Documentación del proyecto final de la materia

SDI 24810 - Sistemas de Comercio Electrónico

Sistema de Comercio Electrónico para la Venta y Distribución de Libros con Gestión de Inventario y Validación de Pago en Tiempo Real

**MIEMBROS DEL EQUIPO**

200821 Leopoldo Morante Castillo

175588 Victoria Ordorica Pardo

203411 Patricio Pizaña Vela

203500 Abraham Martínez Cerón

201432 Eduardo Velázquez Barragán

195989 Asli Alejandra Sandoval Vega

191280 Mauricio de Ariño Bello

# Descripción del Proyecto

Este proyecto se centra en desarrollar una aplicación de comercio electrónico orientada a la venta de libros en línea, que permita a los usuarios registrados realizar compras seguras de ejemplares disponibles en un almacén. La aplicación gestionará el inventario de libros mediante su código ISBN, realizará las transacciones con medios de pago registrados y garantizará la entrega de los productos adquiridos a través de un servicio de paquetería. Asegurando la eficiencia y seguridad en cada operación, el sistema se optimizará para soportar un volumen de transacciones alto, atendiendo la demanda de los usuarios de forma rápida y confiable. Además, el proyecto realiza pruebas de estrés para asegurar que el sistema mantenga un rendimiento óptimo bajo alta carga de usuarios.

[Descripción del Proyecto 1](#_Toc193794206)

[Control de proyecto 2](#_Toc1782929234)

[Requerimientos de Negocio 3](#_Toc3719713)

[Requerimientos Funcionales y No Funcionales 3](#_Toc70148785)

[Historias de usuario 4](#_Toc263176009)

[Happy Paths 4](#_Toc926366418)

[Happy path de compra 4](#_Toc175466231)

[Happy path de alta de medio de pago 4](#_Toc254584918)

[Happy path de entrega 4](#_Toc588821498)

[Happy path integración de servicios 5](#_Toc433140280)

[Rutas alternas 5](#_Toc174590911)

[Ruta Alterna 1: Falta de Inventario 5](#_Toc444837645)

[Ruta Alterna 2: Cliente sin crédito suficiente 5](#_Toc1291554086)

[Documento de arquitectura 6](#_Toc30396631)

[Descripción de la arquitectura propuesta 6](#_Toc78287813)

[Componentes funcionales principales: 6](#_Toc1745949953)

[Responsabilidades de los componentes de la Arquitectura y relación con las reglas del negocio 8](#_Toc505189023)

[Control de avance y desviaciones 9](#_Toc772231776)

[Documentación de las pruebas funcionales 12](#_Toc1970311064)

[Documentación de pruebas no funcionales 16](#_Toc699177541)

[Prueba WS\_Pedido 16](#_Toc1804323307)

[Prueba WS\_Crédito 18](#_Toc1915893960)

[Prueba Pojo\_Proyecto 18](#_Toc164991920)

[Arquitectura y versiones 20](#_Toc1595411719)

[Web Service de Pedidos 21](#_Toc1716678716)

[Relación con Reglas y Requerimientos de Negocio 21](#_Toc918318737)

[Pruebas Funcionales 22](#_Toc74943552)

[Pruebas No Funcionales 22](#_Toc970649953)

[Web Service de Crédito 23](#_Toc1595245387)

[Relación con Reglas y Requerimientos de Negocio 23](#_Toc1736522105)

[Pruebas Funcionales 23](#_Toc1991741601)

[Pruebas No Funcionales 24](#_Toc1678410475)

[Web Service de Envíos 24](#_Toc1026191447)

[Relación con Reglas y Requerimientos de Negocio 24](#_Toc425808423)

[Pruebas Funcionales 25](#_Toc806141478)

[Pruebas No Funcionales 25](#_Toc671446770)

[Estrés general de la aplicación 26](#_Toc544434592)

[Anexos 26](#_Toc8408182)

[Scripts de inicialización de BD de los servicios básicos 26](#_Toc1014897788)

[Código de web services y acceso a BD 26](#_Toc570720774)

[Base de Datos de Pedidos 26](#_Toc368400979)

[Base de Datos de Crédito 28](#_Toc1152866930)

[Base de Datos de Envíos 29](#_Toc676592695)

[Código de web services y acceso a BD 30](#_Toc1164018985)

[Código de WS\_Pedido 30](#_Toc1575460856)

[Código WS\_Crédito 34](#_Toc1828502302)

[Código WS\_Envios 36](#_Toc992919344)

[Pojos para pruebas funcionales de los tres servicios, el integrado y servicios adicionales. 38](#_Toc1513350113)

[Pojo Pedidos 38](#_Toc2069341618)

[Pojo Crédito 41](#_Toc1556097146)

[Pojo Envíos 42](#_Toc453057772)

[BPEL\_Proyecto 43](#_Toc727845055)

[Pojo\_Proyecto 48](#_Toc168960294)

# Control de proyecto

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Actividad** | **Personas involucradas** |
| 11/29/2024 | Creación de las bases de datos | Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 11/30/2024 | Implementación de los Web Services | Patricio Pizaña Vela, Leopoldo Morante Castillo, Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 12/2/2024 | Creación del BPEL integrador | Patricio Pizaña Vela, Leopoldo Morante Castillo, Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 12/4/2024 | Implementar el BPEL en una CA | Patricio Pizaña Vela, Leopoldo Morante Castillo, Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 11/30/2024 | Crear el POJO del CA | Patricio Pizaña Vela, Leopoldo Morante Castillo, Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 11/30/2024 | Probar el happy path | Patricio Pizaña Vela, Leopoldo Morante Castillo, Asli Alejandra Sandoval Vega y Eduardo Velázquez Barragán |
| 11/30/2024 | Realizar los requerimientos de negocio | Abraham Martínez Cerón, Victoria Ordorica Pardo y Mauricio de Ariño Bello |
| 11/30/2024 | Realizar el documento de arquitectura | Abraham Martínez Cerón, Victoria Ordorica Pardo y Mauricio de Ariño Bello |
| 11/30/2024 | Integrar fotos para el documento de pruebas | Abraham Martínez Cerón, Victoria Ordorica Pardo y Mauricio de Ariño Bello |
| 11/30/2024 | Diseñar los casos de uso | Abraham Martínez Cerón, Victoria Ordorica Pardo y Mauricio de Ariño Bello |
| 12/2/2024 | Probar escenarios alternativos | Abraham Martínez Cerón, Victoria Ordorica Pardo y Mauricio de Ariño Bello |
| 12/6/2024 | Entrega del proyecto | Todos |

# Requerimientos de Negocio

## Requerimientos Funcionales y No Funcionales

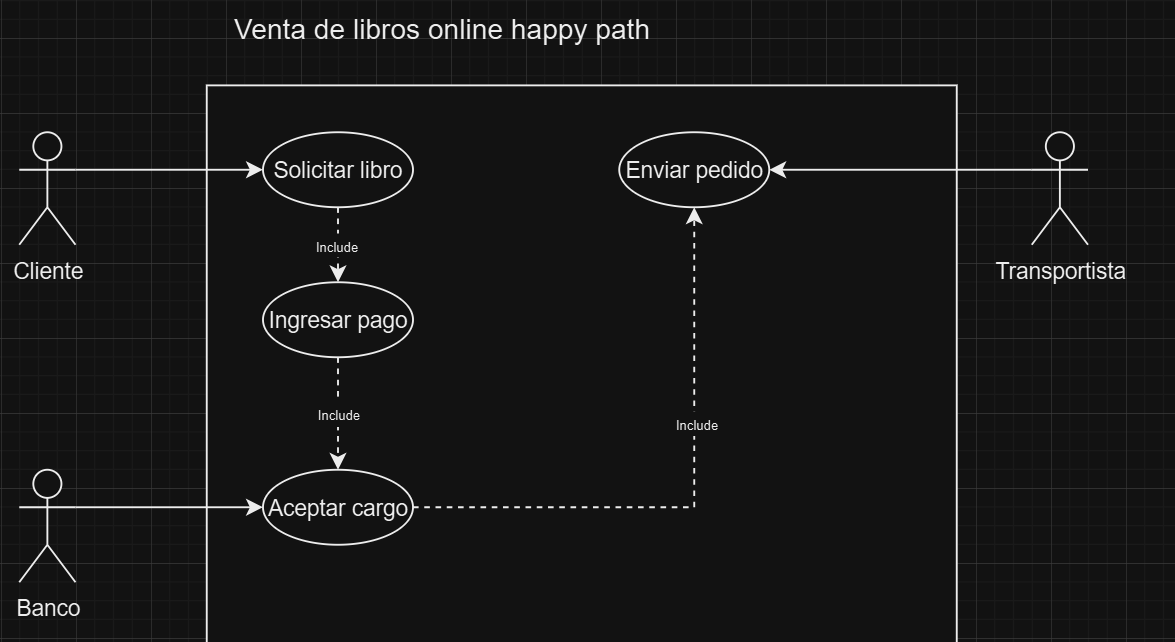
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Descripción del requerimiento | Prioridad | Actor responsable |
| RN1 | Atender solicitudes de compra de usuarios registrados en el medio de pago | Alta | Desarrollador de WS de Pagos |
| RN2 | Aplicación para adquirir artículos registrados en un almacén por medio de su clave ISBN | Alta | 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 | Alta | Desarrollador de WS de Pagos |
| RN4 | No debe permitir la venta de ejemplares no disponibles | Muy alta | Desarrollador de WS de Almacén |
| RN5 | No debe permitir la venta si no se ha registrado un medio de pago o el medio de pago carece de fondos | Muy alta | Desarrollador de WS de Pagos |
| 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 | Media | Desarrollador de WS de Envíos |
| RN7 | Estresamiento de cada uno de los componentes | Media | Estresador |
| RN8 | Estresamiento de los componentes en su conjunto | Muy alta | Estresador |
| RN9 | Interconexión de todos los servicios web | Alta | Integrador |
| RNF1 | Debe soportar 100 transacciones por minuto, incluyendo altas de artículos, modificaciones de stock y modificaciones en el medio de pago | Muy alta | Estresador |
| RNF2 | Simplicidad para el usuario | Media | Desarrolladores |

# Historias de usuario

## Happy Paths

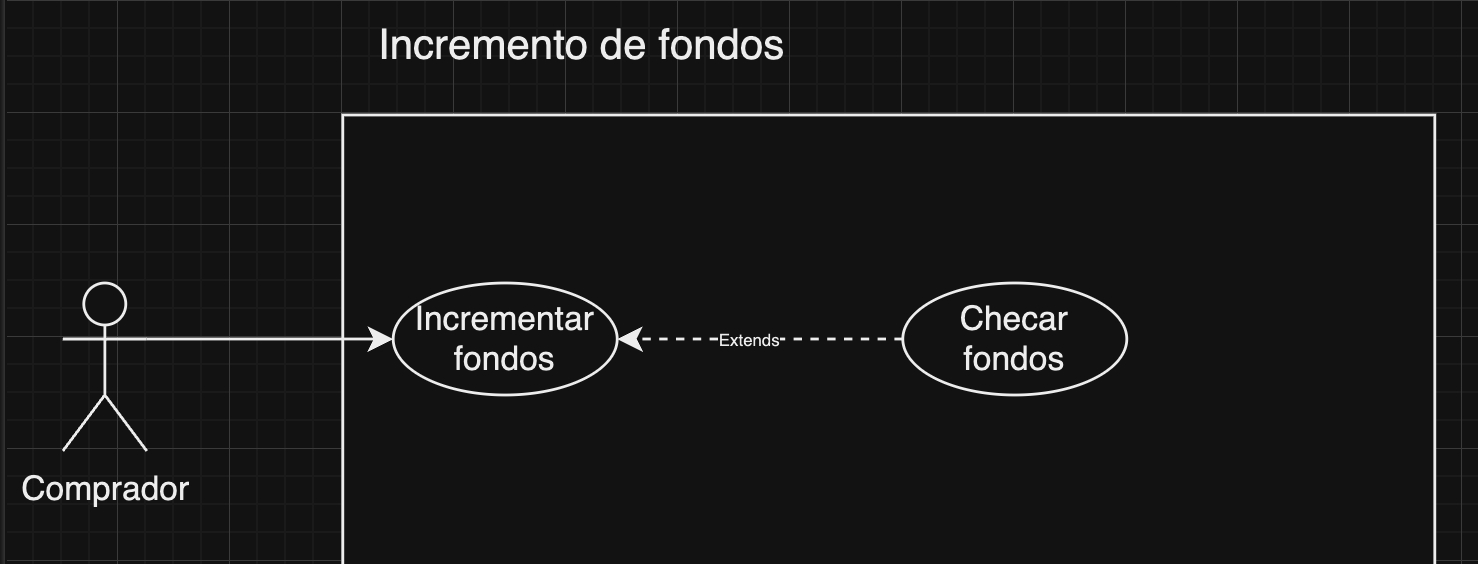
### Happy path integración de servicios

Como usuario, quiero solicitar un libro, dar mi método de pago, el banco me autorice la transacción y tranquilamente llegue a mi casa por medio del transportista



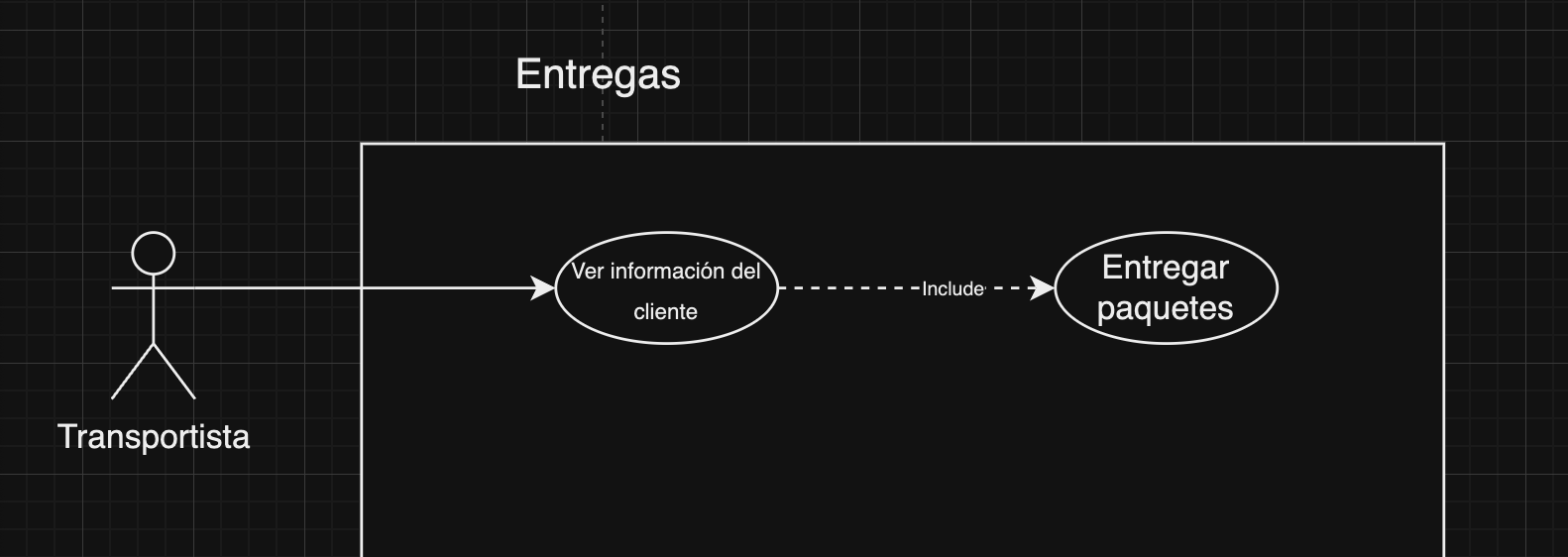
### Happy path de incremento de fondos

Al método autoriza Crédito del WS\_Credito\_libros, darle un monto negativo para aumentar el crédito



### Happy path entrega

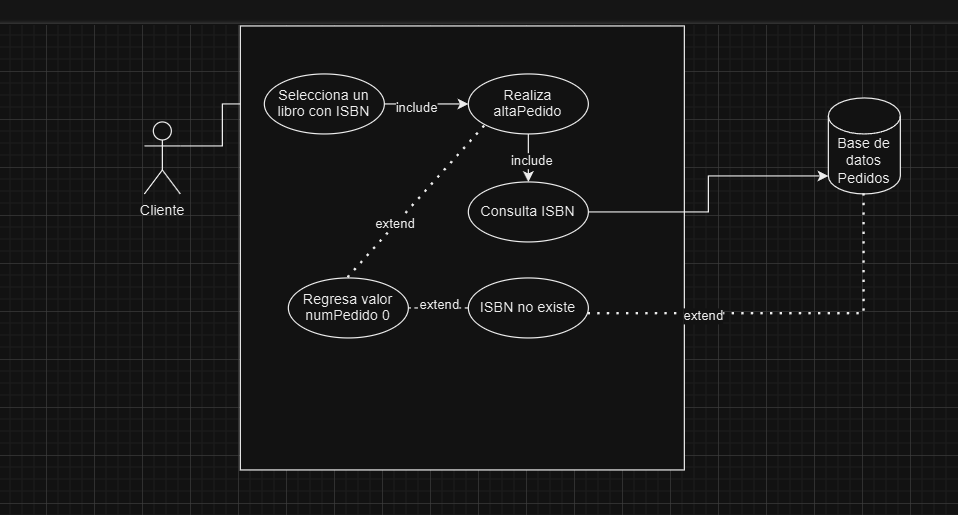
Como usuario, quiero poder comprar mis libros y que lleguen a mi casa sin tener que desplazarme



## Rutas alternas

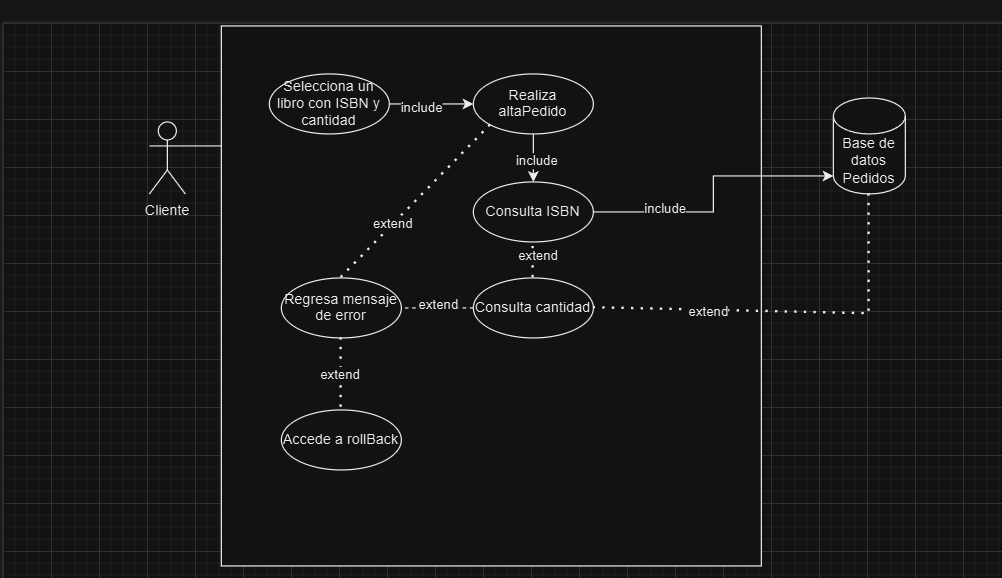
### Ruta Alterna 1: No existe el ISBN

*El usuario solicita un libro con ISBN especifico, por falta de existencia se cancela el pedido.*



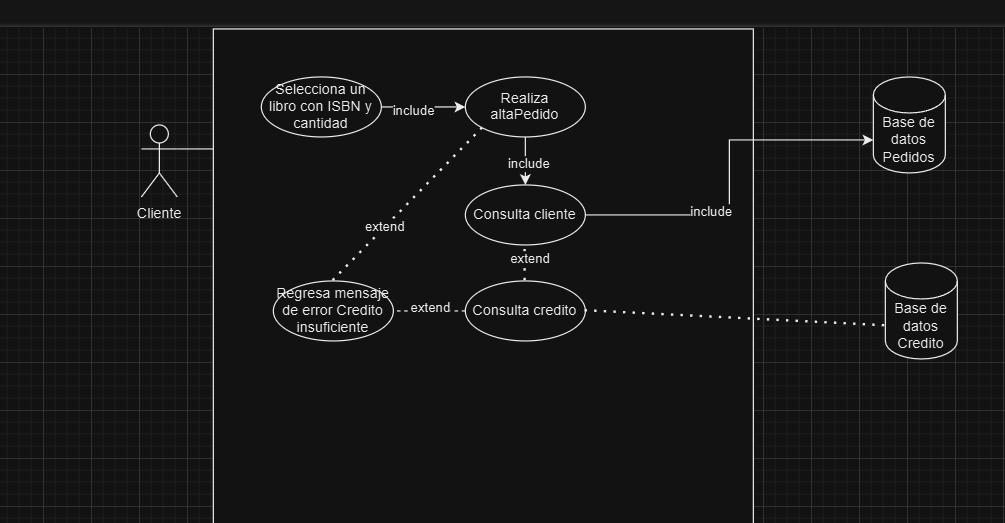
### Ruta Alterna 2: No hay stock

*El usuario solicita un libro, pero por falta de inventario se cancela el pedido.*



### Ruta Alterna 3: Cliente sin crédito suficiente (no hay fondos)

*El usuario solicita un libro, por falta de crédito se cancela el pedido.*



# Documento de arquitectura

## Descripción de la arquitectura propuesta

La arquitectura propuesta se basa en un modelo orientado a servicios (SOA), diseñado para garantizar una separación de responsabilidades y facilitar la escalabilidad y el mantenimiento. Esta arquitectura está dividida en varios componentes funcionales, cada uno de los cuales se encarga de gestionar un aspecto específico del sistema: pedidos, créditos, y envíos.

Adicionalmente, se incorpora una capa de orquestación central para coordinar las interacciones entre los distintos servicios y garantizar el flujo de trabajo global.

#### **Componentes funcionales principales:**

1. **Base de datos independiente por módulo**:

Cada funcionalidad (créditos, envíos, pedidos) cuenta con su propia base de datos independiente, lo que asegura un almacenamiento eficiente y descentralizado de datos. Esto permite gestionar de forma aislada la información de cada dominio, reduciendo la dependencia entre módulos y el riesgo de errores en cascada.

1. **Servicios web (Web Services):**

Cada módulo funcional tiene un servicio web asociado que actúa como intermediario entre las bases de datos y la lógica de negocio. Los servicios son los siguientes:

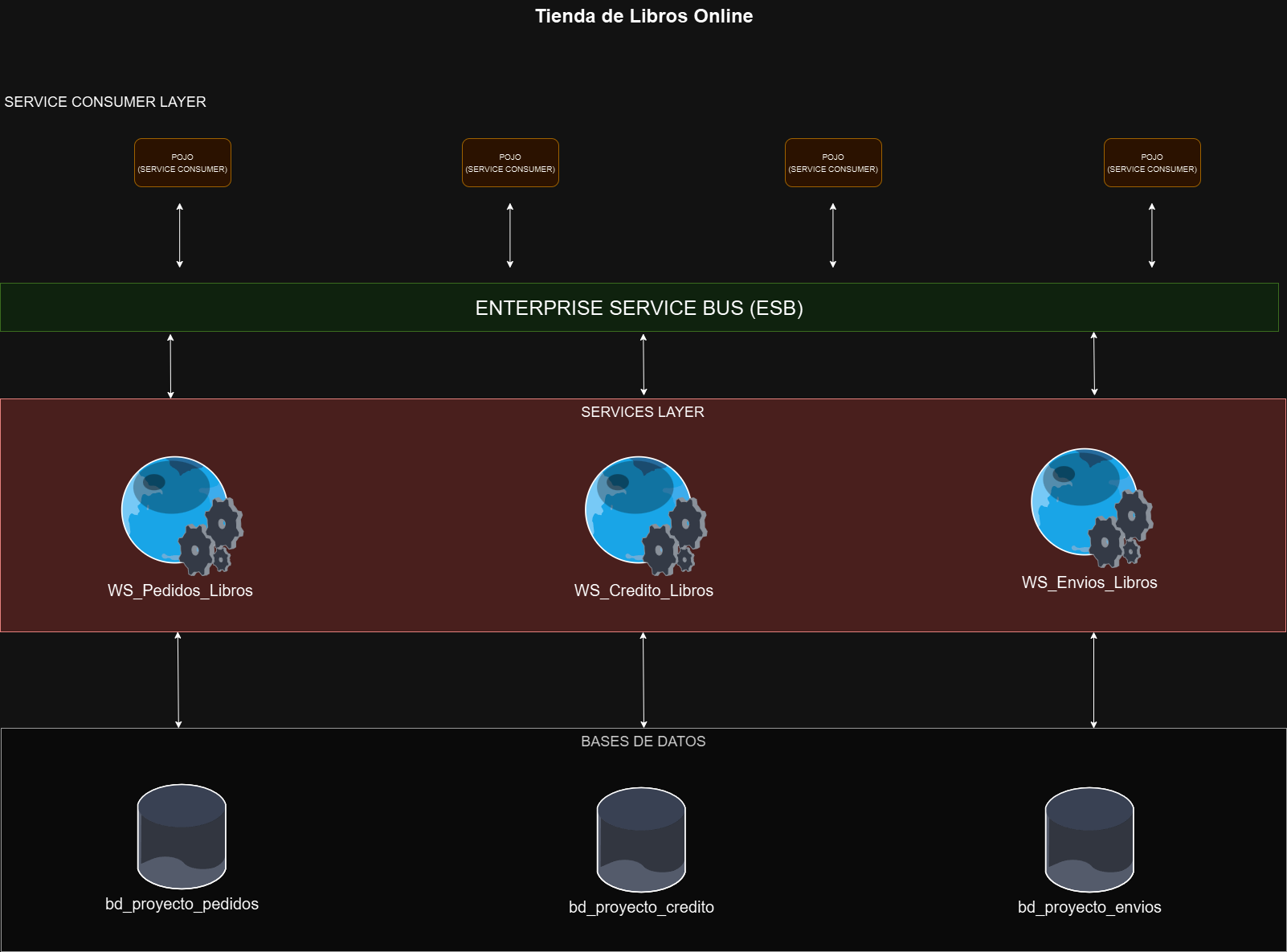
* WS\_Pedidos\_Libros: Procesamiento y seguimiento de pedidos realizados por los usuarios.
* WS\_Credito\_Libros: Gestión de transacciones relacionadas con créditos y fondos de los usuarios.
* WS\_Envios\_Libros: Manejo de la logística de envío, incluyendo información del cliente y las entregas.

1. **Capa de orquestación**:

Un motor de orquestación implementado con BPEL (Business Process Execution Language) coordina la interacción entre los servicios web. Este motor se encarga de integrar las funcionalidades de cada módulo en flujos de trabajo coherentes, asegurando que todas las operaciones (como la compra de un libro) se realicen de forma ordenada.

1. **Clientes de pruebas (POJO)**:

Para evaluar el rendimiento y la estabilidad del sistema, se implementan clientes de tipo POJO que interactúan con el motor de orquestación y los servicios web, simulando distintos escenarios de uso. Estos clientes también realizan pruebas de estrés para identificar posibles cuellos de botella.



*Bosquejo de la Arquitectura SOA de la Tienda de Libros en Línea*

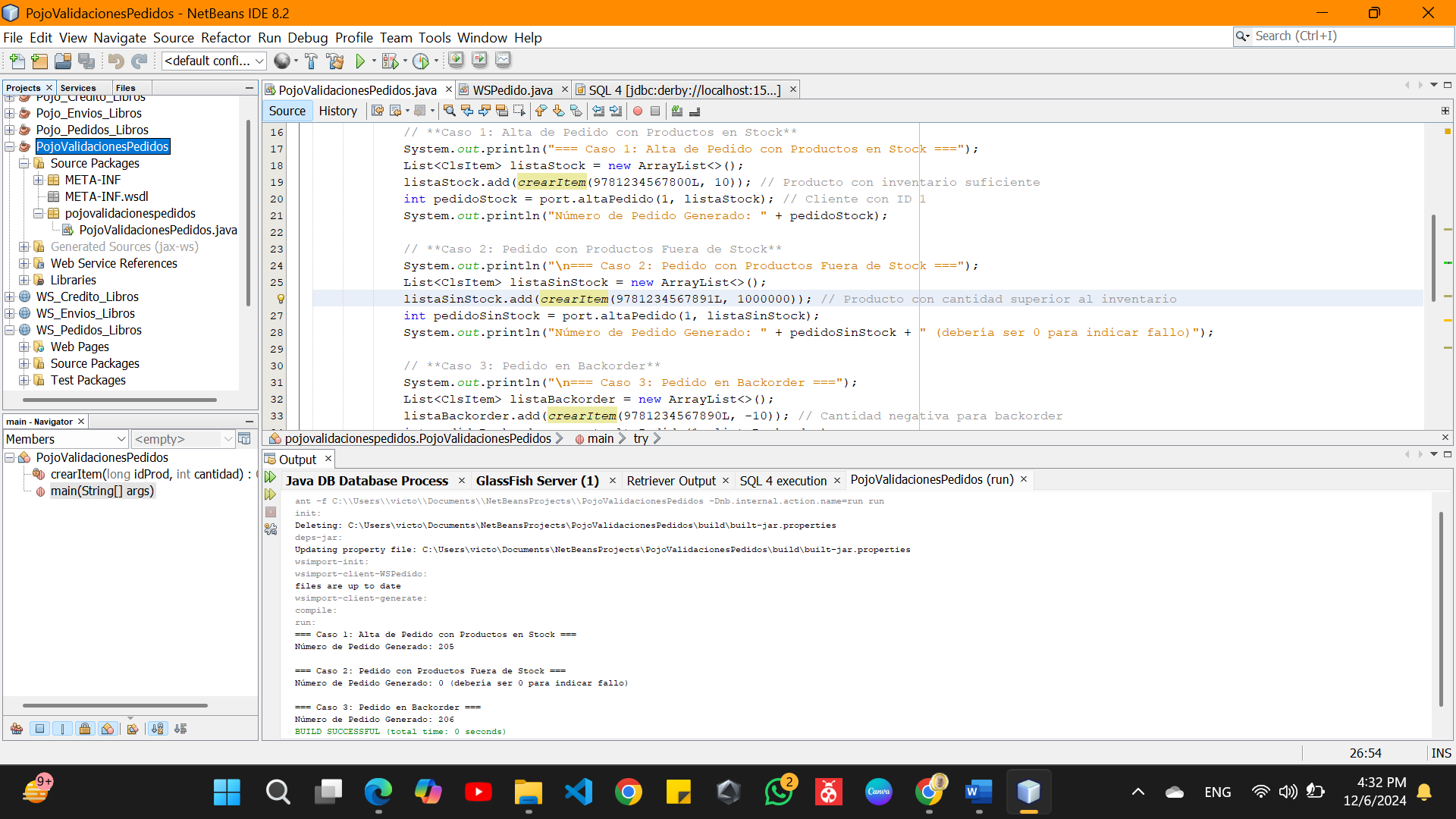
# Responsabilidades de los componentes de la Arquitectura y relación con las reglas del negocio

|  |  |  |
| --- | --- | --- |
| **ID del Componente** | **Responsabilidades Funcionales** | **Responsabilidades No Funcionales** |
| Base de Datos bd\_proyecto\_pedidos | Gestionar y almacenar información sobre los pedidos realizados por los clientes, incluyendo cantidades y estados. | - Escalabilidad para soportar altas transacciones.  - Seguridad para proteger los datos de pedidos. |
| Base de Datos - bd\_proyecto\_credito | Gestionar y almacenar información sobre los créditos asignados a los clientes para la compra de libros. | - Alta disponibilidad para consultas rápidas.  - Seguridad en el almacenamiento de datos sensibles. |
| Base de Datos - bd\_proyecto\_credito | Gestionar y almacenar información sobre envíos, incluyendo estados, direcciones y tiempos de entrega. | - Baja latencia para respuestas en tiempo real.  - Resiliencia ante fallos del sistema. |
| Web Service - WS\_Pedidos\_Libros | Proveer servicios para la gestión de pedidos, incluyendo registro, consulta y modificación de pedidos. | - Desempeño eficiente para soportar múltiples peticiones simultáneas.  - Seguridad en la transmisión de datos. |
| Web Service - WS\_Credito\_Libros | Proveer servicios para consultar y actualizar información relacionada con los créditos de los clientes. | - Mantener la integridad en las transacciones entre la base de datos y el cliente.  - Tiempo de respuesta óptimo. |
| Web Service - WS\_Envios\_Libros | Proveer servicios para la gestión de envíos, como consulta de estado y actualización de datos en tiempo real. | - Tolerancia a fallos para garantizar la operación continua.  - Alta concurrencia para múltiples solicitudes. |
| BPEL (Proceso de Orquestación) | Coordinar las interacciones entre los tres Web Services y asegurar el flujo correcto de las operaciones del negocio. | - Garantizar la consistencia del flujo de datos entre los servicios.  - Modularidad para facilitar la escalabilidad. |
| Clientes POJO (Stress Test) | Ejecutar pruebas de estrés para validar la capacidad y robustez del sistema ante altas demandas de usuarios | - Medición del desempeño en situaciones extremas.  - Asegurar la capacidad de recuperación ante picos de carga. |

# Control de avance y desviaciones

**WS\_Pedidos**

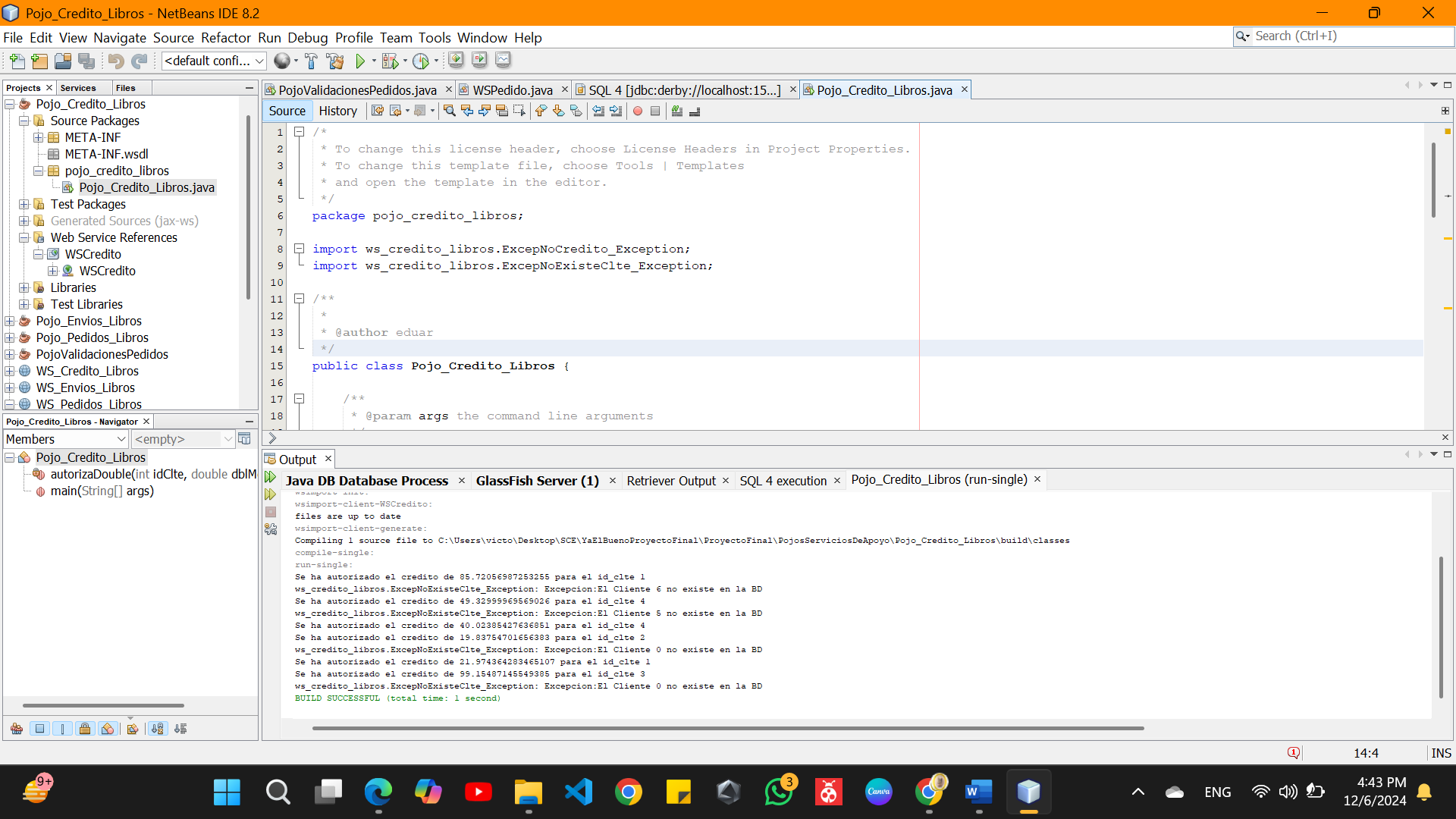
|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Prueba** | **Descripción** | **Resultado Esperado** |
| **WS\_Pedidos** | Validación de alta de pedido | Crear un pedido con productos en stock. | Pedido registrado con ID válido en la base de datos. |
|  | Validación de stock insuficiente | Crear un pedido con productos fuera de stock. | Pedido rechazado con mensaje de error. |
|  | Prueba de backorder | Crear un pedido con productos que no tienen inventario, pero solicitar un backorder. | Pedido registrado con cantidad negativa en inventario. |



*Captura de las pruebas de WS\_Pedidos*

**WS\_Credito**

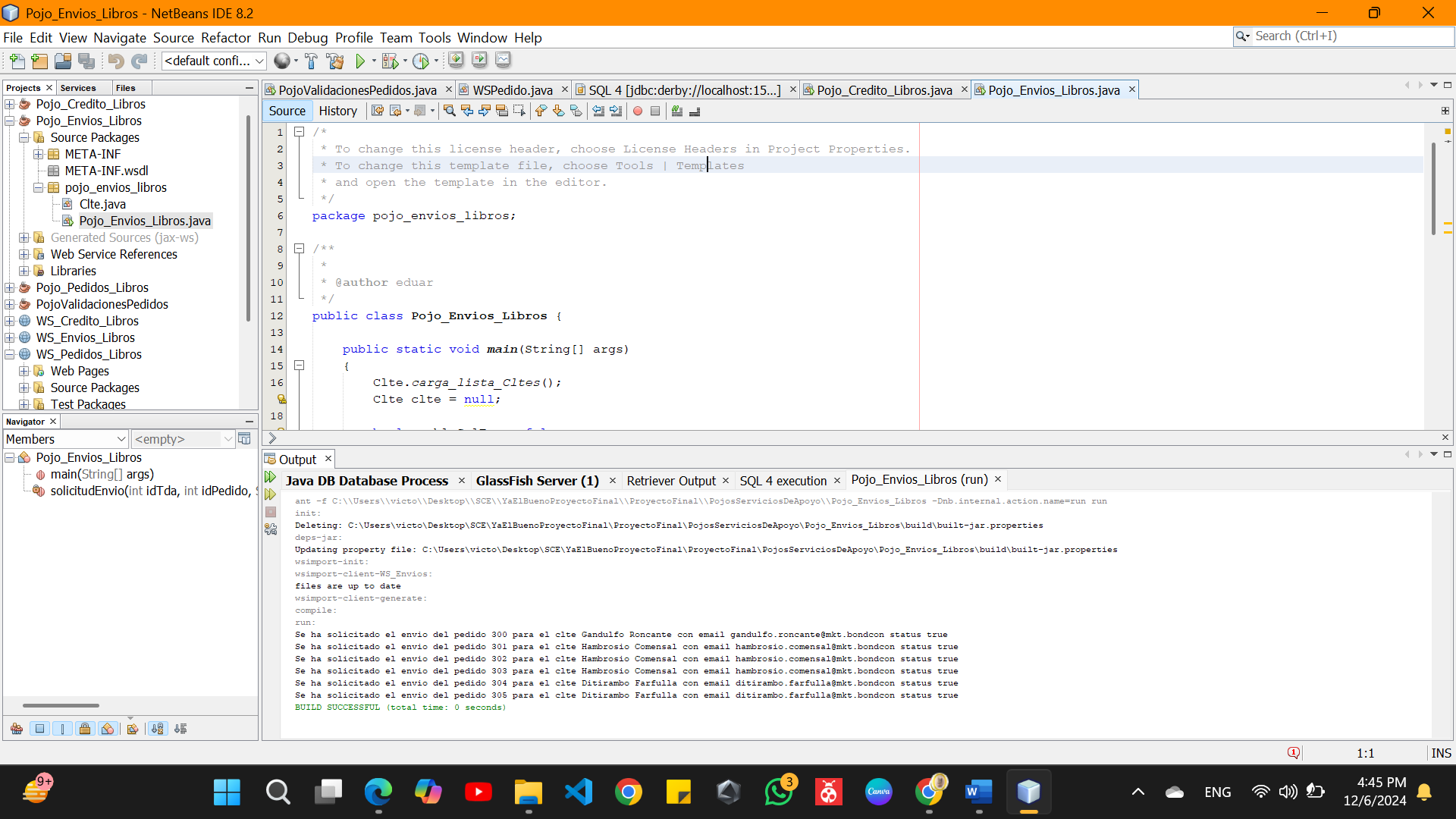
|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Prueba** | **Descripción** | **Resultado Esperado** |
| **WS\_Credito** | Autorización de crédito suficiente | Cliente solicita crédito menor a su límite disponible. | Crédito autorizado; límite actualizado en la base de datos. |
|  | Rechazo por crédito insuficiente | Cliente solicita crédito mayor a su límite disponible. | Crédito rechazado con mensaje de error. |
|  | Cliente no existe en el sistema | Cliente solicita un pedido, pero no se tiene sus datos de registro | Pedido rechazado con mensaje de error. |



*Captura de las pruebas de WS\_Crédito*

**WS\_Envios**

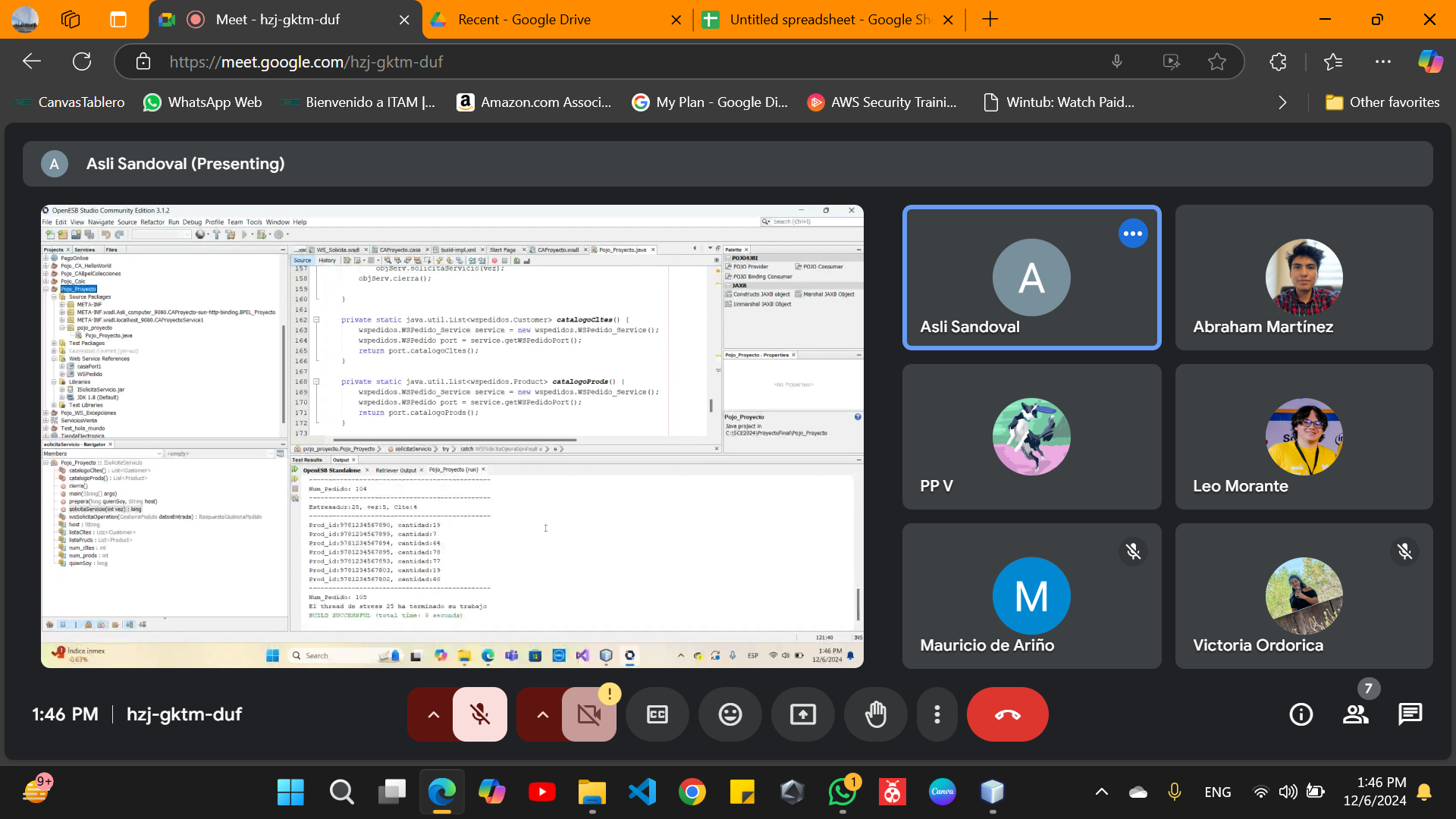
|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Prueba** | **Descripción** | **Resultado Esperado** |
| **WS\_Envios** | Registro de envío | Registrar un envío para un pedido válido. | Envío registrado correctamente en la cola JMS. |
|  | Envío de pedido inexistente | Intentar registrar un envío para un pedido que no existe. | Error de pedido no encontrado. |



*Captura de las pruebas de WS\_Envios*

**BPEL\_Proyecto**

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Prueba** | **Descripción** | **Resultado Esperado** |
| **BPEL\_Proyecto** | Flujo completo (pedido, crédito, envío) | Crear un pedido, validar crédito y registrar el envío como flujo integrado. | Pedido registrado, crédito descontado, envío registrado. |
|  | Validación de fallas combinadas | Flujo completo con fallos en crédito y/o inventario. | Flujo abortado con mensajes de error específicos. |



*Captura de las pruebas de Pojo\_Proyecto*

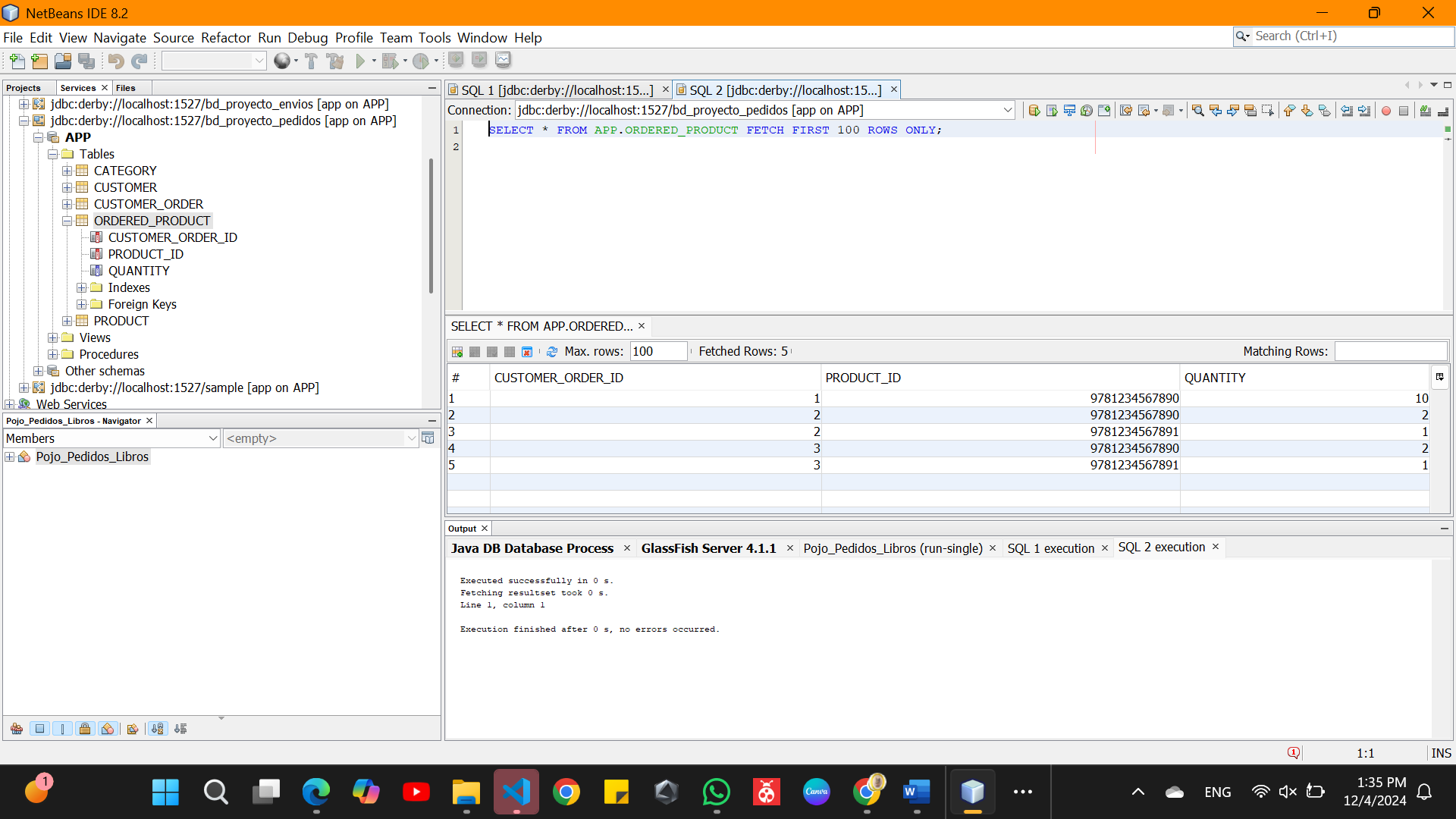
# Documentación de las pruebas funcionales

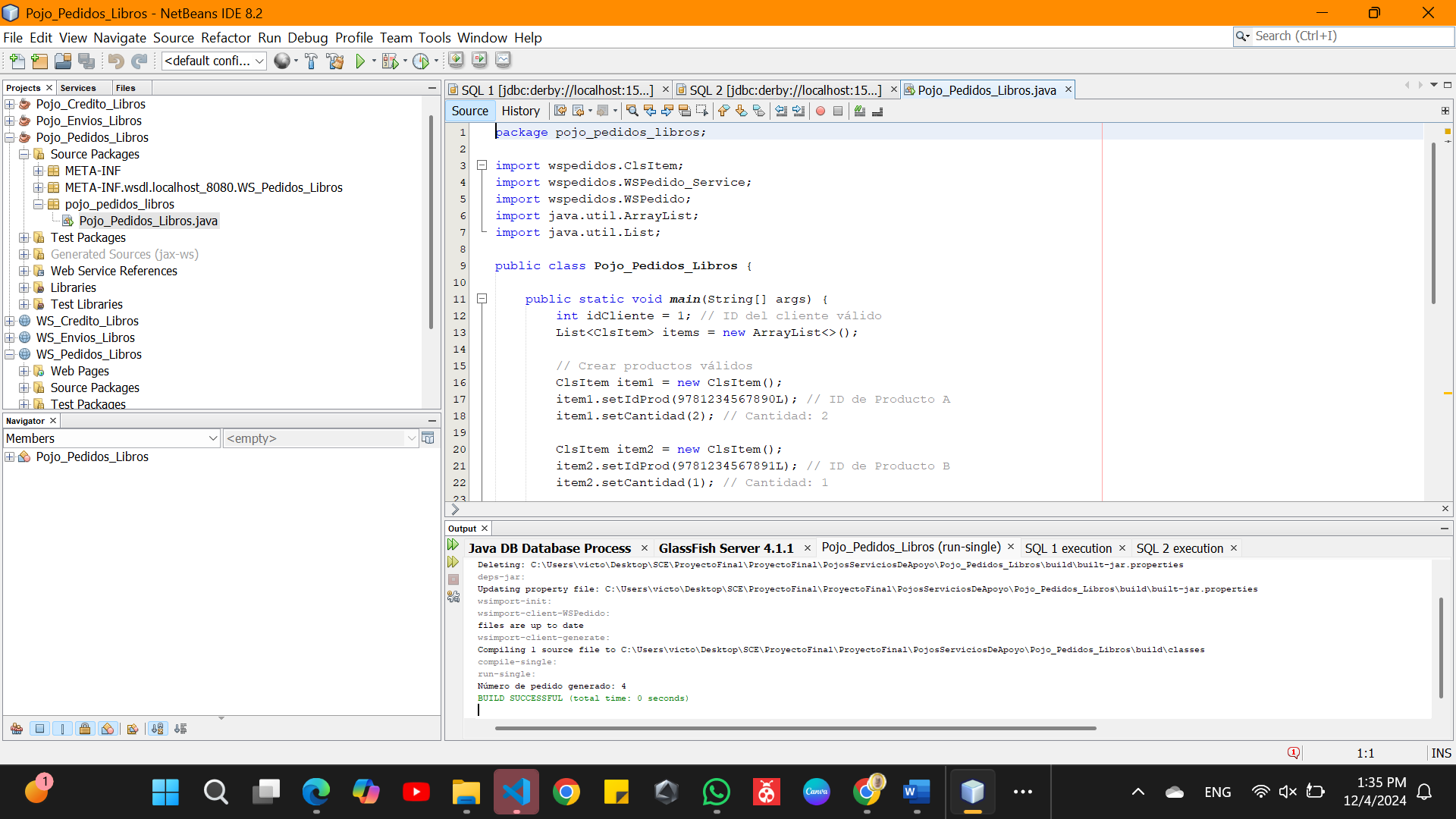
Pruebas en WS\_Pedidos

1. **Prueba: Crear un Pedido Válido**

*El cliente realiza un pedido con productos disponibles en inventario y el pedido se crea correctamente.*

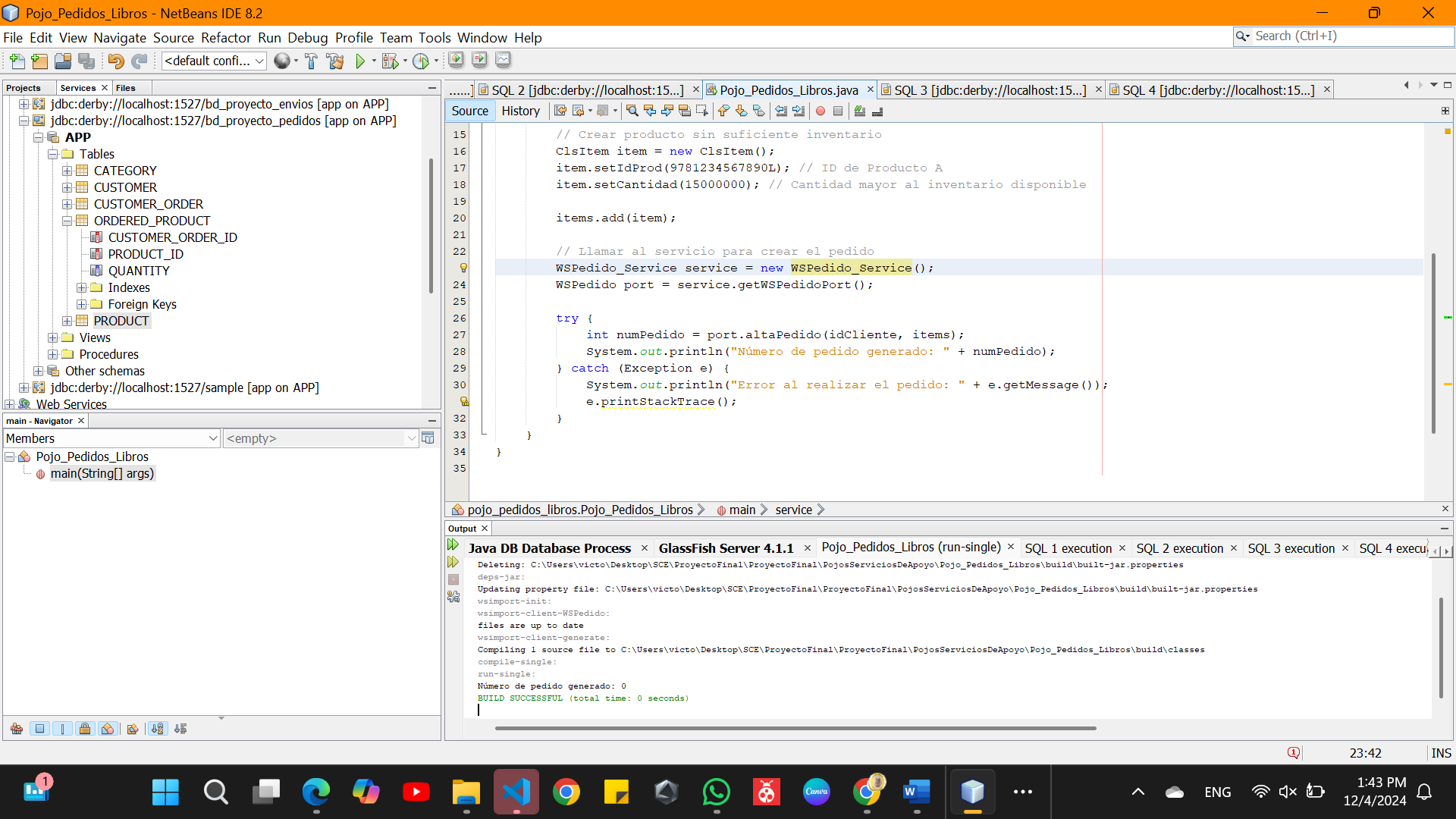
Pruebas:





1. Pedido con Producto Sin Inventario

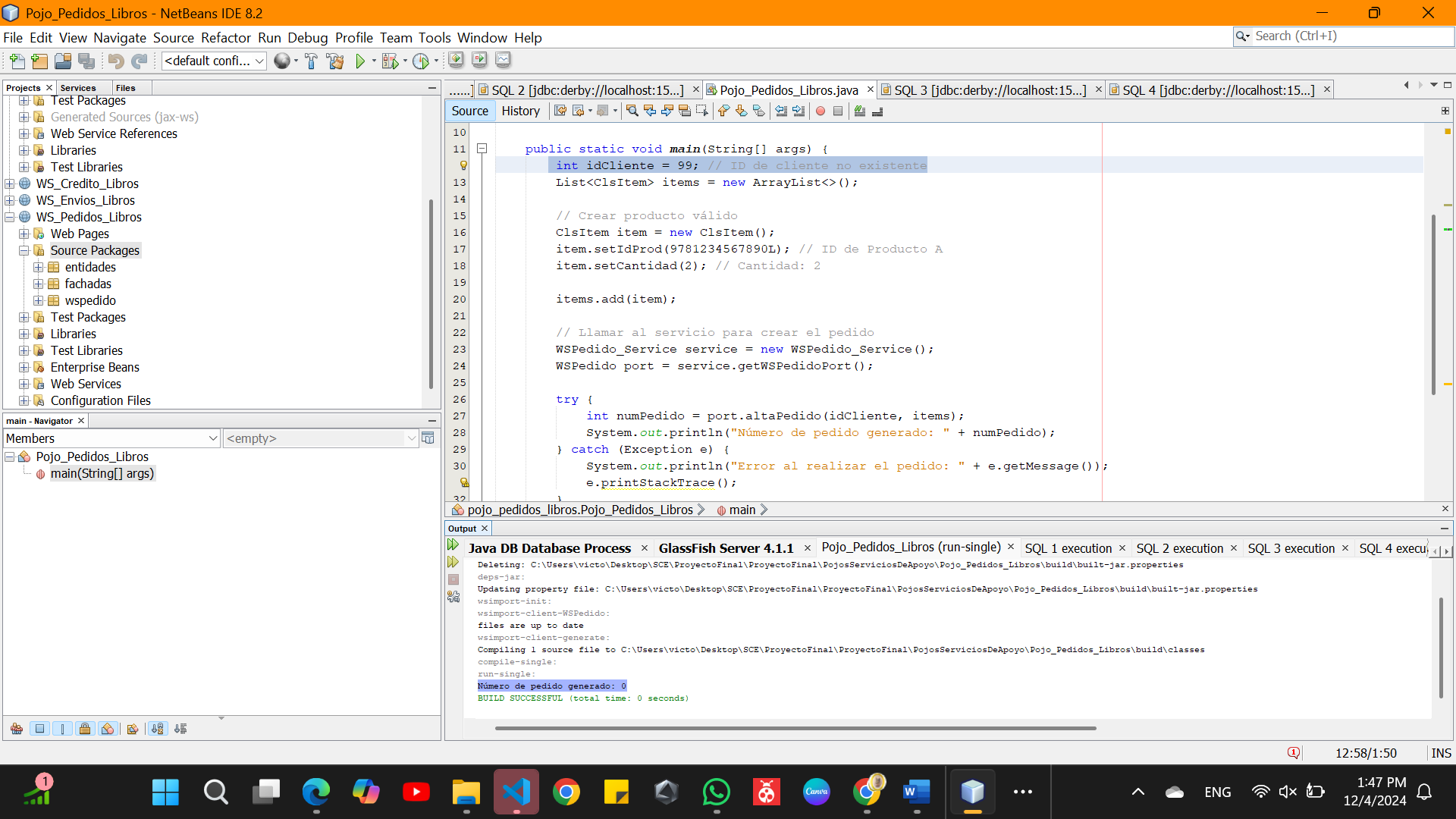
*El cliente realiza un pedido que incluye productos sin inventario disponible.*



Regresa 0 porque no hay suficiente inventario

1. Prueba: Pedido con Cliente No Existente

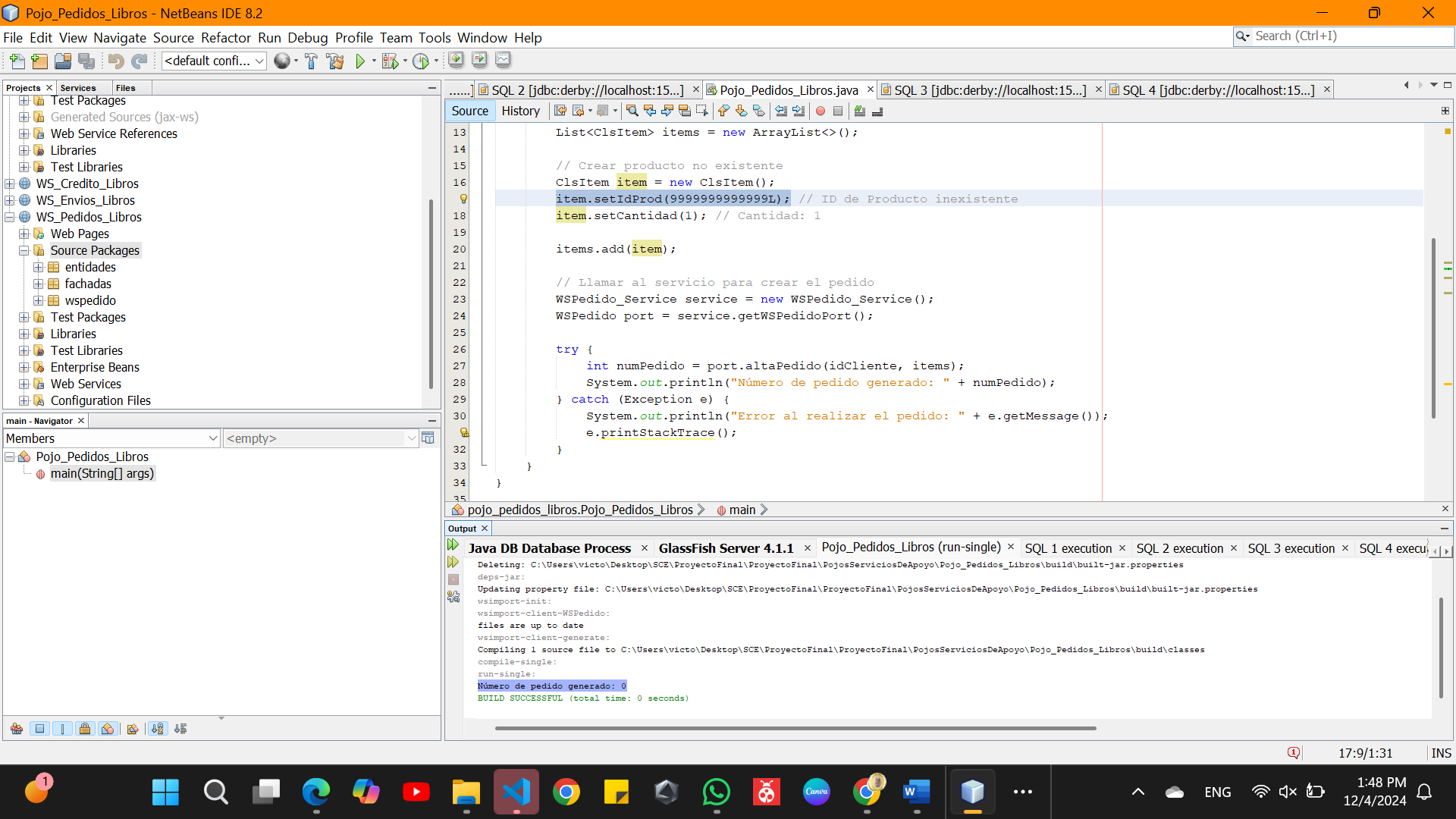
*Un cliente que no está registrado intenta realizar un pedido.*



Regresa 0 porque no existe el usuario

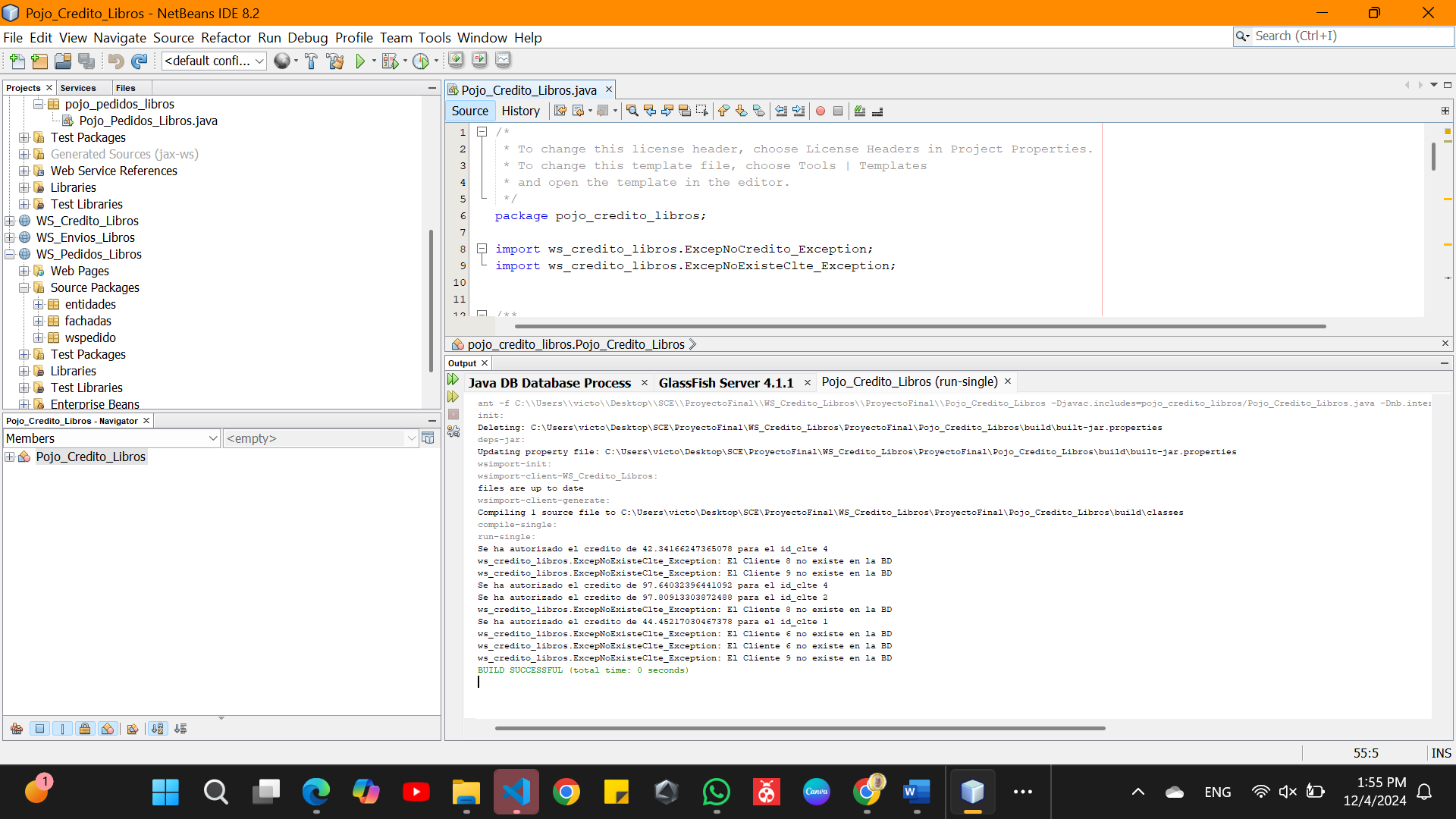
1. Prueba: Pedido con Producto No Existente

*El cliente intenta realizar un pedido con un producto que no existe en la base de datos.*

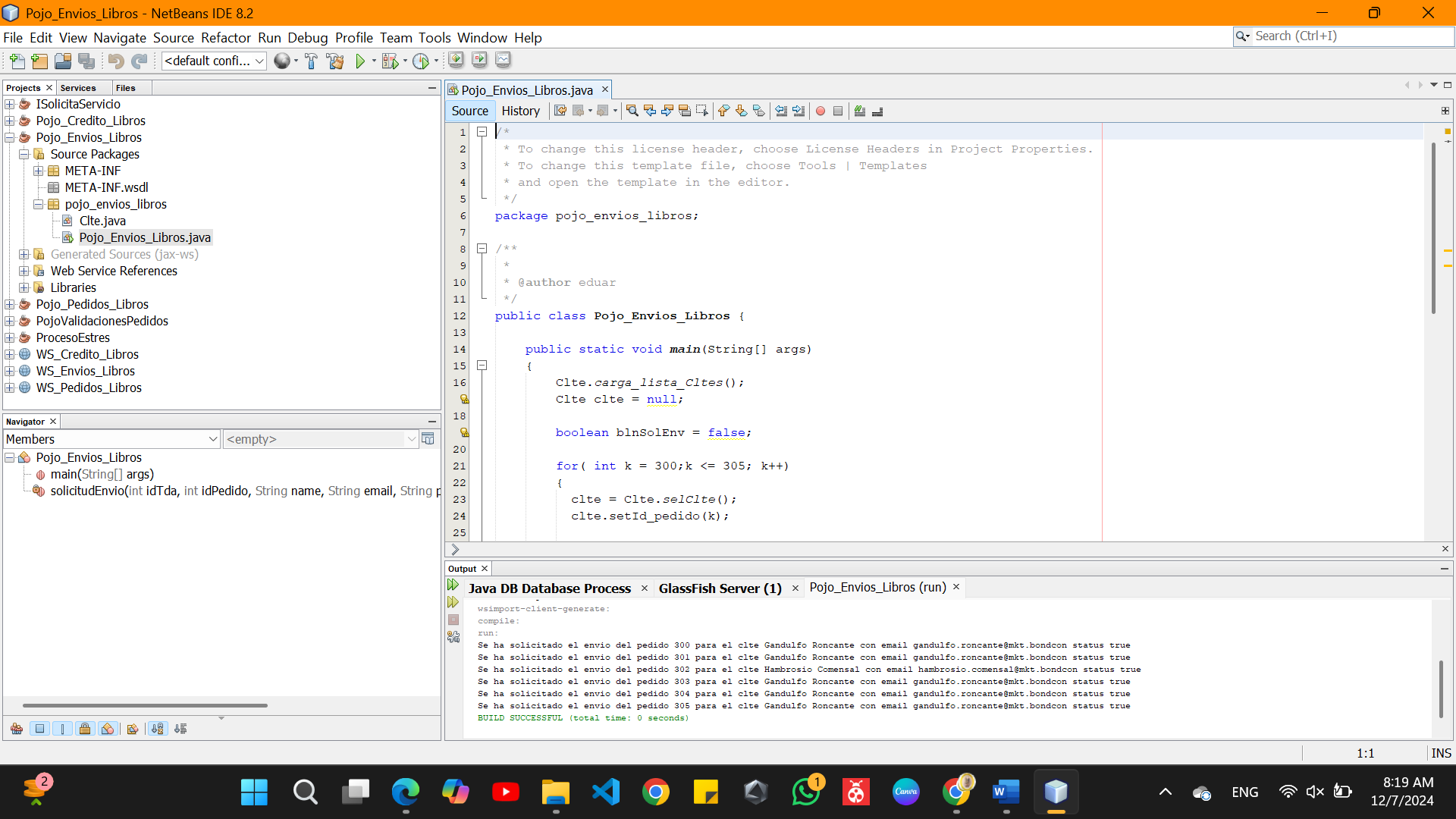


Pruebas en WS\_Credito

Autorización de Crédito Exitosa (Cliente con Crédito Disponible), Cliente con Crédito Insuficiente y Cliente No Existente



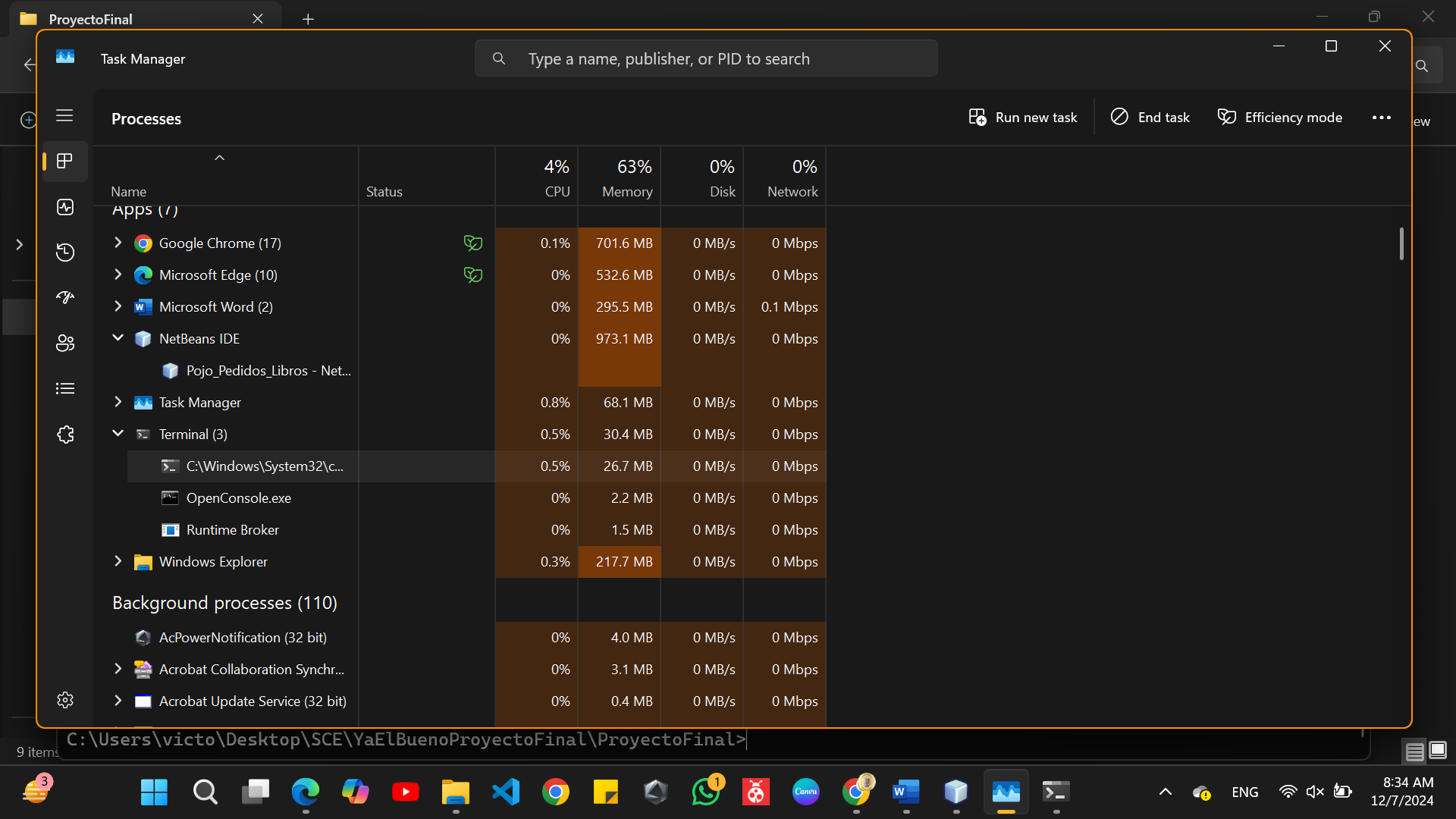
Pruebas en WS\_Envios:



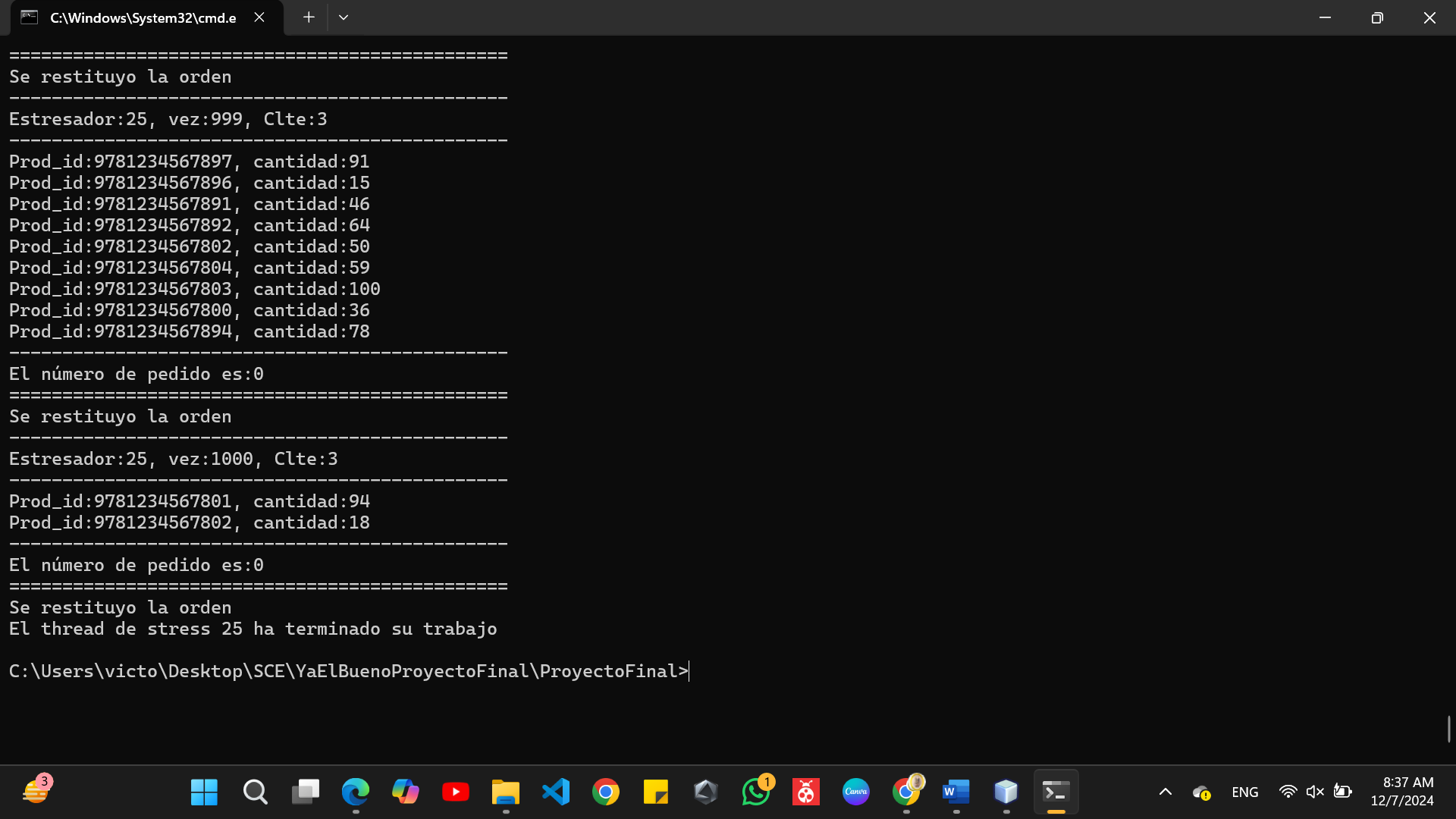
# Documentación de pruebas no funcionales

### Prueba WS\_Pedido

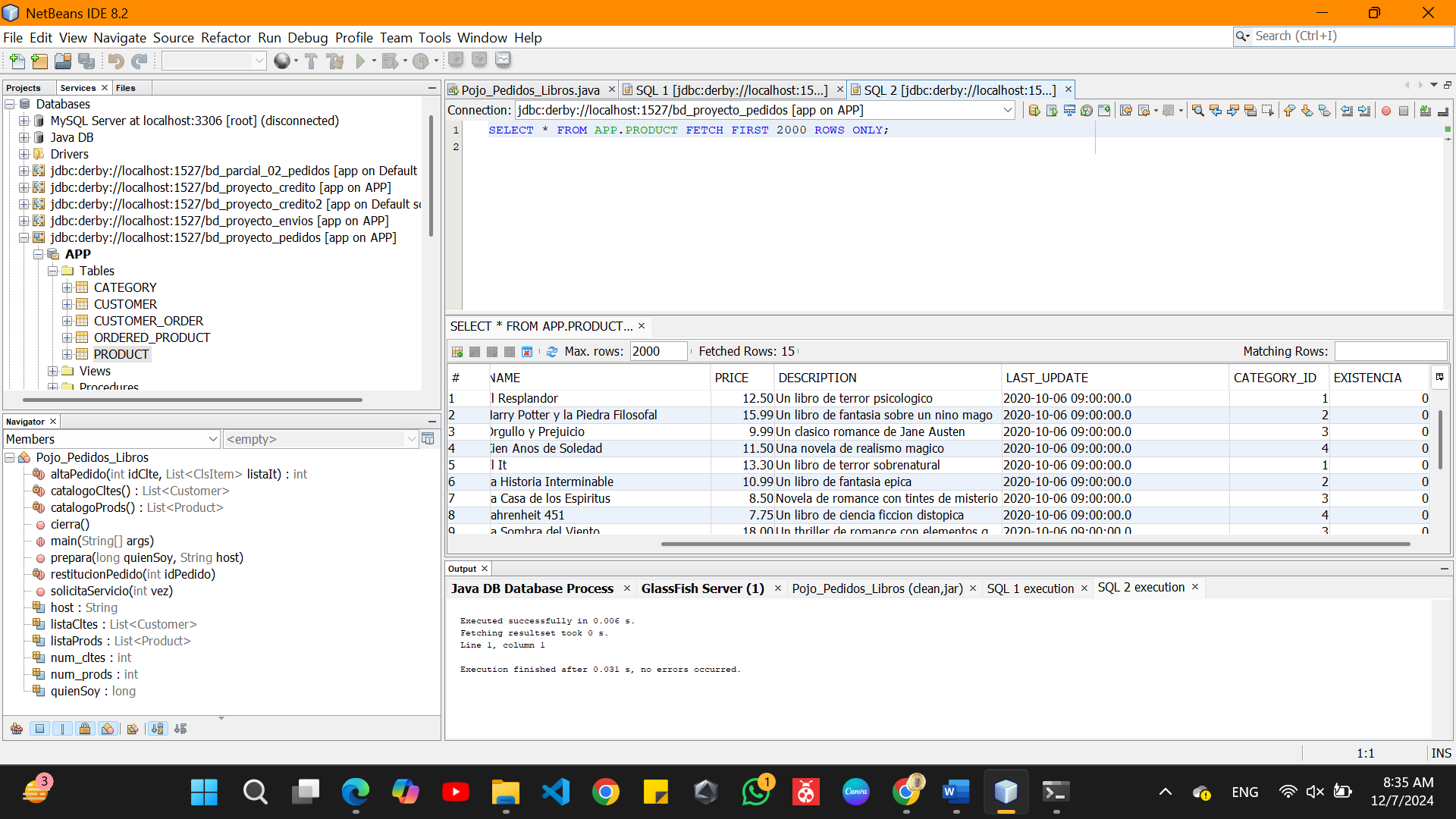
Se realizaron 1000 transacciones desde la terminal con el uso del Pojo de prueba.



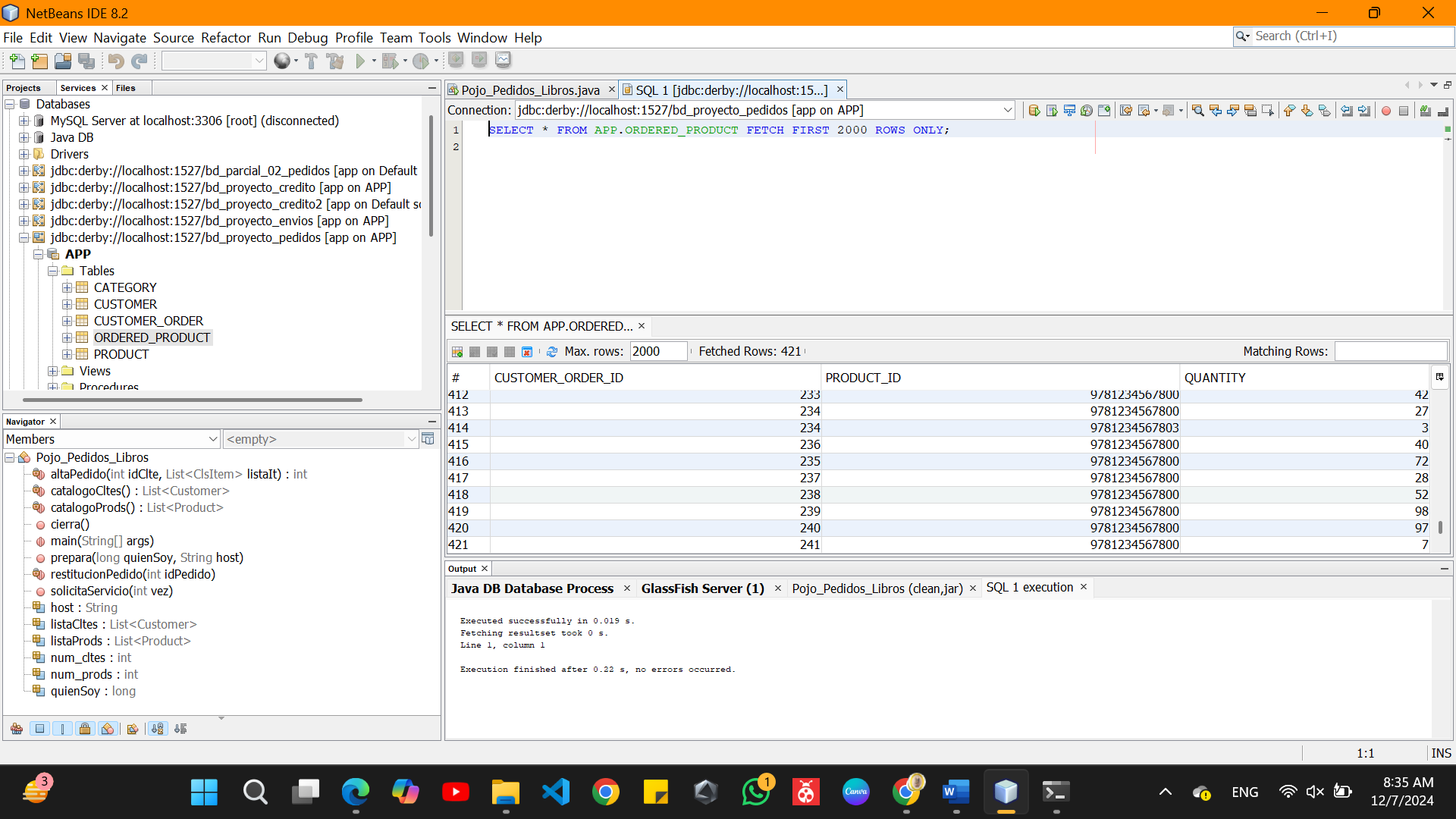
*Uso de CPU para las 1000 transacciones que se realizaron*



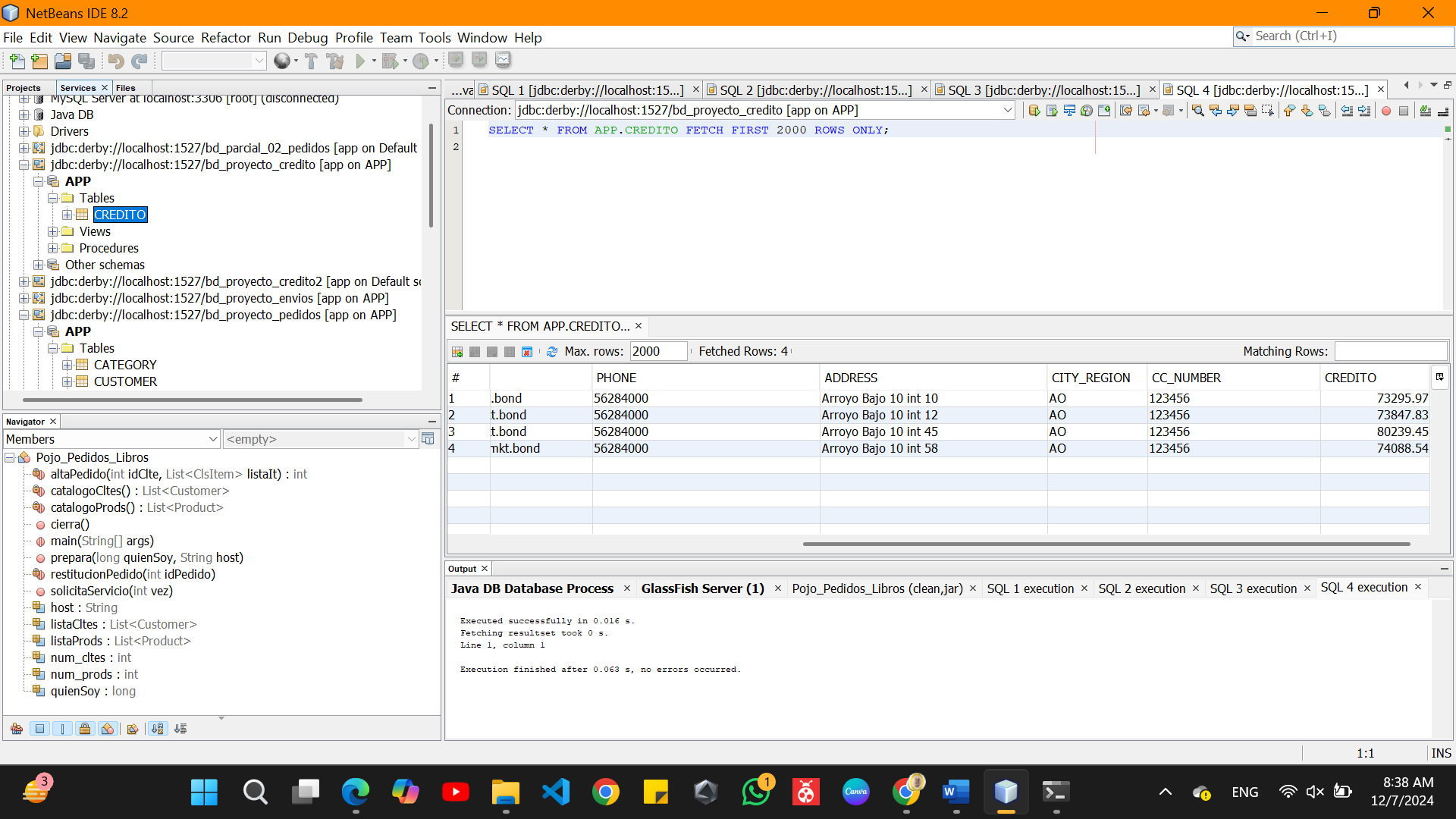
*Prueba de la terminal con 1000 transacciones*



*Modificación de Existencia en Tabla Productos*

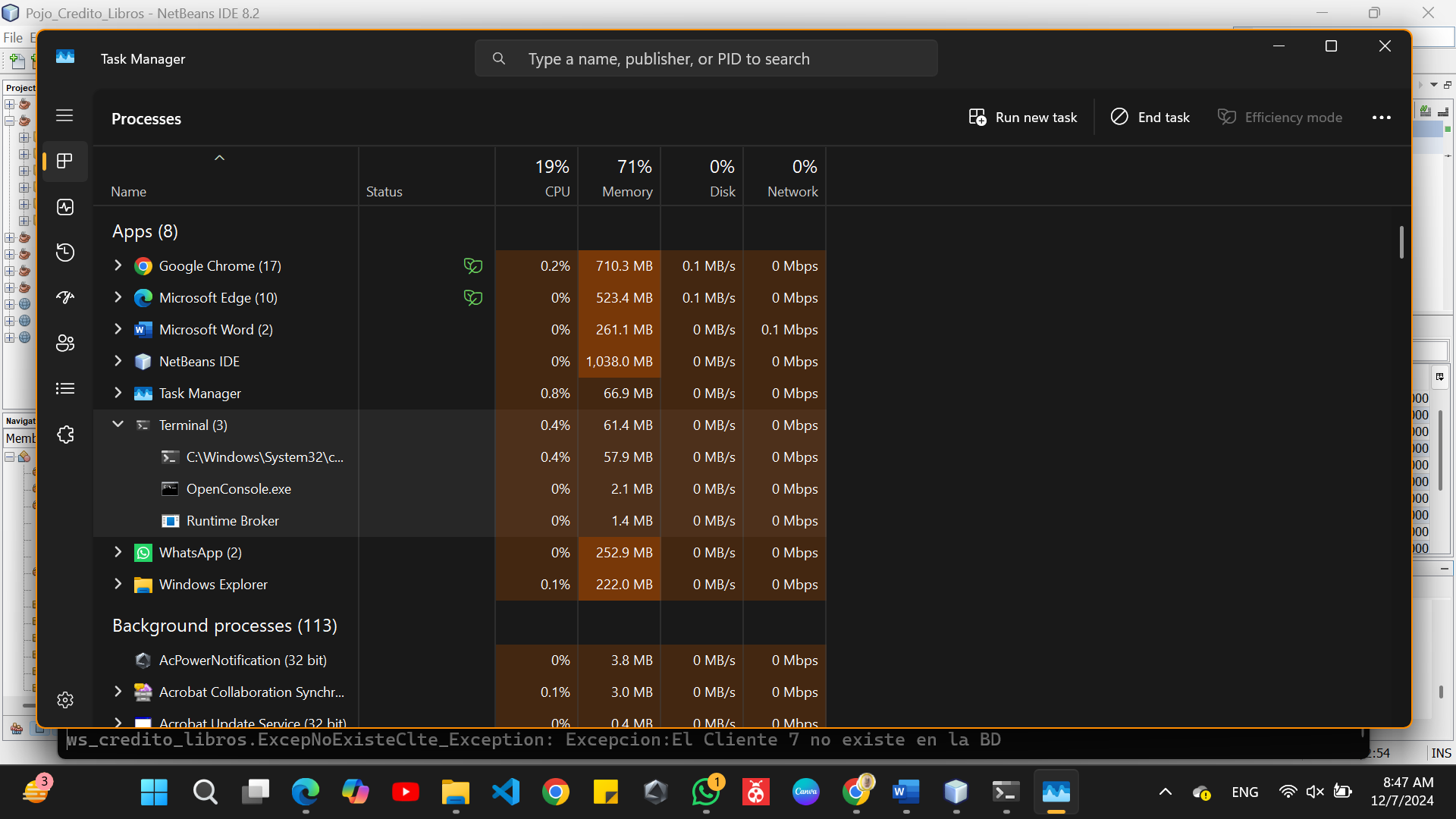
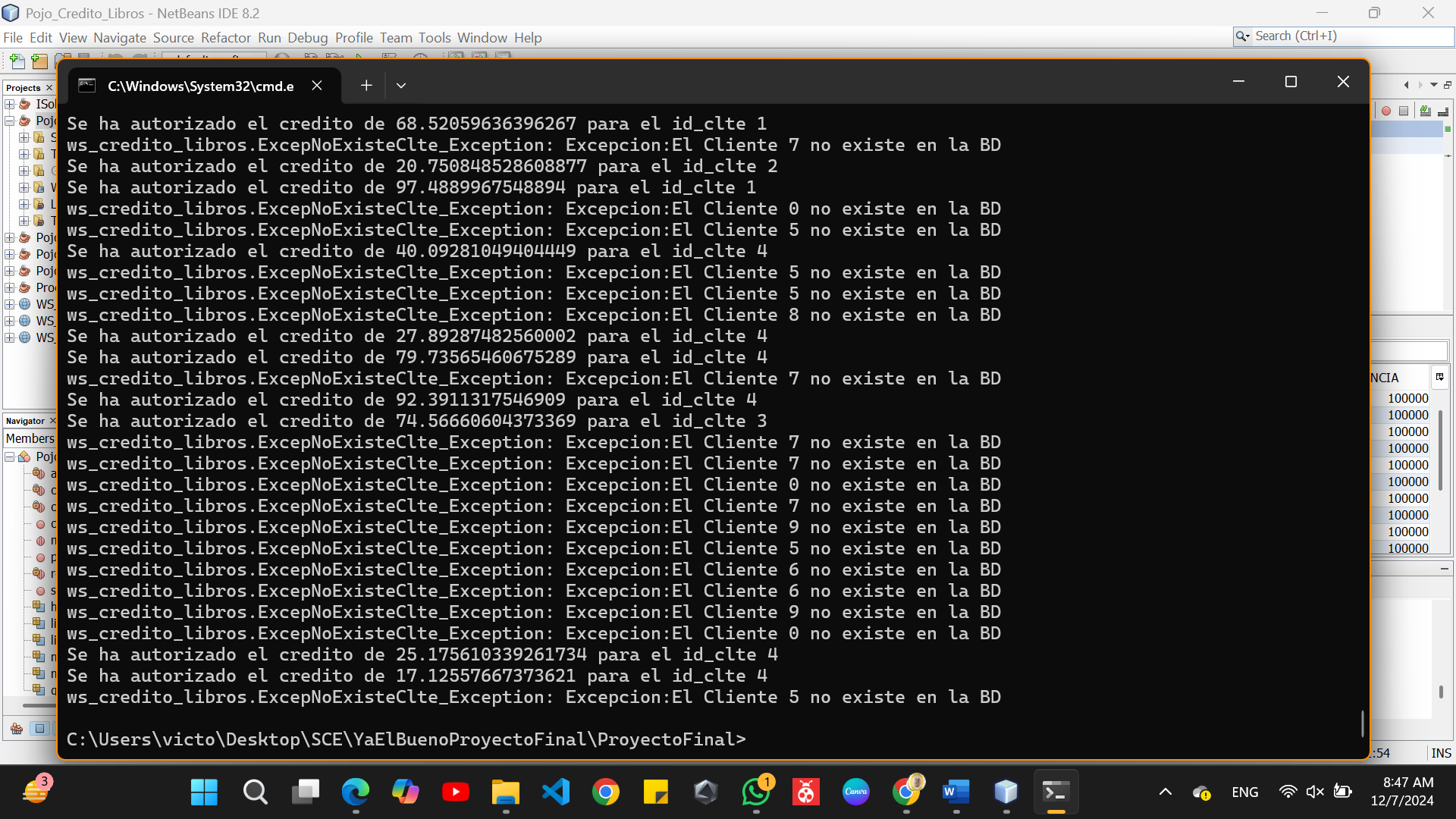


*Modificación de la Tabla Ordered\_Product*

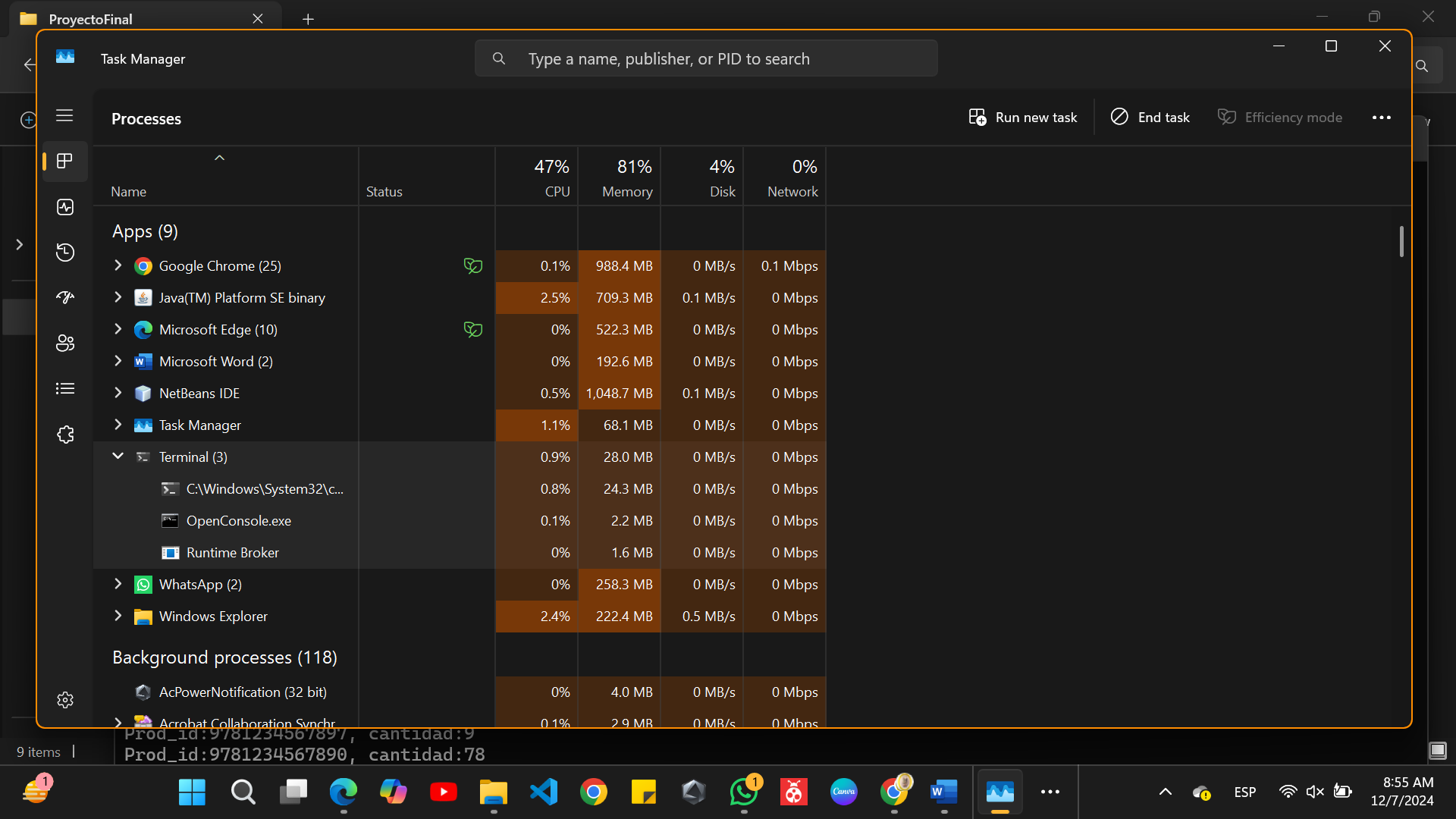


*Modificación de la Tabla de Crédito*

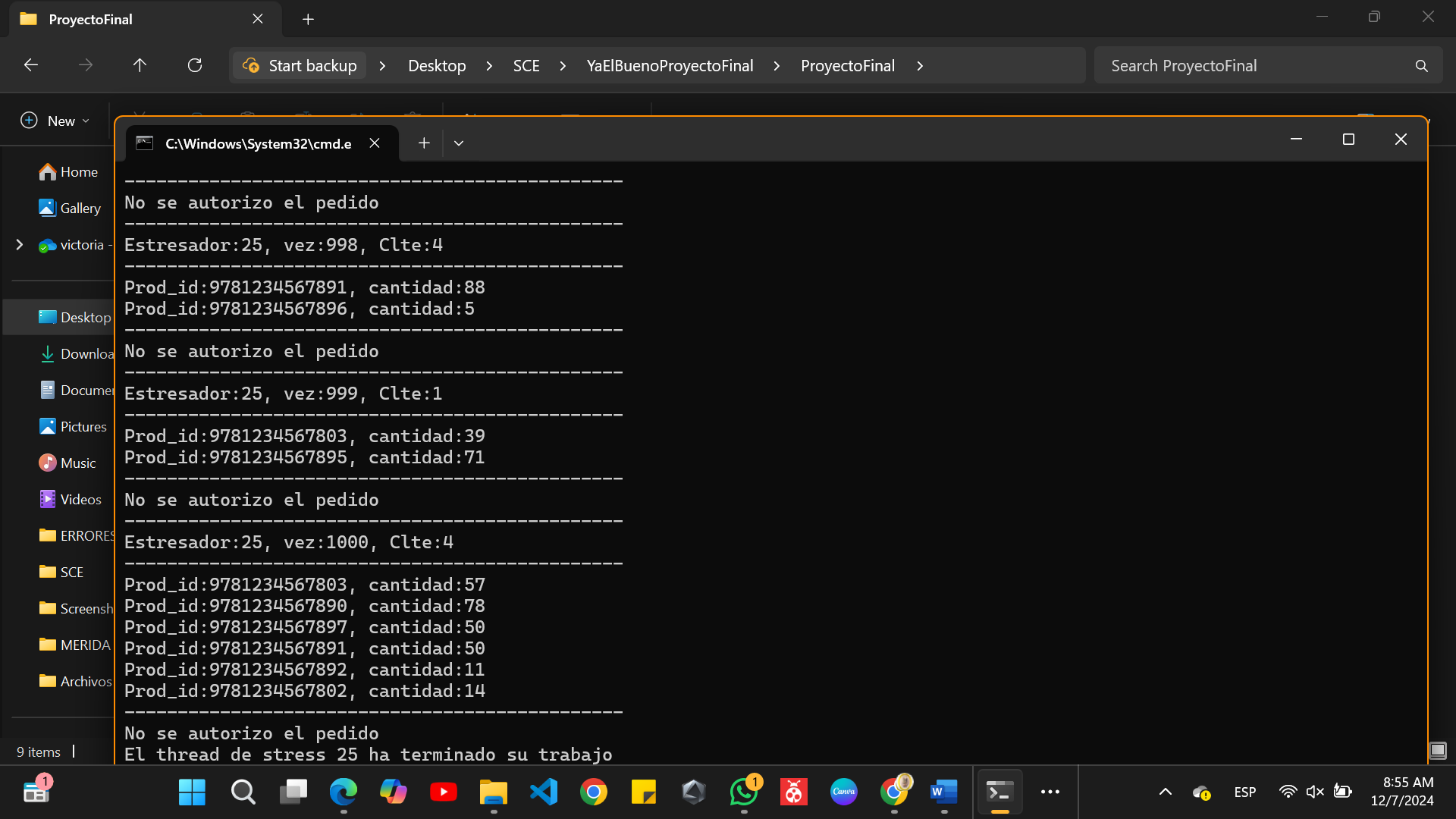
### Prueba WS\_Crédito



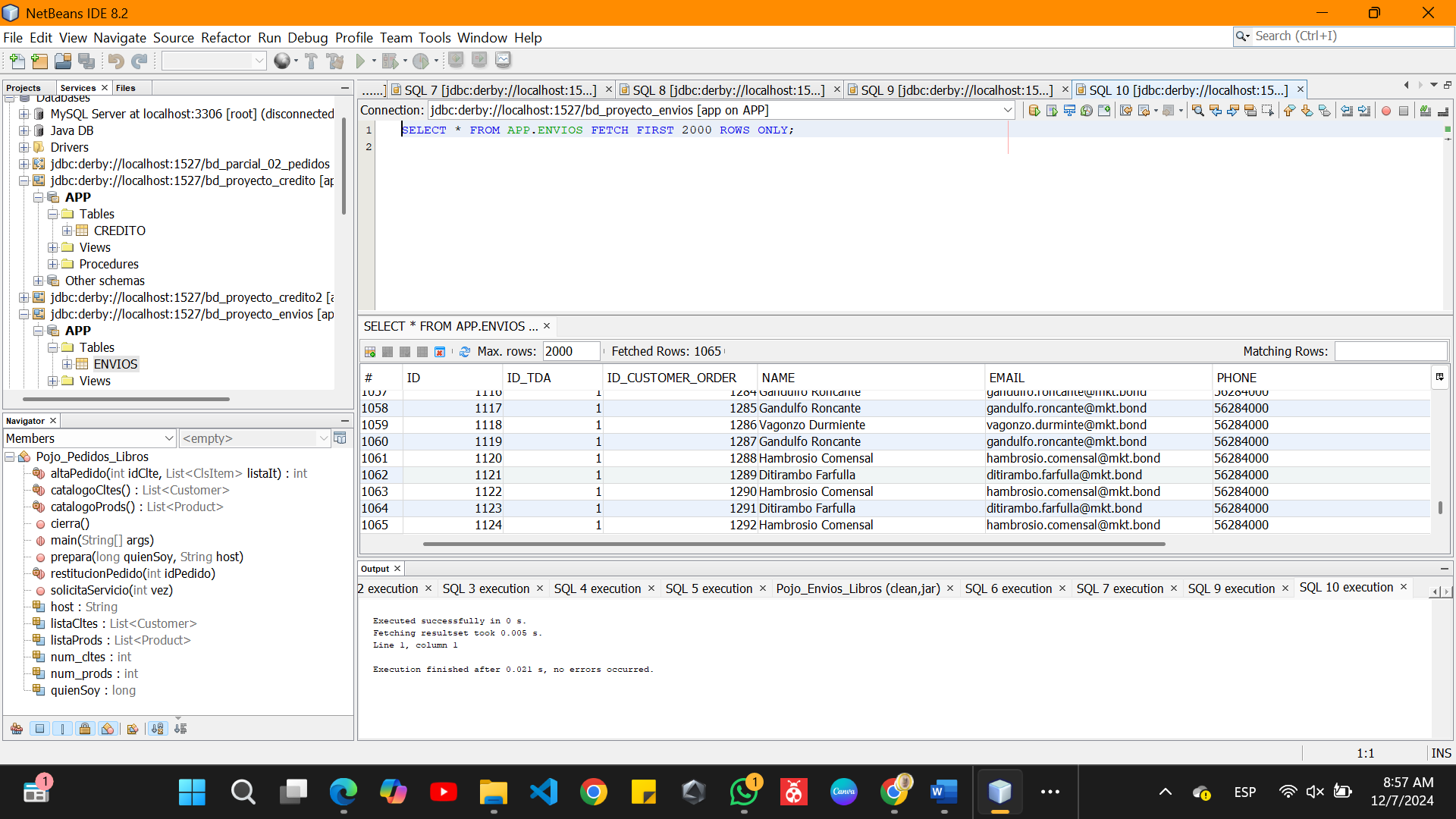
### Prueba Pojo\_Proyecto

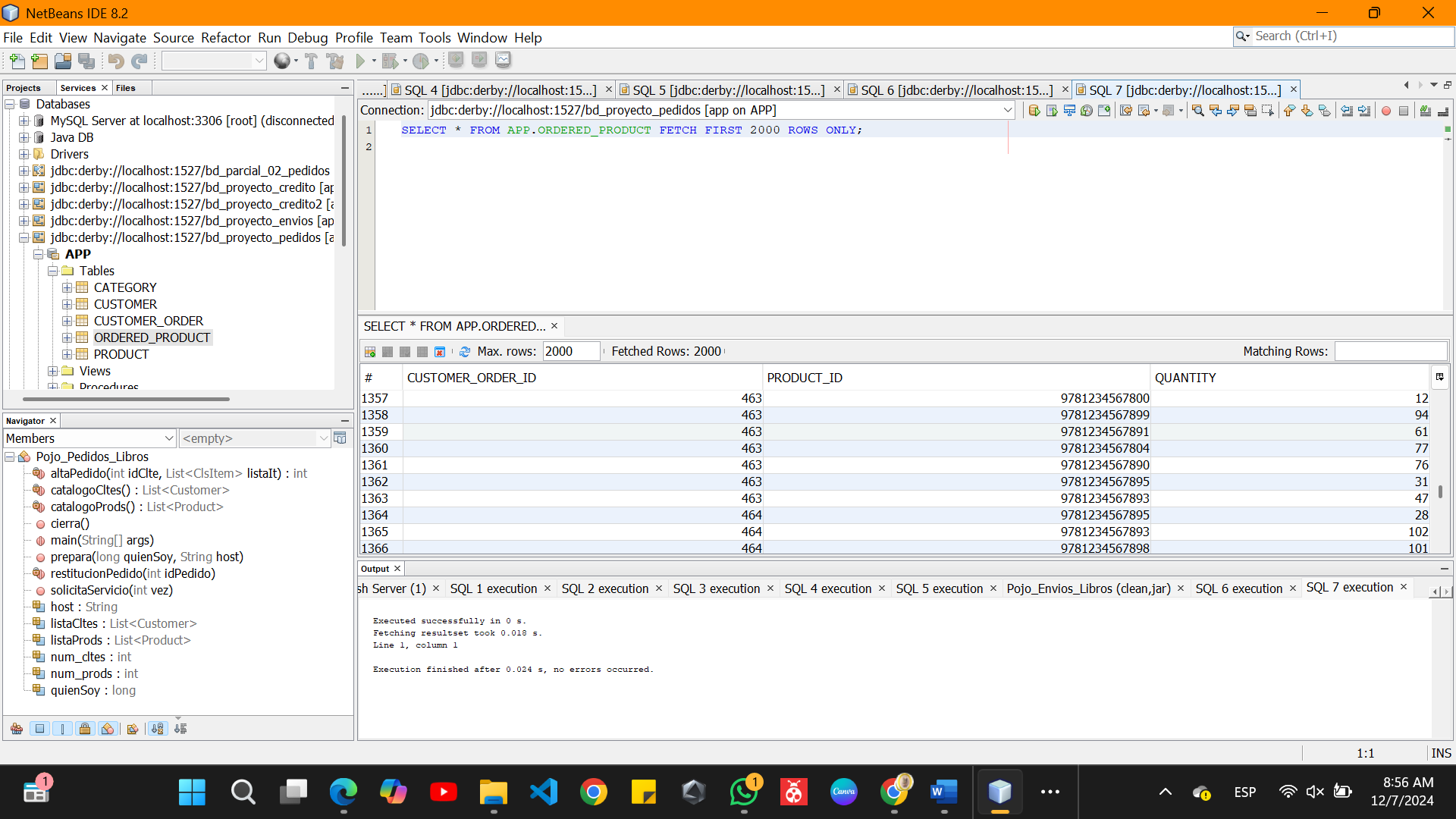
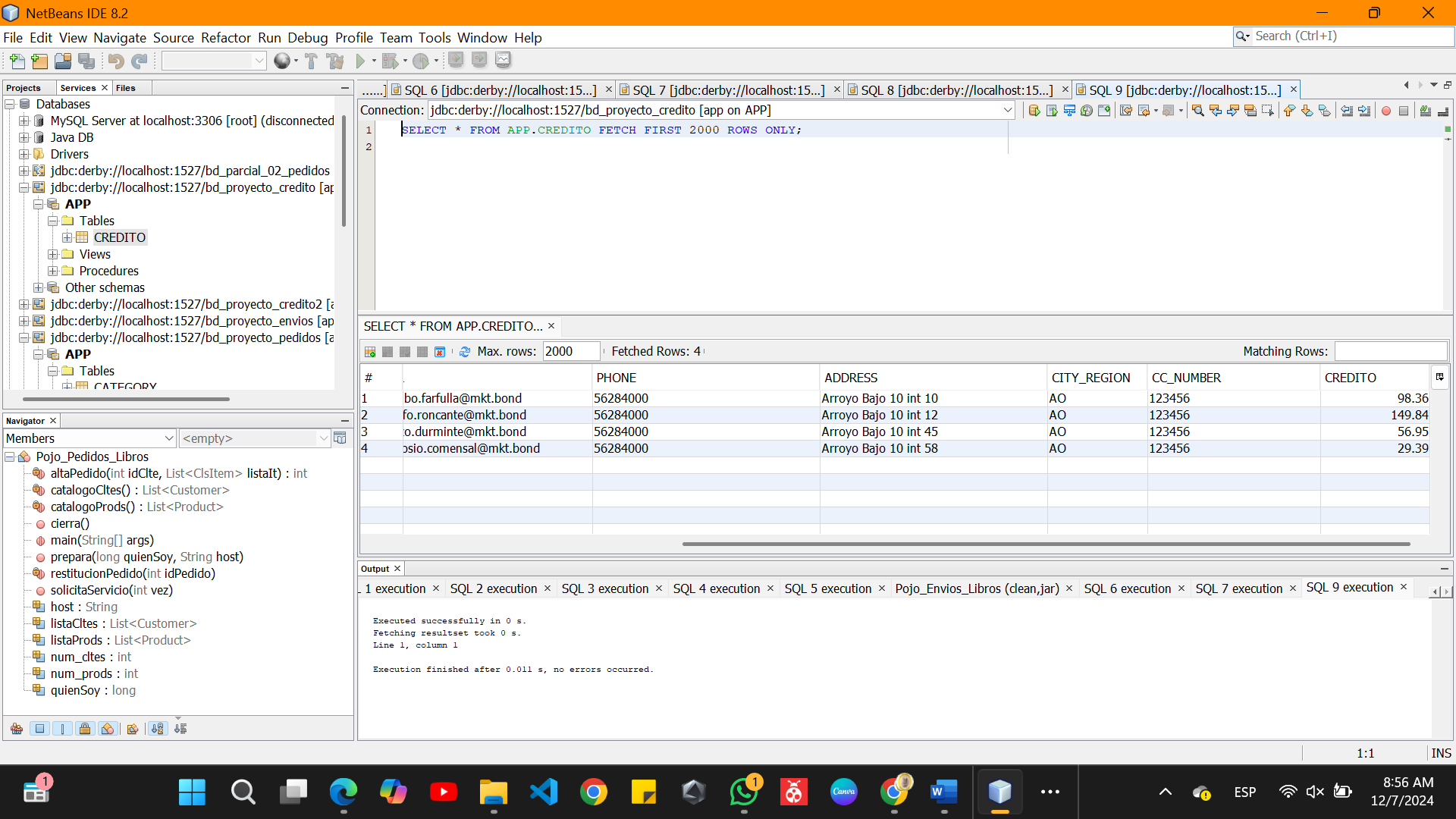


*Prueba de uso de CPU 1000 transacciones Pojo\_Proyecto*



*Prueba de la terminal*





*Prueba de modificación de tablas en Base de Datos*

# Arquitectura y versiones

La arquitectura del proyecto ha evolucionado desde versiones iniciales con funcionalidades básicas de cada Web Service (WS\_Pedidos, WS\_Credito, WS\_Envios) hasta una integración completa mediante BPEL y un sistema de estrés RMI desacoplado. A continuación, se presenta una tabla que detalla las versiones, los entregables de cada fase, y el cumplimiento de las pruebas funcionales y no funcionales.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Fecha de entrega** | **Componentes y Cambios Incluidos** | **Pruebas Funcionales** | **Pruebas No Funcionales** |
| V1.0 | 11/11/2024 | WS\_Pedidos funcional con operaciones básicas (alta de pedido). | Alta de pedido probada. | No aplica. |
| V1.1 | 13/11/2024 | WS\_Credito funcional con validaciones básicas de crédito. | Validación de crédito probada. | No aplica. |
| V1.2 | 15/11/2024 | WS\_Envios funcional con registro de envíos en cola JMS. | Registro de envío probado. | No aplica. |
| V2.0 | 27/11/2024 | Integración inicial mediante BPEL: WS\_Pedidos, WS\_Credito y WS\_Envios conectados. | Flujo integrado probado. | Uso básico de CPU y memoria. |
| V2.1 | 02/12/2024 | Optimización de BPEL con manejo de errores combinados (sin crédito, sin inventario). | Manejo de errores probado. | Uso moderado de CPU y memoria. |
| V3.0 | 06/12/2024 | Implementación del sistema de estrés RMI desacoplado. | Validación funcional de RMI probada. | Uso intensivo de CPU y memoria. |

# Web Service de Pedidos

## Relación con Reglas y Requerimientos de Negocio

|  |  |  |
| --- | --- | --- |
| **ID** | **Regla de Negocio** | **Descripción** |
| RN-01 | Validación de inventarios | El sistema debe verificar si hay existencias suficientes para los productos solicitados. |
| RN-02 | Creación de pedidos | El sistema debe permitir crear pedidos con una lista de productos y cantidades específicas, manejando pedidos en backorder si es necesario. |
| RN-03 | Actualización de inventarios | El inventario debe actualizarse en tiempo real con cada operación de alta de pedido. |
| RN-04 | Generación de números de confirmación | Cada pedido debe tener un número de confirmación único generado automáticamente. |
| RN-05 | Gestión de catálogo de clientes y productos | El sistema debe ofrecer métodos para consultar el catálogo de clientes y productos. |
| RN-06 | Consultas relacionadas con pedidos | El sistema debe permitir consultar el monto total de un pedido y el cliente asociado al mismo. |
| RN-07 | Manejo de errores | El sistema debe manejar casos donde los productos no existan o no haya suficiente inventario, retornando mensajes claros para estos casos. |
| RN-08 | Acceso a inventario | Los usuarios deben poder consultar la cantidad disponible de un producto específico en el inventario. |

## Pruebas Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PF-01 | Alta de pedido con cantidades normales | El pedido se crea correctamente, actualizando el inventario y generando un número de confirmación único. |
| PF-02 | Alta de pedido con cantidades negativas (backorder) | El pedido se crea con cantidades negativas y el inventario se actualiza adecuadamente. |
| PF-03 | Alta de pedido con productos inexistentes | Se devuelve un mensaje indicando que los productos no existen, y no se crea el pedido. |
| PF-04 | Consulta del catálogo de clientes | Devuelve la lista completa de clientes registrados. |
| PF-05 | Consulta del catálogo de productos | Devuelve la lista completa de productos disponibles en el inventario. |
| PF-06 | Consulta del monto de un pedido | Devuelve el monto total del pedido especificado. |
| PF-07 | Consulta del cliente asociado a un pedido | Devuelve los datos del cliente asociado al pedido especificado. |
| PF-08 | Consulta del inventario de un producto | Devuelve la cantidad disponible de un producto específico en el inventario. |

## Pruebas No Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PNF-01 | Prueba de carga: Alta de pedidos con 100 productos simultáneamente | El sistema debe soportar la carga, procesando todos los pedidos sin errores y actualizando los inventarios correctamente. |
| PNF-02 | Prueba de estrés: 100 consultas simultáneas al catálogo de productos | El sistema debe responder a todas las consultas dentro de un tiempo razonable. |
| PNF-03 | Escalabilidad: Incremento progresivo en el número de pedidos | El sistema debe mantener la estabilidad y funcionalidad al incrementar el número de pedidos procesados simultáneamente. |
| PNF-04 | Tolerancia a fallos: Base de datos no disponible | El sistema debe retornar un mensaje claro indicando la indisponibilidad del servicio y manejar el fallo sin comprometer las operaciones en curso. |
| PNF-05 | Seguridad: Acceso no autorizado | El sistema debe impedir accesos no autorizados a las operaciones críticas como la creación de pedidos o modificación de inventarios. |

# Web Service de Crédito

## Relación con Reglas y Requerimientos de Negocio

|  |  |  |
| --- | --- | --- |
| **ID** | **Regla de Negocio** | **Descripción** |
| RN-01 | Gestión de créditos | El sistema debe permitir consultar, crear, editar y eliminar créditos asociados a los clientes. |
| RN-02 | Autorización de crédito | El sistema debe verificar si un cliente tiene suficiente crédito para realizar una compra específica. |
| RN-03 | Manejo de excepciones | El sistema debe generar excepciones claras en los siguientes casos: cliente inexistente (ExcepNoExisteClte) o crédito insuficiente (ExcepNoCredito). |
| RN-04 | Operaciones básicas de gestión | Se deben soportar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para las entidades relacionadas con créditos. |

## Pruebas Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PF-01 | Crear un crédito para un cliente | Se registra correctamente un nuevo crédito en la base de datos asociado al cliente. |
| PF-02 | Editar un crédito existente | El crédito se actualiza correctamente en la base de datos. |
| PF-03 | Eliminar un crédito | El registro del crédito se elimina de la base de datos. |
| PF-04 | Consultar créditos asociados a un cliente | Devuelve la lista de créditos asociados al cliente solicitado. |
| PF-05 | Autorizar crédito con suficiente monto disponible | La operación autoriza el crédito y actualiza el monto restante correctamente. |
| PF-06 | Autorizar crédito con monto insuficiente | Se lanza la excepción ExcepNoCredito y la operación no se realiza. |
| PF-07 | Autorizar crédito para un cliente inexistente | Se lanza la excepción ExcepNoExisteClte indicando que el cliente no está registrado. |
| PF-08 | Consultar todos los créditos | Devuelve la lista completa de créditos registrados en el sistema. |

## Pruebas No Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PNF-01 | Prueba de carga: Autorizar 100 solicitudes de crédito simultáneamente | El sistema debe procesar todas las solicitudes sin errores y actualizar los créditos correctamente en un tiempo aceptable |
| PNF-02 | Prueba de estrés: 100 operaciones CRUD simultáneas | El sistema debe mantener la estabilidad y completar todas las operaciones sin fallos. |
| PNF-03 | Escalabilidad: Incremento progresivo en el número de clientes | El sistema debe funcionar correctamente con un aumento significativo en el número de clientes registrados y sus créditos asociados. |
| PNF-04 | Tolerancia a fallos: Base de datos no disponible | El sistema debe manejar el fallo con un mensaje claro para el usuario y permitir la recuperación sin comprometer las operaciones ya realizadas. |
| PNF-05 | Seguridad: Acceso no autorizado | El sistema debe bloquear accesos no autorizados y proteger la información sensible de los clientes y sus créditos. |

# Web Service de Envíos

## Relación con Reglas y Requerimientos de Negocio

|  |  |  |
| --- | --- | --- |
| **ID** | **Regla de Negocio** | **Descripción** |
| RN-01 | Gestión de envíos | El sistema debe permitir registrar, consultar, actualizar y eliminar los registros de envíos relacionados con los pedidos de los clientes. |
| RN-02 | Encolado de solicitudes de envío | El sistema debe permitir encolar solicitudes de envío mediante el uso de JMS para garantizar la comunicación asíncrona entre los componentes. |
| RN-03 | Generación de mensajes JMS para envío de datos | Las solicitudes de envío deben ser enviadas como mensajes JMS con los datos completos del envío (cliente, pedido, dirección, etc.). |
| RN-04 | Confirmación de recepción de solicitud de envío | El sistema debe confirmar que la solicitud de envío ha sido correctamente encolada para su procesamiento posterior. |
| RN-05 | Integración con base de datos de envíos | Las solicitudes de envío deben ser persistidas en la base de datos para su seguimiento y auditoría posterior. |

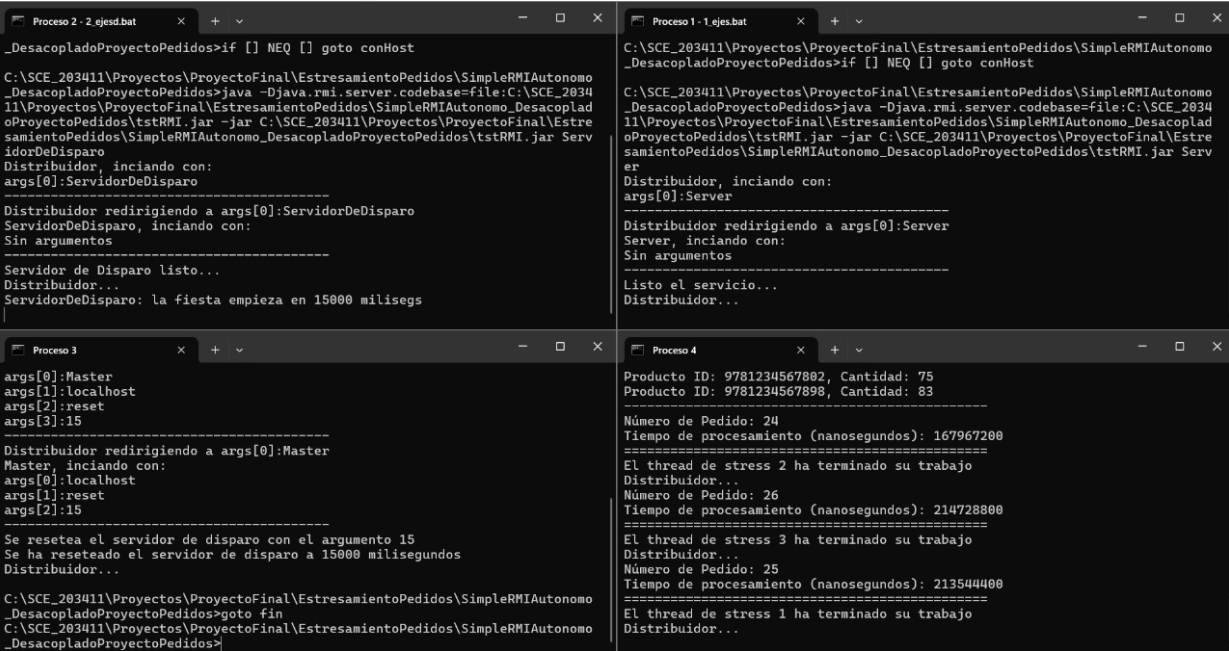
## Pruebas Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PF-01 | Crear un envío | El sistema debe crear correctamente un nuevo registro de envío en la base de datos y enviarlo a la cola de mensajes JMS. |
| PF-02 | Editar un registro de envío | El sistema debe actualizar un registro de envío en la base de datos y reflejar los cambios en la cola de mensajes JMS. |
| PF-03 | Eliminar un envío | El sistema debe eliminar correctamente un registro de envío de la base de datos y la cola de mensajes. |
| PF-04 | Consultar un registro de envío | El sistema debe devolver el registro de envío solicitado con los datos correctos. |
| PF-05 | Consultar todos los registros de envíos | El sistema debe devolver una lista con todos los registros de envíos almacenados en la base de datos. |
| PF-06 | Encolar una solicitud de envío a la cola JMS | El sistema debe encolar correctamente el mensaje de solicitud de envío en la cola JMS y registrar los datos correspondientes. |
| PF-07 | Confirmación de recepción de solicitud de envío | El sistema debe confirmar que la solicitud de envío ha sido correctamente encolada y registrada. |

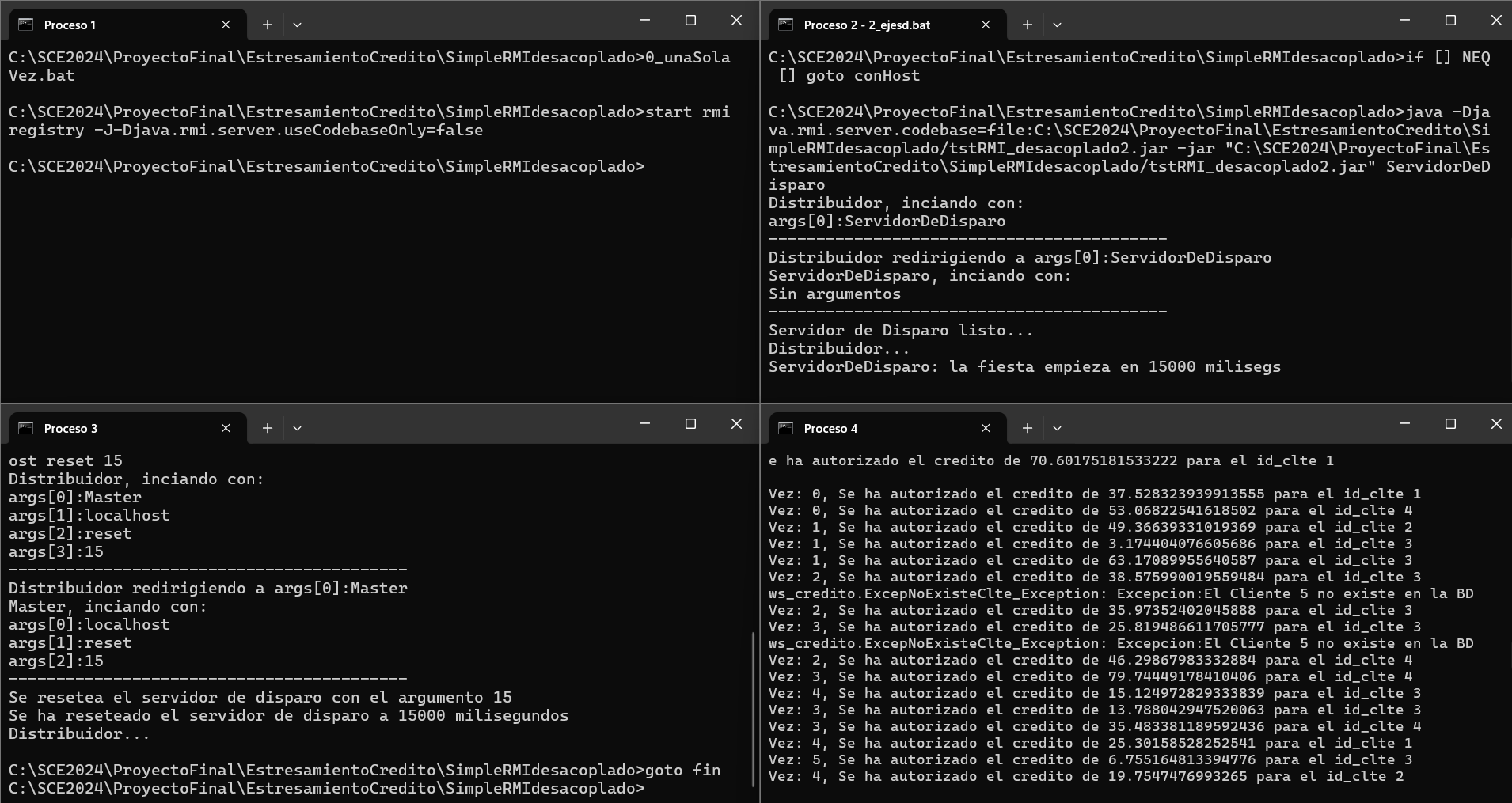
## Pruebas No Funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Descripción** | **Resultado Esperado** |
| PNF-01 | Prueba de carga: Encolar 100 solicitudes de envío simultáneamente | El sistema debe procesar todas las solicitudes sin errores, manteniendo tiempos de respuesta adecuados |
| PNF-02 | Prueba de estrés: 100 operaciones CRUD simultáneas | El sistema debe mantener la estabilidad y completar todas las operaciones sin caídas ni errores durante el proceso de encolado y consulta. |
| PNF-03 | Escalabilidad: Incremento progresivo en el número de solicitudes | El sistema debe funcionar correctamente con un aumento en el número de solicitudes de envío sin afectar el rendimiento ni la disponibilidad. |
| PNF-04 | Tolerancia a fallos: JMS no disponible | El sistema debe manejar el fallo de la cola JMS con un mensaje claro para el usuario y permitir la recuperación sin comprometer las operaciones ya realizadas. |
| PNF-05 | Seguridad: Acceso no autorizado | El sistema debe bloquear accesos no autorizados a la operación de encolado de solicitudes de envío y proteger la información sensible de los clientes. |

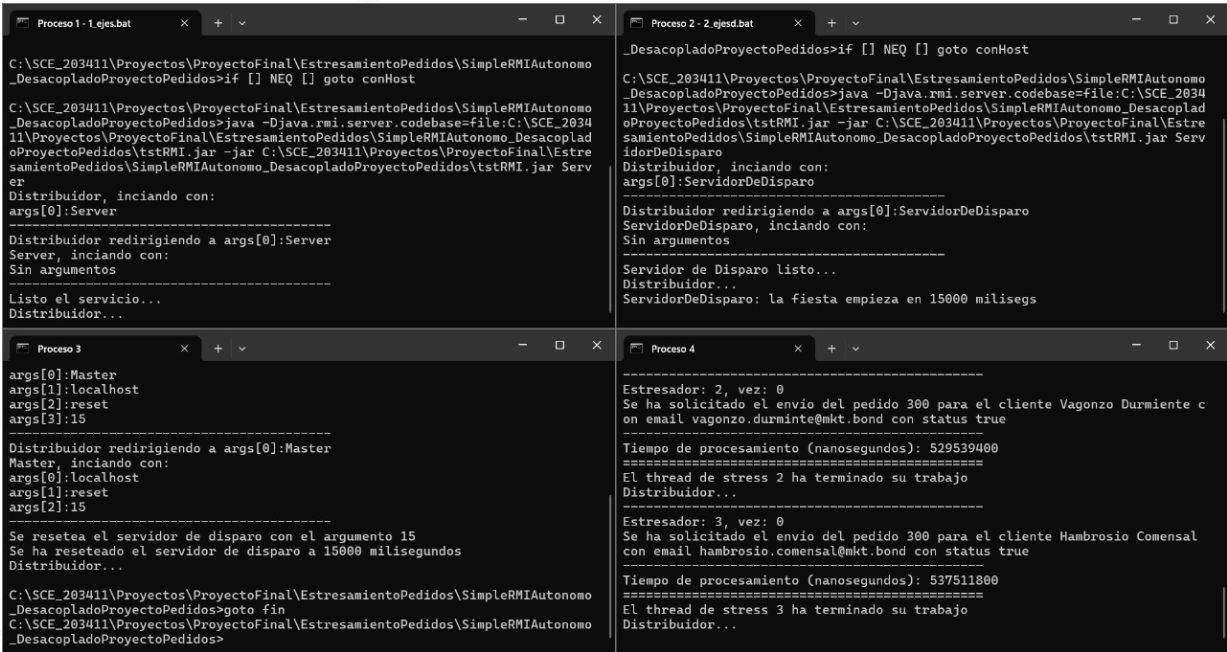
# Estrés del Web Service Pedidos



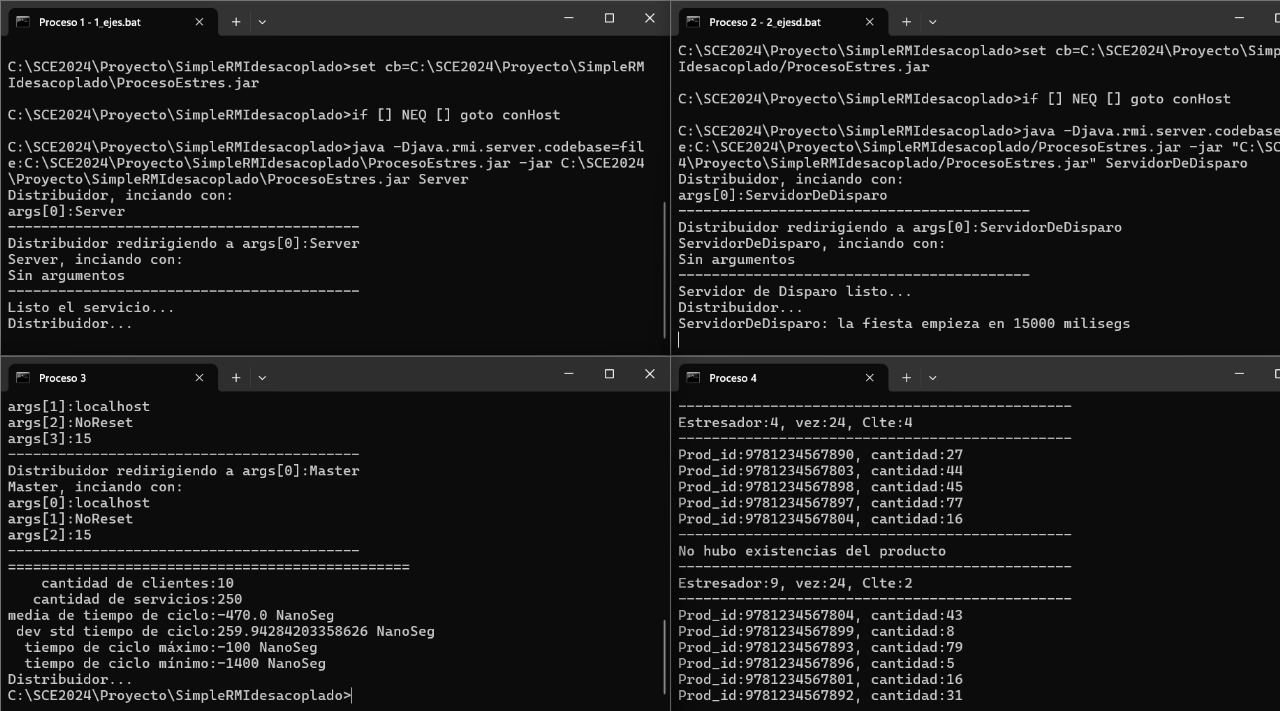
# Estrés del Web Service Crédito



# Estrés del Web Service Envíos



# Estres general de la aplicación



# Anexos

## Scripts de inicialización de BD de los servicios básicos

## Código de web services y acceso a BD

### Base de Datos de Pedidos

1. connect 'jdbc:derby://localhost:1527/bd\_proyecto\_pedidos;user=app;password=app';

2.

3. -- Eliminar tablas en orden correcto

4. DROP TABLE ordered\_product;

5. DROP TABLE customer\_order;

6. DROP TABLE product;

7. DROP TABLE category;

8. DROP TABLE customer;

9.

10. -- Tabla CUSTOMER

11. CREATE TABLE customer (

12. id INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)

13. CONSTRAINT CUSTOMER\_PK PRIMARY KEY,

14. name VARCHAR(45) NOT NULL,

15. email VARCHAR(45) NOT NULL,

16. phone VARCHAR(45) NOT NULL,

17. address VARCHAR(45) NOT NULL,

18. city\_region VARCHAR(2) NOT NULL,

19. cc\_number VARCHAR(19) NOT NULL

20. );

21.

22. INSERT INTO customer (name, email, phone, address, city\_region, cc\_number) VALUES

23. ('Ditirambo Farfulla', '[ditirambo.farfulla@mkt.bond](mailto:ditirambo.farfulla@mkt.bond)', '56284000', 'Arroyo Bajo 10 int 10', 'AO', '123456'),

24. ('Gandulfo Roncante', '[gandulfo.roncante@mkt.bond](mailto:gandulfo.roncante@mkt.bond)', '56284000', 'Arroyo Bajo 10 int 12', 'AO', '123456'),

25. ('Vagonzo Durmiente', '[vagonzo.durminte@mkt.bond](mailto:vagonzo.durminte@mkt.bond)', '56284000', 'Arroyo Bajo 10 int 45', 'AO', '123456'),

26. ('Hambrosio Comensal', '[hambrosio.comensal@mkt.bond](mailto:hambrosio.comensal@mkt.bond)', '56284000', 'Arroyo Bajo 10 int 58', 'AO', '123456');

27.

28. -- Tabla CUSTOMER\_ORDER

29. CREATE TABLE customer\_order (

30. id INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)

31. CONSTRAINT CUSTOMER\_ORDER\_PK PRIMARY KEY,

32. amount DECIMAL(6,2) NOT NULL,

33. date\_created TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

34. confirmation\_number INT NOT NULL,

35. customer\_id INT NOT NULL CONSTRAINT CUST\_ORD\_FK REFERENCES customer

36. );

37.

38. -- Tabla CATEGORY

39. CREATE TABLE category (

40. id INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1)

41. CONSTRAINT CATEGORY\_PK PRIMARY KEY,

42. name VARCHAR(45) NOT NULL

43. );

44.

45. INSERT INTO category (name) VALUES

46. ('terror'),

47. ('fantasia'),

48. ('romance'),

49. ('ciencia\_ficcion');

50.

51. -- Tabla PRODUCT

52. CREATE TABLE product (

53. id BIGINT NOT NULL,

54. name VARCHAR(45) NOT NULL,

55. price DECIMAL(5,2) NOT NULL,

56. description VARCHAR(50),

57. last\_update TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

58. category\_id INT NOT NULL,

59. existencia INT NOT NULL,

60. PRIMARY KEY (id),

61. CONSTRAINT fk\_product\_category FOREIGN KEY (category\_id) REFERENCES category (id)

62. );

63.

64.

65.

66. -- Insertar productos

67. INSERT INTO product (id, name, price, description, last\_update, category\_id, existencia)

68. VALUES

69. (9781234567890, 'El Resplandor', 12.50, 'Un libro de terror psicologico', '2020-10-06 09:00:00', 1, 500),

70. (9781234567891, 'Harry Potter y la Piedra Filosofal', 15.99, 'Un libro de fantasia sobre un nino mago', '2020-10-06 09:00:00', 2, 1000),

71. (9781234567892, 'Orgullo y Prejuicio', 9.99, 'Un clasico romance de Jane Austen', '2020-10-06 09:00:00', 3, 750),

72. (9781234567893, 'Cien Anos de Soledad', 11.50, 'Una novela de realismo magico', '2020-10-06 09:00:00', 4, 300),

73. (9781234567894, 'El It', 13.30, 'Un libro de terror sobrenatural', '2020-10-06 09:00:00', 1, 400),

74. (9781234567895, 'La Historia Interminable', 10.99, 'Un libro de fantasia epica', '2020-10-06 09:00:00', 2, 600),

75. (9781234567896, 'La Casa de los Espiritus', 8.50, 'Novela de romance con tintes de misterio', '2020-10-06 09:00:00', 3, 800),

76. (9781234567897, 'Fahrenheit 451', 7.75, 'Un libro de ciencia ficcion distopica', '2020-10-06 09:00:00', 4, 200),

77. (9781234567898, 'La Sombra del Viento', 18.00, 'Un thriller de romance con elementos goticos', '2020-10-06 09:00:00', 3, 650),

78. (9781234567899, 'Dracula', 10.00, 'Un clasico de terror gotico', '2020-10-06 09:00:00', 1, 450),

79. (9781234567800, 'El Hobbit', 14.99, 'Una fantasia sobre una aventura en la Tierra Media', '2020-10-06 09:00:00', 2, 1200),

80. (9781234567801, 'Eclipse', 12.75, 'Una historia de romance y vampiros', '2020-10-06 09:00:00', 3, 500),

81. (9781234567802, '1984', 9.00, 'Una novela de ciencia ficcion distopica', '2020-10-06 09:00:00', 4, 350),

82. (9781234567803, 'El Exorcista', 13.00, 'Un thriller de terror sobrenatural', '2020-10-06 09:00:00', 1, 600),

83. (9781234567804, 'La Tierra de las Sombras', 16.50, 'Una fantasia oscura de aventuras', '2020-10-06 09:00:00', 2, 700);

84.

85. -- Tabla ORDERED\_PRODUCT

86. CREATE TABLE ordered\_product (

87. customer\_order\_id INT NOT NULL,

88. product\_id BIGINT NOT NULL,

89. quantity SMALLINT NOT NULL DEFAULT 1,

90. PRIMARY KEY (customer\_order\_id, product\_id),

91. CONSTRAINT fk\_ordered\_product\_customer\_order FOREIGN KEY (customer\_order\_id) REFERENCES customer\_order (id),

92. CONSTRAINT fk\_ordered\_product\_product FOREIGN KEY (product\_id) REFERENCES product (id)

93. );

94.

95. disconnect;

96.

### Base de Datos de Crédito

1. connect 'jdbc:derby://localhost:1527/bd\_proyecto\_credito;user=app;password=app';

2.

3. DROP TABLE credito;

4.

5. -------------------------------

6. -- CREDITO

7. -------------------------------

8. CREATE TABLE credito

9. (

10. id INT NOT NULL GENERATED ALWAYS AS IDENTITY

11. (START WITH 1 ,INCREMENT BY 1)

12. CONSTRAINT CUSTOMER\_PK PRIMARY KEY,

13. name VARCHAR(45) NOT NULL ,

14. email VARCHAR(45) NOT NULL ,

15. phone VARCHAR(45) NOT NULL ,

16. address VARCHAR(45) NOT NULL ,

17. city\_region VARCHAR(2) NOT NULL ,

18. cc\_number VARCHAR(19) NOT NULL,

19. credito DECIMAL(10,2) NOT NULL

20. );

21.

22. INSERT INTO credito (name,email,phone,address,city\_region,cc\_number,credito) VALUES

23. ('Ditirambo Farfulla','ditirambo.farfulla@mkt.bond','56284000','Arroyo Bajo 10 int 10','AO','123456',100000.0),

24. ('Gandulfo Roncante','gandulfo.roncante@mkt.bond','56284000','Arroyo Bajo 10 int 12','AO','123456',100000.0),

25. ('Vagonzo Durmiente','vagonzo.durminte@mkt.bond','56284000','Arroyo Bajo 10 int 45','AO','123456',100000.0),

26. ('Hambrosio Comensal','hambrosio.comensal@mkt.bond','56284000','Arroyo Bajo 10 int 58','AO','123456',100000.0);

27.

28.

29.

30. disconnect;

### Base de Datos de Envíos

1. connect 'jdbc:derby://localhost:1527/bd\_proyecto\_envios;user=app;password=app';

2.

3. DROP TABLE envios;

4.

5. -------------------------------

6. -- ENVIOS

7. -------------------------------

8. CREATE TABLE envios

9. (

10. id INT NOT NULL GENERATED ALWAYS AS IDENTITY

11. (START WITH 1 ,INCREMENT BY 1)

12. CONSTRAINT CUSTOMER\_PK PRIMARY KEY,

13. id\_Tda INT NOT NULL,

14. id\_CUSTOMER\_ORDER INT NOT NULL,

15. name VARCHAR(45) NOT NULL ,

16. email VARCHAR(45) ,

17. phone VARCHAR(45) NOT NULL ,

18. address VARCHAR(45) NOT NULL ,

19. city\_region VARCHAR(2) NOT NULL

20. );

21.

22. -------------------------------------

23. -- Ejemplos

24. -------------------------------------

25. --'Ditirambo Farfulla','ditirambo.farfulla@mkt.bond','56284000','Arroyo Bajo 10 int 10','AO'

26. --'Gandulfo Roncante','gandulfo.roncante@mkt.bond','56284000','Arroyo Bajo 10 int 12','AO'

27. --'Vagonzo Durmiente','vagonzo.durminte@mkt.bond','56284000','Arroyo Bajo 10 int 45','AO'

28. --'Hambrosio Comensal','hambrosio.comensal@mkt.bond','56284000','Arroyo Bajo 10 int 58','AO'

29. -------------------------------------

30.

31.

32.

33. disconnect;

34.

## Código de web services y acceso a BD

### Código de WS\_Pedido

1. Código de WS\_Pedido

2. package wspedido;

3.

4. import entidades.Customer;

5. import entidades.Product;

6. import fachadas.CustomerFacade;

7. import fachadas.CustomerOrderFacade;

8. import fachadas.OrderedProductFacade;

9. import fachadas.ProductFacade;

10. import java.math.BigDecimal;

11. import java.math.MathContext;

12. import java.util.logging.Level;

13. import javax.ejb.EJB;

14. import javax.jws.WebService;

15. import javax.jws.WebMethod;

16. import javax.jws.WebParam;

17. import java.util.logging.Logger;

18. import java.util.List;

19. import java.util.ArrayList;

20.

21. import javax.persistence.EntityManager;

22. import javax.persistence.LockModeType;

23.

24. /\*\*

25. \*

26. \* @author RGAMBOAH

27. \*/

28. @WebService(serviceName = "WSPedido")

29. public class WSPedido {

30.

31. @EJB

32. private OrderedProductFacade orderedProductFacade;

33. @EJB

34. private CustomerOrderFacade customerOrderFacade;

35. @EJB

36. private ProductFacade productFacade;

37. @EJB

38. private CustomerFacade customerFacade;

39.

40. /\*\*

41. \* This is a sample web service operation

42. \*/

43. @WebMethod(operationName = "hello")

44. public String hello(@WebParam(name = "name") String txt) {

45. return "Hello " + txt + " !";

46. }

47.

48.

49. @WebMethod(operationName = "altaPedido")

50. public int altaPedido(@WebParam(name = "id\_clte") int id\_clte,

51. @WebParam(name = "lista\_it") List<ClsItem> lista\_it)

52. {

53. List<entidades.OrderedProduct> lista\_orderedProducts = new ArrayList<>();

54. entidades.OrderedProduct ordered\_product;

55. entidades.Product prod;

56. int num\_conf;

57. int num\_pedido;

58.

59. BigDecimal bd\_monto\_pedido = BigDecimal.ZERO;

60. BigDecimal bd\_monto\_item;

61.

62. for (ClsItem it : lista\_it)

63. {

64. Logger.getLogger(this.getClass().getName()).log(Level.INFO,

65. "Product\_id:" + it.getId\_prod() + ", cantidad:" + it.getCantidad());

66.

67. // Verifica si es un pedido en backorder (cantidad negativa)

68. if (it.getCantidad() < 0) {

69. // Crea el pedido con cantidades negativas

70. ordered\_product = new entidades.OrderedProduct();

71. prod = productFacade.find(new Long(it.getId\_prod()));

72. ordered\_product.setProduct(prod);

73. ordered\_product.setQuantity((short) it.getCantidad()); // Almacena cantidad negativa

74. lista\_orderedProducts.add(ordered\_product);

75.

76. // Actualiza el inventario con la cantidad negativa para revertirlo

77. productFacade.actualizaExistencia(it.getId\_prod(), it.getCantidad());

78.

79. // Calcula el monto con el valor absoluto de la cantidad

80. bd\_monto\_item = prod.getPrice().multiply(BigDecimal.valueOf(-it.getCantidad()));

81. bd\_monto\_item = bd\_monto\_item.setScale(2, BigDecimal.ROUND\_HALF\_UP);

82. bd\_monto\_pedido = bd\_monto\_pedido.add(bd\_monto\_item);

83. } else {

84. // Pedido normal: verifica inventario y ajusta cantidad posible

85. int cantidad\_posible = productFacade.actualizaExistencia(it.getId\_prod(), it.getCantidad());

86.

87. if (cantidad\_posible > 0) {

88. ordered\_product = new entidades.OrderedProduct();

89. prod = productFacade.find(new Long(it.getId\_prod()));

90. ordered\_product.setProduct(prod);

91. ordered\_product.setQuantity((short) cantidad\_posible);

92. lista\_orderedProducts.add(ordered\_product);

93.

94. Logger.getLogger(this.getClass().getName()).log(Level.INFO,

95. "Product\_id:" + prod.getId() + ", " + prod.getName() + ", "

96. + prod.getDescription() + ", " + prod.getCategoryId().getName() + ", "

97. + prod.getPrice());

98.

99. bd\_monto\_item = prod.getPrice().multiply(BigDecimal.valueOf(cantidad\_posible));

100. bd\_monto\_item = bd\_monto\_item.setScale(2, BigDecimal.ROUND\_HALF\_UP);

101. bd\_monto\_pedido = bd\_monto\_pedido.add(bd\_monto\_item);

102. } else {

103. Logger.getLogger(this.getClass().getName()).log(Level.INFO,

104. "Clave de producto " + it.getId\_prod() + " INEXISTENTE o sin existencias");

105. }

106. }

107. }

108.

109. if (lista\_orderedProducts.size() > 0) // hay items en el pedido o backorder

110. {

111. entidades.Customer clte = customerFacade.find(new Integer(id\_clte));

112. Logger.getLogger(this.getClass().getName()).log(Level.SEVERE,

113. "Pedido del cliente:" + clte.getId() + ", " + clte.getName() + " por " + bd\_monto\_pedido);

114.

115. entidades.CustomerOrder customer\_order = new entidades.CustomerOrder();

116. customer\_order.setCustomerId(clte);

117. java.util.Date d = new java.util.Date();

118. customer\_order.setDateCreated(d);

119. customer\_order.setAmount(bd\_monto\_pedido);

120. //customer\_order.setOrderedProductList(lista\_orderedProducts);

121.

122. num\_conf = customerOrderFacade.next\_conf\_number();

123. customer\_order.setConfirmationNumber(num\_conf);

124. customerOrderFacade.create(customer\_order);

125.

126. num\_pedido = customer\_order.getId();

127.

128. entidades.OrderedProductPK oppk;

129.

130. List<entidades.OrderedProduct> listaBuena = new ArrayList<>();

131. for (entidades.OrderedProduct op : lista\_orderedProducts)

132. {

133. oppk = new entidades.OrderedProductPK();

134. oppk.setCustomerOrderId(customer\_order.getId());

135. oppk.setProductId(op.getProduct().getId());

136. op.setOrderedProductPK(oppk);

137. op.setCustomerOrder(customer\_order);

138.

139. orderedProductFacade.create(op);

140. listaBuena.add(op);

141. }

142. customer\_order.setOrderedProductList(listaBuena);

143. customerOrderFacade.edit(customer\_order);

144.

145. }

146. else

147. {

148. num\_pedido = 0; // no hay items a surtir en el pedido

149. }

150.

151. return num\_pedido;

152. }

153.

154.

155. // =========================================================================

156. // Para el catálogo de clientes y de productos

157. // =========================================================================

158. /\*\*

159. \* Web service operation

160. \* @return

161. \*/

162. @WebMethod(operationName = "catalogoCltes")

163. public java.util.List<entidades.Customer> catalogoCltes()

164. {

165. return customerFacade.findAll();

166. }

167.

168. /\*\*

169. \* Web service operation

170. \* @return

171. \*/

172. @WebMethod(operationName = "catalogoProds")

173. public java.util.List<entidades.Product> catalogoProds()

174. {

175. return productFacade.findAll();

176. }

177.

178. // =============================================================================

179. // Nuevos servicios

180. // =============================================================================

181. /\*\*

182. \* Web service operation

183. \*/

184. @WebMethod(operationName = "montoPedido")

185. public BigDecimal montoPedido(@WebParam(name = "num\_pedido") int num\_pedido)

186. {

187. java.math.BigDecimal monto\_del\_pedido;

188.

189. monto\_del\_pedido = customerOrderFacade.find(num\_pedido).getAmount();

190.

191. return monto\_del\_pedido;

192. }

193.

194. /\*\*

195. \* Web service operation

196. \*/

197. @WebMethod(operationName = "clteDelPedido")

198. public Customer clteDelPedido(@WebParam(name = "num\_pedido") int num\_pedido)

199. {

200. return customerOrderFacade.find(num\_pedido).getCustomerId();

201. }

202.

203. // =============================================================================

204. // Fin de Nuevos servicio

205. // =============================================================================

206.

207.

208. //Metodo agregado por nuestra parte

209. /\*\*

210. \* Web service operation para obtener la cantidad de inventario disponible de un producto.

211. \* @param idProd ID del producto

212. \* @return Cantidad de inventario disponible

213. \*/

214. @WebMethod(operationName = "obtenerInventario")

215. public int obtenerInventario(@WebParam(name = "idProd") long idProd) {

216. entidades.Product producto = productFacade.find(idProd);

217. if (producto != null) {

218. return producto.getExistencia(); // Usamos el campo existencia para la cantidad de inventario

219. } else {

220. Logger.getLogger(this.getClass().getName()).log(Level.WARNING, "Producto con ID " + idProd + " no encontrado.");

221. return 0; // Retornamos 0 si el producto no existe

222. }

223. }

224.

225. // Método privado para verificar la disponibilidad de inventario para cada item en la lista

226. private boolean verificarInventario(List<wspedido.ClsItem> listaIt) {

227.

228. for (wspedido.ClsItem item : listaIt) {

229. long idProd = item.getId\_prod();

230. int cantidadRequerida = item.getCantidad();

231.

232. // Supongamos que tenemos un método en el servicio de pedidos que verifica la cantidad en inventario

233. entidades.Product producto = productFacade.find(idProd);

234.

235. int cantidadDisponible = producto.getExistencia(); // Método a implementar en el servicio de pedidos

236.

237. if (cantidadDisponible < cantidadRequerida) {

238. return false;

239. }

240. }

241. return true;

242. }

243.

244. // Método privado para verificar la disponibilidad de credito para cada item en la lista

245. private BigDecimal verificarCredito(List<wspedido.ClsItem> listaIt) {

246. BigDecimal suma = new BigDecimal(0);

247.

248. for (wspedido.ClsItem item : listaIt) {

249.

250. long idProd = item.getId\_prod();

251. entidades.Product producto = productFacade.find(idProd);

252. int cantidadRequerida = item.getCantidad();

253. BigDecimal precioporitem = producto.getPrice();

254.

255. // Supongamos que tenemos un método en el servicio de pedidos que verifica la cantidad en inventario

256. suma = suma.add(BigDecimal.valueOf(cantidadRequerida).multiply(precioporitem));

257.

258. // Método a implementar en el servicio de pedidos

259.

260.

261. }

262. return suma;

263. }

264.

265. //Metodo para la restitución de productos de la base de datos

266.

267. /\*\*

268. \* Web service operation

269. \*/

270. @WebMethod(operationName = "restitucionPedido")

271. public void restitucionPedido(@WebParam(name = "idPedido") int idPedido) {

272. //TODO write your implementation code here:

273.

274. entidades.CustomerOrder co = customerOrderFacade.find(idPedido);

275.

276. entidades.Product prod;

277. if(co != null)

278. {

279. java.util.List<entidades.OrderedProduct> items = (java.util.List<entidades.OrderedProduct>) co.getOrderedProductList();

280. for( entidades.OrderedProduct op : items )

281. {

282. prod = productFacade.find(op.getProduct().getId());

283. productFacade.agregaExistencia( prod.getId(), op.getQuantity());

284.

285. }

286.

287. }

288. }

289.

290.

291. }

292.

### Código WS\_Crédito

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package ws\_credito\_libros;

7.

8. import entidades.ExcepNoExisteClte;

9. import entidades.ExcepNoCredito;

10. import entidades.Credito;

11. import fachadas.CreditoFacade;

12. import java.math.BigDecimal;

13. import java.util.List;

14. import javax.ejb.EJB;

15. import javax.jws.Oneway;

16. import javax.jws.WebMethod;

17. import javax.jws.WebParam;

18. import javax.jws.WebService;

19.

20. /\*\*

21. \*

22. \* @author RGAMBOAH

23. \*/

24. @WebService(serviceName = "WSCredito")

25. public class WS\_Credito\_Libros {

26.

27. @EJB

28. private CreditoFacade ejbRef;// Add business logic below. (Right-click in editor and choose

29. // "Insert Code > Add Web Service Operation")

30.

31. @WebMethod(operationName = "create")

32. @Oneway

33. public void create(@WebParam(name = "entity") Credito entity) {

34. ejbRef.create(entity);

35. }

36.

37. @WebMethod(operationName = "edit")

38. @Oneway

39. public void edit(@WebParam(name = "entity") Credito entity) {

40. ejbRef.edit(entity);

41. }

42.

43. @WebMethod(operationName = "remove")

44. @Oneway

45. public void remove(@WebParam(name = "entity") Credito entity) {

46. ejbRef.remove(entity);

47. }

48.

49. @WebMethod(operationName = "find")

50. public Credito find(@WebParam(name = "id") Object id) {

51. return ejbRef.find(id);

52. }

53.

54. @WebMethod(operationName = "findAll")

55. public List<Credito> findAll() {

56. return ejbRef.findAll();

57. }

58.

59. @WebMethod(operationName = "findRange")

60. public List<Credito> findRange(@WebParam(name = "range") int[] range) {

61. return ejbRef.findRange(range);

62. }

63.

64. @WebMethod(operationName = "count")

65. public int count() {

66. return ejbRef.count();

67. }

68.

69. /\*\*

70. \* Web service operation

71. \* @param id\_clte

72. \* @param monto

73. \* @return

74. \* @throws entidades.ExcepNoCredito

75. \* @throws entidades.ExcepNoExisteClte

76. \*/

77. @WebMethod(operationName = "autoriza")

78. public boolean autoriza(@WebParam(name = "id\_clte") final int id\_clte, @WebParam(name = "monto") final BigDecimal monto) throws ExcepNoCredito, ExcepNoExisteClte {

79.

80. boolean blnAutorizado = ejbRef.actualizaCredito(id\_clte, monto);

81.

82. return blnAutorizado;

83. }

84.

85. @WebMethod(operationName = "autoriza\_double")

86. public boolean autoriza\_double(@WebParam(name = "id\_clte") final int id\_clte, @WebParam(name = "dbl\_monto") final double dbl\_monto) throws ExcepNoCredito, ExcepNoExisteClte {

87.

88. BigDecimal monto = BigDecimal.valueOf(dbl\_monto);

89.

90. monto.setScale(2,BigDecimal.ROUND\_HALF\_UP);

91.

92. boolean blnAutorizado = ejbRef.actualizaCredito(id\_clte, monto);

93.

94. return blnAutorizado;

95. }

96.

97. }

### Código WS\_Envios

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package wsenvios;

7.

8. import entidades.Envios;

9. import fachadas.EnviosFacade;

10. import java.util.List;

11. import java.util.Set;

12. import java.util.logging.Level;

13. import java.util.logging.Logger;

14. import javax.annotation.Resource;

15. import javax.ejb.EJB;

16. import javax.jms.Connection;

17. import javax.jms.ConnectionFactory;

18. import javax.jms.MessageProducer;

19. import javax.jms.ObjectMessage;

20. import javax.jms.Queue;

21. import javax.jms.Session;

22. import javax.jws.Oneway;

23. import javax.jws.WebMethod;

24. import javax.jws.WebParam;

25. import javax.jws.WebService;

26.

27. /\*\*

28. \*

29. \* @author RGAMBOAH

30. \*/

31. @WebService(serviceName = "WS\_Envios")

32. public class WS\_Envios

33. {

34. @Resource(mappedName="jms/SolicitudEnviosFactory")

35. private ConnectionFactory connectionFactory;

36.

37. @Resource(mappedName="java:app/jms/Envios")

38. private Queue queue;

39.

40. @EJB

41. private EnviosFacade ejbRef;// Add business logic below. (Right-click in editor and choose

42. // "Insert Code > Add Web Service Operation")

43.

44. @WebMethod(operationName = "create")

45. @Oneway

46. public void create(@WebParam(name = "entity") Envios entity) {

47. ejbRef.create(entity);

48. }

49.

50. @WebMethod(operationName = "edit")

51. @Oneway

52. public void edit(@WebParam(name = "entity") Envios entity) {

53. ejbRef.edit(entity);

54. }

55.

56. @WebMethod(operationName = "remove")

57. @Oneway

58. public void remove(@WebParam(name = "entity") Envios entity) {

59. ejbRef.remove(entity);

60. }

61.

62. @WebMethod(operationName = "find")

63. public Envios find(@WebParam(name = "id") Object id) {

64. return ejbRef.find(id);

65. }

66.

67. @WebMethod(operationName = "findAll")

68. public List<Envios> findAll() {

69. return ejbRef.findAll();

70. }

71.

72. @WebMethod(operationName = "findRange")

73. public List<Envios> findRange(@WebParam(name = "range") int[] range) {

74. return ejbRef.findRange(range);

75. }

76.

77. @WebMethod(operationName = "count")

78. public int count() {

79. return ejbRef.count();

80. }

81.

82. /\*\*

83. \* Web service operation

84. \* @param id\_Tda

85. \* @param id\_pedido

86. \* @param name

87. \* @param email

88. \* @param phone

89. \* @param address

90. \* @param city\_region

91. \* @return

92. \*/

93. @WebMethod(operationName = "solicitudEnvio")

94. public boolean solicitudEnvio(@WebParam(name = "id\_Tda") int id\_Tda,

95. @WebParam(name = "id\_pedido") int id\_pedido,

96. @WebParam(name = "name") String name,

97. @WebParam(name = "email") String email,

98. @WebParam(name = "phone") String phone,

99. @WebParam(name = "address") String address,

100. @WebParam(name = "city\_region") String city\_region)

101. {

102. Logger.getLogger("Entro al metodo");

103. boolean blnRes = false;

104.

105. entidades.Envios envio = new entidades.Envios();

106.

107. envio.setIdTda(id\_Tda);

108. envio.setIdCustomerOrder(id\_pedido);

109. envio.setName(name);

110. envio.setEmail(email);

111. envio.setPhone(phone);

112. envio.setAddress(address);

113. envio.setCityRegion(city\_region);

114.

115. try

116. {

117. Connection connection = connectionFactory.createConnection();

118. Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

119. MessageProducer messageProducer = session.createProducer(queue);

120.

121. ObjectMessage message = session.createObjectMessage();

122.

123. message.setObject(envio);

124. messageProducer.send(message);

125.

126. Logger.getLogger(this.getClass().getName()).log(Level.INFO,

127. "Encolando solicitud de envio para:" + envio.getName() + ", para el pedido:" + envio.getIdCustomerOrder() + " co email:" + envio.getEmail());

128.

129. messageProducer.close();

130. connection.close();

131.

132. blnRes = true;

133. }

134. catch (Exception ex)

135. {

136. Logger.getLogger("Try Catch");

137. blnRes = false;

138. }

139.

140. return blnRes;

141. }

142.

143.

144.

145.

146. }

147.

148.

## Pojos para pruebas funcionales de los tres servicios, el integrado y servicios adicionales.

### Pojo Pedidos

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package pojo\_pedidos\_libros;

7. /\*\*

8. \*

9. \* @author rafael

10. \*/

11. public class Pojo\_Pedidos\_Libros //implements solicita\_servicio.ISolicitaServicio

12. {

13. long quienSoy;

14. String host = null;

15.

16. java.util.List<wspedidos.Customer> listaCltes = new java.util.ArrayList<>();

17. java.util.List<wspedidos.Product> listaProds = new java.util.ArrayList<>();

18.

19. int num\_cltes;

20. int num\_prods;

21.

22. //@Override

23. public void prepara(long quienSoy, String host)

24. {

25. this.quienSoy = quienSoy;

26. this.host = host; // NOTA: no se utiliza

27.

28. listaCltes = catalogoCltes();

29. listaProds = catalogoProds();

30. num\_cltes = listaCltes.size();

31. num\_prods = listaProds.size();

32. }

33.

34. //@Override

35. public void solicitaServicio(int vez)

36. {

37. java.util.List<wspedidos.ClsItem> listaIt = new java.util.ArrayList<>();

38.

39. int num\_pedido;

40. int id\_clte;

41. int num\_it;

42. int cantidad;

43. long id\_prod;

44.

45. int queClte,queProd = 0;

46.

47. wspedidos.ClsItem item;

48.

49. queClte = (int) ( num\_cltes \* Math.random());

50. id\_clte = listaCltes.get(queClte).getId();

51.

52. num\_it = (int) (1.0 + 10.0 \* Math.random());

53. num\_it = num\_it <= this.num\_prods ? num\_it : this.num\_prods;

54.

55. //

56. // para controlar que no se repitan los id\_prod de los items

57. //

58. int [] arr\_id\_prod = new int[num\_it];

59. boolean ya\_esta;

60. //

61. // Se generan los items para el pedido

62. //

63. for( int k = 0; k < num\_it; k++)

64. {

65. ya\_esta = true;

66. while(ya\_esta)

67. {

68. queProd = (int) ( num\_prods \* Math.random());

69. if( k == 0)

70. ya\_esta = false;

71. else

72. {

73. ya\_esta = false;

74. for( int j = 0; j < k; j++)

75. ya\_esta = ya\_esta | queProd == arr\_id\_prod[j];

76. }

77. }

78. arr\_id\_prod[k] = queProd;

79.

80. id\_prod = listaProds.get(queProd).getId();

81. cantidad = (int)( 5.0 + 100 \* Math.random() );

82. item = new wspedidos.ClsItem();

83. item.setIdProd(id\_prod);

84. item.setCantidad(cantidad);

85. listaIt.add(item);

86. }

87. System.out.println("-----------------------------------------------");

88. System.out.println("Estresador:" + this.quienSoy + ", vez:" + vez + ", Clte:" + id\_clte);

89. System.out.println("-----------------------------------------------");

90. boolean A;

91.

92. for(wspedidos.ClsItem it : listaIt)

93. System.out.println("Prod\_id:" + it.getIdProd() + ", cantidad:" + it.getCantidad() );

94. System.out.println("-----------------------------------------------");

95. //

96. // Se solicita registrar el pedido en el WS

97. //

98. num\_pedido = altaPedido(id\_clte,listaIt);

99.

100.

101. System.out.println("El número de pedido es:" + num\_pedido);

102. System.out.println("===============================================");

103.

104. restitucionPedido(num\_pedido);

105. System.out.println("Se restituyo la orden");

106. }

107.

108. //@Override

109. public void cierra()

110. {

111. System.out.println("El thread de stress " + this.quienSoy + " ha terminado su trabajo");

112. }

113.

114. // =========================================================================

115. // main para probar el pojo

116. // =========================================================================

117.

118. /\*\*

119. \* @param args the command line arguments

120. \*/

121. public static void main(String[] args)

122. {

123. Pojo\_Pedidos\_Libros objServ = new Pojo\_Pedidos\_Libros();

124.

125. objServ.prepara(25, null);

126. int n\_veces = args.length > 0 ? Integer.parseInt(args[0]):5;

127. for( int vez = 1; vez <= n\_veces; vez++)

128. objServ.solicitaServicio(vez);

129. objServ.cierra();

130.

131. }

132. // =========================================================================

133. // Utilerías del WS

134. // =========================================================================

135. private static int altaPedido(int idClte, java.util.List<wspedidos.ClsItem> listaIt) {

136. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

137. wspedidos.WSPedido port = service.getWSPedidoPort();

138. return port.altaPedido(idClte, listaIt);

139. }

140.

141. private static java.util.List<wspedidos.Customer> catalogoCltes() {

142. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

143. wspedidos.WSPedido port = service.getWSPedidoPort();

144. return port.catalogoCltes();

145. }

146.

147. private static java.util.List<wspedidos.Product> catalogoProds() {

148. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

149. wspedidos.WSPedido port = service.getWSPedidoPort();

150. return port.catalogoProds();

151. }

152.

153.

154.

155. // =========================================================================

156. // Fin de las Utilerías del WS

157. // =========================================================================

158.

159. private static void restitucionPedido(int idPedido) {

160. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

161. wspedidos.WSPedido port = service.getWSPedidoPort();

162. port.restitucionPedido(idPedido);

163. }

164.

165.

166.

167.

168. }

### Pojo Crédito

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package pojo\_credito\_libros;

7.

8. import ws\_credito\_libros.ExcepNoCredito\_Exception;

9. import ws\_credito\_libros.ExcepNoExisteClte\_Exception;

10.

11. /\*\*

12. \*

13. \* @author eduar

14. \*/

15. public class Pojo\_Credito\_Libros {

16.

17. /\*\*

18. \* @param args the command line arguments

19. \*/

20.

21. public static void main(String[] args) {

22. int VECES = args.length > 0 ? Integer.parseInt(args[0]) : 10;

23. int id\_clte;

24. double dblMonto = 0.0;

25. for(int vez = 1; vez <= VECES; vez++)

26. {

27. dblMonto = 100.0 \* Math.random();

28. id\_clte = (int)(10.0 \* Math.random());

29.

30. try

31. {

32. if( autorizaDouble(id\_clte,dblMonto))

33. {

34. System.out.println("Se ha autorizado el credito de " + dblMonto + " para el id\_clte " +id\_clte);

35. }

36. else

37. {

38. System.out.println("No se ha autorizado el credito de " + dblMonto + " para el id\_clte " +id\_clte);

39. }

40. }

41. catch( Exception ex)

42. {

43. System.out.println(ex.toString());

44. }

45.

46. }

47. }

48.

49. private static boolean autorizaDouble(int idClte, double dblMonto) throws ExcepNoCredito\_Exception, ExcepNoExisteClte\_Exception {

50. ws\_credito\_libros.WSCredito service = new ws\_credito\_libros.WSCredito();

51. ws\_credito\_libros.WSCreditoLibros port = service.getWSCreditoLibrosPort();

52. return port.autorizaDouble(idClte, dblMonto);

53. }

54.

55.

56.

57.

58. }

### Pojo Envíos

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package pojo\_envios\_libros;

7.

8. /\*\*

9. \*

10. \* @author eduar

11. \*/

12. public class Pojo\_Envios\_Libros {

13.

14. public static void main(String[] args)

15. {

16. Clte.carga\_lista\_Cltes();

17. Clte clte = null;

18.

19. boolean blnSolEnv = false;

20.

21. for( int k = 300;k <= 305; k++)

22. {

23. clte = Clte.selClte();

24. clte.setId\_pedido(k);

25.

26. blnSolEnv = solicitudEnvio(clte.getId\_tda(),

27. clte.getId\_pedido(),

28. clte.getName(),

29. clte.getEmail(),

30. clte.getPhone(),

31. clte.getAddress(),

32. clte.getCity\_region());

33. System.out.println("Se ha solicitado el envio del pedido " + clte.getId\_pedido() + " para el clte " + clte.getName() +

34. " con email " + clte.getEmail() + "con status " + blnSolEnv);

35. }

36. }

37.

38. private static boolean solicitudEnvio(int idTda, int idPedido, java.lang.String name, java.lang.String email, java.lang.String phone, java.lang.String address, java.lang.String cityRegion) {

39. wsenvios.WSEnvios\_Service service = new wsenvios.WSEnvios\_Service();

40. wsenvios.WSEnvios port = service.getWSEnviosPort();

41. return port.solicitudEnvio(idTda, idPedido, name, email, phone, address, cityRegion);

42. }

43.

44.

45.

46. }

### BPEL\_Proyecto

1. <?xml version="1.0" encoding="UTF-8" ?>

2. <process

3. name="BPEL\_Proyecto"

4. targetNamespace="http://enterprise.netbeans.org/bpel/BPEL\_Proyecto/BPEL\_Proyecto"

5. xmlns:tns="http://enterprise.netbeans.org/bpel/BPEL\_Proyecto/BPEL\_Proyecto"

6. xmlns:xs="http://www.w3.org/2001/XMLSchema"

7. xmlns:xsd="http://www.w3.org/2001/XMLSchema"

8. xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"

9. xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"

10. xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor2"

11. xmlns:sxat="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Attachment"

12. xmlns:sxeh="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/ErrorHandling" xmlns:ns0="http://xml.netbeans.org/schema/Schema\_Proyecto">

13. <import namespace="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" location="WS\_Solicita.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>

14. <import namespace="http://enterprise.netbeans.org/bpel/WSPedidoWrapper" location="WSPedidoWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>

15. <import namespace="http://wspedido/" location="http://localhost:8080/WS\_Pedidos\_Libros/WSPedido?WSDL" importType="http://schemas.xmlsoap.org/wsdl/"/>

16. <import namespace="http://enterprise.netbeans.org/bpel/WSCreditoWrapper" location="WSCreditoWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>

17. <import namespace="http://ws\_credito\_libros/" location="http://localhost:8080/WS\_Credito\_Libros/WSCredito?WSDL" importType="http://schemas.xmlsoap.org/wsdl/"/>

18. <import namespace="http://enterprise.netbeans.org/bpel/WS\_EnviosWrapper" location="WS\_EnviosWrapper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>

19. <import namespace="http://wsenvios/" location="http://localhost:8080/WS\_Envios\_Libros/WS\_Envios?WSDL" importType="http://schemas.xmlsoap.org/wsdl/"/>

20. <partnerLinks>

21. <partnerLink name="PLPedidos" xmlns:tns="http://enterprise.netbeans.org/bpel/WSPedidoWrapper" partnerLinkType="tns:WSPedidoLinkType" partnerRole="WSPedidoRole"/>

22. <partnerLink name="PLCredito" xmlns:tns="http://enterprise.netbeans.org/bpel/WSCreditoWrapper" partnerLinkType="tns:WS\_Credito\_LibrosLinkType" partnerRole="WS\_Credito\_LibrosRole"/>

23. <partnerLink name="PLEnvios" xmlns:tns="http://enterprise.netbeans.org/bpel/WS\_EnviosWrapper" partnerLinkType="tns:WS\_EnviosLinkType" partnerRole="WS\_EnviosRole"/>

24. <partnerLink name="PLSolicita" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" partnerLinkType="tns:WS\_Solicita" myRole="WS\_SolicitaPortTypeRole"/>

25. </partnerLinks>

26. <variables>

27. <variable name="SolicitudEnvioOut" xmlns:tns="http://wsenvios/" messageType="tns:solicitudEnvioResponse"/>

28. <variable name="SolicitudEnvioIn" xmlns:tns="http://wsenvios/" messageType="tns:solicitudEnvio"/>

29. <variable name="ClteDelPedidoOut" xmlns:tns="http://wspedido/" messageType="tns:clteDelPedidoResponse"/>

30. <variable name="ClteDelPedidoIn" xmlns:tns="http://wspedido/" messageType="tns:clteDelPedido"/>

31. <variable name="RestitucionPedidoOut" xmlns:tns="http://wspedido/" messageType="tns:restitucionPedidoResponse"/>

32. <variable name="RestitucionPedidoIn" xmlns:tns="http://wspedido/" messageType="tns:restitucionPedido"/>

33. <variable name="AutorizaOut" xmlns:tns="http://ws\_credito\_libros/" messageType="tns:autorizaResponse"/>

34. <variable name="AutorizaIn" xmlns:tns="http://ws\_credito\_libros/" messageType="tns:autoriza"/>

35. <variable name="MontoPedidoOut" xmlns:tns="http://wspedido/" messageType="tns:montoPedidoResponse"/>

36. <variable name="MontoPedidoIn" xmlns:tns="http://wspedido/" messageType="tns:montoPedido"/>

37. <variable name="Fault1FaultVar" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" messageType="tns:WS\_SolicitaOperationFault"/>

38. <variable name="AltaPedidoOut" xmlns:tns="http://wspedido/" messageType="tns:altaPedidoResponse"/>

39. <variable name="AltaPedidoIn" xmlns:tns="http://wspedido/" messageType="tns:altaPedido">

40. <sxed:editor>

41. <sxed:predicate path="$AltaPedidoIn.parameters/lista\_it[$ForEach1Counter]" source="to"/>

42. </sxed:editor>

43. </variable>

44. <variable name="WS\_SolicitaOperationOut" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" messageType="tns:WS\_SolicitaOperationResponse"/>

45. <variable name="WS\_SolicitaOperationIn" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" messageType="tns:WS\_SolicitaOperationRequest">

46. <sxed:editor>

47. <sxed:predicate path="$WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items[$ForEach1Counter]" source="from"/>

48. </sxed:editor>

49. </variable>

50. </variables>

51. <sequence>

52. <receive name="Receive1" createInstance="yes" partnerLink="PLSolicita" operation="WS\_SolicitaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" portType="tns:WS\_SolicitaPortType" variable="WS\_SolicitaOperationIn"/>

53. <assign name="Assign1">

54. <copy>

55. <from>$WS\_SolicitaOperationIn.datoEntrada/ns0:id\_clte</from>

56. <to>$AltaPedidoIn.parameters/id\_clte</to>

57. </copy>

58. </assign>

59. <forEach name="ForEach1" parallel="no" counterName="ForEach1Counter">

60. <startCounterValue>1</startCounterValue>

61. <finalCounterValue>count($WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items/ns0:id\_prod)</finalCounterValue>

62. <scope name="Scope1">

63. <assign name="Assign2">

64. <copy>

65. <from>$WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items[$ForEach1Counter]/ns0:cantidad

66. <sxed:editor>

67. <sxed:predicate path="$WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items[$ForEach1Counter]" source="from"/>

68. </sxed:editor>

69. </from>

70. <to>$AltaPedidoIn.parameters/lista\_it[$ForEach1Counter]/cantidad

71. <sxed:editor>

72. <sxed:predicate path="$AltaPedidoIn.parameters/lista\_it[$ForEach1Counter]" source="to"/>

73. </sxed:editor>

74. </to>

75. </copy>

76. <copy>

77. <from>$WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items[$ForEach1Counter]/ns0:id\_prod

78. <sxed:editor>

79. <sxed:predicate path="$WS\_SolicitaOperationIn.datoEntrada/ns0:lista\_items[$ForEach1Counter]" source="from"/>

80. </sxed:editor>

81. </from>

82. <to>$AltaPedidoIn.parameters/lista\_it[$ForEach1Counter]/id\_prod

83. <sxed:editor>

84. <sxed:predicate path="$AltaPedidoIn.parameters/lista\_it[$ForEach1Counter]" source="to"/>

85. </sxed:editor>

86. </to>

87. </copy>

88. </assign>

89. </scope>

90. </forEach>

91. <invoke name="Invoke1" partnerLink="PLPedidos" operation="altaPedido" xmlns:tns="http://wspedido/" portType="tns:WSPedido" inputVariable="AltaPedidoIn" outputVariable="AltaPedidoOut"/>

92. <if name="If1">

93. <condition>$AltaPedidoOut.parameters/return = 0</condition>

94. <sequence name="Sequence1">

95. <assign name="Assign3">

96. <copy>

97. <from>'No hay suficientes existencias'</from>

98. <to>$Fault1FaultVar.datoFault/ns0:message</to>

99. </copy>

100. </assign>

101. <reply name="Reply2" partnerLink="PLSolicita" operation="WS\_SolicitaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" portType="tns:WS\_SolicitaPortType" faultName="tns:fault1" variable="Fault1FaultVar"/>

102. <exit name="Exit1"/>

103. </sequence>

104. </if>

105. <assign name="Assign4">

106. <copy>

107. <from>$AltaPedidoOut.parameters/return</from>

108. <to>$MontoPedidoIn.parameters/num\_pedido</to>

109. </copy>

110. </assign>

111. <invoke name="Invoke2" partnerLink="PLPedidos" operation="montoPedido" xmlns:tns="http://wspedido/" portType="tns:WSPedido" inputVariable="MontoPedidoIn" outputVariable="MontoPedidoOut"/>

112. <assign name="Assign5">

113. <copy>

114. <from>$WS\_SolicitaOperationIn.datoEntrada/ns0:id\_clte</from>

115. <to>$AutorizaIn.parameters/id\_clte</to>

116. </copy>

117. <copy>

118. <from>$MontoPedidoOut.parameters/return</from>

119. <to>$AutorizaIn.parameters/monto</to>

120. </copy>

121. </assign>

122. <scope name="Scope2">

123. <faultHandlers>

124. <catchAll>

125. <sequence name="Sequence2">

126. <assign name="Assign6">

127. <copy>

128. <from>'No se autorizo el pedido'</from>

129. <to>$Fault1FaultVar.datoFault/ns0:message</to>

130. </copy>

131. <copy>

132. <from>$AltaPedidoOut.parameters/return</from>

133. <to>$RestitucionPedidoIn.parameters/idPedido</to>

134. </copy>

135. </assign>

136. <invoke name="Invoke4" partnerLink="PLPedidos" operation="restitucionPedido" xmlns:tns="http://wspedido/" portType="tns:WSPedido" inputVariable="RestitucionPedidoIn" outputVariable="RestitucionPedidoOut"/>

137. <reply name="Reply3" partnerLink="PLSolicita" operation="WS\_SolicitaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" portType="tns:WS\_SolicitaPortType" faultName="tns:fault1" variable="Fault1FaultVar"/>

138. </sequence>

139. </catchAll>

140. </faultHandlers>

141. <invoke name="Invoke3" partnerLink="PLCredito" operation="autoriza" xmlns:tns="http://ws\_credito\_libros/" portType="tns:WS\_Credito\_Libros" inputVariable="AutorizaIn" outputVariable="AutorizaOut"/>

142. </scope>

143. <assign name="Assign7">

144. <copy>

145. <from>$AltaPedidoOut.parameters/return</from>

146. <to>$ClteDelPedidoIn.parameters/num\_pedido</to>

147. </copy>

148. </assign>

149. <invoke name="Invoke5" partnerLink="PLPedidos" operation="clteDelPedido" xmlns:tns="http://wspedido/" portType="tns:WSPedido" inputVariable="ClteDelPedidoIn" outputVariable="ClteDelPedidoOut"/>

150. <assign name="Assign8">

151. <copy>

152. <from>$ClteDelPedidoOut.parameters/return/name</from>

153. <to>$SolicitudEnvioIn.parameters/name</to>

154. </copy>

155. <copy>

156. <from>$ClteDelPedidoOut.parameters/return/email</from>

157. <to>$SolicitudEnvioIn.parameters/email</to>

158. </copy>

159. <copy>

160. <from>$ClteDelPedidoOut.parameters/return/phone</from>

161. <to>$SolicitudEnvioIn.parameters/phone</to>

162. </copy>

163. <copy>

164. <from>$ClteDelPedidoOut.parameters/return/address</from>

165. <to>$SolicitudEnvioIn.parameters/address</to>

166. </copy>

167. <copy>

168. <from>$ClteDelPedidoOut.parameters/return/cityRegion</from>

169. <to>$SolicitudEnvioIn.parameters/city\_region</to>

170. </copy>

171. <copy>

172. <from>$AltaPedidoOut.parameters/return</from>

173. <to>$SolicitudEnvioIn.parameters/id\_pedido</to>

174. </copy>

175. <copy>

176. <from>1</from>

177. <to>$SolicitudEnvioIn.parameters/id\_Tda</to>

178. </copy>

179. </assign>

180. <invoke name="Invoke6" partnerLink="PLEnvios" operation="solicitudEnvio" xmlns:tns="http://wsenvios/" portType="tns:WS\_Envios" inputVariable="SolicitudEnvioIn" outputVariable="SolicitudEnvioOut"/>

181. <if name="If2">

182. <condition>$SolicitudEnvioOut.parameters/return and true()</condition>

183. <assign name="Assign9">

184. <copy>

185. <from>$AltaPedidoOut.parameters/return</from>

186. <to>$WS\_SolicitaOperationOut.datoSalida/ns0:respuesta</to>

187. </copy>

188. </assign>

189. <else>

190. <sequence name="Sequence3">

191. <assign name="Assign10">

192. <copy>

193. <from>'Hubo un error en el envio'</from>

194. <to>$Fault1FaultVar.datoFault/ns0:message</to>

195. </copy>

196. </assign>

197. <reply name="Reply4" partnerLink="PLSolicita" operation="WS\_SolicitaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" portType="tns:WS\_SolicitaPortType" faultName="tns:fault1" variable="Fault1FaultVar"/>

198. <exit name="Exit2"/>

199. </sequence>

200. </else>

201. </if>

202. <reply name="Reply1" partnerLink="PLSolicita" operation="WS\_SolicitaOperation" xmlns:tns="http://j2ee.netbeans.org/wsdl/BPEL\_Proyecto/src/WS\_Solicita" portType="tns:WS\_SolicitaPortType" variable="WS\_SolicitaOperationOut"/>

203. </sequence>

204. </process>

205.

### Pojo\_Proyecto

1. /\*

2. \* To change this license header, choose License Headers in Project Properties.

3. \* To change this template file, choose Tools | Templates

4. \* and open the template in the editor.

5. \*/

6. package pojo\_proyecto;

7.

8. import java.math.BigInteger;

9. import solicita\_servicio.ISolicitaServicio;

10. import wsproy.RespuestaGestionaPedido;

11. import wsproy.WSSolicitaOperationFault;

12.

13. /\*\*

14. \*

15. \* @author eduar

16. \*/

17. public class Pojo\_Proyecto implements ISolicitaServicio {

18.

19. long quienSoy;

20. String host = null;

21.

22. java.util.List<wspedidos.Customer> listaCltes = new java.util.ArrayList<>();

23. java.util.List<wspedidos.Product> listaProds = new java.util.ArrayList<>();

24.

25. int num\_cltes;

26. int num\_prods;

27.

28. @Override

29. public void prepara(long quienSoy, String host)

30. {

31. this.quienSoy = quienSoy;

32. this.host = host; // NOTA: no se utiliza

33.

34. listaCltes = catalogoCltes();

35. listaProds = catalogoProds();

36. num\_cltes = listaCltes.size();

37. num\_prods = listaProds.size();

38. }

39.

40. @Override

41. public long solicitaServicio(int vez)

42. {

43. java.util.List<wspedidos.ClsItem> listaIt = new java.util.ArrayList<>();

44.

45. int num\_pedido;

46. int id\_clte;

47. int num\_it;

48. int cantidad;

49. long id\_prod;

50.

51. int queClte,queProd = 0;

52.

53. wspedidos.ClsItem item;

54.

55. queClte = (int) ( num\_cltes \* Math.random());

56. id\_clte = listaCltes.get(queClte).getId();

57.

58.

59. num\_it = (int) (1.0 + 10.0 \* Math.random());

60. num\_it = num\_it <= this.num\_prods ? num\_it : this.num\_prods;

61.

62. //

63. // para controlar que no se repitan los id\_prod de los items

64. //

65. int [] arr\_id\_prod = new int[num\_it];

66. boolean ya\_esta;

67. //

68. // Se generan los items para el pedido

69. //

70. for( int k = 0; k < num\_it; k++)

71. {

72. ya\_esta = true;

73. while(ya\_esta)

74. {

75. queProd = (int) ( num\_prods \* Math.random());

76. if( k == 0)

77. ya\_esta = false;

78. else

79. {

80. ya\_esta = false;

81. for( int j = 0; j < k; j++)

82. ya\_esta = ya\_esta | queProd == arr\_id\_prod[j];

83. }

84. }

85. arr\_id\_prod[k] = queProd;

86.

87. id\_prod = listaProds.get(queProd).getId();

88. cantidad = (int)( 5.0 + 100 \* Math.random() );

89. item = new wspedidos.ClsItem();

90. item.setIdProd(id\_prod);

91. item.setCantidad(cantidad);

92. listaIt.add(item);

93. }

94. System.out.println("-----------------------------------------------");

95. System.out.println("Estresador:" + this.quienSoy + ", vez:" + vez + ", Clte:" + id\_clte);

96. System.out.println("-----------------------------------------------");

97. boolean A;

98.

99. wsproy.GestionaPedido entrada = new wsproy.GestionaPedido ();

100. for(wspedidos.ClsItem it : listaIt){

101. wsproy.ClsItem itProy = new wsproy.ClsItem();

102. itProy.setCantidad(it.getCantidad());

103. itProy.setIdProd(BigInteger.valueOf( it.getIdProd()));

104. entrada.getListaItems().add(itProy);

105. System.out.println("Prod\_id:" + it.getIdProd() + ", cantidad:" + it.getCantidad() );

106. }

107. System.out.println("-----------------------------------------------");

108.

109.

110. wsproy.RespuestaGestionaPedido salida;

111. entrada.setIdClte(id\_clte);

112.

113. wsproy.ExcepcionGenerica ex;

114.

115. long d2=System.nanoTime();

116. long d1=System.nanoTime();

117. try{

118. salida=wsSolicitaOperation(entrada);

119. System.out.println("Num\_Pedido: "+salida.getRespuesta());

120. }catch( WSSolicitaOperationFault e){

121.

122. ex=e.getFaultInfo();

123. System.out.println(ex.getMessage());

124. }

125. //

126. // Se solicita registrar el pedido en el WS

127. //

128. //num\_pedido = altaPedido(id\_clte,listaIt);

129.

130.

131.

132.

133. return d2-d1;

134. }

135.

136. @Override

137. public void cierra()

138. {

139. System.out.println("El thread de stress " + this.quienSoy + " ha terminado su trabajo");

140. }

141.

142. // =========================================================================

143. // main para probar el pojo

144. // =========================================================================

145.

146. /\*\*

147. \* @param args the command line arguments

148. \*/

149. public static void main(String[] args)

150. {

151. Pojo\_Proyecto objServ = new Pojo\_Proyecto();

152.

153. objServ.prepara(25, null);

154. int n\_veces = args.length > 0 ? Integer.parseInt(args[0]):5;

155. for( int vez = 1; vez <= n\_veces; vez++)

156. objServ.solicitaServicio(vez);

157. objServ.cierra();

158.

159. }

160.

161. private static java.util.List<wspedidos.Customer> catalogoCltes() {

162. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

163. wspedidos.WSPedido port = service.getWSPedidoPort();

164. return port.catalogoCltes();

165. }

166.

167. private static java.util.List<wspedidos.Product> catalogoProds() {

168. wspedidos.WSPedido\_Service service = new wspedidos.WSPedido\_Service();

169. wspedidos.WSPedido port = service.getWSPedidoPort();

170. return port.catalogoProds();

171. }

172.

173. private static RespuestaGestionaPedido wsSolicitaOperation(wsproy.GestionaPedido datoEntrada) throws WSSolicitaOperationFault {

174. wsproy.CAProyectoService1 service = new wsproy.CAProyectoService1();

175. wsproy.WSSolicitaPortType port = service.getCasaPort1();

176. return port.wsSolicitaOperation(datoEntrada);

177. }

178.

181. }

182.