|  |
| --- |
| DUOC UC - Escuela de informática y telecomunicaciones |
| Proyecto Capstone |
| *Proyecto: Airlink* |
|  |
| **Revisión*: [01]*** |
| **23/08/2025** |

|  |
| --- |
| Planificación y Especificación de Requisitos según estándares; IEEE 830, ISO9000 y PMI. |

**Contenido**

[*DUOC UC - Escuela de informática y telecomunicaciones*](#_heading=h.gjdgxs) *1*

[**Ficha del documento**](#_heading=h.1fob9te) **4**

[**1. Introducción**](#_heading=h.3znysh7) **5**

[1.1. Propósito](#_heading=h.2et92p0) 5

[1.2. Ámbito del Sistema](#_heading=h.tyjcwt) 5

[1.3. Definiciones, Acrónimos y Abreviaturas](#_heading=h.3dy6vkm) 6

[1.4. Referencias](#_heading=h.1t3h5sf) 7

[*1) Minuta Kick off*](#_heading=h.cxchta1cd6ob) *7*

[*2) Acta de Proyecto*](#_heading=h.vosspu7iuw4d) *7*

[*3) Matriz Especificación de Requerimientos*](#_heading=h.3ueb6j5jepjg) *7*

[*4)Diagrama de Casos de Uso General*](#_heading=h.otsm70qcozxq) *7*

[*5) Planilla Casos de Uso Extendido*](#_heading=h.o1c055fhkbxi) *7*

[*6) Prototipado de Software*](#_heading=h.2meyys3uywax) *7*

[*7) Matriz EDT. Planilla Detallada Cálculo de Esfuerzo*](#_heading=h.mxqg723o5fc) *7*

[*8) Planilla Carta Gantt*](#_heading=h.ww8crerfbfn2) *7*

[9) Visión General del Documento](#_heading=h.4d34og8) 7

[**2. Descripción General**](#_heading=h.2s8eyo1) **8**

[2.1. Perspectiva del Producto](#_heading=h.17dp8vu) 8

[2.2. Funciones del Producto](#_heading=h.3rdcrjn) 8

[2.3. Características de los Usuarios](#_heading=h.26in1rg) 8

[2.4. Restricciones](#_heading=h.lnxbz9) 9

[2.5. Suposiciones y Dependencias](#_heading=h.35nkun2) 9

[2.6. Requisitos Futuros](#_heading=h.1ksv4uv) 9

[**3. Requisitos Específicos**](#_heading=h.44sinio) **9**

[3.1 Requisitos comunes de las interfaces](#_heading=h.2jxsxqh) 9

[*3.1.1 Interfaces de usuario*](#_heading=h.z337ya) *9*

[*3.1.2 Interfaces de hardware*](#_heading=h.3j2qqm3) *10*

[*3.1.3 Interfaces de software*](#_heading=h.1y810tw) *10*

[3.2 Requisitos funcionales](#_heading=h.4i7ojhp) 10

[3.3 Requisitos no funcionales y de calidad](#_heading=h.2xcytpi) 11

[*3.3.1 Requisitos de rendimiento*](#_heading=h.1ci93xb) *11*

[*3.3.2 Seguridad*](#_heading=h.3whwml4) *11*

[*3.3.3 Fiabilidad*](#_heading=h.2bn6wsx) *11*

[*3.3.4 Disponibilidad*](#_heading=h.qsh70q) *11*

[*3.3.5 Mantenibilidad*](#_heading=h.3as4poj) *11*

[3.4 Otros Requisitos No funcionales](#_heading=h.1pxezwc) 11

[**4. Propuesta de Planificación**](#_heading=h.49x2ik5) **12**

[4.1 Descripción general acerca de la Planificación](#_heading=h.2p2csry) 12

[*4.1.2 Definición del Equipo de Trabajo*](#_heading=h.147n2zr) *12*

[*4.1.3 Definición de Actividades principales del Proyecto*](#_heading=h.3o7alnk) *12*

[*4.1.4 Diagrama EDT*](#_heading=h.23ckvvd) *13*

[4.1.5 Resumen Costos del Desarrollo del Proyecto](#_heading=h.hkw2a1oxscio) 13

[*4.1.6 Carta Gantt*](#_heading=h.32hioqz) *14*

[5. Anexos](#_heading=h.1hmsyys) 15

[*5.1 Minuta Kick off*](#_heading=h.41mghml) *15*

[*5.1 Acta de Proyecto*](#_heading=h.uvmjpgs1wj6a) *15*

[*5.2 Matriz Especificación de Requerimientos*](#_heading=h.vx1227) *15*

[*5.3 Diagrama de Casos de Uso General*](#_heading=h.3fwokq0) *15*

[*5.4 Planilla Casos de Uso Extendido*](#_heading=h.1v1yuxt) *15*

[*5.5 Prototipado de Software*](#_heading=h.4f1mdlm) *15*

[*5.6 Matriz EDT. Planilla Detallada Cálculo de Esfuerzo*](#_heading=h.2u6wntf) *15*

[*5.7 Planilla Carta Gantt*](#_heading=h.19c6y18) *16*

# Ficha del documento

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisión** | **Autor** | **Modificación** |
| 23/08/2025 | 1 | Felipe Lorca | Realización primera prueba |
|  | 2 |  |  |
|  | 3 |  |  |

Documento validado por las partes en fecha:

**Integrantes:**

|  |  |
| --- | --- |
| **Nombre Integrante del Equipo** | **Rol Definido** |
| Felipe Lorca | Tester QA |
| Almendra Pavez | Programador Frontend |
| Sebastián Muñoz | Arquitecto de Software |
| Michael Rubio | Programador Backend |
| Juan Paillan | Jefe de Proyecto |

# 1. Introducción

El proyecto **AirLink** surge como respuesta a una problemática recurrente en el transporte nacional e internacional: la falta de integración entre los vuelos aéreos y los buses interurbanos. Actualmente, los pasajeros enfrentan dificultades al coordinar sus traslados desde aeropuertos hacia ciudades o destinos finales debido a la escasa oferta, los altos costos y la poca confiabilidad del transporte terrestre.

La iniciativa busca desarrollar una plataforma digital que centralice en un solo sistema la compra de pasajes de avión y de bus, ofreciendo al usuario una experiencia más segura, confiable y eficiente. A través de esta solución, se espera optimizar tiempos, reducir costos y evitar abusos en los precios del transporte, mejorando de forma significativa la experiencia de viaje.

Este documento de **Especificación de Requisitos de Software (ERS)** tiene como objetivo detallar el alcance del sistema, los requisitos funcionales y no funcionales, así como las restricciones técnicas y consideraciones de calidad. La estructura del documento contempla secciones como: propósito, ámbito del sistema, definiciones, referencias y visión general del proyecto, garantizando una base sólida para el desarrollo y la gestión del mismo.

## 1.1. Propósito

El propósito de este documento es entregar información relevante al equipo desarrollador y a los interesados del proyecto **AirLink**. El ERS busca definir de manera clara los objetivos, alcances y restricciones del sistema a construir, sirviendo como base para el desarrollo, validación y posterior implementación.

AirLink está orientado a mejorar la experiencia de viaje de los usuarios mediante la integración en una sola plataforma de la compra de pasajes de avión y buses interurbanos, garantizando seguridad, eficiencia y confiabilidad en todo el proceso.

## 1.2. Ámbito del Sistema

El sistema se denomina **“AirLink”** y tendrá como objetivo principal centralizar la compra de pasajes de avión y de bus en un solo flujo de transacción. Su alcance comprende:

* Plataforma **web para usuarios viajeros**, que permitirá:  
  + Buscar y comprar pasajes de avión y bus de manera integrada.
  + Consultar itinerarios y disponibilidad en tiempo real.
  + Generar comprobantes de compra y reportes en formato digital (PDF/Excel).
* Plataforma **de escritorio administrativo** para aerolíneas y empresas de buses, que incluirá:
  + Gestión de convenios con empresas asociadas.
  + Control de ventas, usuarios y reportes de viajes.
  + Administración de cuentas y perfiles de usuario.
* **Integración de APIs** de aerolíneas y empresas de buses (como TurBus y otras).
* **Base de datos MySQL** para la administración de información centralizada.
* **Backend en Node.js** y **frontend en React/TypeScript** para garantizar escalabilidad, seguridad y buena experiencia de usuario.

El sistema **no contempla**:

* Gestión de vacaciones o licencias médicas.
* Administración de pagos de sueldos.
* Uso como aplicación móvil en esta primera fase (previsto como mejora futura).
* El sistema no tiene el horario respectivo de los trabajadores.

## 1.3. Definiciones, Acrónimos y Abreviaturas

BD= Base de datos.

SW= SOFTWARE.

RR. HH= Recursos humanos.

JP= jefe de Proyecto.

APC= Analista Programador.

DBA= Administrador BD.

QA= Calidad y Testing.

DG= Diseñador.

TD= Todo el equipo.

## 1.4. Referencias

Cabe destacar que destacar que estos son los documentos solicitados para la primera evaluación

### 1) Acta constitución proyecto

### 2) Plantilla de Requerimientos

### 3) Carta Gantt

### 4) EDT-Costos

### 5) Documento Caso uso extendido

### 6) Documento de avance de sprint (borrar si no usamos metodología ágil)

### 7) Mockups del sistema completo

### 8) Minuta de reuniones

## 

# 2. Descripción General

El proyecto **AirLink** tiene como objetivo desarrollar una plataforma digital que integre en un solo sistema la compra de pasajes de avión y de bus interurbano. El propósito es mejorar la experiencia de los pasajeros al permitirles planificar sus viajes de forma centralizada, optimizando costos, tiempos y disponibilidad.

El sistema contempla tanto una **plataforma web para usuarios finales**, como una **plataforma de escritorio administrativo** para las aerolíneas y empresas de buses asociadas, que facilitará la gestión de convenios, reportes y control de ventas.

## 2.1. Perspectiva del Producto

1. Construir un módulo de **búsqueda y compra de pasajes** de avión y bus en un solo flujo.
2. Construir un módulo de **gestión administrativa** para aerolíneas y empresas de buses.
3. Construir un módulo de **reportes de ventas y viajes** exportables a PDF/Excel.
4. Implementar la **integración con APIs** de aerolíneas y buses (ej. TurBus).
5. Desarrollar un **sistema de autenticación seguro** para usuarios y administradores.
6. Gestionar **itinerarios y disponibilidad en tiempo real**.
7. Construir un módulo de **gestión de perfiles de usuario** (viajero, administrador, soporte).
8. Habilitar la **administración de convenios** entre empresas asociadas.
9. Desarrollar un **backend escalable** en Node.js y una **interfaz web en React/TypeScript**.
10. Gestionar la información en una **base de datos MySQL** centralizada.

## 2.2. Funciones del Producto

1. **Búsqueda y compra integrada de pasajes** de avión y bus en un solo proceso.
2. **Consulta de itinerarios y disponibilidad en tiempo real** para vuelos y buses.
3. **Gestión de convenios** con aerolíneas y empresas de transporte asociadas.
4. **Gestión de usuarios y perfiles** (viajeros, administradores, soporte).
5. **Autenticación y seguridad de acceso** para usuarios y administradores.
6. **Generación de reportes** de ventas, viajes y convenios exportables en PDF/Excel.
7. **Administración de transacciones** y control de pagos en la plataforma.
8. **Módulo administrativo** para aerolíneas y buses: gestión de rutas, horarios y disponibilidad.
9. **Gestión de credenciales y cuentas** para nuevos usuarios.
10. **Soporte y asistencia virtual** para consultas de los usuarios.

## 2.3. Características de los Usuarios

* **Usuario Viajero:**
  + Público general (nacional e internacional).
  + Nivel básico en el uso de sistemas web y móviles.
  + Requiere una interfaz intuitiva, con procesos de compra claros y rápidos.
* **Administrador de Aerolínea/Empresa de Buses:**
  + Personal de las empresas asociadas.
  + Formación técnica o profesional en administración, logística o sistemas.
  + Nivel intermedio en el uso de plataformas de gestión.
  + Encargados de cargar itinerarios, gestionar disponibilidad y revisar reportes de ventas.
* **Administrador del Sistema (Soporte Técnico):**
  + Formación superior en informática o ingeniería en sistemas.
  + Nivel avanzado en sistemas y bases de datos.
  + Encargado de garantizar el funcionamiento del backend, seguridad y actualizaciones.
* **Director del Proyecto / Supervisor:**
  + Nivel directivo, con formación superior en administración, informática o gestión de proyectos.
  + No requiere conocimientos técnicos profundos, pero sí acceso a reportes estratégicos del sistema.

## 

## 2.4. Restricciones

* La plataforma deberá estar disponible inicialmente como **aplicación web**, no contemplando versión de escritorio en esta primera fase.
* El sistema dependerá de la **integración con APIs externas** de aerolíneas y empresas de buses, lo que puede limitar la disponibilidad de información en tiempo real.
* Se utilizará una **base de datos MySQL** como estándar definido para el proyecto.
* El backend deberá desarrollarse en **Node.js** y el frontend en **React con TypeScript**, de acuerdo con la arquitectura establecida.
* La infraestructura se desplegará en **servicios cloud (AWS o Azure)**, por lo que dependerá de la conectividad a internet y de los costos de la nube.
* El sistema debe ajustarse al **presupuesto inicial definido en la planificación**.
* La seguridad será un requisito crítico: todo acceso debe estar protegido mediante autenticación robusta y cifrado de datos.

## 2.5. Suposiciones y Dependencias

1. Se asume que las **aerolíneas y empresas de buses interurbanos** entregarán acceso estable a sus **APIs** para la integración de itinerarios y disponibilidad.
2. El correcto funcionamiento del sistema depende de la **conectividad a internet** y la **disponibilidad de los servicios cloud** (AWS, Azure o GCP).
3. Se asume que los **usuarios finales** contarán con dispositivos compatibles (computadores o móviles con navegadores modernos).
4. La gestión de ventas y convenios dependerá de la **colaboración activa de las empresas asociadas**.
5. El proyecto depende de que el **equipo de desarrollo** cumpla con los hitos planificados (levantamiento de requisitos, diseño, desarrollo, pruebas y entrega final).
6. Se asume que el **presupuesto estimado** será aprobado y no sufrirá recortes que afecten el desarrollo

## 2.6. Requisitos Futuros

Esta subsección contempla mejoras potenciales que podrán analizarse e implementarse en fases posteriores del proyecto **AirLink**:

1. Extender la plataforma con una **aplicación de escritorio nativa** para Windows, Linux y Mac.
2. Incorporar un **módulo de inteligencia artificial** para recomendar rutas y combinaciones de viaje personalizadas según historial de usuario.
3. Integrar **pasarelas de pago internacionales** (PayPal, Stripe, etc.) para ampliar el alcance a usuarios extranjeros.
4. Implementar un **asistente virtual 24/7** para soporte automatizado de los viajeros.
5. Ofrecer un sistema de **programas de fidelización** y beneficios para usuarios frecuentes.
6. Integrar **servicios adicionales** como reservas de hoteles y arriendo de vehículos en la misma plataforma.

# 3. Requisitos Específicos

## 3.1 Requisitos comunes de las interfaces

Descripción detallada de las entradas y salidas del sistema, diferenciando **Web** y **Aplicación de Escritorio**.

**Pagina Web**

* UI responsive (desktop/tablet/móvil) con React + TypeScript; navegación SPA.
* Tipografía legible (p. ej., Inter/Roboto 14–16 px) y contraste AA (WCAG 2.1).
* Componentes clave: buscador integrado (vuelos+buses), resultados con filtros, carrito/checkout, detalle de itinerario, historial de compras, centro de ayuda.
* Estados y feedback: loading, vacíos, validación en tiempo real, mensajes de error claros.
* Accesibilidad: navegación por teclado, ARIA en controles, foco visible, textos alternativos.
* Idiomas iniciales: ES (extensible a EN).
* Seguridad UX: flujos de autenticación/recuperación de contraseña, doble confirmación en compra.

**Aplicación de Escritorio (Panel Administrativo)**

* Enfoque de productividad (teclado/atajos, tablas, filtros y edición en línea).
* Módulos: gestión de convenios (aerolíneas/buses), rutas/horarios/disponibilidad, precios/tarifas, usuarios/perfiles, reportes (ventas, ocupación, KPIs).
* Exportación/impresión: PDF/Excel desde vistas de datos y reportes.
* Auditoría visible (quién/qué/cuándo) en acciones críticas.
* Acceso por rol: Administrador, Operador Aerolínea, Operador Bus, Soporte.

### 3.1.2 Interfaces de hardware

**Cliente Pagina Web (usuario viajero)**

* Dispositivo con navegador moderno (ver 3.1.3), pantalla ≥ 1280×720 **recomendado**.
* Conectividad a Internet estable (≥10 Mbps **recomendado** para tiempos de respuesta óptimos).

**Cliente Escritorio (operadores)**

* SO: Windows 10/11 (64-bit).
* CPU: Intel i3 de 8ª gen / Ryzen 3 equivalente o superior.
* RAM: 8 GB mínimo (16 GB recomendado si se gestionan grandes listados).
* Almacenamiento: 2 GB libres para instalación + caché/reportes.
* Red: Ethernet o Wi-Fi estable (latencia <100 ms a servicios cloud).

**Servidor / Backend (referencial, cloud)**

* Entorno Node.js LTS.
* 2 vCPU, 4–8 GB RAM para entorno de prueba; autoescalado para producción según demanda.
* Almacenamiento para BD MySQL administrada (backup diario y multi-AZ).

### 

### 

### 

### 

### 3.1.3 Interfaces de software

**Pagina Web**

* Frontend: React 18 + TypeScript, Router, gestor de estado (p. ej., React Query/Redux) y librería de componentes accesibles.
* Navegadores soportados: Chrome, Edge, Firefox y Safari versiones soportadas (últimas 2 releases).
* Integraciones: APIs REST/JSON para búsqueda, disponibilidad, precios y compra.

**Aplicación de Escritorio (Aun por ver)**

* Stack sugerido: Electron + React + TypeScript (packaging MSI/EXE).
* Integración con servicios: mismas APIs REST que Web; manejo de credenciales por entorno.
* Impresión/Exportación: generación de PDF/Excel (biblioteca compatible con Node/TS).

**Backend y terceros (comunes)**

* Backend: Node.js (LTS), framework HTTP (Express/Nest), autenticación JWT/OAuth2.
* Base de datos: **MySQL** administrado (conexión TLS, migraciones versionadas).
* Infraestructura: servicios en la nube (AWS/Azure) con CI/CD, monitoreo, logs centralizados.
* Seguridad: HTTPS obligatorio, CSP, protección CSRF/XSS, cifrado en tránsito y en reposo, gestión de secretos (vault).
* Integraciones externas: **APIs de aerolíneas y buses** (autenticación por API key/OAuth según proveedor).

## 3.2 Requisitos funcionales

Los requisitos funcionales deben estar enumerados (diagrama caso uso) y pueden ser divididos en sub-secciones.

3.2.1 Registrar las contrataciones de empleados

3.2.2 Registrar a los empleados nuevos y antiguos

3.2.3 Registrar la oferta de las capacitaciones

3.2.4 Inscribir a los empleados para capacitaciones

3.2.5 Registrar los bonos

3.2.6 Obtener los resultados de las evaluaciones

3.2.7 Calcular la bonificación

3.2.8 Entregar los reportes

3.2.9 Obtener un listado de las capacitaciones

3.2. Construir un mantenedor de perfiles

3.2.5 Autentificarse en el sistema

## 3.3 Requisitos no funcionales y de calidad

### Apartado Web

### 3.3.1 Requisitos de rendimiento

1.0 El sistema debe responder las búsquedas en **≤ 3 s (P95)**.

1.1 El sistema debe completar el **checkout/compra** en **≤ 2 s (P95)** sin 3-D Secure.

1.2 El sistema debe alcanzar **LCP ≤ 2,5 s** en al menos el **75%** de las sesiones.

1.3 El sistema debe soportar **≥ 300 usuarios concurrentes** con autoescalado en cloud.

1.4 El sistema debe paginar y cargar de forma diferida listados y reportes.

### 3.3.2 Seguridad

1.0 El sistema debe cifrar todo el tráfico mediante **HTTPS (TLS 1.2+)**.

1.1 El sistema debe implementar **autenticación JWT/OAuth2** y **control de acceso por roles (RBAC)**.

1.2 El sistema debe protegerse contra **OWASP Top 10** (XSS, CSRF, SQLi, SSRF, etc.).

1.3 El sistema debe cifrar datos sensibles **en tránsito y en reposo** usando claves gestionadas.

1.4 El sistema debe aplicar **3-D Secure** y validaciones antifraude para pagos.

1.5 El sistema debe exigir contraseñas robustas y bloqueo tras intentos fallidos.

### 3.3.3 Fiabilidad

1.0 El sistema debe manejar fallas de APIs externas con **reintentos** y **circuit breakers**.

1.1 El sistema debe garantizar **idempotencia** en operaciones críticas (órdenes, vouchers).

1.2 El sistema debe mantener **integridad transaccional (ACID)** en MySQL.

1.3 El sistema debe procesar tareas de larga duración de forma **asíncrona**.

### 3.3.4 Disponibilidad

1.0 El sistema debe cumplir un **SLA ≥ 99,5%** mensual en la web pública.

1.1 El sistema debe realizar **backups diarios** con **RPO ≤ 24 h** y **RTO ≤ 4 h**.

1.2 El sistema debe desplegarse en **multi-AZ** con health checks y rollback automático.

1.3 El sistema debe estar disponible en todo momento

### 3.3.5 Mantenibilidad

1.0 El sistema debe desarrollarse en **TypeScript/Node.js** con reglas **ESLint/Prettier**.

1.1 El sistema debe mantener **cobertura de tests ≥ 70%** (unitarios e2e donde aplique).

1.2 El sistema debe disponer de **CI/CD** con ambientes Dev/Stg/Prod y revisión por PR.

1.3 El sistema debe versionar y documentar sus APIs con **OpenAPI**.

### 3.3.6 Usabilidad y Accesibilidad

1.0 El sistema debe cumplir WCAG 2.1 AA (teclado, ARIA, foco visible, contraste).

1.1 El sistema debe ser responsive y proveer feedback claro en cargas/errores.

1.2 El sistema debe soportar i18n iniciando con ES y extensible a EN.

### 3.3.7 Observabilidad y auditoría

1.0 El sistema debe generar logs estructurados y trazas distribuidas.

1.1 El sistema debe exponer métricas (latencia, tasa de error, throughput) con alertas.

1.2 El sistema debe auditar acciones críticas (login, compras, cambios administrativos).

## 3.4 Otros Requisitos No funcionales

1.0 El sistema debe implementar el frontend con **React 18 + TypeScript**.

1.1 El sistema debe implementar el backend con **Node.js LTS** (Express/Nest).

1.2 El sistema debe utilizar **MySQL** administrado como base de datos.

1.3 El sistema debe desplegarse en **AWS/Azure** con **autoescalado** y **CDN** para estáticos.

1.4 El sistema debe integrarse con **APIs de aerolíneas y buses** mediante OAuth/API-Key.

1.5 El sistema debe exportar **PDF/Excel/CSV** en sus reportes.

1.6 El sistema debe ser compatible con **Chrome/Edge/Firefox/Safari** (últimas 2 versiones).

1.7 El sistema debe cumplir la normativa de **protección de datos personales** aplicable (p. ej., Ley 19.628) y gestionar consentimiento de cookies.

1.8 El sistema debe aplicar **rate limiting** y **WAF** frente a abusos/ataques.

1.9 El sistema debe respetar la **identidad visual** del proyecto de forma consistente.

1.10 El sistema debe compartir componentes y autenticación con el **panel de escritorio** cuando este se implemente (p. ej., **Electron + React**).

### Apartado Escritorio

### 3.3.1 Rendimiento

1.0 El sistema debe iniciar sesión en menos de **3 segundos**.  
 1.1 El sistema debe ejecutar búsquedas avanzadas y consultas de historial en menos de **5 segundos**.  
 1.2 El sistema debe generar reportes en PDF/Excel de forma eficiente sin bloquear la interfaz.  
 1.3 El sistema debe optimizar el uso de memoria y CPU en equipos con recursos limitados.

### 3.3.2 Seguridad

1.0 El sistema debe garantizar la **confidencialidad** de los datos mediante cifrado en almacenamiento y transmisión.  
 1.1 El sistema debe implementar control de acceso basado en **roles y permisos** (usuario, administrador, instructor).  
 1.2 El sistema debe validar archivos cargados antes de almacenarlos, evitando inyecciones y malware.  
 1.3 El sistema debe proteger los **registros de auditoría y logs**, impidiendo su manipulación.  
 1.4 El sistema debe integrar autenticación segura con recuperación de contraseña vía correo o pregunta de seguridad.

### 3.3.3 Fiabilidad

1.0 El sistema debe tolerar fallas y recuperarse automáticamente tras un error o caída.  
 1.1 El sistema debe garantizar consistencia en operaciones críticas (registro de títulos, pagos, auditorías).  
 1.2 El sistema debe registrar todas las acciones críticas en logs de auditoría.  
 1.3 El sistema debe soportar **respaldo automático** y **recuperación desde respaldo** sin pérdida de datos.

### 3.3.4 Disponibilidad

1.0 El sistema debe estar disponible los **365 días** del año en el entorno de escritorio.  
 1.1 El sistema debe permitir trabajar **offline** en ciertas operaciones con sincronización posterior.  
 1.2 El sistema debe detectar y cerrar sesiones duplicadas activas.

### 3.3.5 Mantenibilidad

1.0 El sistema debe estar desarrollado de manera modular, permitiendo agregar o modificar mantenedores fácilmente.  
 1.1 El sistema debe ser documentado técnica y funcionalmente para facilitar soporte.  
 1.2 El sistema debe permitir instalación y actualizaciones automáticas.

### 3.3.6 Usabilidad y Accesibilidad

1.0 El sistema debe presentar **colores amigables** y tipografía legible, incluyendo adultos mayores.  
 1.1 El sistema debe ser intuitivo y de fácil navegación.  
 1.2 El sistema debe incorporar **notificaciones en escritorio** y alertas de errores o recordatorios.  
 1.3 El sistema debe contar con **ayuda contextual** y un módulo de soporte integrado.  
 1.4 El sistema debe adaptarse a diferentes resoluciones y soportar PC, notebooks y tablets con Windows.

## 3.4 Otros Requisitos No Funcionales *(Escritorio)*

1.0 El sistema debe ser compatible con **Windows 10/11** y preferentemente con Linux.  
 1.1 El sistema debe incluir instalador con **asistente paso a paso**.  
 1.2 El sistema debe consumir recursos de forma eficiente, optimizando memoria y CPU.  
 1.3 El sistema debe interoperar con sistemas externos (ej. **BancoEstado**) para validación de pagos.  
 1.4 El sistema debe permitir **importar y exportar datos** en formatos CSV, Excel y PDF.  
 1.5 El sistema debe contar con un **buscador avanzado** con filtros por nombre, fecha y estado.  
 1.6 El sistema debe mantener consistencia en la interfaz visual con la aplicación web (identidad gráfica unificada).  
 1.7 El sistema debe escalar en número de usuarios administradores sin pérdida de rendimiento.

# 4. Propuesta de Planificación

## 4.1 Descripción general acerca de la Planificación

El proyecto **AirLink** consiste en el desarrollo de una plataforma digital innovadora que integrará en un mismo sistema la compra de pasajes de **avión y buses interurbanos**, permitiendo a los usuarios planificar sus viajes de forma centralizada, segura y eficiente.

La solución contempla dos componentes principales:

* Una **plataforma web orientada al viajero**, donde podrá buscar vuelos y buses, filtrar opciones, reservar en un único flujo y obtener comprobantes digitales de su compra.
* Una **aplicación de escritorio administrativa**, destinada a las aerolíneas y empresas de transporte terrestre, que permitirá gestionar convenios, administrar horarios y tarifas, controlar ventas y generar reportes en diferentes formatos.

La arquitectura tecnológica estará basada en **React y TypeScript** para el frontend web, **Node.js** para el backend, **MySQL** como base de datos y despliegue en servicios **cloud (AWS/Azure)** para asegurar escalabilidad y disponibilidad.

El proyecto se ejecutará en un plazo estimado de tres meses (septiembre a diciembre de 2025), siguiendo una planificación estructurada en fases de levantamiento de requisitos, diseño, desarrollo, pruebas y entrega final. El cumplimiento riguroso de los hitos permitirá garantizar la calidad del sistema y su disponibilidad dentro del plazo comprometido.

### 4.1.2 Definición del Equipo de Trabajo

|  |  |
| --- | --- |
| ROL ACTOR | NOMBRE ACTOR |
| Jefe de Proyecto | Juan Paillán |
| Programador Frontend | Almendra Pavez |
| Tester QA | Felipe Lorca |
| Arquitecto de Software | Sebastián Muñoz |
| Programador Backend | Michael Rubio |

### 4.1.3 Definición de Actividades principales del Proyecto

1. Fase de Planificación Inicial.
2. Fase de Análisis y Diseño.
3. Fase de Construcción.
4. Fase de Implementación y Cierre.

### 4.1.4 Diagrama EDT



### 

### 

### 4.1.5 Resumen Costos del Desarrollo del Proyecto

Costos por Rol

|  |  |
| --- | --- |
| COSTO HH POR ROL | |
| Jefe de Proyecto | $ 2.187.738 |
| Programador Frontend | $ 266.680 |
| Tester QA | $ 240.000 |
| Arquitecto de Software | $ 1.765.920 |
| Programador Backend | $ 573.362 |
| **TOTAL HH** | **$ 5.033.700** |

En la anterior tabla se refleja cómo se calculan los **costos por horas-hombre (HH)** en un proyecto de software

Costos por Fase

|  |  |
| --- | --- |
| COSTO POR FASE | |
| Fase de Planificación Inicial | $ 841.228 |
| Fase de Análisis y Diseño | $ 2.160.224 |
| Fase de Construcción | $ 821.566 |
| Fase de Implementación y Cierre | $ 1.210.682 |
| **TOTAL HH FASES** | **$ 5.033.700** |

### 4.1.6 Carta Gantt

Fechas importantes:

* Hito 1 (Entrega 1): 10/Septiembre/2025
* Hito 2 (Entrega 2): 8/Octubre/2025
* Hito 3 (Entrega 3): 19/Noviembre/2025

## 5. Anexos

### 5.1 Acta Constitución



### 5.1 Plantilla Requerimientos



### 5.2 Matriz o Plantilla EDT



### 5.3 Diagrama EDT



### 5.4 Documento Casos de Uso Extendido



### 5.5 Documento de avance de sprint



### 5.6 Minuta de reuniones



### 5.7 Documentos Mockups



### 5.8 Listado de Riesgos



### 5.9 Matriz RACI



### 5.10 Carta Gantt

