DESARROLLAR SOFTWARE A PARTIR DE LA INTEGRACIÓN DE SUS MÓDULOS COMPONENTES
GA8-220501096-AA1-EV01.
FICHA: 2879694

Presentado por:
OLGA SOFIA GALVIS MIRANDA

TECNOLOGIA EN ANALISIS Y DESARROLLO DEL SOFTWARE
BARRANCABERMEJA
SENA 2025

DESARROLLO DE INVENTARIO PLUS: INTEGRACIÓN DE MÓDULOS

INTRODUCCIÓN

Para el proyecto Inventario Plus, que consta de Gestión de Usuarios, Gestión de Entradas/Salidas de Artículos y Gestión de Reportes, desarrollaré una solución web integrada siguiendo las mejores prácticas de desarrollo de software.

ARQUITECTURA DEL SISTEMA

- Tecnologías seleccionadas:
- Frontend: React.js con TypeScript
- Backend: Node.js con Express
- Base de datos: MongoDB (NoSQL) para flexibilidad en el modelo de datos
- Autenticación: JWT (JSON Web Tokens)

DIAGRAMA DE PAQUETES (ESTRUCTURA DEL PROYECTO)

INVENTARIO-PLUS/

```
├── client/                # Frontend
│   ├── public/
│   ├── src/
│   │   ├── api/           # Servicios API
│   │   ├── components/    # Componentes reutilizables
│   │   ├── hooks/         # Custom hooks
│   │   ├── pages/         # Vistas/páginas
│   │   ├── store/         # Estado global (Redux)
│   │   ├── types/         # Tipos TypeScript
│   │   ├── utils/         # Utilidades
│   │   └── App.tsx
├── server/                # Backend
│   ├── config/            # Configuraciones
│   ├── controllers/       # Lógica de controladores
│   ├── middleware/        # Middlewares
│   ├── models/            # Modelos de datos
│   ├── routes/            # Rutas API
│   ├── services/          # Lógica de negocio
│   ├── utils/             # Utilidades
│   └── app.js             # Aplicación principal
├── .gitignore
├── package.json
└── README.md
```

INTEGRACIÓN DE MÓDULOS PRINCIPALES

1. **Módulo de Gestión de Usuarios**

Componentes:

- AuthService: Manejo de autenticación (login, registro, recuperación de contraseña)
- UserController: Gestión de CRUD de usuarios
- RoleMiddleware: Control de acceso basado en roles

➢ **Código ejemplo (Backend - UserController):**

Javascript

```
// server/controllers/userController.js
const User = require('../models/User');
const { generateToken } = require('../utils/auth');

class UserController {
  async register(req, res) {
    try {
      const { email, password, role } = req.body;
      const existingUser = await User.findOne({ email });

      if (existingUser) {
        return res.status(400).json({ error: 'Email already in use' });
      }

      const user = new User({ email, password, role });
      await user.save();

      const token = generateToken(user);
      res.status(201).json({ user, token });
    } catch (error) {
      res.status(500).json({ error: error.message });
    }
  }
  // Otros métodos: login, getAll, update, delete, etc.
}
module.exports = new UserController();
```

2. Módulo de Gestión de Entradas/Salidas

Componentes:

- InventoryService: Lógica de negocio para movimientos
- InventoryController: Manejo de API
- InventoryValidator: Validación de datos

➢ **Código ejemplo (Frontend - InventoryForm):**

typescript

```
// client/src/components/InventoryForm.tsx
import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { createInventoryMovement } from '../api/inventoryApi';

interface InventoryFormProps {
  onSuccess: () => void;
}

const InventoryForm: React.FC<InventoryFormProps> = ({ onSuccess }) => {
  const [formData, setFormData] = useState({
    productId: '',
    quantity: 0,
    movementType: 'entry', // 'entry' or 'exit'
    date: new Date().toISOString().split('T')[0]
  });

  const dispatch = useDispatch();

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    try {
      await createInventoryMovement(formData);
      onSuccess();
    } catch (error) {
      console.error('Error creating movement:', error);
    }
  };
```

```
  return (
    <form onSubmit={handleSubmit}>
      {/* Campos del formulario */}
    </form>
  );
};
export default InventoryForm;
```

3. MÓDULO DE GESTIÓN DE REPORTES

Componentes:

- ReportGenerator: Servicio para generar reportes
- ReportController: Manejo de solicitudes de reportes
- ReportExporter: Exportación a diferentes formatos (PDF, Excel)

➢ Código ejemplo (Backend - ReportService):

Javascript

```
// server/services/reportService.js
const InventoryMovement = require('../models/InventoryMovement');
const { generatePDF, generateExcel } = require('../utils/reportGenerator');

class ReportService {
  async generateInventoryReport(params) {
    const { startDate, endDate, movementType } = params;

    const query = {
      date: { $gte: new Date(startDate), $lte: new Date(endDate) }
    };

    if (movementType) {
      query.movementType = movementType;
    }

    const movements = await InventoryMovement.find(query)
      .populate('productId')
      .populate('userId');

    return {
      pdf: await generatePDF(movements),
      excel: await generateExcel(movements)
    };
```

```
  }
}

module.exports = new ReportService();
```

4. INTEGRACIÓN DE LOS MÓDULOS

Configuración de Rutas (Backend)

Javascript

```
// server/routes/index.js
const express = require('express');
const router = express.Router();
const authMiddleware = require('../middleware/authMiddleware');
const roleMiddleware = require('../middleware/roleMiddleware');

// Rutas de autenticación
router.use('/auth', require('./authRoutes'));

// Rutas protegidas
router.use(authMiddleware);

// Rutas de usuarios
router.use('/users', roleMiddleware(['admin']), require('./userRoutes'));

// Rutas de inventario
router.use('/inventory', require('./inventoryRoutes'));

// Rutas de reportes
router.use('/reports', require('./reportRoutes'));

module.exports = router;
```

CONFIGURACIÓN DE ESTADO GLOBAL (Frontend- Redux)

Typscript

```
// client/src/store/store.ts
import { configureStore } from '@reduxjs/toolkit';
import authReducer from './authSlice';
import inventoryReducer from './inventorySlice';

export const store = configureStore({
  reducer: {
    auth: authReducer,
    inventory: inventoryReducer
  }
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

PRUEBAS UNITARIAS

Ejemplo de prueba para el servicio de usuarios:

Javascript

```
// server/tests/userService.test.js
const UserService = require('../services/userService');
const User = require('../models/User');
const { connectDB, disconnectDB } = require('../config/database');

describe('UserService', () => {
  beforeAll(async () => {
    await connectDB();
  });

  afterAll(async () => {
    await disconnectDB();
  });
```

```
  beforeEach(async () => {
    await User.deleteMany({});
  });

  describe('register', () => {
    it('should register a new user', async () => {
      const userData = {
        email: 'test@example.com',
        password: 'password123',
        role: 'user'
      };

      const user = await UserService.register(userData);
      expect(user).toHaveProperty('_id');
      expect(user.email).toBe(userData.email);
    });

    it('should throw error for duplicate email', async () => {
      const userData = {
        email: 'test@example.com',
        password: 'password123',
        role: 'user'
      };

      await UserService.register(userData);
      await expect(UserService.register(userData)).rejects.toThrow('Email already in use');
    });
  });
});
```

CONFIGURACIÓN DE SEGURIDAD

Javescript

```
// server/middleware/authMiddleware.js
const jwt = require('jsonwebtoken');
const config = require('../config/config');

module.exports = (req, res, next) => {
  const token = req.header('Authorization')?.replace('Bearer ', '');

  if (!token) {
    return res.status(401).json({ error: 'Access denied. No token provided.' });
  }

  try {
    const decoded = jwt.verify(token, config.jwtSecret);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(400).json({ error: 'Invalid token.' });
  }
};
```

DOCUMENTACIÓN DE AMBIENTES

Ambiente de Desarrollo:

- Frontend: http://localhost:3000
- Backend: http://localhost:5000
- Base de datos: MongoDB local o Atlas (URI en variables de entorno)

Variables de entorno (.env):

```
# Frontend
REACT_APP_API_URL=http://localhost:5000/api

# Backend
PORT=5000
MONGODB_URI=mongodb://localhost:27017/inventario-plus-dev
JWT_SECRET=your_jwt_secret
```

CONTROL DE VERSIONES (GIT)

Estructura de ramas:

- main - Versión estable en producción
- develop - Integración de features
- feature/* - Desarrollo de nuevas funcionalidades
- hotfix/* - Correcciones urgentes

CONCLUSIÓN

Este documento presenta la estructura y componentes clave para el desarrollo de Inventario Plus, integrando los tres módulos principales con un enfoque en:

1. Arquitectura limpia y escalable
2. Seguridad robusta
3. Componentes reutilizables
4. Pruebas automatizadas
5. Documentación clara

El código fuente completo está disponible en el repositorio Git asociado a este proyecto.