Plan de Calidad y Costos

Mec-In

Versión 2.0

Tabla de contenido

[**Información del Proyecto 2**](#_g6io696nbi9q)

[**Plan de Gestión de Calidad 2**](#_lwt30ogg8u93)

[**Política de Calidad 2**](#_91w2w9m3mspb)

[**Objetivos de Calidad 2**](#_bbbn19pcfi2h)

[**Organización del Proyecto 3**](#_ba9mtdxc1lam)

[**Métricas de Calidad 3**](#_by7qgseyddej)

[**Actividades de Aseguramiento de Calidad 4**](#_s77x3pnro42n)

[**Control de Calidad 5**](#_ms8ipkxociua)

[**Plan de Gestión de Costos 6**](#_ynmq8zyf2kxx)

[**Estimación de Costos 6**](#_jggn8lg6ncyx)

[**Presupuesto del Proyecto 7**](#_9tl9r0mfwwq4)

[**Control de Costos 9**](#_j57x6iskpdfb)

[**Línea Base de Costos 10**](#_cwrra5r2v3vq)

[**Herramientas y Técnicas 10**](#_vsz6tgym6y6c)

[**Aprobaciones 12**](#_kisc0p8t2fno)

[**Historial de revisiones 12**](#_eowvws49k3le)

[**Referencias 13**](#_p9vyn3krkh20)

# 

# 

# 

# 

# Información del Proyecto

| Empresa / Organización | PepsiCo |
| --- | --- |
| Proyecto | Mec-IN |
| Fecha de preparación | 07-10-2025 |
| Cliente | Alexis Gonzales |
| Patrocinador principal | PepsiCo Chile |
| Gerente de proyecto | Marco Peña |
| Gerente TI | Benjamín Díaz |

# Plan de Gestión de Calidad

## Política de Calidad

El proyecto Mec-IN se compromete a entregar una aplicación web de alta calidad que cumpla con todos los requerimientos funcionales y no funcionales establecidos en el Documento de Especificación de Requerimientos **(ERS)**.

La calidad del software será asegurada mediante:

* Cumplimiento estricto de los estándares de codificación y buenas prácticas de desarrollo.
* Validación con el cliente durante reuniones a lo largo del proyecto.
* Implementación de pruebas exhaustivas en cada fase del desarrollo.
* Documentación completa y actualizada de todos los entregables.
* Mejora continua basada en retroalimentación y lecciones aprendidas.

El equipo de trabajo, garantiza que el sistema cumplirá con los criterios de aceptación definidos, priorizando la usabilidad, seguridad, rendimiento y mantenibilidad de la solución.

## Objetivos de Calidad

Los objetivos de calidad del proyecto Mec-IN están alineados con los criterios de aceptación definidos en el ERS:

1. **Funcionalidad:** Implementar el 100% de los requerimientos funcionales (RF-01 a RF-26) definidos en el ERS, validados mediante casos de prueba exitosos.
2. **Rendimiento:** Garantizar tiempos de respuesta inferiores a 15 segundos para operaciones comunes, con una meta de calidad de menos de 10 segundos y objetivo ideal menor a 5 segundos.
3. **Disponibilidad**: Alcanzar una disponibilidad mínima del 98% durante el horario laboral (Lunes a Viernes de 08:00 a 18:00 hrs).
4. **Seguridad:** Lograr cero vulnerabilidades críticas en escaneo con OWASP ZAP y garantizar el cifrado de contraseñas con bcrypt.
5. **Usabilidad:** Obtener una puntuación mínima de 80/100 en auditoría de usabilidad con Lighthouse y permitir que usuarios finales operen el sistema tras máximo 2 horas de capacitación.
6. **Cobertura de pruebas:** Alcanzar cobertura de pruebas unitarias >80% del código backend y ≥95% de éxito en casos de prueba de integración.
7. **Eficiencia operativa:** Demostrar una reducción del 40% en tiempos de registro de vehículos comparado con el proceso manual actual (de 20 minutos a < 12 minutos).
8. **Trazabilidad:** Garantizar el registro del 100% de acciones críticas mediante logs de auditoría funcionales.

## Organización del Proyecto

El proyecto cuenta con la siguiente estructura organizacional.

| Rol | Responsable | Responsabilidad en Calidad |
| --- | --- | --- |
| Gerente de Proyecto | Marco Peña | Supervisión general de calidad, aprobación de entregables, coordinación con cliente |
| Gerente TI | Benjamín Díaz | Ejecución de pruebas, revisión de código, control de calidad técnica |
| Cliente | Alexis Gonzáles (PepsiCo) | Validación de entregables, pruebas de aceptación, aprobación final |
| Usuarios Clave | Personal PepsiCo | Participación en pruebas de usabilidad y aceptación |

## Métricas de Calidad

Se medirán las siguientes métricas para evaluar y controlar la calidad del proyecto.

**Métricas de Funcionalidad:**

* Porcentaje de requerimientos funcionales implementados: Meta 100%
* Tasa de éxito de casos de prueba: Meta >95%
* Número de defectos por módulo: Meta <5 defectos menores por módulo.
* Defectos críticos pendientes: Meta = 0 al momento de entrega.

**Métricas de Rendimiento:**

* Tiempo de respuesta promedio en operaciones comunes: Meta <10 segundos (límite 15 segundos)
* Tiempo de carga inicial de la aplicación: Meta < 5 segundos
* Usuarios concurrentes soportados sin degradación: Meta >35 usuarios
* Tasa de disponibilidad del sistema: Meta >98% en horario laboral

**Métricas de Seguridad:**

* Vulnerabilidades críticas detectadas: Meta = 0
* Vulnerabilidades medias detectadas: Meta <2 (con plan de corrección)
* Contraseñas con cifrado bcrypt: Meta 100%
* Acciones críticas con registro de auditoría: Meta 100%

**Métricas de Usabilidad:**

* Puntuación Lighthouse (usabilidad): Meta >80/100
* Tiempo promedio de capacitación requerido: Meta <2 horas
* Errores de usuario durante pruebas de usabilidad: Meta <3 por sesión

**Métricas de Código:**

* Cobertura de pruebas unitarias (backend): Meta >80%
* Cobertura de pruebas de integración: Meta >95%
* Duplicación de código: Meta <5%

**Métricas de Proceso:**

* Porcentaje de entregables a tiempo: Meta >90%
* Tiempo de resolución de defectos críticos: Meta <48 horas
* Tiempo de resolución de defectos medios: Meta <1 semana
* Asistencia a reuniones de seguimiento: Meta 100%

## Actividades de Aseguramiento de Calidad

Las siguientes actividades se realizan para asegurar la calidad del ciclo de vida del proyecto.

**Revisión de Requerimientos:**

* Validación del Documento ERS.
* Análisis de factibilidad técnica de los requerimientos.
* Trazabilidad entre requerimientos y casos de uso
* Priorización y clarificación de ambigüedades con stakeholders

**Revisión de Diseño:**

* Validación de modelos de datos y arquitectura del sistema
* Revisión de mockups de interfaz
* Evaluación de decisiones de diseño frente a requerimientos no funcionales
* Análisis de escalabilidad y mantenibilidad de la arquitectura propuesta

**Revisión de Código (Code Review):**

* Revisión por pares antes de integrar cambios al repositorio principal
* Identificación de código duplicado o innecesariamente complejo
* Validación de comentarios y documentación adecuados

**Auditorías de Calidad:**

* Auditoría semanal de avance frente a plan de pruebas
* Revisión quincenal de métricas de calidad recopiladas
* Auditoría de seguridad con OWASP ZAP antes de entrega final
* Auditoría de usabilidad con Lighthouse en cada sprint

**Reuniones de Seguimiento:**

* Daily Stand-ups (reuniones diarias) durante la fase de desarrollo (15 minutos)
* Sprint Retrospectives para identificar mejoras en el proceso
* Reunión semanal de seguimiento de calidad con revisión de métricas

**Gestión de Configuración:**

* Control de versiones con Git y repositorio centralizado en GitHub (local)
* Tags de releases para versiones estables
* Backups automáticos de base de datos y repositorio

## Control de Calidad

El control de calidad se enfoca en la detección y corrección de defectos mediante actividades de pruebas exhaustivas, a continuación se mencionan pruebas.

**Pruebas Unitarias:**

* Ejecución automática de pruebas unitarias en cada commit
* Cobertura mínima de 80% del código backend
* Uso de frameworks de testing: Jest/Mocha para Node.js, Jasmine/Karma para Angular
* Validación de funciones individuales, componentes y servicios

**Pruebas de Integración:**

* Pruebas de API con Postman automatizadas
* Validación de integración entre frontend y backend
* Pruebas de interacción con base de datos MySQL
* Tasa de éxito objetivo: >95%

**Pruebas Funcionales:**

* Ejecución de casos de prueba definidos en el Plan de Pruebas Inicial
* Validación de todos los requerimientos funcionales **(RF-01 a RF-26)**
* Uso de Selenium para automatizar pruebas de interfaz
* Verificación de reglas de negocio **(RB-01 a RB-20)**
* Documentación de resultados con evidencias (capturas de pantalla)

**Pruebas de Seguridad:**

* Escaneo con OWASP ZAP para identificar vulnerabilidades
* Revisión de implementación de autenticación y autorización
* Validación de cifrado de contraseñas con bcrypt
* Pruebas de inyección SQL, XSS y otras vulnerabilidades comunes
* Verificación de logs de auditoría para acciones críticas

**Pruebas de Rendimiento:**

* Pruebas de carga con JMeter (35 usuarios concurrentes)
* Medición de tiempos de respuesta en operaciones comunes
* Pruebas de estrés para identificar límites del sistema
* Monitoreo de uso de recursos (CPU, memoria, red)
* Validación de disponibilidad del 98% en horario laboral

**Pruebas de Usabilidad:**

* Sesiones de prueba con usuarios clave identificados en Matriz de Interesados
* Auditoría automática con Google Lighthouse (meta: 80/100)
* Evaluación de responsividad en diferentes dispositivos y navegadores
* Medición de tiempo de capacitación requerido (meta: ≤2 horas)
* Recopilación de retroalimentación cualitativa de usuarios

**Pruebas de Aceptación:**

* Validación con cliente (Alexis González, PepsiCo) en reuniones
* Verificación de cumplimiento de criterios de aceptación del ERS
* Pruebas piloto con usuarios reales en ambiente de pre-producción
* Firma de acta de aceptación al finalizar pruebas exitosas
* Documentación de defectos menores para corrección en versiones futuras

# Plan de Gestión de Costos

## Estimación de Costos

El proyecto Mec-IN no cuenta con un presupuesto asignado por parte del cliente (PepsiCo), quien ha manifestado su disposición a cubrir los costos necesarios siempre que la solución propuesta cumpla con sus expectativas y requerimientos funcionales. Por lo tanto, el proyecto se desarrollará utilizando exclusivamente herramientas gratuitas o de código abierto.

**Costos de Personal**

El equipo de proyecto está compuesto por estudiantes de Ingeniería en Informática de Duoc UC que realizan este desarrollo como proyecto de titulación (Capstone), por lo que no genera costos laborales directos para el cliente. Sin embargo, se estima el valor del esfuerzo invertido.

* Gerente de Proyecto (Marco Peña): 127-140 horas de dedicación durante 12 semanas
* Gerente TI (Benjamín Díaz): 127-140 horas de dedicación durante 12 semanas
* Total esfuerzo estimado: 254-280 horas de trabajo conjunto
* Valor académico aproximado (referencial): $0

**Costos de Infraestructura Tecnológica**

Todas las herramientas utilizadas serán de licencia gratuita o código abierto:

Stack Tecnológico:

* Frontend: Angular 16+ (Código abierto - MIT License) - $0
* Backend: Node.js 18+ con Express.js (Código abierto - MIT License) - $0
* Base de Datos: MySQL 8.0+ (Código abierto - GPL License) - $0
* Control de versiones: GitHub (Plan gratuito con repositorio privado) - $0

**Herramientas de Desarrollo y Pruebas:**

* Herramientas de testing: Jest, Mocha, Jasmine, Karma, Postman, Selenium (Gratuitas) - $0
* Herramientas de seguridad: OWASP ZAP (Código abierto) - $0
* Herramientas de rendimiento: JMeter, Lighthouse (Gratuitas) - $0
* Herramientas de diseño: Figma (Plan gratuito para estudiantes) - $0
* IDE y editores: Visual Studio Code (Gratuito) - $0

**Costos de Hosting y Despliegue**

* Servidor de desarrollo: Equipos personales de los desarrolladores - $0
* Servidor de producción (opcional): Plan gratuito de servicios cloud como Heroku, Railway o Render - $0
* Dominio web (opcional): No contemplado en esta fase - $0
* Certificado SSL (opcional): Let's Encrypt gratuito - $0

**Costos de Capacitación**

* Material de capacitación: Elaborado por el equipo del proyecto - $0
* Sesiones de capacitación: 2 horas máximo por usuario, realizadas por el equipo - $0
* Documentación y manuales: Incluidos en entregables del proyecto - $0

**Total Estimado de Costos Directos para el Cliente: $0 USD**

Nota importante: Aunque el costo directo para PepsiCo es de $0 USD en esta fase del proyecto, el cliente debe considerar costos potenciales futuros para:

* Hosting en servidores propios o servicios cloud de pago para producción a gran escala
* Mantenimiento y soporte post-implementación
* Actualizaciones y nuevas funcionalidades no contempladas en esta versión
* Licencias de software adicional si se requieren herramientas empresariales específicas
* Integración con sistemas corporativos existentes (ERP, sistemas de flota nacional)

## Presupuesto del Proyecto

Dado que el proyecto se desarrolla en un contexto académico sin costos directos para el cliente, el presupuesto se estructura como una estimación de valor y asignación de recursos no monetarios.

***Distribución del Esfuerzo por Fase del Proyecto***

**Fase 1 - Inicio del Proyecto: 26 horas**

Elaboración del Acta de Constitución del proyecto: 10 horas

Identificación y análisis de interesados: 8 horas

Definición de metodología de trabajo: 8 horas

**Fase 2 - Planificación: 40 horas**

Definición del EDT y Diccionario: 10 horas

Elaboración del cronograma (Carta Gantt): 6 horas

Matriz de responsabilidades (RACI): 4 horas

Análisis de riesgos iniciales y elaboración de matriz de riesgos: 8 horas

Definición del plan de pruebas inicial: 6 horas

Plan de calidad y costos: 6 horas

**Fase 3 - Análisis y Diseño: 42 horas**

Documento de Especificación de Requerimientos (ERS): 16 horas

Modelado de datos (Diagrama Entidad-Relación): 8 horas

Diseño de arquitectura del sistema por capas: 8 horas

Mockups y diseño visual de la interfaz: 8 horas

Análisis de riesgos (actualización y elaboración de matriz): 2 horas

**Fase 4 - Desarrollo: 72 horas**

Configuración del entorno de desarrollo: 6 horas

Desarrollo del módulo de registro de vehículos (ingresos/salidas): 10 horas

Desarrollo del módulo de órdenes de trabajo (mantenimientos): 12 horas

Desarrollo del módulo de inventario de repuestos: 10 horas

Desarrollo del módulo de gestión de llaves y recursos: 6 horas

Desarrollo del módulo de usuarios y roles: 8 horas

Desarrollo del módulo de comunicación interna y notificaciones: 6 horas

Desarrollo del módulo de reportes operativos: 8 horas

Integración de módulos desarrollados: 6 horas

**Fase 5 - Pruebas y Validación: 22 horas**

Pruebas unitarias (cobertura >80%): 6 horas

Pruebas integradas (API, BD, módulos): 4 horas

Pruebas funcionales con Selenium: 4 horas

Pruebas de seguridad con OWASP ZAP: 2 horas

Pruebas de rendimiento con JMeter: 2 horas

Validación con usuarios clave (PepsiCo): 2 horas

Ajustes y correcciones según hallazgos: 2 horas (incluye tiempo de corrección de defectos)

**Fase 6 - Cierre del Proyecto: 52 horas**

Validación final por parte del cliente: 4 horas

Entrega de documentación completa (Acta, EDT, Gantt, RACI, Riesgos, ERS, etc.): 20 horas

Presentación final a PepsiCo y Duoc UC: 8 horas

Cierre administrativo y documentación de lecciones aprendidas: 20 horas

**TOTAL GENERAL ESTIMADO: 254 horas**

Se recomienda mantener una reserva de contingencia del 10% del tiempo total (25 horas adicionales) para:

Resolución de problemas técnicos imprevistos

Cambios menores en requerimientos aprobados por el cliente

Retrabajos por feedback del cliente en Sprint Reviews

Tiempo adicional en pruebas si se detectan defectos críticos

**Total con contingencia: 279 horas**

## Control de Costos

Aunque el proyecto no tiene costos monetarios directos para el cliente, es fundamental controlar el uso eficiente del tiempo y recursos disponibles para garantizar el cumplimiento de la fecha límite establecida (28 de noviembre de 2025).

*Mecanismos de Control.*

**Control de Tiempo:**

• Registro semanal de horas trabajadas por cada integrante del equipo

• Identificación temprana de desviaciones

• Ajuste de cronograma si es necesario, manteniendo fecha límite inamovible

• Reuniones semanales de seguimiento para evaluar avance

**Control de Alcance:**

• Evaluación de impacto de cualquier cambio solicitado en tiempo y recursos

• Priorización de funcionalidades críticas (MVP) sobre características adicionales

• Validación en Sprint Reviews cada 2 semanas

**Control de Recursos Tecnológicos:**

• Verificación semanal de que todas las herramientas siguen disponibles

• Identificación de herramientas alternativas en caso de limitaciones inesperadas

• Evaluación de rendimiento de herramientas elegidas (Angular, Node.js, MySQL)

• Backup de código en múltiples ubicaciones (GitHub + repositorios locales)

**Porcentaje de avance semanal:**

• Meta: ≥90% de tareas completadas en el tiempo estimado

• Seguimiento mediante Carta Gantt actualizada semanalmente

**Horas extras requeridas:**

• Fórmula: Horas trabajadas - Horas planificadas

• Meta: <10% de horas extras sobre lo planificado

• Monitoreo semanal para identificar sobrecarga

**Acciones Correctivas**

• Reunión de emergencia del equipo para análisis

• Redistribución de tareas entre integrantes según capacidades

• Priorización de funcionalidades críticas (reducción de alcance si es necesario)

• Incremento de horas de dedicación si es factible sin afectar calidad

• Comunicación inmediata al cliente (Alexis González) sobre riesgos de retraso

• Activación de reserva de contingencia

**Limitaciones de herramientas gratuitas:**

• Búsqueda inmediata de alternativas gratuitas equivalentes

• Ajuste de alcance técnico solo si es estrictamente necesario

**Reportes Semanales:**

• Registro de horas trabajadas y planificadas

• Estado de tareas según EDT y Carta Gantt

• Identificación de bloqueos o impedimentos

• Próximos hitos y entregables

## Línea Base de Costos

La línea base de costos representa el presupuesto de tiempo aprobado contra el cual se medirá el desempeño del proyecto.

**Línea Base Aprobada:** 254 horas de trabajo distribuidas en 12.6 semanas

**Fecha de inicio:** 1 de septiembre de 2025

**Fecha de finalización planificada:** 28 de noviembre de 2025

**Propósito de la Línea Base:**

Esta línea base es el punto de referencia autorizado para:

* Medir el avance del proyecto de acuerdo con el EDT y Diccionario EDT
* Identificar desviaciones en tiempo real
* Tomar decisiones de control y acciones correctivas
* Evaluar el desempeño del equipo de proyecto

**Control de Cambios a la Línea Base**

Cualquier cambio a esta línea base debe ser:

1. Solicitado formalmente por escrito mediante Solicitud de Cambio
2. Evaluado por el Gerente de Proyecto (Marco Peña) en términos de impacto en tiempo, alcance y calidad
3. Aprobado por el cliente (Alexis González - PepsiCo)
4. Documentado con acta de cambio que incluya:

* **Descripción del cambio solicitado**
* **Justificación del cambio**
* **Impacto en cronograma y recursos**
* **Firmas de aprobación**

1. Comunicado a todos los interesados identificados en la Matriz de Interesados

**Reserva de Contingencia (no incluida en línea base)**

25 horas (10% del total).

Esta reserva sólo podrá utilizarse con autorización del Gerente de Proyecto ante situaciones imprevistas que amenacen el cumplimiento de la fecha límite inamovible del 28 de noviembre de 2025. El uso de la reserva debe documentarse indicando:

* Motivo de activación
* Horas utilizadas
* Actividad/fase beneficiada
* Impacto en el cronograma general

## Herramientas y Técnicas

El proyecto Mec-IN utilizará las siguientes herramientas y técnicas para garantizar la calidad y el control de costos:

**Herramientas de Desarrollo**

Control de Versiones:

* Git/GitHub: Control de versiones distribuido con repositorio centralizado
* Estrategia de branching
* Política de commits: Mensajes descriptivos con referencia a tareas del EDT
* Pull Requests: Revisión de código obligatoria antes de merge

Entorno de Desarrollo:

* IDE: Visual Studio Code con extensiones para Angular y Node.js
* Node.js: v18+ para desarrollo del backend
* Angular CLI: v16+ para desarrollo del frontend
* MySQL Workbench: Diseño y administración de base de datos

**Herramientas de Pruebas**

Pruebas Unitarias:

* Jest: Framework de pruebas para Node.js
* Mocha/Chai: Alternativa para pruebas de backend
* Jasmine/Karma: Framework de pruebas para Angular

Pruebas de Integración:

* Postman: Pruebas manuales de API REST

Pruebas Funcionales:

* Selenium WebDriver: Automatización de pruebas de interfaz web
* Navegadores: Chrome, Firefox, Edge (compatibilidad según ERS)

Pruebas de Seguridad:

* OWASP ZAP: Escaneo automático de vulnerabilidades web
* npm audit: Detección de vulnerabilidades en dependencias

Pruebas de Rendimiento:

* Apache JMeter: Pruebas de carga y estrés (35 usuarios concurrentes)
* Locust: Alternativa para pruebas de carga (opcional)
* Google Lighthouse: Auditoría de rendimiento, accesibilidad y SEO

**Herramientas de Gestión de Proyecto**

Planificación y Seguimiento:

* Microsoft Excel/Google Sheets: Carta Gantt, matrices (RACI, Riesgos, Interesados)
* Reuniones Sprint: Daily Stand-ups, Sprint Planning, Sprint Review, Sprint Retrospective

Documentación:

* Microsoft Word/Google Docs: Documentos del proyecto (Acta, ERS, Planes)
* Draw.io/Lucidchart: Diagramas UML, casos de uso, arquitectura
* Figma: Diseño de mockups e interfaces

Comunicación:

* WhatsApp/Email: Comunicación con cliente y equipo
* Microsoft Teams/Zoom: Reuniones virtuales con cliente
* GitHub Issues: Seguimiento de tareas y defectos

**Técnicas de Gestión de Calidad**

Técnicas de Aseguramiento de Calidad:

* Revisiones por pares: Revisión de código entre integrantes
* Inspecciones de diseño: Validación de arquitectura y modelos de datos
* Auditorías de proceso: Verificación de cumplimiento de estándares
* Análisis estático de código: Detección temprana de errores y malas prácticas

Técnicas de Control de Calidad:

* Testing exploratorio: Pruebas manuales no planificadas para descubrir defectos
* Regression testing: Re-ejecución de pruebas tras cambios
* Análisis de causa raíz: Investigación de defectos críticos

Técnicas de Estimación:

* Juicio de expertos: Estimación basada en experiencia del equipo
* Estimación análoga: Comparación con proyectos similares
* Estimación paramétrica: Estimación por puntos de función o líneas de código

# Aprobaciones

Este Plan de Calidad y Costos ha sido revisado y aprobado por las siguientes personas:

| **Rol** | **Nombre** | **Firma** | **Fecha** |
| --- | --- | --- | --- |
| Gerente de Proyecto | Marco Peña | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | *//2025* |
| Gerente TI | Benjamín Díaz | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | *//2025* |
| Cliente / Patrocinador | Alexis González (PepsiCo Chile) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | *//2025* |

## Historial de revisiones

| **Versión** | **Fecha** | **Descripción del Cambio** | **Autor** |
| --- | --- | --- | --- |
| 1.0 | 07-10-2025 | Creación del documento completo | Marco Peña, Benjamín Díaz |
| 2.0. | 08-10-2025 | Validación de la información del documento. | Marco Peña. |

# Referencias

Este Plan de Calidad y Costos se basa en los siguientes documentos del proyecto Mec-IN:

1. Acta de Constitución del Proyecto
2. Diccionario EDT - Estructura de Desglose del Trabajo
3. Carta Gantt - Cronograma del Proyecto
4. Matriz RACI - Matriz de Responsabilidades
5. Matriz de Riesgos - Identificación y análisis de riesgos del proyecto
6. Matriz de Interesados - Stakeholders del proyecto
7. Documento de Especificación de Requerimientos (ERS) - Requerimientos funcionales (RF-01 a RF-26), no funcionales (RNF-01 a RNF-11), de sistema (RS-01 a RS-07) y reglas de negocio (RB-01 a RB-20)
8. Plan de Pruebas Inicial - Estrategia de pruebas del proyecto
9. Casos de Uso (UML) - Descripción de funcionalidades del sistema
10. Script de Base de Datos - Diseño de la base de datos MySQL
11. Caso de Estudio Capstone - Contexto académico del proyecto (Duoc UC)

**Este documento es un instrumento vivo que puede ser actualizado durante el ciclo de vida del proyecto. Cualquier modificación debe seguir el proceso de control de cambios establecido en la sección 3.4 (Línea Base de Costos).**