MarketplaceAI - Arquitectura del Sistema

Documento de Arquitectura Técnica v1.1.0-only-avax

Fecha: 27 Noviembre 2025
Versión: 1.1.0-only-avax
Autor: Equipo MarketplaceAI
Estado: Limpieza Fase 3 completada - Sin código legacy Sui

Tabla de Contenidos

---

1. Visión General

1.1 Descripción

MarketplaceAI es un marketplace descentralizado para modelos de IA construido sobre **Avalanche**. Permite publicar, licenciar y monetizar modelos de IA mediante licencias NFT (perpetuas o suscripción).

1.2 Características Principales

- **Publicación de Modelos**: Wizard de 5 pasos con metadata rica
- **Licenciamiento NFT**: Perpetuas o suscripción mensual
- **Almacenamiento IPFS**: Artifacts y metadata via Pinata
- **Multi-idioma**: Inglés y Español completo
- **Wallet Integration**: MetaMask, WalletConnect, Core Wallet

1.3 Blockchain Soportada

Red	Chain ID	Símbolo	Uso
Avalanche Fuji	43113	AVAX	Testnet
Avalanche Mainnet	43114	AVAX	Producción

---

2. Stack Tecnológico

2.1 Frontend

Tecnología	Versión	Propósito
Next.js	14.2.33	Framework React con SSR/SSG
React	18.x	Biblioteca UI
TypeScript	5.9.3	Tipado estático
Material UI	5.18.0	Sistema de diseño
Emotion	11.14.x	CSS-in-JS
Zustand	4.5.7	Estado global
SWR	2.2.0	Data fetching y cache
React Query	5.90.5	Server state management

2.2 Blockchain

Tecnología	Versión	Propósito
Wagmi	2.19.4	React hooks para Ethereum
Viem	2.38.6	Cliente Ethereum ligero
RainbowKit	2.0.8	UI de conexión de wallet
Ethers.js	6.15.0	Interacción con contratos

2.3 Almacenamiento

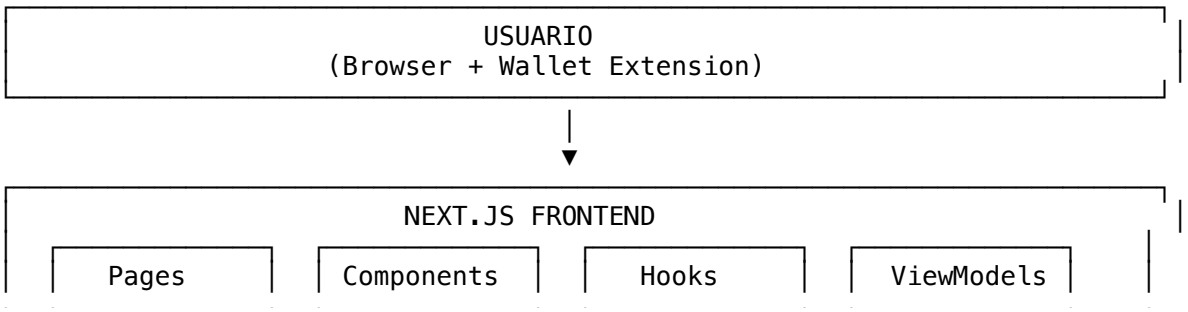
Tecnología	Propósito
IPFS	Almacenamiento descentralizado
Pinata	Gateway y pinning IPFS
PostgreSQL	Cache de metadata (Neon)
Prisma	ORM para PostgreSQL

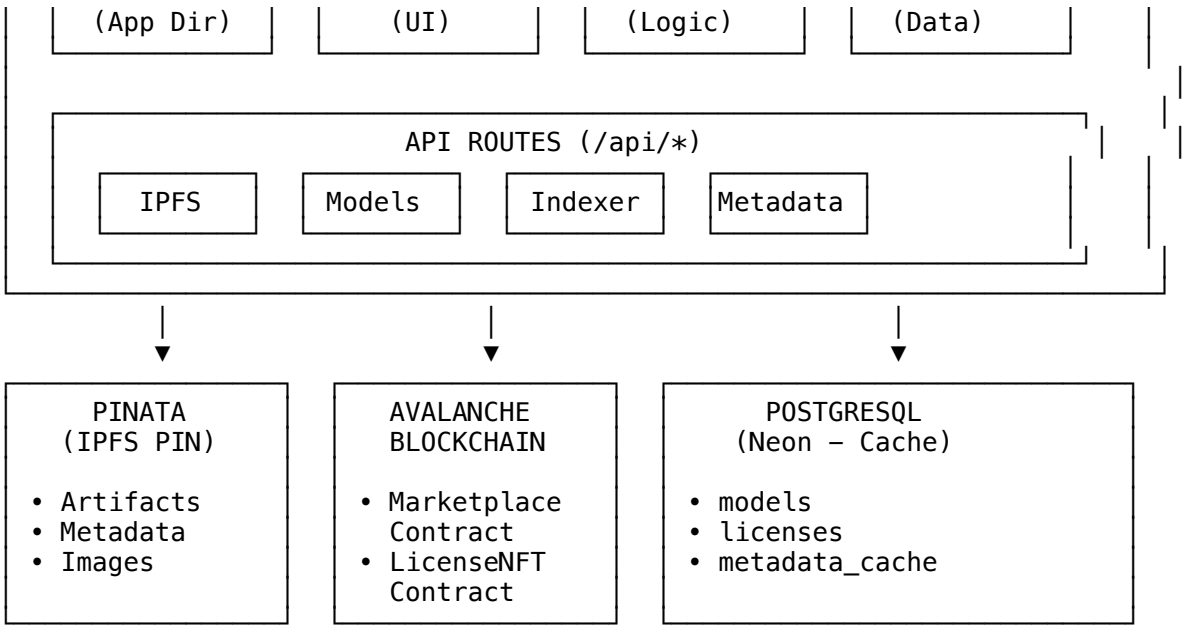
2.4 Internacionalización

Tecnología	Versión	Propósito
next-intl	4.5.0	i18n para Next.js

---

3. Arquitectura de Alto Nivel





---

4. Estructura del Proyecto

```
marketplaceai-frontend/
├── src/
│   ├── abis/                    # ABIs de Smart Contracts
│   │   ├── LicenseNFT.json      # ERC-721 para licencias
│   │   └── Marketplace.json     # Contrato principal
│   │
│   ├── app/                     # Next.js App Router
│   │   ├── [locale]/            # Rutas internacionalizadas
│   │   │   ├── models/          # Listado de modelos
│   │   │   ├── publish/         # Wizard de publicación
│   │   │   │   └── wizard/
│   │   │   │       ├── page.tsx     # Step 0: Intro
│   │   │   │       ├── step1/       # Identidad
│   │   │   │       ├── step2/       # Customer sheet
│   │   │   │       ├── step3/       # Artifacts
│   │   │   │       ├── step4/       # Pricing
│   │   │   │       └── step5/       # Review & Publish
│   │   │   └── evm/
│   │   │       ├── models/[id]/     # Detalle de modelo
│   │   │       └── licenses/        # Mis licencias NFT
│   │   │
│   │   ├── api/                 # API Routes
│   │   │   ├── ipfs/            # Proxy IPFS
│   │   │   ├── models/          # CRUD modelos
│   │   │   ├── metadata/        # Upload metadata
│   │   │   └── pinata/          # Upload archivos
│   │   │
│   │   ├── providers-evm.tsx    # Wagmi/RainbowKit providers
│   │   └── layout.tsx           # Root layout
│   │
│   ├── components/              # Componentes React (17 archivos)
│   │   ├── GlobalHeaderEvm.tsx      # Header con navegación y wallet
│   │   ├── TopProgressBar.tsx       # Barra de progreso de navegación
│   │   ├── NavigationProgress.tsx   # Indicador de progreso
│   │   ├── ModelCard.tsx            # Card de modelo en listados
│   │   ├── ModelDetailView.tsx      # Vista detalle (Step 5 style)
│   │   ├── ModelDetailShared.tsx    # Componentes compartidos
│   │   ├── ModelEditControls.tsx    # Controles de edición
```

```
│   │   ├── IpfsImage.tsx            # Imagen desde IPFS
│   │   ├── OptimizedImage.tsx       # Imagen con lazy loading
│   │   ├── QuickEditDrawer.tsx      # Edición rápida de modelo
│   │   ├── WizardFooter.tsx         # Footer del wizard
│   │   ├── WizardThemeProvider.tsx  # Theme provider wizard
│   │   ├── SelectField.tsx          # Campo select reutilizable
│   │   ├── UnifiedConnectButton.tsx # Botón conexión unificado
│   │   ├── UnifiedConnectButtonEvm.tsx # Implementación EVM
│   │   └── WebVitals.tsx            # Reporte Web Vitals
│   │
│   ├── config/                   # Configuración centralizada
│   │   ├── chains.ts             # Chain IDs, nombres, símbolos
│   │   ├── rpc.ts                # URLs RPC por chain
│   │   └── index.ts              # Exports centralizados
│   │
│   ├── hooks/                    # Custom Hooks (2 archivos)
│   │   ├── useWalletAddress.ts   # Dirección wallet EVM (wagmi)
│   │   └── useWizardNavGuard.ts # Guard navegación wizard
│   │
│   ├── lib/                      # Utilidades core (9 archivos)
│   │   ├── cache.ts              # Cache utilities con TTL
│   │   ├── crypto.ts             # Funciones criptográficas
│   │   ├── db.ts                 # Cliente PostgreSQL/Neon
│   │   ├── draft-utils.ts        # Utilidades drafts wizard
│   │   ├── fetchEvmModel.ts      # Fetch modelos EVM + IPFS
│   │   ├── indexer.ts            # Indexador blockchain → DB
│   │   ├── indexer-single.ts     # Indexador modelo individual
│   │   ├── metrics.ts            # Métricas y contadores
│   │   └── prefetch.ts           # Prefetch datos SSR
│   │
│   ├── messages/                 # Traducciones i18n
│   │   ├── en.json               # Inglés (~950 keys)
│   │   └── es.json               # Español (~950 keys)
│   │
│   ├── store/                    # Estado global (Zustand)
│   │   └── market.ts             # Store del marketplace
│   │
│   └── viewmodels/               # ViewModels (abstracción UI)
│       ├── types.ts              # Interfaces TypeScript
│       ├── factories.ts          # Factory functions
│       └── adapters.ts           # Adaptadores de datos
│
├── contracts/evm/                # Smart Contracts (Solidity)
│   ├── Marketplace.sol
│   └── LicenseNFT.sol
│
├── prisma/                       # Schema Prisma
│   └── schema.prisma
│
└── docs/                         # Documentación
```

---

5. Componentes del Frontend

5.1 Componentes Principales

Componente	Archivo	Descripción
GlobalHeaderEvm	`GlobalHeaderEvm.tsx`	Header con navegación, idioma, wallet

Componente	Archivo	Descripción
ModelCard	`ModelCard.tsx`	Card de modelo en listado con metadata
ModelDetailView	`ModelDetailView.tsx`	Vista completa del modelo
IpfsImage	`IpfsImage.tsx`	Imagen desde IPFS con fallback
QuickEditDrawer	`QuickEditDrawer.tsx`	Edición rápida de precios/rights
UnifiedConnectButtonEvm	`UnifiedConnectButtonEvm.tsx`	Botón conexión wallet

5.2 ModelCard - Estructura

```
┌─────────────────────────────────────┐
│  [Cover Image from IPFS]            │
├─────────────────────────────────────┤
│  Model Name                         │
│  Short summary (3-line clamp)       │
├─────────────────────────────────────┤
│  [Category] [Tasks] ← Purple chips  │
│  🔧 Arch · Framework · Precision    │
├─────────────────────────────────────┤
│  🏆 1.5 AVAX perpetual              │
│  📅 0.1 AVAX/mo subscription        │
├─────────────────────────────────────┤
│  [API] [Download] [Transferable]    │
├─────────────────────────────────────┤
│  [View Model] [Share]               │
└─────────────────────────────────────┘
```

5.3 Wizard de Publicación (5 Steps)

Step	Página	Descripción
0	`wizard/page.tsx`	Introducción y overview
1	`step1/page.tsx`	Identidad: nombre, cover, categoría
2	`step2/page.tsx`	Customer sheet: value prop, inputs/outputs
3	`step3/page.tsx`	Artifacts: upload a IPFS, instrucciones
4	`step4/page.tsx`	Pricing: perpetual, subscription, rights
5	`step5/page.tsx`	Review & Publish: resumen y TX

---

6. Smart Contracts

6.1 Marketplace.sol

Contrato principal del marketplace:

```
// Registro de modelo
function registerModel(
    string memory uri,           // IPFS URI de metadata
    uint256 perpetualPrice,      // Precio perpetuo en wei
    uint256 subscriptionPrice,   // Precio mensual en wei
    uint256 baseDurationMonths,  // Duración base suscripción
    bool canUseAPI,              // Derecho a usar API
    bool canDownload,            // Derecho a descargar
    bool isTransferable,         // Licencia transferible
```

```
    uint256 royaltyPct           // Royalty del creador (%)
) external returns (uint256 modelId)

// Compra de licencia con URI
function buyLicenseWithURI(
    uint256 modelId,
    uint8 kind,           // 0=perpetual, 1=subscription
    uint256 months,       // Meses (0 para perpetual)
    bool transferable,
    string memory tokenUri
) external payable returns (uint256 licenseId)

// Actualización de URI
function updateModelURI(uint256 modelId, string memory newUri) external

// Eventos
event ModelRegistered(uint256 indexed modelId, address owner, string uri)
event LicenseMinted(uint256 indexed licenseId, uint256 modelId, address buyer)
```

6.2 LicenseNFT.sol (ERC-721)

```
struct LicenseStatus {
    bool revoked;
    bool validApi;
    bool validDownload;
    uint8 kind;           // 0=perpetual, 1=subscription
    uint64 expiresAt;     // Timestamp de expiración
}

function licenseStatus(uint256 tokenId) external view returns (LicenseStatus)
function ownerOf(uint256 tokenId) external view returns (address)
```

6.3 Direcciones de Contratos

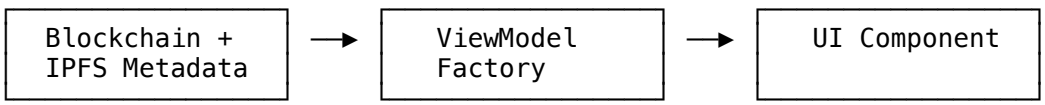
```
Avalanche Fuji (43113):
├── Marketplace: NEXT_PUBLIC_MARKETPLACE_ADDRESS_FUJI
└── LicenseNFT:  NEXT_PUBLIC_LICENSE_NFT_ADDRESS_FUJI

Avalanche Mainnet (43114):
├── Marketplace: NEXT_PUBLIC_MARKETPLACE_ADDRESS_MAINNET
└── LicenseNFT:  NEXT_PUBLIC_LICENSE_NFT_ADDRESS_MAINNET
```

---

7. Sistema de ViewModels

7.1 Arquitectura

Los ViewModels proporcionan una capa de abstracción entre datos crudos y UI:



7.2 Tipos Principales

```typescript
interface UnifiedModelViewModel {
  step1: Step1ViewModel    // Identidad
  step2: Step2ViewModel    // Customer + Technical
```

```
  step3: Step3ViewModel    // Artifacts
  step4: Step4ViewModel    // Pricing & Rights
  isPublished: boolean
}

interface Step1ViewModel {
  name: string
  tagline?: string
  summary: string
  cover?: { cid: string, url?: string }
  businessCategory?: string
  technicalCategories?: string[]
  industries?: string[]
  useCases?: string[]
  chain: 'avalanche'
  chainSymbol: 'AVAX'
  authorName?: string
  authorAddress?: string
}

interface Step4ViewModel {
  pricing: {
    perpetual?: { price: string, priceFormatted?: string }
    subscription?: { pricePerMonth: string, baseDurationMonths: number }
  }
  rights: {
    canUseAPI: boolean
    canDownload: boolean
    isTransferable: boolean
    deliveryMode?: string
  }
  revenueShare: {
    creatorRoyaltyPct: number
    marketplaceFeePct: number
  }
  termsSummary?: string[]
  termsMarkdown?: string
}
```

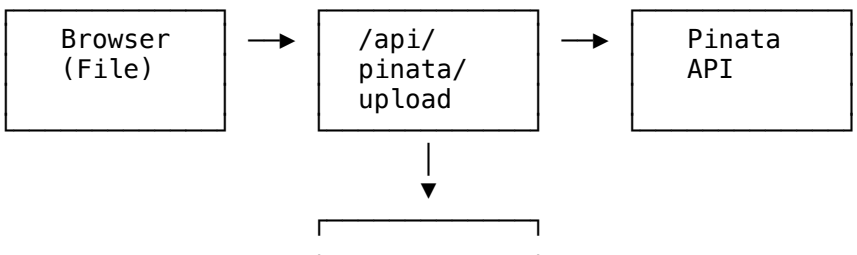
7.3 Prioridad de Datos

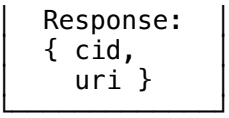
```
1. Blockchain (fuente de verdad para precios/rights)
2. Neon DB (cache indexado)
3. IPFS Metadata (datos enriquecidos)
```

---

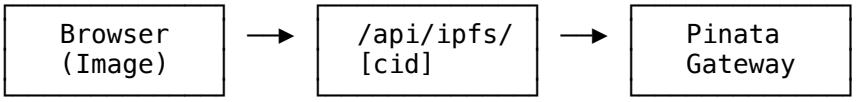
8. Integración IPFS

8.1 Flujo de Upload





8.2 Flujo de Fetch



8.3 Estructura de Metadata IPFS

```
{
  "name": "Customer Segmentation Model",
  "description": "AI model for customer segmentation",
  "image": "ipfs://QmImageCID...",

  "step1": {
    "name": "Customer Segmentation Model",
    "tagline": "Segment customers with ML",
    "summary": "Advanced ML model for...",
    "cover": { "cid": "QmCoverCID..." },
    "businessCategory": "marketingGrowth",
    "technicalCategories": ["tabular"],
    "industries": ["retail", "ecommerce"],
    "useCases": ["customerSegmentation"]
  },

  "step2": {
    "customer": {
      "valueProp": "Increase conversion by 30%",
      "inputs": "Customer transaction data (CSV)",
      "outputs": "Segment labels and scores",
      "risks": "Requires clean data"
    },
    "technical": {
      "frameworks": ["pytorch", "sklearn"],
      "architectures": ["transformer"],
      "vramGB": 8,
      "python": "3.10+"
    }
  },

  "step3": {
    "artifacts": [
      {
        "filename": "model.pt",
        "cid": "QmModelCID...",
        "size": 4200000000,
        "sha256": "abc123..."
      }
    ],
    "downloadInstructions": "pip install torch..."
  },

  "step4": {
    "pricing": {
```

```
      "perpetual": { "price": "1.5", "available": true },
      "subscription": { "pricePerMonth": "0.1", "baseDurationMonths": 1 }
    },
    "rights": {
      "canUseAPI": true,
      "canDownload": true,
      "isTransferable": true
    }
  }
}
```

---

9. Sistema de Wallet

9.1 Provider Configuration

```tsx
// src/app/providers-evm.tsx

const wagmiConfig = createConfig({
  chains: [avalancheFuji], // o avalanche para mainnet
  transports: {
    [avalancheFuji.id]: http(),
  },
  connectors: [injected()],
  ssr: true,
  storage: createStorage({
    storage: typeof window !== 'undefined' ? window.localStorage : undefined,
  }),
})

// Provider tree
<WagmiProvider config={wagmiConfig} reconnectOnMount={true}>
  <QueryClientProvider client={queryClient}>
    <RainbowKitProvider>
      <WalletEcosystemProvider>
        {children}
      </WalletEcosystemProvider>
    </RainbowKitProvider>
  </QueryClientProvider>
</WagmiProvider>
```

9.2 Dynamic Import (SSR Fix)

```tsx
// src/app/layout.tsx
const ProvidersEvm = dynamic(
  () => import('./providers-evm').then(mod => ({ default: mod.ProvidersEvm })),
  { ssr: false }
)
```

9.3 Hooks de Wallet

```tsx
// Obtener dirección conectada
const { address, isConnected } = useAccount()

// Ejecutar transacción
const { writeContract } = useWriteContract()
```

```
// Leer contrato
const { data } = useReadContract({
  address: MARKETPLACE_ADDRESS,
  abi: MarketplaceABI,
  functionName: 'getModel',
  args: [modelId],
})
```

---

10. Internacionalización (i18n)

10.1 Configuración

```
// Locales soportados
export const locales = ['en', 'es']
export const defaultLocale = 'en'
```

10.2 Namespaces

Namespace	Descripción	Keys
`header`	Navegación y header	~20
`explore`	Página de exploración	~15
`modelCard`	Cards de modelos	~25
`modelDetail`	Detalle de modelo	~80
`wizard`	Wizard de publicación	~200
`licenses`	Página de licencias	~30
`common`	Textos comunes	~20
`business`	Categorías de negocio	~50
`technical`	Categorías técnicas	~60

10.3 Uso en Componentes

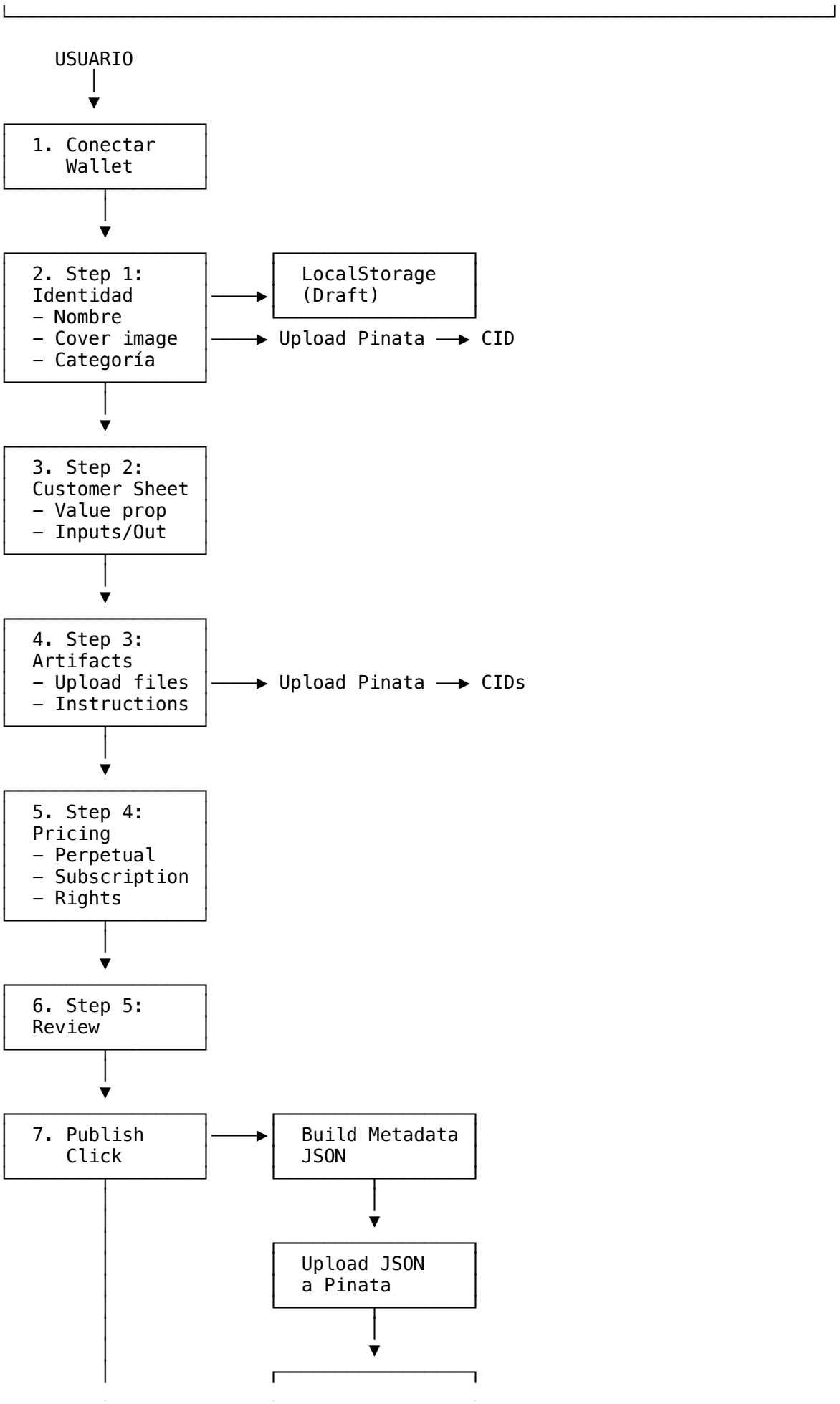
```
// Server Component
import { getTranslations } from 'next-intl/server'
const t = await getTranslations('explore')

// Client Component
import { useTranslations } from 'next-intl'
const t = useTranslations('explore')

// Uso
<Typography>{t('title')}</Typography>
<Typography>{t('subtitle')}</Typography>
```

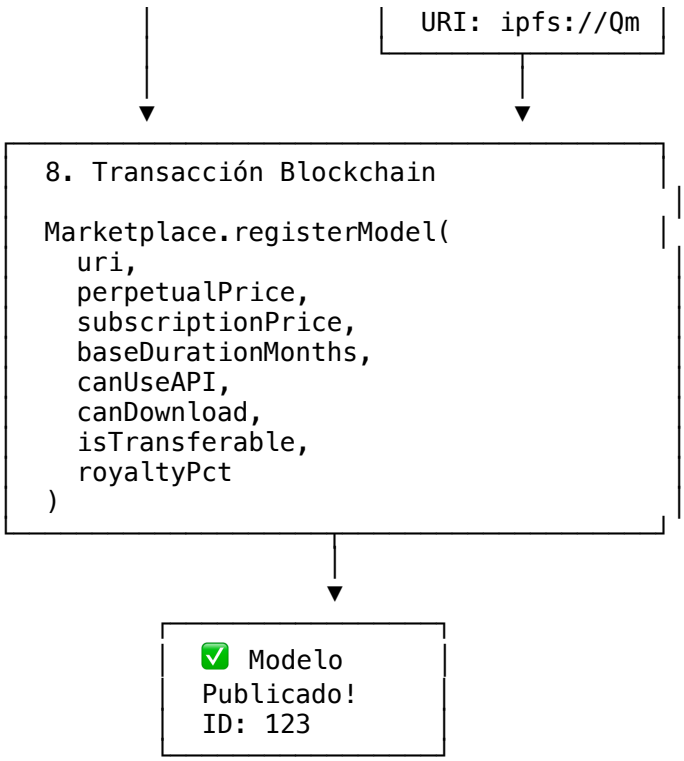
---

11. Flujos End-to-End

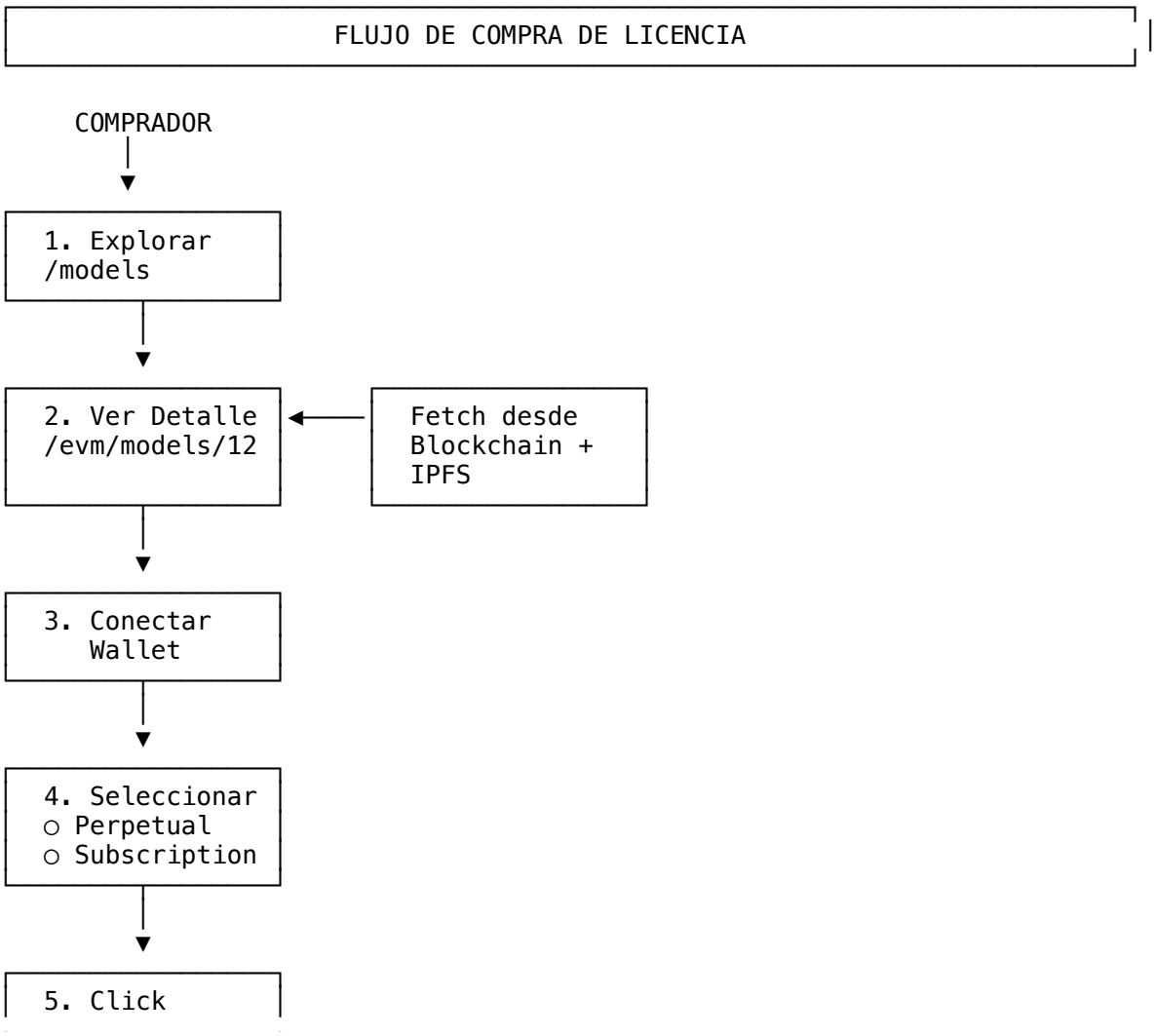
11.1 Flujo de Publicación de Modelo

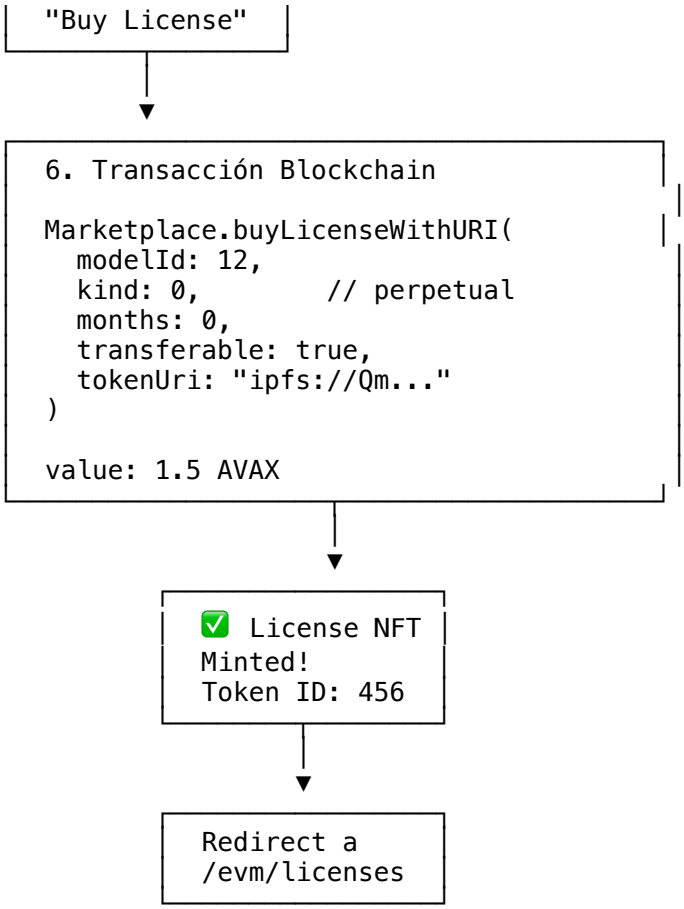
```
┌──────────────────────────────────────────────────────────────────────────┐
│                    FLUJO DE PUBLICACIÓN DE MODELO                        │
```


USUARIO
1. Conectar Wallet
2. Step 1: Identidad
- Nombre
- Cover image
- Categoría
LocalStorage (Draft)
Upload Pinata
CID
3. Step 2: Customer Sheet
- Value prop
- Inputs/Out
4. Step 3: Artifacts
- Upload files
- Instructions
Upload Pinata
CIDs
5. Step 4: Pricing
- Perpetual
- Subscription
- Rights
6. Step 5: Review
7. Publish Click
Build Metadata JSON
Upload JSON a Pinata

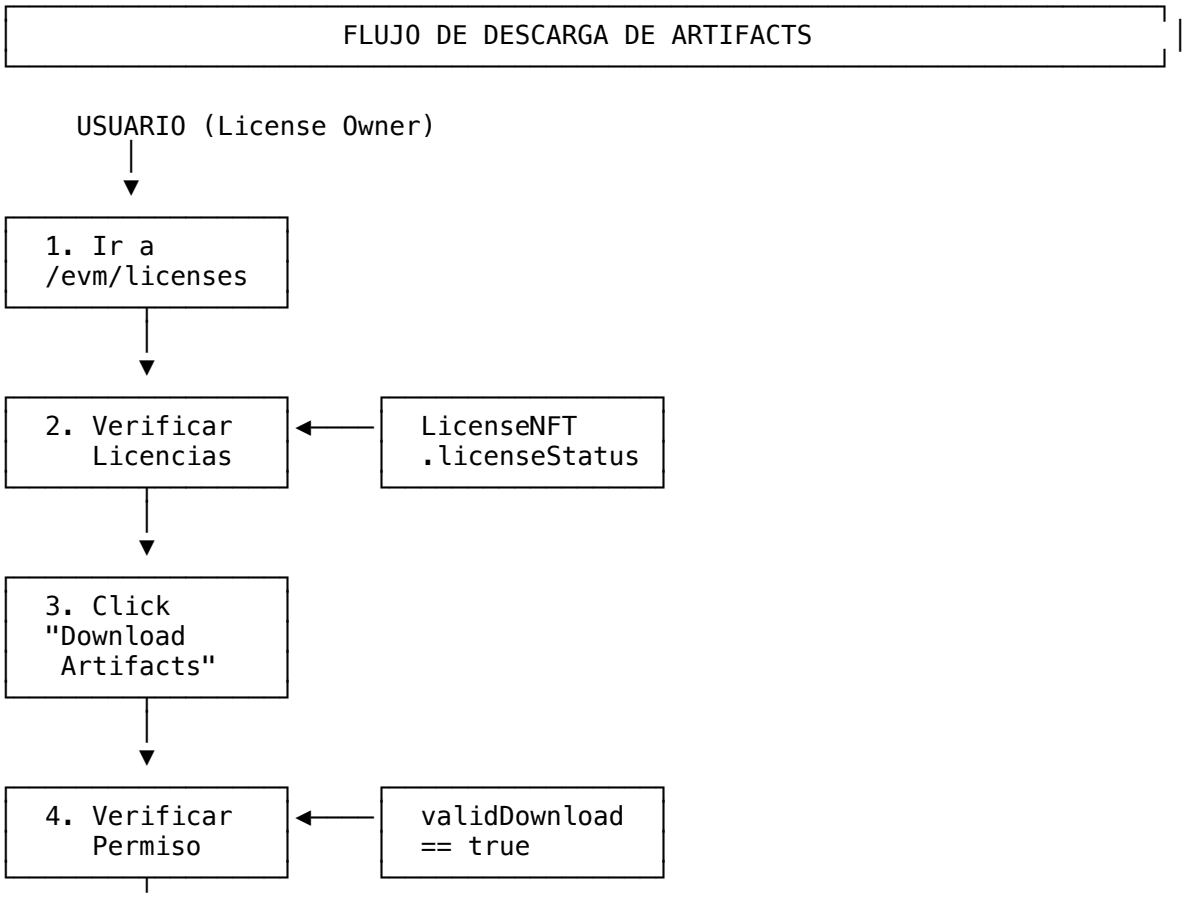


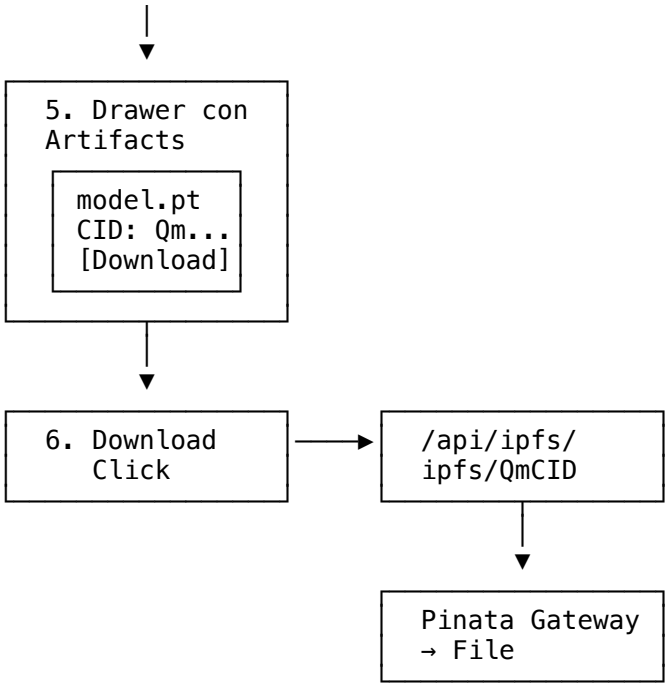
11.2 Flujo de Compra de Licencia





11.3 Flujo de Descarga de Artifacts





---

12. API Routes

12.1 Endpoints Principales

Ruta	Método	Descripción
`/api/ipfs/[...path]`	GET	Proxy para contenido IPFS
`/api/models/evm/[id]`	GET	Obtener modelo por ID
`/api/models/evm`	GET	Listar modelos
`/api/metadata/upload`	POST	Subir metadata JSON a IPFS
`/api/pinata/upload`	POST	Subir archivo a Pinata
`/api/models/evm/[id]/quick-edit-metadata`	POST	Regenerar metadata para Quick Edit

12.2 IPFS Proxy

```
// GET /api/ipfs/ipfs/QmXxx...
export async function GET(request: Request, { params }) {
  const cid = params.path.join('/')
  const response = await fetch(`${PINATA_GATEWAY}/${cid}`)
  return new Response(response.body, {
    headers: {
      'Content-Type': response.headers.get('Content-Type'),
      'Cache-Control': 'public, max-age=31536000', // 1 año
    }
  })
}
```

---

13. Configuración

13.1 Variables de Entorno

```env
# ═══════════════════════════════════════════════════════════
# BLOCKCHAIN
# ═══════════════════════════════════════════════════════════
NEXT_PUBLIC_EVM_DEFAULT_CHAIN_ID=43113  # Fuji testnet

# Contratos Fuji
NEXT_PUBLIC_MARKETPLACE_ADDRESS_FUJI=0x...
NEXT_PUBLIC_LICENSE_NFT_ADDRESS_FUJI=0x...

# Contratos Mainnet
NEXT_PUBLIC_MARKETPLACE_ADDRESS_MAINNET=0x...
NEXT_PUBLIC_LICENSE_NFT_ADDRESS_MAINNET=0x...

# ═══════════════════════════════════════════════════════════
# WALLET CONNECT
# ═══════════════════════════════════════════════════════════
NEXT_PUBLIC_WALLETCONNECT_PROJECT_ID=...

# ═══════════════════════════════════════════════════════════
# IPFS / PINATA
# ═══════════════════════════════════════════════════════════
PINATA_API_KEY=...
PINATA_SECRET_KEY=...
PINATA_JWT=...
NEXT_PUBLIC_PINATA_GATEWAY=https://gateway.pinata.cloud

# ═══════════════════════════════════════════════════════════
# RPC (opcional, tiene fallbacks)
# ═══════════════════════════════════════════════════════════
NEXT_PUBLIC_AVALANCHE_FUJI_RPC=https://api.avax-test.network/ext/bc/C/rpc
NEXT_PUBLIC_AVALANCHE_MAINNET_RPC=https://api.avax.network/ext/bc/C/rpc

# ═══════════════════════════════════════════════════════════
# DATABASE (opcional, para cache)
# ═══════════════════════════════════════════════════════════
DATABASE_URL=postgresql://...
```

13.2 Configuración de Chains

```typescript
// src/config/chains.ts
export const CHAIN_IDS = {
  AVALANCHE_FUJI: 43113,
  AVALANCHE_MAINNET: 43114,
} as const

export const CHAIN_NAMES: Record<ChainId, string> = {
  [CHAIN_IDS.AVALANCHE_FUJI]: 'Avalanche Fuji',
  [CHAIN_IDS.AVALANCHE_MAINNET]: 'Avalanche',
}

export const CHAIN_SYMBOLS: Record<ChainId, string> = {
  [CHAIN_IDS.AVALANCHE_FUJI]: 'AVAX',
  [CHAIN_IDS.AVALANCHE_MAINNET]: 'AVAX',
}
```

13.3 Comandos Útiles

```
# Desarrollo
npm run dev            # Iniciar en puerto 3000
npm run dev:3002       # Iniciar en puerto 3002

# Build
npm run build          # Build de producción
npm run typecheck      # Verificar tipos

# Linting
npm run lint           # Ejecutar ESLint
npm run lint:fix       # Corregir errores
npm run format         # Formatear con Prettier

# Database
npx prisma generate    # Generar cliente Prisma
npx prisma migrate     # Ejecutar migraciones

# Utilidades
npm run clean          # Limpiar .next
npm run doctor:port    # Liberar puerto 3002
```

Apéndice A: Glosario

Término	Definición
CID	Content Identifier - Hash único de contenido IPFS
NFT	Non-Fungible Token - Token único en blockchain
ABI	Application Binary Interface - Interfaz de contrato
SSR	Server-Side Rendering
RPC	Remote Procedure Call - Endpoint de blockchain
ViewModel	Capa de abstracción entre datos y UI
Perpetual	Licencia de pago único, acceso permanente
Subscription	Licencia de pago mensual recurrente

Apéndice B: Tags de Git

Tag	Descripción
`v1.0.0-only-avax`	Versión Avalanche-only con wallet persistence fix
`milestone-evm-detail-v1`	EVM Model Detail page v1 (Udemy-style UX)

Documento generado para MarketplaceAI v1.0.0-only-avax
Última actualización: Noviembre 2025