

**Tema**

VIAJECITO SOAP DOTNET

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU. CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**INTEGRANTES**

Bazurto Christopher & Drouet Stephen

**Fecha**

02/06/2025

Tabla de contenido

[INTRODUCCIÓN 6](#_Toc199713876)

[OBJETIVOS 6](#_Toc199713877)

[Objetivo General 6](#_Toc199713878)

[Objetivos Específicos 6](#_Toc199713879)

[MARCO TEÓRICO 6](#_Toc199713880)

[Arquitectura de Software 6](#_Toc199713881)

[Arquitectura Orientada a Servicios 7](#_Toc199713882)

[Protocolo SOAP 7](#_Toc199713883)

[Protocolo REST 7](#_Toc199713884)

[HERRAMIENTAS 8](#_Toc199713885)

[Aplicación de Consola con Java 8](#_Toc199713886)

[Aplicación de Escritorio con Java 8](#_Toc199713887)

[Node.js con React para la Aplicación Web 8](#_Toc199713888)

[React Native para la Aplicación Móvil 8](#_Toc199713889)

[PARTE PRÁCTICA 8](#_Toc199713890)

[ESTRUCTURA DEL PROYECTO 8](#_Toc199713891)

[SERVIDOR 9](#_Toc199713892)

[CREACION DE LA APLICACIÓN 9](#_Toc199713893)

[MODELO MVC 10](#_Toc199713894)

[IMPLEMENTACIÓN 12](#_Toc199713895)

[HABILITAR CORS 19](#_Toc199713896)

[PRUEBAS UNITARIAS 20](#_Toc199713897)

[EJECUCIÓN DE LA APLICACIÓN 24](#_Toc199713898)

[CLIENTE DE CONSOLA 24](#_Toc199713899)

[CREACION DE LA APLICACIÓN 24](#_Toc199713900)

[MODELO MVC 25](#_Toc199713901)

[IMPLEMENTACIÓN 25](#_Toc199713902)

[EJECUCIÓN DE LA APLICACIÓN 34](#_Toc199713903)

[CLIENTE DE ESCRITORIO 35](#_Toc199713904)

[CREACION DE LA APLICACIÓN 35](#_Toc199713905)

[MODELO MVC 35](#_Toc199713906)

[IMPLEMENTACIÓN 36](#_Toc199713907)

[EJECUCIÓN DE LA APLICACIÓN 39](#_Toc199713908)

[CLIENTE WEB 39](#_Toc199713909)

[CREACION DE LA APLICACIÓN 40](#_Toc199713910)

[MODELO MVC 40](#_Toc199713911)

[IMPLEMENTACIÓN 41](#_Toc199713912)

[EJECUCIÓN DE LA APLICACIÓN 62](#_Toc199713913)

[CLIENTE MÓVIL 64](#_Toc199713914)

[CREACION DE LA APLICACIÓN 64](#_Toc199713915)

[MODELO MVC 65](#_Toc199713916)

[IMPLEMENTACIÓN 65](#_Toc199713917)

[EJECUCIÓN DE LA APLICACIÓN 86](#_Toc199713918)

[CONCLUSIONES 89](#_Toc199713919)

[RECOMENDACIONES 89](#_Toc199713920)

[BIBLIOGRAFÍA 89](#_Toc199713921)

[ANEXOS 90](#_Toc199713922)

**ÍNDICE DE IMÁGENES**

[Figura 1. Arquitectura de Software 7](#_Toc199714614)

[Figura 2. Arquitectura de Orientada a Servicios 7](#_Toc199714615)

[Figura 3. SOAP 7](#_Toc199714616)

[Figura 5. Estructura del proyecto carpeta raíz 9](#_Toc199714617)

[Figura 6. Estructura del proyecto carpetas internas 9](#_Toc199714618)

[Figura 7. Crear Proyecto en NetBeans 10](#_Toc199714619)

[Figura 8. Configuración del Proyecto en NetBeans 10](#_Toc199714620)

[Figura 9. Creación de un nuevo paquete 11](#_Toc199714621)

[Figura 10. Estructura MVC del proyecto 11](#_Toc199714622)

[Figura 11. Creación de pruebas unitarias 21](#_Toc199714623)

[Figura 12. Ejecución de pruebas unitarias 23](#_Toc199714624)

[Figura 13. Ejecución del servidor payara 24](#_Toc199714625)

[Figura 14. Configuración de la aplicación de consola 24](#_Toc199714626)

[Figura 15. Creación de un nuevo paquete en cliente de consola 25](#_Toc199714627)

[Figura 16. Estructura MVC del proyecto de consola 25](#_Toc199714628)

[Figura 17. Creación del cliente del Web Service 26](#_Toc199714629)

[Figura 18. Creación del cliente del Web Service RESTFUL archivos necesarios 26](#_Toc199714630)

[Figura 19. Creación de funciones del cliente RESTFUL en el lado del cliente 26](#_Toc199714631)

[Figura 21. Login en aplicación de consola. 34](#_Toc199714632)

[Figura 22. Funciones en aplicación de consola. 34](#_Toc199714633)

[Figura 23. Configuración de la aplicación de escritorio. 35](#_Toc199714634)

[Figura 24. Creación de un nuevo paquete en cliente de escritorio 35](#_Toc199714635)

[Figura 25. Estructura MVC del proyecto de escritorio 36](#_Toc199714636)

[Figura 26. Crear formularios en aplicación de escritorio 36](#_Toc199714637)

[Figura 27. Crear formularios de inicio de sesión en aplicación de escritorio 37](#_Toc199714638)

[Figura 29. Login en aplicación de escritorio. 39](#_Toc199714639)

[Figura 30. Funciones en aplicación de escritorio 39](#_Toc199714640)

[Figura 31. Crear proyecto base de la aplicación web 40](#_Toc199714641)

[Figura 32. Crear modelo MVC en el proyecto de la aplicación web 40](#_Toc199714642)

[Figura 33. Crear controladores en el proyecto de la aplicación web 41](#_Toc199714643)

[Figura 34. Login en aplicación web. 62](#_Toc199714644)

[Figura 35. Registro en aplicación web. 62](#_Toc199714645)

[Figura 36. Menú principal en aplicación web. 63](#_Toc199714646)

[Figura 37. Mis Boletos en aplicación web. 63](#_Toc199714647)

[Figura 38. Vuelos disponibles en aplicación web. 63](#_Toc199714648)

[Figura 39. Compra de Boletos en aplicación web. 64](#_Toc199714649)

[Figura 40. Creación y configuración del proyecto móvil 64](#_Toc199714650)

[Figura 41. Crear modelo MVC en el proyecto de la aplicación móvil 65](#_Toc199714651)

[Figura 42. Login en aplicación móvil. 87](#_Toc199714652)

[Figura 43. Registro en aplicación de móvil 87](#_Toc199714653)

[Figura 44. Funciones en aplicación de móvil 88](#_Toc199714654)

[Figura 45. Funciones en aplicación de móvil 89](#_Toc199714655)

**ÍNDICE DE TABLAS**

[Tabla 1**. Código Boletos** 14](#_Toc199714013)

[Tabla 2. **Código del modelo Ciudad.** 16](#_Toc199714014)

[Tabla 3. **Código del controlador.** 19](#_Toc199714015)

[Tabla 4. **Habilitar CORS en el servidor.** 21](#_Toc199714016)

[Tabla . **pruebas unitarias.** 23](#_Toc199714017)

[Tabla 7. **Código de vista de consola ConsoleView.** 29](#_Toc199714018)

[Tabla 8. **Código del modelo User en cliente consola.** 34](#_Toc199714019)

[Tabla 9. **Código de eventos para la aplicación de escritorio.** 39](#_Toc199714020)

[Tabla 10. **Código BoletoController.js.** 43](#_Toc199714021)

[Tabla 11. **Código CiudadController.js** 44](#_Toc199714022)

[Tabla 12. **Código UsuarioController.js** 44](#_Toc199714023)

[Tabla 13. **Código de VueloController.js** 46](#_Toc199714024)

[Tabla 14. **Código de la vista LoginView.jsx** 46](#_Toc199714025)

[Tabla 15. **Código RegisterView.jsx** 49](#_Toc199714026)

[Tabla 12. **Código UsuarioController.js** 69](#_Toc199714027)

[Tabla 14. **Código de la vista LoginView.jsx** 71](#_Toc199714028)

**INTRODUCCIÓN A LA ARQUITECTURA DE SOFTWARE**

# INTRODUCCIÓN

En el contexto actual del sector aéreo, contar con sistemas eficientes para la búsqueda y reserva de vuelos es fundamental para mejorar la experiencia del usuario. Este proyecto presenta el diseño y desarrollo de un servicio web para Viajecito S.A., que permite a los usuarios buscar vuelos según ciudad de origen, ciudad destino y fecha de salida, así como consultar sus compras realizadas. La solución garantiza una comunicación estandarizada entre clientes y servidor, facilitando la interoperabilidad y optimizando la gestión de información de vuelos y reservas.

# OBJETIVOS

## Objetivo General

* Diseñar e implementar un servicio web para Viajecito S.A. que permita la búsqueda de vuelos y la gestión de compras de los usuarios.

## Objetivos Específicos

* Desarrollar un servidor en Java que exponga servicios para buscar vuelos filtrando por ciudad de origen, ciudad destino y fecha de salida.
* Implementar funcionalidades que permitan a los usuarios visualizar sus compras a través del servicio web.
* Crear clientes que consuman estos servicios, garantizando la correcta comunicación y funcionalidad entre cliente y servidor..

# MARCO TEÓRICO

## Arquitectura de Software

La arquitectura de software se refiere a la estructura organizativa y conceptual que subyace en el diseño y desarrollo de sistemas de software. Es el proceso de definir una solución estructural que cumple con todos los requisitos funcionales y no funcionales, como la escalabilidad, el rendimiento y la seguridad. Su importancia radica en su capacidad para proporcionar un marco de trabajo que guía las decisiones de diseño y desarrollo.

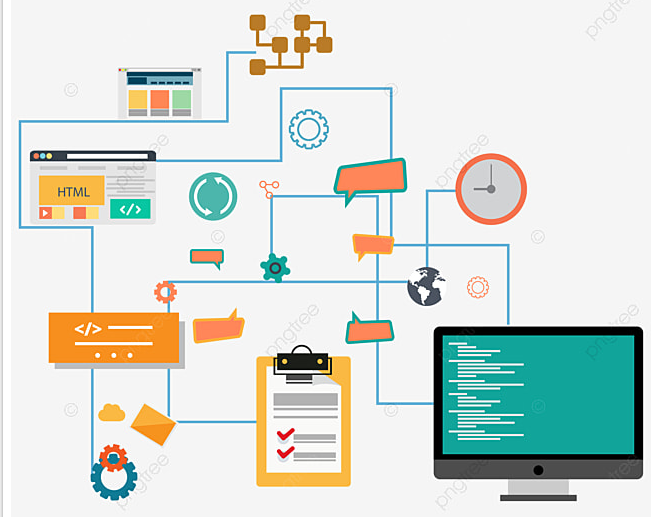


Figura 1. Arquitectura de Software

## Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios es un enfoque moderno y flexible para el diseño de aplicaciones. En lugar de construir una sola aplicación monolítica, se descompone en pequeños servicios independientes. Cada servicio se centra en una tarea específica y se comunica con otros servicios a través de interfaces bien definidas.



Figura 2. Arquitectura de Orientada a Servicios

## Protocolo SOAP

SOAP (Simple Object Access Protocol) es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos. Los mensajes SOAP son las transmisiones de información de remitentes a destinatarios y se pueden combinar para crear patrones de petición/respuesta. SOAP se transporta principalmente a través de HTTP y sistemas de mensajería de middleware, pero también puede utilizar otros protocolos como SMTP y FTP.

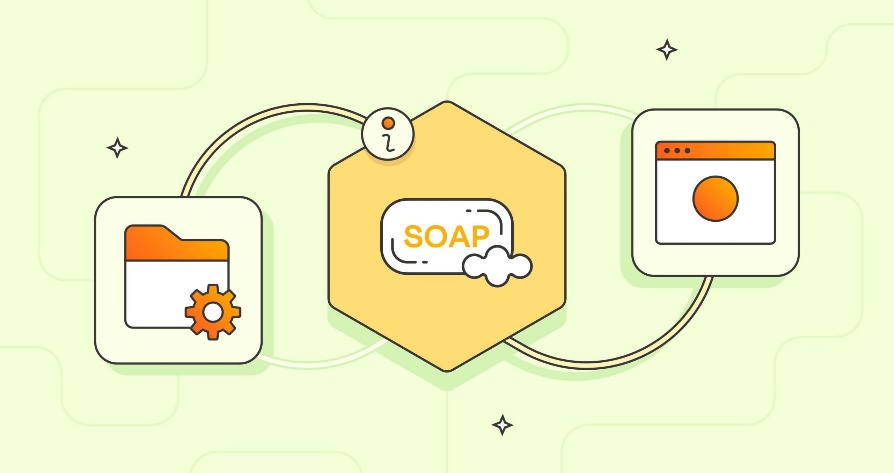


Figura 3. SOAP

## Protocolo REST

RESTful es un estilo de arquitectura para diseñar servicios web que utilizan el protocolo HTTP de manera eficiente y sencilla. En este enfoque, los recursos se representan mediante URLs únicas y se manipulan usando los métodos estándar de HTTP como GET, POST, PUT y DELETE. Los servicios RESTful suelen intercambiar datos en formatos como JSON o XML, lo que facilita la comunicación entre clientes y servidores de diferentes plataformas. Gracias a su simplicidad, escalabilidad y compatibilidad, RESTful se ha convertido en uno de los métodos más populares para construir APIs modernas.

# HERRAMIENTAS

## Aplicación de Consola con Java

Las aplicaciones de consola en Java son programas que se ejecutan en la línea de comandos, ideales para tareas automatizadas o de administración. Java proporciona una sintaxis clara y una amplia biblioteca estándar que facilita el desarrollo de este tipo de aplicaciones.

## Aplicación de Escritorio con Java

Para el desarrollo de aplicaciones de escritorio, Java ofrece bibliotecas como Swing y JavaFX. Swing es una biblioteca más antigua que proporciona componentes gráficos básicos, mientras que JavaFX es una plataforma más moderna que permite la creación de interfaces de usuario ricas y estilizadas, con soporte para multimedia y gráficos avanzados. JavaFX es especialmente adecuado para aplicaciones que requieren una interfaz de usuario atractiva y dinámica.

## Node.js con React para la Aplicación Web

Node.js es un entorno de ejecución para JavaScript que permite el desarrollo del lado del servidor. Su arquitectura orientada a eventos y su modelo de entrada/salida no bloqueante lo hacen ideal para aplicaciones web escalables y en tiempo real. React es una biblioteca de JavaScript desarrollada por Meta (anteriormente Facebook) para la construcción de interfaces de usuario basadas en componentes. La combinación de Node.js y React permite el desarrollo de aplicaciones web modernas, con una separación clara entre el backend y el frontend, y facilita la creación de interfaces de usuario reactivas y eficientes.

## React Native para la Aplicación Móvil

React Native es un framework de código abierto desarrollado por Meta que permite el desarrollo de aplicaciones móviles nativas para iOS y Android utilizando JavaScript y React. Ofrece la ventaja de compartir una base de código entre plataformas, reduciendo el tiempo y los recursos necesarios para el desarrollo. React Native proporciona componentes nativos y acceso a las funcionalidades del dispositivo, lo que resulta en aplicaciones con rendimiento cercano al nativo y una experiencia de usuario consistente.

# PARTE PRÁCTICA

## ESTRUCTURA DEL PROYECTO

Se procede a la creación de la estructura del proyecto, la cual es organizada mediante la generación de las carpetas correspondientes. Estas carpetas son definidas de acuerdo con los módulos, funcionalidades o capas que compondrán la aplicación, permitiendo así una disposición clara y mantenible del código fuente desde las etapas iniciales del desarrollo.

Se crea la carpeta raíz la cual contiene todo el proyecto.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 5. Estructura del proyecto carpeta raíz

Se crean las carpetas internas las cuales dividen el proyecto en el lado del servidor, los clientes a usar y la documentación.

Pantalla negra con letras blancas

El contenido generado por IA puede ser incorrecto.

Figura 6. Estructura del proyecto carpetas internas

## SERVIDOR

### CREACION DE LA APLICACIÓN

Para el desarrollo de una aplicación en Java, se utilizó el entorno de desarrollo NetBeans y se configuró el servidor Payara como contenedor de aplicaciones. Como primer paso, se verificó que estuvieran instaladas todas las herramientas necesarias, entre ellas NetBeans con soporte para Jakarta EE, el servidor Payara Server (preferentemente en su versión Full Profile) y el JDK correspondiente a la versión del servidor.

Una vez verificados los requisitos, se procedió a la creación de un nuevo proyecto web. Para ello, se seleccionó la opción "Java with Maven Web Application" desde el asistente de nuevo proyecto.

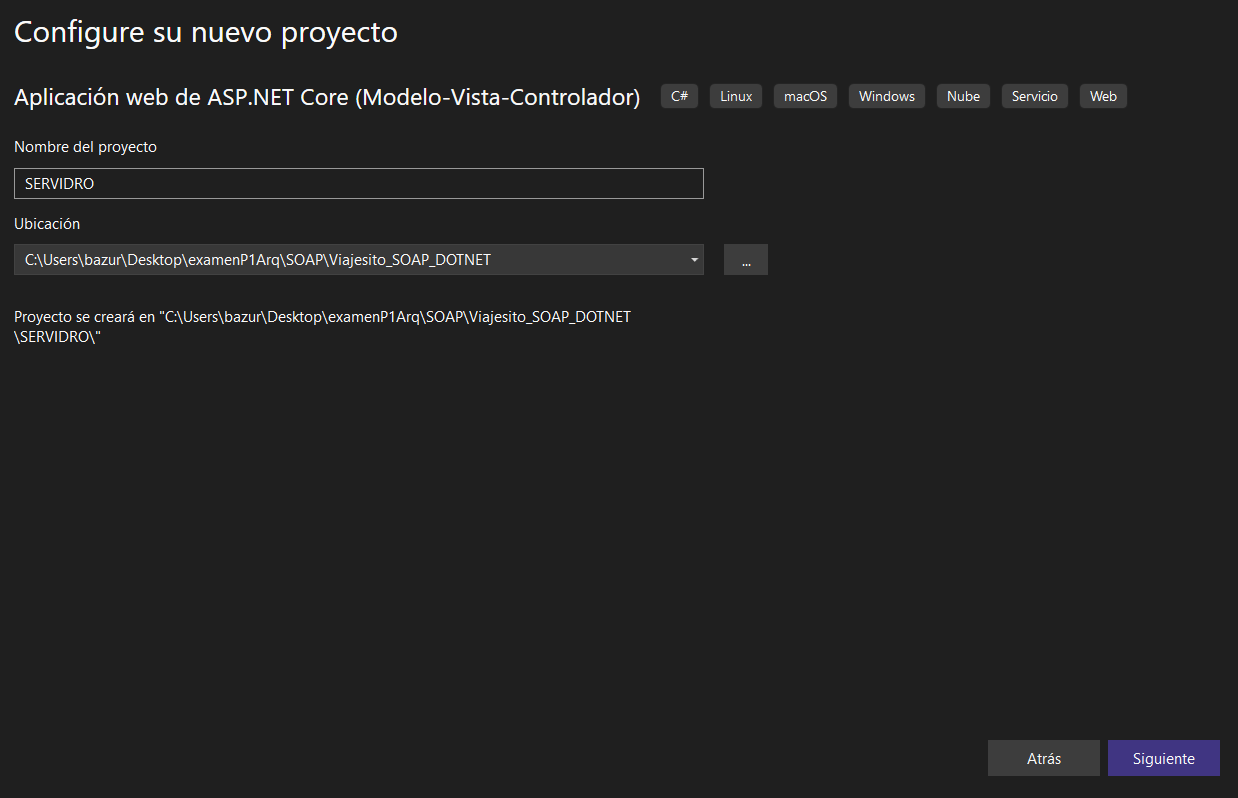


Figura 7. Crear Proyecto en Visual Studio 2022

Se debe asignar un nombre a la aplicación servidor y especificar su ubicación en el sistema de archivos. Para la configuración del servidor, se recomienda seleccionar preferentemente la versión 6 de Payara, y se debe evitar la inclusión de cualquier framework adicional durante esta etapa.

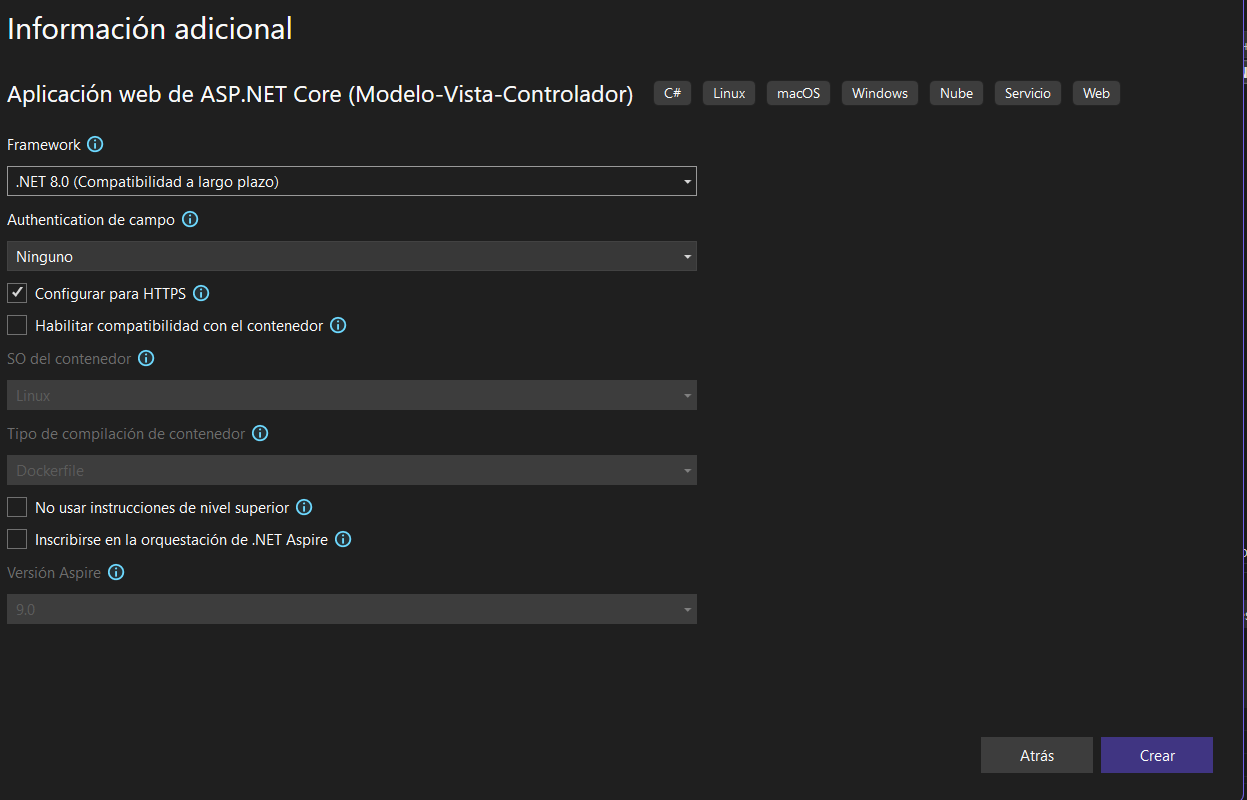


Figura 8. Configuración del Proyecto en Visual Studio 2022

### MODELO MVC

A continuación, será necesario definir la estructura del proyecto mediante la creación de tres paquetes correspondientes a los componentes Controlador, Modelo y Vista. Esta acción debe ser realizada haciendo clic derecho sobre la carpeta "Paquetes de recursos", seleccionando la opción "Nuevo" y posteriormente eligiendo "Paquete Java".

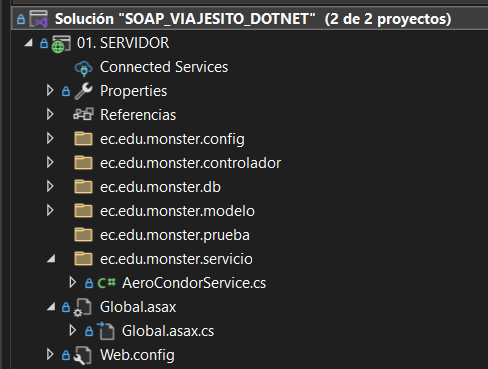


Figura 9. Creación de un nuevo paquete

Cada uno de estos paquetes estará destinado a contener las clases y archivos que representarán las responsabilidades específicas del patrón de arquitectura MVC. En este contexto, el paquete del modelo se encargará de gestionar las clases relacionadas con la lógica de datos y su acceso; el paquete del controlador se encargará de recibir las acciones de la vista y coordinar su ejecución sobre el modelo; mientras que el paquete de vista incluirá los elementos responsables de la interacción con el usuario. Por esta razón, se recomienda asignar a cada paquete un nombre que refleje su función, utilizando un formato como el siguiente: ec.edu.gr03.modelo, ec.edu. gr03.vista, y ec.edu. gr03.controlador.

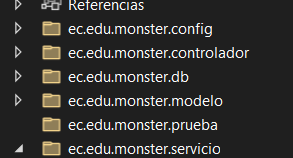


Figura 10. Estructura MVC del proyecto

### IMPLEMENTACIÓN

La lógica correspondiente a los endpoints de aerolíneas condor, se crea primero los modelos y después los endpoints.

Tabla 1**. Código Boletos**

|  |
| --- |
| /\*  \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  \* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template  \*/  package ec.edu.aerolineas\_condor\_server.model;  import jakarta.persistence.Basic;  import jakarta.persistence.Column;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  import jakarta.persistence.JoinColumn;  import jakarta.persistence.ManyToOne;  import jakarta.persistence.NamedQueries;  import jakarta.persistence.NamedQuery;  import jakarta.persistence.Table;  import jakarta.persistence.Temporal;  import jakarta.persistence.TemporalType;  import jakarta.validation.constraints.NotNull;  import jakarta.validation.constraints.Size;  import jakarta.xml.bind.annotation.XmlRootElement;  import java.io.Serializable;  import java.math.BigDecimal;  import java.util.Date;  /\*\*  \*  \* @author Drouet  \*/  @Entity  @Table(name = "boletos")  @XmlRootElement  @NamedQueries({  @NamedQuery(name = "Boletos.findAll", query = "SELECT b FROM Boletos b"),  @NamedQuery(name = "Boletos.findByIdBoleto", query = "SELECT b FROM Boletos b WHERE b.idBoleto = :idBoleto"),  @NamedQuery(name = "Boletos.findByNumeroBoleto", query = "SELECT b FROM Boletos b WHERE b.numeroBoleto = :numeroBoleto"),  @NamedQuery(name = "Boletos.findByFechaCompra", query = "SELECT b FROM Boletos b WHERE b.fechaCompra = :fechaCompra"),  @NamedQuery(name = "Boletos.findByPrecioCompra", query = "SELECT b FROM Boletos b WHERE b.precioCompra = :precioCompra")})  public class Boletos implements Serializable {  private static final long serialVersionUID = 1L;  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Basic(optional = false)  @Column(name = "id\_boleto")  private Integer idBoleto;  @Basic(optional = false)  @NotNull  @Size(min = 1, max = 20)  @Column(name = "numero\_boleto")  private String numeroBoleto;  @Column(name = "fecha\_compra")  @Temporal(TemporalType.TIMESTAMP)  private Date fechaCompra;  // @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation  @Basic(optional = false)  @NotNull  @Column(name = "precio\_compra")  private BigDecimal precioCompra;  @JoinColumn(name = "id\_usuario", referencedColumnName = "id\_usuario")  @ManyToOne(optional = false)  private Usuarios idUsuario;  @JoinColumn(name = "id\_vuelo", referencedColumnName = "id\_vuelo")  @ManyToOne(optional = false)  private Vuelos idVuelo;  public Boletos() {  }  public Boletos(Integer idBoleto) {  this.idBoleto = idBoleto;  }  public Boletos(Integer idBoleto, String numeroBoleto, BigDecimal precioCompra) {  this.idBoleto = idBoleto;  this.numeroBoleto = numeroBoleto;  this.precioCompra = precioCompra;  }  public Integer getIdBoleto() {  return idBoleto;  }  public void setIdBoleto(Integer idBoleto) {  this.idBoleto = idBoleto;  }  public String getNumeroBoleto() {  return numeroBoleto;  }  public void setNumeroBoleto(String numeroBoleto) {  this.numeroBoleto = numeroBoleto;  }  public Date getFechaCompra() {  return fechaCompra;  }  public void setFechaCompra(Date fechaCompra) {  this.fechaCompra = fechaCompra;  }  public BigDecimal getPrecioCompra() {  return precioCompra;  }  public void setPrecioCompra(BigDecimal precioCompra) {  this.precioCompra = precioCompra;  }  public Usuarios getIdUsuario() {  return idUsuario;  }  public void setIdUsuario(Usuarios idUsuario) {  this.idUsuario = idUsuario;  }  public Vuelos getIdVuelo() {  return idVuelo;  }  public void setIdVuelo(Vuelos idVuelo) {  this.idVuelo = idVuelo;  }  @Override  public int hashCode() {  int hash = 0;  hash += (idBoleto != null ? idBoleto.hashCode() : 0);  return hash;  }  @Override  public boolean equals(Object object) {  // TODO: Warning - this method won't work in the case the id fields are not set  if (!(object instanceof Boletos)) {  return false;  }  Boletos other = (Boletos) object;  if ((this.idBoleto == null && other.idBoleto != null) || (this.idBoleto != null && !this.idBoleto.equals(other.idBoleto))) {  return false;  }  return true;  }  @Override  public String toString() {  return "ec.edu.aerolineas\_condor\_server.model.Boletos[ idBoleto=" + idBoleto + " ]";  }    } |

Se procede a implementar el modelo User para realizar el login en la aplicación

Tabla 2. **Código del modelo Ciudad.**

|  |
| --- |
| /\*  \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  \* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template  \*/  package ec.edu.aerolineas\_condor\_server.model;  import jakarta.json.bind.annotation.JsonbTransient;  import jakarta.persistence.Basic;  import jakarta.persistence.CascadeType;  import jakarta.persistence.Column;  import jakarta.persistence.Entity;  import jakarta.persistence.GeneratedValue;  import jakarta.persistence.GenerationType;  import jakarta.persistence.Id;  import jakarta.persistence.NamedQueries;  import jakarta.persistence.NamedQuery;  import jakarta.persistence.OneToMany;  import jakarta.persistence.Table;  import jakarta.validation.constraints.NotNull;  import jakarta.validation.constraints.Size;  import jakarta.xml.bind.annotation.XmlRootElement;  import jakarta.xml.bind.annotation.XmlTransient;  import java.io.Serializable;  import java.util.Collection;  /\*\*  \*  \* @author Drouet  \*/  @Entity  @Table(name = "ciudades")  @XmlRootElement  @NamedQueries({  @NamedQuery(name = "Ciudades.findAll", query = "SELECT c FROM Ciudades c"),  @NamedQuery(name = "Ciudades.findByIdCiudad", query = "SELECT c FROM Ciudades c WHERE c.idCiudad = :idCiudad"),  @NamedQuery(name = "Ciudades.findByCodigoCiudad", query = "SELECT c FROM Ciudades c WHERE c.codigoCiudad = :codigoCiudad"),  @NamedQuery(name = "Ciudades.findByNombreCiudad", query = "SELECT c FROM Ciudades c WHERE c.nombreCiudad = :nombreCiudad")})  public class Ciudades implements Serializable {  private static final long serialVersionUID = 1L;  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  @Basic(optional = false)  @Column(name = "id\_ciudad")  private Integer idCiudad;  @Basic(optional = false)  @NotNull  @Size(min = 1, max = 3)  @Column(name = "codigo\_ciudad")  private String codigoCiudad;  @Basic(optional = false)  @NotNull  @Size(min = 1, max = 100)  @Column(name = "nombre\_ciudad")  private String nombreCiudad;  @OneToMany(cascade = CascadeType.ALL, mappedBy = "idCiudadOrigen")  private Collection<Vuelos> vuelosCollection;  @OneToMany(cascade = CascadeType.ALL, mappedBy = "idCiudadDestino")  private Collection<Vuelos> vuelosCollection1;  public Ciudades() {  }  public Ciudades(Integer idCiudad) {  this.idCiudad = idCiudad;  }  public Ciudades(Integer idCiudad, String codigoCiudad, String nombreCiudad) {  this.idCiudad = idCiudad;  this.codigoCiudad = codigoCiudad;  this.nombreCiudad = nombreCiudad;  }  public Integer getIdCiudad() {  return idCiudad;  }  public void setIdCiudad(Integer idCiudad) {  this.idCiudad = idCiudad;  }  public String getCodigoCiudad() {  return codigoCiudad;  }  public void setCodigoCiudad(String codigoCiudad) {  this.codigoCiudad = codigoCiudad;  }  public String getNombreCiudad() {  return nombreCiudad;  }  public void setNombreCiudad(String nombreCiudad) {  this.nombreCiudad = nombreCiudad;  }  @XmlTransient  @JsonbTransient  public Collection<Vuelos> getVuelosCollection() {  return vuelosCollection;  }  public void setVuelosCollection(Collection<Vuelos> vuelosCollection) {  this.vuelosCollection = vuelosCollection;  }  @XmlTransient  @JsonbTransient  public Collection<Vuelos> getVuelosCollection1() {  return vuelosCollection1;  }  public void setVuelosCollection1(Collection<Vuelos> vuelosCollection1) {  this.vuelosCollection1 = vuelosCollection1;  }  @Override  public int hashCode() {  int hash = 0;  hash += (idCiudad != null ? idCiudad.hashCode() : 0);  return hash;  }  @Override  public boolean equals(Object object) {  // TODO: Warning - this method won't work in the case the id fields are not set  if (!(object instanceof Ciudades)) {  return false;  }  Ciudades other = (Ciudades) object;  if ((this.idCiudad == null && other.idCiudad != null) || (this.idCiudad != null && !this.idCiudad.equals(other.idCiudad))) {  return false;  }  return true;  }  @Override  public String toString() {  return "ec.edu.aerolineas\_condor\_server.model.Ciudades[ idCiudad=" + idCiudad + " ]";  }    } |

Se procede a implementar el controlador para permitir el inicio de sesión llamado Login.

Tabla 3. **Código del controlador.**

|  |
| --- |
| using System; using System.Collections.Generic; using System.Data.SqlClient; using ec.edu.monster.controlador; using ec.edu.monster.db; using ec.edu.monster.modelo;  namespace ec.edu.monster.servicio { public class AeroCondorController : IAeroCondorController { public List GetCiudades() { var lista = new List(); using (var cn = ConexionBD.ObtenerConexion()) { var cmd = new SqlCommand("SELECT \* FROM ciudades", cn); var dr = cmd.ExecuteReader(); while (dr.Read()) { lista.Add(new Ciudades { IdCiudad = (int)dr["id\_ciudad"], CodigoCiudad = dr["codigo\_ciudad"].ToString(), NombreCiudad = dr["nombre\_ciudad"].ToString() }); } } return lista; }  public List<Vuelos> GetVuelos()  {  var lista = new List<Vuelos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM vuelos", cn);  var dr = cmd.ExecuteReader();   var vuelosTemp = new List<Vuelos>();  while (dr.Read())  {  vuelosTemp.Add(new Vuelos  {  IdVuelo = (int)dr["id\_vuelo"],  CodigoVuelo = dr["codigo\_vuelo"].ToString(),  Valor = (decimal)dr["valor"],  HoraSalida = (DateTime)dr["hora\_salida"],  Capacidad = (int)dr["capacidad"],  IdCiudadOrigen = (int)dr["id\_ciudad\_origen"],  IdCiudadDestino = (int)dr["id\_ciudad\_destino"]  });  }   dr.Close();   foreach (var vuelo in vuelosTemp)  {  int vendidos = ObtenerBoletosVendidos(vuelo.IdVuelo);  vuelo.Disponibles = vuelo.Capacidad - vendidos;  lista.Add(vuelo);  }  }  return lista;  }    public List<Usuarios> GetUsuarios()  {  var lista = new List<Usuarios>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM usuarios", cn);  var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Usuarios  {  IdUsuario = (int)dr["id\_usuario"],  Nombre = dr["nombre"].ToString(),  Username = dr["username"].ToString(),  Password = dr["password"].ToString(),  Telefono = dr["telefono"].ToString()  });  }  }  return lista;  }   public List<Boletos> GetBoletos()  {  var lista = new List<Boletos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM boletos", cn);  var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Boletos  {  IdBoleto = (int)dr["id\_boleto"],  NumeroBoleto = dr["numero\_boleto"].ToString(),  FechaCompra = (DateTime)dr["fecha\_compra"],  PrecioCompra = (decimal)dr["precio\_compra"],  IdUsuario = (int)dr["id\_usuario"],  IdVuelo = (int)dr["id\_vuelo"]  });  }  }  return lista;  }  public Usuarios ObtenerUsuarioPorId(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM usuarios WHERE id\_usuario = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  var dr = cmd.ExecuteReader();  if (dr.Read())  {  return new Usuarios  {  IdUsuario = (int)dr["id\_usuario"],  Nombre = dr["nombre"].ToString(),  Username = dr["username"].ToString(),  Password = dr["password"].ToString(),  Telefono = dr["telefono"].ToString()  };  }  }  return null;  }   public bool CrearUsuario(Usuarios usuario)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  INSERT INTO usuarios (nombre, username, password, telefono)   VALUES (@nombre, @username, @password, @telefono)", cn);   cmd.Parameters.AddWithValue("@nombre", usuario.Nombre);  cmd.Parameters.AddWithValue("@username", usuario.Username);  cmd.Parameters.AddWithValue("@password", usuario.Password);  cmd.Parameters.AddWithValue("@telefono", usuario.Telefono ?? "");   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EditarUsuario(Usuarios usuario)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  UPDATE usuarios   SET nombre = @nombre, username = @username, password = @password, telefono = @telefono   WHERE id\_usuario = @id", cn);   cmd.Parameters.AddWithValue("@nombre", usuario.Nombre);  cmd.Parameters.AddWithValue("@username", usuario.Username);  cmd.Parameters.AddWithValue("@password", usuario.Password);  cmd.Parameters.AddWithValue("@telefono", usuario.Telefono ?? "");  cmd.Parameters.AddWithValue("@id", usuario.IdUsuario);   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EliminarUsuario(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("DELETE FROM usuarios WHERE id\_usuario = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  return cmd.ExecuteNonQuery() > 0;  }  }   public int ContarUsuarios()  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT COUNT(\*) FROM usuarios", cn);  return (int)cmd.ExecuteScalar();  }  }   public List<Usuarios> ListarUsuariosPorRango(int desde, int hasta)  {  var lista = new List<Usuarios>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  SELECT \* FROM (  SELECT ROW\_NUMBER() OVER (ORDER BY id\_usuario) AS RowNum, \*   FROM usuarios  ) AS Sub  WHERE RowNum BETWEEN @desde AND @hasta", cn);   cmd.Parameters.AddWithValue("@desde", desde);  cmd.Parameters.AddWithValue("@hasta", hasta);   var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Usuarios  {  IdUsuario = (int)dr["id\_usuario"],  Nombre = dr["nombre"].ToString(),  Username = dr["username"].ToString(),  Password = dr["password"].ToString(),  Telefono = dr["telefono"].ToString()  });  }  }  return lista;  }    public bool Comprar(CompraBoletoRequest request)  {  using (var cn = ConexionBD.ObtenerConexion())  using (var trans = cn.BeginTransaction()) // Añadimos transacción  {  try  {  for (int i = 0; i < request.Cantidad; i++)  {  var cmd = new SqlCommand(@"  INSERT INTO boletos (numero\_boleto, id\_vuelo, id\_usuario, precio\_compra)  VALUES (@num, @vuelo, @usuario,   (SELECT valor FROM vuelos WHERE id\_vuelo = @vuelo))", cn, trans);   cmd.Parameters.AddWithValue("@num", Guid.NewGuid().ToString().Substring(0, 10).ToUpper());  cmd.Parameters.AddWithValue("@vuelo", request.IdVuelo);  cmd.Parameters.AddWithValue("@usuario", request.IdUsuario);   if (cmd.ExecuteNonQuery() <= 0)  {  trans.Rollback();  return false;  }  }   var updateCmd = new SqlCommand(@"  UPDATE vuelos   SET disponibles = disponibles - @cantidad   WHERE id\_vuelo = @vuelo AND disponibles >= @cantidad", cn, trans);   updateCmd.Parameters.AddWithValue("@cantidad", request.Cantidad);  updateCmd.Parameters.AddWithValue("@vuelo", request.IdVuelo);   if (updateCmd.ExecuteNonQuery() <= 0)  {  trans.Rollback();  return false;  }   trans.Commit();  return true;  }  catch  {  trans.Rollback();  return false;  }  }  }    public Usuarios Login(string username, string password)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM usuarios WHERE username = @u AND password = @p", cn);  cmd.Parameters.AddWithValue("@u", username);  cmd.Parameters.AddWithValue("@p", password);  var dr = cmd.ExecuteReader();  if (dr.Read())  {  return new Usuarios  {  IdUsuario = (int)dr["id\_usuario"],  Nombre = dr["nombre"].ToString(),  Username = dr["username"].ToString(),  Password = dr["password"].ToString(),  Telefono = dr["telefono"].ToString()  };  }  }  return null;  }   public Vuelos ObtenerVueloPorId(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM vuelos WHERE id\_vuelo = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  var dr = cmd.ExecuteReader();  if (dr.Read())  {  return new Vuelos  {  IdVuelo = (int)dr["id\_vuelo"],  CodigoVuelo = dr["codigo\_vuelo"].ToString(),  Valor = (decimal)dr["valor"],  HoraSalida = (DateTime)dr["hora\_salida"],  Capacidad = (int)dr["capacidad"],  Disponibles = (int)dr["disponibles"],  IdCiudadOrigen = (int)dr["id\_ciudad\_origen"],  IdCiudadDestino = (int)dr["id\_ciudad\_destino"]  };  }  }  return null;  }   public bool CrearVuelo(Vuelos vuelo)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  INSERT INTO vuelos (codigo\_vuelo, valor, hora\_salida, capacidad, disponibles, id\_ciudad\_origen, id\_ciudad\_destino)  VALUES (@codigo, @valor, @hora, @capacidad, @disponibles, @origen, @destino)", cn);   cmd.Parameters.AddWithValue("@codigo", vuelo.CodigoVuelo);  cmd.Parameters.AddWithValue("@valor", vuelo.Valor);  cmd.Parameters.AddWithValue("@hora", vuelo.HoraSalida);  cmd.Parameters.AddWithValue("@capacidad", vuelo.Capacidad);  cmd.Parameters.AddWithValue("@disponibles", vuelo.Disponibles);  cmd.Parameters.AddWithValue("@origen", vuelo.IdCiudadOrigen);  cmd.Parameters.AddWithValue("@destino", vuelo.IdCiudadDestino);   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EditarVuelo(Vuelos vuelo)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  UPDATE vuelos  SET codigo\_vuelo = @codigo, valor = @valor, hora\_salida = @hora,  capacidad = @capacidad, disponibles = @disponibles,  id\_ciudad\_origen = @origen, id\_ciudad\_destino = @destino  WHERE id\_vuelo = @id", cn);   cmd.Parameters.AddWithValue("@codigo", vuelo.CodigoVuelo);  cmd.Parameters.AddWithValue("@valor", vuelo.Valor);  cmd.Parameters.AddWithValue("@hora", vuelo.HoraSalida);  cmd.Parameters.AddWithValue("@capacidad", vuelo.Capacidad);  cmd.Parameters.AddWithValue("@disponibles", vuelo.Disponibles);  cmd.Parameters.AddWithValue("@origen", vuelo.IdCiudadOrigen);  cmd.Parameters.AddWithValue("@destino", vuelo.IdCiudadDestino);  cmd.Parameters.AddWithValue("@id", vuelo.IdVuelo);   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EliminarVuelo(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("DELETE FROM vuelos WHERE id\_vuelo = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  return cmd.ExecuteNonQuery() > 0;  }  }   public int ContarVuelos()  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT COUNT(\*) FROM vuelos", cn);  return (int)cmd.ExecuteScalar();  }  }   public List<Vuelos> ListarVuelosPorRango(int desde, int hasta)  {  var lista = new List<Vuelos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  SELECT \* FROM (  SELECT \*, ROW\_NUMBER() OVER (ORDER BY id\_vuelo) AS fila  FROM vuelos  ) AS sub  WHERE fila BETWEEN @desde AND @hasta", cn);   cmd.Parameters.AddWithValue("@desde", desde);  cmd.Parameters.AddWithValue("@hasta", hasta);   var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Vuelos  {  IdVuelo = (int)dr["id\_vuelo"],  CodigoVuelo = dr["codigo\_vuelo"].ToString(),  Valor = (decimal)dr["valor"],  HoraSalida = (DateTime)dr["hora\_salida"],  Capacidad = (int)dr["capacidad"],  Disponibles = (int)dr["disponibles"],  IdCiudadOrigen = (int)dr["id\_ciudad\_origen"],  IdCiudadDestino = (int)dr["id\_ciudad\_destino"]  });  }  }  return lista;  }   public Ciudades ObtenerCiudadPorId(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM ciudades WHERE id\_ciudad = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  var dr = cmd.ExecuteReader();  if (dr.Read())  {  return new Ciudades  {  IdCiudad = (int)dr["id\_ciudad"],  CodigoCiudad = dr["codigo\_ciudad"].ToString(),  NombreCiudad = dr["nombre\_ciudad"].ToString()  };  }  }  return null;  }   public bool CrearCiudad(Ciudades ciudad)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("INSERT INTO ciudades (codigo\_ciudad, nombre\_ciudad) VALUES (@codigo, @nombre)", cn);  cmd.Parameters.AddWithValue("@codigo", ciudad.CodigoCiudad);  cmd.Parameters.AddWithValue("@nombre", ciudad.NombreCiudad);  return cmd.ExecuteNonQuery() > 0;  }  }   public bool EditarCiudad(Ciudades ciudad)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("UPDATE ciudades SET codigo\_ciudad = @codigo, nombre\_ciudad = @nombre WHERE id\_ciudad = @id", cn);  cmd.Parameters.AddWithValue("@codigo", ciudad.CodigoCiudad);  cmd.Parameters.AddWithValue("@nombre", ciudad.NombreCiudad);  cmd.Parameters.AddWithValue("@id", ciudad.IdCiudad);  return cmd.ExecuteNonQuery() > 0;  }  }   public bool EliminarCiudad(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("DELETE FROM ciudades WHERE id\_ciudad = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  return cmd.ExecuteNonQuery() > 0;  }  }   public int ContarCiudades()  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT COUNT(\*) FROM ciudades", cn);  return (int)cmd.ExecuteScalar();  }  }   public List<Ciudades> ListarCiudadesPorRango(int desde, int hasta)  {  var lista = new List<Ciudades>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  SELECT \* FROM (  SELECT \*, ROW\_NUMBER() OVER (ORDER BY id\_ciudad) AS fila  FROM ciudades  ) AS sub  WHERE fila BETWEEN @desde AND @hasta", cn);   cmd.Parameters.AddWithValue("@desde", desde);  cmd.Parameters.AddWithValue("@hasta", hasta);   var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Ciudades  {  IdCiudad = (int)dr["id\_ciudad"],  CodigoCiudad = dr["codigo\_ciudad"].ToString(),  NombreCiudad = dr["nombre\_ciudad"].ToString()  });  }  }  return lista;  }  public Boletos ObtenerBoletoPorId(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM boletos WHERE id\_boleto = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  var dr = cmd.ExecuteReader();  if (dr.Read())  {  return new Boletos  {  IdBoleto = (int)dr["id\_boleto"],  NumeroBoleto = dr["numero\_boleto"].ToString(),  FechaCompra = (DateTime)dr["fecha\_compra"],  PrecioCompra = (decimal)dr["precio\_compra"],  IdUsuario = (int)dr["id\_usuario"],  IdVuelo = (int)dr["id\_vuelo"]  };  }  }  return null;  }   public bool CrearBoleto(Boletos boleto)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"INSERT INTO boletos   (numero\_boleto, fecha\_compra, precio\_compra, id\_usuario, id\_vuelo)   VALUES (@num, @fecha, @precio, @usuario, @vuelo)", cn);   cmd.Parameters.AddWithValue("@num", boleto.NumeroBoleto);  cmd.Parameters.AddWithValue("@fecha", boleto.FechaCompra);  cmd.Parameters.AddWithValue("@precio", boleto.PrecioCompra);  cmd.Parameters.AddWithValue("@usuario", boleto.IdUsuario);  cmd.Parameters.AddWithValue("@vuelo", boleto.IdVuelo);   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EditarBoleto(Boletos boleto)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"UPDATE boletos SET   numero\_boleto = @num, fecha\_compra = @fecha, precio\_compra = @precio,  id\_usuario = @usuario, id\_vuelo = @vuelo WHERE id\_boleto = @id", cn);   cmd.Parameters.AddWithValue("@num", boleto.NumeroBoleto);  cmd.Parameters.AddWithValue("@fecha", boleto.FechaCompra);  cmd.Parameters.AddWithValue("@precio", boleto.PrecioCompra);  cmd.Parameters.AddWithValue("@usuario", boleto.IdUsuario);  cmd.Parameters.AddWithValue("@vuelo", boleto.IdVuelo);  cmd.Parameters.AddWithValue("@id", boleto.IdBoleto);   return cmd.ExecuteNonQuery() > 0;  }  }   public bool EliminarBoleto(int id)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("DELETE FROM boletos WHERE id\_boleto = @id", cn);  cmd.Parameters.AddWithValue("@id", id);  return cmd.ExecuteNonQuery() > 0;  }  }   public int ContarBoletos()  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT COUNT(\*) FROM boletos", cn);  return (int)cmd.ExecuteScalar();  }  }   public List<Boletos> ListarBoletosPorRango(int desde, int hasta)  {  var lista = new List<Boletos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  SELECT \* FROM (  SELECT \*, ROW\_NUMBER() OVER (ORDER BY id\_boleto) AS fila  FROM boletos  ) AS sub  WHERE fila BETWEEN @desde AND @hasta", cn);   cmd.Parameters.AddWithValue("@desde", desde);  cmd.Parameters.AddWithValue("@hasta", hasta);   var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Boletos  {  IdBoleto = (int)dr["id\_boleto"],  NumeroBoleto = dr["numero\_boleto"].ToString(),  FechaCompra = (DateTime)dr["fecha\_compra"],  PrecioCompra = (decimal)dr["precio\_compra"],  IdUsuario = (int)dr["id\_usuario"],  IdVuelo = (int)dr["id\_vuelo"]  });  }  }  return lista;  }   public List<Boletos> ObtenerBoletosPorUsuario(int idUsuario)  {  var lista = new List<Boletos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT \* FROM boletos WHERE id\_usuario = @id", cn);  cmd.Parameters.AddWithValue("@id", idUsuario);  var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Boletos  {  IdBoleto = (int)dr["id\_boleto"],  NumeroBoleto = dr["numero\_boleto"].ToString(),  FechaCompra = (DateTime)dr["fecha\_compra"],  PrecioCompra = (decimal)dr["precio\_compra"],  IdUsuario = (int)dr["id\_usuario"],  IdVuelo = (int)dr["id\_vuelo"]  });  }  }  return lista;  }  public int ObtenerBoletosVendidos(int idVuelo)  {  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand("SELECT COUNT(\*) FROM boletos WHERE id\_vuelo = @id", cn);  cmd.Parameters.AddWithValue("@id", idVuelo);  return (int)cmd.ExecuteScalar();  }  }     public List<Vuelos> BuscarVuelos(string origen, string destino, DateTime fechaSalida)  {  var lista = new List<Vuelos>();  using (var cn = ConexionBD.ObtenerConexion())  {  var cmd = new SqlCommand(@"  SELECT \* FROM vuelos   WHERE id\_ciudad\_origen = (SELECT id\_ciudad FROM ciudades WHERE codigo\_ciudad = @origen)  AND id\_ciudad\_destino = (SELECT id\_ciudad FROM ciudades WHERE codigo\_ciudad = @destino)  AND CAST(hora\_salida AS DATE) = @fecha  ORDER BY valor DESC", cn);   cmd.Parameters.AddWithValue("@origen", origen);  cmd.Parameters.AddWithValue("@destino", destino);  cmd.Parameters.AddWithValue("@fecha", fechaSalida.Date);   var dr = cmd.ExecuteReader();  while (dr.Read())  {  lista.Add(new Vuelos  {  IdVuelo = (int)dr["id\_vuelo"],  CodigoVuelo = dr["codigo\_vuelo"].ToString(),  Valor = (decimal)dr["valor"],  HoraSalida = (DateTime)dr["hora\_salida"],  Capacidad = (int)dr["capacidad"],  Disponibles = (int)dr["disponibles"],  IdCiudadOrigen = (int)dr["id\_ciudad\_origen"],  IdCiudadDestino = (int)dr["id\_ciudad\_destino"]  });  }  }  return lista;  } }  } |

### HABILITAR CORS

Para consumir el Web Service desde aplicación web o móviles se requiere configurar CORS en el servidor de modo que permita las peticiones desde estos tipos de clientes para ello se procede a crear un paquete llamado ec.edu.gr03.filtro y la clase CORSFilter.java, aquí se define las reglas para permitir la conexión de los clientes mencionados.

Tabla 4. **Habilitar CORS en el servidor.**

|  |
| --- |
| <appSettings>  <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" /> </appSettings>  <system.web>  <compilation debug="true" targetFramework="4.7.2" />  <httpRuntime targetFramework="4.7.2"/> </system.web>  <system.serviceModel>  <behaviors>  <serviceBehaviors>  <behavior>  <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>  <serviceDebug includeExceptionDetailInFaults="false"/>  </behavior>  </serviceBehaviors>  </behaviors>  <protocolMapping>  <add binding="basicHttpsBinding" scheme="https" />  </protocolMapping>  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" multipleSiteBindingsEnabled="true" /> </system.serviceModel>  <system.webServer>  <modules runAllManagedModulesForAllRequests="true" />   <!-- CORS -->  <httpProtocol>  <customHeaders>  <add name="Access-Control-Allow-Origin" value="\*" />  <add name="Access-Control-Allow-Methods" value="GET, POST, OPTIONS" />  <add name="Access-Control-Allow-Headers" value="Content-Type, SOAPAction" />  </customHeaders>  </httpProtocol>   <directoryBrowse enabled="true"/> </system.webServer> |

### PRUEBAS UNITARIAS

Ahora se procede a crear las pruebas unitarias de los Web Service como se indica en la figura.

A continuación, se muestra el código de las pruebas

Tabla . **pruebas unitarias.**

|  |
| --- |
| /\*  \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  \* Click nbfs://nbhost/SystemFileSystem/Templates/UnitTests/JUnit5TestClass.java to edit this template  \*/  package ec.edu.aerolineas\_condor\_server.controller;  import ec.edu.aerolineas\_condor\_server.model.Ciudades;  import jakarta.persistence.EntityManager;  import java.util.List;  import org.junit.jupiter.api.AfterEach;  import org.junit.jupiter.api.AfterAll;  import org.junit.jupiter.api.BeforeEach;  import org.junit.jupiter.api.BeforeAll;  import org.junit.jupiter.api.Test;  import static org.junit.jupiter.api.Assertions.\*;  /\*\*  \*  \* @author Drouet  \*/  public class CiudadesFacadeRESTTest {    public CiudadesFacadeRESTTest() {  }    @BeforeAll  public static void setUpClass() {  }    @AfterAll  public static void tearDownClass() {  }    @BeforeEach  public void setUp() {  }    @AfterEach  public void tearDown() {  }  /\*\*  \* Test of create method, of class CiudadesFacadeREST.  \*/  @Test  public void testCreate() {  System.out.println("create");  Ciudades entity = null;  CiudadesFacadeREST instance = new CiudadesFacadeREST();  assertEquals(true, true);  }  /\*\*  \* Test of edit method, of class CiudadesFacadeREST.  \*/  @Test  public void testEdit() {  System.out.println("edit");  Integer id = null;  Ciudades entity = null;  CiudadesFacadeREST instance = new CiudadesFacadeREST();  assertEquals(true, true);  }  /\*\*  \* Test of remove method, of class CiudadesFacadeREST.  \*/  @Test  public void testRemove() {  System.out.println("remove");  Integer id = null;  CiudadesFacadeREST instance = new CiudadesFacadeREST();  assertEquals(true, true);  }  /\*\*  \* Test of find method, of class CiudadesFacadeREST.  \*/  @Test  public void testFind() {  System.out.println("find");  Integer id = null;  CiudadesFacadeREST instance = new CiudadesFacadeREST();  Ciudades expResult = null;  assertEquals(true, true);  }  /\*\*  \* Test of findAll method, of class CiudadesFacadeREST.  \*/  @Test  public void testFindAll() {  System.out.println("findAll");  CiudadesFacadeREST instance = new CiudadesFacadeREST();  assertEquals(true, true);  }  /\*\*  \* Test of findRange method, of class CiudadesFacadeREST.  \*/  @Test  public void testFindRange() {  System.out.println("findRange");  assertEquals(true, true);  }  /\*\*  \* Test of countREST method, of class CiudadesFacadeREST.  \*/  @Test  public void testCountREST() {  System.out.println("countREST");  CiudadesFacadeREST instance = new CiudadesFacadeREST();  String expResult = "";  assertEquals(true, true);  }  /\*\*  \* Test of getEntityManager method, of class CiudadesFacadeREST.  \*/  @Test  public void testGetEntityManager() {  System.out.println("getEntityManager");  CiudadesFacadeREST instance = new CiudadesFacadeREST();  assertEquals(true, true);  }    } |

### EJECUCIÓN DE LA APLICACIÓN

A continuación, se procede a con la ejecución del servidor SOAP para habilitar los servicios web.

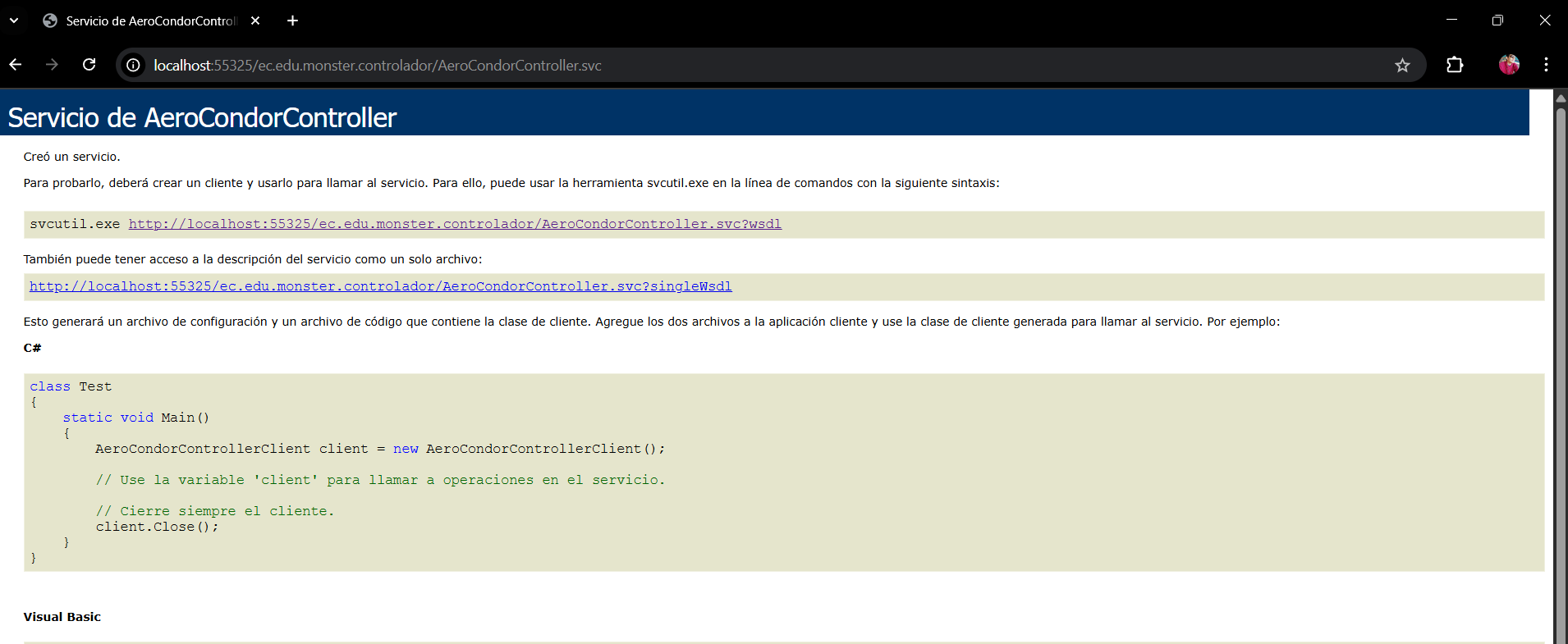


Figura 13. Ejecución del servidor SOAP

## CLIENTE DE CONSOLA

### CREACION DE LA APLICACIÓN

Para la creación de la aplicación de consola se procede a crear un nuevo proyecto de .net de tipo aplicación de consola, para la creación de una aplicación de consola, este proyecto debe estar en la carpeta 02. CLICON.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Figura 14. Configuración de la aplicación de consola

### MODELO MVC

A continuación, será necesario definir la estructura del proyecto mediante la creación de tres paquetes correspondientes a los componentes Controlador, Modelo y Vista. Esta acción debe ser realizada haciendo clic derecho sobre el proyecto y después crear carpeta.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 15. Creación de un nuevo paquete en cliente de consola

Cada uno de estos paquetes estará destinado a contener las clases y archivos que representarán las responsabilidades específicas del patrón de arquitectura MVC. En este contexto, el paquete del modelo se encargará de gestionar las clases relacionadas con la lógica de datos y su acceso; el paquete del controlador se encargará de recibir las acciones de la vista y coordinar su ejecución sobre el modelo; mientras que el paquete de vista incluirá los elementos responsables de la interacción con el usuario. Por esta razón, se recomienda asignar a cada paquete un nombre que refleje su función, utilizando un formato como el siguiente: ec.edu.gr03.modelo, ec.edu. gr03.vista, y ec.edu. gr03.controlador.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 16. Estructura MVC del proyecto de consola

### IMPLEMENTACIÓN

Se crea la referencia al Web Service REST del servidor como se muestra en la figura, para ello se hace clic derecho sobre el proyecto y luego agregar referencias.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Figura 17. Creación del cliente del Web Service

Si todo sale bien se tiene todas las referencias con las funciones listas para usar.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 18. Creación del cliente del Web Service RESTFUL archivos necesarios

Se procede a crear el ViewController el cual contiene los métodos necesarios de los Web Service y exponiendo el método para ser usado en otro módulo.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Figura 19. Creación de funciones del cliente RESTFUL en el lado del cliente

Se procede a crear el ConsoleView el cual contendrá los métodos necesarios para mostrar al usuario.

Tabla 7. **Código de vista de consola ConsoleView.**

|  |
| --- |
| using System;  using System.Linq;  using System.Threading.Tasks;  using ec.edu.monster.controlador;  using ServiceReference1;  namespace ec.edu.monster.vista  {  public class LoginView  {  private static readonly UsuarioController controller = new UsuarioController();  public static async Task MostrarAsync()  {  while (true)  {  Console.WriteLine("===== BIENVENIDO MONSTER VIAJESITOS =====");  Console.WriteLine("1. Iniciar Sesión");  Console.WriteLine("2. Registrarse");  Console.WriteLine("3. Salir");  Console.Write("Seleccione una opción: ");  string opcion = Console.ReadLine();  switch (opcion)  {  case "1":  await IniciarSesionAsync();  break;  case "2":  await RegistrarseAsync();  break;  case "3":  Console.WriteLine("¡Hasta luego!");  return;  default:  Console.WriteLine("Opción inválida\n");  break;  }  }  }  private static async Task IniciarSesionAsync()  {  Console.Write("Usuario: ");  string username = Console.ReadLine()?.Trim();  Console.Write("Contraseña: ");  string password = Console.ReadLine()?.Trim();  if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))  {  Console.WriteLine("Usuario y contraseña son obligatorios.\n");  return;  }  var usuario = await controller.LoginAsync(username, password);  if (usuario != null && usuario.IdUsuario > 0)  {  Console.WriteLine($"\nInicio de sesión exitoso. Bienvenido {usuario.Nombre}!\n");  await MenuView.MostrarAsync(usuario);  }  else  {  Console.WriteLine("Credenciales inválidas.\n");  }  }  private static async Task RegistrarseAsync()  {  var nuevo = new Usuarios();  Console.Write("Nombre: ");  nuevo.Nombre = Console.ReadLine()?.Trim();  Console.Write("Username: ");  nuevo.Username = Console.ReadLine()?.Trim();  Console.Write("Contraseña: ");  nuevo.Password = Console.ReadLine()?.Trim();  Console.Write("Teléfono: ");  nuevo.Telefono = Console.ReadLine()?.Trim();  // Validaciones  if (string.IsNullOrWhiteSpace(nuevo.Nombre) ||  string.IsNullOrWhiteSpace(nuevo.Username) ||  string.IsNullOrWhiteSpace(nuevo.Password))  {  Console.WriteLine("Nombre, username y contraseña son obligatorios.\n");  return;  }  if (!string.IsNullOrWhiteSpace(nuevo.Telefono))  {  if (!nuevo.Telefono.All(char.IsDigit))  {  Console.WriteLine("El teléfono debe contener solo números.\n");  return;  }  if (nuevo.Telefono.Length != 10)  {  Console.WriteLine("El número de teléfono debe tener exactamente 10 dígitos.\n");  return;  }  }  bool exito = await controller.CrearUsuarioAsync(nuevo);  Console.WriteLine(exito ? "Usuario registrado exitosamente.\n" : "Error al registrar usuario. Puede que el username ya exista.\n");  }  }  }  using System;  using System.Globalization;  using System.Collections.Generic;  using System.Threading.Tasks;  using ServiceReference1;  using ec.edu.monster.controlador;  namespace ec.edu.monster.vista  {  public class MenuView  {  private static readonly VueloController vuelosController = new VueloController();  private static readonly BoletoController boletosController = new BoletoController();  public static async Task MostrarAsync(Usuarios usuario)  {  while (true)  {  Console.WriteLine("\n===== MENÚ PRINCIPAL =====");  Console.WriteLine("1. Ver mis boletos");  Console.WriteLine("2. Ver todos los vuelos");  Console.WriteLine("3. Comprar boletos");  Console.WriteLine("4. Cerrar sesión");  Console.Write("Seleccione una opción: ");  var opcion = Console.ReadLine();  switch (opcion)  {  case "1":  await VerBoletosAsync(usuario);  break;  case "2":  await VerVuelosAsync();  break;  case "3":  await ComprarBoletosView.ComprarAsync(usuario);  break;  case "4":  Console.WriteLine("Sesión cerrada.");  return;  default:  Console.WriteLine("Opción inválida.\n");  break;  }  }  }  private static async Task VerBoletosAsync(Usuarios usuario)  {  List<Boletos> boletos = await boletosController.ObtenerPorUsuarioAsync(usuario.IdUsuario);  Console.WriteLine("\n===== TUS BOLETOS =====");  if (boletos == null || boletos.Count == 0)  {  Console.WriteLine("No tienes boletos registrados.");  return;  }  Console.WriteLine("{0,-18} {1,-10} {2,-22} {3,-10}", "Número Boleto", "ID Vuelo", "Fecha Compra", "Precio");  Console.WriteLine(new string('-', 70));  foreach (var boleto in boletos)  {  string fecha = boleto.FechaCompra?.ToString("yyyy-MM-dd HH:mm") ?? "N/A";  Console.WriteLine("{0,-18} {1,-10} {2,-22} ${3,-8}", boleto.NumeroBoleto, boleto.IdVuelo, fecha, boleto.PrecioCompra);  }  }  private static async Task VerVuelosAsync()  {  List<Vuelos> vuelos = await vuelosController.ObtenerTodosAsync();  Console.WriteLine("\n===== TODOS LOS VUELOS =====");  Console.WriteLine("{0,-5} {1,-10} {2,-10} {3,-10} {4,-20} {5,-10} {6,-10}",  "ID", "Código", "Origen", "Destino", "Hora Salida", "Precio", "Disponibles");  Console.WriteLine(new string('-', 90));  foreach (var vuelo in vuelos)  {  string hora = vuelo.HoraSalida.ToString("yyyy-MM-dd HH:mm", CultureInfo.InvariantCulture);  Console.WriteLine("{0,-5} {1,-10} {2,-10} {3,-10} {4,-20} ${5,-8} {6,-10}",  vuelo.IdVuelo,  vuelo.CodigoVuelo,  vuelo.IdCiudadOrigen,  vuelo.IdCiudadDestino,  hora,  vuelo.Valor,  vuelo.Disponibles);  }  }  }  } |

### EJECUCIÓN DE LA APLICACIÓN

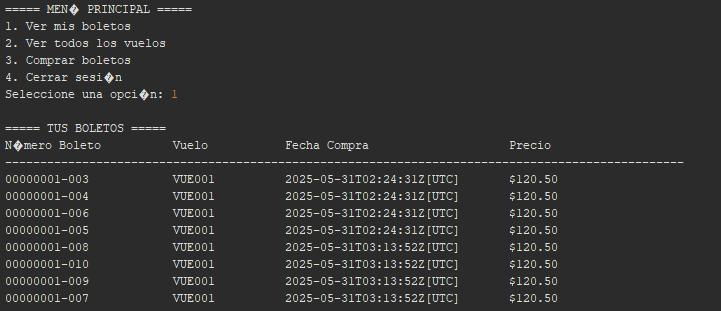
Finalmente se ejecuta la aplicación de consola y se prueba las funcionalidades del cliente RESTFUL, primero se procede a probar el login de la aplicación de consola.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 21. Login en aplicación de consola.

A continuación, se procede a probar las funciones de ver boletos comprados, ver vuelos y comprar vuelo.



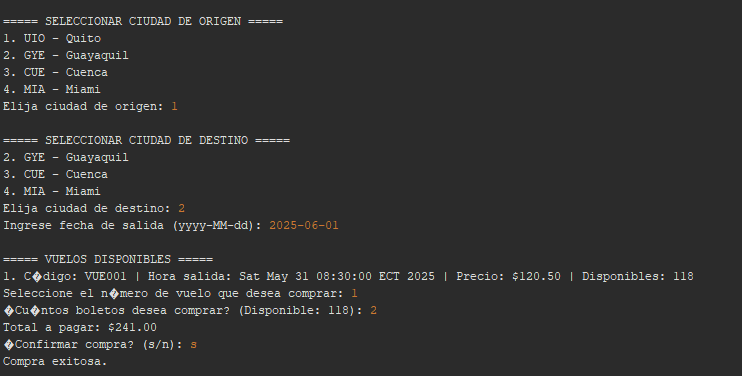


Figura 22. Funciones en aplicación de consola.

## CLIENTE DE ESCRITORIO

### CREACION DE LA APLICACIÓN

Para la creación de la aplicación de consola se procede a crear un nuevo proyecto de Java with Ant, luego se selecciona java application para la creación de una aplicación de consola, este proyecto debe estar en la carpeta 02. CLIESC.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Figura 23. Configuración de la aplicación de escritorio.

### MODELO MVC

A continuación, será necesario definir la estructura del proyecto mediante la creación de tres paquetes correspondientes a los componentes Controlador, Modelo y Vista. Esta acción debe ser realizada haciendo clic derecho sobre la carpeta "Paquetes de recursos", seleccionando la opción "Nuevo" y posteriormente eligiendo "Paquete Java".

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Figura 24. Creación de un nuevo paquete en cliente de escritorio

Cada uno de estos paquetes estará destinado a contener las clases y archivos que representarán las responsabilidades específicas del patrón de arquitectura MVC. En este contexto, el paquete del modelo se encargará de gestionar las clases relacionadas con la lógica de datos y su acceso; el paquete del controlador se encargará de recibir las acciones de la vista y coordinar su ejecución sobre el modelo; mientras que el paquete de vista incluirá los elementos responsables de la interacción con el usuario. Por esta razón, se recomienda asignar a cada paquete un nombre que refleje su función, utilizando un formato como el siguiente: ec.edu.gr03.modelo, ec.edu. gr03.vista, y ec.edu. gr03.controlador.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 25. Estructura MVC del proyecto de escritorio

### IMPLEMENTACIÓN

La implementación inicial con el webservice y la ruta del wsdl se procede a importar las funciones a el proyecto de java

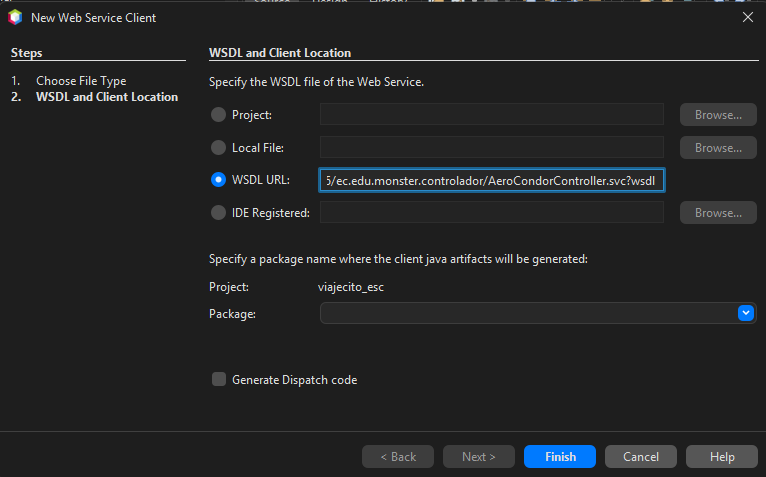


Figura 26. Importar funciones del webservice escritorio

Ahora se procede a crear el client donde se consumirá cada función del webservice

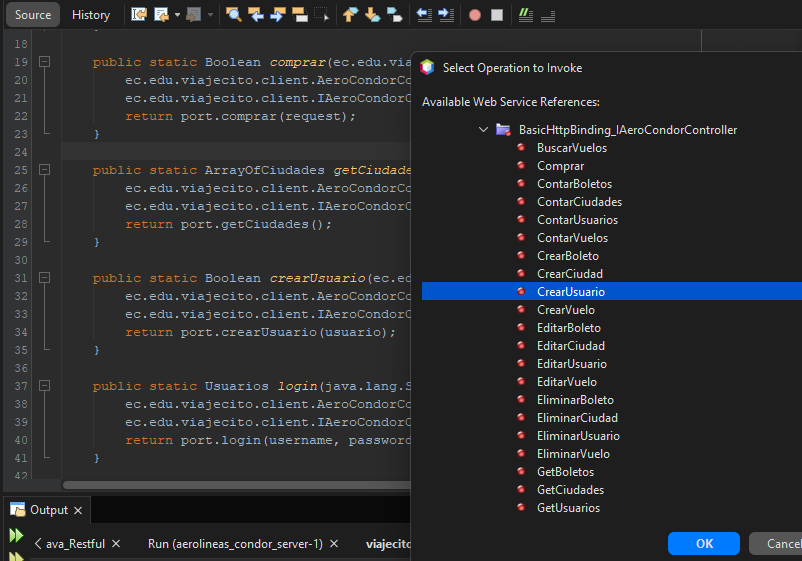


Figura 26. Importar funciones del webservice para poder ser usadas dentro del proyecto de escritorio

Luego se implementa la clase controlador la cual contiene el código que permite interactuar entre el modelo local y el proporcionado por el webservice.

Tabla 7. **Código de controllers de escritorio.**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package ec.edu.viajecito.controller;

import ec.edu.viajecito.client.AeroCondorClient;

import ec.edu.viajecito.client.ArrayOfBoletos;

import ec.edu.viajecito.client.Boletos;

import ec.edu.viajecito.client.Ciudades;

import ec.edu.viajecito.client.Usuarios;

import ec.edu.viajecito.client.Vuelos;

import ec.edu.viajecito.model.Boleto;

import ec.edu.viajecito.model.Ciudad;

import ec.edu.viajecito.model.CompraBoletoRequest;

import ec.edu.viajecito.model.Usuario;

import ec.edu.viajecito.model.Vuelo;

import java.math.BigDecimal;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.List;

/\*\*

\*

\* @author Drouet

\*/

public class BoletosController {

public List<Boleto> obtenerBoletosPorUsuario(String idUsuario) {

ArrayOfBoletos arrayOfBoletos = AeroCondorClient.obtenerBoletosPorUsuario(Integer.valueOf(idUsuario));

List<Boleto> boletos = new ArrayList<>();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");

if (arrayOfBoletos != null && arrayOfBoletos.getBoletos() != null) {

for (Boletos b : arrayOfBoletos.getBoletos()) {

Boleto boleto = new Boleto();

boleto.setIdBoleto(b.getIdBoleto());

boleto.setNumeroBoleto(b.getNumeroBoleto().getValue());

if (b.getFechaCompra() != null) {

boleto.setFechaCompra(sdf.format(b.getFechaCompra().getValue().toGregorianCalendar().getTime()));

} else {

boleto.setFechaCompra(null);

}

boleto.setPrecioCompra(b.getPrecioCompra());

// Usuario

Usuarios u = AeroCondorClient.obtenerUsuarioPorId(b.getIdUsuario());

Usuario usuario = new Usuario();

usuario.setIdUsuario(u.getIdUsuario());

usuario.setNombre(u.getNombre().getValue());

usuario.setUsername(u.getUsername().getValue());

usuario.setPassword(u.getPassword().getValue());

usuario.setTelefono(u.getTelefono().getValue());

boleto.setUsuario(usuario);

// Vuelo

Vuelos v = AeroCondorClient.obtenerVueloPorId(b.getIdVuelo());

Vuelo vuelo = new Vuelo();

vuelo.setIdVuelo(v.getIdVuelo());

vuelo.setCodigoVuelo(v.getCodigoVuelo().getValue());

vuelo.setValor(v.getValor());

vuelo.setHoraSalida(v.getHoraSalida().toGregorianCalendar().getTime());

vuelo.setCapacidad(v.getCapacidad());

vuelo.setDisponibles(v.getDisponibles());

// Ciudad Origen

Ciudades origen = AeroCondorClient.obtenerCiudadPorId(v.getIdCiudadOrigen());

Ciudad ciudadOrigen = new Ciudad();

ciudadOrigen.setIdCiudad(origen.getIdCiudad());

ciudadOrigen.setCodigoCiudad(origen.getCodigoCiudad().getValue());

ciudadOrigen.setNombreCiudad(origen.getNombreCiudad().getValue());

vuelo.setCiudadOrigen(ciudadOrigen);

// Ciudad Destino

Ciudades destino = AeroCondorClient.obtenerCiudadPorId(v.getIdCiudadDestino());

Ciudad ciudadDestino = new Ciudad();

ciudadDestino.setIdCiudad(destino.getIdCiudad());

ciudadDestino.setCodigoCiudad(destino.getCodigoCiudad().getValue());

ciudadDestino.setNombreCiudad(destino.getNombreCiudad().getValue());

vuelo.setCiudadDestino(ciudadDestino);

boleto.setVuelo(vuelo);

boletos.add(boleto);

}

}

return boletos;

}

public boolean comprarBoletos(CompraBoletoRequest compraBoletoRequest) {

ec.edu.viajecito.client.CompraBoletoRequest compraBoletoRequest1 = new ec.edu.viajecito.client.CompraBoletoRequest();

compraBoletoRequest1.setCantidad(compraBoletoRequest.cantidad);

compraBoletoRequest1.setIdUsuario(compraBoletoRequest.idUsuario);

compraBoletoRequest1.setIdVuelo(compraBoletoRequest.idVuelo);

return AeroCondorClient.comprar(compraBoletoRequest1);

}

}

package ec.edu.viajecito.controller;

import ec.edu.viajecito.client.AeroCondorClient;

import ec.edu.viajecito.client.ArrayOfCiudades;

import ec.edu.viajecito.client.Ciudades;

import ec.edu.viajecito.model.Ciudad;

import java.util.ArrayList;

import java.util.List;

public class CiudadesController {

public List<Ciudad> obtenerTodasCiudades() {

ArrayOfCiudades arrayOfCiudades = AeroCondorClient.getCiudades();

List<Ciudad> ciudades = new ArrayList<>();

if (arrayOfCiudades != null && arrayOfCiudades.getCiudades() != null) {

for (Ciudades c : arrayOfCiudades.getCiudades()) {

Ciudad ciudad = new Ciudad();

ciudad.setIdCiudad(c.getIdCiudad());

if (c.getCodigoCiudad() != null) {

ciudad.setCodigoCiudad(c.getCodigoCiudad().getValue());

}

if (c.getNombreCiudad() != null) {

ciudad.setNombreCiudad(c.getNombreCiudad().getValue());

}

ciudades.add(ciudad);

}

}

return ciudades;

}

}

Con ello se procede a realizar la implantación de la vista con los formularios.

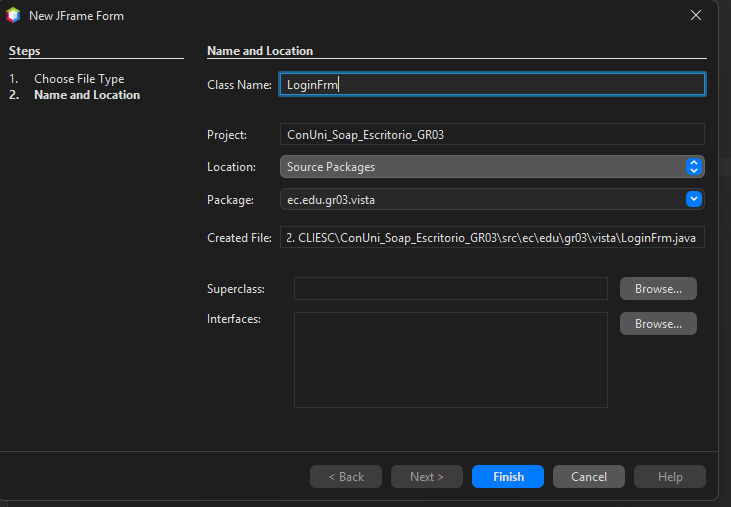


Figura 26. Crear formularios en aplicación de escritorio

Se diseña las vistas de inicio de sesión y navegación, como se indica en el formulario de Login a continuación.

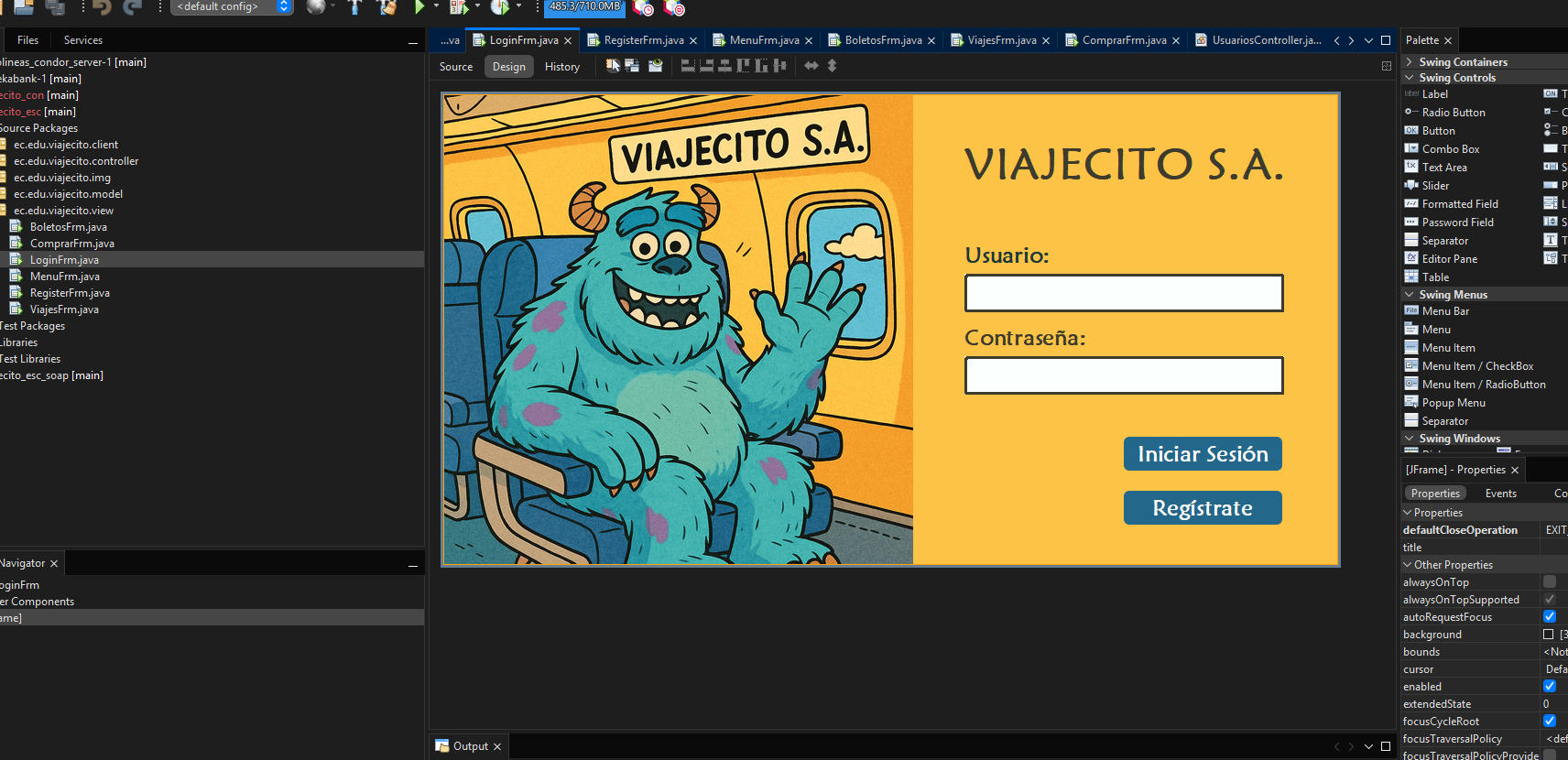


Figura 27. Crear formularios de inicio de sesión en aplicación de escritorio

Finalmente se implementa la llamada a las funciones y la navegación entre ventanas como se muestra en el código a continuación en los eventos generados por los botones.

Tabla 9. **Código de eventos para la aplicación de escritorio.**

|  |
| --- |
| private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {  String username = txtUsername.getText();  String password = new String(txtPassword.getPassword()); // Si usas JPasswordField  if (username.isEmpty() || password.isEmpty()) {  return;  }  UsuariosController controller = new UsuariosController();  Usuario usuario = controller.login(username, password);  if (usuario != null) {  new MenuFrm(usuario).setVisible(true);  this.dispose();  } else {  JOptionPane.showMessageDialog(this, "Credenciales inválidas.", "Error de autenticación", JOptionPane.ERROR\_MESSAGE);  }  }  private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {  new RegisterFrm().setVisible(true);  }  private void btnComprarActionPerformed(java.awt.event.ActionEvent evt) {  if (vuelos.isEmpty()) {  JOptionPane.showMessageDialog(this, "Vuelo no seleccionado.", "Aviso", JOptionPane.WARNING\_MESSAGE);  return;  }    Vuelo vuelo = vuelos.get(cmbVuelos.getSelectedIndex());  int numBoletos = (int) spnNumBoletos.getValue();  double precio = vuelo.getValor().doubleValue() \* numBoletos;    if (numBoletos < 1) {  JOptionPane.showMessageDialog(this, "Indique la cantidad de boletos a comprar.", "Aviso", JOptionPane.WARNING\_MESSAGE);  return;  }    if (numBoletos > vuelo.getDisponibles()) {  JOptionPane.showMessageDialog(this, "La cantidad de boletos eccede los Disponibles.", "Aviso", JOptionPane.WARNING\_MESSAGE);  return;  }  CompraBoletoRequest compraBoletoRequest = new CompraBoletoRequest(vuelo.getIdVuelo(), usuario.getIdUsuario(), numBoletos);    Object[] opciones = {"Confirmar compra", "Cancelar"};  int opcion = JOptionPane.showOptionDialog(  this,  "¿Está seguro que desea comprar el boleto?\n"  + "Vuelo: " + vuelo.getCodigoVuelo() + "\n"  + "Viaje: De " + vuelo.getCiudadOrigen().getNombreCiudad() + " a " + vuelo.getCiudadDestino().getNombreCiudad() + "\n"  + "Fecha: " + vuelo.getHoraSalida() + "\n"  + "Número de Boletos: " + numBoletos + "\n"  + "Precio Total: " + precio + "$\n",  "Confirmación",  JOptionPane.YES\_NO\_OPTION,  JOptionPane.QUESTION\_MESSAGE,  null,  opciones,  opciones[0]  );  if (opcion == JOptionPane.YES\_OPTION) {  // Acción de compra  System.out.println("Compra confirmada.");  boolean wasBuy = new BoletosController().comprarBoletos(compraBoletoRequest);    if (wasBuy) {  JOptionPane.showMessageDialog(this, "Compra realizada con éxito.", "Éxito", JOptionPane.INFORMATION\_MESSAGE);  } else {  JOptionPane.showMessageDialog(this, "Hubo problemas al comprar el boleto intente mas tarde.", "Error", JOptionPane.ERROR\_MESSAGE);  }          }  }  private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {  String origen = ciudades.get(cmbOrigen.getSelectedIndex()).getCodigoCiudad();  String destino = ciudades.get(cmbDestino.getSelectedIndex()).getCodigoCiudad();    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  String horaSalida = sdf.format((Date) spinDate.getValue());    vuelos = new VuelosController().obtenerVuelosPorCiudad(origen, destino, horaSalida);  cmbVuelos.removeAllItems();  if (vuelos.isEmpty()) {  JOptionPane.showMessageDialog(this, "Vuelo no Disponible.", "Información", JOptionPane.INFORMATION\_MESSAGE);  } else {  for (Vuelo v : vuelos) {  cmbVuelos.addItem(v.getCodigoVuelo() + " - Precio: " + v.getValor() + "$ - Disponibles: " + v.getDisponibles());  }  }  } |

### EJECUCIÓN DE LA APLICACIÓN

Finalmente se ejecuta la aplicación de escritorio y se prueba las funcionalidades del Web Service, primero se procede a probar el login de la aplicación.



Figura 29. Login en aplicación de escritorio.

A continuación, se procede a probar las funciones ver boletos comprados y comprar boleto.

Tabla

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Figura 30. Funciones en aplicación de escritorio

Con ello se finalizó la aplicación de escritorio consumiendo un cliente REST del servidor y creando las interfaces de usuario de login y menú con las funcionalidades operativas.

## CLIENTE WEB

La solución web de la agencia Viajecitos SA se concibe como una aplicación desarrollada con Expo y React Native Web, de modo que el mismo código se ejecuta indistintamente en navegadores y, si se requiriese, en dispositivos móviles. A lo largo de todo el proceso de interacción con el sistema de Aerolíneas Cóndor se emplea Axios para cursar solicitudes SOAP, asegurándose así una integración transparente entre los dos entornos involucrados.

### CREACION DE LA APLICACIÓN

Para iniciar el proyecto se recurrió al comando npx create-expo-app, el cual generó una plantilla básica en la que posteriormente se añadieron, mediante npm, las dependencias expo-router y axios. Gracias a estas incorporaciones, la navegación interna quedó resuelta y la capa de comunicación SOAP se integró sin fricciones. Finalmente, la instrucción expo prebuild:web habilitó el objetivo web, lo que permitió verificar la interfaz directamente en un navegador antes de proceder a cualquier despliegue adicional.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Figura 31. Crear proyecto base de la aplicación web

Luego se procede con la instalación de la biblioteca axios con el comando “npm install axios”

### MODELO MVC

Para aplicar el modelo MVC se procede a crear las carpetas model, view y controller dentro de la carpeta src del proyecto creado.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Figura 32. Crear modelo MVC en el proyecto de la aplicación web

### IMPLEMENTACIÓN

En primera instancia se definieron los modelos de dominio. De manera análoga al ejemplo del usuario original, se creó un objeto Vuelo, encargado de representar los parámetros devueltos por la aerolínea (ciudad de origen, ciudad de destino, valor y hora de salida), y un objeto Boleto, responsable de reflejar cada compra realizada. A continuación, se configuró el controlador encargado de consultar el Web Service BuscarVuelos y de filtrar, entre los registros obtenidos, el vuelo de mayor valor. Posteriormente se elaboró el controlador destinado a la operación ComprarBoleto; este, tras recibir la confirmación positiva del servidor, notifica a la vista y registra la transacción en la base de datos de la aerolínea..

Imagen que contiene Forma

El contenido generado por IA puede ser incorrecto.

Figura 33. Crear controladores en el proyecto de la aplicación web

Tabla 10. **Código BoletoController.js.**

|  |
| --- |
| import { XMLParser } from 'fast-xml-parser';  const parser = new XMLParser({ ignoreAttributes: false });  const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';  // Obtener boletos de un usuario específico  export const obtenerBoletosPorUsuario = async (idUsuario) => {  const body = `  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">  <s:Body>  <ObtenerBoletosPorUsuario xmlns="http://tempuri.org/">  <idUsuario>${idUsuario}</idUsuario>  </ObtenerBoletosPorUsuario>  </s:Body>  </s:Envelope>`;  const response = await fetch(endpoint, {  method: 'POST',  headers: {  'Content-Type': 'text/xml;charset=UTF-8',  SOAPAction: 'http://tempuri.org/IAeroCondorController/ObtenerBoletosPorUsuario'  },  body  });  const xml = await response.text();  const json = parser.parse(xml);  try {  const result =  json['s:Envelope']['s:Body']['ObtenerBoletosPorUsuarioResponse']['ObtenerBoletosPorUsuarioResult'];  if (!result) return [];  let boletos = result['a:Boletos'];  // Si es un solo boleto, lo convertimos a array  if (!Array.isArray(boletos)) {  boletos = [boletos];  }  // Mapear para dejar solo los campos necesarios  return boletos.map((b) => ({  idBoleto: b['a:IdBoleto'],  numeroBoleto: b['a:NumeroBoleto'],  fechaCompra: b['a:FechaCompra'],  precio: b['a:PrecioCompra'],  idVuelo: b['a:IdVuelo'],  idUsuario: b['a:IdUsuario']  }));  } catch (error) {  console.error('❌ Error al procesar respuesta SOAP:', error);  return [];  }  };  // Registrar compra de boletos  export const registrarBoleto = async ({ idVuelo, idUsuario, cantidad = 1 }) => {  const body = `  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">  <s:Body>  <Comprar xmlns="http://tempuri.org/">  <request xmlns:a="http://schemas.datacontract.org/2004/07/ec.edu.monster.modelo">  <a:IdVuelo>${idVuelo}</a:IdVuelo>  <a:IdUsuario>${idUsuario}</a:IdUsuario>  <a:Cantidad>${cantidad}</a:Cantidad>  </request>  </Comprar>  </s:Body>  </s:Envelope>`;  const response = await fetch(endpoint, {  method: 'POST',  headers: {  'Content-Type': 'text/xml;charset=UTF-8',  'SOAPAction': 'http://tempuri.org/IAeroCondorController/Comprar'  },  body  });  const xml = await response.text();  const json = parser.parse(xml);  try {  const result = json['s:Envelope']['s:Body']['ComprarResponse']['ComprarResult'];  return typeof result === 'string' ? result.toLowerCase() === 'true' : !!result;  } catch (error) {  console.error('❌ Error al procesar compra:', error);  return false;  }  }; |

Tabla 11. **Código CiudadController.js**

**import { XMLParser } from 'fast-xml-parser';**

**const parser = new XMLParser({ ignoreAttributes: false });**

**const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';**

**export const obtenerCiudades = async () => {**

**const body = `**

**<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">**

**<s:Body>**

**<GetCiudades xmlns="http://tempuri.org/" />**

**</s:Body>**

**</s:Envelope>`;**

**const response = await fetch(endpoint, {**

**method: 'POST',**

**headers: {**

**'Content-Type': 'text/xml;charset=UTF-8',**

**SOAPAction: 'http://tempuri.org/IAeroCondorController/GetCiudades'**

**},**

**body**

**});**

**const xml = await response.text();**

**const json = parser.parse(xml);**

**const result = json['s:Envelope']?.['s:Body']?.['GetCiudadesResponse']?.['GetCiudadesResult'];**

**if (!result || !result['a:Ciudades']) return [];**

**const lista = Array.isArray(result['a:Ciudades']) ? result['a:Ciudades'] : [result['a:Ciudades']];**

**// Mapear los datos útiles**

**return lista.map(c => ({**

**id: c['a:IdCiudad'],**

**codigo: c['a:CodigoCiudad'],**

**nombre: c['a:NombreCiudad']**

**}));**

**};**

**export const obtenerCiudadPorId = async (idCiudad) => {**

**const body = `**

**<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">**

**<s:Body>**

**<ObtenerCiudadPorId xmlns="http://tempuri.org/">**

**<id>${idCiudad}</id>**

**</ObtenerCiudadPorId>**

**</s:Body>**

**</s:Envelope>`;**

**const response = await fetch(endpoint, {**

**method: 'POST',**

**headers: {**

**'Content-Type': 'text/xml;charset=UTF-8',**

**SOAPAction: 'http://tempuri.org/IAeroCondorController/ObtenerCiudadPorId'**

**},**

**body**

**});**

**const xml = await response.text();**

**const json = parser.parse(xml);**

**return json['s:Envelope']['s:Body']['ObtenerCiudadPorIdResponse']['ObtenerCiudadPorIdResult'] || null;**

**};**

Tabla 13. **Código de VueloController.js**

|  |
| --- |
| import { XMLParser } from 'fast-xml-parser';  const parser = new XMLParser({ ignoreAttributes: false });  const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';  export const obtenerVuelos = async () => {  const body = `  <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">  <s:Body>  <GetVuelos xmlns="http://tempuri.org/" />  </s:Body>  </s:Envelope>`;  const response = await fetch(endpoint, {  method: 'POST',  headers: {  'Content-Type': 'text/xml;charset=UTF-8',  SOAPAction: 'http://tempuri.org/IAeroCondorController/GetVuelos'  },  body  });  const xml = await response.text();  const json = parser.parse(xml);  const raw = json?.['s:Envelope']?.['s:Body']?.['GetVuelosResponse']?.['GetVuelosResult'];  const vuelos = raw?.['a:Vuelos'];  const lista = Array.isArray(vuelos) ? vuelos : vuelos ? [vuelos] : [];  // Mapeo limpio de campos  return lista.map(v => ({  IdVuelo: v['a:IdVuelo'],  CodigoVuelo: v['a:CodigoVuelo'],  HoraSalida: v['a:HoraSalida'],  Valor: v['a:Valor'],  Capacidad: v['a:Capacidad'],  Disponibles: v['a:Disponibles']  }));  };  export const buscarVuelos = async (origen, destino, fechaSalida) => {  const body = `  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">  <soapenv:Header/>  <soapenv:Body>  <tem:BuscarVuelos>  <tem:origen>${origen}</tem:origen>  <tem:destino>${destino}</tem:destino>  <tem:fechaSalida>${fechaSalida}</tem:fechaSalida>  </tem:BuscarVuelos>  </soapenv:Body>  </soapenv:Envelope>`;  const response = await fetch(endpoint, {  method: 'POST',  headers: {  'Content-Type': 'text/xml;charset=UTF-8',  SOAPAction: 'http://tempuri.org/IAeroCondorController/BuscarVuelos'  },  body  });  const xml = await response.text();  const json = parser.parse(xml);  const vuelosRaw = json?.['s:Envelope']?.['s:Body']?.['BuscarVuelosResponse']?.['BuscarVuelosResult']?.['a:Vuelos'];  const lista = Array.isArray(vuelosRaw) ? vuelosRaw : vuelosRaw ? [vuelosRaw] : [];  return lista.map(v => ({  IdVuelo: v['a:IdVuelo'],  CodigoVuelo: v['a:CodigoVuelo'],  HoraSalida: v['a:HoraSalida'],  Valor: v['a:Valor'],  Capacidad: v['a:Capacidad'],  Disponibles: v['a:Disponibles']  }));  }; |

Tabla 14. **Código de la vista LoginView.jsx**

|  |
| --- |
| *import React, { useState } from 'react';*  *import {*  *View,*  *TextInput,*  *Button,*  *Text,*  *StyleSheet,*  *Image,*  *TouchableOpacity,*  *Platform,*  *KeyboardAvoidingView,*  *Modal*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import { login } from '../app/controllers/UsuarioController';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function LoginView() {*  *const router = useRouter();*  *const [username, setUsername] = useState('');*  *const [password, setPassword] = useState('');*  *const [secure, setSecure] = useState(true);*  *const [loading, setLoading] = useState(false);*  *const [irARegistro, setIrARegistro] = useState(false);*  *const [modalVisible, setModalVisible] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [esExito, setEsExito] = useState(false);*  *const [usuarioAutenticado, setUsuarioAutenticado] = useState(null);*  *if (irARegistro) {*  *const RegisterView = require('./RegisterView').default;*  *return <RegisterView volver={() => setIrARegistro(false)} />;*  *}*  *const handleLogin = async () => {*  *if (!username.trim() || !password.trim()) {*  *setMensaje('⚠️ Todos los campos son obligatorios');*  *setEsExito(false);*  *setModalVisible(true);*  *return;*  *}*  *setLoading(true);*  *try {*  *const user = await login(username.trim(), password.trim());*  *if (user && user.idUsuario) {*  *await AsyncStorage.clear();*  *await AsyncStorage.setItem('idUsuario', user.idUsuario.toString());*  *setMensaje(`✅ Bienvenido ${user.nombre}`);*  *setUsuarioAutenticado(user);*  *setEsExito(true);*  *} else {*  *setMensaje('❌ Usuario o contraseña incorrectos');*  *setEsExito(false);*  *}*  *} catch (error) {*  *console.error('❌ Error de login:', error);*  *setMensaje('❌ Error al conectar con el servidor');*  *setEsExito(false);*  *} finally {*  *setLoading(false);*  *setModalVisible(true);*  *}*  *};*  *return (*  *<KeyboardAvoidingView*  *style={styles.container}*  *behavior={Platform.OS === 'ios' ? 'padding' : undefined}*  *>*  *<Image source={require('../assets/images/logo\_monster.png')} style={styles.logo} />*  *<Text style={styles.title}>MONSTER - VIAJESITOS</Text>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Usuario"*  *value={username}*  *onChangeText={setUsername}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Contraseña"*  *secureTextEntry={secure}*  *value={password}*  *onChangeText={setPassword}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *<TouchableOpacity onPress={() => setSecure(!secure)} style={styles.toggle}>*  *<Text style={styles.toggleText}>{secure ? 'Mostrar' : 'Ocultar'}</Text>*  *</TouchableOpacity>*  *</View>*  *<View style={styles.buttonContainer}>*  *<Button*  *title={loading ? 'Ingresando...' : 'INGRESAR'}*  *color="#5fb4a2"*  *onPress={handleLogin}*  *disabled={loading}*  */>*  *</View>*  *<TouchableOpacity onPress={() => setIrARegistro(true)} style={{ marginTop: 16 }}>*  *<Text style={{ color: '#4e88a9', fontWeight: '500' }}>*  *¿No tienes cuenta? Regístrate*  *</Text>*  *</TouchableOpacity>*  *<Modal*  *visible={modalVisible}*  *transparent*  *animationType="fade"*  *onRequestClose={() => setModalVisible(false)}*  *>*  *<View style={styles.modalOverlay}>*  *<View style={styles.modalContainer}>*  *<Text style={[styles.modalMessage, { color: esExito ? 'green' : 'red' }]}>*  *{mensaje}*  *</Text>*  *<TouchableOpacity*  *onPress={() => {*  *setModalVisible(false);*  *if (esExito && usuarioAutenticado) {*  *setUsername('');*  *setPassword('');*  *router.replace({*  *pathname: '/views/MenuView',*  *params: {*  *idUsuario: usuarioAutenticado.idUsuario,*  *nombre: usuarioAutenticado.nombre*  *}*  *});*  *}*  *}}*  *style={styles.modalButton}*  *>*  *<Text style={{ color: '#fff', fontWeight: 'bold' }}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 15. **Código RegisterView.jsx**

|  |
| --- |
| *import React, { useState } from 'react';*  *import {*  *View,*  *TextInput,*  *Button,*  *Text,*  *StyleSheet,*  *Platform,*  *KeyboardAvoidingView,*  *Image,*  *TouchableOpacity,*  *Modal*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import { crearUsuario } from './controllers/UsuarioController';*  *export default function RegisterView() {*  *const router = useRouter();*  *const [nombre, setNombre] = useState('');*  *const [username, setUsername] = useState('');*  *const [password, setPassword] = useState('');*  *const [telefono, setTelefono] = useState('');*  *const [secure, setSecure] = useState(true);*  *const [loading, setLoading] = useState(false);*  *const [modalVisible, setModalVisible] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [esExito, setEsExito] = useState(false);*  *const handleRegister = async () => {*  *if (!nombre.trim() || !username.trim() || !password.trim() || !telefono.trim()) {*  *setMensaje('⚠️ Todos los campos son obligatorios');*  *setEsExito(false);*  *setModalVisible(true);*  *return;*  *}*  *setLoading(true);*  *try {*  *const success = await crearUsuario({*  *idUsuario: 0,*  *nombre: nombre.trim(),*  *username: username.trim(),*  *password: password.trim(),*  *telefono: telefono.trim()*  *});*  *if (success === true) {*  *setMensaje('✅ Registro exitoso. Ya puedes iniciar sesión.');*  *setEsExito(true);*  *} else {*  *setMensaje('❌ El usuario ya está registrado o ocurrió un error.');*  *setEsExito(false);*  *}*  *} catch (error) {*  *setMensaje('❌ Error de conexión o del servidor.');*  *setEsExito(false);*  *} finally {*  *setLoading(false);*  *setModalVisible(true);*  *}*  *};*  *const volverAlLogin = () => {*  *setModalVisible(false);*  *if (esExito) router.replace('/LoginView');*  *};*  *return (*  *<KeyboardAvoidingView*  *style={styles.container}*  *behavior={Platform.OS === 'ios' ? 'padding' : undefined}*  *>*  *<Image source={require('../assets/images/logo\_monster.png')} style={styles.logo} />*  *<Text style={styles.title}>Crear cuenta</Text>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Nombre completo"*  *value={nombre}*  *onChangeText={setNombre}*  *placeholderTextColor="#888"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Usuario"*  *value={username}*  *onChangeText={setUsername}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Contraseña"*  *secureTextEntry={secure}*  *value={password}*  *onChangeText={setPassword}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *<TouchableOpacity onPress={() => setSecure(!secure)} style={styles.toggle}>*  *<Text style={styles.toggleText}>{secure ? 'Mostrar' : 'Ocultar'}</Text>*  *</TouchableOpacity>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Teléfono"*  *keyboardType="phone-pad"*  *value={telefono}*  *onChangeText={setTelefono}*  *placeholderTextColor="#888"*  */>*  *</View>*  *<View style={styles.buttonContainer}>*  *<Button*  *title={loading ? 'Registrando...' : 'REGISTRAR'}*  *color="#5fb4a2"*  *onPress={handleRegister}*  *disabled={loading}*  */>*  *</View>*  *<TouchableOpacity onPress={() => router.replace('/LoginView')} style={{ marginTop: 16 }}>*  *<Text style={{ color: '#4e88a9', fontWeight: '500' }}>*  *¿Ya tienes cuenta? Inicia sesión*  *</Text>*  *</TouchableOpacity>*  *<Modal*  *visible={modalVisible}*  *transparent*  *animationType="fade"*  *onRequestClose={() => setModalVisible(false)}*  *>*  *<View style={styles.modalOverlay}>*  *<View style={styles.modalContainer}>*  *<Text style={[styles.modalMessage, { color: esExito ? 'green' : 'red' }]}>*  *{mensaje}*  *</Text>*  *<TouchableOpacity*  *onPress={volverAlLogin}*  *style={styles.modalButton}*  *>*  *<Text style={{ color: '#fff', fontWeight: 'bold' }}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 16. **Código ComprarBoletoView.jsx**

|  |
| --- |
| *import React, { useEffect, useState } from 'react';*  *import {*  *View,*  *Text,*  *FlatList,*  *TouchableOpacity,*  *Modal,*  *ActivityIndicator,*  *TextInput,*  *Alert,*  *KeyboardAvoidingView,*  *StyleSheet,*  *ScrollView,*  *useWindowDimensions*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *import { obtenerCiudades } from '../controllers/CiudadController';*  *import { buscarVuelos as buscarVuelosAPI } from '../controllers/VueloController';*  *import { registrarBoleto } from '../controllers/BoletoController';*  *export default function ComprarBoletoView() {*  *const router = useRouter();*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const [usuario, setUsuario] = useState(null);*  *const [ciudades, setCiudades] = useState([]);*  *const [origen, setOrigen] = useState('');*  *const [destino, setDestino] = useState('');*  *const [fecha, setFecha] = useState('');*  *const [vuelos, setVuelos] = useState([]);*  *const [loading, setLoading] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [modalVisible, setModalVisible] = useState(false);*  *const [cantidad, setCantidad] = useState(1);*  *useEffect(() => {*  *const cargar = async () => {*  *try {*  *const data = await obtenerCiudades();*  *setCiudades(Array.isArray(data) ? data : [data]);*  *} catch (e) {*  *console.error('Error cargando ciudades:', e);*  *setCiudades([]);*  *}*  *const storedId = await AsyncStorage.getItem('idUsuario');*  *if (storedId) setUsuario(parseInt(storedId));*  *else router.replace('/');*  *};*  *cargar();*  *}, []);*  *const getNombreCiudad = (codigo) => {*  *const ciudad = ciudades.find(c => c.codigoCiudad === codigo);*  *return ciudad ? ciudad.nombreCiudad : codigo;*  *};*  *const limpiarFormulario = () => {*  *setOrigen('');*  *setDestino('');*  *setFecha('');*  *setVuelos([]);*  *setCantidad(1);*  *};*  *const transformVuelo = (vuelo) => ({*  *...vuelo,*  *IdVuelo: vuelo.idVuelo,*  *CodigoVuelo: vuelo.codigoVuelo,*  *Valor: vuelo.valor,*  *Disponibles: vuelo.disponibles,*  *HoraSalida: vuelo.horaSalida,*  *Origen: vuelo.idCiudadOrigen?.codigoCiudad || 'ND',*  *Destino: vuelo.idCiudadDestino?.codigoCiudad || 'ND',*  *});*  *const handleBuscarVuelos = async () => {*  *if (!origen || !destino || origen === destino || !/^\d{4}-\d{2}-\d{2}$/.test(fecha)) {*  *Alert.alert('Error', 'Seleccione ciudades válidas y una fecha con formato correcto (YYYY-MM-DD).');*  *return;*  *}*  *setLoading(true);*  *try {*  *const resultados = await buscarVuelosAPI(origen, destino, fecha);*  *const lista = Array.isArray(resultados)*  *? resultados.map(transformVuelo)*  *: resultados*  *? [transformVuelo(resultados)]*  *: [];*  *setVuelos(lista);*  *if (lista.length === 0) Alert.alert('Sin vuelos disponibles');*  *} catch (error) {*  *console.error('Error al buscar vuelos:', error);*  *Alert.alert('Error', 'No se pudo buscar vuelos.');*  *} finally {*  *setLoading(false);*  *}*  *};*  *const handleComprar = async (vuelo) => {*  *if (!usuario || isNaN(usuario)) {*  *setMensaje('❌ ID de usuario no disponible.');*  *setModalVisible(true);*  *return;*  *}*  *if (!cantidad || isNaN(cantidad) || cantidad <= 0 || cantidad > vuelo.Disponibles) {*  *setMensaje(`❌ Cantidad inválida. Disponible: ${vuelo.Disponibles}`);*  *setModalVisible(true);*  *return;*  *}*  *const total = (parseFloat(vuelo.Valor) \* cantidad).toFixed(2);*  *try {*  *const resultado = await registrarBoleto({*  *idVuelo: vuelo.IdVuelo,*  *idUsuario: usuario,*  *cantidad*  *});*  *if (resultado) {*  *setMensaje(*  *`✅ Compra realizada\n\n✈ Vuelo: ${vuelo.CodigoVuelo}\n🛫 Origen: ${getNombreCiudad(vuelo.Origen)}\n🛬 Destino: ${getNombreCiudad(vuelo.Destino)}\n🎟 Cantidad: ${cantidad}\n💵 Total: $${total}`*  *);*  *setModalVisible(true);*  *limpiarFormulario();*  *setTimeout(() => {*  *setModalVisible(false);*  *router.replace({ pathname: '/views/MenuView', params: { idUsuario: usuario } });*  *}, 3000);*  *} else {*  *setMensaje('❌ No se pudo completar la compra');*  *setModalVisible(true);*  *}*  *} catch (e) {*  *console.error('Error al comprar:', e);*  *setMensaje('❌ Error inesperado.');*  *setModalVisible(true);*  *}*  *};*  *const renderCiudadItem = (item, onSelect, selected) => (*  *<TouchableOpacity*  *key={item.codigoCiudad}*  *style={[styles.ciudadBtn, selected === item.codigoCiudad && styles.ciudadBtnSelected]}*  *onPress={() => onSelect(item.codigoCiudad)}*  *>*  *<Text style={styles.ciudadText}>{item.codigoCiudad} - {item.nombreCiudad}</Text>*  *</TouchableOpacity>*  *);*  *const renderVueloItem = ({ item }) => (*  *<View style={isMobile ? styles.card : styles.cardDesktop}>*  *<Text style={styles.title}>✈️ {item.CodigoVuelo}</Text>*  *<Text>🛫 Origen: {getNombreCiudad(item.Origen)}</Text>*  *<Text>🛬 Destino: {getNombreCiudad(item.Destino)}</Text>*  *<Text>Salida: {item.HoraSalida}</Text>*  *<Text>Precio: ${item.Valor}</Text>*  *<Text>Disponibles: {item.Disponibles}</Text>*  *<View style={styles.stepperContainer}>*  *<TouchableOpacity*  *onPress={() => setCantidad(Math.max(1, cantidad - 1))}*  *style={styles.stepperBtn}*  *>*  *<Text style={styles.stepperText}>−</Text>*  *</TouchableOpacity>*  *<Text style={styles.stepperCount}>{cantidad}</Text>*  *<TouchableOpacity*  *onPress={() => setCantidad(Math.min(10, cantidad + 1))}*  *style={styles.stepperBtn}*  *>*  *<Text style={styles.stepperText}>+</Text>*  *</TouchableOpacity>*  *</View>*  *<TouchableOpacity style={styles.btn} onPress={() => handleComprar(item)}>*  *<Text style={styles.btnText}>Comprar</Text>*  *</TouchableOpacity>*  *</View>*  *);*  *return (*  *<KeyboardAvoidingView behavior="padding" style={styles.scroll}>*  *<ScrollView>*  *<View style={styles.container}>*  *<Text style={styles.header}>Buscar Vuelos</Text>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Ciudad de Origen</Text>*  *<ScrollView horizontal contentContainerStyle={styles.selectorContainer}>*  *{ciudades.filter(c => c.codigoCiudad !== destino).map(c => renderCiudadItem(c, setOrigen, origen))}*  *</ScrollView>*  *</View>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Ciudad de Destino</Text>*  *<ScrollView horizontal contentContainerStyle={styles.selectorContainer}>*  *{ciudades.filter(c => c.codigoCiudad !== origen).map(c => renderCiudadItem(c, setDestino, destino))}*  *</ScrollView>*  *</View>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Fecha de Vuelo</Text>*  *<TextInput*  *style={styles.input}*  *placeholder="YYYY-MM-DD"*  *value={fecha}*  *onChangeText={setFecha}*  */>*  *</View>*  *<TouchableOpacity style={styles.btnBuscar} onPress={handleBuscarVuelos}>*  *<Text style={styles.btnText}>Buscar Vuelos</Text>*  *</TouchableOpacity>*  *{loading ? (*  *<ActivityIndicator size="large" color="#35798e" />*  *) : (*  *<FlatList*  *data={vuelos}*  *keyExtractor={(item) => `vuelo-${item.IdVuelo}`}*  *renderItem={renderVueloItem}*  *contentContainerStyle={{ paddingBottom: 80 }}*  *ListEmptyComponent={<Text style={styles.noVuelos}>No hay vuelos disponibles.</Text>}*  */>*  *)}*  *</View>*  *<Modal visible={modalVisible} transparent animationType="slide">*  *<View style={styles.modalContainer}>*  *<View style={styles.modalContent}>*  *<Text style={styles.modalText}>{mensaje}</Text>*  *<TouchableOpacity*  *style={styles.modalButton}*  *onPress={() => {*  *setModalVisible(false);*  *if (mensaje.startsWith('✅')) {*  *router.replace({ pathname: '/views/MenuView', params: { idUsuario: usuario } });*  *}*  *}}*  *>*  *<Text style={styles.modalButtonText}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</ScrollView>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 17. **Código MisBoletosView.jsx**

|  |
| --- |
| *import React, { useState, useCallback } from 'react';*  *import {*  *View,*  *Text,*  *FlatList,*  *ScrollView,*  *StyleSheet,*  *ActivityIndicator,*  *TouchableOpacity,*  *useWindowDimensions,*  *SafeAreaView,*  *StatusBar,*  *} from 'react-native';*  *import { useRouter, useLocalSearchParams, useFocusEffect } from 'expo-router';*  *import { obtenerBoletosPorUsuario } from '../controllers/BoletoController';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function MisBoletosView() {*  *const router = useRouter();*  *const { idUsuario: idParam } = useLocalSearchParams();*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const [boletos, setBoletos] = useState([]);*  *const [loading, setLoading] = useState(true);*  *useFocusEffect(*  *useCallback(() => {*  *const cargarBoletos = async () => {*  *setLoading(true);*  *setBoletos([]);*  *try {*  *let idUsuarioActual = idParam;*  *if (!idUsuarioActual) {*  *idUsuarioActual = await AsyncStorage.getItem('idUsuario');*  *if (!idUsuarioActual) throw new Error('Usuario no autenticado');*  *} else {*  *await AsyncStorage.setItem('idUsuario', idUsuarioActual);*  *}*  *const datos = await obtenerBoletosPorUsuario(idUsuarioActual);*  *if (Array.isArray(datos)) {*  *const ordenados = datos.sort((a, b) =>*  *new Date(b.fechaCompra) - new Date(a.fechaCompra)*  *);*  *setBoletos(ordenados);*  *} else {*  *setBoletos([]);*  *}*  *} catch (error) {*  *console.error('❌ Error al cargar boletos:', error);*  *setBoletos([]);*  *} finally {*  *setLoading(false);*  *}*  *};*  *cargarBoletos();*  *}, [idParam])*  *);*  *const parseFechaCompra = (fechaStr) => {*  *if (!fechaStr || typeof fechaStr !== 'string') return null;*  *const cleanStr = fechaStr.replace(/\[.\*?\]/g, '');*  *const date = new Date(cleanStr);*  *return isNaN(date) ? null : date;*  *};*  *const formatDate = (dateString) => {*  *const date = parseFechaCompra(dateString);*  *if (!date) return 'Fecha inválida';*  *return date.toLocaleString('es-EC', {*  *day: '2-digit',*  *month: '2-digit',*  *year: 'numeric',*  *hour: '2-digit',*  *minute: '2-digit',*  *});*  *};*  *const renderItem = ({ item, index }) => {*  *if (!item || !item.numeroBoleto) return null;*  *return (*  *<View style={[styles.card, { opacity: loading ? 0.5 : 1 }]}>*  *<View style={styles.cardHeader}>*  *<Text style={styles.cardTitle}>🎫 Boleto #{item.numeroBoleto}</Text>*  *<View style={styles.badge}>*  *<Text style={styles.badgeText}>#{index + 1}</Text>*  *</View>*  *</View>*  *<View style={styles.cardContent}>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>✈️ Vuelo:</Text>*  *<Text style={styles.value}>{item.idVuelo?.codigoVuelo || 'Desconocido'}</Text>*  *</View>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>📅 Fecha de Compra:</Text>*  *<Text style={styles.value}>{formatDate(item.fechaCompra)}</Text>*  *</View>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>💰 Precio:</Text>*  *<Text style={styles.priceValue}>${item.precioCompra}</Text>*  *</View>*  *</View>*  *</View>*  *);*  *};*  *const handleVolverMenu = async () => {*  *let idUsuarioActual = idParam;*  *if (!idUsuarioActual) {*  *idUsuarioActual = await AsyncStorage.getItem('idUsuario');*  *}*  *if (idUsuarioActual) {*  *router.replace({*  *pathname: '/views/MenuView',*  *params: { idUsuario: idUsuarioActual }*  *});*  *} else {*  *router.replace('/');*  *}*  *};*  *return (*  *<SafeAreaView style={styles.safeArea}>*  *<StatusBar barStyle="dark-content" backgroundColor="#f8f9fa" />*  *<View style={styles.container}>*  *<View style={styles.headerContainer}>*  *<Text style={styles.header}>🎫 Mis Boletos</Text>*  *<Text style={styles.subtitle}>*  *{loading ? 'Cargando...' : `${boletos.length} boleto${boletos.length !== 1 ? 's' : ''} encontrado${boletos.length !== 1 ? 's' : ''}`}*  *</Text>*  *</View>*  *<ScrollView*  *style={styles.mainContent}*  *contentContainerStyle={styles.scrollContent}*  *showsVerticalScrollIndicator={true}*  *>*  *{loading ? (*  *<View style={styles.loadingContainer}>*  *<ActivityIndicator size="large" color="#4e88a9" />*  *<Text style={styles.loadingText}>Cargando tus boletos...</Text>*  *</View>*  *) : boletos.length === 0 ? (*  *<View style={styles.emptyContainer}>*  *<Text style={styles.emptyIcon}>✈️</Text>*  *<Text style={styles.emptyTitle}>No tienes boletos</Text>*  *<Text style={styles.emptyText}>Aún no has comprado ningún boleto.{"\n"}¡Reserva tu próximo vuelo!</Text>*  *</View>*  *) : isMobile ? (*  *<FlatList*  *data={boletos}*  *keyExtractor={(item, index) =>*  *item && item.numeroBoleto ? String(item.numeroBoleto) : `boleto-${index}`*  *}*  *renderItem={renderItem}*  *showsVerticalScrollIndicator={true}*  *contentContainerStyle={styles.listContainer}*  *ItemSeparatorComponent={() => <View style={styles.separator} />}*  */>*  *) : (*  *<View style={styles.tableContainer}>*  *<View style={styles.tableHeader}>*  *<Text style={styles.tableCellHeader}>#</Text>*  *<Text style={styles.tableCellHeader}>Número</Text>*  *<Text style={styles.tableCellHeader}>Vuelo</Text>*  *<Text style={styles.tableCellHeader}>Compra</Text>*  *<Text style={styles.tableCellHeader}>Precio</Text>*  *</View>*  *{boletos.map((item, index) => (*  *<View key={item.numeroBoleto || index} style={styles.tableRow}>*  *<Text style={styles.tableCell}>{index + 1}</Text>*  *<Text style={styles.tableCell}>{item.numeroBoleto}</Text>*  *<Text style={styles.tableCell}>{item.idVuelo?.codigoVuelo || 'ND'}</Text>*  *<Text style={styles.tableCell}>{formatDate(item.fechaCompra)}</Text>*  *<Text style={styles.tableCell}>${item.precioCompra}</Text>*  *</View>*  *))}*  *</View>*  *)}*  *</ScrollView>*  *<View style={styles.buttonContainer}>*  *<TouchableOpacity*  *onPress={handleVolverMenu}*  *style={styles.volverBtn}*  *activeOpacity={0.8}*  *>*  *<Text style={styles.volverText}>← Volver al Menú</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</SafeAreaView>*  *);*  *}* |

Tabla 18. **Código VuelosDisponiblesView.jsx**

|  |
| --- |
| *import React, { useEffect, useState, useCallback } from 'react';*  *import {*  *View,*  *Text,*  *StyleSheet,*  *ActivityIndicator,*  *FlatList,*  *Pressable,*  *SafeAreaView,*  *useWindowDimensions,*  *} from 'react-native';*  *import { obtenerVuelos } from '../controllers/VueloController';*  *import { useRouter, useLocalSearchParams, useFocusEffect } from 'expo-router';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function VuelosDisponiblesView() {*  *const [vuelos, setVuelos] = useState([]);*  *const [loading, setLoading] = useState(true);*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const router = useRouter();*  *const { idUsuario: idParam, nombre: nombreParam } = useLocalSearchParams();*  *const [idUsuario, setIdUsuario] = useState(null);*  *const [nombre, setNombre] = useState(nombreParam || 'Viajero');*  *useFocusEffect(*  *useCallback(() => {*  *const cargar = async () => {*  *let id = idParam;*  *if (!id) {*  *id = await AsyncStorage.getItem('idUsuario');*  *} else {*  *await AsyncStorage.setItem('idUsuario', id);*  *}*  *setIdUsuario(id);*  *setNombre(nombreParam || (await AsyncStorage.getItem('nombre')) || 'Viajero');*  *};*  *cargar();*  *}, [idParam, nombreParam])*  *);*  *useEffect(() => {*  *const cargarVuelos = async () => {*  *try {*  *const data = await obtenerVuelos();*  *const vuelosOrdenados = Array.isArray(data)*  *? [...data].sort((a, b) => new Date(b.horaSalida) - new Date(a.horaSalida))*  *: [];*  *setVuelos(vuelosOrdenados);*  *} catch (error) {*  *console.error('❌ Error al obtener vuelos:', error);*  *setVuelos([]);*  *} finally {*  *setLoading(false);*  *}*  *};*  *cargarVuelos();*  *}, []);*  *const renderItem = ({ item }) => {*  *if (!item || !item.codigoVuelo) return null;*  *return (*  *<View style={styles.card}>*  *<Text style={styles.title}>✈️ {item.codigoVuelo}</Text>*  *<Text>Ruta: {item.idCiudadOrigen?.nombreCiudad} ➡ {item.idCiudadDestino?.nombreCiudad}</Text>*  *<Text>Hora salida: {item.horaSalida}</Text>*  *<Text>Precio: ${item.valor}</Text>*  *<Text>Capacidad: {item.capacidad}</Text>*  *<Text>Disponibles: {item.disponibles}</Text>*  *</View>*  *);*  *};*  *const renderTabla = () => (*  *<View style={styles.tabla}>*  *<View style={styles.filaHeader}>*  *<Text style={styles.colHeader}>Código</Text>*  *<Text style={styles.colHeader}>Ruta</Text>*  *<Text style={styles.colHeader}>Salida</Text>*  *<Text style={styles.colHeader}>Precio</Text>*  *<Text style={styles.colHeader}>Capacidad</Text>*  *<Text style={styles.colHeader}>Disponibles</Text>*  *</View>*  *{vuelos.map((v, i) =>*  *v && v.codigoVuelo ? (*  *<View key={i} style={styles.fila}>*  *<Text style={styles.col}>{v.codigoVuelo}</Text>*  *<Text style={styles.col}>{v.idCiudadOrigen?.nombreCiudad} ➡ {v.idCiudadDestino?.nombreCiudad}</Text>*  *<Text style={styles.col}>{v.horaSalida}</Text>*  *<Text style={styles.col}>${v.valor}</Text>*  *<Text style={styles.col}>{v.capacidad}</Text>*  *<Text style={styles.col}>{v.disponibles}</Text>*  *</View>*  *) : null*  *)}*  *</View>*  *);*  *return (*  *<SafeAreaView style={styles.safeArea}>*  *<View style={styles.container}>*  *<Text style={styles.header}>Vuelos Disponibles</Text>*  *{loading ? (*  *<ActivityIndicator size="large" color="#4e88a9" />*  *) : vuelos.length === 0 ? (*  *<Text style={styles.vacio}>No hay vuelos disponibles.</Text>*  *) : isMobile ? (*  *<FlatList*  *data={vuelos}*  *renderItem={renderItem}*  *keyExtractor={(item, index) =>*  *item && item.idVuelo ? `${item.idVuelo}` : `vuelo-${index}`*  *}*  *contentContainerStyle={styles.flatListContainer}*  *showsVerticalScrollIndicator={false}*  */>*  *) : (*  *renderTabla()*  *)}*  *<Pressable*  *onPress={() =>*  *router.replace({*  *pathname: '/views/MenuView',*  *params: { idUsuario, nombre },*  *})*  *}*  *style={styles.botonVolver}*  *>*  *<Text style={styles.botonTexto}>🔙 Volver al Menú</Text>*  *</Pressable>*  *</View>*  *</SafeAreaView>*  *);*  *}* |

Tabla 18. **Código index.tsx**

|  |
| --- |
| *// app/index.tsx*  *import LoginView from './LoginView';*  *export default LoginView;* |

Tabla 19. **Código \_layout.tsx**

|  |
| --- |
| *import { Drawer } from 'expo-router/drawer';*  *export default function Layout() {*  *return (*  *<Drawer*  *screenOptions={{*  *headerStyle: { backgroundColor: '#eaf6f6' },*  *headerTintColor: '#35798e',*  *drawerStyle: { backgroundColor: '#f0f4f8' },*  *}}*  *>*  *{/\* PANTALLAS OCULTAS DEL DRAWER \*/}*  *<Drawer.Screen*  *name="index"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *<Drawer.Screen*  *name="LoginView"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *<Drawer.Screen*  *name="RegisterView"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *{/\* PANTALLAS VISIBLES EN EL DRAWER \*/}*  *<Drawer.Screen name="views/MenuView" options={{ title: 'Menú Principal' }} />*  *<Drawer.Screen name="views/MisBoletosView" options={{ title: 'Mis Boletos' }} />*  *<Drawer.Screen name="views/VuelosDisponiblesView" options={{ title: 'Vuelos Disponibles' }} />*  *<Drawer.Screen name="views/ComprarBoletoView" options={{ title: 'Comprar Boletos' }} />*  *</Drawer>*  *);*  *}* |

Finalmente se añade estilos css para dar un estilo agradable a la aplicación web.

Tabla20. **Estilo de la aplicación web styles.css**

|  |
| --- |
| *const styles = StyleSheet.create({*  *safeArea: {*  *flex: 1,*  *backgroundColor: '#fff',*  *},*  *container: {*  *flex: 1,*  *padding: 20,*  *alignItems: 'center',*  *},*  *header: {*  *fontSize: 26,*  *fontWeight: 'bold',*  *marginBottom: 20,*  *color: '#35798e',*  *},*  *vacio: {*  *fontSize: 16,*  *color: '#6c757d',*  *textAlign: 'center',*  *marginTop: 20,*  *},*  *card: {*  *backgroundColor: '#e6f7f7',*  *padding: 16,*  *borderRadius: 10,*  *marginBottom: 15,*  *width: '100%',*  *borderColor: '#cfe0e8',*  *borderWidth: 1,*  *},*  *title: {*  *fontSize: 18,*  *fontWeight: '600',*  *marginBottom: 6,*  *color: '#4e88a9',*  *},*  *tabla: {*  *borderWidth: 1,*  *borderColor: '#cfe0e8',*  *width: '100%',*  *maxWidth: 1000,*  *},*  *filaHeader: {*  *flexDirection: 'row',*  *backgroundColor: '#dff6f6',*  *paddingVertical: 10,*  *},*  *fila: {*  *flexDirection: 'row',*  *borderTopWidth: 1,*  *borderColor: '#cfe0e8',*  *paddingVertical: 8,*  *},*  *colHeader: {*  *flex: 1,*  *fontWeight: 'bold',*  *textAlign: 'center',*  *color: '#35798e',*  *},*  *col: {*  *flex: 1,*  *textAlign: 'center',*  *color: '#212529',*  *},*  *botonVolver: {*  *marginTop: 30,*  *backgroundColor: '#4e88a9',*  *paddingHorizontal: 20,*  *paddingVertical: 10,*  *borderRadius: 8,*  *},*  *botonTexto: {*  *color: '#fff',*  *fontSize: 16,*  *fontWeight: '600',*  *},*  *flatListContainer: {*  *paddingBottom: 20,*  *},*  *});* |

### EJECUCIÓN DE LA APLICACIÓN

Finalmente se ejecuta la aplicación web y se prueba las funcionalidades del Web Service, primero se procede a probar el login de la aplicación.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Figura 34. Login en aplicación web.

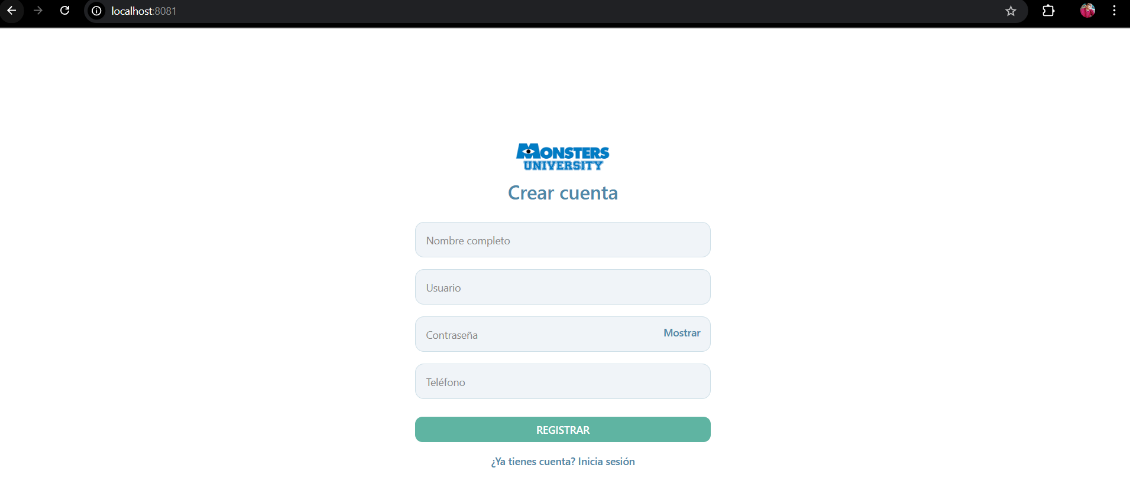


Figura 35. Registro en aplicación web.

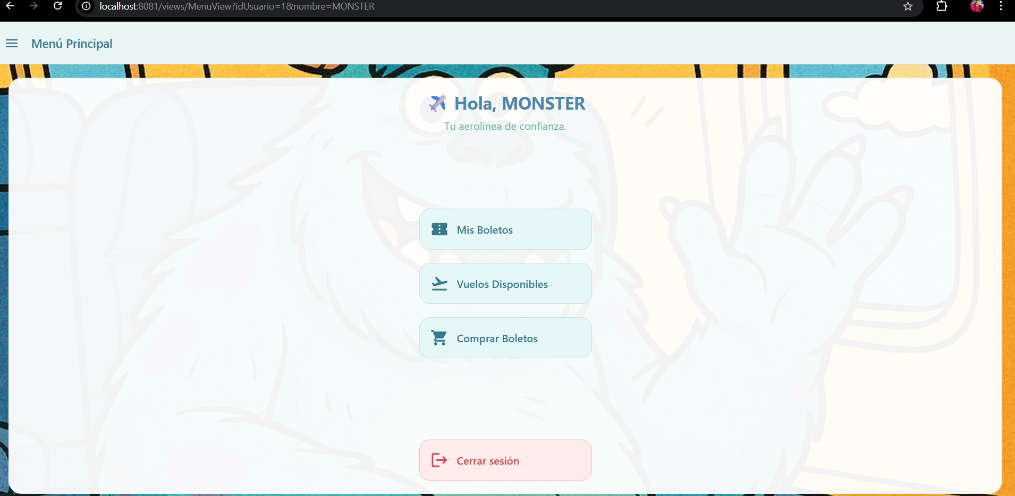


Figura 36. Menú principal en aplicación web.

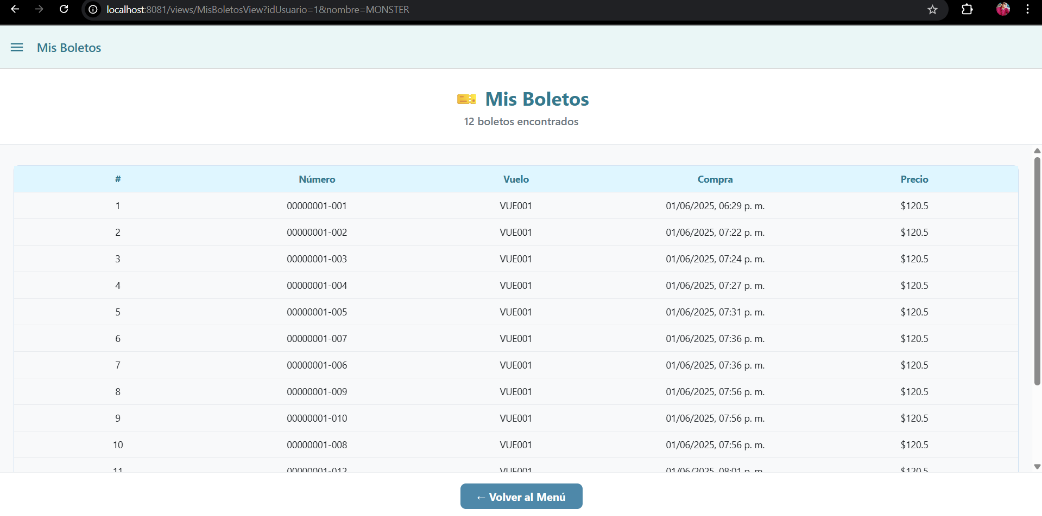


Figura 37. Mis Boletos en aplicación web.

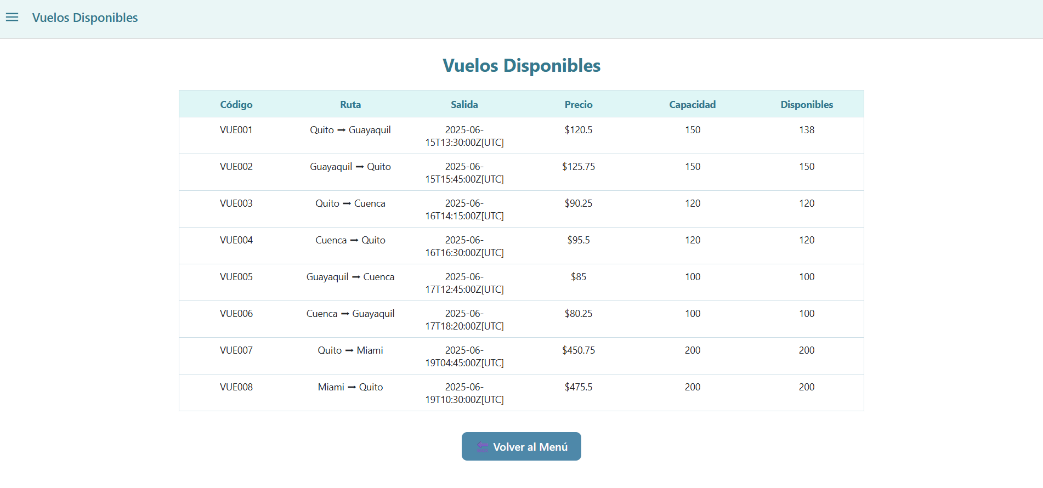


Figura 38. Vuelos disponibles en aplicación web.

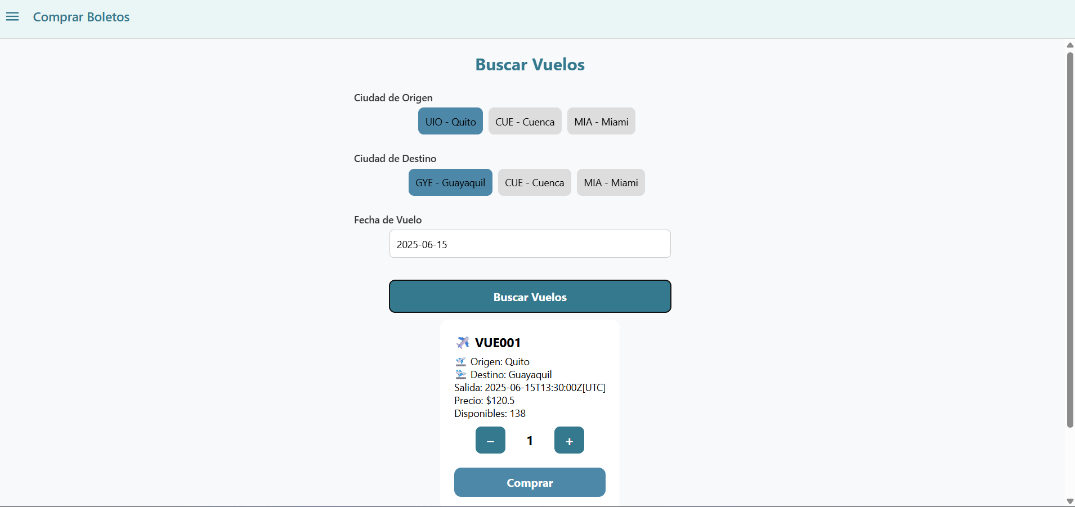


Figura 39. Compra de Boletos en aplicación web.

## CLIENTE MÓVIL

Para complementar la experiencia de compra ofrecida en la web, la agencia Viajecitos SA dispone de un cliente móvil elaborado con Expo y React Native. El empleo de estas tecnologías permite reutilizar la mayor parte de la lógica compartida con la versión de navegador y, al mismo tiempo, ajusta la interfaz a los lineamientos de diseño nativos de Android e iOS. La interacción con los servicios de Aerolíneas Cóndor continúa realizándose mediante solicitudes SOAP canalizadas por Axios, lo que preserva un modelo de integración uniforme entre ambos sistemas.

### CREACION DE LA APLICACIÓN

En primera instancia, el proyecto se generó mediante el comando npx create-expo-app, el cual estableció la configuración esencial de Babel y Metro. Seguidamente, se incorporaron las dependencias axios y expo-router, garantizando con ello la capa de comunicación SOAP y la navegación declarativa entre pantallas. Finalmente, la compatibilidad con dispositivos físicos se corroboró a través de la aplicación Expo Go, desde la cual se verificó el flujo íntegro de búsqueda y compra de boletos.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 40. Creación y configuración del proyecto móvil

Para la realización de pruebas y ejecución del proyecto se usará la aplicación Expo Go la cual permite ejecutar los proyectos en desarrollo en el celular, luego se procede con la instalación de la biblioteca axios y fast-xml-parser con el comando “npm install axios fast-xml-parser”.

.

### MODELO MVC

Para aplicar el modelo MVC se procede a crear las carpetas model, views y controllers dentro de la carpeta app del proyecto creado.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 41. Crear modelo MVC en el proyecto de la aplicación móvil

### IMPLEMENTACIÓN

La lógica de negocio se reutiliza casi en su totalidad, ya que los controladores VueloController y BoletoController consumen los mismos servicios SOAP que emplea la versión web. Sin embargo, las vistas se han rediseñado para ajustarse al contexto móvil y se organizan en tres pantallas funcionales. En primer lugar, BuscarVuelosScreen solicita al usuario la ciudad de origen, la ciudad de destino y la fecha del viaje; a continuación, ResultadoScreen presenta cuando existe el vuelo de mayor valor y, en caso contrario, despliega el mensaje “Vuelo no Disponible”; finalmente, CompraScreen permite confirmar la adquisición y muestra la respuesta devuelta por el servicio. La navegación entre estas tres pantallas se gestiona mediante expo-router, el cual vincula cada archivo de la carpeta app con una ruta específica y, de ese modo, simplifica la transferencia de parámetros a lo largo de todo el flujo.

Tabla 21. **Código BoletoController.js.**

|  |
| --- |
| import { XMLParser } from 'fast-xml-parser';  const parser = new XMLParser({ ignoreAttributes: false });  const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';  // Obtener boletos de un usuario específico  export const obtenerBoletosPorUsuario = async (idUsuario) => {    const body = `      <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">        <s:Body>          <ObtenerBoletosPorUsuario xmlns="http://tempuri.org/">            <idUsuario>${idUsuario}</idUsuario>          </ObtenerBoletosPorUsuario>        </s:Body>      </s:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        SOAPAction: 'http://tempuri.org/IAeroCondorController/ObtenerBoletosPorUsuario'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    try {      const result =        json['s:Envelope']['s:Body']['ObtenerBoletosPorUsuarioResponse']['ObtenerBoletosPorUsuarioResult'];      if (!result) return [];      let boletos = result['a:Boletos'];      // Si es un solo boleto, lo convertimos a array      if (!Array.isArray(boletos)) {        boletos = [boletos];      }      // Mapear para dejar solo los campos necesarios      return boletos.map((b) => ({        idBoleto: b['a:IdBoleto'],        numeroBoleto: b['a:NumeroBoleto'],        fechaCompra: b['a:FechaCompra'],        precio: b['a:PrecioCompra'],        idVuelo: b['a:IdVuelo'],        idUsuario: b['a:IdUsuario']      }));    } catch (error) {      console.error('❌ Error al procesar respuesta SOAP:', error);      return [];    }  };  // Registrar compra de boletos  export const registrarBoleto = async ({ idVuelo, idUsuario, cantidad = 1 }) => {    const body = `      <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">        <s:Body>          <Comprar xmlns="http://tempuri.org/">            <request xmlns:a="http://schemas.datacontract.org/2004/07/ec.edu.monster.modelo">              <a:IdVuelo>${idVuelo}</a:IdVuelo>              <a:IdUsuario>${idUsuario}</a:IdUsuario>              <a:Cantidad>${cantidad}</a:Cantidad>            </request>          </Comprar>        </s:Body>      </s:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        'SOAPAction': 'http://tempuri.org/IAeroCondorController/Comprar'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    try {      const result = json['s:Envelope']['s:Body']['ComprarResponse']['ComprarResult'];      return typeof result === 'string' ? result.toLowerCase() === 'true' : !!result;    } catch (error) {      console.error('❌ Error al procesar compra:', error);      return false;    }  }; |

Tabla 22. **Código CiudadController.js**

|  |
| --- |
| import { XMLParser } from 'fast-xml-parser';  const parser = new XMLParser({ ignoreAttributes: false });  const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';  export const obtenerCiudades = async () => {    const body = `      <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">        <s:Body>          <GetCiudades xmlns="http://tempuri.org/" />        </s:Body>      </s:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        SOAPAction: 'http://tempuri.org/IAeroCondorController/GetCiudades'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    const result = json['s:Envelope']?.['s:Body']?.['GetCiudadesResponse']?.['GetCiudadesResult'];    if (!result || !result['a:Ciudades']) return [];    const lista = Array.isArray(result['a:Ciudades']) ? result['a:Ciudades'] : [result['a:Ciudades']];    // Mapear los datos útiles    return lista.map(c => ({      id: c['a:IdCiudad'],      codigo: c['a:CodigoCiudad'],      nombre: c['a:NombreCiudad']    }));  };  export const obtenerCiudadPorId = async (idCiudad) => {    const body = `      <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">        <s:Body>          <ObtenerCiudadPorId xmlns="http://tempuri.org/">            <id>${idCiudad}</id>          </ObtenerCiudadPorId>        </s:Body>      </s:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        SOAPAction: 'http://tempuri.org/IAeroCondorController/ObtenerCiudadPorId'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    return json['s:Envelope']['s:Body']['ObtenerCiudadPorIdResponse']['ObtenerCiudadPorIdResult'] || null;  }; |

Tabla 12. **Código UsuarioController.js**

|  |
| --- |
| *import { XMLParser } from 'fast-xml-parser';*  *const parser = new XMLParser({ ignoreAttributes: false });*  *const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';*  *// LOGIN*  *export const login = async (username, password) => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:Login>*  *<tem:username>${username}</tem:username>*  *<tem:password>${password}</tem:password>*  *</tem:Login>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/Login'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const result = json['s:Envelope']?.['s:Body']?.['LoginResponse']?.['LoginResult'];*  *if (result && result['a:IdUsuario']) {*  *return {*  *IdUsuario: result['a:IdUsuario'],*  *Nombre: result['a:Nombre'],*  *Username: result['a:Username'],*  *Password: result['a:Password'],*  *Telefono: result['a:Telefono']*  *};*  *}*  *return null;*  *};*  *// CREAR USUARIO*  *export const crearUsuario = async (usuario) => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/" xmlns:ec="http://schemas.datacontract.org/2004/07/ec.edu.monster.modelo">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:CrearUsuario>*  *<tem:usuario>*  *<ec:IdUsuario>${usuario.IdUsuario || 0}</ec:IdUsuario>*  *<ec:Nombre>${usuario.Nombre}</ec:Nombre>*  *<ec:Username>${usuario.Username}</ec:Username>*  *<ec:Password>${usuario.Password}</ec:Password>*  *<ec:Telefono>${usuario.Telefono}</ec:Telefono>*  *</tem:usuario>*  *</tem:CrearUsuario>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/CrearUsuario'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const result = json['s:Envelope']?.['s:Body']?.['CrearUsuarioResponse']?.['CrearUsuarioResult'];*  *return result === 'true' || result === true;*  *};*  *// EDITAR USUARIO*  *export const editarUsuario = async (usuario) => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/" xmlns:ec="http://schemas.datacontract.org/2004/07/ec.edu.monster.modelo">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:EditarUsuario>*  *<tem:usuario>*  *<ec:IdUsuario>${usuario.IdUsuario}</ec:IdUsuario>*  *<ec:Nombre>${usuario.Nombre}</ec:Nombre>*  *<ec:Username>${usuario.Username}</ec:Username>*  *<ec:Password>${usuario.Password}</ec:Password>*  *<ec:Telefono>${usuario.Telefono}</ec:Telefono>*  *</tem:usuario>*  *</tem:EditarUsuario>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/EditarUsuario'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const result = json['s:Envelope']?.['s:Body']?.['EditarUsuarioResponse']?.['EditarUsuarioResult'];*  *return result === 'true' || result === true;*  *};*  *// ELIMINAR USUARIO*  *export const eliminarUsuario = async (id) => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:EliminarUsuario>*  *<tem:id>${id}</tem:id>*  *</tem:EliminarUsuario>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/EliminarUsuario'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const result = json['s:Envelope']?.['s:Body']?.['EliminarUsuarioResponse']?.['EliminarUsuarioResult'];*  *return result === 'true' || result === true;*  *};*  *// OBTENER TODOS LOS USUARIOS*  *export const getUsuarios = async () => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:GetUsuarios/>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/GetUsuarios'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const usuarios = json['s:Envelope']?.['s:Body']?.['GetUsuariosResponse']?.['GetUsuariosResult']?.['a:Usuario'];*  *if (!usuarios) return [];*  *return Array.isArray(usuarios) ? usuarios : [usuarios];*  *};*  *// OBTENER BOLETOS POR USUARIO*  *export const obtenerBoletosPorUsuario = async (idUsuario) => {*  *const body = `*  *<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">*  *<soapenv:Header/>*  *<soapenv:Body>*  *<tem:ObtenerBoletosPorUsuario>*  *<tem:idUsuario>${idUsuario}</tem:idUsuario>*  *</tem:ObtenerBoletosPorUsuario>*  *</soapenv:Body>*  *</soapenv:Envelope>`;*  *const response = await fetch(endpoint, {*  *method: 'POST',*  *headers: {*  *'Content-Type': 'text/xml;charset=UTF-8',*  *SOAPAction: 'http://tempuri.org/IAeroCondorController/ObtenerBoletosPorUsuario'*  *},*  *body*  *});*  *const xml = await response.text();*  *const json = parser.parse(xml);*  *const boletos = json['s:Envelope']?.['s:Body']?.['ObtenerBoletosPorUsuarioResponse']?.['ObtenerBoletosPorUsuarioResult']?.['a:Boleto'];*  *if (!boletos) return [];*  *return Array.isArray(boletos) ? boletos : [boletos];*  *};* |

Tabla 23. **Código de VueloController.js**

|  |
| --- |
| import { XMLParser } from 'fast-xml-parser';  const parser = new XMLParser({ ignoreAttributes: false });  const endpoint = 'http://192.168.18.158:8094/ec.edu.monster.controlador/AeroCondorController.svc';  export const obtenerVuelos = async () => {    const body = `      <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">        <s:Body>          <GetVuelos xmlns="http://tempuri.org/" />        </s:Body>      </s:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        SOAPAction: 'http://tempuri.org/IAeroCondorController/GetVuelos'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    const raw = json?.['s:Envelope']?.['s:Body']?.['GetVuelosResponse']?.['GetVuelosResult'];    const vuelos = raw?.['a:Vuelos'];    const lista = Array.isArray(vuelos) ? vuelos : vuelos ? [vuelos] : [];    // Mapeo limpio de campos    return lista.map(v => ({      IdVuelo: v['a:IdVuelo'],      CodigoVuelo: v['a:CodigoVuelo'],      HoraSalida: v['a:HoraSalida'],      Valor: v['a:Valor'],      Capacidad: v['a:Capacidad'],      Disponibles: v['a:Disponibles']    }));  };  export const buscarVuelos = async (origen, destino, fechaSalida) => {    const body = `    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">      <soapenv:Header/>      <soapenv:Body>        <tem:BuscarVuelos>          <tem:origen>${origen}</tem:origen>          <tem:destino>${destino}</tem:destino>          <tem:fechaSalida>${fechaSalida}</tem:fechaSalida>        </tem:BuscarVuelos>      </soapenv:Body>    </soapenv:Envelope>`;    const response = await fetch(endpoint, {      method: 'POST',      headers: {        'Content-Type': 'text/xml;charset=UTF-8',        SOAPAction: 'http://tempuri.org/IAeroCondorController/BuscarVuelos'      },      body    });    const xml = await response.text();    const json = parser.parse(xml);    const vuelosRaw = json?.['s:Envelope']?.['s:Body']?.['BuscarVuelosResponse']?.['BuscarVuelosResult']?.['a:Vuelos'];    const lista = Array.isArray(vuelosRaw) ? vuelosRaw : vuelosRaw ? [vuelosRaw] : [];    return lista.map(v => ({      IdVuelo: v['a:IdVuelo'],      CodigoVuelo: v['a:CodigoVuelo'],      HoraSalida: v['a:HoraSalida'],      Valor: v['a:Valor'],      Capacidad: v['a:Capacidad'],      Disponibles: v['a:Disponibles']    }));  }; |

Tabla 14. **Código de la vista LoginView.jsx**

|  |
| --- |
| *import React, { useState } from 'react';*  *import {*  *View,*  *TextInput,*  *Button,*  *Text,*  *StyleSheet,*  *Image,*  *TouchableOpacity,*  *Platform,*  *KeyboardAvoidingView,*  *Modal*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import { login } from '../app/controllers/UsuarioController';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function LoginView() {*  *const router = useRouter();*  *const [username, setUsername] = useState('');*  *const [password, setPassword] = useState('');*  *const [secure, setSecure] = useState(true);*  *const [loading, setLoading] = useState(false);*  *const [irARegistro, setIrARegistro] = useState(false);*  *const [modalVisible, setModalVisible] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [esExito, setEsExito] = useState(false);*  *const [usuarioAutenticado, setUsuarioAutenticado] = useState(null);*  *if (irARegistro) {*  *const RegisterView = require('./RegisterView').default;*  *return <RegisterView volver={() => setIrARegistro(false)} />;*  *}*  *const handleLogin = async () => {*  *if (!username.trim() || !password.trim()) {*  *setMensaje('⚠️ Todos los campos son obligatorios');*  *setEsExito(false);*  *setModalVisible(true);*  *return;*  *}*  *setLoading(true);*  *try {*  *const user = await login(username.trim(), password.trim());*  *if (user && user.idUsuario) {*  *await AsyncStorage.clear();*  *await AsyncStorage.setItem('idUsuario', user.idUsuario.toString());*  *setMensaje(`✅ Bienvenido ${user.nombre}`);*  *setUsuarioAutenticado(user);*  *setEsExito(true);*  *} else {*  *setMensaje('❌ Usuario o contraseña incorrectos');*  *setEsExito(false);*  *}*  *} catch (error) {*  *console.error('❌ Error de login:', error);*  *setMensaje('❌ Error al conectar con el servidor');*  *setEsExito(false);*  *} finally {*  *setLoading(false);*  *setModalVisible(true);*  *}*  *};*  *return (*  *<KeyboardAvoidingView*  *style={styles.container}*  *behavior={Platform.OS === 'ios' ? 'padding' : undefined}*  *>*  *<Image source={require('../assets/images/logo\_monster.png')} style={styles.logo} />*  *<Text style={styles.title}>MONSTER - VIAJESITOS</Text>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Usuario"*  *value={username}*  *onChangeText={setUsername}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Contraseña"*  *secureTextEntry={secure}*  *value={password}*  *onChangeText={setPassword}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *<TouchableOpacity onPress={() => setSecure(!secure)} style={styles.toggle}>*  *<Text style={styles.toggleText}>{secure ? 'Mostrar' : 'Ocultar'}</Text>*  *</TouchableOpacity>*  *</View>*  *<View style={styles.buttonContainer}>*  *<Button*  *title={loading ? 'Ingresando...' : 'INGRESAR'}*  *color="#5fb4a2"*  *onPress={handleLogin}*  *disabled={loading}*  */>*  *</View>*  *<TouchableOpacity onPress={() => setIrARegistro(true)} style={{ marginTop: 16 }}>*  *<Text style={{ color: '#4e88a9', fontWeight: '500' }}>*  *¿No tienes cuenta? Regístrate*  *</Text>*  *</TouchableOpacity>*  *<Modal*  *visible={modalVisible}*  *transparent*  *animationType="fade"*  *onRequestClose={() => setModalVisible(false)}*  *>*  *<View style={styles.modalOverlay}>*  *<View style={styles.modalContainer}>*  *<Text style={[styles.modalMessage, { color: esExito ? 'green' : 'red' }]}>*  *{mensaje}*  *</Text>*  *<TouchableOpacity*  *onPress={() => {*  *setModalVisible(false);*  *if (esExito && usuarioAutenticado) {*  *setUsername('');*  *setPassword('');*  *router.replace({*  *pathname: '/views/MenuView',*  *params: {*  *idUsuario: usuarioAutenticado.idUsuario,*  *nombre: usuarioAutenticado.nombre*  *}*  *});*  *}*  *}}*  *style={styles.modalButton}*  *>*  *<Text style={{ color: '#fff', fontWeight: 'bold' }}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 24. **Código RegisterView.jsx**

|  |
| --- |
| *import React, { useState } from 'react';*  *import {*  *View,*  *TextInput,*  *Button,*  *Text,*  *StyleSheet,*  *Platform,*  *KeyboardAvoidingView,*  *Image,*  *TouchableOpacity,*  *Modal*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import { crearUsuario } from './controllers/UsuarioController';*  *export default function RegisterView() {*  *const router = useRouter();*  *const [nombre, setNombre] = useState('');*  *const [username, setUsername] = useState('');*  *const [password, setPassword] = useState('');*  *const [telefono, setTelefono] = useState('');*  *const [secure, setSecure] = useState(true);*  *const [loading, setLoading] = useState(false);*  *const [modalVisible, setModalVisible] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [esExito, setEsExito] = useState(false);*  *const handleRegister = async () => {*  *if (!nombre.trim() || !username.trim() || !password.trim() || !telefono.trim()) {*  *setMensaje('⚠️ Todos los campos son obligatorios');*  *setEsExito(false);*  *setModalVisible(true);*  *return;*  *}*  *setLoading(true);*  *try {*  *const success = await crearUsuario({*  *idUsuario: 0,*  *nombre: nombre.trim(),*  *username: username.trim(),*  *password: password.trim(),*  *telefono: telefono.trim()*  *});*  *if (success === true) {*  *setMensaje('✅ Registro exitoso. Ya puedes iniciar sesión.');*  *setEsExito(true);*  *} else {*  *setMensaje('❌ El usuario ya está registrado o ocurrió un error.');*  *setEsExito(false);*  *}*  *} catch (error) {*  *setMensaje('❌ Error de conexión o del servidor.');*  *setEsExito(false);*  *} finally {*  *setLoading(false);*  *setModalVisible(true);*  *}*  *};*  *const volverAlLogin = () => {*  *setModalVisible(false);*  *if (esExito) router.replace('/LoginView');*  *};*  *return (*  *<KeyboardAvoidingView*  *style={styles.container}*  *behavior={Platform.OS === 'ios' ? 'padding' : undefined}*  *>*  *<Image source={require('../assets/images/logo\_monster.png')} style={styles.logo} />*  *<Text style={styles.title}>Crear cuenta</Text>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Nombre completo"*  *value={nombre}*  *onChangeText={setNombre}*  *placeholderTextColor="#888"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Usuario"*  *value={username}*  *onChangeText={setUsername}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Contraseña"*  *secureTextEntry={secure}*  *value={password}*  *onChangeText={setPassword}*  *placeholderTextColor="#888"*  *autoCapitalize="none"*  */>*  *<TouchableOpacity onPress={() => setSecure(!secure)} style={styles.toggle}>*  *<Text style={styles.toggleText}>{secure ? 'Mostrar' : 'Ocultar'}</Text>*  *</TouchableOpacity>*  *</View>*  *<View style={styles.inputWrapper}>*  *<TextInput*  *style={styles.input}*  *placeholder="Teléfono"*  *keyboardType="phone-pad"*  *value={telefono}*  *onChangeText={setTelefono}*  *placeholderTextColor="#888"*  */>*  *</View>*  *<View style={styles.buttonContainer}>*  *<Button*  *title={loading ? 'Registrando...' : 'REGISTRAR'}*  *color="#5fb4a2"*  *onPress={handleRegister}*  *disabled={loading}*  */>*  *</View>*  *<TouchableOpacity onPress={() => router.replace('/LoginView')} style={{ marginTop: 16 }}>*  *<Text style={{ color: '#4e88a9', fontWeight: '500' }}>*  *¿Ya tienes cuenta? Inicia sesión*  *</Text>*  *</TouchableOpacity>*  *<Modal*  *visible={modalVisible}*  *transparent*  *animationType="fade"*  *onRequestClose={() => setModalVisible(false)}*  *>*  *<View style={styles.modalOverlay}>*  *<View style={styles.modalContainer}>*  *<Text style={[styles.modalMessage, { color: esExito ? 'green' : 'red' }]}>*  *{mensaje}*  *</Text>*  *<TouchableOpacity*  *onPress={volverAlLogin}*  *style={styles.modalButton}*  *>*  *<Text style={{ color: '#fff', fontWeight: 'bold' }}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 25. **Código ComprarBoletoView.jsx**

|  |
| --- |
| *import React, { useEffect, useState } from 'react';*  *import {*  *View,*  *Text,*  *FlatList,*  *TouchableOpacity,*  *Modal,*  *ActivityIndicator,*  *TextInput,*  *Alert,*  *KeyboardAvoidingView,*  *StyleSheet,*  *ScrollView,*  *useWindowDimensions*  *} from 'react-native';*  *import { useRouter } from 'expo-router';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *import { obtenerCiudades } from '../controllers/CiudadController';*  *import { buscarVuelos as buscarVuelosAPI } from '../controllers/VueloController';*  *import { registrarBoleto } from '../controllers/BoletoController';*  *export default function ComprarBoletoView() {*  *const router = useRouter();*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const [usuario, setUsuario] = useState(null);*  *const [ciudades, setCiudades] = useState([]);*  *const [origen, setOrigen] = useState('');*  *const [destino, setDestino] = useState('');*  *const [fecha, setFecha] = useState('');*  *const [vuelos, setVuelos] = useState([]);*  *const [loading, setLoading] = useState(false);*  *const [mensaje, setMensaje] = useState('');*  *const [modalVisible, setModalVisible] = useState(false);*  *const [cantidad, setCantidad] = useState(1);*  *useEffect(() => {*  *const cargar = async () => {*  *try {*  *const data = await obtenerCiudades();*  *setCiudades(Array.isArray(data) ? data : [data]);*  *} catch (e) {*  *console.error('Error cargando ciudades:', e);*  *setCiudades([]);*  *}*  *const storedId = await AsyncStorage.getItem('idUsuario');*  *if (storedId) setUsuario(parseInt(storedId));*  *else router.replace('/');*  *};*  *cargar();*  *}, []);*  *const getNombreCiudad = (codigo) => {*  *const ciudad = ciudades.find(c => c.codigoCiudad === codigo);*  *return ciudad ? ciudad.nombreCiudad : codigo;*  *};*  *const limpiarFormulario = () => {*  *setOrigen('');*  *setDestino('');*  *setFecha('');*  *setVuelos([]);*  *setCantidad(1);*  *};*  *const transformVuelo = (vuelo) => ({*  *...vuelo,*  *IdVuelo: vuelo.idVuelo,*  *CodigoVuelo: vuelo.codigoVuelo,*  *Valor: vuelo.valor,*  *Disponibles: vuelo.disponibles,*  *HoraSalida: vuelo.horaSalida,*  *Origen: vuelo.idCiudadOrigen?.codigoCiudad || 'ND',*  *Destino: vuelo.idCiudadDestino?.codigoCiudad || 'ND',*  *});*  *const handleBuscarVuelos = async () => {*  *if (!origen || !destino || origen === destino || !/^\d{4}-\d{2}-\d{2}$/.test(fecha)) {*  *Alert.alert('Error', 'Seleccione ciudades válidas y una fecha con formato correcto (YYYY-MM-DD).');*  *return;*  *}*  *setLoading(true);*  *try {*  *const resultados = await buscarVuelosAPI(origen, destino, fecha);*  *const lista = Array.isArray(resultados)*  *? resultados.map(transformVuelo)*  *: resultados*  *? [transformVuelo(resultados)]*  *: [];*  *setVuelos(lista);*  *if (lista.length === 0) Alert.alert('Sin vuelos disponibles');*  *} catch (error) {*  *console.error('Error al buscar vuelos:', error);*  *Alert.alert('Error', 'No se pudo buscar vuelos.');*  *} finally {*  *setLoading(false);*  *}*  *};*  *const handleComprar = async (vuelo) => {*  *if (!usuario || isNaN(usuario)) {*  *setMensaje('❌ ID de usuario no disponible.');*  *setModalVisible(true);*  *return;*  *}*  *if (!cantidad || isNaN(cantidad) || cantidad <= 0 || cantidad > vuelo.Disponibles) {*  *setMensaje(`❌ Cantidad inválida. Disponible: ${vuelo.Disponibles}`);*  *setModalVisible(true);*  *return;*  *}*  *const total = (parseFloat(vuelo.Valor) \* cantidad).toFixed(2);*  *try {*  *const resultado = await registrarBoleto({*  *idVuelo: vuelo.IdVuelo,*  *idUsuario: usuario,*  *cantidad*  *});*  *if (resultado) {*  *setMensaje(*  *`✅ Compra realizada\n\n✈ Vuelo: ${vuelo.CodigoVuelo}\n🛫 Origen: ${getNombreCiudad(vuelo.Origen)}\n🛬 Destino: ${getNombreCiudad(vuelo.Destino)}\n🎟 Cantidad: ${cantidad}\n💵 Total: $${total}`*  *);*  *setModalVisible(true);*  *limpiarFormulario();*  *setTimeout(() => {*  *setModalVisible(false);*  *router.replace({ pathname: '/views/MenuView', params: { idUsuario: usuario } });*  *}, 3000);*  *} else {*  *setMensaje('❌ No se pudo completar la compra');*  *setModalVisible(true);*  *}*  *} catch (e) {*  *console.error('Error al comprar:', e);*  *setMensaje('❌ Error inesperado.');*  *setModalVisible(true);*  *}*  *};*  *const renderCiudadItem = (item, onSelect, selected) => (*  *<TouchableOpacity*  *key={item.codigoCiudad}*  *style={[styles.ciudadBtn, selected === item.codigoCiudad && styles.ciudadBtnSelected]}*  *onPress={() => onSelect(item.codigoCiudad)}*  *>*  *<Text style={styles.ciudadText}>{item.codigoCiudad} - {item.nombreCiudad}</Text>*  *</TouchableOpacity>*  *);*  *const renderVueloItem = ({ item }) => (*  *<View style={isMobile ? styles.card : styles.cardDesktop}>*  *<Text style={styles.title}>✈️ {item.CodigoVuelo}</Text>*  *<Text>🛫 Origen: {getNombreCiudad(item.Origen)}</Text>*  *<Text>🛬 Destino: {getNombreCiudad(item.Destino)}</Text>*  *<Text>Salida: {item.HoraSalida}</Text>*  *<Text>Precio: ${item.Valor}</Text>*  *<Text>Disponibles: {item.Disponibles}</Text>*  *<View style={styles.stepperContainer}>*  *<TouchableOpacity*  *onPress={() => setCantidad(Math.max(1, cantidad - 1))}*  *style={styles.stepperBtn}*  *>*  *<Text style={styles.stepperText}>−</Text>*  *</TouchableOpacity>*  *<Text style={styles.stepperCount}>{cantidad}</Text>*  *<TouchableOpacity*  *onPress={() => setCantidad(Math.min(10, cantidad + 1))}*  *style={styles.stepperBtn}*  *>*  *<Text style={styles.stepperText}>+</Text>*  *</TouchableOpacity>*  *</View>*  *<TouchableOpacity style={styles.btn} onPress={() => handleComprar(item)}>*  *<Text style={styles.btnText}>Comprar</Text>*  *</TouchableOpacity>*  *</View>*  *);*  *return (*  *<KeyboardAvoidingView behavior="padding" style={styles.scroll}>*  *<ScrollView>*  *<View style={styles.container}>*  *<Text style={styles.header}>Buscar Vuelos</Text>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Ciudad de Origen</Text>*  *<ScrollView horizontal contentContainerStyle={styles.selectorContainer}>*  *{ciudades.filter(c => c.codigoCiudad !== destino).map(c => renderCiudadItem(c, setOrigen, origen))}*  *</ScrollView>*  *</View>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Ciudad de Destino</Text>*  *<ScrollView horizontal contentContainerStyle={styles.selectorContainer}>*  *{ciudades.filter(c => c.codigoCiudad !== origen).map(c => renderCiudadItem(c, setDestino, destino))}*  *</ScrollView>*  *</View>*  *<View style={styles.formGroup}>*  *<Text style={styles.subheader}>Fecha de Vuelo</Text>*  *<TextInput*  *style={styles.input}*  *placeholder="YYYY-MM-DD"*  *value={fecha}*  *onChangeText={setFecha}*  */>*  *</View>*  *<TouchableOpacity style={styles.btnBuscar} onPress={handleBuscarVuelos}>*  *<Text style={styles.btnText}>Buscar Vuelos</Text>*  *</TouchableOpacity>*  *{loading ? (*  *<ActivityIndicator size="large" color="#35798e" />*  *) : (*  *<FlatList*  *data={vuelos}*  *keyExtractor={(item) => `vuelo-${item.IdVuelo}`}*  *renderItem={renderVueloItem}*  *contentContainerStyle={{ paddingBottom: 80 }}*  *ListEmptyComponent={<Text style={styles.noVuelos}>No hay vuelos disponibles.</Text>}*  */>*  *)}*  *</View>*  *<Modal visible={modalVisible} transparent animationType="slide">*  *<View style={styles.modalContainer}>*  *<View style={styles.modalContent}>*  *<Text style={styles.modalText}>{mensaje}</Text>*  *<TouchableOpacity*  *style={styles.modalButton}*  *onPress={() => {*  *setModalVisible(false);*  *if (mensaje.startsWith('✅')) {*  *router.replace({ pathname: '/views/MenuView', params: { idUsuario: usuario } });*  *}*  *}}*  *>*  *<Text style={styles.modalButtonText}>Aceptar</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</Modal>*  *</ScrollView>*  *</KeyboardAvoidingView>*  *);*  *}* |

Tabla 26. **Código MisBoletosView.jsx**

|  |
| --- |
| *import React, { useState, useCallback } from 'react';*  *import {*  *View,*  *Text,*  *FlatList,*  *ScrollView,*  *StyleSheet,*  *ActivityIndicator,*  *TouchableOpacity,*  *useWindowDimensions,*  *SafeAreaView,*  *StatusBar,*  *} from 'react-native';*  *import { useRouter, useLocalSearchParams, useFocusEffect } from 'expo-router';*  *import { obtenerBoletosPorUsuario } from '../controllers/BoletoController';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function MisBoletosView() {*  *const router = useRouter();*  *const { idUsuario: idParam } = useLocalSearchParams();*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const [boletos, setBoletos] = useState([]);*  *const [loading, setLoading] = useState(true);*  *useFocusEffect(*  *useCallback(() => {*  *const cargarBoletos = async () => {*  *setLoading(true);*  *setBoletos([]);*  *try {*  *let idUsuarioActual = idParam;*  *if (!idUsuarioActual) {*  *idUsuarioActual = await AsyncStorage.getItem('idUsuario');*  *if (!idUsuarioActual) throw new Error('Usuario no autenticado');*  *} else {*  *await AsyncStorage.setItem('idUsuario', idUsuarioActual);*  *}*  *const datos = await obtenerBoletosPorUsuario(idUsuarioActual);*  *if (Array.isArray(datos)) {*  *const ordenados = datos.sort((a, b) =>*  *new Date(b.fechaCompra) - new Date(a.fechaCompra)*  *);*  *setBoletos(ordenados);*  *} else {*  *setBoletos([]);*  *}*  *} catch (error) {*  *console.error('❌ Error al cargar boletos:', error);*  *setBoletos([]);*  *} finally {*  *setLoading(false);*  *}*  *};*  *cargarBoletos();*  *}, [idParam])*  *);*  *const parseFechaCompra = (fechaStr) => {*  *if (!fechaStr || typeof fechaStr !== 'string') return null;*  *const cleanStr = fechaStr.replace(/\[.\*?\]/g, '');*  *const date = new Date(cleanStr);*  *return isNaN(date) ? null : date;*  *};*  *const formatDate = (dateString) => {*  *const date = parseFechaCompra(dateString);*  *if (!date) return 'Fecha inválida';*  *return date.toLocaleString('es-EC', {*  *day: '2-digit',*  *month: '2-digit',*  *year: 'numeric',*  *hour: '2-digit',*  *minute: '2-digit',*  *});*  *};*  *const renderItem = ({ item, index }) => {*  *if (!item || !item.numeroBoleto) return null;*  *return (*  *<View style={[styles.card, { opacity: loading ? 0.5 : 1 }]}>*  *<View style={styles.cardHeader}>*  *<Text style={styles.cardTitle}>🎫 Boleto #{item.numeroBoleto}</Text>*  *<View style={styles.badge}>*  *<Text style={styles.badgeText}>#{index + 1}</Text>*  *</View>*  *</View>*  *<View style={styles.cardContent}>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>✈️ Vuelo:</Text>*  *<Text style={styles.value}>{item.idVuelo?.codigoVuelo || 'Desconocido'}</Text>*  *</View>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>📅 Fecha de Compra:</Text>*  *<Text style={styles.value}>{formatDate(item.fechaCompra)}</Text>*  *</View>*  *<View style={styles.infoRow}>*  *<Text style={styles.label}>💰 Precio:</Text>*  *<Text style={styles.priceValue}>${item.precioCompra}</Text>*  *</View>*  *</View>*  *</View>*  *);*  *};*  *const handleVolverMenu = async () => {*  *let idUsuarioActual = idParam;*  *if (!idUsuarioActual) {*  *idUsuarioActual = await AsyncStorage.getItem('idUsuario');*  *}*  *if (idUsuarioActual) {*  *router.replace({*  *pathname: '/views/MenuView',*  *params: { idUsuario: idUsuarioActual }*  *});*  *} else {*  *router.replace('/');*  *}*  *};*  *return (*  *<SafeAreaView style={styles.safeArea}>*  *<StatusBar barStyle="dark-content" backgroundColor="#f8f9fa" />*  *<View style={styles.container}>*  *<View style={styles.headerContainer}>*  *<Text style={styles.header}>🎫 Mis Boletos</Text>*  *<Text style={styles.subtitle}>*  *{loading ? 'Cargando...' : `${boletos.length} boleto${boletos.length !== 1 ? 's' : ''} encontrado${boletos.length !== 1 ? 's' : ''}`}*  *</Text>*  *</View>*  *<ScrollView*  *style={styles.mainContent}*  *contentContainerStyle={styles.scrollContent}*  *showsVerticalScrollIndicator={true}*  *>*  *{loading ? (*  *<View style={styles.loadingContainer}>*  *<ActivityIndicator size="large" color="#4e88a9" />*  *<Text style={styles.loadingText}>Cargando tus boletos...</Text>*  *</View>*  *) : boletos.length === 0 ? (*  *<View style={styles.emptyContainer}>*  *<Text style={styles.emptyIcon}>✈️</Text>*  *<Text style={styles.emptyTitle}>No tienes boletos</Text>*  *<Text style={styles.emptyText}>Aún no has comprado ningún boleto.{"\n"}¡Reserva tu próximo vuelo!</Text>*  *</View>*  *) : isMobile ? (*  *<FlatList*  *data={boletos}*  *keyExtractor={(item, index) =>*  *item && item.numeroBoleto ? String(item.numeroBoleto) : `boleto-${index}`*  *}*  *renderItem={renderItem}*  *showsVerticalScrollIndicator={true}*  *contentContainerStyle={styles.listContainer}*  *ItemSeparatorComponent={() => <View style={styles.separator} />}*  */>*  *) : (*  *<View style={styles.tableContainer}>*  *<View style={styles.tableHeader}>*  *<Text style={styles.tableCellHeader}>#</Text>*  *<Text style={styles.tableCellHeader}>Número</Text>*  *<Text style={styles.tableCellHeader}>Vuelo</Text>*  *<Text style={styles.tableCellHeader}>Compra</Text>*  *<Text style={styles.tableCellHeader}>Precio</Text>*  *</View>*  *{boletos.map((item, index) => (*  *<View key={item.numeroBoleto || index} style={styles.tableRow}>*  *<Text style={styles.tableCell}>{index + 1}</Text>*  *<Text style={styles.tableCell}>{item.numeroBoleto}</Text>*  *<Text style={styles.tableCell}>{item.idVuelo?.codigoVuelo || 'ND'}</Text>*  *<Text style={styles.tableCell}>{formatDate(item.fechaCompra)}</Text>*  *<Text style={styles.tableCell}>${item.precioCompra}</Text>*  *</View>*  *))}*  *</View>*  *)}*  *</ScrollView>*  *<View style={styles.buttonContainer}>*  *<TouchableOpacity*  *onPress={handleVolverMenu}*  *style={styles.volverBtn}*  *activeOpacity={0.8}*  *>*  *<Text style={styles.volverText}>← Volver al Menú</Text>*  *</TouchableOpacity>*  *</View>*  *</View>*  *</SafeAreaView>*  *);*  *}* |

Tabla 18. **Código VuelosDisponiblesView.jsx**

|  |
| --- |
| *import React, { useEffect, useState, useCallback } from 'react';*  *import {*  *View,*  *Text,*  *StyleSheet,*  *ActivityIndicator,*  *FlatList,*  *Pressable,*  *SafeAreaView,*  *useWindowDimensions,*  *} from 'react-native';*  *import { obtenerVuelos } from '../controllers/VueloController';*  *import { useRouter, useLocalSearchParams, useFocusEffect } from 'expo-router';*  *import AsyncStorage from '@react-native-async-storage/async-storage';*  *export default function VuelosDisponiblesView() {*  *const [vuelos, setVuelos] = useState([]);*  *const [loading, setLoading] = useState(true);*  *const { width } = useWindowDimensions();*  *const isMobile = width < 768;*  *const router = useRouter();*  *const { idUsuario: idParam, nombre: nombreParam } = useLocalSearchParams();*  *const [idUsuario, setIdUsuario] = useState(null);*  *const [nombre, setNombre] = useState(nombreParam || 'Viajero');*  *useFocusEffect(*  *useCallback(() => {*  *const cargar = async () => {*  *let id = idParam;*  *if (!id) {*  *id = await AsyncStorage.getItem('idUsuario');*  *} else {*  *await AsyncStorage.setItem('idUsuario', id);*  *}*  *setIdUsuario(id);*  *setNombre(nombreParam || (await AsyncStorage.getItem('nombre')) || 'Viajero');*  *};*  *cargar();*  *}, [idParam, nombreParam])*  *);*  *useEffect(() => {*  *const cargarVuelos = async () => {*  *try {*  *const data = await obtenerVuelos();*  *const vuelosOrdenados = Array.isArray(data)*  *? [...data].sort((a, b) => new Date(b.horaSalida) - new Date(a.horaSalida))*  *: [];*  *setVuelos(vuelosOrdenados);*  *} catch (error) {*  *console.error('❌ Error al obtener vuelos:', error);*  *setVuelos([]);*  *} finally {*  *setLoading(false);*  *}*  *};*  *cargarVuelos();*  *}, []);*  *const renderItem = ({ item }) => {*  *if (!item || !item.codigoVuelo) return null;*  *return (*  *<View style={styles.card}>*  *<Text style={styles.title}>✈️ {item.codigoVuelo}</Text>*  *<Text>Ruta: {item.idCiudadOrigen?.nombreCiudad} ➡ {item.idCiudadDestino?.nombreCiudad}</Text>*  *<Text>Hora salida: {item.horaSalida}</Text>*  *<Text>Precio: ${item.valor}</Text>*  *<Text>Capacidad: {item.capacidad}</Text>*  *<Text>Disponibles: {item.disponibles}</Text>*  *</View>*  *);*  *};*  *const renderTabla = () => (*  *<View style={styles.tabla}>*  *<View style={styles.filaHeader}>*  *<Text style={styles.colHeader}>Código</Text>*  *<Text style={styles.colHeader}>Ruta</Text>*  *<Text style={styles.colHeader}>Salida</Text>*  *<Text style={styles.colHeader}>Precio</Text>*  *<Text style={styles.colHeader}>Capacidad</Text>*  *<Text style={styles.colHeader}>Disponibles</Text>*  *</View>*  *{vuelos.map((v, i) =>*  *v && v.codigoVuelo ? (*  *<View key={i} style={styles.fila}>*  *<Text style={styles.col}>{v.codigoVuelo}</Text>*  *<Text style={styles.col}>{v.idCiudadOrigen?.nombreCiudad} ➡ {v.idCiudadDestino?.nombreCiudad}</Text>*  *<Text style={styles.col}>{v.horaSalida}</Text>*  *<Text style={styles.col}>${v.valor}</Text>*  *<Text style={styles.col}>{v.capacidad}</Text>*  *<Text style={styles.col}>{v.disponibles}</Text>*  *</View>*  *) : null*  *)}*  *</View>*  *);*  *return (*  *<SafeAreaView style={styles.safeArea}>*  *<View style={styles.container}>*  *<Text style={styles.header}>Vuelos Disponibles</Text>*  *{loading ? (*  *<ActivityIndicator size="large" color="#4e88a9" />*  *) : vuelos.length === 0 ? (*  *<Text style={styles.vacio}>No hay vuelos disponibles.</Text>*  *) : isMobile ? (*  *<FlatList*  *data={vuelos}*  *renderItem={renderItem}*  *keyExtractor={(item, index) =>*  *item && item.idVuelo ? `${item.idVuelo}` : `vuelo-${index}`*  *}*  *contentContainerStyle={styles.flatListContainer}*  *showsVerticalScrollIndicator={false}*  */>*  *) : (*  *renderTabla()*  *)}*  *<Pressable*  *onPress={() =>*  *router.replace({*  *pathname: '/views/MenuView',*  *params: { idUsuario, nombre },*  *})*  *}*  *style={styles.botonVolver}*  *>*  *<Text style={styles.botonTexto}>🔙 Volver al Menú</Text>*  *</Pressable>*  *</View>*  *</SafeAreaView>*  *);*  *}* |

Tabla 27. **Código index.tsx**

|  |
| --- |
| *// app/index.tsx*  *import LoginView from './LoginView';*  *export default LoginView;* |

Tabla 28. **Código \_layout.tsx**

|  |
| --- |
| *import { Drawer } from 'expo-router/drawer';*  *export default function Layout() {*  *return (*  *<Drawer*  *screenOptions={{*  *headerStyle: { backgroundColor: '#eaf6f6' },*  *headerTintColor: '#35798e',*  *drawerStyle: { backgroundColor: '#f0f4f8' },*  *}}*  *>*  *{/\* PANTALLAS OCULTAS DEL DRAWER \*/}*  *<Drawer.Screen*  *name="index"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *<Drawer.Screen*  *name="LoginView"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *<Drawer.Screen*  *name="RegisterView"*  *options={{*  *drawerItemStyle: { display: 'none' },*  *headerShown: false*  *}}*  */>*  *{/\* PANTALLAS VISIBLES EN EL DRAWER \*/}*  *<Drawer.Screen name="views/MenuView" options={{ title: 'Menú Principal' }} />*  *<Drawer.Screen name="views/MisBoletosView" options={{ title: 'Mis Boletos' }} />*  *<Drawer.Screen name="views/VuelosDisponiblesView" options={{ title: 'Vuelos Disponibles' }} />*  *<Drawer.Screen name="views/ComprarBoletoView" options={{ title: 'Comprar Boletos' }} />*  *</Drawer>*  *);*  *}* |

Finalmente se añade estilos css para dar un estilo agradable a la aplicación web.

Tabla 29. **Estilo de la aplicación web styles.css**

|  |
| --- |
| *const styles = StyleSheet.create({*  *safeArea: {*  *flex: 1,*  *backgroundColor: '#fff',*  *},*  *container: {*  *flex: 1,*  *padding: 20,*  *alignItems: 'center',*  *},*  *header: {*  *fontSize: 26,*  *fontWeight: 'bold',*  *marginBottom: 20,*  *color: '#35798e',*  *},*  *vacio: {*  *fontSize: 16,*  *color: '#6c757d',*  *textAlign: 'center',*  *marginTop: 20,*  *},*  *card: {*  *backgroundColor: '#e6f7f7',*  *padding: 16,*  *borderRadius: 10,*  *marginBottom: 15,*  *width: '100%',*  *borderColor: '#cfe0e8',*  *borderWidth: 1,*  *},*  *title: {*  *fontSize: 18,*  *fontWeight: '600',*  *marginBottom: 6,*  *color: '#4e88a9',*  *},*  *tabla: {*  *borderWidth: 1,*  *borderColor: '#cfe0e8',*  *width: '100%',*  *maxWidth: 1000,*  *},*  *filaHeader: {*  *flexDirection: 'row',*  *backgroundColor: '#dff6f6',*  *paddingVertical: 10,*  *},*  *fila: {*  *flexDirection: 'row',*  *borderTopWidth: 1,*  *borderColor: '#cfe0e8',*  *paddingVertical: 8,*  *},*  *colHeader: {*  *flex: 1,*  *fontWeight: 'bold',*  *textAlign: 'center',*  *color: '#35798e',*  *},*  *col: {*  *flex: 1,*  *textAlign: 'center',*  *color: '#212529',*  *},*  *botonVolver: {*  *marginTop: 30,*  *backgroundColor: '#4e88a9',*  *paddingHorizontal: 20,*  *paddingVertical: 10,*  *borderRadius: 8,*  *},*  *botonTexto: {*  *color: '#fff',*  *fontSize: 16,*  *fontWeight: '600',*  *},*  *flatListContainer: {*  *paddingBottom: 20,*  *},*  *});* |

Con ello la aplicación móvil esta lista para ser ejecutada, para este fin se usará expo Go el cual permite ejecutar la aplicación desde el celular.

### EJECUCIÓN DE LA APLICACIÓN

Finalmente se ejecuta la aplicación móvil y se prueba las funcionalidades del Web Service, primero se procede a probar el login de la aplicación.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 42. Login en aplicación móvil.

A continuación, se procede a probar las funciones.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura 43. Registro en aplicación de móvil

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Figura 44. Funciones en aplicación de móvil

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Figura 45. Funciones en aplicación de móvil

Con ello se finalizó la aplicación móvil, consumiendo un Web Service del servidor y creando las interfaces de usuario de login y menú con las funcionalidades operativas.

# CONCLUSIONES

* El servidor Dotnet SOAP y su capa han permitido integrar de forma homogénea los clientes web, móvil, de consola y de escritorio Java, evitando duplicar lógica de negocio.
* La arquitectura MVC usada en todas las interfaces facilita la mantenibilidad y agiliza futuras ampliaciones funcionales.
* La reutilización de controladores SOAP en los distintos clientes demuestra la portabilidad del núcleo de servicios y garantiza coherencia en los resultados mostrados al usuario.

# RECOMENDACIONES

* Incorporar autenticación con JWT para un manejo seguro y unificado de sesiones en todos los clientes.
* Automatizar pruebas de integración entre servidor y clientes para prevenir regresiones en cada despliegue.
* Publicar la app móvil mediante EAS Build en tiendas oficiales, ampliando el alcance comercial de Viajecitos SA.

# BIBLIOGRAFÍA

* IBM. (s.f.). SOAP. Recuperado de <https://www.ibm.com/docs/es/rsas/7.5.0?topic=standards-soap>
* F5. (s.f.). ¿Qué es el Protocolo Simple de Acceso a Objetos (SOAP)?. Recuperado de <https://www.f5.com/es_es/glossary/simple-object-access-protocol-soap>
* Atlassian. (s.f.). Arquitectura de microservicios. Recuperado de <https://www.atlassian.com/es/microservices/microservices-architecture>
* Paradigma Digital. (s.f.). Patrones de arquitectura de microservicios, ¿qué son y qué ventajas ofrecen?. Recuperado de <https://www.paradigmadigital.com/dev/patrones-arquitectura-microservicios-que-son-ventajas/>
* Profile. (2024). ¿Qué es la arquitectura de software? Fundamentos clave. Recuperado de <https://profile.es/blog/que-es-la-arquitectura-de-software/>

# ANEXOS

Diagrama Entidad Relación

Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama de Clases

Diagrama

El contenido generado por IA puede ser incorrecto.

Diagrama de casos de uso

Diagrama

El contenido generado por IA puede ser incorrecto.