Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Análisis y documentación de requerimientos funcionales y estructurales para desarrollo de una aplicación de gestión académica del Centro Universitario de la Defensa (CUD)

Grado en Ingeniería Mecánica

ALUMNO: Carlos Díez de Tejada Faiña

DIRECTORES: José Antonio González Prieto

Santiago Urréjola Madriñán

CURSO ACADÉMICO: 2020-2021

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Análisis y documentación de requerimientos funcionales y estructurales para desarrollo de una aplicación de gestión académica del Centro Universitario de la Defensa (CUD)

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

RESUMEN

En este trabajo se emplea *Unfiyed Modeling Language* (UML) para el análisis y documentación de los requerimientos de una aplicación para la gestión académica del CUD, así como de todas las actividades que se desarrollan en la Escuela Naval Militar fruto del currículum necesario para ser Oficial de la Armada.

Para ello, en primer lugar, se exponen los objetivos de este trabajo, y a continuación se realiza una introducción al lenguaje estandarizado UML, explicándose los tipos de diagramas que lo componen así como sus usos y significados.

Seguidamente se aplican las técnicas de modelado expuestas para analizar las necesidades funcionales y estructurales de una posible aplicación de gestión para la ENM, que sea escalable respecto al desarrollo de nuevas funciones y que permita la integración de modificaciones de forma estandarizada.

Tras analizar la gestión actual de la ENM y sus necesidades respecto a la gestión de la información académicas, se ha creado una propuesta de modelado en UML que permite, en un futuro, realizar el desarrollo de una aplicación orientada a objetos que resuelva las necesidades de gestión del centro.

PALABRAS CLAVE

Análisis, Evento, UML, Horario, Aplicación

AGRADECIMIENTOS

A mis padres, hermanas y familia, por darme apoyo moral estos cinco años.
A mis tutores José Antonio González Prieto y Santiago Urréjola Madriñán, en especial a José Antonio por su tiempo y dedicación.
Al Secretario de Estudios, el Capitán de Fragata Don Manuel Villanueva Torres por la información proporcionada.

Contenido

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN Y OBJETIVOS

El desarrollo de soluciones de gestión de información es uno de los campos de la ingeniería en donde los procesos de planificación, documentación y análisis de las necesidades son fundamentales para resolver un problema propuesto. Sin embargo, es habitual iniciar el proceso de desarrollo de software sin contar con este tipo de herramientas, lo cual, unido a la gran cantidad de opciones de programación genera, de forma habitual, una serie de problemas como los siguientes:

1. Se desarrollan múltiples soluciones que no siguen una estructura de datos común, lo cual requiere realizar trabajos de adaptación cuando se necesita hacer colaborar a las diferentes soluciones desarrolladas. El problema empeora cuando, además de no emplear una estructura de datos y formatos prefijados, se emplean diferentes bases de datos para realizar su almacenamiento.

2. Se desarrollan soluciones que emplean diferentes lenguajes de programación, protocolos y *frameworks* de desarrollo, lo cual encarece mucho el mantenimiento de los sistemas debido a la necesidad de disponer de profesionales expertos en varios entornos y la necesidad de realizar complejas tareas de integración en caso de necesitar la colaboración entre aplicaciones.

3. No se dispone de una documentación unificada que permita analizar nuevas necesidades, por lo que se dificulta la integración de nuevos desarrollos con las soluciones ya existentes.

4. Uno de los problemas habituales que suelen ocurrir, se debe al desarrollo de aplicaciones sin mantener una cohesión, respecto a las interfaces que el usuario debe manjar (además de tener que usar diferentes contraseñas en diferentes plataformas) lo cual dificulta su trabajo y genera, por tanto, una sensación de rechazo por parte de los usuarios al no disponer de un entorno integrado en donde poder realizar todos sus trámites.

Debido a los problemas generados por el desarrollo de soluciones informáticas sin disponer de una base común para la documentación y el desarrollo de las aplicaciones, se entiende la necesidad de establecer una marco común y unificado. El objetivo es que en adelante, el proceso de desarrollo de software, no solo resuelva las funcionalidades requeridas, sino que además lo haga siguiendo un modelo de trabajo previamente establecido en lo que respecta al análisis, documentación y uso de herramientas de desarrollo.

Por tanto, los objetivos de este trabajo son los siguientes:

- Desarrollar un análisis general de las funcionalidades relacionadas con el sistema de gestión de la información relativa a las actividades académicas de la Escuela Naval Militar y del Centro Universitario de la Defensa.

- Documentar el análisis realizado empleando estándares que faciliten el entendimiento entre los responsables de las organizaciones y los programadores que finalmente creen las soluciones de software.

- Proponer una estructura técnica para el desarrollo de nuevas soluciones que establezca las herramientas (bases de datos, lenguajes de programación, …) y los métodos de trabajo que deben ser empleados para realizar el análisis y documentación de las nuevas necesidades.

Para lograr estos objetivos se propone el uso del lenguaje de modelado UML *(Unified Modeling Language)* que representa el estándar actual creado para este tipo de tareas, tal y como se indica, por ejemplo, en la ***Wikipedia*** [1]:

"El lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional, Rational Unified Process o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que programación estructurada es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML solo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. "

1.1 Análisis de soluciones software previamente desarrolladas

Durante los años de historia del centro, se han realizado diversos esfuerzos para crear soluciones informáticas a las necesidades de gestión de la información académica y otros ámbitos relacionados con las actividades propias de la Escuela Naval Militar. Algunos ejemplos de este tipo de soluciones desarrolladas durante el transcurso de TFG son las siguientes:

1. "*Desarrollo de una herramienta de apoyo a la planificación horaria de la ENM*" (2018-2109) [2].

2. "*Desarrollo automático de la sábana de la ENM-CUD*" (2018-2109) [3].

3. "*Sistema de monitorización y control de inventario de fusiles HK en el pañol del condestable de la ENM*" (2017-2018) [4].

Si bien estos desarrollos han tenido un objetivo orientado a la formación como parte del desarrollo de los TFG, constituyen una buena aproximación a la problemática de intentar resolver problemas sin una base de documentación y un protocolo de desarrollo unificado, con el problema añadido de disponer de un tiempo muy limitado para crear aplicaciones que se integren en sistema de gestión común.

En este sentido, la disposición de una herramienta previa de análisis y documentación permitiría que estos desarrollos se realicen asegurando que puedan ser integrados entre sí, de forma que estos trabajos tengan una utilidad práctica real una vez finalizados. Si además, se consigue que se realicen empleando las mismas herramientas y *frameworks*, facilitarán el mantenimiento y las propuestas de TFG que mejoren las capacidades de las soluciones creadas y generen aplicaciones con una utilidad práctica final.

1.2 UML

Tal y como se ha comentado, el ***Unified Modeling Language*** es un lenguaje gráfico de modelado de sistemas aprobado por la ISO como ISO/IEC 19501 que representa el estándar en lo que respecta a la documentación de procesos.

Dicho estándar abarca muchos tipos de sistemas, aunque en este trabajo nos centraremos en su aplicación para el desarrollo de soluciones software. En este caso, el UML se emplea como una herramienta que permite plasmar en un sistema gráfico las necesidades funcionales de la aplicación que debe desarrollarse, de forma que tanto el cliente (que genera la petición para resolver una necesidad) como el programador (que finamente realiza dicho desarrollo) pueden entender y emplear como medio de comunicación y documentación del proyecto.

UML no es un lenguaje de programación, es un intermediario entre la realidad del problema a resolver y el informático que programa la solución, siendo el trabajo del analista la creación y gestión de este entendimiento.

Debido a que hay diversos lenguajes de programación y diferentes formas de programar en sí, la idea es proponer una serie de objetos, funciones y cómo se estructura de forma interna el programa. Para que así cualquier programador lo desarrolle siguiendo un mismo patrón de razonamiento y estructuras funcionales. De esta forma, se facilita el proceso de mantenimiento del software al evitar tener que reescribir código que resuelva la misma necesidad en varias ocasiones y se permite que un programador puede entender mejor el funcionamiento para la creación de extensiones sobre el código original.

Por este afán de crear un estándar para la futura creación de una aplicación que se adecúe a las necesidades de la ENM se ha elegido UML para plasmar el cómo ha de funcionar la aplicación de la gestión académica.

1.3 Objetivos

El ánimo de este trabajo es estudiar las necesidades de la gestión académica y definir todos los requerimientos funcionales y estructurales para que inicialmente, sirva de norma para el desarrollo de la aplicación de gestión de horarios, pudiéndose añadir más funcionalidades de interés para la gestión de la enseñanza en futuras actualizaciones. Por último, cabe mencionar que este trabajo es una propuesta para el desarrollo de la futura aplicación, lo que quiere decir que cualquier aspecto puede ser modificado, ampliado o reducido en función a nuevos puntos de vista o necesidades que se quieran cubrir.

2 ESTADO DEL ARTE

2.1 Historia de UML

Antes de introducir en detalle el lenguaje UML es conveniente poner en contexto el porqué de su creación.

Ilustración 2-1 Logo de UML([5])

2.1.1 *Antecedentes*

Previamente a la existencia de UML existían varias metodologías de modelado orientadas a objetos, pero cada una de estas metodologías resolvía el modelado de una forma distinta. Los métodos más empleados eran *Object Modeling Technique* de *James Rumbaugh*, empleado para el análisis orientado a objetos; *Object-Oriented Software Engineering* de *Ivar Jacobson*, empleado para el diseño orientado a objetos y por último el *método Booch* de *Grady Booch*. Todos estos métodos tenían como finalidad facilitar el desarrollo de software, bien en la fase de análisis previo o en la fase de diseño y desarrollo. El objetivo de estos métodos es crear una técnica que ayude a modelar un sistema, a establecer qué clases de elementos tiene y cómo interactúan entre sí los elementos que componen el sistema.

2.1.2 *Origen*

El problema asociado a estos métodos previos era que, si bien abordaban el mismo problema, lo hacían desde perspectivas diferentes. Debido a esto surgió la necesidad llegar a un punto de acuerdo por medio de la creación de un lenguaje común. En 1996 *Rational Software Corporation* encargó a "los tres amigos" (*Jacobson, Rumbaugh y Booch*) que desarrollaran un lenguaje unificado de modelado. Para

llevarlo a cabo se organizó un consorcio internacional ese mismo año, *UML Partners*, bajo la tutela de estos "tres amigos". Lo que consiguieron fue completar las primeras especificaciones de la primera versión UML, UML 1.0, que presentaron ante el *Object Management Group* (es el consorcio dedicado a establecer los estándares de tecnologías orientadas a objetos) en enero de 1997. Para agosto de ese mismo año presentaron UML 1.1 que fue adoptado como estándar en noviembre. Desde entonces se ha ido actualizando UML hasta su última versión, la 2.5.1 en 2015.

2.1.3 *Modelio*

Debido a que UML se basa en la creación de gráficas que representan de forma clara y sencilla la arquitectura funcional y estructural de un sistema, es necesario disponer de un software que nos permita crear esos diagramas y almacenarlos de forma adecuada. *Modelio* es el programa que se ha seleccionado para crear la documentación de la aplicación asociada a este TFG, principalmente por sus características avanzadas de diseño y por ser software de código libre. Tiene su origen en 2009 de la mano de *Modeliosoft* en París [6]. Tiene las ventajas de que es un entorno de modelado general que trabaja diversos estándares además de UML como BPMN (orientado a los procesos de negocio) y se le puede añadir módulos adicionales. Durante el desarrollo de este TFG se va a emplear la versión, Modelio v4.1 que soporta el estándar UML2.5.

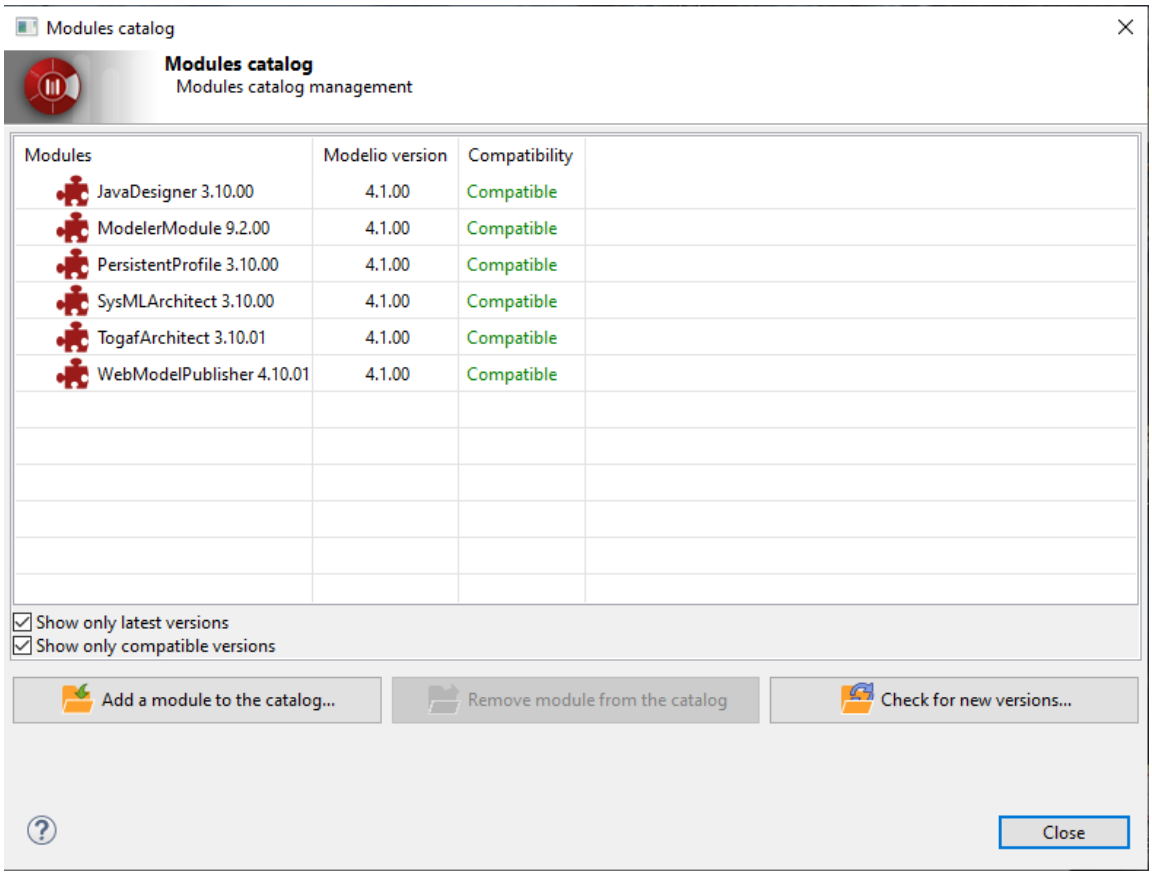

Ilustración 2-2 Módulos que ofrece Modelio ([7])

2.2 Introducción a UML

UML [8] es una herramienta que se utiliza para realizar el modelado de procesos, sistemas y software, para lo cual describe el proceso a documentar por medio de una serie de diagramas

gráficos. De esta forma cabe preguntarse, ¿cuáles son los diagramas de UML y cuál es su significado? En UML cada diagrama representa la visualización de un cierto aspecto concreto de modelado que tiene un propósito específico. Los diagramas a su vez tienen su propia notación o nomenclatura de uso, la cual se explicará en los próximos puntos del estado del arte.

Los principales usos de UML en el ámbito de la programación son:

- Definición de la arquitectura de un sistema (aplicación, sistemas de soporte físicos, comunicaciones, etc.).
- Definición del comportamiento de un sistema, es decir, que cual es la dinámica de la lógica del sistema que le permite cumplir con las funciones a resolver.
- Definición de la estructura de una aplicación a desarrollar.
- Definición de la estructura de los datos que van a almacenar los contenidos de información gestionados por la aplicación a desarrollar.
- Creación de las especificaciones de un sistema, con independencia del dominio de la solución software propuesta.
- Facilitar la creación y el posterior mantenimiento y actualizaciones de código.

Se trata, por decirlo de una forma práctica, de preparar los planos detallados del edificio antes de iniciar su proceso de construcción.

2.2.1 *Orientación a objetos en UML*

UML tiene una clara orientación al desarrollo de soluciones basadas en el paradigma de la programación orientada a objetos (en lugar de la programación orientada a funciones) que permita la creación estructurada de complejos sistemas informáticos de gestión. En UML se diferencia entre clases y objetos de forma que mientras que las clases son una representación abstracta de las propiedades y funciones de un objeto, los objetos son instancias particulares de estas clases o categorías. De esta forma, podemos pensar en la clase coche como una entidad abstracta (vehículo con 4 ruedas) y cada uno de los coches individuales con que trabaje la aplicación, representa un objeto de la clase coche. Como es habitual en la programación orientada a objetos, estas categorías o clases se componen de atributos y operaciones que definen sus capacidades funcionales, de forma que los atributos de una clase son sus propiedades y las operaciones son las funcionalidades de esta. Por otra parte, las clases necesitan interactuar entre ellas para generar secuencias lógicas de cooperación, lo cual genera las **denominadas relaciones ente clases**. En UML se trabaja con los siguientes tipos de relación, que vemos resumidas a continuación:

- **Herencia/generalización**: establece una relación jerárquica en la cual una clase tiene una serie de subclases que heredan su comportamiento básico y lo extienden a su dominio particular de actuación.

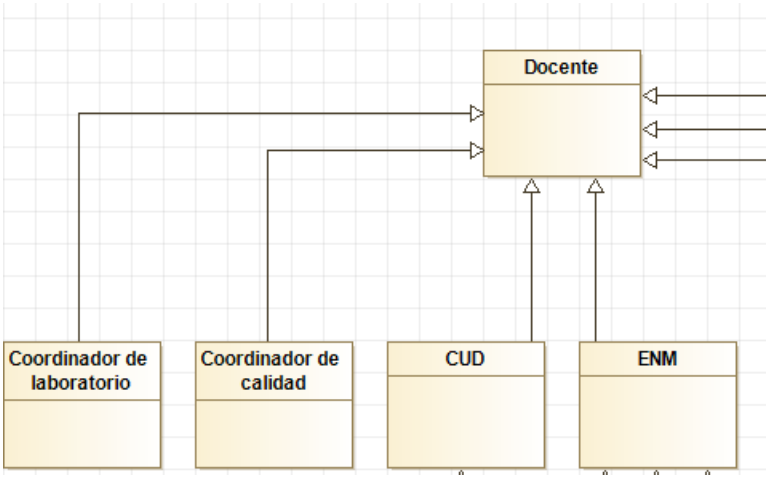

Ilustración 2-3 Ejemplo de herencia (elaboración propia)

- **Asociación**: muestra una relación entre clases que compartan una necesidad de colaboración mutua. En la imagen de ejemplo, una noticia es de un tipo de noticia específico, y un tipo de noticia puede estar asociado a un número de noticias indeterminado (de ahí el uso del * en la relación). De esta forma el tipo de noticia actúa como un elemento de categorización, que permite agrupar las noticias en función de las afinidades temáticas en sus contenidos. El número de elementos que determinan el tipo de asociación se llama multiplicidad/cardinalidad.

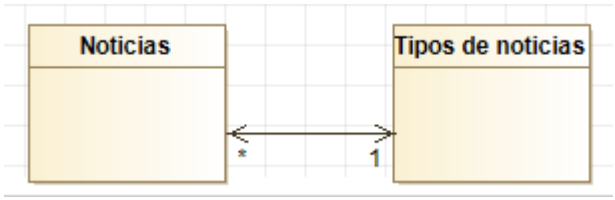

Ilustración 2-4 Ejemplo de asociación (elaboración propia)

- **Agregación**: establece una relación clase contenedora/clases contenidas, donde cada clase contenida de la relación existe por si misma con independencia de la clase contenedora, a diferencia de lo que ocurre con la herencia De esta forma la relación de agregación se asocia a un proceso de construcción de una entidad como el conjunto formado por la unión o agregación de otras entidades o clases.

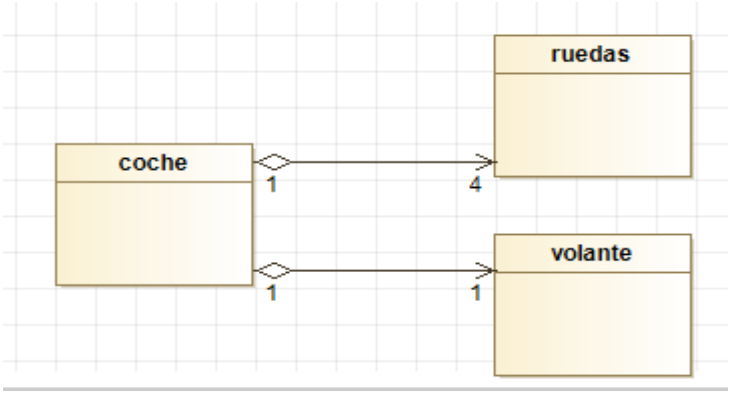

Ilustración 2-5 Ejemplo de agregación (elaboración propia)

- **Composición**: En este tipo de relación tenemos las mismas circunstancias que en el caso de la agregación, pero con la diferencia de que en esta relación las clases contenidas no existen por sí mismas, solo existen en el conjunto que forman con su contenedor, de forma que, si se destruye el conjunto, desaparecen las partes que lo forman. En la siguiente figura vemos un ejemplo de esta situación en el caso de una casa que se compone de distintas habitaciones, que por si solas no tienen entidad propia.

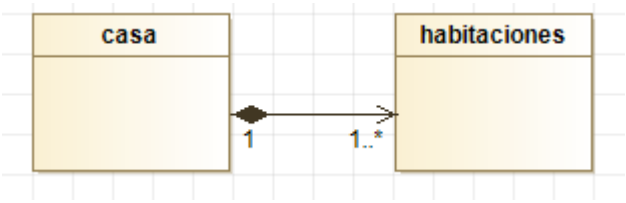

Ilustración 2-6 Ejemplo de composición (elaboración propia)

- **Ligado**: la información que usa una clase la contiene otra.
- **Colaboración**: las dos clases trabajan juntas
- **Acto**: una clase actúa sobre la otra.

2.2.2 *Tipos de diagramas*

Tras ver los tipos de relaciones entre clases que se manejan en UML, el siguiente paso consiste en ver qué tipos de diagramas existen y cuáles son sus propiedades. Los diagramas en UML se pueden dividir en dos categorías principales.

2.2.2.1 *Diagramas de estructura*

Los diagramas de estructura ofrecen una visión estática de los elementos del sistema. Los diagramas englobados en este tipo son los diagramas de clases, objetos, componentes, despliegue y paquetes.

2.2.2.1.1 Diagrama de clases

Se trata de diagramas estáticos que establecen la clasificación de los elementos que componen el sistema. En este tipo de diagramas la información se representa en forma de rectángulos. teniendo en cuenta los tipos de relación entre clases que se comentaron en el punto 2.2.1 Orientación a objetos en UML.

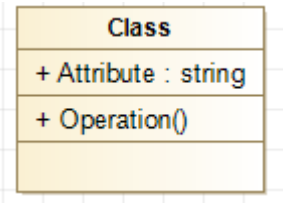

Ilustración 2-7 Ejemplo de clasificador (elaboración propia)

Una clase es una categoría o grupo de elementos que tienen atributos (propiedades o valores internos de la clase) y acciones similares, por lo que una clase está compuesta por su nombre, atributos y

operaciones o funciones que la clase puede ejecutar. Los atributos y operaciones pueden indicar su nivel de protección y visibilidad mediante los siguientes indicadores:

- Pública (+): cualquier clase puede acceder a ese atributo/operación.
- Privado (-): solo la clase contenedora tiene acceso.
- Protegido (#): solo pueden acceder otras clases de la misma cadena de herencia.
- Paquete (~): solo pueden acceder las clases que estén en el mismo paquete.

El formato del diagrama asociado a una clase es, empezando desde arriba: nombre de la clase, atributos de la clase y las operaciones que realiza. Los tipos de datos que se que pueden emplear en los atributos son los siguientes:

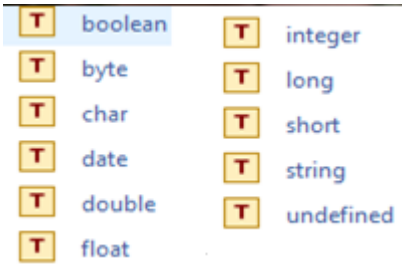

Ilustración 2-8 Tipos de atributo ([7])

2.2.2.1.2 Diagrama de objetos

Teniendo en cuenta que un objeto se define como la instancia de una clase, un diagrama de objetos puede ser visto como una instancia de un diagrama de clases. Los diagramas de objetos describen la estructura estática de un sistema en un momento particular. De esta forma, se trata de diagramas similares a los diagramas de clases, pero con la diferencia de que en este caso se representan instancias particulares de las clases y son usados para generar casos específicos de los diagramas de clases que permiten evaluar su desarrollo.

2.2.2.1.3 Diagrama de paquetes

En algunas ocasiones se encontrará con la necesidad de organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, se pueden agrupar en un paquete, que se representa por una carpeta tabular. De esta forma, los diagramas de paquetes sirven para organizar al agrupar elementos de un diagrama UML (clases, casos de uso, actores, etc.) relacionados entre ellos o que son similares. Gracias a esto se puede mostrar una vista de alto nivel del sistema al mostrar los componentes principales, cómo se relacionan entre ellos y ayuda a dividir un sistema complejo en módulos más manejables.

Los elementos dentro de un paquete tienen asociada una visibilidad, que puede ser:

- Público (+): ese elemento puede ser usado por elementos externos al paquete.
- Privado (-): ese elemento sólo puede ser usado por elementos del paquete.

Además, los paquetes tienen varias formas de relacionarse entre sí:

- **Dependencia**: un paquete depende de los elementos del otro para funcionar.
- **Generalización**: establece una relación de herencia entre los paquetes.
- **Refinamiento**: un paquete mejora (”refina”) otro, ya que tiene los mismos elementos y algunos más.

- **Acceso**: un paquete puede hacer uso de los elementos públicos del otro.
- **Importación**: todos los elementos de un paquete pasan a estar disponibles para otro.
- **Fusión**: fusiona dos paquetes en una nueva unidad individual

2.2.2.1.4 Diagrama de componentes

Un diagrama de componentes describe la organización de los componentes físicos de un sistema, de forma que cada componente es una parte modular del sistema que interactúa y provee servicios a otros componentes mediante interfaces.

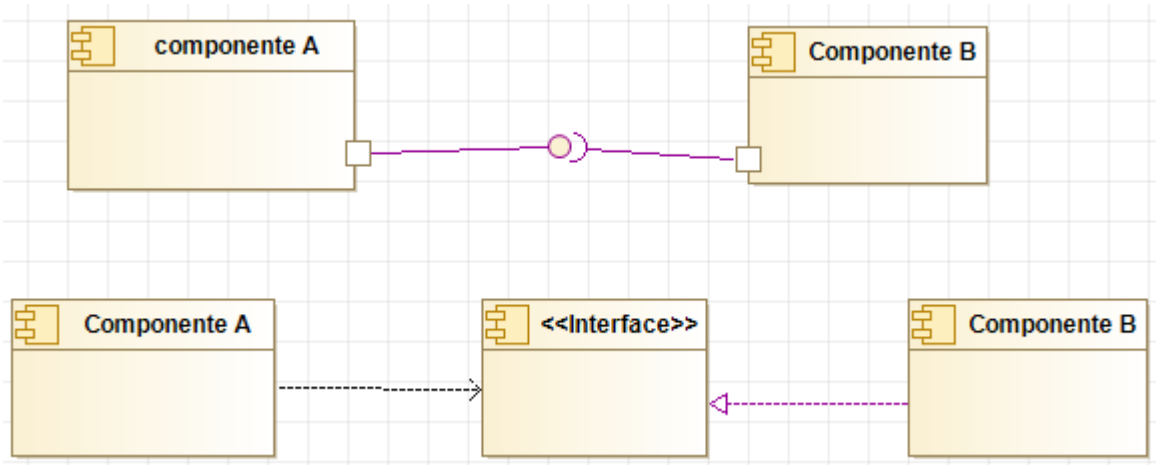

Ilustración 2-9 Ejemplo de diagrama de componentes (elaboración propia)

En la Ilustración 2-9 se muestran dos formas de representar la misma situación. En la primera el componente A provee de una interfaz a B mediante el círculo que tiene unido por una línea, y B se sirve de la interfaz por el semicírculo que rodea a la interfaz de A.

En la segunda forma hay que apoyarse en otro elemento del tipo estereotipo *interface*, que indica que entre ambos componentes existe un tercer elemento que genera las funciones de comunicación.

La flecha que conecta A con interface representa realización, es decir que A es quien realiza/provee la interface. Finalmente, la unión de B con la interface indica una relación de dependencia.

Los estereotipos más comunes son los siguientes:

- *Application*: para interfaces de usuario.
- *DataStore*: localización persistente de los datos.
- *Document*.
- *Entity*: puede guardar u obtener información.
- *Executable*: software que puede ser ejecutado.
- *File*: un archivo.
- *Infrastructure*: componente técnico del sistema.
- *Library*: biblioteca de funciones u objetos.
- *Process*: depende de un estado.
- *Realization*: implementa un componente.
- *Service*.
- *Source Code*: archivo con código fuente.
- *Specification*: solo contiene interfaces y no implementaciones.
- *Subsytem*.

- *Table*: una tabla de datos.
- *Webservice*.
- XML DTD: documento de definición de tipos de XML.

2.2.2.1.5 Diagrama de despliegue

Este diagrama asocia software con hardware, de forma que modela la relación entre los elementos lógicos y los físicos. En estos diagramas hay nodos y artefactos, de forma que los nodos ejecutan los artefactos. Estos artefactos son archivos ejecutables, es decir, el software; y los nodos pueden ser de tipo físico *(device*) o software que contiene a otro software (*execution enviroment*) como un sistema operativo, por ejemplo.

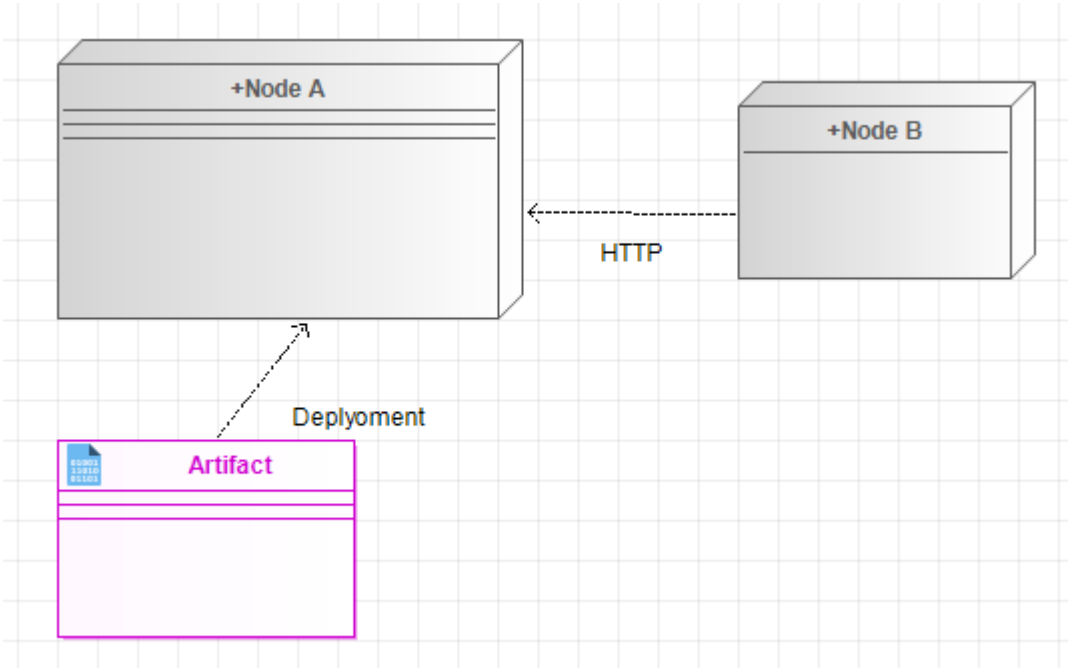

Ilustración 2-10 Ejemplo de diagrama de despliegue (elaboración propia)

2.2.2.2 *Diagramas de comportamiento*

Los diagramas de comportamiento dan un enfoque dinámico de cómo funcionan entre sí los elementos del sistema. Estos diagramas son los de actividad, casos de uso, de máquina de estado, de comunicación y de secuencia.

2.2.2.2.1 Diagrama de casos de uso

Este diagrama nos provee una vista de alto nivel de las interacciones que los diferentes actores (o usuarios del sistema) tiene con el sistema, de forma que se encarga de modelarla visión global de las acciones que los usuarios pueden desarrollar en el sistema que se está analizando. Esto debe realizarse desde la perspectiva de los actores, que pueden ser los usuarios de la aplicación o también otros sistemas o aplicaciones que interactúan con el sistema que está siendo analizado. En resumen, con estos diagramas se visualiza lo que hace el sistema desde una perspectiva general si atender a los detalles de la implementación

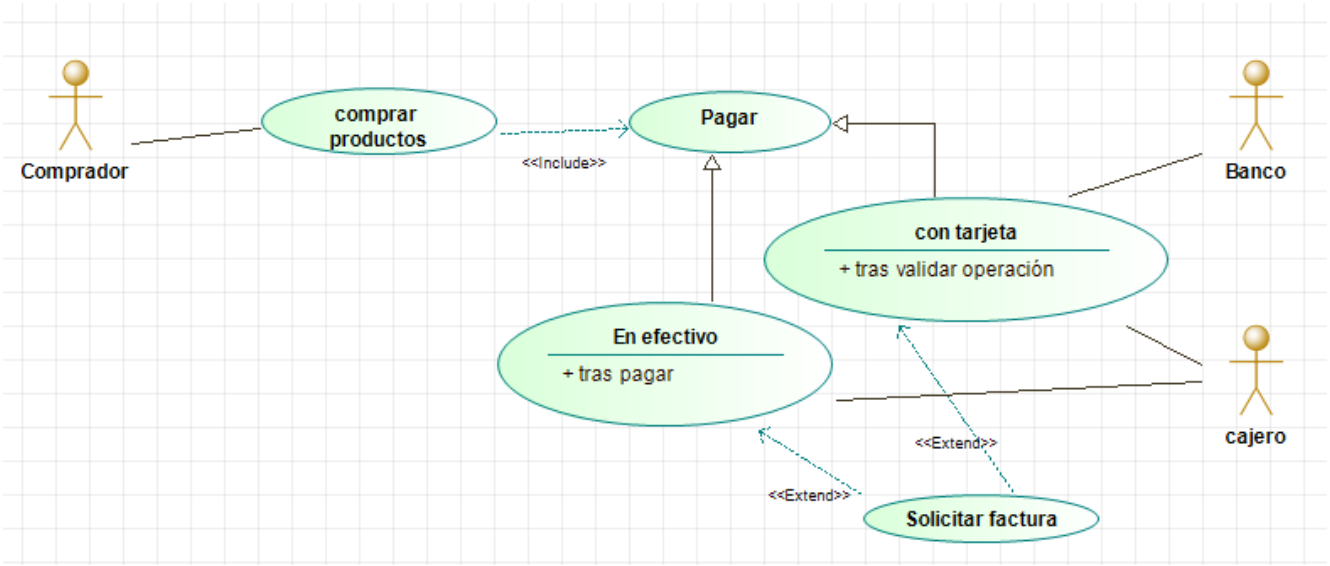

Ilustración 2-11 Ejemplo de diagrama de casos de uso (elaboración propia)

En la Ilustración 2-11 se muestra el proceso asociado a la realización de compra en una aplicación de gestión comercial, en donde distinguimos los siguientes actores: comprador, banco y cajero. Cada uno de estos actores está unido con una línea con las funciones del sistema a las que puede acceder. Por esto el comprador es el que compra y paga, el cajero el encargado de cobrarle en efectivo/tarjeta y el banco en el caso de con tarjeta es el que lleva a cabo la operación a través del lector de tarjetas. Cada acción/evento se representa dentro de una elipse, de forma que estas acciones se pueden relacionar entre si de diversas formas:

- **Inclusión**: establece una relación de dependencia, de forma que hasta que no finalice la acción incluida no finaliza la contenedora. En el ejemplo el comprador no finaliza la compra hasta que no haya pagado.
- **Generalización/herencia**: indica una relación entre una acción más genérica y sus formas más específicas de llevarla a cabo. En el ejemplo la acción general es pagar, que puede ser realizada en efectivo o con tarjeta.
- **Extensión**: a diferencia de la inclusión, en este caso la acción extendida es opcional. En el ejemplo tanto si se paga con tarjeta o en efectivo se puede solicitar factura, pero no es obligatorio. En este tipo de relación cabe destacar el *extension point*, que es el punto dentro de la acción tras el que se puede llevar a cabo la extensión. En el ejemplo los extension points son tras pagar y tras validar operación.

2.2.2.2.2 Diagrama de máquina de estados

En cualquier momento, un objeto se encuentra en un estado particular, por lo que los diagramas de máquina de estados muestran el comportamiento dinámico de los diferentes estados por lo que un objeto puede pasar. Permiten modelar todos los posibles estados finitos de una máquina/sistema, y los hay de dos tipos:

- **De comportamiento**: se centran en la implementación específica de los objetos y cómo se comportan, es decir, establece, para cada objeto, el espacio de los posibles estados en que puede estar, así como las rutas en que el objeto puedo cambiar de un estado a otro
- **De protocolo**: describe la secuencia de eventos sin centrarse en el comportamiento de los objetos.

En la notación de estos diagramas, los nodos representan los estados y los conectores son las transiciones que llevan al cambio de estados. Estas transiciones se corresponden con una acción que

lleva al siguiente estado, y son desencadenadas por lo que se denomina como evento (pulsar un botón, se acaba el contador o la activación del estado anterior, por ejemplo). Por último, las transiciones pueden tener una condición (llamada *guard*) que debe estar activa para que el evento desencadene la transición de estados. Los estados pueden estar activos o no, y en caso de estar activos pueden tener hasta tres tipos de acciones internas:

- **Entry**: es la primera acción que se realiza al activarse un estado. Suponiendo un estado "*escribir",* su entry sería coger lápiz.
- **Do**: es la acción que principalmente se lleva a cabo en el estado, siguiendo con el ejemplo anterior el do sería escribir las letras.
- **Exit**: es la última acción del estado, en el ejemplo "*escribir"* sería soltar el lápiz.

2.2.2.2.3 Diagrama de actividades

Un diagrama de actividades muestra la naturaleza dinámica de un sistema mediante el modelado del flujo ocurrente de actividad en actividad. Asimismo, una actividad representa una operación o función en alguna clase del sistema. Típicamente, los diagramas de actividad son utilizados para modelar el flujo de trabajo interno de una operación, por lo que son similares a los diagramas de flujo.

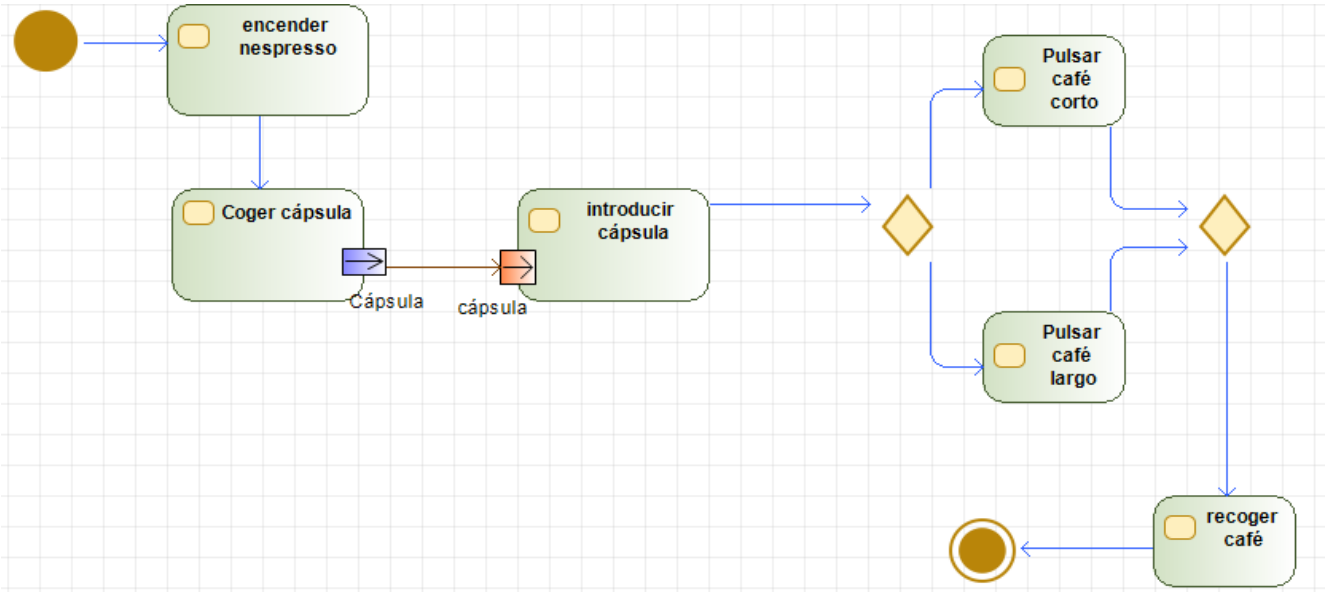

Ilustración 2-12 Ejemplo de diagrama de actividad (elaboración propia)

Como se ve en la Ilustración 2-12, los nodos representan acciones y los gráficos con forma de rombo emplean para abrir o cerrar caminos alternativos. Los cuadrados (rojo y azul) se llaman pins, y sirven para mostrar la salida de un objeto desde una acción para ir como input a la siguiente. También se pueden añadir nodos objeto, de forma rectangular; que sirven para indicar que se produce un cambio de estado en un objeto a causa de una acción.

2.2.2.2.4 Diagrama de comunicación

Muestra el orden en el que suceden los mensajes, centrándose en los enlaces entre objetos.

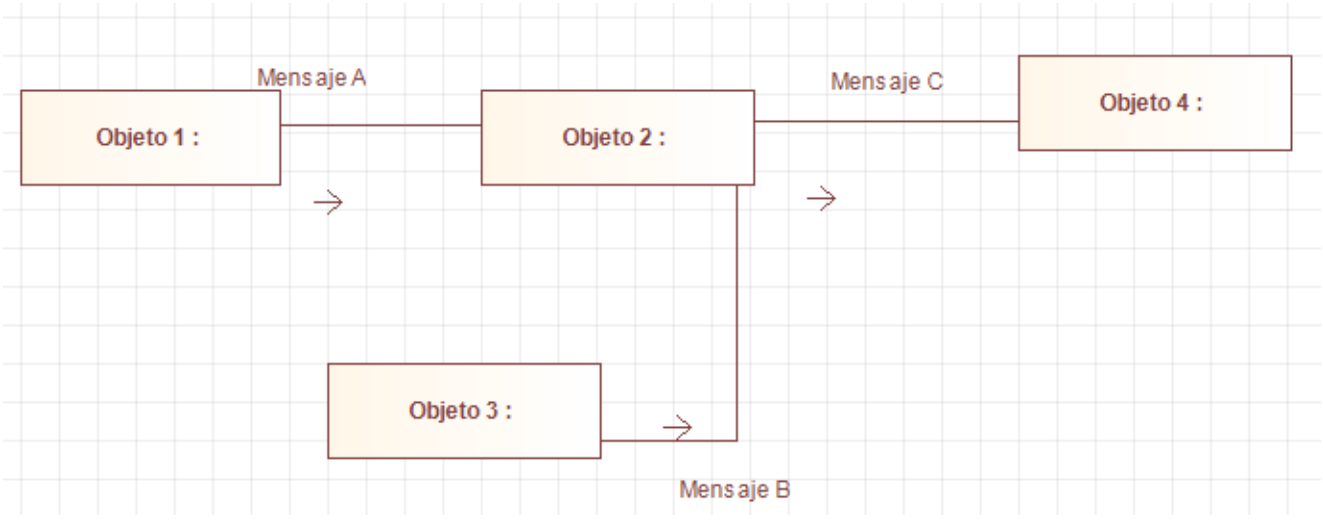

Ilustración 2-13 Ejemplo de diagrama de comunicación (elaboración propia)

2.2.2.2.5 Diagrama de secuencia

Los diagramas de clases y los de objetos representan información estática. No obstante, en un sistema funcional, los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos. Es decir, sirven para representar la comunicación entre los objetos a medida que transcurre el tiempo. El diagrama se lee de arriba abajo siendo el tiempo que transcurre representado en el eje vertical. Cada línea que desciende de un objeto se denomina hilo.

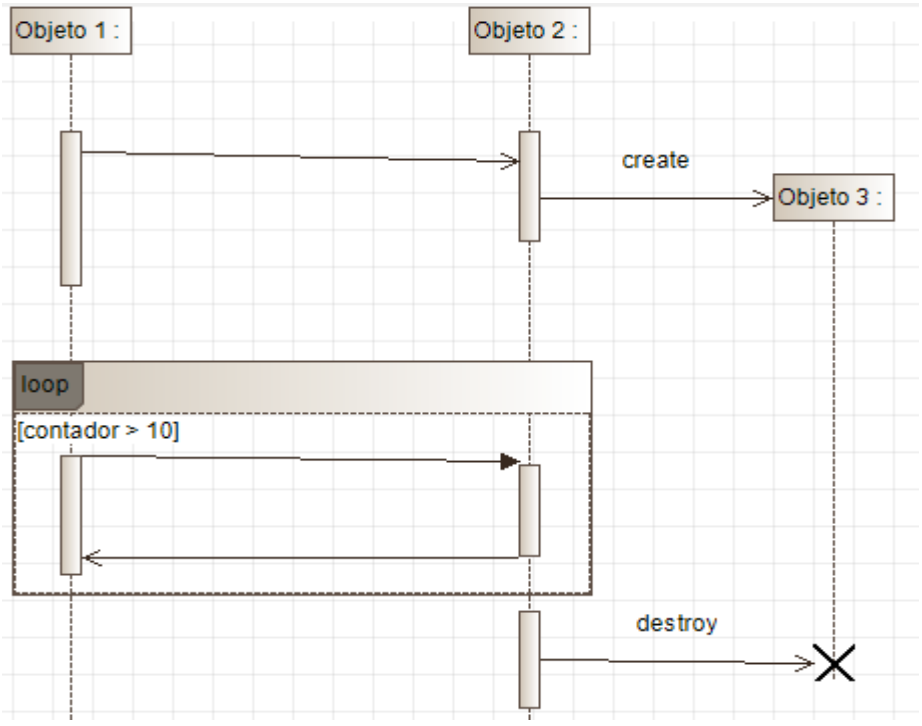

Ilustración 2-14 Ejemplo de diagrama de secuencia (elaboración propia)

Comenzando desde arriba, la primera flecha indica un **mensaje asíncrono**, el cual se envía sin esperar respuesta. A continuación, el objeto 2 crea el objeto 3. Lo siguiente es un marco de tipo *loop*, que indica que las acciones que contiene se repetirán hasta que se cumpla la condición de salida del bucle (entre corchetes). Además de *loop* los marcos pueden tener otras funciones:

- **Opcional** (opt): en caso de que su condición se cumpla, se ejecuta. En caso contrario no pasa nada y se sigue adelante.
- **Alternativa** (alt): el marco se divide en dos partes cuyas condiciones de ejecución son excluyentes, es decir o se realiza una o la otra.
- **Paralelas** (par): el marco se divide en dos partes que suceden simultáneamente.
- **Critical** : indica una región crítica que solo puede ser realizada por un hilo a la vez.

Dentro del loop se encuentra un **mensaje síncrono, es decir, que** espera respuesta por parte del receptor, y es representado por una flecha de punta rellena. Por último, el objeto 2 elimina al objeto 3 con un mensaje destroy.

2.3 Modelo 4+1

Esta forma de realizar el modelado de sistemas que fue creada por *Philipe Krutchen,*. La idea principal es la de describir el problema desde distintos puntos de vista concurrentes.

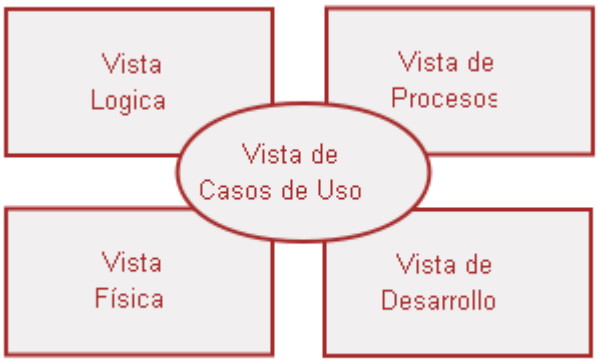

Ilustración 2-15 Modelo 4+1 [9])

2.3.1 *Vista lógica*

Se centra en la estructura y en la forma en que interactúan entre sí las partes del sistema. Los diagramas de UML que tienen uso en esta vista son:

- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de organización.
- Diagrama de máquina de estados.

2.3.2 *Vista de procesos*

Muestra una visión dinámica del sistema, sus procesos y como se comunican entre ellos. Aquí se define la concurrencia, rendimiento y escalabilidad del sistema. El diagrama de UML asociado es el diagrama de actividad.

2.3.3 *Vista física*

Describe el sistema en el ámbito de la topología de elementos del software sobre la capa física además de como estos componentes están conectados. Esta vista se corresponde con el diagrama de despliegue de UML.

2.3.4 *Vista de desarrollo*

Da un enfoque desde el puno de vista de un programador, muestra los módulos y componentes del sistema en el ámbito del sotware. Para plasmar esta vista en UML se emplea:

- Diagrama de componentes.
- Diagrama de paquetes.

2.3.5 *Vista de casos de uso*

Es la vista +1, y la más importante. Porque da una vista desde fuera del sistema, desde el punto de vista del usuario. Esto permite identificar y validar el diseño, ya que en esta vista se ve lo que puede hacer cada usuario en el sistema, es decir, las funcionalidades que el sistema debe ofrecer al usuario en función de su perfil de trabajo. En UML se representa con el diagrama de casos de uso.

3 DESARROLLO DEL TFG

3.1 Organización de la Escuela Naval Militar

Antes de entrar en detalle se va a exponer la organización de la ENM y los órganos dedicados a la gestión de la enseñanza, ya que antes de explicar cómo se organiza el curso académico es conveniente tener una visión de quiénes lo hacen y sus relaciones de dependencia.

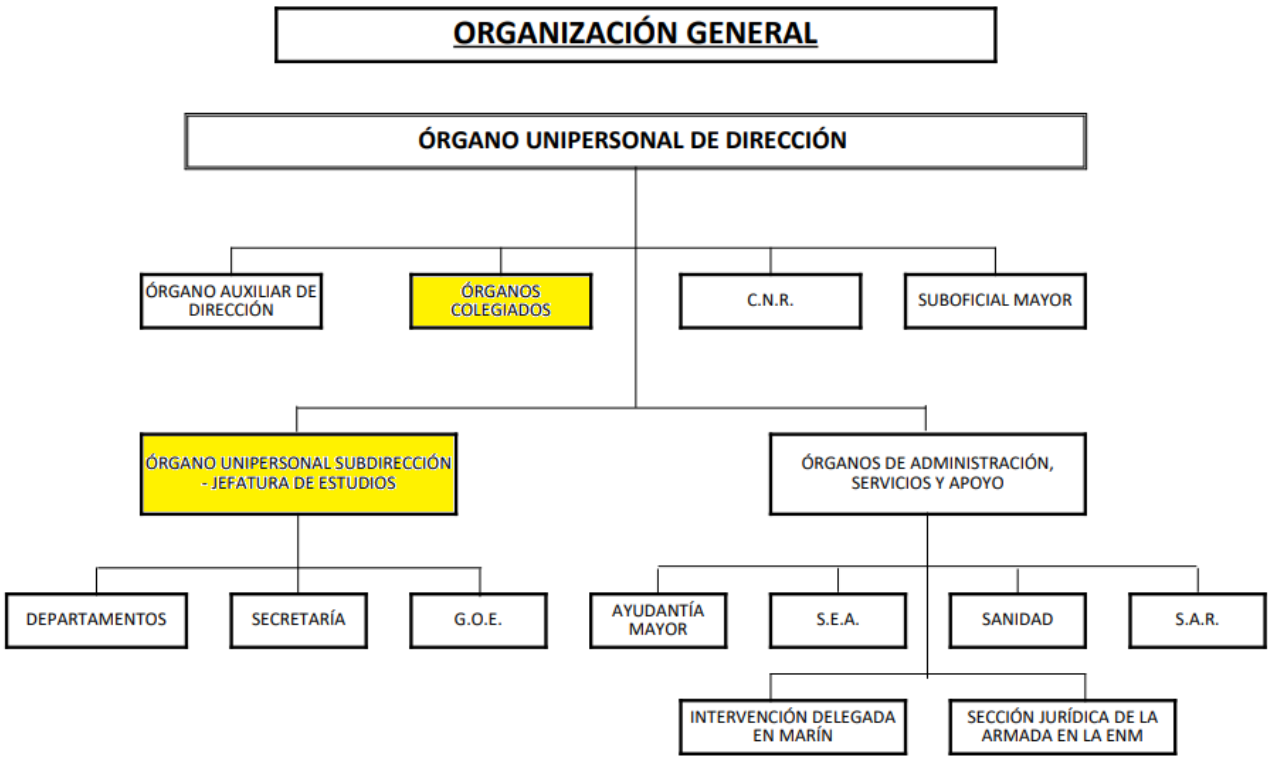

Ilustración 3-1 Organización general de la ENM ([9])

Al mando de la Escuela Naval Militar está un Capitán de Navío, Comandante Director, quien es la máxima autoridad del centro. De él dependen una serie de órganos para la gestión de la actividad de la escuela, en lo relativo a la gestión de la enseñanza se ha destacado en amarillo los Órganos Colegiados y la Jefatura de Estudios.

Los Órganos Colegiados son una serie de juntas para decidir ciertos aspectos relativos a la formación académica. Estos órganos son:

- Junta de Profesores: de carácter asesor y consultivo al Comandante Director, quien la preside. Está integrada por todos los profesores del centro y su objetivo es tratar los resultados académicos. Se suele reunir a comienzo y final de curso.
- Junta Docente: también de carácter asesor y consultivo del Comandante Director, su presidente es el Subdirector Jefe de Estudios. Sus participantes son el secretario, quien es el Secretario de Estudios y los vocales, los cuales son los directores de los departamentos.
  Cuando los temas a tratar involucren al Centro Universitario de la Defensa compondrán también la junta el Subdirector o Secretaria del CUD como vicepresidente de la junta y como vocales los directores de las áreas departamentales del CUD que determine el presidente de la junta docente.
  Las propuestas de esta junta se elevan al Comandante Director para su resolución.
- Junta de Coordinación ENM-CUD: su objetivo es armonizar la actividad docente desarrollada por la ENM y el CUD. El presidente de la junta es el Comandante Director y los demás integrantes son el Secretario de Estudios como secretario y como vocales el Director del CUD, el Subdirector Jefe de Estudios, el Subdirector del CUD, el Secretario del CUD y el Jefe del Departamento de Instrucción y Adiestramiento. Esta junta se reúne semanalmente.

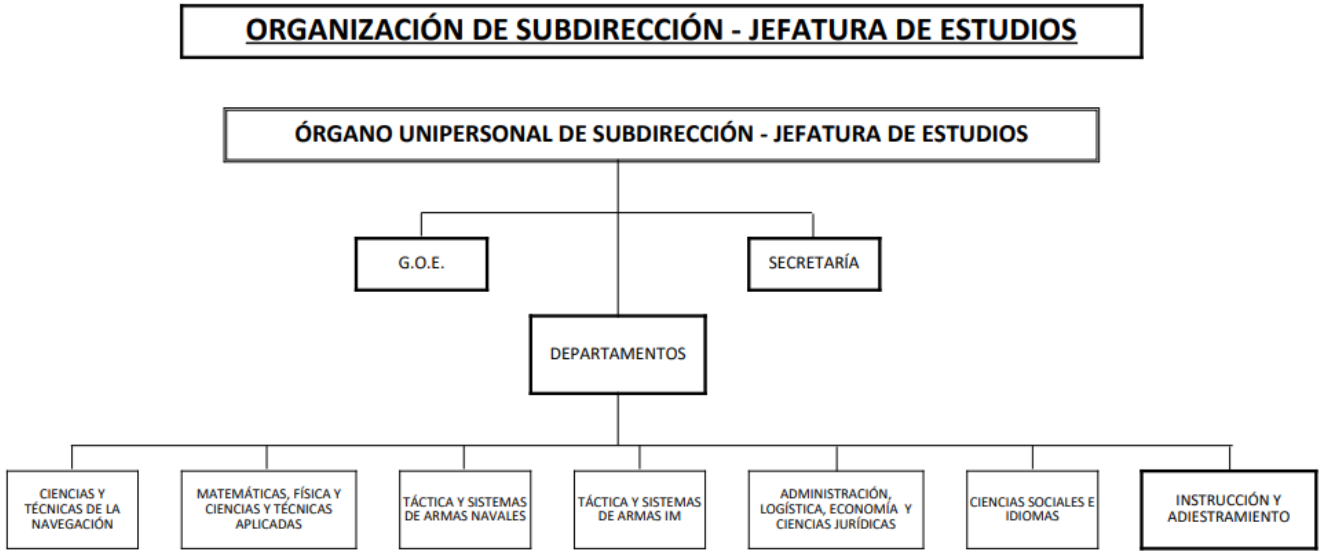

Ilustración 3-2 Organización de Jefatura de Estudios ([9])

La función principal de Jefatura de Estudios es el control de los planes de estudios y actividades de la ENM relacionadas con la formación de los alumnos. Quien dirige Jefatura de Estudios es el Subdirector Jefe de Estudios de la Escuela Naval Militar, apoyado por la Secretaría de Jefatura de Estudios que dirige el Secretario de Estudios, el Gabinete de Orientación Educativa y por los diferentes departamentos. En cada departamento se agrupan las asignaturas no dependientes del Grado de Ingeniería Mecánica del CUD, que a lo largo de este trabajo se van a denominar asignaturas de Jefatura de Estudios y las del CUD asignaturas del Grado.

3.1.1 *Centro Universitario de la Defensa*

El Centro Universitario de la Defensa tiene su propia organización ya que al estar adscrito a la Universidad de Vigo tiene otras necesidades.

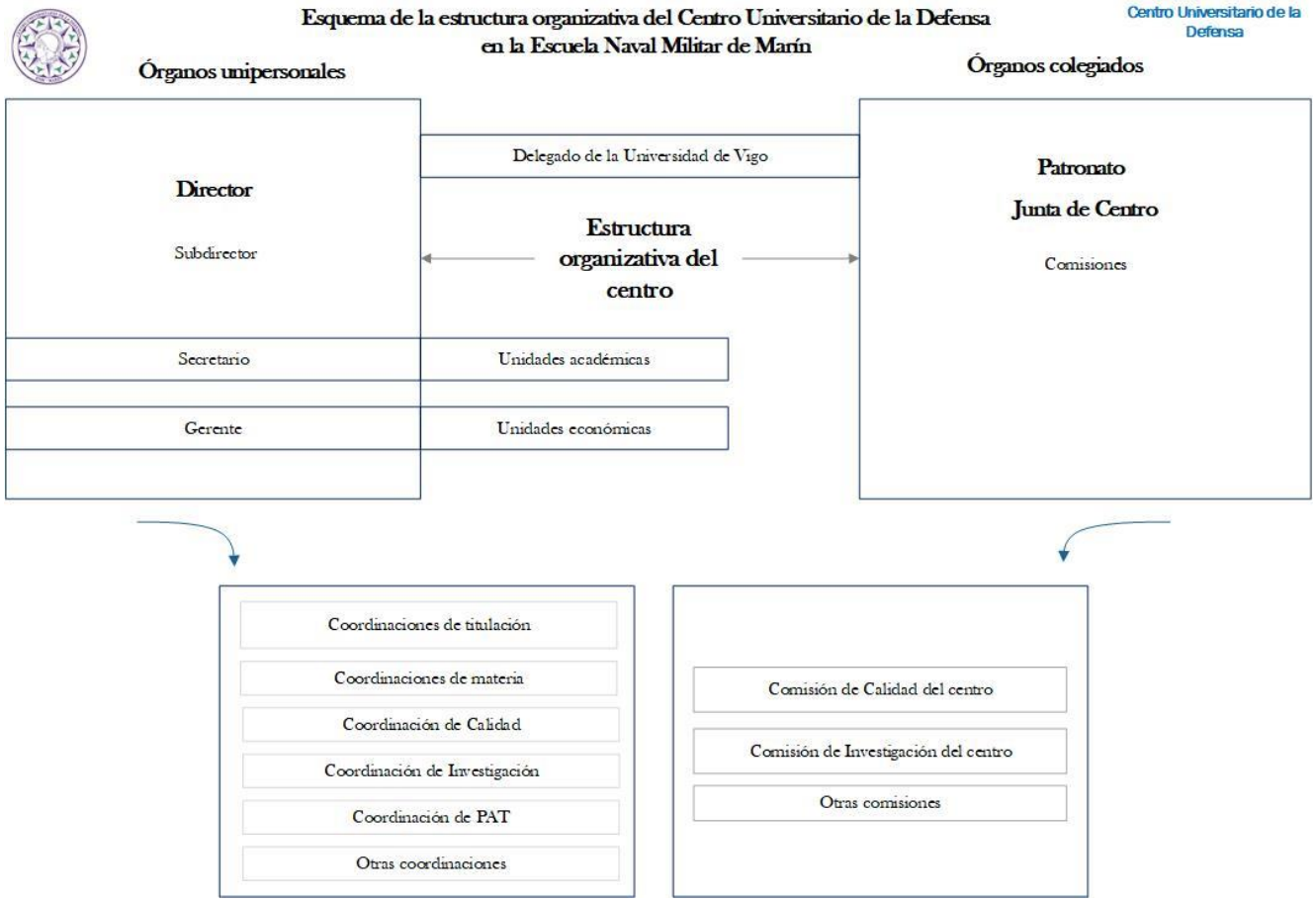

Ilustración 3-3 Estructura del CUD ([10])

Para la toma de decisiones el CUD tiene un órgano colegiado, la Junta de Centro, presidido por el Director del CUD. Además sus vocales son:

- Subdirector del CUD
- Subdirector Jefe de Estudios.
- El delegado de la Universidad de Vigo en el CUD.
- Todos los docentes del CUD.
- Una representación del personal de administración y servicios (PAS).
- Tres representantes de estudiantes.

Aparte el secretario de la junta es la Secretaria del CUD.

3.1.2 *Marco de coordinación entre la ENM y el CUD*

Dada la naturaleza del actual currículum para ser oficial de la Armada Española, que comprende un Grado de Ingeniería Mecánica por la Universidad de Vigo, coexisten en la escuela dos ramas del actual plan de estudios, cada una con sus características y forma de organizarse. Por una parte, están los créditos asociados a las materias del grado, y por otro, los créditos asociados a Jefatura de Estudios. Debido a esto, la coordinación entre estas dos vertientes se hace mediante la Junta de Coordinación CUD-ENM presidida por el Comandante Director, por la participación del Subdirector Jefe de Estudios en las Juntas de Centro del CUD, por la participación del Subdirector o Secretaria del CUD en las Juntas Docentes de

la ENM cuando es necesario y por la cooperación entre el Subdirector del CUD y el Secretario de Estudios.

3.2 Situación actual de la planificación académica

La planificación del curso académico comienza a mediados del anterior, aproximadamente después de la segunda semana de I+A. En su desarrollo se crean una serie de documentos que indica un calendario con los días disponibles, las horas que necesita recibir cada alumno semanalmente de acuerdo con su plan de estudios además de las horas que deben impartir los docentes para suplir las necesidades académicas de los alumnos.

En primer lugar, es necesario saber qué días hay disponibles en el curso, cuando son las semanas de Instrucción y Adiestramiento (I+A), festivos, períodos vacacionales, embarques de fin de curso, etc. Por ello la junta de coordinación ENM-CUD elabora una propuesta del Calendario de Actividades donde se recogen todos los días lectivos, festivos, de permiso, de exámenes y de I+A del curso. Esta propuesta se envía a DIGEREM (Dirección General de Reclutamiento y Enseñanza Militar) que es quien la aprueba.

Teniendo el Calendario de Actividades, Jefatura de Estudios elabora la Planificación Escolar (publicación 003 de la ENM), donde se recogen todas las vicisitudes e hitos del curso escolar junto con el Calendario de Actividades y un desglose por semanas de las horas a impartir por cada asignatura del currículum. Las horas de las asignaturas del Grado las proporciona el Subdirector del CUD.

Respecto a las asignaturas de Jefatura éstas se aglutinan por departamentos, cuyo jefe es el coordinador de las Guías Docentes de su ámbito, las cuales envía al Secretario de Estudios, más el encargado de redactarlas es el coordinador de materia.

La llamada sábana es el documento donde se recoge el horario de clases de toda la escuela por semana, la elaboración de este horario recae sobre el Subdirector del CUD y el Secretario de Estudios, los cuales tiene que casar las necesidades de horario de las asignaturas del Grado con las de Jefatura. Debido a la mayor flexibilidad de las asignaturas de Jefatura primero se planifican las horas del Grado y en base a eso se establecen las asignaturas de Jefatura. A principio de curso se hacen dos semanas, A y B, las cuales se alternan a lo largo de los meses sufriendo eventualmente las modificaciones necesarias según el momento. De hacerse algún cambio sobre estas plantillas, es el coordinador de sábana el encargado.

Si algún docente necesita un cambio se lo eleva a su coordinador de materia, el cual se lo solicita al coordinador de sábana vía correo, llamada o se acerca y se lo dice directamente. Debido a este sistema de planeamiento y de hacer cambios, junto con la enorme casuística que se da en la escuela naval, suele haber problemas del tipo: dos clases en el mismo aula, un alumno que tiene varias clases al mismo tiempo, etc.

En cuanto a los exámenes cabe diferenciar entre parciales y finales. Los parciales de las asignaturas de Jefatura se hacen en los periodos de dicha materia, por lo que no hay ningún problema organizativo salvo el que haya clases disponibles en la sábana, por lo que son los coordinadores de materia quienes les ponen fecha y lo notifican a su director de departamento. En los finales el encargado de su calendario

es el Secretario de Estudios y los sitúa en función al calendario de exámenes ordinarios y extraordinarios del Grado.

3.2.1 *La gestión académica del Centro Universitario de la Defensa*

En el CUD partiendo de la Planificación Escolar del curso que viene, o en su ausencia una estimación de esta, el subdirector del CUD elabora la Planificación Docente de Alumnos (PDA), la cual refleja en función a los créditos que cursa el alumno y los días disponibles las horas que necesita recibir para su formación. Tras esto se calcula en función al número de grupos de clase las horas que tiene que impartir el profesorado, para que así los alumnos reciban las horas de formación acordes a las asignaturas que cursan. Esta planificación se envía en abril del curso anterior a la Universidad de Vigo para informarles.

Una vez generada la PDA hay que traducirla en los Planes de Organización Docente (POD) de cada asignatura, donde se recoge qué docente da cuántas horas de teoría, laboratorio o seminario de cada asignatura. La redacción de estos documentos recae en el coordinador del POD, que actualmente es la misma persona que el subdirector del CUD. Su redacción comienza con un primer borrador que elabora el coordinador de POD donde se especifica en general los POD de cada asignatura. Este primer borrador se lo envía a los coordinadores de cada materia los cuales tienen cierto margen de maniobra para modificar el POD de sus materias para ajustarlos a las particularidades de los docentes disponibles. Tras las aportaciones de los coordinadores de materia es el coordinador de POD quien tiene la última palabra y aprueba el documento, una vez aprobado es enviado a la Uvigo al igual que la PDA. Los coordinadores de materia podrán solicitar cambios al coordinador de POD a lo largo del curso en función de la evolución de éste.

Las clases programadas en la sábana se deberían corresponder con las que establece el POD que son necesarias. Por ello a lo largo del curso se hace un seguimiento de cuántas clases se están impartiendo para comprobar el cumplimiento del POD establecido. Este seguimiento lo lleva el coordinador de materia, el cual es el responsable de su cumplimiento y rinde cuentas semanalmente ante al coordinador de POD. Además de este seguimiento semanal hay dos reuniones por cuatrimestre para tratar este tema.

Por último, está la gestión de los exámenes, pero hay que diferenciar entre parciales y finales/ ordinarios/ extraordinarios. Los primeros los establece el subdirector del CUD junto con los coordinadores de materia a principio del curso, se reparten por semanas y en cada semana se deja un hueco en la sábana para los parciales. Pero esta distribución es flexible y se pueden hacer cambios. Sin embargo, los ordinarios/extraordinarios los establece la Junta de Centro, y establece un día para cada examen de materia al comienzo del curso y sólo se harían cambios en otra Junta de Centro, esto se plasma en el calendario de exámenes ordinarios y extraordinarios.

3.3 Necesidades generales

Desde la perspectiva de alumno, a priori, sería de utilidad en la gestión académica de la ENM una aplicación sencilla que proporcione un flujo de información rápido y constantemente actualizado con el centro y con los docentes de sus materias. Con independencia de como casar todos los planes de estudios en un horario limitado en tiempo y recursos, a veces hay interferencias entre materias. Un ejemplo ocurre cuando coinciden dos exámenes el mismo día, dos clases en el mismo aula, un profesor en dos sitios a la vez, un examen tras la semana de I+A o un fin de semana con pernocta de lanchas o continuada de IM. Algunas de estas casuísticas relacionadas con las fechas de los exámenes son inevitables, pero el

que todo docente pueda consultar fácilmente el horario de sus alumnos y coordinar en tiempo real con los demás docentes puede suponer cierta mejoría.

Tras las reuniones con el Subdirector del CUD y el Secretario de Estudios de la ENM, se llegó a las siguientes conclusiones:

- Es necesario automatizar el sistema, mediante una aplicación donde salten alarmas al programar un evento incompatible con otro en la sábana o bien que ese aula no estuviese disponible evitaría en gran medida ciertas incompatibilidades que luego suceden.
- La aplicación debe ser un medio de comunicación eficaz, robusto y rápido que permita gestionar correctamente los problemas que surjan, de forma que todos los usuarios implicados reciban la información adecuada. Finalmente, también sería de ayuda para llevar un seguimiento automatizado de las horas de clase impartidas o la asistencia de los alumnos, en donde se generen alarmas automáticas que reflejen ciertas situaciones que se consideren importantes

Por último, tras varios correos con el Teniente de Navío Franco Moreu se descubrieron aspectos de serían de utilidad en la aplicación. Un ejemplo sería la posibilidad de consultar fácilmente el Calendario de Actividades a la hora de programar actividades por parte del profesor o también en el ámbito de las horas de Orden Cerrado o de Adiestramiento poder reflejar en la sábana qué instalaciones se van a usar para evitar, por ejemplo, que dos brigadas fuesen a la vez a la pista militar.

3.4 Documentación del problema

Para una mejor comprensión del sistema modelado hay que explicar las clases que intervienen en él, debido a que UML es un lenguaje de modelado orientado a los objetos. Por esto no tendría sentido comenzar por los procesos y actividades sin definir antes qué elementos están involucrados, así que a continuación se detallan los diagramas de clase.

3.4.1 *Diagramas de clase*

Dentro de este punto del trabajo se muestran todos los elementos del sistema, incluyendo actores, eventos, solicitudes, notificaciones, alarmas de evento y las relaciones que tienen entre sí.

3.4.1.1.1 Diagrama de clases general

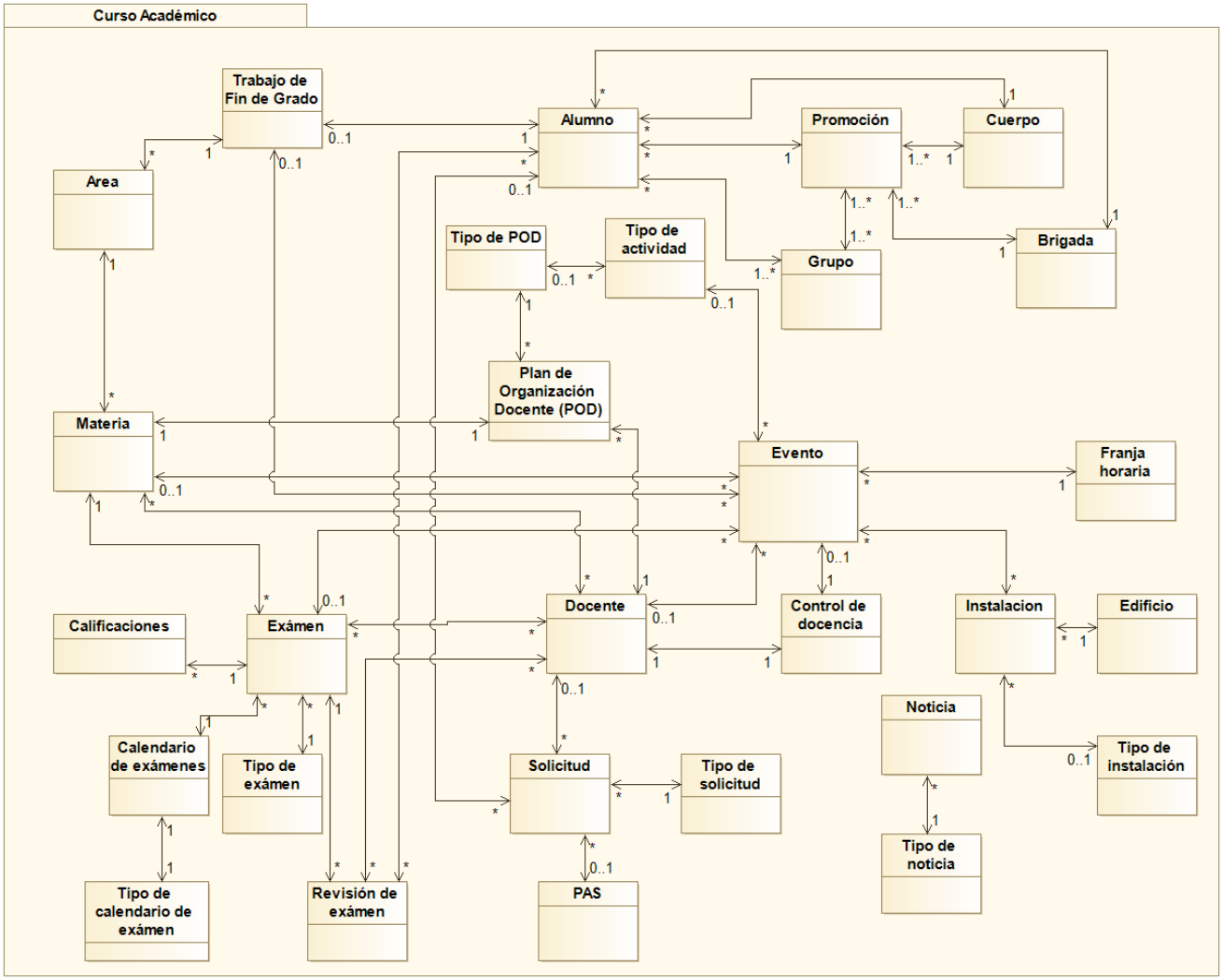

Ilustración 3-4 Diagrama de Clases General (elaboración propia)

Se muestran los elementos más generales del sistema, con relaciones de asociación entre ellos. Cabe destacar que se ha englobado todo dentro del paquete Curso Académico, ya que todos estos elementos tienen su sentido en el contexto de un curso académico, que más adelante se detallará. Esto no implica que el sistema deba replicar las bases de datos de, por ejemplo, profesores, en cada curso académico.

Por el contrario se trata de una forma de indicar que las relaciones entre las diferentes clases que componen la estructura del sistema deben tener en cuenta el marco de trabajo de cada curso académico, al que se asocia el PDA, POD y calendario académico, entre otras aplicaciones principales.

3.4.1.1.2 Diagrama de clases de actor

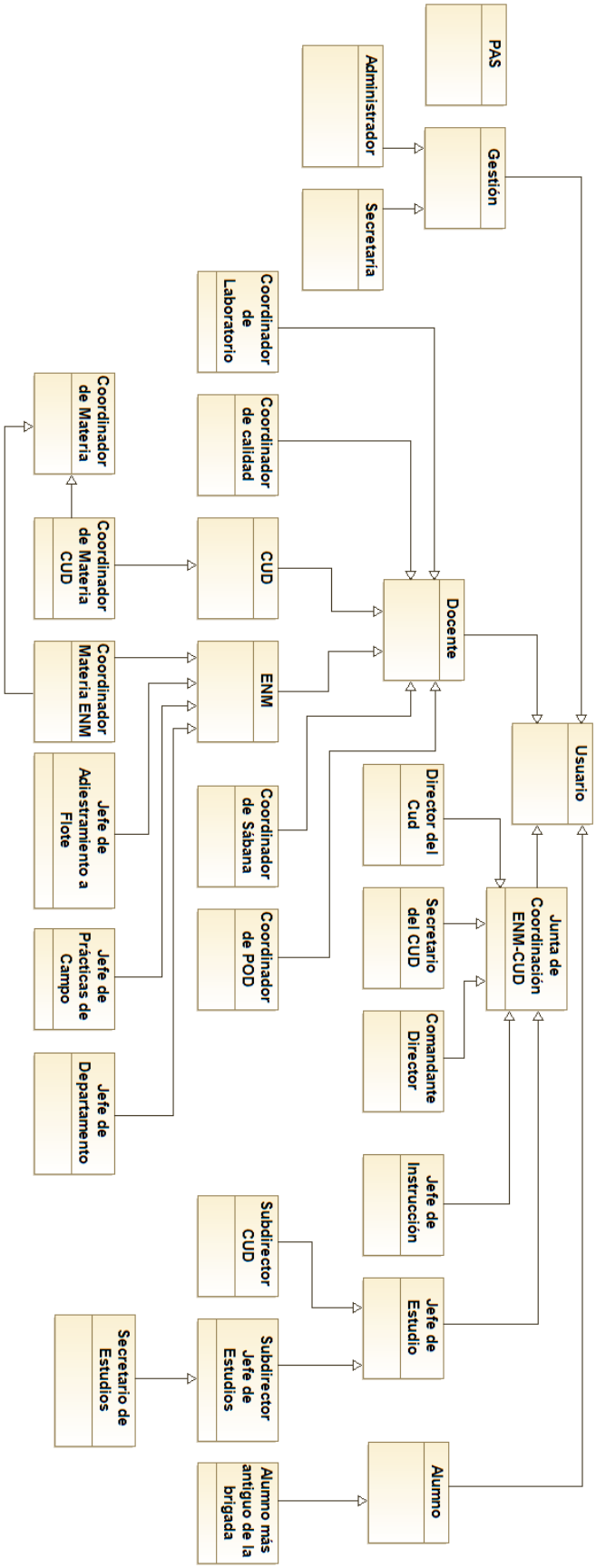

Ilustración 3-5 Diagrama de clases de actor (elaboración propia)

Lo más importante respecto la Ilustración 3-5 es que no es un organigrama que represente jerarquía, y tampoco representa necesariamente a personas particulares. Representa diferentes relaciones entre perfiles de usuario que se han considerado útiles a partir del estudio de la actual organización de la ENM y de posibles mejoras. De esta forma, aunque actualmente los roles de Subdirector del CUD y, lo que se ha tipificado como Coordinador de POD, los desempeña la misma persona, la situación podría cambiarse en el futuro. De esta forma, el sistema de representación debe permitir manejar la situación en la que un mismo usuario tiene activos ambos perfiles de forma simultánea, o la situación en la que estas tareas las desempeñan usuarios diferentes. La atomización de la representación de un sistema de información en los elementos más sencillos posibles (véase el hacer tantos actores como funciones hay en el sistema), permite añadir robustez a la representación de la aplicación ante posibles necesidades de implementar cambios en el futuro.

Por último, cabe recalcar las relaciones de herencia existentes entre los actores, de una clase padre respecto las clases hijas. Los atributos y funciones del actor Usuario los van a tener todos los actores que hereden de él, pero el actor Alumno no va a tener las mismas funciones que un Docente, por ejemplo.

3.4.1.1.3 Diagrama de clases de evento

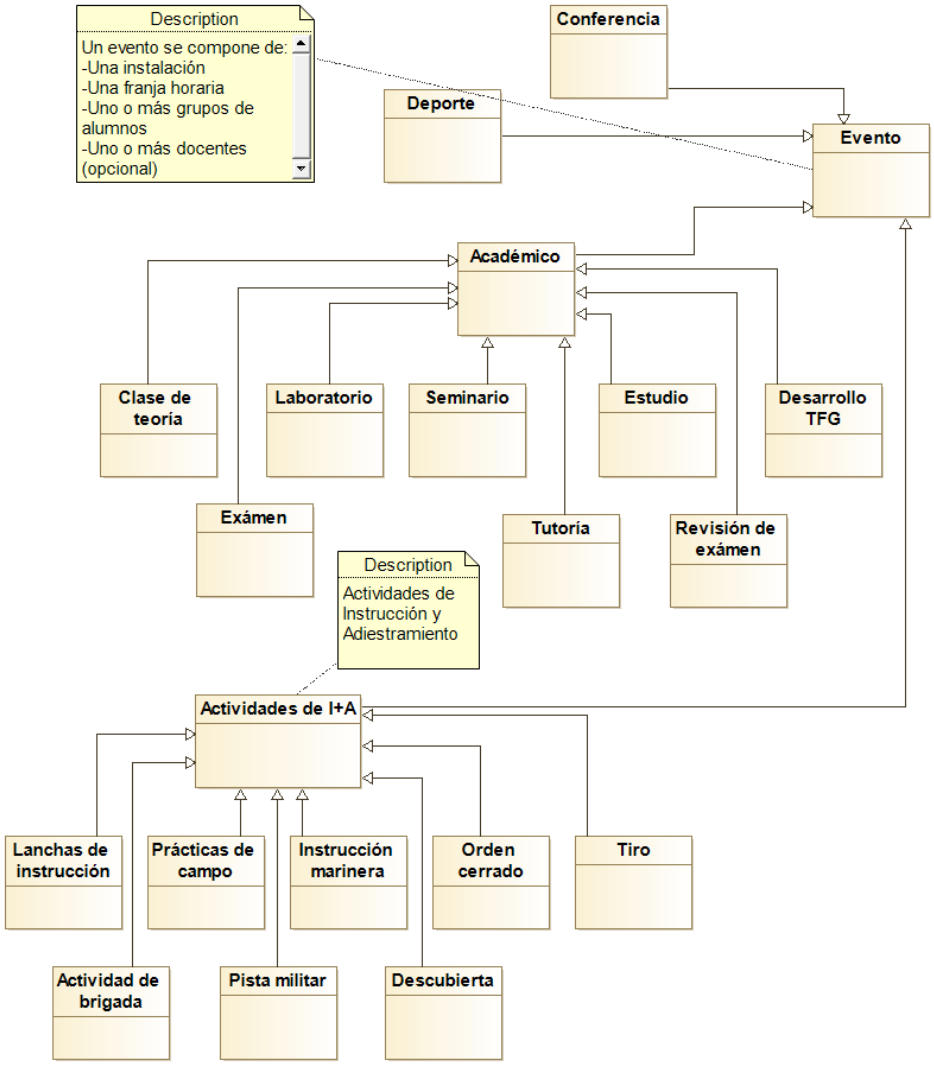

Ilustración 3-6 Diagrama de clases de evento (elaboración propia)

En la Ilustración 3-6 se muestran los tipos de eventos que hay actualmente en la escuela, entre los que cabe destacar que un evento es la unidad mínima que compone la llamada sábana, en adelante Calendario de Eventos. Es decir, un evento se asocia a un recurso que ocupa una única franja horaria, involucra a una instalación, un grupo o grupos de alumnos y puede tener desde ninguno, uno o incluso varios docentes asociados (por ejemplo, horas de estudio, clase de teoría, examen ordinario). De esta forma, una clase de teoría de dos horas sería representada por dos eventos de una clase de una hora en esta propuesta de representación. Así, podría darse el caso de que se va la luz o el profesor tiene que irse por un imprevisto a mitad de las dos horas de teoría, por lo que dividiendo en dos eventos quedaría constancia de que se ha impartido una hora de teoría pero que falta otra por dar en la aplicación.

3.4.1.1.4 Diagrama de clase de solicitud

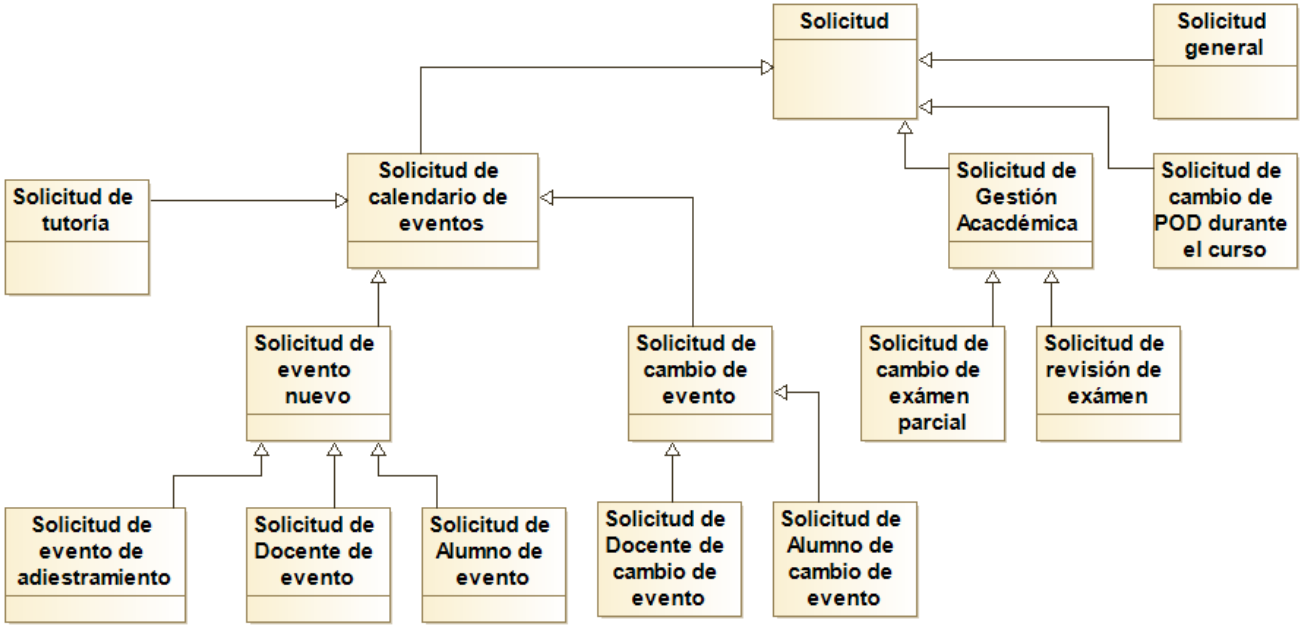

Ilustración 3-7 Diagrama de clases de solicitud (elaboración propia)

En la Ilustración 3-7 se detalla qué tipos de solicitudes serían de utilidad tras el análisis de qué funciones se necesitan en la aplicación.

3.4.1.1.5 Diagrama de clase de notificación

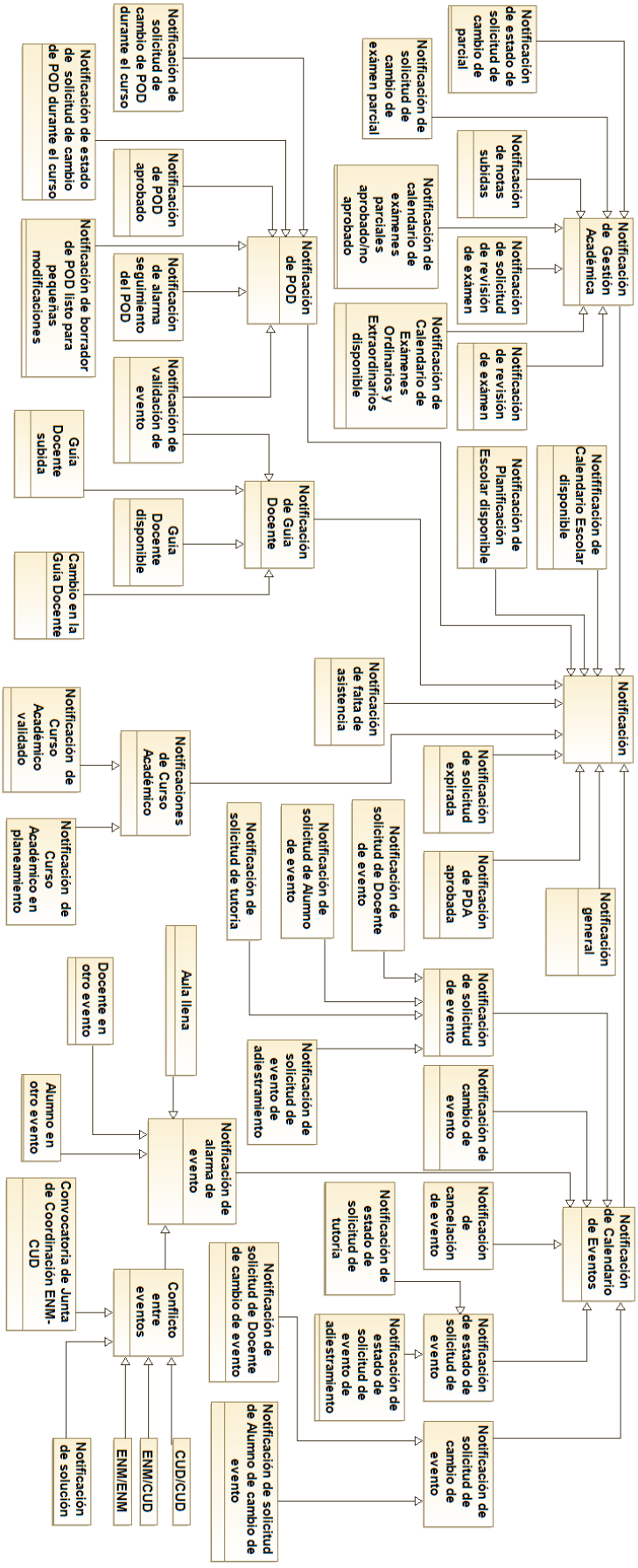

Ilustración 3-8 Diagrama de clases de notificación (elaboración propia)

3.4.1.2 *Diagrama de clases de alarma de evento*

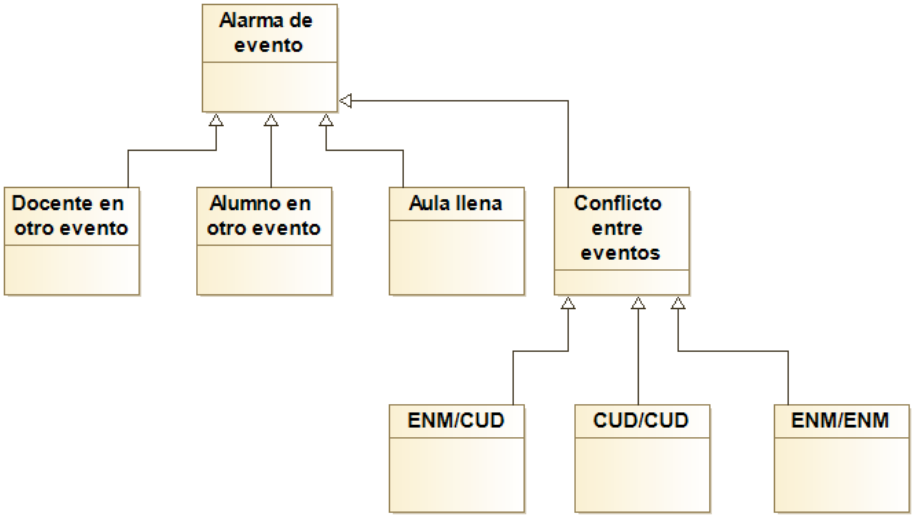

Ilustración 3-9 Diagrama de clases de alarma de evento (elaboración propia)

Cada alarma de evento se corresponde con cada posible conflicto que puede tener un evento a la hora de ser programado.

- Primero: que el docente ya esté ocupado en otro evento en ese instante
- Segundo: que haya alumnos en la misma situación,
- Tercero que no haya sitio en el aula para todos los alumnos.
- Cuarto: se contempla que un evento sea incompatible con otro, sean de naturaleza de las asignaturas del Grado o de Jefatura.

3.4.2 *Diagramas de casos de uso*

En estos diagramas se va a detallar desde un punto de vista general hasta uno más específico las interacciones que tienen los actores con el sistema. Los principales son:

- Casos de uso de Curso Académico.
- Casos de uso de Calendario de Eventos.
- Casos de uso de Planificación Escolar.
- Casos de uso de Planificación Docente de Alumnos.
- Casos de uso de Plan de Organización Docente.
- Casos de uso de Guía Docente.
- Casos de uso de Gestión Académica.
- Casos de uso de Control de Asistencia.
- Casos de uso de Gestión Administrativa de los Datos.
- Casos de uso de Gestión de Usuarios.

Además de estos casos de uso más generales hay subcasos de uso relacionados en algunos apartados.

3.4.2.1 *Casos de uso de Curso Académico*

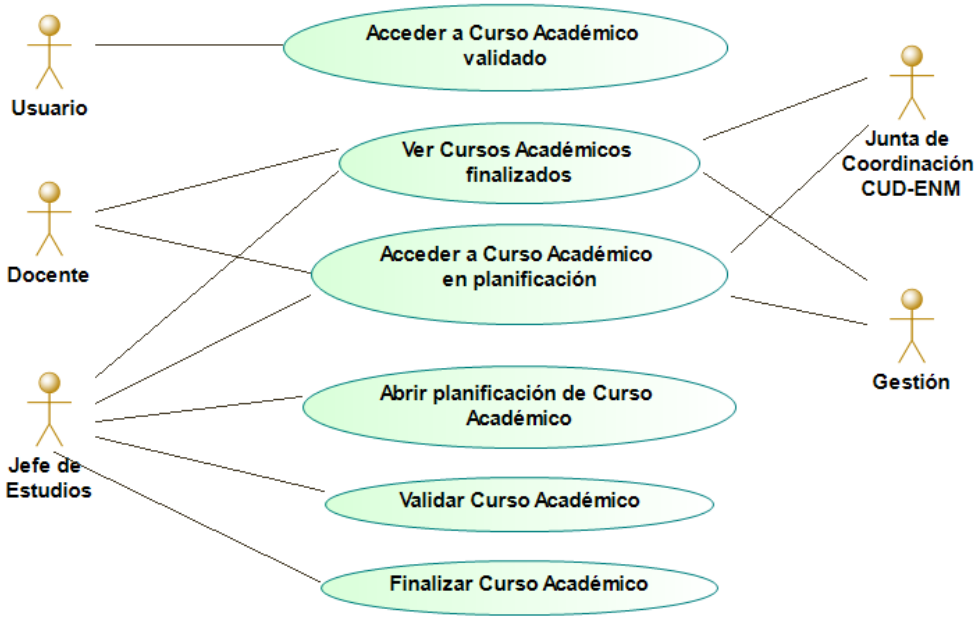

Ilustración 3-10 Diagrama de casos de uso de Curso Académico (elaboración propia)

Debido a que el siguiente Curso Académico comienza a planearse a mediados del que está en curso, se propone el compartimentar por curso académico todos los datos como se muestra en el Diagrama de clases general, además que de un año para otro los cambian. El Curso Académico se plantea de tres formas, validado que es el Curso Académico en curso, el cual todos los Usuarios necesitan ver. También se consideraría útil el poder consultar por el resto de Usuarios, salvo Alumnos, los Cursos Académicos ya finalizados como ayuda y referencia para planificar el siguiente. La potestad de validar, finalizar o abrir la planificación del Curso Académico quedaría bajo los actores de la rama de Jefe de Estudios inicialmente.

3.4.2.2 *Casos de uso del Calendario de Eventos*

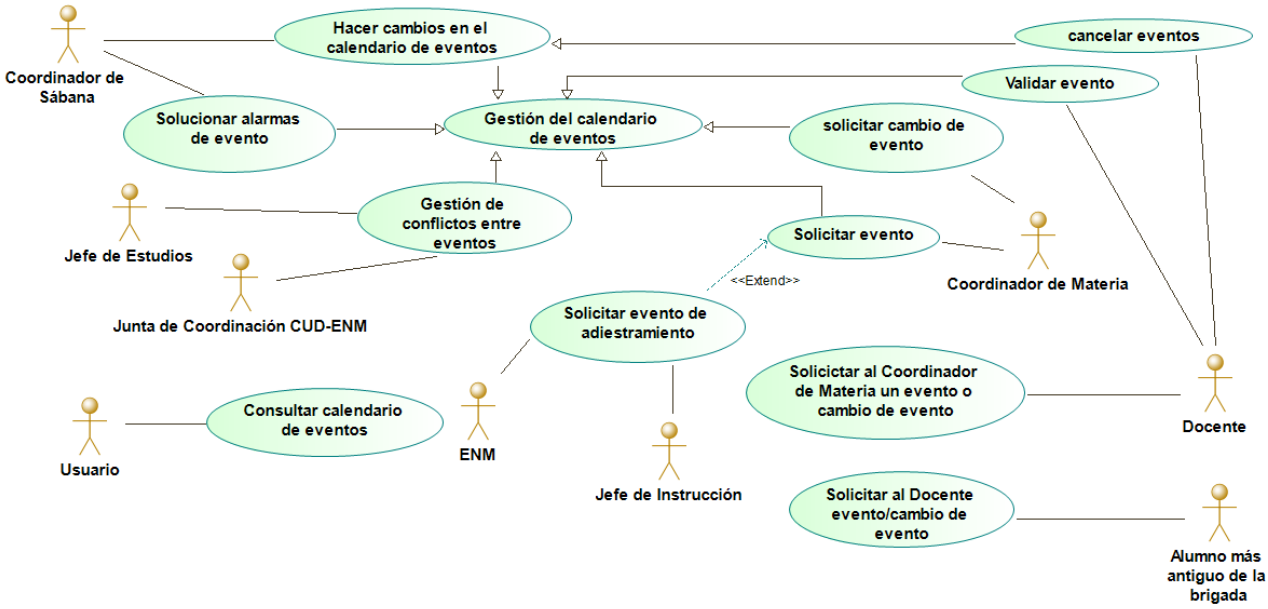

Ilustración 3-11 Diagrama de casos de uso de Calendario de Eventos (elaboración propia)

El Calendario de Eventos representa lo que en la escuela se llama sábana, donde se refleja por cursos y grupos de clase las horas de clase de la semana. El encargado de elaborarla y hacer modificaciones sería el Coordinador de Sábana. La propuesta para gestionar las solicitudes consiste en que los Docentes pueden cancelar directamente un evento, pero es el Coordinador de Materia el que centraliza y hace las solicitudes de evento o cambio por materia. Para así evitar que lleguen al Coordinador de Sábana peticiones contradictorias de una misma asignatura. En caso de que un Docente quiera hacer una solicitud, se lo pediría a su Coordinador de Materia, a su vez se contemplaría que el alumno más antiguo de cada brigada pueda hacer solicitudes de este tipo al Docente.

Los usos de evento de adiestramiento, solucionar alarmas de evento y la gestión de conflictos se detallan en sus respectivos subcasos de uso que vienen a continuación.

3.4.2.2.1 Subcaso de uso solicitud/cambio de evento

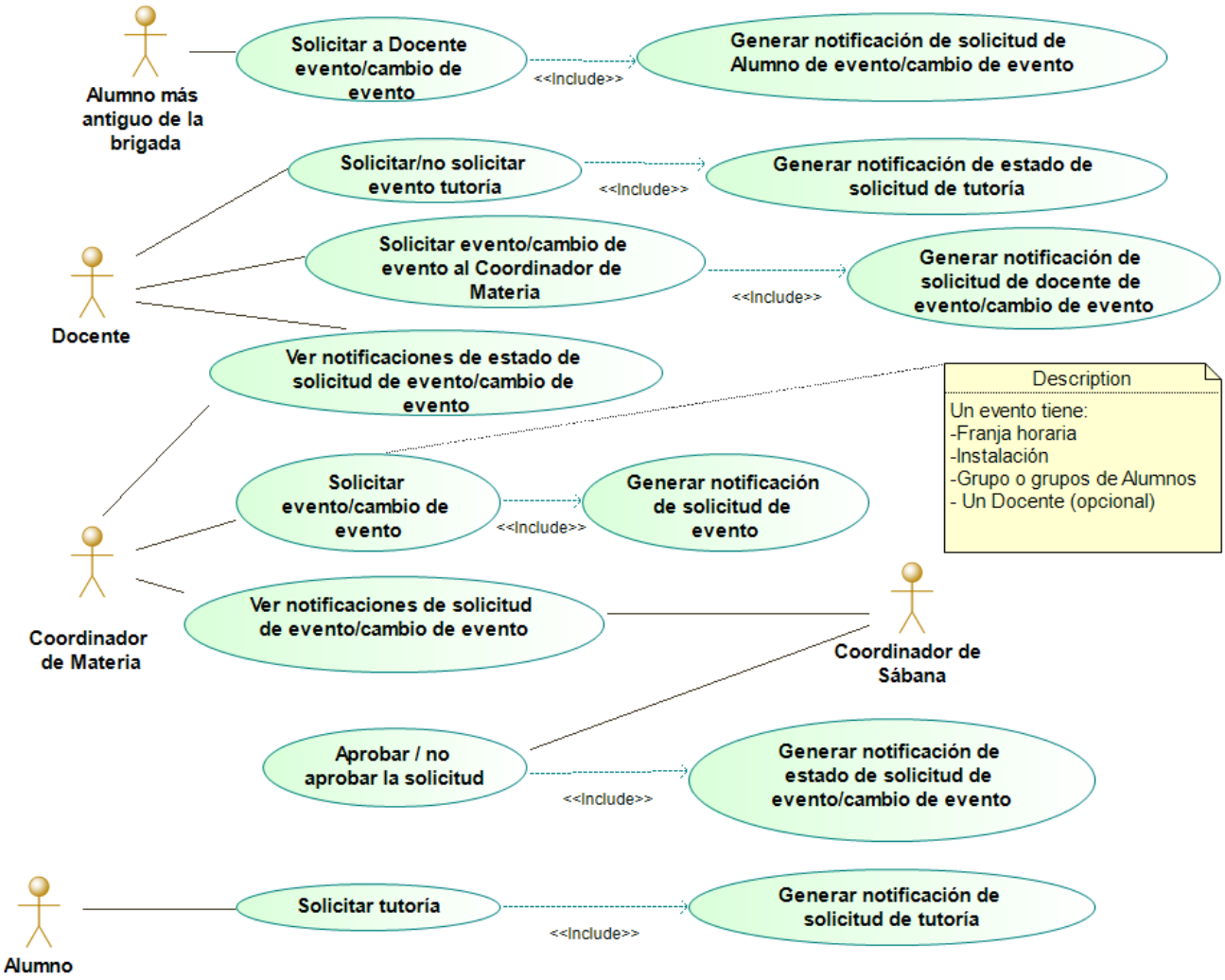

Ilustración 3-12 Subcaso de uso de solicitud/cambio de evento (elaboración propia)

La Ilustración 3-12 Subcaso de uso de solicitud/cambio de evento (elaboración propia) muestra con mayor detalle la solicitud de eventos o de cambio de los mismos.

3.4.2.2.2 Subcaso de uso solicitud de evento de adiestramiento

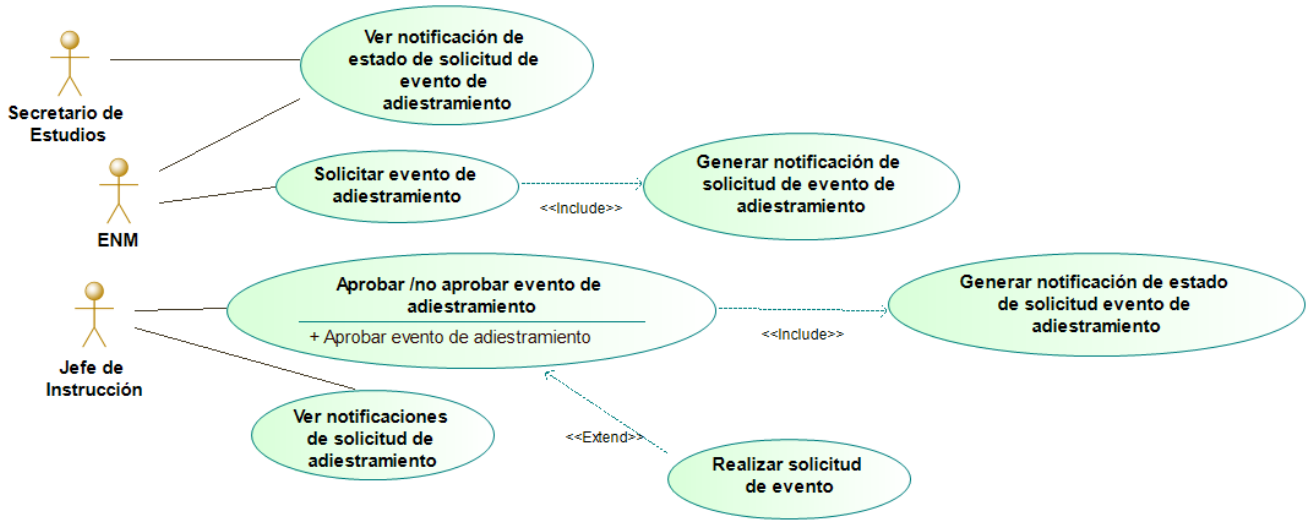

Ilustración 3-13 Subcaso de uso de solicitud de evento de adiestramiento (elaboración propia)

Tras solicitar información a varios Docentes militares sobre cómo se gestionan las actividades de instrucción como salidas de lanchas, instrucción marinera o salidas de campo surgió esta propuesta sobre como podría ayudar una aplicación de este tipo a su gestión. Resumiendo todo evento de I+A debe ser aprobado por el Jefe de Instrucción, tras lo que se informa al Secretario de Estudios para coordinarlo con el resto de actividades, tras esto el evento de I+A pasaría a ser tratado como un evento más del Calendario de Eventos, ya que en esencia se compone de un grupo de alumnos, un Docente y una instalación enmarcados en una ventana horaria.

3.4.2.2.3 Subcaso de uso de cancelación de evento

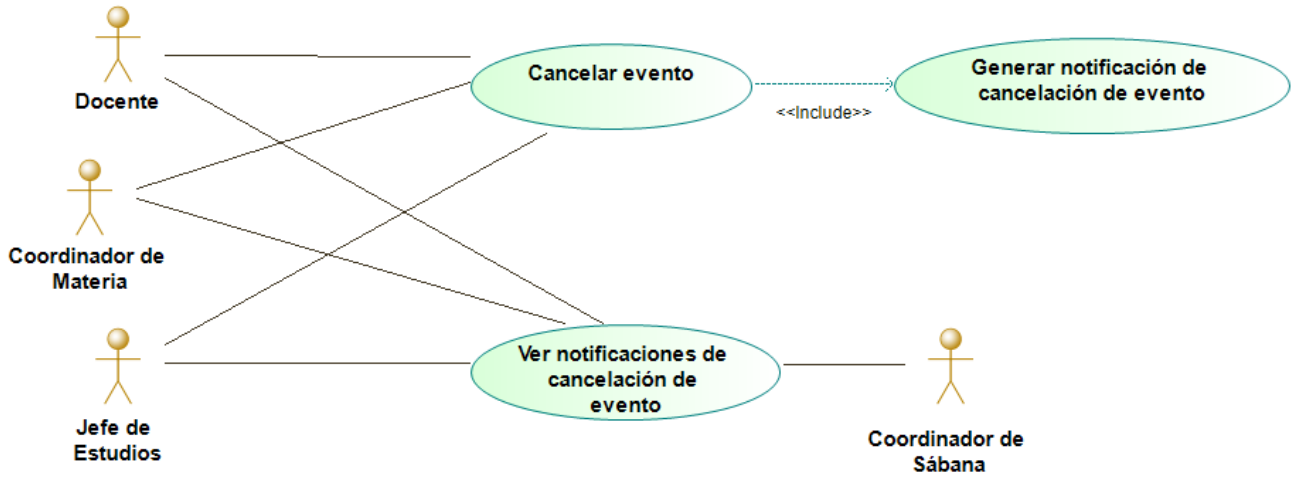

Ilustración 3-14 Subcaso de cancelación de eventos (elaboración propia)

Debido a que cualquier imprevisto podría imposibilitar impartir un evento, cualquier Docente puede cancelarlo directamente en cualquier momento, tras lo que se notificaría a los Usuarios involucrados en el evento.

3.4.2.2.4 Subcaso de uso de conflicto entre eventos

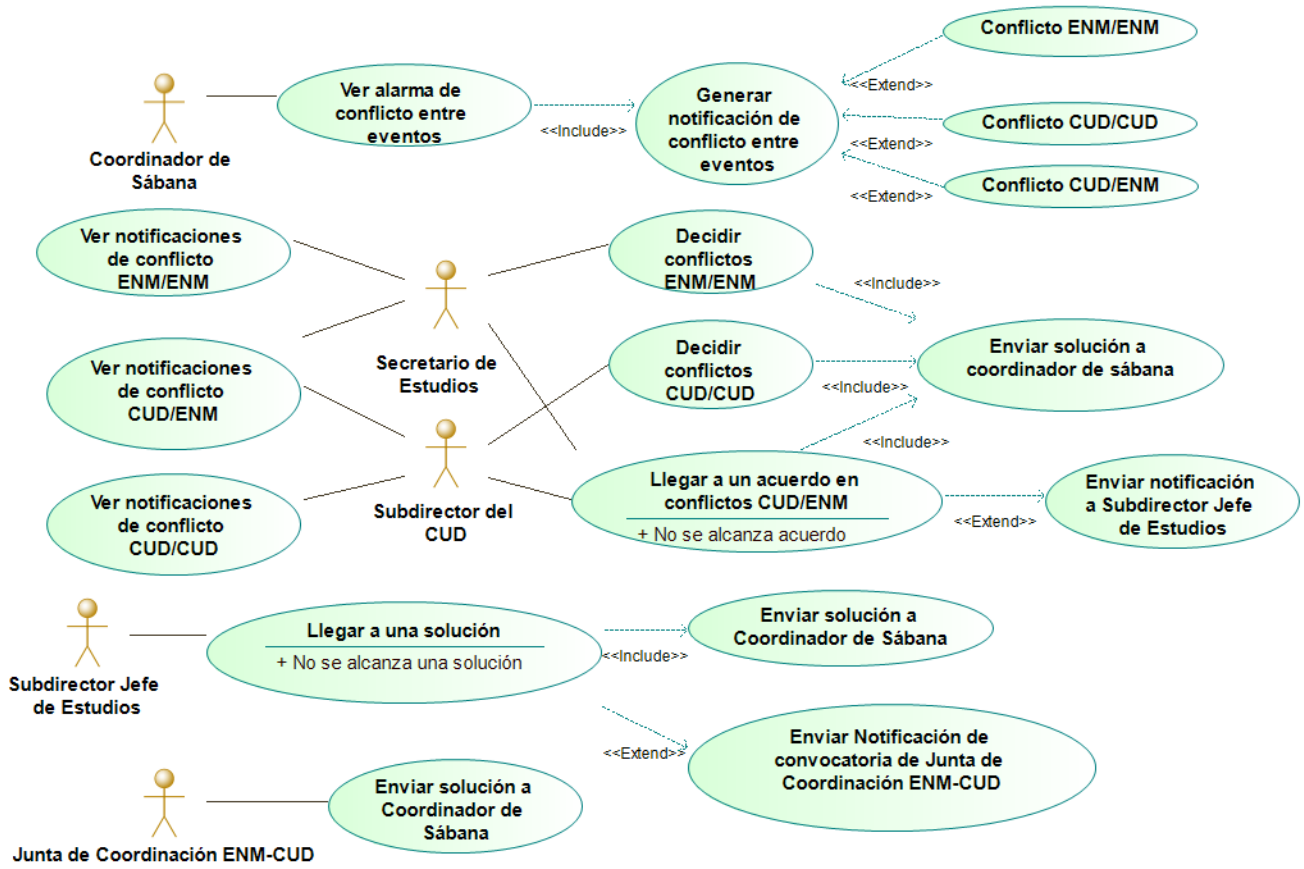

Ilustración 3-15 Subcaso de uso de conflicto entre eventos (elaboración propia)

Ya que se dispone de un tiempo y recursos limitados, una gran casuística y un mínimo de horas que impartir, no siempre hay sitio para todos los eventos en el Calendario de Eventos, por lo que se generan conflictos. En dicha situación se propone que sea el Coordinador de Sábana el encargado de detectarlos y notificarlo al Subdirector del CUD, al Secretario de Estudios o a ambos para que decidan qué solución tomar. En caso de no llegar a una solución se elevaría la decisión al Subdirector Jefe de Estudios, tras lo cual se convocaría la Junta de Coordinación ENM-CUD de ser necesario.

3.4.2.2.5 Subcaso de uso de alarma de evento

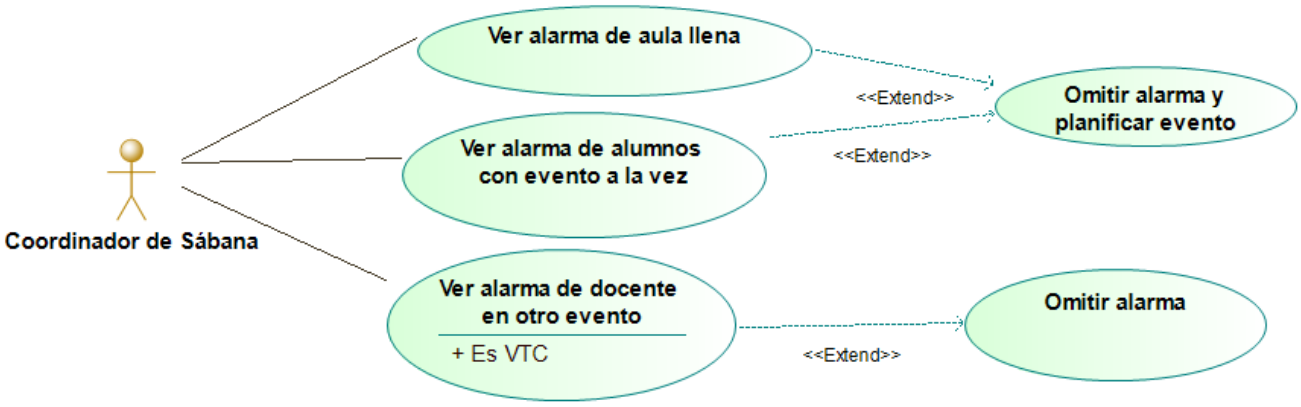

Ilustración 3-16 Subcaso de uso de alarma de evento (elaboración propia)

En relación a las alarmas, hay causas que podrían imposibilitar un evento. Pero contemplaría el poder omitirlas por causas justificadas debido a la situación. Por ejemplo, un Docente no puede estar en varias aulas a la vez, pero en este curso a causa del coronavirus los Docentes han llegado a dar las clases por VTC, mientras los alumnos las recibían en sus aulas habituales, lo cual permitía tener varios eventos en aulas distintas a la vez empelando al mismo Docente. O relacionado con Alumnos que tienen curso de asignaturas pendientes de otros años, a veces les es imposible cuadrar en el horario todas las clases. Por lo que en algunos casos un seminario u hora de teoría no asistían debido a tener dicho curso, por lo cual en ese caso no quedaría más opción que omitir la alarma de Alumno con otro evento a la vez.

3.4.2.3 *Casos de uso de Planificación Escolar*

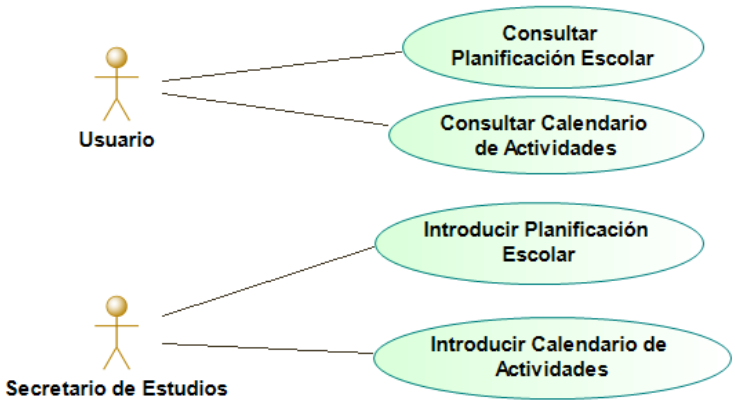

Ilustración 3-17 Diagrama de casos de uso de Planificación Escolar

Como se trató en Necesidades generales sería de ayuda para los Docentes para planificarse y para el resto de Usuarios el poder consultar el Calendario de Actividades y la Planificación Escolar, por lo que se plantea que fuese el Secretario de Estudios quien los subiese a la aplicación para poder consultarse, ya que la Planificación Escolar la hace Jefatura de Estudios.

3.4.2.4 *Casos de uso de Planificación Docente de Alumnos*

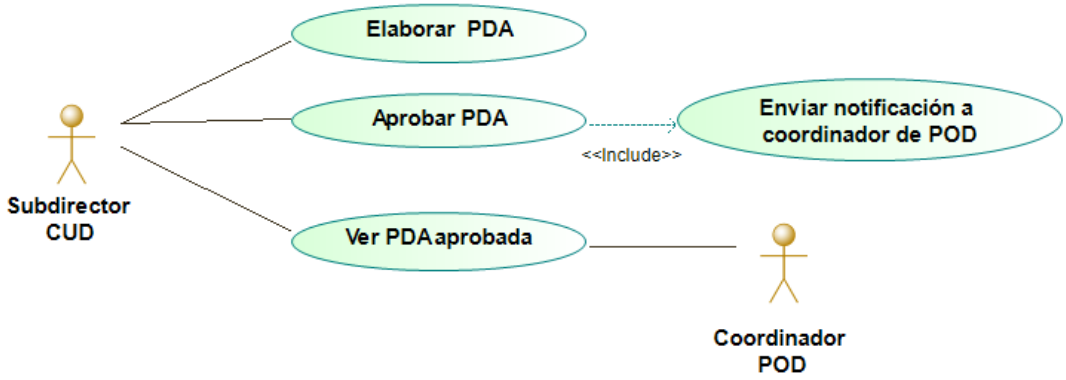

Ilustración 3-18 Diagrama de casos de uso de la PDA (elaboración propia)

Debido a que la PDA a efectos del CUD se usa como punto de partida para elaborar el POD, sería bueno que la aplicación sirviese para elaborarla y posteriormente que el Coordinador de POD la pudiese consultar para hacer el POD.

3.4.2.5 *Casos de uso del Plan de Organización Docente*

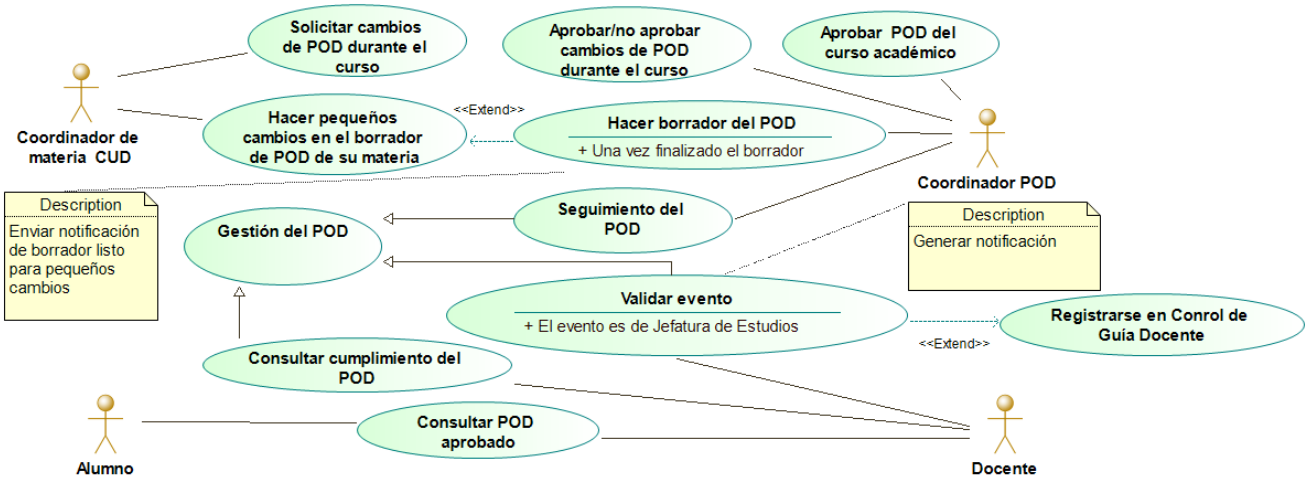

Ilustración 3-19 Diagrama de casos de uso de Plan de Organización Docente (elaboración propia)

En la Ilustración 3-19 se propone inicialmente que los encargados de establecer los Planes de Organización Docente sea el Coordinador de POD junto con pequeñas aportaciones de los Coordinadores de Materia, tras lo que finalmente el Coordinador de POD daría el visto bueno. Si hubiese que hacer alguna modificación del POD a lo largo del curso la solicitaría el Coordinador de Materia, y la aprobaría el Coordinador de POD. Por último, el validar los eventos para el seguimiento de que se cumple el plan de cada asignatura sería bueno que fuese responsabilidad de todos los Docentes. Mas el encargado del seguimiento sería el Coordinador de POD.

3.4.2.5.1 Subcaso de uso validar evento

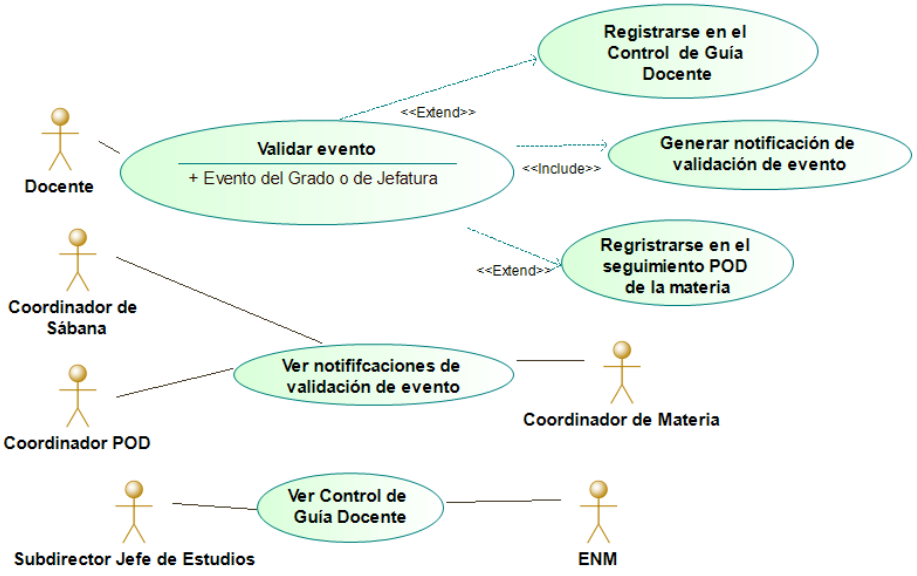

Ilustración 3-20 Subcaso de uso de validar evento (elaboración propia)

Aunque en las asignaturas de Jefatura no se hace un POD debido a su mayor flexibilidad, sería igualmente de utilidad que hubiese un seguimiento de cuantas horas y qué Docente las da debido al usual cambio o ausencia de Docentes militares debido a comisiones o al curso de ascenso. Por lo que dicho seguimiento se encuadraría en el Control de Guía Docente, que es el homólogo del Control de POD de las materias del Grado.

3.4.2.6 *Casos de uso de Guía Docente*

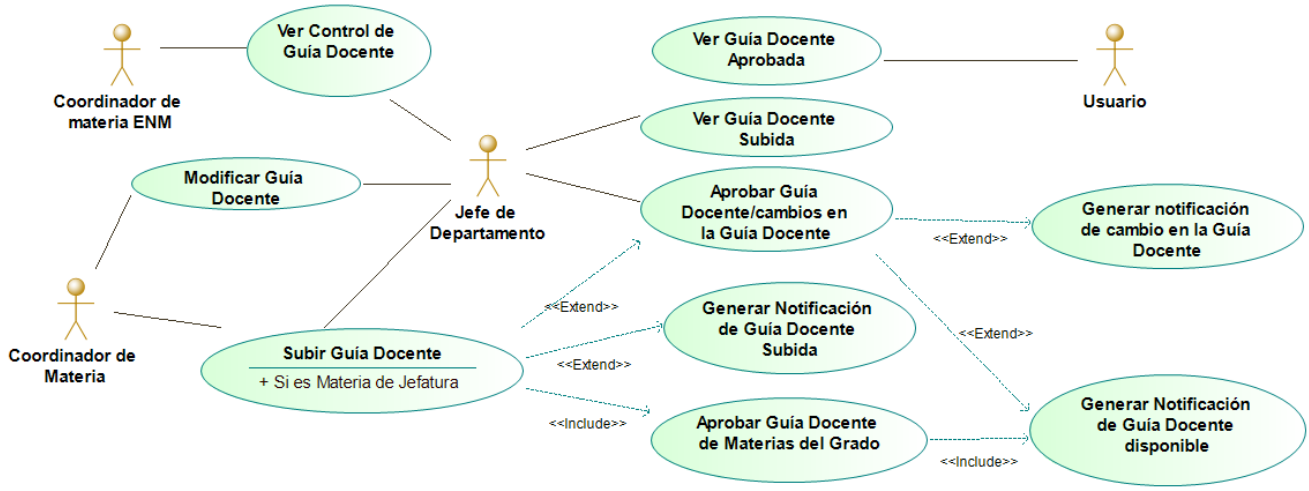

Ilustración 3-21 Diagrama de casos de uso de Guía Docente (elaboración propia)

Todas las materias del currículum para ser Oficial de la Armada tienen su guía docente, con la particularidad de que en las materias de Jefatura el Jefe de Departamento sería quien la aprobase.

3.4.2.7 *Casos de uso de Gestión Académica*

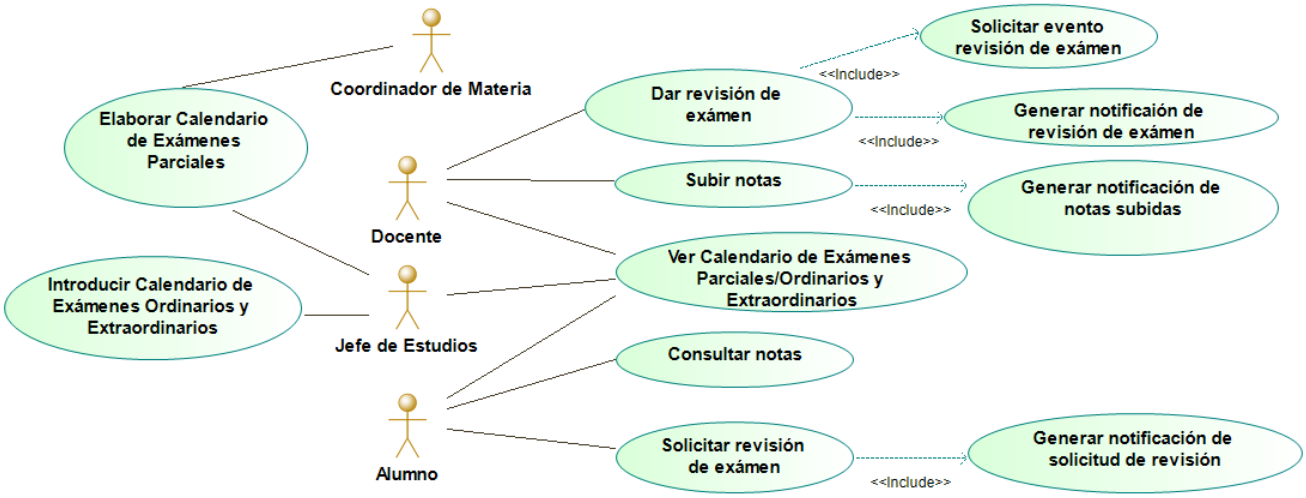

Ilustración 3-22 Diagrama de casos de uso de Gestión Académica (elaboración propia)

La Gestión Académica engloba lo relacionado con los exámenes del curso. Cabe diferenciar entre parciales y ordinarios, ya que al haber una reunión específica para poner fecha a los ordinarios y salvo causa mayor no se modifican. La propuesta sería que en la aplicación el Subdirector del CUD suba el Calendario de Exámenes Ordinarios y Extraordinarios de las materias del Grado y el Secretario de Estudios los de las materias de Jefatura. Sin embargo, para los parciales hay una mayor flexibilidad, por lo que la aplicación se podría hacer cargo de gestionarlo. Aunque nunca habrá conflicto por parte de los exámenes de las materias de Jefatura porque se realizan en las horas asignadas a dar la materia, sería útil igualmente que se participase y tuviesen los Docentes ENM conocimiento global del Calendario de Exámenes Parciales para distribuir en la medida de lo posible la carga de exámenes a los Alumnos.

3.4.2.7.1 Subcaso de uso de Calendario de Exámenes Parciales

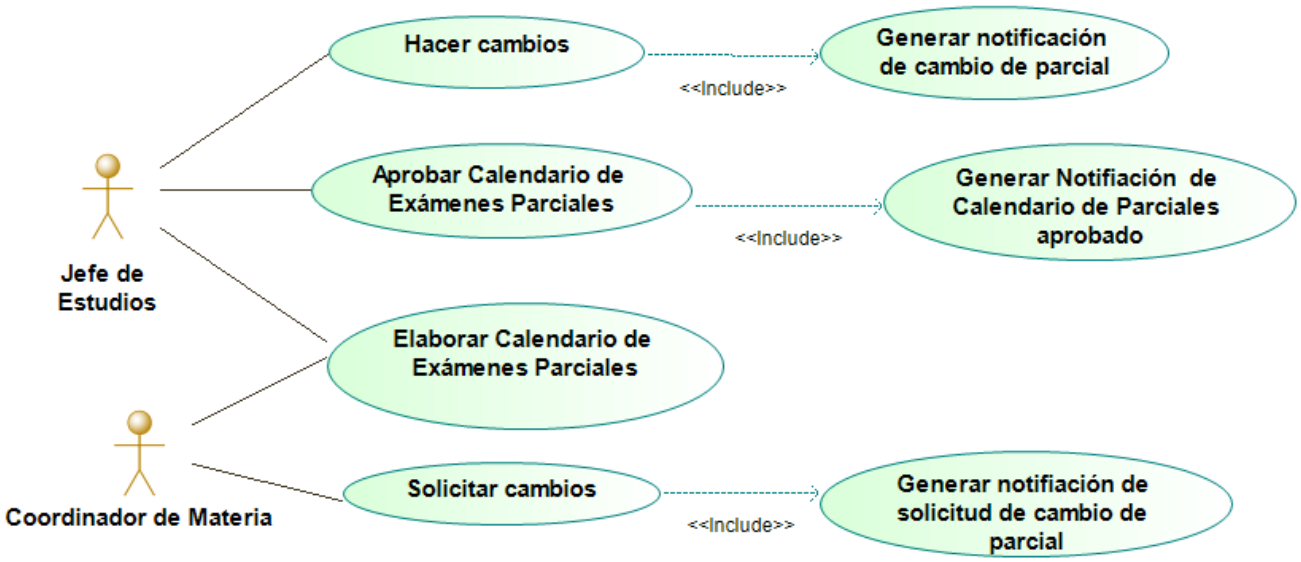

Ilustración 3-23 Subcaso de uso de Calendario de Exámenes Parciales (elaboración propia)

Respecto la elaboración del Calendario de Exámenes Parciales se propone que los del Grado los aprobase el Subdirector del CUD y los de Jefatura el Secretario de Estudios.

3.4.2.8 *Casos de uso de Control de Asistencia*

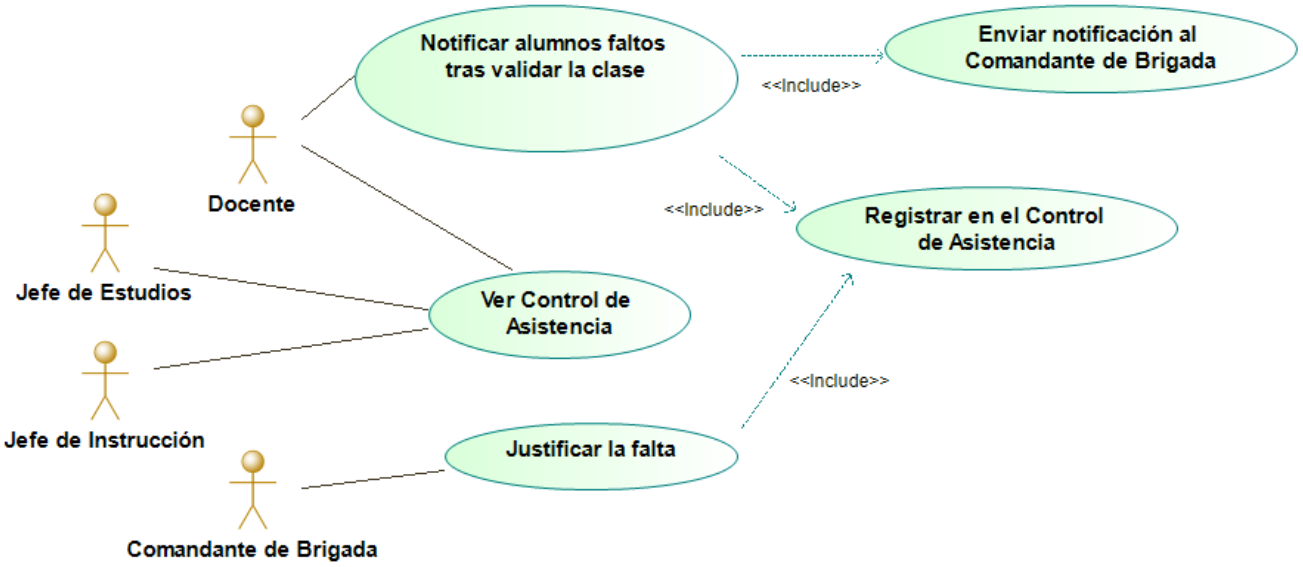

Ilustración 3-24 Diagrama de casos de uso de Control de Asistencia (elaboración propia)

A propuesta del Secretario de Estudios, sería útil el llevar un control de la asistencia de los alumnos. En caso de falta de asistencia sería el Docente quien lo notificaría al Comandante de Brigada, quien podría justificar esa falta si hubiese motivo suficiente.

3.4.2.9 *Casos de uso de Gestión Administrativa de los Datos*

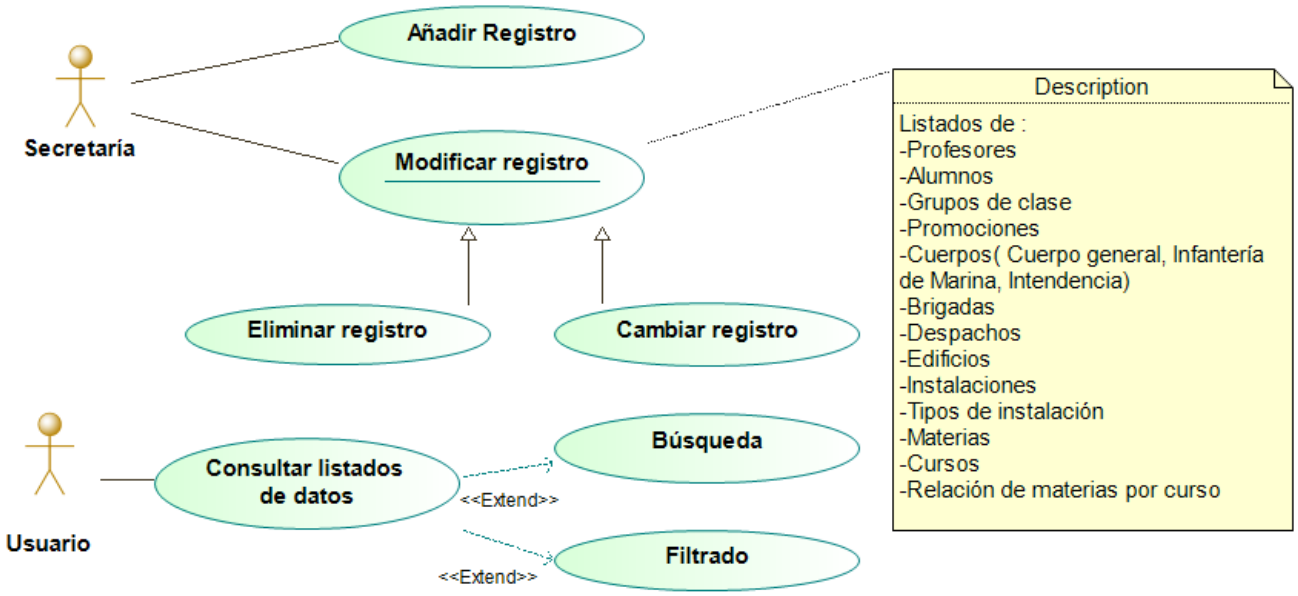

Ilustración 3-25 Diagrama de casos de uso de Gestión Administrativa de los Datos (elaboración propia)

La Gestión Administrativa de los datos englobaría el control de todos los datos de la gestión académica (grupos de clase, datos de contacto, materias, etc.). Para esta gestión se ha propuesto un perfil de Secretaría que se encargue de estas funciones. Además, los Usuarios podrían hacer búsquedas en los registros de datos.

3.4.2.10 *Casos de uso de Gestión de Usuarios*

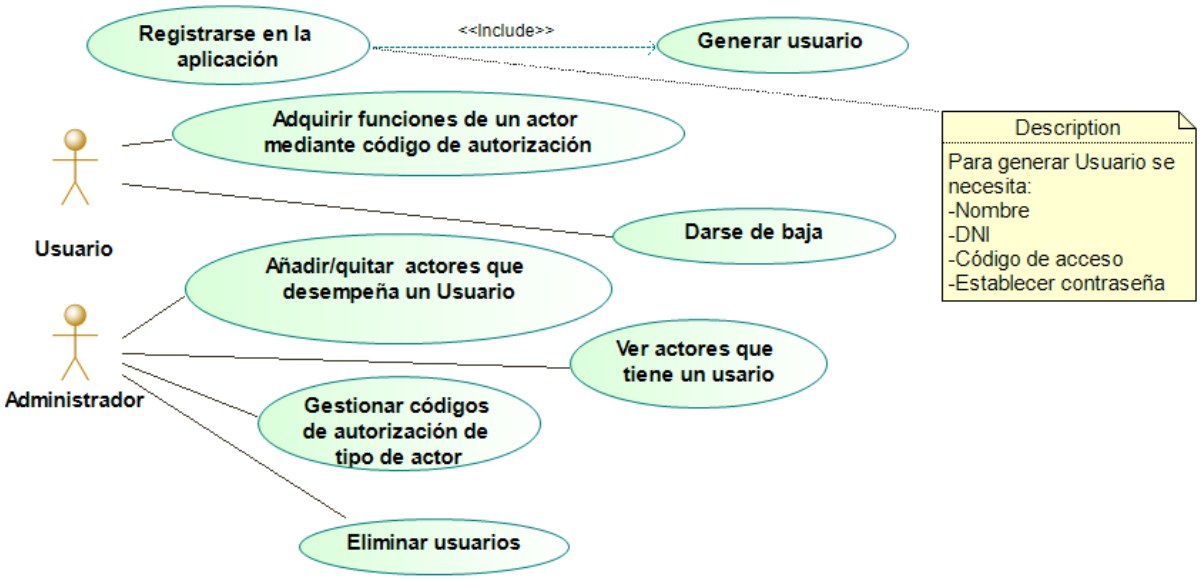

Ilustración 3-26 Diagrama de casos de uso de Gestión de Usuarios (elaboración propia)

En esta propuesta se contempla que todos los Usuarios estén registrados, para lo cual se prevé que el registro en la aplicación sea mediante nombre, DNI, código de acceso y una contraseña. El código de acceso lo establecería el Administrador, para que así solo quienes la ENM quiera puedan registrarse y crear un usuario, además el Administrador podría cargar en un Usuario uno o varios de los actores en función de las necesidades del Usuario. Para agilizar esta gestión se podrían establecer unos códigos de

desbloqueo de actor para que sea el propio usuario quien lo haga, así se evitaría que el Administrador tenga que dar de alta uno a uno a todos los actores de Alumno, por ejemplo.

3.4.3 *Diagramas de máquina de estados*

En los siguientes diagramas se reflejan los estados posibles que tendrían lo objetos más relevantes en la aplicación. Se han representado los diagramas que resultarían de utilidad.

3.4.3.1 *Diagrama de máquina de estados de Curso Académico*

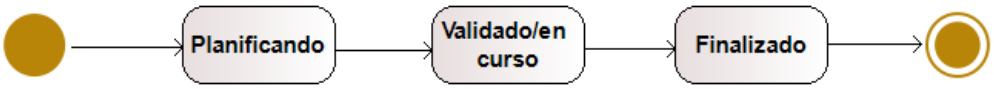

Ilustración 3-27 Diagrama de máquina de estados de Curso Académico (elaboración propia)

3.4.3.2 *Diagrama de máquina de estados de PDA*

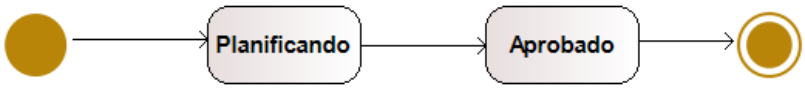

Ilustración 3-28 Diagrama de máquina de estados de PDA (elaboración propia)

3.4.3.3 *Diagrama de máquina de estados de POD*

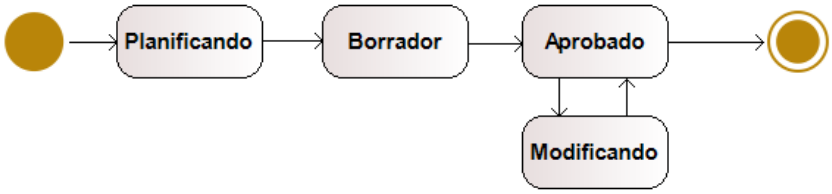

Ilustración 3-29 Diagrama de máquina de estados de POD (elaboración propia)

3.4.3.4 *Diagrama de máquina de estados de evento*

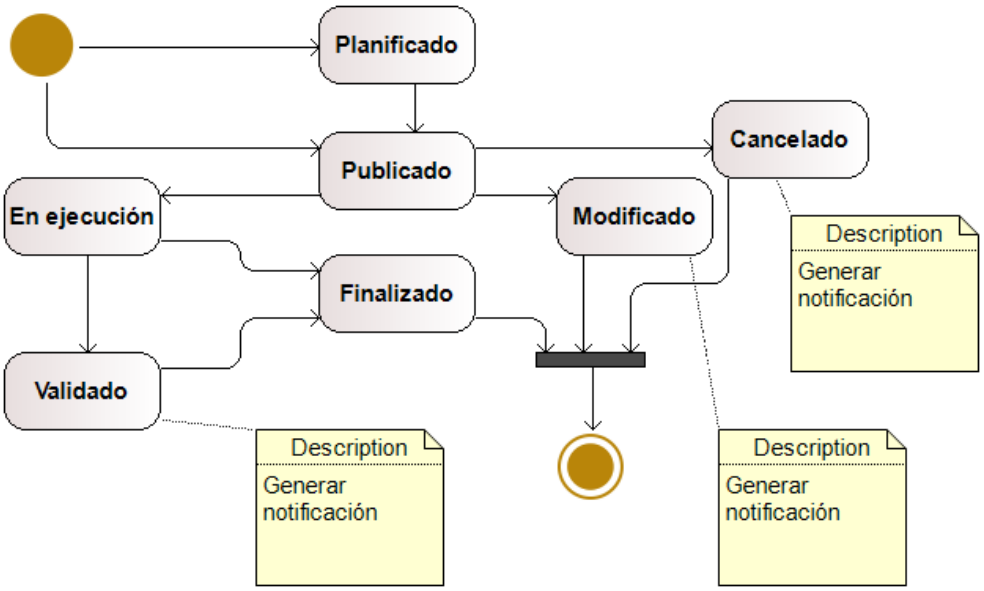

Ilustración 3-30 Diagrama de máquina de estados de evento (elaboración propia)

Cabe destacar que un evento en planificado solo sería visible para el Coordinador de Sábana. También que una vez realizado el evento podría quedar sin validar.

3.4.3.5 *Diagrama de máquina de estados de alarma de evento*

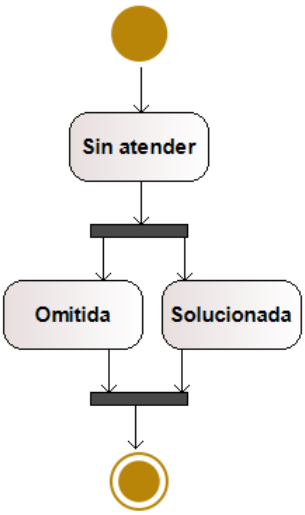

Ilustración 3-31 Diagrama de máquina de estados de alarma de evento (elaboración propia)

Una alarma podría ser omitida o bien al solucionar el motivo de alarma esta desaparecería.

3.4.3.6 *Diagrama de máquina de estados de solicitud*

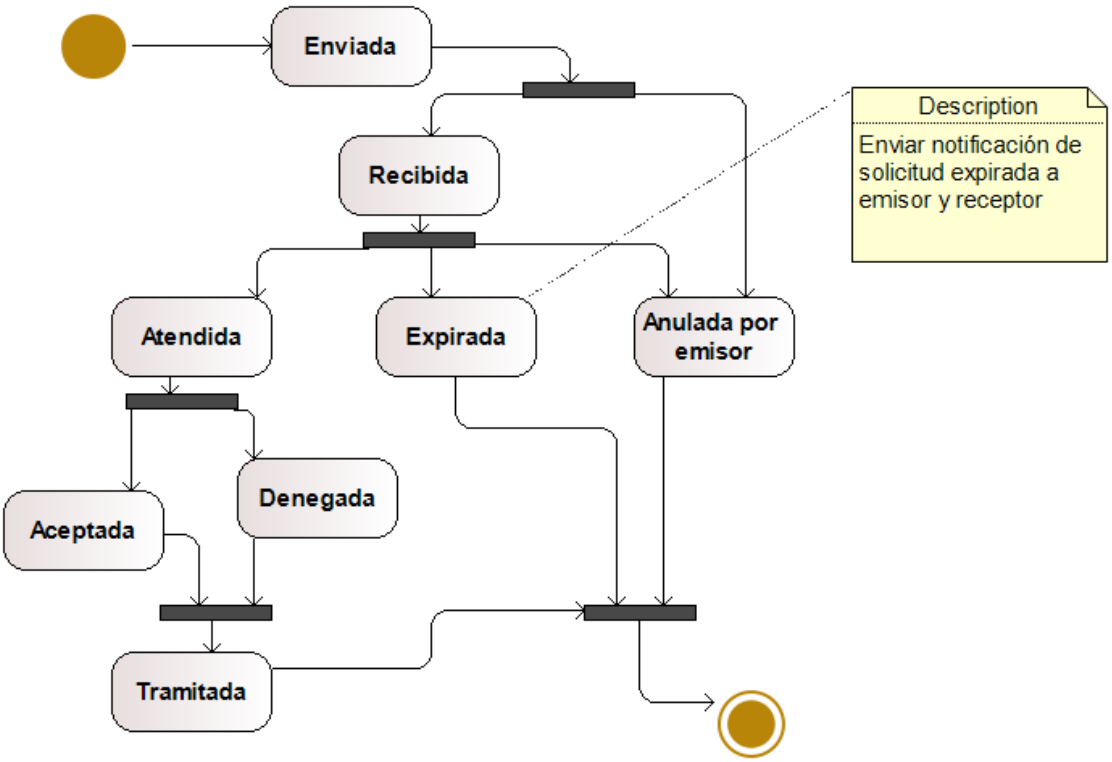

Ilustración 3-32 Diagrama de máquina de estados de solicitud (elaboración propia)

En caso de enviar una solicitud por error o por lo que fuese está perdiese su sentido el emisor podría anularla en cualquier momento. Además si el receptor no la atendiese en un tiempo establecido esta expiraría y se notificaría a emisor y receptor.

3.4.3.7 *Diagrama de máquina de estados de Calendario de Exámenes Parciales*

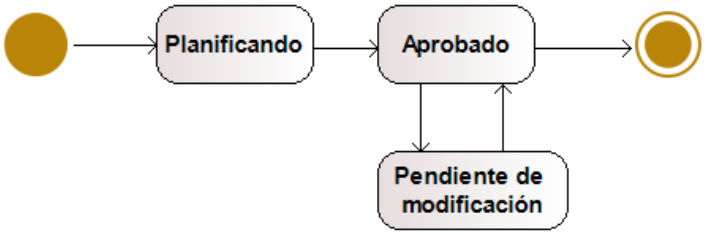

Ilustración 3-33 Diagrama de máquina de estados de Calendario de Exámenes Parciales (elaboración propia)

3.4.4 *Diagramas de actividad*

Por último, se detallan los diagramas que muestran una secuencia de lo que se contempla que puede ocurrir en la aplicación.

3.4.4.1 *Diagrama de actividad de Curso Académico*

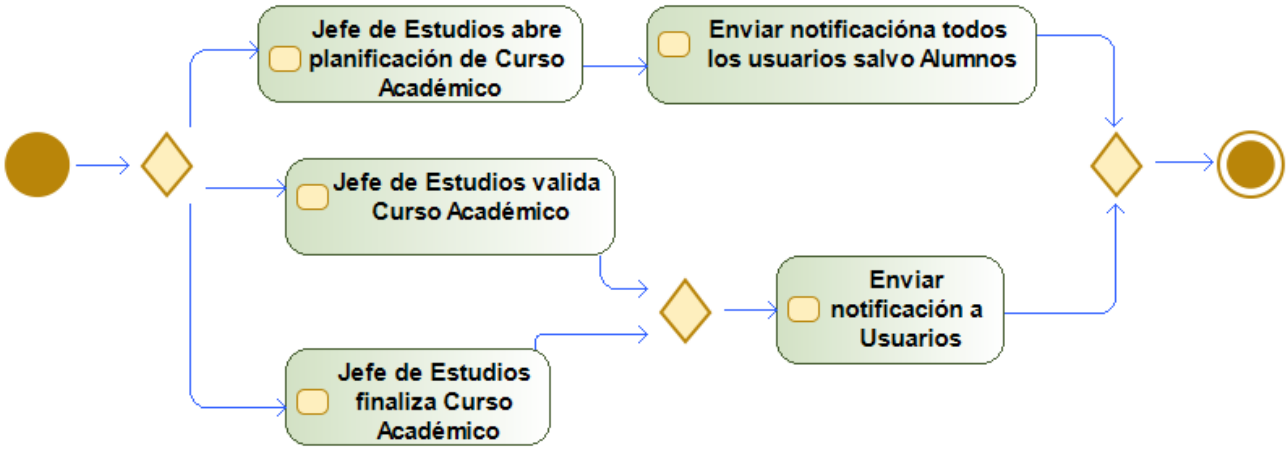

Ilustración 3-34 Diagrama de actividad de Curso Académico (elaboración propia)

3.4.4.2 *Diagrama de actividad de POD*

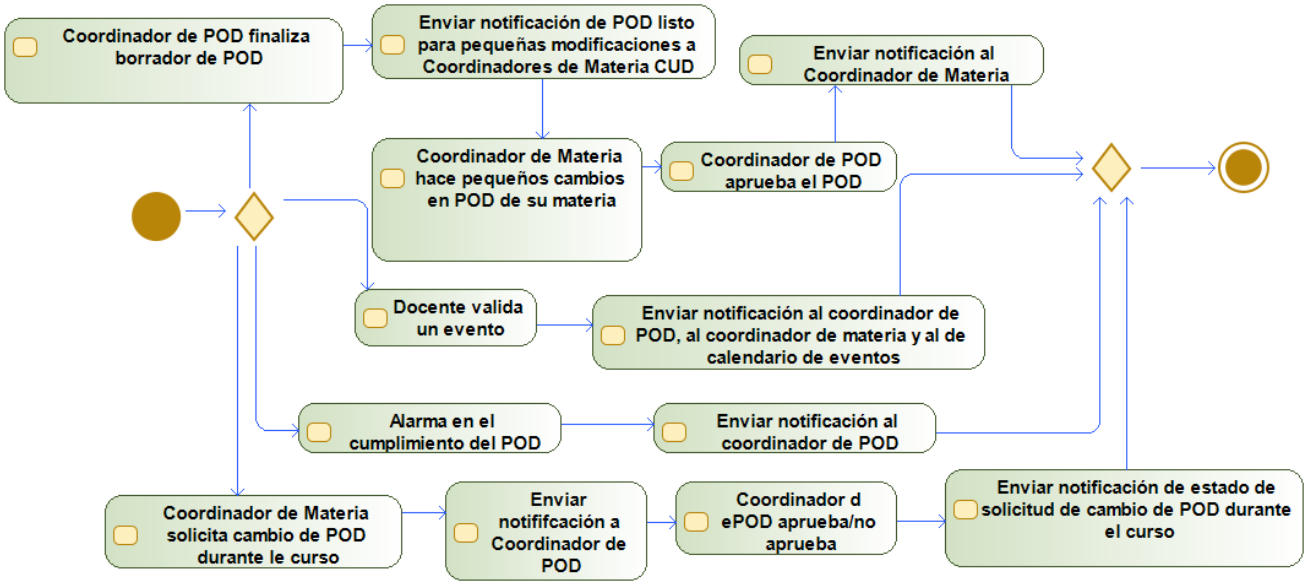

Ilustración 3-35 Diagrama de actividad de POD (elaboración propia)

3.4.4.3 *Diagrama de actividad de Guía Docente*

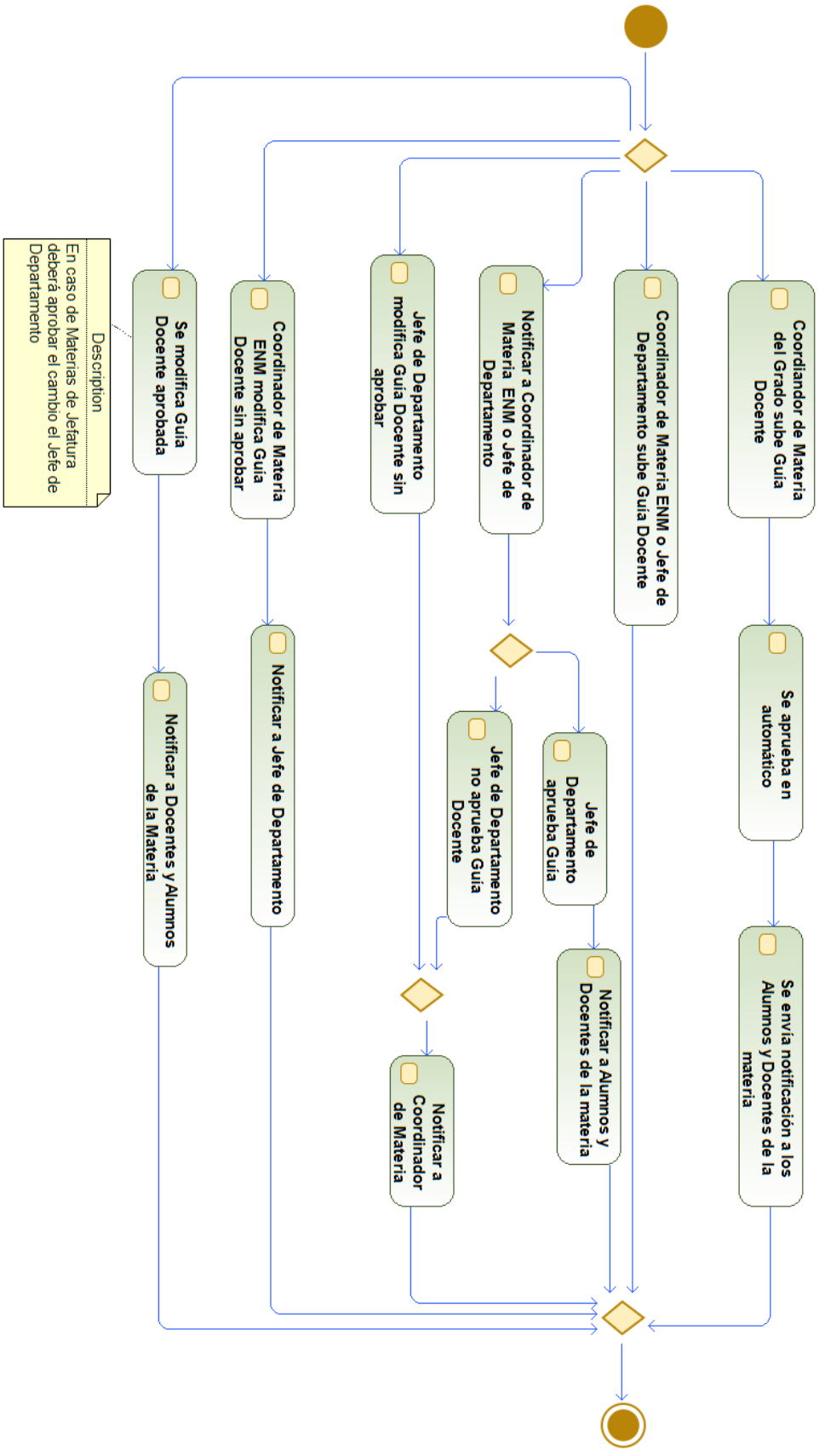

Ilustración 3-36 Diagrama de actividad de Guía Docente (elaboración propia)

3.4.4.4 *Diagrama actividad de Calendario de Eventos*

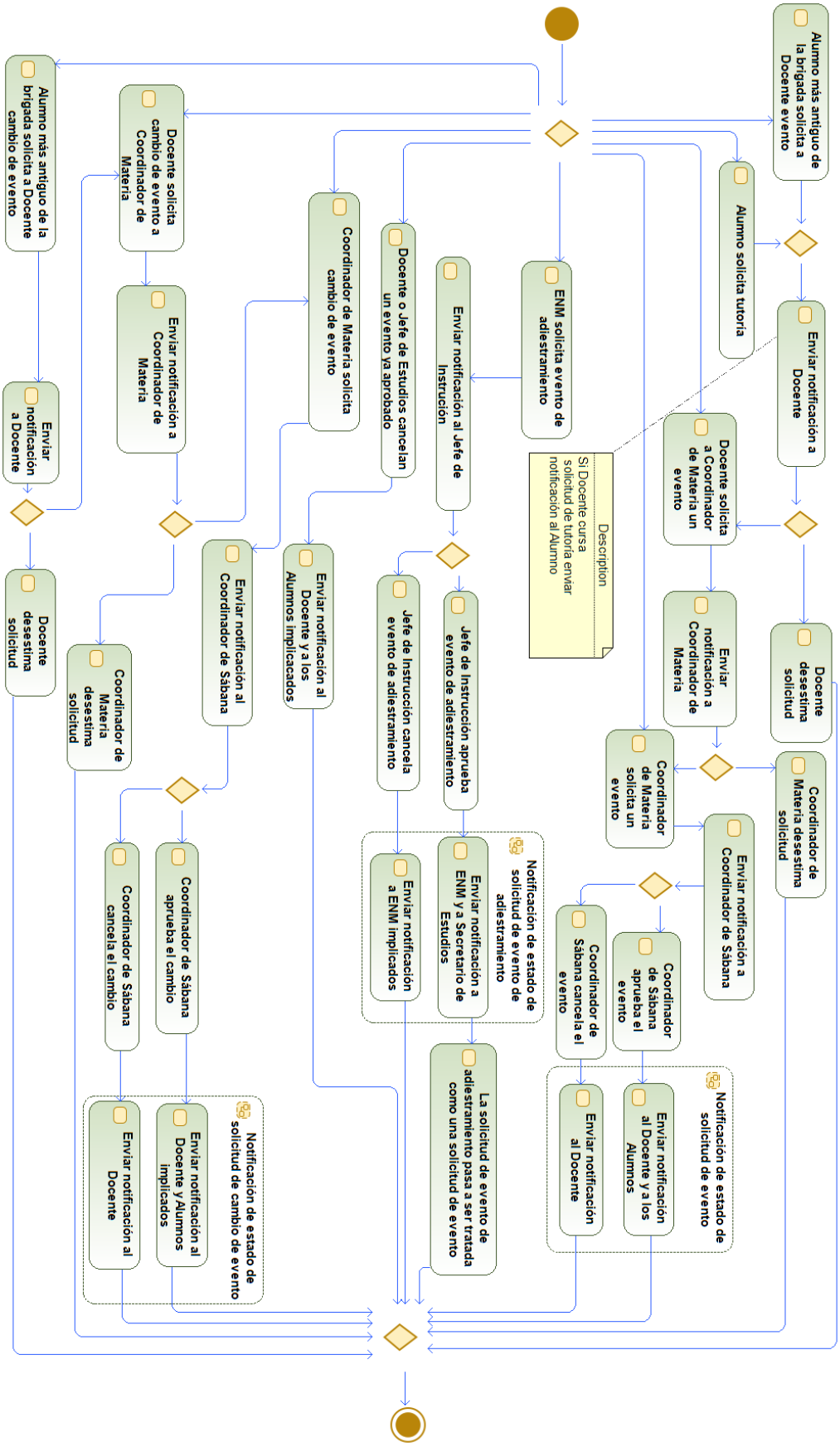

Ilustración 3-37 Diagrama de actividad de Calendario de Eventos (elaboración propia)

En la Ilustración 3-37 no se incluye la gestión de conflictos ya que en la Ilustración 3-15 queda suficientemente explicada la secuencia de acciones que sucederían. Lo mismo sucede con las alarmas de evento en la Ilustración 3-16.

3.4.4.5 *Diagrama de actividad de Gestión Académica*

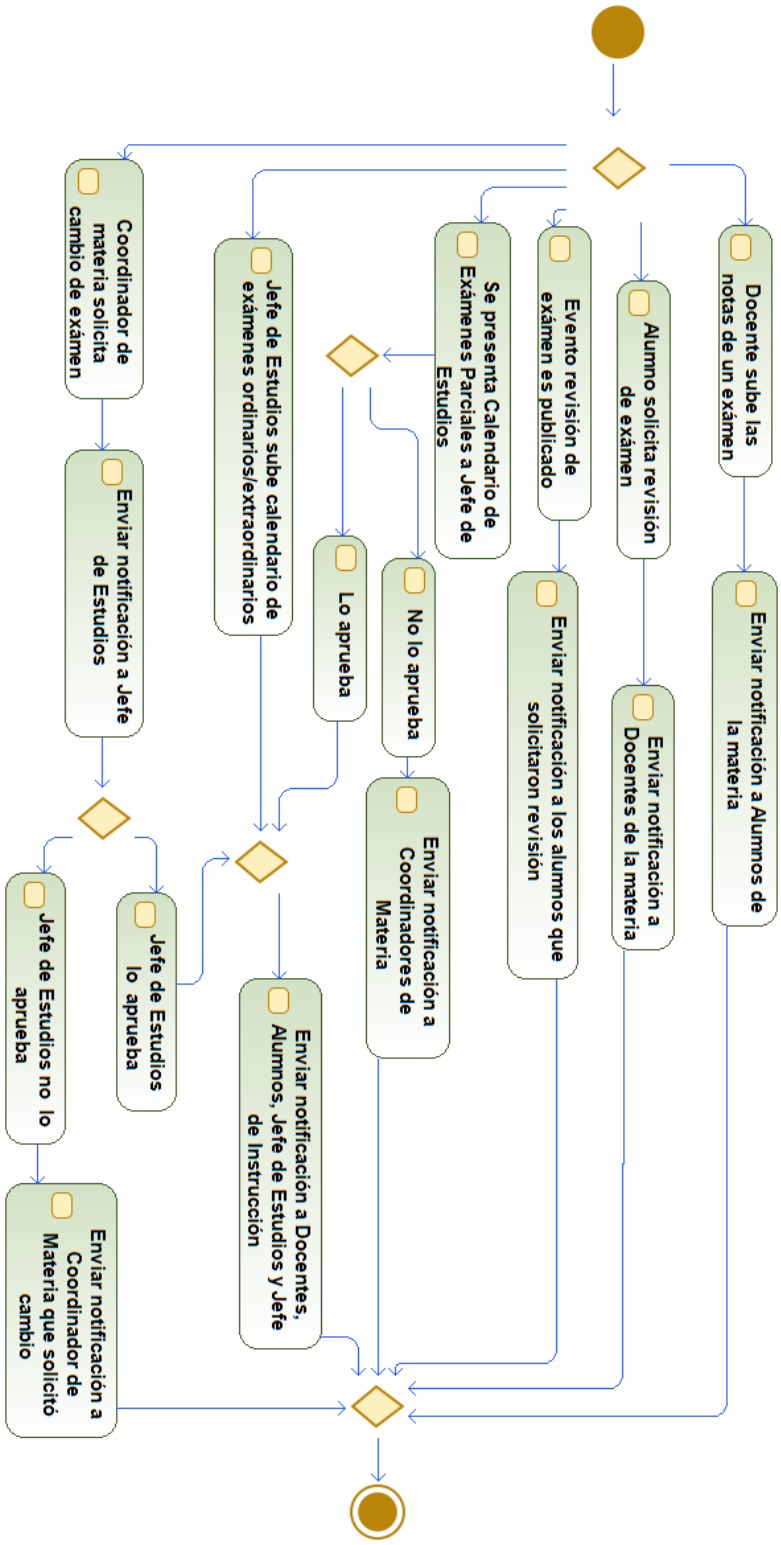

Ilustración 3-38 Diagrama de actividad de Gestión Académica (elaboración propia)

4 RESULTADOS

4.1 Valoración de los diagramas

Tras el desarrollo de este trabajo, se considera que se han cubierto todos los aspectos relacionados con la gestión de horarios, incluso abarcando aspectos que quizás no estén directamente involucrados. Pero que de una forma u otra condicionan el día a día de gestionar un horario con diversos planes de estudios y cursos. Como por ejemplo sucede con la PDA, los calendarios de exámenes, la Gestión Administrativa de los Datos, el Calendario de Actividades o el poder consultar Cursos Académicos anteriores como apoyo al planeamiento del siguiente.

Finalmente hay que resaltar que dichos diagramas son un análisis previo y documentación para que posteriormente se pudiese desarrollar una aplicación, por lo que su contenido podría cambiar durante el desarrollo de la misma o si se necesitasen cubrir otros aspectos o puntos de vista sobre qué fuese necesario en una aplicación de ayuda a la gestión académica.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Durante la recabación de información para el análisis previo de la aplicación de ayuda a la gestión académica del CUD se llegó a la conclusión de que la aplicación debe abarcar toda la ENM, ya que el Grado en Ingeniería Mecánica es parte del currículum necesario para ser Oficial de la Armada, y también porque en esta escuela fruto del currículum existen en el horario tanto las asignaturas del Grado como las de ámbito militar. Por lo que es necesario hacer una aplicación que abarque toda la formación que se imparte en la ENM para optimizar el horario y mejorarlo.

También cabe destacar que, de realizarse esta aplicación, no desaparecerían todas las inclemencias de gestionar tanto personal y asignaturas, pero liberaría de carga de trabajo a los encargados de la gestión académica en gran medida.

5.2 Líneas futuras

5.2.1 *Nuevos campos para el análisis.*

Este análisis se centra en la gestión del horario, pero dada la naturaleza de UML este trabajo es escalable. Por lo que se puede ampliar añadiendo nuevos campos, como añadir la gestión de inventarios de los laboratorios, la gestión de solicitud y propuesta de Trabajos de Fin de Grado o incluso una Orden Diaria interactiva y dinámica que generase avisos al personal involucrado en alguno de sus campos, como mandar notificación de que ese día cierta brigada tiene lavandería, una modificación al horario para el equipo de regatas o cualquier aviso de interés.

5.2.2 *Programación de la aplicación.*

Finalmente, la documentación de la gestión del horario de este trabajo servirá de base para que en un futuro se comenzase a desarrollar la aplicación.

6 BIBLIOGRAFÍA

[1] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado#:~:text=El%20lenguaje%20unificado%20de%20modelado,Object%20Management%20Group%20(OMG).

[2] J. A. Carreño, «calderon.cud.uvigo,» [En línea]. Available: http://calderon.cud.uvigo.es/bitstream/handle/123456789/260/Albadalejo%20Carre%c3%b1o%2c%20Julio%20-%20Memoria.pdf?sequence=1&isAllowed=y.

[3] F. J. R. Fontán, «calderon.cud.uvigo,» [En línea]. Available: http://calderon.cud.uvigo.es/bitstream/handle/123456789/307/Ruiz%20Font%c3%a1n%2c%20Francisco%20Javier%20-%20Memoria.pdf?sequence=1&isAllowed=y.

[4] K. Panjaphon, «calderon.cud.uvigo,» [En línea]. Available: http://calderon.cud.uvigo.es/bitstream/handle/123456789/233/Kathong%2c%20Panjaphon%20memoria%20definitiva%20para%20publicar.pdf?sequence=1&isAllowed=y.

[5] «Visual Paradigm,» [En línea]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-practical-guide/.

[6] «Modeliosoft,» [En línea]. Available: https://www.modeliosoft.com/en/.

[7] «Modelio 4.1».

[8] NICOSIO, «Youtube,» [En línea]. Available: https://www.youtube.com/watch?v=-OWd0tJAK10&list=PLM-p96nOrGcaw5dhv8wOA5tVVWEmXtA2F.

[9] «Synergix,» [En línea]. Available: https://synergix.wordpress.com/2008/07/31/las-4-mas-1-vistas/.

[10] Secretario de la ENM, LIBRO DE ORGANIZACIÓN Y RÉGIMEN INTERIOR DE LA ESCUELA NAVAL MILITAR, 2020.

[11] «cud,uvigo,es,» [En línea]. Available: https://cud.uvigo.es/index.php?option=com_content&view=article&id=62&Itemid=41.