INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO



SDI 24810 - Sistemas de Comercio Electrónico

Documentación del proyecto final de la materia

Pollo Con Frijoles

Alumnos:

José Francisco Altamirano Zevallos 163314

Sebastián Valderrábano Cabrera 163791

Eduardo Daniel Urrutia Ruiz 165395

América Castrejón Porres 166414

Fernando Merino Benítez 166541

Rodrigo Castillo Gómez 168114

16 de diciembre de 2020

# Resumen

El objetivo de este proyecto es realizar un proceso de negocio de venta online de libros. La arquitectura y requerimientos (tanto funcionales como no funcionales) se derivaron de las necesidades que una plataforma de esta naturaleza tendría. Para desarrollar el ejemplo se realizaron dos versiones: la versión loopback y la versión con la base de datos. Se realizó en estas dos etapas para asegurar la escalabilidad de la plataforma. Adicionalmente, se realizaron procesos de estrés de ambas versiones para asegurar la transaccionalidad esperada. Ambas versiones se realizaron en OpenESB 2.1 y es necesario contar con NetBeans y Glassfish. En las siguientes páginas se encuentra la documentación a la solución propuesta.

En caso de querer consultar la versión desplegada, integrada y completa acceder a: <https://github.com/FAltamiranoZ/SistemasDeComercio>

Tabla de contenido

[**Resumen**](#_rg3zjyq9tcrh) **2**

[**Control de proyecto.**](#_v718msxrgufk) **4**

[Plan de trabajo](#_tapscgeg8end) 4

[**Requerimientos De negocio**](#_xauxi14irhru) **5**

[Requerimientos Funcionales](#_bj3by55uzafo) 5

[Requerimientos No Funcionales](#_9tnnzg45cmjw) 5

[**Historias de Usuario**](#_8erccbqhupcn) **6**

[Happy Paths](#_rybkgoamu6ts) 7

[Rutas alternas](#_9tnnzg45cmjw) 8

[**Documento de Arquitectura**](#_5i5ayyz29a6j) **10**

[Descripción general de la arquitectura propuesta](#_9tnnzg45cmjw) 10

[**Responsabilidades de los Componentes de la Arquitectura y relación con las reglas de negocio.**](#_pg6oh43rtxrr) **12**

[**Control de avance y desviaciones**](#_8ulzjiv5y8wo) **13**

[Versión Loopback](#_bjfiwmyz8l9t) 13

[Estress](#_bhrby7mhus00) 13

[Versión con base de datos](#_ddukwasxwdt6) 13

[Estress](#_uxqaauspq093) 13

[**Documentación de pruebas funcionales**](#_9c8z5s821koc) **15**

[**Documentación de pruebas no funcionales**](#_bzoflgtwbk10) **18**

[**Arquitectura y versiones**](#_nzbq7nmbvso0) **21**

[Versiónes Loopback](#_pd752afoi8fc) 21

[Versiónes BD](#_9489c82badxd) 22

[**Web Service de Almacén**](#_9a85q9jlmff4) **23**

[**Web Service de Pago**](#_pzv2p6jryr8x) **26**

[**Web Service de Entregas**](#_sktwdf9dywth) **29**

[**Estress general de cada servicio y a la aplicación**](#_e8rp1pijgo02) **31**

[Pojo de Servicio Integrado](#_rhxducqfqgpr) 33

[**Anexos:**](#_o339prorcooc) **38**

[Scripts de inicialización de BD de los servicios básicos.](#_9tnnzg45cmjw) 38

[Código de web services y acceso a BD.](#_9tnnzg45cmjw) 40

[Pojos para pruebas funcionales de tres servicios, el integrado y servicios adicionales.](#_9tnnzg45cmjw) 47

# **Control de proyecto.**

## Plan de trabajo

# 



# 

# **Requerimientos De negocio**

## 

## 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 un solo ISBN.

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

**RN5.-** No debe permitir la venta si no se haya 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.-** Se debe de permitir dar de alta nuevos clientes y nuevos libros

**RN8.-** Al realizar una compra se debe modificar el saldo de los clientes y modificar la existencia de los libros.

## Requerimientos No Funcionales

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

**RNF2.-** El tiempo de respuesta para cada funcionalidad y transacción debe ser menor a 5 segundos

**RNF3.-** Mostrar mensaje de error en caso de que no se pueda completar la compra

**RNF4.-** Operar adecuadamente con 10 usuarios con sesiones concurrentes

**RNF5.-** Los cambios de la base de datos deben de ser actualizados para todos los usuarios

**RNF6.-** Únicamente los clientes dados de alta pueden realizar compras.

# 

# **Historias de Usuario**

A partir de las características del sistema establecemos un usuario, el comprador, que realizará las acciones y consultas con un portal web.Con esto en mente al desarrollar nos centramos en las siguientes historias de usuario:

*Comprador:*

* “Como comprador quiero poder revisar un libro por ISBN para poderlo comprar”
* “Como comprador quiero comprar un libro para aumentar mis conocimientos”
* “Como comprador quiero tener varios métodos de pago para elegir el más conveniente en un momento”
* “Como comprador quiero poder revisar mis fondos para poder asegurarme que puedo realizar la compra”
* “Como comprador quiero incrementar mis fondos para poder comprar más libros”
* “Como comprador quiero poder revisar el stock de los libros para poder completar los pedidos”

A pesar de que los compradores son los únicos usuarios finales, el sistema requiere de administradores que puedan gestionar el inventario y puedan realizar altas y bajas, etc. Para ello nos basamos en estas historias:

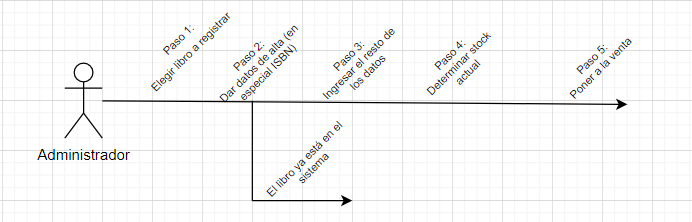
*Administrador:*

* “Como administrador quiero registrar nuevos libros a la venta para poder dar un mejor servicio”
* “Como administrador quiero poder aumentar el stock de libros para continuar con el servicio”
* “Como administrador quiero poder quitar libros descontinuados para sólo mostrar los que hay en existencia”
* “Como sistema quiero disminuir el stock una vez que se haya realizado una compra y envío para poder mantener al corriente el inventario”

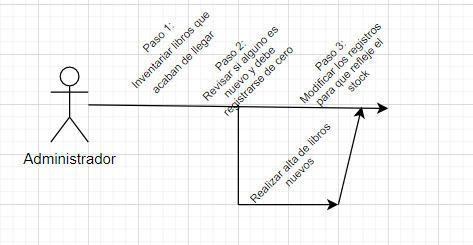
## Happy Paths

### 

Este es el path para poder realizar una compra. Empezando por el ingreso al portal y terminando en la entrega. El usuario inicia su camino buscando adquirir un producto, que en nuestro caso corresponde a un libro. Este va a realizar una búsqueda, lo cual implica que utilizando los ISBN se le indicará al usuario si el libro que desea adquirir se encuentra disponible. Si se encuentra en existencia, se le da al usuario el precio del producto. Con esto, el usuario puede decidir si proseguir con su compra y avanza a seleccionar el método de pago que va a utilizar. Teniendo estos datos, se verifica que el comprador tenga saldo disponible para completar la transacción y si esto se cumple, se finaliza la compra. Para finalizar su happy path se tiene el proceso de envío que termina con la entrega del producto.



Para el usuario administrador se requieren más paths, pues sus actividades son variadas. En el path de arriba podemos ver el path para dar de alta un nuevo libro. En el, el único punto de falla es que el libro ya se encuentre en el sistema (es decir, no sea nuevo).

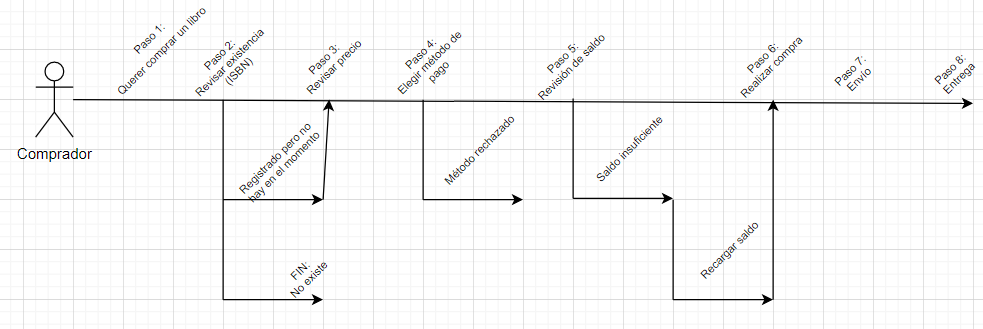


Aquí tenemos el otro path para aumentar el stock de los libros, es más sencillo que los otros y el único punto de falla puede ser que haya libros que requieran ser registrados desde 0 en el sistema.

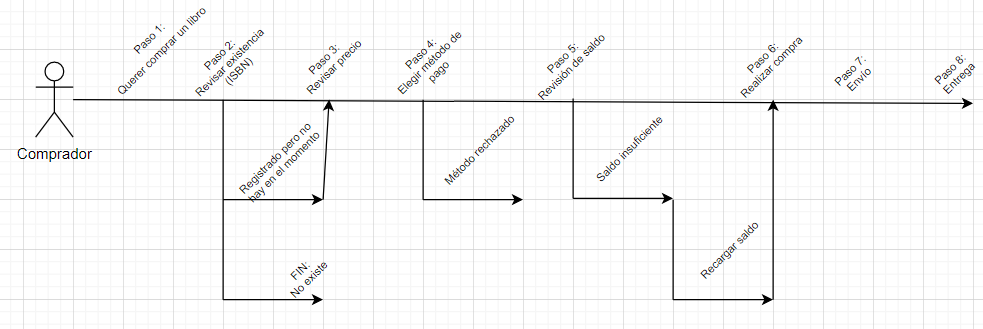
## Rutas alternas

A lo largo del path de comprador, podemos ver que hay puntos de posible falla. Estos serán los indicados a continuación:

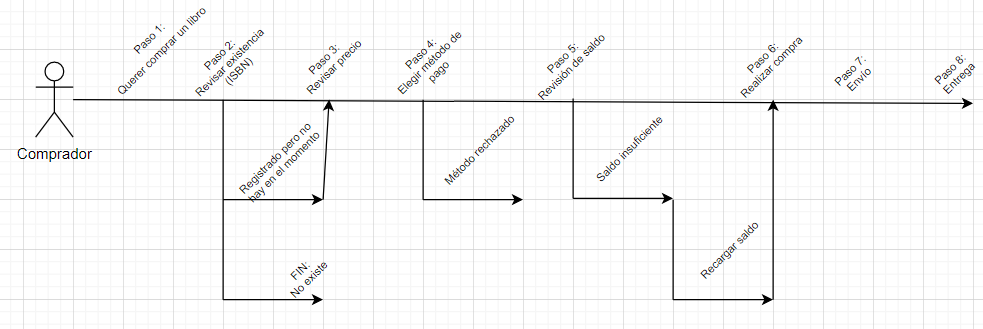
* Si el usuario busca un libro (ISBN) que no se encuentra en el sistema, en este caso el path termina.



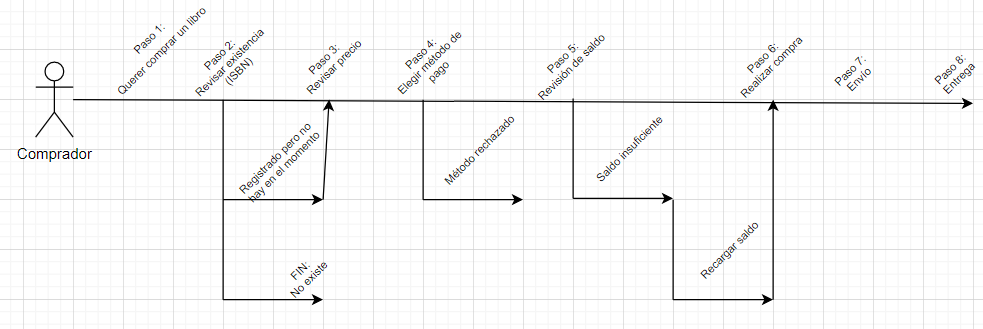
* Si el libro se encuentra en el sistema pero no hay ejemplares disponibles, se tiene una bifurcación pero se puede regresar al path.



* Al elegir un método de pago puede ser rechazado y el usuario puede ingresar otros datos que sean válidos o terminar el path.



* Al tratar de realizar el pago, puede tener saldo insuficiente que requiera ser recargado o el usuario puede desistir.



# 

# 

# 

# 

# 

# **Documento de Arquitectura**

## 

## Descripción general de la arquitectura propuesta

Para el correcto funcionamiento de la tienda online es necesario crear una base de datos que contenga la información de los productos y de los clientes; para poblar la base de datos se crea un script de sql. Una vez que se cuenta con los datos; la arquitectura se puede dividir en tres diferentes partes: glassfish, OpenESb y el estresador.

En el caso de glassfish se tienen tres servicios web que realizan las funcionalidades de la tienda online:

* **Servicio web del almacén:** recibe el ISBN del libro y las unidades de compra para revisar si existe o no stock suficiente.
* **Servicio web del sistema de pago:** recibe el idCliente y la cantidad de libros para generar una factura
* **Servicio web de empresa de mensajería:**  recibe el pedido y responde con el tiempo en días estimado de reparto.

Para comprobar el correcto funcionamiento de los servicios web se crean pojos para cada uno de ellos. Posteriormente en OpenESB se arma el BPEL con su CA y se realiza deploy. En la última etapa se crea un pojo dentro de la estructura del estresador.

## 

### 

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

En la siguiente tabla podemos ver la relación que existe entre los distintos componentes de la arquitectura y las reglas de negocio de las que cada uno es responsable. Los requerimientos no funcionales se reparten por los distintos componentes a lo largo del proceso, mientras que los requerimientos funcionales varían dependiendo del componente.

|  |  |  |
| --- | --- | --- |
| Componente | Requerimientos Funcionales | Requerimientos no Funcionales |
| Estresador |  | RNF1,RNF2,RNF3,RNF4 |
| ESB | RN1,RN2,RN4,RN5 | RNF1,RNF2,RNF3,RNF4 |
| Pojo Almacén | RN1,RN2,RN3,RN4,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF6 |
| Pojo Pago | RN1,RN2,RN4,RN5,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF5,RNF6 |
| Pojo Mensajería | RN6 | RNF1,RNF2,RNF3,RNF4,RNF5 |
| WS Almacén | RN1,RN2,RN3,RN4,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF6 |
| WS Pago | RN1,RN4,RN5,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF5,RNF6 |
| WS Mensajería | RN6 | RNF1,RNF2,RNF3,RNF4,RNF5 |
| Base de Datos | RN7,RN8 | RNF5 |

# **Control de avance y desviaciones**

Como herramienta de apoyo se creó un [repositorio en github](https://github.com/FAltamiranoZ/SistemasDeComercio) para tener el control de avance y desviaciones. El ingeniero en jefe e integrador fue el responsable de realizar los commits a la plataforma.La solución consta de dos sistemas distintos: la versión con LoopBack y la implementación utilizando bases de datos.

## Versión Loopback

|  |  |
| --- | --- |
| Fecha | Descripción |
| 27-nov. | Se crean los proyectos Mensajeria Express, PagoOnline y TiendaElectronica conde se incluyen los webServices originales del proyecto |
| 28-nov. | Se añade el estresador de la versión Loopback |
| Se crea el documento de texto anotacitaciones.txt con información importante para la ejecución del programa |
| 29-nov. | Se agregan los estresadores para los web services de MensajeriaExpress, PagoOnline, TiendaElectronica y se modificó el del general |
| Se agregan los proyectos web PagoOnline, ServiciosVenta y TiendaElectronica y se actualizó el archivo de anotaciones |
| 30-nov. | Se añadieron los BPEL y CA del proyecto, además, se añadieron anotaciones |
| Se realiza un cambio en el funcionamiento de los webServices y se incluyen funcionalidades para que se reduzca stock y saldo |
| 1-dic. | Se modificó la manera en la que se realizaba la conexión en todos los web services para poder asegurar una conexión estable en todo momento; además, se agregó la funcionalidad de reducir el saldo y el inventario de productos al realizar una compra; finalmente se añadieron anotaciones. |
| 2-dic. | Se corrige la funcionalidad de comprobar saldos y unidades |
| Se agregan pruebas en el CA y se incluye la documentación necesaria en el código |
| 14-dic | Se realizan las pruebas de estress para cada uno de los componentes del sistema |

## Versión con base de datos

|  |  |
| --- | --- |
| Fecha | Descripción |
| 29-nov. | Se crea el archivo serviciosVentaCreation.sql, con el cual se puede crear la base de datos |
| Se actualiza el archivo de anotaciones, se corrigió el esquema de creación de la Base de Datos, se agrega el proyecto Web de TiendaElectronica y su POJO. |
| 30-nov. | Se cambia idCliente a tipo String para mantener la congruencia con la aplicación BPEL en la que se especifica idCliente como String. |
| Se corrige la BD y se declara idFactura como String para mantener el mismo tipo usado por el BPEL |
| Se añade la funcionalidad al servicio de pago online y se añadió su POJO |
| Se quita la tabla de Empresas y se modifica el idPedido para que fuera string para mantener la congruencia con el BPEL. |
| Se completa la funcionalidad del servicio web de mensajería express y se añade su POJO |
| Se añaden los BPEL y CA del proyecto y se añaden los comentarios necesarios para documentar el código |
| 1-dic. | Se cambia el funcionamiento general de los webServices y se añaden funcionalidades para que se reduzca stock y saldo |
| 2-dic. | Se modifica la manera en la que se realiza la conexión en todos los web services para poder asegurar una conexión estable en todo momento; además, se agrega la funcionalidad de reducir el saldo y el inventario de productos al realizar una compra; finalmente se añaden anotaciones. |
| 3-dic. | Se corrige la funcionalidad de comprobar saldos y unidades y se agregan pruebas en el CA |
| Se añade el POJO tstGeneral para realizar las pruebas |
| Se agregan los estresadores de todos los pojos |
| 13-dic. | Se incluye funcionalidad para alta y modificación de clientes y productos |
| Se agregan los tests necesarios para cada uno de los casos |
| 14-dic | Se realizan las pruebas de estress para cada uno de los componentes del sistema |

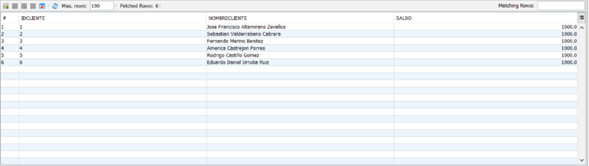
# 

# 

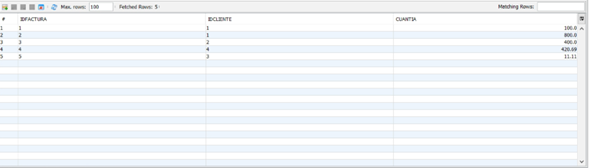
# **Documentación de pruebas funcionales**

El “Steady State” es el estado base con el cual se trabaja para las pruebas funcionales y no funcionales del proyecto. Todos los casos comienzan con el mismo estado base. A continuación se muestra cuál es este estado para todas las pruebas:

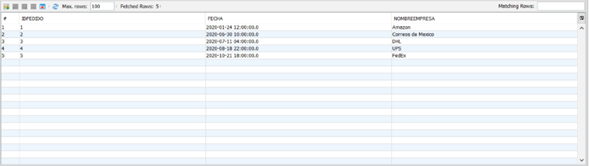
**Tabla Clientes:**

****

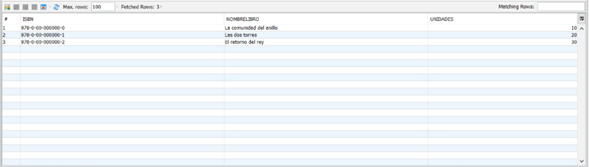
**Tabla Facturas:**

****

**Tabla Pedidos:**

****

**Tabla Productos:**

****

En el primer anexo de este documento se puede encontrar el script de carga de la base de datos para el estado base.

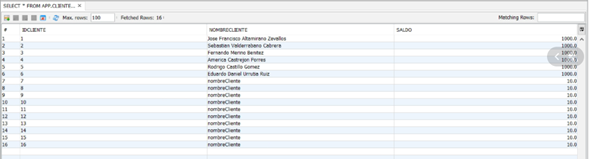
Transacciones Habituales

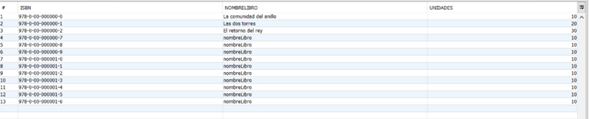
Dado el diseño del sistema las transacciones habituales son:

* Dar de alta un usuario
* Dar de alta un libro
* Aumentar el stock
* Comprar un libro (bajar el stock)

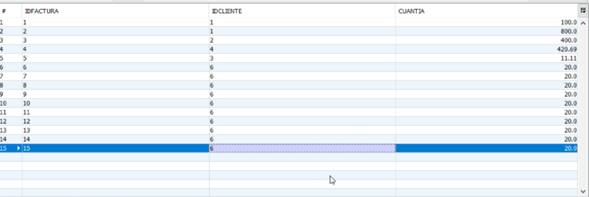
Para probar las pruebas funcionales es necesario tener en cuenta los cambios producidos en la base de datos para verificar que todos los servicios se están ejecutando de la forma esperada. Para verificar se realizan 10 solicitudes por 1 solo cliente. A continuación, se muestra la comparación del contenido de la base de datos en “Steady State” y los resultados después de las pruebas.

Resultados Servicio Almacén:

****

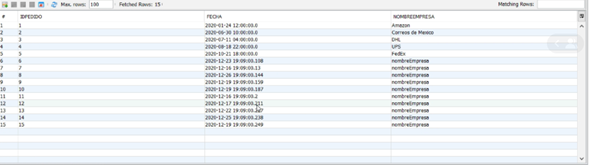
****

Resultados Servicio Pago Online:

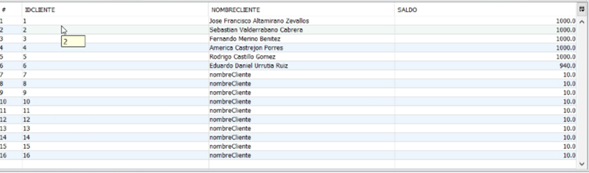
****

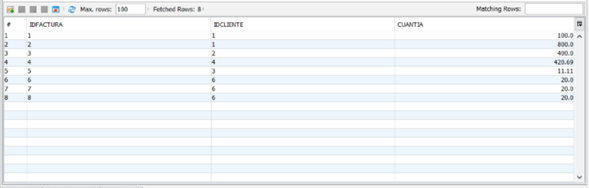
****

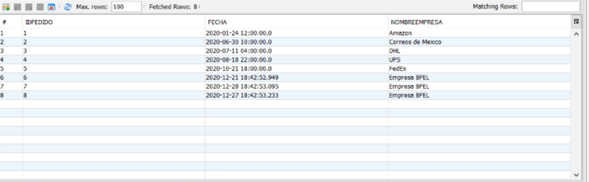
Resultados Servicio Mensajería Exprés:

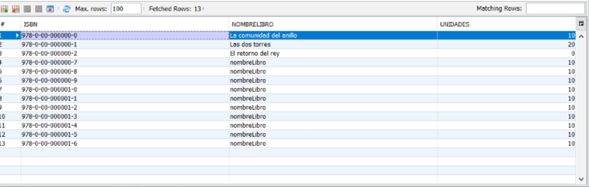
****

Para corroborar el cumplimiento, de principio se realizaron las pruebas individuales por cada uno de los servicios; posteriormente, se realizó la prueba desde un servicio integrado para verificar que los datos sean iguales a los esperados. De esta forma, funciona de manera automática para corroborar que realmente se está cumpliendo la funcionalidad.





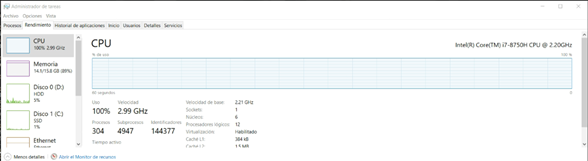




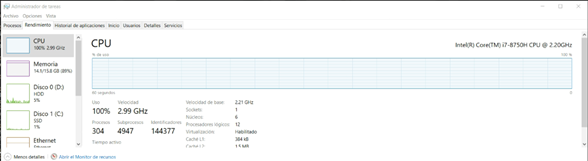
# **Documentación de pruebas no funcionales**

Se establecieron pruebas de estrés para medir los requerimientos no funcionales que se establecieron anteriormente. Se muestran las capturas que incluyen el rendimiento del CPU junto a la Memoria utilizada por cada uno de los servicios. Del mismo modo, todos los datos se recopilaron dentro de la tabla de este apartado.

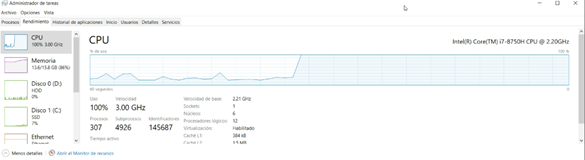
Rendimiento sobre Servicio Integrado:



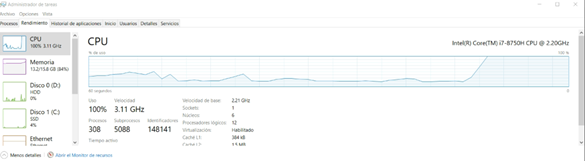
Rendimiento sobre Servicio Almacén:



Rendimiento sobre Servicio Pago Online:



Rendimiento sobre Servicio Mensajería Exprés:



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Servicios | Clientes | Solicitudes | Media tiempo ciclo | CPU | Memoria | Cumplimiento de Volumetrics |
| Servicio Integrado | 20 | 200 | 3,116 ms | 100% | 89% | Sí |
| almacén / Tienda Online | 20 | 200 | 202 ms | 100% | 75% | Sí |
| Pago Online / Pasarela Online | 20 | 200 | 187 ms | 100% | 86% | Sí |
| Mensajería Exprés | 20 | 200 | 77 ms | 100% | 84% | Sí |

De forma concluyente queda demostrado el correcto cumplimiento de los requerimientos no funcionales. A continuación, se muestra el Script de la Base de Datos en la misma versión de “Steady State” que se carga mediante el ij de Glassfish:

*connect 'jdbc:derby://localhost:1527/serviciosVenta;user=app;password=app';*

*DROP TABLE pedidos;*

*DROP TABLE facturas;*

*DROP TABLE clientes;*

*DROP TABLE productos;*

*-------------------------------*

*-- CLIENTES*

*-------------------------------*

*CREATE TABLE clientes*

*(*

*idCliente VARCHAR(100) NOT NULL ,*

*nombreCliente VARCHAR(100) NOT NULL ,*

*saldo FLOAT NOT NULL ,*

*PRIMARY KEY (idCliente)*

*);*

*INSERT INTO clientes (idCliente, nombreCliente, saldo) VALUES*

*('1', 'Jose Francisco Altamirano Zevallos', 1000),*

*('2', 'Sebastian Valderrabano Cabrera', 1000),*

*('3', 'Fernando Merino Benitez', 1000),*

*('4', 'America Castrejon Porres', 1000),*

*('5', 'Rodrigo Castillo Gomez', 1000),*

*('6', 'Eduardo Daniel Urrutia Ruiz', 1000);*

-------------------------------

-- PRODUCTOS

-------------------------------

CREATE TABLE productos

(

ISBN VARCHAR(100) NOT NULL ,

nombreLibro VARCHAR(100) NOT NULL ,

unidades INT NOT NULL ,

PRIMARY KEY (ISBN)

);

INSERT INTO productos (ISBN, nombreLibro, unidades) VALUES

('978-0-00-000000-0', 'La comunidad del anillo', 10),

('978-0-00-000000-1', 'Las dos torres', 20),

('978-0-00-000000-2', 'El retorno del rey', 30);

-------------------------------

-- FACTURAS

-------------------------------

CREATE TABLE facturas

(

idFactura VARCHAR(100) NOT NULL ,

idCliente VARCHAR(100) NOT NULL ,

cuantia FLOAT NOT NULL ,

PRIMARY KEY (idFactura),

CONSTRAINT fk\_idCliente

FOREIGN KEY (idCliente )

REFERENCES clientes (idCliente )

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

INSERT INTO facturas (idFactura, idCliente, cuantia) VALUES

('1', '1', 100),

('2', '1', 800),

('3', '2', 400),

('4', '4', 420.69),

('5', '3', 11.11);

-------------------------------

-- PEDIDOS

-------------------------------

CREATE TABLE pedidos

(

idPedido VARCHAR(100) NOT NULL ,

fecha TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ,

nombreEmpresa VARCHAR(100) NOT NULL ,

PRIMARY KEY (idPedido)

);

INSERT INTO pedidos (idPedido, fecha, nombreEmpresa) VALUES

('1', TIMESTAMP('2020-01-24 12:00:00'), 'Amazon'),

('2', TIMESTAMP('2020-06-30 10:00:00'), 'Correos de Mexico'),

('3', TIMESTAMP('2020-07-11 04:00:00'), 'DHL'),

('4', TIMESTAMP('2020-08-18 22:00:00'), 'UPS'),

('5', TIMESTAMP('2020-10-21 18:00:00'), 'FedEx');

disconnect;

### 

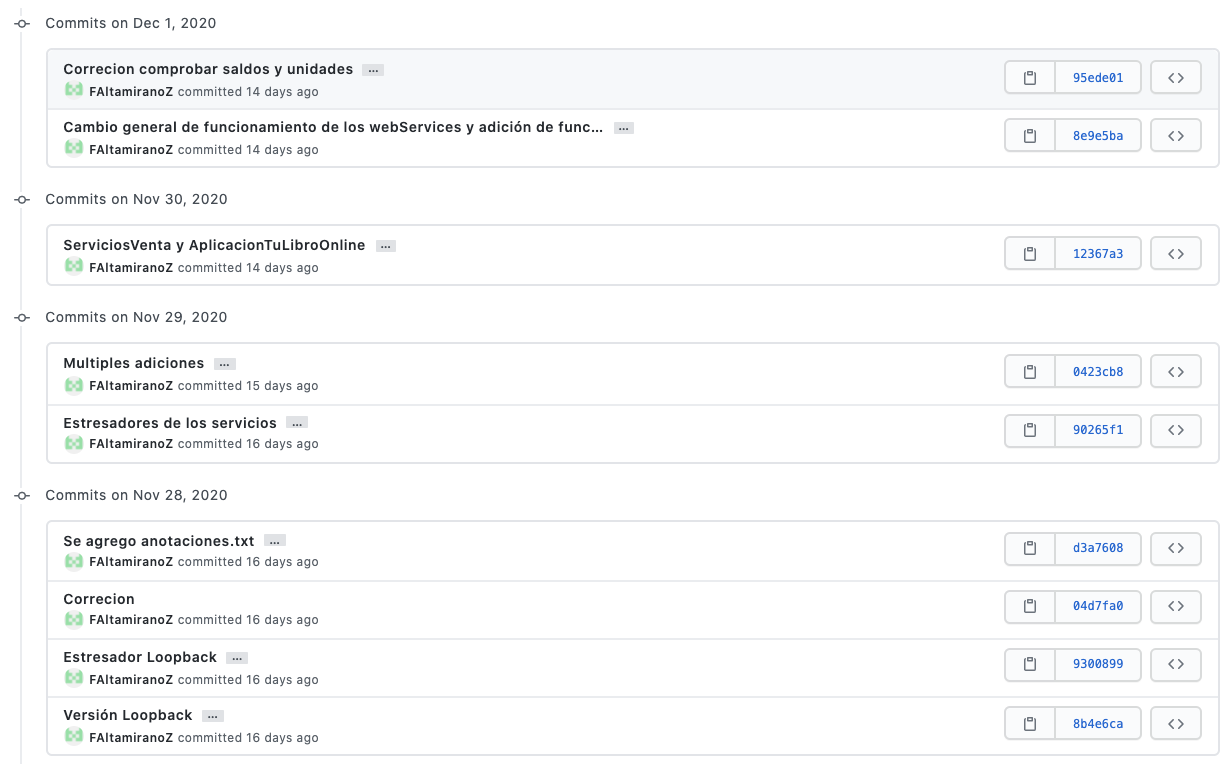
# 

# **Arquitectura y versiones**

Como se ha mencionado en las secciones anteriores, para llevar un control de versiones se utilizó github; es por ello que en la tabla que se muestra a continuación el número de versión corresponde al asignado por la plataforma.

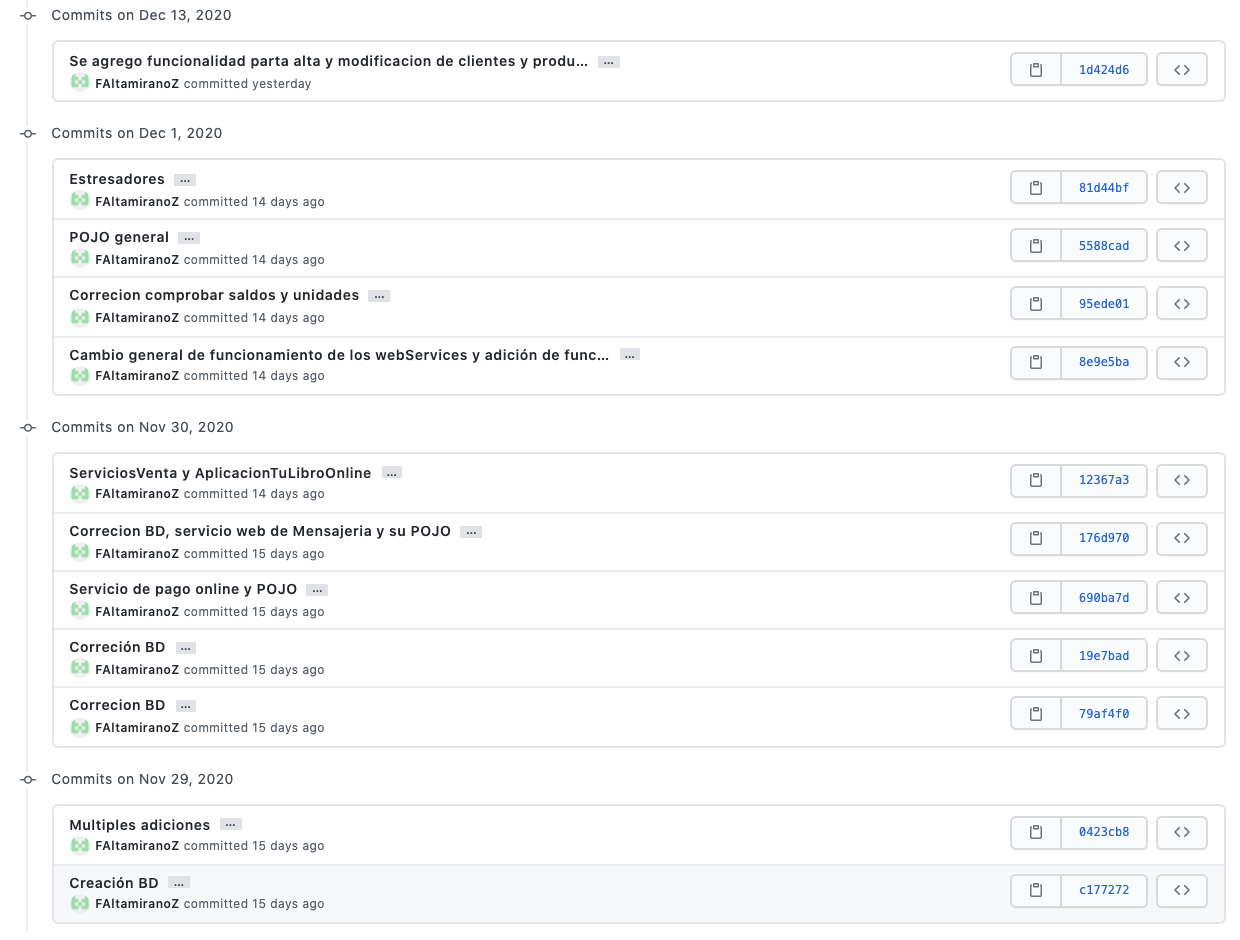
## Versiones Loopback

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 27-nov. | v1 | WebServices |
| 28-nov. | v1 | Pojos de webServices |
| v1 | anotaciones.txt |
| 29-nov. | v1 | Estresador de servicios |
| v1 | Proyectos web |
| 30-nov. | v1 | ServiciosVenta |
| v1 | AplicacionTuLibroOnline |
| 1-dic. | v2 | Cambio de webServices |
| 2-dic. | v2 | Comprobar saldo |
| v3 | Comprobar unidades |



## Versiones BD

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 29-nov. | v1 | Creación BD |
| v1 | TiendaElectronica |
| 30-nov. | v2 | Corrección BD |
| v2 | Funcionalidad servicio de pago |
| v3 | Corrección BD |
| v3 | Serivico web mensajeria y POJO |
| v3 | BPEL y CA |
| 1-dic. | v4 | Modificacion webServices |
| 2-dic. | v5 | Cambio conexión webServices |
| 3-dic. | v6 | Correccion funcionalidad saldos |
| v6 | POJO tstGeneral |
| v6 | Estresadores POJO |
| 13-dic. | v6 | Funcionalidad alta y modificación |
| v6 | Test de funcionalidades |

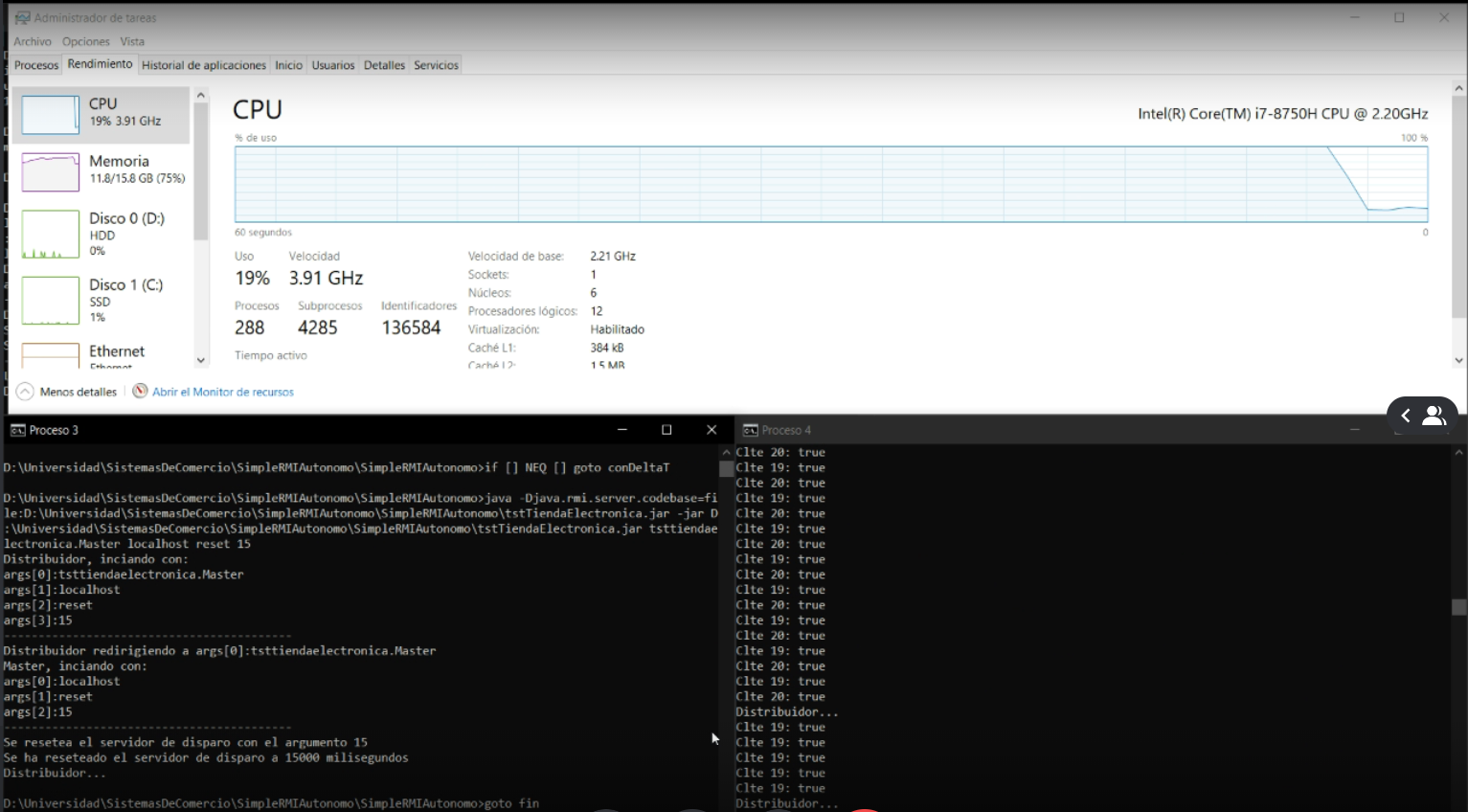


# **Web Service de Almacén**

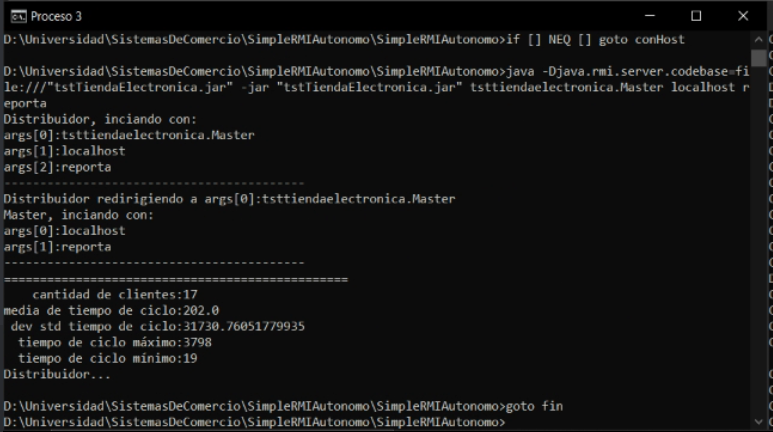
|  |  |  |
| --- | --- | --- |
|  | Requerimientos Funcionales | Requerimientos No Funcionales |
| WS Almacén | RN1,RN2,RN3,RN4,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF6 |

El servicio web de almacén o tienda electrónica es de los más importantes para el correcto funcionamiento del proyecto. Siendo necesario para cumplir cinco requerimientos funcionales y cuatro no funcionales. Al ser este servicio el que se encarga de hacer verificaciones para realizar distintas operaciones es clave que este funcione correctamente.

En nuestras pruebas funcionales, se verificó que los artículos disponibles estén registrados y puedan ser consultados usando la clave ISBN. Al momento en que se realiza una transacción, la base de datos refleja tanto un id del medio de pago y la cantidad de ejemplares disponibles según el ISBN utilizado para registrar un artículo.



Este proceso asegura que no se hagan ventas de ejemplares que no se encuentran disponibles y que cuando se haga una compra se reflejen las modificaciones necesarias tanto en saldo como en existencia de artículos. También se comprobó que se realizan altas de nuevos clientes y libros en la base de datos.

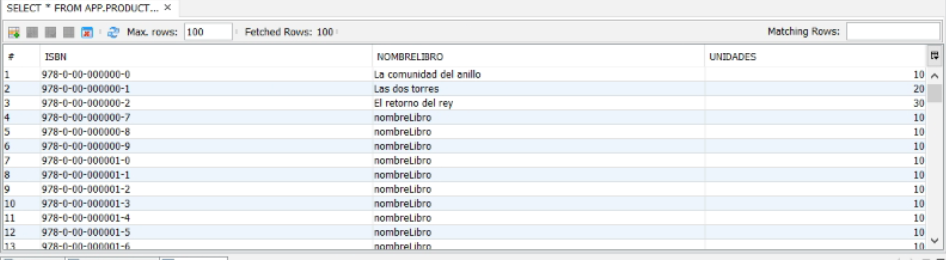


Además se estresó el servicio para verificar que soportara 100 transacciones por minuto. Con estas transacciones se comprueba el funcionamiento de altas y modificaciones, y buscamos que el tiempo de respuesta sea menor a cinco segundos. Para estas operaciones es necesario que funcionen adecuadamente con un mínimo de 10 usuarios concurrentes. Finalmente, se mostrarán mensajes de error cuando no se pueda completar una compra o transacción.

* No. Clientes: 4000



* No. Libros: 4000



* Prueba funcional con 10 clientes.



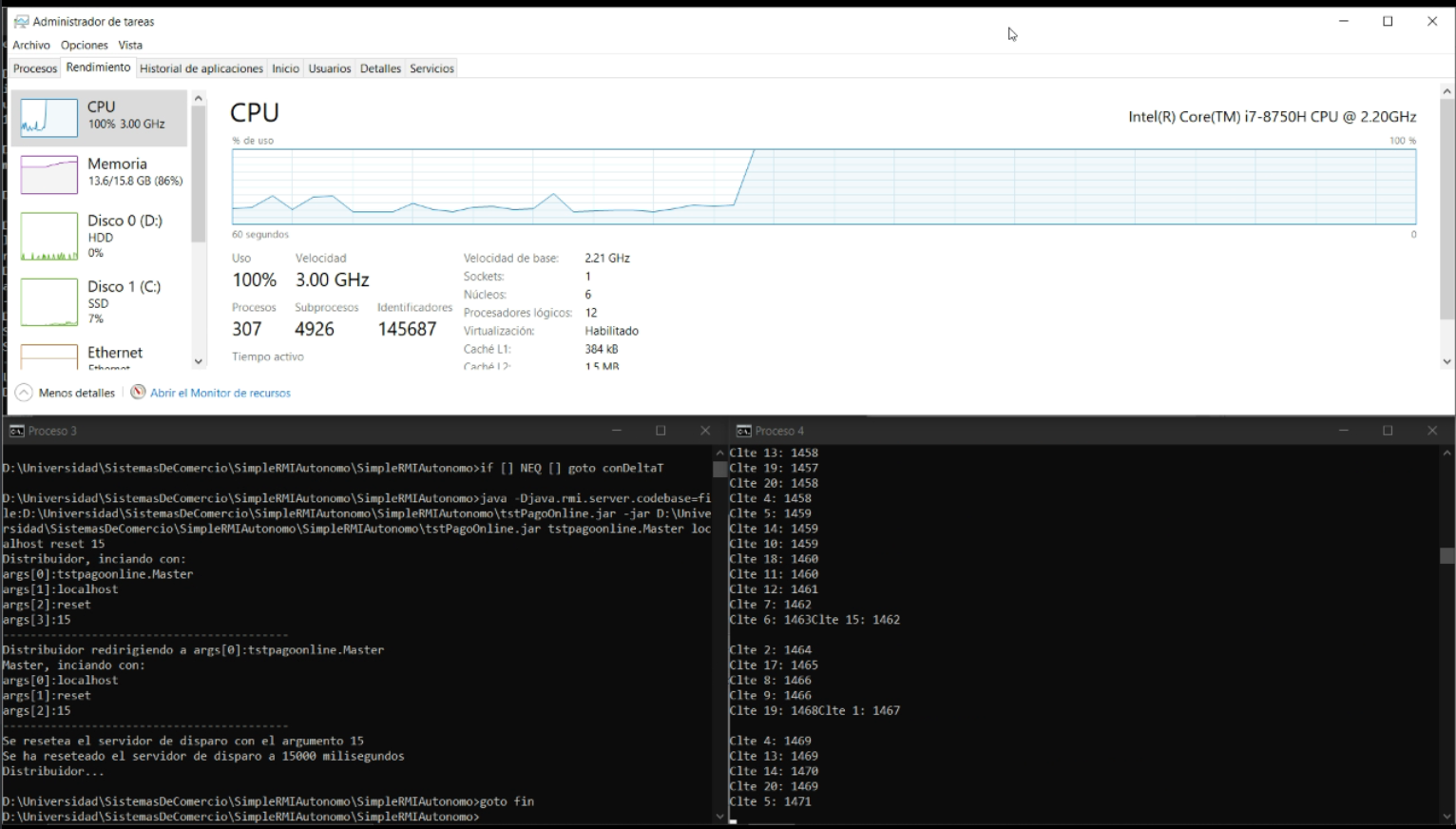
# **Web Service de Pago**

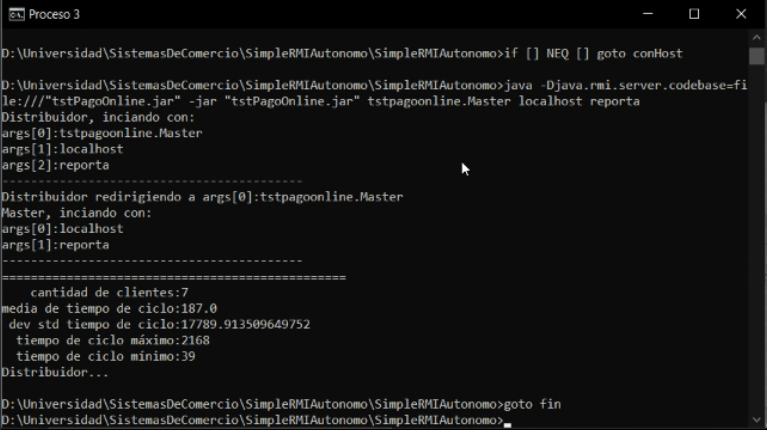
|  |  |  |
| --- | --- | --- |
|  | Requerimientos Funcionales | Requerimientos No Funcionales |
| WS Pago | RN1,RN4,RN5,RN7,RN8 | RNF1,RNF2,RNF3,RNF4,RNF5,RNF6 |

El web service de pago está encargado de los requerimientos de negocio, tanto funcionales como no funcionales que se muestran en la tabla anterior. Este web service es el encargado de realizar la factura y registrarla en la base de datos, así como actualizar el saldo del cliente una vez realizada la compra.

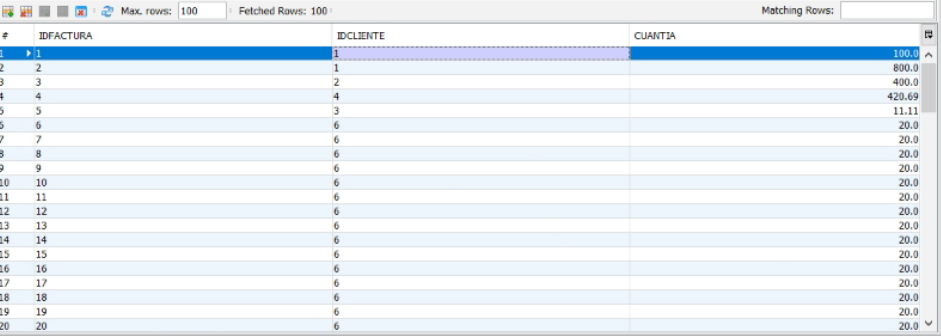
Para las pruebas funcionales, se verificó que al momento en que se realiza una transacción, la base de datos refleja el nuevo saldo del cliente y la cantidad de ejemplares disponibles según el ISBN utilizado para registrar un artículo. De igual manera se verificó que la factura estuviera dentro de la base de datos.

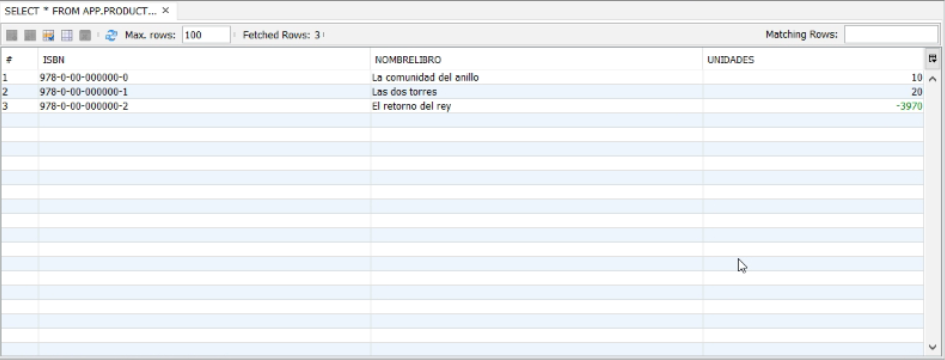
A continuación podemos encontrar los resultados de las pruebas funcionales y no funcionales. Se estreso el web service con 10 clientes concurrentes y se comprobó su correcto funcionamiento al tener que generar 4000 facturas y al realizar 400 compras.



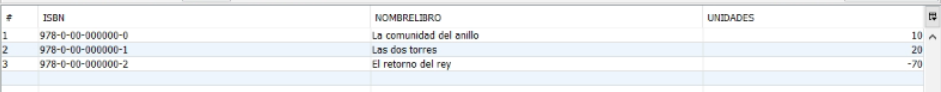
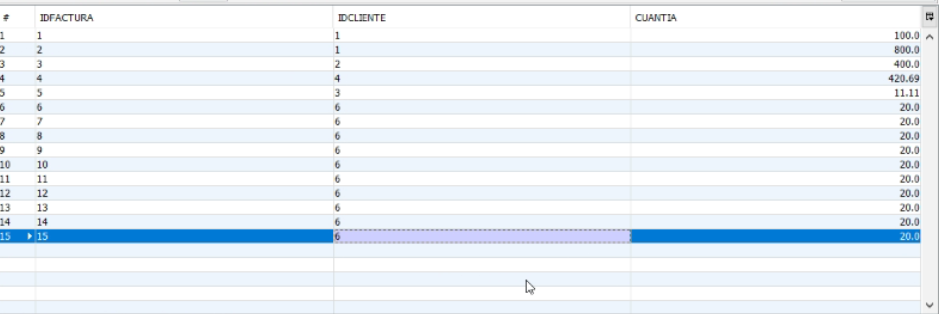


* Genera 4000 facturas





* Comprar 400 veces

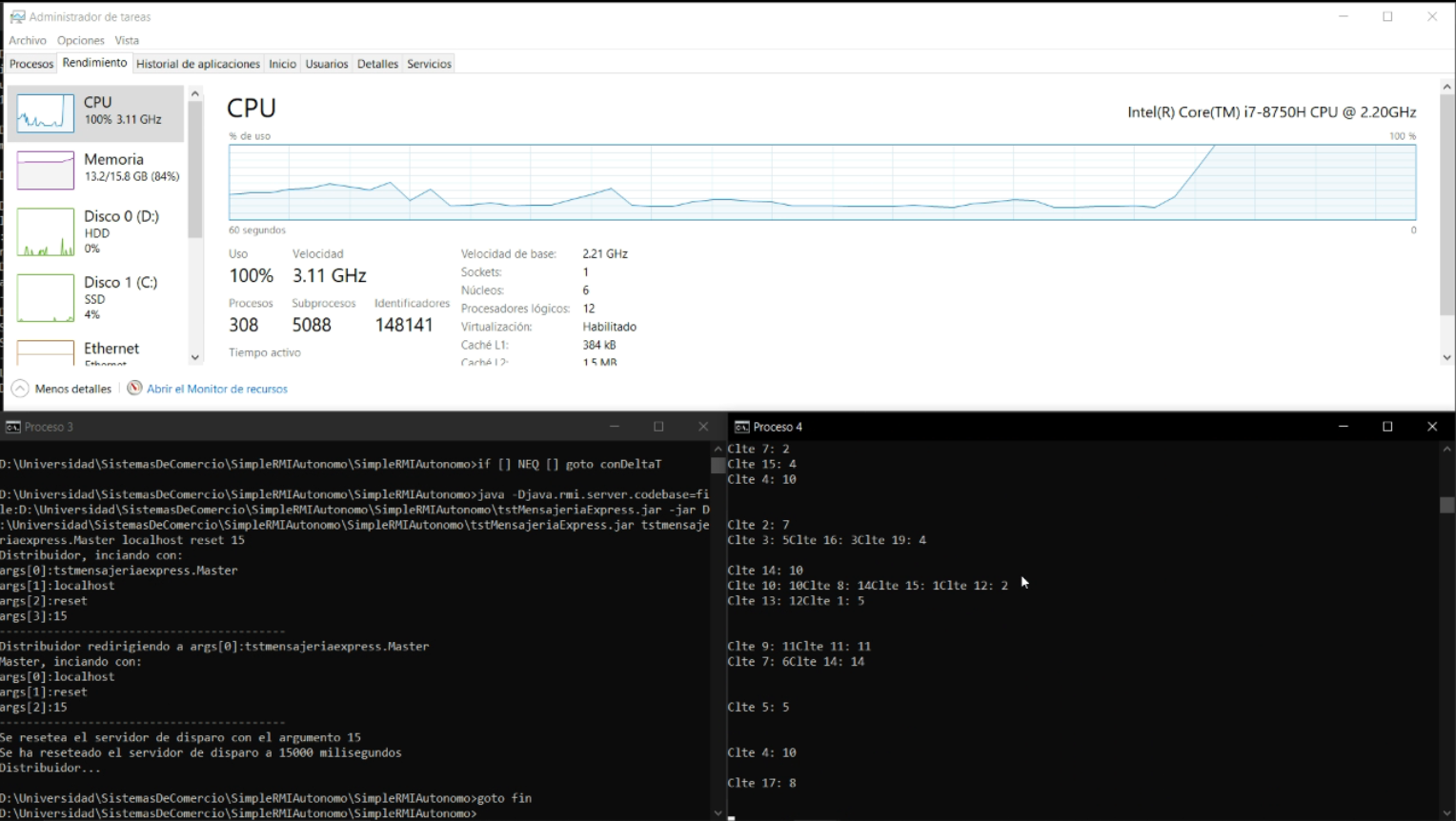


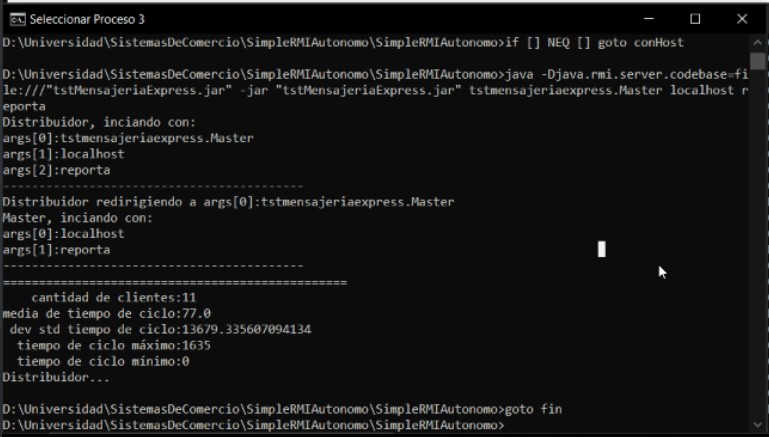
# **Web Service de Entregas**

|  |  |  |
| --- | --- | --- |
|  | Requerimientos Funcionales | Requerimientos No Funcionales |
| WS Mensajería | RN6 | RNF1,RNF2,RNF3,RNF4,RNF5 |

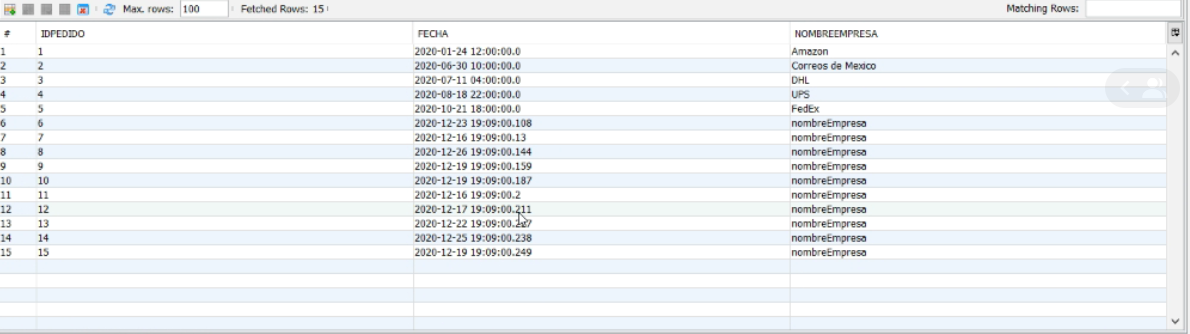
El web service de entregas es el encargado de los requerimientos de negocio, tanto funcionales como no funcionales que se muestran en la tabla anterior. Este web service es el encargado de enviar el pedido al cliente una vez realizada la compra, y agregar dicho pedido a la base de datos.

Para las pruebas funcionales, se verificó que una vez realizada la compra la tabla de pedidos estuviera actualizada. A continuación podemos encontrar los resultados de las pruebas funcionales y no funcionales. Se comprobó el comportamiento del web service con 10 clientes concurrentes y al realizar 10 y 4000 pedidos.

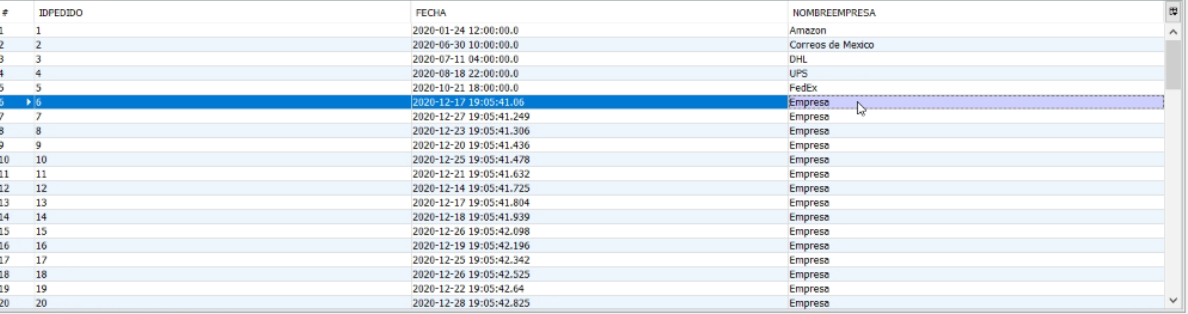




* 10 pedidos



* 4000 pedidos

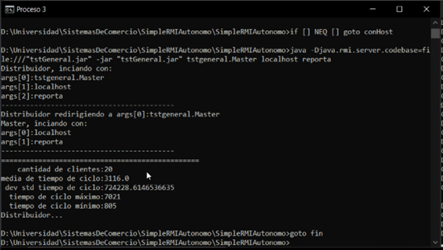


# 

# **Estress general de cada servicio y a la aplicación**

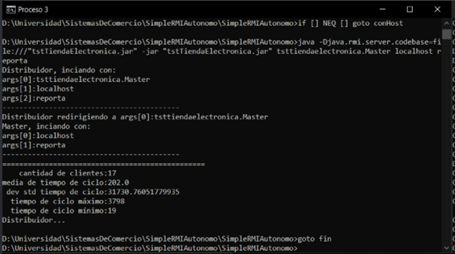
Resultados del estresamiento para las versiones estables

* *Servicio Integrado*





* *Almacén*

**

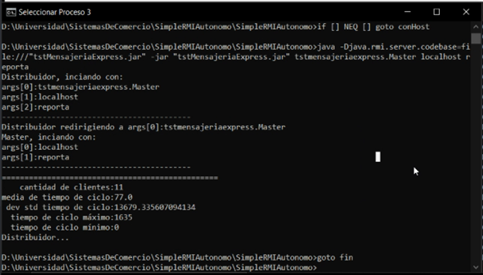
**

* *Pago Online*

**

**

* *Mensajería Exprés*

**

**

Para lograr estos resultados se tuvieron que adaptar las versiones de los pojos de servicio. Esto nos permitió realizar las pruebas exitosamente. A continuación, se muestran los Scripts de los pojos relacionados a cada uno de los servicios que se utilizaron para el estresamiento.

### *Pojo de Servicio Integrado*

//================================================================

// Código del Cliente:

//================================================================

package tstgeneral;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Cliente {

public static void main(String[] args) {

long lngQuienSoy;

long sumDeltaT, sumDeltaT2, dtMin = 0, dtMax = 0;

long lngCuantosMilisFaltan;

String host = (args.length < 1) ? null : args[0];

long i, n, t0, t1, dt;

String response;

if (args[1] != null) {

n = Integer.parseInt(args[1]);

} else {

n = 200;

}

try {

Registry registry = LocateRegistry.getRegistry(host);

IServDisparo servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

lngQuienSoy = servDisparo.quienSoy();

lngCuantosMilisFaltan = servDisparo.deltaTEnMilis();

System.out.println("Cliente " + lngQuienSoy + " faltan " + lngCuantosMilisFaltan + " milisegundos");

sumDeltaT = 0;

sumDeltaT2 = 0;

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service = new org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService();

for (i = 0; i < n; i++) {

t0 = System.currentTimeMillis();

response = ventaLibrosWSDLOperation("978-0-00-000000-2", "6", 10, 2, service);

t1 = System.currentTimeMillis();

dt = t1 - t0;

sumDeltaT += dt;

sumDeltaT2 += dt \* dt;

if (i == 0) {

dtMin = dt;

dtMax = dt;

} else {

if (dt < dtMin) {

dtMin = dt;

}

if (dt > dtMax) {

dtMax = dt;

}

}

System.out.println("Clte " + lngQuienSoy + ": " + response);

}

servDisparo.acumula(sumDeltaT, sumDeltaT2, n, dtMax, dtMin);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

private static String ventaLibrosWSDLOperation(java.lang.String isbn, java.lang.String idCliente, int unidades, int precioUnidad, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.ventaLibrosWSDLOperation(isbn, idCliente, unidades, precioUnidad);

}

}

* *Pojo de Servicio Almacén*

//================================================================

// Código del Cliente:

//================================================================

package tsttiendaelectronica;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Cliente {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

long lngQuienSoy;

long sumDeltaT, sumDeltaT2, dtMin = 0, dtMax = 0;

long lngCuantosMilisFaltan;

String host = (args.length < 1) ? null : args[0];

long i, n, t0, t1, dt;

boolean response;

if (args[1] != null) {

n = Integer.parseInt(args[1]);

} else {

n = 200;

}

try {

Registry registry = LocateRegistry.getRegistry(host);

IServDisparo servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

lngQuienSoy = servDisparo.quienSoy();

lngCuantosMilisFaltan = servDisparo.deltaTEnMilis();

System.out.println("Cliente " + lngQuienSoy + " faltan " + lngCuantosMilisFaltan + " milisegundos");

sumDeltaT = 0;

sumDeltaT2 = 0;

com.adictosaltrabajo.webservice.almacen.AlmacenService service = new com.adictosaltrabajo.webservice.almacen.AlmacenService();

for (i = 0; i < n; i++) {

t0 = System.currentTimeMillis();

comprobarISBN("ISBN", service);

comprobarStock("978-0-00-000000-0", 5, service);

comprobarIdCliente("1", service);

response = comprobarSaldo(5, 10, "1", service);

t1 = System.currentTimeMillis();

dt = t1 - t0;

sumDeltaT += dt;

sumDeltaT2 += dt \* dt;

if (i == 0) {

dtMin = dt;

dtMax = dt;

} else {

if (dt < dtMin) {

dtMin = dt;

}

if (dt > dtMax) {

dtMax = dt;

}

}

System.out.println("Clte " + lngQuienSoy + ": " + response);

}

servDisparo.acumula(sumDeltaT, sumDeltaT2, n, dtMax, dtMin);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

private static Boolean comprobarISBN(java.lang.String isbn, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarISBN(isbn);

}

private static Boolean comprobarStock(java.lang.String isbn, int unidades, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarStock(isbn, unidades);

}

private static Boolean comprobarIdCliente(String idCliente, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarIdCliente(idCliente);

}

private static Boolean comprobarSaldo(int unidades, int precioUnidad, String idCliente, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarSaldo(unidades, precioUnidad, idCliente);

}

}

* *Pojo de Servicio Pago Online*

//================================================================

// Código del Cliente:

//================================================================

package tstpagoonline;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Cliente {

public static void main(String[] args) {

long lngQuienSoy;

long sumDeltaT, sumDeltaT2, dtMin = 0, dtMax = 0;

long lngCuantosMilisFaltan;

String host = (args.length < 1) ? null : args[0];

long i, n, t0, t1, dt;

String response;

if (args[1] != null) {

n = Integer.parseInt(args[1]);

} else {

n = 200;

}

try {

Registry registry = LocateRegistry.getRegistry(host);

IServDisparo servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

lngQuienSoy = servDisparo.quienSoy();

lngCuantosMilisFaltan = servDisparo.deltaTEnMilis();

System.out.println("Cliente " + lngQuienSoy + " faltan " + lngCuantosMilisFaltan + " milisegundos");

sumDeltaT = 0;

sumDeltaT2 = 0;

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service = new com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService();

for (i = 0; i < n; i++) {

t0 = System.currentTimeMillis();

actualizarInventario("6", 20, "978-0-00-000000-2", 10, service);

response = facturar("6", 20, service);

t1 = System.currentTimeMillis();

dt = t1 - t0;

sumDeltaT += dt;

sumDeltaT2 += dt \* dt;

if (i == 0) {

dtMin = dt;

dtMax = dt;

} else {

if (dt < dtMin) {

dtMin = dt;

}

if (dt > dtMax) {

dtMax = dt;

}

}

System.out.println("Clte " + lngQuienSoy + ": " + response);

}

servDisparo.acumula(sumDeltaT, sumDeltaT2, n, dtMax, dtMin);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

private static String facturar(java.lang.String idCliente, int cuantia, com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service) {

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPago port = service.getPasarelaPagoPort();

return port.facturar(idCliente, cuantia);

}

private static boolean actualizarInventario(java.lang.String idCliente, int cuantia, java.lang.String isbn, int unidades, com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service) {

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPago port = service.getPasarelaPagoPort();

return port.actualizarInventario(idCliente, cuantia, isbn, unidades);

}

}

* *Pojo de Servicio Mensajería Exprés*

//================================================================

// Código del Cliente:

//================================================================

package tstmensajeriaexpress;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Cliente {

public static void main(String[] args) {

long lngQuienSoy;

long sumDeltaT, sumDeltaT2, dtMin = 0, dtMax = 0;

long lngCuantosMilisFaltan;

String host = (args.length < 1) ? null : args[0];

long i, n, t0, t1, dt;

Integer response;

if (args[1] != null) {

n = Integer.parseInt(args[1]);

} else {

n = 200;

}

try {

Registry registry = LocateRegistry.getRegistry(host);

IServDisparo servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

lngQuienSoy = servDisparo.quienSoy();

lngCuantosMilisFaltan = servDisparo.deltaTEnMilis();

System.out.println("Cliente " + lngQuienSoy + " faltan " + lngCuantosMilisFaltan + " milisegundos");

sumDeltaT = 0;

sumDeltaT2 = 0;

com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService service = new com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService();

for (i = 6; i < n; i++) {

t0 = System.currentTimeMillis();

response = enviarProducto("Empresa", Long.toString(i), service);

t1 = System.currentTimeMillis();

dt = t1 - t0;

sumDeltaT += dt;

sumDeltaT2 += dt \* dt;

if (i == 0) {

dtMin = dt;

dtMax = dt;

} else {

if (dt < dtMin) {

dtMin = dt;

}

if (dt > dtMax) {

dtMax = dt;

}

}

System.out.println("Clte " + lngQuienSoy + ": " + response);

}

servDisparo.acumula(sumDeltaT, sumDeltaT2, n, dtMax, dtMin);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

private static Integer enviarProducto(java.lang.String empresa, java.lang.String idPedido, com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService service){

com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetes port = service.getEnvioPaquetesPort();

return port.enviarProducto(empresa, idPedido);

}

}

# **Anexos:**

## Scripts de inicialización de BD de los servicios básicos.

connect 'jdbc:derby://localhost:1527/serviciosVenta;user=app;password=app';

DROP TABLE pedidos;

DROP TABLE facturas;

DROP TABLE clientes;

DROP TABLE productos;

-------------------------------

-- CLIENTES

-------------------------------

CREATE TABLE clientes

(

idCliente VARCHAR(100) NOT NULL ,

nombreCliente VARCHAR(100) NOT NULL ,

saldo FLOAT NOT NULL ,

PRIMARY KEY (idCliente)

);

INSERT INTO clientes (idCliente, nombreCliente, saldo) VALUES

('1', 'Jose Francisco Altamirano Zevallos', 1000),

('2', 'Sebastian Valderrabano Cabrera', 1000),

('3', 'Fernando Merino Benitez', 1000),

('4', 'America Castrejon Porres', 1000),

('5', 'Rodrigo Castillo Gomez', 1000),

('6', 'Eduardo Daniel Urrutia Ruiz', 1000);

-------------------------------

-- PRODUCTOS

-------------------------------

CREATE TABLE productos

(

ISBN VARCHAR(100) NOT NULL ,

nombreLibro VARCHAR(100) NOT NULL ,

unidades INT NOT NULL ,

PRIMARY KEY (ISBN)

);

INSERT INTO productos (ISBN, nombreLibro, unidades) VALUES

('978-0-00-000000-0', 'La comunidad del anillo', 10),

('978-0-00-000000-1', 'Las dos torres', 20),

('978-0-00-000000-2', 'El retorno del rey', 30);

-------------------------------

-- FACTURAS

-------------------------------

CREATE TABLE facturas

(

idFactura VARCHAR(100) NOT NULL ,

idCliente VARCHAR(100) NOT NULL ,

cuantia FLOAT NOT NULL ,

PRIMARY KEY (idFactura),

CONSTRAINT fk\_idCliente

FOREIGN KEY (idCliente )

REFERENCES clientes (idCliente )

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

INSERT INTO facturas (idFactura, idCliente, cuantia) VALUES

('1', '1', 100),

('2', '1', 800),

('3', '2', 400),

('4', '4', 420.69),

('5', '3', 11.11);

-------------------------------

-- PEDIDOS

-------------------------------

CREATE TABLE pedidos

(

idPedido VARCHAR(100) NOT NULL ,

fecha TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ,

nombreEmpresa VARCHAR(100) NOT NULL ,

PRIMARY KEY (idPedido)

);

INSERT INTO pedidos (idPedido, fecha, nombreEmpresa) VALUES

('1', TIMESTAMP('2020-01-24 12:00:00'), 'Amazon'),

('2', TIMESTAMP('2020-06-30 10:00:00'), 'Correos de Mexico'),

('3', TIMESTAMP('2020-07-11 04:00:00'), 'DHL'),

('4', TIMESTAMP('2020-08-18 22:00:00'), 'UPS'),

('5', TIMESTAMP('2020-10-21 18:00:00'), 'FedEx');

disconnect;

## Código de web services y acceso a BD.

@WebService()

public class EnvioPaquetes {

/\*\*

\* Operacion de un servicio web implementado con JAX-WS que emite la orden

\* de envio de un producto.

\*

\* @param empresa Identificador de empresa

\* @param idPedido Identificador de pedido

\* @return Tiempo de reparto en dias

\*/

@WebMethod(operationName = "enviarProducto")

public Integer enviarProducto(@WebParam(name = "empresa") String empresa, @WebParam(name = "idPedido") String idPedido) {

Random rnd = new Random();

int dias = rnd.nextInt(15) + 1;

long temp = dias;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

s.executeUpdate("INSERT INTO pedidos (idPedido, fecha, nombreEmpresa) VALUES ('" + idPedido + "', TIMESTAMP('" + new Timestamp((System.currentTimeMillis() + temp \* 24 \* 60 \* 60 \* 1000)) + "'), '" + empresa + "')");

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return dias;

}

}

@WebService()

public class PasarelaPago {

/\*\*

\* Operacion de un servicio web implementado con JAX-WS que factura a un

\* determinado cliente cierta cuantia.

\*

\* @param idCliente Identificador de un cliente

\* @param cuantia Cuantia a cobrar al cliente

\* @return Identificador de factura

\*/

@WebMethod(operationName = "facturar")

public String facturar(@WebParam(name = "idCliente") String idCliente, @WebParam(name = "cuantia") int cuantia) {

String idFactura = "No se registró la factura";

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs;

rs = s.executeQuery("select max(CAST(idFactura as int)) from facturas");

rs.next();

idFactura = rs.getString(1);

// System.out.println(idFactura);

int temp = Integer.parseInt(idFactura) + 1;

// System.out.println(temp);

idFactura = Integer.toString(temp);

// System.out.println("INSERT INTO facturas (idFactura, idCliente, cuantia) VALUES ('" + idFactura + "', '" + idCliente + "', " + cuantia + ")");

s.executeUpdate("INSERT INTO facturas (idFactura, idCliente, cuantia) VALUES ('" + idFactura + "', '" + idCliente + "', " + cuantia + ")");

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return idFactura;

}

/\*\*

\*

\* @param idCliente

\* @param cuantia

\* @param ISBN

\* @param unidades

\* @return

\*/

@WebMethod(operationName = "actualizarInventario")

public Boolean actualizarInventario(@WebParam(name = "idCliente") String idCliente, @WebParam(name = "cuantia") int cuantia, @WebParam(name = "ISBN") String ISBN, @WebParam(name = "unidades") int unidades) {

boolean flag = false;

float saldo;

int unidadesTotales;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs1, rs2;

// System.out.println(idCliente);

// System.out.println(cuantia);

// System.out.println(ISBN);

// System.out.println(unidades);

// System.out.println("select saldo from clientes where idcliente = '" + idCliente + "'");

rs1 = s.executeQuery("select saldo from clientes where idcliente = '" + idCliente + "'");

rs1.next();

saldo = rs1.getFloat(1);

// System.out.println(saldo);

// System.out.println("update clientes set saldo = " + (saldo - cuantia) + " where idcliente = '" + idCliente + "'");

s.executeUpdate("update clientes set saldo = " + (saldo - cuantia) + " where idcliente = '" + idCliente + "'");

rs1.close();

// System.out.println("select unidades from productos where ISBN = '" + ISBN + "'");

rs2 = s.executeQuery("select unidades from productos where ISBN = '" + ISBN + "'");

rs2.next();

unidadesTotales = rs2.getInt(1);

// System.out.println(unidadesTotales);

// System.out.println("update productos set unidades = " + (unidadesTotales - unidades) + " where ISBN = '" + ISBN + "'");

s.executeUpdate("update productos set unidades = " + (unidadesTotales - unidades) + " where ISBN = '" + ISBN + "'");

rs2.close();

s.close();

con.close();

flag = true;

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

}

@WebService()

public class Almacen {

/\*\*

\* @param ISBN

\* @return

\*/

@WebMethod(operationName = "comprobarISBN")

public Boolean comprobarISBN(@WebParam(name = "ISBN") String ISBN) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select ISBN from productos where ISBN = '" + ISBN + "'");

if (rs.next()) {

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "comprobarStock")

public Boolean comprobarStock(@WebParam(name = "ISBN") String ISBN, @WebParam(name = "unidades") int unidades) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select ISBN from productos where ISBN = '" + ISBN + "' and unidades >= " + unidades);

if (rs.next()) {

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "comprobarIdCliente")

public Boolean comprobarIdCliente(@WebParam(name = "idCliente") String idCliente) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select idCliente from clientes where idCliente = '" + idCliente + "'");

if (rs.next()) {

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "comprobarSaldo")

public Boolean comprobarSaldo(@WebParam(name = "unidades") int unidades, @WebParam(name = "precioUnidad") int precioUnidad, @WebParam(name = "idCliente") String idCliente) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select saldo from clientes where saldo >= " + (unidades \* precioUnidad) + "and idCliente = '" + idCliente + "'");

if (rs.next()) {

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "altaCliente")

public Boolean altaCliente(@WebParam(name = "idCliente") String idCliente, @WebParam(name = "nombreCliente") String nombreCliente, @WebParam(name = "saldo") float saldo) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

s.executeUpdate("INSERT INTO clientes (idCliente, nombreCliente, saldo) VALUES ('" + idCliente + "', '" + nombreCliente + "', " + saldo + ")");

s.close();

con.close();

flag = true;

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "altaProducto")

public Boolean altaProducto(@WebParam(name = "ISBN") String ISBN, @WebParam(name = "nombrelibro") String nombreLibro, @WebParam(name = "saldo") int unidades) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

s.executeUpdate("INSERT INTO productos (ISBN, nombreLibro, unidades) VALUES ('" + ISBN + "', '" + nombreLibro + "', " + unidades + ")");

s.close();

con.close();

flag = true;

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "modificarCliente")

public Boolean modificarCliente(@WebParam(name = "idCliente") String idCliente, @WebParam(name = "saldo") float saldo) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select saldo from clientes where idCliente = '" + idCliente + "'");

if (rs.next()) {

s.executeUpdate("update clientes set saldo = " + saldo + " where idcliente = '" + idCliente + "'");

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

@WebMethod(operationName = "modificarProducto")

public Boolean modificarProducto(@WebParam(name = "ISBN") String ISBN, @WebParam(name = "saldo") int unidades) {

boolean flag = false;

try {

Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/serviciosVenta", "app", "app");

Statement s = con.createStatement();

ResultSet rs = s.executeQuery("select ISBN from productos where ISBN = '" + ISBN + "'");

if (rs.next()) {

s.executeUpdate("update productos set unidades = " + unidades + " where ISBN = '" + ISBN + "'");

flag = true;

}

rs.close();

s.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

return flag;

}

}

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

### 

public class Cliente {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

long lngQuienSoy;

long sumDeltaT, sumDeltaT2, dtMin = 0, dtMax = 0;

long lngCuantosMilisFaltan;

String host = (args.length < 1) ? null : args[0];

long i, n, t0, t1, dt;

String response;

/\*

Este parámetro recibe la cantidad de solicitudes deseadas; a pesar de que si el

programa se corrre desde el estresador nunca sucede que args[1] == null

debido a como esta creado el estresador, se deja por si se decide ejecutar

fuera del estresador

\*/

if (args[1] != null) {

n = Integer.parseInt(args[1]);

} else {

n = 200;

}

try {

Registry registry = LocateRegistry.getRegistry(host);

IServDisparo servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

lngQuienSoy = servDisparo.quienSoy();

lngCuantosMilisFaltan = servDisparo.deltaTEnMilis();

System.out.println("Cliente " + lngQuienSoy + " faltan " + lngCuantosMilisFaltan + " milisegundos");

sumDeltaT = 0;

sumDeltaT2 = 0;

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service = new org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService();

int ISBNcont1 = 7;

int ISBNcont2 = 0;

String tempISBN = "";

for (i = 0; i < n; i++) {

t0 = System.currentTimeMillis();

response = ventaLibrosWSDLOperation("978-0-00-000000-2", "6", 10, 2, service);

// response = newClient(Long.toString(i), "nombreCliente", 20, service);

// response = modifyClient(Long.toString(i), 10, service);

if (ISBNcont2 < 10) {

tempISBN = "00000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100) {

tempISBN = "0000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 1000) {

tempISBN = "000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 10000) {

tempISBN = "00" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100000) {

tempISBN = "0" + Integer.toString(ISBNcont2);

} else {

tempISBN = Integer.toString(ISBNcont2);

}

// response = newProduct("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), "nombreLibro", 20, service);

// response = modifyProduct("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), 10, service);

if (ISBNcont1 < 9) {

ISBNcont1++;

} else {

ISBNcont1 = 0;

ISBNcont2++;

}

t1 = System.currentTimeMillis();

dt = t1 - t0;

sumDeltaT += dt;

sumDeltaT2 += dt \* dt;

if (i == 0) {

dtMin = dt;

dtMax = dt;

} else {

if (dt < dtMin) {

dtMin = dt;

}

if (dt > dtMax) {

dtMax = dt;

}

}

System.out.println("Clte " + lngQuienSoy + ": " + response);

}

servDisparo.acumula(sumDeltaT, sumDeltaT2, n, dtMax, dtMin);

} catch (Exception e) {

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

private static String ventaLibrosWSDLOperation(java.lang.String isbn, java.lang.String idCliente, int unidades, int precioUnidad, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.ventaLibrosWSDLOperation(isbn, idCliente, unidades, precioUnidad);

}

private static String newClient(java.lang.String idCliente, java.lang.String nombreCliente, float saldo, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.newClient(idCliente, nombreCliente, saldo);

}

private static String newProduct(java.lang.String isbn, java.lang.String nombreLibro, int unidades, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.newProduct(isbn, nombreLibro, unidades);

}

private static String modifyClient(java.lang.String idCliente, float saldo, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.modifyClient(idCliente, saldo);

}

private static String modifyProduct(java.lang.String isbn, int unidades, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.modifyProduct(isbn, unidades);

}

}

public class Distribuidor

{

public static void main( String args[])

{

System.out.println("Distribuidor, inciando con:");

if( args.length == 0)

System.out.println("Sin argumentos");

else

for(int i = 0; i < args.length; i++)

System.out.println("args[" + i + "]:" + args[i]);

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

String[] argsPar = new String[0];

if( args.length > 1 )

{

argsPar = new String[args.length-1];

for(int i=1;i<args.length;i++)

argsPar[i-1] = args[i];

}

if( args.length == 0 )

{

System.out.println("uso:\n" +

"Distribuidor ServidorDeDisparo ip\_host\_localhost\n" +

"Distribuidor Master ip\_host\_localhost reset deltaTEnSegs\n" +

"Distribuidor Master ip\_host\_localhost [reporta]\n" +

"Distribuidor Cliente ip\_host\_localhost");

}

else

/\*

{

if( args[0].compareToIgnoreCase("Master") == 0) Master.main( argsPar );

if( args[0].compareToIgnoreCase("Cliente") == 0) Cliente.main( argsPar );

if( args[0].compareToIgnoreCase("Server") == 0) Server.main( argsPar );

if( args[0].compareToIgnoreCase("ServidorDeDisparo") == 0) ServidorDeDisparo.main( argsPar );

}

\*/

{

Class[] cArg = new Class[1];

cArg[0] = String[].class;

System.out.println("Distribuidor redirigiendo a args[0]:" + args[0]);

try

{

//Class cl = Class.forName("example.hello." + args[0]);

Class cl = Class.forName(args[0]);

Method m = cl.getMethod("main", cArg);

m.invoke(cl, (Object) argsPar);

System.out.println("Distribuidor...");

}

catch (Exception ex)

{

ex.printStackTrace();

}

}

}

}

public interface IServDisparo extends Remote

{

public long quienSoy( ) throws RemoteException;

public long deltaTEnMilis() throws RemoteException;

public void acumula( long sdt, long sdt2, long n, long dtMax, long dtMin ) throws RemoteException;

public void reset( long dentroDeCuantosMilis )throws RemoteException;

public double deltaTmedia() throws RemoteException;

public double deltaTStdDev() throws RemoteException;

public long deltaTMax() throws RemoteException;

public long deltTMin() throws RemoteException;

public long cuantosCltes() throws RemoteException;

}

public class Master

{

static IServDisparo servDisparo = null;

static void resetea( String strCuantosSegs )

{

long lngDentroDeCuantosMilis;

System.out.println("Se resetea el servidor de disparo con el argumento " + strCuantosSegs );

try

{

lngDentroDeCuantosMilis = 1000 \* Long.parseLong(strCuantosSegs);

if( lngDentroDeCuantosMilis <= 0 ) lngDentroDeCuantosMilis = 10000;

}

catch( Exception e )

{

lngDentroDeCuantosMilis = 10000;

}

try

{

servDisparo.reset( lngDentroDeCuantosMilis );

System.out.println("Se ha reseteado el servidor de disparo a " + lngDentroDeCuantosMilis + " milisegundos");

}

catch(Exception re )

{

re.printStackTrace();

}

}

static void reporta()

{

try

{

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

System.out.println(" cantidad de clientes:" + servDisparo.cuantosCltes());

System.out.println("media de tiempo de ciclo:" + servDisparo.deltaTmedia());

System.out.println(" dev std tiempo de ciclo:" + servDisparo.deltaTStdDev());

System.out.println(" tiempo de ciclo máximo:" + servDisparo.deltaTMax());

System.out.println(" tiempo de ciclo mínimo:" + servDisparo.deltTMin());

}

catch( Exception e )

{

e.printStackTrace();

}

}

public static void main(String[] args)

{

String host = (args.length < 1) ? null : args[0];

System.out.println("Master, inciando con:");

if( args.length == 0)

System.out.println("Sin argumentos");

else

for(int i = 0; i < args.length; i++)

System.out.println("args[" + i + "]:" + args[i]);

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

try

{

if( servDisparo == null )

{

Registry registry = LocateRegistry.getRegistry(host);

servDisparo = (IServDisparo) registry.lookup("ServidorDeDisparo");

}

if( args.length > 1 )

{

if ( args[1].compareToIgnoreCase("RESET") == 0 )

Master.resetea( args.length > 2 ? args[2] : "10" );

else

Master.reporta();

}

else

Master.resetea( "10" );

}

catch (Exception e)

{

System.err.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

}

public class Server implements Hello

{

long cuantos = 0;

String nomMaq = System.getenv("COMPUTERNAME");

public Server() {}

public String sayHello()

{

cuantos++;

System.out.println("Proporcionando el servicio no. " + cuantos);

return "Este es el servicio no. " + cuantos + " desde " + nomMaq;

}

public static void main(String args[ ])

{

String host = (args.length < 1) ? null : args[0];

System.out.println("Server, inciando con:");

if( args.length == 0)

System.out.println("Sin argumentos");

else

for(int i = 0; i < args.length; i++)

System.out.println("args[" + i + "]:" + args[i]);

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

try

{

Server obj = new Server();

Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

// Bind the remote object's stub in the registry

Registry registry = LocateRegistry.getRegistry(host);

registry.rebind("Hello", stub);

System.err.println("Listo el servicio...");

}

catch (Exception e)

{

System.err.println("Server exception: " + e.toString());

e.printStackTrace();

}

}

}

public class ServidorDeDisparo implements IServDisparo

{

long cuantosCltes;

long aQueHora = 0;

long cuantosCltesHanReportado = 0;

long lngDeltaTMax, lngDeltaTMin;

long lngNTot;

long lngSDeltaT, lngSDeltaT2;

boolean banCalc = false;

double dblDeltaTMedia, dblDeltaTDesvStd;

@Override

public synchronized long quienSoy( )

{

return ++cuantosCltes;

}

@Override

public synchronized long deltaTEnMilis() throws RemoteException

{

return aQueHora - System.currentTimeMillis();

}

@Override

public synchronized void acumula( long sdt, long sdt2, long n, long dtMax, long dtMin ) throws RemoteException

{

cuantosCltesHanReportado++;

lngSDeltaT += sdt;

lngSDeltaT2 += sdt2;

if( lngNTot > 0 ) // ya trae algo acumulado

{

if( dtMax > lngDeltaTMax ) lngDeltaTMax = dtMax;

if( dtMin < lngDeltaTMin ) lngDeltaTMin = dtMin;

}

else // primera vez de esta tanda de acumulación

{

lngDeltaTMax = dtMax;

lngDeltaTMin = dtMin;

}

lngNTot += n;

}

@Override

public synchronized void reset( long dentroDeCuantosMilis )throws RemoteException

{

aQueHora = System.currentTimeMillis() + dentroDeCuantosMilis;

cuantosCltes = 0;

cuantosCltesHanReportado = 0;

lngDeltaTMax = 0;

lngDeltaTMin = 0;

lngNTot = 0;

lngSDeltaT = 0;

lngSDeltaT2 = 0;

banCalc = false;

System.out.println("ServidorDeDisparo: la fiesta empieza en " + dentroDeCuantosMilis + " milisegs");

}

@Override

public synchronized double deltaTmedia() throws RemoteException

{

if( !banCalc )

{

obtenEstadisticas();

banCalc = true;

}

return dblDeltaTMedia;

}

@Override

public synchronized double deltaTStdDev() throws RemoteException

{

if( !banCalc )

{

obtenEstadisticas();

banCalc = true;

}

return dblDeltaTDesvStd;

}

@Override

public synchronized long deltaTMax() throws RemoteException

{

return lngDeltaTMax;

}

@Override

public synchronized long deltTMin() throws RemoteException

{

return lngDeltaTMin;

}

private synchronized void obtenEstadisticas()

{

dblDeltaTMedia = lngSDeltaT / lngNTot;

dblDeltaTDesvStd = (1.0/(lngNTot-1)) \* ( lngSDeltaT2 - lngNTot \* dblDeltaTMedia \* dblDeltaTMedia );

}

@Override

public long cuantosCltes() throws RemoteException

{

return cuantosCltesHanReportado;

}

// ======================= Método principal =======================

public static void main(String args[ ])

{

String host = (args.length < 1) ? null : args[0];

System.out.println("ServidorDeDisparo, inciando con:");

if( args.length == 0)

System.out.println("Sin argumentos");

else

for(int i = 0; i < args.length; i++)

System.out.println("args[" + i + "]:" + args[i]);

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

try

{

ServidorDeDisparo obj = new ServidorDeDisparo();

IServDisparo stub = (IServDisparo) UnicastRemoteObject.exportObject(obj, 0);

// Bind the remote object's stub in the registry

Registry registry = LocateRegistry.getRegistry(host);

registry.rebind("ServidorDeDisparo", stub);

System.err.println("Servidor de Disparo listo...");

}

catch (Exception e)

{

System.err.println("Server exception: " + e.toString());

e.printStackTrace();

}

}

}

public class TstGeneral {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service = new org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService();

int n = args.length;

if (n > 0) {

n = Integer.parseInt(args[0]);

} else {

n = 17;

}

for (int i = 0; i < n; i++) {

System.out.println(ventaLibrosWSDLOperation("978-0-00-000000-2", "6", 10, 2, service));

}

int ISBNcont1 = 7;

int ISBNcont2 = 0;

String tempISBN = "";

for (int i = 7; i < n; i++) {

System.out.println(newClient(Integer.toString(i), "nombreCliente", 20, service));

System.out.println(modifyClient(Integer.toString(i), 10, service));

if (ISBNcont2 < 10) {

tempISBN = "00000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100) {

tempISBN = "0000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 1000) {

tempISBN = "000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 10000) {

tempISBN = "00" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100000) {

tempISBN = "0" + Integer.toString(ISBNcont2);

} else {

tempISBN = Integer.toString(ISBNcont2);

}

System.out.println(newProduct("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), "nombreLibro", 20, service));

System.out.println(modifyProduct("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), 10, service));

if (ISBNcont1 < 9) {

ISBNcont1++;

} else {

ISBNcont1 = 0;

ISBNcont2++;

}

}

}

private static String ventaLibrosWSDLOperation(java.lang.String isbn, java.lang.String idCliente, int unidades, int precioUnidad, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.ventaLibrosWSDLOperation(isbn, idCliente, unidades, precioUnidad);

}

private static String newClient(java.lang.String idCliente, java.lang.String nombreCliente, float saldo, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.newClient(idCliente, nombreCliente, saldo);

}

private static String newProduct(java.lang.String isbn, java.lang.String nombreLibro, int unidades, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.newProduct(isbn, nombreLibro, unidades);

}

private static String modifyClient(java.lang.String idCliente, float saldo, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.modifyClient(idCliente, saldo);

}

private static String modifyProduct(java.lang.String isbn, int unidades, org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLService service) {

org.netbeans.j2ee.wsdl.serviciosventa.src.ventalibroswsdl.VentaLibrosWSDLPortType port = service.getVentaLibrosWSDLPort();

return port.modifyProduct(isbn, unidades);

}

}

public class TstMensajeriaExpress {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

String temp;

com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService service = new com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService();

int n = args.length;

if (n > 0) {

n = Integer.parseInt(args[0]);

} else {

n = 116;

}

for (int i = 6; i < n; i++) {

try {

System.out.println(enviarProducto("nombreEmpresa", Integer.toString(i), service));

} catch (Exception e) {

e.printStackTrace();

}

}

}

private static Integer enviarProducto(java.lang.String empresa, java.lang.String idPedido, com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetesService service){

com.adictosaltrabajo.webservice.mensajeria.EnvioPaquetes port = service.getEnvioPaquetesPort();

return port.enviarProducto(empresa, idPedido);

}

}

public class TstPagoOnline {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service = new com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService();

int n = args.length;

if (n > 0) {

n = Integer.parseInt(args[0]);

} else {

n = 100;

}

for (int i = 0; i < n; i++) {

System.out.println(actualizarInventario("6", 20, "978-0-00-000000-2", 10, service));

System.out.println(facturar("6", 20, service));

}

}

private static String facturar(java.lang.String idCliente, int cuantia, com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service) {

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPago port = service.getPasarelaPagoPort();

return port.facturar(idCliente, cuantia);

}

private static boolean actualizarInventario(java.lang.String idCliente, int cuantia, java.lang.String isbn, int unidades, com.adictosaltrabajo.webservice.pasarelapago.PasarelaPagoService service) {

com.adictosaltrabajo.webservice.pasarelapago.PasarelaPago port = service.getPasarelaPagoPort();

return port.actualizarInventario(idCliente, cuantia, isbn, unidades);

}

}

public class TstTiendaElectronica {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

com.adictosaltrabajo.webservice.almacen.AlmacenService service = new com.adictosaltrabajo.webservice.almacen.AlmacenService();

int n = args.length;

int ISBNcont1 = 7;

int ISBNcont2 = 0;

String tempISBN = "";

if (n > 0) {

n = Integer.parseInt(args[0]);

} else {

n = 17;

}

for (int i = 7; i < n; i++) {

System.out.println(comprobarISBN("978-0-00-000000-0", service));

System.out.println(comprobarStock("978-0-00-000000-0", 5, service));

System.out.println(comprobarIdCliente("1", service));

System.out.println(comprobarSaldo(5, 10, "1", service));

System.out.println(altaCliente(Integer.toString(i), "nombreCliente", 20, service));

System.out.println(modificarCliente(Integer.toString(i), 10, service));

if (ISBNcont2 < 10) {

tempISBN = "00000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100) {

tempISBN = "0000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 1000) {

tempISBN = "000" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 10000) {

tempISBN = "00" + Integer.toString(ISBNcont2);

} else if (ISBNcont2 < 100000) {

tempISBN = "0" + Integer.toString(ISBNcont2);

}else {

tempISBN = Integer.toString(ISBNcont2);

}

System.out.println(altaProducto("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), "nombreLibro", 20, service));

// System.out.println(modificarProducto("978-0-00-" + tempISBN + "-" + Integer.toString(ISBNcont1), 10, service));

if (ISBNcont1 < 9) {

ISBNcont1++;

} else {

ISBNcont1 = 0;

ISBNcont2++;

}

}

}

private static Boolean comprobarISBN(java.lang.String isbn, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarISBN(isbn);

}

private static Boolean comprobarStock(java.lang.String isbn, int unidades, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarStock(isbn, unidades);

}

private static Boolean comprobarIdCliente(String idCliente, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarIdCliente(idCliente);

}

private static Boolean comprobarSaldo(int unidades, int precioUnidad, String idCliente, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.comprobarSaldo(unidades, precioUnidad, idCliente);

}

private static Boolean altaCliente(java.lang.String idCliente, java.lang.String nombreCliente, float saldo, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.altaCliente(idCliente, nombreCliente, saldo);

}

private static Boolean altaProducto(java.lang.String isbn, java.lang.String nombrelibro, int unidades, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.altaProducto(isbn, nombrelibro, unidades);

}

private static Boolean modificarCliente(java.lang.String idCliente, float saldo, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.modificarCliente(idCliente, saldo);

}

private static Boolean modificarProducto(java.lang.String isbn, int unidades, com.adictosaltrabajo.webservice.almacen.AlmacenService service) {

com.adictosaltrabajo.webservice.almacen.Almacen port = service.getAlmacenPort();

return port.modificarProducto(isbn, unidades);

}

}

### 