Universidad Tecnológica Nacional - FRRo

Cátedra: Lenguaje de programación JAVA

Ingeniería en Sistemas de Información

Trabajo Práctico Integrador

Gestión de Pasajes

Integrantes:

Legajo	Nombre y apellido	Email
49105	Bianchini, Tomás	tomassbianchini@gmail.com
46826	Saludas, Fausto Jesús	fausaludas14@gmail.com

Docentes:

- Ricardo Tabacman
- Adrián Meca

Índice

CUU - Comprar pasaje

Código y nombre del CU: CUU1-Comprar Pasaje

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

Meta del CU: Comprar Pasaje de Vuelo.
Actores:

Primario: Usuario

Precondiciones de sistema:

- Los usuarios y los vuelos están registrados en el sistema y cuentan con sus respectivos datos ya cargados, necesarios para realizar la compra.

Precondiciones de negocio:<omitido>

Camino Basico:

1. El usuario inicia sesión en su cuenta. El sistema valida los datos de usuario y lo loguea al sistema.
2. El usuario ingresa a la sección de vuelos. El sistema muestra un listado con todas los vuelos disponibles.
3. El cliente ingresa a la sección “comprar”. El sistema muestra un listado con todos los asientos disponibles, con su respectivo precio.
4. El usuario selecciona un asiento a comprar. El sistema valida los datos ingresados
5. El usuario genera la compra del pasaje. El sistema genera el comprobante del pasaje y envía un mail con el comprobante al usuario.

Camino Alternativo:

1.a El usuario no está registrado:
    1.a.1 Usuario se loguea. Sistema valida y registra los datos.
2.a No hay vuelos disponibles:
    2.a.1. Sistema informa. FCU
3.a. No hay asientos disponibles:
    3.a.1 Sistema informa al usuario. FCU.

Postcondiciones de negocio:

Éxito: El cliente obtuvo el pasaje.
Éxito alternativo: El cliente se registró y obtuvo el pasaje.
Fracaso: El cliente no pudo realizar la compra

Postcondiciones de sistema:

Éxito: Se registró correctamente la compra
Éxito alternativo: Se registró el usuario y la compra.
Fracaso: No se registró correctamente la compra

CUU - Cancelar Pasaje

Código y nombre del CU: CUU2-Cancelar Pasaje

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

Meta del CU: Cancelar Pasaje de Vuelo
Actores:

Primario: Usuario

Otros: <omitido>

Precondiciones de sistema:

- Los usuarios y los vuelos están registrados en el sistema y cuentan con sus respectivos datos ya cargados.
- El usuario cuenta con un pasaje comprado.
- El usuario está logueado.

Precondiciones de negocio:<omitido>

Camino Basico:

1. El usuario ingresa a la sección mis pasajes. El sistema muestra un listado con todas los pasajes del usuario.
2. El cliente selecciona la sección “cancelar pasaje”. El sistema cancela el pasaje.

Camino Alternativo:

2.a Faltan menos de 6 horas para la salida del vuelo:

2.a.1 Sistema informa. FCU.

Postcondiciones de negocio:

Éxito: El cliente realizó la cancelación del pasaje.
Éxito alternativo:<omitido>
Fracaso: El cliente no pudo realizar la cancelación del pasaje.

Postcondiciones de sistema:

Éxito: Se registró correctamente la cancelación del pasaje.
Éxito alternativo: <**omitido**>
Fracaso: No se registró la cancelación del pasaje.

CUU - Finalizar Pasaje

Código y nombre del CU: CUU2-Finalizar Pasaje

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

Meta del CU: Finalizar Pasaje de Vuelo.
Actores:

Primario: Administrador

Otros: <omitido>

Precondiciones de sistema:

- El usuario cuenta con un pasaje comprado.

Precondiciones de negocio:<omitido>

Camino Basico:

1. El administrador escanea el código QR. El sistema muestra el pasaje.
2. El administrador selecciona “finalizar pasaje”. El sistema finaliza el pasaje.

Camino Alternativo:

2.a Faltan más de 2 horas para la salida del vuelo:
    2.a.1 Sistema informa. FCU.
2.b Faltan menos de 15 minutos para la salida del vuelo:
    2.b.1 Sistema informa. FCU.

Postcondiciones de sistema:

Éxito: Se registró correctamente la finalización del pasaje.
Éxito alternativo: **<omitido>**
Fracaso: No se registró la finalización del pasaje.

CUR - Gestión de Pasajes

Código y nombre del CU: CUR1-Gestión de Pasajes

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Reestructurar	Sistema	Negra	Real	Semántica

Meta del CU: Realizar un vuelo.
Actores:

Primario: Usuario

Otros: <omitido>

Precondiciones de sistema:

- Los usuarios y los vuelos están registrados en el sistema y cuentan con sus respectivos datos ya cargados.
- El usuario está logueado.

Camino Basico:

1. El usuario elige un pasaje y selecciona “comprar” invocando **<CUU1-Comprar Pasaje.>**
2. El administrador finaliza el pasaje invocando **<CUU3-Finalizar Pasaje.>**

Camino Alternativo:

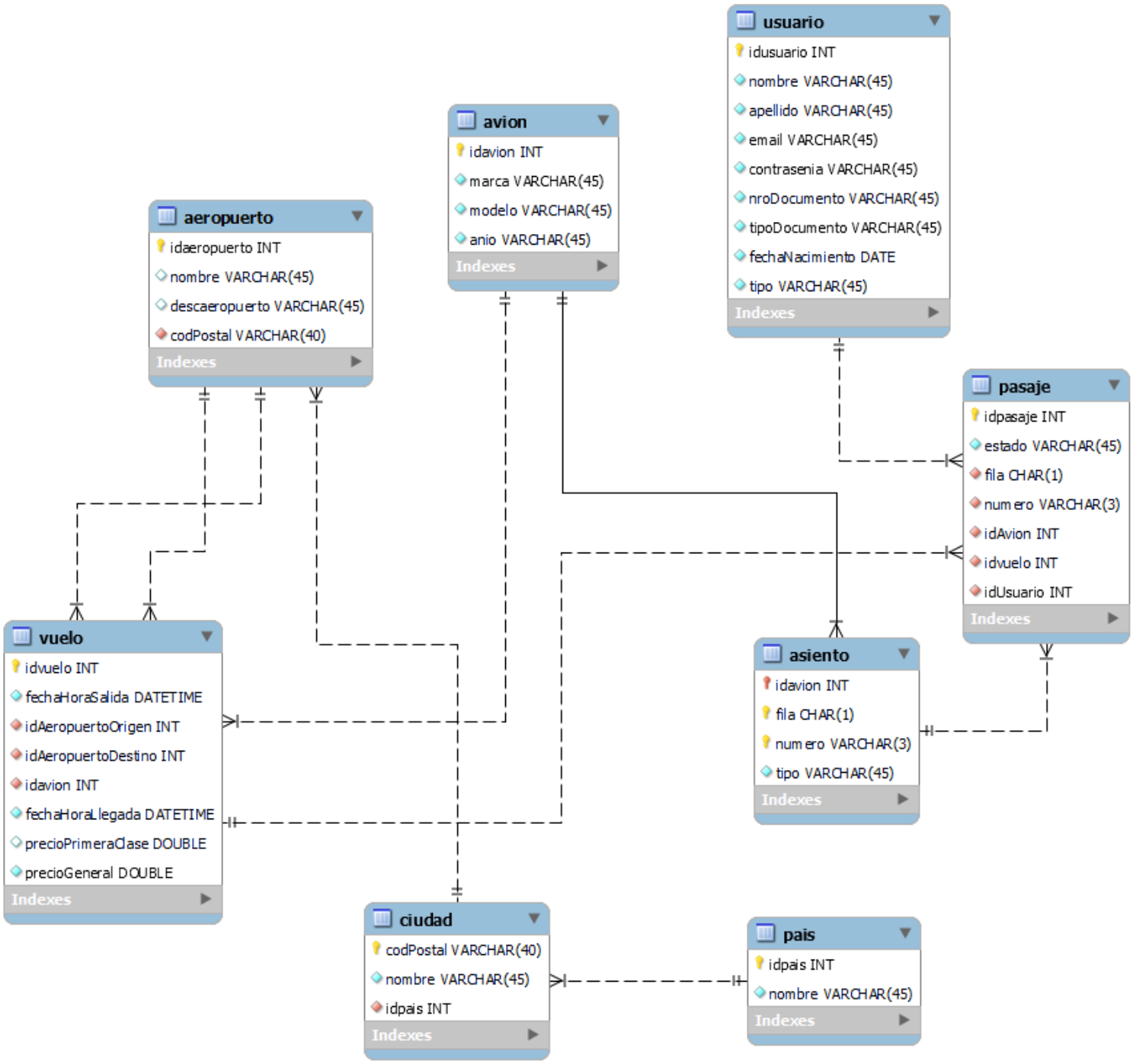
1.a<posterior> El usuario quiere cancelar el Pasaje

1.a.1 El usuario selecciona cancelar invocando **<CUU2-Cancelar Pasaje.>**

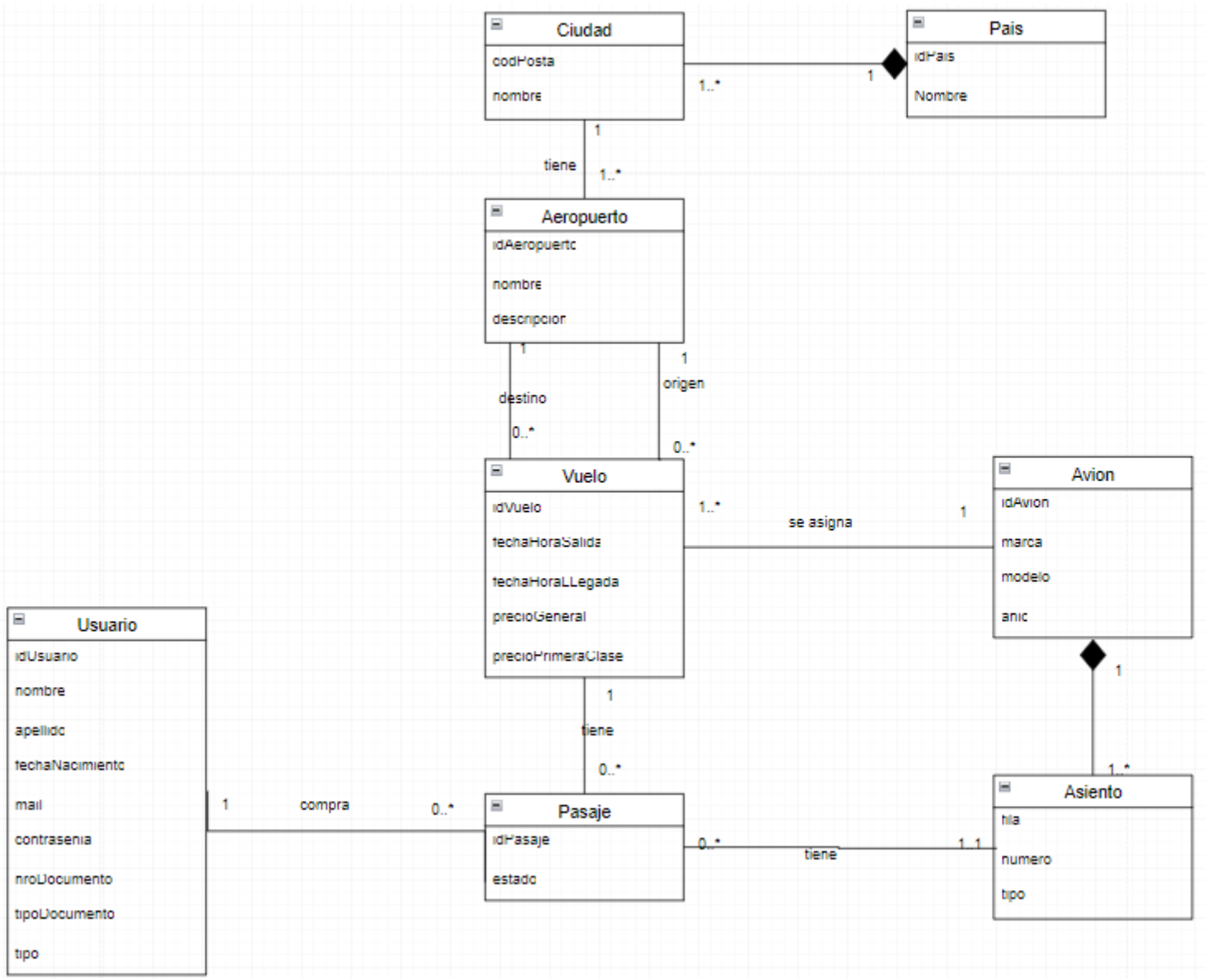
Postcondiciones de sistema:

Éxito: Se registró correctamente la finalización del Pasaje.
Éxito alternativo: Se registró la cancelación del Pasaje.

Modelo de datos

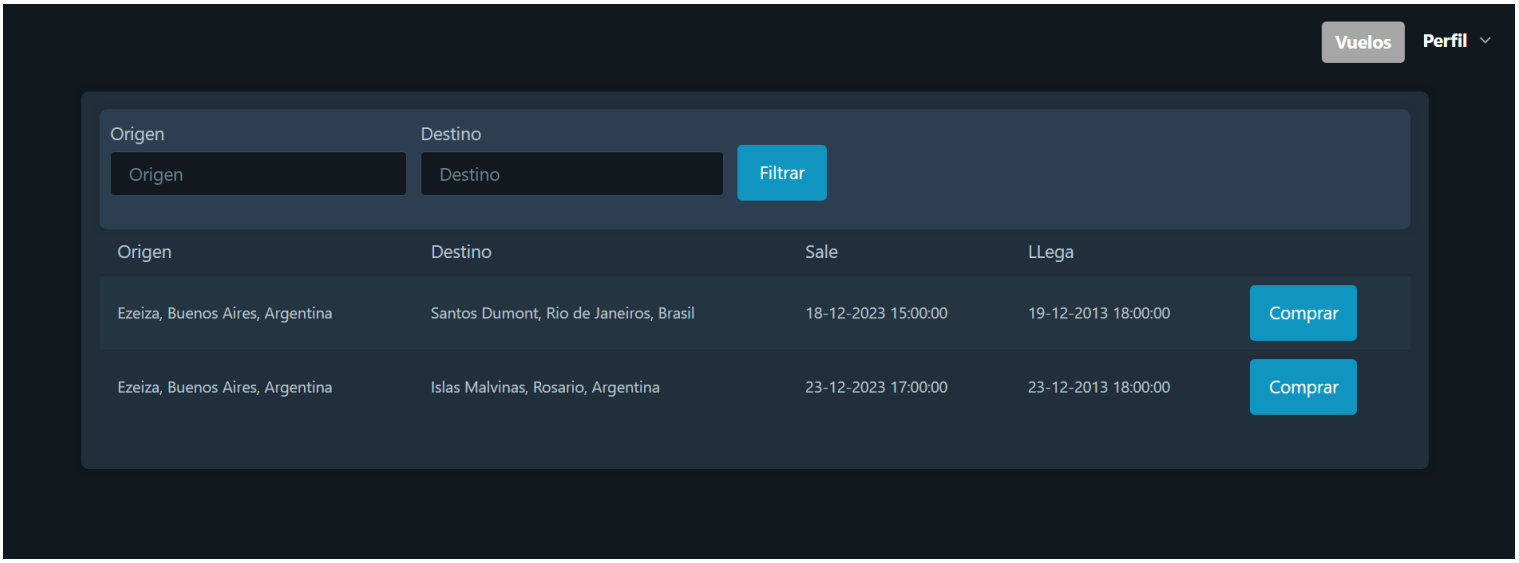


Modelo de Dominio

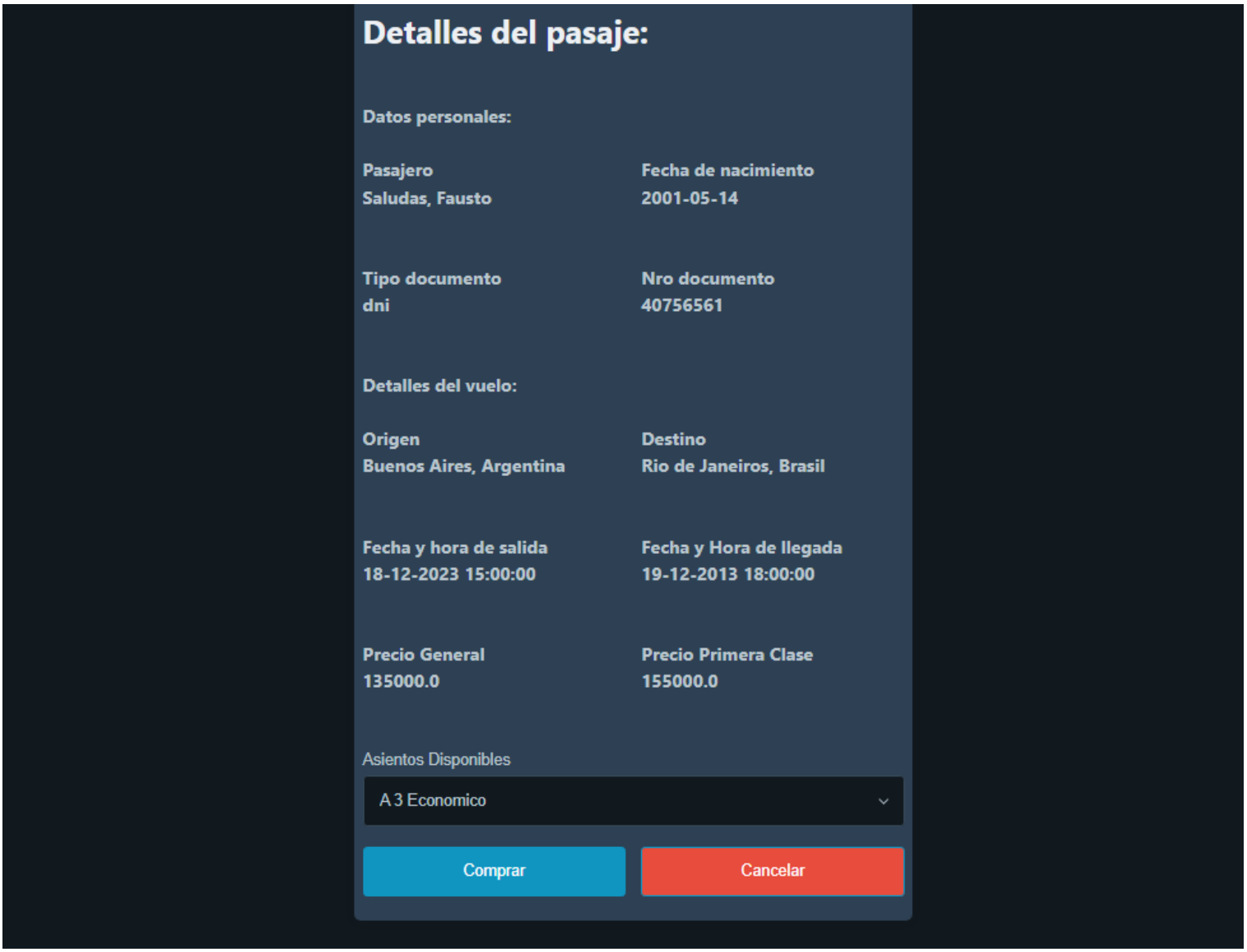


Capturas de pantalla del sistema:

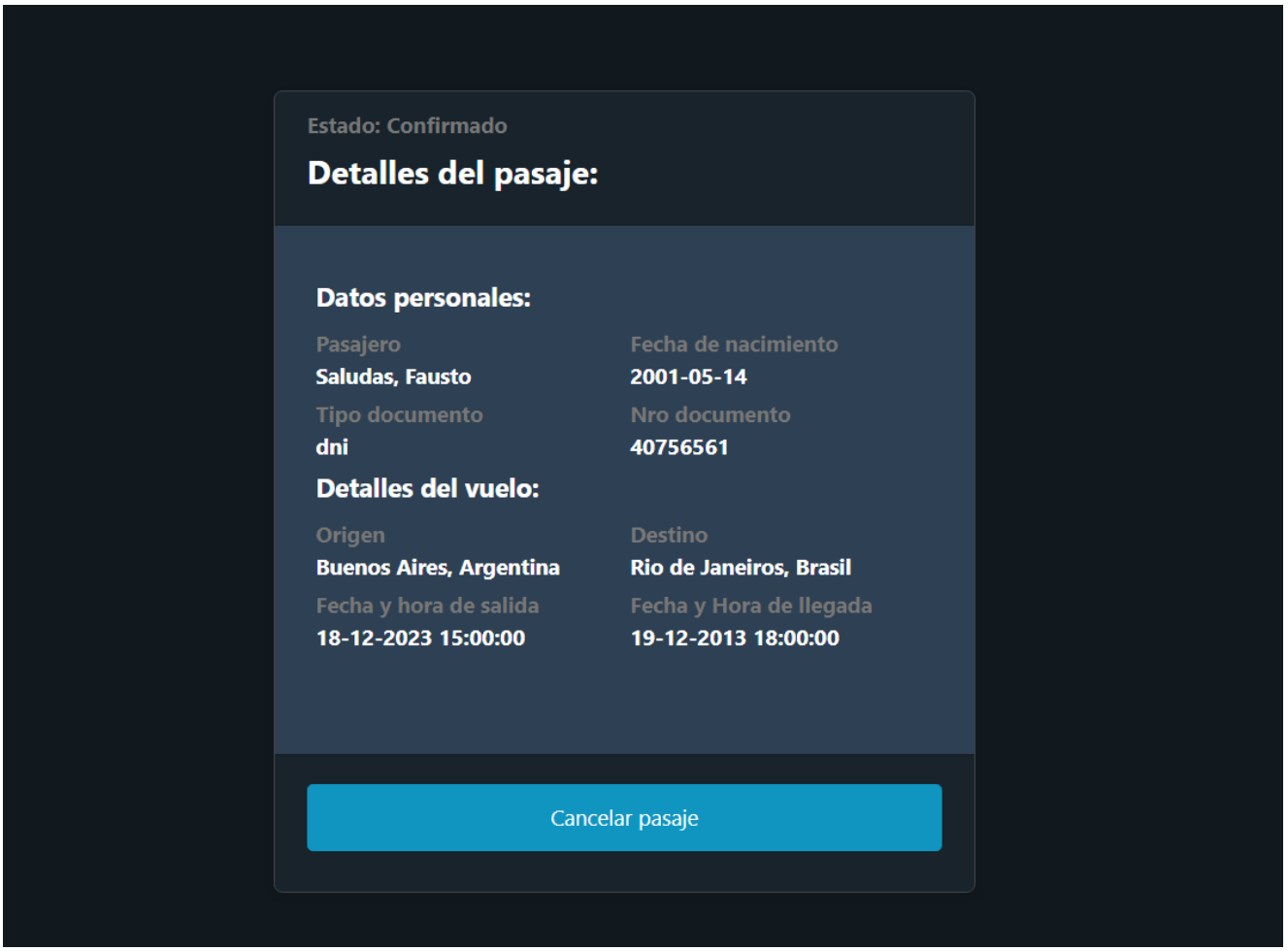
Lista de vuelos disponibles:



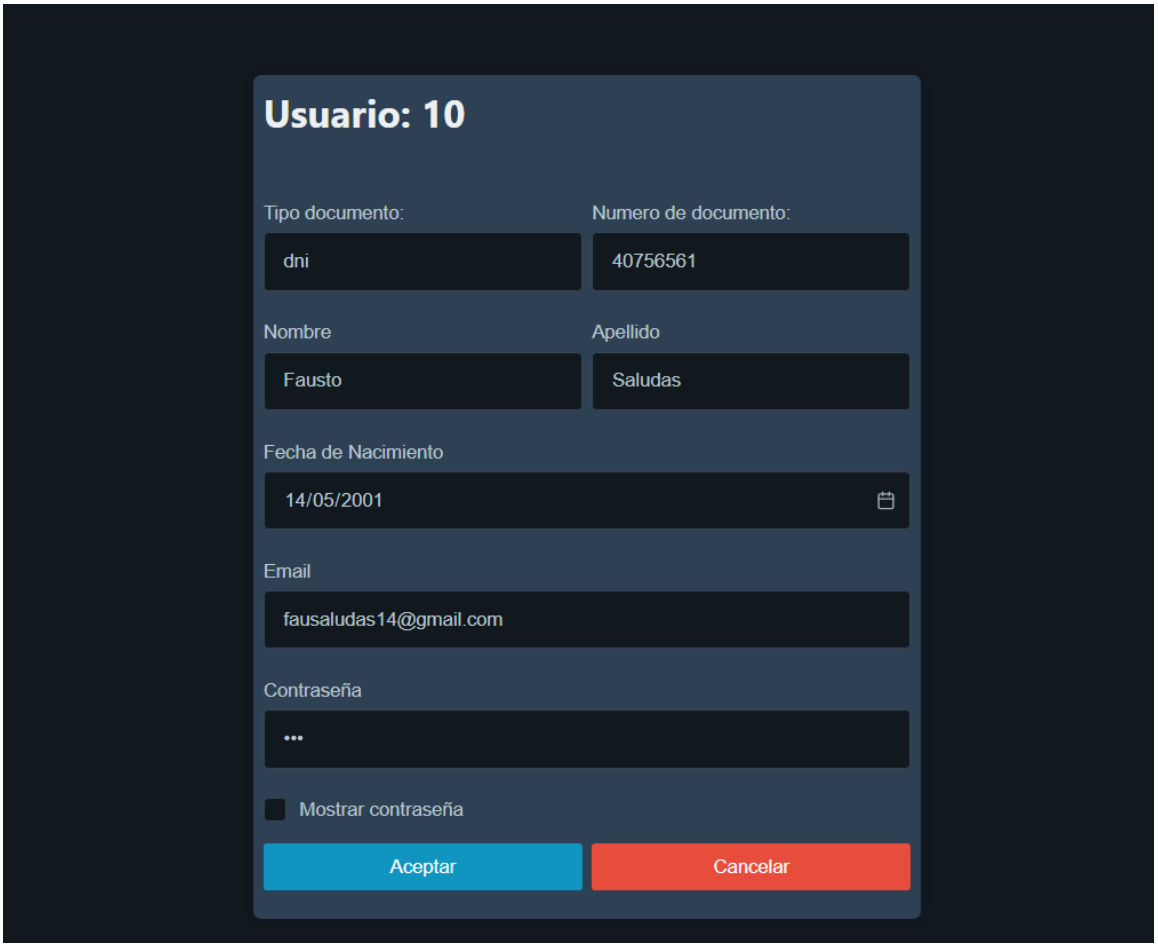
Detalles del vuelo seleccionado con los asientos disponibles:



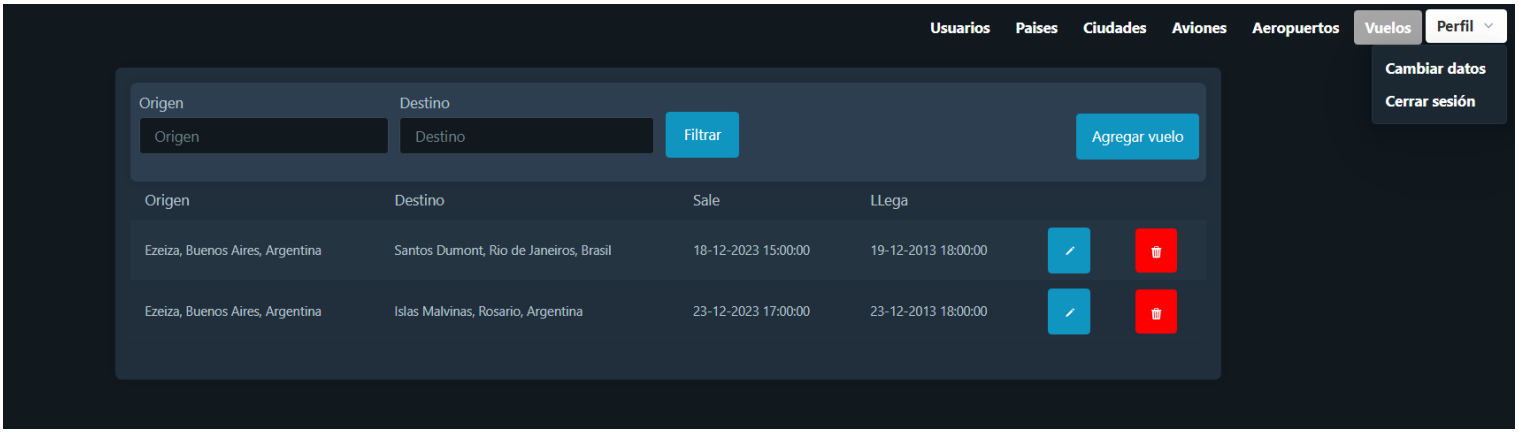
Pasajes confirmados y cancelados de cada usuario:



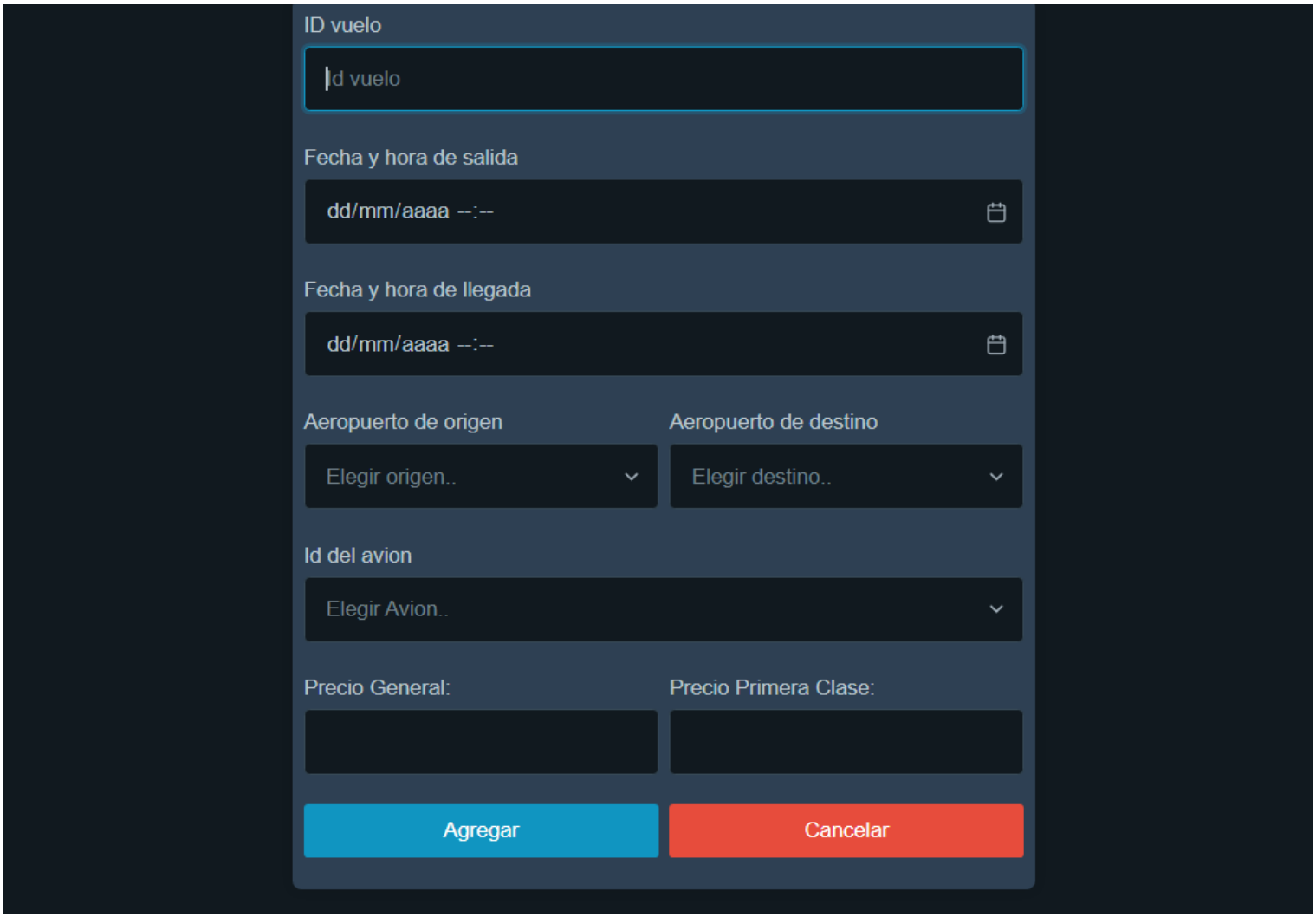
Cambiar datos de cada usuario :



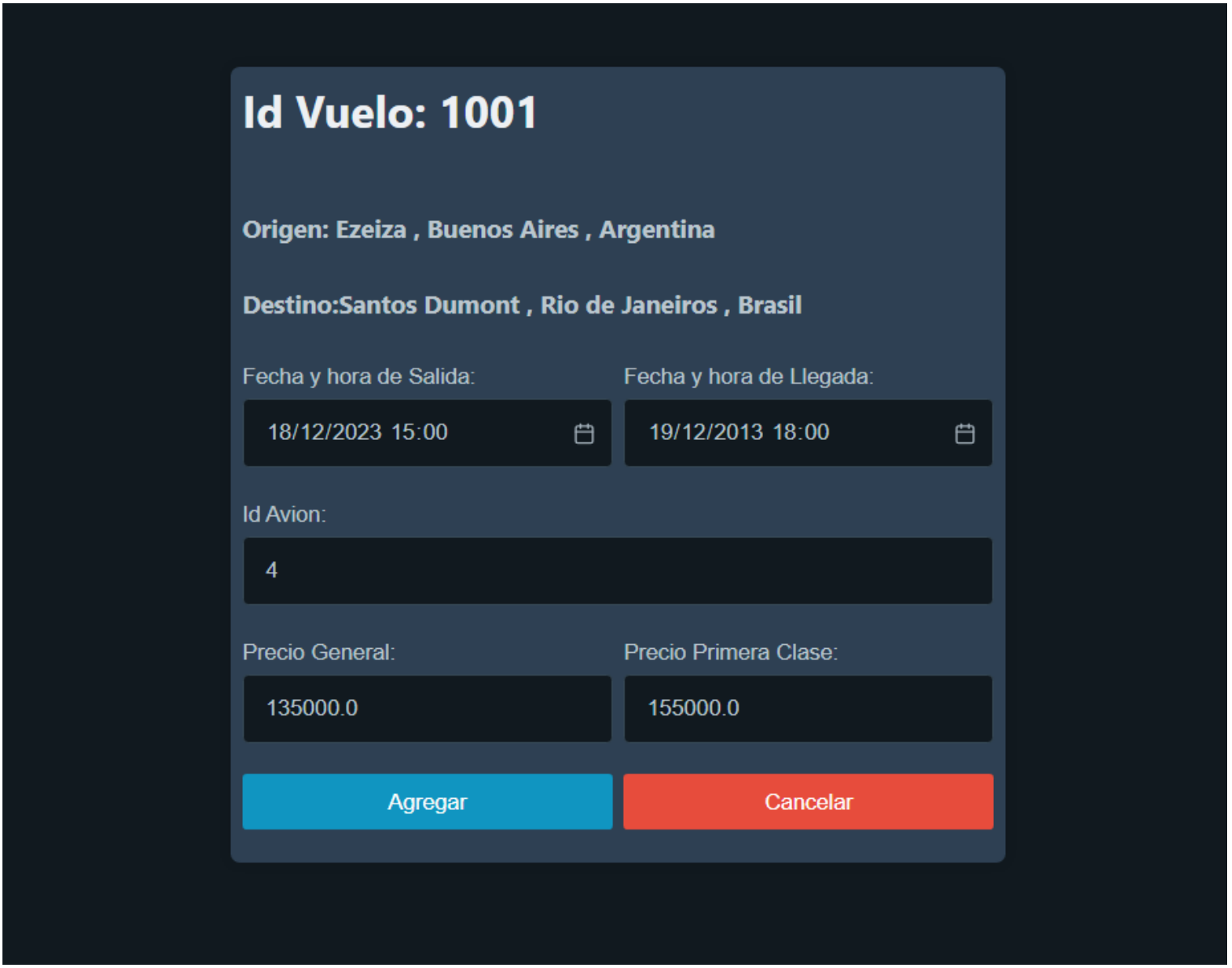
Lista de vuelos para que el Administrador pueda editar:



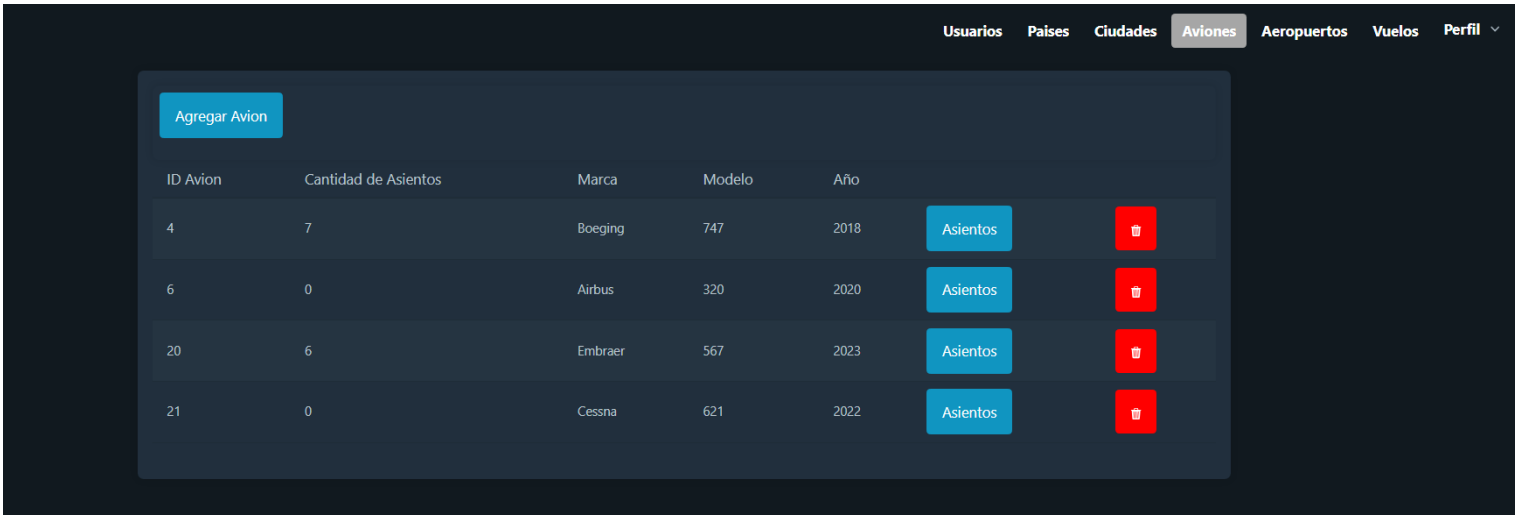
Agregar vuelo:



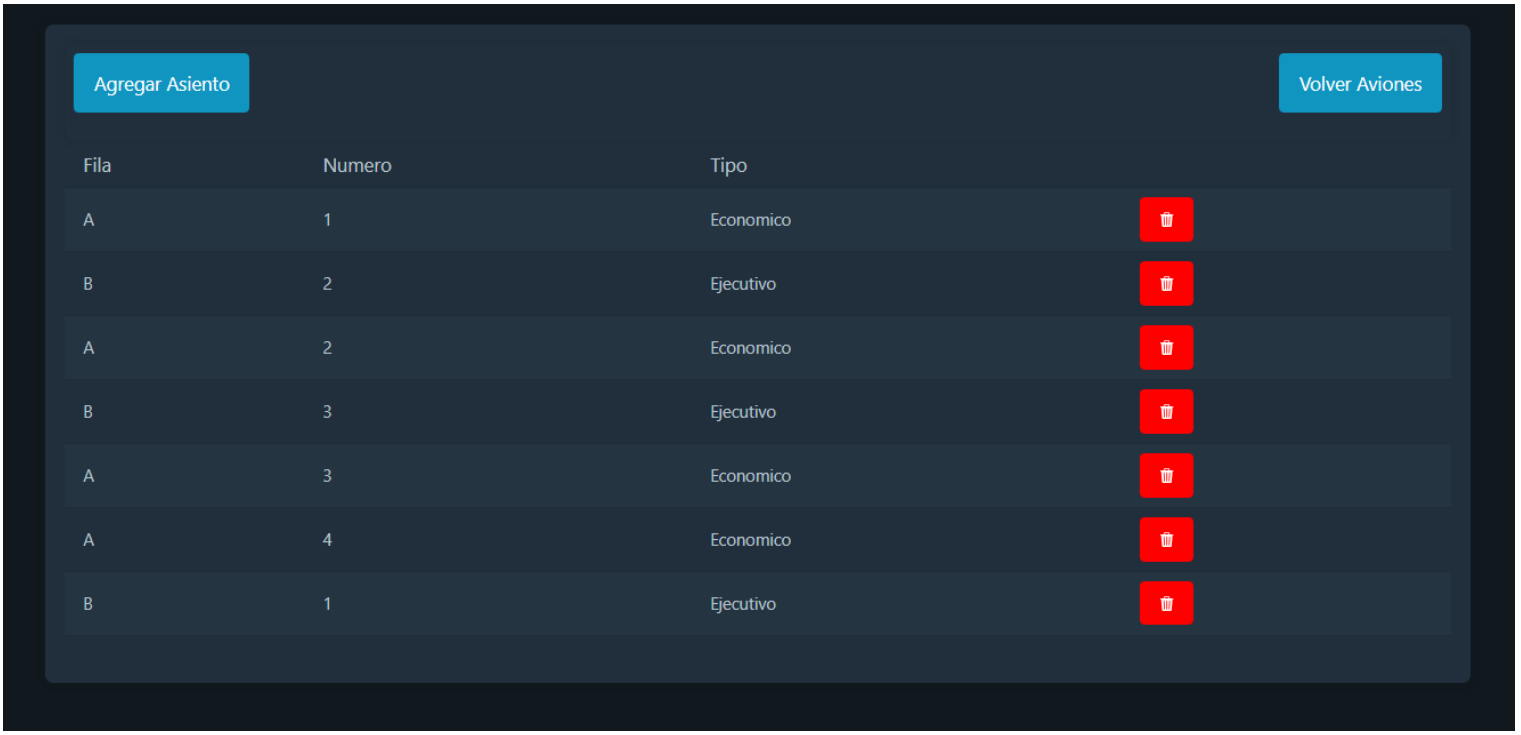
Editar vuelo:



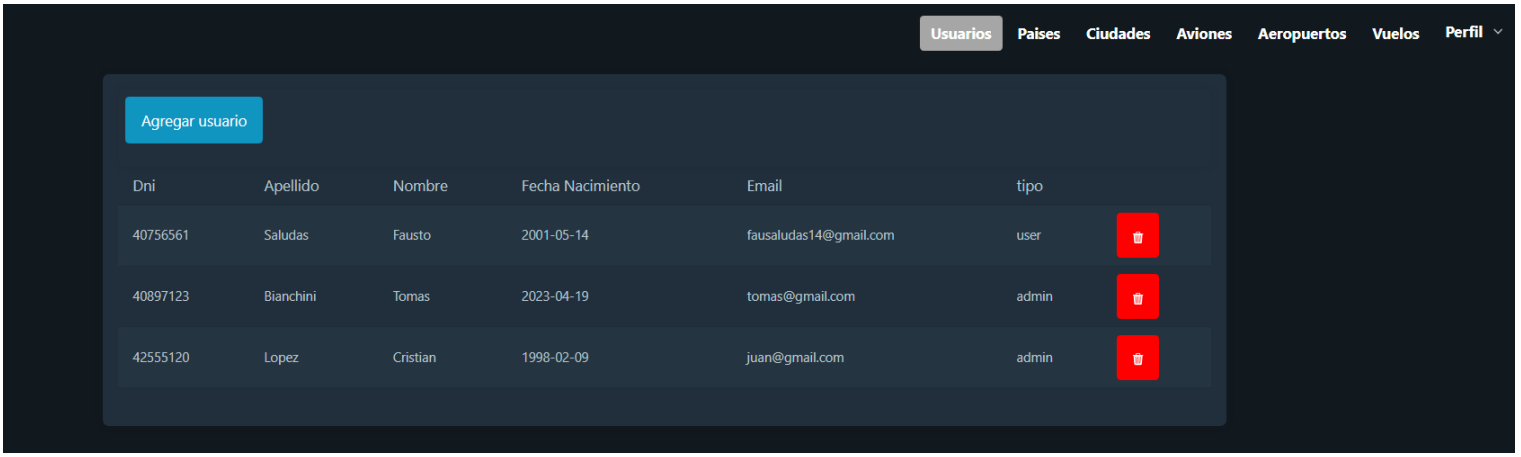
Lista de aviones:



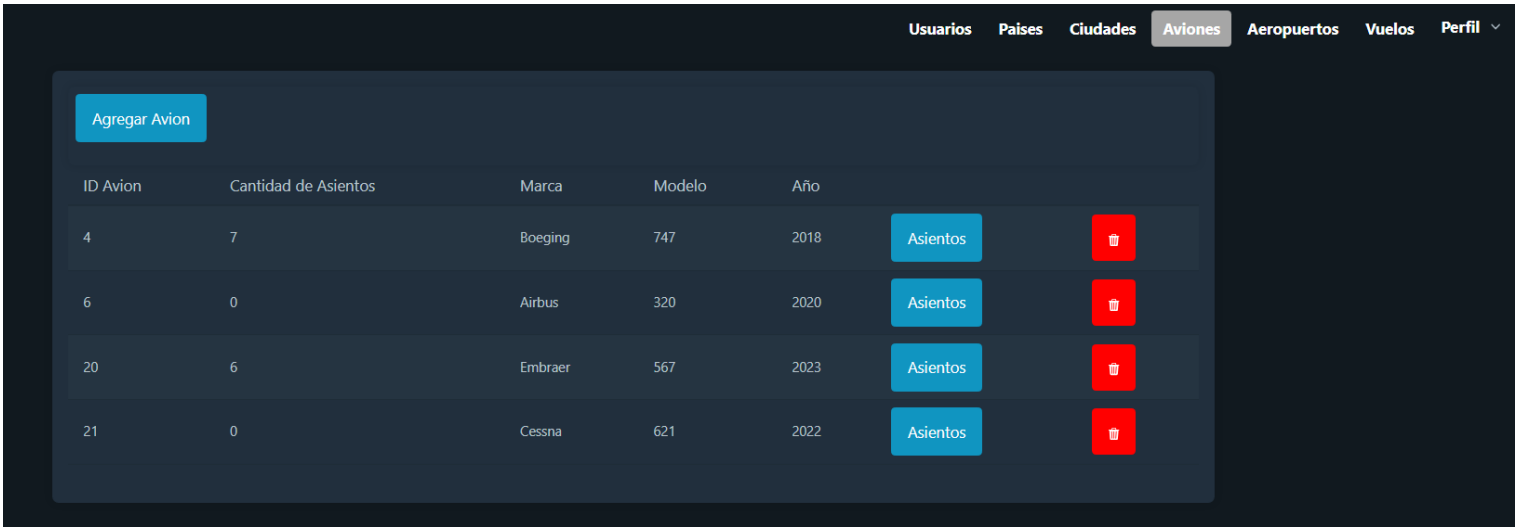
Lista de asientos por cada avión:



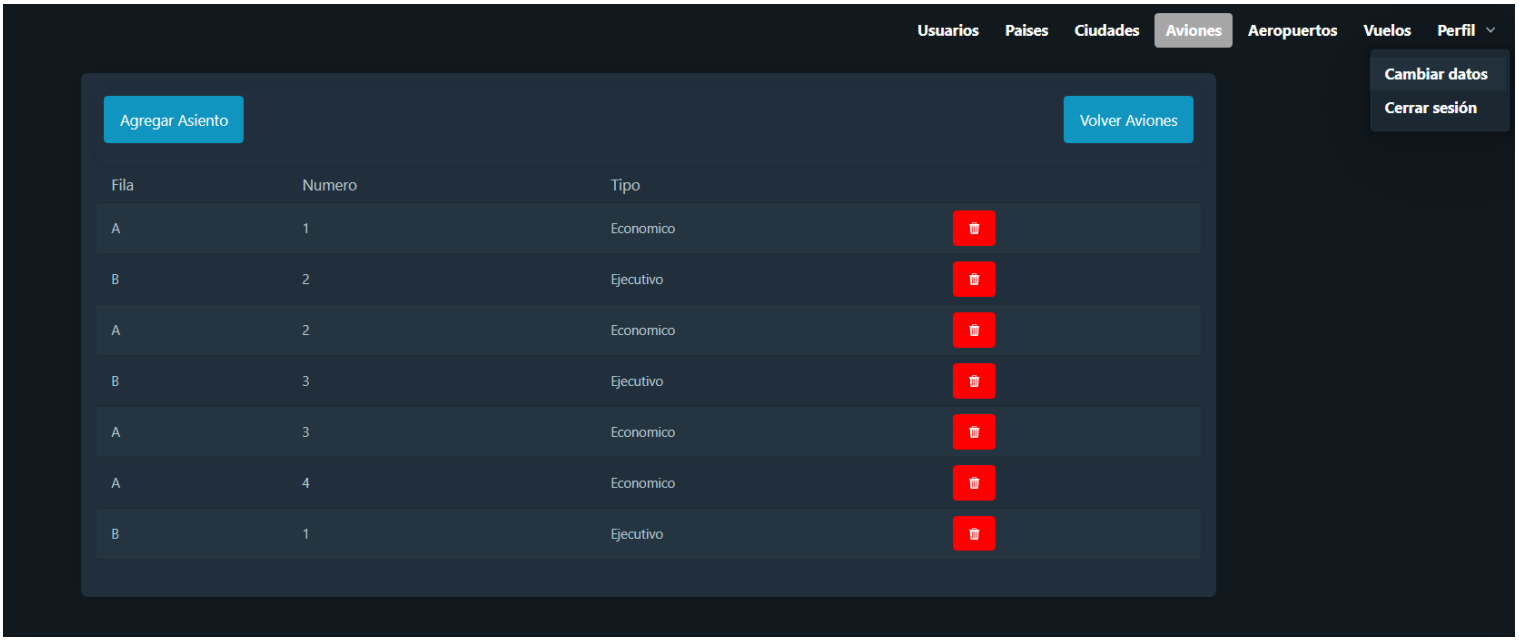
Lista de usuarios:



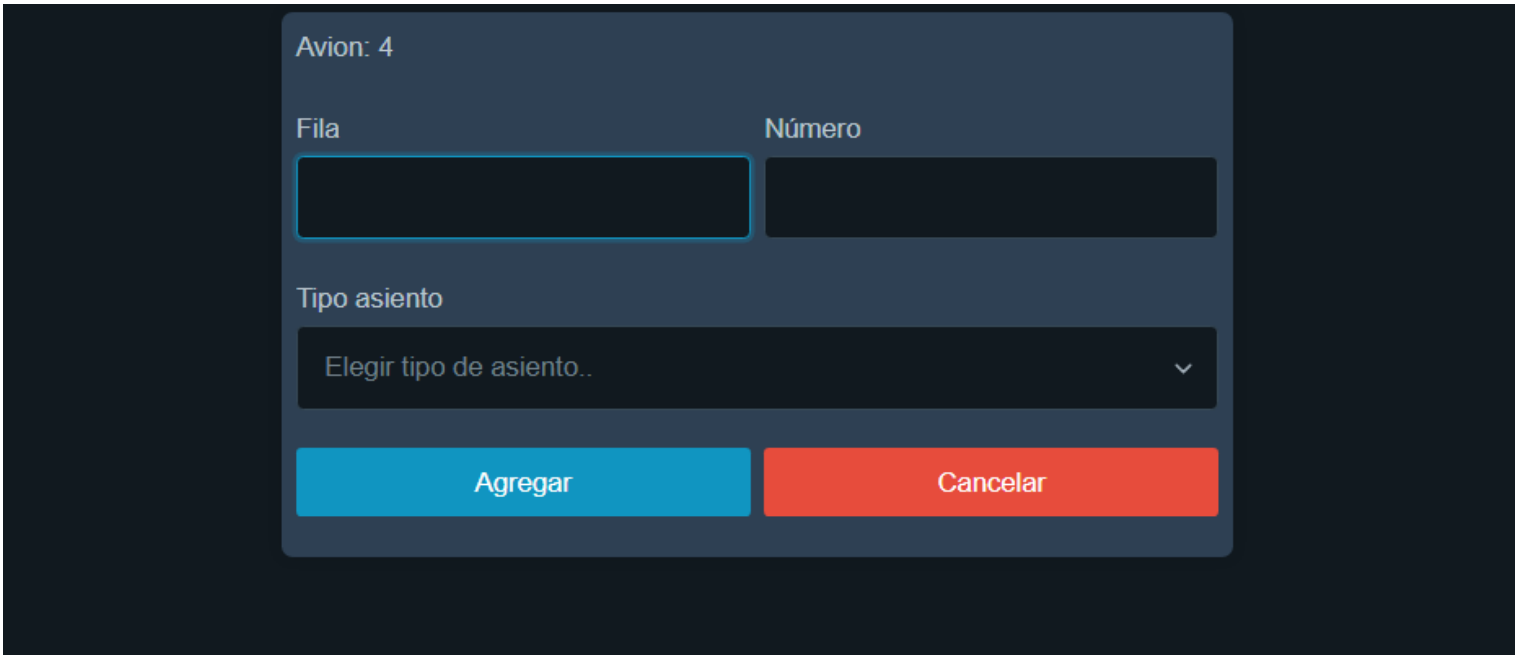
Lista de aviones:



Lista de asientos:



Agregar asiento:



Código fuente del proyecto

El código fuente completo del proyecto se puede encontrar subido a un repositorio público de la plataforma de control de versiones GitHub. A continuación, se mostrarán fragmentos de código utilizados para ejecutar el caso de uso nivel usuario.
https://github.com/TomasBianchini/Trabajo-Practico-Java

Pasaje Servlet:

```
public class PasajeServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public PasajeServlet() {
		super();
	}
	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		String accion = request.getParameter("accion");
		CtrlVuelo cv = new CtrlVuelo();
		Usuario usuario = (Usuario) request.getSession().getAttribute("usuario");
		boolean reenviar = true;
		if (usuario == null) {
			request.getRequestDispatcher("index.html").forward(request, response);
		} else {
			if (accion != null) {
				switch (accion) {
				case "redirecCompraPasaje": {
					try {
						int idvuelo = Integer.parseInt(request.getParameter("idvuelo"));
						Vuelo vue = new Vuelo();
						vue.setIdvuelo(idvuelo);
						Vuelo v = new Vuelo();
						v = cv.getById(vue);
						request.setAttribute("Vuelo", v);
						HashMap<String, Asiento> asientosDisponibles =
cv.getAsientosDisponibles(v);
						request.setAttribute("asientosDisponibles", asientosDisponibles);
```

```
request.getRequestDispatcher("WEB-INF/ui-pasaje/ComprarPasaje.jsp").forward(request, response);
                        reenviar = false;
                    } catch (Exception e) {
                        String message = e.getMessage();
                        request.setAttribute("message", message);
                    }
                    break;
                }
                case "compra": {
                    try {
                        int idvuelo = Integer.parseInt(request.getParameter("idvuelo"));
                        Vuelo vue = new Vuelo();
                        vue.setIdvuelo(idvuelo);
                        int idUsuario = Integer.parseInt(request.getParameter("idUsuario"));
                        Usuario usu = new Usuario();
                        usu.setIdUsuario(idUsuario);
                        int idAvion = Integer.parseInt(request.getParameter("idavion"));
                        String fila = request.getParameter("fila");
                        String numero = request.getParameter("numero");
                        String tipo = request.getParameter("tipo");
                        Asiento asiento = new Asiento();
                        asiento.setAvion(new Avion());
                        asiento.setFila(fila);
                        asiento.setNumero(numero);
                        asiento.setTipo(tipo);
                        asiento.getAvion().setIdAvion(idAvion);
                        CtrlPasaje cpas = new CtrlPasaje();
                        Pasaje pasaje = new Pasaje();
                        pasaje.setVuelo(vue);
                        pasaje.setUsuario(usu);
                        pasaje.setAsiento(asiento);
                        Pasaje p = cpas.add(pasaje);
                        if (p == null) {
                            String message = "error : No se pudo realizar la compra";
                            request.setAttribute("message", message);
                            reenviar = false;
                        } else {
                            String message = "Compra realizada con exito";
                            request.setAttribute("message", message);
                            reenviar = false;
                        }
                    } catch (Exception e) {
                        String message = e.getMessage();
                        request.setAttribute("message", message);
                        reenviar = false;
                    }
                    request.getRequestDispatcher("VueloServlet").forward(request, response);
                    break;
                }
                case "cancelarPasaje": {
```

```
                                try {
                                        Pasaje pas = new Pasaje();
                                        CtrlPasaje cpas = new CtrlPasaje();
                                        int id = Integer.parseInt(request.getParameter("idPasaje"));
                                        pas.setIdPasaje(id);
                                        pas = cpas.getById(pas);
                                        pas.setEstado("Cancelado");
                                        cpas.cambiarEstado(pas);
                                } catch (Exception e) {
                                        String message = e.getMessage();
                                        request.setAttribute("message", message);
                                }
                                request.getRequestDispatcher("VueloServlet").forward(request, response);
                                reenviar = false;
                                break;
                        }
                        default:
                                request.getRequestDispatcher("VueloServlet").forward(request, response);
                                reenviar = false;
                        }
                }
                if (reenviar)
                        request.getRequestDispatcher("VueloServlet").forward(request, response);
        }
}
```

Ctrl Pasaje:

```
public class CtrlPasaje {
        private DataPasaje dp;
        public CtrlPasaje() {
                dp = new DataPasaje();
        }
        public Pasaje add(Pasaje pasaje) throws Exception {
                CtrlUsuario cUsuario = new CtrlUsuario();
                CtrlVuelo cVuelo = new CtrlVuelo();
                CtrlAsiento cAsiento = new CtrlAsiento();
                Usuario usuario = cUsuario.getById(pasaje.getUsuario());
                Vuelo vuelo = cVuelo.getById(pasaje.getVuelo());
                Asiento asiento = cAsiento.getOne(pasaje.getAsiento());
                Pasaje p = new Pasaje();
                if (usuario != null && vuelo != null && asiento != null) {
                        p.setUsuario(usuario);
                        p.setVuelo(vuelo);
                        p.setAsiento(asiento);
                        p.setEstado("Confirmado");
                        if (this.isAsientoDisponible(vuelo, asiento)) {
                                try {
                                        dp.add(p);
                                        this.enviarMail(p);
```

```
                        } catch (Exception e) {
                                throw e;
                        }
                } else {
                        p = null;
                }
        }
        return p;
    }
    public boolean isAsientoDisponible(Vuelo v, Asiento asi) throws Exception {
        LinkedList<Pasaje> pasajes = this.getByVuelo(v);
        return pasajes.stream()
                .noneMatch(pas -> pas.getAsiento().getNumero().equalsIgnoreCase(asi.getNumero())
                        && pas.getAsiento().getFila().equalsIgnoreCase(asi.getFila())
                        && pas.getAsiento().getAvion().getIdAvion() == asi.getAvion().getIdAvion());
    }
    public LinkedList<Pasaje> getByVuelo(Vuelo vue) throws SQLException {
        return dp.getByVuelo(vue);
    }
    public LinkedList<Pasaje> getByIdUsuario(Usuario usu) throws SQLException {
        LinkedList<Pasaje> pasajes = dp.getByIdUsuario(usu);
        Collections.sort(pasajes, (p1, p2) -> Integer.compare(p2.getIdPasaje(), p1.getIdPasaje()));
        return pasajes;
    }
    public void cambiarEstado(Pasaje pas) throws SQLException {
        String[] estados = { "Confirmado", "Finalizado", "Cancelado" };
        LocalDateTime currentDate = LocalDateTime.now();
        boolean estadoValido = Arrays.asList(estados).contains(pas.getEstado());
        if (estadoValido) {
            long diferenciaEnMinutos = ChronoUnit.MINUTES.between(pas.getVuelo().getFechaHoraSalida(),
currentDate);
            if (diferenciaEnMinutos <= 360)
                dp.cambiarEstado(pas);
        } else {
            throw new Error("no se puede cancelar");
        }
    }
    public Pasaje getById(Pasaje pas) throws SQLException {
        return dp.getById(pas);
    }
    public void enviarMail(Pasaje p) throws Exception {
        try {
            Pasaje pas = this.getById(p);
            PdfService ps = new PdfService();
            ps.crearPdf(pas);
            EmailService em = new EmailService();
            em.sendEmail("Gracias por su compra!", pas, pas.getUsuario().getEmail());
        } catch (Exception e) {
            throw e;
        }
    }
```

```
}
```

Email Service:

```
public class EmailService {
	private String password = "igvntsmwdfutypsh";
	private String mail_from = "vuelos316@gmail.com";
	private Session session;
	public EmailService() {
		Properties properties = new Properties();
		properties.put("mail.smtp.host", "smtp.gmail.com.");
		properties.put("mail.smtp.socketFactory.port", "587");
		properties.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
		properties.put("mail.smtp.starttls.enable", "true");
		properties.put("mail.smtp.user", "usuario");
		properties.put("mail.smtp.auth", "true");
		properties.put("mail.smtp.port", "587");
		session = Session.getDefaultInstance(properties);
	}
	public void sendEmail(String subject, Pasaje p, String to) throws Exception {
		try {
			String texto = "La compra se realizo exitosamente! Puede cancelar la compra hasta 6 horas antes del
vuelo. ";
			MimeMessage message = new MimeMessage(session);
			message.setFrom(new InternetAddress(mail_from));
			message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
			message.setSubject(subject);
			MimeMultipart multipart = new MimeMultipart("related");
			MimeBodyPart adjuntoParte = new MimeBodyPart();
			adjuntoParte.attachFile("C:\\Users\\Usuario\\Downloads\\Pasajes\\comprobante_" + p.getIdPasaje() +
".pdf");
			multipart.addBodyPart(adjuntoParte);
			MimeBodyPart messageBodyPart = new MimeBodyPart();
			messageBodyPart.setText(texto);
			multipart.addBodyPart(messageBodyPart);
			message.setContent(multipart);
			Transport t = session.getTransport("smtp");
			t.connect(mail_from, this.password);
			t.sendMessage(message, message.getAllRecipients());
			t.close();
		} catch (Exception me) {
			throw me;
		}
	}
}
```

Pdf Service:

```
public class PdfService {
	public void crearPdf(Pasaje pas) throws Exception {
		try {
			Document document = new Document();
			String destino = "C:\\Users\\Usuario\\Downloads\\Pasajes\\comprobante_" + pas.getIdPasaje() + ".pdf";
```

```
                PdfWriter.getInstance(document, new FileOutputStream(destino));
                document.open();
                agregarContenido(document, pas);
                document.close();
            } catch (FileNotFoundException ex) {
                System.err.println("Error: Archivo no encontrado - " + ex.getMessage());
                ex.printStackTrace();
                throw ex;
            } catch (DocumentException ex) {
                System.err.println("Error en el documento PDF - " + ex.getMessage());
                ex.printStackTrace();
                throw ex;
            }
        }
        private static void agregarContenido(Document document, Pasaje pas) throws DocumentException {
            Font fontTitulo = new Font(Font.FontFamily.HELVETICA, 18, Font.ITALIC, BaseColor.BLACK);
            Paragraph titulo = new Paragraph("Pasaje Aéreo", fontTitulo);
            titulo.setAlignment(Paragraph.ALIGN_CENTER);
            document.add(titulo);
            document.add(new Paragraph("\n"));
            PdfPTable table = new PdfPTable(2);
            table.setWidthPercentage(70);
            table.setHorizontalAlignment(Element.ALIGN_CENTER);
            agregarCelda(table, "Nombre del Pasajero:", pas.getUsuario().getNombre() + " " +
pas.getUsuario().getApellido(),
                    BaseColor.DARK_GRAY);
            agregarCelda(table, "Origen: ",
                    pas.getVuelo().getAeropuertoOrigen().getNombre() + ", "
                            + pas.getVuelo().getAeropuertoOrigen().getCiudad().getNombre() + ", "
                            + pas.getVuelo().getAeropuertoOrigen().getCiudad().getPais().getNombre(),
                    BaseColor.DARK_GRAY);
            agregarCelda(table, "Fecha de Salida:", pas.getVuelo().getFechaHoraSalida()
                    .format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss")).toString(),
BaseColor.DARK_GRAY);
            agregarCelda(table, "Destino: ",
                    pas.getVuelo().getAeropuertoDestino().getNombre() + ", "
                            + pas.getVuelo().getAeropuertoDestino().getCiudad().getNombre() + ", "
                            + pas.getVuelo().getAeropuertoDestino().getCiudad().getPais().getNombre(),
                    BaseColor.DARK_GRAY);
            agregarCelda(table, "Fecha de Llegada:", pas.getVuelo().getFechaHoraLlegada()
                    .format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss")).toString(),
BaseColor.DARK_GRAY);
            agregarCelda(table, "Asiento:", pas.getAsiento().getFila() + " " + pas.getAsiento().getNumero(),
                    BaseColor.DARK_GRAY);
            document.add(table);
            document.add(new Paragraph("\n"));
            Font fontAgradecimiento = new Font(Font.FontFamily.HELVETICA, 14, Font.ITALIC, BaseColor.DARK_GRAY);
            Paragraph agradecimiento = new Paragraph("¡Gracias por elegir nuestra aerolínea!", fontAgradecimiento);
            agradecimiento.setAlignment(Paragraph.ALIGN_CENTER);
            document.add(agradecimiento);
        }
```

```
        private static void agregarCelda(PdfPTable table, String etiqueta, String valor, BaseColor colorFondo) {
                Font fontEtiqueta = new Font(Font.FontFamily.HELVETICA, 12, Font.BOLD, BaseColor.WHITE);
                Font fontValor = new Font(Font.FontFamily.HELVETICA, 12, Font.NORMAL, BaseColor.BLACK);
                PdfPCell etiquetaCell = new PdfPCell(new Phrase(etiqueta, fontEtiqueta));
                PdfPCell valorCell = new PdfPCell(new Phrase(valor, fontValor));
                etiquetaCell.setHorizontalAlignment(Element.ALIGN_RIGHT);
                valorCell.setHorizontalAlignment(Element.ALIGN_LEFT);
                etiquetaCell.setBackgroundColor(colorFondo);
                etiquetaCell.setBorderColor(BaseColor.GRAY);
                etiquetaCell.setBorderWidth(1f);
                valorCell.setBorderColor(BaseColor.GRAY);
                valorCell.setBorderWidth(1f);
                table.addCell(etiquetaCell);
                table.addCell(valorCell);
        }
}
```

Data Pasaje:

```
public class DataPasaje {
        public LinkedList<Pasaje> getAll() throws SQLException {
                Statement stmt = null;
                ResultSet rs = null;
                LinkedList<Pasaje> pasajes = new LinkedList<>();
                try {
                        stmt = DbConnector.getInstancia().getConn().createStatement();
                        rs = stmt.executeQuery(
                                        "select pas.idPasaje, pas.estado, usu.tipoDocumento, usu.nroDocumento, usu.nombre,
usu.apellido,"
                                                        + " asi.*, "
                                                        + " pO.nombre as nPaisO, pD.nombre as nPaisD, ciuD.nombre as
nCiudadD, vue.*, ciuO.nombre as nCiudadO, "
                                                        + " aeroO.nombre as nAeroO, aeroD.nombre as nAeroD " + " from
pasaje pas "
                                                        + " inner join vuelo vue on vue.idVuelo = pas.idVuelo"
                                                        + " inner join usuario usu on usu.idUsuario = pas.idUsuario"
                                                        + "         inner join asiento asi on asi.fila = pas.fila and asi.numero =
pas.numero and asi.idAvion = pas.idAvion "
                                                        + " inner join aeropuerto aeroO on aeroO.idaeropuerto =
vue.idAeropuertoOrigen"
                                                        + " inner join aeropuerto aeroD on aeroD.idaeropuerto =
vue.idAeropuertoDestino"
                                                        + " inner join ciudad ciuO on ciuO.codPostal = aeroO.codPostal "
                                                        + " inner join ciudad ciuD on ciuD.codPostal = aeroD.codPostal "
                                                        + " inner join pais pO on pO.idpais = ciuO.idPais"
                                                        + " inner join pais pD on pD.idpais = ciuD.idPais");
                        if (rs != null) {
                                while (rs.next()) {
                                        Pasaje p = new Pasaje();
                                        p.setIdPasaje(rs.getInt("idPasaje"));
                                        p.setEstado(rs.getString("estado"));
```

```
                                        p.setAsiento(new Asiento());
                                        p.getAsiento().setAvion(new Avion());
                                        p.getAsiento().getAvion().setIdAvion(rs.getInt("asi.idAvion"));
                                        p.getAsiento().setFila(rs.getString("asi.fila"));
                                        p.getAsiento().setNumero(rs.getString("asi.numero"));
                                        p.getAsiento().setTipo(rs.getString("asi.tipo"));
                                        p.setUsuario(new Usuario());
                                        p.getUsuario().setNroDocumento(rs.getString("usu.nroDocumento"));
                                        p.getUsuario().setTipoDocumento(rs.getString("usu.tipoDocumento"));
                                        p.getUsuario().setNombre(rs.getString("usu.nombre"));
                                        p.getUsuario().setApellido(rs.getString("usu.apellido"));
                                        p.setVuelo(new Vuelo());
                                        p.getVuelo().setAeropuertoDestino(new Aeropuerto());
                                        p.getVuelo().setAeropuertoOrigen(new Aeropuerto());
                                        p.getVuelo().getAeropuertoDestino().setCiudad(new Ciudad());
                                        p.getVuelo().getAeropuertoOrigen().setCiudad(new Ciudad());
                                        p.getVuelo().getAeropuertoDestino().getCiudad().setPais(new Pais());
                                        p.getVuelo().getAeropuertoOrigen().getCiudad().setPais(new Pais());
                                        p.getVuelo().setIdvuelo(rs.getInt("idVuelo"));
                                        p.getVuelo().setFechaHoraSalida(rs.getObject("fechaHoraSalida",
LocalDateTime.class));
                                        p.getVuelo().setFechaHoraLlegada(rs.getObject("fechaHoraLlegada",
LocalDateTime.class));
                                        p.getVuelo().getAeropuertoOrigen().setIdAeropuerto(rs.getInt("idAeropuertoOrigen"));
                                        p.getVuelo().getAeropuertoOrigen().setNombre(rs.getString("nAeroO"));

p.getVuelo().getAeropuertoDestino().setIdAeropuerto(rs.getInt("idAeropuertoDestino"));
                                        p.getVuelo().getAeropuertoDestino().setNombre(rs.getString("nAeroD"));

p.getVuelo().getAeropuertoOrigen().getCiudad().setNombre(rs.getString("nCiudadO"));

p.getVuelo().getAeropuertoDestino().getCiudad().setNombre(rs.getString("nCiudadD"));

p.getVuelo().getAeropuertoOrigen().getCiudad().getPais().setNombre(rs.getString("nPaisO"));

p.getVuelo().getAeropuertoDestino().getCiudad().getPais().setNombre(rs.getString("nPaisD"));
                                        pasajes.add(p);
                                }
                        }
                } catch (SQLException e) {
                        throw e;
                } finally {
                        try {
                                if (rs != null) {
                                        rs.close();
                                }
                                if (stmt != null) {
                                        stmt.close();
                                }
                                DbConnector.getInstancia().releaseConn();
                        } catch (SQLException e) {
```

```
                    throw e;
                }
            }
            return pasajes;
        }
        public LinkedList<Pasaje> getByIdUsuario(Usuario usu) throws SQLException {
            PreparedStatement stmt = null;
            ResultSet rs = null;
            LinkedList<Pasaje> pasajes = new LinkedList<>();
            try {
                stmt = DbConnector.getInstancia().getConn().prepareStatement(
                        "select pas.idPasaje, pas.estado, usu.tipoDocumento, usu.idusuario,
usu.nroDocumento, usu.nombre, usu.apellido, \r\n"
                                + "                                                asi.* ,\r\n"
                                + "                                                pO.nombre
as nPaisO, pD.nombre as nPaisD, ciuD.nombre as nCiudadD, vue.*, ciuO.nombre as nCiudadO, \r\n"
                                + "
aeroO.nombre as nAeroO, aeroD.nombre as nAeroD from pasaje pas \r\n"
                                + "                                                inner join
vuelo vue on vue.idVuelo = pas.idVuelo\r\n"
                                + "                                            inner join usuario usu
on usu.idUsuario = pas.idUsuario\r\n"
                                + "                                                    inner
join asiento asi on asi.fila = pas.fila and asi.numero = pas.numero and asi.idAvion = pas.idAvion \r\n"
                                + "                                                inner join
aeropuerto aeroO on aeroO.idaeropuerto = vue.idAeropuertoOrigen\r\n"
                                + "                                                inner join
aeropuerto aeroD on aeroD.idaeropuerto = vue.idAeropuertoDestino\r\n"
                                + "                                                inner join
ciudad ciuO on ciuO.codPostal = aeroO.codPostal \r\n"
                                + "                                                inner join
ciudad ciuD on ciuD.codPostal = aeroD.codPostal \r\n"
                                + "                                                inner join
pais pO on pO.idpais = ciuO.idPais \r\n"
                                + "                                                inner join
pais pD on pD.idpais = ciuD.idPais  where pas.idUsuario = ?");
                stmt.setInt(1, usu.getIdUsuario());
                rs = stmt.executeQuery();
                if (rs != null) {
                    while (rs.next()) {
                        Pasaje p = new Pasaje();
                        p.setIdPasaje(rs.getInt("idPasaje"));
                        p.setEstado(rs.getString("estado"));
                        p.setAsiento(new Asiento());
                        p.getAsiento().setAvion(new Avion());
                        p.getAsiento().getAvion().setIdAvion(rs.getInt("asi.idAvion"));
                        p.getAsiento().setFila(rs.getString("asi.fila"));
                        p.getAsiento().setNumero(rs.getString("asi.numero"));
                        p.getAsiento().setTipo(rs.getString("asi.tipo"));
                        p.setUsuario(new Usuario());
                        p.getUsuario().setIdUsuario(rs.getInt("usu.idUsuario"));
```

```
                                        p.getUsuario().setNroDocumento(rs.getString("usu.nroDocumento"));
                                        p.getUsuario().setTipoDocumento(rs.getString("usu.tipoDocumento"));
                                        p.getUsuario().setNombre(rs.getString("usu.nombre"));
                                        p.getUsuario().setApellido(rs.getString("usu.apellido"));
                                        p.setVuelo(new Vuelo());
                                        p.getVuelo().setAeropuertoDestino(new Aeropuerto());
                                        p.getVuelo().setAeropuertoOrigen(new Aeropuerto());
                                        p.getVuelo().getAeropuertoDestino().setCiudad(new Ciudad());
                                        p.getVuelo().getAeropuertoOrigen().setCiudad(new Ciudad());
                                        p.getVuelo().getAeropuertoDestino().getCiudad().setPais(new Pais());
                                        p.getVuelo().getAeropuertoOrigen().getCiudad().setPais(new Pais());
                                        p.getVuelo().setIdvuelo(rs.getInt("idVuelo"));
                                        p.getVuelo().setFechaHoraSalida(rs.getObject("fechaHoraSalida",
LocalDateTime.class));
                                        p.getVuelo().setFechaHoraLlegada(rs.getObject("fechaHoraLlegada",
LocalDateTime.class));
                                        p.getVuelo().getAeropuertoOrigen().setIdAeropuerto(rs.getInt("idAeropuertoOrigen"));
                                        p.getVuelo().getAeropuertoOrigen().setNombre(rs.getString("nAeroO"));

p.getVuelo().getAeropuertoDestino().setIdAeropuerto(rs.getInt("idAeropuertoDestino"));
                                        p.getVuelo().getAeropuertoDestino().setNombre(rs.getString("nAeroD"));

p.getVuelo().getAeropuertoOrigen().getCiudad().setNombre(rs.getString("nCiudadO"));

p.getVuelo().getAeropuertoDestino().getCiudad().setNombre(rs.getString("nCiudadD"));

p.getVuelo().getAeropuertoOrigen().getCiudad().getPais().setNombre(rs.getString("nPaisO"));

p.getVuelo().getAeropuertoDestino().getCiudad().getPais().setNombre(rs.getString("nPaisD"));
                                        pasajes.add(p);
                                }
                        }
                } catch (SQLException e) {
                        throw e;
                } finally {
                        try {
                                if (rs != null) {
                                        rs.close();
                                }
                                if (stmt != null) {
                                        stmt.close();
                                }
                                DbConnector.getInstancia().releaseConn();
                        } catch (SQLException e) {
                                throw e;
                        }
                }
                return pasajes;
        }
        public LinkedList<Pasaje> getByVuelo(Vuelo vue) throws SQLException {
                PreparedStatement stmt = null;
```

```
		ResultSet rs = null;
		LinkedList<Pasaje> pasajes = new LinkedList<>();
		try {
			stmt = DbConnector.getInstancia().getConn().prepareStatement(
					"select pas.idPasaje, pas.estado, usu.tipoDocumento, usu.nroDocumento, usu.nombre,
usu.apellido,"+ " asi.*, "
+ " pO.nombre as nPaisO, pD.nombre as nPaisD, ciuD.nombre as nCiudadD, vue.*, ciuO.nombre as nCiudadO, "
+ " aeroO.nombre as nAeroO, aeroD.nombre as nAeroD " + " from pasaje pas "
+ " inner join vuelo vue on vue.idVuelo = pas.idVuelo"+ " inner join usuario usu on usu.idUsuario = pas.idUsuario"
+ "inner join asiento asi on asi.fila = pas.fila and asi.numero = pas.numero and asi.idAvion = pas.idAvion "
+ " inner join aeropuerto aeroO on aeroO.idaeropuerto = vue.idAeropuertoOrigen"
+ " inner join aeropuerto aeroD on aeroD.idaeropuerto = vue.idAeropuertoDestino"+ " inner join ciudad ciuO on ciuO.codPostal
= aeroO.codPostal "+ " inner join ciudad ciuD on ciuD.codPostal = aeroD.codPostal "
+ " inner join pais pO on pO.idpais = ciuO.idPais"+ " inner join pais pD on pD.idpais = ciuD.idPais"+ " where vue.idVuelo = ? and
pas.estado= 'Confirmado' ");

stmt.setInt(1, vue.getIdvuelo());
			rs = stmt.executeQuery();
			if (rs != null) {
				while (rs.next()) {
					Pasaje p = new Pasaje();
					p.setIdPasaje(rs.getInt("idPasaje"));
					p.setEstado(rs.getString("estado"));
					p.setAsiento(new Asiento());
					p.getAsiento().setAvion(new Avion());
					p.getAsiento().getAvion().setIdAvion(rs.getInt("asi.idAvion"));
					p.getAsiento().setFila(rs.getString("asi.fila"));
					p.getAsiento().setNumero(rs.getString("asi.numero"));
					p.getAsiento().setTipo(rs.getString("asi.tipo"));
					p.setUsuario(new Usuario());
					p.getUsuario().setNroDocumento(rs.getString("usu.nroDocumento"));
					p.getUsuario().setTipoDocumento(rs.getString("usu.tipoDocumento"));
					p.getUsuario().setNombre(rs.getString("usu.nombre"));
					p.getUsuario().setApellido(rs.getString("usu.apellido"));
					p.setVuelo(new Vuelo());
					p.getVuelo().setAeropuertoDestino(new Aeropuerto());
					p.getVuelo().setAeropuertoOrigen(new Aeropuerto());
					p.getVuelo().getAeropuertoDestino().setCiudad(new Ciudad());
					p.getVuelo().getAeropuertoOrigen().setCiudad(new Ciudad());
					p.getVuelo().getAeropuertoDestino().getCiudad().setPais(new Pais());
					p.getVuelo().getAeropuertoOrigen().getCiudad().setPais(new Pais());
					p.getVuelo().setIdvuelo(rs.getInt("idVuelo"));
					p.getVuelo().setFechaHoraSalida(rs.getObject("fechaHoraSalida",
LocalDateTime.class));
					p.getVuelo().setFechaHoraLlegada(rs.getObject("fechaHoraLlegada",
LocalDateTime.class));
					p.getVuelo().getAeropuertoOrigen().setIdAeropuerto(rs.getInt("idAeropuertoOrigen"));
					p.getVuelo().getAeropuertoOrigen().setNombre(rs.getString("nAeroO"));

p.getVuelo().getAeropuertoDestino().setIdAeropuerto(rs.getInt("idAeropuertoDestino"));
					p.getVuelo().getAeropuertoDestino().setNombre(rs.getString("nAeroD"));
```

```
p.getVuelo().getAeropuertoOrigen().getCiudad().setNombre(rs.getString("nCiudadO"));

p.getVuelo().getAeropuertoDestino().getCiudad().setNombre(rs.getString("nCiudadD"));

p.getVuelo().getAeropuertoOrigen().getCiudad().getPais().setNombre(rs.getString("nPaisO"));

p.getVuelo().getAeropuertoDestino().getCiudad().getPais().setNombre(rs.getString("nPaisD"));
                                        pasajes.add(p);
                                }
                        }
                } catch (SQLException e) {
                        throw e;
                } finally {
                        try {
                                if (rs != null) {
                                        rs.close();
                                }
                                if (stmt != null) {
                                        stmt.close();
                                }
                                DbConnector.getInstancia().releaseConn();
                        } catch (SQLException e) {
                                throw e;
                        }
                }
                return pasajes;
        }
        public void cambiarEstado(Pasaje p) throws SQLException {
                PreparedStatement pstmt = null;
                try {
                        pstmt = DbConnector.getInstancia().getConn()
                                        .prepareStatement("UPDATE pasaje SET estado= ? WHERE IdPasaje=?");
                        pstmt.setString(1, p.getEstado());
                        pstmt.setInt(2, p.getIdPasaje());
                        pstmt.executeUpdate();
                } catch (SQLException e) {
                        throw e;
                } finally {
                        try {
                                if (pstmt != null)
                                        pstmt.close();
                                DbConnector.getInstancia().releaseConn();
                        } catch (SQLException e) {
                                throw e;
                        }
                }
        }
        public Pasaje add(Pasaje p) throws SQLException {
                PreparedStatement stmt = null;
                ResultSet keyResultSet = null;
```

```
		try {
			stmt = DbConnector.getInstancia().getConn().prepareStatement(
					"insert into pasaje(Estado, idVuelo, fila, numero, idAvion, idUsuario) values(?,?,?,?,?,?)",
					PreparedStatement.RETURN_GENERATED_KEYS);
			stmt.setString(1, p.getEstado());
			stmt.setInt(2, p.getVuelo().getIdvuelo());
			stmt.setString(3, p.getAsiento().getFila());
			stmt.setString(4, p.getAsiento().getNumero());
			stmt.setInt(5, p.getAsiento().getAvion().getIdAvion());
			stmt.setInt(6, p.getUsuario().getIdUsuario());
			stmt.executeUpdate();
			keyResultSet = stmt.getGeneratedKeys();
			if (keyResultSet != null && keyResultSet.next()) {
				p.setIdPasaje(keyResultSet.getInt(1));
			}
		} catch (SQLException e) {
			throw e;
		} finally {
			try {
				if (stmt != null)
					stmt.close();
				DbConnector.getInstancia().releaseConn();
			} catch (SQLException e) {
				throw e;
			}
		}
		return p;
	}
	public Pasaje getById(Pasaje pas) throws SQLException {
		PreparedStatement stmt = null;
		ResultSet rs = null;
		Pasaje p = null;
		try {
			stmt = DbConnector.getInstancia().getConn().prepareStatement(
					"select pas.idPasaje, pas.estado, usu.tipoDocumento, usu.nroDocumento, usu.nombre,
usu.apellido, usu.email, "+ " asi.*, "+ " pO.nombre as nPaisO, pD.nombre as nPaisD, ciuD.nombre as nCiudadD, vue.*,
ciuO.nombre as nCiudadO, "
+ " aeroO.nombre as nAeroO, aeroD.nombre as nAeroD " + " from pasaje pas "
+ " inner join vuelo vue on vue.idVuelo = pas.idVuelo"
+ " inner join usuario usu on usu.idUsuario = pas.idUsuario"
+ "	inner join asiento asi on asi.fila = pas.fila and asi.numero = pas.numero and asi.idAvion = pas.idAvion "
+ " inner join aeropuerto aeroO on aeroO.idaeropuerto = vue.idAeropuertoOrigen"
+ " inner join aeropuerto aeroD on aeroD.idaeropuerto = vue.idAeropuertoDestino"
+ " inner join ciudad ciuO on ciuO.codPostal = aeroO.codPostal "
+ " inner join ciudad ciuD on ciuD.codPostal = aeroD.codPostal "
+ " inner join pais pO on pO.idpais = ciuO.idPais"
+ " inner join pais pD on pD.idpais = ciuD.idPais" + " where pas.idpasaje = ?  ");
			stmt.setInt(1, pas.getIdPasaje());
			rs = stmt.executeQuery();
			if (rs != null && rs.next()) {
				p = new Pasaje();
```

```
                p.setIdPasaje(rs.getInt("idPasaje"));
                p.setEstado(rs.getString("estado"));
                p.setAsiento(new Asiento());
                p.getAsiento().setAvion(new Avion());
                p.getAsiento().getAvion().setIdAvion(rs.getInt("asi.idAvion"));
                p.getAsiento().setFila(rs.getString("asi.fila"));
                p.getAsiento().setNumero(rs.getString("asi.numero"));
                p.getAsiento().setTipo(rs.getString("asi.tipo"));
                p.setUsuario(new Usuario());
                p.getUsuario().setNroDocumento(rs.getString("usu.nroDocumento"));
                p.getUsuario().setTipoDocumento(rs.getString("usu.tipoDocumento"));
                p.getUsuario().setNombre(rs.getString("usu.nombre"));
                p.getUsuario().setApellido(rs.getString("usu.apellido"));
                p.getUsuario().setEmail(rs.getString("usu.email"));
                p.setVuelo(new Vuelo());
                p.getVuelo().setAeropuertoDestino(new Aeropuerto());
                p.getVuelo().setAeropuertoOrigen(new Aeropuerto());
                p.getVuelo().getAeropuertoDestino().setCiudad(new Ciudad());
                p.getVuelo().getAeropuertoOrigen().setCiudad(new Ciudad());
                p.getVuelo().getAeropuertoDestino().getCiudad().setPais(new Pais());
                p.getVuelo().getAeropuertoOrigen().getCiudad().setPais(new Pais());
                p.getVuelo().setIdvuelo(rs.getInt("idVuelo"));
                p.getVuelo().setFechaHoraSalida(rs.getObject("fechaHoraSalida", LocalDateTime.class));
                p.getVuelo().setFechaHoraLlegada(rs.getObject("fechaHoraLlegada", LocalDateTime.class));
                p.getVuelo().getAeropuertoOrigen().setIdAeropuerto(rs.getInt("idAeropuertoOrigen"));
                p.getVuelo().getAeropuertoOrigen().setNombre(rs.getString("nAeroO"));
                p.getVuelo().getAeropuertoDestino().setIdAeropuerto(rs.getInt("idAeropuertoDestino"));
                p.getVuelo().getAeropuertoDestino().setNombre(rs.getString("nAeroD"));
                p.getVuelo().getAeropuertoOrigen().getCiudad().setNombre(rs.getString("nCiudadO"));
                p.getVuelo().getAeropuertoDestino().getCiudad().setNombre(rs.getString("nCiudadD"));
                p.getVuelo().getAeropuertoOrigen().getCiudad().getPais().setNombre(rs.getString("nPaisO"));
                p.getVuelo().getAeropuertoDestino().getCiudad().getPais().setNombre(rs.getString("nPaisD"));
            }
        } catch (SQLException e) {
            throw e;
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (stmt != null) {
                    stmt.close();
                }
                DbConnector.getInstancia().releaseConn();
            } catch (SQLException e) {
                throw e;
            }
        }
        return p;       }
}
```

Pasaje:

```java
public class Pasaje {
	private int idPasaje;
	private String estado;
	private Vuelo vuelo;
	private Asiento asiento;
	private Usuario usuario;
	public int getIdPasaje() {
		return idPasaje;
	}
	public void setIdPasaje(int idPasaje) {
		this.idPasaje = idPasaje;
	}
	public String getEstado() {
		return estado;
	}
	public void setEstado(String estado) {
		this.estado = estado;
	}
	public Vuelo getVuelo() {
		return vuelo;
	}
	public void setVuelo(Vuelo vuelo) {
		this.vuelo = vuelo;
	}
	public Asiento getAsiento() {
		return asiento;
	}
	public void setAsiento(Asiento asiento) {
		this.asiento = asiento;
	}
	public Usuario getUsuario() {
		return this.usuario;
	}
	public void setUsuario(Usuario usuario) {
		this.usuario = usuario;
	}
}
```