UNIVERSIDAD DE COLIMA

FACULTAD DE INGENIERÍA ELECTROMECÁNICA

REPORTE DE AVANCE

PROTOCOLO

Carrera

Ingeniería de software

Semestre y Grupo

3.- E

Presenta (n)

García Bautista Dominic Isaí

Nieves Martínez Christopher Eugenio

Quiroz Páez Ricardo

Rosas Chavez Carlos Leonardo

Valdovinos Arias Kevin

Manzanillo Colima, a 05 de Octubre de 2025

Introducción

El presente documento detalla los avances correspondientes al periodo de septiembre-octubre de 2025 para el proyecto "FilaCero". El objetivo fundamental del proyecto es el **desarrollo de una aplicación web funcional tipo Punto de Venta (POS)**, que permita la gestión de catálogos de productos, control de inventario y autenticación de usuarios.

Para la consecución de este objetivo, se ha seleccionado un conjunto de tecnologías modernas que conforman un *stack* de desarrollo robusto y escalable. La arquitectura general del sistema se basa en un modelo cliente-servidor, con una clara separación entre la lógica de negocio del backend y la interfaz de usuario del frontend.

- **Tecnologías Backend:** Se utiliza el framework **NestJS** sobre Node.js por su arquitectura modular y su sólida integración con TypeScript. Para el acceso a la base de datos, se emplea el ORM **Prisma**, que facilita la interacción con una base de datos **PostgreSQL**.

- **Tecnologías Frontend:** Se desarrolla una Single-Page Application (SPA) con **Next.js**, un framework de **React** que permite un renderizado eficiente y una excelente experiencia de desarrollo. La interfaz se construye con la librería de componentes **Tailwind CSS**.

- **Infraestructura:** Todo el entorno de desarrollo y producción se gestiona a través de contenedores **Docker**, lo que garantiza la consistencia y reproducibilidad del sistema en cualquier máquina.

Arquitectura y Metodología de Desarrollo

El proyecto se organiza bajo una estructura de monorepositorio, conteniendo dos aplicaciones principales: Backend/ y Frontend/. Esta aproximación facilita la gestión del código y las dependencias en un único lugar.

La comunicación entre el cliente y el servidor se realiza a través de una API RESTful, donde el backend expone una serie de endpoints que el frontend consume para solicitar o enviar datos.

El ciclo de desarrollo se fundamenta en la orquestación de servicios mediante Docker Compose. Esta herramienta permite definir y ejecutar los tres contenedores principales del proyecto (backend, frontend, base de datos) con un único comando, simplificando drásticamente la configuración del entorno de trabajo.

Resultados y Avances Obtenidos

A continuación, se describen de forma detallada las funcionalidades y componentes implementados en cada una de las áreas del sistema.

Desarrollo del Backend (API RESTful)

El backend constituye el núcleo lógico de la aplicación. Los avances se centraron en construir una API segura y funcional.

Estructura Modular: La API se ha organizado en módulos, siguiendo las mejores prácticas de NestJS. Los módulos principales implementados son:

Auth: Gestiona el registro y la autenticación de usuarios mediante tokens JWT (JSON Web Tokens). Utiliza la librería Passport.js para implementar las estrategias de autenticación.

Products: Expone los endpoints para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la entidad de productos.

Inventory: Controla el stock de cada producto.

Categories: Administra las categorías a las que pertenecen los productos.

Acceso y Persistencia de Datos:

Se ha configurado Prisma como ORM para mapear los modelos de datos definidos en el archivo schema.prisma a las tablas de la base de datos PostgreSQL.

Para la inicialización de la base de datos, se utiliza un script SQL (db_filacero.sql) que crea el esquema, define constraints, triggers (disparadores) e inserta datos base (semillas), como los roles de usuario y las categorías iniciales. Esto garantiza la integridad referencial y la aplicación de reglas de negocio a nivel de base de datos.

Seguridad y Validación:

Se implementaron Guards de Autenticación y Roles (AuthGuard, RolesGuard) para proteger las rutas que requieren permisos específicos (ej. solo un administrador puede crear un producto).

Se utilizan DTOs (Data Transfer Objects) en conjunto con class-validator para validar los datos que llegan en las solicitudes HTTP, asegurando que la información sea correcta antes de ser procesada por los servicios.

Desarrollo del Frontend (Interfaz de Usuario)

El frontend se ha enfocado en la creación de una interfaz de administración funcional que permita interactuar de manera intuitiva con los datos del backend.

Componentes de la Interfaz: Se ha desarrollado una librería de componentes reutilizables con React y Tailwind CSS. Los componentes más relevantes son:

Paneles de Administración (AdminProductGrid, EditProductPanel): Permiten visualizar el listado de productos en una tabla, así como abrir paneles modales para editar la información de un producto o para ajustar su stock.

Formularios de Creación y Edición: Componentes que gestionan la entrada de datos del usuario y se comunican con la API para persistir los cambios.

Gestión del Estado: Para manejar el estado global de la aplicación, como la información del usuario autenticado, se ha implementado la librería Zustand. Su simplicidad permite compartir el estado entre componentes sin la complejidad de otras soluciones y sin recurrir al prop drilling.

Conectividad con la API: Toda la lógica para realizar peticiones HTTP al backend se ha centralizado en un módulo (src/lib/api.ts). Este wrapper se encarga de adjuntar automáticamente el token JWT de autenticación en las cabeceras de las solicitudes y de gestionar las respuestas y errores de forma estandarizada.

Desafíos Técnicos y Soluciones Implementadas

Durante el desarrollo de este periodo, se presentó un desafío metodológico principal:

Desafío: Gestión Híbrida del Esquema de Base de Datos. Inicialmente, se depende de un script SQL para crear la estructura y de las migraciones de Prisma para cambios posteriores. Esto puede conducir a una "doble fuente de verdad" y dificultar el mantenimiento.

Solución Propuesta: Aunque el sistema es funcional, se ha identificado como un punto de mejora a futuro. El plan es migrar la lógica de creación de catálogos y constraints del script SQL a los archivos de migración generados por Prisma, centralizando así toda la gestión del esquema en una única herramienta.

Conclusiones y Trabajo Futuro

Se ha completado con éxito la implementación de una base sólida y funcional para el sistema FilaCero. Actualmente, la aplicación permite la gestión completa del catálogo de productos y el control de inventario desde una interfaz web segura y funcional, todo dentro de un entorno de desarrollo contenerizado y reproducible.

El trabajo futuro se centrará en las siguientes áreas para expandir la funcionalidad y la calidad del proyecto:

Implementación de Pruebas: Desarrollar una estrategia de pruebas automatizadas, utilizando Jest para pruebas unitarias en el backend y React Testing Library para los componentes del frontend.

Desarrollo del Módulo de Ventas: Construir la funcionalidad principal del POS, que permitirá registrar transacciones, seleccionar productos y actualizar el inventario automáticamente a través de los triggers ya definidos en la base de datos.

Mejora de la Gestión de Usuarios: Crear una interfaz para que los administradores puedan gestionar los roles y permisos de otros usuarios del sistema.

Implementación de Logging: Integrar un sistema de registro de eventos (logging) en el backend para facilitar la depuración y el monitoreo de la aplicación.