**📚 DOCUMENTACIÓN TÉCNICA COMPLETA - PROYECTO PLAYMI**

**Sistema de Entretenimiento para Buses Interprovinciales**

**📋 RESUMEN GENERAL DEL PROYECTO**

**🎯 CONCEPTO:**

PLAYMI es una plataforma de entretenimiento offline que funciona en buses interprovinciales. Los pasajeros escanean un QR code, se conectan al WiFi del bus y acceden a películas, música y juegos desde sus dispositivos móviles.

**💼 MODELO DE NEGOCIO:**

* **3 tipos de paquetes:** Básico, Intermedio, Premium
* **Diferencia:** Cantidad de videos publicitarios (empresa vs PLAYMI)
* **Personalización:** Logo, colores y publicidad por empresa
* **Licencias:** Sistema pre-pagado con vencimiento

**🏗️ COMPONENTES PRINCIPALES:**

1. **Panel Admin Web:** Gestión completa del sistema
2. **Sistema Pi:** Interfaz de streaming en Raspberry Pi
3. **Sistema de Paquetes:** Generación e instalación automática
4. **Sistema de Notificaciones:** Alertas por email

**🎯 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 1: FUNDACIÓN DEL SISTEMA**

*Objetivo: Crear la base sólida sobre la cual construir todo*

**FASE 1.1: Estructura Base y Base de Datos**

**Tiempo estimado: 2-3 horas**

**Archivos a crear:**

PLAYMI/

├── database/

│ ├── schema.sql # Estructura completa de tablas

│ ├── initial-data.sql # Datos iniciales (admin, configs)

│ └── test-connection.php # Script para probar conexión

├── admin/

│ └── config/

│ ├── database.php # Clase de conexión a BD

│ ├── system.php # Configuraciones generales

│ └── constants.php # Constantes del sistema

└── logs/

└── .gitkeep # Mantener carpeta en git

**Propósito de cada archivo:**

* **schema.sql** → Define toda la estructura de base de datos
* **initial-data.sql** → Inserta usuario admin y configuraciones básicas
* **database.php** → Maneja todas las conexiones a la base de datos
* **system.php** → Define rutas, URLs y configuraciones del sistema
* **constants.php** → Define precios, límites y valores fijos

**Conexiones entre archivos:**

* schema.sql → Se ejecuta primero en MySQL/phpMyAdmin
* initial-data.sql → Se ejecuta después de schema.sql
* database.php → Será usado por TODOS los modelos y controladores
* system.php → Será incluido en TODAS las páginas del admin
* constants.php → Será usado para cálculos y validaciones

**Criterios de éxito:**

* Base de datos "playmi" creada con todas las tablas
* Usuario admin por defecto funciona
* Script de conexión a BD funciona sin errores
* Todas las constantes están definidas
* Carpeta logs tiene permisos de escritura

**Testing requerido:**

1. Ejecutar schema.sql en phpMyAdmin

2. Ejecutar initial-data.sql en phpMyAdmin

3. Ejecutar test-connection.php desde navegador

4. Verificar que no hay errores de PHP

**FASE 1.2: Modelos de Datos (Capa de Acceso a Datos)**

**Tiempo estimado: 3-4 horas**

**Archivos a crear:**

PLAYMI/

├── admin/

│ └── models/

│ ├── BaseModel.php # Modelo padre con funciones comunes

│ ├── User.php # Modelo para usuarios admin

│ ├── Company.php # Modelo para empresas

│ ├── Content.php # Modelo para contenido multimedia

│ ├── Package.php # Modelo para paquetes generados

│ └── Notification.php # Modelo para notificaciones

**Propósito de cada archivo:**

* **BaseModel.php** → Funciones comunes (CRUD básico) para todos los modelos
* **User.php** → Autenticación, gestión de usuarios admin
* **Company.php** → CRUD de empresas, consultas específicas de empresas
* **Content.php** → CRUD de contenido, filtros por tipo (películas/música/juegos)
* **Package.php** → Gestión de paquetes generados, historial
* **Notification.php** → Gestión de notificaciones, cola de emails

**Conexiones entre archivos:**

* BaseModel.php → Heredado por TODOS los otros modelos
* User.php → Será usado por AuthController (Fase 1.3)
* Company.php → Será usado por CompanyController (Fase 1.3)
* Content.php → Será usado por ContentController (Fase 1.3)
* Package.php → Será usado por PackageController (Fase 1.3)
* Notification.php → Será usado por NotificationController (Fase 1.3)

**Funciones requeridas por modelo:**

**BaseModel:** findAll(), findById(), create(), update(), delete() **User:** authenticate(), createUser(), updatePassword() **Company:** getActiveCompanies(), getExpiringCompanies(), getTotalRevenue() **Content:** getByType(), getMovies(), getMusic(), getGames(), getTotalSize() **Package:** getByCompany(), generatePackage(), updateStatus() **Notification:** sendEmail(), getQueue(), markAsSent()

**Criterios de éxito:**

* Todos los modelos extienden BaseModel correctamente
* Cada modelo tiene sus funciones específicas implementadas
* Conexión a database.php funciona en todos los modelos
* No hay errores de sintaxis PHP

**Testing requerido:**

1. Instanciar cada modelo sin errores

2. Probar función findAll() en cada modelo

3. Probar funciones específicas de cada modelo

4. Verificar que herencia de BaseModel funciona

**FASE 1.3: Controladores (Lógica de Negocio)**

**Tiempo estimado: 4-5 horas**

**Archivos a crear:**

PLAYMI/

├── admin/

│ └── controllers/

│ ├── BaseController.php # Controlador padre

│ ├── AuthController.php # Login/logout/autenticación

│ ├── DashboardController.php # Dashboard principal

│ ├── CompanyController.php # Gestión de empresas

│ ├── ContentController.php # Gestión de contenido

│ ├── PackageController.php # Generación de paquetes

│ └── NotificationController.php # Envío de notificaciones

**Propósito de cada archivo:**

* **BaseController.php** → Funciones comunes, verificación de sesión
* **AuthController.php** → Maneja login, logout, verificación de permisos
* **DashboardController.php** → Estadísticas, gráficos, resumen general
* **CompanyController.php** → CRUD completo de empresas, subida de logos
* **ContentController.php** → CRUD de contenido, subida de archivos multimedia
* **PackageController.php** → Genera paquetes para Pi, descarga, historial
* **NotificationController.php** → Envía emails, gestiona templates

**Conexiones entre archivos:**

* BaseController.php → Heredado por TODOS los controladores
* AuthController.php → Usa User.php, será llamado por login.php
* DashboardController.php → Usa Company.php, Content.php, será llamado por index.php
* CompanyController.php → Usa Company.php, será llamado por views/companies/
* ContentController.php → Usa Content.php, será llamado por views/content/
* PackageController.php → Usa Package.php, Company.php, Content.php
* NotificationController.php → Usa Notification.php, Company.php

**Métodos requeridos por controlador:**

**BaseController:** checkAuth(), requireLogin(), redirect(), setMessage() **AuthController:** login(), logout(), checkSession() **DashboardController:** index(), getStats(), getRecentActivity() **CompanyController:** index(), create(), store(), edit(), update(), delete() **ContentController:** index(), upload(), store(), edit(), update(), delete() **PackageController:** index(), generate(), download(), history() **NotificationController:** index(), send(), templates(), configure()

**Criterios de éxito:**

* Todos los controladores extienden BaseController
* Métodos de autenticación funcionan correctamente
* CRUD completo implementado en cada controlador
* Validaciones de datos implementadas
* Manejo de errores implementado

**Testing requerido:**

1. Probar login/logout completo

2. Verificar que páginas requieren autenticación

3. Probar CRUD de empresas sin errores

4. Probar subida de archivos

5. Verificar manejo de errores

**FASE 2: INTERFAZ DE USUARIO (ADMIN PANEL)**

*Objetivo: Crear todas las pantallas del panel administrativo*

**FASE 2.1: Layout Base y Sistema de Autenticación**

**Tiempo estimado: 3-4 horas**

**Archivos a crear:**

PLAYMI/

├── admin/

│ ├── assets/

│ │ ├── css/

│ │ │ ├── adminlte.min.css # AdminLTE framework

│ │ │ ├── custom.css # Estilos personalizados

│ │ │ └── login.css # Estilos específicos del login

│ │ ├── js/

│ │ │ ├── adminlte.min.js # AdminLTE JavaScript

│ │ │ ├── jquery.min.js # jQuery

│ │ │ └── custom.js # JavaScript personalizado

│ │ └── images/

│ │ ├── logo-playmi.png # Logo principal

│ │ └── login-bg.jpg # Fondo de login

│ ├── views/

│ │ └── layouts/

│ │ ├── header.php # Header común (navbar)

│ │ ├── sidebar.php # Sidebar común (menú)

│ │ ├── footer.php # Footer común

│ │ └── auth-layout.php # Layout para login

│ ├── login.php # Página de login

│ └── logout.php # Script de logout

**Propósito de cada archivo:**

* **adminlte.min.css/js** → Framework AdminLTE para interfaz profesional
* **custom.css/js** → Personalizaciones específicas de PLAYMI
* **header.php** → Navbar superior con notificaciones y perfil
* **sidebar.php** → Menú lateral con todas las opciones del admin
* **footer.php** → Footer con información de copyright
* **login.php** → Página de autenticación con formulario

**Conexiones entre archivos:**

* login.php → Usa AuthController.php → Usa User.php
* header.php → Incluido en TODAS las páginas del admin
* sidebar.php → Incluido en TODAS las páginas del admin
* footer.php → Incluido en TODAS las páginas del admin
* logout.php → Usa AuthController.php

**Criterios de éxito:**

* AdminLTE carga correctamente sin errores 404
* Login funciona y redirige al dashboard
* Logout funciona y redirige al login
* Header, sidebar y footer se muestran correctamente
* Diseño responsive funciona en móviles

**📋 METODOLOGÍA DE DESARROLLO COMPLETA PLAYMI**

* **🎯 CONTEXTO DEL PROYECTO:**
* Sistema web para entretenimiento en transporte público donde:
* Los administradores gestionan empresas, contenido y paquetes desde un panel web
* Los Raspberry Pi en buses emiten WiFi y hospedan contenido
* Los pasajeros escanean QR → se conectan al WiFi → acceden a portal tipo Netflix
* Contenido: películas, música (tipo YouTube) y juegos HTML5/JS
* **🏗️ ESTRUCTURA GENERAL DEL PROYECTO:**
* PLAYMI/
* ├── 📁 admin/ # Panel administrativo (FASE 1-2)
* ├── 📁 passenger-portal/ # Portal tipo Netflix para pasajeros
* ├── 📁 pi-system/ # Sistema completo para Raspberry Pi
* ├── 📁 content/ # Todo el contenido multimedia
* ├── 📁 companies/ # Datos específicos por empresa
* ├── 📁 packages/ # Paquetes generados para distribución
* ├── 📁 database/ # Base de datos y migraciones
* └── 📁 api/ # APIs compartidas
* **🚀 FASE 2.2: GESTIÓN DE CONTENIDO MULTIMEDIA**
* **Objetivo:** Crear sistema completo para subir, organizar y gestionar películas, música y juegos **Tiempo estimado:** 6-8 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── admin/
* │ ├── views/content/
* │ │ ├── index.php # Lista todo el contenido
* │ │ ├── movies.php # Gestión específica de películas
* │ │ ├── music.php # Gestión específica de música
* │ │ ├── games.php # Gestión específica de juegos
* │ │ ├── upload.php # Formulario de subida universal
* │ │ └── edit.php # Editar metadatos de contenido
* │ ├── api/content/
* │ │ ├── upload-movie.php # Subir archivo de película
* │ │ ├── upload-music.php # Subir archivo de música
* │ │ ├── upload-game.php # Subir juego HTML5 (ZIP)
* │ │ ├── process-video.php # Procesar y comprimir video
* │ │ ├── generate-thumbnail.php # Generar miniaturas automáticas
* │ │ └── delete-content.php # Eliminar contenido
* │ └── models/
* │ └── Content.php # Modelo para contenido (ya existe)
* ├── content/
* │ ├── movies/ # Almacén de películas
* │ │ ├── originals/ # Videos originales subidos
* │ │ ├── compressed/ # Videos comprimidos para streaming
* │ │ └── thumbnails/ # Miniaturas generadas
* │ ├── music/
* │ │ ├── audio/ # Archivos de audio
* │ │ ├── thumbnails/ # Carátulas/imágenes
* │ │ └── metadata/ # Información adicional
* │ └── games/
* │ ├── source/ # Archivos ZIP originales
* │ ├── extracted/ # Juegos descomprimidos
* │ └── thumbnails/ # Capturas de pantalla
* **FUNCIONALIDADES REQUERIDAS:**
* **Para Películas:**
* Subida de archivos MP4, AVI, MKV (máximo 2GB)
* Compresión automática para optimizar streaming
* Generación automática de thumbnails
* Metadatos: título, descripción, duración, género, clasificación
* Preview del video antes de publicar
* **Para Música:**
* Subida de archivos MP3, M4A, WAV
* Subida opcional de carátula/imagen
* Metadatos: título, artista, álbum, duración, género
* Reproducción previa antes de publicar
* **Para Juegos HTML5:**
* Subida de archivos ZIP que contengan index.html
* Validación automática de estructura del juego
* Generación de thumbnail desde screenshot
* Metadatos: título, descripción, categoría, instrucciones
* Preview del juego funcionando
* **CONEXIONES ENTRE ARCHIVOS:**
* views/content/\* → Usan ContentController.php → Usa Content.php
* api/content/\* → Procesan uploads y devuelven JSON
* upload-movie.php → Guarda en content/movies/ → Actualiza BD
* upload-game.php → Descomprime ZIP en content/games/extracted/
* process-video.php → Usa FFmpeg para comprimir videos
* **CRITERIOS DE ÉXITO:**
* Subida de los 3 tipos de contenido funciona sin errores
* Validaciones de formato y tamaño implementadas
* Compresión automática de videos funciona
* Generación de thumbnails automática
* Lista de contenido con filtros por tipo
* Edición de metadatos funcional
* Preview de contenido antes de publicar
* **🚀 FASE 2.3: GESTIÓN DE PAQUETES Y DISTRIBUCIÓN**
* **Objetivo:** Sistema para generar paquetes personalizados de contenido para cada empresa/Pi **Tiempo estimado:** 4-5 horas
* **ARCHIVOS A CREAR:**
* PLAYMI/
* ├── admin/
* │ ├── views/packages/
* │ │ ├── index.php # Lista de paquetes generados
* │ │ ├── generate.php # Formulario para generar paquete
* │ │ ├── customize.php # Personalizar contenido por empresa
* │ │ └── history.php # Historial de paquetes
* │ ├── api/packages/
* │ │ ├── generate-package.php # Crear paquete personalizado
* │ │ ├── download-package.php # Descargar paquete para Pi
* │ │ └── update-status.php # Actualizar estado de paquete
* │ └── models/
* │ └── Package.php # Modelo para paquetes (ya existe)
* ├── packages/
* │ ├── templates/
* │ │ ├── default-config.json # Configuración base del paquete
* │ │ ├── wifi-config.template # Template configuración WiFi
* │ │ └── portal-config.template # Template portal pasajeros
* │ └── generated/
* │ ├── [empresa-id]/ # Paquetes por empresa
* │ │ ├── content/ # Contenido seleccionado
* │ │ ├── config/ # Configuraciones específicas
* │ │ ├── portal/ # Portal web para pasajeros
* │ │ └── package.zip # Paquete final para descargar
* **FUNCIONALIDADES REQUERIDAS:**
* Seleccionar contenido específico por empresa/paquete
* Configurar branding personalizado (colores, logo)
* Generar configuración WiFi automática
* Crear QR codes personalizados por bus
* Empaquetar todo en ZIP descargable
* Seguimiento de estado: generando → listo → instalado
* Historial de paquetes por empresa
* **CONEXIONES ENTRE ARCHIVOS:**
* generate-package.php → Lee Content.php y Company.php
* generate-package.php → Copia contenido desde content/ a packages/generated/
* download-package.php → Sirve archivo ZIP desde packages/generated/
* **🚀 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