CURSO 2019 - 2020

ACTIVIDAD EVALUABLE Y CALIFICABLE

1. Portada con:

Asignatura: 71013035 – Diseño de Software

Año de la práctica:

Centro Asociado al que pertenece:

Tutor que le da asistencia:
(o grupo de tutoría Intercampus al que está adscrito)

Datos personales: Nombre y apellidos:

Localidad de residencia:

Datos de coste: Horas de dedicación al estudio de los contenidos:

Nº de actividades no evaluables realizadas y horas de dedicación:

Horas de dedicación para realizar esta actividad:

Juan del Rosal, 16
28040, Madrid
Tel: 91 398 89 10
Fax: 91 398 89 09
swdesign@issi.uned.es

2. El enunciado y planteamiento del caso de estudio.

El escenario en el que se situará el funcionamiento del caso de uso (pregunta 2), consiste en una aplicación para la **gestión de los recursos hídricos** de un territorio (aplicación llamada `HidroGest`).

Los recursos hídricos se refieren al agua de la superficie terrestre expuesta a la atmósfera y a lo relacionado con su tránsito, utilización y aprovechamiento.

Para su administración territorial, los recursos se agrupan en `Regiones`, que contienen `Cuencas` en las que cada una, a su vez, está constituida por todas las corrientes que drenan, finalmente, en un único lugar: el mar o un lago endorreico.

Para la regulación y el aprovechamiento del agua se utilizan embalses o `Presas`: acumulaciones *forzadas* situadas en los cauces de las `cuencas`.

El objetivo principal de la gestión hidrológica es poder realizar el seguimiento de la disponibilidad del agua y otros factores de su uso. Para ello, la `Medida` es el elemento fundamental que lo permite. En este escenario, simplificado, las `medidas` son volumétricas (hm^3) y se recogen, mediante el `sensor` correspondiente, en cada `Presa`. Es decir, se refieren al volumen de llenado de la `Presa`. Por consiguiente, la información más notable de una `Medida` (asociada a una `Presa`) es su valor (cantidad de agua acumulada, en hm^3) y la fecha en que se ha tomado.

Con esta red de sensores, el sistema `HidroGest` monitoriza las medidas recibidas durante una ventana temporal (tanto la frecuencia de muestreo como la amplitud de la ventana son configurables) mediante un conjunto de `Reglas` de supervisión y con la colaboración de un motor de razonamiento o `Sistema Experto` externo. Al final de cada día se almacena, de forma persistente, la última medida recogida, durante ese día, en cada `Presa`.

En definitiva, la funcionalidad proyectada para `HidroGest` se basa en:

- Gestión (CRUD) de los recursos hídricos. Se refiere a todos los datos e información asociada a los recursos que maneja el sistema para su funcionamiento (`Regiones`, `Cuencas`, `Presas`, `Medidas`, `Sensores`, etc.). Dicha información se almacena, de forma persistente, **en un sistema externo a `HidroGest`**.
- Configuración y mantenimiento de los parámetros del funcionamiento de la aplicación: reglas de supervisión, frecuencias de muestreo de los sensores, ventana de monitorización, zonas de enfoque, estrategia de almacenamiento de medidas, etc.
- Consultas de las medidas almacenadas. En general, una consulta se podría plantear con estos 3 aspectos:
    - Objetivo o qué se busca: puede ser las medidas o sus variaciones.
    - Campo o dominio de búsqueda: se puede referir a una `Presa`, a una `Cuenca` (en cuyo caso significa obtener el acumulado de todas sus

Presas) o a una Región (en cuyo caso significa obtener el acumulado de todas sus Cuencas y, así, recursivamente).

- Restricciones. Se refiere a cualquier grupo de condiciones que deban cumplir el objetivo o el dominio de búsqueda. Por ejemplo, '*entre 2 fechas*' o '*cuyo valor supere una cantidad*'.

Cualquiera de estos términos determina el tipo de la consulta, cómo se obtiene y el formato del resultado.

- Monitorización de la red y gestión de alarmas. El sistema permite realizar el seguimiento del estado de los recursos de la red, presenta alarmas o enfoca la atención en una zona de interés para tomar decisiones al respecto.

Detalles y simplificaciones admitidas:

- Como en el resto de la asignatura, la atención del estudio se dirige a la capa de la lógica de la aplicación, a los objetos del dominio del negocio y los mínimos servicios técnicos de apoyo que permitan interactuar con el acceso a los datos u otros sistemas considerados externos. Por tanto, la capa de presentación se considerará *transparente* y la interacción entre los actores humanos y la lógica del negocio será directa (como si se tratara de una comunicación mediante lenguaje de invocación de funciones a través de comandos). Igualmente, la interacción entre la lógica del negocio y los sistemas de apoyo externos se realizará a través de adaptadores.
- Todos los datos e información que requiere el funcionamiento de HidroGest están almacenados, de forma persistente, **en un sistema externo** de cuyo manejo *directo* **no debe** ocuparse la aplicación.
- La capacidad de una Presa es un dato inherente a ella. Sin embargo, la de una Cuenca depende de las presas que tenga registradas y se calcula como la suma de sus capacidades. De igual forma, la capacidad de una Región se calcula como la suma de las capacidades de las Cuencas que tenga registradas. Lo mismo ocurre con la ocupación de estos recursos: se calculan como la suma de los niveles de llenado del grupo de recursos subordinado.
- Dada la correlación existente entre los datos almacenados (que debería mantenerse en el funcionamiento de la aplicación), como simplificación, se considerará que la selección de un recurso en la IU lleva implícita la del recurso al que está subordinado, en el funcionamiento de la aplicación. Por ejemplo, la selección de una Presa implica, también, la selección de la Cuenca y la Región a la que pertenece.

Material recomendado para la realización de esta prueba:

- Libro de texto de la asignatura.
- Manuales y documentación adicional para el enfoque y comprensión de los contenidos del libro y de la asignatura. Referenciados en las indicaciones para el estudio y realización de las actividades en la guía de la asignatura, en su página web informativa y en el curso virtual.
- Soluciones propuestas para las PEC de los cursos 2016, 2017 y 2019.

Los contenidos de este tipo de cuadros son recomendaciones y ayudas para enfocar el planteamiento de las preguntas y elaborar las respuestas en esta actividad. **Dichas recomendaciones no suelen aparecer en los enunciados del examen.**

3. **El enunciado de cada cuestión y las respuestas.** Para cada cuestión, incluirá los desarrollos, listados, diagramas y las argumentaciones que estime necesarios.

Sección 1. Evaluación de los ***Casos de Uso***

1. (0'5 puntos) Represente, en un diagrama UML de casos de uso, los casos de uso primarios más importantes, sus actores principales y de apoyo y las interacciones correspondientes, para la aplicación `HidroGest`.

Los Casos de Uso **no son orientados a objetos**. En la construcción progresiva del Modelo de Casos de Uso inicialmente se utilizan elementos que no son software (ni código, ni clases ni sus relaciones), sino que se refieren a cómo se usa la aplicación y qué información se necesita para ello.

No es hasta la implementación del funcionamiento de cada caso de uso (en el diseño, en la pregunta 4.1 y siguientes), cuando aparecerán las clases y cómo colaboran sus instancias para describir ese funcionamiento.

En este diagrama, la aplicación `HidroGest` delimita la principal funcionalidad que contiene (sus casos de uso primarios). Fuera de ella, se representan los actores primarios, su interacción (mediante la que manejan determinados casos de uso) y los actores de apoyo que, mediante su interacción correspondiente, necesita cada caso de uso para cumplir con su funcionalidad.

Se recuerda: de aquí en adelante, **todas las preguntas** se refieren a las especificaciones **definidas en la siguiente pregunta y para ese caso de uso**. El objetivo es que realice el diseño para que también admita las otras funcionalidades y opciones de la aplicación.

2. (1 punto) Escriba el caso de uso *<<AgregarNuevaPresa>>* en un formato completo (se recomienda la variante '*en dos columnas*') y estilo esencial. Incluya tanto el escenario principal de éxito como 2 extensiones o flujos alternativos que pudieran ser frecuentes.

Consiste en agregar toda la información necesaria relativa a una presa nueva: nombre, capacidad, cuenca a la que pertenece y sensor asociado que realiza las medidas.

Nótese que cada `Presa` tiene asociado un `Sensor` (o grupo de ellos) que provea sus medidas. Se considera, por tanto, que el alta de una `Presa` requiere el alta del `Sensor` correspondiente (por conveniencia, en otro caso, podría no ser así); cuya información es: fabricante, marca, modelo, características de funcionamiento y medida, estado y número de serie.

No escriba un encabezamiento demasiado elaborado del caso de uso (es decir, omita *propósito*, *resumen, antecedentes*...); en su lugar, afronte directamente el transcurso típico de los acontecimientos.

Recuerde que la escritura del caso de uso es un relato de la secuencia ordenada de '*lo que hace el actor*' y '*la respuesta que obtiene*' (la que ve él y, como mucho, algún hito importante en el cumplimiento de su objetivo de uso) al utilizar la aplicación.

Tenga en cuenta que el actor, aunque tenga la voluntad de conseguir su objetivo, no va a realizar ninguna acción a no ser que el sistema la presente como única alternativa para conseguir ese objetivo. Por tanto, es el sistema (sin voluntad, pero con una programación determinista en su comportamiento) el que guía las acciones del actor.

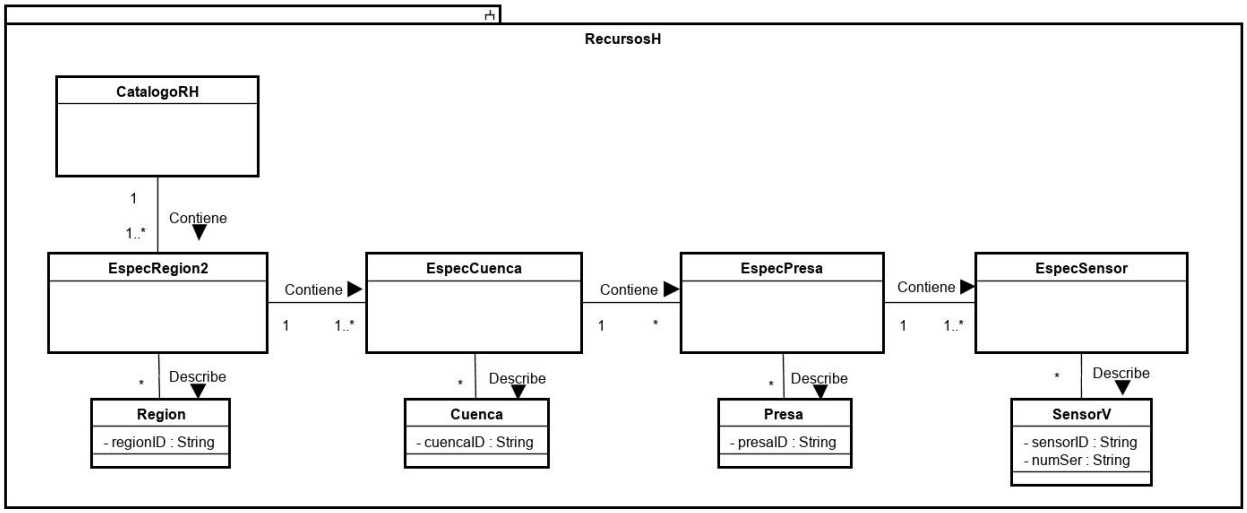
En el relato de esta secuencia, olvídese de lo que hace la IU, la capa del negocio, los actores de apoyo externos o su repercusión en ellos. Considere *lo que no es el actor principal* como una *caja negra* y limítese a describir la secuencia de lo que pasa entre él y dicha caja negra.

Sección 2. Evaluación del ***Modelado Conceptual***

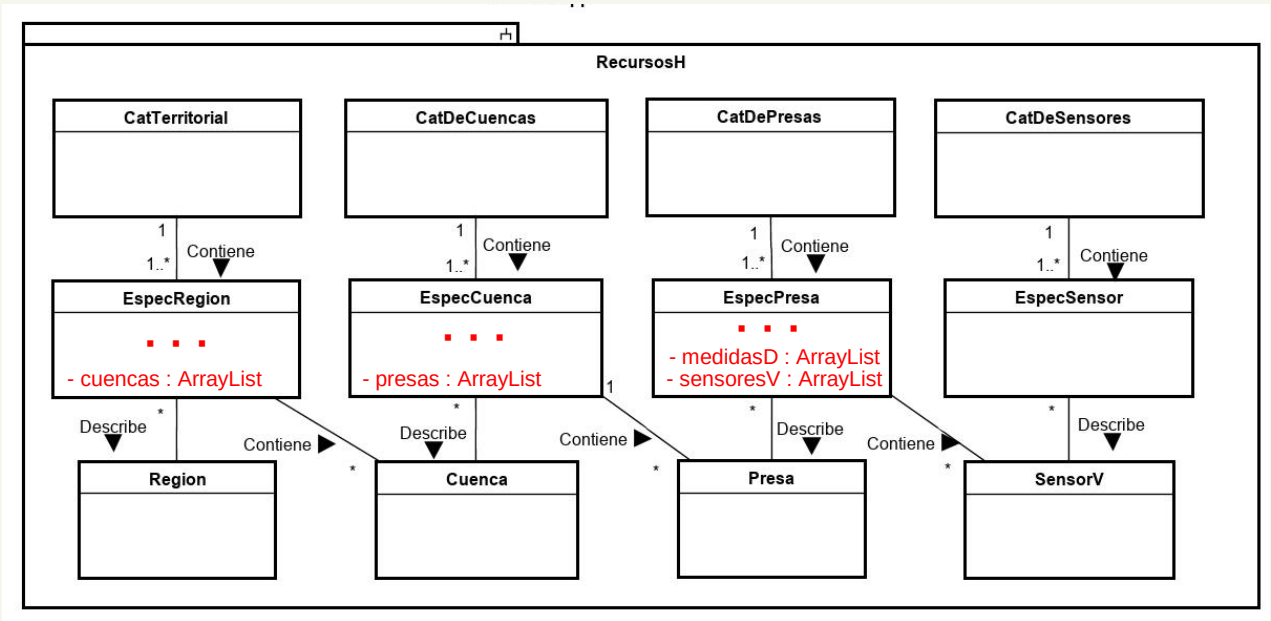
3. (2 puntos) En relación con el caso de uso anterior, *<<AgregarNuevaPresa>>*, construya un Modelo de Dominio y represéntelo en notación UML. Represente los objetos conceptuales, las relaciones relevantes entre ellos, su cardinalidad y los atributos *candidatos* de los objetos.

Sugerencias:

- Las relaciones lógicas entre los datos, que se desprenden de la presentación, sugieren una organización de esta forma:



Esta estructura está <u>fuertemente acoplada</u> y, si organiza el funcionamiento del caso de uso en torno a ella, es muy probable que se obtenga un diseño inaceptable. Por ello, se propone desacoplar las colecciones (con los datos que se manejan en el caso de uso) entre sí:



- Independientemente de cómo se maneje la información en la IU, la llave para el manejo de colecciones mediante catálogos es su organización en función de una clave de búsqueda; que no suele ser una cadena de texto porque no es manejable. En la estructura anterior, cada elemento está representado por un identificador único y cada colección contiene todos los elementos de ese tipo (por ejemplo, todos los sensores… de todas las presas, cuencas, etc.). En el caso del alta de un recurso, **el sistema no tiene capacidad para asignarle un identificador** único. No es hasta que se ha actualizado en el almacén (base de datos) que éste no le asigna dicho identificador.

¡¡Atención!! Los diagramas anteriores **no son el modelo de dominio** de este caso de uso. Sólo son una propuesta de cómo organizar los datos de los recursos hídricos en un módulo dedicado a ellos ('`RecursosH`').

En el modelo de dominio se representan los objetos conceptuales que, según la descripción del comportamiento deseado para el caso de uso, son necesarios para hacerlo posible. Por consiguiente, es muy probable que el funcionamiento descrito requiera sólo parte de los objetos representados en `RecursosH` y, además, también requiera otros que no están.

Los '*objetos conceptuales*' no son clases software y, como elemento constructivo de la representación del funcionamiento deseado, su única coincidencia es que encapsulan tanto los datos como las acciones que se pueden hacer con ellos (como los TAD).

Pero en la concepción de ese funcionamiento, y en su construcción mediante esos objetos conceptuales, es muy importante tener presente por qué se pueden utilizar **con independencia** del resto de la infraestructura software de la aplicación y de su entorno (los otros componentes de su arquitectura y los actores de apoyo externo), y, por consiguiente, **pueden ser como mejor convenga al funcionamiento que se está construyendo**. La razón son mecanismos puramente software que, aunque no se requiere implementar en este ejercicio, **es fundamental comprenderlos**: las interfaces y su manejo.

Los elementos de apoyo externos a la aplicación intercambian datos con ella. Pero, en ese intercambio, se debe salvaguardar, a ultranza, la independencia entre cada parte. Pongamos, por ejemplo, el almacén de los datos, con toda la información que utiliza la aplicación. Puede ser una base de datos, con un sistema que la gestiona o, en su lugar, lo que gestiona es 1 o varios ficheros, locales o distribuidos, etc.

Ese sistema software (externo) que maneja esos datos (externos, aunque se utilicen en la aplicación) lo hace según su propio funcionamiento, completamente ajeno (y oculto) al funcionamiento de la aplicación que se está diseñando. Pero tiene una '*puerta*' (una interfaz) a través de la cual intercambia servicios y datos, en la manera y en el formato que ese sistema externo establece: su fachada.

En el lado de la aplicación, para que su funcionamiento y los datos que maneja no dependan de lo que dicte esa fachada, se restringen todas esas dependencias en otra interfaz, situada en la capa del servicio de acceso a datos (la de servicios técnicos): un adaptador específico para esa fachada. El papel del adaptador es *traducir* todos los intercambios entre la aplicación y la fachada del sistema externo.

Una vez conseguida la independencia de la aplicación respecto a ese sistema externo concreto, la versatilidad se consigue mediante el polimorfismo, a través de una interfaz abstracta con capacidad para resolverse en diversos adaptadores, cada uno especializado en una fachada específica.

La última pregunta es: ¿Cómo se transmite esta independencia y flexibilidad al funcionamiento de la aplicación y de cada caso de uso?

La respuesta está en el funcionamiento de la aplicación en su *nivel del gobierno*, también denominado *gestión del flujo de trabajo* o, simplemente, *nivel de aplicación*: cuando un caso de uso necesita alguna estructura de datos (un *objeto conceptual* o una clase) cuyo contenido proviene y se mantiene desde un sistema externo, es el controlador de nivel superior al caso de uso (típicamente, el del nivel de aplicación) el que crea esa estructura y, al hacerlo, le *inyecta* el adaptador específico que le permite acceder a la fachada de ese sistema externo y, a la vez, replica toda la funcionalidad que la estructura realiza con su contenido privado, pero con los datos externos.

En general, la estructura o la clase de la que estamos hablando es privada o de visibilidad restringida en un módulo o en un ámbito determinado. El mecanismo por el que el controlador del nivel de aplicación consigue inyectar el adaptador especializado como componente de esa clase, consiste en utilizar una factoría de servicios de visibilidad global.

Una factoría es una clase puramente software. Una factoría de servicios tiene como componente una interfaz abstracta de cada servicio (de acceso) que da. La factoría de servicios es una especie de *máquina* que, al invocar alguno de sus servicios, provee el adaptador correcto.

De esta forma, el controlador de nivel superior, al crear la estructura que necesita utilizar el caso de uso, invoca el servicio de acceso al sistema externo (casi desde cualquier parte, puesto que su visibilidad es global), obtiene el adaptador especializado para ese servicio y lo transfiere a la estructura (como argumento de su constructor, por ejemplo).

Por consiguiente, **el único centro de atención al construir el funcionamiento del caso de uso es cómo organizar los datos**, de la forma más conveniente para que cada objeto pueda manejarlos y hacer lo que pretende; no dónde están almacenados esos datos ni cómo se manejan allí.

En el modelo de dominio no deben aparecer ninguno de los objetos software que se han mencionado anteriormente, ni los adaptadores, ni los sistemas de apoyo externos con los que se interactúa. Sólo se incluyen los objetos cuyo rol describe el funcionamiento *cotidiano* en la utilización de la aplicación para el caso de uso y, si en ello se hace imprescindible, la estructura de los datos que resulta de esa interacción con el exterior y que se utilizan en el caso de uso.

Si los datos de los que se está hablando se organizan en colecciones (en las que se requiere hacer búsquedas, modificaciones de sus elementos, etc.) es necesario un contenedor cuya responsabilidad sea gestionar su manejo (un catálogo) y la conexión para acceder al sistema externo (el adaptador) se implementa en él, no en cada uno de sus elementos.

Sección 3. (Diseño) Evaluación de los ***Eventos del Caso de Uso*** y ***Asignación de Responsabilidades***.

4. ***Eventos, Responsabilidades y Contratos***.

4.1. (2 puntos) Circunscrito al caso de uso anterior *<<AgregarNuevaPresa>>*, construya un Diagrama de Secuencia detallado (diagrama de interacción DS) en UML. Represente el actor, sus eventos y el paso de mensajes con las clases software del sistema para este caso de uso (en principio, las instancias de los objetos conceptuales que ha usado en el modelo de dominio).

¡ATENCIÓN!: se pide un diagrama de secuencia en el que represente el paso de mensajes entre el actor, el objeto que los recibe en el sistema y cómo se reparten entre **los distintos objetos del modelo**, y **NO** globalmente entre el actor y el sistema (el denominado Diagrama de Secuencia del Sistema –DSS–). Por tanto, **represente las líneas de tiempo de los objetos identificados en el modelo**, **NO** las interacciones entre los actores y una única línea temporal correspondiente al objeto ***sistema global***.

Esta es la pregunta más importante de todo el ejercicio. Debe contener la descripción, detallada y exacta, del funcionamiento del código para el caso de uso. Por consiguiente, **constituye una solución final del diseño pedido**.

Honestamente, la respuesta completa y correcta a esta pregunta debería significar la superación de todo el ejercicio. Sin embargo, esta respuesta depende de las respuestas a las preguntas anteriores que, a su vez, justifican de dónde salen los contenidos de este diseño y las respuestas a las siguientes preguntas.

Recíprocamente, la mayor parte de los errores de esta solución final están originados por defectos en los planteamientos o en las consideraciones de las respuestas elaboradas para las preguntas anteriores.

Se debería comenzar por las líneas de tiempo del actor que maneja el caso de uso y la de la instancia de la clase que ejerce de controlador (único).

Cada acción del actor (estímulo o evento) es una invocación de un método del controlador (con sus argumentos, tipos, etc.).

Cada evento que recibe el controlador produce una secuencia de invocaciones a métodos, bien pertenecientes a otras clases (a sus instancias) o bien de creación de las instancias de otras clases (que se convierten en componentes suyos, del creador).

En principio, aparte del actor y del controlador, las instancias que intervienen en este intercambio de mensajes (llamadas a métodos) son los objetos conceptuales del modelo de dominio.

Se recomienda aprender los mecanismos estándar de: obtención y manejo de un valor como respuesta a la llamada de una función, búsqueda de un elemento en una colección, creación e inicialización de una instancia, agregación de un elemento a una colección, etc.

Sin embargo, en ocasiones, la resolución de determinadas operaciones requiere la utilización de objetos puramente software y, entonces, aquí sí que hay que representar las líneas temporales de esas instancias, así como el intercambio de mensajes (con sus argumentos) que se necesite para definir, con detalle y exactitud, todo el funcionamiento.

Un ejemplo de ello es el acceso al sistema de almacenamiento externo.

Las colecciones gestionadas a través de una clave o un identificador, único para cada elemento, lo obtienen desde el sistema externo de gestión de esos datos. Dicho sistema tiene la responsabilidad de mantener la consistencia de la organización de sus datos (normalización, claves primarias, etc.), responsabilidad que no se puede compartir en la aplicación. Sea como sea como se organicen allí los datos, los identificadores que se utilicen para cada uno, y en qué formato los entregue su fachada, es el adaptador especializado (*dentro* de la aplicación) el que los convierte en objetos y, ahí, les asigna un identificador único de objeto. Puede ser a partir de la clave primaria o por otro mecanismo, pero es su responsabilidad.

Con relación al comentario del 2º punto de las sugerencias para la organización de las colecciones, en la pregunta 3, al resolver el funcionamiento de cómo se da de alta un elemento, se plantea la necesidad de obtener su identificador no tanto para poder incorporarlo a la colección a la que pertenece (en su catálogo), sino para poder enlazarlo al elemento de la colección con la que está vinculado (‘presa’ con ‘cuenca’ y ‘sensorV’ con ‘presa’). Por consiguiente, al crear un nuevo elemento, habría que enviar una instancia *incompleta* de él al adaptador (sin el identificador), obtener el identificador al incorporar el nuevo elemento en el sistema externo y, entonces, añadirlo al catálogo correspondiente, además de incorporar ese identificador al listado del elemento en el catálogo vinculado.

¿Cuál de estos procedimientos se hace antes, el del sensor o el de la presa?

4.2. (1 punto) A partir de este DS, escriba y desarrolle los contratos de las operaciones ‘`AltaPresa`’ y ‘`AltaSensor`’. Recuerde que es imprescindible utilizar una sintaxis específica en estas descripciones.

Dicha sintaxis consiste en que, en todo momento, debe referirse a las instancias, su tipo (clase a la que pertenece) y su contenido (sus componentes o sus valores) que aparecen representadas en el diagrama DS de la pregunta 4.1.

Igualmente, los antecedentes representan las condiciones en las que estaba el sistema antes de que se realizara la operación; lo que se expresa, únicamente, mediante:

- Existencia de una instancia (su nombre y a qué clase pertenece).
- Existencia de una asociación (entre qué instancias o qué componentes).
- Valor concreto que tiene un elemento (identificar la instancia, o el componente, y su valor).

En cuanto a las postcondiciones, representan las repercusiones de haber realizado la operación, lo que se expresa:

- Creación (*se creó*) de una instancia (su nombre, a qué clase pertenece y cómo se inicializa, nombre de la instancia que la ha creado, etc.).
- Creación de una asociación (entre qué instancias o qué componentes –nombres o identificación—, cómo o en virtud de qué se ha creado esa asociación).
- Cambio del valor o el estado de un elemento (identificar la instancia, o el componente, la causa del cambio y el nuevo valor).

Sección 4. ***Diseño de las Colaboraciones***

5. (1 punto) A partir del contrato de la operación '*AltaPresa*' y de la correspondiente secuencia representada en el DS, que ha indicado en la pregunta 4, complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro que lo justifique. Si añade responsabilidades no explicitadas en el contrato (porque crea que es importante señalarlas), explíquelas brevemente.

Consiste en trasladar la secuencia de acciones incluidas en esa operación, representadas ya en el DS de la pregunta 4.1, con las mismas instancias y clases, al formato (horizontal y sin líneas de tiempo) de los diagramas de colaboración.

No olvide numerar los mensajes en el mismo orden temporal con el que se suceden en el DS de la pregunta 4.1.

6. (1 punto) A partir del contrato de la operación '*AltaSensor*' que haya indicado en la pregunta 4, complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro que lo justifique. Si añade responsabilidades no explicitadas en el contrato (porque crea que es importante señalarlas), explíquelas brevemente.

Sección 5. Evaluación del ***Diagrama de Clases*** de diseño

7. (1 punto) Elabore un diagrama de clases para el caso de uso que se está tratando *<<AgregarNuevaPresa>>* (DCD), centrado en la clase que va a implementar la responsabilidad más característica del caso de uso, la que mejor defina la naturaleza de lo que se hace en él (`Alta`). Represente los nombres de todos los atributos, asociaciones (con la navegabilidad) y métodos de esa clase (excepto '*setters*' y '*getters*' irrelevantes) y de las que estén directamente involucradas con ella en el caso de uso.

Sección 6. Evaluación de la ***Transformación del Diseño en Código***

8. (0'5 puntos) A partir de los anteriores diagramas de clases y colaboraciones, elabore y defina la clase que haya establecido, en el desarrollo anterior, como responsable de controlar la correcta secuencia de acciones en el caso de uso *<<AgregarNuevaPresa>>*. Incluya las definiciones de todas las variables que la componen (miembros), pero escriba solamente la definición completa del cuerpo para el método (o métodos) principal o más significativo: *<<se omite el método>>*. Ignore los pequeños detalles de sintaxis -el objetivo es evaluar la capacidad fundamental para transformar el diseño en código-. Utilice la sintaxis de Java.

   ATENCIÓN: lo que hay entre signos, *<<se omite el método>>*, es un ejemplo, usted debe sustituirlo por el nombre que haya asignado al método principal que haya elegido.