



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Aprendizaje por refuerzo en
redes ópticas pasivas (PON)
Documentación Técnica**



Presentado por David Pérez Moreno
en Universidad de Burgos — 8 de julio de 2024

Tutor: Rubén Ruiz González

Co-tutora: Noemí Merayo Álvarez

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	20
Apéndice B Especificación de Requisitos	25
B.1. Introducción	25
B.2. Objetivos generales	25
B.3. Catálogo de requisitos	26
Apéndice C Especificación de diseño	35
C.1. Diseño de Datos	35
C.2. Diseño Procedimental	35
C.3. Diseño Arquitectónico	36
Apéndice D Documentación técnica de programación	39
D.1. Introducción	39
D.2. Estructura de directorios	39
D.3. Manual del programador	41
D.4. Compilación, instalación y ejecución del proyecto	42
D.5. Pruebas del sistema	44

Apéndice E Documentación de usuario	47
E.1. Introducción	47
E.2. Requisitos de usuarios	47
E.3. Instalación	48
E.4. Manual del usuario	50
Apéndice F Anexo de sostenibilización curricular	53
F.1. Introducción	53
F.2. Reflexión sobre la Sostenibilidad	53
F.3. Conclusión	55
Bibliografía	57

Índice de figuras

A.1. Roadmap del proyecto	3
A.2. 1º Issue	5
A.3. 2º Issue	6
A.4. 3º Issue	7
A.5. 4º Issue	8
A.6. 5º Issue	9
A.7. 6º Issue	10
A.8. 7º Issue	11
A.9. 8º Issue	12
A.10.9º Issue	13
A.11.10º Issue	14
A.12.11º Issue	15
A.13.12º Issue	16
A.14.13º Issue	17
A.15.14º Issue	18
A.16.15º Issue	19
B.1. Diagrama de casos de uso	28
C.1. Diagrama de secuencia	36
C.2. Diagrama de clases UML	37
D.1. Descarga del repositorio	41
D.2. Proyecto abierto en Visual Studio Code	42
D.3. Extensión de Jupyter Notebook en Visual Studio Code	43
D.4. Script.ipynb 1º Escenario	44

Índice de tablas

A.1. Listado con los materiales, su valor y su tiempo de amortización.	20
A.2. Listado con los materiales y coste amortizado.	21
A.3. Desglose del coste de personal.	21
A.4. Librerías utilizadas, junto con sus licencias.	23
B.1. CU-1 Seleccionar Carpeta de Algoritmo.	29
B.2. CU-2 Ejecutar Script del Escenario.	30
B.3. CU-3 Seleccionar Número de Ciclos.	31
B.4. CU-4 Visualizar Gráficas.	32
B.5. CU-5 Modificar Valores.	32
B.6. CU-6 Resetear Entorno.	33
D.1. Librerías utilizadas y sus versiones	43

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este apartado del Trabajo de Fin de Grado se dedica a **examinar detalladamente la viabilidad y la estrategia de implementación de nuestro proyecto de desarrollo de software**. Con el objetivo de asegurar que el proyecto no solo sea técnica y operacionalmente posible, sino también económicamente justificable y legalmente viable, se llevarán a cabo diversas evaluaciones críticas en varias etapas.

- **Planificación Temporal:** Iniciaremos con la planificación temporal, delineando un cronograma detallado que cubre todas las etapas del proyecto, desde su concepción hasta la finalización y despliegue. Esta programación no solo define las fases y hitos clave, sino que también asigna recursos, anticipa posibles obstáculos y establece plazos realistas, garantizando una ejecución fluida y a tiempo.
- **Estudio de Viabilidad:** Seguidamente, abordaremos el estudio de viabilidad, dividido en viabilidad técnica y operativa. Aquí, evaluaremos si tenemos las tecnologías, capacidades y condiciones operativas necesarias para llevar a cabo el proyecto con éxito, considerando la infraestructura actual y el contexto tecnológico.
- **Viabilidad Económica:** La viabilidad económica también ocupa un lugar central en nuestro análisis. Presentaremos un desglose completo de los costos asociados y confrontaremos estos con los beneficios proyectados. A través de un análisis costo-beneficio, determinaremos

la rentabilidad del proyecto y el tiempo estimado para recuperar la inversión inicial.

- **Viabilidad Legal:** Finalmente, exploraremos la viabilidad legal, asegurando que el proyecto cumpla con todas las normativas y leyes aplicables. Desde las licencias de software hasta las regulaciones de protección de datos, cada aspecto será revisado para prevenir implicaciones legales que pudieran surgir.

Este análisis comprensivo no solo respalda la viabilidad del proyecto desde múltiples perspectivas, sino que también establece un **marco de referencia claro para su gestión y ejecución**, permitiendo tomar decisiones informadas y fundamentadas.

A.2. Planificación temporal

En la gestión de la planificación temporal de mi proyecto de desarrollo de software, he empleado la funcionalidad de proyectos de **GitHub**, una herramienta integral que facilita la organización y el seguimiento de las tareas en un entorno colaborativo. Esta sección describe cómo GitHub ha sido utilizado personalmente para optimizar la gestión de mi cronograma de proyecto.

- **Organización de Tareas:** GitHub permite la creación de tarjetas y columnas dentro de un tablero de proyecto, siguiendo un enfoque similar al Kanban. Cada tarjeta representa una tarea específica del proyecto y está clasificada en columnas que representan diferentes estados de progreso (por ejemplo, Pendiente, En Progreso, Completado). Esta organización me ha permitido monitorizar fácilmente las actividades pendientes y las ya realizadas, asegurando una gestión eficiente del tiempo y los recursos.
- **Seguimiento de Avances:** La herramienta de proyectos de GitHub ofrece también la posibilidad de vincular ‘issues’ y ‘pull requests’ a cada tarjeta. Esto me ha permitido mantener un control detallado del progreso, donde cada cambio en el código o cada mejora realizada puede ser directamente asociada con una tarea específica del tablero. Además, los ‘milestones’ asociados a los ‘issues’ han sido fundamentales para definir y seguir los hitos importantes del proyecto.

- **Revisión y Ajustes:** La adaptabilidad es crucial en la gestión de proyectos de software. GitHub me ha permitido hacer ajustes en tiempo real al plan de proyecto, reasignando tareas y redefiniendo plazos según las necesidades emergentes y los obstáculos encontrados. Este enfoque dinámico asegura que el proyecto se mantenga en curso y se ajuste a las condiciones cambiantes.

Respecto a la gestión de las tareas y de la organización de las tarjetas y columnas podemos ver en la siguiente imagen cómo las ‘issues’ han sido representadas en el tablero siguiente desde la parte de proyectos de GitHub projects.

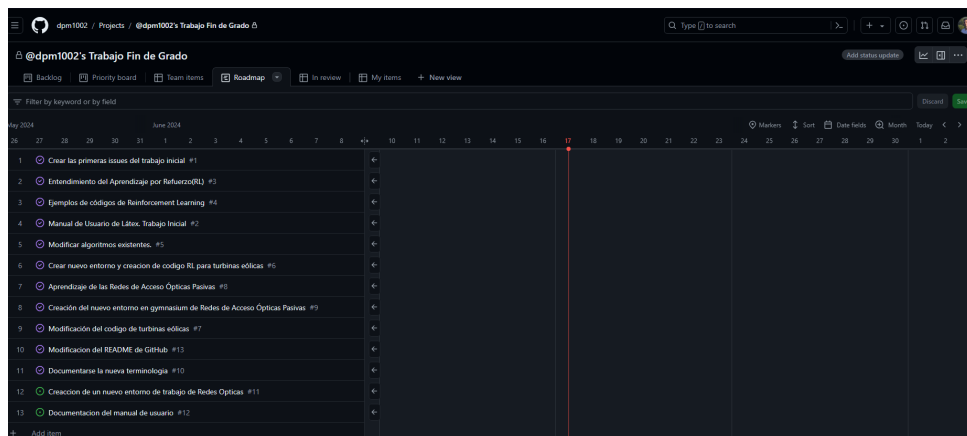


Figura A.1: Roadmap del proyecto

Entre estas actividades se pueden diferenciar 5 estados:

- **Backlog:** El backlog es esencialmente la lista de todas las tareas que se han identificado para el proyecto pero que aún no han comenzado. Esta columna actúa como un depósito de todas las funcionalidades, requisitos, correcciones y mejoras pendientes que necesitan ser abordadas.
- **Ready:** La columna ‘Ready’ contiene todas las tareas que están preparadas para comenzar pero no se han iniciado aún. Estas tareas han sido planificadas, diseñadas y están completamente definidas, con suficientes detalles proporcionados para que un desarrollador pueda empezar a trabajar en ellas sin impedimentos.

- **In Progress:** Las tareas en la columna ‘In Progress’ son aquellas que actualmente están siendo trabajadas por los desarrolladores. Indica que el trabajo ha comenzado y está activamente en curso.
- **In Review:** Una vez que las tareas se completan, se mueven a la columna ‘In Review’. Esto significa que el código o la tarea realizada necesita ser revisada por pares o superiores antes de considerarse completa. La revisión puede incluir pruebas, verificaciones de calidad, y revisiones de código.
- **Done:** Las tareas en la columna ‘Done’ son aquellas que han sido completadas satisfactoriamente y han pasado todas las revisiones y pruebas necesarias.

Estas son cada una de las actividades, pero vamos a ver en concreto en cada tarea el cómo afecta y cómo de desarrollado y detallado está cada una de ellas.

1º Sprint

En este primer sprint nos centraremos en las tareas iniciales, en la familiarización al entorno y la organización de este.

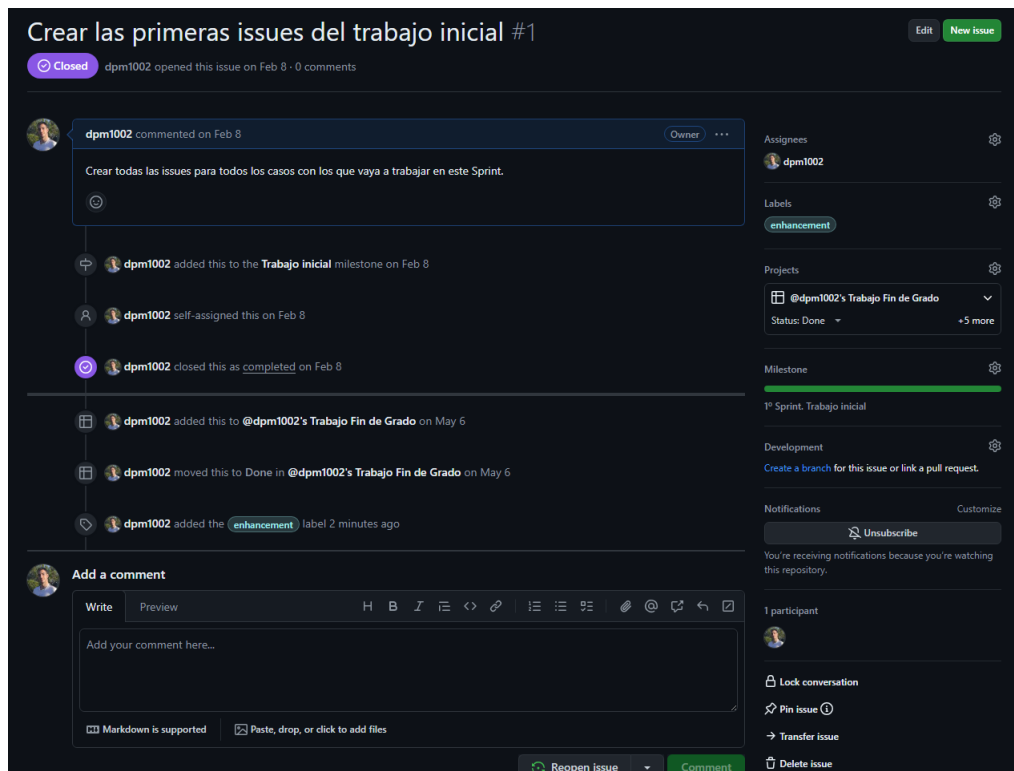


Figura A.2: 1º Issue

- Título: Creación de las Issues del trabajo inicial.
- Descripción: Esta tarea está asociada a la creación de las issues iniciales.
- Asignaciones: David Pérez Moreno
- Labels(enhancement): Esta label se corresponde a la ampliación de las funcionalidades del programa.
- Status(Done): La actividad está finalizada.

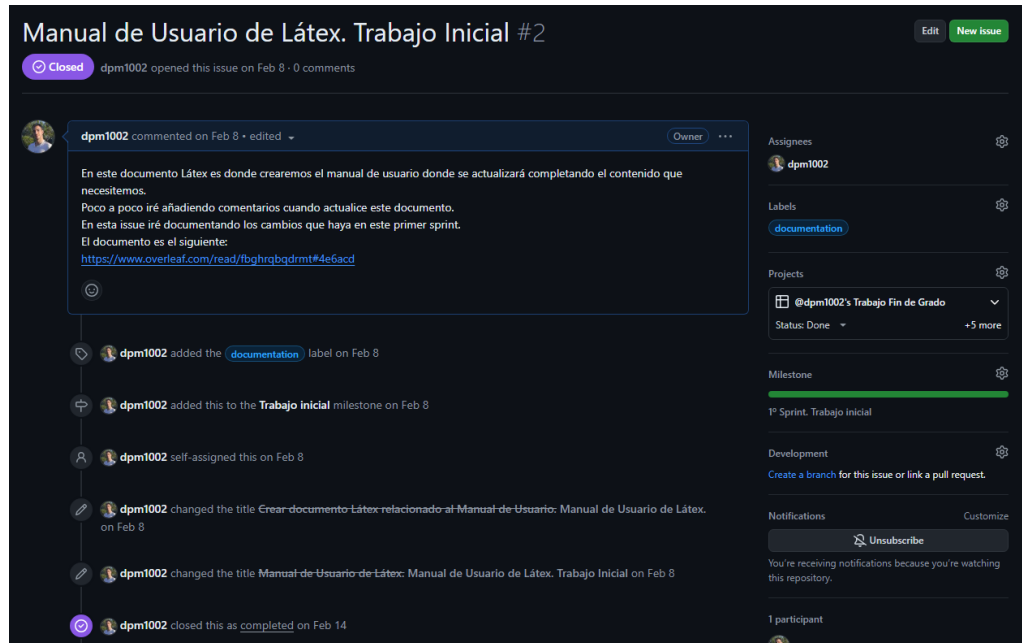


Figura A.3: 2º Issue

- Título: Manual de Usuario de L^AT_EX. Trabajo Inicial.
- Descripción: Esta tarea está asociada a la creación del manual de usuario y anexos en L^AT_EX.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación de los aspectos del código.
- Status(Done): La actividad está finalizada.

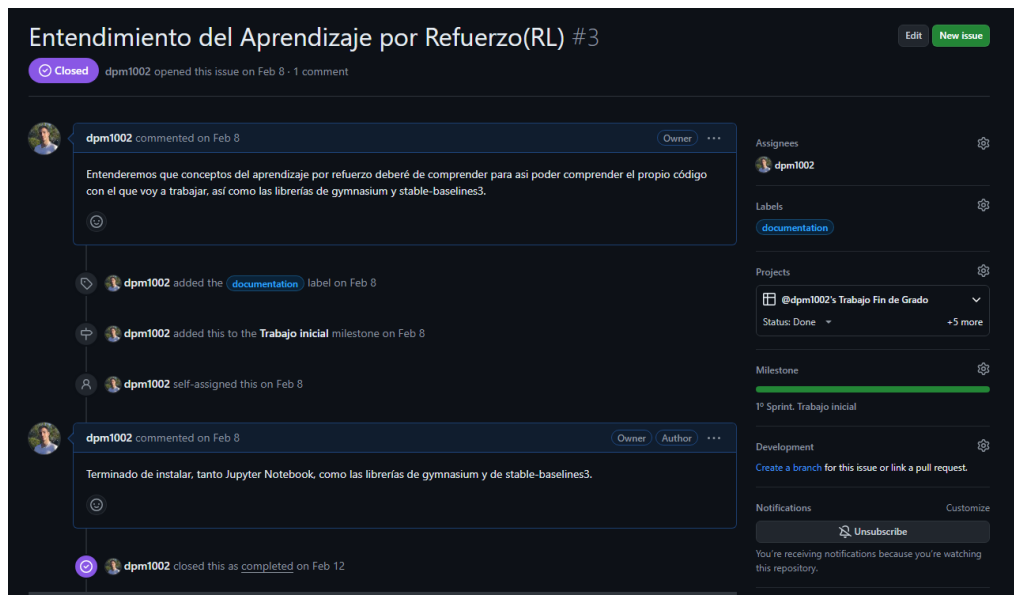


Figura A.4: 3º Issue

- Titulo: Entendimiento del Aprendizaje por Refuerzo(RL).
- Descripción: Esta tarea está asociada al entendimiento de los entornos RL y a las librerías de Gymnasium y StableBaselines_3.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación de la funcionalidad del código y de las librerías que las compone.
- Status(Done): La actividad está finalizada.

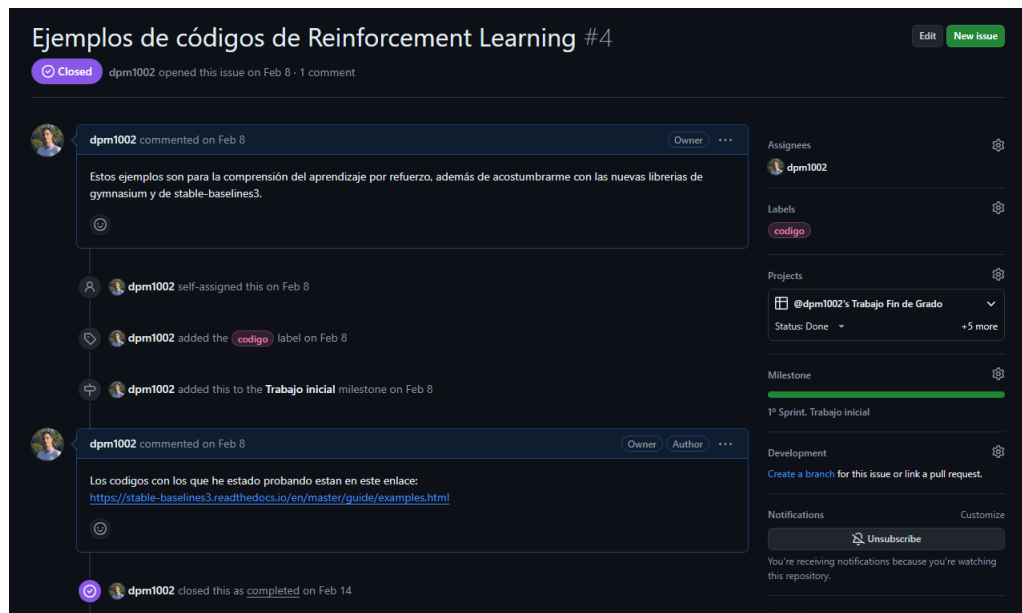


Figura A.5: 4º Issue

- Título: Ejemplos de códigos de Reinforcement Learning.
- Descripción: Esta tarea está asociada a la ejecución de algoritmos de Reinforcement Learning.
- Asignaciones: David Pérez Moreno
- Labels(código): Esta label se corresponde a las pruebas con códigos que implementan las librerías de Gymnasium y StableBaselines_3.
- Status(Done): La actividad está finalizada.

2º Sprint

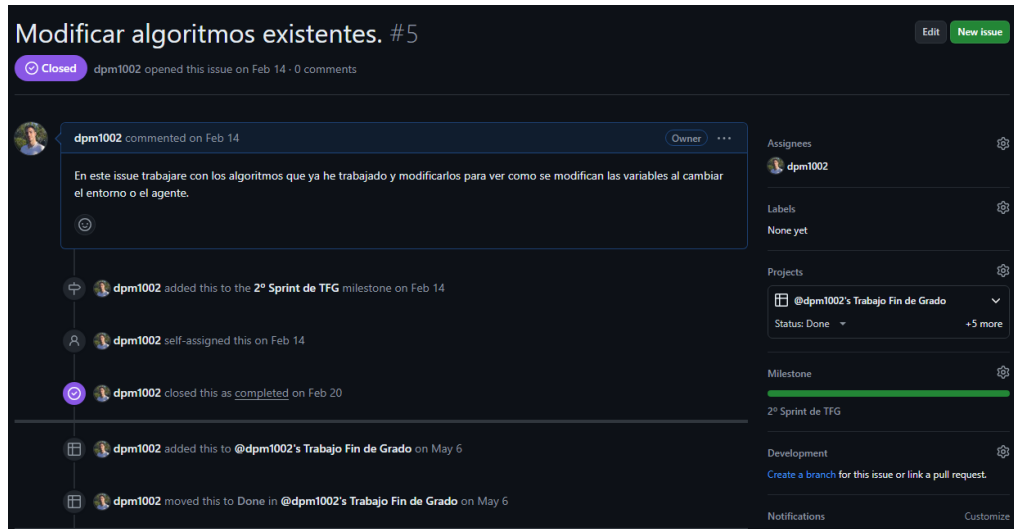


Figura A.6: 5º Issue

- Título: Modificar algoritmos existentes.
- Descripción: Esta tarea está asociada a la modificación de los algoritmos de RL para ver su funcionamiento al cambiar el valor de las variables.
- Asignaciones: David Pérez Moreno
- Labels(código): Esta label se corresponde a las pruebas con códigos de Python.
- Status(Done): La actividad está finalizada.

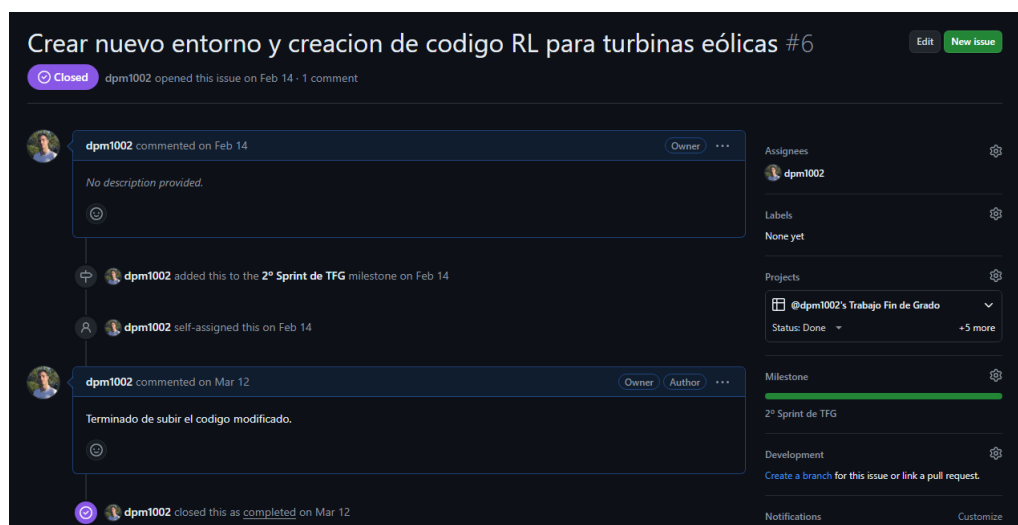


Figura A.7: 6º Issue

- **Título:** Crear nuevo entorno y creación de código DRL para turbinas eólicas.
- **Descripción:** Esta tarea está asociada a la creación del código en Python DRL en un entorno de turbinas eólicas.
- **Asignaciones:** David Pérez Moreno
- **Labels(código):** Esta label se corresponde a la creación de un código Python y Jupyter Notebook que englobe el funcionamiento de este código.
- **Status(Done):** La actividad está finalizada.

3º Sprint

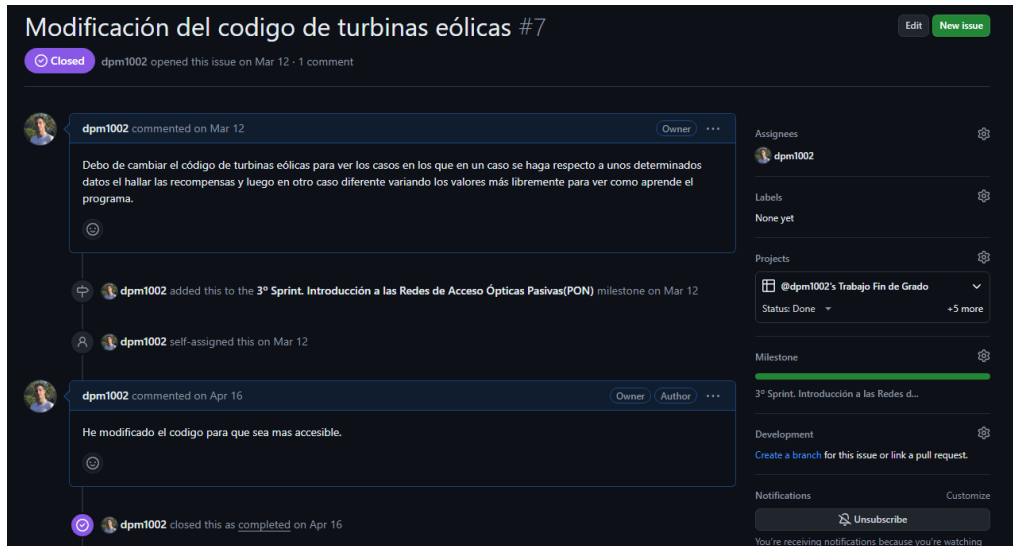


Figura A.8: 7º Issue

- Titulo: Modificación del código de turbinas eólicas.
- Descripción: Esta tarea está asociada a la modificación del código en Python DRL del entorno de turbinas eólicas.
- Asignaciones: David Pérez Moreno
- Labels(código): Esta label se corresponde a la modificación del entorno Python asociado a las turbinas eólicas.
- Status(Done): La actividad está finalizada.

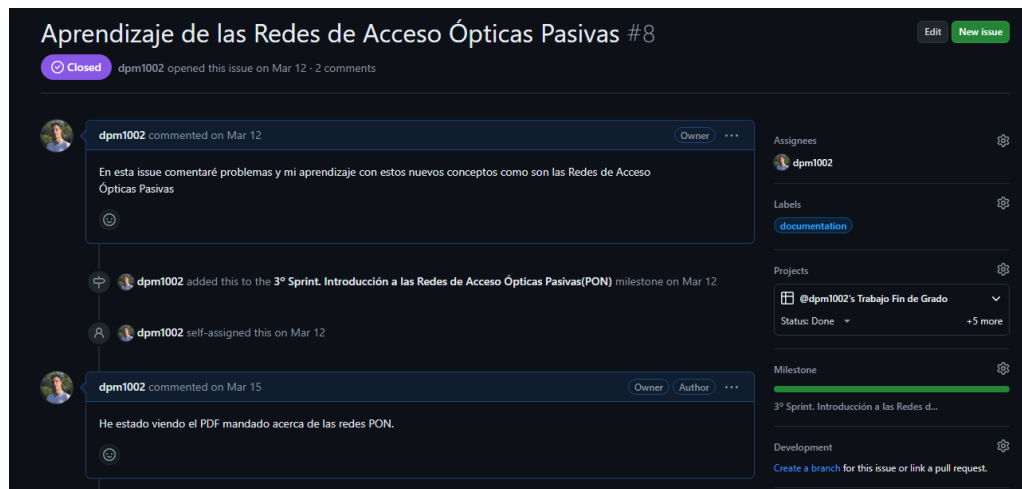


Figura A.9: 8º Issue

- Titulo: Aprendizaje de las Redes de Acceso Ópticas Pasivas.
- Descripción: Esta tarea está asociada al aprendizaje de la nueva terminología en redes ópticas, más en concreto de las redes GPON.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación y aprendizaje sobre esta nueva terminología.
- Status(Done): La actividad está finalizada.

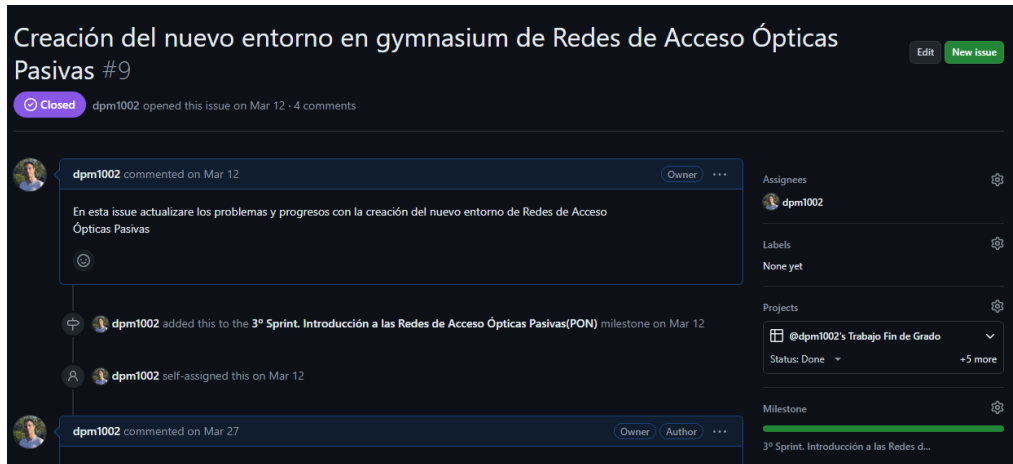


Figura A.10: 9º Issue

- Título: Creación del nuevo entorno en gymnasium de Redes de Acceso Ópticas Pasivas.
- Descripción: Esta tarea está asociada a la creación y modificación del código Python DRL en el entorno de Redes de Acceso Ópticas Pasivas.
- Asignaciones: David Pérez Moreno
- Labels(código): Esta label se corresponde a la creación y modificación del código Python DRL asociado al entorno de Redes de Acceso Ópticas Pasivas.
- Status(Done): La actividad está finalizada.

4º Sprint

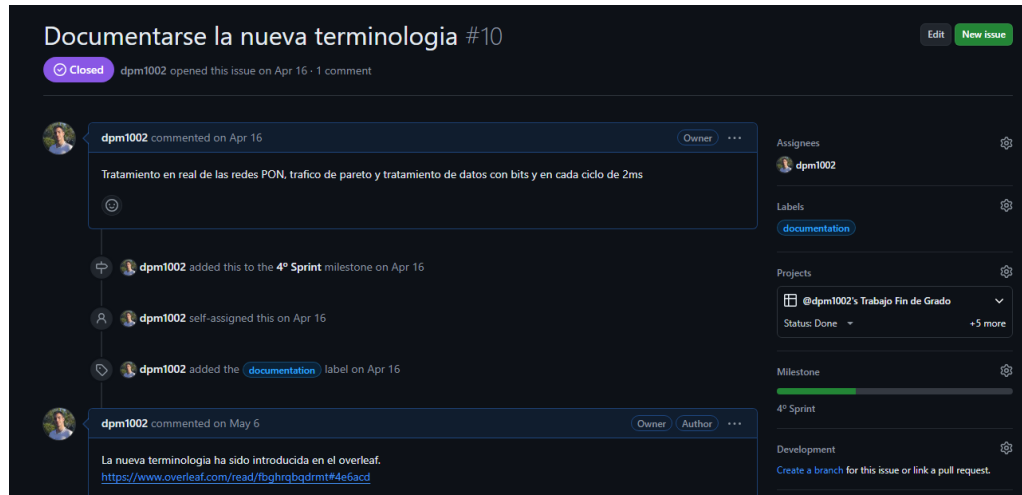


Figura A.11: 10º Issue

- Título: Documentarse la nueva terminología.
- Descripción: Esta tarea está asociada a la documentación de las características de la distribución de Pareto y su implicación en la terminología del Tráfico de Pareto. Posteriormente se aplicará esta característica al entorno modificado.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación nueva de la terminología de la distribución de Pareto.
- Status(Done): La actividad está finalizada.

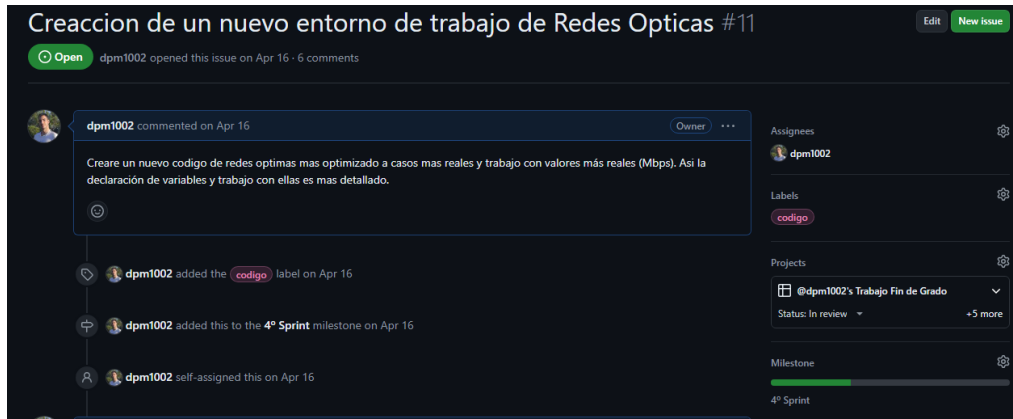


Figura A.12: 11º Issue

- Titulo: Creación de un nuevo entorno de trabajo de Redes Ópticas.
- Descripción: Esta tarea está asociada a la creación y modificación del nuevo código Python DRL en los entornos de Redes Ópticas incluyendo la distribución de Pareto para la búsqueda de un entorno más fiable y real ante los valores que se usan en el simulador XGSPON de la Universidad de Valladolid.
- Asignaciones: David Pérez Moreno
- Labels(código): Esta label se corresponde a la creación y modificación de los avances en el nuevo código del entorno de Redes Ópticas.
- Status(In review): La actividad está finalizada pero pendiente de revisión ante unos futuros cambios.

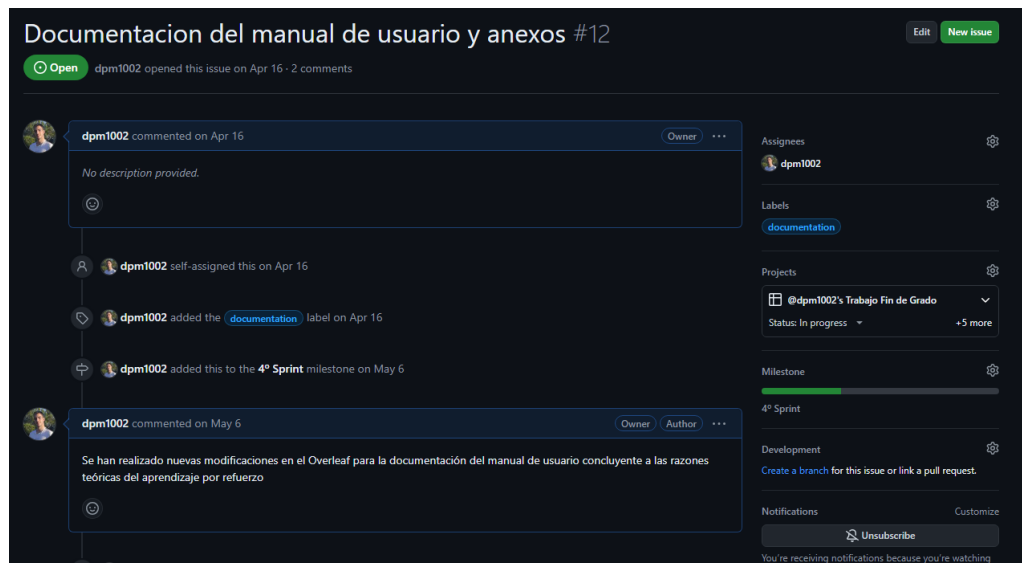


Figura A.13: 12º Issue

- Titulo: Documentación del manual de usuario y anexos.
- Descripción: Esta tarea está asociada a la documentación y desarrollo del manual de usuario y los anexos en $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación del manual de usuario y los anexos.
- Status(In progress): La actividad está siendo realizada.



Figura A.14: 13º Issue

- Título: Modificación del README de GitHub.
- Descripción: Esta tarea está asociada a la modificación del archivo README de GitHub explicando el código realizado.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Esta label se corresponde a la documentación del README de GitHub.
- Status(Done): La actividad está finalizada.



Figura A.15: 14º Issue

- Titulo: Organización del repositorio y refactorización.
- Descripción: Esta tarea está asociada a la modificación de la estructura del código y de su refactorización para mejorar la funcionalidad de este.
- Asignaciones: David Pérez Moreno
- Labels(código, enhancement): Estas labels se corresponden con la modificación del código y la creación de la funcionalidad de la refactorización de este.
- Status(Backlog): La actividad está en la lista backlog, creado su funcionamiento pero todavía no empezada.



Figura A.16: 15º Issue

- Título: Realización de una presentación con transparencias.
- Descripción: Esta tarea está asociada a la realización de una presentación para la exposición del TFG ante el tribunal.
- Asignaciones: David Pérez Moreno
- Labels(documentation): Estas labels se corresponden con la documentación de la exposición del trabajo en diapositivas y sus correspondientes explicaciones.
- Status(Backlog): La actividad está en la lista backlog, creado su funcionamiento pero todavía no empezada.

A.3. Estudio de viabilidad

Viabilidad económica

Introducción

En esta sección realizaré un análisis de viabilidad económica del proyecto desarrollado, **comparando los costes y beneficios si este hubiera sido realizado en un entorno empresarial**. Se tendrán en cuenta los costes materiales y de personal necesarios para el desarrollo del proyecto.

Costes Materiales

A continuación, enumeraré los costes de los materiales utilizados, tanto hardware como software. En este estudio consideraré las versiones destinadas al ámbito profesional y descartaré los materiales gratuitos empleados para simplificar el análisis.

Material	Valor	Amortizado en
Ordenador y periféricos	1.500,00€	5 años
Licencias de software (Python, Stable Baselines, Gymnasium)	0,00€	-
Software de desarrollo (Microsoft Visual Studio)	1.000,00€	3 años
Coste de energía	21,60€	1 año

Tabla A.1: Listado con los materiales, su valor y su tiempo de amortización.

En la Tabla 1 se pueden observar los costes que han supuesto estos materiales al proyecto, teniendo en cuenta que el periodo de desarrollo del proyecto ha sido de aproximadamente seis meses.

El coste de la energía se ha calculado utilizando la siguiente fórmula:

$$\text{Coste de energía} = P \times t \times C$$

Donde:

- P es la potencia consumida en kilovatios (kW).
- t es el tiempo de uso en horas (h).

- C es el coste de la electricidad por kilovatio-hora (€/kWh).

Suponiendo un consumo de potencia de 0.2 kW durante 180 días (6 meses) con un uso promedio de 2 horas por día y un coste de la electricidad de 0.20 €/kWh, el cálculo sería:

$$\text{Coste de energía} = 0,2 \text{ kW} \times 180 \text{ días} \times 2 \text{ horas/día} \times 0,20 \text{ €/kWh} = 21,60 \text{ €}$$

Material	Coste amortizado
Ordenador y periféricos	150,00€
Licencias de software	0,00€
Software de desarrollo	166,67€
Coste de energía	21,60€
TOTAL	338,27€

Tabla A.2: Listado con los materiales y coste amortizado.

Costes de Personal

Estimaré los costes de personal suponiendo que este desarrollo ha sido llevado a cabo por mí, un ingeniero informático recién graduado, trabajando a tiempo parcial (6 horas al día, 5 días a la semana), con un salario bruto anual de 24.000,00€ [3]. El desglose de estos costes para los seis meses de desarrollo se muestra a continuación. Para el cálculo del IRPF, se ha usado la calculadora de sueldo neto disponible en [Cinco Días](#).

Concepto	Coste
Salario anual en bruto	24.000,00€
- IRPF (aproximadamente 15 %)	3.600,00€
- Seguridad Social (6,35 %)	1.524,00€
Salario anual neto	18.876,00€
TOTAL (para 6 meses)	9.438,00€

Tabla A.3: Desglose del coste de personal.

Coste Total del Proyecto

Sumando todos los costes, obtengo que el coste total de la realización del proyecto asciende a:

$$\begin{aligned}\text{Coste total} &= \text{Coste material amortizado} + \text{Coste de personal} = \\ &= 338,27\text{€} + 9,438,00\text{€} = 9,776,27\text{€}\end{aligned}$$

Beneficios y Rentabilidad

En lugar de considerar los beneficios monetarios, este proyecto me ha ofrecido una valiosa oportunidad de **evolución personal y profesional**. Al trabajar en este proyecto, he adquirido y perfeccionado habilidades en áreas como:

- **Desarrollo de Software:** Mejora en la programación y uso de herramientas avanzadas como Python, Stable-Baselines3 y Gymnasium.
- **Aprendizaje por Refuerzo:** Profundización en técnicas de aprendizaje por refuerzo y su aplicación práctica en la optimización de redes ópticas, especialmente redes de acceso.
- **Gestión de Proyectos:** Experiencia en la planificación y ejecución de un proyecto complejo desde su concepción hasta su finalización.
- **Investigación y Análisis:** Capacidades en la investigación y análisis de problemas técnicos y la búsqueda de soluciones innovadoras.

El análisis de viabilidad económica muestra que, aunque el proyecto no generará beneficios monetarios directos, me proporciona un significativo valor añadido a través del desarrollo personal y mi contribución a la Universidad de Valladolid. La inversión en **recursos materiales y humanos se justifica por el potencial educativo y de investigación** que el proyecto ofrece, beneficiando tanto a mi persona como a la institución académica.

Viabilidad legal

Este apartado describe el marco legal que envuelve el proceso de desarrollo del proyecto, su distribución, modificación y utilización. Dado que todo el software utilizado es de código abierto o con licencias académicas, **no se**

han incurrido en costes adicionales por licencias comerciales. He seguido estrictamente las licencias de uso permitidas por las herramientas empleadas.[1]

Las principales consideraciones legales del proyecto son:

- **Licencias de Software:** Todo el software utilizado, incluyendo Python, Stable Baselines y otras bibliotecas, se distribuye bajo licencias de código abierto que permiten su uso, modificación y distribución.[2]
- **Uso Educativo:** El proyecto se desarrolló en un contexto académico y está destinado a ser utilizado con fines educativos y de investigación en la Universidad de Valladolid. No se prevé su comercialización.
- **Cumplimiento de Licencias:** Se han respetado estrictamente las condiciones de las licencias de software utilizadas, asegurando que no se infringen derechos de autor ni se incumplen términos de uso.
- **Protección de Datos:** No se ha manejado ningún dato personal o sensible durante el desarrollo del proyecto, por lo que se ha cumplido con la normativa vigente sobre protección de datos.
- **Propiedad Intelectual:** Los derechos de propiedad intelectual del software desarrollado pertenecerán a la Universidad de Burgos y Universidad de Valladolid y al autor del proyecto.

Librería	Licencia
Gymnasium	MIT License
matplotlib	PSF License
numpy	BSD-3-Clause License
pandas	BSD-3-Clause License
IPython	BSD License
stable_baselines3	MIT License
time	Python Software Foundation License
scipy	BSD License

Tabla A.4: Librerías utilizadas, junto con sus licencias.

Apéndice B

Especificación de Requisitos

B.1. Introducción

La especificación de requisitos es una etapa crucial en el desarrollo de cualquier proyecto, ya que define las necesidades y expectativas que el sistema debe cumplir. En este proyecto, se identifican y documentan los requisitos tanto funcionales como no funcionales necesarios para el correcto funcionamiento del sistema de control de ancho de banda en redes PON mediante algoritmos de aprendizaje por refuerzo.

B.2. Objetivos generales

El objetivo principal del sistema es optimizar la asignación de ancho de banda en redes ópticas pasivas (PON) utilizando un algoritmo de aprendizaje por refuerzo, en particular, Proximal Policy Optimization (PPO). El sistema debe ser capaz de ajustar dinámicamente el tráfico de salida de las Optical Network Units (ONUs) para cumplir con los acuerdos de nivel de servicio (SLA) establecidos, mejorar la eficiencia del uso del ancho de banda y garantizar un rendimiento óptimo de la red.

A continuación, se desglosan los objetivos generales del sistema para proporcionar una comprensión más detallada y facilitar la relación con los requisitos específicos.

- **OG1: Ajuste Dinámico del Tráfico de Salida**

- Descripción: El sistema debe ser capaz de ajustar dinámicamente el tráfico de salida de cada ONU basado en el tráfico de entrada y las condiciones de la red.
- Justificación: Este objetivo es esencial para garantizar que el sistema pueda reaccionar en tiempo real a los cambios en la demanda y las condiciones de la red.

■ **OG2: Cumplimiento de SLA**

- Descripción: El sistema debe garantizar que el tráfico de salida cumpla con los acuerdos de nivel de servicio establecidos para cada usuario.
- Justificación: Cumplir con los SLA es crucial para mantener la calidad del servicio y la satisfacción del cliente.

■ **OG3: Optimización del Uso del Ancho de Banda**

- Descripción: El sistema debe optimizar la asignación del ancho de banda disponible para maximizar la eficiencia de la red.
- Justificación: La optimización del uso del ancho de banda es fundamental para evitar la congestión de la red y asegurar un rendimiento óptimo.

■ **OG4: Monitorización y Reportes**

- Descripción: El sistema debe proporcionar funcionalidades para monitorizar el tráfico de la red y generar reportes detallados sobre el rendimiento y el uso del ancho de banda.
- Justificación: La monitorización y los reportes son esenciales para la gestión y el mantenimiento de la red, así como para la toma de decisiones informadas.

B.3. Catálogo de requisitos

Requisitos Funcionales

Los requisitos funcionales describen las funcionalidades que el sistema debe proporcionar para cumplir con sus objetivos. Estos requisitos son específicos y medibles, y son esenciales para el diseño y la implementación del sistema.

■ RF1: Ajuste Dinámico del Tráfico de Salida

- Relacionado con: **OG1**
- Descripción: El sistema debe ser capaz de ajustar dinámicamente el tráfico de salida de cada ONU basado en el tráfico de entrada y las condiciones de la red.

■ RF2: Cumplimiento de SLA

- Relacionado con: **OG2**
- Descripción: El sistema debe garantizar que el tráfico de salida cumpla con los acuerdos de nivel de servicio establecidos para cada usuario.

■ RF3: Optimización del Uso del Ancho de Banda

- Relacionado con: **OG3**
- Descripción: El sistema debe optimizar la asignación del ancho de banda disponible para maximizar la eficiencia de la red.

■ RF4: Monitorización y Reportes

- Relacionado con: **OG4**
- Descripción: El sistema debe proporcionar funcionalidades para monitorizar el tráfico de la red y generar reportes detallados sobre el rendimiento y el uso del ancho de banda.

Requisitos No Funcionales

Los requisitos no funcionales especifican los criterios que pueden ser utilizados para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos. Estos requisitos son esenciales para asegurar la usabilidad, eficiencia, fiabilidad y mantenibilidad del sistema.

■ RNF1: Rendimiento

- Descripción: El sistema debe ser capaz de procesar y ajustar el tráfico de red en tiempo real, sin causar demoras significativas.

■ RNF2: Escalabilidad

- Descripción: El sistema debe ser escalable para manejar un número creciente de ONUs y volúmenes de tráfico sin degradar su rendimiento.
- **RNF3: Seguridad**
 - Descripción: El sistema debe implementar medidas de seguridad para proteger los datos y las comunicaciones de la red contra accesos no autorizados y ataques.
 - **RNF4: Usabilidad**
 - Descripción: El sistema debe ser fácil de usar y administrar, proporcionando una interfaz intuitiva para los operadores de red.
 - **RNF5: Mantenibilidad**
 - Descripción: El sistema debe ser diseñado de manera modular para facilitar su mantenimiento, actualización y expansión futura.

Especificación de requisitos

A continuación crearemos el diagrama de casos de uso correspondientes al programa.

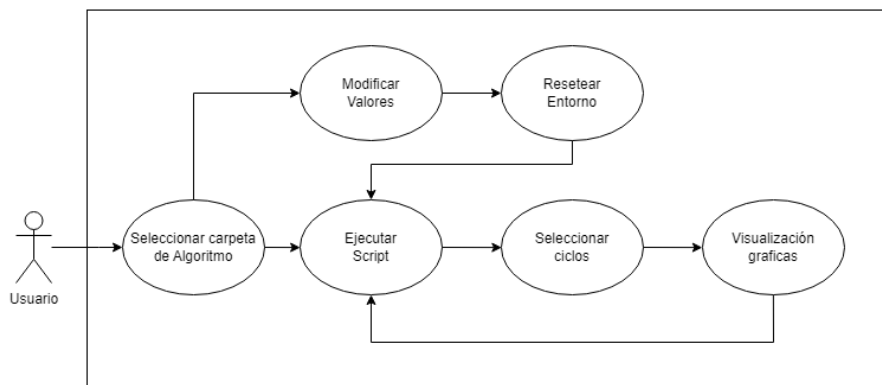


Figura B.1: Diagrama de casos de uso

CU-1	Seleccionar Carpeta de Algoritmo
Versión	1.2
Autor	David Pérez Moreno
Requisitos asociados	RF-01
Descripción	El usuario elige una de las carpetas correspondientes a los diferentes algoritmos.
Precondición	El usuario debe de haber descargado cada una de las carpetas correspondientes.
Acciones	<ol style="list-style-type: none">1. El usuario navega a la sección de selección de carpetas.2. El usuario selecciona una de las carpetas disponibles.
Postcondición	El código de la carpeta seleccionada queda listo para su ejecución.
Excepciones	Si no hay carpetas disponibles, el sistema muestra un mensaje de error.
Importancia	Alta

Tabla B.1: CU-1 Seleccionar Carpeta de Algoritmo.

CU-2	Ejecutar Script del Escenario
Versión	1.2
Autor	David Pérez Moreno
Requisitos asociados	RF-02
Descripción	Una vez seleccionada la carpeta, se ejecuta el script del escenario elegido.
Precondición	La carpeta del algoritmo debe estar seleccionada.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona la opción de ejecutar el script.2. El sistema carga el entorno y ejecuta el script correspondiente.
Postcondición	El script del escenario se ejecuta correctamente.
Excepciones	Si el script no se puede ejecutar, se muestra un mensaje de error.
Importancia	Alta

Tabla B.2: CU-2 Ejecutar Script del Escenario.

CU-3	Seleccionar Número de Ciclos
Versión	1.0
Autor	David Perez Moreno
Requisitos asociados	RF-03
Descripción	El usuario introduce el número de ciclos a ejecutar.
Precondición	El script del escenario debe estar ejecutándose.
Acciones	<ol style="list-style-type: none">1. El usuario accede al menú de configuración de ciclos.2. El usuario introduce el número deseado de ciclos.
Postcondición	El sistema registra el número de ciclos a ejecutar y ejecutará el programa respecto a estos ciclos seleccionados.
Excepciones	Si el número de ciclos no es válido, el sistema muestra un mensaje de error.
Importancia	Media

Tabla B.3: CU-3 Seleccionar Número de Ciclos.

CU-4	Visualizar Gráficas
Versión	1.1
Autor	David Pérez Moreno
Requisitos asociados	RF-04
Descripción	El usuario visualiza las gráficas generadas por el programa.
Precondición	El programa debe haber completado la ejecución de los ciclos.
Acciones	<ol style="list-style-type: none"> 1. El sistema muestra las gráficas generadas.
Postcondición	Las gráficas se muestran correctamente al usuario.
Excepciones	Si no hay gráficas disponibles, se muestra un mensaje de error.
Importancia	Alta

Tabla B.4: CU-4 Visualizar Gráficas.

CU-5	Modificar Valores
Versión	1.5
Autor	David Pérez Moreno
Requisitos asociados	RF-05
Descripción	El usuario modifica los valores, como por ejemplo la velocidad garantizada.
Precondición	El programa debe estar detenido.
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al menú de configuración. 2. El usuario modifica los valores deseados.
Postcondición	Los valores modificados se guardan correctamente.
Excepciones	Si el valor ingresado no es válido, se muestra un mensaje de error.
Importancia	Media

Tabla B.5: CU-5 Modificar Valores.

CU-6	Resetear Entorno
Versión	1.0
Autor	David Pérez Moreno
Requisitos asociados	RF-06
Descripción	Después de modificar la variable, el usuario resetea el entorno para guardar los nuevos valores en el entorno.
Precondición	Los valores deben haber sido modificados.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona la opción en el Microsoft Visual Studio Code de resetear el script para resetear el entorno.2. El sistema resetea el entorno a su estado inicial con los cambios realizados.
Postcondición	El entorno se resetea correctamente y deberíamos de ejecutar el entorno reseteado.
Excepciones	El entorno no se puede resetear si aun no se ha ejecutado.
Importancia	Media

Tabla B.6: CU-6 Resetear Entorno.

Apéndice C

Especificación de diseño

El diseño del sistema es una etapa crucial en el desarrollo de cualquier aplicación, ya que define cómo se **estructurarán y funcionarán los componentes del sistema**. En este proyecto, se han considerado varios aspectos clave del diseño, incluyendo el diseño de datos, el diseño procedimental y el diseño arquitectónico.

C.1. Diseño de Datos

Los datos son fundamentales para el funcionamiento de este sistema. El diseño de datos se enfoca en cómo se almacenan, gestionan y acceden a los datos necesarios para la operación del sistema. En nuestro caso, utilizamos estructuras de datos eficientes para manejar las variables relacionadas con el tráfico de entrada, salida y pendiente, así como las condiciones de la red.

C.2. Diseño Procedimental

El diseño procedimental aborda los procesos más importantes del sistema, describiendo cómo interactúan los distintos componentes para llevar a cabo las tareas requeridas. A continuación, se presenta un diagrama de secuencia que ilustra la interacción de los componentes del sistema para la ejecución de un ciclo típico del algoritmo PPO:

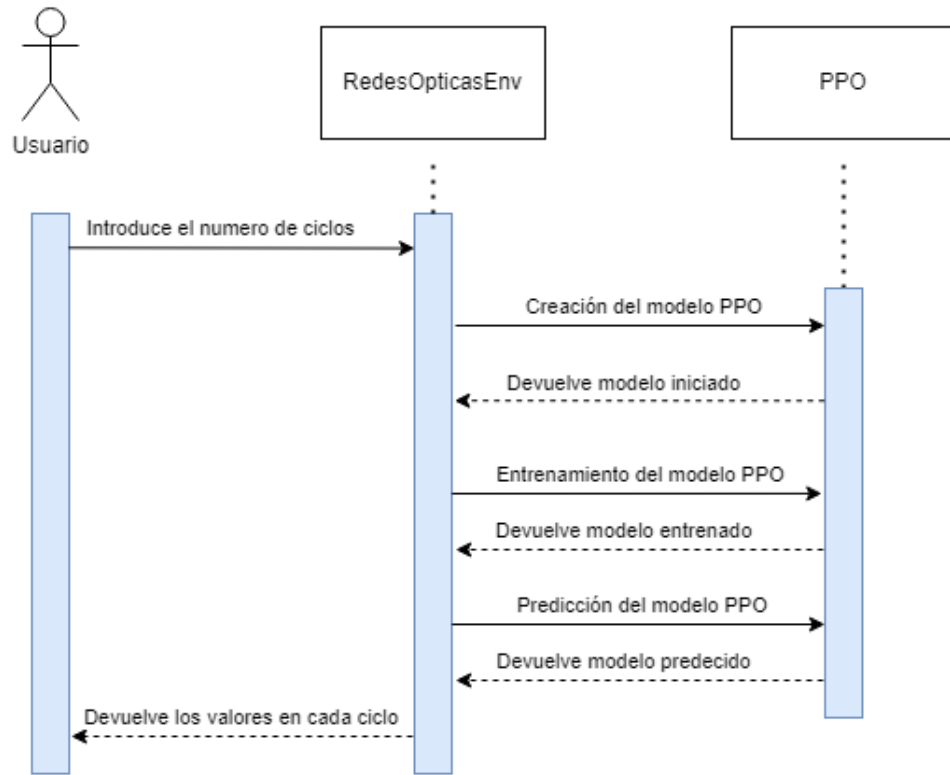


Figura C.1: Diagrama de secuencia

C.3. Diseño Arquitectónico

El diseño arquitectónico describe la estructura del sistema y la organización de sus componentes. Para este proyecto, se ha seguido un patrón arquitectónico modular que facilita la mantenibilidad y la escalabilidad del sistema. La arquitectura del sistema se puede dividir en los siguientes componentes principales:

- **Modelo:** Maneja los datos del sistema, incluyendo el almacenamiento y la gestión de las variables de tráfico.
- **Vista:** Proporciona la interfaz de usuario para interactuar con el sistema, permitiendo la selección de algoritmos, la visualización de gráficas y la modificación de valores.

- **Controlador:** Actúa como intermediario entre el modelo y la vista, gestionando la lógica de negocio y las interacciones del usuario.

El siguiente diagrama de clases muestra la relación entre los componentes principales del sistema:

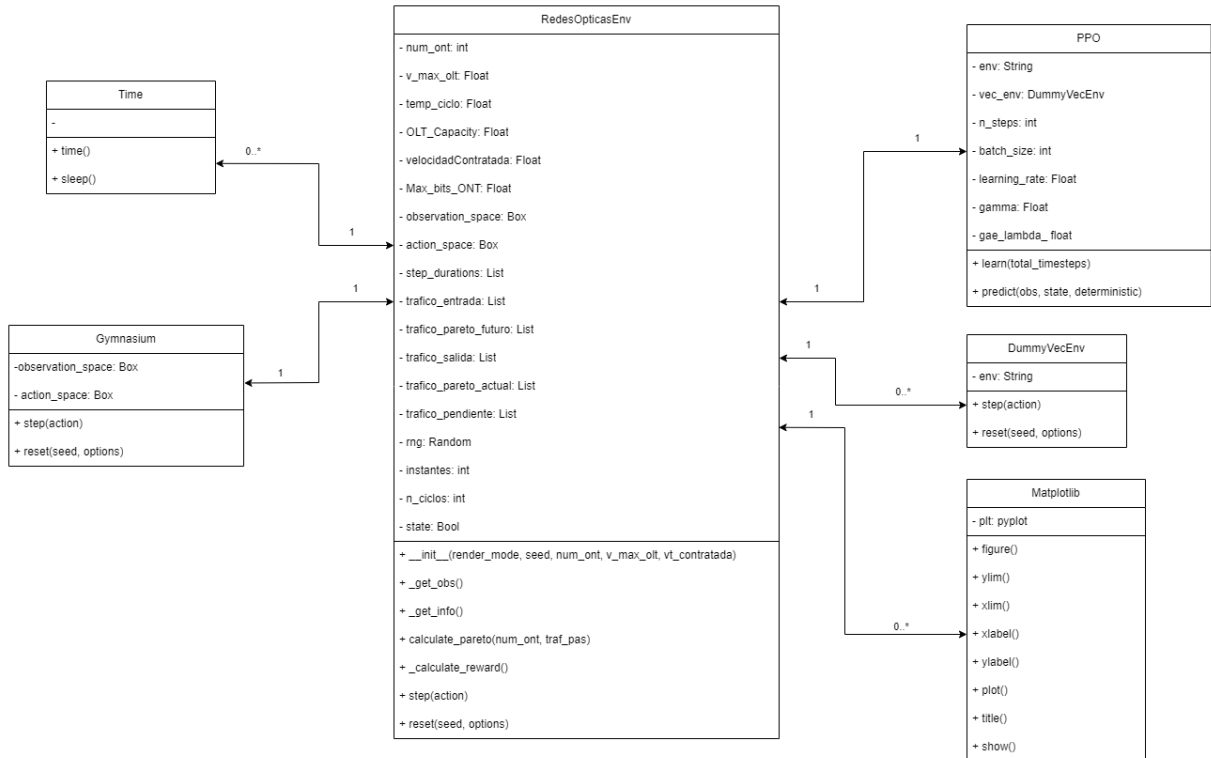


Figura C.2: Diagrama de clases UML

En el diagrama se destacan las clases principales y sus interacciones, proporcionando una visión clara de cómo se estructura el sistema y cómo se gestionan las diferentes funcionalidades.

A continuación, se describe la relación entre las diferentes clases del sistema:

- **RedesOpticasEnv:**

- Relaciones:

- $1 \longleftrightarrow 1$ con **PPO**
- $1 \longleftrightarrow 1$ con **Gymnasium**: Utiliza las clases `spaces.Box` para definir el espacio de observación y acción.
- $1 \longleftrightarrow 1$ con **Time**: Utiliza métodos para medir y controlar el tiempo.
- $1 \longleftrightarrow 0..*$ con **Matplotlib**: Utiliza para graficar los resultados.

■ **PPO:**

- Relaciones:
 - $1 \longleftrightarrow 1$ con **RedesOpticasEnv**: Utiliza el entorno de simulación `RedesOpticasEnv` para entrenar el modelo.
 - $1 \longleftrightarrow 1$ con **DummyVecEnv**: Utiliza para crear un vector de entornos paralelos.

■ **DummyVecEnv:**

- Relaciones:
 - $1 \longleftrightarrow 1$ con **PPO**

■ **Matplotlib:**

- Relaciones:
 - $0..* \longleftrightarrow 1$ con **RedesOpticasEnv**

■ **Time:**

- Relaciones:
 - $0..* \longleftrightarrow 1$ con **RedesOpticasEnv**

■ **Gymnasium:**

- Relaciones:
 - $1 \longleftrightarrow 1$ con **RedesOpticasEnv**

El diseño del sistema se ha realizado teniendo en cuenta la **eficiencia, la escalabilidad y la facilidad de mantenimiento**. La estructura modular y los componentes claramente definidos permiten un desarrollo coherente y facilitan futuras expansiones o modificaciones del sistema.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este apéndice está dedicado a documentar todos los aspectos relacionados con la programación de la aplicación. Se describirá cómo se distribuyen los directorios, cómo descargar y compilar el repositorio, el proceso de pruebas, entre otros detalles. El objetivo es proporcionar una guía completa para futuros desarrolladores que puedan modificar correctamente los distintos componentes de la aplicación. Se usan 3 carpetas de directorios de escenarios para dividir cada uno de los escenarios, el cual cada una de las carpetas tiene el código diferente para cada uno de los escenarios ya que se comportan de manera diferente, para la elección de cada escenario se elegirá el correspondiente.

D.2. Estructura de directorios

La estructura de directorios es la usada en el repositorio de GitHub del proyecto.

- **./:** **Directorio Raíz** del repositorio, en este se encuentran todos los directorios del repositorio, ficheros auxiliares y el README.md.
- **./Documentacion:** En este repositorio se encuentra la memoria y los anexos del proyecto.

- **./github/workflows:** Directorio auxiliar que contiene ficheros auxiliares para el análisis con SonarCloud.
- **./Codigo Turbinas eólicas:** Directorio que contiene el primer código DRL realizado con un entorno de turbinas eólicas.
- **./Codigo Turbinas eólicas/custom__env:** En este directorio dentro del directorio de turbinas eólicas se encuentra la carpeta con el entorno de las turbinas eólicas.
- **./Redes ópticas:** Directorio que contiene la primera versión de las redes ópticas. A pesar de ser descartado, me pareció interesante mantenerlo en el directorio para que se pueda ver como se podría implementar este sistema desde un punto de vista ficticio.
- **./Redes ópticas/custom__env:** En este directorio dentro del directorio de la primera versión de redes ópticas se encuentra la carpeta con el entorno de la primera versión de redes ópticas.
- **./Redes ópticas v2.0:** Este es el directorio donde nos encontraremos con los 3 escenarios creados de la versión final del proyecto.
- **./Redes ópticas v2.0/Escenario 1:** Esta carpeta se corresponde con la versión final del proyecto del primer escenario.
- **./Redes ópticas v2.0/Escenario 1/custom__env:** En este directorio dentro del directorio del primer escenario de la versión final del programa se encuentra la carpeta con el entorno del primer escenario de la versión final del programa, cuyo funcionamiento es el más básico de los 3 escenarios.
- **./Redes ópticas v2.0/Escenario 2:** Esta carpeta se corresponde con la versión final del proyecto del segundo escenario.
- **./Redes ópticas v2.0/Escenario 2/custom__env:** En este directorio dentro del directorio del segundo escenario de la versión final del programa se encuentra la carpeta con el entorno del segundo escenario de la versión final del programa, cuyo funcionamiento se corresponde a la generación de varios grupos con sus correspondientes valores de carga.
- **./Redes ópticas v2.0/Escenario 3:** Esta carpeta se corresponde con la versión final del proyecto del tercer escenario.

- **./Redes ópticas v2.0/Escenario 3/custom__env:** En este directorio dentro del directorio del tercer escenario de la versión final del programa se encuentra la carpeta con el entorno del tercer escenario de la versión final del programa, cuyo funcionamiento se corresponde al cambio de la velocidad garantizada a otro valor en medio del funcionamiento del programa.

D.3. Manual del programador

En esta sección, se explicará cómo un programador puede descargar el proyecto y desarrollar su propia versión.

Descarga de los archivos

El código está disponible en un repositorio de Github, para obtener cada archivo se deberá de dar situados en la raíz del repositorio al botón de Code y posteriormente descargar Zip, en la imagen represento esto último.

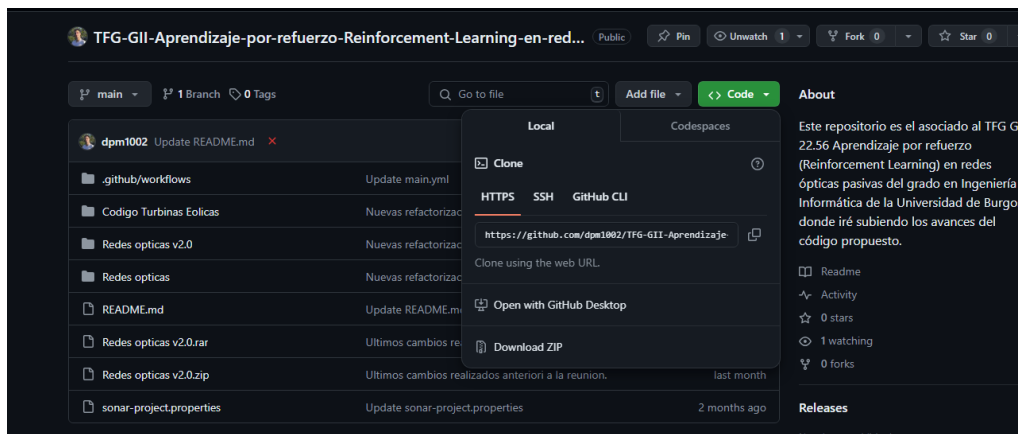


Figura D.1: Descarga del repositorio

Importación en el entorno de desarrollo

El entorno de desarrollo usado y recomendado para los usuarios es el de Microsoft Visual Studio Code, desarrollado por Microsoft y nos proporciona extensiones muy útiles para el trabajo con otros lenguajes. En nuestro caso combina Jupyter Notebook y Python a la perfección. También nos ofrece una extensión para poder subir y bajar fácilmente los archivos desde Github.

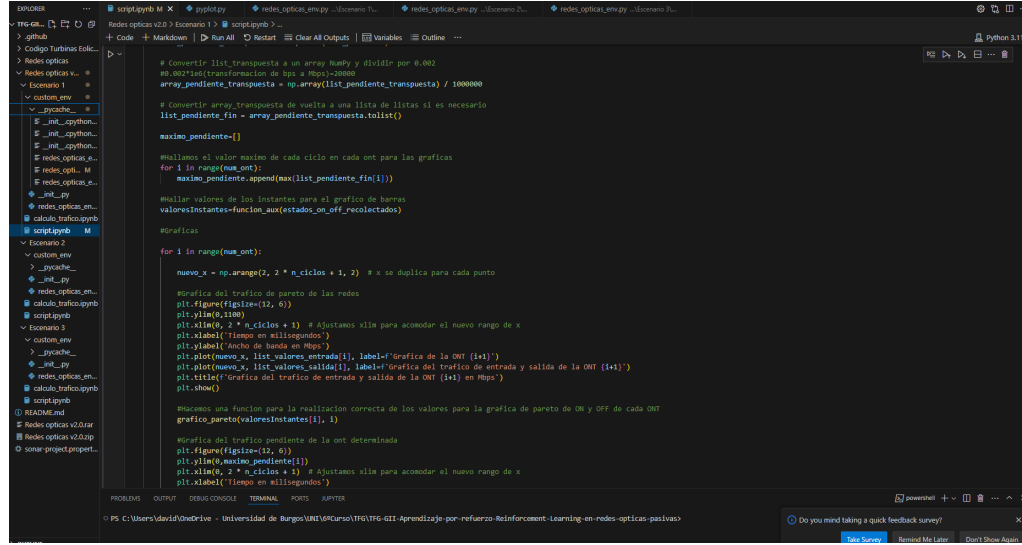


Figura D.2: Proyecto abierto en Visual Studio Code

Una vez importado el programa se quedaría de la siguiente forma abriendo las carpetas de **./Redes ópticas v2.0** donde se encuentran los 3 escenarios.

D.4. Compilación, instalación y ejecución del proyecto

Como hemos mencionado antes, usaremos el entorno de desarrollo de Microsoft Visual Studio Code ya que nos permite trabajar con varios lenguajes siempre y cuando tengamos las dependencias instaladas.

Para ello necesitaremos tener instaladas las siguientes dependencias para que el programa funcione sin problemas:

También deberemos de tener la extensión de Jupyter Notebook en el Visual Studio Code para la ejecución de los entornos, para ello desde la parte de extensiones que nos ofrece este entorno de desarrollo instalamos esta extensión.

Librerías	Versión
Python	3.11.9
Gymnasium	0.27.1
stable_baselines3	1.6.0
Matplotlib	3.8.4
Numpy	1.26.4
Pandas	2.0.3
Scipy	1.11.1

Tabla D.1: Librerías utilizadas y sus versiones

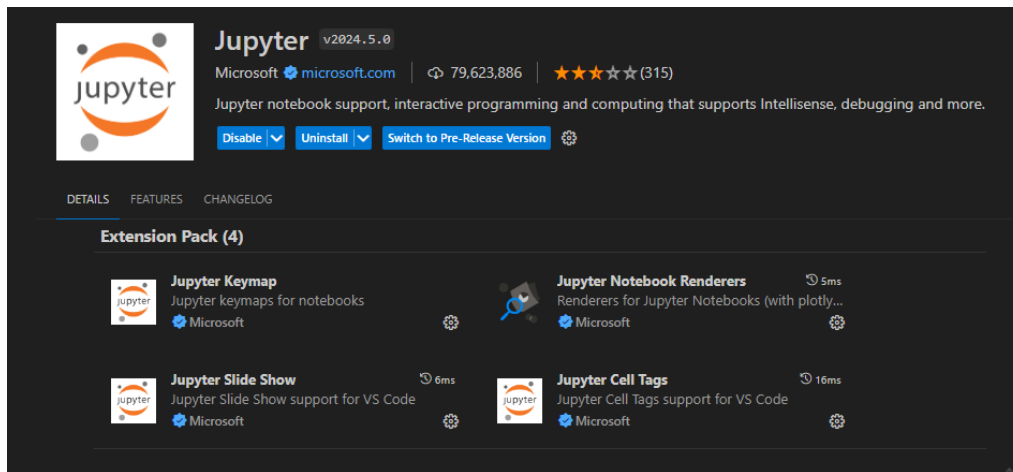
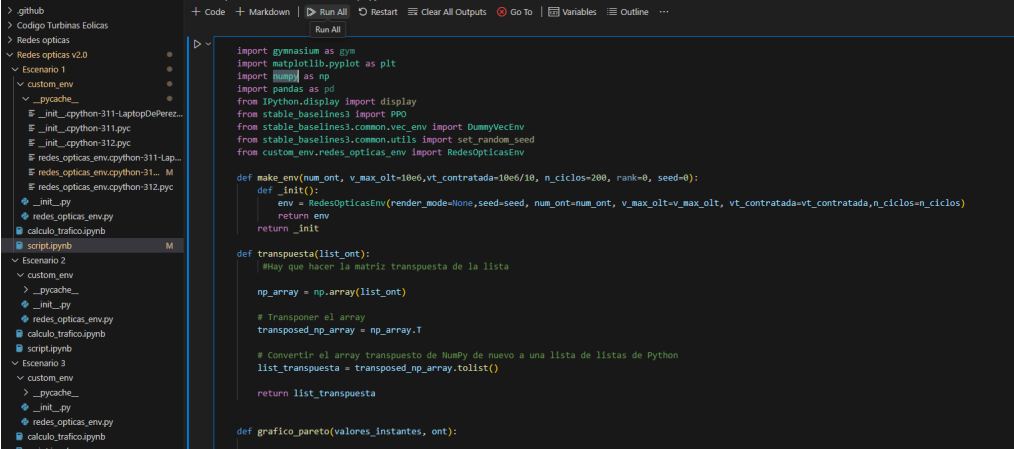


Figura D.3: Extensión de Jupyter Notebook en Visual Studio Code

Una vez instalado todo, ya solo nos quedaría ejecutar el escenario que deseemos, para ello escogemos el escenario que deseemos, en nuestro caso el 1º escenario y nos situamos desde el **script.ipynb**; una vez aquí deberemos de darle al botón de **Run All** para ejecutar el programa, luego nos pedirá por teclado el número de ciclos que deseemos darle y ya automáticamente el programa realizará a desarrollar todo el entorno PPO y finalmente mostrar las gráficas.



```

import gymnasium as gym
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from IPython.display import display
from stable_baselines3 import PPO
from stable_baselines3.common.vec_env import DummyVecEnv
from stable_baselines3.common.utils import set_random_seed
from custom_env.redes_opticas_env import RedesOpticasEnv

def make_env(num_ont, v_max_olt=10e6, vt_contratada=10e6/10, n_ciclos=200, rank=0, seed=0):
    def _init():
        env = RedesOpticasEnv(render_mode=None, seed=seed, num_ont=num_ont, v_max_olt=v_max_olt, vt_contratada=vt_contratada, n_ciclos=n_ciclos)
        return env
    return _init

def transpuesta(list_ont):
    """
    Hay que hacer la matriz transpuesta de la lista
    """
    np_array = np.array(list_ont)

    # Transponer el array
    transposed_np_array = np_array.T

    # Convertir el array transpuesto de NumPy de nuevo a una lista de listas de Python
    list_transpuesta = transposed_np_array.tolist()

    return list_transpuesta

def grafico_pareto(valores_instantes, ont):

```

Figura D.4: Script.ipynb 1º Escenario

D.5. Pruebas del sistema

En esta sección se documentan los casos de prueba realizados para garantizar el correcto funcionamiento del sistema desarrollado. Los casos de prueba se diseñaron para cubrir los diferentes aspectos del sistema, asegurando que se cumplan los requisitos funcionales y no funcionales definidos anteriormente.

1º Caso de prueba

Se ha probado a introducir un **número de ciclos nulo**, para ver si el sistema lo detectaba, cuya respuesta ha sido un mensaje de error ya que necesita esta variable un número obligatoriamente.

2º Caso de prueba

De la misma manera, al introducir un **número de ciclos negativo**, el programa no debería de actuar con estos valores, ya que está programado para tener ciclos positivos. Se muestra también un mensaje de error.

3º Caso de prueba

También se ha probado a introducir un **número mayor a 2.147.483.647**, esto es ya que es el valor límite para los valores int. La variable intenta ejecutar estos valores y se queda en bucle infinito.

4º Caso de prueba

También, Microsoft Visual Studio Code nos permite interrumpir el proceso a mitad de su ejecución. Al dar al botón de **Interrupt** a la mitad de la ejecución el programa detiene su funcionamiento y finalizando con un mensaje de error.

5º Caso de prueba

Al establecer algunas de las **variables clave.^a valor 0 o nulo**, el programa no soporta estos valores, ya que está programado para que sean al menos de un valor razonable. Entre estas variables se encuentran:

- **num__ont:** si dejamos a 0 este valor no habría ninguna unidad óptica.
- **T:** si dejamos a 0 este valor no sabríamos el tiempo de cada ciclo.
- **v__max__olt:** si dejamos a 0 este valor no tendríamos ninguna velocidad de transmisión.
- **vt__contratada:** si dejamos a 0 este valor no sabemos en ningún momento la velocidad garantizada máxima a la que retransmitirían las onts.
- **num__envs:** si dejamos a 0 este valor no tendríamos ningún entorno definido.
- **num__test__episodes:** si dejamos a 0 este valor no tendríamos ningún episodio a ejecutar.

Los casos de prueba documentados en esta sección **han sido probados a mano y permiten asegurar el correcto funcionamiento del sistema desarrollado**. Cada caso de prueba ha sido diseñado para verificar diferentes aspectos del sistema, garantizando así que se cumplan todos los requisitos funcionales y no funcionales.

Apéndice *E*

Documentación de usuario

E.1. Introducción

En este apartado se proporcionará una **guía detallada para los usuarios del sistema desarrollado**. Se explicarán los requisitos necesarios, el proceso de instalación y el uso de la aplicación para que los usuarios puedan aprovechar al máximo las funcionalidades ofrecidas.

E.2. Requisitos de usuarios

Para poder utilizar el sistema, los usuarios deben cumplir con los siguientes requisitos:

- **Conocimientos básicos de informática:** Es necesario que el usuario tenga conocimientos básicos de informática, incluyendo el uso de aplicaciones y la instalación de software.
- **Acceso a Internet:** Se requiere una conexión a Internet para descargar las dependencias y actualizaciones del sistema.
- **Sistema Operativo:** El sistema ha sido probado en las siguientes plataformas:
 - Windows 10 o superior
- **Python 3.11.9:** El sistema está desarrollado en Python 3.11.9, por lo que se requiere esta versión o una superior.

E.3. Instalación

A continuación, se detallan los pasos para instalar el sistema en un entorno local:

Importación del proyecto en Visual Studio Code

- Descargar e instalar Visual Studio Code desde la página oficial (<https://code.visualstudio.com/>).
- Abrir Visual Studio Code.
- Seleccionar **File > Open Folder...** y navegar hasta el directorio del proyecto descargado. Seleccionar la carpeta del proyecto y hacer clic en **Open**.
- Si se le solicita instalar las extensiones recomendadas para Python, aceptar e instalar las extensiones.
- Abrir la barra de comandos presionando **Ctrl+Shift+P** y escribir **Python: Select Interpreter**. Seleccionar la versión de Python 3.11.9 instalada previamente.
- Abrir el archivo principal del proyecto (por ejemplo, `script.ipynb`).
- Ejecutar el programa utilizando la opción **Run All** disponible en el menú superior de Visual Studio Code.

Instalación de Python

- Descargar Python 3.11.9 desde la página oficial (<https://www.python.org/downloads/>).
- Seguir las instrucciones de instalación específicas para el sistema operativo.

Instalación de la extensión Jupyter Notebook en Visual Studio Code

Para poder ejecutar y visualizar Jupyter Notebooks dentro de Visual Studio Code, es necesario instalar la extensión correspondiente. A continuación, se describen los pasos para hacerlo:

1. Descargar e instalar Visual Studio Code desde la página oficial (<https://code.visualstudio.com/>).
2. Abrir Visual Studio Code.
3. Ir a la opción **View > Extensions** (o presionar **Ctrl+Shift+X**).
4. En la barra de búsqueda de extensiones, escribir **Jupyter** y buscar la extensión llamada **Jupyter** proporcionada por Microsoft.
5. Hacer clic en el botón **Install** para instalar la extensión.
6. Una vez instalada la extensión, ir a **File > Open Folder...** y seleccionar la carpeta del proyecto donde se encuentran los Jupyter Notebooks.
7. Abrir el archivo `script.ipynb` haciendo doble clic sobre él. El archivo se abrirá en el entorno de Jupyter Notebook dentro de Visual Studio Code.
8. Seleccionar el intérprete de Python adecuado presionando **Ctrl+Shift+P**, escribiendo **Python: Select Interpreter** y eligiendo la versión de Python 3.11.9 instalada previamente.
9. Ejecutar las celdas del notebook utilizando los botones de control en la parte superior del editor de Jupyter Notebook.

Instalación de librerías específicas

En caso de que no se instalen automáticamente las dependencias, se puede hacer manualmente siguiendo estos pasos:

- **Gymnasium:**

```
pip install gymnasium
```

- **stable_baselines3:**

```
pip install stable-baselines3
```

- **matplotlib:**

```
pip install matplotlib
```

- **numpy:**

```
pip install numpy
```

- **pandas:**

```
pip install pandas
```

- **scipy:**

```
pip install scipy
```

E.4. Manual del usuario

A continuación, se describe cómo utilizar la aplicación paso a paso:

Inicio de la aplicación

1. Abrir Visual Studio Code.
2. Seleccionar **File > Open Folder...** y abrir la carpeta del escenario que deseemos.
3. Abrir el archivo principal del proyecto (`script.ipynb`).
4. Ejecutar el programa utilizando la opción **Run All** disponible en el menú superior de Visual Studio Code.

Introducir el número de ciclos

1. Después de seleccionar el script del escenario que deseemos, se pedirá al usuario que introduzca el número de ciclos a ejecutar.
2. Escribir el número de ciclos deseado y hacer clic en **Aceptar**.

Visualizar gráficas

1. Al finalizar la ejecución del algoritmo, las gráficas generadas se mostrarán automáticamente en la interfaz.
2. El usuario puede analizar las gráficas para obtener información sobre el rendimiento del algoritmo y el tráfico de red.

Este manual proporciona una guía completa para la instalación y uso del sistema desarrollado. Siguiendo los pasos descritos, **los usuarios podrán instalar el sistema en su entorno local y utilizar todas las funcionalidades ofrecidas para gestionar el tráfico de red en redes ópticas pasivas (PON, Passive Optical Network) mediante algoritmos de aprendizaje por refuerzo.**

Apéndice *F*

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluye una **reflexión personal sobre los aspectos de sostenibilidad** abordados en este trabajo. A través del desarrollo del proyecto, se han aplicado y adquirido **competencias relacionadas con la sostenibilidad**, reflejando el compromiso con un **desarrollo sostenible en el ámbito de la ingeniería informática**.^[4]

Más información sobre las competencias de sostenibilidad se puede encontrar en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sostenibilidad_Crue2012.pdf.

F.2. Reflexión sobre la Sostenibilidad

Durante el desarrollo de este Trabajo de Fin de Grado (TFG), he tenido la oportunidad de aplicar **varias competencias de sostenibilidad** que considero fundamentales para mi formación como ingeniero informático. A continuación, detallo algunas de estas competencias y cómo se han integrado en el proyecto.^[6]

Conciencia y Acción Medioambiental

La **ingeniería informática tiene un impacto significativo en el medio ambiente**, principalmente a través del consumo de energía y la

gestión de residuos electrónicos. En este TFG, he implementado **algoritmos de aprendizaje por refuerzo profundo (DRL) para optimizar el uso de ancho de banda en redes ópticas pasivas (PON)**, lo que puede contribuir a una **utilización más eficiente de los recursos de red** y, en consecuencia, a una **reducción del consumo energético**.

El uso de DRL es particularmente adecuado para esta tarea debido a su capacidad para aprender y adaptarse a entornos complejos y dinámicos. A través de la iteración continua y el ajuste de políticas basadas en la retroalimentación del entorno, el DRL puede identificar patrones de tráfico y gestionar el ancho de banda de manera más eficiente que los métodos tradicionales. Esto se traduce en una asignación más precisa de recursos, evitando el desperdicio de energía en la transmisión de datos innecesarios o en tiempos de inactividad.

Esta optimización es crucial para **minimizar la huella ecológica de las infraestructuras de telecomunicaciones**, ya que una mejor gestión del ancho de banda reduce la necesidad de expandir la infraestructura física, que a su vez disminuye la producción de equipos y los residuos electrónicos. En resumen, al mejorar la eficiencia operativa de las redes ópticas pasivas mediante DRL, se logra una **reducción significativa del consumo energético y del impacto ambiental asociado a las telecomunicaciones**.

Eficiencia y Optimización de Recursos

El uso de **técnicas de optimización** en el proyecto no solo busca mejorar el rendimiento de la red, sino también **maximizar la eficiencia en el uso de los recursos disponibles**. El algoritmo **PPO (Proximal Policy Optimization)** empleado en este trabajo permite gestionar el tráfico de datos de manera más eficiente, asegurando que se utilice solo el ancho de banda necesario y **evitando sobrecargas y desperdicios de capacidad**. Esta práctica es un claro ejemplo de cómo la ingeniería puede contribuir a la **sostenibilidad económica y ambiental**.^[5]

Responsabilidad Social y Ética Profesional

A lo largo del desarrollo del proyecto, he reflexionado sobre la **responsabilidad social que implica trabajar en el campo de la informática**. La implementación de soluciones eficientes y sostenibles no solo beneficia a las empresas y a los usuarios, sino que también tiene un **impacto positivo en la sociedad en general**. **Promover el acceso equitativo a**

tecnologías de comunicación eficientes es una forma de **contribuir al bienestar social y al desarrollo sostenible**.

Innovación y Mejora Continua

La sostenibilidad en ingeniería informática también está relacionada con la **capacidad de innovar y mejorar continuamente**. Durante el TFG, he explorado **nuevas metodologías y herramientas para optimizar el tráfico de redes PON**. Esta **búsqueda constante de mejoras** refleja un **compromiso con la sostenibilidad**, asegurando que las soluciones propuestas no solo sean efectivas hoy, sino que puedan adaptarse y mejorar en el futuro.

Educación y Sensibilización

Finalmente, considero que uno de los aspectos más importantes de la sostenibilidad es la **educación y la sensibilización**. A través de este proyecto, espero **inspirar a otros estudiantes y profesionales a considerar la sostenibilidad en sus propios trabajos**. Compartir conocimientos y experiencias sobre prácticas sostenibles es esencial para **fomentar una cultura de responsabilidad ambiental y social en la ingeniería**.

F.3. Conclusión

En conclusión, el desarrollo de este TFG me ha permitido **aplicar y fortalecer varias competencias de sostenibilidad**. Desde la **optimización de recursos** hasta la **responsabilidad social y la innovación**, estos principios han guiado mi trabajo y reflejan mi compromiso con un desarrollo sostenible. Estoy convencido de que la **integración de la sostenibilidad en la ingeniería informática** es fundamental para **enfrentar los desafíos del futuro y contribuir a un mundo más equitativo y sostenible**.

Bibliografía

- [1] Estudio de viabilidad de software. <https://www.ticportal.es/glosario-tic/estudio-viabilidad-software>.
- [2] Licencia de software. https://es.wikipedia.org/wiki/Licencia_de_software.
- [3] Sueldos de programador junior. https://www.glassdoor.es/Sueldos/programador-junior-sueldos-SRCH_K00,18.htm.
- [4] Wikibooks contributors. Sistemas de información/material informático sostenible. https://es.wikibooks.org/wiki/Sistemas_de_informacion/Material_informático_sostenible.
- [5] Inforges. 5 claves para la optimización de procesos y recursos en la empresa. <https://inforges.es/blog/5-claves-optimizacion-procesos-recursos-empresa/>.
- [6] Author or Organization Name. Sustainable it: Benefits and strategies to implement. <https://openit.com/es/sustainable-it-benefits-and-strategies-to-implement/>, 2022.