¿QUÉ ES INVECTA?

Invecta es una aplicación web profesional diseñada para gestionar de forma centralizada todo el inventario de una empresa.

Desde productos y almacenes hasta movimientos y solicitudes, todo está integrado en un sistema organizado y visual.

El objetivo es mejorar la trazabilidad, evitar errores en la gestión manual y ofrecer una solución digital clara para empresas reales como SBS Ingeniería.

Está desarrollada desde cero y pensada para ser escalable, intuitiva y adaptable a diferentes tipos de empresas.

OBJETIVOS DEL SISTEMA

Invecta no es solo una plataforma para ver productos.
El sistema busca que todo el proceso de inventario esté integrado, desde que se crea un producto hasta que un empleado lo solicita, lo retira y se registra el movimiento.

Está orientado a casos reales, como empresas que manejan múltiples almacenes y empleados. Y sobre todo: busca que todo quede controlado, sin depender de hojas sueltas o memoria.

ESTRUCTURA TÉCNICA

Invecta está dividida en dos grandes bloques: el frontend y el backend.

El frontend está construido con Angular y TailwindCSS para un diseño moderno y responsivo.

El backend usa Spring Boot y Gradle, aplicando una arquitectura por capas: controller, service, impl, repository y DTO.

Todo está organizado por módulos: admi, empresa-admin, empleado, auth, main.

La base de datos usa MySQL.

Además, la seguridad está reforzada con JWT y dos interceptores en Angular: uno para añadir el token y otro para gestionar errores.”

Modelo Entidad-Relación

Este es el modelo entidad-relación del sistema Invecta.

La entidad central es empresa, que se relaciona directamente con casi todos los módulos del sistema: productos, usuarios, almacenes, proveedores, categorías, etc.

Las relaciones están organizadas principalmente como uno a muchos, por ejemplo: una empresa puede tener muchos productos, muchos usuarios o muchos almacenes.

También hay trazabilidad completa: desde que un producto es creado hasta que se solicita o se mueve dentro del inventario, todo queda reflejado en las tablas solicitud_movimiento, solicitud_personalizada y movimiento_producto.

Los colores ayudan a diferenciar los bloques funcionales: por ejemplo, en rojo están las solicitudes, en azul las entidades principales de gestión, y en verde la relación de movimiento con productos.

El modelo está optimizado para que cada empresa gestione su propio ecosistema sin interferencias entre otras empresas, y para que todo quede controlado por claves foráneas con integridad referencial.

ESTRUCTURA TÉCNICA

Todo el sistema está protegido por una capa de seguridad basada en JWT.

Cada usuario inicia sesión y recibe un token, que se envía automáticamente en cada petición gracias a los interceptores del frontend.

Hay 4 roles principales: el administrador global, el administrador de empresa, el empleado y el usuario sin empresa.

El acceso a cada endpoint está restringido desde el backend con anotaciones `@PreAuthorize`, por lo tanto, aunque un usuario intente acceder manualmente a una ruta, no podrá si no tiene permisos.

Además, el sistema detecta si un usuario se une o abandona una empresa y actualiza su rol automáticamente para garantizar coherencia y seguridad en tiempo real.

Lógica oculta y funcionalidades clave

Aquí es donde está todo lo que no se ve a simple vista pero que hace que el sistema sea realmente útil.

En primer lugar, toda la lógica del sistema está encapsulada en servicios: no se llama nunca directamente a los repositorios desde el controlador, lo que permite validar, lanzar correos, controlar stock, etc.

Por ejemplo, cuando un empleado envía una solicitud, el sistema la guarda, notifica al administrador, y cambia de estado según se apruebe o no.

Si un producto llega a su stock mínimo, se genera una alerta automática por correo al administrador de la empresa.

Nada se hace de forma directa: todo está sujeto a lógica de negocio y seguridad para evitar errores, duplicaciones o manipulaciones no autorizadas.

Diseño visual y escalabilidad

Aunque aquí se resumen varias capas, quiero destacar especialmente el diseño visual.

Toda la parte estética la he resuelto casi por completo con TailwindCSS, que me ha permitido construir una interfaz moderna, rápida de desarrollar y completamente responsive.

Me he apoyado también en componentes reutilizables como tablas y formularios, para mantener consistencia entre módulos.

Y aunque he usado algo de CSS tradicional, ha sido solo para detalles finos como bordes, márgenes específicos o animaciones suaves.

Todo esto se apoya en una estructura bien organizada, tanto en frontend como en backend, lista para crecer sin romper nada.

Despliegue de la aplicación

La aplicación se despliega en una máquina virtual de Azure usando contenedores Docker.

Tengo tres servicios principales corriendo en contenedores separados: uno para el backend de Spring Boot, otro para el frontend de Angular, y otro para la base de datos MySQL.

Para el acceso público utilizo un proxy inverso con Caddy, que expone solo los puertos 80 y 443 y gestiona automáticamente los certificados SSL.

Además, uso una DNS dinámica de No-IP con la dirección `invecta.ddns.net`, lo que me permite acceder al sistema de forma externa sin depender de IPs fijas.

El entorno está preparado para ampliarse fácilmente o migrar a un clúster si fuese necesario en el futuro.

CONCLUSIONES Y FUTURO DEL PROYECTO

Invecta no es solo un proyecto de prácticas. Es una aplicación completa, realista y útil, pensada para resolver un problema común en muchas empresas: el caos en la gestión de inventario.

La estructura del código es sólida, el diseño es limpio y moderno, y todo está preparado para escalar o integrar nuevas funciones sin tocar lo que ya funciona.

Ya está en producción con proxy, seguridad, base de datos real y módulos organizados.

A futuro, me gustaría integrar estadísticas visuales, exportación a PDF o Excel, un sistema de auditoría interna y un control de usuarios aún más potente.

En resumen, ha sido un proyecto ambicioso que combina backend, frontend, diseño y despliegue en un solo sistema funcional y profesional.