DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

.UBAfiuba
FACULTAD DE INGENIERÍA

Modelo de migración tecnológica para radares

Autor:
Gonzalo Nahuel Vaca

Director:
Iván Andrés León Vásquez (INVAP S.E.)

Esta planificación fue realizada en el curso de Gestión de proyectos entre el 29 de junio de 2022 y el 17 de agosto de 2022.

.UBAfiuba
FACULTAD DE INGENIERÍA

Modelo de migración tecnológica para radares

Autor:
Gonzalo Nahuel Vaca

Director:
Iván Andrés León Vásquez (INVAP S.E.)

Esta planificación fue realizada en el curso de Gestión de la tecnología y la innovación entre el 29 de junio de 2022 y el 17 de agosto de 2022.

Índice

Índice

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	29 de junio de 2022
1	Se completa el documento salvo el diagrama de GANTT	10/07/2022

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

Registros de cambios

Revisión	Detalles de los cambios realizados	Fecha
0	Creación del documento	29/06/2022
1	Se completa el documento salvo el diagrama de GANTT	10/07/2022
2	Se agrega el diagrama de GANTT	17/07/2022

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

Acta de constitución del proyecto

Buenos Aires, 29 de junio de 2022

Por medio de la presente se acuerda con el Mg. Ing. Gonzalo Nahuel Vaca que su Trabajo Final de la Maestría en Sistemas Embebidos se titulará "Modelo de migración tecnológica para radares", consistirá esencialmente en la implementación de un prototipo de un procesador de datos de radar hecho en Python, y tendrá un presupuesto preliminar estimado de 600 hs de trabajo y $200, con fecha de inicio 29 de junio de 2022 y fecha de presentación pública 15 de mayo de 2023.

Se adjunta a esta acta la planificación inicial.

Ariel Lutenberg
Director posgrado FIUBA

Andrés Eduardo Monteverde
INVAP S.E.

Iván Andrés León Vásquez
Director del Trabajo Final

.UBAfiuba FACULTAD DE INGENIERÍA

Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

Acta de constitución del proyecto

Buenos Aires, 29 de junio de 2022

Por medio de la presente se acuerda con el Mg. Ing. Gonzalo Nahuel Vaca que su Trabajo Final de la Maestría en Sistemas Embebidos se titulará "Modelo de migración tecnológica para radares", consistirá esencialmente en la implementación de un prototipo de un procesador de datos de radar hecho en Python, y tendrá un presupuesto preliminar estimado de 600 hs de trabajo y $200, con fecha de inicio 29 de junio de 2022 y fecha de presentación pública 15 de mayo de 2023.

Se adjunta a esta acta la planificación inicial.

Ariel Lutenberg
Director posgrado FIUBA

Andrés Eduardo Monteverde
INVAP S.E.

Iván Andrés León Vásquez
Director del Trabajo Final

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

1. Descripción técnica-conceptual del proyecto a realizar

El proyecto a realizar tiene como cliente al Departamento de ingeniería de software (DISW) de INVAP S.E. en el marco de la Gerencia de gobierno y defensa.

En la cartera de productos de la empresa se pueden encontrar radares meteorológicos, de vigilancia primarios y secundarios. Estos equipos generan un gran volumen de datos que deben ser procesados por algoritmos complejos con la finalidad de obtener información útil.

Los algoritmos son generados por profesionales especializados en la naturaleza física y matemática del problema y entregan como producto un modelo de procesamiento de datos en Python. Luego, el modelo debe correr dentro de un sistema embebido multiprocesador asimétrico (MPSoC). Entonces, se procede a traducir el código al lenguaje de programación C para lograr su compilación en la arquitectura objetivo.

La traducción del modelo genera los siguientes inconvenientes:

- Duplicación de tareas.
- Dificultad para depurar: se torna difícil determinar si un comportamiento erróneo es producto del diseño del modelo o su traducción al lenguaje C.
- Baja de confiabilidad: la traducción del código es una fuente de posibles errores en el sistema.

Con la intención de atenuar estos problemas, se deberá realizar un prototipo que utilice el algoritmo hecho en Python sin traducir. Esto tendría el beneficio adicional de aliviar la demanda de programadores C/C++ y buscar personal en una población mayor.

El proyecto se realizará en un *System on chip* (SoC) Zynq-7000 del fabricante AMD/Xilinx y tendrá los siguientes componentes:

- Generador de datos de radar: se deberá sintetizar en la lógica programable del integrado.
- Modelo procesador: tendrá que estar realizado en Python y podrá ser configurado desde el exterior.
- Generador de mensajes ASTERIX: deberá crear los mensajes ASTERIX a partir de la información generada.
- Generador de telemetría: deberá reportar el estado de funcionamiento del sistema con mensajes MQTT.
- Backend radar corriendo como servicio.
- Creación de una distribución de Linux embebido.

En la figura 1 se puede observar un diagrama en bloques simplificado del sistema propuesto.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

1. Descripción técnica-conceptual del proyecto a realizar

El proyecto a realizar tiene como cliente al Departamento de ingeniería de software (DISW) de INVAP S.E. en el marco de la Gerencia de gobierno y defensa.

En la cartera de productos de la empresa se pueden encontrar radares meteorológicos, de vigilancia primarios y secundarios. Estos equipos generan un gran volumen de datos que deben ser procesados por algoritmos complejos con la finalidad de obtener información útil.

Los algoritmos son generados por profesionales especializados en la naturaleza física y matemática del problema y entregan como producto un modelo de procesamiento de datos en Python. Luego, el modelo debe correr dentro de un sistema embebido multiprocesador asimétrico (MPSoC). Entonces, se procede a traducir el código al lenguaje de programación C para lograr su compilación en la arquitectura objetivo.

La traducción del modelo genera los siguientes inconvenientes:

- Duplicación de tareas.
- Dificultad para depurar: se torna difícil determinar si un comportamiento erróneo es producto del diseño del modelo o su traducción al lenguaje C.
- Baja de confiabilidad: la traducción del código es una fuente de posibles errores en el sistema.

Con la intención de atenuar estos problemas, se deberá realizar un prototipo que utilice el algoritmo hecho en Python sin traducir. Esto tendría el beneficio adicional de aliviar la demanda de programadores C/C++ y buscar personal en una población mayor.

El proyecto se realizará en un *System on chip* (SoC) Zynq-7000 del fabricante AMD/Xilinx y tendrá los siguientes componentes:

- Generador de datos de radar: se deberá sintetizar en la lógica programable del integrado.
- Modelo procesador: tendrá que estar realizado en Python y podrá ser configurado desde el exterior.
- Generador de mensajes ASTERIX: deberá crear los mensajes ASTERIX a partir de la información generada.
- Generador de telemetría: deberá reportar el estado de funcionamiento del sistema con mensajes MQTT.
- Backend radar corriendo como servicio.
- Creación de una distribución de Linux embebido.

En la figura 1 se puede observar un diagrama en bloques simplificado del sistema propuesto.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

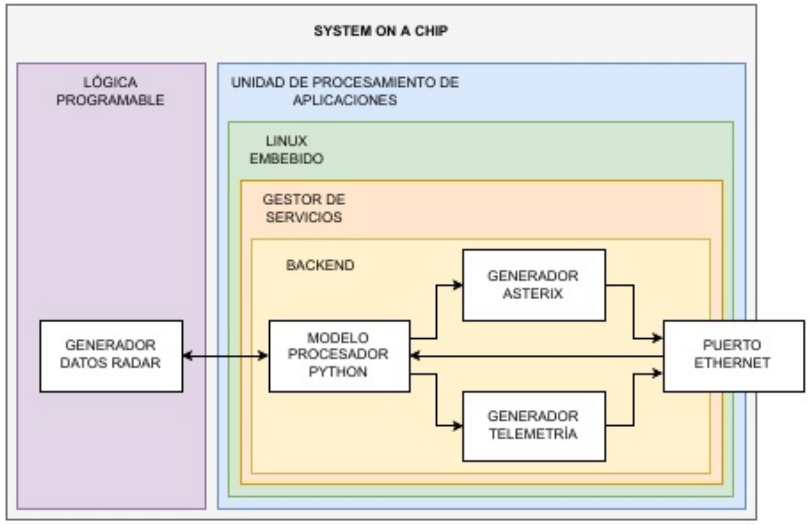

Figura 1. Diagrama en bloques del sistema

2. Identificación y análisis de los interesados

Rol	Nombre y Apellido	Organización	Puesto
Auspíciate	Celeste Musso	INVAP S.E.	Jefa del DISW
Cliente	Andrés Eduardo Monteverde	INVAP S.E.	Ingeniero de software
Impulsor	Maximiliano Graf	INVAP S.E.	Líder de equipo
Responsable	Gonzalo Nahuel Vaca	FIUBA	Alumno
Orientador	Iván Andrés León Vásquez	INVAP S.E.	Director del Trabajo final

3. Propósito del proyecto

El propósito de este proyecto es evaluar la posibilidad de realizar un backend para radares hecho en Python. Con la finalidad de evitar la repetición de tareas, mejorar la confiabilidad de los sistemas y acceder a una población mayor de programadores.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

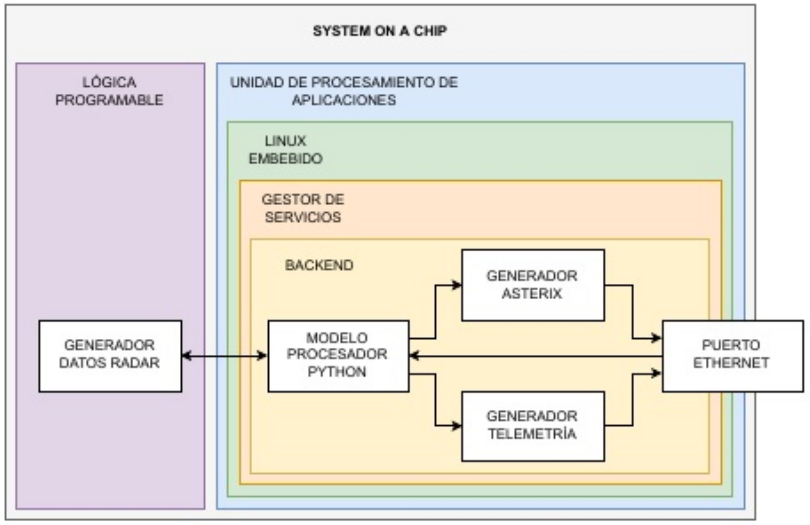

Figura 1. Diagrama en bloques del sistema

2. Identificación y análisis de los interesados

Rol	Nombre y Apellido	Organización	Puesto
Auspiciante	Celeste Musso	INVAP S.E.	Jefa del DISW
Cliente	Andrés Eduardo Monteverde	INVAP S.E.	Ingeniero de software
Impulsor	Maximiliano Graf	INVAP S.E.	Líder de equipo
Responsable	Gonzalo Nahuel Vaca	FIUBA	Alumno
Orientador	Iván Andrés León Vásquez	INVAP S.E.	Director del Trabajo final

3. Propósito del proyecto

El propósito de este proyecto es evaluar la posibilidad de realizar un backend para radares hecho en Python. Con la finalidad de evitar la repetición de tareas, mejorar la confiabilidad de los sistemas y acceder a una población mayor de programadores.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

4. Alcance del proyecto

El presente proyecto incluye:

- Crear un esclavo AXI generador de datos de radar.
- Crear una distribución de Linux embebido.
- Crear un servicio backend radar hecho en Python.
- Documentación y manual de usuario.

5. Supuestos del proyecto

Para el desarrollo del presente proyecto se supone que:

- Se tendrá acceso a los modelos procesadores de datos de radar.
- Se tendrá acceso una captura de datos de radar.

6. Requerimientos

A continuación se enumeran los requerimientos del proyecto:

1. Requerimientos funcionales
   - 1.1. El sistema se debe implementar en un SoC de Xilinx.
   - 1.2. El sistema debe generar un flujo de datos de radar.
   - 1.3. El generador de datos debe ser un esclavo AXI en la lógica programable.
   - 1.4. La generación de datos debe ser configurable.
   - 1.5. El sistema debe procesar los datos y generar mensajes ASTERIX.
   - 1.6. El procesador de datos se debe implementar en Python dentro de la Unidad de procesamiento de aplicaciones.
   - 1.7. El procesador debe correr como servicio dentro del sistema operativo.
   - 1.8. Se debe crear una distribución de Linux embebido.
   - 1.9. Los mensajes ASTERIX se deben transmitir por protocolo ZMQ.
   - 1.10. El sistema debe generar datos de telemetría que indiquen su estado de funcionamiento.
   - 1.11. La telemetría se debe trasmitir por protocolo MQTT.
   - 1.12. El sistema debe presentar una interfaz web para ajustar la configuración.
   - 1.13. Se debe poder visualizar los mensajes ASTERIX y de telemetría.
2. Requerimientos de documentación
   - 2.1. Se debe generar un informe de rendimiento del procesador de datos implementado.
   - 2.2. Se debe crear un manual de uso para el esclavo AXI.
   - 2.3. Se debe crear un manual de uso para el servicio procesador de datos.
3. Se debe proveer una interfaz web para supervisar y configurar el sistema.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

4. Alcance del proyecto

El presente proyecto incluye:

- Crear un esclavo AXI generador de datos de radar.
- Crear una distribución de Linux embebido.
- Crear un servicio backend radar hecho en Python.
- Documentación y manual de usuario.

5. Supuestos del proyecto

Para el desarrollo del presente proyecto se supone que:

- Se tendrá acceso a los modelos procesadores de datos de radar.
- Se tendrá acceso a una captura de datos de radar.

6. Requerimientos

A continuación se enumeran los requerimientos del proyecto:

1. Requerimientos funcionales
   - 1.1. El sistema se debe implementar en un SoC de Xilinx.
   - 1.2. El sistema debe generar un flujo de datos de radar.
   - 1.3. El generador de datos debe ser un esclavo AXI en la lógica programable.
   - 1.4. La generación de datos debe ser configurable.
   - 1.5. El sistema debe procesar los datos y generar mensajes ASTERIX.
   - 1.6. El procesador de datos se debe implementar en Python dentro de la Unidad de procesamiento de aplicaciones.
   - 1.7. El procesador debe correr como servicio dentro del sistema operativo.
   - 1.8. Se debe crear una distribución de Linux embebido.
   - 1.9. Los mensajes ASTERIX se deben transmitir por protocolo ZMQ.
   - 1.10. El sistema debe generar datos de telemetría que indiquen su estado de funcionamiento.
   - 1.11. La telemetría se debe trasmitir por protocolo MQTT.
   - 1.12. El sistema debe presentar una interfaz web para ajustar la configuración.
   - 1.13. Se debe poder visualizar los mensajes ASTERIX y de telemetría.
2. Requerimientos de documentación
   - 2.1. Se debe generar un informe de rendimiento del procesador de datos implementado.
   - 2.2. Se debe crear un manual de uso para el esclavo AXI.
   - 2.3. Se debe crear un manual de uso para el servicio procesador de datos.
3. Se debe proveer una interfaz web para supervisar y configurar el sistema.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

7. Historias de usuarios (*Product backlog*)

Las historias de usuario reciben una calificación numérica que refleja la complejidad de su implementación. A continuación se detallan los valores posibles:

- 1 punto: la historia demanda un esfuerzo trivial para satisfacerla.
- 2 puntos: la historia requiere de conocimientos profundos en una tecnología particular.
- 4 puntos: la historia necesita de conocimientos profundos en más de una tecnología.
- 8 puntos: la historia demanda múltiples tecnologías y debe cumplir con requerimientos de rendimiento.
- 16 puntos: la historia demanda múltiples tecnologías y debe funcionar en tiempo real.

A continuación se enumeran las historias de usuario:

- Historia 1
  - Descripción: como arquitecto de proyectos quiero una interfaz gráfica que me permita evaluar el rendimiento del modelo en el sistema embebido.
  - 8 *History points*: la historia de usuario necesita de conocimientos de Python, protocolos ZMQ y HTTP y además se necesita crear una página web dentro del sistema embebido. Finalmente, se debe medir el rendimiento del sistema.
- Historia 2
  - Descripción: como desarrollador de la *Radar Computer* necesito obtener telemetría MQTT para actualizar el estado del radar.
  - 4 *History points*: la historia requiere conocimientos de programación concurrente en Python y el manejo del protocolo MQTT.
- Historia 3
  - Descripción: como desarrollador de consolas de radar quiero obtener mensajes en formato ASTERIX para poder representar las novedades en campo.
  - 4 *History points*: se necesitan conocimientos de ZMQ, ASTERIX y Python.

8. Entregables principales del proyecto

Los entregables del proyecto son:

- Manual de usuario del generador de datos.
- Manual de usuario del servicio de procesamiento de datos.
- Código fuente del generador de datos.
- Código fuente del servicio de procesamiento de datos.
- Informe de rendimiento del servicio de procesamiento de datos.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

7. Historias de usuarios (*Product backlog*)

Las historias de usuario reciben una calificación numérica que refleja la complejidad de su implementación. A continuación se detallan los valores posibles:

- 1 punto: la historia demanda un esfuerzo trivial para satisfacerla.
- 2 puntos: la historia requiere de conocimientos profundos en una tecnología particular.
- 4 puntos: la historia necesita de conocimientos profundos en más de una tecnología.
- 8 puntos: la historia demanda múltiples tecnologías y debe cumplir con requerimientos de rendimiento.
- 16 puntos: la historia demanda múltiples tecnologías y debe funcionar en tiempo real.

A continuación se enumeran las historias de usuario:

- Historia 1
  - Descripción: como arquitecto de proyectos quiero una interfaz gráfica que me permita evaluar el rendimiento del modelo en el sistema embebido.
  - 8 *Story points*: la historia de usuario necesita de conocimientos de Python, protocolos ZMQ y HTTP y además se necesita crear una página web dentro del sistema embebido. Finalmente, se debe medir el rendimiento del sistema.
- Historia 2
  - Descripción: como desarrollador de la *Radar Computer* necesito obtener telemetría MQTT para actualizar el estado del radar.
  - 4 *Story points*: la historia requiere conocimientos de programación concurrente en Python y el manejo del protocolo MQTT.
- Historia 3
  - Descripción: como desarrollador de consolas de radar quiero obtener mensajes en formato ASTERIX para poder representar las novedades en campo.
  - 4 *Story points*: se necesitan conocimientos de ZMQ, ASTERIX y Python.

8. Entregables principales del proyecto

Los entregables del proyecto son:

- Manual de usuario del generador de datos.
- Manual de usuario del servicio de procesamiento de datos.
- Código fuente del generador de datos.
- Código fuente del servicio de procesamiento de datos.
- Informe de rendimiento del servicio de procesamiento de datos.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

9. Desglose del trabajo en tareas

1. Planificación y recursos
   - 1.1. Creación del documento de planificación (20 hs)
   - 1.2. Instalación del ambiente de desarrollo (15 hs)
   - 1.3. Creación del repositorio (5 hs)
   - 1.4. Gestión de horas (5 hs)
   - 1.5. Obtención de un *set* de datos (15 hs)
   - 1.6. Obtención de un modelo de procesamiento (15 hs)
2. Investigación
   - 2.1. Análisis de dependencias del modelo de procesamiento (15 hs)
   - 2.2. Análisis del funcionamiento del modelo de procesamiento (60 hs)
   - 2.3. Análisis del *set* de datos (50 hs)
3. Generador de datos
   - 3.1. Modelo RTL del encoder absoluto (15 hs)
   - 3.2. Simulación, síntesis y depuración del encoder absoluto (15 hs)
   - 3.3. Modelo RTL del generador de datos (39 hs)
   - 3.4. Simulación, síntesis y depuración del generador de datos (15 hs)
   - 3.5. Creación del IP Core (15 hs)
   - 3.6. Pruebas del IP Core (25 hs)
4. Backend radar
   - 4.1. Creación y prueba del proyecto Petalinux (5 hs)
   - 4.2. Compilación y prueba del intérprete de Python (5 hs)
   - 4.3. Compilación y prueba de la biblioteca ZMQ (10 hs)
   - 4.4. Compilación y prueba de la biblioteca MQTT (5 hs)
   - 4.5. Compilación y prueba de la biblioteca FLASK (10 hs)
   - 4.6. Desarrollo, compilación y prueba del árbol de dispositivos (15 hs)
   - 4.7. Desarrollo, compilación y prueba del driver del generador de datos (40 hs)
   - 4.8. Desarrollo, compilación y prueba de la biblioteca ASTERIX (35 hs)
   - 4.9. Integración del Backend radar (40 hs)
   - 4.10. Pruebas de funcionamiento (20 hs)
5. Cierre
   - 5.1. Creación del manual del generador de datos (5 hs)
   - 5.2. Creación del manual del Backend radar (5 hs)
   - 5.3. Creación del informe de rendimiento del sistema (30 hs)
   - 5.4. Creación de la memoria del Trabajo Final (10 hs)
   - 5.5. Creación de un vídeo de demostración (10 hs)
   - 5.6. Defensa pública y agradecimientos (1 hs)

Cantidad total de horas: 600 hs.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

9. Desglose del trabajo en tareas

1. Planificación y recursos
   - 1.1. Creación del documento de planificación (20 hs)
   - 1.2. Instalación del ambiente de desarrollo (15 hs)
   - 1.3. Creación del repositorio (5 hs)
   - 1.4. Gestión de horas (5 hs)
   - 1.5. Obtención de un *set* de datos (15 hs)
   - 1.6. Obtención de un modelo de procesamiento (15 hs)
2. Investigación
   - 2.1. Análisis de dependencias del modelo de procesamiento (15 hs)
   - 2.2. Análisis del funcionamiento del modelo de procesamiento (60 hs)
   - 2.3. Análisis del *set* de datos (50 hs)
3. Generador de datos
   - 3.1. Modelo RTL del encoder absoluto (15 hs)
   - 3.2. Simulación, síntesis y depuración del encoder absoluto (15 hs)
   - 3.3. Modelo RTL del generador de datos (39 hs)
   - 3.4. Simulación, síntesis y depuración del generador de datos (15 hs)
   - 3.5. Creación del IP Core (15 hs)
   - 3.6. Pruebas del IP Core (25 hs)
4. Backend radar
   - 4.1. Creación y prueba del proyecto Petalinux (5 hs)
   - 4.2. Compilación y prueba del intérprete de Python (5 hs)
   - 4.3. Compilación y prueba de la biblioteca ZMQ (10 hs)
   - 4.4. Compilación y prueba de la biblioteca MQTT (5 hs)
   - 4.5. Compilación y prueba de la biblioteca FLASK (10 hs)
   - 4.6. Desarrollo, compilación y prueba del árbol de dispositivos (15 hs)
   - 4.7. Desarrollo, compilación y prueba del driver del generador de datos (40 hs)
   - 4.8. Desarrollo, compilación y prueba de la biblioteca ASTERIX (35 hs)
   - 4.9. Integración del Backend radar (40 hs)
   - 4.10. Pruebas de funcionamiento (20 hs)
5. Cierre
   - 5.1. Creación del manual del generador de datos (5 hs)
   - 5.2. Creación del manual del Backend radar (5 hs)
   - 5.3. Creación del informe de rendimiento del sistema (30 hs)
   - 5.4. Creación de la memoria del Trabajo Final (10 hs)
   - 5.5. Creación de un vídeo de demostración (10 hs)
   - 5.6. Defensa pública y agradecimientos (1 hs)

Cantidad total de horas: 600 hs.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

10. Diagrama de Activity On Node

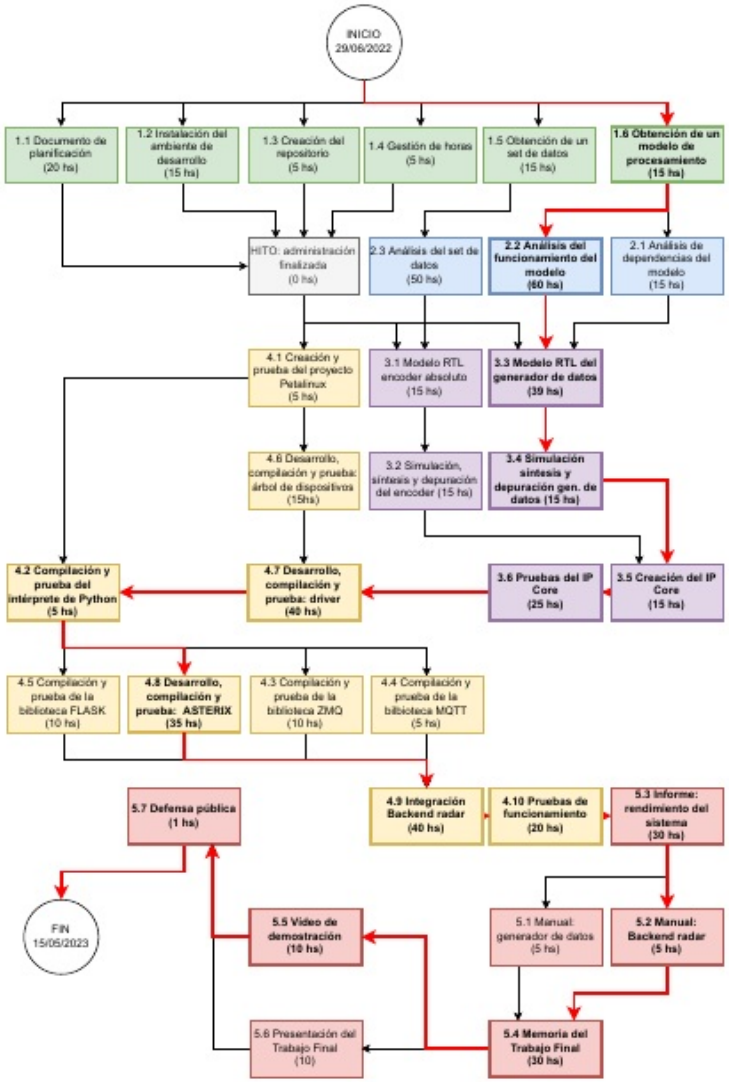

Figura 2. Diagrama en *Activity on Node*

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

10. Diagrama de Activity On Node

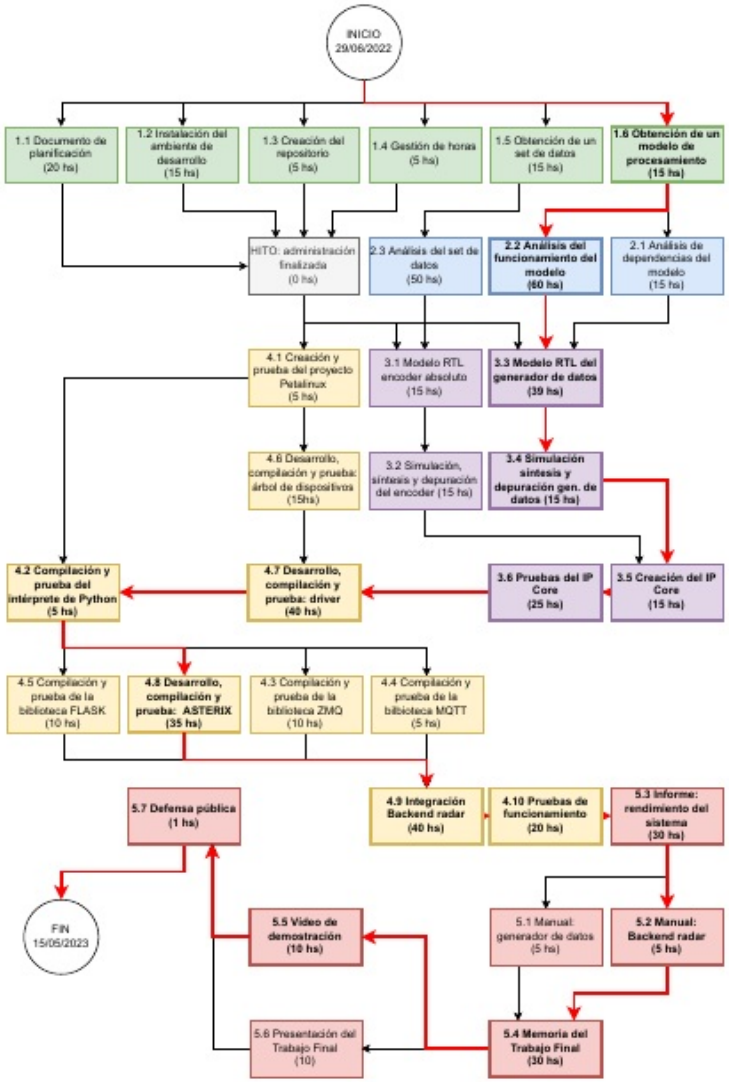

Figura 2. Diagrama en *Activity on Node*

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.

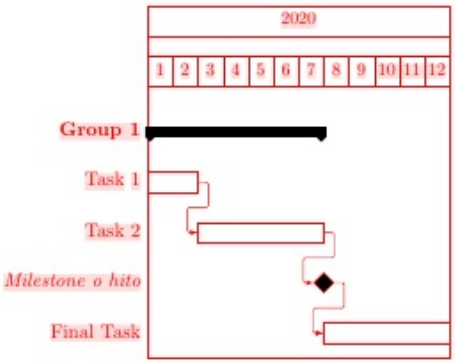

Figura 3. Diagrama de gantt de ejemplo

11. Diagrama de Gantt

A partir del diagrama en *Activity On Node* de la sección 10 se creó un diagrama de Gantt. Se lo puede observar en las figuras 3, 4, 5, 6, 7, 8 y 9.

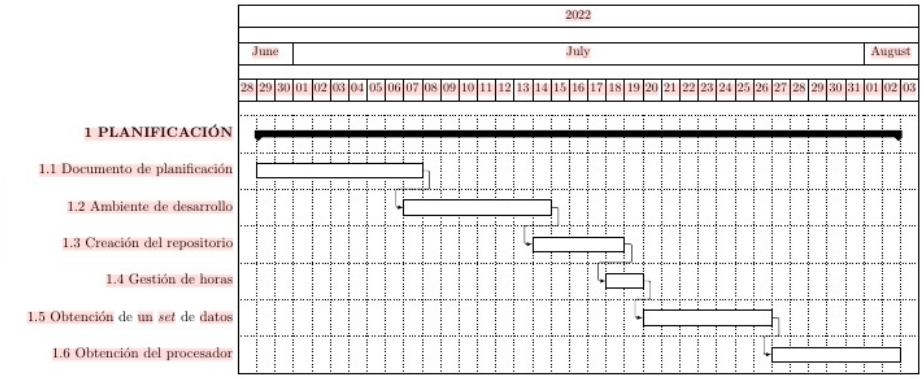

Figura 3. Diagrama de Gantt: parte 1.

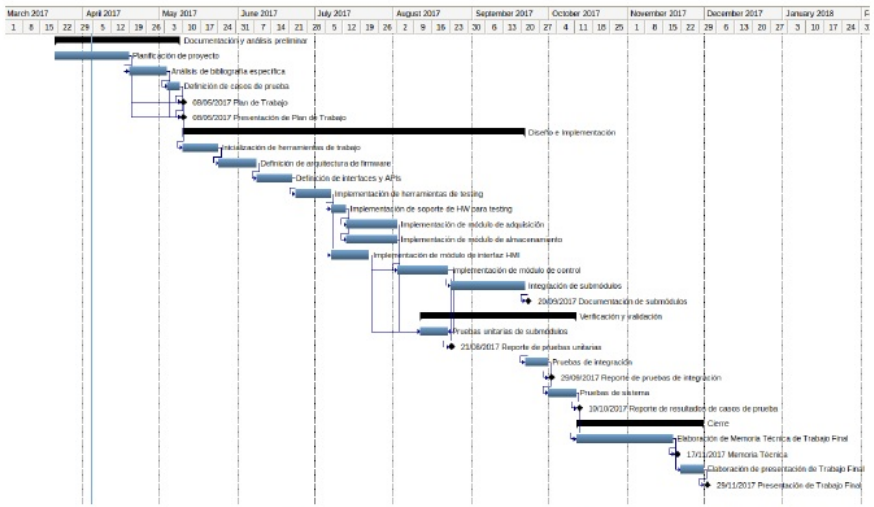

Figura 4. Ejemplo de diagrama de Gantt rotado

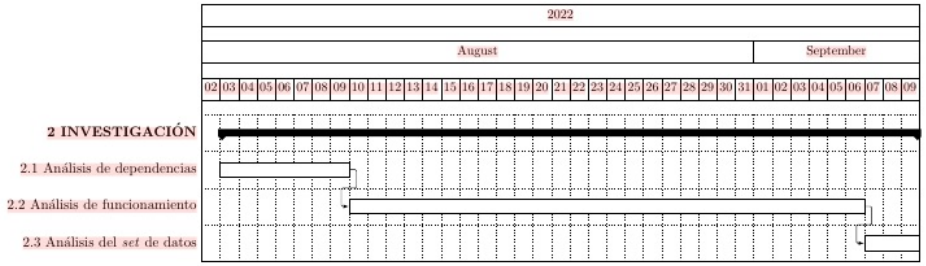

Figura 4. Diagrama de gantt: parte 2.

12. Presupuesto detallado del proyecto

A continuación se detallan los costos previstos para del proyecto:

COSTOS DIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
Arty Z7	1	USD 200	USD 200
SUBTOTAL			USD 200
COSTOS INDIRECTOS			
Descripción	Cantidad	Valor unitario	Valor total
Costos de aduana	1	USD 100	USD 100
SUBTOTAL			USD 100
TOTAL			USD 300

13. Gestión de riesgos

a) A continuación se identifican los riesgos previstos en el proyecto:

Riesgo 1: demoras en la adquisición del hardware.

- Severidad (4): no es posible realizar pruebas de rendimiento sin tener el hardware real, sin embargo es posible realizar el desarrollo del proyecto.
- Probabilidad de ocurrencia (7): es frecuente tener problemas para realizar importaciones en la República Argentina.

Riesgo 2: sin acceso al algoritmo desarrollado por modelística.

- Severidad (10): sin el modelo no es posible realizar el proyecto.
- Ocurrencia (1): el proyecto tiene el apoyo de la jefa del departamento de ingeniería de software.

Riesgo 3: sin acceso a una colección de datos de radar.

- Severidad (7): sin datos reales de radar se pierde la posibilidad de realizar pruebas que arrojen resultados útiles.
- Ocurrencia (1): el proyecto tiene el apoyo de la jefa del departamento de ingeniería de software.

Riesgo 4: sin horas para realizar el proyecto durante el horario laboral.

- Severidad (5): el proyecto contempla el uso de horas dentro del horario laboral para cumplir con la fecha de entrega.
- Ocurrencia (5): las obligaciones contractuales de la empresa pueden forzar la eliminación de las horas otorgadas para el proyecto.

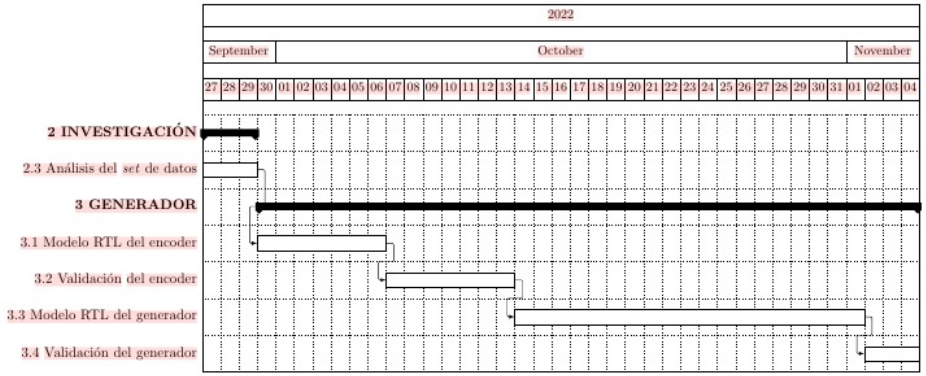

Figura 5. Diagrama de gantt: parte 3.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

Riesgo 5: imposibilidad económica para continuar el posgrado.

- Severidad (10): Significaría el fin del proyecto.
- Ocurrencia (1): Se tienen los recursos necesarios económicos para cursar los cinco bimestres.

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

Riesgo	S	O	RPN	S*	O*	RPN*
Demoras en la adquisición del hardware	4	7	28	1	7	7
Sin acceso al algoritmo desarrollado por modelística	10	1	10			
Sin acceso a una colección de datos de radar	7	1	7			
Sin horas para realizar el proyecto durante el horario laboral	5	5	25	2	5	10
Imposibilidad económica para continuar el posgrado	10	1	10			

Criterio adoptado: Se tomarán medidas de mitigación en los riesgos cuyos números de RPN sean mayores a 20.

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: se pone a disposición del proyecto hardware que es propiedad del responsable del proyecto.
- Severidad (1): al disponer del hardware del responsable del proyecto se elimina los efectos negativos asociados al riesgo.
- Probabilidad de ocurrencia (7): la mitigación adoptada no altera la probabilidad de ocurrencia.

Riesgo 4: el responsable del proyecto planifica su agenda personal para tener a disposición las horas necesarias.
- Severidad (2): al disponer de horas libres en la agenda del responsable del proyecto se atenúa el efecto negativo del riesgo.
- Probabilidad de ocurrencia (7): la mitigación adoptada no altera la probabilidad de ocurrencia.

14. Gestión de la calidad

- Requerimientos funcionales
  - REQ-01: el sistema se debe implementar en un SoC de Xilinx.
    - Verificación: se correrá el sistema en QEMU para luego ser probado en la placa Arty Z7.
    - Validación: se le mostrará al cliente la ejecución del sistema en la placa Arty Z7.
  - REQ-02: el sistema debe generar un flujo de datos de radar.
    - Verificación: se realizará un *script* que capture el flujo de datos y los compare con los valores esperados.
    - Validación: se realizará un *script* que envíe los datos por protocolo UDP para que el cliente pueda realizar sus propias pruebas.

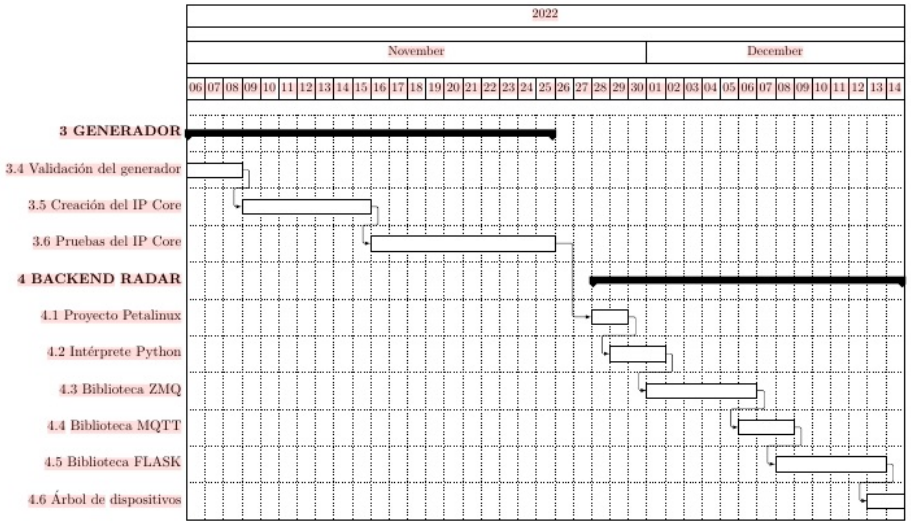

Figura 6. Diagrama de gantt: parte 4.

DiffPDF • /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/TI_Vaca_Gonzalo_V1.pdf vs. /home/gonzalo/workspace/MSE/MSE-Plan/Entragas/

- REQ-03: el generador de datos debe ser un esclavo AXI en la lógica programable.
  - Verificación: se correrán las pruebas automáticas del entorno de desarrollo integrado Vivado.
  - Validación: Se generará un *IP Core* para que el cliente realice sus propias pruebas.
- REQ-04: la generación de datos debe ser configurable.
  - Verificación: se realizará una demostración en donde se cambiará la velocidad de barrido en azimut y se designarán zonas inhibidas.
  - Validación: el cliente entregará una serie de configuraciones durante la demostración y se persistirán los resultados para que el cliente decida si el generador funciona de forma correcta.
- REQ-05: el sistema debe procesar los datos y generar mensajes ASTERIX.
  - Verificación: se usará un programa del tipo *parser* que determina si el protocolo se usó de forma correcta.
  - Validación: se entregará una captura de mensajes ASTERIX al cliente.
- REQ-06: el procesador de datos se debe implementar en Python dentro de la Unidad de procesamiento de aplicaciones.
  - Verificación: se relevará la proporción de lenguajes en la aplicación.
  - Validación: se realizará un control de versiones en un repositorio dentro de la infraestructura de INVAP S.E.
- REQ-07: el procesador debe correr como servicio dentro del sistema operativo.
  - Verificación: se realizará un *boot* del *kernel* y se mostrará el mensaje del sistema operativo que muestra el inicio del servicio.
  - Validación: con el sistema operativo funcionando se pedirá la lista de procesos y debe figurar el servicio.
- REQ-08: se debe crear una distribución de Linux embebido.
  - Verificación: se realizará una demostración de *boot* de la distribución.
  - Validación: el proyecto Petalinux estará en un repositorio dentro de la infraestructura de INVAP S.E.
- REQ-09: los mensajes ASTERIX se deben transmitir por protocolo ZMQ.
  - Verificación: se usará un cliente ZMQ para recibir los datos.
  - Validación: se realizará una demostración de la captura de datos en ZMQ.
- REQ-10: el sistema debe generar datos de telemetría que indiquen su estado de funcionamiento.
  - Verificación: se inducirán fallas en el sistema y se deberán reflejar en la telemetría.
  - Validación: se repetirá el ensayo de verificación pero el cliente será quién decida que componentes se pondrán en falla.
- REQ-11: la telemetría se debe trasmitir por protocolo MQTT.
  - Verificación: se usará un cliente MQTT para capturar la telemetría.
  - Validación: se repetirá la prueba de verificación en presencia del cliente.
- REQ-12: el sistema debe presentar una interfaz web para ajustar la configuración.
  - Verificación: se realizarán cambios desde la interfaz web y se verá el impacto en el archivo de configuración.
  - Validación: se realizarán cambios desde la interfaz web y se verá el cambio de comportamiento en el sistema.
- REQ-13: se debe poder visualizar los mensajes ASTERIX y de telemetría.

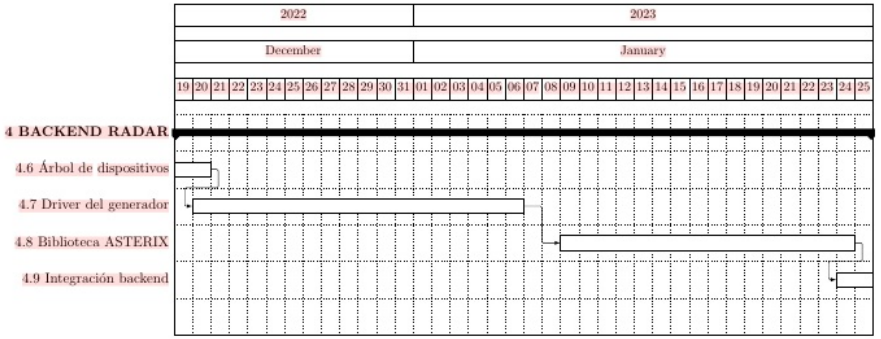

Figura 7. Diagrama de gantt: parte 5.

.UBAfiuba FACULTAD DE INGENIERÍA
Plan de proyecto de Trabajo final
Maestría en Sistemas Embebidos
Gonzalo Nahuel Vaca

- Verificación: se generarán mensajes de pruebas en ASTERIX y MQTT y se medirá su impacto en la interfaz gráfica.
- Validación: se pondrá el sistema en funcionamiento y se medirá su impacto en la interfaz gráfica.

- Requerimientos de documentación
  - REQ-14: se debe generar un informe de rendimiento del procesador de datos implementado.
    - Verificación: se cambiará la velocidad de giro del generador de datos y se estudiará cuantos paquetes se pierde en cada caso.
    - Validación: se conectará el sistema a una consola de radar y se probarán distintas configuraciones del generador de datos.
  - REQ-15: se debe crear un manual de uso para el esclavo AXI.
    - Verificación: se creará un proyecto nuevo y se seguirán los pasos especificados en el manual, el resultado debe ser una aplicación de ejemplo que funcione.
    - Validación: el cliente realizará una revisión del documento.
  - REQ-16: se debe crear un manual de uso para el servicio procesador de datos.
    - Verificación: se creará un proyecto nuevo y se seguirán los pasos especificados en el manual, el resultado debe ser una aplicación de ejemplo que funcione.
    - Validación: el cliente realizará una revisión del documento.
- REQ-17: se debe proveer una interfaz web para supervisar y configurar el sistema.
  - Verificación: se realizará un *log* de los pedidos provenientes de la interfaz web para analizar su correcto funcionamiento.
  - Validación: el cliente probará la interfaz web y dará sus correcciones.

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:
  - El responsable del proyecto analizará el plan original y se verificará el grado de correspondencia con su ejecución.
  - El responsable del proyecto evaluará las tareas que no se ajustaron al plan para detectar las causas y tenerlas en cuentas en próximos proyectos.
- Identificación de las técnicas y procedimientos útiles e inútiles que se emplearon, y los problemas que surgieron y cómo se solucionaron:
  - El responsable del proyecto evaluará el rendimiento del lenguaje de programación Python en un backend radar.
  - El responsable del proyecto generará informes con todos los problemas técnicos no previstos.
- Luego de la presentación del trabajo ante el jurado, el responsable del proyecto realizará un agradecimiento público a todas las personas involucradas en el proyecto.

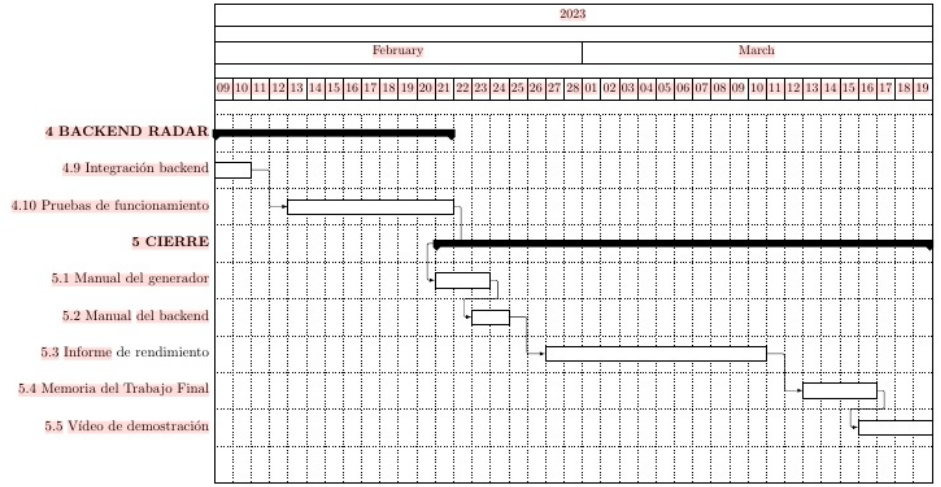

Figura 8. Diagrama de gantt: parte 6.