servicio "getObjectsByTitle". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php. 21
Ilustración 9. Información mostrada en el WSDL sobre el servicio "addObject". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php. 23
Ilustración 10. Símbolo banco de preguntas 24
Ilustración 11. Botón Crear preguntas. 27

Ilustración 12. Crear esqueleto según WSDL. Paso 1. 31
Ilustración 13. Crear esqueleto según WSDL. Paso 2. 32
Ilustración 14. Crear esqueleto según WSDL. Paso 3. 32
Ilustración 15. Menu inicial de la aplicación TestUJA.....37
Ilustración 16. Inicio de sesión de la aplicación TestUJA.....38
Ilustración 17. Menú principal de la aplicación TestUJA.....39
Ilustración 18. Búsqueda del banco de preguntas de la aplicación TestUJA.....40
Ilustración 19. Búsqueda del banco de preguntas según REF_ID de la aplicación TestUJA. 41
Ilustración 20. Creación de un banco de preguntas de la aplicación TestUJA. 42
Ilustración 21. Creación de un banco de preguntas según REF_ID de la aplicación TestUJA.....43
Ilustración 22. Comprobación de documentos de la aplicación TestUJA.....44
Ilustración 23. Subir preguntas de la aplicación TestUJA.....45
Ilustración 24. Diferencia entre Title y Question. 50
Ilustración 25. Formulario pregunta de opción múltiple en ILIAS.....58
Ilustración 26. Menú inicial 61
Ilustración 27. Login 62
Ilustración 28. Información. Login correcto.....62
Ilustración 29. Error. Login incorrecto 63
Ilustración 30. Menú principal.....63
Ilustración 31. Búsqueda del banco de preguntas.....64
Ilustración 32. Error. Banco de preguntas no encontrado.....64
Ilustración 33. Información de un banco de preguntas 65
Ilustración 34. Información de varios bancos de preguntas. 65
Ilustración 35. Búsqueda banco de preguntas según ref_id.....66
Ilustración 36. Información botón ref_ids buscar banco de preguntas 66
Ilustración 37. Información búsqueda banco de preguntas según ref_ids.....67
Ilustración 38. Información banco de preguntas encontrado 67
Ilustración 39. Creación de banco de preguntas 68
Ilustración 40. Error. Banco de preguntas y carpeta sin nombre 68
Ilustración 41. Error. Carpeta no encontrada. 69
Ilustración 42. Búsqueda encontrada de una carpeta.....69
Ilustración 43. Búsqueda encontrada de varias carpetas 69
Ilustración 44. Creación del banco de preguntas según ref_ids 70
Ilustración 45. Botón "Elegir Carpeta" crear banco de preguntas 70
Ilustración 46. Búsqueda correcta carpeta.....71
Ilustración 47. Información banco de preguntas creado.....71
Ilustración 48. Validación de documentos de texto. 72
Ilustración 49. Información Validación de documentos 72
Ilustración 50. Subir preguntas 73
Ilustración 51. Ejemplo subir preguntas 74
Ilustración 52. Información preguntas tipo test subidas 74
Ilustración 53. Acceder a ayuda HTML. 75
Ilustración 54. Ejemplo ayuda HTML. 76

Índice de tablas.

Tabla 1. Servicios web ofrecidos por ILIAS. 17
Tabla 2. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web "login".....19

Tabla 3. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web “getUserIdBySid”. 20
Tabla 4. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web “getObjectByTitle”. 22
Tabla 5. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web” addObject”...24
Tabla 6. Nombres de las clases que muestran las interfaces Gráficas. 36
Tabla 7. Estudios preliminares de tecnologías y especificación de requisitos. 88
Tabla 8. Diseño preliminar 88
Tabla 9. Implementación del servicio. 88
Tabla 10. Diseño definitivo. 88
Tabla 11. Pruebas y corrección de errores. 88
Tabla 12. Elaboración de la documentación. 88
Tabla 13. Resumen presupuesto. 89

1 Memoria

1.1 Introducción

En este proyecto se describe la implementación y el diseño de una aplicación de escritorio Java, denominada TestUJA, que permite, tanto la creación de bancos de preguntas, como la subida de diferentes tipos de preguntas test (respuesta simple, respuesta múltiple, rellenar huecos, detectar errores y respuesta corta) a un banco de preguntas en ILIAS (1).

Este proyecto surge debido a la necesidad de facilitar la tediosa y reiterativa tarea que deben realizar los usuarios de ILIAS a la hora de crear preguntas en un banco de preguntas en ILIAS. Para ello, se ha diseñado la aplicación de escritorio TestUJA, que permite subir preguntas tipo test de manera automatizada y rápida a un banco de preguntas en ILIAS a partir de simples documentos de texto. Estos documentos de texto estarán redactados con unas marcas sencillas que identificarán los diferentes tipos de preguntas, así como los enunciados y puntuaciones de cada pregunta. Por otro lado, como es necesario tener un banco de preguntas creado en la plataforma para poder subir preguntas tipo test, TestUJA también facilita la creación de bancos de preguntas.

Para poder utilizar TestUJA es requisito principal estar registrado como usuario de la Universidad de Jaén. En este proyecto no ha sido necesario la creación de una base de datos de usuarios, pues para registrarse en TestUJA se utilizará el servicio de autenticación que ofrece ILIAS que permite verificar las credenciales de usuario de la Universidad de Jaén.

1.1.1 Objetivos

La herramienta a desarrollar deberá facilitar la tediosa tarea de crear bancos de preguntas en ILIAS, para que, a través de documentos de texto donde se puedan redactar cómodamente los enunciados, y a través de sencillas marcas, poder poner el tipo y puntuación de cada pregunta.

Además, esta herramienta deberá cumplir con los siguientes objetivos:

1. Analizar y detallar el formato empleado para la definición de bancos de preguntas para ILIAS.
    a. Análisis del código fuente de ILIAS.
    b. Análisis de las comunicaciones.
    c. Análisis de los mecanismos de seguridad empleados.
    d. Extraer el formato completo empleado para la definición de bancos de preguntas en ILIAS.
2. Preparar una herramienta que permita leer documentos de texto o texto plano con formato sencillo (que definan preguntas o bancos de preguntas) para realizar la subida automática de test de ILIAS.
    a. La herramienta deberá tener el interfaz más sencillo posible y automatizar el proceso al máximo.
    b. La herramienta podrá estar basada en tecnologías web o de escritorio (o ambas, opcionalmente).
    c. Deberá ser totalmente portable y funcionar sobre sistemas Windows, Mac OS o Linux.
3. Preparar un manual de usuario que detalle perfectamente el uso de la misma, tanto en formato PDF, como HTML.
4. [Opcional] Confeccionar un manual de administración de la herramienta, si fuera necesario.

Confeccionar un manual de mantenimiento detallado con todo el diseño, implementación y pruebas de la herramienta desarrollada.

1.1.2 Antecedentes

Este proyecto se basa en el Proyecto Fin de Carrera denominado “Aplicación para el control de acceso a dependencias y edificios basado en tecnología NFC para terminales Android” y que fue realizado por José Manuel Jiménez Bravo como parte de la titulación Ingeniería de Telecomunicación cursada en la Universidad de Jaén, el cual ha supuesto un principio para el desarrollo de este, ya que nos ha permitido comprender el funcionamiento de los servicios web de ILIAS y se ha podido reutilizar parte de la funcionalidad implementada para manejar los servicios web.

En ese Proyecto Fin de Carrera se desarrolló una aplicación capaz de controlar el acceso de los alumnos a sus asignaturas utilizando tecnología NFC (*Near Field Communication*) y servicios web.

En este proyecto, se utilizará principalmente servicios web, los cuales permitirán realizar el inicio de sesión, búsqueda de los bancos de preguntas y crear bancos de preguntas, y el protocolo HTTP (2), que permitirá crear las diferentes preguntas tipo test en los bancos de preguntas elegidos.

1.1.3 Estado del arte

La aplicación desarrollada en el presente trabajo fin de grado, principalmente, permite la creación de preguntas tipo test en un banco de preguntas de ILIAS.

Una aplicación que existe actualmente en el mercado y que realiza funcionalidades parecidas a las que realiza el programa definido en la presente memoria es:

1.1.3.1 RESPONDUS

Respondus (3) es un potente programa que permite crear y administrar exámenes. Estos exámenes pueden imprimirse en papel o publicarse directamente en Canvas, Moodle, IMS QTI y otros sistemas de aprendizaje. Para crear estos exámenes es necesario crear preguntas tipo test antes. Este programa proporciona dos posibilidades para la creación de preguntas:

- Crear las preguntas utilizando la interfaz de usuario de Respondus.
- Importar las preguntas a través de un documento siguiendo el formato especificado por Respondus.

Se pueden crear cualquier tipo de pregunta test, desde preguntas de tipo respuesta única a preguntas de tipo rellenar huecos.

A continuación, se muestra un ejemplo creando preguntas tipo test utilizando la interfaz de Respondus y otro ejemplo importando las preguntas desde un documento de texto.

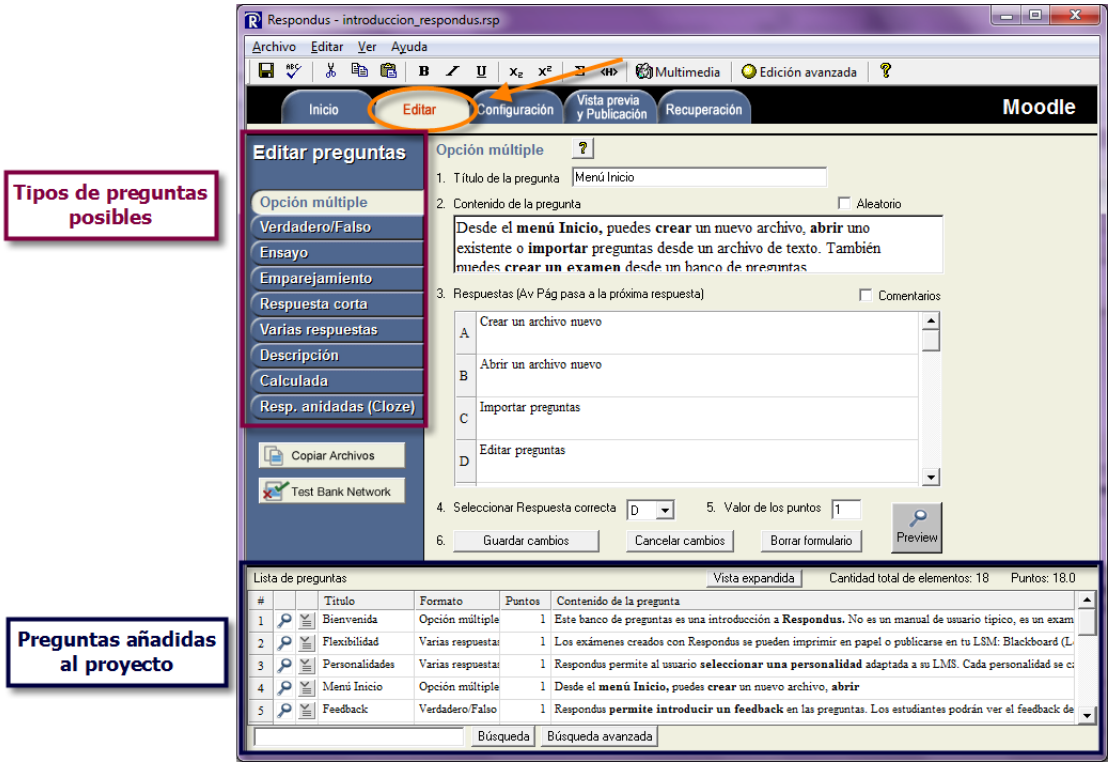


Ilustración 1. Ejemplo interfaz Respondus. Pregunta tipo Respuesta Única.

Por otro lado, para importar preguntas con Respondus debes tenerlas en un documento de word **(.doc o .docx),** un documento de texto enriquecido **(.rtf)** o un bloc de notas **(.txt)** siguiendo el formato especificado por Respondus. Por ejemplo, para poder crear una pregunta de tipo Respuesta Única, el documento de texto debe estar formado siguiendo las siguientes indicaciones:

- El documento debe estar escrito en **texto plano**.
- **Título de la pregunta** escribe la palabra **"Title:"** seguido del título.
- Indica el **número de la pregunta** seguido de paréntesis **")"** o punto **"."**
- Indica las **respuestas** con letras minúsculas seguidas de un punto **"."**
- **La respuesta correcta** se indicará con un asterisco "*" delante de la letra de la respuesta.
- **Puntuación.** Si quieres incluir un valor a las preguntas escribe **"Points:"** seguido del valor que corresponda. Todas las preguntas se puntuarán con la cantidad escrita hasta que escribas de nuevo "Points:" y un valor diferente.

Ejemplo:

```
Title: Características de Respondus

Points: 2.5

1. Indica la afirmación incorrecta

a. Con Respondus puedes publicar exámenes en diferentes plataformas.

b. Puedes imprimir tus exámenes de Respondus en papel.
```

```
*c. Respondus no puede convertir exámenes de una plataforma a
otra.
```

Una vez se hayan creado todas las preguntas utilizando la interfaz de Respondus o mediante documentos de texto, será posible crear exámenes o publicar documentos de estas preguntas o de los exámenes en formato Moodle, IMS QTI (4).

La Universidad de Jaén utiliza ILIAS como sistema de gestión de aprendizaje y una posibilidad que ofrece es crear preguntas tipo test mediante la importación de documentos en formato IMS QTI. Con Respondus es posible obtener estos documentos en formato IMS QTI para, posteriormente, importarlos a ILIAS y crear preguntas tipo test de manera rápida y eficiente.

1.2 Descripción del proyecto

Este proyecto presenta un sistema que permite la creación de preguntas tipo test en un banco de preguntas creado con anterioridad en docencia virtual. También permite la creación de un banco de preguntas para posteriormente subir preguntas tipo test a dicho banco. Para poder crear preguntas tipo test en un banco de preguntas, los alumnos deberán crear documentos de texto donde redactarán las diferentes preguntas siguiendo un formato específico.

Este formato contará con unas marcas sencillas que identificarán los enunciados, la puntuación, el tipo de preguntas qué es, etc.

Este programa deberá ser ejecutado en un entorno de desarrollo de Java (IDE). La comunicación entre el programa y la plataforma de docencia se establecerá mediante la tecnología SOAP (con los servicios web que implementa ILIAS) y el Protocolo de Transferencia de Hipertexto o, mayormente conocido como HTTP.

Para poder utilizar el programa, será necesario iniciar sesión con la cuenta de la Universidad de Jaén. En este proyecto no ha sido necesario la creación de una base de datos con los usuarios, debido a que se utilizará la base de datos de usuarios de ILIAS. Una vez que el usuario se ha registrado correctamente, deberemos buscar o crear el banco de preguntas donde se subirán las preguntas tipo test encontradas en el documento de texto.

Tanto para el inicio de sesión, como para la búsqueda y la creación del banco de preguntas se utilizará la tecnología SOAP (5), empleando los servicios web definidos en el *Web Services Description Language* (WSDL) de ILIAS. Mientras que para la subida de las preguntas tipo test a los bancos de preguntas se utilizará el protocolo HTTP. Este proyecto estaba previamente diseñado para crear preguntas utilizando solo servicios web, sin embargo, debido a algunas inconsistencias detectadas utilizando dichos servicios se tuvo que optar por utilizar otra tecnología.

Para la búsqueda de un banco de preguntas, el programa buscará en ILIAS los bancos de preguntas a los que un determinado usuario tiene acceso con privilegios de administración. Para la creación de un banco de preguntas, el programa buscará en ILIAS las carpetas a los que un determinado usuario tiene acceso con privilegios de administración. Una vez se seleccione la carpeta (si el usuario tiene acceso con privilegios) se podrá crear el banco de preguntas en dicha carpeta.

Para subir preguntas tipo test a un banco de preguntas se utilizará, principalmente, el método POST del protocolo HTTP.

A continuación, se muestra un esquema del sistema de funcionamiento.

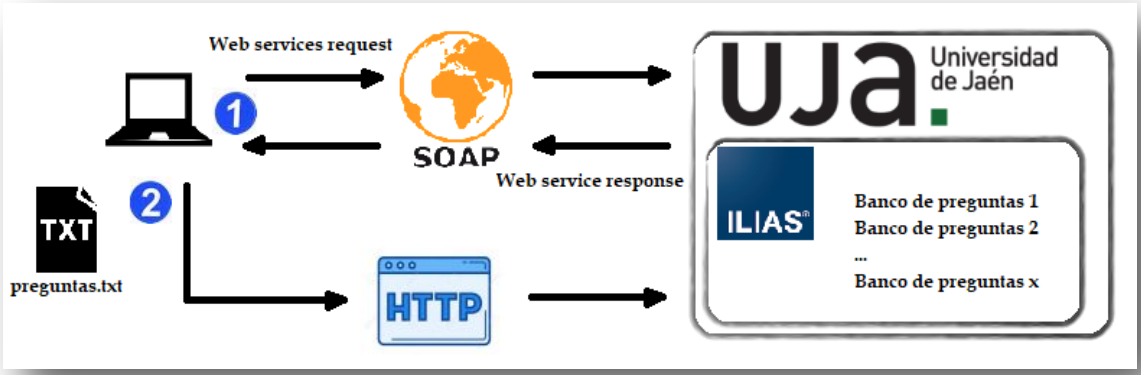


Ilustración 2. Esquema de funcionamiento de TestUJA.

1.3 Estudios y Desarrollo

El proyecto se ha desarrollado, principalmente, en dos fases:

- Estudios necesarios para el desarrollo del proyecto.
- Desarrollo del proyecto.

1.3.1 Estudios

1.3.1.1 Estudio previo de los elementos necesarios para la realización del proyecto.

Esta es la fase principal y fundamental del proyecto, en la cual se necesita información sobre los siguientes elementos:

- **¿Qué es ILIAS?** Se ha realizado un breve estudio sobre qué es ILIAS.
- **¿Qué es el estándar QTI? ¿Para qué sirve? ¿Por qué es necesario en ILIAS?** Se ha realizado un estudio sobre el estándar QTI y todos sus componentes para así poder comprender cómo se lleva a cabo el intercambio de preguntas y exámenes test entre sistemas de gestión de aprendizaje (*Learning Managements Systems* o LMS).
- **¿Qué es un servicio web? ¿Para qué se utiliza?** Se ha realizado un estudio sobre qué son los servicios web y cómo utilizarse. También se ha realizado un estudio de todos los servicios web que ofrece la plataforma ILIAS, así como, de la forma de usarlos.
- **¿Qué es un banco de preguntas? ¿Para qué se utiliza?** Se ha realizado un estudio para ver qué son los bancos de preguntas, para qué se utilizan y cómo es el formato empleado para definir un banco de preguntas en la plataforma.
- **Programación en entorno Java.** Se ha realizado un estudio sobre qué lenguaje de programación es el más adecuado (lenguaje de programación simple, robusto, seguro y con alto rendimiento) y el más estandarizado posible, para que así sea compatible con diferentes sistemas operativos como Windows, Linux y Mac OS.
- **Protocolo HTTP y sus métodos.** Por último, y debido a incompatibilidades en la forma de crear preguntas en un banco de preguntas con servicios web, se ha tenido que realizar un estudio de cómo funciona la creación de preguntas tipo test según el protocolo HTTP y sus métodos.

1.3.1.2 Estudio de ILIAS.

ILIAS es un sistema de gestión para la enseñanza (LMS) que fue desarrollado por la Universidad de Colonia en 1998. Debido al interés de otras universidades, la Universidad de Colonia decidió publicar ILIAS como software libre de código abierto bajo la licencia GPL (GNU General Public License) para que pudiera ser utilizado sin ninguna restricción. ILIAS emplea varias tecnologías, entre las que destacan:

- Servicios Web junto al protocolo SOAP (*Simple Object Access Protocol*).
- Estándar QTI (*Question & Test Interoperability*).
- Lenguaje de marcado XML (*Extensible Markup Language*).
- Protocolo HTTP.
- Lenguaje de programación JavaScript.

Debido a que ILIAS se puede adaptar fácilmente a los requerimientos específicos de cada organización, es muy utilizada por diferentes universidades y organizaciones (entre ellas destacan la Universidad de Jaén, la Universidad de Colonia o la Universidad de Stuttgart) como sistema de gestión para la enseñanza.

Como se ha comentado, la Universidad de Jaén utiliza ILIAS como sistema de gestión para la enseñanza. ILIAS facilita que los profesores puedan distribuir a través de la plataforma el material de sus asignaturas, evaluaciones de alumnos, etc. Por otro lado, a los alumnos se les facilita la posibilidad de acceder al material de las asignaturas, participar en exámenes, participar en foros…

Junto a todas estas facilidades, ILIAS también ofrece un conjunto de servicios web para que sus usuarios puedan realizar investigaciones o aplicaciones relacionadas con ILIAS.

1.3.1.3 Estudio del estándar QTI.

Las especificaciones de interoperabilidad de preguntas y test (QTI) describen la estructura básica para la representación de preguntas o test en una plataforma *e-learning*. Esta especificación permite el intercambio de preguntas y test entre sistemas de gestión para la enseñanza. La especificación QTI está definida en lenguaje XML (6) para que pueda ser adoptada lo más ampliamente posible.

ILIAS, al ser un sistema de gestión de aprendizaje, utiliza el estándar QTI para la definición de preguntas tipo test y exámenes tipo test.

Algunas de las tecnologías que se utilizan para el desarrollo de contenidos QTI, garantizando su interoperabilidad son:

- XML.
- *Document Type Definition* (DTD) y *Schemas*, que contienen una descripción de la estructura y sintaxis de un documento XML.
- Servicios Web.

¿Qué tipos de preguntas pueden hacerse con la especificación QTI?

Se pueden hacer preguntas de todo tipo (respuesta simple, múltiple, rellenar huecos, ordenación…). A continuación, se muestra un ejemplo de cómo debe estar formado un documento XML, utilizando la especificación QTI, para crear una pregunta tipo respuesta simple en la plataforma de la Universidad de Jaén:

```
<questestinterop>
  <qticomment>
    This is a simple True/False multiple choice example using V1.2.
    The rendering is a standard radio button style.
    Response processing is incorporated.
  </qticomment>
  <item ident="IMS_V01_I_BasicExample001">
    <presentation label="BasicExample001">
      <flow>
        <material>
        <mattext>Paris is the Capital of France</mattext>
        </material>
        <response_lid ident="TF01" rcardinality="Single" rtiming="No">
        <render_choice>
          <response_label ident="T">
              <flow_mat>
                <material><mattext>Agree</mattext></material>
              </flow_mat>
          </response_label>
          <response_label ident="F">
              <flow_mat>
                <material><mattext>Disagree</mattext></material>
              </flow_mat>
          </response_label>
          </render_choice>
        </response_lid>
      </flow>
  </presentation>
    <resprocessing>
      <outcomes>
        <decvar/>
      </outcomes>
      <respcondition title="Correct">
        <conditionvar>
          <varequal respident="TF01">T</varequal>
        </conditionvar>
        <setvar action="Set">1</setvar>
        <displayfeedback feedbacktype="Response" linkrefid="Correct"/>
      </respcondition>
    </resprocessing>
    <itemfeedback ident="Correct" view="Candidate">
      <flow_mat>
        <material><mattext>Yes, you are right.</mattext></material>
      </flow_mat>
    </itemfeedback>
  </item>
</questestinterop>
```

Ilustración 3. Ejemplo de pregunta tipo test formada con QTI.

Daría como resultado la siguiente pregunta tipo test:

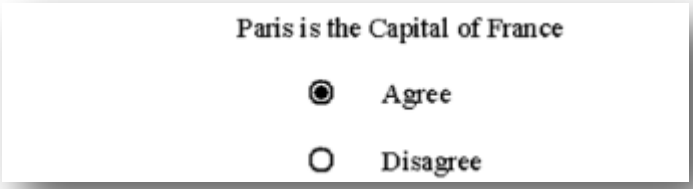


Ilustración 4. Resultado de la pregunta tipo test creada en un LMS.

1.3.1.4 Estudio de los servicios web de ILIAS.

La Universidad de Jaén ofrece sus servicios web mediante el Lenguaje de Descripción de Servicios Web (WSDL) y utiliza el protocolo SOAP que permite llamar, instanciar y ejecutar los servicios Web.

El WSDL (7) describe los servicios web que hay disponibles utilizando varios elementos (identificados con etiquetas XML). Dichos elementos se pueden clasificar como abstractos o concretos. La parte WSDL abstracta describe las operaciones y mensajes con detalle o, en otras palabras, especifica qué hace el servicio:

- Qué operaciones están disponibles.
- Qué entradas, salidas y mensajes de error tienen las operaciones.
- Cuáles son las definiciones de los tipos para los mensajes de entrada, salida y error.

La parte WSDL concreta describe el cómo y dónde del servicio:

- Cómo tiene que llamar un cliente al servicio.
- Qué protocolo debería usar.
- Dónde está disponible el servicio.

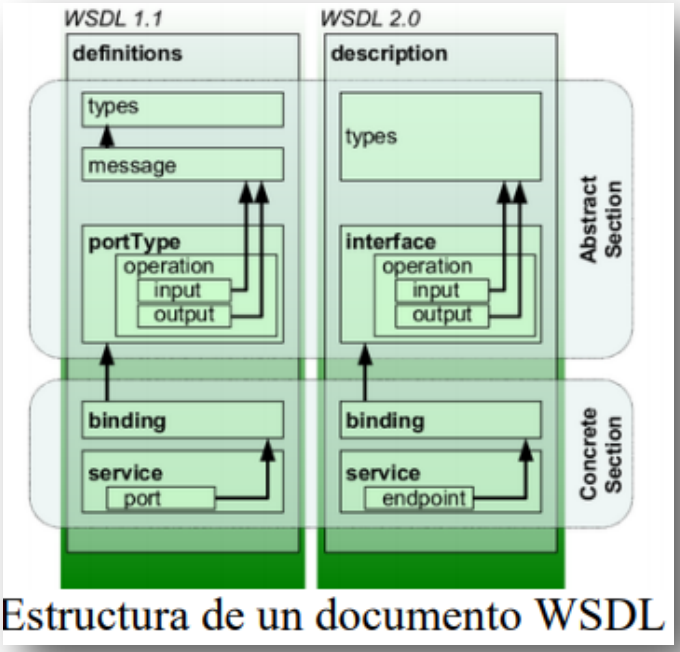


Ilustración 5. Estructura WSDL según versiones.

Algunos de los servicios web más destacados que ofrece ILIAS a sus usuarios se muestran en la siguiente tabla:

Servicio web	Descripción
login	Servicio que realiza el login de un usuario en ILIAS.
logout	Servicio que realiza el logout o desconexión de un usuario en ILIAS.
lookupUser	Servicio que permite comprobar si un usuario existe.
getUser	Servicio obsoleto que permite obtener los datos de un usuario en ILIAS.
deleteUser	Servicio obsoleto que permite borrar todos los datos relacionados con el usuario de ILIAS.
addCourse	Servicio que permite añadir un curso.
deleteCourse	Servicio que permite eliminar un curso.
assignCourseMember	Servicio que permite asignar un usuario a un curso.
excludeCourseMember	Servicio que permite excluir un usuario de un curso.
isAssignedToCourse	Servicio que permite comprobar si un usuario está asignado a un curso.
getCourseXML	Servicio que obtiene la información de un curso en formato XML.
updateCourse	Servicio que permite actualizar los datos de un curso.
getObjIdByImportId	Servicio que obtiene el identificador de un objeto mediante el identificador de importación.
getRefIdsByObjId	Servicio que obtiene los identificadores de referencia mediante el identificador del objeto.
getRefIdsByImportId	Servicio que obtiene los identificadores de referencia mediante el identificador de importación.

getObjectByReference	Servicio que obtiene la descripción XML de un objeto de ILIAS mediante el identificador de referencia.
getObjectsByTitle	Servicio que obtiene la descripción XML de los objetos de ILIAS mediante el título del objeto.
searchObjects	Servicio que permite buscar objetos.
updateObjects	Servicio que permite actualizar objetos de ILIAS.
addReference	Servicio que permite crear un nuevo enlace entre un objeto seleccionado y uno nuevo.
deleteObject	Servicio que permite borrar un objeto existente.
revokePermissions	Servicio que permite eliminar todos los permisos de un objeto o rol específico.
grantPermissions	Servicio que permite conceder permisos para un rol o un objeto específico.
getUserRoles	Servicio que permite obtener los roles asignados a un usuario.
addRole	Servicio que permite crear un rol.
deleteRole	Servicio que permite eliminar un rol.
addGroup	Servicio que permite crear un grupo.
groupExists	Servicio que comprueba si existe un grupo.
getGroup	Servicio que permite obtener un grupo.
saveQuestionResult	Servicio que permite guardar los resultados de un cuestionario.
getTestUserData	Servicio que permite obtener los datos de un test de un usuario.
getQuestionSolution	Servicio que permite obtener la solución de una pregunta.
getStructureObjects	Servicio que permite obtener la estructura de objetos.
importUsers	Servicio que permite importar usuarios.
searchUsers	Servicio que permite buscar usuarios.
addExercise	Servicio que permite crear ejercicios en la plataforma.
getExerciseXML	Servicio que permite obtener el XML de un ejercicio.
updateExercise	Servicio que permite actualizar un ejercicio.
getFileXML	Servicio que permite obtener el XML de un archivo.
addFile	Servicio que permite crear un archivo.
updateFile	Servicio que permite actualizar un archivo.
getUserIdBySid	Servicio que permite obtener el identificador de usuario mediante el identificador de sesión.
getClientInfoXML	Servicio que permite obtener la información en formato XML del cliente.
getPathForRefId	Servicio que permite obtener la ruta para un determinado identificador de referencia.
getGroupsForUser	Servicio que permite obtener todos los grupos en los que participa un usuario.
getCoursesForUser	Servicio que permite obtener todos los cursos en los que participa un usuario.
getTestResults	Servicio que permite obtener los resultados de un test.
removeTestResults	Servicio que permite eliminar los resultados de un test.
getProgressInfo	Servicio que permite obtener la información de progreso de un usuario.
deleteProgress	Servicio que permite eliminar la información de progreso de un usuario.

Tabla 1. Servicios web ofrecidos por ILIAS.

De todos estos servicios web que ofrece ILIAS a sus usuarios, se detallan aquellos que se utilizan para realizar el proyecto:

- **Login:** Servicio que permite realizar un proceso de identificación. Comprueba si el usuario está registrado en la plataforma. Para poder utilizar este servicio web es necesario introducir los siguientes parámetros:
    - **Client:** Especifica el nombre del cliente. Para la plataforma de ILIAS de la Universidad de Jaén el nombre del cliente es "docencia".
    - **Username:** Es el nombre del usuario en la plataforma de ILIAS de la Universidad de Jaén. **Un ejemplo de un usuario de ILIAS es "aom00001" (cualquier usuario de ILIAS, tanto profesor como alumno, está definido normalmente mediante sus iniciales y cinco números).**
    - **Password:** Es la contraseña asignada al usuario dentro de la plataforma de ILIAS de la Universidad de Jaén.

  Como salida, este servicio web devolverá el **identificador de sesión** del usuario registrado. Este identificador de sesión será diferente en cada acceso. Un ejemplo de salida de este servicio web es "b8d888b82447308fa04ab99be2dc40::docencia".

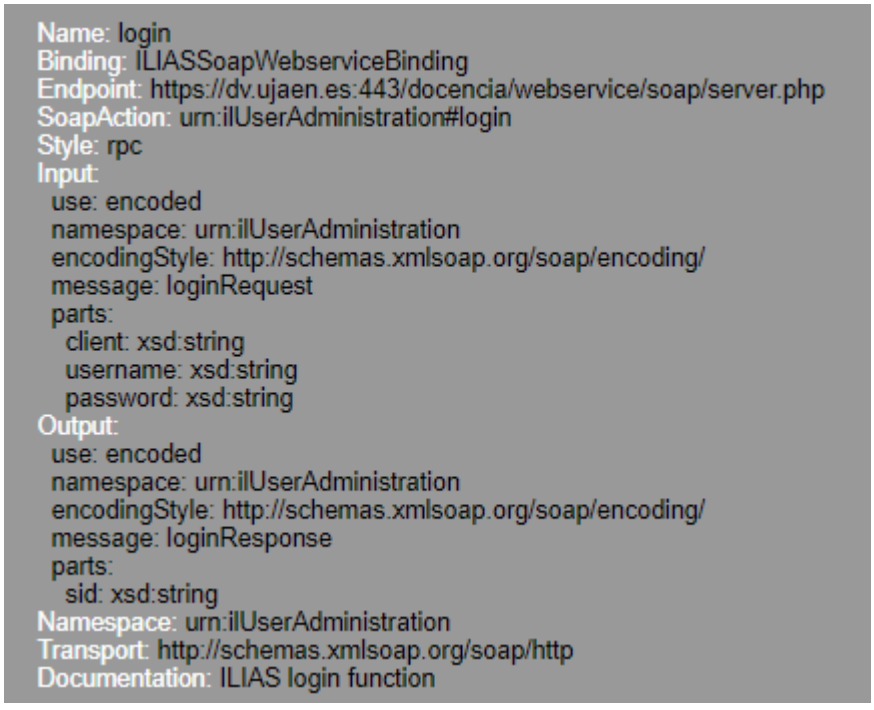


Ilustración 6. Información mostrada en el WSDL sobre el servicio "login". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php.

A continuación, se muestra un ejemplo de los mensajes intercambiados (protocolo SOAP) entre el cliente y el servidor para ejecutar el servicio web "login".

Petición del cliente

`<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:ilUserAdministration">`
`<soapenv:Header/>`
`<soapenv:Body>`

```
      <urn:login
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <client xsi:type="xsd:string">docencia</client>
         <username xsi:type="xsd:string">aom00001</username>
         <password xsi:type="xsd:string">01contrasena</password>
      </urn:login>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta del servidor

```
<SOAP-ENV:Envelope                                                SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"    xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:ilUserAdministration"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"            xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:loginResponse>
         <sid
xsi:type="xsd:string">795de233f521455087aa1343037b70c9::docencia</sid>
      </ns1:loginResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabla 2. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web "login".

- **GetUserIdBySid:** Servicio que permite obtener el identificador de usuario a través del identificador de sesión obtenido del servicio web "login". Este servicio web tiene como parámetro de entrada:
    - **Sid:** Este parámetro es el identificador de sesión que se obtiene cuando un usuario inicia sesión de manera satisfactoria en la plataforma mediante el servicio web "login".

  Como salida, este servicio web devolverá el identificador del usuario. Este identificador es único para cada usuario. Un ejemplo de salida de este servicio web es "822479".

```
Name: getUserIdBySid
Binding: ILIASSoapWebserviceBinding
Endpoint: https://dv.ujaen.es:443/docencia/webservice/soap/server.php
SoapAction: urn:ilUserAdministration#getUserIdBySid
Style: rpc
Input:
  use: encoded
  namespace: urn:ilUserAdministration
  encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
  message: getUserIdBySidRequest
  parts:
    sid: xsd:string
Output:
  use: encoded
  namespace: urn:ilUserAdministration
  encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
  message: getUserIdBySidResponse
  parts:
    usr_id: xsd:int
Namespace: urn:ilUserAdministration
Transport: http://schemas.xmlsoap.org/soap/http
Documentation: ILIAS getUserIdBySid(): returns an ILIAS usr_id for the given sid
```

Ilustración 7. Información mostrada en el WSDL sobre el servicio "getUserIdBySid". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php.

A continuación, se muestra un ejemplo de los mensajes intercambiados entre el cliente y el servidor para ejecutar el servicio web "getUserIdBySid".

Petición del cliente

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ilUserAdministration">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:getUserIdBySid
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <sid
xsi:type="xsd:string">795de233f521455087aa1343037b70c9::docencia</sid>
      </urn:getUserIdBySid>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta del cliente

```
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:ilUserAdministration"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:getUserIdBySidResponse>
         <usr_id xsi:type="xsd:int">822479</usr_id>
      </ns1:getUserIdBySidResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabla 3. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web "getUserIdBySid".

- **GetObjectsByTitle:** Servicio que permite obtener la información en formato XML de objetos (banco de preguntas, ejercicios…) a través de su nombre. Este servicio web tiene como parámetros de entrada:
    - **Sid:** Este parámetro es el identificador de sesión que se obtiene cuando un usuario inicia sesión de manera satisfactoria en la plataforma mediante el servicio web "login".
    - **Title:** Este parámetro es el nombre del objeto buscado. Este objeto puede ser una carpeta, un banco de preguntas, una asignatura…
    - **User_id:** Este parámetro es el identificador del usuario que se obtiene del servicio web "getUserIdBySid".

  Como salida, este servicio web devolverá la información en formato XML de los objetos encontrados con el nombre similar al introducido en el parámetro de entrada "title" y en los que el usuario tenga acceso con privilegios de administración.

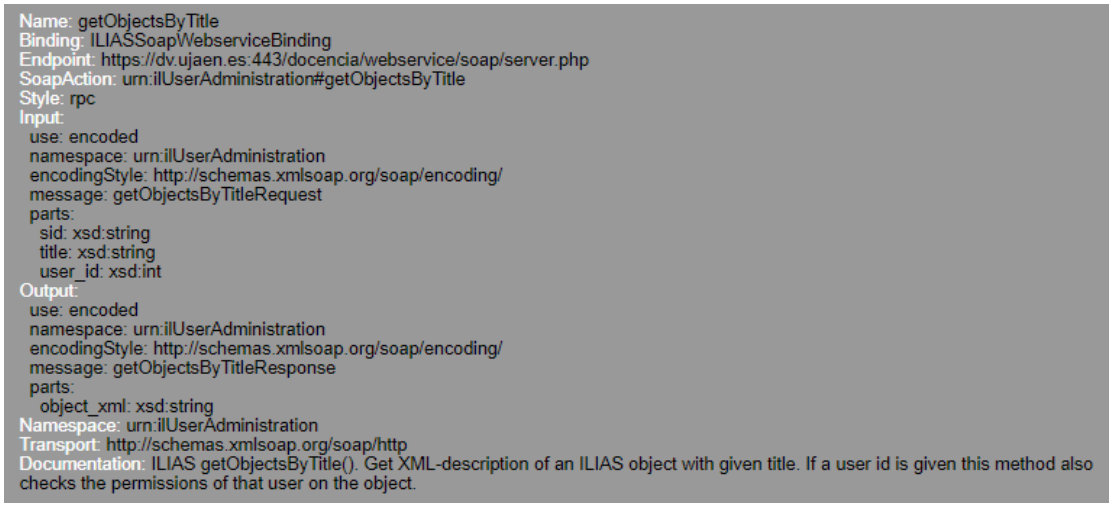


Ilustración 8. Información mostrada en el WSDL sobre el servicio "getObjectsByTitle". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php.

A continuación, se muestra un ejemplo de los mensajes intercambiados entre el cliente y el servidor para ejecutar el servicio web "*getObjectsByTitle*".

Petición del cliente

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ilUserAdministration">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:getObjectsByTitle
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <sid
xsi:type="xsd:string">795de233f521455087aa1343037b70c9::docencia</sid>
         <title xsi:type="xsd:string">banco de preguntas</title>
         <user_id xsi:type="xsd:int">822479</user_id>
      </urn:getObjectsByTitle>
   </soapenv:Body>
</soapenv:Envelope>
```

Respuesta del cliente

```
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:ilUserAdministration"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <ns1:getObjectsByTitleResponse>
         <object_xml xsi:type="xsd:string"><![CDATA[<?xml version="1.0"
encoding="utf-8"?><!DOCTYPE Objects PUBLIC "-//ILIAS//DTD ILIAS
Repositoryobjects//EN"
"https://dv.ujaen.es:443/docencia/xml/ilias_object_4_0.dtd"><!--Export of
ILIAS objects-->
<Objects>
<Object type="qpl" obj_id="1355214">
<Title>banco de preguntas</Title>
<Description></Description>
<Owner>822479</Owner>
```

```
<CreateDate>2020-01-19 16:58:50</CreateDate>
<LastUpdate>2020-01-19 16:58:50</LastUpdate>
<ImportId/>
<References ref_id="1047144" parent_id="1001899" accessInfo="granted">
<TimeTarget type="0">
<Timing starting_time="1579449540" ending_time="1579449540" visibility="0"/>
<Suggestion starting_time="1579449540" ending_time="1579449540"
changeable="0" earliest_start="1579449540" latest_end="1579474500"/>
</TimeTarget>
<Operation>visible</Operation>
<Operation>read</Operation>
<Operation>copy</Operation>
<Operation>write</Operation>
<Operation>delete</Operation>
<Path><Element ref_id="1" type="root">Espacios</Element>
<Element ref_id="30" type="cat">Escuela Politécnica Superior de
Linares</Element>
<Element ref_id="193681" type="cat">Grado en Ingeniería Telemática</Element>
<Element ref_id="351311" type="cat">3º Curso</Element>
<Element ref_id="351489" type="crs">Protocolos de Transporte</Element>
<Element ref_id="1001899" type="fold">TFG</Element>
</Path>
</References>
</Object>
</Objects>]]>
</object_xml>
      </ns1:getObjectsByTitleResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Tabla 4. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web "getObjectByTitle".

- **AddObject:** Servicio que permite la creación de un objeto (banco de preguntas, ejercicios…) en un nodo (asignaturas, carpetas, cursos…). Para poder crear un objeto es necesario definirlo en formato XML. Para conocer cómo deben estar formado la descripción XML ver "ilias_object_4_0.dtd". Este servicio web tiene como parámetros de entrada:
    - **Sid:** Este parámetro es el identificador de sesión que se obtiene cuando un usuario inicia sesión de manera satisfactoria en la plataforma mediante el servicio web "login".
    - **Target_id:** Este parámetro identifica el nodo donde se creará el objeto. Un ejemplo de "target_id" es 1047144.
    - **Object_xml:** Este parámetro tiene la descripción XML del objeto que se quiere crear. Para más detalle ver "ilias_object_4_0.dtd".

  Como salida, este servicio web devolverá el identificador de referencia del objeto creado. Un ejemplo de salida de este servicio web es "1047455".

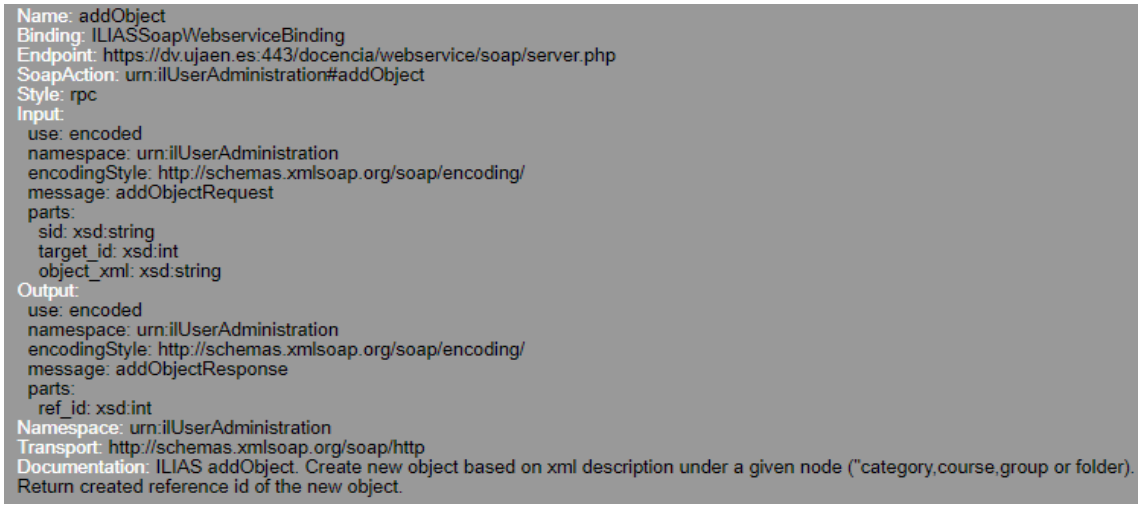


Ilustración 9. Información mostrada en el WSDL sobre el servicio "addObject". Fuente: https://dv.ujaen.es/docencia/webservice/soap/server.php.

A continuación, se muestra un ejemplo de los mensajes intercambiados entre el cliente y el servidor para ejecutar el servicio web "addObject".

Petición del cliente

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:ilUserAdministration">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:addObject
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <sid xsi:type="xsd:string">
795de233f521455087aa1343037b70c9::docencia</sidt>
         <target_id xsi:type="xsd:iny">1047144</target_id>
         <object_xml xsi:type="xsd:string">
<?xml version=\"1.0\" encoding=\"utf-8\"?><!DOCTYPE Objects PUBLIC \"-
//ILIAS//DTD ILIAS Repositoryobjects//EN\"
\"https://dv.ujaen.es/docencia/xml/ilias_object_4_0.dtd\"><!--Export of
ILIAS objects-->
<Objects>
<Object type="qpl" obj_id="1295301">
<Title>banco de preguntas creado con TestUJA</Title>
<Description></Description>
<Owner>822479</Owner>
<References parent_id="1047144" accessInfo="granted">
<Operation>visible</Operation>
<Operation>read</Operation>
<Operation>copy</Operation>
<Operation>write</Operation>
<Operation>delete</Operation>
</References>
</Object>
</Objects>
</object_xml>
      </urn:addObject>
   </soapenv:Body>
```

</soapenv:Envelope>
Respuesta del cliente
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="urn:ilUserAdministration" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body> <ns1:addObjectResponse> <ref_id xsi:type="xsd:int">1047455</ref_id> </ns1:addObjectResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

Tabla 5. Mensajes SOAP intercambiados entre cliente y servidor. Servicio web" addObject".

1.3.1.5 Estudio del formato de los bancos de preguntas.

Un banco de preguntas es un objeto utilizado por ILIAS para almacenar preguntas tipo test. Este objeto se representa en ILIAS de la siguiente forma:

Ilustración 10. Símbolo banco de preguntas

Para crear un test es necesario tener un banco de preguntas, si no dispone de ninguno, la plataforma pedirá crear uno por defecto, dado que toda pregunta de test debe estar ubicada en un banco de preguntas.

La ventaja principal que ofrecen los bancos de preguntas es que todas las preguntas tipo test encontradas en ellos se pueden reutilizar en distintos tests cuantas veces se quiera.

Debido a que en este proyecto se facilita la posibilidad de crear bancos de preguntas, se ha tenido que realizar un estudio previo del formato que deben tener.

Para la creación de bancos de preguntas se utilizará servicios web, concretamente se utilizará el servicio "addObject" que permite crear objetos (bancos de preguntas) en la plataforma de la Universidad de Jaén.

El formato del objeto debe estar especificado en lenguaje XML y debe seguir las marcas y pautas descritas en el DTD llamado "ilias_object_4_0.dtd" (8) que proporciona ILIAS.

El contenido del DTD "ilias_object_4_0.dtd" es el siguiente:

`ilias_object_4_0.dtd`

```
<!ELEMENT Objects (Object*)>

<!ELEMENT Object (Title, Description?, Owner, CreateDate?, LastUpdate?,
ImportId?, References*)>

<!ATTLIST Object

    type CDATA #REQUIRED
```

```
    obj_id CDATA #IMPLIED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Owner (#PCDATA)>
<!ELEMENT CreateDate (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT ImportId (#PCDATA)>
<!ELEMENT References (TimeTarget?,Operation*, Path?)>
<!--                              accesInfo                              is
'granted','no_permission',missing_precondition','no_object_access','no
_parent_access' or 'object_deleted' -->
<!ATTLIST References
    ref_id CDATA #REQUIRED
    parent_id CDATA #IMPLIED
    accessInfo(granted|no_permission|missing_precondition|
no_object_access|no_parent_access|object_deleted) #IMPLIED>
<!ELEMENT TimeTarget (Timing?,Suggestion?) >
<!--
Time target type is:
      0 => Deactivated (no time targets)
      1 => Temporarily available
      2 => Presettings enabled
Visibility normally used in combination with type=1
      0 => not visible outside availability
      1 => visible outside availability-->
<!ATTLIST TimeTarget
      type CDATA #REQUIRED>
<!-- Timing used in combination with timing type=1
      starting_time => Unix time of start
      ending_time => Unix time of end-->
<!ELEMENT Timing EMPTY>
<!ATTLIST Timing
      starting_time CDATA #REQUIRED
```

```
      ending_time CDATA #REQUIRED
      visibility CDATA #REQUIRED>
<!-- Presetting used in combination with timing type=2
      starting_time => Unix time of start
      ending_time => Unix time of end
      earliest_start => Unix time of earliest start
      latest_end => Unix time of latest end-->
<!ELEMENT Suggestion EMPTY>
<!ATTLIST Suggestion
      starting_time CDATA #REQUIRED
      ending_time CDATA #REQUIRED
      changeable CDATA #REQUIRED
      earliest_start CDATA #IMPLIED
      latest_end CDATA #IMPLIED>
<!ELEMENT Operation (#PCDATA)>
<!ELEMENT Path (Element*)>
<!ELEMENT Element (#PCDATA)>
<!ATTLIST Element
      ref_id CDATA #REQUIRED
      type CDATA #REQUIRED>
```

1.3.1.6 Estudio del protocolo HTTP y sus métodos para utilizarlos en ILIAS.

Debido a la falta de un servicio web que permita crear preguntas tipo test en un banco de preguntas, se ha tenido que buscar otra alternativa basada en el protocolo HTTP. Este método se basa en imitar las acciones que realiza el navegador a la hora de crear las preguntas tipo test.

Para ello, se ha tenido que realizar un estudio de todos los elementos que son necesarios para poder crear preguntas vía web, cómo deben estar formadas los diferentes tipos de preguntas (respuesta simple, múltiple, detectar errores…) y qué protocolo se utiliza.

A continuación, se explica paso por paso el estudio realizado.

- El primer paso, y el más sencillo, es detectar qué protocolo de comunicación se utiliza para crear preguntas vía web. Una vez comprobado que el protocolo que se utiliza es el protocolo HTTP (diseñado para comunicación entre navegadores y servidores web) se tiene que comprobar qué método se utiliza para crear preguntas y cómo debe estar formado.

- El segundo paso, es detectar qué método se utiliza para crear preguntas y cómo debe estar formado. Como se quiere crear preguntas vía web, y esto consiste en enviar información al servidor, el método que se utiliza es el método POST. Dependiendo del tipo de pregunta que se quiera crear, el método POST estará formado de una forma u otra. Por ejemplo, en las preguntas de tipo respuesta simple, respuesta múltiple y respuesta corta, el cuerpo de la petición POST estará formado por contenido de tipo **multipart** (9). Mientras que en las preguntas de tipo rellenar huecos y detectar errores el cuerpo de la petición POST estará formado por contenido de tipo **application/x-www-form-urlencoded**.
- El tercer paso, y último, es detectar qué elementos son necesarios para poder crear las preguntas vía web. Estos elementos son parámetros que deben ser enviados al servidor para que la pregunta pueda ser creada correctamente. Los parámetros necesarios son:
    - **RToken:** Este elemento es asignado al usuario una vez inicia sesión y permite que el usuario pueda realizar acciones en la plataforma.
    - **Question_ID:** Este elemento identifica la pregunta que se va a crear. La plataforma proporciona el **question_id** cuando es pulsado el botón "Crear pregunta".

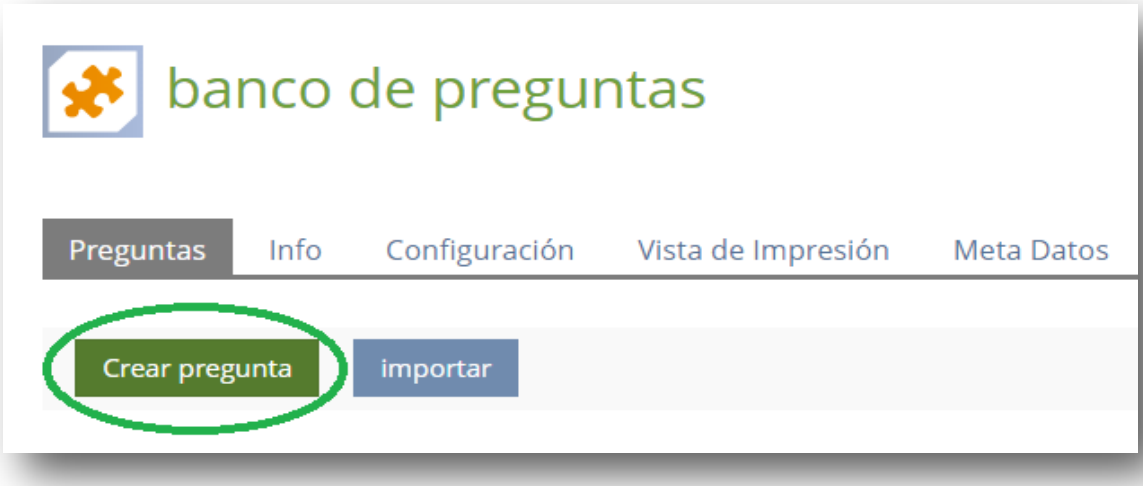


Ilustración 11. Botón Crear preguntas.

    - **Ref_id:** Este elemento identifica el objeto de ILIAS (banco de preguntas, en este caso) donde se van a crear las preguntas. La plataforma proporciona el **Ref_id** cuando se selecciona un objeto. Se puede obtener mediante servicios web.
    - **Sel_question_types:** Este elemento identifica el tipo de preguntas que se va a crear y se tiene que especificar en la *query* de la URL (10) de la petición HTTP. Los tipos disponibles son:
        - Respuesta única: **sel_question_types=assSingleChoice**
        - Respuesta múltiple: **sel_question_types=assMultipleChoice**
        - Detectar errores: **sel_question_types=assErrorText**
        - Rellenar huecos: **sel_question_types=assClozeTest**
        - Respuesta corta: **sel_question_types=assTextQuestion**

A continuación, se muestra un **ejemplo** de cómo está formada la petición HTTP cuando la plataforma crea una pregunta tipo test de rellenar huecos:

Método - URL

Post -
https://dv.ujaen.es/ilias.php?**ref_id=1002436&q_id=2009832&sel_question_types=assClozeTest**&cmd=post&cmdClass=**assclozetest**gui&cmdNode=st:m1:9&baseClass=ilRepositoryGUI&**rtoken=2c053d96a92ea8cd94a15e5071813ecc**

Cabeceras de petición:

Host: dv.ujaen.es

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0

Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate, br

Content-Type: application/x-www-form-urlencoded

Content-Length: 750

Origin: https://dv.ujaen.es

Connection: keep-alive

Referer:
https://dv.ujaen.es/ilias.php?**ref_id=1002436&q_id=2019426&sel_question_types=assClozeTest**&cmd=editQuestion&cmdClass=**assclozetest**gui&cmdNode=st:m1:9&baseClass=ilRepositoryGUI

Cookie:ilClientId=docencia;
PHPSESSID=a9f64a6a1676b40ab8984ecf594e400d

Upgrade-Insecure-Requests: 1

Content-Type: application/x-www-form-urlencoded

Cuerpo de la petición:

title=Titulo+rellenar+huecos&author=Antonio+Osuna+Melgarejo&comment=&question=pregunta+rellenar+huecos&myCounter=&Estimated%5Bhh%5D=0&Estimated%5Bmm%5D=1&Estimated%5Bss%5D=0&cloze_text=%5Bgap+1%5Dhueco+1%5B%2Fgap%5DEsto+es+un+%5Bgap+2%5Dhueco+2%5B%2Fgap%5D+.+Esto+es+un+segundo+%5Bgap+3%5Dhueco+3%5B%2Fgap%5D&myCounter=&textgap_rating=ci&fixedTextLength=&identical_scoring=1&gap_2%5Banswer%5D%5B0%5D=hueco+3&gap_2%5Bpoints%5D%5B0%5D=5&gap%5B2%5D=&clozetype=0&gap_a_numeric=0&gap_a_numeric_lower=0&gap_a_numeric_upper=0&gap_a_numeric_points=0&gap_a_gapsize=&best_possible_solution=0&clozetype_0=0&gap_0_gapsize=&gap_0%5Banswer%5D%5B0%5D=hueco+1&gap_0%5Bpoints%5D%5B0%5D=5&gap%5B0%5D=&clozetype_1=0&gap_1_gapsize=&gap_1%5Banswer%5D%5B0%5D=hueco+2&gap_1%5Bpoints%5D%5B0%5D=5&gap%5B1%5D=&clozetype_2=0&gap_2_gapsize=&gap_2%5Banswer%5D%5

B0%5D=hueco+3&gap_2%5Bpoints%5D%5B0%5D=5&gap%5B2%5D=&gap_json_post=%5B%5B%7B%22type%22%3A%22text%22%2C%22values%22%3A%5B%7B%22answer%22%3A%22hueco+1%22%2C%22points%22%3A%225%22%7D%5D%2C%22text_field_length%22%3A%22%22%7D%2C%7B%22type%22%3A%22text%22%2C%22values%22%3A%5B%7B%22answer%22%3A%22hueco+2%22%2C%22points%22%3A%225%22%7D%5D%7D%2C%7B%22type%22%3A%22text%22%2C%22values%22%3A%5B%7B%22answer%22%3A%22hueco+3%22%2C%22points%22%3A%225%22%7D%5D%7D%5D%5D&gap_json_combination_post=%5B%5D&cmd%5BsaveReturn%5D=Guardar+y+Volver

1.3.1.7 Estudio del lenguaje de programación y del entorno de trabajo utilizado.

Para el desarrollo software de este proyecto se utilizará el lenguaje de programación Java (11) y el entorno de trabajo Eclipse (12)..

Se ha elegido el lenguaje de programación Java porque tiene ciertas ventajas frente a otros lenguajes:

- **Independiente de la plataforma:** El mismo programa, diseñado con lenguaje Java, puede ser ejecutado tanto en un PC con el sistema operativo Windows, como en otro PC con Linux, como en otro PC con MAC OS.
- **Lenguaje orientado a objetos:** Java es un lenguaje orientado a objetos. Las ventajas de un lenguaje orientado a objetos es que agiliza el desarrollo software, permite crear sistemas más complejos, permite la reutilización de código, la programación es similar a la forma de pensar del ser humano.
- Además, algunas de las características más importantes de los lenguajes orientados a objetos son:
    - Herencia: Es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente y compartir automáticamente métodos y datos entre clases, subclases y objetos.
    - Abstracción: La abstracción consiste en seleccionar datos de un conjunto más grande para mostrar solo los detalles relevantes del objeto. Ayuda a reducir la complejidad y el esfuerzo de programación. En Java, la abstracción se logra usando clases e interfaces abstractas.
    - Encapsulación: La encapsulación proporciona la protección de datos. Es un principio del lenguaje java que consiste en hacer que los atributos de las clases se puedan editar sólo a través de métodos.
- **Liberación de memoria:** En java no existe problemas con la liberación de memoria en el sistema. En este lenguaje se decidió romper con el sistema tradicional de liberación de memoria, donde el programador era el encargado de esta tarea.
- **Lenguaje sencillo:** El lenguaje de programación Java es relativamente fácil de aprender en comparación con otros lenguajes de programación. Además, al ser un lenguaje orientado a objetos, es muy intuitivo.
- **Infinidad de librerías estándar:** Una de las características que más potencia aporta al lenguaje Java es que viene acompañado de una serie de librerías estándar que permiten realizar multitud de operaciones a la hora de programar.
- **Entornos de Desarrollo Integrado (IDE):** Hoy en día, existen excelentes IDE que permite ayudas a la hora de programar en lenguaje java, haciendo que el desarrollo de la aplicación sea más fluido. Un ejemplo de IDE es Eclipse.

- **Java es robusto:** Además de proporcionar un control de la memoria, permite gestionar los errores que se puedan producir en el código a través de excepciones.
- **Java es seguro:** Es un lenguaje que evita las posibles vulnerabilidades en las aplicaciones, las comunicaciones, o en la integridad de los datos. Prácticamente todos estos problemas están relacionados con el acceso a la memoria. Para corregir estos problemas el lenguaje Java proporciona métodos como:
    - Comprobación de rangos en el acceso a vectores.
    - Comprobación del comportamiento de las variables sin inicializar, para evitar que si se accediese a ellas se obtuviese un valor desconocido.
    - Eliminación de la liberación de la memoria por parte del programador, para evitar que el programador libere objetos que aún se necesitase.

Por otro lado, se tiene que elegir el entorno de desarrollo con el que se va a trabajar. Para este proyecto, se ha elegido el entorno de desarrollo integrado denominado Eclipse. Este entorno presenta ciertas ventajas frente a otros:

- Dispone de un editor de texto con resaltador de sintaxis para facilitar la labor de programación.
- La compilación es en tiempo real.
- Contiene gran cantidad de librerías que facilita al programador el desarrollo de la aplicación.

1.3.2 Desarrollos necesarios.

1.3.2.1 Desarrollo Software: Acceso a los servicios web de ILIAS

Como se ha comentado anteriormente, ILIAS ofrece sus servicios web a través del WSDL. Para comenzar con el diseño de la aplicación utilizando estos servicios web a través de un entorno de desarrollo integrado, que en este caso será Eclipse, se deben realizar los siguientes pasos:

1. Crear el esqueleto de los servicios a partir del WSDL. Se puede hacer de varias formas:
    a. Gracias a la herramienta de Apache Axis llamada WSDL2Java.
    b. Mediante Eclipse. Será la opción elegida y se deben seguir los siguientes pasos:
    Una vez creado un proyecto, se selecciona **File > New > Other**

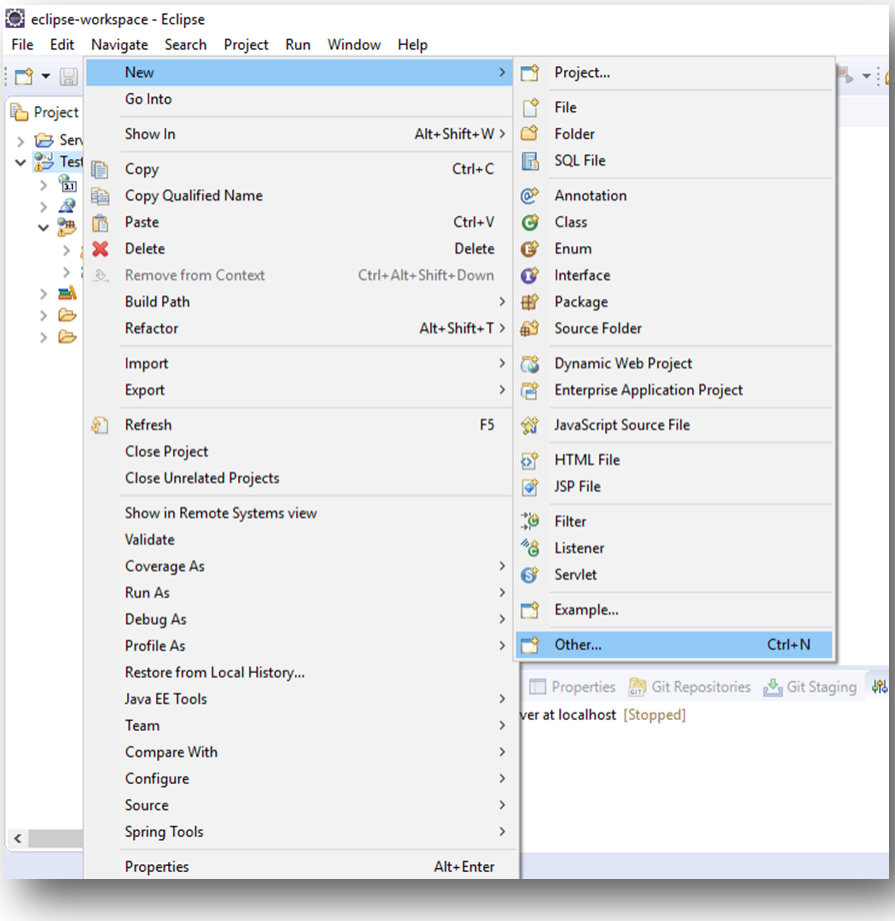


Ilustración 12. Crear esqueleto según WSDL. Paso 1.

A continuación, se selecciona **Web Service Client > Next.**

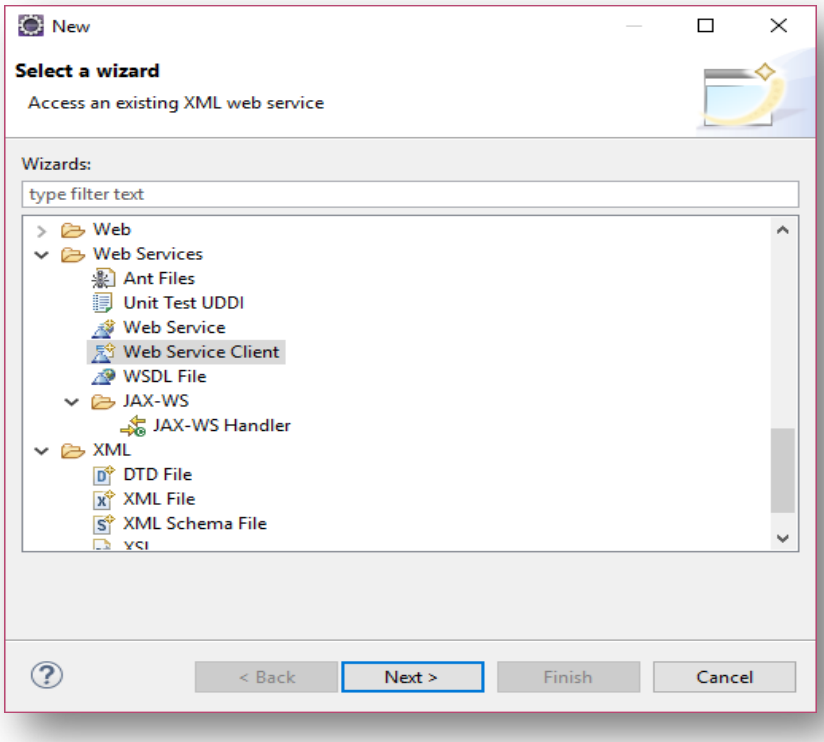


Ilustración 13. Crear esqueleto según WSDL. Paso 2.

Finalmente, se introduce la URL donde se ubican todos los servicios web (WSDL).

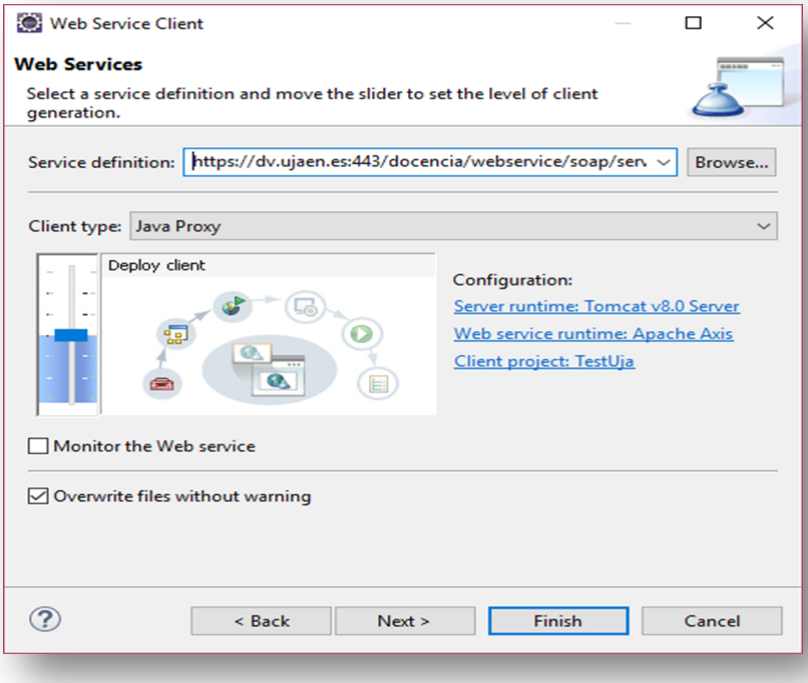


Ilustración 14. Crear esqueleto según WSDL. Paso 3.

Las dos herramientas antes mencionadas generan las clases que implementan el esqueleto de los servicios web a partir de un WSDL. Al usar la segunda opción, el entorno de desarrollo integrado genera un paquete llamado **llUserAdministration** en el que se encuentran todas las clases que se han generado. Las clases (cada una en un fichero) que son generadas por esta herramienta son:

- **ILIASSoapWebservice.java**
- **ILIASSoapWebserviceBindingStub.java**
- **ILIASSoapWebserviceLocator.java**
- **ILIASSoapWebservicePortType.java**
- **ILIASSoapWebservicePortTypeProxy.java**
- **llOperation.java**
- **llUserData.java.**

Una vez generado el esqueleto, es necesario introducir las librerías que permitan utilizar los servicios web en Eclipse a través del protocolo SOAP.

2. Incluir las clases y librerías necesarias para el correcto funcionamiento de los servicios web en ILIAS. Los servicios web de ILIAS usan el protocolo SOAP para realizar la comunicación cliente-servidor.
   La **librería SOAP** que permite el funcionamiento de los servicios web de ILIAS es: **ksoap2-j2se-full-2.1.2.jar.**

Esta librería junto con ejemplos de clases encargadas de establecer la comunicación entre el cliente y el servidor utilizando el protocolo SOAP se encuentran en un paquete que proporciona la plataforma de ILIAS que se llama **SOAPEXAMPLE.**

Después de haber realizado estos dos pasos, ya se podrá hacer uso de los servicios web desde el entorno con el que se trabaja (Eclipse). El cliente realizará peticiones al servidor a través de las clases definidas en el proyecto y el servidor le enviará el mensaje de respuesta correspondiente. Los mensajes intercambiados entre el cliente y servidor emplean el protocolo SOAP.

1.3.2.2 Desarrollo software: API Acceso a los servicios del protocolo HTTP

Como se ha comentado en el punto 1.3.1.6, no hay definido un servicio web que permita crear preguntas tipo test en un banco de preguntas por lo que ha sido necesario utilizar el protocolo HTTP. Para trabajar con el protocolo HTTP a través de Eclipse ha sido necesario:

- Incluir las clases y librerías adecuadas para el correcto funcionamiento del protocolo HTTP.
  Las **librerías HTTP** (13) que permiten el correcto funcionamiento del protocolo HTTP en Eclipse son:
  httpclient-4.5.10.jar, httpclient-cache-4.5.10.jar, httpclient-osgi-4.5.10.jar, httpclient-win-4.5.10.jar, httpcore-4.4.12.jar, httpmime-4.5.10.jar.
  Además de incluir estas librerías, tendremos que incluir en el código los "import" necesarios que permitan ejecutar conexiones HTTP, ejecutar métodos HTTP con contenido multipart…

Una vez se han incluido todas las librerías y clases necesarias, se podrá hacer uso del protocolo HTTP desde Eclipse. Desde la parte del cliente se realizarán peticiones HTTP (a través de los

métodos de este protocolo) al servidor. El servidor le responderá enviando un mensaje HTTP de respuesta.

1.3.2.3 Desarrollo software del proyecto: Uso de los servicios web de ILIAS y del protocolo HTTP.

Para el desarrollo software del proyecto se ha tenido que utilizar los servicios web de ILIAS y el protocolo HTTP. En primer lugar, para iniciar sesión, búsqueda de banco de preguntas y creación de bancos de preguntas se ha utilizado los servicios web, mientras que para la creación de las preguntas test se ha utilizado el protocolo HTTP.

El uso de los servicios web debe tener un orden porque muchos de ellos necesitan parámetros de entrada que solamente pueden ser obtenidos por otros servicios web.

A continuación, se explicará cómo ha sido diseñado este proyecto paso por paso:

1. El primer paso es el **inicio de sesión.** Para el inicio de sesión se utilizará el servicio web llamado **login**. Para este servicio es necesario introducir un **usuario** registrado en ILIAS y su **contraseña**. Como resultado de un inicio de sesión correcto, este servicio devolverá el parámetro **sid.** Este parámetro es muy importante porque permitirá ejecutar otros servicios web y será necesario para ejecutar peticiones HTTP, concretamente será necesario para las cabeceras de petición (cabecera cookie).
2. El segundo paso es seleccionar si desea **crear un banco de preguntas**, **buscar uno o validar un documento de texto**.
    a. Para la **búsqueda del banco de preguntas** se utilizará el servicio web **getObjectsByTitle,** el cual necesita como parámetros de entrada el **title** (título) del objeto (banco de preguntas) a buscar, el **sid** del usuario y el **user_id** que se obtiene del servicio web **getUserIdBySid.** El servicio web **getObjectsByTitle** devolverá el XML de los objetos encontrados con un título similar al introducido en el parámetro **title.** Como este servicio web puede encontrar cualquier tipo de objeto (carpetas, asignaturas, bancos de preguntas…) ha sido necesario crear una clase que permita analizar el XML obtenido por este servicio y solo mostrar los bancos de preguntas encontrados.
        - Si encuentra varios bancos de preguntas, será necesario utilizar otro servicio web para poder decidir cuál es el banco adecuado. Este servicio web será **getObjectByReference,** y permite encontrar un objeto de manera única según su identificador de referencia. **GetObjectByReference** necesita como parámetros de entrada el **sid** del usuario, el **user_id** que se obtiene del servicio web **getUserIdBySid** y el **ref_id** del objeto. Para seleccionar el **ref_id** del objeto que se están buscando se ha definido una clase que permite analizar el documento XML y obtener todos los **ref_ids** de los bancos de preguntas encontrados. Una vez elegido el **ref_id** del banco de preguntas que se está buscando, se procederá a su búsqueda.
        - Si solo encuentra un banco de preguntas no será necesario utilizar otro servicio web.
    b. Para la **creación del banco de preguntas** se utilizará el servicio web **addObject,** el cual necesita como parámetros de entrada el **sid** del usuario, el **target_id** (id donde se creará el banco de preguntas, en este proyecto, será el id de una carpeta) y el **object_xml**. El **object_xml** representa la descripción XML del banco de preguntas a crear y sigue el formato especificado en "**ilias_object_4_0.dtd"**. El servicio web **addObject** devolverá el **ref_id** del objeto creado. Para la creación de este

object_xml se ha diseñado una clase que desarrolla el XML del banco de preguntas que se quiere crear.

El proceso que sigue este proyecto para la creación del banco de preguntas es el siguiente:

- Primero se comprobará que la carpeta donde se quiere crear el banco existe. Para ello se utilizará el servicio web **getObjectsByTitle.** Como este servicio web puede encontrar cualquier tipo de objeto (carpetas, asignaturas, bancos de preguntas…) ha sido necesario crear una clase que permita analizar el XML obtenido por este servicio web y solo mostrar las carpetas encontradas.
- Si este servicio encuentra varias carpetas, será necesario utilizar otro servicio web para poder decidir cuál es la carpeta que se está buscando. Este servicio web será **getObjectByReference,** y permite encontrar un objeto de manera única según su identificador de referencia. **GetObjectByReference** necesita como parámetros de entrada el **sid** del usuario, el **user_id** que se obtiene del servicio web **getUserIdBySid** y el **ref_id** del objeto. Para seleccionar el **ref_id** del objeto se ha diseñado una clase que permite analizar el documento XML y obtener todos los **ref_ids** de las carpetas encontradas con nombres similares. Una vez elegido el **ref_id** de la carpeta que busca el usuario, se procederá a la creación del banco de preguntas.
- Si este servicio solo encuentra una carpeta, se procederá a la creación del banco de preguntas y no será necesario utilizar otro servicio web.

c. Para **validar un documento de texto** no será necesario utilizar ni servicios web ni el protocolo HTTP, ya que el programa TestUJA comprobará de **forma local** si las preguntas localizadas en el documento de texto cumplen el formato TESTUJAFORMAT o no.

3. El último paso es la creación de las preguntas tipo test en el banco de preguntas elegido mediante el protocolo HTTP. Para llevar a cabo este proceso, es necesario que las preguntas tipo test estén escritas en un **documento de texto plano (.txt)** según el formato TESTUJAFORMAT. Este formato será explicado más tarde. Para la creación de preguntas tipo test en este proyecto se siguen los siguientes pasos:
   - Primero se selecciona el documento donde se encuentran las preguntas tipo test definidas según el formato TESTUJAFORMAT.
   - A continuación, la aplicación irá leyendo el documento (.txt) para encontrar todos los elementos de las preguntas tipo test (títulos, puntuaciones, tipos de preguntas…).
   - Una vez que se ha detectado el tipo de pregunta de todas las preguntas encontradas en el documento, se procederá a la creación de éstas en el banco de preguntas utilizando el método POST del protocolo HTTP. Para poder ejecutar el método POST de forma correcta, y como bien se ha comentado en el apartado "**Estudio del protocolo HTTP y sus métodos para utilizarlos en ILIAS**" es necesario especificar algunos parámetros (ref_id, q_id y rTOKEN, cabecera Cookie, etc). Dependiendo del tipo de pregunta, el POST estará formado de una forma u otra. Por ejemplo, para las preguntas de tipo respuesta simple, el cuerpo del mensaje HTTP estará formado por contenido multipart.

1.3.2.4 Desarrollo de la interfaz gráfica. Java SWING.

Para el desarrollo de la interfaz gráfica de este proyecto se ha utilizado la biblioteca gráfica llamada Java SWING.

Java SWING es una herramienta de interfaz gráfica de usuario (GUI) que incluye un amplio conjunto de widgets (botones, cajas de texto, listas desplegables…). Esta herramienta permite la creación de interfaces de manera gráfica, introduciendo widgets utilizando el ratón y sin que el usuario tenga que desarrollar código. Una vez que el usuario introduce un widget en la interfaz, Java SWING genera el código en Java de manera automática.

Para poder utilizar Java SWING en Eclipse es necesario instalar un *plugin* denominado **WindowBuilder** (14).

Una vez que se ha instalado todo lo necesario para utilizar Java SWING en Eclipse, se procede al desarrollo de las interfaces gráficas de este proyecto.

Todas las interfaces gráficas que se han tenido que realizar, junto con las funciones que realiza, se muestra en la siguiente tabla:

Nombre de la clase que muestra la interfaz gráfica	**Función**
inicio.java	Muestra el menú inicial de la aplicación. Permite iniciar la aplicación.
Login.java	Muestra el inicio de sesión de la aplicación. Permite registrarse.
Principal.java	Muestra el menú principal de la aplicación. Permite la búsqueda de un banco de preguntas, la creación de uno o la validación de un documento de texto.
buscarBanco.java	Muestra la interfaz que permite buscar un banco de preguntas mediante su nombre.
buscarBancoRefid.java	Muestra la interfaz que permite buscar un banco de preguntas según su ref_id.
crearBanco.java	Muestra la interfaz que permite crear un banco de preguntas. Será necesario especificar la carpeta donde se creará dicho banco.
crearBancoRefId.java	Muestra la interfaz que permite crear un banco de preguntas. Será necesario especificar el ref_id de la carpeta donde se creará el banco de preguntas.
validarDocumento.java	Muestra la interfaz que permite comprobar si las preguntas localizadas en el documento de texto cumplen el formato TESTUJAFORMAT.
getQuestions.java	Muestra la interfaz que permite crear las preguntas test en un banco de preguntas. Será necesario especificar el directorio de documento donde están situadas las preguntas en el formato especificado.

Tabla 6. Nombres de las clases que muestran las interfaces Gráficas.

A continuación, se explican con más detalle estas clases:

Inicio.java

Esta clase muestra el menú inicial de la aplicación y solo permite iniciarla.

Ilustración 15. Menú inicial de la aplicación TestUJA.

Login.java

Esta clase permite iniciar sesión utilizando servicios web. Una vez que el usuario introduzca sus credenciales (usuario y contraseña) y pulse el botón "LOGIN", se ejecutará el servicio web "**login**". Dependiendo de si las credenciales son correctas o no, la aplicación mostrará un mensaje informando de la situación. El diagrama de flujo de este proceso se muestra en el Plano nº 1, denominado "**Plano inicio de sesión**"

Ilustración 16. Inicio de sesión de la aplicación TestUJA.

Principal.java

Esta clase muestra el menú principal de la aplicación. Permite al usuario elegir entre tres posibilidades:

- Buscar un banco de preguntas.
- Crear un banco de preguntas.
- Comprobar un documento.

El diagrama de flujo de este proceso se muestra en el Plano nº 2, denominado "**Plano menú principal**".

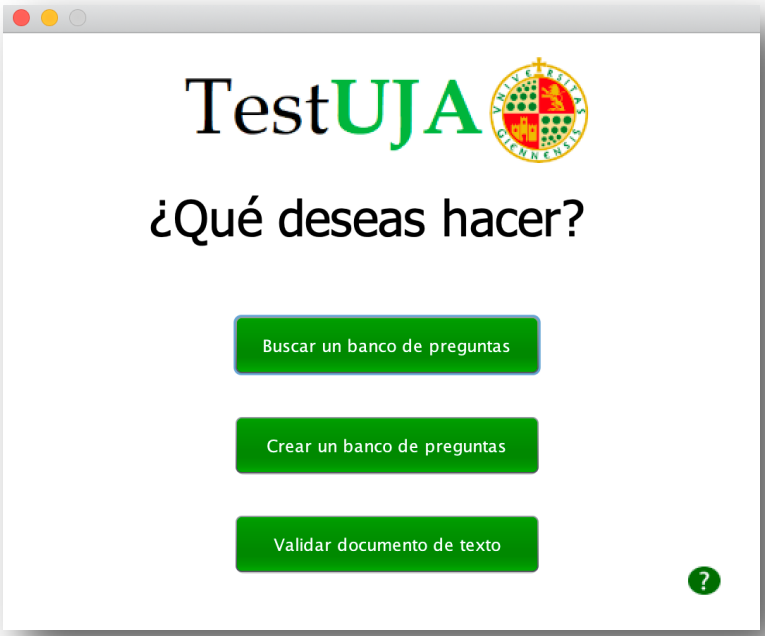


Ilustración 17. Menú principal de la aplicación TestUJA.

buscarBanco.java

Esta clase permite buscar un banco de preguntas utilizando servicios web o volver al menú principal. Una vez que el usuario introduzca el nombre del banco de preguntas y pulse el botón "Buscar banco de preguntas", se ejecutará el servicio web "**getObjectsByTitle**". Si al realizar la búsqueda del banco de preguntas el servicio web encuentra varios bancos de preguntas con nombres similares, será necesario especificar el banco de preguntas que desea el usuario utilizando el servicio web "**getObjectByReference**". Si solo encuentra un banco de preguntas, el usuario podrá comenzar a subir preguntas. Si no encuentra ningún banco de preguntas, la aplicación mostrará un mensaje de error. El diagrama de flujo de este proceso se muestra en el Plano nº 3, denominado "**Plano buscar banco de preguntas**".

Ilustración 18. Búsqueda del banco de preguntas de la aplicación TestUJA.

buscarBancoRefId.java

Esta clase permite buscar un banco de preguntas de forma única utilizando servicios web o volver al menú principal. Una vez que el usuario introduzca o seleccione (mediante el botón "Elegir Banco") el identificador del banco de preguntas y pulse el botón "Buscar banco de preguntas", se ejecutará el servicio web "**getObjectByReference**". Si al realizar la búsqueda del identificador del banco de preguntas el servicio web no lo encuentra, la aplicación mostrará un mensaje de error. Si el identificador del banco de preguntas es correcto, el usuario podrá comenzar a subir preguntas en ese banco de preguntas. El diagrama de flujo de este proceso se muestra en el Plano nº 4, denominado "**Plano buscar banco de preguntas según Ref_ID**".

Ilustración 19. Búsqueda del banco de preguntas según REF_ID de la aplicación TestUJA.

crearBanco.java

Esta clase permite crear un banco de preguntas utilizando servicios web o volver al menú principal. Una vez que el usuario introduzca los datos necesarios para crear un banco de preguntas y pulse el botón "Crear banco de preguntas", se ejecutará el servicio web "**addObject**". Previamente, la aplicación ejecutará el servicio web "**getObjectsByTitle**" para buscar todas las carpetas con nombre similar al introducido y a las que el usuario tiene acceso. Si este servicio web solo encuentra una carpeta, se creará el banco de preguntas con el nombre introducido por el usuario y podrá comenzar el proceso de subida de preguntas a ese banco de preguntas. Si el servicio web encuentra varias carpetas, será necesario especificar la carpeta que desea el usuario mediante el servicio web "**getObjectByReference**" para poder crear un banco de preguntas. Si el servicio web no encuentra ninguna carpeta, mostrará un mensaje de error El diagrama de flujo de este proceso se muestra en el Plano nº 5, denominado "**Plano crear un banco de pregunta**".

Ilustración 20. Creación de un banco de preguntas de la aplicación TestUJA.

crearBancoRefId.java

Esta clase permite crear un banco de preguntas utilizando servicios web o volver al menú principal. Es similar a la clase *crearBanco.java,* con la diferencia de que permite buscar de forma única una carpeta. Para ello, el usuario deberá introducir o seleccionar (mediante el botón "Elegir Carpeta") el identificador de la carpeta donde quiere crear un banco de preguntas. Una vez que el usuario introduzca los datos necesarios para crear un banco de preguntas según su REF_ID y pulse el botón "Crear banco de preguntas", se ejecutará el servicio web "**addObject**". Previamente, la aplicación ejecutará el servicio web "**getObjectsByReference**" para buscar la carpeta de forma única. Si encuentra la carpeta, se ejecutará el servicio web "**addObject**" para crear el banco de preguntas en la carpeta elegida. Si no encuentra la carpeta, la aplicación mostrará un mensaje de error. El diagrama de flujo de este proceso se muestra en el Plano nº 6, denominado "**Plano crear un banco de pregunta según REF_ID**".

Ilustración 21. Creación de un banco de preguntas según REF_ID de la aplicación TestUJA.

validarDocumento.java

Esta clase permite comprobar si el archivo especificado (ruta del archivo) por el usuario tiene preguntas tipo test con el formato TESTUJAFORMAT. Para introducir la ruta o el directorio del archivo, se proporciona un botón. En primer lugar, se comprobará si el archivo especificado existe. Si no existe, muestra un mensaje de error. Si el archivo existe, mostrará un mensaje indicando las preguntas correctas y las incorrectas. Para esta comprobación no será necesario utilizar ningún servicio web.

El diagrama de flujo de este proceso se muestra en el Plano nº 7, denominado "**Plano comprobar documento**".

Ilustración 22. Comprobación de documentos de la aplicación TestUJA.

subirPreguntas.java

Esta clase es muy similar a la clase *comprobarDocumentos.java,* con la diferencia de que aquí se produce la subida de las preguntas tipo test a un banco de preguntas. Una vez que el usuario introduzca la ruta del archivo y pulse el botón "Subir preguntas", comenzará a ejecutarse los métodos POST del protocolo HTTP para comenzar a subir las preguntas encontradas en el archivo. Una vez finalizado el proceso de subida, se mostrará un mensaje informando al usuario de las preguntas subidas.

El diagrama de flujo de este proceso se muestra en el Plano nº 8, denominado "**Plano subir preguntas**".

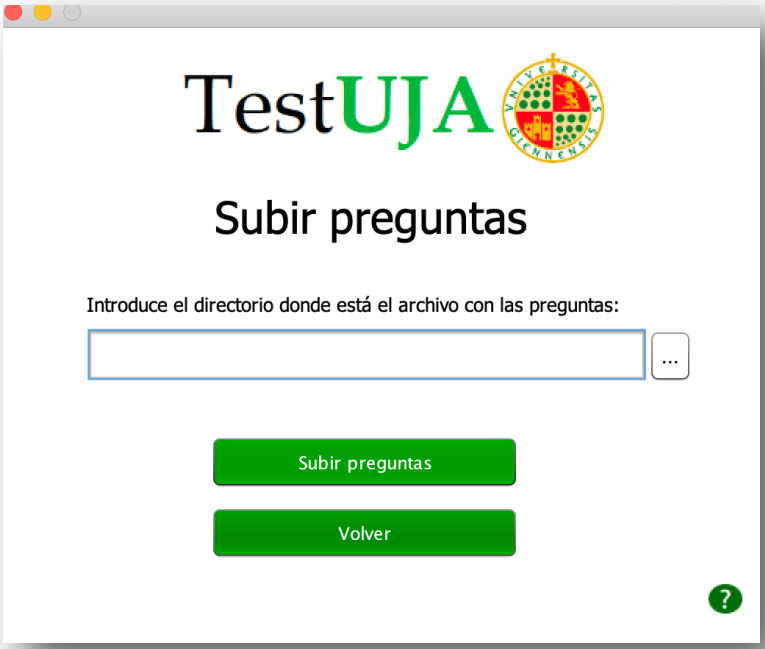


Ilustración 23. Subir preguntas de la aplicación TestUJA.

1.4 Resultado

El resultado de este proyecto es una aplicación que permite la creación de preguntas tipo test en un banco de preguntas elegido. Por otro lado, también permite la creación de bancos de preguntas en una carpeta elegida (situada en una asignatura).

Para el desarrollo de esta aplicación se tuvo que utilizar diferentes tecnologías a las elegidas en un comienzo debido a la falta de un servicio web que permitiese crear las preguntas en un banco de preguntas. A continuación, se muestra el proceso con errores y el proceso correcto que se llevó a cabo para el desarrollo de este proyecto.

1.4.1 Proceso correcto.

Este proceso viene especificado en el apartado **"Desarrollo software del proyecto: Uso de los servicios web de ILIAS y del protocolo HTTP."**.

1.4.2 Proceso con errores.

En un primer comienzo se intentó crear las preguntas tipo test utilizando los servicios web, sin embargo, surgieron algunos inconvenientes debido a la falta de un servicio web que permitiese crear preguntas tipo test en un banco de preguntas. A continuación, se muestra el proceso llevado a cabo:

1. El primer paso era el **inicio de sesión.** Para el inicio de sesión se utilizó el servicio web llamado **login**. Para este servicio era necesario introducir un **usuario** registrado en ILIAS y su **contraseña**. Como resultado de un inicio de sesión correcto, este servicio tenía como salida el parámetro **sid.** Este parámetro nos permitiría ejecutar otros servicios web.

2. El segundo paso era seleccionar si el usuario deseaba **crear un banco de preguntas**, **buscar uno o validar un documento de texto**.
    a. Para la **búsqueda del banco de preguntas** se utilizó el servicio web **getObjectsByTitle,** el cual necesitaba como parámetros de entrada el **title** (título) del objeto (banco de preguntas) a buscar, el **sid** del usuario que se obtenía del servicio web **login** y el **user_id** que se obtenía del servicio web **getUserIdBySid.** El servicio web **getObjectsByTitle** tenía como salida el XML de los objetos encontrados con un título similar al introducido en el parámetro **title.** Como este servicio web podía encontrar cualquier tipo de objeto (carpetas, asignaturas, bancos de preguntas…) fue necesario crear una clase que permitiese analizar el XML obtenido por este servicio y mostrar solo los bancos de preguntas encontrados.
        - Si encontraba varios bancos de preguntas, sería necesario utilizar otro servicio web para que el usuario pudiera decidir cuál era el banco adecuado. Este servicio web era **getObjectByReference,** y permitía encontrar un objeto según su identificador de referencia. **GetObjectByReference** necesitaba como parámetros de entrada el **sid** del usuario, el **user_id** que se obtiene del servicio web **getUserIdBySid** y el **ref_id** del objeto. Para seleccionar el **ref_id** del objeto que buscaba el usuario se definió una clase que permitía analizar el documento XML y obtener todos los **ref_ids** de los bancos de preguntas encontrados. Una vez fuese elegido el **ref_id** del banco de preguntas que buscaba el usuario, se procedía a su búsqueda.
        - Si solo encontraba un banco de preguntas no sería necesario utilizar otro servicio web.
    b. Para la **creación del banco de preguntas** se utilizó el servicio web **addObject,** el cual necesitaba como parámetros de entrada el **sid** del usuario, el **target_id** (id donde se creará el banco de preguntas, en este proyecto, será el id de una carpeta) y el **object_xml**. El **object_xml** representaba la descripción XML del banco de preguntas a crear y seguía el formato especificado en **ilias_object_4_0.dtd**. El servicio web **addObject** devolverá el **ref_id** del objeto creado. Para la creación de este **object_xml** se diseñó una clase que desarrollaba el XML del banco de preguntas que se quiere crear.

        El proceso que seguía este proyecto para la creación del banco de preguntas fue el siguiente:
        - Primero se comprobaba que la carpeta donde el usuario quería crear el banco existía. Para ello se utilizó el servicio web **getObjectsByTitle.** Como este servicio web podía encontrar cualquier tipo de objeto (carpetas, asignaturas, bancos de preguntas…) fue necesario crear una clase que permitiese analizar el XML obtenido por este servicio web y solo mostrar las carpetas encontradas.
        - Si este servicio encontraba varias carpetas, sería necesario utilizar otro servicio web para que el usuario pudiera decidir cuál fue la carpeta que estaba buscando. Este servicio web fue **getObjectByReference,** y permitía encontrar un objeto según su identificador de referencia. **GetObjectByReference** necesitaba como parámetros de entrada el **sid** del usuario, el **user_id** que se obtenía del servicio web **getUserIdBySid** y el **ref_id** del objeto. Para seleccionar el **ref_id** del objeto se ha diseñado una clase que permitía analizar el documento XML y obtener todos los **ref_ids**

de las carpetas encontradas con nombres similares. Una vez fuese elegido el **ref_id** de la carpeta que buscaba el usuario, se procedía a la creación del banco de preguntas.

- Si este servicio solo encontraba una carpeta, se procedía a la creación del banco de preguntas.

c. Para **validar un documento de texto** no sería necesario utilizar ni servicios web ni el protocolo HTTP, ya que el programa TestUJA comprobaría de **forma local** si las preguntas localizadas en el documento de texto cumplían el formato TESTUJAFORMAT o no.

3. El último paso, **y el erróneo**, era la creación de las preguntas tipo test en el banco de preguntas elegido. Para ello, se utilizó el servicio web **addObject,** el cual necesitaba como parámetros de entrada el **sid** del usuario, el **target_id** (id donde se creará el banco de preguntas, en este proyecto, será el id de una carpeta) y el **object_xml**. Se intentó definir las preguntas según el formato IMS QTI en el **object_xml**, pero no funcionó debido a que los objetos en ILIAS deben seguir el formato especificado en el dtd **ilias_object_4_0.dtd.** Por otro lado, se utilizó el servicio web **updateObject** para intentar modificar alguna pregunta ya subida, pero no funcionó tampoco. Este servicio web solo permitía modificar el nombre y la descripción de los objetos de ILIAS (bancos de preguntas, carpetas, asignaturas y cursos). También se intentó con otros servicios web como **addExercise o saveQuestionResult,** sin embargo, estos tampoco permitían la creación de preguntas tipo test.
   Debido a estos inconvenientes a la hora de crear preguntas test utilizando los **servicios web**, se optó por **buscar otra alternativa**, y esta fue el protocolo **HTTP**.

1.5 Formato de las preguntas tipo test. TESTUJAFORMAT.

Antes de conocer cómo deben estar formadas las preguntas tipo test, se mostrarán qué palabras claves están reservadas y cómo se deben utilizar.

Palabras claves comunes.

Las siguientes palabras claves son comunes a todas las preguntas tipo test desarrolladas en este proyecto:

- **“Title:”:** Representa el título de la pregunta y se debe escribir al comienzo de una línea.
- **“Question:**”: Representa la pregunta y se debe escribir al comienzo de una línea.
- **“Author:”:** Representa el autor de las preguntas y se debe escribir al comienzo de una línea. Solo se debe introducir una vez.

Palabras claves no comunes.

Las siguientes palabras claves o caracteres no comunes **identifican las puntuaciones** en los diferentes tipos de preguntas test desarrolladas en este proyecto:

- **“( )~**”: Identifica la puntuación de una respuesta en las preguntas de tipo **Respuesta Única** y se debe escribir al comienzo de una línea.
  - **“*”:** Se utiliza en las preguntas de tipo **Respuesta Única** para identificar que una respuesta es correcta. Se debe escribir antes de la puntuación y solo puede haber un “*” por pregunta.

- **“(,)~”**: Identifica las puntuaciones de una respuesta en las preguntas de tipo **no Respuesta Única** y se debe escribir al comienzo de una línea.

Nota: Si el usuario introduce en la misma pregunta diferentes tipos de puntuaciones, la aplicación marcará como errónea esa pregunta.

Las siguientes palabras claves o caracteres no comunes **identifican qué tipo de pregunta test** es:

- **“#”:** Carácter empleado en las respuestas de tipo **Seleccionar Error** para identificar el error. Se puede introducir tantos errores como desee el usuario.
- **“[gap] [/gap]”:** Conjunto de palabras que se utilizan en las respuestas de tipo **Rellenar Hueco** para identificar el hueco. Se puede introducir tantos huecos como desee el usuario.
- **“[Rcor][/Rcor]”**: Conjunto de palabras que se utilizan en las respuestas de tipo **Respuesta Corta** para identificar la respuesta corta. Solo se debe introducir una respuesta corta por pregunta.

Nota: Si el usuario introduce en la misma pregunta diferentes palabras no comunes que identifican el tipo de pregunta, la aplicación marcará como errónea esa pregunta.

Otros caracteres.

- **Carácter “~”:** Este carácter se utiliza para identificar el final de una respuesta.

Las preguntas tipo test deben seguir el siguiente orden:

1. **Título de la pregunta.**
2. **Pregunta.**
3. **Respuestas.**

1.5.1 Pregunta Respuesta Única

Su formato es el siguiente:

- **Título de la pregunta:** Se debe utilizar la palabra clave “**Title:**” al comienzo de una línea. El título de la pregunta comienza después de la palabra clave “**Title:”** y finaliza cuando un usuario introduce un salto de línea junto al inicio de la palabra clave “**Question:**”. Un ejemplo es: **Title:** Métodos HTTP → El título de la pregunta es **Métodos HTTP**.
- **Pregunta:** Se debe utilizar la palabra clave “**Question:**” al comienzo de una línea. La pregunta comienza después de la palabra clave “**Question:**” y finaliza cuando un usuario introduce un salto de línea junto al inicio de la puntuación de la respuesta de la pregunta. Un ejemplo es: **Question:** ¿Cuáles de las siguientes palabras son métodos http? → La pregunta es **¿Cuáles de las siguientes palabras son métodos http?**
- **Respuestas:** Todas las respuestas están compuestas por una puntuación y por una respuesta.
    - **Puntuación:** La puntuación se define mediante un único valor numérico encerrado entre paréntesis y seguido del carácter ~. El carácter ~ debe ir seguido del paréntesis cerrado sin dejar espacio, si hubiera un espacio, no se obtendrá la respuesta. Un ejemplo es: (5)~ → 5 puntos.
    - **Respuesta:** La respuesta es el texto que se introduce después de la puntuación y finaliza con el carácter ~ (el usuario puede introducir tantos saltos de línea como sean necesarios, con la única condición de que al final del texto introduzca

el carácter ~). Si las respuestas no finalizan con el carácter ~, la aplicación no obtendrá la respuesta. Un ejemplo es: GET ~ → la respuesta es **GET**.

Para este tipo de pregunta, será necesario especificar cuál es la respuesta correcta. Para ello, se utilizará el carácter * al comienzo de la puntuación de la respuesta. Además, la puntuación que contiene el carácter * deberá ser un valor positivo y la puntuación que no contiene el carácter * deberá ser un valor negativo.

A continuación, se muestran algunos ejemplos de preguntas bien formuladas y preguntas mal formuladas:

Pregunta bien formulada:

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método http?

*(5)~ GET ~

(-5)~ GOT ~

(-5)~ PORT ~
```

Pregunta mal formulada - Dos respuestas correctas:

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método http?

*(5)~ GET ~

*(-5)~ GOT ~

(-5)~ PORT ~
```

Pregunta mal formulada - Sin respuesta correcta:

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método http?

(5)~ GET ~

(-5)~ PORT ~
```

Pregunta mal formulada - Una sola respuesta:

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método http?

*(5)~ GET ~
```

Por último, se muestra una imagen explicando la diferencia que hay entre las palabras claves “Title” y “Question”:

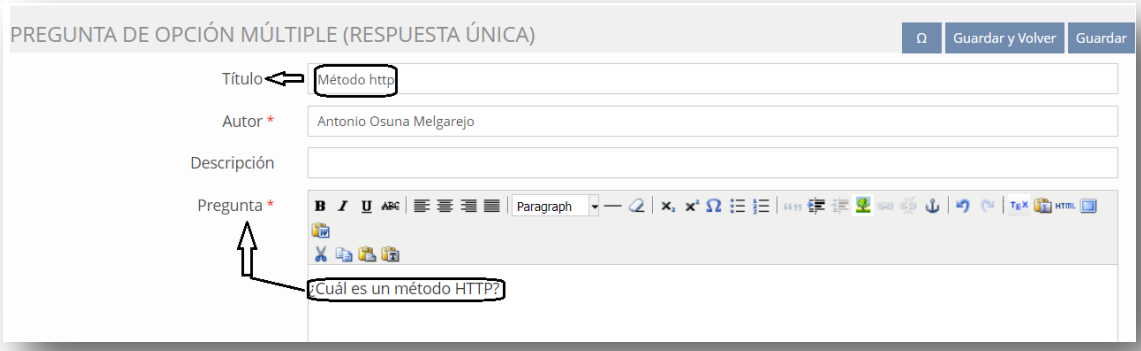


Ilustración 24. Diferencia entre Title y Question.

1.5.2 Pregunta Respuesta Múltiple

Su formato es el siguiente:

- **Título de la respuesta:** Se debe utilizar la palabra clave "**Title:**" al comienzo de una línea. El título de la pregunta comienza después de la palabra clave "**Title:"** y finaliza cuando un usuario introduce un salto de línea junto al inicio de la palabra clave "**Question:**". Un ejemplo es: **Title:** Métodos HTTP → El título de la pregunta es **Métodos HTTP**.
- **Pregunta:** Se debe utilizar la palabra clave "**Question:**" al comienzo de una línea. La pregunta comienza después de la palabra clave "**Question:**" y finaliza cuando un usuario introduce un salto de línea junto al inicio de la puntuación de la respuesta de la pregunta. Un ejemplo es: **Question:** ¿cuáles de las siguientes palabras son métodos http? → La pregunta es **¿Cuáles de las siguientes palabras son métodos http?**
- **Respuestas:** Todas las respuestas están compuestas por una puntuación y por una respuesta.
    - **Puntuación:** La puntuación se define mediante dos valores numéricos separados por una coma y encerrados entre paréntesis. Seguido del paréntesis cerrado, y sin dejar espacio, debe haber un carácter ~. Si el usuario deja un espacio entre el paréntesis cerrado y el carácter ~, la aplicación no obtendrá la puntuación. Un ejemplo de puntuación es: (5,-5)~ , donde el primer valor corresponde a la respuesta marcada, que en este caso son 5 puntos, mientras que el segundo valor corresponde a la respuesta no marcada, que en este caso son -5 puntos.
    - **Respuesta:** La respuesta es el texto que se introduce después de la puntuación y finaliza con el carácter ~ (el usuario puede introducir tantos saltos de línea como sean necesarios, con la única condición de que al final del texto introduzca el carácter ~). Si las respuestas no finalizan con el carácter ~, la aplicación no obtendrá la respuesta. Un ejemplo es: GET ~ → la respuesta es **GET**.

A continuación, se muestran algunos ejemplos de preguntas bien formuladas y preguntas mal formuladas:

Pregunta bien formulada:

```
Title: Métodos HTTP
```

```
Question: ¿Cuáles de las siguientes palabras es un método
http?

(2.5,-2.5)~ GET ~

(2.5,-2.5)~ POST ~

(-5,5)~ PORT ~
```

La puntuación total de esta pregunta será de 10 puntos si todas las respuestas marcadas y no marcadas son correctas.

Pregunta mal formulada – Respuestas correctas con el carácter *:

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método
http?

*(5,-5)~ GET ~

*(5,-5)~ POST ~

(-5,5)~ PORT ~
```

Pregunta mal formulada – Respuestas con caracteres especiales pertenecientes a otro tipo de pregunta.

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método
http?

(5,-5)~ #GET ~

(5,-5)~ POST ~

(-5,5)~ PORT ~
```

Pregunta mal formulada – Respuestas con caracteres especiales pertenecientes a otro tipo de pregunta.

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método
http?

(5,-5)~ [gap] GET [/gap] ~

(-5,5)~ PORT ~
```

Pregunta mal formulada – Respuestas con puntuaciones mezcladas.

```
Title: Métodos HTTP

Question: ¿Cuáles de las siguientes palabras es un método
http?
```

```
(5)~ GET ~

(-5,5)~ PORT ~
```

1.5.3 Pregunta Seleccionar Error (es)

Su formato es el siguiente:

- **Título de la pregunta:** Se debe utilizar la palabra clave "**Title:**" al comienzo de una línea. El título de la pregunta comienza después de la palabra clave "**Title:"** y finaliza cuando un usuario introduce un salto de línea junto al inicio de la palabra clave "**Question:**". Un ejemplo es: **Title:** Métodos HTTP → El título de la pregunta es **Métodos HTTP**.
- **Pregunta:** Se debe utilizar la palabra clave "**Question:**" al comienzo de una línea. La pregunta comienza después de la palabra clave "**Question:**" y finaliza cuando un usuario introduce un salto de línea junto al inicio de la puntuación de la respuesta de la pregunta. Un ejemplo es: **Question:** ¿cuáles de las siguientes palabras son métodos http? → La pregunta es **¿Cuáles de las siguientes palabras son métodos http?**
- **Respuestas:** Todas las respuestas están compuestas por una puntuación y por una respuesta.
    - **Puntuación:** La puntuación se define mediante dos valores numéricos separados por una coma y encerrados entre paréntesis. Seguido del paréntesis cerrado, y sin dejar espacio, debe haber un carácter ~. Si el usuario deja un espacio entre el paréntesis cerrado y el carácter ~, la aplicación no obtendrá la puntuación. Un ejemplo de puntuación es: (5,-5)~ → 5 puntos para la palabra errónea seleccionada correctamente y -5 para la palabra seleccionada que no es un error.
    - **Respuestas:** La respuesta es el texto que se introduce después de la puntuación y finaliza con el carácter ~ (el usuario puede introducir tantos saltos de línea como sean necesarios, con la única condición de que al final del texto introduzca el carácter ~). Si las respuestas no finalizan con el carácter ~, la aplicación no obtendrá la respuesta. **Las palabras erróneas se marcarán con el carácter #.** Un ejemplo es: #PORT es un método http. ~ → la palabra errónea es **PORT**.

A continuación, se muestran algunos ejemplos de preguntas bien formuladas y preguntas mal formuladas:

Pregunta bien formulada:

```
Title: Métodos HTTP

Question: Selecciona los errores de la siguiente frase:

(5,-5)~ #PORT y #GOT son métodos http. ~
```

La puntuación total de esta pregunta será de 10 puntos si todas las palabras erróneas son marcadas correctamente.

Pregunta mal formulada – Respuestas con caracteres especiales pertenecientes a otro tipo de pregunta.

```
Title: Métodos HTTP

Question: Selecciona los errores de la siguiente frase:
```

```
*(5,-5)~ #PORT y #GOT son métodos http. ~
```

Pregunta mal formulada – Dos respuestas diferentes con el carácter #:

```
Title: Métodos HTTP

Question: Selecciona los errores de las siguientes frases:

(5,-5)~ #PORT y #GOT son métodos http. ~

(5,-5)~ #HEAR y #GUT son métodos http. ~
```

Solo será posible introducir UNA RESPUESTA en este tipo de pregunta, es decir, si se define dos respuestas que tengan errores será marcada como errónea

1.5.4 Pregunta Rellenar hueco(s)

Su formato es el siguiente:

- **Título de la pregunta:** Se debe utilizar la palabra clave "**Title:**" al comienzo de una línea. El título de la pregunta comienza después de la palabra clave "**Title:"** y finaliza cuando un usuario introduce un salto de línea junto al inicio de la palabra clave "**Question:**". Un ejemplo es: **Title:** Métodos HTTP → El título de la pregunta es **Métodos HTTP**.
- **Pregunta:** Se debe utilizar la palabra clave "**Question:**" al comienzo de una línea. La pregunta comienza después de la palabra clave "**Question:**" y finaliza cuando un usuario introduce un salto de línea junto al inicio de la puntuación de la respuesta de la pregunta. Un ejemplo es: **Question:** ¿cuáles de las siguientes palabras son métodos http? → La pregunta es **¿Cuáles de las siguientes palabras son métodos http?**
- **Respuestas:** Todas las respuestas están compuestas por una puntuación y por una respuesta.
    - **Puntuación:** La puntuación se define mediante dos valores numéricos separados por una coma y encerrados entre paréntesis. Seguido del paréntesis cerrado, y sin dejar espacio, debe haber un carácter ~. Si el usuario deja un espacio entre el paréntesis cerrado y el carácter ~, la aplicación no obtendrá la puntuación. Un ejemplo de puntuación es: (5,-5)~ → 5 puntos para la palabra escrita correctamente en el hueco y -5 para la palabra escrita incorrectamente en el hueco.
    - **Respuesta:** La respuesta es el texto que se introduce después de la puntuación y finaliza con el carácter ~ (el usuario puede introducir tantos saltos de línea como sean necesarios, con la única condición de que al final del texto introduzca el carácter ~). Si las respuestas no finalizan con el carácter ~, la aplicación no obtendrá la respuesta. **Los huecos se declaran mediante el conjunto de elementos [gap][/gap]**. Un ejemplo es: GET es un método [gap] http [/gap]. ~ → En ILIAS, el hueco deberá ser rellenado con la palabra **HTTP o http**.

A continuación, se muestran algunos ejemplos de preguntas bien formuladas y preguntas mal formuladas:

Pregunta bien formulada:

```
Title: Métodos HTTP

Question: Rellena los huecos correctamente:
```

```
(5,-5)~ GET es un método [gap] http [/gap] y el método [gap] POST
[/gap] permite enviar información al servidor. ~
```

La puntuación total de esta pregunta será de 10 puntos si todos los huecos son rellenados de forma correcta.

Pregunta mal formulada – Respuestas con caracteres especiales escritos de forma incorrecta:

```
Title: Métodos HTTP

Question: Rellena los huecos correctamente:

(5,-5)~ GET es un método [/gap] http [gap] y el método [gap] POST
permite enviar información al servidor. ~
```

Pregunta mal formulada – Dos respuestas diferentes con los elementos [gap][/gap]:

```
Title: Métodos HTTP

Question: Rellena los huecos correctamente:

(5,-5)~ GET es un método [gap] http [/gap] ~

(5,-5)~ POST es un método [gap] http [/gap] ~
```

Solo será posible introducir UNA RESPUESTA en este tipo de pregunta, es decir, si se definen dos respuestas que tengan huecos será marcada como errónea. Pregunta mal formulada – Respuestas con caracteres especiales pertenecientes a otro tipo de pregunta.

```
Title: Métodos HTTP

Question: Selecciona los errores de la siguiente frase:

(5,-5)~ #GET es un método [gap] http [/gap] ~
```

1.5.5 Pregunta Respuesta Corta

Su formato es el siguiente:

- **Título de la pregunta:** Se debe utilizar la palabra clave "**Title:**" al comienzo de una línea. El título de la pregunta comienza después de la palabra clave "**Title:"** y finaliza cuando un usuario introduce un salto de línea junto al inicio de la palabra clave "**Question:**". Un ejemplo es: **Title:** Métodos HTTP → El título de la pregunta es **Métodos HTTP**.
- **Pregunta:** Se debe utilizar la palabra clave "**Question:**" al comienzo de una línea. La pregunta comienza después de la palabra clave "**Question:**" y finaliza cuando un usuario introduce un salto de línea junto al inicio de la puntuación de la respuesta de la pregunta. Un ejemplo es: **Question:** ¿cuáles de las siguientes palabras son métodos http? → La pregunta es **¿Cuáles de las siguientes palabras son métodos http?**
- **Respuestas:** Todas las respuestas están compuestas por una puntuación y por una respuesta.
    - **Puntuación:** La puntuación se define mediante dos valores numéricos separados por una coma y encerrados entre paréntesis. Seguido del paréntesis cerrado, y sin dejar espacio, debe haber un carácter ~. Si el usuario deja un espacio entre el paréntesis cerrado y el carácter ~, la aplicación no obtendrá la puntuación. Un ejemplo de puntuación es: (5,-5)~ → 5 puntos para la respuesta

corta escrita correctamente y -5 para la respuesta corta escrita incorrectamente.

- **Respuesta:** La respuesta es el texto que se introduce después de la puntuación y finaliza con el carácter ~ (el usuario puede introducir tantos saltos de línea como sean necesarios, con la única condición de que al final del texto introduzca el carácter ~). Si las respuestas no finalizan con el carácter ~, la aplicación no obtendrá la respuesta. **Las respuestas cortas se declaran mediante el conjunto de elementos [Rcor][/Rcor]**. Un ejemplo es: GET es un método [Rcor] http [/Rcor]. ~ → la respuesta corta deberá ser rellenada con la palabra **http o HTTP**).

A continuación, se muestran algunos ejemplos de preguntas bien formuladas y preguntas mal formuladas:

Pregunta bien formulada:

```
Title: Métodos HTTP

Question: GET es un método

(5,-5)~ [Rcor] http [/Rcor] ~
```

La puntuación total de esta pregunta será de 5 puntos si la respuesta corta es rellenada de forma correcta.

Pregunta mal formulada - Respuestas con caracteres especiales pertenecientes a otro tipo de pregunta:

```
Title: Métodos HTTP

Question: GET es un método

*(5,-5)~ [Rcor] http [/Rcor] ~
```

Pregunta mal formulada – Dos respuestas diferentes con los elementos [Rcor][/Rcor]:

```
Title: Métodos HTTP

Question: GET es un método

(5,-5)~[Rcor] http [/Rcor] ~

(5,-5)~ [Rcor] método http [/Rcor] ~
```

Solo será posible introducir UNA RESPUESTA CORTA por cada pregunta, es decir, si se define dos respuestas que tengan una respuesta corta será marcada como errónea o si una respuesta contiene dos respuestas cortas también será marcada como errónea.

1.5.6 Ejemplo de fichero TXT enviado por los alumnos.

```
Author: Antonio Osuna Melgarejo

Title: Metodos http 1

Question: Completa el hueco

(5 , -5)~ El método [gap]POST[/gap] permite [gap]enviar[/gap]
desde [gap]el cliente[gap] al servidor ~
```

```
Title: Metodos http 2

Question: ¿Cuál es un método http?

*(2)~ El método GET ~

(-2)~ GOT ~

(-2)~ PORT ~


Title: Metodos http 3

Question: ¿Cuáles son métodos http?

(5, -5)~ GET ~

(-5,5)~ GOT ~

(5, -5)~ POST ~


Title: Metodos http 4

Question: Encuentra el error

(5 , -5)~ Con el método #get podemos enviar información desde el
cliente al servidor ~


Title: Metodos http 5

Question: ¿Qué método envía información desde el cliente para que
sea procesada en el servidor?

(5,0)~ [Rcor]método POST[/Rcor] ~
```

1.6 Conclusiones y líneas de futuro.

1.6.1 Conclusiones

Haciendo uso de los servicios web proporcionados por ILIAS se pretendía conseguir los objetivos de este proyecto. Sin embargo, debido a algunos inconvenientes a la hora de crear las preguntas test utilizando los servicios web se tuvo que utilizar los métodos del protocolo HTTP. Gracias al uso de estas dos tecnologías se facilita un método eficiente y automatizado a los usuarios de ILIAS para permitirles crear preguntas tipo test en un banco de preguntas.

Una vez finalizado el desarrollo del proyecto, se puede evaluar el cumplimiento de los objetivos de este proyecto.

1.6.1.1 Objetivos principales.

Creación de preguntas tipo test en un banco de preguntas

- Las preguntas tipo test, obtenidas de un documento de texto, se crearán en el banco de preguntas (ya creado en la plataforma) que desee el usuario, siempre y cuando, éste tenga acceso con privilegios de administración en dicho banco.
    - Para que sea posible crear preguntas en un banco de preguntas será necesario tener una cuenta de usuario de la Universidad de Jaén.
    - El protocolo utilizado para la creación de las preguntas tipo test en un banco de preguntas será el protocolo HTTP, concretamente se utilizará el método POST.
    - Para poder buscar un banco de preguntas donde crear las preguntas será necesario utilizar los servicios web de ILIAS para comunicarse con la plataforma.
- **Seguridad**.
    - Una de las principales medidas de seguridad es evitar casos de suplantación de identidad. Para evitar casos de suplantación, nuestro programa ofrece un sistema de registro inicial. Para poder iniciar sesión será necesario tener una cuenta de usuario de la Universidad de Jaén. Para la autenticación se utilizarán los servicios web de ILIAS, ya que estos proporcionan la seguridad necesaria.
    - Por otro lado, para poder subir preguntas a un banco de preguntas será necesario tener privilegios de administración en dicho banco. Por lo que, si un usuario no tiene privilegios de administración a un banco de preguntas, no podrá subir preguntas a dicho banco de preguntas.

Desarrollo de manual de usuario y manual de mantenimiento.

Se ha diseñado un manual de usuario de la aplicación que permite informar a los diferentes usuarios (profesores o alumnos) sobre cómo se debe utilizar, errores que se pueden producir…

Por otro lado, se ha diseñado un manual de mantenimiento en formato HTML generado por JavaDOC. En este manual se informa sobre cómo se ha diseñado todo el código de la aplicación, más concretamente, cómo se ha diseñado las clases que permiten ejecutar los servicios web o los métodos HTTP.

1.6.1.2 Objetivos opcionales.

Creación de un banco de preguntas.

Aunque la creación de un banco de preguntas no era un objetivo de este proyecto, se decidió introducir en el proyecto para facilitar la labor de los usuarios a la hora de subir preguntas tipo test.

- Para poder subir preguntas a un banco de preguntas, era necesario que dicho banco de preguntas estuviese ya creado en ILIAS. Para evitar que un usuario tenga que iniciar sesión en Docencia Virtual para crear el banco de preguntas, se ha incorporado a esta aplicación un método que permite la creación de bancos de preguntas.
    - Para poder crear un banco de preguntas será necesario especificar la carpeta donde será creado. Suponga el siguiente ejemplo, un profesor que imparte la asignatura “Protocolos de Transporte” tiene en Docencia Virtual un espacio reservado para dicha asignatura. Dentro de este espacio habrá una serie de carpetas, las cuales podría ser “Teoría”, “Prácticas”, “Carpeta para Banco de

preguntas”. Pues si el profesor quisiera crear el banco de preguntas en esta asignatura, deberá elegir una de las carpetas anteriormente escritas.

- Para la creación de banco de preguntas se utilizará los servicios web de ILIAS.

- **Seguridad**
  - Al igual que pasaba con la creación de preguntas tipo test, será necesario iniciar sesión previamente. Para que el inicio de sesión sea satisfactorio será necesario tener una cuenta de usuario de la Universidad de Jaén.
  - Por otro lado, para poder crear un banco de preguntas será necesario tener privilegios de administración en la carpeta donde se creará dicho banco. Por lo que, si un usuario no tiene privilegios de administración en la carpeta elegida, no podrá crear el banco de preguntas.

Para concluir, se puede afirmar que la aplicación desarrollada cumple con los objetivos pedidos en el proyecto. Además, permite a los usuarios ahorrar tiempo a la hora de crear preguntas tipo test en la plataforma. Puesto que para la creación de una pregunta tipo test utilizando la plataforma es necesario rellenar el siguiente formulario.

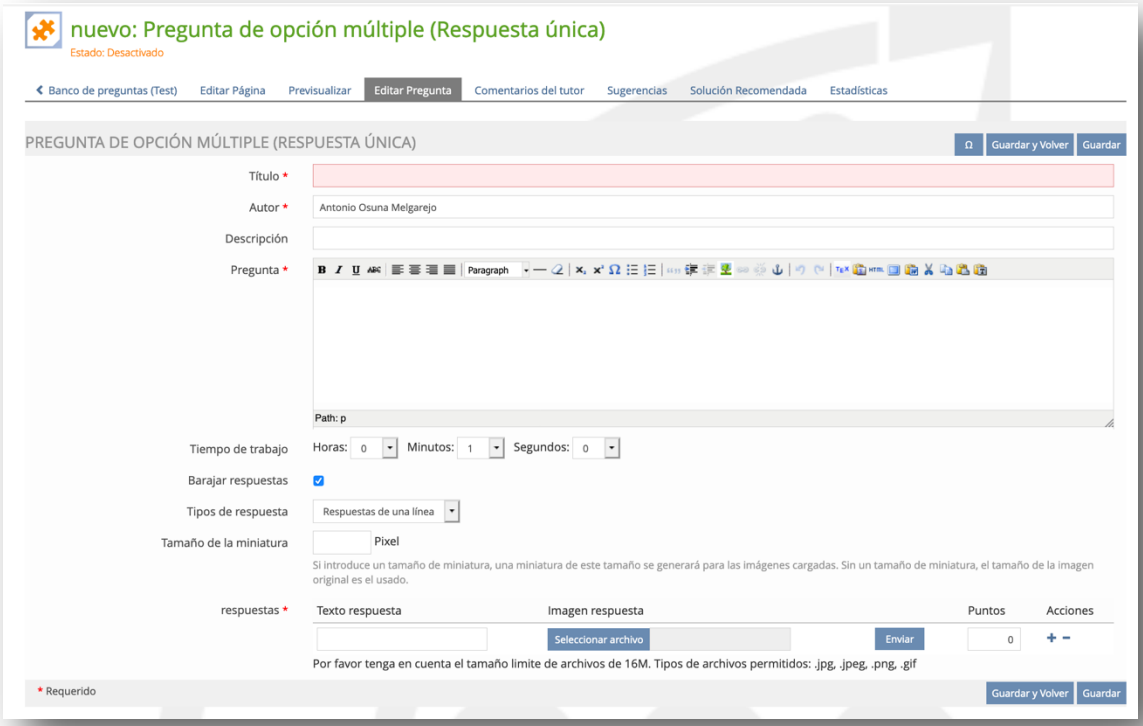


Ilustración 25. Formulario pregunta de opción múltiple en ILIAS.

Tener que rellenar uno o varios formularios para crear preguntas en ILIAS y así poder elaborar un examen tipo test conlleva un tiempo de varios minutos o incluso de algunas horas. Por esta razón, se ha diseñado la aplicación TestUJA, la cual permite crear varias preguntas sin tener que ir rellenando formularios y en un tiempo mucho menor.

1.6.2 Líneas de futuro.

Este programa utiliza principalmente los servicios web y el protocolo HTTP para su funcionamiento. ILIAS ofrece mediante el WSDL gran cantidad de servicios web, lo que permite al usuario mejorar e incluir nuevas funcionalidades a este programa. También, gracias al protocolo HTTP, se pueden incluir nuevas funcionalidades utilizando sus métodos.

Algunas de las mejoras que se podrían incluir en esta aplicación son:

- Permitir la creación en un banco de preguntas de todos los tipos de pregunta test soportados por la plataforma y que no han sido definidos en este proyecto. Esto sería, permitir crear preguntas del tipo “unir parejas”, “preguntas de ordenación”, “preguntas de fórmula”, etc. Para que el programa permita la creación de cualquier tipo de pregunta será necesario modificar tanto el formato de preguntas definido como la clase que se encarga obtener los parámetros de cada pregunta. También será necesario realizar un estudio de cómo están formados los mensajes HTTP intercambiados entre cliente y servidor cuando se crean las preguntas. Una vez comprendido cómo están formados los mensajes, solamente se tendrá que crear métodos que ejecuten los POST necesarios.
- Permitir la creación de exámenes test a través de las preguntas test almacenadas en los bancos de preguntas. Para poder llevar a cabo esta tarea se deberá realizar un estudio de qué mensajes se intercambian entre el cliente y el servidor. También será necesario poder obtener todas las preguntas test de un banco de preguntas y elegir cuáles de ellas se utilizarán para crear el test.
- Permitir que la aplicación te especifique cuál fue el error producido a la hora de crear preguntas test en un banco.

1.7 Bibliografía

1. **ILIAS Society, open source e-Learning.** ILIAS The Open Source Learning Management System. [En línea] 2020. https://www.ilias.de.

2. **Fielding, R., and J. Reschke.** Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230. [En línea] Junio de 2014. https://www.rfc-editor.org/info/rfc7230.

3. **Corporation, Respondus.** Respondus . *Sitio web de Respondus .* [En línea] 2000. https://web.respondus.com/.

4. **T.Anderson.** *IMS Question & Test Interoperability.* [En línea] 11 de Febrero de 2002. https://www.imsglobal.org/question/qtiv1p2/imsqti_asi_bestv1p2.html#1402696.

5. **Goddard, T.** Using NETCONF over the Simple Object Access Protocol (SOAP). *RFC 4743.* [En línea] December de 2006. https://tools.ietf.org/html/rfc4743.

6. **Eastlake 3rd, D., Reagle, J., and D. Solo.** (Extensible Markup Language) XML-Signature Syntax and Processing. *RFC 3275.* [En línea] Marzo de 2002. https://www.rfc-editor.org/info/rfc327.

7. **WSDL, ILIAS.** WSDL Universidad de Jaén. *WSDL de la Universidad de Jaén.* [En línea] 2020. https://dv.ujaen.es/docencia/webservice/soap/server.php.

8. **Github, ILIAS.** Github. *Github.* [En línea] 9 de Septiembre de 2019. https://github.com/ILIAS-eLearning/ILIAS/tree/release_5-4/xml.

9. **Masinter, L.** Returning Values from Forms: multipart/form-data. *RFC 7578.* [En línea] Julio de 2015. https://tools.ietf.org/html/rfc7578.

10. **Berners-Lee, T., Fielding, R., y L. Masinter.** Uniform Resource Identifier (URI): Generic Syntax. *RFC 3986.* [En línea] Enero de 2005. https://www.rfc-editor.org/info/rfc3986.

11. **Oracle Corporation, Sun Microsystems.** JAVA. [En línea] 1996. https://www.java.com/es/.

12. **Eclipse, Fundación.** Eclipse Website. [En línea] 7 de Noviembre de 2001. https://www.eclipse.org.

13. **Foundation, The Apache Software.** [En línea] 2019. http://hc.apache.org/downloads.cgi.

14. **Foundation, Eclipse.** [En línea] 2014. https://www.eclipse.org/windowbuilder/.

2 Anexos

Manual de usuario

Introducción

TestUJA es una aplicación que permitirá subir diferentes tipos de preguntas test (de única respuesta, respuesta corta, múltiple respuesta, encontrar errores y rellenar huecos) en un banco de preguntas. Esta aplicación facilitará la tediosa tarea de crear preguntas test paso por paso en la plataforma.

Inicio

El menú inicial del programa TestUJA es el siguiente:

Ilustración 26. Menú inicial

Solo dará la opción de iniciar el programa.

Login

Una vez se inicie el programa, el usuario deberá registrarse. TestUJA proporcionará al usuario la siguiente interfaz donde deberá introducir el usuario y la contraseña:

Ilustración 27. Login

Una vez que el usuario pulse el botón “LOGIN”, procederá a comprobar sus credenciales.

Si el registro se ha completado (con éxito o no), se mostrará una ventana emergente informando sobre la situación.

Login correcto

Si el inicio de sesión es correcto se muestra en la siguiente imagen:

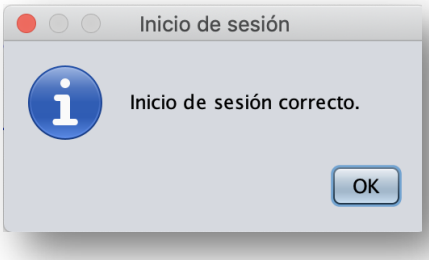


Ilustración 28. Información. Login correcto

Login incorrecto

Si el inicio de sesión es incorrecto se muestra el siguiente mensaje y el usuario tendrá que volver a introducir las credenciales correctamente.

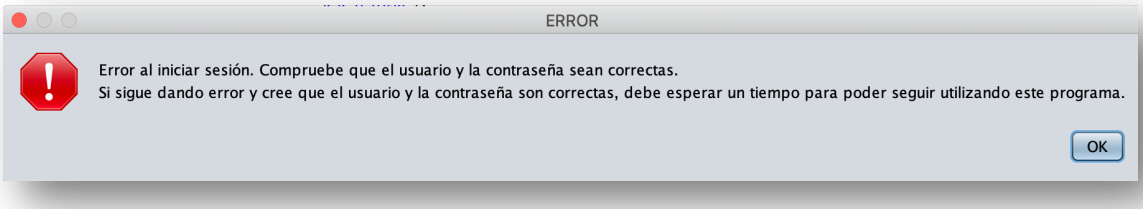


Ilustración 29. Error. Login incorrecto

Nota: Algunas veces da error aunque las credenciales son correctas. Si ocurre esto, el usuario deberá esperar un tiempo para poder utilizar el programa.

Menú principal

Una vez se haya completado el registro correctamente, el usuario se desplazará al menú principal de la aplicación. Aquí, dará tres opciones a elegir que se muestran en la siguiente imagen:

Ilustración 30. Menú principal

Buscar bancos de preguntas

Si el usuario elige la opción “Buscar un banco de preguntas” se mostrará la siguiente interfaz, donde da la opción de escribir el nombre del banco de preguntas que se quiere buscar:

Ilustración 31. Búsqueda del banco de preguntas

Si la búsqueda del banco de preguntas se ha completado (con éxito o no), se mostrará una ventana emergente informando sobre la situación.

Búsqueda del banco de preguntas incorrecto

Si al realizar la búsqueda del banco de preguntas, no se encuentra uno, se mostrará el siguiente error:

Ilustración 32. Error. Banco de preguntas no encontrado

Búsqueda del banco de preguntas correcto

Encuentro de un banco de preguntas.

Si al realizar la búsqueda del banco de preguntas, encuentra solo un banco de preguntas, se mostrará el siguiente mensaje:

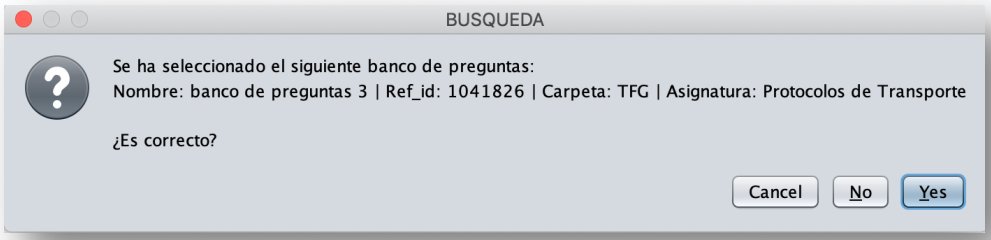


Ilustración 33. Información de un banco de preguntas

Ahora el usuario deberá decidir si es el banco de preguntas que está buscando o no.

Encuentro de varios bancos de preguntas.
Si al realizar la búsqueda del banco de preguntas, encuentra varios bancos de preguntas con nombres similares, mostrará el siguiente mensaje:

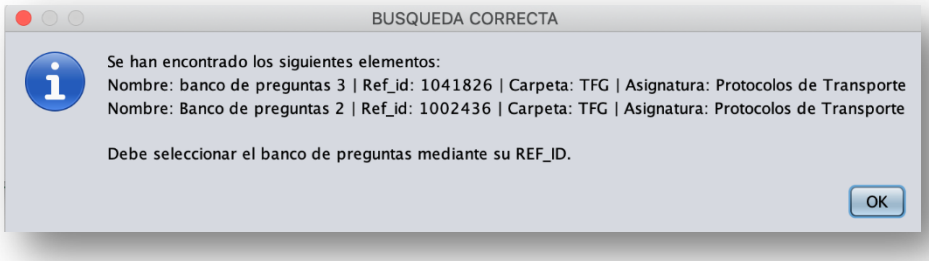


Ilustración 34. Información de varios bancos de preguntas.

A continuación, el usuario deberá elegir el banco de preguntas que está buscando a través de su Ref_id. El programa mostrará la siguiente interfaz:

Ilustración 35. Búsqueda banco de preguntas según ref_id

Si no recuerda el Ref_id del banco de preguntas que está buscando, tendrá que pulsar el botón "Elegir Banco" para que muestre las opciones que se pueden elegir. Por ejemplo, para el caso de que la aplicación encuentre el banco de preguntas 3 y Banco de preguntas 2, si pulsa el botón "Elegir Banco" mostrará las siguientes opciones seleccionables:

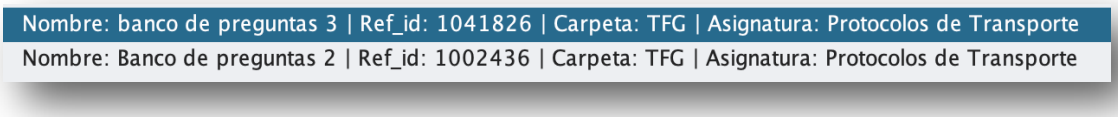


Ilustración 36. Información botón ref_ids buscar banco de preguntas

Una vez el usuario selecciona una de estas opciones, automáticamente se escribirá el ref_id y solo tendrá que elegir el banco de preguntas. Cuando pulse el botón “Buscar banco de preguntas”, la aplicación informará al usuario sobre si el banco de preguntas elegido es correcto o no.

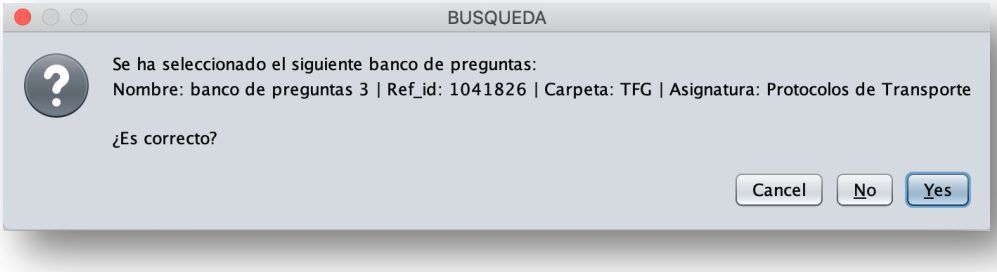


Ilustración 37. Información búsqueda banco de preguntas según ref_ids

Si el usuario confirma que es el banco de preguntas correcto, la aplicación mostrará el siguiente mensaje:

Ilustración 38. Información banco de preguntas encontrado

Crear bancos de preguntas

Si el usuario elige la opción de crear banco de preguntas, deberá introducir el nombre de la carpeta donde se creará el banco (la carpeta debe estar creada anteriormente si no, el usuario no puede crear un banco), el nombre que tendrá el banco de preguntas y una descripción (opcional).

Ilustración 39. Creación de banco de preguntas

Crear un banco de preguntas de manera incorrecta

Si el usuario intenta crear un banco de preguntas sin introducir datos, la aplicación mostrará la siguiente ventana:

Ilustración 40. Error. Banco de preguntas y carpeta sin nombre

Si introduce una carpeta (inexistente) y un banco de preguntas, mostrará el siguiente mensaje:

Ilustración 41. Error. Carpeta no encontrada.

Crear un banco de preguntas correcto

Encuentro de una carpeta para crear el banco de preguntas

Si al realizar la búsqueda de la carpeta, encuentra solo una, la aplicación mostrará el siguiente mensaje:

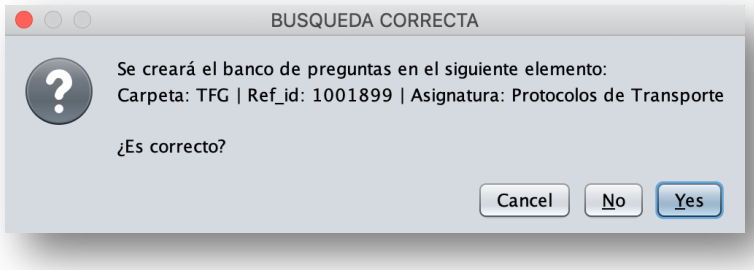


Ilustración 42. Búsqueda encontrada de una carpeta

Ahora, el usuario deberá decidir si es la carpeta que está buscando o no.

Encuentro de varias carpetas para crear un banco de preguntas

Si el usuario intenta crear un banco de preguntas y se encuentran varias carpetas, por ejemplo, una llamada “TFG” y otra llamada “TFG prueba”, la aplicación mostrará un mensaje indicando las carpetas que ha encontrado:

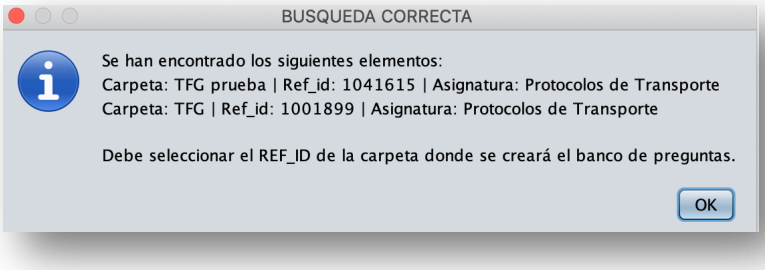


Ilustración 43. Búsqueda encontrada de varias carpetas

Al encontrar varias carpetas, el usuario tendrá que seleccionar la carpeta a través de su ref_id en la siguiente interfaz.

Ilustración 44. Creación del banco de preguntas según ref_ids

Si no recuerda el Ref_id de la carpeta que está buscando, tendrá que pulsar el botón "Elegir Carpeta" para que muestre las opciones que se pueden elegir. Por ejemplo, para el caso de encontrar la carpeta "TFG" y "TFG prueba", si el usuario pulsa el botón "Elegir Carpeta" la aplicación mostrará las siguientes opciones seleccionables:

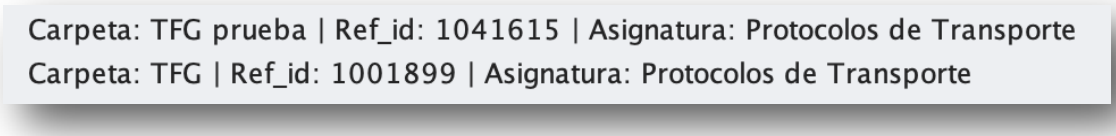


Ilustración 45. Botón "Elegir Carpeta" crear banco de preguntas

Una vez selecciona una de estas opciones, automáticamente se escribirá el ref_id y solo tendrá que volver a introducir el nombre que quiere que tenga el banco de preguntas y su descripción (opcional). Cuando pulse el botón "Crear banco de preguntas", el programa le informará sobre la carpeta elegida para poder conocer si es la correcta o no.

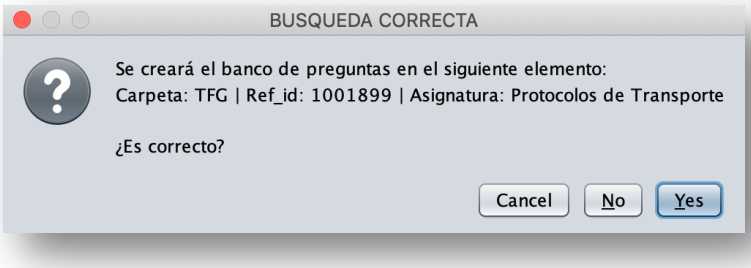


Ilustración 46. Búsqueda correcta carpeta

Si selecciona que la carpeta es correcta, la aplicación comenzará a crear el banco de preguntas. Una vez creado, la aplicación mostrará un mensaje similar al siguiente:

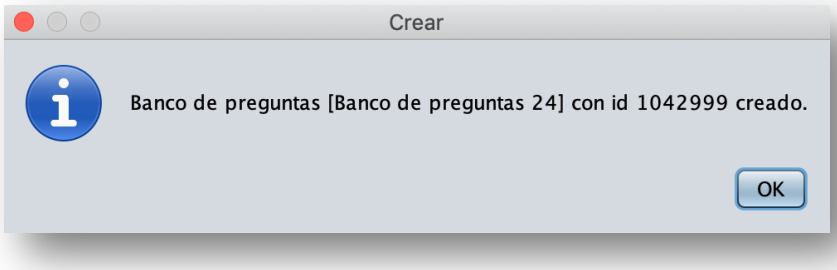


Ilustración 47. Información banco de preguntas creado

Validación de documentos de texto.

Si el usuario elige la opción de validar un documento de texto, el usuario tendrá que introducir el directorio donde está el archivo con las preguntas tipo test en el formato adecuado.

Ilustración 48. Validación de documentos de texto.

Para facilitar la búsqueda de la ruta del archivo se proporciona un botón "…". Una vez que el usuario haya elegido el directorio solo tendrá que pulsar el botón "Comprobar documento" para comenzar el proceso de validación.

Una vez finalizado el proceso de validación, la aplicación mostrará una ventana informando de las posibles preguntas erróneas y del autor de ellas.

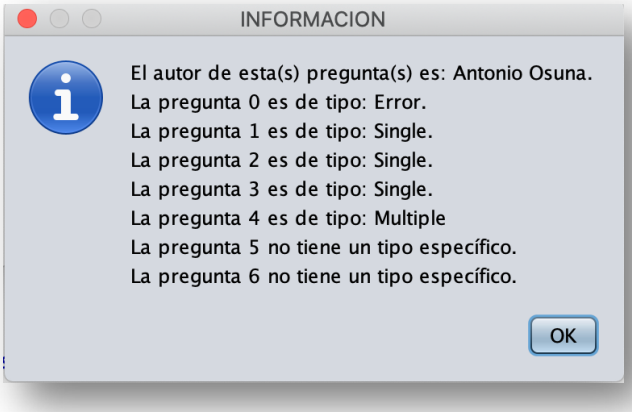


Ilustración 49. Información Validación de documentos

Subir preguntas

Una vez haya finalizado con éxito la creación o búsqueda del banco de preguntas, la aplicación mostrará la interfaz encargada de la subida de las preguntas tipo test. En ella, el usuario tendrá que introducir el directorio donde está el archivo con las preguntas tipo test en el formato específico.

Ilustración 50. Subir preguntas

Para facilitar la búsqueda de la ruta del archivo se proporciona un botón "…". Una vez que el usuario haya elegido el directorio solo tendrá que pulsar el botón "Subir preguntas" para comenzar el proceso de subida.

A continuación, se muestra un ejemplo subiendo las preguntas que se encuentran en un archivo.

Ilustración 51. Ejemplo subir preguntas

Una vez finalizado el proceso de subida, la aplicación mostrará una ventana informando de las posibles preguntas erróneas y del autor de ellas.

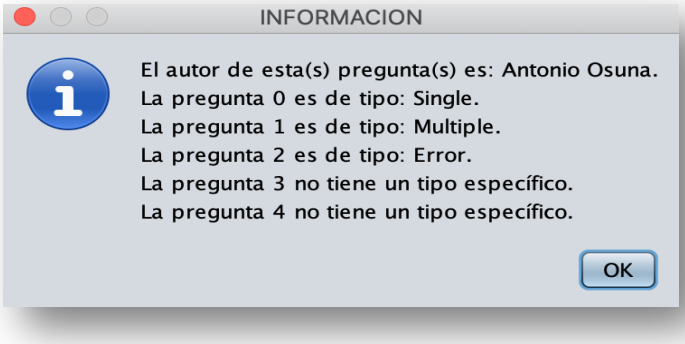


Ilustración 52. Información preguntas tipo test subidas

Ayuda HTML

Además del manual de usuario, este programa facilita una ayuda mediante documentos HTML. Para acceder a esta ayuda, cada interfaz de la aplicación dispone de una imagen que permite el acceso a ella. Dependiendo de la interfaz en la que se encuentre situado el usuario se mostrará un documento de ayuda u otro.

A continuación, se muestra un ejemplo de cómo acceder a esta ayuda:

Ilustración 53. Acceder a ayuda HTML.

Al hacer clic en el icono de ayuda, la aplicación mostrará un documento HTML donde, principalmente, explica el funcionamiento de esa interfaz. Una parte de este documento de ayuda se muestra en la siguiente imagen:

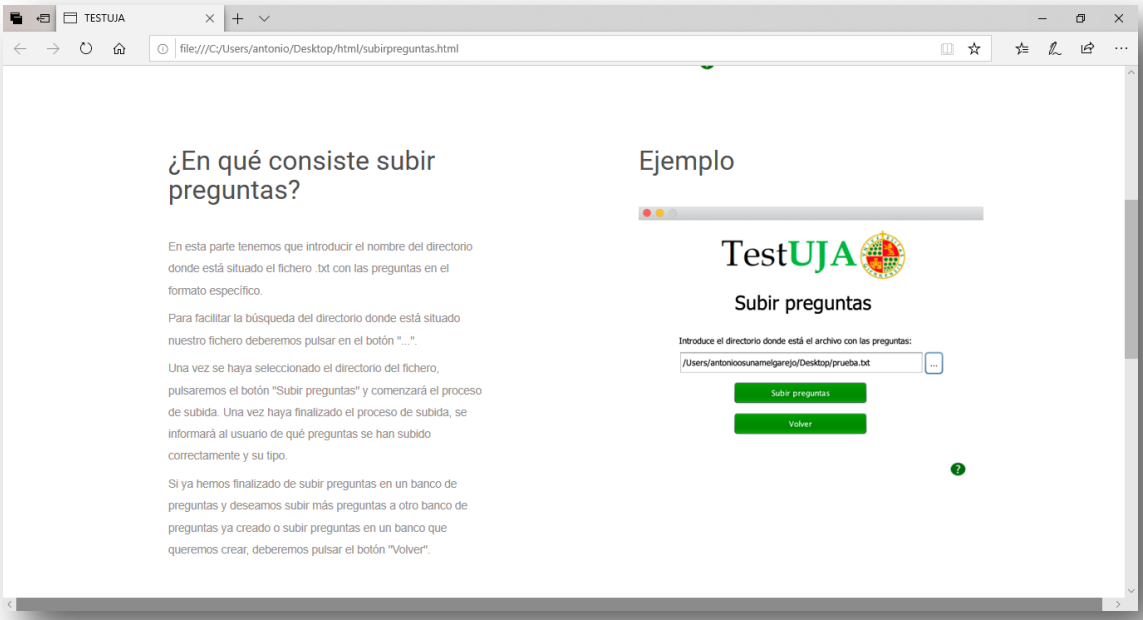


Ilustración 54. Ejemplo ayuda HTML.

Manual de mantenimiento

Para facilitar las labores de mantenimiento de esta aplicación de escritorio se facilita una documentación en formato HTML generada por JavaDOC. En esta documentación se informa sobre todas las clases utilizadas para los servicios web, métodos HTTP, lectura de ficheros (.txt) y todas las funciones utilizadas en las clases juntos con los parámetros de entrada que necesitan y valores que devuelven.

La documentación completa se encuentra situada en la carpeta de este proyecto.

3 Planos

Plano inicio de sesión

Plano menú principal.

Plano buscar banco de preguntas.

Plano buscar banco de preguntas según Ref_ID.

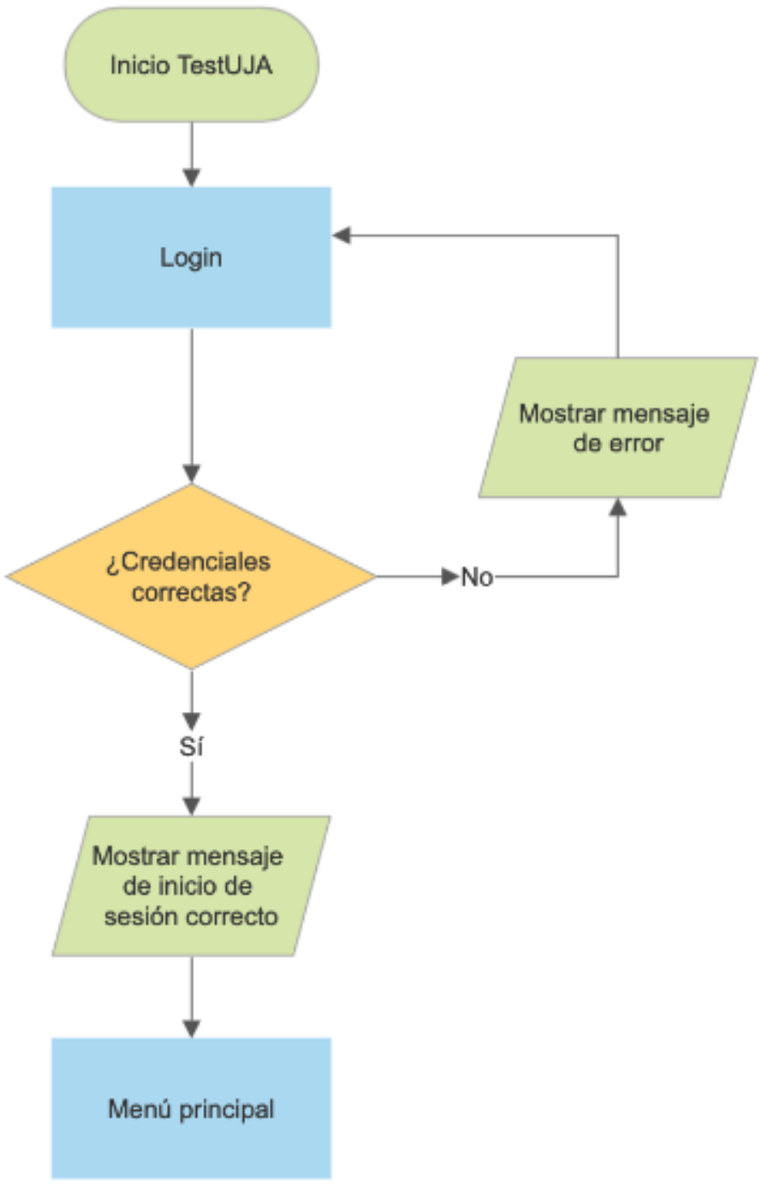
Plano crear banco de preguntas.

Plano crear banco de preguntas según Ref_ID.

Plano comprobar documento.

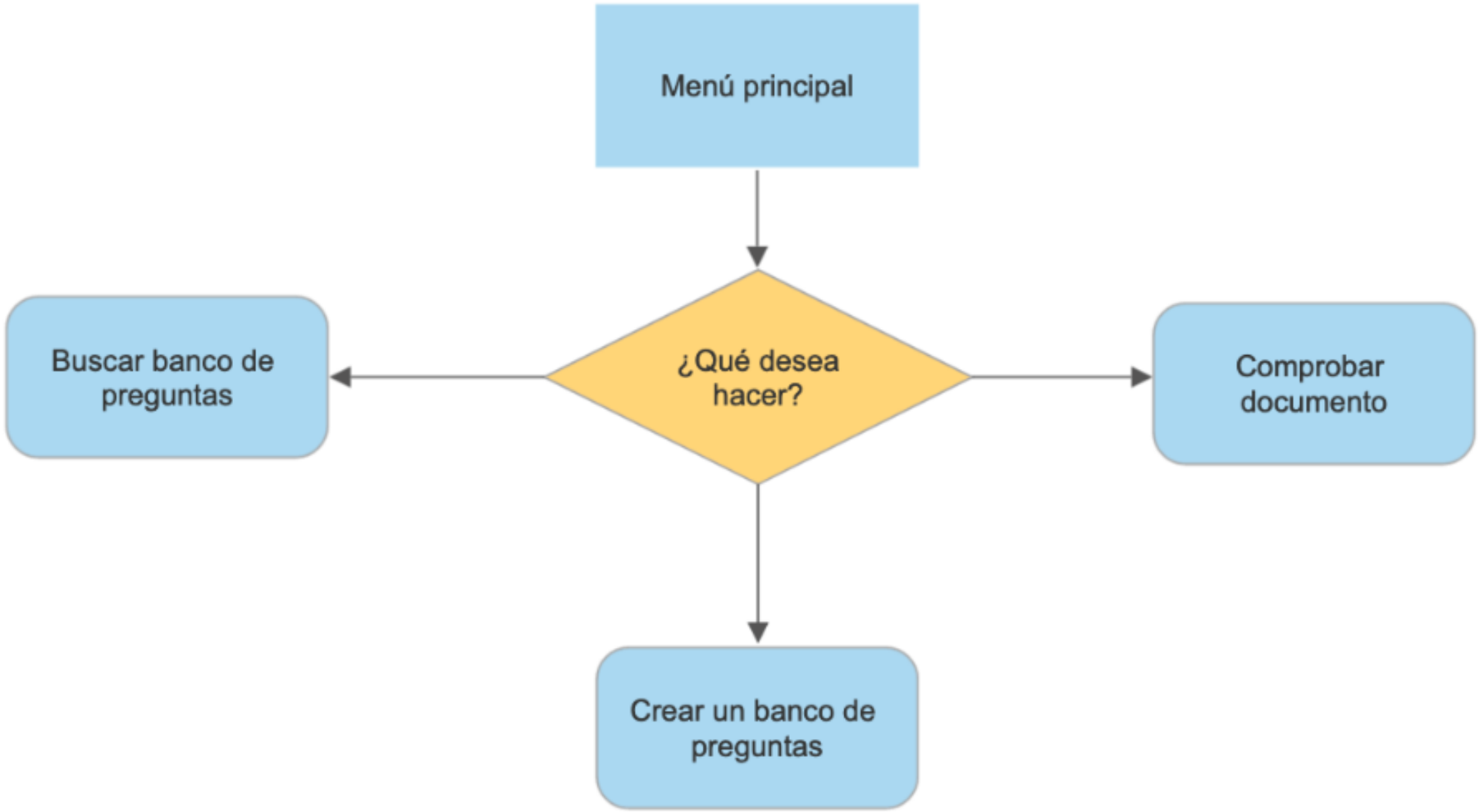
Plano subir preguntas

3.1 Plano inicio de sesión



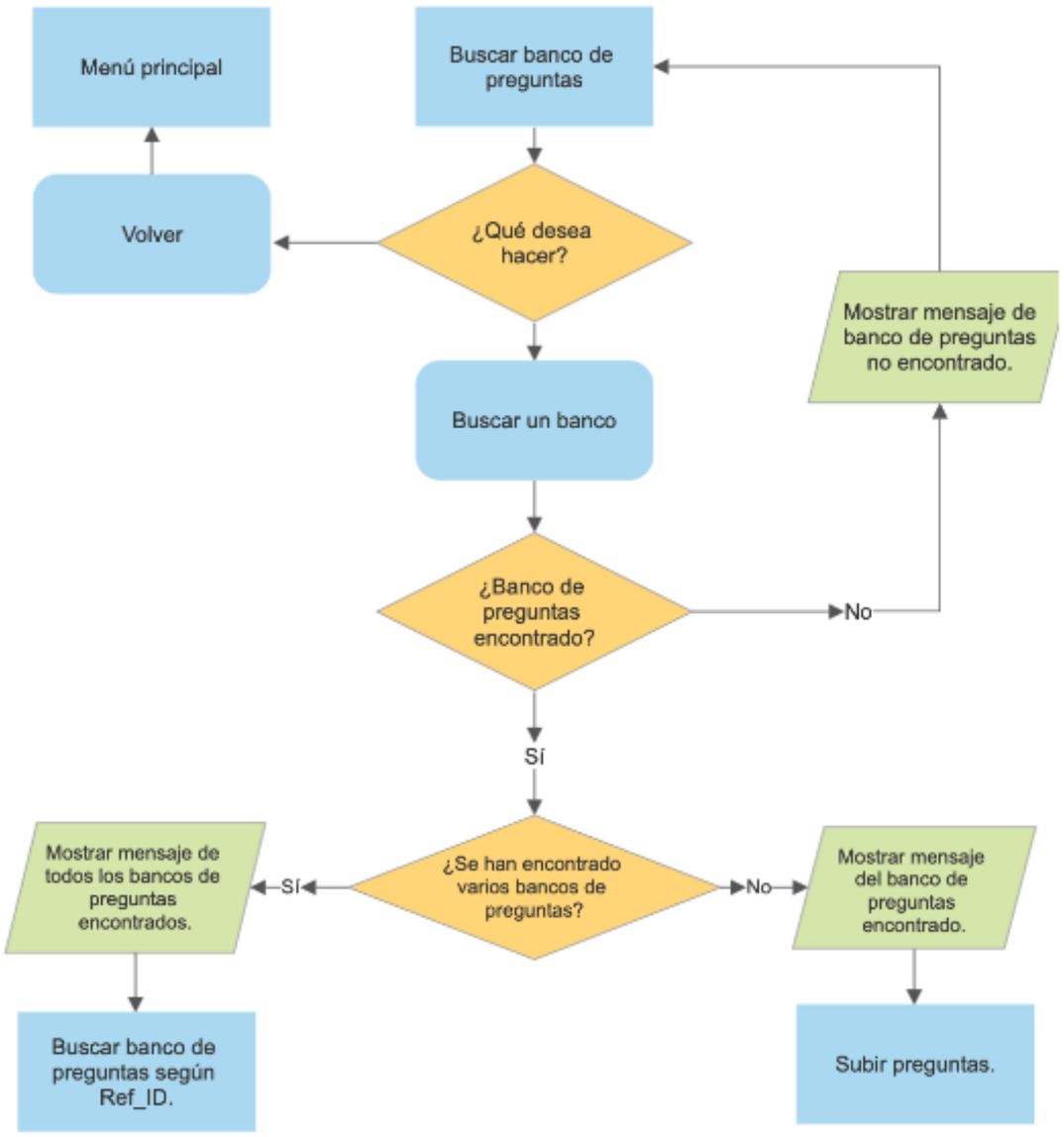
AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **1/8**	Título del TFG Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano** INICIO DE SESIÓN

3.2 Plano menú principal.



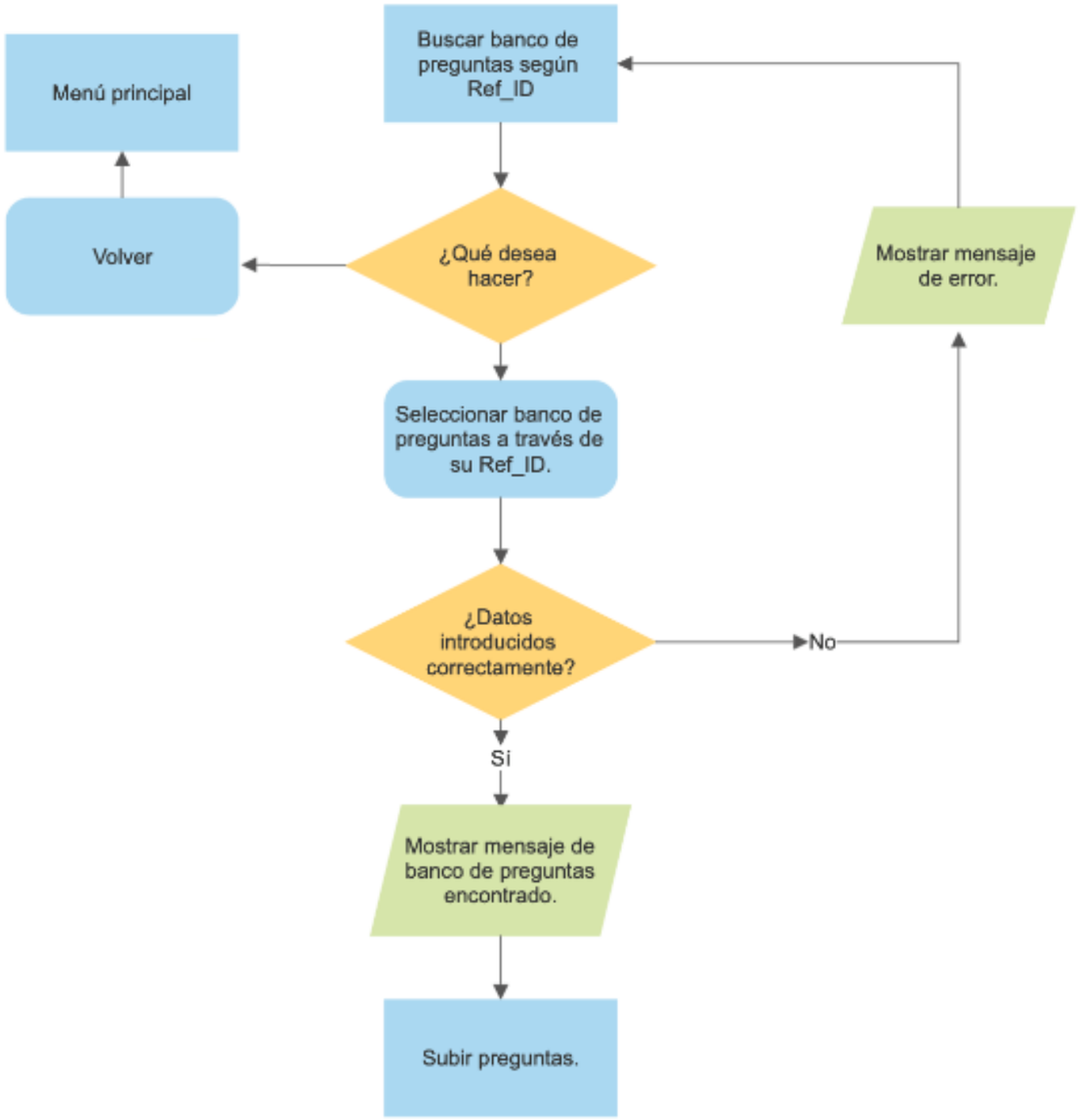
AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **2/8**	**Título del TFG** Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano** MENÚ PRINCIPAL

3.3 Plano buscar banco de preguntas.



AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **3/8**	Título del TFG: Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano**: BUSCAR BANCO DE PREGUNTAS

3.4 Plano buscar banco de preguntas según Ref_ID

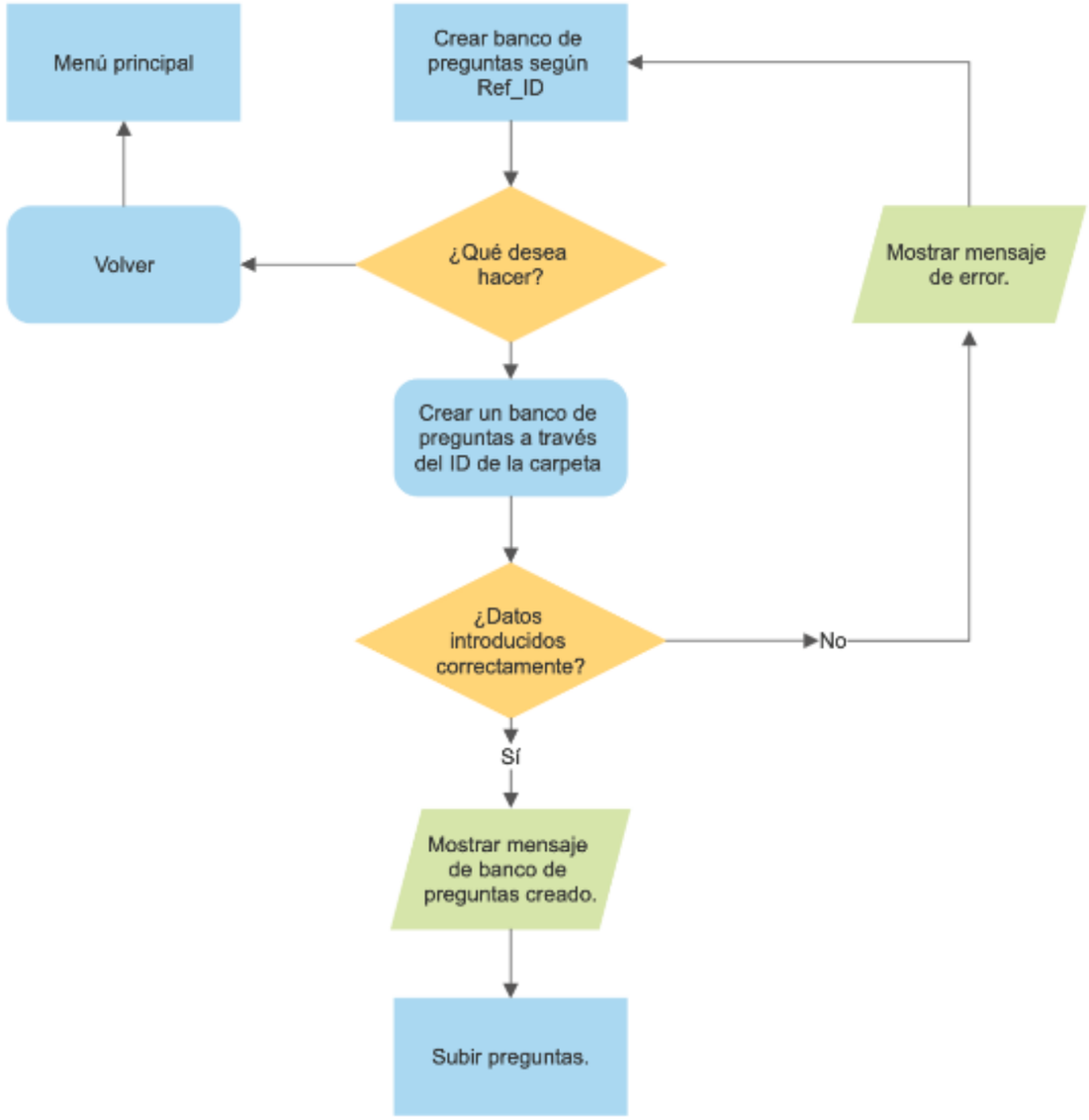


AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **4/8**	Título del TFG: Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano**: BUSCAR BANCO DE PREGUNTAS SEGÚN REF_ID

3.5 Plano crear banco de preguntas.

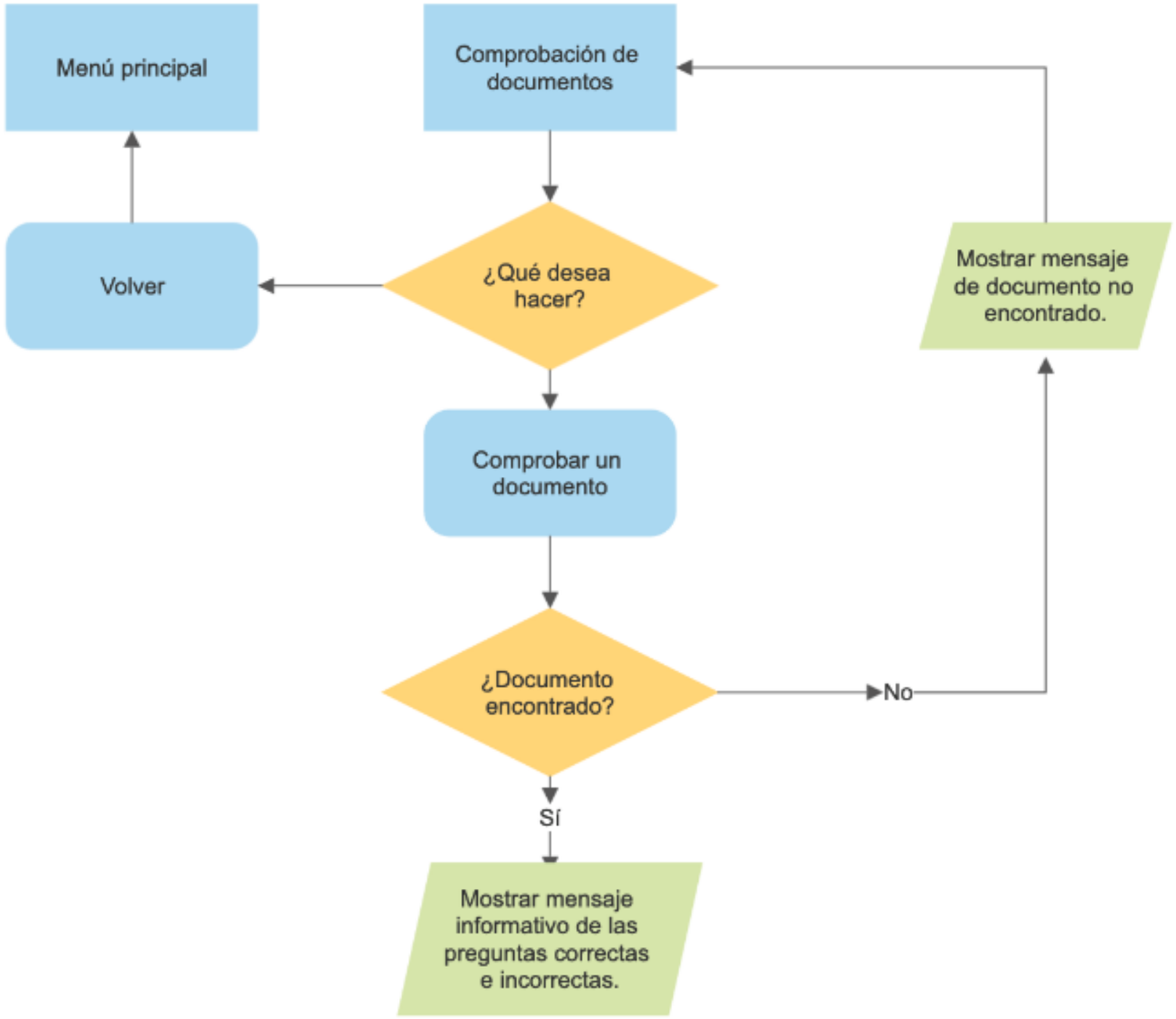
AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: 5/8	Título del TFG: Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. Título del plano: CREAR BANCO DE PREGUNTAS

3.6 Plano crear banco de preguntas según Ref_ID.



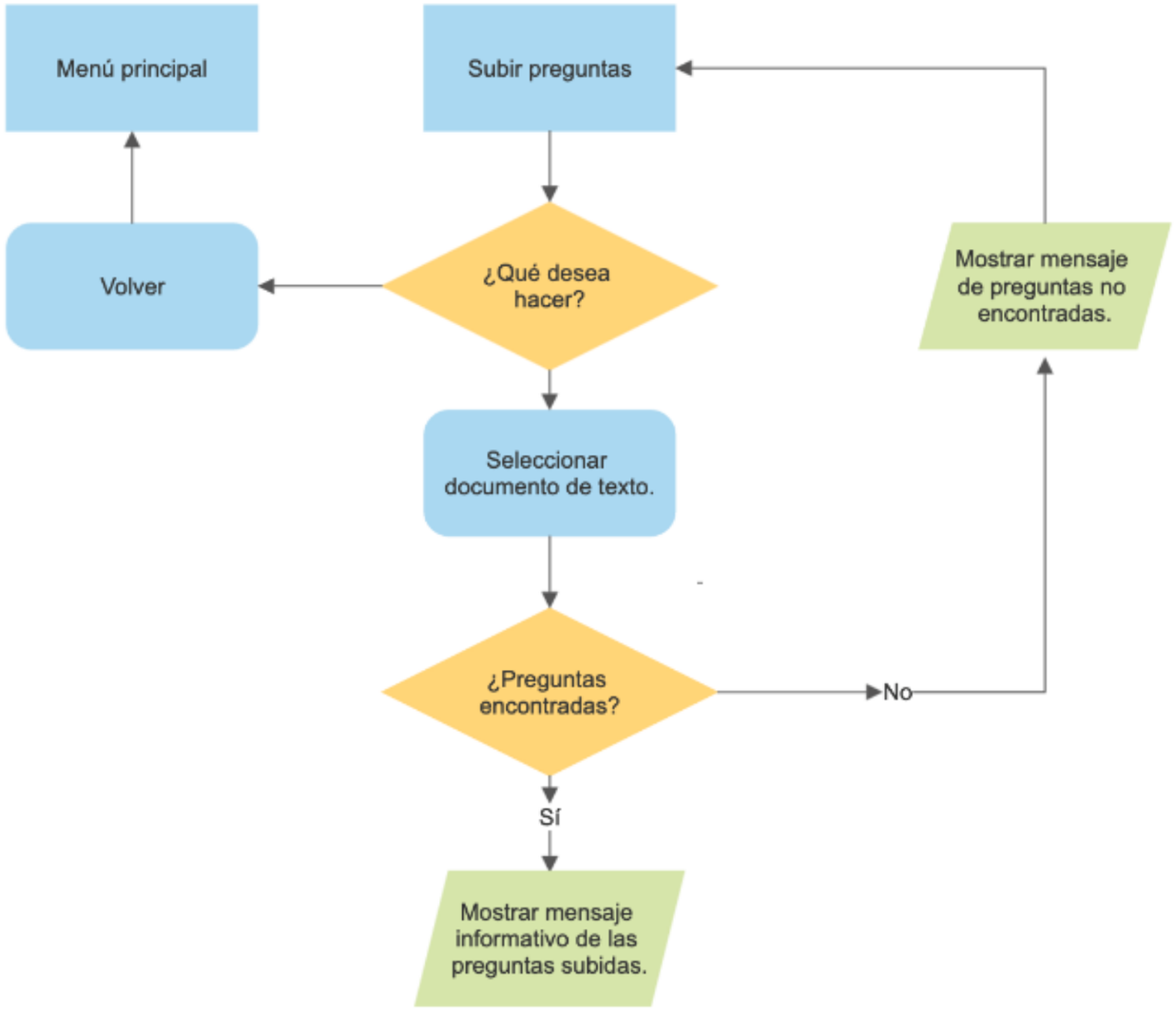
AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: 6/8	Título del TFG: Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. Título del plano: CREAR BANCO DE PREGUNTAS SEGÚN REF_ID

3.7 Plano comprobar documento.



AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **7/8**	Título del TFG Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano** COMPROBAR DOCUMENTOS

3.8 Plano subir preguntas.



AUTOR: ANTONIO OSUNA MELGAREJO	ESCUELA POLITÉCNICA SUPERIOR DE LINARES
Nº PLANO: **8/8**	Título del TFG: Aplicación para el procesado y subida automáticos de test de evaluación a ILIAS. **Título del plano**: Subir preguntas

4 Pliego de condiciones

4.1 Elementos software y licencias.

Los elementos software necesarios para la elaboración de este programa son:

1. Eclipse: Para el desarrollo software del programa se utilizará el IDE Eclipse, el cual está publicado bajo la licencia de software libre denominada *Eclipse Public License*, la cual permite libertad de utilización, distribución, copia y modificación del software.
2. Lenguaje de programación Java: El diseño del programa se realizará con lenguaje Java, el cual está publicado bajo la licencia denominada GNU GPL. GNU GPL es una licencia de derecho de autor ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.
3. Librería Ksoap2: Permite el funcionamiento de los servicios web en ILIAS mediante el protocolo de comunicación SOAP. Esta librería está publicada bajo la licencia MIT (*Massachusetts Institute of Technology*), la cual incluye al usuario final el derecho a usar, copiar, modificar, fusionar, publicar, distribuir, cambiar la licencia y/o vender el software.
4. Librería Httpclient: Permite el funcionamiento del protocolo HTTP y sus métodos. Está librería está publicada bajo la licencia *Apache License 2.0*, la cual permite al usuario del software la libertad de usar el software para cualquier propósito, distribuirlo y modificarlo.
5. ILIAS: Es el sistema de gestión para la enseñanza, LMS, desarrollado en código abierto y utilizado por la Universidad de Jaén. ILIAS está disponible como software libre de código abierto bajo la licencia GNU GPL, la cual garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir y modificar el software.

4.2 Requisitos del usuario

Los requisitos necesarios para poder utilizar el programa TestUJA son los siguientes:

- Un ordenador (con cualquier sistema operativo).
- Acceso a Internet.
- El usuario debe tener una cuenta activa en la Universidad de Jaén.
- El usuario que utilice el programa debe tener permisos de administración en las asignaturas, para poder crear bancos de preguntas o buscar bancos de preguntas.
- Para poder subir preguntas, es necesario especificar el banco de preguntas donde se subirán. Es decir, si no hay bancos de preguntas en la asignatura, no será posible crear preguntas tipo test.
- Al igual que ocurre con la búsqueda/creación de bancos de preguntas, para poder subir preguntas tipo test a un banco de preguntas, será necesario también, por parte del usuario, tener permisos de administración en la asignatura.

5 Mediciones

A continuación, se muestran todas las tareas realizadas para desarrollar este proyecto.

5.1 Estudios preliminares de tecnologías y especificación de requisitos.

Descripción	Horas
Estudio de las tecnologías empleadas por ILIAS para el desarrollo del proyecto.	15
Estudio de los formatos necesarios para el desarrollo de los elementos del proyecto.	10
Especificación de los requisitos necesarios.	5
TOTAL	**30**

Tabla 7. Estudios preliminares de tecnologías y especificación de requisitos.

5.2 Diseño preliminar

Descripción	Horas
Diseño previo de la interfaz gráfica del proyecto.	5
TOTAL	**5**

Tabla 8. Diseño preliminar

5.3 Implementación del servicio

Descripción	Horas
Desarrollo de los servicios utilizando Servicios Web	60
Desarrollo de los servicios utilizando el protocolo HTTP	90
TOTAL	**150**

Tabla 9. Implementación del servicio.

5.4 Diseño definitivo

Descripción	Horas
Diseño definitivo de la interfaz gráfica del proyecto.	35
TOTAL	**35**

Tabla 10. Diseño definitivo.

5.5 Pruebas y corrección de errores

Descripción	Horas
Realización de pruebas de la aplicación de escritorio.	20
Corrección de errores de la aplicación de escritorio.	30
TOTAL	**50**

Tabla 11. Pruebas y corrección de errores.

5.6 Elaboración de la documentación

Descripción	Horas
Elaboración de manual de usuario	10
Elaboración de manual de mantenimiento	10
Elaboración de documentación de ayuda de aplicación de escritorio	10
TOTAL	**30**

Tabla 12. Elaboración de la documentación.

6 Presupuesto

A continuación, se muestra una tabla con el precio total del proyecto, desglosado según los procesos que se han ido realizando:

Unidad (h)	Concepto	Precio/hora	Precio
30	Estudios preliminares de tecnologías y especificación de requisitos.	25 € / hora	750,00 €
5	Diseño preliminar		125,00 €
150	Implementación del servicio		3750,00 €
35	Diseño definitivo		875,00 €
50	Pruebas y corrección de errores.		1250,00 €
30	Elaboración de la documentación		750,00 €
Precio total sin IVA			7500,00 €
Precio total con IVA (21%)			9075,00 €

Tabla 13. Resumen presupuesto.

El presente presupuesto para el proyecto asciende a la expresada cantidad de NUEVE MIL SETENTA Y CINCO EUROS (9075,00 €).

Enero de 2019