

UNIVERSIDAD ORT URUGUAY

FACULTAD DE INGENIERÍA “ING. BERNARD WAND-POLAK”

EVALUACIÓN: OBLIGATORIO (SEGUNDA ENTREGA)

MATERIA: DISEÑO Y DESARROLLO DE APLICACIONES

PROFESOR: DARÍO CAMPALANS

FECHA DE ENTREGA: 11/12/17

GRUPO: M4A

INTEGRANTES:

DANIEL FRIEDMANN (144276)

GIANCARLO BELTRAMINI (195827)

CONTENIDO

Introducción y Autoevaluación	3
Diagrama Conceptual de Dominio	4
Diagrama de Clases	5
Diagrama de Clases de Persistencia.....	6
Usuarios Precargados (en memoria).....	7
Clientes Precargados (en base de datos)	7
Creación de la Base de Datos.....	8
Código – Cliente	10
Código – TipoCliente	12
Código – ClienteComun	13
Código – ClientePreferencial.....	14
Código – ClienteDeLaCasa.....	15
Código – Mozo	16
Código – Gestor.....	18
Código – Mesa.....	19
Código – Servicio.....	22
Código – Item.....	23
Código – Producto.....	25
Código – Transferencia	27
Código – UnidadProcesadora.....	29
Código – Usuario	31
Código – SistemaMozos	32
Código – SistemaGestores	34
Código – Sistema.....	36
Código – ModeloException	39
Código – LoginControladorGestor	40
Código – LoginVistaGestor	40
Código – LoginControladorMozo	41
Código – LoginVistaMozo.....	41
Código – MainControladorGestor.....	42
Código – MainVistaGestor	43
Código – MainControladorMozo	44
Código – MainVistaMozo	47

INTRODUCCIÓN Y AUTOEVALUACIÓN

DIAGRAMAS (DE CLASE Y DOMINIO)

Los diagramas fueron actualizados para dar cuenta de la nueva realidad del modelo. Agregando la entidad cliente, la interfaz para tipos y las clases que la implementan. Otro cambio a notar es la herencia de Thread en la clase Transferencia.

PRECARGA DE DATOS

Como lo solicita la letra, se agregan datos al iniciar el sistema, tanto de usuarios (gestores y mozos) como unidades procesadoras. Para obtener los accesos al sistema, ver *Usuarios precargados*.

El resto de los datos se obtienen de la base: los productos son puestos en memoria y se actualiza su stock, en cambio los clientes se buscan con cada solicitud.

INICIO DE LAS APLICACIONES

La aplicación de escritorio se inicializa una vez que el Servlet del login web es instanciado. A partir de ese momento ambas aplicaciones pueden ser utilizadas para interactuar con el mismo sistema. Nos vimos forzados a hacerlo de ésta manera, ya que todo corre dentro de un mismo equipo.

REGISTRO DE CLIENTES

Los casos de usos han sido cubiertos. Comentario: La letra indica que “si (los clientes) están registrados pueden llegar a tener algún beneficio al pagar el servicio”, por lo que se tomó en cuenta la posibilidad de que hayan clientes registrados sin un tipo definido, a los cuales no se les aplica ningún beneficio a pesar de estar registrado.

PERSISTENCIA

Los casos de usos han sido cubiertos. Todas las sentencias utilizadas están detalladas en el capítulo de persistencia.

LÍMITE DE TIEMPO EN TRANSFERENCIAS

El requerimiento fue cumplido modificando la clase Transferencia para que herede de Thread, y poder ejecutar en paralelo al resto de los procesos la cuenta regresiva.

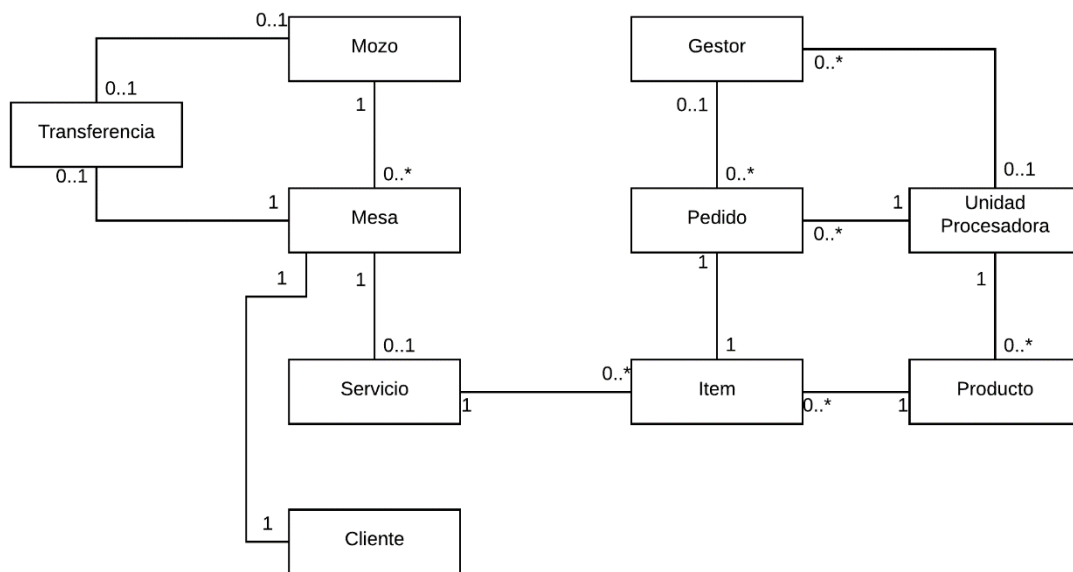
ACCESO WEB PARA LOS MOZOS

Los casos de usos han sido cubiertos, tanto los del primer obligatorio como las nuevas funcionalidades.

GENERAL

No notamos bugs, y las especificaciones han sido cubiertas.

DIAGRAMA CONCEPTUAL DE DOMINIO



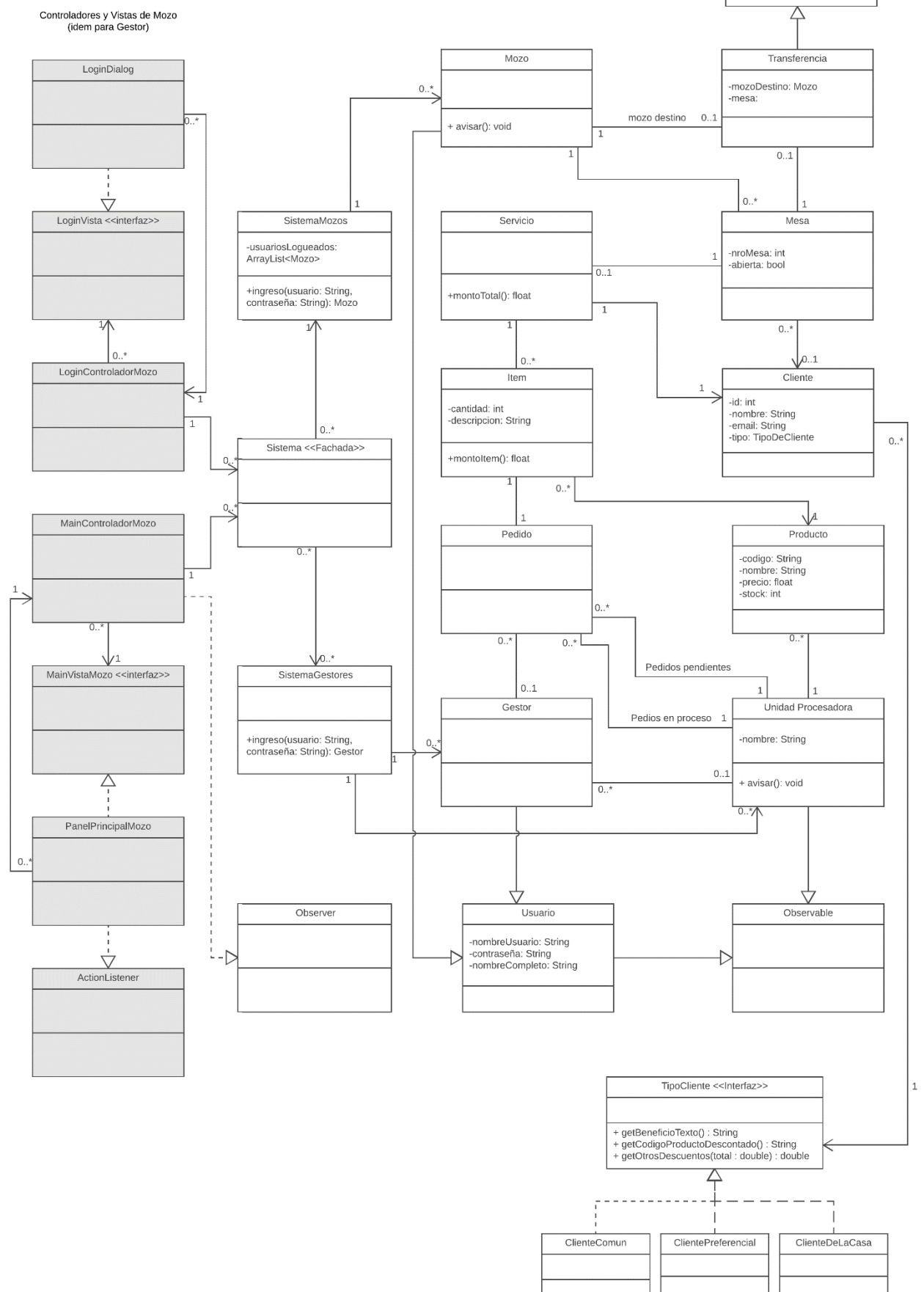
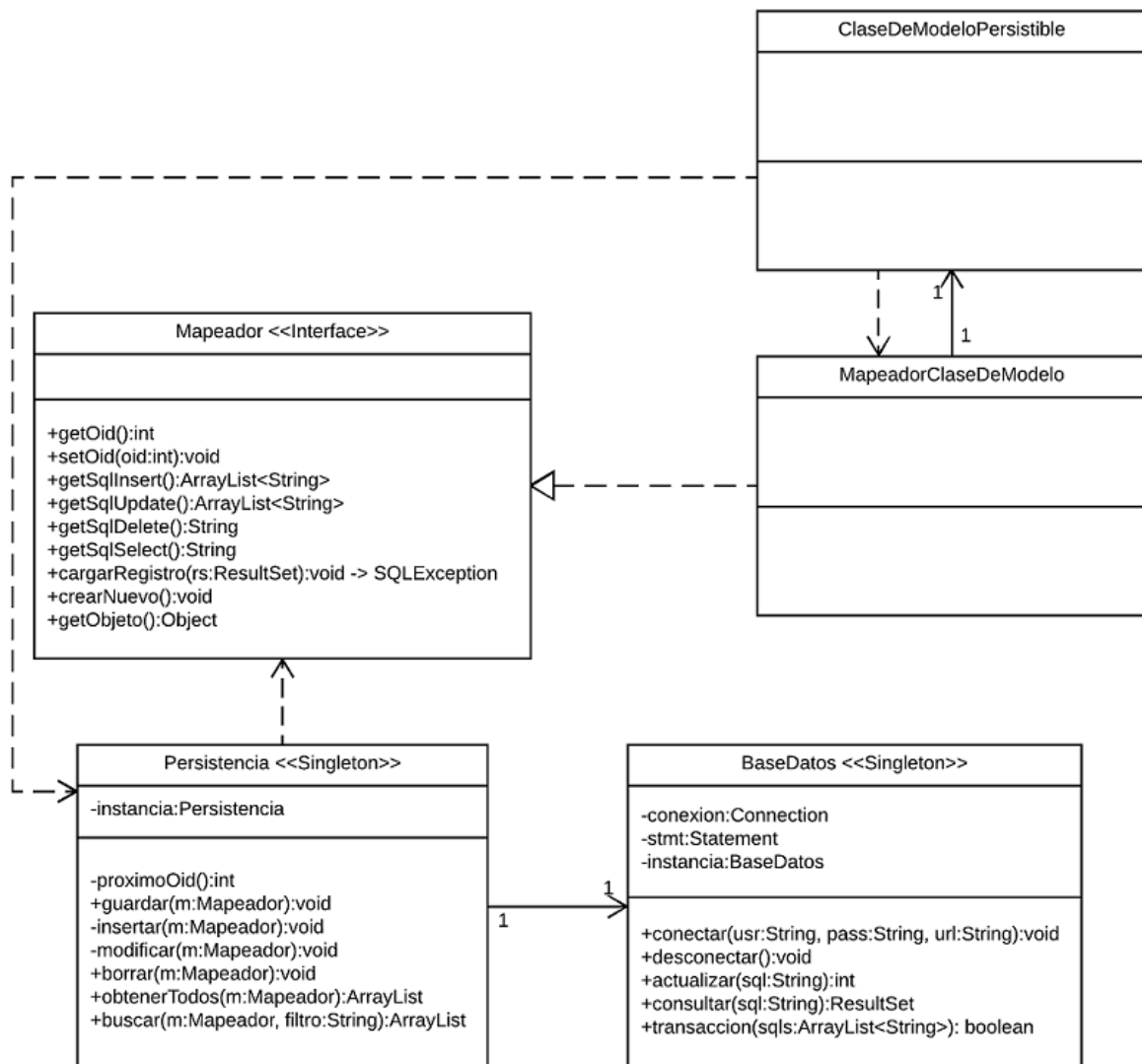
Controladores y Vistas de Mozo
(idem para Gestor)

DIAGRAMA DE CLASES DE PERSISTENCIA



USUARIOS PRECARGADOS (EN MEMORIA)

Mozo 1:

Nombre Completo: Jaime Talero

Usuario: a

Clave: a

Mozo 2:

Nombre Completo: Esteban Quito

Usuario: b

Clave: b

Mozo 3:

Nombre Completo: Aldo Lorigo

Usuario: c

Clave: c

Gestor 1:

Nombre Completo: Marta Tuada

Usuario: d

Clave: d

Gestor 2:

Nombre Completo: Guillermo Nigote

Usuario: e

Clave: e

Gestor 3:

Nombre Completo: Mary Quita

Usuario: f

Clave: f

CLIENTES PRECARGADOS (EN BASE DE DATOS)

Cliente 1:

Id: 11

Nombre: Donald Dump

Tipo: Sin Tipo

Cliente 2:

Id: 11

Nombre: Disect Ed

Tipo: Común

Cliente 3:

Id: 11

Nombre: Jay Decay

Tipo: Preferencial

Cliente 4:

Id: 11

Nombre: Tom Tom

Tipo: De la casa

CREACIÓN DE LA BASE DE DATOS

Nombre de la base de datos: ob_dda

Tablas:

Oid

```
CREATE TABLE `ob_dda`.`oid` (  
  `valor` INT NOT NULL,  
  PRIMARY KEY (`valor`));
```

//Sólo se guarda si contiene al menos un item

Servicios

```
CREATE TABLE `ob_dda`.`servicios` (  
  `oid` INT NOT NULL,  
  `nroMesa` INT NULL,  
  `montoAbonado` DOUBLE NULL,  
  `montoDeDescuento` DOUBLE NULL,  
  `cliente_id` INT(11) NULL,  
  PRIMARY KEY (`oid`));
```

Items

```
CREATE TABLE `ob_dda`.`items` (  
  `iditem` INT NOT NULL,  
  `servicioOid` INT(11) NOT NULL,  
  `cantidad` INT NULL,  
  `descripcion` VARCHAR(100) NULL,  
  `precioUnitario` DOUBLE NULL,  
  `usuarioGestor` VARCHAR(45) NULL,  
  `ProductoOid` INT NULL,  
  PRIMARY KEY (`iditem`, `servicioOid`));
```


Productos

```
CREATE TABLE `ob_dda`.`productos` (  
  `oid` INT NOT NULL,  
  `codigo` VARCHAR(45) NULL,  
  `nombre` VARCHAR(45) NULL,  
  `precioUnitario` DOUBLE NULL,  
  `stock` INT NULL,  
  `nombreUPP` VARCHAR(45) NULL,  
  PRIMARY KEY (`oid`));
```

//Valores validos de tipoClienteCod: COM, PRF, DLC

Clientes

```
CREATE TABLE `ob_dda`.`clientes` (  
  `oid` INT NOT NULL,  
  `idCliente` INT NULL,  
  `nombre` VARCHAR(45) NULL,  
  `email` VARCHAR(100) NULL,  
  `tipoClienteCod` CHAR(3) NULL,  
  PRIMARY KEY (`oid`));
```

Datos pre-insertados:

```
INSERT INTO productos(oid, codigo, nombre, precioUnitario, stock, nombreUPP) VALUES  
(101, "c001", "Pescado a la Plancha", 260, 15, "Cocina"),  
(102, "c002", "Milanesa napolitana", 360, 25, "Cocina"),  
(103, "c003", "Ensalada de la casa", 220, 18, "Cocina"),  
(104, "c004", "Empanadas varias (1un)", 50, 45, "Cocina"),  
(105, "c005", "Tallarines Lomein", 300, 40, "Cocina"),  
(106, "c006", "Arrollado de pollo", 350, 15, "Cocina"),  
(107, "b001", "Refresco 1lt", 120, 15, "Bar"),  
(108, "b002", "Agua Mineral 500ml", 60, 15, "Bar"),  
(109, "b003", "Cerveza 1lt", 180, 15, "Bar"),  
(110, "b004", "Café", 35, 65, "Bar")
```

```
INSERT INTO clientes(oid, idCliente, nombre, email) VALUES  
(111, 11, 'Donald Dump', 'iown@theworld.com');
```

```
INSERT INTO clientes(oid, idCliente, nombre, email, tipoClienteCod) VALUES  
(112, 12, 'Disect Ed', 'Disected@gpk.com', 'COM'),  
(113, 13, 'Jay Decay', 'jaydecay@dominiolargo.com', 'PRF'),  
(114, 14, 'Tom Tom', 'tom@tom.com', 'DLC');
```

Inicializo la tabla oid con el número 115, luego de haber puesto los productos y clientes:
INSERT INTO oid(value) VALUES (115)

CÓDIGO – CLIENTE

```
package modelo;

import java.util.ArrayList;
import modelo.Mapeador.MapeadorCliente;
import persistencia.Persistencia;

public class Cliente {

    private int oid;
    private int id;
    private String nombre;
    private String email;
    private TipoCliente tipo;

    public Cliente() {
    }

    public int getOid() {
        return oid;
    }

    public void setOid(int oid) {
        this.oid = oid;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public TipoCliente getTipo() {
        return tipo;
    }

    public void setTipo(TipoCliente tipo) {
        this.tipo = tipo;
    }

    public void setTipo(String str) {
        if (str == null) {

```

```

        this.tipo = null;
        return;
    }
    switch (str) {
        case "COM":
            this.tipo = new ClienteComun();
            break;
        case "PRF":
            this.tipo = new ClientePreferencial();
            break;
        case "DLC":
            this.tipo = new ClienteDeLaCasa();
            break;
        default:
            this.tipo = null;
            break;
    }
}

public static Cliente getClienteFromDB(int clienteId) {
    Persistencia p = Persistencia.getInstancia();
    MapeadorCliente mc = new MapeadorCliente();
    ArrayList arr = p.buscar(mc, "idCliente = " + clienteId);
    if (arr.size() > 0) {
        return (Cliente) arr.get(0);
    } else {
        return null;
    }
}

public String getBeneficioTexto() {
    if (tipo == null) {
        return "Sin Beneficio";
    }
    return tipo.getBeneficioTexto();
}

String getCodigoProductoDescontado() {
    if (tipo != null) {
        return tipo.getCodigoProductoDescontado();
    }
    return null;
}

double getOtrosDescuentos(double total) {
    if (tipo != null) {
        return tipo.getOtrosDescuentos(total);
    }
    return 0;
}
}

```

CÓDIGO – TIPOCLIENTE

```
package modelo;

import java.util.ArrayList;

public class ClienteComun implements TipoCliente {

    @Override
    public String getBeneficioTexto() {
        return "Café gratis";
    }

    //Codigo de café es b004
    @Override
    public String getCodigoProductoDescontado() {
        return "b004";
    }

    @Override
    public double getOtrosDescuentos(double total) {
        return 0;
    }
}
```

CÓDIGO – CLIENTE COMUN

```
package modelo;

import java.util.ArrayList;

public class ClienteComun implements TipoCliente {

    @Override
    public String getBeneficioTexto() {
        return "Café gratis";
    }

    //Codigo de café es b004
    @Override
    public String getCodigoProductoDescontado() {
        return "b004";
    }

    @Override
    public double getOtrosDescuentos(double total) {
        return 0;
    }
}
```

CÓDIGO – CLIENTEPREFERENCIAL

```
package modelo;

import java.util.ArrayList;

public class ClientePreferencial implements TipoCliente {

    @Override
    public String getBeneficioTexto() {
        return "Agua gratis y -5% (si total >$2000)";
    }

    //Codigo del agua mineral es b002
    @Override
    public String getCodigoProductoDescontado() {
        return "b002";
    }

    @Override
    public double getOtrosDescuentos(double total) {
        if (total >= 2000) {
            return total * 0.05;
        }
        return 0;
    }
}
```

CÓDIGO – CLIENTEDELACASA

```
package modelo;

import java.util.ArrayList;

public class ClienteDeLaCasa implements TipoCliente {

    @Override
    public String getBeneficioTexto() {
        return "$500 de consumo gratis";
    }

    @Override
    public String getCodigoProductoDescontado() {
        return null;
    }

    @Override
    public double getOtrosDescuentos(double total) {
        return 500;
    }
}
```

CÓDIGO – MOZO

```
package modelo;

import java.util.ArrayList;

public class Mozo extends Usuario {

    private ArrayList<Mesa> mesas;
    private Transferencia transferencia;

    // <editor-fold defaultstate="collapsed" desc="Gets, Sets y Agregar-
    Remover Mesas">

    public Mozo() {
        this.mesas = new ArrayList();
    }

    public ArrayList<Mesa> getMesas() {
        return mesas;
    }

    public void agregarMesa(Mesa mesa) {
        if (!this.mesas.contains(mesa)) {
            this.mesas.add(mesa);
            mesa.setMozo(this);
        }
    }

    public void removerMesa(Mesa mesa) {
        this.mesas.remove(mesa);
    }

    public Transferencia getTransferencia() {
        return transferencia;
    }

    public void setTransferencia(Transferencia transferencia) {
        this.transferencia = transferencia;
    }
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="Observable Section">
    public void avisar(eventos eventos) {
        setChanged();
        notifyObservers(eventos);
    }

    public void solicitarTransferencia(Transferencia transferencia) {
        setTransferencia(transferencia);
        avisar(eventos.transferenciaSolicitadaFrom);
        transferencia.notificar();
    }

    public boolean tiene(Mesa mesa) {
        return mesa != null && mesas.contains(mesa);
    }

    public enum eventos {
        mesa,
        transferenciaSolicitadaTo,
    }
}
```



```

        transferenciaSolicitadaFrom,
        transferenciaRechazada,
        transferenciaAceptada,
        transferenciaActualizada,
        transferenciaTiempoTerminado,
        pedido
    }
    // </editor-fold>

    public boolean tieneMesas() {
        return mesas != null && !mesas.isEmpty();
    }

    public boolean tieneMesasAbiertas() {
        if(tieneMesas()){
            for (Mesa m : mesas) {
                if (m.estaAbierta()) {
                    return true;
                }
            }
        }
        return false;
    }

    public void aceptarTransferencia() {
        if (transferencia != null) {
            transferencia.confirmar();
        }
    }

    public void rechazarTransferencia(){
        transferencia.transferenciaRechazada();
    }

    @Override
    public String toString() {
        return super.toString();
    }
}

```

CÓDIGO – GESTOR

```
package modelo;

import java.util.ArrayList;

public class Gestor extends Usuario {

    private UnidadProcesadora unidad;
    private ArrayList<Pedido> pedidos;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets">
    public UnidadProcesadora getUnidad() {
        return unidad;
    }

    public void setUnidad(UnidadProcesadora unidad) {
        this.unidad = unidad;
    }

    public ArrayList<Pedido> getPedidos() {
        return pedidos;
    }

    public Gestor() {
        this.pedidos = new ArrayList();
    }

    // </editor-fold>

    public void trabajarPedido(Pedido pedido) {
        this.pedidos.add(pedido);
        pedido.asignarGestor(this);
        this.unidad.procesarPedido(pedido);
    }

    public void finalizarPedido(Pedido pedido) {
        if(pedidos.contains(pedido)){
            this.pedidos.remove(pedido);
            this.unidad.finalizarPedido(pedido);
        }
    }

}
```

CÓDIGO – MESA

```
package modelo;

import java.util.ArrayList;

public class Mesa {

    private int nro;
    private boolean abierta;
    private Mozo mozo;
    private Servicio servicio;
    private Transferencia transferencia;
    private Cliente cliente;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets y Agregar-
    Remove Items">
    public int getNro(){
        return nro;
    }
    public void setNro(int nro) {
        this.nro = nro;
    }
    public boolean estaAbierta() {
        return abierta;
    }
    public void abrir(){
        abierta = true;
        servicio = new Servicio();
        servicio.setMesa(this);
    }
    public Mozo getMozo() {
        return mozo;
    }
    public void setMozo(Mozo mozo) {
        this.mozo = mozo;
    }
    public Transferencia getTransferencia(){
        return transferencia;
    }
    public void setTransferencia(Transferencia transferencia) {
        this.transferencia = transferencia;
    }
    public Servicio getServicio() {
        return servicio;
    }
    public void setServicio(Servicio servicio) {
        this.servicio = servicio;
    }
    public void removerItemAlServicio(Item item){
        servicio.removerItem(item);
    }
    public Cliente getCliente(){
        return this.cliente;
    }
    // </editor-fold>

    public void agregarItemAlServicio(Item item) {
        Producto producto = item.getProducto();
        int cantidad = item.getCantidad();
```

```

        if(estaAbierta() && producto != null && cantidad > 0){
            if(producto.hayStock(cantidad)){
                Pedido pedido = new Pedido();
                pedido.agregarItem(item);
                this.getMozo().avisar(Mozo.eventos.pedido);
            }
        }
    }

    public void cerrar() throws ModeloException {
        if(estaAbierta()){
            if(!servicio.hayPendientes()){
                this.servicio.guardarServicio();
                this.servicio = new Servicio();
                this.abierta = false;
            } else {
                throw new ModeloException("Hay items sin finalizar, por lo
que no se puede cerrar la mesa");
            }
        }
    }

    @Override
    public boolean equals(Object obj) {
        Mesa mesa = (Mesa)obj;
        return nro == mesa.getNro();
    }

    public void transferenciaRechazada(Mozo mozoDestino) {
        setTransferencia(null);
        mozo.setTransferencia(null);
        mozoDestino.setTransferencia(null);

        mozo.avisar(Mozo.eventos.transferenciaRechazada);
    }

    public void transferenciaAceptada(Mozo mozoDestino){
        Mozo mozoOrigen = this.mozo;
        mozoOrigen.setTransferencia(null);
        mozoDestino.setTransferencia(null);
        setTransferencia(null);

        mozoOrigen.removeMesa(this);
        mozoDestino.agregarMesa(this);

        mozoOrigen.avisar(Mozo.eventos.transferenciaAceptada);
    }

    @Override
    public String toString() {
        return "Mesa:" + nro + ", " + mozo;
    }

    public void avisarMozo() {
        mozo.avisar(Mozo.eventos.pedido);
    }

    public void agregarCliente(int idCliente) throws ModeloException {
        cliente = Cliente.getClienteFromDB(idCliente);
        if (cliente == null) throw new ModeloException("Cliente no
válido");
    }

```

```

    }

    public String getClienteNombre() {
        if(cliente == null){
            return "No hay cliente agregado";
        }
        return cliente.getNombre();
    }

    public String getClienteBeneficioTexto() {
        if (cliente == null){
            return "Sin Beneficio";
        }
        return cliente.getBeneficioTexto();
    }

    public String getCodigoProductoDescontado() {
        if(cliente != null){
            return cliente.getCodigoProductoDescontado();
        }
        return null;
    }

    double getOtrosDescuentos(double total) {
        if(cliente != null){
            return cliente.getOtrosDescuentos(total);
        }
        return 0;
    }
}

```

CÓDIGO—SERVICIO

```
package modelo;

import java.util.ArrayList;

public class Servicio {

    private ArrayList<Item> items;
    private Mesa mesa;

    // <editor-fold defaultstate="collapsed" desc="Agregar-Remover Items">
    public Servicio() {
        this.items = new ArrayList();
    }

    public Mesa getMesa() {
        return mesa;
    }

    public void setMesa(Mesa mesa) {
        this.mesa = mesa;
    }

    public ArrayList<Item> getItems() {
        return items;
    }

    public void agregarItem(Item item) throws ModeloException {
        item.validar();
        this.items.add(item);
        item.hacerPedido(this);
    }

    public void removerItem(Item item) {
        this.items.remove(item);
    }
    // </editor-fold>

    public double montoTotal() {
        double total = 0;
        for(Item i:items){
            total += i.getMonto();
        }
        return total;
    }

    public boolean hayPendientes() {
        for(Item i:items){
            if(i.getEstado().equals(Item.Estado.pendiente)){
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return "Servicio: " + mesa.toString();
    }

    public void actualizarMesa() {
        mesa.avisarMozo();
    }
}
```

CÓDIGO – ÍTEM

```
package modelo;

public class Item {

    private Producto producto;
    private int cantidad;
    private String descripcion;
    private double precioUnitario; //PARA CONGELAR EL PRECIO
    private Pedido pedido;
    private Estado estado;
    private Servicio servicio;

    public void validar() throws ModeloException {
        if(cantidad <= 0)
            throw new ModeloException("Cantidad invalida");
        if(producto == null)
            throw new ModeloException("Debe elegir al menos un producto");
        if(!producto.hayStock(cantidad))
            throw new ModeloException("Sin Stock");
    }

    public void actualizarServicio() {
        servicio.actualizarMesa();
    }

    public enum Estado {pendiente,enProceso,finalizado;}

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets">

    public Producto getProducto(){
        return producto;
    }
    public void setProducto(Producto producto) {
        this.producto = producto;
        this.precioUnitario = this.producto.getPrecioUnitario();
    }
    public int getCantidad(){
        return cantidad;
    }
    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }
    public String getDescripcion() {
        return descripcion;
    }
    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }
    public double getPrecioUnitario(){
        return precioUnitario;
    }
    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }
    public Estado getEstado(){
        return estado;
    }
    public void setEstado(Estado estado) {
        this.estado = estado;
    }
}
```

```

    }
    public Pedido getPedido() {
        return pedido;
    }
    public void setPedido(Pedido pedido) {
        this.pedido = pedido;
    }
    public Servicio getServicio() {
        return servicio;
    }
    public void setServicio(Servicio servicio) {
        this.servicio = servicio;
    }
    // </editor-fold>

    public double getMonto(){
        return precioUnitario*cantidad;
    }

    public void hacerPedido(Servicio servicio){
        setServicio(servicio);
        setPedido(new Pedido(this));
        producto.elaborarProducto(cantidad);
    }

    @Override
    public String toString() {
        return producto + " x" + cantidad;
    }
}

```


CÓDIGO – PRODUCTO

```
package modelo;

public class Producto {

    private String codigo;
    private String nombre;
    private double precioUnitario;
    private int stock;
    private UnidadProcesadora unidad;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets">
    public Producto() {
    }

    public Producto(String codigo, String nombre, double precioUnitario,
int stock, UnidadProcesadora unidad) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.precioUnitario = precioUnitario;
        this.stock = stock;
        setUnidad(unidad);
    }

    public String getCodigo() {
        return codigo;
    }
    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public double getPrecioUnitario() {
        return precioUnitario;
    }
    public void setPrecioUnitario(double precioUnitario) {
        this.precioUnitario = precioUnitario;
    }
    public int getStock() {
        return stock;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
    public UnidadProcesadora getUnidad() {
        return unidad;
    }
    public void setUnidad(UnidadProcesadora unidad) {
        this.unidad = unidad;
        this.unidad.agregarProducto(this);
    }
    // </editor-fold>

    public boolean hayStock(int cantidad){
        if(stock >= cantidad){
            return true;
        }
    }
}
```

```
        }else{
            return false;
        }
    }

    public void elaborarProducto(int cantidad){
        this.stock -= cantidad;
    }

    @Override
    public String toString() {
        return nombre;
    }

}
```

CÓDIGO—TRANSFERENCIA

```
package modelo;

public class Transferencia extends Thread {

    private Mesa mesa;
    private Mozo mozoDestino;
    private int tiempoMax = 20;
    private int tiempoRestante = tiempoMax;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets">
    public Mesa getMesa() {
        return mesa;
    }

    public void setMesa(Mesa mesa) {
        this.mesa = mesa;
        mesa.setTransferencia(this);
    }

    public Mozo getMozoDestino() {
        return mozoDestino;
    }

    public void setMozoDestino(Mozo mozo) {
        this.mozoDestino = mozo;
        mozoDestino.setTransferencia(this);
    }

    public int getTiempoMax() {
        return tiempoMax;
    }

    public void setTiempoMax(int tiempo) {
        this.tiempoMax = tiempoMax;
    }

    public int getTiempoRestante() {
        return tiempoRestante;
    }

    public void setTiempoRestante(int tiempoRestante) {
        this.tiempoRestante = tiempoRestante;
    }
    // </editor-fold>

    public void confirmar() {
        tiempoMax = -1;
        mesa.transferenciaAceptada(mozoDestino);
    }

    public boolean validar() {
        return mesa != null && mozoDestino != null;
    }

    public void notificar() {
        mozoDestino.avisar(Mozo.eventos.transferenciaSolicitadaTo);
        start();
    }
}
```

```

public void transferenciaRechazada() {
    tiempoMax = -1;
    mesa.transferenciaRechazada(mozoDestino);
}

@Override
public void run() {
    while (tiempoMax != -1 && tiempoRestante > 0) {
        System.out.println(tiempoRestante);
        try {
            sleep(1000);
            tiempoRestante--;
            mozoDestino.avisar(Mozo.eventos.transferenciaActualizada);
        } catch (InterruptedException ex) {
            System.out.println(ex);
        }
    }
    if (tiempoRestante == 0) {
        mozoDestino.avisar(Mozo.eventos.transferenciaTiempoTerminado);
        transferenciaRechazada();
    }
}
}

```

CÓDIGO – UNIDADPROCESADORA

```
package modelo;

import java.util.ArrayList;
import java.util.Observable;

public class UnidadProcesadora extends Observable{

    private String nombre;
    private ArrayList<Producto> productos;
    private ArrayList<Pedido> pedidosEnProceso;
    private ArrayList<Pedido> pedidosPendientes;

    // <editor-fold defaultstate="collapsed" desc="Sets, Gets, Agregar y
    Remove Productos-Pedidos">

    public UnidadProcesadora() {
        this.productos = new ArrayList();
        this.pedidosEnProceso = new ArrayList();
        this.pedidosPendientes = new ArrayList();
    }

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public ArrayList<Producto> getProductos() {
        return productos;
    }
    public void agregarProducto(Producto producto) {
        this.productos.add(producto);
    }
    public void removerProducto(Producto producto) {
        this.productos.remove(producto);
    }
    public ArrayList<Pedido> getPedidosEnProceso() {
        return pedidosEnProceso;
    }
    public void agregarPedidoEnProceso(Pedido pedido) {
        this.pedidosEnProceso.add(pedido);
    }
    public void removerPedidoEnProceso(Pedido pedido) {
        this.pedidosEnProceso.remove(pedido);
    }
    public ArrayList<Pedido> getPedidosPendientes() {
        return pedidosPendientes;
    }
    public void agregarPedidoPendiente(Pedido pedido) {
        this.pedidosPendientes.add(pedido);
        avisar(eventos.pedidos);
    }
    public void removerPedidoPendiente(Pedido pedido) {
        this.pedidosPendientes.remove(pedido);
    }
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="Observable Section">
    private void avisar(eventos eventos) {
```

```

        setChanged();
        notifyObservers(eventos);
    }

    public enum eventos {
        pedidos;
    }
    // </editor-fold>

    public boolean procesarPedido(Pedido pedido) {
        if (pedidosPendientes.contains(pedido)) {
            removerPedidoPendiente(pedido);
            pedido.procesar();
            agregarPedidoEnProceso(pedido);
            avisar(eventos.pedidos);
            return true;
        }
        return false;
    }

    public boolean finalizarPedido(Pedido pedido) {
        if (pedidosEnProceso.contains(pedido)) {
            removerPedidoEnProceso(pedido);
            pedido.finalizar();
            avisar(eventos.pedidos);
            return true;
        }
        return false;
    }

    public ArrayList<Producto> getProductosConStock() {
        ArrayList<Producto> lista = new ArrayList();
        if (productos != null) {
            for (Producto p : productos) {
                if (p.getStock() > 0) lista.add(p);
            }
        }
        return lista;
    }

    @Override
    public String toString() {
        return nombre;
    }
}

```

CÓDIGO – USUARIO

```
package modelo;

import java.util.Observable;

public abstract class Usuario extends Observable{

    private String nombreUsuario;
    private String nombreCompleto;
    private String clave;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets">
    public String getNombreUsuario(){
        return nombreUsuario;
    }
    public void setNombreUsuario(String nombreUsuario) {
        this.nombreUsuario = nombreUsuario;
    }
    public String getNombreCompleto(){
        return nombreCompleto;
    }
    public void setNombreCompleto(String nombreCompleto) {
        this.nombreCompleto = nombreCompleto;
    }
    public String getClave(){
        return clave;
    }
    public void setClave(String clave) {
        this.clave = clave;
    }
    // </editor-fold>

    @Override
    public boolean equals(Object obj) {
        Usuario usuario = (Usuario)obj;
        return nombreUsuario.equals(usuario.getNombreUsuario());
    }

    @Override
    public String toString() {
        return nombreCompleto;
    }

}
```

CÓDIGO – SISTEMAMOZOS

```
package modelo;

import java.util.ArrayList;

public class SistemaMozos {

    private ArrayList<Mozo> mozos;
    private ArrayList<Mozo> mozosActivos;

    // <editor-fold defaultstate="collapsed" desc="Agregar y Remover Mozos
+ Constructor">
    protected SistemaMozos() {
        this.mozos = new ArrayList();
        this.mozosActivos = new ArrayList();
    }

    public ArrayList<Mozo> getMozos() {
        return mozos;
    }

    public void agregarMozo(Mozo mozo) {
        this.mozos.add(mozo);
    }

    public void removerMozo(Mozo mozo) {
        this.mozos.remove(mozo);
    }

    public ArrayList<Mozo> getMozosActivos() {
        return mozosActivos;
    }

    public void agregarMozoActivo(Mozo mozo) {
        this.mozosActivos.add(mozo);
    }

    public void removerMozoActivo(Mozo mozo) {
        this.mozosActivos.remove(mozo);
    }
    // </editor-fold>

    public Mozo ingresar(String nombreUsuario, String clave) throws
ModeloException {
        boolean encontrado = false;
        if(nombreUsuario != null && clave != null){
            for (Mozo m : mozos) {
                if (m.getNombreUsuario().equals(nombreUsuario) &&
m.getClave().equals(clave)) {
                    if (!mozoLogueado(m)) {
                        mozosActivos.add(m);
                        return m;
                    } else {
                        throw new ModeloException("El usuario ya está
logueado");
                    }
                }
            }
        }
        if(!encontrado){
            throw new ModeloException("Credenciales incorrectas");
        }
    }
}
```



```

        }
    }
    return null;
}

public boolean salir(Mozo mozo) {
    boolean exito = false;
    if (mozosActivos.contains(mozo)) {
        if (!mozo.tieneMesasAbiertas()) {
            mozosActivos.remove(mozo);
            exito = true;
        }
    }
    return exito;
}

public boolean mozoLogueado(Mozo mozo){
    return mozo != null && mozosActivos.contains(mozo);
}

public void desloguearMozo(Mozo mozo) throws ModeloException {
    if(mozo.tieneMesasAbiertas())
        throw new ModeloException("No es posible salir del sistema
teniendo mesas abiertas\nCierre o transfieralas antes de salir");
    if(mozosActivos.contains(mozo))
        mozosActivos.remove(mozo);
}
}

```

CÓDIGO – SISTEMAGESTORES

```
package modelo;

import java.util.ArrayList;

public class SistemaGestores {

    private ArrayList<Gestor> gestores;
    private ArrayList<Gestor> gestoresActivos;
    private ArrayList<UnidadProcesadora> unidades;

    // <editor-fold defaultstate="collapsed" desc="Agregar y Remover
    Gestores-Unidades + Constructor">
    protected SistemaGestores() {
        this.gestores = new ArrayList();
        this.gestoresActivos = new ArrayList();
        this.unidades = new ArrayList();
    }

    public ArrayList<Gestor> getGestors() {
        return gestores;
    }

    public void agregarGestor(Gestor gestor) {
        this.gestores.add(gestor);
    }

    public void removerGestor(Gestor gestor) {
        this.gestores.remove(gestor);
    }

    public ArrayList<Gestor> getGestorsActivos() {
        return gestoresActivos;
    }

    public void agregarGestorActivo(Gestor gestor) {
        this.gestoresActivos.add(gestor);
    }

    public void removerGestorActivo(Gestor gestor) {
        this.gestoresActivos.remove(gestor);
    }

    public void agregarUnidadProcesadora(UnidadProcesadora unidad) {
        this.unidades.add(unidad);
    }

    public void removerUnidadProcesadora(UnidadProcesadora unidad) {
        this.unidades.remove(unidad);
    }

    public ArrayList<UnidadProcesadora> retornarUnidadesProcesadoras() {
        return this.unidades;
    }
    // </editor-fold>

    public Gestor ingresar(String nombreUsuario, String clave) throws
    ModeloException {
        boolean encontrado = false;
        if(nombreUsuario != null && clave != null){
```

```

        for (Gestor g : gestores) {
            if (g.getNombreUsuario().equals(nombreUsuario) &&
g.getClave().equals(clave)) {
                if (!gestorLogueado(g)) {
                    gestoresActivos.add(g);
                    return g;
                } else {
                    throw new ModeloException("El usuario ya está
logueado");
                }
            }
        }
        if(!encontrado){
            throw new ModeloException("Credenciales incorrectas");
        }
    }
    return null;
}

public boolean salir(Gestor gestor) {
    boolean exito = false;
    if (gestoresActivos.contains(gestor)) {
        gestoresActivos.remove(gestor);
        exito = true;
    }
    return exito;
}

public boolean gestorLogueado(Gestor gestor){
    return gestor != null && gestoresActivos.contains(gestor);
}

public void desloguearGestor(Gestor gestor) throws ModeloException {
    if(false)
        throw new ModeloException("No es posible salir del sistema");
    if(gestoresActivos.contains(gestor))
        gestoresActivos.remove(gestor);
}

public ArrayList<Producto> getProductosConStock(){
    ArrayList<Producto> lista = new ArrayList();
    for(UnidadProcesadora u : unidades){
        lista.addAll(u.getProductosConStock());
    }
    return lista;
}
}

```

CÓDIGO – SISTEMA

```
package modelo;

import java.util.ArrayList;

public class Sistema {

    private SistemaMozos sisM = new SistemaMozos();
    private SistemaGestores sisG = new SistemaGestores();
    private static Sistema instancia = new Sistema();

    public static Sistema getInstancia() {
        return instancia;
    }
    private Sistema() { }

    public Mozo loginMozo(String usuario, String password) throws
ModeloException {
        return sisM.ingresar(usuario, password);
    }

    public void desloguearMozo(Mozo mozo) throws ModeloException {
        sisM.desloguearMozo(mozo);
    }

    public void desloguearGestor(Gestor gestor) throws ModeloException {
        sisG.desloguearGestor(gestor);
    }

    public ArrayList<Producto> getProductosConStock() {
        return sisG.getProductosConStock();
    }

    public ArrayList<Mozo> getMozosLogueados() {
        return sisM.getMozosActivos();
    }

    public Gestor loginGestor(String usuario, String password) throws
ModeloException {
        return sisG.ingresar(usuario, usuario);
    }

    public ArrayList<UnidadProcesadora> getUnidadesProcesadoras() {
        return sisG.retornarUnidadesProcesadoras();
    }

    public void cargarDatos() {
        Mozo m1 = new Mozo();
        Mozo m2 = new Mozo();
        Mozo m3 = new Mozo();
        Gestor g4 = new Gestor();
        Gestor g5 = new Gestor();
        Gestor g6 = new Gestor();

        m1.setNombreCompleto("Jaime Talero");
        m2.setNombreCompleto("Esteban Quito");
        m3.setNombreCompleto("Aldo Lorigo");
        g4.setNombreCompleto("Marta Tuada");
    }
}
```

```

g5.setNombreCompleto("Guillermo Nigote");
g6.setNombreCompleto("Mary Quita");

m1.setNombreUsuario("a");
m2.setNombreUsuario("b");
m3.setNombreUsuario("c");
g4.setNombreUsuario("d");
g5.setNombreUsuario("e");
g6.setNombreUsuario("f");

m1.setClave("a");
m2.setClave("b");
m3.setClave("c");
g4.setClave("d");
g5.setClave("e");
g6.setClave("f");

Mesa me01 = new Mesa();
Mesa me02 = new Mesa();
Mesa me03 = new Mesa();
Mesa me04 = new Mesa();
Mesa me05 = new Mesa();
Mesa me06 = new Mesa();
Mesa me07 = new Mesa();
Mesa me08 = new Mesa();
Mesa me09 = new Mesa();
Mesa me10 = new Mesa();
Mesa me11 = new Mesa();
Mesa me12 = new Mesa();
me01.setNro(3);
me02.setNro(5);
me03.setNro(7);
me04.setNro(8);
me05.setNro(10);
me06.setNro(12);
me07.setNro(13);
me08.setNro(14);
me09.setNro(16);
me10.setNro(18);
me11.setNro(19);
me12.setNro(20);
me07.abrir();
me08.abrir();
me10.abrir();
me11.abrir();

m3.agregarMesa(me01);
m3.agregarMesa(me02);
m3.agregarMesa(me03);
m3.agregarMesa(me04);
m3.agregarMesa(me05);
m1.agregarMesa(me06);
m1.agregarMesa(me07);
m1.agregarMesa(me08);
m2.agregarMesa(me09);
m2.agregarMesa(me10);
m2.agregarMesa(me11);
m2.agregarMesa(me12);

sisM.agregarMozo(m1);

```

```

        sisM.agregarMozo(m2);
        sisM.agregarMozo(m3);
        sisG.agregarGestor(g4);
        sisG.agregarGestor(g5);
        sisG.agregarGestor(g6);

        UnidadProcesadora u1 = new UnidadProcesadora();
        UnidadProcesadora u2 = new UnidadProcesadora();

        u1.setNombre("Cocina");
        u2.setNombre("Bar");

        sisG.agregarUnidadProcesadora(u1);
        sisG.agregarUnidadProcesadora(u2);

        Producto.loadProductosFromBD();

//        Producto p01 = new Producto("c001","Pescado a la
plancha",260.00,15,u1);
//        Producto p02 = new Producto("c002","Milanesa
napolitana",360.00,25,u1);
//        Producto p03 = new Producto("c003","Ensalada de la
casa",220.00,8,u1);
//        Producto p04 = new Producto("c004","Empanadas variadas
(1un)",50.00,45,u1);
//        Producto p05 = new Producto("c005","Tallarines
Lomein",300.00,40,u1);
//        Producto p06 = new Producto("c006","Arrollado de
pollo",350.00,0,u1);
//        Producto p07 = new Producto("b001","Refresco 1Lt",120.00,19,u2);
//        Producto p08 = new Producto("b002","Agua 500ml",60.00,15,u2);
//        Producto p09 = new Producto("b003","Cerveza 1Lt",180.00,1,u2);
//        Producto p10 = new Producto("b004","Café",35.00,100,u2);

    }

}

```

CÓDIGO – MODELOEXCEPTION

```
package modelo;

public class ModeloException extends Exception {

    public ModeloException(String message) {
        super(message);
    }
}
```

CÓDIGO – LOGINCONTROLADORGESTOR

```
package gestorApp.controlador;

import java.util.ArrayList;
import modelo.Gestor;
import modelo.ModeloException;
import modelo.Sistema;
import modelo.UnidadProcesadora;

public class LoginControladorGestor {

    private LoginVistaGestor vista;
    private Sistema sistema = Sistema.getInstance();
    private Gestor gestor;

    public LoginControladorGestor(LoginVistaGestor vista){
        this.vista = vista;
    }

    public void empezarLogin(String usuario, String password){
        try{
            Gestor gestor = sistema.loginGestor(usuario, password);
            this.gestor = gestor;

            vista.seleccionarUnidadProcesadora(sistema.getUnidadesProcesadoras());
        } catch (ModeloException ex){
            vista.error(ex.getMessage());
        }
    }

    public void finalizarLogin(UnidadProcesadora unidad){
        gestor.setUnidad(unidad);
        vista.ingresar(gestor);
    }
}
```

CÓDIGO – LOGINVISTAGESTOR

```
package gestorApp.controlador;

import java.util.ArrayList;
import modelo.Gestor;
import modelo.UnidadProcesadora;

public interface LoginVistaGestor {

    public void error(String mensaje);
    public void ingresar(Gestor gestor);
    public void seleccionarUnidadProcesadora(ArrayList<UnidadProcesadora>
unidades);
}
```


CÓDIGO – LOGINCONTROLADORMOZO

```
package mozoApp.controlador;

import modelo.ModeloException;
import modelo.Mozo;
import modelo.Sistema;

public class LoginControladorMozo {

    private LoginVistaMozo vista;
    private Sistema sistema = Sistema.getInstancia();

    public LoginControladorMozo(LoginVistaMozo vista){
        this.vista = vista;
    }

    public void login(String usuario, String password){
        try{
            Mozo mozo = sistema.loginMozo(usuario, password);
            vista.ingresar(mozo);
        } catch (ModeloException ex){
            vista.error(ex.getMessage());
        }
    }
}
```

CÓDIGO – LOGINVISTAMOZO

```
package mozoApp.controlador;

import modelo.Mozo;

public interface LoginVistaMozo {

    public void error(String mensaje);
    public void ingresar(Mozo mozo);
}
```

CÓDIGO – MAINCONTROLADORGESTOR

```
package gestorApp.controlador;

import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import modelo.Gestor;
import modelo.ModeloException;
import modelo.Sistema;
import modelo.UnidadProcesadora;

public class LoginControladorGestor {

    private LoginVistaGestor vista;
    private Sistema sistema = Sistema.getInstancia();
    private Gestor gestor;

    public LoginControladorGestor(LoginVistaGestor vista){
        this.vista = vista;
    }

    public void empezarLogin(String usuario, String password){
        try{
            Gestor gestor = sistema.loginGestor(usuario, password);
            this.gestor = gestor;

            vista.seleccionarUnidadProcesadora(sistema.getUnidadesProcesadoras());
        } catch (ModeloException ex){
            vista.error(ex.getMessage());
        }
    }

    public void finalizarLogin(UnidadProcesadora unidad){
        gestor.setUnidad(unidad);
        vista.ingresar(gestor);
    }

    public void desloguearGestor() {
        try {
            sistema.desloguearGestor(gestor);
        } catch (ModeloException ex) {

        }
    }
}
```

CÓDIGO – MAINVISTAGESTOR

```
package gestorApp.controlador;

import java.util.ArrayList;
import modelo.Pedido;

public interface MainVistaGestor {

    public void mostrarPedidosPendientes (ArrayList<Pedido>
pedidosPendientes,ArrayList<Pedido> pedidosTomados);

    public void mostrarDialogPedidoSeleccionado (Pedido pedidoSeleccionado);

}
```

CÓDIGO – MAINCONTROLADORMOZO

```
package mozoApp.controlador;

import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;
import java.util.logging.Level;
import java.util.logging.Logger;
import modelo.Item;
import modelo.Mesa;
import modelo.ModeloException;
import modelo.Mozo;
import modelo.Producto;
import modelo.Servicio;
import modelo.Sistema;
import modelo.Transferencia;

/**
 *
 * @author SG0208533
 */
public class MainControladorMozo implements Observer {

    private MainVistaMozo vista;
    private Sistema sistema = Sistema.getInstancia();
    private Mozo mozo;
    private Mesa mesaSeleccionada;

    public MainControladorMozo(MainVistaMozo vista, Mozo mozo) {
        this.vista = vista;
        this.mozo = mozo;
        this.mozo.addObserver(this);
        vista.mostrarMesas(null, this.mozo);
    }

    public void seleccionarMesa(int indexMesa) {
        if(indexMesa < mozo.getMesas().size()) {
            Mesa mesa = mozo.getMesas().get(indexMesa);
            seleccionarMesa(mesa);
        }
    }

    public void seleccionarMesa(Mesa mesa) {
        mesaSeleccionada = mesa;
        vista.mostrarMesas(mesa, mozo);
    }

    public boolean desloguearMozo() {
        try{
            sistema.desloguearMozo(mozo);
            mozo.deleteObserver(this);
            return true;
        } catch (ModeloException ex) {
            vista.mostrarAlerta(ex.getMessage());
            return false;
        }
    }

    public void abrirMesa() {
        mesaSeleccionada.abrir();
        vista.mostrarMesas(mesaSeleccionada, mozo);
    }
}
```

```

    }

    public void cerrarMesa() {
        try {
            mesaSeleccionada.cerrar();
            vista.mostrarMesas(mesaSeleccionada, mozo);
        } catch (ModeloException ex) {
            vista.mostrarAlerta(ex.getMessage());
        }
    }

    public ArrayList<Producto> getProductosConStock() {
        return sistema.getProductosConStock();
    }

    public void agregarItemALaMesa(int indexProd,int cantidad, String
descripcion){
        Item item = new Item();
        item.setProducto(getProductosConStock().get(indexProd));
        item.setCantidad(cantidad);
        item.setDescripcion(descripcion);
        agregarItemALaMesa(item);
    }

    public void agregarItemALaMesa(Item item) {
        try {
            mesaSeleccionada.getServicio().agregarItem(item);
            vista.mostrarMesas(mesaSeleccionada, mozo);
        } catch (ModeloException ex) {
            vista.mostrarAlerta(ex.getMessage());
        }
    }

    private ArrayList<Mozo> otrosMozosLogueados() {
        ArrayList<Mozo> mozosLogueados = new ArrayList();
        for (Mozo m:sistema.getMozosLogueados()) {
            if(!m.equals(mozo)) {
                mozosLogueados.add(m);
            }
        }
        return mozosLogueados;
    }

    public void iniciarTransferirMesa(){
        vista.mostrarTransferirMesa(otrosMozosLogueados(),
mesaSeleccionada);
    }

    public void transferirMesa(int indexMozo){
        transferirMesa(otrosMozosLogueados().get(indexMozo));
    }

    public void transferirMesa(Mozo mozoDestino) {
        Transferencia transferencia = new Transferencia();
        transferencia.setMesa(mesaSeleccionada);
        transferencia.setMozoDestino(mozoDestino);
        mozo.solicitarTransferencia(transferencia);
    }

    @Override
    public void update(Observable origen, Object evento) {

```

```

        if(evento.equals(Mozo.eventos.transferenciaSolicitadaFrom)){
            vista.mostrarMesas(mozo.getTransferencia().getMesa(), mozo);
        } else if(evento.equals(Mozo.eventos.transferenciaSolicitadaTo)){
            vista.mostrarTransferenciaSolicitud(mozo.getTransferencia());
        } else if(evento.equals(Mozo.eventos.transferenciaRechazada)){
            vista.mostrarMesas(mesaSeleccionada, mozo);
            vista.mostrarAlerta("La solicitud de transferecia fue
rechazada.");
        } else if(evento.equals(Mozo.eventos.transferenciaAceptada)){
            mesaSeleccionada = null;
            vista.mostrarMesas(mesaSeleccionada, mozo);
            vista.mostrarAlerta("La solicitud de transferecia fue
aceptada.");
        } else if(evento.equals(Mozo.eventos.transferenciaActualizada)){
            vista.mostrarTransferenciaActualizada();
        } else
if(evento.equals(Mozo.eventos.transferenciaTiempoTerminado)){
            vista.terminarTransferenciaPorTiempoTerminado();
        } else if(evento.equals(Mozo.eventos.pedido)){
            vista.mostrarMesas(mesaSeleccionada, mozo);
        }
    }

    public void transferenciaAceptar() {
        mozo.aceptarTransferencia();
        vista.mostrarMesas(mesaSeleccionada, mozo);
    }

    public void transferenciaRechazar() {
        mozo.rechazarTransferencia();
        vista.mostrarMesas(mesaSeleccionada, mozo);
    }

    public void agregarClienteALaMesa(int idCliente){
        try {
            mesaSeleccionada.agregarCliente(idCliente);
            vista.mostrarMesas(mesaSeleccionada, mozo);
        } catch (ModeloException ex) {
            vista.mostrarAlerta(ex.getMessage());
        }
    }
}

```

CÓDIGO – MAINVISTAMOZO

```
package mozoApp.controlador;

import java.util.ArrayList;
import modelo.Mesa;
import modelo.Mozo;
import modelo.Transferencia;

public interface MainVistaMozo {

    public void mostrarMesas(Mesa mesa, Mozo mozo);
    public void mostrarAlerta(String mensaje);
    public void mostrarTransferirMesa(ArrayList<Mozo> mozosLogueados, Mesa
mesa);
    public void mostrarTransferenciaSolicitud(Transferencia transferencia);
    public void mostrarTransferenciaActualizada();
    public void terminarTransferenciaPorTiempoTerminado();

}
```