Especificación de Requisitos de Software

Proyecto: VisualLab 3D

Integración de Interactive 3D Viewer y FractalLab

Autores:
Mesias Mariscal
Denise Rea
Julio Viche

Institución:
Universidad de las Fuerzas Armadas - ESPE

Fecha:
1 de Diciembre de 2025

Versión	Fecha	Descripción
1.5	1/12/2025	Reordenamiento por prioridad e impacto arquitectónico

Índice

1. Introducción **2**
1.1. Propósito 2
1.2. Alcance 2

2. Requisitos Funcionales **3**

3. Requisitos No Funcionales **7**

1. Introducción

1.1. Propósito

El propósito de este documento es definir los requisitos para **VisualLab 3D**, una plataforma web desarrollada en **React y Vite** que integra las aplicaciones existentes *Interactive 3D Viewer* y *FractalLab*.

1.2. Alcance

El sistema permite la visualización y edición de fractales y escenas 3D mediante una arquitectura de componentes (Features). Actualmente, la autenticación y persistencia operan en modo simulación (Local Storage), por lo que este documento define los requisitos para la implementación final del Backend seguro.

2. Requisitos Funcionales

RF-08: Implementación de API Backend	
Tipo: Funcional	**Prioridad:** Crítica
Procedimiento (Entradas y Salidas):	
Procedimiento: Exponer rutas HTTP que procesen las solicitudes del frontend, interactúen con la base de datos y devuelvan respuestas estructuradas. **Entrada:** Peticiones HTTP (GET, POST, PUT, DELETE) con payloads JSON. **Salida:** Respuestas HTTP (Códigos 200, 400, 500) con datos JSON estandarizados.	
Criterios de Aceptación:	
▪ **Dado** que el frontend necesita datos, ▪ **Cuando** realiza una petición a `/api/v1/*`, ▪ **Entonces** recibe una respuesta JSON válida conforme al contrato OpenAPI.	
Dependencias: Servidor Node.js/Python	**Trazabilidad:** Base para RF-02, RF-04, RF-03

RF-02: Autenticación Segura de Usuarios	
Tipo: Funcional	**Prioridad:** Alta
Procedimiento (Entradas y Salidas):	
Procedimiento: Validar las credenciales ingresadas comparándolas con los registros cifrados en la base de datos backend. **Entrada:** Correo electrónico y contraseña proporcionados en el formulario de Login. **Salida:** Token de acceso (JWT) almacenado en HttpOnly Cookie y redirección al Dashboard (éxito) o mensaje de “Credenciales inválidas” (error).	
Criterios de Aceptación:	
▪ **Dado** un usuario no autenticado en la pantalla de acceso, ▪ **Cuando** envía sus credenciales correctas, ▪ **Entonces** el sistema le otorga acceso y habilita las funciones de guardado.	
Dependencias: API Backend, Base de Datos	**Trazabilidad:** Prerrequisito para RF-04, RF-05

RF-01: Integración de Visores en Modo Dividido	
Tipo: Funcional	**Prioridad:** Alta
Procedimiento (Entradas y Salidas):	

Procedimiento: Renderizar dinámicamente dos contextos gráficos (WebGL y Canvas 2D) en una misma ventana al activar el modo dividido en el dashboard.
Entrada: Interacción del usuario con el interruptor “Vista Dividida” en el menú principal.
Salida: Interfaz dividida al 50 % mostrando el Visor 3D a la izquierda y el Visor de Fractales a la derecha, ambos interactivos.

Criterios de Aceptación:

- **Dado** que el usuario se encuentra en el Dashboard principal,
- **Cuando** activa la opción de “Vista Dividida”,
- **Entonces** el sistema redimensiona el área de trabajo y carga ambos visores sin errores de superposición de contexto WebGL.

Dependencias: Biblioteca Three.js, React Components	**Trazabilidad:** Relacionado con RF-05 (Configuraciones)

RF-06: Manipulación de Parámetros de Visualización	
Tipo: Funcional	**Prioridad:** Alta

Procedimiento (Entradas y Salidas):

Procedimiento: Procesar en tiempo real los cambios en variables matemáticas (iteraciones, zoom, color) y actualizar el renderizado WebGL correspondientemente.
Entrada: Valores numéricos o selección de texturas ingresados a través de los paneles de control del visor (Sidebar).
Salida: Actualización visual inmediata (Hot-reload) del fractal o modelo 3D en el canvas principal.

Criterios de Aceptación:

- **Dado** que el visor de fractales está activo,
- **Cuando** el usuario altera el valor de “Iteraciones”,
- **Entonces** la imagen se redibuja reflejando el nuevo nivel de detalle matemático.

Dependencias: Motor de Renderizado (Three.js/Custom Shaders)	**Trazabilidad:** Core del Negocio

RF-04: Persistencia y Gestión de Proyectos	
Tipo: Funcional	**Prioridad:** Alta

Procedimiento (Entradas y Salidas):

Procedimiento: Capturar el estado actual de los visores, serializarlo a formato JSON y enviarlo al backend para su almacenamiento permanente.
Entrada: Datos del proyecto (Nombre, Tipo, Parámetros del Fractal/Escena 3D) ingresados en el modal de guardado.
Salida: Confirmación de guardado exitoso y actualización de la lista de "Mis Proyectos" en el dashboard.

Criterios de Aceptación:

- **Dado** un usuario autenticado con un diseño fractal activo,
- **Cuando** presiona "Guardar" y asigna un nombre válido,
- **Entonces** los datos se persisten en la base de datos y están disponibles para futuras sesiones.

Dependencias: RF-02 (Autenticación), Backend	**Trazabilidad:** Depende de RF-08 (API REST)

RF-05: Gestión de Preferencias de Usuario	
Tipo: Funcional	**Prioridad:** Media

Procedimiento (Entradas y Salidas):
Procedimiento: Leer y escribir las preferencias de interfaz asociadas al ID del usuario en la base de datos.
Entrada: Selección de tema (Oscuro/Claro) o calidad de renderizado (Baja/Media/Alta).
Salida: Aplicación inmediata de estilos/configuración y persistencia para el siguiente inicio de sesión.

Criterios de Aceptación:

- **Dado** un usuario autenticado,
- **Cuando** modifica la calidad de renderizado a "Alta",
- **Entonces** esta preferencia se mantiene aunque recargue la página.

Dependencias: RF-02	**Trazabilidad:** Afecta a RF-01

RF-07: Exportación de Artefactos Visuales	
Tipo: Funcional	**Prioridad:** Media

Procedimiento (Entradas y Salidas):
Procedimiento: Convertir el buffer gráfico del canvas HTML5 en un archivo de imagen binario o el objeto de estado en un archivo de texto.
Entrada: Clic en el botón "Exportar Imagen" o "Exportar Configuración".
Salida: Descarga automática de un archivo `.png` (Alta Resolución) o `.json` (Configuración) en el dispositivo del usuario.

Criterios de Aceptación:

- **Dado** que existe un renderizado visible en pantalla,
- **Cuando** el usuario solicita la exportación,
- **Entonces** se genera el archivo correspondiente sin corromper la visualización actual.

Dependencias: HTML5 Canvas API	**Trazabilidad:** Relacionado con RF-01

RF-03: Registro de Historial de Actividades	
Tipo: Funcional	**Prioridad:** Baja
Procedimiento (Entradas y Salidas):	
Procedimiento: Registrar cronológicamente las acciones críticas realizadas por el usuario (creación, edición, eliminación) en una colección de auditoría. **Entrada:** Eventos de sistema disparados por acciones del usuario (ej. “Proyecto Guardado”). **Salida:** Registro en base de datos con timestamp, ID de usuario y detalle de la acción, visualizable en la sección “Historial”.	
Criterios de Aceptación:	
- **Dado** que un usuario realiza una modificación en un proyecto, - **Cuando** la operación se completa exitosamente, - **Entonces** se genera automáticamente una entrada en el log de auditoría sin intervención del usuario.	
Dependencias: RF-02 (Identidad de Usuario)	**Trazabilidad:** Soporte para Auditoría

3. Requisitos No Funcionales

A continuación se detallan los atributos de calidad, priorizados según su impacto en la viabilidad del sistema.

Categoría	Prioridad	Descripción y Criterio
Seguridad	**Alta**	**Descripción:** Protección de datos de usuario y credenciales. **Procedimiento:** Implementar hashing (Bcrypt) y tokens JWT. **Dependencia:** RF-02.
Rendimiento	**Alta**	**Descripción:** Tiempos de respuesta y fluidez gráfica. **Procedimiento:** Optimizar WebGL para mantener 60 FPS y respuestas de API <300ms. **Dependencia:** RF-01, RF-06.
Mantenibilidad	**Media**	**Descripción:** Facilidad para actualizar y corregir el código. **Procedimiento:** Adherir a arquitectura Features y reducir deuda técnica (SonarQube). **Dependencia:** Todo el código fuente.
Portabilidad	**Media**	**Descripción:** Capacidad de despliegue en diferentes entornos. **Procedimiento:** Contenerizar mediante Docker y scripts Shell (`setup.sh`). **Dependencia:** Infraestructura.
Usabilidad	**Baja**	**Descripción:** Facilidad de uso de la interfaz gráfica. **Procedimiento:** Diseñar interfaz responsive y dar feedback visual de acciones. **Dependencia:** Frontend.