Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

ESTUDIO Y DESARROLLO DE UN PROTOTIPO DE APLICACIÓN PARA LA GESTIÓN DE CONTENIDOS EN LA PLATAFORMA DE DOCENCIA VIRTUAL DE LA UNIVERSIDAD DE JAÉN

Alumno: David Peinado Carrillo

Tutor: Prof. D. José Ramón Balsas Almagro
Dpto: Informática

Septiembre, 2014

Índice

1. INTRODUCCIÓN

En este apartado se va a describir el propósito del proyecto, los objetivos que debe cumplir el proyecto, la metodología a seguir y cuál va a ser la estructura del documento para saber de lo que se va a hablar en este documento.

1.1. Propósito

Descripción general del proyecto: La Universidad de Jaén dispone de una plataforma de docencia virtual o LMS (*Learning Management System*) [1] como apoyo a la docencia presencial. Ésta plataforma dispone de diferentes funcionalidades que el usuario puede utilizar a través de los servicios SOAP [2] (*Simple Object Access Protocol*). Sirve para que un usuario pueda realizar cualquier funcionalidad que necesite dentro de esta plataforma. Este proyecto tiene como meta estudiar las funcionalidades que proporciona esta capa de servicios SOAP [2] para llegar a crear y desarrollar un prototipo multiplataforma de la aplicación utilizando para ello la tecnología JavaFX [3] y así poder gestionar varios de los servicios que proporciona esta capa de forma remota.

Justificación: El prototipo propuesto evitará tener que acceder a ILIAS (***I****ntegriertes* ***L****ern-,* ***I****nformations- und* ***A****rbeitskooperations-* ***S****ystem*) [1] [4] a través de la web permitiendo el acceso a ILIAS [1] de una forma más sencilla y rápida para realizar cualquier funcionalidad que se desee.

1.2. Objetivos

- Realizar un estudio de la interfaz de servicios WEB del LMS ILIAS [1] para la gestión de los contenidos, la gestión de las instalaciones y la gestión de los usuarios desde una aplicación externa (prototipo) con la que se interactuará de esta manera con estos servicios WEB del LMS ILIAS [1].

- Diseñar una aplicación informática para gestionar de manera externa los contenidos en espacios situados en cualquiera de las instalaciones del LMS ILIAS [1].

- Implementar un prototipo multiplataforma de la aplicación utilizando para ello la tecnología JavaFX [3] y cualquier programa que permita realizar fácilmente una implementación de este prototipo.

1.3. Metodología

- Estudio de los requerimientos del sistema a desarrollar para llegar a obtener todas las funcionalidades deseadas y poner crear un prototipo a la altura de los objetivos.

- Recopilación bibliográfica sobre la temática y tecnologías relacionadas para obtener una información completa de todo lo relacionado con este proyecto y conocer más a fondo las tecnologías utilizadas.

- Estimación temporal que se ha calculado para finalizar el proyecto y estimación de costes del desarrollo para calcular el coste que va a tener este proyecto con respecto a la estimación realizada.

- Diseño del sistema utilizando una metodología de orientación a objetos utilizando para ello el patrón de diseño Modelo-Vista-Controlador.

- Implementación del prototipo del sistema con el que se van a poder utilizar las funcionalidades necesarias para realizar una gestión de los usuarios, una gestión de las instalaciones y una gestión de los contenidos.

1.4. Estructura del documento

El presente documento se ha organizado de la siguiente forma: en el apartado 2 de análisis se estudiará lo que es la plataforma ILIAS [1] y sus herramientas más destacadas; la interfaz pública de servicios web en ILIAS [1], donde se explicará qué es el protocolo SOAP [2], sus características y algunos ejemplos de lo que devuelven algunos servicios SOAP [2]. También se verá como manipular SOAP [2] desde JAVA [5] usando JAXB (*Java Architecture for XML Binding*) [6] y su uso, como se comunican SOAP [2] y JAVA [5]. De qué trata el framework de desarrollo JavaFX [3] y el uso de JavaFX [3] en ILIAS [1]. Por último, se presentará una propuesta de solución junto con una estimación de costes de desarrollo.

En el apartado 3 de diseño se estudiará el diagrama de clases de este proyecto para ver cómo se relacionan unas clases con otras y las clases SOAP [2] que han sido más importantes en este proyecto. Los diagramas de secuencia más relevantes para ver cómo interactúa una parte con otra para realizar una funcionalidad; además del diseño inicial de la interfaz de

usuario donde se mostrará el storyboard realizado inicialmente de la interfaz de usuario.

En el apartado 4 de implementación se estudiará la arquitectura de esta aplicación, que se ha llevado a cabo para llevar el proceso de desarrollo con JavaFX [3] y detalles sobre implementación donde se indican las bibliotecas utilizadas y las clases que han extendido el API (*Application Programming Interface*) [7] de ILIAS [1].

En el apartado 5 de conclusiones se indicarán las conclusiones que he sacado respecto de este proyecto y las mejoras y trabajos futuros que se pueden realizar sobre esta aplicación.

Al final del trabajo se adjuntan una serie de apéndices donde se indicará el manual de usuario en el que describe los pasos para poder realizar todas las funcionalidades que proporciona la aplicación y una descripción de los contenidos suministrados para este proyecto.

2. ANÁLISIS

En este apartado estudiaremos las características de la plataforma ILIAS [1], de su interfaz de servicios web y de las características del entorno de desarrollo JavaFX [3]. Estos elementos nos permitirán plantear una propuesta de solución junto con una estimación de costes para su desarrollo.

2.1. La plataforma ILIAS

Esta plataforma [1] es un programa con el que se pueden administrar y gestionar actividades y recursos que son educativos. Esta plataforma permite tanto crear como gestionar cursos, grupos, etc, de una manera sencilla. Se usa como docencia para los alumnos de la Universidad y para los profesores de la misma.

A continuación se enumera una serie de herramientas que esta plataforma contiene y que son las más destacadas:

- **Escritorio personal**: para cada usuario de ILIAS [1], hay un escritorio personal. En dicho escritorio personal, un usuario puede ver los cursos, grupos, etc, a los que está matriculado y su contenido, ver su correo o cualquier otra cosa que le sea de interés.

- **Gestión de cursos**: en esta plataforma se pueden crear y gestionar cursos desde un entorno que contiene la misma.

- **Gestión de grupos**: desde aquí también se pueden formar y gestionar grupos y pueden variar mucho según las características que se le asignen.

- **Creación de test y cuestionarios**: otra cosa que se puede realizar es tanto test como cuestionarios. Ambos tienen una gran variedad de preguntas y herramientas para realizar estadísticas con las que se puede elaborar un análisis de los resultados que se han obtenido y la evaluación de estos test o cuestionarios.

- **Notificaciones**: por otro lado, un usuario puede comunicarse con otros alumnos o profesores a través de foros, correos, etc.

2.1.1. Interfaz pública de servicios WEB en ILIAS

ILIAS permite el acceso a los usuarios a algunas funcionalidades de la plataforma a través de un API [7] pública de servicios basada en el protocolo SOAP [2]. En este apartado se va a hablar de lo que es la capa de servicios SOAP [2] y de la interfaz SOAP [2] en ILIAS [1].

¿Qué es SOAP?

- ***Simple Object Access Protocol*** [2]: consiste en un protocolo estándar que es orientado a objetos y que facilita que se comuniquen varias aplicaciones entre sí en formato XML, es decir, enviándose entre ellas mensajes en formato XML.

- **Objetivos**: En [8] se describen los objetivos de este protocolo que se describen a continuación:

  - Crear un protocolo estándar que llame a servicios que sean remotos. Este protocolo se basa en los protocolos de Internet como son el HTTP (*Hypertext Transfer Protocol*) [9] para realizar transmisiones de mensajes y el XML para codificar la información.

  - Actuar de manera independiente al lenguaje que se usa para el desarrollo, a la plataforma y a la implementación.

- **Características**: En [2] se definen las características del protocolo SOAP que se muestran a continuación:

  - ***Extensibilidad***: proporciona el uso de extensiones en el desarrollo como la seguridad y el enrutamiento de WS.

  - ***Neutralidad***: se puede utilizar con cualquiera de los protocolos de transporte como pueden ser: SMTP [10], HTTP [9], JMS [11] o TCP [12].

- ***Independencia***: permite el uso de cualquiera de los tipos de programación.

- **Funcionamiento**:

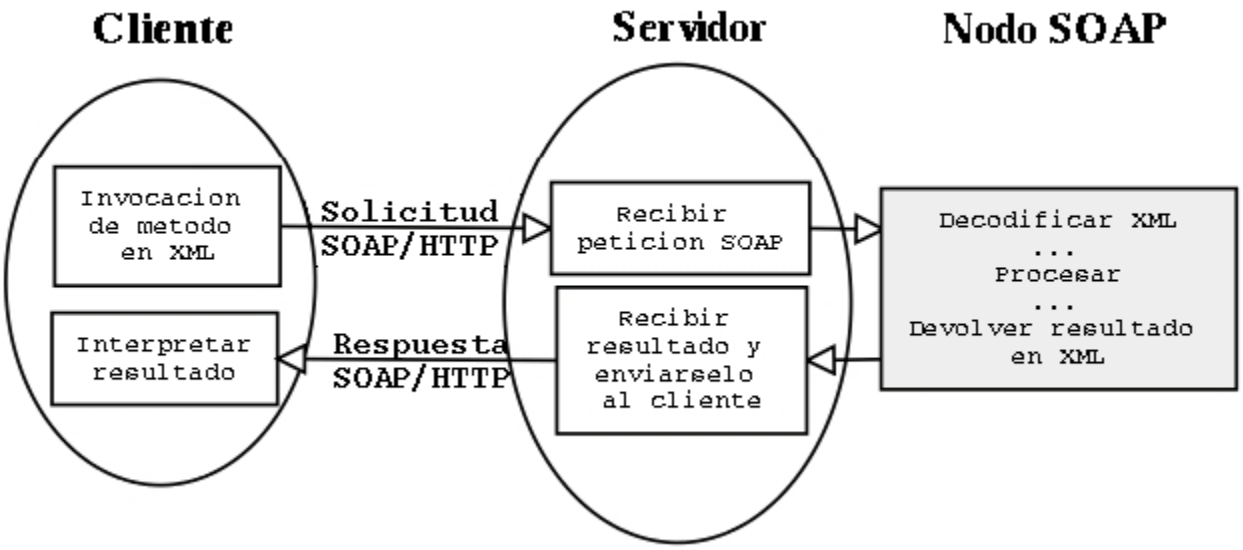


Imagen 2.1.1.1. Funcionamiento SOAP.

- **Ventajas**: En [8] se describen las ventajas que presenta el protocolo SOAP [2] y son las siguientes:

  - Es sencillo a la hora de usarlo, probarlo e implementarlo.

  - Es capaz de atravesar los *firewalls* y los *routers* ya que estos piensan que es una comunicación HTTP [9].

  - Al describirse los datos y funciones en XML (Extensible Markup Language) [13], este protocolo se hace bastante fuerte y a la vez muy sencillo de utilizar.

  - Es independiente del S.O. y del procesador.

  - Puede ser utilizado tanto con autentificación como de manera anónima.

- Tiene una gran comodidad para que se pueda usar cualquier lenguaje de programación que se desee.

- No está agregado de manera sólida a ningún protocolo que sea de transporte ni tampoco a ningún sistema que sea de objetos distribuidos.

- Permite que se puedan comunicar varios entornos entre ellos.

- Se puede transmitir junto con cualquiera de los protocolos de transporte que permita la transmisión de texto.

- **Desventajas**: En [8] [14] se definen las desventajas del protocolo SOAP [2] y aquí se muestran:

  - Tiene dificultad a la hora de entender de manera sencilla cada una de las especificaciones que tiene este protocolo.

  - Convierte elementos como el encabezado en partes opcionales, por lo que puede llegar a generar errores a la hora de intercambiar información.

  - Es dependiente del WSDL [15].

  - Al usar XML [13], este protocolo es más lento que otros middleware al pasar los datos binarios a texto.

- **Estructura de un mensaje SOAP**: En [14] se muestra como un mensaje SOAP [2] está constituido por varias partes que son:

  - ***SOAP Envelope Element* (Envolvente)**: esta parte indica que el mensaje en formato XML [13] se trata de un mensaje SOAP [2]. Es un elemento obligatorio, ya que es el elemento que envuelve este mensaje.

- ***SOAP Header Element* (Cabecera)**: esta parte contiene la información que viene de la aplicación como, por ejemplo, la autenticación. Este elemento debe ser el primero del elemento *SOAP Envelope Element*.

- ***SOAP Body Element* (Cuerpo)**: por último, esta es la que contiene la información con la que se quiere comunicar una aplicación con otra y es un elemento obligatorio en estos mensajes. También tiene un elemento por defecto (*fault*) que contiene cualquiera de los mensajes erróneos que se haya producido en el mensaje SOAP [2] y es opcional indicarlo.

Imagen 2.1.1.2. Estructura de un mensaje SOAP.

Resumiendo, la estructura de un mensaje SOAP [2] se muestra a continuación:

```
<soap:envelope
soap:encodingstyle="http://www.w3.org/2001/12/soap-
encoding" xmlns:soap="http://www.w3.org/2001/12/soap-
envelope">
      <soap:header>
      …
      </soap:header>
```

```
        <soap:body>
              <soap:fault>
              …
              </soap:fault>
        …
        </soap:body>
</soap:envelope>
```

La interfaz SOAP en ILIAS

La interfaz SOAP [2] en ILIAS [1] es una de las herramientas de esta plataforma que permite acceder a la mayor parte de sus funcionalidades accediendo de manera externa desde una aplicación.

El API [7] SOAP [2] de ILIAS [1] dispone, por un lado, de una dirección URL donde el desarrollador puede consultar la información de uso de los servicios disponible; por otro lado, también existe una dirección específica que permite el acceso a la especificación WSDL [15] de dicha API [7].

¿Qué es WSDL?

- ***Web Services Description Language*** [15]: consiste en un documento en formato XML que facilita la descripción y la localización de los servicios web.

- **Objetivos** [16]: la especificación WSDL [15] tiene los siguientes objetivos que son:
    - Realizar la descripción y la localización de los servicios web.

- **Características** [17]:

- Indica a un cliente las funcionalidades que están a su disposición en el servidor.
- Indica cómo se deben utilizar las funcionalidades.
- Permite que un cliente pueda utilizar SOAP [2] para poder llamar a alguna de las funcionalidades que el WSDL [15] facilita.

- **Funcionamiento**:

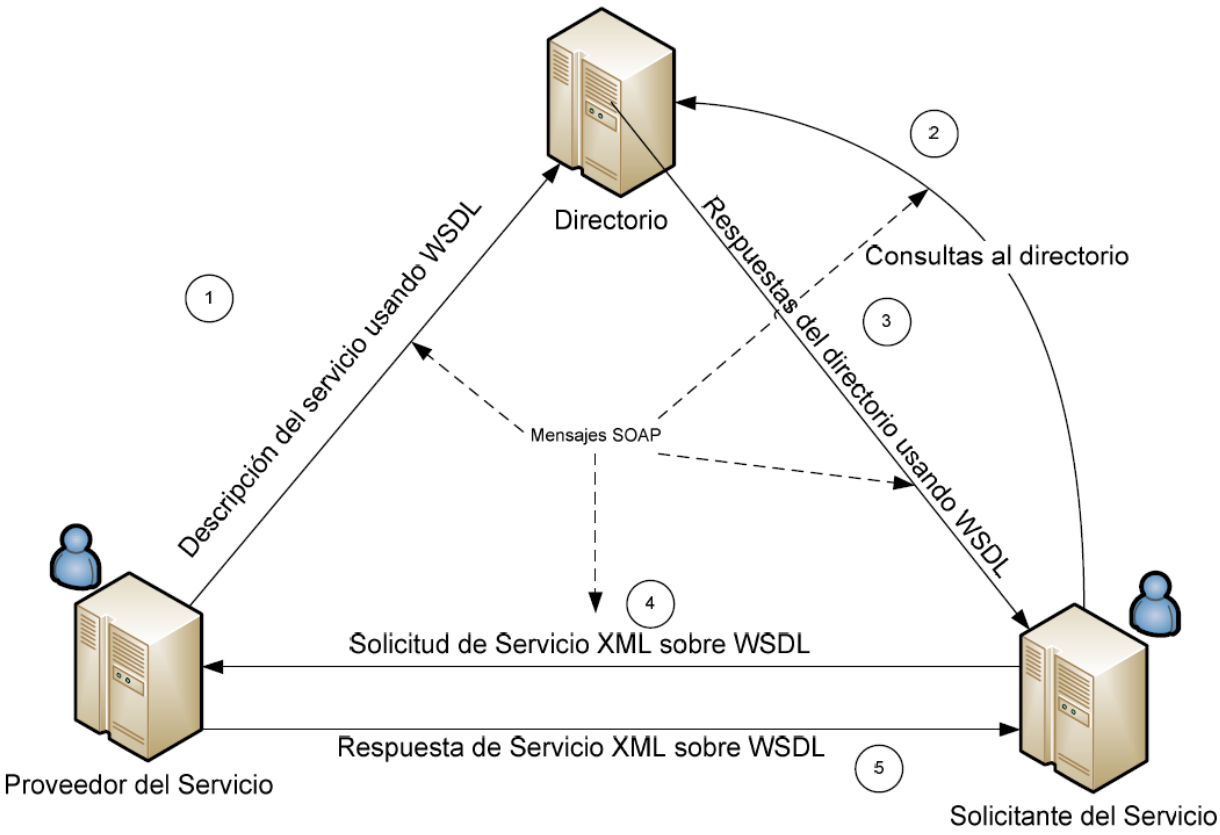


Imagen 2.1.1.3. Funcionamiento WSDL.

1) Se describen los servicios usando el WSDL [15].
2) El solicitante del servicio realiza una consulta de los servicios disponibles en el directorio.

3) El directorio envía una respuesta de la consulta al solicitante.

4) El solicitante envía una solicitud de un servicio al proveedor sobre WSDL [15].

5) El proveedor de dicho servicio envía la respuesta de la consulta al solicitante del mismo.

- **Ventajas**: En [18] se definen las ventajas de esta especificación y son:
    - Proporciona una tecnología que distribuye componentes y que es optimizada.
    - Como SOAP [2] usa el protocolo HTTP [9] como comunicación, esquiva los errores o problemas con firewalls.
    - Con SOAP [2] se pueden llamar fácilmente a los servicios.
    - Los consumidores de los servicios pueden acceder a partir de cualquier plataforma que pueda soportar XML [13] o SOAP [2].
    - Agrupa los datos.

- **Desventajas**: En [19] se describen las desventajas de la especificación WSDL [15] que son:
    - Sólo se puede realizar una operación de servicio WEB por WSDL [15].
    - No puede indicar una serie de operaciones que hacen falta en un intercambio de mensajes.

- **Estructura**: En [20] se describe como está constituida esta especificación por varias partes que son:

    - **Tipos de datos (<*types*>)**: esta parte sirve para indicar cuales son los tipos de datos que se van a utilizar en los mensajes.

    - **Mensajes (<*Message*>)**: aquí se describen los elementos que conforman el mensaje.

    - **Tipos de puerto (<*portType*>)**: en esta parte se indican cuáles son las operaciones que se van a poder realizar y también los mensajes que se intercambian en el servicio.

    - **Bindings (<*binding*>)**: sirve para indicar los protocolos de comunicación que se van a utilizar.

    - **Servicios (<*service*>)**: es donde se definen los puertos y donde se ubican.

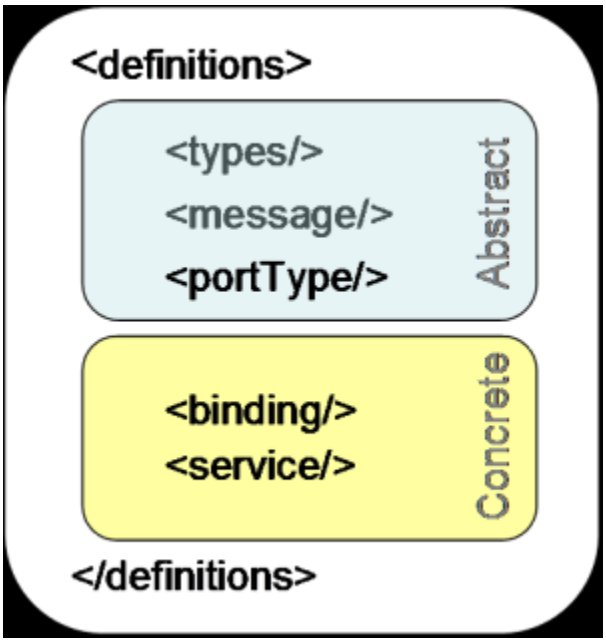


Imagen 2.1.1.4. Estructura del WSDL.

En resumen, la estructura del WSDL [15] se muestra a continuación:

```
<definitions>
      <type>
      …
      </types>
      <message>
      …
      </message>
      <portType>
      …
      </portType>
      <binding>
      …
      </binding>
      <service>
      …
      </service>
</definitions>
```

A continuación se muestran dos tablas resumen, una con los métodos SOAP [2] que se han estudiado y otra con los DTD (*Document Type Definition*) [21] utilizados:

- **Análisis de funcionamiento de métodos SOAP de ILIAS**

  En este apartado mostramos los métodos que se han analizado del API [7] SOAP [2] de ILIAS [1]. Para cada método se han analizado las siguientes características: un resumen describiendo su utilidad, los permisos que tiene para su utilización, el fichero que lo implementa ya que para entender su funcionamiento ha sido necesario revisar el código fuente con la implementación del mismo y por último, también se describen consideraciones, funcionalidades, problemas y propuestas de mejora.

  A continuación se describe cómo deben llamarse a los métodos desde cualquier aplicación que acceda al SOAP [2] de ILIAS [1]:

Para llamar a cualquier método SOAP [2], lo primero es obtener un código SID llamando al método login con el usuario/clave e identificador de instalación. Después, con el SID obtenido, ya puedes llamar a cualquier otro método pasando este dato como parámetro junto con el resto de parámetros que sean necesarios. Por último, una aplicación debería llamar al método logout para cancelar la sesión en plataforma asociada al SID correspondiente.

Un SID es un identificador de seguridad que se va cambiando cada vez que un usuario accede a ILIAS [1], es único y es con el que utiliza todos los métodos SOAP [2].

El SID está asociado a los permisos que el propio usuario tenga en ILIAS [1] para acceder a ciertos elementos. Por lo tanto, de estos permisos que el usuario tenga en la plataforma, dependerá el que pueda completar o no las llamadas a las operaciones SOAP [2] solicitadas. No obstante, esto sería lo ideal en teoría, pero se ha detectado (como se detalla en la tabla 2.1.1.1) que en algunos métodos esto no es así y se requieren permisos de administrador para hacer por SOAP [2] ciertas operaciones que como usuario web en plataforma sí que se podrían hacer sin serlo. Para llegar a esta conclusión, se ha tenido que revisar el código fuente de la implementación para entender estos detalles de restricciones de uso que no estaban claramente especificados en la documentación SOAP [2] de los métodos tal y como la muestra ILIAS [1]. Por este motivo se adjuntan en la tabla 2.1.1.1 los nombres de los ficheros php de ILIAS [1] que contienen cada una de esas implementaciones específicas

Tabla resumen de los métodos SOAP estudiados y utilizados en su caso

Método	**Resumen sobre su utilidad**	**Permisos de uso**	**Fichero de implementación (*dir. webservices/soap/classes*)**	**Consideraciones de uso. Utilizado o no. Donde se usa en la aplicación. Problemas encontrados para NO utilizarlo. Propuestas de mejora del método.**
login	obtiene el token, SID, de autorización a otros webservices	cualquier usuario	class.ilSoapUserAdministration.php	Utilizado para la conexión con la instalación ILIAS [1] seleccionada

assignCourseMember	añade un usuario a un curso o grupo existente	administrador del espacio	class.ilSoapCourseAdministration.php	Utilizado para añadir usuarios a un curso o grupo del que el usuario que lo desea sea administrador de dicho curso o grupo. Propuesta de mejora: poder añadir un usuario pasándole su nombre y apellidos
excludeCourseMember	elimina un usuario de un curso o grupo existente	administrador del espacio	class.ilSoapCourseAdministration.php	Utilizado para eliminar usuarios de un curso o grupo del que el usuario que lo desea sea administrador de dicho curso o grupo
distributeMails	envía correos al correo interno de ILIAS	cualquier usuario	class.ilSoapUtils.php	Utilizado para enviar correos al correo interno de ILIAS [1]. Propuesta de mejora: poder enviar correos al red.ujaen.es
getRoles	devuelve todos los roles de un elemento	cualquier usuario	class.ilSoapRBACAdministration.php	Utilizado para obtener todos los cursos y grupos a los que un usuario está matriculado
logout	cierra la sesión de un usuario	cualquier usuario	class.ilSoapUserAdministration.php	Utilizado para cerrar la sesión de un usuario conectado a ILIAS [1] cuando el usuario pulsa sobre el botón de salir de la aplicación
hasNewMail	comprueba si han llegado nuevos correos al correo interno de ILIAS	cualquier usuario	class.ilSoapUserAdministration.php	Utilizado para comprobar si a un usuario le han llegado nuevos correos al correo interno de ILIAS [1]. Propuesta de mejora: obtener también el autor del correo recibido
getUserIdBySid	devuelve el identificador de un usuario a través de su sid	cualquier usuario	class.ilSoapUserAdministration.php	No utilizado. Se utilizó en un principio para obtener el identificador de un usuario pasándole su sid de usuario pero al final no ha sido necesario utilizar este método
getCourseXML	devuelve en formato XML un descripción de un curso específico	Administrador del espacio	class.ilSoapCourseAdministration.php	No utilizado. No permite obtener la descripción en formato XML [13] de un curso específico a los usuarios que no sean administradores del curso
getCoursesForUser	devuelve los cursos en los que un usuario está matriculado	Administrador del espacio	class.ilSoapCourseAdministration.php	No utilizado. No permite obtener cursos para usuarios que no sean administradores del curso. Además, sólo permite obtener cursos y no grupos. Ver

				versión más actualizada y genérica getRoles
getTestResults	devuelve el resultado de un test	Administrador del espacio	class.ilSoapTestAdministration.php	No utilizado. Se intentó utilizar para obtener los datos de los usuarios pero sólo lo pueden usar los administradores del espacio y sólo para alumnos que hayan realizado el tests. Para obtener dichos datos ya se vió que se podía usar el método getUsersForContainer
getTestUserData	devuelve los datos de un usuario activo	Cualquier usuario	class.ilSoapTestAdministration.php	No utilizado. Se intentó utilizar para obtener los datos de los usuarios. Sólo puede ser utilizado para usuarios que están haciendo el tests. Para obtener dichos datos ya se vió que se podía usar el método getUsersForContainer
getUserXML	devuelve en formato XML los datos de un usuario	Administrador del espacio	class.ilSoapUserAdministration.php	No utilizado. No permite obtener los datos de un usuario en formato XML [13] para usuarios que no sean administradores del espacio
getUsersForContainer	devuelve todos los usuarios de un identificador de referencia específico	cualquier usuario	class.ilSoapUserAdministration.php	Utilizado para obtener a todos los usuarios que pertenecen a un curso o grupo específico. Propuesta de mejora: devolver también el nombre y los apellidos de cada usuario

Tabla 2.1.1.1. Métodos SOAP estudiados y utilizados en su caso.

Resumen de los documentos DTD con formatos de mensajes empleados

- Documento 1:

  - XML: ilias_mail_transport_4_0.dtd
  - Utilidad: formato XML para colección de mensajes de correo en plataforma.
  - Uso: utilizado para envío de mensajes simples de la aplicación.

- Elementos (y sus atributos) usados: **Mails** [**Mail+** (**type, usePlaceholders**) [Sender (obj_id, name), **To+** (obj_id, name), Cc* (obj_id, name), Bcc* (obj_id, name), **Subject** [#PCDATA], **Message?** [**P*** [#PCDATA]], Attachment* [#PCDATA] (name)]]

- Documento 2:

- XML: ilias_role_object_3_10.dtd
- Utilidad: formato XML para colección de roles.
- Uso: utilizado para obtención de cursos y grupos a los que un usuario está matriculado.
- Elementos (y sus atributos) usados: **Roles** [**Role*** (**id, role_type**) [**Title** [#PCDATA], **Description** [#PCDATA], **Translation?** [#PCDATA], **AssignedObject?** (**obj_id, ref_id, type**) [**Title, Description, Path?** [**Element*** (**ref_id, type**) [#PCDATA]]]]]

- Documento 3:

- XML: ilias_style_0_1.dtd
- Utilidad: formato XML para estilo de la aplicación.
- Uso: sólo se ha utilizado el atributo StyleSheet en cada una de las ventanas de la aplicación llamando al .css creado para esta aplicación (Style.css)

- Documento 4:

- XML: ilias_user_4_0.dtd
- Utilidad: formato XML para colección de usuarios.
- Uso: utilizado para obtención de usuarios de un curso o grupo. Creada extensión de las clases SoapClientUsers y SoapClientUser para obtener todos los datos necesarios llamadas SoapClientUsersExt y SoapClientUserExt.
- Elementos (y sus atributos) usados: **Users** [UDFDefinitions? [UDFDefinition* (Id, Type, Visible, Changeable, Required, Searchable, CourseExport, Export)

[UDFName, UDFValue*]], **User*** (**Id, Language, Action**) [**Login** [#PCDATA], **Role*** (**Id, Type**, Action) [#PCDATA], Password? (Type) [#PCDATA], **Firstname?** [#PCDATA], **Lastname?** [#PCDATA], Title? [#PCDATA], PersonalPicture? (imageType) [#PCDATA], Gender? [#PCDATA], Email? [#PCDATA], Birthday? [#PCDATA], Institution? [#PCDATA], Street? [#PCDATA], City? [#PCDATA], PostalCode? [#PCDATA], Country? [#PCDATA], PhoneOffice? [#PCDATA], PhoneHome? [#PCDATA], PhoneMobile? [#PCDATA], Fax? [#PCDATA], Hobby? [#PCDATA], Department? [#PCDATA], Comment? [#PCDATA], Matriculation? [#PCDATA], **Active?** [#PCDATA], ClientIP? [#PCDATA], **TimeLimitOwner?** [#PCDATA], **TimeLimitUnlimited?** [#PCDATA], **TimeLimitFrom?** [#PCDATA], **TimeLimitUntil?** [#PCDATA], **TimeLimitMessage?** [#PCDATA], **ApproveDate?** [#PCDATA], **AgreeDate?** [#PCDATA], iLincID? [#PCDATA], iLincLogin? [#PCDATA], iLincPasswd? [#PCDATA], **AuthMode?** (%A_Mode) [#PCDATA], **ExternalAccount?** [#PCDATA], Look? (Skin, Style) [#PCDATA], **LastUpdate?** [#PCDATA], **LastLogin?** [#PCDATA], UserDefinedField* (Id, Name) [#PCDATA], AccountInfo* (Type) [#PCDATA], GMapsInfo? (longitude, latitude, zoom)]]

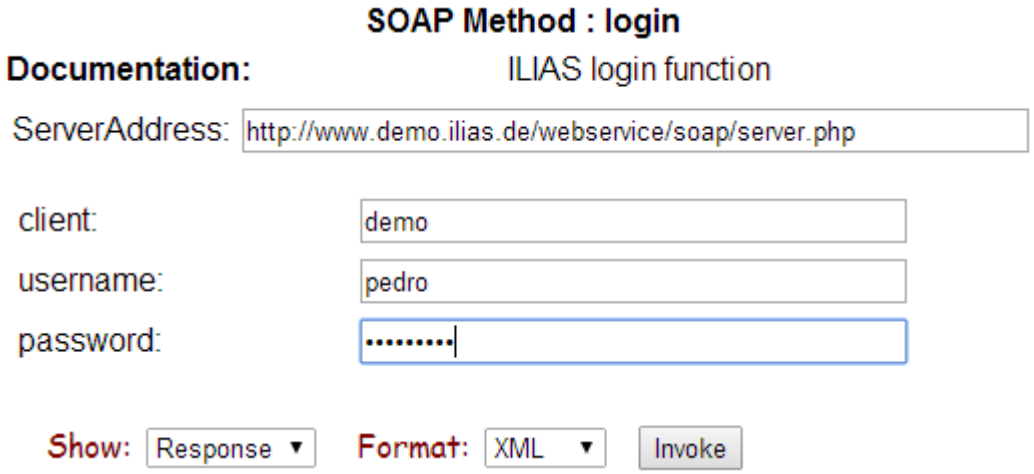
- ★ En **negrita** se muestran los elementos y sus atributos que han sido mapeados y utilizados.
- ★ Los () significan que son atributos de ese elemento.
- ★ Los [] significan que son elementos que contiene ese elemento.

A continuación se muestran algunos ejemplos simples de uso (XML [13] resultantes) obtenidos utilizando la herramienta SoapClient: http://www.soapclient.com/SoapClient.

Lo primero que hay que hacer es introducir la URL de la especificación WSDL [15] como se muestra en el ejemplo a continuación:

Una vez se haya pulsado sobre el botón Retrieve, hay que buscar el método deseado de entre todos los disponibles e introducir los datos que se pidan en dicho método.

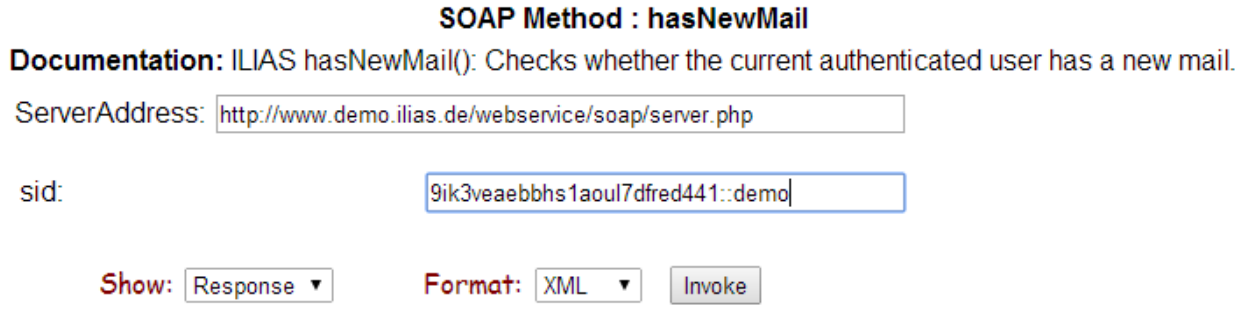
- En el siguiente ejemplo se va a llamar al método login:



Al pulsar sobre el botón Invoke se muestra el XML [13] resultante y es el siguiente:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" 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 xmlns:ns1="urn:ilUserAdministration">
            <sid xsi:type="xsd:string">9ik3veaebbhs1aoul7dfred441::demo</sid>
            </ns1:loginResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

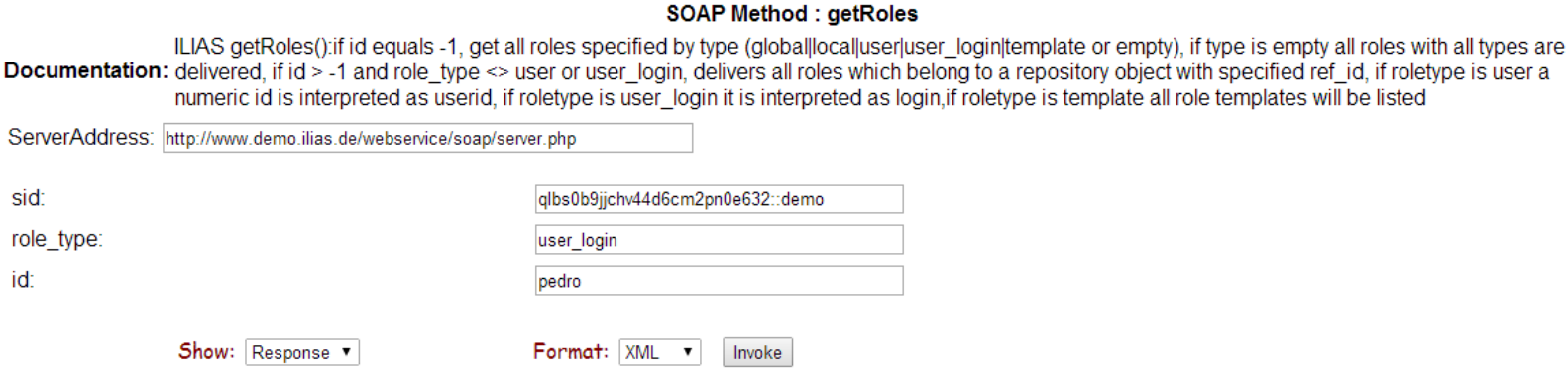
- En el siguiente ejemplo se va a llamar al método hasNewMail:



Al pulsar sobre el botón Invoke se muestra el XML [13] resultante y es el siguiente:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" 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:hasNewMailResponse xmlns:ns1="urn:ilUserAdministration">
            <status xsi:type="xsd:boolean">false</status>
            </ns1:hasNewMailResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- En el siguiente ejemplo se va a llamar al método getRoles:

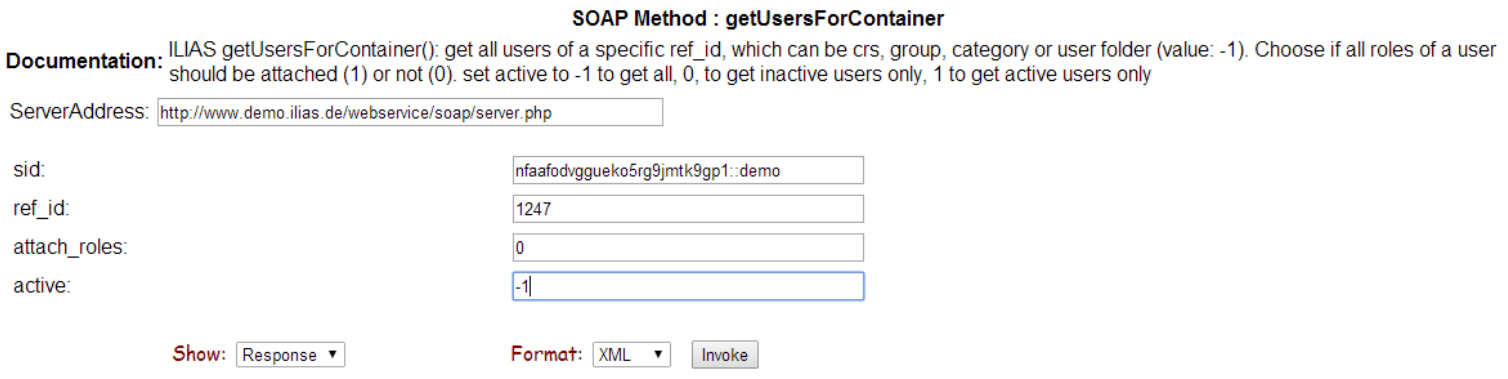


Al pulsar sobre el botón Invoke se muestra el XML [13] resultante y es el siguiente:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" 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:getRolesResponse xmlns:ns1="urn:ilUserAdministration">
<role_xml xsi:type="xsd:string">
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE Roles PUBLIC "-//ILIAS//DTD
ILIAS Roles//EN"
"http://www.demo.ilias.de:80/xml/ilias_role_object_3_10.dtd"><!--Roles
information of ilias system--><Roles><Role role_type="Global"
id="il_4242_role_435"><Title>Tutor</Title><Description>Member of education
institution - create permissions for content in selected
areas</Description><Translation>Tutor</Translation></Role><Role
role_type="Local"
id="il_4242_role_2161"><Title>il_crs_admin_1247</Title><Description>Admin of
course obj_no.2159</Description><Translation>Administrador del
```

```
curso</Translation><AssignedObject obj_id="il_4242_crs_2159" ref_id="1247"
type="crs"><Title>Curso 1</Title><Description/><Path><Element ref_id="1"
type="root">Espacios</Element><Element ref_id="574"
type="cat">Sandbox</Element></Path></AssignedObject></Role><Role
role_type="Local"
id="il_4242_role_2166"><Title>il_grp_admin_1249</Title><Description>Groupadmi
n of group
obj_no.2164</Description><Translation>Administrador</Translation><AssignedObj
ect obj_id="il_4242_grp_2164" ref_id="1249" type="grp"><Title>Grupo
1</Title><Description/><Path><Element ref_id="1"
type="root">Espacios</Element><Element ref_id="574"
type="cat">Sandbox</Element></Path></AssignedObject></Role><Role
role_type="Local"
id="il_4242_role_2170"><Title>il_grp_admin_1251</Title><Description>Groupadmi
n of group
obj_no.2168</Description><Translation>Administrador</Translation><AssignedObj
ect obj_id="il_4242_grp_2168" ref_id="1251" type="grp"><Title>Grupo
2</Title><Description/><Path><Element ref_id="1"
type="root">Espacios</Element><Element ref_id="574"
type="cat">Sandbox</Element></Path></AssignedObject></Role><Role
role_type="Local"
id="il_4242_role_2192"><Title>il_crs_admin_1265</Title><Description>Admin of
course obj_no.2190</Description><Translation>Administrador del
curso</Translation><AssignedObject obj_id="il_4242_crs_2190" ref_id="1265"
type="crs"><Title>Peligros</Title><Description>Aprendes a identificar los
peligros</Description><Path><Element ref_id="1"
type="root">Espacios</Element><Element ref_id="274" type="cat">Features and
Scenarios</Element><Element ref_id="280" type="cat">Course
Management</Element><Element ref_id="1026"
type="cat">Sandbox</Element></Path></AssignedObject></Role></Roles>
</role_xml>
</ns1:getRolesResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- En el siguiente ejemplo se va a llamar al método getUsersForContainer:



Al pulsar sobre el botón Invoke se muestra el XML [13] resultante y es el siguiente:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" 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:getUsersForContainerResponse xmlns:ns1="urn:ilUserAdministration">
<user_xml xsi:type="xsd:string">
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE Users PUBLIC "-//ILIAS//DTD
UserImport//EN" "http://www.demo.ilias.de:80/xml/ilias_user_4_0.dtd"><!--User
of ilias system--><Users><UDFDefinitions></UDFDefinitions><User
Id="il_4242_usr_2081" Language="en"
Action="Update"><Login>maria</Login><Active>true</Active><TimeLimitOwner>7</T
imeLimitOwner><TimeLimitUnlimited>1</TimeLimitUnlimited><TimeLimitFrom>138537
6480</TimeLimitFrom><TimeLimitUntil>1385376480</TimeLimitUntil><ApproveDate>2
013-11-25 11:53:13</ApproveDate><AgreeDate>2014-08-29
03:04:57</AgreeDate><AuthMode type="default"/><LastUpdate>2013-11-25
11:53:14</LastUpdate><LastLogin>2014-08-31 19:02:57</LastLogin></User><User
Id="il_4242_usr_2083" Language="en"
Action="Update"><Login>carlos</Login><Active>true</Active><TimeLimitOwner>7</
TimeLimitOwner><TimeLimitUnlimited>1</TimeLimitUnlimited><TimeLimitFrom>13853
76900</TimeLimitFrom><TimeLimitUntil>1385376900</TimeLimitUntil><ApproveDate>
2013-11-25 12:00:19</ApproveDate><AgreeDate>2014-08-29
03:05:22</AgreeDate><AuthMode type="default"/><LastUpdate>2013-11-25
12:00:19</LastUpdate><LastLogin>2014-08-29 03:05:17</LastLogin></User><User
Id="il_4242_usr_2082" Language="en"
Action="Update"><Login>lola</Login><Active>true</Active><TimeLimitOwner>7</Ti
meLimitOwner><TimeLimitUnlimited>1</TimeLimitUnlimited><TimeLimitFrom>1385376
840</TimeLimitFrom><TimeLimitUntil>1385376840</TimeLimitUntil><ApproveDate>20
13-11-25 11:55:33</ApproveDate><AgreeDate>2014-08-29
03:05:42</AgreeDate><AuthMode type="default"/><LastUpdate>2013-11-25
11:55:33</LastUpdate><LastLogin>2014-08-29 18:27:32</LastLogin></User><User
Id="il_4242_usr_2084" Language="en"
Action="Update"><Login>pedro</Login><Active>true</Active><TimeLimitOwner>7</T
imeLimitOwner><TimeLimitUnlimited>1</TimeLimitUnlimited><TimeLimitFrom>138537
7200</TimeLimitFrom><TimeLimitUntil>1385377200</TimeLimitUntil><ApproveDate>2
013-11-25 12:03:05</ApproveDate><AgreeDate>2014-08-28
20:19:02</AgreeDate><AuthMode type="default"/><LastUpdate>2013-11-25
12:03:05</LastUpdate><LastLogin>2014-09-01
09:37:32</LastLogin></User></Users>
</user_xml>
</ns1:getUsersForContainerResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

De esta manera se han estado haciendo pruebas del XML [13] que se devolvía para comprobar si el XML [13] resultante era el deseado. Estos métodos son algunos de los que han sido utilizados en ese proyecto y se encuentran definidos en la tabla 2.1.1.1.

ILIAS [1] es una plataforma basada en software libre, lo que facilita la extensión de sus funcionalidades básicas, tal y como se refleja en varios proyectos fin de carrera [22], [23] y [24].

2.2. Acceso a servicios SOAP desde JAVA usando JAXB

Para que haya comunicación entre SOAP [2] y Java se utiliza JAXB [] JAXB [6] facilita una API [7] para poder mapear entre los documentos en formato XML [13] y los objetos JAVA [5] de manera automática a través de una aplicación.

2.2.1. ¿Qué es JAXB?

- ***Java Architecture for XML Binding*** [6]: consiste en un estándar que realiza la modificación de un documento en formato XML [13] a un objeto JAVA [5] (*unmarshall*) y viceversa (*marshall*).

- **Objetivos** [25]:
    - Intercambiar datos entre aplicaciones o programas.

- **Características** [26]:
    - Utiliza tanto la tecnología XML [13] como Java.
    - Aseguran que los datos que se transmiten son válidos.
    - Es una aplicación rápida.
    - Es sencilla de utilizar.
    - Permite que los datos se puedan limitar.
    - Permite extenderlo.

- **Funcionamiento**:

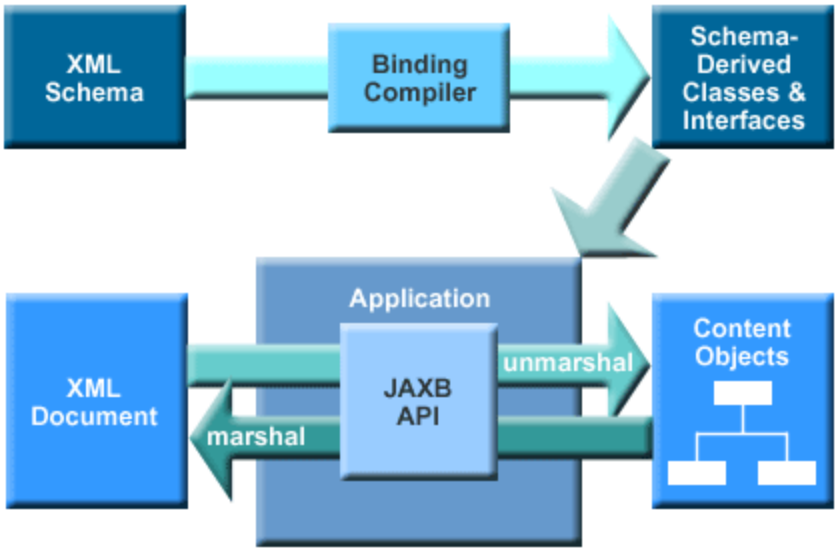


Imagen 2.2.1. Funcionamiento JAXB.

- **Ventajas**: En [25] se describen las ventajas de JAXB [6] y son las siguientes:
    - Concede los que desarrollan en lenguaje Java que puedan acceder a los esquemas de XML [13] y poder tratarlos sin que haya que ver cómo funcionan por dentro.
    - Si alguno de los esquemas es dificultoso y es muy dinámico, JAXB [6] tiene una personalización a la hora de realizar el marshall y el unmarshall que es muy conveniente usarlo en estos casos, ya que al ponerse a regularizar definiciones de Java puede acarrear bastante tiempo y por lo tanto pueden aparecer errores con más facilidad.
    - Es capaz de acceder a los datos que hay en memoria.
    - Lo que hace es procesar solamente aquellos datos que realmente son válidos.
    - Transforma los datos a distintos tipos.
    - Es capaz de generar clases que están asentadas en algún DTD [21].

- Es capaz de construir objetos a partir de documentos XML [13] y viceversa.

- **Desventajas**: En [25] se describen las desventajas de JAXB [6] y son las siguientes:

  - En el caso de tener que validar un archivo XML [13] ya generado, tiene que leerlo para validarlo.

2.2.2. Uso de JAXB en JAVA

JAXB [6] hace uso de anotaciones con las que se mapean las clases junto con los atributos y métodos que se quieren mapear para poder utilizar la API [7] que proporciona.

En este proyecto se han usado las siguientes anotaciones [27] [28] [29] [30]:

- **@XmlRootElement**: esta anotación le dice a JAVA [5] que este va a ser el elemento raíz XML [13] del fichero XML [13].

  - Parámetros: esta anotación tiene dos parámetros que son:

    - *name*: indica el nombre local del elemento.

    - *namespace*: indica el nombre del espacio de nombres del elemento.

- **@XmlSeeAlso**: esta anotación marca que una clase está enlazada con otra clase.

  - Parámetros: esta anotación tiene un parámetro llamado value.

- **@XmlAccessorType**: esta anotación marca como se quiere que se realice un marshall a un archivo a o desde XML [13].

- Parámetros: esta anotación tiene un parámetro llamado value y que puede contener los siguientes valores:

  - *XmlAccessType.FIELD*: Cada atributo y elemento que no sea estático y tampoco temporal será de manera automática mapeado al documento XML, al menos que haya algún atributo y elemento con la anotación @XmlTransient.

  - *XmlAccessType.NONE*: Ninguno de los atributos y elementos será mapeado a XML [13] a no ser que se indique con alguna anotación de JAXB [6].

  - *XmlAccessType.PROPERTY*: Cada uno de los getter y setter será mapeado de manera automática a XML [13] sino tiene la anotación @XmlTransient.

  - *XmlAccessType.PUBLIC_MEMBER*: Cada uno de los getter y setter que sea público y cada atributo y elemento que también sea público será mapeado de manera automática a XML [13], a no ser que tenga la anotación @XmlTransient.

- **@XmlType**: esta anotación marca cual es el orden que van a tener los elementos XML [13].

  - Parámetros: esta anotación tiene varios parámetros que son los siguientes:

    - *factoryClass*: indica una clase que contiene un constructor por defecto para crear una instancia de la clase.

    - *factoryMethod*: indica el nombre del constructor por defecto de factoryClass.

        - *name*: indica el nombre del tipo de esquema XML [13] que se asigna a la clase.
        - *namespace*: indica el nombre del espacio de nombres del tipo de esquema XML [13].
        - *propOrder*: indica el orden de los elementos de esquema XML [13] cuando la clase se asigna a un tipo complejo de esquema XML [13].

- **@XmlTransient**: esta anotación marca que este elemento no se va a mapear.
- **@XmlElement**: esta anotación marca que este elemento va a ser un elemento unido.
    - Parámetros: esta anotación tienes varios parámetros que son los siguientes:
        - *defaultValue*: indica el valor por defecto del elemento.
        - *name*: indica el nombre del elemento XML [13].
        - *namespace*: indica el espacio de nombres XML [13] del esquema XML [13].
        - *nillable*: indica la personalización del elemento.
        - *required*: indica la personalización del elemento que se requiera.
        - *type*: indica la clase java a la que hace referencia.
- **@XmlAttribute**: esta anotación marca que este elemento va a ser un atributo.
    - Parámetros: esta anotación tienes varios parámetros que son los siguientes:

- *name*: indica el nombre del atrbuto XML [13].
- *namespace*: indica el espacio de nombres XML [13] del atributo de esquema XML [13].
- required: indica si el atributo es opcional u obligatorio.

2.2.3. Ejemplos de unas clases sencillas mapeadas a código XML

- Esta es la clase que realiza la petición para obtener los roles (cursos, grupos, etc) de un usuario:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "getRoles", namespace =
"urn:ilUserAdministration")
@XmlType(propOrder = {"sid", "role_type", "id"})
public class SoapClientGetRolesRequest extends
SoapClientRequest {

    @XmlTransient
    private static final Logger logger =
Logger.getLogger(SoapClientGetRolesRequest.class.getName());

    private String role_type;

    private String id;

    public SoapClientGetRolesRequest() {
    }

    public String getRole_type() {
        return role_type;
    }

    public void setRole_type(String role_type) {
        this.role_type = role_type;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
```

```
    }
}
```

- Esta es la clase que contiene todos los roles (cursos, grupos, etc) de un usuario:

```
@XmlRootElement(name = "Roles")
public class SoapClientRoles {

@XmlTransient
private static Logger logger =
Logger.getLogger(SoapClientRoles.class.getName());

private List<SoapClientRole> roles;

@XmlElement(name = "Role")
public List<SoapClientRole> getRoles() {
      if(roles == null) {
            return roles = new ArrayList<SoapClientRole>();
      }
      return roles;
}
}
```

- Esta es la clase que contiene a un rol (curso, grupo, etc) de un usuario:

```
@XmlAccessorType(XmlAccessType.FIELD)
public class SoapClientRole {

    @XmlTransient
    private static Logger logger =
Logger.getLogger(SoapClientRole.class.getName());

    @XmlAttribute( name = "role_type")
    private String role_type;

    @XmlAttribute( name = "id")
    private String id;

    @XmlElement (name = "Title")
    private String title;

    @XmlElement (name = "Description")
    private String description;
```

```
    @XmlElement (name = "Translation")
    private String translation;

    @XmlElement (name = "AssignedObject")
    private SoapClientAssignedObject assignedObject;

    public static Logger getLogger() {
        return logger;
    }

    public String getId() {
        return id;
    }

    public String getRole_type() {
        return role_type;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getTranslation() {
        return translation;
    }

    public SoapClientAssignedObject getAssignedObject() {
        return assignedObject;
    }
}
```

2.3. Framework de desarrollo JavaFX

2.3.1. ¿Qué es JavaFX?

- **JavaFX** [31]: este framework es un paso más allá de la plataforma Java creado para que los desarrolladores de aplicaciones puedan crear aplicaciones RIA (*Rich Internet Applications*) [32] que tienen el mismo comportamiento en diferentes plataformas. Permite utilizar cualquier biblioteca de JAVA [5] para aplicaciones en JavaFX [3].

- **Objetivos** [33]:

    - Reducir lo mejor que se pueda la complejidad del desarrollo que de otra manera solicita JAVA [5].

- **Características** [31]:

    - *FXML* [34]: se trata de un lenguaje que está basado en el lenguaje XML [13] y facilita una estructura para la creación de la interfaz de usuario con la que se puede trabajar de manera independiente a como se trabaja con el resto del código. Para trabajar con este lenguaje, al empezar se usa lo siguiente: el modelo (es una clase JAVA [5] que contiene el código necesario para llevar a cabo una aplicación FXML [34]), la vista (es un archivo FXML [34] que contiene la interfaz de usuario) y el controlador (es una clase JAVA [5] que controla, cuando se interactúa con la interfaz de usuario, todos los movimientos).

    - *Double binding*: un binding es una clase que contiene muchas funciones que son muy útiles para el desarrollador.

Un double binding facilita gran parte de estas funciones para crear un binding de un valor de tipo double.

- Ejemplo de funciones de un binding:

```
textDb = TextBuilder.create()
  .layoutX(18)
  .layoutY(69)
  .textOrigin(VPos.TOP)
  .fill(Color.web("#131021"))
  .font(Font.font("SansSerif", FontWeight.BOLD, 18))
  .build(),
slider = SliderBuilder.create()
  .layoutX(135)
  .layoutY(69)
  .prefWidth(162)
  .min(acModel.minDecibels)
  .max(acModel.maxDecibels)
  .build(),
```

- *Patrón builder*: se trata de un patrón de diseño que permite diseñar un objeto que sea complejo desde unas partes que juntándose forman este objeto complejo.

- Permite introducir en una aplicación lo que son sonidos, videos, animaciones y gráficos.

- Permite usar cualquiera de las bibliotecas que tiene JAVA [5] en una aplicación con JavaFX [3].

- Permite que tanto diseñadores como desarrolladores puedan trabajar y colaborar unos con otros de manera cómoda.

- **Arquitectura**:

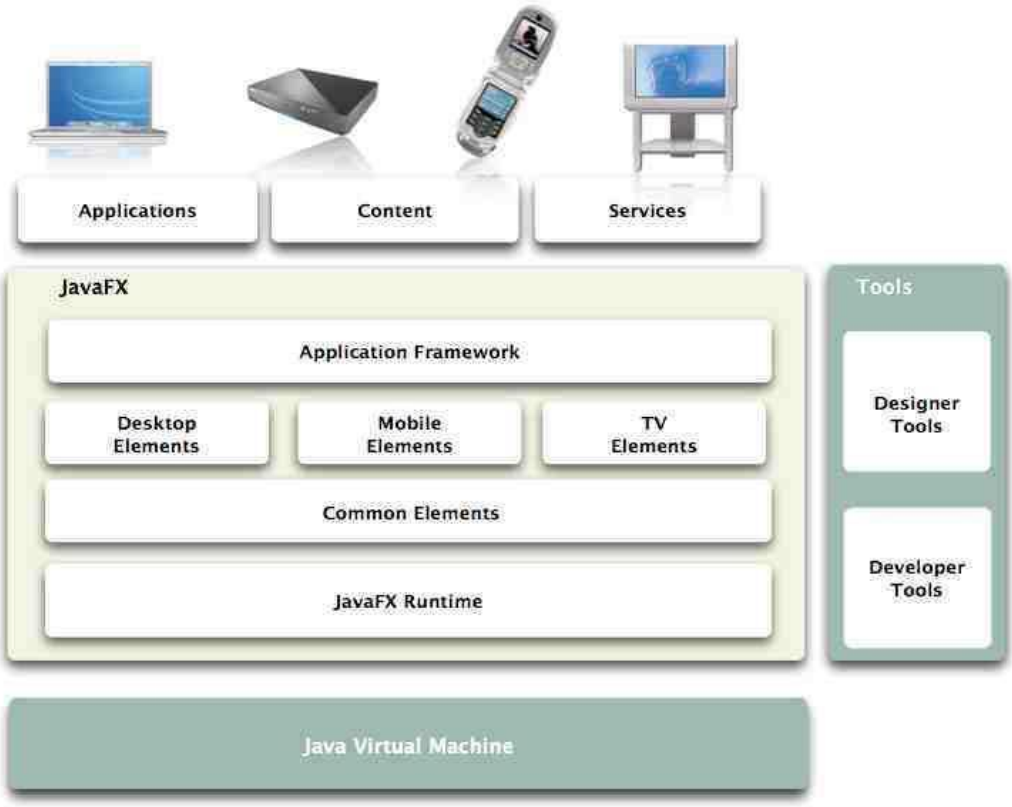


Imagen 2.3.1. Arquitectura JavaFX.

Con JavaFX [3] se pueden realizar tanto aplicaciones como contenidos como servicios, por lo que es un framework muy completo.

- **Ventajas**: En [35] se describen las ventajas de JavaFX [3] y son las siguientes:
    - Proporciona interactividad con otros lenguajes. Además de proporcionar también programación y animación con otros.
    - Permite que se puedan crear contenidos bien acomodados en medios.
    - Es multiplataforma, es decir, puede ser para los escritorios, los móviles, etc.

- Usa siempre el mismo lenguaje tanto para desarrollar en web, como para el escritorio o para los móviles.

- Compila las aplicaciones en vez de interpretarlas, por lo que hace que sean más rápidas.

- Tiene una sintaxis de código muy inteligente, por lo que se pueden ver las modificaciones que se realicen en directo dentro del IDE.

- **Desventajas**: En [35] se describen las desventajas de JavaFX [3] y son las siguientes:

  - Hace falta incluir un lenguaje de script que sea complejo y este debe sostener en la memoria una estructura semántica y unos objetos de plataforma propios a la vez que lo hace con objetos gráficos.

  - Como se trata de un lenguaje de script, introduce una ralentización muy grande, haciendo así que el trabajo no llegue a su punto óptimo.

  - Como las APIs para el manejo del XML [13] ya están incluidas en el JDK, sería más ligero trabajar con templates que trabajar con JavaFX [3].

2.3.2. Uso de JavaFX en ILIAS

Se ha analizado cómo se usaría el framework JavaFX [3] para poder crear una aplicación que permita interactuar con ILIAS [1] y así poder realizar cualquier función que se desee dentro de los objetivos de este proyecto.

Existe un applet implementado con JavaFX [3] que se encarga opcionalmente de la gestión de archivos desde el navegador. Además, en el código de este applet existe una clase llamada FileManager que es la clase principal. Este fichero contiene toda la información tanto del usuario que se conecta a ILIAS [1] como de

la conexión en sí misma. Esta clase es la que da comienzo a que se ejecute la aplicación ya que es la encargada de realizar la conexión con la plataforma ILIAS [1], pero en este proyecto se ha utilizado una clase principal separada de esta llamada ILIAS [1] y es con la que se va a trabajar.

La clase FileManager tiene algunas limitaciones debido a que está pensada para que funcione en una applet, por lo que para este proyecto se ha tenido que encajar un poco para que pueda funcionar en el mismo.

También existe su controlador llamado MainController que se encarga del control de todo lo que se visualiza en la vista del usuario.

Este código se podría utilizar como punto de partida para el desarrollo del prototipo lo que permitiría garantizar su mantenimiento en el futuro al estar basado en código de la propia plataforma.

2.4. Propuesta de solución

El sistema que se va a desarrollar se trata de una aplicación de escritorio con el que se va a poder acceder a ILIAS [1] y se va a poder realizar la gestión de los contenidos, la gestión de los usuarios y la gestión de las instalaciones.

Para ello va a ser necesario un programa de desarrollo como es el caso de Netbeans, un programa para realizar una interfaz de usuario de manera más sencilla como es el JavaFX Scene Builder y como venimos comentando a lo largo de este documento la tecnología JavaFX [3] junto con el lenguaje XML [13] y el lenguaje JAXB [6].

Otra cosa que hay que tener en cuenta es el tiempo que se le va a dedicar al proyecto para su finalización, que en este caso ha sido de unos 178 días, y lo que va a cobrar el analista programador en estos 178 días.

Se ha utilizado una metodología Scrum en la que se ha ido desarrollando de un forma incremental en cada uno de los sprint (cada periodo de tiempo comprendido entre una y cuatro semanas) en las que se han

ido realizando desarrollos que se puedan utilizar. En cada sprint se ha realizado una reunión en la que se ha ido viendo lo que se ha realizado en el actual sprint y lo que se realizará en el siguiente, de esta manera el analista programador se compromete a realizar el trabajo hablado en la reunión para el siguiente sprint. En esta reunión también ha participado el cliente para ver qué es lo que quiere y lo que necesita.

2.5. Requerimientos del sistema

- Requerimientos funcionales:

1) El sistema permitirá el acceso autentificado a la plataforma utilizando las mismas credenciales de acceso.

2) El sistema permitirá visualizar las instalaciones que tiene en ese momento guardado.

3) El sistema permitirá añadir una instalación en sí mismo.

4) El sistema permitirá borrar una instalación de sí mismo.

5) El sistema permitirá que editar una instalación que contiene guardada.

6) El sistema permitirá comprobar si el usuario tiene nuevos correos.

7) El sistema permitirá visualizar la lista de cursos y/o grupos a los que está matriculado un usuario.

8) El sistema permitirá visualizar la lista de los usuarios que están matriculados en un curso o grupo.

9) El sistema permitirá dar de alta un usuario a un curso o grupo.

10) El sistema permitirá dar de baja a un usuario de un curso o grupo.

11) El sistema permitirá enviar un correo a usuarios matriculados en un curso o grupo.

12) El sistema permitirá obtener un fichero Excel que contendrá la lista de todos los usuarios matriculados en un curso o grupo.

13) El sistema permitirá gestionar los contenidos de un curso o grupo.

- Requerimientos no funcionales:

1) La interfaz debe ser intuitiva para el usuario.

2) La aplicación debe realizar lo pedido en los objetivos de este proyecto, pudiéndose añadir cualquier otra funcionalidad que sea de utilidad.

3) La aplicación debe ser multiplataforma pudiendo ejecutarse en cualquier SO de escritorio.

4) El desarrollo debe seguir en la medida de lo posible las pautas de implementación de otro software dentro del proyecto ILIAS [1] para facilitar su integración y mantenimiento en el futuro.

5) La aplicación debe hacer todo lo que el usuario le indique, dentro de las funcionalidades que la aplicación facilita.

6) La aplicación dispondrá de un fichero Excel con las instalaciones que se vayan guardando en ella.

2.6. Planificación temporal

La planificación para conocer cuáles son las tareas que se van a realizar a lo largo de la vida del proyecto, la duración que va a tener el proyecto y los días que va a trabajar el analista programador se va a calcular a continuación:

Lo primero es conocer cuáles van a ser las tareas que se van a realizar a lo largo de la vida del proyecto y son las siguientes:

- Búsqueda de la bibliografía - **A**
- Revisión de la bibliografía - **B**
- Estudio de la API de ILIAS - **C**
- Estudio de los servicios web de SOAP de ILIAS - **D**
- Diseño del storyboard de la interfaz de usuario - **E**
- Desarrollo de la interfaz de usuario - **F**
- Desarrollo de la gestión de las instalaciones - **G**
- Desarrollo del acceso a ILIAS - **H**
- Desarrollo de la gestión de los alumnos - **I**
- Uso de la gestión de los contenidos - **J**
- Desarrollo de funcionalidades suplementarias - **K**
- Realización de pruebas - **L**
- Completar la memoria del proyecto - **M**

A continuación se muestra el diagrama de PERT en el que se puede observar la duración de cada una de las tareas que se han ido realizando para la finalización del proyecto:

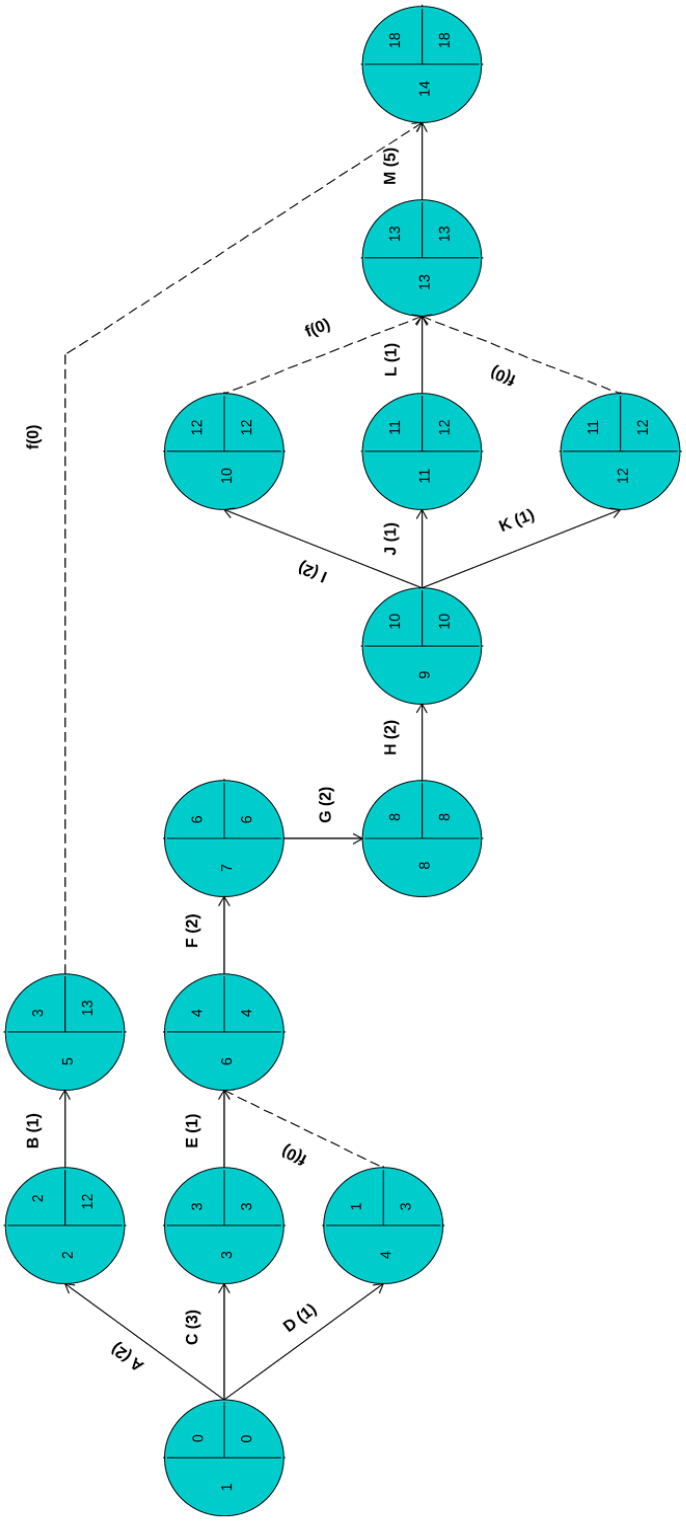


Imagen 2.6.1. Diagrama PERT.

Ahora se muestra el diagrama de Gantt para conocer la planificación del proyecto, así como la duración de las tareas:

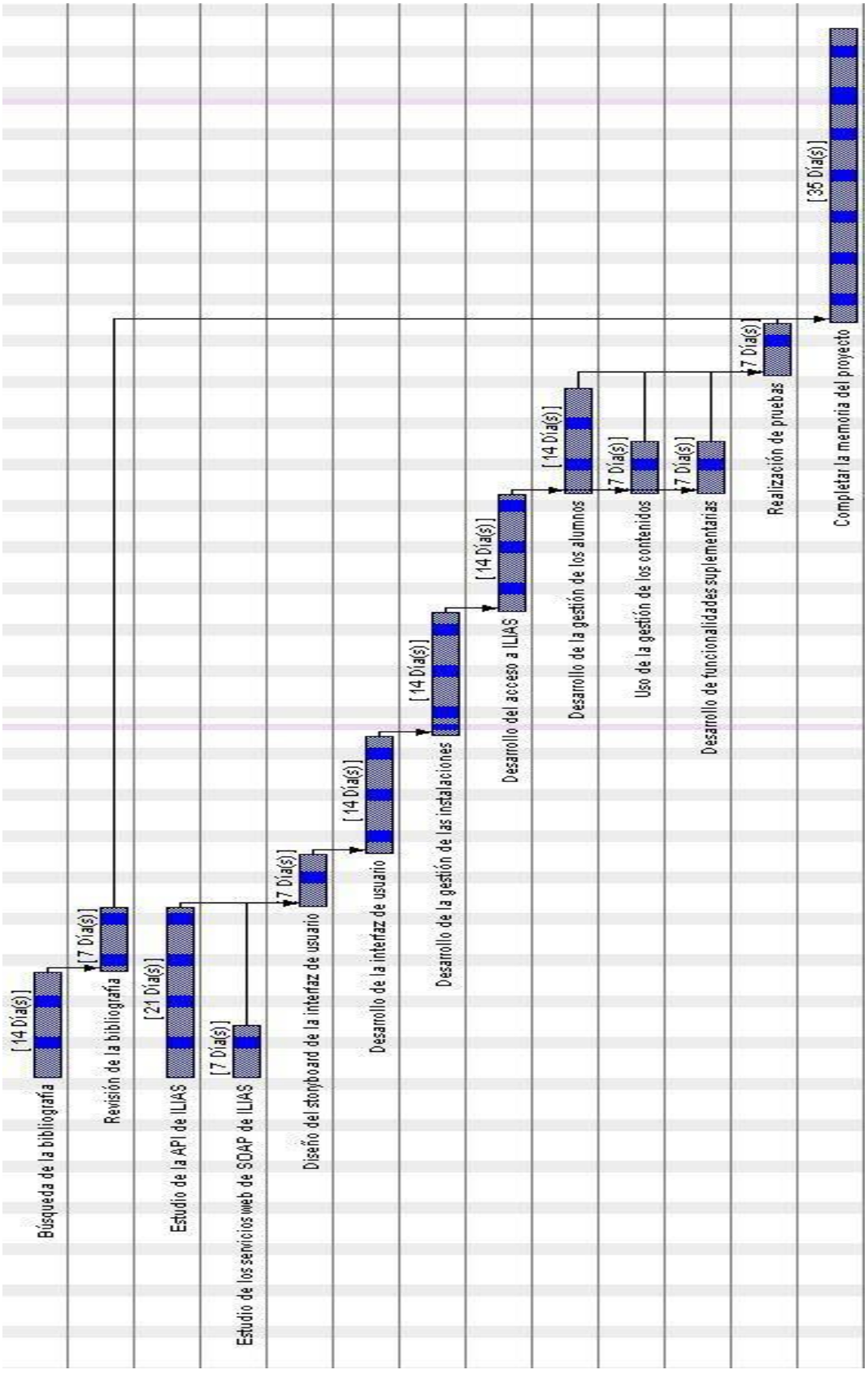


Imagen 2.6.2. Diagrama de Gantt.

Son días laborables

Son días festivos y de fin de semana

El comienzo del proyecto es el día 3 de marzo de 2014 y la finalización del proyecto es el día 27 de agosto de 2014.

Total de días de la vida del proyecto: 178 días.
Total de días laborables: 126 días.

2.7. Estudio de viabilidad

En este apartado se va a realizar un estudio del coste que supone la realización de este proyecto, el desarrollo que va a conllevar y la explotación en su caso.

Se muestra el presupuesto del analista programador en este proyecto, con los datos del convenio colectivo de Telefónica publicado en BOE para conocer su sueldo:

Analista programador: 2.327,19 euros al mes.
Días de trabajo del analista programador: 126 días (18 semanas).
Duración estimada de la jornada: 4 horas al día.

Tabla del presupuesto

Gastos	Precios
Gastos materiales (Hardware)	
Equipo para análisis y desarrollo	699 euros
Gastos materiales (Software)	
Acceso a Internet	40 euros / mes
Licencia de Netbeans	0 euros
Licencia de JavaFX Scene Builder	0 euros
Gastos de personal	
Analista programador	2.327,19 euros / mes
Total:	5356,6 euros

2.8. Casos de uso

Diagrama de casos de uso entre el usuario y la plataforma ILIAS

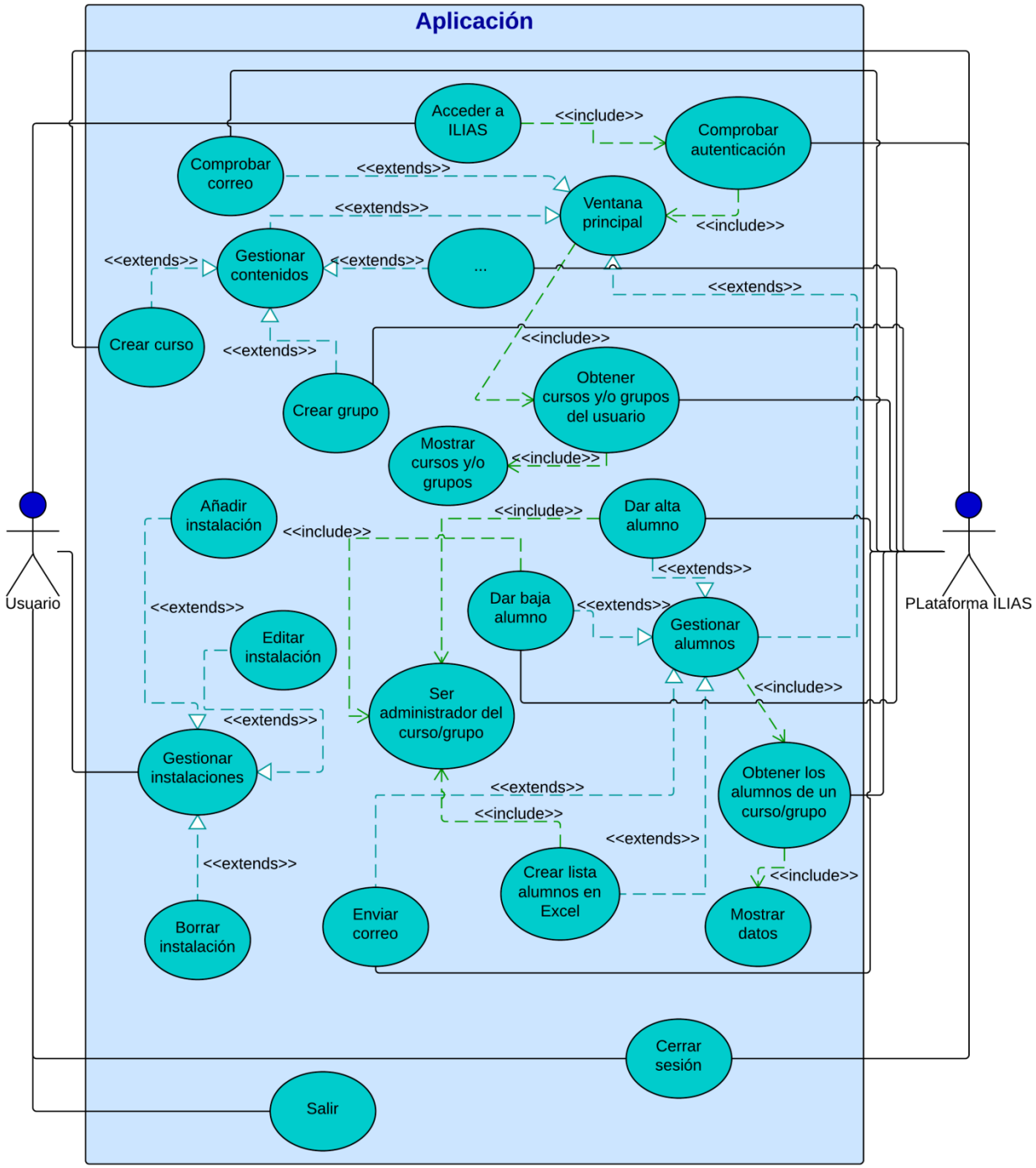


Imagen 2.8.1. Diagrama de casos de uso entre el usuario y la plataforma ILIAS.

En este diagrama de casos de uso se puede observar cómo interactúa el usuario con la aplicación para poder realizar cualquier funcionalidad que esté dentro de los objetivos de este proyecto. Para poder gestionar a los alumnos de un curso o grupo o gestionar los contenidos, el usuario debe de iniciar la sesión. En el caso de gestionar las instalaciones no es necesario iniciar la sesión con ILIAS [1]. Además, el usuario puede realizar cualquier otra función complementaria como es comprobar si tiene algún correo nuevo. Por último, se puede ver que la plataforma ILIAS [1] actúa cuando el usuario realiza alguna funcionalidad.

3. DISEÑO

En este apartado estudiaremos el diagrama de clases de este proyecto, las clases SOAP más importantes, algunos diagramas de secuencia para mejor comprensión de algunas funcionalidades de la aplicación. Además, se estudiará el diseño inicial de la interfaz de usuario.

3.1. Diagrama de clases

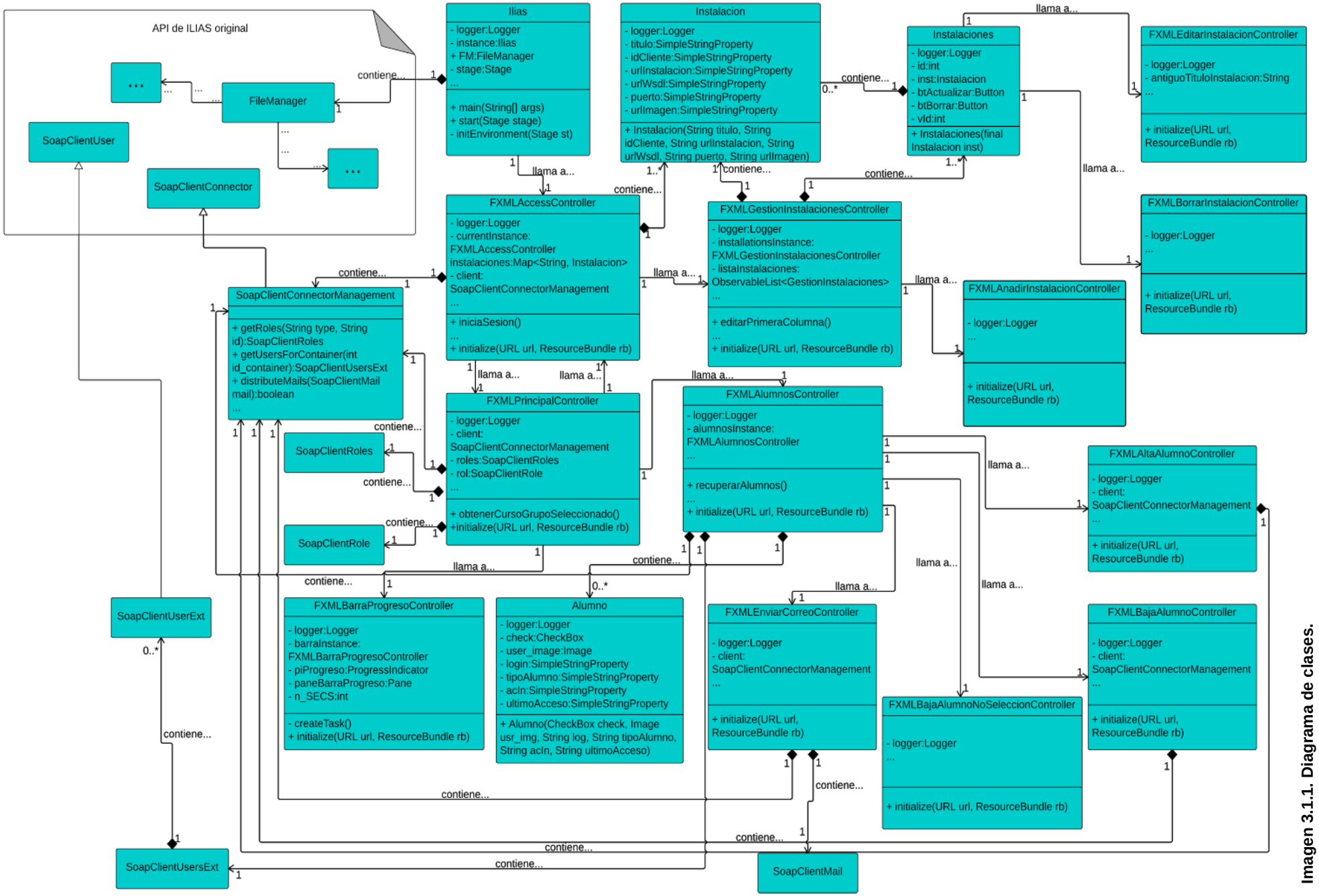


Imagen 3.1.1. Diagrama de clases.

En este diagrama de clases se muestra como están interconectadas todas las clases que forman parte de este proyecto entre sí, desde dónde se llaman a otros controladores para pasarle el control a ellos y también como se conecta al API [7] de ILIAS [1] para, por ejemplo, poder realizar la gestión de los contenidos desde el prototipo de este proyecto.

A continuación se describen las clases SOAP [2] más importantes en este proyecto que son:

- **SoapClientRoles**: Esta clase sirve para almacenar todos los roles (cursos, grupos, etc) que se obtengan llamando al servicio SOAP [2] getRoles().

- **SoapClientRole**: Esta clase almacena un role (curso, grupo, etc) que se ha obtenido llamando al servicio SOAP [2] getRoles().

- **SoapClientUsersExt**: Esta clase sirve para almacenar a los usuarios obtenidos a través del servicio SOAP [2] llamado getUsersForContainer().

- **SoapClientUserExt**: Esta clase almacena un usuario que se ha obtenido a través del servicio SOAP [2] llamado getUsersForContainer().

- **SoapClientMails**: Esta clase sirve para almacenar los correos que se envían a través del servicio SOAP [2] llamado distributeMails().

- **SoapClientMail**: Esta clase almacena un correo que se va a enviar a través del servicio SOAP [2] llamado distributeMails().

Cada una de estas clases contiene los atributos y elementos que necesiten para su correcto funcionamiento. Estos atributos y elementos pueden verse en la Tabla 2.1.1.2. Documentos DTD [21] con formatos de mensajes empleados.

3.2. Diagramas de secuencia

Diagrama de secuencia para acceder a ILIAS

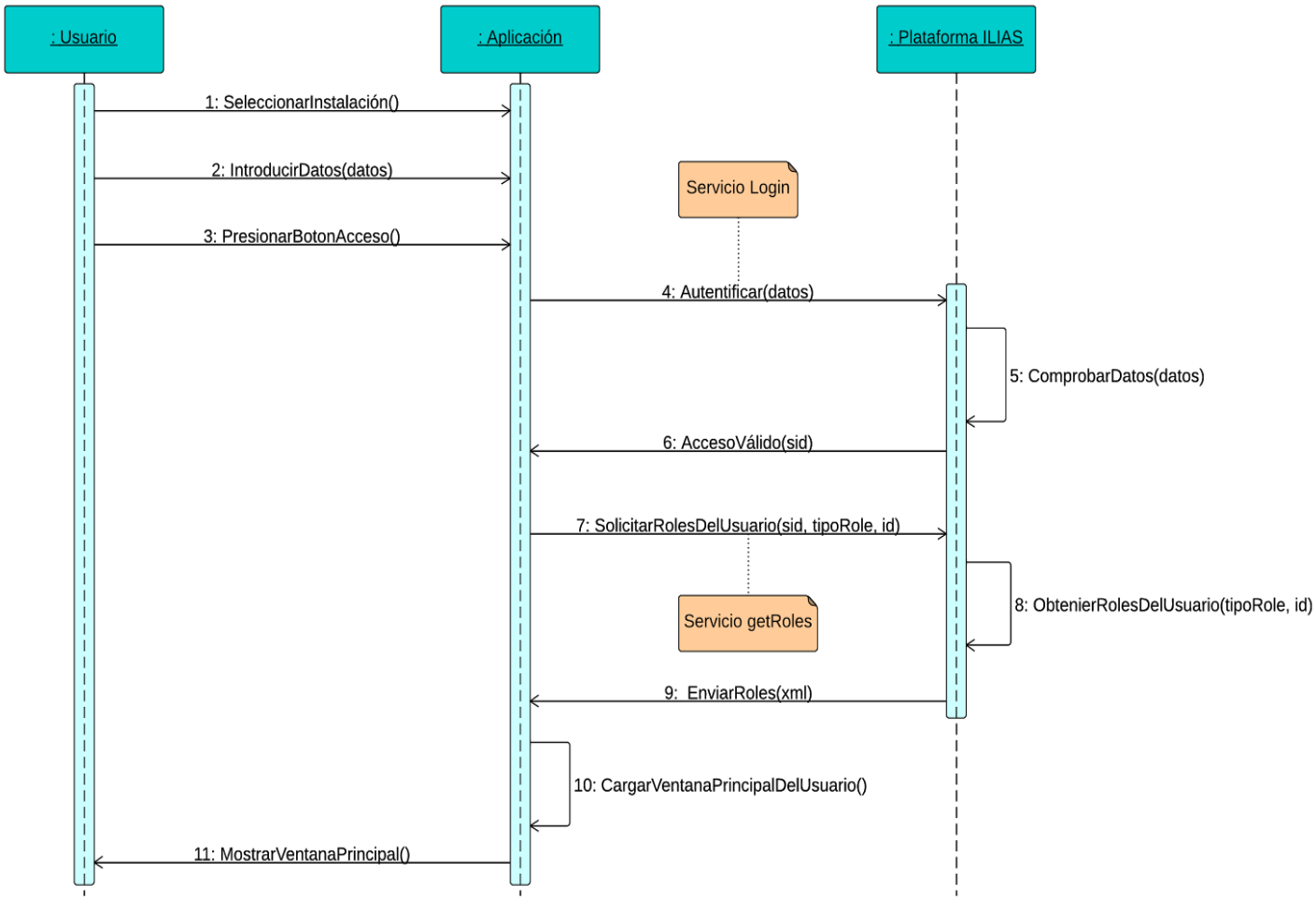


Imagen 3.2.1. Diagrama de secuencia para acceder a ILIAS.

En este diagrama de secuencia se muestran los pasos que se realizan para que un usuario pueda acceder a ILIAS [1]. En este caso se utilizan los servicios Login para loguearse y getRoles para mostrar al usuario que se ha conectado tanto los cursos y/o grupos a los que está matriculado.

Diagrama de secuencia para añadir un alumno

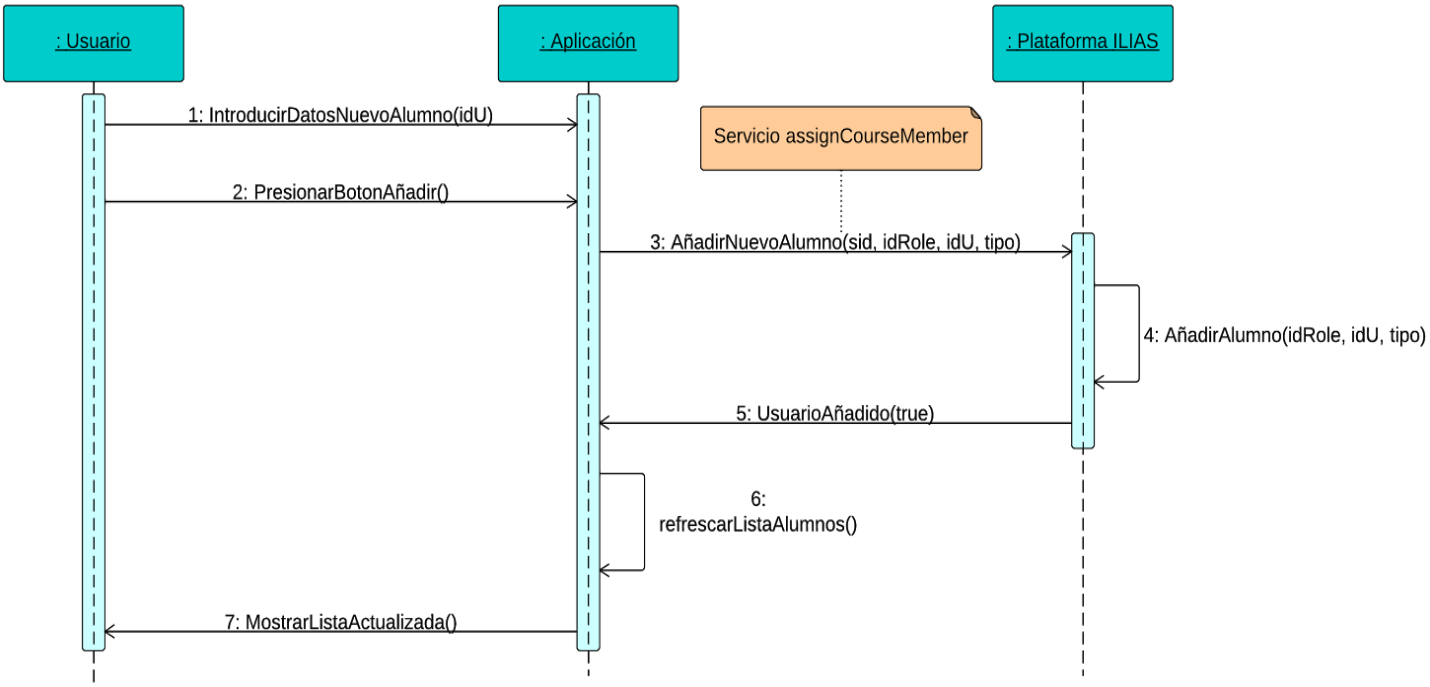


Imagen 3.2.2. Diagrama de secuencia para añadir un alumno.

En este diagrama de secuencia se muestran los pasos que se realizan para que un usuario añadir un nuevo alumno a un curso o grupo del que sea propietario. En este caso se utiliza el servicio assignCourseMember para asignar dicho alumno al curso o grupo en el que se desee añadir.

Diagrama de secuencia para editar una instalación

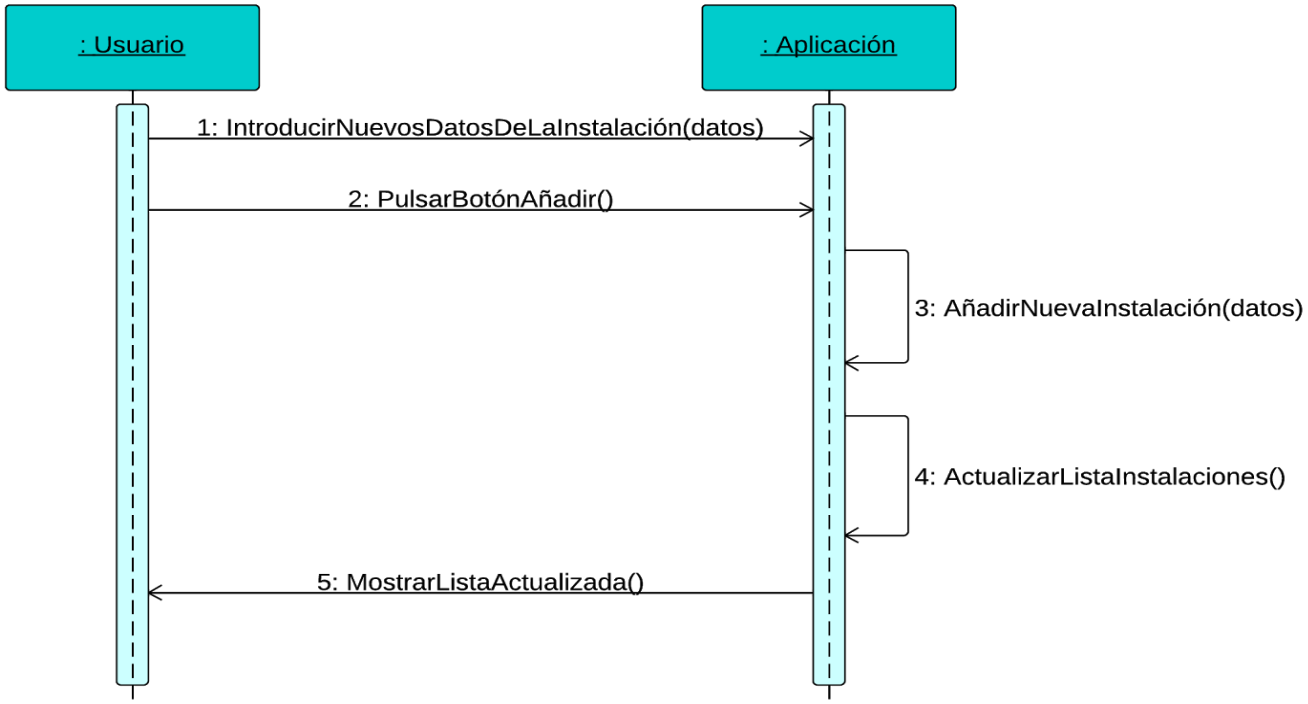


Imagen 3.2.3. Diagrama de secuencia para editar una instalación.

En este diagrama de secuencia se muestran los pasos que se realizan para que un usuario pueda editar una instalación. En este caso no es necesario el uso de un servicio SOAP [2] de ILIAS [1], ya que la gestión de las instalaciones se realiza de forma aislada a estos servicios.

Diagrama de secuencia para enviar un correo

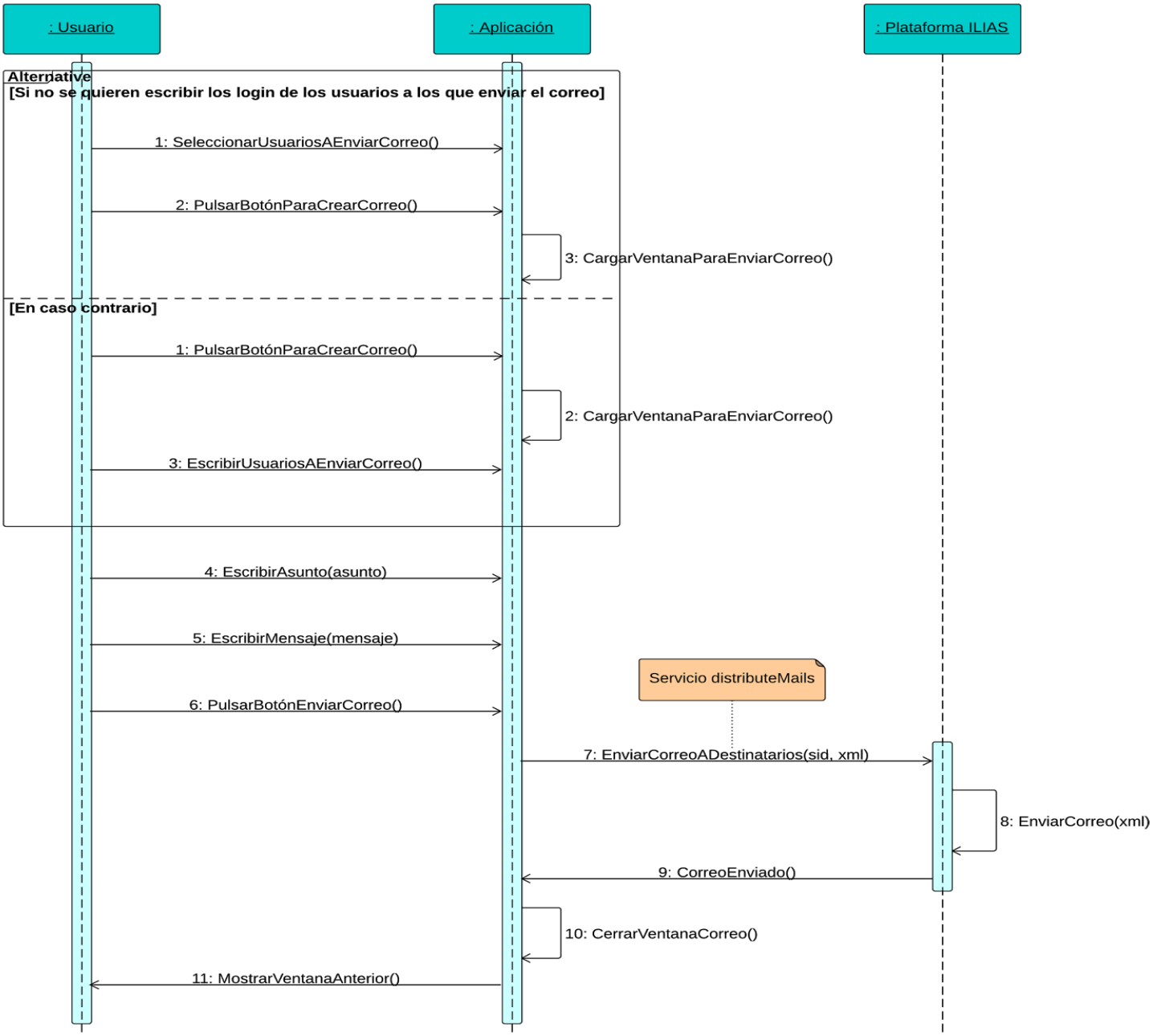


Imagen 3.2.4. Diagrama de secuencia para enviar un correo.

En este diagrama de secuencia se muestran los pasos que se realizan para que un usuario pueda enviar un correo a un usuarios o varios

usuarios de un curso o grupo al que esté matriculado. En este caso se utiliza el servicio distributeMails para el envío de correos a los distintos destinatarios.

Todos los diagramas de secuencia anteriores están realizados sin tener en cuenta los errores que se pudieran producir por la falta de algún dato o de cualquier otra cosa, solamente se muestran los pasos con los que funciona perfectamente.

3.3. Diseño de la interfaz

En este apartado se van a mostrar diferentes diseños de wireframe y storyboards que explicarán cuál es el diseño inicial que se hizo para llevar una idea de cómo va a quedar el prototipo.

Al iniciar la aplicación se muestra la siguiente ventana con la que se accede a ILIAS [2]:

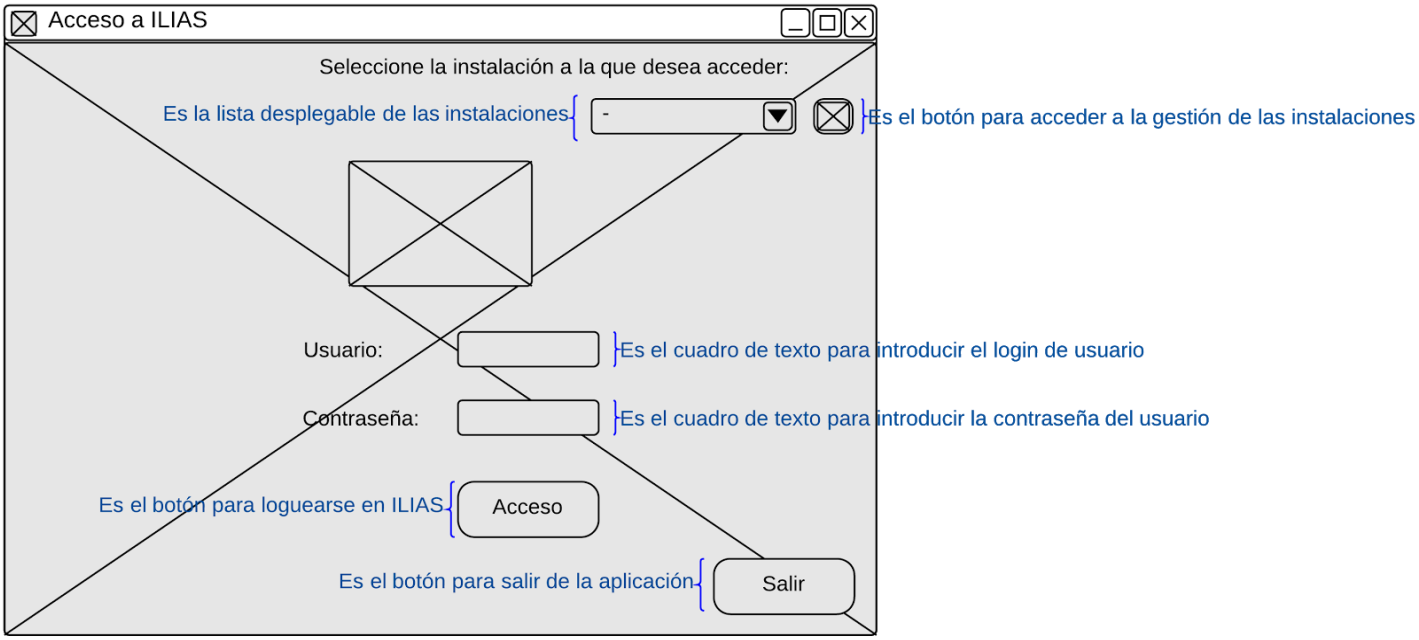


Imagen 3.3.1. Ventana de acceso a ILIAS.

Para acceder a gestionar los contenidos, se pulsa sobre el botón indicado para ello y en la siguiente ventana ya se pueden gestionar los contenidos:

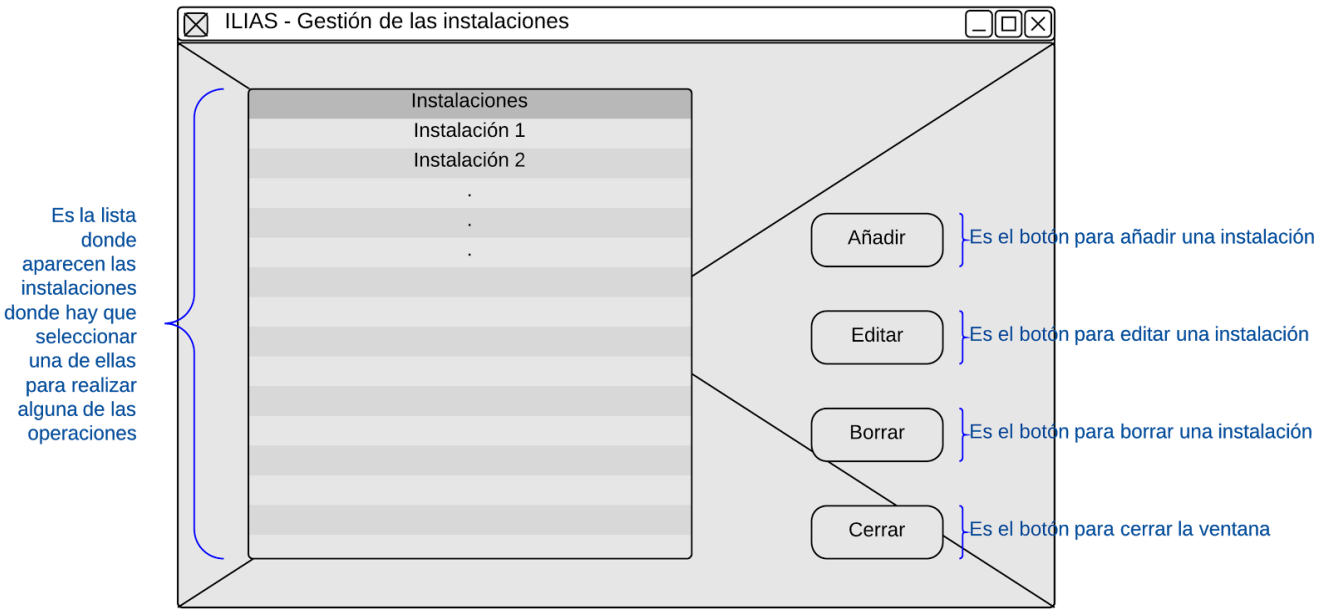


Imagen 3.3.2. Ventana para gestionar las instalaciones.

Gestionando las instalaciones se puede añadir una nueva instalación, editar una instalación o borrar una instalación. A continuación se muestran los diseños para ello.

El diseño para añadir o editar una instalación es el mismo:

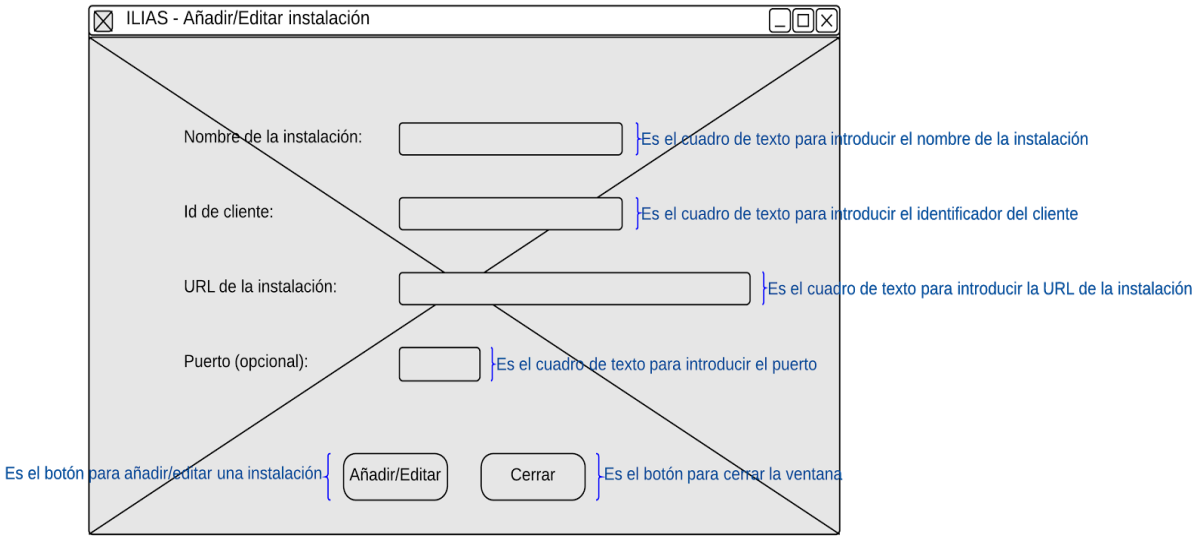


Imagen 3.3.3. Ventana añadir o editar una instalación.

El diseño para borrar una instalación es una ventana de confirmación y es la siguiente:

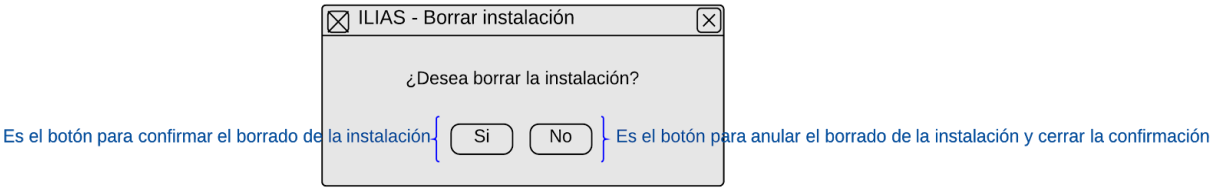


Imagen 3.3.4. Ventana de confirmación para borrar una instalación.

A continuación se muestra el diseño de la interfaz de la ventana una vez se ha accedido a ILIAS:

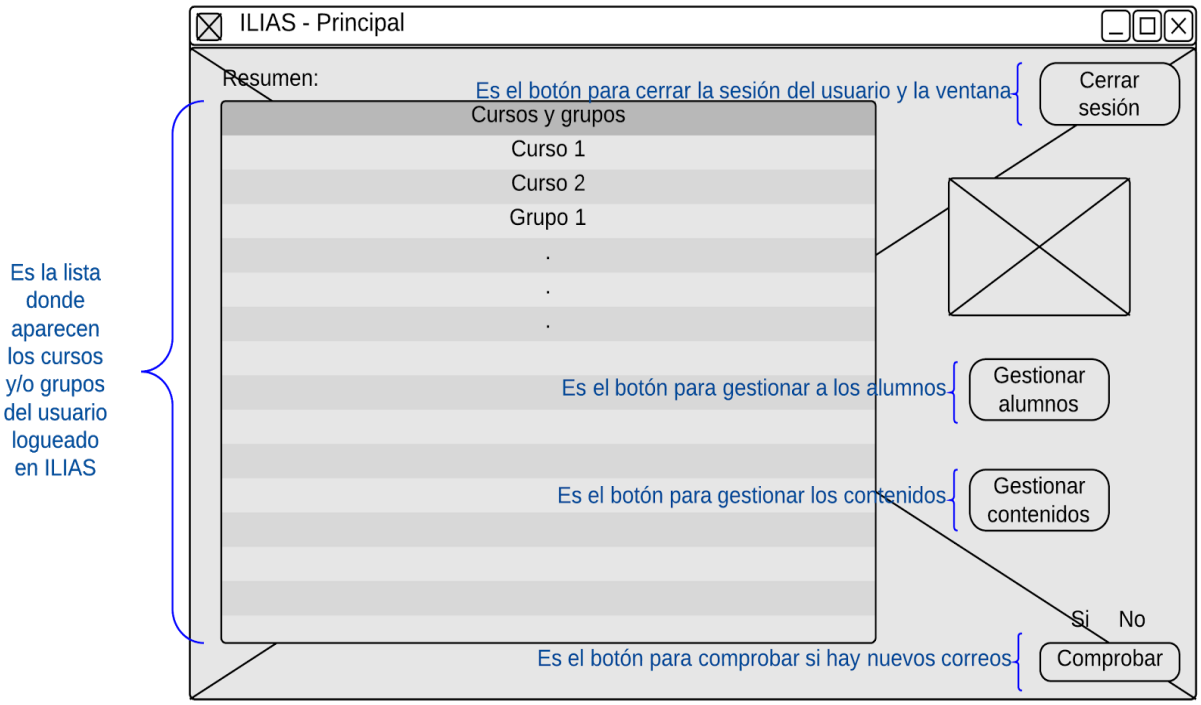


Imagen 3.3.5. Ventana principal.

Para poder gestionar tanto a los usuarios como los contenidos, debe seleccionarse un curso o grupo de la lista.

Al acceder a la ventana para gestionar los contenidos aparece la misma ventana que se muestra cuando se accede desde la web.

El diseño de la ventana para gestionar a los alumnos es el siguiente:

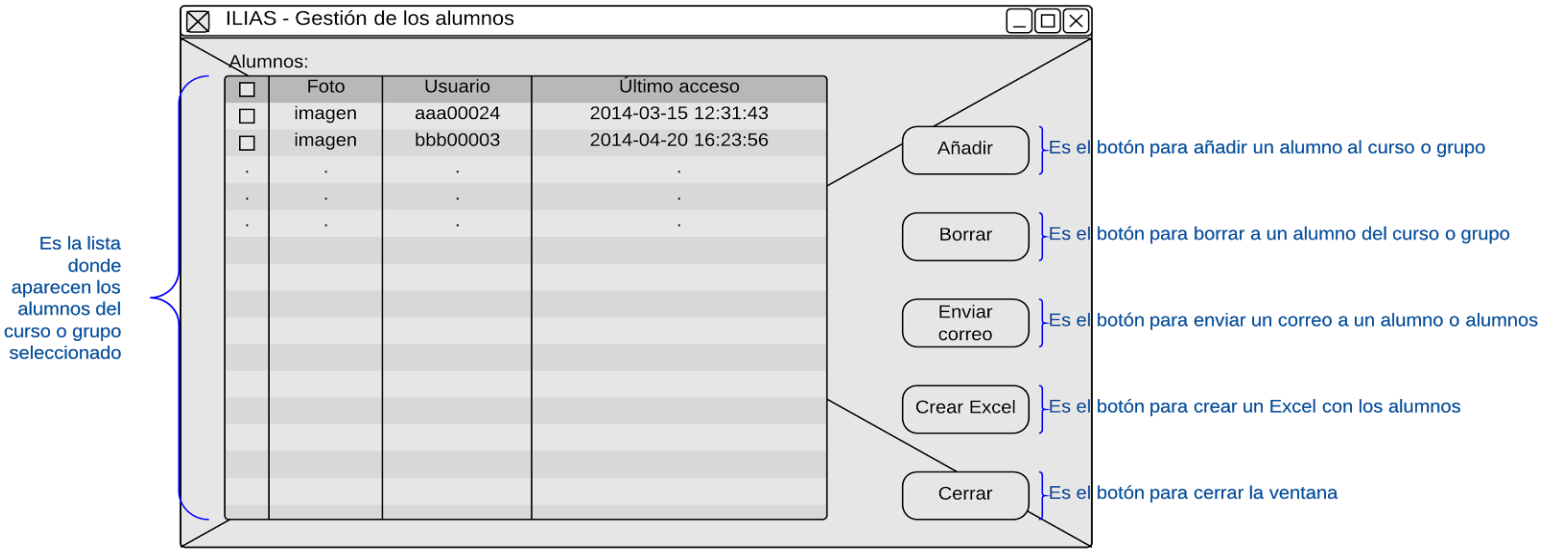


Imagen 3.3.6. Ventana de los alumnos de un curso o grupo.

El diseño de la ventana para añadir a un alumno es el siguiente:

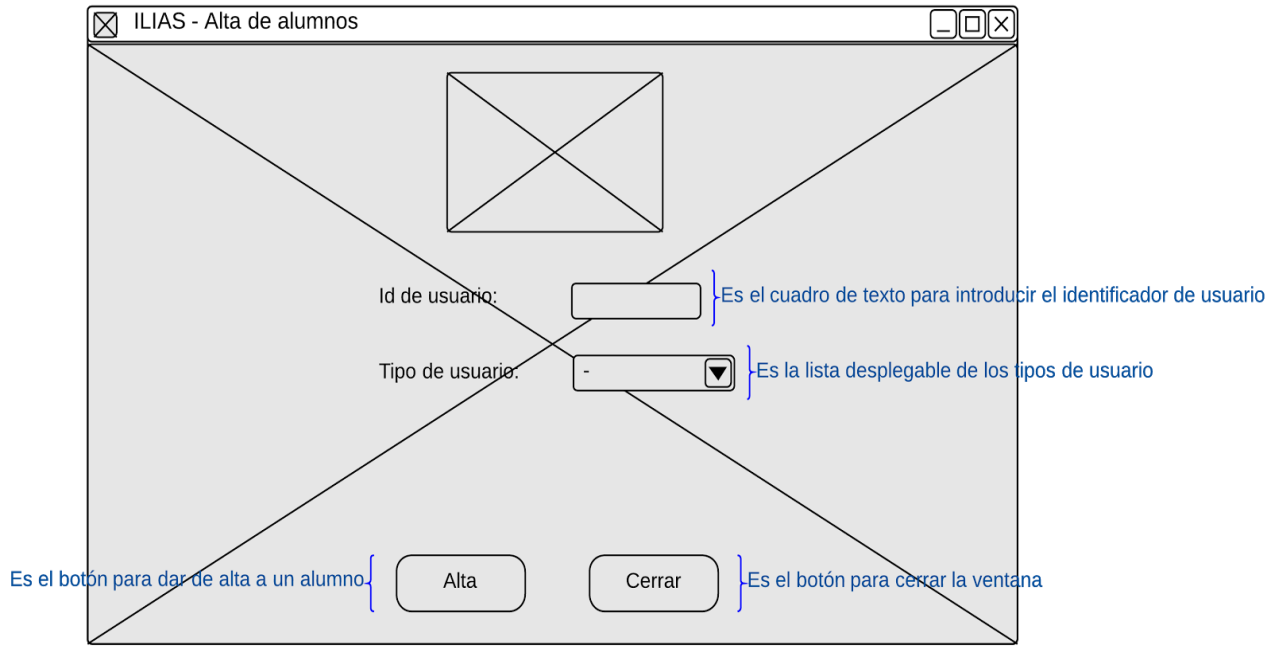


Imagen 3.3.7. Ventana para dar de alta a un alumno en un curso o grupo.

El diseño de la ventana para borrar a un alumno es el siguiente:

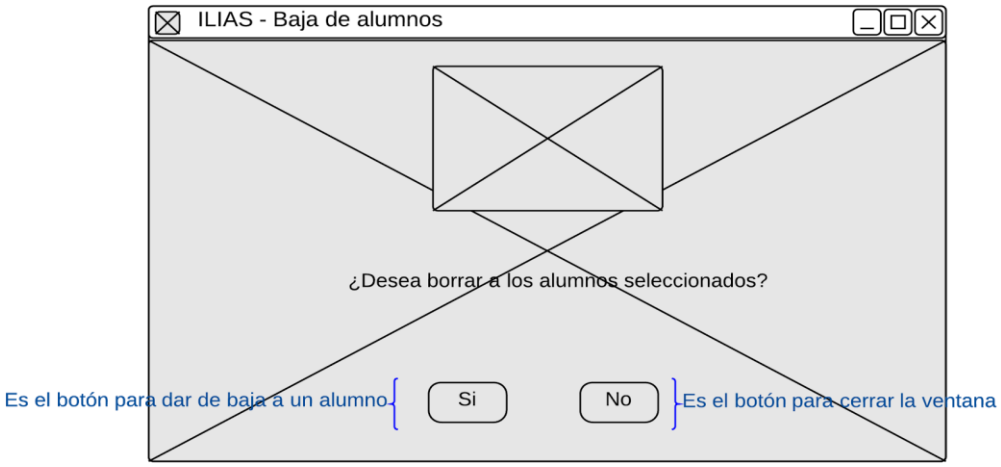


Imagen 3.3.8. Ventana para dar de baja a uno o varios alumnos de un curso o grupo.

El diseño de la ventana para enviar un correo es el siguiente:

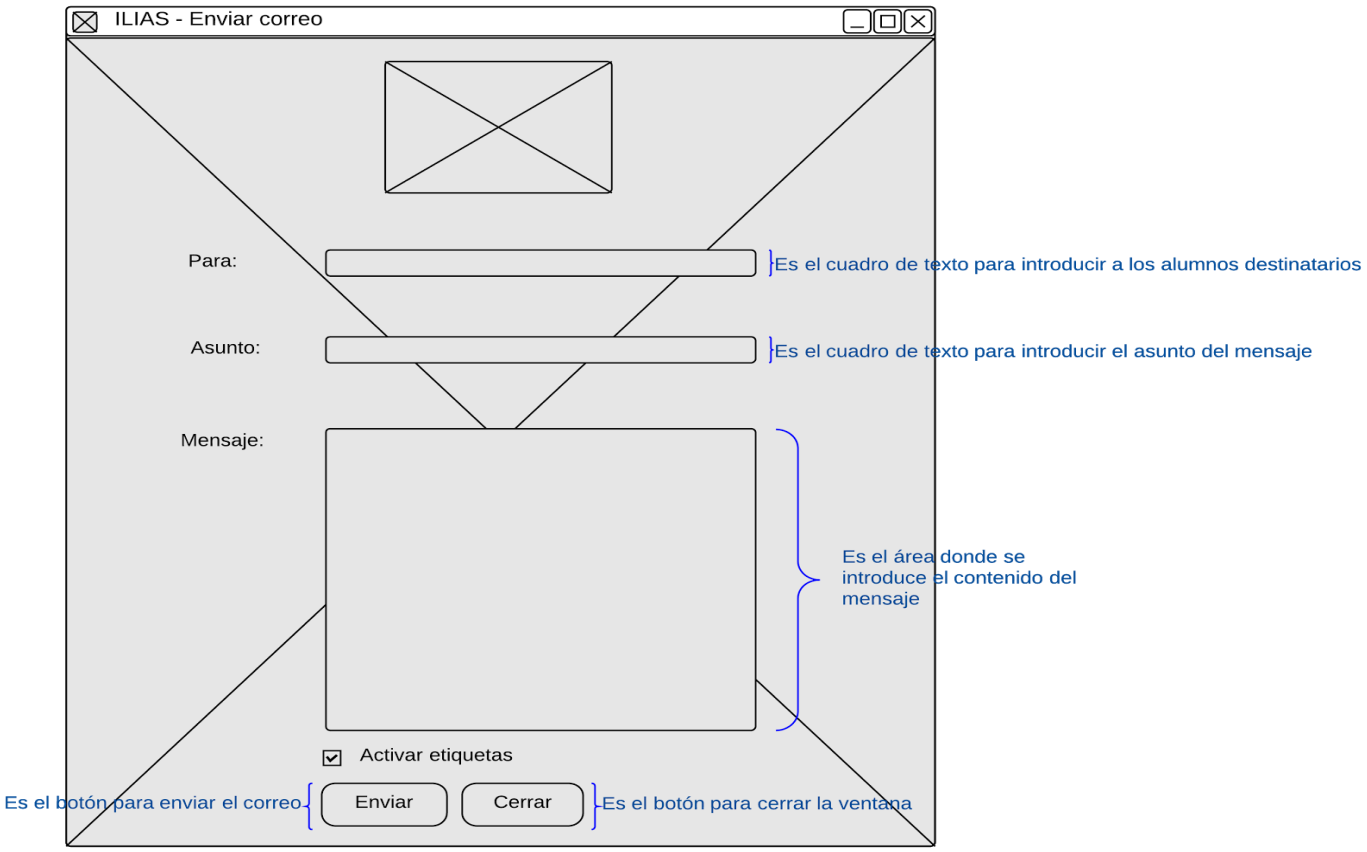


Imagen 3.3.9. Ventana para enviar un correo a los alumnos de un curso o grupo.

El diseño de la ventana para crear un Excel con la lista de los alumnos del curso o grupo seleccionado es básicamente el mismo diseño de la ventana para guardar un archivo.

4. IMPLEMENTACIÓN

En este apartado estudiaremos la arquitectura de la aplicación, el proceso de desarrollo que se ha llevado con JavaFX [3] y algunos detalles sobre implementación para saber todo lo que ha sido necesario para el desarrollo de este proyecto.

4.1. Arquitectura

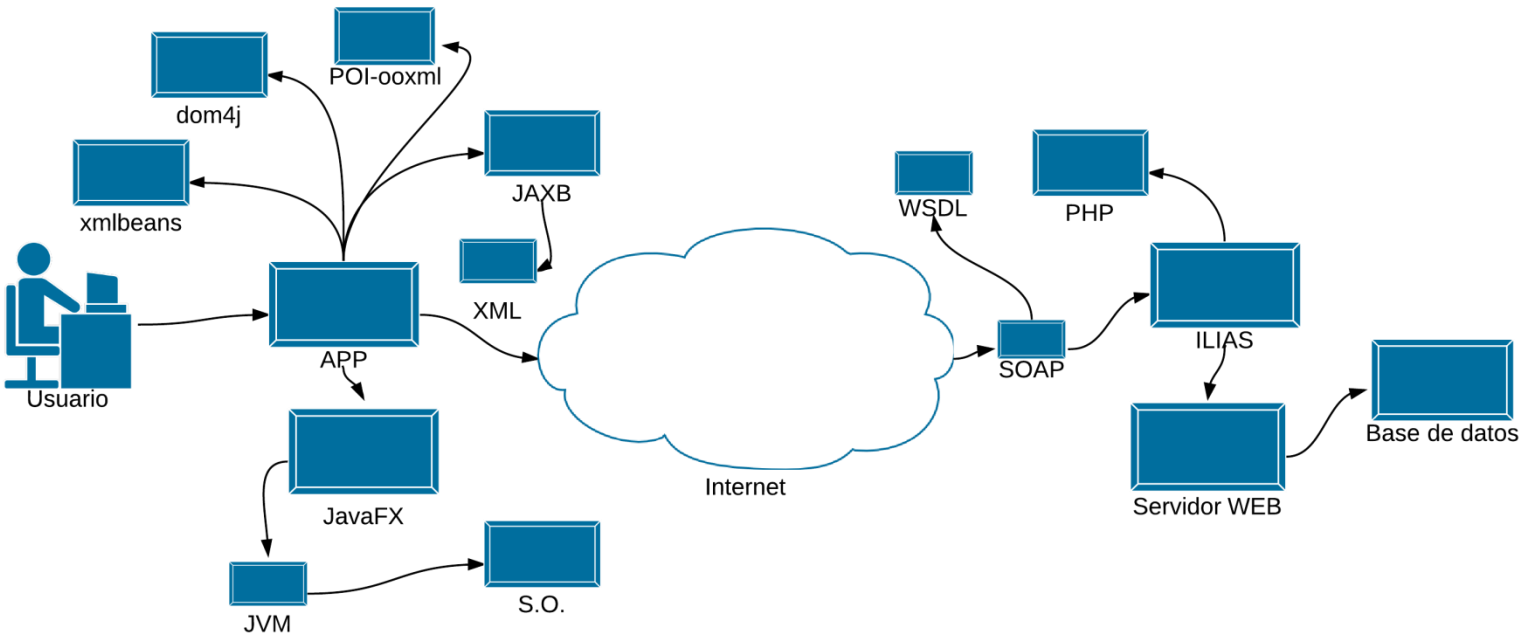


Imagen 4.1.1. Arquitectura de la aplicación.

Esta es la arquitectura que tiene esta aplicación y en ella se puede observar cómo está todo interconectado para su funcionamiento. En ella se ve que el usuario interactúa con la aplicación y ésta ya va llamando a unos y otros para poder realizar bien el funcionamiento de dicha aplicación. En este caso, se ve que la aplicación está desarrollada en JavaFX [3] y se ejecuta en una máquina virtual de Java (JVM). También utiliza JAXB [6] para la comunicación entre SOAP [2] y JAVA [5]; y contiene las librerías llamadas POI-ooxml, dom4j y xmlbeans para la realización del archivo Excel. A través de la aplicación se llaman a los distintos servicios SOAP [2] que permite usar la aplicación y con los que se obtiene el resultado que el usuario espera.

4.2. El proceso de desarrollo con JavaFX

Se ha comenzado la implementación de esta aplicación descargando el API [7] de ILIAS [1] 4.3.6 [36]

El entorno de desarrollo utilizado para este proyecto ha sido Netbeans y se ha utilizado la versión 7.4.

Se han descargado las imágenes que se han utilizado para el diseño de cada una de las ventanas de la aplicación [37]

Para una mayor facilidad en la creación de la aplicación se ha utilizado un programa llamado JavaFX Scene Builder con el que se ha ido creando cada una de las ventanas de la aplicación, sus estilos, etc. Este programa va enlazado con Netbeans para una comunicación y programación rápida.

A continuación se puede observar el programa JavaFX Scene Builder:

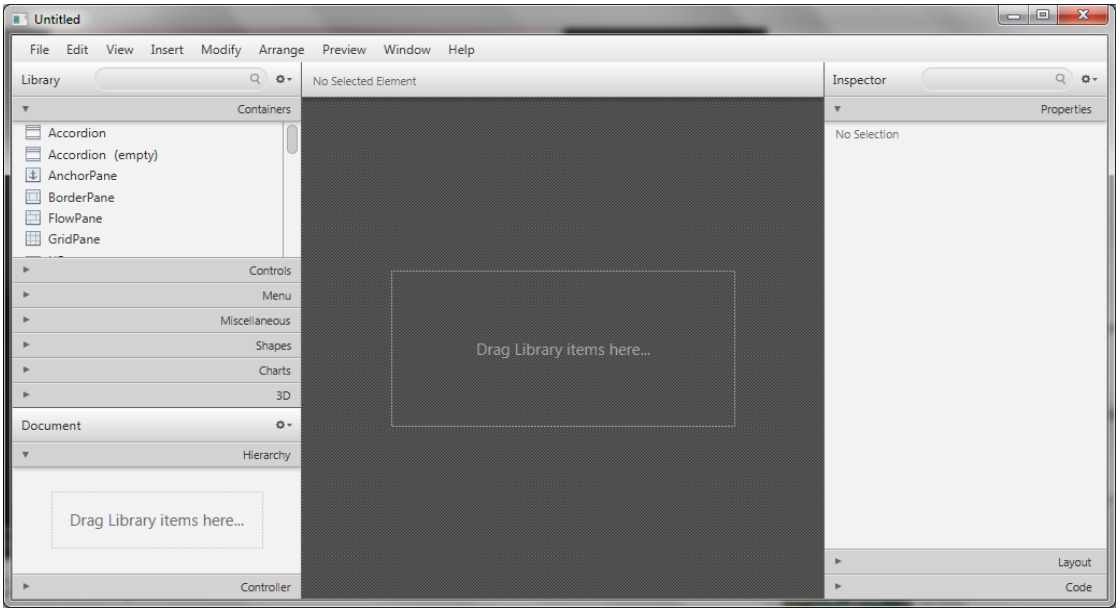


Imagen 4.2.1. Ventana del programa JavaFX Scene Builder.

Este es el programa cuando se inicia en el que se pueden observan que tanto a la izquierda como a la derecha tiene herramientas y otras utilidades con las que se realiza el diseño de cada una de las ventanas de una aplicación.

A continuación se puede observar como se ha estado diseñando una de las ventanas de esta aplicación:

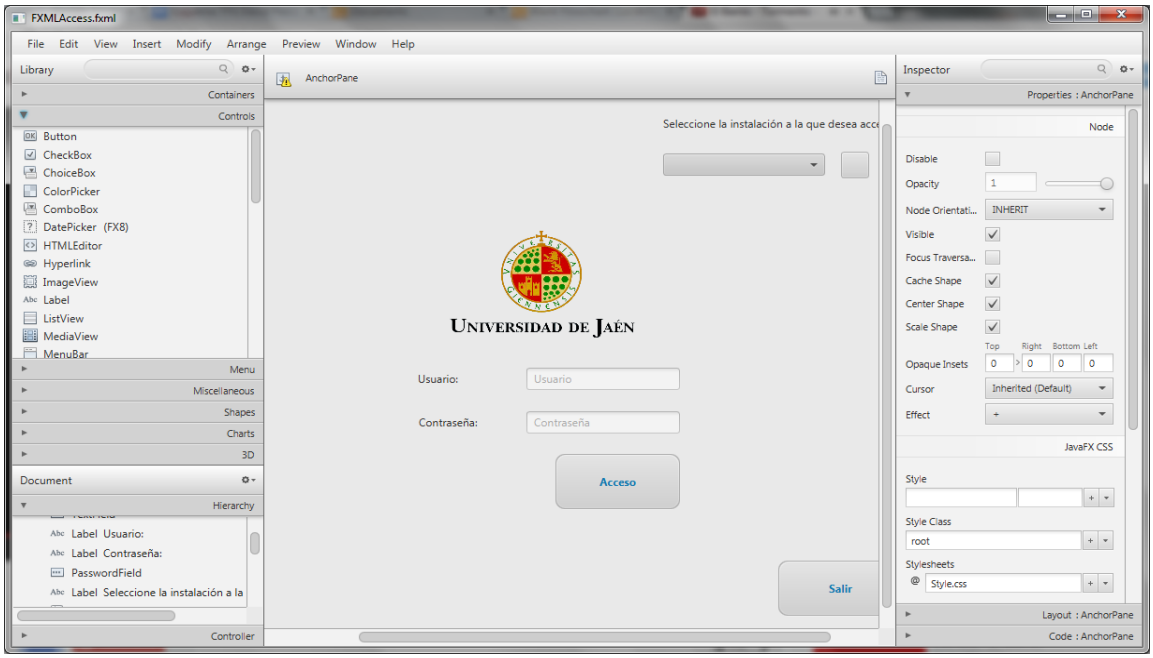


Imagen 4.2.2. Ejemplo de diseño de la ventana de acceso a ILIAS.

4.3. Detalles sobre implementación

Para poder ir comprobando que lo que se ha ido implementado funciona correctamente, se ha necesitado el java 8.0 instalado y colocado en el PATH del sistema operativo.

Por otro lado, las bibliotecas que se han utilizado para este proyecto son las siguientes:

- POI-ooxml - poi-ooxml-3.10-FINAL-20140208.jar
- POI-ooxml - poi-ooxml-schemas-3.10-FINAL-20140208.jar
- POI-ooxml - poi-3.10-FINAL-20140208.jar
- POI-ooxml - poi-examples-3.10-FINAL-20140208.jar
- POI-ooxml - poi-excelant-3.10-FINAL-20140208.jar
- POI-ooxml - poi-scratchpad-3.10-FINAL-20140208.jar
- dom4j - dom4j-1.6.1.jar
- xmlbeans - jsr173_1.0_api.jar
- xmlbeans - resolver.jar
- xmlbeans - xbean.jar
- xmlbeans - xbean_xpath.jar

- xmlbeans - xmlbeans-qname.jar
- xmlbeans - xmlpublic.jar

Estas bibliotecas han sido utilizadas para poder guardar la lista de alumnos en un archivo Excel.

Se ha llegado a extender el API [7] original de ILIAS [1] tanto para poder adaptarlo a la aplicación realizada en este proyecto y también para poder realizar diferentes funcionalidades desde dicha aplicación. Las clases que han extendido el API [7] de ILIAS [1] son las siguientes:

- **SoapClientAssignCourseMemberRequest.java**: Esta clase realiza la petición para asignar a un usuario a un rol específico (curso, grupo, etc).

- **SoapClientAssignCourseMemberResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, si se ha asignado el usuario con éxito en un curso, grupo, etc..

- **SoapClientAssignedObject.java**: Esta clase contiene datos de un curso, grupo, etc.

- **SoapClientAuthMode.java**: Esta clase contiene el modo en que un usuario se autentifica.

- **SoapClientDistributeMailsRequest.java**: Esta clase realiza la petición para enviar un correo a un usuario o a varios usuarios.

- **SoapClientDistributeMailsResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, si el correo ha sido enviado con éxito.

- **SoapClientElement.java**: Esta clase contiene los elementos del path de un curso, grupo, etc.

- **SoapClientExcludeCourseMemberRequest.java**: Esta clase realiza la petición para excluir o borrar a un usuario de un curso, grupo, etc.

- **SoapClientExcludeCourseMemberResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, si el usuario ha sido excluido correctamente del curso, grupo, etc.

- **SoapClientGetCourseXMLRequest.java**: Esta clase realiza la petición para obtener la descripción de un curso específico.

- **SoapClientGetCourseXMLResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, se obtiene un XML [13] con la descripción de dicho curso.

- **SoapClientGetCoursesForUserRequest.java**: Esta clase realiza la petición para obtener los cursos a los que está matriculado un usuario.

- **SoapClientGetCoursesForUserResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, los cursos en formato XML [13] a los que está matriculado un usuario.

- **SoapClientGetRolesRequest.java**: Esta clase realiza la petición para obtener los roles (cursos, grupos, etc) a los que está matriculado un usuario.

- **SoapClientGetRolesResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, los cursos, grupos, etc a los que está matriculado un usuario.

- **SoapClientGetTestResultsRequest.java**: Esta clase realiza la petición para obtener los datos de los resultados de un test.

- **SoapClientGetTestResultsResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, los resultados de un test.

- **SoapClientGetTestUserDataRequest.java**: Esta clase realiza la petición para obtener los datos del usuario que ha realizado el test.

- **SoapClientGetTestUserDataResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, los datos del usuario que ha realizado el test.

- **SoapClientGetUserIdBySidRequest.java**: Esta clase realiza la petición para obtener el identificador del usuario a través de su Sid.

- **SoapClientGetUserIdBySidResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, el identificador del usuario.

- **SoapClientGetUserXMLRequest.java**: Esta clase realiza la petición para obtener la descripción de un usuario.

- **SoapClientGetUserXMLResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, la descripción de un usuario.

- **SoapClientGetUsersForContainerRequest.java**: Esta clase realiza la petición para obtener a los usuarios que están matriculados en algún rol (curso, grupo, etc).

- **SoapClientGetUsersForContainerResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, a los usuarios que están matriculados en algún rol (curso, grupo, etc).

- **SoapClientHasNewMail.java**: Esta clase realiza la petición para comprobar si hay nuevos correos en el correo de un usuario de ILIAS [1].

- **SoapClientHasNewMailResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, si un usuario tiene nuevos correos o no.

- **SoapClientLogoutRequest.java**: Esta clase realiza la petición para aplicar un log out y así cerrar la sesión del usuario.

- **SoapClientLogoutResponse.java**: Esta clase obtiene la respuesta a la petición realizada en la clase anterior, en este caso, si la sesión del usuario se ha cerrado correctamente o no.

- **SoapClientMail.java**: Esta clase contiene un correo que va a ser enviado a uno o varios usuarios de un curso, grupo, etc.

- **SoapClientMails.java**: Esta clase contiene el conjunto de correos que van a ser enviados a un usuario o a varios usuarios.

- **SoapClientPath.java**: Esta clase contiene los elementos del PATH de un curso, grupo, etc.

- **SoapClientRole.java**: Esta clase contiene la información de un role (curso, grupo, etc).

- **SoapClientRoles.java**: Esta clase contiene a un conjunto de roles (cursos, grupos, etc).

- **SoapClientUserExt.java**: Esta es una clase que hereda de la clase SoapClientUser.java y que contiene mucha más información de un usuario que la clase de la que hereda.

:

- **SoapClientUserRole.java**: Esta clase contiene el role que realiza cada uno de los usuarios.

- **SoapClientUsersExt.java**: Esta clase contiene las instancias de la clase SoapClientUserExt.java.

A parte de estas clases anteriores, también se ha añadido otra clase para extender la clase SoapClientConnector.java llamada SoapClientConnectorManagement.java con la que se realizan las peticiones de los servicios que solicita el usuario a través de la aplicación y por donde se devuelve el resultado a dicha petición.

Todos los servicios SOAP [2] utilizados han sido estudiados en la siguiente página:
http://dv.ujaen.es/docencia/webservice/soap/server.php#

5. CONCLUSIONES

Se ha llegado a cumplir el nivel de cumplimiento de los objetivos que se han indicado inicialmente para realizar este proyecto, donde se ha estudiado la interfaz WEB de ILIAS [1] para comprobar su estructura y comportamiento. También se ha realizado el prototipo de la aplicación con todas las funcionalidades que se han podido realizar para este proyecto y ha sido desarrollado en JavaFX [3].

Al usar el programa llamado JavaFX Scene Builder he podido ahorrar bastante tiempo en la realización del diseño de la interfaz del usuario, por lo que he podido dedicar tiempo a otras cosas más importantes de este proyecto.

El enfoque que se ha propuesto para este proyecto me ha servido bastante para llevar un ritmo de trabajo muy bueno y sencillo, ya que al utilizar una metodología SCRUM se ha podido seguir un buen proceso de trabajo y se ha ido dividiendo el trabajo en periodos de tiempo para que sea más llevadero.

En mi opinión, lo más interesante que me ha resultado en este proyecto ha sido el conectar la aplicación con los servicios SOAP [2] para poder realizar las funcionalidades deseadas y comprender cómo se interconectan unas cosas con otras para su funcionamiento. Para mí estos son los aspectos más importantes de este proyecto.

Este proyecto me ha servido para saber mejor como trabajar dentro de un límite de tiempo y ser más eficiente y organizado a la hora de ir realizando un proyecto. Todo esto ha sido posible gracias a los conocimientos adquiridos durante el grado. Ha sido una gran experiencia al poder realizar un gran proyecto como este.

También he comprobado que al gestionar los contenidos hay funcionalidades que no funcionan correctamente en ninguna de las instalaciones, por lo que estaría perfecto si todas las funcionalidades que se pueden realizar al gestionar los contenidos funcionen correctamente para todas las instalaciones.

Me he encontrado con problemas en los permisos a la hora de usar ciertos servicios SOAP [2], ya que para usar algunos de estos servicios hay que ser administrador. Sería necesario replantear la implementación de los métodos SOAP [2] de ILIAS [1] para que permitieran a cualquier usuario obtener con este API [7] la misma funcionalidad sin tener que recurrir a utilizar usuarios con rol de

administrador. Por otro lado, parece haber movimiento en la comunidad de desarrollo de introducir una nueva API [7] basada en REST [38].

5.1. Mejoras y trabajos futuros

Por un lado se podrían realizar algunas mejoras en este proyecto como son:

- Añadir al prototipo una ventana de presentación de la aplicación. Para ello como ayuda podría usarse el programa JavaFX Scene Builder.

- Acceder al correo de un usuario a través de la aplicación. Para ello se podría añadir un nuevo botón para mostrar una ventana con los correos del usuario.

- Modificar el código del método que devuelve los datos de un grupo (alumnos, etc) para que puedan ser mostrados al pulsar sobre uno de ellos.

- Mejorar el indicador de proceso al estar cargándose la ventana con los alumnos de un curso o grupo.

- Almacenar en el archivo Excel el nombre y los apellidos de cada usuario además de la foto personal y el login de cada usuario.

Por otro lado, se podrían realizar mejoras en la implementación de servicios SOAP y/o añadir nuevos métodos:

- Obtener el SID del usuario a través de su identificador.

- Poder comprobar si hay nuevo correo en otras instalaciones a parte de la instalación demo.

- Obtener las fotos personales de los usuarios de cualquier instalación.

- Devolver el nombre y los apellidos de un usuario cuando se devuelven sus datos. Para ello habría que modificar el servicio llamado getUsersForContainer().

- Añadir un alumno a un curso o grupo a través de su nombre, apellidos y DNI. Para ello habría que modificar el servicio llamado assignCourseMember().

Para poder realizar mejoras en la implementación de servicios SOAP [2] y/o añadir nuevos métodos, hay que modificar la implementación de los servicios SOAP [2] de ILIAS [1].

6. BIBLIOGRAFÍA

[1] María del Mar Ramos Tejada (Abril, 2010) *La plataforma ILIAS como apoyo a la docencia presencial en Ingeniería Técnica Industrial*. Revista. Recuperado en Marzo de 2014 de http://revista.inie.ucr.ac.cr/uploads/tx_magazine/ilias.pdf

[2] *Simple Object Access Protocol* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol

[3] James L. Weaver, Weiqi Gao, Ph.D., Stephen Chin and Dean Iverson, with Johan Vos, Ph.D. (2012). *Pro JavaFX 2: A definitive Guide to Rich Clients with Java Technology*. New York. Apress.

[4] *ILIAS* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/ILIAS

[5] *Java (lenguaje de programación)* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[6] *Tutorial JAXB* (n.d.) Xarw. Recuperado en Marzo de 2014 de http://biblioteca.xarw.net/uploads/TUTORIAL JAXB.doc

[7] *Interfaz de programación de aplicaciones* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[8] Edgar Ramirez (Octubre, 2008) *SOAP*. Wordpress. Recuperado en Marzo de 2014 de http://edgarramirez.wordpress.com/2008/10/14/soa/

[9] *Hypertext Transfer Protocol* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[10] *Simple Mail Transfer Protocol* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[11] *Java Message Service* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Java_Message_Service

[12] *Transmission_Control_Protocol* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Transmission_Control_Protocol

[13] *Extensible Markup Language* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Extensible_Markup_Language

[14] John Fernando Vergara Arroyo (2005) *Protocolo simple de acceso a objetos (SOAP)*. Monografías. Recuperado en Marzo de 2014 de http://www.monografias.com/trabajos29/protocolo-acceso/protocolo-acceso.shtml

[15] *WSDL* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/WSDL

[16] *Web Services Description Language* (n.d.) Mercurio. Recuperado en Marzo de 2014 de http://mercurio.ugr.es/pedro/tutoriales/cursos/curso_soap/wsdl.htm

[17] *Web Service: Definición, utilización y estructura del WSDL* (2012) Apuntes de programación. Recuperado en Marzo de 2014 de http://programacion.jias.es/2012/01/web-service-definicion-utilizacion-estructura-del-wsdl

[18] Marco Besteiro y Miguel Rodríguez (n.d.) *Web Services*. ehu. Recuperado en Marzo de 2014 de http://www.ehu.es/mrodriguez/archivos/csharppdf/ServiciosWeb/WebServices.pdf

[19] *WSDL* (Noviembre, 2012) ClubEnsayos. Recuperado en Marzo de 2014 de http://clubensayos.com/Tecnolog%C3%ADa/WSDL/401635.html

[20] Benjamín Zepeda (Mayo, 2009) *¿Qué es SOAP?*. Probando código. Recuperado en Marzo de 2014 de http://www.probandocodigo.com/2009/05/que-es-soap.html

[21] Bartolomé Sintes Marco (Abril, 2014) *DTD: Definición de Tipo de Documento*. Mclibre. Recuperado en Marzo de 2014 de http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html

[22] David Polo Gómez (2013). *Desarrollo de una aplicación Android para interactuar con la plataforma de docencia virtual ILIAS*.

[23] Félix Paulano Godino (2009). *Integración de un Sistema de Multiconferencia Web en la Plataforma de Docencia Virtual de la Universidad de Jaén*.

[24] Jessica Aguilera Garrido (2013). *Módulo para integración de contenidos de Google Drive en ILIAS*.

[25] Ricard Lou Torrijos (n.d.) *El API JAXB*. Programación en castellano. Recuperado en Marzo de 2014 de http://programacion.net/articulo/apis_de_java_para_xml_112/3

[26] Walter M. Jenny (Agosto, 2011) *Interacción de WebSphere Process Server mediante interfaz API de servicios web y método de organización JAXB*. IBM, developerWorks. Recuperado en Marzo de 2014 de http://www.ibm.com/developerworks/ssa/websphere/library/techarticles/1004_jenny/1004_jenny.html

[27] Pablo Rojas (n.d.) *Tutorial JAXB: Marshalling*. Campus Curicó. Recuperado en Marzo de 2014 de http://campuscurico.utalca.cl/~pabrojas/?page_id=325

[28] *Annotation Type XmlSeeAlso* (n.d.) Oracle. Recuperado en Marzo de 2014 de http://docs.oracle.com/javase/6/docs/api/javax/xml/bind/annotation/XmlSeeAlso.html

[29] JAXB : Is the annotation @XmlAccessorType is only for Serialization and nothing to do with Binding of data? (Abril, 2012) stackoverflow. Recuperado en Marzo de 2014 de http://stackoverflow.com/questions/10013282/jaxb-is-the-annotation-xmlaccessortype-isonly-for-serialization-and-nothing

[30] *Annotation Type XmlTransient* (n.d.) Oracle. Recuperado en Marzo de 2014 de http://docs.oracle.com/javase/7/docs/api/javax/xml/bind/annotation/XmlTransient.html

[31] Andrés Meza Núñez (n.d.) *Qué es JavaFX.* Scribd. Recuperado en Marzo de 2014 de http://es.scribd.com/doc/80876237/Que-es-JavaFX

[32] *Rich Internet Applications* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Rich_Internet_Applications

[33] Lisandro Pardo (Febrero, 2009) *JavaFX: ¿Nuevo rival para Flash y Silverlight?.* Neoteo. Recuperado en Marzo de 2014 de http://www.neoteo.com/javafx-nuevo-rival-para-flash-y-silverlight-14775/

[34] *FXML* (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://en.wikipedia.org/wiki/FXML

[35] *JavaFX* (n.d.) WordPress. Recuperado en Marzo de 2014 de http://williamrsl.wordpress.com/javafx/

[36] http://www.ilias.de: Esta es la página donde se encuentra el API de ILIAS para descargárselo.

[37] http://www.freepik.es: Esta es la página desde donde se han descargado las imágenes que se han utilizado en la interfaz de usuario de forma gratuita.

[38] Representational State Transfer (n.d.) Wikipedia. Recuperado en Marzo de 2014 de http://es.wikipedia.org/wiki/Representational_State_Transfer

7. APÉNDICES

7.1. Apéndice I. Manual de usuario

En este manual de usuario se van a describir las funcionalidades que han sido implementadas en la aplicación, mostrando las ventanas de la aplicación para facilitar su comprensión y manejo.

Al iniciar la aplicación se muestra la siguiente ventana:

Imagen 7.1.1. Ventana de la aplicación de acceso a ILIAS.

Desde esta ventana se puede acceder a ILIAS. También se pueden gestionar las instalaciones sin tener que acceder a ILIAS. Para salir de la aplicación basta con pulsar sobre el botón 

.

En primer lugar, si se desea gestionar las instalaciones hay que pulsar sobre el botón  y aparecerá la siguiente ventana:

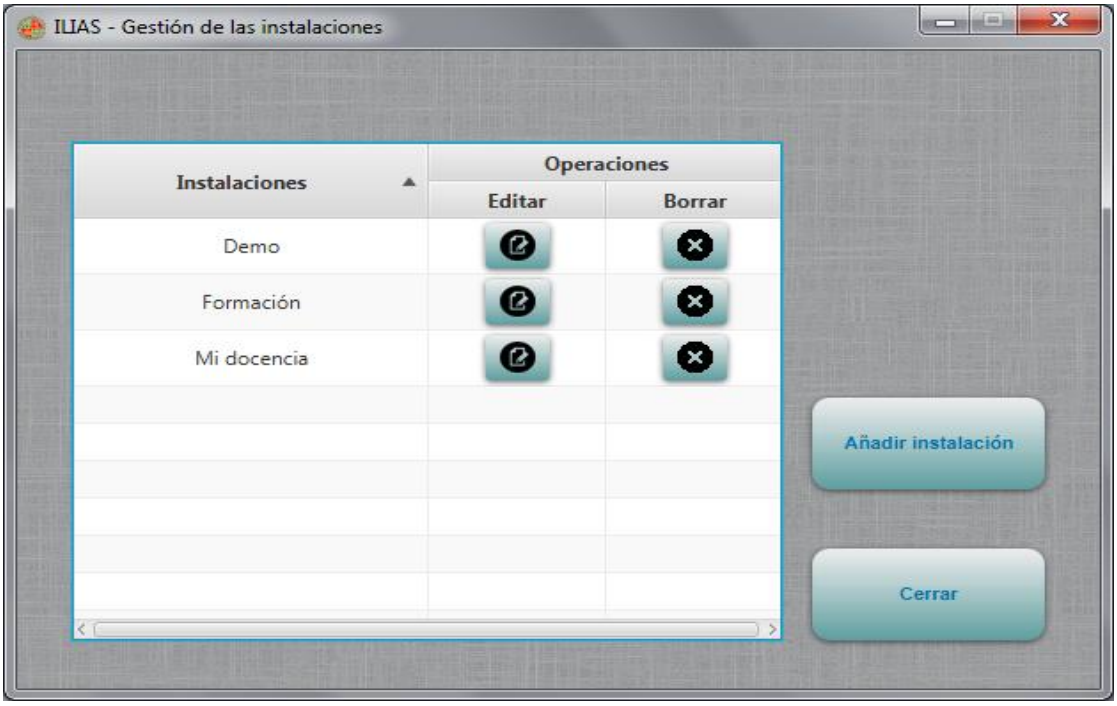


Imagen 7.1.2. Ventana de la aplicación para gestionar las instalaciones.

En esta ventana se muestra la lista de las instalaciones que hay agregadas hasta el momento (en el caso de que haya alguna). Desde aquí se puede añadir una nueva instalación para acceder a ella, editar una instalación de las agregadas e incluso borrar una instalación para no acceder a ella desde la aplicación. Para salir de esta ventana se pulsa sobre el botón 

.

Para añadir una nueva instalación hay que pulsar sobre el botón 

y se mostrará la siguiente ventana:

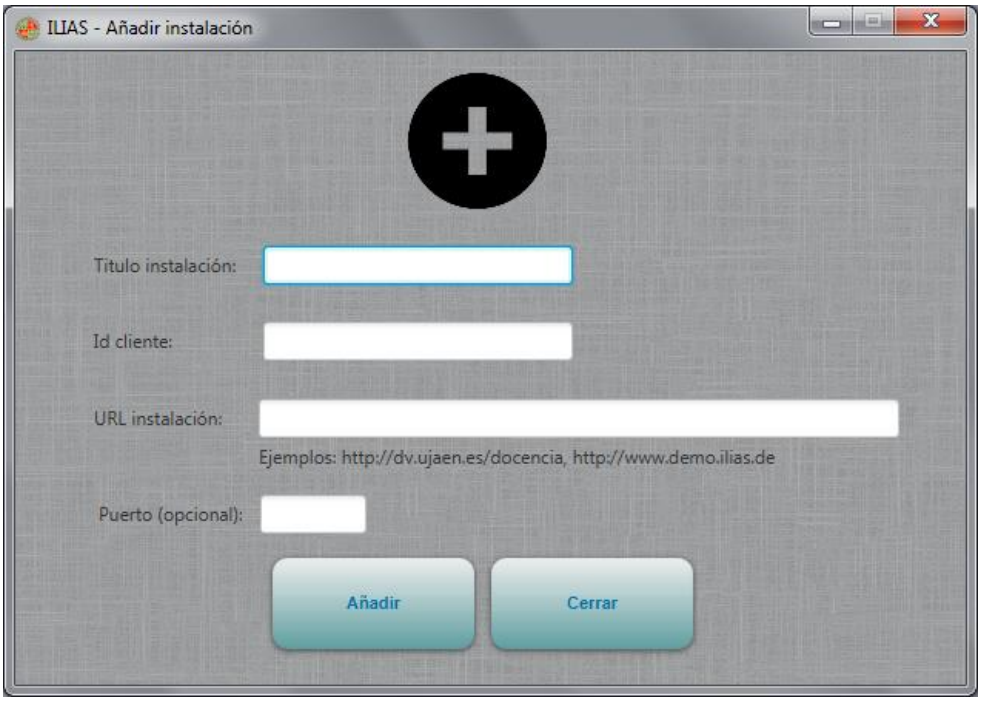


Imagen 7.1.3. Ventana de la aplicación para añadir una nueva instalación.

A la hora de agregar una nueva instalación a la aplicación, hay que introducir el título de la instalación con la que se desea que se muestre en la aplicación, el identificador del cliente como, por ejemplo, docencia y la URL de la instalación. También se puede introducir el puerto si se desea.

Una vez hecho esto, se pulsa sobre el botón 

para agregar la nueva instalación o en caso de querer salir de esta ventana sin realizar ningún cambio se pulsa sobre el botón 

.

Ahora si lo que se desea es editar una instalación ya agregada se pulsa sobre el botón  de la instalación que se quiera editar y se muestra la siguiente ventana:

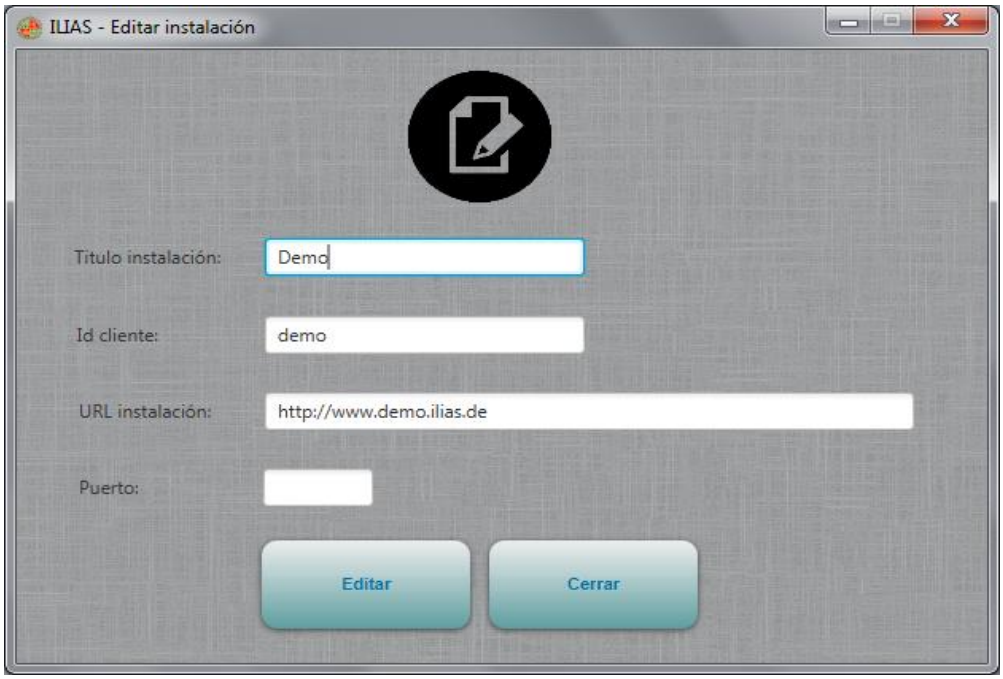


Imagen 7.1.4. Ventana de la aplicación para editar una instalación.

Nada más se muestra la ventana, se mostrarán también los datos de la instalación para poder saber cuáles son los datos que contiene cada instalación y así poder editarlos.

Una vez editados los datos se pulsa sobre el botón 

para confirmar la edición. En el caso de querer salirse de la ventana sin realizar la edición de la instalación, se pulsa sobre el botón 

.

Si lo que se quiere es borrar una instalación agregada, sólo hay que pulsar sobre el botón  de la instalación y aparecerá la siguiente ventana de confirmación:

Imagen 7.1.5. Ventana de la aplicación para borrar una instalación de la aplicación.

En este caso, solamente hay que pulsar sobre el botón 

para confirmar el borrado de la instalación y así poder borrarse. En caso de no querer borrarla se pulsa sobre el botón 

.

Estas son todas las funcionalidades que se pueden realizar para gestionar las instalaciones.

A continuación se va a describir el acceso a ILIAS y todas las funcionalidades que se pueden realizar una vez se haya accedido. Dicho esto, para acceder ILIAS hay que seleccionar la instalación a la que se desea acceder de la lista desplegable, después introducir el usuario y la contraseña; y seguidamente pulsar sobre el botón ENTER del teclado o sobre el botón 

apareciendo la siguiente ventana:

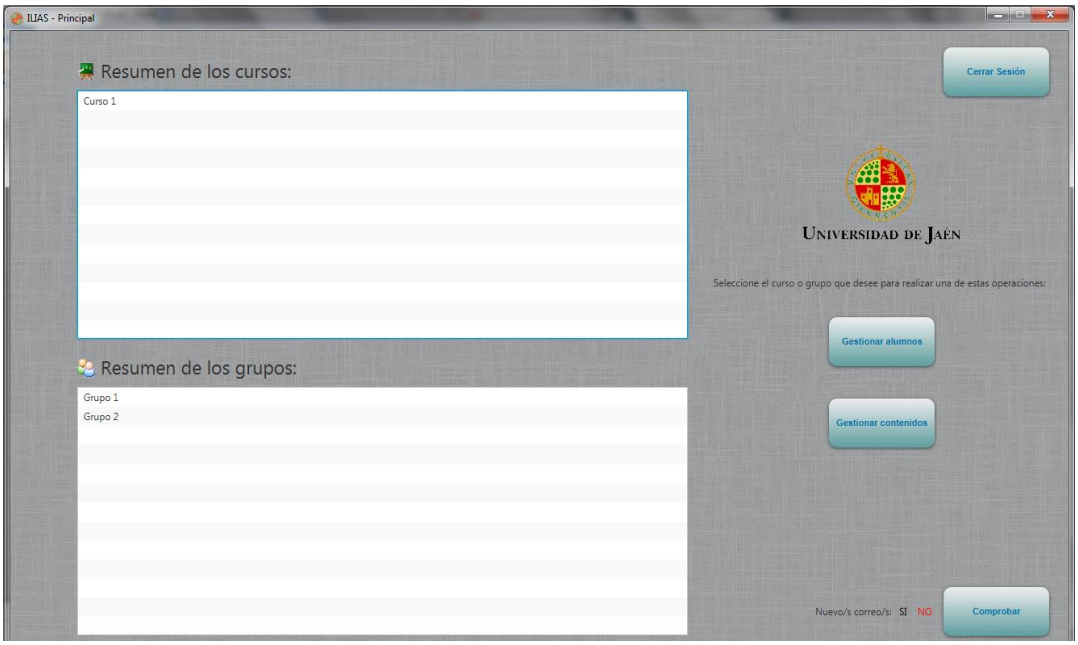


Imagen 7.1.6. Ventana de la aplicación principal del usuario.

En esta ventana se muestran los cursos y/o grupos a los que el usuario que ha accedido a ILIAS está matriculado o es administrador. A partir de aquí se puede realizar la gestión de los alumnos de un curso o grupo e incluso la gestión de los contenidos de un curso o grupo. También se puede comprobar si el usuario que ha accedido a ILIAS tiene nuevos correos o no pulsando sobre el botón 

. Para cerrar la sesión del usuario hay que pulsar sobre el botón 

.

Ahora si lo que se desea es gestionar los alumnos de un curso o grupo tan solo hay que seleccionar dicho curso o grupo de la respectiva lista y pulsar sobre el botón 

. Seguidamente aparecerá la siguiente ventana:

Imagen 7.1.7. Ventana de la aplicación para gestionar a los alumnos de un curso o grupo.

En esta ventana se muestra la lista de los alumnos junto con los administradores (profesores, etc) del curso o grupo seleccionado. Por cada uno de los usuarios que aparecen en la lista se muestra bastante información útil como es la foto personal, el login o usuario, el tipo de usuario que es, si está activo o inactivo actualmente en el curso o grupo y el último acceso que ha realizado a ILIAS.

También se puede tanto dar de alta como de baja a un alumno de un curso o grupo, enviar un correo a uno o varios alumnos e incluso sacar en un Excel la lista de los alumnos del curso o grupo. Para cerrar la ventana hay que pulsar sobre el botón 

.

Para dar de alta a un alumno hay que pulsar sobre el botón  y se mostrará la siguiente ventana:

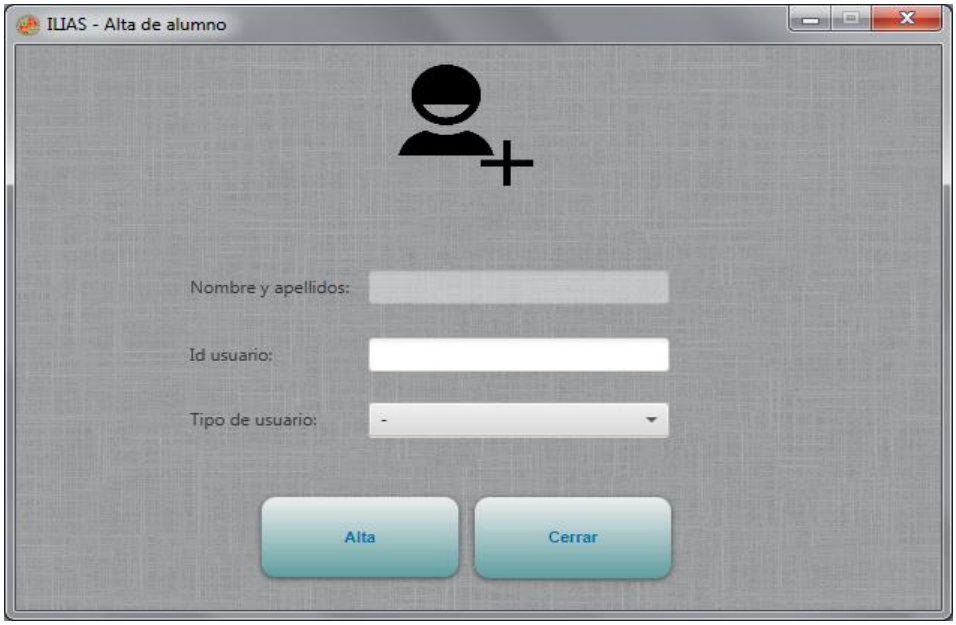


Imagen 7.1.8. Ventana de la aplicación para dar de alta a una lumno en un curso o grupo.

En esta ventana hay que introducir el identificador del usuario al que se quiere dar de alta en el curso o grupo (ahora mismo hay que hacerlo así por el servicio que da de alta a los alumnos en un curso o grupo, en otra versión posterior se hará de tal forma que introduciendo el nombre y los apellidos del alumno ya se dé de alta o incluso introduciendo su DNI), se selecciona el tipo de usuario que va a ser en dicho curso o grupo y seguidamente se pulsa sobre el botón 

. En el caso de que no se quiera dar de alta al alumno en el curso o grupo se pulsa sobre el botón 

.

Para dar de baja a un alumno hay que seleccionar mediante los checkbox ☐ a los alumnos que se van a dar de baja en el curso o grupo y pulsar sobre el botón  mostrándose la siguiente ventana de confirmación:

Imagen 7.1.9. Ventana de la aplicación para dar de baja a un alumno de un curso o grupo.

En esta ventana solamente hay que pulsar sobre el botón 

para confirmar que se desea dar de bajar al alumno o alumnos seleccionados. En el caso de no querer darlos de baja se pulsa sobre el botón 

.

Para enviar un correo a uno o varios alumnos del curso o grupo se pueden seleccionar a los alumnos a lo que se quiere enviar el correo y pulsar sobre el botón  mostrándose la siguiente ventana:

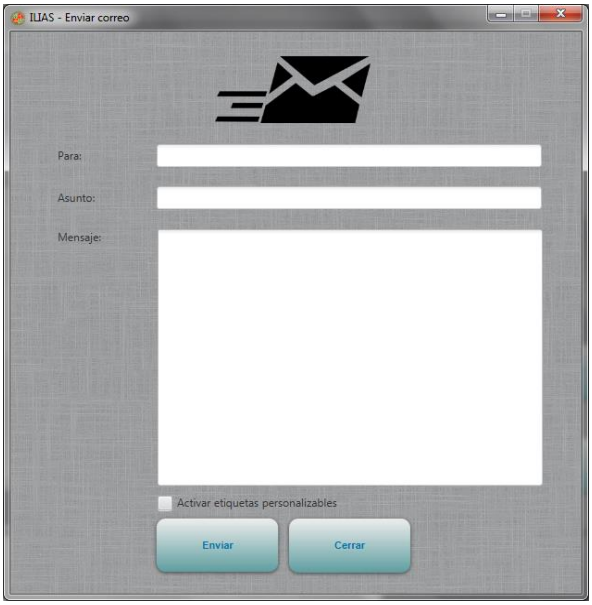


Imagen 7.1.10. Ventana de la aplicación para enviar un correo a uno o varios alumnos de un curso o grupo.

En esta ventana hay que introducir a los destinatarios del correo (login de los destinatarios), el asunto del correo y el contenido del correo. También se pueden usar las etiquetas personalizables activando el checkbox ☐.

Para enviar el correo hay que pulsar sobre el botón 

y para cerrar esta ventana en caso de no querer enviar el correo se pulsa sobre el botón 

.

Por último, para crear la lista con los alumnos de un curso o grupo hay que pulsar sobre el botón 

y se mostrará la siguiente ventana:

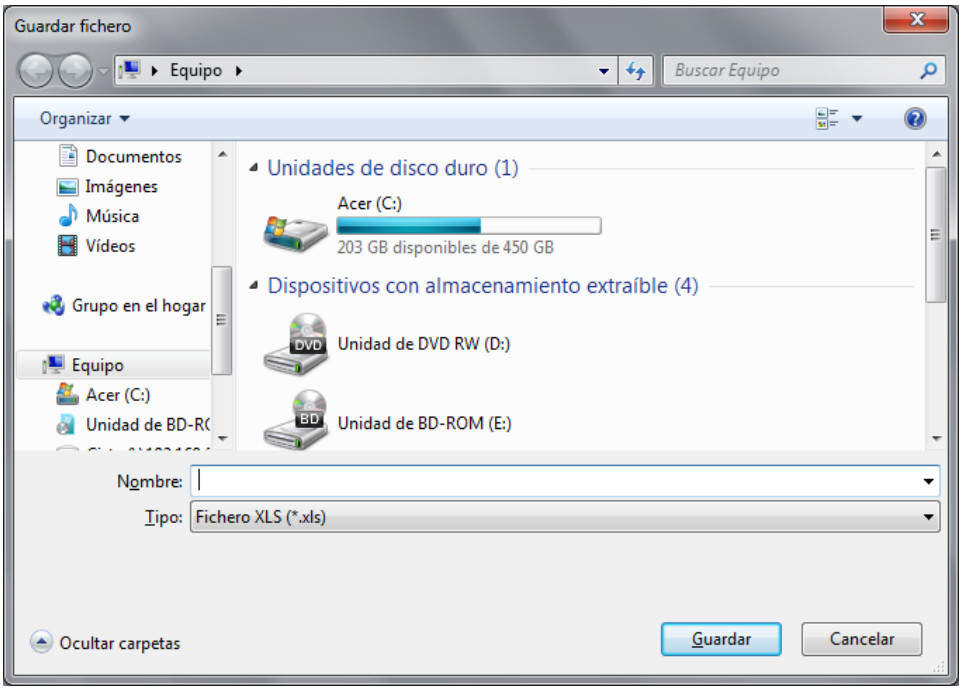


Imagen 7.1.11. Ventana de la aplicación para guardar un fichero Excel con la lista de los alumnos de un curso o grupo.

Esta ventana es la típica ventana para guardar un fichero, por lo que no hay mucho que explicar salvo que el fichero se puede almacenar con las extensiones XLS y XLSX.

Por último, si lo que se desea es gestionar los contenidos de un curso o grupo hay que seleccionarlo de la lista correspondiente y a continuación pulsar sobre el botón 

mostrándose la siguiente ventana:

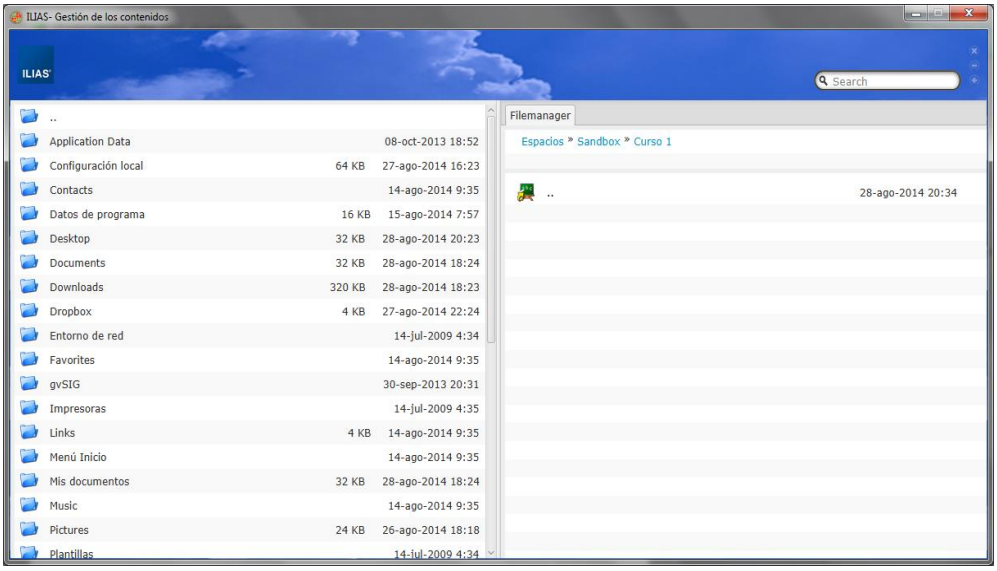


Imagen 7.1.12. Ventana de la aplicación para gestionar los contenidos.

Desde esta ventana se pueden gestionar los contenidos tanto del curso que se ha seleccionado como de cualquier otro curso o grupo al que el usuario esté matriculado o sea administrador.

7.2. Apéndice II. Descripción de contenidos adicionales.

Para poder ejecutar esta aplicación hay que tener instalado el jdk 1.8.0

También se suministra el código fuente de esta aplicación en formato de proyecto Netbeans 7.4, junto con las librerías utilizadas para algunas funcionalidades de dicha aplicación, la memoria de este proyecto, el jdk e incluso una versión compilada del proyecto para poder ejecutar la aplicación directamente.

Para poder utilizarla completamente hace falta tener acceso a Internet.