CURSO 2018 - 2019

SOLUCIONES 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:

DNI o número de expediente:

Contacto (teléfono o correo electrónico):

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 dominio del problema es una aplicación (EnDP) que supervisa y da soporte al **entrenamiento deportivo personal**.

La idea es desarrollar una aplicación *ligera*, para un dispositivo móvil, que permita al deportista organizar un Plan de Entrenamiento e integrarlo en su preparación.

Un Programa o *Plan de Entrenamiento* consiste en un número consecutivo de días, jornadas o *sesiones*, en cada una de las cuales se indica la realización de un conjunto de *actividades* o ejercicios; todo ello orientado a un tipo específico de deporte u objetivo (por ejemplo, "carrera de 21 km" o "rehabilitación de lesión en pierna").

Cada sesión se desarrolla en un único día y tiene una duración variable (aunque deberían oscilar poco entre sí), que depende de lo que dure cada uno de los ejercicios planificados para esa sesión, los cuales, también se realizan consecutivamente.

Los ejercicios de cualquier sesión consisten en actividades, de diversa índole, pero de aplicación u objetivo específico, con una duración e intensidad variable, que se secuencian y organizan, en la sesión, según unos criterios y reglas que debería establecer un supervisor o experto. Por ejemplo:

Sesión *n* Objetivo: Duración máx.:	Duración	Actividad / Intensidad
Actividad 1	5 minutos	de calentamiento /
Actividad 2	25 minutos	de carrera / a ritmo fácil
Actividad 3	5 a 10 minutos	de enfriamiento /
Actividad 4	5 minutos	de estiramiento /

En función del número de jornadas o sesiones del entrenamiento, puede haber alguna de ellas cuya única actividad sea "día de descanso".

Los criterios por los que el experto organiza las sesiones, y sus actividades, responden al tipo de deporte u objetivo que se persigue con el entrenamiento, a las características del usuario (parámetros de configuración: edad, complexión o masa corporal, nivel de exigencia para los objetivos, etc.) e incluso se pretende que puedan correlacionarse con alguna medida biométrica (ritmo cardiaco, tensión arterial, nivel de oxigenación en sangre, etc.) que se obtendría a través de un sistema externo a la aplicación.

Las principales funcionalidades previstas para la aplicación son:

- Configuración de la aplicación y configuración de los perfiles de usuario, para el uso diferenciado.
- Cada usuario puede tener una batería de *Planes de Entrenamiento*, por lo que puede crear o confeccionar un *Plan de Entrenamiento* nuevo y modificar cualquier parte, o borrar completamente uno antiguo.
- Un *Plan de Entrenamiento* se puede "activar", asignándole un día de inicio y un intervalo horario en el "*calendario – planificador de agenda*" (externo a la aplicación) y programando su ejecución en esas fechas. Se pretende que el *Plan de Entrenamiento activo* se sincronice con los avisos y alertas del planificador.
- Un *Plan de Entrenamiento activo* se puede "ejecutar", realizando, día tras día, los ejercicios que correspondan a cada sesión. También se desea que, durante los ejercicios, el sistema pueda recoger datos biométricos (a través de un sistema externo a la aplicación) con los que el "módulo experto de planificación" (el que supervisa al usuario al crear un nuevo *Plan de Entrenamiento* y al organizar las actividades en él) podría ajustar (modificar) el *Plan de Entrenamiento* y las actividades que contiene.

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 comandos). Igualmente, la interacción entre la lógica del negocio y los sistemas de apoyo externos se realizará a través de adaptadores e interfaces. De esta forma, dado el escaso volumen de información que maneja la aplicación (en un dispositivo móvil), la referida a los ejercicios, y a las reglas para organizarlos en un plan, puede almacenarse en un repositorio que, junto con la información de las medidas biométricas (obtenidas mediante los periféricos adecuados), es manejada por el "módulo experto de planificación" a través, también, de las interfaces adecuadas.
- La extensión de un Plan de Entrenamiento es de 10 días consecutivos (10 sesiones). En esos 10 días, no debe haber más de 2 sesiones o jornadas de descanso.

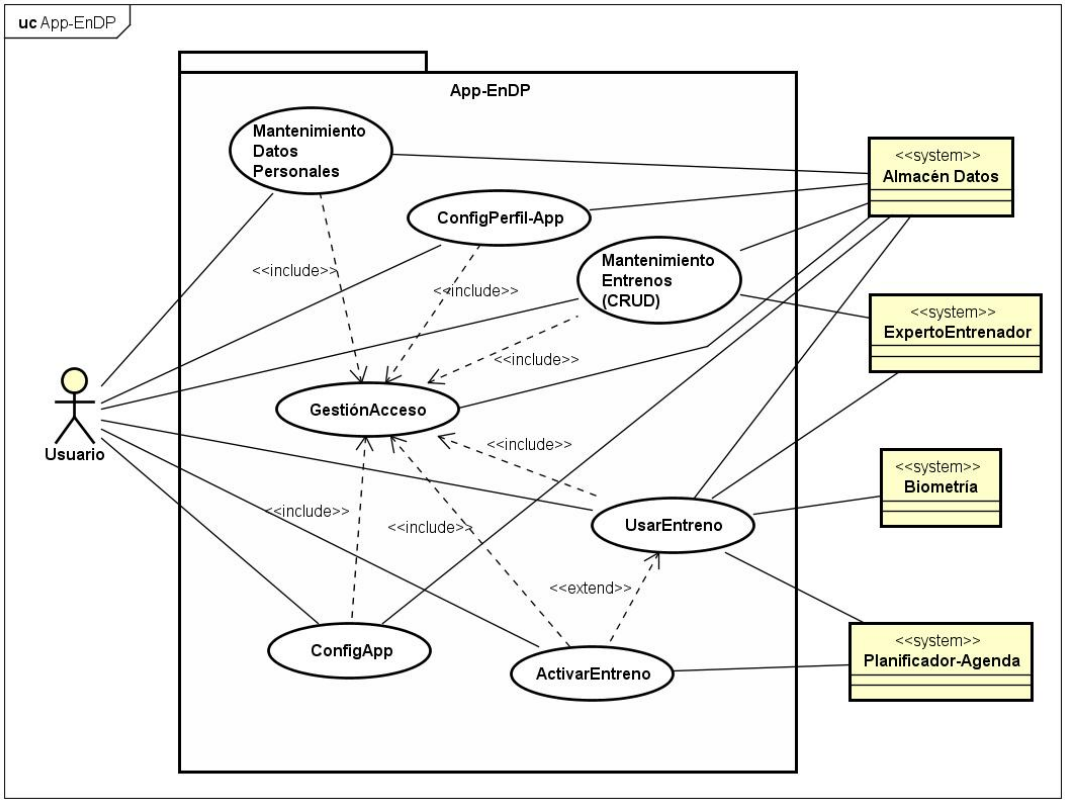
Material recomendado para la realización de esta prueba:

- Libro de texto de la asignatura.
- Documento 'Recomendaciones para las Evaluaciones de la Asignatura'.
- Documentación adicional al respecto, que se publicará en el curso virtual y se avisará a través de los foros.
- Soluciones propuestas para las PEC de los cursos 2016, 2017 y otras que se publiquen posteriormente.

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 ***Casos de Uso***

1. (0'5 puntos) En relación con la aplicación EnDP, represente los casos de uso primarios más importantes, sus actores principales y de apoyo y las interacciones correspondientes en un diagrama UML de casos de uso.



Desde el primer momento hay que asumir que toda la información y los datos que se necesitan para el funcionamiento de una aplicación están almacenados en algún sitio (Almacén Datos), disponibles para ella, pero no integrados en su funcionamiento. A través de adaptadores (objetos software) la aplicación tomará de ahí cualquier dato que necesite y dicho adaptador lo convertirá en la estructura de datos que se requiera para su manejo. Así mismo, los sistemas de apoyo pueden necesitar información que se está manejando en la aplicación para, con sus propios datos, proporcionar el servicio o la información que requiere la aplicación. Esa interacción también se hace mediante adaptadores. Por último, también hay que tener presente que la información que aporta el usuario no tiene ninguna significación directa *dentro* del funcionamiento de la aplicación, no son datos (objetos) utilizables por ella. Suelen ser textos, valores o códigos que deben ser interpretados y transformados (otra vez mediante los adaptadores) para obtener los datos o construir las estructuras (objetos) que sí se manejan en el funcionamiento de la aplicación.

Aunque el control del acceso es algo fundamental en aplicaciones multi-usuario (o con distintos roles de uso), dicho acceso no se corresponde nunca con un objetivo principal de uso (EBP). Sin embargo, es una operación incluida en todos los casos de uso primarios y, en el nivel de la aplicación y de la lógica del flujo de trabajo, es determinante para el funcionamiento de cada uno (pero **no es un caso de uso primario**).

En el examen, no es obligatorio representar las relaciones «include» o «extend» entre los casos de uso. Pero, en el caso de no hacerlo, todos los casos de uso representados deben ser y se consideran primarios.

2. (1 punto) Escriba el caso de uso *<<AgregarPlan>>* 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. Supóngase que el flujo básico al que se refiere la pregunta tiene como antecedentes que la aplicación ya está configurada, el usuario (común) ha creado su perfil y ya está configurado para su uso. Además, dicho usuario ya ha entrado en su sesión y ha solicitado la operación de mantenimiento de sus entrenamientos. Este flujo de acciones de éxito discurre desde que el usuario inicia la creación de un nuevo Plan de Entrenamiento, para un deporte u objetivo determinado, hasta que el Plan queda almacenado en su perfil.

Supóngase que el deporte u objetivo de este caso de uso es "carrera urbana de 10 km", que el entrenamiento es de 10 días (sesiones) y que, a la hora de organizar los ejercicios en cada sesión, **el "módulo experto de planificación" no actúa**: es el usuario el que utiliza su experiencia para seleccionar los ejercicios y su intensidad y para determinar su orden y su duración.

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.

Caso de uso: **AgregarPlan**

Formato completo (variante 'a dos columnas'), estilo esencial.

Evolución típica de los acontecimientos

Acciones del actor (el Usuario de la aplicación)	Respuesta del sistema
1. El caso de uso comienza cuando el usuario solicita la creación de un nuevo programa de entrenamiento.	2. El sistema solicita el nombre del programa (plan), el objetivo del plan, la disponibilidad temporal en cada sesión y si se requiere o no la intervención del experto.
3. El usuario aporta la información solicitada.	4. El sistema solicita la situación (orden) de una jornada.
5. El usuario añade una sesión al plan de entrenamiento.	6. El sistema solicita, por orden, un ejercicio o actividad de esa jornada.
7. El usuario selecciona la actividad e indica su intensidad y duración.	8. El sistema presenta los contenidos del ejercicio y solicita confirmación.
9. El usuario confirma la actividad. Se repiten los pasos 6 a 9 hasta finalizar la jornada.	10. El sistema presenta los contenidos de la sesión y solicita confirmación.
11. El usuario confirma la sesión. Se repiten los pasos 4 a 11 hasta que el usuario finalice todas las sesiones del programa de entrenamiento.	12. El sistema comprueba la validez y coherencia temporal de las actividades en las sesiones del plan. 13. El sistema agrega el plan a la colección de entrenamientos del perfil del usuario, presenta un cuadro con la distribución de actividades en el plan y solicita su confirmación.
14. El usuario confirma el contenido del plan de entrenamiento.	15. El sistema termina la creación de un plan de entrenamiento y retorna al nivel lógico inmediato.

Alternativas

- 4 a 12 En 3) el usuario solicita la intervención del experto, en cuyo caso el programa de entrenamiento lo confecciona él y se continúa en 13.
- 9 El usuario no confirma el contenido de la actividad porque desea hacer modificaciones, en cuyo caso pasa a 6.
- 11 El usuario no confirma el contenido de la sesión porque desea hacer modificaciones, en cuyo caso pasa a 4 y debe agregar todos sus ejercicios.
- 14 El usuario no confirma el contenido del plan porque desea hacer modificaciones, en cuyo caso, en 15, se pasa a la modificación de ese programa de entrenamiento.

Este podría ser un planteamiento inicial de la lógica del funcionamiento, de la interacción entre el usuario y el sistema en la construcción de un nuevo plan de entrenamiento. Habría que refinarla para facilitar las alternativas de '*vuelta atrás*' en la modificación de alguna de las partes del plan.

Como se ve, la descripción se refiere a los elementos que se manejan en la interacción en un lenguaje propio del uso en la actividad que se está representando, *como si no hubiera máquinas*. No se hace mención de ningún objeto, conceptual o software, ni a ningún detalle técnico sobre la obtención de la información o cómo se transforma.

Es indudable que hay innumerables formas de realizar una tarea. O, dicho de otra forma, se pueden concebir muchas maneras del funcionamiento de un sistema para obtener resultados semejantes. Sin embargo, el diseño es la definición del funcionamiento de un sistema y, una vez definido éste, el resultado es, básicamente, único.

En estos ejercicios se exige coherencia y continuidad entre las respuestas de las preguntas. En ésta, en concreto, se describe la lógica del funcionamiento de la operación desde el punto de vista del uso cotidiano, sin la intervención, necesariamente, de la máquina. De aquí a la definición de cómo hace la máquina para realizar esa tarea, aún queda un largo recorrido. A pesar de haber tantos grados de libertar, en algunos casos sí se puede apreciar un defecto en la lógica que, si no se rectifica, puede imposibilitar la obtención del resultado pedido.

Aunque no se pide en el ejercicio, y tradicionalmente se recomienda que se realice justo antes del diseño, es probable que, para refinar la lógica del caso de uso, resulte de gran utilidad confeccionar un Diagrama de Secuencia del Sistema, como se explica en el Anexo I.

¡ATENCIÓN!: de aquí en adelante, **todas las preguntas se refieren al escenario y las especificaciones, definidas en esta pregunta, para este caso de uso**.

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

3. (2 puntos) En relación con el caso de uso anterior, <<*AgregarPlan*>>, 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.

Para la respuesta a esta pregunta se comienza a pensar en el lado del sistema, en cómo va a implementar la lógica del funcionamiento descrita en la pregunta anterior, en qué elementos se están manejando (objetos conceptuales), qué información deben contener para que sea posible dicho funcionamiento y qué otros objetos (quizás no mencionados en la pregunta 2, porque no interactúan directamente con el actor) son necesarios para el comportamiento (lógico) descrito; siempre bajo el axioma de que un objeto sólo puede manejar su propio contenido. En definitiva, es otra forma de describir la lógica del caso del uso, ante los estímulos del actor, pero desde el punto de vista de los objetos

del sistema, de su rol y su funcionalidad (aunque nunca se ponen los métodos en este diagrama, sólo sus atributos (lógicos, no técnicos o software, sin *tipo*), sus relaciones, generalmente de contenido, y su cardinalidad). Tampoco deberían ponerse objetos software que, estrictamente, serán soluciones técnicas, usadas posteriormente en el diseño, para la implementación de ese comportamiento.

Por ello, la recomendación es comenzar la construcción por el objeto representante del objetivo del caso de uso: el Plan de entrenamiento.

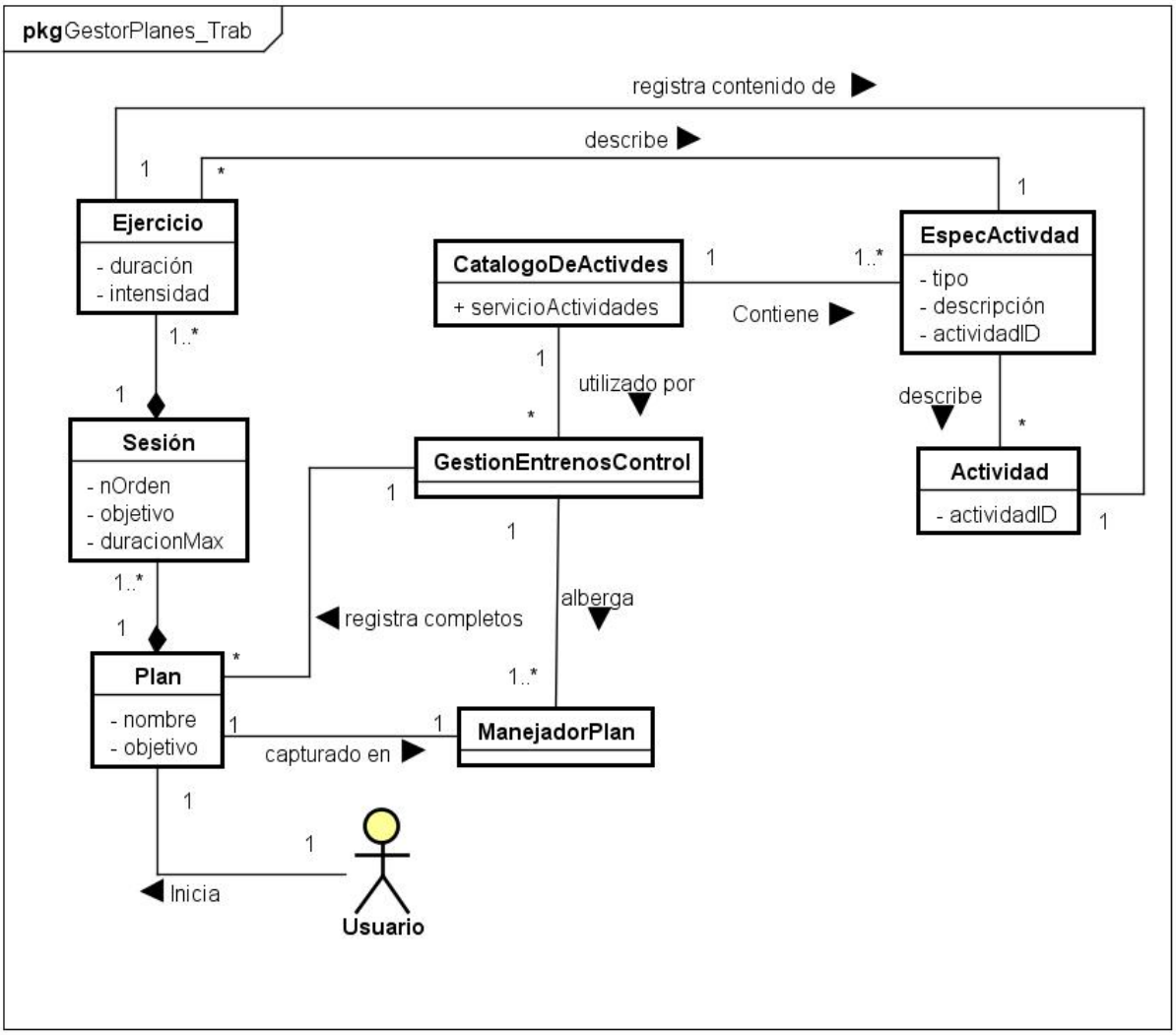


Figura 1 Modelo de Dominio. *Suficiente* para examen.

Así, a partir de la descripción de un Plan, objetivo del usuario, se puede representar la colección de jornadas, o Sesiones, que contiene y la colección de Actividades, o Ejercicios, que tiene cada Sesión. Los atributos de cada uno de estos objetos son la información *variable* que aportará el actor al instanciarlos específicamente.

A excepción de la mencionada información *variable* del Ejercicio, su contenido está formado por las características de una Actividad. En este punto, por la utilidad que se deduce de la aplicación EnDP en sí, parece lógico pensar que el objeto Actividad debería tener alguna estandarización dentro del programa, incluso sin la intervención del ExpertoEntrenador. Para poder manejar las actividades de una forma *regular*, homogénea incluso con las que pudiera aportar el ExpertoEntrenador y sin pretender que *se las invente* el actor al usarlas, todo parece apuntar a que deberían estar *tabuladas* en el programa. Es decir, como opciones en una colección sobre la que se pudiera seleccionar la más adecuada: un CatalogoDeActivdes, alojado en EnDP o *equivalente* (GestionEntrenosControl).

Queda un penúltimo elemento, el director de la orquesta, el que va a ejercer el rol de control en el funcionamiento del caso de uso: el controlador al que denominamos ManejadorPlan.

Este diagrama es compatible con el enunciado, con el objetivo del Modelo de Dominio y con la descripción de la respuesta a la pregunta 2, por lo que **sería suficiente para un examen**. Sin embargo, deja sin explicación una serie de cuestiones que es necesario tener en cuenta porque fundamentan decisiones importantes para las siguientes preguntas:

- Cómo se incorpora el nuevo plan de entrenamiento a la colección de planes del usuario.
- Qué estructura se maneja para esa colección de planes y cómo se relaciona con el perfil del usuario, qué estructura tiene éste y cuáles son los antecedentes en el funcionamiento de todo esto.

Por consiguiente, lo que se desarrolla en el Anexo II, ni se debe poner en el examen, ni hay posibilidad temporal para ello; aunque sí se debería tener muy presente, al menos, la lógica que se describe, tanto para perfilar el análisis como para construir el diseño en las siguientes preguntas.

Como consecuencia de esos razonamientos, queda justificado el papel que cumple el controlador de nivel superior, GestionEntrenosControl, y algunas pequeñas modificaciones, en el siguiente diagrama del Modelo de Dominio, cuya necesidad se descubre, también, al implementar posteriormente el diseño:

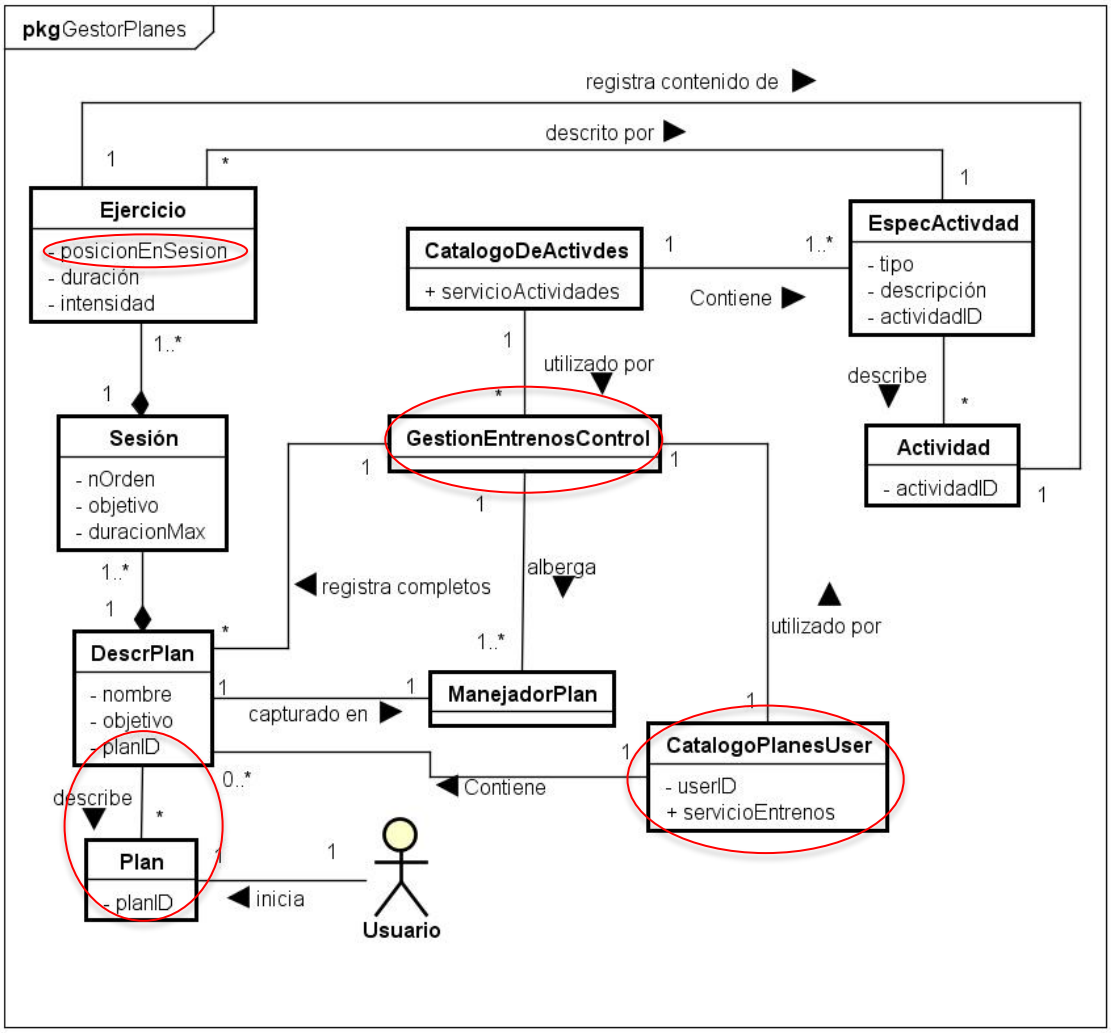


Figura 2 Modelo de Dominio refinado.

Aunque aún no se ha hecho nada de diseño, sí se debe haber pensado en *la logística* de la aplicación, en cómo se conjuga con ella el funcionamiento del caso de uso y cómo se implementaría esa coordinación con software. Las conclusiones allanan notablemente el camino hacia el diseño:

1. Se ha definido una lógica para el funcionamiento del caso de uso, acorde con la descripción del enunciado y desde la perspectiva de su uso.
2. Se ha identificado qué información o datos se manejan en el caso de uso, cuáles aporta el usuario para dar especificidad a qué elementos que significan el objetivo en la manipulación del caso de uso. Se ha decidido cómo agrupar esos elementos de información y qué estructura darles (objetos conceptuales) para que su manejo sea posible: la estructura del Plan-DescrPlan y sus componentes, el contenedor para la búsqueda y selección de las Actividades (CatalogoActivdes) y el objeto con el rol de control para realizar las operaciones (ManejadorPlan).

3. Atendiendo al funcionamiento global de la aplicación y al uso que se le puede dar, fuera del caso de uso, a alguno de los anteriores objetos (el CatalogoActivdes y el mismo Plan, al incorporarse a la colección de entrenamientos del usuario, CatalogoPlanesUser, y, seguramente, a su perfil), se ha ideado y propuesto una secuencia de ese funcionamiento global en la que se puede articular, de forma compatible con él, el funcionamiento del caso de uso: el CatalogoPlanesUser, contenido en el perfil de usuario (DescrPerfilUsr), se ha tenido que crear en niveles de ejecución anteriores al caso de uso y se maneja desde ahí y desde entonces. Lo mismo ocurre con el CatalogoActivdes, que debe manejarse desde el momento en el que se vayan a manipular los entrenamientos, Planes, o sus componentes. Vamos a denominar GestionEntrenosControl al controlador de nivel superior, inmediatamente anterior al caso de uso, que recoge qué opción CRUD se selecciona y le transfiere ambos catálogos (el equivalente a Tienda, en el caso de uso Ventas, de PdV).

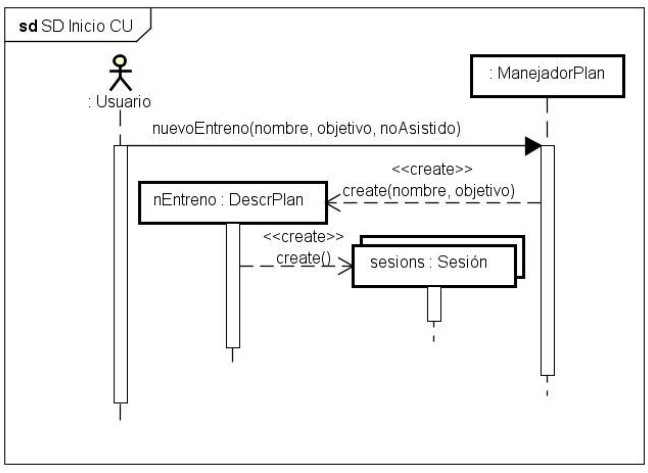
Sección 3. Evaluación de los ***Eventos del Caso de Uso***

4. ***Eventos y Contratos***.

4.1. (2 puntos) Circunscrito al caso de uso anterior *<<AgregarPlan>>*, construya un Diagrama de Secuencia (diagrama de interacción DS) en UML. Represente los actores y los eventos de los componentes del sistema para este caso de uso.

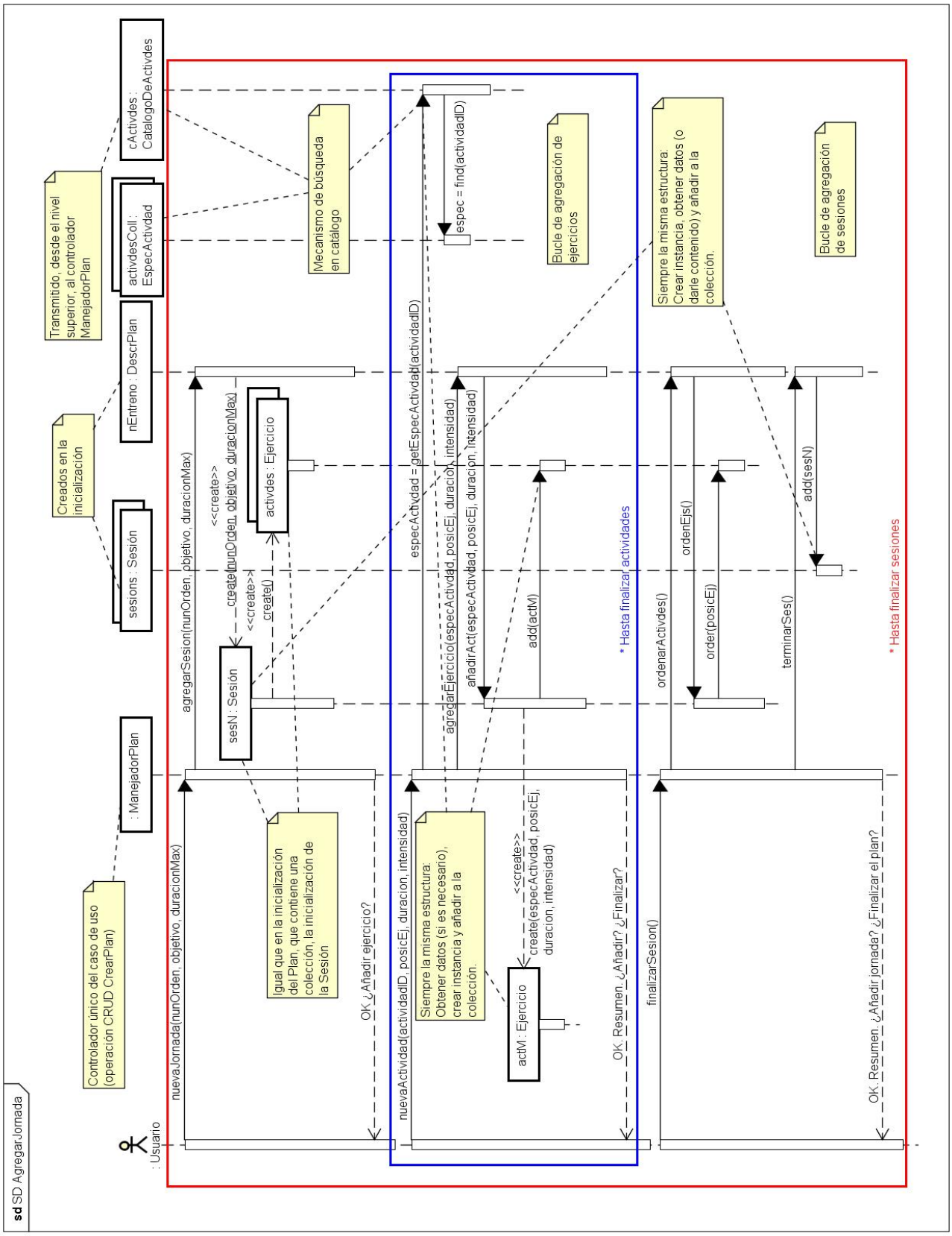
A la vista del análisis realizado, de las conclusiones indicadas inmediatamente antes (se insiste en su importancia) y del planteamiento que se desprende de un Diagrama de Secuencia del Sistema como el del Anexo I (si no se ha hecho antes, sería muy útil hacerlo ahora, aunque sea un esquema muy sencillo), se va a dividir la secuencia en tres partes: Inicialización, el bucle de Agregar Sesiones (que contiene otro, el de Añadir Ejercicios) y la Finalización (para agregar el nuevo Plan a la colección del usuario).

El DS detallado de la Inicialización sería el siguiente:



Se crea una instancia con la estructura del nuevo Plan, uno de cuyos componentes es una colección de Sesiones, que también se instancia, vacía.

La operación más tupida, aunque contiene acciones sencillas, es la del doble bucle anidado para formar los componentes del Plan, la agregación de Sesiones:



El bucle de Sesión (rojo) que, al igual que Plan, también contiene una colección, comienza de la misma forma: inicializando la instancia sesionN y creando la colección vacía activdes.

A continuación, arranca el bucle de Ejercicio (azul) que, de forma análoga al bucle externo, debería inicializar una instancia de Ejercicio, actM, y añadirla a la colección de activdes. La diferencia es que el Ejercicio, además de concretarse con los valores que aporta el usuario, está descrito mediante la EspecActivdad de la Actividad que haya seleccionado el usuario. En un examen, con poco tiempo, se podría abreviar este bucle interno con la inicialización de actM y su agregación a activdes; si no fuera por la variante que obliga a buscar en el catálogo cActivdes la Actividad seleccionada antes de inicializar actM. En el examen es *muy conveniente* que se refleje el conocimiento del mecanismo de la búsqueda en catálogo y, a la vez, se justifique por qué se ha incluido el objeto CatalogoDeActivdes en el Modelo de Dominio.

Antes de seguir, es muy importante resaltar que, aunque todos los estímulos u órdenes del usuario se reciben en el controlador, las maniobras del bucle rojo se hacen en, y desde, clases distintas a las del bucle interno. **Cada clase maneja sólo su contenido**. En el bucle externo, la agregación de Sesiones (componente de Plan) se hace desde el Experto en ese componente: el Plan (o, más exactamente, DescrPlan). De la misma manera, en el bucle interno, la orden de agregación de Ejercicios (componente de Sesión) llega al Plan, se transfiere a la instancia de la Sesión *activa*, sesN, y se hace desde el Experto en ese componente: la Sesión sesN.

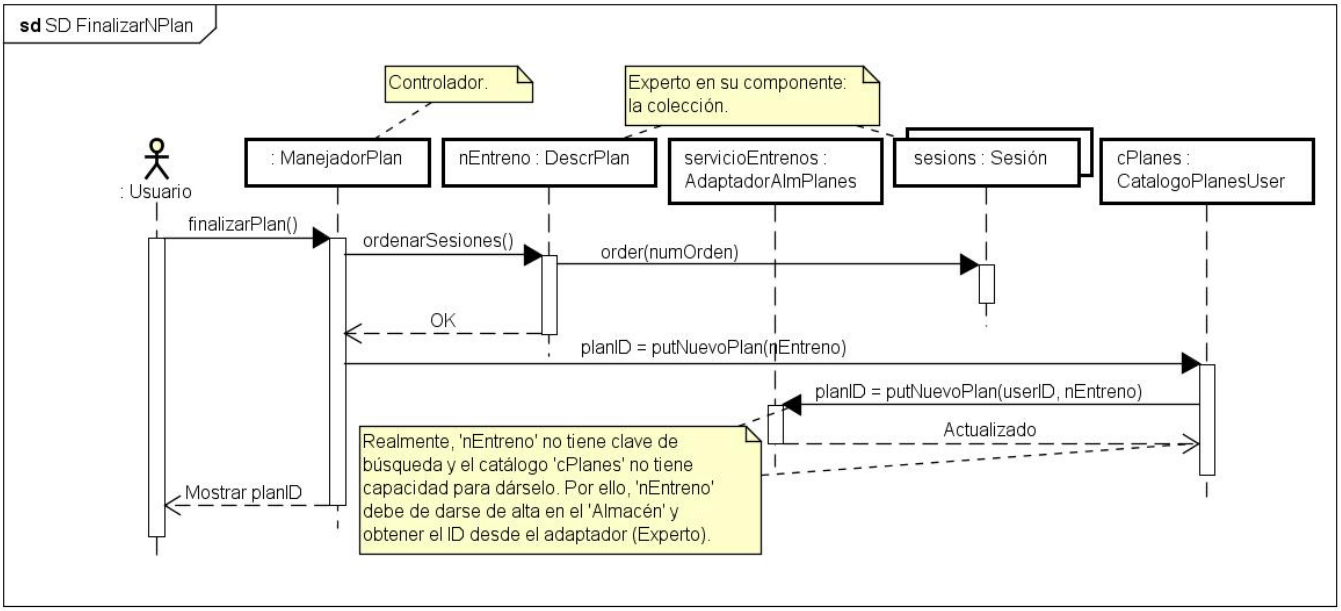
El *cierre* del bucle interno lo realiza el usuario, al indicar que ha finalizado los Ejercicios de esa Sesión; en cuyo caso sesN (ya completa) se agrega a la colección sesions. De igual forma, el bucle externo se termina cuando el usuario, tras finalizar una Sesión, indica que no añade más; en cuyo caso se acaba la formación del Plan.

Al margen de las respuestas que cabría esperar en un examen, se ha pensado (también en el análisis) que la utilidad de los elementos que se están usando (Ejercicios agrupados en Sesiones) depende, en gran medida, de su ordenación temporal. Por este motivo se ha añadido el atributo posicEj en el objeto Ejercicio del Modelo de Dominio, indicando la necesidad de que el usuario (o el ExpertoEntrenador) determine el orden de los Ejercicios dentro de una Sesión (y puesto que el enunciado ya establecía una característica equivalente para la Sesión: numOrden).

Por otro lado, se ha establecido que toda esta información, siempre que se contemple su existencia, reside en el Almacén, en un formato desconocido para la aplicación. En cualquier caso, si la aplicación necesita obtener la información sobre un Entrenamiento o un Plan, le corresponde al adaptador tener implementada la consulta, de forma que la información le llegue ordenada, o bien contener algún procedimiento de forma que, tras recibir la información, la ordene.

Por ello, de manera ilustrativa, se ha incluido la ordenación temporal de los Ejercicios en una Sesión, que se repite para las Sesiones de un Plan, ante la posibilidad de que el nuevo Plan creado pueda ser utilizado directamente, sin incorporarse a la colección y su catálogo y, por consiguiente, sin ser procesado por el adaptador correspondiente ni actualizado en el Almacén.

Por último, el DS detallado de la finalización:



Lo más obvio de este diagrama es que, una vez completado el Plan, el controlador lo añade al catálogo de entrenamientos del usuario, cPlanes.

Sin embargo, lo singular es que el nuevo Plan creado no tiene una estructura que pueda formar parte de un catálogo: en todo momento se ha estado manejando la construcción de los componentes de la descripción de un Plan, pero no hay una clave de búsqueda, por lo que no se puede incorporar a un HashMap. De hecho, el catálogo tampoco tiene capacidad (no es Experto en…) para asignarle una clave como planID. Debería ser el Almacén, o su gestión, la única entidad capaz de determinar cuál puede ser la clave de búsqueda y recuperación de la información que almacena. En los capítulos finales del libro, se exponen algunos procedimientos para extraer un código de identificación de un objeto. Pero, en este caso, el adaptador del catálogo con el Almacén, servicioEntrenos, necesita conocer el significado de ese código para traducirlo a la semántica que el Almacén utiliza para la información que gestiona. Por estos motivos, la adición de nEntreno al catálogo no puede ser un simple put(…) que lo añada al HashMap. La función debe contener un código que, a través del adaptador con el Almacén, incluya la nueva información en él (persistencia) y, de paso, obtenga la clave asociada a esa información, planID, con lo que el catálogo cPlanesUser ya puede añadirlo a su colección (no persistente).

4.2. (1 punto) A partir de este DS, escriba y desarrolle los contratos de **2** operaciones principales: (llamémoslas, aquí, '*OperacionA*' y '*OperacionB*'. Usted puede llamarlas como le convenga; pero, en adelante, debe mantener esa denominación). Estas operaciones deben ser principales, consecutivas (en la medida de lo posible) y cubrir todo o la mayor parte del caso de uso. De otra forma **no se calificarán, ni en esta pregunta ni en las siguientes**.

Contrato CO1: agregarSesion

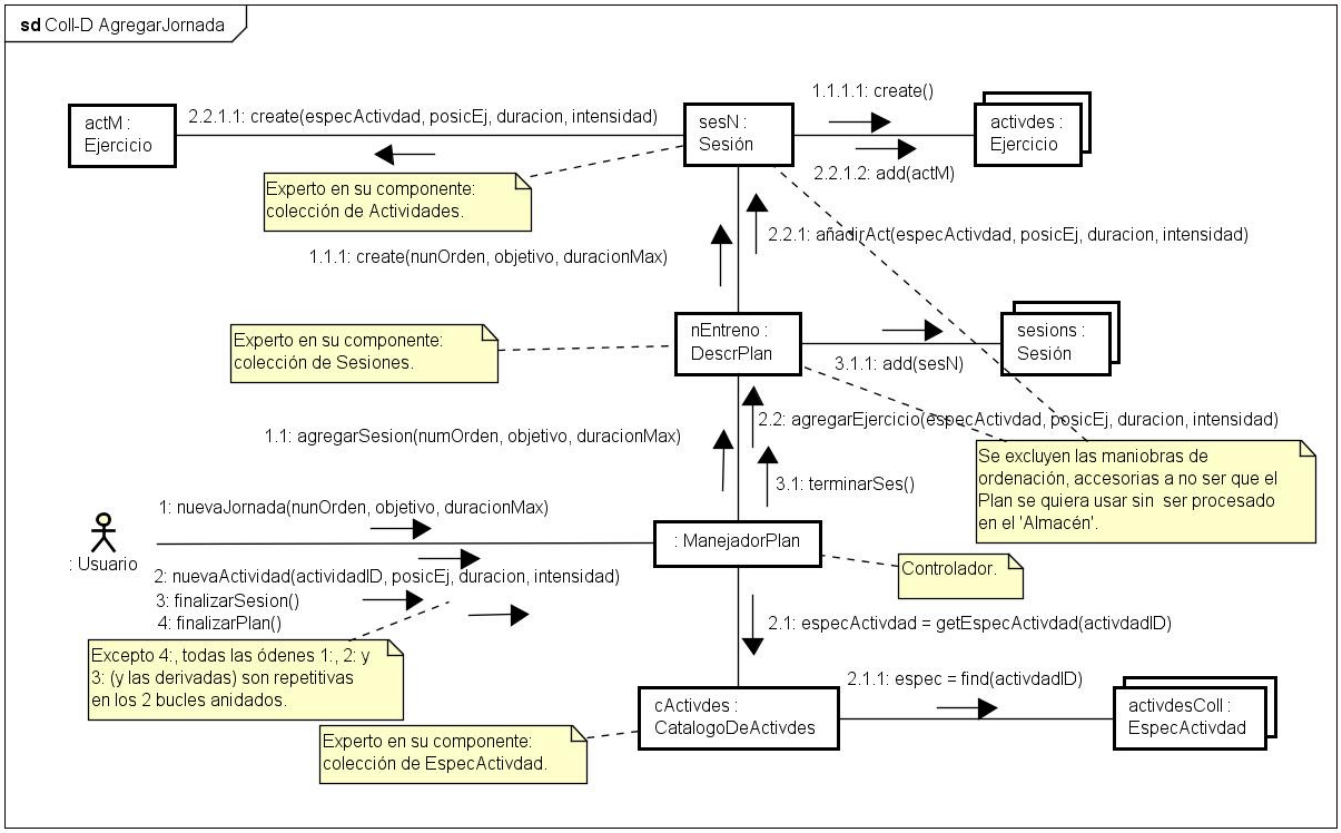
Operación:	* nuevaJornada (nunOrden, objetivo, duracionMax), que incluye * nuevaActividad (actividadID, posicEj, duracion, intensidad)
Referencias cruzadas:	Caso de Uso: AgregarPlan.
Precondiciones:	– Hay una sesión activa, con un usuario autorizado, en la que se ha llegado a la opción CRUD para el mantenimiento de los elementos de la colección del CatalogoPlanesUser, la instancia **cPlanes**. Además, en dicha situación hay una instancia de CatalogoDeActivdes, **cActivdes**. – Hay una instancia activa, **nEntreno**, de DescrPlan, inicializada y asociada a **ManejadorPlan**. – Hay una colección vacía (multiobjeto) de Sesion, **sesions**, asociada a **nEntreno**.
Postcondiciones:	– *(1) Se creó e inicializó una instancia de Sesion **sesN** y se asoció con **nEntreno**. – Se creó una colección vacía (multiobjeto) de Ejercicio, **activdes**, y se asoció con **sesN**. – *(2) Se creó e inicializó una instancia de Ejercicio, **actM**. – El atributo de **actM**, **actividad**, se inicializó con **especActivdad**, de DescrActivdad, en virtud de la coincidencia de 'actividadID' (aportado por el usuario) en el CatalogoDeActivdes, **cActivdes**. – **actM** se añadió a la colección **activdes**. * Se repitió desde (2) hasta completar **activdes**. – **sesN** se añadió a la colección **sesions**. * Se repitió desde (1) hasta completar la colección **sesions**.

Contrato CO2: finalizarPlan

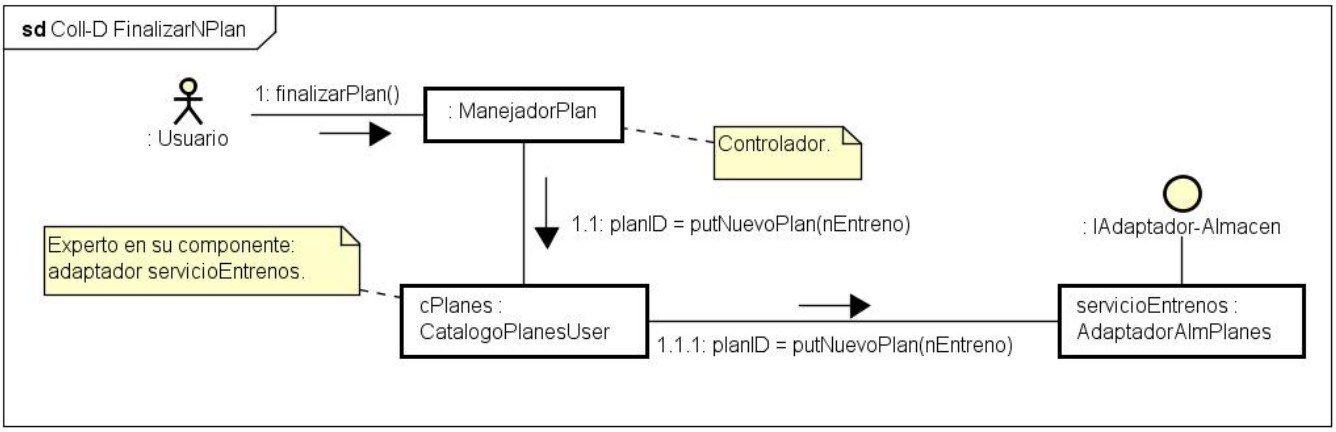
Operación:	finalizarPlan ()
Referencias cruzadas:	Caso de Uso: AgregarPlan.
Precondiciones:	– En la sesión activa, **ManejadorPlan**, hay una instancia de CatalogoPlanesUser, **cPlanes**, y contiene una colección de DescrPan, **planesUsrColl**. – Hay una instancia activa de DescrPlan, **nEntreno**, que contiene una colección (multiobjeto) de Sesion, **sesions**, que, a su vez, contiene otra colección (multiobjeto) de Ejercicio, **activdes**. – La identidad del usuario **usuarioID**, está asociada a la sesión activa, **ManejadorPlan**, y a **cPlanes**. – Hay una instancia de AdaptadorAlmPlanes, **servicioEntrenos**, y asociado a **cPlanes**.
Postcondiciones:	– El atributo de **nEntreno**, **idPlan**, se inicializó con el valor aportado por **servicioEntrenos**, **planID**. – **nEntreno** se añadió a la colección **planesUsrColl**, quedando asociado a **cPlanes**.

Sección 4. Evaluación de la ***Asignación de Responsabilidades*** y ***Diseño de Colaboraciones***

5. (1 punto) A partir del contrato de la operación <<*OperaciónA*>> que haya indicado en la pregunta 4 (como la haya llamado usted), 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.



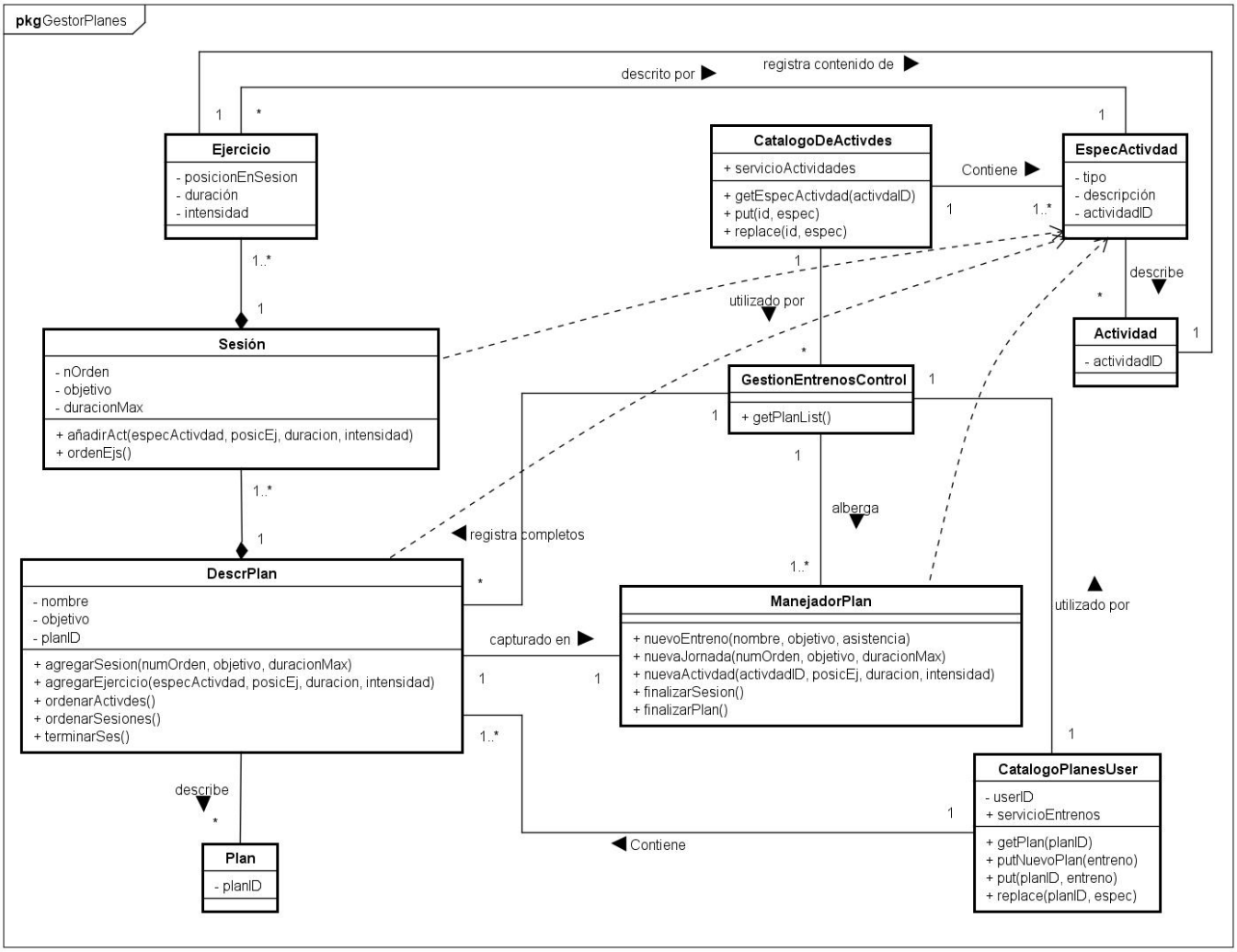
6. (1 punto) A partir del contrato de la operación <<*OperacionB*>> que haya indicado en la pregunta 4 (con la denominación que haya utilizado allí para esa operación), complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro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



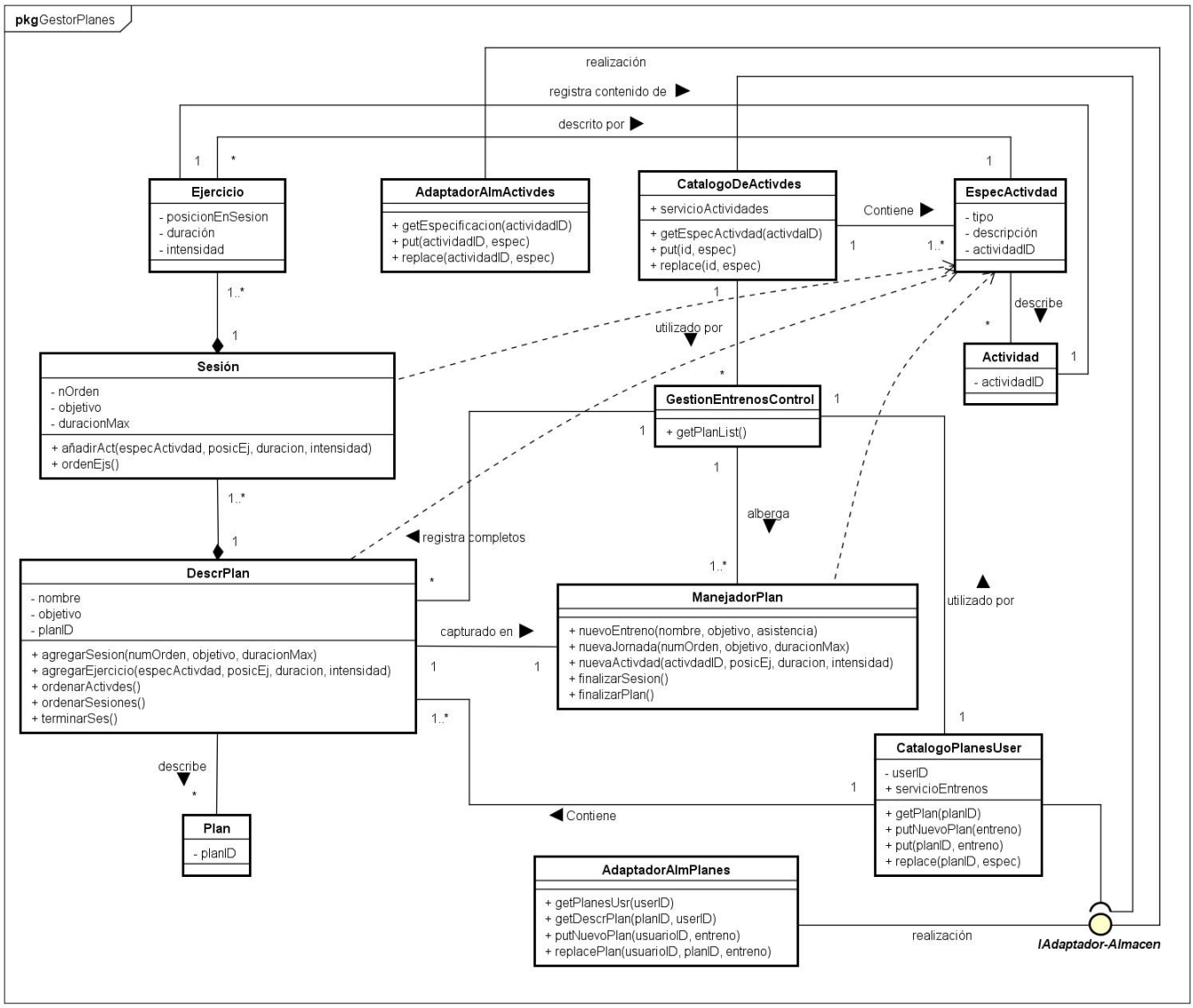
Sección 5. Evaluación de los ***Diagramas de Clases*** de diseño

7. (1 punto) Elabore un diagrama de clases para el caso de uso que se está tratando <<*AgregarPlan*>> (DCD), centrado en la clase que va a implementar la responsabilidad más característica del caso de uso, la que mejor define la naturaleza de lo que se hace en él (PlanEntrenamiento). 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.

   Los hallazgos obtenidos, complementan el modelo de dominio en el siguiente DCD:



Sin embargo, en el DCD, sí se pueden reflejar las clases estrictamente software que implementan el comportamiento:


pkg GestorPlanes
realización
registra contenido de
descrito por
Ejercicio
- posicionEnSesion
- duración
- intensidad
AdaptadorAlmActivdes
+ getEspecificacion(actividadID)
+ put(actividadID, espec)
+ replace(actividadID, espec)
CatalogoDeActivdes
+ servicioActividades
+ getEspecActivdad(activdalD)
+ put(id, espec)
+ replace(id, espec)
Contiene
EspecActivdad
- tipo
- descripción
- actividadID
describe
Actividad
- actividadID
utilizado por
Sesión
- nOrden
- objetivo
- duracionMax
+ añadirAct(especActivdad, posicEj, duracion, intensidad)
+ ordenEjs()
GestionEntrenosControl
+ getPlanList()
alberga
registra completos
DescrPlan
- nombre
- objetivo
- planID
+ agregarSesion(numOrden, objetivo, duracionMax)
+ agregarEjercicio(especActivdad, posicEj, duracion, intensidad)
+ ordenarActivdes()
+ ordenarSesiones()
+ terminarSes()
capturado en
ManejadorPlan
+ nuevoEntreno(nombre, objetivo, asistencia)
+ nuevaJornada(numOrden, objetivo, duracionMax)
+ nuevaActivdad(actividadID, posicEj, duracion, intensidad)
+ finalizarSesion()
+ finalizarPlan()
utilizado por
describe
Plan
- planID
Contiene
CatalogoPlanesUser
- userID
+ servicioEntrenos
+ getPlan(planID)
+ putNuevoPlan(entreno)
+ put(planID, entreno)
+ replace(planID, espec)
AdaptadorAlmPlanes
+ getPlanesUsr(userID)
+ getDescrPlan(planID, userID)
+ putNuevoPlan(usuarioID, entreno)
+ replacePlan(usuarioID, planID, entreno)
realización
IAdaptador-Almacen

PEC 2019. ANEXO I

DIAGRAMA DE SECUENCIA DEL SISTEMA.

El diagrama de secuencia del sistema representa la interacción entre el actor principal del caso de uso y un único objeto, sustituto del sistema software que implementa su comportamiento.

Es un diagrama que no se pide en el examen. Pero, por ser una antesala del diseño detallado, en la que ese único objeto se descompone en los distintos objetos que colaboran entre sí para definir, con exactitud, cómo es el funcionamiento y en la que los estímulos del actor se diversifican entre esos objetos para describir cómo es la colaboración, se recomienda su elaboración justo antes de realizar el Diagrama de Secuencia Detallado, de escribir los contratos de las operaciones principales o de realizar los Diagramas de Colaboración de esas operaciones.

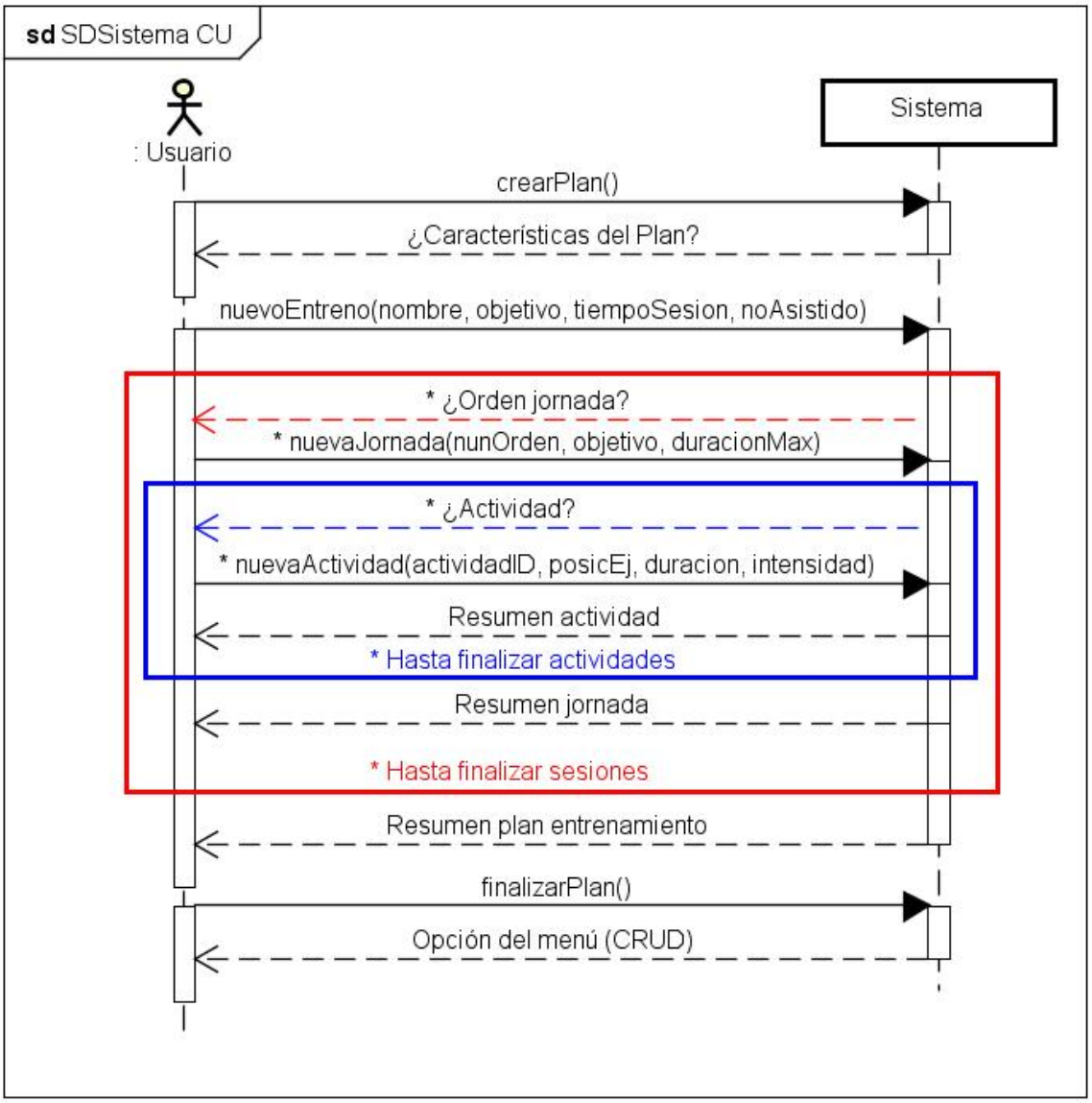
Esta recomendación se justifica porque permite tomar una perspectiva general sobre el sistema software y la interacción con el actor principal y decidir qué operaciones (o agrupación secuenciada de ellas) son principales y cuáles no.

Sin embargo, justificado por esa perspectiva general y por la representación de esa interacción estímulo—respuesta, entre el actor y el sistema, es muy probable que también resulte muy útil para refinar y afianzar la lógica del funcionamiento, desde la perspectiva de su uso, del caso de uso; descrita en la pregunta 2.

Por descontado, hay que tener en cuenta que, si se hace en este punto del análisis (inmediatamente después de la escritura del caso de uso e inmediatamente antes del Modelo de Dominio), aún no se conocen, con exactitud, los nombres de las funciones que sirven de estímulo al sistema ni, mucho menos, los argumentos (y sus tipos) que se van a utilizar en ellas. Sin embargo, además de la utilidad mencionada para refinar la lógica del caso de uso, puede resultar un interesante ejercicio de *avance* y previsión.

Sea cual sea el momento en que se haga, y para qué parte se le encuentre más utilidad, la recomendación es que se utilice como ayuda; aunque sea un boceto apresurado y no se entregue en el examen.

A partir del esquema conceptual de la lógica que se ha propuesto para este caso de uso, el Diagrama de Secuencia del Sistema podría ser el siguiente:



PEC 2019. ANEXO II

ANTECEDENTES DEL CASO DE USO. SITUACIÓN Y ORGANIZACIÓN DE ALGUNOS ELEMENTOS SOFTWARE, PREVIOS AL CU Y NECESARIOS PARA LA TOMA DE DECISIONES EN SU DISEÑO.

La piedra angular en el conocimiento de esta asignatura es mantener un exquisito cuidado, a la hora de construir el funcionamiento e implementación software de alguna operación, para asignar, *a cada cuál, lo suyo* (principios de Experto y Bajo Acoplamiento, principalmente).

Dejando a un lado la perspectiva de la presentación (IU) y la de los servicios técnicos (acceso a datos, etc.), para alcanzar la independencia funcional, y que sea útil, en el comportamiento de cualquier aplicación, no basta con realizar una descomposición en los elementos independientes que intervienen (desde la perspectiva de la lógica del negocio, en la única en la que nos centramos) en cada operación concreta; sino que esos elementos, y todos los demás, también deben organizarse en agrupaciones (módulos, paquetes o como se quieran denominar) funcionalmente independientes: el diseño arquitectónico.

Sea como sea la organización de los módulos, la de su colaboración, y en cuál o cuáles se desarrolle cada caso de uso, en toda aplicación se requiere una coordinación de las distintas operaciones de su funcionamiento y de las diferentes opciones para su uso. Es lo que se denomina el 'Nivel de Aplicación' o el 'Control de la Aplicación'; que, al igual que los demás módulos, está en estrecha sincronía con la capa de presentación (IU).

Tras el arranque, en la inicialización de una aplicación multiusuario, como ésta, parece lógico contemplar que las características de acceso de los usuarios será de las primeras cosas que va a requerir el control de la aplicación.

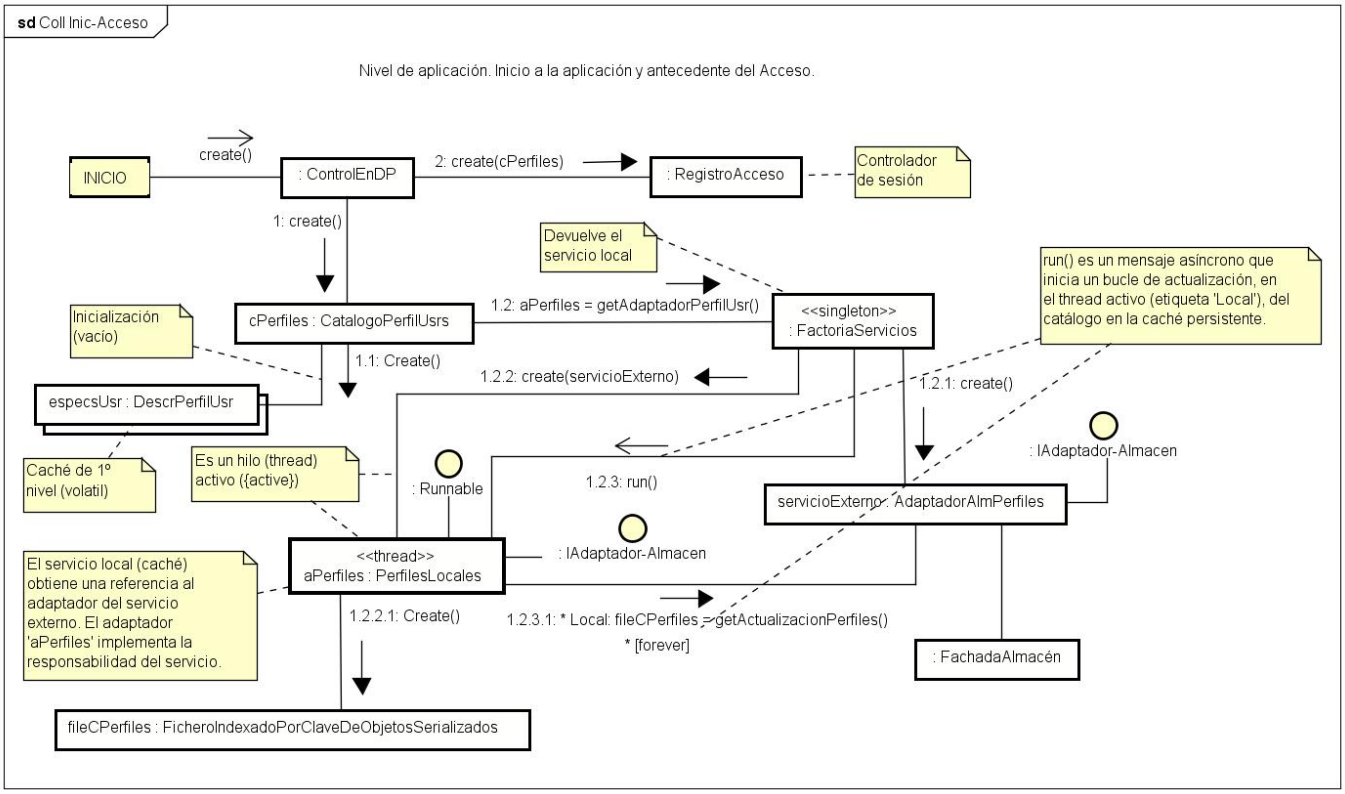
Por otro lado, en la asignatura también se ha asumido la hipótesis de que los datos e información que puede necesitar la aplicación, en cualquier momento, están almacenados, o se suministran, desde fuera de la aplicación. De manera genérica, vamos a llamar 'Almacén' a la entidad en la que se encuentra la información, incluida la relativa a cada usuario.

Según se han establecido los límites del funcionamiento del caso de uso, realmente no se necesita tomar ninguna decisión sobre cómo se va a estructurar (con qué clases), en el funcionamiento de la aplicación, los datos relativos al usuario. Si denominamos 'Perfil' a esos datos, tampoco es muy relevante si contiene sus datos de acceso, los personales, los parámetros de funcionalidad asociados a su rol, su colección de planes de entrenamiento, el histórico de sus datos biométricos, los de su evolución en los entrenamientos, etc.; juntos en el Perfil o en estructuras diferenciadas, es irrelevante. Estrictamente, la aplicación debe obtener de Almacén justo la información que

necesita para realizar la operación siguiente y esa información la manejará mediante la estructura (clase) más conveniente para la operación y que, también, habrá que diseñar.

Para simplificar todo el proceso, vamos a suponer que toda esa información del usuario, la que se requiere desde el acceso hasta el inicio del caso de uso, está contenida en la clase 'DescrPerfilUsr'. Esta simplificación significa poco detalle y fidelidad en la estructura del objeto DescrPerfilUsr, porque lo que se quiere resaltar es el esquema de funcionamiento.

En este escenario, en el inicio:

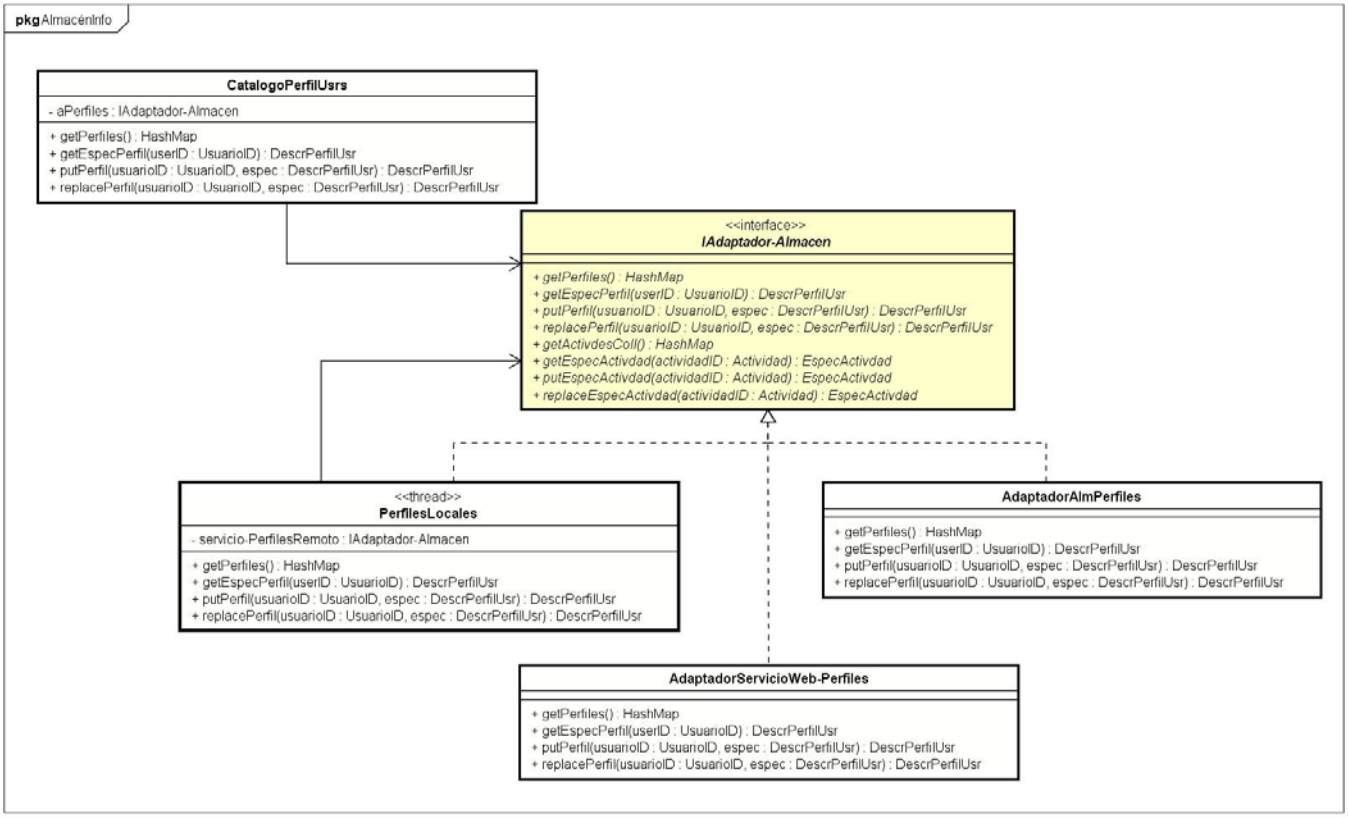


Se ha representado, con un diagrama de colaboración, la secuencia para la inicialización de un contenedor para una colección de DescrPerfilUsr, persistente, mediante un esquema de caché similar al de Productos en PdV, en las páginas 475-479 del libro.

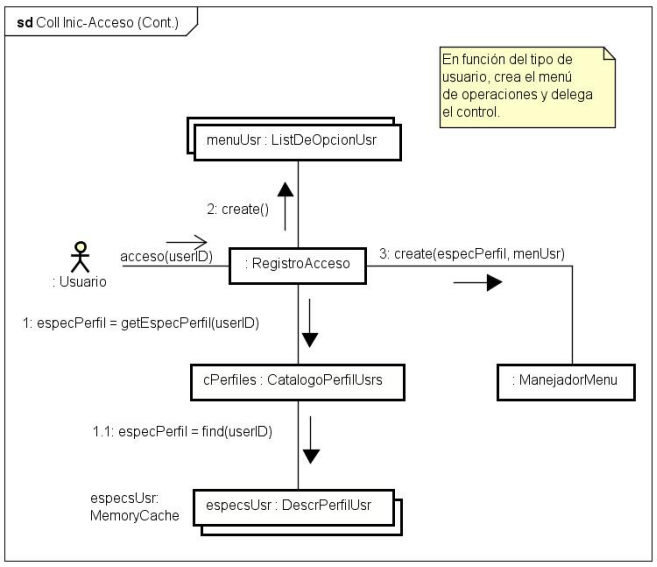
En realidad, dada la naturaleza *ligera* de la aplicación, el Almacén podría ser un fichero persistente con un formato cualquiera, o un servicio de información en *cloud*.

Como se puede observar, el catálogo tiene como componente una instancia del adaptador que le permite acceder a los datos que se gestionan en el Almacén. Pero ese acceso, sólo le está permitido para los datos que maneja, la colección de DescrPerfilUsr. El servicio de acceso, la instancia del adaptador aPerfiles, a través del servicioExterno, se encarga de traducir, tanto la estructura de la información como la forma de manipularla, a la manera en que Almacén gestiona su contenido.

A esta *mecánica* de funcionamiento, le corresponde esta estructura de clases:

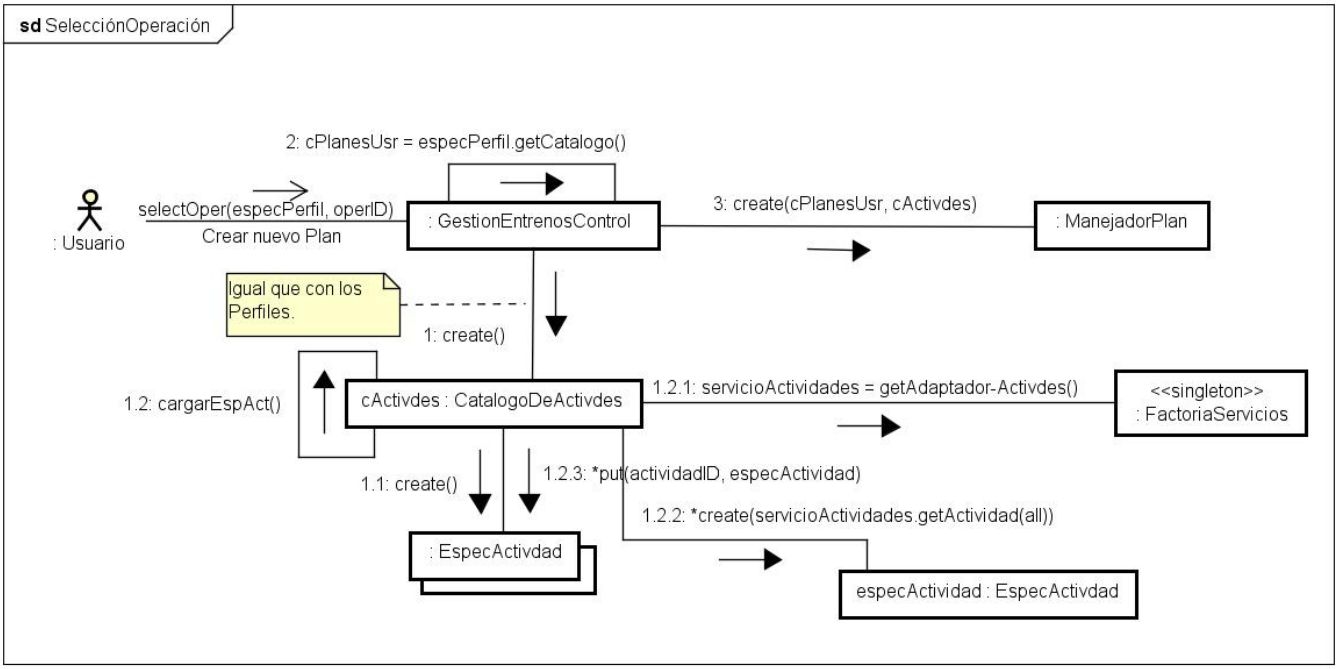


Una vez creado el controlador de sesión, RegistroAcceso, con el catálogo de perfiles, cPerfiles, la aplicación ya tiene los datos necesarios para que el usuario se identifique y acceda:



A partir de aquí, se pueden suceder más o menos etapas de delegación del control y de transmisión de la información relevante, según se organice el esquema de la funcionalidad, y de los menús, en la aplicación.

Como se ha mencionado, en cada una de estas etapas, con el userID y, quizás, los datos imprescindibles de las selecciones anteriores, la aplicación debería obtener la información que necesite desde el Almacén. Pero, en esta simplificación, toda la información que se requiere del usuario se transmite en especPerfil. Y, así, hasta llegar a la opción de 'Mantenimiento de Entrenamientos', operación CRUD controlada por GestionEntrenosControl y en la que, debido al objetivo de su tratamiento, se requiere extraer la colección de Planes de Entrenamiento (cPlanesUsr), desde especPerfil, y la colección de ejercicios '*tipo*' o Actividades (cActivdes):



En este último diagrama, se han simplificado alguno de los pasos para la obtención de los catálogos.

En el comienzo del caso de uso, el control se ha delegado en ManejadorPlan y dispone de toda la información para realizar su cometido.