Documento de Análisis Funcional

# NOMBRE DEL PRODUCTO: Metasketch

# Versión: 2.0 (Fusionada y Refinada)

# 1. Introducción

**Descripción:**

* Metasketch es una herramienta colaborativa basada en IA que permite a analistas funcionales transformar ideas, requisitos y documentación del cliente en documentos funcionales estructurados y vivos.
* El proceso se divide en dos fases:

1. Fase 1: Composición del documento funcional.

Los usuarios suben documentos (notas, imágenes, requisitos, etc.) que alimentan un árbol de contenidos editable. La IA sugiere textos que pueden ser modificados mediante un editor Markdown y un chat interactivo.

1. Fase 2: Generación y exportación.

Una vez finalizado, el documento puede exportarse a Word o PDF, manteniendo una fuente editable para futuras iteraciones.

**Objetivo**:

* Facilitar y automatizar el trabajo de análisis funcional, desde la recopilación de requisitos hasta la generación de documentación profesional.

# 2. Alcance (Scope)

**Usuarios objetivo:**

* Analistas funcionales
* Equipos de desarrollo
* Stakeholders

**Limitaciones**:

* Interfaz intuitiva
* Soporte para múltiples formatos de entrada
* No incluye gestión de tareas ni desarrollo del sistema final del cliente

# 3. Requisitos Funcionales

## 3.1. Input y Gestión de Documentos

* Subida de múltiples archivos (PDF, DOCX, TXT, imágenes, etc.)
* Visualización en lista con enlaces
* Eliminación o reemplazo de versiones

## 3.2. Análisis de Documentos

* Lectura automática de contenido
* Extracción de requisitos funcionales y no funcionales
* Identificación de ambigüedades, inconsistencias y requisitos faltantes
* Generación de preguntas para sesiones de análisis

## 3.3. Generación del Documento Funcional

**Estructura sugerida:**

1. Introducción
2. Resumen ejecutivo
3. Contexto / Background
4. Alcance
5. Descripción general
6. Requisitos del usuario
7. Requisitos funcionales
8. Requisitos no funcionales
9. Documentación adicional
10. Guía de estilo
11. Sketches o prototipos
12. Preguntas al cliente

## 3.4. Árbol de Contenidos

* Visualización jerárquica editable del documento funcional

## 3.5. Editor Markdown

* Edición en tiempo real
* Vista dividida (edición y previsualización)
* Secciones específicas para resumen y detalle de requisitos

## 3.6. Chat Asistido por IA

* Reorganización, resumen y modificación de contenido
* Generación de secciones completas mediante prompts

## 3.7. Exportación

* Exportación a Word y PDF
* Conservación de la fuente editable

# 4. Requisitos No Funcionales

* Usabilidad: Interfaz intuitiva y accesible
* Rendimiento: Procesamiento eficiente de grandes volúmenes de datos
* Seguridad: Acceso restringido, cifrado de datos, autenticación de dos factores
* Escalabilidad: Soporte para múltiples usuarios y proyectos
* Multilingüe: Interfaz y documentos en varios idiomas, con traducción automática

# 5. Funcionalidades Adicionales

* Colaboración en tiempo real: Edición simultánea, comentarios y anotaciones
* Historial de versiones: Control y comparación de versiones
* Automatización: Plantillas, macros y scripts
* Integración: Con herramientas como Teams, Slack, bases de datos
* Análisis avanzado: Sentimiento, impacto de cambios

# 6. Documentación

* Manual de usuario
* Guía de instalación
* Ejemplos de documentos funcionales
* Sketches de interfaz (MVP y detallado)

# 7. Feedback y Preguntas al Cliente

* ¿Qué formatos específicos necesita soportar?
* ¿Qué nivel de personalización espera del editor?
* ¿Qué políticas de seguridad y acceso requiere?
* ¿Cómo desea gestionar versiones?
* ¿Qué integraciones necesita?
* ¿Requiere instrucciones online detalladas?

# 8. Metodología de Desarrollo

**Vibecoding:**

* Metasketch se desarrolla mediante un enfoque de programación conversacional, donde analistas funcionales colaboran con IA para construir la herramienta sin necesidad de escribir código. El equipo trabaja en ciclos cortos, con entregas iterativas y revisión continua.

# 9. Enfoque de Desarrollo: Vibecoding

**Descripción del enfoque:**

* Metasketch se construye como un experimento para validar un nuevo patrón de trabajo llamado Vibecoding, que consiste en una programación conversacional entre analistas funcionales y una IA. Este enfoque permite que personas sin conocimientos técnicos profundos puedan construir aplicaciones funcionales mediante diálogo iterativo con la IA.

**Características clave del enfoque:**

* Iteración continua: Los analistas describen lo que desean, revisan lo que la IA propone y solicitan ajustes mediante un chat.
* Soporte técnico ligero: Un desarrollador actúa como apoyo técnico, no como arquitecto, interviniendo solo cuando la IA no es suficiente.
* Ciclos cortos y cercanos al cliente: Se trabaja con entregas rápidas, con un miembro del equipo actuando como cliente simulado.
* Aprendizaje y validación: El objetivo no es solo construir la herramienta, sino validar que el enfoque Vibecoding puede transformar el análisis y diseño de software.

# 10. Filosofía del Producto

**Metasketch no es solo una herramienta, sino un medio para experimentar y validar una nueva forma de trabajo.**

* Diseños funcionales vivos: El documento funcional no es estático, sino una fuente viva que evoluciona con el proyecto.
* Autonomía del analista: Los analistas se convierten en sus propios clientes, lo que fomenta la responsabilidad y el aprendizaje.
* Competencia saludable: Se promueve la colaboración y la mejora continua entre equipos que desarrollan la misma aplicación.

# 11. Sketch Propuesto por el Cliente (MVP)

**Sketch MVP (mínimo producto viable):**

**🧩 Componentes del prototipo MVP:** Interfaz sencilla con:

* Panel de carga de documentos
* Vista de documentos subidos
* Árbol de contenidos editable
* Editor Markdown con vista previa
* Chat lateral con IA
* Botón de exportación (Word/PDF)

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Sketch Detallado:**

**Añade:**

* Panel de control de versiones
* Vista de comparación de versiones
* Panel de comentarios y anotaciones
* Integración con herramientas externas (Slack, Teams)
* Configuración de idioma y seguridad

# DISEÑO VISUAL

## Quiero avanzar con el diseño de prototipos.

* ¡Perfecto! Para avanzar con el diseño de prototipos de la herramienta **Metasketch**, te propongo crear primero un **wireframe básico** del MVP (mínimo producto viable), basado en los requisitos y sketches mencionados. Este incluirá:
* **🧩 Componentes del prototipo MVP:**

1. **Panel de carga de documentos**
2. **Vista de documentos subidos**
3. **Árbol de contenidos editable**
4. **Editor Markdown con vista previa**
5. **Chat lateral con IA**
6. **Botón de exportación (Word/PDF)**

* ¿Quieres que el primer prototipo sea una vista general de toda la interfaz (como un dashboard), o prefieres que diseñemos primero una pantalla específica (por ejemplo, la de carga y análisis de documentos)?

## Quiero ver una vista general del dashboard.

Aquí tienes el diseño general del **dashboard de Metasketch** que solicitaste. El prototipo muestra una vista integral de la herramienta, con todos los elementos clave organizados de forma clara y profesional. La imagen se generará a continuación.

## Quiero ver cómo se ve el editor Markdown.

Aquí tienes el diseño del **editor Markdown** de Metasketch en proceso. La imagen mostrará una interfaz clara y moderna con paneles de edición y vista previa en tiempo real, ideal para analistas funcionales. Se generará a continuación.

## Quiero ver la funcionalidad del botón de exportación.

Estoy generando la interfaz visual para mostrar cómo funcionaría el botón de exportación en Metasketch. La imagen aparecerá en breve.

## Quiero ver cómo se guarda el documento.

stoy generando la interfaz visual que muestra cómo se guarda un documento en Metasketch. La imagen aparecerá en breve.

## ¿Puedes incluir opciones de guardado automático?

¡Claro! Incluir una opción de **guardado automático** es una excelente idea para mejorar la experiencia del usuario y evitar la pérdida de información. Aquí te detallo cómo podría integrarse esta funcionalidad en la herramienta **Metasketch**:

**🛠️ Funcionalidad: Guardado Automático**

**Descripción:**

* El sistema guarda automáticamente los cambios realizados en el documento funcional cada cierto intervalo de tiempo (por ejemplo, cada 30 segundos o al detectar inactividad).
* Se muestra una notificación discreta (por ejemplo, “Cambios guardados automáticamente”) en la parte inferior del editor.
* El usuario puede activar o desactivar esta opción desde la configuración.

**Elementos visuales sugeridos:**

* Un interruptor (toggle) en la barra superior del editor:  
  🔘 Guardado automático: Activado / Desactivado
* Indicador de estado junto al botón de guardar manual:  
  💾 Guardado hace 10 segundos

**Opciones de configuración:**

* Intervalo de guardado (15s, 30s, 1min)
* Activar/desactivar notificaciones visuales
* Guardado local (en caché del navegador) o en la nube

## ¿Te gustaría que genere una imagen del diseño visual de esta funcionalidad de guardado automático en el editor?

# IMPLEMENTACIÓN MVP CON VIBECODING

## • 🧩 Componentes del prototipo MVP:

1. Panel de carga de documentos
2. Vista de documentos subidos
3. Árbol de contenidos editable
4. Editor Markdown con vista previa
5. Chat lateral con IA
6. Botón de exportación (Word/PDF)

¡Perfecto! Vamos a implementar el MVP de **Metasketch** utilizando el enfoque **Vibecoding**, es decir, mediante una colaboración conversacional paso a paso. Empezaremos por construir los componentes uno a uno, asegurándonos de que cada parte funcione correctamente antes de avanzar.

### 🧩 MVP - Paso 1: Panel de carga de documentos

**Objetivo:** Crear una interfaz web simple donde el usuario pueda:

* Subir múltiples archivos (PDF, DOCX, TXT, imágenes)
* Ver la lista de archivos subidos
* Eliminar o reemplazar archivos

***¿Quieres que esta primera versión sea una aplicación web sencilla en HTML + JavaScript (por ejemplo, usando React o solo HTML/JS puro)?  
¿O prefieres que lo hagamos como un prototipo funcional en Python (por ejemplo, con Streamlit o Flask)?***

👉 Indícame tu preferencia y comenzamos con la implementación del primer componente.

#### Qué me recomiendas?

Buena pregunta. Aquí tienes una comparación rápida para ayudarte a decidir:

**🔧 Opción 1: Streamlit (Python)**

**Ventajas:**

Muy rápido para prototipar interfaces funcionales.

Ideal para pruebas internas y validación de ideas.

Permite cargar archivos, mostrarlos, y trabajar con IA fácilmente.

No necesitas preocuparte por HTML/CSS/JS.

**Desventajas:**

No es ideal para producción o diseño visual avanzado.

Menos control sobre la interfaz que con frameworks web.

**Recomendado si:** quieres validar la lógica y flujo funcional del MVP rápidamente, sin preocuparte aún por el diseño visual final.

**🌐 Opción 2: React (JavaScript)**

**Ventajas:**

Ideal para construir interfaces modernas y escalables.

Total control sobre el diseño visual y experiencia de usuario.

Fácil de integrar con APIs y backend más adelante.

**Desventajas:**

Requiere más tiempo y conocimientos técnicos.

Necesita configuración de entorno (Node.js, bundlers, etc.)

**Recomendado si:** ya tienes claro el diseño final y quieres construir una interfaz profesional desde el inicio.

**✅ Mi recomendación:**

**Empezar con Streamlit** para construir el MVP funcional rápidamente. Una vez validado, puedes migrar a React o similar para una versión más pulida y lista para producción.

¿Te gustaría que empecemos ahora con la implementación del **panel de carga de documentos** en Streamlit?

#### Sí, empecemos con Streamlit.

* He generado el código para el **panel de carga de documentos** en Streamlit, que incluye:
* Subida de múltiples archivos (PDF, DOCX, TXT, imágenes)
* Visualización de los archivos subidos
* Botones para eliminar archivos
* Sin embargo, no puedo mostrarte directamente la app funcionando aquí, pero puedes ejecutarla fácilmente en tu entorno local. Aquí tienes el archivo con el código listo para usar:
* ¿Quieres que te lo prepare como archivo .py descargable para que lo ejecutes directamente con streamlit run?

#### Quiero ver cómo se ejecuta en mi entorno local.

Aquí tienes el archivo .py con el código de la primera parte del MVP de **Metasketch**: el panel de carga de documentos en **Streamlit**.

Para ejecutarlo en tu entorno local:

Asegúrate de tener Python y Streamlit instalados:

Guarda el archivo como metasketch\_app.py.

Ejecuta el siguiente comando en tu terminal:

Ahora generaré el archivo .py para que puedas descargarlo directamente.

Analysis

Aquí tienes el archivo listo para descargar y ejecutar en tu entorno local:

📄 [metasketch\_app.py](blob:https://m365.cloud.microsoft/dc82fad6-f7f6-4a6f-a789-3b1104ee7eb9" \t "_blank)

Una vez descargado, solo tienes que ejecutar: streamlit run metasketch\_app.py

### 🧩 MVP – Paso 2: Árbol de contenidos editable

Aquí tienes el archivo listo para descargar y ejecutar en tu entorno local:

📄 [arbol\_contenidos\_metasketch.py](blob:https://m365.cloud.microsoft/0394e311-aaf3-4000-a74d-7e230a6126b5" \t "_blank)

Para ejecutarlo, usa este comando en tu terminal: streamlit run arbol\_contenidos\_metasketch.py

### 🧩 MVP - Paso 3: Editor Markdown con vista previa

Aquí tienes el archivo listo para descargar y ejecutar en tu entorno local:

📄 [editor\_markdown\_metasketch.py](blob:https://m365.cloud.microsoft/678a6d22-6f33-42ce-9ff1-d3a8a3ddbfec" \t "_blank)

Para ejecutarlo, usa este comando en tu terminal: streamlit run editor\_markdown\_metasketch.py

### 🧩 MVP - Paso 4: Chat lateral con IA

Aquí tienes el archivo listo para descargar y ejecutar en tu entorno local:

📄 [chat\_ia\_metasketch.py](blob:https://m365.cloud.microsoft/ad71226c-5172-4c7b-9d5b-d27b3941c787" \t "_blank)

Para probarlo, ejecuta este comando en tu terminal:

streamlit run chat\_ia\_metasketch.py

### 🧩 MVP - Paso 5: Botón de exportación (Word/PDF)

Parece que hubo un error porque el módulo fpdf (necesario para generar archivos PDF) no está disponible en este entorno. Sin embargo, la funcionalidad de exportación a Word sí funciona correctamente.

Voy a generar ahora el archivo .py con la funcionalidad de exportación a **Word**, y dejaré comentada la parte de PDF para que puedas activarla fácilmente si tienes fpdf instalado en tu entorno local.

Analysis

Aquí tienes el archivo listo para descargar y ejecutar en tu entorno local:

📄 [exportador\_metasketch.py]

Para ejecutarlo, usa este comando:

streamlit run exportador\_metasketch.py

🔧 Si deseas habilitar la exportación a PDF, solo necesitas:

1. Instalar el módulo fpdf: pip install fpdf
2. Descomentar las líneas correspondientes en el archivo.

## Integremos todos los componentes del MVP en una sola aplicación Streamlit

Aquí tienes el archivo completo de la aplicación Streamlit que integra todos los componentes del MVP de **Metasketch**:

📄 [metasketch\_mvp\_app.py](blob:https://m365.cloud.microsoft/775e39c4-a326-48a1-944c-5a55c78a6390" \t "_blank)

Para ejecutarlo en tu entorno local:

1. Asegúrate de tener instalado Streamlit y python-docx: pip install streamlit python-docx
2. Ejecuta el archivo con: streamlit run metasketch\_mvp\_app.py

## ¿Cómo se vería la aplicación final?

Estoy generando una imagen que muestra cómo se vería la aplicación final de **Metasketch** con todos los componentes del MVP integrados. La verás en unos momentos.

## Quiero ver ejemplos de otras aplicaciones similares.