Documento de Solicitud de Desarrollo de Aplicación Web Full Stack

---

Empresa: **Tech Solutions S.A. de C.V.**
Proyecto: Desarrollo de Aplicación Web para Gestión de Inventarios
Responsable del Proyecto: Juan Pérez, CTO
Desarrollador Asignado: [Nombre del Desarrollador Full Stack]

---

1. Descripción del Proyecto

Tech Solutions S.A. de C.V. está buscando desarrollar una aplicación web para la gestión de inventarios que optimice el control de productos en almacenes. Esta aplicación permitirá a los empleados y administradores de la empresa gestionar el stock, realizar búsquedas de productos, llevar un registro de entradas y salidas, y generar reportes de inventario en tiempo real. El desarrollador deberá utilizar **React.js**, **Vue.js**, o **Angular** en el front-end y conectar esta aplicación con un servidor **back-end** que utilice una **API RESTful** para manejar las reglas de negocio y la lógica del sistema.

---

2. Requerimientos del Proyecto

2.1. Requerimientos Funcionales

1. **Gestión de Inventario**
    - Crear, leer, actualizar y eliminar (CRUD) productos en el sistema.
    - Visualización de inventario general.
    - Búsqueda de productos por nombre, categoría, y código.
    - Visualización de productos con bajo stock (filtros automáticos).
2. **Registro de Movimientos**
    - Registro de entradas y salidas de productos.
    - Historial de transacciones de cada producto.
3. **Control de Usuarios**

- Autenticación de usuarios mediante credenciales (registro, login y logout).
- Diferenciación de roles: usuario común y administrador.
- Los administradores pueden gestionar usuarios (crear, editar y eliminar).

4. **Reportes**
   - Generación de reportes detallados de inventario y movimientos.
   - Descarga de reportes en formato PDF.
5. **Notificaciones**
   - Alertas automáticas por email cuando un producto esté por agotarse.

2.2. Requerimientos No Funcionales

1. **Seguridad**
   - Autenticación mediante **JWT (JSON Web Tokens)**.
   - Encriptación de contraseñas con **bcrypt**.
   - Protección contra vulnerabilidades comunes como **XSS**, **CSRF** e inyecciones SQL.
2. **Escalabilidad**
   - El sistema debe ser capaz de manejar hasta **10,000 productos** y **500 usuarios concurrentes**.
3. **Performance**
   - Respuesta del API en menos de **300ms** para la mayoría de las operaciones CRUD.
   - Uso de **Lazy Loading** en el front-end para optimizar el tiempo de carga.
4. **Usabilidad**
   - Diseño intuitivo y responsive (adaptable a dispositivos móviles).
   - Interfaz fácil de usar para empleados no técnicos.

---

3. Lógica del Negocio

La empresa maneja varias reglas de negocio específicas para el control del inventario y la gestión de usuarios, que deben ser implementadas en la API del back-end.

3.1. Reglas de Inventario

- **Mínimo de Stock:** Cada producto tendrá un umbral mínimo de stock que, al alcanzarse, enviará una notificación por correo electrónico al administrador responsable para reabastecer dicho producto.
- **Productos en Agotamiento:** Si el stock de un producto cae por debajo del 5% del stock total, deberá aparecer en la sección de "Agotamiento" para que los usuarios puedan monitorear esos productos de forma prioritaria.
- **Control de Entradas y Salidas:** Cada vez que se añada o retire stock de un producto, deberá registrarse una transacción con la siguiente información:
    - Fecha y hora.
    - Usuario que realizó la transacción.
    - Cantidad de productos añadidos o retirados.

3.2. Reglas de Usuario

- **Roles y Permisos:**
    - Los usuarios comunes pueden ver el inventario y registrar entradas o salidas, pero no pueden modificar los datos de los productos ni gestionar usuarios.
    - Los administradores tienen permisos completos para gestionar productos y usuarios.
- **Seguridad de Sesión:** Las sesiones de usuario deberán expirar después de **30 minutos** de inactividad para asegurar la protección de los datos.

3.3. Reglas de Reportes

- **Reporte de Inventario:** El sistema debe generar un reporte completo del inventario con la fecha, cantidad de productos en stock y productos en agotamiento.
- **Reporte de Movimientos:** Los administradores podrán generar un reporte de todas las transacciones realizadas en el sistema con filtro de fechas, productos, y usuarios.

---

4. Arquitectura de la Aplicación

La aplicación se dividirá en dos partes principales: el front-end y el back-end, que estarán conectados a través de una API RESTful.

4.1. Front-end

El desarrollador debe elegir entre **React.js**, **Vue.js**, o **Angular** para implementar el front-end. La elección debe considerar la eficiencia, el rendimiento y la usabilidad.

- **Tecnologías recomendadas para el front-end:**
  - **Framework:** React.js, Vue.js o Angular.
  - **Librerías de UI:** Material UI, Bootstrap o Tailwind CSS.
  - **Gestión de estado:** Redux (para React) o Vuex (para Vue), NgRx (para Angular).
  - **HTTP:** Uso de Axios o Fetch para consumir el API.
  - **Rutas:** React Router, Vue Router, o el enrutador de Angular.

4.2. Back-end

El back-end debe estar basado en **Node.js** con **Express.js** o **Python con Django** para manejar la API, junto con la lógica de negocio y conexión a la base de datos.

- **Tecnologías recomendadas para el back-end:**
  - **Servidor:** Node.js con Express o Python con Django.
  - **Base de datos:** MongoDB (NoSQL) o MySQL/PostgreSQL (SQL).
  - **Autenticación:** JWT para sesiones de usuario.
  - **ORM:** Mongoose para MongoDB o Sequelize para SQL.

4.3. API RESTful

El servidor back-end expondrá una serie de endpoints para manejar los datos del inventario, usuarios, transacciones y reportes. Algunos de los endpoints más importantes incluyen:

- **/api/products**
  - **GET:** Obtener todos los productos.
  - **POST:** Crear un nuevo producto (solo administrador).
  - **PUT:** Actualizar un producto (solo administrador).
  - **DELETE:** Eliminar un producto (solo administrador).
- **/api/transactions**

    - **GET:** Obtener todas las transacciones.
    - **POST:** Registrar una nueva entrada o salida de stock.
- **/api/users**
    - **GET:** Obtener todos los usuarios (solo administrador).
    - **POST:** Crear un nuevo usuario (solo administrador).

---

5. Funcionalidades en Producción

La aplicación deberá estar completamente funcional y desplegada en un entorno de producción accesible. Algunas consideraciones para la fase de producción son:

- **Hosting:** La aplicación debe estar alojada en una plataforma como **Heroku**, **AWS**, **Vercel**, o **DigitalOcean**.
- **Base de Datos:** La base de datos debe estar alojada en un servicio escalable y seguro como **MongoDB Atlas** o **Amazon RDS**.
- **Certificados SSL:** La aplicación debe contar con un certificado SSL para asegurar las comunicaciones entre el cliente y el servidor.
- **Escalabilidad:** La infraestructura debe ser capaz de escalar vertical y horizontalmente según la demanda.
- **Monitorización:** Se debe integrar una herramienta de monitorización como **New Relic** o **Datadog** para controlar el rendimiento y los posibles errores.

---

6. Entregables Esperados

- Código fuente documentado del **front-end** y **back-end**.
- API RESTful funcional con autenticación.
- Base de datos configurada y con datos de prueba.
- Manual de instalación y despliegue.
- Documentación técnica para los endpoints de la API.
- Aplicación desplegada en un entorno de producción.

Fecha Estimada de Entrega: [Fecha]