Componente Frontend Del Proyecto Formativo Y Proyectos De Clase (Listas De Chequeo).

GA7-220501096-AA4-EV03.

Proyecto De Software – DOKI VET

Aprendiz: Stefanny Vanegas Marin

Tecnólogo De Análisis y Desarrollo De Software

Instructor: Milton Ivan Barbosa Gaona

Ficha: 2977481

Centro De La Tecnología De Diseño y De La Productividad Empresarial

SENA (Sena Nacional De Aprendizaje)

Girardot – Cundinamarca

Octubre 10 del 2025

Tabla De Contenido

Introducción

El presente documento formaliza el proceso de verificación y aseguramiento de la calidad de los principales artefactos generados durante la fase de análisis y desarrollo del *software* Doki Vet. Mediante la **Lista de Chequeo Integral**, se evalúa la completitud, consistencia y adherencia a los estándares definidos en tres áreas críticas del proyecto: **Documentación y Diseño (UML)**, **Calidad y Estándares de Codificación**, y **Gestión de Versiones (Versionamiento)**.

Justificación

La etapa de construcción de *software* requiere una **validación rigurosa** de sus fundamentos para prevenir errores costosos en fases posteriores (pruebas y despliegue).

La aplicación de esta lista de chequeo se justifica por los siguientes motivos:

1. **Alineación Funcional:** Permite confirmar que el diseño de *software* (Diagrama de Clases, Historias de Usuario) esté directamente alineado con los requerimientos funcionales de los módulos de Doki Vet (ej. *Gestión de Citas*, *Lista de Clientes*).
2. **Cumplimiento de Estándares:** Garantiza que el código cumpla con prácticas de **buena programación** (uso de comentarios, separación de lógica MVC), haciendo el proyecto más mantenible, legible para nuevos integrantes del equipo y menos propenso a errores.
3. **Trazabilidad:** Asegura que los procedimientos de control de versiones con Git se estén aplicando correctamente, facilitando el seguimiento de los cambios y la gestión de *releases* del proyecto alojado en el repositorio.

Objetivos

1. **Verificar la Existencia de Artefactos:** Confirmar la presencia y la calidad de los documentos de diseño (Diagrama de Clases, Historias de Usuario) y prototipos, según lo especificado en el alcance del proyecto.
2. **Evaluar la Calidad del Código:** Determinar, mediante una inspección de alto nivel, si el código fuente cumple con los estándares de codificación definidos, incluyendo el uso adecuado de comentarios y la separación de capas (MVC).
3. **Asegurar la Gestión de Versiones:** Validar que el proyecto esté utilizando correctamente un sistema de versionamiento como Git, con una estructura de ramas y mensajes de *commit* que soporten el desarrollo colaborativo y la trazabilidad.

Lista de Chequeo

	LISTA DE CHEQUEO **Verificación de Procedimientos para la Definición de Componentes Frontend (React)** **Doki Vet**	**Código:**	DOC-RE-EV-REL
		Versión	2.0
		Fecha:	26 de octubre de 2025
		Pagina:	1 de 2

Fecha:		**Asunto:**	FRONTEND-COMP - UML-DIAG - GESTION-CITAS.
Entidad:	SENA-ADSO, Equipo-Dev, Gerencia-DokiVet		
Radicado:	ARQ-2025-10-25	**Proyecto**:	Doki Vet - Gestión Veterinaria

Categoría	Ítem de Verificación	Procedimiento de Verificación Específico	Observaciones	Cumple
I. Documentación y Diseño UML				
1.1	**Diagrama de Clases**	El diagrama de clases principal define correctamente las entidades clave (ej. Cliente, Paciente, Cita, Usuario) con sus atributos y relaciones.		**SI**
1.2	**Diagramas de Casos de Uso**	Se ha modelado un diagrama que cubre la interacción de los roles principales (ej. Veterinario, Administrador) con los módulos (ej. Gestión de Citas, Registro de Cliente).		**SI**
1.3	**Historias de Usuario (HU)**	Las HU están redactadas con el formato "Como [Rol], quiero [Objetivo], para [Beneficio]" y cubren las funcionalidades de los enlaces proporcionados.		**SI**
1.4	**Diseños y Prototipos**	Los prototipos Web y Móvil (ej. Exámenes, Registro de Cambios) reflejan el diseño final y son consistentes con la estructura del *layout* de las páginas JSP.		**SI**
II. Calidad y Estándares de Codificación				
2.1	**Comentarios en el Código**	El código fuente incluye comentarios descriptivos en las secciones críticas, explicando la lógica de negocio y las funciones complejas (ej. Lógica de sesión en dashboard.jsp).		**SI**

2.2	**Cumplimiento de Estándares**	Se sigue un estándar de codificación consistente (ej. indentación, nomenclatura clara en CamelCase para variables y PascalCase para Clases/Componentes).		
2.3	**Separación de Lógica (MVC)**	Se verifica que el código frontend (HTML/JSP/CSS) está separado de la lógica de negocio *backend* (Servlets/Java), siguiendo el patrón MVC/MVVM.		
III. Gestión del Proyecto y Versionamiento				
3.1	**Herramientas de Versionamiento**	El proyecto ha sido inicializado y está siendo gestionado activamente en un repositorio de control de versiones.	Enlace del repositorio: https://github.com/teffy0302/Software-DokiVet	**SI**
3.2	**Flujo de Trabajo Git**	Se ha definido un flujo de trabajo (ej. Gitflow o Trunk-Based) con el uso de ramas para nuevas funcionalidades (feature), corrección de errores (fix) y la rama principal (main/develop).		**SI**
3.3	**Mensajes de Commit Descriptivos**	Los mensajes de commit son concisos y claros, permitiendo el seguimiento de los cambios en el repositorio.		**SI**

Conclusión

La aplicación exitosa de la Lista de Chequeo Integral concluye que el proyecto Doki Vet posee una **base documental sólida** (confirmada en el PDF adjunto) y ha adoptado herramientas fundamentales de la ingeniería de *software*, como el versionamiento Git. Si bien la verificación de los estándares de codificación (comentarios y estilo) requiere una inspección detallada del código, la existencia de todos los artefactos de diseño y de la herramienta de versionamiento coloca al proyecto en un estado maduro para avanzar a las siguientes fases de desarrollo, con una **alta probabilidad de éxito** en términos de calidad y organización.