Diagrama de Componentes (Sprint 1)
Enfoque: UML – Diagrama de Componentes
Componentes principales:

- Frontend Web (React o HTML/CSS/JS): Interfaz de usuario, validación de formularios, almacenamiento local.
- Backend (Node.js + Express): Lógica de negocio, rutas, controladores.
- Base de Datos (SQLite): Persistencia de usuarios, sesiones, registros.
- API de Autenticación (JWT): Generación y verificación de tokens.
- Servicio de Bienvenida (Modal personalizado): Componente visual que se activa tras login exitoso.

Relaciones:

- Frontend se comunica con Backend vía HTTP (fetch/AJAX).
- Backend accede a la base de datos mediante SQLite.
- Backend genera tokens JWT y los envía al frontend.
- Frontend activa el modal de bienvenida tras recibir respuesta exitosa.

Flujo detallado:

1. Usuario ingresa credenciales en el formulario.
2. Frontend valida campos localmente.
3. Envía solicitud POST al backend.
4. Backend verifica credenciales en la base de datos.
5. Si son válidas, genera token JWT.
6. Envía token al frontend.
7. Frontend almacena token en localStorage.
8. Activa modal de bienvenida personalizado.

Notas técnicas:

- Validación doble (frontend y backend).
- Comunicación asíncrona con manejo de errores.
- Modal se activa solo si el token es válido.

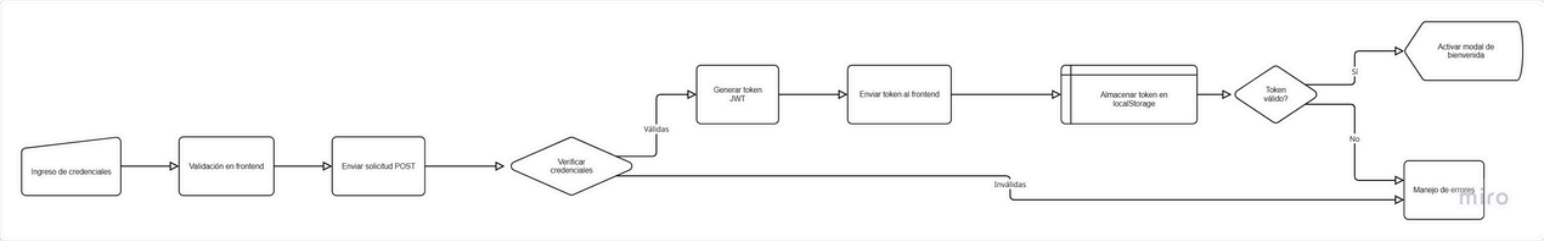

Diagrama de Despliegue (Sprint 3)
Enfoque: UML – Diagrama de Despliegue
Infraestructura:

- Servidor Frontend (Nginx + HTML/CSS/JS): Despliegue estático.
- Servidor Backend (Node.js/Express): API REST.
- Servidor de Base de Datos (SQLite): Local o en contenedor.
- Contenedor de Autenticación (JWT Service): Microservicio.
- Red Privada Virtual (VPN): Comunicación segura entre nodos.

Distribución:

- Frontend y Backend en contenedores Docker.
- Orquestación con Docker Compose o Kubernetes.
- Comunicación interna por puertos 3000 (backend) y 80 (frontend).
- Base de datos montada como volumen persistente.

Descripción de la Arquitectura

Durante tres semanas se desarrollaron los diagramas arquitectónicos del sistema, siguiendo una metodología incremental basada en entregables semanales. El objetivo fue representar de forma clara y estructurada la arquitectura lógica, los flujos de interacción y la infraestructura de despliegue del sistema web.

En el primer sprint se elaboró el diagrama de componentes utilizando el enfoque UML. Se identificaron los módulos principales: frontend web, backend con Node.js/Express, base de datos SQLite, servicio de autenticación JWT y un componente visual para el modal de bienvenida. La comunicación entre frontend y backend se realiza mediante solicitudes HTTP, mientras que el backend interactúa con la base de datos y genera tokens JWT para sesiones seguras. Esta separación de responsabilidades permite una arquitectura modular, escalable y fácil de mantener.

El segundo sprint se centró en el diagrama de secuencia, modelando el flujo de inicio de sesión. El usuario ingresa sus credenciales, que son validadas localmente en el frontend y luego enviadas al backend.

Tras verificar en la base de datos, el backend genera un token JWT y lo devuelve al cliente. Si la autenticación es exitosa, el frontend activa un modal de bienvenida personalizado. Este flujo refleja una interacción asincrónica con validación doble, manejo de errores y persistencia local, lo que mejora la experiencia de usuario y la seguridad.

En el tercer sprint se diseñó el diagrama de despliegue. Se optó por una arquitectura basada en contenedores Docker, donde el frontend se sirve desde Nginx, el backend corre en Node.js y la base de datos SQLite se monta como volumen persistente. Los servicios se comunican dentro de una red privada virtual, asegurando la integridad de los datos. Esta infraestructura permite escalar horizontalmente, facilitar el mantenimiento y asegurar la portabilidad entre entornos de desarrollo y producción.

Cada diagrama fue revisado con compañeros e instructor, y ajustado según retroalimentación para mejorar su claridad y coherencia. Se priorizó el cumplimiento de estándares como ISO/IEC 25010 en cuanto a mantenibilidad, seguridad y eficiencia. Esta documentación arquitectónica servirá como base para futuras decisiones técnicas, facilitará la comunicación entre equipos y permitirá una implementación más robusta y alineada con buenas prácticas de desarrollo de software.