Diseño de Arquitectura del Sistema - TaskManager

**Fecha:**

**Autor:**

**Versión:** 1.0

# 1. Introducción

Este documento describe de manera detallada la arquitectura del sistema para la aplicación web **Panorama**, cuyo propósito es optimizar la gestión de turistas, mejorar la comunicación entre la empresa y los visitantes, y contribuir al incremento de la satisfacción y fidelización de los clientes nacionales e internacionales.

Se define la estructura general del sistema, los componentes que lo conforman, sus interacciones y las tecnologías empleadas.

# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

El sistema adopta una **arquitectura modular híbrida (monolito modular en 3 capas)**, fundamentada en el patrón **cliente-servidor**. Esta organización busca un equilibrio entre simplicidad de implementación y escalabilidad futura, permitiendo que el sistema pueda evolucionar sin perder orden ni mantenibilidad.

Las **tres capas principales** que conforman la arquitectura son:

* **Capa de Presentación (Frontend):** Implementada como una aplicación web responsiva desarrollada en **React** . Es el punto de interacción directo con los usuarios (turistas y personal administrativo). Proporciona interfaces intuitivas para el acceso a itinerarios, reportes y paneles de administración.
* **Capa de Lógica de Negocio (Backend):** Desarrollada en **Node.js con Express**, organizada en **módulos independientes** (auth, turistas, itinerarios, reportes, notificaciones, administración). Esta capa gestiona las reglas de negocio, la validación de datos, la seguridad, el control de sesiones y la comunicación entre la interfaz de usuario y la base de datos.
* **Capa de Persistencia de Datos (Base de Datos):** Basada en **MySQL** como sistema gestor relacional. Se encarga del almacenamiento estructurado, recuperación eficiente de información y mantenimiento de integridad entre entidades como turistas, itinerarios, hoteles, visitas y usuarios.

## 2.2. Diagrama General de Arquitectura:

Diagrama

El contenido generado por IA puede ser incorrecto.

# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

**Framework:** React.js

**Estilo:** Tailwind CSS, CSS3

**Responsabilidades:**

* Renderizar interfaces web responsivas e intuitivas para turistas y personal administrativo.
* Capturar entradas y acciones de los usuarios.
* Mostrar mensajes, errores y validaciones de manera clara.
* Comunicarse con la capa de lógica de negocio (backend) mediante fetch/axios para obtener o enviar datos.
* Visualizar itinerarios, reportes y paneles de administración.

## 3.2. Backend (Lógica de negocio)

**Lenguaje y Framework:** Node.js

**Base de datos:** MySQL (con ORM como Sequelize o Prisma)

**Responsabilidades principales:**

* Exponer **API RESTful** para que el frontend (React) pueda interactuar con la aplicación.
* Validar datos de entrada y gestionar errores, enviando **mensajes claros** al frontend.
* Aplicar **reglas de negocio**:  
  + Validación de campos obligatorios al registrar turistas.
  + Control de permisos según rol (administrativo, personal, gerente, turista).
  + Auditoría de cambios en turistas e itinerarios.
  + Generación de reportes y estadísticas básicas.
  + Notificaciones internas sobre cambios en itinerarios o datos de turistas.
* **Control de acceso** mediante JWT (JSON Web Tokens) para proteger rutas sensibles.
* Coordinación con la **base de datos MySQL** para operaciones CRUD y consultas.  
  Preparación para integración futura con microservicios de IA o chatbots de asistencia al turista.

### 3.2.1. Módulos del Backend

El backend está organizado en módulos específicos para mantener la modularidad, escalabilidad y facilidad de mantenimiento. Cada módulo tiene responsabilidades concretas dentro del sistema:

**1. Gestión de Base de Datos y Administración**

* **dbController.js**: Creación de base de datos, tablas y relaciones (HU 1.1, 1.2, 1.3)
* **backupController.js**: Configuración y restauración de respaldos automáticos (HU 1.4, 1.5, 1.6)
* **pilotTestController.js**: Pruebas piloto con usuarios reducidos y registro de incidencias (HU 1.7, 1.8, 1.9)

**2. Gestión de Turistas**

* **turistaController.js**: Registrar, editar, eliminar y consultar turistas; validaciones, trazabilidad y autorización (HU 2.1 – 2.17)
* **exportController.js**: Exportar listados y reportes de turistas en Excel o PDF (HU 2.12, 2.14, 2.15)

**3. Autenticación y Seguridad**

* **authController.js**: Login, recuperación de contraseña, gestión de sesiones y roles (HU 3.1 – 3.6)
* **roleController.js**: Asignación y modificación de permisos de usuarios (HU 3.7 – 3.9)
* **sessionController.js**: Cierre automático de sesiones, alertas y trazabilidad de accesos (HU 3.10 – 3.14)

**4. Gestión de Itinerarios**

* **itinerarioController.js**: Crear, editar, eliminar y consultar itinerarios (HU 4.1 – 4.6)
* **pdfController.js**: Generar PDF de itinerarios con diseño y logos (HU 4.7 – 4.9)
* **excelItinerarioController.js**: Exportar itinerarios y reportes en Excel (HU 4.10 – 4.12)
* **notificacionController.js**: Notificaciones y alertas sobre cambios en itinerarios, registro y trazabilidad (HU 4.13 – 4.18)

**5. Reportes, Estadísticas y Personalización**

* **reportController.js**: Generación y almacenamiento de reportes internos y gráficos (HU 5.1 – 5.6)
* **idiomaController.js**: Configuración y cambio de idioma en tiempo real (HU 5.7 – 5.9)
* **monedaController.js**: Gestión de precios en diferentes monedas y tasas de cambio (HU 5.10 – 5.12)

## 3.3. Base de Datos (Persistencia)

* Sistema Gestor: MYSQL
* ORM: Sequelize
* Esquema de Datos:

| **Tabla** | **Campos** | **Descripción** |
| --- | --- | --- |
| **usuarios** | id\_usuario, nombre\_usuario, email, password, rol, created\_at, updated\_at | Credenciales y roles de los usuarios del sistema |
| **turistas** | id\_turista, id\_usuario, nombre, apellido, dni, pasaporte, nacionalidad, fecha\_nacimiento, genero | Información personal de los turistas/clientes |
| **grupos** | id\_grupo, nombre, descripcion | Organiza grupos de turistas |
| **itinerarios** | id\_itinerario, id\_grupo, fecha\_inicio, fecha\_fin, estado\_presupuesto\_id, created\_at, updated\_at | Almacena itinerarios con fechas y estado |
| **itinerario\_turistas** | id\_itinerario, id\_turista | Relaciona turistas con itinerarios (N:M) |
| **programas** | id\_programa, nombre, descripcion, tipo, duradon, costo | Catálogo de programas y actividades turísticas |
| **itinerario\_programas** | id\_itinerario\_programa, id\_itinerario, id\_programa, fecha, hora\_inicio, hora\_fin | Relaciona itinerarios con programas y horarios |
| **detalle\_machu\_itinerario** | id\_itinerario\_programa, empresa\_tren, horario\_tren\_ida, horario\_tren\_reto, nombre\_gula, ruta, tiempo\_visita | Detalles específicos para itinerarios de Machu Picchu |
| **transportes** | id\_transporte, empresa, tipo, capacidad, contacto | Catálogo de transportes disponibles |

# 4. Integraciones Externas (Opcionales)

| **Integración** | **Prioridad** | **Descripción** | **Tecnología / Servicio Sugerido** |
| --- | --- | --- | --- |
| Pasarelas de pago online | Media | Aunque no se cobrará ahora, permite preparar futuras fases de reservas y pagos | Stripe, PayPal, Culqi, MercadoPago |
| Conversión de moneda en tiempo real | Alta | Mostrar costos de hoteles, transportes y actividades en USD, Soles o Euros | API de tipo de cambio (Open Exchange Rates, Fixer.io) |
| Email transaccional y notificaciones | Alta | Envío de alertas de cambios en itinerarios, recuperación de contraseña y recordatorios | SendGrid, Mailgun |
| WhatsApp Business API | Media | Notificaciones rápidas a turistas sobre itinerarios y cambios | WhatsApp Business API |
| PDFs personalizados | Alta | Generar itinerarios descargables con diseño claro y logo de la empresa | jsPDF, PDFKit, Puppeteer |
| Exportación a Excel / CSV | Alta | Exportar reportes de turistas e itinerarios para análisis externo | SheetJS (xlsx) |
| Traducción / Multilenguaje | Media | Permitir uso en español e inglés para turistas internacionales | i18next, Google Translate API, DeepL API |
| Mapas y rutas | Alta | Mostrar ubicación de hoteles, transporte y actividades en itinerarios | Google Maps API, OpenStreetMap |
| Analítica y dashboards | Media | Monitoreo del uso de la aplicación, estadísticas de turistas e itinerarios | Google Analytics, Matomo, Metabase, Grafana |
| Backup en la nube | Media | Respaldo automático de base de datos para prevenir pérdida de información | AWS S3, Google Cloud Storage, scripts de backup automático |
| Autenticación externa | Baja | Permitir login seguro mediante cuentas externas (opcional para futuras fases) | OAuth 2.0, OpenID Connect (Google, Facebook) |
| Monitoreo de seguridad | Media | Registrar errores, auditoría de accesos y acciones críticas | Sentry, LogRocket |

# 5. Seguridad

* **Autenticación y roles:** Todos los usuarios ingresan con usuario y contraseña; se controlan permisos según rol (administrador, personal, turista).
* **Protección de datos:** Información personal cifrada y comunicación segura vía HTTPS; validación de datos para evitar ataques o bcrypt.
* **Sesiones:** Cierre automático por inactividad y control de accesos duplicados.
* **Respaldo:** Copias automáticas de la base de datos y restauración ante fallos; notificaciones de errores en respaldos.
* **Auditoría:** Registro de cambios y acciones del personal administrativo para trazabilidad y transparencia.
* **Documentos y notificaciones:** PDFs, reportes y alertas solo accesibles por usuarios autorizados.

# 6. Escalabilidad y Despliegue

Posibles problemas que pueden surgir durante el desarrollo.

* **Frontend:** Desplegado en **Vercel**, fácil de actualizar y con planes gratuitos, ideal para proyectos React.
* **Backend:** Deploy en **Render**, compatible con Node.js y MySQL, estable y gratuito para estudiantes.
* **Base de Datos:** MySQL en **Render**, con respaldo automático y fácil integración con el backend.
* **Escalabilidad:** La arquitectura modular permite escalar horizontalmente el backend y separar la base de datos en instancias dedicadas si aumenta el número de usuarios.
* **Costo y comodidad:** Prioriza soluciones gratuitas o con planes para estudiantes, evitando infraestructura compleja o cara.

# 7. Conclusiones

* Permite implementar **funcionalidades básicas primero** (turistas, itinerarios, reportes) y añadir **extensiones futuras** (reservas, pagos, app móvil).
* Cumple con **buenas prácticas de desarrollo moderno**, incluyendo seguridad, modularidad y escalabilidad.