Tecnologías y Frameworks de Testing Implementados

Backend:

El proyecto implementa un ecosistema de testing completo utilizando Jest como framework principal, que proporciona el motor de ejecución de pruebas, sistema de assertions y reporting de cobertura. Para las pruebas de APIs RESTful se integra Supertest, permitiendo simular peticiones HTTP complejas con autenticación por cookies y validación de respuestas JSON. En el ámbito de pruebas de rendimiento, se emplea K6 como herramienta especializada para generar carga concurrente y medir métricas de performance en escenarios realistas.

Como tecnologías complementarias, pg-mock facilita el aislamiento de pruebas unitarias que involucran operaciones de base de datos, mientras que debug se utiliza para logging controlado durante la ejecución de tests. La configuración de entorno se gestiona mediante dotenv con archivos específicos para testing (.env.test), y eslint asegura la calidad del código de pruebas. Para el manejo de contenedores de prueba, se aprovecha Docker Compose que despliega instancias aisladas de PostgreSQL con datos de prueba predefinidos.

La suite incluye Jest para pruebas síncronas y asíncronas con capacidades de mocking avanzado, Supertest para integración HTTP con soporte para middlewares de Express, y K6 para pruebas de carga con métricas personalizadas y thresholds configurables. Esta combinación tecnológica permite cubrir todo el espectro de testing desde unidades básicas hasta validación de rendimiento bajo estrés, utilizando cada herramienta en su ámbito de especialización para maximizar efectividad y eficiencia en el proceso de calidad del software.

Frontend:

El proyecto implementa una estrategia de testing integral que combina Vitest para pruebas unitarias de componentes individuales como botones, formularios y inputs, Playwright para pruebas end-to-end que simulan flujos completos de usuario como login, navegación y CRUDs, y React Testing Library para pruebas de integración que validan la interacción entre componentes. Esta arquitectura de testing resuelve el problema de cubrir diferentes niveles de calidad desde el código individual hasta los flujos de negocio completos, siendo fundamental porque permite la detección temprana de errores en el desarrollo local y la validación automática de funcionalidades críticas en el pipeline de despliegue. Beneficia a los desarrolladores con un ciclo de retroalimentación rápida durante el desarrollo, a los QA con pruebas automatizadas regresivas, y a los usuarios finales con una aplicación estable y confiable que mantiene su funcionalidad tras cada actualización.