# Documento de Diseño y Arquitectura: Sistema Interactivo Smart Money

**Versión:** 1.0 **Autor:** MiniMax Agent **Fecha:** 22 de Junio de 2025

## 1. Visión General y Objetivos del Proyecto

### 1.1. Propósito

El objetivo de este proyecto es diseñar y documentar la arquitectura para una landing page interactiva que proporcione análisis de activos financieros (acciones y criptomonedas) basados en la metodología “Smart Money” de Esteban. La herramienta permitirá a los usuarios introducir un ticker, recibir un análisis técnico y fundamental completo, y obtener una recomendación clara de “Comprar”, “Vender” o “Mantener”, junto con una explicación didáctica del veredicto.

### 1.2. Criterios de Éxito y Métricas

* **Técnicos:**
  + **Tiempo de Carga de la Página (LCP):** < 2.5 segundos.
  + **Disponibilidad del Servicio:** 99.9%.
  + **Tasa de Éxito de las Peticiones a API:** > 99%.
* **De Usuario:**
  + **Tasa de Engagement:** > 40% de los usuarios realizan al menos un análisis.
  + **Tiempo Promedio en la Página:** > 3 minutos.
  + **Tasa de Retorno:** 15% de usuarios recurrentes en un mes.
* **De Negocio:**
  + **Generación de Leads:** Captura de correos electrónicos a través de un formulario de suscripción (opcional).

## 2. Arquitectura Técnica

Se ha seleccionado una arquitectura moderna basada en Jamstack, que garantiza un alto rendimiento, seguridad y escalabilidad.

### 2.1. Pila Tecnológica (Tech Stack)

* **Framework Frontend:** **Next.js 14+** (con App Router) - Para renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG), lo que optimiza el SEO y el rendimiento inicial.
* **Estilos CSS:** **Tailwind CSS** - Para un desarrollo rápido y un diseño consistente a través de un sistema de utilidades.
* **Visualización de Datos:** **Chart.js** y react-chartjs-2 - Para crear gráficos de velas interactivos, de volumen y otros. Es ligero y altamente personalizable.
* **Gestión de Estado:** **React Context** o **Zustand** - Para gestionar el estado global de la aplicación (ticker seleccionado, datos, estado de carga) de manera eficiente.
* **Despliegue:** **Vercel** - Plataforma nativa para Next.js que ofrece despliegue continuo, funciones serverless y escalabilidad global.

### 2.2. Arquitectura del Backend y Flujo de Datos

El backend se gestionará a través de **Next.js API Routes**, que actuarán como un proxy seguro entre el cliente y las APIs externas.

1. **Solicitud del Cliente:** El usuario introduce un ticker (ej. AAPL) en la interfaz.
2. **Llamada a la API Route:** El frontend realiza una petición a un endpoint interno, por ejemplo, /api/analyze?ticker=AAPL.
3. **API Route (Backend):**
   * La API Route recibe la solicitud.
   * Realiza llamadas seguras a las APIs externas (Yahoo Finance, CoinGecko) utilizando claves de API almacenadas como variables de entorno en el servidor.
   * Recopila los datos técnicos (precios, volumen) y fundamentales (P/E, etc.).
4. **Motor de Scoring “Smart Money”:**
   * Los datos recopilados se pasan al motor de scoring, un módulo de TypeScript/JavaScript que implementa las reglas objetivas definidas en el informe académico.
   * El motor calcula el “Scoring de Smart Money” y genera la recomendación (“Comprar”, “Vender”, “Mantener”) y la explicación textual.
5. **Respuesta al Cliente:** La API Route devuelve un objeto JSON consolidado al frontend, que incluye los datos del activo, el scoring y la recomendación.

### 2.3. Especificaciones de APIs y Fuentes de Datos

* **Datos de Acciones:**
  + **Yahoo Finance (a través de una librería como yahoo-finance2):** Para datos históricos de precios (OHLC), volumen, y datos fundamentales básicos (P/E, ROE, etc.). Es gratuita y no requiere clave de API para datos públicos.
  + **Alpha Vantage (Free Tier):** Como fuente secundaria. Requiere clave de API, que se mantendrá segura en el backend. Ofrece datos fundamentales más detallados.
* **Datos de Criptomonedas:**
  + **CoinGecko (API Pública):** Para datos de precios, volumen, capitalización de mercado y métricas on-chain básicas. Es gratuita y tiene límites de tasa generosos.
* **Noticias y Sentimiento:**
  + Se puede integrar una API de noticias como **NewsAPI (Free Tier)** para obtener titulares relevantes que puedan influir en el sentimiento del mercado.

## 3. Diseño de la Interfaz de Usuario (UI) y Experiencia de Usuario (UX)

### 3.1. Principios de Diseño

* **Claridad y Minimalismo:** La interfaz será limpia, moderna y libre de desorden. El enfoque estará en presentar la información de manera clara y directa.
* **Jerarquía Visual:** Se utilizará un sistema de tipografía y espaciado consistente para guiar la atención del usuario hacia los elementos más importantes (como el veredicto “Smart Money”).
* **Diseño Responsivo (Mobile-First):** La interfaz se diseñará primero para dispositivos móviles y luego se adaptará a pantallas más grandes, garantizando una experiencia óptima en todos los dispositivos.
* **Modo Oscuro:** Se ofrecerá un modo oscuro opcional para mejorar la legibilidad en condiciones de poca luz y reducir la fatiga visual.

### 3.2. Flujo de Usuario

1. **Llegada a la Página:** El usuario aterriza en la landing page. Ve un titular claro, un campo de entrada para el ticker y una breve descripción del servicio.
2. **Introducción del Ticker:** El usuario escribe un ticker (ej. AAPL) en el campo de entrada. Un sistema de autocompletado sugiere tickers válidos a medida que escribe.
3. **Inicio del Análisis:** El usuario presiona “Enter” o hace clic en el botón “Analizar”.
4. **Estado de Carga:** La interfaz muestra un indicador de carga mientras se recopilan y procesan los datos en el backend.
5. **Presentación de Resultados:** Una vez que el análisis está completo, la página se actualiza para mostrar los resultados en una disposición de paneles organizada:
   * **Panel Principal: Veredicto Smart Money:** Muestra de forma prominente la recomendación (“Comprar”, “Vender”, “Mantener”), el “Scoring de Smart Money” y la explicación textual.
   * **Panel de Gráficos:** Muestra el gráfico de velas interactivo, con la opción de superponer indicadores técnicos.
   * **Panel de Métricas Técnicas:** Presenta los valores de los indicadores técnicos clave (RSI, MACD, etc.).
   * **Panel de Métricas Fundamentales:** Muestra los datos fundamentales relevantes (P/E, ROE, etc.).
   * **Panel de Noticias:** Muestra los titulares de noticias más recientes para el activo.
6. **Interacción y Exploración:** El usuario puede interactuar con los gráficos, leer las explicaciones y explorar los diferentes paneles de datos a su propio ritmo.
7. **Nuevo Análisis:** El usuario puede introducir un nuevo ticker en cualquier momento para repetir el proceso.

### 3.3. Wireframes y Mockups

**Wireframe de la Página Principal (Estado Inicial):**

Wireframe de la Página Principal

**Mockup del Panel de Resultados (Vista General):**

Mockup del Panel de Resultados

**Mockup Detallado del “Veredicto Smart Money”:**

Mockup del Veredicto Smart Money

## 4. Especificaciones Funcionales del Motor de Señales

El motor de señales es el corazón de la aplicación. A continuación, se detallan sus especificaciones funcionales.

### 4.1. Lógica del Scoring “Smart Money”

El motor implementará el sistema de “Scoring de Smart Money” definido en el informe académico. El proceso será el siguiente:

1. **Recopilación de Datos:** El motor recibirá los datos técnicos y fundamentales de la API Route.
2. **Cálculo de Indicadores:** Calculará los indicadores clave (medias móviles, RSI, MACD, estructura de mercado, etc.).
3. **Evaluación de Condiciones:** Evaluará cada uno de los factores del scoring (tendencia, estructura, fase de Wyckoff, etc.) y asignará la puntuación correspondiente.
4. **Cálculo del Scoring Total:** Sumará las puntuaciones de todos los factores para obtener el “Scoring de Smart Money” final.
5. **Generación de la Recomendación:** Basándose en el scoring total, generará la recomendación (“Comprar”, “Vender”, “Mantener”) según los umbrales definidos.
6. **Generación de la Explicación:** Creará una explicación textual dinámica que justifique la recomendación, mencionando los factores que más contribuyeron al scoring.

### 4.2. Algoritmo de Ejemplo (Pseudocódigo)

function calculateSmartMoneyScore(data: FinancialData): AnalysisResult {  
 let score = 0;  
 let explanation = [];  
  
 // Factor: Tendencia Semanal  
 if (data.price > data.sma50\_weekly) {  
 score += 2;  
 explanation.push("La tendencia a largo plazo es alcista.");  
 }  
  
 // ... (evaluar todos los demás factores)  
  
 let recommendation = "Mantener";  
 if (score > 7) recommendation = "Comprar";  
 if (score < -7) recommendation = "Vender";  
  
 return { score, recommendation, explanation: explanation.join(' ') };  
}

## 5. Plan de Implementación por Sprints

Se propone un plan de desarrollo ágil de 4 sprints de 2 semanas cada uno.

**Sprint 1: Configuración del Proyecto y Backend**

* **Objetivos:** Configurar el proyecto, desarrollar las API Routes y conectar con las fuentes de datos externas.
* **Entregables:**
  + Repositorio de código en GitHub.
  + Proyecto Next.js inicializado.
  + API Routes funcionales para obtener datos de Yahoo Finance y CoinGecko.

**Sprint 2: Motor de Señales y Lógica de Negocio**

* **Objetivos:** Desarrollar e implementar el motor de scoring “Smart Money”.
* **Entregables:**
  + Módulo del motor de scoring en TypeScript.
  + Pruebas unitarias para el motor de scoring.
  + API Route /api/analyze que devuelve un análisis completo.

**Sprint 3: Desarrollo del Frontend y UI**

* **Objetivos:** Construir la interfaz de usuario con React y Tailwind CSS.
* **Entregables:**
  + Componentes de React reutilizables (selector de ticker, paneles de métricas, etc.).
  + Página principal con el formulario de entrada.
  + Panel de resultados que consume y muestra los datos de la API.

**Sprint 4: Gráficos, Pulido y Despliegue**

* **Objetivos:** Integrar los gráficos interactivos, pulir la interfaz y desplegar la aplicación.
* **Entregables:**
  + Gráfico de velas interactivo con Chart.js.
  + Implementación del modo oscuro.
  + Optimización del rendimiento y SEO básico.
  + Despliegue de la aplicación en Vercel.

## 6. Ejemplos de Salida (Mocks)

### 6.1. Ejemplo de Análisis para AAPL (Apple Inc.)

Ejemplo de Análisis para AAPL

### 6.2. Ejemplo de Análisis para BTC-USD (Bitcoin)

Ejemplo de Análisis para BTC-USD