



AI VIDEO EDITOR

Plan de Trabajo Detallado & Hoja de Ruta Técnica

Editor web de video con IA · Nivel Enterprise · 2025–2026

 **Full Stack**
Next.js + Rust/WASM

 **IA Engine**
Python FastAPI + GPUs

 **Cloud**
AWS + Cloudflare

1. Definición del Stack Tecnológico

Stack seleccionado para máximo rendimiento, escalabilidad y capacidades de IA avanzada. Cada tecnología fue elegida evaluando alternativas en benchmarks de renderizado, comunidad activa y compatibilidad entre capas.

1.1 Frontend — Interfaz y Renderizado

Categoría	Tecnología & Justificación
Framework	Next.js 15 (App Router) — SSR/ISR + React 19 con Server Components para UI ultra-rápida
Estado Global	Zustand v5 — Estado atómico sin boilerplate; ideal para timeline con miles de nodos
Estado Servidor	TanStack Query v5 — Cache, sincronización optimista y revalidación de proyectos cloud
UI Components	shadcn/ui + Radix UI — Accesible, headless, totalmente personalizable
Animaciones	Motion (Framer Motion) + CSS @keyframes — 60fps garantizado en transiciones del timeline
Drag & Drop	dnd-kit — Mejor rendimiento que react-beautiful-dnd; soporte para timeline multitracks
Estilos	Tailwind CSS v4 + CSS Modules para componentes críticos de performance
Canvas/Renderizado	WebGPU (con fallback WebGL2) via wgpu-rs compilado a WASM

1.2 Procesamiento de Video — Core Engine

Categoría	Tecnología & Justificación
Codec Engine	FFmpeg.wasm 0.12+ — Decodificación/encoding completo en browser; compilado con Emscripten
WebAssembly	Rust → WASM (wasm-pack) — Lógica crítica de timeline, compositing y efectos en tiempo real
GPU Compute	WebGPU Compute Shaders — Filtros de color, blur, compositing de capas (10x más rápido que WebGL2)
Workers	SharedArrayBuffer + Atomics — Comunicación zero-copy entre main thread y workers dedicados
Video Decode	WebCodecs API — Decodificación nativa de H.264/H.265/AV1 sin overhead de JS

Streaming	ReadableStream + OPFS (Origin Private File System) — Acceso a archivos grandes sin cargar en RAM
Thumbnails	OffscreenCanvas en Worker — Generación asíncrona de previews sin bloquear UI

1.3 Backend & APIs

Categoría	Tecnología & Justificación
API Gateway	Next.js Route Handlers + tRPC v11 — Type-safety end-to-end sin codegen manual
IA Microservicio	Python FastAPI — Hosts de modelos ML; async nativo; compatible con PyTorch/HuggingFace
Cola de Jobs	BullMQ (Redis) — Jobs de renderizado/export; reintentos automáticos; progress real-time
WebSockets	Partykit / Socket.io — Colaboración en tiempo real (cursos, cambios de timeline)
Auth	Auth.js v5 (NextAuth) — OAuth (Google, GitHub) + JWT + sessions edge-compatible
Storage	AWS S3 + CloudFront — Assets de usuario; signed URLs; CDN global para playback rápido
Export Cloud	AWS Lambda + ECS Fargate + FFmpeg — Renderizado serverless con GPU spot instances

1.4 Base de Datos

Categoría	Tecnología & Justificación
Principal	PostgreSQL 16 via Neon (serverless) — Proyectos, usuarios, assets; branching para dev/staging
Cache & Sessions	Redis 7 (Upstash) — Sessions, rate limiting, cola BullMQ, caché de metadatos
Búsqueda	Meilisearch — Búsqueda full-text en biblioteca de assets, plantillas y stickers
Analytics	ClickHouse (cloud) — Telemetría de uso, métricas de performance de exportaciones
ORM	Drizzle ORM — Type-safe, ligero, compatible con edge runtimes; migraciones automáticas

1.5 Infraestructura & DevOps

Categoría	Tecnología & Justificación
Deploy Frontend	Vercel Edge Network — Zero-config, CDN global, previews por PR
Deploy IA	Modal.com o Replicate — GPUs on-demand para inferencia; cold start < 2s
CI/CD	GitHub Actions — Tests, linting, preview deploys, push to production
Monorepo	Turborepo + pnpm workspaces — apps/web, apps/api-ai, packages/shared
Monitoreo	OpenTelemetry + Sentry + Datadog — Traces distribuidos end-to-end
IaC	Pulumi (TypeScript) — Infraestructura como código type-safe; stacks dev/staging/prod

2. Desglose de Módulos — Backlog de Desarrollo

El proyecto se divide en 6 fases progresivas. Cada fase produce un entregable funcional y testeado. Tiempo estimado total: 8–12 meses con equipo de 3–5 developers.

FASE 0 — Setup & Fundaciones (Semanas 1–2)

Objetivo

Monorepo configurado, CI/CD operativo, design system base, entorno local reproducible.

- ▶ Inicializar monorepo Turborepo + pnpm workspaces
- ▶ Configurar Next.js 15 con TypeScript strict mode + ESLint + Prettier
- ▶ Setup Tailwind CSS v4 + shadcn/ui + sistema de tokens de diseño
- ▶ Configurar Drizzle ORM + Neon PostgreSQL + Upstash Redis
- ▶ Implementar Auth.js v5 con Google OAuth
- ▶ GitHub Actions: lint → test → build → deploy preview
- ▶ Docker Compose para desarrollo local (postgres, redis, meilisearch)
- ▶ Storybook para componentes UI aislados

FASE 1 — Core Engine: Timeline & Capas (Semanas 3–8)

Objetivo

Editor de timeline funcional con reproducción en tiempo real, capas múltiples, cortes y transiciones básicas. Este es el corazón de la aplicación.

1A — Timeline Engine (Rust/WASM)

- ▶ Diseñar estructura de datos del Timeline: pistas (tracks), clips, keyframes
- ▶ Compilar core de timeline en Rust → WASM con wasm-pack
- ▶ Implementar modelo de datos inmutable con historial de cambios (undo/redo)
- ▶ API WASM expuesta: addClip(), moveClip(), trimClip(), splitClip()
- ▶ Playhead con scrubbing de alta precisión (sub-frame accuracy)

1B — Reproductor de Video

- ▶ Integrar WebCodecs API para decodificación nativa de frames
- ▶ Implementar VideoRenderer con WebGPU para compositing de capas en tiempo real
- ▶ Sistema de caché de frames decoded (LRU cache con límite de memoria)
- ▶ Fallback a WebGL2 para browsers sin soporte WebGPU
- ▶ Preview a 30fps con degradación inteligente según capacidad del dispositivo

1C — UI del Timeline

- ▶ Componente Track con dnd-kit (drag clips entre pistas)
- ▶ Zoom temporal: de microsegundos a horas en el eje X
- ▶ Multi-select de clips con Shift+Click y Ctrl+A
- ▶ Atajos de teclado completos (J/K/L para playback, C para corte, etc.)
- ▶ Waveform de audio generada en Worker con OffscreenCanvas
- ▶ Thumbnails de video en clips (generadas via FFmpeg.wasm en background)

1D — Capas y Compositing

- ▶ Soporte para tracks: Video, Audio, Texto, Imagen, Sticker, Efecto
- ▶ Mezclado de capas con blend modes (Normal, Multiply, Screen, Overlay...)
- ▶ Opacidad y transformaciones (posición, escala, rotación) con keyframes
- ▶ Inspector de propiedades lateral con valores numéricos editables

FASE 2 — Procesamiento de Medios (Semanas 9–14)

Objetivo

Import de medios robusto, manejo de archivos grandes con OPFS, y sistema de exportación local y cloud.

2A — Import & Gestión de Archivos

- ▶ Drag & drop de archivos desde el sistema operativo
- ▶ Integración con File System Access API para acceso directo a archivos locales
- ▶ OPFS (Origin Private File System) para staging de archivos grandes sin RAM
- ▶ Extracción de metadatos: resolución, fps, codec, duración, bitrate
- ▶ Transcoding automático de formatos incompatibles via FFmpeg.wasm
- ▶ Progress bar de procesamiento con cancelación
- ▶ Límite de tamaño configurable (plan Free: 2GB, Pro: sin límite)

2B — Exportación Local

- ▶ FFmpeg.wasm encoder: H.264, H.265, AV1, WebM/VP9
- ▶ Presets: TikTok (1080x1920), YouTube (2160p), Instagram (1:1, 4:5, 9:16)
- ▶ Control de bitrate, resolución, fps y formato de audio (AAC, MP3, Opus)
- ▶ Exportación por segmentos para proyectos largos (evita timeout de browser)
- ▶ Progress tracking con tiempo estimado restante

2C — Exportación Cloud (Renderizado Remoto)

- ▶ Job encolado a BullMQ cuando el proyecto supera 10 min o 4K
- ▶ Worker Lambda o ECS Fargate con FFmpeg nativo (GPU-accelerated: NVENC)
- ▶ Progreso en tiempo real via WebSocket
- ▶ Archivo final uploadado a S3 con signed URL de descarga (72h)
- ▶ Notificación por email al completar

FASE 3 — Módulo de IA (Semanas 15–24)

Objetivo

Funciones de IA que diferencian el producto: eliminación de fondo, subtítulos automáticos, generación de contenido y filtros inteligentes.

3A — Background Removal & Segmentación

- ▶ Modelo: SAM 2 (Segment Anything Model v2) de Meta — estado del arte en segmentación
- ▶ Servicio FastAPI en GPU (Modal.com) con procesamiento frame-by-frame o batch
- ▶ Resultado: máscara alfa PNG por frame, aplicada en compositing WebGPU
- ▶ Opción de refinamiento manual con brush tool (add/subtract de máscara)
- ▶ Soporte para segmentación de objetos específicos (no solo fondo)

3B — Subtítulos Automáticos (ASR)

- ▶ Modelo: Whisper large-v3 de OpenAI vía FastAPI o Groq API (ultra-rápido)
- ▶ Detección automática de idioma (99+ idiomas)
- ▶ Timestamps por palabra para sincronización perfecta con timeline
- ▶ Editor de subtítulos integrado: click en texto → jump al frame
- ▶ Exportación: SRT, VTT, ASS/SSA con estilos personalizados
- ▶ Traducción automática a 50+ idiomas vía DeepL o NLLB-200

3C — Text-to-Speech & Voice Cloning

- ▶ TTS de alta calidad: ElevenLabs API o Coqui XTTS-v2 (open source)
- ▶ Clonación de voz: grabar 30 segundos → clonar voz para narración
- ▶ Generación de narración desde script de texto con emoción controlable
- ▶ Sincronización automática de audio generado con clips de texto en timeline

3D — Filtros Generativos & Efectos IA

- ▶ Style Transfer en tiempo real: aplicar estilo artístico a clips de video
- ▶ Color grading IA: analizar referencia visual → aplicar LUT equivalente
- ▶ Mejora de video: upscaling 2x/4x con Real-ESRGAN para videos de baja calidad
- ▶ Estabilización de video: DAIN o RIFE para interpolación de frames y smooth motion

- ▶ Auto-reencuadre: detección de sujeto + crop automático para cambio de aspect ratio

FASE 4 — Assets & Biblioteca (Semanas 19–22)

Objetivo

Biblioteca completa de recursos listos para usar: música, efectos de sonido, stickers animados, plantillas y fuentes.

- ▶ Biblioteca de música sin derechos (integración con Pixabay/Freesound API o catálogo propio)
- ▶ Efectos de sonido categorizados y buscables con Meilisearch
- ▶ Stickers animados: Lottie (JSON) y APNG con motor de renderizado optimizado
- ▶ Plantillas de proyecto: estructuras prediseñadas por categoría (Reels, Vlogs, Presentaciones)
- ▶ Fuentes tipográficas: integración con Google Fonts API + fuentes premium opcionales
- ▶ Transiciones premium: 50+ transiciones animadas con preview en hover
- ▶ Upload de assets custom: el usuario sube su música, logos, fuentes
- ▶ Sistema de favoritos y carpetas personales de organización

FASE 5 — Colaboración & Proyecto Cloud (Semanas 23–28)

Objetivo

Sistema de proyectos cloud con versionado, compartir y colaboración multiusuario en tiempo real.

- ▶ Guardado automático de proyectos en PostgreSQL (JSON del estado del timeline)
- ▶ Sistema de versiones: snapshot por cada 5 minutos + versiones manuales nombradas
- ▶ Compartir proyecto: link público de sólo lectura o edición con permisos
- ▶ Colaboración en tiempo real: múltiples usuarios editando el mismo proyecto (via Partykit)
- ▶ Dashboard de proyectos con búsqueda, filtros y previsualización de thumbnail
- ▶ Papelera con recuperación (30 días) y limpieza automática de storage
- ▶ Exportación/importación de proyectos como archivo .aivep (JSON comprimido)

3. Arquitectura de Datos

3.1 Manejo de Archivos de Video Grandes

El mayor reto de un editor web es manejar archivos de varios GB sin saturar la RAM del navegador. La estrategia es mantener los archivos en disco y acceder solo a los frames necesarios.

📁 Estrategia: OPFS (Origin Private File System)

Los archivos se copian al OPFS al importarlos. OPFS proporciona acceso binario sincrónico desde Workers, sin pasar por la red ni ocupar la RAM heap de JavaScript. El video nunca se carga completo en memoria.

Etapa	Estrategia
Import	Usuario arrastra archivo → Web Worker lo copia a OPFS (streaming, sin bloquear UI)
Decode	WebCodecs API decodifica solo los frames visibles en la ventana actual del timeline
Frame Cache	LRU Cache de frames decoded en VideoFrames (límite: 200MB configurable)
Thumbnails	FFmpeg.wasm extrae 1 frame/segundo → guardado como JPEG en OPFS para previews rápidos
Audio	WebAudio API con AudioBuffer cacheado por segmento de 10 segundos
Cloud Upload	Archivos subidos a S3 en chunks de 5MB (multipart upload) con progreso
CDN Playback	CloudFront signed URL con range requests para streaming parcial desde S3

3.2 Estructura de Base de Datos

Modelo relacional en PostgreSQL. El estado del timeline se almacena como JSON (JSONB) para flexibilidad, con índices en los campos más consultados.

```
-- USUARIOS
users { id, email, name, avatar_url, plan, storage_used_bytes, created_at }

-- PROYECTOS
projects {
  id, user_id, title, description,
```

```

  thumbnail_url,           -- S3 URL del frame de preview
  duration_ms,            -- Duración total en milisegundos
  resolution,              -- "1920x1080", "1080x1920", etc.
  fps,                     -- 24, 30, 60
  timeline_state JSONB,   -- Estado completo del editor serializado
  created_at, updated_at, deleted_at
}

-- VERSIONES DE PROYECTO (snapshots)
project_versions {
  id, project_id, version_number,
  label,           -- Nombre opcional ("Versión final")
  timeline_state JSONB,
  created_at
}

-- ASSETS DE USUARIO
user_assets {
  id, user_id, type,      -- "video" | "audio" | "image" | "font"
  filename, mime_type,
  s3_key, cdn_url,
  size_bytes, duration_ms,
  metadata JSONB,        -- fps, resolution, codec, etc.
  created_at
}

-- COLABORADORES DE PROYECTO
project_collaborators {
  project_id, user_id,
  role,           -- "owner" | "editor" | "viewer"
  invited_at, accepted_at
}

-- JOBS DE EXPORTACIÓN
export_jobs {
  id, project_id, user_id,
  status,           -- "queued" | "processing" | "done" | "failed"
  preset,           -- "mp4-1080p" | "mp4-4k" | "webm" etc.
  progress_pct,
  output_s3_key, download_url, download_expires_at,
  error_message,
  started_at, completed_at, created_at
}

```

3.3 Estructura del Estado del Timeline (JSONB)

El timeline_state se serializa como un JSON estructurado. Permite guardar y restaurar el estado completo del editor.

```
{
  "version": "1.0",
  "duration": 120000,           // ms
  "resolution": { "w": 1920, "h": 1080 },
  "fps": 30,
}
```

```
"tracks": [
  {
    "id": "track_01",
    "type": "video",
    "muted": false,
    "clips": [
      {
        "id": "clip_01",
        "assetId": "asset_abc",
        "startMs": 0,
        "endMs": 5000,
        "trimStartMs": 500, // Punto de entrada en el asset original
        "trimEndMs": 5500, // Punto de salida en el asset original
        "effects": [
          { "type": "color_grade", "params": { "brightness": 0.1 } }
        ],
        "keyframes": {
          "opacity": [ { "t": 0, "v": 0 }, { "t": 500, "v": 1 } ]
        },
        "transition": {
          "type": "crossfade", "durationMs": 300
        }
      }
    ],
    "subtitles": [...],
    "audioMix": { "masterVolume": 1.0 }
  }
]
```

4. Funciones "Plus" — Mejor que CapCut

Tres capacidades innovadoras basadas en modelos de IA de 2024–2025 que CapCut no domina todavía, creando una ventaja competitiva real.

⚡ PLUS #1 — Generación de Escenas por Prompt (Text-to-Video Integrado)

¿Por qué es diferente?

CapCut ofrece generación de imágenes básica. Nosotros integramos generación de video completo (3–10 segundos) directamente en el timeline como un clip nativo, combinable con footage real.

Implementación técnica:

- ▶ Integrar Runway Gen-3 Alpha API o Kling AI API (mejores modelos text-to-video 2025)
- ▶ UI: click derecho en gap del timeline → "Generar clip con IA" → prompt de texto
- ▶ Opciones: duración (3/5/10s), estilo (realista/animado/cinemático), seed para reproducibilidad
- ▶ El clip generado aparece directamente en el timeline como un asset de video estándar
- ▶ Inpainting de video: seleccionar región del frame + prompt → el modelo la reemplaza manteniendo movimiento y contexto
- ▶ Camera controls: prompt de movimiento de cámara ("slow zoom in", "pan left")

Diferenciador clave

Integración nativa en el timeline, no como feature separada. El usuario combina clips reales y generados sin fricciones, con previsualización en tiempo real.

⚡ PLUS #2 — Colaboración Multiusuario en Tiempo Real

¿Por qué es diferente?

CapCut no tiene colaboración en tiempo real. Nuestra solución permite que 2–10 personas editen simultáneamente con sincronización de microsegundos y resolución de conflictos automática.

Implementación técnica:

- ▶ CRDT (Conflict-free Replicated Data Types) con Yjs — el estándar de colaboración en tiempo real
- ▶ Partykit como servidor de WebSockets para sincronización (edge-deployed, latencia < 50ms)

- ▶ Cursos de usuario en tiempo real en el timeline (avatar + nombre flotante)
- ▶ Selección de clips compartida: ver qué está editando cada colaborador
- ▶ Sistema de "locks" suaves: advertencia si dos usuarios editan el mismo clip
- ▶ Chat integrado con reactions y comentarios anclados a timestamps del video
- ▶ Historial de cambios por usuario: ver quién hizo qué y cuándo (con avatar)
- ▶ Modo "Revisor": el cliente ve el proyecto y deja comentarios en frames específicos

Diferenciador clave

Flujo de trabajo agencias/equipos: director edita en el timeline mientras cliente deja feedback en frames específicos en tiempo real. Elimina ciclos de revisión por email.

⚡ PLUS #3 — Director IA (AI Editing Autopilot)

¿Por qué es diferente?

CapCut tiene "auto-edit" básico. Nuestro Director IA usa LLMs + modelos multimodales para entender el contenido semánticamente y crear un montaje con narrativa coherente, ajustado al ritmo de la música.

Implementación técnica:

- ▶ Análisis multimodal de todos los clips: Gemini 1.5 Pro / GPT-4o Vision describen cada escena
- ▶ Análisis de audio: detección de beats, BPM, estructura musical (intro/verso/coro/outro)
- ▶ LLM crea un "guión de montaje": ordena clips por narrativa, asigna duraciones según beat
- ▶ Detección de "best moments": identifica faces con buenas expresiones, acción dinámica, frases clave
- ▶ Input del usuario: "Quiero un video de 60s estilo energético para Instagram de esta boda"
- ▶ Output: timeline pre-armado listo para ajustes finales, con transiciones y música sincronizada
- ▶ Mejora iterativa: "Hazlo más corto", "Añade más close-ups", "Cambia el ritmo en el minuto 1"

Diferenciador clave

Reduce de 3 horas a 10 minutos el primer corte de un video. El editor humano refina; la IA hace el trabajo pesado inicial. Ideal para creadores de contenido de alto volumen.

5. Guía de Implementación — Prompts de Ejecución para IA Programadora

Serie de prompts listos para copiar y pegar en tu agente de programación (Cursor, Aider, Claude Code, Copilot Workspace). Ejecutar en orden. Cada prompt produce código funcional y testeado.

⚠️ Instrucciones de uso

Copia cada prompt completo y pégalo en tu IA programadora. Espera a que complete cada tarea antes de pasar al siguiente. Si la IA pide clarificación, usa el contexto de este documento para responder.

⚡ PROMPT 01: Setup del Monorepo y Entorno Base

Crear un monorepo con Turborepo y pnpm workspaces con la siguiente estructura:

```
apps/
  web/           → Next.js 15 + TypeScript strict
  api-ai/        → FastAPI Python para servicios de IA
packages/
  ui/            → Componentes compartidos con shadcn/ui
  db/            → Drizzle ORM + esquemas PostgreSQL
  timeline-wasm/ → Proyecto Rust compilado a WASM
```

Configura:

1. turbo.json con pipelines: build, dev, test, lint
2. pnpm-workspace.yaml
3. Next.js 15 con App Router, TypeScript strict, Tailwind v4
4. ESLint + Prettier con reglas para monorepo
5. Husky + lint-staged para pre-commit hooks
6. Docker Compose con: postgres:16, redis:7, meilisearch:latest
7. .env.example con todas las variables necesarias documentadas

Al terminar, el comando `pnpm dev` debe levantar Next.js en :3000 y `docker compose up` debe levantar los servicios de infraestructura.

⚡ PROMPT 02: Base de Datos y Autenticación

En el package packages/db, implementa el esquema completo con Drizzle ORM:

1. Crea las tablas: users, projects, project_versions, user_assets, project_collaborators, export_jobs
(usa el esquema SQL definido en el documento de arquitectura)
2. Configura la conexión a Neon PostgreSQL con connection pooling
3. Crea migraciones iniciales con `drizzle-kit generate`

4. En `apps/web`, implementa `Auth.js` v5 con:
 - Google OAuth provider
 - GitHub OAuth provider
 - Adapter de Drizzle para persistir sessions en PostgreSQL
 - Middleware de protección de rutas (todo en `/dashboard/*` requiere auth)
5. Crea las páginas:
 - `/login` → Página de login con botones OAuth
 - `/dashboard` → Layout protegido con sidebar
 - `/dashboard/projects` → Lista de proyectos del usuario
6. Implementa el seed script para datos de desarrollo

Tests requeridos: test de conexión DB, test de flujo de autenticación con mocks.

⚡ PROMPT 03: Timeline Engine en Rust/WASM

Crea el proyecto Rust en `packages/timeline-wasm`:

1. Inicializa con `wasm-pack` y `wasm-bindgen`
2. Implementa las siguientes estructuras de datos en Rust:
 - `TimelineState` { `tracks`, `duration_ms`, `fps`, `resolution` }
 - `Track` { `id`, `type` (video/audio/text), `clips`, `muted`, `volume` }
 - `Clip` { `id`, `asset_id`, `start_ms`, `end_ms`, `trim_start_ms`, `trim_end_ms`, `transform`, `keyframes`, `effects` }
 - `Keyframe` { `time_ms`, `value`, `easing` }
3. Exponer via `#[wasm_bindgen]` las funciones:
 - `new_timeline(fps, width, height)` → `TimelineState`
 - `add_clip(state, track_id, clip)` → `TimelineState`
 - `move_clip(state, clip_id, new_start_ms, new_track_id)` → `TimelineState`
 - `trim_clip(state, clip_id, trim_start_ms, trim_end_ms)` → `TimelineState`
 - `split_clip(state, clip_id, split_at_ms)` → `TimelineState`
 - `delete_clip(state, clip_id)` → `TimelineState`
 - `undo(history)` → `TimelineState`
 - `redo(history)` → `TimelineState`
4. Implementar sistema de historial inmutable (Vec de states)
5. Compilar a WASM y crear los bindings TypeScript
6. Tests unitarios en Rust + tests de integración en TypeScript

⚡ PROMPT 04: Reproductor de Video con WebCodecs + WebGPU

En `apps/web/src/components/editor/player`, implementa el reproductor:

1. VideoPlayer component con:
 - Canvas WebGPU para renderizado (con fallback a WebGL2)
 - Controles: play/pause, seek, volumen, fullscreen
 - Playhead sincronizado con el timeline (bidireccional)
2. VideoDecoder (Web Worker) usando WebCodecs API:
 - Leer frames desde OPFS con FileSystemSyncAccessHandle
 - Decode de H.264/H.265 con VideoDecoder
 - Transferir VideoFrames al main thread via transferable objects
 - LRU cache de frames decodificados (máx 200MB)
3. Compositor WebGPU:
 - Shader WGLSL para componer múltiples VideoFrames por frame
 - Soporte blend modes: Normal, Multiply, Screen, Overlay
 - Aplicar transform (posición, escala, rotación, opacidad)
4. AudioPlayer usando WebAudio API:
 - Sincronización precisa con video (usando AudioContext.currentTime)
 - Mezcla de múltiples tracks de audio
5. Performance target: inicio de playback < 100ms, seek < 200ms

Incluye performance tests y documentación del pipeline de renderizado.

⚡ PROMPT 05: UI del Timeline (Componente Principal)

Implementa el componente Timeline en `apps/web/src/components/editor/timeline`:

1. TimelineContainer:
 - Regla de tiempo (ruler) con marks adaptativas al nivel de zoom
 - Playhead draggable con snap a frames
 - Zoom: rueda del ratón en el timeline (range: 10px/s a 1000px/s)
 - Scroll horizontal e inertia scroll
2. TrackList:
 - Tracks de Video, Audio, Texto, Sticker
 - Agregar/eliminar tracks con botones
 - Reordenar tracks con drag (dnd-kit)
 - Mute/Solo por track, control de volumen
3. ClipComponent:
 - Thumbnail del video en el clip
 - Waveform del audio (generada en Worker)
 - Trim handles: arrastrar bordes para recortar
 - Drag para mover entre posiciones y tracks
 - Selección múltiple (Shift+Click, Ctrl+A)
 - Context menu: Cortar, Copiar, Eliminar, Dividir, Propiedades
4. Atajos de teclado:

Space=Play/Pause, C=Cortar, Delete=Eliminar,
Ctrl+Z=Undo, Ctrl+Y=Redo, J/K/L=Retroceder/Pause/Avanzar

5. Estado global con Zustand (useTimelineStore)

Target de rendimiento: 60fps de scroll/zoom con 100+ clips en el timeline.

⚡ PROMPT 06: Sistema de Import y OPFS

Implementa el sistema de importación de medios:

1. MediaImporter component:

- Drag & drop zone en el área del editor
- File picker con filtros por tipo (video, audio, imagen)
- Progress bar por archivo con cancelación

2. FileProcessingWorker (Web Worker):

- Copiar archivo al OPFS usando FileSystemSyncAccessHandle
- Extraer metadatos con FFmpeg.wasm (resolución, fps, duración, codec)
- Generar thumbnails: 1 frame/segundo guardado como JPEG en OPFS
- Transcoding si el formato no es compatible con WebCodecs
- Reportar progreso al main thread via postMessage

3. MediaLibrary component (panel lateral):

- Grid de assets importados con thumbnails
- Búsqueda por nombre
- Drag desde library al timeline para añadir clips
- Menú contextual: Eliminar, Renombrar, Ver propiedades

4. OPFS Manager:

- Gestión del storage (quota, limpieza de archivos huérfanos)
- Índice de archivos en memoria (Map<assetId, opfsHandle>)

5. Upload a S3:

- Multipart upload con progress
- Reanudar uploads interrumpidos

⚡ PROMPT 07: Microservicio de IA (FastAPI)

Crea el servicio en apps/api-ai con FastAPI:

1. Setup:

- FastAPI con async endpoints
- Poetry para gestión de dependencias
- Dockerfile con CUDA support para GPU
- Autenticación via JWT (verificar tokens del frontend)

2. Endpoint /ai/remove-background:

- Input: video file (upload) o S3 URL
- Modelo: rembg con modelo u2net o SAM2 para video
- Output: video con canal alpha (WebM VP9 con transparencia)
- Job asíncrono con progress via SSE (Server-Sent Events)

3. Endpoint /ai/transcribe:
 - Input: audio file o S3 URL
 - Modelo: Whisper large-v3 via faster-whisper
 - Output: JSON con segmentos { start, end, text, words: [{word, start, end}] }
 - Detección automática de idioma
4. Endpoint /ai/generate-video:
 - Input: prompt text, duration, style, seed
 - Integración con Runway Gen-3 o Kling API
 - Output: video file URL en S3
5. Queue de jobs con Redis (rq library):
 - Jobs pesados van a la queue (no bloquean HTTP)
 - Endpoint /ai/jobs/{job_id}/status para polling
6. Tests con pytest + mocks de modelos ML

⚡ PROMPT 08: Sistema de Exportación

Implementa el pipeline completo de exportación:

1. ExportDialog component:
 - Presets: TikTok 9:16, YouTube 16:9 1080p/4K, Instagram 1:1
 - Configuración avanzada: codec, bitrate, fps, resolución custom
 - Estimación de tamaño de archivo
 - Toggle: exportar localmente o en la nube
2. Exportación Local (browser):
 - FFmpeg.wasm encoder con los parámetros seleccionados
 - Renderizar frames secuencialmente usando el compositor WebGPU
 - Progress bar con tiempo estimado
 - Descarga directa con anchor download
3. Exportación Cloud:
 - Endpoint POST /api/exports que crea un export_job en DB
 - Job encolado en BullMQ
 - Worker: descarga assets de S3, ejecuta FFmpeg nativo, sube resultado
 - Progress via WebSocket (useExportProgress hook)
 - Notificación en UI + email al completar
4. Historial de exportaciones en /dashboard/exports:
 - Lista de exports con estado, tamaño, fecha
 - Botón de descarga (signed URL con 72h de validez)
 - Re-exportar con mismos o diferentes settings

⚡ PROMPT 09: Subtítulos Automáticos y Editor

Implementa el sistema de subtítulos:

1. Auto-transcripción:
 - Botón "Generar subtítulos" en la toolbar del editor
 - Llama a /ai/transcribe con el audio del proyecto
 - Muestra progress y resultado en un panel lateral
2. SubtitleTrack en el Timeline:
 - Track especial tipo "subtitle" debajo de los tracks de video
 - Cada subtítulo aparece como un clip con el texto visible
 - Editable: doble click para editar texto in-place
 - Drag para mover temporalmente, handles para ajustar duración
3. SubtitleEditor panel:
 - Lista de subtítulos con timestamps editables
 - Click en subtítulo → salta al frame en el player
 - Importar SRT/VTT externos
 - Exportar como SRT, VTT o ASS con estilos
4. Estilos de subtítulos:
 - Fuente, tamaño, color, outline, background semitransparente
 - Posición en el frame (superior/central/inferior)
 - Animación de entrada (fade, slide, karaoke por palabra)
5. Renderizado en el compositor:
 - Los subtítulos se renderizan como una capa de texto
 - Soporte Unicode completo (árabe, chino, etc.)

⚡ PROMPT 10: Deploy y Configuración de Producción

Configura el entorno de producción completo:

1. Vercel (Frontend):
 - Conectar repositorio GitHub a Vercel
 - Variables de entorno de producción
 - Headers de seguridad: CSP, CORS, COEP/COOP (requeridos para SharedArrayBuffer)
 - next.config.ts con headers correctos para WASM y SharedArrayBuffer:
"Cross-Origin-Opener-Policy": "same-origin"
"Cross-Origin-Embedder-Policy": "require-corp"
2. AWS Setup:
 - S3 bucket con CORS configurado para el dominio de producción
 - CloudFront distribution con signed URLs
 - IAM roles con permisos mínimos necesarios
3. Neon PostgreSQL:
 - Proyecto de producción con connection pooling
 - Branching: main (prod), dev (desarrollo)
 - Backup automático diario
4. Modal.com (IA Service):
 - Deploy del FastAPI como Modal App
 - GPU: A10G para inferencia

- Auto-scaling configurado
5. Monitoreo:
- Sentry para error tracking (frontend + backend)
 - Upstash Redis para BullMQ y rate limiting
 - Health check endpoints en /api/health
6. CI/CD final:
- GitHub Actions: push a main → tests → deploy a Vercel + Modal
 - Preview deploys en cada PR
 - Rollback automático si health check falla

PROMPT BONUS: Validación Final de MVP

Revisa toda la aplicación y valida que los siguientes criterios del MVP estén cumplidos:

- Import de video MP4/MOV funciona sin crashes con archivos de 1GB+
- Timeline con 10+ clips en 2+ tracks se scrollea/zooma a 60fps
- Undo/redo funciona correctamente en 20+ operaciones
- Exportación local de 60s de video en menos de 3 minutos
- Transcripción automática de 5 minutos de audio en menos de 60s
- Login/logout y persistencia de proyectos en la nube funciona
- Lighthouse score > 90 en Performance, Accessibility, Best Practices
- Sin memory leaks después de 30 minutos de uso continuo

6. Hoja de Ruta — Timeline de Desarrollo

Período	Hito
Semanas 1–2	📝 FASE 0: Setup monorepo, CI/CD, DB, Auth, deploy inicial
Semanas 3–5	⚙️ FASE 1A-B: Core engine WASM + Reproductor WebCodecs/WebGPU
Semanas 6–8	📋 FASE 1C-D: UI del Timeline, capas, compositing, atajos
Semanas 9–11	📁 FASE 2A-B: Import de medios, OPFS, exportación local
Semanas 12–14	☁️ FASE 2C: Exportación cloud, BullMQ workers, S3
Semanas 15–17	🎥 FASE 3A-B: Background removal (SAM2) + Subtítulos (Whisper)
Semanas 18–20	🎙️ FASE 3C-D: TTS/Voice cloning + Filtros IA + Auto-reencuadre
Semanas 19–22	🎵 FASE 4: Biblioteca de assets, música, stickers, plantillas
Semanas 21–22	⚡ PLUS #2: Colaboración real-time (Yjs + Partykit)
Semanas 22–24	⚡ PLUS #1: Generación de video por prompt (Runway/Kling API)
Semanas 24–26	⚡ PLUS #3: Director IA (análisis multimodal + montaje automático)
Semanas 27–28	🔗 FASE 5: QA final, performance, seguridad, beta pública

💡 Nota del Arquitecto

Las fases 4 y PLUS pueden solaparse con las fases 3. El equipo se divide: un developer en features de IA mientras otro avanza en assets y colaboración. El timeline asume 2–3 developers full-time. Con 5 developers se puede reducir a 5–6 meses.

Recursos y Documentación Recomendada

Recurso	URL
WebCodecs API	developer.chrome.com/docs/web-platform/best-practices/webcodecs
WebGPU	gpuweb.github.io/gpuweb/explainer
FFmpeg.wasm	ffmpegwasm.netlify.app/docs
wasm-pack (Rust)	rustwasm.github.io/wasm-pack

Drizzle ORM	orm.drizzle.team/docs
Auth.js v5	authjs.dev/getting-started
Yjs (CRDT)	docs.yjs.dev
BullMQ	docs.bullmq.io
Modal.com (GPU)	modal.com/docs/guide
Runway Gen-3 API	dev.runwayml.com/docs

AI Video Editor — Plan Técnico Maestro · Generado como guía de desarrollo