LYCEOS: Una herramienta para la gestión de la licitación electrónica en el ámbito de la Contratación Pública

Antonio Pablo Vázquez Freijomil
Grado en Ingeniería Informática

Gregorio Robles Martínez (Consultor)

9 de junio de 2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	LYCEOS: Una herramienta para la gestión de la licitación electrónica en el ámbito de la Contratación Pública
Nombre del autor:	Antonio Pablo Vázquez Freijomil
Nombre del consultor:	Gregorio Robles Martínez
Fecha de entrega (mm/aaaa):	06/2022
Área del Trabajo Final:	Desarrollo web
Titulación:	*Grado en Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Este trabajo trata la creación de una herramienta web que permita a las Administraciones Públicas del Estado Español disponer de un espacio para la recepción de ofertas en los expedientes de Contratación Pública que manejan.

Las disposiciones legales vigentes exigen que la recepción de ofertas se haga de forma electrónica, y las herramientas actualmente disponibles no ofrecen un marco adecuado en términos de eficiencia y usabilidad.

Así surge la idea de LYCEOS, una herramienta moderna, sencilla y fácil de usar, tanto para las Administraciones Públicas, como para las empresas licitadoras.

El proceso de creación de una licitación que pueda recibir ofertas nunca llevará más que un par de minutos por parte del empleado público encargado de esta tarea. A la empresa licitadora, la presentación de su oferta le resultará también sencilla y breve, teniendo toda la información necesaria a su alcance en una única pantalla. Y, por último, el proceso de apertura y valoración de las ofertas es rápido y lo suficientemente flexible como para que se pueda considerar este proyecto un éxito.

En la realización de la aplicación se ha priorizado la utilización de herramientas de código abierto y gratuitas, que, además de reducir costes, garantizan a los usuarios el acceso al código y la no existencia de código malintencionado en un proceso que debe ser escrupulosamente cuidadoso con el tratamiento de los datos y de la seguridad.

Abstract (in English, 250 words or less):

This work deals with the creation of a web tool that allows the Public Administrations of the Spanish State to have a space for the reception of offers in the Public Procurement files that they handle.

Current legal provisions require that the reception of offers must be done electronically, and the tools currently available do not offer an adequate framework in terms of efficiency and usability.

This is how the idea of LYCEOS arose: a modern, simple, and easy-to-use tool, both for Public Administrations and for bidding companies.

The process of creating a tender that can receive offers will never take more than a couple of minutes for the public employee in charge of this task. For the bidding company, the presentation of its offer will also be simple and brief, having all the necessary information at its fingertips on a single screen. And, finally, the process of opening and evaluating the offers is fast and flexible enough for this project to be considered a success.

In the development of the application, the use of free and open-source tools has been prioritized, which, in addition to reducing costs, guarantee users access to the code and the absence of malicious code in a process that must be scrupulously careful with data processing and security.

Palabras clave (entre 4 y 8):

Administración electrónica, Contratación pública, aplicación web, modernización

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La Administración Pública, cuando pretende contratar un servicio, un bien o una obra, debe, por cuestiones de transparencia pública y para evitar influencias y nepotismos, publicar estas licitaciones y permitir que cualquier operador económico interesado, y que cumpla los requisitos exigidos, pueda presentar una oferta.

En estos momentos, desde la entrada en vigor de la Ley 9/2017 [1], estas licitaciones deben ser electrónicas, a fin de permitir presentaciones mucho más sencillas y con un acceso más universal.

La realidad, sin embargo, es que el empleo de las herramientas electrónicas actualmente disponibles para realizar estas tareas no ha simplificado ni la presentación de ofertas ni la confección y preparación de las licitaciones. Mas bien, todo lo contrario; actualmente una gran mayoría de administraciones locales de tamaño pequeño tienen que recurrir a técnicos externos para poder realizar estas configuraciones, y el número de pequeñas empresas y autónomos que cometen errores en sus presentaciones por causa de estas herramientas ha crecido considerablemente.

Ilustración 1 Página principal de la PLACE

Además, la principal herramienta empleada para gestionar las licitaciones electrónicas (PLACE), que está proporcionada por el Estado, adolece de

múltiples defectos, entre los que podemos destacar su excesiva rigidez de uso, la dificultad y complejidad de manejo y la desactualización tecnológica. Y, sobre todo, la repercusión de las decisiones tecnológicas en la implementación al usuario final, al cual la organización interna de la aplicación debería resultarle transparente en cuanto a su manejo final.

En este contexto es donde se ubica nuestro trabajo, que pretende solucionar un problema real que el alumno conoce de primera mano, puesto que se dedica a formar a usuarios en el manejo de la PLACE y a dar asistencia técnica sobre la misma.

1.2 Objetivos del Trabajo

El objetivo principal es disponer de un producto mínimo entregable funcional de la herramienta que permita configurar una licitación por parte de los usuarios de la administración y recibir las ofertas de las empresas conforme a dicha configuración.

Los subobjetivos del Trabajo son los siguientes:

- Disponer de una aplicación web fácilmente escalable.
- Ofrecer una herramienta que cumpla con los requisitos legales en cuanto a seguridad, confidencialidad y fiabilidad.
- Permitir el uso de la herramienta desde cualquier navegador moderno.
- Diseñar una herramienta conforme a los principios de usabilidad en la interacción persona-ordenador, con un entorno de uso sencillo e intuitivo.
- Emplear tecnologías modernas, con una previsión de vida operativa larga.
- Requerir el mínimo de aplicaciones y/o recursos externos (aparte de un navegador web) a los usuarios.

1.3 Enfoque y método seguido

Para realizar este trabajo se ha considerado que la mejor estrategia de enfoque es la de realizar una aplicación nueva, que no se base en la que actualmente está funcionando. Esta nueva aplicación deberá centrarse en las acciones que efectivamente quieren realizar los usuarios, y no en cómo las tienen que realizar.

El enfoque de la aplicación existente, en el que hay una rigidez excesiva en el flujo de trabajo es uno de los principales inconvenientes y problemas de los que adolece. En el enfoque que proponemos la aplicación es mucho

más flexible, siempre dentro de las limitaciones que la Ley impone a este tipo de productos.

Para asegurar que el producto se ajuste a este requerimiento, la definición de requisitos por medio de las entrevistas a los usuarios es fundamental. De esta manera, podremos detectar los inconvenientes, principalmente en materia de usabilidad, y proponer mecanismos para solventarlos

1.4 Planificación del Trabajo

Basándonos en las fechas clave (hitos) que se nos han asignado para la realización de este TFG, se ha desarrollado la siguiente planificación temporal:

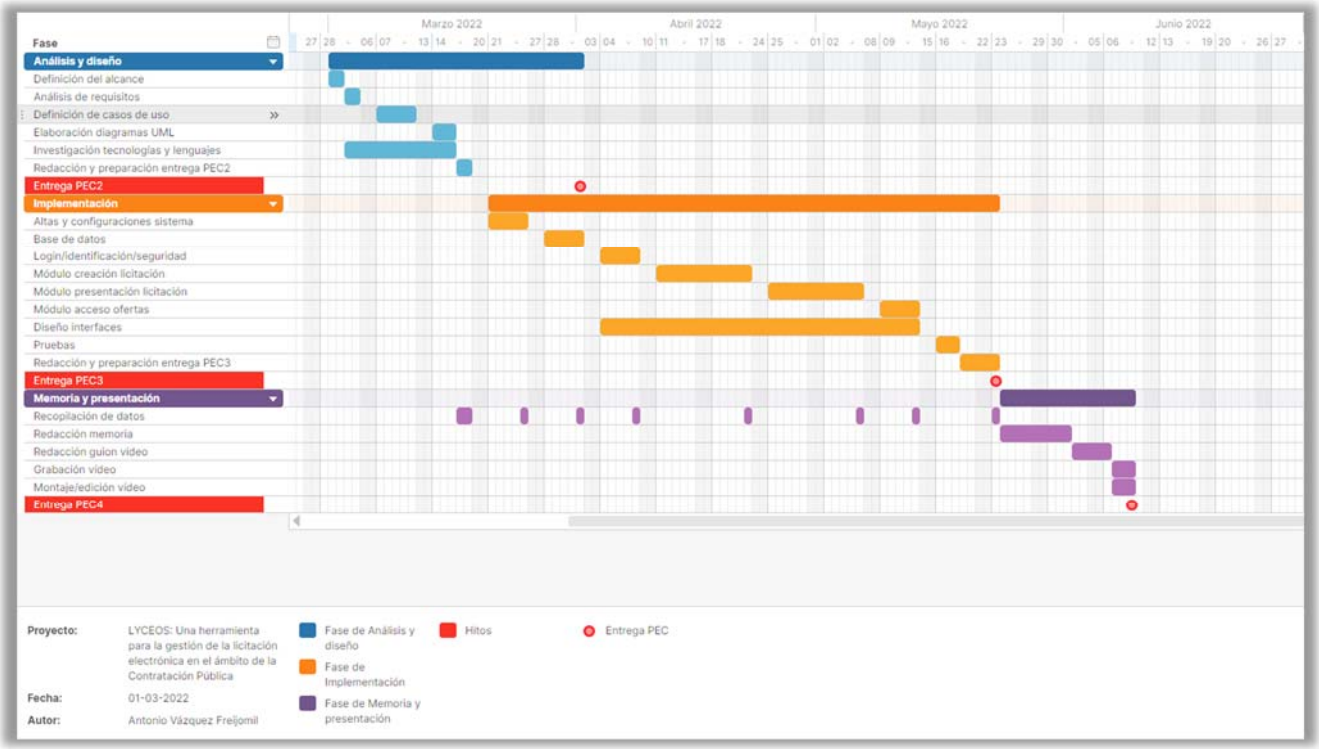

Ilustración 2 Diagrama de Gantt con la planificación temporal

En el diagrama de Gantt podemos observar cómo se han tenido en cuenta las tareas de recopilación de datos para la confección de la memoria final, repartiendo el coste temporal de estas tareas a lo largo del desarrollo del proyecto.

Los hitos de entrega de las correspondientes PEC's son fijos y nos han sido determinados al principio del proyecto.

1.5 Breve sumario de productos obtenidos

Como producto final de ha obtenido un producto mínimo entregable conforme con los requerimientos iniciales.

Así, el producto final es una aplicación construida conforme a los estándares web, que es sólida y estable, y que permite una fácil escalabilidad para poder incorporar nuevas funcionalidades.

La herramienta realizada permite crear licitaciones con la especificación de los documentos que se deben aportar y la agrupación de estos en “sobres”, que deben ser abiertos por orden. Asimismo, al crear una licitación se deben especificar los criterios por los que se evaluarán las ofertas con la indicación de las puntuaciones mínima y máxima que estas puedan obtener.

También permite a los licitadores presentar ofertas dentro de las configuraciones realizadas, e incluye una funcionalidad para realizar requerimientos a las empresas que presenten ofertas para que estas últimas contesten y aporten la documentación que consideren oportuna para atender a estos requerimientos.

Hay que destacar que durante el desarrollo del proyecto se modificó el alcance de este, debido a las limitaciones temporales existentes y la necesidad de mayor investigación al respecto, en cuanto al uso de los certificados digitales para el encriptado y firmado de las ofertas. En principio, se contaba con el empleo de herramientas que existen a tales efectos por parte de la Administración Pública, pero nos hemos encontrado con que tales herramientas, a pesar de tener una licencia abierta y pública, sólo se ofrecen a Administraciones Públicas directamente y a los desarrolladores que forman parte de estas.

1.6 Breve descripción de los otros capítulos de la memoria

En el resto de los capítulos de esta memoria abordaremos las decisiones tomadas en cuanto al empleo de tecnologías, las tareas de preparación y diseño realizadas, la ejecución e implementación del proyecto y las conclusiones obtenidas tras su realización.

Por último, ofreceremos las direcciones correspondientes para poder acceder al código del trabajo realizado y a la implementación de este en AWS.

2. Alcance y descripción del proyecto

Como ya se explicó en la breve descripción del proyecto, las Administraciones Públicas tienen la necesidad de contratar diferentes suministros, obras o servicios a empresas privadas. Como es evidente, en un sistema democrático y abierto, este debe ser un proceso transparente y limpio, en el que toda empresa que cumpla los requisitos exigidos pueda participar y presentar sus ofertas para que éstas sean evaluadas y puntuadas objetivamente, y, eventualmente, conseguir el objetivo de ser adjudicatarias del contrato.

Así, hay una normativa legal para regular este proceso y viene recogida, principalmente, en la Ley 9/2017, de ámbito nacional. En esta norma se definen una serie de interesados u operadores que se deben tener en cuenta.

Por un lado, tenemos al órgano de contratación (OC), que es el órgano de la administración pública que va a contratar los servicios de una empresa privada y que va a pagar por los mismos. Este órgano es el que define las características de la licitación concreta y las normas específicas por la que se regirá el proceso.

Por otro lado, y también dentro del ámbito de la administración, tenemos el órgano de asistencia, que está formado por las personas responsables de recibir las ofertas de las empresas interesadas, de verificar el cumplimiento de los requisitos de participación de éstas y de valorar conforme a las normas de la licitación las ofertas, estableciendo unas puntuaciones para cada oferta.

En último lugar, tenemos los operadores económicos o licitadores que son las empresas interesadas en el procedimiento de contratación y que presentan ofertas para ser tenidas en cuenta y valoradas.

El proceso que va a cubrir la aplicación será el que la ley denomina “procedimiento abierto”, en el que cualquier empresa interesada puede presentar una oferta en “sobres” cerrados. En este tipo de licitación, las licitaciones pueden tener criterios de valoración basados en juicios de valor (subjetivos) o criterios basados en fórmulas (objetivos). La documentación referida a cada tipo de criterio debe presentarse agrupada en “sobres” separados.

Además, habrá un sobre que contenga la documentación administrativa correspondiente a la empresa ofertante. Normalmente, y con la legislación vigente, esta documentación estará limitada a una declaración responsable por parte del representante de la empresa, pero queda siempre a la discreción de la Administración Pública convocante.

En base a esta contextualización se hace necesario definir concretamente el alcance de la aplicación a desarrollar, puesto que la casuística de la

normativa vigente y de las circunstancias que pueden concurrir en cada caso concreto es muy amplia, y el tiempo y los recursos disponibles para este Trabajo Fin de Grado son limitados. Así, se ha optado por la confección de un proyecto que, aunque limitado en su escenario de aplicación, emplee patrones y esquemas que permitan una posible futura ampliación de funcionalidades.

Por ello, y como se comentará posteriormente en el apartado de *Evaluación de riesgos*, se ha limitado el alcance del proyecto, no incluyéndose los procedimientos de firma de documentos y encriptado de sobres, debido al limitado acceso a recursos para trabajar con certificados electrónicos conforme a los estándares que marca la legislación vigente en España.

De esta forma, este proyecto contemplará únicamente los procedimientos de contratación llamados abiertos sin utilización de lotes, y restringidos a un único órgano de contratación.

3. Evaluación de riesgos

En la definición del proyecto, se ha realizado una evaluación de posibles riesgos que pudieran influir en el desarrollo de aquel, y una especificación de acciones que puedan mitigar su efecto.

Así, los riesgos detectados son los siguientes:

Riesgo 1		Falta de conocimiento de la tecnología empleada
	Descripción	Las tecnologías que se prevé emplear son novedosas y desconocidas para el alumno. Cabe la posibilidad de que el tiempo de adecuación a éstas no sea el adecuado
	Impacto	Alto
	Probabilidad	Baja/Media
	Acción de mitigación	Reajustar la temporalización de la implementación

Riesgo 2		Alcance del proyecto inadecuado
	Descripción	El trabajo fin de grado abarca 12 créditos, que se corresponden a 300 horas de trabajo. Es posible que se defina un alcance que sea muy corto o que exceda mucho el tiempo disponible para la realización
	Impacto	Alto
	Probabilidad	Media
	Acción de mitigación	Reajustar el alcance y redefinir los casos de uso

Riesgo 3		Complejidad tecnológica
	Descripción	Es posible que la integración de las tecnologías que se pretenden emplear no se pueda llevar a cabo en el tiempo previsto, o que surjan imprevistos durante el acoplamiento de los módulos
	Impacto	Alto
	Probabilidad	Media
	Acción de mitigación	Priorizar las funcionalidades que debe ofrecer la herramienta, para tener claro cuáles deben incorporarse en el entregable final, si fuera necesario sacrificar alguna

Riesgo 4		Enfermedad o problemas personales
	Descripción	El riesgo de contraer alguna enfermedad que incapacite al estudiante o que retrase el proyecto siempre está presente, pero en tiempos de pandemia es aún más evidente. Además, siempre pueden surgir problemas personales que impidan la total dedicación al proyecto
	Impacto	Alto
	Probabilidad	Baja/Media

	Acción de mitigación	Reajustar la temporalización, y, en caso extremo, redefinir el alcance y el producto mínimo entregable

Riesgo 5		Tecnología inadecuada
	Descripción	Se está seleccionando una tecnología que implica un riesgo de ventura, puesto que nunca ha sido utilizada por el autor. Es posible que la tecnología seleccionada no sea la solución más adecuada para la solución propuesta
	Impacto	Alto
	Probabilidad	Baja
	Acción de mitigación	Cambiar la tecnología seleccionada por una conocida y que sea asumible sin necesidad de aprendizaje de esta

A lo largo de las fases de diseño y, sobre todo, de implementación, se han materializado diversos riesgos de entre los que se habían previsto, y que han precisado de medidas correctoras para poder alcanzar el buen término de aquel.

Así, concretamente, se han sufrido incidencias que han afectado a los riesgos 3 y 4, complejidad tecnológica y circunstancias personales. De estos dos tipos de incidencias, el segundo ha sido coadyuvante al primero en el sentido de que por sí solo podría haber sido apenas significativo, pero unido a la incidencia de complejidad tecnológica ha sido un potenciador de la importancia de esta última.

La incidencia concreta del tipo de complejidad tecnológica que se ha producido es la poca información que existe respecto al uso de tecnologías para el empleo de certificados electrónicos para la identificación, firmado y encriptado, sobre todo en operaciones web en el lado cliente.

La tendencia en los últimos años es limitar el acceso de las aplicaciones web a los elementos del ordenador cliente, tanto hardware como software, a fin de evitar problemas de seguridad relacionados con software malintencionado. Esto impide, por ejemplo, la ejecución de applets Java, que eran las soluciones tradicionalmente empleadas en el lado cliente para la firma y la encriptación de documentos.

Actualmente, la administración pública posee una serie de servicios web disponibles para la verificación de firmas de certificados, así como una colección de software para implementar soluciones de este tipo. Desgraciadamente, estos recursos pertenecen a la red SARA, y, a pesar de estar licenciados bajo código abierto, no están efectivamente disponibles, salvo para organismos directamente dependientes de las Administraciones Públicas.

Por todo, ello, y en coordinación con el profesor asesor, se ha decidido limitar el alcance del proyecto, eliminando del mismo las funcionalidades

de firmado y cifrado de documentos y sobres. Estas funcionalidades no se eliminan de los requerimientos del proyecto global, sino únicamente del producto mínimo viable que se entregará con la memoria de este TFG, y así quedarían pendientes para una segunda versión de la aplicación.

Asimismo, dadas las especiales circunstancias de despliegue de la herramienta en el *free tier* de AWS, y que no se han previsto costes de implementación del proyecto, no se ha implementado la funcionalidad de antivirus de la documentación presentada. Por el mismo motivo no se ha implementado la funcionalidad de sellado de tiempo, puesto que se debe efectuar por un servicio externo que implica un coste de acceso al servicio.

4. Requisitos del proyecto

En base al contexto definido y a los requisitos recogidos, se definirán los siguientes roles en la aplicación:

- Responsable del órgano de contratación
- Usuario del órgano de contratación
- Usuario del órgano de asistencia
- Licitador

El sistema tendrá que permitir el acceso a cada uno de los usuarios de la Administración Pública, por un lado, basándose en su propia identidad personal, y a los licitadores como usuarios autorregistrados por otro, basados en un perfil de empresa.

Los requisitos del sistema se han agrupado por cada rol de participante en el proyecto. Así, la formulación de requisitos se expresa bajo la fórmula de “Como (rol) quiero (requisito)”.

- Como responsable del órgano de contratación quiero…
  - … poder dar de alta los usuarios del órgano de contratación
  - … que quede registro de todas las operaciones efectuadas (día / hora / usuario)
  - … que sólo los usuarios con dicho permiso puedan cifrar y descifrar ofertas
  - … que los usuarios puedan identificarse con usuario y contraseña o con certificado electrónico

- Como usuario del órgano de contratación quiero…
  - … poder configurar una licitación de tipo “abierto” con sus sobres correspondientes
  - … disponer de modelos ya configurados de tipos de documentos

- Como usuario del órgano de asistencia quiero…
  - … poder acceder a las ofertas de una licitación en bloque o individualmente
  - … poder calificar en cualquier momento a un usuario como admitido o excluido
  - … que las calificaciones y las puntuaciones de los criterios se puedan establecer de una forma cómoda y rápida

- Como licitador quiero…
  - … que la aplicación funcione sobre cualquier SO y navegador
  - … que el proceso de firma de documentos sea sencillo
  - … que se puedan aportar distintos documentos en un mismo apartado
  - … poder aportar documentos adicionales a los requeridos

5. Casos de uso

Teniendo en cuenta los requisitos y el alcance definido para el proyecto, se han formulado los siguientes casos de uso:

- Gestionar personal OC/OA
- Dar de alta una licitación
- Requerir documentación
- Proponer adjudicación
- Valorar ofertas
- Dar de alta un licitador
- Presentar una oferta
- Leer requerimiento
- Contestar requerimiento

El diagrama de casos de uso es el siguiente:

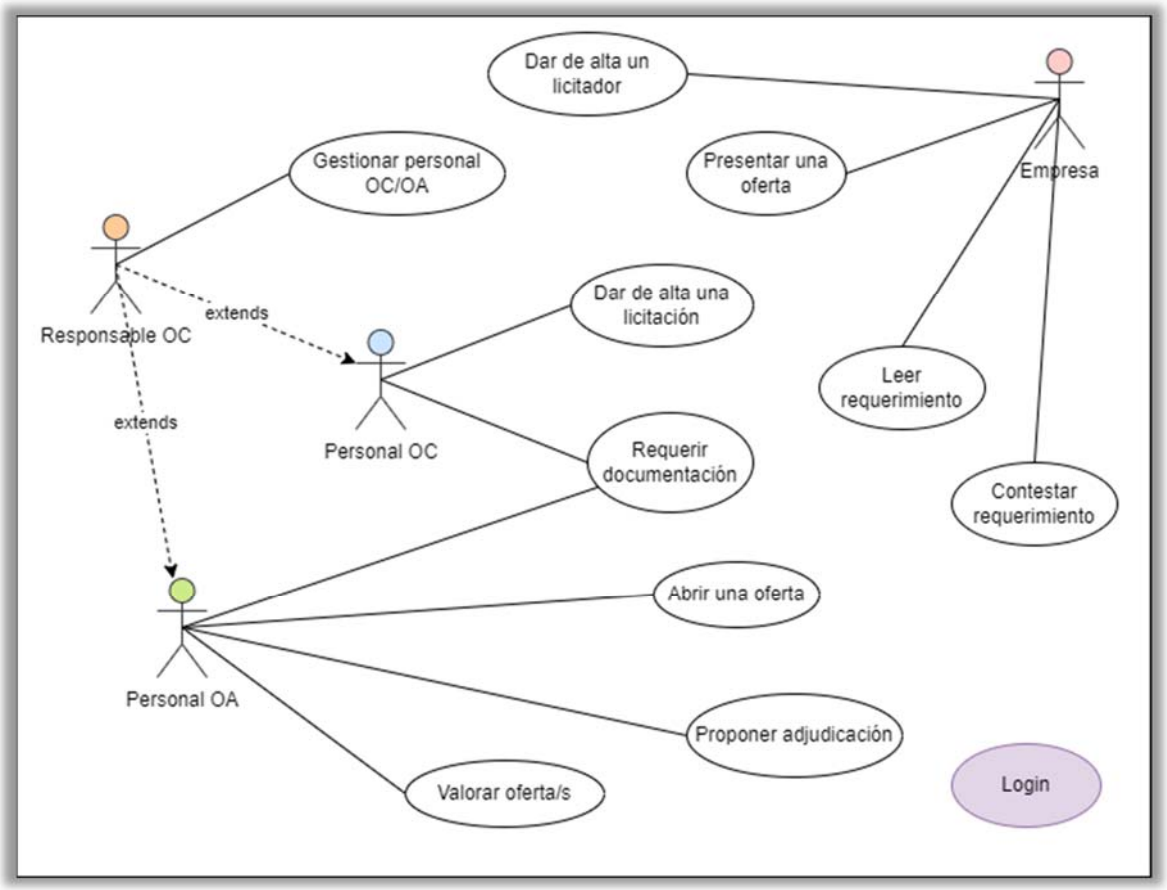

Ilustración 3 Diagrama de casos de uso

Como se puede ver en el diagrama de casos, hay además un caso de *Login*, que es común a todos los actores. Además, el actor *Responsable OC* incluye todas las funcionalidades de los actores con perfiles *Personal OC* y *Personal OA*.

La definición en detalle de los casos de uso planteados es la siguiente:

5.1 Caso de uso **Gestionar personal OC/OA**

Resumen de la funcionalidad: Dar de alta personal de los órganos de contratación o asistencia. No es un caso para ambos simultáneamente, sino el mismo caso para cada uno de los órganos.

Actores: Responsable de OC

Casos de uso relacionados: Ninguno

Precondición: El responsable de OC está identificado en el sistema.

Postcondición: Se ha modificado/dado de alta/borrado personal del órgano correspondiente

Proceso normal principal:

1. El sistema busca el personal del órgano y lo muestra en una tabla
2. El actor escoge modificar un usuario
3. El sistema muestra los datos personales del usuario
4. El actor modifica los datos que desea
5. El sistema modifica los datos
6. Volvemos al punto 1

Alternativas de proceso y excepciones:

2a. El usuario escoge dar de alta un usuario
2a1. El actor introduce un nuevo DNI
2a2. El sistema comprueba que no existe el DNI en la base de datos
2a2a. El DNI ya existe en la base de datos
2a2a1. El sistema muestra los datos del usuario
2a2a2. El actor modifica los datos del usuario
2a2a2a. El actor no desea hacer modificaciones
2a2a2a1. Volvemos al punto 2a
2a2a3. El sistema guarda las modificaciones
2a2a4. Volvemos al punto 1
2a3. El actor introduce los datos personales del usuario
2a3a. El actor no desea dar de alta al usuario
2a3a1. Volvemos al punto 2a
2a4. El sistema da de alta el usuario
2a5. Volvemos al punto 1
2b. El actor escoge dar de baja un usuario
2b1. El sistema desasigna el usuario
2b2. El sistema comprueba si el usuario tiene todavía asignaciones
2b2a. El usuario no tiene más asignaciones
2b2a1. El sistema borra el usuario de la base de datos
2b3. Volvemos al punto 1

5.2 Caso de uso **Admitir/Excluir un licitador**

Resumen de la funcionalidad: Se trata de calificar un licitador como admitido o excluido en una licitación

Actores: Personal de OA

Casos de uso relacionados: Ninguno

Precondición: El usuario con rol personal de OA está identificado en el sistema

Postcondición: Los licitadores seleccionados tienen un estado de Admitido o Excluido

Proceso normal principal:

1. El sistema muestra las ofertas recibidas
2. El actor selecciona las empresas que quiere calificar
3. El actor elige entre admitir o excluir los licitadores seleccionados

Alternativas de proceso y excepciones:

2a. El actor cancela la operación
    2a1. Finaliza el caso de uso
3a. El actor cancela la operación
    3a1. Finaliza el caso de uso

5.3 Caso de uso **Dar de alta una licitación**

Resumen de la funcionalidad: Se trata de dar de alta una licitación para que puedan presentarse ofertas y ser evaluadas

Actores: Personal de OC

Casos de uso relacionados: Ninguno

Precondición: El usuario con rol personal de OC está identificado en el sistema

Postcondición: La licitación está dada de alta y dispuesta para recibir ofertas

Proceso normal principal:

1. El actor introduce los datos de la licitación
2. El sistema guarda los datos de la licitación
3. Finaliza el caso de uso

Alternativas de proceso y excepciones:

2a. El actor cancela la introducción de datos
    2a1. Finaliza el caso de uso

5.4 Caso de uso **Requerir documentación**

Resumen de la funcionalidad: Se requiere a una o varias empresas para que, en relación con una oferta, presenten una documentación concreta

Actores: Personal de OC, Personal de OA

Casos de uso relacionados: Leer requerimiento, Contestar requerimiento

Precondición: La empresa requerida ha presentado una oferta a una licitación activa. El usuario con rol personal de OC o personal de OA está identificado en el sistema. La empresa requerida tiene un estado Admitido

Postcondición: La empresa recibe un requerimiento para presentar una documentación concreta

Proceso normal principal:

1. El actor accede a la licitación
2. El actor selecciona la/s empresa/s a requerir
3. El actor introduce la lista de documentación a requerir
4. El sistema registra el requerimiento
5. El sistema envía un correo electrónico a la empresa informando del requerimiento
6. El sistema genera un documento de constancia del requerimiento

7. El sistema ofrece el documento generado para su descarga
8. Finaliza el caso de uso

Alternativas de proceso y excepciones:

4a. El actor cancela la operación

4a1. Finaliza el caso de uso

5.5 Caso de uso **Abrir una oferta**

Resumen de la funcionalidad: El usuario abre un/varios sobre/s de ofertas recibidas dentro del marco de una licitación

Actores: Personal de OA

Casos de uso relacionados: Descifrar oferta, Admitir/excluir un licitador

Precondición: El OC ha dado de alta una licitación. Hay al menos una oferta presentada por empresas con estado Admitido. El usuario con perfil personal de OA está identificado en el sistema

Postcondición: La oferta está abierta y su documentación disponible para consultar

Proceso normal principal:

1. El sistema presenta una lista de las licitaciones en curso
2. El actor escoge una licitación
3. El sistema muestra los tipos de sobre de la licitación
4. El actor escoge un sobre
5. El sistema muestra las ofertas recibidas por empresas con estado Admitido
6. El actor escoge la/s oferta/s que desea abrir
7. El sistema registra el acto de descifrado y realiza sellado de tiempo
8. El sistema pasa un antivirus en los contenidos del sobre
9. Volvemos al punto 5

Alternativas de proceso y excepciones:

1a. No hay licitaciones en curso

1a1. El sistema muestra un mensaje en la lista vacía

1a2. Finaliza el caso de uso

4a. El sobre elegido requiere la apertura de otro sobre o la calificación

4a1. El sistema informa de esta circunstancia

4a2. El actor confirma la elección del sobre

4a2a. El actor no confirma la elección del sobre

4a2a1. Volvemos al paso 4

4a3. Volvemos al punto 5

5a. No hay ofertas recibidas

5a1. El sistema muestra un mensaje en la lista vacía

5a2. El caso de uso no continua

6a. Entre las ofertas recibidas hay alguna fuera de plazo o duplicada

6a1. El sistema informa y pide confirmación

6a2. El actor confirma

6a2a. El actor no confirma

6a2a1. Volvemos al punto 5

6a3. Volvemos al punto 7

5.6 Caso de uso **Valorar oferta/s**

Resumen de la funcionalidad: Se introducen las puntuaciones otorgadas a cada oferta en los diferentes criterios de valoración

Actores: Personal de OA

Casos de uso relacionados: Ninguno

Precondición: Los licitadores a puntuar tienen un estado Admitido

Postcondición: Se han asignado las puntuaciones deseadas

Proceso normal principal:

1. El actor selecciona una licitación
2. El sistema recupera las ofertas presentadas y las muestra
3. El actor selecciona la/s oferta/s a valorar e introduce la/s valoración/es
4. El sistema registra la/s valoración/es
5. Finaliza el caso de uso

Alternativas de proceso y excepciones:

2a. La licitación seleccionada no es valorable

2a1. El sistema informa de esta circunstancia

2a2. Volvemos al punto 1

3a. El actor cancela la operación

2a1. Finaliza el caso de uso

5.7 Caso de uso **Dar de alta un licitador**

Resumen de la funcionalidad: Una empresa quiere registrarse en la plataforma para poder presentar ofertas a licitaciones

Actores: Empresa

Casos de uso relacionados: Ninguno

Precondición: Ninguna

Postcondición: La empresa está dada de alta en la plataforma y puede presentar ofertas

Proceso normal principal:

1. El actor se identifica con un certificado digital
2. El actor introduce sus datos personales
3. El sistema da de alta los datos de la empresa de forma temporal
4. El sistema envía un correo para confirmar los datos
5. El actor confirma sus datos y su correo
6. El sistema da de alta los datos definitivos
7. Finaliza el caso de uso

Alternativas de proceso y excepciones*:*

5a. El actor no confirma el correo en 24h

5a1. El sistema borra los datos temporales

5a2. Finaliza el caso de uso

5.8 Caso de uso **Presentar una oferta**

Resumen de la funcionalidad: Una empresa presenta una oferta a una licitación

Actores: Empresa

Casos de uso relacionados: Cifrar oferta

Precondición: La licitación está creada y configurada

Postcondición: Se añade una oferta de la empresa en cuestión a la licitación

Proceso normal principal:

1. El actor accede a la licitación deseada
2. El sistema aporta al navegador del actor la clave pública de cifrado
3. El sistema muestra los documentos que tiene que aportar
4. El actor adjunta los documentos requeridos
5. El actor elige enviar la oferta
6. El sistema envía la oferta
7. El sistema recibe y guarda la oferta
8. El sistema guarda un log de la acción
9. El sistema genera un acuse de recibo y lo envía por correo electrónico al actor
10. El sistema informa del resultado y ofrece el acuse de recibo para descargar
11. Finaliza el caso de uso

Alternativas de proceso y excepciones:

4a. El actor ya tiene una oferta guardada que desea continuar
- 4a1. El sistema le pide el archivo con la oferta guardada
- 4a2. El actor aporta el archivo de oferta
- 4a3. El sistema abre el archivo y completa los datos de la oferta
- 4a4. Volvemos al punto 4

6a. El actor elige guardar la oferta
- 6a1. El sistema le permite guardar un archivo con toda la oferta
- 6a2. El sistema guarda la oferta
- 6a3. Volvemos al punto 3

5.9 Caso de uso **Leer requerimiento**

Resumen de la funcionalidad: Una empresa contesta a un requerimiento de documentación

Actores: Empresa

Casos de uso relacionados: Contestar requerimiento

Precondición: Se ha enviado previamente un requerimiento a la empresa y ésta ha recibido un correo

Postcondición: La empresa ha leído el requerimiento

Proceso normal principal:

1. El actor accede a sus notificaciones
2. El actor elige la notificación a la que desea acceder
3. El sistema comprueba que el requerimiento está sin leer
4. El sistema registra el acceso
5. El sistema muestra el requerimiento y permite descargarlo
6. Finaliza el caso de uso

Alternativas de proceso y excepciones:

4a. El requerimiento ya ha sido leído con anterioridad
- 4a1. Volvemos al punto 5

5.10 Caso de uso **Contestar requerimiento**

Resumen de la funcionalidad: Una empresa contesta a un requerimiento de documentación

Actores: Empresa

Casos de uso relacionados: Leer requerimiento
Precondición: Se ha leído el requerimiento que se pretende contestar
Postcondición: La empresa contesta al requerimiento
Proceso normal principal:

1. El actor accede a sus notificaciones
2. El actor elige la notificación a la que desea responder
3. El sistema comprueba que el requerimiento ya ha sido leído
4. El sistema muestra la lista de documentos requeridos
5. El actor adjunta los documentos requeridos
6. El actor elige enviar la respuesta
7. El sistema envía la respuesta
8. El sistema recibe y guarda la respuesta
9. El sistema guarda un log de la acción
10. El sistema genera un acuse de recibo y lo envía por correo electrónico al actor
11. El sistema informa del resultado y ofrece el acuse de recibo para descargar
12. Finaliza el caso de uso

Alternativas de proceso y excepciones:

4a. El requerimiento no ha sido leído
    4a1. Se pasa al caso de uso Leer requerimiento
5a. El actor ya tiene una respuesta guardada que desea continuar
    5a1. El sistema le pide el archivo con la respuesta guardada
    5a2. El actor aporta el archivo de respuesta
    5a3. El sistema abre el archivo y completa los datos de la respuesta
    5a4. Volvemos al punto 5
7a. El actor elige guardar la oferta
    7a1. El sistema le permite guardar un archivo con toda la respuesta
7a2. El sistema guarda la respuesta
7a3. Volvemos al punto 4

6. Modelo UML

Para poder realizar un diseño correcto, se ha elaborado un diagrama UML, que muestre las clases necesarias en la aplicación y las relaciones entre estas.

De esta forma, partiendo de los requisitos y de la definición de funcionalidades, hemos obtenido el siguiente diagrama UML:

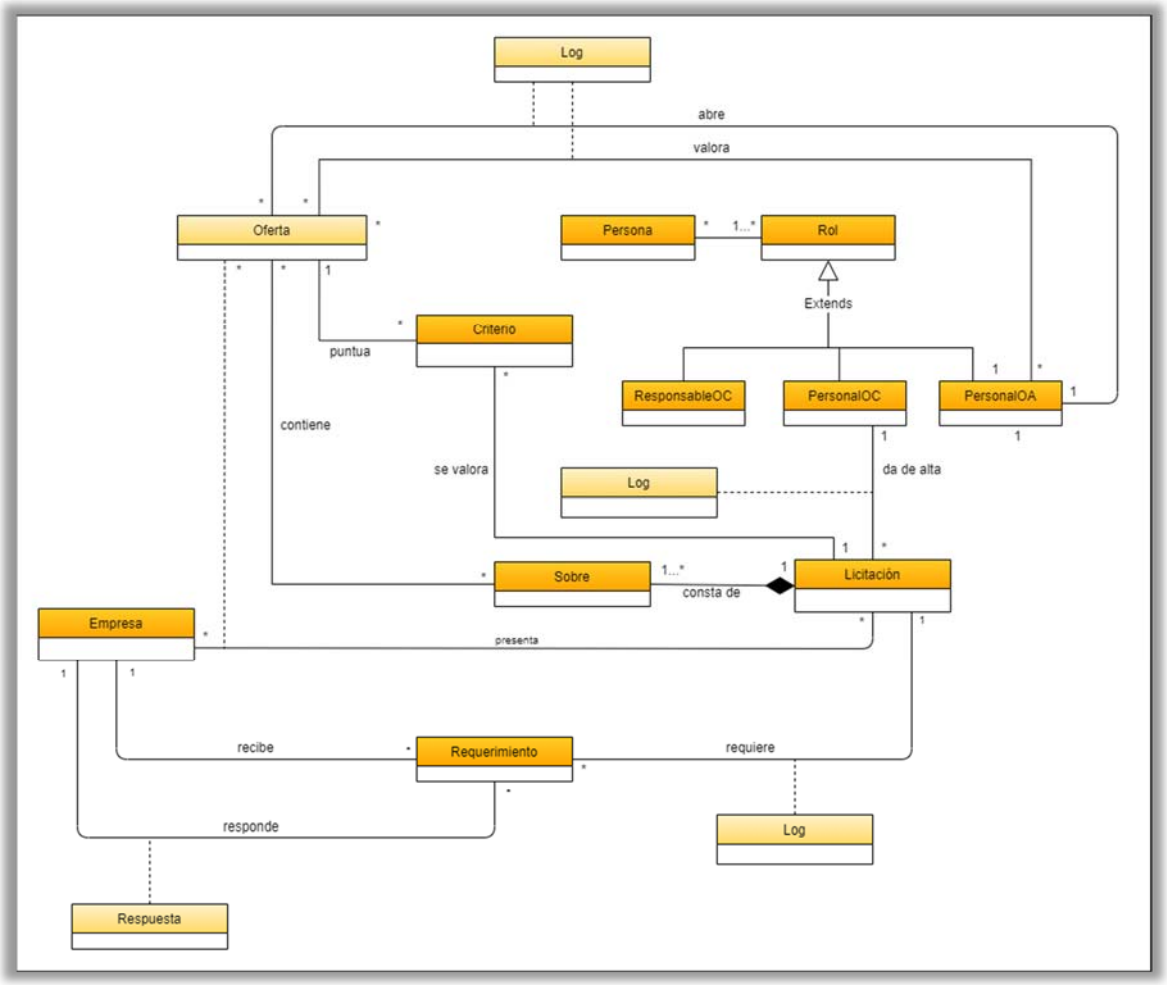

Ilustración 4 Diagrama UML

En el diagrama UML podemos ver cómo se deposita la responsabilidad de decidir la capacidad de realizar una acción en el rol que el usuario tenga asociado. Un usuario podrá tener uno o más roles asignados, y la implantación se realizará mediante una clase de roles con los registros correspondientes a cada uno de los posibles roles asignables. De esta forma, nos aseguramos la posibilidad de escalar el número de roles diferentes de una forma más sencilla que implantando este control directamente en el código.

Como se puede observar, la clase Log aparece triplicada en el esquema a efectos de aportar mayor claridad en éste, pero se trata de la misma entidad.

7. Punto de vista de la información

7.1 Modelo conceptual

Para confeccionar el modelo conceptual de datos, hemos partido del diagrama UML, para poder obtener un diagrama ER en el cual se define la estructura de datos relacional subyacente a la aplicación.

Además de las clases presentes en el diagrama UML, se ha añadido la entidad *Fichero*, que será la encargada de guardar los datos de almacenamiento físico del fichero concreto. De esta forma, desacoplamos las características del almacenamiento concreto del fichero de la estructura de la licitación, favorecemos la escalabilidad en el caso de que se pretendan reutilizar ficheros y podemos guardar datos concretos sobre el fichero de datos, como puede ser el nombre original (que habitualmente no se emplea para el almacenamiento por cuestiones de seguridad).

Esta entidad *Fichero* mantiene una relación polimórfica con las entidades *DocumentoOferta*, *DocumentoModelo* y *DocRespRequerimiento*. Las entidades polimórficas se pueden definir de forma muy sencilla mediante la creación de un atributo adicional en la entidad, en la cual se almacena la clase con la que está relacionada el elemento concreto al que se refiere la clave foránea. Como veremos más adelante, en la implementación del trabajo, las relaciones polimórficas [2] están integradas en el *ORM Eloquent*, que emplea *Laravel* por defecto.

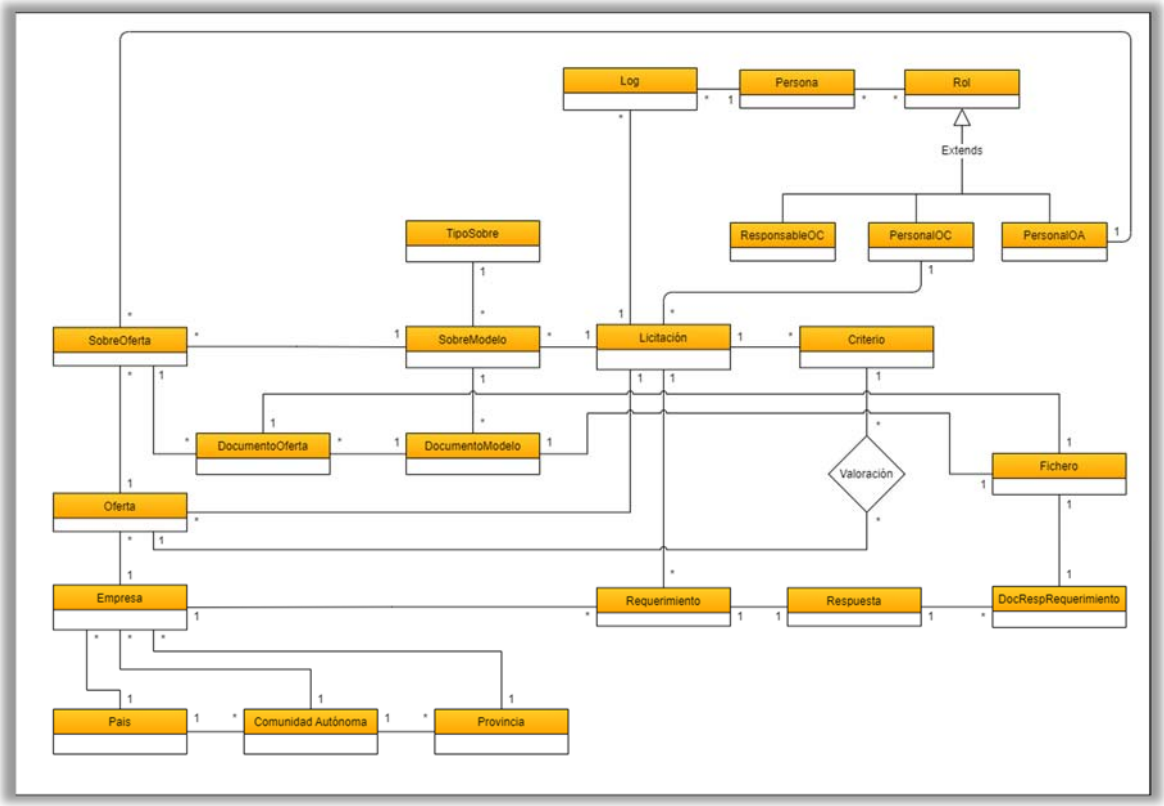

Ilustración 5 Modelo conceptual

7.2 Modelo lógico

Una vez obtenido el modelo conceptual de los datos, podemos generar el modelo lógico, en el que reflejamos los atributos de cada entidad, y definimos las claves primarias (atributos subrayados).

Además, el modelo lógico también incorpora información acerca de lo atributos que pueden ser nulos (aquellos que no aparecen en negrita.

Con estas características, obtenemos el siguiente modelo lógico:

Persona (**id**, **dni**, **nombre**, **correoE**, **cargo**, **fechaAlta,** fechaBaja)

Rol (**id**, **nombre**, **descripcion**)
{dni} is relation ManyToOne to Persona

PersonaRol (**persona**, **rol**)
{persona} is relation ManyToOne to Persona
{rol} is relation ManyToOne to Rol

Empresa (**id**, **cif**, **razon**, **direccion**, **cp**, **poblacion**, **provincia**, **ccaa**, **pais**)

Licitacion (**id**, **expediente**, **fechaInicio**, **fechaFin**, **objeto**, adicional, **personalAlta**)
{personalAlta} is relation ManyToOne to PersonalOC

Oferta (**id**, **fecha**, adicional, **licitacion**, **empresa**, **estado**, **selloTiempo**)
{licitacion} is relation ManyToOne to Licitacion
{empresa} is relation ManyToOne to Empresa

Criterio (**id**, **puntuacionMinima**, **puntuacionMaxima**, **licitacion**)
{licitacion} is relation ManyToOne to Licitacion

Valoracion (**id**, **fecha**, **puntuacion**, **criterio**, **oferta**, **personalValoracion**)
{criterio} is relation ManyToOne to Criterio
{oferta} is relation ManyToOne to Oferta
{personalValoracion} is relation ManyToOne to PersonalOA

Requerimiento (**id**, **fecha**, **contenido**, adicional, receptor, **fechaLimite**, fechaRecepcion, **licitacion**, **empresa**)
{licitacion} is relation ManyToOne to Licitacion
{empresa} is relation ManyToOne to Empresa

SobreModelo (**id**, **nombre**, **tipo**, adicional, **licitacion**)
{tipo} is relation ManyToOne to TipoSobre
{licitacion} is relation ManyToOne to Licitacion

SobreOferta (**id**, **sobreModelo**, personaApertura, personaDescifrado, fechaApertura, fechaDescifrado, **oferta**)
{sobreModelo} is relation ManyToOne to SobreModelo
{personaApertura} is relation ManyToOne to PersonalOA
{personaDescifrado} is relation ManyToOne to PersonalOA
{oferta} is relation ManyToOne to Oferta

Respuesta (**id**, **respuesta**, **fecha**, **requerimiento**)
{requerimiento} is relation OneToOne to Requerimiento

Log (**id**, **persona**, **fecha**, **acción**, **url**, **method**, **ip**, **agent**, table, record)
{persona} is relation ManyToOne to Persona

TipoSobre (**id**, **nombre**, **vaCifrado**, **vaFirmado**)

DocumentoModelo (**id**, **nombre**, **descripcion**, **hayFirma**, **hayPlantilla**, archivoPlantilla, **esObligatorio, sobre**)
{sobre} is relation ManyToOne to SobreModelo

DocumentoOferta (**id**, **sobre**, **modelo**, **archivoOferta**)
{sobre} is relation ManyToOne to SobreOferta
{modelo} is relation ManyToOne to DocumentoModelo

DocRespRequerimiento (**id**, **respuesta**, **archivoRespuesta**)
{respuesta} is relation ManyToOne to Respuesta

Fichero (**id**, **ruta**, **nombreOriginal**, **documento_id**, **documento_type**)
{documento_id, documento_type} is relation MorphOneToOne to [DocumentoOferta, DocumentoModelo, DocRespRequerimiento]

Pais (**id**, **nombre_pais**)

Comunidad (**id**, **nombre**, **pais_id**)
{pais_id} is relation ManyToOne to Pais

Provincia (**id**, **nombre**, **codigo**, **ccaa_id**)
{ccaa_is} is relation ManyToOne to Comunidad

Como se puede observar, la relación múltiple entre *Persona* y *Rol* da lugar a una nueva entidad *PersonaRol* para poder satisfacer esta multiplicidad.

8. Patrones y decisiones de diseño

Se ha decidido aplicar un patrón *MVC*, en el que se separarán por un lado el Modelo, encargado de realizar los accesos a los datos, por otro la Vista, encargada de generar las pantallas y recoger la interacción del usuario, y por último el Controlador, encargado de gestionar la lógica de negocio de la aplicación.

De esta forma, se desacoplan los diferentes niveles de acceso y se gana en escalabilidad y capacidad de mantenimiento, mejora y ampliación de funcionalidades de la aplicación.

Para la implementación se ha optado por desarrollar la aplicación en lenguaje PHP para toda la parte de *backend* y JavaScript para la parte de *frontend*, por el conocimiento que ya se dispone de estos lenguajes.

En lo que se refiere al almacenamiento de datos, se ha optado por una solución de motor de bases de datos relacional, basado en *SQL*, y de uso gratuito y abierto, de acuerdo con las decisiones previas que ya se habían considerado.

9. Punto de vista de la computación

Desde el punto de vista de la computación, dispondremos de cuatro componentes claramente diferenciados, basados, principalmente, en los roles de los usuarios que los emplean. De esta forma, tendremos los siguientes componentes:

- Administración
- Configuración
- Presentación
- Evaluación

Cada uno de estos componentes deberá ofrecer las interfaces necesarias para cada uno de los métodos que se relacionan a continuación y que están asociados a los diferentes casos de uso que se han detectado:

- Administración:
  - loginAP (dni: String, password: String)
  - loginLic (cif String, password: String)
  - registerLic(empresa: Empresa)
  - addUser(user: Persona, type: String)
  - removeUser(user: Persona, type: String)
  - modifyUser(user: Persona, type: String)

- Configuración:
  - addContract(contract: Licitacion, user, UsuarioOC)

- Presentación:
  - registerOffer(offer: Oferta)
  - readRequirement(requirement: Requerimiento, licitator: Empresa)
  - answerRequirement(response: Respuesta)

- Evaluación:
  - qualifyOffer(offer: Oferta, state: String)
  - addRequirement(requirement: Requerimiento)
  - openEnvelope(envelope: SobreOferta, user: UsuarioOA)
  - evaluateOffer(evaluation: Valoracion; punctuation: real)

10. Tecnologías y recursos empleados

A nivel de infraestructura hardware, se ha optado por realizar una instalación final de la aplicación en la capa gratuita (*free tier*) de AWS. Esta decisión se ha tomado por varios motivos:

En primer lugar, y bastante importante, el carácter gratuito de esta capa. En este momento, se trata de la realización de un trabajo fin de grado, y no parece muy conveniente emplear recursos de pago que supongan una carga añadida al mismo o que puedan dificultar la implementación y confección del trabajo.

Por otra parte, los servicios de AWS son fácilmente escalables si se decidiera pasar de este trabajo a un entorno de producción. Esta escalabilidad se puede realizar tanto en número de instancias de la aplicación ejecutadas simultáneamente con todo lo que ello implica (balanceadores de carga y otras estructuras) gestionadas directamente por AWS, como en la ampliación de funcionalidades de la aplicación.

En tercer lugar, los servicios de AWS se ofrecen en la nube, y constituyen un entorno *IaaS* que nos permitirá desacoplar la aplicación propiamente dicha del entorno en el que se ejecuta.

Por último, el conjunto de servicios de AWS constituye un entorno moderno y actualizado, con continuas revisiones e implementaciones de las tecnologías más modernas y punteras.

En cuanto a lo que se refiere a tecnologías software, como ya se ha dicho anteriormente se ha optado por la programación en lenguaje PHP para el *backend*, y en lenguaje JavaScript para el *frontend*.

Partiendo de estas premisas, se han evaluado las diferentes soluciones disponibles, con la intención de poder emplear los *frameworks* que nos permitan realizar una programación sencilla, a la vez que segura y robusta.

Con estas condiciones, se ha optado por emplear el *framework Laravel* para la programación del lado del servidor. Los motivos para tomar esta decisión han sido, principalmente:

- Solución segura y basada en buenas prácticas
- Rapidez de ejecución
- Integración de diferentes módulos para soluciones personalizadas
- Amplia comunidad de soporte
- Código abierto

Entre los diferentes módulos que incorpora *Laravel*, destaca *Eloquent*, un ORM que nos permite acceder de forma transparente a la base de datos elegida. *Eloquent* cuenta con drivers para la mayoría de los gestores de bases de datos relacionales, y proporciona un acceso mediante clases de

PHP, lo que simplifica mucho el trabajo de programación de los modelos de datos.

Para la gestión de la autentificación de usuarios se ha decidido emplear *Laravel Breeze*. *Breeze* es un paquete de *Laravel* que incorpora los componentes y vistas necesarios para gestionar la autenticación de usuarios. Posteriormente veremos como se ha tenido que retocar la instalación de *Breeze* para poder admitir el doble sistema de autenticación que pretendemos en nuestra aplicación.

En lo que respecta a las vistas, se ha empleado el motor de plantillas por defecto de *Laravel*, que es *Blade*. Por medio de este sistema de plantillas, podemos disponer de un sistema de herencia de plantillas HTML, en el que además se puede incrustar directamente código PHP en caso de ser necesario, y proporciona un conjunto de sentencias propias para implementar estructuras de control (bucles, alternativas…) y acceso a variables.

En cuanto a los aspectos meramente visuales, se ha optado por emplear *AdminLTE* [3], que es un panel de administración para Bootstrap, desarrollado y mantenido por el estudio *Almsaseed*. Es una solución modular, de código abierto, que permite, de manera sencilla, configurar y construir plantillas de datos. De forma estándar, incluye diferentes soluciones como ventanas de notificaciones, tablas de datos y diferentes widgets provenientes de otras soluciones de código abierto. Para poder incorporarlo en *Laravel* se ha empleado una solución de integración llamada *Laravel-AdminLTE* [4] y que permite integrar los bloques y módulos de *AdminLTE* con el motor de plantillas de *Blade*.

El desarrollo de la aplicación se ha llevado a cabo en local en el ordenador del estudiante, mediante la instalación típica de *Laravel*, a través de *Docker* y con el servicio *sail*.

11. Desarrollo e implementación del proyecto

11.1 Componentes de autenticación (*userauth* y *empresaauth*)

Como ya se ha comentado, se ha optado por emplear el paquete *Breeze* [5] de *Laravel*, que proporciona de una manera muy sencilla las tareas básicas de autenticación de usuarios, registro, confirmación de cuentas de correo electrónico y reinicio de contraseñas.

En nuestro caso, vamos a emplear dos instancias de *Breeze*, o *guards*, como las llama *Laravel*. Una se utilizará para identificar a los usuarios de la Administración Pública, que accederán mediante su número de DNI y una contraseña, y la otra para identificar a los usuarios de tipo empresa, que acceden mediante su correo electrónico verificado y una contraseña. Mientras los usuarios de tipo empresa se pueden autorregistrar en la aplicación, los usuarios de la Administración Pública han de ser dados de alta por un administrador, de la aplicación o del órgano.

Para poder evitar que los usuarios de la Administración Pública se puedan autorregistrar, eliminamos la ruta creada a esos efectos, como se puede ver en la Ilustración 6, que en comparación con la Ilustración 8, correspondiente al *guard* de empresas, muestra que la ruta *register* no existe en el primer caso.

```
Route::group(['middleware' => ['guest:admon'], 'prefix' => 'usuario', 'as' => 'user.'], function() {
    Route::get('login', [AuthenticatedSessionController::class, 'create'])
                ->name('login');
    Route::post('login', [AuthenticatedSessionController::class, 'store']);
    Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
                ->name('password.request');
    Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
                ->name('password.email');
    Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
                ->name('password.reset');
    Route::post('reset-password', [NewPasswordController::class, 'store'])
                ->name('password.update');
});
```

Ilustración 6 Rutas públicas del guard de usuario de AAPP

```php
Route::group(['middleware' => ['auth:admon', 'permisos'], 'prefix' => 'usuario', 'as' => 'user.'], function() {
    Route::get('verify-email', [EmailVerificationPromptController::class, '__invoke'])
                ->name('verification.notice');
    Route::get('force-change-password', [ForceChangePasswordController::class, '__invoke'])
                ->name('password.forcechange');
    Route::post('force-change-password', [ForceChangePasswordController::class, 'store'])
                ->name('password.storeforcechange');
    Route::get('change-password', [ChangePasswordController::class, '__invoke'])
                ->name('password.change');
    Route::post('change-password', [ChangePasswordController::class, 'store'])
                ->name('password.storechange');
    Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
                ->name('password.confirm');
    Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);
    Route::get('logout', [AuthenticatedSessionController::class, 'destroy'])
                ->name('logout');
    Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
                ->name('logout');
});
```

Ilustración 7 Rutas privadas del guard de usuario de AAPP

```php
Route::group(['middleware' => ['guest:empresa'], 'prefix' => 'empresa', 'as' => 'empresa.'], function() {
    Route::get('register', [RegisteredUserController::class, 'create'])
                ->name('register');
    Route::post('register', [RegisteredUserController::class, 'store']);
    Route::get('login', [AuthenticatedSessionController::class, 'create'])
                ->name('login');
    Route::post('login', [AuthenticatedSessionController::class, 'store']);
    Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
                ->name('password.request');
    Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
                ->name('password.email');
    Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
                ->name('password.reset');
    Route::post('reset-password', [NewPasswordController::class, 'store'])
                ->name('password.update');
});
```

Ilustración 8 Rutas públicas del guard empresa

```php
Route::middleware('auth:empresa')->group(function () {
    Route::get('verify-email', [EmailVerificationPromptController::class, '__invoke'])
                ->name('verification.notice');
    Route::get('verify-email/{id}/{hash}', [VerifyEmailController::class, '__invoke'])
                ->middleware(['signed', 'throttle:6,1'])
                ->name('verification.verify');
    Route::post('email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
                ->middleware('throttle:6,1')
                ->name('verification.send');
});

Route::group(['middleware' => ['auth:empresa'], 'prefix' => 'empresa', 'as' => 'empresa.'], function() {
    Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
                ->name('password.confirm');
    Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);
    Route::match(['get', 'post'], 'logout', [AuthenticatedSessionController::class, 'destroy'])
                ->name('logout');
});
```

Ilustración 9 Rutas privadas del guard de empresas

Es de destacar que, en caso de que un usuario de la Administración Pública olvide su contraseña puede solicitar de un administrador que le suministre una nueva. Sin embargo, como las contraseñas deben ser personales, el hecho de que otra persona le asigne una nueva contraseña incumple este principio de privacidad. Para evitar este fallo de seguridad,

cuando un administrador cambie la contraseña de un usuario, éste debe cambiarla en su primer acceso a la herramienta, y no podrá acceder a ninguna funcionalidad hasta que no realice este cambio de contraseña.

11.2 Rutas

Hemos dividido el conjunto de rutas definidas para toda la aplicación en diferentes archivos, para ofrecer una mayor claridad y visibilidad al agrupar las rutas por los componentes a los que corresponden. Esta distribución la podemos ver en la siguiente captura de pantalla:

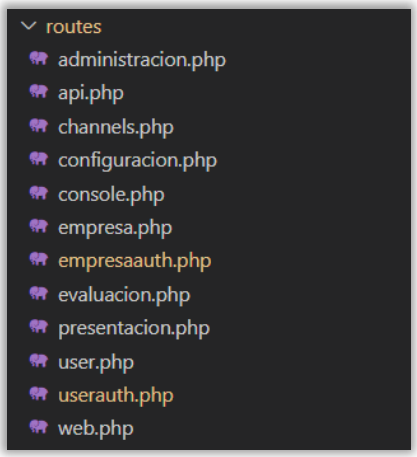

Ilustración 10 Ficheros con las definiciones de rutas

Para una mayor legibilidad de las rutas en cada fichero, se han agrupado las rutas que comparten *middleware*, controlador, *path* y prefijos de nombre, de forma que no haya que repetir todos estos datos en cada ruta. Esta característica se puede observar en las Ilustraciones de la 6 a la 9.

11.3 Middleware

También es importante destacar los componentes de *middleware* de *Laravel*. A través del *middleware*, el *framework* nos ofrece unas clases que actúan como intermediarias antes de la ejecución de los controladores.

Como se observa en las ilustraciones de la 6 a la 9, las rutas invocan unas clases de middleware u otras, según interese en cada caso. Concretamente las que podemos ver en esas capturas corresponden a la clase *auth*, que es la encargada de cargar los datos del usuario autenticado en cada momento.

Nosotros hemos creado una clase de *middleware* propia, llamada *Permisos*, cuya función es cargar los permisos que tiene el usuario autenticado, según el rol que posea. Estos permisos se habilitan en una clase específica de *Laravel*, llamada *Gate*, que luego puede ser invocada mediante un *helper* y que nos permite, entre otras cosas, mostrar las opciones de menú a las que el usuario tiene acceso, o impedir la ejecución

de determinadas partes del código. En la siguiente captura podemos ver parte de la declaración de esta clase.

```php
class Permisos
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        if (Auth::user() instanceof User) {
            Gate::define('admon', function() {
                return true;
            });
            Gate::define('gestusers', function(User $user) {
                return $user->is('Admin') || $user->is('ResponsableOC');
            });
            Gate::define('gestcontracts', function(User $user) {
                return $user->is('Admin') || $user->is('ResponsableOC') || $user->is('PersonalOC');
            });
```

Ilustración 11 Fragmento de la declaración del middleware Permisos

11.4 Controladores

En lo que se refiere a los controladores, se ha creado uno por cada componente definido en la aplicación. En cada uno de estos controladores se ha incluido una función por cada ruta del componente, en la cual se ha codificado la lógica de negocio necesaria para procesar tanto las entradas como las salidas de dicha ruta.

Como se puede ver en la Ilustración 12, que muestra parte del código del controlador del componente Configuración, la configuración típica de una función de un controlador incorpora un control de los permisos de ejecución, el acceso al modelo de datos y una *renderización* de una vista con los datos necesarios.

```php
public function removeContract(Request $request) {
    if (Gate::denies('gestcontracts')) {
        abort(403);
    }

    $licitacion = Licitacion::find($request->id);
    $expediente = $licitacion->expediente;
    $licitacion->delete();

    Session::flash('alert-message', 'Se ha eliminado correctamente la licitación ' . $expediente);
    Session::flash('alert-theme', 'success');

    Log::addToLog('Licitación borrada (expediente ' . $licitacion->expediente . ')', ['table' => 'licitacions', 'record' => $request->id]);

    return redirect()->route('configuracion.listcontracts');
}

public function viewContract(Request $request) {
    $licitacion = Licitacion::find($request->id);
    $sobres = $licitacion->sobreModelos()->get();
    $criterios = $licitacion->criterios()->get();

    return view('configuracion.viewcontract', [
        'licitacion'    => $licitacion,
        'sobres'        => $sobres,
        'criterios'     => $criterios,
    ]);
}
```

Ilustración 12 Fragmento del código del controlador Configuración

11.5 Modelos

Como ya se ha comentado en otros apartados, *Laravel* nos ofrece el ORM *Eloquent* para poder acceder a la base de datos mediante objetos de PHP. *Eloquent* proporciona también un constructor de consultas, que, mediante el uso de métodos en las clases de los modelos, nos permite realizar consultas y actualizaciones de un modo sencillo, y desacoplando el gestor de bases de datos elegido de nuestra aplicación.

Los modelos en *Eloquent* se suelen corresponder con las entidades que tenemos definidas en nuestro modelo conceptual, y con las tablas de la base de datos.

Eloquent emplea una notación específica que permite, cuando se sigue, simplificar extremadamente la configuración de los modelos. Así, por ejemplo, las claves foráneas en las tablas se deben llamar *nombredetabla_id*, y *Eloquent* la reconocerá automáticamente, estableciendo la correspondiente relación entre las entidades.

```php
public function sobresAbiertos() {
    return $this->hasMany(SobreOferta::class, 'personaApertura');
}

public function sobresDescifrados() {
    return $this->hasMany(SobreOferta::class, 'personaDescifrado');
}

public function acciones() {
    return $this->hasMany(Log::class);
}

public function is($rolename) {
    foreach ($this->roles()->get() as $role) {
        if ($role->nombre == $rolename)
            return true;
    }
    return false;
}
```

Ilustración 13 Fragmento del código del modelo User

Además, cada clase de modelo puede incorporar las funciones específicas necesarias para poder acceder a los datos deseados en cada momento, como se puede ver en la ilustración superior

Nosotros hemos hecho uso de todas estas capacidades que permiten, además de simplificar, generar un código mucho más robusto y con menos errores.

En la Ilustración 13 también podemos observar con qué facilidad se pueden obtener los sobres que han sido abiertos por un usuario concreto, simplemente accediendo al método *sobresAbiertos*, que nos devolverá una colección Eloquent (una colección es una estructura de datos propia de *Laravel* que extiende las capacidades de los *arrays* para hacerlos más fácilmente accesibles) con los elementos de la tabla correspondiente al modelo *sobreOferta* que ese usuario haya abierto. El código de la función sólo precisa la instrucción *return* para devolver los registros relacionados.

También podemos ver en la función *is* lo sencillo que resulta crear funciones personalizadas para poder acceder a los datos subyacentes.

11.6 Vistas

Como ya se ha dicho, las vistas de la aplicación se han implementado con el sistema de gestión de plantillas *Blade*. Éste nos proporciona un motor de plantillas sencillo, pero muy potente. *Blade* no sólo puede generar vistas en formato HTML, sino también en formato JSON o XML.

En nuestro caso, hemos empleado el conjunto de plantillas AdminLTE para confeccionar nuestras vistas. Sin embargo, hemos mantenido las vistas generadas por *Breeze* para el registro, login y confirmación de contraseñas de los usuarios.

Ilustración 14 Pantalla de login usuario AAPP

El aspecto general de las vistas generadas con este conjunto de plantillas se puede ver en la Ilustración 15, y en ella podemos comprobar como se muestra un menú lateral adecuado para cada tipo de usuario, y cómo se han integrado los mensajes en la parte superior de la pantalla (en este caso, se trata de una captura de la pantalla tras la configuración de una nueva licitación).

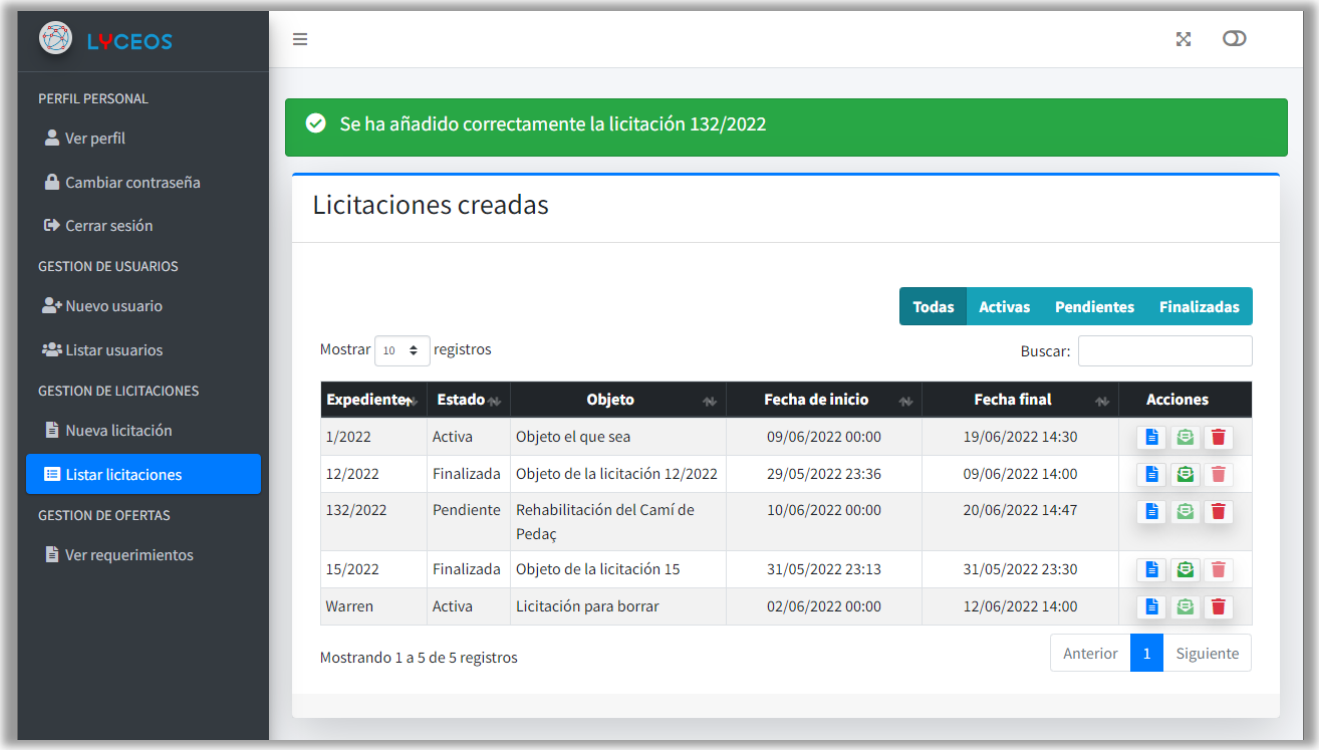

Ilustración 15 Pantalla de listado de licitaciónes (AAPP)

La vista de configuración de licitaciones merece una mención especial. Esto se debe a que hace un uso intensivo de *JQuery* y *JavaScript* para ofrecer al usuario una experiencia cómoda e intuitiva.

La problemática que presentaba esta vista tiene que ver con cuál es el propio proceso de creación de una licitación. Por una parte, todas las licitaciones tienen que contener al menos un sobre, pero éste puede ser de cualquier tipo, y no hay un límite en el número de ellos que se pueden crear. Además, cada sobre debe tener al menos un “hueco” para poder introducir documentos en él, pero de nuevo no hay límite en el número de documentos exigidos en cada sobre. Y, por último, una licitación debe tener al menos un criterio de evaluación, pero tampoco los criterios tienen un tope máximo.

Intentar recoger todos estos datos en varias pantallas habría exigido, o bien hacer un uso extensivo de las variables de sesión, para almacenar temporalmente los datos que se van introduciendo, o bien guardar registros temporales en la base de datos o en el sistema de archivos.

Estas soluciones presentan ventajas e inconvenientes. Por un lado, como ventaja, se simplifican mucho las vistas a generar, pero por otro, como inconvenientes, hacen más farragosa la experiencia de usuario y complican la lógica de negocio del programa, que debe cuidar del borrado de estos datos temporales cuando son abandonados por el usuario.

Otra alternativa a las vistas múltiples es hacer uso de JavaScript y de las funciones de manipulación del DOM para ir añadiendo elementos de introducción de datos a medida que estos se precisen. Y, para no

complicar visualmente la estructura de introducción de datos, separar el nivel de documentos a unas ventanas modales.

Esta última ha sido la opción elegida en este caso, principalmente porque hace mucho más agradable la experiencia de usuario, evitándole la navegación entre páginas, que puede hacer que se llegue a perder la noción de ubicación en la aplicación.

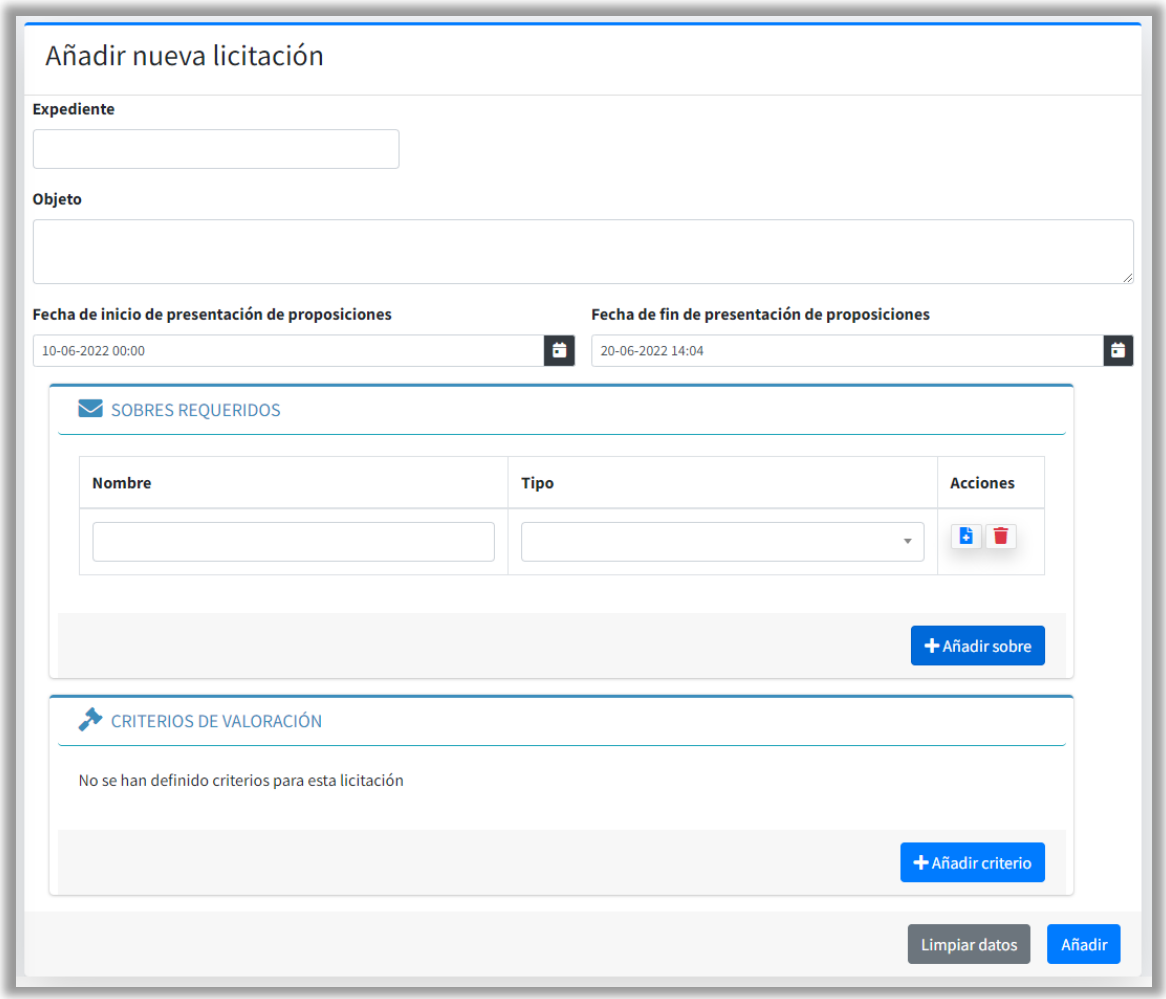

Ilustración 16 Vista de creación de licitación

Como se puede ver en la anterior ilustración, la solución implementada permite añadir nuevos sobres y criterios, y cada sobre incorpora una acción para añadir documentos en aquel. Como se ha dicho, la introducción de documentos se realiza a través de una ventana modal.

Es importante reseñar que se han hecho pruebas de ambas alternativas con usuarios potenciales reales de la aplicación, y a todos ellos les ha resultado mucho más satisfactoria la solución aplicada.

12. Acceso al código y a la aplicación

El código de la última versión de la aplicación desarrollada se puede encontrar en el siguiente repositorio de GitHub:

https://github.com/avazquezfr/lyceos

Además, la aplicación se ha instalado en AWS en producción. El enlace a la aplicación es:

http://lyceos-env-1.eba-dfiru2wb.eu-west-3.elasticbeanstalk.com/

Y las credenciales para poder acceder son las siguientes:

Usuarios Administración Pública		
Rol	**Usuario**	**Contraseña**
Administrador Aplicación	32657340P	*password*
Personal OC	00000014Z	*password*
Personal OA	00000001R	*password*

Usuarios Empresas Licitadoras		
Rol	**Usuario**	**Contraseña**
Empresa	tonivf@gmail.com	*password*

Hay que destacar que, por limitaciones del uso de correos electrónicos en la capa gratuita de AWS (prevención de *spam*), no es posible enviar correos electrónicos a direcciones que no hayan sido verificadas previamente en los servicios correspondientes de AWS. Por ello, el autorregistro de las empresas en la plataforma instalada no es posible sin este paso previo, puesto que la aplicación envía un correo electrónico para verificar la dirección suministrada por el usuario, y no permite el acceso del usuario hasta que no la haya confirmado.

13. Conclusiones

El trabajo realizado se puede evaluar en dos vertientes: por un lado, la referida al producto final, y por otro la referida al proyecto.

En cuanto al producto final, al entregable, las conclusiones son satisfactorias. Se ha logrado una aplicación funcional, moderna, sencilla y que cumple con los objetivos planteados.

Y en lo tocante al proyecto, si bien éste se ha logrado concluir satisfactoriamente, durante su desarrollo han surgido diferentes riesgos que lo han hecho peligrar.

Del proceso de elaboración del TFG se ha de decir que se ha profundizado en el conocimiento de buenas prácticas en programación y seguridad de los flujos de datos. Las competencias técnicas del estudiante se han reforzado y, en parte a la fuerza, se ha aprendido a planificar mejor.

Debe comentarse que, si bien los objetivos planteados se han alcanzado, estos han tenido que ser replanteados por la redefinición del alcance del proyecto. Desde el punto de vista de este estudiante, la carga de trabajo de un proyecto de este tipo excede considerablemente la dedicación de tiempo de una asignatura de 12 créditos (300 horas).

En cuanto al seguimiento de la metodología y de la planificación, se ha seguido en su inmensa mayoría. Además, la documentación generada en la fase de diseño ha sido de gran utilidad en la fase de implementación.

En cuanto al futuro del proyecto, se puede decir que éste era un proyecto de especial interés personal, puesto que el estudiante se dedica profesionalmente a tiempo completo a la instrucción y asesoramiento en el manejo de la Plataforma actualmente existente, y que es un proyecto viable, en el que se deben retomar las características descartadas por causa de las limitaciones de tiempo de la asignatura. Además, la propia definición del diseño se ha realizado teniendo en mente la facilidad para poder introducir nuevos elementos y funcionalidades en la aplicación.

14. Glosario

PLACE. *PLAtaforma de Contratación del Estado*. Herramienta proporcionada por el Gobierno de España para la publicación de licitaciones y asistencia en licitación electrónica.

AWS. *Amazon Web Services*. Plataforma en la nube propiedad de Amazon, que suministra distintos servicios de tecnologías de la información complementarios entre sí.

MVC. *Model-View-Controller*. Patrón arquitectónico en el que se separa el código encargado de la gestión de la lógica de negocio (controlador) del encargado del acceso a los datos (modelo) y del que proporciona la entrada/salida al usuario (vista).

IaaS. *Infrastructure As A Service*. Servicio en la nube que proporciona infraestructura de tecnologías de la información a los usuarios finales.

ORM. *Object Relational Mapping*. Modelo de programación que permite mapear las estructuras de una base de datos relacional sobre una estructura lógica de entidades.

HTML. *HyperText Markup Language*. Lenguaje de marcas estándar para la elaboración de páginas web.

CSS. *Cascading Style Sheets*. Lenguaje de definición de atributos para aplicar en documentos creados con un lenguaje de marcas.

JSON. *JavaScript Object Notation*. Formato sencillo de intercambio de datos.

XML. *eXtensible Markup Language*. Metalenguaje de marcas ampliamente empleado para intercambio de datos.

DOM. *Document Object Model*. Estructura de objetos que genera el navegador en memoria para cada uno de los elementos de una página web.

15. Bibliografía

[1] «Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público,» [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2017-12902. [Último acceso: 9 junio 2022].

[2] «Eloquent: Relationships - Laravel - The PHP Framework For Web Artisans,» [En línea]. Available: https://laravel.com/docs/9.x/eloquent-relationships#custom-polymorphic-types. [Último acceso: 9 junio 2022].

[3] «AdminLTE,» [En línea]. Available: https://adminlte.io/. [Último acceso: 9 junio 2022].

[4] «Easy AdminLTE integration with Laravel,» [En línea]. Available: https://github.com/jeroennoten/Laravel-AdminLTE. [Último acceso: 9 junio 2022].

[5] «Laravel kits,» [En línea]. Available: https://laravel.com/docs/9.x/starter-kits. [Último acceso: 9 junio 2022].

[6] «Laravel,» [En línea]. Available: https://laravel.com/. [Último acceso: 9 junio 2022].

16. Agradecimientos

Dicen que es de bien nacidos ser agradecidos. Y no puedo terminar este trabajo sin reconocer los apoyos y ayudas recibidas, ya no sólo durante su realización, sino durante el trayecto de los estudios del Grado. Por supuesto, el orden no refleja ningún tipo de importancia o preferencia, ya que todos ellos han contribuido de una forma u otra a que yo haya podido acabar este camino.

Si alguien se ha ganado a pulso el derecho a mi gratitud, esa persona es Itziar. No ha sido sólo una compañera en el ámbito académico, sino una amiga y en muchas ocasiones confidente, paño de lágrimas y receptora de mis frustraciones. Gracias por ser como eres, por estar ahí y por haber coincidido en mi vida.

Aunque su participación no haya sido académica, la siguiente persona a la que agradecerle su apoyo, comprensión, paciencia, estímulo y tantas cosas que necesitaría otros veinte folios más para poder hacerlo correctamente, es Carmen, mi pareja, la persona más importante de mi vida. Eres el amor de mi vida, y siempre estarás por delante de todo.

Y, cómo no, a mis padres. Lamento que no lo hayáis podido ver. Gracias, *viejiños*, va por vosotros, os lo debía.

Por último, y no menos importante, a quien haya leído hasta aquí, por saber disculpar mis errores y perdonar mis defectos.

Gracias.