

Manual de Instalación del sistema EcoMarket

Preparación de los entornos de ejecución de todos los nodos componentes del sistema

En esta parte explicaremos cómo habilitar todos los entornos para poder ejecutar. Para la ejecución necesitaremos descargar el proyecto desde el respectivo repositorio de Git.

Para instalar Git en CentOS 8:

```
$ sudo yum install git
```

Para instalar Git en Ubuntu 18:

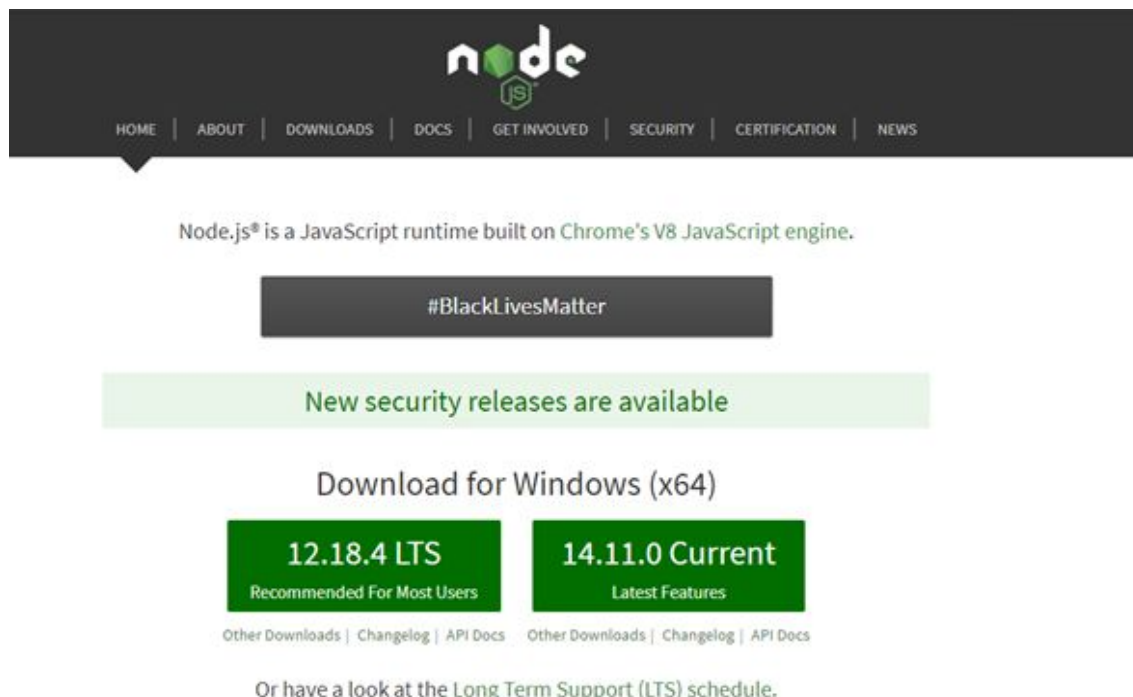
```
$ sudo apt install git
```

Ahora debemos descargar la carpeta del proyecto desde el repositorio de git:

```
$ sudo git clone  
https://github.com/wacarrear3128/Middleware.git
```

Instalación del entorno para el módulo de Procesamiento de Órdenes, con NodeJs en Windows 10

Instalar NodeJs y su npm (gestor de paquetes)



Una vez instalado el gestor de paquetes y NodeJs lo que debemos hacer es instalar todas las dependencias que están asociadas al módulo de Procesamiento de Órdenes.

Ojo: Tener en cuenta la ruta donde está ubicado el módulo de Procesamiento de Órdenes , en este caso la ruta es "C:\Users\pc\Documents\NodeJs\ProcesamientoOrdenes"

Ejecutar el comando "npm install".

```
C:\Users\pc\Documents\NodeJs\ProcesamientoOrdenes>npm install
```

Esto nos permitirá instalar todas las dependencias del módulo.

Para ejecutar el módulo ingresamos.

```
C:\Users\pc\Documents\NodeJs\ProcesamientoOrdenes>node index.js
```

El cual nos permitirá correr la aplicación y se habilitará el puerto 3000 en el local host del navegador.

```
PS C:\Users\pc\Documents\NodeJs\ProcesamientoOrdenes> node .\index.js  
Hola estoy corriendo en el puerto 3000
```

Instalación del entorno para los módulos de Inventario y Reserva, con Python en CentOS

Instalar Python 3.8 y pip, su gestor de paquetes

```
$ sudo yum -y install python3.8  
$ sudo yum -y install pip3
```

Actualizar pip e instalar ZeroMQ para Python

```
$ sudo pip install --upgrade pip  
$ sudo pip install zmq
```

Abrimos el puerto respectivo en el firewall de CentOS:

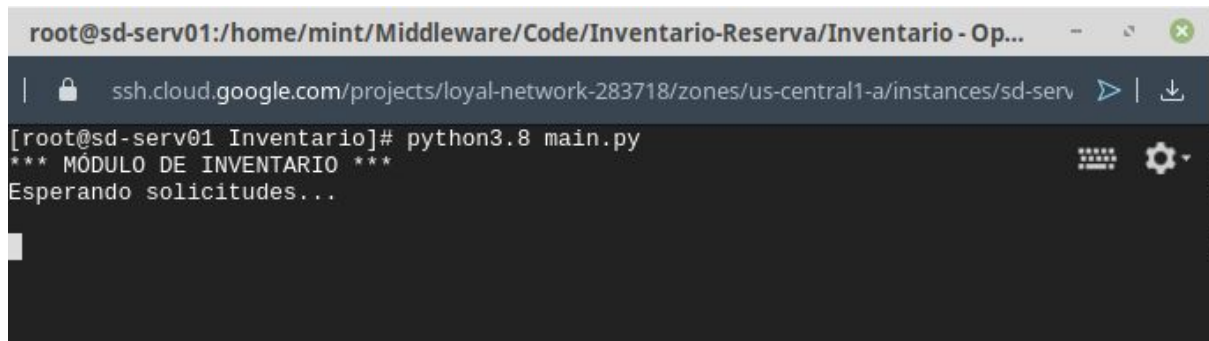
```
$ sudo firewall-cmd --permanent --zone=public  
--add-port=5050/tcp  
$ sudo firewall-cmd --permanent --zone=public  
--add-port=5052/tcp  
$ sudo firewall-cmd --permanent --zone=public  
--add-port=5053/tcp  
$ sudo firewall-cmd --reload
```

Para correr el módulo de Inventario, debemos posicionarnos en la carpeta respectiva del proyecto

```
$ cd Middleware/Code/Inventario-Reserva/Inventario/
```

Ejecutamos el fichero principal

```
$ sudo python3.8 main.py
```

A terminal window titled 'root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Inventario - Op...' is shown. The terminal displays the command '[root@sd-serv01 Inventario]# python3.8 main.py' and its output: '*** MÓDULO DE INVENTARIO ***' and 'Esperando solicitudes...'. The cursor is on a new line below the output.

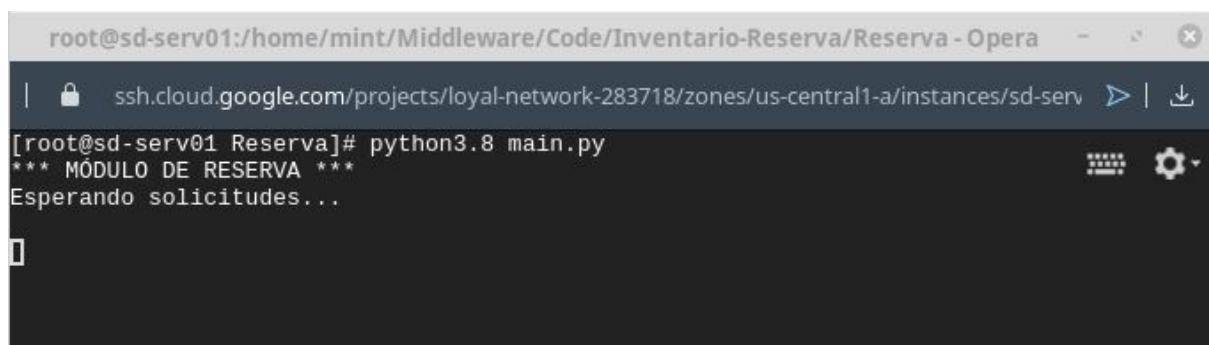
```
root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Inventario - Op...
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
[root@sd-serv01 Inventario]# python3.8 main.py
*** MÓDULO DE INVENTARIO ***
Esperando solicitudes...
```

Para correr el módulo de Reserva, debemos posicionarnos en la carpeta respectiva del proyecto

```
$ cd Middleware/Code/Inventario-Reserva/Reserva/
```

Ejecutamos el fichero principal

```
$ sudo python3.8 main.py
```

A terminal window titled 'root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Reserva - Opera' is shown. The terminal displays the command '[root@sd-serv01 Reserva]# python3.8 main.py' and its output: '*** MÓDULO DE RESERVA ***' and 'Esperando solicitudes...'. The cursor is on a new line below the output.

```
root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Reserva - Opera
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
[root@sd-serv01 Reserva]# python3.8 main.py
*** MÓDULO DE RESERVA ***
Esperando solicitudes...
```

Instalación del entorno para el módulo de Cuentas por cobrar, con PHP en Ubuntu

Instalar PHP 7.2 con sus paquetes de desarrollo y la librería de ZeroMQ para PHP

```
$ sudo apt-get install php php-dev
$ sudo apt-get install php-zmq
```

Abrimos el puerto respectivo en el firewall de Ubuntu:

```
$ sudo ufw allow 5051/tcp
$ sudo ufw reload
```

Para correr el módulo de Cuentas por cobrar, debemos posicionarnos en la carpeta respectiva del proyecto

```
$ cd Middleware/Code/Cuentas/
```

Ejecutamos el fichero principal

```
$ sudo php main.php
```

A screenshot of a terminal window titled 'root@sd-serv04: /home/mint/Middleware/Code/Cuentas - Opera'. The terminal shows the command 'php main.php' being executed. The output is '*** MÓDULO DE CUENTAS POR COBRAR ***' followed by 'Esperando solicitudes...' and a cursor. The terminal window has a dark background and a light-colored title bar.

Instalación del entorno para el módulo de Facturación, con .NET en Ubuntu

Descargamos los paquetes correspondientes de .NET para Linux

```
$ sudo wget
https://packages.microsoft.com/config/ubuntu/18.04/packages-m
icrosoft-prod.deb -O packages-microsoft-prod.deb
```

Instalamos la lista de paquetes descargada

```
$ sudo dpkg -i packages-microsoft-prod.deb
```

Actualizamos las versiones de los paquetes

```
$ sudo apt-get update; \
sudo apt-get install -y apt-transport-https && \
sudo apt-get update && \
sudo apt-get install -y dotnet-sdk-3.1
$ sudo apt-get update; \
sudo apt-get install -y apt-transport-https && \
sudo apt-get update && \
sudo apt-get install -y aspnetcore-runtime-3.1
```

Con los paquetes actualizados, instalamos .NET

```
$ sudo apt-get install -y dotnet-runtime-3.1
```

Abrimos el puerto respectivo en el firewall de Ubuntu:

```
$ sudo ufw allow 5053/tcp
$ sudo ufw reload
```

Para correr el módulo de Cuentas por cobrar, debemos posicionarnos en la carpeta respectiva del proyecto:

```
$ cd Middleware/Code/Facturacion/
```

La primera vez, tenemos que instalar el paquete de ZeroMQ para .NET:

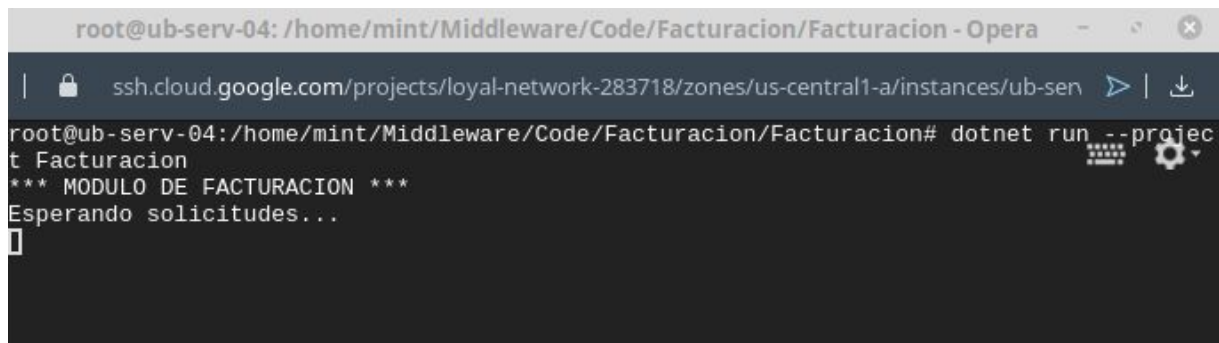
```
$ sudo dotnet add package ZeroMQ --version 4.1.0.31
```

Compilamos toda la solución:

```
$ sudo dotnet restore
$ sudo dotnet build Facturacion/Facturacion/
```

Ejecutamos el proyecto principal de la solución:

```
$ sudo dotnet run --project Facturacion/Facturacion/
```

A screenshot of a terminal window titled "root@ub-serv-04: /home/mint/Middleware/Code/Facturacion/Facturacion - Opera". The terminal shows the command "dotnet run --project Facturacion" being executed. The output displays "*** MODULO DE FACTURACION ***" and "Esperando solicitudes..." followed by a cursor. The terminal window has a dark background and standard Ubuntu window controls at the top.

Instalación del sistema gestor de base de datos en Ubuntu

Instalar el sistema de base de datos

```
$ sudo apt install mysql-server
```

Ahora, la configuración inicial del entorno de MySQL

```
$ sudo mysql_secure_installation
```

Levantamos los servicios de MySQL y nos aseguramos de abrir el puerto correspondiente en el firewall de Ubuntu.

```
$ sudo systemctl restart mysql
$ sudo ufw allow 3306/tcp
$ sudo ufw reload
```

En la configuración inicial, estableceremos la contraseña para el usuario root como "root123". También podríamos ejecutar lo siguiente en la consola de MySQL.

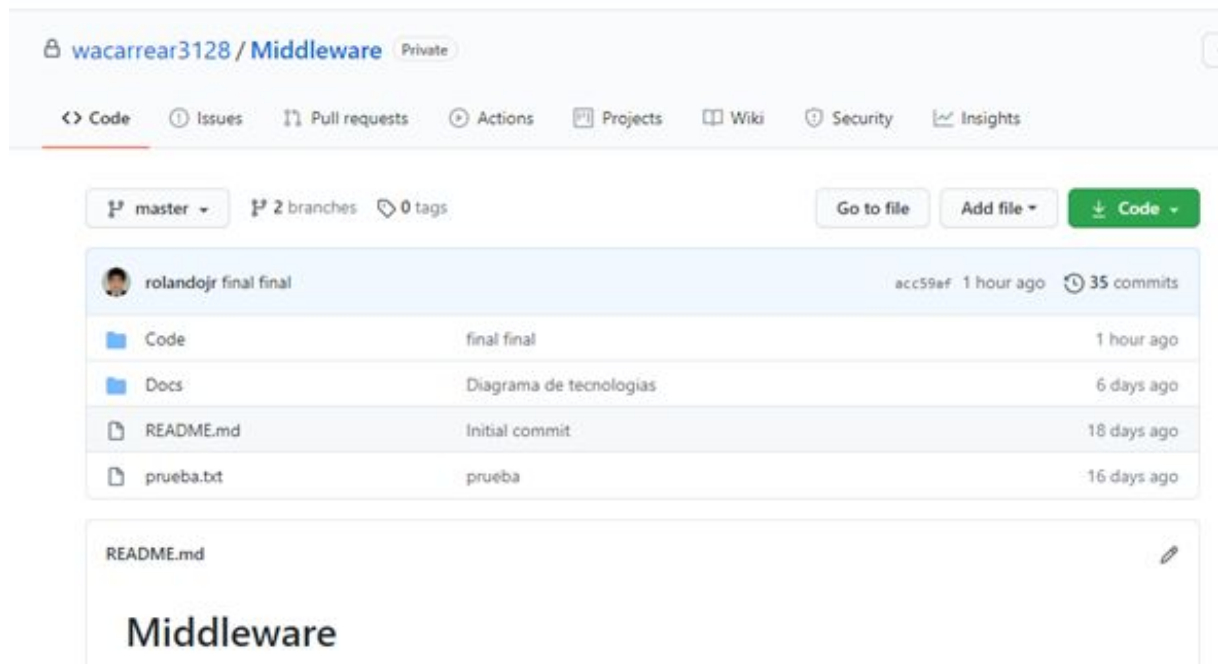
```
$ sudo mysql -u root -p
> UPDATE mysql.user SET Password PASSWORD('root123') WHERE
USER = 'root' AND Host = 'localhost';
> FLUSH PRIVILEGES;
> EXIT;
```

Para correr el script de base de datos, debemos posicionarnos en la carpeta respectiva del proyecto:

```
$ cd Middleware/Docs/
$ sudo mysql -u root -p sd_db < sd_db.sql
$ sudo mysql -u root -p sd_db < sd_db_ins.sql
```

Manual de Usuario: Orquestación de procesos de negocio

Para Iniciar el módulo de procesamiento de órdenes lo que debemos hacer es descargar la aplicación completa desde el repositorio Github <https://github.com/wacarrear3128/Middleware>.




Este módulo se encargará de realizar una solicitud de procesamiento de órdenes, para ser más preciso, va a generar un mensaje hacia el módulo de Inventario para verificar si realmente existe en el inventario la cantidad solicitada por el cliente.

En el módulo de procesamiento de órdenes se ha utilizado el framework NodeJS el cual tendrá instalado ZeroMQ, el cual le permite realizar la comunicación con el módulo de Inventario. La aplicación se desarrolló en una plataforma web.

Para comenzar el flujo, lo que el cliente debe hacer es ingresar su usuario y contraseña en la interfaz "Login", cual es la primera interfaz que se muestra al ejecutar la aplicación.

MiniMarket



En los campos Username y Password lo que nosotros debemos ingresar es el DNI de un cliente registrado en la base de datos.

A continuación, mostraremos la lista de DNIs de clientes que están en la base de datos.

- 12457823
- 23568945
- 14253685
- 14853652
- 95846231
- 95482615
- 62518475
- 23654815

Una vez ingresado alguno de esos DNIs lo que nos mostrará a continuación será la lista de productos que posee el Minimarket, para ser más preciso, la sección de frutas con sus respectivos precios.

Veremos un campo debajo de cada producto en el cual nosotros podremos ingresar el número de kilos de la fruta que deseamos comprar.

Selección de las mejores frutas del Perú

En este apartado encontrará la mejor selección de frutas del Perú.



Platano

Precio x Kg : \$2.0

La banana, plátano, guineo, banano, maduro, cambur o gualele, es un fruto comestible, botánicamente una baya, de varios tipos de grandes plantas herbáceas del género Musa.

Cantidad

Agregar



Chirimoya

Precio x Kg : \$3.5

Annona cherimola es un árbol perteneciente a la familia de las anonáceas cuyo fruto comestible es la chirimoya o chimoyo. Es considerada una de las frutas tropicales más apreciadas dentro del género.

Cantidad

Agregar



Uva

Precio x Kg : \$3.5

La uva es la común denominación que reciben los frutos formados en los racimos de la vid. Usada mundialmente para su fermentación, ya que ésta da lugar al vino. Crecidas en vides crecen agrupadas en las parras de las vides entre 6 y 300 uvas por racimo.

Cantidad

Agregar



Manzana

Precio x Kg : \$3.5

La manzana es el fruto comestible de la



Mango

Precio x Kg : \$3.5

El mango es el nombre de las frutas de



© Anna Kucharska / iStock

Lo que se realizará a continuación será agregar alguno de los productos al carrito de compras para eso debemos ingresar un número debajo de cada producto y pulsar sobre el botón “agregar”.

Se nos mostrará un mensaje de confirmación del producto agregado con la cantidad de kilos respectivo.



Platano

Precio x Kg : \$2.0

La banana, plátano, guineo, banano, maduro, cambur o gualele, es un fruto comestible, botánicamente una baya, de varios tipos de grandes plantas herbáceas del género Musa.

3

Agregar



Precio x Kg : \$3.5

La uva es la común denominación que reciben los frutos formados en los racimos de la vid. Usada mundialmente para su fermentación, ya que ésta da lugar al vino. Crecidas en vides crecen agrupadas en las parras de las vides entre 6 y 300 uvas por racimo.

1

Agregar



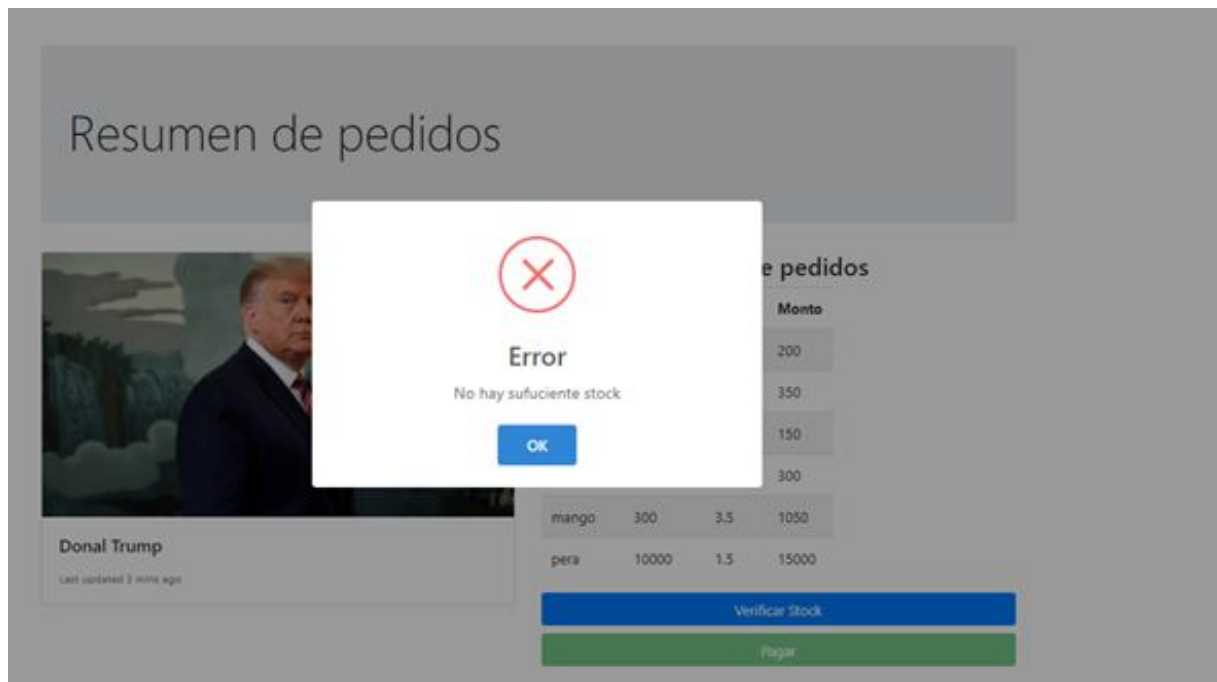
Una vez que hayamos terminado de agregar todos los productos que deseamos, debemos pulsar sobre el botón de cotización.



Este botón nos redirigirá hacia una nueva página la cual nos mostrará la lista de productos que hemos agregado en forma de tabla, esta tabla contendrá el nombre, cantidad, precio y total de cada producto.



Lo que haremos primero será verificar el Stock que posee el módulo de Inventario, si el stock es menor a la cantidad solicitada por el cliente nos retornará un mensaje de error.



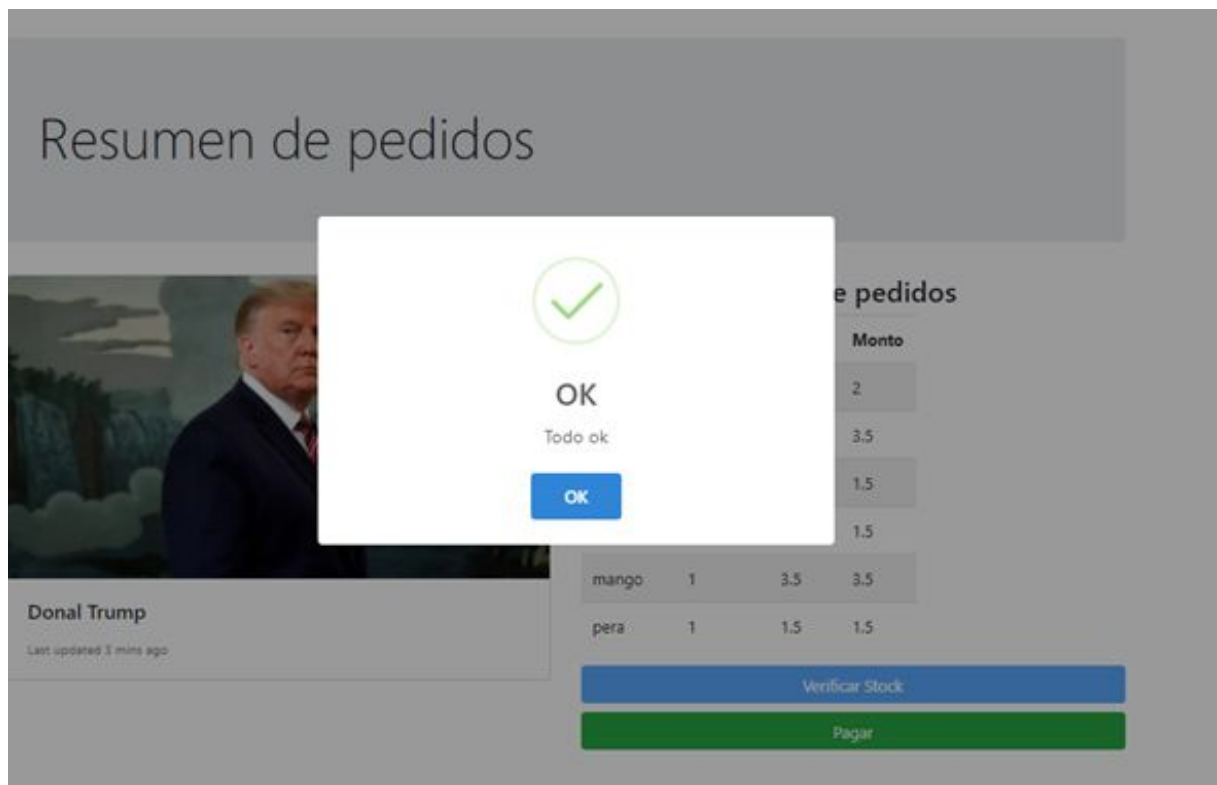
Del lado del módulo de Inventario, el log que se muestra es el siguiente:

```
root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Inventario - Op...
|  ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
Esperando solicitudes...
Getting bytes... Returning json (dict)...
> Solicitud Recibida
> Conexión a la bd: abierta.
Stock: 24.0 --> Cantidad: 2 --> DNI: 12457823
Suficiente: si
Stock: 25.0 --> Cantidad: 3 --> DNI: 12457823
Suficiente: si
Stock: 26.0 --> Cantidad: 4 --> DNI: 12457823
Suficiente: si
Stock: 25.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 26.0 --> Cantidad: 100000 --> DNI: 12457823
Suficiente: no
Stock: 27.0 --> Cantidad: 2 --> DNI: 12457823
Suficiente: si
Getting string... Returning json (dict)...
Esperando solicitudes...
```

Podemos ver que, de todos los productos que se desea reservar, solo uno no está cubierto por el stock. En este caso, la reserva no se realizará. Por el contrario, se devolverá al módulo de Procesamiento de órdenes un json indicando aquello.

El resultado es el mostrado, una alerta indicando que “No hay suficiente stock”.

En caso el módulo de inventario tenga la cantidad solicitada por el cliente entonces nos mostrará un mensaje exitoso, el cual nos permitirá habilitar la opción de pago, en la cual el usuario podrá realizar el pago.



Ahora, vemos el lado del módulo de Inventario. Observaremos que esta vez sí se realizó la reserva debido a que hay suficiente stock para cubrir todos los productos solicitados.

```
root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Inventario - Op...
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
[root@sd-serv01 Inventario]# python3.8 main.py
*** MÓDULO DE INVENTARIO ***
Esperando solicitudes...

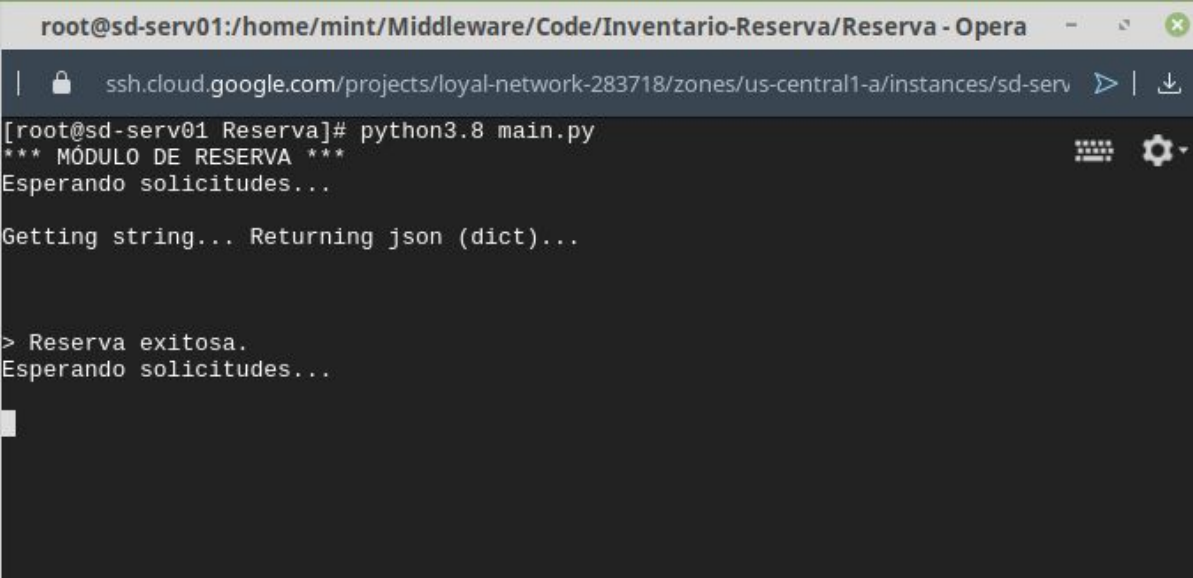
Getting bytes... Returning json (dict)...

> Solicitud Recibida
> Conexión a la bd: abierta.
Stock: 25.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 26.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 27.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 26.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 27.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Stock: 28.0 --> Cantidad: 1 --> DNI: 12457823
Suficiente: si
Getting string... Returning json (dict)...

> Enviando a Reserva
b'Solicitud reservada exitosamente.'

> Enviando a Facturacion
b'Recibido por Facturacion'
Esperando solicitudes...
```

Una vez verificado el stock, el flujo pasará al módulo de Reserva. Se le enviará un json conteniendo la información de los pedidos (idp y cantidad) que deberá separar en la base de datos. Podremos observar algo así:



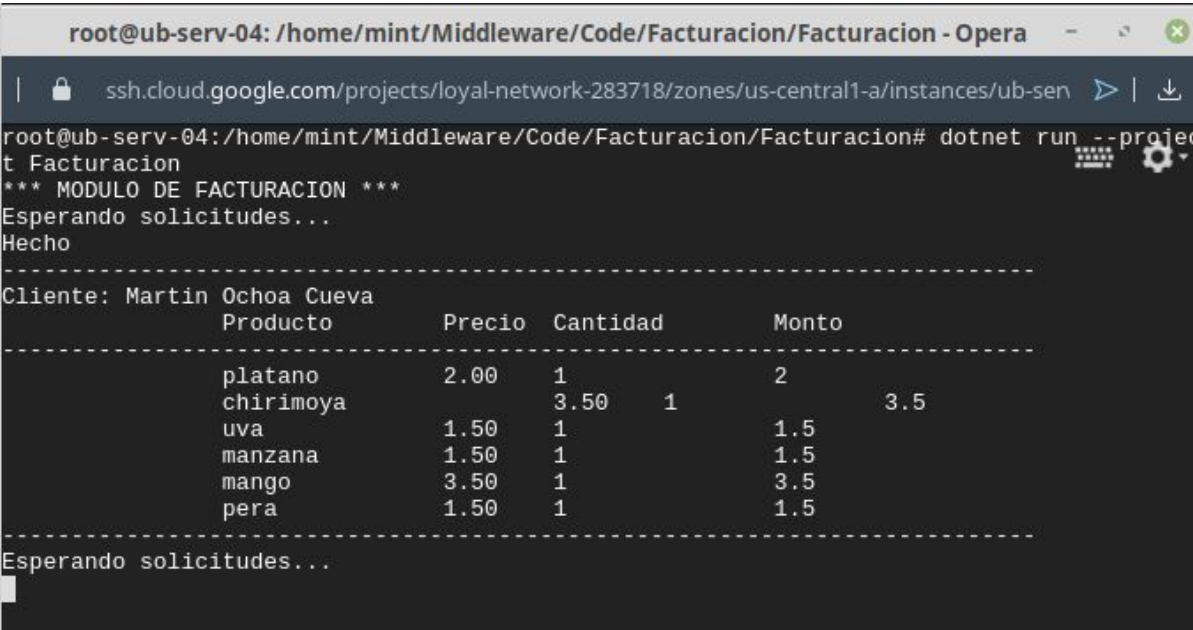
```
root@sd-serv01:/home/mint/Middleware/Code/Inventario-Reserva/Reserva - Opera
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
[root@sd-serv01 Reserva]# python3.8 main.py
*** MÓDULO DE RESERVA ***
Esperando solicitudes...

Getting string... Returning json (dict)...

> Reserva exitosa.
Esperando solicitudes...
```

Esto indica que la reserva se recibió exitosamente, así como su ejecución. Se le devolverá un mensaje de éxito al módulo de Inventario “Solicitud reservada exitosamente” (que se puede ver en la imagen del módulo de Inventario).

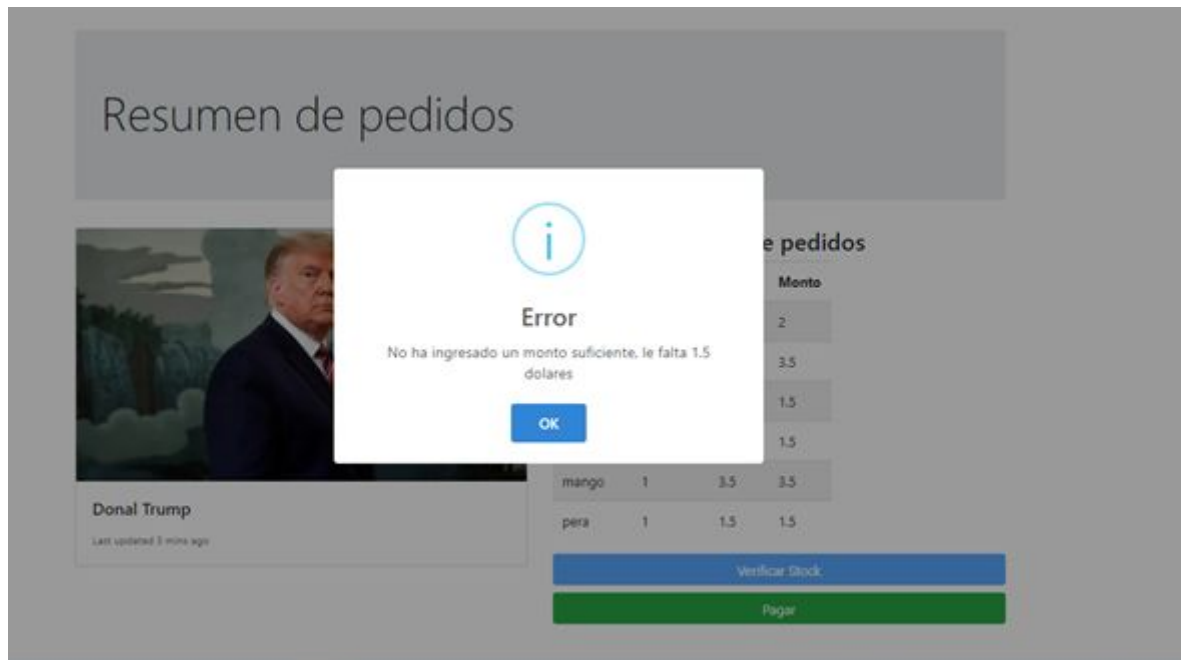
Lo siguiente será el envío del json con toda la información de Inventario y Reserva al módulo de Facturación, que mostrará un pequeño reporte con una factura con los datos del cliente, productos y sus respectivos precios. En la siguiente imagen se muestra el reporte del módulo de Facturación.



```
root@ub-serv-04: /home/mint/Middleware/Code/Facturacion/Facturacion - Opera
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/ub-sen
root@ub-serv-04:/home/mint/Middleware/Code/Facturacion/Facturacion# dotnet run --projec
t Facturacion
*** MODULO DE FACTURACION ***
Esperando solicitudes...
Hecho

-----
Cliente: Martin Ochoa Cueva
Producto      Precio  Cantidad  Monto
-----
    platano      2.00      1          2
    chirimoya    3.50      1          3.5
    uva          1.50      1          1.5
    manzana     1.50      1          1.5
    mango       3.50      1          3.5
    pera        1.50      1          1.5
-----
Esperando solicitudes...
```


Si el usuario no realiza el pago completo, entonces nos mostrará un mensaje en la cual se le informa que debe realizar el pago y se le incitara a volver a ingresar un nuevo monto, en caso de que el pago corresponda al monto solicitado se le mostrará un mensaje de “compra exitosa”.



En el módulo de Cuentas por cobrar, podremos ver un log que nos muestra la información detallada que se intercambian los módulos de Procesamiento de órdenes y Cuentas por cobrar. La imagen se muestra a continuación:

```
root@sd-serv04: /home/mint/Middleware/Code/Cuentas - Opera
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
root@sd-serv04:/home/mint/Middleware/Code/Cuentas# php main.php
*** MÓDULO DE CUENTAS POR COBRAR ***
Esperando solicitudes...
Getting string... Returning json...
> Conexion a la bd: abierta.
Cliente: 12457823
>> Filas: 2
Cliente: Martín Ochoa Cueva - ID: 12457823
Cliente: Martín Ochoa Cueva - ID: 12457823
>> Filas: 1
Deuda: 13.5 - Estado: Sin Cancelar
double -> 13.5
Resultado: 1
Vuelto: -1.5
Getting json... Returning string...
Esperando solicitudes...
█
```

Lo resaltante de este log es que se devuelve un resultado etiquetado como “vuelto” que almacenará la diferencia entre lo que se ha pagado y lo que falta pagar. Es decir, si se paga

lo suficiente, será el vuelto a devolver al cliente; y si no se paga lo suficiente, será el monto que resta pagar. En la siguiente ventana se muestra este valor (1.5) como monto por pagar.

Además, se terminará de pagar la factura pendiente con un monto de 1.5.

Pagos

Total: 13.5

Ingresar Pago: 1.5

Por pagar: 1.5

Cerrar **Pagar**

Resumen de pedidos

Item	Cantidad	Unidad	Precio Unitario	Total
uva	1	kg	1.5	1.5
manzana	1	kg	1.5	1.5
mango	1	kg	3.5	3.5
pera	1	kg	1.5	1.5

Donal Trump
Last updated 3 mins ago

Finalmente, como el monto ingresado es suficiente para saldar la deuda, se muestra un mensaje diciendo que la compra se ha realizado con éxito.

Felicidades

La compra se realizo con exito . gracias!!!

OK

Resumen de pedidos

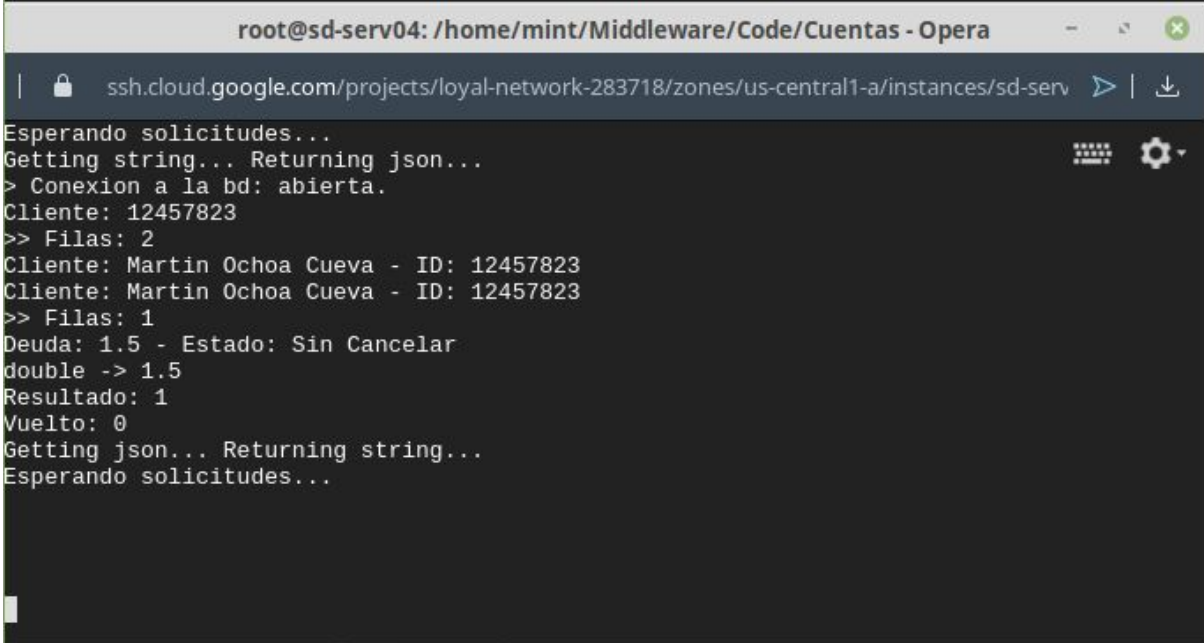
Item	Cantidad	Unidad	Precio Unitario	Total
uva	1	kg	1.5	1.5
manzana	1	kg	1.5	1.5
mango	1	kg	3.5	3.5
pera	1	kg	1.5	1.5

Donal Trump
Last updated 3 mins ago

Verificar Stock

Pagar

El resultado respectivo en el módulo de Cuentas por cobrar es el siguiente:



```
root@sd-serv04: /home/mint/Middleware/Code/Cuentas - Opera
ssh.cloud.google.com/projects/loyal-network-283718/zones/us-central1-a/instances/sd-serv
Esperando solicitudes...
Getting string... Returning json...
> Conexion a la bd: abierta.
Cliente: 12457823
>> Filas: 2
Cliente: Martin Ochoa Cueva - ID: 12457823
Cliente: Martin Ochoa Cueva - ID: 12457823
>> Filas: 1
Deuda: 1.5 - Estado: Sin Cancelar
double -> 1.5
Resultado: 1
Vuelto: 0
Getting json... Returning string...
Esperando solicitudes...
```

En este caso, notamos que el valor ingresado fue suficiente para saldar la deuda, por lo que el valor “vuelto” se vuelve 0.