Emulador de microprocesador Leon3

Autor:
Iriarte Fernandez, Nicolás Ezequiel

Director:
Horro, Nicolás Eduardo (INVAP.S.E.)

Esta planificación fue realizada en el curso de Gestión de proyectos entre el 29/08/2023 y el 10/10/2023.

Índice

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	29/08/2023
1	Se completa hasta el punto 5 inclusive	07/09/2023

Acta de constitución del proyecto

Buenos Aires, 29/08/2023

Por medio de la presente se acuerda con el Ing. Iriarte Fernandez, Nicolás Ezequiel que su Trabajo Final de la Carrera de Especialización en Sistemas Embebidos se titulará "Emulador de microprocesador Leon3", consistirá esencialmente en la implementación de un prototipo de un emulador de Leon3 para desarrollo de SW satelital y simuladores, y tendrá un presupuesto preliminar estimado de 600 h de trabajo, con fecha de inicio 29/08/2023 y fecha de presentación pública 30/04/2024.

Se adjunta a esta acta la planificación inicial.

Dr. Ing. Ariel Lutenberg
Director posgrado FIUBA

Pinedo, Matías
INVAP.SE

Horro, Nicolás Eduardo
Director del Trabajo Final

1. Descripción técnica-conceptual del proyecto a realizar

Para productos de ámbitos espaciales, como lo son los satélites, muchas veces es difícil, y en ocasiones imposible, generar escenarios realistas para pruebas de los elementos que los componen. Ya sea por no poder generar las mismas condiciones ambientales, o porque la naturaleza de la maniobra que se busca probar implicaría un daño a los equipos bajo revisión.

En este contexto, es común replicar los elementos de interés de manera programada, cumpliendo con cierto grado de representación. De manera tal que se comporten de la manera más similar posible a su contraparte física. Estos elementos, íntegramente desarrollados en software, se los llaman emulados o simulados. Uno de los componentes que se suele tener mayor interés en simular es el procesador de la computadora a bordo.

El término Computadora a Bordo (OBC, por las siglas en inglés de On Board Computer) suele referirse a la unidad en la que se ejecuta el Software A Bordo (OBSW, por sus siglas en inglés de On Board Software) y su rol principal es el control de los subsistemas del satélite. Esto incluye recolectar información de diferentes subsistemas, analizarla y tomar las decisiones y acciones apropiadas cuando sea requerido.

El foco principal de este trabajo es el microprocesador contenido en la OBC. Cabe destacar, que hoy en día existen emuladores tanto de código abierto como privativos para distintos microprocesadores. Un ejemplo claro de emulador de código abierto es Qemu, que abarca un amplio abanico de microprocesadores, entre ellos, algunos utilizables en el ámbito espacial.

En el contexto de los emuladores, cada uno aborda el problema en cuestión, pero conlleva sus propias desventajas. Por ejemplo, en el caso de emuladores como Qemu, diseñados para ser genéricos, pueden requerir un esfuerzo adicional para adaptarlos a un procesador específico, como es el GR712RC. Esto se debe a que la flexibilidad genérica puede implicar una pérdida de rendimiento o la necesidad de familiarizarse con su API.

Por otro lado, los emuladores comerciales suelen estar optimizados para procesadores específicos, lo que puede brindar un excelente rendimiento en esas circunstancias particulares. Sin embargo, estos emuladores pueden ser limitados en cuanto a su capacidad de personalización, instrumentación y capacidad de integración, lo que puede dificultar su adaptación a necesidades específicas o la depuración de software en entornos no nominales.

Bajo estas premisas se plantea crear un emulador de microprocesador Leon3 para desarrollo de software satelital y simuladores. En la Figura 1 se deja en evidencia un diagráma de bloques del hardware que se busca replicar. Al ser un desarrollo a medida, se tendrá la ventaja de la no-diversificación del procesador, es decir, estará únicamente orientado a un solo microprocesador. Esperando una ganancia en performance comparado con su contraparte de código abierto. Al mismo tiempo, se tendrá un conocimiento extenso del alcance y limitaciones de las capacidades del software en cuestión. Haciendo, de esta manera, más simple la integración y depuración en su uso.

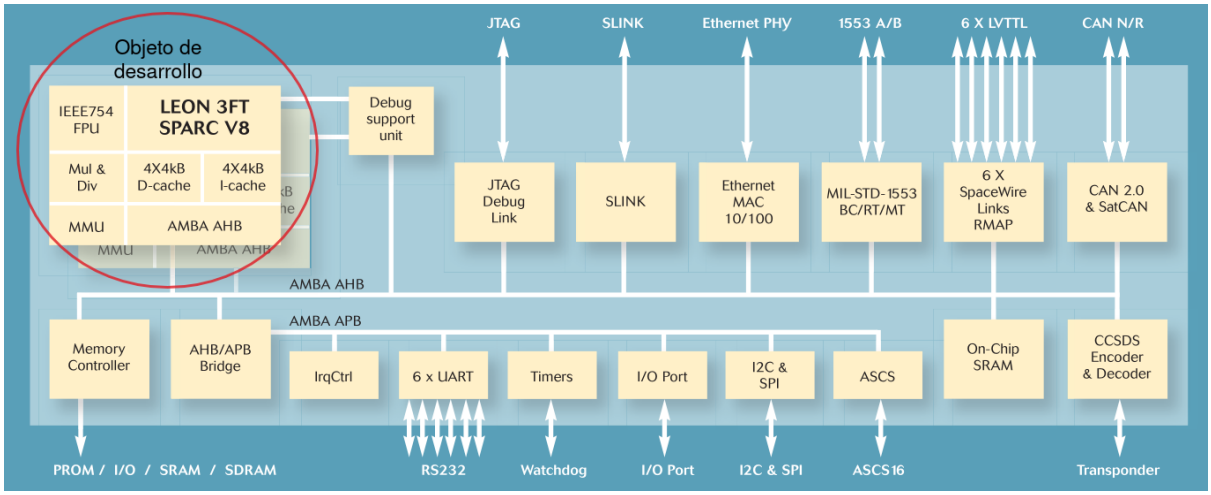

Figura 1. Diagrama en bloques del sistema

Para lograr el objetivo descrito, se ha decidido utilizar el framework LLVM. Este se empleará para generar el conjunto de instrucciones admitido por el microprocesador Leon3. La elección de LLVM se fundamenta en varios factores clave que hacen de esta plataforma una opción sólida y versátil. Tales como que cuenta con una comunidad de desarrollo activa y comprometida, lo que garantiza un soporte continuo y actualizaciones regulares que son esenciales para mantener la robustez y eficiencia del sistema. También, al poseer soporte nativo con Leon3, es una fuente de análisis y guía en los casos donde los manuales no sean lo suficientemente claros.

Por último, es importante subrayar que el presente trabajo no se iniciará desde cero, sino que se aprovechará una base de código preexistente gentilmente proporcionada por el director de tesis. Esta base no solo servirá como una hoja de guía para el proceso de desarrollo, sino que también desempeñará un papel clave en la aceleración de todo el proyecto.

2. Identificación y análisis de los interesados

Rol	Nombre y Apellido	Organización	Puesto
Cliente	Lic. Pinedo, Matías	INVAP.SE	Especialista Tecnológico
Responsable	Iriarte Fernandez, Nicolás Ezequiel	FIUBA	Alumno
Orientador	Esp. Lic. Horro, Nicolás Eduardo	INVAP.S.E.	Director Trabajo final
Usuario final	Departamento de Embebidos y Sistemas Críticos	INVAP.S.E.	-

- Cliente: Lic. Matías Pinedo, del Departamento de Embebidos y Sistemas Críticos de la empresa INVAP S.E., quien aporta su visión y experiencia necesaria para desarrollar el proyecto.
- Orientador: Es el director del trabajo final, quien aportará sus conocimientos técnicos y experiencia para la realización del proyecto.
- Usuario final: será el Departamento de Embebidos y Sistemas Críticos de la empresa INVAP S.E.

3. Propósito del proyecto

El propósito de este proyecto es desarrollar un emulador para ejecución de software de vuelo para microprocesadores de la familia Leon3 utilizando el framework LLVM a partir de un código preexistente provisto.

4. Alcance del proyecto

El presente trabajo incluye:

- El desarrollo del software de emulación.
- Una API para su uso en el lenguaje de programación C.
- El desarrollo de un sistema de depuración para una ejecución.

El proyecto no incluye:

- Desarrollo de software para el procesador Leon3 representativo con el uso que se le piense dar.
- Librerías utilitarías para generación de código fuente.
- Emulación completa sobre todos periféricos asociados.

5. Supuestos del proyecto

Para el desarrollo del presente proyecto se supone que:

- La empresa va a proveer de un modelo de referencia (físico o emulado) para la realización del proyecto.
- El proyecto puede ser realizado sin necesidad de hacer pruebas en campo.
- Se contará con soporte y apoyo por parte de expertos de la empresa.
- El proyecto puede ser realizado sin necesidad de ir a las oficinas de la empresa.

6. Requerimientos

1. Requerimientos funcionales
   1.1. El emulador deberá poder cargar y ejecutar binarios valídos para Leon3, tales como los que se cargarían al hardware real.
   1.2. Deberá poder ejecutarse en Linux.
   1.3. Se deberá desarrollar un ambiente de automatización de pruebas en Gitlab CI.

1.4. Se deberán desarrollar tests unitarios que verifiquen parte de set de instrucciones del emulador.

1.5. Se deberá poder utilizar el software como librería compartida.

2. Requerimientos de documentación

2.1. La API que será expuesta como librería deberá estár documentada con Doxygen.

2.2. Se realizará un manual de usuario que describa los funcionamientos clave del software.

3. Requerimientos de desarrollo de software

3.1. El software deberá ser escrito en C++.

3.2. La API expuesta deberá ser en C.

3.3. El software debe mantenerse bajo control de versiones en Gitlab.

7. Historias de usuarios (*Product backlog*)

Roles:

- Desarrollador de modelos simulados: quien se encarga de desarrollar dispositivos de manera simulada. Tales como podrían ser memorias, FPGAs y/o periféricos.
- Desarrollador de software de vuelo: quien se encarga de desarrollar un driver o software que usa interfaces de bajo nivel. Durante el diseño e implementación, utiliza este sistema para interactuar con el hardware.
- Ingeniero de pruebas: quien se encarga de automatizar los ensayos de calificación de hardware y/o de componentes que lo emulen.

Story Points:

- Se analizarán las historias según dificultad, complejidad e incertidumbre, tomando valores de Fibonacci usando el siguiente criterio:
  1. Dificultad: cantidad de trabajo a realizar. Representado con la letra "**D**".
     - Bajo: 1
     - Medio: 3
     - Alto: 13
  2. Complejidad: complejidad de trabajo a realizar. Representado con la letra "**C**".
     - Bajo: 1
     - Medio: 5
     - Alto: 13
  3. Incertidumbre: Riesgo del trabajo a realizar. Representado con la letra "**I**".
     - Bajo: 1
     - Medio: 3
     - Alto: 5

Historias de Usuarios:

- Como desarrollador de modelos simulados quiero poder comunicarme con el software de vuelo. D: 3, C: 5, I: 5 Total = 13p.
- Como desarrollor de modelos simulados quiero poder ejecutar el software de vuelo como una librería. D: 3, C: 1, I: 1. Total = 5p.
- Como desarrollador de software de vuelo quiero poder estimular los registros, memorias y hardware para el desarrollo de controladores (*drivers*). D: 13, C: 13, I: 5. Total = 34p.
- Como desarrollador de software de vuelo quiero que mi software corra a tiempo real. D: 5, C: 5, I: 5. Total = 21p.
- Como ingeniero de pruebas quiero que mis pruebas puedan ser ejecutadas automáticamente en un entonces de desarrollo continuo (por ejemplo GitLab CI). D: 3, C: 2, I: 2. Total = 8p.

8. Entregables principales del proyecto

Los entregables del proyecto son:

- Librería de emulación con sus respectivos headers.
- Documentación Doxygen de la API.
- Manual de uso.

9. Desglose del trabajo en tareas

1. Gestión del proyecto (136 hs):
   1.1. Definición de requerimientos con el cliente (4 hs).
   1.2. Planificación (20 hs).
   1.3. Preparación de las memorias (40 hs).
   1.4. Redacción de las memorias (40 hs).
   1.5. Presentación del trabajo (32 hs).
2. Investigación y capacitación en la arquitectura del microprocesador (33 hs):
   2.1. Recopilación de documentación online (3 hs).
   2.2. Lectura y comprensión de los documentos (10 hs).
   2.3. Capacitación sobre el conjunto de instrucciones del procesador (20 hs).
3. Desarrollo de ambiente de automatización (16 hs):
   3.1. Creación de repositorio de desarrollo (2 hs).
   3.2. Creación de ambiente de CI/CD para tests (4 hs).

3.3. Desarrollo de ambiente CI/CD para documentación (6 hs).
3.4. Creación de ambiente de CI/CD para empaquetado (4 hs).

4. Internalización de código pre-existente (46 hs):
   4.1. Revisión y análisis de código existente (12 hs).
   4.2. Selección de código a reutilizar e importación (30 hs)
   4.3. Refactorización de código (4 hs)

5. Investigación y capacitación en el framework de LLVM (50 hs):
   5.1. Lectura de documentación (20 hs).
   5.2. Internalización con su API (20 hs).
   5.3. Pruebas de limites y alcances del framework (10 hs).

6. Desarrollo del software (180 hs):
   6.1. Desarrollo de modelos de periféricos (40 hs).
   6.2. Desarrollo de modelos de Memorias y registros (30 hs).
   6.3. Desarrollo de intrucciones. Se estima funcionalidad minima de un subset de instrucciones, calculando 1 hs por instrucción. (110 hs)

7. Desarrollo de tests unitarios y de API (90 hs):
   7.1. Pruebas unitarias de modelos (20 hs).
   7.2. Pruebas unitarias de intrucciones (40 hs).
   7.3. Pruebas de API (20 hs).
   7.4. Corrección de errores (10 hs).

8. Documentación (50 hs):
   8.1. Creación del manual de usuario (40 hs).
   8.2. Documentación de API en Doxygen (10 hs).

Cantidad total de horas: hs

10. Diagrama de Activity On Node

Armar el AoN a partir del WBS definido en la etapa anterior.

Indicar claramente en qué unidades están expresados los tiempos. De ser necesario indicar los caminos semicríticos y analizar sus tiempos mediante un cuadro. Es recomendable usar colores y un cuadro indicativo describiendo qué representa cada color, como se muestra en el siguiente ejemplo:

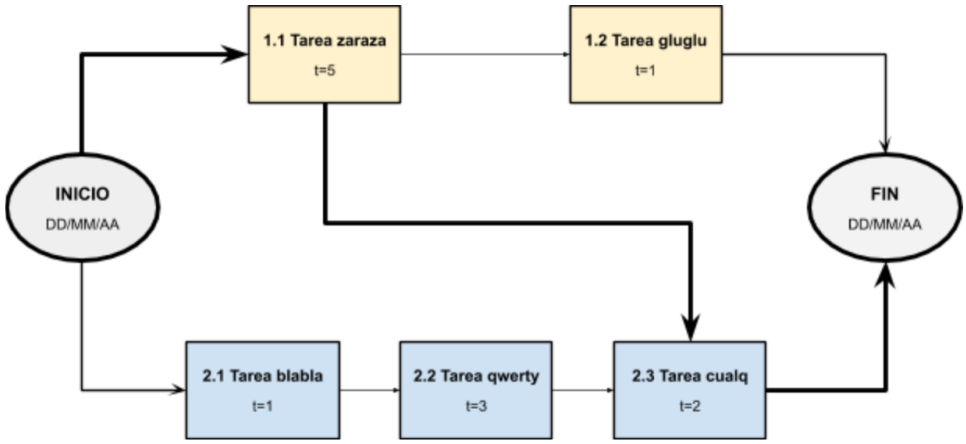

Figura 2. Diagrama de *Activity on Node*.

11. Diagrama de Gantt

Existen muchos programas y recursos *online* para hacer diagramas de Gantt, entre los cuales destacamos:

- Planner
- GanttProject
- Trello + *plugins*. En el siguiente link hay un tutorial oficial:
  `https://blog.trello.com/es/diagrama-de-gantt-de-un-proyecto`
- Creately, herramienta online colaborativa.
  `https://creately.com/diagram/example/ieb3p3ml/LaTeX`
- Se puede hacer en latex con el paquete *pgfgantt*
  `http://ctan.dcc.uchile.cl/graphics/pgf/contrib/pgfgantt/pgfgantt.pdf`

Pegar acá una captura de pantalla del diagrama de Gantt, cuidando que la letra sea suficientemente grande como para ser legible. Si el diagrama queda demasiado ancho, se puede pegar primero la "tabla" del Gantt y luego pegar la parte del diagrama de barras del diagrama de Gantt.

Configurar el software para que en la parte de la tabla muestre los códigos del EDT (WBS).
Configurar el software para que al lado de cada barra muestre el nombre de cada tarea.
Revisar que la fecha de finalización coincida con lo indicado en el Acta Constitutiva.

En la figura 3, se muestra un ejemplo de diagrama de Gantt realizado con el paquete de *pgfgantt*. En la plantilla pueden ver el código que lo genera y usarlo de base para construir el propio.

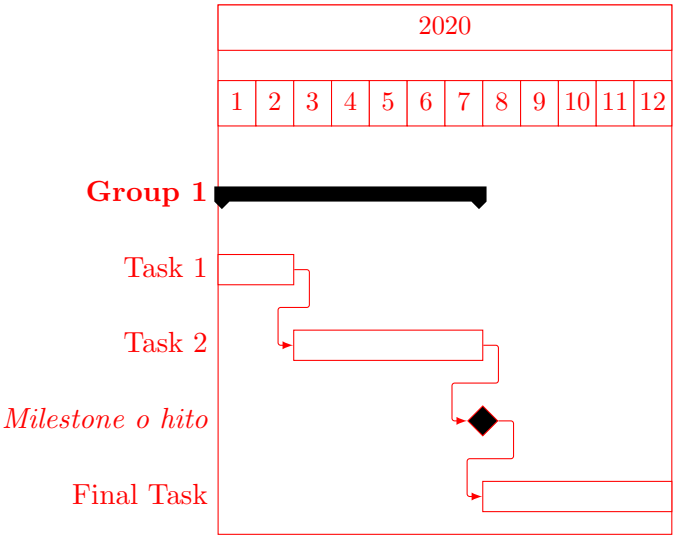

Figura 3. Diagrama de Gantt de ejemplo

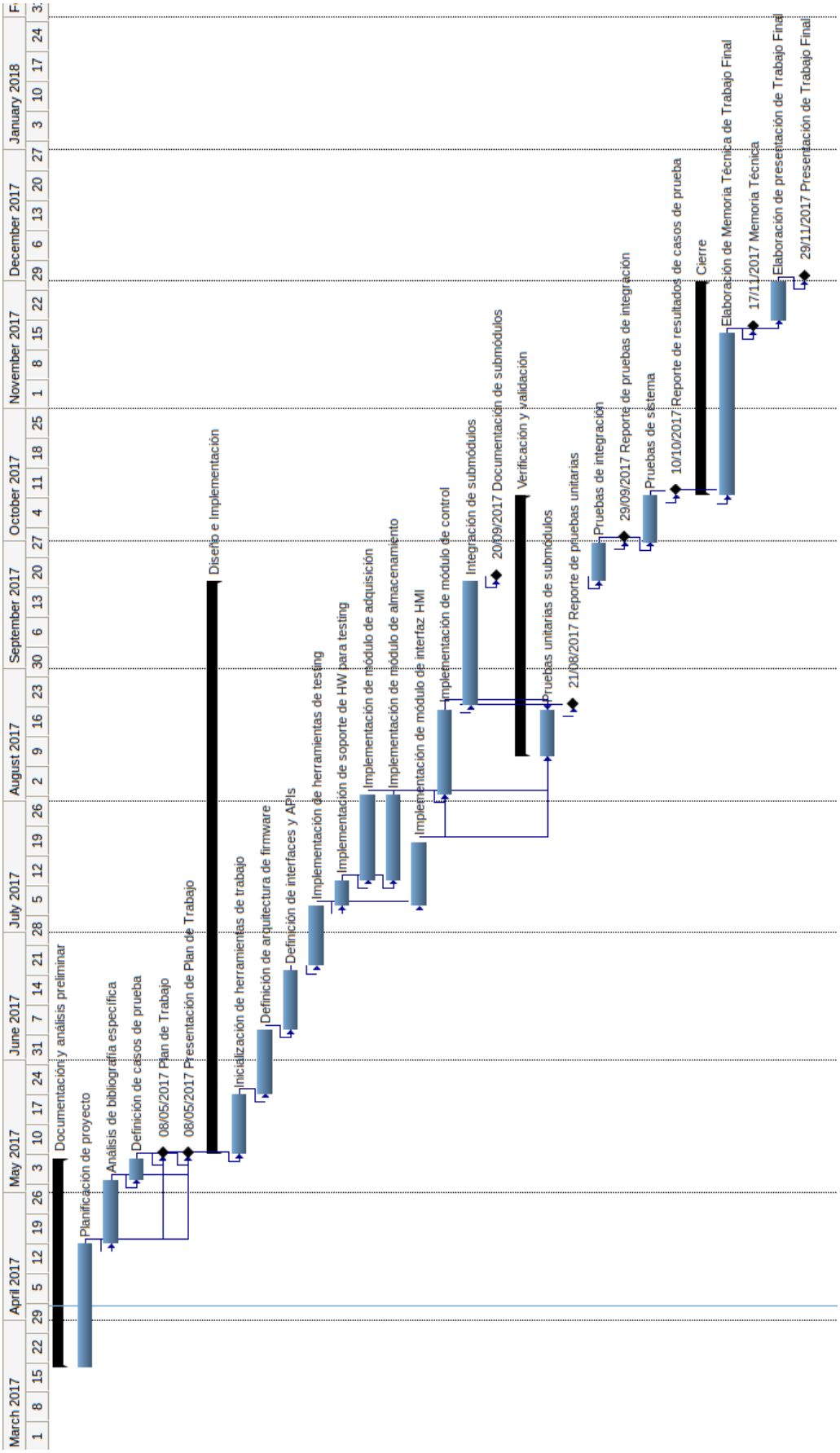

Figura 4. Ejemplo de diagrama de Gantt rotado

12. Presupuesto detallado del proyecto

Si el proyecto es complejo entonces separarlo en partes:

- Un total global, indicando el subtotal acumulado por cada una de las áreas.
- El desglose detallado del subtotal de cada una de las áreas.

IMPORTANTE: No olvidarse de considerar los COSTOS INDIRECTOS.

COSTOS DIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
COSTOS INDIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
SUBTOTAL			
TOTAL			

13. Gestión de riesgos

a) Identificación de los riesgos (al menos cinco) y estimación de sus consecuencias:

Riesgo 1: detallar el riesgo (riesgo es algo que si ocurre altera los planes previstos de forma negativa)

- Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de severidad (S).
- Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10).
  Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2:

- Severidad (S):
- Ocurrencia (O):

Riesgo 3:

- Severidad (S):

- Ocurrencia (O):

b) Tabla de gestión de riesgos: (El RPN se calcula como RPN=SxO)

Riesgo	S	O	RPN	S*	O*	RPN*

Criterio adoptado: Se tomarán medidas de mitigación en los riesgos cuyos números de RPN sean mayores a...

Nota: los valores marcados con (*) en la tabla corresponden luego de haber aplicado la mitigación.

c) Plan de mitigación de los riesgos que originalmente excedían el RPN máximo establecido:

Riesgo 1: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación). Nueva asignación de S y O, con su respectiva justificación: - Severidad (S): mientras más severo, más alto es el número (usar números del 1 al 10). Justificar el motivo por el cual se asigna determinado número de severidad (S). - Probabilidad de ocurrencia (O): mientras más probable, más alto es el número (usar del 1 al 10). Justificar el motivo por el cual se asigna determinado número de (O).

Riesgo 2: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

Riesgo 3: plan de mitigación (si por el RPN fuera necesario elaborar un plan de mitigación).

14. Gestión de la calidad

Elija al menos diez requerientos que a su criterio sean los más importantes/críticos/que aportan más valor y para cada uno de ellos indique las acciones de verificación y validación que permitan asegurar su cumplimiento.

- Req #1: copiar acá el requerimiento.
  - Verificación para confirmar si se cumplió con lo requerido antes de mostrar el sistema al cliente. Detallar
  - Validación con el cliente para confirmar que está de acuerdo en que se cumplió con lo requerido. Detallar

Tener en cuenta que en este contexto se pueden mencionar simulaciones, cálculos, revisión de hojas de datos, consulta con expertos, mediciones, etc. Las acciones de verificación suelen considerar al entregable como “caja blanca”, es decir se conoce en profundidad su funcionamiento interno. En cambio, las acciones de validación suelen considerar al entregable como “caja negra”, es decir, que no se conocen los detalles de su funcionamiento interno.

15. Procesos de cierre

Establecer las pautas de trabajo para realizar una reunión final de evaluación del proyecto, tal que contemple las siguientes actividades:

- Pautas de trabajo que se seguirán para analizar si se respetó el Plan de Proyecto original: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento a aplicar.
- Identificación de las técnicas y procedimientos útiles e inútiles que se emplearon, y los problemas que surgieron y cómo se solucionaron: - Indicar quién se ocupará de hacer esto y cuál será el procedimiento para dejar registro.
- Indicar quién organizará el acto de agradecimiento a todos los interesados, y en especial al equipo de trabajo y colaboradores: - Indicar esto y quién financiará los gastos correspondientes.