**📋 RESUMEN GENERAL DEL PROYECTO PLAYMI  
Entretenimiento que viaja contigo**

**🎯 DESCRIPCIÓN DEL PROYECTO:**

**PLAYMI** es una plataforma web integral para entretenimiento en transporte público que permite a empresas de buses ofrecer contenido multimedia (películas, música, juegos) a sus pasajeros a través de Raspberry Pi instalados en cada vehículo.

**🔄 FLUJO GENERAL DEL SISTEMA:**

**1. ADMINISTRACIÓN (Panel Web):**

* Los administradores gestionan empresas, contenido y configuraciones desde un panel web profesional
* Se generan paquetes personalizados de contenido para cada empresa
* Se crean códigos QR específicos para cada bus

**2. INSTALACIÓN EN BUSES:**

* Cada bus tiene un Raspberry Pi que emite su propia red WiFi
* El Pi contiene todo el contenido multimedia y el portal web localmente
* No requiere conexión a internet para funcionar (contenido offline)

**3. EXPERIENCIA DEL PASAJERO:**

* Pasajero escanea QR code pegado en el bus
* Se conecta automáticamente al WiFi del bus
* Accede a portal tipo Netflix desde su celular/tablet
* Consume películas, música y juegos durante el viaje

**4. SINCRONIZACIÓN:**

* Los Pi se sincronizan automáticamente con el servidor central cuando hay WiFi disponible
* Descargan nuevo contenido y configuraciones
* Reportan estadísticas de uso

**🏗️ ARQUITECTURA TÉCNICA:**

**SERVIDOR CENTRAL:**

* **Backend:** PHP 8+ con patrón MVC
* **Frontend Admin:** AdminLTE 3 + Bootstrap 4
* **Base de datos:** MySQL 8+
* **Servidor web:** Apache/Nginx

**RASPBERRY PI (En cada bus):**

* **SO:** Raspberry Pi OS (Debian)
* **WiFi AP:** hostapd + dnsmasq
* **Servidor web:** Apache + PHP
* **Portal:** HTML5/CSS3/JavaScript vanilla
* **Contenido:** Almacenamiento local

**FRONTEND PASAJEROS:**

* **Diseño:** Estilo Netflix responsive
* **Tecnología:** HTML5, CSS3, JavaScript vanilla
* **Video:** HTML5 Video API
* **Audio:** Web Audio API
* **Juegos:** HTML5 Canvas/WebGL en iframe sandbox

**📚 MÓDULOS PRINCIPALES:**

**1. PANEL ADMINISTRATIVO (AdminLTE 3):**

* **Dashboard:** Métricas y estadísticas en tiempo real
* **Gestión de Empresas:** CRUD completo con branding personalizado
* **Gestión de Contenido:** Upload y organización de multimedia
* **Generación de Paquetes:** Crear paquetes personalizados para Pi
* **Sistema QR:** Generar códigos QR por bus
* **Monitoreo Pi:** Estado en tiempo real de toda la flota

**2. PORTAL PASAJEROS (Estilo Netflix):**

* **Página Principal:** Hero banner + carruseles de contenido
* **Catálogo Películas:** Grid responsive con filtros
* **Reproductor Música:** Estilo Spotify con playlists
* **Catálogo Juegos:** Launcher de juegos HTML5
* **Búsqueda Global:** Búsqueda en tiempo real

**3. SISTEMA DE JUEGOS:**

* **Juegos HTML5:** Subida de archivos ZIP con index.html
* **Sandbox Seguro:** Iframe con restricciones de seguridad
* **Leaderboard:** Sistema de puntuaciones
* **Categorías:** Puzzle, Arcade, Estrategia, Cartas

**4. SINCRONIZACIÓN:**

* **APIs REST:** Comunicación servidor-Pi
* **Descarga Delta:** Solo cambios incrementales
* **Health Monitoring:** Estado de cada Pi
* **Auto-recovery:** Recuperación automática de errores

**🎨 TECNOLOGÍAS UTILIZADAS:**

**BACKEND:**

* **PHP 8+:** Lenguaje principal del servidor
* **MySQL 8+:** Base de datos relacional
* **Apache/Nginx:** Servidor web
* **Las librerías que se descarguen de forma manual para integrarlas no usar composer**

**FRONTEND ADMIN (AdminLTE 3):**

* **AdminLTE 3.2.0:** Framework de administración
* **Bootstrap 4:** Framework CSS responsive
* **jQuery 3.6+:** JavaScript para interactividad
* **DataTables:** Tablas con filtros y paginación
* **Chart.js:** Gráficos y estadísticas
* **SweetAlert2:** Modales y confirmaciones elegantes
* **Toastr:** Notificaciones toast
* **FontAwesome:** Iconografía completa

**FRONTEND PASAJEROS:**

* **HTML5:** Estructura semántica
* **CSS3:** Animations, flexbox, grid
* **JavaScript Vanilla:** Sin frameworks para performance
* **Web APIs:** Video, Audio, Canvas, LocalStorage
* **Responsive Design:** Mobile-first approach

**RASPBERRY PI:**

* **Raspberry Pi OS:** Sistema operativo
* **hostapd:** WiFi Access Point
* **dnsmasq:** DHCP y DNS server
* **Apache + PHP:** Stack web local
* **Bash Scripts:** Automatización y mantenimiento   
  inicio automatico del proyecto cada vez que tenga corriente in intervención manual

**📁 ESTRUCTURA DE ARCHIVOS:**

PLAYMI/

├── 📁 admin/ # Panel administrativo

│ ├── 📁 assets/ # AdminLTE3 + personalizaciones

│ │ ├── 📁 css/ # AdminLTE + custom CSS

│ │ ├── 📁 js/ # AdminLTE + custom JS

│ │ ├── 📁 plugins/ # Bootstrap, DataTables, etc.

│ │ └── 📁 images/ # Logos e imágenes

│ ├── 📁 config/ # Configuraciones PHP

│ ├── 📁 controllers/ # Lógica de negocio MVC

│ ├── 📁 models/ # Acceso a datos MVC

│ ├── 📁 views/ # Interfaces de usuario

│ │ ├── 📁 layouts/ # Templates base (header, sidebar, footer)

│ │ ├── 📁 companies/ # CRUD empresas

│ │ ├── 📁 content/ # Gestión contenido

│ │ ├── 📁 packages/ # Generación paquetes

│ │ └── 📁 qr-system/ # Sistema QR codes

│ └── 📁 api/ # APIs REST internas

├── 📁 passenger-portal/ # Portal tipo Netflix

│ ├── 📁 assets/ # CSS/JS del portal

│ ├── 📁 player/ # Reproductores multimedia

│ ├── 📁 games/ # Sistema de juegos

│ └── 📁 api/ # APIs del portal

├── 📁 pi-system/ # Software para Raspberry Pi

│ ├── 📁 install/ # Scripts de instalación

│ ├── 📁 config/ # Configuraciones Pi

│ └── 📁 scripts/ # Mantenimiento y sync

├── 📁 content/ # Almacén de contenido

│ ├── 📁 movies/ # Películas y videos

│ ├── 📁 music/ # Audio y música

│ └── 📁 games/ # Juegos HTML5

├── 📁 packages/ # Paquetes generados

├── 📁 companies/ # Datos por empresa

└── 📁 database/ # Schema y migraciones

**🎨 DISEÑO Y UX:**

**PANEL ADMIN (AdminLTE 3):**

* **Tema:** Profesional y moderno
* **Colores:** Azul corporativo personalizable
* **Layout:** Sidebar + contenido principal
* **Componentes:** Cards, tablas, gráficos, modales
* **Responsive:** Funciona en desktop, tablet, móvil

**PORTAL PASAJEROS (Estilo Netflix):**

* **Tema:** Oscuro tipo streaming
* **Colores:** Personalizables por empresa
* **Layout:** Mobile-first, táctil
* **Navegación:** Carruseles horizontales, grids
* **Interacción:** Touch gestures, animaciones fluidas

**🔧 CARACTERÍSTICAS TÉCNICAS:**

**PERFORMANCE:**

* **Admin:** < 2 segundos de carga
* **Portal:** < 3 segundos en móviles
* **Video:** Streaming inmediato < 2 segundos buffering
* **Sincronización:** Delta sync para eficiencia

**ESCALABILIDAD:**

* **Empresas:** Sin límite teórico
* **Contenido:** Limitado por espacio en disco
* **Pi concurrentes:** Cientos de Pi sincronizando
* **Usuarios por Pi:** 10-70 usuarios simultáneos

**SEGURIDAD:**

* **Autenticación:** Sesiones seguras con timeout
* **APIs:** Rate limiting y validación
* **Juegos:** Sandbox iframe con restricciones
* **Datos:** Encriptación de datos sensibles

**🚀 FLUJO DE DESARROLLO:**

**DESARROLLO SECUENCIAL POR FASES:**

1. **Fase 1:** Base de datos y backend ✅
2. **Fase 2.1:** Panel admin + login ✅
3. **Fase 2.2:** Gestión de contenido 🔄
4. **Fase 2.3:** Generación de paquetes
5. **Fase 3:** Sistema QR + WiFi
6. **Fase 4:** Portal pasajeros
7. **Fase 5:** Sistema de juegos
8. **Fase 6:** Software Raspberry Pi
9. **Fase 7:** Sincronización
10. **Fase 8:** Optimización final

**METODOLOGÍA:**

* **Una fase a la vez:** No avanzar sin completar anterior
* **Testing obligatorio:** Cada módulo debe probarse
* **Documentación clara:** Cada archivo documentado
* **Conexiones explícitas:** Saber qué llama a qué

**🎯 OBJETIVOS DEL PROYECTO:**

**PARA EMPRESAS DE TRANSPORTE:**

* Diferenciarse de la competencia
* Mejorar experiencia del pasajero
* Generar ingresos adicionales (publicidad)
* Fidelizar clientes

**PARA PASAJEROS:**

* Entretenimiento durante viajes largos
* Acceso gratuito a contenido premium
* Experiencia moderna tipo Netflix
* Sin uso de datos móviles

**PARA ADMINISTRADORES:**

* Panel intuitivo sin conocimiento técnico
* Gestión centralizada de toda la flota
* Monitoreo en tiempo real
* Actualización automática de contenido

**📱 COMPATIBILIDAD:**

**DISPOSITIVOS SOPORTADOS:**

* **Móviles:** Android 6+, iOS 12+
* **Tablets:** Android, iPad
* **Navegadores:** Chrome, Safari, Firefox, Edge
* **Raspberry Pi:** Pi 4, Pi 5,

**RESOLUCIONES:**

* **Móviles:** 320px - 767px
* **Tablets:** 768px - 1023px
* **Desktop:** 1024px+ (admin panel)

**🎯 METODOLOGÍA DE DESARROLLO**

**PRINCIPIOS FUNDAMENTALES:**

* **Una fase a la vez** - No avanzar sin completar la anterior
* **Testing obligatorio** - Cada fase debe ser probada antes de continuar
* **Documentación clara** - Cada archivo debe tener su propósito explicado
* **Conexiones explícitas** - Saber exactamente qué archivo llama a cuál

**📋 FASES DE DESARROLLO SECUENCIAL**

* **🚀 FASE 3: SISTEMA QR Y CONFIGURACIÓN WIFI**
* **Objetivo:** Sistema para generar QR codes y configuración WiFi automática **Tiempo estimado:** 3-4 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── admin/
* │ ├── views/qr-system/
* │ │ ├── index.php # Gestión de códigos QR
* │ │ ├── generate.php # Generar QR por bus/empresa
* │ │ └── print.php # Página para imprimir QR
* │ ├── api/qr/
* │ │ ├── generate-qr.php # API generar código QR
* │ │ └── wifi-config.php # API configuración WiFi
* │ └── libraries/
* │ └── qr-generator/ # Librería PHP para QR codes
* ├── companies/
* │ └── [empresa-id]/
* │ ├── qr-codes/ # QR codes generados
* │ └── wifi-configs/ # Configuraciones WiFi
* **FUNCIONALIDADES REQUERIDAS:**
* Generar QR codes únicos por bus/empresa
* QR contiene configuración WiFi + URL del portal
* Configuración automática de WiFi en dispositivos
* Personalización de SSID y contraseña por empresa
* Códigos QR descargables para impresión
* Seguimiento de QR codes activos
* **FLUJO DEL QR CODE:**
* Admin genera QR para empresa/bus específico
* QR contiene: WIFI:T:WPA;S:[SSID];P:[PASSWORD];H:false;; + URL
* Pasajero escanea → se conecta automáticamente al WiFi
* Browser abre automáticamente la URL del portal
* **🚀 FASE 4: PORTAL WEB PARA PASAJEROS (TIPO NETFLIX)**
* **Objetivo:** Interfaz moderna y atractiva para que pasajeros consuman contenido **Tiempo estimado:** 8-10 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── passenger-portal/
* │ ├── index.php # Página principal tipo Netflix
* │ ├── movies.php # Catálogo de películas
* │ ├── music.php # Reproductor tipo YouTube
* │ ├── games.php # Catálogo de juegos HTML5
* │ ├── player/
* │ │ ├── video-player.php # Reproductor de video
* │ │ ├── music-player.php # Reproductor de música
* │ │ └── game-launcher.php # Lanzador de juegos
* │ ├── assets/
* │ │ ├── css/
* │ │ │ ├── netflix-style.css # Estilos tipo Netflix
* │ │ │ ├── mobile-optimized.css # Optimizado para móviles
* │ │ │ └── games.css # Estilos para juegos
* │ │ ├── js/
* │ │ │ ├── portal-main.js # JavaScript principal
* │ │ │ ├── video-controls.js # Controles de video
* │ │ │ ├── music-controls.js # Controles de música
* │ │ │ └── games-manager.js # Gestión de juegos
* │ │ └── images/
* │ │ ├── hero-banners/ # Banners principales
* │ │ └── icons/ # Iconos de la interfaz
* │ ├── api/
* │ │ ├── get-content.php # API obtener contenido
* │ │ ├── track-usage.php # Seguimiento de uso
* │ │ └── company-branding.php # Obtener branding por empresa
* │ └── config/
* │ └── portal-config.php # Configuración del portal
* **CARACTERÍSTICAS DE LA INTERFAZ:**
* **Página Principal (tipo Netflix):**
* Hero banner con contenido destacado
* Carruseles por categoría: "Películas Populares", "Música Trending", "Juegos Nuevos"
* Navegación intuitiva con iconos grandes
* Branding personalizado por empresa (colores, logo)
* Optimizado para pantallas táctiles móviles
* **Reproductor de Video:**
* Controles táctiles grandes y claros
* Reproducción en pantalla completa
* Controles de volumen y progreso
* Botones de pausa/play prominentes
* **Reproductor de Música (tipo YouTube):**
* Lista de reproducción visual
* Carátulas grandes y atractivas
* Controles de siguiente/anterior
* Visualización del progreso
* **Lanzador de Juegos:**
* Grid de juegos con screenshots atractivos
* Categorías: Puzzle, Arcade, Estrategia
* Lanzamiento en iframe o ventana nueva
* Instrucciones claras de cada juego
* **🚀 FASE 5: SISTEMA DE JUEGOS HTML5**
* **Objetivo:** Infraestructura completa para ejecutar juegos web **Tiempo estimado:** 4-5 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── passenger-portal/
* │ ├── games/
* │ │ ├── launcher.php # Lanzador universal de juegos
* │ │ ├── iframe-sandbox.php # Sandbox seguro para juegos
* │ │ └── leaderboard.php # Tabla de puntuaciones
* │ └── api/games/
* │ ├── load-game.php # Cargar juego específico
* │ ├── save-score.php # Guardar puntuaciones
* │ └── get-leaderboard.php # Obtener rankings
* ├── content/games/extracted/
* │ └── [game-id]/
* │ ├── index.html # Punto de entrada del juego
* │ ├── assets/ # Recursos del juego
* │ └── game-config.json # Configuración del juego
* **ESTRUCTURA DE JUEGO HTML5:**
* Cada juego debe tener:
* index.html - Punto de entrada principal
* assets/ - Imágenes, sonidos, sprites
* game-config.json - Metadatos y configuración
* Comunicación con portal vía postMessage para puntuaciones
* **TIPOS DE JUEGOS SOPORTADOS:**
* Puzzle (Tetris, Match-3, Sudoku)
* Arcade (Snake, Breakout, Pac-Man)
* Cartas (Solitario, Memory)
* Estrategia (Chess, Checkers)
* Casual (Bubble Shooter, Word Games)
* **🚀 FASE 6: SISTEMA RASPBERRY PI**
* **Objetivo:** Software completo para instalar en Raspberry Pi **Tiempo estimado:** 6-8 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── pi-system/
* │ ├── install/
* │ │ ├── setup.sh # Script de instalación completo
* │ │ ├── wifi-ap.sh # Configurar WiFi Access Point
* │ │ ├── web-server.sh # Instalar servidor web
* │ │ └── auto-start.sh # Configurar inicio automático
* │ ├── config/
* │ │ ├── hostapd.conf # Configuración WiFi hotspot
* │ │ ├── dnsmasq.conf # Configuración DNS/DHCP
* │ │ ├── apache-config.conf # Configuración Apache
* │ │ └── company-config.json # Configuración por empresa
* │ ├── scripts/
* │ │ ├── sync-content.sh # Sincronizar contenido del servidor
* │ │ ├── update-system.sh # Actualizar sistema Pi
* │ │ └── health-check.sh # Verificar estado del sistema
* │ └── web/
* │ └── [symlink to passenger-portal] # Portal web para pasajeros
* **FUNCIONALIDADES DEL SISTEMA PI:**
* Configuración automática como WiFi Access Point
* Servidor web Apache con contenido local
* Sincronización periódica con servidor central
* Auto-inicio al encender el Pi
* Monitoreo de salud del sistema
* Logs de uso y conexiones
* **🚀 FASE 7: SISTEMA DE SINCRONIZACIÓN**
* **Objetivo:** Sincronización automática entre servidor central y Raspberry Pi **Tiempo estimado:** 4-5 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── api/sync/
* │ ├── pi-register.php # Registrar nuevo Pi
* │ ├── check-updates.php # Verificar actualizaciones
* │ ├── download-package.php # Descargar paquete actualizado
* │ └── report-status.php # Reportar estado del Pi
* ├── admin/
* │ ├── views/pi-management/
* │ │ ├── index.php # Lista de Pi registrados
* │ │ ├── monitoring.php # Monitoreo en tiempo real
* │ │ └── logs.php # Logs de sincronización
* │ └── api/pi/
* │ ├── push-update.php # Enviar actualización a Pi
* │ └── get-pi-status.php # Obtener estado de Pi específico
* **FLUJO DE SINCRONIZACIÓN:**
* Pi se registra en servidor al iniciar
* Pi verifica actualizaciones cada X minutos
* Servidor notifica cuando hay nuevo contenido
* Pi descarga y actualiza contenido automáticamente
* Pi reporta estado y estadísticas de uso
* **🚀 FASE 8: OPTIMIZACIÓN Y FINALIZACIÓN**
* **Objetivo:** Optimizar rendimiento y completar funcionalidades **Tiempo estimado:** 3-4 horas
* **OPTIMIZACIONES FINALES:**
* Compresión de assets CSS/JS
* Optimización de imágenes y videos
* Cache de contenido en Pi
* Mejoras de velocidad de carga
* Testing completo del sistema
* Documentación de instalación
* Manual de usuario para administradores
* **📋 CRITERIOS DE ÉXITO TOTAL:**
* **Al completar todas las fases:**
* ✅ Administradores pueden gestionar empresas y contenido
* ✅ Sistema genera paquetes personalizados automáticamente
* ✅ Códigos QR conectan pasajeros automáticamente al WiFi
* ✅ Portal tipo Netflix funciona fluido en móviles
* ✅ Juegos HTML5 se ejecutan sin problemas
* ✅ Raspberry Pi funciona de forma autónoma
* ✅ Sincronización automática servidor-Pi operativa
* ✅ Sistema completo listo para producción
* **🔗 CONEXIONES CRÍTICAS ENTRE FASES:**
* **Fase 2.2** alimenta contenido → **Fase 2.3** genera paquetes
* **Fase 2.3** crea configuración → **Fase 3** genera QR codes
* **Fase 3** define WiFi → **Fase 4** portal accesible vía WiFi
* **Fase 4** consume contenido → **Fase 5** ejecuta juegos
* **Fase 6** instala sistema → **Fase 7** mantiene sincronizado
* **Todas las fases** → **Fase 8** optimiza y finaliza

**🚀 FASE 2.3: GESTIÓN DE PAQUETES Y DISTRIBUCIÓN**

**📋 DIVISIÓN EN MÓDULOS:**

**FASE 2.3 se divide en 8 MÓDULOS específicos:**

**📚 MÓDULO 2.3.1: admin/views/packages/index.php**

**Propósito:** Lista principal de paquetes generados para empresas

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de visualización con tabla interactiva
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 40 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::index()
* **Modelo:** Package::findAll(), Package::getStats()
* **Layout:** layouts/base.php
* **APIs:** api/packages/update-status.php, api/packages/delete-package.php

**📊 DATOS A MOSTRAR:**

* Nombre del paquete
* Empresa asociada
* Versión del paquete
* Fecha de creación
* Estado (generando/listo/instalado/error)
* Tamaño del paquete
* Número de archivos incluidos
* Fecha de última descarga

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Tabla con DataTables (paginación, ordenamiento, búsqueda)
* Filtros por empresa
* Filtros por estado del paquete
* Descarga directa de paquetes listos
* Regenerar paquete con contenido actualizado
* Eliminar paquetes obsoletos
* Ver log de instalación

**📈 ESTADÍSTICAS A MOSTRAR:**

* Total de paquetes generados
* Paquetes por estado
* Empresas con paquetes activos
* Tamaño total de todos los paquetes

**🎯 CRITERIOS DE ÉXITO:**

* Lista carga correctamente
* Filtros funcionan
* Descarga de paquetes funciona
* Regeneración funciona
* Estados se actualizan correctamente

**📚 MÓDULO 2.3.2: admin/views/packages/generate.php**

**Propósito:** Formulario para generar nuevos paquetes personalizados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Formulario multi-paso
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (mostrar) / POST (generar)
* **Tiempo estimado:** 1 hora 15 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::generate()
* **Modelo:** Company::findAll(), Content::findAll()
* **API:** api/packages/generate-package.php
* **JavaScript:** Wizard multi-paso, selección de contenido

**📝 PASOS DEL FORMULARIO:**

1. **Paso 1:** Seleccionar empresa
2. **Paso 2:** Configurar branding (colores, logo)
3. **Paso 3:** Seleccionar contenido (películas/música/juegos)
4. **Paso 4:** Configurar WiFi (SSID, contraseña)
5. **Paso 5:** Configurar portal (nombre, configuraciones)
6. **Paso 6:** Revisar y generar

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Wizard con navegación entre pasos
* Validación en cada paso
* Preview de selecciones
* Estimación de tamaño del paquete
* Guardado de borrador
* Generación en background con progress bar

**⚙️ CONFIGURACIONES INCLUIDAS:**

* Branding personalizado por empresa
* Selección específica de contenido
* Configuración WiFi automática
* Settings del portal pasajeros
* Configuración de QR codes

**🎯 CRITERIOS DE ÉXITO:**

* Wizard navega correctamente
* Validaciones funcionan en cada paso
* Preview se muestra correctamente
* Generación se ejecuta en background
* Progress bar funciona

**📚 MÓDULO 2.3.3: admin/views/packages/customize.php**

**Propósito:** Personalizar contenido específico por empresa

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Interfaz de personalización avanzada
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (mostrar) / POST (guardar)
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::customize()
* **Modelo:** Company::findById(), Content::getByCompany()
* **APIs:** api/packages/save-customization.php
* **JavaScript:** Drag and drop, preview en tiempo real

**🎨 OPCIONES DE PERSONALIZACIÓN:**

* Orden de contenido en portal
* Contenido destacado en hero banner
* Colores del portal según empresa
* Logo de empresa en portal
* Nombre personalizado del servicio
* Configuración de categorías

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Drag and drop para reordenar contenido
* Preview en tiempo real del portal
* Color picker para branding
* Upload de logo personalizado
* Guardado automático de cambios
* Restaurar configuración por defecto

**👁️ PREVIEW EN TIEMPO REAL:**

* Vista previa del portal con cambios aplicados
* Simulación en dispositivo móvil
* Vista de diferentes secciones del portal

**🎯 CRITERIOS DE ÉXITO:**

* Personalización se aplica correctamente
* Preview funciona en tiempo real
* Guardado automático funciona
* Restaurar defaults funciona

**📚 MÓDULO 2.3.4: admin/views/packages/history.php**

**Propósito:** Historial completo de paquetes por empresa

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de historial con timeline
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 35 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::history()
* **Modelo:** Package::getHistory(), PackageLog::getByPackage()
* **Layout:** layouts/base.php

**📊 DATOS DEL HISTORIAL:**

* Todas las versiones de paquetes por empresa
* Fechas de generación e instalación
* Cambios entre versiones
* Logs de errores o problemas
* Estadísticas de uso por versión
* Tiempo de generación de cada paquete

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Timeline visual del historial
* Comparación entre versiones
* Descarga de versiones anteriores
* Ver logs detallados de cada versión
* Filtros por empresa y fecha
* Exportar historial a PDF

**📈 MÉTRICAS MOSTRADAS:**

* Frecuencia de actualizaciones por empresa
* Tiempo promedio de generación
* Errores más comunes
* Empresas más activas

**🎯 CRITERIOS DE ÉXITO:**

* Timeline se muestra correctamente
* Comparación entre versiones funciona
* Logs se muestran detalladamente
* Filtros funcionan correctamente

**📚 MÓDULO 2.3.5: api/packages/generate-package.php**

**Propósito:** API para generar paquetes completos en background

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint (proceso pesado)
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** POST únicamente
* **Tiempo estimado:** 1 hora 30 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::generatePackage()
* **Modelos:** Package::create(), Company::findById(), Content::findByIds()
* **Utilidades:** ZipArchive, file operations
* **Templates:** packages/templates/

**⚙️ PROCESO DE GENERACIÓN:**

1. **Validar datos** de entrada (empresa, contenido, configuraciones)
2. **Crear estructura** de directorios del paquete
3. **Copiar contenido** seleccionado a directorio temporal
4. **Generar configuraciones** (WiFi, portal, QR codes)
5. **Personalizar portal** con branding de empresa
6. **Crear archivos** de configuración JSON
7. **Comprimir todo** en archivo ZIP
8. **Mover a directorio** final
9. **Actualizar base de datos** con información del paquete
10. **Limpiar archivos** temporales

**📁 ESTRUCTURA DEL PAQUETE GENERADO:**

packages/generated/[empresa-id]/[timestamp]/

├── content/

│ ├── movies/

│ ├── music/

│ └── games/

├── config/

│ ├── wifi-config.json

│ ├── portal-config.json

│ ├── company-branding.json

│ └── qr-codes.json

├── portal/

│ ├── [todos los archivos del portal]

│ └── assets/[personalizados]

├── install/

│ ├── install.sh

│ └── setup-pi.sh

└── package-info.json

**🎯 CRITERIOS DE ÉXITO:**

* Paquete se genera sin errores
* Estructura es correcta y completa
* Configuraciones se aplican bien
* ZIP se crea correctamente
* Base de datos se actualiza
* Archivos temporales se limpian

**📚 MÓDULO 2.3.6: api/packages/download-package.php**

**Propósito:** API para descargar paquetes generados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint de descarga
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** GET
* **Tiempo estimado:** 20 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::download()
* **Modelo:** Package::findById(), Package::updateDownloadCount()
* **Sistema:** Headers de descarga, stream de archivos

**⚙️ PROCESO DE DESCARGA:**

1. **Validar permisos** de descarga
2. **Verificar que archivo** existe
3. **Registrar descarga** en base de datos
4. **Configurar headers** para descarga
5. **Stream del archivo** al cliente
6. **Log de actividad**

**🔒 SEGURIDAD:**

* Validación de permisos por empresa
* Rate limiting para descargas
* Verificación de integridad del archivo
* Log de todas las descargas

**🎯 CRITERIOS DE ÉXITO:**

* Descarga inicia correctamente
* Archivo se descarga completo
* Contadores se actualizan
* Logs se registran

**📚 MÓDULO 2.3.7: api/packages/update-status.php**

**Propósito:** API para actualizar estado de paquetes

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** POST únicamente
* **Tiempo estimado:** 15 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PackageController::updateStatus()
* **Modelo:** Package::updateStatus(), PackageLog::create()

**📝 ESTADOS POSIBLES:**

* **generando:** Paquete en proceso de creación
* **listo:** Paquete completado y disponible
* **descargado:** Paquete descargado por empresa
* **instalado:** Paquete instalado en Raspberry Pi
* **activo:** Paquete funcionando en producción
* **error:** Error en generación o instalación

**⚙️ PROCESO DE ACTUALIZACIÓN:**

1. **Validar nuevo estado** es válido
2. **Verificar permisos** de cambio
3. **Actualizar estado** en base de datos
4. **Registrar cambio** en log
5. **Notificar cambio** si es necesario

**🎯 CRITERIOS DE ÉXITO:**

* Estado se actualiza correctamente
* Logs se registran
* Notificaciones se envían si aplica

**📚 MÓDULO 2.3.8: packages/templates/default-config.json**

**Propósito:** Template base para configuración de paquetes

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Archivo de configuración JSON
* **Propósito:** Template base que se personaliza por empresa
* **Tiempo estimado:** 25 minutos

**🔗 ARCHIVOS RELACIONADOS:**

* **Usado por:** api/packages/generate-package.php
* **Personalizado con:** Datos de Company y configuraciones específicas

**📝 ESTRUCTURA DEL TEMPLATE:**

json

{

"package\_info": {

"version": "1.0.0",

"generated\_at": "TIMESTAMP",

"company\_id": "COMPANY\_ID",

"package\_type": "standard"

},

"wifi\_config": {

"ssid": "COMPANY\_WIFI\_NAME",

"password": "WIFI\_PASSWORD",

"security": "WPA2",

"hidden": false

},

"portal\_config": {

"company\_name": "COMPANY\_NAME",

"service\_name": "SERVICE\_NAME",

"primary\_color": "PRIMARY\_COLOR",

"secondary\_color": "SECONDARY\_COLOR",

"logo\_path": "LOGO\_PATH",

"welcome\_message": "WELCOME\_MESSAGE"

},

"content\_config": {

"movies\_enabled": true,

"music\_enabled": true,

"games\_enabled": true,

"max\_concurrent\_users": 50

},

"system\_config": {

"auto\_sync": true,

"sync\_interval": 3600,

"log\_level": "INFO",

"analytics\_enabled": true

}

}

**🎯 CRITERIOS DE ÉXITO:**

* Template es válido JSON
* Todos los placeholders están definidos
* Estructura es completa
* Se personaliza correctamente

**🚀 FASE 3: SISTEMA QR Y CONFIGURACIÓN WIFI**

**📋 DIVISIÓN EN MÓDULOS:**

**FASE 3 se divide en 5 MÓDULOS específicos:**

**📚 MÓDULO 3.1: admin/views/qr-system/index.php**

**Propósito:** Gestión central de códigos QR por empresa y bus

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de gestión con tabla y generación
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** QRController::index()
* **Modelo:** QRCode::findAll(), Company::findAll()
* **Layout:** layouts/base.php
* **APIs:** api/qr/generate-qr.php

**📊 DATOS A MOSTRAR:**

* Lista de QR codes generados
* Empresa asociada
* Número de bus
* SSID del WiFi
* URL del portal
* Fecha de generación
* Estado (activo/inactivo)
* Número de escaneos

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Tabla con filtros por empresa
* Generar QR individual o masivo
* Preview del QR code
* Descargar QR para impresión
* Estadísticas de uso por QR
* Activar/desactivar QR codes

**📈 ESTADÍSTICAS MOSTRADAS:**

* Total de QR codes generados
* QR codes activos por empresa
* Escaneos totales hoy/semana/mes
* QR codes más utilizados

**🎯 CRITERIOS DE ÉXITO:**

* Lista se muestra correctamente
* Generación funciona sin errores
* Preview se muestra bien
* Descarga funciona
* Estadísticas son precisas

**📚 MÓDULO 3.2: admin/views/qr-system/generate.php**

**Propósito:** Formulario para generar QR codes personalizados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Formulario de generación con preview
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (mostrar) / POST (generar)
* **Tiempo estimado:** 40 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** QRController::generate()
* **Modelo:** Company::findAll(), QRCode::create()
* **API:** api/qr/generate-qr.php
* **Librería:** QR code generator (PHP)

**📝 CAMPOS DEL FORMULARIO:**

* Selección de empresa
* Número de bus
* SSID personalizado (opcional)
* Contraseña WiFi personalizada
* URL del portal (auto-generada)
* Tamaño del QR code
* Nivel de corrección de errores

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Preview en tiempo real del QR
* Validación de SSID y contraseña
* Generación masiva para flota completa
* Diferentes formatos de descarga (PNG, SVG, PDF)
* Configuración avanzada de WiFi

**📋 TIPOS DE QR GENERADOS:**

* **QR WiFi:** Conexión automática al WiFi
* **QR Portal:** Directo al portal web
* **QR Combinado:** WiFi + Portal en secuencia

**🎯 CRITERIOS DE ÉXITO:**

* Preview se actualiza en tiempo real
* Validaciones funcionan correctamente
* Generación masiva funciona
* Diferentes formatos se descargan bien

**📚 MÓDULO 3.3: admin/views/qr-system/print.php**

**Propósito:** Página optimizada para impresión de QR codes

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de impresión sin layout
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 25 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** QRController::print()
* **Modelo:** QRCode::findById(), Company::findById()
* **CSS:** Estilos específicos para impresión

**🖨️ LAYOUT DE IMPRESIÓN:**

* QR code grande y centrado
* Información de la empresa
* Instrucciones para el pasajero
* Número de bus si aplica
* Información de contacto

**📐 CONFIGURACIONES DE IMPRESIÓN:**

* Tamaño estándar (A4, Carta)
* Múltiples QR por página
* Versión en blanco y negro
* Versión con colores de empresa

**🎯 CRITERIOS DE ÉXITO:**

* Página se imprime correctamente
* QR code es escaneable en impresión
* Layout es profesional
* Instrucciones son claras

**📚 MÓDULO 3.4: api/qr/generate-qr.php**

**Propósito:** API para generar códigos QR con configuración WiFi

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** POST únicamente
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** QRController::generateQR()
* **Modelo:** QRCode::create(), Company::findById()
* **Librería:** PHP QR Code generator
* **Destino:** companies/[id]/qr-codes/

**⚙️ PROCESO DE GENERACIÓN:**

1. **Validar datos** de entrada (empresa, configuración WiFi)
2. **Generar string** WiFi: WIFI:T:WPA;S:SSID;P:password;H:false;;
3. **Agregar URL** del portal al string
4. **Generar QR code** con librería
5. **Guardar imagen** en directorio de empresa
6. **Registrar en BD** con metadatos
7. **Retornar información** del QR generado

**📱 FORMATO DEL QR WIFI:**

WIFI:T:WPA;S:[SSID];P:[PASSWORD];H:false;;

URL:[PORTAL\_URL]

**🎨 PERSONALIZACIÓN:**

* Colores de empresa en QR
* Logo de empresa en centro (opcional)
* Diferentes niveles de corrección
* Tamaños personalizados

**🎯 CRITERIOS DE ÉXITO:**

* QR se genera correctamente
* WiFi se conecta automáticamente
* URL del portal se abre automáticamente
* Archivo se guarda en ubicación correcta
* Base de datos se actualiza

**📚 MÓDULO 3.5: api/qr/wifi-config.php**

**Propósito:** API para generar configuraciones WiFi personalizadas

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 30 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** QRController::wifiConfig()
* **Modelo:** Company::getWifiConfig()
* **Templates:** WiFi configuration templates

**📝 CONFIGURACIONES GENERADAS:**

* Configuración para Android
* Configuración para iOS
* Configuración para Windows
* Archivo de configuración para routers

**⚙️ TIPOS DE CONFIGURACIÓN:**

1. **Automática:** SSID basado en nombre de empresa
2. **Personalizada:** SSID y contraseña específicos
3. **Segura:** Contraseñas generadas aleatoriamente
4. **Temporal:** Configuraciones con expiración

**🔒 SEGURIDAD:**

* Contraseñas seguras generadas automáticamente
* Rotación periódica de contraseñas
* Configuración de seguridad WPA2/WPA3

**🎯 CRITERIOS DE ÉXITO:**

* Configuraciones se generan correctamente
* Son compatibles con todos los dispositivos
* Seguridad es apropiada
* Se integran bien con QR codes

**🚀 FASE 4: PORTAL WEB PARA PASAJEROS (TIPO NETFLIX)**

**📋 DIVISIÓN EN MÓDULOS:**

**FASE 4 se divide en 12 MÓDULOS específicos:**

**📚 MÓDULO 4.1: passenger-portal/index.php**

**Propósito:** Página principal estilo Netflix para pasajeros

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página principal SPA (Single Page Application)
* **Autenticación:** No requerida (acceso público local)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora 30 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PortalController::index()
* **Modelo:** Content::getActiveContent()
* **APIs:** api/get-content.php, api/company-branding.php
* **Assets:** netflix-style.css, portal-main.js

**🎨 DISEÑO ESTILO NETFLIX:**

* **Hero Banner:** Contenido destacado rotativo
* **Carruseles:** Películas populares, música trending, juegos nuevos
* **Navegación:** Menú superior con categorías
* **Responsive:** Optimizado para móviles y tablets
* **Branding:** Colores y logo personalizado por empresa

**📱 SECCIONES PRINCIPALES:**

1. **Header:** Logo empresa, menú navegación, búsqueda
2. **Hero Banner:** Contenido destacado con trailer/preview
3. **Películas:** Carrusel horizontal con posters
4. **Música:** Grid con carátulas y artistas
5. **Juegos:** Grid con screenshots atractivos
6. **Footer:** Información de la empresa y soporte

**🎛️ FUNCIONALIDADES INTERACTIVAS:**

* Carruseles deslizables tactiles
* Hover effects en contenido
* Search bar con resultados en vivo
* Filtros rápidos por categoría
* Favoritos (localStorage)
* Contenido recientemente visto

**📊 DATOS DINÁMICOS:**

* Contenido personalizado por empresa
* Estadísticas de popularidad
* Contenido agregado recientemente
* Recomendaciones basadas en uso

**🎯 CRITERIOS DE ÉXITO:**

* Carga rápida (< 3 segundos)
* Navegación fluida en táctil
* Responsive en todos los dispositivos
* Branding se aplica correctamente
* Contenido se carga dinámicamente

**📚 MÓDULO 4.2: passenger-portal/movies.php**

**Propósito:** Catálogo completo de películas estilo Netflix

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de catálogo con grid responsive
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PortalController::movies()
* **Modelo:** Content::getMovies()
* **APIs:** api/get-content.php?type=movies
* **Player:** player/video-player.php

**🎬 LAYOUT DE PELÍCULAS:**

* **Filtros superiores:** Género, año, duración
* **Grid responsive:** 2-6 columnas según dispositivo
* **Cada película:** Poster, título, duración, rating
* **Hover/Tap:** Preview con descripción y botón play
* **Paginación:** Infinite scroll o páginas

**🎭 INFORMACIÓN POR PELÍCULA:**

* Poster de alta calidad
* Título y subtítulo
* Duración y año
* Género y clasificación
* Descripción corta
* Rating/estrellas

**🔍 FUNCIONALIDADES:**

* Búsqueda en tiempo real
* Filtros múltiples combinables
* Ordenamiento (popular, reciente, A-Z)
* Vista en lista o grid
* Favoritos personales

**🎯 CRITERIOS DE ÉXITO:**

* Grid es responsive y rápido
* Filtros funcionan sin recargar
* Búsqueda es instantánea
* Player se abre correctamente
* Navegación es intuitiva

**📚 MÓDULO 4.3: passenger-portal/music.php**

**Propósito:** Reproductor de música estilo YouTube/Spotify

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de reproductor de música
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora 15 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PortalController::music()
* **Modelo:** Content::getMusic()
* **Player:** player/music-player.php
* **APIs:** api/get-content.php?type=music

**🎵 LAYOUT DE MÚSICA:**

* **Player principal:** En parte superior fija
* **Lista de canciones:** Tabla con artista, álbum, duración
* **Sidebar:** Playlists, favoritos, recientes
* **Visualización:** Carátula grande del álbum actual
* **Controles:** Play/pause, siguiente, anterior, volumen

**🎼 FUNCIONALIDADES DEL PLAYER:**

* Reproducción continua
* Lista de reproducción dinámica
* Shuffle y repeat
* Control de volumen
* Barra de progreso interactiva
* Información del track actual

**📱 CONTROLES TÁCTILES:**

* Botones grandes para móviles
* Gestures para cambiar canción
* Control de volumen por gestos
* Lock screen controls

**🎯 CRITERIOS DE ÉXITO:**

* Player funciona sin interrupciones
* Controles son responsive táctiles
* Lista de reproducción funciona
* Audio se reproduce sin cortes
* Interfaz es intuitiva

**📚 MÓDULO 4.4: passenger-portal/games.php**

**Propósito:** Catálogo de juegos HTML5 con launcher

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Galería de juegos con launcher integrado
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PortalController::games()
* **Modelo:** Content::getGames()
* **Launcher:** games/launcher.php
* **APIs:** api/games/load-game.php

**🎮 LAYOUT DE JUEGOS:**

* **Grid de juegos:** Screenshots atractivos
* **Categorías:** Puzzle, Arcade, Estrategia, Cartas
* **Cada juego:** Screenshot, título, descripción, rating
* **Preview:** Modal con información e instrucciones
* **Launcher:** Iframe o ventana nueva

**🕹️ INFORMACIÓN POR JUEGO:**

* Screenshot o GIF animado
* Título del juego
* Categoría y dificultad
* Instrucciones básicas
* Controles necesarios
* Tiempo estimado de juego

**🎪 FUNCIONALIDADES:**

* Filtros por categoría
* Búsqueda de juegos
* Favoritos personales
* Leaderboard si aplica
* Fullscreen para juegos

**🎯 CRITERIOS DE ÉXITO:**

* Grid se carga rápidamente
* Juegos se lanzan sin errores
* Fullscreen funciona correctamente
* Controles son claros
* Navegación entre juegos es fluida

**📚 MÓDULO 4.5: passenger-portal/player/video-player.php**

**Propósito:** Reproductor de video optimizado para móviles

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Player de video HTML5 personalizado
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PlayerController::video()
* **Modelo:** Content::getMovie()
* **APIs:** api/track-usage.php
* **Assets:** video-controls.js, player.css

**📺 CARACTERÍSTICAS DEL PLAYER:**

* **HTML5 Video:** Compatibilidad universal
* **Controles personalizados:** Grandes para táctil
* **Fullscreen:** Orientación automática
* **Calidad adaptativa:** Según ancho de banda
* **Subtítulos:** Si están disponibles

**🎛️ CONTROLES INCLUIDOS:**

* Play/Pause (botón grande central)
* Barra de progreso (draggable)
* Control de volumen
* Botón fullscreen
* Botón configuración
* Tiempo actual/total

**📱 OPTIMIZACIONES MÓVILES:**

* Controles auto-hide
* Tap para mostrar/ocultar controles
* Pinch to zoom en fullscreen
* Orientación landscape automática
* Prevención de sleep del dispositivo

**📊 TRACKING:**

* Tiempo de visualización
* Pausas y abandono
* Calidad de reproducción
* Errores de buffering

**🎯 CRITERIOS DE ÉXITO:**

* Video se reproduce sin buffering
* Controles son táctiles y responsivos
* Fullscreen funciona correctamente
* Calidad se adapta automáticamente
* Tracking registra correctamente

**📚 MÓDULO 4.6: passenger-portal/player/music-player.php**

**Propósito:** Reproductor de audio con controles avanzados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Player de audio HTML5 con visualizaciones
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PlayerController::music()
* **Modelo:** Content::getMusic()
* **Assets:** music-controls.js, audio-visualizer.js

**🎵 CARACTERÍSTICAS DEL PLAYER:**

* **Audio HTML5:** Alta calidad de reproducción
* **Visualizador:** Barras de frecuencia animadas
* **Metadata:** Artista, álbum, duración
* **Artwork:** Carátula del álbum grande
* **Playlist:** Cola de reproducción visible

**🎚️ CONTROLES AVANZADOS:**

* Play/Pause/Stop
* Anterior/Siguiente
* Shuffle/Repeat
* Control de volumen con mute
* Ecualizador básico (opcional)
* Speed control

**📱 DISEÑO RESPONSIVO:**

* Layout vertical para móviles
* Controles grandes táctiles
* Gestures para navegación
* Background play (si es posible)

**🎯 CRITERIOS DE ÉXITO:**

* Audio se reproduce con calidad
* Visualizador funciona fluido
* Controles responden inmediatamente
* Playlist navega correctamente
* Diseño es atractivo

**📚 MÓDULO 4.7: passenger-portal/games/launcher.php**

**Propósito:** Launcher universal para juegos HTML5

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Iframe sandbox para juegos
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 40 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** GamesController::launch()
* **Modelo:** Content::getGame()
* **APIs:** api/games/load-game.php, api/games/save-score.php
* **Assets:** games-manager.js

**🎮 CARACTERÍSTICAS DEL LAUNCHER:**

* **Iframe seguro:** Sandbox con permisos limitados
* **Fullscreen:** Modo pantalla completa
* **Comunicación:** PostMessage con portal principal
* **Error handling:** Manejo de juegos que no cargan
* **Exit button:** Volver al portal fácilmente

**🛡️ SEGURIDAD:**

* Sandbox attributes apropiados
* Dominio aislado para juegos
* Validación de contenido HTML5
* Prevención de scripts maliciosos

**📊 INTEGRACIÓN:**

* Tracking de tiempo de juego
* Guardado de puntuaciones
* Comunicación de eventos
* Manejo de errores de carga

**🎯 CRITERIOS DE ÉXITO:**

* Juegos cargan correctamente en iframe
* Fullscreen funciona sin errores
* Comunicación portal-juego funciona
* Exit button siempre accesible
* No hay vulnerabilidades de seguridad

**📚 MÓDULO 4.8: passenger-portal/api/get-content.php**

**Propósito:** API para obtener contenido dinámico del portal

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API REST endpoint
* **Autenticación:** No requerida (local)
* **Método HTTP:** GET
* **Tiempo estimado:** 35 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** PortalController::getContent()
* **Modelo:** Content::getActiveContent()
* **Cache:** Sistema de cache local

**📋 PARÁMETROS SOPORTADOS:**

* type: movies/music/games/all
* category: Categoría específica
* limit: Número de resultados
* offset: Para paginación
* search: Término de búsqueda
* featured: Solo contenido destacado

**📊 RESPUESTA JSON:**

json

{

"success": true,

"data": [

{

"id": "movie\_123",

"title": "Título",

"type": "movie",

"thumbnail": "url\_thumbnail",

"duration": 7200,

"metadata": {...}

}

],

"pagination": {

"total": 150,

"page": 1,

"per\_page": 20

}

}

**⚡ OPTIMIZACIONES:**

* Cache de respuestas frecuentes
* Compresión JSON
* Lazy loading de thumbnails
* Minificación de metadata

**🎯 CRITERIOS DE ÉXITO:**

* Respuesta rápida (< 200ms)
* JSON bien formateado
* Filtros funcionan correctamente
* Paginación funciona
* Cache mejora rendimiento

**📚 MÓDULO 4.9: passenger-portal/api/track-usage.php**

**Propósito:** API para tracking de uso de contenido

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint para analytics
* **Autenticación:** No requerida
* **Método HTTP:** POST
* **Tiempo estimado:** 25 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** AnalyticsController::track()
* **Modelo:** UsageLog::create()

**📊 DATOS TRACKED:**

* Contenido reproducido/jugado
* Tiempo de uso/reproducción
* Tipo de dispositivo
* Momento del día
* Abandonos prematuros
* Errores de reproducción

**📝 EVENTOS SOPORTADOS:**

* content\_view: Ver contenido
* content\_play: Reproducir contenido
* content\_pause: Pausar reproducción
* content\_complete: Completar contenido
* content\_error: Error en reproducción
* search\_query: Búsqueda realizada

**⚡ PROCESAMIENTO:**

* Guardado asíncrono
* Batch processing para múltiples eventos
* Limpieza automática de datos antiguos

**🎯 CRITERIOS DE ÉXITO:**

* Tracking no afecta rendimiento
* Datos se guardan correctamente
* No hay pérdida de eventos
* Procesamiento es eficiente

**📚 MÓDULO 4.10: passenger-portal/api/company-branding.php**

**Propósito:** API para obtener branding personalizado por empresa

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 20 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** BrandingController::get()
* **Modelo:** Company::getBranding()

**🎨 DATOS DE BRANDING:**

* Colores primario y secundario
* Logo de la empresa
* Nombre del servicio
* Mensaje de bienvenida
* Configuraciones de tema
* Assets personalizados

**📱 RESPUESTA JSON:**

json

{

"success": true,

"branding": {

"company\_name": "Transportes ABC",

"service\_name": "EntretenimientoABC",

"primary\_color": "#FF6B35",

"secondary\_color": "#F7931E",

"logo\_url": "url\_logo",

"welcome\_message": "Bienvenido a bordo",

"theme": "light"

}

}

**💨 CACHE:**

* Cache agresivo (24 horas)
* CDN para assets si disponible
* Compresión de respuestas

**🎯 CRITERIOS DE ÉXITO:**

* Branding se aplica correctamente
* Colores se actualizan dinámicamente
* Logo se carga rápidamente
* Cache funciona eficientemente

**✅ AL COMPLETAR TODAS LAS FASES:**

**🎯 FUNCIONALIDADES CORE:**

1. ✅ **Panel administrativo** completo y funcional
2. ✅ **Gestión de empresas** con CRUD completo
3. ✅ **Gestión de contenido** multimedia (películas/música/juegos)
4. ✅ **Generación de paquetes** personalizados automática
5. ✅ **Sistema QR + WiFi** para conexión automática
6. ✅ **Portal tipo Netflix** optimizado para móviles
7. ✅ **Sistema de juegos HTML5** con sandbox seguro
8. ✅ **Instalación automática** en Raspberry Pi
9. ✅ **Sincronización automática** servidor-Pi
10. ✅ **Monitoreo en tiempo real** de toda la red

**📊 KPIS DE ÉXITO:**

* **Performance:** Portal carga en < 3 segundos
* **Uptime:** Sistema disponible > 99.5% del tiempo
* **User experience:** Conexión WiFi automática funciona > 95%
* **Sync reliability:** Sincronización exitosa > 95% de las veces
* **Content delivery:** Video inicia en < 2 segundos
* **System stability:** Pi funcionan sin reiniciar > 7 días

**🔒 SEGURIDAD:**

* Sin vulnerabilidades críticas de seguridad
* Datos de empresas protegidos apropiadamente
* Juegos ejecutan en sandbox seguro
* APIs protegidas contra ataques comunes

**📱 COMPATIBILIDAD:**

* Funciona en 95% de dispositivos móviles modernos
* Compatible con diferentes modelos de Raspberry Pi
* Funciona en diferentes navegadores móviles

**🎯 EXPERIENCIA DE USUARIO:**

* **Admin:** Puede gestionar todo sin conocimiento técnico
* **Pasajero:** Conexión y uso intuitivo en < 30 segundos
* **Empresa:** Instalación y operación sin soporte técnico

**📋 DOCUMENTACIÓN DETALLADA - MÓDULOS FALTANTES DEL PANEL ADMINISTRATIVO**

**🎯 MÓDULO: SISTEMA DE PUBLICIDAD**

**📋 DIVISIÓN EN SUBMÓDULOS:**

**El módulo de publicidad se divide en 10 SUBMÓDULOS específicos:**

**📚 MÓDULO ADV.1: admin/views/advertising/index.php**

**Propósito:** Panel principal de gestión de publicidad y campañas

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Dashboard de publicidad
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** AdvertisingController::index()
* **Modelo:** Advertisement::findAll(), Campaign::getActiveStats()
* **Layout:** layouts/base.php
* **APIs:** api/advertising/toggle-status.php, api/advertising/get-stats.php
* **JavaScript:** DataTables, Chart.js para métricas

**📊 DATOS A MOSTRAR:**

* Lista de anuncios activos/inactivos
* Campañas en curso
* Métricas de visualización
* Ingresos generados
* CTR (Click Through Rate)
* Espacios publicitarios disponibles

**🎛️ FUNCIONALIDADES REQUERIDAS:**

* Tabla con todos los anuncios (DataTables)
* Filtros por estado, tipo, empresa
* Gráficos de rendimiento
* Activar/desactivar anuncios rápidamente
* Preview de anuncios
* Calendario de campañas

**📈 ESTADÍSTICAS A MOSTRAR:**

* Total de impresiones hoy/semana/mes
* Ingresos totales por período
* Top 5 anuncios más vistos
* Tasa de conversión promedio
* Espacios publicitarios ocupados

**🎯 CRITERIOS DE ÉXITO:**

* Dashboard carga en < 2 segundos
* Métricas se actualizan en tiempo real
* Filtros funcionan correctamente
* Preview muestra anuncios exactos
* Cambios de estado son inmediatos

**📚 MÓDULO ADV.2: admin/views/advertising/create.php**

**Propósito:** Formulario para crear nuevos anuncios y campañas

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Formulario multi-paso
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (mostrar) / POST (crear)
* **Tiempo estimado:** 1 hora 15 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** AdvertisingController::create(), store()
* **Modelo:** Advertisement::create(), Campaign::create()
* **APIs:** api/advertising/upload-media.php, api/advertising/validate-campaign.php
* **JavaScript:** Wizard multi-paso, preview en tiempo real

**📝 PASOS DEL FORMULARIO:**

1. **Información básica:** Nombre, tipo, duración
2. **Targeting:** Empresas, rutas, horarios
3. **Contenido:** Upload de imágenes/videos
4. **Ubicación:** Selección de espacios publicitarios
5. **Presupuesto:** Modelo de cobro, límites
6. **Preview y confirmación**

**🎨 TIPOS DE ANUNCIOS:**

* **Banner estático:** Imagen JPG/PNG
* **Banner animado:** GIF animado
* **Video corto:** MP4 (máx 30 segundos)
* **Intersticial:** Pantalla completa entre contenido
* **Native ad:** Integrado en catálogo

**📍 ESPACIOS PUBLICITARIOS:**

* Hero banner principal
* Entre carruseles de contenido
* Antes de reproducir video
* Entre canciones (audio)
* Pantalla de carga de juegos
* Footer del portal

**⚙️ CONFIGURACIONES:**

* **Frecuencia:** Cada X minutos/vistas
* **Límite diario:** Máximo de impresiones
* **Horario:** Horas específicas del día
* **Segmentación:** Por tipo de contenido visto

**🎯 CRITERIOS DE ÉXITO:**

* Wizard navega sin errores
* Upload de media funciona correctamente
* Preview se actualiza en tiempo real
* Validaciones previenen errores
* Campaña se crea correctamente

**📚 MÓDULO ADV.3: admin/views/advertising/campaigns.php**

**Propósito:** Gestión completa de campañas publicitarias

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Panel de gestión de campañas
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** CampaignController::index()
* **Modelo:** Campaign::findAll(), Campaign::getMetrics()
* **APIs:** api/advertising/campaign-stats.php
* **JavaScript:** Calendar view, charts

**📊 VISTA DE CAMPAÑAS:**

* Calendario visual de campañas
* Timeline de campañas activas
* Estado: planificada/activa/pausada/finalizada
* Presupuesto consumido vs total
* Performance metrics por campaña

**🎛️ FUNCIONALIDADES:**

* Crear nueva campaña
* Pausar/reanudar campañas
* Editar parámetros de campaña
* Clonar campañas exitosas
* Extender duración
* Ajustar presupuesto

**📈 MÉTRICAS POR CAMPAÑA:**

* Impresiones totales
* CTR (Click-through rate)
* CPC (Costo por clic)
* ROI (Return on Investment)
* Conversiones
* Alcance único

**🎯 CRITERIOS DE ÉXITO:**

* Calendario muestra campañas claramente
* Métricas son precisas y actuales
* Cambios se aplican inmediatamente
* Timeline es interactivo
* Export de datos funciona

**📚 MÓDULO ADV.4: admin/views/advertising/reports.php**

**Propósito:** Reportes detallados de rendimiento publicitario

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Dashboard de reportes
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** AdvertisingReportsController::index()
* **Modelo:** AdvertisementMetrics::getReports()
* **APIs:** api/advertising/generate-report.php
* **Libraries:** Chart.js, jsPDF para exportación

**📊 TIPOS DE REPORTES:**

* **Performance general:** Métricas globales
* **Por anunciante:** Desglose por empresa
* **Por ubicación:** Rendimiento por espacio
* **Por período:** Comparativas temporales
* **ROI analysis:** Retorno de inversión
* **A/B testing:** Comparación de variantes

**📈 VISUALIZACIONES:**

* Gráficos de línea (tendencias)
* Gráficos de barras (comparativas)
* Pie charts (distribución)
* Heat maps (horarios pico)
* Tablas detalladas exportables

**🎛️ FILTROS DISPONIBLES:**

* Rango de fechas personalizado
* Empresa/anunciante específico
* Tipo de anuncio
* Espacio publicitario
* Dispositivo (móvil/tablet)

**📄 EXPORTACIÓN:**

* PDF con gráficos
* Excel con datos raw
* CSV para análisis externo
* Programar envío automático

**🎯 CRITERIOS DE ÉXITO:**

* Reportes generan en < 5 segundos
* Gráficos son claros e informativos
* Exportación incluye todos los datos
* Filtros funcionan correctamente
* Datos son 100% precisos

**📚 MÓDULO ADV.5: api/advertising/upload-media.php**

**Propósito:** API para subir contenido publicitario

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint de upload
* **Autenticación:** Requerida (token/sesión)
* **Método HTTP:** POST
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** AdvertisingController::uploadMedia()
* **Modelo:** Advertisement::saveMedia()
* **Storage:** content/advertising/
* **Validación:** Formatos y tamaños permitidos

**📁 TIPOS DE ARCHIVO:**

* **Imágenes:** JPG, PNG, GIF (máx 5MB)
* **Videos:** MP4, WebM (máx 50MB)
* **Dimensiones:** Según espacio publicitario

**⚙️ PROCESO DE UPLOAD:**

1. Validar tipo y tamaño de archivo
2. Verificar dimensiones según espacio
3. Optimizar imagen/comprimir video
4. Generar thumbnails
5. Guardar en directorio apropiado
6. Registrar en base de datos
7. Retornar URLs de acceso

**🎯 CRITERIOS DE ÉXITO:**

* Upload maneja archivos grandes
* Optimización reduce tamaño sin perder calidad
* Validaciones previenen archivos incorrectos
* Thumbnails se generan correctamente

**📚 MÓDULO ADV.6: api/advertising/track-impression.php**

**Propósito:** API para registrar impresiones y clics

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint de tracking
* **Autenticación:** No requerida (público)
* **Método HTTP:** POST
* **Tiempo estimado:** 40 minutos

**📝 DATOS TRACKED:**

* Advertisement ID
* Timestamp
* Tipo de evento (impression/click)
* Device info
* Session ID
* Ubicación del anuncio
* Duración de visualización (videos)

**⚙️ PROCESAMIENTO:**

* Validación de datos
* Deduplicación de impresiones
* Guardado asíncrono para performance
* Actualización de contadores en tiempo real
* Detección de fraude básica

**🎯 CRITERIOS DE ÉXITO:**

* Tracking no afecta performance del portal
* Datos se guardan correctamente
* Métricas se actualizan en tiempo real
* Sistema detecta y filtra bots básicos

**📚 MÓDULO ADV.7: admin/models/Advertisement.php**

**Propósito:** Modelo de datos para anuncios

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Modelo MVC
* **Extends:** BaseModel
* **Tiempo estimado:** 30 minutos

**📝 ESTRUCTURA DE DATOS:**

php

- id

- campaign\_id

- title

- type (banner/video/intersticial/native)

- media\_url

- thumbnail\_url

- target\_url

- placement (hero/carousel/preroll/etc)

- status (active/paused/ended)

- impressions\_limit

- impressions\_count

- clicks\_count

- start\_date

- end\_date

- created\_at

- updated\_at

**🔧 MÉTODOS PRINCIPALES:**

* getActiveAds()
* getByPlacement($placement)
* incrementImpression()
* incrementClick()
* checkLimit()
* getMetrics($dateRange)

**📚 MÓDULO ADV.8: admin/models/Campaign.php**

**Propósito:** Modelo de datos para campañas publicitarias

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Modelo MVC
* **Extends:** BaseModel
* **Tiempo estimado:** 25 minutos

**📝 ESTRUCTURA DE DATOS:**

php

- id

- advertiser\_id

- name

- budget\_total

- budget\_spent

- payment\_model (CPM/CPC/CPA)

- target\_companies

- target\_schedule

- status

- start\_date

- end\_date

**🔧 MÉTODOS PRINCIPALES:**

* getActiveCampaigns()
* updateBudgetSpent()
* checkBudgetLimit()
* getROI()
* getCampaignMetrics()

**📚 MÓDULO ADV.9: passenger-portal/api/get-ads.php**

**Propósito:** API para obtener anuncios en el portal

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint público
* **Autenticación:** No requerida
* **Método HTTP:** GET
* **Tiempo estimado:** 35 minutos

**📋 PARÁMETROS:**

* placement: Ubicación del anuncio
* limit: Número de anuncios
* exclude: IDs para no repetir

**📊 RESPUESTA:**

json

{

"success": true,

"ads": [

{

"id": "ad\_123",

"type": "banner",

"media\_url": "/content/advertising/banner.jpg",

"target\_url": "http://example.com",

"tracking\_url": "/api/advertising/track-impression.php"

}

]

}

**🎯 CRITERIOS DE ÉXITO:**

* Anuncios se seleccionan según targeting
* Rotación evita repetición excesiva
* Performance es óptimo
* Tracking URL funciona correctamente

**📚 MÓDULO ADV.10: database/migrations/advertising\_tables.sql**

**Propósito:** Estructura de base de datos para publicidad

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** SQL Migration
* **Tiempo estimado:** 20 minutos

**📝 TABLAS A CREAR:**

sql

*-- Tabla de anuncios*

CREATE TABLE advertisements (

id INT PRIMARY KEY AUTO\_INCREMENT,

campaign\_id INT,

title VARCHAR(255),

type ENUM('banner', 'video', 'intersticial', 'native'),

media\_url VARCHAR(500),

thumbnail\_url VARCHAR(500),

target\_url VARCHAR(500),

placement VARCHAR(100),

status ENUM('active', 'paused', 'ended'),

impressions\_limit INT,

impressions\_count INT DEFAULT 0,

clicks\_count INT DEFAULT 0,

start\_date DATETIME,

end\_date DATETIME,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (campaign\_id) REFERENCES campaigns(id)

);

*-- Tabla de campañas*

CREATE TABLE campaigns (

id INT PRIMARY KEY AUTO\_INCREMENT,

advertiser\_id INT,

name VARCHAR(255),

budget\_total DECIMAL(10,2),

budget\_spent DECIMAL(10,2) DEFAULT 0,

payment\_model ENUM('CPM', 'CPC', 'CPA'),

target\_companies JSON,

target\_schedule JSON,

status ENUM('draft', 'active', 'paused', 'completed'),

start\_date DATE,

end\_date DATE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

*-- Tabla de métricas*

CREATE TABLE advertisement\_metrics (

id INT PRIMARY KEY AUTO\_INCREMENT,

advertisement\_id INT,

event\_type ENUM('impression', 'click', 'conversion'),

device\_info JSON,

session\_id VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (advertisement\_id) REFERENCES advertisements(id),

INDEX idx\_ad\_date (advertisement\_id, created\_at)

);

**🎯 MÓDULO: SISTEMA DE NOTIFICACIONES**

**📋 DIVISIÓN EN SUBMÓDULOS:**

**El módulo de notificaciones se divide en 8 SUBMÓDULOS específicos:**

**📚 MÓDULO NOT.1: admin/views/notifications/index.php**

**Propósito:** Centro de control de notificaciones del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Dashboard de notificaciones
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** NotificationController::index()
* **Modelo:** Notification::findAll(), NotificationQueue::getQueue()
* **Layout:** layouts/base.php
* **APIs:** api/notifications/send.php, api/notifications/mark-read.php

**📊 SECCIONES PRINCIPALES:**

* **Cola de envío:** Notificaciones pendientes
* **Historial:** Notificaciones enviadas
* **Templates:** Plantillas disponibles
* **Configuración:** Settings de envío

**🎛️ FUNCIONALIDADES:**

* Ver todas las notificaciones del sistema
* Filtrar por tipo, estado, destinatario
* Reenviar notificaciones fallidas
* Crear notificación manual
* Ver logs de envío
* Estadísticas de entrega

**📈 MÉTRICAS:**

* Notificaciones enviadas hoy/semana/mes
* Tasa de entrega exitosa
* Notificaciones fallidas
* Tiempo promedio de envío

**🎯 CRITERIOS DE ÉXITO:**

* Lista se actualiza en tiempo real
* Filtros funcionan correctamente
* Reenvío funciona sin duplicar
* Estadísticas son precisas

**📚 MÓDULO NOT.2: admin/views/notifications/create.php**

**Propósito:** Crear y enviar notificaciones manuales

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Formulario de creación
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (form) / POST (send)
* **Tiempo estimado:** 40 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** NotificationController::create(), send()
* **Modelo:** Notification::create(), NotificationQueue::add()
* **Templates:** Sistema de plantillas

**📝 TIPOS DE NOTIFICACIÓN:**

* **Email:** Correo electrónico
* **SMS:** Mensaje de texto (futuro)
* **Push:** Notificación push (futuro)
* **In-app:** Dentro del sistema

**📋 CAMPOS DEL FORMULARIO:**

* Tipo de notificación
* Destinatarios (individual/grupo/todos)
* Asunto
* Mensaje (editor WYSIWYG)
* Template a usar
* Prioridad (alta/normal/baja)
* Programar envío

**🎨 FUNCIONALIDADES:**

* Preview en tiempo real
* Selector de destinatarios
* Variables dinámicas {{nombre}}, {{empresa}}
* Adjuntar archivos
* Programación de envío

**🎯 CRITERIOS DE ÉXITO:**

* Preview muestra resultado final
* Variables se reemplazan correctamente
* Envío programado funciona
* Editor WYSIWYG funciona bien

**📚 MÓDULO NOT.3: admin/views/notifications/templates.php**

**Propósito:** Gestión de plantillas de notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Gestor de templates
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** NotificationTemplateController::index()
* **Modelo:** NotificationTemplate::findAll()
* **Editor:** Code editor para HTML

**📋 TEMPLATES PREDEFINIDOS:**

* Bienvenida nueva empresa
* Paquete listo para descarga
* Pi offline/online
* Actualización disponible
* Reporte mensual
* Alerta de sistema
* Vencimiento de suscripción

**🎛️ FUNCIONALIDADES:**

* Lista de templates
* Editor de código HTML
* Preview de template
* Variables disponibles
* Clonar template
* Activar/desactivar

**📝 ESTRUCTURA DE TEMPLATE:**

html

*<!-- Header -->*

<h1>{{subject}}</h1>

*<!-- Body -->*

<p>Hola {{company\_name}},</p>

<p>{{message\_content}}</p>

*<!-- Footer -->*

<p>Saludos,<br>Equipo PLAYMI</p>

**🎯 CRITERIOS DE ÉXITO:**

* Editor funciona con syntax highlighting
* Preview renderiza correctamente
* Variables se documentan claramente
* Templates se guardan sin errores

**📚 MÓDULO NOT.4: admin/views/notifications/settings.php**

**Propósito:** Configuración del sistema de notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Página de configuración
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET (mostrar) / POST (guardar)
* **Tiempo estimado:** 35 minutos

**⚙️ CONFIGURACIONES:**

* **Email:**
  + Servidor SMTP
  + Puerto y seguridad
  + Credenciales
  + Email remitente
  + Nombre remitente
* **Límites:**
  + Máximo por hora
  + Máximo por día
  + Reintentos en fallo
  + Timeout de envío
* **Notificaciones automáticas:**
  + Nueva empresa registrada
  + Pi offline por X tiempo
  + Espacio en disco bajo
  + Errores críticos

**🎯 CRITERIOS DE ÉXITO:**

* Configuración SMTP funciona
* Test de envío exitoso
* Límites se respetan
* Notificaciones automáticas se activan

**📚 MÓDULO NOT.5: api/notifications/send.php**

**Propósito:** API para enviar notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token)
* **Método HTTP:** POST
* **Tiempo estimado:** 45 minutos

**📝 PARÁMETROS:**

json

{

"type": "email",

"recipients": ["email@example.com"],

"subject": "Asunto",

"message": "Contenido",

"template\_id": 1,

"variables": {

"company\_name": "Transportes ABC"

},

"priority": "normal",

"schedule": "2024-12-25 10:00:00"

}

**⚙️ PROCESO:**

1. Validar datos de entrada
2. Cargar template si aplica
3. Reemplazar variables
4. Agregar a cola de envío
5. Procesar según prioridad
6. Enviar notificación
7. Registrar resultado

**🎯 CRITERIOS DE ÉXITO:**

* Validaciones funcionan correctamente
* Variables se reemplazan bien
* Cola procesa por prioridad
* Logs registran todo

**📚 MÓDULO NOT.6: api/notifications/get-queue.php**

**Propósito:** API para obtener estado de cola de notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token)
* **Método HTTP:** GET
* **Tiempo estimado:** 25 minutos

**📊 RESPUESTA:**

json

{

"success": true,

"queue": {

"pending": 15,

"processing": 3,

"failed": 2,

"sent\_today": 145

},

"notifications": [

{

"id": 123,

"type": "email",

"recipient": "user@example.com",

"status": "pending",

"priority": "high",

"scheduled\_at": "2024-12-25 10:00:00"

}

]

}

**📚 MÓDULO NOT.7: cron/process-notifications.php**

**Propósito:** Script cron para procesar cola de notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Script PHP para cron
* **Ejecución:** Cada minuto
* **Tiempo estimado:** 40 minutos

**⚙️ PROCESO:**

1. Obtener notificaciones pendientes
2. Ordenar por prioridad y fecha
3. Procesar batch según límites
4. Enviar cada notificación
5. Manejar errores y reintentos
6. Actualizar estados
7. Limpiar cola antigua

**🎯 CRITERIOS DE ÉXITO:**

* Procesa eficientemente sin saturar
* Respeta límites de envío
* Maneja errores graciosamente
* No duplica envíos

**📚 MÓDULO NOT.8: database/migrations/notification\_tables.sql**

**Propósito:** Tablas para sistema de notificaciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** SQL Migration
* **Tiempo estimado:** 20 minutos

**📝 ESTRUCTURA:**

sql

*-- Tabla de notificaciones*

CREATE TABLE notifications (

id INT PRIMARY KEY AUTO\_INCREMENT,

type ENUM('email', 'sms', 'push', 'in-app'),

recipient VARCHAR(255),

subject VARCHAR(255),

message TEXT,

template\_id INT,

variables JSON,

priority ENUM('high', 'normal', 'low') DEFAULT 'normal',

status ENUM('pending', 'sent', 'failed') DEFAULT 'pending',

attempts INT DEFAULT 0,

sent\_at DATETIME,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (template\_id) REFERENCES notification\_templates(id)

);

*-- Tabla de templates*

CREATE TABLE notification\_templates (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255),

type ENUM('email', 'sms', 'push', 'in-app'),

subject VARCHAR(255),

content TEXT,

variables JSON,

active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*-- Tabla de logs*

CREATE TABLE notification\_logs (

id INT PRIMARY KEY AUTO\_INCREMENT,

notification\_id INT,

status VARCHAR(50),

message TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (notification\_id) REFERENCES notifications(id)

);

**🎯 MÓDULO: SISTEMA DE REPORTES**

**📋 DIVISIÓN EN SUBMÓDULOS:**

**El módulo de reportes se divide en 10 SUBMÓDULOS específicos:**

**📚 MÓDULO REP.1: admin/views/reports/index.php**

**Propósito:** Dashboard principal de reportes del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Centro de reportes
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** ReportsController::index()
* **Modelo:** Report::getAvailableReports()
* **Layout:** layouts/base.php
* **JavaScript:** Chart.js, DataTables

**📊 CATEGORÍAS DE REPORTES:**

1. **Operacionales:**
   * Uso del sistema
   * Performance de Pi
   * Estadísticas de contenido
2. **Financieros:**
   * Ingresos por empresa
   * Costos operativos
   * ROI por servicio
3. **Analíticos:**
   * Comportamiento de usuarios
   * Tendencias de consumo
   * Análisis predictivo

**🎛️ FUNCIONALIDADES:**

* Catálogo de reportes disponibles
* Generación bajo demanda
* Programación de reportes
* Historial de reportes generados
* Favoritos personales

**🎯 CRITERIOS DE ÉXITO:**

* Navegación intuitiva entre reportes
* Generación rápida (< 10 segundos)
* Interfaz clara y organizada

**📚 MÓDULO REP.2: admin/views/reports/usage.php**

**Propósito:** Reporte detallado de uso del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Reporte analítico
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora 15 minutos

**📊 MÉTRICAS INCLUIDAS:**

* **Usuarios únicos:** Por día/semana/mes
* **Sesiones totales:** Duración promedio
* **Contenido consumido:**
  + Top películas vistas
  + Música más escuchada
  + Juegos más jugados
* **Horarios pico:** Heat map de uso
* **Dispositivos:** Distribución móvil/tablet

**📈 VISUALIZACIONES:**

* Timeline de usuarios activos
* Gráfico de barras por tipo de contenido
* Pie chart de dispositivos
* Heat map de horarios
* Top 10 listas

**🔍 FILTROS:**

* Rango de fechas
* Empresa específica
* Tipo de contenido
* Dispositivo
* Ubicación/Pi

**📄 EXPORTACIÓN:**

* PDF con gráficos
* Excel detallado
* CSV raw data
* Programar envío

**🎯 CRITERIOS DE ÉXITO:**

* Datos son 100% precisos
* Gráficos cargan rápidamente
* Filtros actualizan en tiempo real
* Export incluye todos los datos

**📚 MÓDULO REP.3: admin/views/reports/financial.php**

**Propósito:** Reportes financieros y de facturación

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Reporte financiero
* **Autenticación:** Requerida (admin + permisos financieros)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora 30 minutos

**💰 REPORTES FINANCIEROS:**

1. **Ingresos:**
   * Por empresa
   * Por período
   * Por servicio
   * Proyecciones
2. **Costos:**
   * Infraestructura
   * Ancho de banda
   * Mantenimiento
   * Personal
3. **Rentabilidad:**
   * Margen por empresa
   * ROI por servicio
   * Break-even analysis
   * Profit & Loss

**📊 FUNCIONALIDADES:**

* Estado de cuenta por empresa
* Facturas pendientes/pagadas
* Aging de cuentas por cobrar
* Comparativas período anterior
* Forecast financiero

**📈 DASHBOARDS:**

* KPIs financieros principales
* Gráfico de ingresos vs costos
* Cash flow mensual
* Distribución de ingresos
* Tendencias de crecimiento

**🎯 CRITERIOS DE ÉXITO:**

* Cálculos 100% precisos
* Seguridad de datos financieros
* Auditoría de accesos
* Reportes cumplen normas contables

**📚 MÓDULO REP.4: admin/views/reports/performance.php**

**Propósito:** Reportes de rendimiento del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Reporte técnico
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**📊 MÉTRICAS DE PERFORMANCE:**

* **Servidor central:**
  + CPU/Memoria/Disco
  + Response time APIs
  + Uptime del servicio
  + Errores por hora
* **Red de Pi:**
  + Estado general de flota
  + Pi con problemas
  + Temperatura promedio
  + Sincronización exitosa %
* **Portal pasajeros:**
  + Tiempo de carga
  + Errores JavaScript
  + Performance por dispositivo

**🎛️ FUNCIONALIDADES:**

* Alertas de performance
* Comparativas históricas
* Identificar bottlenecks
* Recomendaciones automáticas

**🎯 CRITERIOS DE ÉXITO:**

* Identifica problemas proactivamente
* Datos en tiempo real
* Recomendaciones son útiles

**📚 MÓDULO REP.5: admin/views/reports/content.php**

**Propósito:** Análisis detallado de contenido

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Reporte de contenido
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**📊 ANÁLISIS DE CONTENIDO:**

* **Popularidad:**
  + Top contenido por vistas
  + Contenido menos visto
  + Trending esta semana
  + Rating por usuarios
* **Distribución:**
  + Contenido por categoría
  + Espacio ocupado por tipo
  + Antigüedad del catálogo
* **Engagement:**
  + Completion rate videos
  + Skip rate música
  + Tiempo jugado promedio

**🎯 CRITERIOS DE ÉXITO:**

* Insights accionables
* Identifica contenido poco popular
* Sugiere optimizaciones

**📚 MÓDULO REP.6: admin/views/reports/custom.php**

**Propósito:** Creador de reportes personalizados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Report builder
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 2 horas

**🎛️ REPORT BUILDER:**

* Drag & drop de métricas
* Filtros personalizables
* Múltiples visualizaciones
* Fórmulas personalizadas
* Guardado de templates

**📊 ELEMENTOS DISPONIBLES:**

* Métricas del sistema
* Dimensiones (tiempo, empresa, etc)
* Filtros avanzados
* Tipos de gráficos
* Tablas pivote

**🎯 CRITERIOS DE ÉXITO:**

* Builder es intuitivo
* Reportes se generan correctamente
* Templates se pueden compartir
* Performance es aceptable

**📚 MÓDULO REP.7: api/reports/generate.php**

**Propósito:** API para generar reportes

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token)
* **Método HTTP:** POST
* **Tiempo estimado:** 1 hora

**📝 PARÁMETROS:**

json

{

"report\_type": "usage",

"date\_range": {

"start": "2024-01-01",

"end": "2024-12-31"

},

"filters": {

"company\_id": 123,

"content\_type": "movies"

},

"format": "pdf",

"email\_to": "admin@example.com"

}

**⚙️ PROCESO:**

1. Validar permisos y parámetros
2. Ejecutar queries necesarias
3. Procesar y agregar datos
4. Generar visualizaciones
5. Crear archivo en formato solicitado
6. Enviar por email si solicitado
7. Guardar en historial

**🎯 CRITERIOS DE ÉXITO:**

* Genera reportes sin timeout
* Maneja grandes volúmenes de datos
* Formatos son profesionales
* Email delivery funciona

**📚 MÓDULO REP.8: api/reports/schedule.php**

**Propósito:** API para programar reportes automáticos

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (token)
* **Método HTTP:** POST
* **Tiempo estimado:** 40 minutos

**📝 CONFIGURACIÓN:**

json

{

"report\_config": {

"type": "financial",

"frequency": "monthly",

"day\_of\_month": 1,

"time": "08:00",

"recipients": ["cfo@company.com"],

"format": "excel"

}

}

**🎯 CRITERIOS DE ÉXITO:**

* Programación funciona confiablemente
* Reportes se envían puntualmente
* Gestión de errores apropiada

**📚 MÓDULO REP.9: cron/generate-scheduled-reports.php**

**Propósito:** Generar reportes programados

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Script cron
* **Ejecución:** Cada hora
* **Tiempo estimado:** 35 minutos

**⚙️ PROCESO:**

1. Verificar reportes programados para esta hora
2. Generar cada reporte en queue
3. Enviar por email
4. Registrar en log
5. Manejar errores

**📚 MÓDULO REP.10: database/migrations/reports\_tables.sql**

**Propósito:** Estructura para sistema de reportes

**📝 TABLAS:**

sql

*-- Configuración de reportes*

CREATE TABLE report\_configs (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255),

type VARCHAR(50),

config JSON,

created\_by INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*-- Reportes programados*

CREATE TABLE scheduled\_reports (

id INT PRIMARY KEY AUTO\_INCREMENT,

report\_config\_id INT,

frequency ENUM('daily', 'weekly', 'monthly'),

schedule\_config JSON,

recipients JSON,

active BOOLEAN DEFAULT TRUE,

last\_run DATETIME,

next\_run DATETIME,

FOREIGN KEY (report\_config\_id) REFERENCES report\_configs(id)

);

*-- Historial de reportes*

CREATE TABLE report\_history (

id INT PRIMARY KEY AUTO\_INCREMENT,

report\_type VARCHAR(50),

parameters JSON,

generated\_by INT,

file\_path VARCHAR(500),

file\_size INT,

generation\_time INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**🎯 MÓDULO: CONFIGURACIÓN DEL SISTEMA**

**📋 DIVISIÓN EN SUBMÓDULOS:**

**El módulo de configuración se divide en 8 SUBMÓDULOS específicos:**

**📚 MÓDULO CFG.1: admin/views/settings/index.php**

**Propósito:** Panel principal de configuración del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Dashboard de configuración
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** SettingsController::index()
* **Modelo:** Settings::getAll()
* **Layout:** layouts/base.php

**⚙️ SECCIONES DE CONFIGURACIÓN:**

1. **General:**
   * Nombre del sistema
   * URL base
   * Timezone
   * Idioma predeterminado
2. **Email:**
   * Servidor SMTP
   * Credenciales
   * Templates email
3. **Almacenamiento:**
   * Rutas de archivos
   * Límites de tamaño
   * Políticas de limpieza
4. **Performance:**
   * Cache settings
   * Límites de sistema
   * Optimizaciones
5. **Seguridad:**
   * Políticas de contraseña
   * Timeout de sesión
   * IPs permitidas

**🎯 CRITERIOS DE ÉXITO:**

* Configuraciones se cargan correctamente
* Cambios se aplican inmediatamente
* Validaciones previenen errores
* Backup automático de configs

**📚 MÓDULO CFG.2: admin/views/settings/general.php**

**Propósito:** Configuraciones generales del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Formulario de configuración
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 40 minutos

**⚙️ CONFIGURACIONES:**

php

- site\_name: "PLAYMI"

- site\_url: "https://playmi.com"

- admin\_email: "admin@playmi.com"

- timezone: "America/Lima"

- date\_format: "Y-m-d"

- time\_format: "H:i:s"

- default\_language: "es"

- maintenance\_mode: false

- debug\_mode: false

**🎛️ FUNCIONALIDADES:**

* Validación de URL
* Selector de timezone
* Preview de formatos
* Modo mantenimiento con mensaje
* Toggle debug mode

**🎯 CRITERIOS DE ÉXITO:**

* Cambios se guardan correctamente
* Validaciones funcionan
* Timezone afecta todo el sistema
* Modo mantenimiento funciona

**📚 MÓDULO CFG.3: admin/views/settings/email.php**

**Propósito:** Configuración del sistema de email

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Configuración SMTP
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 35 minutos

**📧 CONFIGURACIONES SMTP:**

* Host SMTP
* Puerto (25/465/587)
* Seguridad (none/ssl/tls)
* Usuario SMTP
* Contraseña SMTP
* From email
* From name

**🎛️ FUNCIONALIDADES:**

* Test de conexión SMTP
* Envío de email de prueba
* Configuración de templates
* Logs de email

**🎯 CRITERIOS DE ÉXITO:**

* Test SMTP funciona
* Emails se envían correctamente
* Configuración se encripta
* Logs registran envíos

**📚 MÓDULO CFG.4: admin/views/settings/storage.php**

**Propósito:** Configuración de almacenamiento

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Gestión de storage
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 45 minutos

**💾 CONFIGURACIONES:**

* **Rutas:**
  + Contenido multimedia
  + Archivos temporales
  + Backups
  + Logs
* **Límites:**
  + Max upload size
  + Max video size
  + Max storage per company
  + Temp files lifetime
* **Políticas:**
  + Auto-limpieza
  + Compresión automática
  + Archivado de logs

**📊 INFORMACIÓN:**

* Espacio usado/disponible
* Distribución por tipo
* Archivos más grandes
* Sugerencias de limpieza

**🎯 CRITERIOS DE ÉXITO:**

* Muestra uso actual correcto
* Límites se aplican
* Limpieza automática funciona
* No afecta archivos en uso

**📚 MÓDULO CFG.5: admin/views/settings/security.php**

**Propósito:** Configuraciones de seguridad

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Panel de seguridad
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 1 hora

**🔒 CONFIGURACIONES DE SEGURIDAD:**

1. **Contraseñas:**
   * Longitud mínima
   * Requiere mayúsculas
   * Requiere números
   * Requiere símbolos
   * Expiración días
2. **Sesiones:**
   * Timeout inactividad
   * Sesiones concurrentes
   * Remember me duración
3. **Acceso:**
   * IP whitelist
   * Bloqueo tras X intentos
   * 2FA habilitado
4. **Auditoría:**
   * Log de accesos
   * Log de cambios
   * Retención de logs

**🎯 CRITERIOS DE ÉXITO:**

* Políticas se aplican inmediatamente
* No bloquea acceso legítimo
* Logs son comprensivos
* 2FA funciona correctamente

**📚 MÓDULO CFG.6: admin/views/settings/backup.php**

**Propósito:** Gestión de backups del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Sistema de backups
* **Autenticación:** Requerida (super admin)
* **Método HTTP:** GET/POST
* **Tiempo estimado:** 1 hora 15 minutos

**💾 TIPOS DE BACKUP:**

* **Full backup:** Sistema completo
* **Database only:** Solo BD
* **Content only:** Solo multimedia
* **Config only:** Configuraciones

**⚙️ CONFIGURACIONES:**

* Frecuencia automática
* Retención de backups
* Destino (local/remoto)
* Compresión
* Encriptación

**🎛️ FUNCIONALIDADES:**

* Backup manual inmediato
* Restaurar backup
* Descargar backup
* Verificar integridad
* Schedule automático

**🎯 CRITERIOS DE ÉXITO:**

* Backups se crean sin errores
* Restauración funciona
* No interrumpe servicio
* Backups son íntegros

**📚 MÓDULO CFG.7: api/settings/update.php**

**Propósito:** API para actualizar configuraciones

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint
* **Autenticación:** Requerida (super admin token)
* **Método HTTP:** POST
* **Tiempo estimado:** 40 minutos

**⚙️ PROCESO:**

1. Validar permisos super admin
2. Validar configuración según tipo
3. Backup configuración actual
4. Aplicar nuevos valores
5. Verificar aplicación correcta
6. Registrar en log de auditoría
7. Limpiar cache si necesario

**🎯 CRITERIOS DE ÉXITO:**

* Solo super admin puede cambiar
* Validaciones previenen configs inválidas
* Rollback automático si falla
* Auditoría registra todo

**📚 MÓDULO CFG.8: database/migrations/settings\_tables.sql**

**Propósito:** Estructura para configuraciones

**📝 ESTRUCTURA:**

sql

*-- Tabla de configuraciones*

CREATE TABLE settings (

id INT PRIMARY KEY AUTO\_INCREMENT,

category VARCHAR(50),

key\_name VARCHAR(100) UNIQUE,

value TEXT,

type ENUM('string', 'integer', 'boolean', 'json'),

description TEXT,

updated\_by INT,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

*-- Tabla de backups de configuración*

CREATE TABLE settings\_backup (

id INT PRIMARY KEY AUTO\_INCREMENT,

settings\_data JSON,

backed\_up\_by INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

*-- Auditoría de cambios*

CREATE TABLE settings\_audit (

id INT PRIMARY KEY AUTO\_INCREMENT,

setting\_key VARCHAR(100),

old\_value TEXT,

new\_value TEXT,

changed\_by INT,

changed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**🎯 MÓDULO: LOGS DEL SISTEMA**

**📋 DIVISIÓN EN SUBMÓDULOS:**

**El módulo de logs se divide en 7 SUBMÓDULOS específicos:**

**📚 MÓDULO LOG.1: admin/views/logs/index.php**

**Propósito:** Visor principal de logs del sistema

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Dashboard de logs
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 1 hora

**🔗 CONEXIONES DE ARCHIVOS:**

* **Controlador:** LogsController::index()
* **Modelo:** SystemLog::findAll()
* **JavaScript:** DataTables, real-time updates

**📝 TIPOS DE LOGS:**

1. **Sistema:** Errores PHP, warnings
2. **Acceso:** Login/logout, páginas visitadas
3. **Actividad:** CRUD operations
4. **API:** Llamadas API, responses
5. **Seguridad:** Intentos fallidos, cambios críticos
6. **Performance:** Queries lentas, timeouts

**🎛️ FUNCIONALIDADES:**

* Filtros avanzados
* Búsqueda en tiempo real
* Tail -f (seguimiento en vivo)
* Export de logs
* Limpieza de logs antiguos
* Alertas para errores críticos

**📊 ESTADÍSTICAS:**

* Errores por hora/día
* Tipos de error más comunes
* IPs con más actividad
* Páginas con más errores

**🎯 CRITERIOS DE ÉXITO:**

* Logs cargan rápidamente
* Búsqueda es eficiente
* Real-time funciona sin lag
* Filtros son precisos

**📚 MÓDULO LOG.2: admin/views/logs/access.php**

**Propósito:** Logs de acceso y autenticación

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Log viewer especializado
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 45 minutos

**📊 INFORMACIÓN DE ACCESO:**

* Usuario
* IP address
* User agent
* Timestamp
* Acción (login/logout/failed)
* Duración de sesión

**🔍 FUNCIONALIDADES:**

* Filtro por usuario
* Filtro por IP
* Detección de patrones sospechosos
* Mapa de ubicaciones (GeoIP)
* Timeline de accesos

**🚨 ALERTAS:**

* Múltiples intentos fallidos
* Acceso desde IP nueva
* Cambio de ubicación drástico
* Sesiones simultáneas

**🎯 CRITERIOS DE ÉXITO:**

* Detecta accesos sospechosos
* GeoIP funciona correctamente
* Timeline es visual y claro
* Alertas son precisas

**📚 MÓDULO LOG.3: admin/views/logs/activity.php**

**Propósito:** Registro de actividad de usuarios

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Activity log viewer
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 50 minutos

**📝 ACTIVIDADES REGISTRADAS:**

* Create: Nuevos registros
* Update: Modificaciones
* Delete: Eliminaciones
* View: Consultas sensibles
* Export: Descargas de datos
* Config: Cambios de configuración

**📊 INFORMACIÓN POR ACTIVIDAD:**

* Usuario responsable
* Timestamp exacto
* Recurso afectado
* Valores anteriores/nuevos
* IP de origen

**🎯 CRITERIOS DE ÉXITO:**

* Auditoría completa de cambios
* Búsqueda por entidad funciona
* Diff de cambios es claro
* Timeline por usuario

**📚 MÓDULO LOG.4: admin/views/logs/errors.php**

**Propósito:** Visor especializado de errores

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Error log analyzer
* **Autenticación:** Requerida (admin)
* **Método HTTP:** GET
* **Tiempo estimado:** 55 minutos

**❌ CATEGORÍAS DE ERROR:**

* Fatal errors
* Warnings
* Notices
* JavaScript errors
* API errors
* Database errors

**📊 ANÁLISIS:**

* Stack traces completos
* Frecuencia de errores
* Errores agrupados
* Tendencias temporales
* Impacto en usuarios

**🔧 HERRAMIENTAS:**

* Búsqueda por stack trace
* Marcar como resuelto
* Asignar a desarrollador
* Crear ticket automático
* Ignorar errores conocidos

**🎯 CRITERIOS DE ÉXITO:**

* Agrupa errores similares
* Stack traces son útiles
* Permite workflow de resolución
* Identifica errores críticos

**📚 MÓDULO LOG.5: api/logs/stream.php**

**Propósito:** API para streaming de logs en tiempo real

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** API endpoint (SSE/WebSocket)
* **Autenticación:** Requerida (token)
* **Método HTTP:** GET (EventSource)
* **Tiempo estimado:** 45 minutos

**🔄 FUNCIONALIDAD:**

* Server-Sent Events
* Filtros en real-time
* Buffer limitado
* Auto-reconnect

**📊 FORMATO DE EVENTOS:**

javascript

data: {

"type": "error",

"level": "critical",

"message": "Database connection failed",

"timestamp": "2024-12-25 10:30:45",

"context": {...}

}

**🎯 CRITERIOS DE ÉXITO:**

* Streaming sin interrupciones
* Bajo consumo de recursos
* Filtros funcionan en real-time
* Maneja reconexiones

**📚 MÓDULO LOG.6: cron/cleanup-logs.php**

**Propósito:** Limpieza automática de logs antiguos

**📋 ESPECIFICACIONES TÉCNICAS:**

* **Tipo:** Script cron
* **Ejecución:** Diaria
* **Tiempo estimado:** 30 minutos

**⚙️ PROCESO:**

1. Identificar logs antiguos según política
2. Comprimir logs antes de archivar
3. Mover a almacenamiento de archivo
4. Eliminar logs según retención
5. Optimizar tablas de BD
6. Generar reporte de limpieza

**📋 POLÍTICAS DE RETENCIÓN:**

* Errores críticos: 1 año
* Logs de acceso: 6 meses
* Actividad normal: 3 meses
* Debug logs: 7 días

**🎯 CRITERIOS DE ÉXITO:**

* No elimina logs importantes
* Compresión eficiente
* No afecta performance
* Respeta políticas

**📚 MÓDULO LOG.7: database/migrations/logs\_tables.sql**

**Propósito:** Estructura para sistema de logs

**📝 ESTRUCTURA:**

sql

*-- Logs del sistema*

CREATE TABLE system\_logs (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

level ENUM('debug', 'info', 'warning', 'error', 'critical'),

category VARCHAR(50),

message TEXT,

context JSON,

user\_id INT,

ip\_address VARCHAR(45),

user\_agent TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_level\_date (level, created\_at),

INDEX idx\_category\_date (category, created\_at)

);

*-- Logs de acceso*

CREATE TABLE access\_logs (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

action ENUM('login', 'logout', 'failed\_login'),

ip\_address VARCHAR(45),

user\_agent TEXT,

location JSON,

session\_id VARCHAR(128),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_user\_date (user\_id, created\_at),

INDEX idx\_ip\_date (ip\_address, created\_at)

);

*-- Logs de actividad*

CREATE TABLE activity\_logs (

id BIGINT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

action ENUM('create', 'update', 'delete', 'view', 'export'),

entity\_type VARCHAR(50),

entity\_id INT,

old\_values JSON,

new\_values JSON,

ip\_address VARCHAR(45),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

INDEX idx\_entity (entity\_type, entity\_id),

INDEX idx\_user\_action (user\_id, action)

);

**📊 RESUMEN DE MÓDULOS FALTANTES**

**TOTAL DE SUBMÓDULOS A DESARROLLAR: 50**

1. **PUBLICIDAD:** 10 submódulos
2. **NOTIFICACIONES:** 8 submódulos
3. **REPORTES:** 10 submódulos
4. **CONFIGURACIÓN:** 8 submódulos
5. **LOGS:** 7 submódulos