UNIVERSIDAD ORT URUGUAY

FACULTAD DE INGENIERÍA "ING. BERNARD WAND-POLAK"

EVALUACIÓN: OBLIGATORIO (PRIMERA ENTREGA)

MATERIA: DISEÑO Y DESARROLLO DE APLICACIONES

PROFESOR: DARÍO CAMPALANS

FECHA DE ENTREGA: 26/10/17

GRUPO: M3A

INTEGRANTES:

DANIEL FRIEDMANN (144276)

GIANCARLO BELTRAMINI (195827)

CONTENIDO

Introducción y Autoevaluación

Diagramas (de clase y dominio)

Ambos diagramas fueron utilizados para la generación del modelo y luego actualizados con la información de la interfaz, siguiendo la arquitectura MVC.

Precarga de datos

Como lo solicita la letra, se agregan datos al iniciar el sistema, tanto de usuarios (gestores y mozos) como unidades procesadoras y otros datos necesarios para el uso inicial de la aplicación (Productos). Para obtener los accessos al sistema, ver *Usuarios precargados*.

Inicio de las aplicaciones

La aplicación se inicia mostrando 2 botones, uno para cada tipo de usuario del sistema. El botón Mozos, da acceso a la aplicación de atención de mesas y el botón Gestores da a la aplicación procesadora de pedidos. Nos vimos forzados a hacerlo de ésta manera, ya que todo corre dentro de un mismo equipo.

Aplicación de atención de mesas

No notamos bugs, y los casos de usos han sido cubiertos.

Aplicación procesadora de pedidos

No notamos bugs, y los casos de usos han sido cubiertos. Le agregamos la posibilidad al gestor de cerrar su sesión, aunque como mejora posible se debería impedir la salida del mismo teniendo pedidos sin finalizar, similar al caso de uso del logout de mozos.

Requerimientos de diseño

Creemos haber contemplado todos los requerimientos solicitados. La división lógica está hecha en 3 paquetes, uno de modelo, otro de la aplicación de gestores y otro de la de mozos, y a su vez ambos paquetes se dividen en su controlador y vista respectivos. El patrón Experto fue aplicado en la asignación de responsabilidades de la manera que mejor consideramos, así como el uso de una fachada (Sistema) para ocultar los sistemas de la interfaz de usuario.

DIAGRAMA CONCEPTUAL DE DOMINIO

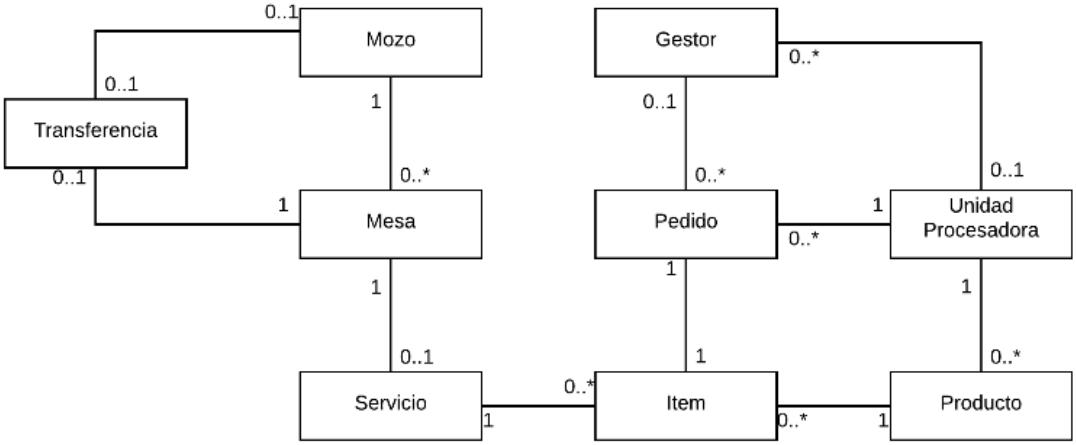
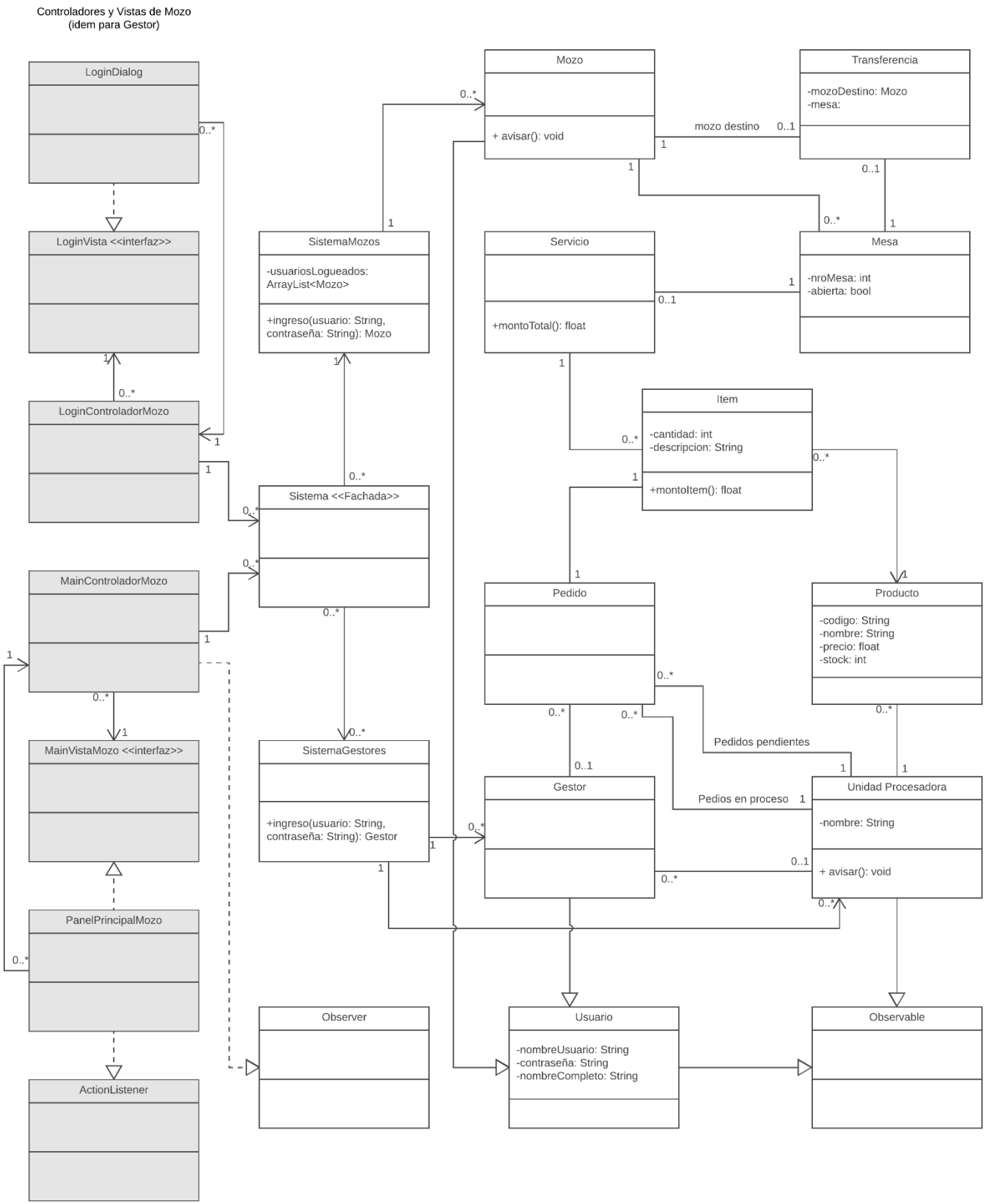


DIAGRAMA DE CLASES



USUARIOS PRECARGADOS

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 Lorido
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

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,
            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;

    // <editor-fold defaultstate="collapsed" desc="Gets y Sets y Agregar-
Remover 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);
    }
    // </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.abierta = false;
                this.servicio = new Servicio();
            } 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.removerMesa(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);
    }
}
```

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 – ITEM

```
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 {

    private Mesa mesa;
    private Mozo mozoDestino;

    // <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);
    }
    // </editor-fold>

    public void confirmar(){
        mesa.transferenciaAceptada(mozoDestino);
    }

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

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

    public void transferenciaRechazada() {
        mesa.transferenciaRechazada(mozoDestino);
    }

}
```

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
Remover 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() { cargarDatos(); }

    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();
    }



    private 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 Lorido");
```

```
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");

        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);

        sisG.agregarUnidadProcesadora(u1);
        sisG.agregarUnidadProcesadora(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.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);
    }
}
```

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.Observable;
import java.util.Observer;
import modelo.Gestor;
import modelo.Item;
import modelo.Item.Estado;
import modelo.ModeloException;
import modelo.Pedido;
import modelo.Sistema;
import modelo.UnidadProcesadora;

public class MainControladorGestor implements Observer {

    private MainVistaGestor vista;
    private Sistema sistema = Sistema.getInstancia();
    private Gestor gestor;
    private UnidadProcesadora unidad;
    private Pedido pedidoSeleccionado;

    public MainControladorGestor(MainVistaGestor vista, Gestor gestor) {
        this.vista = vista;
        this.gestor = gestor;
        this.unidad = gestor.getUnidad();
        this.unidad.addObserver(this);
        mostrarPedidos();
    }

    public void desloguearGestor() throws ModeloException {
        sistema.desloguearGestor(gestor);
        unidad.deleteObserver(this);
    }

    public void mostrarPedidos() {
        ArrayList<Pedido> pedidosPendientes =
gestor.getUnidad().getPedidosPendientes();
        ArrayList<Pedido> pedidosTomados = gestor.getPedidos();
        vista.mostrarPedidosPendientes(pedidosPendientes, pedidosTomados);
    }

    public void procesarPedidoSeleccionado(){

if(pedidoSeleccionado.getItem().getEstado().equals(Estado.enProceso)){
            gestor.finalizarPedido(pedidoSeleccionado);
        }else if
(pedidoSeleccionado.getItem().getEstado().equals(Estado.pendiente)){
            gestor.trabajarPedido(pedidoSeleccionado);
        }
    }

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

    public void seleccionarPedido(Pedido pedidoSeleccionado) {
        this.pedidoSeleccionado = pedidoSeleccionado;
        vista.mostrarDialogPedidoSeleccionado(pedidoSeleccionado);
```

```
    }

    public void deseleccionarPedido(){
        this.pedidoSeleccionado = null;
    }

}
```

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 modelo.Item;
import modelo.Mesa;
import modelo.ModeloException;
import modelo.Mozo;
import modelo.Producto;
import modelo.Servicio;
import modelo.Sistema;
import modelo.Transferencia;

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

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

    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.pedido)){
            vista.mostrarMesas(mesaSeleccionada, mozo);
        }
    }

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

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

}
```

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);

}
```