Documento de Pruebas de Concepto (PoC) - Pre-Alpha 0.0.X

Proyecto: ChemsTools

Versión: Pre-Alpha 0.0.3

Fecha: 9 de julio de 2025

1. Introducción

Este documento formaliza los resultados de las Pruebas de Concepto (PoC) realizadas como parte de la fase de Conceptualización (Pre-Alpha 0.0.X). El objetivo de estas pruebas es validar las decisiones tecnológicas y arquitectónicas fundamentales antes de proceder con el desarrollo a gran escala.

Las pruebas se han ejecutado sobre el prototipo conceptual desarrollado, el cual integra un frontend en React (Next.js) y un backend en Python (Django), ambos orquestados a través de Docker.

2. Pruebas de Concepto Realizadas

PoC #1: Conectividad Frontend-Backend

- **Objetivo:** Validar que la aplicación frontend puede comunicarse exitosamente con la API del backend a través de la red de Docker, respetando las políticas de CORS.
- **Hipótesis:** El componente HealthCheck en el frontend realizará una petición GET al endpoint /api/health/ del backend y recibirá una respuesta exitosa.
- **Procedimiento de Prueba:**
  1. Iniciar el entorno completo con docker-compose up --build.
  2. Abrir un navegador web y navegar a http://localhost:3000.
  3. Observar el componente "Estado del Sistema" en la interfaz.
- **Criterios de Éxito:**
  - El componente debe mostrar el estado **"Online"** en color verde.
  - La consola de desarrollador del navegador no debe mostrar errores de CORS o de red relacionados con la petición a /api/health/.
- **Resultados:**
  - La prueba fue **exitosa**. El componente HealthCheck mostró correctamente el estado "Online", confirmando la comunicación bidireccional entre los contenedores de frontend y backend.
- **Conclusión: Concepto VALIDADO.** La arquitectura cliente-servidor y la

configuración de red y CORS son correctas y funcionales.

PoC #2: Integración del Núcleo Químico (RDKit)

- **Objetivo:** Validar que la librería científica RDKit puede ser instalada y ejecutada correctamente dentro del entorno Docker del backend para realizar cálculos químicos básicos.
- **Hipótesis:** El backend podrá recibir una fórmula química, procesarla con RDKit para calcular su peso molecular y devolver el resultado sin errores de ejecución o dependencia.
- **Procedimiento de Prueba:**
  1. Utilizar la interfaz de la calculadora en http://localhost:3000.
  2. Introducir una fórmula química válida (ej. "H2O") en el campo de entrada.
  3. Hacer clic en el botón "Calcular".
  4. Observar los logs del contenedor del backend (docker-compose logs -f backend) para verificar que no se producen errores relacionados con RDKit o NumPy.
- **Criterios de Éxito:**
  - La API debe devolver un código de estado 200 OK con un JSON que contenga el peso molecular calculado.
  - Los logs del backend no deben mostrar tracebacks o errores de importación de rdkit.
  - La advertencia de incompatibilidad de NumPy ha sido resuelta especificando la versión 1.26.4.
- **Resultados:**
  - La prueba fue **exitosa**. El endpoint /api/calculate/molecular-weight/ procesó correctamente las fórmulas "H2O" y "C6H12O6" utilizando RDKit y devolvió los pesos moleculares esperados.
- **Conclusión: Concepto VALIDADO.** La librería RDKit y sus dependencias son compatibles con nuestro entorno Docker basado en Python 3.11 y pueden ser utilizadas para la lógica de negocio principal.

PoC #3: Flujo de Datos End-to-End

- **Objetivo:** Validar el ciclo completo de una petición del usuario: desde la entrada de datos en la interfaz de React hasta la visualización del resultado procesado por el backend.
- **Hipótesis:** Un usuario puede introducir datos en el frontend, estos datos serán enviados al backend a través de una petición POST, el backend los procesará y devolverá un resultado que será mostrado correctamente en la interfaz.
- **Procedimiento de Prueba:**

    1. Navegar a http://localhost:3000.
    2. Introducir la fórmula "C6H12O6" en la calculadora.
    3. Hacer clic en "Calcular".
    4. Verificar la respuesta mostrada en la interfaz de usuario.
    5. Introducir una fórmula inválida (ej. "XYZ").
    6. Verificar el mensaje de error mostrado en la interfaz.
- **Criterios de Éxito:**
    - Para una fórmula válida, la interfaz debe mostrar el peso molecular correcto (aprox. 180.156 g/mol para C6H12O6).
    - Para una fórmula inválida, la interfaz debe mostrar un mensaje de error claro proveniente del backend.
    - El estado de la aplicación (carga, resultado, error) debe gestionarse correctamente en el frontend.
- **Resultados:**
    - La prueba fue **exitosa**. El flujo de datos completo funcionó como se esperaba tanto para casos de éxito como de error, demostrando una correcta integración entre la lógica de la interfaz y la lógica de la API.
- **Conclusión: Concepto VALIDADO.** El patrón de comunicación y la gestión de estado entre el frontend y el backend son robustos y adecuados para el desarrollo de funcionalidades más complejas.

3. Conclusión General de la Fase Pre-Alpha 0.0.X

Todos los objetivos de la fase de conceptualización han sido cumplidos:

1. Se ha definido la **visión, el alcance y la arquitectura** del proyecto.
2. Se ha **configurado un repositorio y un entorno de desarrollo** local automatizado y replicable.
3. Se ha **desarrollado un prototipo conceptual** funcional.
4. Se han **validado los conceptos técnicos críticos** a través de las PoC documentadas anteriormente.

El proyecto está listo para avanzar a la siguiente fase de desarrollo: **Pre-Alpha 0.1.X - Fase de Estructura Base**.