Simulador de escáner de rayos X: creación y análisis de imágenes

Jefferson Cárdenas Carrillo

Luis Redruello Fernández

UNIVERSIDAD
COMPLUTENSE
MADRID

Trabajo de fin de grado del Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Director: Manuel Montenegro Montes

Autorización de difusión

Los abajo firmantes, matriculados en el Grado en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo de Fin de Grado: “Simulador de escáner de rayos X: creación y análisis de imágenes”, realizado durante el curso académico 2015-2016 bajo la dirección de Manuel Montenegro Montes en el Departamento de Sistemas Informáticos y Computación, y a la Biblioteca de la UCM, a depositarlo en el Archivo Institucional E-Prints Complutense con el objetivo de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

_____ _____

Luis Redruello Fernández Jefferson Cárdenas Carrillo

_____

Manuel Montenegro Montes

Agradecimientos

Queremos aprovechar este espacio para agradecer a todas y cada una de las personas que nos han ayudado a llegar hasta aquí, y de manera muy especial a nuestro director Manuel Montenegro, por su paciencia, dedicación y disponibilidad. Por animarnos y por ayudarnos en todo momento, gracias Manuel.

Finalmente agradecer al centro de formación SEF los recursos que nos han prestado para la realización de este proyecto.

Índice

Resumen

Actualmente cada vez son mayores las medidas de seguridad que nos rodean. Cada vez son más importantes y severas estas medidas de seguridad en infraestructuras y edificios públicos. Como consecuencia de estos hechos, cada vez se aumenta la necesidad de contar con personal de seguridad más cualificado y preparado.

Esto ha provocado que aumente la demanda de formación en esta área, ya que las empresas de seguridad que operan en el aeropuerto deben formar a su personal para que puedan ejercer su trabajo con la mayor eficacia.

Ante esta situación, hemos decidido desarrollar una aplicación para la formación de los alumnos en materia de seguridad aeroportuaria. Con esta aplicación, los usuarios pueden conseguir cualquiera de las tres certificaciones que emite AESA (Agencia Estatal de Seguridad Aérea). Cada certificación es distinta según la zona del aeropuerto en donde opere el alumno.

Por tanto, el objetivo de este proyecto es la implementación de una herramienta de aprendizaje y evaluación por parte de alumnos, que está compuesta por dos aplicaciones que se relacionan mediante la arquitectura cliente-servidor.

Palabras clave

Java, Servicio web, Arquitectura Cliente Servidor, Seguridad Aeroportuaria, Certificación, simulador rayos X, Modelo-Vista-Controlador, Peticiones (GET POST), Aprendizaje, XML.

Abstract

There is currently an increasing amount of security measures around us. These security measures are becoming more important and severe in infrastructure and public buildings. As a result of these developments, the need for more qualified and prepared security personnel is increased.

This has led to increased demand for training in this area, as security companies operating at the airport must train their staff so they can carry out their work with greater efficiency.

In this situation, we have decided to develop an application for the training of students in airport security. With this application, users can get any of the three certificates issued by AESA. Each certification is different depending on the area of the airport where the student operates

Therefore, the objective of this project is the implementation of a learning tool by students, which consists of two applications that are related by the client - server architecture.

Keywords

Java, web services, customer service architecture, airport security, certification, X-ray simulator, model view controller, request (GET POST), learning, XML.

1. Introducción

En el primer punto de este documento vamos a explicar la motivación que nos llevó a realizar este proyecto y de manera general, comentaremos los objetivos que tenemos con la implementación del mismo. Asimismo, también comentaremos el plan de trabajo que hemos seguido, indicando la cantidad de trabajo que nos ha llevado cada uno de estos objetivos, así como en qué parte del presente documento se explica cada uno de dichos objetivos.

a. Motivación y contexto.

En nuestro día a día vemos como cada vez tiene un papel más importante la seguridad. Es una imagen ya bastante común para nosotros salir de viaje y pasar un control de seguridad. En estos controles, pasamos nuestros objetos por un escáner y nosotros pasamos por un arco detector de metales, y en caso de que algo no estuviera en orden, nuestro equipaje tendría que someterse a una inspección manual.

En el marco de esta situación, nos surgió una pregunta: *"¿Qué criterio sigue ese vigilante para saber si mi equipaje es peligroso o no?"* Según la normativa vigente, cada vigilante que está en el aeropuerto debe tener una acreditación con la cual indica que ha superado unas determinadas competencias para poder ejercer este trabajo. Esas competencias vienen determinadas en el PNS (Programa Nacional de Seguridad para la Aviación Civil), documento en el cuál, se recogen todas las aptitudes y habilidades que, a nivel estatal, debe cumplir un vigilante de seguridad del aeropuerto.

En el PNS, por tanto, se indica qué tipo de formación debe recibir cada vigilante, y con qué periodicidad debe refrescar sus habilidades. Dicha formación también está condicionada por la zona del aeropuerto en la que esté cada vigilante. Por tanto, las empresas de seguridad que operan en los aeropuertos están obligadas a enviar a sus trabajadores a centros de formación para que reciban una formación inicial y además (y de manera irrevocable, independientemente de la zona del aeropuerto en la que operen), están obligados a ir cada 6 meses a recibir una instrucción en la que refresquen las competencias adquiridas.

Por tanto, este tipo de instrucción es un nicho de mercado importante para los centros de formación, ya que además de la formación inicial que tienen que impartir, el hecho de tener que dar otro curso cada 6 meses es una importante vía de negocio para tener, con periodicidad fija, unos ingresos de manera constante.

Para este tipo de formación, se suelen usar unas herramientas llamadas CBT (*Computer Based Training*), un software para el aprendizaje de alumnos por medios digitales. Con este software, se pretende integrar todas las partes que componen el curso a través de un programa informático. En este caso, la aplicación, además de contener todo el temario para la formación del alumno, incluye un simulador que emula las funciones de un escáner del aeropuerto, para

que así, el propio alumno pueda interactuar con la aplicación como lo haría en su puesto de trabajo.

Para esto, existe algún software en el mercado, como puede ser el software "*EAGLE"*[5] desarrollado por la empresa ICTS. Esta herramienta es prácticamente la única que existe en el mercado. Por tanto, ser capaces de crear otra, causaría un impacto importante en el mercado, ya que muchos centros de formación querrían una herramienta similar para entrar en el mundo de la formación aeroportuaria.

b. Objetivos

El objetivo principal de este Trabajo Fin de Grado es la implementación de una herramienta para aprendizaje por parte de alumnos, en este caso destinada a la formación del personal de seguridad aeroportuario. Consistirá en dos aplicaciones de escritorio que se relacionan mediante la arquitectura cliente-servidor, y que permitirán evaluar a un alumno para comprobar si tiene adquiridas las competencias necesarias para desarrollar una actividad en el ámbito de la seguridad. Esta aplicación va a tener las siguientes características:

- Introducir o eliminar usuarios de la propia aplicación.
- Presentación de lecciones teóricas en forma de archivos PDF, para que el alumno las vaya leyendo y estudiando por su cuenta, en el orden en el que desee.
- Evaluación de las competencias teóricas de los alumnos adquiridas en el apartado anterior.
- Análisis e interpretación de las imágenes de un aparato de rayos X, para detectar si un equipaje contiene o no, artículos no permitidos en el reglamento de viajeros.
- Evaluación de las competencias prácticas referentes al análisis e interpretación de imágenes descrito en el apartado anterior.
- Posibilidad de mantenimiento por parte del administrador: añadir preguntas de examen y lecciones teóricas.

Las lecciones teóricas, los exámenes y las imágenes varían según el tipo de acreditación que desee obtener el alumno. Nuestra aplicación permitirá adquirir un total de 3 certificaciones. La certificación de tipo 1 está destinada a aquellos vigilantes que están en zonas aeroportuarias pero que no examinan equipaje y que, por tanto, no usan imágenes de escáner.

La certificación de tipo 2 está orientada a operadores de equipos de inspección, tanto de equipaje de mano, bodega, como provisiones de a bordo y suministros. Finalmente, la certificación de tipo 3 está dirigida a operadores de carga y correo. Por tanto, como cada certificación tiene una finalidad distinta, los recursos son distintos para cada una de ellas.

c. Plan de trabajo

En este apartado vamos a indicar en qué orden se realizan cada uno de los objetivos descritos en el apartado anterior, la cantidad de tiempo dedicado a cada uno, y en qué parte de la memoria se desarrolla la explicación de cada uno de estos objetivos.

1. Definición de requisitos y selección de tecnologías a utilizar.

En primer lugar, definimos el alcance del proyecto fijando unas bases y dejando abiertas posibles ampliaciones o mejoras que se podrían implementar en caso de disponer de tiempo para ello. Este punto lo tratamos en el capítulo *3.a Requisitos* de este documento. Una vez marcadas estas pautas, escogimos la arquitectura necesaria para implementar nuestra aplicación, así como las tecnologías que se verían implicadas en este desarrollo. Este punto lo tratamos en el capítulo *3.b Tecnologías utilizadas* de este documento.

2. Diseño de la Base de Datos, *login* y dar de alta usuarios.

Una vez definidos los requisitos, formalizamos un primer diseño de nuestra base de datos, creando todas las tablas y las relaciones entre ellas. Asimismo, implementamos una primera versión de cómo el usuario se registra en la aplicación del lado del cliente, sin peticiones al servidor. Además implementamos funciones de tipo administrador como dar de alta usuarios, también en un entorno local. Este punto lo tratamos en el capítulo *3.d Diseño de la Base de Datos* de este documento.

3. Implementación del servidor

El siguiente paso es desarrollar un servicio web para que funciones como las descritas en el apartado anterior no se realicen del lado del cliente, sino que se implementen mediante peticiones al servidor. Este punto lo tratamos los capítulos *3.c Arquitectura General del Sistema* y *3.e Servicios Soportados* de este documento.

4. Parte teórica

Una vez implementado el servicio web, seguimos añadiendo funcionalidades, tanto de usuarios como de administradores. En este punto, añadimos la posibilidad de que el alumno pueda ver un módulo teórico, así como que el administrador pueda colgarlo en la aplicación. Este punto lo tratamos en el capítulo *3.e Servicios Soportados* y *3.f Implementación del cliente* de este documento.

5. Exámenes

Dado que el alumno debe realizar un examen después de la visualización de la parte teórica, el siguiente punto que implementamos es la posibilidad de la realización de exámenes. A su vez, también implementamos la posibilidad de que el administrador pueda añadir preguntas de las disponibles en la base de datos. Este punto lo tratamos en el capítulo *3.e Servicios Soportados* y *3.f Implementación del cliente* de este documento.

6. Análisis de imágenes

Finalmente, dotamos a nuestra aplicación de la posibilidad de visualizar imágenes para que el alumno las pueda analizar. Este análisis consiste en que el alumno pueda visualizar la imagen con cualquiera de los tres contrastes que posee la aplicación, para poder detectar si el equipaje que se le presenta contiene objetos prohibidos o por el contrario puede continuar sin ser inspeccionado manualmente. Este punto lo tratamos en el capítulo *3.e Servicios Soportados* y *3.f Implementación del cliente* de este documento.

Introduction

In the first section of this document we will explain the motivation that led us to undertake this project and generally discuss the goals we have with its implementation. It explains the work plan we have followed, indicating the amount of work required by each of these objectives, and what part of this document explains each of them.

Motivation and context.

In our daily life security is taking on a more important role. It's a common situation for us to take a trip and pass through a security check. In these controls, our objects are analysed by a scanner and we pass a metal detector, and if something is not in order, our luggage would have to undergo manual inspection.

Under this situation, a question came to us : " *What criteria does the security guard follow to determine if my luggage is dangerous or not?* " Under the current rules, each guard that works at the airport must have an accreditation which certifies some given skills to exercise this job. These skills are determined in the PNS (National Security Programme for Civil Aviation), a document which statewide determine all the skills and abilities that must be met by a security guard working at the airport.

Therefore, the PNS shows what kind of training should receive each guard, and how often to refresh their skills. Such training is also conditioned by the airport area in which each guard does his job. Therefore, security companies operating at airports must send their employees to training centers to receive initial training and also (and irrevocably, regardless of the airport area in which they operate) are forced to go every 6 months to receive instruction to refresh the skills acquired.

Hence, this type of instruction is an important market niche for training centers, as well as the initial training they have to teach. The fact of having to take another course every 6 months is an important of steadily and periodically fixed income for business.

For this type of training some tools called CBT (Computer Based Training) are often used, software for student learning by digital means. This software integrates all the parts of the course through a computer program. In this case, the application contains all the contents for the training of students and includes a simulator that emulates the functions of an airport scanner, so that the students themselves can interact with the application as they would in office of work.

For this purpose, there exists software on the market, such as the " EAGLE " [5] software, developed by the company ICTS. This tool is practically the only one in the market. Therefore, if we are able to create another, we will cause a significant impact on the market, as many training centers would want a tool to enter in the world of airport training.

Aims

The main objective of this Final Project is the implementation of a learning tool for students, in this case for the training of airport security personnel. It will consist of two desktop applications that are related by the client-server architecture, and that will assess a student to check if he has acquired the necessary skills to perform an activity in the field of security responsibilities. This application will have the following features:

- Add or remove users from the application itself.
- Presentation of theoretical lessons in the form of PDF files, so that the student will read and study on their own, in the order in which he wants.
- Evaluation of the theoretical skills of students acquired in the previous section.
- Analysis and interpretation of images of an X- ray machine to detect whether a baggage contains or not items not allowed by the current regulations.
- Assessment of practical skills concerning the analysis and interpretation of images described in the previous section
- Possibility of maintenance by the administrator : add exam questions and theoretical lessons

The theoretical lessons, exams and images vary depending on the type of accreditation want to get the student. Our application will allow to student to receive a total of 3 certifications. Type 1 certification is intended for those guards who are in airport zones but do not examine luggage and therefore not using scan images.

Type 2 certification is oriented inspection equipment operators both hand luggage, and aircraft stores and supplies. Finally, type 3 certification is aimed at operators of cargo and mail Therefore, as each certification has a different purpose, resources are different for each of them.

Work Plan

In this section we will indicate in what order are performed each of the objectives described in the previous section, the amount of time devoted to each, and which part of the memory contains the explanation of each of these objectives is developed.

1. Definition of requirements and selection of technologies to be used.

First, we define the project scope by setting bases and leaving possible extensions or improvement open to implementation if we have time for them. This point is discussed in Chapter *3.a Requirements* of this document. Once marked these guidelines, we choose the

necessary architecture to implement our application, as well as technologies that would be involved in this development. This point is discussed in Chapter *3.b Technologies used* in this document.

2. Database design, *login* and user registration.

Once the requirements have been defined, we made a first design of our database, creating all tables and relations between them. In addition, we implemented an early version of how the user is logged into the client application, without issuing requests to the server. In addition we develop administrator functions such as user registration, also in a local environment. This point is discussed in Chapter 3-d design Database.

3. Web service implementation

For those functions, such as the ones described in the preceding paragraph that are not performed by the client, but rather by issuing request to the server. We treat this point in the chapters *3.c General System Architecture* and *3.e Supported Services* of this document.

4.Theoretical part

Once implemented the web service, we continue to add features, for users and administrators. At this point, we add the possibility that the student can see a theoretical module as well as the administrator to hang in the application. This point is discussed in Chapter *3.e Supported Services* and *3.f Client Deployment* of this document.

5.Exams

Since the student must take an exam after viewing the theoretical part, the next point that we implemented is the possibility of conducting examinations. In turn, we also implement the possibility that the administrator can add questions available in the database. This point is discussed in Chapter *3.e Supported Services* and *3.f Client Deployment* of this document.

6.Image analysis

Finally, we equip our application with the possibility of displaying images to that the student can analyze. The student can view the image with any of the three contrast that owns the application, to detect if the luggage is presented contains prohibited items or otherwise can continue without being inspected manually. This point is discussed in sections *3.e Services Client* and *3.f Supported Deployment* of this document.

2. Antecedentes

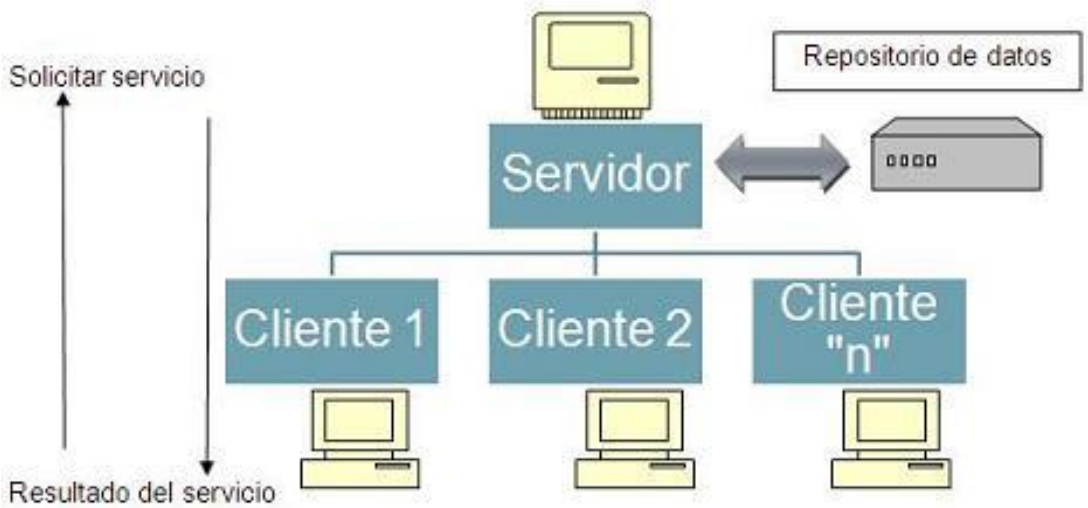
El objetivo de este proyecto es la implementación de una herramienta de aprendizaje por parte de alumnos, que está compuesta por dos aplicaciones de escritorio que se relacionan mediante la arquitectura cliente-servidor. En este apartado, queremos comentar de manera conceptual las principales arquitecturas, herramientas y patrones que hemos utilizado para el desarrollo.

Arquitectura Cliente-Servidor

En primer lugar vamos a definir el paradigma cliente/servidor como un modelo de aplicación en el cual los clientes realizan peticiones de ejecución o realización de tareas a los servidores que son los encargados de proporcionar los recursos.

Una de las principales ventajas de esta arquitectura es la posibilidad de añadir tantos clientes o servidores como sean necesarios, sin incrementar la complejidad del sistema, y, en consecuencia, sin afectar al rendimiento. Otra ventaja muy importante es que los recursos están centralizados y controlados por el servidor. De esta manera las actualizaciones de los mismos se hacen de manera sencilla.

En la imagen adjunta[1] se muestra un diagrama de cómo funciona esta arquitectura.



[1] Imagen obtenida de: http://www.yourerpsoftware.com/content/25-arquitectura

Si aplicamos estos conceptos a nuestra aplicación, nuestros clientes serían los alumnos que ejecutan la aplicación desde sus equipos. Esta aplicación interactúa con el servidor, para recibir los recursos necesarios. Por ejemplo, cuando el usuario quiere realizar un examen teórico, la aplicación le envía una petición al servidor. Este busca en la base de datos las preguntas necesarias y se las devuelve al cliente, que se las muestra al usuario mediante su interfaz gráfica.

Servicio web

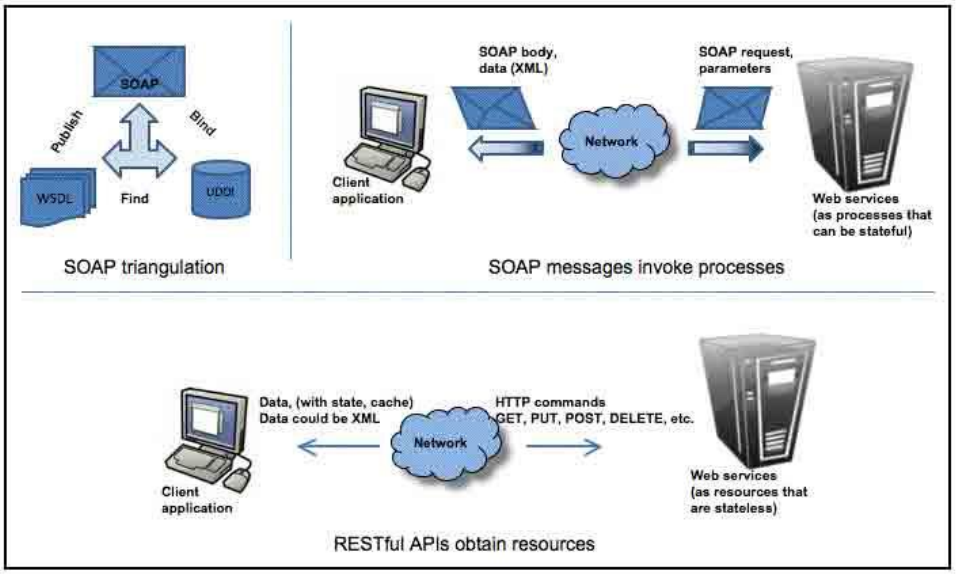
Un servicio web es una herramienta que permite la comunicación entre dos aplicaciones mediante un conjunto de normas y estándares previamente definidos. Así, las aplicaciones pueden intercambiarse mensajes en forma de código XML o JSON entre otros.

Las dos principales arquitecturas orientadas a la implementación de servicios web son SOAP (*Simple Object Access Protocol*) y REST (*Representational State Transfer*). La característica más importante de las arquitecturas SOAP es que generan ficheros WSDL[2] (*Web Service Description Language*) añadiendo así una capa adicional. En el fichero WSDL se declaran todas las funciones de la arquitectura.

Por su parte, las arquitecturas REST no necesitan esta capa, ya que se basan en HTML y XML. Las funciones de la arquitectura están predefinidas, entre las que destacan GET, POST, PUT, DELETE y PATCH. Comentaremos más en detalle estas funciones en los capítulos *3.c Arquitectura General del Sistema* y *3.e Servicios Soportados* del presente documento.

En la siguiente imagen[3] exponemos una comparativa entre ambas arquitecturas.

[2] Más información en: https://es.wikipedia.org/wiki/WSDL
[3] Imagen obtenida de: http://www.programacion.com.py/web/web-services-rest-vs-soap



En la imagen podemos apreciar las diferencias entre ambas arquitecturas. Mientras SOAP es un protocolo de mensajería, que proporciona acceso a muchas tipos de mensaje y operaciones sobre ellos, REST basa su funcionamiento en operaciones concretas y limitadas representadas en comandos HTTP.

Patrón Modelo-Vista-Controlador

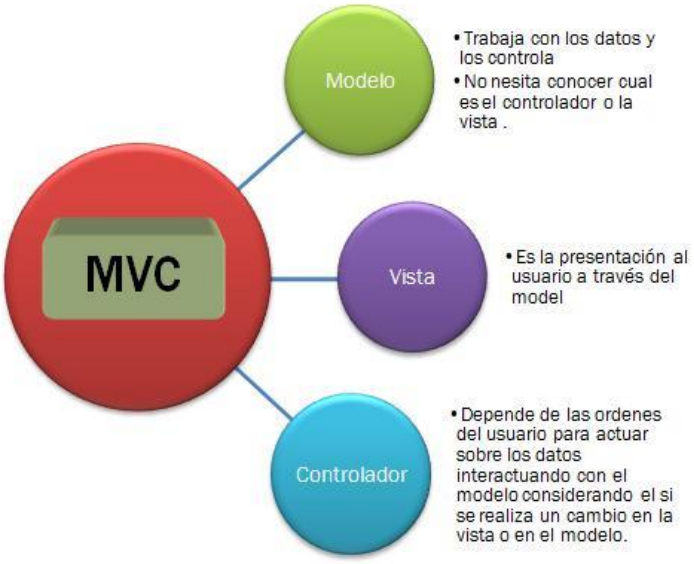
Para definir el patrón modelo-vista-controlador, vamos a citar su definición en Wikipedia debido a la claridad con la que está expuesta. *"Modelo-Vista-Controlador es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones"*[4].

Este patrón divide la aplicación en 3 capas, con la finalidad de distribuir mejor las funcionalidades de la aplicación, consiguiendo un mantenimiento más sencillo. Además permite reducir el contenido de código redundante y aumenta la escalabilidad del sistema a la hora de añadir nuevas funcionalidades.

En la capa del modelo, se encuentra toda la lógica de la aplicación mientras que en la vista se presentan a los usuarios estos datos. El encargado de interactuar entre ambas es la capa controlador. El controlador recibe las peticiones del usuario trasladándoselas al modelo, que devuelve al controlador los datos solicitados, que son presentados de nuevo al usuario mediante la vista.

[4] Más información en: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

En la siguiente imagen[5] mostramos un ejemplo de funcionamiento del patrón modelo-vista-controlador:



[5] Imagen obtenida de: http://u201201013.blogspot.com.es/

3. Desarrollo

En este capítulo vamos a explicar los aspectos de implementación de nuestro proyecto. En primer lugar, en el apartado de *Requisitos* vamos a describir la funcionalidad del sistema desde el punto de vista del usuario. En el siguiente punto (*Tecnologías utilizadas)* vamos a explicar los lenguajes, librerías, entornos de programación y herramientas adicionales que hemos necesitado para el desarrollo del proyecto. En el apartado de *Arquitectura general del sistema*, comentaremos las partes que componen al sistema y el flujo de trabajo. Continuaremos con la sección *Diseño de la base de datos*, en la que hablaremos del diseño de la misma y de cada una de las tablas. En la sección *Servicios soportados* vamos a describir la API del servidor y los servicios que proporciona. Finalmente en el apartado *Implementación del cliente* vamos a comentar la estructura de clases del cliente, explicando el diseño utilizado.

a. Requisitos

El simulador que queremos desarrollar es una herramienta de aprendizaje sobre las competencias en seguridad de los vigilantes de zonas aeroportuarias. Consta de 3 tipos distintos de certificaciones. La certificación de tipo 1 (de aquí en adelante C1) es sólo teórica y va dirigida a aquellos vigilantes que están en zonas aeroportuarias, pero que no manejan objetos y, por tanto, no usan imágenes. El vigilante, una vez certificado, no tiene que volver a "refrescar" su formación, es decir, no necesita ningún tipo de formación adicional o reciclaje. Para obtener la acreditación, el alumno, una vez estudiado todo el temario, debe superar un examen tipo test de 50 preguntas con 4 posibles respuestas de las que sólo una es correcta.

La certificación de tipo 2 (C2) va dirigida a operadores de equipos de inspección, tanto de equipaje de mano, bodega, como provisiones de a bordo y suministros. A diferencia del C1, en esta certificación hay parte teórica y parte práctica. En este caso, los vigilantes tienen que realizar una formación inicial y, una vez adquiridas las competencias, tienen que volver a repetir la parte práctica de manera periódica a modo de reciclaje.

Dado lo anterior hemos de distinguir si el alumno va a realizar una formación inicial o un reciclaje. En la parte de la formación inicial tendrá parte teórica y parte práctica, y en el reciclaje sólo práctica. Al final de cada parte, tanto teórica como práctica, se tiene que realizar un examen. El examen teórico es un examen tipo test de 25 preguntas con 4 posibles opciones de las cuáles sólo una es correcta. El examen práctico consta de un catálogo de imágenes con la mayor variedad posible, en la que el alumno debe decidir si el equipaje o paquete cumple con la normativa, o si, por el contrario, debería ser inspeccionado manualmente (es decir, hay sospecha de que pueda contener objetos prohibidos). El tiempo debe ser limitado y el número de fallos por cada catálogo también, pero la normativa dice que eso queda a elección del formador.

La certificación de tipo 3 (C3), va dirigida a operadores de carga y correo. Es análogo al C2, es decir, también consta de parte teórica y parte práctica y, de nuevo, los alumnos tienen que realizar periódicamente un reciclaje en su parte práctica. En este caso, el examen teórico es de 10 preguntas de tipo test con 4 posibles respuestas correctas, pero la normativa del examen práctico es totalmente igual a la del C2, salvo que, obviamente, son otro tipo de imágenes.

La siguiente tabla muestra a modo de resumen, los componentes del examen correspondientes a cada certificación:

Certificación	Periodicidad	TipoExamen	
C1	Formación Inicial	Teoría	Examen teórico
C2	Formación Inicial	Teoría	Examen teórico
	Formación Inicial	Práctica	Examen práctico
	Reciclaje	Práctica	Examen práctico
C3	Formación Inicial	Teoría	Examen teórico
	Formación Inicial	Práctica	Examen práctico
	Reciclaje	Práctica	Examen práctico

A continuación, pasamos a describir los casos de uso de la aplicación:

1. Login

El usuario que accede a la aplicación es cualquier vigilante de seguridad que trabaje en zonas aeroportuarias o que trabaje con equipos de rayos X. Cuando abre la aplicación tiene que proceder a identificarse. Para ello introduce su DNI sin letra y su contraseña. Una vez iniciada la sesión, puede volver al proceso donde lo dejó en su sesión anterior, es decir, se guardan los aprobados y acreditaciones que haya conseguido. La aplicación no funciona si el usuario no se identifica.

Por otro lado, distinguimos un tipo de usuario administrador, que se encarga controlar las altas y bajas de los usuarios y también se encarga de gestionar la parte de la formación en sí, como por ejemplo, subir módulos teóricos. La interfaz de login es igual para ambos tipos de usuario. Será el sistema el encargado de mostrar la vista de la aplicación adecuada al tipo de usuario.

2. Selección Certificación

Después de identificarse en la aplicación el usuario, de tipo alumno, elegirá el tipo de certificación para la que desea ser evaluado (C1, C2, C3). En la primera pantalla habrá tres botones, cada uno para las distintas certificaciones. Tras elegir una, se mostrarán las opciones de formación inicial o reciclaje.

3. Formación Inicial

Después de escoger una certificación el alumno irá a la pantalla de formación inicial, siempre que no haya obtenido ya esa certificación, es decir, que la haya aprobado previamente. La formación inicial consta de formación teórica y formación práctica.

3.1Formación teórica

Al elegir la formación teórica se mostrará un listado con los distintos módulos teóricos cargados en el sistema, correspondientes a la certificación en la que se encuentra el usuario de tipo alumno. Un módulo es una unidad de teoría, que se presenta en formato PDF o cualquier otro formato de texto, para que el alumno adquiera los conocimientos suficientes de manera previa a la visualización de imágenes[6]. También existirá un botón para empezar el examen, aunque solo estará activo después de terminar todos los módulos teóricos.

3.2 Examen Teórico

Al examen teórico sólo se accede después de terminar el aprendizaje con todos los módulos. Por tanto sólo se realiza un único examen teórico en cada certificación. El examen consta de varias preguntas tipo test con solo una respuesta correcta. Tras terminar el examen se mostrarán los resultados obtenidos por el alumno.

3.3 Formación Práctica

El alumno accede a esta formación una vez concluida su formación teórica y habiendo aprobado previamente su examen teórico. Para la formación práctica, en primer lugar se le mostrarán al alumno una colección de imágenes a modo de entrenamiento de un tipo en concreto (Equipaje, Bodega).

El funcionamiento del aparato original de rayos X consiste en lanzar unas ondas que miden principalmente la densidad de los objetos que traspasa. En base a este resultado, el procesador de la máquina asigna un color determinado a cada uno de los objetos. Estos colores se muestran según un código acordado de manera universal. De esta forma el negro representa objetos muy densos, el naranja materia orgánica y el azul materia inorgánica.

En esta pantalla existen tres botones para cambiar el contraste de la imagen. Estos son:

- Contraste orgánico: Muestra sólo los materiales orgánicos mientras que inhabilita los materiales inorgánicos (por ese motivo suelen salir tonalidades más anaranjadas). Es recomendable para la búsqueda de drogas o explosivos.
- Contraste inorgánico: De manera análoga al anterior, muestra sólo los materiales inorgánicos mientras que inhabilita los materiales orgánicos (por

[6] En la certificación 1 sólo hay parte práctica

este motivo suelen salir tonalidades más azuladas). Suele ser el más utilizado para la búsqueda de armas.

- Contraste blanco y negro: Ayuda a ver la opacidad de los objetos, por tanto es de mucha utilidad para ver detalles que con los contrastes anteriores se nos escapen.

Después de analizar la imagen el usuario podrá elegir si se trata de una maleta sin artículos prohibidos o por el contrario hay que proceder a su inspección. En este último caso, es decir, si el alumno sospecha que el objeto es peligroso, se debe elegir si existe una amenaza obvia o hay que retirar el objeto.

Si el usuario elige amenaza obvia (al pulsar el botón correspondiente), primero seleccionará la parte de la imagen donde considera que se encuentra esta amenaza. Además, saldrá un listado por pantalla con los diferentes tipos de posibles objetos (Arma blanca, explosivos, etc...) Al elegir uno de ellos, se indicará por pantalla si ha acertado o ha cometido un error.

Para analizar cada imagen, el alumno no tiene tiempo límite. Una vez analizada la imagen, se muestra al usuario si su análisis es correcto o no, es decir, si el usuario ha acertado en su decisión.

3.4 Examen Práctico

Un examen práctico es un catálogo de imágenes aleatorias que el usuario debe analizar. La interfaz del examen práctico es similar a la de formación, con la salvedad de que ahora tiene un tiempo limitado y que tampoco se muestran los resultados entre una imagen y otra. Cuando el alumno termine el examen se le mostrará una pantalla con el resultado obtenido. Además se ofrecen las opciones de volver a repetir el examen o de volver a la formación práctica.

4. Reciclaje

Si el usuario de tipo alumno selecciona una certificación que ya ha conseguido (ya la ha aprobado), accede a un repaso de los conocimientos de la misma. Para ello únicamente va a disponer tanto de la formación práctica como del examen práctico. Para poder mantener/renovar la certificación, debe volver a aprobar el examen práctico.

Perfil Administrador

Dentro de la misma aplicación, si el usuario que se loguea es de tipo Administrador, podrá realizar gestionar usuarios (profesores y alumnos) y contraseñas y controlar los permisos.

4.1 Alta de Usuarios

El Administrador será el responsable de dar de alta un usuario en el sistema. Dispone de un formulario para dar de alta en el sistema formado por diferentes campos. Primero insertará el DNI como nombre de usuario. Después le asignará una contraseña. El sistema validará que el usuario no esté ya dado de alta y que la contraseña tenga una longitud adecuada, mostrando un mensaje de error al administrador en caso contrario. Si se realiza la inserción de manera correcta se visualizará el mensaje de éxito al administrador. El sistema debe mantener, para cada usuario, el estado en el que se quedó la última vez que accedió dicho usuario al sistema, es decir, mantiene todos sus logros, calificaciones y certificaciones.

4.2 Eliminar Usuarios

El Administrador puede eliminar un usuario de la siguiente manera. En el menú de administración habrá un submenú para eliminar un determinado usuario. En esta página se verá un desplegable con el listado de usuarios dados de alta en el sistema. Buscará al usuario que desea eliminar, lo marcará y tras pulsar el botón de *eliminar,* el usuario será dado de baja del sistema. Al eliminar un usuario el sistema también eliminará sus diferentes certificaciones, calificaciones y logros.

4.3 Insertar Módulos

El administrador elegirá un archivo con el contenido teórico correspondiente al módulo que desea cargar en el sistema. Primero elegirá de un desplegable el certificado al que pertenece el módulo teórico en cuestión. Después se abrirá una ventana de explorador de archivos. El administrador elegirá el archivo de su sistema y lo cargará en la aplicación. Una vez cargado le asignará como nombre al módulo su id y pulsando sobre el botón *aceptar* se guardarán todos los datos en el sistema. También existe un botón de *cancelar* para deshacer la acción en todo momento.

4.4 Añadir Pregunta

El administrador primero selecciona la certificación a la que pertenece el examen, y posteriormente se mostrará un formulario donde el profesor deberá insertar el enunciado de la nueva pregunta, además de las posibles respuestas del mismo. También debe señalar qué pregunta es la correcta. Tras insertar todos estos datos y pulsando sobre el botón *aceptar* se insertarán las preguntas creadas en el examen seleccionado. Un examen teórico, es una colección de preguntas de una certificación en concreto.

4.5 Añadir Imagen

A la hora de realizar formación práctica, existen unos determinados catálogos de imágenes, en los que se agrupan imágenes de un tipo en concreto para que el alumno las visualice. Cada

certificación va a tener unas galerías propias para la formación práctica y para los exámenes prácticos.

b. Tecnologías utilizadas

En este apartado vamos a mencionar y explicar todas las tecnologías que hemos usado para llevar a cabo nuestro proyecto.

- Java [6]

Es un lenguaje de programación orientado a objetos que permite desarrollar para cualquier plataforma que tenga una máquina virtual de Java instalada. Ha sido elegido para el desarrollo en el lado del cliente, debido a que es con la que más familiarizados estamos los integrantes del grupo.

- Swing [3]

Es una librería para el desarrollo de interfaces gráficas en Java. En primera instancia nos decantamos por Java FX[7] para el desarrollo de la interfaz, pero acabamos descartándola por su elevada curva de aprendizaje inicial. Por tanto elegimos, Swing por su facilidad de uso y porque contenía todos los componentes necesarios para el desarrollo de nuestra interfaz, además de ser la herramienta mejor conocida por los integrantes del grupo.

- Java XML

Es un conjunto de librerías pertenecientes a Java Platform 7 que hemos utilizado para transformar código XML en objetos Java. Cuando nuestro web service accede a la base de datos, devuelve un objeto que transformamos en código XML. Por tanto, a la hora de devolver al cliente esta información, necesitamos esta librería para realizar la transformación de XML a Java.

- Eclipse [8]

Eclipse es un IDE (Entorno de Desarrollo Integrado) compuesto por herramientas de desarrollo. Ha sido utilizado tanto para el desarrollo de la parte del cliente como del servicio web. Permite incluir librerías para añadir funcionalidades que mejoren el funcionamiento de la aplicación.

[7] http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

- Maven [12]

Maven es una herramienta software diseñada para facilitar el uso de librerías externas y su importación. Es una herramienta que puede ser usada para construir y gestionar cualquier proyecto basado en Java. Ha sido utilizada con el propósito de empaquetar el proyecto con las librerías externas..

- Tomcat [13]

Tomcat es software que permite ejecutar servlets en aplicaciones web. Es de código abierto y ha sido desarrollado por la Apache Software Foundation. Entre sus ventajas frente a otras herramientas similares se encuentran que es multiplataforma, que tiene menos complejidad y que probablemente sea uno de los servidores de aplicación más populares, lo que se traduce que ante cualquier problema va a ser más fácil encontrar solución entre la comunidad de desarrolladores. Estas ventajas fueron las que nos hicieron decantarnos por ella.

- MySQL [1]

MySQL es un sistema gestor de bases de datos de código abierto. Es muy conocido entre los desarrolladores por su ligereza y simplicidad. Estos dos motivos, sumados a que tiene licencia GPL, nos hicieron decantarnos por ella frente a Oracle, que era la otra posibilidad que contemplábamos. Para su integración en Java usamos la librería *Mysql Connector*

- Github [10]

Github es una plataforma de desarrollo simultáneo alojada en la nube, que permite trabajo colaborativo. Gracias a su popularidad entre los programadores de todo el mundo, permite dar gran difusión a las aplicaciones que aloja. Se ha utilizado debido a su integración con el entorno de trabajo Eclipse, facilitando el desarrollo en paralelo y el control de versiones, permitiendo así a los miembros del equipo trabajar de manera independiente en el mismo repositorio.

- Google Drive [9]

Google Drive es una de las herramientas online de Google, en este caso para el almacenamiento de archivos. A través de nuestras cuentas institucionales de la UCM, hemos ido usando la herramienta para compartir documentos de investigación, así como las versiones de la memoria.

- Librerías

Para el desarrollo de nuestra aplicación hemos tenido que añadir múltiples librerías a nuestro código para que lo dataran de la funcionalidad necesaria. Del lado del cliente hemos usado librerías como *Commons io* y *Commons Codec*, para lecturas y aperturas de ficheros binarios (como la carga de imágenes y PDF). Por su parte, del lado del *web service* hemos usado *JAX-RS*, que es necesaria para las anotaciones en REST.

c. Arquitectura general del sistema

En este apartado vamos a explicar la arquitectura de nuestra implementación. El sistema se compone de un cliente y un servidor. La aplicación cliente es la encargada de la interacción con el usuario y recoge los eventos de la aplicación (cargar un examen, cargar un módulo, cargar una imagen, inicio de sesión, etc.). Además, contiene toda la lógica de la aplicación. Para su implementación hemos seguido el patrón modelo-vista-controlador. Para más detalles de implementación consultar el apartado *3.f Implementación del cliente* de este documento.

Por su parte el servicio web se encarga de acceder a la base de datos y enviar al cliente los recursos que este le pide (imágenes, PDFs, preguntas…etc.). Las peticiones del cliente al servidor se realizan mediante funciones GET, POST y DELETE. Estas funciones son métodos HTTP que suelen corresponder con operaciones CRUD (*Create Read Update Delete*).

Si el cliente necesita algún recurso realiza una petición GET. Si por el contrario, si el cliente quiere insertar datos, realiza una petición de tipo POST al servicio web. Estas peticiones también se realizan para el envío de datos sensibles, como por ejemplo el Login ya que la contraseña va cifrada. Finalmente las peticiones DELETE se utilizan para borrar información del sistema.

Ante estas peticiones el servidor devuelve un *Reponse Status Code*, que son códigos que devuelven información sobre el estado de la solicitud del cliente. Existen diversos tipos, como mensajes de éxito, de error, informativos o de redireccionamiento[8]. Entre los que más hemos utilizado, están el código 200 OK, que se devuelve cuando una petición GET se realiza de manera satisfactoria o el código 201 MODIFIED cuando una petición UPDATE se realiza correctamente. Si por el contrario, hay un error en la petición de un recurso o este no existe, se devuelve un código 404 NOT FOUND.

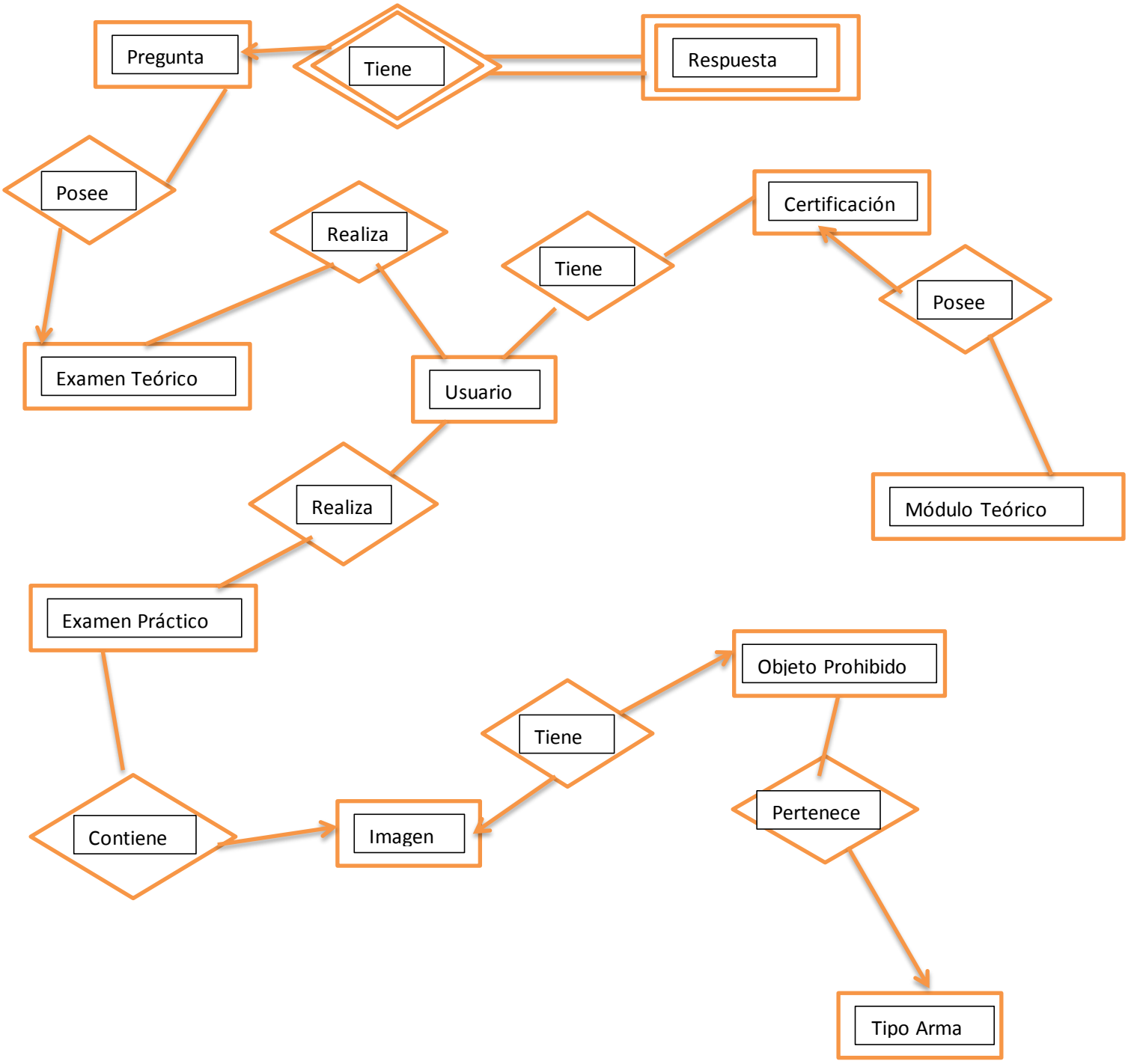
Por ejemplo, si el alumno decide realizar un examen práctico, está solicitando mediante la interfaz del cliente una imagen, por lo que la aplicación cliente envía una petición de tipo GET al *web service*. Por su parte el servicio web busca el directorio de la imagen en la base de datos, que es enviada en forma de array de bytes al cliente. Una vez recibido, el cliente crea un archivo con este array de bytes, que es el que muestra al usuario como imagen.

[8] Se pueden consultar todos los *Reponse Status Code* en este enlace: https://developer.mozilla.org/es/docs/Web/HTTP/Status

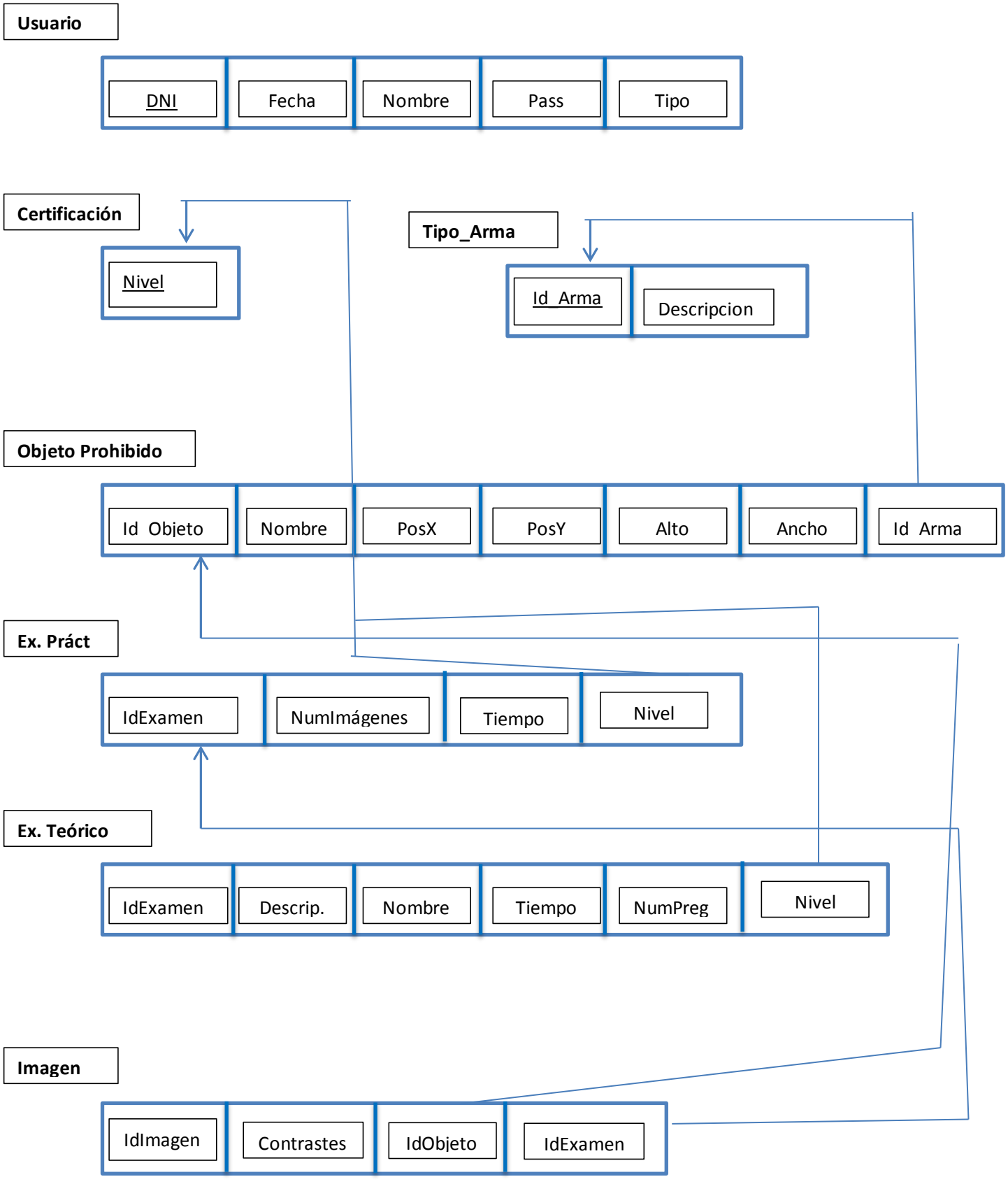
d. Diseño de la Base de Datos

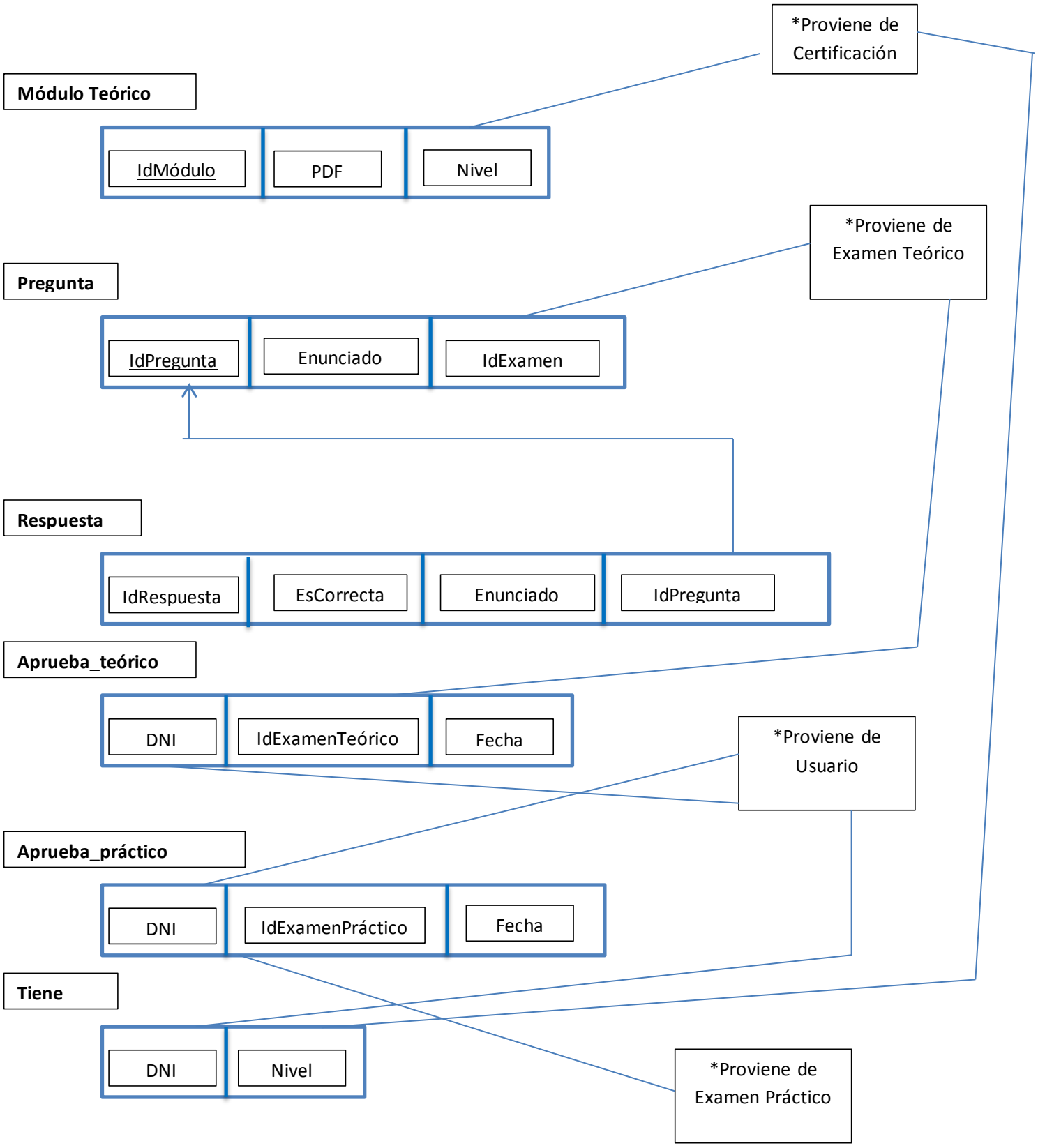
Para esta aplicación hemos diseñado una base de datos para contener los recursos del sistema, como puede ser el almacenamiento de usuarios, administradores, preguntas o lecciones teóricas. Para las imágenes estamos guardando en la base de datos la ruta del servidor en la que se encuentran, para reducir de manera significativa el tamaño de la base de datos.

Se trata de una base de datos relacional, que hemos diseñado siguiendo el modelo entidad relación. A continuación mostramos un primer boceto de este diseño. Por simplicidad, vamos a dibujar únicamente las entidades y las relaciones, prescindiendo de los atributos



Si de este modelo Entidad Relación, pasamos a modelo relacional, obtenemos las tablas de nuestra base de datos:





A continuación, vamos a explicar, a modo de resumen, el contenido de cada una de las tablas de nuestra base de datos anteriormente mencionadas:

- Usuarios: Almacena toda la información de los usuarios: DNI, nombre y tipo de usuario (alumno o administrador).
- Certificación: Hace referencia a la acreditación que consigue el usuario, que viene indicada en el campo nivel.
- Tipo Arma: En esta tabla guardamos todos los posibles tipos de objetos peligrosos que se encuentran registrados en la aplicación, con un código y una descripción.
- Objeto Prohibido: En esta tabla se guardan las coordenadas del objeto prohibido de cada imagen.
- Examen práctico: Almacena el número de imágenes que tiene cada examen práctico según a la certificación que corresponda (nivel).
- Examen Teórico: Almacena el número de preguntas que tiene cada examen práctico según a la certificación que corresponda (nivel).
- Imagen: En esta tabla se guardan referencias a todos los contrastes posibles de la imagen.
- Módulo Teórico: En esta tabla se guardan los PDF (como tipo BLOB) correspondientes a cada certificación (nivel)
- Pregunta: Almacena todas las preguntas teóricas, guardando una referencia a la certificación que pertenecen.
- Respuesta: Almacena todas las respuestas, guardando una referencia de la pregunta a la que pertenecen.
- Aprueba Teórico: Lista todos los usuarios que han aprobado un examen teórico.
- Aprueba Práctico: Lista todos los usuarios que han aprobado un examen teórico.
- Tiene: Relaciona los usuarios con las certificaciones que han obtenido.

e. Servicios soportados

En esta sección de la memoria vamos a comentar los detalles de implementación que hemos realizado para la parte del servicio web. El *web service* funciona como un proveedor de servicios para la parte del cliente. Por tanto está configurado para recibir peticiones, y solamente devuelve *Reponse Status Code* o los recursos solicitados.

Si por ejemplo, el cliente solicita un usuario, pueden darse dos alternativas. Si el usuario existe y se encuentra en la base de datos, el servicio web le devolverá al usuario el recurso solicitado, en este caso un string que representa un XML con la información del usuario. Si por el contrario, el usuario no existe, el web service devuelve un código de error, en este caso 404 NOT FOUND.

Para nuestra implementación, hemos dividido la implementación en un total de 9 paquetes, que pasamos a describir a continuación:

Config

El paquete Config contiene clases para la configuración del sistema. En él se implementan las clases *db.properties*, y *log4j2properties*. En la primera se establecen los parámetros de configuración del usuario de la base de datos, tanto para MySQL como para Oracle. Por su parte, la clase *log4j2properties* sirve para configurar un registro (log), donde van a ir los mensajes que vaya devolviendo la aplicación. Los mensajes pueden ser de tipo depuración (debug), de tipo información, de tipo error…etc. En función de la configuración se muestran unos mensajes u otros.

Data

Este paquete contiene una clase por cada recurso de la base de datos, como pueden ser usuarios, imágenes, preguntas o respuestas. En cada una de estas clases se definen los atributos de las tablas y se crea la constructora. Además, se implementan métodos getters y setters.

Database

El paquete Database se encarga de interactuar con la base de datos. Contiene las clases *DAO* y *DBConnection*. La primera accede a la base de datos para realizar operaciones de consulta, inserción, borrado o actualización sobre las tablas del sistema.

Por su parte la clase DBConnection crea una instancia de la conexión a BBDD. Obtiene los datos de la base de datos del archivo *properties* y realiza una conexión utilizando el Driver

Services

El paquete Services creamos una clase Java por cada atributo de la aplicación: exámenes teóricos, prácticos, imágenes usuarios…etc. Si analizamos clase a clase:

- CertificadoResource: recibe todas las peticiones relacionadas con un certificado.
- ExamenPracticoResource: recibe todas las peticiones relacionadas con un examen Práctico.
- ExamenToericoResource: recibe todas las peticiones relacionadas con un examen Teórico.
- ImagenResource: recibe todas las peticiones relacionadas con la imagen, además accede a las distintas rutas donde están alojadas las imágenes y se la envía al cliente.
- ModuloTeoricoResource: recibe todas las peticiones relacionadas con un módulo teórico, además es la encargada de devolver los archivos PDF al cliente.
- ObjetoProhibidoResource: recibe todas las peticiones relacionadas con un objeto prohibido.

- PreguntaResource: Además de recibir las peticiones relacionadas con las preguntas, tiene métodos de inserción y consulta sobre la base de datos.
- RespuestaResource: Además de recibir las peticiones relacionadas con las respuestas, tiene métodos de inserción y consulta sobre la base de datos.
- UserResource: Todo lo relacionado con los usuarios (insert, update and delete) además hace algunas comprobaciones útiles para el cliente como comprobar si existe un usuario o si ha aprobado un determinado examen.
- ServerAplication: es la clase que inicializa el web service y se encarga de instanciar la clase que accede a la BBDD y todas las clases que reciben las peticiones (las mencionadas arriba).

A continuación, pasamos a describir todos los servicios soportados:

- Class UserResource:
    - Public String getUserApp(@FormParam("dni") String dni,@FormParam("pass") String pass)
        - Método para consultar un usuario de la base de datos.
        - Tipo: POST
        - Path: “get”
        - Parámetros: ‘dni’ , ‘pass’ que representan los datos del usuario
        - Produces: “application/xml”

    - public String getListUsers()
        - Método usado para obtener todos los usuarios.
        - Tipo:POST
        - Path: "get"
        - Produces: "application/xml"

    - Public Response insertUser(@FormParam("nombre") String nombre, @FormParam("dni") String dni,@FormParam("pass") String pass)
        - Método que inserta un nuevo usuario
        - Tipo:POST
        - Path: "add"
        - Parámetros: ‘nombre’, dni’ , ‘pass’ que representan los datos del usuario a insertar.
        - Produces: " MediaType.TEXT_HTML"

    - public Reponse deleteUser(@PathParam("dni") String dni)
        - Método para borrar un usuario.
        - Tipo:DELETE
        - Path: "{dni}"
        - Parámetros: dni’ , ‘pass’ que representan los datos del usuario

- Public Response checkUser(@PathParam("dni") String dni)
  - Método para consultar un usuario.
  - Tipo: GET
  - Path: "{dni}"
  - Parámetros: dni', que representa al usuario.

- Public Response updateUser (@PathParam("dni") String dni, @FormParam("nombre") String nombre, @FormParam("pass") String pass)
  - Método para actualizar los datos del usuario.
  - Tipo: POST
  - Path: "update/{dni}"
  - Parámetros: 'nombre', 'dni', 'pass' que representan los datos del usuario.

- Public Response tieneTeoricoAprobado(@FormParam("dni") String dni, @FormParam("id") int id)
  - Método para comprobar si el alumno ha aprobado el examen que se indica por parámetro.
  - Tipo: POST
  - Path: "tieneteorico"
  - Parámetros: 'dni' que representa al usuario que ha aprobado el examen. 'id' representa el examen teórico que ha sido aprobado.

- Public Response tienePracticoAprobado(@FormParam("dni") String dni, @FormParam("id") int id)
  - Método para comprobar si el alumno ha aprobado el examen que se indica por parámetro.
  - Tipo: POST
  - Path: " tienepractico "
  - Parámetros: 'dni' que representa al usuario que ha aprobado el examen. 'id' representa el examen práctico que ha sido aprobado.

- Class ExamenPracticoResource

  - public String getExamenPractico(@PathParam("nivel") int nivel)
    - Método para obtener un examen práctico de la base de datos.
    - Tipo:GET
    - Path: "{nivel}"
    - Produces: "application/xml"
    - Parámeteros: 'nivel' que representa la certificación a la que pertenece el examen.

  - public Response insertaAprobadoPractico(@FormParam("dni") String dni, @FormParam("id") int id)

        - Método para indicar que el alumno ha aprobado un examen práctico.
        - Tipo: POST
        - Path: "/aprueba"
        - Produces: MediaType.TEXT_HTML
        - Parámetros: 'dni' que representa al usuario que ha aprobado el examen. 'id' representa el examen práctico que ha sido aprobado.

- Class ExamenResource

    - public String getExamenTeorico(@PathParam("nivel") int nivel)
        - Método para obtener un examen teórico de la base de datos.
        - Tipo:GET
        - Path: "/teorico/{nivel}"
        - Produces: "application/xml"
        - Parámetros: 'nivel' que representa la certificación a la que pertenece el examen.

    - Public String getExamenPractico(@PathParam("nivel") int nivel)
        - Método para obtener un examen práctico de la base de dato.
        - Tipo:GET
        - Path: "/teorico/{nivel}"
        - Produces: "application/xml"
        - Parámteros: 'nivel' que representa la certificación a la que pertenece el examen.

- Class Examen Teorico

    - public String getExamenTeorico(@PathParam("nivel") int nivel)
        - Método para obtener un examen teórico de la base de datos.
        - Tipo:GET
        - Path: "/teorico/{nivel}"
        - Produces: "application/xml"
        - Parámetros: nivel' que representa la certificación a la que pertenece el examen.

    - public Response insertAprobadoTeorico(@FormParam("dni") String dni, @FormParam("id") int id)
        - Método Método para indicar que el alumno ha aprobado un examen teórico.
        - Tipo: POST
        - Path: "/aprueba"
        - Produces: "MediaType.TEXT_HTML
        - Parámetros: 'dni' que representa al usuario que ha aprobado el examen. 'id' representa el examen práctico que ha sido aprobado.

- Class Certificado Resource

    - public String getCertificadosFromDNI(@PathParam("dni") String dni)
        - Método que obtiene los certificados que posee un alumno.
        - Tipo: GET
        - Path: "{dni}"
        - Produces: "application/xml"
        - Parámetros: ‘dni’ que representa al usuario que ha aprobado el examen.

    - public Response insertaCertificacion(@FormParam("nivel") int nivel, @FormParam("dni") String dni)
        - Método que actualize la base de datos cuando el usuario adquiere una certificación.
        - Tipo: POST
        - Path: "/obtiene"
        - Produces: MediaType.TEXT_HTML
        - Parámetros: ‘dni’ que representa al usuario que ha aprobado el examen. ‘nivel representa la certificación que ha sido obtenida por el alumno.

- Class Imagen Resource
    - public byte[] getImage(@PathParam("tipo") String tipo, @FormParam("examen") int examen, @FormParam("id") int id)
        - Método para obtener una imagen de la base de datos.
        - Tipo: POST
        - Path: "/{tipo}"
        - Produces: "image/png"
        - Parámetros: ’id’ representa el id de la imagen. ‘examen’ que representa al examen al que pertenece la imagen. ‘tipo’ representa si la imagen contiene objetos prohibidos o no.

    - public String getImageFromId(@PathParam("id") int id)
        - Método que devuelve una imagen dado su identificador.
        - Tipo: GET
        - Path: "{id}"
        - Produces: "application/xml”
        - Parámetros: ’id’ representa el id de la imagen.

    - public String getListImagesFromExam(@PathParam("examen") int examen)
        - Método que devuelve todas las imágenes de un examen.
        - Tipo: GET
        - Path: "/imagenes/{examen}"
        - Produces: "application/xml”

        - Parámetros: ‘examen’ que representa al examen al que pertenecen las imágenes.

    - public Response insertaImagenLimpia(@FormParam("id") int id, @FormParam("normal") String normal, @FormParam("bn") String bn, @FormParam("organico") String organico, @FormParam("inorganico") String inorganico)
        - Método que inserta una imagen sin objetos prohibidos en la base de datos.
        - Tipo: POST
        - Path: "limpia/insert"
        - Produces: MediaType.TEXT_HTML
        - Parámetros: ‘id’ representa el identificador de la imagen. ‘normal’ es la imagen sin ningún contraste. ‘bn’ es la imagen con el contraste blanco/negro. ‘organico’ es la imagen formada con el contraste orgánico. ‘inorganico’ es la imagen con el contraste inorgánico.

    - public Response insertaImagenProhibido(@FormParam("id") int id, @FormParam("normal") String normal, @FormParam("bn") String bn, @FormParam("organico") String organico, @FormParam("inorganico") String inorganico, @FormParam("x") int x, @FormParam("y") int y, @FormParam("ancho") int ancho, @FormParam("alto") int alto, @FormParam("tipoarma") int tipoarma)
        - Método que inserta una imagen con objetos prohibidos en la base de datos.
        - Tipo: POST
        - Path: "prohibido/insert"
        - Produces: MediaType.TEXT_HTML
        - Parámetros: ‘id’ representa el identificador de la imagen. ‘normal’ es la imagen sin ningún contraste. ‘bn’ es la imagen con el contraste blanco/negro. ‘organico’ es la imagen formada con el contraste orgánico. ‘inorganico’ es la imagen con el contraste inorgánico.

- Class ModuloTeoricoResource

    - public byte[] getPDF(@PathParam("nivel") int nivel,@PathParam("modulo") int modulo)
        - Método para transformar un archivo PDF en un array de bytes.
        - Tipo: GET
        - Path: "pdf/{nivel}/{modulo}"
        - Produces: "application/pdf"
        - Parámetros: ‘nivel’ representa la certificación a la que pertenece el módulo. ‘módulo’ es el indentificador del módulo que se desea obtener.

- public Response uploadPDF (@PathParam("nivel"), int nivel, @PathParam("modulo") int modulo, @FormParam("file") String file)
  - Método para subir un fichero a la aplicación.
  - Tipo: POST
  - Path: "upload/{nivel}/{modulo}"
  - Produces: "application/x-www-form-urlencoded"
  - Parámetros: 'nivel' representa la certificación a la que pertenece el módulo. 'módulo' es el identificador del módulo que se desea añadir. 'file' es el archivo en cuestión que se desea añadir.

- public String getModuloTeorico(@PathParam("nivel") int nivel,@PathParam("modulo") int modulo)
  - Método que obtiene un módulo teórico de la base de datos.
  - Tipo: GET
  - Path: "upload/{nivel}/{modulo}"
  - Produces: "application/xml"
  - Parámetros: 'nivel' representa la certificación a la que pertenece el módulo. 'módulo' es el identificador del módulo que se desea obtener.

- Class ObjetoProhibidoResource

  - public String getObjetoProhibido(@PathParam("id") int id)
    - Método para obtener un objeto prohibido desde la base de datos.
    - Tipo: GET
    - Path: " {id}"
    - Produces: "application/xml"
    - Parámetros: 'id' que representa al tipo de objeto prohibido.

- Class PreguntaResource

  - public String getListaPreguntas(@PathParam("examen") int examen)
    - Método que devuelve las preguntas de un examen.
    - Tipo: GET
    - Path: " {examen}"
    - Produces: "application/xml"
    - Parámetros: 'examen' representa el examen al que pertenece la pregunta.

  - public String insertaPregunta(@FormParam("enunciado") String enunciado, @FormParam("examen") int examen)
    - Método que añade preguntas a la base de datos.
    - Tipo: POST
    - Path: " /insert"
    - Produces: "application/xml"

        - Parámetros: ‘examen’ representa el examen al que pertenece la pregunta. ‘enunciado’ representa al enunciado de la pregunta.

- Class RespuestaResource
    - public String getListaRespuestasFromPregunta (@PathParam("pregunta") int pregunta)
        - Método que devuelve todas las posibles respuestas a una pregunta en concreto.
        - Tipo: GET
        - Path: "{pregunta}"
        - Produces: "application/xml"
        - Parámetros: ‘pregunta’ que representa la pregunta cuyas respuestas se quieren obtener.

    - public Response insertaRespuesta(@FormParam("id") int id,@FormParam("respuesta") String respuesta, @FormParam("correcta") int correcta)
        - Método para inserter posibles respuestas a preguntas
        - Tipo: POST
        - Path: " /insert"
        - Produces: MediaType.TEXT_HTML
        - Parámetros: ‘id que representa la pregunta cuyas respuesta se quiere insertar. ‘respuesta’ que representa el enunciado de la misma. ‘correcta’ que indica si la respuesta insertada es la correcta o no.

- Class TipoArmaResource

    - public String getTipoArma(@PathParam("id") int id)
        - Método obtiene el tipo de objeto prohibido.
        - Tipo: GET
        - Path: " {id}"
        - Produces: "application/xml"
        - Parámetros: ‘id’ que representa el objeto prohibido.

    - public String getListTipoArma()
        - Método que devuelve una lista con todos los posibles tipos de objetos prohibidos
        - Tipo: GET
        - Path: " list"
        - Produces: "application/xml"

Servlet

El paquete Servlet solamente implementa la clase *UserServlet*, que sirve para inicializar un Servlet. Devuelve el *path* de la aplicación.

Teoría

En el paquete Teoría no implementamos ninguna clase, simplemente lo usamos como contenedor para almacenar todos los módulos teóricos. A este paquete accede la clase *ModuloTeoricoResource.*

Tools

El paquete Tools simplemente implementa la clase *Tools*, que implementa un método para devolver la fecha del sistema.

f. Implementación del cliente

En este apartado vamos a explicar los detalles de la implementación que hemos realizado para la parte del cliente. En este caso hemos seguido un patrón Modelo-Vista-Controlador, por la facilidad de su mantenimiento y por permitir reducir el contenido de código redundante y por la posibilidad de aumentar la escalabilidad del sistema a la hora de añadir nuevas funcionalidades

El flujo de la aplicación empieza cuando un alumno interactúa con la vista, demandando algún tipo de servicio, como puede ser ingresar en el sistema o realizar un examen. El controlador recibe esa petición y se la transmite al modelo, que busca en la lógica de la aplicación el recurso solicitado. Finalmente, el modelo le da una respuesta al controlador, que presenta la solución al usuario mediante la interfaz gráfica.

Para nuestra implementación, hemos dividido la implementación en un total de 6 paquetes, que pasamos a describir a continuación:

Controlador

Como indica su nombre es el controlador de la aplicación. Contiene las clases *AppMain* y *Controlador*. La primera simplemente contiene el método Main del cliente, mientras que la segunda tiene todas las operaciones que definen la lógica de la aplicación.

Esta clase recibe los parámetros que ha generado la vista a partir de la interacción del usuario con la aplicación. Se encarga de enviar estos datos a la clase *DBService*, que es la encargada de hacer las llamadas al servicio web.

Database

En el paquete Database se encuentran las clases encargadas de realizar las distintas conexiones a la base de datos. En este paquete están las clases: *DBConnection*, *DBInterface*, y *DBServer*. La primera clase se encarga de abrir de abrir una conexión con la base de datos mediante la librería mysqlConnector.

Por su parte, *DBInterface* define todas las operaciones de inserción, borrado, actualización de los distintos atributos de objetos de la base de datos. Finalmente, la clase *DBServer* recibe los datos del servicio web y se los devuelve al controlador. Por tanto es la encargada de interactuar con el *web service*, formando las peticiones correctamente mediante la clase *HttpConnection*.

Lógica

En el paquete Lógica se encuentra todo el modelo de la aplicación. En sus clases se almacenan los registros correspondientes a las tablas de la base de datos, para ser proporcionados al controlador cuando este los demande. Por tanto, representa las tablas de la base de datos y representa instancias de las mismas cuando necesitamos manipular los objetos en el cliente.

PDF

En este paquete se incluye la clase PDFReader, que es la encargada de llamar al lector de PDF del sistema operativo. Recibe el path de un archivo con extensión PDF y lo abre, llamando al lector de PDF del sistema.

Tools

En este paquete se incluyen las clases *Task* y *Utilities*. La clase *Task,* que extiende a una clase SwingWorker, es utilizada para calcular, en segundo plano, el límite de tiempo que el alumno tiene para realizar un examen, mientras va contestando al resto de preguntas.

Por su parte utilizamos a la clase *Utilities* la utilizamos como recurso para llamar a operaciones que utilizamos bastante a menudo. Entre estas operaciones se encuentran crear una nueva fecha a partir de la fecha del sistema, generar números aleatorios para elegir preguntas o imágenes de manera aleatoria, o cifrar una contraseña con el algoritmo MD5.

Vista

Este paquete contiene las clases utilizadas para la implementación de la interfaz gráfica. A su vez, está dividida en dos subpaquetes, admin y alumno, que implementan la interfaz de su tipo de usuario correspondiente.

4.Conclusiones

Para finalizar este documento, nos gustaría describir los resultados que hemos obtenido desarrollando este trabajo y las dificultades encontradas. En líneas generales podríamos decir que sí que hemos cumplido con los objetivos fijados, aunque nos hemos encontrado con diversos problemas que han dificultado el cumplimiento de estos objetivos.

Por tanto, en primer lugar vamos a analizar si los objetivos descritos en el capítulo *1.b Objetivos* de esta memoria, han sido cumplidos o no. Como primer objetivo nos fijamos la posibilidad de introducir o eliminar usuarios de la propia aplicación. Este objetivo ha sido completado, pero si bien es cierto que tuvo un mayor grado de complejidad del que nos esperábamos al principio. El hecho de tener que realizar peticiones al servicio web para obtener los usuarios de la base de datos, nos obligaba a tener una buena implementación del mismo para que pudiera estar listo este objetivo. Por tanto fue un paso importante en el desarrollo de la aplicación.

Como segundo objetivo, teníamos la presentación de lecciones teóricas en forma de archivos PDF, que también ha sido cumplido no sin dificultades encontradas a la hora de cargar el PDF desde el *web service*. Entraremos a comentarlas en más detalle en el capítulo *4.a Dificultades encontradas* de este documento.

El siguiente objetivo era dar al usuario la posibilidad de realizar exámenes teóricos, sobre los contenidos que había estudiado previamente. Se puede decir que este objetivo cumple con nuestras expectativas, ya que lo hemos podido implementar de la forma que pensamos originalmente, usando un hilo para calcular el tiempo de examen.

Los dos siguientes objetivos están muy relacionados. Son por un lado, el análisis e interpretación de las imágenes de un aparato de rayos X, para detectar correctamente si un equipaje contiene o no artículos no permitidos en el reglamento de viajeros, y por otro lado, la evaluación de las competencias prácticas referentes a este análisis. Para estos objetivos tuvimos que investigar profundamente en la normativa, qué objetos están permitidos y cuáles no lo están en cada una de las formas de transporte de equipaje (el reglamento es distinto para el equipaje de la cabina de pasajeros que para el equipaje facturado). Una vez tuvimos estos conceptos claros, tuvimos que investigar la manera más óptima para cargar y almacenar las imágenes. Estas dificultades las comentaremos en el capítulo *4.a Dificultades encontradas* de este documento.

Por último, nos fijamos como objetivo la posibilidad de mantenimiento de la aplicación por parte de un administrador, para que pudiera dar de alta o baja usuarios, añadir o eliminar preguntas o subir nuevas lecciones teóricas como funcionalidades principales. Estas funciones las fuimos desarrollando en paralelo, según construíamos las funcionalidades para el perfil alumno.

a.Dificultades encontradas

En este apartado vamos a enumerar las principales dificultades encontradas a la hora de la implementación de este proyecto.

Conexión al Servicio Web

El principal problema que nos hemos encontrado en el desarrollo de esta aplicación, ha sido la configuración del servicio web y su conexión con la aplicación. Para esto, investigamos en diversas fuentes entre las que destacan [2], [7] y [11]

El primer paso que dimos para la implementación, fue ajustar la configuración para que se inicie el *web service*. Una vez tuvimos esto claro, pasamos a entender las anotaciones, el paso de parámetros y el envío y la recepción de datos. Por último, otro punto de dificultad fue la realización de las llamadas al servicio web desde el cliente.

Implementación de la interfaz gráfica

En paralelo con la implementación del servicio web, construimos la interfaz gráfica (en primer lugar del *login*). En un primer momento, nos decantamos por el uso de JavaFX, porque nos permitía personalizar aún más nuestra interfaz, pudiendo utilizar hojas de estilo. Sin embargo, nuestro avance era muy lento, ya que la curva de aprendizaje inicial era muy elevada. Nos documentamos en el API y necesitábamos mantener varios controladores por cada vista, lo cual nos era muy complicado de mantener en el diseño.

Además, teníamos problemas a la hora de integrar *Scene Builder* (herramienta de diseño gráfico), con Eclipse, y por tanto no podíamos avanzar a la velocidad que queríamos. Por tanto, decidimos cambiar la interfaz y pasarla a Swing, y a partir de ese momento construir el resto de pantallas de este modo.

Carga de ficheros de tipo PDF

Otra de las principales dificultades con la que nos hemos encontrado ha sido la funcionalidad del administrador de poder cargar módulos teóricos en la aplicación. La idea que tuvimos fue transformar el archivo PDF en un array de bytes. Sin embargo, cuando el cliente realiza al *web service* una petición, no podíamos mandar este array de bytes, ya que es muy complicado enviar estos bytes en la cabecera de una petición POST.

Como solución, desde el cliente codificamos esos bytes a una codificación en base 64 para convertirlos a tipo cadena, que ahora sí podíamos pasarlo como parámetro en una petición. Posteriormente, el servicio web recibe la petición e invierte el proceso, es decir, convierte el

string recibido en base 64 a un array de bytes que son guardados en la tabla módulo teórico en el campo PDF, de tipo BLOB.

Carga de imágenes

Finalmente querríamos destacar las dificultades encontradas en la carga de imágenes. En primera instancia decidimos guardar todas las imágenes en la base de datos, pero vimos que ocupaban demasiado tamaño. Como solución, en vez de guardar las imágenes en la base de datos, las guardamos en el propio servicio web y en las tablas guardamos la ruta de la imagen.

Para acceder a las imágenes el *web service* busca, en función del examen práctico, la ruta de la misma. Este le envía la imagen al cliente en forma de array de bytes, que se encarga de crear un archivo y mostrárselo al usuario en la interfaz Swing.

b. Aplicación del proyecto

Como hemos comentado en el capítulo *1a. Motivación y contexto* de este documento, cada vez son más importantes y severas las medidas de seguridad en infraestructuras y edificios públicos. Como consecuencia de estos hechos, cada vez se aumenta la necesidad de contar con personal de seguridad más cualificado y preparado.

Por este motivo, cada vez son más los centros de formación que ven una oportunidad en el mundo de la seguridad. Como dato estadístico, en la Comunidad de Madrid hay aproximadamente unos 40 centros de formación acreditados para poder impartir el curso de vigilante de seguridad.

A este dato se le suma que es un sector muy poblado, ya que en los últimos años ha crecido de manera notable el número de titulados en seguridad. Con motivo de la crisis económica que ha sufrido el país, muchas personas de entre 40 y 60 años que se quedaron sin empleo vieron una salida en el mundo de la seguridad.

Por tanto los centros de formación tienen que adaptar sus cursos a las necesidades reales del mercado y explorar nuevas vías de ingresos. El mundo de la seguridad aeroportuaria es una importante fuente de ingresos ya que, como hemos comentado en este documento, existe una obligación reflejada en la normativa vigente para que el personal de seguridad acuda a cursos de reciclaje para mejorar y refrescar las cualidades obtenidas.

Teniendo en cuenta que son muy pocas las herramientas de este estilo que existen en el mercado, consideramos que una aplicación como la que hemos desarrollado, puede llegar a tener un peso importante en el sector, ya que, por todo lo expuesto anteriormente, las empresas de seguridad y centros de formación están interesadas en iniciarse en la seguridad aeroportuaria.

Además, aunque nuestra aplicación está claramente orientada al aprendizaje de personal de seguridad, podría ser reutilizada para cualquier otro tipo de alumnos. El esqueleto de peticiones al servidor se mantendría y el principal cambio que habría que realizar serían los recursos que se muestran, pero el funcionamiento sería análogo al de nuestra aplicación.

Conclusions

To end this document, we would like to describe the results we have obtained with the development of this work and the difficulties encountered. In general we could say yes we have met the objectives, although we have encountered various problems that have made meeting these objectives more difficult.

Therefore, first we will analyze whether the objectives described in section 1.b Objectives of this specification have been met or not. The first objective we have set ourselves is the possibility to introduce or remove users from the application itself. This objective has been completed, but if it is true that had a greater degree of complexity we expected at first. The fact of having to make requests to the web service for users of the database, forced us to have a good implementation of it so he could be ready this goal. Therefore it was an important step in the development of the application.

The second objective that we had the presentation of theoretical lessons in the form of PDF files, which also has been fulfilled not without difficulties encountered when loading the PDF from the web service. We enter to discuss them in more detail in Chapter *4.a Difficulties encountered* this document

The next goal was to give the user the ability to perform theoretical examinations on the contents that he had previously studied. One can say that this objective meets our expectations as we were able to implement it the way we originally thought, using a thread to calculate the time of examination.

The following two objectives are closely related. They are on the one hand, the analysis and interpretation of images of an X- ray machine to detect correctly if baggage does not contain or not allowed in the rules of travelers, and on the other hand, the assessment of practical skills concerning this analysis. For these objectives we had to thoroughly investigate the regulations, which objects are allowed and which ones are not in each of the forms of transportation of luggage (the rules are different for cabin baggage of passengers for checked baggage). Once we had these concepts clear, we had to investigate the optimal way to charge and store images. These difficulties will be discussed in chapter *4.a Difficulties encountered* in this document.

Finally, we aimed at the possibility of maintenance of the application by an administrator, so that he could introduce or remove users, add or delete questions or raise new theoretical lessons as main features. These functions were developed in parallel, as we built the functionality for the student profile.

Difficulties encountered

In this section we will list the main difficulties encountered when implementing this project.

Connecting to Web Service

The main problem that we have encountered in the development of this application has been setup web service and its connection with the application. For this, we investigate various sources among which we highlight [2], [7] and [11]

The first step we took in the implementation was to adjust the settings for the web service initialization. Once we had this clear, we understood the annotation system, parameter passing and sending and receiving data. Finally, another point of difficulty was making calls to the web service from the client

Implementation of the GUI

In parallel with the implementation of the web service, we built the graphical interface (the login secret). At first, we opted for the use of JavaFX, because it allowed us to further customize our interface and can use style sheets. However, our progress was very slow, since the initial learning curve was very high. We consulted the API and needed to maintain multiple controllers for each view, which was very difficult to keep us in the design.

We also had problems in integrating Scene Builder (graphic design tool), with Eclipse, and therefore we could not go at the speed we wanted. Therefore, we decided to change the interface to Swing, and from then build the rest of screens in this way.

Upload PDF files

Another major difficulties that we have encountered has been the admin functionality to load theoretical modules in the application. The idea we had was to transform the PDF file into an array of bytes. However, when the client makes a request to the web service, we could not send this array of bytes, as it is very difficult to send these bytes in the header of a POST request.

As a solution, from the client we encrypt these bytes base 64 encoding to convert a string type, that now could pass as a parameter in a request. Subsequently, the web service receives the request and reverses the process, i.e., converts the received base 64 string to an array of bytes that are stored in the moduloTeorico table.

Proyect aplication

As we discussed in chapter 1a. Motivation and *context* of this document, security measures are becoming increasingly important and severe in infrastructure and public buildings. As a result of these developments, the need for more security personnel qualified and prepared it is increased.

For this reason, more and more training centers who see an opportunity in the world of security. As statistical data, in the Community of Madrid there are approximately 40 accredited training centers to teach the course security guard.

To this fact is added that is a very populated sector, and in recent years has increased significantly the number of graduates in security. Due to the economic crisis that has hit the country, many people between 40 and 60 years who were unemployed saw a way out in the world of security.

Therefore training centers have to adapt their courses to the real needs of the market and explore new revenue streams. The world of airport security is an important source of income since, as discussed herein, there is an obligation reflected in current regulations for security personnel go to refresher courses to improve and refresh the qualities obtained.

Given that very few tools of this kind that exist in the market, we consider an application as we developed, can have an important weight in the sector, since, for all the above, safety companies and training centers are interested in starting airport security.

Although our application is clearly oriented learning security personnel, it could be reused for any other students. The skeleton of requests to the server would remain and the main change we would have to make is modified the resources would be shown, but the operation would be similar to that of our application.

5.Trabajo futuro

En este apartado queremos comentar funcionalidades que nos parecen muy interesantes para mejorar y complementar la aplicación. Si bien son múltiples las mejoras que se pueden realizar, pensados que las siguientes son las más importantes.

Cambiar aplicación de escritorio por aplicación web

Otra de las posibles mejoras a implementar sería la migración de aplicación de escritorio a aplicación web. De esta manera, la aplicación podría estar alojada en plataformas de aprendizaje como Moodle, que complementarían y mejorarían la experiencia del alumno, de nuevo ofreciendo un mejor servicio. Además, al estar en la web, no haría falta que esta herramienta estuviera instalada en el equipo, facilitando así su difusión.

Generación de contrastes

Otra posible funcionalidad que nos parece muy destacable sería dotar a este software de propiedades para el reconocimiento de imágenes, es decir, poder incluir en la herramienta un motor para la creación de imágenes, tal y como realiza un escáner de verdad.

La idea sería dotar a la aplicación de funcionalidades para que ella misma genere sus propios contrastes, emulando al funcionamiento de las máquinas de rayos X, que como ya hemos explicado, basan sus colores en una escala predefinida según la densidad del objeto. De esta manera se reducirían los recursos necesarios para la creación de la aplicación.

Creación de imágenes

Finalmente pero no por ello menos importante, nos gustaría mejorar la aplicación con una funcionalidad de creación de imágenes. La idea sería poder tener distintos tipos de objetos, tanto objetos cotidianos como objetos no permitidos para su transporte, y combinarlos de diversas maneras, generando así infinitas imágenes.

De esta forma, se podrían crear diversos grados de complejidad en las imágenes, pudiendo introducir niveles en la parte práctica del análisis de imágenes. De esta manera, los alumnos podrían ir superando distintos grados de complejidad, antes de enfrentarse al examen para poder obtener su acreditación.

Interfaz con Java FX

El primer cambio que nos gustaría llevar a cabo, es uno de los ya comentados problemas a la hora de realizar la implementación, y no es otro que desarrollar la interfaz en Java FX. Aunque Swing está obsoleto[9], pensamos que Java Fx es una herramienta mucho más potente y moderna. Entre sus ventajas se encuentra la posibilidad de poder tener un mayor control sobre los elementos visuales, pudiéndoles asignar hojas de estilo para modificarlos, así como su sencillez a la hora de permitir la migración a interfaz web, además de permitir el uso de animaciones.

[9] Más información aquí: http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html#6

6.Referencias

[1] Boronczyk, Timothy. Jump Start MySQL.SitePoint. 2015.

[2] Burke, Bill. RESTful Java with JAX-RS 2.0, 2nd Edition. O'Reilly Media, Inc. 2013

[3] Friesen,Jeff. Beginning Java 7. Berkeley, CA. 2011

[4] Friesen,Jeff. Java XML and JSON. Berkeley, CA. 2016

[5] Equipo de Desarrolladores de ICTS Hipania.
http://eagle.ictseuropesystems.com/media/es/ICTS-Europe-Systems-Eagle-CBT.PDF.2011.

[6] Jiménez Marín, Alfonso. Aprende a Programar con Java. Paraninfo.2016

[7] Kalin, Martin. Java Web Services: Up and Running, 2nd Edition. O'Reilly Media, Inc. 2013

[8] Kulkarni,Ram. Java EE development with Eclipse : develop, debug, test, and troubleshoot Java EE 7 applications rapidly with Eclipse. Birmingham. 2015

[9] Lamont, Ian. Google Drive & Docs in 30 minutes: the unofficial guide to the new Google drive, docs, sheets & slides. i30 Media. 2015

[10] Pipinellis, Achilleas. GitHub essentials: unleash the power of collaborative workflow development using GitHub, one step at a time. Birmingham. 2015

[11] Purushothaman, Jobinesh. RESTful Java Web Services - Second Edition. Packt Publishing. 2015

[12] Siriwardena, Prabath. Maven essentials : get started with the essentials of Apache Maven and get your build automation system up and running quickly. Birmingham. 2015

[13] Vukotic, Aleksa y Goodwill, James. Apache Tomcat 7. Berkeley, CA. 2011

7. Aportaciones de los integrantes

Por último, en este apartado de la memoria queremos indicar la aportación personal de cada uno de los integrantes.

Luis Redruello Fernández

En primer lugar me gustaría manifestar que la mayor parte del trabajo que hemos realizado ha sido en total colaboración entre los dos miembros del equipo. Si bien es cierto que cada uno llevaba más peso en la parte del cliente y otro en la del servidor, en todo momento nos hemos apoyado para ayudarnos mutuamente cuando alguno de los dos lo necesitaba.

Dicho esto paso a enumerar las principales tareas que he desarrollado en este proyecto:

- En primer lugar, la búsqueda de algún producto similar en el mercado con el que poder inspirarnos a la hora de implementar nuestro proyecto.

- Seguidamente, inicié una búsqueda de documentación para determinar qué alcance podía tener nuestro proyecto y hasta dónde podíamos abarcarlo.

- Viendo que no había muchos productos similares en el mercado, empecé una búsqueda de los recursos que nos harían falta para hacer viable el proyecto, como son las imágenes y los módulos teóricos.

- El siguiente paso fue debatir sobre las tecnologías a utilizar, ventajas e inconvenientes de cada una de ellas.

- Una vez decididas las tecnologías y la arquitectura del sistema, fui configurando e instalando todos los entornos necesarios para la implementación: Eclipse, Scene Builder, mySQL…etc.

- En paralelo con el punto anterior, comencé con un primer diseño de la base de datos, y su transformación a modelo relacional.

- Una vez hecho esto, empecé a plantear posibles casos de uso para su discursión y aís establecer de manera definitiva el alcance del proyecto.

- Investigación de las nuevas tecnologías a tratar, como era Java FX

- El siguiente punto, fue el desarrollo de una primera versión de la función login. Este desarrollo fue en un entorno local, mientras mi compañero investigaba el funcionamiento del servicio web.

- Una vez conseguida esta primera función, y una vez que mi compañero había configurado el servicio web, investigamos la manera de poder conectar ambas aplicaciones.

- El siguiente paso fue crear una interfaz para el perfil administrador, para poder probar las funciones de insertar y borrar usuarios.

- Dadas las dificultades que estábamos teniendo con Java FX, decidimos cambiar toda la interfaz gráfica a Swing.

- Para seguir avanzando con la aplicación, incluí todos los recursos necesarios en la base de datos.

- El siguiente paso fue la implementación de la interfaz del examen teórico y la configuración de las preguntas con el web service.

- Debido a la manera en la que íbamos implementando ambas aplicaciones, tuve que modificar el diseño de la base de datos y por tanto sus tablas.

- El siguiente paso fue finalizar la implementación de los exámenes teóricos.

- Debido a que íbamos con algo de retraso con respecto de la planificación establecida, comencé a elaborar el presente documento.

- Finalmente, realicé el testeo de la aplicación, intentando ver posibles mejoras y corrigiendo puntos débiles que nos habían quedado pendientes.

Jefferson Cárdenas Carrillo

Antes de nada me gustaría destacar que para el desarrollo de esta aplicación, hemos estado en contacto de manera constante, para intentar avanzar lo máximo posible. En mi caso, me he centrado más en toda la parte del servicio web, y debido a problemas de planificación, también implementé algunas funciones del cliente.

A continuación, paso a listar las principales funciones que he realizado:

- En primer lugar comencé con la investigación del servicio web, ya que nunca había implementado ninguno. Los tipos de arquitecturas posibles y las ventajas e inconvenientes de cada uno.

- En paralelo con esta investigación, también me documenté sobre cuál sería la mejor arquitectura posible para realizar la implementación. De esta manera tanto mi compañero como yo tendríamos una visión completa de la aplicación y cada uno

expondría sus argumentos, para de este modo poder entender ambos todo el funcionamiento y estar de acuerdo desde un inicio.

- Instalación de todo el entorno de desarrollo: Eclipse, mySQL, Tomcat...etc.

- Una vez tenía claro qué tipo de servicio web quería implementar, empecé a investigar sobre su comportamiento para intentar conseguir que funcionara. En este apartado incluyo todo lo relacionado con el funcionamiento y aplicación de las peticiones GET/POST, el funcionamiento de *los Reponse Status Code,* el tratamiento del código XML…etc.

- El siguiente paso fue conseguir la función login haciendo llamadas al web service. Este paso fue uno de los más complicados de toda la implementación. Las mayores dificultades estuvieron en el funcionamiento de los *Reponse Status Code,* y en el tratamiento del código XML, ya que si el usuario existe y se encuentra en la base de datos, el servicio web le devolvía al usuario un string que representa un XML con la información del usuario.

- La siguiente tarea a realizar era la manera de poder enviar archivos PDF desde el servidor al cliente. El principal problema con el que me encontré era cómo enviar este archivo en una petición al cliente. Para ello tuve que consultar y revisar diversa documentación al respecto.

- Finalmente para la implementación del punto anterior, decidí transformar el archivo PDF en un array de bytes, codificando esos bytes a una codificación base 64 para poder convertirlos a tipo cadena y de esta manera poder enviarlos en una petición.

- Una vez conseguido cargar archivos PDF en la aplicación, y que el administrador fuera capaz de subirlos, comencé la parte de los exámenes teóricos para poder devolver preguntas y respuestas al cliente y que el perfil administrador fuera capaz de añadir y eliminar preguntas.

- El siguiente punto, fue intentar cargar una imagen desde la base de datos en la aplicación. En un principio estábamos guardando todas las imágenes y sus contrastes en la base de datos pero ocupaban demasiado. Así que tuve que volver a rediseñar la base de datos teniendo esto en cuenta.

- Para cargar las imágenes intenté realizar una función análoga a la de la carga de archivos PDF, en la que el servicio web busca el directorio de la imagen en la base de datos, que es enviada en forma de array de bytes al cliente. Una vez recibido, el cliente crea un archivo con este array de bytes, que es el que muestra al usuario como imagen.

- Una vez conseguí que el sistema cargase imágenes, empecé a implementar la interfaz del examen práctico en el lado del cliente.

- Una vez hecho esto, corregí algunos fallos en la aplicación que detectó mi compañero haciendo un testeo de la aplicación.

- Para terminar con la implementación, conseguí que el usuario administrador pudiera subir imágenes a la aplicación. Consideramos esta mejora muy importante, ya que de este modo el administrador puede estar renovando constantemente los recursos de la aplicación.

- Una vez desarrollada la aplicación, he de decir que la labor más importante y que más tiempo me ha llevado, ha sido la de investigación de todas las funciones del servicio web. Desde su funcionamiento y configuración, hasta todas las librerías usadas, han sido una ardua tarea de búsqueda y sobre todo de prueba y error que, aunque han retardado los tiempos de planificación, ha sido bastante gratificante conseguir que funcionase.

- Para finalizar, desarrollé los puntos de la memoria referentes al *web service*.

- Finalmente, he reunido todo el material digital y lo he grabado en un CD.