# 

# Agradecimientos

# Índice

# Índice de figuras

# Índice de tablas

# Resumen

**Palabras clave**

En esta memoria se va a describir el funcionamiento de la herramienta creada para poder gestionar con más facilidad los acuerdos de los estudiantes de la Universidad de Alcalá que van a realizar una estancia Erasmus

La herramienta consiste en una aplicación Java Web donde existen dos tipos de usuarios, con sus funciones propias: los alumnos que van a realizar una estancia Erasmus y los coordinadores (tutores) de estos alumnos.

Toda esta información es almacenada en una base de datos PostgreSQL, la cual se conecta con la aplicación a través del driver JDBC.

# Abstract

# Keywords

This report will describe the functioning of the computer program created to be able to more easily manage the agreements of the students of the University of Alcalá who are going to carry out an Erasmus stay.

The computer program consists of a Java Web application where there are two types of users, with their own functions: the students who are going to carry out an Erasmus stay and the coordinators (tutors) of these students.

All this information is stored in a PostgreSQL database, which is connected to the application through the JDBC driver.

# Resumen extendido

El proyecto realizado consiste en la creación de una herramienta para administrar los acuerdos de los estudiantes de la Universidad de Alcalá que van a llevar a cabo una estancia Erasmus de manera más eficiente, permitiendo ahorrar tiempo innecesario al tutor de cada alumno.

Dicha herramienta consiste en una página web interna de la Universidad de Alcalá, cuyo servidor será propio de la Universidad. La página web se ha desarrollado en Java, utilizando el IDE NetBeans, en el cual se han tenido que instalar los plugins necesarios para el correcto funcionamiento de la aplicación. Además, para crear dicha web, se ha utilizado la tecnología JSP, permitiendo crear el contenido que los usuarios dispondrán.

La información necesaria para el correcto funcionamiento de la web se obtiene o se manipula en una base de datos relacional diseñada con PostgreSQL. Para realizar la conexión entre la página web a la base de datos, se utiliza el driver JDBC (Java DataBase Connectivity).

Para poder compilar y ejecutar la aplicación, se ha utilizado el servidor web Apache Tomcat, pero como se ha comentado anteriormente, se usará un servidor de la Universidad cuando se lance la herramienta.

Una vez lanzada la aplicación para su función, tendrán acceso a ella los alumnos que hayan sido aceptados para realizar una estancia Erasmus y los tutores de dichos estudiantes. Evidentemente, habrá dos tipos de usuarios: los alumnos y los profesores.

Los alumnos podrán dar de alta asignaturas, tanto de la Universidad de Alcalá como de la universidad de destino. Para dar de altas las asignaturas deberán introducir la información necesario como los créditos, el nombre o el código de dicha asignatura. Con estas asignaturas, podrán dar de alta las convalidaciones que deseen realizar en una estancia determinada. También, podrán consultar el estado de las convalidaciones, es decir, si ha sido aceptada, rechazada, cancelada (decisión propia) o todavía está pendiente de confirmar. Además, podrán buscar las antiguas convalidaciones realizadas de las asignaturas que deseen para facilitar la elección de las asignaturas que compondrán el acuerdo.

Por otro lado, los profesores podrán ver las información de las convalidaciones que los alumnos desean realizar seleccionando la estancia que ellos deseen y, a su vez, aceptarlas o rechazarlas, además de poder cancelarlas si un alumno rectifica en una convalidación que ha recibido la aceptación del tutor. Igualmente, podrán dar de altas a nuevos alumnos, los cuales han sido adjudicados al coordinador, además de otorgarles una estancia. También, podrá dar de alta grados de la Universidad de Alcalá y universidades de destino. Por último, podrá cerrar estancias una vez que el alumno ya haya vuelto de su estancia o, si el alumno decide renunciar a la estancia, establecerla como renunciada.

En esta memoria se redacta la motivación, contexto…….

Manual de usuario, presupuesto…….

Metodología, requisitos, diagrama………………

# Introducción

# Contexto

El programa Erasmus (European Community Action Scheme for the Mobility of University Students) es aquel que fomenta el intercambio de estudiantes entre universidades europeas, cuya duración puede tener entre 1 semestre a un curso completo como máximo. La Universidad de Alcalá participa en este programa y posee convenios con más de 50 países de todo el mundo.

Este programa tiene multitud de ventajas para los estudiantes, desde aprender nuevos idiomas hasta conocer otras culturas. Antes de iniciar la estancia, es necesario seleccionar las asignaturas que se desean cursar en la Universidad de Destino, las cuales serán usadas para convalidar por asignaturas de la Universidad de Alcalá siempre que se cumplan una serie de requisitos. Para ello, se crea el documento Learning Agreement.

El Learning Agreement es el documento más importante que tiene que rellenar un alumno que ha recibido la confirmación de admisión de una universidad de destino para realizar una estancia Erasmus. Consiste en un acuerdo de aprendizaje, donde cada estudiante introducirá información acerca de las asignaturas que cursará en la universidad de destino y sus correspondientes asignaturas de la universidad de origen, las cuales serán solicitadas para su correcta convalidación.

Antes de poder completar el Learning Agreement, el alumno enviará a su tutor académico las asignaturas que desea convalidar, decidiendo este último si las convalidaciones son posibles y, si se rechazan, buscar otra posible convalidación. Cuando el tutor y el alumno se ponen de acuerdo, el alumno ya puede empezar a completar el Learning Agreement.

Una vez que el alumno complete todos los campos necesarios, su universidad de destino y su universidad de origen recibirán una copia de él.

Para poder disfrutar de tu Erasmus, deberá obtener este acuerdo firmado por el coordinador Erasmus de su nueva Universidad, acordando el plan de estudios que se recoge en el Learning Agreement.

Además, se puede modificar durante la estancia. El alumno puede modificar acuerdos ya aceptados por las razones que él considere, siempre y cuando deliberándolo con el coordinador académico de la universidad de destino.

Finalmente, una vez que concluye el contrato de estudios, el alumno deberá entregarlo con todas las firmas necesarias en la Oficina de Relaciones Internacionales.

En la Fig.1 se muestra una parte del Learning Agreement que el alumno debe completar.



**Fig.1 Parte del Learning Agreement**

# Motivación

A priori, rellenar todo el contenido necesario en el Learning Agreement puede parecer algo simple y sencillo, pero no es así. Completar todo el documento puede llegar a ser algo verdaderamente costoso para el estudiante, incluso incrementaría su dificultad si recibe el rechazo de algún acuerdo una vez entregado.

El estudiante debe llevar a cabo una exhausta investigación, la cual trata de explorar la web de la universidad de destino para ver el temario de cada asignatura, hablar con otros estudiantes que han realizado la estancia allí o preguntar a su coordinador, quien es posible que no tenga soluciones.

En cuanto a la labor del tutor de la universidad de origen, puede ser incluso más costosa para el alumno. Habitualmente, antes de recibir el Learning Agreement oficial, el profesor recibe por parte del alumno las propuestas de asignaturas tanto de origen como de destino. Estas se pueden recibir de diversas maneras, desde un documento Word bien estructurado hasta en el contenido de un correo electrónico. Sin embargo, lo más tedioso para el profesor consiste en recorrerse todas las webs, informaciones y guías docentes de cada asignatura de la Universidad de destino y comparar su temario con las asignaturas de la Universidad de origen. A esto hay que tener en cuenta que regularmente el profesor no recibe toda la información necesaria por parte del alumno, omitiéndose información relevante como número de créditos, página web de la asignatura de la Universidad de destino, etcétera. Por lo tanto, esta herramienta es creada para evitar todos los inconvenientes comentados.

Respecto al alumno, dispondrá de una interfaz manejable y sencilla de entender para introducir las convalidaciones que él desee, además de poder observar las convalidaciones de cursos anteriores de las asignaturas va a solicitar convalidar.

En cuanto al profesor, tendrá la posibilidad de ver toda la información necesaria para decidir si una convalidación es aceptada o rechazada. Esta información consiste en los créditos de ambas asignaturas, un link que redirige a la web de cada asignatura, nombre, etcétera.

En conclusión, la motivación por la cual se crea esta herramienta es la de resolver todos los problemas que tienen tanto alumno como profesor para decidir cuáles serán las convalidaciones que se incluirán en el Learning Agreement definitivo.

# Objetivos

El objetivo principal de la implementación de esta herramienta es facilitar la elección de asignaturas a los alumnos Erasmus que desea convalidar y que su tutor académico pueda decidir si acepta la convalidación con más rapidez. Gracias a la elaboración de esta herramienta, se agilizará el proceso anterior a la realización del Learning Agreement.

Además, se proponen los siguientes objetivos secundarios que complementará al objetivo principal:

* Desarrollar una aplicación web a la que se pueda acceder desde Google Chroome e Internet Explorer.
* Desplegar una interfaz gráfica intuitiva para el usuario que sea simple de manejar.
* Diseñar un sistema gracias al estudio de herramientas como Java, NetBeans, JSP, PostgreSQL o HTML, además de seguir técnicas de diseño de programación orientada a objetos.
* Estudiar el funcionamiento de algunos patrones de diseño como el patrón DAO o el patrón Modelo Vista Controlador.
* Elaborar un manual de usuario sobre el funcionamiento de la herramienta.
* Redactar esta memoria, la cual será muy importante debido a posibles mejoras de la herramienta, así como futuros estudios de los datos almacenados.

Análisis y Especificación de requisitos

# Descripción del sistema

La aplicación web desarrollada incluye las funcionalidades necesarias para que los usuarios, tanto profesores como alumnos puedan realizar las operaciones que ellos deseen. La herramienta debe distinguir si el usuario que inicia sesión es alumno o profesor. Una vez que lo distingue, accede a la interfaz correspondiente.

En el caso del alumno, muestra una pantalla con diferentes opciones, las cuales él elegirá según desee y se abrirá una nueva página con la interfaz de la opción que él ha elegido. Estas opciones pueden ser insertar nueva asignatura de origen o de destino, realizar una nueva convalidación, buscar convalidaciones antiguas, ver el estado de las convalidaciones actuales o cambiar la contraseña.

Respecto al profesor esto se amplía. Al logearse un usuario profesor aparecerá una página con varios aspectos, por un lado, enlaces para realizar diversas operaciones en las que se incluyen dar de alta a alumnos, profesores, estancias de alumnos, asignaturas universidades y grados, cerrar estancias o renunciar a ellas (con petición previa del alumno) y cambiar la contraseña. y por otro lado un desplegable de opciones con las estancias de los alumnos que tiene asignados, tanto actuales como pasadas. Al seleccionar una opción del desplegable se completa la tabla de la parte inferior con la información de las convalidaciones que componen esa estancia. Un poco más abajo, se encuentran las opciones de modificar asignatura (aceptar o rechazar la convalidación) y cancelar asignatura para las convalidaciones aceptadas que el alumno desea cancelar. En la parte derecha aparece información del alumno seleccionado.\*\*\*\*\*

# Catálogo de requisitos

Tras explicar la descripción del sistema, llegamos al turno de listar los requisitos necesarios para el correcto comportamiento del sistema.

# Requisitos funcionales

**Tabla.1 Lista de requisitos funcionales**

|  |  |  |
| --- | --- | --- |
| IDENTIFICADOR | DESCRIPCIÓN | PRIORIDAD |
| RF-01 | El sistema debe solicitar al usuario que inicie sesión para acceder a las funcionalidades | Muy alta |
| RF-02 | El sistema comprobará que el usuario y la contraseña coinciden con lo registrado en la base de datos | Muy alta |
| RF-03 | El sistema proporcionará distintas vistas según el usuario que haya iniciado sesión | Muy alta |
| RF-04 | El sistema registrará toda la información que se desee añadir en la base de datos | Muy alta |
| RF-05 | Ambos usuarios podrán agregar asignaturas tanto de origen como de destino a ambos usuarios | Muy alta |
| RF-06 | Ambos usuarios podrán cambiar su contraseña | Alta |
|  |  |  |
| IDENTIFICADOR | DESCRIPCIÓN | PRIORIDAD |
| RF-07 | El alumno podrá añadir una nueva convalidación a la estancia que él considere | Muy alta |
| RF-08 | El alumno podrá buscar antiguas convalidaciones de la asignatura de origen que él desee introduciendo el código de dicha asignatura | Muy alta |
| RF-09 | El alumno podrá contemplar el estado de las convalidaciones que ha solicitado de la estancia que él desee | Muy alta |
| RF-10 | El coordinador podrá dar de alta a un nuevo alumno | Muy alta |
| RF-11 | El coordinador podrá dar de alta una nueva estancia para un alumno registrado | Muy alta |
| RF-12 | El coordinador podrá dar de alta a un nuevo coordinador | Alta |
| RF-13 | El coordinador podrá dar de alta nuevas universidades de destino | Muy alta |
| RF-14 | El coordinador podrá dar de alta nuevos grados de la UAH | Muy alta |
| RF-15 | El coordinador podrá establecer como renunciada a una estancia de un alumno | Alta |
| RF-16 | El coordinador podrá cerrar una estancia de un alumno que ya ha terminado su estancia Erasmus | Muy alta |
| RF-17 | El coordinador podrá contemplar la información de las convalidaciones de la estancia de un alumno que él desee | Muy alta |
| RF-18 | El coordinador podrá aceptar o rechazar una convalidación según él crea oportuno | Muy alta |
| RF-19 | El coordinador podrá realizar comentarios acerca de por qué acepta o rechaza una convalidación | Alta |
| RF-20 | El coordinador podrá cancelar una convalidación a petición del alumno | Muy alta |
| RF-21 | El coordinador podrá realizar comentarios acerca de por qué cancela una convalidación | Alta |

# Requisitos no funcionales

**Tabla.2 Lista de requisitos no funcionales**

|  |  |  |
| --- | --- | --- |
| IDENTIFICADOR | DESCRIPCIÓN | PRIORIDAD |
| RNF-01 | Los nombres y contraseñas de usuarios serán únicos | Muy alta |
| RNF-02 | Se almacenará toda la información necesaria para el correcto funcionamiento del sistema en una base de datos PostgreSQL | Muy alta |
| RNF-03 | Se desarollará el proyecto utilizando el lenguaje de programación Java | Muy alta |
| RNF-04 | Se utilizará la tecnología JSP para la creación de la página web | Muy alta |
| RNF-05 | El proyecto se desarrolla en el entorno de desarrollo NetBeans IDE | Alta |
| RNF-06 | La aplicación se desarrolla siguiendo unos patrones software | Alta |
| RNF-07 | El sistema deberá ser capaz de operar adecuadamente con varios usuarios de manera concurrente | Muy alta |
| RNF-08 | El sistema estará disponible a cualquier hora del día | Alta |
| RNF-09 | El sistema será accesible desde dentro del servidor de la Universidad de Alcalá | Muy alta |
| RNF-10 | El sistema será accesible a través de un ordenador de forma online | Muy alta |
| RNF-11 | El sistema proporcionará mensajes de confirmación o error orientativos a los usuarios | Alta |
| RNF-12 | El sistema será escalable, por lo que permitirá la implementación de nuevas funcionalidades | Alta |
| RNF-13 | Se entregará un manual de usuario que detallará el uso de la aplicación | Muy alta |
|  |  |  |

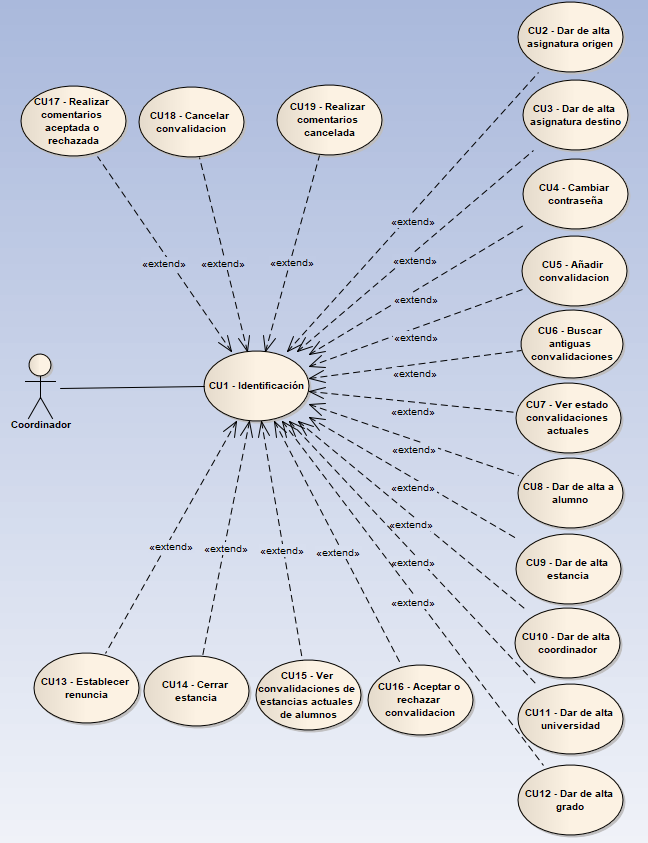
# Casos de uso

Una vez establecida la especificación de requisitos, pasamos al estudio de los casos de uso del sistema, con el fin de admirar los diferentes puntos de vista del sistema.

Habrá dos tipos de actores: el profesor y el alumno. Cada uno realizará unas funciones distintas que se ven reflejadas en los siguientes diagramas.



**Fig.2 Diagrama de casos de uso del actor alumno**



**Fig.3 Diagrama de casos de uso del actor coordinador**

# CU1 – Identificación

**Descripción:** caso de uso que representa el acceso a las funcionalidades del sistema por los dos tipos de usuario: alumno y profesor.

**Requisitos:** RF01, RF02 y RF03

# CU2 – Dar de alta asignatura origen

**Descripción:** caso de uso que representa la inserción por parte de un alumno o de un profesor de una asignatura que se cursa en la Universidad de Alcalá. Para ello, habrá que introducir el código, el nombre, el número de créditos, el tipo (básica, obligatoria, optativa o transversal) y la información (la página web de la asignatura).

**Requisitos:** RF-04 y RF-05

# CU3 – Dar de alta asignatura destino

**Descripción:** caso de uso que representa la inserción por parte de un alumno o de un profesor de una asignatura que se desea cursar en la Universidad donde el alumno ha sido admitido. Para ello, habrá que introducir el código, el nombre, el número de créditos, la universidad donde se imparte y la información (la página web de la asignatura).

**Requisitos:** RF-04 Y RF-05

# CU4 – Cambiar contraseña

**Descripción:** caso de uso que representa la posibilidad por parte de un alumno o un profesor de cambiar su contraseña. El usuario deberá introducir la contraseña antigua y la nueva dos veces para su correcta modificación.

**Requisitos:** RF-04 y RF-06

# CU5 – Añadir convalidación

**Descripción:** caso de uso que representa la inserción por parte del alumno de una convalidación de una estancia determinada. El alumno deberá elegir la estancia en la que quiere realizar la convalidación (por si tiene más de una sin cerrar), la asignatura de origen (será un desplegable con todas las asignaturas registradas de la Universidad de Alcalá) y la asignatura de destino (será un desplegable con todas las asignaturas de las universidades de las estancias que tiene en curso).

**Requisitos:** RF-04 y RF-07

# CU6 – Buscar antiguas convalidaciones

**Descripción:** caso de uso que representa la posibilidad por parte del alumno de buscar antiguas convalidaciones de la asignatura de origen que él desee. El alumno deberá introducir el código de la asignatura de origen y le aparecerá toda la información de las convalidaciones aceptadas realizadas con dicha asignatura. Esta información recoge, por parte de la asignatura de origen, el código, nombre, tipo, créditos y un link que redirige a la página de la asignatura y, en la parte derecha por parte de la asignatura de destino, el código, nombre, créditos, un link que redirige a la página de la asignatura, universidad, fecha de convalidación, curso académico, estado (siempre aceptada) y comentarios.

**Requisitos:** RF-08

# CU7 – Ver estado convalidaciones actuales

**Descripción:** caso de uso que representa la posibilidad por parte del alumno de poder observar el estado de las convalidaciones actuales que desea realizar. El alumno elegirá la estancia que él desee comprobar y le aparecerá la información. Esta información recoge, por parte de la asignatura de origen, el código, nombre, tipo, créditos y un link que redirige a la página de la asignatura y, en la parte derecha por parte de la asignatura de destino, el código, nombre, créditos, un link que redirige a la página de la asignatura, universidad, fecha de convalidación, curso académico, estado (siempre aceptada) y comentarios. Además, aparecen al final los créditos totales aceptados en la Universidad de Alcalá y de la Universidad de destino.

**Requisitos:** RF-09

# CU8 – Dar de alta a alumno

**Descripción:** caso de uso que representa la inserción por parte del profesor de un alumno que ha sido asignado a dicho profesor. Para darlo de alta en el sistema habrá que introducir su nombre, apellidos, contraseña, nombre de usuario, DNI y grado que cursa en la Universidad de Alcalá.

**Requisitos:** RF-04 y RF-10

# CU9 – Dar de alta estancia

**Descripción:** caso de uso que representa la inserción por parte del profesor de una estancia de un alumno. El profesor tendrá que seleccionar el alumno, el tipo de estancia, curso académico, duración y la universidad de destino.

**Requisitos:** RF-04 y RF-11

# CU10 – Dar de alta coordinador

**Descripción:** caso de uso que representa la inserción de un profesor por parte de otro profesor. Se tendrán que introducir los campos nombre, apellidos, contraseña, nombre de usuario, DNI y departamento.

**Requisitos:** RF-04 y RF-12

# CU11 – Dar de alta universidad

**Descripción:** caso de uso que representa la inserción por parte de un profesor de una universidad posible de destino. Se tendrán que introducir los campos, código de universidad, nombre, país y ciudad.

**Requisitos:** RF-04 y RF-13

# CU12 – Dar de alta grado

**Descripción:** caso de uso que representa la inserción por parte de un profesor de un grado en la Universidad de Alcalá. Se tendrán que introducir los campos, código de grado, nombre y facultad.

**Requisitos:** RF-04 y RF-14

# CU13 – Establecer renuncia

**Descripción:** caso de uso que representa la renuncia de una estancia por parte del alumno. Dicho alumno deberá comunicar al profesor previamente su deseo de renuncia y, una vez confirmada su renuncia, el profesor lo introducirá en el sistema.

**Requisitos:** RF-01 Y RF-15

# CU14 – Cerrar estancia

**Descripción:** caso de uso que representa el cierre de una estancia. La mayoría de veces, este cierre se confirmará cuando el alumno vuelva tras realizar su estancia con éxito. Una vez confirmado su cierre, el profesor lo introducirá en el sistema.

**Requisitos:** RF-04 Y RF-16

# CU15 – Ver convalidaciones de estancias actuales de alumnos

**Descripción:** caso de uso que representa la posibilidad de que el profesor vea la información de las convalidaciones de una estancia que un alumno desea realizar. Esta información recoge, por parte de la asignatura de origen, el código, nombre, tipo, créditos y un link que redirige a la página de la asignatura y, en la parte derecha por parte de la asignatura de destino, el código, nombre, créditos, un link que redirige a la página de la asignatura, universidad, fecha de convalidación, curso académico, estado (siempre aceptada) y comentarios. Además, aparecen al final los créditos totales aceptados en la Universidad de Alcalá y de la Universidad de destino.

**Requisitos:** RF-17

# CU16 – Aceptar o rechazar convalidación

**Descripción:** caso de uso que representa la posibilidad de que el profesor, tras revisar la información de la convalidación, acepte o rechace la convalidación. Si es aceptada, podrá formar parte del Learning Agreement del alumno.

**Requisitos:** RF-04 y RF-18

# CU17 – Realizar comentarios aceptada o rechazada

**Descripción:** caso de uso que representa la posibilidad de que el profesor, tras aceptar o rechazar una convalidación, redacte el motivo de su decisión.

**Requisitos:** RF-04 y RF-19

# CU18 – Cancelar convalidación

**Descripción:** caso de uso que representa la posibilidad de que el profesor pueda cancelar una convalidación que había sido previamente aceptada. Esto se realizará cuando un alumno lo desee y se lo comunique al profesor

**Requisitos:** RF-04 y RF-20

# CU19 – Realizar comentarios cancelada

**Descripción:** caso de uso que representa la posibilidad de que el profesor, tras cancelar una convalidación, redacte el motivo de su decisión.

**Requisitos:** RF-04 y RF-21

Diseño

A continuación, se va a explicar la arquitectura software que va a tener el sistema, desde el diseño de la base de datos hasta el modelado de clases del proyecto.

# Modelado de datos

Para el modelado de datos, se ha optado por la opción de una base de datos relacional, debido a que facilitará las relaciones entre las diferentes asignaciones, como por ejemplo los alumnos asignados a un profesor, las estancias a los alumnos, las convalidaciones a las estancias, etc. Esto permitirá que los datos almacenados sean más accesibles independientemente de la acción que quiera realizar el usuario que esté utilizando la aplicación.

Para el proceso de creación de la base de datos que se va a utilizar, primero hay que crear el modelo entidad relación, compuesto de las tablas necesarias con sus campos, claves, claves foráneas y relaciones entre ellas, teniendo en cuenta la cardinalidad. Para la creación del modelo entidad relación se ha utilizado la herramienta PgModeler, ya que permite, una vez creado el modelo, exportar el modelo a código PostgreSQL, el cual se utilizará para la crear la base de datos en el servidor.

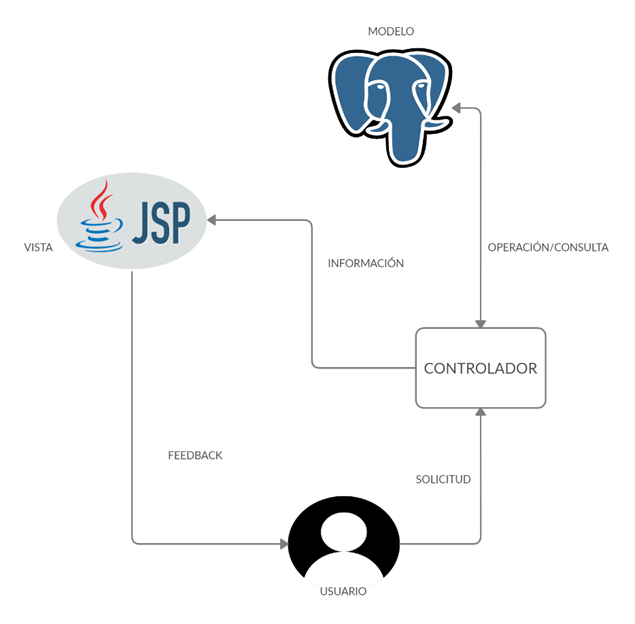
**Fig.4 Diagrama entidad relación del sistema**

# 

# Modelado de clases

La estructura de las clases de la aplicación estará basada en el patrón de arquitectura Modelo Vista Controlador para dividir la aplicación en módulos.

La arquitectura de la aplicación es la siguiente:



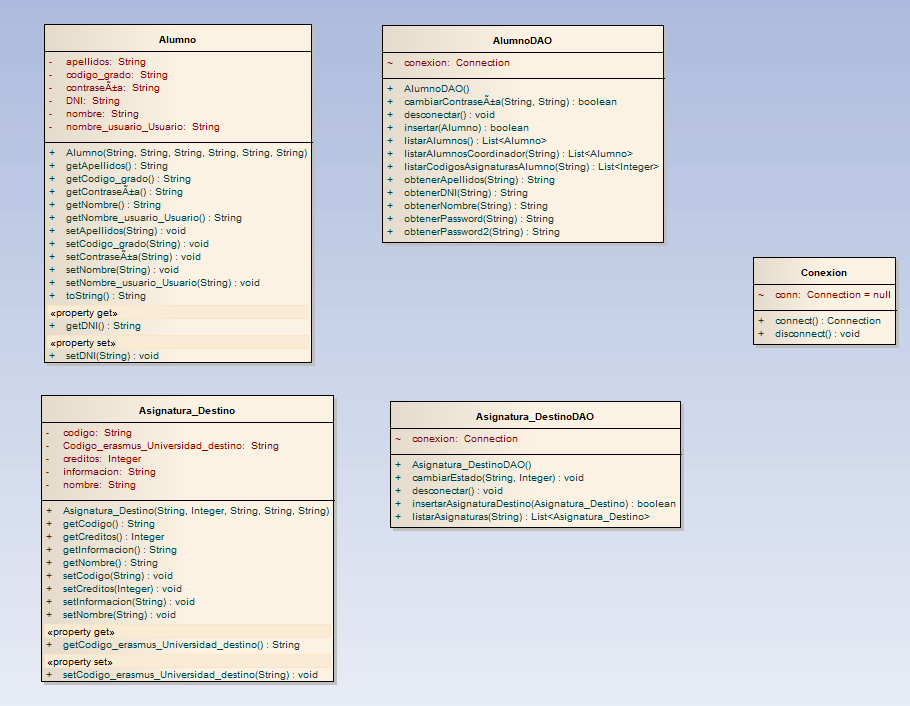
El primer módulo será la lógica del programa donde se almacenará la base de datos, este módulo se denomina modelo. El modelo será el encargado de interactuar con la base de datos realizando operaciones con ella, tanto de lectura y escritura. Las clases que constituyen el modelo son las siguientes:

* Clases con el mismo nombre que las tablas de la base de datos (Alumno, Coordinador, Estancia, Convalidación…). Estas clases tendrán los mismos atributos que las de la base de datos con su correspondiente constructor y métodos getter y setter.
* Una clase adicional denominada RelacionAsignaturas, la cual se utiliza para mostrar en diferentes tablas información sobre asignaturas de origen y de destino de una misma convalidación.
* Las clases DAO de cada una de las anteriores (AlumnoDAO, CoordinadorDAO, RelacionAsignaturasDAO…), las cuales sirven para interactuar directamente con la base de datos utilizando los métodos de las clases anteriores.

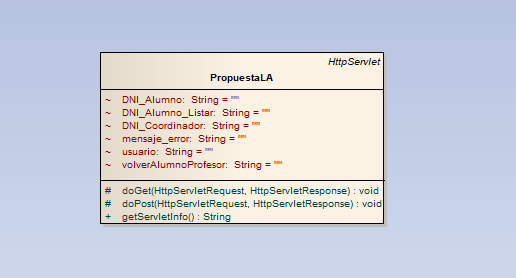
El siguiente módulo será la interfaz donde el usuario podrá interactuar con la aplicación, este módulo se denomina vista. La vista se encarga de recibir los datos del modelo y mostrárselos al usuario. Las clases que componen la vista son de tipo JSP, las cuales se comunican con el controlador para realizar las acciones necesarias que el usuario requiere en cada momento. En estas clases se crean toda la estructura de la vista, como cuadros de texto necesarios, desplegables, tablas, botones, enlaces, etc, a través de etiquetas HTML y se dan formato a través de estilos CSS.

El último será el módulo de las comunicaciones donde el usuario podrá comunicarse con la base de datos, este módulo se denomina controlador. Este módulo recibe los eventos de entrada, como puede ser una cadena de un campo de texto que inserte algún registro en la base de datos. La clase que representa al controlador se denomina PropuestaLA y va recibiendo continuamente peticiones de la vista a través de los métodos doGet y doPost. En el momento que recibe estas peticiones, realiza la acción requerida y responde a la vista con el resultado de aplicar la acción en ese preciso momento. Por ejemplo, en el caso de la vista login.jsp, recibe como entrada por parte del usuario, un nombre de usuario y su contraseña. Se envía al controlador y se ejecuta la acción “comprobarlogin” donde comprueba si los campos introducidos pertenecen a un usuario profesor, un alumno o son incorrectos. En los dos primeros casos abre la vista correspondiente a cada uno y, en el último, muestra una ventana de error que indica que las credenciales introducidas son incorrectas.

Por último, existirá una clase Conexión donde se realizará la conexión con la base de datos. Esta clase se instanciará en las clases DAO del modelo para que puedan interactuar con la base de datos una vez realizada la conexión.



**Fig.5 Algunas clases del modelo y la clase Conexión**

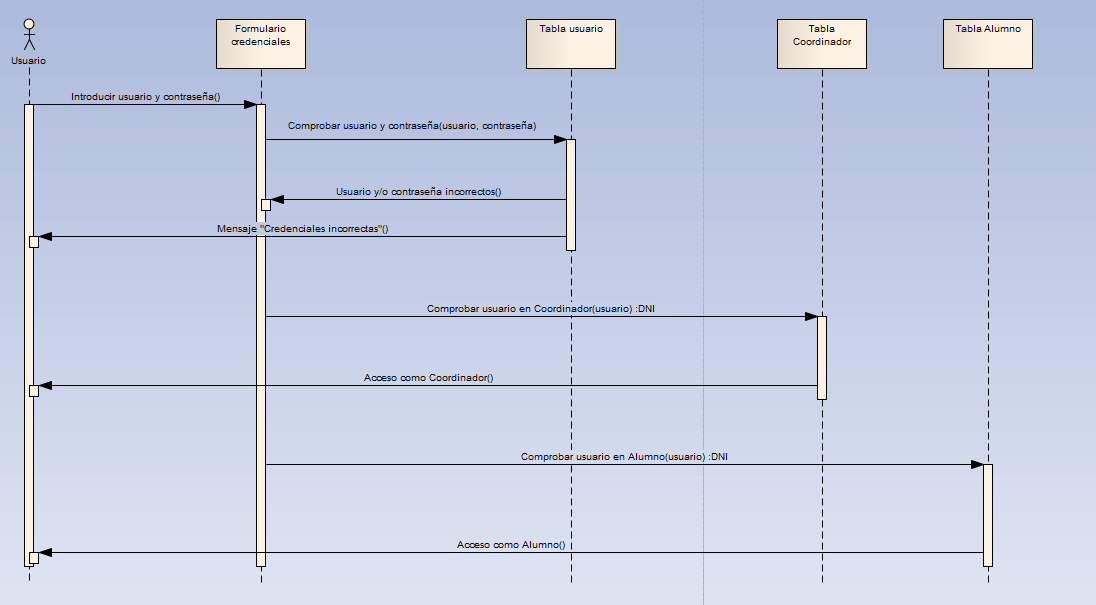


**Fig.6 Clase controlador**

# Diagramas de secuencia

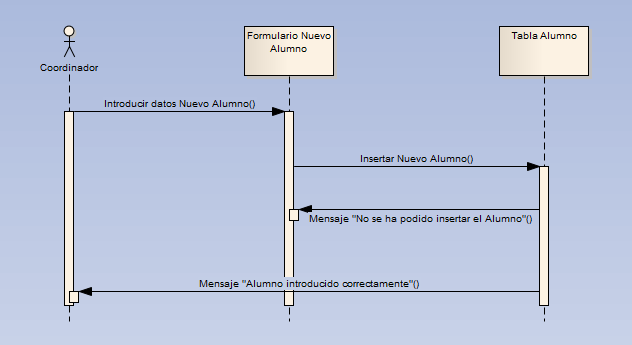
# Login

Este diagrama de secuencia consiste en el acceso de un usuario al sistema. El usuario se encontrará un formulario para introducir sus credenciales (usuario y contraseña). Si estas existen en el sistema, comprueba si es un usuario Coordinador y, si lo es, accede al sistema como Coordinador. Si no, comprueba si es Alumno y accede al sistema como Alumno. En caso contrario, devuelve error.



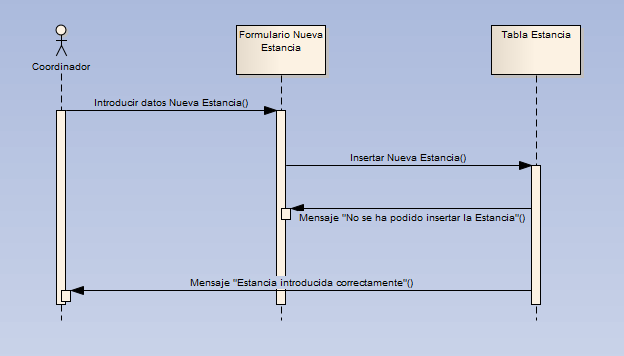
# Nuevo Alumno

Este diagrama de secuencia consiste en dar de alta a un alumno en el sistema. El Coordinador introduce los datos del alumno y, si son correctos, se inserta un nuevo Alumno y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



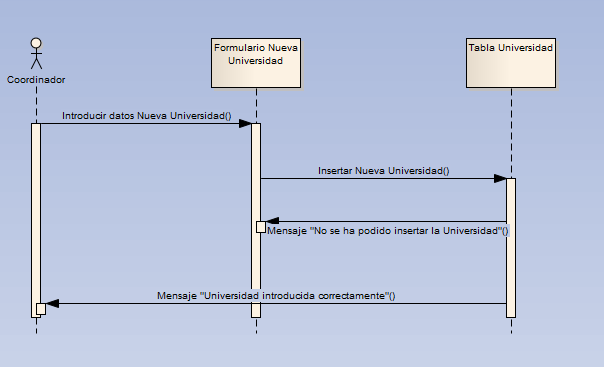
# Nueva Estancia

Este diagrama de secuencia consiste en dar de alta a una estancia en el sistema. El Coordinador introduce los datos de la estancia y, si son correctos, se inserta una nueva Estancia y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



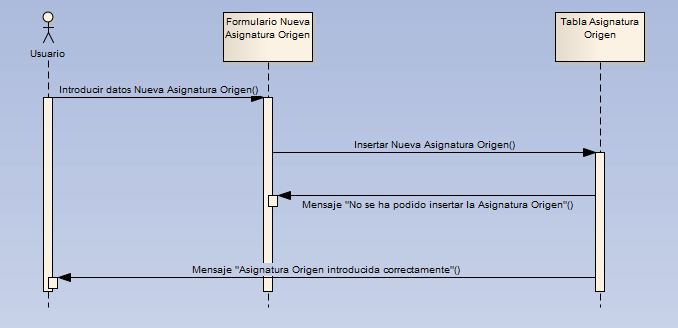
# Nueva Universidad

Este diagrama de secuencia consiste en dar de alta a una Universidad en el sistema. El Coordinador introduce los datos de la Universidad y, si son correctos, se inserta una nueva Universidad y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



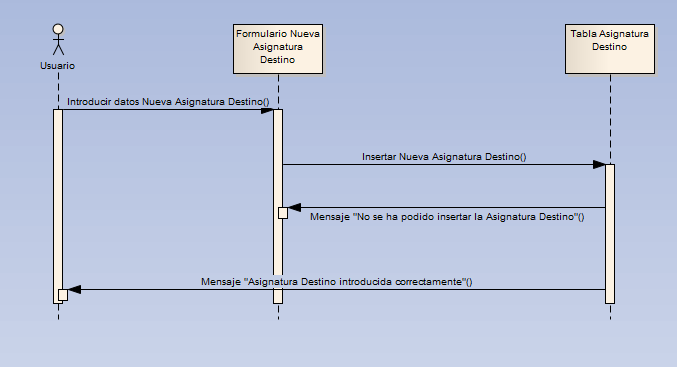
# Nueva Asignatura Origen

Este diagrama de secuencia consiste en dar de alta a una asignatura origen en el sistema. El usuario introduce los datos de la asignatura origen y, si son correctos, se inserta una nueva asignatura origen y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



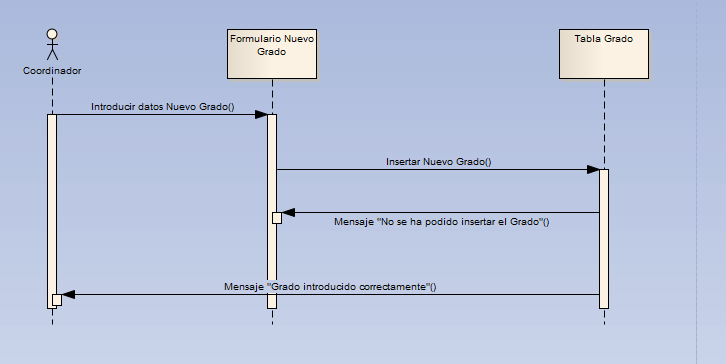
# Nueva Asignatura Destino

Este diagrama de secuencia consiste en dar de alta a una asignatura destino en el sistema. El usuario introduce los datos de la asignatura destino y, si son correctos, se inserta una nueva asignatura destino y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



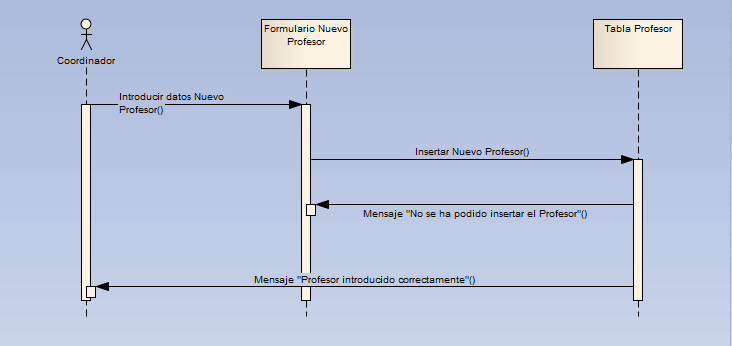
# Nuevo Grado

Este diagrama de secuencia consiste en dar de alta a un grado en el sistema. El Coordinador introduce los datos del grado y, si son correctos, se inserta un nuevo grado y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



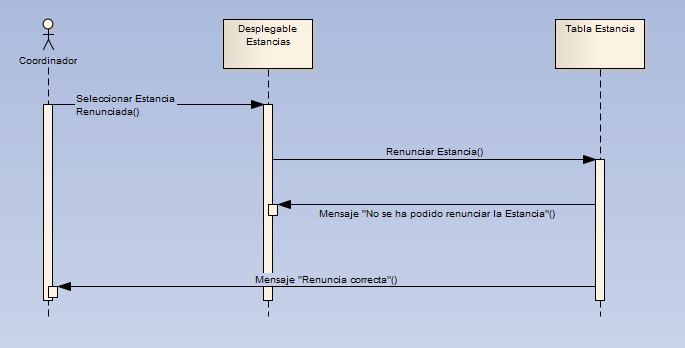
# Nuevo Profesor

Este diagrama de secuencia consiste en dar de alta a un profesor en el sistema. El Coordinador introduce los datos del profesor y, si son correctos, se inserta un nuevo profesor y lo confirma con un mensaje. En caso contrario, muestra un mensaje de error.



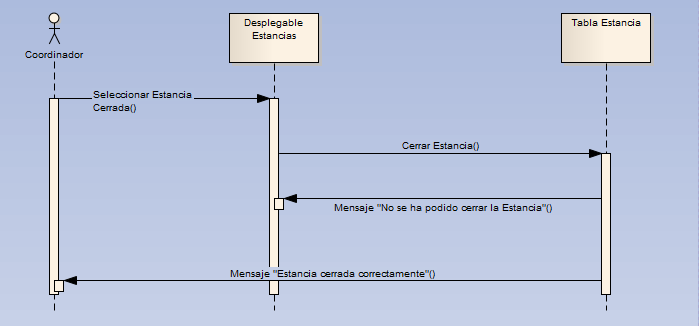
# Renuncia Estancia

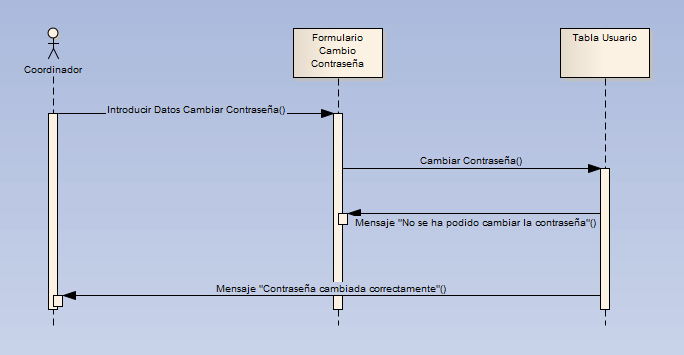
Este diagrama de secuencia consiste en renunciar una estancia. El coordinador selecciona la estancia que va a ser renunciada por el alumno. Si todo es correcto muestra un mensaje de confirmación y si no lo es muestra un mensaje de error.



# Cerrar Estancia

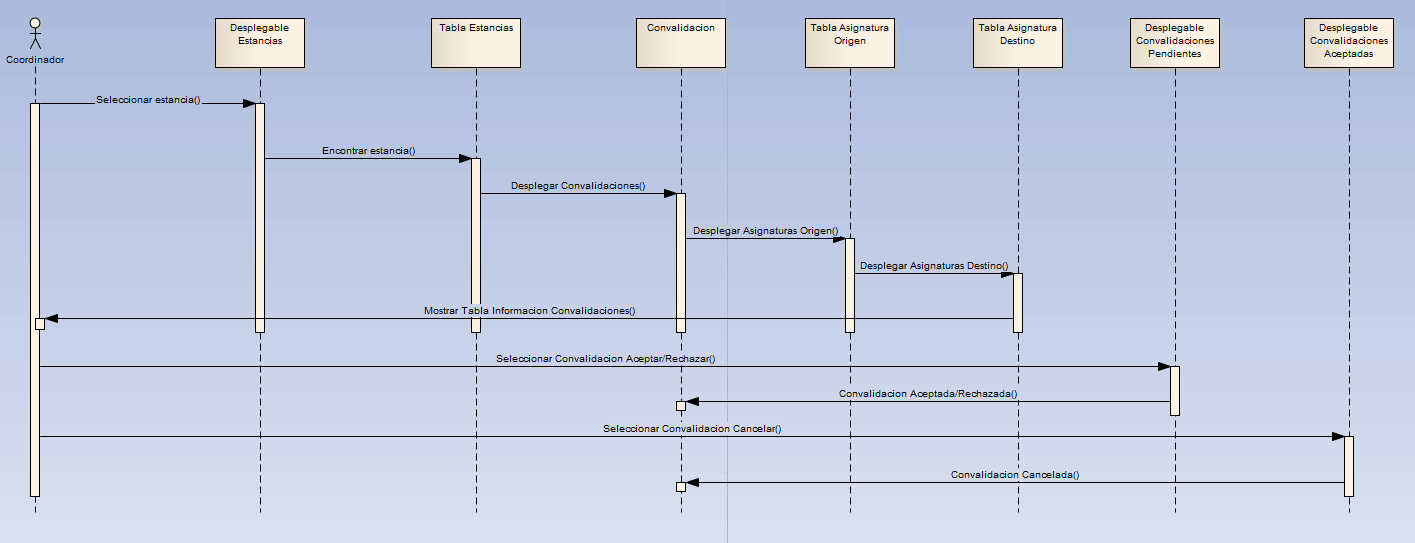
Este diagrama de secuencia consiste en cerrar una estancia. El coordinador selecciona la estancia que va a ser cerrada. Si todo es correcto muestra un mensaje de confirmación y si no lo es muestra un mensaje de error.





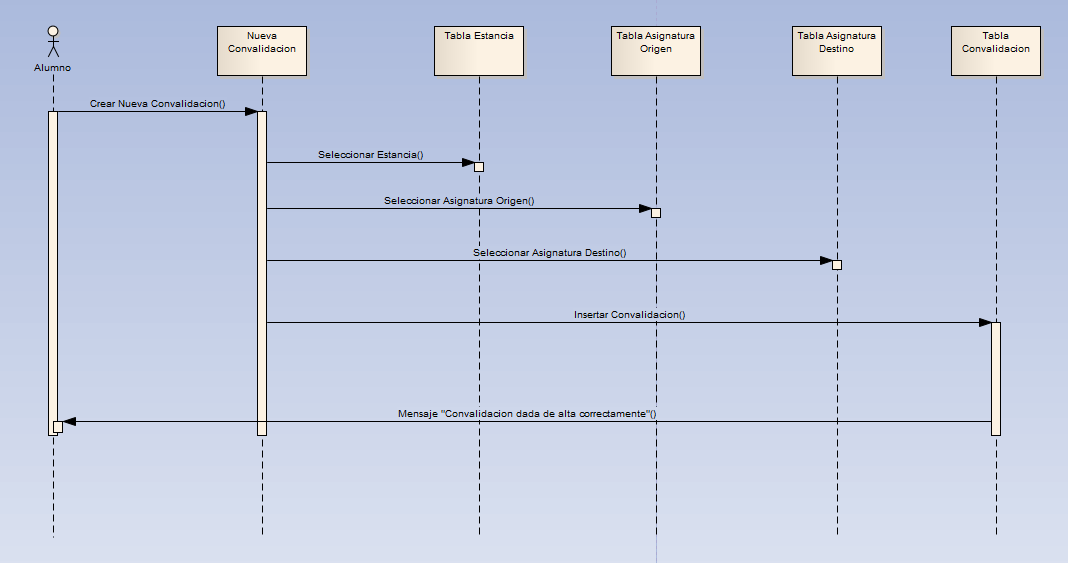
# Modificar convalidación

Este diagrama de secuencia consiste en modificar una convalidación, ya sea aceptarla, rechazarla o cancelarla. El coordinador selecciona la estancia que quiere contemplar. Una vez seleccionada, muestra una tabla con los datos de las convalidaciones y asignaturas de dicha estancia. A partir de ahora, el coordinador tiene toda la información necesaria para poder aceptar, rechazar o cancelar. Aparecerán dos desplegables, uno con las asignaturas pendientes, las cuales se podrán aceptar o rechazar y otro con la asignaturas aceptadas las cuales se podrán cancelar.



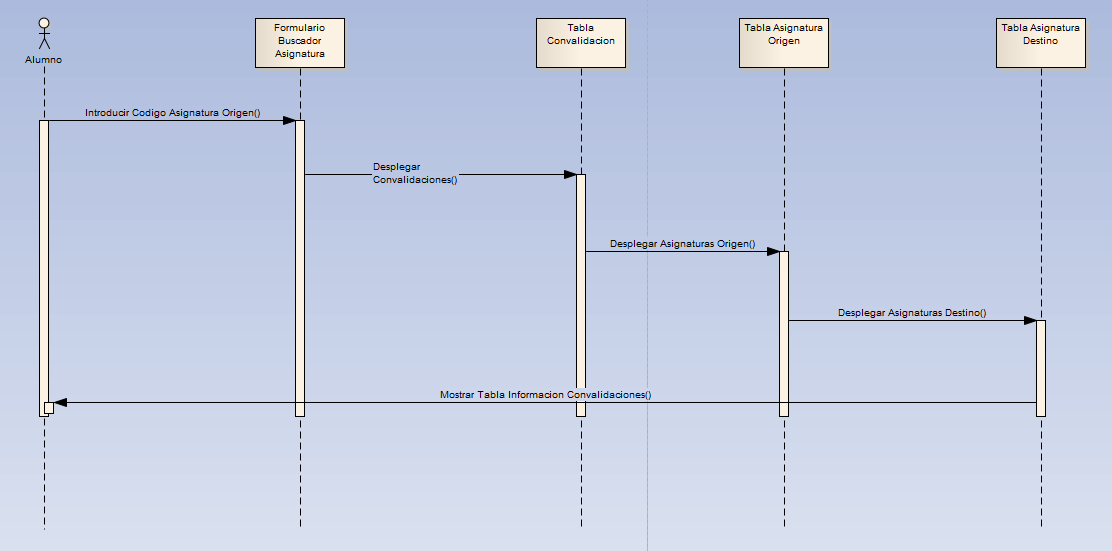
# Nueva convalidación

Este diagrama de secuencia consiste en insertar una nueva convalidación. Aparecen tres desplegables, uno con las estancias abiertas donde el alumno puede realizar la convalidación, otro con las asignaturas de la universidad de origen y otro con las asignaturas en el sistema de las universidades de las estancias abiertas del alumno. Si todo se realiza correctamente, aparece un mensaje de confirmación.



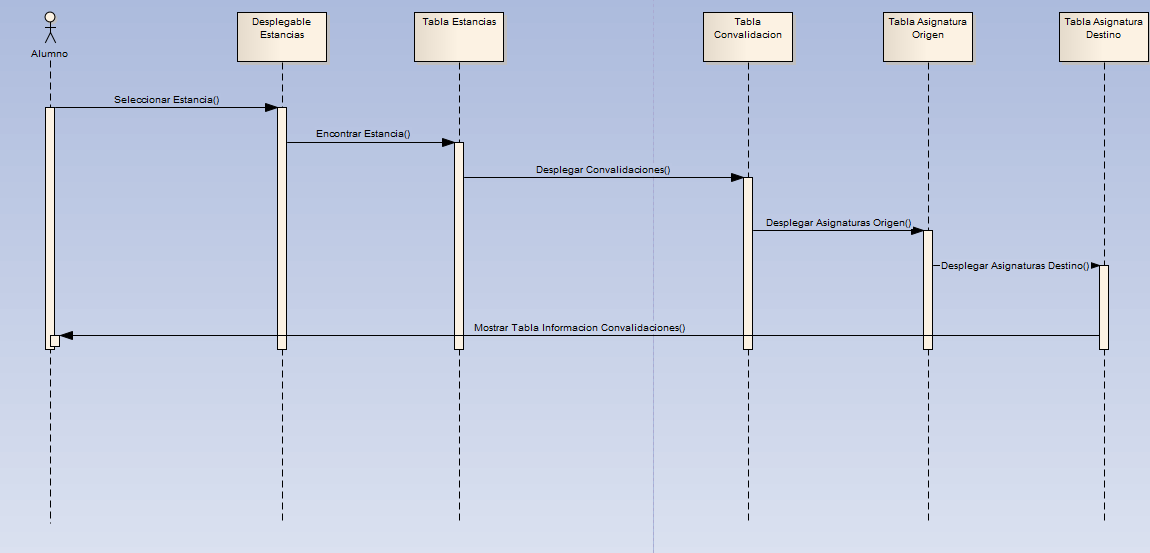
# Buscar antiguas convalidaciones

Este diagrama de secuencia consiste en buscar antiguas convalidaciones de otros alumnos. El alumno introduce el código de una asignatura de origen y se le muestra una tabla con todas las convalidaciones aceptadas con esa asignatura en todas las universidades.



# Ver estado convalidaciones

Este diagrama de secuencia consiste en modificar una convalidación, ya sea aceptarla, rechazarla o cancelarla. El alumno selecciona la estancia que quiere contemplar. Una vez seleccionada, muestra una tabla con los datos de las convalidaciones y asignaturas de dicha estancia.



Implementación

# Medios utilizados

Se han utilizado diversos medios para la realización del proyecto, los cuales algunos se han ido comentado y otros aún no. A continuación, vamos a describir cada uno de ellos indicando su versión si lo requiere. Los más relevantes son los siguientes:

* NetBeans (8.2): programa que se utiliza como IDE que permite programar en diferentes lenguajes (en nuestro caso Java, JSP y HTML).
* Java (JDK 1.8): lenguaje de programación orientado a objetos.
* JSP: tecnología diseñada para crear páginas web con programación en Java.
* Apache Tomcat (9.0.39): servidor web usado en tradicionalmente para proyectos Java por su implementación de servelets o páginas JSP.
* PostgreSQL (10): sistema gestor de bases de datos relacionales que incluye técnicas de orientación a objetos. Es multiplataforma y OpenSource.
* PgModeler (0.9.1): herramienta creada para diseñar las estructuras de datos de nuestras bases de datos en PostgreSQL.
* PgAdmin (4): aplicación de diseño y administración de bases de datos para su uso con PostgreSQL.
* HTML: lenguaje de marcado utilizado para el desarrollo de páginas de Internet.
* JDBC: interfaz de acceso a bases de datos SQL que proporciona un acceso uniforme a una gran variedad de bases de datos relacionales.
* CSS:
* Bootstrap:

# Estructura de paquetes

Siguiendo la estructura del patrón modelo vista controlador, se han creado diferentes paquetes para hacer más legible la organización de las clases. Se ha creado un paquete denominado modelo, el cual contiene las clases pertenecientes al módulo modelo comentadas previamente, como Alumno, AlumnoDAO, Coordinador, CoordinadorDAO… También, se ha creado un paquete donde se encuentra el controlador y otro, denominado “ACUERDOS” donde se encuentran todas las vistas JSP. Además, se ha creado otro paquete adicional denominado config con la clase Conexión, donde se realiza la conexión con la base de datos.

# Acceso y obtención de datos

Los datos estarán alojados en un servidor donde estarán seguros y serán accesibles. Cada vez que se ejecute la aplicación, el usuario debe poder trabajar con los datos con la mayor facilidad posible. Por lo tanto, se han implementado una serie de clases que facilitan esto.

Como hemos ido comentando, dentro del modelo se encuentran una serie de clases con la nomenclatura ClaseDAO, donde se realiza la interacción con la base de datos. Estas clases, al instanciarse, se conectan con la base de datos a través de la clase Conexión y, una vez establecida la conexión, comienzan a operar. Existen dos tipos de métodos: los de escritura y los de lectura y los que combinan ambos.

* Los de escritura consisten en enviar información a la base de datos para realizar una operación con ella sin recibir nada. Habitualmente son los métodos que insertan o modifican registros en la base de datos. Reciben como parámetro variables que contienen la información que se quiere enviar a la base de datos y no devuelven nada. Un caso de ejemplo sería “INSERT INTO TABLE VALUES (?,?,?,?,?)” donde las interrogaciones representan valores (recogidos de los parámetros) que se van a introducir en la base de datos.
* Los que combinan lectura son aquellos que envían información a la base de datos, para que realice una determinada operación y devuelva un resultado. Reciben como parámetro variables que contienen la información que se quiere enviar a la base de datos y devuelven el resultado de la acción. Un caso básico sería “SELECT \* FROM TABLE WHERE = ?”.

Para crear un método, antes de nada, hay que tener clara qué acción ejecutará el método y a cuál de las tres anteriores pertenece. Los métodos que tengan parámetros, los recibirán del controlador, que a su vez los recibe de la vista. Al inicio del método, es necesario instanciar las clases PreparedStatement (necesaria para enviar la acción) y ResulSet (necesaria para ejecutar la acción). Si queremos enviar algo a la base de datos, ya sea para escribir en ella o para una condición de una consulta, debemos utilizar el método setString(POSICIÓN, VARIABLE), donde POSICIÓN es la posición que ocupa la interrogación en la consulta y VARIABLE el contenido que queremos que vaya en esa interrogación. Por ejemplo, si queremos realizar la consulta “select DNI from Alumno where Nombre\_usuario\_Usuario = ?” para obtener el DNI de un alumno a partir de su nombre de usuario. Debemos escribir la línea “ps.setString(1, usuario);” donde el contenido de la variable usuario irá donde la interrogación.

Para obtener el resultado de la consulta, debemos utiliza el método getString(NOMBRE\_CAMPO) donde NOMBRE\_TABLA es el nombre del campo de la base de datos que queremos obtener. En el ejemplo anterior habría que escribir rs.getString("DNI"); para obtener el DNI. Cabe destacar que si alguna consulta devuelve más de un registro, como es habitual, habría que introducir en un bucle while(rs.next()). El resultado de este método lo guardamos en una variable que retornará el método.

Por lo tanto, este es el proceso que hay que seguir para que la información llegue desde la base de datos a la vista, pasando por el controlador, quien decide qué acción se debe realizar.

# Solution | Core J2EE Patterns: Data Access Object Pattern | InformIT

# <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.informit.com%2Farticles%2Farticle.aspx%3Fp%3D1398621%26seqNum%3D3&psig=AOvVaw3_OvoLrzD4GZVKsjyI5169&ust=1624201282223000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCOCVqLj7o_ECFQAAAAAdAAAAABAS>

# Manejo de acciones del usuario

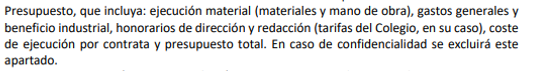
Toda la información se debe introducir y se debe leer a través de una vista. A través de JSP y CSS se crean las vistas, las cuales consisten en unas interfaces sencillas para que los usuarios puedan trabajar con ellas sin dificultad. A través de las etiquetas <input>, para que el usuario introduzca unos datos de entrada o <select>, para que el usuario seleccione una opción entre varias de un desplegable, el usuario puede interactuar con la herramienta y enviar información al sistema.

Sin embargo, esta información no se envía directamente a la base de datos, ya que existe un intermediario que realiza todas las conexiones: el controlador. Para ello, hemos creado una clase que extiende de un Servlet, más concretamente de HttpServlet. Un servlet se utiliza para ejecutar un servidor web y construir páginas web. En esta clase existen dos métodos, el método doGet y el método doPost.

* doGet(HttpServletRequest request, HttpServletResponse response): se instancian las clases DAO para acceder a la base de datos a través de ellas. Se crea una variable RequestDispatcher, la cual recibirá solicitudes y las enviará al recurso JSP. A través del método getParameter() del parámetro request se obtienen variables de la vista para poder realizar las operaciones requeridas en el modelo. Con el método setAttribute() enviamos información a la vista. También, hay una variable “acción” donde se almacena el valor de la acción que llega de la vista y el usuario quiere que se realice.
* doPost(HttpServletRequest request, HttpServletResponse response): le llegan todas las peticiones del usuario y llama al método doGet() para que ejecute estas peticiones.

Cada vez que el usuario quiere realizar una acción en la aplicación, la vista la envía al controlador a través del método POST. Por ejemplo, para expresar al sistema que queremos logearnos en el sistema, la línea necesaria es <form method="POST" action="PropuestaLA?accion=comprobarLogin" autocomplete="off" >.

Presupuesto



A continuación, vamos a detallar el presupuesto aproximado que supone la elaboración de este proyecto. Dentro del presupuesto, es importante diferenciar las diferentes partes en las que se dividirán los costes del proyecto: costes de mano de obra, costes de hardware y materiales y coste de software.

# Coste de mano de obra

Para que un proyecto se realice de forma eficiente y se haga una planificación óptima para cumplir los plazos previstos, es necesario dividirlo en tareas. A estas tareas se les asocia un determinado periodo de tiempo. En nuestro caso, vamos a establecer ese periodo de tiempo a un número de horas y, a su vez, vamos a determinar un coste por hora para establecer el coste total de la mano de obra. Según lo requerido en este proyecto las etapas que se han de realizar son las siguientes:

1. Estudio previo de los requerimientos del cliente: consiste en constantes reuniones con el cliente para entender de la manera más eficiente qué es la herramienta que hay que desarrollar. Aproximadamente, esta tarea puede durar unas 20 horas a un precio por hora de 10€, por lo que el coste estimado de esta tarea sería de 200€.
2. Análisis y especificación de requisitos: una vez hecho el estudio previo, se lleva a cabo la especificación de requisitos. Aproximadamente, esta tarea puede durar unas 40 horas a un precio por hora de 12€, por lo que el coste estimado de esta tarea sería de 480€.
3. Diseño y arquitectura software: tras la especificación de requisitos, comienza el diseño y la arquitectura del sistema, donde se organizan todas las herramientas qué hay que usar, su estructura y cómo se van a usar. Aproximadamente, esta tarea puede durar unas 80 horas a un precio por hora de 20€, por lo que el coste estimado de esta tarea sería de 1600€.
4. Programación e implementación: pasamos a la implementación de la aplicación, donde se va a realizar toda la lógica que hará que el sistema funcione correctamente. . Aproximadamente, esta tarea puede durar unas 120 horas a un precio por hora de 20€, por lo que el coste estimado de esta tarea sería de 2400€.
5. Pruebas y revisión: antes de cerrar el proyecto, hay que realizar una serie de pruebas que confirmen el correcto funcionamiento del sistema. Esto consiste en probar todas las decisiones posibles que pude tomar el usuario al usar la aplicación, por si alguna puede causar problemas. . Aproximadamente, esta tarea puede durar unas 30 horas a un precio por hora de 12€, por lo que el coste estimado de esta tarea sería de 360€.
6. Documentación: una vez terminado el sistema, llegamos a la etapa fundamental de documentación, donde se documenta el código y se realiza una memoria con la explicación del proyecto. . Aproximadamente, esta tarea puede durar unas 10 horas a un precio por hora de 10€, por lo que el coste estimado de esta tarea sería de 100€.

Tras el tiempo y coste estimando de cada etapa, el coste total de la mano de obra sería de 5140€ en un total de 300 horas. La media del precio por hora serían de 17.13€ la hora.

# Coste de hardware y materiales

El coste de los materiales físicos para el correcto funcionamiento del sistema se dividen en dos: implementación y lanzamiento. Para la implementación se necesita el hardware habitual en todos los proyectos: un ordenador, un teclado, un ratón, webcam, micrófono…Y para el lanzamiento se necesita un servidor donde se aloje el sistema.\*\*\*

# Coste de software

Conclusiones y trabajo futuro

Manual de usuario (anexo)

# Manual del alumno

# Manual del profesor

Bibliografía

<http://www.todoerasmus.es/learning-agreement/>

<https://www.deusto.es/cs/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Expires&blobheadername2=content-type&blobheadername3=MDT-Type&blobheadername4=Content-Disposition&blobheadervalue1=Thu%2C+10+Dec+2020+16%3A00%3A00+GMT&blobheadervalue2=application%2Fpdf&blobheadervalue3=abinary%3Bcharset%3DUTF-8&blobheadervalue4=inline%3Bfilename%3D%22GUIAcumplimentarLA_1516_BILATERAL.pdf%22&blobkey=id&blobtable=MungoBlobs&blobwhere=1344380071820&ssbinary=true>

<https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>

<https://sites.google.com/site/sistemadeacademiaelohim/especificacion-de-casos-de-uso>

<http://www.jtech.ua.es/j2ee/2002-2003/modulos/servlets/apuntes/apuntes1_1.htm>

<https://docs.oracle.com/javaee/7/api/javax/servlet/RequestDispatcher.html>

<https://www.occamagenciadigital.com/blog/conoce-las-fases-de-un-proyecto-de-desarrollo-de-software>