Planteamiento del escenario de estudio.

El escenario en el que se situará el funcionamiento del caso de uso (pregunta 2), consiste en una aplicación que supervisa la obtención de **billetes del transporte público de una ciudad** (aplicación llamada Voy!).

A efectos del servicio a los usuarios, el sistema de transporte público de una ciudad ha unificado el ferrocarril subterráneo (metro), el ligero de superficie, el tranvía y el autobús urbano. La red de líneas de cada medio se combina en un mapa único para su utilización, de manera que se puede ir desde cualquier estación hasta cualquier otro punto de la ciudad (con parada).

Para realizar un `viaje`, hay que comprar un `billete` en el que se recoge la información del itinerario contratado y da acceso a los vehículos que lo cubren. Además, el `billete` tiene una vigencia determinada.

Se denomina '`Tramo`' al recorrido que realiza la línea de un determinado medio de transporte entre 2 paradas consecutivas. Cada uno tiene su `precio`.

El coste del `billete` es la suma de los `precios` de los `tramos` que constituyen el `viaje` contratado.

El sistema dispone de:

- Terminales fijos en cada parada, con los que se puede obtener el `billete` o utilizar la '*tarjeta transporte*'. Admiten el pago en metálico o con cualquier tipo de tarjeta, incluida la *tarjeta transporte*. Una vez abonado el `viaje`, emiten un `billete` físico o instalan un `billete` digital en la *tarjeta transporte*.
- Terminales de control de acceso en cada vehículo de transporte, capaces de comunicarse con el `billete` físico, con la aplicación instalada en un dispositivo móvil particular o con la *tarjeta transporte*.
- La *tarjeta transporte* es un dispositivo electrónico personal, con un saldo de prepago alojado en un '*monedero electrónico*', y por la que se abona una cuota mensual que otorga una rebaja en el precio del `billete`. Una vez comprado el `viaje` con ella, también permite el acceso a los vehículos de transporte correspondientes. Desde los terminales fijos de parada, se puede renovar la cuota mensual y recargar el saldo.

Para obtener un `billete`, el usuario debe poder seleccionar cada `tramo` por el que quiere que transite su `viaje` (por ejemplo, en un mapa interactivo que distinga entre las diferentes líneas y medios de transporte; aunque coincidan geográficamente). Los `tramos` seleccionados pueden ser discontinuos, pero la vigencia del `billete` siempre corresponde al cálculo entre la parada de origen y el final del itinerario continuo.

Se pretende construir Voy! como una aplicación ligera y flexible, que opere tanto en los terminales fijos de parada como desde los dispositivos móviles particulares que la tengan instalada, y que permita manejar los servicios con estos dispositivos y los descritos anteriormente.

Igual que ocurre con la *tarjeta transporte*, la aplicación instalada en móvil también utiliza una cuenta personal con un saldo de prepago con el que se abona el coste del `viaje` y, una vez obtenido el `billete` digital, queda instalado en el móvil; por lo que debe estar encendido y activo para poder acceder a los vehículos de transporte.

Para mejorar la funcionalidad, la aplicación dispone de un módulo 'Planificador de Itinerarios' con geolocalización, acceso a la información de tráfico y a la de densidad de utilización en todos los vehículos de la red, accesibilidad de las paradas, etc. Con estos datos, la función del Planificador (Voy!Mejor) es calcular el mejor itinerario en función de algún filtro o criterio: generalmente "más barato", "más rápido" o "más cómodo". **¡¡Ojo!! El Planificador está embebido en la aplicación Voy!**, no es externo a ella.

Al usar el Planificador el usuario sólo debe seleccionar la estación de origen, de destino y el criterio de optimización del itinerario. A partir de ahí, se *compone* la secuencia de tramos contiguos, según el criterio seleccionado, hasta llegar al destino.

Se use o no el Planificador, el billete sólo tiene validez para los tramos contratados y se comprueba en los puntos de control de acceso de los vehículos que se utilicen. Así mismo, esa validez termina 30 min. después del tiempo necesario, estimado por el Planificador, para alcanzar el destino según el itinerario continuo que corresponda.

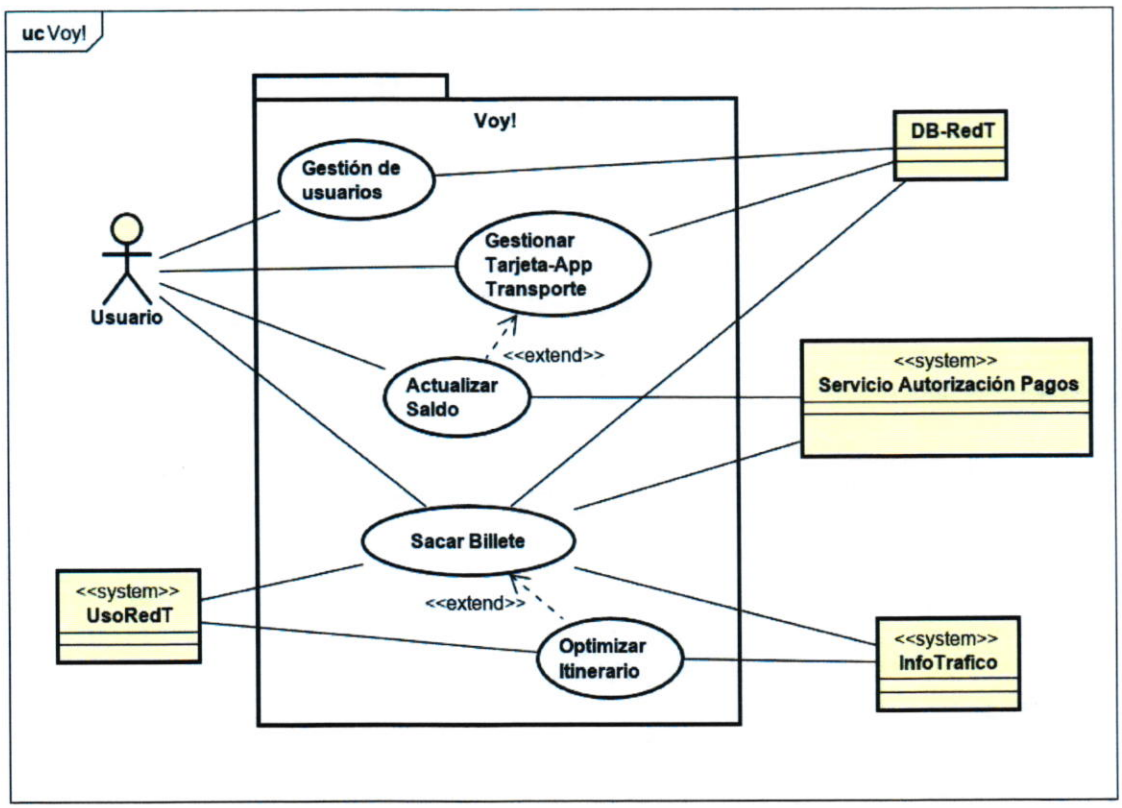
Detalles y simplificaciones admitidas:

- Todos los tramos de la red, las paradas, los terminales y los vehículos del servicio están operativos en todas las líneas y tipos de transporte, y en buen estado.
- En el modo 'manual' (sin la optimización del Planificador), la selección de los tramos debe permitir su edición flexible (anulación, modificación, etc.).
- Mientras esté vigente el billete, todos sus tramos se pueden utilizar cuantas veces se desee, en cualquier sentido. De cualquier forma, el billete es personal e intransferible.
- 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.

Enunciado de las preguntas.

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

1. 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 Voy!



2. Escriba el caso de uso *<<ProcesarVentaViaje>>* 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.

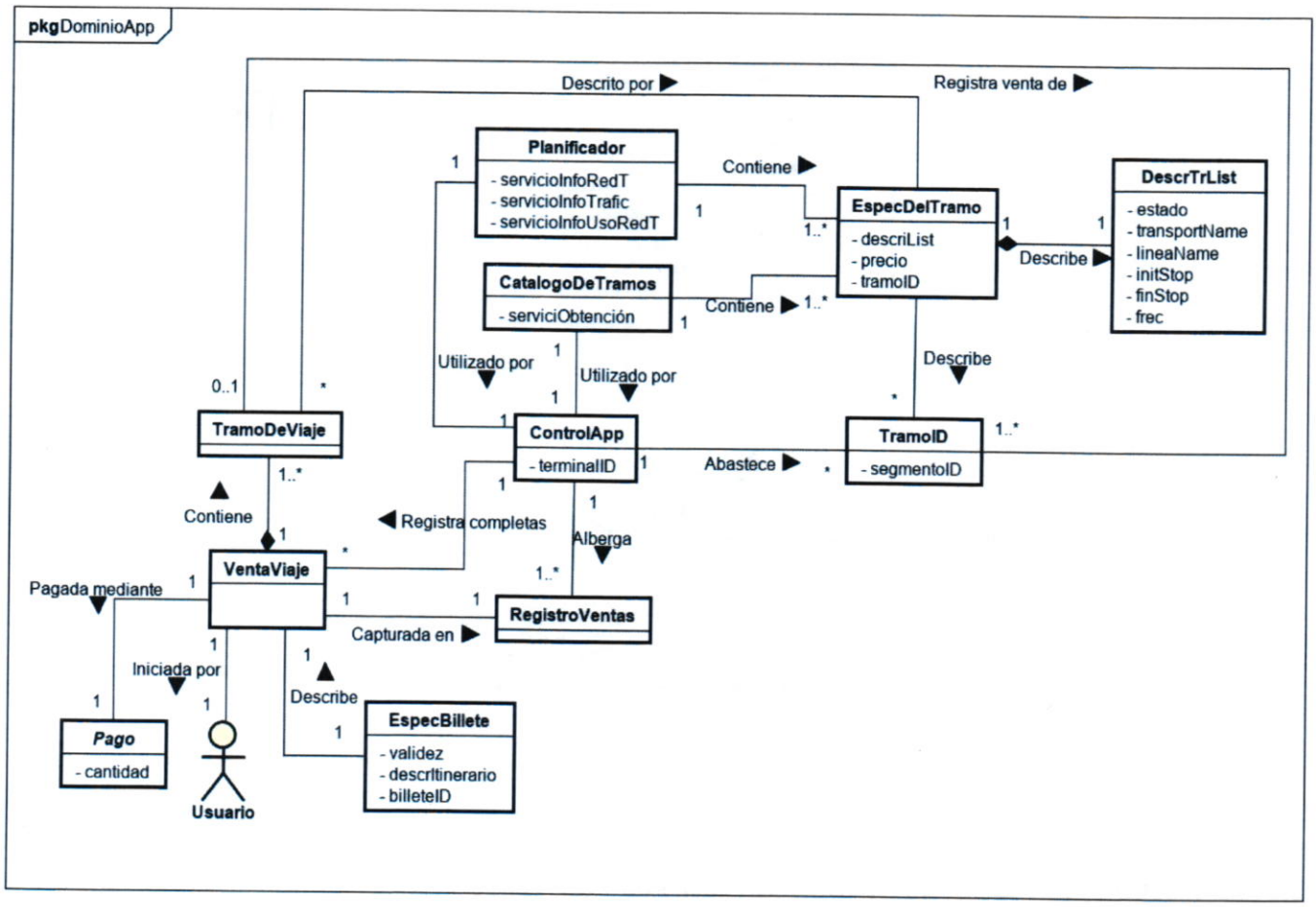
> Consiste en la compra de un `Billete` y la línea de acciones del flujo básico de éxito discurre desde que el usuario solicita un `Billete` para un viaje, sin optimizar el recorrido, hasta que lo obtiene.
>
> Supóngase que el usuario utiliza el terminal de una parada, rehúsa la optimización del módulo `Planificador` y selecciona 5 `tramos`, que paga con su tarjeta de débito. **Aténgase, estrictamente, a esta operación**.

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.

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

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

3. En relación con el caso de uso anterior, *<<ProcesarVentaViaje>>*, 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.



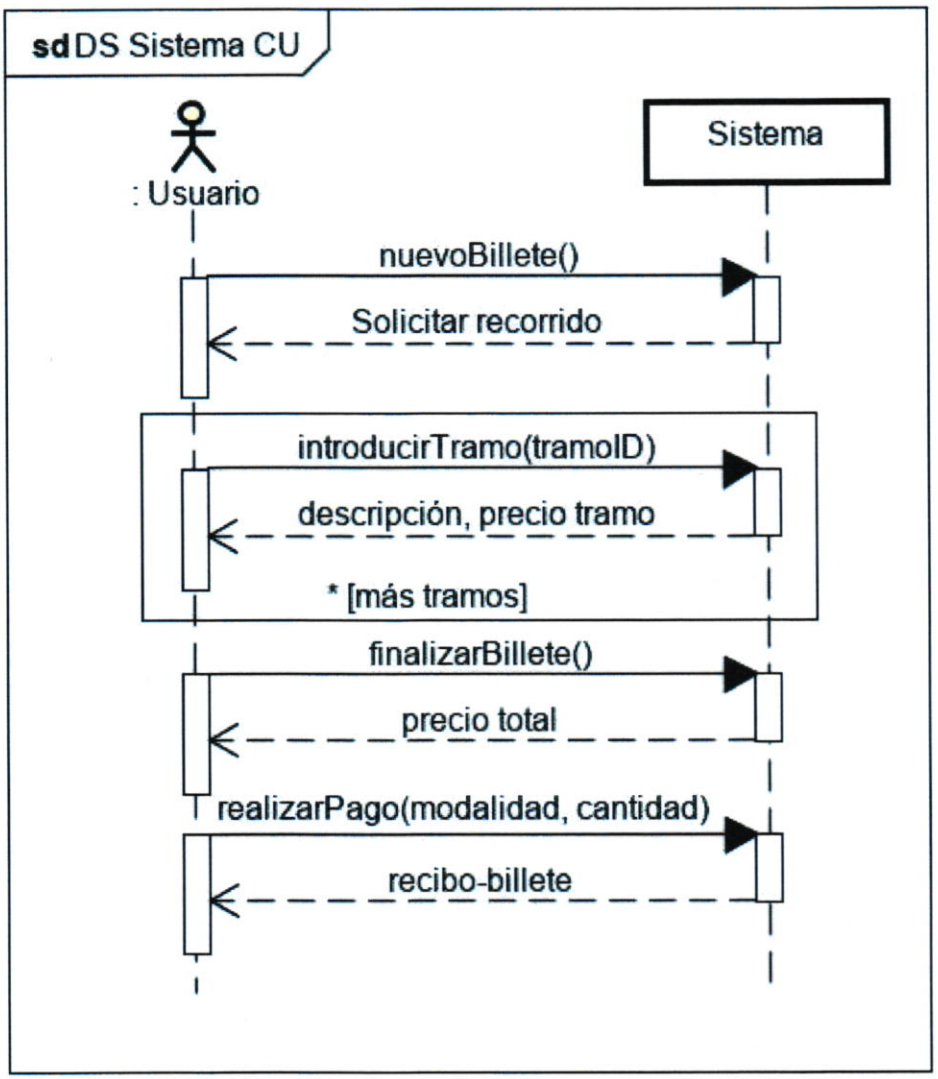
Sección 3. (Diseño) Evaluación de los ***Eventos del Caso de Uso***

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

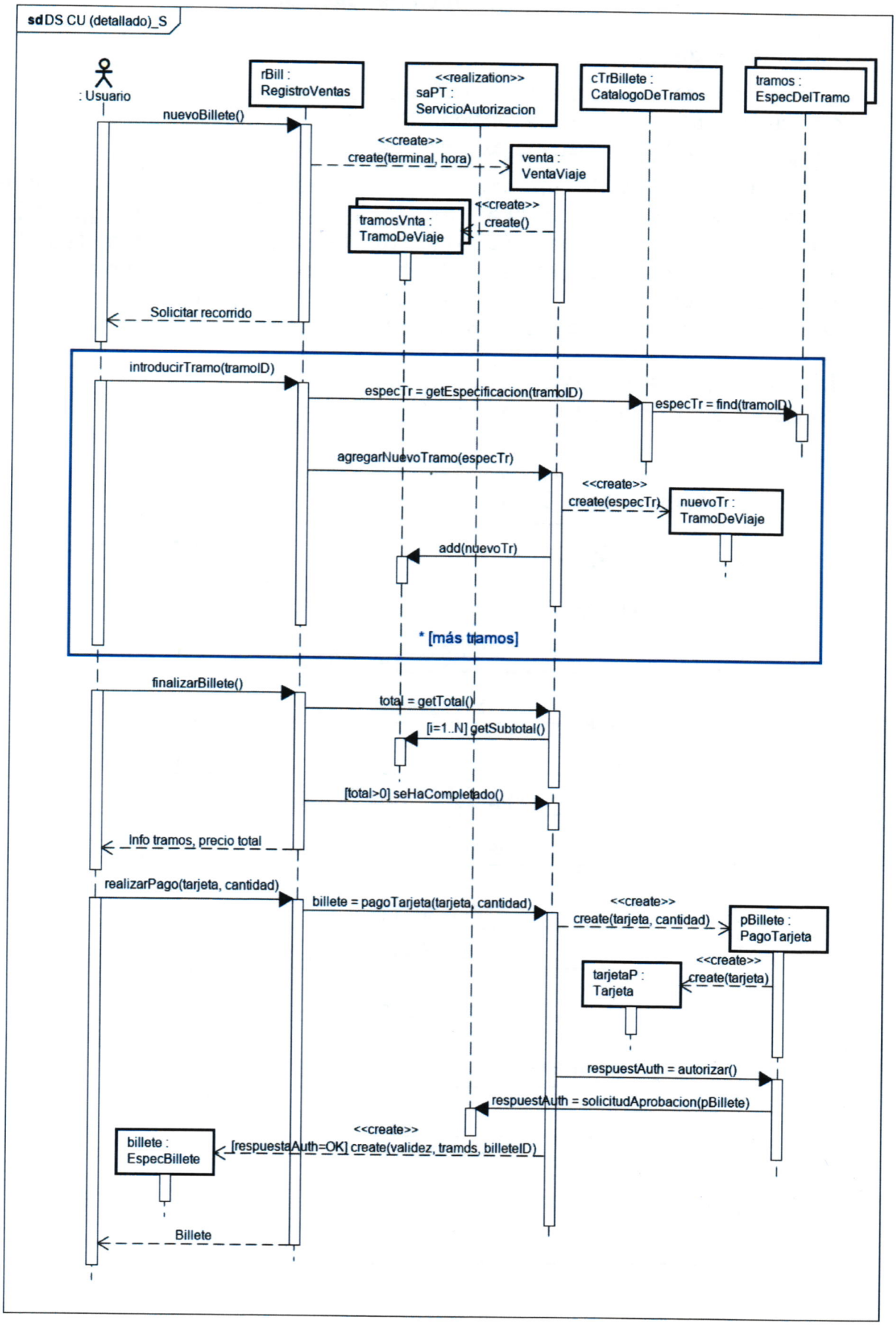
4.1. Circunscrito al caso de uso anterior *<<ProcesarVentaViaje>>*, construya un Diagrama de Secuencia (diagrama de interacción DS) en UML. Represente los actores y los eventos de los componentes del sistema para este caso de uso.

¡ATENCIÓN!: se pide un diagrama de secuencia en el que represente el paso de mensajes entre los actores y **los distintos objetos del modelo**, **NO** 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***.

Lo que no se pide (pero conviene hacerlo, *en sucio*, para empezar a construir el DS pedido y separar operaciones principales):



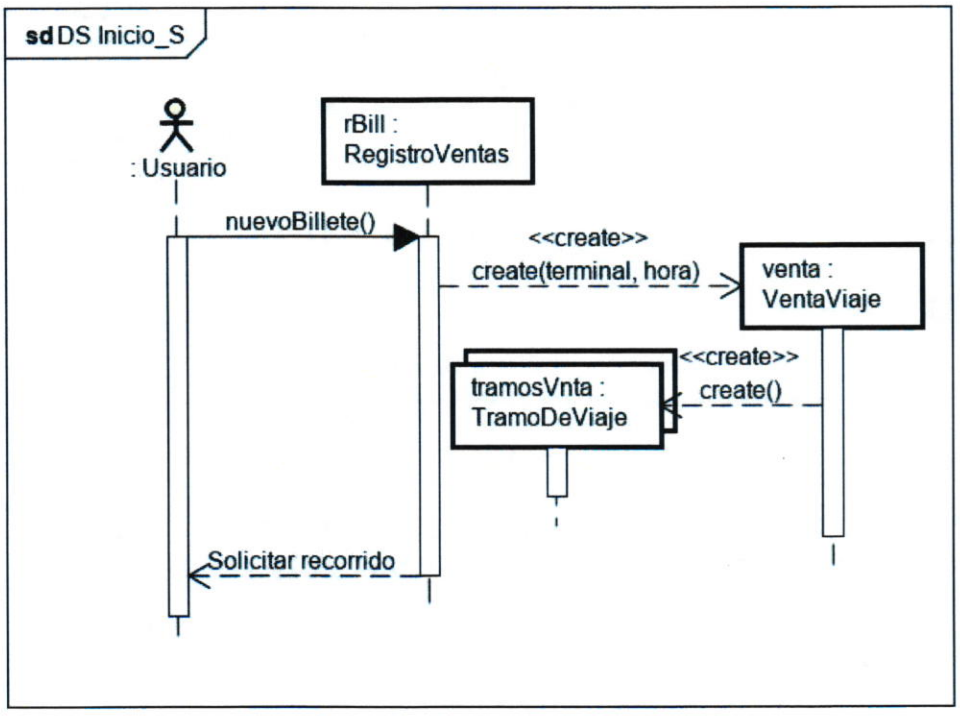
Ahora sí, el diagrama completo y simplificado[1]:



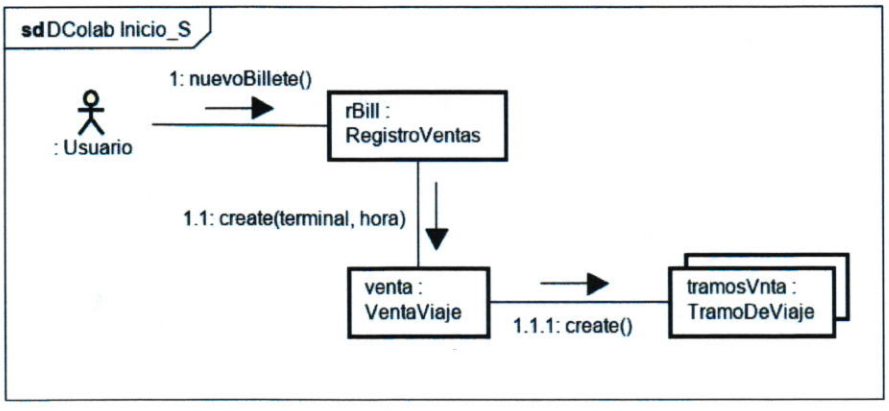
[1] Se han simplificado u omitido algunas acciones que, aunque pueden ser relevantes y fundamentan el funcionamiento de algunas partes del CU, no se espera que se reflejen en las respuestas de los exámenes.

Descomposición por operaciones. Se admiten ambas modalidades. Lo importante es que se reflejen correctamente los mecanismos de búsqueda, obtención y organización de la información (datos, instancias y clases), preservando los principios de 'Controlador', 'Experto' – 'Creador' y 'Bajo Acoplamiento', la privacidad de los objetos y la coherencia del funcionamiento con el comportamiento pedido en el enunciado.

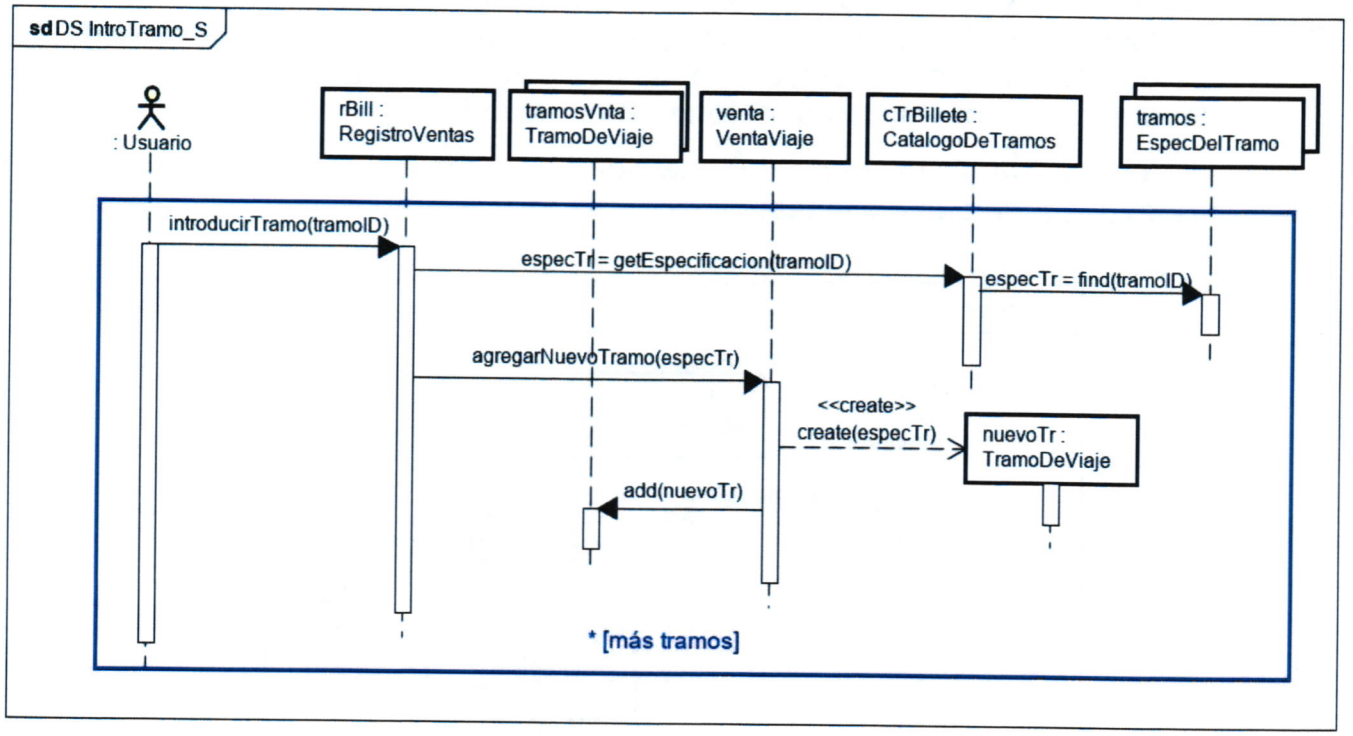
La operación de inicio no debería admitirse como principal, debido a la menor relevancia de sus acciones:



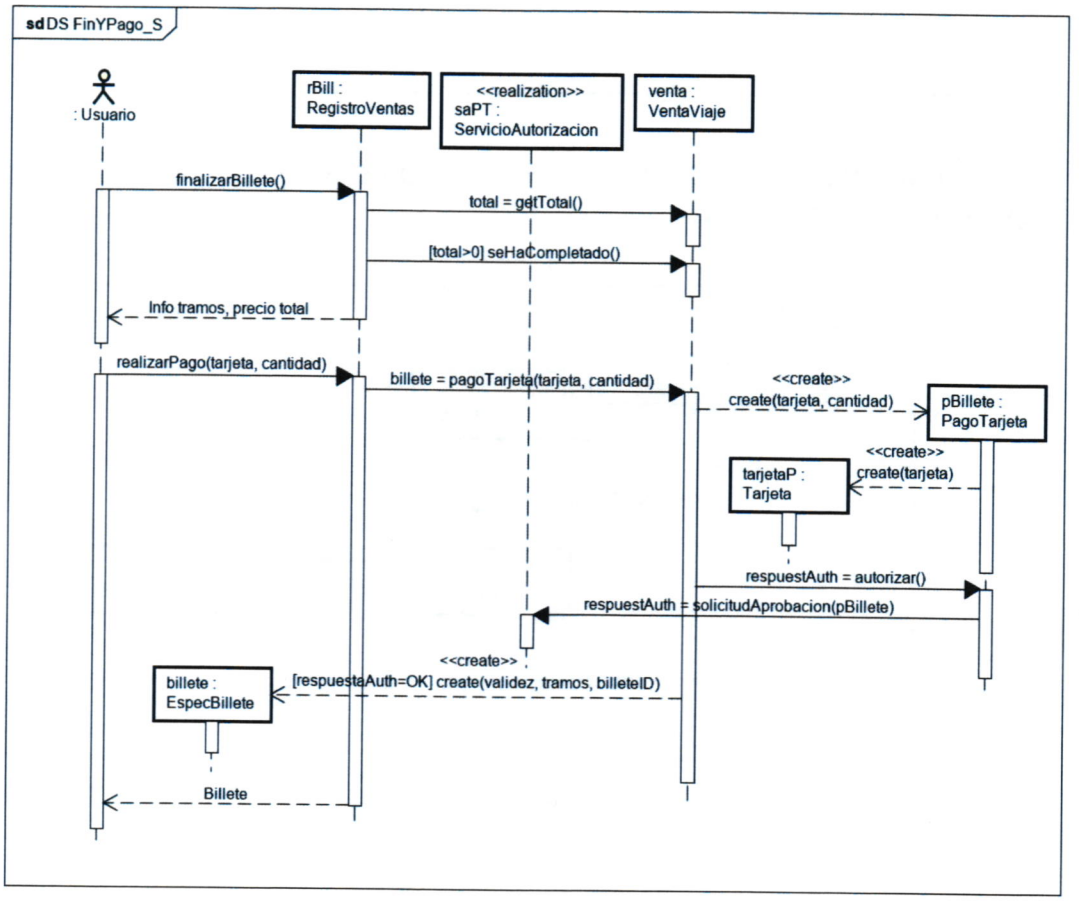
Por no considerarlo, se incluye aquí el diagrama de colaboración a que daría lugar en la pregunta 5:



A la *operación A*, la llamaremos 'IntroducirTramo'. Su diagrama de secuencia sería:



Para la *operación B*, se ha unido la operación 'Finalizar' (que no es relevante, ni puede ser principal, pero intermedia con la de 'RealizarPago':

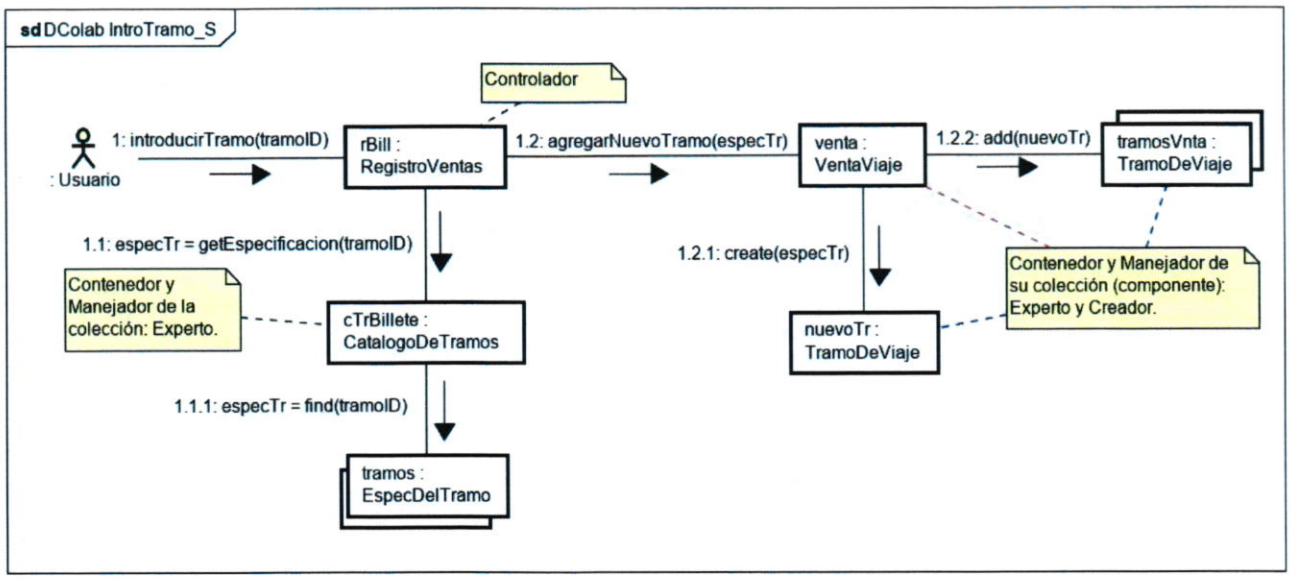


4.2. A partir de este DS, escriba y desarrolle los contratos de **2** operaciones principales: (llamémoslas, aquí, '*OperacionA*' y '*OperacionB*'. Usted puede llamarlas como le convenga; pero, en adelante, debe mantener esa denominación. Estas operaciones deben ser principales, consecutivas (en la medida 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**.

IntroducirTramo y Finalizar+ RealizarPago.

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

5. A partir del contrato de la operación <<*IntroducirTramo*>> que ha 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.

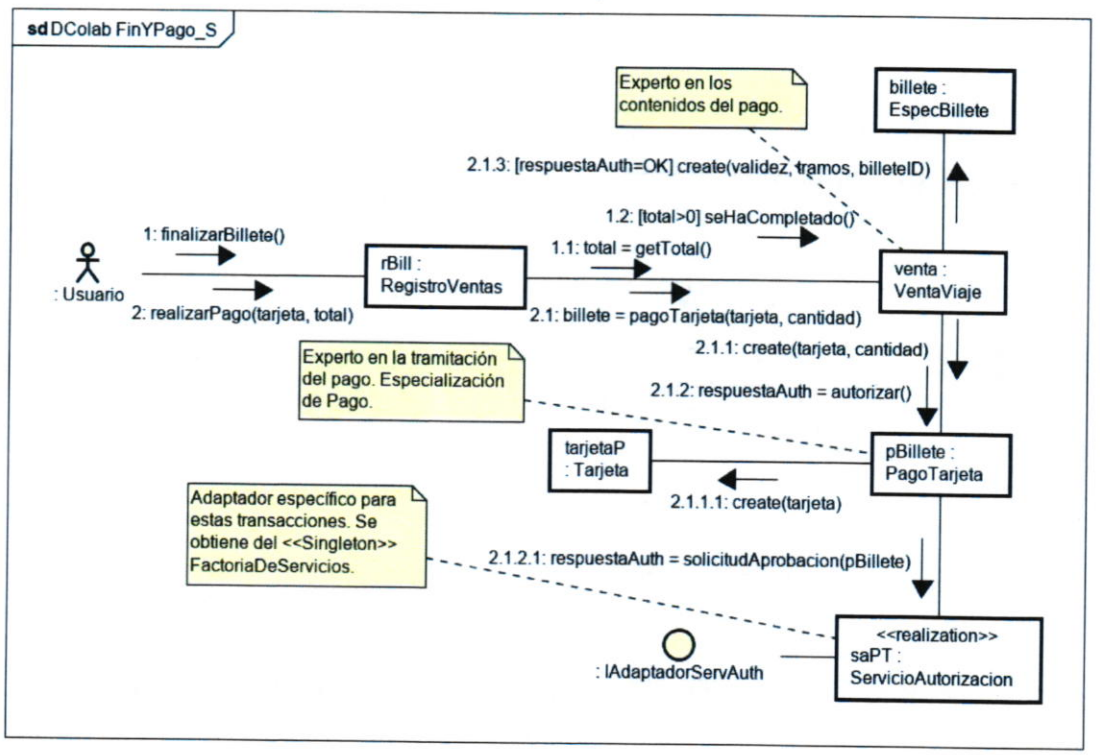


6. A partir del contrato de la operación <<*Finalizar+ RealizarPago*>> que ha indicado en la pregunta 4 (con la denominación que haya utilizado allí para esa operación), complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro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

Aquí es donde se recogen la mayoría de las simplificaciones:

- La utilidad efectiva del billete no se debe a la EspecDelTramo de cada tramo, usado para la venta, sino a la lista de datos que hay en su descripción, DescTrList (ver modelo de dominio). Por ello, en operaciones anteriores, a la vez que se ha formado una lista con las EspecDelTramo, se debería haber formado otra con sus descripciones, DescTrList, para *pasársela* a billete como componente.
- De igual forma, se obviado la obtención del tiempo de validez del billete, obtenido del Planificador a partir de esa lista de los datos de los tramos (la formada con elementos DescTrList); y la obtención de billeteID, que requeriría una consulta a un nivel de control superior.

- La gestión de las transacciones del Pago, y de los de Autorización, también se gestionan desde un nivel superior. En realidad, la operación RealizarPago se desdobla en varios eventos desde el actor, uno de los cuáles significa una solicitud de autorización del cargo en su tarjeta. La obtención de ese servicio (de acceso a un elemento externo) se realiza desde el objeto que lo gestiona (la especialización de Pago para el pago con tarjeta, PagoTarjeta) y a través de una Factoría de Servicios (una clase <<Singleton>>, global, que proporciona el acceso a través de una interfaz abstracta; la misma clase que proporciona, para los catálogos, el acceso al almacén de datos de la aplicación). La solicitud de este servicio también implica consultas a nivel de la aplicación para obtener comercianteID y terminalID.



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

7. Elabore un diagrama de clases para el caso de uso que se está tratando *<<ProcesarVentaViaje>>* (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 (`Viaje` o `Itinerario`). 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. 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 *<<ProcesarVentaViaje>>*. 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.