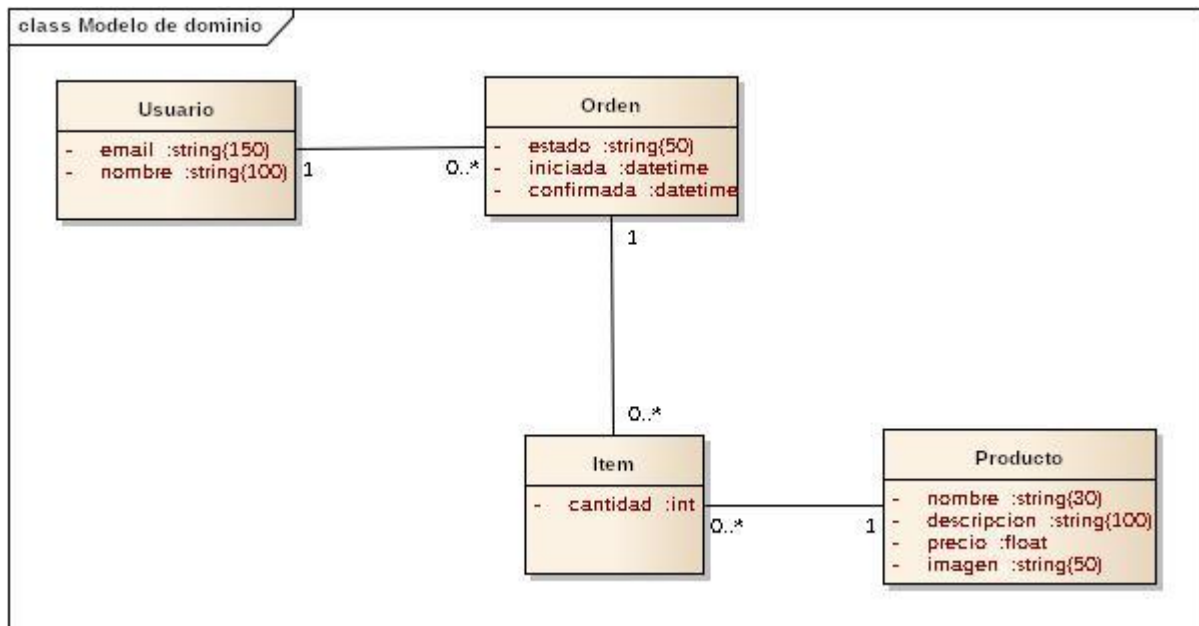


Ejercicio 6

Agregar producto (FP - Paso 5-8)

5	El cliente ingresa la cantidad a comprar y confirma el ítem
6	Si el cliente no tiene una orden iniciada, el sistema crea una orden para el cliente
7	El sistema agrega el ítem a la orden, con el producto y la cantidad (RN02)
8	El sistema vuelve al paso 1 informando que se ingresó un nuevo producto a la orden

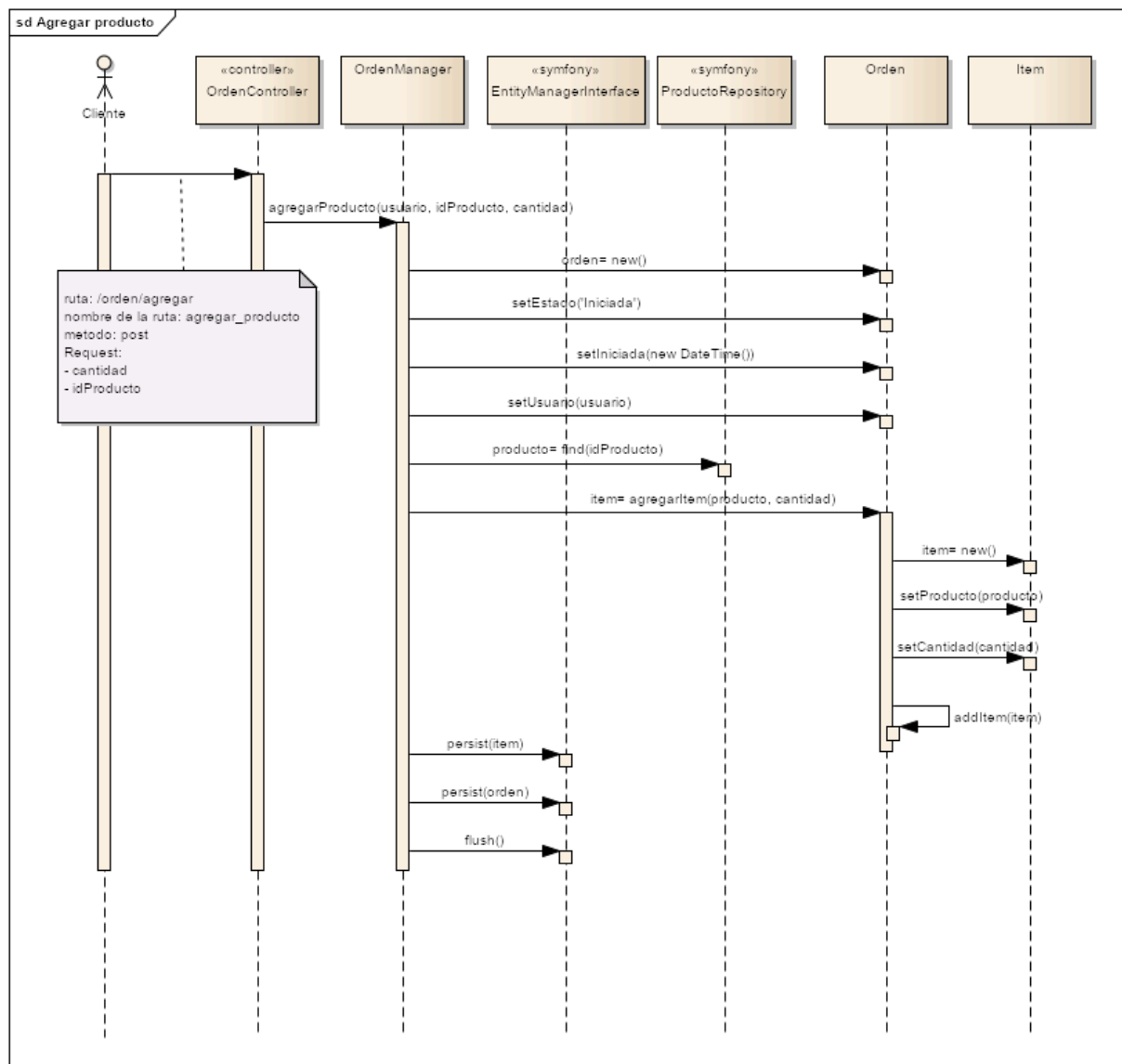
Análisis del modelo de dominio



1. Orden:
 - a. Tiene atributos propios
 - b. Pertenece al usuario
 - c. Contiene objetos items
2. Item:
 - a. Tiene el atributo cantidad
 - b. contiene un objeto producto

Ejercicio 6 - Iteración 1 - Crear la orden y agregar un item

Diagrama de secuencia



Análisis de dependencias

1. OrdenController envía mensajes a OrdenManager (Inyectar)
2. OrdenManager envía mensajes a:
 - a. Orden (crear)
 - b. EntityManagerInterface (Inyectar en el constructor)
 - c. ProductoRepository (Inyectar en el constructor)
3. Orden envía mensajes a Item (crear)

Codificación

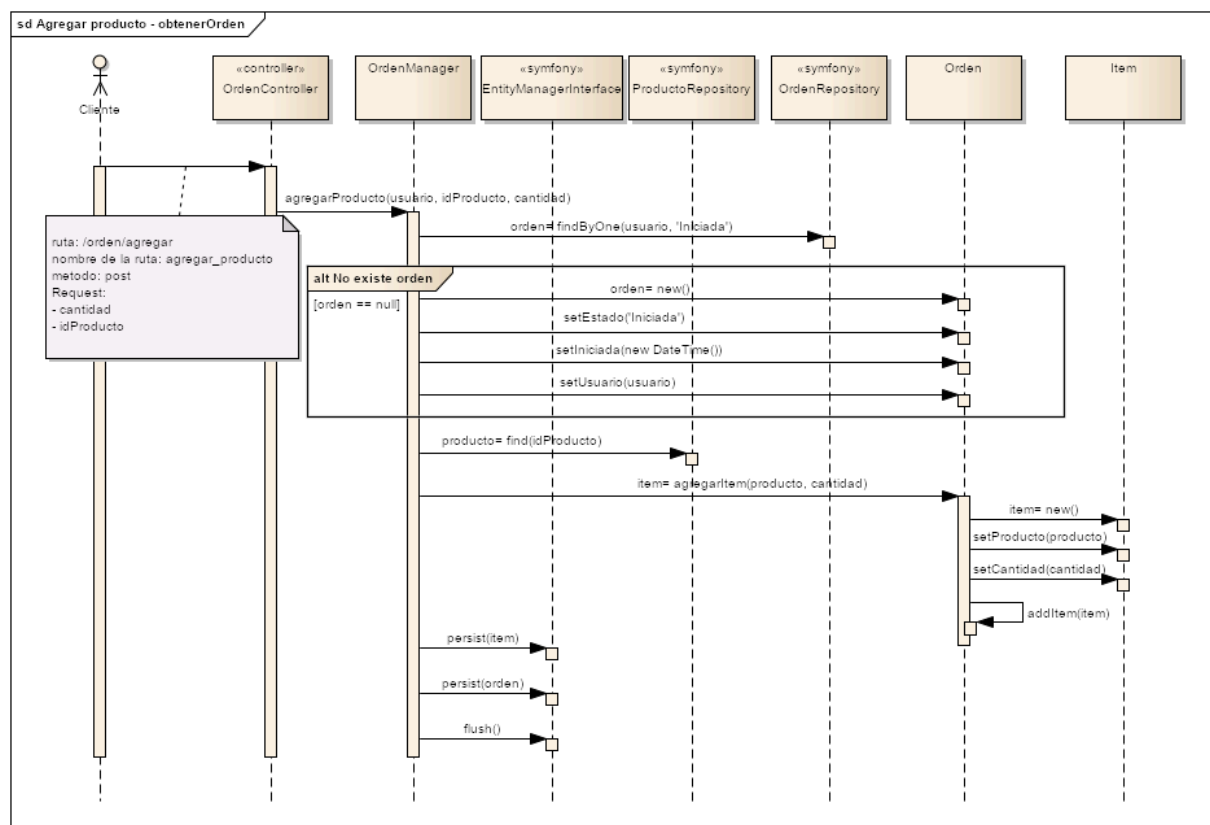
1. Codificar el método agregarItem de la clase Orden
2. Crear la clase OrdenManager y crear el método agregarProducto
3. Modificar el método agregarProducto de la clase OrdenController para que invoque al método agregarProducto de OrdenManager

Ejercicio 6 - Iteración 2 - Verificar si el usuario ya tiene iniciado una orden

Agregar producto (FP - Paso 5-8)

5	El cliente ingresa la cantidad a comprar y confirma el ítem
6	Si el cliente no tiene una orden iniciada, el sistema crea una orden para el cliente
7	El sistema agrega el ítem a la orden, con el producto y la cantidad (RN02)
8	El sistema vuelve al paso 1 informando que se ingresó un nuevo producto a la orden

Diagrama de secuencia



Análisis de dependencias

4. OrdenManager envía mensajes a:
 - a. OrdenRepository (Inyectar en el constructor)

Codificación

1. Crear el método privado obtenerOrden dentro de la clase OrdenManager que devuelva una orden nueva o una ya iniciada

Ejercicio 6 - Iteración 3 - Verificar si el ítem ya existe

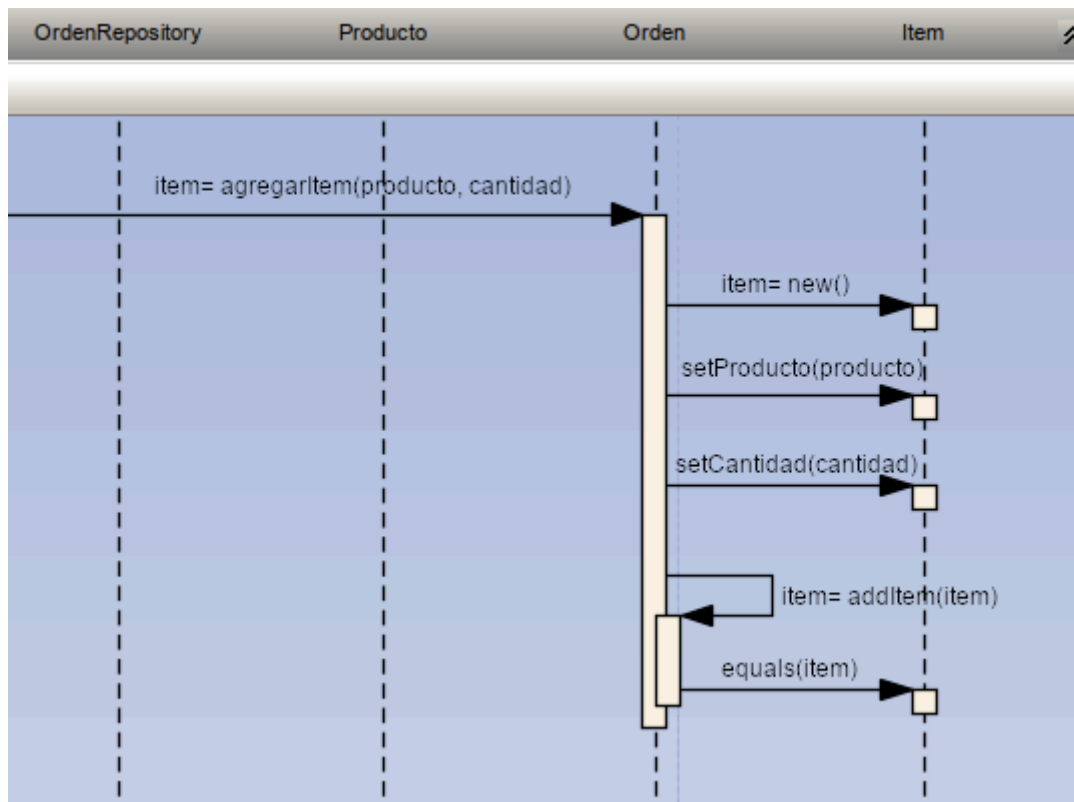
Verificar si el ítem ya existe y hay que modificar la cantidad o se debe agregar un nuevo ítem

Agregar producto (FP - Paso 5-8)

5	El cliente ingresa la cantidad a comprar y confirma el ítem
6	Si el cliente no tiene una orden iniciada, el sistema crea una orden para el cliente
7	El sistema agrega el ítem a la orden, con el producto y la cantidad (RN02)
8	El sistema vuelve al paso 1 informando que se ingresó un nuevo producto a la orden

Reglas de Negocio (RN)
RN01 - Cantidad a comprar La cantidad a comprar debe ser un valor numérico mayor a 0
RN02 - Ingresar un ítem al carrito Si ya existe el ítem en la orden se debe reemplazar la cantidad ingresada

Diagrama de secuencia



Codificación

1. Agregar a la clase item el método equals que compara si dos items tienen el mismo producto
2. Modificar el método addItem de la clase orden para que verifique si el item a insertar ya está en su lista de items. Si ya existe se debe actualizar la cantidad. Si no existe se debe agregar al array de items. Este método debe retornar el item nuevo o el modificado.

Ejercicio 7



Crear la página para ver la orden (FP - Paso 9-10)

9	El cliente repite los pasos 2 al 8 todas las veces que desee. Cuando lo desee el cliente finaliza la compra
10	<p>El sistema muestra una lista de los productos ingresados a la orden. Por cada producto muestra:</p> <ul style="list-style-type: none">- Imagen de producto- Nombre del producto- Cantidad adquirida- Precio unitario- Precio total <p>Además el sistema muestra el total de la orden (la suma de todos los productos).</p>

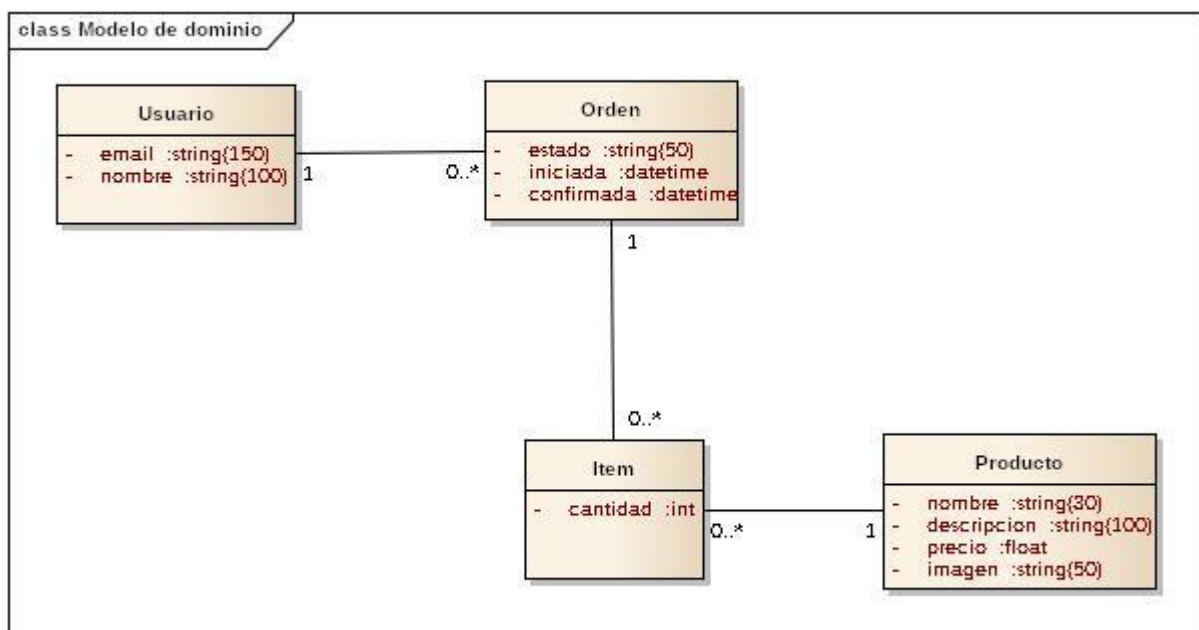
Ubicación dentro del diagrama de navegabilidad (Detalle de la orden)



Su orden

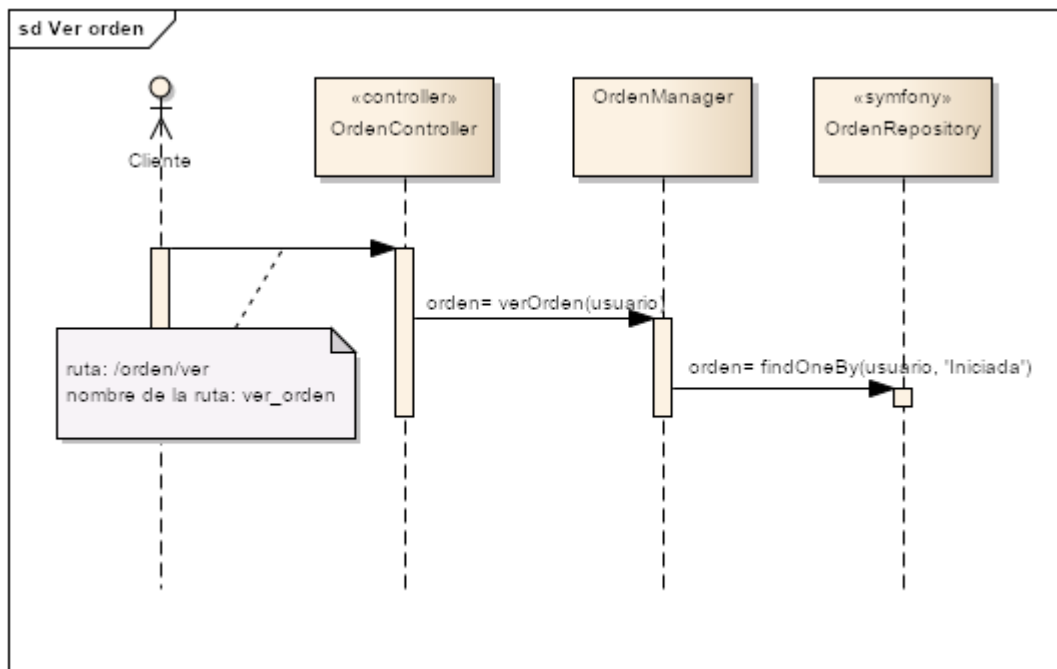
Items					Resumen	
Producto	Cantidad	Precio	Total		Total	\$ 1118
 Producto 2	4	\$ 67	\$ 268	<button>Eliminar</button>	<button>Confirmar compra</button>	
 Producto 7	2	\$ 425	\$ 850	<button>Eliminar</button>		

Análisis del modelo de dominio



1. Orden:
 - a. Contiene objetos items
 - b. Conoce su total
2. Item:
 - a. Tiene el atributo cantidad
 - b. Contiene un objeto producto
 - c. Conoce su total

Diagrama de secuencia



Codificación

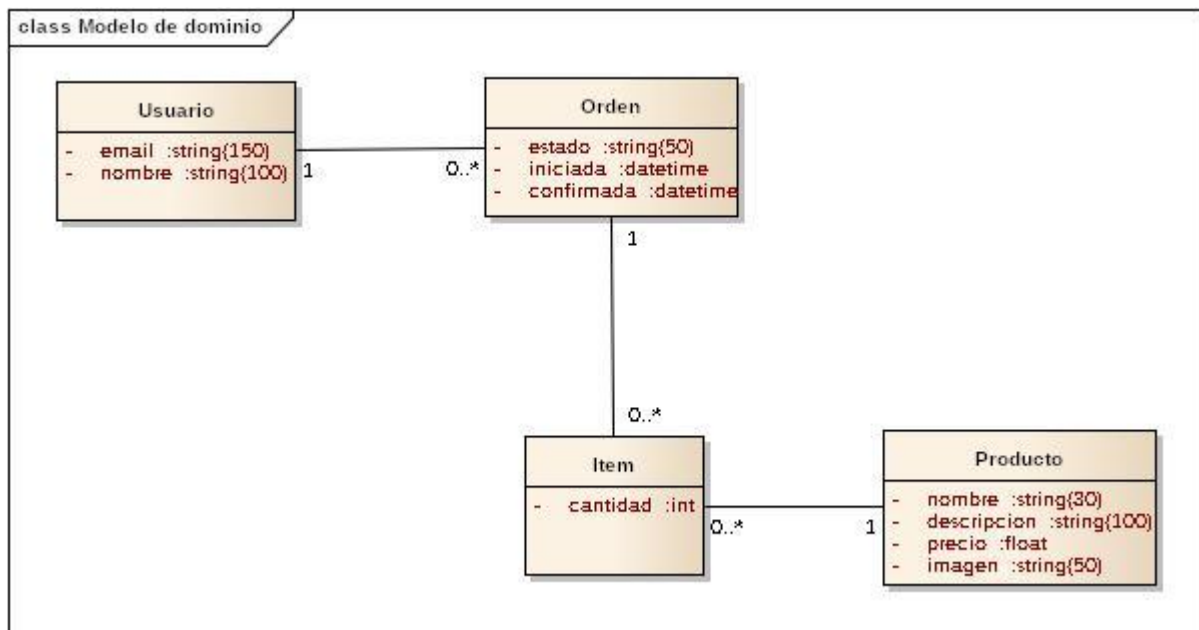
1. Crear el método verOrden en la clase OrdenManager
2. Programar el método getTotal en la clase Item. Utilizar este método para mostrar el total del ítem en la plantilla resumen.html.twig.
3. Programar el método getTotal en la clase Orden. Utilizar este método para mostrar el total de la orden en la plantilla resumen.html.twig
4. En OrdenController.php crear el método verOrden
 - a. ruta: /orden/ver
 - b. nombre de la ruta: ver_orden
 - c. template: orden/resumen.html.twig (se pasa como parámetro la orden)
5. En header.html.twig agregar el enlace a la ruta "ver_orden" en el botón "Ver orden"

Ejercicio 8

Finalizar compra (FP - Paso 11-12)

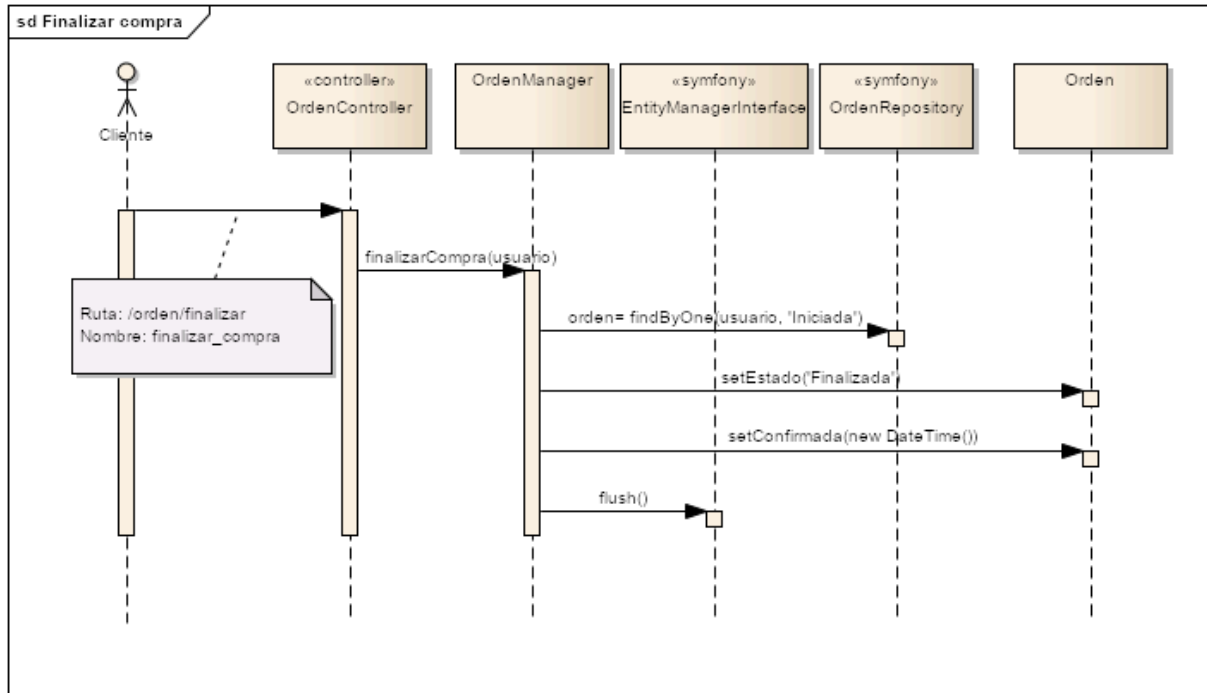
11	El cliente confirma la orden
12	El sistema guarda la orden

Análisis del modelo de dominio



3. Orden:
 - a. Setear el estado a "Finalizada"
 - b. Setear la fecha de confirmada

Diagrama de secuencia



Codificación

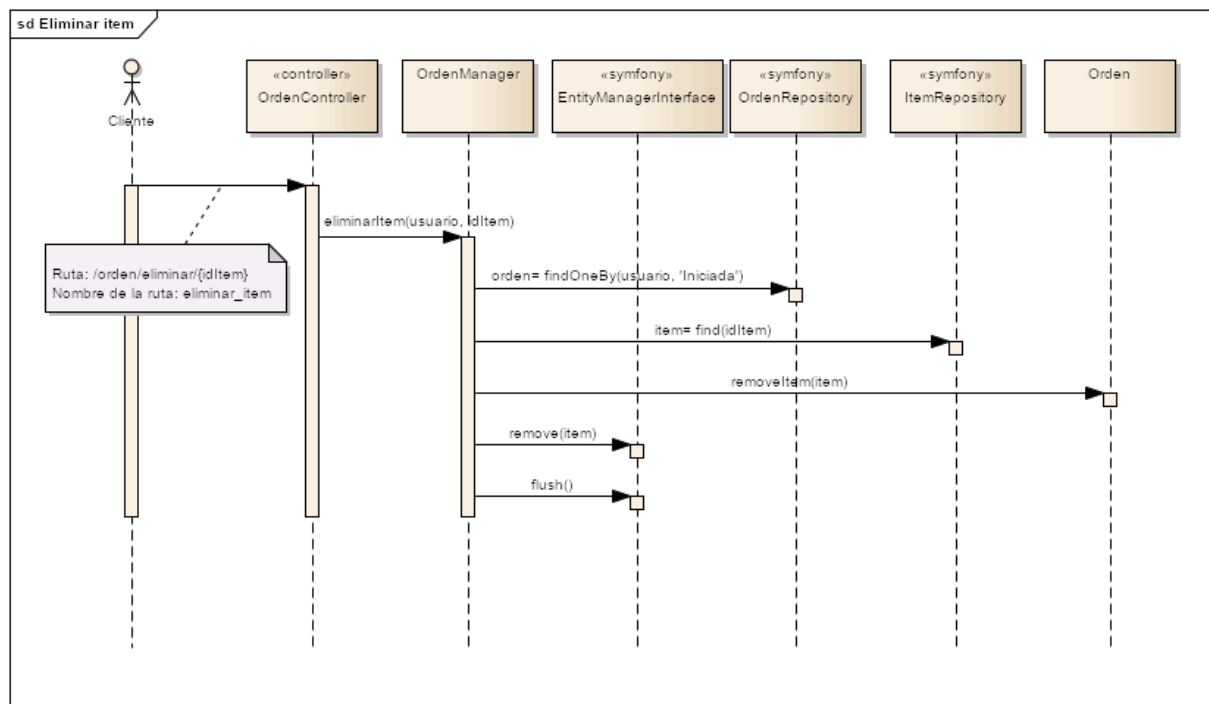
1. Crear el método `finalizarCompra` en `OrdenManager`
2. Crear el método `finalizarCompra` en `OrdenController`
 - a. Ruta: `/orden/finalizar`
 - b. Nombre: `finalizar_compra`
 - c. Invocar al método `finalizarCompra` de `OrdenManager`
 - d. Mostrar mensaje de operación exitosa
 - e. Redirigir a la ruta `listar_productos`
3. En `resumen.html.twig` agregar el enlace a la ruta `"finalizar_compra"` en el botón "Confirmar compra"

Ejercicio 9

Eliminar un ítem de la orden (FA01 - Paso 1-3)

1	En el paso 10 del flujo básico el cliente elimina un ítem de la orden
2	El sistema elimina el ítem de la orden
3	El sistema vuelve al paso 10 del flujo principal

Diagrama de secuencia



Análisis de dependencias

1. OrdenManager envía mensajes a:
 - a. OrderRepository (ya inyectado)
 - b. ItemRepository (inyectar en el constructor)
 - c. EntityManagerInterface (ya inyectado)

Codificación

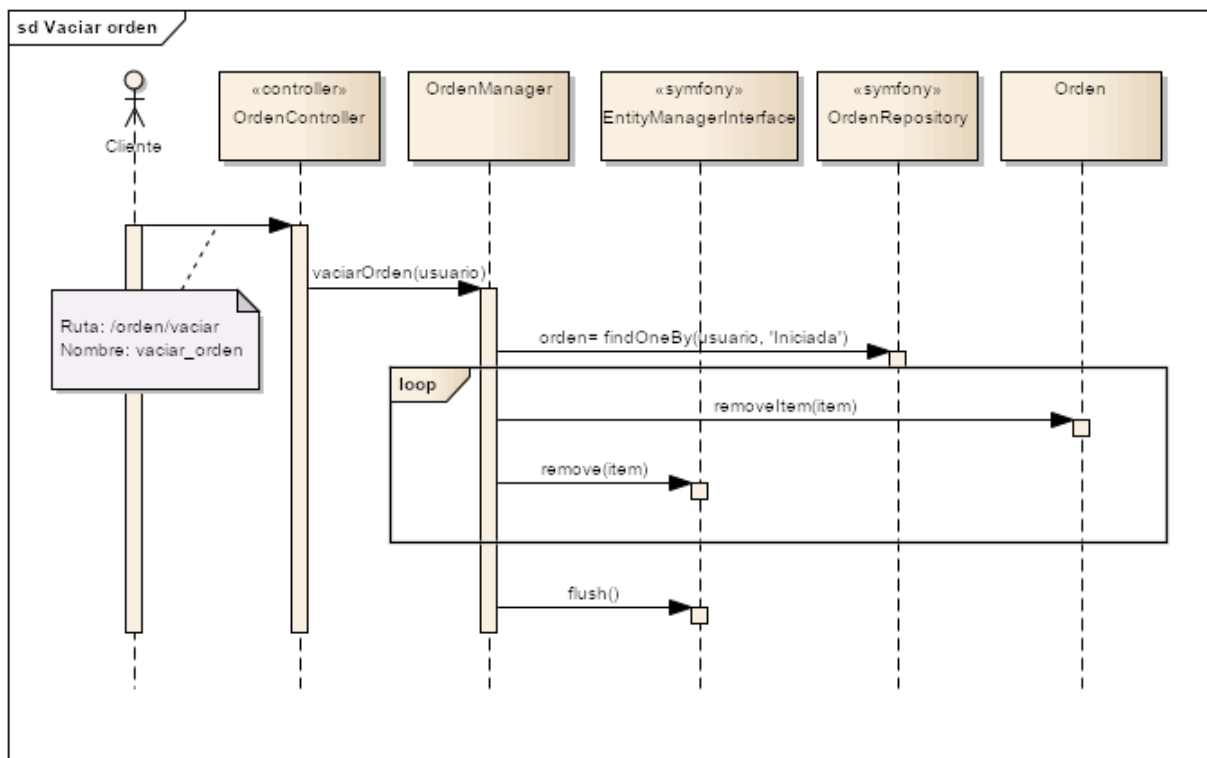
1. Crear el método eliminarItem en OrdenManager
2. Crear el método eliminarItem en OrdenController
 - a. Ruta: /orden/eliminar/{idItem}
 - b. Nombre: eliminar_item
 - c. Invocar al método eliminarItem de OrdenManager
 - d. Redirigir a la ruta ver_orden
3. Modificar resumen.twig.html para que invoque a la ruta eliminar_item con el parámetro correspondiente

Ejercicio 10

Vaciar la orden (FA02 - Paso 1-3)

1	En el paso 10 de flujo básico el cliente vacía la orden
2	El sistema elimina todos los ítems de la orden
3	El sistema vuelve al paso 10 del flujo principal

Diagrama de secuencia



Codificación

1. Crear el método `vaciarOrden` en `OrdenManager`
2. Crear el método `vaciarOrden` en `OrdenController`
 - a. Ruta: `/orden/vaciar`
 - b. Nombre: `vaciar_orden`
 - c. Invocar al método `vaciarOrden` de `OrdenManager`
 - d. Redirigir a la ruta `ver_orden`
3. Modificar `resumen.twig.html` para que invoque a la ruta `vaciar_orden`