**📋 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