Documentación Técnica: Fase de Funcionalidad Esencial (Pre-Alpha 0.2.X)

Proyecto: ChemsTools

Versión: 0.2.2

Fecha: 12 de julio de 2025

1. Introducción

Este documento consolida la documentación técnica generada durante la fase **Pre-Alpha 0.2.X**. Su propósito es servir como una guía central para entender la arquitectura actual, las interfaces de programación de aplicaciones (API), los flujos de integración entre subsistemas y los procedimientos de desarrollo del proyecto ChemsTools.

2. Documentación de API

La documentación de nuestra API es fundamental para el desarrollo y la integración. Se proporciona de dos maneras: una documentación interactiva en vivo y un resumen manual de los endpoints clave.

2.1. Documentación Interactiva (Recomendado)

Gracias a la implementación de drf-spectacular, el proyecto cuenta con una documentación de API auto-generada y siempre actualizada. Esta es la fuente de verdad para entender los detalles de cada endpoint, sus parámetros y las estructuras de respuesta.

Una vez que el entorno de desarrollo esté en ejecución, puedes acceder a ella a través de las siguientes URLs:

- **Swagger UI:** http://localhost:8000/api/schema/swagger-ui/
  - *Ideal para probar los endpoints de forma interactiva.*
- **Redoc:** http://localhost:8000/api/schema/redoc/
  - *Ofrece una vista más limpia y legible de la documentación.*

2.2. Resumen de Endpoints Principales

A continuación, se presenta un resumen de alto nivel de los endpoints implementados hasta la fecha.

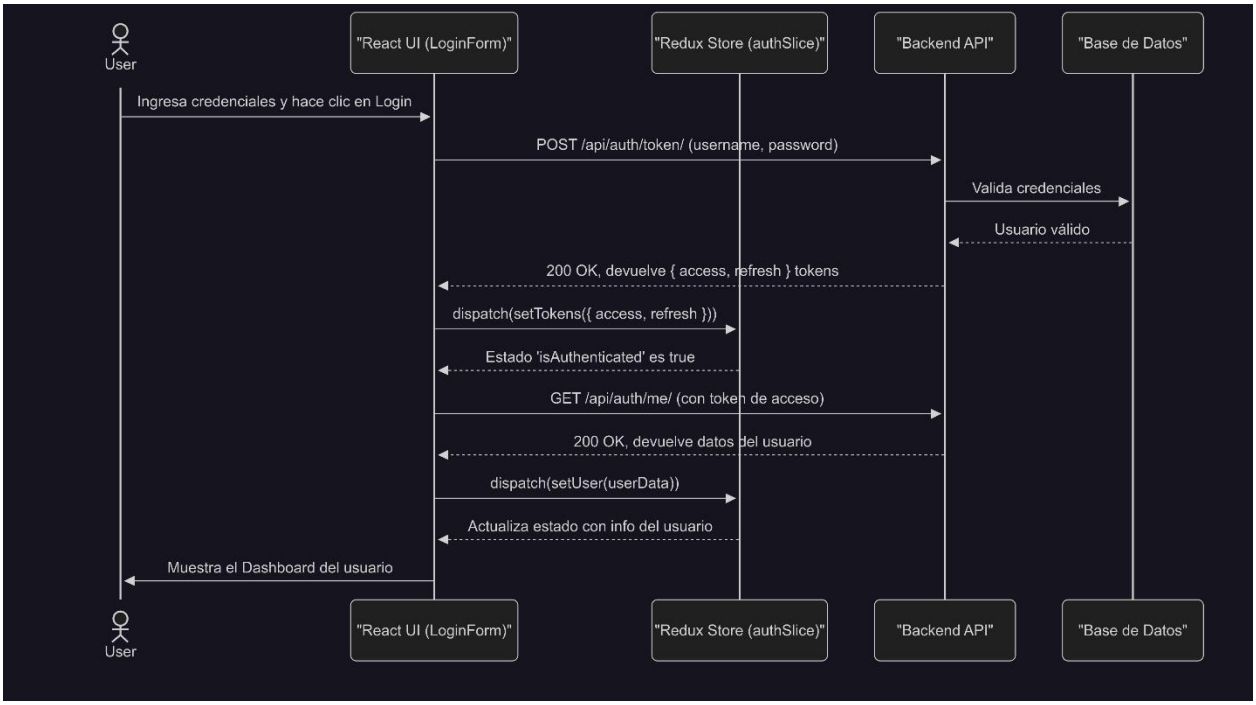
Endpoint	Método	¿Protegido?	Descripción
Autenticación			
/api/auth/register/	POST	No	Registra un nuevo usuario.
/api/auth/token/	POST	No	Inicia sesión (obtiene un par de tokens JWT: access y refresh).
/api/auth/token/refresh/	POST	No	Refresca un token de acceso expirado usando un token de refresco.
/api/auth/me/	GET	**Sí**	Obtiene la información del perfil del usuario actualmente autenticado.
Moléculas			
/api/molecules/	GET	**Sí**	Obtiene la lista (paginada) de las moléculas del usuario.
/api/molecules/	POST	**Sí**	Crea una nueva molécula para el usuario.
/api/molecules/{id}/	GET	**Sí**	Obtiene los detalles de una molécula específica.
/api/molecules/{id}/	PUT	**Sí**	Actualiza una molécula específica.
/api/molecules/{id}/	DELETE	**Sí**	Elimina una molécula específica.
Datos Estáticos			
/api/data/periodic-table/	GET	**Sí**	Devuelve los datos de la tabla periódica en formato JSON.

3. Guías de Integración de Subsistemas

Esta sección describe cómo los componentes principales del frontend y el backend se comunican para crear una experiencia de usuario cohesiva.

3.1. Flujo de Autenticación de Usuario

El proceso de inicio de sesión es un flujo crítico que involucra al usuario, la UI de React, el estado global de Redux y la API del backend.



3.2. Integración a través de Estado Global (Redux)

Redux actúa como el sistema nervioso central de nuestra aplicación frontend, permitiendo que componentes desacoplados se comuniquen y reaccionen a cambios de estado. El mejor ejemplo es la interacción entre la Tabla Periódica y la Lista de Moléculas.

4. Manuales Preliminares

Esta sección está destinada a los desarrolladores y proporciona la información esencial para empezar a trabajar en el proyecto.

4.1. Guía de Inicio Rápido

1. **Prerrequisitos:** Asegúrate de tener instalados Git, Python 3.11+, Node.js 20.x+, Docker y Docker Compose.
2. **Clonar Repositorio:** git clone <url-del-repositorio>
3. **Ejecutar Script de Configuración:** Navega a la raíz del proyecto y ejecuta python sutup.py. Selecciona la opción de configuración completa.
4. **Configurar Variables de Entorno:**
   - Crea una copia del archivo backend/.env.example en la **raíz del proyecto** y renómbrala a .env.
   - Abre el nuevo archivo .env y genera una nueva SECRET_KEY para Django.
5. **Iniciar Entorno Docker:**
   - Ejecuta docker-compose up --build para construir e iniciar todos los servicios.
6. **Aplicar Migraciones de Base de Datos:**
   - En una nueva terminal, ejecuta docker-compose exec backend python manage.py migrate.
7. **¡Listo!** La aplicación estará disponible en:
   - **Frontend:** http://localhost:3000
   - **Backend API:** http://localhost:8000

4.2. Flujo de Desarrollo

El desarrollo sigue el modelo **Git Flow simplificado**. Todo nuevo trabajo se realiza en una rama feature/ o fix/ creada a partir de develop. Los cambios se integran de nuevo en develop a través de **Pull Requests** que deben ser revisados por otro miembro del equipo y pasar todas las verificaciones de la **Integración Continua (CI)** en GitHub Actions.

4.3. Estructura del Proyecto

- **chemstools_project/**
  - **backend/**: Contiene la aplicación Django.
    - **chems_tools/**: El proyecto principal de Django (configuración y URLs principales).
    - **api/, core/, data/, molecules/, users/**: Apps de Django, cada una con una responsabilidad única.
  - **frontend/**: Contiene la aplicación React (Next.js).
    - **src/app/**: Rutas y páginas de la aplicación.
    - **src/components/**: Componentes de React, organizados por tipo (core, features, common).
    - **src/services/**: Lógica para interactuar con APIs externas.
    - **src/store/**: Configuración y slices de Redux para el estado global.
  - **docker/**: Configuraciones de Docker adicionales (ej. Nginx).
  - **.github/workflows/**: Archivos de configuración para GitHub Actions (CI).
  - **docker-compose.yml**: Orquesta los servicios de desarrollo.