



## HISTORIAS DE USUARIOS


Nombre de la HU:		Setup inicial del proyecto
Objetivo de la HU		La empresa SportsLine ha decidido dar el primer paso hacia la transformación digital. Como parte del equipo backend, tu reto es levantar las bases sólidas del sistema: preparar el proyecto para que pueda crecer de forma ordenada y confiable. Por lo cual, como desarrollador backend tienes la responsabilidad de configurar la base del proyecto con Node.js, Typescript y PostgreSQL para garantizar un entorno estandarizado, versionado y listo para futuras integraciones.
TASK1	Inicializar proyecto Node.js + TS	
	<ul style="list-style-type: none"><li>• Crear <code>package.json</code>, instalar dependencias básicas (<code>express</code>, <code>typescript</code>, <code>ts-node</code>, <code>nodemon</code>).</li><li>• Configurar <code>tsconfig.json</code>.</li></ul>	
TASK2	Configurar estructura de carpetas	
	<ul style="list-style-type: none"><li>• Crear módulos separados (<code>models</code>, <code>controllers</code>, <code>routes</code>, <code>middlewares</code>).</li><li>• Separación de capas (DTO, DAO).</li></ul>	
TASK3	Configurar Sequelize con PostgreSQL	
	<ul style="list-style-type: none"><li>• Definir conexión en archivo <code>.env</code>.</li><li>• Crear modelos iniciales: <code>Producto</code>, <code>Cliente</code>, <code>Usuario</code>.</li></ul>	
TASK4	Crear seeds de datos iniciales	
	<ul style="list-style-type: none"><li>• Poblar tabla <code>Usuario</code> con un admin inicial.</li></ul>	




	<ul style="list-style-type: none"><li>• Poblar tabla <code>Producto</code> con 2-3 registros de ejemplo.</li></ul>
TASK 5	Configurar repositorio en GitHub con Gitflow
	<ul style="list-style-type: none"><li>• Rama <code>main</code>, <code>develop</code> y ramas de <code>feature</code>.</li></ul>
TASK 6	Configurar Docker + Docker Compose
	<ul style="list-style-type: none"><li>• Crear <code>Dockerfile</code> para Node.js.</li><li>• Configurar servicio PostgreSQL con volúmenes y <code>networks</code>.</li><li>• Limitar CPU y RAM.</li></ul>
<p>Criterios de aceptación:</p> <ul style="list-style-type: none"><li>• Configuración de proyecto con Node.js + Typescript.</li><li>• Estructura modular de carpetas con separación de responsabilidades.</li><li>• Integración de Sequelize con PostgreSQL.</li><li>• Creación de modelos iniciales (Producto, Cliente, Usuario) así como el modelo entidad relación.</li><li>• Poblar datos iniciales mediante seeds.</li><li>• Configuración de GitHub (repositorio, branches con estrategia Gitflow).</li><li>• Configuración de Docker y Docker Compose para Node.js + PostgreSQL con buenas prácticas: Limitación de CPU y RAM. o Manejo de volúmenes para persistencia. Uso de <code>networks</code> para comunicación interna entre el motor de la base de datos y el contendor donde se ejecutará Node.js</li></ul>	
<b>History points:</b> 20 puntos	
<p><b>Cierre de actividad:</b></p> <p>Al finalizar esta primera semana, serás capaz de levantar un proyecto backend en Node.js con Typescript, estructurado de manera modular y conectado a PostgreSQL mediante Sequelize. Habrás configurado un entorno estandarizado con Docker, definido los modelos iniciales y sembrado datos básicos para pruebas. Con esto, habrás sentado las bases sólidas de la aplicación, garantizando un punto de partida ordenado y preparado para futuras integraciones como autenticación y seguridad.</p>	



<b>Nombre de la HU:</b>		Autenticación y roles de usuarios
<b>Objetivo de la HU</b>		Como usuario de la API quiero registrarme e iniciar sesión con un rol específico (admin o vendedor) para acceder solo a las funcionalidades que me corresponden.
<b>TASK1</b>	Implementar modelo y servicio de autenticación	
	<ul style="list-style-type: none"><li>• Crear <code>AuthController</code> y <code>AuthService</code>.</li><li>• Registro e inicio de sesión con validaciones.</li></ul>	
<b>TASK2</b>	Configurar JWT + refreshToken	
	<ul style="list-style-type: none"><li>• Generar tokens seguros.</li><li>• Implementar endpoint de refresh.</li></ul>	
<b>TASK3</b>	Crear middlewares de protección de rutas	
	<ul style="list-style-type: none"><li>• Middleware para verificar JWT.</li><li>• Middleware para validar roles (admin/vendedor).</li></ul>	
<b>TASK4</b>	Implementar DTO y DAO en autenticación	
	<ul style="list-style-type: none"><li>• DTO para login/register.</li><li>• DAO para gestión de usuarios en DB.</li></ul>	
	Implementar cifrado híbrido	

TAS K 5	 <ul style="list-style-type: none"> <li>• Cifrado de mensajes con AES-256-GCM.</li> <li>• Clave AES cifrada con RSA.</li> </ul>
<b>Criterios de aceptación:</b> <ul style="list-style-type: none"> <li>• Registro e inicio de sesión.</li> <li>• Roles: admin y vendedor.</li> <li>• Autenticación con JWT + refreshToken.</li> <li>• Middlewares para proteger rutas.</li> <li>• Uso de DTO y DAO.</li> <li>• Cifrado híbrido (AES-256-GCM + RSA).</li> </ul>	
<b>History points:</b> 20 puntos	
<b>Cierre de actividad:</b> <p>Al finalizar esta segunda semana, serás capaz de implementar un sistema de autenticación robusto con JWT y refresh tokens, donde cada usuario tendrá un rol claramente definido (administrador o vendedor). Habrás protegido las rutas con middlewares, aplicado DTO y DAO para separar responsabilidades y agregado cifrado híbrido para operaciones sensibles. Con esto, el sistema contará con control de acceso seguro y estarás listo para gestionar datos críticos como productos y clientes.</p>	

<b>Nombre de la HU:</b>		Gestión de productos y clientes
<b>Objetivo de la HU</b>		Como administrador quiero gestionar productos y clientes para mantener información confiable y actualizada en la tienda.
TASK1	Implementar CRUD de productos	
	<ul style="list-style-type: none"> <li>• Endpoints: crear, leer, actualizar, eliminar.</li> <li>• Validar código único.</li> </ul>	
TAS K2	Implementar CRUD de clientes	
	<ul style="list-style-type: none"> <li>• Endpoints: crear, leer, actualizar, eliminar.</li> </ul>	
T A	Configurar DTO y validaciones centralizadas	
	<ul style="list-style-type: none"> <li>• Middlewares para validación de entradas.</li> </ul>	
		Implementar DAO de productos y clientes

TAS K 4	 <ul style="list-style-type: none"> <li>Definir esquemas y ejemplos de respuestas.</li> </ul>
<b>Criterios de aceptación:</b> <ul style="list-style-type: none"> <li>CRUD de productos y clientes.</li> <li>Validación de código único de productos.</li> <li>Validaciones con DTO y middleware centralizado.</li> <li>Swagger con ejemplos de respuestas (200, 201, 400, 404, 500).</li> </ul>	
<b>History points:</b> 20 puntos	
<b>Cierre de actividad:</b> <p>Al finalizar esta tercera semana, serás capaz de construir endpoints CRUD para productos y clientes, asegurando la validación de datos con DTOs y middlewares centralizados. Habrás implementado reglas de negocio como la validación de códigos únicos, documentado las operaciones en Swagger con ejemplos de respuestas y organizado la lógica en DAOs. Con esto, el sistema tendrá la capacidad de administrar el core del negocio y estarás preparado para integrar la gestión de pedidos y ventas.</p>	

<b>Nombre de la HU:</b>		Gestión de pedidos y validaciones de negocio
<b>Objetivo de la HU</b>		Como vendedor quiero registrar y consultar pedidos para garantizar trazabilidad de ventas y control de stock.
TASK 1	Implementar modelo y DAO de pedidos	
	<ul style="list-style-type: none"> <li>Relación muchos a muchos con productos.</li> </ul>	
TASK2	Crear endpoint para registrar pedidos	
	<ul style="list-style-type: none"> <li>Validar stock.</li> <li>Reducir inventario automáticamente.</li> </ul>	
TASK 3	Crear endpoint para consultar pedidos	
	<ul style="list-style-type: none"> <li>Filtrar por cliente.</li> <li>Filtrar por producto.</li> </ul>	
TAS K 4	Implementar cifrado híbrido en pedidos	
	<ul style="list-style-type: none"> <li>AES + RSA en operaciones sensibles.</li> </ul>	

TAS  
K 5

Actualizar documentación Swagger

- Definir request/response con ejemplos.

**Criterios de aceptación:**

- Registro de pedidos con múltiples productos.
- Historial de pedidos por cliente y producto.
- Validación de stock antes de confirmar pedido.
- Cifrado híbrido en operaciones.
- Swagger documentando endpoints.

**History points:** 20 puntos

**Cierre de actividad:**

Al finalizar esta cuarta semana, serás capaz de registrar y consultar pedidos vinculados a múltiples productos, asegurando la validación de stock y la correcta trazabilidad de las ventas. Habrás aplicado nuevamente el cifrado híbrido para proteger operaciones críticas y documentado los endpoints en Swagger. Con esto, el sistema contará con una gestión confiable de pedidos y estarás listo para enfocarte en la calidad del código, pruebas unitarias y despliegue final.



Nombre de la HU:		Calidad, seguridad y despliegue
Objetivo de la HU		Como líder técnico requiero que la API cumpla con estándares de calidad y despliegue para garantizar mantenibilidad y confiabilidad.
TASK1	Implementar pruebas unitarias con Jest	
	<ul style="list-style-type: none"><li>• Tests para controladores y servicios.</li><li>• Cobertura mínima del 40%.</li></ul>	
TASK2	Aplicar principios de Clean Code	
	<ul style="list-style-type: none"><li>• Refactor de controladores y servicios.</li><li>• Eliminar código duplicado.</li></ul>	
TASK3	Actualizar Swagger con todos los endpoints	
	<ul style="list-style-type: none"><li>• Revisar consistencia de cada endpoint y los ejemplos integrados</li></ul>	
TASK4	Redactar README completo	
	<ul style="list-style-type: none"><li>• Instrucciones de instalación.</li><li>• Ejecución con Docker Compose.</li></ul>	
TASK5	Verificar el control de features en GitHub	
	<ul style="list-style-type: none"><li>• Verificación de Flujo Gitflow + issues asociados a HU.</li><li>• Verificación de Integración con Azure DevOps.</li></ul>	
<b>Criterios de aceptación:</b> <ul style="list-style-type: none"><li>• Pruebas unitarias con Jest (mínimo 40% coverage).</li><li>• Principios de Clean Code.</li><li>• Swagger actualizado.</li><li>• README completo.</li><li>• Control de features en GitHub + flujo DevOps.</li></ul>		
<b>History points:</b> 20 puntos		



### Cierre de actividad:

Al finalizar esta quinta semana, serás capaz de entregar una API lista para producción, documentada con Swagger y respaldada por pruebas unitarias con Jest que garantizan al menos un 40% de cobertura. Habrás aplicado principios de Clean Code, preparado un README completo con instrucciones de despliegue en Docker Compose y gestionado las features en GitHub con flujo de trabajo en Azure DevOps. Con esto, habrás alcanzado un producto final mantenible, seguro y confiable, demostrando calidad en cada etapa del desarrollo.



[www.riwi.io](http://www.riwi.io)



301 732 53 27



Cl. 16 # 55 - 129