

Caso de uso para la defensa	2
CU-01: Comprar pasaje para viaje.	2
CU-01 Comprar pasaje para viaje (Reestructurado).	4
Pantallas implicadas en el caso de uso	5
El botón viajar dispara el caso de uso. Luego se ingresan los destinos entre los que se quiere viajar.	5
Cuestiones de seguridad:	9
Código	10
HTML Index.jsp	10
AppDataException.java	13
NoDestinoException.java (NoServiceExcepcion.java es similar)	13
FiltroNormalUser.java	14
SevletBuscarServicio.java	15
ServletVentaPasaje.java	16
ServletLogin.java	17
FactoryConexion.java	19
DataDestino.Java	20
DataServicio.Java	23
DataPersona.Java	30
DataMicro.Java	33

Caso de uso para la defensa

CU-01: Comprar pasaje para viaje.

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Sin-estructurar	Sistema	Negra	Real	Dialogal

Meta: Comprar un pasaje.

ACTORES

Primario: Cliente.

PRECONDICIONES (de sistema): El cliente está logueado en el sistema. Existen servicios disponibles.

DISPARADOR: El cliente ingresa a la pagina para comprar un pasaje y presiona viajar.

FLUJO DE SUCECOS

CAMINO BÁSICO:

1. El cliente ingresa origen y destino.
2. El sistema valida que existan servicios para los destinos.
3. El cliente ingresa numero de servicio.
4. El sistema
 - 4.1 valida que exista el número ingresado.
 - 4.2 muestra los micros disponibles.
5. El cliente ingresa patente del micro.
6. El sistema
 - 6.1 valida que exista la patente ingresada.
 - 6.2 muestra las butacas disponibles.
7. El cliente selecciona butaca.
8. El sistema valida.
9. El cliente ingresa dni, numero de tarjeta, nombre del titular, fecha de vencimiento,ccv y confirma el pago.
- 10.El sistema valida datos ingresados y muestra pasaje.
- 11.El cliente ingresa e-mail para recibir el pasaje.
- 12.El sistema envía el pasaje al mail.

CAMINO ALTERNATIVO:

- 2.a No existen servicios disponibles para el destino solicitado.
 - 2.a.1 El sistema informa la situación.
 - 2.a.2 Termina el CU.
- 4.1.a El número de servicio ingresado es incorrecto.

- 4.1.a.1 El sistema informa.
- 4.1.a.2 vuelve al paso 3.
- 6.1.a La patente del micro ingresada es incorrecta.
 - 6.1.a.1 El sistema informa.
 - 6.1.a.2 vuelve al paso 5.
- 10.a Faltan datos que ingresar.
 - 10.a.1 El sistema informa.
 - 10.a.2 vuelve al paso 9.

POSTCONDICIONES (de negocio)

Éxito: Se vendió un nuevo pasaje.

Fracaso: No se vendió el pasaje por no existir servicios disponibles.

Éxito alternativo: Se vendió un pasaje luego de que el usuario eligiera otro destino.

POSTCONDICIONES (de sistema)

Éxito: Se registró la venta de un pasaje.

Fracaso: No hubo registros nuevos.

CU-01 Comprar pasaje para viaje (Reestructurado).

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Usuario	Reestructurado	Sistema	Negra	Real	Semantico

ACTORES

Primario: Cliente.

PRECONDICIONES (de sistema): El cliente está cargado en el sistema.

DISPARADOR: El cliente ingresa a la página para comprar un pasaje

FLUJO DE SUCESOS

CAMINO BÁSICO:

1. El cliente ingresa a la página y se loguea invocando a <CUU.01 Iniciar sesión>.
2. El cliente ingresa destinos, servicio, patente de micro y selecciona butaca invocando a <CUU.02 Seleccionar servicio>.
3. El cliente ingresa datos personales y datos de la tarjeta de pago invocando a CUU.03 Realizar pago>.

CAMINO ALTERNATIVO:

- 1.a <Reemplaza> El cliente no se loguea en el sistema.
 - 1.a.1 Continúa con el paso 2.
- 2.a No existen servicios disponibles para el destino solicitado.
 - 2.a.1 Termina el CU.

POSTCONDICIONES (de negocio)

Éxito: Se vendió un nuevo pasaje.

Fracaso: No se vendió el pasaje por no existir servicios disponibles.

POSTCONDICIONES (de sistema)

Éxito: Se registró la venta de un pasaje.

Fracaso: No hubo registros nuevos.

Pantallas implicadas en el caso de uso

El botón viajar dispara el caso de uso. Luego se ingresan los destinos entre los que se quiere viajar.



SERVICIOS

BUSCAR

Cargar destinos disponibles

El Servlet ServletBuscarServicios se encarga de buscar los servicios correctos y devolver la siguiente pantalla:

Lista de servicios encontrados:

Numero	Fecha partida	Hora partida
3	2018-12-27	13:00
4	2019-01-10	19:00

N° de servicio:

4

- +

Siguiente

Se muestran los micros asociados al servicio:

Lista de micros asignados al servicio elegido:

Patente	Marca	Fecha Ultimo Control	Porcentaje aumento	Tipo
123ABC	Fiat	2017-01-01	0.0	Micro
89ABC00	Ford	2017-09-26	10.0	MicroCama
SSNNN11	Mercedes Benz	2016-04-05	0.0	Micro

Ingrese la patente del micro:

Patente:

Ej: AA123BB o 961ASD

Continuar

Se muestran las butacas disponibles:

Pasajero23

Lista de butacas disponibles:

Numero	Seleccion
1	<input checked="" type="radio"/>
2	<input checked="" type="radio"/>
3	<input checked="" type="radio"/>
4	<input type="radio"/>
5	<input checked="" type="radio"/>
6	<input type="radio"/>
7	<input type="radio"/>
8	<input type="radio"/>
9	<input type="radio"/>
10	<input type="radio"/>
11	<input type="radio"/>
12	<input type="radio"/>
13	<input checked="" type="radio"/>
14	<input type="radio"/>
15	<input type="radio"/>
16	<input type="radio"/>
17	<input type="radio"/>
18	<input type="radio"/>
19	<input checked="" type="radio"/>
20	<input type="radio"/>


Butaca seleccionada

N°:

Ej: 1, 5, 12.

Continuar

El servlet `ServletVentaPasaje` obtiene y verifica los datos del pago:

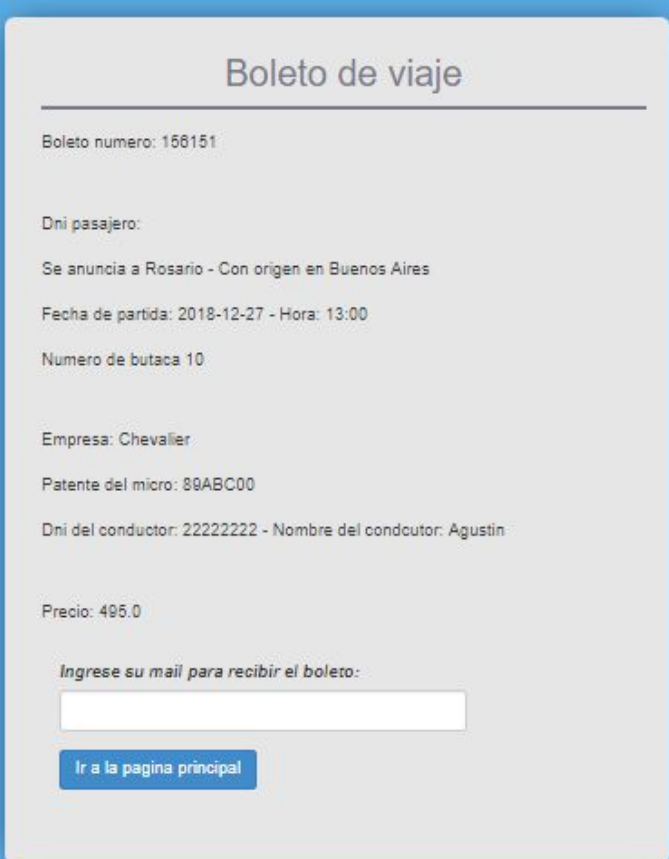


Formulario de pago con tarjeta Visa. El formulario está sobre un fondo verde claro. En la parte superior hay una imagen de una tarjeta Visa roja con el texto "CARD HOLDER" y "EXPIRES". Debajo de la tarjeta, el formulario tiene los siguientes campos:

- DNI DEL PASAJERO:
- NUMERO:
- NOMBRE DEL TITULAR:
- FECHA DE VENCIMIENTO:
- CCV:

En la parte inferior del formulario hay un botón azul con el texto "CONFIRMAR PAGO".

Finalmente, se muestra el boleto por pantalla:



Boleto de viaje

Boleto numero: 158151

Dni pasajero:

Se anuncia a Rosario - Con origen en Buenos Aires

Fecha de partida: 2018-12-27 - Hora: 13:00

Numero de butaca 10

Empresa: Chevalier

Patente del micro: 88ABC00

Dni del conductor: 22222222 - Nombre del conductor: Agustin

Precio: 495.0

Ingrese su mail para recibir el boleto:

[Ir a la pagina principal](#)

Cuestiones de seguridad:

- Uso de preparedStatements para hacer consultas a la base de datos.
- Uso de filtros para no devolver páginas sin permisos adecuados. (FiltroRoot y FiltroNormalUser).
- En cuanto al login:
 - Del lado del cliente usamos el tag input con el type password para que el usuario no vea en pantalla la contraseña que ingresó.
 - Del lado del servidor, se guarda la contraseña hasheada en a la base de datos (Faltó implementar).
- Para asegurar que no abra múltiples conexión a la base de datos, se implementa el patrón SINGLETON en la clase FactoryConexion, de manera que esta se pueda instanciar una sola vez. Además, se libera la conexion automáticamente cuando no existen aplicaciones conectadas.

Se usó el método executeBatch de la clase PreparedStatement para optimizar la ejecución de multiples operaciones de carga en la base de datos.

Se podría haber agregado el uso de commit y rollback (métodos de Connection), para que en caso de fallo de una consulta, se pueda volver hacia atrás y la base de datos quede consistente.
- Se manejaron distintos tipos de Excepciones:
 - AppDataException para errores relacionados al acceso a la base de datos
 - NoServiceException para errores de existencia de servicios en la base de datos.
 - NoDestinoException para errores de existencia de destinos en la base de datos.

Código

HTML Index.jsp

```
%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%
<%@page import="java.util.ArrayList"%>
<%@page import="entities.Destino"%>
<%@page import="entities.DestinoDirecto"%>

<!DOCTYPE html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport"
    content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">

<title>Terminal</title>

<!-- Bootstrap core CSS -->
<link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom fonts for this template -->
<link href="vendor/font-awesome/css/font-awesome.min.css"
    rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
    rel="stylesheet" type="text/css">
<link href="https://fonts.googleapis.com/css?family=Kaushan+Script"
    rel="stylesheet" type="text/css">
<link
    href="https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,700italic"
    rel="stylesheet" type="text/css">
<link
    href="https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700"
    rel="stylesheet" type="text/css">

<!-- Custom styles for this template -->
<link href="css/agency.min.css" rel="stylesheet">
<link href="css/agency.css" rel="stylesheet">
<link href="css/mystyle.css" rel="stylesheet">

</head>

<body id="page-top">

    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark fixed-top" id="mainNav">
        <div class="container">
            <a class="navbar-brand js-scroll-trigger" href="#page-top">Pasajero23</a>
            <button class="navbar-toggler navbar-toggler-right" type="button"
                data-toggle="collapse" data-target="#navbarResponsive"
                aria-controls="navbarResponsive" aria-expanded="false"
                aria-label="Toggle navigation">
                Menu <i class="fa fa-bars"></i>
            </button>
            <div class="collapse navbar-collapse" id="navbarResponsive">
                <ul class="navbar-nav text-uppercase ml-auto">
```

```

<li class="nav-item"><a class="nav-link js-scroll-trigger"
href="#services">Servicios</a></li>
<li class="nav-item"><a class="nav-link js-scroll-trigger"
href="#contact">Contacto</a></li>
<li class="fa fa-user nav-link js-scroll-trigger user"
href="/pages/loginUsuario.jsp"></a></li>
</ul>
</div>
</div>
</nav>

<!-- Header -->
<header class="masthead">
<div class="container">
<div class="intro-text">
<div class="intro-lead-in">Bienvenidos a Pasajero23</div>
<div class="intro-heading text-uppercase">Nos encanta que
estes aqui</div>
<div></div>
<a class="btn btn-primary btn-xl text-uppercase js-scroll-trigger"
href="#services">Viajar</a>
</div>
</div>
</header>

<!-- Services -->
<section id="services">
<div class="container">
<div class="row">
<div class="col-lg-12 text-center">
<h2 class="section-heading text-uppercase">Servicios</h2>
<div class="container" id="container"></div>
<!--
<script type="text/javascript">
document.getElementById('cargarDestinos').click();
</script>
<a id="cargarDestinos" href="ServletIndex"></a>
-->
<!--
ArrayList<Destino> dd = (ArrayList<Destino>) request.getSession().getAttribute("listaDestinos");
if (!(dd == null || dd.isEmpty())) {
<datalist id="destinos">
<!--
for (Destino d : dd) {
<option value="<%=d.getLocalidad()%"
label="<%=d.getLocalidad()%"></option>
<!--
}
</datalist>
<!--
}
<form method=get action="ServletBuscarServicios">
<input type="text" name="textOrigen" placeholder="Ingrese Origen"
class="search-bar" list="destinos"><br> <input
type="text" name="textDestino" placeholder="Ingrese Destino"
class="search-bar" list="destinos"> <br>
<!--
String m = null;
m = (String) request.getSession().getAttribute("mensaje");
if (m != null) {
<div style="color: red">
<i><%=m%></i>
</div>
<!--
}
<button id="sendMessageButton"
class="btn btn-primary btn-xl text-uppercase" type="submit">Buscar</button>
</form>
<form method=get action="ServletIndex">
<button id="cargarDestinos" class="btn" type="submit">
<i>Cargar destinos disponibles</i>
</button>
</form>
</div>
</div>
</div>
</section>

<!-- Contact -->
<section id="contact">
<div class="container">
<div class="row">
<div class="col-lg-12 text-center">
<h2 class="section-heading text-uppercase">Contáctanos</h2>
</div>
</div>
<div class="row">
<div class="col-lg-12">

```


AppDataException.java

```
package util;

public class AppDataException extends Exception{
    private Throwable innerException;
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public AppDataException(Throwable e, String message){
        this.innerException=e;
        this.setMessage(message);
    }
}
```

NoDestinoException.java (NoServiceException.java es similar)

```
package util;

public class NoDestinoException extends Exception{
    private Throwable innerException;
    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public NoDestinoException(Throwable e, String message){
        this.innerException=e;
        this.setMessage(message);
    }

    public NoDestinoException(String message){
        this.setMessage(message);
    }
}
```


FiltroNormalUser.java

```
package util;

import java.io.IOException;

/**
 * Servlet Filter implementation class FiltroNormalUser
 */
@WebFilter(
    description = "Rechazar conexiones a welcome.jsp si no hay un usuario logeado.",
    urlPatterns = {
        "/welcome.jsp",
        "/welcome.jsf"
    })
public class FiltroNormalUser implements Filter {

    /**
     * Default constructor.
     */
    public FiltroNormalUser() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#destroy()
     */
    public void destroy() {
        // TODO Auto-generated method stub
    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpSession s = ((HttpServletRequest) request).getSession();
        Usuario u = (Usuario) s.getAttribute("usuarioLogeado");
        if( u != null ) {
            chain.doFilter(request, response);
        }else {
            ((HttpServletResponse) response).sendRedirect("index.jsp");
        }
    }

    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
}
```

ServletBuscarServicio.java

```
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import business.LogicDestino;
import business.LogicServicio;
import entities.Destino;
import entities.Servicio;
import util.AppDataException;
import util.NoDestinoException;
import util.NoServiceException;

@WebServlet("/ServletBuscarServicios")
public class ServletBuscarServicios extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletBuscarServicios() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Destino origen = null;
        Destino destino = null;
        ArrayList<Servicio> ss = null;
        LogicDestino logd = new LogicDestino();

        String orig = request.getParameter("textOrigen");
        String dest = request.getParameter("textDestino");
        try {
            origen = logd.getByNombre(orig);
            destino = logd.getByNombre(dest);
            LogicServicio logSer = new LogicServicio();
            ss = logSer.getAllByDestinos(origen, destino);

            request.getSession().setAttribute("mensaje", null);
            request.getSession().setAttribute("desOrigen", origen);
            request.getSession().setAttribute("desLlegada", destino);
            request.getSession().setAttribute("serviciosEcontrados", ss);
            response.sendRedirect("pages/listarServiciosEcontrados.jsp");

        } catch (NoDestinoException e) {
            request.getSession().setAttribute("mensaje", e.getMessage());
            response.sendRedirect("index.jsp#services");
        } catch (NoServiceException e) {
            request.getSession().setAttribute("mensaje", e.getMessage());
            response.sendRedirect("index.jsp#services");

        } catch (AppDataException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

ServletVentaPasaje.java

```
package servlet;

import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import business.LogicPersona;
import business.LogicServicio;
import entities.Destino;
import entities.Micro;
import entities.Servicio;
import entities.Usuario;
import util.AppDataException;
import util.NoServiceException;

@WebServlet("/ServletVentaPasaje")
public class ServletVentaPasaje extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletVentaPasaje() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Servicio serv = null;
        LogicServicio logs = new LogicServicio();
        ArrayList<Servicio> ss = (ArrayList<Servicio>) request.getSession().getAttribute("serviciosEcontrados");
        int id = Integer.parseInt(request.getParameter("idServicio"));

        try {
            serv = (Servicio) logs.getById(id);

            request.getSession().setAttribute("mensaje", null);
            if(ss.contains(serv)) {
                request.getSession().setAttribute("servicio", serv);
                request.getSession().setAttribute("estadoventa", "SELECCIONARMICRO");
                ArrayList<Micro> mm = serv.getMicros();
                request.getSession().setAttribute("listaMicros", mm);
                System.out.println("servicios: " + serv.getIdServicio());
                response.sendRedirect("pages/ventaPasaje.jsp");
            }
            else {
                request.getSession().setAttribute("mensaje", "servicio no encontrado para la lista");
                System.out.println("servicio no encontrado para la lista");
                request.getSession().setAttribute("serviciosEcontrados", ss);
                response.sendRedirect("pages/listarServiciosEcontrados.jsp");
            }
        }
        catch (AppDataException e) {
            e.printStackTrace();
        }
        catch (NoServiceException e) {
            request.getSession().setAttribute("mensaje", e.getMessage());
            request.getSession().setAttribute("serviciosEcontrados", ss);
            response.sendRedirect("pages/listarServiciosEcontrados.jsp");
        }
        catch (SQLException e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        LogicPersona logp = null;
        Servicio ser = null;
        String estado = (String) request.getSession().getAttribute("estadoventa");
        switch (estado) {
            case "SELECCIONARMICRO":
                String patente = (String) request.getParameter("patente").toUpperCase();
                ser = (Servicio) request.getSession().getAttribute("servicio");
                Micro micro = null;
                for(Micro m: ser.getMicros()) {
                    if(m.getPatente().equals(patente)) {
                        micro = m; //Hago este asignacion para conocer el micro encontrado afuera del foreach.
                        request.getSession().setAttribute("micro", m);
                        request.getSession().setAttribute("estadoventa", "SELECCIONARBUTACA");
                        response.sendRedirect("pages/ventaPasaje.jsp");
                    }
                }
            }
        if (!(micro instanceof Micro)) {
            //No encontro el micro ingresado, redirige al mismo lugar.
        }
    }
}
```



```

        //No encontro el micro ingresado, redirige al mismo lugar.
        response.sendRedirect("pages/ventaPasaje.jsp");
    }
    break;

case "SELECCIONARBUTACA":
    int numButaca = Integer.parseInt(request.getParameter("numButaca"));
    request.getSession().setAttribute("numButaca", numButaca);
    request.getSession().setAttribute("estadoventa", "PAGARBOLETO");
    response.sendRedirect("pages/detallesPagoTarjeta/mostrarTarjeta.jsp");
    break;

case "PAGARBOLETO":
    String dni = null;
    LogicServicio logics= new LogicServicio();
    Usuario user = (Usuario) request.getSession().getAttribute("usuarioLogeado");
    Micro mic = (Micro) request.getSession().getAttribute("micro");
    Servicio servicio = (Servicio) request.getSession().getAttribute("servicio");
    int butaca = (int) request.getSession().getAttribute("numButaca");

    //Calculo del precio del boleto
    //*****

    Double precioOrigen = null, precioLlegada = null, precioSinAumento, aumentoMicro, aumentoDestino, precioFinal;
    Destino desllegada = null;
    Destino desOrigenSinPrecio = (Destino) request.getSession().getAttribute("desOrigen");
    Destino desllegadaSinPrecio = (Destino) request.getSession().getAttribute("desLlegada");
    for(Destino d: servicio.getDestinos()) {
        if(d.getLocalidad().equals(desOrigenSinPrecio.getLocalidad())){
            precioOrigen = d.getPrecioDestino();
        }
        if(d.getLocalidad().equals(desllegadaSinPrecio.getLocalidad())){
            precioLlegada = d.getPrecioDestino();
            desllegada = d;
        }
    }
    precioSinAumento = precioLlegada - precioOrigen;
    // 35 / 100= 0.35
    aumentoMicro = precioSinAumento * (mic.getAumento() / 100);
    aumentoDestino = precioSinAumento * (desllegada.getPorcentajeAumento() / 100);
    precioFinal = precioSinAumento + aumentoDestino + aumentoMicro;
    request.getSession().setAttribute("precio", precioFinal);

    //Fin calculo del precio *****

    if(user == null) {
        dni = (String) request.getParameter("dniPasajero");
        Usuario u = new Usuario(dni);
        logp = new LogicPersona();
        try {
            logp.add(u);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (AppDataException e) {
            e.printStackTrace();
        }
    } else {
        dni = user.getDni();
    }
    request.getSession().setAttribute("dni", dni);
    try {
        logics.addPasajero(servicio.getIdServicio(), dni, mic.getPatente(), butaca, desOrigenSinPrecio.getIdDestino(), desl
    } catch (AppDataException e) {
        e.printStackTrace();
    }
    request.getSession().setAttribute("estadoventa", "IMPRIMIRBOLETO");
    response.sendRedirect("pages/impresionBoleto/impresionBoleto.jsp");
    break;

case "IMPRIMIRBOLETO":
    user = (Usuario) request.getSession().getAttribute("usuarioLogeado");
    if(user == null) {
        response.sendRedirect("index.jsp");
    } else {
        //Para que actualice la estadística de destinos despues de la venta de uno, debería usar AJAX en welcome.jsp
        response.sendRedirect("welcome.jsp");
    }
    break;
}
}
}

```

ServletLogin.java

```

package servlet;

import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import business.LogicPersona;
import business.LogicServicio;
import entities.Usuario;
import util.AppDataException;

@WebServlet(description = "Server para manejar el login al sistema.", urlPatterns = { "/ServletLogin" })
public class ServletLogin extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ServletLogin() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        LogicPersona logPer = new LogicPersona();

        Usuario user = new Usuario();
        user.setNombreUsuario(request.getParameter("textUsuario"));
        user.setContraseña(request.getParameter("textContraseña"));
        try {
            user = logPer.getLogedUser(user);
            if(user != null) {
                if(user.esAdmin()) {
                    System.out.println("Ingreso correcto como administrador");
                    request.getSession().setAttribute("mensajeLogin", null);
                    LogicServicio logs = new LogicServicio();
                    try {
                        request.getSession().setAttribute("listaRecaudacionMes", logs.getRecaudacionPorMes());
                    } catch (AppDataException e) {
                        e.printStackTrace();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                    response.sendRedirect("../pages/adminPage.jsp");
                } else {
                    System.out.println("Ingreso correcto");
                    ArrayList<Usuario> usuarios = logPer.getAllUsuarios();
                    request.getSession().setAttribute("listaUsuarios", usuarios);
                    request.getSession().setAttribute("mensajeLogin", null);

                    //Mostrar informe de destinos mas solicitados
                    LogicServicio logics = new LogicServicio();
                    String[][] datos = null;
                    Calendar cl = Calendar.getInstance();
                    int ano = (cl.get(Calendar.YEAR));
                    int mes = (cl.get(Calendar.MONTH));
                    System.out.println("EL MES de hoy es " + mes);
                    System.out.println("EL AÑO de hoy es " + ano);
                    try {
                        datos = logics.getDestinosByMesAno(mes, ano);
                    } catch (AppDataException e) {
                        e.printStackTrace();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                    request.getSession().setAttribute("informeDestinos", datos);
                    response.sendRedirect("../welcome.jsp");
                }
                request.getSession().setAttribute("usuarioLogeado", user);
            } else {
                request.getSession().setAttribute("mensajeLogin", "Usuario incorrecto o inexistente");
                request.getSession().setAttribute("mensaje", null);
                response.sendRedirect("../pages/loginUsuario.jsp");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}

```

FactoryConexion.java

```
package data;
import java.sql.*;
import util.AppDataException;

public class FactoryConexion {

    private String driver="com.mysql.cj.jdbc.Driver";

    /*String host="node24713-env-4846480.jelastic.saveincloud.net";
    private String user="root";
    private String password="fmPwz0o1pE";*/
    String host="localhost";
    private String user="usertpjava";
    private String password="usertpjava";

    private String db="terminalTPJava";
    private String port="3306";
    private Connection conn;
    private int cantConn=0;
    private static FactoryConexion instancia;

    private FactoryConexion(){
        try {
            Class.forName(driver);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public static FactoryConexion getInstancia(){
        if (FactoryConexion.instancia == null){
            FactoryConexion.instancia=new FactoryConexion();
        }
        return FactoryConexion.instancia;
    }

    public Connection getConn() throws SQLException , AppDataException{
        try {
            if(conn==null || conn.isClosed()){
                conn = DriverManager.getConnection(
                    "jdbc:mysql://" + host + ":" + port + "/" + db + "?user=" + user + "&password=" + password);
            }
        } catch (SQLException e) {
            throw new AppDataException(e, "Error al conectar a la base de datos");
        }
        cantConn++;
        return conn;
    }

    public void releaseConn() throws SQLException{
        try {
            cantConn--;
            if(cantConn==0){
                conn.close();
            }
        } catch (SQLException e) {
            throw e;
        }
    }
}
```

DataDestino.Java

```
1 package data;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.Objects;
8
9 import javax.swing.JOptionPane;
10
11 import entities.*;
12 import util.AppDataException;
13 import util.NoDestinoException;
14
15 public class DataDestino {
16
17
18     public Destino getIdDestino(Destino d) throws AppDataException, SQLException {
19
20         Destino des = null;
21         PreparedStatement stmt=null;
22         ResultSet rs=null;
23         try {
24             stmt=FactoryConexion.getInstance().getConnection().prepareStatement("select idDestino, localidad, porcentajeAumento from Destino where idDestino=?");
25             stmt.setInt(1, d.getIdDestino());
26             rs=stmt.executeQuery();
27             if(rs!=null && rs.next()){
28                 double por = rs.getDouble("porcentajeAumento");
29                 if(Objects.equals(null, por)) {
30                     des = new Destino(d.getIdDestino());
31                     des.setLocalidad(rs.getString("localidad"));
32                 }else {
33                     des = new DestinoDirecto();
34                     des.setIdDestino(rs.getInt("idDestino"));
35                     des.setLocalidad(rs.getString("localidad"));
36                     ((DestinoDirecto) des).setPorcentajeAumento(rs.getDouble("porcentajeAumento"));
37                 }
38             }
39
40         } catch (SQLException e) {
41             throw new AppDataException(e, "Error al consultar destinos en la base da datos");
42         } finally{
43             try {
44                 if(rs!=null)rs.close();
45                 if(stmt!=null)stmt.close();
46                 FactoryConexion.getInstance().releaseConn();
47             } catch (SQLException e) {
48                 throw e;
49             }
50         }
51         return des;
52     }
53
54     public Destino getIdDestino(String nombre) throws AppDataException, SQLException, NoDestinoException {
55
56         Destino d=null;
57         PreparedStatement stmt=null;
58         ResultSet rs=null;
59         try {
60             stmt=FactoryConexion.getInstance().getConnection().prepareStatement("select idDestino, localidad from Destino where localidad=?");
61             stmt.setString(1, nombre);
62             rs=stmt.executeQuery();
63             if(rs!=null && rs.next()){
64
65                 d = new Destino();
66
67                 d.setIdDestino(rs.getInt("idDestino"));
68                 d.setLocalidad(rs.getString("localidad"));
69             }
70             if(!rs.first()) {
71                 throw new NoDestinoException("No se ha encontrado origen/destino");
72             }
73
74         } catch (SQLException e) {
75             throw new AppDataException(e, "Error al conectar a la base da datos");
76         } finally{
77             try {
```



```

77         try {
78             if(rs!=null)rs.close();
79             if(stmt!=null)stmt.close();
80             FactoryConexion.getInstancia().releaseConn();
81         } catch (SQLException e) {
82             throw e;
83         }
84     }
85     return d;
86 }
87 public void insert(Destino d) throws AppDataException, SQLException{
88     PreparedStatement st = null;
89     ResultSet rs = null;
90     try {
91         st = FactoryConexion.getInstancia().getConn().prepareStatement("select max(idDestino) as id from Destino");
92         rs = st.executeQuery();
93         rs.first();
94         int idDes = (Integer.parseInt(rs.getString("id")) + 1);
95         st.close();
96         st = null;
97         rs.close();
98     }
99
100     st = FactoryConexion.getInstancia().getConn().prepareStatement("insert into Destino (idDestino, localidad) values (?,?)");
101     st.setInt(1, idDes);
102     st.setString(2, d.getLocalidad());
103     st.executeUpdate();
104     JOptionPane.showMessageDialog(null, "Destino agregado correctamente");
105 } catch (SQLException e) {
106     JOptionPane.showMessageDialog(null, "Error al agregar al nuevo Destino");
107     throw new AppDataException(e, "Error al agregar al nuevo Destino");
108 } finally{
109     try {
110         if(rs!=null)rs.close();
111         if(st!=null)st.close();
112         FactoryConexion.getInstancia().releaseConn();
113     } catch (SQLException e) {
114         throw e;
115     }
116 }
117 }
118 public void insert(DestinoDirecto d) throws AppDataException, SQLException{
119     PreparedStatement st = null;
120     ResultSet rs = null;
121     try {
122         st = FactoryConexion.getInstancia().getConn().prepareStatement("select max(idDestino) as id from Destino");
123         rs = st.executeQuery();
124         rs.first();
125         int idDes = (Integer.parseInt(rs.getString("id")) + 1);
126         st.close();
127         st = null;
128         rs.close();
129     }
130
131     st = FactoryConexion.getInstancia().getConn().prepareStatement("insert into Destino (idDestino, localidad , porcentajeAumento) values (?,?,?)");
132     st.setInt(1, idDes);
133     st.setString(2, d.getLocalidad());
134     st.setDouble(3, d.getPorcentajeAumento());
135     st.executeUpdate();
136     JOptionPane.showMessageDialog(null, "Destino agregado correctamente");
137 } catch (SQLException e) {
138     JOptionPane.showMessageDialog(null, "Error al agregar al nuevo Destino");
139     throw new AppDataException(e, "Error al agregar al nuevo Destino");
140 } finally{
141     try {
142         if(rs!=null)rs.close();
143         if(st!=null)st.close();
144         FactoryConexion.getInstancia().releaseConn();
145     } catch (SQLException e) {
146         throw e;
147     }
148 }
149 }
150
151 public ArrayList<Destino> getAll() throws SQLException, AppDataException {
152     ArrayList<Destino> dd= new ArrayList<Destino>();
153

```

```

153 ArrayList<Destino> dd= new ArrayList<Destino>();
154 PreparedStatement stmt=null;
155 ResultSet rs=null;
156 try {
157     stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select * from Destino");
158     rs=stmt.executeQuery();
159     while(rs!=null && rs.next()){
160         double pAumento = rs.getDouble("porcentajeAumento");
161         if(pAumento != 0) {
162             DestinoDirecto d = new DestinoDirecto();
163             d.setPorcentajeAumento(pAumento);
164             d.setIdDestino(rs.getInt("idDestino"));
165             d.setLocalidad(rs.getString("localidad"));
166             dd.add(d);
167         } else {
168             Destino d = new Destino();
169             d.setIdDestino(rs.getInt("idDestino"));
170             d.setLocalidad(rs.getString("localidad"));
171             dd.add(d);
172         }
173     }
174     return dd;
175 } catch (SQLException e) {
176     throw e;
177 } finally{
178     try {
179         if(rs!=null)rs.close();
180         if(stmt!=null)stmt.close();
181         FactoryConexion.getInstancia().releaseConn();
182     } catch (SQLException e) {
183         throw e;
184     }
185 }
186 }
187 }
188 }
189 }
190 }

```

```

191 public void update(Destino eIDestino) throws Exception{
192     PreparedStatement stmt=null;
193     try {
194         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
195             "update Destino set "
196             + "localidad=?, porcentajeAumento=? where idDestino=?");
197         stmt.setString(1, eIDestino.getLocalidad());
198         if (eIDestino instanceof DestinoDirecto){
199             stmt.setDouble(2,((DestinoDirecto) eIDestino).getPorcentajeAumento());
200         } else {
201             stmt.setNull(2, java.sql.Types.DOUBLE);
202         }
203         stmt.setInt(3,eIDestino.getIdDestino());
204         stmt.executeUpdate();
205     } catch (Exception e) {
206         throw e;
207     } finally{
208         try {
209             if(stmt != null)stmt.close();
210             FactoryConexion.getInstancia().releaseConn();
211         } catch (SQLException e) {
212             throw e;
213         }
214     }
215 }
216
217 public void delete(Destino eIDestino) throws Exception{
218     PreparedStatement stmt = null;
219     try {
220         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
221             "delete from Destino WHERE idDestino=?");
222         stmt.setInt(1, eIDestino.getIdDestino());
223         stmt.executeUpdate();
224     } catch (Exception e) {
225         throw e;
226     }
227 } finally{
228     try {
229         if(stmt != null)stmt.close();
230         FactoryConexion.getInstancia().releaseConn();
231     } catch (SQLException e) {
232         throw e;
233     }
234 }
235 }
236 }
237 }
238 }

```

DataServicio.Java

```
1 package data;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7
8 import entities.*;
9 import util.AppDataException;
10 import util.NoServiceException;
11
12 public class DataServicio {
13
14
15     public double[] getRecaudacionPorMes() throws AppDataException, SQLException {
16
17         double[] datos = new double[7];
18         PreparedStatement stmt=null;
19         ResultSet rs=null;
20         try {
21             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("call getRecaudacionPorMes()");
22             // Devuelve "Total ingresos"
23             rs=stmt.executeQuery();
24             while(rs!=null && rs.next()) {
25                 int mes = rs.getInt("Mes");
26                 System.out.println("Mes " + mes + " ingreso" + rs.getDouble("ingresos"));
27                 if(mes == 12) {
28                     datos[0] = rs.getDouble("ingresos");
29                 }else {
30                     datos[mes] = rs.getDouble("ingresos");
31                 }
32             }
33         } catch (SQLException e) {
34             throw new AppDataException(e, "Error al conectar a la base da datos");
35         } finally{
36             try {
37                 if(rs!=null)rs.close();
38                 if(stmt!=null)stmt.close();
39                 FactoryConexion.getInstancia().releaseConn();
40             } catch (SQLException e) {
41                 throw e;
42             }
43         }
44         return datos;
45     }
46
47     public String[][] getDestinosByMesAño(int mes, int año) throws AppDataException, SQLException{
48
49         String[][] datos = new String[5][2];
50         PreparedStatement stmt=null;
51         ResultSet rs=null;
52         try {
53             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("call getDestinosByMesAño(?,?)");
54             // Devuelve columnas "localidad" y "totalPasajes"
55             System.out.println("El año es " + año);
56             stmt.setInt(1, 12);
57             stmt.setInt(2, año);
58             rs=stmt.executeQuery();
59             int rowcount = 0;
60             if (rs.last()) {
61                 rowcount = rs.getRow();
62                 rs.first(); // not rs.first() because the rs.next() below will move on, missing the first element
63             }
64             for(int i = 0; i < rowcount && i < 5; i++) {
65                 datos[i][0] = rs.getString("localidad");
66                 datos[i][1] = String.valueOf(rs.getInt("totalPasajes"));
67                 rs.next();
68             }
69         } catch (SQLException e) {
70             throw new AppDataException(e, "Error al conectar a la base da datos");
71         } finally{
72             try {
73                 if(rs!=null)rs.close();
74                 if(stmt!=null)stmt.close();
75                 FactoryConexion.getInstancia().releaseConn();
76             } catch (SQLException e) {
77                 throw e;
78             }
79         }
80         return datos;
81     }
82
83
84     public void addPasajero(int idSer, String dni, String patente, int numButaca, int origen, int destino, double precio) throws AppDataException {
85
86         PreparedStatement stmt=null;
87         try {
```

```

87     try {
88         stmt=FactoryConexion.getInstancia().getConn()
89             .prepareStatement(
90                 "insert into PersonaServicioMicro (idServicio, dniPersona, patenteMicro, numButaca, origen, destino, precio) values (?, ?, ?, ?, ?, ?, ?)"
91             );
92         stmt.setInt(1, idSer);
93         stmt.setString(2, dni);
94         stmt.setString(3, patente);
95         stmt.setInt(4, numButaca);
96         stmt.setInt(5, origen);
97         stmt.setInt(6, destino);
98         stmt.setDouble(7, precio);
99         stmt.executeUpdate();
100     } catch (SQLException e) {
101         throw new AppDataException(e, "Error ocurrido en el metodo addPasajero"
102             + "(int idSer, String dni, String patente, int numButaca, int origen, int destino, double precio) en la clase DataServicio al conectar a la base de datos");
103     }
104 }
105
106 public Servicio getServicioParaVenta(int id) throws Exception {
107     Servicio ser = this.getByid(id);
108     //No seria necesario, ya que al momento de vender, el usuario ya definio cual es el recorrido.
109     //Podria servir para calcular el precio a sea un destino normal o un destino directo.
110     ser.addDestinos(this.getDestinos(ser.getIdServicio()));
111     ser.addMicros(this.getMicros(ser));
112 }
113
114 return ser;
115
116 public ArrayList<Micro> getMicros(Servicio s) throws Exception{
117     ArrayList<Micro> mm=null;
118     Micro m = null;
119     PreparedStatement stmt=null;
120     ResultSet rs=null;
121     try {
122         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("")
123             + "select sm.patente, marca, \n" +
124             "porcentajeAumento, fechaUltimoControl, count(b.numButaca) as cantButacas\n" +
125             "from ServicioMicro sm\n" +
126             "inner join Micro m on sm.patente = m.patente\n" +
127             "inner join Butaca b on b.patenteMicro = m.patente\n" +
128             "where sm.idServicio = ?\n" +
129             "group by m.patente";
130         stmt.setInt(1, s.getIdServicio());
131         rs=stmt.executeQuery();
132
133         rs=stmt.executeQuery();
134         mm = new ArrayList<Micro>();
135         while(rs!=null && rs.next()){
136             double pAumento = rs.getDouble("porcentajeAumento");
137             if(pAumento != 0) {
138                 m = new MicroCama();
139                 ((MicroCama) m).setAumento(pAumento);
140             }
141             else {
142                 m = new Micro();
143             }
144             m.setMarca(rs.getString("marca"));
145             m.setPatente(rs.getString("patente"));
146             m.setFechaUltimoCtrl(rs.getDate("fechaUltimoControl"));
147             m.setCantidadButacas(rs.getInt("cantButacas"));
148             m.setConductores(this.getConductoresMicro(s, m));
149             m.setButacas(this.getButacasMicro(s, m));
150             mm.add(m);
151         }
152     } catch (SQLException e) {
153         throw new AppDataException(e, "Error al conectar a la base de datos");
154     } finally{
155         try {
156             if(rs!=null)rs.close();
157             if(stmt!=null)stmt.close();
158             FactoryConexion.getInstancia().releaseConn();
159         } catch (SQLException e) {
160             throw e;
161         }
162     }
163     return mm;
164 }
165
166 public Butaca[] getButacasMicro(Servicio s, Micro m) throws Exception {
167     Butaca[] pasajeros = new Butaca[m.getCantidadButacas()];
168     for(int i = 0; i < pasajeros.length; i++) {
169         pasajeros[i] = new Butaca(i + 1);
170     }
171     PreparedStatement stmt = null;
172     ResultSet rs = null;
173     try {
174         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select dniPersona, numButaca from PersonaServicioMicro where patenteMicro = ? and idServicio = ?");
175         stmt.setString(1, m.getPatente());
176         stmt.setInt(2, s.getIdServicio());

```



```

173         stmt.setInt(2, s.getIdServicio());
174         rs=stmt.executeQuery();
175         while(rs!=null && rs.next()){
176             int num = rs.getInt("numButaca");
177             Usuario u = new Usuario(rs.getString("dniPersona"));
178             pasajeros[num - 1].setPasajero(u);
179         }
180     } catch (SQLException e) {
181         throw new AppDataException(e, "Error al conectar a la base da datos");
182     } finally{
183         try {
184             if(rs!=null)rs.close();
185             if(stmt!=null)stmt.close();
186             FactoryConexion.getInstancia().releaseConn();
187         } catch (SQLException e) {
188             throw e;
189         }
190     }
191     return pasajeros;
192 }
193 }
194
195 private ArrayList<Conductor> getConductoresMicro(Servicio s, Micro m) throws AppDataException, SQLException {
196     ArrayList<Conductor> cc=null;
197     Conductor c = null;
198     PreparedStatement stmt=null;
199     ResultSet rs=null;
200     try {
201         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select distinct p.dni, p.nombre, p.apellido, "
202             + "p.tipoDni, p.fechaNac, p.fechaInicio, p.contacto\n" +
203             "from ServicioMicro sm\n" +
204             "inner join MicroConductor mc on sm.patente = mc.patente\n" +
205             "inner join Persona p on mc.dniConductor = p.dni\n" +
206             "where sm.idServicio = ? and sm.patente = ?");
207         stmt.setInt(1, s.getIdServicio());
208         stmt.setString(2, m.getPatente());
209         rs=stmt.executeQuery();
210         cc = new ArrayList<Conductor>();
211         while(rs!=null && rs.next()){
212             c = new Conductor();
213             c.setDni(rs.getString("dni"));
214             c.setNombre(rs.getString("nombre"));
215             c.setApellido(rs.getString("apellido"));
216             c.setTipoDni(rs.getString("tipoDni"));
217             c.setTipoDni(rs.getString("tipoDni"));
218             c.setFechaNacimiento(rs.getDate("fechaNac"));
219             c.setFechaInicio(rs.getDate("fechaInicio"));
220             c.setContacto(rs.getString("contacto"));
221             cc.add(c);
222         }
223     } catch (SQLException e) {
224         throw new AppDataException(e, "Error al conectar a la base da datos");
225     } finally{
226         try {
227             if(rs!=null)rs.close();
228             if(stmt!=null)stmt.close();
229             FactoryConexion.getInstancia().releaseConn();
230         } catch (SQLException e) {
231             throw e;
232         }
233     }
234     return cc;
235 }
236
237 }
238
239 public ArrayList<Destino> getDestinos(int idSer) throws AppDataException, SQLException{
240     ArrayList<Destino> dd=null;
241     Destino d = null;
242     PreparedStatement stmt=null;
243     ResultSet rs=null;
244     try {
245         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select sd.idDestino, localidad, precio, ordenDestinos, porcentajeAumento"
246             + " from ServicioDestino sd"
247             + " inner join Destino d on d.idDestino = sd.idDestino"
248             + " where sd.idServicio = ?");
249         stmt.setInt(1, idSer);
250         rs=stmt.executeQuery();
251         dd = new ArrayList<Destino>();
252         while(rs!=null && rs.next()){
253             double pAumento = rs.getDouble("porcentajeAumento");
254             if(pAumento != 0) {
255                 d = new DestinoDirecto();
256                 ((DestinoDirecto) d).setPorcentajeAumento(pAumento);
257             } else {
258                 d = new Destino();
259             }
260             d.setIdDestino(rs.getInt("idDestino"));

```

```

259         d.setIdDestino(rs.getInt("idDestino"));
260         d.setLocalidad(rs.getString("localidad"));
261         d.setOrdenDestino(rs.getInt("ordenDestinos"));
262         d.setPrecioDestino(rs.getDouble("precio"));
263         dd.add(d);
264     }
265
266     } catch (SQLException e) {
267         throw new AppDataException(e, "Error al conectar a la base da datos");
268     } finally{
269         try {
270             if(rs!=null)rs.close();
271             if(stmt!=null)stmt.close();
272             FactoryConexion.getInstancia().releaseConn();
273         } catch (SQLException e) {
274             throw e;
275         }
276     }
277     return dd;
278 }
279 public void addAll(Servicio s) throws AppDataException, SQLException{
280
281     s.setIdServicio(this.generateIdServicio());
282     this.insert(s);
283     this.insertAllDestinos(s.getDestinos(), s.getIdServicio());
284     this.insertAllMicros(s.getMicros(), s.getIdServicio());
285     this.insertAllConductores(s.getMicros(), s.getIdServicio());
286 }
287 public void insertAllConductores(ArrayList<Micro> mm, int idServicio) throws SQLException, AppDataException {
288
289     PreparedStatement stmt = null;
290
291     try {
292         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("
293             + "INSERT INTO MicroConductor (patente, dniConductor, idServicio) "
294             + "VALUES (?,?,?)");
295         for (Micro m : mm) {
296             for(Conductor c: m.getConductores()) {
297
298                 stmt.setString(1, m.getPatente());
299                 stmt.setString(2, c.getDni());
300                 stmt.setInt(3, idServicio);
301                 stmt.addBatch();
302             }
303         }
304         stmt.executeBatch();
305     } finally{
306         try {
307             if(stmt!=null)stmt.close();
308             FactoryConexion.getInstancia().releaseConn();
309         } catch (SQLException e) {
310             throw e;
311         }
312     }
313 }
314 public void insertAllMicros(ArrayList<Micro> mm, int idServicio) throws SQLException, AppDataException {
315
316     PreparedStatement stmt = null;
317
318     try {
319         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("
320             + "INSERT INTO ServicioMicro (idServicio, patente) "
321             + "VALUES (?,?)");
322         for (Micro m : mm) {
323             stmt.setInt(1, idServicio);
324             stmt.setString(2, m.getPatente());
325             stmt.addBatch();
326         }
327         stmt.executeBatch();
328     } finally{
329         try {
330             if(stmt!=null)stmt.close();
331             FactoryConexion.getInstancia().releaseConn();
332         } catch (SQLException e) {
333             throw e;
334         }
335     }
336 }
337 public int generateIdServicio() throws AppDataException, SQLException {
338     ResultSet rs = null;
339     PreparedStatement stmt=null;
340     try {
341         stmt = FactoryConexion.getInstancia().getConn().prepareStatement("select max(idServicio) as id from Servicio");
342         rs = stmt.executeQuery();
343         rs.first();
344         return (rs.getInt("id") + 1);
345     } catch (SQLException e) {

```

```

345     } catch (SQLException e) {
346         throw new AppDataException(e, "Error al conectar a la base da datos");
347     } finally{
348         try {
349             if(rs!=null)rs.close();
350             if(stmt!=null)stmt.close();
351             FactoryConexion.getInstancia().releaseConn();
352         } catch (SQLException e) {
353             throw e;
354         }
355     }
356 }
357 private static final String SQL_INSERT = "INSERT INTO ServicioDestino (idServicio, idDestino, precio, ordenDestinos) VALUES (?, ?, ?, ?)";
358 public void insertAllDestinos(ArrayList<Destino> dd, int idServicio) throws SQLException, AppDataException {
359     int i = 0;
360     PreparedStatement stmt = null;
361
362     try {
363         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(SQL_INSERT);
364         for (Destino d : dd) {
365             i++;
366             stmt.setInt(1, idServicio);
367             stmt.setInt(2, d.getIdDestino());
368             stmt.setDouble(3, d.getPrecioDestino());
369             stmt.setInt(4, i);
370             stmt.addBatch();
371         }
372         stmt.executeBatch();
373     } finally{
374         try {
375             if(stmt!=null)stmt.close();
376             FactoryConexion.getInstancia().releaseConn();
377         } catch (SQLException e) {
378             throw e;
379         }
380     }
381 }
382 }
383 public ArrayList<Servicio> getDetalles() throws AppDataException, SQLException {
384     ArrayList<Servicio> uu= new ArrayList<Servicio>();
385     PreparedStatement stmt=null;
386     ResultSet rs=null;
387     try {
388     try {
389         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select * from getDetallesServicios");
390         rs=stmt.executeQuery();
391         while(rs!=null && rs.next()){
392             Micro m = new Micro();
393             Conductor c = new Conductor();
394             Servicio s = new Servicio();
395
396             s.setIdServicio(rs.getInt("idServicio"));
397             s.setFechaServicio(rs.getDate("fechaServicio"));
398             s.setHoraServicio(rs.getString("HoraServicio"));
399             s.setRecorrido(rs.getString("Recorrido"));
400             //m.setMarca(rs.getString("marca"));
401             //m.setPatente(rs.getString("patente"));
402             //c.setNombre(rs.getString("nombresApellidos"));
403             //m.addConductor(c);
404             //s.addMicro(m);
405             uu.add(s);
406         }
407     }
408     catch (SQLException e) {
409         throw new AppDataException(e, "Error al conectar a la base da datos");
410     } finally{
411         try {
412             if(rs!=null)rs.close();
413             if(stmt!=null)stmt.close();
414             FactoryConexion.getInstancia().releaseConn();
415         } catch (SQLException e) {
416             throw e;
417         }
418     }
419     return uu;
420 }
421 }
422 public ArrayList<Servicio> getAllByDestinos(Destino origen, Destino destino) throws AppDataException, SQLException, NoServiceException {
423     ArrayList<Servicio> ss=null;
424     Servicio s = null;
425     PreparedStatement stmt=null;
426     ResultSet rs=null;
427     try {
428         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("call getServiciosByDestinos(?,?)");
429         stmt.setInt(1, origen.getIdDestino());
430         stmt.setInt(2, destino.getIdDestino());
431

```

```

431 stmt.setInt(2, destino.getIdDestino());
432 rs=stmt.executeQuery();
433 ss = new ArrayList<Servicio>();
434 int i = 0;
435
436 while(rs!=null && rs.next()){
437     i++;
438     s = new Servicio();
439
440     s.setIdServicio(rs.getInt("idServicio"));
441     s.setFechaServicio(rs.getDate("fechaServicio"));
442     s.setHoraServicio(rs.getString("horaServicio"));
443     ss.add(s);
444 }
445 if(!rs.first()) {
446     throw new NoServiceException("No se ha encontrado servicios para el origen y destino ingresado");
447 }
448
449 } catch (SQLException e) {
450     throw new AppDataException(e, "Error al conectar a la base da datos");
451 } finally{
452     try {
453         if(rs!=null)rs.close();
454         if(stmt!=null)stmt.close();
455         FactoryConexion.getInstancia().releaseConn();
456     } catch (SQLException e) {
457         throw e;
458     }
459 }
460 return ss;
461 }
462
463 public Servicio getById(int idServicio) throws SQLException, AppDataException, NoServiceException{
464     Servicio s=null;
465     PreparedStatement stmt=null;
466     ResultSet rs=null;
467     try {
468         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select * from Servicio where idServicio=?");
469         stmt.setInt(1, idServicio);
470         rs=stmt.executeQuery();
471         if(rs!=null && rs.next()){
472
473             s = new Servicio();
474
475             s.setIdServicio(rs.getInt("idServicio"));
476             s.setFechaServicio(rs.getDate("fechaServicio"));
477             s.setHoraServicio(rs.getString("horaServicio"));
478             s.addDestinos(this.getDestinos(s.getIdServicio()));
479             s.addMicros(this.getMicros(s));
480         }
481         if(!rs.first()) {
482             throw new NoServiceException("No se ha encontrado el servicio indicado");
483         }
484
485     } catch (SQLException e) {
486         throw new AppDataException(e, "Error al conectar a la base da datos");
487     } catch (Exception e) {
488         // TODO Auto-generated catch block
489         e.printStackTrace();
490     } finally{
491         try {
492             if(rs!=null)rs.close();
493             if(stmt!=null)stmt.close();
494             FactoryConexion.getInstancia().releaseConn();
495         } catch (SQLException e) {
496             throw e;
497         }
498     }
499     return s;
500 }
501
502 public void insert(Servicio s) throws AppDataException{
503     PreparedStatement stmt=null;
504     try {
505         /* lo borre, porque cuando tengo que agregar las Foreign Keys a las tablas ServicioDestino, ServicioMicro
506         * ya tengo que conocer cual es el id de servicio que agregue.
507         stmt = FactoryConexion.getInstancia().getConn().prepareStatement("select max(idServicio) as id from Servicio");
508         rs = stmt.executeQuery();
509         rs.first();
510         int idSer = ((rs.getInt("id")) + 1);
511         stmt.close();
512         stmt = null;
513         rs.close();
514         */
515         stmt=FactoryConexion.getInstancia().getConn()
516             .prepareStatement(
517                 "insert into Servicio(idServicio, fechaServicio, horaServicio) values (?,?,?)");
518         stmt.setInt(1, s.getIdServicio());

```



```

518         stmt.setInt(1, s.getIdServicio());
519         stmt.setDate(2, s.getFechaServicio());
520         stmt.setString(3, s.getHoraServicio());
521         stmt.executeUpdate();
522     } catch (SQLException e) {
523         throw new AppDataException(e, "Error ocurrido en el metodo insert(Servicio s) en la clase DataServicio al conectar a la base de datos");
524     }
525 }
526 public boolean getTieneRefuerzo(Servicio ser) throws AppDataException, SQLException {
527     boolean rta = false;
528     PreparedStatement stmt=null;
529     ResultSet rs=null;
530     try {
531         stmt=FactoryConexion.getInstance().getConn().prepareStatement("select getTieneRefuerzo(?) as ref");
532         stmt.setInt(1, ser.getIdServicio());
533         rs=stmt.executeQuery();
534         if(rs!=null && rs.next()){
535             //rta = rs.getBoolean(1);          ANDA BIEN
536             rta = rs.getBoolean("ref");
537         }
538     } catch (SQLException e) {
539         throw new AppDataException(e, "Error al conectar a la base de datos");
540     } finally{
541         try {
542             if(rs!=null)rs.close();
543             if(stmt!=null)stmt.close();
544             FactoryConexion.getInstance().releaseConn();
545         } catch (SQLException e) {
546             throw e;
547         }
548     }
549     return rta;
550 }
551 }
552 public void update(Servicio elServicio) throws Exception{
553     PreparedStatement stmt=null;
554     try {
555         stmt=FactoryConexion.getInstance().getConn().prepareStatement(
556             "update Servicio set "
557             + "fechaServicio=?, horaServicio=? where idServicio=?");
558         stmt.setDate(1, elServicio.getFechaServicio());
559         stmt.setString(2, elServicio.getHoraServicio());
560         stmt.setInt(3, elServicio.getIdServicio());
561         stmt.executeUpdate();
562     } catch (Exception e) {
563         throw e;
564     } finally{
565         try {
566             if(stmt != null)stmt.close();
567             FactoryConexion.getInstance().releaseConn();
568         } catch (SQLException e) {
569             throw e;
570         }
571     }
572 }
573
574
575 public void delete(Servicio elServicio) throws Exception{
576     PreparedStatement stmt = null;
577     try {
578         stmt=FactoryConexion.getInstance().getConn().prepareStatement(
579             "delete from Servicio WHERE idServicio=?");
580         stmt.setInt(1, elServicio.getIdServicio());
581         stmt.executeUpdate();
582     } catch (Exception e) {
583         throw e;
584     } finally{
585         try {
586             if(stmt != null)stmt.close();
587             FactoryConexion.getInstance().releaseConn();
588         } catch (SQLException e) {
589             throw e;
590         }
591     }
592 }
593
594 }
595

```

DataPersona.Java

```
1 package data;
2
3 import java.sql.PreparedStatement;
4
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8
9 import entities.*;
10 import util.AppDataException;
11
12 public class DataPersona {
13
14     public ArrayList<Usuario> getAllUsuarios() throws SQLException, AppDataException{
15         ArrayList<Usuario> uu= new ArrayList<Usuario>();
16         PreparedStatement stmt=null;
17         ResultSet rs=null;
18         try {
19             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select `dni`, `nombre`, `apellido`, `fechaNac`, `nombreUsuario`, `contrasena`, `email` "
20                 + "from Persona where esAdmin = 0 and nombreUsuario is not null");
21             rs=stmt.executeQuery();
22             while(rs!=null && rs.next()){
23
24                 Usuario u = new Usuario();
25                 u.setDni(rs.getString("dni"));
26                 u.setNombre(rs.getString("nombre"));
27                 u.setApellido(rs.getString("apellido"));
28                 u.setFechaNacimiento(rs.getDate("fechaNac"));
29                 u.setNombreUsuario(rs.getString("nombreUsuario"));
30                 u.setContrasena(rs.getString("contrasena"));
31                 u.setEmail(rs.getString("email"));
32                 uu.add(u);
33             }
34             return uu;
35
36         } catch (SQLException e) {
37             throw e;
38         } finally{
39             try {
40                 if(rs!=null)rs.close();
41                 if(stmt!=null)stmt.close();
42                 FactoryConexion.getInstancia().releaseConn();
43             } catch (SQLException e) {
44                 throw e;
45             }
46         }
47     }
48
49     public ArrayList<Conductor> getAllConductores() throws SQLException, AppDataException{
50         ArrayList<Conductor> uu= new ArrayList<Conductor>();
51         PreparedStatement stmt=null;
52         ResultSet rs=null;
53         try {
54             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select `dni`, `nombre`, `apellido`, `fechaNac`, `fechaInicio`, `contacto` "
55                 + "from Persona where contacto is not null");
56             rs=stmt.executeQuery();
57             while(rs!=null && rs.next()){
58
59                 Conductor c = new Conductor();
60                 c.setDni(rs.getString("dni"));
61                 c.setNombre(rs.getString("nombre"));
62                 c.setApellido(rs.getString("apellido"));
63                 c.setFechaNacimiento(rs.getDate("fechaNac"));
64                 c.setFechaInicio(rs.getDate("fechaInicio"));
65                 c.setContacto(rs.getString("contacto"));
66                 uu.add(c);
67             }
68             return uu;
69
70         } catch (SQLException e) {
71             throw e;
72         } finally{
73             try {
74                 if(rs!=null)rs.close();
75                 if(stmt!=null)stmt.close();
76                 FactoryConexion.getInstancia().releaseConn();
77             } catch (SQLException e) {
78                 throw e;
79             }
80         }
81     }
82
83
84
85     public Persona getByDni(String dni) throws SQLException, AppDataException{
86         Persona p=null;
87     }
```

```

87     Persona p=null;
88     PreparedStatement stmt=null;
89     ResultSet rs=null;
90     try {
91         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select * from Persona where dni=?");
92         stmt.setString(1, dni);
93         rs=stmt.executeQuery();
94         if(rs!=null && rs.next()){
95
96             if (rs.getString("fechaInicio") == null){
97                 p = new Usuario();
98                 ((Usuario) p).setNombreUsuario(rs.getString("nombreUsuario"));
99                 ((Usuario) p).setContraseña(rs.getString("contrasena"));
100                 ((Usuario) p).setEmail(rs.getString("email"));
101
102             } else {
103                 p = new Conductor();
104                 ((Conductor) p).setFechaInicio(rs.getDate("fechaInicio"));
105                 ((Conductor) p).setContacto(rs.getString("contacto"));
106             }
107
108             p.setDni(rs.getString("dni"));
109             p.setNombre(rs.getString("nombre"));
110             p.setApellido(rs.getString("apellido"));
111             p.setTipoDni(rs.getString("tipoDni"));
112             p.setFechaNacimiento(rs.getDate("fechaNac"));
113         }
114
115     } catch (SQLException e) {
116         throw e;
117     } finally{
118         try {
119             if(rs!=null)rs.close();
120             if(stmt!=null)stmt.close();
121             FactoryConexion.getInstancia().releaseConn();
122         } catch (SQLException e) {
123             throw e;
124         }
125     }
126     return p;
127 }
128
131 public void add(Usuario per) throws SQLException, AppDataException{
132
133     PreparedStatement stmt=null;
134     try {
135         stmt=FactoryConexion.getInstancia().getConn()
136         .prepareStatement(
137             "insert into Persona(dni, nombre, apellido, tipoDni, fechaNac, nombreUsuario, contraseña, email, esAdmin) values (?, ?, ?, ?, ?, ?, ?, ?, ?)"
138         );
139         stmt.setString(1, per.getDni());
140         stmt.setString(2, per.getNombre());
141         stmt.setString(3, per.getApellido());
142         stmt.setString(4, per.getTipoDni());
143         stmt.setDate(5, per.getFechaNacimiento());
144         stmt.setString(6, per.getNombreUsuario());
145         stmt.setString(7, per.getContraseña());
146         stmt.setString(8, per.getEmail());
147         stmt.setBoolean(9, ((Persona) per).getEsAdmin());
148
149         stmt.executeUpdate();
150     } catch (SQLException e) {
151         throw e;
152     }
153 }
154
155 public void add(Conductor per) throws SQLException, AppDataException{
156
157     PreparedStatement stmt=null;
158     try {
159         stmt=FactoryConexion.getInstancia().getConn()
160         .prepareStatement(
161             "insert into Persona(dni, nombre, apellido, tipoDni, fechaNac, fechaInicio, contacto, esAdmin) values (?, ?, ?, ?, ?, ?, ?, ?)"
162         );
163         stmt.setString(1, per.getDni());
164         stmt.setString(2, per.getNombre());
165         stmt.setString(3, per.getApellido());
166         stmt.setString(4, per.getTipoDni());
167         stmt.setDate(5, per.getFechaNacimiento());
168         stmt.setDate(6, per.getFechaInicio());
169         stmt.setString(7, per.getContacto());
170         stmt.executeUpdate();
171     } catch (SQLException e) {
172         throw new AppDataException(e, "Error en al conexion a la base de datos");
173     }
174 }

```

```

174 }
175
176 public Usuario getLoggedUser(Usuario per) throws Exception{
177     Usuario u = null;
178     PreparedStatement stmt = null;
179     ResultSet rs = null;
180     try {
181         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
182             "select dni, nombre, apellido, tipoDni, fechaNac, email, esAdmin from Persona where nombreUsuario=? and contrasena=?");
183         stmt.setString(1, per.getNombreUsuario());
184         stmt.setString(2, per.getContrasena());
185         rs=stmt.executeQuery();
186         if(rs!=null && rs.next()){
187             u = new Usuario();
188             u.setDni(rs.getString("dni"));
189             u.setNombre(rs.getString("nombre"));
190             u.setApellido(rs.getString("apellido"));
191             u.setTipoDni(rs.getString("tipoDni"));
192             u.setFechaNacimiento(rs.getDate("fechaNac"));
193             u.setEmail(rs.getString("email"));
194             u.setEsAdmin(rs.getBoolean("esAdmin"));
195         }
196     } catch (Exception e) {
197         throw e;
198     } finally{
199         try {
200             if(rs != null)rs.close();
201             if(stmt != null)stmt.close();/*
202             FactoryConexion.getInstancia().releaseConn();*/
203         } catch (SQLException e) {
204             throw e;
205         }
206     }
207 }
208
209 return u;
210 }
211 public void update(Persona laPersona) throws Exception{
212     PreparedStatement stmt=null;
213     try {
214         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
215             "update Persona set "
216             + "nombre=?, apellido=?, fechaNac=?, esAdmin=?, nombreUsuario=?, contrasena=?, email=?, fechaInicio=?, contacto=? "
217             + "where dni=?");
218         stmt.setString(1, laPersona.getNombre());
219         stmt.setString(2, laPersona.getApellido());
220         stmt.setDate(3, laPersona.getFechaNacimiento());
221         stmt.setBoolean(4, laPersona.getEsAdmin());
222
223         if (laPersona instanceof Usuario){
224             stmt.setString(5,((Usuario) laPersona).getNombreUsuario());
225             stmt.setString(6,((Usuario) laPersona).getContrasena());
226             stmt.setString(7,((Usuario) laPersona).getEmail());
227             stmt.setDate(8, null);
228             stmt.setString(9,null);
229
230         } else {
231             stmt.setString(5,null);
232             stmt.setString(6,null);
233             stmt.setString(7,null);
234             stmt.setDate(8,((Conductor) laPersona).getFechaInicio());
235             stmt.setString(9,((Conductor) laPersona).getContacto());
236         }
237         stmt.setString(10, laPersona.getDni());
238         stmt.executeUpdate();
239     } catch (Exception e) {
240         throw e;
241     } finally{
242         try {
243             if(stmt != null)stmt.close();
244             FactoryConexion.getInstancia().releaseConn();
245         } catch (SQLException e) {
246             throw e;
247         }
248     }
249 }
250
251 public void delete(Persona laPersona) throws Exception{
252     PreparedStatement stmt = null;
253     //No tengo que dejar que borre las cuentas propia.
254     try {
255         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
256             "delete from Persona WHERE dni=?");
257         stmt.setString(1, laPersona.getDni());
258         stmt.executeUpdate();
259     } catch (Exception e) {
260         throw e;

```



```

261         } finally{
262             try {
263                 if(stmt != null)stmt.close();
264                 FactoryConexion.getInstancia().releaseConn();
265             } catch (SQLException e) {
266                 throw e;
267             }
268         }
269     }
270
271 }
272

```

DataMicro.Java

```

1 package data;|
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.Objects;
8 import javax.swing.JOptionPane;
9 import entities.*;
10 import util.AppDataException;
11
12 public class DataMicro {
13
14     public boolean esCama(String p) throws SQLException, AppDataException {
15
16         PreparedStatement stmt=null;
17         ResultSet rs=null;
18         try {
19             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select porcentajeAumento from Micro where patente=?");
20             stmt.setString(1, p);
21             rs=stmt.executeQuery();
22             rs.next();
23             return (rs.getDouble("porcentajeAumento") > 0);
24         } catch (SQLException e) {
25             throw new AppDataException(e, "Error al consultar destinos en la base da datos");
26         } finally{
27             try {
28                 if(rs!=null)rs.close();
29                 if(stmt!=null)stmt.close();
30                 FactoryConexion.getInstancia().releaseConn();
31             } catch (SQLException e) {
32                 throw e;
33             }
34         }
35     }
36     public ArrayList<Micro> getAll() throws SQLException, AppDataException{
37
38         ArrayList<Micro> mm= new ArrayList<Micro>();
39         PreparedStatement stmt=null;
40         ResultSet rs=null;
41         try {
42             stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select * from Micro");
43             rs=stmt.executeQuery();
44             while(rs!=null && rs.next()){

```

```

44 while(rs!=null && rs.next()){
45     double pAumento = rs.getDouble("porcentajeAumento");
46     if(pAumento != 0) {
47         MicroCama m = new MicroCama();
48         m.setAumento(pAumento);
49         m.setMarca(rs.getString("marca"));
50         m.setPatente(rs.getString("patente"));
51         m.setFechaUltimoCtrl(rs.getDate("fechaUltimoControl"));
52         getCantidadServicios(m);
53         mm.add(m);
54     } else {
55         Micro m = new Micro();
56         m.setMarca(rs.getString("marca"));
57         m.setPatente(rs.getString("patente"));
58         m.setFechaUltimoCtrl(rs.getDate("fechaUltimoControl"));
59         getCantidadServicios(m);
60         mm.add(m);
61     }
62 }
63 return mm;
64 } catch (SQLException e) {
65     throw e;
66 } finally{
67     try {
68         if(rs!=null)rs.close();
69         if(stmt!=null)stmt.close();
70         FactoryConexion.getInstancia().releaseConn();
71     } catch (SQLException e) {
72         throw e;
73     }
74 }
75 }
76
77 public Micro getByPatente(Micro mic) throws AppDataException, SQLException{
78     Micro m = null;
79     PreparedStatement stmt=null;
80     ResultSet rs=null;
81     try {
82         stmt=FactoryConexion.getInstancia().getConn().prepareStatement("select patente, marca, fechaUltimoControl, porcentajeAumento from Micro where patente=?");
83         stmt.setString(1, mic.getPatente());
84         rs=stmt.executeQuery();
85         if(rs!=null && rs.next()){
86             double por = rs.getDouble("porcentajeAumento");
87             if(Objects.equals(null,por)){
88                 m = new Micro();
89                 m.setPatente(rs.getString("patente"));
90                 m.setMarca(rs.getString("marca"));
91                 m.setFechaUltimoCtrl(rs.getDate("fechaUltimoControl"));
92             } else {
93                 m = new MicroCama();
94                 m.setPatente(rs.getString("patente"));
95                 m.setMarca(rs.getString("marca"));
96                 m.setFechaUltimoCtrl(rs.getDate("fechaUltimoControl"));
97                 ((MicroCama) m).setAumento(rs.getDouble("porcentajeAumento"));
98             }
99         }
100     } catch (SQLException e) {
101         throw new AppDataException(e, "Error al conectar con la BD producido en el metodo getByPatente(Micro m)");
102     } finally{
103         try {
104             if(rs!=null)rs.close();
105             if(stmt!=null)stmt.close();
106             FactoryConexion.getInstancia().releaseConn();
107         } catch (SQLException e) {
108             throw e;
109         }
110     }
111     return m;
112 }
113
114 public void insert(MicroCama m) throws AppDataException, SQLException{
115     PreparedStatement st = null;
116     ResultSet rs = null;
117     try {
118         st = FactoryConexion.getInstancia().getConn().prepareStatement("insert into Micro (patente, marca, fechaUltimoControl, porcentajeAumento) values (?, ?, ?, ?)");
119         st.setString(1, m.getPatente());
120         st.setString(2, m.getMarca());
121         st.setDate(3, m.getFechaUltimoCtrl());
122         st.setDouble(4, m.getAumento());
123         st.executeUpdate();
124         JOptionPane.showMessageDialog(null, "Micro agregado correctamente");
125     } catch (SQLException e) {
126         JOptionPane.showMessageDialog(null, "Error al agregar el nuevo Micro");
127         throw new AppDataException(e, "Error al agregar el nuevo Micro");
128     } finally{
129         try {
130             if(rs!=null)rs.close();

```

```

130         if(rs!=null)rs.close();
131         if(st!=null)st.close();
132         FactoryConexion.getInstancia().releaseConn();
133     } catch (SQLException e) {
134         throw e;
135     }
136 }
137 }
138
139 public void insert(Micro m) throws AppDataException, SQLException{
140     PreparedStatement st = null;
141     ResultSet rs = null;
142     try {
143         st = FactoryConexion.getInstancia().getConn().prepareStatement("insert into Micro (patente, marca, fechaUltimoControl) values (?,?,?)");
144         st.setString(1, m.getPatente());
145         st.setString(2, m.getMarca());
146         st.setDate(3, m.getFechaUltimoCtrl());
147         st.executeUpdate();
148         JOptionPane.showMessageDialog(null, "Micro agregado correctamente");
149     } catch (SQLException e) {
150         JOptionPane.showMessageDialog(null, "Error al agregar el nuevo Micro");
151         throw new AppDataException(e, "Error al agregar al nuevo Micro");
152     } finally{
153         try {
154             if(rs!=null)rs.close();
155             if(st!=null)st.close();
156             FactoryConexion.getInstancia().releaseConn();
157         } catch (SQLException e) {
158             throw e;
159         }
160     }
161 }
162
163 public void update(Micro m) throws Exception{
164     PreparedStatement stmt=null;
165     try {
166         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
167             "update Micro set "
168             + "marca=?, fechaUltimoControl=?, porcentajeAumento=? where patente=?";
169         stmt.setString(1, m.getMarca());
170         stmt.setDate(2, m.getFechaUltimoCtrl());
171
172         if (m instanceof MicroCama){
173             stmt.setDouble(3, ((MicroCama) m).getAumento());
174
175             stmt.setDouble(3, ((MicroCama) m).getAumento());
176         } else {
177             stmt.setNull(3, java.sql.Types.DOUBLE);
178         }
179         stmt.setString(4, m.getPatente());
180         stmt.executeUpdate();
181     } catch (Exception e) {
182         throw e;
183     } finally{
184         try {
185             if(stmt != null)stmt.close();
186             FactoryConexion.getInstancia().releaseConn();
187         } catch (SQLException e) {
188             throw e;
189         }
190     }
191 }
192 public void delete(Micro m) throws Exception{
193     PreparedStatement stmt = null;
194     try {
195         stmt=FactoryConexion.getInstancia().getConn().prepareStatement(
196             "delete from Micro WHERE patente=?";
197         stmt.setString(1, m.getPatente());
198         stmt.executeUpdate();
199     } catch (Exception e) {
200         throw e;
201     } finally{
202         try {
203             if(stmt != null)stmt.close();
204             FactoryConexion.getInstancia().releaseConn();
205         } catch (SQLException e) {
206             throw e;
207         }
208     }
209 }
210
211 public void getCantidadServicios(Micro m) throws AppDataException, SQLException {
212     PreparedStatement st = null;
213     ResultSet rs = null;
214     try {
215         st = FactoryConexion.getInstancia().getConn().prepareStatement("select count(patente) cantidad from ServicioMicro WHERE patente = ?;");
216         st.setString(1, m.getPatente());

```

```

216         st.setString(1, m.getPatente());
217         rs=st.executeQuery();
218         if(rs!=null && rs.next()){
219             m.setCantidadServicios(rs.getInt("cantidad"));
220         }
221         else {
222             m.setCantidadServicios(0);
223         }
224     }catch (SQLException e) {
225         throw new AppDataException(e, "Error al consultar la cantidad de servicios por micro");
226     }finally{
227         try {
228             if(rs!=null)rs.close();
229             if(st!=null)st.close();
230             FactoryConexion.getInstancia().releaseConn();
231         } catch (SQLException e) {
232             throw e;
233         }
234     }
235 }
236 }
237 }

```