

Documentación del proyecto final de la materia

Equipo: Equipo IA

Integrantes:

CU	Nombre
181335	Mariano Franco
181336	Andres Quevedo
181866	Carlos Delgado
176262	Diego Hernández

Descripción del documento:

El Equipo IA ha asumido el desafío de desarrollar una tienda en línea para abordar un problema específico. Nuestro objetivo es crear una solución que cumpla con los requisitos funcionales de una tienda de libros en línea, así como alcanzar un número determinado de transacciones en un período de tiempo específico.

En este proyecto, hemos documentado el control del proyecto, los requisitos de negocio, las historias de usuario, la arquitectura de la solución y las responsabilidades de los componentes de la arquitectura. También hemos establecido un sistema de seguimiento y control para evaluar el progreso y las desviaciones del proyecto. Además, hemos llevado a cabo pruebas funcionales y no funcionales, así como diseñado la arquitectura de los servicios web utilizados en la implementación de la solución.

En resumen, hemos trabajado en el desarrollo de una tienda en línea que cumpla con los requisitos y expectativas establecidos, abordando de manera eficiente el problema planteado.

Contenido

1 Control de proyecto.	1
2 Requerimientos De negocio	3
2.1 Requerimientos Funcionales	3
2.2 Requerimientos No Funcionales	3
3 Historias de Usuario	4
3.1 Happy Paths	4
3.1.1 Compra en línea	4
3.2 Rutas alternas	4
4 Documento de Arquitectura	5
4.1 Descripción general de la arquitectura propuesta	5
5 Responsabilidades de los Componentes de la Arquitectura y relación con las reglas de negocio.	5
6 Control de avance y desviaciones	6
7 Documentación de pruebas funcionales	7
7.1 Pruebas del BPEL	7
7.1.1 RN1.- Intento de compra con un cliente que no está en la BD	7
7.1.2 RN2.- Intento de comprar un libro que no existe	7
7.1.3 RN4.- Intento de compra de un libro sin suficiente stock	7
7.1.4 RN5. - Intento de compra con fondos insuficientes	7
7.1.5 Compra Exitosa (RN3, RN6, RN7, RN8, RN9)	7
8 Documentación de pruebas no funcionales	7
9 Arquitectura y versiones	8
10 Web Service de Almacén	8
11 Web Service de Pago	8
12 Web Service de Entregas	8
13 Estress general de cada servicio y a la aplicación	9
14 Anexos:	9

1 Control de proyecto.

A continuación se incluye el calendario de entregas completadas. El símbolo ☐ indica el día en el que se trabajó en el entregable, mientras que el símbolo ☒ indica que el trabajo se acabó en ese día.

Entregable	_ /05/2023													
	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Plan de trabajo en Excel	<input checked="" type="checkbox"/>													
Documento de requerimientos de Negocio en formato tabular simple	<input checked="" type="checkbox"/>													
Requerimientos no funcionales	<input checked="" type="checkbox"/>													
Reglas de Negocio	<input checked="" type="checkbox"/>													
Historias con casos de uso														
Happy Path	<input checked="" type="checkbox"/>													
Rutas Alternas para Reglas de Negocio	<input checked="" type="checkbox"/>													
Bosquejo de Arquitectura con los nombre de los componentes con:														
Descripción breve del componente con tecnología de soporte	<input checked="" type="checkbox"/>													
Tabla con los componentes de la Arquitectura con:														
Responsabilidades en cuanto a requerimientos funcionales	<input checked="" type="checkbox"/>													
Responsabilidades en cuanto a requerimientos no funcionales	<input checked="" type="checkbox"/>													
Tabla de detalle de plan de trabajo con bitácora diaria de avance de las actividades y desviaciones del proyecto	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>											
Batería de pruebas funcionales es por servicio de negocio y en general:														
Descripción del "Steady State" para cada caso	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>											
Script de carga de Bases de datos para "Steady State"	<input checked="" type="checkbox"/>													
Transacciones habituales	<input type="checkbox"/>	<input checked="" type="checkbox"/>												
Datos a inyectar	<input checked="" type="checkbox"/>													
Respuesta esperada y forma de verificación	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>									

Proceso automatizado de detección de cumplimiento	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>										
Descripción de carga para la prueba no funcional																	
Volumetrics por tipo de transacción para cada servicio								<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
Condiciones de "Steady State" y Script de carga de Bases de Datos								<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>					
Volumetrics por tipo de transacción y modalidad para servicio integrado													<input type="checkbox"/>	<input checked="" type="checkbox"/>			
Bitácora de Entregas y desviaciones de integración con																	
Entregas y versiones de componentes																	<input checked="" type="checkbox"/>
Instalación de la versión "Steady" para cada entrega parcial																	<input checked="" type="checkbox"/>
Cumplimiento con la integración																	<input checked="" type="checkbox"/>
Documento de Arquitectura del servicio con (WS de Almacén)																	
Relación a Reglas y Requerimientos de Negocio	<input checked="" type="checkbox"/>																
Pruebas funcionales														<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Pruebas no funcionales														<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Documento de Arquitectura de servicio con (WS de Pagos)																	
Relación a Reglas y Requerimientos de Negocio	<input checked="" type="checkbox"/>																
Pruebas funcionales														<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Pruebas no funcionales														<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Documento de Arquitectura de servicio con (WS de Entregas)																	
Relación a Reglas y Requerimientos de Negocio	<input checked="" type="checkbox"/>																
Pruebas funcionales														<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Pruebas no funcionales														<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Estresador																	
Script, Proceso y Pojos de estrés														<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Tabla con resultados de las Pruebas de estrés para las versiones estables														<input type="checkbox"/>	<input checked="" type="checkbox"/>		

2 Requerimientos De negocio

Con base en la información sobre el proceso de compra-venta de libros en una tienda virtual en línea, se han establecido reglas de negocio que se consideran requisitos funcionales para la implementación de la tienda en línea.

Estas reglas de negocio definen las funcionalidades y características específicas que deben estar presentes en la tienda en línea para garantizar un proceso de compra-venta eficiente y satisfactorio. Estos requisitos funcionales pueden incluir aspectos como la capacidad de búsqueda y filtrado de libros, la visualización de detalles del producto, la gestión de carritos de compra, el proceso de pago seguro, la generación de confirmaciones de pedido, la gestión de inventario, entre otros aspectos relevantes para la operación de la tienda en línea.

Al establecer estas reglas de negocio como requisitos funcionales, nos aseguramos de que la implementación de la tienda en línea cumpla con los criterios y expectativas necesarios para proporcionar una experiencia de compra fluida y satisfactoria para los usuarios.

2.1 Requerimientos Funcionales

La aplicación a desarrollar debe cumplir con las siguientes reglas de negocio:

RN1.- Atender solicitudes de compra de usuarios registrados en el medio de pago.

RN2.- Aplicación para adquirir exclusivamente artículos registrados en un almacén por medio de su clave ISBN de 13 caracteres.

RN3.- Cada transacción se lleva a cabo con un id del medio de pago y por una cantidad de ejemplares de uno o varios ISBNs.

RN4.- No debe permitir la venta de ejemplares no disponibles.

RN5.- No debe permitir la venta si no se ha registrado en medio de pago o el medio de pago carece de fondos.

RN6.- La entrega del material comprado se lleva a cabo por medio de un servicio de paquetería, identificando la entrega con la empresa de venta de libros en línea y el id de la factura correspondiente.

RN7.- Los usuarios tienen la opción de hacer pagos a meses con intereses y a meses sin intereses.

RN8.- La decisión de otorgar a un cliente un descuento se hará con base en su puntaje crediticio. Un usuario con una BC score mayor a 600 podrá pagar a meses sin intereses.

RN9.- Un usuario con un puntaje crediticio menor a 600 no tendrá descuento, si su puntaje es menor a 700 y mayor a 600 tendrá un descuento de 25% y finalmente si el usuario tiene un puntaje de 700 a 800 este tendrá un descuento de 50%.

2.2 Requerimientos No Funcionales

Además de los requisitos funcionales, también existen requisitos no funcionales en este caso, que se refieren al rendimiento de la aplicación. El requisito no funcional fundamental será en torno al rendimiento de la aplicación. Este requisito garantiza una experiencia fluida y sin problemas para los usuarios, independientemente del volumen de actividad. Se busca que la

aplicación pueda manejar eficientemente el alto volumen esperado, manteniendo un rendimiento óptimo en todo momento.

RNF1.- Debe soportar 100 transacciones por minuto, incluyendo altas de artículos, modificaciones de stock y modificaciones en el medio de pago.

3 Historias de Usuario

Con el fin de proporcionar al equipo de desarrollo una visión de las tareas necesarias desde la perspectiva de los usuarios de la aplicación, se presentan las siguientes historias de usuario. Estas historias representan el funcionamiento requerido por cada usuario involucrado en el uso de la aplicación. Además, se incluyen las rutas a seguir cuando alguno de los requisitos de los usuarios no se cumple.

3.1 Happy Paths

3.1.1 Compra en línea

- a. Realizar una compra exitosa:
 - i. El usuario busca el libro por su ISBN en la tienda en línea.
 - ii. El libro es encontrado y se muestra la información del producto.
 - iii. El usuario agrega el libro al carrito de compra.
 - iv. El usuario selecciona un medio de pago válido.
 - v. El usuario confirma la compra y se procesa el pago.
 - vi. El libro se retira del inventario.
 - vii. Se genera una confirmación de compra para el usuario.
 - viii. La transacción se completó con éxito.
- b. Actualizar el stock de libros:
 - i. El administrador de la tienda accede al panel de administración.
 - ii. El administrador verifica el nivel de stock de los libros.
 - iii. Se registra la entrada de nuevos libros al inventario.
 - iv. El stock se actualiza correctamente en la base de datos.
 - v. Los usuarios pueden ver el nuevo stock disponible al buscar los libros.

3.2 Rutas alternas

Si alguno de los siguientes casos ocurre, la aplicación no permitirá que se realice una transacción de compra-venta. Los casos que interrumpen una transacción son los siguientes:

1. ISBN inexistente: Si el ISBN (International Standard Book Number) no está registrado en la base de datos de la tienda, la transacción no puede continuar.
2. Medio de pago inexistente: Si el medio de pago seleccionado por el usuario no está disponible en la aplicación, la transacción no puede ser procesada.
3. Stock insuficiente: Si no hay suficiente stock del libro deseado para satisfacer la compra, la transacción no puede ser completada.
4. Fondos insuficientes: Si el usuario no cuenta con los fondos necesarios en su cuenta o medio de pago para realizar la compra, la transacción no puede llevarse a cabo.

En caso de que cualquiera de los casos anteriores se cumpla, la transacción será anulada y será necesario reiniciar el proceso desde el principio.

4 Documento de Arquitectura

En esta sección se detalla la arquitectura global de la aplicación, incluyendo el patrón arquitectónico empleado y los componentes de cada nivel involucrado.

4.1 Descripción general de la arquitectura propuesta

La arquitectura utilizada en este aplicativo sigue el patrón de arquitectura por capas. Se compone de la capa de base de datos, que contiene dos bases de datos distintas para almacenar los datos relacionados con almacén, pago y envío. La capa de persistencia está representada por los web services, los cuales acceden y modifican los datos de la base de datos. Por último, la capa de negocios se implementa mediante BPEL, donde se integran las reglas de negocio para garantizar el correcto funcionamiento de la tienda en línea.

A diferencia de la arquitectura por capas convencional, en este caso no se incluye una capa de presentación con una interfaz gráfica para el usuario. En su lugar, las pruebas se realizan a través de POJOs (Plain Old Java Objects) y el estresador.

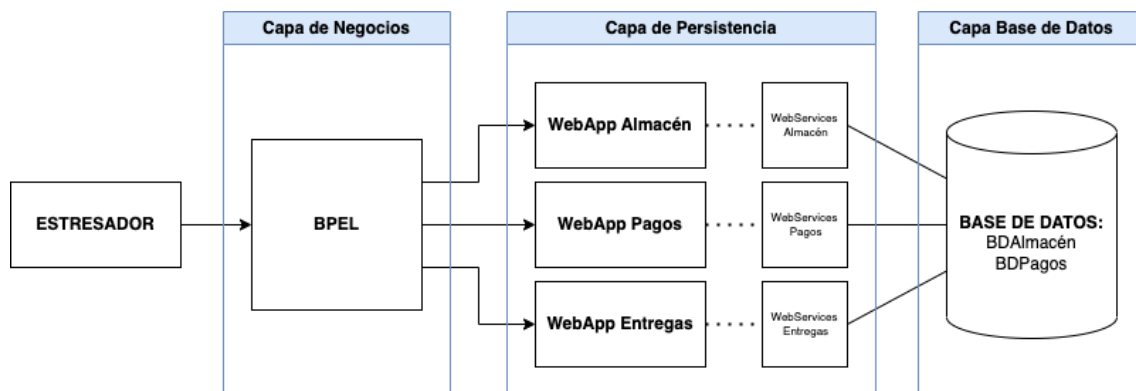


Diagrama de Arquitectura por Capas

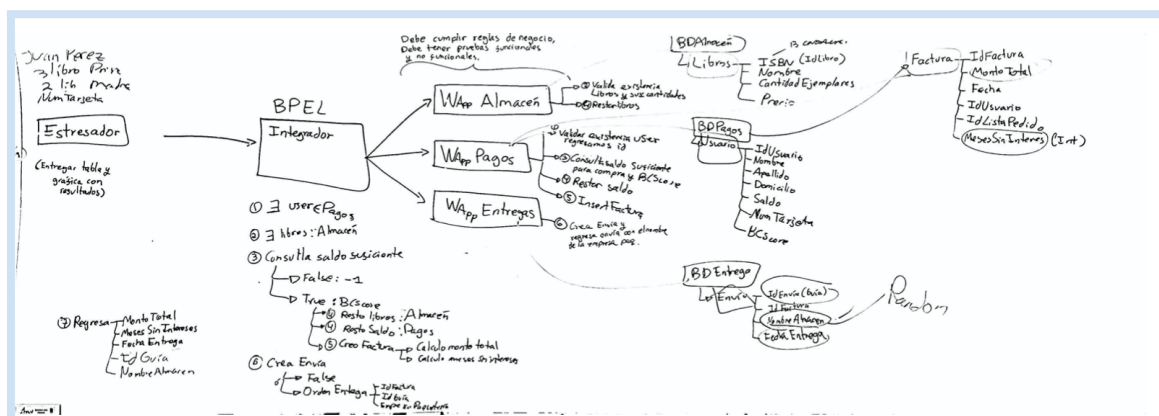


Diagrama de Planeación de Arquitectura

5 Responsabilidades de los Componentes de la Arquitectura y relación con las reglas de negocio.

En esta sección se muestra en una tabla la relación entre los requerimientos funcionales y no funcionales con su individuo responsable.

Requerimientos funcionales	Responsable
RN1.- Atender solicitudes de compra de usuarios registrados en el medio de pago.	Desarrollador de WS de Cobro
RN2.- Aplicación para adquirir exclusivamente artículos registrados en un almacén por medio de su clave ISBN de 13 caracteres.	Desarrollador de WS de Almacén
RN3.- Cada transacción se lleva a cabo con un id del medio de pago y por una cantidad de ejemplares de un solo ISBN.	Integrador (BPEL)
RN4.- No debe permitir la venta de ejemplares no disponibles.	Desarrollador de WS de Almacén
RN5.- No debe permitir la venta si no se ha registrado en medio de pago o el medio de pago carece de fondos.	Desarrollador de WS de Almacén
RN6.- La entrega del material comprado se lleva a cabo por medio de un servicio de paquetería, identificando la entrega con la empresa de venta de libros en línea y el id de la factura correspondiente.	Desarrollador de WS de Envíos
RN7.- Los usuarios tienen la opción de hacer pagos a meses con intereses y a meses sin intereses.	Desarrollador de WS de Pagos
RN8.- La decisión de otorgar a un cliente un descuento se hará con base en su puntaje crediticio. Un usuario con un puntaje crediticio menor a 600 no tendrá descuento, si su puntaje es menor a 700 y mayor a 600 tendrá un descuento de 25% y finalmente si el usuario tiene un puntaje de 700 a 800 este tendrá un descuento de 50%.	Integrador (BPEL)
RN9.- Un usuario con un BC score menor a 600 tendrá que pagar a meses con intereses o de contado, con una tasa de interés que dependa de su BC score. La fórmula para calcular la tasa de interés será: $10 \times (1 - \frac{BC-456}{456})\%$.	Integrador (BPEL)

Requerimientos no funcionales	Responsable
RNF1.- Debe soportar 100 transacciones por minuto, incluyendo altas de artículos, modificaciones de stock y modificaciones en el medio de pago.	Estresador

6 Control de avance y desviaciones

La tabla de avance muestra el progreso del proyecto, el cual se dividió en tres entregas para pruebas de estrés. La primera entrega se enfocó en el desarrollo de servicios web utilizando

loopback sin acceder a la base de datos. En la segunda entrega se desarrolló una versión de servicios web que sí accedían a la base de datos.

	Loopback	Funcional
WS Almacén	_/05/2023	_/05/2023
WS Pago	_/05/2023	_/05/2023
WS Entregas	_/05/2023	_/05/2023
BPEL e Integración	_/05/2023	_/05/2023

7 Documentación de pruebas funcionales

7.1 Pruebas del BPEL

7.1.1 RN1.- Intento de compra con un cliente que no está en la BD

7.1.2 RN2.- Intento de comprar de un libro que no existe

7.1.3 RN4.- Intento de compra de un libro sin suficiente stock

7.1.4 RN5. - Intento de compra con fondos insuficientes

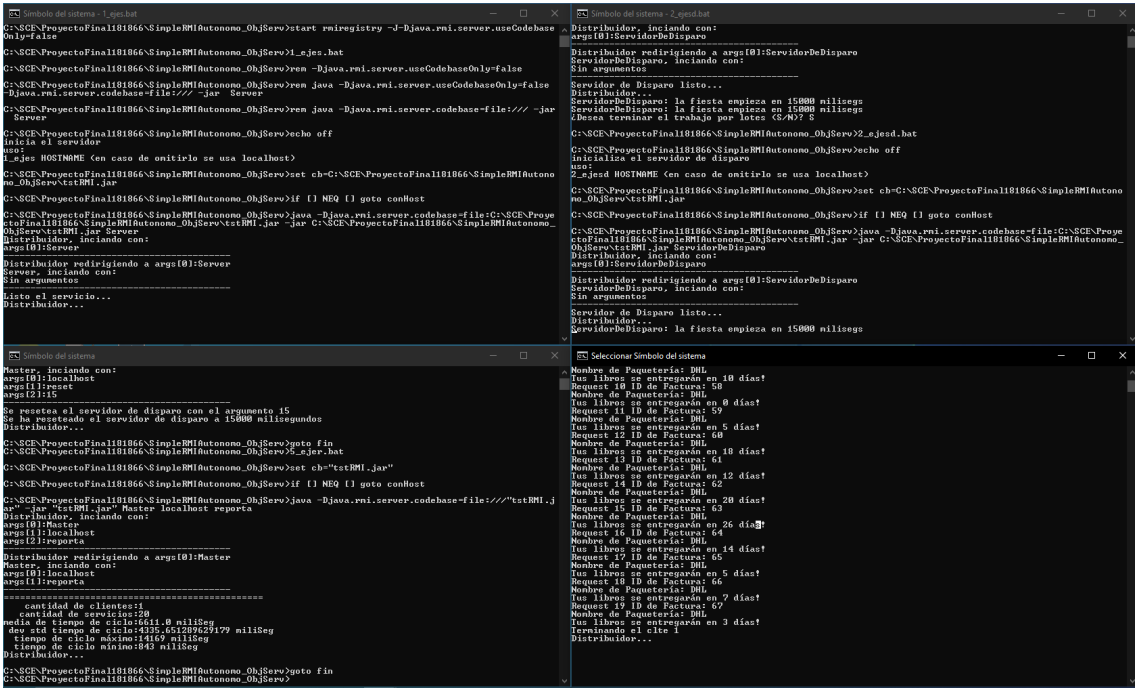
7.1.5 Compra Exitosa (RN3, RN6, RN7, RN8, RN9)

{Para cada historia de usuario mostrar las evidencias de la realización de la ejecución del pojo (con un pantallazo basta, por ello hay que adecuar el pojo para que muestre lo que está haciendo. Reportar: estado inicial, deltas de afectación en stock, montos y entregas, Si lo desean pueden hacer un volcado en texto del contenido de los registros correspondientes de las tablas en la base de datos con el antes y el después. Además se debe reportar en forma de check-list el avance en el cumplimiento de las pruebas funcionales, en términos de las reglas de negocio y las historias de usuario.)}

- Descripción del “Steady State” para cada caso;
- Script de carga de Bases de datos para “Steady State”;
- Transacciones habituales
- Datos a inyectar;
- Respuesta esperada y forma de verificación;
- Proceso automatizado de detección de cumplimiento

8 Documentación de pruebas no funcionales

Ver secciones 10, 11, y 12 para los resultados de las pruebas.



WS	Transacción	Clientes	# de Transacciones
WS Almacén	getCantidad()	10	100
	restarLibros()	10	100
WS Pago	validarUsuario()	10	100
	restarSaldo()	10	100
	checkMonto()	10	100
	crearFactura()	10	100
	vincularLibro()	10	100
	crearEnvio()	10	100
WS Entregas	crearEnvio()	10	100

9 Arquitectura y versiones

El desarrollo se hizo en dos iteraciones debido a limitaciones de tiempo. La siguiente tabla muestra las fechas de entrega de pruebas funcionales y no funcionales, con sus versiones y cumplimientos de integración:

Versión	Cumplimiento de integración	Pruebas funcionales	Pruebas no funcionales
WS Almacén v.1	Loopback	20/05/2023	20/05/2023
WS Pago v.1	Loopback	20/05/2023	20/05/2023

WS Entregas v.1	Loopback	20/05/2023	20/05/2023
BPEL e Integración v.1	Loopback	_/05/2023	_/05/2023
WS Almacén v.f	Funcional	24/05/2023	24/05/2023
WS Pago v.f	Funcional	24/05/2023	24/05/2023
WS Entregas v.f	Funcional	24/05/2023	24/05/2023
BPEL e Integración v.f	Funcional	_/05/2023	_/05/2023

10 Web Service de Almacén

Este web service permite que la aplicación consulte cuántos libros hay disponibles de cierto título y también permite que se resten de la base de datos los libros que se desean comprar. Estas funcionalidades nos permitirán implementar la regla de negocio **RN4** en el BPEL.

Prueba getCantidad():

Regresa un entero de cuantos ejemplares hay dado un ISBN.

Caso ISBN válido:

getCantidad Method invocation

Method parameter(s)

Type	Value
java.lang.String	a234567898765

Method returned

int : "7"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:getCantidad xmlns:ns2="http://wsalmacen/">
      <isbn>a234567898765</isbn>
    </ns2:getCantidad>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:getCantidadResponse xmlns:ns2="http://wsalmacen/">
      <return>7</return>
    </ns2:getCantidadResponse>
  </S:Body>
</S:Envelope>
```

Caso ISBN no válido:

getCantidad Method invocation

Method parameter(s)

Type	Value
java.lang.String	isbnQueNoEsta

Method returned

int : "-1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:getCantidad xmlns:ns2="http://wsalmacen/">
      <isbn>isbnQueNoEsta</isbn>
    </ns2:getCantidad>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:getCantidadResponse xmlns:ns2="http://wsalmacen/">
      <return>-1</return>
    </ns2:getCantidadResponse>
  </S:Body>
</S:Envelope>
```

Prueba restarLibros():

Regresa un 1, -10 o 0 “status” que indica si pudo restar libros o no. La función permite que se resten libros que no hay ya que siempre se consultará cuántos libros hay disponibles antes de utilizar esta función. Un 1 indica éxito, un -10 indica que no hay stock suficiente, y un 0 indica que no existe el isbn.

Caso 1:

restarLibros Method invocation

Method parameter(s)

Type	Value
java.lang.String	a234567898765
int	1

Method returned

int : "1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibros xmlns:ns2="http://wsalmacen/">
      <isbn>a234567898765</isbn>
      <cantidad>1</cantidad>
    </ns2:restarLibros>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibrosResponse xmlns:ns2="http://wsalmacen/">
      <return>1</return>
    </ns2:restarLibrosResponse>
  </S:Body>
</S:Envelope>
```

Caso -10:

restarLibros Method invocation

Method parameter(s)

Type	Value
java.lang.String	b234567898365
int	100

Method returned

int : "-10"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibros xmlns:ns2="http://wsalmacen/">
      <isbn>b234567898365</isbn>
      <cantidad>100</cantidad>
    </ns2:restarLibros>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibrosResponse xmlns:ns2="http://wsalmacen/">
      <return>-10</return>
    </ns2:restarLibrosResponse>
  </S:Body>
</S:Envelope>
```

Caso 0:

restarLibros Method invocation

Method parameter(s)

Type	Value
java.lang.String	isbnQueNoExiste
int	1

Method returned

int : "0"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibros xmlns:ns2="http://wsalmacen/">
      <isbn>isbnQueNoExiste</isbn>
      <cantidad>1</cantidad>
    </ns2:restarLibros>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarLibrosResponse xmlns:ns2="http://wsalmacen/">
      <return>0</return>
    </ns2:restarLibrosResponse>
  </S:Body>
</S:Envelope>
```

Pruebas POJOs:

Caso 1: Se restan exitosamente unidades ordenadas.

```
Java DB Database Process x GlassFish Server x Retriever Output x SQL 1 execution x Pojo_Almacen (run-single) x
ant -f C:\SCE\PojosPrueba\Pojo_Almacen -Djavac.includes=pojo_almacen/Pojo_Almacen.java -Dnb.internal
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
wsimport-init:
wsimport-client-WSAlmacen:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Almacen\build\classes
compile-single:
run-single:
Probamos con Libro con ISBN: e204567898765.
Vamos a ordenar 5 ejemplares.
Ejecutamos getCantidad: 30

Hacemos una orden de 5 libros.

Ejecutamos getCantidad:25
Se restarán 5 libros de id e204567898765
BUILD SUCCESSFUL (total time: 1 second)
```

Caso 2: No se restan ejemplares ya que se ordenan más de los que hay en existencias.

```
Java DB Database Process x GlassFish Server x Retriever Output x SQL 1 execution x Pojo_Almacen (run-single) x
ant -f C:\SCE\PojosPrueba\Pojo_Almacen -Djavac.includes=pojo_almacen/Pojo_Almacen.java -Dnb.internal
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
wsimport-init:
wsimport-client-WSAlmacen:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Almacen\build\classes
compile-single:
run-single:
Probamos con Libro con ISBN: e204567898765.
Vamos a ordenar 5000 ejemplares.
Ejecutamos getCantidad: 30
No hay suficiente stock disponible
BUILD SUCCESSFUL (total time: 1 second)
```

Caso 3: No existe el ISBN dado.

```
Java DB Database Process x GlassFish Server x Retriever Output x SQL 1 execution x Pojo_Almacen (run-single) x
ant -f C:\SCE\PojosPrueba\Pojo_Almacen -Djavac.includes=pojo_almacen/Pojo_Almacen.java -Dnb.internal
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Almacen\build\build-jar.properties
wsimport-init:
wsimport-client-WSAlmacen:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Almacen\build\classes
compile-single:
run-single:
Probamos con Libro con ISBN: isbnInvalido.
Vamos a ordenar 5 ejemplares.
Ejecutamos getCantidad: -1
El libro no existe en la base de datos.
BUILD SUCCESSFUL (total time: 1 second)
```

Pruebas No Funcionales:

Transacción	# Clientes	#	Tiempo	Tiempo	Tiempo	Tiempo (s)	Transacc.
-------------	------------	---	--------	--------	--------	------------	-----------

		Transacciones	Medio de Ciclo (ms)	Ciclo Máximo (ms)	Ciclo Mínimo (ms)		por Minuto
getCantidad()	10	100	217.0	1575.0	21.0	217	276
restarLibros()	10	100	220.0	162.0	20.0	220	272

Evidencia Pruebas No Funcionales:

```

Proceso 4
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:220.0 miliSeg
dev std tiempo de ciclo:162.06991286681392 miliSeg
tiempo de ciclo máximo:1641 miliSeg
tiempo de ciclo mínimo:20 miliSeg
Distribuidor...
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>goto fin
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>5_ejer.bat
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>set cb="tstRMI.jar"
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>if [] NEQ [] goto conHost
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar" -j
r "tstRMI.jar" Master localhost reporta
Distribuidor, inciendo con:
args[0]:Master
args[1]:localhost
args[2]:reporta
-----
Distribuidor redirigiendo a args[0]:Master
Master, inciendo con:
args[0]:localhost
args[1]:reporta
-----
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:217.0 miliSeg
dev std tiempo de ciclo:168.72780764614075 miliSeg
tiempo de ciclo máximo:1575 miliSeg
tiempo de ciclo mínimo:21 miliSeg
Distribuidor...
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>goto fin
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>

```

```

Proceso 4
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:209.0 miliSeg
dev std tiempo de ciclo:158.4640427758072 miliSeg
tiempo de ciclo máximo:1606 miliSeg
tiempo de ciclo mínimo:36 miliSeg
Distribuidor...
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>goto fin
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>5_ejer.bat
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>set cb="tstRMI.jar"
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>if [] NEQ [] goto conHost
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar" -j
r "tstRMI.jar" Master localhost reporta
Distribuidor, inciendo con:
args[0]:Master
args[1]:localhost
args[2]:reporta
-----
Distribuidor redirigiendo a args[0]:Master
Master, inciendo con:
args[0]:localhost
args[1]:reporta
-----
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:220.0 miliSeg
dev std tiempo de ciclo:162.06991286681392 miliSeg
tiempo de ciclo máximo:1641 miliSeg
tiempo de ciclo mínimo:20 miliSeg
Distribuidor...
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>goto fin
C:\SCE_181335\ProyectoFinal\SimpleRMIAutonomo>

```

11 Web Service de Pago

Este web service permite que la aplicación valide si el usuario que está haciendo la compra existe en la base de datos, validar su saldo para después verificar si tiene suficientes fondos para pagar lo que debe, y restar de su saldo en el sistema de pagos después de hacer una compra. Por lo tanto ese web service cumple con **RN1** y **RN5**.

Prueba validarUsuario():

Dado un número de tarjeta, regresa el id del cliente asociado si existe dicha tarjeta en la base de datos. Si no existe, regresa -1.

Prueba Válida:

validarUsuario Method invocation

Method parameter(s)

Type	Value
java.lang.String	12345

Method returned

java.lang.Integer : "1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:validarUsuario xmlns:ns2="http://wspagos/">
      <numTarjeta>12345</numTarjeta>
    </ns2:validarUsuario>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:validarUsuarioResponse xmlns:ns2="http://wspagos/">
      <return>1</return>
    </ns2:validarUsuarioResponse>
  </S:Body>
</S:Envelope>
```

Prueba -1:

validarUsuario Method invocation

Method parameter(s)

Type	Value
java.lang.String	noexiste

Method returned

java.lang.Integer : "-1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:validarUsuario xmlns:ns2="http://wspagos/">
      <numTarjeta>noexiste</numTarjeta>
    </ns2:validarUsuario>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:validarUsuarioResponse xmlns:ns2="http://wspagos/">
      <return>-1</return>
    </ns2:validarUsuarioResponse>
  </S:Body>
</S:Envelope>
```

Prueba restarSaldo():

Dado un número de tarjeta y un monto, resta el monto del saldo del número de tarjeta dado.

Prueba Válida:

restarSaldo Method invocation

Method parameter(s)

Type	Value
java.lang.Double	100
java.lang.String	12345

Method returned

int : "1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarSaldo xmlns:ns2="http://wspagos/">
      <monto>100.0</monto>
      <numTarjeta>12345</numTarjeta>
    </ns2:restarSaldo>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:restarSaldoResponse xmlns:ns2="http://wspagos/">
      <return>1</return>
    </ns2:restarSaldoResponse>
  </S:Body>
</S:Envelope>
```

Prueba checkMonto():

Dado un número de tarjeta y un monto, regresa 1 si el usuario puede pagar dicho monto y -1 si no puede.

Prueba 1:

checkMonto Method invocation

Method parameter(s)

Type	Value
double	1
java.lang.String	12345

Method returned

java.lang.Integer : "1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:checkMonto xmlns:ns2="http://wspagos/">
      <monto>1.0</monto>
      <numTarjeta>12345</numTarjeta>
    </ns2:checkMonto>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:checkMontoResponse xmlns:ns2="http://wspagos/">
      <return>1</return>
    </ns2:checkMontoResponse>
  </S:Body>
</S:Envelope>
```

Prueba -1:

checkMonto Method invocation

Method parameter(s)

Type	Value
double	99999999
java.lang.String	12345

Method returned

java.lang.Integer : "1"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:checkMonto xmlns:ns2="http://wspagos/">
      <monto>9.99999999</monto>
      <numTarjeta>12345</numTarjeta>
    </ns2:checkMonto>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:checkMontoResponse xmlns:ns2="http://wspagos/">
      <return>1</return>
    </ns2:checkMontoResponse>
  </S:Body>
</S:Envelope>
```

Prueba crearFactura():

Dado un número de tarjeta, crea una factura nueva con dicho número y regresa el id de la factura.

Prueba Exitosa:

crearFactura Method invocation

Method parameter(s)

Type	Value
java.lang.String	12345

Method returned

java.lang.Integer : "3"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOA<br><SOAP-ENV:Header/><S:Body><br><ns2:crearFactura xmlns:ns2="http://wspagos/"><br><numTarjeta>12345</numTarjeta><br></ns2:crearFactura><br></S:Body><br></S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOA<br><SOAP-ENV:Header/><S:Body><br><ns2:crearFacturaResponse xmlns:ns2="http://wspagos/"><br><return>3</return><br></ns2:crearFacturaResponse><br></S:Body><br></S:Envelope>
```

Prueba vincularLibro():

Dado un id factura, una cantidad, y un ISBN actualiza la factura asociado con el monto total de la transacción y genera una entrada nueva en la tabla “ListaLibros”.

Prueba True:

vincularLibro Method invocation

Method parameter(s)

Type	Value
int	1
int	10
java.lang.String	a234567898765

Method returned

java.lang.Boolean : "true"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOA<br><SOAP-ENV:Header/><S:Body><br><ns2:vincularLibro xmlns:ns2="http://wspagos/"><br><idFactura>1</idFactura><br><cantidad>10</cantidad><br><isbn>a234567898765</isbn><br></ns2:vincularLibro><br></S:Body><br></S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOA<br><SOAP-ENV:Header/><S:Body><br><ns2:vincularLibroResponse xmlns:ns2="http://wspagos/"><br><return>true</return><br></ns2:vincularLibroResponse><br></S:Body><br></S:Envelope>
```

Prueba False:

vincularLibro Method invocation

Method parameter(s)

Type	Value
int	100
int	10
java.lang.String	noexiste

Method returned

java.lang.Boolean : "false"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SO<br><SOAP-ENV:Header/><br><S:Body><br><ns2:vincularLibro xmlns:ns2="http://vspagos/"><br><idFactura>100</idFactura><br><cantidad>10</cantidad><br><isbn>noexiste</isbn><br></ns2:vincularLibro><br></S:Body><br></S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SO<br><SOAP-ENV:Header/><br><S:Body><br><ns2:vincularLibroResponse xmlns:ns2="http://vspagos/"><br><return>false</return><br></ns2:vincularLibroResponse><br></S:Body><br></S:Envelope>
```

Pruebas POJOs:

```
Probamos validarUsuario
Para el numero de tarjeta '12345': Exito! - el id del usuario es 1
Para el numero de tarjeta 'tarjetaInvalida': Error - no existe usuario con esa tarjeta
Probamos checkMonto
Para el numero de tarjeta '12345' y el monto '1.0': Exito! - el saldo es suficiente para el monto
Para el numero de tarjeta '12345' y el monto '9999999.0': Error - el monto excede el saldo
Probamos crearFactura
Para el numero de tarjeta '12345': Exito! - Se creó la factura y se le asignó el id 5
Probamos restarSaldo
Para el numero de tarjeta '12345' y el monto '1.0': Exito! - el saldo es suficiente para el monto y restamos el monto
Para el numero de tarjeta '12345' y el monto '9999999.0': Error - el monto excede el saldo
Probamos vincularLibro
Para el idFactura '1', la cantidad '3' y el isbn 'a234567890765': Exito! - se logró vincular el libro
Para el idFactura '-1', la cantidad '3' y el isbn 'isbnInvalido': Error - idFactura o isbn no existe
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
F:\11_Semestre\Sistemas_De_Comercio_Electronico\E-Commerce-Final-Project\SimpleRMIAutonomo_ObjServ\SimpleRMIAutonomo_ObjServ>java -Djava.rmi.server.co
debase=file:/// "tstRMI.jar" -jar "tstRMI.jar" Master localhost reporta
Distribuidor, iniciando con:
args[0]:Master
args[1]:localhost
args[2]:reporta
=====
Distribuidor redirigiendo a args[0]:Master
Master, iniciando con:
args[0]:localhost
args[1]:reporta
=====
=====
cantidad de clientes:8
cantidad de servicios:40
media de tiempo de ciclo:111285.0 miliSeg
dev std tiempo de ciclo:90322.53414604852 miliSeg
tiempo de ciclo máximo:586133 miliSeg
tiempo de ciclo mínimo:377 miliSeg
Distribuidor...
F:\11_Semestre\Sistemas_De_Comercio_Electronico\E-Commerce-Final-Project\SimpleRMIAutonomo_ObjServ\SimpleRMIAutonomo_ObjServ>goto fin
F:\11_Semestre\Sistemas_De_Comercio_Electronico\E-Commerce-Final-Project\SimpleRMIAutonomo_ObjServ\SimpleRMIAutonomo_ObjServ>
```

Pruebas No Funcionales:

Transacción	# Clientes	# Transacciones	Tiempo Medio de Ciclo (ms)	Tiempo Ciclo Máximo (ms)	Tiempo Ciclo Mínimo (ms)	Tiempo (s)	Transacc. por Minuto
validarUsuario()	10	100	667.0	2449.0	131.0	667	90
restarSaldo()	10	100	643.0	2139.0	152.0	643	93
checkMonto()	10	100	631.0	2632.0	112.0	631	95
crearFactura()	10	100					
vincularLibro()	10	100	633.0	2040.0	143.0	633	95

Evidencia Pruebas No Funcionales:

```

C:\Proceso 4
El dispositivo no está listo.
C:\SCE\SimpleRMIAutonomo\batsDeInicio>cd ..
C:\SCE\SimpleRMIAutonomo>5_ejer.bat
C:\SCE\SimpleRMIAutonomo>set ch="tstRMI.jar"
C:\SCE\SimpleRMIAutonomo>if [!] NEQ [!] goto conHost
C:\SCE\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar" -jar "tstRMI.jar" Master
localhost reporta
Distribuidor, iniciando con:
args[0]:Master
args[1]:localhost
args[2]:reporta
Distribuidor redirigiendo a args[0]:Master
Master, iniciando con:
args[0]:localhost
args[1]:reporta
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:667.0 miliSeg
dev std tiempo de ciclo:255.35598422335568 miliSeg
tiempo de ciclo máximo:2449 miliSeg
tiempo de ciclo mínimo:131 miliSeg
Distribuidor...
C:\SCE\SimpleRMIAutonomo>goto fin
C:\SCE\SimpleRMIAutonomo>

```

```
Proceso 4
El dispositivo no está listo.
C:\SCE\SimpleRMIAutonomo\batsDeInicio>cd ..
C:\SCE\SimpleRMIAutonomo>5_ejer.bat
C:\SCE\SimpleRMIAutonomo>set ch="tstRMI.jar"
C:\SCE\SimpleRMIAutonomo>if [] NEQ [] goto conHost
C:\SCE\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar -jar "tstRMI.jar" Master
localhost reporta
Distribuidor, iniciando con:
args[0]:Master
args[1]:localhost
args[2]:reporta
-----
Distribuidor redirigiendo a args[0]:Master
Master, iniciando con:
args[0]:localhost
args[1]:reporta
-----
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:643.0 miliSeg
dev std tiempo de ciclo:241.02357428966025 miliSeg
tiempo de ciclo máximo:2139 miliSeg
tiempo de ciclo mínimo:152 miliSeg
Distribuidor...
C:\SCE\SimpleRMIAutonomo>goto fin
C:\SCE\SimpleRMIAutonomo>_
```

```
Proceso 4
El dispositivo no está listo.
C:\SCE\SimpleRMIAutonomo\batsDeInicio>cd ..
C:\SCE\SimpleRMIAutonomo>5_ejer.bat
C:\SCE\SimpleRMIAutonomo>set ch="tstRMI.jar"
C:\SCE\SimpleRMIAutonomo>if [] NEQ [] goto conHost
C:\SCE\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar -jar "tstRMI.jar" Master
localhost reporta
Distribuidor, iniciando con:
args[0]:Master
args[1]:localhost
args[2]:reporta
-----
Distribuidor redirigiendo a args[0]:Master
Master, iniciando con:
args[0]:localhost
args[1]:reporta
-----
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:631.0 miliSeg
dev std tiempo de ciclo:254.1404776543524 miliSeg
tiempo de ciclo máximo:2632 miliSeg
tiempo de ciclo mínimo:112 miliSeg
Distribuidor...
C:\SCE\SimpleRMIAutonomo>goto fin
C:\SCE\SimpleRMIAutonomo>5
```

```
Proceso 4
El dispositivo no está listo.
C:\SCE\SimpleRMIAutonomo\batsDeInicio>cd ..
C:\SCE\SimpleRMIAutonomo>5_ejer.bat
C:\SCE\SimpleRMIAutonomo>set ch="tstRMI.jar"
C:\SCE\SimpleRMIAutonomo>if [] NEQ [] goto conHost
C:\SCE\SimpleRMIAutonomo>java -Djava.rmi.server.codebase=file:///tstRMI.jar -jar "tstRMI.jar" Master
localhost reporta
Distribuidor, iniciando con:
args[0]:Master
args[1]:localhost
args[2]:reporta
Distribuidor redirigiendo a args[0]:Master
Master, iniciando con:
args[0]:localhost
args[1]:reporta
=====
cantidad de clientes:10
cantidad de servicios:1000
media de tiempo de ciclo:633.0 miliSeg
dev std tiempo de ciclo:249.15486177304496 miliSeg
tiempo de ciclo máximo:2040 miliSeg
tiempo de ciclo mínimo:143 miliSeg
Distribuidor...
C:\SCE\SimpleRMIAutonomo>goto fin
C:\SCE\SimpleRMIAutonomo>_
```

12 Web Service de Entregas

Este web service permite que la aplicación de de alta entregas en la tabla “ENVÍO” de la base de datos. Si se proporciona un IDFactura válido se genera un renglón en dicha base de datos. Por lo tanto este web service permite que se cumpla la regla de negocio **RN6**.

Prueba crearEnvio():

Dado un IDFactura se genera un renglón en la tabla “ENVÍO” de la base de datos que está vinculado a la factura proporcionada.

Caso Exitoso:

crearEnvio Method invocation

Method parameter(s)

Type	Value
int	1

Method returned

java.lang.String : "ID de Factura: 1 Nombre de Paqueteria: DHL Tus libros se entregarán en 8 días!"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvio xmlns:ns2="http://wsentregas/">
      <idFactura>1</idFactura>
    </ns2:crearEnvio>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvioResponse xmlns:ns2="http://wsentregas/">
      <return>ID de Factura: 1
      Nombre de Paqueteria: DHL
      Tus libros se entregarán en 8 días!</return>
    </ns2:crearEnvioResponse>
  </S:Body>
</S:Envelope>
```

Caso existente:

crearEnvio Method invocation

Method parameter(s)

Type	Value
int	1

Method returned

java.lang.String : "YAESTA"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvio xmlns:ns2="http://wsentregas/">
      <idFactura>1</idFactura>
    </ns2:crearEnvio>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvioResponse xmlns:ns2="http://wsentregas/">
      <return>YAESTA</return>
    </ns2:crearEnvioResponse>
  </S:Body>
</S:Envelope>
```

Caso factura inexistente:

crearEnvio Method invocation

Method parameter(s)

Type	Value
int	-1

Method returned

java.lang.String : "NOEXISTE"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvio xmlns:ns2="http://wsentregas/">
      <idFactura>-1</idFactura>
    </ns2:crearEnvio>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:crearEnvioResponse xmlns:ns2="http://wsentregas/">
      <return>NOEXISTE</return>
    </ns2:crearEnvioResponse>
  </S:Body>
</S:Envelope>
```

Pruebas POJOs:

Caso 1: Se genera una entrega exitosamente.


```

Java DB Database Process x GlassFish Server x Retriever Output x SQL 3 execution x Pojo_Entregas (run-single) x SQL 4 execution x
ant -f C:\SCE\PojosPrueba\Pojo_Entregas -Djavac.includes=pojo_entregas/Pojo_Entregas.java -Dnb.internal.action.name=run-single
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
wsimport-init:
wsimport-client-WSEntregas:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Entregas\build\classes
compile-single:
run-single:
Generamos entrega de factura con ID: 1.
ID de Factura: 1
Nombre de Paqueteria: DHL
Tus libros se entregarán en 28 días!
BUILD SUCCESSFUL (total time: 1 second)

```

Caso 2: No se genera una entrega ya que ya existe dicha entrega en la BD.

```

Java DB Database Process x GlassFish Server x Retriever Output x SQL 3 execution x Pojo_Entregas (run-single) x
ant -f C:\SCE\PojosPrueba\Pojo_Entregas -Djavac.includes=pojo_entregas/Pojo_Entregas.java -Dnb.internal.action.name=run-single
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
wsimport-init:
wsimport-client-WSEntregas:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Entregas\build\classes
compile-single:
run-single:
Generamos entrega de factura con ID: 1.
La entrega de esta factura ya existe en la tabla.
BUILD SUCCESSFUL (total time: 1 second)

```

Caso 3: No existe la idFactura dada.

```

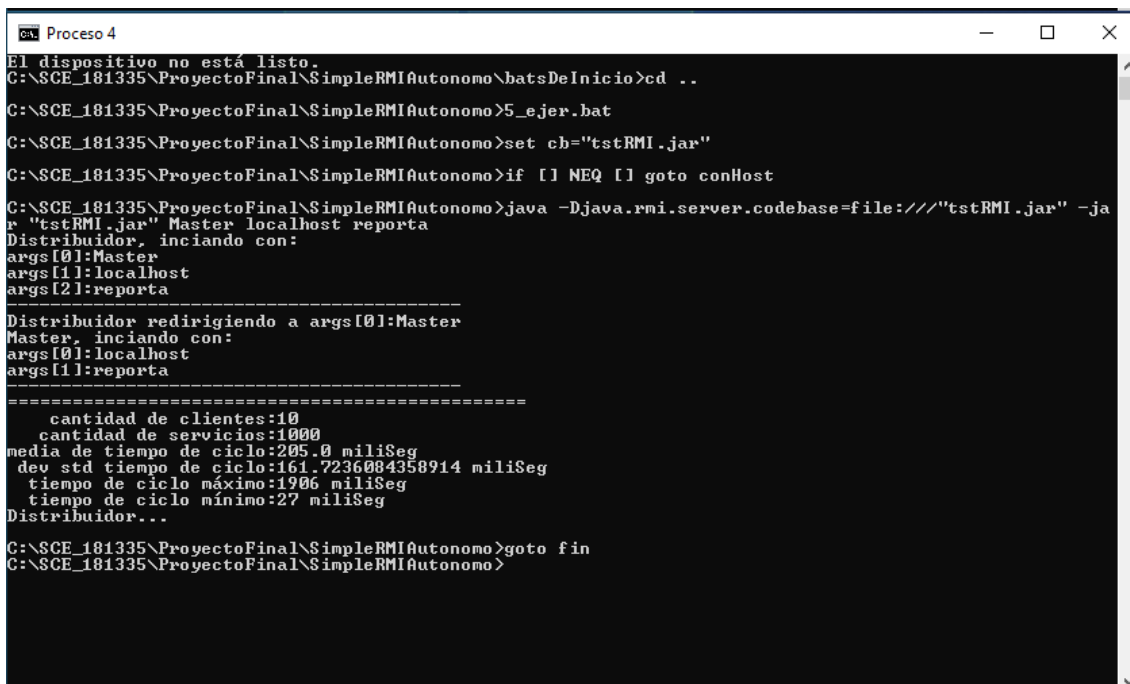
Java DB Database Process x GlassFish Server x Retriever Output x SQL 3 execution x Pojo_Entregas (clean.jar) x Pojo_Entregas (run-single) x
ant -f C:\SCE\PojosPrueba\Pojo_Entregas -Djavac.includes=pojo_entregas/Pojo_Entregas.java -Dnb.internal.action.name=run-single
init:
Deleting: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
deps-jar:
Updating property file: C:\SCE\PojosPrueba\Pojo_Entregas\build\build-jar.properties
wsimport-init:
wsimport-client-WSEntregas:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\SCE\PojosPrueba\Pojo_Entregas\build\classes
compile-single:
run-single:
Generamos entrega de factura con ID: -1.
El id de factura no existe.
BUILD SUCCESSFUL (total time: 1 second)

```

Pruebas No Funcionales:

Transacción	# Clientes	# Transacciones	Tiempo Medio de Ciclo (ms)	Tiempo Ciclo Máximo (ms)	Tiempo Ciclo Mínimo (ms)	Tiempo (s)	Transacc. por Minuto
crearEnvio()	10	100	205.0	1906	27	205	293

Evidencia Pruebas No Funcionales:



13 Estress general de cada servicio y a la aplicación

- Script, Proceso y Pojos de estrés;
- Tabla con resultados de las Pruebas de Estrés para las versiones estables.

14 Anexos:

El proyecto se desarrolló utilizando Git como herramienta de control de versiones. El repositorio de GitHub que contiene todos los elementos del proyecto se encuentra en el siguiente enlace: <https://github.com/DiegoHuesos/E-Commerce-Final-Project>.

El repositorio contiene los siguientes elementos:

- Script de base de datos para pruebas funcionales
- Script de base de datos para pruebas no funcionales
- Web services
- Integración en BPEL
- Pojos de estresamiento